Energy-Aware Resource Allocation

Damien Borgetto, Georges Da Costa, Jean-Marc Pierson, Amal Sayah

IRIT, Toulouse University

{borgetto,dacosta,pierson,sayah}@irit.fr

***Abstract*—This paper deals with the reduction of energy consumption in large scale systems, especially by taking into account the impact of energy consumption for server consolidation. Decreasing the number of physical hosts used while ensuring a certain level of quality of services is the goal of our approach. We introduce a metric called energetic yield which represents the quality of a task placement on a subset of machines, while taking into account quality of service and energy efficiency aspects. It measures the difference between resources required by a job and what the system allocates ultimately, while trying to save energy. Our work aims at minimizing this difference. We propose placement heuristics that are compared to the optimal solution and to a related system. In this paper, we present a set of experiments showing the relevance of this metric in order to reduce significantly energy consumption.**

I. Introduction and Motivation

For several years there as been an increase of power consumption for computational servers and data centers. Recent studies in the U.S [7], and in Europe [2] showed that power consumption becomes a matter of concern. Operators of such centers are investing substantial amounts in order to supply their infrastructure or to disperse emitted heat .

There are several solutions to reduce the energy bill in both financial and environmental aspects. The first one is based on the energy efficiency of the infrastructure itself, promoting air circulation or alternative cooling of machine rooms. The second one is to deploy less energy-hungry hardware, where the flops/watt ratio is better [5]. Major companies continuously improve their products in this way.

The third solution is algorithmic and tries to optimize processes in order to reduce their energy fingerprint. This solution aggregates numerous approaches, such as network protocols [10], modelling of the consumption [6], task scheduling predicting idle time [4]. Our work aims at providing an energy-efficient job placement, while guaranteeing a certain quality of service, that can lead to turning off hosts.

This approach has been studied before by Stillwell et al. [11] in a different context. While the approach used by Stillwell et al. relies on server consolidation to optimize the utilization of a consistent subset of machines (see section II for details), we go further by integrating in the approach an electric consumption consideration so that the global energy impact is reduced. The idea is here to create a compromise between cost in terms of energy consumption and quality of service. On the one hand the use of only one machine, chosen to have the lowest energy consumption, optimizes the energy consumption but does not guarantee the task's quality of service. On the other hand using the maximum amount of machines fits better the needs of the jobs, but consumes a large amount of energy. Wasting energy can be avoided by grouping jobs on the same host. Although, as shown in this paper, the naïve approach that allows one to turn off unused machines, even after sorting these by energy consumption, can be improved by defining more precisely a metric allowing one to place tasks tending to a compromise.

This paper is organized as follows. Section II provides a set of notations and explains the server consolidation approach used by Stillwell et al. on which this work is based. Section III, presents the theoretical approach to include energy efficiency considerations in the task placement context, by extending and generalizing the previous model. Section IV proposes a few results showing the relevance of our approach and evaluates the energy gains (potentially substantial) regarding the task's quality of service. Finally, section V concludes and opens discussion on future works, for example to extend this model, or to implement it in a distributed system via the use of virtual machines.

II. Server Consolidation

The works of Stillwell et al. in [11] consists of a heuristic addressing the job placement problem in a multi-constrained context of memory and computational capability. The assumption is made that hardware is fully homogeneous. The works of Stillwell et al. introduces a user-centric metric, the *yield*, defining the quality of a placement. In order to make a good placement, one will want to maximize this metric.

$Y_{ij} = \sum_{j=1}^{H}(\frac{\alpha_{ij}}{\alpha_i})$

α_i : fraction of computational capability used by the job i if alone on a physical host.

α_{ij} : fraction of computational capability of host j currently allocated to the job i.

H : Number of hosts

The Y_{ij} notation is used in order to be consistent with the formula in section III, and to show on which host the job is actually allocated.

