HAL
open science

Involving Users in Energy Conservation: A Case Study in Scientific Clouds

David Guyon, Anne-Cécile Orgerie, Christine Morin, Deb Agarwal

To cite this version:

David Guyon, Anne-Cécile Orgerie, Christine Morin, Deb Agarwal. Involving Users in Energy Conservation: A Case Study in Scientific Clouds. International Journal of Grid and Utility Computing, 2019, 10 (3), pp.272 - 282. hal-01711515

HAL Id: hal-01711515

https://hal.science/hal-01711515v1

Submitted on 18 Feb 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Involving Users in Energy Conservation: A Case Study in Scientific Clouds

David Guyon

Univ Rennes, Inria, CNRS, IRISA, France E-mail: david.guyon@irisa.fr

Anne-Cécile Orgerie

Univ Rennes, Inria, CNRS, IRISA, France E-mail: anne-cecile.orgerie@irisa.fr

Christine Morin

Univ Rennes, Inria, CNRS, IRISA, France E-mail: christine.morin@inria.fr

Deb Agarwal

Lawrence Berkeley National Lab. Berkeley, USA e-mail: daagarwal@lbl.gov

Abstract: Services offered by cloud computing are convenient to users for reasons such as their ease of use, flexibility, and financial model. Yet data centres used for their execution are known to consume massive amounts of energy. The growing resource utilisation following the cloud success highlights the importance of the reduction of its energy consumption. This paper investigates a way to reduce the footprint of HPC cloud users by varying the size of the virtual resources they request. We analyse the influence of concurrent applications with different resources sizes on the system energy consumption. Simulation results show that resources with larger size are more energy consuming regardless of faster applications' completion. Although smaller-sized resources offer energy savings, it is not always favourable in terms of energy to reduce too much the size. High energy savings depend on the user profiles' distribution.

Keywords: Cloud computing; green computing; HPC applications; energy savings; users involvement.

Reference to this paper should be made as follows: David Guyon, Anne-Cécile Orgerie, Christine Morin and Deb Agarwal 'Involving Users in Energy Conservation: A Case Study in Scientific Clouds', *International Journal of Grid and Utility Computing*, pp.1–14, 2018.

Biographical notes: David Guyon is a PhD student in Computer Science at the University of Rennes 1 (France). He is part of the Myriads project-team in the IRISA laboratory (Rennes, France). His thesis research is in the context of green cloud computing and focuses on the involvement of cloud users in order to achieve energy savings. He will defend his thesis on 2018.

Anne-Cécile Orgerie received her PhD degree in Computer Science from Ecole Normale Superieure de Lyon (France) in September 2011. From October 2011 to September 2012, she was working as a postdoc at the University of Melbourne (Australia). She has been a full time researcher at CNRS in the IRISA laboratory (Rennes, France) since October 2012. Her research interests focus on green computing, energy efficiency, cloud computing, and distributed systems. She has authored 5 book chapters, 10 journal publications, and 34 international peer-reviewed conference articles.

Christine Morin is a senior researcher at Inria and leads Myriads team on the design and implementation of autonomous distributed systems. She has also been an affiliate at the Lawrence Berkeley Laboratory since 2011. Her research interests are in distributed systems, green and cloud computing. She has advised more than 20 PhD students. She coordinated the XtreemOS and Contrail European projects, respectively on grid and cloud computing. She co-founded the Kerlabs start-up in 2006. She received her PhD degree and Habilitation à Diriger des Recherches in Computer Science from the University of Rennes 1 in 1990 and 1998, respectively.

Dr. Deborah Agarwal is a Senior Scientist and the Data Science and Technology Department (`http://dst.lbl.gov`), Head at Lawrence Berkeley National Laboratory (LBNL). Dr. Agarwals research is in distributed systems and data science. Her current research focuses on developing computational tools to enable scientists to more effectively organize and use their data to understand and mitigate climate change. She has worked on projects involving watershed understanding, tropical forests, soil carbon, carbon capture, cosmology, particle accelerators, and satellite data.

1 Introduction

High Performance Computing (HPC) infrastructures are usually massive buildings that contain hundreds of servers with powerful hardware (insideHPC 2012). These infrastructures run scientific applications requiring tremendous amounts of computing resources to execute. These applications are often organised in workflow structures and each step of the workflow may be a computation that needs important amounts of CPU, memory and storage resources.

Cloud computing has become a cost effective alternative to HPC machines (Gupta et al. 2013) and some of the less resource intensive HPC applications tend to migrate to clouds (Gupta & Milojicic 2011). Cloud computing offers *elasticity*, which allows to adjust the provisioning of resources according to the applications varying workload while maintaining the desired SLA. A user only pays for the resources its application is using and, in some cases, the application execution has an overall lower pricing compared to HPC solutions (Gupta & Milojicic 2011).

As a consequence of the cloud computing success, the global energy consumed by data centres has increased significantly. They consumed 1% of the global electricity in 2010 and this consumption is predicted to reach between 3 to 13% in 2030 (Andrae & Edler 2015). We are currently facing important climate changes which call for a reduction of the ecological impact of computing. To reduce the electrical consumption of cloud infrastructures, *consolidation* mechanisms pack the virtual machines (VMs) on the least number of servers, without impacting application performance, in order to turn off the unused servers in case of moderate load.

Idle servers indeed consume extensive amounts of energy (Orgerie et al. 2014). However, such consolidation techniques are only efficient if virtual resources are not kept idle by the users for no work. Indeed, if the cloud provider does not over-commit the physical resources, the user that uses only partly the virtual machines resources is wasting the rest. Thus, energy-efficient users need to properly size their VMs. For a given parallel application, several VM sizes are possible, each offering a different trade-off between the overall energy consumption and the performance (i.e. runtime). This trade-off is complex to determine: small-sized VMs may be easier to pack into server machines, while larger VMs may end their work faster. While it is logical that well-dimensioned machines are more energy efficient, defining their size is not an easy task for the users.

