Save Watts in your Grid: Green Strategies for Energy-Aware Framework in Large Scale Distributed Systems

Anne-Cécile Orgerie, Laurent Lefèvre, Jean-Patrick Gelas
INRIA RESO - Université de Lyon - LIP (UMR CNRS, INRIA, ENS, UCB)
École Normale Supérieure - 46, allée d'Italie - 69364 LYON Cedex 07 - FRANCE ,
laurent.lefevre@inria.fr, {annececile.orgerie|jean-patrick.gelas}@ens-lyon.fr

Abstract

While an extensive set of research project deals with the saving power problem of electronic devices powered by electric battery, few have interest in large scale distributed systems permanently plugged in the wall socket. However, a rapid study shows that each computer, member of a distributed system platform, consume a substantial quantity of power especially when those resources are idle. Today, given the number of processing resources involved in large scale computing infrastructure, we are convinced that we can save a lot of electric power by applying what we called green policies. Those policies, introduced in this article, propose to alternatively switch On and Off computer nodes in a clever way.

1 Introduction : Understanding large-scale distributed systems usage

The question of energy savings is a matter of concern since a long time in the mobile distributed systems. However, for the large-scale non-mobile distributed systems, which nowadays reach impressive sizes, the energy dimension just starts to be taken into account.

Some previous works on operational Grids [8] show that grids are not utilized at their full capacity. Then, to better understand stakes and potential savings with the scaling effects, we focus on the utilization and the energy analysis of experimental Grids. By relying on Grid5000[1][1], a french experimental Grid platform, we obtain a case study from which we were able to get a comprehensive view of our concern.

[1]Some experiments of this article were performed on the Grid5000 platform, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (http://www.grid5000.fr). This research is supported by the INRIA ARC GREEN-NET project (http://www.ens-lyon.fr/LIP/RESO/Projects/GREEN-NET/).

Grid5000 is an experimental testbed for research in grid computing which owns more than 3400 processors geographically distributed on 9 sites in France. This platform can be defined as a highly reconfigurable, controllable and monitorable experimental Grid equipment. The Grid5000 utilization is very specific. Each user can indeed reserve in advance some nodes and then during its reservation time, he has root rights on these nodes and can even deploy his own system images, collect data, reboot and so on. Nodes are entirely dedicated to the user during his reservation.

We analyze the node reservation traces of Grid5000 for each site over a one-year period (the 2007 year). Figure 1 shows reservations trace of one site (Toulouse). The plain line indicates the number of reservations per week (we call "job" a resource reservations). For each week, we represent the time during which some cores are *dead*: they are down; when they are *suspected*: they do not work properly; when they are *absent*: they do not answer; and when they are *working*: a reservation is running. For this site, the real percentage of work time is 50.57%.

We also see on Figure 1 that during some weeks, the usage of the site is low, but the real matter of concern of such a Grid is to be able to support burst periods of work and communication specially before well-known deadlines and we can see that such periods exist.

Based on this analysis, we realize that the energy consumption can be reduced when the platform is not used. A framework able to control the Grid nodes must deal with :

- switching OFF unused nodes;
- predicting nodes usage in order to switch ON the nodes which are required in a near future;
- aggregating some reservations to avoid frequent ON/OFF cycles.

We propose an energy saving model applying these aspects and we design its implementation through the

