An Energy Management System for Cluster Infrastructures

Carlos de Alfonso[a], Miguel Caballer[a,*], Fernando Alvarruiz[a], Vicente Hernández[a]

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

Abstract

This paper presents a general energy management system for High Performance Computing (HPC) clusters and cloud infrastructures that powers off cluster nodes when they are not being used, and conversely powers them on when they are needed. This system can be integrated with different HPC cluster middleware, such as Batch-Queuing Systems or Cloud Management Systems, and can also use different mechanisms for powering on and off the computing nodes. The presented system makes it possible to implement different energy-saving policies depending on the priorities and particularities of the cluster. It also provides a hook system to extend the functionality, and a sensor system in order to take into account environmental information.

The paper describes the successful integration of the system proposed with some popular Batch-Queuing Systems, and also with some Cloud Management middlewares, presenting two real use-cases that show significant energy/costs savings of 27% and 17%.

Keywords: HPC, green computing, dynamic power management, cloud management systems, batch-queuing systems

1. Introduction

One of the challenges arising from the use of HPC clusters is reducing their power consumption. This problem is especially important in clusters that are underutilized, either because they form part of large scale distributed systems (grids or clouds) [1], where load can have important variations, or because the clusters have been in production for several years and their usage has decreased in favour of other more modern systems. However, in the last years there have been advances in the energetic efficiency of HPC clusters, which have come as a result of two different approaches: Static Power Management (SPM) techniques that use low-power energy-efficient hardware to reduce energy usage, and Dynamic Power Management (DPM) techniques that are based on the knowledge of resource utilization and application workloads to reduce energy usage [2].

In the case of SPM there are efforts pursuing higher efficiency for power sources [3], [4], which is usually lower than 80%. The hardware designers are also introducing new types of memory to increase the efficiency and reduce consumption. New technologies,

*Corresponding author
Email address: `micafer1@upv.es` (Miguel Caballer)

such as Solid State Drives (SSD), are also being adopted for disks in order to reduce the energy consumed by mechanical parts, which accounts for up to 65% of the total amount of energy consumed by a computer [5]. Dynamic Voltage and Frequency Scaling (DVFS) is an efficient technology to control the processor power consumption [6].

The DPM approach takes advantage of the fact that many computing nodes that are part of infrastructures such as clusters are usually powered on even when they are not being used (e.g. the workload is low, some computing nodes are not suitable for current calculations, there are reserved nodes for priority users, etc.). These clusters are usually dimensioned for peaks of workload that are not the most common situation. Therefore, energy can be saved by putting the idle nodes into power-saving mode (e.g. turning nodes off). There are different mechanisms that may be used to power on or off the nodes depending on the workload, that go from managing power by hand (e.g. powering off part of the nodes when they are not going to be used for a period of time) to introducing automated mechanisms into the job submission tools (e.g. monitoring a queue of jobs and powering off the computing nodes when the queue is empty).

A further step in automating the power management of nodes is to use energy-aware scheduling/allocation algorithms for assigning resources to jobs. For instance, schedulers may try to use the minimum number of computing nodes, in order to enable energy reduction by powering off the idle nodes. However, implementing an energy-aware allocation method in existing clusters of an organization is a difficult task, since it is necessary to modify the scheduling code of the resource management middleware. Even if the source code is available, modifying it can be a complex task, and maintaining modification through new releases of the middleware makes it even worse.

In this context, this paper presents CLUES (Cluster Energy Saving System), which is a general power management tool for computer clusters that can work in connection with different resource management middleware by means of easy-to-develop connectors. Thus, the tool is an effective way to implement power management policies in existing clusters, without having to modify the underlying control middleware. It could even be used in multipurpose clusters where different management middlewares coexist, thus enabling cluster-wide energy management policies for those situations. While CLUES was previously introduced in [7], this paper describes the tool in more depth, and provides details about the connectors that are currently available for the interaction with existing cluster management systems. It also presents some other features such as a hook system and a sensor system. A discussion is provided about the algorithm that is considered in CLUES for the power management of the nodes, and how it behaves when it is applied to real computing infrastructures that are currently under production.

The remainder of the paper is structured as follows. First, section 2 presents the general power management approach followed, and section 3 analyzes related work. Then, section 4 presents the architecture of CLUES and describes all its components. Section 5 discusses the features of CLUES to support more than one LRMS coexisting in the same cluster. Section 6 presents an extended analysis of results to demonstrate the proper interaction of CLUES with the underlying system. Finally, section 7 provides conclusions and points to future work.

2. Power management approach

Current clusters are usually managed by a Batch-Queuing System (BQS) or, in the case of Cloud Computing, a Cloud Management System (CMS). From now on this middleware will be referred in general as Local Resource Management System (LRMS) or resource manager. Examples of BQS are Torque/PBS[1], SLURM[2], Son of GE[3]. Examples of CMS are OpenNebula[4], OpenStack[5] or CloudStack[6].

There are two alternatives to provide an energy saving mechanism based on powering off idle nodes: (a) modify the LRMS scheduler, or (b) treat the scheduler as a black box (BB) and connect it to some energy saving system that powers nodes on/off as needed.

Modifying the scheduler may achieve better results, but presents the disadvantage that it requires the creation of a modified version of the original scheduler, and the new versions released by the developers of the LRMS will also need new modifications. Moreover, the power schedule mechanism would be tied to the specific LRMS.

On the other side, a BB approach implies that the LRMS must contact the energy saving system to provision the resources needed by the jobs. It requires some degree of coordination between the job scheduler and the energy saving system, i.e. the energy saving system should not power off a node if that node is useful from the point of view of the scheduler, and conversely, a node that is not useful from the point of view of the scheduler should be powered off to save energy. A BB approach may not provide the best results because the energy saving system does not have the whole information about the workload, and does not control in which nodes the jobs are allocated. However, decoupling the scheduling of jobs and the decision of suspending or restoring nodes eases the incorporation of energy-saving policies in production clusters, since there is no need to modify the resource manager.

