Expert Systems with Applications 39 (2012) 9443–9450

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Smart scheduling for saving energy in grid computing

A. Fernández-Montes [a,*], L. Gonzalez-Abril [b], Juan A. Ortega [a], Laurent Lefèvre [c]

[a] Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Spain
[b] Departamento de Economía Aplicada I, Universidad de Sevilla, Spain
[c] INRIA RESO – Université de Lyon – École Normale Supérieure, France

ARTICLE INFO

Keywords:
Energy policies
Efficiency
Arranging policies
Simulation software

ABSTRACT

Energy saving involves two direct benefits: sustainability and cost reduction. Within the field of Information Technology, clusters, grids and data centres represent the hungriest consumers of energy and therefore energy (saving) policies for these infrastructures should be applied in order to maximize their resources. It is proved in this paper that approximately 40% of energy can be saved in a data centre if an adequate policy is applied. Furthermore, a software tool is presented where simulations can be run and results for real scenarios can be obtained.

1. Introduction

In today's increasingly pessimistic times not only are we faced with major economic issues worldwide, but also with those of sustainability. As a part of sustainability, researchers are encouraged to save energy in all domains, from Information Technology (IT) to transport. Saving energy is directly related with cutting costs and environmental sustainability. Energy efficiency is therefore sought in a wide range of systems from small devices to large-scale computing.

Information Technology energy consumption represents a mere 3–5% of CO_2 emissions worldwide which is similar to that of aviation transport. While apparently trivial in quantity, this usage is symbolic since IT can greatly influence other industrial and research domains (Ruth, 2009). As computing requirements are ever greater, microprocessor manufacturers are doubling the electrical efficiency of computation every year and a half (Sanchez, Wong, Berard, & Koomey, 2011). Nevertheless, energy consumption is still rising despite these good results, with energy consumption of data centres increasing an average of 16.7% over the last decade (Koomey, 2008).

Some companies, such as Google (Ren, Tune, Moseley, Shi, & Hundt, 2010), are committed to increasing energy efficiency in data centres and in cloud computing. The research community has also been searching for improvements in energy efficiency, whereas the majority of companies have focused their efforts on improving facilities.

The huge amount of energy consumed by grid computing provides justification for a study into energy-saving methodologies, either from an economic or ecological point of view. To this end, grid operational policies must be analyzed in order to be optimized.

An experimental grid organization, located in France and called Grid'5000, is analyzed in this paper. Grid'5000 is a scientific instrument designed to support experiment-driven research in all areas of computer science related to parallel, and large-scale and distributed computing and networking. Its purpose is to supply a highly reconfigurable, controllable and monitorable experimental platform to its users.

The rest of this paper is structured as follows: Section 2 includes a brief introduction to Grid'5000 organization and its current energy consumption is analyzed. Various on–off policies, designed to save energy are presented, and a comparison between current energy consumption and the results of each on–off policy are given in Section 3. The way in which jobs can be scheduled between resources is shown in Section 4. Software developed for the testing and simulation is explained in Section 5. Finally, in Section 6, results and conclusions are drawn.

2. Grid'5000 at a glance

Grid'5000 has been built upon a network of dedicated clusters. It is not an *ad hoc* grid. The infrastructure of Grid'5000 is geographically distributed over various sites, of which the initial 9 are located in France: Bourdeaux, Grenoble, Lille, Lyon, Nancy, Orsay, Rennes, Sophia-Antipolis and Toulouse. Porto Alegre, in Brazil, and Luxemburg, are currently being officialy included as the 10th and 11th sites, respectively. The project began in 2004 as an initiative by the French Ministry of Education and Research, INRIA, CNRS, the Universities of on all the aforementioned and several regional councils.

The initial aim was to achieve 5000 processors on the platform. This objective, reframed at 5000 cores, was reached during the

* Corresponding author. Tel.: +34 626215333; fax: +34 954557139.
E-mail address: afdez@us.es (A. Fernández-Montes).

winter of 2008–2009. On March 16, 2010, 1569 nodes (5808 cores) were in production in Grid'5000. These sites can now connect each other within the same VLAN at 10Gbps thanks to the dark fibre infrastructure which connects them, in an incomplete graph scheme.

Grid'5000 allows experiments at grid or at cluster level, which guarantees a more homogeneous hardware and bandwidth, although grid level experiments are preferred in planning. Each site of Grid'5000 hosts several clusters, since hardware has been acquired in incremental steps on each site, whereby clusters have been formed at each purchase. Each cluster is composed of two kind of nodes:

- Compute nodes, which constitute the base elements of a cluster, on which computations are run.
- Service nodes, which are dedicated to hosting the grid infrastructure services, such as control and deploy.

Each node can supply several cores, which are the finest grains of resource in Grid'5000. This means that if a machine has a microprocessor with *n* cores, it offers *n* resources to the grid.

2.1. Jobs

The platform can be used in two different modes: submission and reservation.

- *Submission*: This is used when a job is submitted to the grid and the user expects it to be launched immediately. The scheduler decides whether the job can be run, by taking into account the occupation of current resources and the agenda of future jobs. Users usually check these requirements before the submission of a new job through a web interface that presents the state and the agenda of the grid and its resources.
- *Reservation*: This is used when a job is to be launched on the grid in the future. The scheduler checks the requirements of time and resources and decides if the reservation can be made or not. Again, users usually check these requirements before making a reservation.