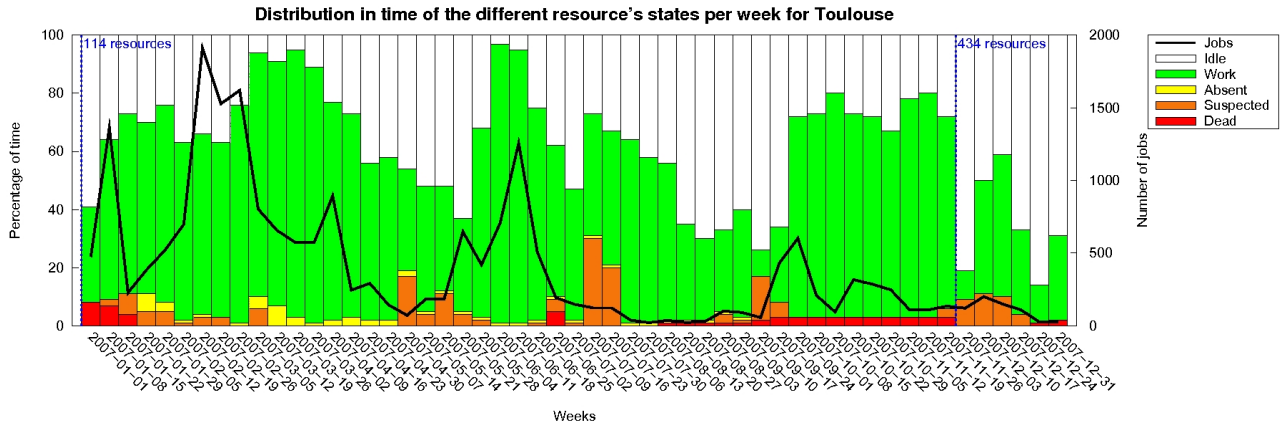

Figure 1. Global weekly diagram for Toulouse

Energy-Aware Reservation Infrastructure (EARI). We conduct some experiments on Grid5000 platform to validate our approach.

This paper is organized as follows. Section 2 presents our Energy Aware Reservation Infrastructure (EARI). We describe the resource managing algorithms of EARI in section 3 and present our first experimetnal validations in section 5. Section 6 present some related works and last section gives our conlusions and future works.

2 Proposition for an Energy-Aware Reservation Infrastructure (EARI)

2.1 Global architecture

In the context of large-scale distributed systems, in order to reduce the global energy consumption, we need to directly act on the nodes, on the network devices and on the scheduler. This paper presents an On/Off algorithm for the resources managed by the scheduler which is also the resource manager.

Figure 2 presents the global architecture of our infrastructure. Each user is connected to a portal to submit a reservation. The scheduler processes the submission and validates it. Then it manages the resources and gives access to the resources to the users who have made reservations on them according to the scheduler agenda. Energy sensors monitor energy parameters from the resources (which can be nodes, routers, etc.) and these data are collected by our infrastructure. Data are used to compute "green" advices which are sent to the user in order to influence his reservation choice. Our infrastructure computes also the consumption diagrams of the past reservations and it sends them as a feedback on the portal, so the users can see them. And last, but not least, it decides which resources should be on and which resources should be off.

2.2 Energy monitoring

Our objective is the measurement of the power consumption of the Grid nodes in Watts in order to exhibit the link between electrical cost and applications or processes.

In order to measure the real consumption of some machines, we use a watt-meter provided by the SME Omegawatt[2]. This equipment works as a multiplug adapter: we plug six nodes on it and we obtain every second the consumption of each node via a serial link.

We have dynamically collected the consumption in Watts of six different nodes representing the three hardware architectures available on Lyon site : two IBM eServer 325 (2.0GHz, 2 CPUs per node), two Sun Fire v20z (2.4GHz, 2 CPUs per node) and two HP Proliant 385 G2 (2.2GHz, 2 dual core CPUs per node).

We observe that some nodes reach impressive powers consumption during the boot period (300 to 400 Watts). They also have a high idle power consumption (not less than 190 Watts). Finally, they also consume a quantity of power not negligible when shutting down (+20 Watts). Other experiments we made on these nodes show that an intensive disk access application (measured with `hdparm` running) consume almost +10 Watts. A high performance network communication (measured with `iperf` running) consumes between +20 and +22 watts and finally, a CPU intensive application (measured with `cpuburn` running) cost between +20 and +26 Watts. These experiments represent a typical life of an experimental Grid node, *i.e* nodes are down but plugged in the wall socket, boot, have intensive disks ac-

[2]Omegawatt is a SME established in Lyon (`http://www.omegawatt.fr/gb/index.php`).