This paper considers a BB approach, where the resource manager scheduler is connected to an external energy saving system that powers nodes on/off.

3. Related Work

In the last years, many efforts have focused on energy-aware allocation of tasks in clusters, both for virtualized and non-virtualized environments. For instance, [1] presents a method to reduce power in large-scale distributed systems by switching nodes on and off according to the load. The approach considers the possibility of reserving resources in advance, and assumes that the duration of a job (or an estimate of it) is provided by the user when submitting the job. The system interacts with the user that submits a job, suggesting job starting times that are most suitable for energy reduction. This work is actually a booking system for computing nodes but not an energy saver. It decides whether a cluster can be powered off because it is not reserved. It does not integrate with the LRMS and therefore it is not suitable for interactive systems.

[1] http://www.clusterresources.com/products/torque
[2] https://computing.llnl.gov/linux/slurm
[3] https://arc.liv.ac.uk/trac/SGE
[4] http://www.opennebula.org
[5] http://www.openstack.org
[6] http://cloudstack.org

[8] presents an approach for virtualized data centres which is based on workload consolidation using virtualization, combined with turning off idle servers. The system uses machine learning in order to predict the consequences of different possible allocations for each job, in terms of performance and energy. It then decides task placing and reallocation in order to concentrate jobs in a reduced number of nodes without degrading performance. The paper deals with task placement but not with infrastructure management. It does not consider integration with the LRMS, and it also assumes that the user provides information on the job duration.

[9] and [10] also deal with the problem of resource allocation in virtualized clusters, considering workload consolidation in order to be able to switch off idle machines, while at the same time reducing the impact on the system performance. The approach uses heuristics based on multicapacity bin packing over memory and CPU load. [6] considers a power-aware scheduling algorithm for DVFS-enabled clusters, where processor frequencies are scaled down in order to minimize power consumed without substantially increasing execution times. [11] describes an approach to load balance Virtual Machine (VM) provisioning across different servers to save energy and to maintain the performance of the system. The underlying idea of such technique is to try to reduce energy consumption even if nodes cannot be idle. [12] explores the combination of using DVFS and putting idle servers into low-power mode, but a workload profiling phase is needed in order to determine the optimal power configuration.

All of the reviewed results are related to the placement of the jobs and virtual machines (VMs), with the idea of either packing the jobs to get some idle nodes, or altering processor voltage to get less power consumption. However, they do not describe how to manage the idle computing elements. According to [13], one important research topic for getting energy efficiency by applying DPM techniques is to schedule powering on and off computer's components (the whole server in most cases) to adapt to the workload. The survey [2] also explains some DPM works from other authors that would get idle resources, but the reviewed works usually assume that those idle resources are powered on or off automatically and do not consider any scheduling strategy. Most of them are basically job schedulers that would substitute the existing schedulers or cluster management middlewares and would obviously modify the way that users interact with them.

There are many other scientific works exploring this area. However, from the point of view of system administrators, there are not many available tools to implement green policies in clusters. In the case of BQS schedulers, MOAB (which is the Enterprise version of Maui) introduces some features to pack workload and to place idle servers in power-saving modes [14]. The latest versions of SLURM introduced the ability to change CPU frequency and voltage in order to save energy. However, these solutions are of course tied to a particular BQS. Since the choice of BQS is conditioned by many factors, administrators may find that the most suitable BQS does not take into account energy saving mechanisms. In the case of cloud middleware, Convirt 2.0 Enterprise Edition introduces scheduling policies to consolidate VMs to enable the operation of the datacenter in power saving mode, but it does not provide tools to automatically power off idle nodes. VMWare vCenter includes tools to power off hosts when they are not needed. Other cloud middleware do not take into account power consumption.

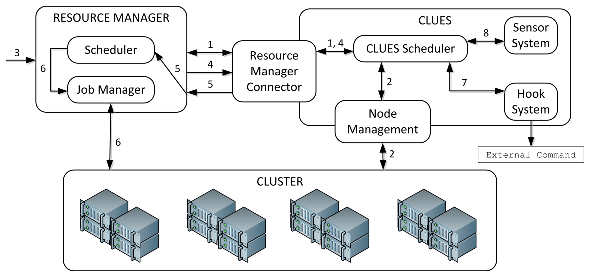

Figure 1: Architecture of the CLUES system.

4. System description

The purpose of the system proposed in this paper is to provide energy saving mechanisms for a computer cluster, by powering off idle nodes, and conversely powering on nodes when required. The system is able to interoperate with different resource management middleware by means of a plug-in based architecture. Using this approach, energy saving can be easily integrated with LRMS, and also with clusters of generic applications such as Web Servers or emerging Platform as a Service cloud systems. The design goals of the system are:

- It must be unobtrusive. From the point of view of the user, the way job submission or VM launching is done should not be altered by the use of CLUES.
- No changes to the underlying LRMS should be necessary to use the system, unless the developer wants to implement specific features or tighten the coordination between the job scheduling and the power management.
- It should be possible to use different mechanisms for switching on/off the nodes, e.g. mechanisms such as Wake-on-Lan (WOL), Power Device Units (PDU), Intelligent Platform Management Interface (IPMI) or infrastructure-specific mechanisms.
- The system should be easy to extend, e.g. adding the capability to use another LRMS, or adding another mechanism to switch on/off nodes.

As depicted in Figure 1, the system consists of a *scheduling component*, a set of one or more *resource manager connectors*, a set of *node management connectors*, and the *hook* and *sensor* subsystems.

