Multi-facet approach to reduce energy consumption in clouds and grids: The GREEN-NET Framework[*]

Georges Da Costa
IRIT, Université Paul Sabatier
Toulouse, France
dacosta@irit.fr

Marcos Dias de Assunção
INRIA RESO - ÉNS
Université de Lyon
marcos.dias.de.assuncao@ens-lyon.fr

Jean-Patrick Gelas
INRIA RESO - ÉNS
Université de Lyon
jean-patrick.gelas@ens-lyon.fr

Yiannis Georgiou
MESCAL, Laboratoire ID-IMAG
Grenoble, France
Yiannis.Georgiou@imag.fr

Laurent Lefèvre
INRIA RESO - ÉNS
Université de Lyon
laurent.lefevre@inria.fr

Anne-Cécile Orgerie
INRIA RESO - ÉNS
Université de Lyon
annececile.orgerie@ens-lyon.fr

Jean-Marc Pierson
IRIT, Université Paul Sabatier
Toulouse, France
pierson@irit.fr

Olivier Richard
MESCAL, Laboratoire ID-IMAG
Grenoble, France
Olivier.Richard@imag.fr

Amal Sayah
IRIT, Université Paul Sabatier
Toulouse, France
Amal.Sayah@irit.fr

ABSTRACT

This paper presents an integrated framework for energy savings in large scale distributed systems such as grids and clouds. The framework comprises tools and mechanisms: to measure and log data about the energy consumed by resources; to present this information to users; to involve the users in decisions to reduce the energy consumed by their applications; and to enforce energy reduction decisions automatically while respecting the users' requirements and achieving the resource availability demanded by current services. Experiments demonstrate the energy savings achieved by the proposed mechanisms and explore trade-offs between energy efficiency and performance degradation.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications; D.2.8 [**Software Engineering**]: Metrics—*energy metrics*; C.4 [**Performance of systems**]: Design studies

[*]The experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several universities as well as other funding bodies (see https://www.grid5000.fr)

1. INTRODUCTION, RELATED WORK

In recent years, there has been an increasing interest to reduce the electricity bill of large-scale distributed systems. Funding agencies and industry players have attempted to raise awareness and develop solutions for reducing energy consumption, which could consequently decrease the carbon dioxide footprint of ICT at large and lead to more environmentally friendly systems.

Several promising approaches have been proposed in order to reduce the energy consumption of ICT, tackling the challenge from different perspectives. Part of existing work focuses on the physical infrastructure itself, aiming to use more energy efficient hardware (i.e. CPU, storage devices and power supplies) or enable innovative cooling systems that work at several levels, such as at the motherboard, the rack, and outside the box. Another set of solutions propose mechanisms to improve the software stack at different layers, including the operating system [23], the network protocols, the middleware, and the applications. Moreover, the community is organizing itself in forums such as the Green500 [8], the Green Grid [12], and the COST 804 European Action[1]. The majority of these efforts propose scattered, but nevertheless compatible, approaches.

We present in this paper an integrated framework, termed as GREEN-NET[2], that provides a top-down approach with three levels. Representing the information delivered to users, the first level aims to raise their awareness of energy con-

[1]http://www.cost804.org
[2]This research is supported by the GREEN-NET INRIA Cooperative Research Action: http://www.ens-lyon.fr/LIP/RESO/Projects/GREEN-NET/

sumption. The second level involves the users in decisions to trade performance for energy savings. The automatic behavior and adaptation of the distributed system to save energy represents the third level.

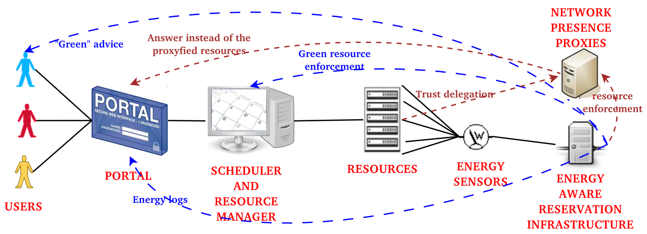

"Green" advice
Answer instead of the proxyfied resources
Green resource enforcement
NETWORK PRESENCE PROXIES
Trust delegation
resource enforcement
PORTAL
SECURE WEB INTERFACE - 1 PASSWORD
NAME:
PASSWORD:
USERS
PORTAL
Energy logs
SCHEDULER AND RESOURCE MANAGER
RESOURCES
ENERGY SENSORS
ENERGY AWARE RESERVATION INFRASTRUCTURE

Figure 1: The GREEN-NET framework.

The GREEN-NET framework shown in Figure 1 comprises three components :

- Energy Aware Resource Infrastructure (EARI): which obtains and logs information from distributed autonomic energy sensors. EARI enforces "Green" decisions using the scheduler and requests network presence of cluster resources to the Network Presence Proxies. Moreover, EARI gives "Green advices" to end users.
- Adapted Resource and Job Management System (OAR): which provides a specific module for automatic node shut down during cluster under-utilization periods, and a new PowerSaving job type to conserve energy of devices.
- Trust delegation component: when nodes are switched off due to energy reduction choices, this component allows the migration of basic services to Network Presence Proxies which must be trusted on different sites.

This work focuses on large scale distributed systems, more precisely grids and clouds, where a resource management system manages the resources available and selects from them to serve user jobs. We also consider that computers can be turned off and that their hardware is tunable. Existing work has described promising energy efficient mechanisms for resource and job management systems, such as Moab[3] and Slurm[4] [22]. However, to the best of our knowledge, results on the energy savings achieved by such mechanisms have not been reported.

