Energy-Aware Server Provisioning at the Middleware level through Green Scheduling

Daniel Balouek-Thomert
daniel.balouek-thomert@ens-lyon.fr

Eddy Caron
eddy.caron@ens-lyon.fr

Laurent Lefevre
laurent.lefevre@inria.fr

LIP Laboratory.
UMR CNRS - ENS de Lyon - INRIA - UCB Lyon 5668
University of Lyon, France

***Abstract*—Going green is neither a reflex nor mandatory in computing for most of the users. In a context of heterogeneous resources, performance is the traditional criteria when it comes to provisioning. But, nowadays, taking into account the power consumption of distributed computing architectures has become mandatory. This paper addresses energy-efficiency challenges in mapping of jobs in distributed systems. It also proposes smart energy efficient resource provisioning by enabling the middleware to manage energy-related events with user-defined rules. Through the use of the DIET middleware, which enables custom-based workload placement, we aim at putting in light tradeoffs between performance and electric consumption. Our experimental results evaluate the performance and power consumption of three scheduling policies, with significant gain in terms of energy-efficiency while observing minimal impact on the global performance. We also evaluate reactivity improvements in context-aware resource provisioning.**

Keywords: Distributed computing, energy-efficiency, workload placement, resource provisioning

I. Introduction

In recent years, distributed computing have proven to be mandatory in IT. The amount of services and their diversity are constantly increasing through different usages. Computing has considerably evolved, going from small isolated nodes to large cluster-like architectures driven by efficiency and scalability. An instance of these architectures is known under the name of *Clouds*, which offer virtualized resources as a service over the Internet. Those innovations were led by a race to performance, symbolized by the TOP500 list of supercomputers which annually ranks the world fastest computers. Within 20 years, the top machine improved performance by a factor 100 (measured in Linpack Performance)[1]. But nowadays, the target of reaching exascale computing is limited by an essential factor: the power consumption [1]. ICT technologies took an important portion of the world electric consumption, and the trend is likely to continue and to further increase for years to come [2] [1]. The Green500 list raises awareness of power and energy consumption in supercomputers by reporting the energy-efficiency of large-scale HPC facilities[2].

There exist numerous approaches for reducing the electric consumption of computing resources. Beside hardware optimizations, software techniques includes Dynamic Power Management (DPM) and Dynamic Voltage and Frequency Scaling (DVFS). As its name suggests, DVFS enables the scaling of CPU frequency to reduce its power needs, leading to a lowering of the overall energy consumption of the system, while a server system can be switched off using DPM. A few other techniques emphasize with the dimensioning and usage of resources. In virtualized environments, consolidation [3] and proportional provisioning [4] are often used to dynamically adjust the sharing of physical servers. Those techniques are to be triggered according to the operational requirements of the system, allowing tradeoffs between performance, availability and durability of computing resources.

This work aims at considering the green criteria in the frame of server provisioning and workload management. We suggest a metric allowing infrastructure administrators to put a preference between performance and energy savings without prior knowledge or assumptions on the hardware. As the scheduling process requires access to target platforms and dynamic information about the resources, we integrate our proposition within the DIET [5] project that ensures the ability to access heterogeneous nodes in the middleware layers. Results shows energy and performance improvements with minimal impact on the applications and systems. The evaluation is performed by the means of simulations and real-life experiments on the GRID'5000 testbed.

This paper is organized as follows. Section 2 describes related work. Section 3 presents the DIET middleware and its features. In Section 4, we discuss the implementation of our use case by computing independent tasks over heterogeneous nodes. Section 5 details our context-aware resource provisioning proposition, by taking into account periods of time with specific properties. These algorithms are validated with experiments in Section 6. Section 7 discusses about out-of-scope aspects of the current article. Section 8 concludes the paper and presents some perspectives.

II. Related Work

Despite an increasing demand in Cloud computing, data centers are rarely fully utilized [6], mostly as a consequence of overprovisioning. Highly fluctuating resource demands with low utilization on average are a limitation to energy efficiency, particularly with light loads on nodes [7]. Energy saving techniques consists of two main approaches. One can slow down

[1]The TOP500 ranking. http://www.top500.org
[2]The Green500 ranking. http://www.green500.org

server components [8] [9]. However, Le Sueur *et al.* found out that those methods are becoming less attractive on modern hardware [10]. Actual software proceeds by transitioning entire servers into a sleep state [3] [4]. Those improvements are well suited for Cloud Computing, which virtualized infrastructures to process any type of workload.

As Grid and Utility computing aims "to enable resource sharing and coordinated problem solving in dynamic, multi-institutional virtual organizations" [11], it is often harder to transition servers to shutdown states due to availability reasons and the specific nature of services they could be offering. In this context, many works uses load concentration [12] and consolidation [13] when the nature of the tasks (or virtual machines) allows those mechanisms.

Many works, such as [14] [15], assume that nodes from a homogeneous cluster have the same power consumption. In practice, due to their different uses, nodes from the same cluster can present different ranges of performance and energy consumption. It has been shown in [16] that nodes from a same cluster may have a different power consumption due to fluctuations caused by the external environment, such as external temperature and position of the node in the rack. Additionally, Diouri *et al.* [17] observed that power heterogeneity could be due to intensive usage of specific hardware components and leakage power that vary over time.

Most Grids use a batch-scheduler compute model, in which a Local Resource Manager manages the compute resources for a Grid site, and users submit batch jobs to request some resources for some time [18]. Cloud aggregators such as RightScale[3] provide automatic cloud management and load balancing but are application-specific. At the application level, distributed OSes such as fos [19] propose programming models that allows OS systems services to scale with demand. Only a few managers among those systems offers Green capabilities [20].

The contribution of this paper is to enable energy-aware scheduling at the middleware level by providing an abstract software layer that can be automated and controlled centrally, in a reasonable way. The previous studies suggest us to consider the dynamic context of execution without prior assumption of the underlying hardware. Our proposition relies on the DIET middleware by taking advantage of its ability to interact with heterogeneous resources and scalable properties.

III. The DIET Middleware

A. DIET *overview*

The DIET open-source project is focused on the development of a scalable middleware with initial efforts relying on distributing the scheduling problem across a hierarchy of agents. It is implemented in CORBA and benefits from the many standardized, stable services provided by freely-available, high-performance CORBA implementations.