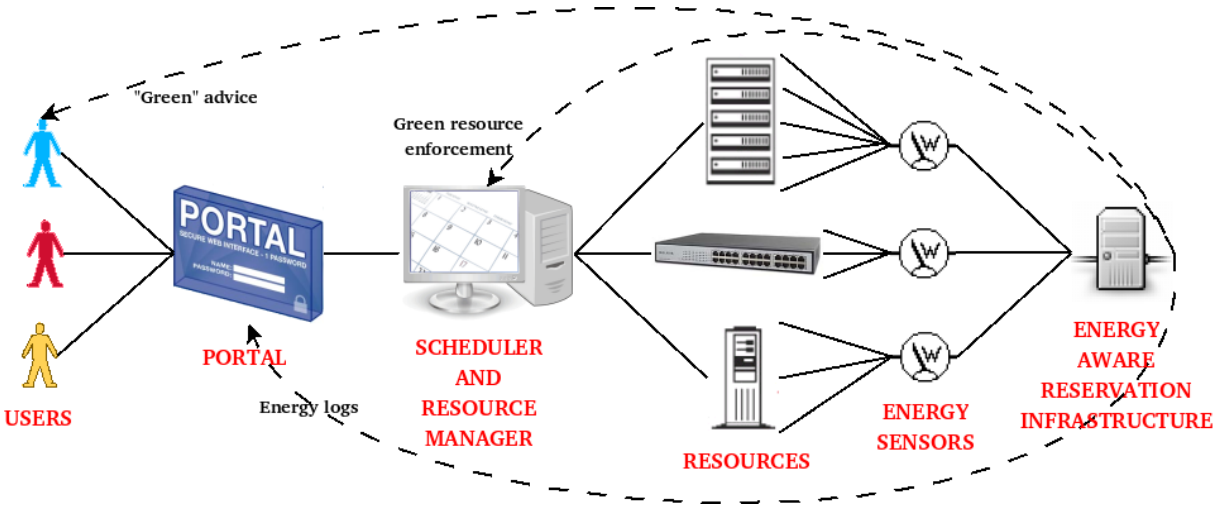

Figure 2. Global architecture

cesses, experiment high performance communication, have intensive CPU usage, and stay idle for long period of time.

These results show the impact on energy usage resulting from node utilization. Then, we use this analysis to design an energy-aware reservation infrastructure.

3 The resource managing algorithms of EARI

3.1 Definitions of the EARI components

We define a reservation R as a tuple: (l, n, t_0) where l is the length in seconds of the reservation, n is the required number of resources and t_0 is the wished start time. N denotes the total number of resources managed by the scheduler. So we should always have $n \leq N$ and $t_0 \geq t$ where t is the actual time and $l \gg 1$ to get a valid reservation. For example, in the case of a large-scale distributed system, a reservation is a reservation of n nodes during l seconds starting at t_0.

When a reservation is accepted by the scheduler, it writes it down into the agenda. The agenda contains all the future reservations. The history contains all the past and current reservations. Then, reservation is moved from the agenda to the history when started.

Lets P_{idle} refers to the power consumption in Watts of a given resource when it is idle (this value can vary from one resource to another) . P_{OFF} refers to the consumption in Watts of a given resource when it is off ($P_{OFF} < P_{idle}$). $E_{ON \to OFF}$ ($E_{OFF \to ON}$) refers to the required energy (in Joules) for a given resource to switch between On to Off modes (Off to On modes respectively). Figure 3 illustrates these definitions.

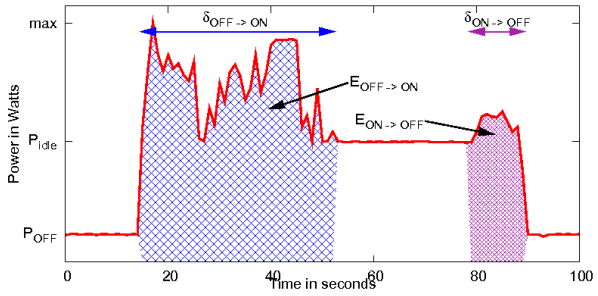

Figure 3. Booting and shutting down of a resource

3.2 Principle of the resource managing algorithm of EARI

We split our algorithm into two parts: when a reservation is submitted and when a reservation ends.

We will show the algorithm used for a reservation arrival: $R = (l, n_0, t_0)$. At t_0, we know that there will be at least n busy resources (because of previously arrived reservations). So, first of all, we wonder whether this reservation is acceptable, ie. $n_0 \leq N - n$. If it is not acceptable, we compute the earliest possible start time after t_0 (by taking into account the reservations which are already registred in the agenda) which is called t_1.

