IBERGRID

2013

7th IBERIAN GRID INFRASTRUCTURE CONFERENCE PROCEEDINGS

Madrid , Spain September 19-20, 2013

EDITORS

Ignacio Blanquer

Isabel Campos

Gonçalo Borges

Jorge Gomes

http://www.ibergrid.eu/2013

ibergrid13@upv.es

IBERGRID

7th IBERIAN Grid Infrastructure Conference Proceedings

Madrid, Spain

September 19th–20th

2013

Editorial Universitat Politècnica de València

The contents of this publication have been subject to revision by Scientific Committee

First edition, 2013

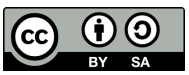
Editors

Ignacio Blanquer
Universitat Politècnica de València
Isabel Campos
Instituto de Física de Cantabria
Gonçalo Borges
Laboratório de Instrumentação e Física Expermiental de Particulas
Jorge Gomes
Laboratório de Instrumentação e Física Expermiental de Particulas

Distribution: Tel. +34 – 963 877 012 / http://www.lalibreria.upv.es / Ref.: 2241

ISBN 978-84-9048-110-3 (Versión impresa)

Print on demand



Contents

Session IV: Scientific Clouds

Session IV: Scientific Applications in Distributed Infrastructures

Message from the conference chairs

2013 is becoming a challenge year for the research in Europe in general and both in Spain and Portugal in particular. Reduction on investment is making scientist to become more and more efficient on the use of resources and international cooperation will become a key target. These major aims are clearly reflected on the activity of IBERGRID.

The usage of the resources of the IBERGRID NGI has increased in a 18% from last year reaching almost 180 million KSI2K hours from July 2012 until June 2013. The Large Hadron Collider (LHC) experiments continue representing the major share of these usage, but multidisciplinary Virtual Organizations keep the same share as previous years, being IBERGRID one of top NGIs in EGI dedicating a major share to non-LHC scientists. On the other side, the usage of the EGI infrastructure by Spanish users have increased up to 161 million KSI2K hours from July 2012 until June 2013, 20% more than in the same period of the previous year. The participation of Spanish institutions in European projects in Research Infrastructures has increased up to 12 European projects, some of them showing their result on this book.

This year's edition specially addresses cloud computing and expertise from user communities. Challenges such as the federation of virtual infrastructures and the easy deployment of services and virtual appliances constitute a major trend in today's research. The Spanish participation in the EGI's Virtualized Clouds Task Force has been important and relevant.

Regarding the use cases, experiences in the LHC, energy, biodiversity, engineering and security will be presented and discussed, surely showing many common points for exchanging approaches and solutions. ICT advances in distributed systems in terms of data, fault-tolerance and interoperability will also be presented.

Finally, it is important to mention that IBERGIRD organizes the 2013 Technical Forum of EGI in Madrid, Spain. This event will bring over 400 of participants with interest on distributed infrastructures for research, including top members of the European Commission. This seventh IBERGRID conference has been co-located with this major event. We believe that attendees to IBERGRID will strongly benefit from the wider scope of the technical forum while increasing the changes of impact to their presentations.

IBERGRID is a mature and lively community that seamlessly work with a wide impact on the Spanish and Portuguese research communities. It constitutes a major example of bi-lateral cooperation and a solid partner for international cooperation.

The Conference Chairs
Ignacio Blanquer, Isabel Campos
Gonçalo Borges, Jorge Gomes

Chairs

Conference Chairs

Ignacio Blanquer, Universitat Politècnica de València - UPV
Isabel Campos, Instituto de Física de Cantabria - IFCA-CSIC
Gonçalo Borges, Laboratório de Instrumentação e Física Expermiental de Particulas - LIP
Jorge Gomes, Laboratório de Instrumentação e Física Expermiental de Particulas - LIP

Scientific Advisory Committee

Ignacio Blanquer
Isabel Campos
Carlos Fernández
Francisco Castejón
Jesus Marco
Vicente Hernández
Jorge Gomes
Gonçalo Borges
Mario David
João Pina
Alvaro Fernández
Andreu Pacheco
António Pina
Eduardo Huedo
Enol Fernńdez
Gaspar Barreira
Gonzalo Merino
Ignacio López
Javier G. Tobo
Javier Lopez
Jose Salt
Josep Flix
Tomás de Miguel
Victor Castelo
Carlos Redondo
Rosa Badia
Miguel Cárdenas-Montes
Germán Moltó
Erik Torres
Damián Segrelles
Carlos de Alfonso
Vítor Oliveira
Helmut Wolters

Local Organizing Committee

Ignacio Blanquer
Isabel Campos
Carlos Fernández
Antonio Fuentes
Francisco Castejón
Miguel Caballer
Enrique Bayonne

Sponsors

Jointly promoted by
Spanish Ministry of Science and Innovation
Fundação para a Ciência e Tecnologia

With the Support of

Instituto de Física de Cantabria - IFCA-CSIC
Universitat Politècnica de València - UPV

Session I

Energy Efficiency Infrastructures

SPARKS, a dynamic power-aware approach for managing computing cluster resources

J. Martins G. Borges, N. Dias, H. Gomes, J. Gomes, J. Pina, C. Manuel

Laboratório de Instrumentação e Física Experimental de Partículas, Portugal
martinsj@lip.pt, goncalo@lip.pt

Abstract. SPARKS is a new flexible and modular solution to optimize the execution of computing tasks with the minimum of available resources, focusing on power consumption reduction. SPARKS acts as a third party agent inter playing with the Local Resource Management System (LRMS) and the cluster resources. The agent tries to influence the resource allocation decisions with the objective to concentrate executing tasks in non-filled hosts, minimize the necessity to wake-up new resources, and power down unused hosts. The proposed approach provides the necessary resources to the LRMS without really interfering on any of its scheduling decisions guaranteeing that the resource allocation policies are never infringed.

1 Introduction

High Energy and Particle Physics research institutions are well known to host scientists performing state of the art research with extraordinary computing power requirements. Such researchers are normally part of wider worldwide collaborations focused, at a technical level, on the processing of large amounts of experimental data collected at big particle accelerators such as the LHC at CERN, or on the production of complex Monte-Carlo studies used for the analysis of the experimental data or computation of theoretical expectations. The profiles and needs of researchers are very diverse. Depending on their precise work or analysis tasks, users may need to access to different layers of computing power ranging from interactive machines for the execution of short-time tasks, to the submission of heavily expensive memory, time and cpu jobs to computing clusters or to distributed grid infrastructures. Therefore, High Energy and Particle Physics research institutions have to operate and offer a high number of services.

As the computing power requirements for these communities keep increasing, funding agencies do not follow the same trend and are normally reluctant to devote funds for the replacement of obsolete IT infrastructures, or pay for its operational costs such as electricity consumption which is becoming more and expensive as time goes by. Research institutions are facing this challenge through innovation, applying IT technologies either based on virtualization or in other green computing approaches, focused on reducing the operational costs of their e-science infrastructures [1]. One standard direction to tackle this problem is

through the renovation of computing resources by more environment-friendly (less power-consuming) replacements. Another valid approach, which can be adopted simultaneously, is an attempt to minimize the power consumption. The large size of computing clusters and the cyclic variations of load, where resources may be overloaded during certain periods of time or underutilized in others, are immediate candidates.

In this paper we present a modular software solution (`SPARKS`) to manage idle computing resources included in a computing cluster and managed by an external Local Resource Management System (`LRMS`). `SPARKS` provides a reliable yet flexible mechanism to power off unused nodes, and power then up again, if necessary, without compromising the level of services promised to a single or to multiple user communities. Its main objective is to optimize the execution of computing tasks with the minimum of available resources focusing on power consumption reduction. Following this brief introduction, Section 2 provides a general insight of related work in terms of infrastructure advances as well as software solutions aiming for the reduction of electricity costs. The paper continues in Section 3 with the description of the system including its motivation, requirements, architecture, policies and directives, and with an explanation of the power machine states and algorithm. Section 5 presents and discuss the first results of the prototype.

2 Related Work

The area of green computing applied for resource management has been widely exploited by the general IT community. Under the scope of pursuing "greener" hardware solutions to enhance current infrastructures, the following factors are considered as the most relevant ones:

- The processor choice is of great importance namely the relationship between its frequency and the so called *"Thermal Design Power (TDP)"*, i. e. the amount of power a processor is expected to dissipate to prevent overheating. For example, a frequency change from 3.2 to 3.0 GHz might decrease power consumption by 30%-40% at the cost of slowing down the actual work for only 10% [2]. Some hardware designs may even support *"Dynamic Voltage and Frequency Scaling" (DVFS)* mechanisms where the processor voltage and frequency are modified upon circumstances.
- The multi-core processor evolution allowed to deliver more computing power per Watt as compared to single core processors. Multiple execution cores consume about the same amount of energy meaning that each individual core runs at lower frequency performing slightly less work, but introducing a large drop in the amount of energy consumed. This fact triggered a quest for the replacement of old-fashion IT resources, and the appliance of multiple energy optimization techniques in multiprocessor environments such as the ones described in [3, 4, 5]. An extensive study performed at `CERN`, the biggest particle physics laboratory in the world, using the `SPECINT 2000` benchmark [6], showed that the introduction of dual-core systems based on the Intel Core

micro architecture increased the computing power per Watt ratio by a factor of 20 and 30 when compared to single core processor performance [2].
- The ability to accommodate more tasks in the same multi-core processor implicitly comes with the requirement to increase available memory, and therefore, preserve a constant memory per core ratio. Depending on the technology, memory modules may consume between 5 - 10 Watts per GB, and become an important source of concern for power reduction costs. New types of hardware memory modules are emerging in the market, such as the *"Double Data Rate 4"* series, improving their bandwidth rates and decreasing power consumption at each release. DDR4 increases performance up to 50% over DDR3, delivers a 20% reduction in voltage over DDR3 but however, when DDR4s additional power-saving features are taken into account, total overall power savings versus DDR3 can be as much as 40%.
- Power supplies technologies are also pursuing strong enhancements. Until some time ago, the standard efficiency for this precise components was lower than 80% simply because power suplies generate to much heat while converting AC current into the DC current needed by the different components. Today, it is now possible to reduce energy losses by a factor of four, and achieve efficiencies higher that 90% [7, 8].
- The adoption of *"Solid State Drives (SSD)"* as disks results in a mitigation of the energy consumed by mechanical parts normally accounting for 65% of the total amount of energy consumed [9].

The evolution of an IT computer centre is, on itself, a very complex process. The replacement of IT resources is normally progressive due to the nature of the available funding, or to the diversity of requirements that may emerge with time. As a result, there is a high probability that computing centres become very heterogeneous, either during the transition phase to "greener" hardware or due to the natural evolution of the centre. The heterogeneity of resources may open a new window for a more optimized management of hybrid resources, mixing low power systems with high performance ones, as detailed in [10].

Solutions focused on the management of the resources themselves are also a topic of great interest. A valid approach is to empower the Local Resource Management Systems (`LRMS`) with the capability to power up and power down resources on demand. `LRMS` are complex software solutions designed to optimize the management of computational tasks, requiring the access to large sets of computing resources, normally identical and traditionally connected through small local area networks. Its architecture consists of a standalone server, responsible for mapping the execution tasks to be best available computational resources, and of `LRMS` agents or sensors, responsible for the local monitoring and management of the tasks executing in the computing nodes. The executing tasks are submitted by the end-users, and the `LRMS` assigns them a quantitative priority, computed according to `LRMS` pre-defined policies, depending on specific institutional requirements and conditions. For example, some institutions may be interested in guaranteeing the fulfillment of service level agreements celebrated with certain communities, serving all the others on a opportunistic basis.

Other institutions may be interested in providing the same level of service to all communities, leveraging the access to computing resources with an adequate fair share mechanism.

Some commercial LRMS offer a tight power-saving integration in the resource allocation algorithm [11]. There are, nevertheless, other power-aware configurable softwares that, depending on configurable green computing policies, may power up or power down computing resources on demand. The work reported in [12] proposes an Energy-Aware Reservation Infrastructure (EARI), based on the philosophy that it is possible to predict nodes usage and switch on the nodes required in a near future. The systems relies on a negotiation process with the user to be able to establish the most appropriate advance reservation: the users give an estimate of the duration of the job, the system proposes job starting times that best fit the green computing policies in place, and the user selects one of the offered possibilities. The system evaluation is based on the recent history (regarding reservations, user activity and resources usage), and is under test in the Lyon pole of the Grid'5000 machine [13]. The same authors have recently extend the concept with EIRIS, an Energy-Efficient Reservation Infrastructure for large-scale Distributed Systems [14], aiming to provide a unified and generic framework to manage resources from Grids, Clouds and dedicated networks in an energy-efficient way.

3 System description

3.1 Motivation and requirements

Having in consideration the fact that the activity in High-Energy Research institutions is based on sequential processing jobs, either submitted locally or remotely, by different user communities expecting the fulfillment of Service Level Agreements, and with random load peaks impossible to predict, the following functional requirements emerged for a system that could be used to optimize power consumption in computing farms:

1. Minimum power consumption: The system should decrease the electricity operation costs, optimizing the requests for computing power with the minimum powered-up resources.
2. Plugable power-aware functionalities: The system must introduce the power-aware functionalities without disrupting the cluster operation and working as a plugin that can be added or removed on demand.
3. Fulfillment of Service Level Agreements: The system must guarantee that Service Level Agreements are not infringed. In practice, this means that the scheduling and resource allocation approaches determined by the LRMS can not be changed or disrupted.
4. High-granularity "on demand" policies: Different policies may apply to different topology levels (hosts, groups of hosts, LRMS queues or globally) following an agreed schema of prevalences. The system must be able to apply and change policies on demand.

5. `Time dependent power-aware policies:` The administrator must be able to define and impose operation thresholds to limit the resource offer. The policies should consider scenarios where the electricity cost changes along the time of the day, or along the day of the week.
6. `Resource dependent power-aware policies:` The system must power on primordially the resources with the best (cpu power / power consumption) ratio, and power off primordially the resources with the worse (cpu power / power consumption) ratio.
7. `Power management directives:` The administrator must be able to force a host or a group of hosts to migrate and maintain itself in a well-known established state. In such cases, the system must be able to adjust itself in order to guarantee that the power-management policies continue to be fulfilled.
8. `LRMS Support:` The system must be able to support multiple `LRMS` implementations. The architecture must be flexible enough to allow the inclusion of specific `LRMS` connectors without changing the core of the system.
9. `Multiple power management protocols support:` The system must support general power management protocols such as SSH or IPMI as well as other power management solutions (such as proprietary tools) working as modules, to cope with the wide spectrum of resource types included in the computing cluster.

A study of the market showed that most of the available software solutions either propose the migration to a different scheduling approach (most of the times commercial software or open source / open license software tuned to a precise and concrete environment), or suggest changes to be implemented in the `LRMS` scheduling algorithm currently in use. Such approaches might offer a very tight and controlled way to manage resource allocation having power consumption reduction in mind but, their adoption, implies a cost effort transition process from an administration point of view. The enhancement of the `LRMS` scheduling algorithm raises strong concerns about code scalability and maintenance, and represents lost of flexibility during software upgrades.

There are not so many plugable systems that could be introduced in a production computing cluster system without disrupting its operation and the policies applied. [15] and [16] are technical solutions that do fit the previous requirements. However, the testing of [15] showed that it lacks flexibility since it takes for granted the flat assumption that resources are in a unique `LRMS` queue or that the same resource does not spawn through different `LRMS` queues. From the documentation, it was not clear how [16] would work under such context. It was also not clear that both solutions could apply power reduction policies based on the properties of resources. The premise of a homogeneous cluster is frequent but is often not the case in many research institutes, where resources are acquired by different research groups depending on their funding periods and available budgets. Consequently, computing clusters can significantly become very heterogeneous, and an efficient power-aware cluster management system should take into consideration the different consumption profiles, giving preference to the

most efficient resources for power up, and to the less power efficient resources for primary power down. Finally, it was difficult to find a system able to adjust itself to multiple electricity regimes imposed upstream by the power companies, where the cost of electricity may vary along the day time and week day.

In conclusion, it was very difficult to find a tool that could fulfill the imposed requirements. Therefore, it was necessary to start development towards a new power aware cluster computing resource manager, able to adapt itself to very conservative or to more relaxed regimes, both in terms of electricity costs and level of services, changing as a function of time. The system must also be capable to deal with intrinsic properties of resources it has to manage (such as their electricity consumption values versus cpu performance) so that is able to take educated decisions regarding the best resources to enable or disable within a computing cluster.

3.2 System architecture

`SPARKS` follows a blackbox approach in the sense that it acts as an external actor constantly monitoring tasks pending for execution. Through a pooling cycle, `SPARKS` will deliver resources to the `LRMS` once they are needed for the execution of tasks, and will remove unnecessary resources from the computing cluster if they have not been used during a certain period of time. All decisions are taken according to predefined power management policies, and with the ultimate goal to reduce the general cluster power consumption without compromissing the communities perspective regarding the level or quality of delivered service.

Figure 1 depicts the component architecture of the `SPARKS` system composed of 3 execution modules (`LRMS Prober`, `Transition Analyser` and `Transition Gatekeeper`) coordinated by the `SPARKS Orchestrator`. The workflow of the different components executed at each cycle is the following:

1. The `Orchestrator` bootstraps the processing sequence by reading the configuration file. If a configuration is not available, the system is capable of generating one, based on the image captured by the `LRMS prober`. This self-generated configuration file imposes the default power-management policies, and traduces the current `LRMS` queue and resource topology into `SPARKS` format.
2. The `LRMS Prober` is a modular implementation relying on the invocation of specific `LRMS` plugins. The `LRMS` probe is responsible for executing the appropriate `LRMS` commands, and for translating their response into a generic format, feeded back to the `Orchestrator`.
3. The `Transition Analyser` is `SPARKS` decision maker. It is the core component that, based on the information collected by the `LRMS Prober` and on the implemented fine grained power management policies (defined in the configuration file), decides if a new set of hosts should be turned-on and in which order, or if any unused hosts should be turned-off.
4. The `Transition Gatekeeper` is the component that, once the decisions are taken, enforces those decisions by the execution of power management directives. The configuration file defines which protocol, command or tool should

be used in the context of a specific host, generating the invocation of the specific power management plugins.

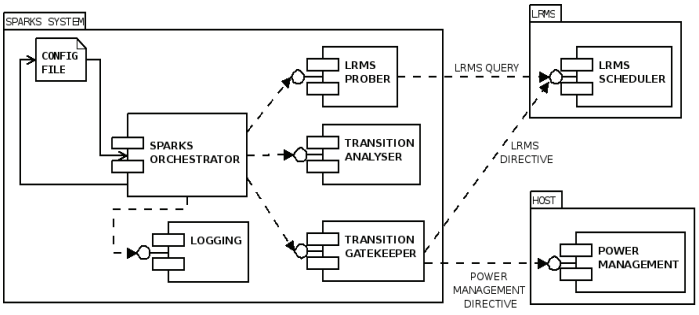


Fig. 1. Sparks agent system architecture

3.3 System policies and directives

SPARKS foresees the possibility to enforce different power management policies and directives. Through policies it is possible to condition the behaviour of the system and the execution of its algorithm in order to achieve a specific long term behaviour. Directives are system instructions used to enforce the persistence of well defined resource states. Both policies and directives apply at four hierarchical layers: global, LRMS queues, groups of hosts and individual hosts. The fours levels of granularity provide the necessary flexibility to act on the most appropriate set of resources. In case of divergence between multi-level policies, a pre-defined prevalence schema establishes that the policy applied at the highest granularity layer prevails, avoiding incoherent or divergent states. As an example, a directive targeted to a predefined group of hosts will only act on an individual resource if a different policy is not already applied for that individual host. This schema may also be used to establish relevant exceptions. The available set of policies and directives are:

Prime time policy: Allows the administrator to define time slots as a function of the daily hour, or alternatively, as a function of the day of the week. Such time periods, combined with limits on the maximum and minimum number of available slots, restricts the availability of computing resources as a function of the day time or week day. The final objective is to pursue the less expensive time periods in terms of electricity costs, allowing more hosts to power up during less expensive periods. The limits can be defined in absolute numbers, specifyng the exact number of cores that may be in use, or in relative numbers, specifying the fraction of cores that may be available with respect to the total. The limits established at a global level define default values for all the LRMS queues but can

be further refined in the proper queue configuration section.

`Resource property policy:` Enables the system to classify resources according to specific sets of properties. One predefined value in the system, which can be set by the system administrator at configuration time, is the host `score` representing any useful arbitrary quantity. As an example, the `score` of each host or group of hosts may be mapped to their (cpu performance / power consumption) ratio. Following this definition, the system will always prefer to power up hosts with higher `scores` (i.e. with more cpu capacity and less power consumption) and power down non-used hosts with lower scores.

`Condition-Action policy:` Enables the system to trigger a directive based on a specific condition. This is a useful utility where the administrator may fine tune their power management goals under very specific conditions. By default, the only implemented condition is the `kernel` one. If the administrator defines a specific kernel version, the system will drain and subsequently disable all the resources that do not have it in operation. There is here the implicit assumption that the resource will then power up with the appropriate kernel version otherwise the resources may enter in several reboot cycles. This condition was implemented to provide a fast draining mechanism for urgent or emergency kernel upgrades due to security issues.

`Continuous directive:` The resources will remain powered-up in a continuous mode, even if eligible for shutdown. Case the resource is manually powered-off by an administrator, the system will power it up at the next pooling cycle.

`Shutdown directive:` The resources will be shutdown, and will remain in that state even if more resources are needed. Case the resource has been manually powered-up by an administrator, the system will power it off at the next cycle.

`Disable directive:` The system will ask the `LRMS` to put the resources in quarantine, i. e., the resources will not accept new tasks.

`Exclude directive:` The system will stop acting on the resource and the resource is removed from the analysis process.

3.4 Algorithm and resource transitions

The system establishes well defined transitions between sets of deterministic states that caracterize the power status of each host. The transitions between the different states are illustrated in Figure 2 which may be triggered by system automatic decisions or by the invocation of specific system directives. The exception to the previous rule is the case of the `exclude directive` meaning that the system will neglect a specific resource from the analysis process. The full arrows in Figure 2 represent synchronous deterministic transitions, and the dashed arrows stand for asynchronous undeterministic transitions which may

Type	Global	Queues	Host Groups	Hosts
`prime time (P)`	X	X	-	-
`resource property (P)`	-	-	X	X
`condition-action (P)`	X	X	X	X
`continuous (D)`	-	-	X	X
`shutdown (D)`	-	-	X	X
`disable (D)`	-	X	X	X
`exclude (D)`	-	X	X	X

Table 1. Focus of `SPARKS` policies (P) and directives (D).

only finish in later cycles. The following states and transitions are expected to exist:

- `off state:` The host is powered-off. A host may appear in this state as a consequence of a system decision, or as a direct invocation of the `shutdown directive`. On the contrary, a host may never reach this state if the `continuous directive` is invoked.
- `recruited state:` A power on signal has been sent and the host is going through the boot process. The host is expected to appear in `idle state` within $t \leq$ `boot_wait` s otherwise it is considered in an `unknown state`.
- `idle state:` The host is powered up but it is not executing any task. All its computing processing units (slots) are free. The host may transit to `busy state` if it is eligible to execute tasks within $t \leq$ `shutdown_hold` s, otherwise it is eligible for shutdown and might enter in the `dismissed state`.
- `busy state:` The host is powered-up and available to execute tasks. The host may have all its processing units partially free or completely occupied.
- `draining state:` The host is powered up but it will not receive further executing tasks even if it has free processing units. A host may appear in this state as a consequence of the system decision process, or as a direct invocation of the `disable directive`.
- `dismissed state:` The host has received a power down signal, and it is going through the shutdown process. It is is expected to appear in the `off state` for $t \leq$ `shutdown_wait` s otherwise it is considered in an `unknown state`.
- `unknown state:` This system state is a placeholder for a wide set of occurrences that may leave the host in an incoherent or unknown state. Examples are kernel panics, problems in hardware components, lost of network, etc.

The algorithm followed by `SPARKS` to define how and when a transition must occur is described hereafter:

1. `SPARKS` obtains the list of hosts, computes the initial states and initializes counters.
2. The hosts for which directives have been defined are separated, and the corresponding transitions are applied.

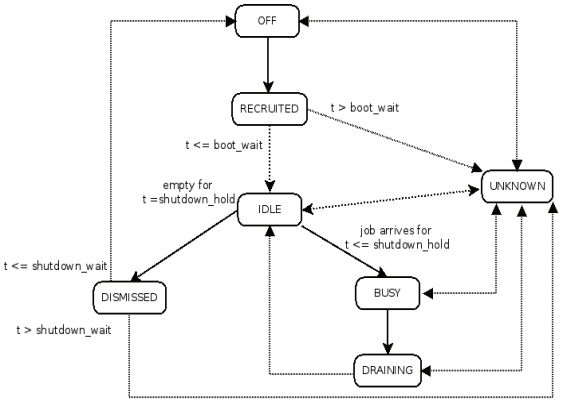


Fig. 2. Power state machine diagram. The full lines represent transition between syncronous states. The dash lines represent transitions to / from asynchronous states.

3. Hosts on `dismissed state` are analyzed to determine if the timeout has been exceded and if the host should go to the `unknown state`.
4. All remaining hosts are separated according to their states `busy, draining, idle, recruited` and `off state` are inversely ordered according to their score property. Busy host are also inversily ordered according to the number of executing tasks.
5. All hosts in `busy state` are analysed and if there is excess of slots offer in any host, it is reduced. We try to favour the more efficient hosts (best scores) by trying to keep the machines packed with jobs and analyze the hosts on reverse order according to their score. This way we promote the worst and less used hosts for shutdown.
6. All hosts in `busy, draining, idle, recruited` and `off state` are analyzed to check which ones need their offer increased.
7. When this step is reached, all available hosts have been analyzed and the system is ready to target the `idle state` hosts for shutdown as long they are in this state for longer than the `shutdown_hold` timeout.

4 Results

The first `SPARKS` prototype has been developed and is under testing since mid-May 2011 at LIP, a High Energy Particle Physics institution heavily involved in the WLCG computing activities. Figure 3 presents the average power consumption (in KWh/h) measured at LIP's main computing centre since April 2011. From its analysis, it is possible to define two important transition instants with significant impact on power consumption reduction:

1. The introduction of `SPARKS` as a power manager of LIP computing cluster in mid-May 2011 resulted in an immediate reduction of 17% comparing the average power consumption between April 2011 and June 2011. Moreover, the

average power consumption from June 2011 to December 2012 is 48.9 KWh/h which should be compared to values in the interval between 60-65 KWh/h for the periods before June 2011. Some resources were decommissioned along the referred period, and it is currently impossible to determine that specific contribution, but we believe that given the amount of disconnected resources and their power consumption, SPARKS was the biggest contribution to the sustainable decrease of electricity costs.

2. In January 2013, a strong decommission of storage servers occurred which introduced a further reduction on power consumption. Although uncorrelated, the fact that huge sets were migrated to other locations, less computing tasks start being scheduled at LIP which allowed a further improvement on the number of resources powered-off in the computing cluster. The average power consumption value for the period between January 2013 to May 2013 resulting from both contribution is 39.1 KWh/h.

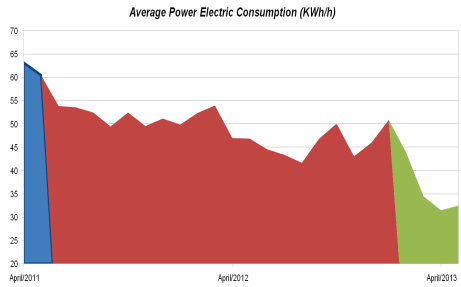


Fig. 3. Average power consumpting for LIP's Computing Centre (KWh/h) from April 2011 to May 2013.

The effectiveness of SPARKS is even more visible in figures under Table 2 where one can clearly identify a correlation between the number of running jobs and the number of powered-up hosts between May 2012 and May 2013. Although the number of pending jobs is normally higher, the system never enables all the resources, following the specific policies and limits that have been imposed.

5 Summary and Conclusions

SPARKS has been prototyped to satisfy a minimum set of basic requirements focused on serving execution tasks with the minimum of resources to reduce power operational costs. Development was started from basis since other open-source solutions either imply changes or substitution of the LRMS system, or do not allow a friendly interaction without breaking the current operation functionality and scheduling policies. There was also the requirement that the level of service for certain communities couldn't be broken, and that the system had to adjust

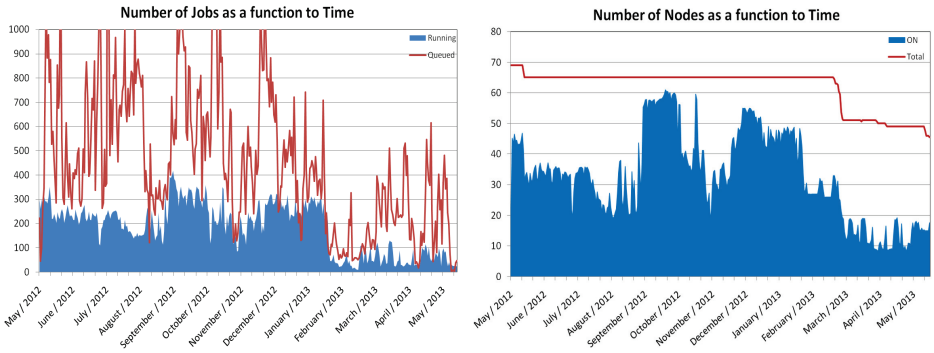


Table 2. Top: Number of pending and running jobs as a function of time; Bottom: Number of nodes powered-up as a function of time. Results are obtained for the period between May 2012 and May 2013

.

itself to power regimes imposed by power companies and that can change as a function of the day time or the week day. A set of policies was defined to allow the configuration of the system behaviour at several granularity levels, giving the desired flexibility for power management at the resource level. After almost two years in operation, the draft results showed that the introduction of `SPARKS` implied a decrease of the average consumption from 60-65 KWh/h to 49 KWh/h. The system is still in a draft phase and some further verifications have to be done. For example, it needs further developments to deal with MPI jobs. On the other hand its needs to increase its potential of customization of policies so that it can be fully adaptable and customizable by administrators.

Acknowledgements

G. Borges would like to thank to the Portuguese Foundation for Science and Technology under the context of the Ciência 2007/2008 programs jointly funded by the European Social Fund and by MCTES national funds (POPH - NSRF-Type 4.2).

References

1. Green Grid Consortium, 2009. http://www.thegreengrid.org/
2. A. Hirstius, S. Jarp, A. Nowak, "Increasing data centre power efficiency: An overview of CERNs approach to energy efficient computing." CERN Technical Reports (2008).
3. R. Bianchini and R. Rajaniony. "Power and energy management for server systems". Computer (Long Beach, CA), 37(11):6876, 2004.
4. C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. Keller. "Energy Management for Commercial Servers". Computer, 36(12):3948, 2003.

5. E. Pinheiro, R. Bianchini, E. Carrera, and T. Heath. "Load balancing and unbalancing for power and performance in cluster-based systems". Workshop on Compilers and Operating Systems for Low Power, volume 180, pages 182195. Citeseer, 2001.
6. Sandard Performance Evaluation Corporation, Available at http://www.spec.org/
7. Liang, S.A. Low cost and high efficiency PC power supply design to meet 80 plus requirements. IEEE Int. Conf. on Industrial Technology (2008), pp. 1-6.
8. Hoelzle, U., Weihl, B. High-efficiency power supplies for home computers and servers. Available at http://services.google.com/blog_resources/PSU_white_paper.pdf.
9. Heath, T., Diniz, B., Carrera, E.V., Meira, W., and Bianchini, R. Energy conservation in heterogeneous server clusters. Proc. tenth ACM SIGPLAN symposium on Principles and practice of parallel programming (2005) 186-195.
10. B.-G. Chun, G. Iannaccone, G. Iannaccone, R. Katz,L. Gunho, and L. Niccolini. "An Energy Case for Hybrid Datacenters. HotPower, 10 2009.
11. Green Computing powered by MOAB, Available at http://www.clusterresources.com/solutions/green-computing.php
12. Lefvre, L., Orgerie, A. "Towards Energy Aware Reservation Infrastructure for Large-Scale Experimental Distributed Systems Parallel Processing Letters", 19 (3) 2009 419-433.
13. Cappello, F. "Grid'5000: a large scale and highly reconfigurable grid experimental testbed", 6th IEEE/ACM International Workshop on Grid Computing (2005) 99-106 (DOI: 10.1109/GRID.2005.1542730)
14. Lefvre, L., Orgerie, A. "ERIDIS: Energy-efficient reservation infrastructure for large-scale distributed systems", Parallel Processing Letters, 21 (2) 2011 133-144.
15. http://www.ciul.ul.pt/∼ATP/SPIRIT/
16. Fernando Alvarruiz, Carlos de Alfonso, Miguel Caballer, Vicente Hernández "An Energy Manager for High Performance Computer Clusters" 10th IEEE International Symposium on Parallel and Distributed Processing with Applications (2012), pp. 231-238 (ISBN: 978-1-4673-1631-6, DOI: 10.1109/ISPA.2012.38)

Performance and Energy Aware Scheduling Simulator for High-Performance Computing

César Gómez-Martín (CénitS)[1], Miguel A. Vega-Rodríguez (University of Extremadura)[2], José-Luis González-Sánchez (CénitS)[1], Javier Corral-García (CénitS)[1] and David Cortés-Polo (CénitS)[1]

[1]CénitS (Research, Technological Innovation and Supercomputing Center of Extremadura) - N-521, Km. 41,8 10071 Cáceres (Spain)
{cesar.gomez,joseluis.gonzalez,javier.corral,david.cortes}@cenits.es
[2]University of Extremadura - Escuela Politécnica de Cáceres, Avda. Universidad s/n 10003 Cáceres (Spain)
mavega@unex.es

Abstract. Job scheduling is a very complex task that influences the performance of High-Performance Computing (HPC) infrastructures, thus, a simulator of real computing environments for an accurate evaluation of scheduling and resource selection policies helps ICT and data center managers to make decisions with a solid experimental basis. There are several simulators that try to address performance and somehow estimate energy consumption, but there are none in which the energy model is based on benchmark data that have been countersigned by independent bodies such as the Standard Performance Evaluation Corporation (SPEC), this is the reason why we have implemented a Performance and Energy Aware Scheduling (PEAS) Simulator for HPC. In order to validate the simulator and understand how different computing workloads behave depending on scheduling policies, several user-centric evaluation metrics have been evaluated and compared with previous studies.

1 Introduction

Today job scheduling is still a very complex task that may have a significant influence on the performance of computing infrastructure, thus, a good job scheduling system is very important to reduce operating costs. A study of actual workloads can significantly increase the energy efficiency of IT infrastructure. For more than half a century the scientific computing community has mainly focused on system performance while forgetting other important aspects like energy efficiency. Nowadays, with the arrival of embedded computers, the boom in battery dependent devices, higher energy costs and global warming, energy efficiency is becoming more important not only for hardware vendors, but also for ICT and data center managers, and green computing is no longer seen as an oxymoron. Moreover, the notion of power awareness or low power supercomputing is new [1], while extending battery life is the main goal in mobile devices, the main motivation of saving energy in data centers is to reduce operating costs, that

is the reason why new low-power Atom/ARM based servers are being introduced in supercomputing and regular data centers. When dealing with power savings among mobile, embedded and High-Performance Computing (HPC) applications, there are some differences and most of them are related to the need to address real-time operations in mobile environments or meeting wall-clock time requirements in some HPC scientific applications. There have been several works that study the impact of DVFS (Dynamic Voltage and Frequency Scaling) combined with a power-aware algorithm to save energy [2, 3], and some others about how to balance and save energy in clusters with virtual machines [4]. The innovation of the simulator presented in this paper is the inclusion of a Power and Performance benchmark, from the Standard Performance Evaluation Corporation (SPEC), to model the energy consumption and the performance of a cluster of servers, or a supercomputer, when running a set of jobs with a job scheduler.

1.1 Related Work

Application capabilities and data volumes are increasing exponentially, this leads to the need for more computing capacity and better management of HPC environments, so several business solutions use the above job data for intelligent and complex workload scheduling:

- IBM Platform LSF (Load Sharing Facility): a workload management platform for demanding and distributed HPC environments.
- SLURM is an open-source resource manager that is becoming popular because it can work with heterogeneous clusters of all sizes.
- PBS (Portable Batch System): originally developed for NASA to allocate computational tasks. There are various open implementations, OpenPBS and TORQUE, together with a commercial version called PBS Pro.
- Oracle Grid Engine: a distributed resource management (DRM) system that manages the distribution of user workloads to available computer resources.
- HTCondor: is an open-source computing framework for coarse-grained parallelization of computationally intensive tasks.

In [5] Krallmann, Schwiegelshohn, and Yahyapour defined three categories for job data as an approach to designing a job scheduling system:

- User Data: used to determine job priorities.
- Resource Requests: these data usually specify the number and architecture of the processors, the amount of memory, an estimation of the running time, etc.
- Scheduling Objectives: data that help the scheduler to generate "good" schedules.

According to the authors, a scheduling system should be divided into three parts: scheduling policy, objective function and scheduling algorithms. Other authors introduce a fourth decision making element called resource selection policy [6] to allocate jobs depending on the resource requirements or limitations.

There are simulators like MuPSiE [7] that follow the core principles of [5] and some others that also use a separate resource selection policy component like Alvio [6] and Kento-sim [8]. Other works propose energy management in multiprocessor environments with machine learning techniques such as [9].

1.2 Contributions

To properly manage a data center in general and a supercomputer in particular, in CénitS, along with the University of Extremadura, we are currently developing a Performance and Energy Aware Scheduling (PEAS) Simulator that will help us in decision making for the configuration of our workload management platform. Furthermore, with the simulator we intend to develop new resource selection and scheduling policies for better management of compute resources and energy consumption. In this paper we present the PEAS Simulator as an evaluation tool for system analysts and data center managers to define scheduling policies for their business solutions. The PEAS simulator has the following improvements over existing simulators:

- Implementation of a configurable resource selector able to meet certain constraints (energy consumption, wall clock time, performance, etc.).
- Use of actual power and performance standard benchmarks of current computing servers that generate new metrics and will allow new resource selection policies to be added in the future.
- Inclusion of updated workloads and benchmarking results in the simulator by simply adding standard files from the Parallel Workloads Archive and SPEC Power and Performance repository.

An early version of PEAS Simulator has been used to simulate real workloads from several supercomputing centers using industry accepted and widely recognized scheduling and resource selection policies.

In the remainder of the paper, we briefly describe the PEAS Simulator architecture in Section 2. Thereafter, in Section 3, we explain how the jobs are processed by the simulator to calculate power and time metrics and evaluate how "well" the scheduler behaves. Section 4 presents simulation results of some real computing center workloads that are used to validate the PEAS Simulator. In Section 5 we introduce the expected improvements of the simulator in upcoming versions. Finally, in Section 6, we outline the conclusions of this paper.

2 Architecture of the simulator

The PEAS Simulator is an event-driven job scheduler simulator that is meant to help in the evaluation and optimal utilization of hardware and energy resources of HPC environments. In Figure 1 the architecture of the simulator is sketched:

- Prior to the start of the simulation, a file containing the jobs workload in SWF format and a CSV file with the cluster configuration are processed.

– After the workload and the hardware resources are loaded into the simulator, the scheduling and resource selection policies are read from a configuration file.
– Once all the configuration files are properly processed, the scheduler starts to decide which jobs are processed (Policy Scheduler) and, with the help of the Resource Selector, assigns free resources to the highest priority request. If the job requests more resources than the total number of resources provided by the whole cluster, the request is not accepted by the scheduler (Unaccepted Jobs). If at a particular moment there are not enough free resources, the job is queued (Queued Jobs) and waits until enough resources are available.
– Finally, when all the jobs are finished, an output file containing all the evaluation metrics of every job and the whole schedule is generated. It is also possible to generate a log file that includes debugging information.

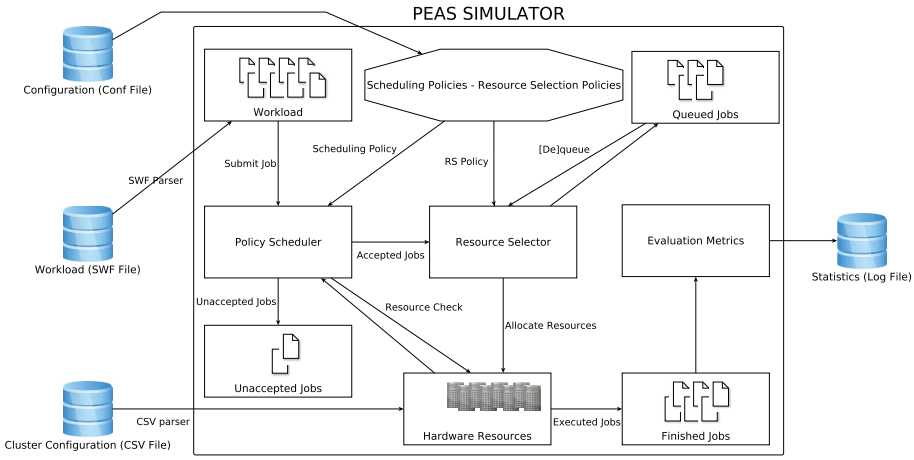


Fig. 1. PEAS Simulator Architecture

2.1 Workload modeling

A job can be defined by the following parameters:

– Submit time in seconds. Usually the first submitted job has a submit time of 0, meanwhile the rest of the jobs are sorted by ascending submittal times.
– Estimated duration of the job. This field is used for the user runtime estimate (or upper bound) and it is useful for backfilling policies.
– Actual duration of the job. The wall clock time the job was running (end time minus start time).
– Requested resources (number of processing cores, amount of main memory, etc.).

– Some other parameters of interest for some scheduling policies.

Today most supercomputing centers have a resource management software that is able to easily generate job traces, this is very important for an accurate study of scheduling policies because it is not easy to randomly generate traces for "real world" workloads. In order to face the latter problem, the parallel workloads that are being used to test the PEAS simulator have been extracted from the Parallel Workload Archive [10], which has a wide variety of real traces from supercomputing centers of very diverse nature. The standard workload format (SWF) was proposed by David Talby and refined through discussions by Dror Feitelson, James Patton Jones, and others [11]. We decided to use this format because it is very easy to parse, since every job is well defined and contains all the data that is needed to properly simulate a real HPC environment. In the simulator we have developed a parser for version 2.2, which is the last published version. In table 1 there is a real example of some parameters from a workload of the RIKEN Integrated Cluster of Clusters, RIKEN is an independent scientific research and technology institution of the Japanese government.

Table 1. Example Workload - RICC: RIKEN Integrated Cluster of Cluster, Oct 6, 2011

Job #	Submit Time	Runtime Estimate	Runtime	# Used CPUs	...
1	0	14400	222	80	...
2	1136	259200	244682	128	...
3	1160	259200	249628	128	...
4	1877	259200	259209	128	...
5	1903	259200	211161	128	...
6	1920	86400	78509	128	...
7	1920	86400	78391	128	...
8	1920	86400	77206	128	...
9	1920	86400	79139	128	...
10	1920	86400	77584	128	...
11	...	...	...	...	...

2.2 Power and Performance modeling

The power a microprocessor consumes and dissipates varies depending on the manufacturer, moreover, power by itself is not a measurement of efficiency, a compromise between performance and power consumption is better for defining energy efficiency. A system which consumes low levels of power but does not perform well will take longer to perform a task, and may ultimately consume more energy, thus, a more efficient server is the one that has the best performance per watt [12]. When comparing power specifications it is important to use metrics that are able to measure the same parameters. Intel and AMD have designed their own power specifications:

– Intel Thermal Design Power (TDP): is the maximum power a processor can draw for a thermally significant period while running commercially useful software.

– AMD Average CPU Power (ACP): is the average (geometric mean) power a processor was measured to dissipate while running a collection of four different benchmarks: TPC Benchmark*-C, SPECcpu, SPECjbb and STREAM.

Intel and AMD agree on the fact that it is not a good idea to measure power consumption by only looking at the spec sheets of different components and adding the totals together, because these only report the maximum power consumption. The best way to calculate the power consumed by "real world" workloads is to use a power meter [13]. In order to evaluate not only performance but also power usage, SPEC designed the SPEC Power and Performance benchmark which is the first industry standard that evaluates the power and performance characteristics of volume server class and multi-node class computers [14]. In the PEAS Simulator we have implemented a CSV parser that is able to process the files provided by SPEC and complies with version 1.12 of the SPEC Power and Performance benchmark. In table 2 the most relevant fields of a cluster configuration file in CSV format are shown.

Table 2. Example of Cluster configuration file

Vendor	Processor	#Chips	#Cores/Chip	RAM (GB)	OPS@100%Load	Watts@100%Load	...
ASUS	Xeon X3360	1	4	4.00	165064	118	...
ASUS	Xeon L5420	2	4	8.00	270621	170	...
ASUS	Xeon L5430	2	4	8.00	278927	173	...
Acer	Xeon X3470	1	4	8.00	309401	125	...
Acer	Xeon E3-1260L	1	4	8.0	295443	58	...
Acer	Xeon X5670	2	6	12.0	898544	267	...
Acer	Xeon X5670	2	6	12.0	897310	265	...
Acer	Xeon X3470	1	4	8.00	308318	124	...
Acer	Xeon X5670	2	6	12.0	890144	272	...

2.3 Scheduling Policy component

This component is in charge of storing all the jobs in a list-like structure that is ordered by arrival time. First come first served (FCFS) scheduling policies are the most common in queuing systems, they are easy to understand from a user point of view, although they are probably not the best option for a system administrator. Traditional FCFS is only used for experiments or studies, it is not a real alternative because it leaves a lot of resources idle. This issue is solved in modern job scheduling system with the inclusion of backfilling variants. With a backfilling mechanism all the idle resources may be utilized by lower priority jobs. In the current version of the PEAS Simulator we have implemented three scheduling policies:

– FCFS: first come first served is a well known scheduling policy that generates fair schedules. It is very inefficient due to fragmentation but it is easy to understand; jobs that arrive later are started later.

– FCFS Conservative Backfilling: this variant moves jobs forward only if all the previously queued jobs are not delayed. It does not affect any queued job, hence it is known as the conservative version [15].
– FCFS EASY Backfilling: EASY stands for "Extensible Argonne Scheduling System", it was developed for the IBM SP2 supercomputer and is also called Aggressive Backfilling because it allows short jobs to move forward if they do not delay only the first job of the queue, although the rest of the jobs ahead of it in the queue may be delayed [16].

2.4 Resource Selection Policy component

The resource selection policy component decides the nodes and processors among which a job must be allocated. Depending on the number of cores, memory, energy or power constraints, a job would be allocated in a different server. We have implemented First Fit Policy (FFP) in the current version of the PEAS Simulator. It is the simplest and most widely used selection policy. FFP finds the first n idle processors that meet the selected job constraints and allocates it to start the execution. Although in the first version of the simulator there is only one resource selection policy, this component has been designed to be easily extended in the future with new policies or algorithms.

3 Methodology

Since not all the processors and computing servers have the same performance (operations per second (OPS)), in order to compute a more reliable runtime for our simulation (SimulatedRunTime), we have to calculate the number of operations that correspond to every job of the workload. For that purpose, the actual time of the job, which can be obtained from the SWF file, is assumed to be the time spent by the job in a fictitious average processor. The fictitious processor is calculated as the average processor of all the servers that are inside the cluster configuration file at a 100% load (see subsection 2.2).

$$OPS_{CPU_{Average}} = \frac{\sum_{i=1}^{n} OPS_{CPU_i}}{n} \tag{1}$$

where:

– OPS_{CPU_i} is the number of operations per second at 100% load in CPU_i.
– n is the total number of CPUs of the cluster.

Based on the actual runtime of a job, the simulated runtime of the same job on the corresponding processor of the cluster is calculated as follows:

$$SimulatedRunTime = (\max_{i=1...AllocCPUs} \frac{OPS_{CPU_{Average}}}{OPS_{CPU_i}}) * Runtime \tag{2}$$

where:

- *Runtime* is the actual runtime of the job.
- OPS_{CPU_i} is the number of operations per second at 100% load in CPU_i.
- $OPS_{CPU_{Average}}$ is the number of operations per second at 100% load in $CPU_{Average}$. (See equation 1).
- $AllocCPUs$ is the number of allocated CPUs in which the job has been run.

3.1 Performance metrics

To evaluate how "good" the different policies are two user-centric performance metrics are calculated by the simulator:

- The average waiting time of the jobs is calculated as follows:

$$AWT = \frac{\sum_{i=1}^{n} t_i^w}{n} \tag{3}$$

  where:
  - t_i^w is the waiting time of Job_i.
  - n is the total number of jobs that have been accepted by the scheduler.

- The average response time can be also calculated:

$$ART = \frac{\sum_{i=1}^{n} t_i^r}{n} \tag{4}$$

  where:
  - t_i^r is the response time of Job_i.
  - n is the total number of jobs that have been accepted by the scheduler.

3.2 Energy metrics

Finally, to calculate the power consumption of every job, the energy needed by all the working processors to accomplish a given job is calculated with the following formula:

$$PowerConsumption_{Job} = \sum_{i=1}^{AllocCPUs} Watts@100\%Load_{CPU_i} * SimulatedRunTime \tag{5}$$

where:

- $Watts@100\%Load_{CPU_i}$ is the energy consumption of CPU_i at 100% load. (See table 2).
- $AllocCPUs$ is the number of allocated CPUs in which the job has been run.

Table 3. Performance and energy metrics for HPC2N, LLNL and RICC (100 jobs)

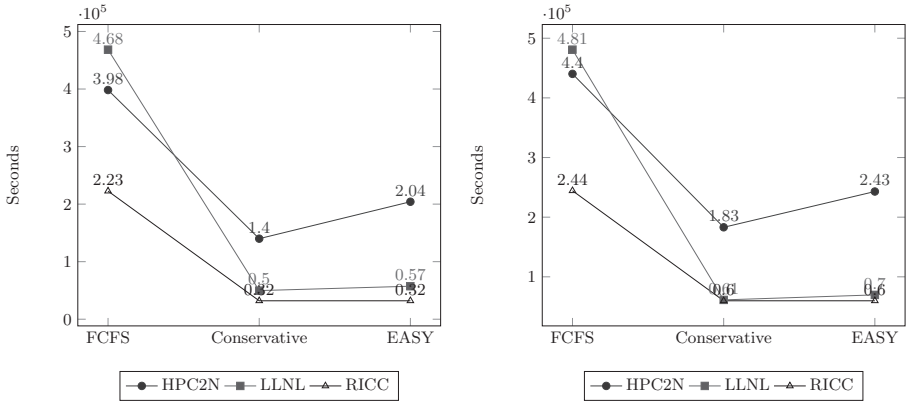
	FCFS			Conservative			EASY		
	HPC2N	LLNL	RICC	HPC2N	LLNL	RICC	HPC2N	LLNL	RICC
AWT (seconds)	398258	468456	222651	139969	49734	32029	204112	57325	32029
ART (seconds)	440243	480897	244332	183034	61326	59830	243002	69698	59830
Power Consumption (KWh)	269.417	214.058	34.0465	249.363	202.647	38.2591	237.208	205.421	38.2591

4 Experimental Results

In order to validate the PEAS Simulator, three workloads from large scale parallel systems in production have been tested. The configuration of the cluster can be seen in table 2. Since most of the logs contain several years of submitted jobs, only the first hundred of each logfile have been used for the validation process.

- HPC2N: is the workload of the High-Performance Computing Center North (HPC2N) in Sweden.
- LLNL: is the workload of a large Linux cluster called Thunder installed at Lawrence Livermore National Lab (LLNL).
- RICC: contains several months of jobs from the RIKEN Integrated Cluster of Clusters (RICC).

The results can be observed in Table 3.



(a) Average Waiting Time (b) Average Response Time

Fig. 2. Average Waiting Time & Average Response Time

Figure 2.a shows the average waiting time (AWT) for FCFS, FCFS Conservative Backfilling (Conservative) and FCFS EASY Backfilling (EASY). Most of the jobs within the three workloads are wide jobs, i.e. they need a lot of hardware resources, so it is hard for them to easily find enough idle processors. The reason

why there is a better AWT in conservative backfilling is because the scheduler guarantees them a start time avoiding starvation. In Figure 2.b the average response times (ART) for FCFS, Conservative and EASY are compared. There is again a better ART in conservative backfilling because of long wide jobs. EASY backfilling does not give a reservation for them, so more jobs can backfill ahead. There is no clear advantage between EASY and conservative backfilling, it depends on the characteristics of the workload. While EASY clearly benefits long narrow jobs, there is no clear advantage for long wide jobs. Conservative backfilling provides reservation while EASY offers better backfilling opportunities due to fewer blockages in the schedule [17].

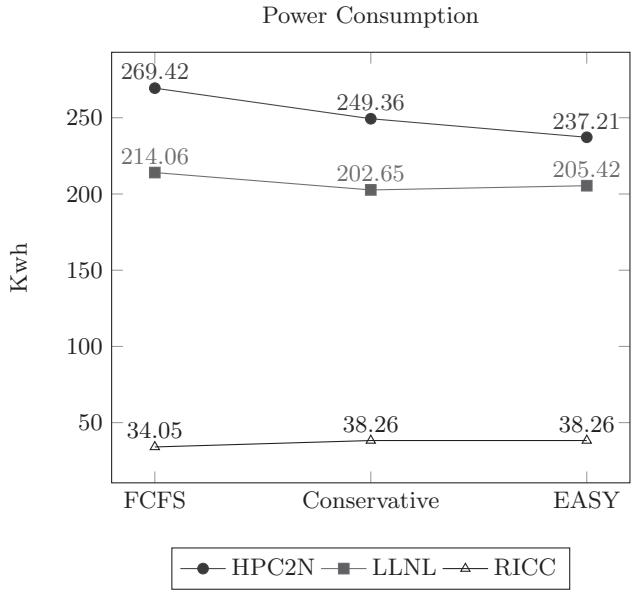


Fig. 3. Power Consumption

As expected, due to the resource selection policy (first-fit), energy consumption does not follow a clear pattern because it greatly depends on which processors are free when a job starts running (see Figure 3). However, this metric is especially important for future implementations of hardware selection policies that will try to reduce power consumption with optimization algorithms.

5 Future Work

Future contributions to the simulator are expected, including the following:

- Implementation of additional user-centric performance metrics weighted by job width [18]: average response time weighted by job width and average slowdown weighted by job area.
- Implementation of classic scheduling policies for experimental purposes with different sorting criteria such as: shortest job first (SJF), longest job first (LJF) and others.
- Multiobjective optimization algorithms will be used for a more "intelligent" scheduling policy.
- In order to quantify the impacts and benefits of modern power management functions, the SPEC Power and Performance benchmark also provides power consumption and the number of operations per second for a range of eleven throughput levels, from idle to 100%. We currently do not use graduated measurement intervals but we plan to use them to find a good compromise between power and performance.

6 Conclusions

Although green computing is somehow still seen as an oxymoron for HPC center managers, most of them are aware of the importance of reducing operational costs even though performance is still the main goal for HPC. With the PEAS Simulator we intend to help in the management of data center workloads trying not only to achieve good response times but also to consume as little energy as possible. There is a wide field of study for scheduling and resource selection policies combined with artificial intelligent and multicriteria optimization and the PEAS Simulator has been structured and designed to implement new algorithms to address some of the unknowns of system analysts and administrators. In this paper we present a tested early version of PEAS Simulator, with some of the scheduling policies that are implemented by modern queuing and scheduling systems. The main contribution of the PEAS Simulator is the implementation of an innovative Power and Performance aware component that is able to model the actual energy consumption of real computing center workloads. Further versions of the PEAS Simulator are expected to include improvements as well as possibilities for helping ICT managers manage computing infrastructures, and will include artificial intelligence and multi-objective optimization algorithms for better utilization of hardware resources.

Acknowledgements

This research is part financed by the ERDF Fund Programme: Extremadura Operational Programme 2007-2013, Development of the Knowledge Economy (R&D&I: Information Society and ICTs), Research and Technology Development (R&TD) Activities in Research Centres.

References

1. Hsu, C. H., & Feng, W. C. (2005, November). A power-aware run-time system for high-performance computing. In Proceedings of the 2005 ACM/IEEE conference on Supercomputing (p. 1). IEEE Computer Society.
2. Choi, K., Soma, R., & Pedram, M. (2004, August). Dynamic voltage and frequency scaling based on workload decomposition. In Proceedings of the 2004 international symposium on Low power electronics and design (pp. 174-179). ACM.
3. Chen, J. J., & Kuo, C. F. (2007, August). Energy-efficient scheduling for real-time systems on dynamic voltage scaling (DVS) platforms. In Embedded and Real-Time Computing Systems and Applications, 2007. RTCSA 2007. 13th IEEE International Conference on (pp. 28-38). IEEE.
4. Hu, L., Jin, H., Liao, X., Xiong, X., & Liu, H. (2008, September). Magnet: A novel scheduling policy for power reduction in cluster with virtual machines. In Cluster Computing, 2008 IEEE International Conference on (pp. 13-22). IEEE.
5. Krallmann, J., Schwiegelshohn, U., & Yahyapour, R. (1999, January). On the design and evaluation of job scheduling algorithms. In Job Scheduling Strategies for Parallel Processing (pp. 17-42). Springer Berlin Heidelberg.
6. Guim, F., Corbalan, J., & Labarta, J. (2007, October). Modeling the impact of resource sharing in backfilling policies using the alvio simulator. In Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2007. MASCOTS'07. 15th International Symposium on (pp. 145-150). IEEE.
7. Heine, F., Hovestadt, M., Kao, O., & Streit, A. (2005). On the impact of reservations from the grid on planning-based resource management. In Computational ScienceICCS 2005 (pp. 155-162). Springer Berlin Heidelberg.
8. Guim, F., Rodero, I., Corbalan, J., & Parashar, M. (2010, September). Enabling gpu and many-core systems in heterogeneous hpc environments using memory considerations. In High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on (pp. 146-155). IEEE.
9. Berral, J. L., Goiri, Í., Nou, R., Julià, F., Guitart, J., Gavaldà, R., & Torres, J. (2010, April). Towards energy-aware scheduling in data centers using machine learning. In Proceedings of the 1st International Conference on energy-Efficient Computing and Networking (pp. 215-224). ACM.
10. Feitelson, D. G., Tsafrir, D., & Krakov, D. (2012). Experience with the parallel workloads archive. Technical Report 2012-6. The Hebrew University of Jerusalem.
11. Chapin, S. J., Cirne, W., Feitelson, D. G., Jones, J. P., Leutenegger, S. T., Schwiegelshohn, U., ... & Talby, D. (1999, January). Benchmarks and standards for the evaluation of parallel job schedulers. In Job Scheduling Strategies for Parallel Processing (pp. 67-90). Springer Berlin Heidelberg.
12. Huck, S. (2011). Measuring Processor Power: TDP vs. ACP. White Paper, Revision 1.1. Intel.
13. AMD. ACP - The truth about power consumption starts here. White Paper: Power Consumption. AMD.
14. Poess, M., Nambiar, R. O., Vaid, K., Stephens Jr, J. M., Huppler, K., & Haines, E. (2010, April). Energy benchmarks: a detailed analysis. In Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking (pp. 131-140). ACM.
15. Lifka, D. A. (1995, January). The anl/ibm sp scheduling system. In Job Scheduling Strategies for Parallel Processing (pp. 295-303). Springer Berlin Heidelberg.

16. Mu'alem, A. W., & Feitelson, D. G. (2001). Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. Parallel and Distributed Systems, IEEE Transactions on, 12(6), 529-543.
17. Srinivasan, S., Kettimuthu, R., Subramani, V., & Sadayappan, P. (2002). Characterization of backfilling strategies for parallel job scheduling. In Parallel Processing Workshops, 2002. Proceedings. International Conference on (pp. 514-519). IEEE.
18. Streit, A. (2003). Self-tuning job scheduling strategies for the resource management of HPC systems and computational grids (Doctoral dissertation, PhD thesis, Faculty of Computer Science, Electrical Engineering and Mathematics, University Paderborn).

Session II

Interoperability and Federation of Infrastructures

Towards Federated Cloud Image Management

A. Simón[1], E. Freire[1], R. Rosende[1], I. Díaz[1], A. Feijóo[1], P. Rey[1], J. López-Cacheiro[1], C. Fernández[1], O. Synge[2]

[1] Fundación Centro de Supercomputación de Galicia, Santiago de Compostela, Spain
grid-admin@cesga.es
[2] owen.synge@jaysnest.de

Abstract. The new federated Cloud infrastructures require the development of specific utilities and technical solutions. One of the most crucial features is the virtual image management and distribution system. This paper summarizes the work developed within EGI FedCloud taskforce during the last year to deploy a sustainable and scalable federated VM image management.

1 Introduction

The number of projects based on Cloud IaaS resources are increasing each day. Projects like EGI-inspire are making efforts to help abstract away the differences between different API's and allow resource consumers to be Cloud agnostic. Cloud agnosticism will become the new Rosetta Stone for Cloud developers and site administrators. Different projects are using different API's, Cloud frameworks, storage systems etc. However this technology agnosticism increases the interoperability challenges. One of the most pressing forces is the VM images managing within a federated Cloud infrastructure. A federated Cloud increases the difficulty, since it uses different Cloud APIs, works with different Cloud vendors which in turn require different image formats, different contextualisation mechanism and so on. This paper is focused on some of the issues surrounding image management in a federated Cloud environment and how it was solved by the EGI FedCloud taskforce. It is organised in the following sections. First, Section 2 presents the state of the art regarding Cloud frameworks VM management, Section 3 describes the VMcaster/VMcatcher image management tools, how it was implemented by EGI FedCloud project and how it was configured at CESGA. Section 4 explains how VMcatcher event handlers work, and VMcatcher plugins to support different Cloud frameworks like OpenNebula or OpenStack. Finally Section 5 will present the conclusions and future work.

2 Related Work

The use of Cloud infrastructures in science has been documented, and it is a very promising field, but integrating Clouds with the Grid is a challenge, as related on Dillon et al. [1]. Goasguen et al. [2] presents the results of an internal production

Cloud service in CERN and suggestions to expand it to another Grid sites. Zhao et al. [3] presents a infrastructure of a dozen computing sites using OpenNebula as the management solution. It concludes that Clouds are very useful for science, but there are still many performance issues to be resolved. Hoffa et al. [4] reached similar conclusions regarding Cloud vs local deployments.

There are also many works that compare the different solutions for VM Management, like Xiaolong et al. [5], which compares OpenNebula and Openstack, and Laszewski et al. [6], which does a more complete survey including Eucalyptus, Nimbus and some other solutions. The existence of numerous trade-offs and fragmented market for these tools motivated us to support cross-grid environments.

On the realm of security, our solution emphasizes the authentication of users and the validation of VMs. There are other works on the area, but some, like Xi et al. [7] are concerned more with running trusted VMs on on untrusted environments, which can be seen as the opposite problem, and many others, like Schwarzkopf et al. [8] are concerned with improving the internal security of VMs maintained by Cloud users instead of infrastructure operators.

There are still other comparable solutions, Lagar-Cavilla et al. [9] use a non-local fork mechanism to spawn many copies of a VM across many sites, but this method would be at odds with current Grid practices. Diaz et al. [10] have a similar system that bridges OpenNebula and OpenStack, but it uses the Amazon EC2 API, which has licensing issues preventing us for using it, and does not address the authorization and validation of VMs. On a more partial resemblance, Maurer et al. [11] also automates some aspects of VM management and updates using an autonomous system, and Django et al. [12] changes the context of VMs on the fly to do load balancing and improve brokering. This last functionality would be invaluable for Grid operators, which must frequently tend to processes that get stuck due to unrealistic brokering requirements, and would also avoid many downtimes due to reconfiguration.

3 VMcaster/VMcatcher image management interface

VMcaster [13] and VMcatcher [14] are two different tools to generate and subscribe to virtual machine image lists. These tools use an internal database (similar to a podcast syndication or a Linux package manager) where images information and lists are stored into an internal SQLite database. SQLite has proved more than adequate for the low transaction rate of a image list subscriber and so deployment issues are just backing up a database file. In many senses VMcaster/VMcatcher tools are similar in concept to Debian's *aptitude* or RedHat's *yum* utilities.

These tools try to match the requirements set by the now completed HEPiX virtualisation working group [15]. The main task of this working group was to provide to sites with a way to control and manage Virtual Machine (VM) Image's provided by experiments, and execute the VMs in a trusted environment (similar to the current computing environment provided under Grid computing). Besides,

during the last year the EGI federated Cloud task force has recommended VMcatcher installation in its Cloud resource providers.

In this case since the software is made with the Grid in mind and to avoid *man in the middle* security issues, VMcaster/VMcatcher tools are based on the X.509 certificate authentication model. All the image lists are signed by an authenticated endorser with his/her personal X.509 certificate. This means all images are referenced by an Virtual Machine Image List which contains a secure hash (SHA512) signed using X.509 personal certificates (provided by the image list endorser). These Virtual Machine Image Lists are published, and interested sites subscribe to the Lists in the resulting catalogue.