The yield can be thought as a measure of the job's satisfaction rate, meaning the yield of a job represents the fraction of the maximum achievable computing rate that is achieved. For example, a job that requires 60% of the computational capability of an host, but is allocated at 30% of it, will achieve a yield of $\frac{30}{60} = \frac{1}{2}$. The works of Stillwell et al. are about resource allocation using *multi capacity bin-packing* (MCB) [8] over memory and CPU load. Stillwell et al. uses this metric to aggregate services over a small (hopefully minimal) subset of machines, while guaranteeing a good quality of service by maximizing the minimum yield. Stillwell et al. are using an algorithm that can be described in several steps, by:

1 Fixing a value Y of the yield to achieve.
2 Deducing from Y the CPU requirements of every jobs in order to achieve the yield Y.
3 Sorting the job list in 2 different lists, one with the jobs that have higher CPU requirements than memory requirements, the other with the jobs that have higher memory requirements than CPU requirements.
4 Applying MCB, meaning filling every hosts by choosing a job from the list that goes against the current imbalance. For instance, an host is loaded at 60% CPU and 50% memory according to what have been assigned to it so far. In this case the list to be scanned will be the one with higher memory requirements than CPU requirements. If no fitting job is found, the other list is scanned, and if no fitting job is found either, the algorithm fails. If the host is full, the next is picked to be filled until no host is left (failure) or no job is left (success).
5 Executing the steps 1, 2, 3 and 4 in order to do a binary search on the yield value (Y) and find the best tradeoff.

This heuristic is in polynomial time to the number of jobs and hosts. Therefore this technique allows to group jobs over a small number of physical machines, thus turning off those that aren't used. However, considering energy consumption is only a by-product: when by chance a host is left unused, it can be turned off in order to save energy.

The performance of this heuristic depends heavily on the functions used to sort the job lists. In [11], the most efficient sorting function is MCB8 which consists of sorting in descending order the maximum of the memory and CPU requirements (i.e.: *max(memory,CPU)* in descending order). In the next sections, after having studied all 8 sorting candidates described in [11], we were able to separate two different class of sorting functions. We will choose one of each class for energy studies. In our case, we will be using MCB4 in addition to MCB8 for its energy saving efficiency. The sorting function MCB4 sorts in ascending order the maximum of memory and CPU requirements (i.e.: max(memory,CPU) in ascending order).

III. Taking Energy Into Account

The approach presented by Stillwell et al. focuses on a metric (the *yield*) measuring the quality of a placement. The more a job is allocated close to its requirements, the larger the yield. The assumption is made that there is a full homogeneity of the physical hosts. We chose to keep this assumption because in a cluster context, we often find machines with similar specifications in a same rack. Furthermore, studies [9] showed that physical location of the machines induces differences in their energy consumption.

In order to address the problem of energy efficiency without having too much of a complex model, we make the following simplifying assumptions:

(H1) Machines are heterogeneous in terms of energy, but have the same computational capability.
(H2) A machine constantly consumes C^{min} watts while empty and C^{max} watts fully loaded.
(H3) The extra energy consumed by the execution of a job on a host is a (linear) function f of its computational capability and the host's specifications. It means that $\forall i \in [1..J], \forall j \in [1..H] : \delta C_{ij} = \alpha_{ij} \times f(j)$ where δC_{ij} is the induced energy by the job i on the host j. For now, f is assumed to be $f(j) = (C_j^{max} - C_j^{min})$.
(H4) As in [11], jobs are assumed to be infinite and being able to migrate.

A. Energetic yield

It is necessary to change the metric in order to take into account the energy efficiency with those new assumptions. We use the following formula to define the energetic yield.

For a job $i \in [1..J]$, allocated on a host $j \in [1..H]$, we have :

$$YE_{ij} = \frac{\left[\sum_{j=1}^{H}(\frac{\alpha_{ij}}{\alpha_i})\right]^{1-k}}{\left[\lambda\delta C_{ij}+(1-\lambda)(A_j(1-\sum_{i'=1,i'\neq i}^{J}(\alpha_{i'j})))\right]^{k}} = \frac{(Y_{ij})^{1-k}}{(E_{ij})^k}$$

With $0 \leq \lambda \leq 1$ and $0 \leq k \leq 1$.

This equation is used to conciliate 3 different goals that are:

- aggregating jobs on a reduced number of hosts in order to shut down unused ones;
- placing jobs on energy efficient hosts;
- taking into consideration the quality of service of the jobs.