The software used for task schedule is OAR. This is a resource manager (or batch scheduler) for large clusters which allows cluster users to submit or reserve nodes either in an interactive or in a batch mode. Job information includes submission time, start and stop time, job identification given by the manager software, the owner of the job, and the set of resources assigned, which are going to run the job. The job information includes other information which is irrelevant to this research.

2.2. Resources

The Grid'5000 platform features different kinds of machines depending on the location and the cluster they belong to, and when these machines were included on the platform. Two families can be found: Intel Xeon and AMD Opteron. Each machine offers its CPU cores (usually 2, 4 or 8) to the grid to execute jobs. Each CPU core is called resource and each job is related with a set of these resources. Although the performance of each resource is not identical, the assumption that performances are very similar is made, and hence there is no effective difference between running a job on one resource or another. The same assumption is also made about the consumption of these resources, and therefore each resource uses the same amount of energy.

These assumptions enable job resources to be rearranged for energy-saving purposes, without having to consider which type of resources a job originally belonged; hence there is no difference between running a job on one a set of resources or on another.

The various states of the resources and their estimated power required are listed below:

- *On*. A resource is *On* when it is occupied by a job; the resource is running the job. A job is usually deployed over a set of resources. The power needed is approximately 108 W.
- *Off*. A resource is *Off* when it has been switched off. This means the resource is not occupied with any job. The power needed is approximately 5 W.
- *Idle*. A resource is *Idle* when it has been switched on and waiting for new jobs, but it is not occupied with any job. The power needed is approximately 50 W.
- *Booting*. A resource is *Booting* when it is being switched on from *Off* to *On*. The power needed is approximately 110 W.
- *Shutting*. A resource is *Shutting* when it is being switched off from *On* or *Idle* to *Off*. The power needed is approximately 110 W.

Fig. 1 shows the life cycle of the resources where colours are representative for future figures, that is, green for *On*, blue for *Idle*, red for *Shutting*, grey for *Off* and yellow for *Booting*. The time needed for a status change is shown along the edges. Notice that status changes between *Idle* and *On* are immediate. The times $T_{booting}$ and $T_{shutting}$ have been established for simulation purposes as 100 sgs and 10 sgs, respectively.

2.3. Performance

Current Grid'5000 behaviour leaves resources *Idle* while waiting for new jobs to run. This policy is the so-called *Always On* policy which is the best for the fast satisfaction of users needs, but the worst in terms of energy consumption. This paper is focused in replacing this current policy with new policies which are presented in the following section.

3. Scheduling energy policies

Energy policies establish the managing of grid resources. While other research attempts to reduce the makespan (Tseng, Chin, & Wang, 2009), the policies shown in this work aim to describe and compute what to do with a resource when it finishes the exe-

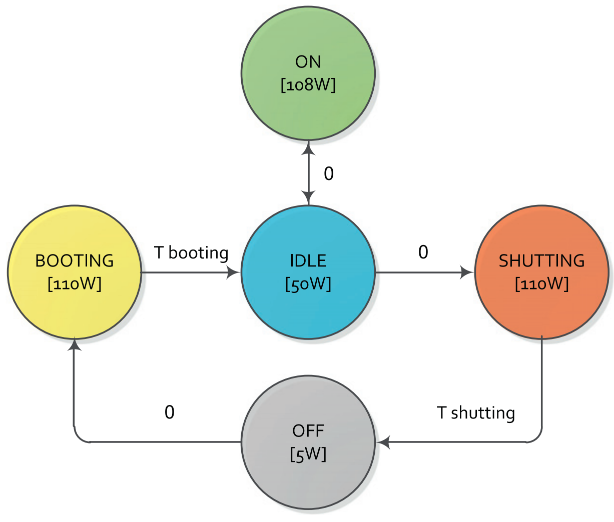


Fig. 1. Life cycle of the resource.

cution of a job. Thus, each energy policy decides to either leave resources switched on or switch them off depending on the purpose of the policy. Each energy policy is illustrated with a screenshot of the Graphical User Interface (GUI) where the horizontal axis is the timeline and the vertical axis represents the resources. The following energy policies are implemented in the Grid'500 Toolbox (Section 5):

3.1. Always On

This is the simplest energy policy, whereby resources are never switched off, under any condition, and therefore resources remain idle, waiting for a new job to be run. Currently Grid'5000 works under this mode, and hence these consumption results are used for comparison with the results of other energy policies in order to determine how much energy can be saved. The number of times resources are switched off or on are always zero, and therefore their stress is minimal. Fig. 2 shows the typical appearance of resources while the Grid'5000 Toolbox is running this energy policy. In this figure, four resources are shown, each resource is denoted by a row, and four jobs have been carried out.

3.2. Always Off

Under this policy, resources are always switches off under any condition, and therefore resources start shutting down after any job finishes, and later they remain switched off. If a new job arrives, the assigned resources have to be booted to run that job. This booting is done within reservation limits, and thus the user is unable to make effective use of the resources until they are booted. This policy is usually the best regarding energy consumption results, but the number of times that booting and shutting is always at a maximum, and therefore the stress produced on the hardware components is the highest, which is not desirable. The typical appearance of resources while the Grid'5000 Toolbox is running under this energy policy is shown in Fig. 3.