When a resource provider receives an user instantiation request, the image validity is checked. If the Virtual Machine Image List is valid, the Image is contextualised and then instantiated (see figure 1). Using this mechanism a virtual image can be checked for validity, all image lists are signed and they provide a version number and expiration date. If the image list does not satisfy these requirements the image is not instantiated and the request is rejected.

Another important feature of VMcaster/VMcatcher is the support of Cloud framework agnostic tools, i.e. these tools do not depend on the Cloud solution used by the sites. Besides it can be integrated with different frameworks using different plugins to use the new images directly from for example, OpenNebula or OpenStack (see section 4 for more information).

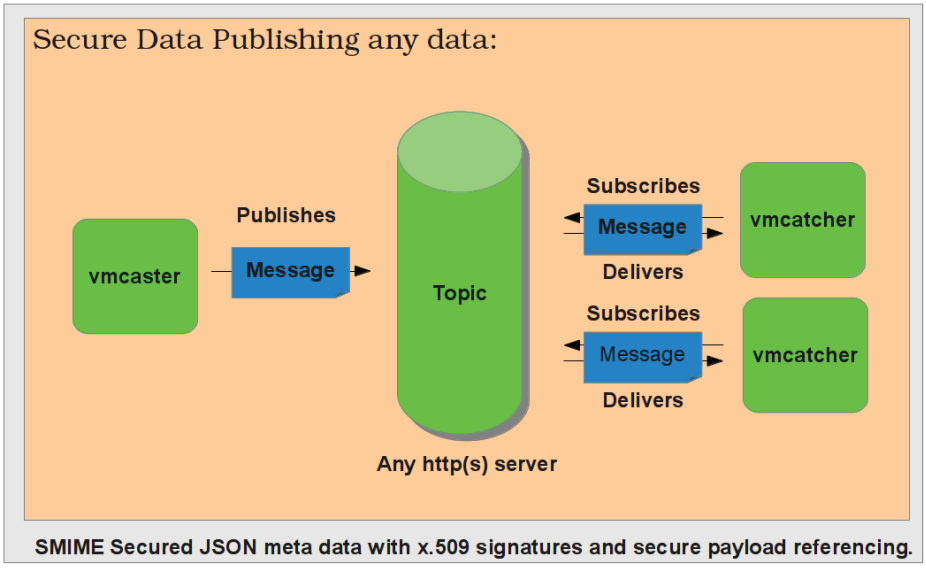


Fig. 1. VMcaster and VMcatcher infrastructure.

3.1 VMcaster

VMcaster is a simple tool for publishing, managing and updating virtual machines image lists which follows the HEPix image list specifications. All the image lists and metadata created by VMcaster are signed and trusted with a X.509 certificate. This provides a mechanism by which a virtual image can be checked for validity by any subscriber. Any user can check the image list expiration date, if it was revocated or it has been tampered by a third party. All images and image lists have an Universal(ly) Unique Identifier (UUID). These UUID's should be globally unique and consequently the UUID should be generated using a UUID generator using suitable seeds. As example for Debian, Redhat and Scientific Linux users can execute the following UUID generator:

```
$ uuidgen

dfc470ab-0845-4c3b-bc6a-02f990388a17
```

We can use the new UUID with VMcaster to create a empty image list:

```
$ vmcaster --select-imagelist dfc470ab-0845-4c3b-bc6a-02f990388a17 \
--add-imagelist
```

This command generates an empty image list in our local *vmcaster.db* database (not published yet) with a few predefined objects. We can query the image list ID to see the current object list. The database information is shown in JSON format:

```
$ vmcaster --select-imagelist dfc470ab-0845-4c3b-bc6a-02f990388a17 \
--show-imagelist
{
    "hv:imagelist": {
        "dc:identifier": "dfc470ab-0845-4c3b-bc6a-02f990388a17"
    }
}
```

The object *dc:identifier* contains always an UUID and it is included in images or image lists. At this moment the image list does not include any relevant information but thanks to VMcaster utility the image list endorser can introduce new objects and information. The most important objects are: the image title, its description, its endpoint (a valid *url*) and also the endorser unique certificate DN. Only a trusted endorser can modify or update an VMcaster image list to include or remove VM images. These values can be included into the internal database running these commands, as example:

```
$ vmcaster --select-imagelist <image_list_UUID> \
--key-set-imagelist "dc:title"\
--key-value-imagelist "New image list"

$ vmcaster --select-imagelist <image_list_UUID> \
```

```
--key-set-imagelist "dc:source" --key-value-imagelist "CESGA"

$ vmcaster --select-imagelist <image_list_UUID> \
--key-set-imagelist "dc:description"\
--key-value-imagelist "My image list for internal users"

$ vmcaster --select-imagelist <image_list_UUID> \
--key-set-imagelist "hv:uri" --key-value-imagelist \
"http://cloud.cesga.es/files/image.list"

$ vmcaster --select-endorser "/DC=es/DC=irisgrid/O=cesga/CN=alvarosimon" \
--key-set-endorser "dc:creator" --key-value-endorser "Alvaro Simon"
```

Endorsers can also import image lists (in JSON or MIME format) to streamline the image list creation. At this moment the endorser can include new images metadata into the new image list catalogue. Image insertion procedure is similar to image list creation *vmcaster –select-image ¡UUID¿ –key-set-image ¡IMAGE TAG¿ –key-value-image ¡VALUE¿*. The image list endorser only has to generate a new UUID and insert the relevant objects, in this case:

- *dc:title*: Image title name
- *sl:comments*: It includes image comments (user login and password, software included etc).
- *sl:osversion*: The Operating System version as LSB compliant. As example Scientific Linux release 6.4 (Carbon).
- *sl:arch*: System architecture (x86_64, i386 etc).
- *sl:os*: Operating System name (Ubuntu, Debian, RedHat..).
- *hv:uri*: Image location endpoint. The image must be accesible from this url. As example `http://cloud.cesga.es/images/debian-6.0.5-x86_64-base.qcow2`
- *hv:format*: VM image format (QCOW2, RAW)

All the new images should be assigned to a image list, we can include the same image in different image lists. To add a new image to a specific image list:

```
$ vmcaster --select-imagelist <IMAGE_LIST_UUID> --imagelist-add-image\
--select-image <IMAGE_UUID>
```

VMcaster can be configured to syncronize and upload local images to a specific server endpoint. This mechanism allows to VMcaster users to upload images and image lists to a web server in an automated way. This information is located into a configuration file (*/etc/vmcaster/vmcaster.cfg*). This mechanism is very useful for Image List endorsers as they can use this configuration file to include several servers to keep the image lists up to date in a short period of time. *vmcaster.cfg* file uses this schema:

```
[SERVER NAME]
server = "myserver.org"
```

```
protocol = "scp"
uriMatch = "https://myserver.org/"
uriReplace = "user@myserver.org:/var/www/html/"
```

In this case VMcaster will use the `myserver.org` web page to upload any image or image list. *vmcaster.cfg* accepts different communication protocols such scp, GSIdCap [16] or a local transmission. If this configuration file is set, VMcaster will upload images automatically using this command:

```
vmcaster --upload-image <local_image_path> --select-image <IMAGE_UUID>
```

VMcaster detects the image size and generates a SHA512 hash for each uploaded image, when this process is complete the updated information is included into the VMcaster database automatically.

At the end the information can be gathered from the local database, as example using JSON format image list:

```
{
    "hv:imagelist": {
        "dc:date:created": "2013-03-18T16:52:55Z",
        "dc:date:expires": "2014-04-15T16:52:55Z",
        "dc:description": "CESGA image list for internal usage",
        "dc:identifier": "2204eed5-f37e-45b9-82c6-85697356109c",
        "dc:source": "CESGA",
        "dc:title": "CESGA image list",
        "hv:endorser": {
            "hv:x509": {
                "dc:creator": "Alvaro Simon Garcia",
                "hv:ca": "/DC=es/DC=irisgrid/CN=IRISGridCA",
                "hv:dn": "/DC=es/DC=irisgrid/O=cesga/CN=alvarosimon",
                "hv:email": "asimon@cesga.es"
            }
        },
        "hv:images": [
            {
            {
                "hv:image": {
                    "dc:description": "UI-UMD3.0.0",
                    "dc:identifier":\
"9d6b140f-7a08-4f8c-8c25-3564bcb50e33",
                    "dc:title": "EMI-UI",
                    "hv:format": "QCOW2",
                    "hv:hypervisor": "QEMU,KVM",
                    "hv:uri":\
"http://cloud.cesga.es/images/test_ui_image.QCOW2",
                    "hv:version": "0.0.1",
                }
```

```
            }
        ],
        "hv:uri": "http://cloud.cesga.es/files/image.list",
        "hv:version": "2.9"
    }
}
```

By reading the image list example in JSON format, a future subscriber can identify the available images, the image list creation/expiration dates or the endorser DN.

When the image list is ready and updated the endorser can publish it to all sites included into the *vmcaster.cfg* file. The procedure is quite similar to the image uploading, but in this case, the image must be signed by the endorser certificate to validate its authenticity. VMcaster asks for user certificate password and uploads the image list to its final endpoint, this can be done with a single command:

```
$ vmcaster --select-imagelist <IMAGE_LIST_UUID> --upload-imagelist
```

This procedure allows different image endorsers to distribute and update image catalogs on different endpoints and sites (which is suitable for a federated architecture). Besides all image lists has endorsed information about endorser certificate public key, image download endpoint, initial global validity, etc. That means valid images can be selected, downloaded and instantiated and tested by different resource provides. The new images should be verified by the image list endorser but fedcloud resource provides have the final say about its inclusion into their Cloud image repositories. Image subscription task is done by VMcatcher utility described in the next section.

3.2 VMcatcher

VMcatcher utility allows image consumers to subscribe to VM image list generated by VMcaster. Using this utility users can select and download trusted images. This utility caches the selected images in a image list, validates the list with X.509 based public key cryptography, and also checks the images SHA512 hashes. Another important feature of VMcaster is that it provides events for further applications to process, update or expire changes of virtual machine images. These events can be used by event listeners or event handlers to react to specific changes (when a image is set as invalid or a new image update is available).

VMcatcher and VMcaster work in a similar way. They are based upon a local database that stores subscriptions to VM image lists, registered endorsers, and which images belong to which subscriptions. This enables images to be selected for subscription. An user can select the desired image from the lists and then set the desired image subscriptions. Subscribed images can be downloaded, verified and cached. VMcatcher also verifies if the cached images have expired or not. If an image is invalid or it has expired it is moved to an expiry directory. The image consumer must trust in a endorser, in this case users can include and confirm

their trust in a image endorser based on his/her X.509 certificate Distinguished Name (DN). To include a new trusted endorser:

```
$ vmcatcher_endorser --create --endorser_uuid='Alvaro Simon' \
--subject='/DC=es/DC=irisgrid/O=cesga/CN=alvarosimon' \
--issuer='/DC=es/DC=irisgrid/CN=IRISGridCA'
```

And now the user only has to download the desired image list from the endpoint and import it into a local VMcaster database:

```
$ wget http://cloud.cesga.es/files/image.list
$ vmcatcher_subscribe -s file:////`pwd`/image.list
```

At this point the image user can show and select any image UUID from the new list to be downloaded. The image list may vary (image endorse includes new images, revoke old ones, etc), in any case the local image list database should be updated frequently executing the command *vmcatcher_subscribe -U*. This command checks the image lists endoint and updates the local database accordingly. After image selection the images can be downloaded and updated to a local cache running:

```
$ vmcatcher_cache
```

The new downloaded images are stored into *¡VMCATCHER_DIR¿/cache* directory, meanwhile revoked or expired images are moved to *¡VMCATCHER_DIR¿-/cache/expired* directory. This mechanism prevents that old or revoked images are used by mistake. VMcatcher also raises some events if the image is downloaded, revoked or removed. These pre-defined events can be used by the Cloud frameworks to perform different actions.

4 Image management event handlers

VMcatcher and VMcaster are useful tools to disseminate and keep updated our images but they do not interact with Cloud frameworks directly. VMcatcher was written in Python and generates pre-defined events that can be received by an asynchronous callback subroutine or event handler.

Fortunately the Cloud community has developed event handlers to interact with the most popular frameworks like OpenNebula or CloudStack. OpenNebula event handler [17] was developed by the CESGA team and currently is available from the VMcaster repository. The *vmcatcher_eventHndlExpl_ON* package provides a new python and cron script which detects VMcatcher events. This script detects several VMcatcher event types. In this case OpenNebula handler only waits for a new Expire or Available VMcatcher event. If VMcatcher raises a *AvailablePostfix* event, this is detected by *vmcatcher_eventHndlExpl_ON* event handler and reads the new image attributes such UUID, image name, description and image format.

If a new image is downloaded (*AvailablePostfix* event), the OpenNebula event handler gets the image information, generates a new OpenNebula template and

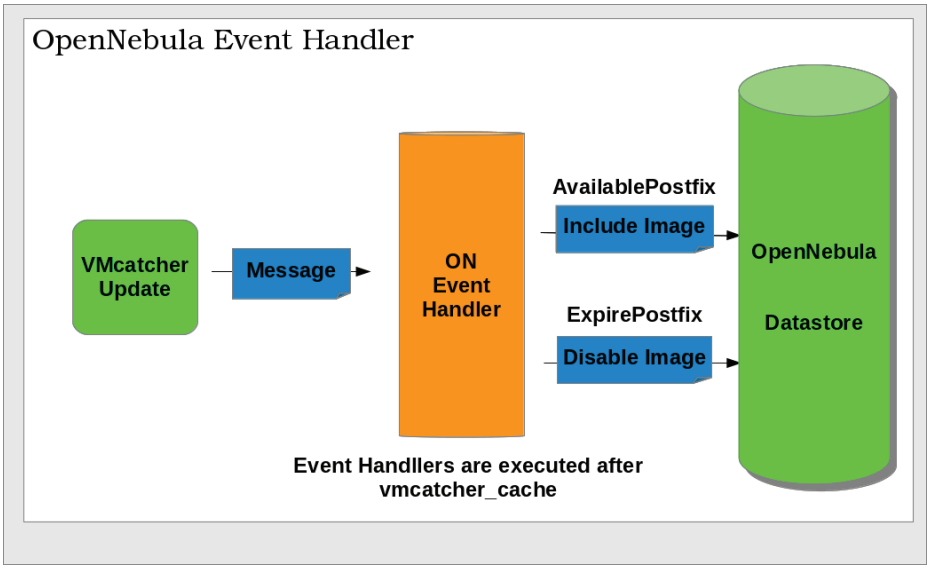


Fig. 2. OpenNebula event handler.

includes the new image into the local OpenNebula datastore (see figure 2). For security reasons, the new images are not public, they are only available for the oneadmin user. The OpenNebula administrator should verify the new image first (checking its contextualisation script, if the image is executed correctly etc).

After this period of time the image status is changed to be available for external users. Moreover if VMcatcher detects an image revocation the OpenNebula event handler searches the image UUID from OpenNebula image database and it is set to disable status. The image is not removed by the event handler, it should be removed by the site administrator from the OpenNebula datastore.

The OpenNebula event handler is not the only available one. OpenStack administrators can also use Glancepush [18] service to keep their local image catalog updated. This service was developed at IN2P3 and it works in a similar way than the OpenNebula event handler. In this case Glancepush updates the OpenStack Image Service (Glance) if it detects any image change from VMcatcher tool. The new package (*glancepush-vmcatcher*) is available from the IN2P3 ftp server, and it only requires a working glance service and an OpenStack user account to push images into the catalog.

5 Conclusions and Future work

The management of virtual machine images is a critical task within a federated Cloud architecture. It involves certain safety standards and special scalability and availability and stability patterns. Taking these requirements into account Fedcloud taks force have chosen VMcaster and VMcatcher utilities to

distribute and validate VM images between the resource providers. Fedcloud resource providers are using a heterogenous Cloud frameworks ecosystem (OpenStack, OpenNebula, WNoDES [19], etc). This kind of federated infrastructure requires agnostic tools. Fortunately as we have explained in this paper, VMcatcher can be used by any Cloud framework to distribute and update images in a transparent way. The new image management tools are being successfully used by EGI Fedcloud providers since last year and it will be used in more use cases in the near future.

A new use cases is the EGI SA2.3 verification image repository. The EGI SA2 testbed is used to verify and test the new middleware before reaching the production software repository (UMD). One of the most important new features is the ability to distribute and publish VM images in an automated way. These new tools, like EGI MarketPlace[3] and VMcatcher[4] can be also used within SA2 verification process to distribute and publish new UMD services after its verification. This is a still on going work and it will be available in the next months. Using this infrastructure the new EGI certified image it will be available to be used and tested by EGI site administrators after each successful verification. For the moment after each software verification the images are stored locally into CESGA OpenNebula datastore but thanks to image management tools like VMcaster the new images will be published in an automated way. This new paradigm will be useful to users that want to check the latest software changes without the need to install a new service from scratch.

Acknowledgements

This work is partially funded by the EGI-InSPIRE (European Grid Initiative: Integrated Sustainable Pan-European Infrastructure for Researchers in Europe) is a project co-funded by the European Commission (contract number INFSO-RI-261323) as an Integrated Infrastructure Initiative within the 7th Framework Programme. EGI-InSPIRE began in May 2010 and will run for 4 years. Full information is available at: `http://www.egi.eu/`.

References

1. Dillon, T., Wu, C., Chang, E.: Cloud computing: Issues and challenges. In: Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on. (2010) 27–33
2. Goasguen, S., Moreira, B., Roche, E., Schwickerath, U.: Lxcloud : a prototype for an internal cloud in hep. experiences and lessons learned. Journal of Physics: Conference Series **396**(3) (2012) 032098
3. Zhao, Y., Zhang, Y., Tian, W., Xue, R., Lin, C.: Designing and deploying a scientific computing cloud platform. In: Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on. (2012) 104–113

[3] EGI MarketPlace: `http://marketplace.egi.eu`

[4] VMcatcher: `https://github.com/hepix-virtualisation/vmcatcher`

4. Hoffa, C., Mehta, G., Freeman, T., Deelman, E., Keahey, K., Berriman, B., Good, J.: On the use of cloud computing for scientific workflows. In: eScience, 2008. eScience '08. IEEE Fourth International Conference on. (2008) 640–645
5. Wen, X., Gu, G., Li, Q., Gao, Y., Zhang, X.: Comparison of open-source cloud management platforms: Openstack and opennebula. In: Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on. (2012) 2457–2461
6. von Laszewski, G., Diaz, J., Wang, F., Fox, G.: Comparison of multiple cloud frameworks. In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. (2012) 734–741
7. Li, C., Raghunathan, A., Jha, N.: A trusted virtual machine in an untrusted management environment. Services Computing, IEEE Transactions on **5**(4) (2012) 472–483
8. Schwarzkopf, R., Schmidt, M., Strack, C., Martin, S., Freisleben, B.: Increasing virtual machine security in cloud environments. Journal of Cloud Computing **1**(1) (2012) 1–12
9. Lagar-Cavilla: Snowflock: rapid virtual machine cloning for cloud computing. In: Proceedings of the 4th ACM European conference on Computer systems. EuroSys '09, New York, NY, USA, ACM (2009) 1–12
10. Diaz, J., von Laszewski, G., Wang, F., Fox, G.: Abstract image management and universal image registration for Cloud and HPC infrastructures. In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. (2012) 463–470
11. Maurer, M., Brandic, I., Sakellariou, R.: Adaptive resource configuration for cloud infrastructure management. Future Generation Computer Systems **29**(2) (2013) 472 – 487
12. Armstrong, D., Espling, D., Tordsson, J., Djemame, K., Elmroth, E.: Runtime virtual machine recontextualization for clouds. In Caragiannis, I., Alexander, M., Badia, R., Cannataro, M., Costan, A., Danelutto, M., Desprez, F., Krammer, B., Sahuquillo, J., Scott, S., Weidendorfer, J., eds.: Euro-Par 2012: Parallel Processing Workshops. Volume 7640 of Lecture Notes in Computer Science., Springer Berlin Heidelberg (2013) 567–576
13. Synge, O.: Vmcaster vm image publication tool. `https://github.com/hepix-virtualisation/vmcaster` (May 2013)
14. Synge, O.: Vmcatcher image subscription tool. `https://github.com/hepix-virtualisation/vmcatcher` (May 2013)
15. Cass, T.: The hepix virtualisation working group: Towards a grid of clouds. Journal of Physics: Conference Series **396**(3) (2012) 032020
16. Riese, M.D.: The dcache book. `http://www.dcache.org/manuals/Book/` (May 2013)
17. Rosende, R.: Opennebula event handler. `https://github.com/grid-admin/vmcatcher_eventHndlExpl_ON` (May 2013)
18. Puel, M.: Openstack glancepush service. `https://github.com/EGI-FCTF/glancepush/wiki` (2013)
19. Salomoni, D., Italiano, A., Ronchieri, E.: Wnodes, a tool for integrated grid and cloud access and computing farm virtualization. Journal of Physics: Conference Series **331** (2011) 052017

Infrastructure-Agnostic Programming and Interoperable Execution in Heterogeneous Grids

Enric Tejedor[1], Javier Álvarez[1], Rosa M. Badia[1,2]

[1] Barcelona Supercomputing Center (BSC-CNS)
Jordi Girona 29, 08034 Barcelona (Spain)
enric.tejedor@bsc.es, javier.alvarez@bsc.es, rosa.m.badia@bsc.es
[2] Artificial Intelligence Research Institute (IIIA),
Spanish Council for Scientific Research (CSIC)
E-08193 Bellaterra, Barcelona (Spain)

Abstract. In distributed environments, no matter the type of infrastructure (cluster, grid, cloud), portability of applications and interoperability are always a major concern. Such infrastructures have a high variety of characteristics, which brings a need for systems that abstract the application from the particular details of each infrastructure. In addition, managing parallelisation and distribution also complicates the work of the programmer.
In that sense, this paper demonstrates how an e-Science application can be easily developed with the COMPSs programming model and then parallelised in heterogeneous grids with the COMPSs runtime. With COMPSs, programs are developed in a totally-sequential way, while the user is only responsible for specifying their tasks, i.e. computations to be spawned asynchronously to the available resources. The COMPSs runtime deals with parallelisation and infrastructure management, so that the application is portable and agnostic of the underlying infrastructure.

1 Introduction

In distributed environments, no matter the type of infrastructure (cluster, grid, cloud), portability of applications and interoperability are always a major concern [3, 2]. Different infrastructures can have very diverse characteristics. Besides, even in the scope of a given infrastructure, there is typically a plethora of alternatives to implement and execute an application, and often several vendors compete to dominate the market. Choosing one of the alternatives usually ties the application to it, e.g. due to the use of a certain API. As a result, it may be hard to port the application, not only to another kind of infrastructure, but also to an equivalent platform provided by another vendor or managed by different software.

Standards do appear, either 'de facto' or produced by collaborative organisations that develop them, as in the case of the Open Grid Forum [6], but it is often complicated for them to be widely accepted. This situation, which is likely

to keep happening in future scenarios, increases the importance of systems that free the user from porting the same application over different platforms.

On the other hand, some of the difficulties of programming applications for distributed infrastructures are not related to their particular characteristics, but to the duty of parallelisation and distribution itself [16]. This includes aspects like thread creation and synchronisation, messaging, data partitioning and transfer, etc. Having to deal with such aspects can significantly complicate the work of the programmer as well.

In that sense, this paper demonstrates how the COMPSs programming model and runtime system can be used to easily develop and parallelise applications in distributed infrastructures. More precisely, we discuss an example of an e-Science application that was programmed with the COMPSs model. Such application does not include any API call, deployment or resource management detail that could tie it to a certain platform. In addition, the application is programmed in a fully-sequential fashion, freeing the programmer from having to explicitly manage parallelisation and distribution.

Furthermore, we present some experiments that execute that application in large-scale heterogeneous grids controlled by different types of middleware. A runtime is responsible for hiding that heterogeneity to the programmer, interacting with the grids and making them interoperable to execute the application. Consequently, the application remains agnostic of the underlying infrastructure, which favours portability.

The paper is structured as follows. Section 2 provides an overview of the COMPSs programming model and runtime system. Section 3 introduces the use-case e-Science application. Section 4 describes the Grid testbed used in the experiments. Section 5 presents the results of the experiments. Finally, Section 2 discusses some related work and Section 6 concludes the paper.

2 Overview of COMP Superscalar

This section introduces the COMP Superscalar (COMPSs) programming model, as well as the runtime system that supports the model's features. COMPSs is tailored for Java applications running on distributed platforms like clusters, grids and clouds. For a more detailed description of COMPSs, please see [20, 22, 21].

2.1 Programming Model

The COMPSs programming model can be defined as task-based and dependency-aware. In COMPSs, the programmer is only required to select a set of methods and/or services called from a sequential Java application, for them to be run as *tasks* - asynchronous computations - on the available distributed resources.

The task selection is done by providing a Task Selection Interface (TSI), a Java interface which declares those methods/services, along with some metadata. Part of these metadata specifies the direction (input, output or in-out) of each task parameter; this is used to discover, at execution time, the data dependencies

between tasks. The TSI is not a part of the application: it is completely separated from the application code and it is not implemented by any of the user's classes; its purpose is merely specifying the tasks.

With COMPSs, sequential Java applications can be parallelised with no modifications: the application code does not contain any parallel construct, API call or pragma. All the information needed for parallelization is contained in the TSI. Besides, the application is not tied to a particular infrastructure: it does not include any resource management or deployment information.

2.2 Runtime System

The runtime system receives as input the class files corresponding to the sequential code of the application and the TSI. Before executing the application, the runtime transforms it into a modified bytecode that can be parallelised. In particular, the invocations of the user-selected methods/services are automatically replaced by an invocation to the runtime: such invocation will create an asynchronous task and let the main program continue its execution right away.

The created tasks are processed by the runtime, which dynamically discovers the dependencies between them, building a task dependency graph. The parallelism exhibited by the graph is exploited as much as possible, scheduling the dependency-free tasks on the available resources. The scheduling is locality-aware: nodes can cache task data for later use, and a node that already has some or all the input data for a task gets more chances to run it.

The interaction of the runtime with the infrastructure is done through JavaGAT [11], which offers a uniform API to access different kinds of Grid middleware. COMPSs uses JavaGAT for two main purposes: submitting tasks and transferring files to Grid resources. Thus, the runtime is responsible for transferring task data and managing task execution through JavaGAT, while the application is totally unaware of such details.

3 The SimDynamics Application

The SimDynamics application, which will be used in the experiments presented in Section 5, is a sequential Java program that makes use of DISCRETE [9], a package devised to simulate the dynamics of proteins using the Discrete Molecular Dynamics (DMD) methods.

Starting from a set of protein structures, the objective of SimDynamics is to find the values of three parameters that minimise the overall energy obtained when simulating their molecular dynamics with DISCRETE. Hence, SimDynamics is an example of a parameter-sweeping application: for each parameter, a fixed number of values within a range is considered and a set of simulations (one per structure) is performed for each combination of these values (configuration). Once all the simulations for a specific configuration have completed, the configuration's score is calculated and later compared to the others in order to find the best one.

In order to run SimDynamics with COMPSs, a total of six methods invoked from the application were chosen as tasks. This was done by defining a TSI that declares those methods. Figure 1 contains a fragment of this TSI, more precisely the selection of method simulate as a task. The parameters of simulate are three input files, an input string and an output file. The declarations of the other five methods are analogous to this one.

```
public interface SimDynamicsItf {

    @Method(declaringClass = "simdynamics.SimDynamicsImpl")
    void simulate(
        @Parameter(type = FILE) String paramFile,
        @Parameter(type = FILE) String topFile,
        @Parameter(type = FILE) String crdFile,
        String natom,
        @Parameter(type = FILE, direction = OUT) String average
    );

    ...

}
```

Fig. 1. Code snippet of the Task Selection Interface for the SimDynamics application, where the simulate method is selected as a task. The @Method annotation specifies the class that implements simulate, and the @Parameter annotation contains parameter-related metadata (type, direction).

4 Testbed Infrastructure

The SimDynamics application was executed with COMPSs on real large-scale scientific grids. The whole infrastructure used in the tests is depicted in Figure 2, and it includes three grids: the Open Science Grid, Ibergrid and a small grid owned by the Barcelona Supercomputing Center [1].

Such infrastructure represents an heterogeneous testbed, comprised by three grids belonging to different administrative domains and managed by different middleware. The next subsections briefly describe the topology of these grids and explain how the COMPSs runtime was able to hide the complexity of their heterogeneity, keeping the Grid-related details transparent to the application.

4.1 Grids

Open Science Grid Each of the Open Science Grid (OSG) [8] sites is configured to deploy a set of Grid services, like user authorisation, job submission and storage management. Basically, a site is organised in a *Compute Element* (CE), running in a front-end node known as the *gatekeeper*, plus several *worker nodes* (or execution nodes). The CE allows users to run jobs on a site by means

of the Globus GRAM (Grid Resource Allocation Manager) [14] interface; at the back-end of this GRAM gatekeeper, each site features one or more local batch systems - like Condor [23] or PBS [7] - that process a queue of jobs and schedule them on the worker nodes. Besides, the standard CE installation includes a GridFTP server; typically, the files uploaded to this server are accessible from all the nodes of the site via a distributed file system like NFS (Network File System [5]).

Ibergrid Similarly to OSG, the Ibergrid infrastructure [10, 4] is composed by different sites, each one with a gatekeeper node interfacing to the cluster, a local resource management system (batch) and a set of worker nodes. However, in Ibergrid the middleware installed is gLite [15] and job management is a bit different: instead of submitting the jobs to a given CE directly, the user proceeds by interacting with a *Workload Management Server* (WMS), which acts as a meta-scheduling server. Therefore, matchmaking is performed at a higher level: the WMS interrogates the Information Supermarket (an internal cache of information) to determine the status of computational and storage resources, and the File Catalogue to find the location of any required input files; based on that information, the WMS selects a CE where to execute the job.

BSC Grid The BSC Grid is a cluster located in the BSC premises and formed by five nodes. Three of them have a single-core processor at 3.60GHz, 1 GB of RAM and 60 GB of storage. The other two have a quad-core processor at 2.50GHz each core, 4 GB of RAM and 260 GB of storage. The BSC Grid supports interactive execution: the user can connect to any of the nodes separately via SSH and launch computations on them. Moreover, files can be transferred to/from the local disk of each node through SSH as well.

4.2 Configuration and Operation Details

In order to run the SimDynamics application in the described testbed, the testing environment was configured as shown in Figure 2.

The access point to the Grid was a laptop equipped with a dual-core 2.8 GHz processor and 8 GB RAM. This machine hosted the main program of the application, and therefore it had the COMPSs runtime and the JavaGAT library and adaptors installed. In addition, prior to the execution, the credentials for each grid were obtained and installed as well.

Concerning the Grid middleware, the points below list the GAT adaptors and the corresponding grids where they were used:

- *Globus GRAM and OSG*: a total of six OSG sites that support our virtual organisation (VO), Engage, were used in the tests, each with its own CE. The gatekeeper of every CE was contacted by means of the Globus GRAM adaptor, used for task submission and monitoring in OSG.

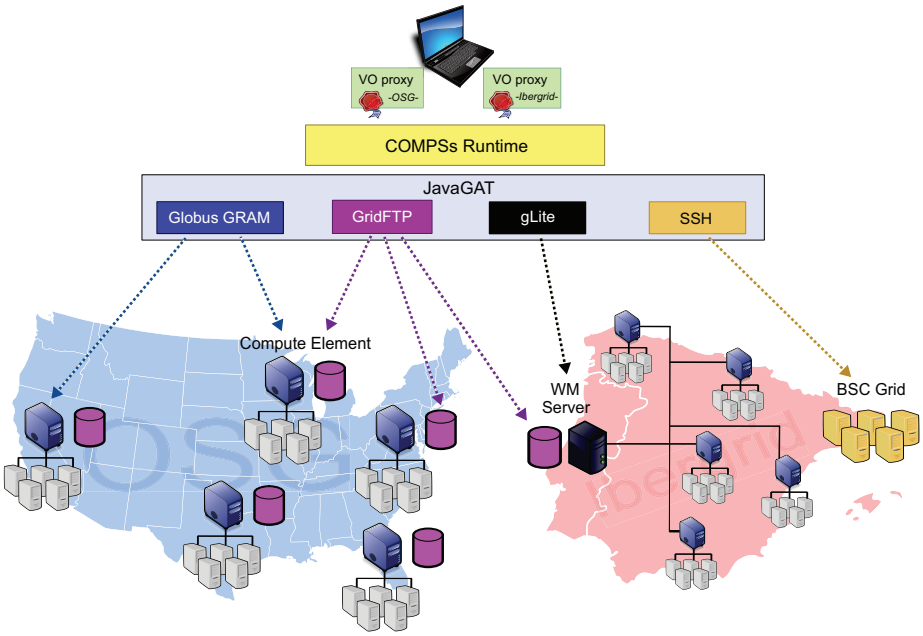


Fig. 2. Testbed comprising two large-scale scientific grids (Open Science Grid, Ibergrid) and a BSC-owned grid. The SimDynamics application, running on a machine with COMPSs, interacts with the grids through JavaGAT and its middleware adaptors.

- *gLite and Ibergrid*: the gLite adaptor was used to submit and monitor tasks by connecting to an Ibergrid WMS, which is in charge of selecting the execution site in Ibergrid. Among all the WMS at the disposal of our VO (ICT), the one with most availability was chosen.
- *GridFTP (OSG and Ibergrid)*: the SOG CEs and the Ibergrid WMS offer each a GridFTP server. The GAT GridFTP adaptor was used to transfer files to those servers during execution.
- *SSH and BSC Grid*: two nodes of BSC Grid were used in the tests, being accessed through the GAT SSH adaptors for task submission and file transfer.

Before execution, there was a previous phase of deployment where some required files were installed in the grids: the worker runtime and the classes and executables of the application tasks. In OSG, the files to be deployed were copied to the GridFTP server of each CE, so they could be accessed from the worker nodes. In Ibergrid, the files were transferred to the GridFTP server of the WMS, since the final execution site is not known in advance in this scenario; each time a job is created in Ibergrid, those files are copied by the worker runtime from the GridFTP server to the site where the job will run. Finally, in BSC Grid the files were placed in the local disk of the nodes.

At execution time, the master runtime of COMPSs sends the SimDynamics tasks and transfers files to the three grids by means of JavaGAT. In OSG, the input files of each task are first pre-staged to the GridFTP server of the target CE, thus being accessible through the NFS server of that CE too; after that, when the job is created in the CE to execute the task, the worker runtime copies the input files from NFS to the local disk of the target worker node; similarly, the output files are copied from local to NFS at the end of the task, thus being available in the GridFTP server as well. In Ibergrid, the task input files are transferred to the GridFTP server of the WMS; the pre and post-staging of those files to/from the final worker node is taken care by gLite: the WMS chooses the execution site, sends the job to the head node of that site, then the task is locally scheduled and the input files are copied from the GridFTP server to the local disk of the worker node (the process is inverse for the output files). Lastly, the BSC Grid scenario is simpler since the files can be directly transferred to/from the local disk of the final execution node.

When scheduling tasks on the grids, the COMPSs runtime takes into account locality: a task will be assigned, if possible, to a resource that already possesses one or more of the task's input files (in its GridFTP server or local disk). Whenever a resource is freed (a task finishes), the scheduler chooses the task with the best score among the pending ones, the score being the number of task input files in the resource. Note that Ibergrid counts as a single entity for locality, because the final destination of the job is not decided by COMPSs. If some input file is missing in the chosen resource, such file is replicated to that resource. If the source and destination resources share the same credentials (e.g. two OSG sites) such transfer happens directly between them; otherwise, the file is first copied to the laptop and then to the destination resource.

5 Evaluation

This section presents the results of executing the SimDynamics application (Section 3) in the described testbed (Section 4). These tests will show how the tasks of an e-Science application are executed in three different grids with COMPSs.

From the point of view of the application, *all the Grid management* discussed in Section 4 *is transparent*. The application deals with its parameters, like number of structures and coefficients. For these experiments, 27 different configurations were considered in the parameter sweeping. This leads to a total of 586 tasks, including 270 simulation tasks - the most computationally-intensive with about two minutes of execution time each. The rest of the tasks are lightweight, with a duration of less than 10 seconds.

Figure 3(a) shows how tasks were distributed among the three grids during an execution of SimDynamics with COMPSs. The six OSG resources were the ones that consumed more tasks; indeed, among all the OSG sites that support our VO, the ones with most availability were chosen. The two BSC Grid nodes also executed a considerable number of tasks because they are directly accessible and therefore those tasks did not suffer from queue waiting times. Ibergrid received

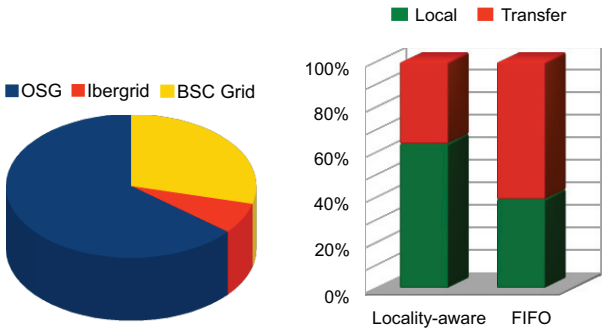


Fig. 3. Test results for the SimDynamics application when run with COMPSs in the Grid testbed: (a) distribution of the SimDynamics tasks among the three grids; (b) comparison of percentage of transfers between the locality-aware and FIFO scheduling algorithms.

Table 1. Task submission and file transfer statistics for SimDynamics.

		# Task sub.		# File tra.	
Grid	**Resource**	*OK*	*Failed*	*OK*	*Failed*
OSG	brgw1.renci.org	72	4	102	1
	gridgk01.racf.bnl.gov	43	0	70	1
	rossmann-osg.rcac.purdue.edu	57	14	89	11
	smufarm.physics.smu.edu	69	1	92	1
	stargrid02.rcf.bnl.gov	55	0	90	1
	u2-grid.ccr.buffalo.edu	62	1	96	0
	TOTAL	358	20	539	15
Ibergrid	wms01.ific.uv.es	33	209	58	0
	TOTAL	33	209	58	0
BSC Grid	bscgrid05.bsc.es	122	0	116	0
	bscgrid06.bsc.es	73	0	79	0
	TOTAL	195	0	195	0
	TOTAL	586	229	792	15

less load because of three factors. First, the Ibergrid queue times in these tests were high, which caused tasks scheduled in Ibergrid to wait. Second, regarding the internal scheduling policies of the Ibergrid sites, several sites offer to our VO only opportunistic access to their resources; some other sites reserve a certain number of slots with priority but they are shared by all the Ibergrid VOs. Finally, the errors when submitting tasks to the WMS were quite frequent, which made tasks go through a (sometimes long) resubmission process.

In that sense, Table 1 contains the statistics of errors in task submissions and file transfers for the different grids and a particularly faulty execution of

```
public interface SimDynamicsItf {
    @Constraints(operatingSystem = "Scientific Linux")
    @Method(...)
    void genReceptorLigand(...);

    @Constraints(appSoftware = "DISCRETE")
    @Method(...)
    void simulate(...);

    @Constraints(memory = 4)
    @Method(...)
    void evaluate(...);

    ...
}
```

Fig. 4. Detail of the task constraint specification in the TSI of SimDynamics.

SimDynamics, in order to demonstrate the fault tolerance mechanisms of the COMPSs runtime. In general, the OSG sites presented only occasional failures in task submissions and file transfers, which were easily solved with resubmissions and retransfers with no need for task rescheduling. On the contrary, the errors when connecting to the Ibergrid WMS were common, possibly because of a bug in the JavaGAT gLite adaptor or because of the WMS itself; in order to face that issue, several retries were attempted when necessary for a task (6 per task on average), progressively increasing the time between two resubmissions. The most reliable combination of grid/adaptor was BSC Grid/SSH, for which no errors of any kind were registered.

Regarding data locality, Figure 3(b) illustrates the benefits of using a locality-aware task scheduling algorithm. Such algorithm is especially important in a highly-distributed testbed like the one in Figure 2, where data transfers are costly. Figure 3(b) compares two executions of SimDynamics, one using locality-aware scheduling and another one applying a FIFO (First In First Out) strategy, and it shows the percentage of transfers actually performed versus the percentage of locality (the transfer was not necessary because the input file was already on the target execution resource), the total being the number of input files of all tasks. The locality-aware algorithm achieved remarkable results, preventing almost 2 out of every 3 transfers.

A final series of tests intended to demonstrate how to use constraints to force the scheduling of tasks on certain resources, in case those tasks have some hardware/software requirements. Let us assume that each kind of task in SimDynamics has some resource requirements; Figure 4 shows how they can be specified in COMPSs by means of the @Constraints annotation, at method level, in the TSI. In this example, genReceptorLigand must be executed in nodes running Scientific Linux, which is the operating system installed in Ibergrid. Second, simulate is supposed to run in resources where the DISCRETE software is present; here, such capability was assigned only to OSG sites. Finally, evaluate has a hardware constraint attached - more precisely, the amount of physical memory - which

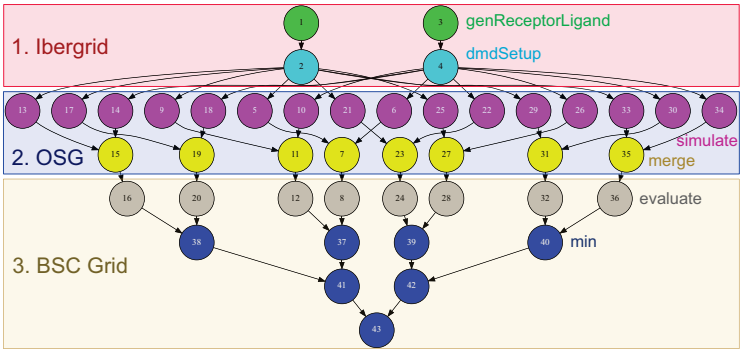


Fig. 5. Reduced version of the SimDynamics graph (the real one contains 586 tasks). The constraints in Figure 4 lead to the task scheduling on the grids represented by this figure.

was only known and specified in the resources file for the BSC Grid nodes. The three other kinds of task not shown in Figure 4 have analogous constraints.

As a result of the constraints, at execution time the scheduling of tasks on resources was the one depicted in Figure 5. This graph is a smaller version (only 8 configurations) just for illustration purposes. In conclusion, the programmer can use task constraints to make sure that a given group of tasks will be executed in one or more resources that conform to a set of requirements.

6 Related Work

Apart from COMPSs, there exist other programming models for Grid applications. Ninf-G [18] offers a programming model where client programs can call libraries on remote resources using a client API that is built on top of the Globus Toolkit. Ninf-G's model is more complex than COMPSs', since the programmer has to substantially modify the original application code by including the invocations to the GridRPC API. Furthermore, COMPSs can submit tasks using different kinds of Grid middleware. Satin [24] permits to express divide-and-conquer parallelism in Java applications, marking method invocations for asynchronous spawning. Nevertheless, the programmer must explicitly use a synchronisation primitive to wait for the spawned tasks; unlike Satin, COMPSs takes care of task and data synchronisation automatically, and it is not restricted to the divide-and-conquer paradigm. OpenWP [12] is a Grid programming and runtime environment with a set of directives that have to be included in the application code to express parallelism and distribution. The main difference between COMPSs and OpenWP is that the latter requires to indicate the dependencies between tasks in the application code, whereas the former finds them automatically at execution time.

With respect to workflow managers, some systems have been proposed to specify the elements of a workflow and the connections between them, either graphically or by means of a high-level workflow description language; in this sense they differ from COMPSs, where the workflow graph is implicitly defined by a concrete execution of an application and built automatically and dynamically at runtime. Taverna [17] is a well-known graphical tool for designing and executing Grid workflows. A Taverna workflow is specified by a directed acyclic graph where nodes represent software components. Each edge in the graph denotes a data dependency from an output port of the source node to an input port of the destination node. The nodes of a Taverna workflow can be computations executed in the Grid and also Web Services, similarly to COMPSs. Triana [19] also permits to describe applications by dragging and dropping their components and connecting them together to build a workflow graph; like in COMPSs, Triana workflows can access the Grid through JavaGAT. Pegasus [13] is a workflow management system that takes high-level workflow descriptions and automatically maps them to Grid resources; Pegasus performs execution site selection, manages the input data and provides directives for data transfer and registration.

7 Conclusions and Future Work

This paper has shown how an e-Science application can be easily developed with the COMPSs programming model and then parallelised in heterogeneous grids with the COMPSs runtime. Such application is programmed sequentially, while the user is only responsible for specifying its tasks. No API call or resource management details appears in the application, so that it is portable and agnostic of the underlying infrastructure. All the burden of parallelisation and infrastructure management is left to the COMPSs runtime; this paper has demonstrated how this runtime can deal with grids managed by different middleware, making them interoperable while keeping the application unaware of Grid details.

The future work includes supporting the use of logical files in COMPSs executions, possibly by creating a JavaGAT adaptor that manages them; such files are referenced with logical names that can be associated to several physical locations. Furthermore, we plan to extend the locality-aware algorithm to take into account not only the number of input files but also their size when deciding the target resource of a task.

Acknowledgements

This work has been supported by the following institutions: the Universitat Politècnica de Catalunya with a UPC Recerca predoctoral grant; the projects of Computación de Altas Prestaciones V and VI (TIN2007-60625, TIN2012-34557); the Spanish Government with grant SEV-2011-00067 of Severo Ochoa Program. On the other hand, the Ibergrid and the Open Science Grid organisations have granted us access to their infrastructures.

References

1. Barcelona Supercomputing Center. http://www.bsc.es.
2. Cloud interoperability and portability remain science fiction. http://searchcloudcomputing.techtarget.com/feature/Cloud-interoperability-and-portability-remain-science-fiction.
3. Grid Interoperation Now Community Group (GIN-CG). http://www.ogf.org/gf/group_info/view.php?group=gin-cg.
4. Iniciativa Nacional Grid. http://www.gridcomputing.pt.
5. Network File System. http://www.ietf.org/rfc/rfc3010.
6. Open Grid Forum. http://www.gridforum.org/.
7. Open Portable Batch System. http://www.openpbs.org/.
8. Open Science Grid. http://www.opensciencegrid.org.
9. ScalaLife Pilot Applications - DISCRETE. http://www.scalalife.eu/applications.
10. Spanish National Grid Initiative. http://www.es-ngi.es/.
11. G. Allen, K. Davis, T. Goodale, A. Hutanu, H. Kaiser, T. Kielmann, A. Merzky, R. van Nieuwpoort, A. Reinefeld, F. Schintke, T. Schütt, E. Seidel, and B. Ullmer. The Grid Application Toolkit: Towards Generic and Easy Application Programming Interfaces for the Grid. In *Proceedings of the IEEE*, volume 93, pages 534–550, 2005.
12. M. Cargnelli, G. Alleon, and F. Cappello. OpenWP: Combining annotation language and workflow environments for porting existing applications on grids. In *Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing*, GRID '08, pages 176–183, Washington, DC, USA, 2008. IEEE Computer Society.
13. E. Deelman, G. Singh, M. hui Su, J. Blythe, A. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming Journal*, 13:219–237, 2005.
14. I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *Int. Journal of Supercomputer Applications*, 11(2):115–128, 1997.
15. E. Laure, C. Gr, S. Fisher, A. Frohner, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, M. Barroso, P. Buncic, R. Byrom, L. Cornwall, M. Craig, A. D. Meglio, A. Djaoui, F. Giacomini, J. Hahkala, F. Hemmer, S. Hicks, A. Edlund, A. Maraschini, R. Middleton, M. Sgaravatto, M. Steenbakkers, J. Walk, and A. Wilson. Programming the Grid with gLite. In *Computational Methods in Science and Technology*, page 2006, 2006.
16. P. E. McKenney. *Is Parallel Programming Hard, And, If So, What Can You Do About It?* kernel.org, Corvallis, OR, USA, 2012. Available: http://kernel.org/pub/linux/kernel/people/paulmck/perfbook/perfbook.html.
17. P. Missier, S. Soiland-Reyes, S. Owen, W. Tan, A. Nenadic, I. Dunlop, A. Williams, T. Oinn, and C. Goble. Taverna, reloaded. In M. Gertz, T. Hey, and B. Ludaescher, editors, *SSDBM 2010*, Heidelberg, Germany, June 2010.
18. Y. Tanaka, H. Nakada, S. Sekiguchi, T. Suzumura, and S. Matsuoka. Ninf-G: A Reference Implementation of RPC-based Programming Middleware for Grid Computing. *Journal of Grid Computing*, 1(1):41–51, 2003.
19. I. Taylor, M. Shields, I. Wang, and A. Harrison. Visual Grid Workflow in Triana. *Journal of Grid Computing*, 3(3-4):153–169, September 2005.

20. E. Tejedor and R. M. Badia. COMP Superscalar: Bringing GRID Superscalar and GCM Together. In *Eighth IEEE International Symposium on Cluster Computing and the Grid*, CCGrid '08, Lyon, France, pages 185–193, May 2008.
21. E. Tejedor, J. Ejarque, F. Lordan, R. Rafanell, J. Álvarez, D. Lezzi, R. Sirvent, and R. M. Badia. A Cloud-unaware Programming Model for Easy Development of Composite Services. In *3rd IEEE International Conference on Cloud Computing Technology and Science*, CloudCom '11, Athens, Greece, November 2011.
22. E. Tejedor, M. Farreras, D. Grove, R. M. Badia, G. Almasi, and J. Labarta. A high-productivity task-based programming model for clusters. *Concurrency and Computation: Practice and Experience*, 24(18):2421–2448, 2012.
23. D. Thain, T. Tannenbaum, and M. Livny. Condor and the Grid. In F. Berman, G. Fox, and T. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons Inc., December 2002.
24. R. V. van Nieuwpoort, G. Wrzesińska, C. J. Jacobs, and H. E. Bal. Satin: A high-level and efficient grid programming model. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 32(3):1–39, 2010.

Session III

Advanced Data Processing

Graph Database for Structured Radiology Reporting

Lorenzo Díaz[1], Damià Segrelles[1], Erik Torres[1], Ignacio Blanquer[1]

Instituto de Instrumentación para Imagen Molecular (I3M), Universitat Politècnica de València, València, Spain
lodiade@upv.es, dquilis@dsic.upv.es, ertorser@upv.es, iblanque@dsic.upv.es

Abstract. The scalability of the NoSQL database model and the document-centric data structure of XML databases appear to be promising features for effective clinical data management and research. This paper presents a practical implementation of a new data model for DICOM SR-based structured reports, which uses a graph database as the underlying data storage. The new data model is integrated into TRENCADIS, a framework that offers unified access to medical applications built on top of Grid and Cloud infrastructures. The graph data model is compared with the currently used database, which is based on AMGA. The new model facilitates the assimilation of new types of reports and improves the performance of insertion. However, further research is needed to improve querying time.

1 Introduction

Understanding the reasoning of experts is an essential prerequisite to further automation of the process of diagnosis and treatment of cancer. Success in this endeavor will require the correlation with previous studies and interpretation of these findings. However, the landscape of medical practice is dominated by the use of proprietary image archiving and retrieval systems, the majority of them are incompatible with modern trends of information processing. Among the causes of this lack of compatibility are the increase of interest in medical imaging for research and teaching, and the advent of revolutionary information and communication technologies.

Picture Archiving and Communication Systems (PACS) were introduced in the decade of the 80s to provide economical storage of medical images with easy access of radiologists to images [1] [2]. However, in recent years, the focus of medicine has shifted toward a global, integrated view of disease processes that acknowledges the role of other factors that might influence the diagnostic and treatment of diseases, such as cellular mechanisms, epidemiological distribution or psychosocial context [3] [4] [5]. This has led to a change in the architecture and the functional scope of PACS to make them more useful to the entire organization (not only the radiology department), and therefore supporting interoperability standards is becoming more necessary in order to enable the new generation of PACS for cooperation [6].

Simultaneously, new and improved databases and tools have emerged that are vastly increasing our ability to analyze large amounts of information with very complex relationships structures. "Big data" has become a major force of innovation in many areas of technologies applicable to medical imaging applications, reshaping the research agenda and stimulating the development of innovative computational methods to mine this information [7] [8]. Imaging informatics and PACS are aware of the advantages offered by these new methods, but they cannot react fast enough to adequate their practices and to benefit from the apparition of these opportunities [9]. While new emerging technologies challenge the established historical approaches, they also raise important issues about the security of the patient information stored in PACS, RIS (Radiology Information System) and HIS (Hospital Information System). Also, the role of open-source in providing such innovative and challenging tools (that, in many cases, respond to users demands even before industry and commercial vendors) is questioned for an extended use in hospitals because the software produced in this way do not follow the traditional conformance and certification required for commercial medical software.

Researchers in computational sciences are embracing an alternative to overcome this current limitation to the use of modern computational methods with PACS that consists on pseudonymising the information to store them with virtual repositories, based on Grid and Cloud computing technologies [10] [11] [12]. These storages are optimized to work with different programming models for processing large, complex data sets using parallel or distributed algorithms.

NoSQL is one of the new technologies that is receiving substantial attention from researchers in medical informatics. In particular, the scalability of the NoSQL database model and the document-centric data structure of XML databases appear to be promising features for effective clinical data management and research [13, 14, 15].

This paper presents a practical implementation of a novel data model for structured reports. Digital Imaging and Communications in Medicine (DICOM) is a standard for managing digital images that is generally used in clinical practice. Structured Reporting (SR) extends DICOM to share documentary information [16]. Furthermore, DICOM SR has proven to be particularly valuable in improving the expressiveness, precision and compatibility of documentation about diagnostic images [17].

The new data model is integrated into the TRENCADIS framework, which provides access to virtualized storages of structured reports, built on top of Grid and Cloud infrastructures [18]. These storages were designed to complement traditional medical imaging storages, such as PACS and RIS, with flexible indexing, browsing and searching capabilities. The main target of TRENCADIS is to support research and training that is conducted using medical images. However, often these activities require changes in the structure of the information (for example, to use data-mining methods or to evaluate new clinical procedures). In these cases, a flexible model for storing the data items, along with their relationships, may be more convenient than a traditional approach.

This paper also presents the results of an experiment conducted to compare the new data model presented in this paper with the previous one, which is based on AMGA [19]. The experiment evaluates both approaches in terms of flexibility to assimilate new data types and relationships in the storages and the performance of insertions and searches.

The rest of the paper is organized as follows: Section 2 describes how structured reports are implemented in TRENCADIS. Section 3 briefly describes the currently used data model, which is based on AMGA, and presents the new graph data model. Section 4 describes the experiments conducted to evaluate the new model. Finally, concluding remarks and future lines of work are presented.

2 Structured Reporting

TRENCADIS extracts the metadata contained in the DICOM headers and uses it to organize medical images in virtual repositories – which are independent from the PACS where the images are actually stored –. The reports are encoded with DICOM SR, a medical standard that extends DICOM with a means of encoding structured information, allowing applications to share documentary information [16]. The format of the reports is defined by an SR Template, which is a mechanism defined in the DICOM SR standard for establishing patterns of applications [20]. These patterns describe and constrain the information that can be represented through a SR document. For example, the Supplement 50 to the DICOM standard defines the computer-aided detection templates for mammography. However, only a few SR templates are available covering a particular application or medical procedure. Therefore, TRENCADIS uses an additional set of templates that have been defined by consensus by a group of radiologists from different hospitals, who are involved in the development of TRENCADIS.

The SR Templates defined in this way use coded entries rather than free-text fields to facilitate indexing and searching in TRENCADIS. Indeed, DICOM SR allows to combine different standardized terminologies and ontologies into a document, and even to use custom codification schemes for organizing the structured reports. Depending on the application area, different standards can be used. In the reports, the semantic relationships are organized by means of tree-like structures. The meaning of each tree node is identified by a concept name (CONCEPT_NAME) associated to a given standard terminology like RadLex [21], SNOMED-CT[22][23], ICD-10[24] or an own terminology.

TRENCADIS uses XML as specification language for templates, which allows the definition of terms and rules using the DICOM SR specification. Figure 1 (left side) shows an example of a SR Template defined in XML. This kind of templates has a hierarchical tree-like structure that can be efficiently represented using graphs, which facilitates indexing and searching DICOM images and SR.

Templates contain attributes, restrictions and relations among fields, following the DICOM-SR standard rules. These fields can be filled by an authorized user to create and upload new reports into the system (Figure 1, right side).

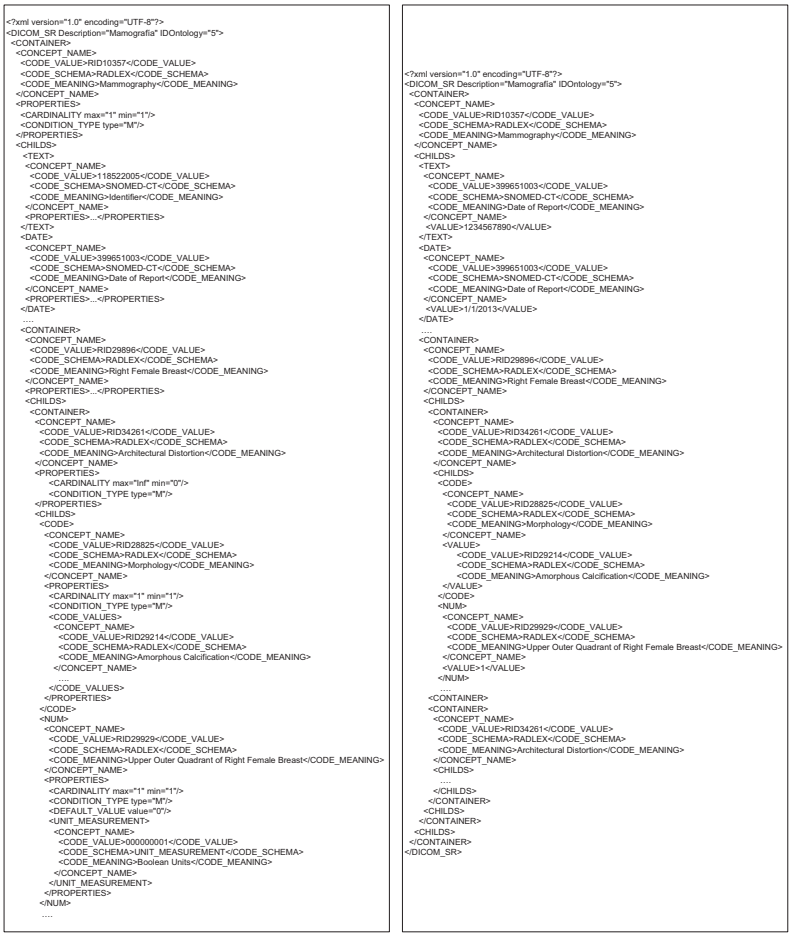


Fig. 1. Left side of the figure shows a Structured Mammography Reporting Template based on DICOM-SR. Right side shows a instance of this template.

3 Data Models

One of the main benefits of TRENCADIS is that supports the deployment and operation of information systems especially designed for medical applications, which have very specific requirements in terms of security and performance. To this end, TRENCADIS defines a data model for describing, storing and sharing DICOM objects – specifically, DICOM images and DICOM SR-based reports –.

Figure 2 shows the TRENCADIS storage architecture. Hospitals can use the DICOM Storage Service to share images in a TRENCADIS-based infrastructure. Each DICOM Storage Service can be registered with the "Indexer" service, which indexes the metadata and other relevant information extracted from DICOM and DICOM SR collections. These collections are stored with one

of storage backends available to the infrastructure, which can rely on several types of storage technologies, including relational databases (e.g. PostgresSQL), Grid storage services (e.g. LFC+SE) and Cloud storage services (e.g. Amazon S3).

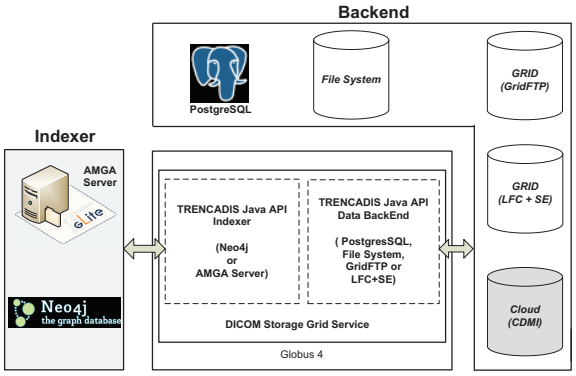


Fig. 2. Components of DICOM Storage Grid service.

3.1 AMGA Data Model

The TRENCADIS data model is based on AMGA [19] to leverage on the large computing facilities operated by the European Grid Infrastructure (EGI)[1]. AMGA is a Metadata Catalogue Service, which provides the means to describe and discover the data required in the different Grid sites. This data model is described in [25, 18].

Basically, in this model, the DICOM-SR "CONCEPT_NAMEs" are mapped to an AMGA collection, which is internally stored in tables of a relational database. The reports are indexed by the value of the "CONCEPT_NAMEs". These indexes are managed by the TRENCADIS DICOM Storage Service, which provides the necessary methods to insert, delete and query the reports stored with the AMGA Service.

3.2 Graph Data Model

The data model presented in this paper is based on the graph approach, where objects and their relationships are modelled and persisted as nodes and edges of a graph. Previous studies have reported that the graph model can improve flexibility and performance of database systems, especially when complex relations exist between the modelled entities [26]. Also, graph databases have proven

[1] EGI – European Grid Infrastructure: http://www.egi.eu/

to be useful for managing dynamic data models that represent highly connected data. This has motivated us to explore the use of graph databases for Structured Reporting.

It is a common practice to combine information from multiple studies in order to improve decision-making in the diagnosis, prognosis and treatment of cancer. In fact, several terminologies and ontologies have been introduced in recent years to facilitate the comparison of such studies, which has led to an increase in the need for tools that automate the Structured Reporting processing and annotation.

The graph data model addresses this need by defining a dynamic model in which the Structured Reporting Templates are represented as sub-graphs that can be built out of simpler templates. In this way, new templates with more complex structures can be obtained and new casual relationships that improve the model can be identified.

Representation of DICOM-SR Templates Figure 1 illustrates the structure of an XML document defining a DICOM-SR Template. The sub-graphs that represent the templates are built directly from these documents. Each XML tag element is mapped to a node in the graph model. The "typeNode" of the node represents the name of the tag. An "idOwn" attribute is used to differentiate between nodes with the same "typeNode". Hierarchical relationships between XML tag elements are represented by directed edges, where the source of the edge is the parent element and the destination is the child. Graphs created in this way are connected rooted graphs.

Figure 3 shows one of the sub-graphs created from the template shown in Figure 1, with root in the DICOM-SR "CONCEPT_NAME". Instead of creating two different structures in the graph for the template fields "Identifier" and "Date of Report", the part of the structure that coincides in both fields has been reused. Since these fields in particular are defined by SNOMED-CT, this node can be reused too. In summary, once a concept name is created in the graph, there are several nodes and paths that can be reused to define new concept names and only a subset of the elements that represent the new concept name is created in the graph, which are the code meaning and the code value in the example.

The edges are annotated with additional information to complete the graph. In particular, all possible paths are identified and the edges that represent them are labelled with the format: idOwn0#route0, idOwn1#route1, ..., idOwnN#routeN, where "idOwn" represents each node of the path and the "route" is used to indicate that the same path can go through different routes.

Maintaining path information in the graph allows us to store the data in such a way that it semantically represents the structure of the original template.

Initially, every possible path is considered to build the model, in order to create a highly connected graph. However, after analysing the graph created in this way, experts can decide which paths are irrelevant for their analysis, reducing the complexity of the graph.

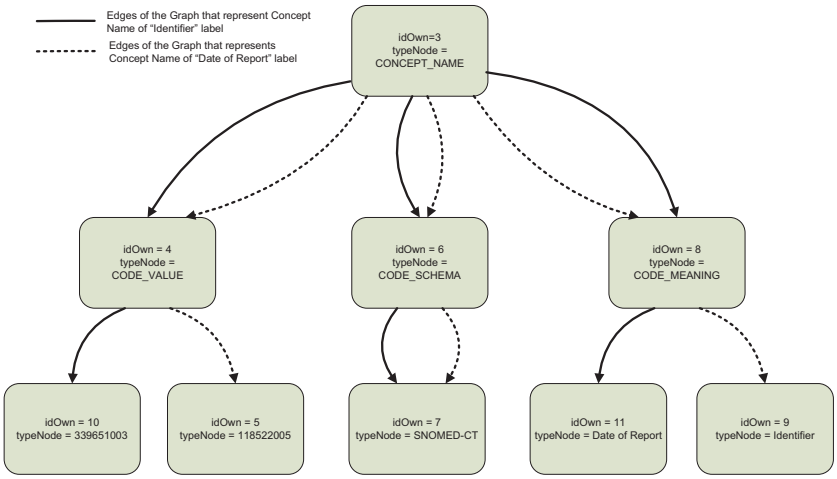


Fig. 3. Structure of the sub-graph with root in the "CONCEPT_NAME" for the fields "Date of Report" and "Identifier" from Template shown in the Figure 1.