The DIET toolkit is composed of several elements, illustrated in Fig. 1. The first element is a **Client**, an application that uses the DIET infrastructure to remotely solve problems. The second element is the **SeD** (**Server Daemon**) which acts as the service provider, exposing functionality through a standardized computational service interface; a single SeD can offer any number of computational services. The third element of the DIET architecture is the **agent**. Deployed alone or in a hierarchy, the agent facilitates the service location and invocation interactions between clients and SeDs. Collectively, a hierarchy of agents provides higher-level and scalable services such as scheduling and data management. The head of a hierarchy of agents is called a **Master Agent (MA)** while the others are **Local Agents (LA)**.

[3]Rightscale home page. http://www.rightscale.com/

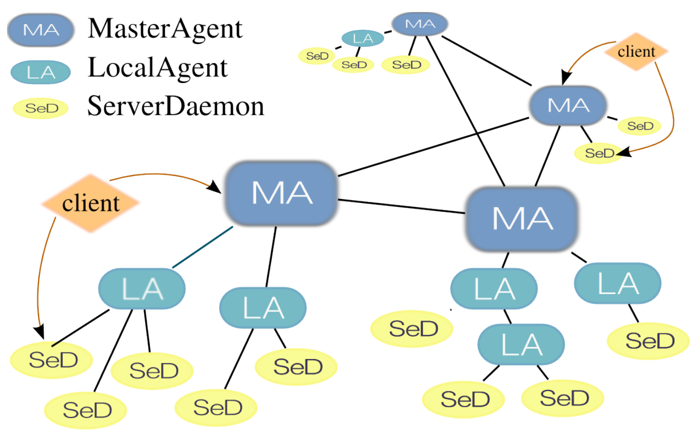

Fig. 1. A DIET hierarchy.

B. DIET *Plug-in Scheduler*

By default, when a user request arrives at a SeD, an estimation vector is created via a default estimation function; typically, this function populates the vector with standard values which are identified by system-defined tags. Table I lists the tags that may be generated by a standard installation.

Consequently, applications targeted for the DIET platform are able to exert a degree of control over the scheduling subsystem via plug-in schedulers. For example, a SeD that provides a service to query particular databases may need to include information about which databases are currently resident in its disk cache, so that an appropriate server may be identified for each client request. If the application developer includes a custom **performance estimation function** in the implementation of the SeD, the DIET framework will associate the estimation function with the registered service.

Each time a user request is received by a SeD associated with such an estimation function, that function, instead of the default estimation procedure, is called to generate the performance estimation values. These features are invoked after a user has submitted a service request to the MA, which broadcasts the request to its agent hierarchy.

As the physical infrastructures that are to be used vary greatly in terms of demands, we used this DIET plug-in scheduler facility at the server level to express contextual information about performance and power consumption, that will be taken into account when servers are provisioned. Such vectors are then the basis on which the suitability of different SeDs regarding to energy efficiency is evaluated. The following section describes a metric based on customized estimation

Information tag starts with EST_	multi-value	Explanation
TCOMP		the predicted time to solve a problem
TIMESINCELASTSOLVE		time since last solve has been made (sec)
FREECPU		amount of free CPU between 0 and 1
LOADAVG		CPU load average
FREEMEM		amount of free memory (Mb)
NBCPU		number of available processors
CPUSPEED	x	frequency of CPUs (MHz)
TOTALMEM		total memory size (Mb)
BOGOMIPS	x	the BogoMips
CACHECPU	x	cache size CPUs (Kb)
NETWORKBANDWIDTH		network bandwidth (Mb/sec)
NETWORKLATENCY		network latency (sec)
TOTALSIZEDISK		size of the partition (Mb)
FREESIZEDISK		amount of free place on partition (Mb)
DISKACCESREAD		average time to read from disk (Mb/sec)
DISKACCESWRITE		average time to write to disk (Mb/sec)
ALLINFOS	x	[empty] fill all possible fields

TABLE I. EXPLANATION OF THE STANDARD ESTIMATION TAGS.

tags in order to balance tradeoffs between performance and energy.

IV. WORKLOAD PLACEMENT

The resources where tasks are computed presents a significant impact on the overall energy consumption of the system. By prioritizing efficient resources, administrators can achieve a finer control on the different performance regimes of the infrastructure. We propose a metric for the sorting of available computing nodes according to a hybrid of their electric consumption and a secondary parameter, the performance of the node.

As a result, the GREENPERF metric seeks to optimize power utilization while maintaining the throughput requirements imposed by the application. We thus evaluate the tradeoffs of green scheduling for reducing the energy consumption while matching performance objectives.

We consider the problem with independent tasks on heterogeneous computing nodes with energy monitoring capabilities and assume that tasks are not assigned priorities. Using the ratio

$$\frac{\text{Power Consumption}}{\text{Performance}}$$

related to each computing server, a ranking of available nodes is set.

Figure 2 shows a simple example with 5 servers and 7 tasks, where the most energy-efficient servers are prioritized (S0 being the ”better” server for the GreenPerf metric).

The computation of this metric requires to obtain data related to the target servers from the physical infrastructure, namely the energy consumption and performance.

Regarding to the energy consumption metric, two approaches are possible in order to obtain the data for a type of service/task. A static way would imply to perform some benchmarking by computing a job on all nodes and measure the instantaneous electric consumption corresponding to the completion time on each node. That method is not significant for long periods of time cause the power consumption of the machine may vary, depending on the actual load or the external

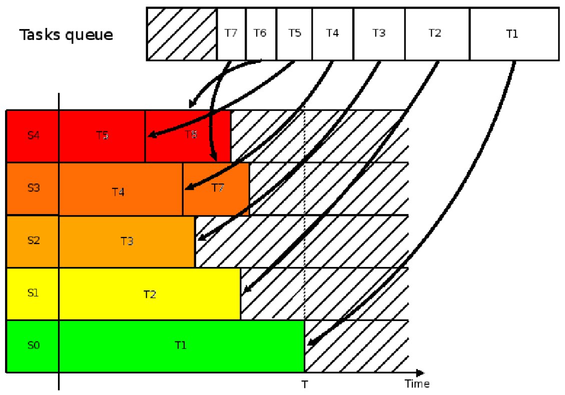

Fig. 2. Example of task placement using the GreenPerf metric.

conditions such as the physical location of the server in the rack.