The rest of this paper presents the proposed framework from three perspectives: informing users, involving them in making more energy efficient decisions, and automatically adapting the system to be more energy efficient. Section 2 discusses the tools and libraries used to provide users with information on energy consumption. Then, Section 3 shows how users can appropriately give hints to the resource manager to help it be more energy efficient, and demonstrates through experiments the advantages of energy reduction and the resulting performance loss. Section 4 details the methodology corroborated with experiments to reduce energy consumption automatically by migrating tasks and switching off nodes when they are not used for a given time, or when a prediction shows that they will not be utilised during the given period. Finally, Section 5 concludes the paper and presents future work.

[3] *http : //www.clusterresources.com/solutions/green − computing.php*
[4] *https : //computing.llnl.gov/linux/slurm/power_save.html*

2. INFORMING THE USERS

The best way to reduce energy consumption is not to consume it all. However, in order to make informed choices, users should be more aware of what exactly their applications consume. This requirement has led to developing new information systems for end users. For instance, Google with its PowerMeter [1] and Microsoft with its Hohm project [2] will allow users to view precisely their energy consumption at home and to receive advices on how to reduce the energy consumption. One key point of these two systems is that they will aggregate the data about users' energy usage over several months and even years, helping them to understand how their consumption evolves over time.

Although these systems target home users, the main point is valid for any highly energy consuming system: a good way to reduce consumption is to help users understand exactly how their applications are consuming energy. By placing this information into the spotlight, users can realize that some parts of their system, which might be optional and energy wasting, can be disabled, or that they can decide to optimize the system with energy consumption in mind.

2.1 The Hardware

To present information to users, the first step is to determine the detailed energy consumption of the system. Being generally quite coarse grained, current technology usually does not allow us to measure how much energy consume the individual components of a computer. The only possibility is to measure the consumption of a whole computer. Our experiments currently use two types of measurement appliances: the Hameg and the Omegawatt box.

The Hameg is an electronic laboratory power meter (HAMEG HM8115-2), which offers a precision of 0.5%. It can measure the energy consumed by only one element (usually a computer), but can provide all the relevant information (effective and reactive power, phase, etc). As this power meter is calibrated for laboratory use, it is utilized for verification of the other systems.

The Omegawatt box is a customized box produced by Omegawatt to measure the consumption of several nodes at the same time. The current system monitors 162 nodes and is deployed on three sites of Grid'5000, located at Grenoble, Lyon, and Toulouse. We use 6-port boxes in Grenoble and Toulouse, and 48-port boxes in Lyon. These systems are able to take a measurement per second. Each site contains a server responsible for logging the measurements using a dedicated library to capture and handle the data.

The interface with Hameg and Omegawatt equipments uses serial ports, but we are currently investigating alternative hardware (i.e. Plogg) that communicates via Bluetooth.

2.2 A Library to Interface with Energy Meters

Obtaining energy consumption information from several heterogeneous sensors such as those described above is a challenging task. We have chosen to develop a library that simplifies the measurements. As requirements this library had to be optimized for near "real time" data acquisition and be extensible to take into account future developments; hence, its implementation using the object-oriented language C++.

As first step we defined a sensor abstraction that describes the services expected from such equipments, regardless their real implementations. Using only the interface defined by this abstraction, a user can program all accesses to the sensors. From this abstraction we derive classes describing the types and behaviours of physical sensors; two types of sensors are currently interfaced, and a third type is under test.

Having completed the software layer to interface with sensors, we worked on developing: client-side applications that collect and log energy comsumption measurements from sensors at time intervals; and applications that access previously collected information. Each measurement contains a time-stamp that specifies when it was performed. On a server responsible for controlling a number of energy sensors, a thread is associated with each sensor to capture the energy consumption of the monitored devices at each second. An additional thread is responsible for responding to requests made by client applications. Clients-server communications are done via Remote Procedure Call (RPC).

For data storage we adopted a format similar to logrotate files (e.g. based on periodical backup or on file size) to optimize the access to collected measurements. We have introduced the concept of a virtual sensor that treats a flat file and a logrotate-like file as a sensor. This is used for simulations in order to replay a set of previously acquired measurements.

Customizing the system according to its deployment environment is simply done by changing three configuration files: one describing the properties of local power meters (e.g. Hameg or OmegaWatt), one for remote power meters (e.g. IP address, and port number) and one file per sensor describing the computers monitored by the sensor (e.g. IP address, and host name).

In the context of large scale systems like Grid'5000, the growing number of monitored devices complicates the creation of Web pages for displaying the energy consumption. We developed an application which, from the description of the managed devices, automatically produces the RRDTools files, the graphs (i.e. gif files) and Web pages (i.e. html files). Using the developed library, a simple program for generating the Web pages is reduced to the following code. The program produces a web interface containing a set of graphs; a sample graph is given in Figure 2.

```
// instantiation of the observed system
DistributedPowerMeterSystem NRJ_SYST;
// periodical update
do
  {
   NRJ_SYST.UpDateRRDandHtml();
   sleep(60);
   }
 while (1);
```

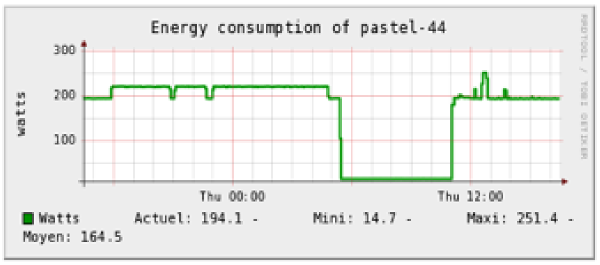


Figure 2: Energy consumption of one node.