The CLUES scheduler uses a connector to periodically ask the resource manager for information on the cluster state (label 1 in the figure). Based on this information, the scheduler determines if new nodes must be switched on, or if there are nodes that can be switched off, and acts consequently (2). When a job is submitted to the resource manager (3), a request for nodes is made to CLUES by means of the resource manager connector (4). When CLUES finishes processing this request, the job is actually submitted to the resource manager (5), where it will be processed by the scheduler and finally sent to the cluster for execution (6). There are two more components that are called periodically by the CLUES scheduler: the hook system, that enables to perform user defined actions when an event happens (7), and the sensor system, that enables access to a set of environmental values to be stored in the scheduler (8).

The *CLUES scheduler* is the main component, and is described in the following section. The *resource manager connectors* provide a uniform way to interact with different LRMS. This mechanism makes it easy to extend the system so as to consider additional resource managers, by writing the corresponding connectors.

The *node management connectors* are responsible for switching on/off the cluster nodes. The method to switch nodes on and off will be different depending on the particular cluster, e.g. WOL can be used for switching on, and a remote "poweroff" command can be used for ordered switching off, or PDUs can be used for both switching on and off. Additionally, in some cases the underlying middleware must be informed when a node is powered on/off, to activate or deactivate the node in the resource manager. By providing several node management connectors, these different situations can be accommodated. Currently, connectors have been developed for three different mechanisms: WOL, IPMI and a proprietary software to manage PDUs used in IBM clusters.

Note that CLUES intercepts any incoming job and retains it while trying to provide resources for it. Once this has been done, the job is released to the LRMS. Importantly, the jobs are released following a FIFO (First In First Out) strategy, therefore preserving the order in which they are taken into account for its execution. Another possible approach would be not to intercept the jobs at all, but instead make periodic inspections of the LRMS queue in order to detect if there is a need to power on additional nodes.

The approach of intercepting the job enables to prepare the context for the LRMS, instead of modifying it once the job has been scheduled under a state that is going to be changed by CLUES. The idea is that when the jobs arrives to the LRMS all the resources needed are already powered on. It also presents the advantage that it provides a faster response, because the need for resources is detected at the moment the job arrives. A disadvantage is that it can introduce small delays in the start of some jobs, if they are submitted shortly after other less-priority jobs which require nodes to be powered on.

In any of the two approaches, CLUES might try to bootstrap a node for a job that, according to the LRMS policies, does not have the right to execute, e.g. because the user has exceeded the execution quota. This can reduce the effectiveness of the power saving strategy, since there might be more powered-on nodes than necessary. However, this cannot be avoided with a BB approach because it is unaware of the LRMS policies.

4.1. CLUES Scheduler

The CLUES scheduler is the component in charge of: (i) processing requests for available resources and powering-on nodes if necessary; and (ii) powering-off idle nodes. To carry out these tasks it performs the following procedure:

1. When a new request for nodes arrives, the request is evaluated in order to determine if new nodes must be powered on for the request. If this is the case, the appropriate actions are taken. CLUES has a synchronous behavior, blocking the request and appending it to a list of pending requests while the necessary nodes become ready.
2. Periodically, the state of nodes is updated according to the information provided by the resource manager connectors. After each update, the queue of pending requests is examined. Each request is evaluated again and the necessary power-on actions are taken. If the request is at the head of the queue and the corresponding nodes

are ready, it is removed from the queue and released so that the associated job can proceed to its execution. Note that a request can be released either because there are enough free nodes, or because there are no more nodes that can be switched on. In both cases, no further action can be done for the request.
In addition to examine the queue of pending requests, idle nodes are detected and they are powered off if the inactivity time is larger than a predefined value.

Different policies can be used in order to determine if new nodes must be powered on to serve a request, each of them producing a different effect on desirable objectives: minimizing the power consumption, minimizing the impact on the users, minimizing the heat dissipation, etc. The selection of the policy is an important decision to obtain the desired behavior of the cluster. CLUES implements a set of basic policies:

- The most simple one is to switch on all the nodes of the cluster when a job arrives to the system. This is a coarse strategy but it is very simple to implement and can obtain good results with some specific workloads (e.g. large waves of jobs and long inactivity periods) and with clusters where powering off some nodes may affect the network topology and the connectivity of the remaining nodes.
- Switch on the minimum number of nodes to fulfill the request needs. This strategy enables minimum power consumption, but may increase the waiting time of incoming jobs.
- Switch on the nodes using a block size: instead of powering on the exact number of needed nodes, this strategy powers on an extra number of nodes, thus providing extra spare idle nodes that may prevent subsequent requests from waiting.

Obtaining the number of nodes available for a request
In order to apply any of the last two strategies, the scheduler must obtain the number of nodes available for a request, taking into account the following considerations:

- The process of booting up a node takes some time, during which the request will be queued until the nodes are ready. This means that when a new request arrives, there can be previous pending requests, and there can be nodes booting up.
- A node can be shared by more than one job, e.g. a node typically contains several cores, so it is possible to assign some of the cores to a job and other cores to another job. Consequently, a node is considered to contain a number of processing units or "slots" (e.g. cores), and a request asks for "virtual nodes", which are groups of slots in the same physical node.
- Requests can be made for nodes meeting certain conditions, e.g. nodes belonging to a particular batch queue, nodes with a given minimum amount of memory, installed software or configuration.

In order to determine the number of available nodes, the scheduler needs to use information about the current request, such as the number of requested virtual nodes (rv), number of slots per virtual node (sv), and possible conditions on the nodes. It also needs information about previous requests that are waiting for resources to become available, such as the total number of requested slots (trs). Finally, information on the cluster nodes is also necessary. For each node i, the scheduler needs to know its state (e.g. on, off, booting, failed...), number of free slots (fs_i) and total slots (s_i), and other information (e.g. amount of free memory, administrator-defined tags...).