In this context, we favor a more dynamic approach. The energy consumption metric associated to a server corresponds to the number of requests handled, divided by the measured power consumption since the first request. It results on a value based on the recent activity of the resource rather than an initial benchmark.

We integrate those metrics within DIET scheduling facility for each SED to fill its estimation vector using new estimation tags, presented in Table II. The mechanics of this ranking process comprises an **aggregation method**, which is simply the logical process of sorting the servers responses according to the GreenPerf metric.

Using those, the servers are advised to forward an estimation vector to the scheduler, that will sort them and select the appropriate one to perform the client request according to the GreenPerf metric. Every time a client is submitting a request for a specific application, each server retrieves its energy consumption and the total number of requests.

The steps of the scheduling process are explained below:

1) *Submission of a problem*
A client issues a request describing a problem. If none of the servers are able to solve it, an error is returned

Information tag starts with EST_	multi-value	Explanation
ENERGYAVG		average energy consumption on solved requests (J)
FLOPS		Node performance (Gflops)

TABLE II. EXPLANATION OF THE CUSTOMIZED ESTIMATION TAGS.

to the client.

2) *Propagation of the request*
The Master Agent communicates with all the agents in order to forward the request to the SEDs.
3) *Collect of estimation values*
Each server computes and gathers its custom metrics, particularly performance and energy consumption. A reply containing this estimation vector is sent back to the Master Agent.
4) *Sorting of candidates*
Once the Master Agent retrieves all the estimation vectors, it proceeds to a sort according to a specific criterion. The first SED is then elected and notified.
5) *Solving the problem*
The client can contact the elected SED, which will start the computation of the problem.

Other criteria exist in the literature, involving the consideration of idle consumption [17] or the use rate [21] of the physical nodes. In Section VI, an evaluation of the GreenPerf metric is performed by the means of simulation, showing the tradeoffs between power and performance in high and low heterogeneity environments.

V. CONTEXT-AWARE RESOURCE PROVISIONING

In the previous Section, we described a custom-based metric that considers the energy-efficiency of resources when ranking available servers to assign computing tasks.

Our aim is to provide a simple framework for resource management which provides control for informed and automated provisioning. Its implementation lies at the scheduler level by putting at disposal of the developer (administrator or end-user) an abstract layer to implement aggregation and ranking methods based on the contextual information such as the infrastructure status, the will of users and the energy-related external events occurring over time.

It establishes relationships between the physical infrastructure and the logical behaviour.

We consider that the sizing of computing nodes resources must take into account the will of the user (who expresses the requests) and the provider (who manages the physical machines). In this context, we suggest to offer the ability for the user and provider to express levels of preference in the allocation of resources in terms of performance and energy efficiency.

A. Provider Preference

Providers indicate their preference about the energy efficiency of the infrastructure according to specific periods of time or events. It enables the management of budget limits and can be used to take advantage of the fluctuations of energy price or avoid intensive use of specific machines.

We modelize the provider preference in accordance with:

- Resource usage forecast: Using historical usage data to identify time patterns, and ensure the responsiveness of the platform during peak periods.
- Electricity costs: Minimizing the cost of computation by using low cost periods

We define the provider preference as a weighted average between the resource usage and the electricity cost.

Let c the cost of electricity defined as a ratio between the cost for a given period and the theoretical maximum cost.

Let u the utilization of the resources defined as a ratio between the power consumption for a given period and the total power.

Supposing $c, u \in [0, 1]$ for each time period:

$$Preference_{provider}(u, c) \rightarrow \frac{\alpha(1-c) + \beta u}{\gamma}$$

We obtained a $Preference_{provider}(u, c) \in [0, 1]$. By adjusting the multiplying factors α, β, γ, one can favor a specific metric over the function. The higher the value of $Preference_{provider}(u, c)$, the larger the number of available servers for the time period.

B. User Preference

On the user side, the $Preference_{user}$ indicates the level of consideration in terms of energy efficiency. When submitting a request, the user sets $Preference_{user} \in [-1, 1]$.

$$Preference_{user} \begin{cases} -1 & \Leftrightarrow & \text{maximize performance} \\ 0 & \Leftrightarrow & \text{no preference} \\ 1 & \Leftrightarrow & \text{maximize energy efficiency} \end{cases}$$

As an example, with $Preference_{user} = 1$, the user favors the maximization of energy efficiency on the target platform. In the case of all users indicating that setting, it would result in waiting queues on the most energy-efficient nodes. In practice, it is better to limit that value to $[-0.9, 0.9]$ The user preference is then weighted by the administrator one.

$$(P_{provider}, P_{user}) \Leftrightarrow P_{provider}(P_{user} - 1)$$

C. Event Management

Another aspect of our proposition is to adapt the provisioning of resources with consideration of energy-related events, such as fluctuations of energy cost or heat peaks.

We propose the use of a provisioning planning as a mechanism to facilitate the monitoring of utilization metrics over time. That allows the scheduler to perform operations in an autonomic fashion before executing placement and/or provisioning decisions with consideration of thresholds.

This information can be obtained by predicting future usage from historical data, checking schedules provided by the energy provider or using the infrastructure monitoring system.

We consider that administrators sets thresholds to limit the number of active nodes in case of out-of-range values according to the following variables:

- Cost of energy for a given time period
- Temperature conditions

Using these variables, the scheduler is defining a forecast of resource usage to adapt the number of nodes available for computation. We use the term of *candidate nodes* to designate those resources.

The scheduling process is adapted (cf. Section IV):

1) The Master Agent receives a request describing a task and a value for $Preference_{user}$
2) The request is propagated and estimation vectors are retrieved (cf. Section 4)
3) The scheduler checks the temperature and energy costs thresholds defined by the administrator and adjust the number of candidate nodes regarding to the $Preference_{provider}(u, c)$
4) The list of candidates is sorted according to the scheduling criteria
5) The number of candidates is returned to the client

D. Server selection

We proposed in Section IV a dynamic scheduling process to select the appropriate node based on a client request, involving retrievals of metrics at each request.