The Y_{ij} part is the yield and evaluates the quality of the computational resources allocation to the jobs. The E_{ij} part takes into account the energy efficiency problematic. k allows us to make a tradeoff between computational performances and energy savings. The energy part can be separated in two parts : $E_{ij} = \lambda\delta C_{ij} + (1-\lambda) \times A_j(1 - \sum_{i'=1,i'\neq i}^{J}(\alpha_{i'j}))$

- Term δC_{ij} represents the job contribution to the energy consumption of the host j.
- Term $A_j(1 - \sum_{i'=1,i'\neq i}^{J}(\alpha_{i'j}))$ allows to group jobs on attractive hosts. A_j is the attractiveness of the host j and the smaller the value, the more attractive the machine is. In the following $A_j = C_j^{min}$. This part goes smaller (thus leading to a big energetic yield) as the host is both attractive and loaded.

The term A_j is used to weight hosts in order to be inclined to choose energy-efficient machines. λ allows us to weight between the 2 terms above, so that we can make more important one or the other. So if it is not possible to turn off machines, we will set λ to 1. Our approach allows to define the quality of the placement of a job, without caring of placements to come.

B. Properties

Our metric is bound by the three following properties. **Property 1:** The metric's value will be larger with a host that consumes less than another when other host's specifications are the same. It means that if we have 2 machines with the same proposed yield and the same computational load, when a job arrives, the metric will be better for the host in which the increase of energy consumption is the lowest.

$$\forall i \in [1, J]; \forall j, h \in [1, H]; j \neq h; k \neq 0; \lambda \neq 0 :$$
$$(\sum_{i'=1, i' \neq i}^{J}(\alpha_{i'j}) = \sum_{i'=1, i' \neq i}^{J}(\alpha_{i'h})$$
$$\wedge \sum_{j=1}^{H}(\frac{\alpha_{ij}}{\alpha_i}) = \sum_{h=1}^{H}(\frac{\alpha_{ih}}{\alpha_i})$$
$$\wedge A_j = A_h$$
$$\wedge \delta C_{ij} < \delta C_{ih})$$
$$\Rightarrow YE_{ij} > YE_{ih}$$

Property 2: With the same specifications, the metric will be inclined to have a larger value with an allocation of a job on a host that already executes jobs (thus that can't be turned off) than on an empty machine. Thereby, if we have 2 hosts and 2 jobs, we will prefer putting the 2 jobs on only one host rather than allocating one job to each host.

$$\forall i \in [1, J]; \forall j, h \in [1, H]; j \neq h; k \neq 0; \lambda \neq 1 :$$
$$(\sum_{i'=1, i' \neq i}^{J}(\alpha_{i'j}) > \sum_{i'=1, i' \neq i}^{J}(\alpha_{i'h})$$
$$\wedge \sum_{j=1}^{H}(\frac{\alpha_{ij}}{\alpha_i}) = \sum_{h=1}^{H}(\frac{\alpha_{ih}}{\alpha_i})$$
$$\wedge A_j = A_h$$
$$\wedge \delta C_{ij} = \delta C_{ih})$$
$$\Rightarrow YE_{ij} > YE_{ih}$$

Property 3: It is possible to set the system sensitivity to energy. Here, the parameter k varies between 0 and 1, such as if $k = 0$ only the yield will be taken into account and if $k = 1$ only the energy will be taken into account. Thereby, when k increases, the properties above are step by step extended to accept a bigger loss of yield.

C. Optimizations

The algorithm proposed in [11] relies on an algorithm of bin-packing using the yield to properly allocate jobs on hosts. The paper compares several techniques to sort jobs and thus evaluate the impact of the sorting function on the quality of the resulting placement.

One of our contributions is to take into account the energy heterogeneity of the machines. Indeed, the machines sort that the bin-packing takes into account has an impact over the algorithm's performance. In this paper, we will compare the following sorting functions for the hosts :

- TH1 : C^{min}, ascending order
- TH2 : C^{max}, ascending order
- TH3 : $C^{min} + C^{max}$, ascending order
- TH4 : $C^{max} - C^{min}$, ascending order

IV. Experiments

In order to evaluate our approach, we ran various simulations, watching the evolution of the yield deterioration with the reduction of the system energy consumption, while increasing the parameter k. The system energy is the sum of all energies consumed by each host with at least one job running on it.