Based on this information, the scheduler performs two steps:

1. Determine the number of virtual nodes that are usable by the current request, without taking into account previous requests. A virtual node is considered usable if it is located in a node that satisfies the conditions of the request, and its slots are not in use. The process followed can be seen in algorithm 1, which obtains the number of usable virtual nodes in powered-on nodes (uv_{on}) and in booting nodes (uv_{bt}). The algorithm goes through all the nodes that satisfy the conditions of the request, and for each of them the number of virtual nodes provided is obtained and accumulated (e.g. if a node has 5 free slots and the request asks for virtual nodes of 2 slots, 2 virtual nodes are provided).

Algorithm 1 Computing the number of usable virtual nodes.

```
// $uv_{on}$: number of usable virtual nodes in powered-on nodes.
// $uv_{bt}$: number of usable virtual nodes in booting nodes.
// $s_i$: total number of slots in node $i$
// $fs_i$: number of free slots in node $i$
// $sv$: number of slots per virtual node of current request
$uv_{on} \leftarrow 0$; $uv_{bt} \leftarrow 0$
for all node $i$ that matches current request do
  if $state_i = on$ then
    $uv_{on} \leftarrow uv_{on} + \lfloor fs_i/sv \rfloor$
  else if $state_i = booting$ then
    $uv_{bt} \leftarrow uv_{bt} + \lfloor s_i/sv \rfloor$
  end if
end for
```

2. Correct these numbers of usable virtual nodes, by taking into account previous requests, that may take some of the slots of the usable virtual nodes. Since the allocation of the requests to particular nodes/slots is decided later at the LRMS level, the corrections are based only on estimations. The resulting numbers are referred to as uv'_{on} for powered-on nodes and uv'_{bt} for booting nodes. Best-case estimates are derived, based on simplifying assumptions. First, the details of previous requests are not taken into account, and only the total number of previously requested slots (trs) is used. Second, it is assumed that these slots will be placed preferably in nodes that do not satisfy the conditions of the request. Thus, previous requests will produce minimum disturbance. According to this, uv'_{on} is:

$$uv'_{on} = min(max(0, \lfloor \frac{tfs - trs}{sv} \rfloor), uv_{on}) \tag{1}$$

where tfs is the total number of free slots.

If $tfs \geq trs$, uv_{bt} is not corrected ($uv'_{bt} = uv_{bt}$). Otherwise, the previous requests may also use booting nodes, and uv_{bt} is corrected accordingly:

$$uv'_{bt} = min(max(0, \lfloor \frac{tfs + tbs - trs}{sv} \rfloor), uv_{bt}) \tag{2}$$

where tbs is the total number of slots in booting nodes. Once uv'_{on} and uv'_{bt} have been obtained, the sum of them is computed to get the estimated number of virtual nodes available for the request.

An example. In order to illustrate the procedure described above, a cluster is considered with 20 nodes of 4 slots each. There are 2 completely free nodes, a node with one free slot and a node with 3 free slots. 2 other nodes are booting and the rest are switched off. There is a pending request which asked for 7 virtual nodes of 2 slots each, and in this context a new request arrives for 4 virtual nodes of 2 slots each. According to algorithm 1, the number of usable virtual nodes are computed, obtaining $uv_{on} = 5$ and $uv_{bt} = 4$. Then, pending requests are taken into account as explained in step 2. Taking into account that $tfs = 12$, $trs = 14$ and $tbs = 8$, equations (1) and (2) yield:

$$uv'_{on} = min(max(0, (12 - 14)/2), 5) = 0$$
$$uv'_{bt} = min(max(0, (12 + 8 - 14)/2), 4) = 3$$

This shows that the current request can get only 3 virtual nodes from currently booting nodes. Since it needs 4 virtual nodes, more nodes have to be powered on.

Selecting the nodes to be powered on

The next step is to select which nodes, of the list of nodes that match the request, will be switched on. There are also different strategies:

- Homogeneous Clusters: In this case, basic strategies such as selecting the nodes using a fixed order or a random algorithm are good solutions, as all the nodes provide the same features to all the jobs.

- Heterogeneous Clusters: In this case, more advanced strategies can provide advantages by selecting the nodes according to different node features: performance, power consumption, heat dissipation, etc. It is also possible to use a combination of some factors, e.g. selecting the nodes with the best ratio of performance / power consumption. In order to realize these strategies, CLUES must obtain additional information about the nodes (e.g. performance or power consumption). Currently, the information must be provided by the system administrator using a set of static files, but CLUES is prepared to use in the future some sensors or systems such as IPMI that can provide the information automatically.

Powering off idle nodes

Another task to be done after each state update is to detect idle nodes that can be switched off. The time of inactivity used to power off the nodes must be specified by the system administrator. It is important to correctly select this time to obtain good results with CLUES. Using a short time may reduce the power consumption, but it can also increase the number of jobs having to wait, and the number of power on/off operations. On the other hand, using a long time will produce opposite results.

Some other factors are also considered when powering off nodes. In some cases, because of the particularities of the hardware or the network topology, it is required that some of the nodes (or all of them) remain powered in order for the cluster to work properly (such as in the first cluster shown in the results evaluation section).

Re-evaluation of Jobs

A re-evaluation mechanism has been implemented, by means of which the queue of the LRMS is periodically inspected, identifying jobs that have remained queued for a specified amount of time. For each of those jobs, a request for nodes is sent again to be re-evaluated by CLUES. This mechanism is introduced to correct some possible undesirable conditions that arise when following a black-box approach (e.g. a node may be powered off while a job that has not enough nodes is in the queue). CLUES processes re-evaluation requests just like ordinary requests, checking the resources needed by the job and switching on nodes if necessary.

4.2. Resource Manager Connectors

The *resource manager connectors* provide a uniform way to interact with different LRMS. Each connector consists of two parts.