We consider the ability to estimate the duration of pending tasks. The following information is assumed to be known for each server at any time:

f	number of operations per floating point (FLOPS) of the server
c	average consumption when fully loaded (W)
bc	consumption during the boot process (W)
bt	boot time (seconds)
w	estimation of tasks waiting queue (seconds)
P	$Preference_{user}$
n	number of operations per floating point (FLOPS) to perform a specific task

The knowledge of those variables allows the scheduler to consider inactive nodes in the decision process and thus evaluate the costs of turning on servers if necessary. The execution time of a task is defined by the number of operations and the performance of a server ($\frac{n}{f}$). The total computation time and the energy consumption of a task both depends of the state of the assigned server at the moment of the scheduling decision.

The computation time (1) and energy consumption (2) of a task on a specific server can be divided up to two cases, depending on the state of the server.

$$computation\ time = \begin{cases} w + \frac{n}{f} & \text{active server} \\ bt + \frac{n}{f} & \text{inactive server} \end{cases} \tag{1}$$

$$energy\ consumption = \begin{cases} c * \frac{n}{f} & \text{active server} \\ bt * bc + \frac{n}{f} & \text{inactive server} \end{cases} \tag{2}$$

By using those two functions in the scheduler, we can assign a score S to each server and establish a sorting (3).

$$S : P \rightarrow computation\ time^{\frac{2}{P+1}-1} * energy\ consumption \tag{3}$$

This score is coherent with our expectations, regarding to the previous definitions of $Preference_{user}$ and $Preference_{provider}$ (4):

$$S : \begin{cases} P \rightarrow -1 \Rightarrow S \sim computation\ time \\ P \rightarrow 0 \Rightarrow \sim computation\ time * energy\ consumption \\ P \rightarrow 1 \Rightarrow \sim energy\ consumption \end{cases} \tag{4}$$

In order to form a set of candidate nodes, we aim to minimize the total power consumed by the active servers of the infrastructure by maximizing the number of servers among the most energy efficient ones. We do not consider any bound for the makespan and we assume that servers have continuous speeds in terms of performance.

We use a greedy algorithm for selecting candidates servers with the objective of maximizing the power consumption among servers (Algorithm 1).

Be T the list of servers sorted according to the GreenPerf metric, RES the result set of servers, P_{Total} the accumulated power of each server and $P_{required}$ the required power among the candidate nodes.

```
$P_{Total} \leftarrow 0$
for $server \in T$ do
    $P_{Total} += server.get\_power()$
end
$P_{required} \leftarrow Preference_{provider} * P_{Total}$
$P \leftarrow 0$
$RES \leftarrow []$
while $P < P_{required}$ do
    $P += T.get\_first\_element().get\_power()$
    $RES.add(T.get\_first\_element())$
    $T.remove\_first\_element()$
end
return $RES$
```

Algorithm 1: Selection of candidate servers within a power consumption cap

This section mentioned the mechanisms and the variables involved through the context-aware resource provisioning. In order to use those, the developer has to provide platform-specific functions that retrieves the value of the described variables. Those can usually be obtained by accessing directly the physical nodes or using third-party monitoring tools.

In Section VI, we illustrate those mechanisms by the mean of two scenarios on the GRID'5000 testbed. The first scenario focuses on workload placement, while the second one puts in light the reactivity of the scheduler.

VI. EXPERIMENTAL RESULTS

A. *The* GRID'5000 *Testbed*

The GRID'5000 testbed has been designed to support experiment-driven research in parallel and distributed systems. By leveraging the GRID'5000 platform, users may perform experiments on all layers of the software stack of distributed infrastructures, including high-performance computing, grids, peer-to-peer, and cloud computing architectures. Located in France, GRID'5000 is composed of 29 heterogeneous clusters, 1,100 nodes, and 7,400 CPU cores with various generations of technology among 10 physical sites interconnected by a dedicated 10 Gbps backbone network.

The power measurement is performed with an energy-sensing infrastructure of external wattmeters from the SME Omegawatt. This energy-sensing infrastructure, which was also used in [22], enables to get at each second the mean power consumption in Watts computed over up to 6000 power samples for each monitored node [23]. We consider that this mean power consumption displayed each second is a very accurate instantaneous power measurement. Logs provided by the energy-sensing infrastructure are displayed in live and stored into a database, in order to enable users to get the power and the energy consumption of one or more nodes between a start date and an end date.

The performance metric is based on a measure of the node performance using all its CPU cores. It produces a value in flops, indicating the number of floating points operations per second. Those benchmarks are based on measurements using ATLAS[4], HPL[5] and Open MPI[6].

B. *Workload placement*

The first evaluation aims to compare distributions of tasks among nodes on GRID'5000, depending on three different policies named PERFORMANCE, POWER and RANDOM.

PERFORMANCE and POWER correspond, respectively, to the setting of a priority on the fastest nodes and on the most energy-efficient nodes. They represent the bounds of the GreenPerf metric. The RANDOM policy corresponds to a shuffle of servers. We do not need to use an estimation tag for random provisioning: the shuffle of servers is performed at the Master Agent level using random numbers.

To be able to construct such scenario, we consider a client submitting a set of tasks. A single task to solve is a CPU-bound problem which consists in $1e8$ successive additions, enabling the distinction of nodes in terms of performance. As each task is using a single core, a server cannot compute a number of tasks superior to its number of cores at a time.

[4]Automatically Tuned Linear Algebra Software. http://sourceforge.net/projects/math-atlas/

[5]Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. http://www.netlib.org/benchmark/hpl/

[6]High Performance Message Passing Library. http://www.open-mpi.org/

We deploy the DIET middleware on 14 physical nodes from the GRID'5000 testbed that offers energy measurement capabilities:

- 12 dedicated nodes for the Server Daemons (SED)
- One dedicated node for the Master Agent
- One dedicated node for the Client

The machines are picked among 3 different clusters. The specifications are presented in Table III. The nodes are connected to a switch with a bandwidth of 1Gbit/s using Debian wheezy as their Operating System.

Cluster	Nodes	CPU	Memory	Role
Orion	4	2x6cores @2.30Ghz	32GB	SED
Sagittaire	4	2x1cores @2.40Ghz	2GB	SED
Taurus	4	2x6cores @2.30Ghz	32GB	SED
Sagittaire	1	2x1cores @2.40Ghz	2GB	MA
Sagittaire	1	2x1cores @2.40Ghz	2GB	Client

TABLE III. EXPERIMENTAL INFRASTRUCTURE.