3.3. Switch Off Randomly

This policy switches off and randomly leaves the resources idle by following a Bernoulli distribution whose parameter is equal to 0.5 when a job finishes. Hence, the times that resources are switched off or left idle tends towards 50%. Results tend to be half-way between those of the *Always Off* and *Always On* policies regarding the two kinds of results: the times resources are switched off and energy consumption. The typical appearance of resources while the Grid'5000 Toolbox is running under this energy policy is shown in Fig. 4.

3.4. Load

Load can be defined as the percentage of resources that are *On* among the clusters of a location. This policy uses this information and either switches resources off if the load, when finishing a job,

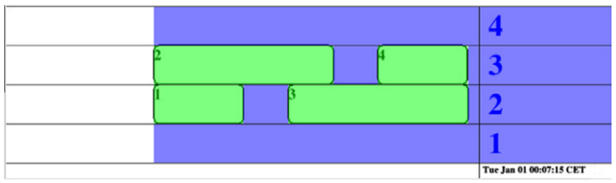


Fig. 2. Example of *Always On* policy. Idle (blue), On (green). (For interpretation of references to colors in this figure legend, the reader is referred to the web version of this article.)

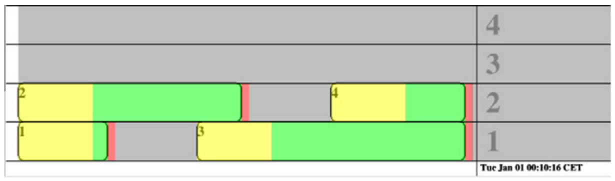


Fig. 3. Example of *Always off* policy. On (green), Shutting Down (red), Off (grey), Booting (yellow). (For interpretation of references to colors in this figure legend, the reader is referred to the web version of this article.)

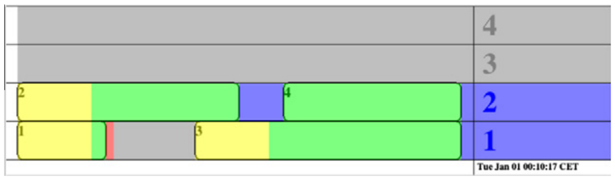


Fig. 4. Example of *Switch off randomly* policy. Idle (blue), On (green), Shutting down (red), Off (grey), Booting (yellow). (For interpretation of references to colors in this figure legend, the reader is referred to the web version of this article.)

is greater than the threshold or leaves the resources idle if the load is less than the threshold. The threshold is a parameter, ranging from 0 to 1, selected from the GUI. The typical appearance of resources while the Grid'5000 Toolbox is running under this energy policy with 0.6 as its threshold parameter is shown in Fig. 5.

3.5. Switch off T_S

T_S is defined *as the minimum time that ensures energy saving if a resource is switched off between two jobs* (Orgerie, Lefèvre, & Gelas, 2008). T_S can be computed as follows:

$$T_S = \frac{E_s - P_{Off} * \delta_{tot} + E_{On\to Off} + E_{Off\to On}}{P_{Idle} - P_{Off}}$$

where P_{Off} and P_{Idle} refer to the power consumption in watts of a given resource when it is *Off* and *Idle*, respectively. $E_{On\to Off}$ and $E_{Off\to On}$ refer to the energy required in joules for a given resource to boot or switch it off respectively. E_S is the energy saved for T_S seconds. Finally, $\delta_{tot} = \delta_{On\to\ Off} + \delta_{Off\to On}$ which is the total time a given resource needs for it to switch itself off and switch itself on.

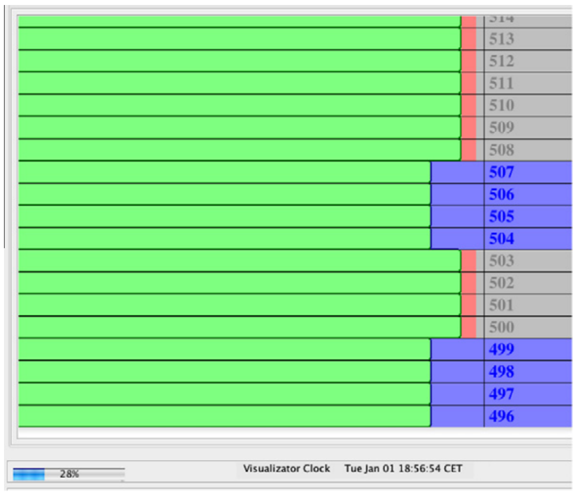


Fig. 5. Example of *Load* policy. Idle (blue), On (green), Shutting Down (red), Off (grey). (For interpretation of references to colors in this figure legend, the reader is referred to the web version of this article.)

This energy policy uses the agenda to check whether the subsequent submitted jobs are going to be run in the grid in less than T_S. This policy computes the number of resources that are needed in a time period less than T_S, and leaves the resources of the recently terminated job idle or shuts them down depending on this computation. In this way, the simulator attempts to minimize booting and shutting-down cycles when no energy can be saved. The typical appearance of resources while the Grid'5000 Toolbox is running under this energy policy, where $T_S = 130$ s, is shown in Fig. 6.