The first one is a monitoring system, that obtains information about the nodes of the LRMS and presents it in a uniform way. The monitoring system connectors are implemented as external executable files, which can be created using any programming language. The connectors get the information directly from the LRMS and publish it as a list of key-value pairs separated by semicolons, with one line for each node, e.g.

```
host=node1;state=down;total_slots=2;free_slots=2;keywords=ok;queues=sci
host=node2;state=free;total_slots=2;free_slots=1;keywords=ok;queues=sec,sci
...
```

There are only four mandatory fields: `host`, `state`, `total_slots` and `free_slots`. The rest of fields depend on the type of LRMS used. For example the `queues` field is used in the batch systems but not in the cloud ones.

The second part is a job interceptor, that comes into action whenever a new job is to be submitted to a LRMS. Before the job is actually submitted, the connector requests the necessary resources to the CLUES scheduler. When a response to the request is received, the job is submitted to the LRMS.

4.2.1. Batch system connectors

Torque/PBS and SGE, two of the most popular queue systems, have been considered here and a connector has been implemented for each of them.

As mentioned above, one of the functions of a connector is to provide information about the node states. In the case of PBS, this information is extracted by using the command `pbsnodes`, and in the case of SGE with the `qhost` command. In both cases, an option of the command is used in order to get the output in XML format, which can be parsed more easily.

The other function of the connector is to catch incoming job submissions, in order to request the corresponding resources to the CLUES scheduler. PBS provides a feature known as "job submission filter" (or "qsub wrapper") which is useful to intercept the

Hook Name	*Description*
POWEREDON, POWEREDOFF	Before powering on/off a node
POWEREDON UNEXPECTED, POWEREDOFF UNEXPECTED	When CLUES detects that a node has been powered on/off unexpectedly
MONITORING, MONITORED	Before or after the monitoring procedure
ENABLED, DISABLED	Once a node has been enabled or disabled
SENSOROVER, SENSORBELOW	The value of a sensor is over or below a threshold

Table 1: Hook types

submission of jobs. A similar feature called "job submission verifier" has been used in the case of SGE. By means of these features, a script can be specified to be run before the effective submission of a job into the queue system. In this case, the script must first determine relevant information of the job being submitted (such as the number of required virtual nodes, the number of slots per virtual node or the queue name), then send a request for nodes to CLUES, and wait for a response. When a response is obtained, job submission can proceed.

4.2.2. Cloud system connectors

In the case of CMS, OpenNebula and OpenStack connectors have been developed. The OpenNebula connector intercepts the creation of VMs by a mechanism provided by the middleware called "hooks". Such mechanism enables the execution of an application whenever a VM is created, and it is used to ask CLUES for working nodes. The OpenStack connector does not provide any similar feature, and it was necessary to modify one file of the middleware API to connect to CLUES.

The result is that each newly created VM is held while CLUES decides whether extra nodes should be powered on, and in such case, while the nodes are booted. If the VM were released before the node being ready to accept VMs, the scheduler might try to assign it to a working node that does not have enough resources. While the VM is retained, CLUES tries to make its best for provisioning resources.

The information about the hosts is extracted in both cases using the corresponding internal API for direct access. The main issue in the OpenNebula case is that the information provided about the hosts is not enough for the CLUES scheduler, regarding both the memory and the virtual CPUs booked: the internal information system tracks the number of VMs that are running in a particular host (but not the virtual CPUs), and the remaining free memory that is reported by the operating system (that considers swap memory as real memory). The workaround has been to extract the information from the description of the VMs.

4.3. Hook system

The hook system enables the extension of the functionality of CLUES without the need to modify the source code. It specifies user defined actions (e.g. custom scripts) to be executed before or after some event happens (eg. a node is being powered on). The user must provide an executable file that receives as a parameter some value related with the hook event. The events considered in the hook system are shown in table 1.

This system also covers the CLUES monitoring system, to enable tasks to be performed each time the CLUES scheduler monitors the system, or some measures to be taken when a particular state is detected, e.g a message can be sent to the system administrator when a node does not power on correctly or when it powers off unexpectedly.

4.4. Sensor System

Nowadays it is quite common for the clusters to be monitored using some kind of sensors to know some environmental parameters. Typical examples are the temperature or the humidity. In some cases these sensors are managed by a piece of software (e.g. Nagios) that can send notifications to the administrators to take corrective measures.

A sensor system has been included in CLUES, enabling access to environmental information, which can be used by the hook system to take automatic corrective measures. CLUES can call periodically a set of sensor plugins (typically scripts) that return a set of key-value pairs with the name of the parameter and the value measured by the sensor. These values are stored and it is possible to configure the system so that actions are taken whenever the value of any parameter is over or below a given threshold.

In particular, the hook system can be used in order to take an action when the parameter values are out of the specified limits. To implement this feature a new type of hooks has been added to the scheduler where the user must define an upper and/or lower limit for a measured value in the sensor system, and a command that must be executed when the “exception” happens. The executed command will receive as a parameter the string with the key-value pair obtained by the sensor system. Corrective measures could be e.g. powering off the idle nodes, or even powering off all the nodes, or, if a software provides the functionality, sending a signal to switch on or off the air cooling system.

5. Mixed cluster

It is not very frequent to have more than one LRMS coexisting in the same cluster. However, with the advent of cloud management systems, this option is not unreasonable. Additionally, in some clusters used for testing purposes, it makes sense to have two LRMSs installed, such as PBS and SGE. CLUES has been designed to support this kind of mixed clusters, making it possible to manage nodes shared by two or more LRMSs.

Although one node can be shared by different LRMSs, the number of slots must be divided among them. For example, if a node has 6 slots, one LRMS could be using 2 of them and another one could use 4. It is a task of the system administrator to configure the LRMSs properly.

The CLUES scheduler can manage a list of nodes included in each of the configured LRMSs, storing the state and the features provided by the different connectors. When processing an incoming request, the scheduler checks for available nodes only within the list of nodes of the LRMS corresponding to the request. When selecting the nodes to switch off, the scheduler must check the combined information about all the nodes to select only the nodes that are considered idle in all the LRMSs.

