Chasing Gaps between Bursts: Towards Energy Efficient Large Scale Experimental Grids

Anne-Cécile Orgerie, Laurent Lefèvre, Jean-Patrick Gelas

Chasing Gaps between Bursts : Towards Energy Efficient Large Scale Experimental Grids

Anne-Cécile Orgerie, Laurent Lefèvre, Jean-Patrick Gelas
INRIA RESO - Université de Lyon - 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

The question of energy savings has been a matter of concern since a long time in the mobile distributed systems and battery-constrained systems. However, for large-scale non-mobile distributed systems, which nowadays reach impressive sizes, the energy dimension (electrical consumption) just starts to be taken into account.

In this paper, we analyze the usage of an experimental grid over a one-year period. Based on this analysis, we propose a resource reservation infrastructure which takes into account the energy issue. We validate our infrastructure on the large scale experimental Grid5000 platform and present the obtained gains in terms of energy.

1. Introduction

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 work on operational Grids [9] show that grids are not utilized at their full capacity. We focus on the utilization and the energy analysis of experimental Grids by relying on the case study of Grid5000[2][1], a french experimental Grid. Based on this analysis, we propose an energy saving model and we conduct some experiments to validate our approach.

Section 2 presents our approach on understanding the usage in large scale experimental Grids over a one year period. In section 3, we describe our model for an energy aware Grid. We present our first validation results of this model in section 5 and link this approach with some related works in section 6. Section 7 concludes this paper and presents some future works.

[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)

2. Understanding Large Scale Experimental Grids usage

Lots of computing and networking equipments are concerned by overall observations on the waste of energy: PCs, switches, routers, servers, etc, because they remain fully powered-on during idle periods. In a grid context, different policies can be applied depending on the level we want to make savings: node level, data center level or network level.

On a node level, we can use Direct Voltage Scaling techniques and frequency scaling techniques to reduce the energy consumption of the CPU. But one can also imagine put into sleep cores or memory benches or disks for example. On cluster and grid levels, different solutions are also possible to reduce the energy consumption like energy-efficient task scheduling, proxying techniques to ensure network presence or resource virtualization. With the scale effect, the potential savings are huge. In order to better understand the stakes and the potential savings with the scaling effects, we need to have an overview at different levels: grid, cluster and node levels.

2.1. The Grid5000 testbed

The Grid5000 platform 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. Its utilization is specific : each user can reserve in advance some nodes and use them as super user in order to deploy his own system image. The node is entirely dedicated to the user during his reservation. So Grid5000 is different from

Site	nb of reservations	nb of cores	nb of core per reservation	mean length of a reservation	real work
Bordeaux	45775	650	55.50	5224.59 s.	47.80%
Lille	330694	250	4.81	1446.13 s.	36.44%
Lyon	33315	322	41.64	3246.15 s.	46.38%
Nancy	63435	574	22.46	19480.49 s.	56.41%
Orsay	26448	684	47.45	4322.54 s.	18.88%
Rennes	36433	714	54.85	7973.39 s.	49.87%
Sophia	35179	568	57.93	4890.28 s.	51.43%
Toulouse	20832	434	12.89	7420.07 s.	50.57%

Figure 1. Grid5000 usage over one-year period: 2007

an operational Grid (exclusive usage, deployment, etc.), but the energy issue is still the same and we can propose solutions which fit for both experimental and operational Grids as well.

2.2. In search of the bursts : A year in the life of an experimental Grid

We analyze the node reservation traces of Grid5000 for each site over a one-year period (the 2007 year). The users can indeed reserve some nodes during a period of time. We have obtained these traces by using the history consultation mechanism of the Grid5000 scheduler OAR[2]. Then, we have made a parser written in Perl to use the 1.2 GBytes of traces and to make the statistics reported below.

We observe that the usage greatly varies from one site to another (Fig. 1) in terms of number of cores, average number of cores per reservation, length of a reservation and the percentage of real work (without including the dead and absent time periods during which the nodes do not consume any energy because they are unplugged).

This heterogeneous usage can be due to geographical purposes (the most distant sites are interesting to conduct communication experiments) and to hardware purposes: each site has different nodes with different architectures (storage, network capabilities...).

