IBERGRID 2014

8th Iberian Grid Infrastructure Conference

UNIVERSITY OF AVEIRO
AVEIRO · PORTUGAL

8 › 10 SEPTEMBER 2014

EDITORS
Ilídio Oliveira
Jorge Gomes
Isabel Campos
Ignacio Blanquer

IBERGRID

8th Iberian Grid Infrastructure Conference Proceedings

Aveiro, Portugal • September 8-10, 2014

Editors
Ilídio Oliveira • Jorge Gomes • Isabel Campos • Ignacio Blanquer

Editorial Universitat Politècnica de València

Colección Congresos

II IBERGRID

The contents of this publication have been subject to revision by the Scientific Commitee.

First edition, 2014.

EDITORS

Ilídio Oliveira
Institute of Eletronics and Informatics Engineering of Aveiro (IEETA)
Dep. of Eletronics, Telecomm. and Informatics, University of Aveiro

Jorge Gomes
Laboratório de Instrumentação e Física Experimental de Partículas

Isabel Campos Plasencia
Instituto de Física de Cantabria – CSIC, Santander

Ignacio Blanquer
Universitat Politècnica de València

www.lalibreria.upv.es / Ref.: 6177_01_01_01

Cover design: Vítor Teixeira, Universidade de Aveiro.

ISBN: 978-84-9048-246-9 (versión impresa)

CONTENTS

IV IBERGRID

IBERGRID Biomedical image analysis workshop

MESSAGE FROM THE CONFERENCE CHAIRS

The IBERGRID 2014 is the 8th edition of the Iberian Grid infrastructure conference that is being organized since 2007 in the context of bilateral agreements between the governments of Portugal and Spain. The IBERGRID conferences are a forum for researchers, application developers and infrastructure managers to share experiences, new ideas and research results. The conferences have been instrumental to enable the creation of an Iberian distributed computing infrastructure. The IBERGRID infrastructure became reality in 2010, when the grid resource centres from both countries joined forces towards a common participation in the European Grid Initiative (EGI). Since then, the conferences are also an opportunity for technical discussions regarding the infrastructure and its evolution.

This year's event has a strong focus on cloud computing and related technologies. It is a reflection of the strong interest and ongoing activities in this domain where IBERGRID is playing an important role in shaping the federation of scientific clouds in Europe. In fact 2014 is already marked by the launch of the EGI federated cloud where the IBERGRID community is highly visible both as resource and technology provider. The increased weight of cloud computing is also visible in the conference programme through the one day workshop and training dedicated to the technical aspects of cloud deployment, organized in partnership with FCCN.

Cloud computing is clearly becoming the preferred solution for new scientific projects. However, grid computing is still the service through which IBERGRID delivers most of its capacity to the user communities. Between mid 2013 and mid 2014, IBERGRID delivered more than 117,000,000 processing hours, and executed more than 28,000,000 jobs, supporting a wide range of scientific communities and domains. The delivered normalized processing capacity increased by a factor of 12% in comparison with the previous period. The infrastructure also delivered 7% of the whole European Grid Initiative (EGI) worldwide processing time. Furthermore, the IBERGRID partners are currently delivering key services to the EGI global infrastructure such as staged rollout, 1st and 2nd level support, accounting and metric portals.

The IBERGRID infrastructure is supporting flagship scientific projects such as ALICE, AUGER, ATLAS, CMS, CTA, LHCb and others. A stronger engagement with ESFRI user communities is also emerging driven by new opportunities made possible through the Horizon 2020 programme associated to the evolution towards cloud computing. In this context IBERGRID is already deeply committed to collaborate with the Lifewatch ESFRI in supporting its community, services and applications.

The IBERGRID infrastructure is the result of an excellent Iberian collaboration in science, which is having a significant role in the European e-infrastructures landscape.

Finally, we would like to express our sincere thanks to all IBERGRID collaborators, invited speakers, participants, organizations and sponsors that contributed to the IBERGRID 2014 event.

Welcome to Aveiro!

The Conference Chairs,

Ilídio Oliveira, Jorge Gomes, Isabel Campos, Ignacio Blanquer

CHAIRS

Ilídio Oliveira
Institute of Eletronics and Informatics Engineering of Aveiro (IEETA)
Dep. of Eletronics, Telecomm. and Informatics, University of Aveiro

Jorge Gomes
Laboratório de Instrumentação e Física Experimental de Partículas

Isabel Campos Plasencia
Instituto de Física de Cantabria – CSIC, Santander

Ignacio Blanquer
Universitat Politècnica de València

SCIENTIFIC ADVISORY COMMITTEE

General track:

Alvaro Fernández
Andrés Tomás
Andreu Pacheco
Antonio Fuentes Bermejo
António Pina
Carlos de Alfonso
Carlos Fernández
Carlos Redondo
Damià Segrelles
Diogo Gomes
Eduardo Huedo
Enol Fernández
Erik Torres
Francisco Castejón
Gaspar Barreira
Germán Moltó
Gonçalo Borges
Helmut Wolters
Ignacio López
Ignácio Blanquer
Ilídio C Oliveira
Isabel Campos
Javier López Cacheiro
Javier G. Tobío
Jesus Marco
João Pina
João Nuno Ferreira
João Paulo Barraca
Jorge Gomes
Jose Salt
Josep Flix
Lígia Ribeiro
Mario David
Miguel Caballer
Miguel Cárdenas
Rosa M. Badia
Tomas de Miguel
Victor Castelo

Biomedical image analysis workshop:

Damiàn Segrelles Quilis
Ignacio Blanquer
Ilídio C Oliveira
Miguel A. Guevara
Raúl Ramos-Pollán
Erik Torres
Frederico Valente

SPONSORS

With the support of:
Institute of Eletronics and Informatics Engineering of Aveiro (IEETA)
University of Aveiro
Laboratório de Instrumentação e Física Experimental de Partículas (LIP)
Instituto de Física de Cantabria – CSIC, Santander
Universitat Politècnica de València

Industry Sponsors:
HP Portugal.

Session 1: eInfrastructure achievements and evolution

The role of IBERGRID in the Federated Cloud of EGI

Álvaro López García, Pablo Orviz, Fernando Aguilar, Enol Fernández-del-Castillo **, Isabel Campos, Jesús Marco

Advanced Computing and e-Science Group
Instituto de Física de Cantabria, CSIC - UC, Spain
{aloga,orviz,aguilarf,enolfc,iscampos,marco}@ifca.unican.es

Abstract. EGI has recently launched a Federated Cloud service that integrates academic private clouds across Europe. The service aims to empower researchers with a flexible Infrastructure as a Service cloud. In this article we describe the motivation, architecture and design of the EGI Federated Cloud, the services it offers to the users, and possible evolution paths.

1 Introduction

Cloud computing is replacing many classical ICT services with considerable advantages to final users which include pay-per-use and elasticity. Organizations operating computing centres are adopting the several Infrastructure as a Service (IaaS) cloud solutions available in the market to manage their resources. These solutions are typically based on private clouds directly installed at the computing centre level, either open source (like OpenNebula [1] or OpenStack [2]) or closed solutions (such as VMware [3]).

On the other hand, in the scientific domain, the provisioning of IT services in cloud mode either using private or public cloud solutions, seems to be a source of potential concern. There is still some confusion regarding the very nature of cloud computing in research centres. In some cases it is difficult to distinguish a real new service, with added value for researchers, from a re-branding of earlier technologies coming from distributed computing Grids.

In the EGI ecosystem we encounter many of these sensibilities. The resource providers in EGI range from computing centres dedicated to deploy services for the long tail of users, to research centres oriented to specific domains, and therefore have the need to deploy specific services.

EGI took the strategic decision of exploring service provisioning through the cloud model. This decision was fostered by the need to reinforce the role and engagement of well structured international user communities. We foresee in the upcoming years a landscape of computing services produced in co-development between expert users and technology providers. For this strategy to succeed it is essential to guarantee that the infrastructure will be capable of supporting the flexible deployment of those services.

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

Therefore, as a first step to support this strategy towards direct engagement with user communities, EGI has deployed the EGI Federated Cloud (FedCloud). FedCloud is the result of the collaborative work of several software teams from different NGIs, to develop the next generation of services aimed at defining and implementing distributed cloud computing and storage infrastructures.

At its launch, the Federated Cloud pools resources and the expertise of 19 countries around Europe: Bulgaria, Croatia, Czech Republic, France, Germany, Greece, Hungary, Israel, Italy, Latvia, the Republic of Macedonia, the Netherlands, Poland, Portugal, Slovakia, Spain, Sweden, Turkey, United Kingdom.

The current EGI Federated Cloud provides advanced users with a flexible, scalable, standards-based cloud infrastructure. The service includes the necessary support to ensure that researchers benefit fully from the infrastructure. In particular it provides operating system images already available into the EGI Applications Database, and the creation of virtual machine images containing the scientific software and other required components. The users can instantiate virtual machines on the EGI cloud by using a stand-alone command-line client, which interacts with the the standard OCCI [4] interface.

The EGI FedCloud is a production infrastructure targeting advanced users, i.e. it is at the level of an Infrastructure as a Service (IaaS). Clearly in order to reach its full potential major aspects that have not been yet developed need to be realized. The focus of research in this field needs to be put in the development of suitable higher level platforms and software services, able to interact with distributed cloud infrastructures. Such services would hide the complexity of the infrastructure in such a way that the user does not need to deal with the construction and deployment of virtual machines, but rather directly with the composition of the services needed, via an abstraction layer, in the form of Platform as a Service (PaaS) or Software as a Service (SaaS).

As will be explained through this article, the IBERGRID partners have participated actively in the middleware development and deployment of the first production release of the FedCloud. At the level of proofs of concept the IBERGRID teams have been involved in the implementation of use cases related to area of environment and biodiversity (under the umbrella of the ESFRI Lifewatch). Several examples will be presented during the IBERGRID 2014 conference as well.

All through the design phase, EGI has been working with many research communities to gather requirements, advice and test many aspects of the infrastructure. The communities involved are the WeNMR project (structural biology), the European Space Agency (satellite image processing), Peachnote (musicology), Lifewatch (biodiversity) and EISCAT-3D (astronomy) among others. The launch of the Federated Cloud is only the beginning, more resource providers and users both from the private and public sectors are expected to join, contributing to expand and improve the infrastructure.

The layout of this article is as follows. First we will describe the architecture and design considerations that have led to FedCloud. Second we will describe the services currently available to the users, and the methods to access them. In doing so, we will also highlight the contributions of the developers from the IBERGRID team, and the current status of deployment in the Iberian area.

2 Architecture and Design of EGI Federated Cloud

The EGI Federated Cloud model is based on the delivery of a well-defined set of services that each participant Resource Provider (RP) must fulfill in order to become part of the federation. These services are defined around capabilities, not technologies. Hence the EGI Federated Cloud infrastructure does not impose any technology solution and lets Resource Providers choose the middleware that suits best their particular needs, as long as they are able to provide the interfaces and services required for federation.

The FedCloud architecture is designed considering the requirements of ten different scenarios: (1) VM Management (2) Managing my own data (3) Integrating multiple Resource Providers (4) Accounting across Resource Providers (5) Reliability/Availability of Resource Providers (6) VM/Resource state change notification (7) Authentication and Authorization across Resource Providers (8) VM images across Resource Providers (9) Brokering, and (10) Contextualization.

The proposed architecture (in Figure 1) derives from the analysis of the requirements of each scenario and the possible technical solutions to meet them. The EGI Federated Cloud is built as a layer on top of a Cloud Management System that provides a set of standardized cloud interfaces and integrates these services with the components of the EGI Core Infrastructure. The federation of cloud resources is achieved via the following infrastructure interfaces and components.

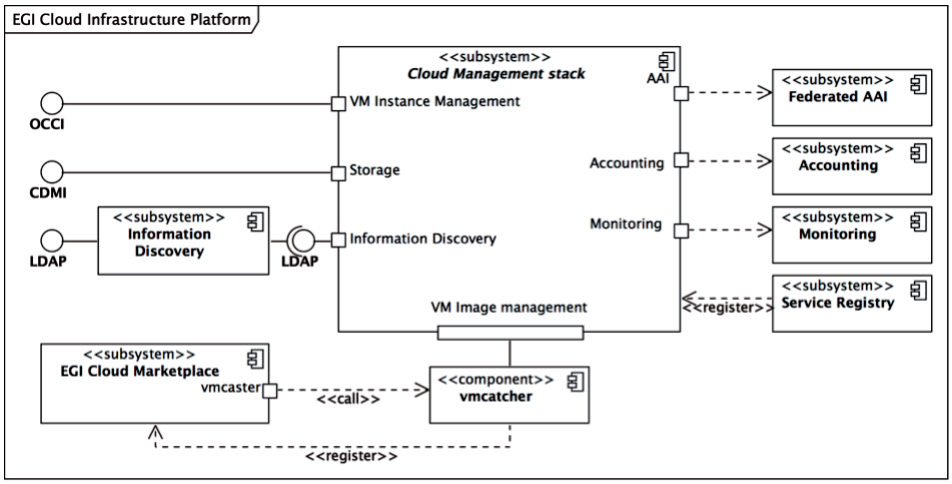


Fig. 1. EGI Federated Cloud Architecture

Interfaces. Resource Providers must support three standard based interfaces which enable the integration of local resources into the federated cloud, namely: Open Cloud Computing Interface (OCCI), Cloud Data Management Interface (CDMI) [5] and GlueSchema v2 [6]. These interfaces are detailed in the next section.

Authentication and Authorization. Users must be able to authenticate against the Resource Providers using their VOMS [7] certificates as they would for any other resources of the EGI platform.

Accounting. Resource Providers must account for resource usage and send their records to a EGI central Cloud Accounting repository using the APEL SSM infrastructure [8].

Monitoring. The SAM (Service Availability Monitoring) [9] includes probes for monitoring the proper functionality and availability of the interfaces required to join the Federated Cloud.

Appliance Repository. EGI's AppDB [10] stores and publishes VM image related metadata and VMcatcher [11] manages and publishes images to the Resource Providers.

3 FedCloud User Services

EGI Federated Cloud provide services to users through a set of standard interfaces. These are described in the following sections.

3.1 OCCI

The Open Cloud Computing Interface (OCCI) is a RESTful Protocol and API designed to facilitate interoperable access to cloud-based resources across multiple resource providers and heterogeneous environments. The formal specification is maintained and actively worked on by the Open Grid Forum (OGF) [12]. It is a community-lead interface providing a wide range of management tasks for remote IaaS, PaaS and SaaS.

OCCI consists on: a formal definition of a core model [13]; a HTTP rendering [14] that specifies how OCCI can be used with a RESTful API; and an extension of the core model for the IaaS domain [15] (see Figure 2) that defines additional type of resources (storage, network, compute, etc.), their attributes and the actions that can be taken on each resource type. These resources can be associated through links, e.g. a network resource can be linked to a compute resource resulting in a new network interface attached to a existing VM. OCCI also introduces the concept of *Mixins*, which provide the means to extend the functionality and define provider-specific features.

There are several OCCI implementations for the Cloud management systems available in the FedCloud:

rOCCI-server. rOCCI [16] is a implementation of an OCCI client and server in Ruby. The server is designed to support any cloud management platforms as backend and currently supports OpenNebula.

OCCI-OS. OCCI-OS [17] extends the OpenStack Compute service to support the OCCI interface with several extensions (*mixins*) for specific OpenStack features.

snf-occi. Synnefo [18] also includes an OCCI implementation on top of its proprietary interface.

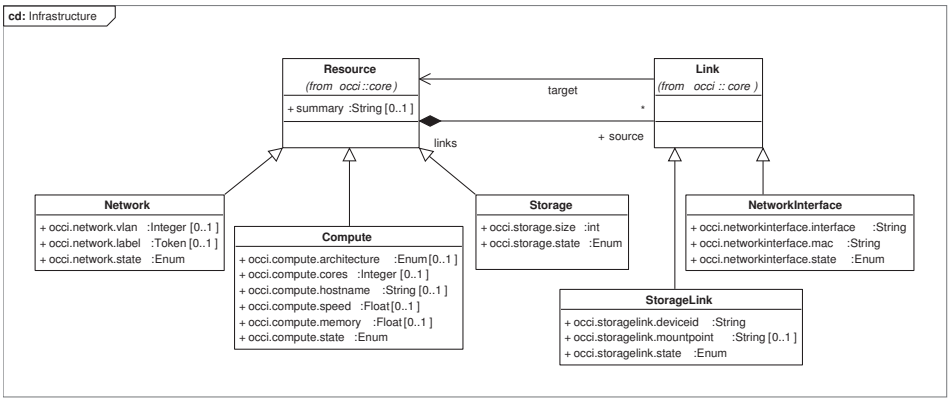


Fig. 2. OCCI infrastructure model (taken from OCCI Infrastructure v1.1)

Contextualization Contextualization is the process of installing, configuring and preparing software upon boot time on a pre-defined virtual machine image (e.g. setting the hostname, IP addresses, SSH authorized keys, starting services, installing applications, etc.). OCCI specification does not support the contextualization of VMs, although it is a requirement for most use cases identified in FedCloud.

We have proposed the use of a new OCCI *mixin* with an attribute to hold arbitrary data that is passed to the VM on creation. This *mixin* is supported in all the OCCI implementations available in FedCloud.

The *mixin* does not specify how the data will be made available at the VM and each cloud management framework provides its own mechanism. FedCloud recommends the use of cloud-init [19] for handling the data at the VMs. Cloud-init abstracts the specific ways for handling the contextualization information and it's widely available in most Operating Systems and IaaS cloud platforms.

3.2 CDMI

The Cloud Data Management Interface (CDMI) defines a RESTful interface for operations on data elements in the cloud proposed by the Storage Networking Industry Association (SNIA) [20].

CDMI is based on the concept of objects, which vary in terms of supported operations and metadata schema. CDMI provides operations for create, retrieve, update and delete these objects. Besides the basic operations, clients can discover the capabilities of storage and provides administrative and management operations of the system.

In EGI Federated Cloud CDMI is the standard interface for operating with blob data. It is supported through native interfaces of the cloud stack or through stoxy [21], a CDMI proxy developed in FedCloud, that includes support to the authentication and authorization of EGI.

3.3 VM Image Management

The VM image management subsystem provides users with the means for efficiently manage and distribute their VM images across the FedCloud infrastructure. The EGI Applications Database (AppDB) service and two command-line tools, the "vmcaster" and the "vmcatcher" provide the main functionality for this subsystem. The EGI Applications Database is a VM Image marketplace that holds and populates Images related metadata as provided by the user communities, and provides the necessary mechanisms for automatic distribution of the Images to the sites. "Vmcatcher" is the command line tool responsible for subscribing to the lists produced by the AppDB, and finally, "vmcaster" is capable of publishing, ready to be used image lists, either directly to the AppDB or to to any other 3rd-party web server.

Through the EGI Applications Database service (either using its API or using its graphical web interface) VO Managers can compose VM Image lists with images that are considered of interests by the VO and publishes them. The Resources Providers then subscribe to changes in the VM lists by regularly downloading the list from AppDB and comparing it against local copies. New and updated VM images are downloaded and included into the local cloud management framework, so VO users can instantiate them.

3.4 GlueSchema

The EGI Information System is based on the Lightweight Directory Access Protocol (LDAP) an Internet Engineering Task Force standard for distributed directory information services. The EGI FedCloud integrates with the EGI Information System with the GlueSchema to represent cloud resources. Even if the GlueSchema defines generic computing and storage entities, it was developed originally for grid resources and can represent only partially the information needed by the cloud users. The EGI Fedederated Cloud is working to profile and extend the schema to represent IaaS resources and in the future, PaaS and SaaS services.

Most of the data currently published is semi-static data, configured by the site administrator during installation and update of the infrastructure. FedCloud is working on a system to automatically retrieve these data from the cloud management frameworks and to complement the information published with dynamic information.

4 IBERGRID in Fedcloud

IBERGRID partners have played an active role in the development and deployment of FedCloud since its inception. We describe below the major contributions from IBERGRID that have enabled FedCloud to become a production infrastructure.

4.1 IBERGRID Resource Providers

IBERGRID Resource Providers have participated in EGI Federated Cloud since the early stages of the initiative. Currently there are four fully integrated and

certified as production-ready Resource Providers in IBERGRID: three sites (BIFI, CESGA and IFCA) offering an OCCI endpoint, and one site (BSC) providing a CDMI endpoint for cloud storage. Table 1 summarizes the committed resources for running VMs.

Site	Cloud Middleware	Resources
BIFI	OpenStack	720 cores with 740GB RAM: 2 testbeds x 360 cores with 370GB RAM (in Xeon servers 2 x hexacore, 24GB RAM)
CESGA	OpenNebula	296 cores with 592GB RAM (37 Xeon servers with 8 cores, 16GB RAM)
IFCA	OpenStack	2288 cores with 7592GB RAM (several servers configurations: 32 nodes x 8 cores x 16GB RAM; 36 nodes x 24 cores x 48GB RAM; 34 nodes x 32 cores x 128GB RAM; and 1 node x 80 cores x 1TB RAM)

Table 1. Certified sites in FedCloud

4.2 Integration of OpenStack in FedCloud

The OpenStack project is an open source project that provides software to run a private IaaS clouds. Currently 10 out of 18 of the certified Resource Providers in FedCloud are using OpenStack.

One of the core services of OpenStack is Keystone, which provides a common identity management for all the other services and acts as entry point to the cloud. IFCA has developed an extension to enable VOMS authentication in Keystone [22] allowing users to log in using a valid VOMS proxy certificate for a set of configured VOs. The extension takes care of automatically creating new users for trusted VOs on the fly. Once authenticated, Keystone returns a token to the user that grants the access to other services of the OpenStack installation.

IFCA has also contributed with several bug fixes and updates to the OCCI-OS extension for supporting OCCI in OpenStack. These updates allow to use the extension in the most recent releases of OpenStack and includes improvements for supporting contextualization in the API.

The original software package (osssm [23]) for getting accounting information from OpenStack and publishing it into the EGI Accounting repository is no longer maintained and uses the legacy metrics API of OpenStack which will be deprecated in upcoming versions. The team at IFCA has developed a new package (os-cloud-ur [24]) for generating the usage records by querying the OpenStack Ceilometer service (which is in charge of collecting and processing metrics in the latest versions of OpenStack) and publishing them to the repository.

4.3 Contextualization

IBERGRID leads the contextualization scenario of FedCloud which proposed new OCCI *mixins* for passing data to the VMs and inject ssh keys into those VMs.

IFCA provided the initial implementation of these *mixins* in OCCI-OS and contributed with patches for the usage of cloud-init with OpenNebula. As leaders of the scenario, we have also written extensive documentation on contextualization and maintain cloud-init packages in EGI's AppDB.

A dedicated EGI mini-project [25], with IFCA and CESGA as participants, dealt with the contextualization of VMs and the automatic deployment of applications for the final users. This work resulted in the implementation of a new service for defining recipes for applications and a user-friendly web interface for starting VM instances with those applications installed during the instantiation.

4.4 Accounting and Monitoring

The Accounting scenario of FedCloud counts with the participation of BSC and CESGA to define the format of the usage record that Resource Providers must send to the central accounting repository. CESGA, as developers of the Accounting Portal, have also implemented new views [26] to display the data received.

FedCloud includes the monitoring of the availability and reliability of the cloud resources with the SAM Framework. The framework includes probes for monitoring the status of the OCCI interface, the information system and the publication of accounting data. CESGA collaborates in the development of the probes and provides testing SAM instances for deploying the latests versions without disrupting the production infrastructure.

4.5 Brokering

The Brokering scenario contains relevant IBERGRID participation as well. This workgroup investigates the possible brokering solutions that can be used in the Federated Cloud Infrastructure. The group is leaded by CESGA, that are responsible for collecting information about existing solutions compatible with the OCCI and CDMI management interfaces.

Within this group, two brokering solutions developed by partners in the Iberian Peninsula are considered:

COMPSs. COMPSs [27] provides a programming model that allows the execution of applications on distributed infrastructures (grids, clusters, cloud). It performs the automatic selection of the VMs types depending on the tasks constraints.

VMDIRAC. VMDIRAC [28] is an extension of DIRAC to transparently integrate Federated Clouds. It dynamically starts VMs using the standard OCCI interface to execute the jobs submitted to DIRAC. It also includes a Web UI to browse and monitor started VMs.

5 An example of Use Case: Supporting Applications in the ESFRI LifeWatch

ESFRI, the European Strategy Forum on Research Infrastructures, is a strategic instrument to develop the scientific integration of Europe and to strengthen its

international outreach. The forum is dedicated to the identification of the new research infrastructures of European interest, with the goal of promoting the competitiveness of European research.

In this context Lifewatch, the European e-infrastructure for Biodiversity and Ecosystem Research is included in the ESFRI roadmap as an example of infrastructure which can take advantage of operational facilities already distributed in the different countries.

The core ICT infrastructure of Lifewatch will be develop on top of the IBERGRID infrastructure, profiting from the work done in the context of EGI. The joint endeavor among researchers in biodiversity of Spain and Portugal, striving to develop appropriate ICT tools for European ecosystems has been denominated IBERLIFE.

The ICT core services will include generic user oriented tools, such as portals and catalogs, but also community specific tools such as tools for provenance and annotation of data, workflow composition elements, and facilities to access computational resources. Data and tool resources need to include the possibility of accessing analytical and modeling tools.

The composition of such core services will take place in the so-called virtual research environments, which will act as virtual laboratories where researchers can access advanced ICT tools to perform basic tasks.

5.1 A first example: running a watershed model

A first example has been implemented in FedCloud. IFCA collaborates within the European LIFE+ project ROEM+ where a Spanish SME, Ecohydros, is addressing the problem of modeling Water Quality in a water reservoir (Cuerda del Pozo in Soria) that is supplying drinking water to an small city in Spain.

Understanding the Water Quality model requires a complete simulation of the processes that affect the water, and then validating this simulation with real data. Our group at IFCA has worked together with Ecohydros for the last five years, first to implement a near real time data acquisition system that provides this data directly to an offline data management system, and more recently, using a well known software suite, Delft3D [29], to model the physical, chemical and biological conditions of the water.

The key process to be modeled is eutrophication, a process caused by the excess of nutrients in the water reservoirs leading to an increase in vegetation and other organisms and microorganisms, in particular algae, deriving in algae bloom when combined with certain conditions of solar radiation and water temperature. The following depletion of oxygen in the water leads to the death of many microorganisms and in general to a large reduction of life in it, and their impact is important in water reservoirs used for urban supply. The final aim of the project is to develop an early warning system for this reservoir that allows policy makers and authorities to know when an algae bloom is going to happen in order to take actions.

Delft3D is an Open Source software suite that works over a mesh made from a map of the modeled water body, in our case Cuerda del Pozo reservoir in Soria

(Spain). The program runs with 2D and 3D meshes with a number of layers that can be edited by the user. Mesh resolution is an important factor for program performance. With a low or medium resolution mesh (cells larger than 250x250 meters with few vertical layers) execution can be successfully accomplished with standard PCs. However, when a detailed simulation is required—as it is the case to model the complex conditions leading to eutrophication—the resolution must be increased (e.g. 100x100 meter cells with more than 30 vertical layers) and more powerful computers are needed. Given our project requirements in CPU (>=2.5Ghz, few cores), memory (>12Gb) and disk (up to a few Terabytes), EGI FedCloud was considered the best option as resource provider.

Although Delft3D is parallelizable we use a special kind of model named Z model where every single layer has the same size and so it cannot be easily parallelized. However, the model needs to be calibrated by running the simulation with different parameters or sub-models (e.g. murakami or ocean sub-model for heat flux modeling). EGI, with a large and distributed offer of powerful resources, provides the ideal platform for improving our productivity. Moreover, water quality model can be parallelized so using multiple cores for this step of the analysis also improves performance. Delft3D software is a highly customized code with a lot of dependencies that needs to be configured and tuned before compiling. FedCloud allows us to launch virtual machines according to the specific features that the software needs (OS flavor, disk, memory and processor) and we can install directly everything needed without the site administrators intervention.

We have used FedCloud infrastructure to process the first main part of the model (hydrodynamic simulation) using a Delft3D module called Delft3D-FLOW. This module simulates the reservoir behavior with a set of input parameters like the rivers that flow into the reservoir. Delft3D-FLOW generates as output the modeled data, such as water level, salinity or temperature in each layer. This last parameter is one of the keys for the algae bloom, so special care must be taken to simulate it correctly. Figure 3 compares the real data taken from the reservoir with the output data obtained from the simulation. On the left water evolution level in meters during the simulation period (May 27th - November 3rd) is shown. On the right, the plot shows the temperature of water (°C) against depth (m) during a profile made on June 29th.

We selected CESNET and IFCA sites of FedCloud for our tests. We launched Ubuntu 12.10 virtual machines with 8 cores, 14Gb of memory and 190Gb of disk at IFCA and Ubuntu 12.04 with 4 cores, 15Gb of memory and 10GB of disk at CESNET. Those VMs allow root access, so installation and configuration of software are simpler than using a grid infrastructure (where the software platform is managed by the site administrators).

6 Future perspective

EGI established the Federated Cloud Task Force in July 2011 to explore the federation of private institutional Cloud deployments as part of the EGI infrastructure. In May 2014, the FedCloud testbed officially became part of EGI production infrastructure.

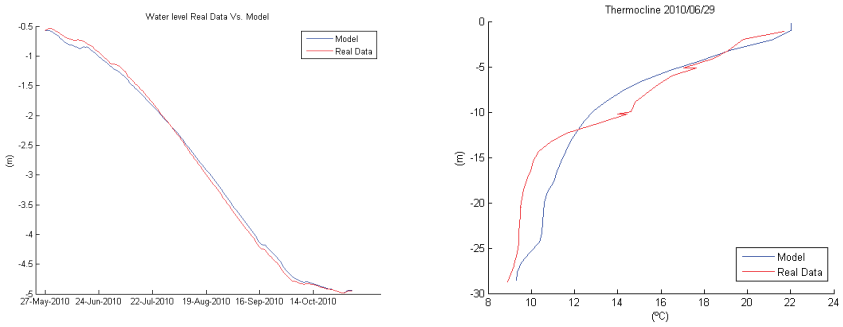


Fig. 3. Model vs. real data. Left: Water Level. Right: Thermocline

In the upcoming months, new users will start to test and port their applications to the new environment. The infrastructure will be open to new VOs and new User Access Policies will be defined. With heavier use, issues not detected during the pre-production phase may be discovered. New users will also bring new requirements to the infrastructure not met by the current level of support of FedCloud. The task force will work to provide a smooth experience to all platform users.

New resource providers will also join and become part of the infrastructure. FedCloud counts at launch with 5,000 cores and 225TB of storage. It is expected that by the last quarter of 2014, 18,000 cores and 6,000TB will be available. Within IBERGRID, four sites have already expressed their interest in joining to FedCloud: CETA-CIEMAT, IFAE/PIC, NCG-INGRID and UPV. They will need to integrate their cloud management platforms with the EGI services and pass through the certification process.

Current platform features focus on Infrastructure as a Service (IaaS), which is too complex for most scientific end users. FedCloud will collaborate with software initiatives to provide higher level services (PaaS or SaaS) that will aid on the adoption of the cloud technology by hiding the complexity of the infrastructure.

Acknowledgements

This work is partially funded by the EGI-InSPIRE (European Grid Initiative: Integrated Sustainable Pan-European Infrastructure for Researchers in Europe), a project co-funded by the European Commission (contract number INSFO-RI-261323).

References

1. R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, "Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures," *Computer*, vol. 45, no. 12, pp. 65–72, 2012.
2. "Openstack." http://www.openstack.org, 2014.

3. "Vmware." http://www.vmware.com, 2014.
4. "Open Cloud Computing Interface (OCCI)." http://occi-wg.org/, 2014.
5. ISO, "Inormation technology – Cloud Data Management Interface (CDMI)," Tech. Rep. ISO/IEC 17826:2012, ISO, NOV 2012.
6. S. Andreozzi, S. Burke, F. Ehm, L. Field, G. Galang, B. Konya, M. Litmaath, P. Millar, and J. Navarro, "Glue specification v. 2.0," Tech. Rep. GFD-R-P.147, Open Grid Forum, MAR 2009.
7. R. Alfieri and et al., "Voms, an authorization system for virtual organizations," in *Grid computing*, pp. 33–40, Springer, 2004.
8. M. Jiang, C. Novales, G. Mathieu, J. Casson, W. Rogers, and J. Gordon, "An apel tool based cpu usage accounting infrastructure for large scale computing grids," in *Data Driven e-Science* (S. C. Lin and E. Yen, eds.), pp. 175–186, Springer New York, 2011.
9. "Service Availabilty and Monitoring (SAM)." https://wiki.egi.eu/wiki/SAM, 2014.
10. "EGI Application Database (AppDB)." https://appdb.egi.eu, 2014.
11. "VMCatcher." https://github.com/hepix-virtualisation/vmcatcher, 2014.
12. "Open Grid Forum (OGF)." http://ogf.org/, 2014.
13. R. Nyren, A. Edmonds, A. Papaspyrou, and T. etsch, "OCCI core (v1.1)," Tech. Rep. GFD-P-R.183, Open Grid Forum, JUN 2011.
14. T. Metsch and A. Edmonds, "Open Cloud Computing Interface - RESTful HTTP Rendering (v1.1)," Tech. Rep. GFD-P-R.185, Open Grid Forum, JUN 2011.
15. T. Metsch and A. Edmonds, "Open Cloud Computing Interface - Infrastructure (v1.1)," Tech. Rep. GFD-P-R.184, Open Grid Forum, JUN 2011.
16. "rOOCI – a Ruby OCCI framework." http://gwdg.github.io/rOCCI/, 2014.
17. "Occi – OpenStack." https://wiki.openstack.org/wiki/Occi, 2014.
18. "Synnefo." http://www.synnefo.org, 2014.
19. "cloud-init documentation." http://cloudinit.readthedocs.org/en/latest/, 2014.
20. "Storage Networking Industry Association(SNIA)." http://snia.org/, 2014.
21. "STOrage proXY (STOXY)." https://stoxy.readthedocs.org/, 2014.
22. A. Lopez Garcia, E. Fernandez-del Castillo, and M. Puel, "Identity federation with VOMS in cloud infrastructures," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1, pp. 42–48, Dec 2013.
23. "APEL/SSM OpenStack connector." https://github.com/EGI-FCTF/osssm, 2014.
24. "OpenStack Cloud Usage Record." https://github.com/enolfc/os-cloud-ur, 2014.
25. "Automatic deployment and execution of applications using cloud services." https://wiki.egi.eu/wiki/VT_AppDeployment, 2014.
26. "Egi cloud accounting portal." http://accounting-devel.egi.eu/cloud.php, 2014.
27. F. Lordan and et al, "Servicess: An interoperable programming framework for the cloud," *Journal of Grid Computing*, vol. 12, no. 1, pp. 67–91, 2014.
28. V. Méndez Muñoz and et al, "Rafhyc: an architecture for constructing resilient services on federated hybrid clouds," *Journal of Grid Computing*, vol. 11, no. 4, pp. 753–770, 2013.
29. "Delft3D." http://oss.deltares.nl/web/delft3d, 2014.

Leveraging EGI Federated Cloud Infrastructure for Hadoop analytics

J. López Cacheiro[1], A. Simón[1], J. Villasuso[1], E. Freire[1], I. Díaz[1], C. Fernández[1] and B. Parak[2]

[1] Fundación Centro de Supercomputación de Galicia, Spain
grid-admin@cesga.es
[2] CESNET, Czech Republic
boris.parak@cesnet.cz

Abstract. FedCloud, the federated cloud infrastructure developed inside EGI, is in production since mid May 2014. In this paper we evaluate its suitability to perform Big Data analytics using Hadoop. Due to the size of the benchmarks performed, they represent also a real benchmark of the performance of FedCloud under heavy usage conditions, providing the first results about the scalability of the system.

1 Introduction

Nowadays, the possibility to run Big Data calculations over federated clouds is attracting the interest of the research community. Federation of resources poses serious challenges both from the cloud and Big Data perspectives. The aim of this paper is to evaluate the suitability of the EGI Federated Cloud infrastructure (FedCloud) to run Big Data analytics using Hadoop.

The EGI Federated Cloud Task Force [1] started its activity in September 2011 with the aim of creating a federated cloud testbed (FedCloud) to evaluate the utilization of virtualized resources inside the EGI production infrastructure. FedCloud is in production since mid May 2014 as presented at the EGI Community Forum 2014 [2], and during the last year it has been already available for experimentation by the different user communities. A more detailed description of the EGI FedCloud infrastructure is given in [3].

Big Data is an emerging field that can benefit from existing developments in cloud technologies. One of the most well-know solutions in this arena is Apache Hadoop [4], an open-source framework that implements the MapReduce programming model [5]. It provides both a distributed filesystem (HDFS), a framework for job scheduling, and a MapReduce implementation for parallel processing of large data sets.

In the MapReduce programming model users simply specify two functions: a *map* function and a *reduce* function. The *map* function processes a set of key/value pairs–usually the lines of a text or sequence file–and generates a new set of intermediate key/value pairs which are later on passed–partitioned and sorted–to the *reduce* function provided by the user merging all intermediate values associated with the same key.

The aim of our work is to do an initial assessment of the suitability of FedCloud to run Hadoop jobs through a series of real-world benchmarks. The paper is organized as follows: in Section 2 we describe the methodology used to deploy Hadoop inside the FedCloud federated cloud infrastructure; in Section 3 we include a comparison of startup times with Amazon EC2 and we present the results of the TeraGen and TeraSort benchmarks as well as the two selected use cases; finally in Section 4 we present the main conclusions of this work, summarizing the main problems encountered as well as outlining the future steps that could be taken to provide a Hadoop on-demand service in FedCloud.

2 Methodology

In this Section we describe the methodology that we have followed to deploy a Hadoop cluster instance inside the FedCloud federated cloud infrastructure.

2.1 Cluster startup

The Hadoop clusters consists of one *master* node and a variable number of *slave* nodes depending on the size of the cluster. For example in the case of a 101 node cluster, one node will be configured as master and the remaining 100 as slaves. The master node will run the *namenode*, *secondarynamenode* and *jobtracker* services and each of the slaves will run just the *tasktracker* and *datanode* services.

A customized virtual machine (VM) image was created including different versions of Hadoop (up to 1.2.1) and Oracle Java JDK (both 1.6 and 1.7). This customized image was registered in the EGI Marketplace [6] which provides a common place where where we can can store images. The Marketplace does not store the VM images into a common repository, it just shows which resource providers (RPs) have the VM image available at their endpoint. This means that each RP has to manually download the image to his local site in order to make it available at his endpoint.

To instantiate the virtual machines that will form the Hadoop cluster we used rOCCI [7], the implementation of the OCCI 1.1 specification developed by one of the authors, using VOMS proxy credentials. The new rOCCI implementation provides several advantages, like the fact that we are able to avoid the previous issue that all user certificates request are mapped to a single ONE_USER account. Additionally, the usage of VOMS over plain X509 auth greatly simplifies the access to the different sites that form FedCloud without the need to request each site to give access to our X509 certificate's DN.

FedCloud does not count with a workload management system, so each VM creation request must be sent to the appropriate endpoint, and we should take care of partitioning the cluster between the different FedCloud resource providers.

In order to increase the concurrent number of VMs, we used small size VM instances with 1GB of RAM and 1 CPU for the small and medium-size use cases and 2GB of RAM and 1 core for the TeraGen and TeraSort bechmarks.

Table 1. Tuned Hadoop configuration for small VM instances. It includes both the HDFS and MapReduce parameters used.

Parameter	Type	Value
fs.inmemory.size.mb	core-site	200MB
io.file.buffer.size	core-site	128KB
mapreduce.task.io.sort.factor	core-site	100
mapreduce.task.io.sort.mb	core-site	100
mapred.tasktracker.map.tasks.maximum	mapred-site	1
mapred.tasktracker.reduce.tasks.maximum	mapred-site	1
mapred.reduce.tasks	mapred-site	0.95×*num. slaves*
dfs.datanode.du.reserved	hdfs-site	1GB
dfs.block.size	hdfs-site	1MB / 64MB
dfs.replication	hdfs-site	3
HADOOP_HEAPSIZE	hadoop-env	512MB

The tuned configuration parameters for running the selected use cases are given in Table 1. The column type identifies the configuration file where the parameter is set, the complete configuration files can be found at [8].

These parameters have been selected to tune the Hadoop cluster to the resources available (1GB of RAM and 1 CPU per node). The *dfs.replication* parameter indicates how many copies we want to store of each block, in this case we use three replicas that will allow different nodes to execute the same MapReduce tasks–speculative execution–mitigating the performance degradation that could be experienced in a heterogeneous cluster with VMs running under hypervisors with different load conditions. The *mapred.tasktracker.map.tasks.maximum* and *reduce.tasks.maximum* are set to 1 because we only have 1 CPU available to run both services and the HADOOP_HEAPSIZE is set to 512MB to allow both running at the same time without exhausting the node's memory. The number of reduce tasks, *mapred.reduce.tasks*, is set to 95% of the number of slaves in the Hadoop cluster.

Our Hadoop installation will cope with two very different use cases that influence our decission about the *dfs.block.size* values to use. In the case of the Encyclopædia Britannica's use case the block size is set to 1MB and in the Wikipedia's use case, that uses a much larger data set, to 64MB. Under these conditions both use cases generate less than 700 total MapReduce tasks which can be handled well by the Hadoop master node which has only 1GB of RAM and 1 CPU. For the TeraGen and TeraSort becnhmarks we use an even larger block size of 128MB.

2.2 Benchmarks

In order to evaluate the suitability of FedCloud to perform Big Data analytics using Hadoop we selected the TeraGen and TeraSort benchmarks as well as two use cases representing usual small and medium-size jobs. These use cases use real data sets and a fairly common MapReduce operation.

In the Big Data arena the TeraGen and TeraSort benchmarks could be considered the equivalent of Linpack in HPC. There is even a list equivalent to the well-know Top500 list of the most powerful supercomputers in the world. This list was created by Jim Gray and it is known as the Sort Benchmarks list [9]. This list is not restricted to Hadoop clusters, and any Big Data system that is able to sort files can compete to appear there. Currently the largest sort rate is achieved by a Hadoop cluster of 2100 nodes that reaches a sort rate of 1.42TB/min [10].

TeraGen is the part of the bechmark that generates the input data set that will be used by TeraSort. It can be configured to run in parallel using the whole cluster so the throughtput it obtains is much higher than a standard HDFS put operation.

TeraSort is a standard MapReduce sort job that uses a custom partitioner with a sorted list of $N-1$ sampled keys to guarantee that the output keys of reduce n are lower than the output keys of reduce $n+1$. This allows the concatenation of all the output files generated by each Reducer to produce one ordered output file.

The selected use cases using two well-known data sets which are quite representative of a broad range of jobs that could be performed in a Hadoop cluster–most MapReduce jobs are based on the same type of operations–. The use cases were run both in a federated cluster deployment and in a one-site-only cluster deployment, allowing us to compare the results and analyze the degradation when using remotely distributed resources.

The first use case, representing small-size Hadoop jobs, is based in the Encyclopædia Britannica[11] data set–the 1911 version that is already in the public domain–. This represents a rather small data set of just 176MB, so that this use case can be considered as a small scale benchmark that will serve to evaluate the expected degradation in performance due to fact that we will be using federated resources located at different sites. The data set is downloaded directly from the master node and later on it is distributed from there to the slaves, what is called a *put* operation in Hadoop argon. This use case has a great overload due to the creation of rather small MapReduce tasks that will stress the communication system because it uses a rather small block size of just 1MB, so each map operation has little to process.

The second use case, representing medium-size Hadoop jobs, is based in the Wikipedia[12] data set. This represents a much larger data set comprising 41GB, so it can be considered a more realistic benchmark of a medium-size Hadoop job. Due to the size of the data set the files could not be downloaded directly in the master node so the *put* operation is done directly from our local Hadoop client. This use case will incur in a lower task creation penalization because it uses a larger block size of 64MB, so each map task will take much longer than the time needed for its creation.

Both use cases were run for different cluster sizes ranging from 10 to 101. In all the executions we measured two parameters:

1. *The put time:* the time to load the files into the Hadoop HDFS filesystem
2. *The wordcount time:* the time to perform a simple wordcount MapReduce job that computes the number of occurrences of each word over the complete data set.

All the measurements where repeated at least 5 times in order to evaluate the variability of the results, the only exception were the *put* times for the Wikipedia that due to its duration could only be run once.

3 Results

Following the methodology described in previous Section we deployed Hadoop in FedCloud and executed the benchmarks and use cases mentioned in Subsection 2.2.

In Subsection 3.1 we present the results related to the cluster startup process. In this case we were dealing with the instantiation of a number of VMs ranging from 10 to 101, so it can serve also as a stress test of both rOCCI and the underlying cloud management backends.

In Section 3.2 we present the benchmark results both for the two selected use cases–Encyclopædia Britannica and Wikipedia–and for the TeraGen and TeraSort benchmarks.

3.1 Cluster Startup: Comparison with Amazon EC2

The startup times for each cluster deployment varied, being the most representative ones those obtained for the larger deployment: the 101-node cluster. In this case, the time needed by the rOCCI client to return all the resource endpoints corresponding to each VM ranged from 71 to 86 minutes. This is just the time to get the identifier corresponding to each VM instance, actually to have all the VMs running took longer: around 80 minutes more. So the total cluster startup time ranged from 2.5 to 3 hours.

Some of the rOCCI requests failed with an *execution expired* error and some of the VMs did not start. In the worst case 21 out of 101 failed. This type of errors where observed when deploying more than 20 VMs at CESGA. This analysis could not be done in CESNET because there were only 10 VM slots available at the time of the benchmarks, due to a restriction in the amount of public IPs that they had available.

Trying to understand the source of this issue the performance of the OpenNebula frontend was tracked during the cluster startup. As it can be seen in Figure 1 the frontend experiences a high load several minutes after the start of the cluster deployment, this causes it to respond slower, producing the rOCCI execution expiration errors mentioned above. Looking at the processes in the OpenNebula frontend, we could see that this increase in the load is mainly due to the *scp* processes launched to copy the image template to the OpenNebula nodes, more than 20 simultaneous scp processes seem to affect considerably the system performance, reducing its response times considerably. The copy process is also limited by the fact that the frontend has only 1 Gbps connectivity, acting as a bottleneck for the image delivery process.

In order to optimize the startup times we reduced the size of the image to 4GB. But this configuration would not leave enough space for applications, to solve

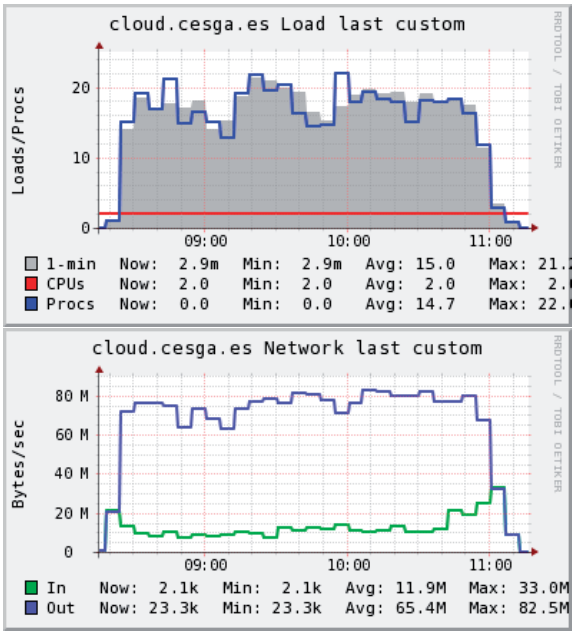


Fig. 1. OpenNebula frontend load and network usage graphs taken from Ganglia during the deployment of the 101 nodes cluster.

this issue an additional disk of 70GB was created on-the-fly using a *fs* disk type in OpenNebula and an *ephemeral* disk in OpenStack. Furthermore, we modified the *datastore* used in OpenNebula, so that the transfer manager (TM) used is *shared* instead of *ssh*. Since the instances are declared as non-persistent, this allows to perform a powerful optimization in the instance deployment process done by OpenNebula because the image copy process will be no longer done in a centralized way from the front-end node but in parallel from each node using a simple copy operation from the shared storage. The optimized startup times can be found in Table 2.

Table 2. Optimized startup times at CESGA after tuning OpenNebula configuration.

Cluster size	Startup time (s)
10	249
21	269
51	822
101	1096

With the aim of comparing the startup times obtained with those of other commercial clouds we performed performed similar measurements of cluster startup times in Amazon EC2, this way we we can have a better understanding of how FedCloud performs compared to commercial clouds.

Amazon Elastic Compute Cloud (Amazon EC2) is a very popular commercial cloud platform that provides resizable compute capacity in a pay-per-use model. Even if the infrastructure behind Amazon EC2 is much larger than the one behind FedCloud, notice should be given to the fact that to launch more than 20 instances in Amazon EC2 you have to fill a form indicating the number of instances you need and their expected usage. For some users, especially commercial users, this requirement may be unnacceptable. This request is not validated automatically so you have to wait several hours until it is validated.

There are different tools available to create clusters in Amazon EC2, being Apache Whirr [13] one of the most popular. The Apache Whirr project started as a set of bash scripts distributed with Apache Hadoop for running Hadoop clusters on Amazon EC2, but it evolved to a Java version based on Apache jclouds [14], an open source multi-cloud toolkit that allows Whirr to support many different cloud providers. Unfortunately Whirr does not scale well due to the large number of REST requests it sends to the Amazon API that causes Amazon to temporarily block the client with a *Request limit exceeded* error. The largest cluster we were able to launch in the eu-west-1 region was 21 nodes (see Table 3).

To avoid this issue a custom tool [15] to create and configure the Hadoop cluster was developed. This tool used Amazon EC2 *instance-count* functionality to request all the instances in the same REST request avoiding the *Request limit exceeded* problem. The results are shown in Table 4. It can be seen that all the startup times are below 5 minutes. It is especially relevant the fact that both in the 51 and 101 clusters there where nodes that could not be reached for different reasons. In some cases the instance launch directly failed and the error was reported by Amazon. However, in most cases the failing instances were reported as ready, but they were not reachable from the other nodes in the cluster. The reason seems to be a security group internal network issue at Amazon.

Table 3. Amazon EC2 startup times using Whirr: the AMI and instance type used are also specified.

Cluster size	AMI	Instance type	Startup time (s)
10	eu-west-1/ami-c37474b7 EBS	t1.micro	555
10	eu-west-1/ami-c37474b7 EBS	m1.small	891
21	eu-west-1/ami-c37474b7 EBS	m1.small	1670
31	eu-west-1/ami-c37474b7 EBS	m1.small	Fails

Table 4. Amazon EC2 startup times using our custom tool: the AMI used was *eu-west-1/ami-54d43023* EBS plus instance store. The instance type used was *m1.small* in all cases.

Cluster size	Startup time (s)	Comments
10	191	
21	178	
51	583*	2 nodes not working
101	276*	16 nodes not working
101	297*	1 node not working

3.2 Benchmarks

In Table 5 we show the results obtained for the TeraGen and TeraSort benchmarks. In this case we can generate much larger data sets than the Encyclopædia Britannica and Wikipedia data sets, so we start at 50GB and go up to 2TB. Due to the larger size of the data sets the $dfs.blocksize$ was increased to $128MB$. We can see the scaling is good but after reaching 600GB the TeraSort benchmark fails because there is no enough space–total storage available is 2.65GB–to store the intermediate data. In order to evaluate larger data sets we reduced the number of HDFS replicas to one–reducing them to two would not provide much additonal margin–and rerun the whole set of benchmarks. In this way, we could reach 2TB in the Teragen benchmark. Unfortunately having just one replica eliminates the fault tolerance of the system, making the system much less reliable, and causing the larger TeraSort benchmarks to consistently fail.

The best results were obtained using 100 mappers in TeraGen and 100 reducers in TeraSort (the number of mappers is determined by the number of blocks).

We saw the importance of using a number of parallel tasks higher than the number of nodes so we can take advantadge of Hadoop's speculative execution. This is something critical in a cloud environment where the performance of the instances is very heterogeneous, depending greatly on the site where it runs and the activity of the other instances that share the same host.

Rack awareness was used take into account the node's physical location. Each instance's network gateway is used to define the datacenter part of the *datanode rack id*: */¡GW¿/default-rack*. This allows to define the topology automatically in a federated environment–taking into account local rack location is not feasible in this environment.

The MapOutput in-memory buffer was also especifically tuned for TeraSort using $io.sort.mb = 150$, $io.sort.record.percent = 0.138$ and $io.sort.spill.percent = 1.0$. This configuration avoids the expensive cost of *spills*–spilling to disk more than once would triple the disk I/O required.

Additionally to the previous large-scale benchmarks we also evaluated some real-world examples of medium and small Hadoop jobs. These use cases are, from

Table 5. Results obtained for the TeraGen and TeraSort benchmarks: times where measured in a federated cluster that included 20 slave nodes running at CESGA and 20 at CESNET with the master node at CESGA.

Dataset size	replicas	teragen (s)	terasort (s)
50GB	3	1342	750
100GB	3	2510	1504
200GB	3	4575	3278
300GB	3	5934	5766
400GB	3	7771	7244
600GB	3	11595	Failed
800GB	3	15351	Failed
50GB	1	86	820
100GB	1	140	1506
200GB	1	272	3472
300GB	1	379	5529
400GB	1	538	7679
500GB	1	624	10094
600GB	1	715	Failed
800GB	1	1014	Failed

our experience, more representative of the typical needs of most users, so they will help to evaluate the suitability of FedCloud for these tasks.

In Table 6 we show the results obtained for the Encyclopædia Britannica use case. All measurements were repeated 5 times and the average and standard deviation are displayed, allowing to have an estimation of the variability involved in the measurement. As it can be seen even if the block size used in this case is very small (1MB), the wordcount map reduce job scales really well and the overhead introduced by the federated cluster is small, being in general in order of 10-15%, and 28% in the worst case that corresponds to the 101 cluster. This is mainly due to the large overhead that MapReduce task creation and distribution imposes in this use case, especially in the larger cluster sizes that involve more reduce tasks (we are using a number of reduce tasks equal to the 95% of the total number of slaves).

As expected the *put* times are much lower in one-site-only scenario because in this case the connection between all the VMs is much faster because all of them are generally located in the same data centre.

In Table 7 we show the results obtained for the Wikipedia use case. It should be noted that due to the high duration of the transfers, we did not repeat the *put* measurements 5 times, and only the wordcount measurements were repeated. Also due to the longer duration of the benchmarks we only used the two larger cluster sizes: 51 and 101.

In this case the wordcount mapreduce results are quite surprising because the federated cluster instance is able to outperform the one-site-only instance in both

Table 6. Benchmark times obtained for the Encyclopædia Britannica use case. Both the average time and standard deviation are reported. Federated times where measured in a federated cluster that included resources both from CESGA and CESNET; all one-site-only times where obtained in a one site deployment at CESGA except the 10 node cluster indicated in the second row that run at CESNET.

	Federated		One-site-only	
Cluster size	put (s)	wordcount (s)	put (s)	wordcount (s)
10 (CESGA)			47 ± 2	169 ± 1
10 (CESNET)			9.5 ± 0.2	160 ± 1
21	235 ± 12	108 ± 2	37 ± 1	97 ± 1
31	189 ± 4	90 ± 2	36 ± 2	78 ± 2
41	190 ± 11	81 ± 4	33 ± 2	71 ± 3
51	133 ± 7	73 ± 3	33 ± 1	66 ± 2
101	94 ± 7	67 ± 22	33 ± 4	52 ± 5

scenarios. This is attributed mainly to the faster execution of the benchmark in CESNET nodes compared to CESGA nodes (CESNET VMs use Xen paravirtualization and a faster CPU whereas CESGA VMs use KVM and QEMU). In this case the relative overhead due to MapReduce task creation is much lower than in the previous use case because we are using a much larger block size of 64MB.

The *put* times are much larger in the federated cluster due to the fact that the amount of data to transfer is much larger: HDFS stores three copies of the Wikipedia, 42GB each, so that the available bandwidth between the VMs and the UI located at CESGA plays a key role, being much lower in the case of CESNET VMs.

Table 7. Benchmark times obtained for the Wikipedia use case. Both the average time and standard deviation are reported except for the put times that where only measured once for each deployment. Federated times where measured in a federated cluster that included resources both from CESGA and CESNET; all one-site-only times where obtained in a one site deployment at CESGA.

	Federated		One-site-only	
Cluster size	put (s)	wordcount (s)	put (s)	wordcount (s)
51	19190	1001 ± 39	6705	1347 ± 117
101	13208	705 ± 14	5665	725 ± 18

4 Conclusions

The results presented in this paper serve as an initial evaluation of the performance of Apache Hadoop running on the EGI federated cloud infrastructure. They show that FedCloud is especially suitable for small and medium-size Hadoop jobs where the data set is already pre-deployed in the Hadoop HDFS filesystem. Considering the two use cases selected, we see that the federated cluster is even able to outperform the local one if the remote nodes are faster than the ones available locally.

This results show that small VM instances are good enough to run Hadoop workloads assuming an appropriate tuning of the configuration is done, of course using larger instances would simplify the configuration and it would allow to reach larger problems. In our benchmarks the *tasktrackers* run out of memory when running the Wikipedia's use case if we used a small number of reduce tasks, additionally when using a small *dfs.block.size* it was the *jobtracker* the service that run out of memory because it was not able to cope with the large amount of tasks.

The initial results showed very disappointing results about the startup times and reliability of the system. Almost 3 hours were needed to start the larger cluster with 101 nodes, a high penalty for small and medium size jobs that usually finish in less than an hour. Additionally, failures started to appear when trying to start more than 20 VMs. After analyzing in detail the startup process we saw these failures could be mainly attributed to limitations in the resource provider cloud management backend. We were able to appropriately tune both the cloud provider backend and our marketplace image, reducing the startup time to less than 20 minutes for the 101-node cluster. This optimization also lead to an improvement of the reliability of the startup process, so no more instances failed during the cluster startup process.

The startup times were compared with those of Amazon EC2. In order to run the measurements, a custom tool had to be developed for deployment of cluster larger than 20 nodes because of the request rate limit imposed by Amazon that causes Whirr to fail in such scenarios. Our measurements showed startup times of around 5 minutes for the larger 101-node cluster, but there were reliability issues in Amazon's infrastructure that caused several nodes not to work properly.

There is also a large overhead introduced by the upload–*put* times–of the data sets to the HDFS filesystem. In the medium-size use case it takes 10 times more to upload the data set than to run the actual job. This aspect should be taken into account when considering Hadoop on demand services.