6. Results Evaluation

In a previous work [15] of the authors, an analysis was made of the jobs launched to the Torque/PBS LRMS of a HPC cluster, in order to have an estimate of the benefits

	Using CLUES			**Not using CLUES (est.)**			**Consumption per node**
	PCT	**kWh**	**€**[7]	**PCT**	**kWh**	**€**	**W**
N. Off	45.6%	350	32	0.0%	0	0	3
N. Idle	4.9%	1,624	148	50.4%	16,234	1,477	130.9
N. Used	49.6%	26,051	2,371	49.6%	26,051	2,371	205.4
Other	100%	10,296	973	100%	10,296	973	2,012
TOTAL		38,321	3,487		52,581	4,785	

Table 2: Cluster 1 power consumption and cost

of applying green computing techniques to that cluster. Now the first version of CLUES has been developed and it has been installed and working during seven months in two different clusters. During this time period an evaluation, using the current real workload of the clusters, has been made of the software behaviour, and of the real impact on the power consumption and on the cluster users. This evaluation was also useful to detect some aspects that were not initially considered but are important in a production version.

6.1. Cluster 1

The tests have been performed in a cluster composed of 51 bi-processor nodes with Intel Xeon CPUs at 2.80GHz, interconnected by a SCI network in a 10x5 2D torus topology. Each node has 2 GB of RAM memory. The front-end node is the access point to the cluster, and the other (50) are used as the working nodes. This cluster is configured with a NFS system that is exported by the front-end node and accessed by the computing nodes.

This cluster is used as a development and private testbed platform for parallel and sequential high performance applications . The cluster is typically used to execute both sequential and parallel CPU-intensive applications. During the seven months considered for the evaluation, the system was used normally, with a total of 20,497 jobs submitted. 41% of the jobs were parallel and used an average of 14 nodes. The average time per job was 14 hours, 41 minutes and 14 seconds.

A clamp meter was used to get the power consumption of each component of the whole rack, and the corresponding data are shown in the rightmost column of table 2. In particular, power consumption has been obtained for three different states of a cluster node: switched off ("N. off"), switched on but idle ("N. idle"), and fully used ("N. used"). Finally, the entry "Other" refers to the power consumption of the essential components of the cluster (front-end node, switches, KVMs) that are always on.

Based on the analysis made in [15], a period of inactivity of 2 hours was considered in order to switch off idle nodes. This period of time is the appropriate for the deployment of the use-case, but it should be adjusted according to the features of the actual deployment in which CLUES is used (the usage pattern of the cluster, the power consumption for the nodes, etc.). The fact that the SCI network has a 2D torus topology implies that a message from a node A to another node B can be routed through other intermediate

[7]Cost: 0.091 €/kWh. Data obtained from the Ministerio de Industria, Turismo y Comercio del Gobierno de España

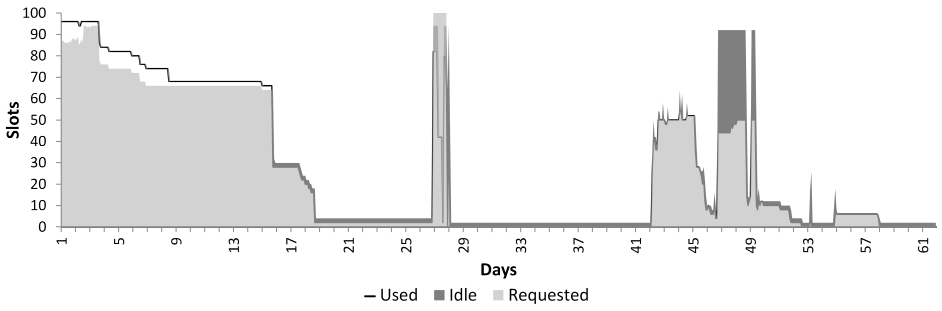

Figure 2: Evolution of the number of used, idle and requested slots in cluster 1

nodes. Thus, these intermediate nodes should not be powered off even if they are idle. In order to tackle this problem, the whole cluster is kept switched on whenever a parallel job (using more than one node) is running.

Figure 2 shows an evolution over the time of the number of requested slots in the LRMS, and of the number of used and idle slots in the cluster. In the figure, a slot is marked as used not only when a job is using it, but also when the slot is part of a node that has at least one used slot. In this case, "used" means the slot cannot be switched off. The vertical axis has been truncated to 100 to remove peaks of requested slots that would make it difficult to see the figure. The period of time considered in the figure has been reduced to the first two months, also for the sake of clarity. In this first case there is a clear correlation between the number of requested slots in the system and the number of nodes switched on by CLUES. The figure shows that one node is always powered on, because this node had some problems with the WOL configuration. Near the end of the two-month period (about days 47 - 50) there is a peak where the whole cluster is switched on with a reduced number of requested slots. This is produced by some parallel jobs requiring all the cluster to be switched on due to the commented network topology restrictions.

Table 2 shows the results of energy and money spent during the considered period, in both the cases of using CLUES and not using it. The left part of the table contains the data for the case of using CLUES. The first column (titled "PCT") represents the percentage of time a node spent on average in each state. The second column (titled "kWh") represents the total amount of energy consumed by the specified components of the cluster, expressed in kilowatt hours. Last column (titled "€") contains the amount of money dedicated to those components. The center part of the table corresponds to the estimation of the energy and money spent if the CLUES system had not been used, presenting the same columns as the left half. The energy consumption without CLUES has been estimated by changing all the accumulated time of nodes in "Off" state to the "Idle" state, because without CLUES these nodes would have never been switched off.

Table 2 shows that the total amount of energy saved is 14,260 kWh, which represents 27.1% of the total amount of energy, but also means saving 1,297 €.