Then, we estimate different amounts of energy consumption by R if it starts:

- at t_0 (or t_1, if t_0 was not possible; t_1 is the next possible start time);

- just after the next possible end time (of a reservation) which is called t_{end};
- l seconds before the next possible start time which is called t_{start};
- during a slack period (time ≥ 2 hours and usage under 50%), at t_{slack}.

We will see on the next sections the prediction algorithms. In order to achieve these estimations, we need to compute: t_1 (done previously), t_{end}, t_{start} and to estimate t_{slack}. Our goal is to aggregate the reservations in order to avoid bootings and turnings off which consume energy. Our infrastructure does not impose any solution, it just offers several of them and the user chooses.

3.3 The EARI algorithm for the resource allocation

In order to calculate t_{end}, we look for the next reservation end in the agenda and we verify if it is possible to start R at that time (enough resources for the total length). If it is not possible, we look for the next one in the agenda and so on. t_{end} is then defined as the end time of this reservation.

In the same way, we calculate t_{start} by looking for the next reservation start time in the agenda and we check out if it is possible to place R before (this start time should then be at least at $t + l$ where t is the current time and l the duration of R). If the found start time does not match, we try the next one and so on.

An enhancement consists in finding several possible reservation end times and start times. We can then take the ones which minimize the energy consumption: the ones with which we should the least turn on and turn off resources.

Finally, we give all these estimations to the user (energy estimations and corresponding start times) who selects its favorite solution. The reservation is then written down in the agenda and the reservation number is given to the user. With this approach, the user can still make his reservation exactly when he wants to, but he can also delay it in order to save energy. Through this approach, it will raise user awareness upon energy savings.

The scheduler makes the resource allocation by choosing the resources with the smallest power coefficient. That coefficient is calculated depending on the mean power consumption of the resource (calculated during reservations on a great period of time). Thus, a resource which consumes few energy will have a big power coefficient and then will be choosed in priority by the scheduler. Indeed, resources are not identical (different architectures,...), and then show different consumptions.

This allocation policy is used when we give resources for a reservation without constraints. In fact, when the scheduler places a reservation just after another (by using t_{end} or not) or just before another, it allocates the resources which are already up (and in priority those which have the biggest power coefficient). Moreover, the user can explicitly choose certain resources, so in that case, this policy is not applicable. The power coefficient is calculated when the resource is added to the platform and will not change after.

3.4 The EARI algorithm for the resource release

First of all, we compute the total real consumption of this reservation. We give this information to the user and we store it in the history for the prediction algorithms. Moreover, we compute the error made when we have estimated the consumption of this reservation with the corresponding start time: this is the difference between the true value and the predicted one. We will use it in the next section to compute a feedback error in order to improve our estimation algorithms.

An *imminent* reservation is a reservation that will start in less than T_s seconds in relation to the present time. The idea is to compute T_s such as it will be the minimum time which ensures an energy saving if we turn off the resource during this time. In fact, we define T_s so that if we turn off a resource during T_s seconds, we save E_s Joules. E_s is a fixed energy, it is the minimum energy that we don't go to a lot of trouble to save it.

To this definition, we add a special time, denoted by T_r, which is related to the resource type. For example, if we turn on and turn off often an Ethernet card, it has not the same consequences, in terms of hardware wear out, compared to the same for a hard disk. The hard disk is indeed mechanical and can support a limited number of ignitions. Thus, we should not turn it off too often or too quickly. So T_r reflects this concern and differs from one resource to another.

So, if we denote $\delta_{tot} = \delta_{ON\to OFF} + \delta_{OFF\to ON}$, T_s is defined by:

$$T_s = \frac{E_s - P_{OFF} \times \delta_{tot} + E_{ON\to OFF} + E_{OFF\to ON}}{P_{idle} - P_{OFF}} + T_r$$

As we can see, T_s varies from one resource to another because it depends on P_{idle}, P_{OFF}, $\delta_{ON\to OFF}$, $\delta_{OFF\to ON}$, $E_{ON\to OFF}$, $E_{OFF\to ON}$ and T_r which depend on the resource. We can fix $E_s = 10$ Joules for example.