Considering the scalability of the system we were able to run the TeraGen benchmark up to 2TB and the TeraSort benchmark up to 500GB. Even if larger targets could be reached increasing the size of the system, we found networking issues when trying to add additional instances running at additional sites. These issues seemed site-specific.

There are some aspects that could improve the usability of FedCloud like adding a central workload management system, an automated way to distribute and synchronize the image between all sites, or a mechanism to query the resources available at a given site.

Acknowledgements

We would like to thank BIFI, IFCA, CESNET and CESGA for providing the resources for the benchmarks as well as for their support during all this work.

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

References

[1] FedCloud: EGI Federated Cloud Task Force. https://wiki.egi.eu/wiki/Fedcloud-tf:FederatedCloudsTaskForce (June 2014)
[2] FedCloud: EGI Federated Cloud. https://wiki.egi.eu/wiki/Fedcloud-tf:Main (June 2014)
[3] Simón, A., Freire, E., Rosende, R., Díaz, I., Feijóo, A., Rey, P., López-Cacheiro, J., Fernández, C.: Egi fedcloud task force. In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. (2012) –
[4] Hadoop: Apache hadoop. http://hadoop.apache.org/ (June 2014)
[5] Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: Sixth Symposium on Operating System Design and Implementation. (2004) –
[6] EGI: EGI Marketplace. http://marketplace.egi.eu (June 2014)
[7] EGI: rOCCI API. https://github.com/gwdg/rOCCI (June 2014)
[8] EGI: Hadoop on fedcloud deployment tools. https://github.com/grid-admin/hadoop (June 2014)
[9] Gray, J.: Sort benchmark. http://sortbenchmark.org (June 2014)
[10] Graves, T.: Graysort and minutesort at yahoo on hadoop 0.23. http://sortbenchmark.org/Yahoo2013Sort.pdf (June 2014)
[11] Britannica: Encyclopædia Britannica. http://www.britannica.com (June 2014)
[12] Wikipedia: Wikipedia. http://www.wikipedia.org (June 2014)
[13] Apache: Apache whirr. https://whirr.apache.org (June 2014)
[14] Apache: Apache jclouds. http://jclouds.apache.org (June 2014)
[15] López Cacheiro, J.: Hadoop on-demand. https://github.com/alcachi/hadoop-on-demand (June 2014)

Exploring Containers for Scientific Computing

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. Researchers have access to a diversified ecosystem of computing resources ranging from notebooks to computing clusters and distributed infrastructures. Running applications across these different environments requires considerable effort. The Linux Containers technology allows multiple operating system environments to be executed on top of a shared Linux kernel space. It takes advantage of standard Linux kernel features to provide lightweight isolated Linux environments similar to virtual machines, but without the overheads of conventional virtualization. In this paper we highlight the benefits of this approach, and explore paths towards the use of containers to simplify the packaging and execution of scientific applications.

1 Introduction

Most scientific applications are developed for Unix-like environments. Still porting applications to each specific computing environment requires considerable effort and expertise. Furthermore users are today faced with a wide range of computing resources ranging from notebooks to computing clusters and distributed infrastructures. These resources have their own peculiarities and interfaces that often constitute obstacles to a transparent usage.

In order to exploit the computing resources at their disposal, the researchers need to port applications to multiple computing environments, and adapt them each time the environment changes. This is one of the reasons behind the interest on more flexible paradigms such as cloud computing, where users can have their own customized environments backed by virtual machines. This flexibility comes at a price, the scientists are then confronted with the need of managing virtual networks, virtual storage, virtual machines and their operating systems. In addition, at each site the management of these virtual resources is often performed differently.

Researchers thus face a complex ecosystem of heterogeneous computing resources. In this context lightweight forms of operating system virtualization may provide consistent portable run-time environments for the scientific applications, potentially facilitating their execution across computing resources.

2 Background

2.1 Lightweight Virtualization

Operating systems enable the coordinated exploitation of computing resources by multiple processes. Traditionally all processes share one common operating system configuration and compete individually for the system resources. Conventional virtual machines are frequently used to provide better separation between sets of activities running in the same physical machines, and to provide differentiated system configurations tailored to each activity. However, this separation can also be achieved in some operating systems through *operating system-level virtualization* [1] with the following benefits:

- No emulation or hypervisor overheads. All virtual execution environments share a single host kernel through process confinement.
- Multiple file system environments can be supported within a single host by confining the processes to specific file system trees.
- Processes within virtual execution environments are transparently scheduled and managed by the host kernel, allowing more efficient use of resources such as memory.
- No extra resource consumption related to the simultaneous execution of multiple operating system kernels and operating system services.
- Faster provisioning and environment start-up.

The main disadvantages are: inability of running software that requires an operating system kernel different from the one provided by the host, and exposure of all virtual environments to security vulnerabilities in the host kernel. Both are consequences of sharing the kernel.

2.2 State of the art

Examples of operating system-level virtualization are: Solaris Zones [2] available on Solaris and OpenSolaris, FreeBSD Jails [3] available on FreeBSD systems, and OpenVZ [4], Linux-VServer [5], Linux Containers (LXC) [6], and libvirt-LXC [7] on Linux. Since most scientific computing software and infrastructures are based on Linux, we therefore focus on Linux implementations.

Both OpenVZ and Linux-VServer are mature implementations but require modified Linux kernels which constitutes a major obstacle to their wider adoption. Conversely LXC and libvirt_LXC rely on vanilla Linux kernel features and therefore do not require modified kernels. LXC is available out of the box in many Linux distributions, and is more mature than libvirt_LXC. Both allow the manipulation of kernel features to create comprehensive contained environments, enabling the execution of multiple Linux distributions over one single kernel. The most relevant Linux kernel features for operating system-level virtualization are:

- Kernel namespaces [8]: are used to isolate a particular global system resource making it appear to the processes within the namespace that they are using the full resource.

  - Mount namespaces: isolate the mount points accessible to a group of processes.
  - UTS namespaces: provide hostname and domain name isolation.
  - IPC namespaces: isolate interprocess communication resources such as queues, semaphores and shared memory.
  - PID namespaces: isolate process identifiers, so that processes within a group cannot see processes in other groups, furthermore the process identifiers are remapped allowing processes within different groups to have the same process number.
  - Network namespaces: isolate network resources such as network devices, IP addresses, and routing tables.
  - User namespaces: isolate user identifiers and group identifiers. These identifiers are remapped and can be different inside and outside of the namespace.
- AppArmor [9] and SELinux [10]: are kernel security modules that implement mechanisms to support mandatory access control security policies. They can be used to protect the host system against accidental abuses from privileged users from inside the contained environment such as changes to cgroups, or writing into devices.
- Seccomp [11]: provides system call filtering.
- chroot [12]: provides isolated directory trees.
- cgroups [13]: provide hierarchical task grouping and per-cgroup resource accounting and limits. They are used to limit block and character device access and to suspend sets of processes. They can be further used to limit memory and block i/o, guarantee minimum CPU shares, and to bind processes to specific CPUs.
- POSIX capabilities [15]: split the privileges traditionally associated with the root account into distinct units, which can be independently enabled and disabled.

The availability of these features is fostering the interest around operating system-level virtualization in Linux. Nevertheless their correct and safe use even through tools such as LXC requires considerable knowledge and configuration.

2.3 Application Containers

Docker [16] is an open-source engine that automates the creation and deployment of applications in lightweight portable contained environments (containers). Initially Docker relied on LXC to manipulated the necessary kernel features, but currently it can support multiple execution drivers. It has its own native driver named *libcontainer*, and its modular nature may in the future allow it to run with other implementations (e.g. OpenVZ, FreeBSD jails, Solaris zones). Docker inherits many of the LXC features. However with Docker the creation of containers is greatly automated and simplified with a higher level of abstraction.

The Docker containers encapsulate the applications payload allowing them to run in almost any Linux host with a recent distribution. Docker is inspired in

the intermodal shipping container metaphor, where a multiplicity of goods can be packaged and transported via a multiplicity of means on standard containers. The availability of a common agreed format to ship goods has simplified their transport, storage and management. Similarly Docker offers a uniform way to package, distribute, deploy and execute applications.

When creating a container for an application only the software components strictly required for execution are needed. Since the operating system kernel is shared, there is no need of having one in the containers. Similarly usual processes that are started at boot time are also not needed. Many functions such as time synchronization, device management, power management, network configuration and others are simply performed by the host. Still an environment that resembles a common Linux system can also be made available inside the containers, including:

- A network interface with an IP address.
- A dedicated file system for each container.
- A set of system devices.
- Interactive shell support via pseudo-ttys.

LXC and Docker have a modular storage architecture that supports Union Filesystems namely AuFS [14], Snap-shotting Filesystems and copy-on-write block devices. Docker goes further by implementing a container image format with a layered structure. This approach allows each container file system to be mounted as a stack of superimposed read-only layers, on top of which a read-write layer is added. Any changes are thus performed and reflected in the read-write layer. When execution finishes this layer can be either discarded or added to the container image as another layer. Images are tar files that contain layers, each layer is also a tar file.

Docker can act as an image builder automating the steps required to create a new image that otherwise would have to be performed manually. This process takes a *Dockerfile* that specifies as input an existing image to which a sequence of actions will be applied. These are the actions that would need to be performed manually to compile or install the required software. Moreover new files can also be added. The image resulting from this process is a new storage layer that contains the changes performed during the build process. Existing images can thus be used as building blocks to create more complete images.

To facilitate image distribution and deployment Docker can push and pull images from a remote registry. Since images are made of layers, the registry implements a transparent incremental download and upload of the images transferring only the required layers.

2.4 Orchestration

Containers offer interesting features to support the execution of applications across computing resources. However most implementations such as LXC, libvirt_LXC and Docker operate at the host level as virtual machine managers. They do not offer any orchestration or clustering features. Hence, Docker is being incorporated in many software products such as OpenStack [17] or Apache mesos [19] allowing

the execution of containers in clouds and dedicated clusters. Other related projects are: CoreOS [18] a Linux distribution designed to support the execution of services in clusters and cloud resources. CoreOS uses Docker to encapsulate and execute services. Flynn [20] and DEIS [21] use Docker to provide platform as a service features and scale to cloud providers. Both Flynn and DEIS use a git push deployment model and are targeted at facilitating the flow from development to deployment. Several other software projects around Docker are emerging. These systems are mostly intended at creating local clusters for the execution of permanent services, and to scale out to public clouds.

3 Computing with Containers

For scientific computing, the described tools do not currently offer a comprehensive solution capable of simplifying the execution of applications across the e-infrastructures ecosystem. In addition researchers want to focus on science instead of computing. They require easier tools to simplify their work, enabling them to address their scientific goals. Although cloud computing is gaining expression, most intensive simulations and data processing continues to be performed on batch farms. Both approaches are complementary and they will likely be used for a long time. Therefore scientists need a simple way to execute applications across these and other computing resources without changing them. This implies running transparently on grid computing, cloud computing and other resources. Based on these assumptions the authors developed a proof of concept aimed at testing the usability of state-of-the-art containers in the IBERGRID infrastructure. The table 1 shows the software versions used in this study.

Distribution	Virtual machine manager	Kernel
Centos 6.5	Docker 0.11.1 + execution driver lxc-0.9.0	2.6.32
	Docker 1.0.0 + execution driver native-0.2	
Centos 6.5	LXC 0.9.0	
	LXC 1.0.0	
Fedora 20	Docker 1.0.0 + execution driver native-0.2	3.14.6
Fedora 20	LXC 0.9.0	
Ubuntu 14.04	Docker 0.9.1 + execution driver native-0.1	3.13.0-29
Ubuntu 14.04	LXC 1.0.3	

Table 1. Software versions used in this study

3.1 Use case

The proof of concept is based on the following use case. The researcher uses its own workstation to develop applications which are compute and/or data intensive. He wants to execute those applications on local computers accessible via SSH, and also in IBERGRID computing resources in Portugal. The IBERGRID resources are

computing farms accessible via CREAM Computing Elements (CREAM-CE) [22]. The operating systems in the researcher workstation and in the target execution hosts are different. He wants to execute his applications in the computing resources at his disposal with a minimum effort.

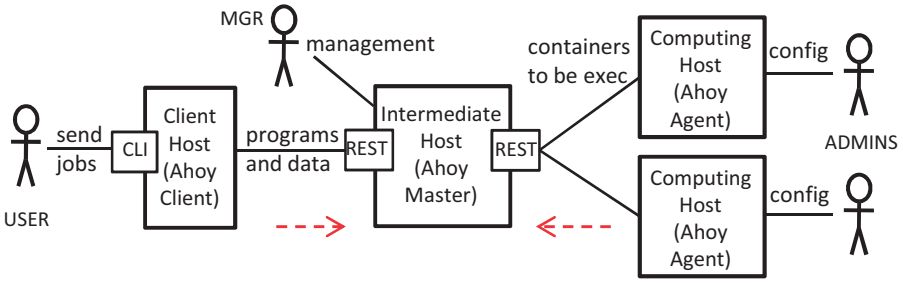


Fig. 1. Context diagram

The figure 1 shows the context diagram of the proof of concept. It shows three main components: the *Ahoy client* installed in the desktop computer, the *Ahoy master* installed in an intermediate host, and the actual computing hosts that execute the user applications which are launched through the *Ahoy agent*.

The *master* receives from the *clients* requests to perform actions on the user behalf, including building images and submitting them for execution in compute hosts. Those requests are validated and translated into container management actions. The *agents* contact the *master* periodically to obtain jobs for execution. The jobs are in fact containers to be launched. The *master* acts as an intermediate host between the user and the resources, hiding the details of accessing computing hosts and managing containers. It requires inbound and outbound Internet connectivity. A basic flow is as follows:

- Preparation of an application image:
  - The user invokes the *Ahoy client* to prepare an application image build request . The request is sent to the *Ahoy master* for processing. Therefore the user does not need to install in his computer any software besides the client.
  - As defined in the request, the *Ahoy master* builds a container using a pre-existing operating system image to which the user provided files and software packages are added.
- Execution of the application:
  - The user sends a request for application execution (job) to the *Ahoy master*. The request contains: metadata describing the job namely: the name of the application image built in the previous step, and the input data files. The files can be sent with the request itself or can be downloaded later at execution time.
  - The request is validated, prepared and queued by the *Ahoy master*.

- If needed the *Ahoy master* starts instances of the *Ahoy agent* in the computing hosts via the SSH or CREAM interfaces.
- The *Ahoy agent* contacts the *Ahoy master* to get jobs for execution.
- The *Ahoy agent* pulls the container image and starts it with the appropriate environment and restrictions.
- The user can check the job status by querying the *Ahoy master* which periodically gets information from the agents.
- Upon completion the output can be uploaded to the *Ahoy master* or other external destination accessible via supported protocols.
- Finally a cleanup of the computing host is performed by the agent.

Besides the user itself, in this scenario there are the following roles. The manager (labeled MGR) is the person responsible for the coordinated use of the distributed resources. In his role he may need to have a tighter control of how resources are shared and the need to implement policies and job resource requirements, therefore he manages the *Ahoy master*. The site administrators (labeled *ADMINS*) are responsible for managing the actual computing hosts, namely they install the software, perform its configuration, control the usage and ensure that usage policies are respected. They may work in different organizations and they have a direct concern on how and by whom their resources are used. The deployment and use of any form of virtualization requires their acceptance and cooperation.

3.2 Architecture

The figure 2 shows an architecture designed to explore the described use case using Linux containers. A demonstrator was developed in Python [25], the current prototype is focused at exploring the Linux containers features. The architecture was conceived to be generic and modular, enabling to experiment with other virtual machine managers. The *Ahoy master* implements two RESTful APIs the first is used to receive requests from the *Ahoy client* and the second to receive status information, send commands and jobs to the *Ahoy agents*. Both clients and agents need to communicate with the master. Since they are frequently behind firewalls the communication is always started by them. There are no permanently open communication channels. Connections are established by the agents periodically to send status information and pool for new jobs to process. From the point of view of the user most details are hidden, and he can use a set o simple actions to prepare and execute applications.

Creating base images With the *Ahoy client* tool the user can produce base operating system images on top which applications can be added. The aim is to empower the user to create base images of his operating systems, thus ensuring that applications will run on the same environment. The images are minimal by default but the user is free to add packages. These images are mainly produced with tools such as *yum* [23] or *apt* [24] depending on the flavor of the Linux distribution. Unfortunately creating such images requires privileges that may not be available to the user. In that case pre-built images available from the *Ahoy master* server may be used instead. The base images are stored in the *Ahoy master* repository, and become available as building blocks to create application images.

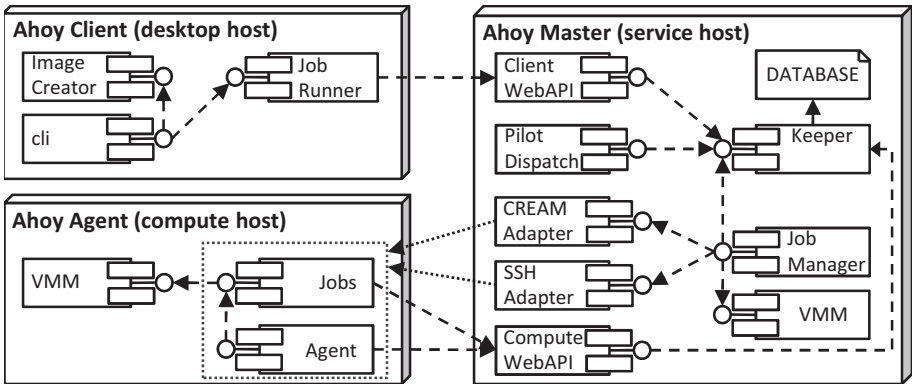


Fig. 2. Architecture diagram

Creating application images Application images are created by combining a base operating system image with the user specified software. The *Ahoy client* takes as input programs or directories to be merged with an existing operating system image. The operating system image is identified by its unique name and must already exist in the *Ahoy server*. The user can also specify additional software packages to be added to the final combined image, these are installed via *yum* or *apt*. The user does not need to specify the software dependencies since the *Ahoy client* is capable of analyzing the input files and packages and find their dependencies. This ensures that any required software packages and libraries are added. Finally the user may choose the program to be executed when the container is started, otherwise a wrapper will be the default executable. The *Ahoy wrapper* can hide much of the details of downloading input files and uploading results, in addition can also be used to download the executable thus making the application image more flexible. The actual operation of building an application image is performed on the *Ahoy master*. The resulting image is then made available by pushing it to a Docker repository to take advantage of the Docker layered images. The image can also be made available via a web server or via the *Ahoy master* itself.

Executing an application A job execution request contains: the name of the file to be executed, a list of input files and a list of output files. Additional information regarding the job requirements can also be added such as the amount of memory and number of processors. There is no need to pass information about the operating system software environment because those requirements are already encapsulated in the application image. In addition the mounting of volumes from the computing host can also be requested. The volume names must be coherent across the infrastructure. Each name is mapped by the *ADMINS* at each site into a local directory to be mounted in the container. The mapping is defined in the *Ahoy agent* configuration file. This feature allows access to data already available

at the site. The input files can be uploaded from the *Ahoy client* to the *Ahoy master* together with the job execution request. Alternatively the list of input files may contain references to files available from an external ftp or web server. The executable can be: a file previously added to the image, a file uploaded by *Ahoy client* jointly with the other input files, or a file to be downloaded from an external server via the *Ahoy wrapper*.

Upon receiving a request, the *Ahoy master* performs a validation and checks if there are free resources capable of executing the job. This is verified by checking the status and capabilities of the *Ahoy agents* that are active and ready to pull jobs. If enough *Ahoy agents* matching the request are available and free then the *Ahoy master* just waits to be contacted by one of them, at that moment the job is given to that agent. If no resources with the required characteristics are free, then the *Ahoy master* searches the list of potential dynamic resources that are alive and submits the required *Ahoy agents*. The *Ahoy master* can be configured to over provision agents thus allowing request to be quickly served.

The *Ahoy agent* has two modes of operation. A static mode where it runs as a persistent system process (daemon), and a dynamic mode aimed to be started on-demand by the *Ahoy master*. If a dedicated pool of compute hosts is needed then these nodes can be setup in static mode. The user workstation can also be setup as a static compute host. The agent is a lightweight process that can be launched manually whenever needed. The dynamic mode is more suitable for transient resources such as cloud and grid computing resources that may be available for a short period of time and have to be activated when needed. Currently the *Ahoy master* has adapters to support the activation of *Ahoy agents* in dynamic mode via SSH and via the grid cream-ce. The adapters encapsulate the details of starting the *Ahoy agents* in each type of resource. They are modular and easily implementable. The *Ahoy master* itself is also a lightweight process that can be installed and operated by an individual user to exploit resources at his disposal. However it requires inbound network connectivity.

Currently, starting the application images as containers requires privileges on the compute host. Therefore one of the functions of the *Ahoy agents* is to allow an unprivileged user to start the containers in a secure way. Both the *Ahoy master* and the *Ahoy agent* supports abstraction interfaces to perform container actions using virtual machine management systems. Abstraction interfaces for Docker and LXC are implemented. Deploying an agent implies the installation of LXC or Docker, as well as the agent itself. The installation requires privileges and therefore the cooperation of the *ADMINS*.

The abstraction interface for Docker needs to communicate with the Docker service via its protected UNIX socket. For security reasons only trusted users should have direct access to this socket. Therefore the *Ahoy agent* must be installed under a dedicated UNIX user with access to the socket. This can be easily accomplished by creating a dedicated unprivileged user and adding it to the docker UNIX group. The agent can then be started by end-users through the UNIX *sudo* command, which needs be configured to start the agent under the control of the dedicated user. This is the default approach followed by the *Ahoy master* to start the agents on computing hosts.

The agent can be configured by the *ADMINS* to impose limits such as memory consumption. The specification of volume names and their mapping into local directories is also accomplished in the same manner.

The *Ahoy agent* periodically contacts the master to send its status and check for applications to be executed. When it gets a request it downloads the application image either from a web server or from a Docker repository. In simple scenarios the *Ahoy master* can also serve the images directly from its internal repository. The image is then started by the agent with the appropriate parameters such as: local volumes to be mounted, resource usage restrictions, and also with job metadata that is translated into environment variables. Once started, the *Ahoy wrapper* inside the container uses the environment variables to perform actions such as downloading the input files and starting the application. Upon completion the wrapper transfers the output files back to the *Ahoy master* or to an ftp or web server. The *Ahoy agent* can then transfer the container log files, notify the master that the execution has finished, delete the image, and eventually exit. The *Ahoy agent* can log information about the execution of the application for accounting and auditing purposes which becomes locally available to the site administrators.

The *Ahoy master* periodically receives information about each job from the *Ahoy agent*. This information can be polled by the *Ahoy client*. When the job finished the user can download the output files from the master or other external location.

During execution the user can cancel the execution of its jobs, the request is sent to the *Ahoy master* and relayed to the agent upon one of its contacts. In such case only the execution logs are made available.

3.3 Findings

Docker actions are performed by a daemon process that requires root privileges to create network bridges, manipulate firewall rules, mount file systems, or extract image layers with files owned by others. It uses a Unix socket for communication with the clients using a RESTful API. However this protocol does not have authentication. Any user with access to the socket is thus able to have full control over the Docker operation.

By default the processes running in Docker containers appear both to the host and to the container as root processes. This may lead to traceability and auditing issues since it is not obvious who started and is actually executing the contained processes. However, it is also possible to execute Docker containers under unprivileged UNIX user and group identifiers. In this case both from the host and container perspectives the user running the processes will be same unprivileged user.

Docker can mount local directories inside containers. Therefore a user with access to the Docker can start a container with both root privileges and a host file system mounted in. In this situation the user can from inside the container change the host file system. Docker can also be told to remove several important security restrictions potentially allowing breaches from the container. Due to these and other features the Docker cannot be made directly available to the end-users.

In this respect Docker it is not different from most virtual machine managers, but unfortunately this constitutes a limitation to its direct use by end-users.

The ability to run as root inside the isolated containers is an attractive feature that akin to conventional virtualization largely contributes to the versatility of the containers. There are however considerations that should be carefully evaluated. Processes inside containers run directly on top of the host kernel, therefore security vulnerabilities in the kernel can be exploited by processes inside containers to gain access to the host system and to other containers. There are several measures that LXC and Docker perform to mitigate this and other risks, such as dropping some of the POSIX capabilities to reduce the root privileges, or the use of Seccomp to limit the access to system calls. It is highly advisable to enable SELinux or AppArmor in the host system to further constraint the attack surface in case of container breach. It is important to understand that the Linux containers stack is still fairly recent and not yet complete, therefore processes inside containers should preferably run without privileges.

Due to the extensive security restrictions imposed on containers there are operations that by default even the root inside a container cannot perform. Examples are: mounting file system, use raw sockets, use protocols such as GRE, create device nodes, change certain file attributes, load kernel modules. These limitations may in some circumstances create problems. For instance using GPUs requires direct access to devices. Fortunately many of these issues can be circumvented by passing appropriate directives upon container start.

The use of Docker with batch schedulers is challenging mainly because all processes within containers are started by the Docker daemon which is not controlled by the schedulers. This poses additional accounting and policy enforcement problems. In this regard LXC has the advantage of not requiring a daemon thus allowing the control of the processes by the scheduler. Furthermore LXC already supports the direct execution of containers by unprivileged users while still enabling root privileges inside the container via user namespace. This feature can simplify the use of containers with benefits in terms of security, traceability and accounting. However it requires both very recent Linux kernels and security components not yet widely available in the Linux distributions.

4 Conclusions and Future Work

LXC and Docker are powerful tools that ease the creation and deployment of containers. However they still have limitations in terms of security. Several issues have been identified, but LXC and Docker are still being heavily developed and many of the current limitations will be addressed as they become more mature.

Docker is currently more suitable to the development and execution of services, thus many of its design characteristics at intended to be used by privileged users. The evolution of LXC towards direct usability by unprivileged users seems to be more promising as a method to encapsulate applications. However the Docker layered images are more flexible, and they fit very well the incremental building and transfer of applications in distributed computing environments.

The *Ahoy* demonstrator is currently able to execute across the intended computing resources with the described limitations. It was possible to executed several common operating system environments and applications on these resources. We aim at continue the development of the presented architecture and framework to explore the coordinated deployment of scientific applications across heterogeneous computing resources. The next steps include: addressing the authentication issues, exploit cloud resources, enhance the current abstraction interfaces and improve the orchestration.

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. operating system-level virtualization http://wikipedia.org/wiki/Operating_system-level_virtualization
2. Solaris Zones https://java.net/projects/zones
3. P.-H. Kamp and R. N. M. Watson: Jails: Confining the Omnipotent Root. In Proceedings of the 2nd International SANE Conference, Maastricht, The Netherlands, May 2000.
4. OpenVZ http://openvz.org
5. Linux-Vserver http://linux-vserver.org/
6. Linux Containers https://linuxcontainers.org
7. libvirt LXC container driver http://libvirt.org/drvlxc.html
8. Namespaces in operation https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt
9. Application Armor http://en.wikipedia.org/wiki/AppArmor
10. Security Enhanced Linux http://selinuxproject.org
11. SECure COMPuting with filters https://www.kernel.org/doc/Documentation/prctl/seccomp_filter.txt
12. chroot http://wikipedia.org/wiki/Chroot
13. Control Groups https://www.kernel.org/doc/Documentation/cgroups/cgroups.txt
14. AnotherUnionFS http://aufs.sourceforge.net/aufs.html
15. Linux Capabilities https://www.kernel.org/pub/linux/libs/security/linux-privs/kernel-2.2/capfaq-0.2.txt
16. Docker http://www.docker.com
17. OpenStack https://www.openstack.org
18. CoreOS http://coreos.com
19. Apache Mesos http://mesos.apache.or
20. Flynn https://flynn.io
21. DEIS http://deis.io
22. Cream Computing Element http://grid.pd.infn.it/cream
23. Yum Package Manager http://yum.baseurl.org
24. Advanced Packaging Tool https://wiki.debian.org/Apt
25. G. van Rossum, Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

Session II: eInfrastructure users, tools and services

New Challenges of the Spanish ATLAS Tier-2 to address the Run-2 period of LHC

J. Salt[1], A. Pacheco Pages[2], J. del Peso[3], F. Fassi[1,4], A. Fernández[1], V. Lacort[1], M. Kaci[1], S. González de la Hoz[1], J. Sánchez[1], E. Oliver[1], M. Villaplana[1], V. Sánchez[1], A. Montiel[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
[4] LPNR, Faculté des Sciences, Université Mohammed V - Agdal , Rabat (Maroc)
Jose.Salt@ific.uv.es

Abstract. The goal of this work is to describe and establish the way of addressing the main challenges of Run-2 by the Spanish ATLAS Tier2. Starting from the general update of the ATLAS Computing Model, several action areas will be explained in this Framework.

1 Introduction

The Higgs boson Discovery acknoledged by the 2013 Nobel prize in physics is a major success for the experiments at LHC. The LHC physics will be pursued through step-wise increases in energy, instantaneous luminosity and data taking rate, with the goals: the search for new physics, the precise understanding of the standard model and the study of matter-antimatter asymmetry. ATLAS will have to manage with five times more of events with respect to Run-1 (2009-2013) .

During Run-1, 2 billion real events and 4 billion simulated events were produced in 2011 and the same in 2012. The Spanish ATLAS Tier-2 (ES-ATLAS-T2) operations and performances during this Run are reported in [1, 2].

Section 2 of this paper presents the update of the ATLAS Computing Model for Run-2. In section 3 the maintainance and operation of the ES-ATLAS-T2 Infrastructure is reviewed. One of the important issues of Run-2 is the fast access to the data. A Federated Data Storage System is being deployed in ATLAS and is discussed in Section 4. Section 5 is devoted to the improvements of the Distributed Analysis System to face with Run-2. The involvement of the ES-ATLAS-T2 in the construction of an Event Index catalogue will be discussed in section 6.

2 Update of the ATLAS Computing Model

The computing system for the ATLAS experiment has performed very well during Run-1. However, the considerable increase of energy and luminosity for the upcoming Run-2, which is due to start in 2015, has led to a revision of the ATLAS Computing Model. The LHC will run at the designed parameters for the center

of mass energy (14 TeV) and luminosity ($L = 10^{34} cm^{-2} s^{-1}$ with 25 ns bunch spacing). As a consequence, a factor five more data will be produced along with an increase of a factor 20 in the pile-up w.r.t. Run-1. To deal with this new scenario, changes in both the ATLAS software and the computing technology are required [3]. For the former, the most significant innovation is a software able to run in parallel on different CPU cores. As a result, a substantial amount of memory is saved: a parallel job running on an 8-core CPU uses 8 times less memory than eight jobs running on that CPU, being the processing time equivalent. Most, if not all, data centers of the ATLAS Collaboration are equipped with multi-core platforms and they already set a new batch queue for this purpose [4]. Even though this software have been tested successfully, the code must still be optimized to profit from new features of modern compilers.

Concerning changes related to computing technologies, the Distributed Data Management (DDM) has been improved to become more efficient on the Grid, and new computing technologies, in particular Cloud Computing, have been considered. As a matter of fact, about 20% of the total computing power in ATLAS is performed on Cloud Computing at present. In addition, a significant grow in the amount of data to be store also implies to improve the clean-up procedure of files to avoid disk storage becoming full quickly. This deletion procedure is performed in a centralized way. As an example, the occupancy of the DATADISK space-token is shown in Figure 1.

All these upgrades will be tested at a large scale during Data Challenge DC14, which is foreseen for the period July - September 2014. The three sites of the ES-ATLAS-T2 have done the necessary system implementations already to afford the challenge successfully.

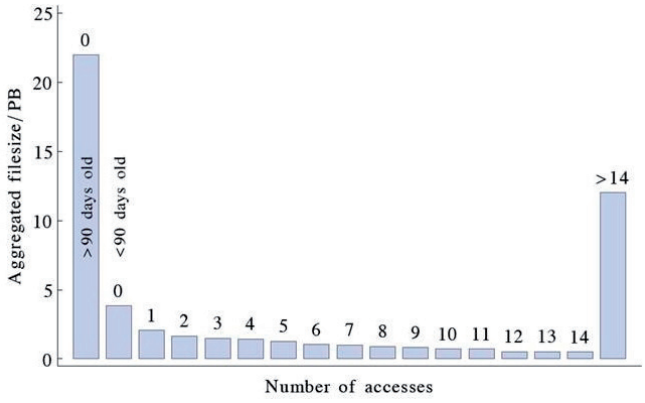


Fig. 1. ATLAS DATADISK: volumes of data versus number of accesses in the 90 days ending 14 March 2014. Data created in the last 90 days but not accessed are in the second bin. The total volume of all DATADISK is 52 PB. Data supplied by ATLAS.

3 Maintenance and Operation of the Infraestructure

The ES-ATLAS-T2 is a competitive project devoted to build, maintain and operate a Tier-2 Grid computing infrastructure dedicated to the ATLAS experiment. The following is an overview of the most important goals set for the next three years.

3.1 Evolution of the ES-ATLAS-T2 facing with Run-2

A Tier-2 site is a computer center which must meet very demanding requirements. According to the features established in the ATLAS Computer Model a regular Tier-2 must run round-the-clock every day of the year, and it must be served by personnel 12 hours a day during working days (12h/5d).

The ES-ATLAS-T2 project is a coordinated project formed by UAM (Universidad Autónoma de Madrid), IFAE (Institut d'Altes Energies, Barcelona) and IFIC (Inst. de Física Corpuscular, Valencia) being IFIC the coordinator center.

Table 1 shows the evolution of the pledges of CPU and Disk in ES-ATLAS-T2 . These numbers are the 5% of the total resources pledged for all the ATLAS Tier-2 centers, and represent the commitment of Spain for ATLAS.

Table 1. Pledges for ES-ATLAS-T2. The quantities for 2016 and 2017 are estimates.

T2-ES	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017
CPU(HS06)	92	243	1750	5390	10308	13900	13300	18000	20600	26100	30000	36875
DISK(TB)	14	63	387	656	1107	1880	2350	2550	2800	3250	3900	6875

ES-ATLAS-T2 is categorized as a Tier-2D. These are Tier-2 sites which may receive data from any other ATLAS Grid center in contrast with a regular Tier-2 which only receives data from centers that belong to the same cloud. A Tier-2D site must be stable enough to guarantee the data processing.

3.2 Expert System

The operation of a Tier-2 computer center requires certain expertise due to the complexity of the systems and the very demanding operational requirements. The difficulties to keep expert operators and managers working for the project for long periods lead us to start developing a tool to help the operation and management of the center.

The idea is to develop an expert system which can make automatic decisions whenever a problem appears on the site. The system will need to go through log files to obtain information about the problem and determine the cause based on learning experience. In addition, it will include a monitoring system of the computer center that is able to hide complexity and present to operators the information in a very simple and intuitive way. For those problems an automatic decision cannot be made or is too risky to make, the system will just notify the operators about the issue through the monitoring system.

For convenience, the same framework of the ATLAS detector monitoring and control can be used as online monitoring system. This framework is the Siemens WinCC Scada system [5]. A useful approach is to build a finite state machine on top of the Scada system. The different Tier-2 computing services appear as nodes on a tree structure. To illustrate this idea a simplified example can be seen on Figure 2. Using a color convention (for example: green for OK, orange for Warning and red for Faulty) it is very intuitive for operators to get a first sight of a problem. The operators can click on any box for a more detailed information about the problem.

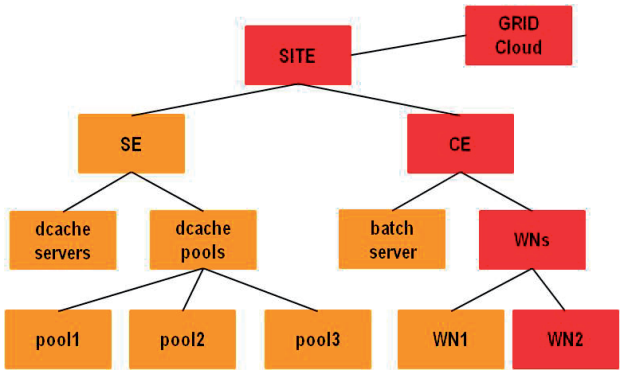


Fig. 2. An example of tree structure for a finite state machine of the monitoring of the expert system. In this example a site is composed of a storage element and a computing element. The storage element uses the dCache distributed storage system.

The initial development and the demonstration that the system can reduce the load of operators significantly will be done at UAM first. In a second stage, the system would be implemented to the other two centers, IFIC and IFAE, with the possibility to operate the centers in a distributed way. This distributed operation would allow that an operator from a center can solve issues of another center. If the system can be implemented successfully, the centers may be operated by people with less expertise, or by fewer expert system managers. Hence, the cost on personnel may be reduced considerably.

3.3 Progress in Multicore

AthenaMP is the multi-core implementation of the ATLAS software framework. It allows efficient memory pages sharing between multiple threads of execution while retaining event-based parallelism; through Linux kernel's Copy-On-Write mechanism (CoW) which is an effective memory sharing technique. This is now validated for production and delivers a significant reduction on the overall application memory footprint with negligible CPU overhead; however, with 32 cores or more, event-parallelism may stop scaling because of memory bandwidth issues.

Before AthenaMP can run routinely on the LHC Computing Grid, the computing resources available to ATLAS need an upgrade to exploit the notable improvements delivered by switching to this multi-process model. AthenaMP needs specific queues in batch systems, though Multicore and Singlecore queues can coexist in the same environment and underlying resources. Singlecore queues assign one core per job per core intending to maximize the available resources.

A Multicore queue in a batch system can be created using Static or Dynamic job resource allocation. Since Dynamic allocation is in a very early stage of development, Static allocation is broadly used by Grid sites today. Indeed, Dynamic allocations depend on the available features of sites batch systems and their job brokering mechanism (like Torque and Maui), that should support certain operations. It is desirable to have a way to decide which cores are available per node, mechanisms to avoid CPU and memory fragmentation, assignment of nodes to both kinds of queues (Single and Multi), different amount of cores required for Multicore jobs, and more specifications being analyzed. Job Description Language (JDL) is able to detail some requirements for multi-processing, like amount requested cores, total memory for the job or for core, if job requires all the cores of a node, min/max number of cores (optional), min amount of local disk space (optional), and similar.

Multi-processing may mimic single-process model, by setting number of cores to 1. Each site of ES-ATLAS-T2 has created a Multicore queue with one or two dedicated nodes for this first stage of Multicore processing deployment. Static allocation for job resources has been configured, based on the idea that the resources are only for multicore jobs, so no mix between multicore or monocore is possible in a CPU server. This simplifies the brokering and allocation of cores in nodes running monocore jobs. A usual approach is to assign eight cores per job, with the same wall time restrictions than current ATLAS single core jobs. By now the amount of Multicore jobs being processed is minimal, but increasing, compared to Singlecore ones as this new multi-process model is being using more and more resources among ATLAS. Although efficient Dynamic job scheduling needs to be addressed, increasing number of sites are providing resources for Multicore use; to maximize the use of these resources, sites need a consistent job flow. Information System on sites specifies the largest number of cores supported per job, and if site accepts Singlecore and/or Multicore processing, which will help to carry out a global scope dynamic scheduling.

In the long run multi-process model will be widely settled, by developing a more robust design to overcome current technical problems and meet best performance.

4 Towards a Federated Data Storage System

During most of Run-1 period in LHC, the processing of data within the ATLAS Computing Model has been done with strong requirement in the CPU-data locality. This fact had several non-positive effects, i.e., the multiplication of data replicas in different sites or the inability or running ATLAS Jobs in disk-less computing infrastructures. Due to this situation it was established a strategy to use data federation using XRootD called FAX [6]. FAX is a way to unify direct access to a

Diversity of storage services used by ATLAS. FAX brings together Tier-1, Tier-2 and Tier-3 storage resource into a common namespace, accessible from anywhere, thus relaxing the tradicional requeriment of data CPU locality [7–9].

The increases in network banwidth and data strucure aware of caching mechanism (such as TTreeCache) make this possible. The most important features of FAX is that it is a read-only access and it uses a global namespace. Moreover, this approach wants to cope with several use cases: (i) the failover form stage-in problems with local storage; (ii) to gain access to more CPUs using WAN direct real access with two main goals: allowing brokering to Tier-2 with parcial datasets, and, take profit of the opportunistic resources without local ATLAS storage; and (iii) use as caching mechanics at sites to reduce local data managements tasks. Therefore, the involvement of the Tier-2 sites as the spanish ones will be positive.

Since the start of the FAX initiative, several ATLAS sites belonging to different regions/clouds have joined the effort. Figure 3 shows the aggregated XRootD traffic corresponding to Jobs running under FAX approach.

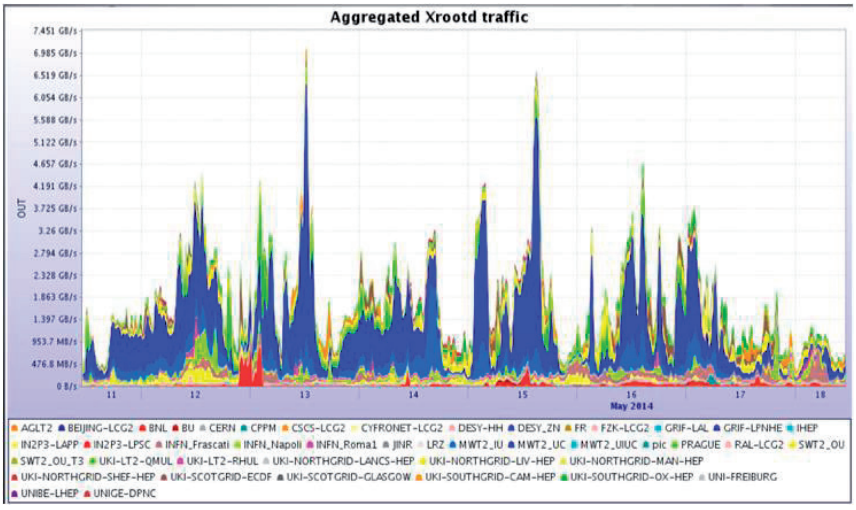


Fig. 3. Aggregated XRootD Traffic at various Tiers sites of ATLAS

The status regarding FAX in ES-ATLAS-T2 is as follows: the IFAE site is serving its disk space within the ATLAS FAX federation. Anyway, in all sites of ES-ATLAS-T2 PANDA is using FAX when a local copy of the input file fails. In the coming months the IFAE and UAM sites will proceed to be activated in the federation.

5 Distributed Analysis in Run-2

The Distributed Analysis in the ATLAS experiment has worked very well during Run-1, but several improvements are needed to speed up the workflow and take profit of more powerful computer architectures.

The central production of the common analysis data collections is much more scalable than the current system used during the first collision period of the LHC where all physicists were creating their own private data collections.

Moreover, as the collisions are in principle independent between them, the analysis of the data is by nature parallelizable. This is achieved by running the same program over diferent parts of the data at the same time but also this can be improved by running the same program using all the CPU cores available in the same server. This is known as multicore parallelism. In ATLAS this is achieved by using the PROOF [10] and PROOF-Lite [11] frameworks.

5.1 Analysis model: the Derivation Framework

The Derivation Framework, see Figure 4, is the key to the success of the Run-2 Analysis Model. The framework performs bulk data processing using the format xAOD [12] for the majority of physicists, producing targeted samples, DxAODs, which physics groups and users will later process themselves. A typical use case in Run-1 was that if one or more variables were missing from a particular private end-user collection of collision physics data known as "D3PD", the D3PD had to be recreated from scratch, wasting time and resources.

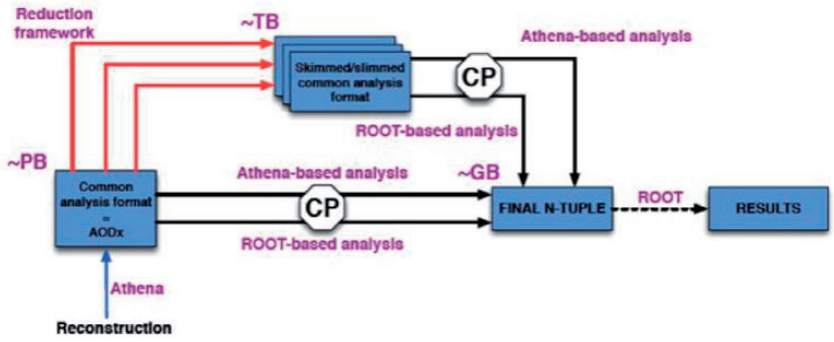


Fig. 4. Data Analysis Model for ATLAS Run-2: Derivation Framework.

The new analysis model attempts to tackle this problem by producing automatically common data collections for all physics groups in the DxAOD format as soon as new data is produced or a new version of the data processing software is available. The Derivation Framework is the chain of jobs that are executed regularly that produce either one or several common group data collections for a

given input provided by the detector or a simulation in a format known as xAOD. As reference to the railway system, the terminology of trains is used in reference to the numerous outputs produced, each individual output being referred to as a carriage.

The data to be analyzed is created with an initial calibration and alignment which is the best known at the time of producing the files, but later can be re-calibrated and realigned by the Combined Performance (CP) tools and will keep up-to-date all of the necessary variables for every carriage.

The computing resource constraints for Run-2 motivate the limit of the number of outputs to approximately 10, thus there will be several trains and in the simplest organisational model, each group has at least one train. The target output size for a derivation is 1 - 10 TB, based on feedback on group workflows summing over the 2012 dataset and MC samples. This output volume fits well with the computing resource constraints. Data reduction can be achieved either by skimming (removing events) or slimming (removing per-event information). Book-keeping tools will record the sum of the weights of events removed.

5.2 The Analysis release: xAOD and DxAOD in Root

The new Root-readable xAOD removes the need to convert xAOD into a Root-readable format, i.e. there is largely no need to make large D3PDs. They may be easily adapted to producing a directly Root-readable format rather than having intermediate datasets. When viewing an xAOD in a TBrowser, it looks like a Root TTree even if no xAOD class libraries are loaded. With the additional xAOD class libraries, the user has access to the full xAOD interface, which in most cases will be easier for use in analysis. In particular, the dual-use CP tools will use this xAOD interface, both in Root and in Athena. The Root-based Analysis release should be of order 100 MB in size, very lightweight and is also intended to serve the "disconnected" laptop use case, as well as being distributed on the grid. DxAODs will also look like a Root TTree and will likely be written in fully split mode for most derivations, meaning each variable lives in its own branch. This is exactly the same format as a D3PD, and yet the xAOD interfaces will still work as before, robustly ignoring variables that have been removed by slimming. Smart slimming in the Derivation Framework will guarantee that the variables needed by the dual-use CP tools will be available in the DxAODs.

5.3 use case of a real anaysis example

As it has been done in previous stages of the ATLAS experiment, each update in Computing must be verified and validated with a Data Challenge test. At this time, this test is called Data Challenge 14 (DC14). The Data Challenge 2014 will start in July and will end in September 2014, the goals is to test all software versions of the programs to be used in the Run-2 datataking period as well as all the workflows needed to collect from the detector, simulate and analyze the data. In addition the data from the Run-1 period will be reprocessed to have homogeneity in all the data. The DC14 test will check also the capacity planning and resource limits of

the Tier1 and Tier2 centers participating in the exercise. This is something similar to a full-dress rehearsal to know what is not operating efficiently.

6 The Event Index Project

Currently the ATLAS experiment has a catalogue of all real and simulated events in a database (TAGDB), implemented in Oracle,with the main purpose of the selection of events on the basis of physical variables, the identification of files that contain a list of events already selected by other means, and consistency checks on the completeness of event processing. Nevertheless the TAGDB is slow, intrinsically complex and at times unreliable [13–16].

The Event Index project [17, 18] consists in the development and deployment of a catalogue of events for experiments with large amounts of data, based on the previous work on the TAGDB, but improving it to reach good performance. The most innovative part of this project is the adaptation and application of the NoSQL technology for cataloguing data of a large experiment. This work is being developed and tested at ES-ATLAS-T2.

ATLAS accumulated in 2011-2012 2.10^9 real and 6.10^9 simulated events per year. If the production of events were uniform in time, this would correspond to a rate of creation of the records in TAGDB of about 100 Hz. In reality, each event is processed several times, so that in TAGDB the number of records is considerably larger.

The major use cases of the EventIndex project are:

- Event picking: give me the reference (pointer) to "this" event in "that" format for a given processing cycle.

- Production consistency checks: technical checks that processing cycles are complete (event counts match).

- Event service: give me the references (pointers) for "this" list of events, or for the events satisfying given selection criteria, to be distributed individually to processes running on (for example) HPC or cloud clusters.

6.1 Core architecture

NoSQL technologies offer several advantages over relational databases for applications like the EventIndex, as they scale linearly with the amount of data, there are many external tools to optimize data storage and data searches, they use clusters of low-cost Linux servers, including disks, and finally they belong to the world of open source software, so there is no license for installation and use of these products.

As CERN-IT is providing a Hadoop [19] service, and Hadoop and the many associated tools seem to be a good solution for the EventIndex, we developed our solutions based on plain HDFS (the Hadoop file system) and/or HBase [20] (the Hadoop database system). The data are stored in Hadoop map files which may be later upgraded into some of their sub-formats in case further optimisation is needed. The data can be indexed in various ways (for example using inverted

indices for the trigger information). Some index files will be created at upload, others will be added later.

Next Figure 5 (left) shows the file organization and relationships in the Hadoop file system (HDFS).The data are stored in collections of "filesets", where each collection represents an event collection (grouping all events that have been processed by the same software version). The TagFiles can be files or directories of files. A collection of TagFiles makes a TagSet, each of which has: a master record, pointers to before/after filesets (vertical partitions), slave TagFiles (horizontal partitions) and index TagFiles. The catalogue checks the presence and consistency of all components. Each TagFile is represented by one entry in the HBase table.

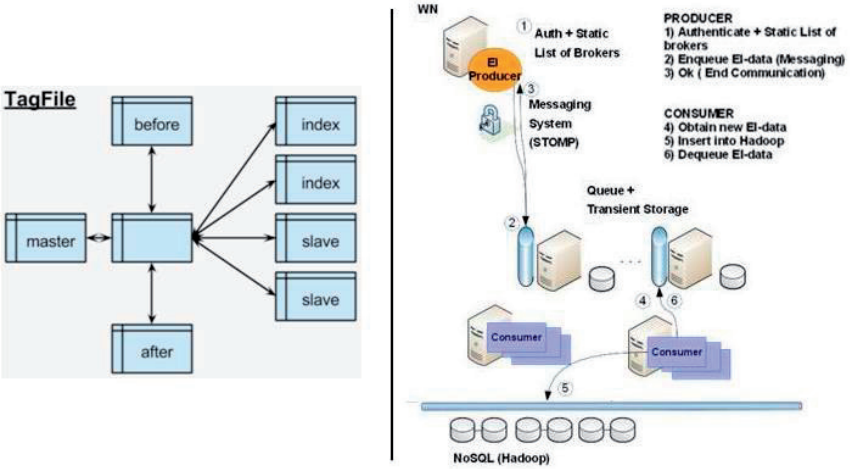


Fig. 5. Event Index Architecture: Tag Files and Data collection and Upload.

6.2 Data collection and upload

Data to be stored in the EventIndex are produced by all production jobs that run at CERN or on the Grid. For every permanent output file, a snippet of information, containing the file unique identifier and for each event the relevant attributes, has to be sent to the central server at CERN. The estimated rate (in May 2013, during the LHC shutdown) would be 20 Hz of file records containing 7 kHz of event records. During data-taking periods these numbers should be doubled; as both the data-taking rate and the Grid processing power are expected to increase by perhaps a factor two by 2015, the system should be ready to accept over 80 Hz of file records and insert into the back-end over 30 kHz of event records.

The architecture, as shown in Figure 5 (right), is based on producers of the EventIndex information on the worker nodes where event data are processed, a messaging system to transfer this information and asynchronous consumers that

effectively insert the data in the Hadoop database. The producers run within the "pilot" process on each worker node and transmit the data using a messaging system to a queue in one of the endpoints. ActiveMQ [21] has been chosen as the messaging service, and STOMP [22] as the message protocol, as they are supported at CERN. An ssl endpoint at the broker can be used to secure the communications, and we can also have replication of the message queues in other sites, to permit duplication and high availability. The consumers are asynchronous processes, continuously reading data. The task of the consumers is to get new data, check their validity with the production system, and pass them into Hadoop.

6.3 Data queries and retrieval

The performance of data query interfaces is crucial for the usability of the EventIndex. For the initial tests, we imported a full year worth of ATLAS data (2011, first processing version) from TAGDB into Hadoop and catalogued them. This corresponds to just over 1 TB of data in CSV format, before internal replication. Simple data scan operations on this dataset using the available Hadoop cluster of 20 nodes take about 15 minutes; we consider this as the worst case, as normally no query would need to read in all data from disk for a full year of data taking. Queries that give the run number and event number (the event picking use case) are very fast; even using the so-far unoptimised "accessor" method these queries return their results in 3.7 seconds. For comparison, the same queries using a full Map-Reduce job return in 90 seconds. Again, these should be considered as worst cases as they are the results of the first prototype implementation.

Conclusions and perspectives

Run-2 data taking will begin in 2015 and the ATLAS experiment has to address new challenges in Computing and Data Management. Under this circumstance it has arisen an update of the existing Computing Model and we focused on the most important aspects and more closely related to ES-ATLAS-T2.

First, the essential purpose of the Tier-2 is to provide computing infrastructure carrying increasing both CPU and disk which has been committed to ATLAS. Moreover, the maintenance and operation of this Tier-2 is our daily work. In the next period, it is forseen to provide an Expert System to help the administrators to manage the site in an automatized way. Each site of ES-ATLAS-T2 has created a Multicore queue with one or two dedicated nodes for this first stage. The amount of multicore Jobs is increasing and sharing with single core Jobs. Several non-positive situation data have originated an initiative to implement a Federated Data Storage System. The Federated ES-ATLAS-T2 is immersed in the deployment of this system which will be validated, like other elements, during the DC14 campaign.

Several improvements in the Distributed Analysis are needed and the Derivation Framework is emerging as the key to success for Run-2. All these changes in the Computing Model will also be checked an verified during the DC14 campaign and ES-ATLAS-T2 will participate on it.

Finally, the ES-ATLAS-T2 groups have opted for a clear research and development activity through participation and task leadership in the ATLAS Event Index Project. Currently it is in the deployment phase of the full chain and first operation.

Acknowledgements

This work has been partly supported by MICINN, Spain (Proj. Ref. FPA2010-21919-C03-01,02,03).

References

1. E. Oliver *et al.*, Operation of the ATLAS Spanish Tier-2 during the LHC Run I, Ibergrid 2013 Conference, September 2013, Madrid
2. V. Sánchez-Martínez *et al.*, Leasons learned from ATLAS performance Studies of the Iberaian Cloud for the first LHC running period, contribution to CHEP 2013, Amsterdam
3. I. Bird *et al.*, Update of the Computing Models of the WLCG and the LHC Experiments, CERN-LHCC-2014-014, LCG-TDR-002, http://cds.cern.ch/record/1695401
4. CERN openlab Whitepaper on Future IT Challenges in Scientific Research, May014, CERN OpenLab, http://dx.doi.org/10.5281/zenodo.8765
5. http//www.siemens.com/wincc
6. I. Vukotic, Data Federation Strategies for ATLAS using XRootD, talk at the CHEP 2013, Amsterdam (NL)
7. https://twiki.cern.ch/twiki/bin/view/AtlasComputing/AtlasXrootdSystems
8. https://indico.cern.ch/event/320296/contribution/4/material/slides/1.pdf
9. https://twiki.cern.ch/twiki/bin/view/AtlasComputing/UsingFAXforEndUsersTutorial
10. PROOF: http://root.cern.ch/drupal/content/proof
11. PROOF-LITE: http://root.cern.ch/drupal/content/proof-multicore-desktop-laptop-proof-lite
12. Report of the xAOD design group, CERN, ATL-COM-SOFT-2013-022
13. J. Cranshaw *et al.*, Building a scalable event-level metadata service for ATLAS, J. Phys. Conf. Ser. 119:072012, 2008, http://iopscience.iop.org/1742-6596/119/7/072012
14. J. Cranshaw *et al.*, Integration of the ATLAS tag database with data management and analysis components, J. Phys. Conf. Ser. 119:042008, 2008, http://iopscience.iop.org/1742-6596/119/4/042008
15. J. Cranshaw *et al.*, A data skimming service for locally resident analysis data, J. Phys. Conf. Ser. 119:072011, 2008, http://iopscience.iop.org/1742-6596/119/7/072011
16. M. Mambelli *et al.*, Job optimization in ATLAS TAG-based distributed analysis, J. Phys. Conf. Ser. 219:072042, 2010, http://iopscience.iop.org/1742-6596/219/7/072042
17. D. Barberis *et al.*, The ATLAS Event Index: an event catalogue for experiments collectiong large amounts of data, CHEP2013, 14-18 October 2013, Amsterdam (NL)
18. D. Barberis *et al.*, ATLAS Event Index prototype for cataloguing large amounts of data, EGI Forum, Helsinki, 19-23 May 2014
19. Hadoop: http://hadoop.apache.org
20. HBase: http://hbase.apache.org
21. ActiveMQ: http://activemq.apache.org
22. STOMP: http://stomp.github.io

Grid Scheduling Strategies For Applications That Need Data Transfer

Kiran Ali[†1], Marco Amaro Oliveira[‡2], Inês Dutra[§1,2,3]

[1] Departamento de Ciência de Computadores, Universidade do Porto
[2] INESC TEC Porto, Portugal
[3] CRACS

Abstract. This work describes a methodology and strategies to allow better execution performance of grid applications that make use of files located outside of the grid. Our methodology consists of two phases. The first phase, which is static, ranks machines in the grid according to file transfer sizes. The second phase uses this information to choose good machines to run jobs according to the amount of data transfer required. Initial results, using files of different sizes show that our methodology can reduce the failure rate in approximately 30%.

Key words: scheduling strategies, grid data transfer, grid computing

1 Introduction

Totally relying on a grid middleware, like gLite [5], to execute jobs that require data transfers can be very inefficient. The successful execution of these jobs depends on a good choice of machines in order to reduce the job failure rate. Results in the literature indicate that 20 to 50% of the total jobs submitted to a grid infrastructure may fail [7, 2, 6]. Jobs that require file transfers, such as the ones used to retrieve and plot spatio-temporal data, may have a higher failure rate due to various reasons. These jobs, very often, need to retrieve large amounts of data from the remote servers in order to process and visualize information. Retrieving data from a far remote server may incur problems such as:

- there is a need for storage as large as the data that is being retrieved;
- transfer of large amounts of data may not be feasible;
- even if it is feasible to transfer a large amount of data, it can cause undesired delays;
- the processed data needs to be stored locally and may be too big for the storage space available on the sandbox.