3.6. Exponential

The Exponential distribution, denoted by $Exp(\lambda)$, describes the time between events in a Poisson process, i.e. a process in which events occur continuously and independently at a constant average rate $(1/\lambda)$. Under the hypothesis that the arrival of new jobs follows an Exponential distribution, this energy policy attempts to predict the arrival of new jobs. Thus, to compute the λ parameter, every time a job finishes, the mean time between the last jobs is computed, and denoted by μ. Hence, $\lambda = 1/\mu$ by using the method of maximum likelihood. The probability of the arrival of a new job can then be computed by means of the exponential cumulative density function (cdf) as $cdf(T_s) = 1 - e^{-T_s/\mu}$. Therefore, given a *threshold* value, the following conditions are imposed:

$$\begin{cases} \text{if} \quad cdf(T_s) >= threshold & \text{then leave resources } \textit{Idle} \\ \text{if} \quad cdf(T_s) < threshold & \text{then switch resources } \textit{Off} \end{cases}$$

3.7. Gamma

The Gamma distribution, denoted by $\Gamma(\theta, \kappa)$, is frequently used as a probability model for waiting times and presents a more general model than the Exponential. Under the hypothesis that the arrival of new jobs follows a Gamma distribution, this energy policy attempts to predict the arrival of new jobs. The parameters computed every time a job finishes are:

- number of resources available as *resourcesAvailable*. These are the resources that are *Idle* and ready to accept new jobs.
- mean resources used by last jobs as *meanResources*. Total number of resources used by the last jobs is computed and divided by the number of jobs. The number of jobs is a selectable window size.
- mean duration between the previous number of last jobs as *meanDuration*. The sum of the duration of these last jobs is computed and divided by the previous number of last jobs.
- the floor of *resourcesAvailable*/*meanResources* as *z*.

The parameters of the Gamma distribution are then estimated as: $\theta = 1/meanDuration$ and $\kappa = z + 1$. Finally the probability of the arrival of a new job is computed by means of the cumulative density function (cdf) with

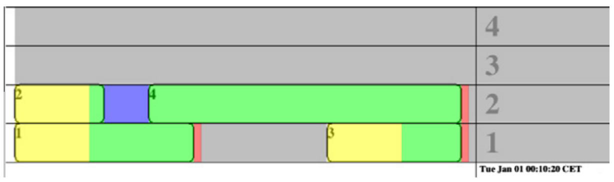


Fig. 6. Example of *Switch off* T_S policy with T_S = 130 seconds. Idle (Blue), On (Green), Shutting Down (Red), Off (Grey). (For interpretation of references to colors in this figure legend, the reader is referred to the web version of this article.)

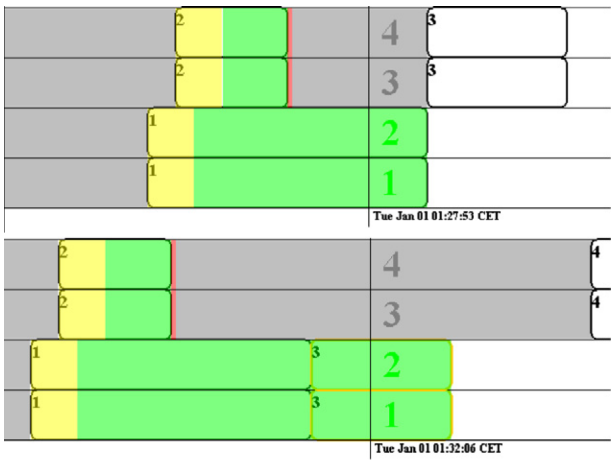


Fig. 7. Example of arranging policies. i) Before jobs have been arranged. Job #3 is about to start, assigned to *Off* resources. (ii) After jobs have been arranged. Job #3 has been moved to available resources. No need to boot resources.

$$cdf(T_s) = \frac{\gamma(\kappa, T_s/\theta)}{\Gamma(\kappa)}$$

Hence, given a *threshold* value, the following conditions are imposed:

$$\begin{cases} \text{if} \quad cdf(T_s) >= threshold & \text{then leave resources } \textit{Idle} \\ \text{if} \quad cdf(T_s) < threshold & \text{then switch resources } \textit{Off} \end{cases}$$

4. Arranging policies

Arranging policies establish the arranging of the jobs for their execution. A job can be moved from one set of resources to another, or a planned job execution can even be moved in time in order to take advantage of resources that are already switched on.

- *Do Nothing (DN)*: does not move jobs in time nor from one resource to another; they are executed as defined in the agenda. This together with the energy policy *Always On* offers the current Grid'5000 behaviour.
- *Simple Aggregation of Jobs (SA)*: This policy tries to find resources available (*Idle*) for new jobs. In this way, if a job is assigned to a set of resources which are *Off* and some other resources are available, the time and the energy needed to be switched on can be saved. Notice that this policy does not change start or stop times, and hence it is transparent to users.