We can notice that we should have: $T_s - \delta_{tot} \geq 0$. We want indeed to have at least enough time to turn off the resource and turn on it again. Now, we look for the freed resources that have an imminent reservation. We can see this in the agenda as depicted on Figure 4. These resources are

considered as busy and are left turned on (top part of Fig. 4). During this idle period, we loose less than E_s Joules per resource and then they are used again.

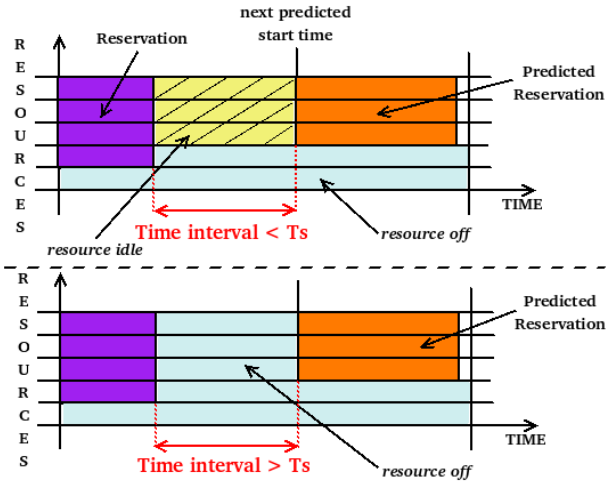

Figure 4. An agenda example which shows the role of T_s

We look for other awake resources: resources which are waiting for a previous estimated imminent reservation. For these m resources up, we need to estimate when will occur the next reservation and how many resources it will take. We call this reservation $R_e = (l_e, n_e, t_e)$. We can now verify if R_e is imminent. If it is not the case, all the remaining resources are turned off (as in the bottom part of Figure 4).

If R_e is imminent, we look for $min(m, n_e)$ resources or less that can accept this potential reservation: they are free at t_e for at least l_e seconds. We keep these resources awake during $T_s + T_c$ seconds and we turn off the other ones. T_c is a fixed value that corresponds to the mean computation time of a reservation for the scheduler. It is the mean time between the user request and the reservation acceptation by the scheduler (it includes among other things the time to compute the energy estimations and a minimum time to answer for the user).

At the next reservation arrival, we will compute the estimation errors we have done and we will use them as feedback in our prediction algorithms. Moreover, if there are idle resources (which are turned on without any reservation) and if the reservation which is just arrived is not imminent, we turn off the idle resources.

3.5 Global mechanisms: selection of specific resources and load balancing

Before a reservation which is written down in the agenda, if some resources are not turned on, the scheduler puts them on at least $\delta_{ON \to OFF}$ seconds before the beginning of the reservation. In this general infrastructure, we have not distinguished the different resources to make it clearer: in fact, different resources have different associated consumptions and different components (Ethernet cards, hard disks, etc.). But, instead of giving just a number of resources, the user can give a list of wished resources with the wanted characteristics.

Moreover, we add to our infrastructure a load balancing system which ensures that the scheduler will not always use the same resources. This load balancing system includes a topology model (the geographic position of the different resources). This mechanism allows us to distribute geographically the reservations. The reserved resources are not "sticked" together, so we have less heat production. Then when a reservation is submitted, the scheduler does not allocate resources to it at random but with following this load balancing policy.

4 Predictions

The efficiency of EARI, compared to a simple algorithm where the resources are put into sleep state from the time that they have nothing to do, 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. But our prediction algorithm should remain sufficiently simple in terms of computation in order to be efficient and applicable during reservation scheduler run time.

We need to predict three different type of values:

- the next reservation (start time, length and number of resources),
- the energy consumed by a given reservation,
- when will occur the next slack period (time $\geq$ 2 hours and usage under 50%).

We describe in details our prediction algorithms in [11]. The general idea is to compute average values by taking some recent past values. We also use feedbacks which are the average errors made by computing the lasts predictions. Those prediction algorithms [11] remain time efficient and are sufficiently accurate aswe will see in section 5.2.

5 Experimental validation of EARI

5.1 The replay principle

To evaluate EARI, we conduct experiments based on a replay of the 2007 traces of the Grid5000 platform. In a first

step, we do not move the reservations: we always respect the reservation characteristics given by the user. So we can fully test our prediction algorithm. In a second step, we move the reservations on a time scale by respecting several policies. The results show that EARI makes energy savings.