[†] e-mail of corresponding author: kiran.ali.awan@gmail.com
[‡] e-mail of corresponding author: mao@inesctec.pt
[§] e-mail of corresponding author: ines@dcc.fc.up.pt

Preliminary results [3, 4] with experiments dealing with large amounts of data show that besides distributing the processing, techniques based on prefetching and caching can help mitigating the delay problems. Using the grid to launch jobs for these kinds of applications may help solving some of the problems, but the high failure rate of the grid infrastructure may limit its potential.

In this work, we propose and evaluate a methodology to efficiently distribute jobs that need to transfer files from a remote server. We designed a planner that uses static information collected about datasources and computer resources, but that can dynamically change according to the system's behaviour.

In order to achieve our goal, we use profiling to classify the grid machines into two categories, fully responsive (Good machines) and limited-responsive machines (Bad machines). The profiling method runs different jobs on these machines to get the aforementioned classification. Based on the experimental results and this classification, we proposed a scheduler that selects resources according to the size of the data transfers. This process can be automated and easily performed with low complexity.

2 Methodology

Our experimental environment consists of machines that belong to the European Grid Infrastucture (EGI) and we used the `biomed` Virtual Organization (VO). All our jobs were submitted using the gLite middleware. Jobs were submitted from the GridUP User Interface, `ui01.up.pt`, located at University of Porto, Porto, Portugal.

The application used in our methodology fetches text files that contain binary annotations of temperature around the world from the Wold Climate Global Data website (http://www.worldclim.org). The reason for choosing this data is because it is available in different resolution sizes and is used by geospatial applications that fetch this data for processing and analysis. The data is stored on a website in four different resolutions: data collected in an interval of 10 minutes, 5 minutes, 2.5 minutes and 30 seconds. The size of the files corresponding to each of these resolutions is given in Table 1. Each instance of our application is called a job. Each job has a basic task of a spatio-temporal application processing, i.e., to fetch the files, in zip format, from the aforementioned website using WGET, unzip it and process it. It basically consists of a python program that fetches binary files, decodes the binary format and generates 12 comma-separated-value (CSV) files that contain monthly mean temperatures in the world. Each zipped file contains 12 * 2 files, two per each month. One of the two files contains the data in binary format. The second one contains the data header. Each generated CSV contains temperatures related to a given month. Size of the binary and of the CSV files vary according to the data resolution (30 seconds, 2.5, 5 and 10 minutes). The 10 Min resolution generates CSV files of 10 MBytes, the 5 Min dataset generates CSV files of 45 MBytes and the 2.5 Min dataset has CSV files of 180 MBytes. The average runtime for this processing, excluding file transfer, on a machine, for example, `svr014.gla.scotgrid.ac.uk:8443/cream-pbs-q1d`, for the 10 Min

resolution dataset, is 812 seconds. Similarly, for dataset 5 Min and 2.5 Min the processing took 3058 seconds and 12054 seconds, respectively.

Resolution	Size (MBytes)
10 Minutes	6.6
5 Minutes	22.5
2.5 Minutes	79
30 Seconds	1435

Table 1. File sizes of different datasets

In a first step, we determined all the computing elements (CE) available in the `biomed` VO.[4]

The total number of CEs available in the `biomed` VO is 188. Each one of them has many processors/cores.

Our methodology to design the planner is divided into three phases. These phases are illustrated in fig. 1. All of these phases are described as follows.

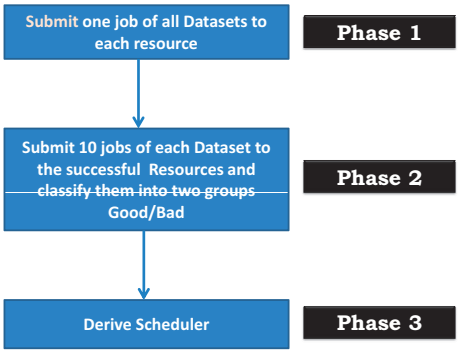


Fig. 1. Methodology of our Approach

Phase 1 and phase 2 are executed once to initialize and populate the tables. Once these tables are built, phases 1 and 2 will not be performed again. The scheduling strategies built using this methodology will only modify the tables reranking Good and Bad machines.

2.1 Phase 1

In the first phase, we submit four jobs corresponding to fetching one of the dataset sizes (10 minutes, 5 minutes, 2.5 minutes and 30 seconds) to each machine available

[4] The words: "machines", "resources" and "computing elements" are used interchangeably throughout this text.

in the `biomed` VO. This gives us a total of $188 * 4 = 752$ jobs submitted. Once these jobs are terminated, results are collected from the WMS. A job is declared successful if it executes without any error. Errors are categorized into two classes:

1. *the job finished with a status Done and an exit code different from zero.*
   This means that the job terminated, but some failure occurred in the middleware (such as a grid site misconfiguration problem, or a version of python not compatible)
2. *the job finishes with a status Done and an exit code zero, but did not terminate properly.*
   For example, it may happen that the job is aborted because the proxy expired.

In this first phase, we want to rule out machines that are not responding or can not run the jobs for some reason. For example, a machine may fail the execution of a job because: (1) it may not have the python version or libraries we need, (2) may not be responding or (3) may not be properly configured. In the end of the first phase, we keep only the machines that successfully executed the job for the corresponding file size.

2.2 Phase 2

In the second phase, we will check the robustness of the successful machines of the first phase by submitting various jobs of the same dataset size for each of these machines. For example, if machines M1 and M2 successfully executed the 5 minute dataset then we submit a batch of 10 jobs of 5 minute resolution to both M1 and M2.

Similarly, the other successful machines for the other dataset sizes are selected and they are tested with a batch of 10 jobs of the same dataset. When the jobs are completed, we retrieve the output files from the WMS. The machines are collected per each dataset size and further divided into two groups: 1) Good machines and 2) Bad machines. The purpose of the further evaluation of machines for each dataset size is to perform the classification of machines according to their performance while executing each dataset. The machines that managed to successfully execute at least a single job are declared as Good machines for that particular dataset.

2.3 Phase 3

Based on the results obtained with the first two phases, we are ready to create strategies that can avoid selecting Bad machines to execute our jobs. We have proposed two different scheduling algorithms based on the results of the previous phases to rank the available machines in the grid to increase the efficiency. After ranking, we choose the best ranked machine according to the data size. These two scheduling algorithms are named: (1) Class based Scheduling Algorithm (CBSA) and (2) Global Scheduling Algorithm (GSA).

Each algorithm has two versions: offline and online. The offline version is used to build the initial machine ranking. The online version is used during execution and can modify the original rank according to the machine's dynamic behaviour.

Class Based Scheduling Algorithm (CBSA) In the class based scheduling algorithm (CBSA), we initially create clusters based on the results of the first two phases. Assume there are n datasets used to determine the good or bad machines. We create $n+1$ clusters numbered from 1 to $n+1$. In our case there are 4 datasets and hence the number of clusters will be 5. A machine in the grid that is good for p datasets is added into a cluster $n-p+1$. For example, a machine which is good for all the datasets will be added into cluster number 1. All the machines in the grid are added to their corresponding clusters. Afterwards, machines inside the cluster are ranked. The ranking inside a cluster is performed based on a metric called κ, defined as follows.

$$\kappa = \frac{\text{Total Execution Time of all Successful Jobs}}{\text{Total Number of Jobs}} \tag{1}$$

The total execution time of successful jobs is divided by the total number of jobs. All the machines inside each cluster are sorted in ascending order with respect to their corresponding values of κ.

Similarly, we also propose another method to rank the machines inside the clusters as well. The machines are ranked differently for each dataset in each cluster. The specific dataset will consult the corresponding ranking of its dataset inside the cluster. Assume, there are 4 datasets and we are interested in cluster number 1. The proposed algorithm creates four tables inside cluster number 1 for each dataset. The ranking of the machines in each table is performed based on the number of successful jobs executed in the previous phase for that specific dataset. Hence, if the machine needs to be selected for a dataset b, all the machines inside the clusters corresponding to dataset b are selected to give the overall ranking.

Algorithm 1 Class based Scheduling Algorithm

Offline Phase: Rank Reliable Machines
Assume there are n different datasets
Create $n+1$ clusters, numbered from 1 to $n+1$
A machine that successfully executes p datasets will be allocated to $n-p+1$
i) Rank the machines within each cluster by a descending order of κ,
where $\kappa = \frac{\text{Total Execution Time of all Successful Jobs}}{\text{Total Number of Jobs}}$
OR
ii) Create table for each dataset inside each cluster and rank the machines in each table with respect to the number of jobs executed in phase 2 of that specific dataset
The ranking of the machines in this phase starts from the first machine in cluster 1 and ends with the last machine in group $n+1$

Online Phase: Dynamically Updating the Ranking of the Machines
Assume a machine M_i and its current cluster number h
if (M_i is unsuccessful for x times (where $x \geq 1$) AND $h \neq n+1$) **then**
    Demote M_i from cluster h to next cluster $h+1$,
else if (M_i is successful for y times (where $y \geq 1$) AND $h \neq 1$) **then**
    Promote M_i from h to $h-1$
end if

Irrespective to the method used to sort the machines inside a cluster, the overall ranking of the machines starts from the first machine in the first cluster to the last machine in the $n+1$ cluster. The machines in the last cluster $n+1$ can be ordered with respect to their indice or alphabetical order of their names.

In the online phase, if a machine M_i is successful for x times then it is promoted to the next best cluster, i.e., its current cluster number minus 1. The value of x can be set to any integer greater or equal to 1. Similarly, if a machine M_i has unsuccessfully executed the y jobs, then it is demoted by one cluster, i.e., its current cluster number plus 1. However, obviously, if the machine is in cluster number 1 it cannot be promoted anymore and similarly, a machine in a last cluster $n+1$ cannot be further demoted. We do not remove the machines from the cluster based on the bad performance but we keep them in the $(n+1)th$ cluster. This decision is made on the assumption that some of the machines can perform well in the future. The pseudo-code presented in Algorithm 1 shows these steps.

Global Scheduling Algorithm(GSA) The global scheduling algorithm (GSA) globally ranks the machines based on the results given in phase 2 of our methodology. We run total number of α_i jobs on each machine M_i for all datasets. Assume α_i^j is the total number of jobs of j^{th} dataset executed on M_i. Hence,

$$\alpha_i = \sum_{j=1}^{n} \alpha_i^j,$$

where n is the total number of datasets. Assume β_i^j and γ_i^j are the successful and unsuccessful jobs of j^{th} dataset on machine M_i, respectively. The execution time of the successful jobs can be easily computed from their results. However, the unsuccessful jobs should be penalized with extra time. In order to do so, we assume a job that is unsuccessful on a machine has taken a time of U time units. The value of U can be set to any arbitrary big number greater than the successful job execution time. We compute a metric λ_i^j given in (2) for each dataset j on each machine. For each dataset j, all the machines in the grid are ranked in ascending order with respect to their λ_i^j values. If two machines have the same λ_i^j, their ties are broken based on their indice. This procedure is repeated for each dataset and finally, we have a global list of machines ranked with respect to λ_i^j for each dataset.

$$\lambda_i^j = \frac{(\text{Execution Time of } \beta_i^j \text{ jobs}) + (\gamma_i^j * U)}{\alpha_i^j} \tag{2}$$

In the online phase of this scheduling algorithm, the ranking of each machine is dynamically changed based on its performance. Assume a machine successfully executed a job of dataset j then its λ_i^j is re-evaluated by adding the execution of this new job and its ranking in the corresponding dataset list is updated accordingly. This procedure is performed to penalize a machine if it has unsuccessfully executed a job. The reevaluation of λ_i^j may be expensive to perform on completion of each job. This problem can be solved by storing the results of x jobs of dataset j and then the value of λ_i^j can be collectively updated for its x jobs. Please note that the value of $x \geq 1$. The pseudo-code of GSA is presented in Algorithm 2.

Algorithm 2 Global Scheduling Algorithm

1: **Offline Phase: Global Ranking of Machines**
2: Assume there are n different datasets. Now consider a machine M_i. Let β_i^j and γ_i^j be the number of successful and unsuccessful jobs on machine M_i respectively for dataset j. Total number of jobs submitted on M_i is equal to $\alpha_i = \sum_{j=1}^{n} \alpha_i^j$. Similarly, the total number of jobs of dataset j is equal to $\alpha_i^j = \beta_i^j + \gamma_i^j$
3: A penalty of unsuccessful job execution is U time units, where U is a very big number.
4: For each machine M_i, and each dataset j
compute $\lambda_i^j = \dfrac{(\text{Execution Time of } \beta_i^j \text{ jobs}) + (\gamma_i^j * U)}{\alpha_i^j}$
5: Rank all the machines in ascending order with respect to their corresponding value of λ_i^j for each dataset j
6: Ties are broken with respect to their indice

7: **Online Phase: Dynamically Updating the Ranking of the Machines**
8: Assume a machine M_i has completed a job k of dataset j
9: **if** (M_i has unsuccessfully executed a job k of dataset j) **then**
10: $\gamma_i^j = \gamma_i^j + 1$
11: **else if** (M_i has successfully executed a job k of dataset j) **then**
12: $\beta_i^j = \beta_i^j + 1$
13: **end if**
14: $\alpha_i^j = \alpha_i^j + 1$
15: Update $\lambda_i^j = \dfrac{(\text{Execution Time of } \beta_i^j \text{ jobs}) + (\gamma_i^j * U)}{\alpha_i^j}$ of M_i
16: Re-rank this machine for dataset j

3 Results obtained in Phase 1

As mentioned previously, we have used the VO Biomed and the EGI. The jobs are submitted from the GridUP user interface, `ui01.up.pt`. The jobs were defined using a JDL script which invokes a python program and collects timings from the execution phase (start, start fetching file, start processing, end processing). We used a shell script that selects the resource and submits the jobs of each one of the datasets.

In the first phase, we submitted a job of each dataset resolution on all machines available in the grid. While submitting the jobs of 10 minute and 2.5 minute resolutions, the grid infrastructure had 188 machines available. However, at the time of submission of 5 Min and 30 seconds dataset jobs, it had only 185 and 184 machines, respectively. A subset of the machines that presents the results corresponding to different machines for different datasets in the first phase are available in Table 2.

A machine that successfully executes the job in phase one is marked as *Success*, while the machine that failed to execute the job is represented as *Failed*. For example, machine number 83 in Table 2 (*cream-ce02.marie.hellasgrid.gr:8443/cream-pbs-biomed*) has successfully executed the 10 and 2.5 Min dataset job, while failed

to execute the 5 Min dataset in the first phase. Note that the column corresponding to the 30 second dataset is omitted in the tables. The job corresponding to the 30 second dataset in the first phase failed on all the machines available in the grid infrastructure because the output sandbox in grid infrastructures has a limit in size, we can not fetch files whose total size exceeds the size limit imposed by the grid user sandbox.

No	Machines In the Grid	10 Min	5 Min	2.5 Min
1	arc-ce01.gridpp.rl.ac.uk:2811/nordugrid-Condor-grid3000M	Failed	Failed	Failed
2	arc-ce02.gridpp.rl.ac.uk:2811/nordugrid-Condor-grid3000M	Failed	Failed	Failed
3	arc-ce03.gridpp.rl.ac.uk:2811/nordugrid-Condor-grid3000M	Failed	Failed	Failed
4	cale.uniandes.edu.co:8443/cream-pbs-biomed	Success	Success	Success
5	cccreamceli09.in2p3.fr:8443/cream-sge-long	Failed	Failed	Failed
⋮	⋮	⋮	⋮	⋮
79	cream-ce02.cat.cbpf.br:8443/cream-pbs-biomed	Failed	Success	Failed
80	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid1000M	Success	Success	Success
81	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Success	Success	Success
82	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Success	Success	Success
83	cream-ce02.marie.hellasgrid.gr:8443/cream-pbs-biomed	Success	Failed	Success
84	cream.afroditi.hellasgrid.gr:8443/cream-pbs-biomed	Success	Success	Success
85	cream.egi.cesga.es:8443/cream-sge-GRIDEGI_large	Success	Success	Success
86	cream.grid.cyf-kr.edu.pl:8443/cream-pbs-biomed	Failed	Success	Failed
⋮	⋮	⋮	⋮	⋮
185	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Success	Success	Success
186	t2-ce-01.to.infn.it:8443/cream-pbs-biomed	Failed	Failed	Failed
187	tochtli64.nucleares.unam.mx:8443/cream-pbs-biomed	Success	Success	Success
188	wario.univ-lille1.fr:8443/cream-pbs-biomed	Failed	Failed	Failed

Table 2. Successful and Failed Machines of Phase 1

In the first phase, for the 10 Min resolution dataset, out of 188 jobs (submitted to 188 machines) only 89 jobs successfully executed the jobs and 99 jobs failed. It means that only 47% of the machines were successful. Similarly, for the 2.5 Min resolution dataset, 84 jobs were successful and 104 failed. The percentage of the successful machines is equal to 44.6%. As the number of machines available for the 5 minutes resolution dataset was 185, out of 185 submitted jobs, 96 jobs successfully executed and 89 failed. In this case, the percentage of successful machines is 51.89%.

For 30 second resolution, 184 machines were available in the grid infrastructure and one of them was successful. So the successful percentage is close to zero. These results are summarized in Table 3.

Dataset	No of Machines Used	Tot. Submitted Jobs	Success	Failed	% of Success Machines
10 Min	188	188	89	99	47
5 Min	185	185	96	89	51.8
2.5 Min	188	188	84	104	44.6
30 Sec	184	184	0	0	0

Table 3. Summary of First Phase Results

No	Machines In the Grid	10 Min
1	cale.uniandes.edu.co:8443/cream-pbs-biomed	Good
2	ce-01.roma3.infn.it:8443/cream-pbs-grid	Good
3	ce.fesb.egi.cro-ngi.hr:8443/cream-pbs-sunx2200	Good
4	ce.hpgcc.finki.ukim.mk:8443/cream-pbs-biomed	Good
5	ce.irb.egi.cro-ngi.hr:8443/cream-pbs-hpdl580	Good
⋮	⋮	⋮
37	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid1000M	Good
38	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Good
39	cream-ce01.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Good
40	cream-ce01.marie.hellasgrid.gr:8443/cream-pbs-biomed	Good
41	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid1000M	Good
42	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid2000M	Good
43	cream-ce02.gridpp.rl.ac.uk:8443/cream-condor-grid3000M	Good
⋮	⋮	⋮
86	svr014.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
87	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q1d	Bad
88	svr026.gla.scotgrid.ac.uk:8443/cream-pbs-q2d	Bad
89	tochtli64.nucleares.unam.mx:8443/cream-pbs-biomed	Good

Table 4. Good/Bad Machines for the 10 Min Dataset of Phase 2

4 Results obtained in Phase 2

In phase 2, for each dataset, we categorize Good and Bad machines according to the results of phase 1. A subset of those machines corresponding to the 10 Min resolution dataset is presented in Table 4. The complete list for each file size dataset generated in phase 2 is presented in [1].

For the second phase we analysed the percentage of successful jobs. A summary of the results is presented in Table 5. For the 10 minutes resolution, 89 machines were successful in the first phase. We submitted a batch of 10 jobs of the 10 minutes resolution dataset on each of these machines. Therefore, a total of 890 jobs were submitted to these 89 selected machines. Among those 890 jobs, 661 jobs successfully completed their execution, while 229 jobs failed. This gives us a success rate of 74%, which is a very good improvement when compared with the experiment with the 10 minutes resolution dataset during the first phase (47%). These numbers show an improvement of about 30%, for this file size, over an

Dataset	Machines Used	Jobs Submitted	Failed	Success	% of Success	Machines % of Good
10 Min	89	890	229	661	74	76
5 Min	96	960	398	562	58.5	71.8
2.5 Min	84	840	279	561	66.7	72.6
30 Sec	0	0	0	0	0	0

Table 5. Summary of Second Phase Results

execution that indiscriminately chooses any machine of the VO. Similar results are shown for the other file sizes.

It is important to note that in a dynamic environment like the grid, decisions cannot be taken statically, therefore our algorithms allow for continuous learning where, during execution, machines are dynamically reranked.

5 Conclusions and Future Work

Grid infrastructures offer a good opportunity for harnessing resources of all types, which are made available through coordinated sharing policies. Grid middleware allows researchers to access these resources through Virtual Organizations in order to advance their science using the resources available to make experiments. However, despite the success of utilization of this infrastructure by researchers, there is still scope for improvements. First, there are still several communities that could benefit from using those resources, but do not know how. Second, various reasons prevent more people from using grid infrastructures: (1) the bureaucracy involved in using grids (for example, obtaining certificates or the need to have an account on a user interface), (2) the overheads of grid services due to the various layers of software and centralized servers, (3) the high rate of job failures, and (4) the change in methodology and environment to execute applications.

In this work, we show that totally relying on a grid middleware, like gLite, to execute jobs that require data transfers can be very inefficient. The successful execution of these jobs depends on a good choice of machines. Specially when applications need to transfer and process remote data. We have shown that a careful selection of resources can reduce the failure rate in 30%. Based on these results, we proposed a methodology to select machines when starting a grid application according to the size of the data transfers. We also proposed various scheduling strategies that can be dynamically adapted during execution using the initial rank of machines that were selected to start the application.

Besides providing a methodology to build scheduling strategies, the information collected through the tables can also be useful to system administrators and other users.

One of the issues not addressed in this work is the size limit of the user's sandbox. Our methodology and strategies assume that the data can fit in the sandbox. One interesting path to follow would be to use the same methodology to rank machines based on data stored on the Storage Elements. Actually, in this

work, we fetch data from a central server. We could use the Storage Element to store several of the datasets with different resolutions and guide the choice for a machine according to the distance to the data source chosen.

Acknowledgements

We would like to thank Rui Ramos and Sérgio Afonso for all help with the GridUP infrastructure and support given. This work was supported by the Fundação para a Ciência e Tecnologia (FCT/Portugal).

References

1. Kiran Ali. *Grid Scheduling Strategies for Applications that require Data Transfer*. PhD thesis, University of Porto, Department of Computer Science, Porto, Portugal, Feb 2014.
2. Konstantinos Christodoulopoulos, Vasileios Gkamas, and Emmanouel A. Varvarigos. Statistical analysis and modeling of jobs in a grid environment. *Journal of Grid Computing*, 6(1):77–101, 2008.
3. Alexandre Valle de Carvalho, Marco Amaro Oliveira, and Artur Rocha. Retrieval of very large temporal datasets for interactive tasks. In *Information Systems and Technologies (CISTI), 2013 8th Iberian Conference on*, pages 1–7, June 2013.
4. Alexandre Valle de Carvalho, Marco Amaro Oliveira, and Artur Rocha. Improvements to efficient retrieval of very large temporal datasets with the travellight method. In *Information Systems and Technologies (CISTI), 2014 9th Iberian Conference on*, June 2014.
5. gLite 4. *User Guide*. http://www.electro.fisica.unlp.edu.ar/eela/docs/gLite-3-UserGuide.pdf.
6. Edgard Quirino Neto. *Performance Analysis of a Grid Infrastructure with an Application in Spatio-Temporal Data Transfers*, June 2012. Technical Report, Department of Computer Science.
7. Demetrios Zeinalipour-Yazti, Kyriakos Neocleous, Chryssis Georgiou, and Marios D. Dikaiakos. Failrank: Towards a unified grid failure monitoring and ranking system. In Marco Danelutto, Paraskevi Fragopoulou, and Vladimir Getov, editors, *CoreGRID Workshop - Making Grids Work*, pages 247–259. Springer, 2007.

Py4Grid, a user centred tool for the long tail of science

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

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

Abstract. Py4Grid is a simple command line tool that abstracts users from the complexity of grid infrastructures and middlewares. It is developed in python and provides job configuration and data management functionalities in a transparent way. It follows a modular architecture with components that deal with proxy creation, script generation, job submission and file transfer from and to remote sites. Py4Grid has been especially developed to engage with the long tail of science which normally struggles to overcome the grid learning curve.

1 Introduction

Grid computing has shown that large scale data simulation, processing and analysis are possible despite the underlying complexity of joining distributed computing resources. An emblematic use case is the one of High Energy Physics (HEP), which due to the well structured nature of HEP experiments, was able to adjust to the heterogeneity of federated distributed resources. To take full advantage of the grid, these scientific communities have implemented frameworks that receive the user payload, submit thousands of tasks simultaneously, verify the environment, execute the scientific workload, resubmit it in case of unsuccessful executions, and blacklist the resources with high failure rates. Such frameworks provide an easy grid fronted for some predefined activities, and are mostly used by HEP for the central generation, processing and distribution of the data. Nevertheless, the end-scientists responsible for the fine-grained physics analysis, continue to prefer standard local computing infrastructures such as laptops, desktops or computing clusters. Higher flexibility, easier interaction, better support and easier application debugging are common reasons that justify their choice.

The heterogeneity and complexity of grid infrastructures are even a more complex challenge to research fields which, due to the nature of their work, are intrinsically more unstructured. Lifesciences, Structural Biology or Hydro-Meteorology are emblematic examples. These communities support multiple workflows, executed by users either working incoherently or in cooperation, in the same or in different domains. To aim for an easier interaction with grid technologies and to increase the success rate of most applications, those communities often choose for the deployment of dedicated scientific gateways. A scientific gateway is the common designation for a set of web or desktop collaborative tools providing users

with a Graphical User Interface (GUI) access to the eScience environment. Gateways are normally application or community specific. Very few are focused on multidisciplinary research and can not be re-used under different scientific contexts. As a consequence, the long tail of science, where researchers mostly work independently and without profiting from any community effort, are left with the standard job submission and data management grid command line client tools. For non-experienced users, unaware of grid architectures, and with a strong need for computing resources, the interaction with the grid is normally frustrating and error-prone.

In this paper we present `Py4Grid` [1, 2], a simple command line tool user interface that abstracts users from the middleware complexity. It was specially developed to allow a simple engagement with long-tail science researchers which know how to profit from local computing resources but are completely unaware of how the grid is supposed to work. `Py4Grid` abstracts the technical details regarding distributed architecture and service deployment. The main idea is to allow grid job submissions and transfers of big data files in an automatic and seamless way from an user perspective. In the following chapters we will discuss `Py4Grid` design as well as some of the implementation strategies.

2 State of the Art

The CERN LHC HEP communities were the first ones dealing with the difficulties of operating and using large scale federated distributed resources. To address the high failure rates from executions in heterogeneous environments, CERN collaborations developed their own job and data management frameworks: PANDA [3] from ATLAS, CRAB [4] from CMS and DIRAC [5] from LHCb experiment. All use pilot jobs to increase the success rate of executions. Pilots are empty resource reservation containers that perform local sanity checks and pull down user payloads for execution.

Soon it became evident that HEP suites were difficult to extend to other contexts, and other tools emerged to address multidisciplinary scientific requirements. One successful long-living approach is the Migrating Destop (MD) [6–8], still supported and widely used. The MD is a GUI front-end for the Roaming Access Server, a dedicated set of web services to cover a full application lifecycle, from job definition, launching, monitoring until visualization of job results. It has been designed using the OSGi framework, a modular platform for `JAVA` that implements dynamic component model. The `JAVA` based implementation provides easy portability to multiple operating systems. Moreover, the MD can be installed in laptops or desktops and used behind private networks or firewalls.

Today, GANGA [12] is one of the most (if not the most) popular non-specific software suite that researchers can deploy. It is based in `PYTHON` and aims for easy job submission and management. It provides client command tools, a Graphical User Interface (GUI), and a WebGUI. A job in GANGA is constructed from a set of building blocks. All jobs must specify the software to be run (application) and the processing system (backend) to be used. Many jobs will specify an input dataset to be read and/or an output dataset to be produced. Therefore,

GANGA provides a framework for handling different types of applications, backends and datasets, implemented as plugin classes. Pragmatically, this means that GANGA can be used to submit jobs to the localhost where it is installed, to a local farm or to a computing grid, as long as the appropriate client command tools are available. GANGA GUI, besides basic job management functionalities, provides a configurable job monitoring window that keeps track of running jobs and their status, organises past jobs and quick-scripting tools to run favorite code snippets all within an integrated graphical environment.

DIANE [13] is a lightweight job execution control framework for parallel scientific applications to improve the reliability and efficiency of job execution by providing automatic load balancing, fine-grained scheduling and failure recovery. The backbone of DIANE communication model is based on pilot jobs architecture. DIANE uses GANGA to allocate resources by sending worker agent jobs. The DIANE framework takes care of all synchronization, communication and workflow management details on behalf of the application. The execution of a job is fully controlled by the framework which decides when and where the tasks are executed. Thus the existing applications are very simple to interface as python plugin modules. Application plugin modules contain only the essential code directly related to the application itself without bothering about networking details.

Scientific gateways are also widely adopted by research communities providing a wide set of collaborative tools as a service to their users. One of the most popular generic grid portals is gridsphere [9]. Other gateway families focused on managing basic user interaction with the grid are the gridPort [10] and GENIUS [11] frameworks. In what concerns oriented community gateways, there are many worth to mention such as the BIRN Portal for biomedical imaging, RENCI BioPortal for biology, GEON Portal for geosciences, Cactus portal, ServoGrid Portal, NEES grid Portal for earthquake science, NanoHub for nanotechnology and nanoscience and VODesktop for astronomy.

3 System Design

`Py4Grid` is a user-centric tool developed in `PYTHON`. The software is distributed as tarballs under the GNU General Public License 2. It is available for download in [1, 2]. The installation procedure is trivial and does not require any privilege access. The full process consists on the extraction, uncompression and deployment of the `Py4Grid` components and modules in a central or user dedicated area, followed by the environment customization (`$PATH` and `$PYTHONPATH`). The tool must be deployed on a grid User Interface where (any version of) the Unified Middleware Distribution should be already available.

`Py4Grid` follows a cluster-wise philosophy where users can define their execution envelope and requirements through a configuration file, submit the job, query job status and retrieve job output results. Four different components are available: `JobSubmit` for job submission, `JobState` for job monitoring, `JobGetOutput` for job output retrieval and `JobDelete` for job deletion and purger. Each component relies on multiple and reusable modules that can be used by more than one component. Their main objective is to guarantee the interoperability with the grid

services. Figure 1 presents a snapshot of the `Py4Grid` components as well as their interactions with the different modules. A brief description of each module follows:

- `Py4Grid-CONF`: Core configuration module that parses the user configuration file and customizes the execution environment.
- `Py4Grid-VOMS`: Interoperates with the Virtual Organization Management Service (VOMS) to test the validity of user credentials and VO membership.
- `Py4Grid-LFC`: Create and verifies the existence of unique namespaces at the LCG File Catalogue (LFC) based on the user's Distinguish Name (DN).
- `Py4Grid-UI2SE:` Uploads big data files to the closest Storage Element (SE) at job submission time, and registers Logical File Names (LFN) in the LFC.
- `Py4Grid-PROLOG/EPILOG:` Creates prolog/epilog scripts to guarantee the seamless transfer of big input data files (at the beginning of user payload execution), and big output data files (at the end of the user payload execution).
- `Py4Grid-JDL`: Builds the Job Description Language (JDL) script.
- `Py4Grid-SE2UI`: Pulls big output data from Storage Elements back to the user.
- `Py4Grid-SEDEL`: Purges all (relevant) data from SEs and LFC.
- `Py4Grid-WMS`: Job management module to interoperates with the Workload Management System (WMS).

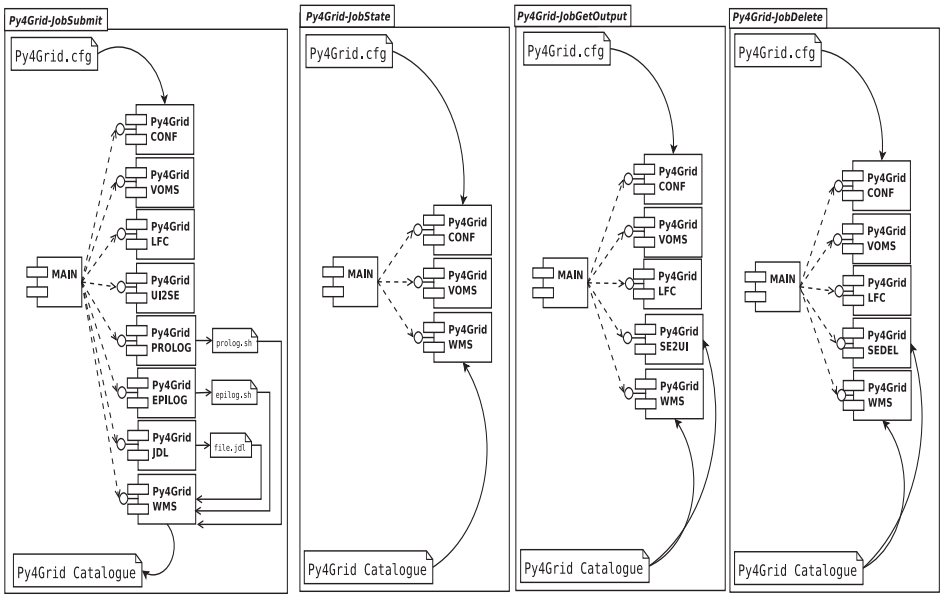


Fig. 1. `Py4Grid` components and module architecture. Dashed arrows: modules invocations. Full arrows: flow of input/output files.

3.1 User customization

Users may customize their job submissions, job requirements and file data transfers via a configuration file (`Py4Grid.cfg`). The configuration file consists of a set of mandatory and optional key-value pairs distributed through 4 sections (**vo**, **job**, **requirements** and **data**).

The **'vo'** section defines the user virtual organization (VO) name and the lifetime for the user proxy (case it is necessary to create a new one). The **'job'** section is where the user defines the execution envelope (the binaries or scripts to run, its arguments and inputs), standard error and standard output files and (optional) small file transfers via inputsandbox and outputsandbox. The executables do not have to be included in the inputsandbox since they are automatically considered. Similarly, the standard error and standard output files do not have to be included in the outputsandbox. This inputsanbox and outputsandox transfer mechanism relies on the WMS gridftp service, and on the scratch storage space available in the WMS. Traditionally, the WMS accepts input transfers no larger that 10 MB, and output transfers of 100 MB, and therefore, it does not scale for large big data transfers, either in file size or file quantity. An alternative method to copy big data files can be defined in the **'data'** section of the configuration file.

```
[vo]
# mandatory: virtual organization
voname = <VO>
# mandatory: proxy lifetime in hours (max = 168)
lifetime = 72

[job]
# mandatory: The executable ( a bash script, a C program, a Python program, ...)
executable = <EXE>
# optional: List of arguments (comma separated list)
arguments = <ARG1>,<ARG2>,<ARG3>
# mandatory: Standard output file
stdout = <STDOUT>
# mandatory: Standard error file
stderr = <STDERR>
# optional: List of files (comma separated list) transferred via inputsandbox
inputsandbox = <INPUTFILE1>,<INPUTFILE2>,<INPUTFILE3>
# optional: List of files (comma separated list) transferred via outputsandbox
outputsandbox = <OUTPUTFILE1>,<OUTPUTFILE2>,<OUTPUTFILE3>
```

The **'requirements'** section allows the user to define his expectations in terms of necessary cpu time or resident memory for a successful execution. Moreover, the user has the capability to blacklist specific sites (because the failure rate is to high) or redirect his jobs to specific computing services. This functionality provides a fine-grained control regarding the destination of user jobs, saving him time and effort. The underlying assumption is that the user already experienced some successful or unsuccessful job submissions, and can determine which values to apply on those configuration fields.

```
[requirements]
# optional: Estimation of execution time (minutes)
cputime = 120
# optional: Estimation of memory consumption (MB)
rammem = 3000
# optional: List of clusters to exclude from job submissions (comma separated list)
blacklist = <CEFQDN1>,<CEFQDN2>,<CEFQDN3>
```

```
# optional: Exclusive list of clusters for job submissions (comma separated list)
exclusive = <CEFQDN1>,<CEFQDN2>,<CEFQDN3>
```

Finally, the **'data'** section triggers the transfer of big data files using grid technologies (gridftp and SRM). For a complete description of data workflow, please refer to section 3.4.

```
[data]
# optional: List of big input files (comma separated list ) to be transferred
inputfiles = <BIGINPUTFILE1>,<BIGINPOUTFILE2>,<BIGINPOUTFILE3>
# optional: List of big output files (comma separated list ) to be retrieved
outputfiles = <BIGOUTPUTFILE1>,<BIGOUTPUTFILE2>,<BIGOUTPUTFILE3>
```

3.2 Job submission and monitoring

The job submission and job monitoring capabilities are provided by the `JobSubmit` and `JobStatus` components. Figure 2 details the workflow sequence diagram of each of those components.

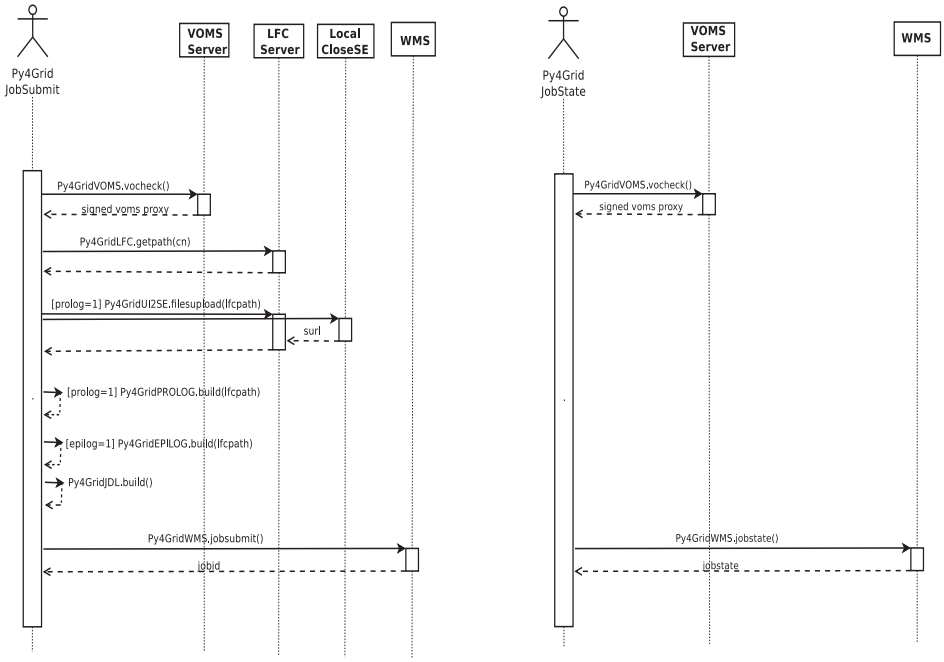


Fig. 2. `JobSubmit` (left) and `JobState` (right) sequence diagram.

The `Py4Grid-CONF` parses the `Py4Grid.cfg` user configuration file, and prepares the execution environment with the correct middleware contextualization. The `Py4Grid-VOMS` module checks if the user has a valid proxy. If it does not exist or if it about to expire, it requests the creation of a new one. The application

assumes that the user X509 certificate is properly installed. If this is not the case, it will exit with a standard middleware error. The lifetime and the virtual organization for the new generated proxy are taken from the user configuration file. Once the user presents a valid proxy, the component parses the user Distinguish Name (DN) and extracts the user's common name (CN). Based on that information, `Py4Grid-CONF` constructs a unique and immutable namespace in the LCG File Catalogue (LFC). This namespace will be latter used to keep track of Logical File Names (LFN) of transferred files.

If the user needs to transfer big input data files or software, he can declare that requirement under the **'data'** section of the `Py4Grid.cfg` configuration. This triggers the compression of the data as zipped tarballs, their transfer to the site Close Storage Element (CloseSE), and their registration in the user unique LFC namespace (`Py4Grid-UI2SE`). A prolog script is automatically built, shipped with the user job and executed in the remote site before the user payload. It will transfer input files from the Storage Element (where data was initially stored) to the local resource where the user payload will execute. At submission time, an epilog script is also build and shipped with the user job. Its goal is to execute in the remote site after the user payload. It will compress and transfer the output data files to a remote Storage Element, and register those transfers in the user's LFC namespace.

The full job definition is assembled by the `Py4Grid-JDL` module, which builds the Job Description Language (JDL) script, and submits it to the Workload Management System (WMS) service using the `Py4Grid-WMS` module. The information regarding job unique identifiers and input / output LFNs are registered in a hidden text catalogue stored under the user home directory. This catalogue is updated for each new job submission or job deletion.

The following block presents a snapshot of information displayed to the user during a grid job submission. The implemented workflow (proxy verification, LFC namespace verification, transfer of big input files, creation of epilog and prolog scripts, JDL building and job submission) is clearly reported back to the user, and transparently executed on user's behalf.

```
$ Py4Grid-JobSubmit.py
################################################################################################
 Welcome to Py4Grid.py
    Running Py4Grid-JobSubmit.py
    Executed on: 2014-06-09 19:38:46
################################################################################################
Proxy not found: /home/hpc/goncalo/.py4gridproxy/x509up_u605 (No such file or directory)
[Py4GridVOMS]   WARN: Your VOMS proxy is valid for less than 12 hours
[Py4GridVOMS]   WARN: Regenerating VOMS proxy
Enter grid pass phrase for this identity:
[Py4GridVOMS]   INFO: Your VOMS proxy has been regenerated for 72:00 hours
[Py4GridLFC]    WARN: /grid/<VO>/CN=<Common Name> exists in lfc01.ncg.ingrid.pt
[Py4GridUI2SE]  INFO: Successful transfer: py4grid-biginputdata-2014-06-09-193846.tgz
[Py4GridPROLOG] INFO: Prolog script successfully built
[Py4GridEPILOG] INFO: Epilog script successfully built
[Py4GridJDL]    INFO: JDL successfully built
[Py4GridWMS]    INFO: grid submission state: success
[Py4GridWMS]    INFO: grid Job ID: https://wms01.ncg.ingrid.pt:9000/dSJR_bTLx20ZKz1kq910ew
```

Once the job has been submitted, the user can monitor its status using the `JobStatus` component. This component knows which jobs have been submitted by the user parsing the text catalogue file created at submission time. With that

information, it queries the WMS service, parses the response, and retrieves the job status information. Information for all tracked jobs is ordered by submission time. Since the unique WMS job identifiers are complex and hard to memorize, the system tags each submission with a numeral identifier. This numeral can be latter used by other `Py4Grid` components to act on a given job.

```
$ Py4Grid-JobState.py
################################################################################################
 Welcome to Py4Grid.py
    Running Py4Grid-JobStatus.py
    Executed on: 2014-06-09 19:39:11
################################################################################################
[Py4GridVOMS] INFO: Your VOMS proxy is valid for more than 12 hours
[Py4GridWMS]  INFO: ------------------------------------------------------------
[Py4GridWMS]  INFO: JOB ID [1]:  https://wms01.ncg.ingrid.pt:9000/dSJR_bTLx20ZKz1kq910ew
[Py4GridWMS]  INFO: SUBMITED ON: Mon Jun  9 19:39:03 2014 WEST
[Py4GridWMS]  INFO: STATUS:      Done(Success)
[Py4GridWMS]  INFO: REASON:      Job Terminated Successfully
[Py4GridWMS]  INFO: DESTINATION: ngiescream.i3m.upv.es
```

3.3 Job retrieval and deletion

Figure 3 displays the `JobGetOutput` and `JobDelete` component workflow sequence for job output retrieval and job clearance.

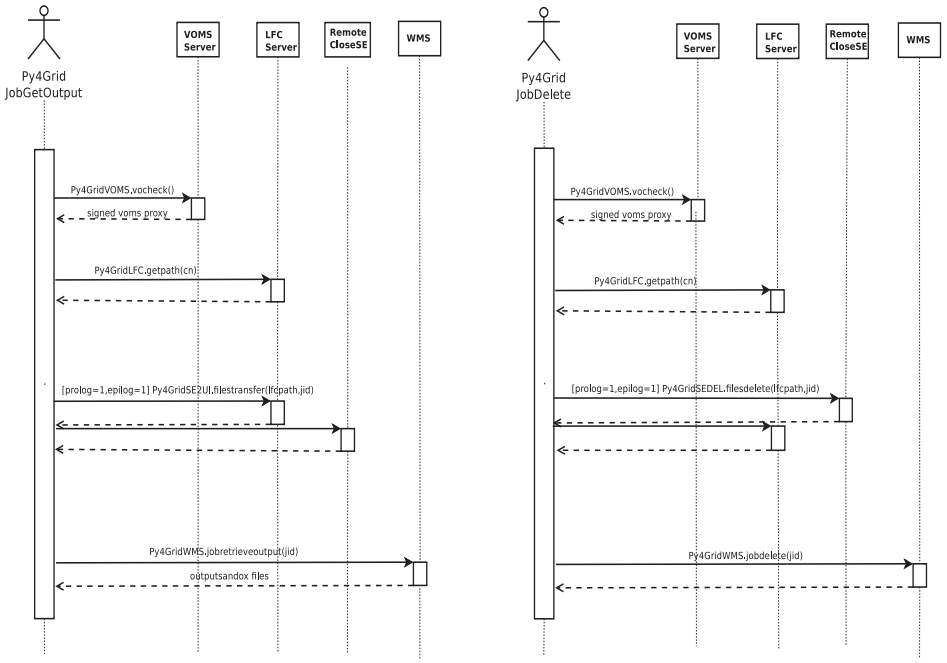


Fig. 3. `JobGetOutput` (left) and `JobDelete` (right) sequence diagram.

These components invoke methods to transfer big and small data files back to the users. After verifying the user's proxy and LFC namespace, the `Py4Grid-SE2UI` module is able to build the Logical File Name of output files based on the information stored in the hidden text catalogue file. It then contacts the LFC, and based on the LFN-SURL mapping, it determines the Storage Element where such data was stored, and transfers it back to the user under a single compressed tarball. Simultaneously, the small files (standard output, standard error, ...) are purged from the WMS, and retrieved back to the user.

```
$ Py4Grid-JobGetOutput.py 1
################################################################################################
 Welcome to Py4Grid.py
    Running Py4Grid-JobGetOutput.py
    Executed on: 2014-06-10 15:16:57
################################################################################################
[Py4GridVOMS]  INFO: Your VOMS proxy is valid for more than 12 hours
[Py4GridLFC]   WARN: /grid/<VO>/CN=<Common Name> exists in lfc01.ncg.ingrid.pt
[Py4GridSE2UI] INFO: Successful transfer: py4grid-bigoutputdata-2014-06-09-193846.tgz
[Py4GridWMS]   INFO: ------------------------------------------------------
[Py4GridWMS]   INFO: RETRIEVE JOB ID: https://wms01.ncg.ingrid.pt:9000/dSJR_bTLx20ZKz1kq910ew
[Py4GridWMS]   INFO: Output available in $HOME/SandboxDir/dSJR_bTLx20ZKz1kq910ew
```

The big input and output data files are kept indefinitely in the Storage Elements. Similarly, the information regarding big data transfers is also permanently kept in the LFC. That data is only cleared once a job deletion is triggered by the user. This execution deletes files from Storage Elements, clear LFC entries and removes a given job entry from the list of jobs in the system catalogue. Once a job is deleted, `Py4Grid` is no longer able to retrieve information with respect to that specific job.

```
$ Py4Grid-JobDelete.py 1
################################################################################################
 Welcome to Py4Grid.py
    Running Py4Grid-JobDelete.py
    Executed on: 2014-06-10 16:29:16
################################################################################################
[Py4GridVOMS]  INFO: Your VOMS proxy is valid for more than 12 hours
[Py4GridLFC]   WARN: /grid/<VO>/CN=<Common Name> exists in lfc01.ncg.ingrid.pt
[Py4GridSEDEL] INFO: Successful deletion: py4grid-biginputdata-2014-06-09-193846.tgz
[Py4GridSEDEL] INFO: Successful deletion: py4grid-bigoutputdata-2014-06-09-193846.tgz
[Py4GridWMS]   INFO: DEL JOB ID: https://wms01.ncg.ingrid.pt:9000/dSJR_bTLx20ZKz1kq910ew
[Py4GridWMS]   INFO: Removing JID entries from $HOME/.py4gridjids/Py4Grid.jid
```

3.4 Data flow

It is now worthwhile to detail the complete data transfer mechanism for the full job lifecycle, from job submission to its deletion in the system. Figure 4) presents the data flow. The full black arrows represent data pushes to the destiny while the dashed arrows depict data pulls from the destiny.

In a regular job submission, the User Interface pushes the inputsandbox data to the WMS (2) which is latter pulled by the WNs (3) together with the user payload. Once the user job executes and finishes, the outputsandbox data is pushed back to the WMS (6) and then pulled by the user (8).

When the user configures input and output transfers under the **'data'** section of the configuration file, the previous data flow cycle is changed with the introduction

of some intermediate steps. All files are compressed and compacted under a tarball, and pushed by the `Py4Grid-UI2SE` method to the UI CloseSE (1) prior to job submission. Steps 2) and 3) are then executed as regularly. Immediately before running the user payload, a prolog script (automatically built by `Py4Grid` and shipped with the user job) pulls the data from the Storage Element to the WNs (4). Once the job completes, and there are big output files to be retrieved by the user, an epilog script (also automatically built by `Py4Grid` and shipped with the user job), compresses, packs and pushes the data to the remote site CloseSE (5). While other outputsandbox files are retrieved back to the user through the standard transfer chain (steps 6 and 8), the `Py4Grid-SE2UI` method is responsible for pulling down the output data from the Storage Element where it was originally kept.

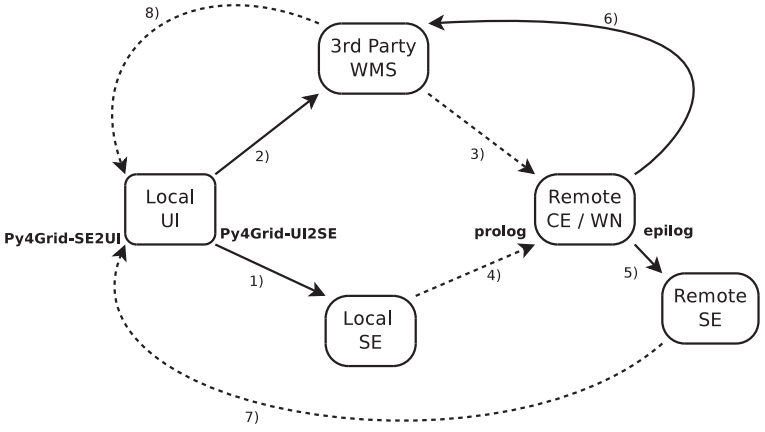


Fig. 4. Py4Grid data transfer workflow

4 Discussion

`PYTHON` was selected as the main development language due to its simplicity, to its support of object oriented programming and to its easy portability into multiple platforms. The full framework has been tested with `PYTHON 2.6` but no major issues are expected in higher versions.

The tool does not provide any GUI or Web interface since those front-ends were not considered a priority. The majority of scientific applications are developed under Unix-like environments, and today's scientists have, at some stage, used big computing clusters to scale their work. The main objective was to provide a similar interaction with the grid as the one researchers experience while submitting jobs to Linux batch system clusters. The four `Py4Grid` executables cover a large majority of the interactions scientists normally do with any computing infrastructure.

To keep the software portable, simple and flexible, `Py4Grid` does not use any database back-end. The information is kept in a user text catalogue updated at each job submission or deletion. This decision implies performance concerns specially in cases when users stress the infrastructure with bulk job submissions. Also, the probability of data loss is not minimal. Nevertheless, this first approach has showed to be sufficient for the long-tail of science use cases under evaluation.

As compared to UMD, `Py4Grid` command line tools have been considered simpler. `Py4Grid` has some minimal functionalities for error handling from an user perspective. If possible, it captures and interprets the error messages and displays them to the user with a friendly format. This avoids to expose the user to large amounts of information normally displayed when problems are observed, and to the cryptic nature of error messages, more suitable for the service administrators than service users. Moreover, the Py4Grid framework encapsulates job management and data management simultaneously, both invoked from single `Py4Grid` components/executables. Job submissions, job retrievals and data transfers are run in a transparent and seamless way, avoiding the direct interaction of the user with grid components such as the LFC, the remote storage and the information system. To achieve the same objectives with UMD command client tools, the user has to construct, validate and submit his JDL script, and take care of big data transfers as a separate process. Besides using multiple and different UMD commands, this is normally an error prone process, difficult to embrace by non-expert researchers. The drawback of this approach is that it decreases the system flexibility once users want to explore scenarios different from the envisioned use cases.

Middleware is always under evolution. This introduces a problem, and simultaneously, a challenge. Past experience showed that middleware clients may be renamed or refactored, with core functionalities changing over time. For example, presently, **'lcg_utils'** software suite, used to create grid equivalents of UNIX copy commands and file registration spanning storage systems and catalogues of choice, is being deprecated and substituted by **'gfal2-util'**. `Py4Grid` application is somehow agnostic to that evolution in the sense that the appropriate middleware clients can be easily redefined in the `PyGrid-CONF` module. A more worrying fact is that HEP is moving away from the LCG File Calogue (LFC). HEP experiments are administrating their dedicated namespaces and storage destinations inside their job and data management frameworks, avoiding to rely on external single-point of failure components that they do not control. In the `Py4Grid` framework, LFC plays a critical role in the present schema since it is the middleware component that introduces a unified namespace, and maps the destiny storage of transferred files. The fact that the community driving the initial LFC development is stepping away raises numerous questions regarding the future of this component. `Py4Grid` may mitigate this issue through the implementation of its own dedicated data namespace in the `Py4Grid` user catalog.

To conclude, `Py4Grid` intends to establish a first user friendly interaction with the grid, so that long-tail science researchers are kept motivated to explore the grid potential. Once researchers understand grid concepts, and are willing to try more complex job submission chains, it is no longer the best tool. For example, it is not prepared for parallel, dag or parametric requests. Moreover, it is also not

properly optimized for massive job submissions, or submissions of arrays of jobs. The software suite should be parallelized to deal with a high number of requests, and the catalogue schema has tobe revised to deal with concurrent read/write accesses. The output formats are also not adjusted to those use cases, resulting in very long outputs. Finally, `Py4Grid` does not implements any protocol of mechanism to improve the success rate of executions on the remote sites. Frameworks like GANGA or DIANE should be tried in alternative once researchers are more comfortable with the grid.

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. Py4Grid in EGI AppDB, https://appdb.egi.eu/store/software/py4grid
2. Py4Grid in GitHub, https://github.com/GoncaloBorges/Py4Grid
3. T. Maeno et al., "PanDA: distributed production and distributed analysis system for ATLAS", J. Phys.: Conf. Ser. 119 (2008) 062036
4. D. Spiga et al., "The CMS Remote Analysis Builder (CRAB)" Lecture Notes in Computer Science Volume 4873, 2007, pp 580-586.
5. A. Tsaregorodtsev et al, "DIRAC: A Community grid Solution", J. Phys.: Conf. Ser. 119 (2008) 062048
6. Micha Owsiak, Bartek Palak, and Marcin Plćiennik, "Graphical Framework for grid Interactive and Parallel Applications", Computing and Informatics, 27(2), 2012. 48.
7. Miroslaw Kupczyk, Rafal Lichwala, Norbert Meyer, Bartosz Palak, Marcin Plóciennik, and Pawel Wolniewicz, "Applications on demand as the exploitation of the Migrating Desktop", Future Gener. Comput. Syst., 21(1) 3744, 48.
8. Bartek Palak, Pawel Wolniewicz, Marcin Plóciennik, Michal Owsiak, and Tomasz Zok, "User-Friendly Frameworks for Accessing Computational Resources", In PL-Grid, pages 191204. 2012. 48
9. Jason Novotny, Michael Russell, and Oliver Wehrens, "GridSphere: a portal framework for building collaborations", Concurrency and Computation: Practice and Experience, pages 503513, 2004 (19).
10. Mary Thomas, Stephen Mock, Maytal Dahan, Kurt Mueller, Don Sutton, and John R. Boisseau, "The grid-Port Toolkit: A System for Building grid Portals", In HPDC, pages 216227. IEEE Computer Society, 2001 (19).
11. R. Barbera, A. Falzone, V. Ardizzone, and D. Scardaci, "The GENIUS grid Portal: Its Architecture, Improvements of Features, and New Implementations about Authentication and Authorization", In Enabling Technologies: Infrastructure for Collaborative Enterprises, 2007. WETICE 2007. 16th IEEE International Workshops on, pages 279283, 2007 (19).
12. J.T. Mościcki et al., "Ganga: A tool for computational-task management and easy access to grid resources", Computer Physics Communications, Volume 180, Issue 11, (2009) 2303-2316
13. DIANE: Distributed Analysis Environment, http://it-proj-diane.web.cern.ch/it-proj-diane/

Design and implementation of a Generic and Multi-Platform Workflow System

Abel Carrión[1], Nelson Kotowski[2,**], Miguel Caballer[1], Ignacio Blanquer[1], Rodrigo Jardim[2], and Alberto MR Dávila[2]

[1]Instituto de Instrumentación para Imagen Molecular (I3M). Centro mixto CSIC - Universitat Politècnica de València - CIEMAT Camino de Vera s/n, 46022 Valencia, Spain
{abcarcol,micafer1,iblanquer}@upv.es
[2]Laboratório de Biologia Computacional e Sistemas, Instituto Oswaldo Cruz (IOC), Fundação Oswaldo Cruz (FIOCRUZ), Avenida Brasil 4365, 21040-360, Rio de Janeiro, Rio de Janeiro, Brazil
{nelsonpeixoto,rodrigo_jardim,davila}@fiocruz.br

Abstract. Nowadays e-Science experiments require computational and storage resources that most research centres cannot afford. Fortunately, researchers have at their disposal many distributed computing platforms where to run their experiments: clusters, supercomputers, grids and clouds. None of these platforms has showed to be the ideal choice and each of them has its own advantages and disadvantages, depending on various factors. However, the use of these powerful systems poses a challenge for scientists who don't have a computer science background. For that reason, this paper describes the design and implementation of an intuitive workflow system capable of executing any kind of application on a mix of computing platforms. Moreover, a bioinformatics use case called Ortosearch that will exploit the workflow system is detailed.