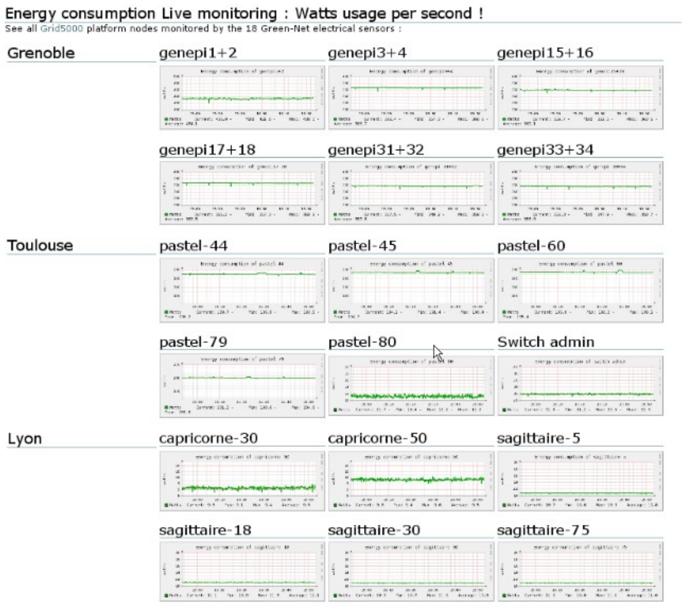

Energy consumption Live monitoring : Watts usage per second !
See all Grid5000 platform nodes monitored by the 18 Green-Net electrical sensors :

Figure 3: Web page example of energy monitoring of 18 nodes.

The proposed library meets the requirements by providing an open interface to meet all needs and being extensible to include other types of sensors.

2.3 Usage Exposure

Having measured and aggregated the energy consumption data, it is necessary to present it to users. At present there are two systems to provide this feedback to the user:

- A Web page (Figure 3) that provides the energy consumption graphs over time for a number of monitored nodes. Using rrd-tools it shows at different time scales, from minutes to months, the energy consumed by nodes without any context. It is mainly used to give a fast and simple feedback to the user, who has to select only the nodes relevant to her application.
- A Web service that using XML provides the energy consumption data from nodes, based on a list of nodes

and a time frame given by the user. Hence, the user can obtain exactly the relevant energy consumption data concerning a job that has run on the grid.

Using these two systems the user is able to monitor the behavior of her application both at coarse and at a fine grain.

3. INVOLVING THE USERS

Once informed about the energy consumption, users can act at different levels:

- First, when they submit their tasks, users can explicitly express whether it is possible to relax part of the hardware requirements, such as CPU for non-urgent tasks and disks if they are not used by the application. Users are in the best position to express such behaviors since they know the applications and their needs. This impacts only their applications.
- Second, users can ask the middleware to act more or less aggressively in terms of energy savings by aggreeing to green policies, and expressing whether they accept to trade performance for energy savings. This has an impact on the whole system as some nodes can be shut down, for instance.

3.1 The PowerSave Mode in OAR

In an effort to allow users to perform an efficient execution of their applications according to their specific performance needs, we have developed mechanisms for cluster Resource and Job Management Systems (RJMSs). These mechanisms enable secure, on-the-fly tuning of the hardware performance for each job submission depending on the user requirements. Our mechanisms have been realized using a flexible RJMS called OAR.

OAR [5] is an open source resource and job management system for clusters. It provides a robust solution currently used as a production system by various platforms such as the experimental grid platform Grid'5000 [6] dedicated for computer science, and the regional grid infrastructure Ciment[5] used for scientific computing in disciplines like environment, chemistry and astrophysics. OAR has been designed considering a modular approach with open architectural choices. It is based upon high-level components: an SQL database and the Perl/Ruby script programming languages. Therefore, it can be easily extended to integrate new features and manage different environments.

The two most interesting levers used by OAR to manage the trade-offs between energy consumption and performance are processors/cores and hard drives. Furthermore users are becoming more energy conscious and at the same time they know better the hardware needs of their applications. While some programs use more CPU and network resources without disk I/O, others perform intensive access to disk and require no communication.

Important research has been made at the application layer in order to analyze and control the energy efficiency of MPI programs [10],[9] or provide special dynamic voltage and frequency scaling mechanisms for reducing energy upon MPI applications [21],[13]. A common feature of modern microprocessors is the mechanism for Dynamic Frequency and Frequency Selection (DVFS). Operating at lower frequencies, CPUs can reduce their energy consumption. As observed in previous work, adaptive use of frequency scaling may result in important energy benefits with small performance degradation [16]. A similar technique for reducing energy consumption is available for hard disks. Tradicionally, disks are made to service requests at their maximum speeds. Even if a disk is not servicing requests, it continues to spin at the maximum rotational speed and hence wastes energy. Therefore, the possibility of spining down the disk when not in use may contribute to its energy efficiency [17].

Our development upon OAR consists of a simple support of CPU frequency scaling and hard disk spin-down using modern platforms that provide this type of hardware features. Hence, users that know their application codes can take advantage of this feature and find the correct balance of energy consumption and execution time.

For this we have introduced a new type of jobs called *PowerSaving*, which allows the user to choose and control the device performance and consequently the power consumption of the computing nodes during their job execution. Unlike most resource management systems, in OAR there is no specific daemon running on the computing nodes of the cluster. Nevertheless, during the execution the server communicates with every node assigned to a job, where it can obtain root privileges and perform all the demanded hardware frequency scaling modifications. At the end of the job, all computing nodes return to their initial states. Currently, only CPU and hard-disk speed scaling can be carried out, but support to other devices is planned. The mechanism makes use of specific linux commands like cpufreq-set [6] for CPU frequency scaling and sdparm [7] for hard-disk spin down.

To experiment with this feature we conducted tests using MPI applications to compare the gain in energy consumption when using the different options of PowerSaving jobs. In particular, we make the comparison among four cases: 1) a normal execution with no CPU frequency scaling or HDD spin-down, 2) only CPU frequency scaling, 3) only HDD spin-down, 4) both CPU frequency scaling and HDD spin-down. For this experiment we use Grid'5000 platform; more specifically, 9 nodes of Genepi cluster with Intel Xeon E5420 QC 2.5GHz DualCPU-QuadCore, 8GB of RAM and Infiniband 20G network. We deploy one node as OAR server and 8 computing nodes. We execute the NAS NPB benchmarks [3], which are widely used to evaluate the performance of parallel supercomputers. We execute class D benchmarks with their MPI3.3 implementation[8] and 64 processes (one process per core). Our study evaluates the percentage trade-off gains between energy consumption and performance for the different powersaving cases compared with a normal execution for the NPB benchmarks.