5.2 Without moving the reservations: validation of the prediction algorithm

Here, we take the example of Bordeaux on the 2007 year (Fig. 5 and Fig. 6). Figure 5 shows the percentages of energy consumption of EARI with prediction and the percentages of energy consumption of EARI without prediction where T_s varies from 120 seconds to 420 seconds and P_{idle} (the power consumed by a single resource when it is idle: on but not working) is 100, 145 or 190 Watts. These percentages are given in relation to the energy consumption of EARI by knowing the future: in that ideal and not reachable case we don't need any prediction algorithm because we always know when to turn on and turn off resources. In fact, this is the theoritical lower bound.

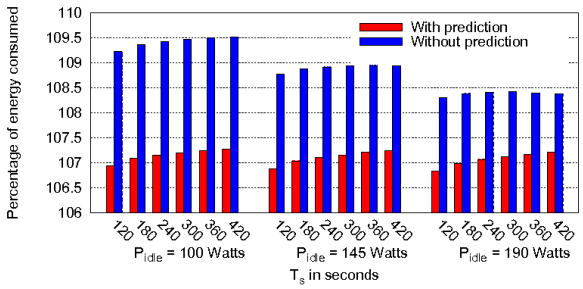

Figure 5. Percentage of energy consumption by using EARI in relation to the energy consumed by knowing the future

Based on our experimental measures, we set $P_{work} = 216$ Watts, $P_{OFF} = 10$ Watts and $\delta_{ON \to OFF} + \delta_{OFF \to ON} = 110$ seconds. Also based on our measures, we set P_{idle} to 190 Watts, but we make P_{idle} vary to simulate the future capacity of EARI to shut down resource's components (like a core or a disk for a node for example). In the same way, we make T_s vary to simulate the future possibility to use hibernate modes (T_s is at least equal to the time needed to boot a resource plus the time to shut down it, so if we use suspend to disk or suspend to RAM mechanisms, it will decrease T_s).

These percentages are in relation to the aimed used energy: the energy we would consume if we knew the future (it is as if our prediction algorithm made always perfect predictions), so it is the lower limit we try to be close to. We see that EARI with prediction is better than without prediction in all the cases. However, we have still room for maneuver to improve our prediction algorithm in order to be closer to the aimed case.

Figure 6 shows the surprise reservations impact for EARI with and without prediction. The surprise reservations are reservations that arrive less than T_s seconds after we have turned off some resources (which can instead have been used for these arriving reservations). This is the reason why we are not closer to the future known case on Figure 5. As expected, EARI with prediction has better results because it tries to predict such reservations, but it is not possible to achieve this goal perfectly (the future is not known!).

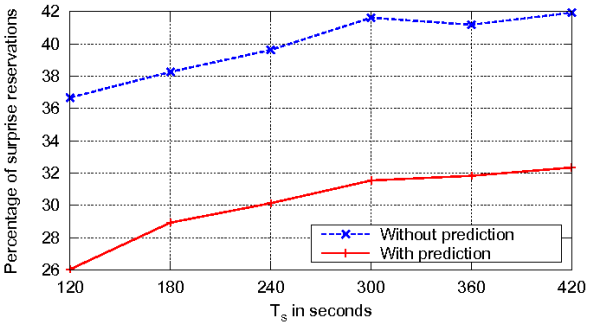

Figure 6. Percentage of surprise reservations in relation to total reservation number

5.3 By moving the reservations: validation of our "green" policy

Now, we evaluate the complete EARI by allowing our simulator to move the jobs at a better date. We design six policies to conduct our experiments:

- *user*: we always select the solution that fits the most with the user's demand (we select the date asked by the user or the nearest possible date of this asked 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 doesn't delay the reservation from the initial user's demand for more than 24 hours, otherwise we use the *user* policy.

These policies simulate the behavior of real users: there is a percentage of "green" users who follow the advice given by EARI. Maybe they do not want to delay too long their reservation as in the *deadlined* policy. And some users do not want to move their reservation even if they can save energy by doing this, that is the *user* policy. The *green* policy illustrates the case of an administrator decision: the administrator always chooses the most energy-efficient option.