A. Methodology

We have generated a set of problems based on the problem generation in [11], a problem being a set of hosts and a set of jobs to be allocated, using the following methodology. CPU needs of the jobs are generated using a normal distribution with a mean of 0.5, and a coefficient of variance of 0.25 and 0.75. Memory needs are also generated using a normal distribution, with a coefficient of variance of 0.25 and 0.75, but with a mean of $H * (1 - slack)/J$. H being the number of hosts, J the number of jobs, and *slack* representing the percentage of free memory on the system once the jobs are generated.

Then the problems are generated with, for the small problems, 4 hosts, and 6, 8, 10 and 12 jobs. Which makes a total of 144 different problem specifications. For each specification, 10 problems are generated, which makes a total of 1440 simulations for the small problems. We are doing the same for the medium sized problems, multiplying the number of hosts and jobs by 6, thus having 24 hosts with 36, 48, 60 and 72 jobs.

In order to highlight the influence of our metric on the system energy consumption, we search for a near optimal solution with the different MCB algorithms described in [11], while, for each generated problems, varying k (the weight of the energy part of our metric) from 0 to 1 by steps of 0.01. Moreover, these tests were made with values of λ varying from 0 to 1 by steps of 0.1.

Finally, in order to define the energy consumption of the hosts, we assigned to each host a C^{min} value between 100 and 200 watts, and a value of C^{max} between 200 and 400 watts, both generated using a uniform distribution. For further details about the existing, it will be necessary to generate those values using a more realistic model.

Each of those 1440 problems will be the input of a simulator that we specifically developed for those experiments. This simulator evaluates accurately the metric using the problem in input.

B. Results

In order to analyze the results of our simulations, we focused on the energy consumption of the system and on the average yield. Indeed, the system energy will directly be used to evaluate the impact of our method over the energy reduction, whereas

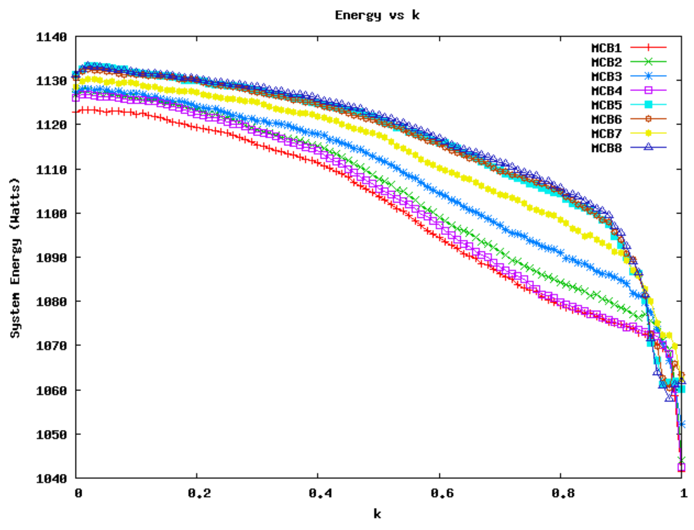

Figure 1. Variation of the system energy versus k, small problems

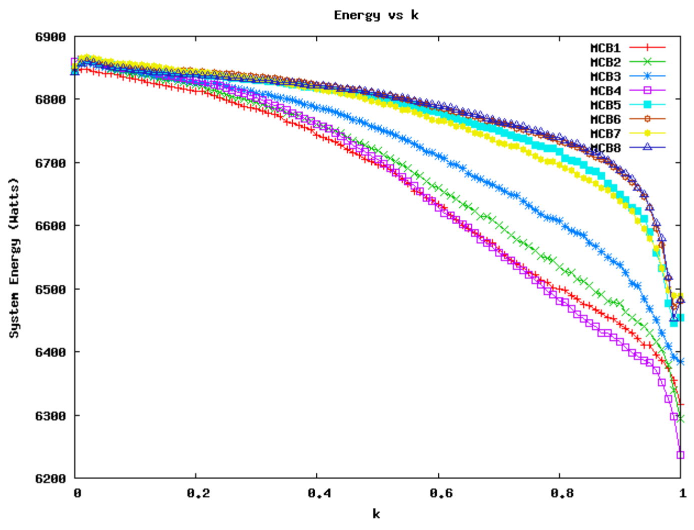

Figure 2. Variation of the system energy versus k, medium problems

analyzing the average yield will allow us to see how much the performances are affected.