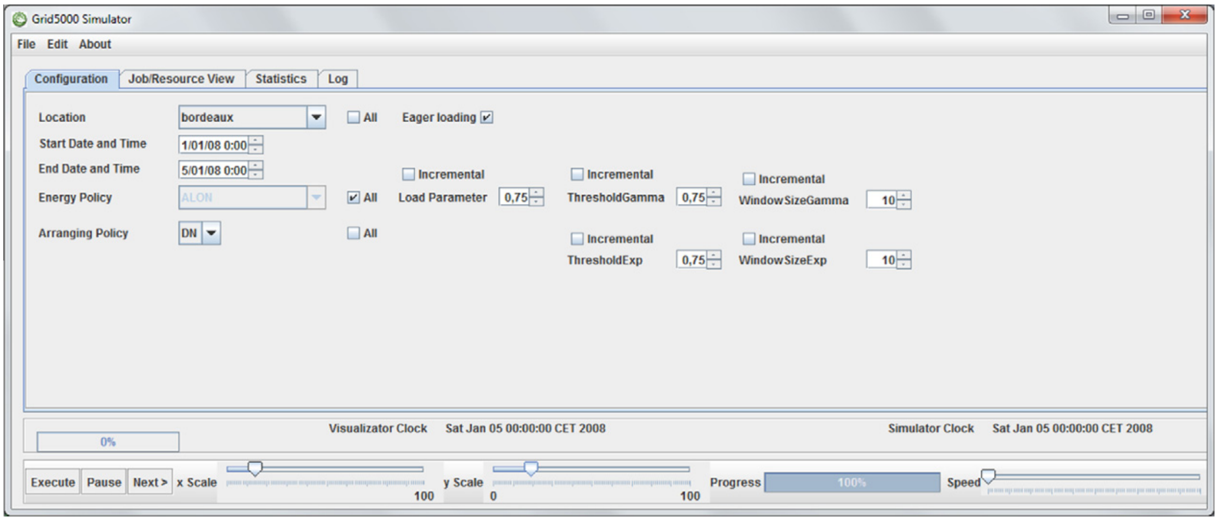
An example of these arranging policies can be seen in Fig. 7.

5. Grid'5000 Toolbox Simulator

Grid'5000 Toolbox[1] replays the progress of the real grid regarding the operation of jobs and resources. Grid'5000 Toolbox is able to compute energy consumption of Grid'5000, and enables the user to set up several parameters including: (*a*) simulation start-time, (*b*) simulation stop-time, (*c*) location, (*d*) energy policy and (*e*) arranging policy. These parameters can be set up through the *Configuration* tab as shown in Fig. 8.

Grid'5000 Toolbox (Grid Toolbox, 2011) is a Java Desktop application using libraries to: 1. deal with energy and time magnitudes (JScience Martin-Michiellot, 2008); 2. communicate with RDBMS

[1] This software can be downloaded and executed from the web of the Idinfor research group (Idinfor, 2011).



Location bordeaux ☐ All Eager loading ✓
Start Date and Time 1/01/08 0:00
End Date and Time 5/01/08 0:00
Energy Policy ALON ✓ All ☐ Incremental Load Parameter 0,75 ☐ Incremental ThresholdGamma 0,75 ☐ Incremental WindowSizeGamma 10
Arranging Policy DN ☐ All ☐ Incremental ThresholdExp 0,75 ☐ Incremental WindowSizeExp 10

0%
Visualizator Clock Sat Jan 05 00:00:00 CET 2008
Simulator Clock Sat Jan 05 00:00:00 CET 2008
x Scale 100; y Scale 0 100; Progress 100%

Fig. 8. Configuration tab presenting setup parameters for a batch of simulations.

(JDBC connector API Reese, 2000); 3. annotate database entities (Java Persistence API JPA Keith and Schincariol, 2006; 4. model statistical distributions (JSC Java Statistical Classes Bertie, 2002); 5. logging simulation information (Apache Logging Services Log4Java Gupta et al., 2011); 6. write results in Excel files (JExcel API Khan, 2010); Grid'5000 Toolbox includes a module to parse raw log files from Grid'5000 systems and stores these files in an standard RDBMS through JPA annotations. Each log file is related to one location. The data found in these log files includes: past jobs, resources, machines, clusters, dead-state resources, users, and relations between jobs and resources.

The simulator operation is based on an agenda where jobs are registered and on a list of resources representing the real resources at the sites. The simulator starts querying the agenda from start-time to stop-time. Each query is related to current simulation time, and hence the agenda seeks jobs and events that occur at given current time. Once the agenda returns new events, the simulator processes them and changes resources states as would be needed for execution in the real world, whilst taking into account the policies selected in order to manage resources and jobs. The consumed energy is computed step by step by means of the information on energy consumption of each resource and the resource states detailed in the resource list. The results of simulation executions are stored on a spreadsheet where researchers can find details about consumption, the number of shutting and booting of resources, the comparison between minimal energy consumed and current energy consumed, etc. Results are also shown in the *Statistics* tab in a more visual way (see Fig. 9).

6. Experimentation

In order to present the results, the recommendations for measuring and reporting Overall Data Centre Efficiency (Green Grid, 2011) were taken into account. Results for every combination of energy and arranging policy summarize the behaviour of these policies for each location and for each time period selected. The computed information includes:

- total number of bootings and shuttings,
- total energy consumed
- energy saved as compared with the energy consumed by current Grid'5000 policies (*Always Leave On* and *Do not Arrange* policies)
- comparison between the minimal[2] energy consumable for an execution and actual energy consumed by each combination of policies,
- comparison between the energy consumed by current Grid'5000 policies for an execution and actual energy consumed by each combination of policies,
- saving in energy attained per shutting down, which shows the validity of the shutting down decisions.