2.3. The Grid view in terms of usage

We define the grid reservations as at least two reservations on different sites with the same user and which have at least five minutes in common during their execution time. That is to say, a user has launched at least two reservations on at least two different sites and they are simultaneous for at least five minutes. Fig. 2 shows the number of working hours by week and by site spent by the grid reservations. The graph presents the grid reservations per site in hours for the histogram and in number for the black line. The red line shows the total number of work hour for all the sites per week. This gives a global view of the grid. This figure shows that we cannot apply a particular energy saving policy on each cluster, this policy should be globalized at least for the reservation placement (in term of time and resources). Numerous users need indeed to have coordinated reservations on different clusters as we can see on figure 2.

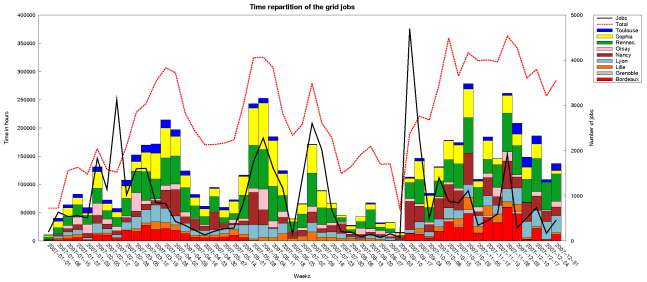

Figure 2. Working hours of the grid jobs per week and per site

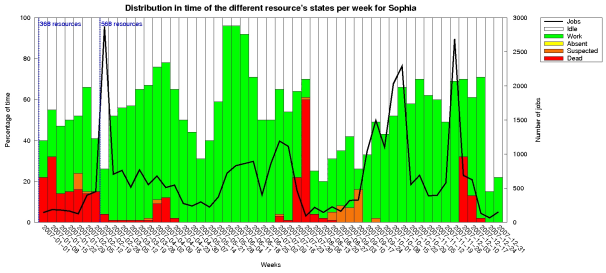

Figure 3. Global weekly diagram for Sophia Grid5000 site

2.4. The cluster view

Figure 3 shows the example of the site of Sophia with 568 cores. The black line indicates the number of reservations per week. For each week, we have represented in red the time during which some cores are dead, that is to say they are down; in orange when they are suspected, they do not work properly; in yellow when they are absent, they do

[2]OAR is a resource manager (or batch scheduler) for large clusters [1]

not answer and in green when they are working (a reservation is running). For this site, the real percentage of work time is 51.43%. We see on figure 3 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.

2.5. The node view in terms of usage

We analyse the weekly repartition of three particular resources: the minimal (Fig 4) (*bad resource*), the median (Fig 5) (*not-so-good resource*) and the maximal resources (Fig 6) (*good resource*) in terms of usage. The minimal (maximal) resource is the resource which provides the minimal (maximal respectively) work time among the resources which are present during the whole monitored period. Indeed, the minimal one is always powered on, but does not work so much, thus it wastes a lot of energy in idle state. The median resource is the resource which is the nearest to the median value of cumulative work over the experiment duration.

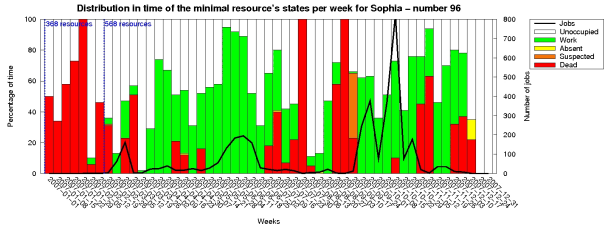

Figure 4. Resource with a minimal usage on Sophia site (*the bad resource*)

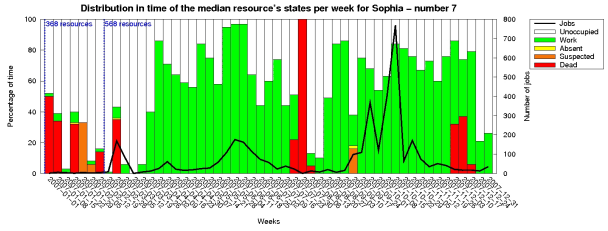

Figure 5. Resource with a median usage on Sophia site (*the not-so-good resource*)