First we present the evolution of the system energy in order to observe how it behaves when the parameter k increases. The figure 1, was generated using small problems specification, with the 8 MCB algorithms, without sorting the hosts (thereby the first host is filled regardless of its characteristics). As we can see, the more we get close to $k = 1$, the more the energy decreases. This is mainly because of the fact that if the weight is on the energy component of the metric, only the memory will be limiting the job placement, thus having the best energy-efficient placement and grouping of the jobs on the hosts for a value of k close to 1.

We can also see that there is still a difference, in energy terms, between the different MCBs. In fact, the MCB8 is the one that achieves the best performance being the closest to the optimal, so we could think that it is the best in term of energy. However, as we can see in figure 1, the opposite occurs, even if the difference is low. This is because the MCB4 algorithm for instance, which achieves one of the lowest energies, fails more often than the MCB8. We have to remind ourselves that MCB4 sorts the jobs by max(CPU, memory) in ascending order, whereas MCB8 is in descending order. Thereby, when MCB4 doesn't fail, the resulting placement will be that after the average yield increasing phase, the global load of the system will be smaller than using MCB8, thus having not only a smaller system energy, but also a lower average yield.

Figure 2 gives an overview of the energy variation linked to the increase of the problem size. Those are the medium problems. Unlike figure 1, we can see a faster decrease of the energy when increasing k with MCB8 for instance. We can see that the gap between the best and the worst MCB increases as the problem becomes larger. We can think that the placement using MCB4 for example, which is higher than the placement using MCB8 in terms of energy, will give better results with large problems. However, the average yield is also higher for MCB8 than for MCB4.

As in [11], once the bin-packing is done, we can switch off machines that have no jobs to execute, once the jobs are grouped on other machines. We can go further in energy reduction, going beyond the naïve approach in order to globally take into account the energy consumption. Thus saving more energy by placing jobs on hosts regarding their specifications, while using those specifications to make a good placement. In fact, in [11], a homogeneous context is considered. Obviously, it isn't the case in a realistic case, because even if the machines are identical, they won't necessarily consume the same power based on where they are physically placed [9]. Figure 3 shows the significance of taking into account the heterogeneity of the host's configuration. We can see that with some simple host sorting functions, on a small problem such as 4 hosts, we are already able to save a significant of energy. Without host sorting, we achieve a lower performance because the host to fill in first is the first on the host list. Thereby, we can imagine that on a number of hosts way above the number of jobs, which is a common use case in grids and clusters when not in burst periods, we can easily achieve a substantial increase of the energy efficiency of the system.

Figure 4 shows the energy loss percentage and the average yield loss percentage, while varying the parameter k. We can see, for example, that with the MCB8 algorithm and with $k = 0.6$, a loss of yield of 2.5% enables about 1.5% of energy saving. We can imagine for example in the case of jobs that are not too much time constrained, this yield loss is acceptable. Moreover,