Figure 4 shows an example of a graph where all possible paths have been identified and labelled. In this example, three different levels of hierarchy have been described for the "CONCEPT_NAME". Each hierarchy is built by following a path from the "CONCEPT_VALUE" to the "CONCEPT_SCHEMA" and finally to the "CONCEPT_NAME". For example, the field "Architectural Distortion" has been defined with the maximum infinite cardinality in the DICOM-SR Template, which is shown in the left side of Figure 1. Thus, several nodes can appear for this field in the same template. In this case, two different apparitions of the field will generate two different routes in the path: 2#1,3#1,9#1,10#1 and 2#1,3#1,9#1,10#2.

Representation of DICOM-SR Instances The graphs that represent the template instances are built in a similar way to the template graphs. However, there are a few differences in the building of the instance graphs:

- In a similar way as indexes are used with relational databases to speed-up the access to secondary keys, additional edges and paths are created in the instance graphs to facilitate rapid access to the data. Those are created from the root of the graph to the nodes: report identifier, patient identifier, date of report and study identifier.
- Since instance and template graphs are stored in the same database, they share the nodes that are common to both.
- New nodes are created for those fields that are specific to the instance (value nodes).

To create a new report, the corresponding template is used to find the path to the nodes in the database.

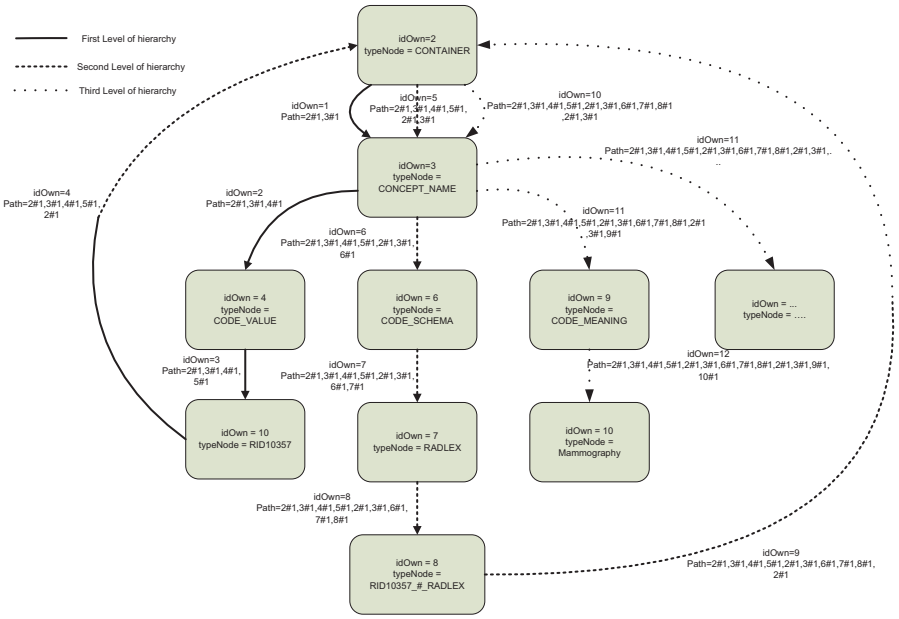


Fig. 4. Path construction for the "CONCEP_NAME" nodes.

4 Validation and Analysis of the Graph Data Model

This section presents the results of an experiment conducted to compare the new data model presented in this paper with the previous one, which is based on AMGA. The experiment evaluates both approaches in terms of flexibility to assimilate new data types and relationships in the storage and the performance of insertions and searches.

4.1 Flexibility

In general lines, both AMGA and graph-based models allows us to handle the structured reports, in a flexible way. Adding, removing or modifying the structure of the database do not require a considerable effort. In AMGA, these operations are mapped to SQL sentences that are executed in the back-end database with the objective of modifying the collection where the reports are stored. On the other hand, in the graph database, any modification of the structure of the database does affect the nodes and edges of the graph.

One of the main differences observed between AMGA and the graph database is that in AMGA, each template is stored in a separated collection. This design makes difficult the use of fields defined in different templates within the same query. For example, the field "Architectural Distortion" – which is defined in the Structured Mammography Reporting Template shown in Figure 1 – can be used in other templates, such as Clinical Examination. This kind of queries can be

useful to retrieve all the information available about the "Architectural Distortion" of a lesion studied both with a mammography and a clinical exploration. With AMGA, two different queries must be executed at the AMGA Server and the result must be processed before it can be used by the physician.

The graph model reuses the nodes that are common to more than one template, such as the "CONCEPT_NAME". This facilitates the execution of such queries where the same field is searched across the database, no matter how many different templates are traversed in the search. One additional benefit of this approach is that the execution time of the query remains constant when more templates are inserted into the database.

4.2 Performance

A test environment was set up to investigate the potential impact of the data model in the performance of the database. The graph model was implemented with Neo4j[2]. Both graph and AMGA databases were created from a dataset that includes 9,000 mammography reports from 1,000 patients. Before conducting the experiments, the dataset was modified, applying random changes to the reports in order to increase the randomness in terms of the number of fields (50-500) included in each report.

The database management systems were deployed in two identical servers, each one running Scientific Linux 5.9 on a 2-core (2.4 GHz) processor with 4 GB of RAM memory.

Inserting new reports in the database Table 1 shows the times required to insert 10 reports with 50-500 fields in the database.

Table 1. Structured reports insertion times in AMGA and the graph database

Number of fields	Insertion time in AMGA (ms)	Insertion time in Neo4j (ms)
50	0.589	0.475
100	0.702	0.506
150	0.801	0.469
200	1.281	0.570
250	1.423	0.636
300	1.784	0.598
350	2.635	0.683
400	3.070	0.871
450	3.304	0.775
500	3.636	0.852

The insertion time improves significantly in the graph data model, especially for reports with a large number of fields.

[2] Neo4j – Graph Database: http://www.neo4j.org/

Querying the database for reports Report fields form trees that follow the structure defined in the template. In these trees, each node has a distance to the root of the tree. A second experiment measures the time spent in searching the database with 4 queries, which each query targeting a field set stored at a different distance in the template:

- Query 1: Retrieve the total number of mammography studies that are stored in the database (0 fields with distance 0).
- Query 2: Retrieve the identifiers of all the studies carried out for the patient 461 (1 field with distance 1).
- Query 3: Retrieve the identifiers of all the studies carried out for the patient 461, with date after 01/01/2013 (2 fields with distance 1).
- Query 4: Retrieve the identifiers of all the studies carried out for the patient 461, with date after 01/01/2013 and featuring at least one lesion localized in the "retroareolar" region of the right breast, with the classification "mass" (2 fields with distance 1 and 1 field with distance 3).

Table 2. Structured reports searching times in AMGA and the graph database

Query	Searching time in AMGA (ms)	Searching time in Neo4j (ms)
Q1	0.055	0.91
Q2	0.060	0.93
Q3	0.064	1.02
Q4	2.344	4.31

Searching time is negatively affected in both models by the level of deepness that the searched fields have in the template. This time increases even in the new model, where this result is not expected.

In the case of AMGA, each level descended in the tree structure that represents the template is mapped to a SQL "INNER JOIN", which is a very expensive operation in terms of execution time. However, in the graph model, this is mapped to an additional jump between two nodes of the graph connected by an edge, which, at first sight, does not justify the increase in the searching time.

After analysing the structure of the graph database, we have found that certain nodes acting as links between the templates have a determining contribution to the increase of the searching time. The graph model maps the structure of the XML shown in Figure 1 to the database. Although this is desirable from the point of view of the physicians who want to use the system to discover relationships between different studies carried out with different approaches or on different patients, this design has an extra cost in terms of response time to access the results of the queries.

Further works are being performed to reduce the number of connection points between the templates. The implementation reported in this paper connects the

templates by using all the possible combinations of nodes – including those that contain one of the values defined by the terminologies and ontologies used with the template (e.g. "mass", "retroareolar") and also those that define the structure of the template (e.g. "CONCEP_NAME") –. As a result, 3,722,110 edges are created in the database to represent 9,000 records, but some of them provide irrelevant or redundant information. Future releases will omit these edges that do not provide relevant information to the physicians.

5 Conclusions

Automating the extraction of relevant information from radiology reports and helping physicians to improve their use of this information is of paramount importance for the improvement of cancer diagnosis and treatment. TRENCADIS provides a comprehensive way for creating and searching structured reports. This paper has presented a new data model that aims at improving these capabilities already present in TRENCADIS with more powerful mechanisms for discovering new information in the reports database.

The new data model was implemented and studied. The insertion of new reports in the database has been found to be faster than in the previous data model, which is based on AMGA. However, searching the graph database is not as efficient as expected at a first sigh. Future works will extend the present model to include a mechanism based on expert decisions for pruning the irrelevant edges from the graph.

Acknowledgements

The authors wish to thank the financial support received from The Vicerectorat d'Investigació de la Universitat Politècnica de València (UPV) to develop the project "Diseño de Componentes Cloud Facilitadores del Despliegue y la Alta Disponibilidad de Servicios TRENCADIS, para compartir Imágenes Médicas DICOM e informes Asociados DICOM-SR", with reference 20111013.

References

1. H. K. Huang, Short history of PACS. Part I: USA., European journal of radiology 78 (2) (2011) 163–76. doi:10.1016/j.ejrad.2010.05.007.
   URL http://www.ncbi.nlm.nih.gov/pubmed/21440396
2. H. U. Lemke, Short history of PACS (Part II: Europe)., European journal of radiology 78 (2) (2011) 177–83. doi:10.1016/j.ejrad.2010.05.031.
   URL http://dx.doi.org/10.1016/j.ejrad.2010.05.031
3. P. Margaret A. Hamburg, M.D., and Francis S. Collins, M.D., The Path to Personalized Medicine - NEJMp1006304, The New England Journal of Medicine 363 (4) (2010) 301–304. doi:10.1056/NEJMp1006304.
   URL http://www.nejm.org/doi/pdf/10.1056/NEJMp1006304

4. A. D. Weston, L. Hood, Systems biology, proteomics, and the future of health care: toward predictive, preventative, and personalized medicine., Journal of proteome research 3 (2) (2004) 179–96.
URL http://www.ncbi.nlm.nih.gov/pubmed/15113093
5. G. S. Ginsburg, J. J. McCarthy, Personalized medicine: revolutionizing drug discovery and patient care., Trends in biotechnology 19 (12) (2001) 491–6.
URL http://www.ncbi.nlm.nih.gov/pubmed/11711191
6. E. Bellon, M. Feron, T. Deprez, R. Reynders, B. Van den Bosch, Trends in PACS architecture., European journal of radiology 78 (2) (2011) 199–204. doi:10.1016/j.ejrad.2010.05.025.
URL http://www.ncbi.nlm.nih.gov/pubmed/20566253
7. T. B. Murdoch, A. S. Detsky, The inevitable application of big data to health care., JAMA : the journal of the American Medical Association 309 (13) (2013) 1351–2. doi:10.1001/jama.2013.393.
URL http://jama.jamanetwork.com/article.aspx?articleid=1674245
8. L. Hood, R. Balling, C. Auffray, Revolutionizing medicine in the 21st century through systems approaches., Biotechnology journal 7 (8) (2012) 992–1001. doi:10.1002/biot.201100306.
URL http://www.ncbi.nlm.nih.gov/pubmed/22815171
9. O. Ratib, A. Rosset, J. Heuberger, Open Source software and social networks: disruptive alternatives for medical imaging., European journal of radiology 78 (2) (2011) 259–65. doi:10.1016/j.ejrad.2010.05.004.
URL http://www.ncbi.nlm.nih.gov/pubmed/21444166
10. P. Kenneth D. Mandl, M.D., M.P.H., and Isaac S. Kohane, M.D., Escaping the EHR Trap The Future of Health IT NEJM, The New England Journal of Medicine 366 (24) (2012) 2240–2242. doi:10.1056/NEJMp1203102.
URL http://www.nejm.org/doi/full/10.1056/NEJMp1203102
11. M J A Eugster,M Schmid,H Binder,M Schmidberger, Grid and cloud computing methods in biomedical research., Methods of information in medicine 52 (1) (2013) 62 – 4.
URL http://www.researchgate.net/publication/234133698_Grid_and_cloud_computing_methods_in_biomedical_research
12. A. K.-B. Lukas Forer, Sebastian Schönherr, Hansi Weißensteiner, Günther Specht, Florian Kronenberg, Computational Medicine. Chapter 2: Cloud Computing - Bringing Computational Power to Medical Genetics, Springer Vienna, Vienna, 2012. doi:10.1007/978-3-7091-0947-2.
URL http://www.springerlink.com/index/10.1007/978-3-7091-0947-2
13. K. K.-Y. Lee, W.-C. Tang, K.-S. Choi, Alternatives to relational database: comparison of NoSQL and XML approaches for clinical data storage., Computer methods and programs in biomedicine 110 (1) (2013) 99–109. doi:10.1016/j.cmpb.2012.10.018.
URL http://www.ncbi.nlm.nih.gov/pubmed/23177219
14. S. J. Rascovsky, J. A. Delgado, A. Sanz, V. D. Calvo, G. Castrillón, Informatics in radiology: use of CouchDB for document-based storage of DICOM objects., Radiographics : a review publication of the Radiological Society of North America, Inc 32 (3) (2012) 913–27. doi:10.1148/rg.323115049.
URL http://www.ncbi.nlm.nih.gov/pubmed/22403116
15. A. Madaan, W. Chu, Y. Daigo, S. Bhalla, Quasi-Relational Query Language Interface for Persistent Standardized EHRs: Using NoSQL Databases, in: A. Madaan, S. Kikuchi, S. Bhalla (Eds.), Databases in Networked Information Systems SE -

15, Vol. 7813 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 182–196. doi:10.1007/978-3-642-37134-9_15.
URL http://dx.doi.org/10.1007/978-3-642-37134-9_15

16. R. Hussein, U. Engelmann, A. Schroeter, H.-P. Meinzer, DICOM structured reporting: Part 1. Overview and characteristics., Radiographics : a review publication of the Radiological Society of North America, Inc 24 (3) (2004) 891–6. doi:10.1148/rg.243035710.
URL http://www.ncbi.nlm.nih.gov/pubmed/15143238
17. D. Sluis, DICOM SR-integrating structured data into clinical information systems, Medicamundi.
URL http://www.healthcare.philips.com/pwc_hc/main/about/assets/Docs/medicamundi/mm_vol46_no2/sluis.pdf
18. J. S. Torres, J. Damian Segrelles Quilis, I. B. Espert, V. H. García, Improving knowledge management through the support of image examination and data annotation using DICOM structured reporting., Journal of biomedical informatics 45 (6) (2012) 1066–74. doi:10.1016/j.jbi.2012.07.004.
URL http://www.ncbi.nlm.nih.gov/pubmed/22841747
19. B. Koblitz, N. Santos, V. Pose, The AMGA Metadata Service, Journal of Grid Computing 6 (1) (2007) 61–76. doi:10.1007/s10723-007-9084-6.
URL http://link.springer.com/10.1007/s10723-007-9084-6
20. National Electrical Manufacturers Association (NEMA), DIGITAL IMAGING AND COMMUNICATIONS IN MEDICINE (DICOM) - PART 16: CONTENT MAPPING RESOURCE, Tech. rep. (2004).
URL http://medical.nema.org/dicom/2004/04_16PU.PDF
21. C. P. Langlotz, RadLex: a new method for indexing online educational materials., Radiographics : a review publication of the Radiological Society of North America, Inc 26 (6) (2006) 1595–7. doi:10.1148/rg.266065168.
URL http://radiographics.rsna.org/content/26/6/1595.full
22. P. López-García, M. Boeker, A. Illarramendi, S. Schulz, Usability-driven pruning of large ontologies: the case of SNOMED CT., Journal of the American Medical Informatics Association : JAMIA 19 (e1) (2012) e102–9. doi:10.1136/amiajnl-2011-000503.
URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3392868&tool=pmcentrez&rendertype=abstract
23. P. Ruch, J. Gobeill, C. Lovis, A. Geissbühler, Automatic medical encoding with SNOMED categories., BMC medical informatics and decision making 8 Suppl 1 (Suppl 1) (2008) S6. doi:10.1186/1472-6947-8-S1-S6.
URL http://www.biomedcentral.com/1472-6947/8/S1/S6
24. W. H. Organization, ICD-10 Version:2010.
URL http://apps.who.int/classifications/apps/icd/icd10online
25. I. Blanquer, V. Hernández, J. Salavert, D. Segrelles, Integrating TRENCADIS components in gLite to share DICOM medical images and structured reports., Studies In Health Technology And Informatics 159 (2010) 64–75.
URL http://www.ncbi.nlm.nih.gov/pubmed/20543427
26. J. J. Miller, Graph Database Applications and Concepts with Neo4j, in: Proceedings of the Southern Association for Information Systems Conference, AIS Electronic Library (AISeL), Atlanta, USA, 2013.
URL http://aisel.aisnet.org/sais2013/24

Big data and urban mobility

Antònia Tugores**, Pere Colet

Instituto de Física Interdisciplinar y Sistemas Complejos, IFISC(UIB-CSIC)

Abstract. Data sources have been evolving the last decades and nowadays a huge amount of information is available through web navigation. One of the new sources of information are social networks and they provide large datasets to be studied. In this context we are using geolocalized tweets as a source of information for mobility patterns. Data retrieval is achieved through public APIs and smart users selection. Besides, efficient data storage and fast access to the data is a challenging task for which we rely on NoSQL technology.

Keywords: Data storage, data management, big data, NoSQL, distributed database

1 Introduction

The amount of data available electronically has been exploding in the last years. In parallel, the world's technological capacity to store information has increased unexpectedly. As of 2012, every day 2.5 quintillion (2.5×10^{18}) bytes of data were created, and 90% of the data in the world today has been created in the last two years alone. This data comes from everywhere: sensors used to gather climate information, social networks, digital pictures and videos, transaction records, cell phone calls and GPS signals, genomics or complex physics simulations to name just a few. Capturing, curating, storing, searching, sharing, transfering, analysing, and visualizating those large data sets, or big data, are becoming key basis for studying the society.

Taking into account the 3Vs model [1][2] (high volume, high velocity and/or high variety information), the data we are working with, obtained from Twitter [3] social network, fits in the definition of big data. Although it has a low information density, its huge volume allows to infer laws and, in our case, it may be helpful to study commuting and human mobility patterns.

The objective of this paper is to discuss efficient ways to retrieve, store and manage large amounts of data from social networks such as Twitter to study human mobility patterns. The paper is organized as follows, data retrieval is described in section 2, section 3 summarizes current databases and explains our decision to use MongoDB. Section 4 describes the database configuration we have implemented, and finally, concluding remarks are given in section 5.

** e-mail of corresponding author: antonia@ifisc.uib-csic.es

2 Data retrieval

In order to study the mobility in some big cities (mainly London, Barcelona and Zurich) we are using geolocalized tweets. One can get 1% of all the tweets by using the stream API provided by Twitter, but in this case, less than 12 % of the collected tweets are geolocalized. As the geolocalized tweets are distributed all over the world, only a small fraction of them are located in the cities we are focusing in. As a consequence the network of users that can be constructed is smaller than desired.

To solve this issue after some months of data recollection we identified the users that have tweets geolocalized in the cities considered here. Then, in addition to the stream data, we download the Twitter timeline of these specific users.

2.1 Stream

We use the Public Stream that offers samples of the public data flowing through Twitter and it is suitable for data mining.

Twitter allows for applications to establish a connection to the streaming endpoint and through this connection a random sample of 1% of the tweets can be downloaded. This avoids the limitations imposed by Representational State Transfer (REST) APIs. The only limitation is that each account can create only one standing connection to the public endpoints, and connecting to a public stream more than once with the same credentials causes the oldest connection to be disconnected. In the same way, IPs of clients that make excessive connection attempts run the risk of being automatically banned.

One of the particularities of the Streaming API is that messages are not delivered in the same precise order as they were generated. In particular messages can be slightly shifted in time and it is also possible that deleted messages are received before the original tweet. This is not critical for the case considered here because we are interested at slower time scales (from minutes to hours) and therefore we do not need to have an exact timing and order of the messages.

Twitter streaming API requires keeping a persistent HTTP connection permanently open, and, by using listeners, the process that opens the connection should perform all parsing, filtering, and aggregation needed before storing the result. In our case, we store the tweets we download in the same form as they are received while deleted tweets have to be modified since they have a different structure. To facilitate data search and manipulation we use the tweet id as one of the indices of the database.

2.2 Users selection

In order to increase the number of geolocalized tweets in the cities to be studied, we identify users that on a specific period of time have posted at least one geolocalized tweet in one of these cities and then we download its timeline (that is the tweets the user has posted with some limitations).

First of all, we identify the tweets geolocalized in the three cities. This can be done though geoNear [6] MongoDB command. With this command we can specify a point for which the geospatial query returns the closest documents not exceeding a desired distance (radius) from the given point. Some databases limit the size of the results returned by a query. In the case of MongoDB this limitation is of 16MB [7] if not using GridFS [8]. In order to avoid exceeding this limitation we use a value for the radius of exploration of one mile and in order to cover all space in the city we make use of a fine grained mesh in which the points are separated by one mile.

2.3 Geolocalized data and users network

Twitter REST API has number of queries per time limitations, as for the methods we use the limit is 1 query every 15s. That makes a total of 43200 queries per month. And to avoid IP banning we try to keep away from the maximum.

For all the users in the selected group, we continuously get the timeline from the last tweet we collected and store the id of the last tweet we retrieve.

To retrieve the users network we get the list of uids collected and for each one, we get the followers and friends uids at the moment of running the query. In order to see the network evolution, the process is continuously running to get the network in different moments of time.

3 SQL vs NoSQL

The data type to be stored and managed was suited to be stored in a database, as they allow organized data storage, efficient data management (addition, removal, update and retrieval), and secure multi user access.

There are basically two kinds of databases: relational and non relational or NoSQL. Relational databases (RBMS or SQL databases), have a collection of tables of data items, all of which are formally described and organized according to a relational model. On the other hand, NoSQL databases are non-relational, distributed and horizontally scalable.

NoSQL database management systems are useful when working with large quantities of data and when the data's nature does not require a relational model. Even for data that can be structured, NoSQL databases can be useful for applications in which what really matters is the ability to store and retrieve great quantities of data, not the relationships between the elements. If the amount of data is large and relationships are needed, indexing and duplication of information in the stored documents are essential.

NoSQL databases are categorized according to the way they store the data and fall under categories such as key-value stores, document store databases, graph databases, multivalue databases, ordered key-value stores or key-value cache in RAM amongst others.

In our case, two conditions had to be taken into account when choosing a database. Streaming retrieval volume is not constant, ranging from 4 to 10 million tweets per day. Additionally we constantly retrieve the tweets of the selected

geolocalized users. We also note that significant political or social events may cause traffic spikes. Thus the database must have the capability and scalability to handle a large and steadily growing volume of data while providing enough margin and flexibility to cope with unexpected traffic peaks.

Apart from that, the individual messages or tweets are JSON [9] encoded. JSON, JavaScript Object Notation is a format used to transmit data in a human readable format (key, value) over a network connection as an alternative to XML. The attributes of a JSON-encoded object are unordered and not all the fields must appear in all the messages. In addition, JSON information contained in the tweets can change over time (fields can be added or removed) and the format of the field values (integer, string, datetime, ...) can also change.

Regarding scalability, which is one of our main considerations, for SQL databases this mainly relies on improving the server by adding memory, disk devices and/or cores. Even that in MySQL, master-master configuration allows writing to one and reading from the other, it was impossible to have a large percentage of data in memory to speed up queries.

Attending the scalability needs we focus on NoSQL databases out of which we consider CouchDB and MongoDB. CouchDB [4] uses JSON to store data and JavaScript as its queying language. Queries are basically map/reduce operations mapped in views, when adding a new query, a new view has to be added. It also provides ACID (Atomicity, Consistency, Isolation, Durability) [10] semantic and it does this by implementing a form of Multi-Version Concurrency Control, meaning that CouchDB can handle a high volume of concurrent readers and writers without conflict. Replication and failover are achieved by having multiple copies of the same data.

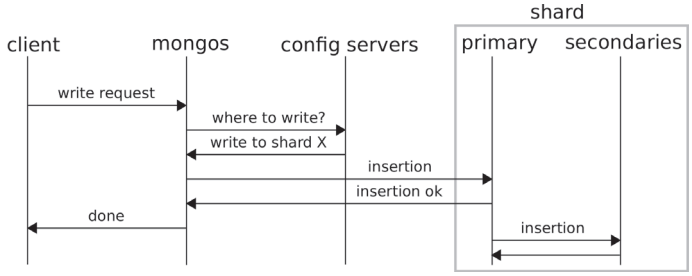


Fig. 1. MongoDB default write concern. MongoDB structure will be explained in section

On the other hand, MongoDB [5] uses BSON to store data (JSON-like documents with dynamic schemas). Supports SQL-like queries, map/reduce ones and aggregation, thus not restricting developers to a pre-defined set of queries. Indexing (up to 1024 indices per collection) allows queries to speed up and be run in realtime. Replication and failover are achieved the same way CouchDB does. Atomic transactions are only possible within the scope of a single document. But

attending to amount of concern the application has for the outcome of the write operation, durable writes can be achieved. With default write concern (Fig. 1), the application sends a write operation to MongoDB and the database confirms the receipt of the write operation. With stronger write concerns, write operations wait until MongoDB acknowledges or confirms a successful write operation. MongoDB provides different levels of write concern to better address the specific needs of applications such as confirm the write operation only after it has written the operation to the journal (it already implies durable operations) or after the write operation has propagated to the members of a replica set (group of computers with replicated data).

Both databases are quite similar and allow JSON documents. Even that MongoDB lacks real ACID transactions, it is more suitable when using large amounts of data. Horizontal scalability is much clear in MongoDB and the possibility to allow users to run their own queries without requiring additional configuration made MongoDB more suitable to our needs.

4 MongoDB configuration

In order to achieve high availability and scalability we use multiple replica sets. A MongoDB replica set, Fig. 2, is a cluster of mongo daemons (mongod) instances that replicate amongst one another and ensure automated failover. Replica sets consist of two or more mongod instances with one of these designated as the primary and the rest as secondary or delayed members. Clients direct all writes to the primary, while the secondary members replicate from the primary asynchronously with a delay of a few milliseconds, so that, reads can be performed either in the primary or the secondaries. Database replication with MongoDB adds redundancy, helps to ensure high availability and increases read capacity. Delayed members replicate the primary with a predefined delay time and we use them as backup devices.

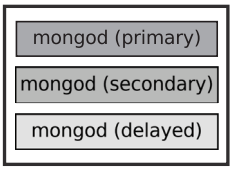


Fig. 2. Basic Replica Set

The current amount of data stored is more than 15GB of plain text per day, which represents an increase of 6TB per year in the size of the database. The approach to scale out, when one machine is not enough to store all the data, or write capacity needs to be increased, is sharding. It partitions a collection and stores the different portions (chunks) on different machines. Sharding

automatically balances data and load across machines and provides additional write capacity by distributing the write load over the computers. In addition to that, when a database collection becomes too large for the existing storage, a new machine (horizontal scalability) can be added and sharding automatically distributes collection data to the new server.

A sharded cluster consists of the following components:

- *Shards*. A shard is a container that holds a subset of a collections data. Each shard is either a single mongo daemon (mongod) instance or a replica set (RS)
- *Config servers*. Each config server (CS) is a mongod instance that holds metadata about the cluster. The metadata maps chunks or collection portions to shards.
- *Client instances (mongos)*. The mongos instances (CL) route the reads and writes from client applications to the shards. Applications do not access the shards directly.

Sharded MongoDB arquitecture requires a minimum of three configuration servers. The shards are usually replica sets and its number and internal structure depends on the amount of data to be stored and the reading speed needed for the applications. As for the number of client instances, this depends on the applications that need to access the database. We use just one client instance (CL).

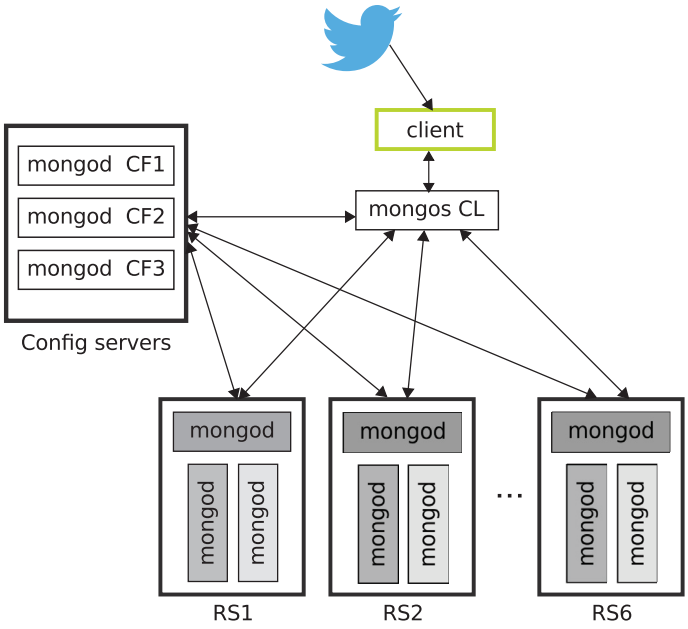


Fig. 3. Global MongoDB Configuration

In our case (Fig. 3), we use six replica sets with three members each. There are two elegible primary members and the third one is a delayed copy by 72 hours. This gives us failover security because if primary server crashes the secondary one move to primary status. And the third member helps us to recover from various kinds of human error such as inadvertently deleted databases or botched application upgrades.

The load of the configuration servers is small because instance maintains a cached copy of the configuration database and the total amount of activity is relatively low, therefore they are deployed as virtual machines with just one core and 1GB of RAM. The Client Instance (CL) also uses minimal resources and is also placed in a virtual machine alongside the application server and has two cores and 2 GB of RAM.

The shard key used is the tweet identifier and we added indices by user identifier and latitude/longitude to speed up usual queries.

To improve writing performance we took into account several MongoDB features when customizing the operating system in the servers that form the replica sets. One of the first considerations is that MongoDB uses write ahead logging to an on-disk journal to guarantee write operation durability and to provide crash resiliency. If the filesystem does not implements journaling and mongod exits unexpectedly the data can be in an inconsistent state. To avoid this issues we use ext4 since it implements journaling. Besides choosing a convenient filesystem, writing speed can be increased by mounting the file system where the database is located with the option noatime avoiding the logging of the record of the last time the file has been accessed or modified.

Apart from the filesystem configuration, some system parameters can be tuned for better write/read performance. In particular, the number of open files was increased to 96000 from the 1024 default value and the number of processes or threads per user to 64000 (in the replica sets all the MongoDB activity is handled by the user mongodb). We also avoided using hugepages related to NUMA kernels.

Finally, we note that the write concern policy we are using is the default one, see Fig. 1, because insertion speed in our case is more important than assuring that every single tweet correctly stored (the eventual loss of a single tweet is of little relevance for the study of the overall mobility patterns).

5 Results

We compared MySQL and MongoDB insertion speed. In MySQL we used Django Object Relational Mapping, ORM, to map JSON to objects. Django ORM is one of the most used Python object relational mapping alongside SQLAlchemy. In MongoDB we just inserted the JSON objects with no preprocessing.

5.1 Insertion performance

Figures 4 and 5 compare the results for the time it takes to store 100000 tweets in the database using a MySQL server and the MongoDB system described above with three replica sets.

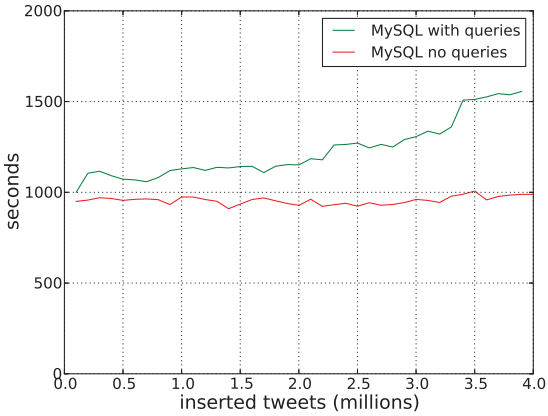


Fig. 4. Time to insert 100000 tweets in MySQL using an empty database and tweets processed with ORM. Linking (green) and duplicating information (red)

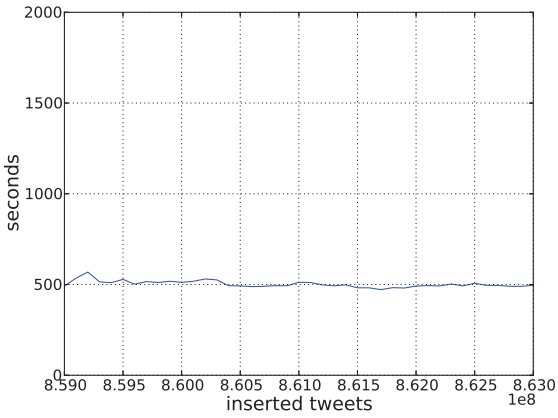


Fig. 5. Time to insert 100000 tweets in MongoDB by using direct insertion in a database with millions of tweets.

In Fig. 4 MySQL data is shown starting from a completely empty database. When inserting tweets in MySQL, as it is a relational database, we first perform a search to find if the twitter user exists, if not, a new record is created, while if the user is already there, a link to the existing register is performed. This requires a search for every tweet to be inserted which results in a larger storage time and in the fact that as the database grows the search takes longer and the insertion rate decreases. It takes 1000 s when it is empty, above 1500 s when there are four million tweets and almost 4000 s when the database has twelve million tweets.

Then, taking into account that in MongoDB no queries are done when inserting tweets and duplicated information is stored, we tested the same in MySQL, even that it means not using the relational properties of a relational database such as MySQL. In this case we just used the ORM to convert from JSON to MySQL objects. This is further illustrated Fig. 4 and shows that the injection rate remains almost constant over the whole range of 4 million tweets, just showing a minor reduction when the number of tweets increases.

In MongoDB, Fig. 5, on the contrary, tweets are inserted without searches and finding if there are tweets of the same user is done upon client request as part of the search query. Therefore the storage time is much smaller, around 500 s for the 100000 tweets, which is a speed up factor two with respect to the MySQL when no search is performed. Although the speed up is smaller than the factor three expected from the fact of having three replica sets, it is still substantial. What is more important, since we do not need to perform searches, this performance is maintained as the database size grows and in Fig. 5 shows the performance after the insertion of 850 million tweets.

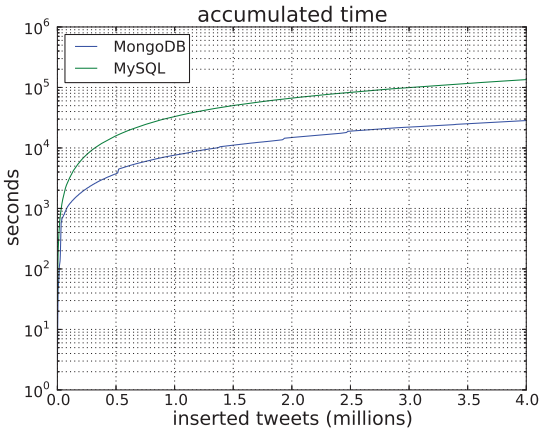


Fig. 6. Database insertion comparison, accumulated time

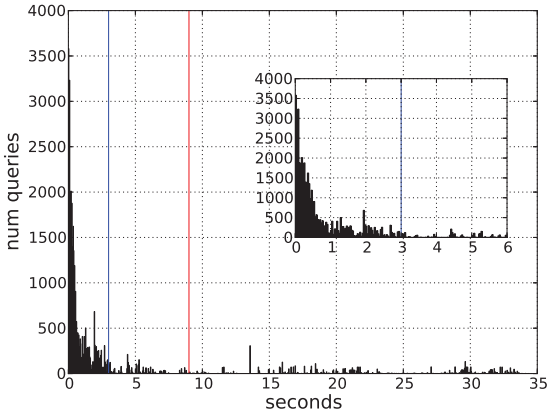


Fig. 7. Queries timing histogram for Barcelona metropolitan area. Blue line shows the median and red line the 70th percentile.

Finally, Fig. 6 shows the accumulated time in logarithmic scale to insert four million tweets in an empty MySQL database with relational queries and MongoDB. The difference in accumulated time grows exponentially as the database size increases.

5.2 Query performance

We are mostly interested in the fraction of the tweets stored in the database that are geolocalized, therefore MongoDB spatial indexing and geospatial commands play an important role.

MongoDB offers a specific geospatial index 2d for data stored as points on a two-dimensional plane. As of version 2.4 MongoDB also includes the index 2dsphere which conveniently supports queries that calculate geometries on a sphere. This index supports data stored as GeoJSON objects [11] which is the way geospatial data is stored in the tweets. Despite that, since we started with MongoBD 2.2, we are currently using 2d indices (latitutde, longitude) to determine the localization of the used when the tweet was posted.

MongoDB also includes the command geoNear which returns the documents on the database which have a geospatial location closer to a given location. The geoNear command can be used either with 2d data as well as with GeoJSON objects. In Fig. 7 we can see the histogram of geoNear queries in a database with one thousand million documents. We made use of a radius of one mile and a fine grained mesh over Barcelona metropolitan area in which the points were separated by one mile.

Even there is a group of slow queries, thirty seconds or more, the median is just of three seconds, and in 70% of the queries to get the tweets localized in a radius of one mile of a given point lasted less than nine seconds.

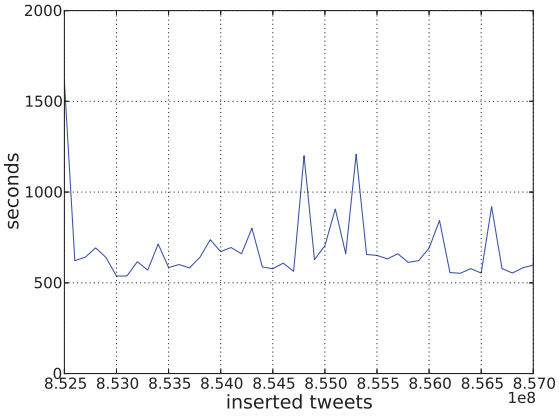


Fig. 8. MongoDB insertion with a database with millions of tweets while querying the DB with CPU and memory consuming geoqueries.

Finally we address the influence of the queries in the insertion data rate. In Fig. 8 we show the insertion rate in MongoDB when simultaneously performing queries on a database which has already stored 850 million tweets, so that queries require searching over a non trivial amount of data. The structure of MongoDB allows the queries to be performed on the primary nodes or on the secondary ones. The most disturbing situation for the insertion rate is when queries are performed on the primary nodes, and this is the case shown in Fig. 8. The presence of the queries induce peaks in the time to insert 100000 which can go over 1000 s but nevertheless the overall response of the system is quite satisfactory and the performance sufficient to keep storing all the tweets. Fig. 8 shows, in fact, the worst-case scenario. In practice, queries are performed over secondary nodes and in that way the insertion data rate is practically unaffected.

5.3 Preliminary results for mobility patterns

Preliminary results after retrieving data for six months show that London and Barcelona commuting areas are well defined just by using geolocalized tweets, see Fig. 9. A visual inspection shows that geolocalized data is distributed as the population density for the different cities, which means that already the six months sampling is representative. In order to further assess that the data is statistically adequate we plan to compare the statistics obtained from this six months retrieval with the ones obtained after one year.

Finally, in the framework of EUNOIA project public transport data and Twitter data amongst others will be used to characterise and compare mobility and location patterns in different European cities.

6 Concluding remarks

In summary, we have presented an example of big data retrieval and efficient data storage based on a no-SQL database, MongoDB. We have seen that geolocalized queries are suitable for large datasets and are not time consuming. Besides, we have been retrieving the maximum amount of geolocalized data from Twitter just taking into account some cities where mobility patterns are currently being studied.

Because NoSQL databases have weaker data consistency models, they can trade off consistency for efficiency and stand out in speed and volume. As for this, applications that need to use large amounts of data, data that grows over the time, schemaless data or geolocalized data are suitable to use MongoDB as storage.

Acknowledgements

Financial support from CSIC through project GRID-CSIC (Ref. 200450E494), from MINECO (Spain) and FEDER (EU) through projects FIS2007-60327 (FISICOS) and FIS2012-30634 (INTENSE@COSYP) and from European Comission through project FP7-ICT-2011-8 (EUNOIA) is acknowledged.

References

1. Douglas, Laney. "3D Data Management: Controlling Data Volume, Velocity and Variety". Gartner. 6 February 2001. http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Contr olling-Data-Volume-Velocity-and-Variety.pdf
2. Douglas, Laney. "The Importance of 'Big Data': A Definition". Gartner. Retrieved 21 June 2012. http://www.gartner.com/resId=2057415
3. Twitter web page. http://www.twitter.com
4. CouchDB web page. http://couchdb.apache.org

Fig. 9. Geolocalized tweets

5. MongoDB web page. http://www.mongodb.org
6. geoNear documentation. http://docs.mongodb.org/manual/reference/command/geoNear
7. BSON document size limitation. http://docs.mongodb.org/manual/reference/limits
8. GridFS. http://docs.mongodb.org/manual/core/gridfs/
9. JSON. http://www.json.org/
10. ACID. http://en.wikipedia.org/wiki/ACID
11. GeoJSON. http://geojson.org/geojson-spec.html

Analyzing File Access Patterns in Distributed File-systems

J. Gomes[1], J. Pina[1], G. Borges[1], J. Martins[1], N. Dias[1], H. Gomes[1], C. Manuel[1]

Laboratório de Instrumentação e Física Experimental de Partículas Lisboa, Portugal
jorge@lip.pt

Abstract. High performance distributed file-systems are commonly deployed at scientific computing datacenters to provide on-line storage capacity for applications running in processing clusters. Frequently these file-systems are used by many applications and users with heterogeneous file access patterns. A deeper understanding of these patterns may provide valuable information for both storage planning and tuning. This paper discusses methods to gather information about file access patterns and describes a tool currently being developed for this purpose. Finally data extracted from some Portuguese grid sites using this tool is presented.

1 Introduction

High performance distributed file-systems are often used in scientific computing datacentres. They provide the large storage capacity, scalability and fast data access often required by increasingly demanding scientific applications. However the exact way in which these complex systems are exploited by the applications and end users is frequently not fully understood or even disregarded. Most of the monitoring of the file-systems and underlying hardware and software components is performed by measuring the overall behavior of the system, for instance by looking at the global data access bandwidth, device read and write rates and volume occupation. A global vision of the storage usage and performance is very important and can be sufficient for most every day operational needs. However for capacity planning, optimal performance and for cost effectiveness is also important to understand and consider who is using the file-systems, when and how.

High performance distributed file-systems can be expensive solutions in terms of purchase, operation and maintenance. A better understanding of storage access patterns may provide relevant information to plan, improve and optimize the storage architecture and its configuration. These aspects assume today particular relevance giving the concerns with energy efficiency and total cost of ownership. In this context organizations can be faced with strategic scenarios and options which can require detailed information about its data access needs. A typical situation is when the storage system is achieving its end of life and a new solution needs to be planned. Another is when an organization has several storage systems possibly with different characteristics and even at different

locations and needs to understand how to better exploit them. Finally in face of commercial cloud computing offerings organizations need to be fully aware of their own storage access patterns so that costs can be properly estimated.

2 Portuguese Grid

The largest sites of the Portuguese grid infrastructure rely on the Lustre [1] distributed filesystem to implement their online disk storage. The Lustre filesystem can be connected to the grid infrastructure via a Storage Resource Management (SRM) [2] middleware. The StoRM [3] middleware is the SRM implementation used at these sites, that provides a thin layer between the SRM interface and a POSIX filesystem. Through StoRM, grid applications can access the filesystem indirectly using gridftp, a high performance, secure reliable data transfer protocol optimized for high bandwidth wide area networks, or directly when the filesystem is mounted in the computing worker node (WN). In the later case the grid applications can take full advantage of the underlying file systems performance and capabilities, including advanced features such as parallel I/O.

The StoRM model is simpler, more generic, and has better throughput than many other grid SRM solutions that implement both the grid interface and the storage system in a single monolithic system (eg DPM, dCache[4, 5]). It has also the advantage of proving a clean decoupling between the POSIX filesystem and the grid, therefore it can be used to integrate any POSIX compliant filesystem into the grid enabling a wider range of filesystem options.

This storage architecture allows the efficient sharing of resources between the Portuguese Tier-2 (which serves the CERN [6] experiments ATLAS and CMS [7, 8] at the Large Hadron Collider) and other researchers.

The three sites that are targets of this study are managed by the LIP Computing Group, which is also coordinating the Portuguese Grid operations. The sites are:

- LIP-Lisbon: the LIP datacenter in Lisbon that supports LIP research groups, the ATLAS tier-3, and grid virtual organizations
- LIP-Coimbra: the LIP datacenter in Coimbra that supports the ATLAS tier-2/3, LIP research groups and grid virtual organizations
- NCG: a national grid service that provides capacity to ATLAS, CMS, Portuguese researchers and grid virtual organizations of interest to the Portuguese scientific community

The analysis of the file access patterns was born out of the need of a deeper understanding of how the storage is being used at these sites to establish requirements and plan a future upgrade of the storage hardware and software.

3 State of the art

3.1 High Performance Distributed Filesystems

To provide good scalability and performance in large installations, the data needs to be stored across many storage nodes interconnected by fast networks. Some

Distributed Filesystems enable the integrated use of multiple storage nodes, and the creation of large data volumes that can span across many physical disks and nodes taking advantage of the inherent parallelism of the hardware. Examples of well-known distributed file-systems intended for high performance are Lustre, PVFS [9] and GPFS [10], among others. These three file-systems have the particularity of being mostly POSIX compliant. They offer the semantics of typical UNIX file-systems which makes them easily usable by most existing scientific applications. The version 4.2 of NFS [11] also has a similar design in which data can be spread across multiple servers.

The file system metadata, that is the attributes of the files, is handled differently by each file-system implementation, but is often kept separately in one or more file-system servers. Accessing the metadata can be a time and resource consuming operation that often becomes a bottleneck. Therefore it is also important to understand the impact of file-system operations that require metadata such as the opening of files.

3.2 Storage Access Monitoring Tools

There are many tools available to assist system administrators in monitoring data storage access. Most of these tools act at the device level and therefore can only provide information about device read and write operations. These tools miss the file system access details and fail to provide information about who is doing the operations. Some of these are: iostat [12], vmstat [13], iotop [13], atop [14], and dstat[15] among others. In addition tools such as Ganglia [16] can be used to collect device access information from multiple nodes and provide and integrated view. The information can also be collected via SNMP.

Ganglia has been used at the three sites to monitor the Lustre file-systems, it can provide per node and overall information about many metrics but again without providing file access details.

To obtain greater detail about the storage usage it is necessary to obtain information from the file-system layer. Unfortunately there are not many monitoring tools operating at this level. Some of these are: lsof [17] that provides a snapshot of open files at a given moment, dnotify [18] and inotify [19] which are Linux kernel features that allow to monitor file system events, FAM [20] a file change monitoring system that uses dnotify, Garmin [21] similar to FAM but works for a single user.

The inotify mechanism fulfills many of the requirements for a file-system activity monitoring it enables listening to events such as file create, open, close, delete, move, and attribute change. However it requires to specify which directories should be listened for events, and does not support listening directories recursively making much harder to monitor an entire filesystem. Additionally inotify can loose events if they are not read fast enough, therefore if a directory creation event is lost an entire directory sub-tree can become invisible for the monitoring application. Finally fanotify [22] is another potential Linux kernel feature for notification and interception of file system events, it has a global mode that enables to listen for any event without specifying directories, fanotify

is more intended for intrusion detection or antivirus software hence instead of providing the file details it provides a readonly file descriptor to the file that triggered the event.

Event driven file-system monitoring capabilities do exist in the Linux kernel, but they seem more appropriate to the monitoring of a reduced set of files or directories. The exception is fanotify but it has a shorter set of events that does not include file deletion, although promising fanotify is not generally available.

4 Monitoring with SystemTap

Another approach is SystemTap [23], which provides a generic approach for gathering information from a running Linux system. According to its developers can be used to collect information to assist in the diagnosis of performance and functional problems by probing or tracing the kernel or user space applications. SystemTap provides a scripting language that simplifies the writing of instrumentation probes which are then inserted in the running system as kernel modules. SystemTap is available on RedHat based distributions.

The SystemTap scripts contain handlers for specific events such as the invocation of system calls or kernel functions. When such an event occurs the code inside the handler is executed and can access information such as the original arguments of the call invocation. In the SystemTap nomenclature an event and its corresponding handler is called a probe. The scripts written in SystemTap language are automatically compiled into C code from which a kernel module is produced. The whole process is fully automated by the SystemTap tools.

4.1 Monitoring File System Operations with SystemTap

Using SystemTap it is possible to listen for file system activity such as the invocation of the relevant system calls. A proof of concept for a file access monitoring probe was created by the authors of this paper using as basis the examples provided with the SystemTap documentation. By probing on the open and close system calls at their return points it became possible to obtain its arguments and return status. Through this information the filenames, opening flags and call return status, and number of calls where collected. SystemTap provides several functions to access context information, by using them the process identification, user identification, elapsed between open and close, time spent on I/O and the event time were collected.

At this stage the major limitation observed was that filenames are returned exactly as they are passed in the call invocation, meaning that relative filenames do not have the complete pathname making impossible to establish the actual file location. Fortunately it was possible to integrate native code that translates opened file descriptors into absolute filenames.

The second challenge was to obtain the same information for file deletion. Although it is easy to probe the *unlink* system call, the same filename problem applies, however there are no opened file descriptor within the *unlink* context.

An analysis of the kernel source code enabled to understand the lines where within the *unlinkat* function it was possible to access *dentry* and *nameidata* data structures that contain valid information regarding the file. Therefore instead of probing into the *unlink* system call a probe was placed at a specific source code line number within the kernel function *unlinkat*, and from the data structures the filename and pathname was obtained. Listening for the *close_files* kernel function was also needed to cleanup information when files are not properly closed. Additionally several improvements were added to improve performance and robustness. The probe is fully filesystem agnostic and can be used with any Linux supported filesystem.

4.2 Monitoring System and Deployment

The SystemTap script to be deployed could monitor the *open*, *close*, *read*, *write* and *unlink* system functions. However within the context of the Lustre file system these operations are meaningful only on the client nodes were they are invoked. Also context information such as the process identifier could only be obtained on the client side. Therefore the SystemTap scripts needed to be executed on all nodes mounting Lustre filesystems. This is in itself a challenging task. Fortunately SystemTap allows remote execution of scripts. With this method the script compilation is executed on central nodes which then can start the script on target hosts via SSH. This method has the advantage of not requiring the installation of kernel-debug packages on every node.

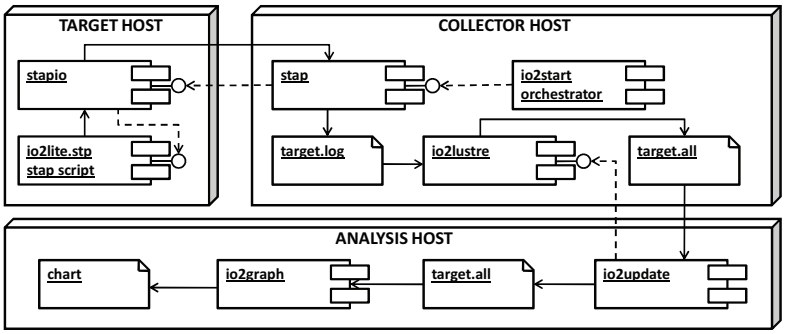


Fig. 1. System architecture

The figure 1 depicts the system architecture. A tool named *io2start* was developed to automatize the data collection process, namely: verify that the targets are up and reachable, check their environment including the presence of required

software and kernel versions, and finally start the SystemTap script. The tool continuously monitors the remote target hosts and the execution of the probes, and can act upon target reboot, probe failure, or environment change. This is particularly important in large infrastructures where the nodes can be restarted for management reasons. For instance in some of the three mentioned centers there are power management systems that automatically power down or power up the computing machines depending on the workload. As such it was essential to automatize the launch of the probes. The results from the probes are also collected centrally by the tool and stored in plain files, one per target. The tool is run on a collector host, depending from site size there can be multiple collectors, and each collector may provide support for a different runtime environment such as operating system version or kernel version.

Another function of the collector host is to complete the information gathered by the SystemTap script by fetching additional information from the filesystem regarding each file being accessed. This function is performed by *io2lustre* which adds Lustre filesystem specific information, namely it uses the absolute filename to obtain the Lustre file location in terms of Lustre OST server and OST volume, additionally it obtains the file size and creation date. The objective of splitting this functionality is to make the SystemTap probe as simple, fast and filesystem independent as possible. The slower metadata related operations are thus performed at a second stage.

4.3 Analyzing the Data

A tool to produce graphics from the data is currently being developed. The plots shown in this paper were already produced by this tool. The amount of information produced in one week of data collection for the three sites can reach more than 12GB. The processing of several weeks of collected data needs to be efficient. The tool is being developed in Python and the initial work has been focused in designing a simple framework that could process the information with flexibility and performance taking advantage of multi-core architectures. Simultaneously the framework simplifies the development of new data plots by focusing the development in the actual filtering of the data and in the design of the plot.

The framework has a main process that orchestrates the data processing and finally draws the chart. The main process can start multiple data input processes that read the data in parallel, perform some validation and prepare it for processing. The data is then written in a common queue which is also shared with multiple worker processes. Each worker process receives multiple blocks of data and creates vectors with the variables relevant to the intended chart. This is achieved by calling a predefined function specific for each plot that receives as input one recorded file-system event at a time. Once the workers finish they return their partial vectors to the main program which merges the data via another predefined function. Finally the main program calls the predefined plot drawing function.

4.4 Performance

To validate SystemTap as a performance gathering tool for the analysis of distributed filesystems several performance measurements were done to assess the impact of the probes in the intended environment. An isolate computing node was used to perform I/O operations over an idle Lustre volume. The probe for *open*, *close*, *read*, *write* and *unlink* was used, but the tests were focused to the read and write operations as these are the most frequent. The node was an HP DL160 G6 server with 24 GB of RAM and two Xeon E5540 quad core processors. The node was connected to the Lustre filesystem via Gigabit Ethernet.

A file of 2GB located on Lustre and already cached on the local system was read by a test program using a block size of 1KB. The tests were performed using up to four instances of the test program running in parallel. A clear degradation of performance (table 1 and 2) can be observed when the tests are performed with the SystemTap probes loaded. This degradation becomes more evident with the increase of the number of parallel instances.

Instances	Probe	Operation	Block Size	Size(GB)	Wall clock(s)	System(s)
1	NO	Read	1KB	2GB	3.90	3.87
2	NO	Read	1KB	2GB	9.49	18.13
4	NO	Read	1KB	2GB	23.60	63.94
1	YES	Read	1KB	2GB	7.45	7.42
2	YES	Read	1KB	2GB	16.75	31.14
4	YES	Read	1KB	2GB	60.22	233.93

Table 1. SystemTab read I/O performance on Lustre cached file

The SystemTap probes are triggered for all processes in the system. Probes cannot be applied selectively to the system calls where Lustre is involved, or to a specific set of processes. Therefore to minimize delays the probes try to return as soon as possible when the calls are not related with the pathname prefixes that are being monitored. The pathname prefix filtering is performed in the file open probe. The file descriptor plus the process identifier are kept to identify which opened files should be monitored. Still the probes are triggered for all files and devices. Therefore the probes may impact any read and write operation even if unrelated to Lustre. To evaluate this potential impact tests where performed on /dev/null using again multiple test instances in parallel.

As it can be observed in table 2, the impact is consistent with the previous results and when using multiple I/O instances in parallel the impact becomes very high. It was also observed that the amount of time spent by the probes caused the SystemTap module execution to abort due to a high fraction of the total real time being taken by the probe handlers. This safety protection can

Instances	Probe	Operation	Block Size	Size(GB)	Wall clock(s)	System(s)
1	NO	Write	1KB	2GB	0.20	0.14
2	NO	Write	1KB	2GB	0.53	0.90
4	NO	Write	1KB	2GB	2.03	7.60
1	YES	Write	1KB	2GB	2.95	2.85
2	YES	Write	1KB	2GB	10.99	20.70
4	YES	Write	1KB	2GB	54.38	194.54

Table 2. SystemTab Write I/O times on /dev/null

be removed though probe compilation directives but it will ultimately result in overloading the system with a very large fraction of time being used by the probes.

The explanation for these issues is in the Linux Kernel Kprobes [24] mechanism which is used by SystemTap to implement the probe points. The Kprobe mechanism shares information that could be corrupted on a SMP system, therefore it uses a spin-lock to control access on probe operations. Furthermore it disables interrupts and process preemption during probe handling. Therefore on SMP systems probe operations are in fact serialized causing a deep impact clearly observable in the measurements. The placement of the probes on call return increases badly this effect. In table 3 the probes were placed on call invocation only resulting in a much smaller effect, however with this method the access to the system call elapsed time and return status is lost.

Instances	Probe	Operation	Block Size	Size(GB)	Wall clock(s)	System(s)
1	YES	Write	1KB	2GB	1.04	0.99
2	YES	Write	1KB	2GB	2.07	3.37
4	YES	Write	1KB	2GB	3.75	12.98

Table 3. SystemTab Write I/O times on /dev/null without waiting for return

Due to these results three versions of the probe have been created. The *full* version monitors the read and write calls on their return points enabling to capture detailed information but with potentially high performance impact. The *ops* version that does not wait for call return, and the *lite* version that does not probe on read and write calls.

While running the proof of concept on the production systems the *lite* probe was used to prevent potential performance degradation. Since the main goal is to observe what files are being accessed, when and by whom it was considered acceptable to run probes just on *open*, *close* and *unlink* calls. The implemented

architecture is still capable of processing information from all probe types when needed.

4.5 Preliminary Results

The figure 2 shows the number of open operations performed at the three datacenters by the several scientific domains supported in these facilities namely through the grid but also via local access to the computing resources. The center names appear as *lipc*, *lipl* and *ncg*. The data was collected over a period of 18 days. The domain that has the larger number of file open operations is engineering, followed by the CERN LHC experiments, and Astroparticles research.

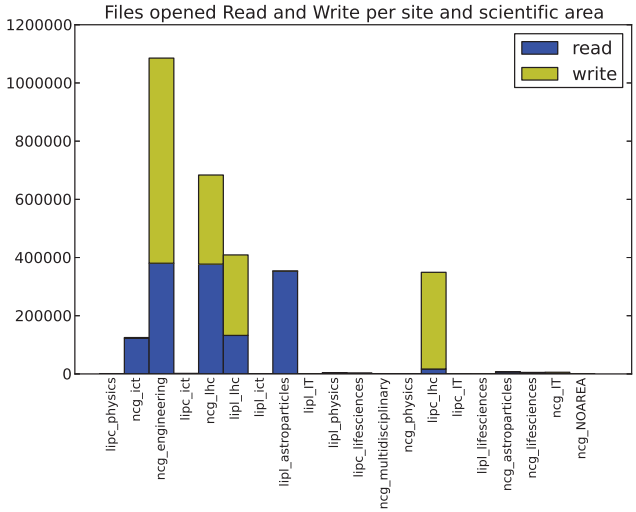


Fig. 2. Number of open operations per scientific domain

In figure 3 the number of unique opened files per scientific domains in shown. Although the Engineering community had the larger number of opened files many of these open operations are performed over the same files. Actually the Engineering community used less than 150000 different files while the LHC community used a much larger number of different files. This shows how different the usage of the storage resources can be across the user base.

The figure 4 shows the evolution of the number of file open operations along 18 days of data capture. The figure shows a peak of usage at the LIP Lisbon datacentre that would not have been noticed if the analysis would be performed only on the accumulated total. This shows how asymmetric the data access can be. This behavior shows that further monitoring is needed to observe and better understand these events.

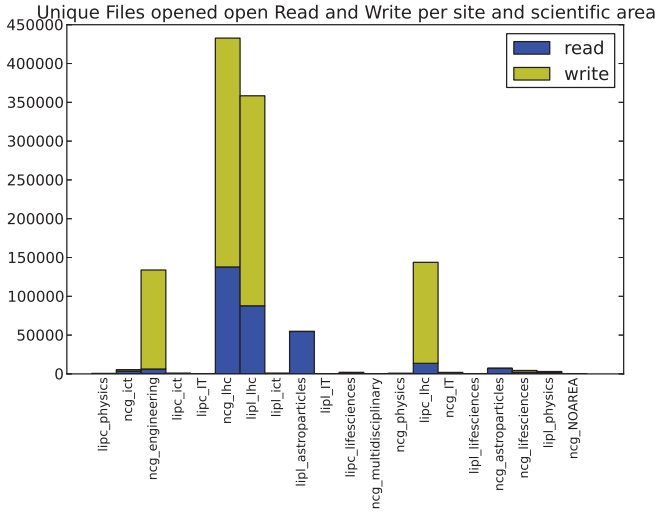


Fig. 3. Number of unique files opened per scientific domain

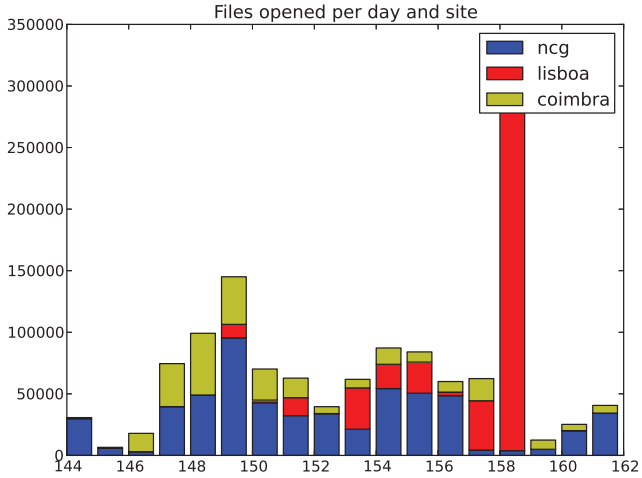


Fig. 4. Number of files opened per day

The figure 5 shows the number of open operations for files located in several Lustre servers. It shows how the several Lustre servers holding data were called to serve files located in their local disk arrays. It can be seen that one server at LIP Lisbon had to serve a large number of opened files. This was the same

server involved in the peak seen in figure 4. A deeper investigation concluded that the peak was caused by a massive file transfer operation.

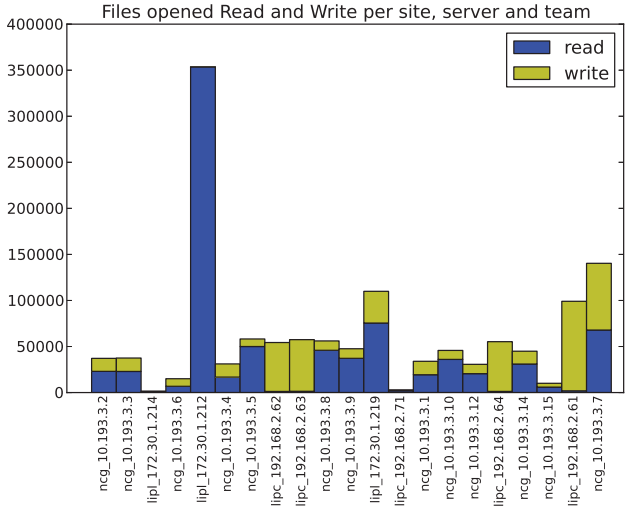


Fig. 5. Number of files opened per location and Lustre server

The range of charts and capabilities of the tool is considerably larger than what is shown in this paper and is under active development, it aims to fully exploit the range of data collected with the probes. This set of selected charts show already some of the capabilities and potential. The number of days during which data was collected is still small and as such it is yet too early to take conclusions.

5 Conclusions

The SystemTap infrastructure is a powerful and robust tool for system monitoring. Thanks to its flexibility it was possible to instrument the Linux kernel by placing probes at selected places enabling the catching of selected file system operations. The proposed architecture was able to perform well and collect information from Lustre installations at three datacenters in different geographic locations while they were being operated in production, running high throughput and high performance computing applications being submitted via grid and local access. The system deployed as a proof of concept allowed a first glimpse at a more detailed view of the file systems usage.

Although SystemTap is mentioned as a tool suitable for assisting in the diagnosis of performance issues, we have observed in the tests (tables 1, 2 and 3), considerable performance degradation when using SystemTap probes on operating

system calls when intensively invoked. This effect must be carefully considered when deploying both this tool and SystemTap probes in general.

The overall architecture of the system performed well and by placing the probes on the filesystem client nodes a good level of information could be collected with no noticeable impact on the file servers. Furthermore it allowed a selective deployment of the tool.

6 Future Work

The tool is being further developed aiming at better reliability, performance and completeness. More charts are being developed to enable a deeper understanding of the file systems usage and better exploit the available data. The use of non-SQL databases for data storage and analysis is also being considered. Finally a longer, deeper and more complete analysis of the three Lustre installations will be conducted.