6.1. General results

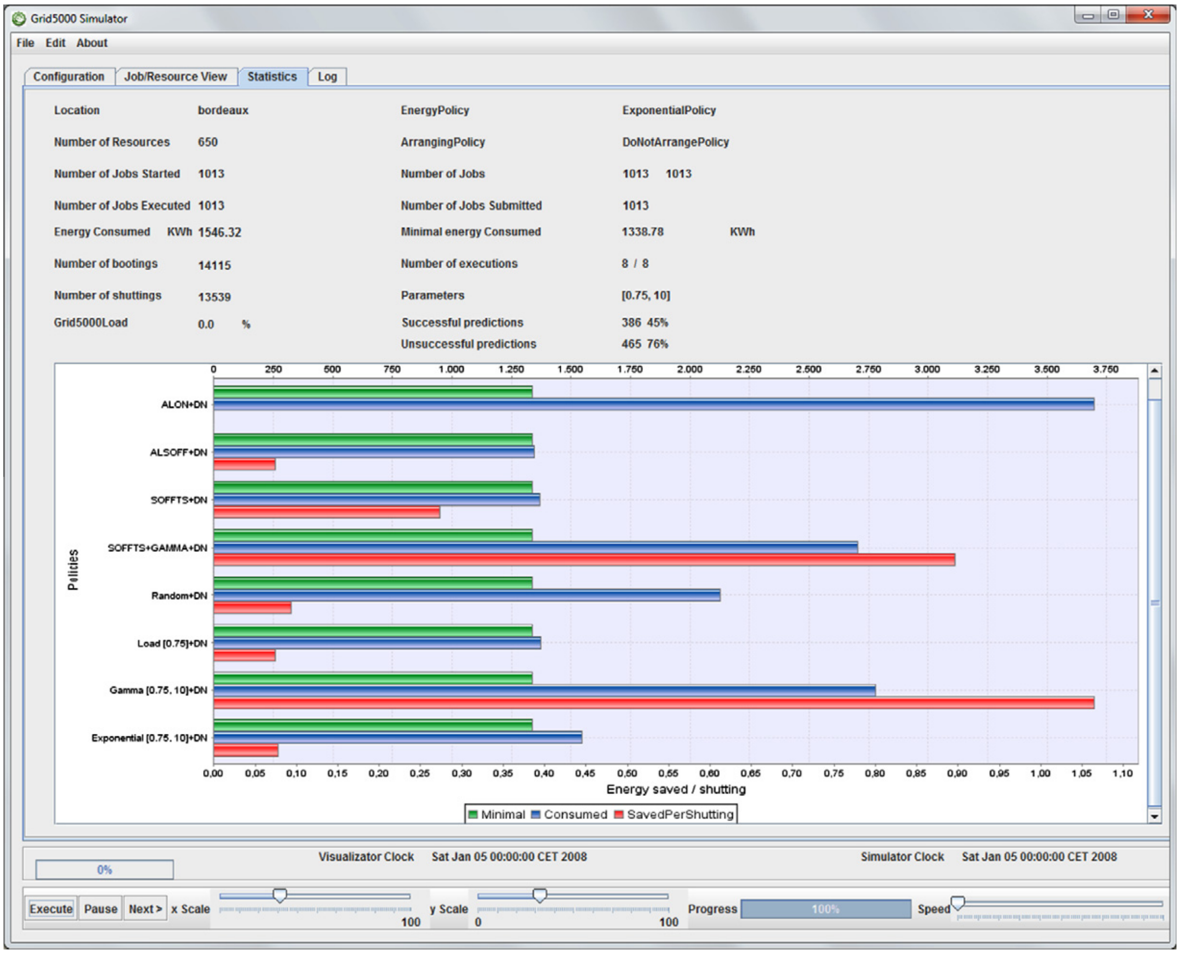
General results compare all possible combinations of energy policies and arranging policies. Grid'5000 Toolbox enables researchers to run a batch of simulations while defining the parameters of each policy. This paper presents a summary of 324 different simulations as follows:

- two different periods of six months. From 1st January, to 30th June and from 1st July to 31st December 2008.
- two arranging policies, *Do Nothing* and *Simple Aggregation of Jobs*
- the seven energy policies listed in Section 3.
- each parameterizable energy policy has been run with various values of for several parameters as follows:
  1. *Load* policy. Load threshold parameter from 0.0 to 1 by 0.3. A total of four scenarios.
  2. *Exponential* and *Gamma*. Threshold probability parameter from 0.0 to 1 by 0.3, and window size from 2^0 to 2^8. Hence there are thirty-six different scenarios for each policy.

From the 324 setups run, the best energy savers have been selected for each policy. Tables 1 and 2 show selected results for the two periods.

With respect to the first period, the minimal energy consumable is 149,202 kW h for a total of 74,035 deployed jobs, and the current energy consumed by Grid'5000 is 217,803 kW h. It can be seen that the simplest energy policy, *Always Off*, is the best in terms of energy saving. However, it is the policy which forces the highest number of power cycles, and hence the stress on the hardware is the greatest. *Load* policy with 0.9 threshold returns very similar results for energy saving and number of power cycles.

[2] The theoretical minimum energy consumed is the sum of the energy needed to run all the jobs of a period. The consumption by the *Idle*, *Booting* and *Shutting Down* states is not computed.