To conclude, we have seen that the platform is used at about 40% in average. However, during some bursting periods, it is used at more than 95%. Yet, between the bursts, the resources are idle during substantial periods, we could

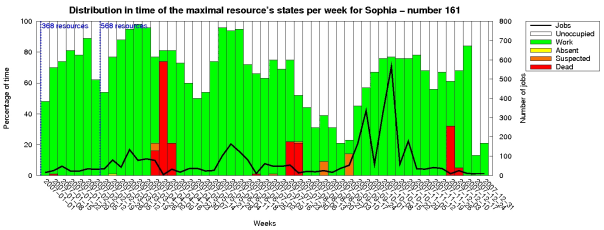

Figure 6. Resource with a median usage on Sophia site (*the good resource*)

thus save energy by intelligently turning them off during specific periods of times.

3. Towards energy aware Grid

In the context of grids, with the mind to reduce the global energy consumption, we can directly act on the nodes, on the network devices and on the scheduler. This paper presents a centralized On/Off algorithm for the resources managed by the scheduler. Each user should be connected to the portal to submit a reservation. The scheduler processes the submission and validates it, then it manages the resources and gives access to them to the users who have made reservations on them according to its agenda. The scheduler treats dynamically the reservations when they are submitted by the user.

3.1. Energy monitoring

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

In order to measure the real consumption of some machines, we use a watt-meter furnished by the SME Omegawatt. This equipment works as a multi-socket: we plug six nodes on it and we obtain the consumption via a serial link (RS232). So we have written scripts that send a request by the serial link to the watt-meter each second, then it converts the answer (the watt-meter accept requests and send results in a specific hexadecimal format) and sends it to a distant collector which stores them. This architecture can easily be adapted to other cluster infrastructures.

Figures 7 and 8 show our results concerning the energy consumption on the site of Lyon. We have dynamically collected the consumption in Watts of six different nodes representing the 3 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 can see that the nodes have a high idle power consumption and that some of them reach impressive powers during their boot and that they consume power even turning off. Other experiments that we have made on the same nodes show that an `Iperf`[3] experiment consumes between 10 and 12 watts more than the `hdparm`[4] upper bound and a `cpuburn`[5] experiment between 0 and 6 watts more than the `Iperf` experiment.

These experiments represent a typical life of an experimental Grid node : nodes down but plugged in the wall socket, booting, having intensive disks accesses (`hdparm`), experimenting intensive high performance network communications (`Iperf`), or having intensive CPU usage (`cpuburn`).

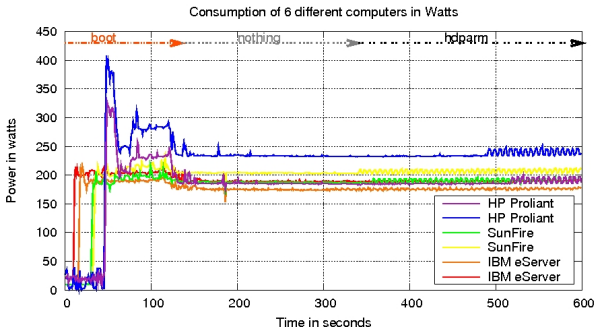

Figure 7. Booting consumption of the nodes

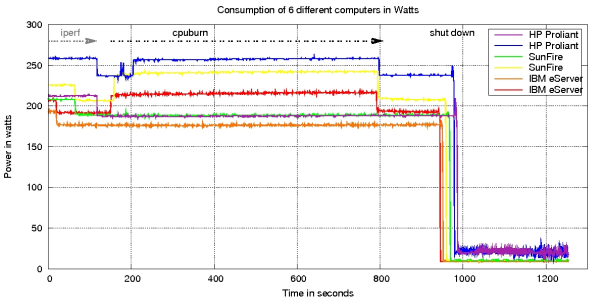

Figure 8. Intensive communication and computing

We can explain the difference of consumption between the two HP Proliants by the fact that the one which consumes the more embeds a powerful programable Ethernet card.