[5] *https://ciment.ujf-grenoble.fr/cigri*

[6] *http://linux.die.net/man/1/cpufreq-set*

[7] *http://linux.die.net/man/8/sdparm*

[8] *http://www.nas.nasa.gov/Resources/Software/npb_changes.html*

Table 1: Percentages of energy gain vs. performance degradation (execution time).

Method	Energy/Performance		
	HDD Spin	CPU Freq	HDD Spin + CPU Freq
EP	2.5 / 0	10.3 / -18.9	12.2 / -20.5
SP	1.6 / 0.3	8.5 / -1.3	10.2 / -1.5
BT	2 / -0.4	9 / -5.4	10.4 / -5.5
LU	2.2 / 0.2	9.5 / -7.6	11.5 / -10.8
CG	2 / -0.13	8.2 / -1.4	10 / -3.1
IS	1.4 / 1.5	6.4 / -1.5	10 / -7.2
MG	1.2 / -1.1	8.2 / -0.5	9.8 / -3.4
Overall	1.8 / 0.05	8.5 / -5.2	10.5 / -7.4

Our results, summarized in Table 1, show that the use of PowerSaving options achieve good trade-offs between energy reduction and performance loss. Naturally the final gain is marginal if one is interested to have energy reduction with no performance loss. Nevertheless, most of the time some energy benefits are followed by a rather small increase in execution time. Similar results were also observed in previous work [10], which experimented only with CPU frequency scaling. It is suprising to observe how SP, LU and especially IS benchmarks present a gain not only in energy reduction but also in performance when HDD spin-down techniques are performed. This strange behaviour can be explained by the presence of system noise and more particular it could be related to the processor cache and Translation Lookaside Buffer (TLB) behaviour [18]. Nonetheless, recent studies that predict disk idle times and used multi-speed disks have shown important energy savings with small performance losses [11]. Finally, the only case where the trade-off between energy reduction and performance loss is not good is that of the EP benchmark.

3.2 GreenNet Policies

PowerSave mode of OAR is able to improve energy efficiency of individual jobs. However, it is also possible to improve the energy efficiency at the scale of the whole RJMS system by smartly managing two elements: jobs and nodes.

Shutting down nodes leads to vast improvements in energy efficiency. On the other hand, it can delay some jobs in order to avoid switching nodes on for a very short period. For instance, if a job using one node finishes in a few seconds and another job requiring one node is about to start, it can be energy-wise to delay the latter during a few seconds.

We have proposed 6 policies adapted to the GREEN-NET framework :

- *user*: we always select the solution that fits the most the user's demand (we select the date asked by the user or the nearest possible date to the requested date);
- *fully-green*: we always select the solution that saves the most energy (where we need to boot and to shut down the smallest number of resources);
- *25%-green*: we treat 25% of the submission, taken at random, with the previous *fully-green* policy and the remaining ones with the *user* policy;
- *50%-green*: we treat 50% of the submission, taken at random, with the *fully-green* policy and the others with the *user* policy;
- *75%-green*: we treat 75% of the submission, taken at random, with the *fully-green* policy and the others with the *user* policy;
- *deadlined*: we use the *fully-green* policy if it does not delay the reservation from the initial user's demand for more than 24 hours, otherwise we use the *user* policy.

These policies attempt to simulate the behavior of real users: there is a percentage of "green" users who accept a delay as long as they reduce their energy footprint. Other users do not want to delay the execution of their reservations for too long, which is the case of the *deadlined* policy. In addition, some users do not want to change their reservations even if doing so saves energy, that is the *user* policy. The *green* policy illustrates the case of an administrator decision: the administrator always chooses the most energy-efficient option.

4. AUTONOMIC ENERGY AWARE SUPPORT

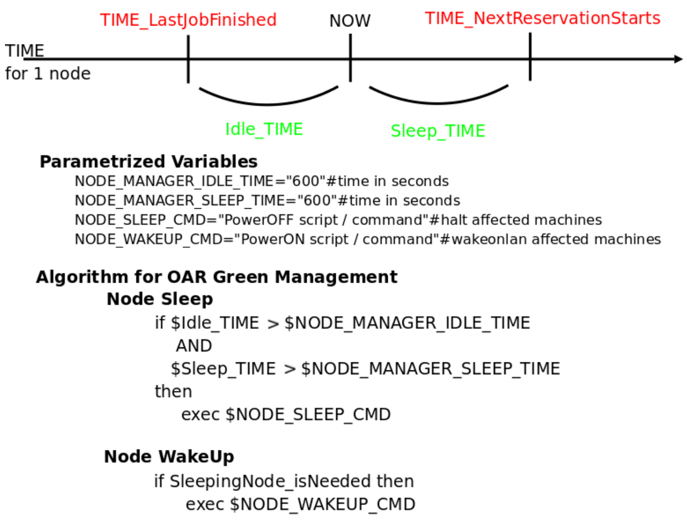
4.1 Energy-Efficient System Exploitation

According to prior work, idle computing resources consume a considerable amount of energy [19],[14]. This energy could be saved if specific actions were taken while the nodes are not allocated by a user. Local RJMSs can play a significant role in the energy management of a platform, since they have overall knowledge and control of the hardware resources and the users' workload. We would like to take advantage of this priviledged position of the middleware to integrate actions performed on idle resources for saving energy.