1 Introduction

In the business context, the term workflow can be defined as the orchestration of a set of activities in order to accomplish a larger and sophisticated goal. A specialization of this term can be found on e-Science, the Scientific Workflows (SWFs). SWFs are a formalization of the particular process that a scientist must carry on to obtain publishable results from raw data. SWFs can also be seen as a collection of tasks that are processed on computing resources in a well-defined order to accomplish a goal. Due to the high computational cost of e-Science experiments, the computing resources needed are no longer localized but, rather, distributed and hence, developers of scientific applications (workflows) have many options when choosing the platform where to run their applications. In the last decade, these options included: clusters, supercomputers and Grid. Grid computing focuses on secure and collaborative resource sharing across multiple, geographically distributed institutions. One of the main drawbacks that Grid users face is that while

** CAPES/PDSE scholarship BEX 2137/14-3

a Grid offers access to many heterogeneous resources, a user normally needs a very specific environment that is customized to support a particular legacy application. Obviously, resource providers cannot support the diversity of all the required environments. In the last years, Cloud Computing has emerged as another viable [1] platform for running scientific applications. In particular, the use of virtualization provides many useful benefits for scientific workflow applications, including: customization of the software stack by the user, performance isolation, check-pointing and migration, better reproducibility of scientific analyses, and enhanced support for legacy applications [2] [3]. However, in other to compete with existing HPC systems in terms of performance, Cloud providers must begin offering high-speed networks and parallel file systems [4].
Because each platform has its own advantages and disadvantages in terms of usability, performance and cost, this work exposes the design and implementation of a generic (from the point of view of the applications that can be used) and multiplatform workflow system. One of the main concerns is to ease the combined use of the before mentioned platforms for executing SWFs, even for scientists with little knowledge about computer science.
The remainder of this paper is structured as follows. Firstly, Section 2 offers a brief description of similar tools and remarks the differences with respect the work presented in this paper. Next, Section 3 details the design of the architecture of the tool and Section 4 its correspondent implementation. As use case, Section 5 shows a bioinformatics workflow called Ortosearch that will exploit this work. To wrap up, the most relevant conclusions are extracted and the imminent work to do is proposed.

2 State-of-the-art

The use of workflow systems for executing scientific experiments is a topic widely covered in the literature. In this section, the criteria adopted for classifying these solutions has been: firstly, its generality (i.e. if it addresses any kind of application or only applications of a concrete field) and secondly, the distributed computing infrastructures supported.
Thus, the first category to be considered includes the generic solutions. With the rise of the Grid Computing paradigm, a large number of tools were proposed and used by the Grid Community. ASKALON [5] simplified the execution of workflow applications on the Grid with and XML-based programming interface that hides Grid middleware details to the user and allows the high-level composition of workflow applications. GridAnt [6] is an extensible and platform-independent workflow system that allows orchestrating a set of activities and expressing dependencies between them in XML. GridFlow [7] provides an user portal and an agent-based mechanism for dynamic scheduling of Grid jobs. Karajan is a workflow-based graphical problem solving environment which is effective at composing jobs in a complex Grid environment. The Kepler project [8] provides a workflow system derived from the Ptolomey II project. In this project a workflow system is modelled as a composition of actors (that are independent). The extensibility of this

system is given by the actor-oriented architecture, adding new actors. The system has a set of actors for performing typical Grid operations (authentication, file copy, job execution, job monitoring, service discovery, etc.). Nevertheless, the user needs to know a lot of details about the Grid resources. Pegasus [9] is a popular framework for mapping scientific workflows onto distributed resources. It uses an abstract workflow definition (users does not need to know details about the beneath systems) and workflow restructuring that clusters tasks in order to improve the overall performance. Webflow and Triana [10] offer a visual programming model for dynamic orchestration of high performance applications from a group of predefined commodity software modules. Both are powerful visual programming tools and enablers for Grid technology, abstracting the scientist from computer science details. When the Cloud Computing appeared, some tools from the Grid period were updated for supporting this new paradigm while others were developed anew. A good example of a Cloud-updated tool is Taverna [11]. Taverna is the workflow management system of the myGrid project. It is based in a definition language named Simple Conceptual Unified Flow Language (SCUFL). Taverna provides data models, enactor task executions, and graphical user interfaces. In Taverna, all computational workflow steps are web services. The introduction of the Taverna Server allows workflows to be executed on remote computational infrastructures (such as clusters, Grids and clouds). On the other hand, a project named e-Science Central (e-SC) [12] has developed a cloud-based Science Platform that allows scientists to store, analyse and share data in the cloud. e-SC can be deployed on both private and public Clouds. The CARMEN e-science project has also designed a generic e-science platform that runs in the cloud (in particular, Amazon AWS) and enables data sharing, integration and analysis. Although these tools are multi-platform, the workflow must be executed entirely in one of them, without the possibility of combining resources from different types of infrastructures. Moreover, when using Cloud Computing the virtual machines are statically deployed (previously to the execution), instead on-demand or dynamically as the NIST Cloud Computing definition says.

The second category entails non-generic systems, in concrete solutions related with the scientific field of the use case presented in this paper: bioinformatics. Galaxy [13] is an open web-based platform for genomic research, that makes computation accessible, ensures that all analysis are reproducible and allows the transparency via the sharing of experimental results between users. A Galaxy instance can utilize compute clusters for running jobs, and can be easily interfaced with PBS (Portable Batch System) and SGE (Sun Grid Engine). Chipster [14] offers a wide collection of data analysis methods within the reach of bioscientists via an intuitive graphical user interface. The analysis options are complemented with interactive visualizations and the possibility of saving workflows, which can be shared with other users.

The main difference between all the previous solutions and the one proposed in this work is that it comprises the following features in a single tool:

1. **Platform-agnostic client**: The client has been developed using a platform-independent programming language.

2. **Easy-to-use**: Although still not developed, the final version will offer a rich and usable Graphic User Interface (GUI).
3. **Generic**: The workflow system admits any kind of application that can be defined using its Workflow Specification Language.
4. **Multi-platform**: *Each part* of the workflow can be executed using different computing back-ends (clusters, supercomputers, Grids and Clouds).
5. **Cloud Computing features**: Meets the requirements expressed by the NIST Cloud Computing definition (dynamic provisioning of resources among others).

3 Architecture

The purpose of this section is twofold: firstly, to describe the design principles that guided the definition of the architecture and secondly, to show an overview of the architecture of the tool.

3.1 Design principles

The architecture has been designed taking into account three principles:

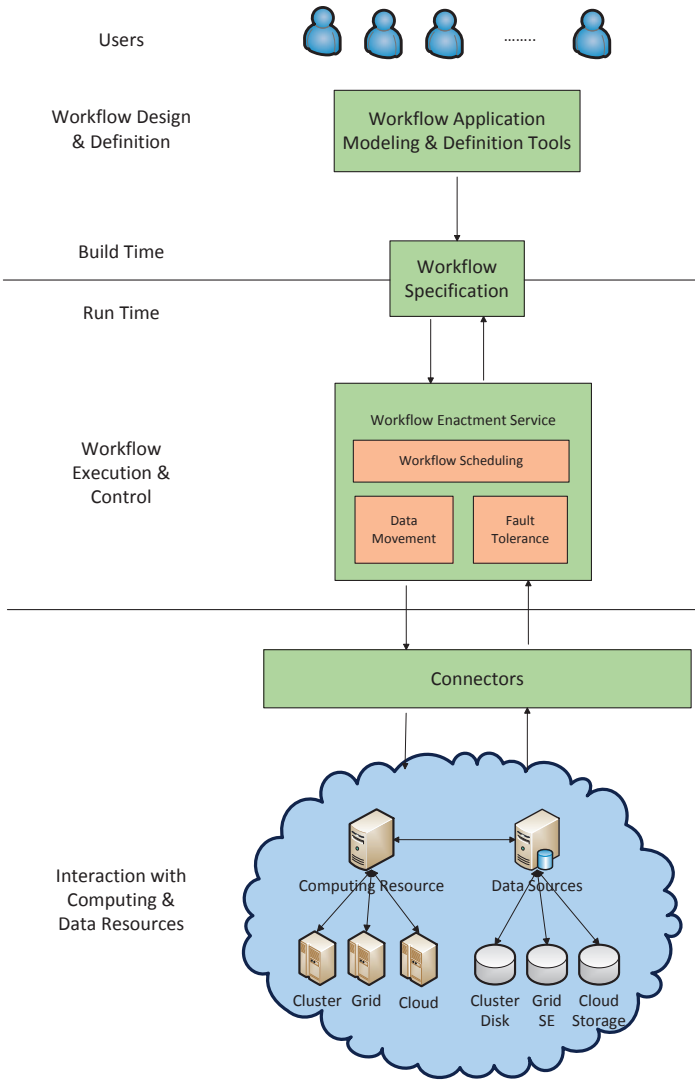
1. **Generality**: Possibility of executing a wide range of applications and supported by any kind of distributed computing infrastructure.
2. **Extensibility**: Enables a user to add a new application or a new computing back-end.
3. **Modularity**: Allows that new elements can be added without the need to change other parts of the system.

3.2 Overview

The architecture schema shown in Figure 1 is inspired by the one showed in [15], but updated accordingly to include the Cloud Computing paradigm. Basically, the architecture is divided into three sections: Workflow Design and Definition, Workflow Execution and Control and Interaction with the Resources. In turn, each of these three sections contains the components of the architecture. 'Workflow Design & Definition' is related with the Workflow Application Modelling & Definition Tools and the deployment time part of the Workflow Specification. 'Workflow Execution & Control' is the core of the architecture and includes the run time part of the Workflow Specification and the Workflow Enactment Service. Finally, the interaction with the computing and data resources is done through a set of connectors (plug-ins) for the different computing and storage back-ends.

4 Implementation

Once showed the architecture in the previous point, this section describes what technologies and/or software components have been used in the implementation of each element of the architecture. The system has been mostly implemented on Java because it is a platform-independent language and so, it can reach a wider number of users.


Users
Workflow Design & Definition
Workflow Application Modeling & Definition Tools
Build Time
Workflow Specification
Run Time
Workflow Execution & Control
Workflow Enactment Service
Workflow Scheduling
Data Movement
Fault Tolerance
Connectors
Interaction with Computing & Data Resources
Computing Resource
Data Sources
Cluster
Grid
Cloud
Cluster Disk
Grid SE
Cloud Storage

4.1 GUI

As many other workflow tools, in a future version the system will have a graph-base editing environment. Users compose applications by dragging programming components, called units or tools, from toolboxes, and dropping them onto a scratch pad, or workspace. Connectivity between the units is achieved by drawing cables subject to type-checking.
A workflow concrete specification is then generated by this visual tool and passed to the workflow planner. In order to improve the usability of the tool, this process is transparent to the user.

4.2 Workflow specification language

The workflow specification language is designed to bridge the gap between the GUI and the workflow execution engine. The format chosen for describing the workflows has been JSON (JavaScript Object Notation) because it has some benefits over XML, such as: shorter, easier to write and read and does not use end tags. The basic structure of a workflow specification document comprises three main sections (or objects in the JSON context): 'hosts', 'environments' and 'stages'. 'Hosts' is the part of the document that contains all the information about the infrastructure front-ends to be used. 'Environments' is a field for selecting the characteristics of the nodes and the packages that they must contain. Allows selecting nodes on heterogeneous environments and defining the Virtual Appliances (VAs) in a Cloud Computing scenario. Last but not least, Stages is an array of JSON objects that describes the different stages of the workflow. In turn, a stage object must: reference a host from the list of 'Hosts', include the features of the VM in the case of the Cloud (number of nodes, memory size, number of disks and capacity of each of them), list the command-lines that the stage executes, indicate fault-tolerance parameters and describe the stage-ins and stage-outs of the stage.
Currently, because the GUI component is under development, the entry point to the system is the JSON document. Once defined, the JSON document is passed to Jackson (a popular JSON parser) for parsing and validation and if it all is correct the workflow information is dumped to the program memory as Java objects.

4.3 Planner

The process of mapping from the concrete to the executable workflow can be automated by the planner. During that mapping the original workflow undergoes a series of refinements geared towards transforming the workflow to an executable description and towards optimizing the performance of the overall application. Figure 2 shows the three conversions that the planner performs. The first refinement (a) fuses two o more stages when the following two conditions are given: firstly, all of them are executed on the same infrastructure with the same environment and secondly, only the first stage could have input dependencies with other stages. This conversion is not mandatory and its purpose is to improve the performance.

The second transformation (b) is focused to channelize all the stage-in processes of a stage into a synthetic stage that is added before every stage of the workflow. Finally, if a stage is executed on the Cloud it is necessary a new transformation (c) that adds an stage for deploying the infrastructure before the stage created by (b) and another stage for the undeployment of the infrastructure after the stage-in of the subsequent stages.

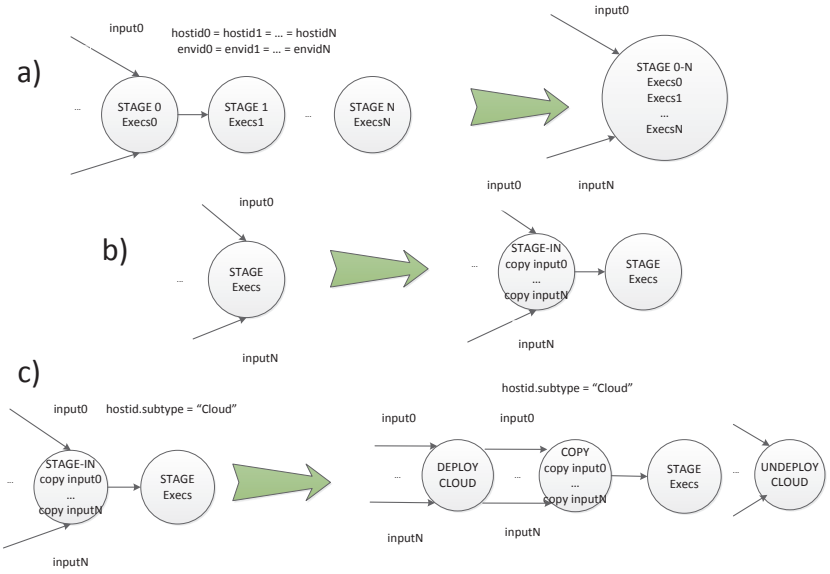


Fig. 2. Planner conversions

4.4 Runtime

The workflow execution engine is the core of the system and it is basically a runtime program in charge of monitoring the execution of the stages of the workflow and when it is necessary performing the proper actions. At the start of the execution, every stage has by default set its status to 'DISABLED'. Because the runtime is data-flow oriented, only when all the inputs of a stage are available, the status of the stage changes to 'ENABLED' and is ready to be executed. Then, the runtime periodically queries the status of the stage until it finishes. If it has finished successfully, the stage-outs (which in turn are stage-ins of subsequent stages) are enabled. The process is repeated until all the stages have been executed correctly and the final output is served to the user.

4.5 Connectors

The connectors are plug-ins that allow using different computational back-ends. In the Java programming language it has been implemented as an abstract class with two abstract methods, 'run_stage' and 'get_status'. Thus, when the runtime launches a stage or queries the status of an execution, it invokes these methods using the right connector. Currently, there are two connectors implemented: a PBS connector and a Cloud connector.

PBS Connector The PBS Connector is intended for executing stages on systems with a PBS (Portable Batch System) job scheduler. The task of PBS is to allocate computational tasks, i.e., batch jobs, among the available computing resources.

IM(Cloud) Connector The Cloud Connector uses a virtual infrastructure manager developed in the GRyCAP (Grid and High-Performance Computing Group). This manager called Infrastructure Manager [16] (on advance IM) has the following components: two APIs (XML-RPC and REST) for calling the services provided by it and cloud connectors for deploying, monitoring,etc. the Virtual Machines in different types of Cloud providers (OpenStack [17],Amazon EC2 [18],OpenNebula,etc). The IM selects the rightmost VMI (Virtual Machine Image) from a Virtual Machine Repository Catalog (VMRC) [19] and contextualizes it with another tool called Ansible [20].
One important detail when dealing with huge amounts of data to be processed in the Cloud Computing is that it is more efficient to move the computation to the data rather than the other way around. This requires having computational resources closely coupled to the servers holding the data. Cloud computing offers the chance to do this if the cloud is internally engineered with fast networking between the storage and compute servers.

4.6 Persistence

Due to the high-computational cost of e-Science experiments it is mandatory to allow users to interrupt the client execution and to resume it later. The persistence in the workflow system has been implemented using a No-SQL or document oriented database, called MongoDB. This database is suitable to be used by our tool because it stores the information as JSON-style documents, allowing the straightforward translation between workflow objects and database documents.

5 Use case: Orthosearch

This section briefly describes the use case that will exploit the system presented above. Among several topics in comparative genomics, homology inference aids on providing a better insight on species evolutionary history [21]. Orthology, one of the several scenarios related to homology, refers to genes or proteins that share a common ancestral, usually inter-species; such genes or proteins usually perform

equivalent functions in both species, although this is not mandatory.
OrthoSearch [22] is a scientific workflow, implemented as a comparative genomics pipeline designed for homology inference among Protozoa organisms. It uses a reciprocal best hits, HMM-profile based approach, having as input data both an ortholog database and an organism proteome.
Such data is processed in several steps with the aid of bioinformatics tools. Briefly, all ortholog groups are aligned with Mafft [23], generating multiple sequence alignments (MSAs); later, HMMER3 [24] [25] processes such output data, builds individual HMM profiles (*hmmbuild*) and a concatenated single file containing all the profiles (*hmmpress*) is generated. These concatenated profiles, will be confronted against the organism proteome (via *hmmsearch* and *hmmscan*).
At last, HMMER3s *hmmsearch* and *hmmscan* output data are processed in a bidirectional approach by a reciprocal best hits algorithm, which infers orthologous groups for this scenario (orthologous database versus organism proteome). As more organisms proteomes are submitted to OrthoSearch against the same orthologous database, common orthologous groups might be identified and analyzed in order to provide evolutionary data insight on such organisms.
Mafft, as well as HMMER3s *hmmbuild*, receives multiple files as input data and generates multiple output files; HMMER3s hmmsearch confronts each of the HMM profiles, one a at time, against the organism proteome, generating one file for each HMM profile input data; and HHMER3s hmmscan confronts a single, binary compressed data collection of profiles (generated by *hmmpress*) against the organism proteome, generating a single output file (see Figure 3).

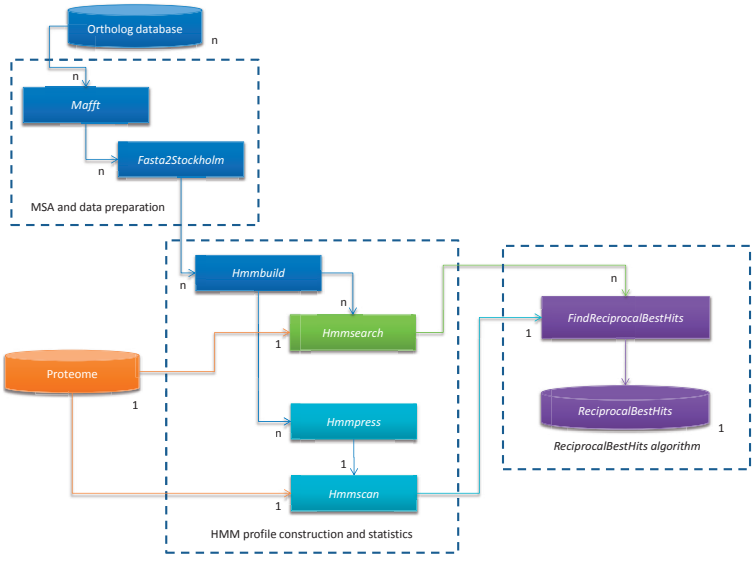


Fig. 3. Orthosearch workflow

Both Mafft and HMMER3 are applications which demand extensive computational power and that account for considerate amount of execution time. OrthoSearch effectiveness has been demonstrated while confronting five Protozoa species against COG/KOG databases [26] in order to infer common orthologous groups [27].
At it was mentioned before, at this moment, the entry point to the workflow system is the JSON document, and the only effort needed to analyse the Orthosearch use case in the platform is to define a simple JSON document. In the future, when the GUI is available, this task will be more automatic.

6 Conclusions and Future Work

The system presented in this paper is a work in progress. Although an initial prototype is available for evaluation, it will undergo significant extensions based on community feedback. The current implementation supports PBS-based systems and Cloud platforms; but efforts are being made to support other environments such as Grid. Workflow management also brings new challenges on issues like security, as it requires more flexible cooperation among different platforms. Another important aspect is that papers often draw conclusions from data that is not available to others to examine and analyse themselves, and so reproducibility is a key aspect on e-Science. These will be addressed when the tool becomes mature.
Over the long term, we believe that a system such as the one presented here can have an impact on use cases like Orthosearch by making various distributed computing infrastructures accessible to the entire science and community.

Acknowledgements

The authors would like to thank the Spanish "Ministerio de Economía y Competitividad" for the project TIN2013-44390-R. This work has been developed under the support of the programme "Formación de Personal Investigador (FPI)" (year 2012) from the "Universidad Politécnica de Valencia", granted to Abel Carrión Collado.

References

1. C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good. On the Use of Cloud Computing for Scientific Workflows. In *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, pages 640–645, Washington, DC, USA, December 2008. IEEE.
2. Renato J. Figueiredo, Peter A. Dinda, and José A. B. Fortes. A Case For Grid Computing On Virtual Machines. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, Washington, DC, USA, 2003. IEEE Computer Society.
3. Wei Huang, Jiuxing Liu, Bulent Abali, and Dhabaleswar K. Panda. A case for high performance computing with virtual machines. In *Proceedings of the 20th annual international conference on Supercomputing*, ICS '06, pages 125–134, New York, NY, USA, 2006. ACM.

4. Gideon Juve, Ewa Deelman, Karan Vahi, Gaurang Mehta, Bruce Berriman, Benjamin P. Berman, and Phil Maechling. Scientific workflow applications on Amazon EC2. In *2009 5th IEEE International Conference On E-Science Workshops*, pages 59–66. IEEE, December 2009.
5. T. Fahringer, R. Prodan, Rubing Duan, F. Nerieri, S. Podlipnig, Jun Qin, M. Siddiqui, Hong L. Truong, A. Villazon, and M. Wieczorek. ASKALON: A Grid Application Development and Computing Environment. In *GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, GRID '05, pages 122–131, Washington, DC, USA, 2005. IEEE Computer Society.
6. *GridAnt: a client-controllable grid workflow system*, 2004.
7. *GridFlow: workflow management for grid computing*, 2003.
8. Scientific workflow management and the Kepler system. *Concurrency Computat.: Pract. Exper.*, 18:1039–1065, 2006.
9. Ewa Deelman, Gurmeet Singh, Mei H. Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G. Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, and Daniel S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, July 2005.
10. Ian Taylor, Matthew Shields, Ian Wang, and Andrew Harrison. Visual Grid Workflow in Triana. *Journal of Grid Computing*, 3(3):153–169, September 2005.
11. Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R. Pocock, Anil Wipat, and Peter Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20:3045–3054, 2004.
12. S. Woodman H. Hiden, P. Watson and D. Leahy. e-Science Central: Cloud-based e-Science and its application to chemical property modelling. 2011.
13. Jeremy Goecks, Anton Nekrutenko, James Taylor, and The Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11:R86+, 2010.
14. M. Aleksi Kallio, Jarno Tuimala, Taavi Hupponen, Petri Klemela, Massimiliano Gentile, Ilari Scheinin, Mikko Koski, Janne Kaki, and Eija Korpelainen. Chipster: user-friendly analysis software for microarray and other high-throughput data. *BMC Genomics*, 12:507+, 2011.
15. Jia Yu and Rajkumar Buyya. A Taxonomy of Workflow Management Systems for Grid Computing, April 2005.
16. Miguel Caballer, Ignacio Blanquer, Germán Moltó, and Carlos de Alfonso. Dynamic management of virtual infrastructures. *Journal of Grid Computing*, 2014.
17. Antonio Corradi, Mario Fanelli, and Luca Foschini. VM consolidation: A real case based on OpenStack Cloud. *Future Generation Computer Systems*, June 2012.
18. F. Miller, A. Vandome, and J. McBrewster. Amazon Web Services. 2010.
19. Jose V. Carrión, Germán Moltó, Carlos De Alfonso, Miguel Caballer, and Vicente Hernández. A Generic Catalog and Repository Service for Virtual Machine Images. In *2nd International ICST Conference on Cloud Computing (CloudComp 2010)*, 2010.
20. ANSIBLE. `http://www.ansible.com/`, 2014. [Online; accessed 14-July-2014].
21. Eugene V. Koonin. Orthologs, paralogs, and evolutionary genomics1. *Annual Review of Genetics*, 39(1):309–338, 2005.
22. Sergio M. Cruz, Vanessa Batista, Edno Silva, Frederico Tosta, Clarissa Vilela, Rafael Cuadrat, Diogo Tschoeke, Alberto M. R. Davila, Maria L. Campos, and Marta Mattoso. Detecting distant homologies on protozoans metabolic pathways using scientific workflows. *International Journal of Data Mining and Bioinformatics*, 4(3):256+, 2010.

23. Kazutaka Katoh, Kei-ichi Kuma, Hiroyuki Toh, and Takashi Miyata. MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Research*, 33(2):511–518, January 2005.
24. Robert D. Finn, Jody Clements, and Sean R. Eddy. HMMER web server: interactive sequence similarity searching. *Nucleic Acids Research*, 39(suppl 2):W29–W37, July 2011.
25. Sean R. Eddy. A new generation of homology search tools based on probabilistic inference. *Genome informatics. International Conference on Genome Informatics*, 23(1):205–211, October 2009.
26. Roman L. Tatusov, Natalie D. Fedorova, John D. Jackson, Aviva R. Jacobs, Boris Kiryutin, Eugene V. Koonin, Dmitri M. Krylov, Raja Mazumder, Sergei L. Mekhedov, Anastasia N. Nikolskaya, B. Sridhar Rao, Sergei Smirnov, Alexander V. Sverdlov, Sona Vasudevan, Yuri I. Wolf, Jodie J. Yin, and Darren A. Natale. The COG database: an updated version includes eukaryotes. *BMC bioinformatics*, 4(1):41+, September 2003.
27. R. Cuadrat, S. Cruz, D. Tschoeke, E. Silva, F. Tosta, H. Juca, M. Campos, M. Mattoso, and A. Davila. An orthology-based analysis of pathogenic protozoa impacting global health: an improved comparative genomics approach with prokaryotes and model eukaryote orthologsorthologs, paralogs, and evolutionary genomics. *OMICS: A Journal of Integrative Biology*, 20143001in press.

Session III:
Clouds for e-Science

Virtual Desktop Infrastructure (VDI) Technology: FI4VDI project

Redondo Gil, C.[1,2] carlos.redondo@fcsc.es, Ruiz-Falcó Rojas, A.[1,2] antonio.ruizfalco@fcsc.es, Alonso Martínez, R.[1,2] ruth.alonso@fcsc.es, Fanego Lobo. A.[1,2] alvaro.fanego@fcsc.es, Martínez García, J.M.[1,2] jose.martinez@fcsc.es, and Lorenzana Campillo, J.[1,2] jesus.lorenzana@fcsc.es

[1] Castille and Leon Technological Center for Supercomputing (FCSCL)
[2] FI4VDI Project http://fi4vdi-sudoe.eu

Abstract. This paper presents an analysis of the FI4VDI project, whose innovative model is the development of service delivery using cloud computing in order to create an infrastructure suite aims at large companies and SMEs alike, educational institutions, universities and/or research centres that would contribute to competitiveness and provide an efficient solution to reducing the carbon footprint derived from the use of information technology.

The penetration of Cloud Computing in sectors as public administration and industry is resulting slow, consequently it is missing the competitive advantages that this technology can provide for both ICT-provider companies and end users thereof. For encouraging the use of cloud computing, from the R+D Centers and services involved in this project, the experience and knowledge of this technology could be transferred and the use of virtual desktops to the clusters of ICT companies, thus including in this way a differentiating service in their portfolios products and also contributing to the development of the economy and the society.

The aim of the FI4VDI project is: to develop a federated infrastructure network for the creation of virtual desktop services. Firstly, by evaluating the position and perception of public and private organisations in the southwest Europe as regards the desirability of the virtualising IT operating environments, and secondly, by promoting the spread of cloud computing pretending achieve savings, efficiency, simplicity and productivity improvement. In pursuing the project resources are used to generate highly innovative services through a PAAS (Platform as a Service) and SaaS (Software as a Service) to enable the generation of virtualization services jobs for users of public and private sector inside southwest Europe. The development of this infrastructure is to ensure that users data protection and compliance with the rules relating to safety information and established SLAs. The launch of this service will result in an improvement through technological innovation and cost savings in targeted sectors thereof.

The provision of cloud computing services by supercomputing centres has a positive effect on the ecological footprint; dependence on physical ICT infrastructures is reduced when these are replaced by virtual ones, and this in turn produces a marked reduction in energy consumption in these institutions.

With a federated cloud computing model, desktops can be connected to dedicated servers with high rates of effective utilisation, greatly increasing energy efficiency and reducing the carbon footprint associated with the service.
The goal of the project is to develop a federated infrastructure network for the creation and management of energy efficient ICT services.
Organizations involved in the project FI4VDI
Organisations that participated actively as infrastructure providers included
- The Supercomputing Centre Castille and Leon
  Castille and Leon Region ES41 (Spain)
- The Computing and Advanced Technologies Centre Extremadura
  Extremadura Region ES43 (Spain)
- The University of Lerida Faculty of Arts Computer Centre Ponent
  Catalonia Region ES51 (Spain)
- The University of Montpellier 2 Sciences et Techniques
  Languedoc-Roussillon Region FR81 (France)

Organisations that participated actively as business associations included
- The Innovative Business Association for Network Security and Information Systems (Spain)
- Aveiro Association of Companies for an Innovation Network
  Central Region PT16 (Portugal)
- The Science and Technology Park Agri-Food Consortium Lerida
  Catalonia Region ES51 (Spain)

Keywords: VDI, Federated Infrastructure, Distributed Data Centre, Energy Efficiency, Carbon Footprint.

1 FI4VDI-SUDOE Project Objectives

The aim of the FI4VDI project is to develop a federated infrastructure network for the generation of virtual desktop services, and to promote sustainable development by leveraging the benefits deriving from transnational cooperation.
In brief, the project proposed the creation of a private cloud infrastructure using the resources available in various supercomputing centres located in different SUDOE regions, trying to ensure the protection of users' data and compliance with regulations pertaining to information security and the service-level agreements established. Implementation of this service would entail improved competitiveness and cost savings in the sectors targeted, where energy savings and efficiency are a distinguishing feature.
With a federated cloud computing model, desktops can be connected to dedicated servers with high rates of effective utilisation, greatly increasing energy efficiency and reducing the carbon footprint associated with the service. [1]
The problem addressed by the project was the need to determine the position and perception of public and private entities located in the SUDOE Space as regards the desirability of virtualisation of IT operating environments, and to promote the spread of cloud computing as a means to achieve savings, efficiency and simplicity

with the primary objective of ensuring improved productivity.
The territorial cooperation program of the European Space Southwest (SUDOE), Fig. 1., supports regional development through transnational co-financing projects through the FEDER (Fondo Europeo de Desarrollo Regional). Involved Spanish, French, Portuguese and British regions (Gibraltar) government agencies, can contribute to the growth and sustainable development of the area South-West Europe by developing transnational cooperation projects in different issues.

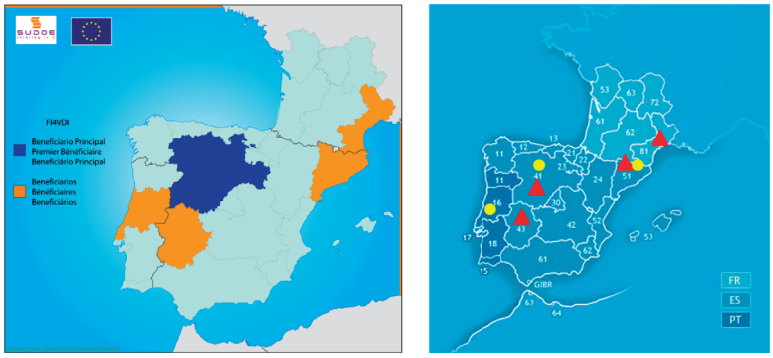


Fig. 1. Scope of the FI4VDI project in the SUDOE Space `www.sudoe.eu`. Setting of the FI4VDI project: profile of the partners participating in the project.

The origin of the project: The provision of cloud computing services by supercomputing centres has a positive effect on ecological footprints; dependence on physical ICT infrastructures is reduced when these are replaced by virtual ones, and this in turn produces a marked reduction in energy consumption in these institutions.
Project strategy and structure:

- Task group 0. Preparation
- Task group 1. Project coordination and management
- Task group 2. Technical infrastructure development
- Task group 3. Adapting applications to the cloud environment
- Task group 4. Integration. Prototypes
- Task group 5. Project monitoring and evaluation
- Task group 6. Publicity and information. Market capitalisation

2 Virtual Desktop Infrastructure-VDI as a technology proposal [2] [3]

The term desktop virtualization was introduced in the 1990s to describe the process of separation between the desktop, which encompasses the data and programmes that users employ for their work, and the physical machine. A ”virtual” desktop is

stored remotely on a central server rather than on the hard disk of an individual personal computer. This means that when users work on their desktop from their laptop or PC, all their programmes, applications, processes and data are stored and run centrally, allowing users to access their desktops from any device that can connect remotely to the central server, such as laptops, PCs, smartphones, or thin clients.

Through desktop virtualisation, the entire environment of an information system or the environment itself is encapsulated and delivered to a remote device. This device can be based on a completely different hardware architecture from the one used by the projected desktop environment. It can also be based on a completely different operating system.

Desktop virtualisation consists of the use of virtual machines to enable multiple network users to maintain their individual desktops on a single server or host computer. The central computer may be located in a residence, in the company or in a data centre. Users may be geographically dispersed and connected to the central computer via a local area network (LAN), wide area network (WAN) or via the Internet.

Desktop virtualisation offers advantages over the traditional model, in which each computer functions as a complete and independent unit with its own operating system, peripherals and applications. Energy costs are reduced because resources can be shared and allocated to users according to their needs, and the integrity of user information is enhanced because the data centre stores and safeguards all data and backups. Furthermore, software conflicts are minimised by reducing the total number of programmes stored on computers, and although the resources are distributed, all users can personalise and customise their desktops to meet their specific needs. Thus, desktop virtualisation provides greater flexibility than the client/server paradigm.

The limitations of desktop virtualisation include the possibility of security risks if the network is not well managed, the loss of user autonomy and data privacy, the challenges involved in creating and maintaining drivers for printers and other peripherals, difficulties in managing complex multimedia applications and problems in maintaining the addresses of virtual machine users consistent with those held by the data centre.

Benefits of VDI technology

Like any other technology, desktop virtualisation provides a number of key benefits that render this technology the first choice for a large number of users:

- Enhanced security and reduced desktop support costs [4]
- Reduced general hardware costs [5]
- Ensured business continuity [6]
- An eco-friendly alternative [7]
- Improved data security
- Instant implementation of new desktops and use of applications
- Virtually zero downtime in the case of hardware failure
- Significant reduction in the cost of new deployments
- The PC replacement cycle is extended from 2-3 years to 5-6 years or more

– Existing desktops include multiple monitors, bidirectional audio/video, video streaming, USB port supports, etc.
– Company employees can access their virtual desktops from any PC, including a PC in the employee's home
– Resources tailored to desktop needs
– Multiple desktops on demand
– Free provision of desktop computers (controlled by the policies of each corporation)

3 Description of the technical task groups of FI4VDI PROJECT

3.1 Infrastructure development

A **Technical design:**
A federated infrastructure is an aggregate of multiple resources, including virtualization resources, likely with heterogeneous architectures and geographically distributed members; the main goal is to make these characteristics transparent to the end-user. The end-user should be able to use and manage the whole resources using a single user/password set.
The design of the federation infrastructure includes a central node, known as "Federator Node" and any number of federated nodes, known as "Federated Zone" each. The "Federator Node" acts as Master of the federation infrastructure. This Master is located somewhere over any public or private cloud. [10]
The functions of the "Federator Node" are:
– Store and manage the root database for users and authentication
– Serve as single access point for users. Web interfaces and CLI are available
– Manage the deployment of virtual machines over the "Federated Zones"

There is not virtualization infrastructure (hypervisors or storage) in the "Federator Node".
"Federated Zones" are controlled by the "Federator Node". This one receives and processes the users requests for creating and deleting virtual machines. Orders are sent to single or multiple "Federated Zones". Every "Federated Zone" has a synchronous read-only copy of the users database. Hypervisors and storage are locally available in "Federated Zones". There is a local "Deployment Manager" receiving orders from "Federator Node" too.
End-Users perform deployment requests at "Federator Node". According to the permissions schema (ACL), virtual machines can be launched to single or multiple "Federated Zones". In order to avoid SPOFs (Single Point of Failure), the "Federated Node" is built over High Availability infrastructure, in a mix of physical and virtual servers.
Every launched Virtual Machine is independent from any other one. This is a purpose for multiple VMs in the same or different "Federated Zones". User credentials (User/Password) must be valid all over the federation infrastructure. And this will be valid even if new "Federated Zones" are added in the future.

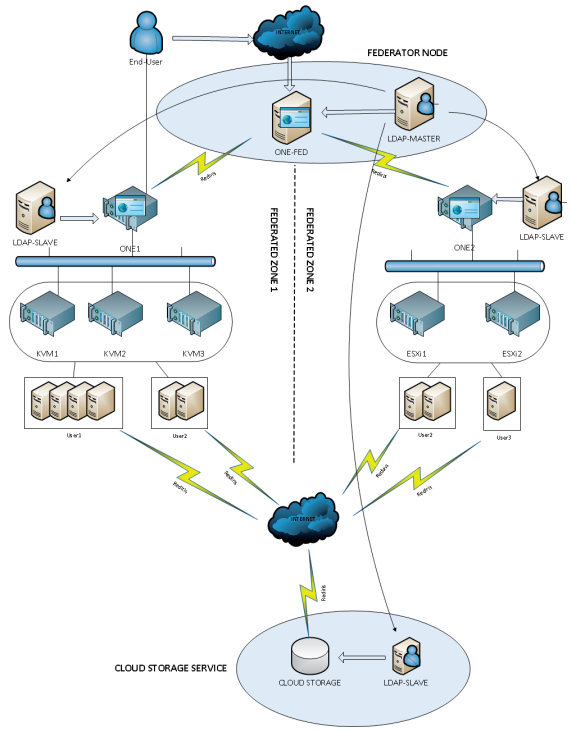


Fig. 2. Lower layer technical design. Main Items Source. FCSCL, 2014.

"Federated Zones" has XEN, KVM and VMWare hypervisors. Users can choose between then. A pool of heterogeneous hypervisors simplifies the job when clients want to migrate their virtual machines from other clouds to this one. In most cases their virtual machines will be compatible with any of the hypervisors without converting them.

The software that handles the whole federation infrastructure is "OpenNebula" (ONE) [8]. ONE is a deployment manager for Virtual Machines. It is licensed as Free Software (Apache License v2). Version 4.6 and following ones integrate a federated working mode. This mode fully matches the proposed technical scheme.

There is an ONE instance in Master Mode (ONE-FED) in "Federator Node". Users database is managed by an OpenLDAP instance (LDAP-MASTER). This instance is stored in "Federator Node" too. ONE provides multiple user interfaces, including web (known as Sunstone), CLI, OCCI (Open Cloud Computing Interface) and EC2 (Elastic Compute Cloud).

Any "Federated Zone" is locally governed by an ONE instance in slave mode (ONE1, ONE2, ONE3). This ONE performs any deployment operation ordered by the MASTER ONE. In order to reduce network traffic a read-only synchronous copy of users database (LDAP-SLAVE) is stored for local consul-

tations. Communications efficiency is one of the main goals of the infrastructure.
ONE-FED much reach any "ONE SLAVE" anytime in order to manage de-

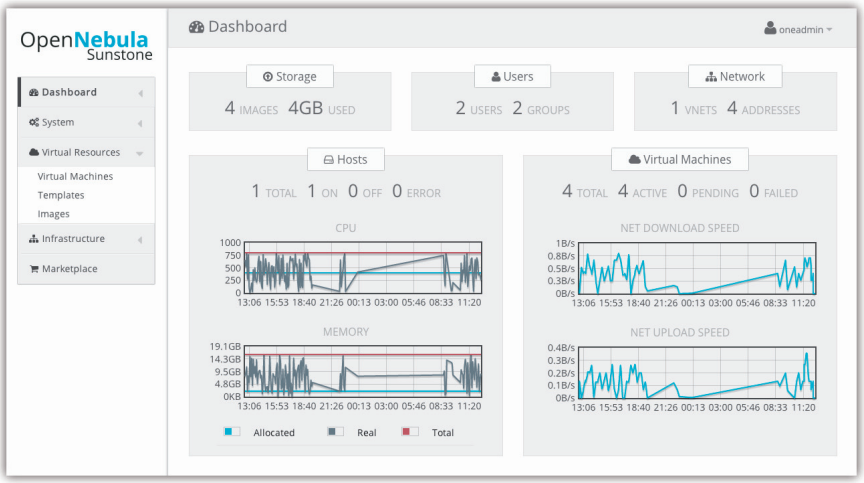


Fig. 3. Lower layer technical design. Main Items Source. FCSCL, 2014.

ployments and monitoring VMs. Because every "ONE SLAVE " is usually remotely located, with different security policies, the communications are supported by Lan-to-Lan IPSec VPNs. When possible, links are also supported over 802.1Q VLANs with R+D+i background networks (vg.- Spanish RedIRIS, French RENATER, European GEANT).
As value added service, a cloud storage stack is available for every end-user (CLOUD STORAGE). This one is built over the OwnCloud [9] software. OwnCloud is Free Software (AGPL3) and is located in "Federator Node", although it could be located anywhere over the Internet. The storage service is available from any VM in any "Federated Zone". It is available from any other place over Internet.
CLOUD STORAGE service uses the same LDAP users database in order to be completely integrated in the federation infrastructure.
One of the main purposes of this design is scalability. New "Federated Zones" can be fastly added in a few steps. The federation infrastructure is a "living entity" and can grow in order to accommodate to peak loads.
Nowadays, the federation infrastructure is composed of 3 "Federated Zones". The "Federated Node" and the "CLOUD STORAGE" are physically located in the same datacenter that one of the "Federated Zones", but logically isolated.
Each "Federated Zone" belongs and is provided by a different Project Partner.

B **Timeline Deployment**

3 phases have been defined in the project roadmap. To deploy the federated infrastructure many IT items (servers, storage systems, network devices, etc) are needed.

Phase 1 is known as restoration of value for IT systems. Project partners' datacenters were mostly equipped with HPC (High Performance Computing) IT systems. As most partners are 'Supercomputing Centers', HPC IT systems were the usual equipment. Nowadays, the logical evolution of 'Supercomputing Centers' has been from HPC to Cloud. Some HPC IT systems has been reused as Cloud IT systems. In these systems, many of the old computing servers are now hypervisors.

So, many of the necessary IT systems for the project were currently available. Specifically:

- COMPUTAEX Partner:
  - 4 HP Proliant BL460c-G6 servers
  - HP EVA8100 Fiber Channel NAS/SAN Storage
  - HP Storageworks EML 245e Tape Storage
  - RedIRIS 10Gb connectivity
- UdL Partner:
  - 1 HP Proliant DL380 G5 Server
  - RedIRIS 10Gb connectivity
- FCSCL Partner:
  - 3 HP Proliant DL580 servers
  - Dell Equallogic 6100 iSCSI SAN Storage
  - RedIRIS 1Gb connectivity (10Gb update in progress)
- UM2 Partner:
  - 2 IBM dx360 M3 servers
  - IBM DCS9900 Parallel Storage
  - RENATER 1Gb connectivity

Phase 2. Many of the necessary IT systems involved in the project has been obtained in Phase 1. But the pool of necessary IT systems is not complete. Because of this, many IT equipment has been acquired. During phase 2 we will invest in new infrastructure servers. Specifically:

- COMPUTAEX Partner:
  - 3PAR 7200 FC SAN Storage
  - 2 HP BL465 G8 Servers
- FCSCL Partner:
  - Dell Equallogic 6100 iSCSI SAN Storage

Phase 2 is finished with the technical work for setting the federation up and running.

The foundations for **Phase 3** falls over communications [11]. The federated infrastructure is extended with a new partner: CESGA (Centro de Supercomputacin de Galicia, www.cesga.es). The new partner is a Supercomputing Center too. Storage systems will be deployed at CESGA's datacenter and will be also connected with the network of the Federated Infrastructure.

The main goal, using storage and high capacity network, is to use the new

member for backup tasks and contingency plan. As a prototype, the obtained knowhow will allow to add new "Backup Zones" in a few single steps.
Backup Zones store copies of data and virtual machines running at Federated Zones. It is not a fully equipped Federated Zone as it hasn't a complete pool of hypervisors, but it can be used in this way if necessary.

Fig. 4. Physical locations included in the Federated Infrastructure. Red for "Federated Zones" and purple for "Backup Zones" Source: FCSCL. Google Maps, 2014.

3.2 Prototypes and adaptation processes to the Cloud paradigm of applications

After designing a global interface access and monitoring for all the participating entities and potential end users, we proceed to implement 4 prototypes and Cloud adaptation processes, with applications in different scenarios, both public and private sector:

1. VDI University and Academic Management: eGovernment - Procedures for university management replicated with the same hw and sw resources. Mass deployment of virtual desktops on demand, persistent or non-persistent in the environment of a classroom working. Electronic Administration of Public Universities of Castilla y Leon.
2. VDI Health management: management model and formalization of knowledge (study of cases) Surgical Oncology : model and formalization of knowledge management (colorectal cancer study of cases - CCR).
3. VDI Public Administration: Municipalities Digital Network - a machine on demand application that handles certain specificities linked to the LOPD (Ley

Orgánica de Protección de Datos in spanish). Application in the cloud for managing the Data Protection Act for Provincial Councils and Deputations. Migration of the corporate systems of the Council of Leon, a Cloud Computing environment on FCSCL systems.
4. VDI Outsourcing. Outsourcing of resources and services. Automated Diagnostic System for Energy Efficiency

By the interconection of various Networks Communications Science and Technology of EU Member States-Regions SUDOE facilitates access to prototypes and applications and/or developments adapted to the Cloud paradigm developed by the project participants entities or any other end user of interest in other SUDOE territories.

4 Indicators of FI4VDI project

The indicators are a useful tool to determine the degree of progress of the project. These measuring instruments used to assess changes in relation to the initial situation and to detect difficulties or deviations from the work plan.
There are three categories:
Performance indicators - generally corresponds to the products resulting from the implementation of the project.

- R+D projects with added value from an environmental perspective
- Companies and SMEs that have been part of innovation partnerships funded
- Projects of telecommunications networks and for encouraging the application of ICTs
- Deployment Type: Private/Public/Hybrid/Community
- Participation in Conferences, Workshops and Working day diffusion

Outcome indicators - reports on the short-term effects on the target audience of the operation financed.

- New technologies developed
- Transnational cooperation in innovative networks created
- Tools (applications and services) for technological transfer between companies and technological centers and SMEs adopted in SUDOE countries/regions
- Companies and SMEs that have benefited from project results conducted
- SUDOE area with improved access to ICTs
- Publications resulting from the project

Impact indicators - This indicator will reflect the result of the action taken in the long term. This is to assess the impact beyond the direct and immediate effect on direct beneficiaries.

- New technologies transferred to enterprises, SMEs and/or managing entities
- Permanent networks of cooperation established
- Agents (institutions, enterprises, SMEs, etc.) connected to telecommunication networks created
- Rate of adoption of Cloud computing model
- Rate of knowledge of Cloud computing and virtualization technology

5 Results

Here FI4VDI Project expected results are described:

- Creation of a Platform as a Service (PaaS) for mass deployment of virtual desktops. Federation of the participant supercomputing centres infrastructures.
- Creation of an innovative cloud computing service aimed at users in the public and private sectors.
- Enhanced competitiveness and cost savings in the sectors targeted by the service created.
- Establishment of strategic recommendations: definition of reliable models for cloud computing (service levels, system capacity, system restoration, interoperability through shared service infrastructures, migration models), identification of service areas and evaluate and promotion of cloud computing as a tool for achieving cost savings and technological optimisation.
- Establishment of technological recommendations: identification of existing solutions and possibilities, assessment of the real capacity of suppliers, selection of an applications and systems map, initiation of a cloud computing strategy by adopting private clouds and infrastructure and platform services, establishment of system migration plans and identification of cloud computing as a model that fosters other technologies which are either emerging or in the process of expansion, such as energy sustainability in the area of IT or open source solutions.
- Establishment of management recommendations: definition of systems for assessing investment returns, analysis of organisational impact and identification of change management models, development of new contracting models and practices, standardisation and organisation of common services and definition of risk analysis models.

6 Conclusions

The spread of cloud computing in the Spanish public sector is still limited, and is more common among local authorities than among regional and state institutions. In general, the most common mode of deployment is private.
Savings, efficiency and simplicity are the reasons that have prompted public authority institutions to contract cloud computing services.
Those public institutions that decided to adopt cloud computing did so after carrying out a legal analysis focused primarily on data protection legislation.
According to the public authority bodies that have adopted cloud computing, the principle benefits of this model are the savings in costs and time it represents, whilst integrity of services and data was identified as the main difficulty.
The Spanish public sector perceived the cloud as a technological and, above all, operational advantage which has met their initial expectations regarding cloud computing.
Among the public authority bodies already using cloud computing, the future

prospects are very bright: they intended to continue working in the cloud, they would recommend this technology to other institutions and they expected to continue to obtain future benefits from using the cloud.
However, those public institutions that are not yet using the cloud were more wary: few intended to incorporate technological solutions, and only a minority of these would consider cloud computing.

Acknowledgments
Project FI4VDI was funded by the INTERREG IV B SUDOE Territorial Cooperation Programme.

Fig. 5. Ref. SOE4/P3/E804-FI4VDI. Project FI4VDI-SUDOE corporate logo.

References

1. Redondo Gil, C, Virtual Desktop Infrastructure (VDI) Technology: impact of ICTs on the ecological footprint FI4VDI - SUDOE Project. ICREPQ14, n 466. 2014 Available: http://www.icrepq.com/icrepq
2. Introduction to desktop virtualization, www.escritorios-virtuales.net. April 2014
3. Technical definition desktop virtualization, www.escritorios-virtuales.net. April 2014
4. Permanent availability of the applications, www.escritorios-virtuales.net. April 2014
5. How to save and perform at their best thanks to remote access, www.escritorios-virtuales.net. April 2014
6. Virtualization Management, one of the fastest growing areas in 2009, www.techweek.es. 2014
7. Virtualization, efficiency and environment, www.escritorios-virtuales.net. April 2014
8. OpenNebula `http://opennebula.org/documentation/`. 2014
9. ownCloud `http://owncloud.org/`. 2014
10. Arstides Guillermo Gonzlez Pol, Ing. Lilia Rosa Garca Perellada, Ing. Pedro Eliesther Vigil Portela, Dr. Alain Abel Garfalo Hernndez. Propuesta de las arquitecturas de servidores, red y virtualizacin de una nube privada que brinde infraestructura como servicio (IAAS1). 2012
11. Fan Yang; Comput. Sch., China Univ. of Geosci., Wuhan, China ; Li Pan ; Muzhou Xiong ; Shanyu Tang. Establishment of Security Levels in Trusted Cloud Computing Platforms. 2013

Virtual Hybrid Elastic Clusters in the Cloud

A. Calatrava**, G. Moltó, M. Caballer, and C. De Alfonso

Instituto de Instrumentación para Imagen Molecular (I3M).
Centro mixto CSIC - Universitat Politècnica de València - CIEMAT
Camino de Vera s/n, 46022 Valencia, España
Tel. +34963877356, Fax +34963877359

Abstract. Clusters of PCs are one of the most widely used computing platforms in science and engineering, supporting different programming models. However, they suffer from lack of customizability, difficult extensibility and complex workload-balancing. To this end, this work introduces virtual hybrid elastic clusters that can simultaneously harness on-premise and public Cloud resources. They can be fully customized, dynamically and automatically enlarged and shrinked (in terms of the number of nodes) and they offer migration capabilities to outsource workload from one datacenter to another (or to a public Cloud) in different situations. This is exemplified with case studies that involve a parallel computationally intensive gyro kinetic plasma turbulence code running on such hybrid clusters with resources provisioned from an on-premise OpenNebula Cloud and the Amazon Web Services public Cloud.

Keywords Cloud Computing, Elasticity, High Throughput Computing, Cluster computing, Migration

1 Introduction

The usage of clusters of PCs as a computing facility is widespread in the scientific community with high success for both High Performance Computing (HPC) and High Throughput Computing (HTC). Nevertheless, these computing platforms suffer several drawbacks. On the one hand, physical clusters cannot be easily enlarged or shrinked, without downtime, according to the dynamic requirements of the organization. On the other hand, they lack the ability to provide a tailored execution environment customized for each application to be executed, specially when incompatible libraries are required by different applications running on the same cluster. Moreover, the ability to transparently migrate running jobs to be executed on other physical resources is not a trivial task, which is an important feature for effective workload balancing. Another important limitation is the large upfront investment together with the maintenance cost of such computing equipment for small and medium-sized research groups or organizations [1].

Traditionally, virtualization was not considered a viable option for HPC due to the overhead costs in I/O and network devices, but the major improvements in hypervisor technologies and virtualization have paved the way for Cloud computing.

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

This paradigm can solve those disadvantages with customizable virtual machines (VMs) that decouple the application execution from the underlying hardware and are dynamically provisioned and released on a pay-as-you-go basis [2]. This is the case of public Infrastructure as a Service (IaaS) Cloud providers such as Amazon Web Services (AWS) [3] or on-premise Cloud infrastructures deployed by Cloud Management Platforms (CMPs) such as OpenNebula [4], OpenStack [5] or Eucalyptus [6].

In the last years, the adoption of on-premise Cloud solutions for datacenters has increased [7], in part due to the maturity of the CMPs, which are becoming reliable enough for production-ready workloads. This enables to create virtual hybrid elastic clusters that ease extensibility and customizability in contrast to physical clusters. Virtualization also enables to partition a physical cluster into different virtual clusters specifically customized for the applications. Moreover, decoupling the application execution from the underlying hardware paves the way for migrating running applications from one hardware to another. Migration is convenient for several reasons. First of all, it enables to move workload from an overloaded datacenter (source infrastructure) to another (where the former or the latter could be a public Cloud infrastructure) to achieve workload balancing. Second, a planned maintenance on the source infrastructure might affect an application that has been running for a long time, so migration introduces the flexibility required in a modern datacenter.

This work focuses on harnessing resources from hybrid scenarios that simultaneously include resources from on-premise and public Clouds in order to create ready-to-use customized virtual elastic clusters that satisfy the dynamic computing requirements of an individual or organization. The remainder of this paper is structured as follows. First, section 2 covers the related work in this area. Next, section 3 focuses on the main characteristics of the proposed cluster deployment architecture pointing out the main scenarios where the use of this type of clusters is beneficial. Then, the paper introduces the elasticity rules employed to automatically enlarge and shrink the cluster. Afterwards, two case studies are presented in section 4 that justify the benefits of the proposed approach. Finally, section 5 summarizes the main achievements and discusses the future work.

2 Background and Related Work

There are previous works in the literature that aim at deploying virtual clusters on Cloud infrastructures. For example, StarCluster [8] enables the creation of clusters in Amazon EC2 from a predefined configuration of applications (Open Grid Scheduler, OpenMPI, NFS, etc.). Along with StarCluster a plugin called Elastic Load Balancer has been developed that is able to enlarge and shrink the cluster according to the length of the cluster's job queue. This plugin typically runs on the local computer from which the cluster was deployed and requires permanent connection to the Cloud infrastructure, in order to create and destroy the VMs. Therefore, any disconnection on the client computer means a loss of control of the elasticity capabilities of the cluster. Instead, our proposal focuses on self-managed

hybrid clusters, where the elasticity rules are embedded in the cluster and no external entity is required. ViteraaS [9] allows creating virtual clusters to manage the execution of user-defined jobs, but the user cannot remotely access the cluster. The standard distribution of Hadoop [10] includes a utility to create a virtual cluster in the Amazon EC2 infrastructure, managed by the Hadoop middleware. The utility powers on the master virtual machine, using a pre-defined Amazon Machine Image (AMI) and creates the computing nodes using another AMI, performing the required network configuration.

There are commercial solutions, like IBM Platform Dynamic Cluster [11] that aims at partitioning the owned resources to deliver each user a custom cluster with specific features. It has features such as live job migration and automated checkpoint restart. The drawback in this case is that this product is oriented to manage on-premise infrastructures and cannot be connected to commercial Cloud providers. CycleCloud [12] is a service provided by CycleComputing that deploys virtual clusters, but it is restricted to Amazon EC2. This service provides the user with a virtual cluster based on Torque [13] or Condor where a subset of popular scientific applications, offered by CycleCloud, are installed. The user is able to manage the virtual nodes using a web interface, and it is possible to configure the cluster so that it is automatically sized based upon pending jobs.

All of the above examples only consider the usage of one Cloud infrastructure (no hybrid scenarios supported). Concerning the creation of clusters over hybrid Cloud infrastructures, the authors of works such as [14], [15] and [16] have analyzed architectures, algorithms and frameworks to deploy HTC clusters over this type of infrastructures. They analyze the performance of virtual clusters deployed on top of hybrid Clouds obtaining good results that demonstrate the feasibility of these kind of deployments. The works use a fixed number of on-premise nodes, and scale out the cluster using public nodes. However, workload migration from one infrastructure to another is not considered. In [17], [18], [19], the Nimbus toolkit is employed to implement and evaluate an elastic site manager, which dynamically extends existing physical clusters based on Torque with computational resources provisioned from Amazon EC2 according to different policies. A similar approach is employed in [20], where the benefits of using Cloud computing to augment the computing capacity of a local infrastructure are investigated, but no details about the underlying technologies are given.

An important point in a hybrid environment is the interconnection between nodes that are deployed in different Clouds. The authors of [21] introduced the concept of Sky Computing, which enables the dynamic provisioning of distributed domains over several Clouds. They pointed that one of the shortcomings of this approach is the need of trusted networking environments. The same authors of the previous work [22] study the inter-Cloud communication problem across independent Clouds in detail.

Our previous work in the field is Elastic Cloud Computing Cluster (EC3) [23], a tool that creates elastic virtual clusters out of computational resources provisioned from IaaS Clouds, but no hybrid scenarios are supported. These clusters are self-managed entities that scale out to a larger number of nodes on demand, up to a maximum size specified by the user. Whenever idle resources are detected, the

clusters dynamically and automatically scale in, according to some simple policies, in order to cut down the costs in the case of using a public Cloud provider. This creates the illusion of a real cluster without requiring an investment beyond the actual usage.