Furthermore, the two Sun Fires are identical in terms of components (same architectures, same features) and in the same way the two IBM eServers are identical. However, we can observe that one Sun Fire consumes more than the other one and that one IBM eServer consumes more than the other one. We should add that the two ones which consume the less are on the bottom of the rack, while the two ones which consume the more are on the top of the rack. So we observe that the position in the rack influences the consumption. This should be due to the air cooling infrastructure position. The more ventilated nodes consume less energy.

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

3.2. Definition of the energy efficiency model

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 $l \geq 1$ to get a valid reservation. For example, in the case of a large-scale distributed system, a reservation is a combination of n nodes during l seconds starting at t_0. In the case of a high-performance data transfer, a reservation is a bandwidth portion which size is n (n can be in Mbps for example) during l seconds starting at t_0.

When a reservation is accepted by the scheduler, it writes it down into the corresponding agenda. Each site indeed has got its own agenda. The agenda contains all the future reservations. The history contains all the past and current reservations. So when a reservation starts, it is deleted from the agenda and added to the history.

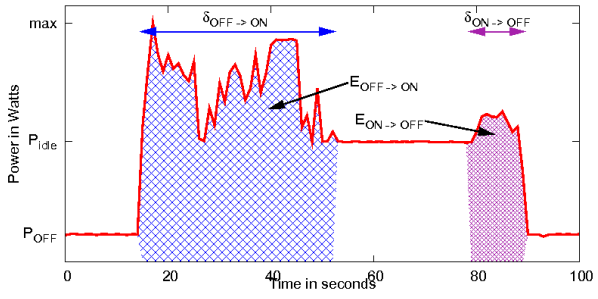

Figure 9. Booting and shutting down of a resource

[3]Iperf is a commonly used network tool to measure TCP and UDP bandwidth performance that can create TCP and UDP data streams (`http://dast.nlanr.net/Projects/Iperf/`).

[4]hdparm is a command line utility for Linux to set and view IDE hard disk hardware parameters (`http://sourceforge.net/projects/hdparm/`). We have made a loop of hdparm instructions to simulate intensive disk accesses.

[5]cpuburn is a tool designed to heavily load CPU chips in order to test their reliability (`http://pages.sbcglobal.net/redelm/`).

P_I refers to the power consumption in Watts of a given resource (it can vary from one resource to another) when it is idle. P_{OFF} refers to the consumption in Watts of a given resource when it is off ($P_{OFF} < P_I$). $E_{ON \to OFF}$ ($E_{OFF \to ON}$) refers to the needed energy (in Joules) for a given resource to switch between On and Off modes (Off and On modes respectively). Figure 9 illustrates these definitions.

So, we can roughly estimate the energy consumption in Joules of a given reservation $R = (l, n, t_0)$:

$$E_m(R) = l \times \sum_{i=1}^{n} P_m(i)$$

where $P_m(i, S)$ is the mean consumption of the resource i.

3.3. Principle of the resource managing algorithm of our model

We split our algorithm into two parts: when a reservation is submitted (section 3.4) and when a reservation ends (section 3.5).

When a reservation arrives ($R = (l, n_0, t_0)$); 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 written down in the agenda) which is called t_1.

Then, we estimate different amounts of energy, the energy consumed 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%, see section 4.4), 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 glue 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.

3.4. 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. 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 will so be choose as a priority by the scheduler. Indeed, resources are not identical (different architectures, ...), so they do not have the same 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 awake (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.

3.5. The resource release

When a reservation ends; M resources are freed. 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.

We need to define an *imminent* reservation: it 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 resistance, compared to the same for a hard disk for example. 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:

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

As we can see, T_s varies from one resource to another because it depends on P_I, 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. These resources are considered as busy and are left turned on. During this active watch, we loose less than E_s Joules per resource and then they are used again.

We look for other awake resources: resources which are waiting for a previous estimated imminent reservation. For these m awake resources (M minus the previous busy ones and plus the other awake resources), 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.

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.

4. Predictions

The efficiency of our model, compared to a simple algorithm where the resources are put into sleep state from the moment 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.

4.1. Estimation of the next reservation

First of all, we take care about 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 reservation's start times for each day are similar to those of the two previous days and to 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 remember 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 the method 2. This method is based on the similarity between the close reservations in term 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:

$$t_e = t + 1/5[t_5 - t_0] + t\_feedback$$

We define the estimations of the length and of the number of resources required by the next reservation similarly.