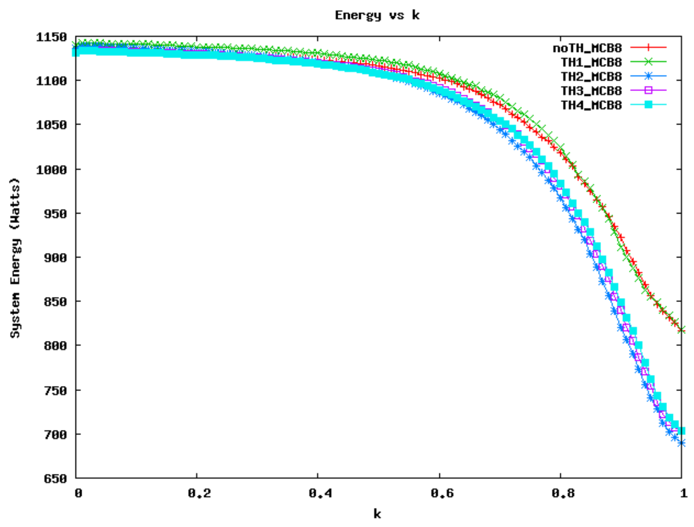

Figure 3. Different sorting functions for the hosts, small problems

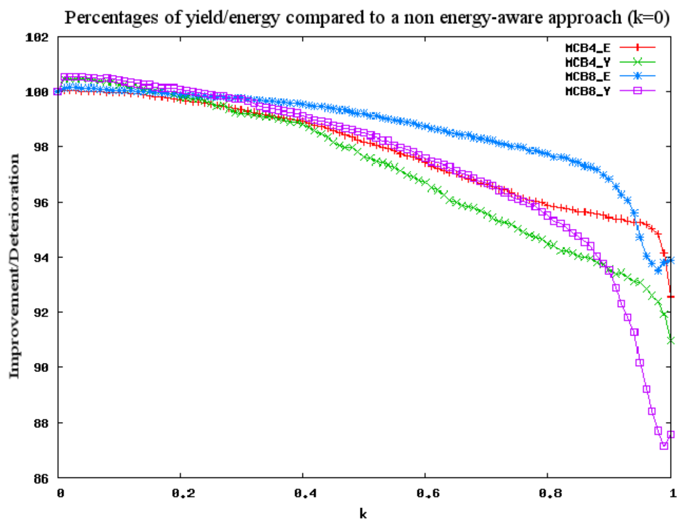

Figure 4. Comparison between energy and yield losses by varying k, small problems

a loss of 2.5% in average yield isn't necessarily lead to an increase of 2.5% of the computation time, or an increase of 2.5% in energy consumption, because the major part of the energy consumption is due to C^{min}. Of course, this is only for the small problems, thus only for 4 hosts, which means that the leeway is relatively restricted. We have to add that the percentages were generated using the respective values of the algorithms. In other words for example, losing 2% of average yield for the MCB8 is larger in value than 2% for the MCB4 algorithm, because the average yield of the MCB8 is higher than the average yield of the MCB4. For a value of $k = 0.99$, which means where the difference between loss of average yield and energy efficiency profit is maximal, there is about 6% of energy gain, for a loss of yield of less than 13%. This case corresponds to the situation where the yield efficiency is the worst, but it also corresponds to the case where the energy consumption is the lowest.

Figures 5 and 6 show the evolution of the average yield and energy when varying both parameters k and λ. As expected, the energy is the lowest around values $k = 1$ and $\lambda = 1$, and the average yield is also the lowest around those values. Those values are corresponding to the case where we take into account in the energy component only the placement of jobs on good hosts. Besides, it should be noted that the performances achieved with $\lambda = 0$ probably aren't what they seems to be, because there is too little room to regroup jobs on hosts. For such values of the parameters, we are saving around 40% of energy compared to the case where we do not take into account energy ($k = 0$). We also see deterioration of the average yield of around 45% . One should note though that simulations ran to achieve those results were made with the sorting function TH3 (see section III-C) for hosts, and sorting function MCB8 for jobs.

V. Conclusion and Future Works

In this paper, we presented a theoretical study of jobs placement over a set of homogeneous machines, while taking into account parameters *reduction of the energy consumption* and job satisfaction in terms of obtaining wanted resources on chosen sites. The impact of gathering jobs on a reduced number of machines were also handled, which allows to consider if machines can be switched off or not. The first simulations that we ran in a *small problems* context have :

- validated the interest considering the energy consumption parameter as such and not only as a side-effect induced by a placement tending to load as much as possible a smaller number of machines.
- showed the impact of the energy saving parameter over job satisfaction: the more the energy consumption is considered, the more the yield deteriorates. Thereby an extreme placement achieves to save 40% of energy, while loosing 45% of the average yield.
- showed the importance of the sorting functions, therefore the choice of the hosting machine the most energetically beneficial in order to increase energy savings.
- confirmed that the context of *small problems* allows too few room to achieve both a significant energy saving and a low average yield deterioration.

We hope that from the different cases studied, we will be able to characterize placement strategies suitable to configurations in order to obtain the best energy saving achievable. The context of this study (homogeneous machines, infinite jobs that we manage only the initial placement, ...) has to be extended in order to be closer to real situations, and to consider aspects such as :

- the heterogeneity of the considered machines.