On the user impact side, an analysis has been made of the number of jobs that needed to wait to access the resources. During all the period, 268 jobs had to wait for some node to be switched on (1.31% of total jobs). The average waiting time for these jobs was 1 minute and 40 seconds. This is a short enough waiting time, considering that the average

	Using CLUES			**Not using CLUES (est.)**			**Consumption per node**
	PCT	**kWh**	**€**	**PCT**	**kWh**	**€**	**W**
N. Off	39.3%	181	16	0.0%	0	0	9
N. Idle	7.8%	363	33	47.1%	2,853	260	65
N. Used	52.9%	7,407	674	52.9%	7,407	674	187
Other	100%	3,070	279	100%	3,070	279	600
TOTAL		11,021	1,003		13,331	1,213	

Table 3: Cluster 2 power consumption and cost

time per job exceeded 14 hours.

Another important issue is related with the number of switch-on/off cycles performed in the cluster nodes. These operations can damage the hardware (mainly the disk drives) and may cause consumption peaks that could increase the total power consumption. During the evaluated period, an average of 38 switch-on/off cycles were performed for each node, with a maximum of 54 cycles. This means that a node completes a switch-on/off cycle once every 6 days on average, with a maximum of once every 4 days.

6.2. Cluster 2

The CLUES system has also been tested in a cluster composed of an M1000e blade server chassis with 6 Dell M610 and 3 Dell M910 nodes. Each M610 node has two quad-core Intel Xeon E5620 processors, making a total of 8 cores and 16 GB of RAM per node. The M910 node has four quad-core Intel Xeon E7520 processors, with a total of 16 cores and 64 GB of RAM per node. The cluster uses Torque/PBS and a NFS system is exported by the front-end node and accessed by the computing nodes.

This cluster is used in a production grid environment, as one of the computing nodes of the Spanish National Grid Initiative[8] in the European Grid Infrastructure[9]. The cluster is typically used to execute high throughput applications launching sequential jobs. There is a wide range of different applications with different behavior and requirements in terms of CPU, memory and I/O access patterns. In particular, the workload of the system during the evaluation period was composed of a total of 107,197 jobs, 13% of which were parallel and used an average of 2.28 nodes. The average time per job was 2 hours, 39 minutes and 20 seconds. In contrast to the previous case, this cluster has no network restriction and the nodes can be switched on individually. A time of 30 minutes has been selected as the time of inactivity to power off the nodes.

Figure 3 shows an evolution over the time of the number of requested slots and the number of used and idle slots in the cluster. As in the previous case, the "used" state represents the slots that cannot be switched off. The vertical axis has been truncated to 200 to remove peaks of requested slots that would make it difficult to see the figure. As in the previous case, the period of time for the figure is two months. The correlation between the number of requested slots and the number of switched-on nodes is not as clear as in the previous case. The main reason is that there are heterogeneous multicore

[8]http://www.es-ngi.es
[9]http://www.egi.eu

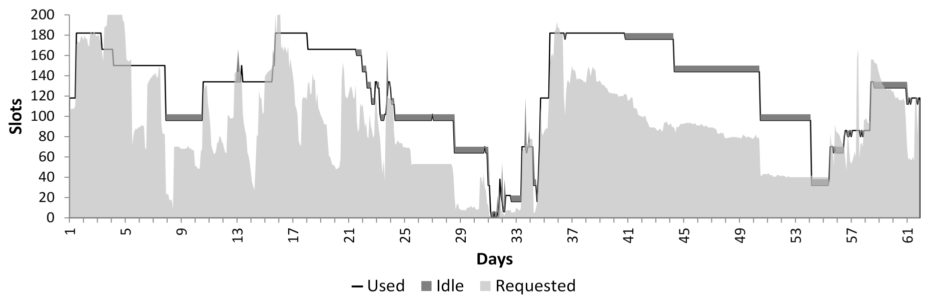

Figure 3: Evolution of the number of used, idle and requested slots in cluster 2

nodes (with 16 or 32 slots per node), and the job distribution among all the nodes depends on many factors: the LRMS scheduler, the finalization of the jobs, the arrival of new jobs, etc. The LRMS can be configured to pack the jobs in the minimum number of nodes, but other factors cannot be controlled. It is possible, for instance, that only 9 jobs requesting 1 slot each, end up keeping all the nodes switched on.

There are also some restrictions in the LRMS such as a maximum of 40 running slots for each user group. This issue explains the behavior of the system about the days 3 - 5 and 55 - 59, where the number of requested slots is bigger than the number of used slots and no new nodes are switched on.

Table 3 shows the economic and energetic saving obtained by using the green computing software. The left part of the table shows the power consumption using CLUES, and the center part shows an estimation of the power consumption without CLUES. The M1000e chassis has a complete set of energy management tools to monitor the power consumption of the whole system and the individual blades. These tools have been used to obtain the power consumption to perform this study. The rightmost column of the table shows the power consumption of one blade system in different states: switched off, switched on but idle, and fully used, with the maximum number of jobs running. In this case the "Other" row corresponds to the chassis.

The estimated economic saving is 210€, which means a reduction of 17.3% of the total expenses. Unsurprisingly, the impact of the application of green measures in this cluster is lower than in the previous case. The main reason is that, as a production node of the EGI infrastructure, the cluster is periodically receiving jobs in order to monitor the status of the system. These monitoring jobs cause that at least 2 of the 9 nodes are always on. Other reasons are the important power consumption of the chassis compared to that of the 9 nodes, and the fact that the number of cores per node in this case is larger, which makes it easier for the nodes to be only partially used.

On the user impact side, 2.9% of the jobs had to wait for some node to be switched on, with an average waiting time of 1'54". These are short enough values, considering that the average time per job exceeded 2 hours.

The average number of switch-on/off cycles for a node was 38, and 62 for the node with the maximum number of these operations. It means that a node completes a cycle once every 6 days, with a maximum of once every 4 days, that are very low ratios.