4.1.1 Energy Reduction through Idle Resources Manipulation

Under this context we have extended the resource management mechanisms of OAR [5] in order to achieve energy-efficient resource exploitation by manipulating idle resources. In particular we have implemented a management optimization for OAR that powers off idle computing nodes of a cluster under specific conditions. Following this method a cluster can benefit from its idle periods and consume less energy. Conversely, if a job demands powered-off nodes, OAR turns them on and allocates them for the job when they are up.

Figure 4 describes the algorithm of the green management mode. It uses specific variables that are parametrized by the administrator of the cluster. The `Idle_TIME` represents the duration that a node has not been allocated by a job. Similarly the `Sleep_TIME` represents the duration that the node will remain unallocated, which means that a job reservation is not planned upon it. As shown in the figure, the Green Management algorithm examines the idleness conditions, according to the parametrized variables, and executes the predefined actions upon the nodes. The executed commands depend on the platform and operating system, but could be any type of shutdown, standby or hibernate modes along with the relevant wake-up commands. In case of sudden job arrival which needs the node, the wake-up command powers-on the node and OAR initiates the job when all nodes are ready for utilization. Certainly a significant job waiting-time is expected depending on the reboot time of the nodes.



```
  Parametrized Variables
      NODE_MANAGER_IDLE_TIME="600"#time in seconds
      NODE_MANAGER_SLEEP_TIME="600"#time in seconds
      NODE_SLEEP_CMD="PowerOFF script / command"#halt affected machines
      NODE_WAKEUP_CMD="PowerON script / command"#wakeonlan affected machines

 Algorithm for OAR Green Management
          Node Sleep
                if $Idle_TIME > $NODE_MANAGER_IDLE_TIME
                   AND
                  $Sleep_TIME > $NODE_MANAGER_SLEEP_TIME
                then
                    exec $NODE_SLEEP_CMD

          Node WakeUp
                if SleepingNode_isNeeded then
                     exec $NODE_WAKEUP_CMD
```

Figure 4: Green management mode with OAR

4.1.2 Performance Evaluation

In order to evaluate our implementation for automatic energy reduction through OAR green management we have executed experiments upon Grid'5000 platform. For this, we have used workload traces from the DAS2 [15] clusters. In particular we have extracted specific parts of the traces according to the system utilization percentage and have replayed those traces using OAR on a cluster of the same size deployed upon Grid'5000. Our goal is to evaluate the different management modes (Normal and Green) of OAR using different workloads.

Our experiments were performed on Lyon Capricorne cluster with AMD Opteron 246 Dual CPU (2.0GHz/1MB/400MHz), 2GB of RAM and Gigabit Ethernet network. As the trace file was collected from a 32 node (Dual CPU) cluster, we have selected 33 nodes of Lyon Capricorne cluster and deployed OAR frontal server on one of them and 32 OAR computing nodes. For the execution of the workload jobs we launch only sleep jobs since we are only interested to evaluate the system behaviour when idle nodes shutdown or not during the experiments. Our interest is to observe the differences on energy consumption and jobs waiting-time for the normal scheduling mode with idle powered-ON nodes compared with the green scheduling mode with idle powered-OFF nodes.

For our tests we have used workload traces of 50.32% and 89.62% system utilization. Figures 5 and 6 present two different experiment plots, one with normal and another with green management upon the same nodes. It is interesting to observe the energy gain in both workload cases.

We can note that between the two figures there are differences in energy consumption of idle state (difference of 2KW.h in Total consumption of Normal scheduling mode), even if we are using the same cluster. This difference is

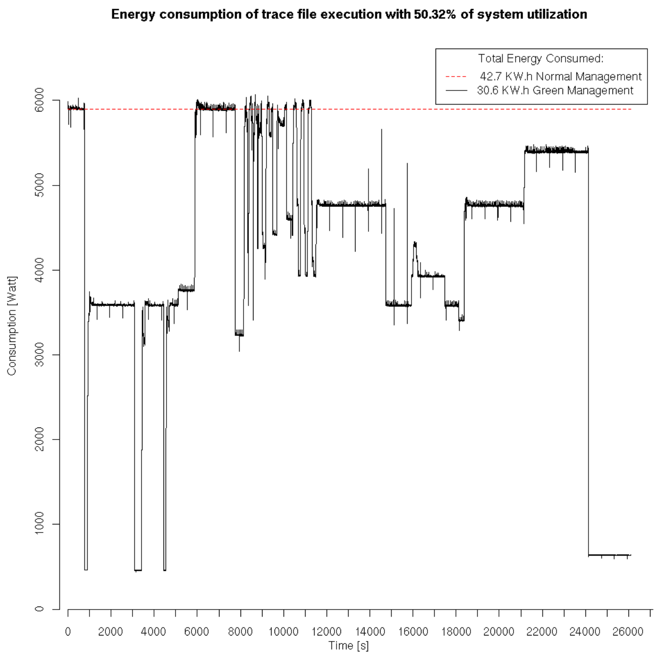


Figure 5: Energy consumption for normal and green management with 50.32% system utilization.

because we did not reserve exactly the same nodes for the experiments shown in the different figures. It is a fact that there is a significant difference in instant energy consumption between the nodes of the same cluster. Indeed, in this particular cluster it has been observed that nodes situated on the bottom of the racks consume less than those located on higher shelves.

The peaks of energy consumption at the beginning of green scheduling mode of figure 6 are due to the fact that nodes were just rebooted when the execution was started, whereas it was not the case in all other experiments of both figures. This event was irrelevant to our experiment and it influences slightly the total energy consumption.

Table 2 shows significant results of the experiment runs. In the case of 50.32% utilization, for experiments 1 and 2, we observe an important gain in energy consumption of 12.1KW.h for 7.25 hours of total workload execution. On the other hand, in the case of 89.62% system utilization, for experiments 3,4 and 7 hours of total workload execution we obtain only 4.1KW.h gain in energy consumption. This is normal due to the high utilization rate which let us only a 10.38% of unutilization periods where we could perform machines Power-OFF and reduce the total energy consumption of the system. In figures 5 and 6 we can clearly see the difference on the number of Powered-ON and Powered-OFF nodes between high and medium utilization rates. However, the gain on energy consumption is followed by an increase on jobs waiting time.