3 Virtual Hybrid Elastic Clusters

The objective of the proposed architecture is to ease the deployment and management of customized virtual hybrid elastic clusters whose computational resources are simultaneously provisioned from on-premise Clouds and from different public IaaS Cloud providers. In order to have all the nodes that compose the cluster in the same subnetwork, regardless of their physical location, it is convenient to use VPN (Virtual Private Network) techniques. A VPN enables a computer on a public network, such as the Internet, to exchange data as if it was connected to the private network.

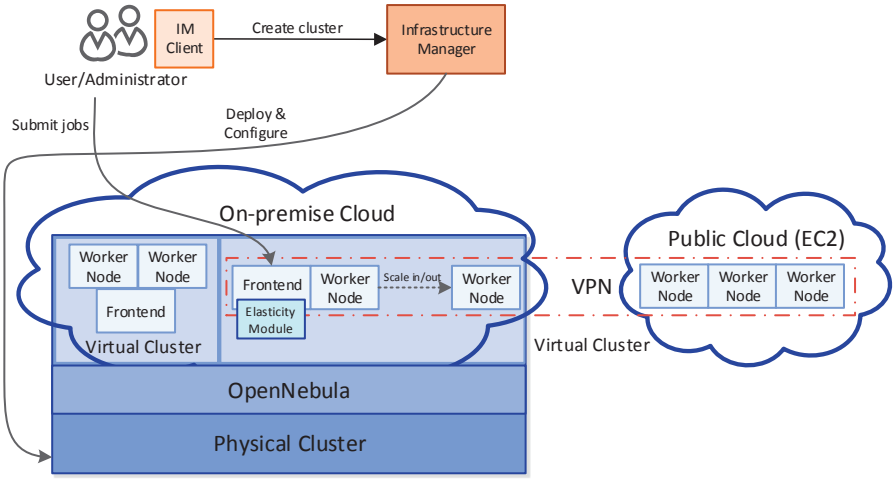


Fig. 1. Proposed cluster deployment architecture.

Figure 1 shows the proposed architecture to deploy virtual hybrid elastic clusters. The user (or the system administrator of the datacenter) requests a cluster deployment to the Infrastructure Manager (IM) [24], a component that is in charge of contacting different CMPs in order to deploy the VMs that compose the virtual cluster. As a summary, the lifecycle of the virtual hybrid elastic cluster consist of three phases: (1) the creation of the infrastructure, (2) configuring the computing nodes to behave as a cluster and (3) managing the elastic cluster.

Firstly, the user provides the IM with the initial number of nodes of the cluster, and their distribution among the on-premise and/or public Clouds accesible with his credentials. The IM is in charge of phase (1). In the scenario shown in the Figure, the frontend (head node) is deployed on the on-premise Cloud, but it could also be deployed on a public Cloud. Two corner cases are also considered, which are deploying a virtual cluster fully on a public Cloud or fully on an on-premise Cloud. The latter complies with the requirements of research communities that require instant access to cluster-based computing with data that does not leave the boundaries of the organization.

Once the computing resources for the cluster have been provisioned, the Ansible tool is employed in phase (2) for the configuration via a set of high-level recipes. This requires to configure the VPN-based network. For that, we rely on OpenVPN [25] to implement network tunnels between each individual Cloud resource and the organization's private network. By default, the frontend of the cluster hosts a VPN server, to produce self-managed clusters that do not require external VPN services to work. However, our solution could also rely on the VPN services provided by the organization. The configuration process also installs and configures the Torque Local Resource Management System (LRMS) that is in charge of scheduling the execution of the jobs. Moreover, the dependences of the application that is going to be executed in the virtual cluster are installed and configured in this step.

When the virtual cluster is running, the users can connect via SSH to the frontend and execute their jobs. At this point starts phase (3). The workload of the cluster is periodically evaluated by the elasticity module, a component that can run inside the frontend (to produce self-managed clusters) or outside, and is in charge of monitoring the state of the LRMS queue to enforce a set of elasticity rules (described in section 3.1) to trigger actions in order to scale out (increase) or scale in (decrease) the number of nodes of the cluster. The user can choose from a set of elasticity rules as well as the maxium number of nodes of the cluster (growth limit).

From the point of view of the user, the use of a virtual hybrid elastic cluster enables proper customization of the execution environment, thus ensuring compatibility with the applications to be executed. The automatic elasticity, together with the workload balancing capability, enables to reduce the total execution time of the jobs by dynamically adapting the size of the cluster to the workload.

In addition, the system administrator is able to manage the resources of the virtual cluster with unprecedented flexibility due to the migration capabilities. For example, this enables a system administrator to temporarily outsource cluster-based workloads to a public Cloud (or to another datacenter running an on-premise Cloud) to deal with a planned outage.

3.1 Elasticity Rules

The elasticity module included in the architecture dinamically adds and removes worker nodes from the cluster by monitoring the frontend job queue. This module uses various policies to determine when to deploy additional VMs (scale out) in

the Cloud or terminate them (scale in) based on the monitored information. The behaviour of this module is described in Algorithm 1.

Algorithm 1 Elasticity management

```
Require: Growth limit, l > 0, scale_out policy, scale_in policy
  while the frontend is running do
    Obtain the number of jobs in queue, j , and the total number of nodes, n
    if j > 0 and n < l then
      Apply scale_out policy
    end if
    if j == 0 then
      Obtain state of the nodes
      if some node state is free or offline then
        Apply scale_in policy
      end if
    end if
  end while
```

The elasticity rules or policies can be proactive or reactive. Proactive rules can be employed if job execution patterns in the clusters are known, in order to deploy and configure the nodes just in time for the execution of the jobs arriving at the LRMS. However, proactive rules typically underperform with stochastic job execution patterns. Therefore, this paper focuses mainly on reactive elasticity policies that deploy and relinquish resources based on the actual state of the cluster.

The scale out policies determine when it is necessary to increase the number of worker nodes of the cluster. Two policies are included: i) *On demand*, where for each job in the queue a worker node is deployed. Therefore, the jobs will wait for the deployment and contextualization of the node before they start their execution and ii) *Bursts*, this policy deploys a group of VMs for each job in the queue, assuming that if a job arrives at the LRMS there is an increased chance for other jobs to arrive shortly (thus including some proactivity). For HTC-based applications, such as Bag-of-Tasks or parameter sweeps, this policy is expected to reduce the average waiting time of the jobs at the expense of an increased cost (economic, in the case of public Clouds or due to idle resources in the case of on-premise Clouds).

The scale in policies determine when it is necessary to decrease the number of worker nodes. Two policies are considered: i) *On demand*, to terminate the idle worker nodes when there are no pending jobs in the LRMS and ii) *Delayed shutdown*, where idle worker nodes are terminated after a certain amount of configurable time. This is of interest when using public Clouds that bill by the hour, where idle nodes are kept available for job executions before the hour expires, even if no jobs are available to be executed at the moment.

Notice that when there are no jobs in the LRMS for a period of time, all the worker nodes will be eventually terminated, regardless of the elasticity policies, thus resulting in a cluster with only the frontend running.

4 Case Study

In order to evaluate the benefits and impact and to validate the developed system on real scenarios, two case studies are proposed using a computationally intensive parallel (hybrid MPI/OpenMP approach) scientific application. The GENE [26, 27] version 11 (release 1.7) provides a state-of-the-art nonlinear gyrokinetic solver aimed to efficiently compute the microturbulence and the resulting transport coefficients in magnetized fusion and astrophysical plasmas. The application requires an MPI-2 environment, Fortran and C compilers, and the BLAS and LAPACK libraries.

The infrastructure used to deploy the on-premise Cloud is composed by an heterogeneous blade-based system that has 4 kind of nodes: 2 x (2 quad-core L5430@2.6 Ghz, 16 GB), 2 x (2 quad-core multithreaded E5520@2.26 GHz, 16 GB), 6 x (2 quad-core multithreaded E5620@2.4 GHz, 16 GB) and 3 x (4 quad-core multithreaded E7520@1.86 GHz, 64 GB), with a total amount of 128 cores and 352 GB of RAM. The blade system is backed by a 16 TB SAN connected via a private gigabit ethernet network. This system is managed by OpenNebula 4.4, using KVM as the underlying hypervisor. With respect to the public Cloud, we relied on Amazon Web Services. The instance type chosen is `m1.medium`, with one (virtual) processor, 3.75 GB of RAM and 410 GB of HD. In this way, the VMs deployed on the on-premise Cloud have the same characteristics. The Virtual Machine Image (VMI) used in the on-premise Cloud provides the same execution environment as the EC2 AMI employed (`ami-e50e888c`). Both images provide a pristine installation of Ubuntu 12.04 LTS.

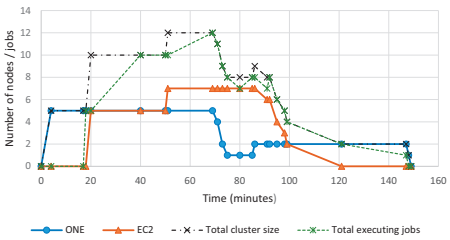
The case studies consist of jobs composed by 4 MPI processes in each node, using a maximum of 1000 MB per process. These processes communicate each other inside the node, but they do not communicate with external processes. The case studies have been downsized to get job executions in the order of minutes (instead of hours) to better focus on the dynamic cluster topology and job scheduling. The limit to the growth of the cluster has been fixed to 15 nodes.

4.1 Cloud bursting

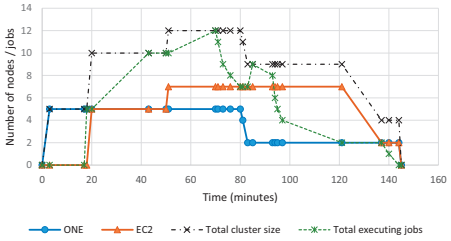
Our first case study focuses on automatic Cloud bursting, where a cluster running on an on-premise Cloud is automatically enlarged with computational resources provisioned from a public Cloud, with no service disruption. This enables to cope with an increased workload. For that, the user deploys the cluster and submits the jobs for execution. The cluster eventually becomes overloaded and, therefore, the elasticity module in the cluster that is monitoring the status of the queue decides to deploy new worker nodes (scale out) to handle the execution of the pending tasks. The new worker nodes can be on-premise resources, which are favoured, or/and public resources if no more resources can be allocated on-premise. We are going to assume that the user is restricted to 5 VMs simultaneously running on the on-premise Cloud, and the frontend is going to be able to execute jobs (but it cannot be shutdown unless the user terminates the cluster). The goal of this case study is to analyze the behaviour of the scale in/out policies described in section 3.1.

In the three cases, the workflow is as follows: the user requests an initial cluster of 5 nodes on the on-premise Cloud to the IM. When the initial cluster is deployed and configured (minute 16-18), the user submits 10 jobs to the cluster. Regardless of the decision of the elasticity module, the user will submit two new jobs in minute 50. This behavior will be repeated in minute 85, submitting two more jobs to the cluster (14 jobs in total). It should be noticed that the average duration of each job performed in the case study is 55 minutes, regardless of the underlying infrastructure.

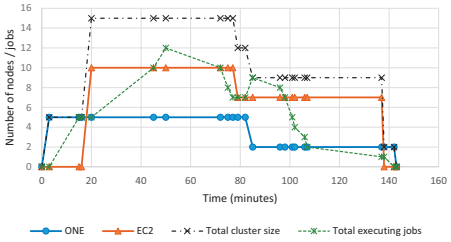
The bursts policy has been configured to deploy twice as much jobs were in the job queue. The behaviour of the delayed shutdown scale out policy works as follows: the on-premise nodes are terminated 10 minutes after they finished executing their jobs, if there are no more jobs waiting in the queue. Since Amazon bills by the hour, the nodes deployed in Amazon EC2 are terminated minutes before the paid hour goes by.



(a) Scale out: on demand. Scale in: on demand



(b) Scale out: on demand. Scale in: delayed shutdown



(c) Scale out: bursts. Scale in: delayed shutdown

Fig. 2. Evolution of cluster's size and nodes distribution for different policies.

Figure 2 shows the behaviour of the three possible combinations of the elasticity policies. Note that the combination of the bursts (scale out) policy with on demand (scale in) policy is not possible because the extra nodes that are deployed for future jobs would be shutdown immediately by the monitor system. The elasticity rules periodically examine the job queue (every 10 seconds for these tests), executing a policy and resizing the cluster if required.

The Figure shows that in (a) there are no idle periods in the worker nodes. The lifetime of the VMs is adapted to the workload of the cluster, but every time the cluster has a job waiting in the queue, a period of deployment plus contextualization is needed (4 periods in this case, minutes 4-17, 20-40, 51-69 and 85-92). In contrast, (c) has only 2 periods of deployment and contextualization (minutes 3-15 and 20-45), but the use of the nodes is not optimized, having idle machines the most part of the time. (b) is the intermediate solution, where the utilization of computing resources is better than (c) and there are 3 periods of deployment plus contextualization (minutes 3-17, 20-43 and 51-70). The contextualization process is faster in the ONE nodes than in the EC2 nodes because of the network, since the IM is deployed in the infrastructure as the on-premise Cloud deployment.

Notice that every time a node is terminated, the cluster needs another period of contextualization in order to reconfigure and restart Torque, what requires an average of 3 minutes. A new node form the public Cloud can be included in the cluster in 15 to 20 minutes, and it does not affect to the execution time of jobs because this process is done in parallel. Similarly, when a new node is added to the deployed infrastructure, the contextualization process mainly affects to the new nodes, that need to install all the software. The rest only need to reconfigure their `/etc/hosts` file and restart Torque, but it does not affect the execution of the jobs. The total execution time of the jobs is similar in the three combinations (142 minutes in (c), 144 in (b), and 148 minutes in (a). This is because the execution of the last job burst (composed by 2 jobs) in minute 85, needs almost an hour to be executed.

4.2 Maintenance period in a datacenter

Maintenance periods or planned outages are very common situations in a datacenter, and they might cause several inconveniences for the users of the datacenter resources that the system administrator must deal with. Therefore, the second case study is focused on the migration capabilities of the cluster, a very useful feature from the point of view of the system administrator.

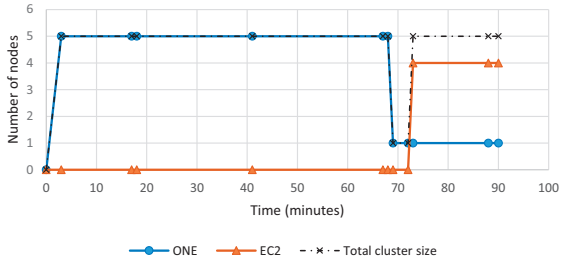


Fig. 3. Ordered shutdown of nodes during a planned maintenance period.

First of all, when the user or system administrator requests a shutdown for some worker nodes, two different approaches can be used: i) *Aggressive shutdown*, that removes worker nodes without waiting for the completion of the jobs being executed, and ii) *Ordered shutdown*, which waits for the jobs to finalize their executions before shutting down the worker nodes.

An aggressive shutdown should not be used on tightly coupled parallel applications, where the loss of a worker node ruins the execution of the job unless application checkpointing is employed. However, for embarrassingly parallel independent jobs, this approach can be combined with a job resubmission system that enables to execute again the failed tasks. This, of course, affects the total execution time of the jobs. In contrast, this total execution time is not affected by using the ordered shutdown, because the system waits until the end of the jobs execution to terminate the nodes. Therefore, we will use this latter approach to illustrate the migration capabilities of our developed system.

Figure 3 shows the evolution of the distribution of nodes when the cluster is asked to migrate part of its nodes. In this case study, the chosen policies where on demand (scale out) and delayed shutdown (scale in). It started with an initial cluster of 5 on-premise nodes, where the user is executing 5 jobs (minute 18). In the minute 41, the sysadmin requests the migration of 4 worker nodes to the IM client because of a maintenance period in the physical infrastructure. So, the system disables these worker nodes, by setting their state to `offline`, so that they are not assigned new tasks. This state change does not affect the execution of the current jobs, which terminate in minute 67-68. One minute later, the monitor terminates the nodes, reconfigures the cluster and deploys 4 new nodes in the public Cloud. As a result, the migration process is completed in minute 73, where the 4 worker nodes that originally composed the cluster are now allocated on Amazon EC2 resources. When the user launches new jobs (minute 90), they are going to be executed by the new nodes, so the migration process was completely transparent for the user.

5 Conclusions and Future Work

This paper has introduced a software architecture that abstracts the details of cluster deployment and configuration over hybrid Clouds. The system features the provision of virtual hybrid elastic clusters, composed by on-premise resources and public resources from public Cloud providers. Moreover, the system is able to configure these resources to support the execution of the applications, and to adapt the cluster's size and topology to the dynamic characteristics of the application and the needs of the datacenter. The benefits of the proposed architecture have been exemplified by the execution of a computationally intensive gyro kinetic plasma turbulence application, that demonstrates the feasibility of this type of architectures into de scientific community.

The future work involves improving the migration capabilities of the cluster. Different schemes are going to be considered, from the migration of the virtual clusters to another physical infrastructure (involving the migration of the frontend), to live virtual machine migration. This will introduce an unprecedented flexibility

to decouple cluster-based computations from the physical infrastructure on which the cluster was initially deployed. Moreover, this will enable datacenter migration, or the ability to migrate running computational resources from one datacenter to another. Also, we consider to develop new elastic policies that improve the performance of our virtual hybrid elastic clusters. Spot instances from Amazon EC2 are going to be considered in order to reduce the total cost over public Clouds.

Acknowledgements

This work has been developed under the support of the program "Ayudas para la contratación de personal investigador en formación de carácter predoctoral, programa VALi+d", grant number ACIF/2013/003, from the Conselleria d'Educació of the Generalitat Valenciana. Also, the authors wish to thank the financial support received form The Spanish Ministry of Economy and Competitiveness to develop the project "CLUVIEM", with reference TIN2013-44390-R.

References

1. De Alfonso, C., Caballer, M., Alvarruiz, F., Moltó, G.: An economic and energy-aware analysis of the viability of outsourcing cluster computing to a cloud. Future Gener. Comput. Syst. **29**(3) (March 2013) 704–712
2. Mell, P., Grance, T.: The NIST definition of cloud computing. Technical Report 800-145, National Institute of Standards and Technology (NIST), Gaithersburg, MD (September 2011)
3. Amazon: Amazon Web Services (AWS). `http://aws.amazon.com` [Online; accessed 5-December-2013].
4. Sotomayor, B., Montero, R.S., Llorente, I.M., Foster, I.: Capacity Leasing in Cloud Systems using the OpenNebula Engine, Cloud Computing and Applications 2008 (CCA08) (2009)
5. OpenStack: OpenStack Cloud Software. `http://openstack.org` [Online; accessed 5-December-2013].
6. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: The eucalyptus open-source cloud-computing system. In: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. CCGRID '09, Washington, DC, USA, IEEE Computer Society (2009) 124–131
7. Gartner: Gartner Says Nearly Half of Large Enterprises Will Have Hybrid Cloud Deployments by the End of 2017. `http://www.gartner.com/newsroom/id/2599315` [Online; accessed 5-December-2013].
8. MIT: Starcluster. `http://web.mit.edu/stardev/cluster/` [Online; accessed 9-December-2013].
9. Doelitzscher, F., Held, M., Reich, C., Sulistio, A.: Viteraas: Virtual cluster as a service. In: Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science. CLOUDCOM '11, Washington, DC, USA, IEEE Computer Society (2011) 652–657
10. Bialecki, A., Taton, C., Kellerman, J.: Apache Hadoop: a framework for running applications on large clusters built of commodity hardware (2010)

11. IBM: IBM platform dynamic cluster. http://www-03.ibm.com/systems/technicalcomputing/platformcomputing/products/lsf/dynamiccluster.html [Online; accessed 10-December-2013].
12. CycleComputing: Cyclecloud. http://www.cyclecomputing.com/cyclecloud [Online; accessed 10-December-2013].
13. AdaptiveComputing: Torque resource manager. http://www.adaptivecomputing.com/products/open-source/torque/ [Online; accessed 5-December-2013].
14. Montero, R.S., Moreno-Vozmediano, R., Llorente, I.M.: An elasticity model for high throughput computing clusters. Journal of Parallel and Distributed Computing **71**(6) (2011) 750 – 757 Special Issue on Cloud Computing.
15. Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M.: Elastic management of cluster-based services in the cloud. In: Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds. ACDC '09, New York, NY, USA, ACM (2009) 19–24
16. Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M.: Multicloud deployment of computing clusters for loosely coupled mtc applications. IEEE Transactions on Parallel and Distributed Systems **22**(6) (2011) 924–930
17. Marshall, P., Keahey, K., Freeman, T.: Elastic site: Using clouds to elastically extend site resources. In: Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on. (2010) 43–52
18. Marshall, P., Tufo, H.M., Keahey, K., Bissoniere, D.L., Woitaszek, M.: Architecting a large-scale elastic environment - recontextualization and adaptive cloud services for scientific computing. In Hammoudi, S., van Sinderen, M., Cordeiro, J., eds.: ICSOFT, SciTePress (2012) 409–418
19. Duplyakin, D., Marshall, P., Keahey, K., Tufo, H., Alzabarah, A.: Rebalancing in a multi-cloud environment. In: Proceedings of the 4th ACM Workshop on Scientific Cloud Computing. Science Cloud '13, New York, NY, USA, ACM (2013) 21–28
20. de Assuncao, M.D., di Costanzo, A., Buyya, R.: Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In: Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing. HPDC '09, New York, NY, USA, ACM (2009) 141–150
21. Keahey, K., Tsugawa, M., Matsunaga, A., Fortes, J.: Sky computing. IEEE Internet Computing **13**(5) (2009) 43–51
22. Tsugawa, M., Matsunaga, A., Fortes, J.: User-level virtual network support for sky computing. In: Proceedings of the 2009 Fifth IEEE International Conference on e-Science. E-SCIENCE '09, Washington, DC, USA, IEEE Computer Society (2009) 72–79
23. Caballer, M., De Alfonso, C., Alvarruiz, F., Moltó, G.: Ec3: Elastic cloud computing cluster. J. Comput. Syst. Sci. **79**(8) (December 2013) 1341–1351
24. de Alfonso, C., Caballer, M., Alvarruiz, F., Molto, G., Hernandez, V.: Infrastructure deployment over the cloud. In: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. (2011) 517–521
25. OpenVPN: Open Source VPN. http://openvpn.net/ [Online; accessed 4-December-2013].
26. Dannert, T., Jenko, F.: Gyrokinetic simulation of collisionless trapped-electron mode turbulence. Physics of Plasmas **12**(7) (2005)
27. Görler, T., Lapillonne, X., Brunner, S., Dannert, T., Jenko, F., Merz, F., Told, D.: The global version of the gyrokinetic turbulence code GENE. Journal of Computational Physics **230**(18) (2011) 7053 – 7071

Migrating HPC applications to the Cloud: a use case of weather forecast

André Monteiro**[1], Cláudio Teixeira[1], Joaquim Sousa Pinto[1]

[1]Institute of Electronics and Telematics Engineering of Aveiro (IEETA), University of Aveiro, 3810-193 Aveiro, Portugal

{andremonteiro, claudio, jsp}@ua.pt

Abstract. On the last few years, Cloud Computing has grown and matured in a way that managed to compete with clusters and grids for scientific applications supported. The use of this technology has allowed minor projects to allocate proper computational resources to their demands, with lower budgets. On the other hand many conventional users keep ignoring the Cloud potential – fast deployment, pay-per-use, elasticity, adaptability, among others. Our objective was to prove the feaseability of scientific applications and delineate an outline work that researchers could follow to take advantage of the Cloud infrastructure.
This work describes the migration of a weather forecast application to the Cloud, demonstrating a system contextualization, application analysis, virtualization and deployment. We compare performance and combined costs of the several types of Clouds with the original solution and also future solutions.

1 Introduction

Scientific computing applications have historically been supported by datacenters' clusters or grids, but Cloud Computing (CC) evolved and matured in a way that can compete with these types of resources. CC is becoming extremely suitable for computing scientific applications as they advertise providing virtually unlimited amount of computing resources on demand and nearly in real-time [1] and high proficiency all along [2, 3]. For small organizations this paradigm change has been especially encouraging, as they do not easily access computational resources adequate to their needs. Another aspect is the fast resource and applications deployment. The Cloud scales in and out quickly on demand, showing no constraints on hardware deploying, configuration and maintaining. A dynamic shared computing infrastructure offers resources

** e-mail of corresponding author: andremonteiro@ua.pt

available on demand on pre-settled basis and is much more cost effective for a university to operate than public providers [4].

Migrating scientific applications to the Cloud is not a conventional procedure. Running systems need to be virtualized in order to create templates, operating systems need tuning with the new infrastructure and software requires adaptation to Cloud specifications. CC offers many potential benefits to researchers and scientists. However, many traditional resources' users seem to disregard the potential of the Cloud, based on the existing disadvantages – and there are a few - and inherent different deployment model. The maturing of the Cloud on the High-Performance Computing (HPC) field seems to be an excellent opportunity to take advantage of its intrinsic characteristics as elasticity and scalability, but also pay-per-use infrastructure. The Cloud adaptability makes possible the deployment of completely customized environments that perfectly fit the requirements of the final scientist's applications [5].

This paper describes the use case of transposing a scientific application to the Cloud, demonstrating a system migration and analyzing the output data. Our expectation is to lead to an increase of the number of scientific applications that could benefit from Cloud infrastructures, which can reduce the time for research cycles.

We start by contextualizing with the application – a weather forecast application – then analyzing the original problem, followed by a model implementation and Cloud application translation. Then we set the possible scenarios, ran the simulations and consolidated the results, leading to solid conclusions.

2 Related Work

There are several examples of Cloud migration, though the literature on HPC is short. Authors in [6] describe an experiment of migration the Hackystat application, an open-source collaboration tool. The study identifies some steps and requirements, leading to a modified architecture of the software at the end. In [7], the author presents a migration framework, which includes analysis, planning and implementation; after the integration, there is also an approach of tuning and workload standardization. However, this solution is only a model without practical example.

On the other hand, there are projects that aim to make na automatic handover to the Cloud. CloudMIG is model-based approach which addresses which aims supporting SaaS providers to semi-automatically migrate existing software systems to scalable and resource-efficient PaaS and IaaS-based applications [8]. It implements a workflow that analyzes and classifies software in order to map a model to support reengineering. The Cloud Motion Framework (CMotion) tries to address the problem of

migrating applications into and between different clouds [9]. To make previously incompatible technologies able to work together CMotion uses adapters, generating alternatives to deploy the application with the same functionality. Another project is CloudGenius, a framework that provides a migration process and decision support. CloudGenius has a limited applicability since it focuses on a migration process for single-tier Web applications.

3 CLiM@UA

In contrast with early models that were essentially statistic, modern weather forecast models are produced by numerical weather prediction systems, with sophisticated data assimilation systems for determining the initial state plus advanced high resolution dynamical global or regional models of the atmosphere [11]. These numerical simulations were one of the first applications of HPC remaining one of the most important today in terms of impact and relevance to the public [12].

The Weather Research and Forecasting (WRF) model, a widely employed mesoscale model used by the Advanced Research (ARW) solver, is based on geographic slices, called domains. Each domain can have sublevels of domains, named nested domains. Nesting stands for a form of refinement that allows higher resolution computation to be focused above a region of interest [13]. The WRF model on **Fig. 1**, for instance, is built over a parent domain (D1) with 90 km of spatial resolution, covering a portion of the North-Western Atlantic Ocean and all of the Iberian Peninsula. The first nested domain (D2), with a spatial resolution of 18 km, encompasses the Northern and Central part of the Portuguese territory. The innermost domain (D4) has a spatial resolution of 3.6 km and it is focused on the chosen area to simulate, located in central Portugal [14].

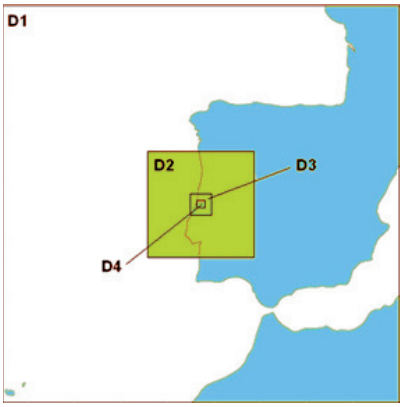


Fig. 1. Nest simulation domains (retrieved from [14])

The performance and reliability are extremely important assets of these numerical systems, both for people and industry. They can help not only to predict when and where it will rain but also the wind speed and the tides for electricity production.

One of the existing projects led by University of Aveiro Department of Physics consist in daily weather forecasts computed by CliM@UA [15]. The system performs regular forecasts over Portuguese mainland, Azores and Madeira archipelagos. Using WRF-ARW model [16], the system relies on sparse horizontal resolution in order to perform the task, but newer versions are much more complex and heavier in terms of computational resources. At the moment the forecast uses two nested domains for Portugal mainland, a sparse with 25 km of horizontal resolution and a finer one with 5km. Madeira and Azores also have nested grids with the same approach. In Azores due to the huge area involved one sparse domain is used with 25km and then 3 finer domains with 5 km are used for the east, center and west parts of the archipelago.

The weather forecast is performed four times per day, resulting in a 7 day forecast, 6 hours ahead from the original global model. The process starts with the download of the analysis and forecasts data at the synoptic hours - 00, 06, 12 and 18 Coordinated Universal Time (UTC.) Every day, weather data from the Global Forecast System (GFS) model [17] is downloaded to its servers, trimmed to precise geographical areas and then intensely computed to forecast fields as temperature, humidity, rain and fog, among others.

After successful download, the WRF Preprocessing System (WPS) converts the data which is then cropped to the geographic area of interest. The WRF model then uses the data prepared by WPS as input and boundary conditions and begins the simulation for Portuguese mainland, Azores and Madeira archipelago. After GFS data download from United States' servers, the pre-processing begins. The forecast data is decoded from the `grib` format into an intermediate format - which includes version number, gridded data common information, particular grid type specific information and a 2-dimensional slab of data - with the `ungrib` application. Then, the `metgrid` executable horizontally interpolates the data and vertical interpolation is made by the `real` program.

The regular forecast performed by CliM@UA requires 12 CPU-core for Portuguese mainland, 4 CPU-cores for Azores archipelago and 2 CPU-cores for Madeira archipelago. With the available hardware solution, there is a marginal window of a few minutes per forecast. Considering this window, finer scales are not feasible, since the overall forecast would be ready pass the beginning of the expected forecast.

HPC is one way to perform this task, recurring to Messaging Passing Interface (MPI) with parallel processing [18]. Currently this laboratory is using a head node with 4 CPU-cores and 4 physical slaves, each one built on 12 CPU-core at 2.66GHz with 12GB RAM for its projects and tasks. For

the weather forecast, a head node and a slave are normally used, but one more slave may be obtainable in the occurrence of unexpected events like storage faults or power outage (nature phenomena). This is enough to delay forecasts on our tight 6h window because it would be unsuitable to forecast the 13pm to 19pm period at 18pm. Currently the forecast horizontal scale is 5 km, because higher resolutions require more computing resources to deliver the results on time.

4 Migrating to the Cloud

The Institute of Electronic and Telematics Engineering of Aveiro (IEETA) has a private Cloud installed on a 52 CPU-core blades, running OpenNebula [19] and hosting other test bed services. Our expectation is to use this Cloud to aid weather forecast computing, enhancing the results and lowering the costs when possible. A key advantage to this project is also the possibility to burst to the public Cloud providing means to completion when some unexpected event occurs, causing delays. While the traditional cluster, currently used, is physically limited to its hardware capacity, the virtual cluster provided through the Cloud can extend beyond its outer bonds, filling the user with seamless unfinite resources. Using a template is an appropriate approach of standardizing a virtual machine to replicate on clusters, once it is completed some parameters (as CPU or RAM) can be adjusted on the template's configuration to adapt to cluster's specifications. The deployment is faster as well, including contextualization - every template contains VM context information, which provides configuration settings such as hostname, ip address and users.

4.1 Proposed model

Our structure, derived from the legacy structure, is composed by two VM templates: a first template containing one head node, with service configuration, and a second template for slave worker nodes, all with the same node configuration.

This structure can be easily redefined to include several head nodes with different configurations or different sets of worker nodes. These sets of working nodes can be characterized by computing capacity – it is mandatory to find suitable common hardware configurations. More modest machine configurations are able to work in smaller clusters, thus taking advantage of more machines, and bigger configurations which allow to minimize the VMs overhead and delays.

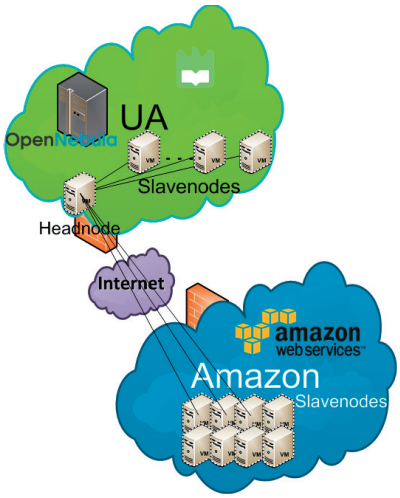


Fig. 2. Proposed model

As referred above, OpenNebula was used to provide the IaaS, installed on a Scientific Linux 6 OS. On the used model depicted on **Fig. 2**, where we can see the private IaaS to instantiate these VM templates but with the optional deployment on the public Amazon Elastic Compute 2 (Amazon EC2). EC2's choice was related to the single existing API for public providers held by OpenNebula. No price comparison and cost evaluation were made.

Starting our task to obtain results, we used templates to instantiate the nodes, using a setup driven from the original configuration – CPU, RAM and disk. The process was to replicate the initial schema, leaving enhancements for a later stage. The allocated extra disk for the head node, shared on NFS, is needed for the weather forecast raw data download and post processing; to schedule jobs and manage slave nodes, 2GB of memory and 2 cores is enough. As for the slave node templates, we started with 4GB of memory and 4 cores, which should be able to fit on most of the clusters' hardware specification.

After a clean install of CentOS6, all required software was installed: Fortran and C++ compilers, MPI, netcdf, zlib, jasper and libpng libraries were compiled. Once all requirements are fulfilled, the WRF-ARW was installed.

4.2 Configuring and deploying Virtual Machines

The head node shares the subnet of the worker nodes and information about shared disk partitions is already also configured. The head node maintains on `/etc/hosts` and `hydra.pt` files information about the

slaves' IP address and CPU core number. Information on the head node is crucial to MPI for making the best schedule for the jobs.

The creation of slave nodes is very quick due to the use of qcow2 images – copy on write from QEMU [20] – that leave the main file intact and create a separate one with only the write changes. Since in this case all data is handled on a shared storage, the OS image is not going to change and we may benefit from qcow2 to keep the I/O footprint as lowest as possible. A new VM instantiation and initial scripting for network configuration is done in less than 2 minutes, while a server installation would require a couple of hours. To deploy a new VM, we can create it from the template for OpenNebula in the command line interface or by a web browser application. When a new slave node is created, an OS boot script is executed to configure hostname, network, NFS sharing and update the necessary files on the head node with the node information.

To experiment a wide range of options and variables, 6 scenarios were built changing master, slave and shared disk locations. This gave us the opportunity to test public, private and hybrid Clouds on simulations. The transition from the traditional approach to a new Cloud approach was made smoothly, even with some tuning to be done, for instance in OS, network and storage. The used network was the UA Campus network with no defined Quality of Service (QoS), all data went through the Infiniband channel with all other traffic. As for storage, the master node was running NFS and sharing the folder with the slaves, but a centralized dedicated server (not virtual) with NFS or other could be helpful too.

To test these possible scenarios, we used the local IaaS with a host with a X5650@2.66GHz CPU as the private provider and Amazon EC2 Cloud as the public provider.

Table 1. Tested scenarios

Scenario	Headnode location	Slavenode location	Storage location
A	UA	UA	UA
B	UA	AWS	UA
C	UA	AWS	AWS
D	UA	UA+AWS	UA
E	AWS	AWS	UA
F	AWS	AWS	AWS

After building the master and slave templates, both on OpenNebula and Amazon EC2, we instantiated several VMs and ran the weather simulations. **Table 1** illustrates the analyzed scenarios where every possibility was considered to allow discarding worst cases and tune the remaining. Firewall and proxies were also configured to avoid network issues. The scenarios were set to optimize not only the servers' geographic location but

also the storage, a very important part of the system to test. On our experiments, we increasingly changed the location from on-premises resources to outside resources to foresee all possible results.

5 Results

After setting the possible scenarios, we ran the forecast simulations for a single period of a day (6h) to maintain equivalent data. To experiment a wide range of options and variables, these six scenarios were built changing master, slave and disk locations, as described before. Starting the scenario experiments, the first simulations were run with a single virtual server (scenario A), which acted as both master and slave for simplification purposes and storage via NFS. After a set of tests and tuning, we started to test all scenarios, firstly for performance in the form of elapsed time and secondly for energy consumption.

5.1 Performance

After running the simulations, the average results of three runs were gathered in **Table 2**, ordered by execution time. The prices shown take only in consideration the public Cloud usage prices, in-house costs as electricity and maintenance were not included. Regarding the private Cloud cost, there is no additional cost in using the existing campus' servers as they already utilize cooling and maintenance staff, whether they use 10%, 50% or 100%. As for power consumption, the difference in cost is slightly higher on intensive computing consuming, as detailed on the next subsection, but is not reflected here in the same way some authors do not consider it [21].

Table 2. Simulation results

Scen.	Head node	Slave node	NFS	Cores	CPU Clock	$/h	$ total	Elapsed Time ↑
F	AWS	AWS	AWS	12	2.60	$2.40	$12.00	4:34:31
Orig.	UA	UA	UA	12	2.66	$0.00	$0.00	4:56:30
A	UA	UA	UA	12	2.66	$0.00	$0.00	5:07:22
E	AWS	AWS	UA	12	2.93	$1.30	$7.80	5:36:20
B	UA	AWS	UA	12	2.93	$2.40	$14.00	5:34:44
D	UA	UA + AWS	UA	12	2.66 + 2.93	$2.40 + $0.00	$16.80	5:41:30
C	UA	AWS	AWS	12	2.93	$2.40	$16.80	5:45:10

Every MPI managed simulation started a program thread on each CPU-core, corresponding to the legacy system. The selected hardware from Am-

azon was the closest as we could get to our infrastructure, Intel® Xeon® processors from 2.60 to 2.93GHz. A precise match is very hard to reach due to hardware heterogeneity. Even when CPU clock is nearly the same (2.60 and 2.66GHz), the operations per second may differ because of CPU design and architecture. We can also rule out result normalization, because the time and money columns are compared straightforwardly.

5.2 Evaluating the solutions

There are penalties on late results regarding the data computing, which have the same effect as providing no results at all. In fact, the costs in not being able to deliver results to clients would reduce the monthly income and could prejudice future contracts. What is the better solution?

To compare the possible solutions, we took in account the cost of Cloud processing, either private or public, and the acquisition of new hardware, with all the associated costs: maintenance, cooling, power and warranty. We can neglect installation and maintenance - the IT center has space and all wiring/cooling in place and HP has a 3-year direct replacement warranty. A new similar blade, a HP ProLiant BL460c Gen8 E5-2650v2 32GB Server, would cost 4800 euros [22], about $3478. The blade's additional energy consumption is 0.3KWh when workload is 100% [23], which multiplied by the current 0.12$/KWh (enterprise medium cost in Portugal, assuming an equal probability of failure all day) sums a total of 0.86$ per day and 25.92$ per month.

On the public side there is also the cost of data transfer to EC2 and from EC2. Data transfer into Amazon EC2 from Internet is $0.00 using a private IP address and out of EC2 is $0.12/GB, up to 10TB per month [24]. This cost was not included on the previous results' section because it was below the 1GB free limit (0.7GB). On a prolonged use it becomes rather significant – 2.8GB per day.

We have 4 daily forecasts (6h), which sum a total of 120 per month, and 1460 per year. The new scenario of new hardware acquisition is also considered on a 3-year basis. Usually hardware has a 3-year depreciation, where it will be fully deducted on accounting terms and obsolete on computing capabilities, so we built a comparison with costs for a 3-year length. **Table 3** shows the data comparison relative to all solutions using a single solution 24/7.

We also made calculations for a hybrid solution, using the local infrastructure as a starting point and bursting to the public Cloud on failure occurrences, for instance occupied resources or technical fails. The graphic on **Fig. 3.** Solutions' cost break even shows the point of interception between the use of new hardware and resorting to scenario F on failures occurrences on a 3-year basis, approximately 7% of fails.

Table 3. Detailed costs of observed solutions

Scenario	Usage	Initial cost	Per 6h forecast	Per month	Per year	End of 3rd year
F (AWS	100%	$0.00	$12.08	$1377.75	$16762.61	$52915.83
Original	100%	$0.00	$0.00	$0.00	$0.00	$0.00
A	100%	$0.00	$0.00	$0.00	$0.00	$0.00
New hardware	100%	$3478.26	$1.02	$4.07	$122.22	$4461.17
F (AWS) (1% fails)	1%	$0.00	$12.08	$14.77	$177.24	$531.72
New hardware (1% fails)	1%	$3478.26	$79.25	$96.89	$1162.71	$3488.14
F (AWS) (5% fails)	5%	$0.00	$12.08	$72.49	$869.85	$2609.55
New hardware (5% fails)	5%	$3478.26	$16.33	$97.96	$1175.58	$3526.73
F (AWS) (7% fails)	7%	$0.00	$12.08	$98.44	$1181.26	$3543.77
New hardware (7% fails)	7%	$3478.26	$12.08	$98.45	$1181.36	$3544.09
F (AWS) (10% fails)	10%	$0.00	$12.08	$144.97	$1739.70	$5219.10
New hardware (10% fails)	10%	$3478.26	$8.28	$99.31	$1191.74	$3575.21

As the break even details, fails above 7% would imply that new hardware would be a more inexpensive solution after 3 years

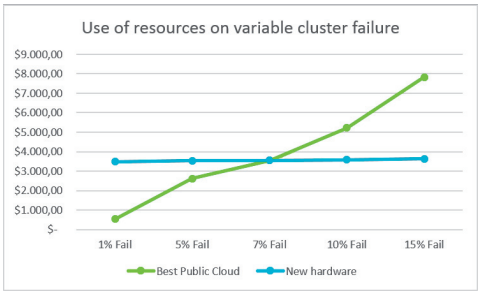


Fig. 3. Solutions' cost break even

6 Conclusions

Regarding the virtualization aspect, one of the keys for migrating to the Cloud, authors in [25] compare the performance of systems in a virtualized environment and on a physical environment, reporting about 5% CPU loss and 30% for I/O. Therefore we were expecting bigger virtualization overhead losses but the outcome revealed a more tightened window with non-virtualized/virtualized without HT (about 3.2%) and a wider one with virtualization with HT (about 6.5%). HT allows the CPU cores to share workloads, which normally can improve parallelization, especially on parallel programming. The original server had HT deactivated, but on a shared Cloud to utilize efficiently the resources we could not afford to disable it. Amazon EC2 also has this processor capability enabled, so it was also more consistent to build a faithful comparison. The first hybrid scenarios, E and B, had elapsed times approximately 11% higher than non-virtualized environment, while D and C had outcomes in the 14-15% range. This was a good result for a hybrid scenario, and validates using available on-premise resources, with minor additional cost inside the university campus. It also assures simulations on schedule with Cloud bursting to public Cloud providers if problems like hardware crash or over demand arise.

The problem with hybrid scenarios was the latency between both the master and slave nodes, but also the latency on NFS, experienced on both sides. With a more direct and dedicated connection between Amazon EC2 and UA the latency problem could be solved. A use of a public Cloud geographically closer, like Amazon Europe in Ireland would probably enhance the delay. Deployments in the public Portuguese universities' network can also work well, since the internal bandwidth situates around 80MBps for download and 10MBps for upload respectively. In terms of NFS the latency issue was already expected, as authors in [26] also point the same problem. However these times scales (seconds) can be mitigated within the final results, in a bigger scale (hours).

While investigating for the inexpensive and suitable solution, we could affirm that a full migration to the public Cloud would be very expensive at a medium-term. As we could see on **Table 3**, assigning tasks to public Clouds is economically viable to scenarios up to 7% of local node fails, on our case. After the 3 years period, the new hardware solution becomes more inexpensive.

We must also take in account that measuring performance in virtualized environments is complex; there can be changes in running VMs' usage and other services that can influence the entire infrastructure. A vast set of Cloud providers' benchmarks is detailed on [27] and yet it is not very simple to affirm the best within the results due to unidentified hardware and considerable heterogeneity. Like CPU comparisons that cannot be

only compared by their clock speed, there is not a unique variable to compare.

Although the migration result and cost can still be questionable, it is clear that CC can provide rapid access to computational resources as and when needed without the need for significant financial outlay [21]. However, a Cloud environment is unable to provide fix capacity planning during execution, and becomes an unpredictable scenario for applications such as the tested weather forecast. About our work we can state that the experience and successfully made the migration can point out some directions to many scientific applications.

Acknowledgements

We have a special gratitude to the Department of Physics of UA, especially to Tiago Luna, who presented all the insight about the forecast system, helping on hands-on tasks and also his coordinator Professor Alfredo Rocha for his kind support on this research.

References

1. Jakovits, P., Srirama, S.N.: Adapting Scientific Applications to Cloud by Using Distributed Computing Frameworks. Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on. pp. 164–167 (2013).
2. Brown, J.R., Dinu, V.: High performance computing methods for the integration and analysis of biomedical data using {SAS}. Comput. Methods Programs Biomed. 112, 553–562 (2013).
3. Roberto R. Expósito, Guillermo L. Taboada, Sabela Ramos, Jorge González-Domínguez, Juan Touriño, R.D.: Analysis of I/O Performance on an Amazon EC2 Cluster Compute and High I/O Platform. J. Grid Comput. 11, 613–631 (2013).
4. Carlyle, A.G., Harrell, S.L., Smith, P.M.: Cost-Effective HPC: The Community or the Cloud? Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. pp. 169–176 (2010).
5. García, Á.L., del Castillo, E.F.: Analysis of Scientific Cloud Computing requirement. (2013).
6. Babar, M.A., Chauhan, M.A.: A tale of migration to cloud computing for sharing experiences and observations. Proceeding of the 2nd international workshop on Software engineering for cloud computing - SECLOUD '11. p. 50. ACM Press, New York, New York, USA (2011).
7. Banerjee, J.: Moving to the cloud: Workload migration techniques and approaches. 2012 19th International Conference on High Performance Computing. pp. 1–6. IEEE (2012).
8. Frey, S., Hasselbring, W., Schnoor, B.: Automatic conformance checking for migrating software systems to cloud infrastructures and platforms. J. Softw. Evol. Process. 25, 1089–1115 (2013).

9. Binz, T., Leymann, F., Schumm, D.: CMotion: A framework for migration of applications into and between clouds. 2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA). pp. 1–4. IEEE (2011).
10. Jamshidi, P., Ahmad, A., Pahl, C.: Cloud Migration Research: A Systematic Review. IEEE Trans. Cloud Comput. 1, 142–157 (2013).
11. Zwiers, F.W., von Storch, H.: On the role of statistics in climate research. Int. J. Climatol. 24, 665–680 (2004).
12. Michalakes, J., Loft, R., Bourgeois, A.: Performance-Portability and the Weather Research and Forecast Model. HPC Asia. , Queensland, Australia (2001).
13. Dudhia, J., Gill, D.O., Henderson, T.B., Klemp, J.B., Skamarock, W., Wang, W.: The Weather Research and Forecast Model: software architecture and performance. 11th Workshop on the Use of High Performance Computing in Meteorology. p. 6. European Centre for Medium Range Weather Forecasts, Reading, United Kingdom (2011).
14. Carvalho, D., Rocha, A., Gómez-Gesteira, M., Santos, C.: A sensitivity study of the {WRF} model in wind simulation for an area of high wind energy. Environ. Model. Softw. 33, 23–34 (2012).
15. GMCUA: CliM@UA, http://climetua.fis.ua.pt.
16. Skamarock Klemp, J. 420 B., Dudhia, J., Gill, J., Barke, D. M., Wang, W. and Powers, J. G., W.C.: A description of the Advance Reaserch WRF version. (2005).
17. NCEP: The GFS Atmospheric Model, (2003).
18. Friedley, A., Lumsdaine, A.: Communication Optimization Beyond MPI. IEEE (2011).
19. C12G: The Open Source Toolkit for Data Center Virtualization, http://opennebula.org/.
20. Fabrice Bellard: Open source processor emulator, http://wiki.qemu.org.
21. Glenis, V., McGough, A.S., Kutija, V., Kilsby, C., Woodman, S.: Flood modelling for cities using Cloud computing. J. Cloud Comput. Adv. Syst. Appl. 2, 7 (2013).
22. Hewlett-Packard Development Company, L.P.: HP ProLiant BL460c Gen8 E5-2650v2 1P 32GB-R P220i/512 FBWC Server, http://www8.hp.com/pt/pt/products/proliant-servers/product-detail.html?oid=5409979#!tab=specs.
23. Beckett, J., Bradfield, R.: Power Efficiency Comparison of Enterprise-Class Blade Servers and Enclosures. (2011).
24. Amazon Web Services LLC: AWS | Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting, http://aws.amazon.com/ec2.
25. Huber Marcel Von Quast, Michael Hauck, and Samuel Kounev, N.: Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments, http://sdqweb.ipd.kit.edu/publications/descartes-pdfs/HuQuHaKo2011-CLOSER-ModelVirtOverhead.pdf, (2011).
26. Montero, R.S., Moreno-Vozmediano, R., Llorente, I.M.: An elasticity model for High Throughput Computing clusters. J. Parallel Distrib. Comput. 71, 750–757 (2011).
27. Gillam, L., Li, B., O¿Loughlin, J., Tomar, A.P.: Fair Benchmarking for Cloud Computing systems. J. Cloud Comput. Adv. Syst. Appl. 2, 6 (2013).

Automatic Genetic Sequences Processing Pipeline in the Cloud for the Study of Hereditary Diseases

Manuel Alfonso López Rourich (CénitS)[1], José-Luis González Sánchez (CénitS)[1], Pablo García Rodríguez (University of Extremadura)[2] and Felipe Lemus Prieto (CénitS)[1]

[1] CénitS (Research, Technological Innovation and Supercomputing Center of Extremadura) - N-521, Km. 41,8 10071 Cáceres (Spain)
{alfonso.lopez,joseluis.gonzalez,felipe.lemus}@cenits.es
[2] University of Extremadura - Escuela Politécnica de Cáceres, Avda. Universidad s/n 10003 Cáceres (Spain)
pablogr@unex.es

Abstract. Processing genetic sequences is a very complex task whose success depends on the appropriate orchestration of many tools with the objective of obtaining high-level genetic information to study hereditary diseases. To achieve this objective, geneticists perform a resequencing experiment which consists of mapping intermediate genetic sequences against a *consensus* reference to obtain variations which can influence the development of diseases whose origin is genetic. When performing a resequencing work flow, it is very important to know not only what are its different phases and the purpose of each task but to configure properly the used software. Determining whether the intermediate and especially the final results are correct is extremely important because in many cases this kind of analyses can be used within a clinic scope. In addition, when it comes to Next-Generation Sequencing (NGS) platforms, it is necessary to count on infrastructure which performs well during all processing tasks, includes significant storage capacity, strong Data Center security and the ability to manage access to user data. For these reasons, we propose a pipeline with which geneticists can complete the computational phases of a resequencing experiment, regardless of their knowledge of bioinformatics, in an automated manner so that at the end of the process high-level genetic information, useful to the study of hereditary diseases, can be obtained.

Key words: NGS, exome, pipeline, SNP.

1 Introduction

1.1 DNA sequencing

The progress of sanitary researching has always been one of the main challenges facing humanity. One of the most important discoveries has been the realization that the study of the DNA (Deoxyribonucleic Acid) sequence allows the functioning of fundamental biological processes within organisms to be understood.

DNA sequencing is a set of biochemical techniques and methods whose purpose is to determine the nucleotides within a DNA oligonucleotide. Until some years ago, DNA sequencing involved a huge temporal and economic cost. In the presence of these problems emerges NGS technology (the second-generation sequencing methods [1]), and with it, a new genetic paradigm which allows sequencing a whole genome or sections of it on a broad scale, with a very important reductions in the time taken and the cost of processing. However, although the cost of sequencing with NGS platforms has been considerably reduced, when resequencing, obtaining the genetic sequence of the whole genome is not worthwhile thus sequencing the whole exome (the section of a genome from which proteins are obtained in the transcription process) is preferred.

The COMPUTAEX Foundation[3] has undertaken several projects with the Hospitals Infanta Cristina in Badajoz and San Pedro de Alcántara in Cáceres, focused on the study of the sequence of the exome of certain patients, processed in the CénitS Data Center by using the LUSITANIA supercomputer.

In addition, its staff work on the second phase of Estirpex[4], a subproject of the CENITAL-2 project, whose main objective is the development of an on-line platform which allows access to a repository of historical and clinical data of residents from the region of Extremadura and therefore helps geneticists study hereditary diseases.

1.2 Resequencing experiment

Main phases. A resequencing experiment consists in sequencing the DNA of an organism to compare its genetic sequence with a *consensus* reference so that variations between the sequences can be found in order to study hereditary diseases. Its phases are described below:

1. **Primary phase**: The main objective is to create a collection of clones of DNA sections of interest from which millions of sequences of a certain length, called *reads*, will be generated.
2. **Secondary phase**: In this phase the *reads* are mapped (the verb "to align" is also used indistinctly) against a *consensus* reference sequence. The computational requirements are very high because NGS platforms generate a large volume of *reads* so both the infrastructure where this phase will be executed and the configuration of the software is critical.
3. **Tertiary phase**: The purpose of the last phase is to obtain genetic variations (SNPs, a change of a single nitrogenous base within a *read* aligned against a *consensus* reference sequence and small indels, that is, an insertion or a deletion of a small genetic sequence within an aligned *read*) and annotate them (annotating a genetic variant is a process in which text, from an ontology, or numerical data, from an interval, is associated with it in order to obtain its functionality). The results from this phase will be stored in a database of annotated variants.

[3] http://www.computaex.es/
[4] http://www.computaex.es/enlaces/publicaciones/estirpex

Genetic sequences *reads*. Since every biotechnology company which sells NGS platforms has its own sequencing techniques, *reads* generated by their sequencers represent the same data by using a different format. However, certain parameters should be known:

- The ***library***. Is a collection of clones of DNA sections which has been created in the first phase of a resequencing experiment. There are several types of *libraries* used to sequence a biological sample but the most popular is the *paired-end* (this kind of *library* generates pairs of sequences, called F3 and F5-P2, separated by an unknown sequence of *insert size* nucleotides), which allows genetic information obtained from processed *reads* to be more precise.
- The **length** of all the *reads* (those obtained from the sequencing of an exome must all be the same length).

Illumina *reads* are represented in FASTQ [2], a text-based format for storing biological sequences and their corresponding quality scores. Each one of those *reads* has an associated header which includes general data about the sequencing process itself and other information which specifically describes the *read*.

Variants representation. The result of a resequencing experiment is a VCF [3] (Variant Call Format) file, used to store variations detected in a genetic *reads* processing work flow and their annotations, which depend on the software used to perform the variant call (a process whose purpose is to detect variations within a mapped *read*) process.

In the subsection **2.4. Output data**, within the section which describes the implementation, **2. Pipeline implementation**, the information obtained by a pipeline user when completing different jobs will be described.

1.3 Genetic sequencing in CénitS

The service delivery model deployed in CénitS allows the last two phases of a resequencing experiment to be undertaken in an effective and efficient way because users can access computational resources on demand (IaaS, Infrastructure as a Service) and use HPC (High Performance Computing) applications to obtain results in the shortest possible time (which is called HPC2).

The virtual machine deployed to test the performance of the processing genetic sequences work flow consists of 24GB of RAM memory, 16 cores and 3TB of storage, and allows an exome (from a Illumina HiSeq System) of 152 million *paired-end reads* of 91x91 *base pair* (that is, every read consist of two sequences 91 nucleotides long) to be processed in four and a half days (the test is described in detail in the subsection **3.1. Main work flow steps performance**, within the section **3. Pipeline performance**). That is one of the reasons why HPC solutions are ideal for processing genetic sequences.

With regards to information security, the LOPD[5] (Organic Spanish Law 15/1999 on the Protection of Personal Data), specifies that medical information must be

[5] http://www.boe.es/buscar/doc.php?id=BOE-A-1999-23750

treated according to the highest security level. However, since throughout the whole genetic sequences processing work flow personal data is not used (through dissociation), according to the Spanish Data Protection Agency[6] LOPD security measures do not need to be taken. Apart from this, high-level genetic results are secure thanks to the security of the Data Center in which the pipeline jobs are undertaken.

1.4 Motivation

When processing genetic sequences many software components have to be correctly assembled to obtain useful information to study hereditary diseases. The problem is that currently many users do not know how to perform those processing steps and they have to test whether the configuration of the tools is correct before they can use the final results. On the other hand, there are platforms that manage genetic sequences processing work flows, but their cost is often too high.

1.5 FI4VDI project

The main objective of the European FI4VDI[7] (Federation Infrastructure for Virtual Desktop Infrastructure) project is to develop a provision of Cloud Computing services to generate combined infrastructure destined for enterprises of all sizes (small, medium and large), universities and educational or research centers.

The Federation Infrastructure will consist in several OpenNebula zones [4] interconnected with a master one. Each slave zone represents a virtualization pool (managed by a certain hypervisor) from a technological centre which participates in the FI4VDI project.

In the last phases of the FI4VDI project several prototypes of Cloud Computing services related to user job will be proposed. One of the prototypes will be this pipeline, therefore users will be able to obtain high-level genetic results by deploying a virtual machine with an image containing the necessary pre-loaded data and the pipeline. VM resources will be configured on demand and will depend on the resources from the Data Center of every member of the FI4VDI federated network.

1.6 State of the art in resequencing software

In order to perform the steps that the computational phases of a resequencing experiment require, different applications have to be used, each one with a specific purpose. In this subsection the different kinds of programs available to do this task will be analysed:

– ***Reads* quality assessment**: their purpose is to analyse raw sequence quality and to perform certain treatments or filters.