7. Conclusion and Future Jobs

The proposed CLUES tool is an energy manager for both HPC clusters and cloud infrastructures, that is able to power off the nodes when they are not being used, and power them on when they are needed. CLUES considers the underlying LRMS as a BB. The advantage of this approach is that it can be integrated with different resource management middleware, without needing any modification of that middleware. Because of this flexibility, it can be used both for HPC clusters and for cloud infrastructures. It can also be used with multipurpose clusters where different management middleware coexist, thus enabling cluster-wide energy management policies. Additionally, it considers different mechanisms for powering on and off the cluster nodes. The performance of CLUES is shown with two real use-cases that show significant energy and cost savings of 27% and 17%.

Future directions of work include the introduction of modifications to the CLUES scheduler, the use of alternative energy saving mechanisms such as DVFS, or the use of other heuristic methods which may take into account prediction of performance and energy consumption. At the same time, the integration with other middlewares such as Eucalyptus or CloudStack is an ongoing work.

Another important issue to be considered in the future is the impact of CLUES in systems using some kind of parallel file system like Lustre, GFS, GlusterFS, etc. This kind of systems supports data replication, making it possible to switch off some nodes of the infrastructure without losing access to the data. Configuration issues imposed by this kind of systems must be analyzed, as well as the impact of switching off nodes on the data access performance.

Finally, CLUES also opens possibilities for research in the field of scheduling policies for powering on and off the working nodes in multi-purpose clusters governed by several coexisting middleware, with the aim of reducing energy consumption.

Acknowledgement

The authors wish to thank the financial support received from The Spanish Ministry of Science and Innovation to develop the project "CodeCloud", with reference TIN2010-17804.

References

[1] L. Lefèvre, A.-C. Orgerie, Towards energy aware reservation infrastructure for large-scale experimental distributed systems, Parallel Processing Letters 19 (03) (2009) 419–433.

[2] G. Valentini, W. Lassonde, S. Khan, N. Min-Allah, S. A. Madani, J. Li, L. Zhang, et al., An overview of energy efficiency techniques in cluster computing systems, Cluster Computing (2011) 1–13.

[3] S.-A. Liang, Low cost and high efficiency pc power supply design to meet 80 plus requirement, in: Industrial Technology, 2008. ICIT 2008. IEEE International Conference on, 2008, pp. 1 –6.

[4] U. Hölzle, B. Weihl, High-efficiency power supplies for home computers and servers, Tech. rep., Google, presented at the Intel Developer Forum, September, 2006. (2006).
URL http://services.google.com/blog_resources/PSU_white_paper.pdf

[5] T. Heath, B. Diniz, E. V. Carrera, W. Meira, Jr., R. Bianchini, Energy conservation in heterogeneous server clusters, in: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming, PPoPP '05, ACM, New York, NY, USA, 2005, pp. 186–195.

[6] G. von Laszewski, L. Wang, A. Younge, X. He, Power-aware scheduling of virtual machines in dvfs-enabled clusters, in: Cluster Computing and Workshops, 2009. CLUSTER '09. IEEE International Conference on, 2009, pp. 1 –10.

[7] F. Alvarruiz, C. de Alfonso, M. Caballer, V. Hern'ndez, An energy manager for high performance computer clusters, in: Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on, 2012, pp. 231 –238.

[8] J. L. Berral, I. n. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, J. Torres, Towards energy-aware scheduling in data centers using machine learning, in: Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking, e-Energy '10, ACM, New York, NY, USA, 2010, pp. 215–224.

[9] M. Stillwell, D. Schanzenbach, F. Vivien, H. Casanova, Resource allocation using virtual clusters, in: Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on, 2009, pp. 260 –267.

[10] D. Borgetto, G. Da Costa, J.-M. Pierson, A. Sayah, Energy-aware resource allocation, in: Grid Computing, 2009 10th IEEE/ACM International Conference on, 2009, pp. 183 –188.

[11] H. N. Van, F. Tran, J.-M. Menaud, Performance and power management for cloud infrastructures, in: Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, 2010, pp. 329 –336.

[12] M. M. Rafique, N. Ravi, S. Cadambi, A. R. Butt, S. Chakradhar, Power management for heterogeneous clusters: An experimental study, 2012 International Green Computing Conference (IGCC) 0 (2011) 1–8.

[13] A. Beloglazov, R. Buyya, Y. Lee, A. Zomaya, A taxonomy and survey of energy-efficient data centers and cloud computing systems, Advances in Computers 82 (2011) 47–111.

[14] Cluster Resources Inc, Green Computing Powered by Moab.
URL http://www.clusterresources.com/solutions/green-computing.php

[15] C. De Alfonso, M. Caballer, V. Hernandez, Efficient power management in high performance computer clusters, in: Proceedings of the 1st International Multi-Conference on Innovative Developments in ICT, INNOV 2010, 2010, pp. 39–44.

Carlos de Alfonso obtained the B.Sc. degree in Computer Science in 2000. Then he joined the Grid and High Performance Computing research group, at Universitat Politècnica de València (UPV). He has been part of several national, European and international projects and contracts in the fields of Grid and Cloud computing. He has oriented his activity to Cloud and Green Computing.

Miguel Caballer obtained the B.Sc. and M.Sc degree in Computer Science from the Universitat Politècnica de València (UPV), Spain, in 2000 and 2012. He is member of Grid and High Performance Computing research group (GRyCAP) of the Research institute of Instrumentation for Molecular Imaging (I3M) since 2001. His research interests include Parallel, Grid, Cloud and Green computing techniques.

Fernando Alvarruiz obtained the M.Sc. degree in Computer Science from the Universitat Politècnica de València (UPV), Spain, in 1995. In 1996 he joined the Grid and High Performance Computing research group, at UPV. Since 2003 he has been an assistant lecturer in UPV. His research interests include Green and Cloud computing, and application of High Performance Computing techniques in engineering.