Acknowledgements

This article was only possible due to the financial supported provided by the FEDER funds trough the COMPETE program and by Portuguese national funding trough the Portuguese Foundation of Science and Technology (FCT).

References

1. Lustre `http://www.lustre.org`
2. Storage Resource Manager `http://en.wikipedia.org/wiki/Storage_Resource_Manager`
3. StoRM `http://storm.forge.cnaf.infn.it/home`
4. DPM: Disk Pool Manager `https://svnweb.cern.ch/trac/lcgdm/wiki/Dpm`
5. Patrick Fuhrmann and Volker Gulzow: dCache, Storage System for the Future, Lecture Notes in Computer Science, Volume 4128, Euro-Par 2006 Parallel Processing, 1106–1113, (2006) `http://www.dcache.org/`
6. European Organization for Nuclear Research `https://www.cern.ch`
7. ATLAS Detector `http://atlas.ch/`
8. CMS Detector `http://cms.web.cern.ch/`
9. PVFS: The Parallel Virtual FileSystem `http://www.pvfs.org/`
10. GPFS: General Parallel File System `http://www-03.ibm.com/systems/software/gpfs/`
11. NFS: Network File System `http://nfs.sourceforge.net/`
12. iostat - Report Central Processing Unit (CPU) statistics and input/output statistics for devices and partitions `http://sebastien.godard.pagesperso-orange.fr/man_iostat.html`
13. Procps - The /proc file system utilities `http://procps.sourceforge.net/`
14. Atop tool `http://www.atoptool.nl/`
15. Dstat: Versatile resource statistics tool `http://dag.wieers.com/home-made/dstat/`

16. Ganglia ganglia.sourceforge.net
17. LSOF http://people.freebsd.org/~abe/
18. dnotify https://github.com/Dav1dde/dnotify
19. inotify: inode-based filesystem notification technology http://inotify.aiken.cz/
20. FAM http://oss.sgi.com/projects/fam/
21. http://people.gnome.org/~veillard/gamin
22. fanotify http://www.xypron.de/projects/fanotify-manpages/man7/fanotify.7.html
23. SystemTap http://sourceware.org/systemtap/
24. Kprobes http://sourceware.org/systemtap/

Studying the improving of data locality on distributed Grid applications in bioinformatics

José Herrera**[1] and Ignacio Blanquer[2]

[1] Universitat Politècnica de València
[2] Universitat Politècnica de València, Institute of Instrumentation for Molecular Imaging - I3M

Abstract. Data locality severely affects performance on Grid distributed applications. Many applications require accessing remotely stored large blocks of data over thousands of kilometers. The replication of data along the Grid infrastructure reduces the risks of bottlenecks and increases availability, however proper selection of the rightmost source of data is not automatically provided by the infrastructure. Moreover, a number of applications running on the Grid require the retrieval of small sections of reference files which ought to be copied locally. EGI middleware provides APIs which enable the opening of files and fetching collecting portions of datasets, although performance is affected when accessing multiple random blocks. This paper proposes an extension that provides a better selection of the rightmost storage resource, according to historical records and network criteria, combined with the ability of caching blocks of data retrieved through GFAL calls. In order to validate the model, an experiment has been carried out using a nucleotide alignment code based on the Burrows-wheeler transform, widely used in modern Next Generation Sequencing data.

1 Introduction

In a highly distributed environment such as Grids, data access constitutes a major element affecting the performance of Grid applications. Modern Grid infrastructures rely on the replication of data to reduce bottlenecks and increase the chances of approaching computing power to data. Data catalogues can keep indexed several copies of the same data, providing in a round-robin or random fashion the actual location of a logical file. However, this approach faces two relevant restrictions: Firstly, data files should be either retrieved completely or manually split into smaller chunks in case that only partial access is needed; secondly catalogues do not provide the rightmost location for each job depending on the bandwidth.

The first issue is partially overcome by software libraries such as GFAL [17], which exposes a POSIX-like interface to applications. Thereafter, applications can open files, seek pointers and retrieve only the bytes required overriding the

** e-mail of corresponding author: jherrera@upv.es

need of download the whole file. This may also facilitate the migration of applications which can be easily modified by pointing to external locations. In addition, GFAL supports the access to local files with the same API. Nonetheless, GFAL access to remote files is costly and may not be efficient. In the case of high data-locality access patterns, GFAL may be combined with caching strategies which could considerably improve performance. This is one of the key points tackled by this paper. Other approaches consider using automatic caching systems [19] for the whole access to remote files. Despite that this will be less intrusive, we want to concentrate in an approach in which the programmer may have deeper control.

The second problem might be better dealt with at two levels. At first level, the GLUE schema [18] provides a way to define the closest Storage Element (SE) to any given Computing Element (CE) which manages the batch queues that will effectively run the jobs. If a replica is available at the closest SE, then the retrieving APIs could be amended to get the most suitable replica. However, if no replica is available at the closest SE, then the choice is not trivial. The best approach would be to keep periodic records of the latency and bandwidth performance among the different pairs CE (actually, the working nodes of the CE) and the SEs. This can be kept distributed and continuously updated to keep track of the performance.

This article presents an experiment to measure these two approaches and presents early conclusions on the results. The article presents also a proposal of software architecture to implement data caching and network performance recording for the matchmaking of the most suitable replica in the storage. The article is structured as follows. This section introduces the issues and the approaches. Section 2 describes a proposed architecture to improve data locality. Section 3 describes the use case, and section 4 describes the experimental results. Section 5 lays down the conclusions.

2 Architecture Overview

To improve the current Grid I/O system, we propose a new architecture gfal-CE (Grid File Access Library Cache Extension) with new enhancements based in temporal and spatial locality. The present article describes the proposal of the architecture and the results of a set of experiments which will justify the suitability of the approach.

Figure 1 shows the proposal to reduce remote file access impact. The design of architecture is guided by the operational and performance requirements:

- Increase performance in remote file read operations.
- Reduce, on cache miss, wait time.
- Uses multi-source files transfer to reduce transfer time.
- Applies strategies to decrease bandwidth use.
- Ensures and supports common GFAL access API.
- Transparency hides algorithms complexity for end user.

We define temporal locality as the probability that at one point in time a memory location is referenced by the process; thereafter it is likely that such location will be referenced again sometime in the near future. Similarly, spatial locality can be defined as the probability that when a memory location is referenced at a particular time, nearby memory locations will be referenced in the near future. The use case proposed in the article, as it can be deducted from the experiments, is a good example of both temporal and spatial locality.

Nowadays in most computer systems scenario, there are numerous examples of cache systems. From L1 and L2 system cache in multicore processors to proxy cache servers that store web pages (in order) to bring to web browser in a fast and efficient fashion. In Grid systems, denote Dcache, there is a cache system to speed up the retrieval of slow-performing data storage. Some free proposals have been found such as [13] however, currently EGI clients side lacks of cache system to improve data transfer features.

We suggest that a process can use a new GFAL-API when it may benefit from caching data. This new layer would spot if the file block is locally saved. In such a case, data is delivered to process. When file block is not in a local position, the new layer looks elsewhere for data block in CloseSE. If this second retrieval attempt fails then it downloads the file block using GFAL classic API. In such a way, our proposal is a two level cache system to be used prior to the remote access. The first level would be associated with local process and local data, while the second level would be associated to EGI infrastructure provider facilities.

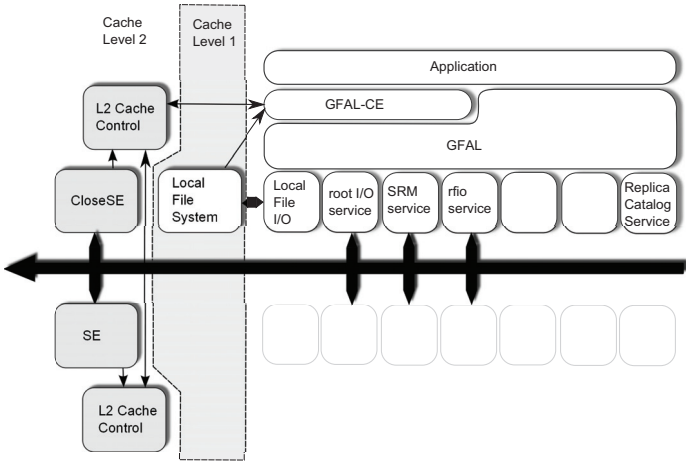


Fig. 1. gfal-CE Architecture preview.

A file must be divided in chunks that are transferred using GFAL API to services provider and local node cache. It is possible to share files in this way between different nodes. This would overcome the need of physically splitting

files, which ultimately has an impact on the code execution. Having said that, further effort is needed to design in the new API a conversion system to map out real access to block access.

The first level cache saves file blocks in local CE disk infrastructure. Blocks are fetched and downloaded in data miss operation. No data is shared between processes that do not share the same file system. Only threads running in the same processor must use same first level cache data. Saving data to file system increases cache size.

When an application data request cannot be met by local cache, then gfal-CE layer calls to a second-level control cache server. Since control cache is not the supplier of data, the first step is to determine the position of the file blocks stored in the SEs and the bandwidth reduction strategies. If data is not in second-level cache then it must be downloaded using different techniques such as deduplication[15], delta encoding[14], compression, etc... or using GFAL API and/or using file replicas or stripping. Next request to same block will be served quickly. Second-level cache is shared between all CE's running in a NGI infrastructure provider. When a file is used by a CE, the next time that a different CE uses the same file, it is served by cache system. A second function of control cache server is to download data in anticipation of data request for jobs.

Gfal-CE layer is located above GFAL. The most suitable solution would be to extend GFAL API. It is necessary maintain previous GFAL API to migrate some code and develop new code as soon as possible, so that by increasing API interface with new primitives efficiency is increased. Only processes that user gfal-CE use cache services, whereas other processes using original GFAL library do not benefit from such improvements. Therefore, two proposals are able to coexist in the same infrastructure. Jobs may use the new library not requiring changes in the infrastructure level, by providing a second lightweight layer.

Bearing in mind that reading operating operations in files are greater in number that writes, which does not happen on the reference data of the use case proposed, our first proposal overlooks writing operations in cache system. A feature in gfal-CE and other cache system is that cache size is limitless. Hard Disk storage in either SE or CE are far greater that in other cache proposals with limited space. This will affect only client-ends.

A cache system could hide the complexity and access data needs and prevent needing to downloading a whole file previously, either through scripts or the job description language (JDL) sandboxes. Extend GFAL (using gfal-CE) avoids this error source thus we can increase fault tolerance of the entire system.

3 Use case: Burrows-Wheeler alignment

The so-called Next Generation Sequencing constitutes a new type of sequencing devices which are providing an unprecedented large of raw data but in a more fragmented fashion. These features (billions of fragments below 200 basis) created the need for new methods for alignment and processing. Among them, the Burrows-Wheeler transformation [1] and the FM-Index [11] have been in-

tensively used for the implementation of tools such as Bowtie [6], BFAST [8], SOAP [4], HPG-Aligner [5] and BWA [10].

BWT-based alignment is an embarrassingly parallel process that takes a variant number of input fragments and compares them with a reference (normally an existing consensus genome). It uses three pre-computed data structures from the reference genome and the computational cost depends on the length of the input sequences to search for and the number of errors (freedom degrees) allowed. Details on the algorithm can be found at [10]. The reference data structures are huge (in the order of more than 40 bytes per nucleotide base), so different techniques are used to reach a compromise solution among memory restrictions, performance and accuracy. In a restriction-free model, each run will access to a limited number of elements in the reference data, depending on the size of the input data and the number of errors. However, those accesses cannot be predicted a-priori, thus requiring the whole reference data. The subsequent accesses for the different basis of each sequence are normally located in different areas of the reference data.

However, when processing input sequences that have common prefixes, the same elements of the reference data will be required. If data is somehow sorted, the changes of reusing the same data increases. Reordering the whole 200-length sequences may be prohibitive, but sorting only by the first basis will require a reasonable computing cost. This would suffice for increasing the hits on accessing elements of the reference data.

An out-of-core version of the BWT algorithm for the alignment of an input sequence with respect to a reference has been implemented based on [5] and [10]. This algorithm stores directly the three main pre-computed data structures (the BWT transformation, the number of occurrences in each basis at each position and the reference genome) in three different files. In the case of the Human Genome [2], the total size required will be above 100 GB. The experiments have been carried out with the genome of Drosophila Melanogaster [3], which requires substantially less memory. When accessing a single element in these data, the algorithm checks if the data is available and if not it fetches a block in local disk. The elements can be randomly accessed since the position is known.

4 Experimental testing

The experiments aim at verifying the hypothesis posed in the introduction (caching chunks will have a positive effect on performance and SE selection will have an important impact on performance). These experiments will also identify key parameters that will be used to tune in the architecture. The experiments will cover the execution of an out-of-core version of the algorithm with different configurations, comparing to the execution of the same case with an on-memory application. For the out-of-core case, several strategies will be tested. We start downloading the whole file, with and without multi-streaming. Multi-streaming will be consider both at the level of the APIs and manually.

Additionally, we measure the importance of choosing the nearest (in terms of bandwidth) site. Finally, we implement a last version using directly the GFAL software library to analyze the potential impact on the locality of reference. The different experiments are described next.

4.1 A first and key experiment

Using the BWT algorithm described above, we first developed a two-part implementation to reduce memory requirements. Based on a recursive search code developed in [9], we coded a new two step program. For comparison purpose, we need other implementation running under the same conditions.

Now we present the general conditions for the first experiment. It consists on two main data files: the sequence file to search and the reference data file that contains the known data. The search file contains only one line 200 nucleotides wide (Sequence A). The second one used the sequence of the chromosome one from HG19 Homo sapiens in FASTA format from [3]. This implies a search space of 254,235,634 nucleotides. We allow up to 2 mismatches in the searching process.

The execution of original code (BW_search) has a runtime of 864s using a Intel Core i5 760 with 8 GB memory and swap file employment around 16 GB. The total memory for execution was 24 GB. Both main memory and swap file have high occupation level. Due to high memory usage we are trying to compare execution time only in local processor.

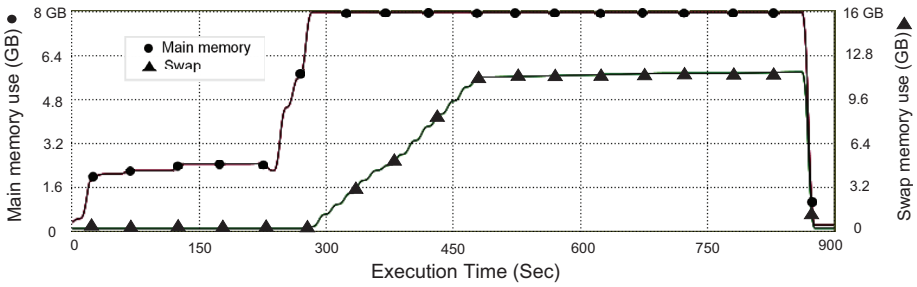


Fig. 2. Memory use for BW_search.

With the purpose of reducing the total execution time in BW_search and decreasing the total memory used in search process, we divide the code in two steps: BW_build and BW_quest. BW_build writes files in hard disk with the BW transformation and all data search. BW_quest use files created in a previous step for search algorithm.

BW_build, the first step, generated four files with data structures that before were in memory. All big structures data was stored in separated files than they were uploaded to SE in grid infrastructure if it is necessary. Total execution time in test environment was 353s. BW_build has Independence from one of the parameter, the search file.

BW_quest use files created in BW_build to search chain location. BW_quest will not load files in central memory supported structures. Every access has been converted to file accesses in order to reduce the total use of RAM memory. BW_quest uses only one parameter search file. Almost all operations were reading.

In BW_quest, in an obvious way, use level of memory is reduced. Total time for the execution of BW_quest is 472s . Execution time is reduced to since BW_build is executed one time and intermediate results remain unchanged as reference data remains unaltered. Therefore, in order to evaluate and measure the execution time in the new solution (BW_quest) we divide the process in three parts:

QuestA - Download source files time from SE to WN and have data ready for run BW_quest.
QuestB - Process BW_quest time.
QuestC - Upload result file time. Save result data in SE.

(iii) Measuring time for QuestC. Usually, the size of the resulting file will be directly depending on the size of the input data. Thus, in our case the result file will be of a reduced size. If we have the same parameter on both sides of the equation then we can disregard it. BW_search and BW_quest have the same result report. We always consider this time zero.

(ii) Measuring time for QuestB. QuestB time is the execution of the algorithm in a single processor without file transfer. This step measures algorithm speed compared to the original algorithm. Using the same testing conditions, BW_search execution time is 472s . Time improvement becomes 45% better. Total executing time with this enhancement is around 55% of the previous total executing time. If we pay attention, the profit occurs if the creation of the intermediate data file only occurs once, and it keeps these data in a non-volatile structure for reuse. A large amount of time was used to create search and sort Burrows-Wheeler Transform matrix.

(i) Measuring QuestA time. This is the time required to move input data files from SE to WN. In our proposal, we downloaded all files required to run the process. Intermediate files always have a total size around 4 GB. In order to measure QuestA time, it is necessary to define a proper strategy. Different strategies are being considered in different experiments and results are shown in the following sections and in the table 1.

Strategy 1. Sequential download of files. We tested two source locations. The nearest CE, sometimes referred to as the CloseSE, and other node with a good response time. As expected, the result with the files stored in the closeSE are better. The result is a download time of 259 seconds ending up with a total execution time of 731 seconds (259+472) for group QuestA+QuestB time.

Strategy 2. Get compressed files and uncompress them before running. A second approach to the solution is to compress the files before downloading them. In this case, QuestA time includes both downloading and uncompressing time. Gzip UNIX command is used for uncompressing step. Results are

similar to the ones in the previous table but with lower time. The total execution time: 364s (115 download + 250 uncompress) added to execution time is 836 seconds (364+472).

Strategy 3. Get files using multi source access from two or more source locations. We used the multi-stream lcg-cp option but we could not make any download from distant nodes. We note that multi-stream lcg-cp download requires certain port range open in the firewall on the destination host. The performance is slightly better in this case. Total execution time was 726s (254+472).

Strategy 4. Parallelize file download. Reference data can be divided in two or more parts, which can be manually downloaded in parallel to reduce total time. For testing purposes, in this case we downloaded every file at the same time using parallel job running lcg-cp process. Total time 685s (213+472).

Strategy 5. Select best SE. We have replicas in several geographically distributed SEs. Keeping historical records and selecting the fastest node for downloading is a right strategy. This approach is described in the second experience where we try to find a way to establish the best download node.

Strategy 6. Access the file remotely using GFAL, avoiding downloading the whole file. The best way to reduce the downloading time is to reduce the total number of downloaded bytes at QuestA and try to access remotely to only the required information.

Table 1. Strategies estimated times downloading to ngiesui.i3m.upv.es

Strategy	Streams	from SE ngiesse.i3m.upv.es	from SE se01.dur.scotgrid.ac.uk
1	1	259s	441s
2	1	364s	474s
3	2	255s	—
3	5	254s	—
3	10	261s	—
4	1	213s	339s

4.2 A second experience. Describing the 5th strategy

Files used as intermediate storage system increase the data time access and made file access one of the purposes of the improvement process. We introduce a second idea, file replication. When there are many replicas in grid, a copy is most likely to be available. This idea is normally used in grid jobs to reduce bottlenecks. The files were distributed along SEs nodes and the job running in a WN locates and transfers files to a local storage space.

To evaluate access time and availability of a file in the biomed VO SEs, we designed a test that downloads replicated files from SE and save access time and total errors.

We measured access time as the SE response time and its bandwidth. The experiment is extended reducing the size of the file and therefore decreasing the requirements on bandwidth. For testing purposes, files measure 4 bytes, in that way, the infrastructure is evaluated of speed in communication lines instead. The experiment consists of performing several file downloads using lcg-cp command. This command tries to download the file from different servers and it applies a recover algorithm to download them from other server if some error occurs.

File was replicated in six SEs selected by their distance to running environment (grid130.sinp.msu.ru, ngiesse.i3m.upv.es, se001.ipp.acad.bg, se01.marie.hellasgrid.gr, svr018.gla.scotgrid.ac.uk and tlapiacalli.nucleares.unam.mx).

Default time-out values for lcg-cp command were: for BDII access 60s, for connection 60s, for total time 3,600s (1 hour) and send or receive ratio of 60 blocks. The number of errors is reduced with these values. Lower time-outs led to higher error rates.

After the test run, it is observed that error rates are randomly appearing on different machines. At 05/01/2013 to 05/07/2013 week transfers time between SE an UI has an error ratio of 15-18%. However, during the following week error ratios fell down to zero. In some cases, time access errors have been caused not by the SE slowness but by the BDII. If we are more demanding in BDII performance, then total errors will raise. We note that response time is not a function of the physical proximity and is not a constant during the experiment life-span.

The conclusion of this simple experiment is that if we would like to consider the response speed of the node containing a file replica, the values must be dynamic and they must be updated frequently. Also, it indicates that the measurement of access time in the transfer of files is not constant.

4.3 A third experiment. Explaining 6th strategy

In this third experiment we try to direct access to remote data. Before that, in order to learn more about BW_quest I/O behaviour, we did some tests. Using iostat Linux command we measured I/O access for BW_quest. Analyzing data obtained, it highlights that read access are grouped around a short time-frame.

In to clarify what happens in the read I/O period, we have altered original code for previous experiment BW_quest. We need to known read positions executing therefore we altered code to report I/O from local access saving it in a log file. We selected only accesses to the largest file (file_o.txt) and, in order to show data after capture, we created groups of total access. The group size is 4,000 integers. This means that if job reads an integer from 1 to 4,000 positions then this access is represented in the first column and also the remaining ones.

We created 254,235 blocks. Figure 3 shows the number of block accesses. A detailed analysis shows that the execution uses a small number of blocks. A total of 24,541 blocks are accessed from a total of 254,235. Only the 9,6% of the blocks

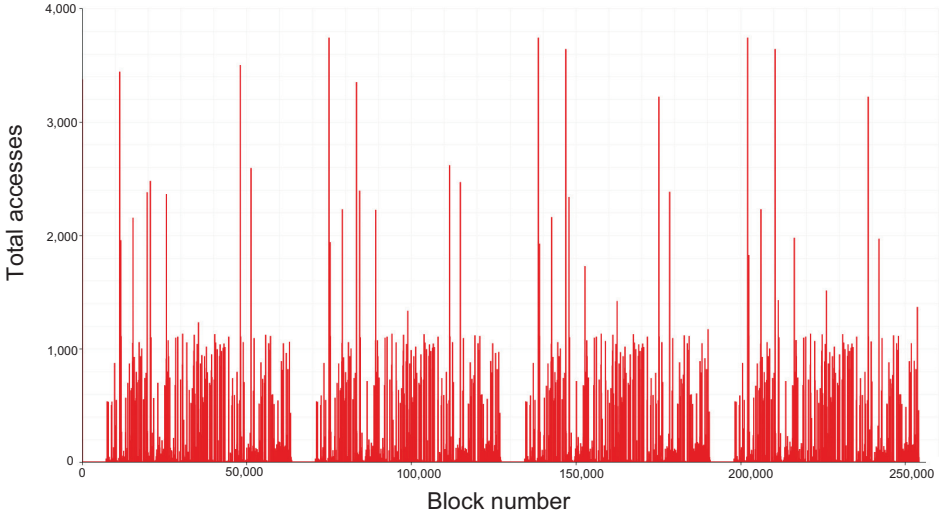


Fig. 3. Total accesses by block (Sequence A).

are used. In the case of the intermediate file, only 364 MB out of 3,7 GB are needed for the execution. We also note a replication pattern that exists in the blocks access. There are four access big groups that have a similar pattern.

Finally, we try to access remotely. A 4 GB transfer has approximately 254s cost tested in EGI infrastructure that it is added to compute time (QuestB time). At this point, we would suggest not to download all files prior to calculation, but access the data remotely. In order to exploit this feature, we altered BW_quest to access directly to data file using GFAL API. When we use gfal_seek and gfal_read commands for data access, it is not necessary to download the file. This reduces files download time (QuestA time) but it increases drastically QuestB time. Capture time, for one search, extends up to more that 1,500s.

The table 2 summarize the behaviour for each strategy.

Table 2. Estimated access time for one sequence alignment.

	QuestA	QuestB	Total
BW_search	–	–	864s
Strategy 1 - Sequential	259s	472s	731s
Strategy 2 - Compressed	364s	472s	836s
Strategy 3 - Multi-stream	254s	472s	726s
Strategy 4 - Parallel	339s	472s	685s
Strategy 5 - Best SE	200s	472s	672s
Strategy 6 - Direct Access	–	1,500s	1,500s

The best way to make out access pattern for sequences alignments is to compare two similar sequences. We choose two similar sequences, A and B, it shares the 32 first bases, from position 1 to 32 was the same, position 33 and later was different in some places. In table 3, we compare some data from the two accesses. If we run sequentially, first A and then B, we note that total blocks read by the two processes were 599 and only accessed by one process are 23,942 blocks. All blocks used by B sequence was accessed previously by sequence A. All blocks have not the same number of accesses. The worst case scenario would be a unique access per block. We measure this option in table 3 (see Accessed only one time row). When we calculated the total number of blocks accessed more that one time in Seq. A, we note that total blocks accessed more that one time is greater that accessed only one (see Accessed blocks gt 1 row).

Table 3. Blocks accessed from sequence A and B.

	Seq. A	Seq. B
Total Blocks	254,235	254,235
Accessed Blocks	24,541 (9.6%)	599 (0.3%)
No accessed Blocks	229,694 (90.3%)	253,636 (99.7%)
Accessed only one time	1,996 (0.8%)	63 (0.03%)
Accessed blocks gt 1	22,545 (99.2%)	536 (99.97%)

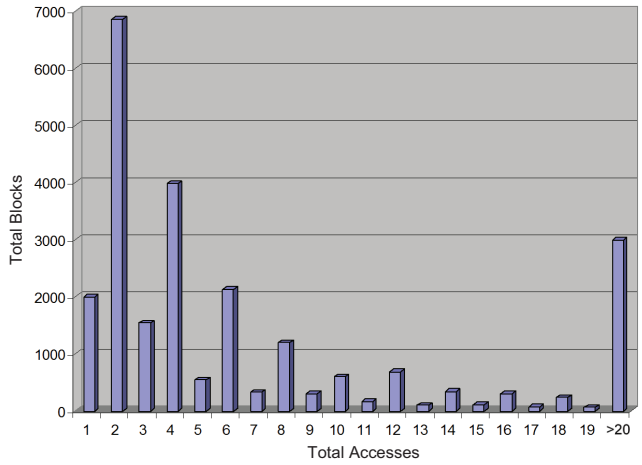


Fig. 4. Blocks grouped by total accesses executing Sequence A.

5 Conclusions

The article shows the susceptibility of improving applications that require accessing small parts of huge data files. The selection of the rightmost location of the data and the use of caching may have a clear impact on their performance. This is the case of BWT-based alignment methods. Using previous BWT search code we produce a faster new version. To apply this solution is necessary a quick file transfer. We tested several strategies and measure downloading times. Transfer time may vary depending on infrastructure conditions (see section 4.2) and this will alter search total time.

By analyzing access pattern in intermediate file we can highlight that only a reduced group of blocks are used. At that time we suggest direct access to remote file. This choice was more expensive that previous ones using full file downloads, but a new system based in temporal and spatial locality references is suggested. A new layer above GFAL can reduce total access time and data transfer needs. This architecture will provide a new two level cache memory. Next step will be tuned up parameters and development of mock-ups to evaluate the proposed architecture.

References

1. Burrows, M. and Wheeler, D.J.: "A block-sorting loss-less data compression algorithm." Technical Report 124, DEC SRC, 1994.
2. Genome Reference Consortium Human Reference 37 HG19. http://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/
3. Celniker, S.E. and Rubin, G.M.: "The Drosophila melanogaster genome.", Annu Rev Genomics Hum Genet. vol. 4, pp. 89-117, 2003.
4. R. Li and C. Yu and Y. Li and T.-W. Lam and S.-M. Yiu and K. Kristiansen and J. Wang, : "SOAP2: an improved ultrafast tool for short read alignment.". J. Bioinformatics, vol 25. num. 15, pp. 1966–1967, 2009.
5. High Performance Genomics Aligner. http://docs.bioinfo.cipf.es/projects/hpg-aligner
6. Langmead, B.;Trapnell, C.; Pop, M. and Salzberg,S.L. : "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.". Genome Biology, vol. 10, number R25, 2009.
7. Homer, N.; Merriman, B.; Nelson, S.F.: "BFAST: An Alignment Tool for Large Scale Genome Resequencing".PLoS ONE, Ed.Chad Creighton, vol. 4, issue 11, p. e7767. 2009. http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0007767
8. BFAST- Blat-like Fast Accurate Search Tool. http://sourceforge.net/apps/mediawiki/bfast/
9. Salavert, J.; Blanquer, I.; Tomas, A.; Hernandez, V.; Medina, I.; Tarraga J. and Dopazo, J.: "Using GPUs for the Exact Alignment of Short-Read Genetic Sequences by Means of the Burrows-Wheeler Transform.". IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 9, number 4, issn 1545-5963, pp. 1245-1256. 2012.
10. Li, H. and Durbin, R.: "Fast and accurate short read alignment with Burrows-Wheeler transform.". Bioinformatics, pp. 1754-1760, vol 25, number 14, 2009.

11. Ferragina, P. and Manzini, G.: "Opportunistic Data Structures with Applications.". FOCS, pp. 390–398, 2000.
12. lcg-cp(1) - Linux man page. http://linux.die.net/man/1/lcg-cp
13. Grid Cache Server site. http://sourceforge.net/p/gridcacheserver/home/Home/
14. Mogul, J. et al. : "Delta Encoding for HTTP". RFC 3229. 2002. http://tools.ietf.org/pdf/rfc3229.pdf.
15. Koller, R.; Rangaswami, R. : I/O Deduplication: Utilizing content similarity to improve I/O performance.. Trans. Storage 6, 3, Article 13 (September 2010), 26 pages. http://doi.acm.org/10.1145/1837915.1837921
16. ES-NGI site. Web: http://www.es-ngi.es
17. Frohner, Á.; Baud, J. P.; Rioja, R. M. G.; Grosdidier, G.; Mollon, R.; Smith, D. and Tedesco, P.: "Data management in EGEE". Journal of Physics: Conference Series. Vol. 219, No. 6, p. 062012. IOP Publishing. April 2010.
18. Brooke, J.; Fellows, D.; Garwood, K. and Goble, C.:"Semantic matching of grid resource descriptions".Grid Computing. pp. 240-249. Springer Berlin Heidelberg. January 2004.
19. CERNVM web site, http://cernvm.cern.ch/portal/filesystem

Session IV

Scientific Clouds

Service Construction Tools for Easy Cloud Deployment

Jorge Ejarque[1], Anthony Sulistio[2], Francesc Lordan[1], Pierre Gilet[2], Raül Sirvent[1], and Rosa M. Badia[1,3]

[1] Grid Computing and Clusters Group, Barcelona Supercomputing Center (BSC), Spain
[2] Dept. of Service Management & Business Processes, HPC Center Stuttgart (HLRS), Germany
[3] Artificial Intelligence Research Institute - Spanish National Research Council (CSIC), Spain
{jorge.ejarque, francesc.lordan, raul.sirvent, rosa.m.badia}@bsc.es, {sulistio, gilet}@hlrs.de

Abstract. Developing services for cloud computing can be a tedious work for developers which are not experts on distributed computing. They have to select the infrastructure provider, adapt the application to the provider specifics, build custom images for running virtual machines, and use the provider's API to deploy the application. In this paper, we present the OPTIMIS service construction tools, which aim at facilitating the service development for the cloud. They consist of a Programming Model, which provides a infrastructure unaware way to implement cloud services; an Integrated Development Environment, which provides a graphical user interface to develop services and automatically runs all the processes to build an deploy them based on the information specified by the user at implementation time; and an Image Creation Service, which provides the creation of customized images. In this work, we also present a Gene Detection application that has been implemented using the OPTIMIS construction tools. This use case exemplifies how these tools take away a lot of the technical complexities and decisions regarding to the implementation, deployment and operation of services in the cloud.

1 Introduction

One challenge posed by cloud computing is that the developer writing services aimed to be run in a cloud environment must bear the specifics of such an environment in mind. For instance, he/she has to look for the right infrastructure provider, come up with an agreement with that provider, deploy the service, let it run, monitor its execution, and so on. All of these tasks come at cost and they are time consuming. Small and Medium Enterprises (SMEs) may not always have the right budget nor sufficient time to develop new services for a cloud environment. Hence, the need has arisen to try to simplify as much as

possible the development and deployment of a service aimed at running in a cloud environment.

The Optimized Infrastructure Services (OPTIMIS) project [1] addresses the aforementioned problem by developing various service construction tools for the cloud, including Programming Model (PM), Integrated Development Environment (IDE), and Image Creation Service (ICS). These tools take away a lot of the technical complexities and decisions regarding to the deployment and running of services in the cloud. Thus, by using these tools, developers can focus more on the business processes to be implemented, without having to worry too much about technicalities relating to service deployment and execution in a cloud environment.

The rest of this paper is organized as follows. Section 2 describes a brief overview of the OPTIMIS project, whereas Section 3 mentions the OPTIMIS construction tools for an easy cloud deployment. Section 4 presents a case study that highlights the benefits of these construction tools. Section 5 describes related work. Finally, Section 6 concludes the paper and proposes future work.

2 The OPTIMIS Project

The OPTIMIS project aims at optimizing cloud services in the Infrastructure as a Service (IaaS) level by producing an architectural framework and a development toolkit. The optimization covers a full cloud service lifecycle, i.e. service construction, cloud deployment and operation. The OPTIMIS Toolkit comprises a set of tools to be used by Service Providers (SPs), Infrastructure Providers (IPs), Software Developers (SDs), and end users. Therefore, the OPTIMIS Toolkit gives the SPs the capability to easily orchestrate cloud services from scratch, run legacy applications on the cloud, and make intelligent deployment decisions based on their preferences with regards to trust, risk, eco-efficiency and cost (TREC) parameters. Moreover, it supports an end-to-end security, and complies with data protection and green legislation. It also gives the SPs the option to develop these services only once and deploy them across all types of cloud environments like private, hybrid, federated or multi-clouds.

The OPTIMIS Toolkit can be broken down into three main groups of components, as seen in Figure 1:

- The OPTIMIS Base Toolkit with functionalities that are common to all components;
- The OPTIMIS SP Tools that enable the SPs to implement, package, deploy and operate services; and
- The OPTIMIS IP Tools with the functionality to manage the cloud infrastructure (e.g. virtual machines (VMs), servers, data storage, etc.) for operating these services.

The OPTIMIS Base Toolkit helps to make an optimal service deployment and operation decisions, and provides fundamental services like monitoring and security. It contains a Monitoring Infrastructure (MI) component that delivers

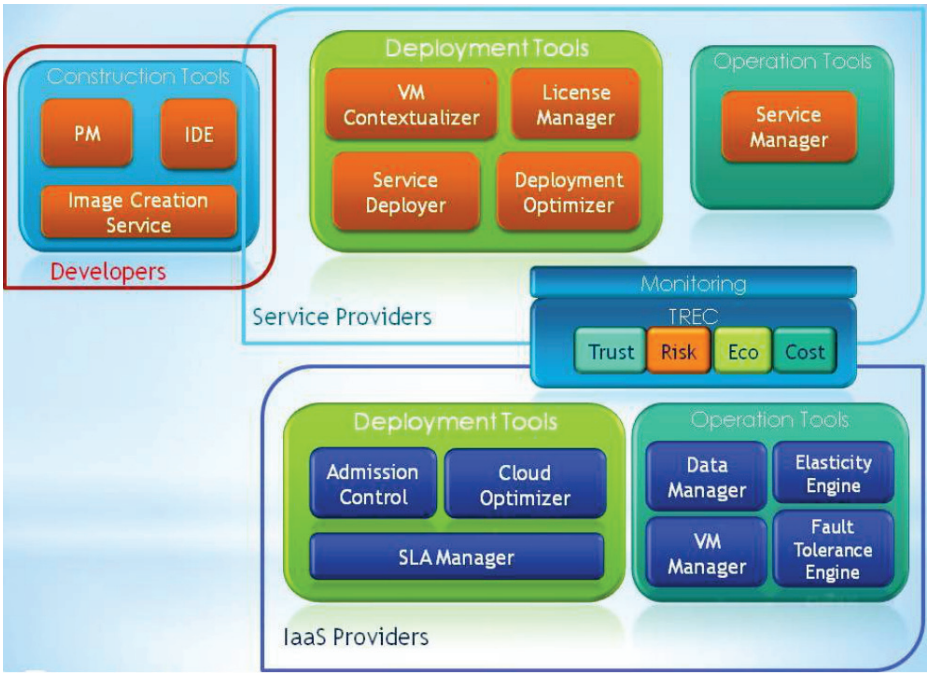


Fig. 1. An overview of the OPTIMIS Toolkit.

runtime information about physical hosts, VMs, energy consumption, and service performance, and a set of components to assess the TREC parameters.

The OPTIMIS Toolkit allows IPs to effectively and efficiently manage infrastructure resources at a higher level of abstraction, and enables SPs to create, deploy and operate services with assessed and guaranteed TREC levels. However, this paper is mainly focused on the OPTIMIS construction tools, i.e. Programming Model (PM), Integrated Development Environment (IDE), and Image Creation Service (ICS). For more details on other OPTIMIS components, please refer to the OPTIMIS publications [2].

3 OPTIMIS Construction Tools

The OPTIMIS Construction Tools are developed in order to help a cloud developer to write and deploy services without having to worry too much about cloud technicalities. The construction tools consist of Programming Model (PM), Integrated Development Environment (IDE), and Image Creation Service (ICS). The PM allows the creation of complex services by composing pieces of source code, licensed software, legacy applications and services. The IDE automates and simplifies the implementation of new complex services by providing a Graphical User Interface (GUI) where these can be programmed. The ICS automates the generation of service VMs according to service requisites.

Figure 2 depicts the overall sequence on how these construction tools interact and work. A service developer mainly interacts with the IDE in order to build a service according to the syntax described in the PM. Based on the implementation done by the user, the IDE defines the types of VMs required to execute the service, creates VM images according to the hardware constraints, and installs different application packages and the PM runtime on these images. Finally, the IDE creates a service manifest, which describes the service VMs and other deployment requirements, and uses the OPTIMIS deployment tools to deploy the application in the cloud infrastructure. At operation time, the PM runtime detects the different executions done by the application users, and interacts with the OPTIMIS operation tools to adapt the cloud infrastructure according to the application demand.

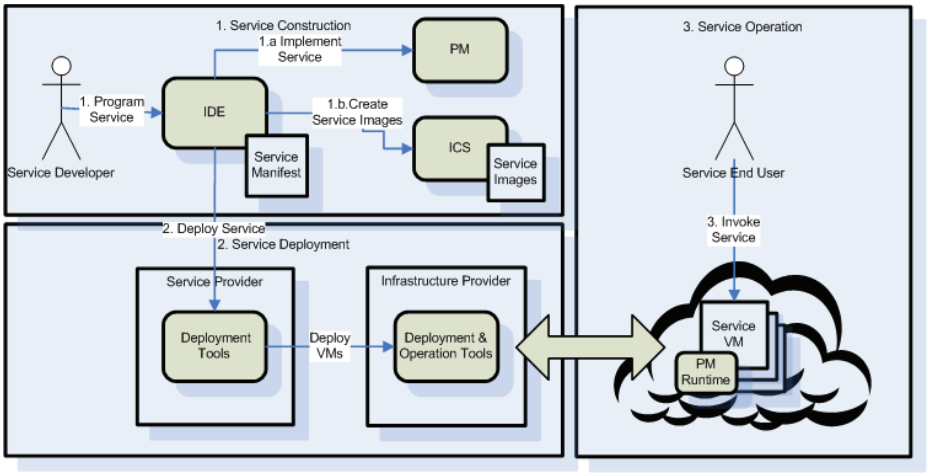


Fig. 2. Optimis Construction Tools overview

3.1 Programming Model

The OPTIMIS Programming Model (PM) [3] is a new approach for an easy development of cloud applications as *composite* services, from which other services and regular methods are called. Composites are codes that reuse functionalities wrapped in services or methods, adding some value to create a new product that can also be published as a service. The most important idea behind the PM is to offer to the developer the possibility of writing a service as a sequential code, that is completely unaware of the underlying infrastructure by executing its components in parallel on top of the resource pool deployed on the cloud.

The PM is defined as a dependency-aware task-based model. The developer selects which parts of the composite (Orchestration Element) become a task

(Core Element). Thus, the OPTIMIS PM services can be composed by two different kinds of Core Elements (CE): Method CE and Service CE. On one hand, Method CEs are regular methods of the application selected to be run remotely. On the other hand, Service CE corresponds to SOAP web service operations described in the WSDL documents [4]. The selection of CE is done by means of an interface called Core Element Interface (CEI). There, the service developer defines all the services and methods along with a set of metadata describing the CE invocation and its data accesses. Optionally, the programmer can indicate the features required to the executing host.

Figure 3 contains a CEI example with two CEs declared: *update* and *sampleService*. *Update* corresponds to a method CE implemented in the *sample.Example* class with three parameters: *option*, an integer that is read; *value*, an object that is modified; and *log*, a file created along the task execution. The *@Constraints* annotation on line 2 restricts its execution to resources with more than 4 cores. The other CE defined in the *SampleCEI* corresponds to a SOAP web service operation in the service *WSName* service detailed in the *@Service* annotation. The *sampleOperation* CE operates on two pieces of data: a *Query* object read by the service and the return value of the operation: a *Reply* object.

```
1   @Method(declaringClass = "sample.Example")
2   @Constraints( processorCPUCount = 4)
3   void update(
4       @Parameter(direction = IN)
5       int option
6       @Parameter(direction = INOUT)
7       Reply value
8       @Parameter(type=FILE, direction = OUT)
9       String log );
10
11  @Service(namespace="http://servicess.com/example", name="WSName", port="WSPort")
12  Reply sampleService(
13      @Parameter(direction = IN)
14      Query query );
```

Fig. 3. Sample Core Element Interface. The *update* method is designated as a method CE implemented in the *sample.Example* class. (*sampleService*) is declared as Service CEs linked to the *samplePort* port of the *sampleWS* service.

Once the service is deployed in the cloud and running, a runtime system is in charge of orchestrating the execution of the service CEs distributing the computation among a pool of resources. In order to achieve this goal, the runtime is composed by two main components: the Task Processor (TP), in charge of managing the data dependencies; and the Task Dispatcher (TD), responsible for the actual execution of the tasks in the resources. Since a single service may have multiple endpoints, an independent instance of the TP is deployed on every server ensuring the sequential consistency for the OE invocations executed on it. When an OE is invoked, its sequential code runs in the endpoint server creating tasks as the code is executed. When the local TP receives a new task, it analyzes each parameter looking for data dependencies according to its description in the

CEI and adds the task into a dependency graph automatically generating a workflow of tasks.

Once all predecessors of a task have been executed, it becomes free of dependencies and is submitted to a central TD which schedules and monitors every task execution in a remote resource. TD manages all the VMs deployed in the cloud as a single resource pool. It picks one of them to run the task taking into care the specific CE constraints and submits its execution to the VM through SSH.

Besides using cloud resources to execute tasks on the deployed VMs, the runtime system also takes advantage of the cloud by exploiting its elasticity feature. TD monitors constantly the workload of the whole service taking into account the current number of Orchestration Elements (OEs) invocations, the number of CEs ready to be executed and the average execution time to run each CE, and compares it with the available capacity of the system. If the workload is higher, new VMs are instantiated to face the workload excess. Symmetrically, when the resources are underused, some VM instances are freed to reduced the economical cost of the execution.

3.2 Integrated Development Environment

The Integrated Development Environment (IDE) provides a graphical interface for facilitating the construction of services following the PM syntax (described in Section 3.1) and automating the building and deployment of these services in the cloud infrastructure. The IDE is implemented as an Eclipse [5] plug-in, and extends the Java Development Tools with a Service Editor and a set of wizards and actions. These tools generate the code and classes for the different service elements, compile and build the application packages, and deploy the service in an easy way. With the IDE, Service Developers mainly interact with the Service Editor, which guides them during the different service construction phases. It is composed of two tabs: the *Implementation* tab, which contains actions and wizards for development operations, and the *Build and Deploy* tab which contains the widgets for preparing the service for deployment.

The first tab (Figure 4) provides an overview of the service implementation status showing the OEs and CEs defined for each service class according to the PM syntax. Moreover, it allows developers to create new OEs and CEs from scratch or to import from existing software like jar libraries, binaries or services. For each of these processes, the developers will be guided by a set of wizards, which will request the required information to create the service elements as well as introducing constraints like the required hardware resources, software dependencies and elasticity boundaries. At the end of the wizard executions, all the annotations and code required by the PM syntax will be automatically created by the IDE. Therefore, at the end, the service developer only has to write sequential Java code.

Once the service implementation has been finished, developers move to the *Build and Deploy* tab (Figure 5) to build the service packages defining how the service elements are grouped. At this stage, the IDE can work in two modes: the

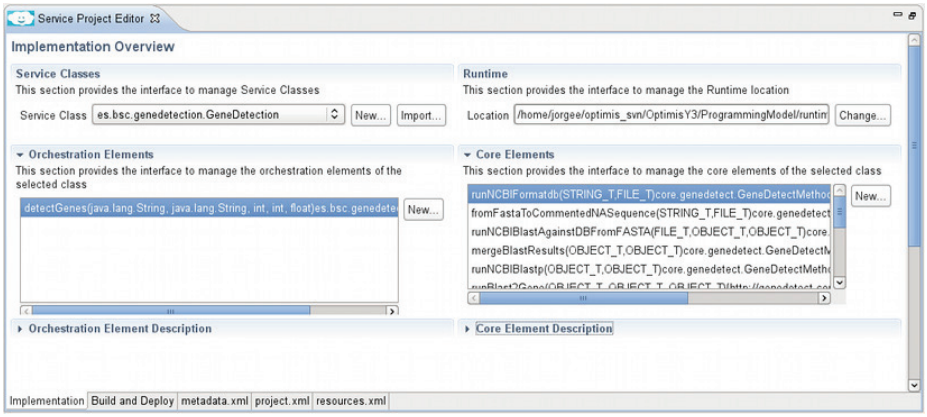


Fig. 4. IDE Service Editor 'Implementation' tab screenshot

automatic mode, where OEs and CEs are automatically grouped according to the defined constraints and minimizing the number of packages; and the manual mode, where developers select which service elements are deployed together. Once the elements have been grouped, the application code is compiled and instrumented. This process takes as input the class of the service (containing the OEs) and the annotated CEI, where CEs were selected. With the help of Javassist [6], a java library for class editing, the IDE replaces the original calls to the selected CEs by asynchronous calls to the PM runtime and inserts code to trigger data synchronization at OE level. The building process is finalized by packaging the OE, CE and the PM runtime libraries in different war and jar files depending on the type of OE and CE (services or libraries).

After the package creation, the IDE provides a deployment widget which allows developers to deploy the application either in the localhost, to test and debug the application, or in a production cloud infrastructure. For the localhost deployment, the IDE simulates the deployment environment in the developer's machine installing the CE packages and publishing the services with the PM runtime in a service container. In the case of a cloud deployment, the IDE will generate a service manifest with a description of a type of VM for each of the created packages, taking into account the resource constraints specified by the user. Afterward, it contacts the ICS to create the corresponding image for each of these VMs installing the service packages inside. Moreover, the developer can define with the IDE the most appropriate cloud deployment by defining a set of constraints like the minimum levels of TREC parameters of the potential provider, as well as some legal constraints like the provider location or the intellectual property rights claim. Finally, developers can start the deployment process by just clicking the *deploy* button in the *Build and Deploy* tab. Once the service is deployed, whenever a request to one of the service OEs is received by the deployed service container, the execution of that OE starts and the PM

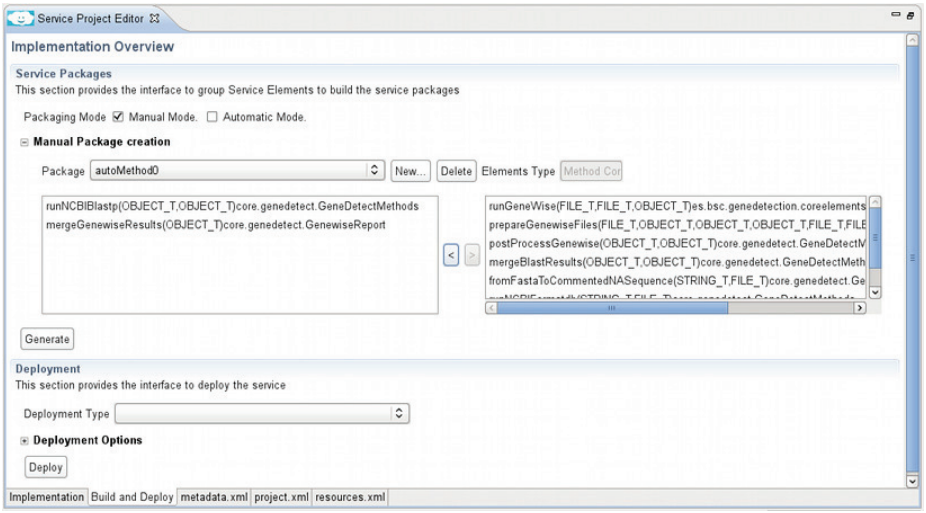


Fig. 5. IDE Service Editor 'Build and deploy' tab screenshot

runtime will orchestrate its execution in the different virtual resources deployed for the service.

3.3 Image Creation Service

The Image Creation Service (ICS) component is developed for the OPTIMIS project using RESTFul web services to create custom VM images. Firstly, a base image is selected from the repository according to a service manifest or a set of requirements, such as operating system (e.g. Ubuntu or CentOS), architecture (32- or 64-bit), and image size (e.g. 5, 10, or 15 GB). Then, an image is cloned from the selected base image and customised by adding war or zip files, setting permissions of files in the image, or extracting archives to an image.

The objective of ICS is to address interoperability issues by means of developing an image manipulation service that can be deployed in various cloud environments. Another issue tackled relates to the need of customizing the selected image before it is being deployed. For example, a pre-installation of Apache Tomcat and addition of WAR files are conducted by ICS before the image is being finalized and deployed in the cloud. This allows automated installation steps and the release of an image ready for usage right after it has been deployed. As a result, manual installation steps can be avoided.

4 Use Case Scenario: Gene Detection service

In order to better illustrate the advantages of the OPTIMIS service construction tools, a real-world Life Sciences application has been ported by the OPTIMIS

PM and executed it on a cloud testbed. This section describes the Gene Detection algorithm and the required steps to implement it with the objective of running it in the cloud, highlighting the simplicity of the proposed solution.

The Gene Detection algorithm [7], developed by the Life Sciences department at the Barcelona Supercomputing Center, is a program to identify the relevant genes in a genomic DNA sequence. It is mainly implemented on top of BLAST [8] and GeneWise [9] programs. The application finds a set of relevant regions in the DNA sequence, and then runs the GeneWise program for only on those regions, which is faster than scanning the whole DNA. An overview of the main Gene

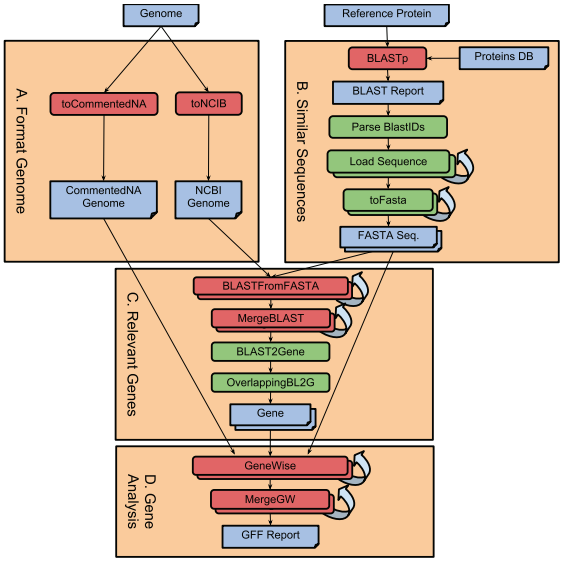


Fig. 6. Gene detection workflow. Service CE invocations in green, while method CE invocations in red.

Detection workflow is depicted in Figure 6. Each box represents a different part of the application which contributes to the overall process: translation of the input genomic DB to a given format; obtention of a list of aminoacid sequences which are similar to a reference input sequence; and search of the relevant regions of the genomic database and execution of the GeneWise algorithm on them.

The Gene Detection service has been ported following the next steps with the *Build and Deploy* tab of the IDE: first, create a service class and an OE to implement the main algorithm code. Then, define the different CEs, importing the services CE from the WSDL of the Bio-informatic services, generating the Method CEs from the Blast and GeneWise binaries, and selecting the merge methods from Jar libraries. In addition to the interface of the CE, a set of OS and processor architecture constraints have been defined for the CE which invokes Blast and Genewise binaries. These constraints ensure that the resources

used for invoking these methods are compatible with the binaries compilation. As a result of this part, the IDE has created the CEI required by the PM. The last step is implementing the OE as a sequential application that invokes the selected CEs. Figure 7 shows a snippet of the Gene Detection code and the CEI after the implementation. Note that the Gene Detection OE is programmed sequentially and no cloud APIs are used and either service or method CEs are called as any other regular method.

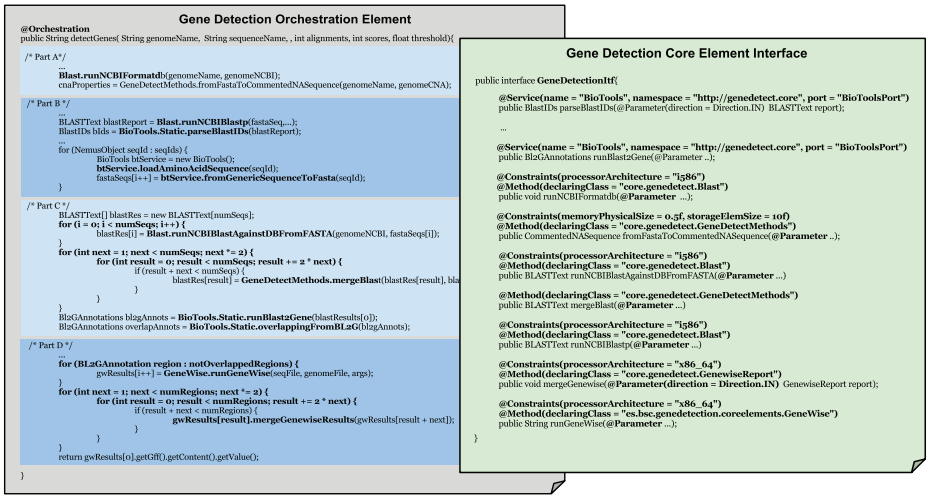


Fig. 7. Orchestration Element and Core Element Interface for the Gene Detection.

After the implementation, the Gene Detection service packages are created with the automatic mode. Due to the CE constraints, the IDE creates one package for the OE and two for the CEs, one which includes the method core elements which requires 64-bit architecture and another with the CE which requires 32-bit architecture. This constraints is also taken into account to select the templates for creating the service images.

Once the service is deployed, the PM runtime is intercepting the OE calls and generate a task-dependency graph with the different CE calls. Figure 8 shows the task-dependency graph generated by the execution of the OE code where each node represents a task (CE invocation) and each color represents a CE type. Note that there are parts where a certain parallelism can be achieved. In these parts, the PM runtime contacts the OPTIMIS IP Tools to deploy new VMs in order to speed up the overall execution, and undeploy them when the possible parallelism decreases.

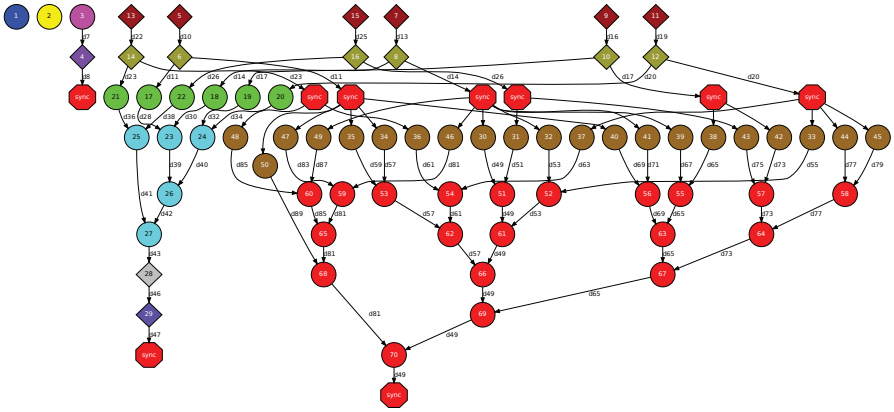


Fig. 8. Graph generated for a Gene Detection execution. The different colors for circles and diamonds represent the different invocations of method CEs and Service CEs. Arrows represent data dependencies and stops represent synchronization points

5 Related Work

Many Platform as a Service (PaaS) solutions have appeared to facilitate the process of developing, deploying and running applications in the cloud. Some of them propose programming models that offer APIs (for writing the applications), and graphical environments to be used as programming tools. For the Microsoft Azure's cloud PM, applications are structured in roles, which use APIs to communicate (queues) and to access persistent storage (blobs and tables) [10]. The .NET environment needs to be used to program and prepare a package that is to be deployed in the Azure compute nodes. On the other hand, Google App Engine [11] provides libraries to invoke external services and queue units of work (tasks) for execution. Furthermore, it allows to run applications programmed in the MapReduce model. They also provide a plug-in for the Eclipse environment.

Most of cloud middleware (e.g. Emotive [12], OpenNebula [13] or OpenStack [14]) have limited image management functionalities. They only provide a basic image repository for storing image templates that will be used to deploy a new VM. The CernVM project [15] also provides basic images for running Large Hadron Collider (LHC) experiments. Then, a special read-only network file system is used to access software located on one or more servers.

The main difference between our service construction tools and the above works is the underlying programming model (OPTIMIS PM) and the automatic image creation for service deployment. The OPTIMIS PM does not require including any API calls in the application code. The CE creation (either from regular methods or services), data transfer and synchronization are handled automatically by the PM runtime. Moreover, data dependencies between CEs do not need to be managed manually in the application code, since they are resolved

by the PM runtime. Users of Microsoft Azure and Google App Engine must create the images manually based on pre-existing templates. On the contrary, ICS automatically finds a suitable image depending on the application requirements, and allows service developers to upload and modify the image before it is being deployed on the cloud. Finally, the aforementioned PaaS proposed by Microsoft and Google restrict the deployment and execution of their applications to their own infrastructure. In contrast, our Programming Model can potentially work on top of any cloud providers.

Cloud applications could be also developed directly with parallel programming model frameworks or workflow editors like (Java threading, MPI [16] or WS-BPEL [17]), However, these technologies require special programming background, while the OPTIMIS PM only requires skills in sequential programming; no knowledge in multi-threading, parallel/distributed programming or service invocation is necessary. More details about the evaluation of the Programming Model can be found at [3].

6 Conclusions and Future Work

In this paper, we have presented how a set of OPTIMIS construction tools, i.e. Programming Model (PM), Integrated Development Environment (IDE), and Image Creation Service (ICS), are working together in order to ease the way an application is developed and deployed in the cloud. As a showcase for the end user, the IDE offers a graphical and easy interface that includes both features of PM and ICS. In IDE, the user has to first develop his/her new application or service before deployment can take place. The first step of the deployment is the generation of application images, which are automatically done by ICS, based on a set of requirements provided by the user. Then, the new application is ready to be deployed in the cloud.

We have also seen in details the features included in the IDE and ICS, and an overview on how to program an application using the PM. This paper also demonstrates a practical use case of these tools, e.g. when programming a new Gene Detection service. From the use case, it can be shown how well these OPTIMIS service construction tools work together, and how developers can take advantage of them.

For future work, we consider extending our approach to run not only for the OPTIMIS Toolkit, but also other cloud middleware, such as OpenNebula and OpenStack. As an overview, this extension will be mainly focused on integrating the ICS with the repository solutions provided by the cloud middleware and adapting the deployment processes to interact with the cloud middleware APIs.

Acknowledgment

We would like to thank Tinghe Wang and Roland Kuebert for their contributions towards the initial Image Creation Service code. This work has been supported by the EU within the 7th Framework Programme under contract

ICT-257115-Optimized Infrastructure Services (OPTIMIS), by the Spanish Ministry of Science and Innovation (contracts TIN2012-34557, CSD2007-00050 and CAC2007-00052), by Generalitat de Catalunya (contract 2009-SGR-980) and by the grant SEV-2011-00067 of Severo Ochoa Program, awarded by the Spanish Government.

References

1. A. J. Ferrer, F. Hernandez, J. Tordsson, E. Elmroth, C. Zsigri, R. Sirvent, J. Guitart, R. Badia, K. Djemame, W. Ziegler, T. Dimitrakos, S. Nair, G. Kousiouris, K. Konstanteli, T. Varvarigou, B. Hudzia, A. Kipp, S. Wesner, M. Corrales, N. Forgo, T. Sharif, and C. Sheridan, "OPTIMIS: A Holistic Approach to Cloud Service Provisioning," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 66–77, 2012.
2. The OPTIMIS project web site. https://www.optimis-project.eu, last access July 2013.
3. E. Tejedor, J. Ejarque, F. Lordan, R. Rafanell, J. Álvarez, D. Lezzi, R. Sirvent, and R. M. Badia, "A Cloud-unaware Programming Model for Easy Development of Composite Services," in *Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science*, CloudCom '11, Athens, Greece, November 2011.
4. Web Service Description Language. http://www.w3.org/TR/wsdl, last access July 2013.
5. "Eclipse: The Eclipse Foundation open source community website." http://www.eclipse.org/, last access July 2013.
6. "Java programming assistant (Javassist)." http://www.javassist.org, last access July 2013.
7. R. Royo, J. López, D. Torrents, and J. Gelpi, "A BioMoby-based workflow for gene detection using sequence homology," in *International Supercomputing Conference (ISC'08), Dresden (Germany)*, 2008.
8. S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, pp. 403–410, 1990.
9. E. Birney, M. Clamp, and R. Durbin, "GeneWise and Genomewise," *Genome Research*, vol. 14, pp. 988–995, May 2004.
10. "Microsoft Azure." http://www.microsoft.com/azure/, last access July 2013.
11. "Google App Engine." http://code.google.com/appengine/, last access July 2013.
12. EMOTIVE Cloud, "Elastic Management Of Tasks In Virtualized Environments." http://www.emotivecloud.net, last access July 2013.
13. OpenNebula, "Open Source Data Center Virtualization." http://www.opennebula.org, last access July 2013.
14. OpenStack, "Open Source Cloud Computing Software." http://www.openstack.org, last access July 2013.
15. "CernVM." http://cernvm.cern.ch/, last access July 2013.
16. Message Passing Interface Forum, "MPI: A message-passing interface standard," Tech. Rep. UT-CS-94-230, PSU, 1994.
17. "OASIS Web Services Business Process Execution Language." http://www.oasis-open.org/committees/wsbpel/, last access July 2013.

Platform to Ease the Deployment and Improve the Availability of TRENCADIS Infrastructure

Damià Segrelles[1], Miguel Caballer[1], Erik Torres[1], Germán Moltó[1], Ignacio Blanquer[1]

Instituto de Instrumentación para Imagen Molecular (I3M), Universitat Politècnica de València, València, Spain
dquilis@dsic.upv.es, micafer1@upv.es, ertorser@upv.es, gmolto@dsic.upv.es, iblanque@dsic.upv.es

Abstract. TRENCADIS is a Grid infrastructure to store and to process large amounts of medical images and its associated data in DICOM objects. This system enables radiologists to effectively group, search and manipulate images and structured reports in order to relate clinical findings and to be of practical value in the diagnosis and treatment of diseases. The paper presents a new platform for the deployment of TRENCADIS infrastructures, using virtualization and Cloud computing techniques. The presented platform avoids intrusive deployment of services to reduce the amount of effort required to install and maintain new TRENCADIS services. Also, it provides mechanisms to monitor and handle performance and reliability requirements by elastically provisioning computational resources from the Cloud to cope with increased demand of the platform.

1 Introduction

Modern medicine cannot be conceived without medical imaging. These techniques play a major role in the diagnosis and treatment of diseases and they are gaining in importance in the prevention and control of epidemics. Hospitals have made large inversions to implement Picture Archiving and Communication Systems (PACSs) and Radiology Information Systems (RISs). These technologies provide storages of medical images, but they are often limited in their access to Distributed Computing Infrastructures (DCIs), such as Grid and Cloud computing environments, which are reducing costs of computing service provision [1], fostering innovative practices and accelerating their adoption by the healthcare professionals [2]. Reaching the level of security that is necessary to ensure the protection from disclosure of the identity of patients is a difficult task. Policy decision-makers have recently become aware of the value of DCIs to improve public health and they are lowering the barriers to Cloud adoption in the European Community [3].