[6] https://www.agpd.es/
[7] http://fi4vdi-sudoe.org/

- ***Reads* alignment**: this is the main step in the computational phases of the work flow. Many issues have to be dealt with, such as the alignment scope, the *reads* size of the program, the index data structures used by the software, etc.
- **Variant calling**: the objective of this task is to obtain variations within the aligned *reads*. The differences are brought about by the type of variation that the program looks for: SNPs, structural variations, chromosomal aberrations, etc.
- **Variant annotating**: this software mostly obtains the functionality of the variants analysed by connecting to public databases from genetics projects, but others do the same thing by accessing a local database.
- **Data visualization / filtering**: the objective is allow users to query genetic data easily but also to apply previous filters to all of the data generated.
- **Processing platforms**: several pipelines which help users to undertake all steps of a processing work flow have been implemented, but in all cases their license is proprietary.

1.7 Data from public databases

For the purpose of helping the tools to validate and to filter variations detected in the pipeline, databases from several genetic projects need to be downloaded or accessed. Variants within every one of the database files will have a certain weight when deciding whether detected variants are false-positives. The most important public databases are the following ones:

1. **1000Genomes** [5]: this project works on the relation between genotype and phenotype, researched by sequencing the genome of more than 1,000 people from different populations.
2. **OMIM** [6]: its main objective is to study Mendelian diseases.
3. **dbSNP** [7]: a public database of genetic variants.
4. **HapMap** [8]: provides information about haplotypes (a set of variants which tend to be inherited together).

In the following sections, the implementation and the performance of the pipeline will be described.

2 Pipeline implementation

2.1 Input data

The input pipeline data is a set of files containing *reads* generated by an Illumina platform. The pipeline has been developed to process the most typical *reads* generated by a NGS platform from that company: *paired-end reads* whose tags size is less than 100 *base pair* (about 6 GB per exome).

2.2 Computational phases of a resequencing experiment in the pipeline

In order to obtain high-level genetic information from the input data, a script which performs the computational phases of a resequencing experiment has been developed. The most important parameters for the internal tools can be specified in the pipeline parameters text file.

The second phase of a resequencing process is the alignment of the *reads*, with the **BWA-SW** tool [9] against an indexed and *consensus* reference sequence. The *assembly* (genome) used by the pipeline is the **hg19** (human genome version 19) and has been downloaded from the **UCSC**[8] (University of California Santa Cruz).

The result of the mapping is a **SAM** (Sequence Aligment/Map) file (the best tool to manipulate it is SamTools[9]) which contains data describing mainly where input *reads* fit with regards to the reference sequence. However, several treatments need to be completed before calling variants. The most important changes in the data are the ones which correct the information so that the variants detected in the last phase are as accurate as possible: marking duplicated alignments and correcting quality data within a mapping file are only some of the steps performed by useful tools like **PicardTools**[10] or **GATK** [10] (Genome Analysis Toolkit, very useful because of both the variety of tools and its users community).

The purpose of the third phase is to detect SNPs and small indels and to annotate them, two important tasks whose results determine the success of the *reads* processing because variants reported have to be correct and their annotations should describe their functionality. On the other hand, most of the false positives are filtered out by the execution of a GATK tool called **VQSR** (Variant Quality Recalibration Score). The most important issue is that it needs variations from at least thirty exomes, which have been downloaded from the **1000Genomes FTP** (its location is IBS, that is, Iberian population in Spain), and included in the image containing this pipeline to deploy VMs.

Annotations are assigned to detected variants with **VEP** (Variant Effect Predictor) [11] and with two scripts developed by the staff of the CénitS center by using **Ensembl v75** PERL **API** [12]

2.3 Configuration of the tools within the pipeline

With regards to the configuration of the tools themselves, there are some parameters which allow the pipeline to obtain the desired information regarding the genetic sequences:

– When starting the alignment process, a *reads group* has to be specified otherwise GATK will not be able to process BAM files (Binary Alignment/Map, the binary version of a SAM file).

[8] http://hgdownload.cse.ucsc.edu/downloads.html
[9] http://samtools.sourceforge.net/
[10] http://picard.sourceforge.net/

- The parameter *fixMisencodedQuals* (will have to be used in all the following work flow steps) is used in GATK in order to process *reads* whose nucleotide Phred scores (the quality of a *bp* (*base pair*) is measured with the Phred scale which is $-10 \times \log_{10}(P)$, where P is the probability of a *bp* to be erroneous) start at Q0 (ASCII 33). Q0 means that P is equal to one.
- To call variants with GATK, Haplotype Caller is used instead of Unified Genotyper, because is designed specifically for the human genome.
- In order to validate variations with the VQSR application, the first time the HaplotypeCaller is used it takes thirty BAM files containing thirty exomes, downloaded from the 1000Genomes FTP. A file with variations from those exomes will become the input for the VQSR application. On the other hand, when detecting variations in the following exomes, the result is appended with variants from previous processed exomes so that VQSR has enough data to validate the new variants.

With respect to *reads* generated by platforms from other companies, although the pipeline has been developed to process *reads* from Illumina, its internal tools could be configured to enable the processing of *reads* from different platforms. As an example, reads from Pacific Biosciences sequencers could be processed if **BWA-MEM** is used to map them. In addition, when calling and annotating variants, GATK should be configured to be more permissive of indels in the input data.

2.4 Output data

After finishing the computational phases of a resequencing experiment a set of annotated variants which will be stored in a database are generated (the data set size is approximately 200 MB). Below, the annotated variants fields are listed:

- Fields describing a variant: **chromosome**, **gene**, its **identifier** in the Ensembl database and a **description** of its functionality, **locus** (position where the sequence has changed), **reference** and **altered** alleles (alternative forms of the same gene or same genetic locus), **type** (SNP or small indel), **zygosity**[11], **rsID** (an identifier from dbSNP database) and a **phenotype** the variant is related to.
- Fields describing the effect of the variant in the transcription process (the conversion of exons into proteins): the identifier of the **transcript**[12] in the Ensembl database and the **exons** which participates in its generation, the **effect** of the variation, positions of the mRNA, of the genome and of the amino acid chain affected by a variant (**CDNA**, **CDS** and **aa change**, respectively) and the **amino acid** which will be changed in the transcription process because of the variant.

[11] The grade of similarity between the alleles within a certain locus of a gene from an organism. In a homozygous variation it is considered that both alleles are the same and that contains the variation whereas in a heterozygous one, an allele has not been affected

[12] A possible result of the transcription of a gene exons into mRNA (messenger RNA), as a consequence of changes that alternative splicing makes in it. A change in the transcription means that different proteins can be generated by the same gene

– Other fields which complement the information of a variant:
  - **GERP** (Genomic Evolutionary Rate Profiling) **score**: The grade of conservation of a genome locus in the evolutionary scope. The bigger the score is, the bigger the impact of a variation in that locus will be.
  - **HGVS** (Human Genome Variation Society) [13] ID: The identifier of a variation and a change in the transcription process according to the HGVS nomenclature.
  - **SIFT** (Sorting Intolerant From Tolerant) [14] and **Polyphen** [15] **predictions**: The pathogenic nature of the variation and its score, calculated by the SIFT and the Polyphen tools, respectively.
  - **Genotype**: Data describing a locus within a *read* where a variation has been called.
  - **PUBMED**[13] **ids**: The identifier of scientific articles from PUBMED associated with the variation.
  - **InterPro** [16] **ID** and its **description**: Describe the protein domain affected by a variation.

2.5 Variations database

In order to store variations (and their annotations), an entity relationship (**Fig. 1**) has been designed. The database is created only once and populated after *reads* from an exome are processed.

There are two types of entities: on the one hand the ones which store data related to the object their name refers to: **gene**, **variation**, **article** (entries from PUBMED database associated with variations), **exome**, **transcript**, **interpro** (the protein domain affected by a variation), **hgvs**, **exon**, **effect** (the change of a variation in the transcription process), **consequence** (a change in proteins generated by the transcription process), **prediction** (the predicted pathogenicity of the variation) and **phenotype**.

On the other hand, *n:m entities* are designed to enable the entities listed above to relate to each other.

2.6 Variants viewer

In order to query and to view annotated variants, many tools can be used (Genome Maps [17] is a good example) but together with the pipeline a viewer has been provided so that geneticists can compare variants detected within a set of exomes whose *reads* have processed, which is useful for genetic counseling[14]. The main features of the developed viewer are the following:

[13] A database which comprises more than 23 million citations for biomedical literature from MEDLINE, life science journals and online books http://www.ncbi.nlm.nih.gov/pubmed

[14] The process by which patients or relatives at risk of an inherited disorder are advised of the consequences and nature of it, the probability of developing or transmitting the disorder, and the options open to them in management and family planning

- A login interface to prevent users without credentials from accessing annotated variants.
- A main window in which geneticists can compare and query variants detected in exomes whose *reads* have been processed by the pipeline (**Fig. 2**), which is useful for genetic counseling.
- A series of filters with which a user can customize the variants whose annotations they want to query: the effect of the variant in the scope of the transcription process, the gene whose sequence may have been changed and the phenotype related to them (its description, the source of the data and its signification). **Fig. 3** shows an example of the application of two filters as well as the interface to filter variants by the phenotype they are related to.
- The possibility of consulting annotations of a certain variant (**Fig. 4**).

3 Pipeline performance

3.1 Main work flow steps performance

The table below contains the execution time of all the tasks the pipeline consists of, which are performed in every execution but the reference sequence indexing, which is made only the first time the pipeline is executed:

Resequencing phase	**Genetic processing step**	**Run time**
Second phase	Reference sequence indexing	2hr 15min
	Reads mapping against hg19	2hr 20min
	Sorting and indexing alignment results	28min
	Indel realignment	4hr
	Marking duplicated alignments	30min
	A recalibration of BAM results quality score	4hr 20min
	An alignment results file reduction	2hr 30min
Third phase	Variant Calling	19hr
	False variations deleting (validation and filtering)	1hr 10min
	Variant Effect Predictor execution	2d
	Annotation with PERL scripts	24hr
Total *Reads* processing execution time: 4d and a half		

Table 1. Pipeline tasks to call variants execution time

At the end, the whole amount of time that the pipeline takes to obtain annotated variants within an exome is about four and a half days. It is a poor performance but it could be improved if the main pipeline tools take advantage of the infrastructure below in order to benefit from parallelization.

With regards to mapping with BWA-SW, it has a good performance but could be improved by using parallel implementations like pBWA [18].

In the case of GATK, multi-threading can be configured when lunching jobs generated by its tools (each one of them has a recommended configuration) which

can be combined with a scatter-gather approach by using the GATK-Queue framework.

With regards to variants annotation, Variant Effect Predictor has parameters to specify the number of parallel threads which will be used but it can easily divide input datafile into chunks.

3.2 Database performance

Concerning the database performance, two issues have to be dealt with: the time to store annotated variants and the time need for database queries. First of all, storing annotated variants from an exome takes three days but the execution of the insertion program can take up to two weeks when data from five more exomes is stored, due to the amount of database operations performed by it. Regarding queries performance, when gene or effects filters are applied to variations from six exomes, the database efficiency is poor because users obtain filtered variants in sixty to ninety seconds.

In order to improve the database performance other mechanisms for storage and retrieval of data should be considered, like the implementation of a NoSQL database.

4 Future work

With regard to future work, several lines of work have been identified:

- Developing a graphical interface to allow users to configure the parameters of the pipeline.
- Designing a new variations database to improve its performance, for example based on NoSQL.
- Increasing the number of types of genetic sequences the pipeline can process, to allow geneticists to work with data from other companies.
- Incorporating more programs to process the pipeline tasks so that users can choose which software they want to use to obtain their results.
- Taking advantage of the parallelism implementations of the tools used by the pipeline, in order to improve the final performance.

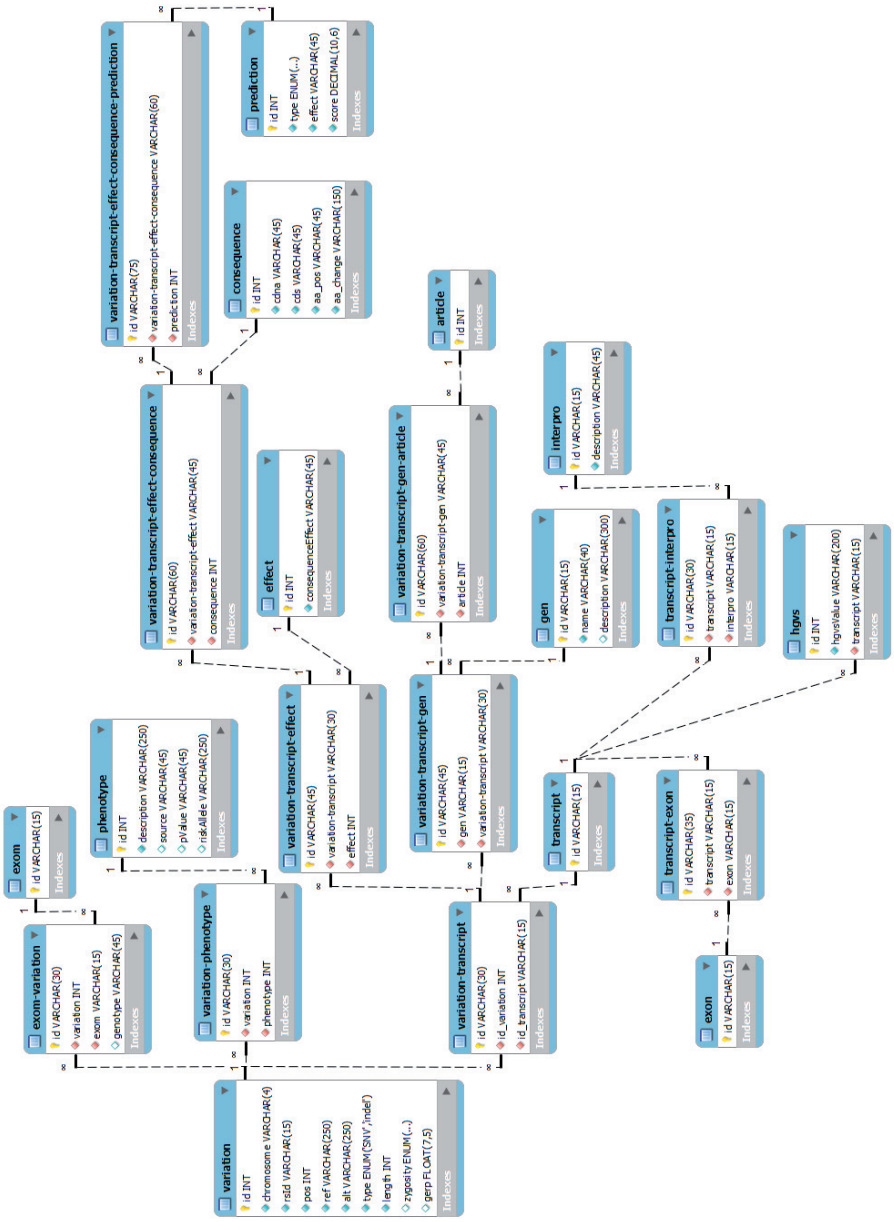


Fig. 1. Entity relationship of the variations database.

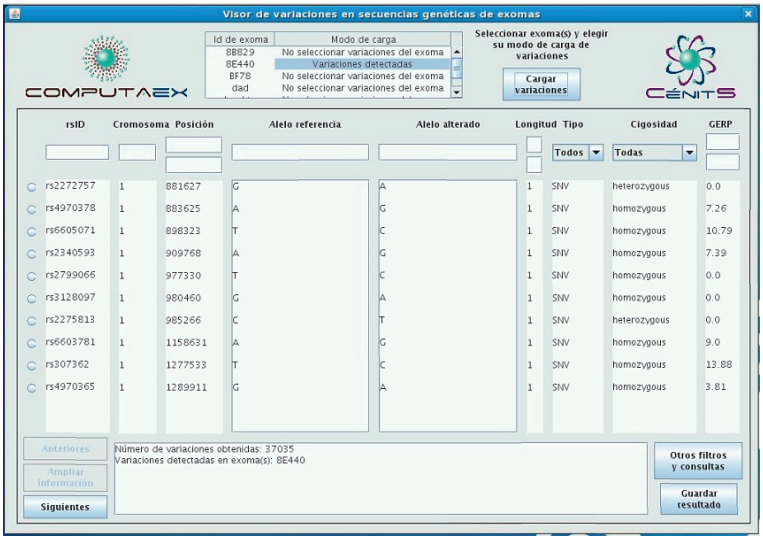


Fig. 2. The first ten variants detected in the *reads* of the exome 8E440.

5 Conclusions

A pipeline to process *reads* from exomes sequenced with Illumina platforms has been developed. The main advantage is that its execution allows geneticists to obtain and to view annotated variants automatically, without detailed knowledge of bioinformatics. However, the performance of the whole pipeline and above all the efficiency of the database needs to be improved so that genetic results for the study of hereditary diseases can be obtained in the shortest possible time. On the other hand, users can execute the pipeline in a Data Center infrastructure which has important benefits.

Acknowledgements

This work has been financed in part by the European Regional Development Fund through the project SOE4/P3/E804, FI4VDI, "Federation Infrastructure for Virtual Desktop Infrastructure" and charged against the registered grant exposed in the Extremadura Autonomous Community Regional Budget in 2014, budget heading 14.02.332A.444.00, super-project 2014.14.02.9007 "Actividades de I+DT en centros de investigación", project 2014.14.02.0010 "Financiación de la Fundación Computación y Tecnologías Avanzadas de Extremadura" (CENITAL-2).

We would like to thank the Genetics Unit and the Inmunology and Molecular Genetics Service in the Hospitals Infanta Cristina and San Pedro de Alcántara, respectively, and Manuel Corpas (http://manuelcorpas.com/), for providing us *reads* from six exomes in total.

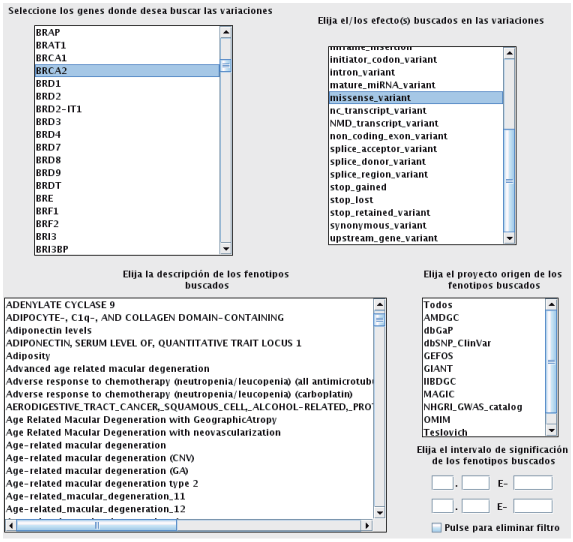


Fig. 3. An example of the application of two filters to obtain variants which change the sequence of the BRCA2 gene and an amino acid in the transcription. In addition, the figure shows the phenotype filters.

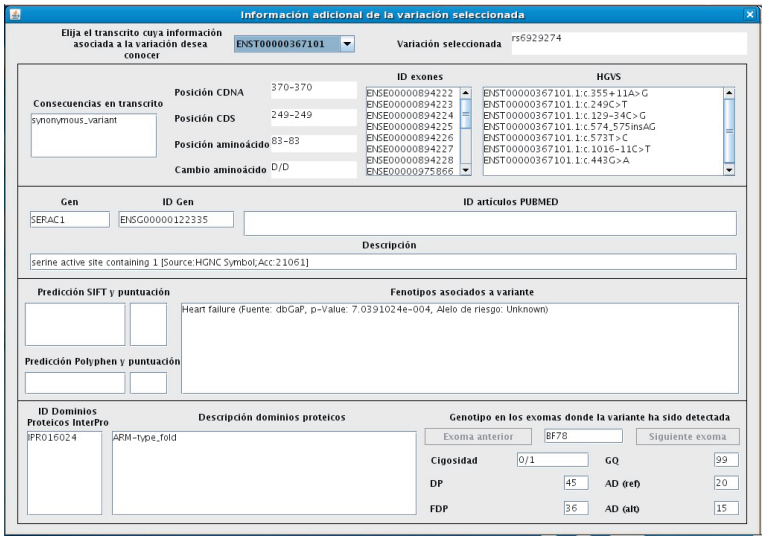


Fig. 4. Annotations of the variant rs6929274.

References

1. Ayman Grada, Kate Weinbrecht, *"Next-Generation Sequencing: Methodology and Application"*, Journal of Investigative Dermatology: 133, e11; doi:10.1038/jid.2013.248, 2013
2. Cock et Al, *"The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants"*, Nucleic Acids Research: doi:10.1093/nar/gkp1137, 2009
3. Danecek, Auton, Abecasis et Al, *"The variant call format and VCFtools"*, Bioinformatics, 27(15): 2156–2158, 2011
4. Moreno-Vozmediano, Montero, Llorente, *"IaaS Cloud Architecture: From Virtualized Datacenters to Federated Cloud Infrastructures"*, IEEE Computer: vol. 45, pp. 65-72, Dec, 2012
5. McVean et Al, *"An integrated map of genetic variation from 1,092 human genomes"*, Nature 491: 56–65, 2012, doi:10.1038/nature11632
6. McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University (Baltimore, MD), *"Online Mendelian Inheritance in Man, OMIM®"*, http://omim.org/
7. Sherry, Ward, Kholodov et Al, *"dbSNP: the NCBI database of genetic variation"*, Nucleic Acids Research: 1;29(1):308-11, 2001
8. The International HapMap Consortium, *"Integrating common and rare genetic variation in diverse human populations"*, Nature 467: 52-58, 2010
9. Li and Durbin, *"Fast and accurate long-read alignment with Burrows-Wheeler Transform"*, Bioinformatics: Epub. [PMID: 20080505], 2010
10. Van der Auwera, Carneiro, Hartl et Al, *"From FastQ Data to High-Confidence Variant Calls: The Genome Analysis Toolkit Best Practices Pipeline"*, Current Protocols in Bioinformatics, 43:11.10.1-11.10.33, 2013
11. McLaren, Pritchard, Rios et Al, *"Deriving the consequences of genomic variants with the Ensembl API and SNP Effect Predictor"*, Bioinformatics 26(16): 2069-70, 2010, doi:10.1093/bioinformatics/btq330
12. Flicek, Amode, Barrell et Al, *"Ensembl 2014"*, Nucleic Acids Research: 42, D749–D755, 2014
13. Johan T. den Dunnen, Stylianos E. Antonarakis, *"Mutation Nomenclature Extensions and Suggestions to Describe Complex Mutations: A Discussion"*, Human Mutation: 15:712, 2000
14. Kumar P, Henikoff S, Ng PC., *"Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm"*, Nat Protoc: 4(7):1073-81, 2009
15. Adzhubei, Schmidt, Peshkin et Al, *"A method and server for predicting damaging missense mutations"*, Nat Methods: 7(4):248-249, 2010
16. Hunter, Jones, Mitchell et Al, *"InterPro in 2011: new developments in the family and domain prediction database"*, Nucleic Acids Research, doi: 10.1093/nar/gkr948, 2011
17. Medina, Salavert, Sanchez et Al, *"Genome Maps, a new generation genome browser"*, Nucleic Acids Research, 41:W41-W46, 2013
18. Peters D, Luo X, Qiu K, Liang P, *"Speeding Up Large-Scale Next Generation Sequencing Data Analysis with pBWA"*. J Biocomput 1:1, 2012

IBERGRID Biomedical image analysis workshop

A large-scale graph processing system for medical imaging information based on DICOM-SR

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

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

Abstract. This paper presents Gpf4Med, a framework for the integration and analysis of medical reports, which aims at becoming a useful tool in healthcare research. Gpf4Med was designed to address big data analysis, considering the actual trends of exponential data growth and the explosion of data formats. Solutions are proposed from the point of view of the architecture design of the framework to face these problems. Preliminary studies are presented with very promising results for the applicability of the framework to the identification of non-evident correlations in the reports, while maintaining the desired levels of usability and performance that will allow radiologists and other possible users and stakeholders to use Gpf4Med in their daily work. Additional works are in progress to enhance the framework with more complex algorithms to discover clinically-relevant information in wide-disease analysis of large datasets of medical reports.

1 Introduction

Over the past two decades computing technologies have been used in clinical practice as a key tool to support diagnosis and treatment planning. Today, these tools would benefit from a better use of health data and the new advances in information processing. Despite the widespread use of computer-based systems for medical information acquisition, manipulation, transmission and storage (e.g. PACS [11], RIS [9], HIS [10]), there is much knowledge that is not exploited from these massive, heterogeneous datasets consisting of medical images and clinical data from thousands of patients treated at different centres.

This gap that exists between medical information storage and processing has motivated stakeholders within the healthcare industry to explore new options for managing medical data in order to enable the access to the new data processing technologies. Moreover, the availability of medical data is changing rapidly, challenging the sustainability of the actual medical databases and the storage and analysis services that they provide to the healthcare research community. One example is that the amount of medical images is growing exponentially worldwide and is estimated to represent 30% of the total data stored at present [3]. Having tools to effectively analyse this information may cause a major breakthrough in healthcare research.

However, there are additional challenges that need to be resolved before this information can be made available for research. The techniques for signal acquisition

are continuously improving, which leads to a wide spectrum of medical imaging modalities, such as X-ray, CT, MRI, ultrasound, PET, SPECT, etc. Moreover, there are many other forms of information that are produced in clinical practice and that researchers may find useful like unstructured reports, voice & video recordings, as well as many other data sources that provide supplementary information from simulations and experiments in models. Addressing this variety of information and data formats is a challenge for traditional database systems, which are designed to operate with a static data structure.

Also, the diversification of standards is a problem that affects many areas of research. In particular, there are several ontologies and terminologies like RadLex[1] and SNOMED-CT[2] that are commonly used to report the findings obtained in diagnostic studies. Each centre or department uses the standard that best suits its needs. Furthermore, it is a common practice to combine multiple standards in a single report to address the different aspects of the study. This variety causes problems of incompatibility between reports generated at different hospitals and even different departments of the same hospital. For example, the DICOM-SR specification provides a data format that leverages on the existing standards for representing and storing medical reports. This format allows hospitals to define their own templates for medical reports, although the templates from one hospital do not necessarily coincide with those from other hospital. Therefore, combining medical reports produced with different templates to find possible correlations among studies is a challenging task that requires a novel approach.

All these challenges are part of a wider challenge that arises from the increasing availability of data and the urgent need for new data processing techniques to address this unprecedented burst of information. Big data is the term coined to describe the technologies that have emerged in the last decade to tackle this challenge. While many of these technologies have been available to researchers for years, the adoption into healthcare industry has been slow. Facilitating the access of biomedical and healthcare researchers to big data analytic pipelines will improve their capabilities to acquire new knowledge, boosting new applications and services in the fields of personalised medicine and patient-centred, evidence-based care [8].

On the other hand, the focus of medicine has shifted in the last two decades 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 [5] [14] [6]. For these reasons, improving the use of health data is a major challenge in medical informatics and any advance in this field may contribute to improve clinical outcomes and to reduce the costs of patient care.

Currently, relational databases fail from representing multidimensional and temporal data in a way that new correlations can be discovered. Sequences of episodes and treatment plans give much more information that isolated episodes

[1] RadLex – a unified language of radiology terms: http://www.rsna.org/radlex/
[2] SNOMED-CT – Systematized Nomenclature of Medicine-Clinical Terms: http://www.ihtsdo.org/snomed-ct

with weak relations. Grouped visualization of patients, treatments and diagnoses can reveal undiscovered biomarkers and increase the repeatability of procedures.

This paper presents a novel approach for storing and querying medical reports using a graph database [1], which aims at identifying non-evident correlations that may occur between clinical features, radiological findings and treatment outcomes. The foundations of this approach has been established in the recent years with the popularization of social networks and the advent of new computational tools to mine the relevant relationships from these networks. The progress made in this field has rapidly gained attention from healthcare researchers who have found similar patterns in the analysis of radiology images and reports [12]. A critical obstacle to the applicability of graph database models to the study of medical reports is the lack of appropriate analysis tools and data. Despite the availability of open-source graph database systems, such as Neo4j[3], PACS and RIS are currently the most commonly used systems for storing raw data at the hospitals and they do not provide support for this model. Therefore, the first objective of this paper is to allow storing medical reports out of the storage system of the hospital, using a graph database. The second objective is to integrate all the available information in a single database to facilitate a disease-wide analysis that includes patients and reports from a large-scale, multicentre dataset, which is the third objective.

These objectives are addressed in this paper through the development of a modular framework for the integration and study of clinical data collected at different centres using multiple exploration methods. The data is combined in a graph and the different medical terminologies and ontologies annotations are used to interconnect the objects represented in the graph (e.g. patient, modality). Other types of information can be integrated, such as genetic, histopathological, biomolecular and pharmacological data.

The framework consists of a graph store and a search engine that allows users to query the graphs. The framework was designed to support medical standards to facilitate interoperability with other systems. Along with the efficient indexing of large, heterogeneous health datasets in a graph store, the other main goal of the framework is to study large datasets using Cloud computing, which could provide distributed computing resources to address the analysis of large graphs. This paper aims at supporting queries that cannot be addressed solely using the traditional storage and computing systems of the hospital.

The rest of the paper is structured as follows: Section 2 presents two relevant cases where the application of graph analysis appears promising based on a preliminary study. Section 3 describes the approaches that were used to design and develop the framework. Section 4 presents a pilot implementation of the framework that is currently being evaluated by developers with the help of radiologists and healthcare researchers. Finally, Section 5 presents concluding remarks and future lines of work.

[3] Neo4j – an open-source native graph database: http://www.neo4j.org/

2 Applicability of graph analysis to the study of medical reports

There are many examples in the literature that demonstrate the usefulness of reporting clinical data (e.g. diagnostic studies, treatment outcomes) using medical ontologies and terminologies. This approach facilitates the automation of clinical processes to assist radiologists and other medical specialists in their daily work. Compared to free-text reports, the use of terminologies allows developers to focus their efforts in developing new algorithms for analysing the data instead of dealing with the interpretation of plain text annotations, resolving ambiguities and other problems that arise in the processing of plain-text. DICOM and DICOM-SR provide a widely used format for storing and annotating medical images and reports. In particular, DICOM-SR sets the bases for the definition of structured report templates.

This capability of DICOM-SR to define templates is especially useful for developing new graph-based analysis tools. However, graph databases are just one of the many possible ways to store clinical data and to make them available in a usable form for data analysis. Other types of NoSQL data models include key-value, column family and document-oriented databases [7]. The reason for choosing the graph approach is that previous studies have demonstrated the suitability of this model to analyse complex relationships between objects and to find patterns within network structures. These features can be applied to the study of DICOM-SR reports stored in a graph database.

Finding nodes that have similar neighbouring nodes and finding the shortest path between two nodes are two examples of operations that are known to perform well in graph databases [7]. Having a quality DICOM-SR graph database is the first step towards the development of new algorithms that use this approach to identify patients with similar background and to optimize the clinical pathways, reducing the number of radiation treatments and the total dose of radiation per patient.

Moreover, graph algorithms have the additional advantage of not requiring a specific infrastructure, which is the case of other NoSQL-based data processing algorithms like map-reduce [4] that requires a highly specialized environment that includes the Hadoop Distributed File System (HDFS).

The following subsections describes two scenarios where the applicability of graph algorithms appears promising with potential for the development of new clinical applications. These scenarios are: 1) management of radiation dose in medical imaging procedures; and 2) clinical pathway for the management of breast cancer.

2.1 Management of radiation dose in medical imaging

Understanding radiation dose is necessary to make informed decisions regarding imaging modalities. Moreover, image quality and patient safety are closely related to the calculation and application of the optimal dose needed to balance among them. However, studying radiation dose requires the analysis of various types of information that are stored in different systems using a variety of data formats.

This variety is caused by the use of different image modalities that complement each other, contributing with additional information to the study. Beyond the differences among image modalities, the acquisition protocols vary from one device to another, as well as the data and the report formats they use. The largest variations appear when the patient is not treated in a single hospital, which is a common practice nowadays due to the high specialization of hospitals.

Improving the actual estimation of radiation dose will contribute to reduce the induced morbidity, reducing healthcare costs, but also will contribute to the prevention of occupational risks.

The DICOM Dose Structured Report (DICOM Dose SR) is a standard format for storing information about radiation dose that is implemented in the majority of the image acquisition devices. This format allows those devices to record all of the radiation used during a procedure, reporting the duration of the exposure to the radiation emitted by the device, as well as other parameters that influence the radiation dose. This information is supplementary to the DICOM headers of the image stored in the PACS of the hospital and complements the MPPS (Modality Performed Procedure Step) messages that the device records in the RIS of the hospital to report the performed exams. A particular feature of this process is that the images that are discarded from the study (which are not recorded to the PACS) also contribute to the dose and this information is only available in the DICOM Dose SR report.

Currently, there are 14 standards that define DICOM Dose SR templates for different image modalities[4]. Besides the differences between the formats used by the image modalities, there are other factors that make the analysis of radiation dose reports a complex problem. For example, the standards defines the CT Radiation Dose Reporting, which is the format supported in the majority of the modern CT devices. However, often manufacturers include their own features, units and customizations of the standard specification. This causes that each report needs to be preprocessed before it can be compared to other report, especially if the reports were produced with different devices or even different versions of the same device.

2.2 Clinical pathway for the management of breast cancer

BI-RADS [5] is the widely adopted standard for classification of breast cancer findings. BI-RADS provides a terminology that is used in all the imaging modalities that are relevant for the diagnosis of this disease: mammography, ultrasound and RMI. In addition, the RadLex ontology includes BI-RADS as one of the available terminologies. Despite this, the format of the medical reports on breast cancer is open to variable interpretation, which causes that every hospital implements its own format including the terminologies (in addition to BI-RADS) that best suit to its requirements. This causes a wide variability in the reports produced in different hospitals.

[4] DICOM Standard Status – http://www.dclunie.com/dicom-status/status.html

[5] BI-RADS – Breast Imaging-Reporting and Data System: http://www.acr.org/Quality-Safety/Resources/BIRADS

On the other hand, there exists many evidences that support the influence of external factors that are not related to the radiological study, but to clinical exploration or background risks, which may affect the outcome of the treatment. The study of these factors jointly with the radiological reports may provide additional information about the risks of developing breast cancer [13]. Also, the combined analysis of radiological reports with clinical and epidemiological information, as well as with expert opinion, may contribute to increase the accuracy of diagnosis [2].

3 Architecture

Figure 1 presents a general overview of the architecture of the framework. The `DICOM Graph Study Service` is the main entry point to the system. This service is provided as a RESTful web service and also as a Java-based application programming interface (API), which is intended for advanced use. In its most simple form (no loaded plugins) the REST interface exposes only five basic operations: list available DICOM SR templates & graph plugins, create & destroy graphs in the graph store and retrieve the latest statistics from the service. New resources (and operations) can be added to REST interface by loading new plugins into the service. Similarly, to add a new type of study to the interface the corresponding DICOM SR template is loaded into the service, as for example, a new medical imaging modality.

To use the graph store, clients must perform the following operations on the `DICOM Graph Study Service`: 1) select the type of graph from the list of available graph plugins; 2) submit a collection of references to the dataset that can be read with the available DICOM SR templates; 3) create the graph; and 4) use the operations provided by the plugin to submit queries to the graph.

The `Document Fetcher` fetches the corresponding file from the `DICOM Store` for each report included in the dataset. The file is then parsed, validated with the appropriate DICOM SR template and loaded in the graph. Before the files can be fetched, valid access credentials must be made available to the `DICOM Graph Study Service` for those `DICOM Stores` that implement secure access. This is normally done during the configuration of the service.

Files fetched from the `DICOM Stores` are cached in the local file-system of the `DICOM Graph Study Service` for future reuse. Encryption is provided as a configuration option of the `DICOM Graph Study Service`, in which case decryption keys are maintained in memory, avoiding that cached files can be read after a service restart. However, to preserve the security and privacy of the information, only anonymized and pseudo-anonymized reports should be used.

The `DICOM Graph Study Service` was designed as a very simple set of queues and event responders that consume the least amount of computing resources. The graph store is by defect an in-memory database that stores one single graph, which is the graph under study. This design responds to the need of making the study of large datasets practical, and ensures that the `DICOM Graph Study Service`

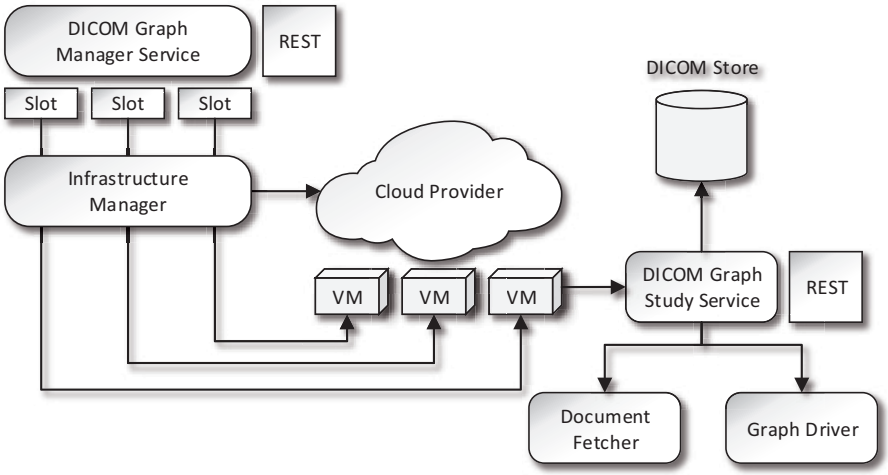


Fig. 1. Architecture of the graph processing framework for medical information.

can be used with common Cloud infrastructure providers, such as Amazon EC2[6] and OpenStack[7]. To ensure that the `DICOM Graph Study Service` is cloud-ready, implementations must at a minimum support the following requirements:

- Provide a property-based configuration mechanism to manage access to the DICOM SR templates and the graph plugins, and to specify the network port where the service listens for incoming requests.
- Support secure access to DICOM stores and encryption of local files.
- Provide a single package that contains all the necessary dependencies, configuration and documentation.
- Support a simplified deployment of the system that consists only of two steps: 1) adjust configuration; and 2) run the HTTP server.

The `DICOM Graph Manager Service` is a high-level component of the framework that implements a *split-process-merge* schema to study large datasets. The RESTful web service interface of this service provides the same operations as the `DICOM Graph Study Service`, but internally, the `DICOM Graph Manager Service` manages a pool of slots that can be used to reduce the size of the specified study. Each slot provides access to a different instance of the `DICOM Graph Study Service`. The `DICOM Graph Manager Service` is connected to an `Infrastructure Manager` that allocates the necessary computing resources (e.g. virtual machines) to deploy the `DICOM Graph Study Services` in a Cloud computing environment.

Figure 2 shows the pipeline of a large-scale study conducted with the framework. First, the `DICOM Graph Manager Service` inspects the dataset (represented

[6] Amazon EC2 – Amazon Elastic Compute Cloud: http://aws.amazon.com/ec2/
[7] OpenStack – Open Source Cloud Computing Software: http://www.openstack.org/

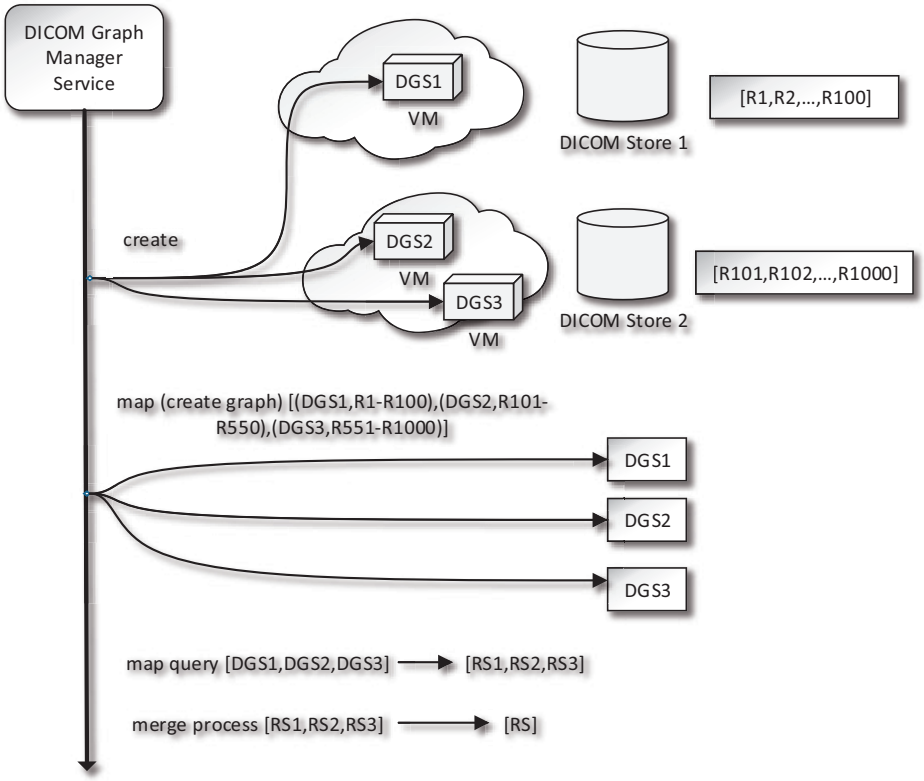


Fig. 2. Pipeline of a large-scale study.

in Figure 2 by $[R1, ..., RN]$) to discover where the report files are stored and uses the `Infrastructure Manager` to create an appropriate number of `DICOM Graph Study Services` (represented in Figure 2 by $DGS1$, $DGS2$ and $DGS3$) to accommodate the graph store. Where possible, the new virtual machines for the $DGSs$ will be created in the same administrative domain as the `DICOM Store` where the dataset is stored. The computation of the necessary number of virtual machines to accommodate the dataset in the graph store takes into account the specific requirements of the type of graph (e.g. RAM memory). Therefore, changing the type of graph used in a specific study could lead to a reconfiguration of the back-end, and in the worst case to the creation of new virtual machines and the redistribution of the dataset over the $DGSs$. Finally, after the back-end is created for the study, the `DICOM Graph Manager Service` distributes the operations and queries over the $DGSs$ using a *split-process-merge* schema (represented in Figure 2 in a functional programming idiom). In the last step of the pipeline, the result sets (represented in Figure 2 by $[RS1, ..., RSN]$) obtained in the individual $DGSs$ are merged before being sent to the client. During the merge, an optional process can be applied to the result sets where they are still in the $DGSs$, as for example a filter.

3.1 Implementation and available graph types

The Graph Processing Framework for Medical Information (Gpf4Med) provides an implementation of the framework described in this paper. Gpf4Med is mostly implemented in the Java programming language, and is distributed with an open-source licence[8].

Neo4j[9] was used as the NoSQL database technology that supports the graph store in the `DICOM Graph Study Service`. This service is distributed with the HTTP server provided by the project Grizzly[10], which provides HTTP/HTTPS-based access to the REST interface.

Apache jclouds[11] was used to provide `DICOM Graph Manager Service` with access to a variety of Cloud infrastructure providers.

The current version of Gpf4Med is distributed with a set of graph plugins for breast cancer, which provides access to an environment for the study of clinical data collected on patients of this disease. These plugins are currently under evaluation with the objective of finding the most relevant operations to detect non-evident correlations in the reports. Once these operations are identified, the next step will be to develop pipelines to facilitate the use of the framework, as well as developing a friendly user interface.

[8] Gpf4Med – Graph Processing Framework for Medical Information: https://sourceforge.net/projects/gpf4med/
[9] Neo4j – an open-source graph database: http://www.neo4j.org/
[10] Grizzly – NIO framework: https://grizzly.java.net/
[11] Apache jclouds – an open-source library to Cloud: http://jclouds.apache.org/

Visualization capabilities are currently limited to static views of the graph produced with Graphviz[12]. These views are stored in the Graphviz format and they are made available for download through the REST interface.

4 Preliminary results

During this initial phase, development efforts were focused on the design of the graph database and the architecture of the Gpf4Med framework. Virtualization was considered critical for the installability of the framework with the minimum number of manual steps. This design allowed us to deploy a virtual infrastructure to support the framework using storage and computing resources provided by Amazon EC2, as well as a second virtual infrastructure where an on-premises cloud deployment based on OpenNebula was used. Also, the modular design of the framework allowed us adding and removing instances of the `DICOM Graph Study Services` from the virtual infrastructure, which will be useful in production deployments to adapt the framework to the workload.

Loading data in the graph from the `DICOM Store` instances was one the most challenging activities faced in this phase. Initially, the approach used was to represent as many relationships as possible in the graph, which causes a massive creation of edges in the graph. However, after the first round of refinements a large number of these relationships where discarded since they were found clinically irrelevant. This led to the first optimization of the database, which in the current version is focussed on the representation of the elements that appear promising for the analysis that will be performed in the next phase of the development.

Figure 3 shows a simplified graph of the clinical reports in breast cancer, where the nodes and relationships that were found clinically irrelevant for the study have been omitted. This graph represents the information using the same nomenclature and terminology that radiologists use in their daily work (i.e. patient, breast, lesion, finding), in order to reduce the complexity of the graph, which aims at reducing the learning curve. Graph nodes are annotated with information extracted from the radiological reports, such as type, location and morphology of the lesion, presence of relevant features like calcifications and other relevant information. BI-RADS classification is represented in the graph as a node that provides many connections to other studies that use different imaging modalities or even studies from other patients or related to other processes in the clinical management of breast cancer, such as follow-up and response to treatment episodes. This information is natively represented by the edges of the graph, which also represents the relationships between different fields of the report. However, due to many relationships have been pruned from the graph, reconstruction of the original report is impossible without additional information. This design has one major consequence for the applicability of the framework: the graph database is not a substitute for PACS, RIS or other hospital systems that store raw data. However, this was never an objective of the framework.

[12] Graphviz – open source graph visualization software: http://www.graphviz.org/

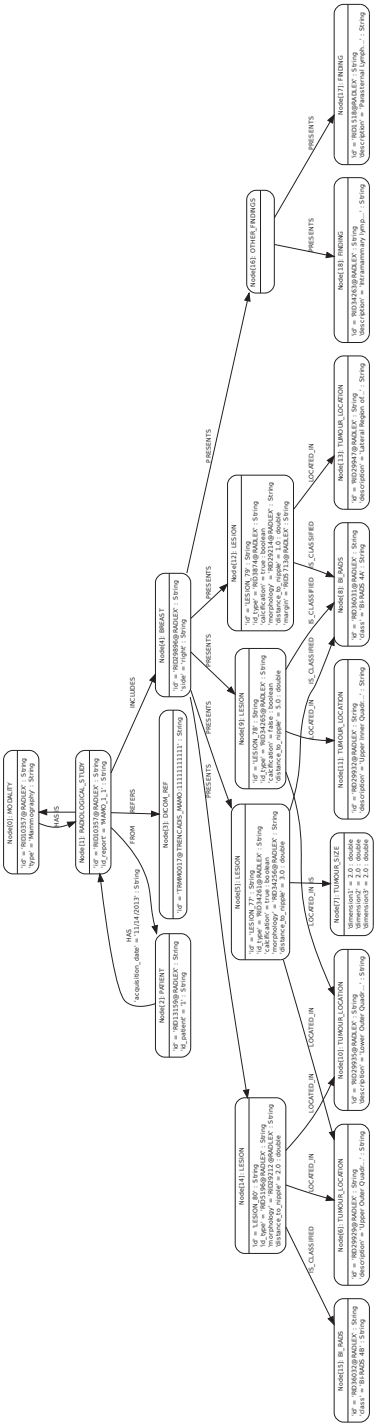


Fig. 3. Simplified graph of the clinical reports in breast cancer.

Preliminary tests have showed that this graph supports loading a massive amount of reports (in the order of hundreds of thousands), which will allow us to address wide-disease analysis in the future. With regard to graph operations, very simple test were performed with this graph and the results showed that the performance of queries is also promising with response times in the order of a few seconds. More complex tests that involves several operations with the graph also exhibit promising results. For example, the graph was queried to find how many palpable lesions in category 4 were confirmed as BI-RADS 2 (benign –non-cancerous– finding) in a later biopsy and how many were confirmed as BI-RADS 6 (known biopsy-proven malignancy). This type of queries requires the use of more sophisticated algorithms that will be developed in the next phase of development.

5 Conclusions

This paper presents Gpf4Med, a framework for the integration and analysis of medical reports, which aims at becoming a useful tool in healthcare research. Gpf4Med was designed to address big data analysis, considering the actual trends of exponential data growth and the explosion of data formats. Solutions were proposed from the point of view of the architecture design of the framework to face these problems.

Preliminary tests with Gpf4Med were performed with very promising results for the applicability of the framework to the identification of non-evident correlations in the reports, while maintaining the desired levels of usability and performance that will allow radiologists and other possible users and stakeholders to use the framework in their daily work.

Future lines of work includes the compilation of a validation dataset from real cases, as well as completing the development of the framework. In particular, it is expected that the next releases will include new algorithms to discover clinically-relevant information in wide-disease analysis of large datasets of medical reports.

Acknowledgements

The authors wish to thank to the members of the radiology unit of the Hospital Universitario y Politécnico La Fe for advising and giving us support in the identification and validation of clinically-relevant use cases for the Gpf4Med framework.

References

1. R. Angles and C. Gutierrez. Survey of graph database models. *ACM Computing Surveys*, 40(1):1–39, Feb. 2008.
2. J. Chhatwal, O. Alagoz, M. J. Lindstrom, C. E. Kahn, K. A. Shaffer, and E. S. Burnside. A logistic regression model based on the national mammography database format to aid breast cancer diagnosis. *AJR. American journal of roentgenology*, 192(4):1117–27, Apr. 2009.
3. E. Comission. Riding the Wave European Commission report on Scientific Data.

4. J. Dean and S. Ghemawat. MapReduce. *Communications of the ACM*, 51(1):107, Jan. 2008.
5. G. S. Ginsburg and J. J. McCarthy. Personalized medicine: revolutionizing drug discovery and patient care. *Trends in biotechnology*, 19(12):491–6, Dec. 2001.
6. 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):301–304, 2010.
7. D. McCreary and A. Kelly. *Making Sense of NoSQL: A guide for managers and the rest of us*. Manning Publications Co., Shelter Island, NY, 2013.
8. T. B. Murdoch and A. S. Detsky. The inevitable application of big data to health care. *JAMA : the journal of the American Medical Association*, 309(13):1351–2, Apr. 2013.
9. J. Oakley. *Digital Imaging: A Primer for Radiographers, Radiologists and Health Care Professionals*. Cambridge University Press, 2003.
10. H. U. Prokosch and J. Dudeck. *Hospital Information Systems: Design and Development Characteristics; Impact and Future Architecture, 1e (International Congress Series)*. Elsevier, 1995.
11. O. Ratib. Imaging informatics: from image management to image navigation. *Yearbook of medical informatics*, pages 167–72, Jan. 2009.
12. O. Ratib, A. Rosset, and J. Heuberger. Open Source software and social networks: disruptive alternatives for medical imaging. *European journal of radiology*, 78(2):259–65, May 2011.
13. J. H. Shin, B.-K. Han, E. Y. Ko, Y. H. Choe, and S.-J. Nam. Probably benign breast masses diagnosed by sonography: is there a difference in the cancer rate according to palpability? *AJR. American journal of roentgenology*, 192(4):W187–91, Apr. 2009.
14. A. D. Weston and L. Hood. Systems biology, proteomics, and the future of health care: toward predictive, preventative, and personalized medicine. *Journal of proteome research*, 3(2):179–96, 2004.

The ARTFIBio Web Platform

J.C. Mouriño**[1], A. Gómez[1], M. Sánchez[1], and A. López-Medina[2]

[1] Fundación Pública Galega Centro Tecnológico de Supercomputación de Galicia (CESGA), Santiago de Compostela, Spain
[2] Medical Physics Department, Complexo Hospitalario Universitario de Vigo (CHUVI), Spain

Abstract. In the last years, there were big steps in the treatment of head and neck cancer (HNC), but local relapse rates are still very high. For this reason, multicentric research is needed to overcome them, focused on quantifying tumor response of HNC patients by functional images like PET/CT and MRI. The joint use of MRI and PET/CT can be useful for delimiting the hypoxic areas. To leverage the collaboration among hospitals and researchers, a platform architecture specifically designed to suit the needs of the study of tumor response quantification is being developed. It allows the sharing of the patient's images, the radiotherapy treatment planning, and the information of the final delivered doses. Other additional information can be added for each patient, as chemotherapy treatment data or the surgical procedures, so a full set of information can be gathered for each treatment. Additionally, the platform includes a registration tool to analyze correctly the different patient's images. Our tool is being successfully used in the frame of the ARTFIBio project, focused on the study of predictive individualized models of head and neck tumor response to radiotherapy.

1 Introduction

Head and neck cancer (HNC) is the fifth tumor with the highest incidence in the world population and, despite the great improvements in treatment achieved in recent years, its local relapse rates are still higher than those of other malignant pathologies. The ultimate goal of our multidisciplinary research, framed in the ARTFIBio project (Adaptive radiotherapy and prediction of the tumor response based on functional studies of MR and PET/CT in head and neck cancer), is to create an information network to develop predictive individualized models of the tumor response to radiotherapy, able to define more effective adaptive treatments [**?**].

Radiotherapy (RT) is changing from image guided to biology guided [**?**]. In the next years, the daily clinical practice will be modified passing from prescribing doses to determined volumes with anatomic information to optimize the treatment in order to minimize the tumor cells surviving the treatment and with acceptable secondary effects.

** jmourino@cesga.es

For this purpose, instead of the currently established methods based on `in vitro` data and animal experimentation, `in vivo` measurements from real patients are used. Functional studies from PET/CT (Positron Emission Tomography) and MRI (Magnetic Resonance Imaging) obtained along treatment (before, during and after), are the most promising techniques to measure tumor response, since these imaging modalities provide crucial and complementary information (like malignancy, oxygenation or tumor density) to analyze the treatment evolution.

Bearing in mind that we need to extract spatially coherent information from multiple imaging modalities (CT, PET/CT, and different MR techniques) that are obtained at different stages of treatment, a co-registration scheme among the different images has been also developed. Hence, a web platform has been developed fully adapted to the specific requirements of the project.

Taking it into account, the use of all these image modalities is considered the ideal starting point for the simulation of the RT patients [**?**][**?**], although it is not yet available in most of the centers for their use in a daily basis. The use of PET/CT and MR in the delimitation of tumoral volumes represents an evident clinical improvement and is gaining acceptance in daily clinical practice.

Given the means to share information, it is customary in research to have repositories of objects in order to share and make easier the access to information. In the case of medical images there are several image repositories, some of them are specialized in a particular type while others are generalist. As an example, in the United States there are numerous repositories for data sharing, some of them are listed in [**?**]. From these repositories we may outline the Cancer Imaging Archive [**?**], which keeps sets of images related to cancer research. There are also other repositories that provide access to the specific DICOM objects for radiotherapy, like the deployed in Denmark [**?**] or the NROR [**?**] under development in the United States, with a pilot over prostate cancer. In Spain there are other repositories of DICOM objects like TRENCADIS [**?**], developed by a group of the Polytechnic University of Valencia directed by Ignacio Blanquer. This repository is currently under deployment stage in the health services environment of the Valencian Community and it allows sharing DICOM objects. The PIC (Puerto de Informacin Cientfica - Scientific Information Port) keeps also an storage system of medical images for research [**?**] which is employment as an use case in Cloud environments of the Helix-Nebula project. The main difference among this type of sharing platforms and ARTFIBio is that in addition to the access to information of images (that in the ARTFIBio case it includes several modalities like CT, PET/CT, and MRI, all of them related to the study case) and to information about the treatment (like chemotherapy or surgery), it allows the integration with desktop applications for the analysis of the stored information.

The structure of this paper is as follows. In Section 2 the ARTFIBio project is presented. Then, in Section 3 we describe the developed web platform. In section 4, the connection with the visualization and registry tool is explained. Finally, Section 5 concludes the paper.

2 The ARTFIBio Project

The ARTFIBio project is a collaboration project, funded by the Spanish Government, developed by CESGA, Complexo Hospitalario de Vigo and the University of Vigo. The aim of this project is to create an information network to develop predictive individualized models of the tumor response to radiotherapy in patients with head and neck cancer based on `in vivo` functional data. For this purpose, several studies of MRI and PET/CT are performed, and biopsy is analyzed. Quantification of tumor cell density and oxygenation are calculated using image data and biopsied tissue. Functional images are obtained from pre- and post-treatment PET/CT, and from a serial 4 MRI studies (perfusion- and diffusion-weighted), which can be used to assess tumor response and oxygenation. `In vivo` measurements from real patients will allow us to develop a predictive models much better than the currently established based on `in vitro` data and in animal experimentation.

The global goals of this coordinated project are:

1. Creating a web infrastructure for research in radiotherapy and other oncology specializations that allows the evaluation of the tumoral response to several treatments based on functional images and other data.
2. Developing all the tools that make possible the evaluation of the tumoral response in head and neck cancer of squamous cells based on functional images, dosimetry, incidents during the treatment and biopsies, with a predictive analytic model that considers the initial state and the expected treatment and an adaptive model that considers the early tumoral response to the treatment.

To allow the collaboration among the researchers and between several hospitals in the near future, CESGA is developing a web platform with several tools: treatment data, anonymized image and treatment planning database, non-rigid co-registration of several image modalities, and the difference between delivered dose and planned dose. Other tools will be included in the future as visualization of tumor cell density as treatment is delivered, clinical prediction of the tumor response, evaluation of the treatment considering usual setbacks (breakdowns, non delivered beams,...), and variation between the predicted tumor cell density and the quantified one using control images performed three months after treatment.

The availability of this information will permit the development and integration of new modules that are under development in the project as quantification tools of tumoral density coming from functional images and additional data from the biopsy; of oxygenation of a tumor, and interfaces to evaluate graphically the evolution of a tumoral response.

This web application will be the basis for the integration of data from several hospitals that have experience in the treatment of oncological patients, allowing the development of new treatment response predictive models from functional images and other information.