The total amount of client requests is based on the total number of cores available. Let n be the total number of cores present among the DIET infrastructure. The total number of requests is equivalent to $10n$.

The temporal distribution of jobs contains two phases:

- A burst phase, when the client submit simultaneously n requests
- A continuous phase, when the client submit requests a rate of 2 requests per second.

Considering that the scheduler does not have any particular information on the nodes nor establishes assumption about the hardware, the dynamic information is gathered as tasks are computed by the servers.

Figures 3,4 and 5 show the results of this experiment. The x-axis presents the different nodes available to solve the problem. The y-axis shows the number of tasks executed by the node.

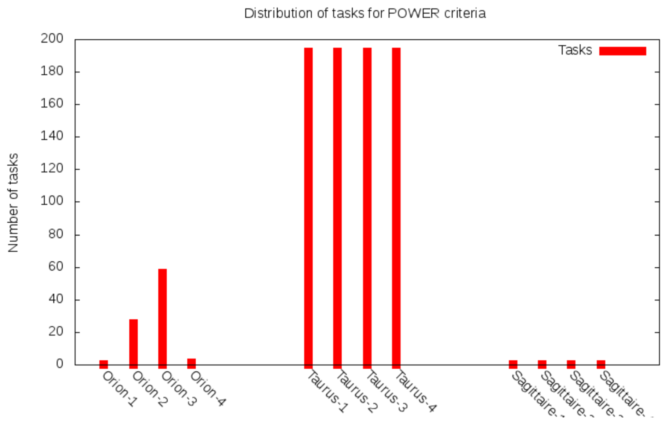

Fig. 3. Distribution of jobs using power consumption as placement criteria.

Figure 3 describes the distribution according to the power consumption criteria. Observing Figure 3, we observe that most of the jobs are computed on *Taurus* nodes, which appears to be the most energy-efficient. Computation on *Orion* and *Sagittaire* occurs during the "learning" phase or when the *Taurus* nodes are out of capacity.

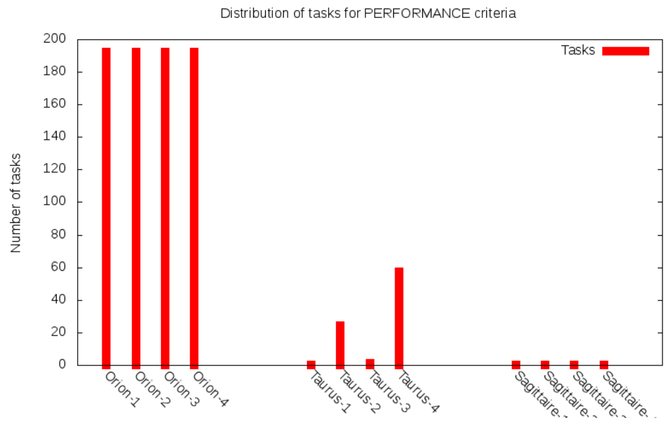

Fig. 4. Distribution of jobs using performance as placement criteria.

Figure 4 describes the distribution of tasks when performance is the criteria when selecting a node. The load balancing of jobs is similar to Figure 3, with the majority of tasks on Orion nodes.

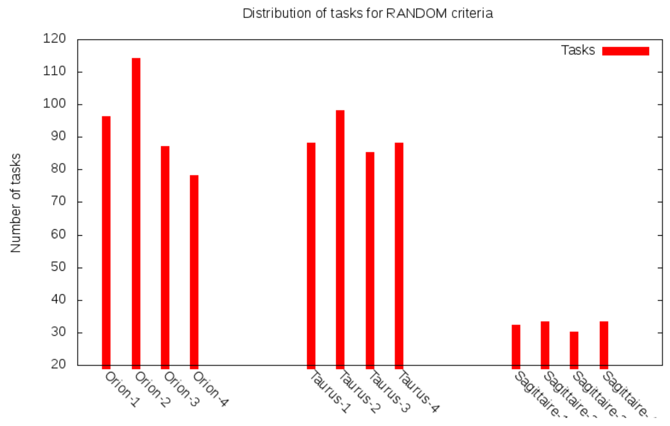

Fig. 5. Distribution of jobs with random placement.

In Figure 5, despite a random distribution of jobs, *Sagittaire* nodes are computing less tasks than other nodes. It is explained by the fact that a single task is computed slower on those nodes. Thus, they are less frequently available when decisions are performed.

Figure 6 presents the energy consumption of the whole infrastructure grouped by clusters. We precise that the energy consumption measured on the DIET agents were constant when executing the three algorithms and does not present any influence on the comparison. We can observe that distributing the workload using the RANDOM policy is not particularly energy efficient as it guarantees that all the resources are in use during the experiment.

We use Table IV to compare makespan and energy consumption metrics among the scheduling policies.

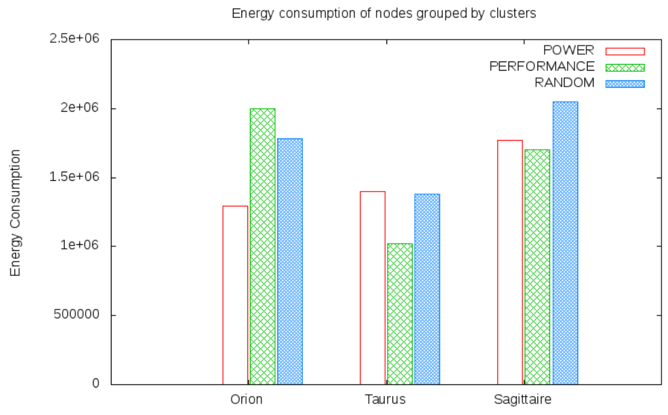

Fig. 6. Energy consumption per cluster.

	RANDOM	POWER	PERFORMANCE
Makespan (s)	2336	2321	2228
Energy (J)	6041436	4528547	5618175

TABLE IV. EXPERIMENTAL RESULTS

Observing the performance, the best case is rationally observed when setting a priority on nodes with a high number of Flops (PERFORMANCE). Comparing that value with the POWER makespan, we noticed a loss of performance up to 6%.

In terms of energy consumption, the POWER policy presents a gain of 25% when compared to the RANDOM, and up to 19% compared to PERFORMANCE.