Location	bordeaux	EnergyPolicy	ExponentialPolicy
Number of Resources	650	ArrangingPolicy	DoNotArrangePolicy
Number of Jobs Started	1013	Number of Jobs	1013 1013
Number of Jobs Executed	1013	Number of Jobs Submitted	1013
Energy Consumed KWh	1546.32	Minimal energy Consumed	1338.78 KWh
Number of bootings	14115	Number of executions	8 / 8
Number of shuttings	13539	Parameters	[0.75, 10]
Grid5000Load	0.0 %	Successful predictions	386 45%
		Unsuccessful predictions	465 76%

0%

Visualizator Clock Sat Jan 05 00:00:00 CET 2008

Simulator Clock Sat Jan 05 00:00:00 CET 2008

Progress 100%

Fig. 9. Statistics tab presenting results for a batch of simulations.

Table 1
Selected results for Bordeaux from 1st January to 30th June, sorted according to energy consumed.

Energy policy		Arranging policy	Boot. + Shutt.	Energy consumed (kW h)	Simulated vs		Saved per shutt.
Name	Params				Min	Curr	
Alwz Off	n/a	*SA*	1,668,900	149,396	100.13	68.59	0.04
Load	[0.9]	*SA*	1,569,934	150,304	100.74	69.01	0.04
S.Off Ts	n/a	*SA*	653,856	150,527	100.89	69.11	0.10
Exp.	[0.9, 16]	*SA*	1,047,198	151,733	101.70	69.67	0.06
S.Off Rdm	n/a	*SA*	890,195	157,645	105.66	72.38	0.07
Gamma	[0.9, 1]	*SA*	183,850	186,929	125.29	85.82	0.17
Alwz On	n/a	*SA*	0	217,921	146.06	100.05	0.00

Table 2
Selected results for Bordeaux from 1st July to 31st December, sorted according to energy consumed.

Energy Policy		Arranging policy	Boot.+ Shutt.	Energy consumed (kW h)	Simulated vs		Saved per shutt.
Name	Params				Min	Curr	
Alwz Off	n/a	*SA*	2,365,598	169,364	100.80	73.61	0.03
S.Off Ts	n/a	*SA*	1,584,070	169,894	101.11	73.84	0.04
Load	[0.9]	*SA*	2,104,456	170,109	101.24	73.93	0.03
S.Off Rdm	n/a	*SA*	1,321,702	173,323	103.15	75.33	0.04
Exp.	[0.9, 1]	*SA*	521,878	175,945	104.71	76.47	0.10
Gamma	[0.9, 1]	*SA*	957,784	193,180	114.97	83.96	0.04
Alwz On	n/a	*SA*	0	230,093	136.94	100.00	0.00

Switch Off Ts policy is the only policy that accesses the agenda for future reservations before deciding what to do. This fact explains why the percentage of prediction success is the greatest. It also returns very good results in terms of energy saving, but with

Table 3
Comparison of the two arranging policies comparison for Bordeaux and a selected set of energy policies.

Energy policy		Arranging policy	Boot.+ shutt.	Energy consumed (kW h)	Simulated vs		Saved per shutt.
Name	Params				Min	Curr	
Load	[0.9]	*SA*	1,569,934	150,304	100.74	69.01	0.04
Load	[0.9]	*DN*	1,561,122	154,141	103.31	70.77	0.04
Load	[0.6]	*SA*	902,932	158,205	106.03	72.64	0.07
Load	[0.3]	*SA*	299,304	177,839	119.19	81.65	0.13
Load	[0.6]	*DN*	892,656	186,406	124.94	85.58	0.04
Load	[0.3]	*DN*	292,025	208,141	139.50	95.56	0.03
Load	[0.0]	*SA*	650	216,386	145.03	99.35	2.18
Load	[0.0]	*DN*	650	216,531	145.13	99.42	1.96
S.Off Rdm	n/a	*SA*	890,195	157,645	105.66	72.38	0.07
S.Off Rdm	n/a	*DN*	879,681	183,073	122.70	84.05	0.04

Table 4
Summary of environmental and economic results for Bordeaux.

Energy policy	Energy consumed (kW h)	Saved energy (kW h)	Euros saved	CO_2 kg saved	Energy Consumed (kW h)	Saved energy (kW h)	Euros saved	CO_2 kg saved	Euros saved	CO_2 kg saved
Alwz S.Off	149,396	68,525	9593	18,776	169,364	60,729	8502	16,640	18,096	35,416
Load	150,304	67,617	9466	18,527	170,109	59,984	8398	16,436	17,864	34,963
S.Off Ts	150,527	67,394	9435	18,466	169,894	60,198	8428	16,494	17,863	34,960
Exp.	151,733	66,188	9266	18,135	175,945	54,148	7581	14,836	16,847	32,972
S.Off Rnd	157,645	60,276	8439	16,516	173,323	56,770	7948	15,555	16,386	32,070
Gamma	186,929	30,992	4339	8492	193,180	36,912	5168	10,114	9507	18,606
Alwz On	217921	0	0	0	230093	0	0	0	0	0
	First period				Second period				Total	

a much lower number of power cycles: about 40% of the number of power cycles of the *Always Off* policy. This implies a great advantage over *Always Off* and *Load*-0.9 policies.