3 Web platform

The ARTFIBio web application[3] is able to analyze, process and storage the following DICOM objects (many of them images):

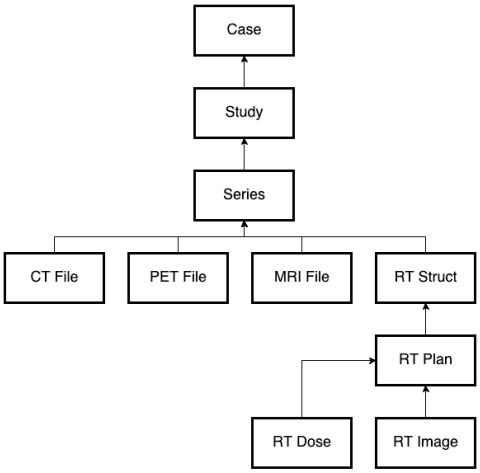


Fig. 1. Hierarchy of the DICOM files

- Magnetic Resonance Imaging (**MRI**): diffusion and perfusion studies, including T1, T2, Apparent Diffusion Coefficient (ADC) and Dynamic Contrast Enhanced Magnetic Resonance Imaging (DCEMRI).
- Positron Emission Tomography (**PET**).
- X-Ray Computed Tomography (**CT**): including both the CT in the PET/CT series, and the CT simulation for radiotherapy treatment
- **DICOM-RT**: Support for specic DICOM objects for radiotherapy, spatially co-registered with the CT simulation:
  - **RTSTRUCT**: File with structure sets of different regions of interest of the image (external body, organs, planning target volume (PTV), clinical target volume (CTV), gross tumor volume (GTV), etc., stored as different series of contour points. This information is used to calculate the treatment.
  - **RTPLAN**: Geometric and dosimetric data relating to treatment plan. External Beam: treatment unit, isocenter, gantry/couch/collimator angles, collimation.
  - **RTDOSE**: 3D Image with the spatial dose distribution calculated by the radiotherapy treatment planning system of the hospital.
  - **RTIMAGE**: Images relevant to RT acquired using conical geometry. Generated images: DRR, DCR, BEV, etc. and other RT images: Simulator, portal images, etc.

[3] http://artfibio.cesga.es/Artfibio/application

In general, any study from a single patient involved in the ARTFIBio project will consist of: two PET/CT studies (pre and post-treatment), one CT simulation, several MRI studies (before, during and after the treatment) of different modalities, one RTSTRUCT object with the regions of interest marked over the CT simulation, one RTPLAN, and one RTDOSE. The DICOM hierarchy is shown in Figure **??**.

The web platform runs in the Apache Tomcat 7 [**?**] application server using Oracle MySQL 5 [**?**] as database management system and Vaadin 6 [**?**] as Java web development framework.

An Entity-Relationship diagram of the design of the database is detailed in Figure **??**.

A typical use case of the web application includes the selection and upload of a set of DICOM objects related to a treatment, the preview and completion of the missing metadata, and the storage of the files in the system.

It is very important to remark that all personal data of the patient is deleted using a Java applet executed locally in the web browser before every DICOM file is uploaded to the system thus they can remain anonymous. This applet permits the multiple selection of DICOM files and uploads them compressed to the server. The server side reads the headers in order to identify the type of file, sorts and groups them in a tree by study and series before showing a preview to the user.

Once the files have been uploaded and before they are saved in the database, the user has the chance to check if they are the correct ones and to edit the metadata from the file header that is going to be stored in the database together with the image. In Figure **??** we can see on the left the tree generated during the upload process. In this tree, the user can select a file in order to preview its content on the right form. The right panel also shows the editable metadata, which depends on the DICOM object type.

After the DICOM files have been reviewed and saved, the software is able to present the DICOM series as a monthly schedule as it is shown in Figure **??**. The top of the screen is occupied by the monthly calendar that shows the temporal distribution of the DICOM objects during the treatment of the case. The bottom of the screen presents a similar DICOM objects preview layout as the one that was explained previously. This time the metadata is obtained from the database of the application instead of directly from the DICOM file as before, but it allows editing those fields as well.

4 Access to Registry and visualization tool

As commented before, a Visualization and co-registration tool has been developed by the University of Vigo. It is not the aim of this paper to describe this tool. Details can be found at [**?**]. The tool has been developed in C++, using ITK, GDCM, VTK and Qt libraries, and can be installed and executed directly in the computers inside the hospital facilities, using Microsoft Windows. Besides, to permit a smooth integration in the platform, the application has been recompiled and installed in a Linux based virtual machine allocated in CESGA IaaS Cloud. The application is integrated in the web browser of the client using three modules in this virtual machine: one VNC server, one noVNC server to allow the connection

Fig. 2. ER model

of noVNC client from the browser, and the xinetd module to start both of them plus the application.

Virtual Network Computing (VNC) is a remote display system which permits to view and interact with a virtual desktop environment that is running on another computer on the network. Using VNC, graphical applications can run on a remote machine and send only the display from these applications to a local desktop. VNC is platform-independent and supports a wide variety of operating systems and architectures as both servers and clients. In this project, TigerVNC [**?**] version

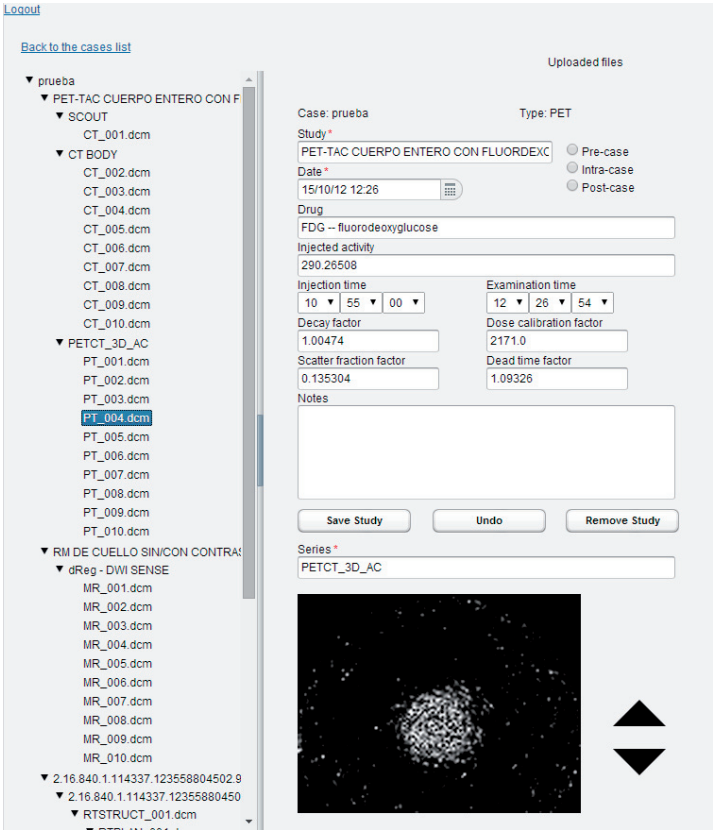


Fig. 3. DICOM Preview

1.2.0 has been used as server and noVNC by kanaka [**?**] version 0.4-139-g98c1275 as client.

TigerVNC is a high-speed version of VNC based on the RealVNC 4 and X.org code bases. noVNC is a VNC client using HTML5 (WebSockets, Canvas) with encryption support that runs in any modern browser including mobile ones (iOS and Android). It comprises a VNC client that runs on the server which is also a server for the remote browser client, using Websockets for the communication.

Finally, xinetd super-server daemon is an advanced network daemon, which can launch and maintain different network services (HTTP, FTP, Proxy servers etc.). Therefore, using it to start the VNC server, it is possible to configure each instance with different properties. In this project, 9 servers from port 5901 to 5909 have been configured (this allows us to have 9 concurrent users, each one in a different server). As a matter of example, the configuration parameters for port 5091 is shown in Table **??**, where the main arguments we have used are:

– **disable**: Set to no, then enabled. If disabled, xinetd will not start the service.

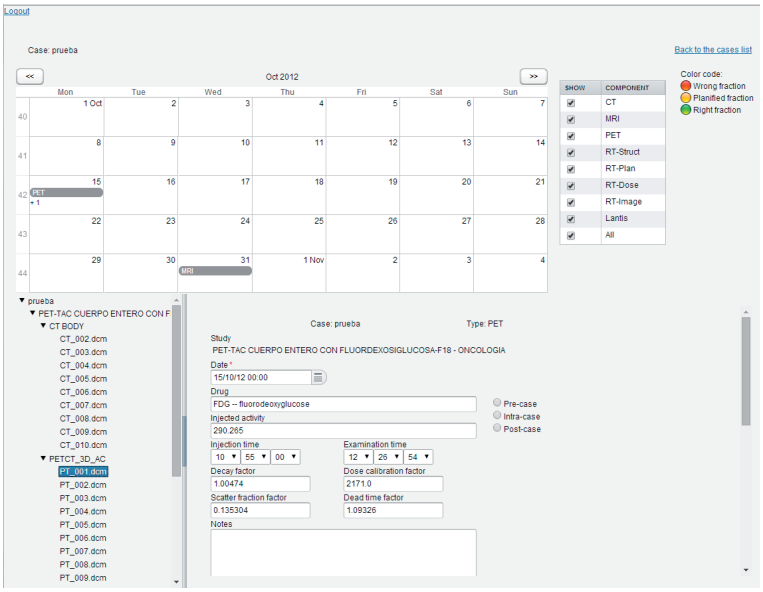


Fig. 4. Calendar view

Parameter	Value
port	5901
disable	no
socket_type	stream
protocol	tcp
wait	no
user	nobody
server	/opt/TigerVNC/bin/Xvnc
server_args	:1 -inetd -once -query localhost -geometry 1024x768 -depth 24 -fp catalogue:/etc/X11/fontpath.d -SecurityTypes None
type	UNLISTED

Table 1. xinetd configuration parameters for VNC

- **user**: Under which user account the service will be started. We use 'nobody' for security reasons. The actual user will log in using the login screen anyway.
- **wait**: Sets if the service is started automatically at the system startup, or the xinetd daemon waits on the port and starts it at first connection to that port. We selected no, so xinetd will wait for connections.
- **server_args**: The VNC server arguments:
  - **-inetd**: Specifies that the VNC server is started by xinetd.
  - **:1**: Specifies the VNC server desktop number. Should be linked to the port (:1 for 5901, :2 for 5902, etc.).

- **-query localhost**: This ensures loading of the XMDCP, the login screen and the desktop environment.
- **-geometry, -depth and -fp**: VNC screen size, color depth and fonts.
- **-once**: The VNC session will be terminated on log out.
- **-SecurityTypes=None**: No password will be needed for the connection. The user always authorizes itself on the login screen.

When the xinetd daemon detects an incoming connection in one of the configured ports, it deploys the VNC server. The VNC server shuts down when the user disconnects from it, leaving the port and the server available for new connections.

Parameter	Value
port	8081
disable	no
socket_type	stream
protocol	tcp
wait	no
user	nobody
server	/usr/bin/novnc_inetd_httpd
server_args	1024 768
type	UNLISTED

Table 2. xinetd parameters for noVNC server

xinetd is used also to start noVNC server. The used parameters are shown in Table **??**.

In this case the server is a script. This script looks for the first VNC server port available, for the first available port for noVNC WebSocket (from 6081 to 6090) and detects the size of the display, because the VNC menubar is 20 pixels high. Finally, the Kanaca noVNC launch script is executed. This script deploys a mini-webserver and WebSockets proxy (websockify). Websockify is a WebSocket to TCP proxy/bridge. This allows a browser to connect to any application/server/service. At the most basic level, websockify just translates WebSockets traffic to normal socket traffic. Websockify accepts the WebSockets handshake, parses it, and then begins forwarding traffic between the client and the target in both directions (See Figure **??**)

The command to run this script is:

```
<noVNC_path>/launch.sh --listen $portw --vnc $IP_host:$port
                       --web <web_path>
```

where

– –**listen PORT**: Indicates the port for proxy/webserver to listen on. In our case portw is the first available port.

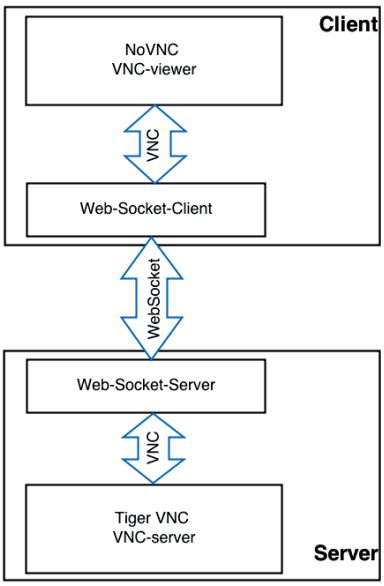


Fig. 5. websockify schema

- –**vnc VNC_HOST:PORT**: VNC server host:port proxy target. Specifies the location of a running VNC server.
- –**web PATH**: Path to Web files

Using these configured modules, the ARTFIBio web server can integrate the co-registration application. When the user selects this option, the web server creates a new virtual desktop on the deployed Virtual Machine, starting the co-registration application with the configuration file (currently, only the path to access the treatment files is included in, but other parameters can be added in the future). Later, it connects to port 8081 which executes the described script. Finally, the browser is redirected to the URL:

```
http://$IP_host:$portw/vnc_auto.html?host=$IP_host\&port=$portw
```

The VNC connection to the registry tool of the remote application can be seen in Figure **??**

5 Conclusions and future work

In this paper the ARTFIBio web platform has been presented. It has been designed to suit the necessities in the study of tumor response quantifications, including the capacities to upload and manage several DICOM fobjects from images (CT, PET/CT and MRI) and radiotherapy (RTSTRUCT, RTPLAN, RTDOSE and RTIMAGE). The web platform integrates a co-registration desktop application

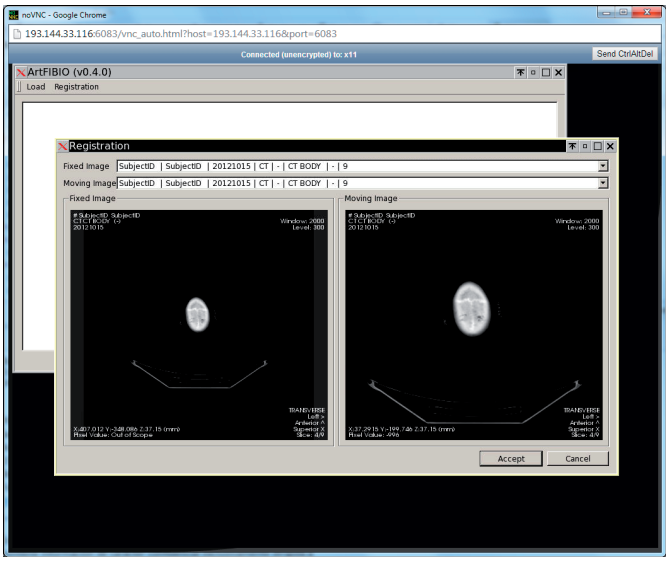


Fig. 6. VNC connection window

using noVNC, which permits a smooth integration with the client web browser without installing additional applications in the client. This solution will enable the integrattion of other desktop applications that are under developement in the project or from third parties without major modifications. The platform is at this moment being verified and validated by the radiotherapists at the Hospital. Therefore, there are not availabe experimental results yet.

To support more concurrent users, the ARTFIbio platform must evolve to use the elasticity of the Cloud IaaS, managing the deployment of more VMs for running deskop applications. Also, instead of presenting in the browser the full VM desktop, we plan to show only the desktop application interface.

Acknowledgements

This work has been supported by ISCIII Grant PI11/02035.

References

1. J. Heukelom, O. Hamming, H. Bartelink, F. Hoebers, J. Giralt, T. Herlestam, M. Verheij, M. v. d. Brekel, W. Vogel, N. Slevin, E. Deutsch, J. Sonke, P. Lambin, and C. Rasch, "Adaptive and innovative Radiation Treatment FOR improving Cancer treatment outcome (ARTFORCE); a randomized controlled phase II trial for individualized treatment of head and neck cancer," BMC Cancer, vol. 13, no. 1, p. 84, 2013.
2. Stewart RD, Li XA. "BGRT: biologically guided radiation therapy-the future is fast approaching", Med Phys 2007;34:3739-51

3. Grgoire V et al "PET-based treatment planning in radiotherapy: a new standard?", J Nucl Med 2007;48 Suppl 1:68S-77S
4. http://www.nlm.nih.gov/NIHbmic/nih_data_sharing_repositories.html
5. Kenneth C. et al., "The Cancer Imaging Archive (TCIA): Maintaining and Operating a Public Information Repository", Journal of Digital Imaging, Vloume 26, Number 6, December 2013, pp 1045-1057. http://www.cancerimagingarchive.net
6. Westberg J. et al. "A DICOM based radiotherapy plan database for reseacrch collaboration and reporting", Journal of Physics: Conference SEries 489 (2014) 012100
7. Palta J.R. et al. "Developing a national radiation oncology registry: From acorns to oaks", Practical Radiation Oncol. 2 10-7.
8. Blanquer I. et al. "TRENCADIS - secure architecture to share and manage DICOM objects in a ontological framework based on OGSA", Stud. Health. Technol. Inform. 2007;126:115-24
9. J. Delgado-Mengual, Y. Vives-Gilabert, A. Sainz-Ruiz, M. Delfino-Reznieck, B. Gómez-Ansón. "PICNIC - Portal-based Platform for Processing of Neurodegenerative Diseases". BIOINFORMATICS 2010:239:244
10. http://tomcat.apache.org/
11. http://www.mysql.com/
12. https://vaadin.com/home
13. Iago Landesa-Vázquez, José Luis Alba-Castro, Moisés Mera-Iglesias, David Aramburu-Núñez, Antonio López-Medina, and Víctor Muñoz-Garzón. "ARTFIBio: A Cross-Platform Image Registration Tool for Tumor Response Quantification in Head and Neck Cancer", 2nd IEEE International Conference on Biomedical and Health Informatics (BHI 2014), Valencia 1-4 June, 2014.
14. http://sourceforge.net/apps/mediawiki/tigervnc/index.php
15. http://kanaka.github.io/noVNC/
16. https://github.com/kanaka/websockify

ALOE platform: An overview of a service-oriented architecture for research in breast cancer diagnosis supported by e-infrastructures

José M. Franco-Valiente[1], César Suárez-Ortega[1], Miguel A. Guevara-López[2], Frederico Valente[2], Naimy González-de-Posada[3], Joana P. Loureiro[4], and Isabel Ramos[4]

[1] CETA-CIEMAT, Centro Extremeño de Tecnologías Avanzadas, Trujillo, Spain
josemiguel.franco@ciemat.es, cesar.suarez@externos.ciemat.es
[2] IEETA-UA, Institute of Electronics and Telematics Engineering of Aveiro
{mguevaral}{fmvalente}@ua.pt
[3] Faculty of Engineering, University of Porto, Porto, Portugal
nlgezposada@inegi.up.pt
[4] Faculty of Medicine, University of Porto, Porto, Portugal
joanaploureiro@gmail.com, radiologia.hsj@mail.telepac.pt

Abstract. This article presents an overview of the ALOE platform. ALOE provides a service-oriented architecture aimed at the research in the early detection of breast cancer diagnosis. The development of the ALOE platform is carried out by collaboration among CETA-CIEMAT, INEGI, FMUP-HSJ and UA. ALOE supports two research lines in breast cancer diagnosis: the development of well-performing Computer Aided Diagnosis (CAD) systems and the development of new tools based on e-Learning techniques to improve radiologists training. All ALOE modules are designed to work as a whole system but can be used individually in other systems and expose RESTful interfaces to be exploited by third party systems. ALOE components make use of e-Infrastructure resources to accomplish their tasks. The final objective of this work is to provide a reference platform for researchers, specialists, and students in breast cancer diagnosis.

1 Introduction

According to the World Health Organization, breast cancer is the second most common cancer worldwide, and first in women [13]. This disease caused more than half a million deaths in 2010 [13], and in the European Union it is responsible of one every six deaths from cancer in women [14].

The detection of this kind of cancer in its early stages is a very effective method to reduce mortality [10] and mammography reading is the most common technique used by specialists [10] for early detection. Visual inspection of anomalies is a repetitive and fatiguing process which leads to a diagnosis error rate between 10% and 30% [11] [12].

There are two main practises in order to improve the cancer detection rate: double-reading of mammographies and single-reading of mammographies supported

by Computer-Aided Detection or Computer-Aided Diagnosis systems (CADe and CADx respectively) [8]. In the first one, two radiologists check the same study to determine a diagnosis. In the second practise, a CAD system assists the human expert to identify abnormalities and diagnose them. Though it is very effective [1], the double-reading method is expensive in time and economical cost because it needs two radiologists for each study. This problem is aggravated by the fact that there is a lack of specialists in the area of breast cancer diagnosis [17]. Consequently, CAD systems have increased their popularity in recent years [12].

Since 2008, the Centre of Extremadura in Advanced Technologies (CETA-CIEMAT), the Institute of Mechanical Engineering and Industrial Management (INEGI) and the Faculty of Medicine of the University of Porto - Centro Hospitalar São João (FMUP-HSJ) collaborate to improve the diagnosis of breast cancer. In 2013, the Institute of Electronics and Telematics Engineering of Aveiro (IEETA) from the University of Aveiro joined the collaboration.

The research activities of the consortium focused on the generation of well-performing CAD systems for breast cancer diagnosis during the first years of the collaboration. These systems were based on machine learning classifiers (MLCs) developed and tested by exploiting the computing resources of the GRID infrastructure [6] from IBERGRID/EGI. The best configurations of these classifiers have obtained results greater than 85% AUC (ROC), but there are some configurations under development with results that can be considered as "excellent" (being more than 90% AUC) [18].

Nowadays, a new research line has been launched by the project team. It is focused on reducing the rate of missed cancers (not detected cancers) by improving the training process of new specialists in the breast cancer detection. To achieve the objective set, new software applications based on e-Learning principles [21] are being developed.

This recently opened research line is motivated by the importance acquired by e-Learning at the moment in education [3]. E-learning is the use of Internet related technologies to deliver solutions that enhance knowledge and performance [20]. It is natural that the medical community has shown interest in this topic, because of the importance of Medical Informatics (MI) in the modern medical practice [4]. The main challenge of the introduction of e-Learning in medical education lies on the need of specific tools for medicine students in most of the cases [2].

This fact is very important in the case of radiologists training, not only because of the need of new specialists, but also for the high rate of students that abandon their training [12]. The early exposure of training in radiology to medical students has shown a high improvement in the impression of this speciality among them, leading to higher rates of individuals considering radiology as a career to study [2]. The typical way to train new radiology students consists of the readings of radiology images (in our case mammograms) and consulting sessions based on real cases. This methodology perfectly fits with some e-Learning methods, especially with simulated cases with self-assessment.

Consequently, the consortium considered that the know-how acquired during the first years of the collaboration and the direct contact with target users will lead to develop new tools based on e-Learning techniques that improve the training of

new specialists in breast cancer diagnosis. Moreover, these tools aim at being also used by experienced radiologists that want to improve their skill in diagnosis, so that they can practice at their own pace with real cases out of their daily work.

Both research lines coordinate their activities to develop the ALOE platform, a service-oriented architecture based on advanced computing and storage resources from e-Infrastructures. The following sections in this article present the ALOE platform, its main components, current status and future work.

2 Description of the ALOE platform

This section describes the architecture of the ALOE platform, as shown in Figure 1. ALOE is formed by a group of services, data repositories and user interfaces (both web and desktop applications) interconnected among them and supported by advanced computing and storage infrastructures.

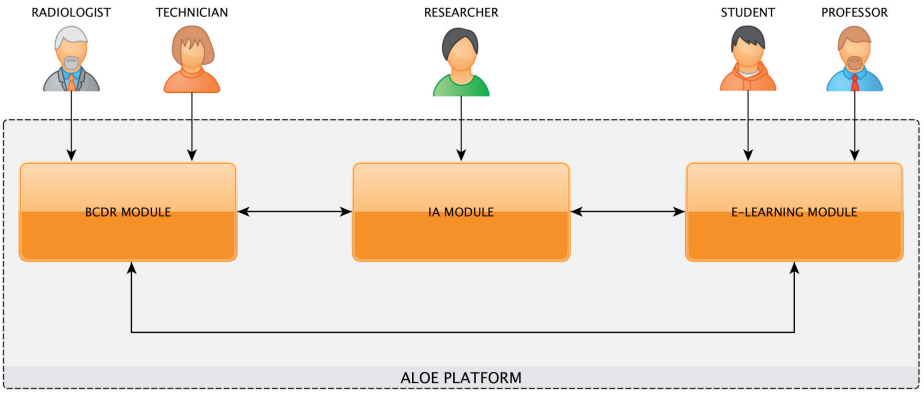


Fig. 1. ALOE platform architecture

As shown in the architectural model exposed in Figure 1, the system can be described mainly in three main modules that cooperate but which can also work autonomously. This autonomy is given thanks to the degree of specialisation of each of those modules since they have been designed with the aim of enabling the reuse in future systems, either as separate individual modules, or in collaboration with others.

These three modules implement services that expose RESTful [5] interfaces. These interfaces enable their connection via the Internet when using the Hypertext Transfer Protocol Secure protocol (HTTPS). This feature allows a federated deployment of the components in different physical locations (i.e. in different computing centres from an e-infrastructure such as that of IBERGRID/EGI [24], in a cloud computing provider such as Amazon AWS or Microsoft Azure, or in both

of them simultaneously). Besides, it also provokes that the overall system performance depends on the available bandwidth and the underlying network latency.

The following subsections try to describe in detail the three main modules that shape the ALOE platform.

2.1 BCDR module

This module was named after the first Iberian digital repository for breast cancer diagnosis called Breast Cancer Digital Repository or BCDR [22]. BCDR is the first opened repository (registration required) of historic cases of breast cancer in the Iberian Peninsula. It is wide-ranging annotated and it includes a compilation of breast cancer anonymized patients' cases evaluated by expert radiologists containing clinical data (detected anomalies, breast density, BI-RADS [16] classification, etc.), lesions outlines, and image-based computed features. These cases belong to the FMUP-HSJ and the students of radiology handle them in their training period. At the moment of writing this article, the BCDR repository contains more than 2000 lesions from real patients and contains several kinds of digital images such as mammography, ultrasound and MRI images.

The BCDR repository is compliant with the digital repository specification set by the Digital Repository Infrastructure (DRI) platform [23]. DRI was created in order to ease the exploitation of the storage resources offered by GRID infrastructures to host digital repositories. Nowadays, DRI has evolved to support a wider range of storage providers. This allows a fast and easy generation, deployment and management of digital repositories from different nature without considering concrete aspects of the underlying storage used. Regarding the DRI platform instance for the BCDR digital repository, a MySQL database has been configured to store all metadata of the repository and a distributed file system based on LUSTRE is used to store the digital content (i.e. mammograms, ultrasound images and other non-structured data). The aforementioned instance is hosted in the computing infrastructure of CETA-CIEMAT.

The Mammography Image Workstation for Analysis and Diagnosis (MIWAD) [7] is used to feed the BCDR digital repository. MIWAD is a software suite of desktop applications based on Java technologies. MIWAD accesses the Application Programming Interface (API) provided by DRI to operate with BCDR content. MIWAD is composed of two interconnected applications (via Remote Method Invocation or RMI) called MIWAD-DB and MIWAD-CAD. Of them, MIWAD-DB allows the management of the metadata from the BCDR digital repository and the execution of CRUD (Create, Read, Update and Delete) operations. On the other hand, MIWAD-CAD, an image workstation, allows the handling of the digital content of the repository by means of image processing operations. Besides, it also allows radiologists to identify the abnormalities in the images and the calculation of a set of image descriptors based on shape, color and texture. This set of descriptors can be checked in [15] [18].

By itself, BCDR module counts on BCDR database. The data model of this database is a simplified version of the metadata of the BCDR digital repository [9] where all fields related to the lesion traceability have been removed as

they are not considered relevant for breast cancer diagnosis. BCDR web site (http://www.bcdr.eu) and BCDR-WS web service (https://api.bcdr.eu) use BCDR database as the data layer. To feed the BCDR database, a set of batch processes are used to: 1) check data integrity and anonymous character, 2) map the data sources from the digital repository to the database and 3) execute optimisation operations over the digital content of the repository to ease their publication over the Internet.

Finally, the BCDR-WS exposes the content of the BCDR database to be accessed by the rest of the modules of the ALOE platform or third-party applications. BCDR-WS implements a RESTful web service developed with Eve framework. Eve is a framework that simplifies the development of RESTful web services built in Python. Moreover, Eve is powered by other technologies such as the web microframework Flask, the NoSQL database MongoDB and the in-memory database engine Redis.

The BCDR module also includes a web site, which allows researchers and specialists in breast cancer diagnosis to visualise the content of the BCDR database. The BCDR database is divided in two different data sets: 1) a Film Mammography-based Repository (BCDR-FM) and 2) a Full Field Digital Mammography-based Repository (BCDR-DM). The BCDR-FM is composed by 1010 (998 female and 12 male) patient cases (with ages between 20 and 90 years old), including 1125 studies, 3703 mediolateral oblique (MLO) and craniocaudal (CC) mammography incidences and 1044 identified lesions clinically described (820 already identified in MLO and/or CC views). With this, 1517 segmentations were manually made and BI-RADS classified by specialised radiologists. MLO and CC images are grey-level digitized mammograms with a resolution of 720 (width) by 1168 (height) pixels and a bit depth of 8 bits per pixel, saved in the TIFF format.

The BCDR-DM, still in construction at the time of writing, is currently composed by 724 (723 female and 1 male) Portuguese patients cases (with ages between 27 and 92 years old), including 1042 studies, 3612 MLO and/or CC mammography incidences and 452 lesions clinically described (already identified in MLO and CC views). With this, 818 segmentations were manually made and BI-RADS classified by specialised radiologists. The MLO and CC images are grey-level mammograms with a resolution of 3328 (width) by 4084 (height) or 2560 (width) by 3328 (height) pixels, depending on the compression plate used in the acquisition (according to the breast size of the patient). The bit depth is 14 bits per pixel and the images are saved in the TIFF format. It must be said that both data sets were created with anonymous cases from medical archives (complying with current privacy regulations as they are also used to teach regular and postgraduate medical students) supplied by the FMUP-HSJ.

2.2 AI module

The AI module (from Artificial Intelligence) includes all components and services based on Artificial Intelligence techniques from the ALOE platform. These components are the Classifier Engine (see Figure 2), the MLC-WS web service and the Recommender Engine.

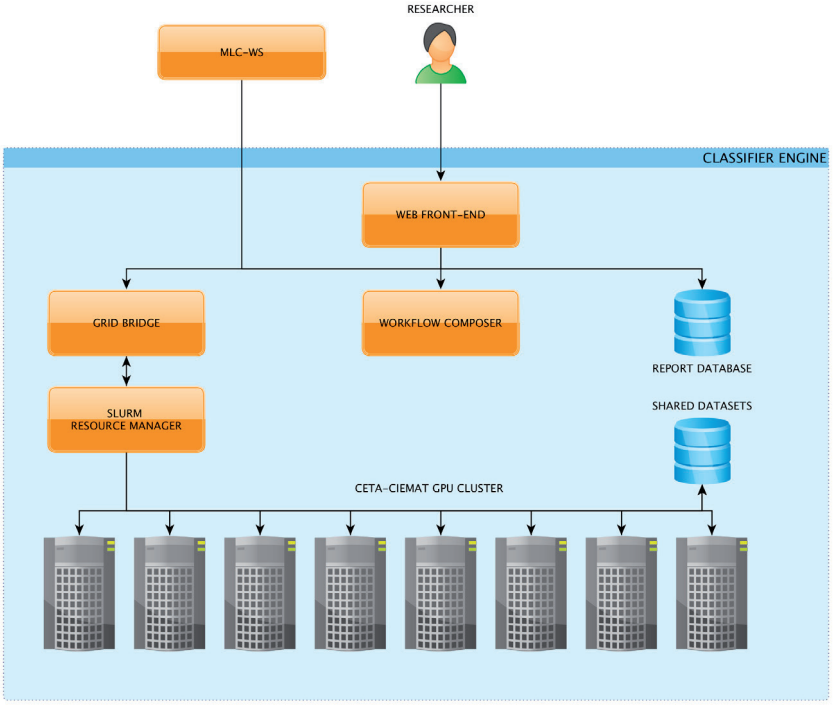


Fig. 2. Classifier Engine deployment at CETA-CIEMAT

The Classifier Engine is responsible for the construction, training and evaluation of machine learning classifiers (MLCs) for breast cancer diagnosis. This component allows researchers to design workflows for the evaluation of several configurations of MLCs. These workflows can be stored and reused in the Workflow Composer. Once there, the workflow is translated to a set of jobs to be sent to the computing infrastructure by the Grid Bridge. According to the current deployment of the ALOE platform, the target infrastructure is the GPU cluster at CETA-CIEMAT. All jobs are sent to the resource manager SLURM from the aforementioned infrastructure. The data sets used by the jobs are stored in a shared directory and can be accessed from any of the compute nodes. The Grid Bridge supervises the state of the jobs related to the workflows, gather the results when finished and store them in the Report database. Researchers can get the workflow results by using a web application that accesses the data through the Classifier Engine RESTful API.

The Classifier Engine is developed in Python and, at the moment of writing, supports the machine learning libraries supported by Python-Scikit, PSVN and Weka.

The MLC-WS web service provides a RESTful interface to the IA module so that it can be exploited for the rest of the modules of the ALOE platform and/or

third-party applications. This interface consists of an evolution of the API provided by MIWAD to integrate MLCs [7]. This web service uses the Classifier Engine to generate the results. The MLC-WS web service is also based on the Flask web micro-framework.

Finally, the Recommender Engine is the latest component added to the IA module. This component is currently in its early stage of development at the moment of writing, and it will allow to retrieve related studies of breast cancer given another one. The Recommender Engine will use the associated data of a study (clinical data and computed image descriptors) to identify all studies from the BCDR database that may be of interest for the users of the ALOE platform. This will be achieved by executing a clustering algorithm (Nearest Neighborhood) [19] to calculate the shortest distance among the studies.

2.3 E-learning module

The e-learning module is closely bound to the recently opened research line which, supported by collaborators, focuses itself on the improvement of the diagnosis of breast cancer by means of providing better training to specialists from this field. The e-learning module has four components (see Figure 3): 1) the ALOE database, which stores all information related to users, courses, tests and practical cases, 2) the knowledge base Breast Cancer Ontology, which stores all the concepts related to breast cancer diagnosis, 3) the ALOE-WS web service, which, at the same time exposes the contents of the previous databases and 4) the ALOE website, a frontend by means of which users can instruct themselves in the distance thanks to the Internet.

As shown in Figure 3, the e-learning module presents the usual components of a web application, using a database as backend and getting access to external services such as MLC-WS web service, provided by the IA Module or the BCDR-WS web service provided by the module with the same name.

For the development of the ALOE website, responsive design techniques have been applied so that users can use the tool from a variety of devices such as mobile phones, tablets, laptops and desktop computers. The team is also working on being able to offer User Experience (UX) according to the feedback given by specialists from the FMUP-HSJ. By these means, needs in this field of training will be covered and the percentage of refusals is expected to decrease. It is for this reason that the ALOE website is being developed using frameworks and libraries such as AngularJS and the latest features offered by HTML5 and CSS3 languages.

The exploitation of the BCDR-WS web service enables users of this educational tool to work with real cases, while the use of the ALOE-WS web service allows the execution of diagnosing operations for those cases from the BCDR database without a given diagnosis.

The social capacities of the tool are another of the team's concern. The final goal is that users will be able to get in touch with other users, share relevant studies and build a contact network. This way, teachers and students will share and comment cases of lesions that drew their attention among others. Finally, it

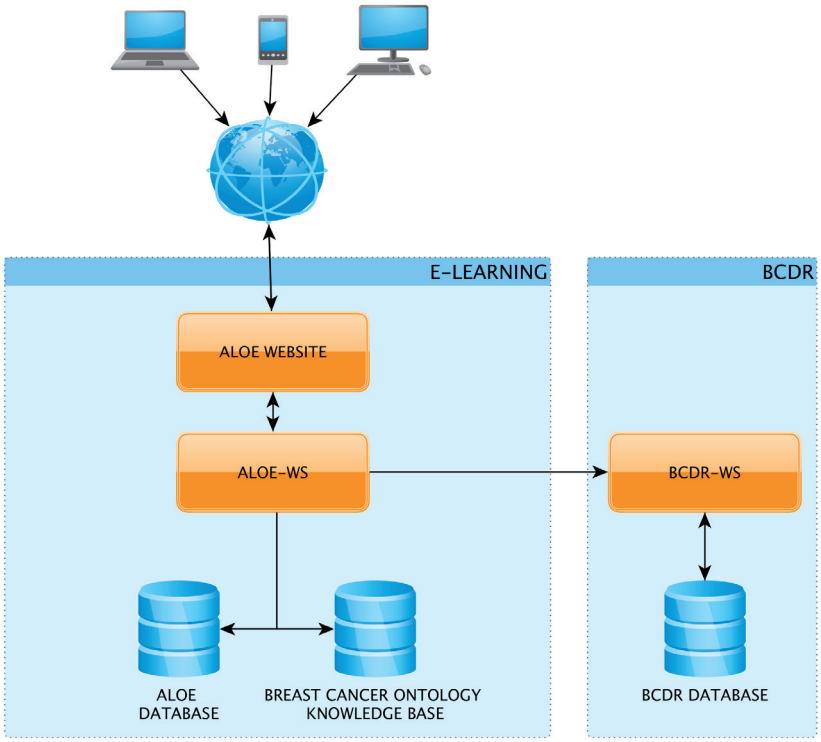


Fig. 3. e-Learning module

needs to be pointed out that the exposure of a REST API through the ALOE-WS web service will enable third-party applications to exploit this module's capacities.

3 Current status

Nowadays ALOE Platform is on trial. The main components of the different modules have been implemented and their communication interfaces published internally for testing purposes.

BCDR Module development is in its final stage; this is mainly due to the fact that its components were developed in the first phases of this collaboration. At present, the work stream is aimed at virtualizing MIWAD application into a Virtual Desktop Infrastructure (VDI) based on Citrix XenDesktop technology so that its maintenance can be centralised making its administration easier. Besides, the enrichment of the BCDR Repository with more cases from the FMUP-HSJ is also being dealt with. Apart from this, five data sets have been published on the BCDR website and a line of work is being carried out too in order to publish a new Golden Dataset from cases containing high definition images and endorsed by biopsies. As far as IA Module is concerned, the Classifier Engine is on testing and work is mainly

being done on the evaluation of new configurations of breast cancer classifiers in order to be able to improve those results obtained by former classifiers. MLC-WS web service is completely developed and waiting for validation. As previously mentioned, the Recommender Engine is now in its initial stage of development.

Finally, as for the e-Learning module, the web site ALOE is being developed and a first testing version has been released for its evaluation. FMUP-HSJ specialists are carrying out this evaluation. This released version allows the access for consultation of the data held by the BCDR database at the same time it allows the execution of tests based on real cases taken from the BCDR database itself.

Regarding the generation of the breast cancer ontology knowledge base, members of IEETA-UA and FMUP-HSJ are currently working in its definition.

4 Future work

ALOE Platform is being developed at the expected pace and a first production release (openly accessible for all users) is expected by the end of the present year. The e-Learning website will be developed according to the received feedback given by FMUP-HSJ specialists and it will be provided with the ability to interact with the IA module to classify undiagnosed lesions belonging to the BCDR database. This development will be carried out with the aim to introduce these means in the the training process of new specialists from the beginning of 2015 onwards.

Due to the specialisation of the different modules of the ALOE Platform, the collaborating team attempts to explore possibilities to apply them to other kinds of lesions, such as those related to lung and/or kidney cancer and the detection of cerebral palsy.

Likewise, migration of some components to the EGI Federated Cloud Platform will also be considered because of the resources such infrastructure offers.

The members of the consortium will carry on working on the search of new data sources to keep on enriching the BCDR repository creating partnerships with universities, R&D centres, public health administrations and medical associations always keeping as final objective that of making BCDR repository become a tool of reference for research on methods for breast cancer detection and diagnosis.

5 Conclusions

The collaboration among CETA-CIEMAT, INEGI, FMUP-HSJ and IEETA-UA is working to improve the diagnosis of breast cancer in two research lines: the research in developing well-performing machine learning classifiers to build Computer Aided Diagnosis systems and the research in new tools based on e-learning principles to enhance the radiologists training and to increase their skill in breast cancer diagnosis. These research lines are motivated by the lack of specialists in the area and by the high costs associated to current techniques to diagnose breast cancer.

ALOE platform includes a set of specialised modules that reflects the research work developed in the aforementioned research lines and aims at being a reference platform for researchers in breast cancer diagnosis. To accomplish this, ALOE will

publish a set of opened services and data repositories supported by the computing and storage resources of e-Infrastructures like IBERGRID/EGI.

Nevertheless, further research is needed to determine if CAD systems and e-Learning tools are relevant in breast cancer diagnosis, and ALOE platform needs to be tested in a real scenario to determine if it covers the final users needs. Owing to this, the participation of the specialists from FMUP-HSJ is crucial to get early feedback that will enable the settling of ALOE as a reference platform in breast cancer diagnosis research.

Acknowledgements

This work is part of the GRIDMED research collaboration project among CETA-CIEMAT (Spain), INEGI, FMUP-HSJ and UA (Portugal). This work is also partially supported by project Cloud Thinking (University of Aveiro) (CENTRO-07-ST24-FEDER-002031), co-funded by QREN, "Mais Centro" program.

The four institutions express their gratitude to the European Regional Development Fund (ERDF-FEDER), which partially funds the project.

References

1. Brown J., et al., "Mammography screening: an incremental cost effectiveness analysis of double versus single reading of mammograms". *BMJ (Clinical research ed.)* (Vol. 312, pp. 809–812), 1996.
2. Branstetter B. F. IV, et al., "Preclinical Medical Student Training in Radiology: The Effect of Early Exposure", *American Journal of Roentgenology*, 2007 188:1, W9-W14
3. Ellet B et al. "Introduction: Is e-learning better than... ?" Piskurich GM, editor. *The AMA handbook of e-learning: Effective design, implementation, and technology solutions*. New York: American Management Association. 2003: 112-136.
4. Fain J. A., "Technological innovation: retooling for the future". *Diabetes Educ* 1997 Mar-Apr; 23(2): 109.
5. Fielding R. T., Taylor R. N., "Principled design of the modern Web architecture". *Proceedings of the 2000 International Conference on Software Engineering*, 2000
6. Foster I., Kesselman C., "The grid 2: Blueprint for a new computing infrastructure". 2nd ed. Elsevier, 2004.
7. Franco-Valiente J. M. et al., "MIWAD: A Software Suite For Building CAD Methods". *15th International Conference on Experimental Mechanics* (22-17 July 2012), Porto, Portugal
8. Gromet M., "Comparison of Computer-Aided Detection to Double Reading of Screening Mammograms: Review of 231,221 Mammograms", *American Journal of Roentgenology* 2008 190:4, 854-859.
9. Guevara-López M. A. et al., "BCDR: A Breast Cancer Digital Repository. 15th International Conference on Experimental Mechanics" (22-17 July 2012), Porto, Portugal
10. Lee C. H., "Screening mammography: proven benefit, continued controversy", *Radiologic clinics of North America*, vol. 40, pp. 395-407, 2002.
11. Kerlikowske K., et al., "Performance of Screening Mammography among Women with and without a First-Degree Relative with Breast Cancer." *Annals of Internal Medicine*. 2000 Dec;133(11):855-863.

12. Kolb T. M. et al., “Comparison of the performance of screening mammography, physical examination, and breast US and evaluation of factors that influence them: an analysis of 27,825 patient evaluations". *Radiology*. 2002; 225(1): 165-175.
13. Matheus B. R. et al., “Online Mammographic Images Database for Development and Comparison of CAD Schemes" *Journal of Digital Imaging*. 24 (2011) 500-506.
14. Moreira I. C. et al., “INbreast: Toward a Full-field Digital Mammographic Database", *Academic radiology*. 19 (2012) 236-248.
15. Moura, D. C., Guevara-López M. A., “An evaluation of image descriptors combined with clinical data for breast cancer diagnosis". *International Journal of Computer Assisted Radiology and Surgery*, 8(4), 561–574, 2013.
16. D’Orsi C. J., et al. “Breast Imaging Reporting and Data System: ACR BI-RADS mammography", *4th Edition ed.: American College of Radiology*, 2003.
17. Penhoet E. E. et al., “Saving Women’s Lives: Strategies for Improving Breast Cancer Detection and Diagnosis", *The National Academies Press*, 2004.
18. R. Ramos-Pollan et al., “Discovering Mammography-based Machine Learning Classifiers for Breast Cancer Diagnosis." *Journal of Medical Systems* (9 April 2011), pp. 1-11.
19. Ricci F., Rokach L. and Shapira R., “Introduction to Recommender Systems Handbook", *Recommender Systems Handbook*, Springer, 2011, pp. 1-35
20. Rosenberg M., “E-Learning: Strategies for Delivering Knowledge in the Digital Age" New York, McGraw-Hill, 2001.
21. Suárez-Ortega C. et al., “Improving the breast cancer diagnosis using digital repositories" *HealthCom 2013, Lisboa*. p. 571-578.
22. BCDR webpage: http://bcdr.inegi.up.pt/
23. DRI webpage: http://dri.ceta-ciemat.es/
24. EGI webpage: https://www.egi.eu/

Towards Grid Environment to Perform Filtering Techniques for Improving Medical Images: an Use case of Mammographic Images

Estibaliz Parcero[1], Damian Segrelles[2], Vicent Vidal[3], Ignacio Blanquer[2], and Gumersindo Verdu[1]

[1] Instituto de Seguridad Industrial, Radiofísica y Medioambiental (ISIRYM)
Universitat Politècnica de València (UPV), València, Spain
esparig@upvnet.upv.es, gverdu@iqn.upv.es
[2] Instituto de Instrumentación para Imagen Molecular (I3M)
Universitat Politècnica de València (UPV), València, Spain
dsegrelles@dsic.upv.es, iblanque@dsic.upv.es
[3] Departamento de Sistemas Informáticos y Computación (DSIC)
Universitat Politècnica de València (UPV), València, Spain
vvidal@dsic.upv.es

Abstract. Medical image processing promises major advances in medicine as higher fidelity images are produced. One challenge is to effectively apply filtering techniques in order to facilitate diagnosis or reduce radiation exposure. In many cases, achieving good results in filtering requires set proper input parameters. Those optimum parameters can be found through a multiparametric experiment, therefore Grid computation could be useful in this regard. The experiment conducted seeks to get the optimum input parameter for FPGA filter technique, a method that has proven to be useful when processing colour images, but has not been tested in the medical domain. We take this opportunity to measure the performance of the Grid infrastructure in a real scenario of medical image domain. The experiment showed a peak performance of the Grid of 6 tasks/hour in contrast to 1 task/hour in the sequential execution (local), and 4 tasks/hour in a cluster of 4 nodes. However, the Grid system needs to be continuously monitored and the performance could be heavily affected by the overhead caused by the scheduler, although further analysis is needed to verify the reasons behind that.

1 Introduction

Filtering techniques to improve images, i.e. to detect and correct noises, has been a challenge in the last years, particularly in medical image (X-Rays or computer tomography - CT methods) where the quality of the image can influence the diagnosis of a disease (for instance, in the detection of microcalcifications in a mammogram). That is why a good filter can be employed to improve the output image when using a reduced radiation dose. This is particularly important in CT images, where the aim is minimize the exposure to X-Rays as the amount of radiation tends to be very high. However, a filtering technique is usually a sophisticated method that

has to be tuned with proper numerical parameters in order to get good results. In this work, we use the Fuzzy Peer Group Average [1][2] (FPGA), which is an effective method in color images. Different filtering techniques are applied in function of the type of noise to remove. For instance, methods based on the space or in the frequency domain [3], methods based on solving regularized least squares problems [4] and methods based on the use of nonlinear Diffusion equations [5–13] are applied to remove Gaussian noise. Other example is the case of impulsive noise in which recent techniques based on the concept of Peer Group with Fuzzy metric have provided good results in RGB images [14–16]. In this paper, in order to adequate Fuzzy Peer Group Averaging Filter (FPGA) to mammograms, we conduct an experiment to get the best values for the parameters needed by FPGA filter. The two parameters that FPGA uses are F_t and F_σ, related to the amount of noise present in the image. The images used in the experiment are the mammograms provided by miniMIAS database [17]. The conducted experiment consists in varying the values of the input parameters and applying the filter over all the images in miniMIAS database that have been contaminated with impulsive and Gaussian noise. Afterwards the resulting images are compared with the original images. These results determine the best values for the input parameters. This experiment is a typical multiparametric problem, with a high amount of repetitions, which leads to consider parallelization. In this case, a good approach is the Grid paradigm. The Grid facilitates the execution of concurrent processes and, since our experiment does not need of thread synchronization, messaging, or data partitioning, the execution of the study in the Grid is achievable. Consequently, this presents us the opportunity to monitor the performance of the Grid. The chosen Virtual Organization (VO) is "biomed" that use the resources of the European Grid Infrastructure (EGI) [18]. Our goal in this study is stablish if the use of "biomed" as environment of tests for medical image is feasible and efficient. This paper is structured as follows: Section II explains the materials and methods used in the study: the method FPGA, the miniMIAS database, and the "biomed" VO. Section III shows the experiment conducted, and the architecture of our Grid application. Section IV describes the results and discussion of this experiment. Finally, concluding remarks are presented.

2 Materials and Methods

In this section, the material and methods used in the experiment are described. First, the method FPGA is outlined, secondly a brief summary about the miniMIAS database, and the "biomed" VO.

2.1 Fuzzy Peer Group Averaging Filter (FPGA)

This filter performs in two steps, (i) impulse noise detection and reduction, and (ii) Gaussian noise smoothing. Both steps use a fuzzy peer group of a central pixel x_i in a window W of $n \times n$ and using a fuzzy metric. The definition of the fuzzy peer group is based on the ordering of the pixel neighbours with respect to its similarity to the central pixel x_0. Let ρ be an appropriate similarity measure between two

vectors. Vectors $x_i \in W$ are sorted in a descending order with respect to their similarity to x_0, obtaining an ordered set $W' = \{x_{(0)}, x_{(1)}, \ldots, x_{(n^2-1)}\}$ such that $\rho(x_0, x_{(0)}) \geq \rho(x_0, x_{(1)}) \ldots \geq \rho(x_0, x_{(n^2-1)})$, where $x_0 = x_{(0)}$. The peer group $\rho_m^{x_0}$ of $m+1$ members associated with pixel x_0 is the set

$$\rho_m^{x_0} = \{x_{(0)}, x_{(1)}, \ldots, x_{(m)}\} \tag{1}$$

In [1], a fuzzy logic-based method is proposed to determine the best number of members $\hat{m}$ of a peer group. The fuzzy peer group of a central pixel x_0 in a window W is defined as the fuzzy set $F\rho_m^{x_0}$ defined on the set $\{x_{(0)}, x_{(1)}, \ldots, x_{(\hat{m})}\}$ and given by the membership function $F\rho_m^{x_0} = \rho(x_0, x_{(i)})$. Then the best number $\hat{m}$ of members of $\rho_m^{x_0}$ is defined as the value of $m \in N_W = 1, 2, \ldots, n^2 - 1$ maximizing the certainty of the following fuzzy rule.

Fuzzy Rule 1: Determining the certainty of m to be the best number of members for $\boldsymbol{\rho_m^{x_0}}$ *IF* "x_m is similar to x_0" and the accumulated similarity for $x_{(m)}$ is large *THEN* "the certainty of m to be the best number of members is high". $C_{FR1}(m)$ denotes the certainty of the Fuzzy Rule 1 for m. Then, $C_{FR1}(m)$ is computed for each $m \in N_W$ and the value which maximizes the certainty is selected as the best number $\hat{m}$ of members of $\rho_m^{x_0}$, i.e., $\hat{m} = argmax_{m \in N_W} C_{FR1}(m)$. The certainty of "$x_m$ is similar to x_0" is given by the membership function C^{x_0} determined by the similarity measure

$$C^{x_0}(x_{(i)}) = \rho(x_0, x_{(i)}), i = 0, 1, \ldots, n^2 - 1 \tag{2}$$

The accumulated similarity for x_m denoted $A^{x_0}(x_{(m)})$ is defined by

$$A^{x_0}(x_{(i)}) = \sum_{k=0}^{k=i} \rho(x_0, x_{(k)}), i = 0, 1, \ldots, n^2 - 1 \tag{3}$$

Then, the certainty of "$A^{x_0}(x_{(m)})$ is large" is given by the membership function L^{x_0} defined by

$$L^{x_0}(x_{(i)}) = \frac{(A^{x_0}(x_{(i)}) - 1)(A^{x_0}(x_{(i)}) - 2n^2 + 1)}{(n^2 - 1)^2}, i = 0, 1, \ldots, n^2 - 1 \tag{4}$$

The product t-norm was used as the conjunction operator and therefore no defuzzyfication is needed. Then, $C_{FR1}(m) = C^{x_0}(x_{(m)}) L^{x_0}(x_{(m)})$. The fuzzy similarity function, ρ, used was

$$\rho(x_i, x_j) = e^{\frac{||x_i - x_j||}{F_\sigma}}, i, j = 0, 1, \ldots, n^2 - 1 \tag{5}$$

where $||.||$ denotes the Euclidean norm and F_σ is one of the input parameters of the filter. This function ρ takes values in [0, 1] and satisfies that $\rho(x_0, x_i) = 1$ if and only if $x_0 = x_i$. Another fuzzy rule is used to detect impulse noise.

Fuzzy Rule 2: Determining the certainty of the pixel x_0 to be free of impulse noise IF "accumulated similarity $A^{x_0}(x_{(\hat{m})})$ is large" and "$x_{(\hat{m})}$ is similar to x_0" $THEN$ "x_0 is free of impulse noise" In order to compute the certainty of the Fuzzy Rule 2, denoted by C_{FR2}, the certainty of "$A^{x_0}(x_{(\hat{m})})$ is large" is given by L^{x_0} , (defined in (4)) and the certainty of "$x_{(\hat{m})}$ is similar to x_0" is given by C^{x_0} given by formula (2). The t-norm product is used as conjunction operator and then $C_{FR2}(x_0) = C^(x_0)(x_{(\hat{m})})L^(x_0)(x_{(\hat{m})})$. This certainty is already computed since $C_{FR2}(x_0) = C_{FR1}(\hat{m})$ and then no additional computation is needed. IF the certainty of Fuzzy Rule 2, C_{FR2} satisfies

$$C_{FR2}(x_0) \geq F_t \tag{6}$$

$THEN$ x_0 is free of impulse noise $ELSE$ x_0 is an impulse and it is replaced with VMF_{out} [19]. F_t is a threshold input parameter of the filter, with values in [0,1]. The conducted experiment is designed to get the optimum values of the parameters above-mentioned.

2.2 Mini-MIAS Database

In this work we compare filtering methods over mammographic images from the database of mini-MIAS [17]. This database is a normalized collection of 322 mammograms (black and white images with black sidebars to accommodate a size of 1024 pixels x 1024 pixels). Each one of the images includes information about the character of background tissue (fatty, fatty-glandular, dense-glandular) and the class of abnormality present (calcification, well-defined/circumscribed masses, spiculated masses, other/ill-defined masses, architectural distortion, asymmetry, normal). In case of presence of an abnormality, the severity is indicated (benign or malignant) and located with x, y coordinates.

2.3 Biomed Virtual Organization

The "biomed" VO is a large scale international and multidisciplinary VO supporting communities from the Life Sciences sector. The VO is operated on the European Grid Infrastructure [18] (EGI). EGI provide access to high-throughput computing resources using Grid computing techniques [20]. The EGI links centres in different European countries to support international research in many scientific disciplines like Medical Imaging analysis, giving to the scientists access to more than 370,000 logical CPUs, and 170 PB of disk capacity. The middleware used for accessing to EGI is gLite [21], which became part of the EMI [22] distribution and are now managed as independent projects in their own right, providing software to grid infrastructures such as EGI. In gLite, the job management is implemented through the Workload Management Server (WMS), being the user who interacts with the scheduling server. The WMS determines the status of computational and storage resources and selects a Computer Element to execute the job.

3 Experiment

The goal of the study is to measure the performance of the Grid in terms of getting the optimum values for F_t and F_σ, input parameters of the FPGA filter. These parameters indicate how the filter behave, thus it is important to establish correctly these parameters for its use in mammograms. The experiment conducted consists in apply the FPGA filter to 322 mammograms. The output generated comprises these quality measures obtained by comparison of the original and filtered image: mean absolute error (MAE), mean square error (MSE), and peak signal to noise ratio (PSNR) for each pair consisted of F_t and F_σ. F_t values go from 0.05 up to 0.25, and F_σ values from 50 to 255 in increments of 5 units, this means executing the task 882 times per image. It should be mentioned that completion of the execution of the whole set of images is not entirely necessary for our purpose, to get the optimum input parameters for FPGA in the filtering of mammograms, nevertheless the experiment is conducted with all the images in order to measure the performance of a Grid infrastructure.

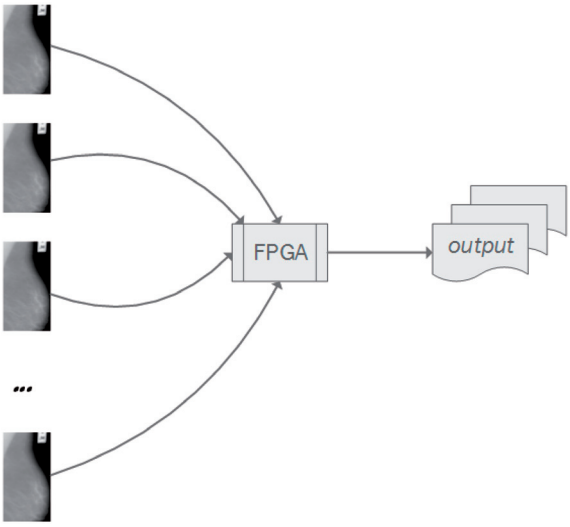


Fig. 1. Workflow of sequential execution.

There are subtle differences between the sequential execution (figure 1) and the Grid execution (figure 2). The execution flow of this experiment follows this basic scheme:

– In the **preprocess** step the input data for the distributed part of the application is prepared on the local resource, i.e. creation of specific job files from a template, and delivery of required files (image to process and FPGA executable).