Random appears to be the worst case due to the fact that it ensures that all the nodes are in use, resulting of a higher energy consumption. The use of slow nodes is also impacting the performance but this effect is reduced by a covering up effect (Fast nodes will compute more tasks in parallel).

C. Evaluation of GreenPerf

We evaluate the GreenPerf metric in order to establish the relevancy of the ratio $\frac{\text{Power Consumption}}{\text{Performance}}$ in high and low heterogeneity environments.

In this part, we use a simulation to manage the level of heterogeneity. After performing a benchmark on the physical nodes of GRID'5000, we obtained for each server its mean computation time for a single task along with the peak and idle power consumptions. Those values are used to sum up the energy consumption of the whole infrastructure during the simulations. Each task is computed with the maximal power consumption of the servers. During the simulation, each server is limited to the computation of one task.

Figure 7 shows the comparison of metrics with low heterogeneity environment. In this scenario, we use 2 different type of servers with similar specifications (Table III).

The coordinates of G, GP and P represent the average values obtained of, respectively, the POWER, GREENPERF and PERFORMANCE metrics. The shadings represent the area of RANDOM values.

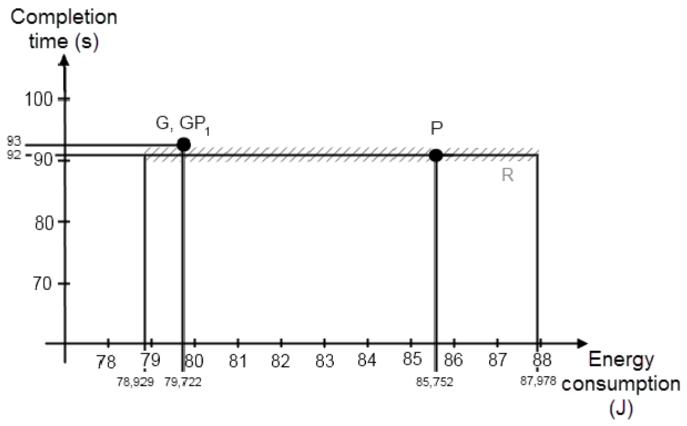

Fig. 7. Comparison of metrics with 2 different types of servers and 2 clients submitting requests.

In a second scenario, we consider two other types of clusters (Table V) to increase the heterogeneity of the platform. Figure 8 shows a better tradeoff between POWER and PERFORMANCE, putting in light the need of a sufficient diversity of hardware to efficiently use GreenPerf.

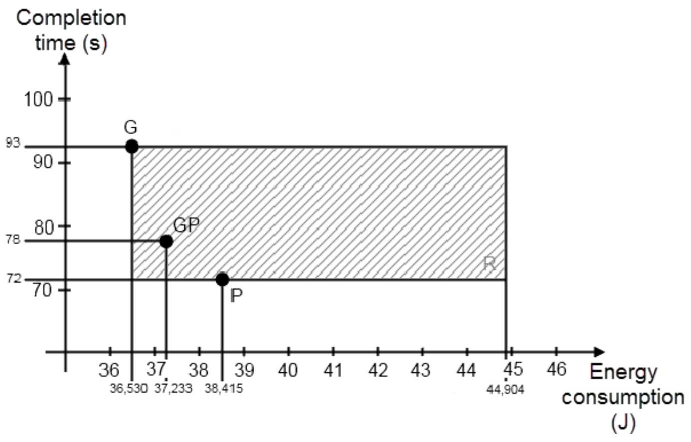

Fig. 8. Comparison of metrics with 2 different types of servers and 2 clients submitting requests.

Cluster	Idle consumption	Peak consumption
Sim1	230	190
Sim2	190	160

TABLE V. ENERGY CONSUMPTION OF SIMULATED CLUSTERS

D. Context-aware resource provisioning

The third experiment intends to demonstrate the reactivity of the scheduler by considering fluctuations of two metrics over time: the cost of electricity and the temperature. We inject energy-related events at the scheduler level while a client, aware of the number of available nodes, is submitting a continuous flow of requests intending to reach the capacity of the infrastructure.

In the interests of simplification, the cost of energy is defined as a ratio between the cost for a given period and the theoretical maximum cost. Related to the cost of energy, we defined three states:

- Regular time, when the electricity cost is the highest (1.0)
- Off-peak time 1, when the electricity cost is a less expensive during than regular time (0.8)
- Off-peak time 2, when the electricity cost is the least expensive (0.5)

Heat measurements are defined through two states. In this example, we consider a single temperature measure within the infrastructure, described by the following states:

- In-range temperature of utilization (< 25 degrees)
- Out-of-range temperature (> 25 degrees)

The scenario involves the injection of four different events in the provisioning planning of the scheduler, splitted into two categories: scheduled and unexpected. The scheduled events correspond to a prediction or a predefined schedule, that is known before its occurrence by the scheduler. Unexpected events can be discovered only at the exact time of occurrence, such as heat alerts.

Let t be the start time of the experiment:

Event 1 At $t+60$ min, the electricity cost decreases at 0.8 (scheduled event)
Event 2 At $t+120$ min, the electricity cost decreases at 0.5 (scheduled event)
Event 3 At $t+160$ min, a raise of temperature is detected (unexpected event)
Event 4 At $t+240$ min, the temperature get back to an acceptable value (unexpected event)

The status of the platform corresponds to the value of the exploitation metrics at an instant *t*. In this example, we consider that the Master Agent is checking the status of the platform every 10 minutes, with the ability to get information about the scheduled events occurring at an instant *i+2*.

On Figure 9, we present a sample of the provisioning planning. It is an XML shared file using a readers-writers lock that refers to a specific time-stamp. For each sample, we defined three tags, namely *temperature*, *candidates* and *electricity_cost*. At each time interval, the scheduler performs decisions according to the value of the tags. Thus, future information, such as forecasts, can be add to the provisioning planning, ensuring a dynamic behavior regarding to the various contexts. The tags and time interval are customizable variables that can be adjusted to fit specific contexts.

```xml
<timestamp value="1385896446">
  <temperature>23.5</temperature>
  <candidates>8</candidates>
  <electricity_cost>0.6</electricity_cost>
</timestamp>
```

Fig. 9. Sample of the server status.