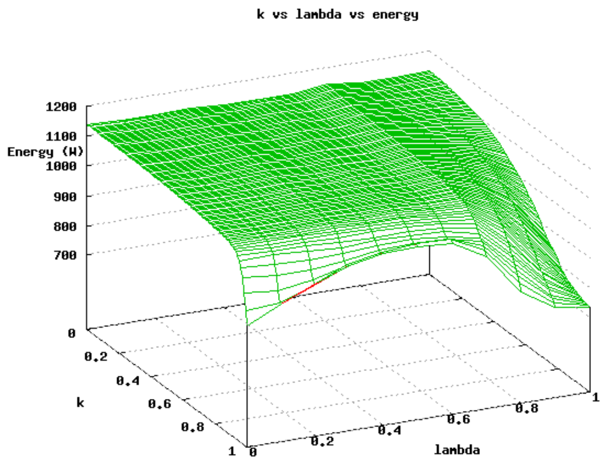

Figure 5. Evolution of the energy while varying k and λ, small problems

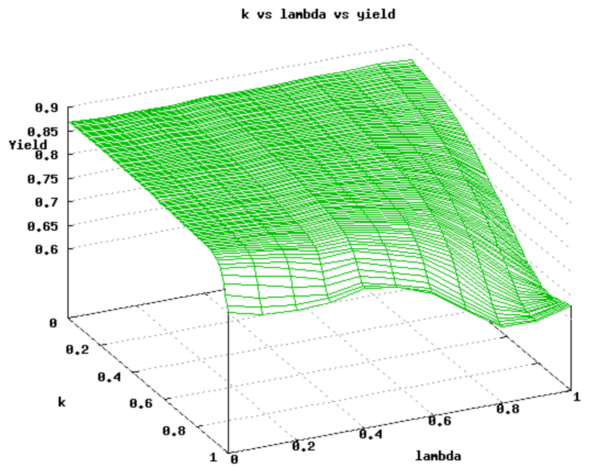

Figure 6. Evolution of the average yield while varying k and λ, small problems

- the dynamic creation/extinction of jobs and the dynamic review of the jobs to the hosts, with the corollary migration of activities[3].
- the consideration of more constrained parameters of quality of service than obtaining required resources : response time, deadlines, planning, starvation, …
- the overhead generated by the energy management.

At the same time, we are seeking to verify theoretical results obtained with these simulations by the implementation of an autonomous system managing the activity placement (virtual machines [1]) on a given hardware architecture and aiming to reduce energy consumption while using the triptych *Sensors* to observe (resources, activities, …) - *Decision* (placement, switching on/off hardware, …) - *Actuators* to implement the decisions.

REFERENCES

[1] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.

[2] P. Bertoldi and B. Atanasiu. Electricity consumption and efficiency trends in the enlarged european union. Technical Report EUR 22753EN, Institute for Environment and Sustainability, 2007.

[3] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

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

[5] W. Feng, M. Warren, and E. Weigle. Honey, i shrunk the beowulf! In *International Conference on Parallel Processing, 2002. Proceedings.*, pages 141–148, 2002.

[6] Helmut Hlavacs, Georges Da Costa, and Jean-Marc Pierson. Energy consumption of residential and professional switches. Rapport de recherche IRIT//RR-2009-7-FR, IRIT, Université Paul Sabatier, Toulouse, mars 2009.

[7] J. G. Koomey. Estimating total power consumption by servers in the u.s. and the world. Technical report, Stanford University, 2007.

[8] William Leinberger, George Karypis, and Vipin Kumar. Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints. In *ICPP '99: Proceedings of the 1999 International Conference on Parallel Processing*, page 404, Washington, DC, USA, 1999. IEEE Computer Society.

[9] A.-C. Orgerie, L. Lefevre, and J.-P. Gelas. Chasing gaps between bursts: Towards energy efficient large scale experimental grids. In *Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2008. PDCAT 2008.*, pages 381–389, Dec. 2008.

[10] Barry Rountree, David K. Lowenthal, Shelby Funk, Vincent W. Freeh, Bronis R. de Supinski, and Martin Schulz. Bounding energy consumption in large-scale mpi programs. In Becky Verastegui, editor, *Proceedings of the ACM/IEEE Conference on High Performance Networking and Computing, SC 2007, November 10-16, 2007, Reno, Nevada, USA*, page 49. ACM Press, 2007.

[11] M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova. Resource allocation using virtual clusters. In *Proceedings of the 9th IEEESymposium on Cluster Computing and the Grid (CCGrid'09)*, may 2009.