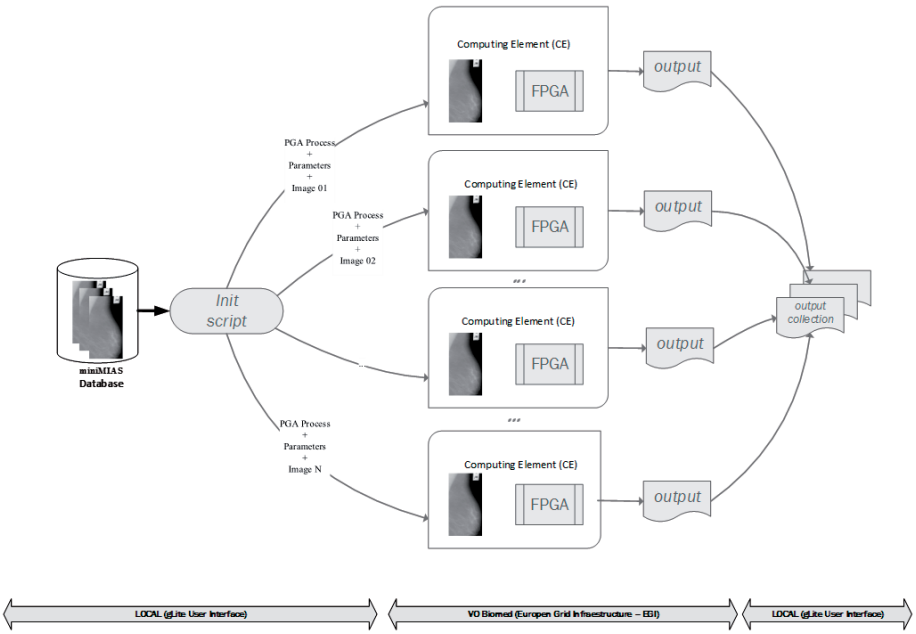


Fig. 2. Workflow of grid execution.

- In the **remote execution** step the application employs a Grid to execute a set of remote tasks generating their own output data.
- Finally, in the **postprocess** step the output data is collected and analysed on the local resource.

We prepare the set of images to process (one is the original image, and the other one has been contaminated with noise), the program to execute (the FPGA filter), and a script to generate the job files mentioned below, in the preprocess step. The submission of jobs is done through gLite middleware, thus the job files are defined using a specific language for gLite, the Job Description Language (JDL). Once we have the collection of JDLs, we can submit the jobs using the following command:

```
glite-wms-job-submit <delegation-opts> [options] <jdl_file>
```

A script that encompass the submission of the whole collection of jobs has been implemented to semi-automate the process. Other scripts have been developed as well in order to monitor, cancel, resubmit jobs (in case something goes wrong), and get the output.

4 Results and Discussion

We compared the performance, in terms of tasks/hour, of a sequential system, a cluster, and a grid system. A sequential execution of one task needs approximately

one hour, depending on the CPU available, therefore the execution of 322 tasks would need 322 hours, that is, more than 13 days. We consider the possibility of having a cluster available, that would mean being able to execute the 322 tasks in 322/number of nodes in the cluster. In our scenario we propose an ideal cluster of 4 nodes, which would need approximately 80 hours of execution. We can assume they both are reliable, and they do not fail.

Table 1. Results of grid execution

Hour	Tasks completed	Comments
0	0	Submission of tasks
10	60	
20	120	Submission ends / Resubmission
30	155	
40	175	Resubmission ends

In other hand, the execution in the grid is problematic due to the need of a more-or-less continuous monitoring, since the available scheduler do not provide the error control level needed for a completely unattended execution of tasks. In table 1 we gather the results at some points of the execution for 40 hours. Despite the initial performance in the grid is 6 tasks/hour, better than our ideal cluster, we expected better performance having into account that more than 20 tasks are running at the same time. One reason for poor performance is a slow down due to the overhead caused by the submission; another reason is the low quality of the nodes. First submission get 120 tasks done, second submission only achieves 55 additional tasks. Several resubmissions need to be done in order to get the completion of every task, this fact can lead to an important overhead, slowing down the entire system.

Another problem we encountered as it shows figure 3 is the decrement in performance of the grid. That is presumably caused by the overload of the system, further analysis would be required to give an accurate explanation.

Regarding the multiparametric experiment performed that analyses the filter performance in terms of PSNR as a function of F_t and F_σ, results are showed in figures 4 and 5 for mammographic image number 1. Best PSNR obtained was of 30.082 for parameters $F_t = 0.72$ and $F_\sigma = 195$. We observed that values can oscillate in an interval of values without much loss of performance. Firstly, for F_t and a fixed value of $F_\sigma = 195$, the bests result for PSNR are obtained for a F_t in $[0.65 - 0.75]$. Secondly, for F_σ and a fixed value of $F_t = 0.72$, the best results for PSNR are obtained for a F_σ in $[175, 225]$. In figure 6 we can compare the result of applying the filter to a corrupted mammogram (Gaussian noise = 10, impulse noise = 0.10). Further results will be presented in future works.

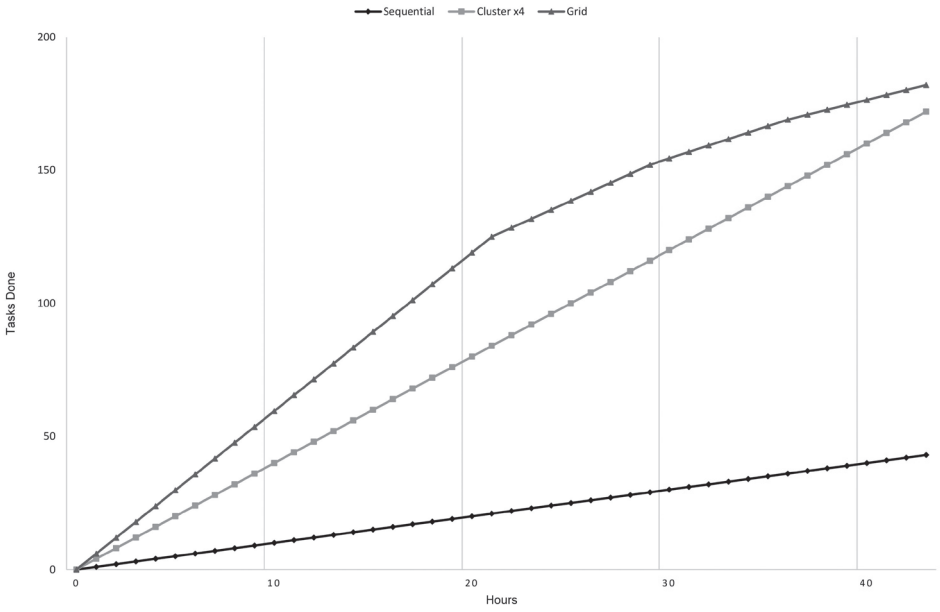


Fig. 3. Tasks per hour in sequential, cluster and grid systems.

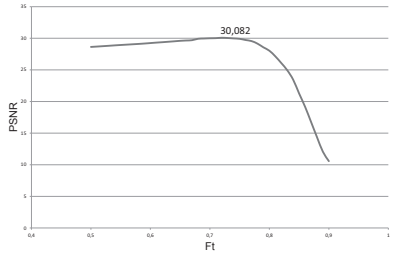


Fig. 4. Performance in terms of PSNR of the FPGA filter as a function of the F_t parameter with $F_\sigma = 195$ in image 1.

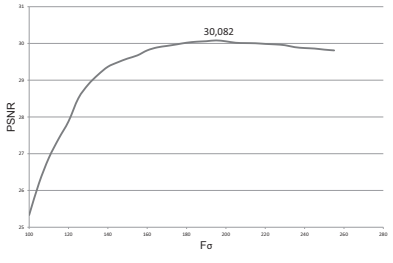
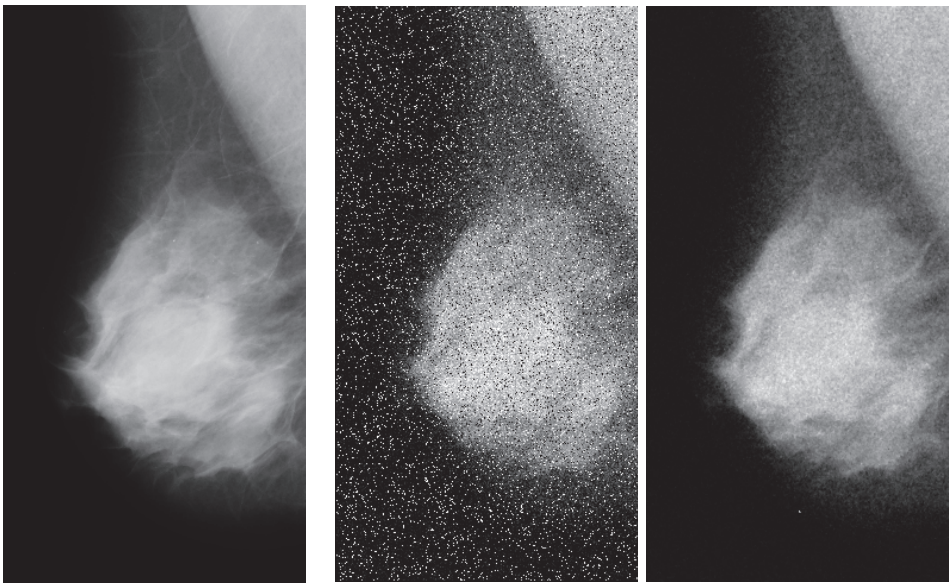


Fig. 5. Performance in terms of PSNR of the FPGA filter as a function of the F_σ parameter with $F_t = 0.72$ in image 1.



(a) (b) (c)

Fig. 6. (a) Original, (b) corrupted and (c) filtered mammogram for visual comparison.

5 Conclusions

In this work, a comparison between sequential, cluster and Grid systems had been presented, using an experiment to get best values for input parameters of an image filter over a mammogram dataset. Beyond the technical effort to adapt the experiment to the Grid, this work presents promising results in terms of performance. Having a Grid infrastructure to our disposal decreases the time consumption in the execution of multiparametric problems. The Grid showed a peak performance of 6 tasks/hour, despite better results were expected, this means a 6x improvement over the sequential process, and a 1.5x improvement over the execution in the cluster. However, a fundamental problem arises, the performance of the Grid tends to tail off over time, further analyses are required to answer that question.

Acknowledgements

This work has been supported by the Universitat Politècnica de València and partially funded by the project "COMPUTACIÓN DE ALTAS PRESTACIONES Y SISTEMAS HÍBRIDOS" (TIN2011-26254).

References

1. S. Morillas, V. Gregori, and A. Hervas. Fuzzy peer groups for reducing mixed gaussian-impulse noise from color images. *IEEE Transactions on Image Processing*, 18(7):1452–1466, July 2009.

2. J. Arnal, L.B. Sucar, M.G. Sanchez, and V. Vidal. Parallel filter for mixed gaussian-impulse noise removal. In *Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), 2013*, pages 236–241, September 2013.
3. J.C. Russ. *The Image Processing Handbook, Fifth Edition*. CRC Press, December 2006.
4. M. Bertero and P. Boccacci. *Introduction to Inverse Problems in Imaging*. CRC Press, January 1998.
5. F. Catté, P. Lions, J. Morel, and T. Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM Journal on Numerical Analysis*, 29(1):182–193, February 1992.
6. L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1–4):259–268, November 1992.
7. J. Weickert, B.M.T.H. Romeny, and M.A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Transactions on Image Processing*, 7(3):398–410, March 1998.
8. A. Marquina and S. Osher. Explicit algorithms for a new time dependent model based on level set motion for nonlinear deblurring and noise removal. *SIAM Journal on Scientific Computing*, 22(2):387–405, January 2000.
9. T. Chan, A. Marquina, and P. Mulet. High-order total variation-based image restoration. *SIAM Journal on Scientific Computing*, 22(2):503–516, January 2000.
10. S.L. Keeling. Total variation based convex filters for medical imaging. *Applied Mathematics and Computation*, 139(1):101–119, July 2003.
11. J. Weickert. Efficient image segmentation using partial differential equations and morphology. *Pattern Recognition*, 34(9):1813–1824, September 2001.
12. P. Mrázek and M. Navara. Selection of optimal stopping time for nonlinear diffusion filtering. *International Journal of Computer Vision*, 52(2-3):189–203, May 2003.
13. D.L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, May 1995.
14. J.G. Camarena, V. Gregori, S. Morillas, and A. Sapena. Fast detection and removal of impulsive noise using peer groups and fuzzy metrics. *Journal of Visual Communication and Image Representation*, 19(1):20–29, January 2008.
15. B. Smolka and A. Chydzinski. Fast detection and impulsive noise removal in color images. *Real-Time Imaging*, 11(5–6):389–402, October 2005.
16. M.G. Sánchez, V. Vidal, J. Bataller, and J. Arnal. A fuzzy metric in GPUs: Fast and efficient method for the impulsive image noise removal. In Erol Gelenbe, Ricardo Lent, and Georgia Sakellari, editors, *Computer and Information Sciences II*, pages 323–330. Springer London, January 2012.
17. J. Suckling et al. The mammographic image analysis society digital mammogram database. In *Exerpta Medica*, pages 375–378, 1994.
18. European grid infrastructure, June 2014.
19. J. Astola, P. Haavisto, and Y. Neuvo. Vector median filters. *Proceedings of the IEEE*, 78(4):678–689, April 1990.
20. C. Lazou. Plans for a future european grid infrastructure organization. *European Parliament Magazine*, 265:8–9, April 2008.
21. CERN. gLite: Lightweight middleware for grid computing.
22. European middleware iniciative, June 2014.

Building Strategies to Discover Mammography Classifiers for Breast Cancer Diagnosis

Raúl Ramos-Pollán[1], Miguel Ángel Guevara López[2], Fabio Augusto González Osorio[3], John Edilson Arévalo Ovalle[3], Isabel Ramos[4]

[1]Universidad Industrial de Santander, Bucaramanga, Colombia
rramosp@uis.edu.co
[2] Universidade de Aveiro, Aveiro, Portugal
mguevaral@ua.pt
[3] Universidad Nacional de Colombia, Bogotá
{fagonzalezo, jaarevaloo}@unal.edu.co
[4]Faculty of Medicine - Centro Hospitalar São João, at University of Porto, Portugal
radiologia.hsj@mail.telepac.pt

Abstract. This work shows a method for building strategies to discover mammography based machine learning classifiers (MLC) for supporting breast cancer diagnosis by using BIGS (the Big Image Data Analysis toolkit), a software framework for large scale machine learning over distributed computing resources. We used cloud computing resources at the Unit for Supercomputing and Scientific Computing (SC3) at UIS to train different configurations of selected MLC and to understand their behavior on the Breast Cancer Digital Repository (BCDR), a comprehensive annotated repository of mammograms built in collaboration with the Faculty of Medicine - Centro Hospitalar São João, at University of Porto, Portugal, and publicly available. The conjunction of these tools, methods and data allows us to guide our quest for well performing MLC targetted for introduction in clinical practice. We report herewith well performing classifiers but, what is most relevant, we show how we acquire strong evidence to support the process of enlarging the dataset and tuning our MLC methods.
Keywords: breast cancer, mammography, machine learning classifiers, computer-aided diagnosis, high performance computing

1 Introduction

Building datasets for developing automatic tools to support diagnosis in bioimage analysis is a lengthy and costly process, involving doctors, engineers and mathematicians in an effort to obtain clean annotated datasets and developing machine learning methods adequate to those. Breast cancer is the second most common form of cancer in the world, causing more than half a million deaths per year [4]. Double reading of mammograms (two radiologists read the same mammograms) has been advocated to reduce the proportion of missed cancers. But the workload and costs associated with double reading are high. Much research effort has been going in the past years in applications of Computer Aided

Diagnosis (CADx) to assist diagnosis and contribute reduce such costs. Machine learning classifiers have been increasingly used to approach this task [5], and a few datasets are being used by the community [9].

The BCDR (http://bcdr.inegi.up.pt) is a wide-ranging annotated public repository composed by some 1700 breast cancer patients' cases of the northern region of Portugal that provides annotated cases including mammography lesions outlines, anomalies observed by radiologists, pre-computed image-based descriptors as well as related clinical data. Using the BCDR, our goal is to produce well performing MLCs for supporting Breast Cancer CADx methods able to be targetted for introduction in clinical practice. Understanding and gaining confidence on how to guide the effort of building such datasets and MLCs is key to avoid sparing human and material resources in this endeavour. Therefore, through the BCDR, in this work we show a method for exploring and discovering mammography based MLCs, combining the human effort to acquire and annotate the datasets with software tools and computing resources. Through this, we aim at gaining statistical confidence to gradually guide our decisions on whether dive into acquiring more data, preprocess better the data or work harder on the improving the machine learning methods we are using. Preliminary work on this subject by the authors can be reviewed in [8] and [7]. This paper is structured as follows. Section 2 describes the BCDR. In Section 3 we describe the software framework developed for these kind of tasks. Section 4 shows the experimental setup devised and, finally, Sections 5 and 6 show the results and concluding remarks.

2 The Breast Cancer Digital Repository

We built the BCDR, the first Portuguese Breast Cancer database, with anonymous cases from medical historical archives (complying with current privacy regulations as they are also used to teach regular and postgraduate medical students) supplied by Faculty of Medicine - Centro Hospitalar São João at University of Porto, Portugal. BCDR is supported and hosted on the Digital Repositories Infrastructure (DRI) platform developed by the authors' team, which also includes an specialized mammography workstation for interacting with the repository. The BCDR data model, hosted at our DRI infrastructure, is a subset of the Digital Imaging and Communications in Medicine (DICOM) file format (http://dicom.nema.org/) customized by radiologists at the Hospital for storing and managing specific case information related to digital mammography images. This work complements results in managing DICOM objects within Grid environments (see the TRENCADIS middleware [1] and also [2] and [3] by applying the DICOM standard at the Hospital and integrating it with the full machine learning classifiers development lifecycle where (1) mammography images of the BCDR are preprocessed through the graphical workstation, (2) specialized radiologists mark and classify biopsied cases which are then sorted in the BCDR, (3) data features are extracted from the stored annotations, (4) MLC configurations are massively searched and (5) selected MLC are integrated

back into the workstation providing automated second opinion diagnosis to doctors. BCDR is subdivided in two different repositories: (1) a Film Mammography Repository (BCDR-FM) and (2) a Full Field Digital Mammography Repository (BCDR-DM). Both repositories were created with anonymous cases from medical archives (complying with current privacy regulations as they are also used to teach regular and postgraduate medical students) supplied by the Faculty of Medicine – Centro Hospitalar São João, at University of Porto (FMUP-HSJ). BCDR provides normal and annotated patients cases of breast cancer including mammography-based lesions outlines, anomalies observed by radiologists, precomputed image-based descriptors as well as related clinical data. The BCDR-FM is composed by 1010 (998 female and 12 male) patients cases (with ages between 20 and 90 years old), including 1125 studies, 3703 mediolateral oblique (MLO) and craniocaudal (CC) mammography incidences and 1044 identified lesions clinically described (820 already identified in MLO and/or CC views). With this, 1517 segmentations were manually made and BI-RADS classified by specialized radiologists. MLO and CC images are grey-level digitized mammograms with a resolution of 720 (width) by 1168 (height) pixels and a bit depth of 8 bits per pixel, saved in the TIFF format. The BCDR-DM, still in construction, at the time of writing is composed by 724 (723 female and 1 male) Portuguese patients cases (with ages between 27 and 92 years old), including 1042 studies, 3612 MLO and/or CC mammography incidences and 452 lesions clinically described (already identified in MLO and CC views). With this, 818 segmentations were manually made and BI-RADS classified by specialized radiologists. The MLO and CC images are grey-level mammograms with a resolution of 3328 (width) by 4084 (height) or 2560 (width) by 3328 (height) pixels, depending on the compression plate used in the acquisition (according to the breast size of the patient). The bit depth is 14 bits per pixel and the images are saved in the TIFF format.

3 The Big Image Data Analysis Toolkit

3.1 Overview

BIGS promotes opportunistic, data locality aware computing for machine learning processes on distributed and opportunistic computing environments through (1) a data partition iterative programming model (as described below), (2) users assembling image processing jobs by pipelining machine learning algorithms over streams of data, (3) BIGS workers are software agents deployed over the actual distributed computing resources in charge of resolving the computing load, (4) a NoSQL storage model with a reference NoSQL central database, (5) removing the need of a central control node so that workers contain the logic to coordinate their work through the reference NoSQL database, (6) simple and opportunistic deployment model for workers, requiring only connectivity to the reference NoSQL database, (7) redundant data replication throughout workers, (8) a two level data caching in workers in memory and disk, (9) a set of strategies for workers for data access so that users can enforce data locality aware computing

or only-in-memory computing; and (10) a set of APIs through which BIGS can be extended with new algorithms, storage and data import modules. More information can be found at http://www.3igs.org. Prototype releases of BIGS are described in [6].

3.2 Computing model

BIGS endorses a data partition iterative computing model through which workers can exploit locality aware computation if so desired by the user. Through this model, distributed data analysis jobs are structured into a repeatable sequence of INIT- STATE, MAP and AGGREGATE-STATE operations using a reference NoSQL database (http://hbase.apache.org/) for storing data and process states. MAP operations loop through the input datasets to be processed are split into a user definable number of partitions feeding multiple MAP operations. BGIS workers take over operations by inspecting the job dependency graph stored in the reference database. Developers program their algorithms by providing implementations for a set of Java API methods. We use Java generics to provide type safety for algorithms developers. When a BIGS worker takes over an operation, it creates the appropriate programming context and makes the corresponding data available (state and/or data partitions) to the implementation being invoked. Currently BIGS includes implementations for different image feature extraction, supervised and unsupervised learning algorithms. Jobs may chain a sequence of algorithms following BIGS computing model so that users can assemble their experimental data processing pipelines into a single configuration object (the job), as shown in next section. Additionally, users can enforce different workers strategies so that they can be restricted to always process the same data partitions throughout a whole job promoting, therefore, locality aware computing within workers and minimizing network traffic.

3.3 Jobs

Data processing pipelines are expressed as jobs (see below) which are then decomposed into execution units so that most of them can be run in parallel following the computing model described above. BIGS workers can be seen as computing agents that, whenever they are available, greedily take over any execution unit by invoking concrete implementations of the Process and Storage APIs, according to the data processing pipeline described by the user and the configured underlying storage. Experimenters use BIGS by (1) deploying workers over the available computing resources as mentioned above, (2) loading the dataset to be processed in the reference NoSQL database, (3) optionally distributing the dataset partitions into the existing workers and (4) defining their data processing pipelines into a job definition file and submitting it for workers to start taking over and executing its operations. BIGS exposes its functionality mainly through a shell command line interface, through which workers can easily be launched over computing resources (through a shell command line '> bigs worker') or also through Java Web Start for the less experienced users.

```
# FIRST STAGE: Patch Sampling and Feature Extraction
stage.01.proc: bigs.modules.learning.LogisticRegression
stage.01.LogisticRegression.gamma: 0.1:0.5:0.7
stage.01.LogisticRegression.beta: 0.05:0.1:0.15
stage.01.LogisticRegression.maxNumberOfIterations: 10:30:100:200

stage.01.validation: bigs.modules.validation.Bootstrapping
stage.01.Bootstrapping.numberOfBootstraps: 20
stage.01.Bootstrapping.percentageForTraining: 0.7
stage.01.Bootstrapping.predictProcess:
          bigs.modules.learning.LogisticRegressionPredict

stage.01.input.table: BCDR
```

Fig. 1. Example of BIGS exploration job

4 Methodology

4.1 Dataset preparation

As described, from BCDR-FM, two specialized radiologists at the Hospital used the mammography workstation to produce the 1517 features vectors BI-RADS classified. A few cases were normal ones (no lesion) and the rest showed one of the following lesions: microcalcifications, calcifications, masses, architectural distortions and asymmetries. However, as some BI-RADS classes were rather scarce, classes were grouped as benign (BI-RADS 1, 2 and 3) and malign (BI-RADS 4, 5 and 6) making the dataset binary with 61% of vectors representing benign cases and 49% malign ones. Several image processing operations were applied and validated on all selected images to improve ROIs details. The goal was to find fast and simple image preprocessing operations for denoising and enhancing possible pathological lesions or normal tissue image regions. This validation included suitable combinations of preprocessing filters, mathematic morphology, thresholding and edge detection among others techniques. However, the most common defect of mammography images was the poor contrast resulting from a reduced, and perhaps nonlinear, image amplitude range. Then, we found that in a first preprocessing step, ROI details can be in general improved by adjusting image intensities (a conventional contrast enhancement technique based on amplitude rescaling of each pixel). To enhance images contrast we mapped gray scale intensity values of input CC and MLO mammography images to new values such that 1% of data is saturated at low and high intensities to produce a new image in which the contrast is increased.

4.2 Exploration of machine learning classifiers

The goals with experiments were to (1) try a number of configurations for MLCs to find well performing models for the datasets extracted from the BCDR-FM and (2) understand what kind of strategy would guide better next phases on

the evolution BCDR in terms on how much effort should we plan for increasing these datasets and/or working with stronger machine learning algorithms. Therefore, we set forth building learning curves for simple classification models (Naive Bayes and Logistic Regression) exploiting computing resources for exploring a wide range of parameter configurations and, what is more important, gaining statistical confidence over the classification results obtained. We prepared two job configuration files such as the one shown in Figure 1. Observe the parameter sweep (in gamma, beta and numberOfIterations) and also the bootstrapping configuration. Choosing bootstrapping as a validation method allows us to run several time the same experiment, sampling each time the available data to split it into a train and a validation subset. In the example job in Figure 1 we use a sampe of 70% of the data for training and repeat the process 20 times (numberOfBootstraps). Performance is the reported as an average and standard deviation for the classification accuracy over the 20 bootstraps. For each method (Logistic Regression and Naive Bayes) we performed this experimental cycle with samples 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80% and 90% and build the learning curves. Therefore, we prepared 18 job definition files like the one in Figure 1, one for each bootstrap sample size. As it can be seen we are adopting a coarse-grained paralellism strategy where the same process is instantiated many times either using different configurations, different train data sample sizes or simply repeating the process (number of bootstraps). We used computing resources at the Unit for Supercomputing and Scientific Computing (SC3) at UIS, by encapsulating 5 BIGS workers within Unix processes on 5 cores within the same computing node (100Gb RAM) and another 5 workers on a second computing node, while sharing an HBase dataset in an additional node in SC3. Job definitions were then fed into HBase for the 10 workers to start processing.

4.3 Results

Figure 2 shows the learning curves of the best performing configuration both for Logistic Regression and Naive Bayes. Dark curves show prediction accuracy in the train part of the dataset as the training sample size increases. Light curves show prediction accuracy on the test part of the dataset. Each point in the chart was obtained averaging the prediction accuracy of running 20 times the processes of sampling the train dataset (leaving the unsampled data as test set), train with NaiveBayes or Logistic Regression and measuring performance on the sampled train or test data. Standard deviations of the prediction accuracy were less than 5% in all cases.

As it can be seen, Logistic Regression seems to perform better, being in a moderate bias and variance regime. Bias can be observed as generalization accuracy (as measured in the test sample) and seems to converge to 0.75 in the case of Logistic Regression and 0.6 for Naive Bayes. Variance can be observed as the difference between the converging accuracies in training and validation and is around 0.15 in both cases. The fact that both methods are relatively simple (thus, prone to high bias) leads us to consider a strategy based in the

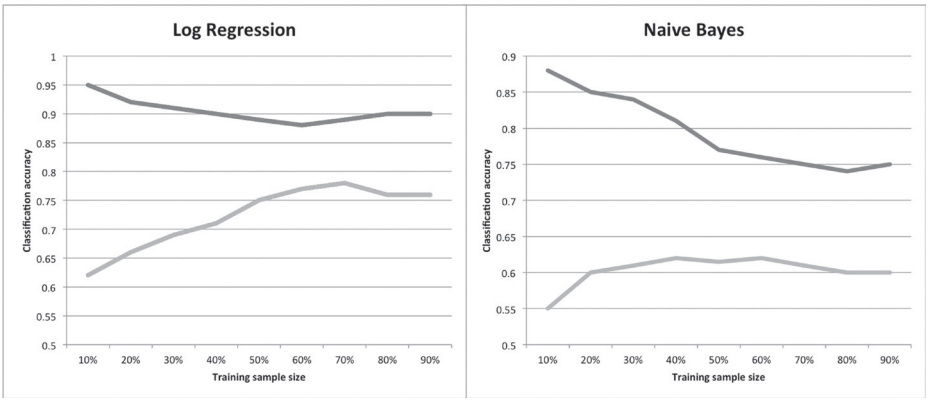


Fig. 2. Learning curves for Logistic Regression and Naïve Bayes classification for different sizes of the training sample. Dark curves show accuracy performance in the training sample, light curves show accuracy performance on the test sample.

following principles: (1) include additional features and/or other classification methods (SVMs, etc.) to target for reducing bias and, hopefully, variance; and (2) continue enriching gradually the dataset with more patient cases assessing the variance continuously as the dataset size increases.

5 Conclusions

This work showed how the combined usage of computing resources and expertise knowledge can guide the effort of building machine learning classifiers for mammography based breast cancer diagnosis, allowing us to settle a strategy based on (1) deciding to focus on the analysis and classification methods, and (2) releasing some pressure off the data acquisition, manual segmentation and annotation process. This enables us to make a better use of the human and material resources in hand and target our efforts into what might yield more cost-effective actions. Note that our focus here has been not so much the specific machine learning methods we used, but on building the capacity to use the computing resources (software framework and cloud computing) to guide the overall process involving medical specialists, engineers and mathematicians. We expect to continue using this methodology to further improve the classifiers performance and build confidence on the products we hand in for testing in clinical practice.

6 Acknowledgements

This work was supported by the Cloud Thinking project (CENTRO-07-ST24-FEDER-002031), co-funded by QREN, "Mais Centro" program. Experiments presented in this work were carried out using the GridUIS-2 experimental testbed,

being developed under the Universidad Industrial de Santander (SC3UIS) High Performance and Scientific Computing Centre, development action with support from UIS Vicerrectoria de Investigación y Extension (VIE-UIS) and several UIS research groups as well as other funding bodies (see http://grid.uis.edu.co and http://sc3.uis.edu.co).

References

1. Ignacio Blanquer Espert, Vicente Hernández García, Fco Javier Meseguer Anastásio, and J Damià Segrelles Quilis. Content-based organisation of virtual repositories of dicom objects. *Future Generation Computer Systems*, 25(6):627–637, 2009.
2. Mark D Halling-Brown, David S Moss, Clare E Sansom, and Adrian J Shepherd. A computational grid framework for immunological applications. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1898):2705–2716, 2009.
3. Klaus Kayser, Jürgen Görtler, Stephan Borkenfeld, and Gian Kayser. Grid computing in image analysis. *Diagnostic Pathology*, 6, 2011.
4. A. Marcano-Cedeño, J. Quintanilla-Domínguez, and D. Andina. Wbcd breast cancer database classification applying artificial metaplasticity neural network. *Expert Syst. Appl.*, 38(8):9573–9579, August 2011.
5. Bruno RobertoNepomuceno Matheus and Homero Schiabel. Online mammographic images database for development and comparison of cad schemes. *Journal of Digital Imaging*, 24(3):500–506, 2011.
6. Raul Ramos-Pollan, F.A Gonzalez, J.C. Caicedo, A Cruz-Roa, J.E. Camargo, J.A Vanegas, S.A Perez, J.D. Bermeo, J.S. Otalora, P.K. Rozo, and J.E. Arevalo. Bigs: A framework for large-scale image processing and analysis over distributed and heterogeneous computing resources. In *E-Science (e-Science), 2012 IEEE 8th International Conference on*, pages 1–8, Oct 2012.
7. Raul Ramos-Pollan, Miguel Angel Guevara-Lopez, and Eugenio Oliveira. A software framework for building biomedical machine learning classifiers through grid computing resources. *Journal of Medical Systems*, 36(4):2245–2257, 2012.
8. Raul Ramos-Pollan, MiguelAngel Guevara-Lopez, Cesar Suarez-Ortega, Guillermo Diaz-Herrero, JoseMiguel Franco-Valiente, Manuel Rubio-del Solar, Naimy Gonzalez-de Posada, MarioAugustoPires Vaz, Joana Loureiro, and Isabel Ramos. Discovering mammography-based machine learning classifiers for breast cancer diagnosis. *Journal of Medical Systems*, 36(4):2259–2269, 2012.
9. John Suckling, J Parker, DR Dance, S Astley, I Hutt, C Boggis, I Ricketts, E Stamatakis, N Cerneaz, Siew-Li Kok, et al. The mammographic image analysis society digital mammogram database. 1994.

A dataflow-based approach to the design and distribution of medical image analytics

Frederico Valente, Augusto Silva, Carlos Costa, José Miguel Franco Valiente, César Suárez-Ortega, Miguel Guevara

DETI/IEETA - Universidade de Aveiro, Aveiro, Portugal
fmvalente@ua.pt

Abstract. Machine Learning and imaging analytics are major algorithmic components of the software used by medical practitioners in the diagnosis and treatment of diseases. Whether employed by CADx (Computer Aided Diagnosis) or CBIR (Content Based Image Retrieval) tools, the accuracy and relevance of the results to the practitioner are paramount to the success of any such application.

In order to improve on the existing results researchers often find themselves in the need to explore various approaches and methodologies, often using very large datasets and multiple sources of information. Each of these trials can, by itself, be a very time-consuming operation.

One tried and true strategy to speed up operations is the use of a distributed computing platform (delivering the computational load to a number of machines). This raises a set of problems which are often orthogonal to a researcher's interest such as which algorithmic implementations scale or how to distribute data and tasks on the Grid.

In this article we present a framework that empowers researchers to quickly design sets of tests, schedule their execution and have them automatically distributed on a Grid environment. We describe the design and implementation of the solution, and present as an example an experiment concerning the classification of mammography segmentations.

Keywords: Machine Learning Classifiers; Medical Image Analysis; Grid; Cloud; Feature Extraction

1 Introduction

The analysis and exploration of large biomedical datasets has become an essential task for biology, biotechnology and health care professionals. Traditionally, the development of accurate analytics found a bottleneck in the amount of computational power and data available [1]. Nowadays, this trend has come to be reversed due to continuous technological advances, which have enabled institutions to create powerful computational facilities

with large data storage capacity. Furthermore, the total amount of information placed at the researchers grasp is almost incalculable. Large amounts of data are produced by multiple sources such as, for instance, medical modalities [2], genomics [3], or unstructured data collected from the internet.

This shifts researcher's focus from merely analyzing the data into how to analyze or use all the data within a reasonable time frame.

This is particularly important in the field of translational medicine and medical image analysis where a large amount of time is required to create and test models or learn them from data. In this context, other aspects, like data harmonization, process streamlining and, accurate, concise reporting also play an important role to increase the effectiveness of the multi-disciplinary parties involved in translational research.

In this article we present the framework we are currently developing in order to speed up the analysis of CAD algorithms and exploration of Content-Based Image Retrieval recommendation systems (CBIR). The solution proposed streamlines the process of workflow definition and subsequent execution, and is extensible via a module mechanism. This project aims at providing an interface that is easy to use by practitioners, to monitor tests, progress and respective outcomes, and by developers of machine learning and analytic algorithms. Furthermore we aim at empowering a final user with a comprehensive list of premade modules that crystalize various parallelization algorithms for machine learning and image analysis. Computations are automatically scheduled for execution on a Grid hence allowing the quick prototyping of solution placing the focus on the task at hand rather than having to acquire competencies on the complex field of data distribution.

Besides the emphasis placed on performance, the pipeline enforces harmonization of data and takes care of the creation of reports to facilitate communication between clinical researchers and developers.

2 Background

Grid

Distinct approaches to decrease the time spent on computations by distributing them across a number of machines have been tried. In the context of distributed machine learning a new field has even emerged in order to explore the performance gains obtained by distributing machine learning algorithms through a number of machines: large-scale learning [1].

A computing Grid is a federated, potentially heterogeneous and geographically distributed, cluster of computing resources. However, in spite of this concept being well-known, Grids have not yet met the initial expectations of penetration and dissemination throughout scientific disciplines and business domains, remaining mostly within specific academic areas

[4]. Among others, some one of the reasons pointed out as being behind this gap are the difficulty of usage of the middleware, and the fact that there is a steep learning curve and cost barrier for new communities having neither the tradition nor the resources to work with Grids. Furthermore, there is a non-trivial skillset that a developer needs to acquire in order to properly parallelize data and computational tasks.

Cloud

The term "Cloud computing" refers to a provisioning model for virtualized processing and storage capacity [5]. Similar to a Grid in the sense that physical processors and storage systems are housed in large data centers geographically distributed, it differs from it in its goals and access mechanisms. Typically in a cloud system, all functionality is provided by web services, accessed via the Internet, and the infrastructure is managed by third party professional IT organizations. Furthermore, cloud systems are used, not so much for performance gains, but to decouple the business logic from the required infrastructure via virtualization, and leverage the scalability mechanisms that allow for a system to grow as needed, when needed, in a seamless way, with the end user essentially renting capacity on demand [6].

In this project we rely on the cloud provision of a standardized infrastructure (commonly referred to as using Infrastructure as a Service) to deliver access to our platform. This allows users to design workflows and consult reports, from any location, using only their web-browser.

Machine Learning

Machine learning (ML) algorithms are able to infer a function when given a set of data points from its domain (the observed examples) and their respective image (their class).

The ability to construct ML-based classifiers that can explore complex relationships in the data has led to their widespread usage both in academia and in commercial data-mining systems. In the course of the last years, ML algorithms have been applied, with increasingly success, in the medical field, where they are used in CADx tools to classify tumors [7], to detect epileptic seizures [8] or pulmonary nodules [9], among a multitude of other examples.

In fact, nowadays, ML plays a major role in medical data analysis where the high dimensional quantitative data provided by the state-of-the-art medical imaging modalities and high-throughput biology technologies make the use of these techniques particularly apt.

In order to improve the classification performance of ML, a very active area of research, a set of strategies are being tried which range from the usage of larger, more detailed, datasets to the application of deep learning

architectures [10] or novel, more complex, features. What these approaches have in common is their requirement for more processing power in order to produce results in the same amount of time it took the previous generation of ML algorithms.

While the concrete mechanisms to use ML are evolving, the general workflow does not display such variance. The general strategy relies on datasets containing features of interest and the ground truth, which in a medical context is typically curated by experts. These datasets are then used as the basis for the statistical learning embodied by a concrete ML algorithm such as SVM, neural networks or nearest neighbor classification scheme among others [11]. From a computational point of view ML algorithms are expensive to train, and seamless access to computing power is key to undertake successfully any comprehensive data analysis endeavor.

3 Material and Methods

CETA-CIEMAT Environment

The cluster architecture for which this solution is being developed is composed by two distinct partitions, one of which is dedicated to testing (17 nodes, 2 GPUs per node). The production partition is composed by 32 nodes, also comprising 2 GPUs per node (see figure 1).

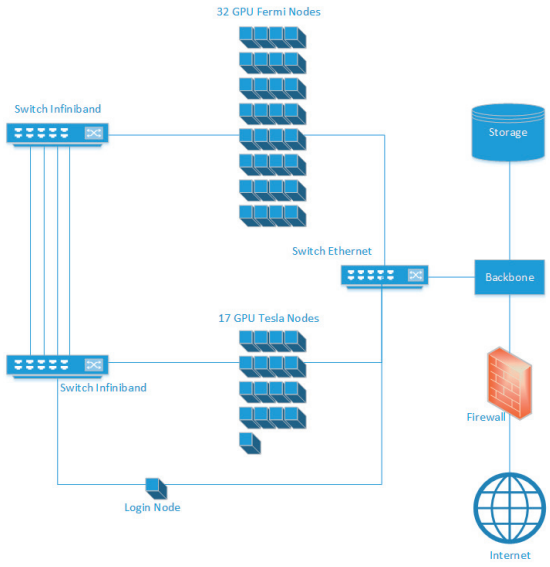


Figure 1 CETA-CIEMAT Grid architecture

A login node to interact with the Grid is made accessible via secure shell. This Grid uses 'slurm' for resource management which is leveraged by the Job Manager in order to distribute the computational load across the various machines.

Design Considerations and Requirements

In order to arrive at the particular architectural design, as presented in figure 2, a set of considerations on the needs of our group were analyzed.

The overarching goal is to provide scientific research teams with intuitive and reusable tools for creating, collecting, storing and sharing clinical and translational research data and results while the underlying implementation places a strong focus on multicore implementations of image analysis and ML algorithms.

The following key features were identified as critical components for supporting our use cases:

- User authentication and role-based security. Provide auditing support. While this solution is not intended for data management and to provide direct modality access is out of the scope of this work, medical information is very sensitive and must remain protected from malicious usage.
- Allow for the submission of datasets in the most common formats. Data import functionality to facilitate bulk import of data from other systems.
- Harmonization of the heterogeneous inputs and outputs of the different modules that make part of our solution.
- Provide a web-based user interface that allows a sensible way to define an experiment's parallel workflow.
- Provide a software generation cycle sufficiently fast to accommodate multiple concurrent projects minimizing the need for custom project-specific programming while allowing it for more complex tasks.
- Facilitate task progress monitoring and administration
- Facilitate component and workflow reuse through well-defined abstractions and component parameterization.
- Real-time data validation, workflow integrity checks and other mechanisms for ensuring model integrity prior to execution.
- Collaborative access to data and workflow reports.
- Automatic scheduling and execution of valid workflows on the Grid with minimal input from the user.
- Scalability. When it comes to any large distributed system, size is just one aspect of scale that needs to be considered. Just as important is the effort required to increase capacity to handle greater amounts of load, commonly referred to as the scalability of the system [12]. Scalability can refer to many different parameters of the system: how much addi-

tional traffic can it handle, how easy is it to add more storage capacity, or even how many more transactions can be processed.
- Performance. A big point of this application is to leverage the Grid in order to speed up the execution of ML tasks that otherwise will take a long time. As such the performance of each of the components matters.
- Explore task parallelism using a dependency-based model and data parallelism by using state-of-the-art distributed ML algorithms.
- Use a reporting mechanism that combines the results from various tasks into a comprehensive report to facilitate communication with medical practitioners.
- Employ a composable model capable of meeting the disparate data analysis needs of various projects.
- Independency of a particular language for the analysis software. Several implementations of Grid middleware force a particular language or interface upon developers. In our particular case we need to leverage a number of tools written by the scientific community in various languages. These include GPU kernels, MPI-based applications and image analysis software written in C, C++ and Java.

System Architecture

In order to support the use cases presented above, we have divided the solution in the multiple decoupled components presented below.

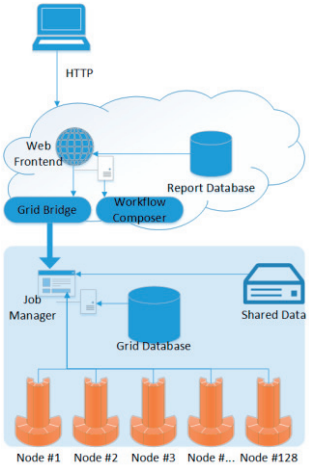


Figure 2 Component diagram for the proposed solution. Both the Grid bridge and workflow composer are deployed on the cloud for optimal accessibility and communicate with the Job manager on the login node of the Grid infrastructure.

The user interface and web-services lay on a virtualized server deployed on a private cloud which also contains a database storing user data such

as test results, workflow definitions and custom analytical modules. Communication with the grid is mediated through the Grid Bridge. This is a python standalone application which provides an API guided towards the dispatching of workflows and datasets to the Grid. Communication between the Bridge and the Grid is encrypted using public key encryption. The Job Manager is the front-end for the Grid. It handles the communication with the Grid Bridge, the scheduling and execution of workflows and the collection and aggregation of each task's results. The Grid infrastructure contains a database as well, in order to handle certain types of datasets, such as the ones provided by the BCDR [13].

A shared file system is provided by the Grid infrastructure and is used to maintain a coherent view of data produced by tasks executing on distinct cores. The Job Manager leverages it to measure progress while the tasks are executing and to keep the application state in a persistent manner hence increasing its robustness. Furthermore, the shared drive contains any dependency required for a task execution, such as a Python interpreter, a Java runtime environment and any data that needs to persist as a file.

Workflow based appliances

In order to explore task parallelism a dependency model must be created so that a user may specify which tasks may run in parallel and which dependencies must be satisfied before a task can run. To do so we use Synchronous Dataflow Graphs [14], a model that has a good track record for modeling concurrent applications on multiprocessor platforms. The semantics of this dataflow graph allows us to detect when a sub task is ready to be launched and how many can be executed in parallel. While in this context some of its properties, such as static throughput analysis, cannot be applied due to the lack of a priori execution times for each module, its expression of task parallelism and scheduling properties are very useful.

Having defined how tasks are modeled, we define a workflow as being our main work unit. It is composed by a dataflow graph, any custom modules and data dependencies plus information on which tasks contribute to the final report. These units are assembled into a single package and forwarded to the Job Manager, which analyses it and subsequently executes it.

In order to create a workflow a user can use the web-based user interface, which contains the analytical modules available, obtained by querying the Job Manager. It is the task of the Web Frontend to present to the user a simple interface, akin to RapidMiner or Matlab's Simulink, and create a workflow file (figure 3, 4) so that a final user does not have to deal directly with workflow definition files.

```
#node definition area
init     = Module('init')
ds     = Module('dataset', src='bcdr_01.csv', truth='last')
split    = Module('split', ratio=0.7, strategy='random')

svm_a    = Module('svm', kernel='rbf', gamma=3)
svm_b    = Module('svm', kernel='linear', c=2)
svm_c    = Module('psvm')
nb     = Module('naive_bayes')
sgd      = Module('sgd')
tree   = Module('extra-trees')

train_svm_a    = Module('train', input=[svm_a,split[0]])
train_svm_b    = Module('train', input=[svm_b,split[0]])
train_svm_c    = Module('train', input=[svm_c,split[0]])
train_nb     = Module('train', input=[nb,split[0]])
train_sgd      = Module('train', input=[sgd,split[0]])
train_tree   = Module('train', input=[tree,split[0]])

test_svm_a   = Module('train', input=[train_svm_a, split[1]])
test_svm_b   = Module('train', input=[train_svm_b, split[1]])
test_svm_c   = Module('train', input=[train_svm_c, split[1]])
test_nb        = Module('train', input=[train_nb, split[1]])
test_sgd     = Module('train', input=[train_sgd, split[1]])
test_tree      = Module('train', input=[train_tree, split[1]])

report   = Module('report', input=[test_svm_a, test_svm_b,
test_svm_c, test_nb, test_sgd, test_tree])

finit  = Module('finit', input=report)

#edge definition area
#edges can be defined as input = name in module definition area
#or here using operator >>
init     >>   ds
ds       >>   split
```

Figure 3 Workflow definition file, employed to dispatch across multiple nodes a simple experiment concerning the effectiveness and comparison of ML classifiers on segmented portions of a mammography.

Modules and Classification Engines

Each of the tasks present in a workflow (such as the "svm_a" or "report" from figure 4) represent explicit instantiations of particular modules whose definition and implementation resides on the Grid. These modules are parameterized and configurable via the user interface.

Furthermore, they encapsulate any exploitable data parallelism as, under the hood, we favor distributed or GPU based algorithms already provided by the scientific community.

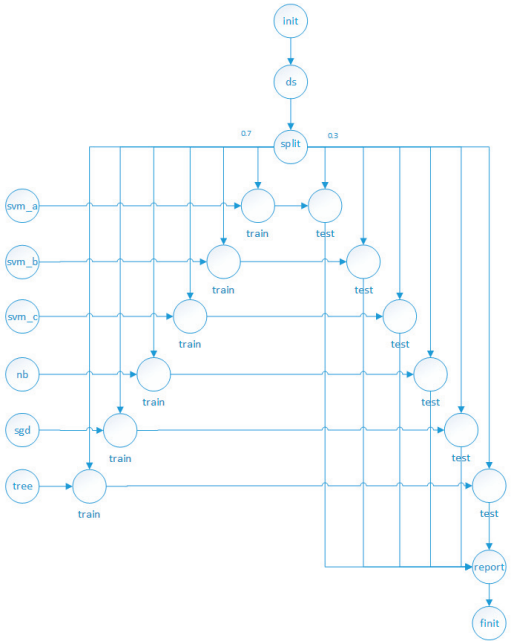


Figure 4 Dataflow graph for the workflow presented in figure 3. Edges from init to the classifier models are kept implicit.

For the module's implementation we are using a multitude of tools such as Python's scikit-learn and Java's Weka which provide a collection of machine learning algorithms such as Neural Networks and Support Vector Machines (SVM) among others. While these libraries were not created for usage on a Grid environment, we provide modules for parameter optimization that automatically distribute several instances in order to estimate which parameters provide a close to optimal response.

It is recognized that most SVMs implementations suffer from a scalability problem in both memory use and computational time [15]. Some of the modules implemented explore data parallelism by using state-of-the-art ML algorithms, such as PSVM [15] that reduces memory usage and are able to perform parallel computations of the objective function using MPI.

Job Manager

The Job Manager's orchestrates the scheduling and execution of workflows, preparing their execution environment beforehand and collecting the results on task termination. It also handles communication with the Grid Bridge via a secure link.

It is an event-based application with transaction support in order to cope with hardware failure.

4 Results and Discussion

The proposed solution is currently under development, however, we can already test some of its functionality. In order to access it we've applied a subset of the implemented classification modules on data provided by the BCDR project.

The experiment consists in the classification of anomalies present in image segmentations. The BCDR repository [13] contains a compilation of over 2000 anonymized studies coupled with annotations provided by expert radiologists and includes the ground truth as provided by a biopsy. Furthermore a set of imaging features, obtained from the segmentations provided by radiologists, are already extracted and present as companions to the datasets.

We have created the dataflow file, as presented in figure 3, and dispatched it to the Grid for execution. The ouput of the classifiers is shown in table 1.

Table 1. Classification results of the application of the workflow from figure 3

Algorithm	Options	Dataset	F1-Score	Std
SVM	Radial kernel, gamma = 3	BCDR_D01	0.865	0.12
	Linear kernel, C=2		0.908	0.18
	PSVM		0.891	0.08
	Radial kernel, gamma = 3	BCDR_ D02	0.890	0.07
	Linear kernel, C=2		0.906	0.204
	PSVM		0.874	0.073
Naïve Bayes		BCDR_D01	0.798	0.23
		BCDR_D02	0.704	0.15
SGD		BCDR_D01	0.854	0.161
		BCDR_D02	0.797	0.155
Extra-trees	estimators=500, criterion=entropy	BCDR_D01	0.854	0.23
	estimators=500, criterion=entropy	BCDR_D02	0.932	0.172

These results sever merely as a test for our infrastructure, nonetheless they seem to match and reinforce the conclusions, already presented in other articles in the literature, that ML classifiers can achieve high discriminatory rates, above 90%.
While this solution is already usable and provides a mechanism to deploy computational tasks into the Grid, some work is still required for it to be useful to a broader audience. Most of the work on that line concerns the user interface, which is yet under development. We have focused, on these initial steps, on the deployment mechanisms and classification algorithms, however, new modules are under development, particularly concerning feature extraction from imaging repositories, similarity sorting, and improvements on the report files generated.

5 Conclusion

Healthcare analytics opens the door to a plethora of opportunities which are nowadays being explored by CAD tools, recommendation and advanced query systems like CBIR among other tools. In order to improve on the current performance of those systems more complex computations must be performed. On the other hand, the volume of medical images is also rapidly increasing, holding significant information that is not exhaustively and sufficiently explored.

In this article we presented the preliminary design of a solution aiming at empower developers and practitioners to design ML models and medical image analysis workflows using a pipeline-based user interface and facilitate collaboration in translational medical investigation. Several solutions have already been proposed to offload processing to the Grid. This project differs from those applications by attempting to be a general approach for test generation and leverage both task and data parallelism for increased performance without requiring an extensive knowledge of distribution methodologies.

Acknowledgments

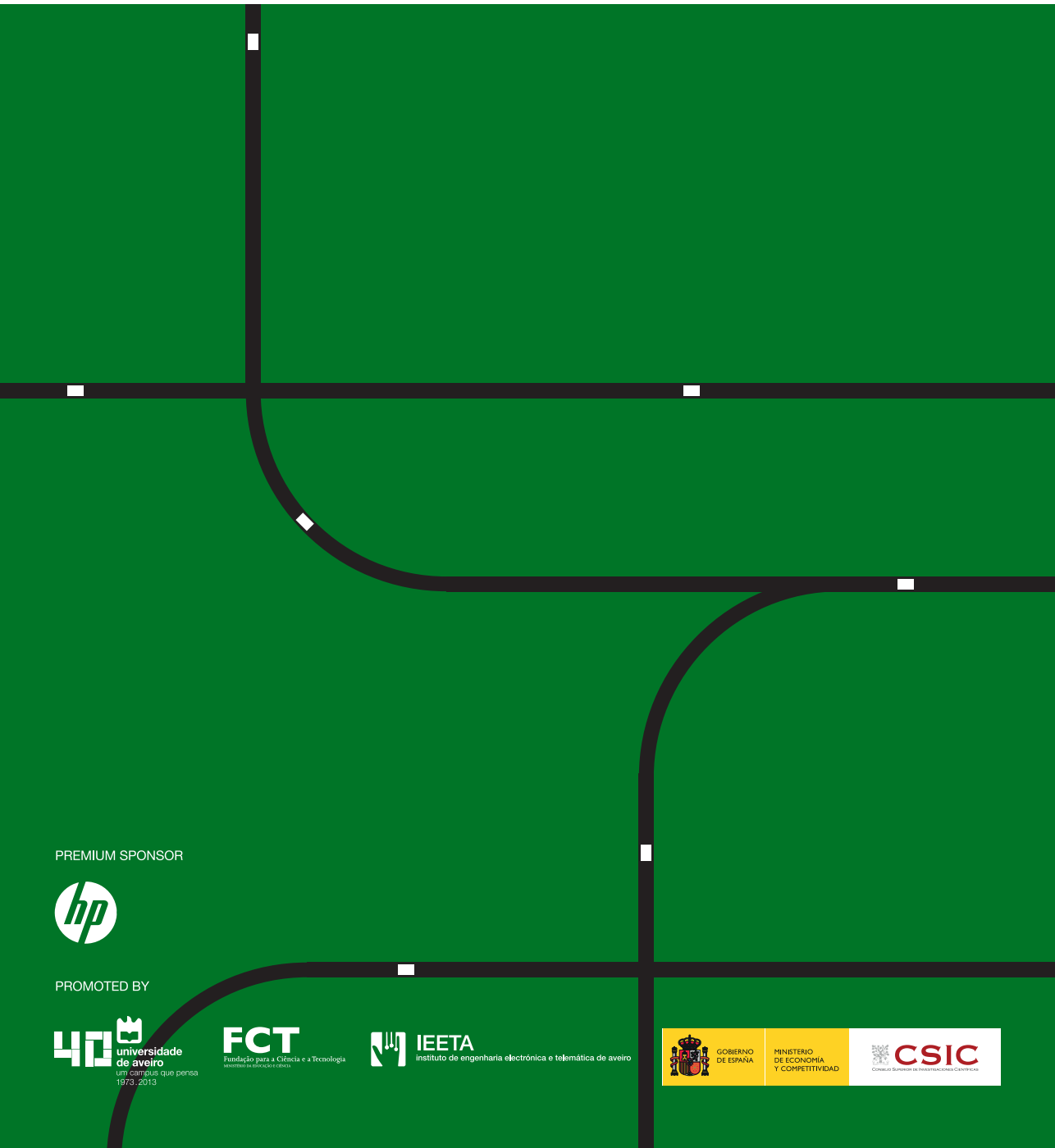
This work was supported by the IMED (Development of Algorithms for Medical Image Analysis) research collaboration project between INEGI, FMUP-HSJ, IEETA-UA (Portugal) and CETA-CIEMAT (Spain). The four institutions express their gratitude for the support of the European Regional Development Fund. Also, this work was partially supported by the Cloud Thinking project (CENTRO-07-ST24-FEDER-002031), co-funded by QREN, “Mais Centro” program.

References

1. Peteiro-Barral, D. and B. Guijarro-Berdiñas, *A survey of methods for distributed machine learning.* Progress in Artificial Intelligence, 2013. **2**(1): p. 1-11.
2. Rubin, G.D., MDCT and Data Explosion: Current Technologies and Directions for Future Development in Managing the Information Overload, in Multidetector-Row Computed Tomography. 2005, Springer. p. 113-120.
3. Schmidt, C.W., *Data explosion: bringing order to chaos with bioinformatics.* Environmental health perspectives, 2003. **111**(6): p. A340.
4. Ramos-Pollán, R., M. Guevara-López, and E. Oliveira, *A Software Framework for Building Biomedical Machine Learning Classifiers through Grid Computing Resources.* Journal of Medical Systems, 2012. **36**(4): p. 2245-2257.
5. Philbin, J., F. Prior, and P. Nagy, *Will the next generation of PACS be sitting on a cloud?* Journal of Digital Imaging, 2011. **24**(2): p. 179-183.
6. Viana-Ferreira, C., C. Costa, and J.L. Oliveira. Dicoogle relay-a cloud communications bridge for medical imaging. in Computer-Based Medical Systems (CBMS), 2012 25th International Symposium on. 2012. IEEE.
7. Ramos-Pollán, R., et al., Discovering mammography-based machine learning classifiers for breast cancer diagnosis. Journal of medical systems, 2012. **36**(4): p. 2259-2269.
8. Conradsen, I., et al., Automatic multi-modal intelligent seizure acquisition (MISA) system for detection of motor seizures from electromyographic data and motion data. Computer Methods and Programs in Biomedicine, 2012. **107**(2): p. 97-110.
9. Keserci, B. and H. Yoshida, Computerized detection of pulmonary nodules in chest radiographs based on morphological features and wavelet snake model. Medical Image Analysis, 2002. **6**(4): p. 431-447.
10. Bengio, Y., A. Courville, and P. Vincent, Representation learning: A review and new perspectives. 2013.
11. Karaçalı, B., *Quasi-supervised learning for biomedical data analysis.* Pattern Recognition, 2010. **43**(10): p. 3674-3682.
12. Brown, A. and G. Wilson, The Architecture of Open Source Applications, Volume II. 2012: CreativeCommons.
13. López, M.A.G., et al., BCDR: A BREAST CANCER DIGITAL REPOSITORY, in 15th International Conference on Experimental Mechanics. 2012.
14. Geilen, M., *Synchronous dataflow scenarios.* ACM Trans. Embed. Comput. Syst., 2011. **10**(2): p. 1-31.
15. Chang, E.Y., et al., *Psvm: Parallelizing support vector machines on distributed computers.* Advances in Neural Information Processing Systems, 2007. **20**: p. 16.

ISBN 978-84-9048-246-9

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
EDITORIAL



PREMIUM SPONSOR

PROMOTED BY

universidade de aveiro
um campus que pensa
1973_2013

FCT
Fundação para a Ciência e a Tecnologia

IEETA
instituto de engenharia electrónica e telemática de aveiro

GOBIERNO DE ESPAÑA
MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD
CSIC
Consejo Superior de Investigaciones Científicas