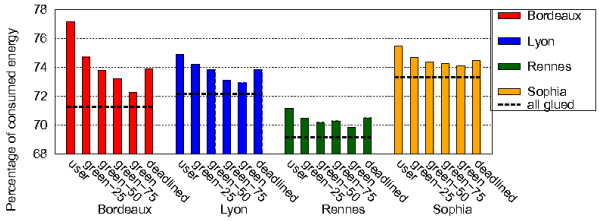

Figure 7. Energy consumption of Grid5000 with EARI with $T_s = 240$ s and $P_{idle} = 100$ watts compared with the consumption when all the nodes are always ON

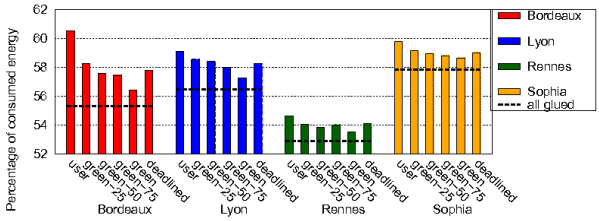

Figure 8. Energy consumption of Grid5000 with EARI with $T_s = 240$ s and $P_{idle} = 190$ watts compared with the consumption when all the nodes are always ON

Figures 7 and 8 present the energy consumption of some Grid5000 sites (Bordeaux, Lyon, Rennes, Sophia) benefiting from EARI with some low power consuming nodes (100 watts) and current nodes (190 watts currently measured on the Lyon site).

We show that our energy policy (fully-green) is more energy-efficient than any other strategies. The *all-glued* line shows the theoritical (unaccessible) lower bound: when we glue all the reservation without idle periods. Globally, our fully-green strategies allow energy gains close the to the theoritical all-glued scenario.

For example, on Bordeaux site, the *fully-green* policy has moved about 98% of the reservations by an average time of 15 hours per reservation. The *50%-green* policy has moved about 75% of the reservations by an average time of 12 hours.

Figure 9 shows the energy consumption of the Lyon site in kilowatt hour (kWh) for the whole 2007 year in the current case (without energy saving policy), with our *user* policy, with our *50%-green* policy, with our *fully-green* policy and the *all glued* consumption. These values are those represented in percentage on Fig. 8.

P_{idle}	present state	user	50%-green	fully-green	all glued
100	135500	101500	100000	98300	97300
145	154000	101700	100300	98500	97300
190	172500	102000	100800	98700	97300

Figure 9. Energy consumption in kWh for Lyon with $T_s = 240$ s. for several policies for 2007

We can notice that the last value is the same for the three lines, it is normal because the "all glued" consumption represents the case where all the reservations are sticked together without any idle period between them. This is the most energy efficient reservation distribution but this distribution is not really possible. Furthermore, we notice that our *fully-green* policy is near the all-glued case (which is the optimal solution in terms of energy) and that the three values for this policy are near: this is normal because we try to reduce as much as possible the useless idle periods. In the present situation (where all the nodes are always powered on and consume 190 Watts when they are idle), we could save 73800 kWh only for Lyon; this represents the consumption[3] of a TGV covering about 5000 km.

We have not yet taken into account the problem of network presence. Indeed, we turn off nodes that we can wake up by their Ethernet cards, but the system monitoring tools (like Monika, Ganglia or Nagios for Grid5000) need to have periodical answers from these nodes. So by turning off them, they will believe that these nodes are dead although this is not the case. This problem can be solved by proxying techniques (see section 6). The Ethernet cards can be configured to be able to answer such basic requests.

6 Related works

Although energy has been a matter of concern for sensor networks and battery constrained systems since their creation, energy issues are recent for full time plugged systems. A first problem that occurs is how to measure the consumption. We have considered an external watt-meter

[3]The consumption of a TGV train is about 14.79 kWh per km.

to obtain the global consumption of a node. A different approach consists of deducing it from the usage of the node components, by using event monitoring counters [10] for example. Lot of other work on server power management based on on/off algorithm has been done [12], [3]. Some take into account thermal issues [2], [12]. The main issue in that case is to design an energy-aware scheduling algorithm with the current constraints (divisible task or not [2], synchronization [9], etc). Some algorithms include DVFS (Dynamic Voltage Frequency Scaling) techniques [9], [7] and some not [3]. Although we are fully aware that such techniques will be available on all processors in a near future, our work does not include this in a first step presented here. Such techniques are indeed difficult to use in presence of a processor and user heterogeneity especially if we want to design a centralized resource managing algorithm. Virtualization seems to become an other promising track [6]. We have not yet speak about network presence. Lot of work have also been done on the network level to reduce the consumption of Ethernet card and switches ports by adaptively modify the link rate [4] or by turning off ports [5]. The problem of ensuring network presence becomes more obvious with such objectives.