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 wait for imminent reservations that do not arrive and so they waste energy or they are turned off just before an imminent reservation arrival and so they waste the energy of one halting plus one reboot per resource.

4.2. 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 is less reservations during the night and more during the morning for the Grid5000 traces. So, early in the morning for example, method 2 will certainly have some difficulties to predict the next reservation start time. Thus, 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 $Err_t > 0$. So, 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 want to do. 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. 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).

4.4. Slack periods

A slack period is a period longer than two hours with a usage percentage of the platform inferior to 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 Grid5000 (see section 2.3). So a lot 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.

5. Experimental evaluation

To evaluate our model, we conduct experiments based on a replay of the year 2007 traces (see section 2). As a first approach, we respect the reservation patterns given by the user by not moving the reservations. So we can fully test our prediction algorithm.

As example, we apply our model to the Bordeaux site logs in Fig. 10 and Fig. 11. Figure 10 shows the percentages of energy consumption of our model with prediction

(we use the method 2 described in paragraph 4.1) and the percentages of energy consumption of our model 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 our model 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. Actually, this is the theoritical lower bound.

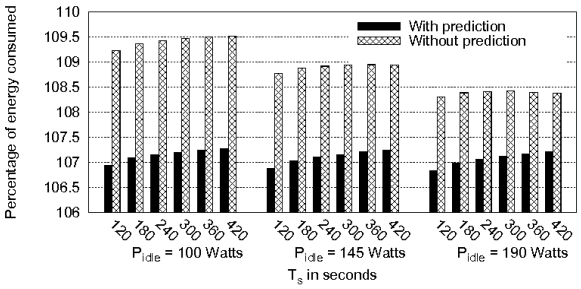

Figure 10. Percentage of energy consumption by using our model in relation to the energy consumed by knowing the future

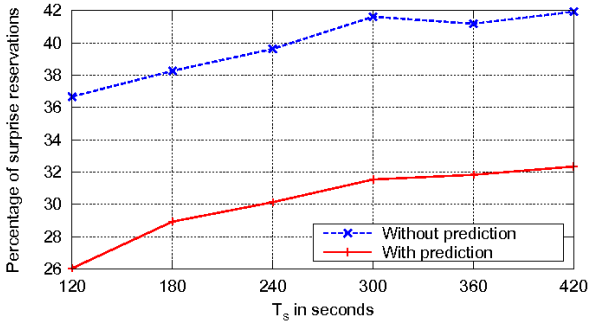

Figure 11. Percentage of surprise reservations in relation to total reservation number

Based on the experimental analysis (Fig. 7), we set $P_{work} = 216$ Watts, $P_{OFF} = 10$ and $\delta_{ON \to OFF} + \delta_{OFF \to ON} = 110$ seconds. These values are a little bit bigger than the mean values shown in Figure 7; this is indeed the worst case for our model.

Based on Fig. 7 results, we can set P_{idle} to 190 Watts, but we make P_{idle} vary to simulate the future capacity of our model 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 our model 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 11 shows the surprise reservations impact for our model 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 10. As expected, our model 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!). We can conclude that a simple prediction model as the one described in section 4.1 is energy efficient.

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 plugged systems. A first problem that occurs is how to measure the consumption. We have considered an external watt-meter 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 [11] for example. Lot of other work on server power management based on on/off algorithm has been done [12], [4]. Some take into account thermal issues [3], [12]. The main issue in that case is to design an energy-aware scheduling algorithm with the current constraints (divisible task or not [3], synchronization [10], etc). Some algorithms include DVFS (Dynamic Voltage Frequency Scaling) techniques [10], [8] and some not [4]. 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 [7]. 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 [5] or by turning off ports [6]. 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; this work will include the implementation of the method 1. We also plan to make the same experiments with the whole grid traces including the grid reservation constraints (as defined in Section 2.3, on several sites at the same time). We will study the possibility to move reservations from one site to another (according 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] N. Capit, G. D. Costa, Y. Georgiou, G. Huard, C. M. n, G. Mounié, P. Neyron, and O. Richard. A batch scheduler with high level components. In *Cluster computing and Grid 2005 (CCGrid05)*, 2005.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[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.