We set thresholds and actions to be triggered according to the value of the thresholds. The actions can be defined through scripts or commands to be called by the scheduler. In this example, we implemented five behaviors associated to the experiment metrics. Let c be the cost of energy for a given period and T the temperature measured at an instant t.

- if $T > 25$ then candidate_nodes $= 20\%$ of all nodes
- if $1.0 >= c > 0.8$ then candidate_nodes $= 40\%$ of all nodes
- if $0.8 >= c > 0.5$ then candidate_nodes $= 70\%$ of all nodes
- if $c < 0.5$ then candidate_nodes $= 100\%$ of all nodes

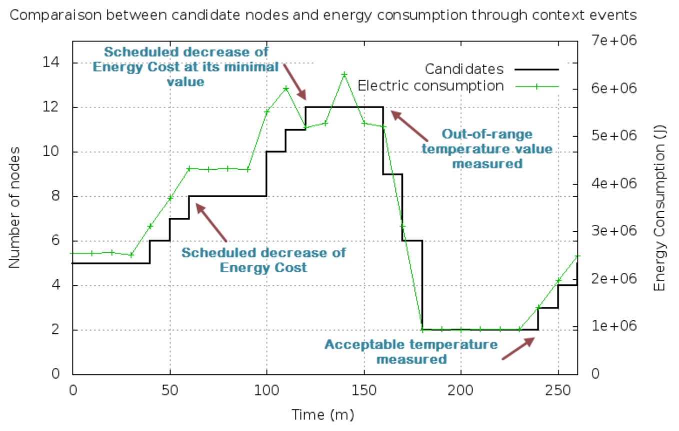

Fig. 10. Evolution of candidate nodes and power consumption.

Figure 10 presents the evolution of the number of candidate nodes and the electric consumption over time. The left y-axis shows the total number of nodes in the infrastructure. The plain line presents the number of candidates during the experiment. The line with crosses is the evolution of the energy consumption, using the right y-axis. Each cross describes an average value of energy consumption measured during the previous 10 minutes. The x-axis represents the time with a total of 260 minutes.

The infrastructure is deployed on GRID'5000, composed of the nodes defined in Table III. The experiment starts with an energy cost value of 1.0 and a $Preference_{provider}(u, c)$ giving priority to energy-efficient nodes. The $Preference_{user}$ is not having any influence in the current scenario as the client dynamically adjusts its flow of request to reach the capacity of available nodes.

Event 1 is a decrease of the electricity cost occurring at $t+60$ min. The Master Agent get aware of that information at $t + 40$ min. Observing a future cost of 0.8, the Master Agent plans ahead to provide 8 candidates nodes at $t + 60$ min. The set of candidates is incremented by small subsets of nodes to obtain a progressive start, at $t + 50$ min and $t + 60$ min. (it avoids heat peaks due to side effect of simultaneous starts).We observe a linear increase of electric consumption through the infrastructure. After each completion of request, the Client is notified of the current amount of candidates nodes, and is free to adjust its request rate.

Event 2 is similar to **Event 1**. The electricity cost is allowing the use of every available node in the architecture. Those nodes are added to the set of candidates during the following 20 minutes, resulting in a use of all possible nodes between $t + 120$ and $t + 160$ min.

Event 3 simulates an instant raise of temperature, detected by the Master Agent at $t+160$ min. According to administrator rules, the predefined behavior is to reduce the number of candidates nodes to 2. It is performed in 3 steps, in order to cause a drop of heat (simulated) and energy consumption (measured). We allow tasks in progress to complete, resulting in a delayed drop of energy consumption. The system keeps on working with 2 candidates until an acceptable temperature is measured at $t + 240$ min (**Event 4**). The Master Agent then starts to provision the pool of candidates, every 10 minutes to reach again the value of 12.

The scenario of that experiment shows the reactivity of the scheduler and the ability to manage energy-related events by adapting the number of provisioned resources of the physical infrastructure, therefore its power consumption.

VII. DISCUSSION

This paper discusses positive results concerning the use and management of a reactive middleware with green capabilities. However, we used arbitrary rules to perform corrective actions during the experiments. Parameters such as the amount of nodes to add/remove from the set of candidates must be expressed in relation with the temperature by taking into account the heat dissipation. Those concerns are considered to be out of the scope of this paper.

VIII. CONCLUSIONS AND PERSPECTIVES

Computing as an utility highly uses distributed computing for scientific as well as commercial applications. Due to aggregation of computing, network and storage hardware, efficient workload placement is a major concern and needs to be made across the whole infrastructure with consideration of energy efficiency. Previous research typically address this problem through power management techniques that aim at maximizing the utilization of resources.

We propose methods for provisioning resources and distribute requests with the objective of meeting performance requirements while reducing power consumption. We validate our strategy through real life experimentation using the DIET toolkit and the GRID'5000 experimental testbed.

Comparing three different scheduling policies by providing tunable comparison based on the performance and the energy consumption, results show a gain of energy consumption up to 25% with a minor loss of performance (6%).

We propose and evaluate a hybrid metric taking into account performance and power consumption as a ratio of energy efficiency. The effectiveness of this metric strongly relies on the heterogeneity of the different servers.

The second part considers the provisioning of resources, while taking into account energy-related events and user preferences. Results show a reactive scheduling, allowing policy management to be abstracted into a software layer that can be automated and controlled centrally. We expect this approach to be very useful when applied to provisioning servers, using contextual data from third-party predicting or monitoring tools.

Future work will revolve around fine-grained scheduling by taking into account spatial information. We intend to leverage control over energy consumption by considering budget constrained scheduling.

Acknowledgements

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under Inria ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies[7].

Some of the work reported here was conducted while Barbara WALTER (INSA Lyon, France) and Joran BIGALET (ENS, France) did their internships in the LIP.