In the last years, many authors have anticipated this trend by developing prototypes that make use of different DCIs to store and process medical images,

e.g. [4]. TRENCADIS [5] is an example of the use of Grid computing infrastructures to store and to process large amounts of medical images, in a secure way. This system enables radiologists to effectively group, search and manipulate Digital Imaging and Communications in Medicine [6] (DICOM) images and structured reports in order to relate clinical findings and to be of practical value in the diagnosis and treatment of diseases. DICOM is the accepted standard format for medical image storage and transfer. Since the introduction of TRENCADIS, several hospitals have reported the value within their organizational context of this technology as a tool for improving the access to DICOM objects [5], and [7]. However, in all these cases the deployment and maintenance of TRENCADIS required a considerable investment of time and effort, mainly because of the intrinsic complexity of Grid systems and also due to the network security restrictions imposed on the hospitals. This fact makes difficult to extend or adapt a particular deployment to meet new challenges, such as an unexpected increase in demand or in the number of images stored in the system. This paper was motivated by these previous results, which provided the basis for new TRENCADIS deployment strategies.

The objective of this paper is to present a new platform for the deployment of TRENCADIS-based applications, using virtualization and Cloud computing techniques. The presented platform avoids intrusive deployment of services to reduce the amount of effort required to install and maintain new TRENCADIS sites. Also, it provides mechanisms to monitor and handle performance and reliability requirements together, allocating and de-allocating different computational resources from the Cloud, such as virtual machines, as required by the applications. Finally, the platform was designed to be portable to other application domains that also rely on Grid computing.

In the case of reliability, the approach in this paper is to use proactive redundancy in the instances of the services that supports critical functions, so that a certain number of faults can be tolerated. In this way, the services are replicated to multiple resources in the Cloud and their management and monitoring is automated to speed-up the creation of new replicas to replace failed ones, so that reliability is maintained.

The rest of this paper is structured as follows. Section 2 analyzes the current tools which allows to automate the deployment of Grid applications to Cloud computing environments. Section 3 presents a comprehensive analysis of the software components and customized required configurations of all TRENCADIS services, from the point of view of their security, availability and performance requirements. Section 4 describes the platform that was developed to automate the deployment of these services on Cloud computing environments, describing the conditions under which the presented platform can be used as a basis for application deployment. Section 5 presents a test deployment to validate the presented platform and to illustrate its capabilities and benefits respect to a traditional TRENCADIS deployment. Finally, section 6 presents the conclusions of the paper, as well as the future lines of work.

2 State of the Art

There can be found works in the literature that aim at easing the process of software deployment on different platforms. For example, Puppet [9] and Chef [10] are open-source configuration management systems that are typically used to deploy applications on provisioned computational resources. This systems can be used to create and manage configurations rules (recipes) that describe a series of resources, such as software libraries that should be installed, services that should be running or files that should be written, on a particular (virtual) machine, thus automating the application deployment process.

With the success of Cloud computing environments like Amazon EC2 [11], many open-source tools have been developed to deploy popular systems to these infrastructures. For example, Apache Whirr [12] is a tool for running Apache Hadoop and related services, such as Cassandra or HBase, in Amazon EC2. In general, although these tools can be adapted to other Cloud providers or other systems (with more or less effort), the use of generic tools seems to be more convenient for our purposes.

In the case of applications that depend on a programming model, such as message-passing, MapReduce or Grid programming models, different approaches are required to deploy the applications. This is a very common requirement, especially in scientific applications. To this end, several platforms support not only the configuration management, but also the specific programming models and languages. CloudFoundry [13] is an example of open-source application deployment system that supports several popular application development frameworks, such as the Spring Framework, Ruby on Rails or Grails. It also provides a set of services to the application developers that includes relational (e.g. MySQL or PostgreSQL) and NoSQL (e.g. Redis) database services, or messaging services (e.g. RabbitMQ). AppScale [14] is an open-source implementation of the Google App Engine [15] Cloud computing technology. It provides Neptune, an extension of the Ruby programming language that supports Message Passing Interface (MPI) and MapReduce programming models. AppScale can be used to automate the configuration and deployment of existing HPC applications to Cloud.

In contrast to "pure" configuration management systems, such as Puppet or Chef, Cloud platforms, such as CloudFoundry or AppScale, have the advantage of integrating support for specific programming models and development frameworks. Often, this support for development that is present in Cloud frameworks limits the flexibility with which the virtual machines can be configured to meet application requirements. For the purposes of this work, this is a serious limitation that makes difficult to configure the services with the desired management and security properties.

On the other hand, to our knowledge, there is no mention on the literature of an efficient and affordable tool which allows to automate the deployment of Grid applications to Cloud computing environments, and their configuration to be used in a secure way, with constraints to minimize the overall system downtime.

3 Analysis of TRENCADIS Infrastructure Services

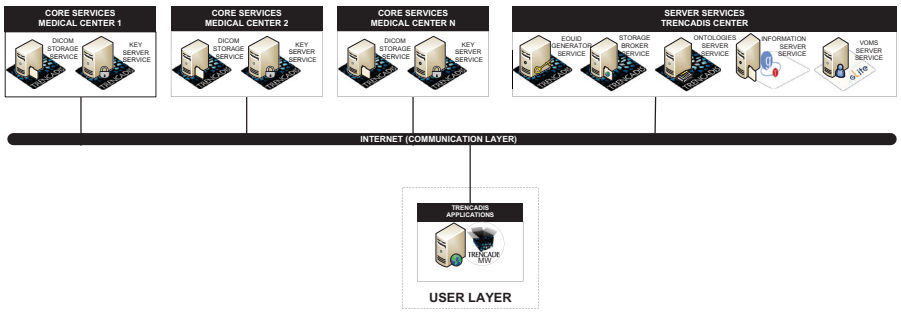


Fig. 1. Deployment model of TRENCADIS infraestructure.

Figure 1 shows the TRENCADIS infrastructure deployment model. This infrastructure is composed by a set of services based on Grid technologies which are integrated in a Virtual Organization (VO). There are two categories, the CORE services and SERVER services. Scientific Linux 6.7 OS is recommended for all of them, although another compatible GNU/Linux OS could be used.

3.1 CORE Services

The CORE services are the DICOM Storage services and the Key Server services. At least one instance of each type should be deployed in each medical center involved in the VO. The **DICOM Storage Service** is composed of the following four software components. The implementation depends on the underlying technology that is used to support the functionality of the component:

- *Base Toolkits*. This component is the base of all services in TRENCADIS that rely on Grid services. It is formed by the Globus 4 toolkit, JDK 1.6, and Ant 1.8. It allows implementing and deploying Grid services. This component does not require a customized configuration.
- *DICOM Storage Grid Service*. This component is deployed on the service container provided by Globus 4. This Grid service uses different versions of APIs to connect to the Indexer and Backend components, depending on how these are implemented. Depending on the version of the APIs used, a customized configuration of each API employed is required. Moreover, it is needed to configure the Grid service for integration in the VO.
- *Indexer*. This component enables to index data contained in DICOM structured reports and can be supported by SQL relational databases (PostgreSQL) or by the Grid component gLite AMGA Server [16]. Currently, support for NoSQL databases, such as Neo4j [18], is being implemented,

although it is not operational yet. This component does not require a customized configuration because it is handled directly through an API by the component DICOM Storage Grid service.
– *BackEnd*. This component stores encrypted DICOM images and associated DICOM structured reports. The backend can be supported by a GridFTP server, Grid components available on gLite (LFC and SE) [17], a File System or a SQL relational database (PostgreSQL). Currently, support for integrating CDMI interfaces [19] is being implemented to use Cloud backends, but it is not operational yet. This component does not require a customized configuration because it is handled directly through an API by the component DICOM Storage Grid service.

The **Key Server Service** is composed of the following software components:

– *Key Server Grid Service*. This component is deployed on the service container provided by Globus 4. This Grid service uses an API for connecting to the SQL Key Database, installed in the backend. Moreover, it is needed to configure the Grid service for integration in the VO.
– *BackEnd*. This component stores the keys used for encrypting/decrypting DICOM data. This is the same component presented above, but in this case it can only be supported by a SQL relational database (PostgreSQL).
– *SQL Keys Database*. This component is a relational database installed into the backend (PostgreSQL). The database has a predefined structure and needs to be created and configured when the backend is deployed.

3.2 SERVER Services

Server services are a set of the five services, which have to be deployed in one center (TRENCADIS Center). This center is external to medical centers involved in the VO. The **EOUID Generator Service** is composed of one software component. This component is deployed on the service container provided by Globus 4 and implements the logical for generating the Encrypted Object Unique Identifier (EOUID). Moreover, it needs to be configured for integrating in the VO. The database has a predefined structure and needs to be created and configured when the backend is deployed. The **Storage Broker Service** is composed of one software component. This component is deployed on the service container provided by Globus 4 and implements the logic for distributing queries among the DICOM Storage services involved and retrieving the results. Moreover, it needs to be configured for integration in the VO. The **Ontologies Server Service** is composed of these software component:

– *Ontologies Server Grid Service*. This component is deployed on the service container provided by Globus 4. This Grid service uses an API for connecting with the SQL Ontologies Database, installed in the backend. Moreover, it is needed to configure the Grid service for integration in the VO.
– *BackEnd*. This component store the Ontologies used for organizing the DICOM data. This is the same component presented in the Key Server service.

– *SQL Ontologies Database*. This component is a relational database installed into the backend (PostgreSQL). The database has a predefined structure and needs to be created and configured when the backend is deployed.

The **Information Server Service (ISS)** is part of the Monitoring and Discovery System (MDS4) of Globus 4. Therefore, it is only composed by the component base toolkits, which have been presented above. Moreover, it requires to configure the service for integration in the VO. The **VOMS Service** is part of the gLite Middleware for Grid Computing [20]. Also it is needed to configure the service for integration in the VO.

4 TRENCADIS Cloud Deployment Platform

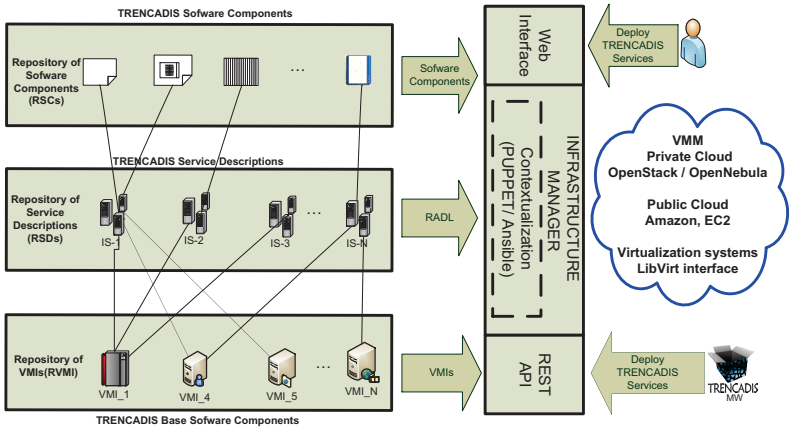


Fig. 2. TRENCADIS Cloud Deployment Platform.

The components of the platform architecture are described in Figure 2 and described in the following subsections:

4.1 Repository of VMIs (RVMI)

RVMI is implemented with the Virtual Machine image Repository and Catalog (VMRC) [21]. VMRC is a software component that enables users to index and store virtual Machine Images (VMIs) together with metadata descriptions about the capabilities of each VMI in terms of CPU architecture, hypervisor for which it was built, OS, applications, etc... This enables users to share and reuse VMIs for different TRENCADIS services. A client-side API is provided in order to query for the most appropriate VMIs that satisfy a given set of both hard and soft

Table 1. Created VMI for TRENCADIS Infrastructure

ID	VMI Name	VMRC Metadata Description
1	Base Toollkits	Scientific Linux 5.7; Globus Toolkit 4.2.1; JDK 1.6.0.35; Ant 1.8.2
2	VOMS Service	Scientific Linux 5.7; VOMS Service
3	PostgreSQL	Scientific Linux 5.7; Postgres 8.4.9
4	LFC	Scientific Linux 5.7; gLite LFC
5	SE	Scientific Linux 5.7; gLite SE
6	File System	Scientific Linux 5.7
7	GridFTP	Scientific Linux 5.7; GridFTP

requirements. Whereas the hard ones must be satisfied by VMIs, the soft ones allow obtaining a ranked list of VMIs depending on the degree of satisfaction of the requisites. For example, one could ask for a VMI created for the KVM hypervisor that has Scientific Linux greater than 5.7, the Globus Toolkit 4 and Java Development Kit (hard requirements) but it would be desirable to have Java 1.6.35 toolkit (soft requirement). Therefore, the VMRC enables to catalog a set of base VMIs from which other VMIs can specifically customized in order to fit a particular deployment.

Table 1 lists the created VMIs for deploying TRENCADIS infrastructures and their VMRC metadata descriptions. These VMIs contain the base software components that can be reused to deploy the different TRENCADIS services.

4.2 Repository of Sofware Components (RSCs)

This repository is a directory with installations files of software components that are needed for creating and deploying the TRENCADIS services.

These software components have to be installed on top of the VMIs listed in table 1. To provide reliability and auto scaling of services a specific configuration is needed that depend on the Cloud used. Table 2 shows the files and their type. For example, Grid Archive Files (Gar) are needed to deploy Grid services in the Globus Toolkit 4 container.

4.3 Repository of Service Descriptions (RSDs)

This repository is a set of documents that describe the composition and configuration of the TRENCADIS services. Each document specifies the set of VMs involved in a service, and for each type of VM, its hardware requirements (number of CPUs, memory, etc.), software components, indicating the configuration required to set up the services (creating users, directories, deploy Grid services, create SQL databases, install Java APIs etc..). To write the documents, the Resource Application Description Language for Cloud environments (RADL) [22] has been used. RADL expresses in a simple and declarative way the requirements of the TRENCADIS services on a set of VMs, as well as to obtain information

Table 2. Installation Files for TRENCADIS Infrastructure

ID	Instalation File	Type of File
1	DICOM Storage Grid Service	Gar
2	Key Server Grid Service	Gar
3	EOUID Generator Grid Service	Gar
4	Storage Broker Grid Service	Gar
5	Ontologies Server Grid Service	Gar
6	TRENCADIS Java API Indexer	Jar
7	TRENCADIS Java API Data Backend	Jar
8	TRENCADIS Java API SQL Keys DB	Jar
9	TRENCADIS Java API SQL Ontologies DB	Jar
10	SQL Keys database	SQL
11	SQL Ontologies Database	SQL
12	Reliability Configuration	Conf
13	Auto scaling Configuration	Conf

from the VMs already instantiated. A RADL document consists of three sections: The first one (system) declares the requirements of different types of VMs required. The second one (configuration) describes the configuration steps required for each type of VMs. The last one (deploy) indicates the number of instances of each one.

Table 3. Created RADLs for TRENCADIS Infrastructure

ID	RADL	ID VMI	ID Installation File
1	DICOM Storage Service	[1,3 or 4 or 5,6 or 7]	[1,6,7]
2	Key Server Service	[1,3]	[2,8,10]
3	EOUID Generator Service	1	[3]
4	Storage Broker Service	1	[4]
5	Ontologies Server Service	[1,3]	[5,9]
6	VOMS Service	2	–
7	Information Server Service	1	–
8	TRENCADIS Center	[1,2,3]	[3,4,5,9]

Table 3 lists RADL documents created in this work, one for each TRENCADIS service. A RADL document has also has been created to describe all services grouped by TRENCADIS center.

As an example, figure 3 shows the RADL document needed to deploy the Key Server service where two VMs are needed (GT4-JAVA-ANT and POSTGRESQL). The first one needs a Scientific Linux (SL) 5.7 VM with Globus Toolkit 4, java and ant installed. The second one also needs a SL 5.7 VM but only with PostgreSQL installed. Also two network interfaces are specified: a pri-

vate network to connect the instances and a public network interface that is used by the Grid service for interacting with other TRENCADIS services.

```
network private
network public (outbound = 'yes')

system GT4_JAVA_ANT (
   cpu.arch='x86_64' and cpu.count>=1 and memory.size>=1024m and
   net_interfaces.count = 2 and net_interface.0.connection = 'public' and
   net_interface.1.connection = 'private' and
   disk.0.os.name='linux' and disk.0.os.flavour='Scientific Linux' and
   disk.0.os.version='5.7' and
   disk.0.application contains (name='globus', version='4') and
   disk.0.application contains (name='java', version='1.6.0.35') and
   disk.0.application contains (name='ant', version='1.8.2')
)
system BACKEND_POSTGRESQL (
   cpu.arch='x86_64' and cpu.count>=1 and memory.size>=1024m and
   net_interfaces.count = 1 and net_interface.0.connection='private' and
   disk.0.os.name='linux' and disk.0.os.flavour='Scientific Linux' and
   disk.0.os.version='5.7' and
   disk.0.application contains (name='PostgreSQL', version>='8.4.9')
)

configure GT4-JAVA-ANT (
     add_user {'trencadis': }
     deploy_gs {'Key_Server_Grid_Service.gar': }
     install_conf_API_SQL_Keys_Database
                            {'TRENCADIS_Java_API_SQL_Keys_DB.jar': }
)

configure BACKEND_POSTGRESQL(
     add_user {'trencadis': }
     create_database {'SQL_Keys':
                         file =>'database.SQL'}
)

deploy GT4_JAVA_ANT 1
deploy POSTGRESQL 1
```

Fig. 3. RADL document for Key Server service.

4.4 Infraestructure Instantiator (II)

This component is in charge of interpreting the infrastructure descriptions specified in the RADL documents, to finally perform the effective deployment of the instances of the selected VMI in a cloud environment or in a virtualization system. It is implemented by means of the Infrastructure Manager (IM) [22]. The IM is a service that provides a high level REST API and Web Interface to enable the deployment and automatic contextualization of Cloud infrastructures. It provides a set of functions to create and destroy virtual infrastructures and also to provision and relinquish computational resources in an elastic manner. The IM enable to connect with different Cloud systems such as OpenNebula [24], OpenStack [25] and Amazon EC2. It also enables to connect with virtualization systems like KVM [26] using the LibVirt interface [23]. The functional scheme is the following: The first step is to connect to the VMRC to select the most suitable image(s) with respect to the user requirements. Then it selects the user

credentials to access the cloud deployments or virtualization systems to launch the VM instances. To launch them, it must translate the RADL requisites into instances of the selected system. Finally it waits the VMs to be running in order to perform the contextualization process using Puppet.

For example, for deploying the Key Server service, the IM connects to the VMRC to select the VMI 1 and 3. Then, it submits the VM instances to a cloud or virtualized infrastructure. Finally, it waits for the VMs to be running in order to perform the contextualization using the software components with ID 2, 8 and 10.

4.5 Virtual Machine Manager (VMM)

This component is in charge of managing the instances of VMs launched by the IM. It must provide functionality to create and destroy infrastructures and also data persistence when undeploying VMs. This component can be implemented in the platform using software such as OpenStack or OpenNebula or public clouds such as Amazon EC2.

5 Test Deployment

To test the platform designed in this work, a complete TRENCADIS infrastructure has been set up following the deployment model presented in section 3. This infrastructure is shown in figure 4 deployment has emulated all the services required for three medical centers and a TRENCADIS center.

For managing the instances of VMIs required to deploy the TRENCADIS services, a public cloud, a private cloud and a virtualization platform have been combined. Figure 4 shows the VMs involved in the deployment and the VMM used in each one.

In particular, the private cloud used has been supported by four Dell Servers in Blade format (M600 and M610 models). Each server has eight cores and 16 Gb of RAM and are mounted on a M1000e chassis. Three nodes were available to run VMs submitted by OpenNebula (version 3.4.1) while one node could run VMs submitted through OpenStack (Essex release). The public cloud used has been Amazon EC2 and the virtualization platform has been supported by the KVM hypervisor. The services Ontologies Grid service and EOUID Generator service have been deployed using the reliability configuration file, setting two instances in Amazon EC2 and the corresponding load balancer for proper workload distribution. The service Storage Broker Grid service has been deployed using the auto scalable configuration file, setting an autoscaling group of Amazon EC2, where a new storage a new Storage Broker is deployed if the CPU Utilization is above 80% for more than 1 minute. The Amazon EC2 API supports auto scaling and allows creating groups of instances, maintaining a minimum and maximum of instances actives, automating the creation of new replicas to replace failed ones [27].

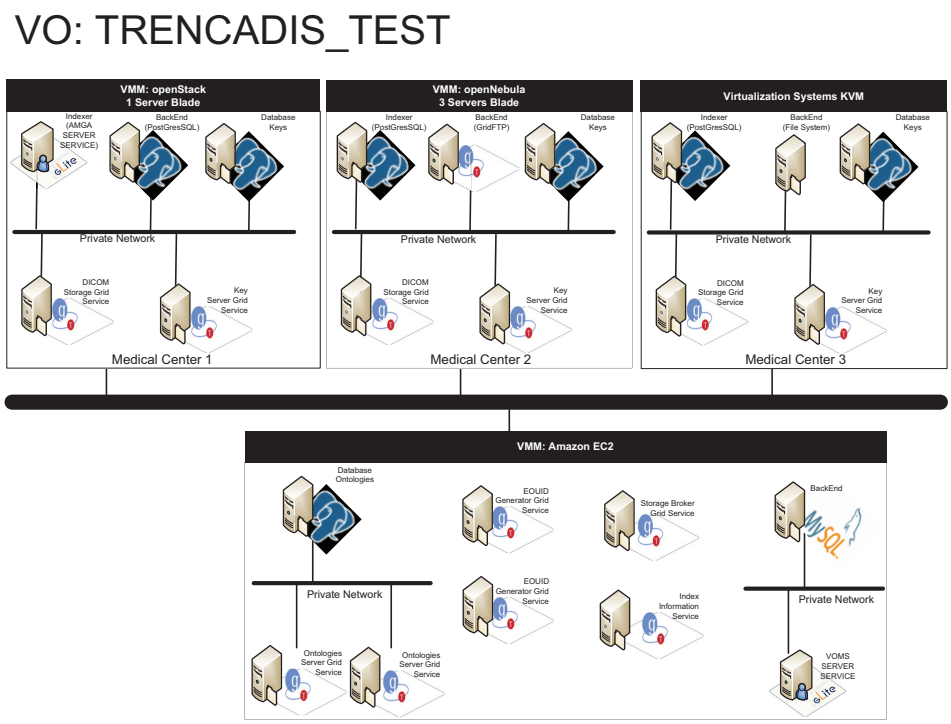


Fig. 4. Test Deployment of a TRENCADIS infraestructure

6 Conclusions and Future Work

The platform designed for deployment TRENCADIS infrastructures enable to combine virtualization technologies and clouds (public and private), depending on the needs of the deployment.

The use of these technologies significantly simplifies the deployment, while improving its performance and reliability, as these technologies enable deploying new services provisioning on demand, in an elastic and dynamic way.

Furthermore, the platform allows you to create, deploy and configure on-demand services, combining different versions according to the required needs, efficiently and without losing its scalability.

In this work, we have defined a methodology that has allowed to identify software components required to deploy an infrastructure TRENCADIS, analyzing all its services. Based on the identified components, these have been implemented in the platform as VMIs or configuration files. This methodology can be applied to other infrastructures and the platform can be used in other areas different from TRENCADIS. Therefore, as a future work we plan to identify and apply the methodology in use cases different from TRENCADIS.

Acknowledgements

The authors wish to thank the financial support received from The Vicerectorat d'Investigació de la Universitat Politècnica de València (UPV) to develop the project "Diseño de Componentes Cloud Facilitadores del Despliegue y la Alta Disponibilidad de Servicios TRENCADIS, para compartir Imágenes Médicas DICOM e informes Asociados DICOM-SR", with reference 20111013.

References

1. C. Vázquez, E. Huedo, R.S. Montero, I.M. Llorente, *"On the use of clouds for grid resource provisioning"*, Future Generation Computer Systems, 27(5): 600-605, 2011
2. A.M. Kuo, *"Opportunities and Challenges of Cloud Computing to Improve Health Care Services"*, J Med Internet Res. 13(3): e67, 2011
3. European Commission: *"Safeguarding privacy in a connected world a European data protection framework for the 21st century"*. Tech. rep., European Commission, Brussels, Belgium (2012)
4. J. Kommeri, M. Niinimki, H. Mller, *"Safe storage and multi-modal search for medical images"*, Stud Health Technol Inform, 169:450-454, 2011
5. I. Blanquer, V. Hernńdez, F.J. Meseguer, J.D. Segrelles,*"Content-based organisation of virtual repositories of DICOM objects"*, Future Gener. Comput. Syst., 25(6):627637 (2009)
6. Medical Imaging & Technology Alliance, *"DICOM - Digital Imaging and Communications in Medicine"*, http://medical.nema.org
7. D. Segrelles, I. Blanquer, J. Salavert, V. Hernandez, J. Franco, G. Diaz, R. Ramos, R. Medina, L. Marti, M. Guevara, N. Gonźlez, J. Loureiro, I. Ramos, *"Exchanging Data for Breast Cancer Diagnosis on Heterogeneous Grid Platforms"*, Computing and Informatics. 31(1): 3-15, 2012
8. B. Narasimhan, R. Nichols, *"State of Cloud Applications and Platforms: The Cloud Adopters' View"*, Computer, 44(3): 24-28, 2011
9. *"Puppet:"* http://puppetlabs.com . visited 16/04/2013
10. *Chef*: http://www.opscode.com/chef/. visited 16/04/2013
11. *Amazon Inc. Amazon Elastic Compute Cloud (Amazon EC2).* http://aws.amazon.com/ec2. visited 30/10/2012
12. *Apache Whirr*: http://whirr.apache.org/. visited 16/04/2013
13. *CloudFoundry Open Source*: http://www.cloudfoundry.org/. visited 16/04/2013
14. N. Chohan, C. Bunch, S. Pang, C. Krintz, N. Mostafa, S. Soman, and R. Wolski. *AppScale: Scalable and Open AppEngine Application Development and Deployment.* In International Conference on Cloud Computing, Oct. 2009
15. *Google App Engine*: http://cloud.google.com/appengine. visited 30/10/2012
16. B. Koblitz, N. Santos, V.Pose. *"The AMGA Metadata Service"*. JGC. ISSN: 1570-7873 (Print) 1572-9814 (Online). Volume 6, Number 1 / March de 2008. Pages 61-76. Springer Netherlands, 2007.
17. I. Blanquer, V. Hernandez, J. Salavert, D. Segrelles. *"Integrating TRENCADIS Components in gLite to Share DICOM Medical Images and Structured Reports"*. HealthGrid Conference, 2010, pp 64-75 .ISBN 978-1-60750-582-2.
18. *"The Worlds Leading Graph Database"*. http://neo4j.org . Visited 16/04/2013.

19. *"Cloud Data Management Interface - Specification Version 1.0.2h"*, SINA, 2010. http://cdmi.sniacloud.com . visited 26/10/2012
20. *"gLite-Lightweight Middleware for Grid Computing"*. http://glite.cern.ch . visited on 16/04/2013
21. Carrion, Jose V., Germn Molto, Carlos De Alfonso, Miguel Caballer, and Vicente Hernandez. 2010. *"A Generic Catalog and Repository Service for Virtual Machine Images."* In 2nd International ICST Conference on Cloud Computing (CloudComp 2010).
22. C. de Alfonso, M. Caballer, F. Alvarruiz, G. Molto, V. Hernandez, Infrastructure Deployment Over the Cloud, in: 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2011), 2011, pp. 517-521.
23. Bolte M, Sievers M, Birkenheuer G, Niehrster O, Brinkmann A. *"Non-intrusive virtualization management using libvirt"*. Proceedings of the 2010 Conference on Design, Automation and Test in Europe (DATE), Dresden, Germany, 2010.
24. J. Fontan, et al. *"OpenNebula: The Open Source Virtual Machine Manager for Cluster Computing"*. In Open Source Grid and Cluster Software Conference, May 2008.
25. *OpenStack Open Source Cloud Computing Software*. http://www.openstack.org, visited 16/04/2013
26. Kivity, A., Kamay, Y., Laor, D., Lublin, U., and Liguori, A. KVM: the linux virtual machine monitor. In Ottawa Linux Symposium (July 2007), pp. 225230.
27. http://aws.amazon.com/developertools/2535 . visited on 16/04/2013

Analysis of Scientific Cloud Computing requirements

Álvaro López García**, Enol Fernández del Castillo

Advanced Computing and e-Science Group
Instituto de Física de Cantabria, CSIC - UC, Spain
aloga@ifca.unican.es,

Abstract. While the requirements of enterprise and web applications have driven the development of Cloud computing, some of its key features, such as customized environments and rapid elasticity, could also benefit scientific applications. However, neither virtualization techniques nor Cloud-like access to resources is common in scientific computing centers due to the negative perception of the impact that virtualization techniques introduce.
In this paper we discuss the feasibility of the IaaS cloud model to satisfy some of the computational science requirements and the main drawbacks that need to be addressed by cloud resource providers so that the maximum benefit can be obtained from a given cloud infrastructure.

1 Introduction

Nowadays Cloud computing has achieved great success in the enterprise world but it is still not common in the scientific computing field. Virtualization —that is a key component on the Cloud computing—, and its associated performance degradation, has traditionally been considered as not compatible with the computational science requirements. However, nowadays it is accepted that virtualization introduces a low CPU overhead [1, 2] and that the penalty introduced in I/O can be significantly reduced with techniques such as SR-IOV [3] and PCI-Passthrough [4] that provide near native performance [5] using modern hardware with specialized support for virtualization.

Moreover, virtualization also brings some benefits that overcome its performance drawback, namely isolation and encapsulation. The isolation of VMs prevents influences from misbehaving VMs to impact on other running VMs, while encapsulation of VMs gives the means to provide load balancing and high-availability techniques. Virtualization also enables the consolidation of services by providing support for a wider range of services with the same physical hardware, that leads to a more efficient usage of the infrastructure and a reduction of maintenance costs.

This work complements some previous studies, such as Blanquer et al. [6], Ramakrishnan et al. [7] and Juve et al. [8]. This paper is focused in the set of

** e-mail of corresponding author: aloga@ifca.unican.es

requirements —for a resource provider and the cloud middlware— that a IaaS Cloud should provide for scientific usage, therefore there are some higher level aspects (namely programming models, job-oriented execution models, etc.) that are not covered by this paper. Also, it is worth noting that we are not focusing on higher level Cloud service models (such as PaaS or SaaS).

In Section 2 we give an outlook of the main benefits of using a Cloud Computing model for scientific research. In Section 3 a set of pilot use cases is described. From this preliminary group of applications, in Section 4 we have identified and established some requirements for a Scientific Cloud Infrastructure.

2 Cloud Computing benefits for scientific applications

Cloud Computing can be defined as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." [9]. This model allows many enterprise applications to scale and adapt to the usage peaks without big investments in hardware with a *pay-as-you-go* model.

On the other hand, Scientific Computing can be defined as the efficient usage of computer processing in order to solve scientific problems. It can be considered as the "intersection of numeral mathematics, computer science and modelling" [10] and spans a broad spectrum of applications and systems, such as High Performance Computing (HPC), High Throughput Computing (HTC), Grid infrastructures, small and mid-sized computing clusters, volunteer computing and even local desktops.

Many of the features of the Cloud Computing model are already present in current scientific computing environments: academic researchers have used shared clusters and supercomputers since long, and they are being accounted for their usage in the same pay-per-use basis —i.e. without a fixed fee— based on their CPU-time and storage consumption. Moreover, Grid computing makes possible the seamless access to worldwide-distributed computing infrastructures composed by heterogeneous resources, spread across different sites and administrative domains. However, the Cloud computing model fills some gaps that are impossible or difficult to satisfy and address with any the current computing models in place at scientific datacenters. In the following sections we describe the major benefits that the cloud computing model can bring to a scientific computing infrastructure.

2.1 Customized environments

One of the biggest differences between the Cloud model and any of the other scientific computing models (HPC, HTC and Grids) is the execution environment flexibility. While in the later ones the execution environment is completely fixed by the infrastructure and/or resource providers (e.g. the European Grid

Infrastructure[1], one of the majors grid infrastructures with 300+ resource centers providing 320, 000+ cores, supports only 3 Operating System flavors), in the Cloud model the execution environments are easily adaptable or even provided by the final users. This makes possible the deployment of completely customized environments that perfectly fit the requirements of the final scientist's applications.

This lack of flexibility in the current computing infrastructures —where a specific (or a very limited group) operating system flavor with a specific set of software and libraries is deployed across all the available computing nodes— forces most applications to go through a preparatory phase before being executed to adapt them to the execution environment idiosyncrasies, such as library and compiler versions. Moreover, some scientific applications use legacy libraries that are not compatible with the available environments, rendering this preparation step quite time-consuming or even impossible in some cases. The users could get rid of this procedure to an extent if they were able to provide its own computing environment, that will be the one used for its computations.

The requirement of a fixed operating system and the absence of customization has been identified [11] as one of the main show-stoppers for many scientific communities to adopt Grid computing technologies. Only large communities are able to tackle this issue, thanks to dedicated manpower to manage and adapt their software development and deployment to the available scenarios.

Providing custom execution environments independently of the underlying physical infrastructure also allows long-term preservation of the application environment and opens the possibility of running legacy software with current and future hardware, which may help in the long-term preservation of data (and analysis methods for those data) of scientific experiments.

2.2 On-demand access with rapid elasticity

The Cloud model is based on on-demand and pay-as-you-go access that gives the illusion of infinite resource capacity that can rapidly adapt to the needs of the user. Although providing an infinite resource capacity is not feasible in scientific datacenters, on-demand access to resources is useful for interactive tasks.

Resources in the cloud model are elastically provisioned and released, opening the door to using disposable environments without the overhead of a physical deployment would imply (hardware preparation, re-installation, configuration). These kind of disposable environments can be used for large-scale scalability tests of parallel applications, or for testing new code or library versions without disrupting production services already in place.

2.3 Non-conventional application models

Most scientific computing resources (supercomputers, shared clusters and grids) are focused on processing and execution of atomic tasks, where each of these

[1] See http://www.egi.eu for details

tasks may be parallel or sequential and they may have interdependencies between them or be executed concurrently. All the tasks have a common life-cycle: they are started, they process some data and eventually return a result.

However, in an Infrastructure as a Service (IaaS) Cloud, this traditional task concept does not exist: instead of tasks, users manage instances of virtual machines, which are started, stopped, paused and terminated according to the needs of the users. This different life-cycle makes possible to create creation of complex and dynamical long-running systems. For example this feature is used in the simulation of dynamic software agents, as in [12, 13]; the decision making process in urban management [14] or behavioral simulations using shared-nothing Map-reduce techniques [15].

3 Application use cases

In this section we present some preliminary use cases deployed in our private Cloud testbed. Although the applications are executed successfully in the current infrastructure, we have identified some drawbacks that should be addressed so that the scientific users could get even a better experience. These topics will be further discussed and described on Section 4.

3.1 PROOF

The Parallel Root Facility (PROOF) [16] is a commonly used tool by the High Energy Physics (HEP) community to perform interactive analysis of large datasets produced by the current HEP experiments. PROOF performs a parallel execution of the analysis code by distributing the work load (input data to process) to a set of execution hosts in a single program, multiple data (SPMD) fashion.

PROOF is used in the last phases of the physics analysis to produce the final plots and numbers, where the possibility of interactively change the analysis parameters to steer the intermediate results facilitates the researchers work and allows them to reach faster to better results. Data analyzed in this phase contains the relevant physics objects in set of files —produced by several previous processing and filtering steps of the original raw data collected from the detector— that may range from several GBytes to a few TBytes.

These analysis tasks are usually I/O bounded [17] due to the big volume of data to process and their relatively low CPU requirements: most codes perform filtering of the data according to the relevant physics to be measured.

Running PROOF requires the pre-deployment and configuration of a master, that acts as entry point and distributes the workload, and a set of workers where the user's analysis code is executed. There are tools that automatize the creation of such deployments, which is not trivial for most users, in batch-system environments [17, 18], but the machines are shared with other jobs, which may cause a degradation of the performance.

A IaaS Cloud testbed provides support to these kind of interactive analysis (i.e short lived sessions initiated on-demand by the users and with high

performance access to data) with customized environments where the PROOF daemons run isolated from other workloads and are disposed as the analysis finishes.

3.2 Particle Physics Phenomenology

As many other communities, the particle physics phenomenology groups develop their own software for producing their scientific results. Software packages developed by the community have evolved independently for several years, each of them with particular compiler and library dependencies. These software packages are usually combined into complex workflows, where each step requires input from previous codes execution, thus the installation and configuration of several software packages are mandatory to produce the scientific results. Moreover, each scientific scenario to be analyzed may require different versions of the software packages, therefore the researchers need to take into account the different package versions characteristics for installing and using them. Some of these packages also require access to proprietary software (e.g. Mathematica) that is license-restricted. Although institutional licenses may be available, these are difficult to control in shared resources (like grids or clusters) due to the lack of fine grained access control to resources.

Setting up a proper computing environment becomes a overhead for the everyday work of researchers: they must solve the potential conflicts that appear when installing them on the same machine; and the fixed execution supported by the resource providers forces them to deploy the tools in ad-hoc clusters or even their own desktops.

A cloud computing testbed allows these researchers to deploy a stable infrastructure built with the exact requirements for their analysis where each machine is adapted to the different scientific scenarios to be evaluated, i.e. with the specific software versions needed for the analysis. The cloud infrastructure should be able to enforce any usage or license restrictions for proprietary software.

The possibility of creating snapshots of the machines also allows the recovery of previous experiments easily without recreating the whole software setup. These users would benefit from contextualization tools that automatically sets up and handles any dependencies of the software packages needed for the analysis upon machine creation [19].

3.3 Pattern Recognition from GIS

The Vegetation Indexes (NDVI[2] and EVI[3]) estimate the quantity, quality and development of the vegetation in a given area [20] by means of remote sensor data, such as satellite images. Using pattern recognition techniques it is possible to analyze the behavior of this index, so as to make a non-supervised vegetation

[2] Normalized Difference Vegetation Index
[3] Enhanced Vegetation Index

classification. The analysis of such data also opens the door to other applications such as fire detection, deforestation and vegetation regeneration. For these data to be analyzed several specialized tools need to be deployed, such as Modis (satellite data analysis tools), GRASS and GDAL (geospatial libraries and tools), PROJ4 (cartographic data management) and R statistical programming environment (along with a large set of additional R modules for interacting with the other pieces of software).

These analysis were carried out in advance in the Grid so all the required software had to be installed beforehand. This required from the intervention of a local support team, so as to ensure the correct deployment of the tools and applications. During this process, incompatibilities were found between the dependencies of the required software and the operating system libraries installed. This process delayed the start-up of the actual data process several weeks. Moreover, the users faced a new computing environment and had to be instructed on how to interact with the installed software in order to use the correct versions that had to be installed in non-standard locations. Finally, the data produced were stored in an external database —that had also to be deployed— so that they could be finally accessed and analyzed by the scientists.

This use-case could profit from the Cloud computing testbed in two ways. Firstly, they could deploy a ready to compute self-contained image, bundling all the required software into it; and secondly, they could deploy their own infrastructure to store and retrieve their data. By doing so, they would reduce the time needed to start with the analysis (as the software is ready to be executed), the usage entry barrier (as they are deploying its own environment and they are familiar with it) and leverage the management of the external database service to the Cloud middleware (so they do not need to host a physical server for it).

4 Requirements for a Scientific Cloud Infrastructure

From our experience supporting the execution of the applications described on Section 3 we have gathered some requirements that a scientific cloud infrastructure should provide to its scientific users (however, this is not an exhaustive list and they are not formal requirements). We have classified these requirements in three groups: application level requirements, for requirements relevant for easing the usage of cloud resources; specialized hardware, for requirements related to high-performance access to specialized facilities; and enhanced scheduling policies, for those policies that the cloud provider should adopt to provide an adequate service for scientific users.

4.1 Application level requirements

The deployment of customized environments is one of the biggest advantages of the Cloud Computing model against any other *traditional* paradigms, but it may also represent a drawback for users that are not familiar with systems administration. In this context, scientific application catalogs and contextualization mechanisms are needed.

Scientific Application Catalogs The Cloud Computing flexibility to deploy customized virtual machines has associated the responsibility of create and manage them. Most scientific users are not prone to create, manage and maintain their own system images, nor have the skills or knowledge to perform those tasks in a secure and efficient way. These users may profit from a Cloud infrastructure where a predefined set of supported images is already deployed, containing a wide range of the software they need. This ready to use Scientific Application Catalog can lower the entry curve for this new infrastructure.

Another aspect of this application catalogs is the access to licensed and institutional software —that is, software specially designed and/or tuned to be executed and integrated within an institution. In this cases, only restricted access to the images will be provided to the users, so that only the allowed ones are able to run the requested software. For example, access to shared and clustering filesystems —such as IBM GPFS, Lustre, etc— can only be provided to machines that are trusted and properly configured. Offering these images in the catalog with restricted access, will give access to these resources easily.

Image contextualization The contextualization of images can be defined as the process of installing, configuring and preparing software upon boot time on a pre-defined virtual machine image. This way, the pre-defined images can be stored as generic and small as possible, since all the customizations will take place on boot time.

The image contextualization is tightly coupled with the Scientific Application Catalogs described in Section 4.1. The catalogs are useful for bundling self-contained and ready to use images, but sometimes this is something not feasible, because the required software evolves and changes frequently its version, because the software is under a development and debugging process and it is not practical to bundle it inside a self-contained image or simply because it needs some user-defined data so that it can be properly customized.

In those cases, instead of creating and uploading a new image for each application version and/or modification (a tedious process that is a time consuming task for the image creator), the installation and/or customization can be delayed until the machine boot time. By means of this mechanism the newest version can be automatically fetched and configured, or the defined and variable user-data provided to the image. This is done by means of ready to use and compatible image that contains all the necessary dependencies and requisites for the scientific applications to be installed. This contextualization-aware images will then be launched with some metadata associated, indicating which the software to install and configure.

Nowadays powerful configuration management tools exist that can help with the implementation of the described contextualization mechanisms. Tools such as *Puppet* [21], *CFengine* [22], *Chef* [23], etc. make possible to define a machine *profile* that will be then applied to a machine, so that an given machine will fit into that profile after applying it. However, these languages and tools introduce

a steep learning curve, so that the cloud middleware should provide a method to expose the defined profiles to the users easily.

4.2 Specialized hardware

Scientific Computing sometimes requires access to specialized hardware that is not often present at a commercial provider, that is not focused towards scientific computing.

High performance communications Most parallel applications need low-latency, high speed network interconnects (such as Infiniband or 10GbE) in order to be efficiently executed. These interconnects are common in HPC environments, but they are not so common in cloud providers. Moreover, this hardware normally does not have the support for being virtualized or shared between several virtual machines. In order to give access to these devices two solutions exist: PCI passtrough with IOMMU or Single Root I/O Virtualization (SR-IOV).

High performance data access It is common that data oriented workloads demand high speed access towards the data to be analyzed. In a cloud framework, the data is normally decoupled from the instance that is running, meaning that it is being stored elsewhere not known a priory by the user. For example, block devices can be attached from a central storage location over the network (by means of Ata over Ethernet or iSCSI) to a running instance. If access to the data is not efficient enough, the computation will be executed on the node will suffer from a performance penalty that will make it unusable.

4.3 Enhanced scheduling policies

Scientific applications need of enhanced scheduling policies that take into account not only the requested and available resources, but also the kind of execution that is going to be done and any special requirement that the scientific user may have.

Instance co-allocation Some workloads require of the parallel execution of tasks across several nodes. In this context, large requests have to be discriminated between non-dependent and tightly-coupled or parallel nodes. Although the former can be provided in a first-come, first-server basis; the later ones need of some advances scheduling features, so that the collocation of instances makes possible that the user's tasks can be properly orchestrated. This way, not only the resources should be reserved in advance, but also the overheads and delays introduced by the cloud management software 4.3 should be taken into account so that the instances have the same boot time.

Short startup overhead When a request is made, the virtual images have to be distributed from the catalog to the compute nodes that will host the virtual machines. If the catalog repository is not shared or the image is not already cached by the compute nodes, this distribution will introduce a penalty on the start time of the requested nodes. This distribution penalty can be quite significant in large systems, or when bigger requests are made by a user.

Large requests are common in scientific workbenches, so a mechanism should be provided to ensure that these request are not penalized by this transfer time. Some possible solutions could be to share the image catalog, to pre-schedule the image transfers in advance or to utilize some efficient and intelligent distribution methods for the requested images instead of downloading them from a central location.

Performance aware placement A virtual machine can potentially share the same physical host with other machines. This can introduce a performance degradation if the machines are competing for the utilization of the system resources. For example, two machines that are executing some I/O consuming task can interfere between them. This issue has to be handled by the scheduler so that two competing machines are not scheduled in the same node.

On the other hand, as we already explained in Section 4.2, the requested nodes may need access to specialized hardware such as low-latency interconnects (for example Infiniband), GPGPUS, etc. In these cases, not only the scheduler has to be aware of these available resources, but also the cloud middleware should be able to manage them. These resources are normally attached to the virtual machines without being virtualized (that is, attaching the PCI device directly to the node) so they deserve a different treatment.

Spot and preemptable instances Long-running tasks are common in computational science. Those kind of workloads do not require from interactivity and normally are not time-bounded. Such tasks can be used to fill the computing infrastructure usage gaps, thus a better utilization of the resources will be obtained. This is normally done in traditional scientific infrastructures by means of several techniques, such as backfilling, priority adjustments, task preemption and chekpointing.

However, a virtual machine can be transparently paused into a safe state that can be resumed later on. This allows to create new execution types at a lower cost for the users, such as the so called by some commercial providers *spot-instances*: machines that will run whenever there is enough room for them, but that can be preempted, paused or even destroyed by higher priority tasks. This is an interesting topic for the scheduling field of Computer Science: the usage of reverse-auction [24] and other economical models [25] opens the door to a better utilization of the resources, by making attractive for some users to utilize the infrastructure in the usage-valley periods whenever they can afford to pay the price for those resources.

Bare metal provisioning There may be some situations where the user needs to run a native operating system instead of a virtual system. Some use-cases for the bare-metal provisioning are the deployment of machines that need to access to a given hardware that cannot be virtualized and/or directly attached to the virtual machine, non-x86 architectures, databases, etc.

4.4 Absence of vendor lock-in

Interoperability is an important feature for many communities. The usage of open standards such as the Open Cloud Computing Interface (OCCI) [26], Cloud Infrastructure Management Interface (CIMI) [27] and Cloud Data Management Interface (CDMI) [28] is way to avoid the vendor lock-in that currently exists with many commercial cloud vendors.

At a lower level it is also important also the adoption of the standards so as to avoid the "hypervisor lock-in". In this context, the adoption of the Open Virtualization Format (OVF) [29] should be considered for the distribution of virtual appliances.

5 Conclusions

Cloud computing has permeated into the IT industry in the last few years, and it is nowadays emerging in scientific computing environments. However, there are still some gaps that need to be filled so that the computational science could completely benefit from it. In this paper we have made a sort review of the advantages that a cloud computing model can offer to scientific users and how either the middlewares and the resource providers need to adapt to satisfy their new potential users.

Cloud middlewares are normally arising from the commercial providers — being Open Source or not— and they are normally focused to satisfy their needs, that are not the same as the scientific users requirements. From our experience, one of the main fields needing from improvement in these middlewares is the scheduling, as described in Section 4.3. There is also room for improvement in additional and higher level applications, such as image catalogs and machine contextualization systems as described in Section 4.1.

Scientific computing datacenters have to move towards a mixed and combined model, where a given user can still access their traditional computational power —that is, using a batch system or using the Grid—, but also they should provide their users with some additional cloud power that will complement the former computing models. This way, either the users the users will benefit from a richer environment, and resource providers can get a better utilization of their resources, since they will allow for new execution models that are currently not available.

References

1. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of the nineteenth ACM symposium on Operating systems principles SE - SOSP '03*, (New York, NY, USA), pp. 164–177, ACM, 2003.
2. A. Ranadive, M. Kesavan, A. Gavrilovska, and K. Schwan, "Performance implications of virtualizing multicore cluster machines," in *HPCVirt '08: Proceedings of the 2nd workshop on System-level virtualization for high performance computing*, (New York, NY, USA), pp. 1–8, ACM, 2008.
3. P. SIG, "PCI-SIG Single Root I/O Virtualization and Sharing Specification," tech. rep., 2010.
4. J. Liu, W. Huang, B. Abali, and D. Panda, "High performance VMM-bypass I/O in virtual machines," in *Proceedings of the Annual Conference on USENIX*, vol. 6, 2006.
5. Y. Dong, X. Yang, J. Li, G. Liao, K. Tian, and H. Guan, "High performance network virtualization with SR-IOV," *Journal of Parallel and Distributed Computing*, vol. 72, pp. 1471–1480, Nov. 2012.
6. I. Blanquer, G. Brasche, and D. Lezzi, "Requirements of Scientific Applications in Cloud Offerings,"
7. L. Ramakrishnan and P. Zbiegel, "Magellan: experiences from a science cloud," *... on Scientific cloud ...*, 2011.
8. G. Juve and E. Deelman, "Scientific workflows and clouds," *Crossroads*, vol. 16, pp. 14–18, Mar. 2010.
9. P. Mell and T. Grance, "The NIST definition of cloud computing," Tech. Rep. Special Publication 800-145, National Institute of Standards and Technology (NIST), 2009.
10. G. Karniadakis and I. Kirby, *Parallel Scientific Computing in C++ and MPI*. Cambridge University Press, 2003.
11. C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, "On the Use of Cloud Computing for Scientific Workflows," *2008 IEEE Fourth International Conference on eScience*, pp. 640–645, Dec. 2008.
12. P. Sethia and K. Karlapalem, "A multi-agent simulation framework on small Hadoop cluster," *Engineering Applications of Artificial Intelligence*, vol. 24, pp. 1120–1127, Oct. 2011.
13. D. Talia, "Cloud Computing and Software Agents: Towards Cloud Intelligent Services," *ceur-ws.org*.
14. Z. Khan, D. Ludlow, R. McClatchey, and A. Anjum, "An architecture for integrated intelligence in urban management using cloud computing," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 1, no. 1, p. 1, 2012.
15. G. Wang, M. Salles, B. Sowell, and X. Wang, "Behavioral simulations in mapreduce," *Proceedings of the ...*, no. 4, 2010.
16. I. Antcheva, M. Ballintijn, B. Bellenot, M. Biskup, R. Brun, N. Buncic, P. Canal, D. Casadei, O. Couet, V. Fine, L. Franco, G. Ganis, A. Gheata, D. G. Maline, M. Goto, J. Iwaszkiewicz, A. Kreshuk, D. M. Segura, R. Maunder, L. Moneta, A. Naumann, E. Offermann, V. Onuchin, S. Panacek, F. Rademakers, P. Russo, and M. Tadel, "ROOT – a C++ framework for petabyte data storage, statistical analysis and visualization," *Computer Physics Communications*, vol. 180, no. 12, pp. 2499 – 2512, 2009.

17. A. Y. Rodríguez-Marrero, I. González Caballero, A. Cuesta Noriega, E. Fernández-del Castillo, A. López García, J. Marco de Lucas, and F. Matorras Weinig, "Integrating PROOF Analysis in Cloud and Batch Clusters," *Journal of Physics: Conference Series*, vol. 396, p. 032091, Dec. 2012.
18. P. Malzacher and A. Manafov, "Proof on demand," *Journal of Physics: Conference Series*, vol. 219, no. 7, p. 072009, 2010.
19. I. Campos, E. Fernández-del-Castillo, S. Heinemeyer, A. López, and F. v. d. Pahlen, "Phenomenology tools on cloud infrastructures using openstack," *CoRR*, vol. abs/1212.4784, 2012.
20. J. Weier and D. Herring, "Measuring vegetation (NDVI & EVI)." `http://earthobservatory.nasa.gov/Features/MeasuringVegetation/`, 2012. Accessed: 08/02/2012.
21. "Puppet." `https://puppetlabs.com/`, 2012.
22. "Cfengine." `http://www.cfengine.com`, 2012.
23. "Chef." `http://www.opscode.com/chef/`, 2012.
24. J. Roovers, K. Vanmechelen, and J. Broeckhove, "A reverse auction market for cloud resources," *Economics of Grids, Clouds, ...*, 2012.
25. T. Pueschel, D. Neumann, and T. Püschel, "Management of Cloud Infastructures: Policy-Based Revenue Optimization," in *ICIS 2009 Proceedings*, 2009.
26. "Open Cloud Computing Interface (OCCI)." `http://occi-wg.org/`, 2012.
27. "Cloud Infrastructure Management Interface (CIMI)." `http://dmtf.org/standards/cloud`, 2012.
28. "Cloud Data Management Interface (CDMI)." `http://www.snia.org/cloud`, 2012.
29. "Open Virtualization Format (OVF)." `http://dmtf.org/standards/ovf`, 2012.

Easing the Structural Analysis Migration to the Cloud

Miguel Caballer, Pedro de la Fuente, Pau Lozano, José M. Alonso, Carlos de Alfonso**[1], Vicente Hernández

Instituto de Instrumentación para Imagen Molecular (I3M). Centro mixto CSIC - Universitat Politècnica de València - CIEMAT, camino de Vera s/n, 46022 Valencia, España.

Abstract. This paper describes the migration of a scientific application, related to the structural analysis of buildings and civil engineering structures, to the Cloud. For that, two different approaches have been carried out: one of them based on the Generic Worker, a web-role implementation that manages the execution of the remote tasks in a Windows Azure-based Cloud infrastructure, and the other one based on CodeCloud, a software implementation that notably eases the application deployment in public or private Clouds. The architecture of the Generic Worker-based approach is shown, together with the different components of the CodeCloud software and an example of the CJDL file needed to launch the structural analysis application in the Cloud. A real building has been used as a case study to compare the performance, in terms of execution time, between the Generic Worker and the CodeCloud-based deployments.

1 Introduction

In the last years, many research groups started to migrate their applications from their local computers to large scientific infrastructures, such as Grids or supercomputers. The promise of access to large computational resources enabled scientists to increase the resolution of the simulations, the amount of data, etc. leading to enhance the results. But the access to those large computational infrastructures is restricted if the applications are not involved in research projects. The scientists who needed a Grid infrastructure must ask for resources to infrastructure projects such as EGEE, EELA or the new NGIs. The same situation occurs when they want to access to supercomputers. Their requests must be evaluated, and although they are usually granted with computational resources, it is not guaranteed.

In the last years of Grid, Cloud Computing appeared to offer on-demand computational resources on a pay-per-use basis to users. This model is very interesting for scientists as it enables them to be self-sufficient to use computational resources. The bureaucracy to obtain the resources (i.e. application for

** e-mail of corresponding author: caralla@upv.es

resources, waiting for calls or evaluations, etc.) is avoided and the time to get the results is reduced, because the resources are almost immediately delivered as they are paid. In addition, it is still cheaper than owning a cluster, because scientists usually want to use it for a limited number of executions [1]. The scientists are less constrained about the hardware of the virtual machine (VM) that could be used, the applications or libraries installed on it, the version of the operating system, etc. It is possible to get a tailored VM for the application to be run. However, the scientists have just spent money and time to migrate their applications to the Grid or a cluster. The transition to a new technology such as the Cloud would involve learning about it and a new investment of time before getting the results.

The aim of the CodeCloud project is to ease the deployment of applications in Cloud environments, so that the effort required to successfully adapt applications to Cloud platforms can be drastically reduced. This would lead to increase the number of scientific applications that could benefit from Cloud infrastructures, and reduce the time for the production cycle. CodeCloud provides high level services for taking advantage of Cloud platform features from the application developer point of view . It also includes automated mechanisms for building the VMs, considering the whole process from the identification of the most suitable VMs from which to start, to its contextualization for the user application.

One use case of a scientific application is the dynamic analysis of buildings. The realistic 3D dynamic analysis of large dimension structures along the time gave place to a resource-consuming application that was successfully ported to Grid infrastructures [2]. The promising results and the opportunity of using a commercial application that takes advantage of these simulations [3] suggested the scientists to port it to the Cloud [4] under the framework of the VENUS-C project [5]. The solution was based on the Generic Worker component [6] to run the application in a Windows Azure-based Cloud infrastructure.

In this paper we describe the porting of the dynamic analysis of buildings to the Cloud, using the tools of the CodeCloud project. The differences and similarities between this approach and the steps needed to run the application are explained, trying to highlight which tools achieve better results or ease the deployment of the application in the Cloud.

The remainder of the paper is structured as follows. First, section 2 describes how the dynamic analysis of buildings is executed in the Cloud, using the Generic Worker approach. Then, section 3 briefly explains the tools available in the CodeCloud project and how they work. Later, section 4 details the work needed to run the original dynamic analysis of buildings application in the Cloud, using the CodeCloud approach. The section 5 makes a comparison between the two approaches shown in this paper. Finally section 6 presents some conclusions about the results and describes the work to do in a short future.

2 Structural analysis in the Cloud using the Generic Worker framework

Structural analysis of civil engineering structures determines their response to different applied loads by computing the stresses, tensions and displacements at any point of the structural elements [7].

The solution of a structural dynamics problem is computationally more time-consuming than a static problem, due to the addition of inertia and damping forces and the time dependency of the force and result quantities. In dynamic analysis, the second order differential equations in time that governs the motion of structural problems must be solved for each time step. Direct time integration algorithms or superposition modal analysis are different techniques usually applied for solving this computationally demanding equation of motion and providing the response of the structure along the time. The accuracy of the results obviously depends on the time increment employed, and 3D realistic structural models together with accurate and numerically efficient methods of analysis must be applied.

In order to find the most appropriate structural design, the structural engineer considers different alternatives that must be analysed, varying the size of the structural elements, the material that composes them or the external loads applied. As an example, the Spanish Earthquake-Resistant Construction Standards (NCSE-02) demands a building to be analysed with at least five different representative earthquakes. Once all these structural alternatives are simulated, the results must be interpreted, maybe giving place to a new iteration in this trial-error scheme. Obviously, this situation multiplies, in several orders of magnitude, the computational cost of the problem. The realistic 3D structural dynamic design of large scale structures can thus demand an important computational power, give place a huge volume of data and become one of the most time consuming phases in the design cycle of a civil engineering structure.

Architects and structural engineers need thus powerful software applications able to simulate efficiently the accurate response of the structure. However, commercial applications just offer traditional approaches, computing sequential structural analysis on the user's local machine. As a result, the size and the complexity of the structure to be analysed, the type of structural analysis employed and the total number of the different structural solutions or even earthquakes evaluated are limited by the performance of the computational resources used by the users.

With the purpose of overcoming these limitations, Architrave®[3], an advanced software environment for the design, 3D linear static and dynamic analysis and visualisation of buildings and civil engineering structures was developed. One of the components of Architrave is the Structural Simulator, a batch HPC application that simulates the response of the structure by means of the Finite Element Method. In a previous work, this Structural Simulator was successfully migrated to the Grid [2], providing a Grid Service that offered high performance static and dynamic structural simulations to the structural engineer community.

Notwithstanding, the users traditionally install and run Architrave in their personal computers, and the time spent on the calculations by means of the Structural Simulator component depends on the performance of their machines.

Fortunately, Cloud Computing technology has emerged as an efficient solution for engineering and architecture studios, sharing computing power, storage space, data and software packages.

In [4], a high throughput and reliable Structural Analysis Cloud Service, responsible of performing on-demand remote static and dynamic simulations over the Windows Azure-based Cloud infrastructure provided by the EC VENUS-C project [5], was developed. The migration to the Cloud was implemented by means of the Generic Worker component [6], a web-role implementation for Windows Azure [8][9] that manages the execution of the remote tasks.

The architecture of that implementation is shown in the Figure 1. Firstly, it consists on a Windows client that executes the Architrave GUI to modify the structural properties, apply the external actions, define the simulation and visualise the results. In addition, a GUI tool, called as the Remote Simulation Manager Client, was implemented to submit and manage the simulations in the Cloud, receive the results and inform the user about the status of the simulations. Data communications between the local client and the Cloud service were carried out by means of the standard CDMI service [10]. On the Cloud-side, user authentication was implemented by means of the Security Token Service of the GW, and a Notification Service, also belonging to the GW, was used to inform the user about the status changes of the simulations. Remote jobs were managed and submitted thanks to the Job Management and Submission Service components of the GW. Structural simulations in the Cloud are executed by means of the Structural Simulator, previously mentioned. A Structural Analysis Service Manager module, in charge of launching the Structural Simulator Worker with the appropriate parameters, depending on the type of analysis, and managing the needed simulation input and output data was implemented. Finally, an Elasticity Manager component was developed to provide the system with the elasticity capability, using the Scaling Service of the GW.

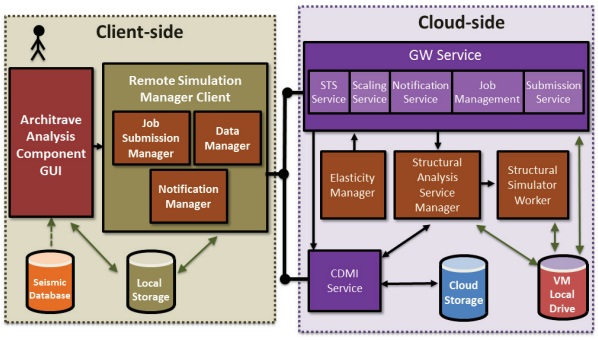


Fig. 1. Architecture of the GW approach.

3 The CodeCloud approach

The aim of the CodeCloud project is to ease the deployment of applications in Cloud environments, so that the effort required to successfully adapt applications to Cloud platforms can be drastically reduced. CodeCloud provides high level services for taking advantage of Cloud platform features from the application developer point of view. It also includes automated mechanisms for building the VMs. Even though some of them are detailed in [11], we provide a concise description for each of them here for the sake of completeness:

- VMRC (Virtual Machine image Repository and Catalog). The VMs consist of a virtual hardware and a Virtual Machine Image (VMI) that represents the hard disk that define the Operating System, applications installed, etc. The VMRC is used to catalogue and index the VMIs in order to be reused by different applications. Every VMI has attached a set of metadata (CPU architecture, Operating System, Hypervisor, etc.) that are used to provide information about its contents. The VMRC includes querying and matchmaking mechanisms that are used to find the VMI that is more suitable for a specific application, according to its requirements and the features of the VMI.
- RADL (Resource and Application Description Language). The RADL is a language that is used to create a declarative description of a virtual infrastructure. Using this language, it is posible to identify the features or to indicate the capabilities of the infrastructure without the need of specifying the concrete values needed by a hypervisor.
- Contextualization and Configuration Manager (CCM): Before delivering a VM to the user, the CCM is in charge of installing the applications, libraries, etc. according to the requirements of the user. It is also responsible for configuring the system to create the specific configuration of the Virtual Infrastructure (i.e. configuring a cluster of VMs according to a Local Resource Management System).
- Infrastructure Manager (IM): The IM coordinates the other components in CodeCloud to deploy and manages the Virtual Infrastructure. It is in charge of processing the RADL requests, selecting the most adequate VMI (with the support of the VMRC), selecting the most appropriate Cloud deployment, and making the effective deployment of the VMs, according to the target hypervisor. It also enables the on-demand modification of the Virtual Infrastructure, by adding or removing nodes.
- Cloud Job Description Language (CJDL): This is the language used to describe a job to be run in the Cloud. The CJDL document includes a piece of RADL with the features of the virtual infrastructure, a block that is used to explain how the application should be executed and a set of elasticity rules that will be employed to vary the number of VMs at runtime.

CodeCloud supports the management of the lifecycle of applications that follow the next programming models: Master/Slave (application based on Bag-of-tasks, Parameter sweep or High throughput schemes), MPI parallel software,

workflows of applications that follow a data-driven or event-driven model and Map/Reduce-based applications. The type of the application to be executed is specified in the CJDL document. Those programming models are supported by a Container system that is specific for each one. The Container gets the RADL document and modifies it to include support for the execution of the application according to the specific model (i.e. including a LRMS system, a separate network, etc.).

The CodeCloud system will be responsible for deploying the infrastructure needed, preparing the VM to meet the user requirements, executing the applications and delivering results to the user. The basic workflow consists as follows: (1) the user prepares the CJDL document that describes the execution of the application and the requirements of the VMs to be deployed; (2) the IM consults the VMRC to get the most suitable VMI for each VM; (3) the IM deploys the infrastructure and uses its Configuration Manager to contextualize the VMs; (4) the CodeCloud Container prepares the input data from the data sources that are provided by the user; (5) the CodeCloud Container executes the application; (6) the results are delivered to the data sinks that are provided by the user; (7) the virtual infrastructure is terminated.

4 Structural Analysis in the Cloud using CodeCloud

The first step to execute the dynamic analysis of buildings in the Cloud using the CodeCloud tools is to identify the requirements of the infrastructure and to write the CJDL document using the proper programming model scheme. In the case of the dynamic analysis of buildings, the simulations are independent tasks that must be executed to get the final results. We are applying a high throughput scheme (its name in CodeCloud is Master/Slave) because we are executing a set of simulations with different parameters, and they are run as if they were submited to a queue system. In the case that we have as many "Slave" VMs as simulations, we will get all the tasks running concurrently. Otherwise, some tasks could be executed sequentially to the previous one. The whole CJDL document is included in listing 12.1.