Statistical-based policies, *Exponential* and *Gamma*, perform reasonably well in terms of the percentage of prediction success (as expected), particularly the *Gamma* policy. Energy saving results for the *Exponential* policy are a bit worse than previous policies, and quite high in terms of power cycles. On the other hand, the *Gamma* policy performs modestly in terms of energy saving, although the number of power cycles is the lowest, just 11% of *Always Off*, and hence energy saved per shutting down is the greatest, and the stress imposed on the hardware is the lowest.

With respect to the second period, similar results are found. Thus, the minimal energy consumable is 168,024 kW h for a total of 271,149 deployed jobs, and the current energy consumed by Grid'5000 is 230,087 kW h. It is worth noting that in this case, the *Exponential* policy achieves better results than the *Gamma* policy. This fact can be explained by taking into account that the number of jobs deployed during the first period is one quarter of the number of jobs deployed during the first period.

6.2. Scheduling arranging policies comparison

A set of energy policies has been selected in order to compare results between the two arranging policies: *Do Nothing* and *Simple Aggregation of Jobs*. Results are shown in Table 3.

Notice that the latter policy, *Simple Aggregation of Jobs* is consistently the best since it provokes fewer power cycles, saves more energy, and the energy saved per shutting is increased in general. Therefore, if jobs are arranged, even with simple policies, the results are much better than allowing users to choose resources.

6.3. Environmental and Economic results

In order to summarize energy-saving results, the costs and CO_2 savings are computed for Bordeaux in Table 4. For the computation of these values, a price of 0.14 euros per kW h and a CO_2 generation of 0.234 kg per kW h are considered.

Note that 18,000 euros and 35 tons of CO_2 per year can be saved by implementing the best energy saving policy. Hence, extrapolating results to all 9 locations of Grid'5000, (Bourdeaux, Grenoble, Lille, Lyon, Nancy, Orsay, Rennes, Sophia-Antipolis and Toulouse), **162,000 euros** and **318 tons of** CO_2 per year could be saved.

In terms of energy, up to 129,254 kW h could be saved for Bordeaux and by extrapolating this result to all 9 locations, 1,163,286 kW h. To illustrate how large this quantity of energy really is, it is equivalent to 78 journeys of AVE (high speed rail) train from Madrid to Barcelona, and it is equivalent to the daily energy consumption of 61,314 citizens in the Euro zone.

7. Conclusions and Future work

Various methodologies for tackling energy saving in grid computing environments, which could easily be applied to data centres and massive computing environments are presented.

We have empirically proven that a suitable policy in grid computing could save a considerable mount of energy and reduce the pollution of CO_2 in the atmosphere.

The authors are planning to apply these techniques to these environments in the future, in addition to contributing towards the improvement of energy and arranging policies and their adjustment to new computing environments.

A recent upgrade of Grid'5000 toolbox enables us to retrieve data from the cluster located at *Centro Informático Científico de Andalucía* (CICA), Spain, and hence future work will include external sites for comparison with Grid'5000 sites.

Acknowledgement

This research is supported by the Spanish Ministry of Science and Innovation R& D project ARTEMISA (TIN2009–14378-C02–01)

References

Ruth, S. (2009). Green IT more than a three percent solution? *IEEE Internet Computing, 13*, 74–78.

Sanchez, M., Wong, H., Berard, S., & Koomey, J. (2011). Implications of historical trends in the electrical efficiency of computing. *IEEE Annals of the History of Computing, 33*, 46–54.

Koomey, J. G. (2008). Worldwide electricity used in data centers. *Environmental Research Letters, 3*, 034008.

Ren, G., Tune, E., Moseley, T., Shi, Y., & Hundt, R.S.R. (2010) Google-Wide profiling: a continuous profiling infrastructure for data centers, *Micro IEEE 30*, 65–79.

Tseng, L.-Y., Chin, Y.-H., & Wang, S.-C. (2009). A minimized makespan scheduler with multiple factors for Grid computing systems. *Expert Systems with Applications, 36*, 11118–11130.

Orgerie, A.-C., Lefèvre, L., Gelas, J.-P.(2008). Save Watts in your grid: green strategies for energy-aware framework in large scale distributed systems. In: *2008 14th IEEE International Conference on Parallel and Distributed Systems*, pp.171–178.

Idinfor, http://madeira.lsi.us.es/GrupoIdinfor, 2011.

Grid Toolbox, http://madeira.lsi.us.es/proyectosWeb/grid.php, 2011.

Martin-Michiellot, S. (2008).JScience, a general scientific API in Java, Practice.

Reese, G. (2000). *Database Programming with JDBC and Java* (2nd ed.). O'Reilly & Associates, Inc..

Keith, M., & Schincariol, M. (2006). *Pro EJB 3: Java Persistence API (Pro)*. Berkely, CA, USA: Apress.

Bertie, A. (2002). Java applications for teaching statistics. *MSOR Connections, 2*, 78–81.

Gupta, S. (2011). *Pro Apache Log4j* (2nd ed.). Apress.

Khan, A. (2010). JExcel: A Java library for reading/writing Excel, Accessed Online at http://jexcelapi.sourceforge.net/.

Green Grid, T. (2011). Recommendations for measuring and reporting overall data center efficiency Version 2 – Measuring PUE for Data Centers.