It is observed that the 50.32% utilization case results in a much greater average jobs waiting time than the 89.62%

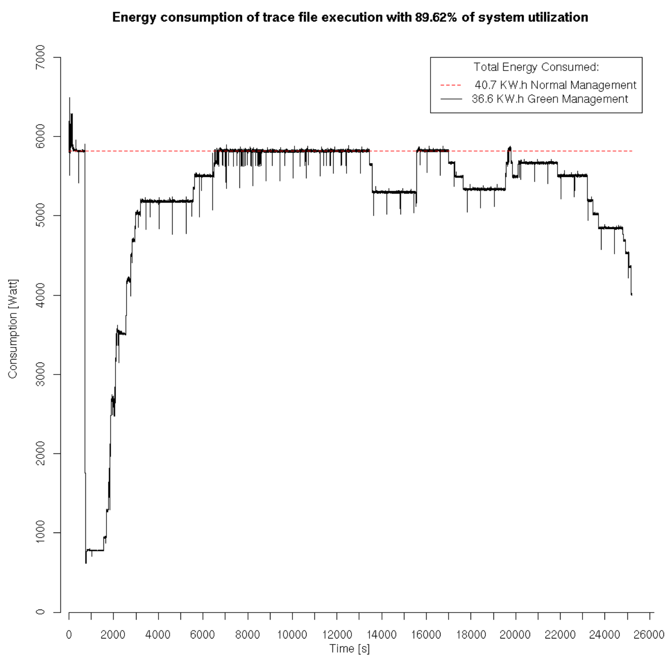


Figure 6: Energy consumption for normal and green management with 89.62% system utilization.

utilization case. In the first case it is nearly 14 minutes and in the second less than 4 minutes. Hence, we obtain a fair trade-OFF between the gain on energy consumption and the loss in average jobs waiting time. The waiting time includes the computers boot time which has been measured as 154 sec, in average, upon the specific computers. The study needs to be extended for other workloads and platforms, but the first results show good trade-offs.

Table 2: Total energy consumption and jobs waiting time for normal and green modes.

Parameters	Experiments			
	1	2	3	4
Management Mode	Normal	Green	Normal	Green
System Utilization (%)	50.32	50.32	89.62	89.62
Total Number of Jobs	309	309	188	188
Total Time of Traces (hours)	7.25	7.25	7	7
Total Energy Consumed (KW.h)	42.7	30.6	40.7	36.6
Average Energy Consumed (KW)	5.9	4.2	5.8	5.2
Average Job Waiting time (sec.)	8	829	1	218

4.2 How Predictions Can Help

In previous work, the time used to decide upon the extinction of a node is based on the calendar of future reservations. In order to improve this, and to predict when it will be used again, we have designed the Energy-Aware Reservation Infrastructure (EARI), which is detailed in [19]. The global idea is to design an infrastructure that works like a garbage collector: it switches off unused nodes, and switches them on again when a user reserves them. A reservation is the allocation of some resources by a user during a certain period of time. In that sense, it resembles the previous approach. The difference lies in the fact that the next reservation is predicted rather than extracted from the reservation calendar, and reservations are aggregated on a limited number of nodes. Hence, our infrastructure is based on three ideas: to switch off the unused resources; to predict the next reservation; and to aggregate the reservations.

We want to predict the next reservation in order to avoid switching off resources that will be used in the really near future. Indeed, such a behavior would consume more energy than keeping the resources powered on. We define T_s as the minimum time which ensures an energy saving if we turn off a resource compared to the energy we use if we let it powered on.

Compared to a simple algorithm where the resources are put into sleep state from the moment that they not in use, the efficiency of our model resides in our ability to make accurate predictions: the estimation of the next reservation (length, number of resources and start time), the estimation of the energy consumed by a given reservation and the estimation of a slack period. However, our prediction algorithm should remain sufficiently simple in terms of computation in order to be efficient and applicable during reservation scheduler run time.

4.2.1 Estimation of the Next Reservation

First of all, we take care of the estimation of the next reservation $R_e = (l_e, n_e, t_e)$ for a given site. To estimate its start time, we take into account the day of the week, the hour of the day and the previous values of arrival times. This method, called method 1, assumes that the reservations' start times for each day are similar to those of the two previous days and those of the same day of the previous week. This method is based on the similarity of the day load and on the cycle of a day (daytime and night) per site.

At a given time t, our estimated start time is the average of the start times of the reservations which are just after t the two days before and the same day one week before on this site, plus the feedback (defined further).

$$t_e = {}^1\!/_3[t_{t,j-1} + t_{t,j-2} + t_{t,j-7}] + t\_feedback$$

where $t_{t,j-i}$ is the start time of the reservation just after t on this site for the day $j - i$ with j which stands for today.

The estimations of the length and of the number of resources required by the next reservation are done in a similar way. We record the three reservations used to make the previous calculation. We call them $R_a = (l_a, n_a, t_{t,j-1})$, $R_b = (l_b, n_b, t_{t,j-2})$, $R_c = (l_c, n_c, t_{t,j-7})$. So we have:

$$n_e = {}^1\!/_3[n_a + n_b + n_c] + n\_feedback$$

$$l_e = {}^1\!/_3[l_a + l_b + l_c] + l\_feedback$$

If we do not observe this day similarity, we use method 2. This method is based on the similarity between the close reservations in terms of start time per site. The basic idea is to calculate the average of the characteristics of the five previous reservations on this site.