The virtual infrastructure needed for the execution is described using the RADL language in lines 3 to 24. For the main VM we have no special requirements, just a "ubuntu linux" OS with 512 Mb. of RAM. However, the structural simulator to be run in the working nodes requires some libraries available (lines 17 to 22) and it needs at least 2.5 Gb. of RAM to avoid swapping problems. Regarding the programming model (lines 26-29), we have to specify which kind of node will be used for each of the roles of the Master/Slave scheme and the number of VMs that should be launched for each role.

The CodeCloud service will deal with the transference of input and output datafiles for the user. In lines 32 to 54 we define the data sources and data sinks that are being used for the simulations. CodeCloud analyses the references in the CJDL document and the data containers at run-time and identifies which files should be transferred to be available for the simulations. In particular, this

example will transfer the files with the geometry of the building and the external actions applied that will be used as input data for the executions (defined in the data group "inputdata") and the executable application from the local computer (defined in "datacont1") to the Cloud. Once the result files are available, CodeCloud will filter them to get only those that are specified as results (defined in "datacont2") and they will be available to be retrieved in the local computer (defined in "datacont3"). In case that we had available a CDMI or FTP account, CodeCloud will automatically transfer the data container to it.

CodeCloud considers the abstraction "activity" that should be conceived as a function in a programming language. We have to declare a name for the activity, the executable application that implements it and the parameters that it will need. In our case, we have only one activity (called "banco" and defined in lines 56-71), that is carried out by the executable file "StructuralSimulator" that is stored in the data container called "datacont1" (line 67).

Finally the activity is instantiated (lines 73 to 85), including the parameters that should be used to perform the runs. Appart from the specific numeric parameters of the calculation, it is noticeable that one of the parameters is a reference to the data container "inputdata" (line 75). Such reference will imply that the files stated in such container will be transferred from "datacont1" to the Cloud in order to be available for the calculations. Also, there is a reference to the data container "datacont2" that will trigger the filtering of the files and its transference to the local computer. The files contained in the data container "datacont3" will be available for the user to download them when they are needed for a time that is dependent from the CodeCloud deployment.

Listing 12.1. The CJDL for the Dynamic Analysis of Buildings in CodeCloud

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <Job type="MasterSlave" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation="cjdl.xsd">
3  <Infrastructure>
4          system main (
5                  cpu.arch='x86_64' and
6                  cpu.count>=1 and
7                  memory.size>=512 and
8                  disk.0.os.name='linux' and
9                  disk.0.os.flavour='ubuntu'
10         )
11         system wn (
12                 cpu.arch='x86_64' and
13                 cpu.count>=1 and
14                 memory.size>=2560 and
15                 disk.0.os.name='linux' and
16                 disk.0.os.flavour='ubuntu' and
17                 soft 10 (disk.0.application contains (name='liblapack-dev')) and
18                 soft 10 (disk.0.application contains (name='libscalapack-mpi-dev')) and
19                 soft 10 (disk.0.application contains (name='libparmetis-dev')) and
20                 soft 10 (disk.0.application contains (name='libparmetis3.1')) and
21                 soft 10 (disk.0.application contains (name='libmumps-scotch-dev')) and
22                 soft 10 (disk.0.application contains (name='libpetsc3.1-dev'))
23         )
24         </Infrastructure>
25
26         <Configuration>
27                 <Role name="Main" type="main" count="1"/>
28                 <Role name="Worker" type="wn" count="5"/>
29         </Configuration>
```

```
30
31          <ExecutionData>
32                  <data>
33                          <dataGroup name="datacont1">
34                                  <container uri="lfile:///root/structural_analysis"/>
35                          </dataGroup>
36
37                          <dataGroup name="inputdata">
38                                  <fileset ref="datacont1" file="banco_earthquake1"/>
39                                  <fileset ref="datacont1" file="banco_earthquake2"/>
40                                  <fileset ref="datacont1" file="banco_earthquake3"/>
41                                  <fileset ref="datacont1" file="banco_earthquake4"/>
42                                  <fileset ref="datacont1" file="banco_earthquake5"/>
43                          </dataGroup>
44
45                          <dataGroup name="datacont2">
46                                  <filter input="*.step*">
47                                          <replica dest="datacont3"/>
48                                  </filter>
49                          </dataGroup>
50
51                          <dataGroup name="datacont3">
52                                  <container uri="lfile:///tmp"/>
53                          </dataGroup>
54                  </data>
55                  <definitions>
56                          <activity name="banco">
57                                  <input name="param0" type="File"/>
58                                  <input name="param1" type="Float"/>
59                                  <input name="param2" type="Float"/>
60                                  <input name="param3" type="Float"/>
61                                  <input name="param4" type="Float"/>
62                                  <input name="param5" type="Float"/>
63                                  <input name="param6" type="Float"/>
64                                  <input name="param7" type="Float"/>
65                                  <deployments>
66                                          <cloudDeployment>
67                                                  <executable ref="datacont1"
                                                        file="StructuralSimulator" mpi="no"
                                                        minnodes="1" maxnodes="1"/>
68                                          </cloudDeployment>
69                                  </deployments>
70                          </activity>
71                  </definitions>
72
73                  <executions>
74                          <execution name="banco" activity="banco" timestamp="10">
75                                  <input name="param0" source="inputdata"/>
76                                  <input name="param1" value="0.25"/>
77                                  <input name="param2" value="0.5"/>
78                                  <input name="param3" value="0"/>
79                                  <input name="param4" value="0"/>
80                                  <input name="param5" value="12"/>
81                                  <input name="param6" value="0.01"/>
82                                  <input name="param7" value="0.05"/>
83                                  <output name="output" destination="datacont2"/>
84                          </execution>
85                  </executions>
86          </ExecutionData>
87  </Job>
```

Once the CJDL document is prepared, the user just has to use the CLI utilities to submit it to the Cloud and the CodeCloud service will take care of the tasks that are needed to carry out the execution. One of the main benefits of this CJDL is that it is independent from the Cloud deployment. That means that if it is available an implementation to deploy VMs in Amazon, Windows

Azure or an on-premises deployment such as Open Nebula or Open Stack, the same CJDL document will run on any of them.

5 Comparing CodeCloud and Generic Worker approaches

In order to compare the performance of the Cloud Services based on CodeCloud and the Generic Worker, the same real case study employed in [4] has been chosen. It consists of a reinterpretation, according to the current usages and regulations, of the original structure of the Nordic Bank, located in Helsinki (see Figure 2). The model has been designed with bars for columns and beams, and 2D medium sized finite elements for slabs and walls.

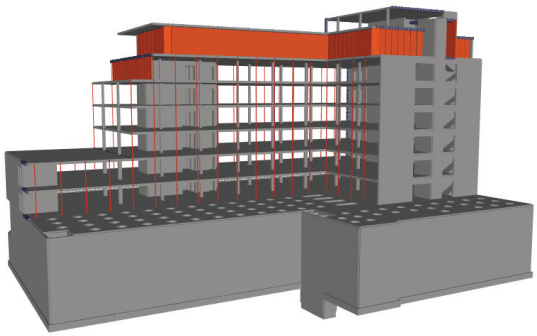


Fig. 2. Nordic Bank building.

The aim of this case study was to select the best one of the ten different available structural solutions that accomplished the structural and safety regulations and presented the most economic final cost. As demanded by the Spanish regulations, each of these structural solutions was tested under the influence of five representative earthquakes. Each structural solution contains 253812 degrees of freedom, with 1306 columns and beams, and 68751 2D finite elements. The dynamic response of the structure was simulated under the influence of a seismic load of 12 seconds of duration, with a simulation time increment equal to 0.01 seconds. These results were stored on disk every 0.05 seconds.

Each of these dynamic simulations lasted about 120.03 minutes using a conventional PC (Intel core i5 architecture with 3.20 GHz of CPU frequency and 4 Gb. of RAM) and 2.48 Gb. of result data were generated if the results of the 240 previously mentioned time steps were stored. For the input file, 5.43 Mbytes were needed. Therefore, the execution of the case study with the 50 simulations spent 100.03 CPU hours in a traditional approach, generating output results in the order of 124.04 Gb. As input data, 271.5 Mbytes were employed.

On the Cloud side, the simulation of one of the structural solutions using the GW approach lasted 277.15 minutes. Table 1 shows the time dedicated to the different phases of executing one simulation in the Windows Azure infrastructure, compared to the local computation. The diferent phases involve (i) sending the input data to the Cloud, (ii) preparing the execution in the Cloud, (iii) executing the application, (iv) sending the results to the Cloud data accounts, and (v) retrieving the results of the most unfavorable time steps in the client computer.

Table 1. Execution time of each of the different stages involved in the simulation in the Cloud in the Generic Worker approach, for each simulation.

	Data and job submission	Job initialization	Structural Simulation	Result to the Cloud Storage	Download to the client
Local	-	-	120m02s	-	-
Remote	6s	39s	275m19s	50s	34s

When launching the simulations using the CodeCloud approach, we have to consider the time needed to deploy and to configure the infrastructure. This overhead is not required to consider in the Windows Azure deployment because it takes place just once during the service deployment process instead of once for each application execution. The VMs deployment time depends on the number of VMs that are used as WNs and the type of data network. Table 2 includes the time needed to have the infrastructure ready to execute each of the executions, depending on the number of VM used. It is needed to deploy one Container, in a VM, that will configure the VMs involved in the execution of the simulations and will contextualize them. The table includes the time involved in each phase, along with the total time employing 1, 5 and 10 VMs.

Table 2. Time to deploy the infrastructure in the OpenNebula-based CodeCloud deployment.

# of VM	Container deployment	Other VMs	Context.	Total
1	4m12s	3m42s	5m59s	13m53s
5	2m12s	6m32s	13m04s	21m47s
10	2m21s	12m42s	22m53s	37m56s

The main difference of the executions is that the Web Role VMs in Windows Azure are based on the Windows Operating System. Since there was no Windows version for the dynamic analysis of buildings application, is was needed to invest additional time to have the application available for this OS after having the application available for the Grid. For the CodeCloud tests, it was used the

native version of the application that was already based on the Ubuntu Linux OS.

In order to compare the Generic Worker and the CodeCloud approaches using a real use case, a set of 50 different 3D dynamic simulations were launched and computed using both alternatives with different number of VMs.

For the Generic Worker, the set of 50 different 3D dynamic simulations were executed on two different deployments composed of 1 and 10 medium-sized Web role Azure instances (1.60 GHz of CPU frequency and 3.5 Gb. of RAM). For CodeCloud, the same number of simulations were run on three distinct deployments, composed of 1, 5 and 10 VMs. For that, we have used an OpenNebula-based Cloud deployment comprised for four nodes with 16 cores and 16 Gb. of RAM each one. The size of the VMs was tailored to each of the roles: the main node was a 2.2GHz CPU node with 512 Mb. of RAM and the working nodes were 2.2GHz CPU nodes with 2.5 Gb. of RAM.

Table 3. Comparing the execution time (in hours): Generic Worker vs CodeCloud.

-	1	5	10
GW	231.21	-	24.55
CodeCloud	151.11	31.04	16.23

Table 3 shows the results of the comparison between the Generic Worker and the CodeCloud approaches. Although better execution results have been obtained employing just one VM in CodeCloud, the Generic Worker implementation offered better response times for a Cloud infrastructure composed of 10 VMs.

Anyway, it should be taken into account the important reduction of the total time involved in the design of the building. Whereas 100.03 hours were needed in the traditional approach, just 24.55 hours and 37.34 were spent when computing remotelly all the simulations in the Cloud, employing the Generic Worker or CodeCloud respectively.

Finally, we should have in mind that an important time was involved in the generation of the Windows-based version of the structural simulator to be executed in the Windows Azure infrastructure, specially in the time related to the compilation of the different needed numerical libraries. Morever, the user must generate a Cloud Service that manages the remote structural simulations, where the API provided by the Generic Worker is used. Therefore, a substantial time was spent on its adaptation to the Cloud. However, the CodeCloud approach executes the original Linux-based application, without any necessity of modification, and the user just has to generate the CJDL file, describing the hardware and software dependencies of the application. In this way, the time-to-the-cloud was almost immediate.

6 Conclusions

In this paper we have described two different approaches to migrate a structural analysis application to the Cloud. One of them is based on the Generic Worker component, where a new version of the structural simulator, able to be executed in a Windows platform, was necessary generated. The other one is based on CodeCloud, where the same Linux legacy structural simulator run in the Grid was employed, without any modification.

The Generic Worker implementation offered better performance when using numerous VMs than the approach based on CodeCloud, from the scalability point of view. Notwithstanding, it is important to take into account that the implementation of the Generic Worker-based application required a notable learning curve and an important development time, whereas the effort of executing the structural application in the Cloud infrastructure managed by CodeCloud was considerably reduced.

For the future it would be interesting to create new plugins for CodeCloud that enabled the execution of the application in the Windows Azure platform. That connectors would allow comparing the different versions of the application. Another interesting approach would be to run the two versions in a "neutral" platform such as Amazon EC2.

The CodeCloud tools have demonstrated that it is possible to reduce the time-to-the-cloud for some scientific applications. Using such tools the user just needs to have its running version of the application, define the CJDL document and execute the simulations.

Acknowledgements

The authors wish to thank the financial support received from the Spanish Ministry of Economy and Competitiveness to develop the project CodeCloud (TIN2010-17804).

References

1. C. de Alfonso, M. Caballer, F. Alvarruiz, G. Moltó, An economic and energy-aware analysis of the viability of outsourcing cluster computing to a cloud, Future Generation Computer Systems 29 (3) (2013) 704 – 712, special Section: Recent Developments in High Performance Computing and Security. doi:10.1016/j.future.2012.08.014.
2. J. M. Alonso, V. Hernández, R. López, G. Moltó, Grid4Build: A High Performance Grid Framework for the 3D Analysis and Visualisation, in: Proceedings of the Iberian Grid Infrastructure, 2007, pp. 353–364.
3. A. Pérez, F. Gómez, A. Alonso, V. Hernández, J. Alonso, P. de la Fuente, P. Lozano, Architrave: Advanced analysis of building structures integrated in computer-aided design, in: C. L. Millan (Ed.), 2nd International Conference on Construction and Building Research (COINVEDI12), 2012, p. 37.

4. J. Alonso, A. Alonso, P. de la Fuente, F. Gómez, V. Hernández, P. Lozano, A. Pérez, A cloud system implementation for the analysis of civil engineering structures, in: Proceedings of the 2012 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA12), CSREA Press, 2012, pp. 210–216.
5. Venus-c project website, http://www.venus-c.eu (2012).
6. C. Geuer-Pollmann, The venus c generic worker, SICS Cloud Day (2011).
7. A. Chopra, Dynamics of Structures: Theory and Applications to Earthquake Engineering, Prentice-Hall International Series in Civil Engineering and Engineering Mechanics, Prentice Hall, Prentice-Hall International, 2001.
8. E. Pace, D. Betts, S. Densmore, R. Dunn, M. Narumoto, M. Woloski, Moving Applications to the Cloud on the Microsoft® Azure™ Platform, Microsoft Press, 2010.
9. E. Pace, D. Betts, S. Densmore, R. Dunn, M. Narumoto, M. Woloski, Developing Applications for the Cloud on the Microsoft® Windows Azure™ Platform, Microsoft Press, 2010.
10. I. Livenson, E. Laure, Towards transparent integration of heterogeneous cloud storage platforms, in: Proceedings of the fourth international workshop on Data-intensive distributed computing, DIDC '11, ACM, New York, NY, USA, 2011, pp. 27–34. doi:10.1145/1996014.1996020.
11. C. de Alfonso, M. Caballer, F. Alvarruiz, G. Molto, V. Hernández, Infrastructure Deployment Over the Cloud, in: 2011 IEEE Third International Conference on Cloud Computing Technology and Science, IEEE, 2011, pp. 517–521. doi:10.1109/CloudCom.2011.77.

Session IV

Scientific Applications in Distributed Infrastructures

Leveraging Hybrid Data Infrastructure for Ecological Niche Modelling: The EUBrazilOpenBio Experience

Ignacio Blanquer**[1], Leonardo Candela[2], Pasquale Pagano[2], and Erik Torres[1]

[1] Institute of Instrumentation for Molecular Imaging (I3M), Universitat Politècnica de València, Camino de Vera s/n, Valencia, Spain,
Email: {iblanque, ertorser}@i3m.upv.es
[2] Istituto di Scienza e Tecnologie dell'Informazione (ISTI), Consiglio Nazionale delle Ricerche (CNR), Via G. Moruzzi 1, Pisa, Italy
Email: {candela, pagano}@isti.cnr.it

Abstract. Ecological niche models are essential instruments in the development of strategies and policies in various application domains. The development of ever better models requires that scientists and practitioners are provided with (*i*) both the data and the processing capacities they need on demand and (*ii*) innovative mechanisms supporting the effective sharing of such products at scale. This paper describes the Ecological Niche Modelling service developed by the EUBrazilOpenBio project. This service is offered by relying on a underlying Hybrid Data Infrastructure (HDI) that (*i*) aggregates data, services and processing capacities from existing infrastructures and information systems and (*ii*) offers the aggregated resources in a seamless way. The paper presents the overall architecture of the service and details how its constituents have been developed by leveraging HDI offering on-demand data and processing facilities.

1 Introduction

Ecological niche models [17, 16, 31] aiming at estimating the presence of a species in a given area are essential instruments in the development of strategies and policies in various application domains. For example, ecological niche models have been used to predict the abundance and impact of invasive species [5, 21], to model distributions of disease vectors at continental scales [15], to study genetic diversity [29], and to study climate change consequences to biodiversity [11, 24]. These models rely on occurrence records to investigate the relationships between observed species presence and the underlying environmental parameters that – either directly or indirectly – determine a species distribution in a known area and use this information to predict the probability of a species occurrence in other areas [18].

In order to support the production of these models in a large scale, it is needed to develop innovative working environments for individual scientists potentially having at their disposal scarce resources. These environments should

** e-mail of corresponding author: iblanque@i3m.upv.es

support complex and iterative processes which include at least the following key steps [16]: (*i*) identification of *relevant data*; (*ii*) *modelling*, i.e., deciding how to deal with the correlated prediction variables, selecting the appropriate algorithm, training the model, assessing the model; and (*iii*) *mapping* predictions to geographic space.

Scientist normally take models created by experts and work with them. The individual problem of creating a model, testing it and making a projection may take few minutes, depending on the number of occurrence points, the algorithm used and the resolution and number of layers. For instance, if the Brazilian Virtual Herbarium would like to create models for their 31,718 angiosperms species (18 Sept. 2012), and assuming that 30% of them will have enough points to generate models, the computational cost will go beyond 10 months (495k models, 540k tests and 90k projections). The problem will be even more complex if the spatial resolution is increased, or if the models consider not only Brazil and different climate conditions. Moreover, the models may be regenerated every time new data is available for each species.

There are some developments that makes the realisation of these working environments feasible including the availability of data and tools. Having a critical amount of species occurrence data is a pre-requisite to create a large repository of models, which have the potential to make significant impact in conservation biology and conservation planning. The consolidation of large biodiversity databases with high-quality data, such as GBIF [14], is a breakthrough in the way that ecological niche modelling is made. While many areas of research, such as life sciences, biomedicine or bioinformatics, are already benefiting from this approach, the exploitation of massive computing and data resources to accomplish ecological niche modelling presents an unprecedented opportunity to improve the quantity and quality of models available, also allowing the automatic computation of new models for geographical areas or species that has not been studied before. The availability of tools supporting the production of ecological niche models, such as openModeller [25], is another opportunity for scientists. openModeller is an open source software realising a single computing framework capable of handling different data formats and multiple algorithms that can be used in potential distribution modelling. The motivations leading to its development result from the willingness to provide scientists with a unified environment for species' distribution modelling. openModeler systematises the steps related to data management and to the algorithms exploited for modelling. It may exploit distributed computing resources through HTCondor [30] but it requires tailored worker nodes specifically equipped with openModeller libraries. It does not provide any facility for data discovery. It is expected that users acquire data by themselves and transform them to the supported formats. The current version supports a number of algorithms including ANN, Bioclim, GARP, SVM, and Envelope Scores. Its functionality is offered via a command line tool, a desktop tool as well as via a Web service.

However, there are some facts that makes the realisation of the envisaged working environments challenging, namely (*i*) relevant data are potentially huge, heterogeneous and scattered across many Information Systems, (*ii*) the produc-

tion of models is a computational intensive task that potentially goes beyond the capacities of an average single scientist.

This paper presents an innovative facility for ecological niche models that benefits from the EUBrazilOpenBio Hybrid Data Infrastructure [9], i.e., an IT infrastructure offering *resource-as-a-service* where the resources range from computing capacities to data and software. The HDI offers a rich array of data and data management services by leveraging other infrastructures (including Grid and Cloud) and Information Systems. Thus the HDI acts as a mediator for scientists by providing them with a unified and transparent way to access the data they need and the computing power by borrowing them on demand from heterogeneous providers.

The remainder of the paper is organised as follows. Section 2 gives a comprehensive description of the Ecological Niche Modelling facilities developed in the context of the EUBrazilOpenBio initiative. In particular, the section reports (*i*) the overall architecture of the services collaboratively realising the needed facilities (Sec. 2.1); (*ii*) the mechanisms and services supporting data management (Sec. 2.2); (*iii*) the mechanisms and services supporting effective data processing (Sec. 2.3); and, (*iv*) the web-based environment providing scientists with user friendly tools for niche modelling (Sec. 2.4). Concluding remarks are given in Section 3.

2 The EUBrazilOpenBio Ecological Niche Modelling Service

The EUBrazilOpenBio Ecological Niche Modelling service was devised to be offered *as-a-Service* in the context of a biodiversity-oriented Hybrid Data Infrastructure. It relies on a number of technologies including openModeller for niche modelling core facilities. In essence, it manifests in a service that can be used via a set of portlets or via a set of Web APIs (Application Programming Interfaces). Behind the scene, a number of Web services and software libraries contribute to realise the back-end needed to implement the resulting environment by integrating and interfacing with existing systems, infrastructures and information systems.

2.1 Overall Service Architecture

Ecological niche modelling requires data-intensive processing capabilities. Figure 1 presents a schema of the Ecological Niche Modelling Service (ENM Service), which was specifically conceived to leverage computing and data resources delivered *as-a-Service* by the EUBrazilOpenBio HDI.

A portlet (ENM GUI) facilitates the use of the ENM Service. This portlet is integrated in the EUBrazilOpenBio Gateway[3], which enables users with access to a variety of tools to support collaborative research. The organization of such tools in Virtual Research Environments (VREs) [8, 6] provides a practical way in which users may effectively customize the tools to suit their specific needs.

[3] EUBrazilOpenBio Gateway: `https://portal.eubrazilopenbio.d4science.org/`

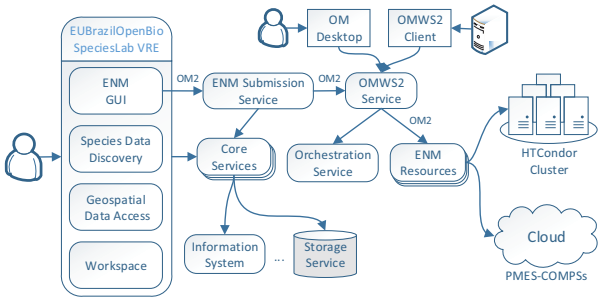


Fig. 1. Ecological Niche Modelling Architecture

In addition to the ecological niche modelling portlet, others, such as the Species Data Discovery and Access Service (cf. Sec. 2.2) or the Workspace, are available to assist users in preparing ecological niche modelling experiments and analysing the results produced by these experiments.

The core services of the EUBrazilOpenBio infrastructure mainly rely on gCube [7, 8] facilities and D4Science.org[4] support. Among these services, one of the most relevant for ecological niche modelling is the Information System, which is used to publish and discover information about the services available in the infrastructure. The Storage Service is also important for storing results and other information produced in the experiments, such as the execution traces.

Moreover, gCube provides a way to seamlessly integrate the different services and applications that supports the execution of ecological niche modelling experiments in the infrastructure. In particular, the services can be registered with the Information Service, facilitating their discovery and usage. Service registration has the additional benefit of not requiring endpoints with fixed paths. Instead of associating a service with a specific endpoint, clients can search the Information System for the services they need. Also, gCube provides system administrators with a way to automatically deploy their services to the gCube Hosting Nodes (gHN). These nodes are continuously monitored and, in some cases, corrective actions can be taken in a fully automatic way.

GUI operations are implemented through the ENM Submission Service, which manages the complete lifecycle of the experiments submitted from the GUI. However, the OMWS2 Service is the central entry point to the ENM Service, providing access to the computing back-ends that are available to execute ecological niche modelling experiments in EUBrazilOpenBio. The access to these computing facilities can occur in several ways: (*i*) through PMES-COMPSs [23] to access Cloud-based resources; or (*ii*) by using customized services to access computational clusters managed with HTCondor. The interaction with the OMWS2

[4] D4science.org: `http://www.d4science.org`

Service, no matter how the access to the computing back-end is carried out, is achieved through a common set of operations provided by the OMWS[5] API.

By using this approach, a variety of OMWS clients, including the openModeller Desktop[6], are supported directly in EUBrazilOpenBio. In this way, users have also the option to either use the openModeller Desktop or any OMWS-compatible client to access directly the OMWS2 Service offered by the infrastructure through a dedicated endpoint[7].

The version 2 of the openModeller data format (OM2)[8] is the language used to intercommunicate the different processes, components and services that make up the ENM Service.

2.2 Data Management

In order to support the ecological niche modelling service, the EUBrazilOpenBio infrastructure should provide facilities for managing (*i*) biodiversity data, namely species occurrences, (*ii*) environmental data, and (*iii*) file-oriented content. The former two typologies of data are used as input data for the niche modelling activity, while the latter is used for storing data accompanying a modelling activity including experiment specification, execution logs, and static images resulting from model projection.

For biodiversity data, the EUBrazilOpenBio HDI has been equipped with the *Species Data Discovery and Access Service* (SDDA). This service provides users with seamless access to both nomenclature data and species occurrences from major information systems including GBIF [14] and speciesLink [2] for occurrence data, CoL [28] and List of Species of the Brazilian Flora [1] for nomenclature data. The SDDA service promotes a data discovery mechanism based on species name(s), either the scientific name or the common name of the target species. Moreover, to overcome the potential issues related to taxonomy heterogeneities across diverse data sources, the service supports an automatic query expansion mechanism, i.e., the query might be augmented with "similar" species names. Also, queries might be augmented with criterion aiming at explicitly selecting the databases to search and the spatial and temporal coverage of the data. Discovered data are presented in a homogenised form, e.g., in a typical Darwin Core [32] format. The service is designed to promote the openness with respect to the information systems it is capable to interface. In fact it is sufficient to implement (or reuse) a plug-in in order to enlarge the number of information systems and data sources integrated into SDDA. Each plug-in interacts with an information system or database by relying on a standard protocol, e.g., TAPIR [12], or by interfacing with its proprietary protocol. Every plug-in

[5] OMWS v. 2 Definition: `http://openmodeller.cria.org.br/ws/2.0/openModeller.wsdl`

[6] openModeller Desktop: `http://openmodeller.sourceforge.net/index.php`

[7] EUBrazilOpenBio OMWS2 endpoint `http://enm.eubrazilopenbio.d4science.org/om/omws2`

[8] openModeller v. 2 data format: `http://openmodeller.cria.org.br/xml/2.0/openModeller.xsd`

mediates queries and results from the query language and data model envisaged by SDDA to the requirements of a particular database.

For environmental data, the EUBrazilOpenBio HDI has been equipped with services realising the facilities of a *Spatial Data Infrastructure* (SDI) by relying on state-of-the-art technologies and standards. In particular, for the sake of the niche modelling, the infrastructure offers standard-based services for data discovery (a catalogue), storage and access (a federation of repositories), and visualisation (a map container). The standard-based catalogue service enables the discovery of geospatial data residing in dedicated repositories by relying on GeoNetwork [27] and its indexing facilities. GeoNetwork is the OGC recommended catalogue supporting ISO19115, ISO19119, and ISO19110 metadata standards represented and transported via the ISO19139 standard. For data storage and access, EUBrazilOpenBio relies on a federation of repositories built by relying on GeoServer [20] and THREDDS [26] technologies. In essence, the infrastructure hosts a number of repositories and a *GIS Publisher Service* that is conceived to enable the publication of geospatial data by relying on an open set of back-end technologies for the actual storage and retrieval of the data. Because of this, the GIS Publisher Service is designed with a plug-in-oriented approach where each plug-in interfaces with a given back-end technology. To enlarge the array of supported technologies it is sufficient to develop a dedicated plug-in. Metadata on available data are published via the GeoNetwork catalogue. For data visualisation, the infrastructure offers *Geo Explorer* and *GIS Viewer*, two components dedicated to support the browsing and visualisation of geospatial data. In particular, the Geo Explorer is a web application that allows users to navigate, organize, search and discovery layers from a GeoNetwork instance via the CSW protocol. The GIS Viewer is a web application that allows users to interactively explore, manipulate and analyse geospatial data.

For file-oriented content, the EUBrazilOpenBio HDI has been equipped with the *gCube Storage Manager*, a Java based software library that presents a unique set of methods for services and applications running on the infrastructure and willing to store files. This software library relies on a network of distributed storage nodes managed via specialized open-source software for document-oriented databases. In its current implementation, two possible document store systems are used [10], MongoDB and Terrastore, while the implementation relaying on a third one, U.STORE [13], is ongoing. The software library was designed to promote the openness with respect to the storage back-end technology thus to reduce the time required to add a new storage system to the HDI and rely on it. In essence, it is sufficient to develop a specific mediator, deploy it and configure the Storage Manager accordingly.

2.3 Data Processing

Most of the ecological niche modelling experiments that are performed in EUBrazilOpenBio go through several stages of computation before a final result can be obtained. This is the case, for example, when a new model is computed with high-resolution climate layers (e.g., 30-arcsec WorldClim Version 1.4 Bioclimatic Variables [19]) for a large number of species occurrence points, and

then validated using a statistical approach based on cross-validation –which is the common practice currently. This process may take several hours or days to complete, depending on the complexity of the modelling algorithm. The EU-BrazilOpenBio infrastructure relies on Distributed Computing Infrastructures (DCI), such as Grid and Cloud computing systems, to provide users with access to massive computing and data resources, which contributes to reduce the average execution time. Through the use of these systems, users have found that in some cases it is more effective to run several experiments in parallel rather than sequentially. After the experiments are completed, the result can be analysed and compared by the expert to select manually the models with sufficient quality to be used in further studies.

The openModeller web service interface (OMWS)[9] provides common programmatic access to the ecological niche modelling facilities that are made available in an open-access form to scientists and developers through the EU-BrazilOpenBio infrastructure. OMWS was extended to meet the specific requirements for the execution of multi-staging and multi-parametric experiments [4]. These extensions were planed and undertaken in conjunction with the openModeller developers team to make them available to the openModeller community.

One of the main advantages of OMWS is that allows the complete specification of ecological niche modelling experiments, using the resources available in the infrastructure. This includes the occurrence datasets required to build and test models, the parameters of the modelling algorithms and the information to access the climate layers. Besides the expected operations to submit new experiments and to retrieve the results of already completed ones, OMWS defines operations that enable clients to discover which ecological niche modelling algorithms the service offers, as well as the climate collections that are available to the clients. Even though biodiversity databases can be directly queried for occurrence points, the SDDA service (cf. Sec. 2.2) provides an integrated, seamless interface for combining results from several databases in the same query.

However, OMWS does not cover all the aspects that are encountered in a multi-tenant application. A further, important consideration in any realistic application is the identification of the user who initiated the experiment request. This feature is not readily supported in the OMWS specification, but it is needed to: (*i*) manage (e.g., index or query) the experiments submitted by a specific user; and (*ii*) plan the use of the resources effectively, prioritizing critical requests (e.g., from a priority project) over non-critical ones, helping to ensure that the infrastructure would support the required performance.

This last aspect has been addressed in the ENM Submission Service (cf. Fig. 1), which is a separate service that manages the complete lifecycle of the experiment. In addition to keeping track of the experiments, the ENM Submission Service monitors the progress of the execution, automatically resubmitting failed jobs and managing the products of the experiments. This relieves applications of accomplishing such goals. Ecological niche models, as well as model projections and statistical results that are produced in the experiments, are initially

[9] The openModeller Web Service: `http://openmodeller.sourceforge.net/web_service.html`

stored in the computing back-end where the experiment is actually executed and then transferred to a shared storage in the infrastructure, before users can use them. This is also the case for the execution traces, which follow a similar pathway. The ENM Submission Service periodically checks the completion of the experiments and moves the results and the execution traces to their final location in the infrastructure. An interesting feature of the ENM Submission Service is that its public API, which is accessible from the GUI, uses the information that is locally available to the service. This means that when a user queries the ENM application for the status of his/her experiments, the ENM Submission Service will return the last status retrieved by the monitoring system, instead of querying recursively the computing back-end where the experiment is actually executed. In this way, the possible delays due to the network traffic are hidden from the user, contributing to provide a sense of real-time interactively in the ENM application.

The OMWS2 Service is a server implementation compatible with the OMWS Version 2.0 API. Contrary to the openModeller server, the OMWS2 Service does not connect directly with either the ecological niche modelling library or the job scheduler[10]. Instead, the OMWS2 Service uses the Orchestration Service of the infrastructure to select the most appropriate computing back-end for the execution of the experiment. The Orchestration Service is a meta-scheduler that dynamically discovers the available execution resources and performs a mapping between requests and resources, based on the requirements of computing power and data availability of the request (e.g., access to a specific modelling algorithm or climate layer set) and the workload of the resources.

Another major difference is that the OMWS2 Service can be extended with new plug-ins to new computing back-ends. EUBrazilOpenBio provides plug-ins to access PMES-COMPSs services [23] and OMWS2 instances. The VENUS-C Cloud infrastructure [3] and a dedicated HTCondor cluster [30] are the two major providers of computing resources in the infrastructure. While VENUS-C is accessible through a PMES-COMPSs service, HTCondor is not publicly available in the infrastructure and the OMWS2 API provides access to this resource.

The OMWS2 Service does not necessarily require synchronized ecological niche modelling facilities –same version of the ecological niche modelling algorithms are not needed– or data between the different computing resource providers because this information is readily accessible to the Orchestration Service, which allocates the resource according to it. In the case of the climate layers, missing layers are automatically downloaded to the local storage of the computing back-end from the GeoServer.

One of the most important details of the EUBrazilOpenBio ecological niche modelling service is that it connects users with experiments and computing resources. For example, when a user submits a new experiment to the infrastructure, the ENM Submission Service automatically registers the submission date and other details that may be important to the user. The request itself is stored as a document based on the openModeller data format that can be copied, edited or shared with other users of the infrastructure. Similarly, the OMWS2 Service

[10] openModeller Server: `http://openmodeller.sourceforge.net/om_server.html`

registers where the request was sent to – the computing back-end where the experiment is actually executed –, as well as the identifiers assigned in the computing back-end to the jobs that are included in the experiment execution. In this manner, the information of the experiment can be retrieved and presented to the user in the most appropriate way. One of the most useful reports allows the user to list his/her experiments, filtering and sorting the list in various ways. This is addressed in detail in the Section 2.4.

The experiment information is stored in two different databases: the user information is stored in the accounting database, while the allocation database stores the information of the resource. These databases only share the identifier of the experiment, which is unique across the infrastructure. The accounting database is accessed mainly from the ENM Submission Service and is designed to provide fast access to the details of the experiments of the authenticated user. On the other hand, the allocation database is mainly accessed to consult the status of completeness of an experiment from the OMWS2 Service. Therefore, this database is optimized to support this kind of access.

2.4 The Graphical User Interface

Usability is one of the key aspects that drive the adoption of new tools and applications. In EUBrazilOpenBio, the use of well-established tools and protocols, such as openModeller and OMWS, allows the engagement of prospective users in the design and validation of the ENM Service, enabling a broader user community the ability to evaluate the service and provide their feedback. Since the interface of the service was designed to be compatible with OMWS, any application that implements this API can easily access the computing and data capabilities provided by EUBrazilOpenBio.

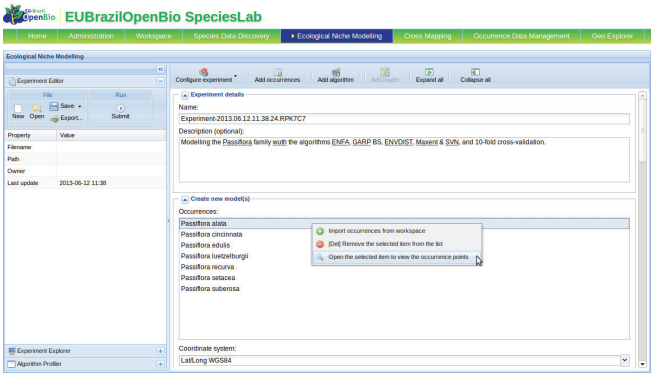


Fig. 2. Ecological Niche Modelling Experiment Editor

However, there is a group of features that were developed in EUBrazilOpenBio to improve data utilization and are not available in OMWS. These features

include the possibility of creating new data collections, for example, by querying simultaneously several databases for occurrences. Moreover, users may upload their own geospatial datasets, which is not possible with OMWS. Data collections and research findings can be shared with other researchers.

All these improvements in data management have lead to the design of a new GUI for ecological niche modelling. This new GUI was conceived as an experiment-centric environment that provides the necessary methods to retrieve, filter and format the input parameters for the experiment, and also to submit the experiment for execution on the computing back-ends, retrieve and analyse its results. Fig. 2 shows a screenshot of the ecological niche modelling exper-

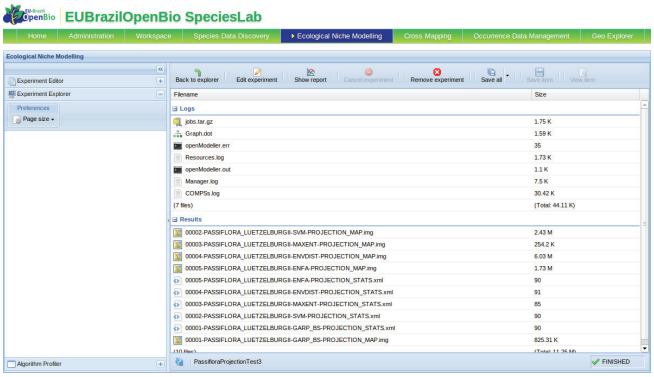


Fig. 3. Ecological Niche Modelling Experiment Explorer

iment editor, which is the main entry point of the GUI. Although the design of the interface was initially inspired by openModeller Desktop, the new GUI has introduced many features that are not present in this application. Among these features, the improvement of the sampling capabilities is an important characteristic of the GUI that allows users to set-up their occurrence datasets to be usable for cross-validation, bootstrapping or sub-sampling, depending on the desired validation strategy. Fig. 3 shows a screenshot of the ecological niche modelling explorer, which list all the experiments and provides access to the results and the report generation facilities. Report tools were also redesigned to produce more expert-like interpretations of the results, including comparative charts, tabular views, error analysis reports, etc. Moreover, results can be exported in comma-separated values (CSV) file format, which facilitates their analysis with other tools as well as format them for papers and reports.

For models visualisation, the GUI relies on the fact that these are published in the SDI. Fig. 4 shows a screenshot of the geospatial information viewer. Projected models can be selected to visualize them in the same view, thus facilitating the comparison of different predictions (e.g., different modelling algorithms or different climate layers).

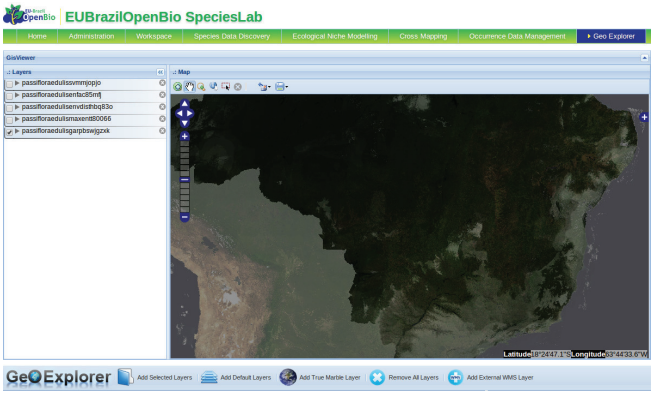


Fig. 4. Geospatial Information Viewer (GIS Viewer)

However, the most significant difference between EUBrazilOpenBio and other applications based on OMWS is that our service is the first implementation of the version 2.0 of this protocol. For this reason, the ENM GUI has a number of unique characteristics and advantages for the type of experiments that are addressed with this new release of OMWS. For example, the ENM GUI facilitates the realization of multi-parametric studies that involve variation in one or more parameters, such as different climate conditions or ecological niche modelling algorithms with customized parameters.

3 Concluding Remarks

The effective and large scale production of ecological niche models is a data and computation intensive gas that calls for innovative working environments. This paper presented the innovative set of facilities developed by EUBrazilOpenBio infrastructure to leverage Hybrid Data Infrastructure offering where data, computing capacities and services are made available transparently and on demand. In particular, the set of services developed to realise the expected facility have been discussed ranging from those supporting the access to the diverse typologies of data to those oriented to support the execution of tasks by relying on diverse computational facilities. All of these services have been integrated in a Service-oriented application.

The potential benefits resulting from this environment are both quantitative and qualitative. Experimental results [22] demonstrate that the ENM service reaches good performance running on an on-demand provided environment (with an average performance loss around 9.6% with respect to a dedicated cluster), reaching a speed-up above 5 with the 10 machines. The potential benefits of such an integrated environment are evident from the final user perspective since they are provided with a working environment offering the data and the tools they need, thus they can focus on the scientific investigation only.

EUBrazilOpenBio is starting a validation process to assess the impression of final users on the whole platform, considering performance and usability. Moreover, a sustainable plan is being defined which will use this validation input to define assets and future exploitation opportunities.

Acknowledgements

EUBrazilOpenBio - Open Data and Cloud Computing e-Infrastructure for Biodiversity (2011-2013) is a Small or medium-scale focused research project (STREP) funded by the European Commission under the Cooperation Programme, Framework Programme Seven (FP7) Objective FP7-ICT-2011- EU-Brazil Research and Development cooperation, and the National Council for Scientific and Technological Development of Brazil (CNPq) of the Brazilian Ministry of Science, Technology and Innovation (MCTI) under the corresponding matching Brazilian Call for proposals MCT/CNPq 066/2010.

References

1. List of Species of the Brazilian Flora. `http://floradobrasil.jbrj.gov.br/`, 2013.
2. speciesLink. `http://splink.cria.org.br`, 2013.
3. VENUS-C: Virtual multidisciplinary ENvironments USing Cloud infrastructures. `http://www.venus-c.eu`, 2013.
4. R. Amaral, R. Badia, I. Blanquer, L. Candela, D. Castelli, R. Giovanni, W. Gray, A. Jones, D. Lezzi, P. Pagano, V. Perez-Canhos, F. Quevedo, R. Rafanell, V. Rebello, and E. Torres. EU-Brazil Open Data and Cloud Computing e-Infrastructure for Biodiversity. In *Proceedings of International Workshop on Science Gateway (ISWG 2013)*, 2013.
5. N. Barve, V. Barve, A. Jiménez-Valverde, A. Lira-Noriega, S. Maher, A. T. Peterson, J. Soberón, and F. Villalobos. The crucial role of the accessible area in ecological niche modeling and species distribution modeling. *Ecological Modelling*, 222(11):1810 – 1819, 2011.
6. L. Candela, D. Castelli, P. Pagano. Virtual Research Environments: an overview and a research agenda. *CODATA Data Science Journal*, –, 2013.
7. L. Candela, D. Castelli, and P. Pagano. gCube: A Service-Oriented Application Framework on the Grid. *ERCIM News*, (72):48–49, January 2008.
8. L. Candela, D. Castelli, and P. Pagano. Making Virtual Research Environments in the Cloud a Reality: the gCube Approach. *ERCIM News*, (83):32–33, October 2010.
9. L. Candela, D. Castelli, and P. Pagano. Managing big data through hybrid data infrastructures. *ERCIM News*, (89):37–38, 2012.
10. R. Cattell. Scalable SQL and NoSQL data stores. *SIGMOD Rec.*, 39(4):12–27, May 2011.
11. G. C. Costa, C. Nogueira, R. B. Machado, and G. R. Colli. Sampling bias and the use of ecological niche modeling in conservation planning: a field evaluation in a biodiversity hotspot. *Biodiversity and Conservation*, 19(3):883–899, 2010.
12. R. De Giovanni, C. Copp, M. Döring, A. Güntscg, D. Vieglais, D. Hobern, J. Torre, J. Wieczorek, R. Gales, R. Hyam, S. Blum, and S. Perry. TAPIR - TDWG Access Protocol for Information Retrieval, 2010. Version 1.0.
13. F. A. Durão, R. E. Assad, A. F. Silva, J. F. Carvalho, V. C. Garcia, and F. A. M. Trinta. USTO.RE: A Private Cloud Storage System. In *13th International Conference on Web Engineering (ICWE 2013) - Industry track*, Aalborg, 2013.

14. J. L. Edwards, M. A. Lane, and E. S. Nielsen. Interoperability of biodiversity databases: Biodiversity information on every desktop. *Science*, 289(5488):2312–2314, 2000.
15. L. Eisen and R. Eisen. Using Geographic Information Systems and Decision Support Systems for the Prediction, Prevention, and Control of Vector-Borne Diseases. *Annual Review of Entomology*, 56:41–61, 2011.
16. J. Elith and J. R. Leathwick. Species distribution models: Ecological explanation and prediction across space and time. *Annual Review of Ecology, Evolution, and Systematics*, 40:677–697, 2009.
17. A. Guisan and W. Thuiller. Predicting species distribution: offering more than simple habitat models. *Ecology Letters*, 8(9):993–1009, September 2005.
18. A. Guisan and N. E. Zimmermann. Predictive habitat distribution models in ecology. *Ecological Modelling*, 135(2-3):147 – 186, 2000.
19. R. J. Hijmans, S. E. Cameron, J. L. Parra, P. G. Jones, and A. Jarvis. Very high resolution interpolated climate surfaces for global land areas. *International Journal of Climatology*, 25(15):1965–1978, Dec. 2005.
20. S. Iacovella and B. Youngblood. *GeoServer Beginner's Guide*. Packt Publishing, 2013.
21. A. Jiménez-Valverde, A. Peterson, J. Soberón, J. Overton, P. Aragn, and J. Lobo. Use of niche models in invasive species risk assessments. *Biological Invasions*, 13(12):2785–2797, 2011.
22. D. Lezzi, R. Rafanell, E. Torres, R. De Giovanni, I. Blanquer, and R. M. Badia. Programming ecological niche modeling workflows in the cloud. In *Proceed. of the 27th IEEE Int. Conf. on Advanced Information Networking and Applications*, AINA-2013, 2013.
23. F. Marozzo, F. Lordan, R. Rafanell, D. Lezzi, D. Talia, and R. Badia. Enabling Cloud Interoperability with COMPSs. In C. Kaklamanis, T. Papatheodorou, and P. Spirakis, editors, *Euro-Par 2012 Parallel Processing SE - 4*, volume 7484 of *Lecture Notes in Computer Science*, pages 16–27. Springer Berlin Heidelberg, 2012.
24. G. Mendoza-González, M. L. Martínez, O. R. Rojas-Soto, G. Vázquez, and J. B. Gallego-Fernández. Ecological niche modeling of coastal dune plants and future potential distribution in response to climate change and sea level rise. *Global Change Biology*, pages n/a–n/a, 2013.
25. M. Muñoz, R. De Giovanni, M. Siqueira, T. Sutton, P. Brewer, R. Pereira, D. Canhos, and V. Canhos. openModeller: a generic approach to species potential distribution modelling. *Geoinformatica*, 15:111–135, 2001.
26. S. Nativi, B. Domenico, J. Caron, E. Davis, and L. Bigagli. Extending THREDDS middleware to serve OGC community. *Advances in Geosciences*, 8:57–62, 2006.
27. Open Source Geospatial Foundation. GeoNetwork Opensource. `http://geonetwork-opensource.org/`.
28. Y. Roskov, T. Kunze, L. Paglinawan, T. Orrell, D. Nicolson, A. Culham, N. Bailly, P. Kirk, T. Bourgoin, G. Baillargeon, F. Hernandez, and A. De Wever. Species 2000 & ITIS Catalogue of Life, March 2013. Digital resource at www.catalogueoflife.org/col/. Species 2000: Reading, UK.
29. S. D. Schoville, A. Bonin, O. François, S. Lobreaux, C. Melodelima, and S. Manel. Adaptive genetic variation on the landscape: Methods and cases. *Annual Review of Ecology, Evolution, and Systematics*, 43(1):23–43, 2012.
30. D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the Condor experience. *Concurrency and Computation: Practice and Experience*, 17(2-4):323–356, Feb. 2005.
31. A. Townsend Peterson, J. Soberón, R. G. Pearson, R. P. Anderson, E. Martínez-Meyer, M. Nakamura, and M. B. Araújo. *Ecological Niches and Geographic Distributions*. Princeton University Press, 2011.
32. J. Wieczorek, D. Bloom, R. Guralnick, S. Blum, M. Döring, R. De Giovanni, T. Robertson, and D. Vieglais. Darwin Core: An Evolving Community-Developed Biodiversity Data Standard. *PLoS ONE*, 7(1), 2012.

An unattended and fault-tolerant approach for the execution of distributed applications

Manuel Rodríguez-Pascual, Rafael Mayo-García**

CIEMAT
Avda Complutense 40. 28040 Madrid (Spain)

Abstract. In this work, the authors present a set of tools to overcome the problem of creating and executing distributed applications on dynamic environments. The objective is to provide a portable, unattended and fault-tolerant set of tools, encapsulating the infrastructure-dependent operations away from the application developers and users. By decoupling the definition of the remote tasks from its execution and control on different infrastructures, the development of distributed applications has been significantly simplified, while increasing their robustness and allowing running them on different computational platforms such as clusters and Grids with no effort. With this proposal, the development of distributed applications has been greatly simplified, allowing non-expert users to build them in a simple and intuitive way. These applications can be unattendedly executed either on local clusters or on Grid Infrastructure having the possible problems related to the distributed execution of tasks automatically managed by a devoted tool.

1 Introduction

Among the different distributed platforms, cluster and Grid infrastructures have emerged as powerful options to face new and more ambitious problems. Clusters are an outstanding solution for providing a significant amount of computational power by means of distributed resources, usually connected through a high speed local area network. If more resources are needed or the user hasnt got access to a cluster- Grid computing can be employed to access large computing infrastructures beyond the in-house facilities.

Some of the greatest research projects of the history of mankind, such as LHC, employ Grid infrastructures to perform their operations and data analysis. For this sake, several tools and manpower are offered to their scientists, hiding the complexity of executing such experiments. The rest of the scientific community can also benefit from the computational capabilities that Grid infrastructures offer. For them, having a reliable and unattended way of creating and executing tasks on distributed infrastructures is key for the adoption of this computational paradigm, as they lack the facilities employed by the large projects.

** e-mail of corresponding authors: {manuel.rodriguez,rafael.mayo}@ciemat.es

The implementation and execution of distributed applications that can run both on cluster and Grid infrastructures is however far from being trivial. Although the execution model of distributed applications on both platforms is identical, their particularities make the application requirements completely different. First, while the cluster is usually managed by someone of the same organization as the user, in the Grid the resources belong to different organizations, each one with different hardware, software, usage and security policies [7]. Also, while in local clusters the physical resources do not evolve in time, Grid infrastructures are highly dynamic. Unlike local clusters, which are considered as reliable, execution of tasks on the Grid is more conflictive, so the application must be designed to overcome these problems.

In order to simplify the creation of distributed applications -those that create a number tasks to be remotely executed, and then process the partial results to obtain the final one-, there exist different tools that perform some of the necessary work. This way the developer can focus on the application to create, isolating it from the underlying infrastructure. However, most of these tools are still partial focused either on Grids or clusters-, non-standard and usually too low-level, obliging the developer to take care of task management and control. This complicates the creation of portable distributed applications, making the process difficult and tedious. Moreover, it can act as an entry barrier, keeping interesting computational resources away from non-expert developers and users.

The work presented here aims to solve this problem, proposing a set of tools that enable a straightforward creation of highly portable distributed applications and their completely unattended and reliable execution on clusters and Grids. These tools are all encapsulated in the so-called DistributedToolbox, which conceptually consists on a job buffer with unattended management, high portability and strong error control.

DistributedToolbox is articulated around RemoteAPI, a very simple API designed to define the tasks to execute on the distributed infrastructure. Then, one of the devoted tools included on the toolbox takes care of the task execution. DistributedToolbox ensures a correct task completion, so the application developer is released from the low level operations of task management and control.

The fields of Grid scheduling, job control and creation of distributed applications have been widely studied and abundant literature exists on the area. Thus, an extensive analysis of the related work has then been conducted in section 2, helping to situate the performed work within its context, compare it with the existing alternatives and arguing the need for the proposed toolbox. Then, the rest of the work is organized as follows: section 3 depicts the content of DistributedToolbox, describing how it fits to the problem to solve and detailing the most significant characteristics of its implementation; in section 4 the feasibility of the proposed approach is tested, showing its behaviour on real production infrastructures; last, section 6 shows the conclusions and lessons learned during the development of this work.

2 Related Work

RemoteAPI is not the first API to describe the execution of tasks in distributed infrastructures: SAGA [16] and DRMAA [18] -just to mention a few-, are APIs devoted to define and control the execution of tasks to one or more resources over a distributed platform.

SAGA and GridRPC are designed for Grid infrastructures, while the proposed solution also embraces clusters. DRMAA has an implementation for clusters, but it relies on certain third party tools to execute the tasks -GridWay for the Grid, PBS or SGE on clusters- thus limiting the portability of the applications employing it. On the contrary, with RemoteAPI DistributedToolbox aims to provide a solution that does not depend on the underlying software, including drivers to execute tasks on different platforms. This way, the need for support of a new standard on the infrastructure is avoided, allowing the execution of tasks on with no software dependences and thus maximizing the possible resources to be used. And last but not least, the idea of this work is to release the application developer from the control of the task execution, delegating it on the provided tools. For that reason, RemoteAPI is much simpler than the rest of existing interfaces.

Meanwhile, the execution of tasks on Grid infrastructures is a widely studied field. There are two basic approaches to the problem: standard task execution and pilot jobs.

gLite WMS [2] and GANGA [3] provide the service responsible for distributing and managing tasks across computing and storage resources available on a Grid; GridWay [10] has arisen as a powerful alternative to WMS and counts with a significant user base, mainly because of its simplicity and superior performance [20]..

The main difference is that none of the aforementioned schedulers provide the error control level needed for a completely unattended execution of tasks, so it must be controlled inside the distributed application. On the contrary, GridController application a lightweight tool inside Distributedtoolbox- acts as an additional software layer over GridWay providing enhanced security and task execution control. Of course, GridWay could be replaced by WMS or any other Grid Scheduler that provides basic job submission and monitoring capabilities, with little effort.

The pilot job technique has been successfully applied [19, 14] for task execution on dynamic infrastructures. Although in this proposal GridController executes the tasks with GridWay, most of the performed work is completely compatible with a pilot job approach, so it could be seamlessly adapted.

How to achieve fault tolerance in distributed and Grid infrastructures is a well-known problem. Many different approaches have been taken to solve it [8], but it is still an open problem where an universal, perfect solution has not been achieved and probably never will.

Checkpointing for distributed tasks presents several challenges [1] but has been extensively employed o Grid Infrastructures [11, 17, 9] to increase the reliability and fault tolerance of long-lasting tasks. As a significant drawback,

the implementation of a checkpointing mechanism on an application induces significant overheads and is not always possible, depending on the nature and design of the application.

Replication has also been widely employed to achieve fault tolerance while reducing the execution walltime [4, 13]. As a drawback, the efficiency of this technique is highly dependent on the kind of application being executed, and it induces an overhead which increases lineally with the proportion of replicas. Moreover, although it provides certain degree of error control, a pure replication mechanism with no further control cannot ensure that the desired tasks will correctly be executed under any possible condition.

In this context, DistributedToolbox aims to represent a powerful and easy to use alternative to the up to the authors knowledge- partial or complex existing tools. The next section will detail the main technical aspects of the current proposal, demonstrating how to achieve the desired portability, robustness and implementation simplicity of distributed applications.

3 DistributedToolbox

The proposed solution for an unattended, reliable execution of distributed applications is a toolbox called DistributedToolbox. It is composed by an API named RemoteAPI and a set of related applications.

The objective of RemoteAPI is to serve as a simple way of specifying the characteristics of the tasks to be executed on the distributed infrastructure. The developer of a distributed application can specify the requirements of the remote tasks, and a devoted tool will then carry out that execution. This toolbox includes several of these tools, aiming to serve on different distributed infrastructures.

A complete description of RemoteAPI together with the rest of the toolbox is presented in the following section.

3.1 A hint on the execution of distributed applications

The execution flow of a distributed application is considered to follow this basic scheme:

- In the *preprocess* step the application is initialized, some of the work is performed on the local resource, and the input data for the distributed part of the application is prepared.
- In the *remote execution* step the application employs a distributed infrastructure cluster or Grid- to execute a set of auxiliary applications, remote tasks, with their own input and output data.
- At last, in the *postprocess* step the output data from the auxiliary applications is processed by the main application on the local resource, some more work is performed and the application ends.

The followed approach employs two different executables: the first one performs the preprocess step; then, the remote execution is carried out by DistributedToolbox; and at last, a second executable performs the postprocess step.

This way, DistributedToolbox can face all the problems related to the Grid executions of tasks and isolate them from the application developer. Moreover, the employment of a centralized devoted tool for the execution of tasks on the Grid occasions that any improvement will be immediately available for all the users.

3.2 DistributedToolbox Architecture

DistributedToolbox is the approach presented in this work towards an efficient and robust implementation of the distributed paradigm as described in the previous section. It includes a set of tools to specify the remote tasks and execute them on different distributed platforms. Depending on the infrastructure this execution is very different:

- ClusterController performs the execution of tasks on clusters. A DRMAA driver is employed for clusters based on PBS or SLURM among others.
- If the cluster does not support DRMAA, a bash-based driver can be employed.
- For Grid infrastructures, DistributedToolbox includes a newly created control and monitoring tool called GridController.
- And if a tool with a different behaviour is needed, the design of DistributedToolbox makes its implementation straightforward.

DistributedToolbox employs a database to store the status of the tasks to execute. Following a producer-consumer model, TaskLoader populates the database with the information of the new tasks to execute, and ClusterController or GridController execute those tasks. Finally, ExecutionAnalyzer extracts information about the execution from this database.

DistributedToolbox is mainly implemented in Python, with the exception of shell script for both the Grid and Cluster drivers. The communication bewteen its applications and the database manager is performed by SQLAlchemy library, which provides out-of-the-box controllers to the most used database management systems (Oracle, MySQL, SQLite...). This interface keeps DistributedToolbox portable, avoiding dependence of third-party software.

After the database has been populated with the tasks to execute, the user can run any of the task execution tools depending on the desired platforms: ClusterController for clusters or GridController for the Grid. At last, when the desired tasks have been executed, ExecutionAnalyzer performs a basic performance analysis. This analysis is presented to the user both as a text and graphically, making use of matplotlib library.

3.3 RemoteAPI

RemoteAPI is an extremely simple API, employed to specify the requirements of the tasks composing a distributed application that will be executed on a distributed infrastructure. The basic idea is not to create yet another job API aiming to be implemented by the maximum possible number of vendors, LMRS and Grid schedulers, but to provide an extremely simple tool that can be employed on existing or new projects and execute them on existing infrastructures. Due to its simplicity, a non-expert individual user can easily create distributed applications, just defining the tasks to distribute and relying on DistributedToolbox to carry on the execution.

The components of a task that can be specified include the application name, input and output files, location, executable and arguments. This API also establishes how this information is processed and transferred between applications: the description of each XML task is stored in an independent XML file, with an index containing all the filenames related to a given instance of the application.

This way, the specification of remote tasks is completely decoupled from their execution. The application developer must only decide what he wants to execute remotely, and different tools will then be employed depending on the underlying architecture.

RemoteAPI has been designed having an easy portability and implementation on different languages capability in mind. At this moment Java and Python versions are included in DistributedToolbox, but others can be created in very little time. Many of the most used languages (C/C++, Python, Perl, FORTRAN, Bash...) include XML parsers or have an Open Source available one, thus reducing the implementation of RemoteAPI to basic file management.

At last, it is worth mentioning that with RemoteAPI being a subset of DRMAA, SAGA and GANGA, any legacy application employing one of those APIs can be ported to employ DistributedToolbox in no time. The only limitation here is the lack of support for MPI or other kind of parallel tasks on RemoteAPI, currently limited to serial ones.

3.4 Task execution on the Grid: GridController

In order to perform a reliable, unattended execution of tasks on Grid infrastructures, the newly created application GridController -an additional software layer over GridWay metascheduler- has been included on DistributedToolbox.

Due to the particularities of Grid infrastructures dynamicity, heterogeneity, probability of failures [7]- ensuring a correct execution of tasks is far from being trivial. This is specially challenging when being performed on a completely unattended way, as the errors and execution issues can come from different sources and it is not always straightforward to automatically identify and correct them.

The main characteristic of GridController is its robustness. It has been developed with the objective of releasing the user from the control of the remote execution, so it must detect any problem along the execution of the task and, if possible, correct it.

This tolerance to failures is achieved with two error control tools: the one included in GridWay metascheduler, and an additional module included in GridController covering a wider set of error sources and verification mechanisms.

Currently, the following failures sources have been detected and solved. Note that GridController is responsible of all the ones that are not explicitly managed by GridWay.

- Certification problems.
  - If the user is not able to properly identify himself by employing a valid Grid certificate, GridWay will detect it.
- Local resource failures.
  - If the specified input files are not present on the system, the job is considered as finished.
  - If communication with GridWay is broken the task submission is stopped. When communication is restored, the execution of the tasks being run is recovered.
  - If GridController fails, no information is lost due the employment of databases for persistence..
  - If the database fails, the execution of GridController is considered to be unsafe and is automatically stopped.
- Remote resource failures.
  - If a task does not start its execution, GridWay detects it.
  - If a task is queued on the remote site for more than a given threshold, GridWay cancels it.
  - If there is any problem with the Grid certificates on the remote site and the task cannot be executed, the problem is detected by GridWay.
  - Some failures in remote clusters lead to a state where the master node thinks that the task is running forever. To detect this, tasks with an extremely long execution time are considered to have failed.
  - In order to avoid performance slowdowns, a small replication factor for every group of tasks has been included.
- Communication failures.
  - If any kind of problem on the transmission of the input data or task executable occurs, it is detected by GridWay on the remote site and the task is cancelled.
  - If the output data is not returned to the local host, the task is considered to have failed.

And although no more failure sources have been located, the architecture of GridController would make their incorporation to the application and ulterior management straightforward.

A cancelled or failed task is always executed again until a correct execution is achieved, ensuring the existence of the specified output files.

Grid infrastructures are assumed to be highly dynamic and the remote sites are not expected to be fault-free. However, if tasks submitted to a certain site fail on a regular basis, that site is banned either by GridWay or by GridController-

for a certain amount of time, avoiding the employment of malfunctioning resources. The banning time is exponentially increased with the number of failures and reset when a task is correctly executed on the site, assuming that any existing problem has been solved. That way GridController submits the tasks where a higher probability of success exists, while avoiding banning functional sites for an isolated problem.

3.5 Task execution on local clusters: ClusterController

The execution of serial tasks on local clusters is usually straightforward: the failure rate on this kind of infrastructures is negligible, there are not problems with data transmission, authorization issues or any of the Grid-related problems depicted on the previous section. Due to this simplicity, the execution of remote tasks is narrowed to read the task requirements specified with RemoteAPI and execute them.

DistributedToolbox includes ClusterController, that can be regarded as a simplification of GridController: its architecture and functionality is identical, although it does not support fault tolerance or error control. Due to the employment of DRMAA and SQLAlchemy it can be seamlessly executed on PBS, SGE and SLURM with a lot of different database management systems, thus covering the most usual Local Resource Management Systems. And if DRMAA is not supported, an included CLI-based driver can be employed to manage the task submission and control. This represents an improvement over DRMAA-based applications, as they can be now executed in a wider set of resources.

At this moment, DistributedToolbox at the moment does not support any advanced technique of task execution on clusters such as live migration or checkpointing. Also, it is designed for distributed applications that require the execution of serial tasks, not providing any MPI support. The advantage of employing this toolbox over other alternatives to create distributed applications for clusters comes from the high portability of the solution: the same application can be executed on Grids an Clusters with no porting effort. Besides that, there is no performance gain on the employment of Cluster Controller.

3.6 An execution example

In order to show how DistributedController can be employed to create distributed multi-platform applications, this section will show how it can be used in existing applications. This example will show how jModelTest2 [6] was adapted to be executed on clusters and Grids, creating a robust and reliable application.

First, the preprocess section was modified in order to define the tasks to distributed and export them into an XML file. In this case, every task was stored on the ModelTest array, with some information on the ModelTest data structure.