References

[1] W. chun Feng, X. Feng, and R. Ge, "Green supercomputing comes of age," *IT Professional*, vol. 10, no. 1, pp. 17–23, 2008. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MITP.2008.8

[2] J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Aloisio, J.-C. Andre, D. Barkai, J.-Y. Berthou, T. Boku, B. Braunschweig, F. Cappello, B. Chapman, X. Chi, A. Choudhary, S. Dosanjh, T. Dunning, S. Fiore, A. Geist, B. Gropp, R. Harrison, M. Hereld, M. Heroux, A. Hoisie, K. Hotta, Z. Jin, Y. Ishikawa, F. Johnson, S. Kale, R. Kenway, D. Keyes, B. Kramer, J. Labarta, A. Lichnewsky, T. Lippert, B. Lucas, B. Maccabe, S. Matsuoka, P. Messina, P. Michielse, B. Mohr, M. S. Mueller, W. E. Nagel, H. Nakashima, M. E. Papka, D. Reed, M. Sato, E. Seidel, J. Shalf, D. Skinner, M. Snir, T. Sterling, R. Stevens, F. Streitz, B. Sugar, S. Sumimoto, W. Tang, J. Taylor, R. Thakur, A. Trefethen, M. Valero, A. van der Steen, J. Vetter, P. Williams, R. Wisniewski, and K. Yelick, "The International Exascale Software Project roadmap," *The International Journal of High Performance Computing Applications*, vol. 25, no. 1, pp. 3–60, Feb. 2011. [Online]. Available: http://hpc.sagepub.com/content/25/1/3.full.pdf+html

[3] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Trans. Parallel Distrib. Syst*, vol. 24, no. 7, pp. 1366–1379, 2013. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.240

[4] E. Feller, L. Rilling, and C. Morin, "Snooze: A scalable and autonomic virtual machine management framework for private clouds," in *CCGRID*. IEEE, 2012, pp. 482–489. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6217395

[5] E. Caron and F. Desprez, "DIET: A scalable toolbox to build network enabled servers on the grid," *International Journal of High Performance Computing Applications*, vol. 20, no. 3, pp. 335–352, 2006.

[6] A. Hawkins, "Unused servers survey results analysis," *The Green Grid*, 2010.

[7] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE computer*, vol. 40, no. 12, pp. 33–37, 2007.

[8] E. V. Carrera, E. Pinheiro, and R. Bianchini, "Conserving disk energy in network servers," in *Proceedings of the 2003 International Conference on Supercomputing (ICS-03)*. New York: ACM Press, Jun. 23–26 2003, pp. 86–97.

[9] D. C. Snowdon, S. Ruocco, and G. Heiser, "Power management and dynamic voltage scaling: Myths and facts," in *Proceedings of the 2005 Workshop on Power Aware Real-time Computing*, Sep. 2005. [Online]. Available: http://www.ertos.nicta.com.au/publications/papers/Snowdon_RH_05.pdf

[10] E. Le Sueur and G. Heiser, "Dynamic voltage and frequency scaling: The laws of diminishing returns," in *Proceedings of the 2010 international conference on Power aware computing and systems*. USENIX Association, 2010, pp. 1–8.

[11] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," *CoRR*, vol. abs/0901.0131, 2009. [Online]. Available: http://arxiv.org/abs/0901.0131

[12] F. Hermenier, N. Loriant, and J.-M. Menaud, "Power management in grid computing with xen," in *Proceedings of the 2006 International Conference on Frontiers of High Performance Computing and Networking*, ser. ISPA'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 407–416. [Online]. Available: http://dx.doi.org/10.1007/11942634_43

[13] A.-C. Orgerie and L. Lefèvre, "When Clouds become Green: the Green Open Cloud Architecture," *Parallel Computing*, vol. 19, pp. 228 – 237, 2010. [Online]. Available: http://hal.inria.fr/ensl-00484321

[14] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron, "PowerPack: Energy profiling and analysis of high-performance systems and applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, no. 5, pp. 658–671, May 2010.

[15] K. H. Kim, R. Buyya, and J. Kim, "Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters," in *CCGRID*. IEEE Computer Society, 2007, pp. 541–548. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2007.85

[16] G. Varsamopoulos, A. Banerjee, and S. K. S. Gupta, "Energy efficiency of thermal-aware job scheduling algorithms under various cooling models," in *Contemporary Computing - Second International Conference, IC3 2009, Noida, India, August 17-19, 2009. Proceedings*, ser. Communications in Computer and Information Science, S. Ranka, S. Aluru, R. Buyya, Y.-C. Chung, S. Dua, A. Grama, S. K. S. Gupta, R. Kumar, and V. V. Phoha, Eds., vol. 40. Springer, 2009, pp. 568–580. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-03547-0

[17] M. E. M. Diouri, O. Glück, L. Lefèvre, and J.-C. Mignot, "Your cluster is not power homogeneous: Take care when designing green schedulers!" in *IGCC-4th IEEE International Green Computing Conference*, 2013.

[18] N. Capit, G. Da Costa, Y. Georgiou, G. Huard, C. Martin, G. Mounié, P. Neyron, and O. Richard, "A batch scheduler with high level components," in *Cluster computing and Grid 2005 (CCGrid05)*, 2005.

[19] D. Wentzlaff, C. Gruenwald, III, N. Beckmann, K. Modzelewski, A. Belay, L. Youseff, J. Miller, and A. Agarwal, "An operating system for multicore and clouds: Mechanisms and implementation," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10. New York, NY, USA: ACM, 2010, pp. 3–14. [Online]. Available: http://doi.acm.org/10.1145/1807128.1807132

[20] C.-Y. Tu, W.-C. Kuo, W.-H. Teng, Y.-T. Wang, and S. Shiau, "A power-aware cloud architecture with smart metering," in *Proc. Second International Workshop on Green Computing (2nd GreenCom'10), 2010 International Conference on Parallel Processing Workshops (39th ICPPW'10) CD-ROM*. San Diego, CA: CPS/IEEE Computer Society, Sep. 2010, pp. 497–503.

[21] S.-H. Lim, B. Sharma, B.-C. Tak, and C. R. Das, "A dynamic energy management scheme for multi-tier data centers," in *ISPASS*. IEEE Computer Society, 2011, pp. 257–266. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5755445

[22] M. D. De Assuncao, A.-C. Orgerie, and L. Lefevre, "An analysis of power consumption logs from a monitored grid site," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*. IEEE, 2010, pp. 61–68.

[23] M. D. De Assuncao, J.-P. Gelas, L. Lefèvre, and A.-C. Orgerie, "The green grid'5000: Instrumenting and using a grid with energy sensors," in *Remote Instrumentation for eScience and Related Aspects*. Springer, 2012, pp. 25–42.

[7]The Grid'5000 testbed. http://www,grid5000.fr