At a given time t, we denote $R_0, \ldots, R_5$ the six previous reservations on this site (with $R_i = (l_i, n_i, t_i)$). They are the six reservations on this site whose start times are the nearest to t (but not necessarily before t, scheduled reservations can be taken into account). These reservations are in order of growing start time (R_0 is the oldest).

So the estimation of the start time is done by calculating the average of the five intervals between the six previous start times. This average is added to t with the feedback to obtain the estimation:

$$t_e = t + {}^1\!/_5[t_5 - t_0] + t\_feedback$$

Similarly, we define the estimations of the length and of the number of resources required by the next reservation.

In the two methods, if we obtain $t_e < t$ (because of the feedback), we set $t_e = t + 1$. The choice between method 1 and method 2 should be done according to the site usage. We should compare their performance on a given site to say which one is better for this given platform.

The accuracy of the next reservation prediction is crucial for our power management. If we make too many wrong estimations, resources either wait for imminent reservations that do not arrive and waste energy or are turned off just before an imminent reservation arrival and waste the energy of one halting plus one reboot per resource.

4.3 Feedback on the Next Reservation Estimation

The feedback is used to improve the energy efficiency of our approach. As we have seen before, the estimation errors are really penalizing in terms of consumed energy. We need to obtain accurate predictions.

Moreover, we have observed that there are less reservations during the night and more during the morning for the Grid'5000 traces. Hence, early in the morning for example, method 2 will certainly have some difficulties to predict the next reservation start time. Therefore, to limit the effects of such errors we use a feedback mechanism. The feedback is in fact a corrective factor calculated with the three previous errors (it can be more).

At each reservation arrival, we compute the estimation errors we have made. More precisely, at a given time t, the reservations $R = (l_0, n_0, t_0)$ arrives. $R_e = (l_e, n_e, t_e)$ is the last reservation that we have predicted. We denote $Err_l = (l_0 - l_e)$: the error done by estimating the length of the next reservation, $Err_n = (n_0 - n_e)$ and $Err_t = (t_0 - t_e)$ the errors done by estimating the number of resources and the start time of the next reservation respectively.

Basically, if we predict the reservation too early, then we have $Err_t > 0$. Hence, if we add Err_t to the next predicted start time, we delay the predicted start time by Err_t seconds and that is exactly what we intended. Then we denote $Err_l(a)$, $Err_l(b)$ and $Err_l(c)$ the three last errors for the length of a reservation. $n\_feedback$ and $t\_feedback$ are similar to $l\_feedback$:

$$l\_feedback = {}^1\!/_3[Err_l(a) + Err_l(b) + Err_l(c)]$$

4.3.1 *Energy Consumption Estimation of a Reservation*

This estimation takes into account the user, the resource type and all the characteristics of the reservation $R = (l, n, t)$. The assumption made here is that each user has almost the same usage of the resources. What we really estimate is the average power during working time per resource for each different type of resource (different architectures for example).

4.3.2 *Slack Periods*

A slack period is a period longer than two hours with a usage of the platform below 50%. Typically, such periods happen during the night. We take two hours because it is just a bit longer than the average length of a reservation on Grid'5000. Hence a large number of reservations can take place during such a period in terms of length. To estimate when the next slack period will occur, we use the values of the three previous days (real values are known at that time). If there was no slack periods during the three previous days, we estimate that there will be no slack period that day.

To be really complete, our model should include the energy consumption of the cooling infrastructure proportionally distributed on each resource. In fact, $P_{real}(type_k, R_a)$ (the real average power for the reservation R_a for a resource which have a $type_k$ type) would include a fraction of the average power consumed by the cooling infrastructure during the duration of the reservation R_a proportionally to its heat production. However, such a fraction can be difficult to estimate, that is why most of the power management systems do not take the cooling infrastructure into account.

4.3.3 *Evaluation*

To evaluate our model, we conduct experiments based on a replay of the Grid'5000 traces. Figure 7 presents our results for 4 different sites with 4 different workloads. By assuming that the user gives a wished start time, we have defined six different policies: *user*, *fully-green*, *25%-green*, *50%-green*, *75%-green*, and *deadlined*. These diagrams show the percentages of energy consumed with EARI compared to the present consumption (when all the nodes are always fully powered on).

The *all-glued* line shows the theoretical lower bound which represents the case where all the reservations are glued together at the end of the year (this minimizes the idle time and the number of switching on and off. We see that our *fully-green* policy is always the best and could lead to huge energy savings (almost 44% on average for these four examples).

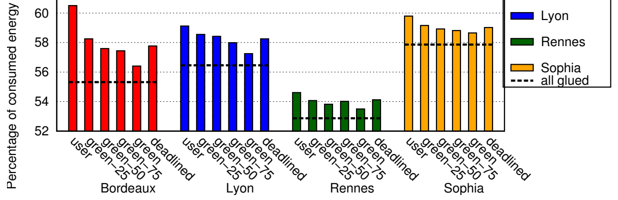


Figure 7: Results of EARI for 4 sites.

4.4 Trust Support for Migrating Services

When nodes are turned off, there is a need to migrate some basic services for ensuring network presence. By adapting a solution initially proposed for pervasive computing [20], we have developed a service migration technique [7]. The approach we followed is based on trust evaluation mechanisms and trust propagation among the set of possible nodes able to host services. The idea is to choose, based on the trust of the hosts' network, the best host for the moving service.

5. CONCLUSION

This paper presents different approaches and their extensive evaluations for reducing the energy consumption of grids and clouds, acting at: the user level by informing users of the energy consumed by their applications and involving users in the making of energy reduction decisions; and at the middleware level by adapting the infrastructure via individual hardware tuning, using ON/OFF models among a set of hosts to switch off nodes and enabling service migration.

Current and future work include further developments on the automatic adaptation of the system, including other factors in the PowerSave mode, and extending the prediction strategies. Another direction is the integration of energy aware task allocation [4] in the framework, together with its extension to full inclusion of task migration.