```
LinkedList<GridTask> gridTasksList = new LinkedList<GridTask>();
GridTask myGridTask = new GridTaskImpl();
```

```
for (int i=0; i<ModelTest.numModels; i++){
        myGridTask = new GridTaskImpl();
        myGridTask.setWorkingDirectory(jModelTestPath);
        myGridTask.setExecutable("executable.sh");
        myGridTask.setTaskName("job_template_"+i);
        myGridTask.setArguments(ModelTest[i].arguments);
        myGridTask.setInputFiles(ModelTest[i].input_files);
        myGridTask.setOutputFiles(new String[]{output_stats + i + .txt});
        gridTasksList.add(myGridTask);
}
gridTaskList.exportGridTasks(storagePath);
```

With this incorporation, the preprocess section is ready to be executed. In order to do so, the final user must first execute jModelTest2 :

```
$jModelTest2 <input parameters> -preprocess
        -distributedTasksLocation=<storagePath>
```

Where storagePath is the place where the XMLs defining the tasks will be stored. Then, the information can be loaded into the database with:

```
$python TaskLoader.py <storagePath>
```

After the tasks have been inserted on the database, they can be executed on clusters or Grid infrastructures. Those operations can be seamlessly performed by the final user with on of the following commands:

```
$python ClusterController.py
$python GridController.py
```

When the distributed tasks have been executed, jModelTest2 must perform the postprocess of the partial output files. In order to locate these output files and the rest of information, the application developer must employ RemoteAPI to load that information on the application:

```
GridTask myGridTask = new GridTaskImpl();
gridTasksList = myGridTask.loadGridTasks(storagePath);
```

After this, gridTaskList will contain the location of every distributed task, allowing the application to process it. This way, the final user only needs to execute the preprocess section indicating the location of the distributed tasks definition files, as done in the preprocess section.

```
$jModelTest2 <input parameters> -postprocess
        -distributedTasksLocation=<storagePath>
```

As can be seen, the creation of a distributed application and its execution on different platforms with the employment of DistributedToolbox is extremely easy and intuitive, lowering the entry point to non-expert users and allowing them to take the most of distributed infrastructures with little time and effort.

4 Results

4.1 Testbed

In order to demonstrate the feasibility of the performed work, DistributedToolbox was tested under different conditions. ClusterController was tested on a cluster belonging to CIEMAT facilities, and in order to test GridController, a local host and a Grid Infrastructure were necessary.

ClusterController was developed and tested at the CIEMAT cluster Euler. Euler is a low latency cluster managed by PBS/Torque. When this experiment was carried out it could be considered a state-of-the-art high performance computer (1196 cores, 32GB RAM/core, Infiniband connection...).

The local resource where GridController was executed consisted on a virtual machine running Debian 4.0, with GridWay 5.4.0 and Python 2.4.3. Except during the tests regarding failures on the local host, this resource was in production status.

The biomed Virtual Organization deployed by the EGI Project was used to test the behaviour of GridController in a production environment. When this work was carried out (Oct-Dec 2012) it counted on more than 140 sites from about 50 different institutions, offering more than 320,000 CPUS.

In order to test the feasibility of this approach under real world conditions, two different applications belonging to the life sciences area were employed, jMotelTest2 [6] and ProtTest3 [5]. They create 88 and 120 respectively- tasks to be remotely executed and up to 100 instances were executed at the same time, representing a significant workload to be executed by DistributedToolbox without a single one failing.

4.2 ClusterController

Euler employs PBS/Torque for job submission and management, thus allowing DRMAA controlled executions. This offered the possibility of developing, debugging and testing the proposed DRMAA Driver for ClusterController on a production environment. Given that DRMAA is a standard API, the correct behaviour of this driver can be then ensured for any DRMAA-based DRM. An updated list of the DRMs supporting this API can be found at the DRMAAs project home page (http://www.drmaa.org/implementations.php). When this work was being carried out it comprised Grid Engine, PBS/Torque, IBM Tivoli LoadLeveler and SLURM.

As PBS/Torque jobs can be submitted and managed via command line, the CLI-based PBS driver was also tested on this cluster. The idea behind this driver is to serve as a template in case that a new one for a different DRM needs to be developed. As the needed commands submit and monitor the status of a job- are simple and available on every DRM, the adaptation of this driver can be seamlessly performed.

Given that ClusterController does not perform any scheduling or task management, it provides no performance gain. On the other hand, being a lightweight

application it does not add any significant overhead to the task execution. The advantage obtained with it comes from the possibility of executing distributed applications based on RemoteAPI, so they can be either executed on clusters on the Grid with no porting effort.

4.3 GridController

In order to test the robustness of GridController under any condition, different experiments focused on the possible failure sources were performed.

The first one consisted on the testing of failure detection on the local resource. Table 1 reflects the result of this experiment, comparing it with GridWay and WMS.

Table 1. Problem detection and recovery with GridWay, WMS and DistributedToolbox

Module	Failure	GridController	GridWay	WMS
Certification	User certificate not present	Yes	Yes	Yes
	Submission to a different VO	Yes	Yes	Yes
File management	Input files not present	Yes	No	Yes
	Output files not present	Yes	No	No
Local software	Scheduler failure	Yes	Yes	N/A
	Database manager failure	Yes	Yes	N/A
	Shutdown of local host	Yes	Yes	Yes
Remote software	Application exit status not zero	Yes	Yes	No

Additionally, there is the problem of the job not correctly finishing on a given site for some reason. This problem is difficult to quantify, as it strongly depends on the application being run and the remote site. Common error sources are library dependences, memory issues, too long execution time, lack of needed software, version issues so in order to face all of these problems, a massive number of tasks were submitted to the Grid.

In order to verify the correctness of the proposed solution, the followed approach consisted then on the execution of a massive number of tasks, hoping that any possible failure happens at some moment. During these tests, which were performed from October to December 2012, a total of 84,217 tasks where submitted. 71,186 out of them presented some kind of issue along their execution, and where automatically resubmitted until they were able to finish correctly. Such a wide number of performed experiments demonstrates that GridController accomplishes the objective it was designed to: execute serial tasks on the Grid on a reliable and unattended way.

Official WMS documentation claims a 0.3% failure rate due to problems with the middleware [2]. Other sources [15] reports a 30% failure on the execution of remote tasks, which falls to about 2% after three retries. Some experiments performed by the authors in the past, see for example [12], showed a 12.7% failure

rate with a single retry and 1,3% with three, which is consistent with the official information. However, exit codes different from zero are not considered failures by themselves and tasks are not resubmitted, although the user is informed of this circumstance and can manually resubmit the problematic task.

In the case of GridWay, it is also capable of detecting most of these failures and resubmitting the tasks. It monitors the exit status of the applications being remotely executed, so a well-designed one which returns a useful exit code in case of failure- will be correctly managed. If contact with the remote wrapper is broken meaning that it has been cancelled or a network problem exists- the task will be considered as failed. GridWay incorporates a site banning mechanism that prevents the submission of tasks to problematic hosts, so the number of errors is usually low. However, some subtle failures -such a problem with the remote DMRs that incorrectly informs of a task running forever- are not detected. Error proportion on executions of jModelTest and ProtTest has consistently remained between 0.5% and 1%.

GridController replicates the slowest tasks on any execution submitting them to a different remote site, ensuring that problems with lost tasks or performance slowdowns are avoided. Also, tasks with an extremely long execution time are considered to have failed and replicated too. It does not however cancel them, so if at any moment they correctly finish the execution, the output data is recovered and the replica immediately cancelled to avoid causing any infrastructure overhead.

It is important to regard that, while the proportion of successfully executed tasks both with GridWay and WMS is about 99% in both cases, this number is not enough to accomplish the objectives of this work. As any fault rate greater than zero obliges the developer of distributed applications to implement error control mechanisms with the negative consequences that have previously been detailed-, the existence of GridController is fully justified.

4.4 overhead

In terms of storage, DistributedToolbox has a negligible influence on the system. When executing a distributed application, it creates a folder with a small XML file defining every task to be remotely executed. With the codes employed on this analysis this represented less than 500 KB per instance.

In order to store the information regarding the remote tasks into the database so it can be read by GridController or ClusterController-, TaskLoader has to read all the XMLs, process them and access the database to store the information. In the performed tests, the process took about 5 seconds per instance. This overhead depends lineally on the number of instances and represents only a small fraction of the total walltime so it can be considered negligible too.

And at last, the execution of ClusterController and GridController does add any overhead, given that they are both very fast and lightweight applications. In the case that there are no failures on the remote execution of tasks, GridController has only worked as a task creator for GridWay, so it hasnt created

any overhead. And if a failure does happen, its detection is immediate, so not overhead is added neither.

5 Conclusions and future work

DistributedToolbox greatly simplifies the creation and execution of tasks on distributed environments. Its design allows the creation of new applications, as well as the porting of existing ones based on DRMAA or SAGA. Due to its efficiency, robustness and simplicity, its usage represents a step forward toward on the creation of portable and distributed applications. With it, developers are provided with a set of high-level tools to define and execute the distributed tasks, making the creation of distributed applications fast and easy.

DistributedToolbox is provided with a GPL license. Together with usage documentation, it can be found and freely downloaded from CIEMAT-TIC home page (http://www.ciemat.es/portal.do?IDR=343&TR=C). The development team is completely open to collaborations and contributions on this matter.

The future of the project is oriented on two different ways, functionality and interface development. First, integration with different scheduling tools and computational platforms will be developed: a controller for WMS-based User Interfaces will be created employing SAGA API, and the integration with GANGA will be analysed. The implementation of RemoteAPI on different programming languages will be performed depending on the needs of the development team: due to the simplicity of the process, it will only be performed when required for an application being adapted. Also, to monitorthe status of the remote tasks and displaythe information provided by ExecutionAnalyzer on a friendly way, a web interface will be created.

Acknowledgements

Mariusz Mamonski, from PSNC, provided really helpful backup on the development of DRMAA applications for PBS clusters. Antonio Juan Rubio-Montero, from CIEMAT, was a great support on the creation of GridController.

The results provided in this work have made use of EGI-InSPIRE project infrastructure (http://www.egi.eu/) and have count on the support from COST Action BETTY (IC1201).

References

1. L. Alvisi, E. Elnozahy, S. Rao, S. A. Husain, and A. de Mel. *Fault-Tolerant Computing, 1999. Digest of Papers. Twenty-Ninth Annual International Symposium on*, pages 242–249, 1999.
2. P. Andreetto et al. *Journal of Physics: Conference Series*, 119(6):062007, 2008.
3. F. Brochu et al.
CoRR, abs/0902.2685, 2009.

4. S. Camarasu-Pop et al. *Future Generation Computer Systems*, 29(3):728–738, Mar. 2013.
5. D. Darriba, G. L. Taboada, R. Doallo, and D. Posada. *Bioinformatics*, 2011.
6. D. Darriba, G. L. Taboada, R. Doallo, and D. Posada. *Nature Methods*, 9(8):772–772, July 2012.
7. I. Foster. *Grid Today*, 1, 2002.
8. R. Garg and A. Kumar Singh. *International Journal of Computer Science & Engineering Survey*, 2(1):88–97, Feb. 2011.
9. P. Gupta. *IJCSMS International Journal of Computer Science & Management Studies*, 11(1):64–69, May 2011.
10. E. Huedo, R. S. Montero, and I. Martín Llorente. *Journal of Systems Architecture*, 52(12):727–736, 2006.
11. G. Jankowski, R. Januszewski, R. Mikolajczak, and J. Kovacs. Technical report, 2008.
12. M. Rodríguez-Pascual et al. In *Proceedings of the 2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*, pages 380–384, Washington, DC, USA, 2011. IEEE Computer Society.
13. M. Rodríguez-Pascual, I. Martín Llorente, and R. Mayo-García. *Computing and Informatics*, 32:113–144, Jan. 2013.
14. A. Rubio Montero et al. *Future Generation of Computer Systemsl*, submitted.
15. A. Sciaba and S. Campana. In *EGEE'06*, pages 1–11, Geneve, Switzerland, Sept. 2006.
16. C. Smith and P. Computing. *OGF Report*, 2008.
17. A. L. S. Therasa, G. Sumathi, and A. S. Dalya. *International Journal of Computer Applications*, 2:95–99, May 2010.
18. P. Tröger and P. Domagalski. *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2007)*, pages 619–626, May 2007.
19. A. Tsaregorodtsev et al. *Journal of Physics: Conference Series*, 119:062048, 2008.
20. J. L. Vazquez-Poletti, E. Huedo, R. S. Montero, and I. Martín Llorente. *Multiagent and Grid Systems*, 3(4):429–439, 2007.

Calculation of Two-Point Angular Correlation Function: Implementations on Many-Core and Multicore Processors

Miguel Cárdenas-Montes**[1], Miguel A. Vega-Rodríguez[2], Jan Westerholm[3], Rafael Ponce[1], Evren Yurtesen[3], Ignacio Sevilla[1], Eusebio Sánchez Alvaro[1], Juan José Rodríguez-Vázquez[1], Mats Aspnas[3], Ville Timonen[4], Nicanor Colino[1], and Antonio Gómez-Iglesias[5]

[1] CIEMAT, Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas, Departamento de Investigación Básica
Avda. Complutense 40, 28040, Madrid, Spain.
[2] ARCO Research Group, Dept. Technologies of Computers and Communications, University of Extremadura, Escuela Politécnica, Campus Universitario s/n, 10003, Cáceres, Spain.
[3] Åbo Akademi University, Joukahainengatan 3-5, FI-20520 Åbo, Finland.
[4] Turku Centre for Computer Science, and Åbo Akademi University, Joukahainengatan 3-5, FI-20520 Åbo, Finland.
[5] CSIRO Mathematical and Information Sciences, Private Bag 33, South Clayton, VIC 3169, Australia

Abstract. The appearance of many-core GPUs and multi-core CPUs have endowed to the computational researchers with powerful platforms to analyse huge amounts of data. The performance of implementations resides on the efficient exploitation of the underlying hardware features. This implies that optimization processes are often necessary after a reference implementation is built. In the present work, this kind of fine-grained optimization is performed over a code aiming to analyse the angular distribution of galaxies in the Universe. Concerning the problem, its relevance can be emphasised by the fact that, by characterizing this distribution, the budget between ordinary matter, dark matter and dark energy in the Universe can be estimated. The Two-Point Angular Correlation Function is a computationally intensive function able to measure this distribution. Today the primary bottleneck is computational. The available data amount will increase even further in the forthcoming years. The performance improvements in the Two-Point Angular Correlation Function code will have substantial effect in scientific HPC applications used by cosmologists. As a consequence of this work, an assessment is stated about the techniques which produce the most relevant acceleration in the execution of the code, as well as the performance gap between the many-core and multicore implementations.

** e-mail of corresponding author: miguel.cardenas@ciemat.es

1 Introduction

The support of GPUs for computational purposes advanced immensely in the recent years. The highly parallel hardware of GPUs is not restricted to graphics programming anymore. New programming paradigms and language extensions are introduced for effectively utilizing the GPU hardware for computational purposes. Most often used extensions for GPGPU programming are currently CUDA and OpenCL. Meanwhile, OpenCL is also suitable for operations on multi-core CPUs and many other devices supporting OpenCL (such as FPGAs or ARM GPUs).

The obtainable performance improvement is highly dependant on the level of parallelism achievable within the code. The performance improvements gained by using GPUs are often governed by Amdahl.s law. Often existing codes must be redesigned for being able to run parallel and exploit hardware features efficiently on GPGPU systems. Certain algorithms and codes are natively suitable for direct porting to GPGPU code and may achieve substantial performance improvements when ported. However, often it is possible to achieve even further performance improvements by fine tuning the code to fit better to the underlying technology. Even a small percent of additional performance improvement can translate into huge time and energy savings in computationally time consuming algorithms.

In this work, the reference CPU code for Two-Point Angular Correlation Function (TPACF) is optimized by taking advantage of the SSE and AVX instructions for utilizing vector operations. In addition, optimized GPGPU versions which exploit the massive parallelism were created.

Recent progresses in observational cosmology have led to the development of the ΛCDM[6] model [1]. It describes a large amount of independent observations with a reduced number of free parameters. However, the model predicts that the energy density of the universe is dominated by two unknown and mysterious components: the dark matter and the dark energy. These two components constitute the 96% of the total matter-energy density of the universe.

Dark energy and dark matter have never been directly observed, and their nature remains unknown. Understanding the nature of the dark matter and the dark energy is one of the most important challenges of the current cosmology and particle physics[7].

The distribution of galaxies in the Universe is one of the main probe of the ΛCDM cosmological model. The most important observable to study the statistical property of this distribution is the Two-Point Angular Correlation Function (TPACF), which is a measure of the excess probability, relative to a random distribution, of finding two galaxies separated by a given angular distance. By comparing different correlation functions, implicit comparisons between cosmological models are made.

[6] Lambda Cold Dark Matter.

[7] The quantification of the budget between ordinary and dark components in the Universe is a major issue as proven by the recognition of the Science magazine in 1998 and 2003 as "Scientific Breakthrough of the Year".

The TPACF is a computationally intensive function with $O(N^2)$ complexity, being N the number of galaxies. In the present work, a sample from cosmological simulations with $4.3 \cdot 10^5$ galaxies has been used. Today, having a very efficient and high performance TPACF implementation is very important due to the CPU time consuming nature of TPACF calculations. Achieving highest possible performance will be even more important in future. This is due to the fact that the astronomical surveys, Dark Energy Survey, the Kilo-Degree Survey or Euclid, which are expected in the forthcoming years will increase the observed number of galaxies from hundreds of thousands, up to hundreds of million galaxies.

This paper is organized as follows: Section 2 summarizes the Related Work and previous efforts done. The specifications of CPUs and GPUs used in this work are presented in Section 3.1. A brief explanation of the TPACF is presented in Section 3.2. The implementation of the TPACF on GPU is described in Section 3.3; whereas the description of the vectorization and parallelization processes of CPU implementation is presented in Sections 3.6 and 3.7. Results for CPU implementation is presented and analysed in Section 4; whereas for GPU implementations are presented in Section 5. Finally, the Conclusions and the Future Work are presented in Section 6.

2 Related Work

The previous efforts done in the acceleration of the analysis of the distribution of galaxies can be classified in two categories. On the one hand, it can be mentioned the implementations of the TPACF problem into more powerful computing platforms: FPGA [2], GPU [3] and, the most recent one comparing three implementations: GPU, MPI and OpenMP [4]. And, on the other hand, it can be cited the use of some tricky mechanism to reduce the complexity of the calculation without losing too much accuracy —i.e. k-trees [5] or pixelization [6]—. The present work can be included in the first category; although no examples of fine-grained optimization process for the TPACF has been published.

The core of the analysis in this work is to implement more advanced strategies to reduce the overall execution time. The very preliminary results of this strategy was presented at [4]. In this work, the results of the initial CPU implementation were compared with a multi-core OpenMP implementation and a GPU implementation.

In spite of these efforts, the new surveys will largely increase the volume of data, and as a consequence, it makes necessary new developments and improvements. This motivates the present work, to evaluate the techniques in order to produce an extra reduction of the execution time, and therefore, to attain a higher efficiency.

To the authors knowledge, up to now, exhaustive performance analysis has not been addressed in the past.

3 Methods and Materials

3.1 Hardware Specification

The hardware used for benchmarking is as follows. The CPUs used are Intel i7-3770K Ivy Bridge and AMD FX-8150 Bulldozer. The GPGPU hardware used for benchmarking are Tesla M2050 Fermi. In addition, AMD Radeon 5870 Cypress and AMD Radeon 7970 Tahiti GPGPU hardware are used. On the other hand, the initial optimization processes and tests were performed on a pre-Fermi card: GTX295, and Fermi C2075. The public code has been optimized on these cards. Finally, a Xeon Phi 5110O card has been also employed.

3.2 TPACF

The TPACF, $\omega(\theta)$, is a measure of the excess or lack of probability of finding a pair of galaxies under a certain angle with respect to a random distribution. In general, estimators TPACF are built combining the following quantities:

- DD(θ) is the number of pairs of galaxies for a given angle θ chosen from the data catalogue D.
- RR(θ) is the number of pairs of galaxies for a given angle θ chosen from the random catalogue R.
- DR(θ) is the number of pairs of galaxies for a given angle θ taking one galaxy from the data catalogue D and another from the random catalogue R.

Although diverse estimators for TPACF do exist, the estimator proposed by Landy and Szalay [7], (Eq. 1), is the most widely used by cosmologists due to its minimum variance.

$$\omega(\theta) = 1 \; + \; (\tfrac{N_{random}}{N_{real}})^2 \cdot \tfrac{DD(\theta)}{RR(\theta)} \; - \; 2 \cdot (\tfrac{N_{random}}{N_{real}}) \cdot \tfrac{DR(\theta)}{RR(\theta)} \qquad (1)$$

In Eq. 1, N_{real} and N_{random} are the number of galaxies in the data and random catalogues.

A positive value of $\omega(\theta)$ —estimator of TPACF— will indicate that galaxies are more frequently found at angular separation of θ than expected for a randomly distributed set of galaxies. On the contrary, when $\omega(\theta)$ is negative, a lack of galaxies in this particular θ is found. Consequently $\omega(\theta) = 0$ means that the distribution of galaxies is purely random.

The calculation of TPACF implies to compute the angle between all pairs in a sample of N galaxies. As a consequence, the complexity of the calculation is as $O(N^2)$.

3.3 CUDA Implementation

In this section, the CUDA baseline implementation of the problem is described, specially focussing on the most relevant aspect for the final performance and the optimization process.

Algorithm 1: Schema of the kernel calculating the correlation between the angles subtended by the galaxies. Algorithm pseudocode.