7 Conclusion and future works

This paper presents a first step of our work whose goal is to better understand the usage of large-scale distributed systems and to propose methods and energy-aware tools to reduce the energy consumption in such systems. Our analysis has provided instructive results about the utilization of an experimental Grid over the example of Grid5000.

Next, we have proposed an energy-aware model to reduce the global consumption of a large scale experimental Grid. This infrastructure is efficient and can be easily implemented and deployed. We have presented our first results which validate our energy-aware reservation model.

We are currently working on tools, portals and frameworks proposing these results in a real-time manner to the users and grid middleware. We are working on such a tool that we will integrate on the Grid5000 website. We plan to make further experiments to fully validate our infrastructure and to enhance our prediction algorithm. We also plan to make the same experiments with the whole grid traces including the grid reservation constraints (on several sites at the same time). We will study the possibility to move reservations from one site to another (according to external temperatures parameters for example).

Our long term goal is to incorporate virtualization and DVFS techniques in our infrastructure with the objective to save more energy without impacting performances. Virtualization could also solve the problem of ensuring network presence and answering basic requests from the monitoring tools of large-scale distributed systems or dealing with the high-performance data transport systems.

References

[1] 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.

[2] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centers. In *SOSP '01: 18th ACM symposium on Operating systems principles*, pages 103–116, New York, NY, USA, 2001. ACM.

[3] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ISCA '07: Proceedings of the 34th annual international symposium on Computer architecture*, pages 13–23, New York, NY, USA, 2007. ACM.

[4] C. Gunaratne, K. Christensen, and B. Nordman. Managing energy consumption costs in desktop pcs and lan switches with proxying, split tcp connections, and scaling of link speed. *Int. J. Netw. Manag.*, 15(5):297–310, 2005.

[5] M. Gupta and S. Singh. Dynamic ethernet link shutdown for energy conservation on ethernet links. *Communications, 2007. ICC '07. IEEE International Conference on*, pages 6156–6161, 24-28 June 2007.

[6] F. Hermenier, N. Loriant, and J.-M. Menaud. Power management in grid computing with xen. In *XEN in HPC Cluster and Grid Computing Environments (XHPC06)*, number 4331 in LNCS, pages 407–416, Sorento, Italy, 2006. Springer Verlag.

[7] Y. Hotta, M. Sato, H. Kimura, S. Matsuoka, T. Boku, and D. Takahashi. Profile-based optimization of power performance by using dynamic voltage scaling on a pc cluster. *IPDPS 2006*, 2006.

[8] A. Iosup, C. Dumitrescu, D. Epema, H. Li, and L. Wolters. How are real grids used? the analysis of four grid traces and its implications. In *7th IEEE/ACM International Conference on Grid Computing*, Sept. 2006.

[9] R. Jejurikar and R. Gupta. Energy aware task scheduling with task synchronization for embedded real-time systems. In *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, pages 1024– 1037. IEEE, June 2006.

[10] A. Merkel and F. Bellosa. Balancing power consumption in multiprocessor systems. *SIGOPS Oper. Syst. Rev.*, 40(4):403–414, 2006.

[11] A.-C. Orgerie, L. Lefèvre, and J.-P. Gelas. Chasing gaps between bursts : Towards energy efficient large scale experimental grids. In *PDCAT 2008 : The Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*, Dunedin, New Zealand, Dec. 2008.

[12] R. K. Sharma, C. E. Bash, C. D. Patel, R. J. Friedrich, and J. S. Chase. Balance of power: Dynamic thermal management for internet data centers. *IEEE Internet Computing*, 9(1):42–49, 2005.