6. REFERENCES

[1] Google powermeter project, accessed Nov 1, 2009. http://www.google.org/powermeter/.

[2] Microsoft hohm project, accessed Nov 1, 2009. http://www.microsoft-hohm.com/.

[3] David Bailey, Tim Harris, William Saphir, Rob Van Der Wijngaart, Alex Woo, and Maurice Yarrow. The nas parallel benchmarks 2.0. Technical report.

[4] Damien Borgetto, Georges Da Costa, Jean-Marc Pierson, and Amal Sayah. Energy-Aware Resource Allocation (regular paper). In *Energy Efficient Grids, Clouds and Clusters Workshop (co-located with Grid 2009) (E2GC2), Banff, 13/10/09-15/10/09*, page (electronic medium), http://www.ieee.org/, octobre 2009. IEEE.

[5] Nicolas Capit, Georges Da Costa, Yiannis Georgiou, Guillaume Huard, Cyrille Marti n, Grégory Mounié, Pierre Neyron, and Olivier Richard. A batch scheduler with high level components. In *Cluster computing and Grid 2005 (CCGrid05)*, 2005.

[6] F. Cappello et al. Grid'5000: A large scale, reconfigurable, controlable and monitorable grid platform. In *6th IEEE/ACM International Workshop on Grid Computing, Grid'2005*, Seattle, Washington, USA, Nov. 2005.

[7] Georges Da-Costa, Jean-Patrick Gelas, Yiannis Georgiou, Laurent Lefèvre, Anne-Cécile Orgerie, Jean-Marc Pierson, Olivier Richard, and Kamal Sharma. The green-net framework: Energy efficiency in large scale distributed systems. In *HPPAC 2009 : High Performance Power Aware Computing Workshop in conjunction with IPDPS 2009*, Rome, Italy, May 2009.

[8] W. Feng and T. Scogland. The green500 list: Year one. In *IPDPS '09: Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*, pages 1–7, Washington, DC, USA, 2009. IEEE Computer Society.

[9] Xizhou Feng, Rong Ge, and Kirk W. Cameron. Power and energy profiling of scientific applications on distributed systems. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers*, page 34, Washington, DC, USA, 2005. IEEE Computer Society.

[10] Vincent W. Freeh, David K. Lowenthal, Feng Pan, Nandini Kappiah, Rob Springer, Barry L. Rountree, and Mark E. Femal. Analyzing the energy-time trade-off in high-performance computing applications. *IEEE Transactions on Parallel and Distributed Systems*, 18(6):835–848, 2007.

[11] Rajat Garg, Seung Woo Son, Mahmut Kandemir, Padma Raghavan, and Ramya Prabhakar. Markov model based disk power management for data intensive workloads. In *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 76–83, Washington, DC, USA, 2009. IEEE Computer Society.

[12] The G. Grid. The green grid data center power efficiency metrics: Pue and dcie. Technical report, 2007.

[13] S. Huang and W. Feng. Energy-efficient cluster computing via accurate workload characterization. In *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 68–75, Washington, DC, USA, 2009. IEEE Computer Society.

[14] Laurent Lefèvre and Anne-Cécile Orgerie. Towards Energy Aware Reservation Infrastructure for Large-Scale Experimental Distributed Systems. *Parallel Processing Letters - Special Issue on Clusters and Computational Grids for Scientific Computing*, 19(3):419–433, September 2009.

[15] Hui Li, David L. Groep, and Lex Wolters. Workload characteristics of a multi-cluster supercomputer. In Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *JSSPP*, volume 3277 of *Lecture Notes in Computer Science*, pages 176–193. Springer, 2004.

[16] Min Yeol Lim, Vincent W. Freeh, and David K. Lowenthal. Adaptive, transparent frequency and voltage scaling of communication phases in mpi programs. In *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 107, New York, NY, USA, 2006. ACM.

[17] Yung-Hsiang Lu and Giovanni de Micheli. Adaptive hard disk power management on personal computers. In *GLS '99: Proceedings of the Ninth Great Lakes Symposium on VLSI*, page 50, Washington, DC, USA, 1999. IEEE Computer Society.

[18] Collin McCurdy, Alan L. Coxa, and Jeffrey Vetter. Investigating the tlb behavior of high-end scientific applications on commodity microprocessors. In *ISPASS '08: Proceedings of the ISPASS 2008 - IEEE International Symposium on Performance Analysis of Systems and software*, pages 95–104, Washington, DC, USA, 2008. IEEE Computer Society.

[19] Anne-Cécile Orgerie, Laurent Lefèvre, and Jean-Patrick Gelas. Save watts in your grid: Green

strategies for energy-aware framework in large scale distributed systems. In *14th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, Melbourne, Australia, December 2008.

[20] Lionel Brunie Rachid Saadi, Jean-Marc Pierson. Security in distributed collaborative environments: limitations and solutions. *Book Chapter in Emergent Web Intelligence, to be published by Springer Verlag (series Studies in Computational Intelligence)*, 2009.

[21] Barry Rountree, David K. Lownenthal, Bronis R. de Supinski, Martin Schulz, Vincent W. Freeh, and Tyler Bletsch. Adagio: making dvs practical for complex hpc applications. In *ICS '09: Proceedings of the 23rd international conference on Supercomputing*, pages 460–469, New York, NY, USA, 2009. ACM.

[22] Andy B. Yoo, Morris A. Jette, and Mark Grondona. SLURM: Simple Linux utility for resource management. In Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, pages 44–60. Springer Verlag, 2003. Lect. Notes Comput. Sci. vol. 2862.

[23] Heng Zeng, Carla S. Ellis, Alvin R. Lebeck, and Amin Vahdat. Ecosystem: managing energy as a first class operating system resource. *SIGPLAN Not.*, 37(10):123–132, 2002.