```
Define shared variables for angle, bin and intermediate shared-built histogram ;
foreach Pairs of galaxies do
    Calculate the dot product;
    if Data malformed then
        Reject data;
    Calculate arc-cosine to get the angle;
    Calculate the bin corresponding for this angle in the shared-built histogram;
    if Bin in the area to make the histogram then
        Increment to the appropriate bin;
Pile up the shared-built histograms into a global-memory built histogram;
```

Before crumbling this, the schema of the pseudocode of the algorithm is presented at Algorithm 1. This is common to all the later implementations.

First of all, the initial strategy for the GPU code paid special attention to the use of *shared memory*. For this reason, the dot product and the arc-cosine calculation for each pair of galaxies are implemented in this type of memory. This avoids the used of the *global memory* —much slower than *shared memory*— for any intermediate calculation which requires frequent read and write processes.

Other expected bottleneck is the construction of the histograms: $\mathrm{DD}(\theta)$, $\mathrm{RR}(\theta)$ and $\mathrm{DR}(\theta)$. Following the sequence of the commands in the kernel, until this point a multithread calculation has operated over the pairs of galaxies, calculating the dot product, next the arc-cosine and, finally, the bin in the histogram where one count ought to be incremented. But, due to the multithreaded nature of the kernel, simultaneous updates of the same bin in the histogram must be avoided in order to do not miss any count. This forces to use atomic functions to create the histogram or alternatively water-fall-if-elseif structures [3].

The use of water-fall-if-elseif structures implies strong drawbacks. It is less flexible to changes in the range of angles of the histogram, and forces to recompile after any modification. With implementing atomic functions, the range of angles for the histogram can be introduced during the invocation.

Besides, the water-fall-if-elseif structures produce less compact kernels — higher number of lines—, thus making more difficult the readiness of the code. By using atomic functions instead water-fall-if-elseif structures reduces significantly the number of lines of the code, so it eases its readiness and its maintenance.

The atomic functions for integers are supported by NVIDIA GPU for computing capability 1.1 and higher in *global memory*; and for computing capability 1.2 or higher in *shared memory* [8]. In our case, the appropriate function is *atomicAdd()*.

The use of atomic functions in *global memory* causes a major performance degradation. Therefore, the solution is to perform more atomic operations in parallel. For this reason, an alternative strategy has been followed. Each block of

threads implements a histogram in *shared memory*; and in these histograms, angles are binned in parallel. Next, all shared-memory-built-histograms are reduced to a single one in *global memory*. This strategy produces a parallel treatment of the most critical operation, being the foundation of the success of the original GPU implementation.

The maximum histogram size is related to the *shared memory* capacity. Larger capacity allows hold larger histogram. Fortunately, practitioners request humble figures in the histogram size: from 16 to 256 bins, which allow cover both: large range of angles with low number of bins per angle, and a high detail implementation low number of angles with large number of bins per angle.

On the other hand, the baseline code implements a coalesced access pattern to the global memory. This is achieved by disposing the x-coordinates of all galaxies in a single array, and similarly for y and z-coordinates. By implementing this data layout, adjacent threads in a block request contiguous data from global memory. Coalesced access maximizes global memory bandwidth usage by reducing the number of bus transactions.

Regarding the constrains, the histogram size is fixed at 64 degrees with 4 bins per degree. This size is selected as a mean of he recommendations of the final users. Slight variations on the histogram size or the data catalogues modify severely the overall performance of the algorithm, preventing the comparison with other works. Indirectly this constrain eliminates the choice to optimize the code by reshaping the threadblock size (aiming to maximize the occupancy).

It exists many GPU applications which use histograms for diverse uses: data mining, image processing, etc. Since one of the bottleneck of the histogram creation arise from bin collisions, two or more threads try to update the same bin, in image processing ad-hoc rearranging data image can mitigate them [9]. This data shuffling has a non-negligible impact over the performance due that it remove the serialization of the atomic operations over the same bin.

However in our case, it is well-known that in the DD catalogue most of the galaxies are subtended small angles, which produce collisions over the lowest bins. Unfortunately, the lack of a priory knowledge on which galaxies subtend the same angle impedes the use of this technique.

3.4 OpenCL Implementation

The GPGPU OpenCL code was optimized by using LDS(local) memory for improved performance on atomic operations. AMD GCN architecture provides up to 2 TB/s LDS bandwidth. Several OpenCL kernels with different float4 and float8 vector data types for hinting the compiler to use vector operations on both CPU and GPU was used. Using the vector data types ensure usage of vector operations and not rely on the compiler for auto-vectorization. Hinting vector elements often give better results in cases where the compiler is not able to auto-vectorize.

Each workgroup used local bins to sum up results using atomic_add() operations. The local bins used unsigned int data type. This size was chosen for reducing the occupied local memory size and increasing efficiency. The results

were then summed to global memory using a global lock on global memory items when all the work-items within a group finish calculation. The bins in global memory used unsigned long int data type for avoiding overflowing.

An additional optimization was combining all DD, DR and RR catalogue operations within a same loop for re-using elements instead of re-loading them from the memory. One disadvantage of this method is it requiring the real and random galaxy databases to include same number of galaxies.

The versions with vector elements also employed a custom acos() function to avoid serialization with OpenCL SDKs which do not have in-built vector acos() function. In addition, acos() function was further simplified to only cover the quadrant involved in the calculations.

The OpenCL implementations performed on par or better compared to CUDA implementations on NVIDIA hardware.

3.5 OpenCL Xeon Phi Implementation

The OpenCL kernels used in GPGPU cases were ran on Xeon Phi accelerators directly. The best results were accomplished with auto-vectorization by the compiler. The maximum hinted vector type in OpenCL test cases was float8 while Xeon Phi devices support float16. However code with float16 elements were not available at the time of this writing. Therefore, it was expected that auto-vectorization will provide more efficient code.

It may be possible to improve on the Xeon Phi implementation however current drivers for Xeon Phi is found to be unreliable. Therefore it was not possible to make additional tests in a timely and reliable manner.

3.6 CPU Code Optimization: Vectorization on CPU

All x86-64 CPUs support SIMD operations on vectors of 4 single-precision values using Streaming SIMD Extensions (SSE). The latest processors from AMD and Intel support operations on 8 single-precision values using Advanced Vector eXtensions (AVX). Most vectorized operations are applied to all elements of the vector simultaneously, therefore increasing the instruction per clock (IPC) count. However, it does not translate into double and quadruple performance increase due to other bottlenecks such as memory bandwidth.

Two vector versions of TPACF using 128 bit SSE-instructions and 256-bit AVX instructions have been implemented. The SIMD versions were constructed using SSE and AVX intrinsic functions. These intrinsic functions map directly into SIMD instructions. This is similar to using assembly instructions directly. However, the compiler is free to optimize and re-arrange the operations which often results in improved performance.

The vector implementation is similar in structure to the reference implementation (Algorithm 1). The most significant difference is that it calculates 4 (with SSE) or 8 (with AVX) iterations simultaneously. This is achieved simply by using vector variables which hold 2 or 4 single-precision elements at a time. TPACF

calculations are completely independent from each other, therefore vectorization is a relatively trivial task. The end results are transferred into bins from the vector variable, one element at a time in a serial fashion.

The vector instruction set is continuously growing to include more data types and more versatile vector operations. In the SSE and AVX instruction sets trigonometric and inverse trigonometric functions —arc-cosine— are not yet available as vectorized operations, but are implemented by a vectorized library function. The arc-cosine is implemented as a vectorized function based on a series expansion of seventh order multiplied by a square root[8]. This simplifies the calculation of the arc-cosine as before entering this code have already taken care of the possibility of an invalid argument for this calculation, this is, a value greater than 1.0 or smaller than -1.0.

3.7 Parallelization on CPU

Parallelization of TPACF can be implemented for shared memory parallelism using OpenMP or pthreads. The use of OpenMP pragmas at the double loop over the galaxies (Algorithm 1) is straightforward bearing in mind that incrementing the histogram bins should be done with mutual exclusion among the threads.

A shared memory parallelization of TPACF using pthreads was also implemented, where each thread being assigned a section of the outer loop of galaxies such that load balance is maintained between the threads. However, as the performance of the pthreads implementation was practically identical to the OpenMP implementation, we do not report the results from using pthreads separately.

The calculation of DD and RR catalogues are symmetrical. Therefore it is enough to calculate only half and simply double the results. This results in spent time to get reduced as calculations proceed forward.

4 CPU Implementation: Results and Analysis

First a reference serial implementation was created in C for baseline results. However today's CPUs have more than one core, therefore it would not be a fair comparison to compare performance of serial CPU version to GPU/ACCELERATOR version. Therefore, threading through OpenMP was used for fully utilizing the cores of the CPUs. Test results showed that threading using pthreads allowed similar performance, however OpenMP was chosen for its simplicity. The execution times for CPU implementations are presented in Table 1.

5 GPU Implementation: Results and Analysis

An initial optimization is performed implementing strategies such as: the use of streams, reducing branching, increment the occupancy and the data locality. The

[8] http://simdmath.sourcearchive.com/documentation/1.0.2-2ubuntu1/ppu_2simdmath_2acosf4_8h-source.html

Table 1. CPU Execution Walltimes (seconds) for diverse CPU implementations.

Implementation	A10-5800K	i7-3770K
Reference	7999	7450
SSE	1353	1117
AVX 1 core	657	753
AVX 2 Core	339	362
AVX 4 Core	260	181
AVX 8 Core HT	-	162

modifications yield diverse degree of success (Table 2) and pointed that further improvements were feasible. These results correspond to the highest level of accuracy possible, no approximations are made. For this reason, these results are valuable and they are presented.

All these positives modifications have been incorporated to the public version of the code freely available at http://wwwae.ciemat.es/cosmo/gp2pcf/.

Table 2. Mean execution time (ms), reduction of the execution time and speedup for original code and when implementing all the positive strategies: the use of streams, reducing branching, increment the occupancy and the data locality.

Implementation		
Single Precision, Compute Capability 1.2, GTX295	Original Code	299,566.4±15.3 ms
	Positive Strategies	275,303.8±17.6 ms
	Reduction	**24,262.6**
	Speedup	**1.09**
Single Precision, Compute Capability 2.0, C2075	Original Code	314,346.8±199.6 ms
	Positive Strategies	269,969.5±69.8 ms
	Reduction	**44,377.3**
	Speedup	**1.16**
Double Precision, Compute Capability 2.0, C2075	Original Code	452,097.3±287.2 ms
	Positive Strategies	438,934.0±132.2 ms
	Reduction	**13,163.4**
	Speedup	**1.03**

The modifications differ in the degree of success (Table 2). As can be appreciated for single precision the modifications have a higher impact over the execution time than for double precision.

Fortunately the use of double precision is only required when binning angles lower than 0.003 degrees. This happens when changing the histogram setup, and the researchers require a high detail level for very short range of angles, for instance from 0 to 4 degrees with 256 bins.

The optimization processes presented in this section aim to be generalist, in such way that the resulting code will be widely applicable. No optimization processes associated to the particularities of the input file, e.g. equality of the number of galaxies in the real and random data sets; or the region of the space covered have been applied until this point, e.g. arc-cosine polynomial series expansion. Therefore, these execution times are the expected ones for the baseline code; meanwhile shorter execution times will be presented in the next section based on extra advance optimization processes.

5.1 Advanced Optimization

The GPU implementations use CUDA and OpenCL events to measure kernel runtimes. The event timers are very accurate and allows detection of small changes in the runtime when optimizing the kernel. The walltime is used for determining total execution time on CPU, GPU and ACCELERATOR implementations.

The best execution times for the cards used are presented at Table 3. The results show spectacular reduction of the execution time when executing on AMD GPUs.

Table 3. GPU Execution Walltimes (s)

Implementation	GPU	Walltime (s)
OpenCL	Radeon 7970 Tahiti GCN	6.1
OpenCL	Radeon 5870 Cypress	10.2
OpenCL	Tesla M2050	46.1
Cuda	Tesla M2050	53.2
OpenCL	Xeon Phi 5110P	126

6 Conclusions

In this work, the efforts leading to improve the performance of diverse implementations over accelerators: GPU and Xeon Phi, and over CPU hardware calculating the Two-Point Angular Correlation Function have been presented.

Beyond the technical challenge and independently of the implementation, this work produces an effective increment of the productivity of the application through the increment of the number of executions by time unit. For this reason, the contributions are relevant for the exploitation phase of the code.

Besides, the embarrassingly parellizable nature of TPACF calculations allowed a good scaling for the vector and parallel versions. The performance improvement on the CPU devices was significant with use of the new AVX vector instructions combined with OpenMP for parallelization. The OpenCL GPGPU

code resulted in phenomenal performance on GPU devices compared to today's most advanced CPU and Accelerator devices.

Finally, the performance improvement attained in TPACF will pave the road for attempting to calculate more computing intensive algorithms. As offspring of this work, a public version has been released to the community, being freely accessible at: http://wwwae.ciemat.es/cosmo/gp2pcf/. The algorithm becomes substantially more computationally intensive when a galaxy with shape (elliptic) is considered, and not as simple points. This improved algorithm will allow the measure of distortion in view by the mass present in the line sight between the galaxy and the observer.

More comparative works and additional optimizations are proposed as Future Work. First of all, an in-depth experimental asymptotic analysis where other metrics, more stringent, will be performed: $\Omega()$ and $\Theta()$. Besides, finer-grained performance analysis due to diverse optimization techniques is considered for the CUDA-GPU implementation. This analysis will allow an in-detail understanding of execution time reduction depending on the specific modification. Full comparison with an OpenCL implementation is also proposed, specially when executing it on Xeon Phi processor. Furthermore, a scenario where OpenMP-CPU and CUDA-GPU are concurrently exploited is foreseen to be analysed.

Acknowledgements

We thank the Spanish Ministry of Science and Innovation (MICINN) for funding support through grants AYA2009-13936-C06-03 and through the Consolider Ingenio-2010 program, under project CSD2007-00060.

E. Yurtesen acknowledges the use of the resources of the Finnish Grid Infrastucture FGI.

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) via the project EGI-InSPIRE under the grant agreement number RI-261323.

We acknowledge the use of data from the MICE simulations, publicly available at http://www.ice.cat/mice

References

1. Frieman, J., Turner, M., Huterer, D.: Dark energy and the accelerating universe. Ann.Rev.Astron.Astrophys. **46** (2008) 385–432
2. Kindratenko, V.V., Myers, A.D., Brunner, R.J.: Implementation of the two-point angular correlation function on a high-performance reconfigurable computer. Sci. Program. **17** (August 2009) 247–259
3. Roeh, D.W., Kindratenko, V.V., Brunner, R.J.: Accelerating cosmological data analysis with graphics processors. In: Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units. GPGPU-2, New York, NY, USA, ACM (2009) 1–8

4. Cárdenas-Montes, M., Rodríguez-Vázquez, J.J., Ponce, R., Sevilla, I., Sánchez, E., Colino, N., Vega-Rodríguez, M.A.: New computational developments in cosmology. In: Ibergrid. (2012) 101–112
5. Moore, A., Connolly, A., Genovese, C., Gray, A., Grone, L., Kanidoris, N., Nichol, R., Schneider, J., Szalay, A., Szapudi, I., et al.: Fast algorithms and efficient statistics: N-point correlation functions. astroph0012333 (2000) 71
6. Eriksen, H.K., Lilje, P.B., Banday, A.J., Górski, K.M.: Estimating n-point correlation functions from pixelized sky maps. The Astrophysical Journal Supplement Series **151**(1) (2004)
7. Landy, S.D., Szalay, A.S.: Bias and variance of angular correlation functions. American Journal of Physics **412** (July 1993) 64–71
8. Sanders, J., Kandrot, E.: CUDA by Example: An Introduction to General-Purpose GPU Programming. 1 edn. Addison-Wesley Professional (July 2010)
9. Nugteren, C., van den Braak, G.J., Corporaal, H., Mesman, B.: High performance predictable histogramming on gpus: exploring and evaluating algorithm trade-offs. In: GPGPU, ACM (2011) 1

Performance Assessment of a Chaos-based Image Cipher on Multi-GPU

Juan José Rodríguez-Vázquez and Miguel Cárdenas-Montes

Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Department of Fundamental Research. Avda. Complutense, 40. 28.040. Madrid (Spain)
{jj.rodriguez, miguel.cardenas}@ciemat.es

Abstract. There is an ever-increasing need for security on public networks these days, in part, because of the rise of mobile devices with multimedia capabilities along with the blossoming of the Web 2.0. Chaos-based ciphers have demonstrated to be faster than standard ones (RSA, AES, etc.) ciphering multimedia information. Nevertheless, the former ones still need a lot of computational power when dealing with high resolution images. In this work, and following this context, several parallel algorithms of a chaotic cipher are evaluated from a performance standpoint in order to gain an insight into the appropriateness of the Graphics Processing Units as accelerators for this sort of ciphers.

1 Introduction

In the last decade we have seen how digital video and image information flows have been on the increase throughout open networks. This has been so due to many reasons, of which we can highlight: the blossoming of the Web 2.0 and the rise of compute capability in mobile devices along with the improvement of public networks' bandwidth and lower charges. All this has led to the reinforcement of security, assuring authorized access to sensible data. Furthermore, special security measures are necessary for certain storage and image transmission applications. These security requirements have resulted in the development, during the last decade, of ciphers based on manifold principles. Among them, chaos-based encryption techniques are excellent for a practical use since they are: fast, secure, complex and their computational needs are reasonable, unlike traditional ciphers (DES, AES, RSA, etc.) which require much more computational power [1].

The features of chaotic functions [2],[3] have attracted cryptographers' attention when developing new ciphers. These chaotic functions have some fundamental properties like: ergodicity (that is, time or statistical average is the same as state space average); mixing (any given region of its state space will eventually overlap with any other given region as the system evolves over time); and sensitivity to initial conditions (a tiny variation in the starting point will lead to a very different future trajectory). This features can be considered similar

to some ideal cryptographic properties like: confusion (the relationship between ciphertext and key should be as complex as possible), diffusion (idem between plaintext and ciphertext) and avalanche effect (a slight change in the input leads to a significant change in the output).

Our main goal in this work is to parallelize and assess the performance of the chaotic image cipher proposed by Pareek *et al.* [4]. This cipher uses two logistic maps (3) in their most chaotic fashion whose initial conditions are calculated from the external 80-bit symmetric key.

Chebyshev Map	$x_{n+1} = \cos(2^k \cdot \cos^{-1} x_n)$	where $x_n \in [-1, 1],\ k \geq 2$	(1)
Discrete Chaotic Sequence	$x_{n+1} = (1 - u \cdot x_n^2)$	where $x_n \in [-1, 1],\ u \in [1.41, 2]$	(2)
Logistic Map	$x_{n+1} = r \cdot x_n \cdot (1 - x_n)$	where $x_n \in [0, 1],\ 3.99465 < r \leq 4$	(3)
Tent Map	$x_{n+1} = \begin{cases} \mu \cdot x_n & \text{for } x_n < \frac{1}{2} \\ \mu \cdot (1 - x_n) & \text{for } x_n \geq \frac{1}{2} \end{cases}$	where $x_n \in [0, 1],\ \mu = 2$	(4)

Table 1. An example of chaotic maps which can be used to develop chaotic cryptosystems

Ciphering a pixel is a very costly process because, aside from the operations on the pixel, many iterations of the map are needed too, which translates into an intensive use of the floating-point unit. For this reason, GPGPU[1] [5],[6] comes in handy since graphics processing units may have a very large number of cores per chip, being each core capable of running tens of threads "concurrently" which results in thousands of threads in flight simultaneously along with the huge throughput in double-precision floating-point calculations.

The rest of the paper is organized as follows: Section 2 reviews the related work. Section 3 explains how the chaotic cipher works. Section 4 describes the experimental setup, the proposed parallel implementations and the statistical tools. Section 5 presents the results of the tests and the analysis and Section 6 closes the paper with the conclusions and the future work.

2 Related Work

Although many software encryption schemes based on chaos have been developed in the last two decades, since the discovery of the principles of chaotic synchronization by Pecora and Carroll [7] at the end of the 1980s, few of them are parallel. As examples, we can cite the FPGA-implemented real-time image

[1] General-Purpose Computing on Graphics Processing Units.

cipher designed by Azzaz *et al.* [8], which is based on switching two continuous chaotic systems (Lorenz's and L's) for key generation, and the one proposed by Zhang *et al.* [9] that uses two chaotic maps for permutation, diffusion and substitution of the image's pixels on an ASIC[2] using reconfigurable logic.

As far as GPGPU is concerned, there are a few papers that use GPU as a platform for speeding up ciphers. In this sense, it is worth mentioning the respective implementations of the AES cipher for ciphering text and images, carried out by Seshadrinathan and Dempski [10], and of the SSL protocol, by Jang *et al.* [11]. But, to the best of our knowledge, this is the first time a muti-GPU system has been used to accelerate a chaotic image cipher.

3 Cipher

Group	Ranges of Y	Operations
1	0.100.13, 0.340.37, 0.580.62	$\overline{R}, \overline{G}, \overline{B}$
2	0.130.16, 0.370.40, 0.620.66	$R \otimes K_4, G \otimes K_5, B \otimes K_6$
3	0.160.19, 0.400.43, 0.660.70	$(R + K_4 + K_5, G + K_5 + K_6, B + K_6 + K_4)_{10}$ (mod 256); $(R + 256 - K_4 - K_5, G + 256 - K_5 - K_6, B + 256 - K_6 - K_4)_{10}$
4	0.190.22, 0.430.46, 0.700.74	$\overline{R \otimes K_4}, \overline{G \otimes K_5}, \overline{B \otimes K_6}$; $\overline{R} \otimes K_4, \overline{G} \otimes K_5, \overline{B} \otimes K_6$
5	0.220.25, 0.460.49, 0.740.78	Like in group two but changing K_4, K_5 and K_6 by K_7, K_8 and K_9
6	0.250.28, 0.490.52, 0.780.82	Like in group three but changing K_4, K_5 and K_6 by K_7, K_8 and K_9
7	0.280.31, 0.520.55, 0.820.86	Like in group four but changing K_4, K_5 and K_6 by K_7, K_8 and K_9
8	0.310.34, 0.550.58, 0.860.90	No operation is performed

K_i is the i-th session key and R, G and B are the pixel's color components

Table 2. Groups with the different ranges and their corresponding operations for both encryption and decryption

As has already been said, this work is based on the cipher proposed by Pareek *et al.* [4]. The explanation about the workings of this is shown next.

The proposed encryption procedure uses an external 80-bit symmetric key which is divided into 8-bit blocks; each block is called a *session key*.

$$K = k_1 k_2 \dots k_{10} \text{ (in ascii)}. \tag{5}$$

$$K = k_1 k_2 \dots k_{20} \text{ (in hex)}. \tag{6}$$

The cipher employs two logistic maps (3) with $r = 3.9999$; this case corresponds to a highly chaotic one. The value of this parameter is kept constant all the time while the initial condition for each map is calculated as follows.

[2] Application Specific Integrated Circuit.

The first map's initial condition is given by:

$$X_0 = (X_{01} + X_{02}) \bmod 1. \tag{7}$$

For calculating the first term, three session keys are chosen: K_4, K_5 and K_6. These three bytes are arranged forming the binary string: $B_1 = K_{41}K_{42}...K_{48}$ $K_{51}K_{52}...K_{58}K_{61}K_{62}\ldots K_{68}$ where K_{ij} is the j-th binary digit of the i-th session key. Next, the real number X_{01} is computed as shown:

$$X_{01} = (K_{41}\cdot 2^0 + \ldots + K_{48}\cdot 2^7 + K_{51}\cdot 2^8 + \ldots + K_{58}\cdot 2^{15} + K_{61}\cdot 2^{16} + \ldots + K_{68}\cdot 2^{23})/2^{24}. \tag{8}$$

The second term X_{02} is calculated from the following formula:

$$X_{02} = \sum_{i=13}^{18} (K_i)_{10}/96, \tag{9}$$

being each K_i a hexadecimal character of the key as indicated in (6).

Once the first map's initial condition has been figured out, a sequence of 24 real numbers: $f_1, f_2, \ldots, f_{24}$, is generated by iterating the map; taking only into account the values within the range $[0.1, 0.9]$. After that, the sequence is converted into integer:

$$P_k = \text{ int } (23\ (f_k - 0.1)/0.8) + 1 \text{ where } k \in [1, 24]. \tag{10}$$

In the same manner, the second map's initial condition is given by:

$$Y_0 = (Y_{01} + Y_{02}) \bmod 1. \tag{11}$$

As before, another three session keys: K_1, K_2 and K_3, are taken to form a second binary string: $B_2 = K_{11}K_{12}\ldots K_{18}K_{21}K_{22}\ldots K_{28}K_{31}K_{32}\ldots K_{38}$ where K_{ij} is the j-th binary digit of the i-th session key. From this string, Y_{01} can be calculated:

$$Y_{01} = (B_2)_{10}/2^{24}. \tag{12}$$

Afterwards, the second term Y_{02} is calculated in the following way:

$$Y_{02} = (\sum_{i=1}^{24} B_2[P_k] \cdot 2^{k-1})/2^{24}, \tag{13}$$

being $B_2[P_k]$ the value of the P_k-th bit of the binary string B_2.

Having already calculated both initial conditions, we can proceed to encrypt the image. After reading the first pixel, the second map is iterated once; the result decides what operation to apply to each pixel's components (as shown in Table 2). This step is repeated $(K_{10})_{10}$ times. The same is done for the next 15 pixels.

After encrypting a 16-pixel block, the first nine session keys are modified as follows:

$$(K_i)_{10} = (K_i + K_{10})_{10} \bmod 256 \ (1 \leq i \leq 9), \tag{14}$$

and a new sequence of 24 real numbers is created iterating the first logistic map (from the last state) in order to create a new second logistic map and encrypt other 16-pixel block. This is repeated until all the image has been processed. The decryption method is the same except for the operations on the pixels which are carried out in reverse order.

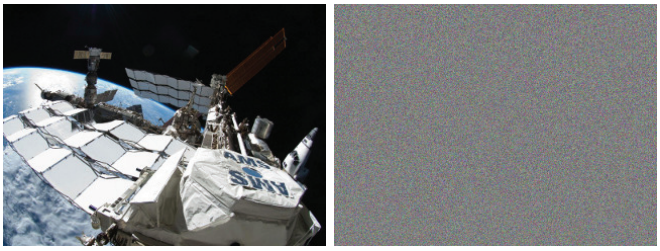
4 Methodology

4.1 Experimental Setup

The tests were conducted on a machine with two quad-core Intel Xeon E5520 processors at 2.27 GHz, 6GB of RAM, two Nvidia GeForce GTX 295 (compute capability 1.3) and two Nvidia Tesla C2075 (compute capability 2.0). The development tools used were G++ 4.4.5, Qt 4.7.0 and CUDA Toolkit 4.0 with graphics driver 270.41.19 under Fedora Core 13 (kernel 2.6.33.3-85.fc13.x86_64).

4.2 Algorithms

As was explained in Section 3, the logistic map and the key each pixel block uses are different and depend completely on the previously-generated ones. This implies that the generation of keys and logistic maps has to be accomplished serially. In contrast, each pixel ciphering can be done in parallel once keys and maps are created. All algorithms on this work, both on CPU and on GPU, follow this approach: generating first on CPU keys and logistic maps before proceeding to the ciphering stage which is done on the respective platform.



(a) (b)

Fig. 1. (a) Color image (1,600 x 1,200 pixels) used throughout the tests and (b) encrypted color image using the external 80-bit symmetric key "i0QYELdEI5" [3]

[3] Image credit: NASA.

There is one slight difference between encryption and decryption process: the order in which operations are applied on each pixel. For this reason, a buffer - whose size is dictated by the key's least significant byte K_{10} - is employed by the decryption routines to store the operations and execute them in reverse; hence, these routines consume a little bit more memory and time than encryption ones.

Additionally, each algorithm has been coded to support both single and double precision so as to see how this impacts performance. This is a very important detail for chaotic ciphers implemented on digital computers since it affects map's dynamical properties. Analog chaotic ciphers do not have this kind of problem since they are based on an infinite field of numbers; but for digital ones, which are bound to a finite field, it is an important consideration for it affects the length of the orbits, making them shorter and repetitive as the precision lowers, and therefore, more prone to attacks.

Regarding execution, 20 launches were done for each configuration: <algorithm, precision, platform and operation [encryption or decryption]>. The timing only takes into account the interval went by while creating keys and logistic maps plus the ciphering process itself. This last one includes memory allocations and deallocations and initialization of data structures along with the aforementioned operations on the pixels. On GPU, time spent on data copies between host and device, and vice versa, is also added, being lower than four milliseconds for each sense. All measurements were done with millisecond precision using the most appropriate method for each platform.

Next, the specific details for each implementation will be described.

CPU Aside from the sequential version and for comparisons sake, it was created an OpenMP variant - this version as well as the sequential one are compiled with the third level of optimization - which is able to exploit all the CPU cores on the system. This algorithm splits up every image's row into as many pieces as there are threads, being therefore each thread in charge of the ciphering process of its fragment. Each piece is a multiple of 16 pixels (one ciphering block); an exception to this rule may occur with the last fragment if the number of pixels in a row is not a multiple of 16.

GPU For this platform, four different algorithms along with their variants have been developed. Besides precision, the hardware architecture (compute capability) will also be taken into consideration for every setup.

The features of every implementation are shown next:

– **Baseline kernel:** this is the entry-level implementation with no optimization at all, and which has been parameterized in such a way that the user can specify the number of pixels each thread is going to cipher and the number

of threads every CTA is going to have. Moreover, the pixel array is treated as a rectangular one so indexes are disposed properly in this sense.

– **Optimized global-memory kernel:** in this case, the code was modified to harness on-chip registers, taking special care to avoid register spilling; the use of shared memory was discarded since threads do not need to communicate among them, avoiding at the same time the higher latency and any bank conflicts might appear. Arithmetic was optimized for each architecture as well. For devices with compute capability 1.3, intrinsic functions (24-bit instructions implemented on hardware, like: __fadd, __fmul, etc.) were employed where precision was not a requirement since they are faster than 32-bit ones, which are emulated. On devices with compute capability 2.0, the opposite is true, so 32-bit instructions were used instead of the 24-bit ones. In both cases, modulo operations were replaced with bitwise ones since the former ones are costlier.
  Another aspect that has been polished is how pixel's components are extracted from and assembled to the integer; instead of using bitwise operations, the pixel has been expressed using bit fields. This reduces obviously the amount of necessary operations.
  In general, the kernel has been redesigned in a more specialized and streamlined fashion: now, image is treated as a linear array and the number of pixels ciphered by each thread is fixed to one block (16 pixels); this can be translated as less overhead to figure indexes and boundaries.

– **Optimized texture-memory kernel:** one possible improvement over the previous optimized implementation lies in data retrieval. As was mentioned earlier, texture hardware possesses some cache memory which could come in handy to increase the memory bandwidth and, hence, the performance. Following this criterion, all data: image, keys and maps, have been fetched from device's global memory using this method and all subsequent reads are done from the texture cache. Since texture memory is read-only, this method does not affect writes, which are carried out in the same way. As a drawback, texture units do not support double precision, so the tests have been accomplished using single precision only.

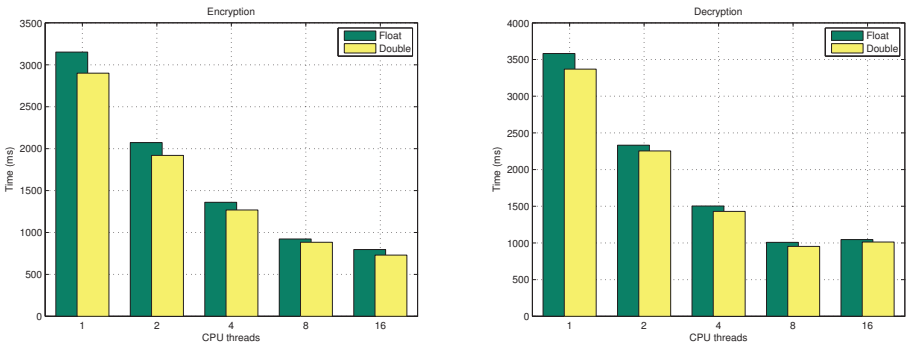
– **Optimized multi-GPU kernel:** this is the multi-GPU version of the texture-memory kernel. The image is divided horizontally into two halves, being every GPU responsible for its own slice.

4.3 Statistical Tools

Since this is a multiple comparison problem, in order to determine if all the datasets belong to diverse distributions, the Kruskal-Wallis test and the Wilcoxon signed-rank test with the Bonferroni adjustment have been applied. The former one is used as an overall test for knowing if at least one pair is different; if so,

the latter test is applied in a pairwise fashion to see which ones are different and which ones are not. Both tests are non-parametric, which are suitable when normality assumption can not be satisfied, as in this case. For each pairwise comparison, the null hypothesis would be that both data sets pertain to the same distribution ($H_0 : m_0 = m_1$) against the alternative hypothesis ($H_1 : m_0 \neq m_1$) which states that both data sets belong to unlike distributions and, hence, possess diverse medians.

5 Results and Analysis



(a) (b)

Fig. 2. Execution times (in ms) of the OpenMP algorithm for (a) encryption and (b) decryption using varying setups on CPU.

5.1 CPU

As can be appreciated in Table 3, the best encryption acceleration is obtained with sixteen threads; unlike decryption process which performs better with eight threads. In the first case, it seems that hyper-threading technology allows to harness a bit better all the functional units the CPU possesses. On the contrary, increasing the number of threads beyond eight in the decryption process worsens slightly the performance. One of the reasons can lead to this is the fight among threads for accessing memory since the decryption process firstly needs to store the operations in order to apply them in reverse.

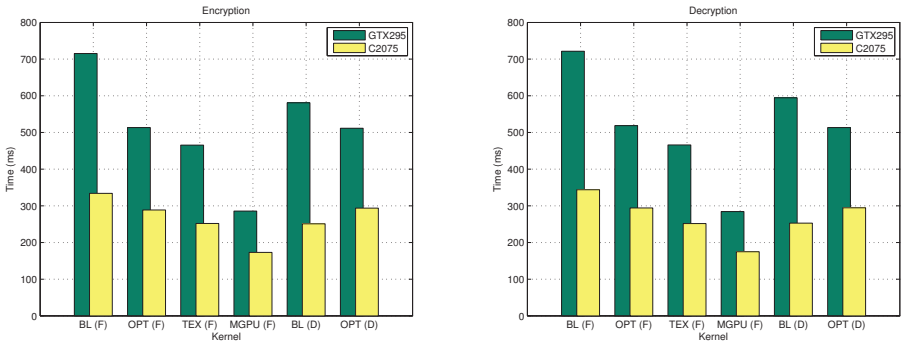
Another fact can be observed is the performance degradation which occurs when using single (float) over double precision. FPU's (Floating Point Unit) do all computations using double precision; so if float is required, figures have to be rounded, spending more time. In this case, this is something beneficial since precision is critical.

CPU Threads	Encryption				Decryption			
	Execution Time ($\overline{x} \pm s$)		Speed-up		Execution Time ($\overline{x} \pm s$)		Speed-up	
	Float	Double	Float	Double	Float	Double	Float	Double
1	$3,151.80 \pm 28.34$	$2,899.80 \pm 23.33$	N/A	N/A	$3,582.10 \pm 27.95$	$3,369.00 \pm 33.35$	N/A	N/A
2	$2,071.80 \pm 43.54$	$1,919.10 \pm 89.25$	1.52	1.51	$2,332.30 \pm 64.68$	$2,254.40 \pm 77.78$	1.54	1.49
4	$1,359.90 \pm 59.82$	$1,268.10 \pm 53.50$	2.32	2.29	$1,502.90 \pm 32.16$	$1,429.80 \pm 25.65$	2.38	2.36
8	921.75 ± 42.20	883.00 ± 64.35	3.42	3.28	$1,008.10 \pm 12.21$	952.65 ± 37.11	3.55	3.54
16 (HT)	796.30 ± 66.31	730.15 ± 56.73	3.96	3.97	$1,045.70 \pm 59.70$	$1,012.50 \pm 105.85$	3.43	3.33
-	ms		-		ms		-	

Table 3. Results for the OpenMP tests using different configurations

Regarding statistics, the Kruskal-Wallis tests returned zero as result for both encyption and decryption data sets; indicating that at least one pair in each group was different. Wilcoxon signed-rank tests with Bonferroni adjustment were carried out right after for every encryption or decryption pair and whose results are commented next. For encryption, the p-values for the 100% of the pairs are below alpha, within the range $[0, 5.18 \cdot 10^{-5}]$; thus, the null hypothesis can be rejected and it can be stated with a 95% of probability that the data sets which comprise every pair belong to different distributions. In the case of decryption, for the 95.5% of the pairs the null hypothesis was discarded (their p-values are within the range $[0, 10^{-4}]$); the opposite happened for the 4.5% of them: two pairs, since their p-values ($1.217 \cdot 10^{-3}$ and $3.904 \cdot 10^{-3}$ respectively) exceeded the alpha value.

5.2 GPU



(a) (b)

Fig. 3. (a) Encryption and (b) decryption results (in ms) for the diverse configurations run on GPU (from left to right: baseline, optimized global-memory, optimized texture-memory and optimized texture-memory multi-GPU kernels using single precision; and baseline and optimized global-memory kernels using double precision).

Observing the results in Tables 4 and 5, it can be appreciated how speed-up grows as optimization increases since faster versions use more registers and rely much less on global memory.

Kernel	Precision	Execution Time ($\overline{x} \pm s$)			Speed-up		
		CPU	GTX295	C2075	GTX295/CPU	C2075/CPU	C2075/GTX295
Baseline	Float	$3,151.80 \pm 28.34$	715.20 ± 5.30	334.03 ± 1.75	4.41	9.44	2.14
Opt. Global		$3,151.80 \pm 28.34$	513.28 ± 1.97	288.68 ± 1.85	6.14	10.92	1.79
Opt. Texture		$3,151.80 \pm 28.34$	465.44 ± 6.48	251.93 ± 1.33	6.77	12.51	1.85
Multi-GPU		$3,151.80 \pm 28.34$	285.77 ± 1.54	173.08 ± 1.60	11.03	18.21	1.65
Baseline	Double	$2,899.80 \pm 23.33$	581.02 ± 6.57	250.96 ± 1.68	4.99	11.55	2.31
Opt. Global		$2,899.80 \pm 23.33$	511.60 ± 5.42	293.87 ± 3.04	5.67	9.87	1.74
		ms			-		

Table 4. Encryption results for diverse configurations run on GPU

Kernel	Precision	Execution Time ($\overline{x} \pm s$)			Speed-up		
		CPU	GTX295	C2075	GTX295/CPU	C2075/CPU	C2075/GTX295
Baseline	Float	$3,582.10 \pm 27.95$	721.47 ± 2.10	343.88 ± 2.86	4,96	10,42	2.10
Opt. Global		$3,582.10 \pm 27.95$	518.76 ± 6.34	294.15 ± 1.43	6,90	12,18	1.76
Opt. Texture		$3,582.10 \pm 27.95$	466.02 ± 2.39	251.61 ± 1.10	7,69	14,24	1.85
Multi-GPU		$3,582.10 \pm 27.95$	284.35 ± 1.69	174.98 ± 1.91	12,60	20,47	1.62
Baseline	Double	$3,369.00 \pm 33.35$	594.64 ± 2.51	252.83 ± 1.48	5,67	13,32	2.35
Opt. Global		$3,369.00 \pm 33.35$	513.26 ± 6.57	294.68 ± 1.87	6,56	11,43	1.74
		ms			-		

Table 5. Decryption results for diverse configurations run on GPU

Surprisingly, the optimized double-precision version performs slightly worse on Fermi than the unoptimized one. One possible explanation for this could be the additional pressure exerted on registers - in the form of bank conflicts - by the former one since now map data are 64-bits in size (double than registers). By contrast, the double-precision baseline kernel is slightly faster than the single-precision counterpart (the same as happens on CPU), which demonstrates that these algorithms are not bound arithmetically.

Overall, the improvements introduced with Fermi make C2075 outperform GTX295 in all the cases with gains which vary from a 50% to a 100%. This is due, to a greater extent, to the cache memory hierarchy of the former one which allows to reduce the number of transactions to the global memory.

Occupancy varies somewhat from GTX295 to C2075 since the number of registers per block has been doubled in C2075 with respect to GTX295; which allows a higher number of threads per block.

The statistical tests were conducted in an identical fashion as the CPU ones - this time there are 12 data sets -. There is not too much to comment about Kruskal-Wallis tests since the outcomes are identical. For the encryption, Wilcoxon signed-rank tests show that the null hypothesis was rejected, with a 95% of probability, for the 100% of the pairs (their p-values are within the range $[0, 1.26 \cdot 10^{-4}]$). For decryption, the hypothesis was admitted for one pair (with p-value: $3.626 \cdot 10^{-3}$) and rejected for the remaining 98.48% of the pairs.

As some final remarks, there are several things to bear in mind when designing new ciphers for GPU regarding performance. Firstly, what has the most negative impact on performance is thread divergence (promoted by control structures), so all threads within a warp would have to follow the same path for all GPU resources to be seized properly; secondly, memory requests should be as coalesced as possible in order to achieve a higher bandwidth; and last but not least, the algorithm's parallel fraction should be one hundred percent - in this case, there would not have to be dependences among keys or maps so they could be generated in parallel on GPU -.

6 Conclusions

The aim of this work was to evaluate the suitability of a multi-GPU system as accelerator for reducing the execution times of a chaos-based image cipher. For this task, several versions of it have been developed, both for CPU and GPU, using OpenMP and CUDA. The good results obtained: more than eighteen-fold and twenty-fold speed-up for encryption and decryption respectively, show the high potential of GPU for speeding up greatly this kind of ciphers.

Nonetheless, some extra work could not have been included due to space constraints. This involves the section on GPU metrics and CTA optimization which is useful to understand better the differences among the diverse algorithms. Moreover, tests using additional image resolutions along with an introductory explanation on GPU architecture were also excluded.

References

1. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook Of Applied Cryptography. CRC Press (2001)
2. Devaney, R.L.: An Introduction To Chaotic Dynamical Systems. 1 edn. Benjamin/Cummings Publishing Company (1986)
3. Strogatz, S.H.: Nonlinear Dynamics And Chaos: With Applications To Physics, Biology, Chemistry And Engineering. 1 edn. Westview Press (January 2001)
4. Pareek, N.K., Patidar, V., Sud, K.K.: Image encryption using chaotic logistic map. Image and Vision Computing **24**(9) (2006) 926–934
5. Kirk, D.B., mei Hwu, W.: Programming Massively Parallel Processors: A Hands-on Approach. 1 edn. Morgan Kaufmann (February 2010)
6. Sanders, J., Kandrot, E.: CUDA By Example: An Introduction To General-Purpose GPU Programming. 1 edn. Addison-Wesley (June 2010)
7. Pecora, L.M., Carroll, T.L.: Synchronization in chaotic systems. Physical Review Letters **64** (1990) 821–823
8. Azzaz, M., Tanougast, C., Sadoudi, S., Bouridane, A., Dandache, A.: Fpga implementation of new real-time image encryption based switching chaotic systems. IET Conference Publications **2009**(CP559) (2009) 56–56
9. Zhang, Y., Liu, Z., Zheng, X.: A chaos-based image encryption asic using reconfigurable logic. In: Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on. (December 2008) 1782 –1785

10. Seshadrinathan, M., Dempski, K.L.: Implementation of advanced encryption standard for encryption and decryption of images and text on a gpu. Computer Vision and Pattern Recognition Workshop **0** (2008) 1–6
11. Jang, K., Han, S., Han, S., Moon, S., Park, K.: Accelerating ssl with gpus. SIGCOMM Comput. Commun. Rev. **40** (August 2010) 437–438

Operation of the ATLAS Spanish Tier-2 during the LHC Run I

E. Oliver[1], M. Kaci[1], M. Villaplana[1], F. Fassi[1], A. Fernández[1], S. González de la Hoz[1], V. Lacort[1], J. Nadal[2], A. Pacheco Pages[2], J. del Peso[3], J. Salt[1], J. Sánchez[1], V. Sánchez[1], S. Sayzhenkova[3]

[1] Instituto de Física Corpuscular, CSIC/Universitat de València, Valencia, Spain
[2] Institut de Fisica d'Altes Energies, Universitat Autonoma de Barcelona, Spain
[3] Universidad Autónoma de Madrid, Madrid, Spain
oliverg@ific.uv.es

Abstract. This work reports on the operation of the Spanish ATLAS Tier-2 during the Run I (2009-2013) of the LHC. The data production, transfer and storage, as well as their distributed analysis using the computing infrastructures of the Spanish ATLAS Tier-2 are presented, and the Tier-2 availability, reliability and operation performance during this period are discussed.

1 Introduction

The ATLAS Spanish federated Tier-2 (T2-ES) is formed by three institutions: the Institut de Física d'Altes Energies of Barcelona (IFAE), the Universidad Autónoma of Madrid (UAM), and the Instituto de Física Corpuscular of Valencia (IFIC) [1]. It is part of the Iberian Cloud (South West of Europe) and its associated Tier-1 is Port d'Informació Científica (PIC) of Barcelona.

The T2-ES federation started operating in 2005 with the objective of building and maintaining a Tier-2 infrastructure for Distributed Analysis and Monte Carlo Production for ATLAS. This infrastructure had to fulfil the ATLAS Collaboration requirements and provide a stable operation serving the ATLAS physicists in their data analyses.

The contribution of the T2-ES has been established at 5% of the ATLAS computing resources. These resources are shared between the three sites involved. IFIC assumes the coordination of the federated T2-ES activities and it provides 50% of the resources, while UAM and IFAE provide 25% each. During the period 2004-2009, T2-ES participated in several data and service challenges, which demonstrated its good operation performance. In 2006, T2-ES started contributing to the production of Monte Carlo simulated events for ATLAS. During this first data taking period, 2009-2013, T2-ES has operated with high reliability and has provided good services to thousands of ATLAS users.

The operation of the ATLAS Tier-2 sites/federations has evolved during the Run-I according to the changes introduced by the Collaboration in the ATLAS computing model. These changes affected the previous hierarchical model based

on the MONARC project [2]. This model considers that the best connection between sites is reached when inter-connecting each big ATLAS Tier-1 site with few associated Tier-2 sites, forming an ATLAS Cloud. Only direct connection between Tier-1 sites is admitted. File transfers between Tier-2 sites belonging to different Clouds must be done through their associated Tier-1.

However, transfer tests [3] during the Run-I showed that some Tier-2 sites have enough network capacity to connect directly to Tier-1 and Tier-2 sites from different Clouds. The ATLAS computing model has been modified to allow inter-connection between such sites, even if they are from different Clouds. These well connected Tier-2 sites, called 'Tier-2 Directly' (T2D) [4], have to fulfill two requirements: the overall transfer rate of big files to or from Tier-1 sites must have been above 5 MB/s during the last week and 3 out of the last 5 weeks, and the availability of the site must be higher than 90%. These requirements are fulfilled by each site of the T2-ES federation, so T2-ES has been promoted to T2D.

ATLAS has changed its data placement policy as well. A rigid way of placing the data replicas at the sites has been substituted by a more flexible data placement policy. In the new data model, additional replicas of popular data are placed on selected sites using the Panda Dynamic Data Placement (PD2P) tool [5], thus reducing the waiting time of analysis jobs and making a better use of the available storage capacity. Thanks to its T2D qualification, the T2-ES now hosts a bigger fraction of popular data, thus receiving a higher number of analysis jobs, and is now able to send and receive data from various sites around the world.

A previous report on the performance of T2-ES was presented only a couple of months after the proton-proton collisions started at the LHC [6]. In the next sections, a detailed description is given of the operation and performance of the T2-ES along the LHC Run-I.

2 The T2-ES computing resources

The computing resources required by the ATLAS experiment are fixed every year. Table 1 shows the CPU power and the disk storage capacity that T2-ES have provided each year to ATLAS.

Table 1. The evolution of the pledges in CPU and disk for the T2-ES.

T2-ES	2006	2007	2008	2009	2010	2011	2012	2013	2014
CPU(HEP-SPEC06)	92	243	1750	5390	10308	13900	13300	18000	20600
DISK(TB)	14	63	387	656	1107	1880	2350	2550	2800

It is worth highlighting the important increments of the T2-ES hardware, particularly just before the data taking started in 2009. The most significant increment was in 2008, where the resources where extended a 600%. During

2010-2014, the increase have been smoother, varying from 15% to 80%. The CPU power and disk capacity provided by T2-ES in March 2013 are reported in Table 2. These quantities satisfy the pledges shown in Table 1.

Table 2. The ATLAS Spanish Tier-2 resources in June 2013.

T2-ES	CPU(HEP-SPEC06)	DISK(TB)
IFIC	14573	1316
UAM	3780	618
IFAE	4599	679
Total	22952	2613

In T2-ES 6 Computing Elements (CE) are used to process the user's analysis jobs and the official Monte Carlo production. The outputs from these jobs use 4 Storage Elements (SE). According to the ATLAS Collaboration, a Tier-2 must divide its resources evenly among these two tasks (50% each).

To manage the SEs, dCache [7], a system specially developed to manage the huge amount of data files typical in High Energy Physics, is used in UAM and IFAE (disk+SRMposix), while the Lustre [8] distributed file system, which provide a POSIX interface, is used at IFIC.

PUPPET [9] is used to manage the software installation in T2-ES: to install and configure the operating systems, the Grid middleware, and the storage systems. ATLAS software is accessed by the sites machines using the CernVM-FS [4].

The network is provided by the Spanish National Research and Education Network (NREN) RedIRIS [10] and local providers for the last mile. The links are 10 Gbps between POPs (Point of Presence) with alternate paths for backup. The connectivity from the T2-ES sites (IFIC, IFAE, UAM) to network is 10 Gbps. In addition, these sites are interconnected by a triangular backbone link provided by RedIRIS.

3 The Reliability and Availability of T2-ES

The ATLAS collaboration considers essential to control the Tiers established commitments. Apart from confirming the resource pledges, a Tier-2 infrastructure must be available as long as possible for the end-user. This is measured using the Service Availability Monitoring (SAM) [11] metrics, namely the reliability and the availability of the sites.

The availability is defined as the ratio of the time the site is available over the total time, while the reliability is defined as the time the site is available over the total time corrected by taking away the scheduled downtime of the site (for technical stop for example). To measure these metrics, SAM runs a set of critical tests at regular intervals of time along the day on the sites. The sites are considered to be available/reliable when these tests complete successfully.

Figure 1 shows the measured availability and reliability of T2-ES for the period of time from January 2010 to February 2013. It also shows that sometimes the metrics drop down for a given site, but the value keeps around 90-100% for the whole federation.

In particular, the availability has more falls because the scheduled downtimes are included in the statistics. Scheduled downtimes do not reflect a problematic situation, however, the resources are not accessible for the users, and thus, availability tests fail. Therefore, both metrics are necessary in order to find out the cause of a inaccessibility of a Tier.

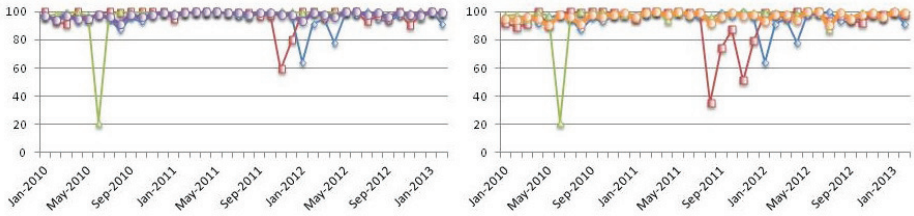


Fig. 1. Reliability (left) and Availability (right) measured at T2-ES for the last 3 years.

4 The data transfers involving T2-ES

In ATLAS, data are transferred among Tiers after the reprocessing, or after the ATLAS official production in the case of simulations. Additional data replicas are transferred automatically with PD2P (Panda Dynamic Data Placement) based on usage. On-demand replicas can be requested by users with DATRI. In addition, there are data transfer tests everyday and sometimes specific checking such as Sonar tests [3] to measure the connections between sites.

A transfer operation involves a source site, a destination site, a number of files to move, a transfer speed of the data volume and the status of the transfer (success or failure). The metrics used to evaluate a transfer operation between sites are the throughput, the transfer efficiency and the number of files transferred successfully. The transfer operations in ATLAS cover various activities: T0 Export, Data consolidation, Data brokering, Functional Test, Production, Group Subscription and User Subscription, as organized in the ATLAS dashboard [12].

During Run-I, the transfer throughputs within ATLAS have been increasing from 782 MB/s in January 2010 to a maximum of 9426 MB/s in January 2013. Accordingly, the number of files transferred successfully has been increasing as well, with a peak of 41.7 million files in November 2012.

Around 1007 million transfers have been done for last three years, and 398 PB of data have been transferred with an efficiency of 80% - 100% with punctual drops bellow 75% in some Clouds. The average value of the efficiency in data

transfers within ATLAS, over the last three years, is measured to be 92%. The highest number of data transfers within ATLAS, by activity, have been done in 'Production' (35%), 'User Subscription' (29%), and 'Data consolidation' (19%). Part of these transfer operations involved the T2-ES in the following way:

4.1 T2-ES as a source of file transfers

Figure 2 shows the throughput and the number of files successfully transferred from T2-ES to other ATLAS sites between January 2010 and March 2013.

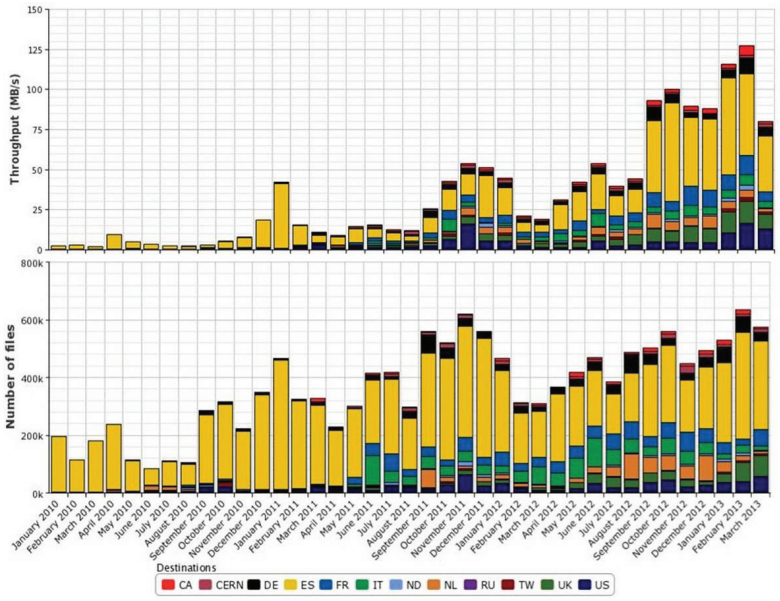


Fig. 2. Transfers from T2-ES to other sites of ATLAS, from January 2010 to March 2013. Throughput (top) and number of files transferred (bottom).

This figure shows that the throughput has been increasing since 2010 and reached 127 MB/s in February 2013. The change in the computing model, from hierarchical to mesh, is clearly reflected at the beginning of 2011, since then, T2-ES sends and receives files from various Clouds.

An upward tendency is also noticed in the number of files transferred. However, it is not constant, with a peak of 633 thousand transferred files in February 2013. A total of around 14.2 million files have been transferred successfully by the T2-ES over the last 3 years, which represents 3547 TB of data, with an average efficiency of 94%, a little higher than the whole ATLAS efficiency.

Concerning the successful transfers from the T2-ES as a function of the activities, it is observed that most transfers done from T2-ES are for Production

65%. The next significant activity is the user subscription (22%). This is due to the fact that ATLAS users from all over the world use DATRI to move their output files, which are stored in T2-ES, to their institutes.

4.2 T2-ES as a destination of file transfers

Figure 3 displays the transfer throughput as well as the number of files received by T2-ES from the rest of ATLAS sites. The throughput increased reaching a maximum of 243 MB/s in February 2012.

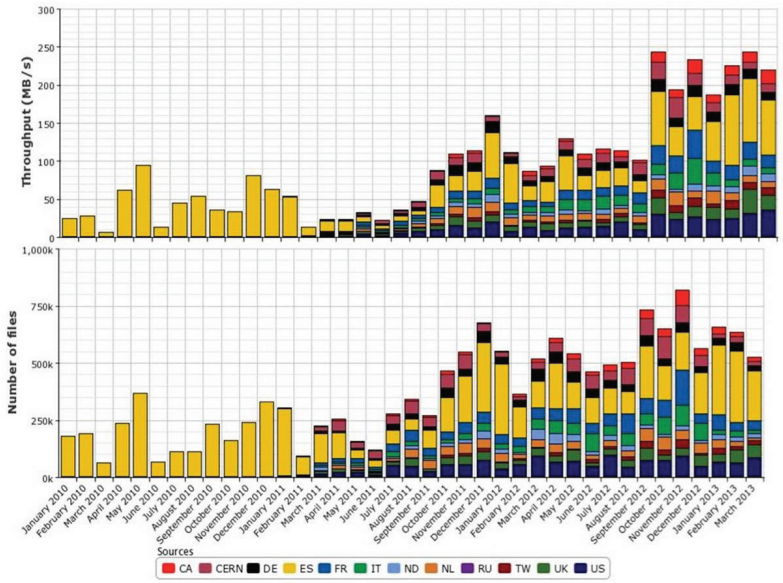


Fig. 3. Transfers from ATLAS sites to T2-ES, from January 2010 to March 2013. Throughput (top) and number of files transferred (bottom).

Taking into account all the transfers to T2-ES, 14.6 million files have been transferred. The trend is to grow up reaching 810000 transferred files in November 2012. Also, the computing model change is noticed at the beginning of 2011.

According to this figure, 9.60 PB of data have been transferred to T2-ES with an efficiency of 80% - 100%, then stabilizes at 90% - 100% after 2012. Taking into accounting all the transfers to the T2-ES, the efficiency is around 88%. The received number of files increases by a factor two after July 2011.

Concerning the transfers as a function of the activity since november 2010, an almost constant flow of around 200000 user subscriptions per month is observed. The activities with the most contribution are Data Consolidation 29%, User Subscription 27%, Production 16% and Data Brokering 14%. The great

percentage in the user data transfers reflects the Spanish users analysis activity. The small contribution of 'T0- Export' 2% is due to the transfers from Tier-0 to IFIC, which has been assigned to perform ATLAS calibration tasks since April 2011. In these tasks, information on detector components is used to derive corrections that are later applied to physics analysis.

5 The data storage in the T2-ES

Independently of the file system used, every ATLAS Tier-2 has data stored in space tokens [13], which are reserved labeled storage spaces. In particular, permissions are granted to users or activities to read, add or remove data in a space token according to the Grid certificate role and to the approval of the people in charge. The ATLAS Tier-2 space tokens are the following:

- DATADISK: where the official ATLAS data are stored, from the experiment or from the Monte Carlo production.
- HOTDISK: files with frequent use are stored here like database files.
- CALIBDISK: is the place reserved for calibration operations of the ATLAS detector, only in some sites of Tier-2, for instance IFIC.
- PRODDISK: is the specific space for the files needed during the process of the ATLAS simulation production.
- GROUPDISK: only users working in the physics group can store data under the approval of the conveners of the group.
- SCRATCHDISK: a temporary space where all output analysis job files are stored when the process has finished.
- LOCALGROUPDISK: is the space for local users of an institute or a Cloud. It is frequently a Tier-3 storage subject to ATLAS pledges.

Figure 4 shows the occupancy as a function of the space token of all the Tier-2 sites involved in ATLAS. The greatest space token occupancy is the DATADISK (62% occupancy), followed by the GROUPDISK (19%), and the LOCALGROUPDISK (17%). Figure 4 also shows the occupancy in T2-ES as a function of the space token: DATADISK (57%) and GROUPDISK (31%).

The total data stored in all the space tokens of T2-ES reaches 2 PB, and are distributed as follows: IFIC:48%, IFAE:29% and UAM:23%, proportional to the storage resources of each site. If we compare the T2-ES occupancy with all data stored in the ATLAS Tier-2 sites, the contribution of the T2-ES is 5.1%.

The evolution of the data occupancy in the whole ATLAS storage space over the last four years shows a steady increase every year since mid-2008, reaching 140 PBs in total data. The evolution with time of the storage space occupancy for T2-ES shows a similar trend as the one of all ATLAS. An example of this evolution is given in Figure 5 for the DATADISK space token, for each site of the T2-ES federation, since the token creation date to February 2013.

The green line (labeled SRM total) represents the available space, the blue one (labeled DQ2) is the size of the registered files in DQ2 and the red line (labeled SRM used) is related to the real physical space occupied. From Figure 5,

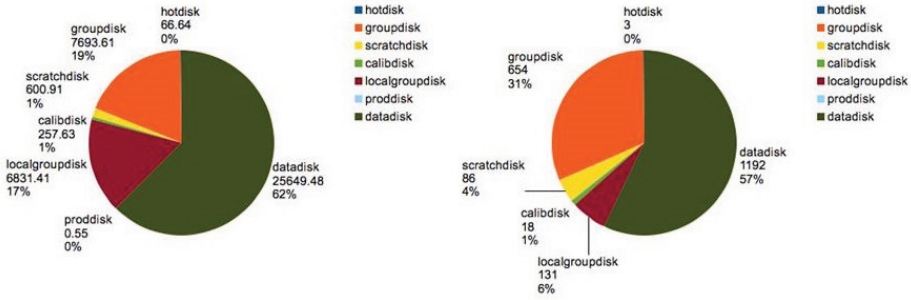


Fig. 4. Occupancy of the space tokens. All ATLAS Tier-2s (left), T2-ES (right).

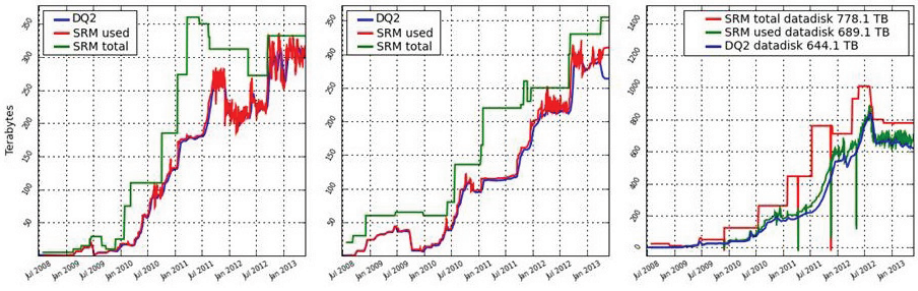


Fig. 5. Occupancy of the T2-ES DATADISK space token: IFAE (left), UAM (center), IFIC (right).

it is discerned that in all plots the occupancy has been increasing since November 2009, when the first proton-proton collisions started in the LHC.

6 The production and analysis jobs in the T2-ES

Production of simulated events and distributed data analysis are the most important ATLAS computing activities in a Tier-2. These activities have been the subject of studies and continuous monitoring throughout their development within the Iberian Cloud, and particularly in T2-ES [14, 15].

Monitoring these activities is vital to watch a Tier-2 performance. T2-ES demonstrates to be efficient during the data taking period in Grid job execution as well as in the optimal use of the resources. The ATLAS Dashboard monitoring web site [12] has been used to produce statistics about the production and analysis jobs, in the period of time from January 2010 to April 2013. This tool was developed by ATLAS to monitor several operations.

6.1 Production Jobs

ATLAS produces the official simulation data and the official DPDs for the physics groups. These production jobs are useful to test any time the T2-ES because a reliable code is executed and a real job flow is present.

The performance of this job flow in ATLAS and T2-ES is shown in Figure 6. In the histogram at the bottom, ATLAS completed jobs are displayed and it is noticed that the production is not totally regular per month. For instance, there were low activity periods like June-July 2010, and the opposite in September 2011 and August 2012.

Around 7 million jobs per month have been processed in ATLAS reaching 10 million in September 2011. 120 thousand jobs per month have been executed in T2-ES, although 250 thousand have been able to succeed. During this period of three years, 265 million production jobs have been run, 4.7 million in T2-ES. The main contribution is from IFIC with 2.3 million jobs. This is expected because this site is providing 50% of the T2-ES resources. The next contributions are from IFAE with 1.4 million and the last UAM with 958 thousand. The T2-ES contribution to the production of simulated data corresponds to a 1.78% of ATLAS total production.

We continue with the job efficiency comparison, which is shown in Figure 6. In the histogram at the top, the T2-ES sites efficiencies are displayed. The average value of the efficiency is 93%. In the ATLAS histogram (bottom), the average efficiency in this period is around 95.7%.

6.2 Analysis Jobs

The activity called Analysis consists on the jobs which are sent by ATLAS physicists for their studies. The number of analysis jobs was low in the beginning, however, this tendency changed completely when data-taking started.

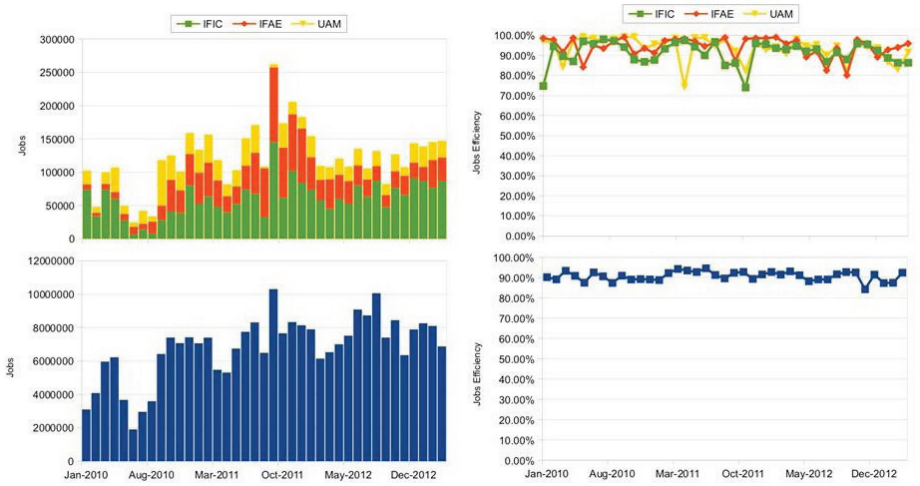


Fig. 6. Successful production jobs (left) and job efficiencies (right). T2-ES (top), all ATLAS (bottom).

Figure 6 displays the number of successful analysis jobs and their corresponding job efficiency for ATLAS and T2-ES. Ups and downs are observed in the ATLAS histogram of completed analysis jobs. These peaks can be explained if an important conference is close in time like July (iCHEP conference). In August 2012 case, the increment of 4.6 million could be on account of the intensive activity of the Higgs Physics group after the announcement of the discovery of a new particle, to confirm it has exactly the Higgs properties. In any case, an overall steady increase can be noticed since April 2010.

In the T2-ES histogram, a progressive increment is better noticed, with a marked peak in February 2013 of over 400 thousand jobs. This value is higher than production jobs in the same month. In Figure 7, the total number of completed analysis jobs in ATLAS in this period is 407 million, of which 7.2 million went to T2-ES, the 1.76% of ATLAS jobs.

The analysis job efficiencies vary in an irregular way, both for all ATLAS and for T2-ES. The ATLAS efficiency average is 87.9% and T2-ES has 89.3% for what this last value exceeds a little the general efficiency average. Both values are smaller compared to the production job efficiencies, although this is logical since production jobs are more stable and more reliable.

In the case of analysis jobs, errors can happen for reasons different from computing resources, they can be due to wrong user algorithms or input data not found. Therefore, analysis jobs efficiency should not be considered a site status parameter. For that reason, the HammerCloud has been created, to have a constant analysis job flow and, thus, to test the sites. Its jobs have the analysis characteristics, many input/output data different from production jobs with more CPU waste.

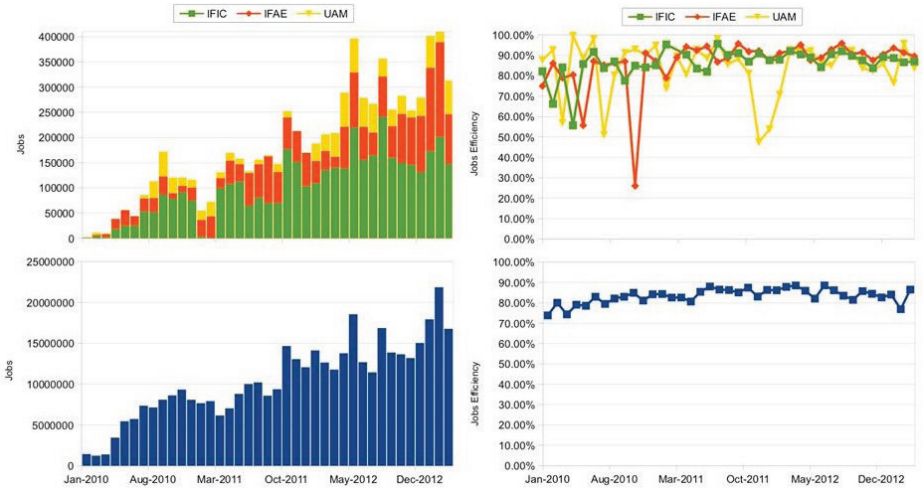


Fig. 7. Successful analysis jobs (left) and job efficiencies (right). T2-ES (top), all ATLAS (bottom).

7 The user support in the T2-ES

One of the T2-ES activities is the support provided to users at each centre in their data analysis using the adequate tools on the Grid infrastructures. Each institute owns its user support team, which takes the control in creating and updating web pages with the essential information to start and to access the Grid in each centre. In addition, the team attends questions and problems of these users, by e-mail, phone or face to face.

User support teams make also a liaison between the users and the system administrators to solve specific problems with the infrastructures. The teams can notify Grid news as downtimes of sites or services because of a local or ATLAS global issue.

Sometimes, external links are required to solve the user problems. In this case, the DAST [16] list is consulted or a savannah ticket [17] is created when is a tool bug or a GGUS ticket [18] made because of an external site error. The Grid use experience of the user support teams allow them to better identify the causes of problems and know the steps to follow in order to solve them.

The user support teams also organize meeting and tutorials as it happened in October 2009 when an ATLAS Distributed Data Analysis Tutorial has been organized before the data taking. Over the time, the three user support teams have been adapted to their user analysis needs. For instance, IFAE team is focused on the Tier-3 activities, IFIC in the GANGA use and UAM in pathena.

As an example, the user support team at IFIC attented more than 200 cases since 2009. This experience has been very productive for the execution of the physics analysis in Jet Substructure and helping other studies as top-antitop

resonances, tau lepton searches, SUSY, asymmetries and Higgs searches. The IFIC user support team provides to the ATLAS local users a twiki web page [19] which shows the basic knowledge required to start a distributed analysis using the IFIC infrastructure.

8 Conclusion

The Grid infrastructure of the ATLAS T2-ES federation is operating with high availability and reliability and is providing good quality services both for production of simulated events and for distributed data analysis, including the file transfer operations required in these activities, satisfying this way the requirements of the ATLAS Collaboration.

The availability and reliability of T2-ES is around 90% - 100%. The availabilities by activity are 83% for data analysis and 93% for simulated events production.

Concerning the transfer operations, T2-ES moved a total of around 28 million files, which corresponds to around 13 Petabytes of data, with a throughput reaching 243 MB/s of incoming data and 127 MB/s of outgoing data. The respective total efficiencies measured for these transfers are 88% and 94%.

The total data stored at T2-ES is a little over 2 PB, which represents around 5% of the data stored in all ATLAS Tier-2 sites.

Jobs of the two most important ATLAS activities have been studied: Analysis and Production. 4.7 million production jobs have arrived at this Tier-2 in the last two years with a high efficiency, 93%, which contributes 1.78% to all ATLAS. In Analysis, where 7.2 million jobs have run, contributing with 1.76% to all the sites, with an efficiency of 87.9%, higher than global average.

Finally, T2-ES has provided a group of experts in computing know-how to help the Spanish physicists from the three centres and part of them have collaborated in global ATLAS tasks as DAST. All these aspects put the Spanish ATLAS Tier-2 in a good position for the future challenges.

Acknowledgements

This work has been partly supported by MICINN, Spain (Proj. Ref. FPA2010-21919-C03-01,02,03).

References

1. E. Oliver *et al.*, Readiness of the ATLAS Spanish Federated Tier-2 for the Physics Analysis of the early collision events at the LHC, J. Phys. Conf. Series **219** (2010) 072046
2. Atlas Computing: Technical Design Report, CERN, ATL-SOFT-PROC-2012-016, May 2012
3. `https://twiki.cern.ch/twiki/bin/viewauth/Atlas/DDMSonarTests`

4. I. Ueda, ATLAS Distributed Computing Operations in the First Two Years of Data Taking, ATL-SOFT-PROC-2012-003, Mar (2012)
5. T. Maeno, K. De and S. Panitkin, PD2P : PanDA Dynamic Data Placement for ATLAS, CERN, 2005
6. G. Amoros *et al.*, Data analysis on the ATLAS Spanish Tier2, 4th Iberian Grid Infrastructure Conference Proceedings, Braga, Portugal, May 24-27, (2010)
7. http://www.dcache.org/manuals/dcache-whitepaper-light.pdf
8. http://twiki.ific.uv.es/twiki/bin/view/Atlas/LustreStoRM
9. http://puppetlabs.com/solutions/configuration-management/
10. http://www.rediris.es/
11. https://wiki.egi.eu/wiki/SAM
12. http://dashboard.cern.ch/atlas/
13. https://indico.cern.ch/materialDisplay.py?contribId=2&materialId=0&confId=20248
14. M. Kaci *et al.*, Contribution of the Iberian Grid Resources to the Production of Simulated Physics Events for the ATLAS Experiment, 4th Iberian Grid Infrastructure Conference Proceedings, Braga, Portugal, May 24-27, (2010)
15. M. Kaci *et al.*, Response of the Iberian Grid Computing Resources to the ATLAS activities during the LHC data-taking, 6th Iberian Grid Infrastructure Conference Proceedings, Lisbon, Portugal, November 7-9, (2012)
16. https://twiki.cern.ch/twiki/bin/viewauth/Atlas/AtlasDAST
17. https://savannah.cern.ch/
18. https://ggus.eu/pages/home.php
19. http://twiki.ific.uv.es/twiki/bin/view/Atlas/AtlasDataProcessingAtIFIC

The LHC Tier1 at PIC: experience from first LHC run and getting prepared for the next

J. Flix[§1,2], E. Acción[1,3], V. Acin[1,3], C. Acosta-Silva[1,3], G. Bernabeu[1,2¶] A. Bria[1,3‖], J. Casals[1,2], M. Caubet[1,2], R. Cruz[1,3], M. Delfino[1,4], X. Espinal[1,3**], E. Lanciotti[1,3**], F. López[1,2], F. Martinez[1,2††], V. Méndez[1,3], G. Merino[1,2‡‡], A. Pacheco Pages[1,3], A. Perez-Calero Yzquierdo[1,2], E. Planas[1,3], M.C. Porto[1,2], B. Rodríguez[1,3], and A. Sedov[1,3]

[1] Port d'Informació Científica (PIC), Universitat Autònoma de Barcelona, Bellaterra (Barcelona), Spain
[2] Also at Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas, CIEMAT, Madrid, Spain
[3] Also at Institut de Física d'Altes Energies, IFAE, Edifici Cn, Universitat Autònoma de Barcelona, Bellaterra (Barcelona), Spain
[4] Also at Universitat Autònoma de Barcelona, Department of Physics, Bellaterra (Barcelona), Spain

Abstract. The Large Hadron Collider (LHC) at CERN is the largest scientific instrument ever built. It started operating in November 2009 and its exploitation has generated a few hundreds of Petabytes of raw, simulated and processed data, from all of its detectors, since the stop of the successful first run (Run1) in February 2013. Managing this amount of data and enabling its analysis by thousands of physicists around the world is, and has been, a technological challenge. This is addressed by the largest scientific distributed computing infrastructure in the world: the Worldwide LHC Computing Grid (WLCG), adding up the computing resources from more than 170 centers in 34 countries. In the WLCG, the computing centers are functionally classified in Tiers. Spain contributes to the WLCG with one Tier1 center: Port d'Informació Científica (PIC), located in the campus of the Universitat Autònoma de Barcelona. PIC provides services to three of the LHC experiments: ATLAS, CMS and LHCb, and acts as a reference Tier1 for the Tier2 centers in Spain, Portugal, Chile and France. This contribution summarizes the operational experience of the center from first LHC run, how it helped in discovering the Higgs Boson and hunting for exciting new physics, the current activities, and how the center is preparing to cope with the restart of the LHC program by 2015.

¶ Currently at FNAL, Chicago, USA
‖ Currently at center de Regulació Genòmica, Barcelona, Spain
** Currently at CERN, Switzerland
†† Currently at Telefónica I+D, Barcelona, Spain
‡‡ Currently at Wisconsin IceCube Particle Astrophysics Center, USA
§ e-mail of corresponding author: jflix@pic.es

1 Introduction

The LHC started producing proton-proton collisions in November 2009 and its exploitation has generated around 200 Petabytes of raw, simulated and processed data, from all of its detectors, until the stop of the successful Run1 in February 2013. After a few months of commissioning at 0.9 TeV of collision energy, the conditions of Run1 happened at an intermediate center of mass energy of 7 TeV and 8 TeV, the maximum energy considered safe for the LHC in those conditions. In Run2, the LHC will run close to or at design energy, 13 or 14 TeV, with an expected luminosity even a bit higher to the original specifications (10^{34} cm^2s^{-1}).

There are six detectors installed in the LHC. Two of them are large general purpose detectors: ATLAS and CMS. There are also two medium sized detectors, which are indeed dedicated experiments: ALICE and LHCb. ALICE studies the quark-gluon plasma using high energy heavy ions collisions, while LHCb focus on indirect searches for new physics through precise measurement of CP violation and other observables in heavy quark decays. There are as well two much smaller experiments, TOTEM and LHCf, which focus on forward particles and are positioned near the CMS and ATLAS detectors, respectively.

So far, the four LHC experiments have generated about 15 Petabytes (PB) of raw data in 2010, 23 PB in 2011 and 27 PB in 2012. For the processing and analysis of this data, additional secondary and simulated data was generated, adding up to around 135 PB for this initial collisions period (see Figure 1). Taking into account the expected lifetime of the LHC, this places the LHC as the first scientific experiment ever in the Exabyte scale by 2020.

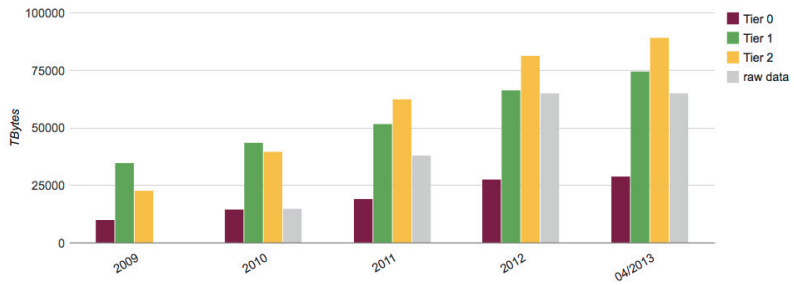


Fig. 1. Disk space deployed in WLCG in its tiered structure and the amount of raw data.

Collision data from LHC was anticipated to be produced at an unprecedented rate of PB per year, to be analysed by a Grid-based computer network infrastructure. The WLCG project started at CERN in 2002 with the objective to build and maintain the computing infrastructure to store, process and analyse

the data gathered by the LHC experiments. Today, the WLCG is a global collaboration of more than 170 computing centers in 34 countries which provide uniform access to computing resources through Grid services (see Refs. [1][2]).

The WLCG centers are organised in a tiered architecture (see Refs. [3][4][5][6]). Tier0 is at CERN. It receives the raw data from the experiments, records them on permanent mass storage and performs a first-pass reconstruction of the data. As of today, an extension of the Tier0 is being built in Hungary, connected to CERN via dedicated 2x100Gbps network. The raw and reconstructed data are distributed via a dedicated high-speed network to eleven Tier1 centers located in Canada (TRIUMF), France (CC-IN2P3), Germany (KIT), Italy (CNAF), Netherlands (NIKHEF/SARA), Nordic Countries (NDGF), Spain (PIC), Taipei (ASGC), UK (RAL) and USA (BNL and FNAL). Tier1 centers are in charge of providing permanent storage for the second copy of the raw data as well as performing massive reprocessing and data-intensive analysis. There are two new Tier1s being built at the moment, in Russia and Korea.

The processed data are normally distributed from the Tier1 centers to the Tier2 centers, where the analysis activity takes place. Tier2 centers are also in charge of generating Monte Carlo (MC) simulated data which each center then uploads to its associated Tier1 for permanent storage. In ATLAS and CMS, the Tier2 centers are designed to also be the places where scientific community do the main bulk of analysis. PIC Tier1 provides services to three of the LHC experiments, ATLAS, CMS and LHCb, accounting for 5% of the total Tier1 resources, acting also as the reference Tier1 for the Tier2 centers in Spain and Portugal, and two Tier2 sites located in Valparaiso (Chile/ATLAS) and Marseille (France/LHCb). There are three Tier2 federations in Spain, each of them serving one LHC experiment: the ATLAS Federation (CSIC-IFIC, IFAE and UAM), the CMS Federation (CIEMAT and CSIC-IFCA) and the LHCb Federation (UB and USC). In Portugal, the LIP Tier2 Federation is composed by three sites (LIP-Coimbra, LIP-Lisbon and NCG) and provides resources for the ATLAS and CMS experiments.

PIC has been actively participating in the WLCG project since its start. In the first phase, contributing to prototyping and testing of the Grid middle-ware and services that were being developed. Later, participating in the Service Challenges carried out by the experiments, testing campaigns aimed to progressively ramp-up the level of load on the infrastructure under as realistic as possible conditions, achieving breakthrough performances. PIC participated sucessfully in all of these tests and showed its readiness for the LHC data taking period.

2 PIC Tier1 Services

Tier1 centers are large and stable facilities. According to the LHC accounting of Run1, these 11 centers have provided about 45% of the total disk capacity of the distributed WLCG tiered system, and of about 65% of processing power. Due to this, and the criticality of their services, it is crucial that their capacity growth adjusts to the profile pledged agreed between CERN and the institutions

hosting Tier1 and Tier2 centers (MoU, see Ref. [7]). Figure 2 shows the evolution of the installed CPU capacity at PIC since January 2006 and it is compared to the official pledge in the MoU. For all of the offered resources, PIC has kept up with the pledged capacity ramp-up, modulus minor delays which had no impact in the service operation. Also, some of the running services where extended in warranty to help in the Higgs boson discovery and hunting for new physics (most of the Tier1s did it, indeed). As of today, the site has converged to the pledges of 2013. As compared to other Tier1s, the use of PIC in Run1 has been at the expected level of 5%, for all of the offered resources.

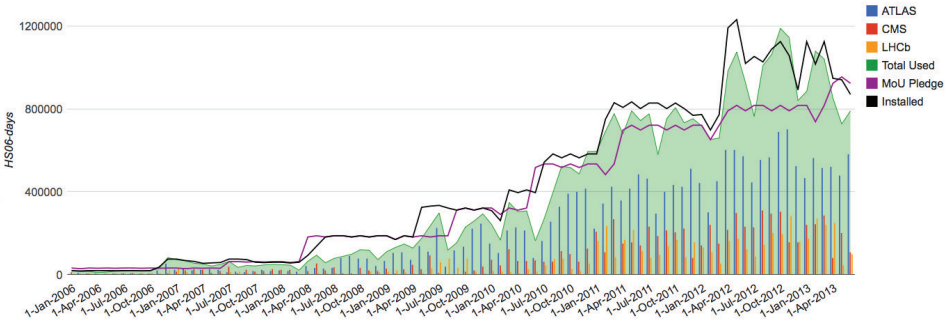


Fig. 2. CPU resources installed, used, and pldged at PIC, since January 2006.

As of today, the computing resources installed in PIC are of around 4000 cores (on ∼600 CPUs) managed by Torque/Maui. This corresponds to about 35000 HEP-SPEC06 (HS06, see Ref. [8]). The servers are typically two quad-core x86 CPUs, with at least 2GB RAM per core. Each of these nodes has two 10Gbps Ethernet interfaces which are then aggregated in switches and connected to the storage infrastructure. The main servers consist of Blades (HP) and Dual-Twins (Dell), recently migrated to Scientific Linux 6 OS, as required by the LHC experiments.

Millions of jobs run annually at PIC, with main customer being the Tier1, but also covering other needs (such as ATLAS Tier2, ATLAS Tier3, Cosmology projects,...). The LHC jobs CPU efficiency has increased along the years, being around 90% since the LHC start, as seen in Figure 3. This is in part possible by the improvement of the processing softwares used by the LHC experiments and the improved performance of the PIC infrastructure, which provides a good, stable and fast access to data stored at the site. Typically, of about of 45% of LHC jobs in Spain are processed in PIC Tier1.

The storage service at PIC is managed by the dCache (see Ref. [9]) and Enstore (see Ref. [10]) software stacks. The dCache software provides uniform

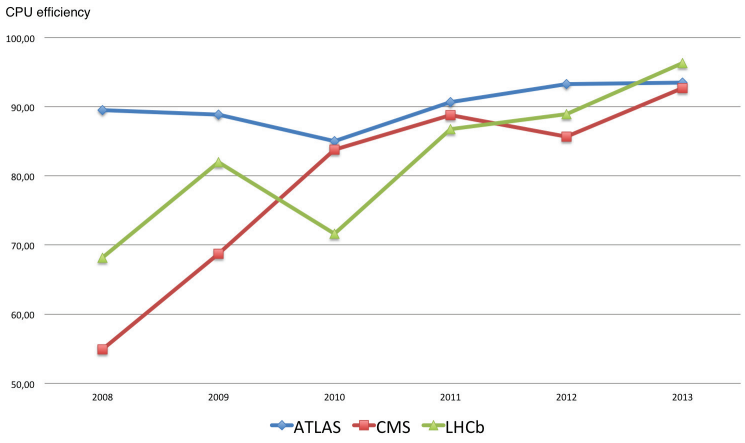


Fig. 3. Yearly average of CPU efficiency for ATLAS, CMS and LHCb jobs run at PIC.

and efficient access to the disk space provided by many file servers, and interfaces with Enstore for permanent storage in magnetic tape.

As of today, 5.5 PB of disk space is installed, by means of around 3000 hard disks of 1, 2 and 3 TBs, distributed on around 70 servers x86, each server connected by 4 aggregated 1Gbps Ethernet or one 10Gbps Ethernet, depending on the hardware. The servers brands comprises DataDirect, SGI, and SuperMicro.

The tape infrastucture at PIC currently consists of two automated tape libraries, Sun StorageTek 8500SL and IBM TS3500, together providing around 8500 tape slots. 8 PB of tape storage is managed by Enstore, with access to a total of 4.6 millions of files. The supported technologies are LTO-3 (read-only), LTO-4, LTO-5 and T10KC, and they contain 2%, 25%, 36% and 38% of the total data, respectively, in around 7000 tape cartridges. A total of 26 tape units are installed to read/write the data (16 LTO-4, 4 LTO-5 and 6 T10KC). Aggregated read/write rate has achieved hourly average rates peaking at 1 GB/s.

In order to ensure the needed quality and performance in the critical data transfers from the Tier0 to the Tier1 centers, the WLCG project deployed an Optical Private Network (OPN, see Ref. [12]) using GÉANT2 and the National Research and Educational Networks (NRENs). The OPN connects CERN and the Tier1 centers through point-to-point dedicated links of 10Gbps. The OPN link connecting PIC and CERN at 10Gbps is operative since 2007 and uses infrastructure from the Catalan NREN (Anella Científica). Many extended periods of high load and a notably high average load for a link of this capacity is observed in PIC, with a highly stable behaviour. A back-up line is as well provided, in case the primary goes down by an incident.

WLCG involves massive data transfers between Grid sites. Good performance links and reliable data transfer systems are a must. During Run1 the

monthly averaged value for incoming (outgoing) transfers to PIC has been of about 250 (400) MB/s, with hourly peaks exceeding 2 GB/s. Recently, a dedicated PerfSONAR-PS [11] installation to monitor the health of the network has been deployed, performing bandwidth and latency tests. PIC is also coordinating this deployment for all of the Iberian sites.

3 A reliable, high-capacity service

One of the main characteristics of Tier1 centers is that, beyond providing a very large storage and computing capacity, they must do so trough services that need to be extremely reliable. Being closely connected to the detectors data acquisition, a maximum time for unintended interruption of the services in a Tier1 is set to 4 consecutive hours, and a maximum degradation time forf Tier0 to Tier1 data acceptance of 6 consecutive hours. All of the critical services in a Tier1 typically operate in 365x24x7 mode.

Being service quality and stability one of the cornerstones of the project, they are closely tracked by monitoring two metrics provided by the SAM monitoring framework: site Availability and Reliability. These are built from dozens of sensors which hourly probe all of the site Grid services, which ensures peer pressure and guarantees that the reliability of WLCG service keeps improving.

Figure 4 shows the monthly Reliability results for PIC since May 2006. It can be seen that PIC Reliability is almost always above the target and the Tier0/Tier1 average. Worth to mention that PIC needs to have an expert contact person on site (the liaison), communicating and coordinating priorities with the experiments, and resolving operational problems. This helps PIC being at top of reliability and stability levels.

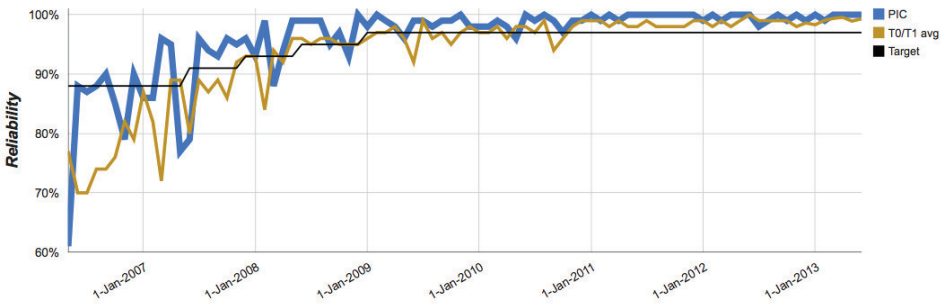


Fig. 4. Evolution of site Reliability for Tier0 and Tier1 centers since May 2006. The results for PIC Tier1 (blue) are compared to average from other centers (yellow) and to the internal target set by the project (black)

4 ATLAS activities in Run1

During the LHC Run1 large amount of data was collected by the ATLAS experiment. In the year 2011 more than 100 times luminosity equivalent data was collected than in 2010, while in 2012 it was about 4 times more.

In CPU usage, PIC contributed to ATLAS to about 5.6% in 2011 and 4.7% in 2012, which corresponded to 4.4 millions and 6.5 millions of HS06-days of CPU usage, respectively. In disk occupancy, the contribution was of 6.4% in 2011 and 6.3% in 2012, which correspondeds to 1.6 PB and 2.0 PB of disk occupancy at the end of the corresponding years. Concerning tape usage, of about 5.3% in 2011 and 5.4% in 2012, which corresponded to 978 TB and 1.7 PB of tape occupancy at the end of corresponding years.

During Run1 period many improvements in ATLAS computing model were introduced, such as changes in the number of replicas and types of data formats at the sites, which improved the situation with disk space management. Additionally, the ratio of ATLAS production and user analysis jobs at Tier1s was set to be 95% and 5%, respectively.

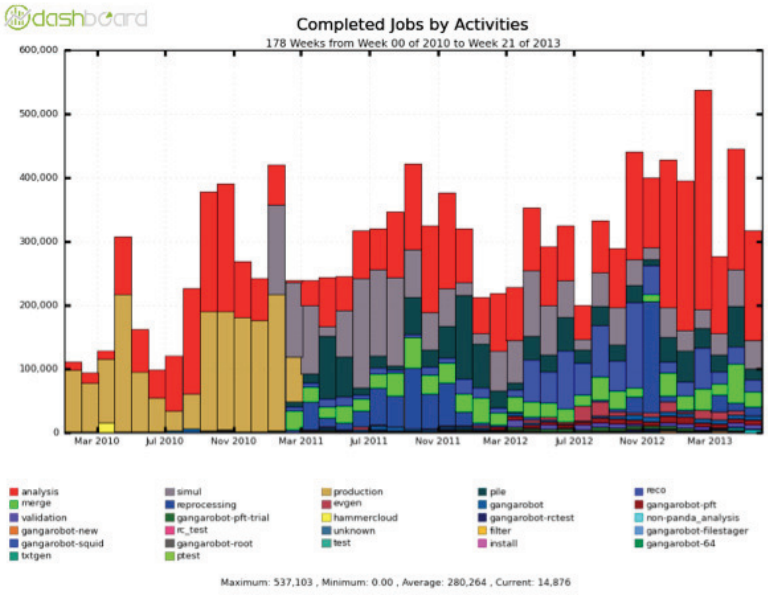


Fig. 5. ATLAS completed jobs at PIC, per activity.

ATLAS adopted the *xrootd* protocol to access data during Run1. The benefit of this protocol is that allows reading parts of a file and it allows for remote access to files as well. This improves job efficiencies and remote data accesses.

Indeed, ATLAS xrootd storage resources joint into a federation (FAX), so users do not have to know the exact location of data file, but can ask through central access point and get automatic location of the file and use it as if it was locally available. PIC deployed and habilitated xrootd access for ATLAS and joined the federation, exposing its data and helping the users to access to the files placed in PIC disk from remote processing nodes.

Original ATLAS data distribution schema was star-like with start center at Tier0 connected to Tier1s only. Each Tier1 center was connected only to Tier2s and Tier3s from the same cloud. Validation of network stability allowed to improve such schema adding more connection between Tier1s and good outside the cloud Tier2s that are in the so-called "Tier 2D" list. Such Tier2s are regularly checked for network transfer performances.

In the middle of Run1, another improvement in the ATLAS structure consisted in the consolidation of local services at CERN. By this mean, ATLAS specific services running in PIC were migrated to CERN: Frontier services in 2011 and ATLAS LFC in 2012. Another improvement was introduced with the use of CVMFS (CERN VM file system) as a standard solution for software releases and software deployments at the sites. PIC deployed CVMFS in 2011 and benefited from ease of new release installation/testing as well as from increased number of available software releases. The PIC squid services were adapted to cope with increased load due to the CVMFS use (ex. as of june 2013 5 squid machine were installed, and bigger resources were made available for ATLAS in each squid server). Since April 2013 all software validation is done through CVMFS and ATLAS has completely deprecated software installations in shared disk spaces at all of the sites.

The use of MultiProcessing (MP) jobs allows to use efficiently new multicore machines and reduce memory per core requirement which leaded to cheaper computer buying options. PIC has successfully implemented a MP Panda queue in 2012, which is designed for tasks that use whole node (all cores of the node) to run one job, and ATLAS has started making tests at the site to validate their software and check for improvements.

ATLAS-PIC liaisons have as well played, and play, important roles in ATLAS, such as co-coordination of ADC (ATLAS Distributed Computing) shifts, at all of the levels (from trainees up to experts), design of ATLAS monitoring tools, and being active within the ATLAS Cloud computing group.

5 CMS activities in Run1

During the LHC Run1 PIC has been one of the most stable and efficient sites participating in the Tier1 structure associated to CMS. Computing resources, as well as data transfers and storage have been used in agreement with the pledges providing a high quality service to the experiment.

On average, more than 500 jobs have been running simultaneously in PIC, at any time during Run1. Monthly averages peaked at values of almost 1800 jobs. More than 113,000 jobs have been run each month on average, with peaks of

over 300,000 jobs per month. Around 90% of the jobs are 'production' jobs, that is, dedicated to processing of data or generation of simulated events, while the rest have been used mainly to test and continuously monitor the site status and performance.

In comparison with other Tier1s, the workload at PIC has been close to its expected value of 5%. Monthly averages for job success rate when running at PIC has been consistently around 80% and above, one of the best proportions among the CMS Tier1s.

Critical data transfers from the Tier0 at CERN to PIC have been extremely stable in Run1. Transfer quality, measured as the percentage of successfully transferred files, has been excellent, in general above 95%, once LHC operation started. Almost 1.3 PB of data has been received, with maximum weekly averages of ∼55 MB/s. Overall, PIC has received 2.9 PB of data from all CMS sites and sent 4.6 PB to them. Exchange of data with Iberian sites has amounted to 450 TB received and 500 TB sent.

CMS has stored a total of 39 PB of data distributed between its associated Tier1 sites. Being a Tier1 site, PIC has received custodial copies of data and MC to be archived in tape. The total amount of data stored in tape is of 1.837 PB by June 2013, with 760 TB of data and 1077 TB of MC. The distribution of archival data between sites, shows that PIC is being used according to its pledged share of 5%.

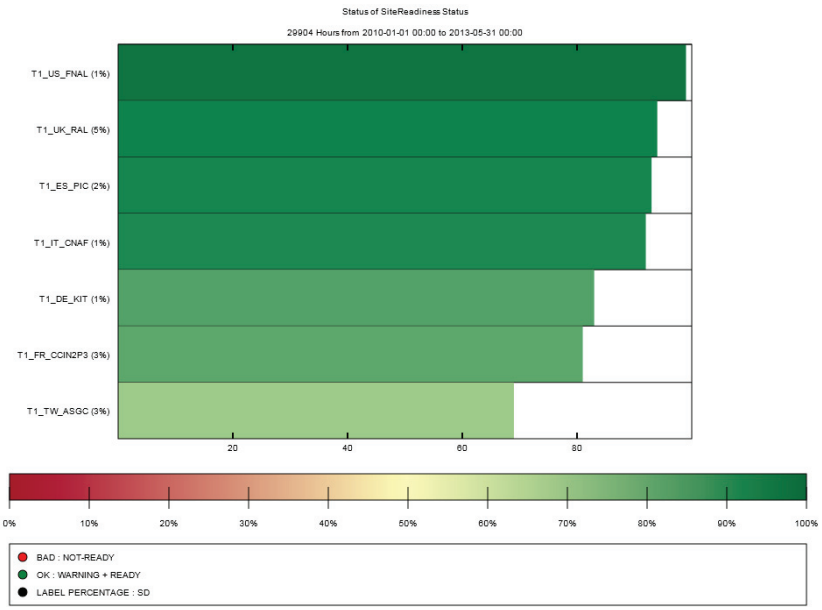


Fig. 6. Tier1 sites ranking according to Site Readiness metric from 2010 to 05/2013.

CMS Grid structure has implemented a set of tests evaluating the availability and service quality for all its computing sites. The results of these tests are combined into a single metric called Site Readiness (Ref. [13]), which is taken into account to prioritize the use of those resources locate at the most stable sites. According to this composite value, PIC was ready 93% of the time in Run1, being on scheduled downtime for maintenance or upgrades 2% of the time, thus being not ready only 5% of the time. Comparing PIC results to other sites (Figure 6), shows that is has been one of the most reliable sites supporting CMS, despite its relatively modest size.

CMS-PIC liaisons have as well played, and play, important roles in CMS, such as deployment of the CMS Site Readiness, co-coordinating the CMS Facilities Operations and Infrastructure working group, succeeded by the CMS Computing Operations working group, acting as a representative in the CMS Computing Management and CMS Resource Board, serving as Computing Run Coordinator, expert on duty reporting to the daily operations meeting during data taking periods, representative in WLCG Technical Evolution Groups and WLCG Operations, co-coordination of CMS Storage Federation activities and the evolution of CMS computing towards the use of multicore jobs.

6 LHCb activities in Run1

In the LHCb computing model, Tier1 main production activities are *Reconstruction*, *Striping* & *Merging* of the raw data, as well as planned *Reprocessing* of the previous data with new physics considerations. These activities convert the original raw data files to files ready to be used by physicists, namely merged DSTs.

Tier1s are also considered for MC simulation campaigns when the main activities allow for it. Additionally, user jobs can be processed at the Tier1s with low priority. Tier2s process MC jobs, user jobs and help in reprocessing campaigns providing CPU to compute the data which is in the permanent storage of the associated Tier1. Tier2s do not poses permanent disk storage, and their results are uploaded to Tier1s.

In general, the LHCb computing model has been successfully addressed by the DIRAC *interware* [14], a flexible community solution which has demonstrated to be adaptable to dynamic scenario constrains and multiple requirements.

During the LHC Run1, LHCb has normally ran over its specifications, in terms of event sizes and processing times. The main reason, besides of the yearly increase of LHC luminosity, was the possible CP violation in charm decays, observed at the end of 2011. For this reason, the HLT was re-adjusted in charm event filter, producing new processing requirements and larger data. This impacted the usage of the Tier1s, in particular with the need of higher storage and CPU capacities to successfuly process and store the data. Three sites were associated to PIC to help in the reprocessing campaigns: USC (Santiago de Compostela), CPPM (Marseille) and UB (Barcelona).

Under these conditions, in autumn of 2011, the first data *Reprocessing* pass was successfully done in PIC. In addition, the regular *Reconstruction*, *Striping* and *Merging* were as well done, producing more data than previously anticipated. In 2011, about 260 TB of raw data was processed obtaining 120 TB of merged DSTs, which were replicated to the rest of Tier1s.

In 2012, more intensive and larger reprocessing campaigns were performed. This impacted the MC production at Tier1s, which was reduced. The MC productions were re-allocated using Tier2 resources from WLCG and additional resources from the EGI infrastructure in opportunistic mode. During the spring data *Reprocessing* campaign of 2012 in PIC, and due to the huge demand of data archived in tape, the pre-staging from tape to disk mechanism needed to be improved. This contribution was positively applied to all of the Tier1s in the autumn data *Reprocessing*. During this year, the computing model was redefined in depth to fit the new luminosity and charm physics. A new disk token scheme was deployed in Tier1s, resulting in many internal data moves (of about 660 TBs worth of data in PIC). During 2012, 800TB of LHCb data entered PIC, mainly from Tier0, and 350 TB of data was transferred, mainly associated with the merged DST replicas, to others Tiers1.

Figure 7 shows an efficiency metric of average number of jobs for each HS06 of the different Tiers1. HS06 is a CPU benchmark, while the job processing is also involving the rest of the site resources, plus overheads. Bigger jobs per CPU metric indicates a better use of all the involved resources. As can be seen in the picture, PIC shows a good value. The figure takes into account all of the computing activities, including processing activities and also the MC and user analysis during Run1.

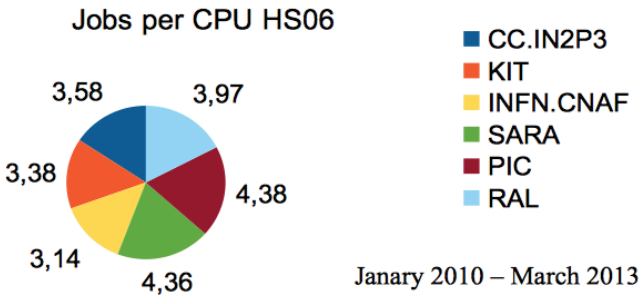


Fig. 7. LHCb job efficiency metric for all of the Tier1s.

PIC has about the 6% of the installed capacity of the total resources at LHCb Tier1s. Despite being a small site, PIC efficiency metrics place the site around the top of the LHCb Tier1s, with the help of the personnel know-how and management expertise.

LHCb-PIC liaisons have as well played, and play, important roles in LHCb, such as R&D and integration of DIRAC-LHCb core for Cloud Computing, and developing a VM manager tool (VMDIRAC), which works on federated clouds of OpenNebula, OpenStack, CloudStack and EC2 Amazon.

7 Getting prepared for the LHC restart

Since February 2013, data taking is stopped. However, the experiments are eventually reprocessing all of the collected data, parked data in Tier0 (CMS case), and producing new MC files with the expected Run2 conditions. Additionally, new functionalities and tools are being tested and integrated into the system.

In 2015, LHC experiments will restart data taking at increased collision energy and trigger rates. LHC computing needs to prepare for, at least, twice the amount of data. A number of improvements have been proposed, which are currently under development. Firstly, advance towards generic tools used by the LHC VOs (job submission, network and storage monitoring, etc). Integrating data transfers and storage resources (e.g. CMS AAA project, or storage federations) benefiting from increased network rates and including new transfer protocols, e.g. xrootd/http. Once storage federations are setup, jobs can run at Grid site A with remote data input from site B and output at site C. This will decouple where data is and where jobs run. Cloud Computing and opportunistic resources is as well a key point: integrate Grid infrastructure providing baseline resources with cloud resources on demand to absorb peaks with the use of commercial clouds, HLT farms or even SCC sites. Parallel Computing and multi-core job might be a must. Increased luminosity and pileup require processing events with improved memory management. This is provided with multi-thread applications running on multi-core CPUs. PIC is involved in many of these experiment tasks, either participating or coordinating them.

Taking into account hardware retirements, the PIC resources will continue growing to cope with the experiments requirements, being full ready to enter into LHC Run2. Among many ongoing tasks, it is worth to mention that an extension of a private network to Tier2 and Tier3 sites is being built, the so called LHCONE, bringing those sites into a more robust framework. PIC is coordination the Iberian deployment of this new setup. The current network is based in IPv4, and PIC is extensively testing all of the services in IPv6, to migrate to the new schema by 2015.

Acknowledgements

The Port d'Informació Científica (PIC) is maintained through a collaboration between the Generalitat de Catalunya, CIEMAT, IFAE and the Universitat Autònoma de Barcelona. This work was supported in part by grants FPA2007-66152-C02-00 and FPA2010-21816-C02-00 from the Ministerio de Ciencia e Innovación, Spain. Additional support was provided by the EU 7th Framework Programme INFRA-2007-1.2.3: e-Science Grid infrastructures Grant Agreement

Number 222667, Enabling Grids for e-Science (EGEE) project and INFRA-2010-1.2.1: Distributed computing infrastructure Contract Number RI-261323 (EGI-InSPIRE).

The PIC team would like to thank WLCG and all of the Computing Teams of ATLAS, CMS and LHCb for all of the help offered when needed.

References

1. LHC Computing Grid Technical Design Report, CERN-LHCC-2005-024, 20 June 2005.
2. "Computing for the Large Hadron Collider", Ian Bird, "Annual Review of Nuclear and Particle Science", Vol. 61: 99-118, November 2011.
3. ALICE Collaboration, ALICE Technical Design Report of the Computing, CERN-LHCC 2005-018, 2005.
4. ATLAS Collaboration, ATLAS Computing Technical Design Report, ATLAS TDR017, CERN-LHCC 2005-022, 2005.
5. CMS Collaboration, CMS Computing Project: Technical design report, CERN-LHCC 2005-023, 2005.
6. LHCb Collaboration, LHCb Computing Technical Design Report, CERN-LHCC 2005-019, 2005.
7. Memorandum of Understanding for Collaboration in the Deployment and Exploitation of the Worldwide LHC Computing Grid, http://wlcg.web.cern.ch/documents-reference
8. https://hepix.caspur.it/benchmarks/doku.php.
9. http://www.dcache.org.
10. J. Bakken et al., Enstore Technical Design Document, http://www-ccf.fnal.gov/enstore/design.html.
11. http://psps.perfsonar.net/
12. LHC Optical Private Network, http://lhcopn.cern.ch.
13. "Monitoring the Readiness and Utilization of the Distributed CMS Computing Facilities", J Flix et al 2011 J. Phys.: Conf. Ser. 331 072020 doi:10.1088/1742-6596/331/7/072020.
14. Status of the DIRAC project, A Casajus et al 2012 J. Phys.: Conf. Ser. 396 032107 doi:10.1088/1742-6596/396/3/032107

Author Index

IBERGRID

7th IBERIAN GRID INFRASTRUCTURE CONFERENCE PROCEEDINGS

Madrid, SPAIN

PROMOTED BY

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA EDUCAÇÃO E CIÊNCIA

LIP

GOBIERNO DE ESPAÑA MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD

CSIC Consejo Superior de Investigaciones Científicas

ES-NGI

ISBN 978-84-9048-110-3

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

EDITORIAL