In a previous work (Guyon et al. 2015), we present a cloud system involving users in the energy optimisation system. A user who agrees to reduce her impact on the environment can choose a more energy-efficient execution mode, implying a loss in performance, by executing her application on less resources on the infrastructure. The unused resources are free for another application and thus, this approach favours a better consolidation of the whole system. The better the consolidation, the lower the electrical consumption. The proposed system offers three execution modes based on (Villebonnet et al. 2015): *Big*, *Medium* and *Little*. An algorithm selects the size of the VMs for executing each task of the workflows depending on the selected execution mode. The *Medium* mode executes using the user-specified VM resources for each workflow stage. The *Little* and *Big* modes respectively decrease or increase the VMs by one size for the whole workflow.

In the present paper, we evaluate the impact of the proportion of users selecting the *Big*, *Medium* or *Little* mode on a data centre's energy consumption. Our evaluations have been done using three kinds of scientific workflows, energy consumption measurements for the execution of these workflows on a real platform and traces of jobs submitted to a production HPC centre. We evaluated the data centre energy consumption for different proportions of users selecting the three available modes. The simulation results show promising energy savings when the amount of users selecting the *Big* mode is low. They also show that using the *Little* mode compared to

the *Medium* mode does not always provide the best performance/energy saving trade-off.

The paper is organised as follows. Section 2 presents our methodology. The experimental setup is explained in Section 3 and the simulations' results detailed in Section 4. Section 5 discusses the limitations of our system. We present the related work in Section 6 and conclude the paper in Section 7.

2 Methodology

For evaluating the impact of energy-aware users on an HPC cloud, we conducted an experimental study using a real public workload trace from a production data centre. A job in our workload is an execution of one of three different scientific applications. The energy consumption of these applications running with all possible execution modes was measured on a real cloud infrastructure. Each job runs with an execution mode and we varied this distribution of the modes in order to have different profiles of user population. The energy consumption of the data centre is calculated with each profile distribution in order to evaluate the impact of the execution mode choices on the data centre's electrical consumption.

The nodes of the simulated data centre are inspired by the nodes of the Taurus cluster of Grid'5000, a French platform for experimenting distributed systems. Each node of this cluster has 12 Intel Xeon E5-2630 CPU cores, 32GB of memory, 598GB of hard drive and a 10 Gigabit Ethernet connection.

For executing the applications on the hardware we used virtualisation based on the KVM technology (Kivity et al. 2007). A VM has a fixed size in terms of CPU, memory and disk resources. We considered different kinds of VMs with different amounts of CPU, memory and disk resources, called *flavors*. We designed 5 flavors with a virtual resource configuration similar to those offered by *Amazon EC2* (2017) cloud. The list of flavors used in our system is presented in Table 1. This table also contains the EC2 instance equivalent and their US East hourly pricing.

In our cloud system, incoming jobs are executed directly and cannot be batched for a later execution. Any job submission implies a VM creation for each task of the workflow. A consolidation mechanism creates the VMs on specific servers in order to optimise their resource utilisation (*greedy* algorithm). If a server does not host any VM, it is powered down in order to reduce the data centre's power consumption.

The workload corresponds to the job submission distribution over a day. We took a 2 year long trace from a real production HPC platform located in the Czech republic (Feitelson et al. 2014). From this trace we analysed the daily submission distribution and used a k-mean algorithm to find different distribution profiles. From these profiles we retained one with a submission peak during the working hours and another one with a constant submission rate.

A job submission is a request to start an application. We selected real scientific applications that execute as a workflow. Workflows are composed of sequence of sequential and/or parallel tasks with data dependencies. We selected 3 applications from different scientific areas that exhibit different behaviours in terms of resource consumption: disk-intensive, CPU-intensive and memory-intensive. The chosen applications are the following ones: *Montage*, *Blast* and *Palmtree*. They are presented in more detail in Section 3.3.

Each job runs according to an execution mode. This mode has an impact on the size of the VMs where the job is running and consequently on the execution time of the job. A probabilistic distribution algorithm takes as input the percentage of job in each execution mode. As output, the algorithm fairly distributes the 3 workflows to each job and sets each job execution mode following the input.

The electrical consumption of the servers has been measured thanks to the fine-grained watt-meters available on them (De Assuncao et al. 2012). Three measures have been recorded: when the server is powered down, when the server is on but not used (*idle*) and when

Table 1 Details of the VM flavors used in the system with their Amazon EC2 instance equivalent and their US East hourly pricing.

Flavor	RAM	CPU	Disk	EC2 instance equivalent	
tiny	0.5 GB	1	5 GB	t2.nano	$0.0065
small	2 GB	1	20 GB	t2.small	$0.026
medium	4 GB	2	40 GB	t2.medium	$0.052
large	8 GB	4	80 GB	c4.xlarge	$0.209
xlarge	16 GB	8	160 GB	c4.2xlarge	$0.419

it is fully used. The energy consumption of the workflows has also been measured. The execution logs of each workflow in each execution mode contains the run time and the energy consumption of each task. To obtain accurate electrical measures, the servers were loaded at their maximum capacity by duplicating the tasks running on them. Then the dynamic consumption is distributed evenly across the tasks. This part is explained in more details in Section 3.4.

2.1 Assumptions

In this work, we consider resource-intensive scientific applications submitted by users to a dedicated Cloud platform. We make the following assumptions:

- a user application is a workflow composed of one or more sequential steps, each step having one or more parallel tasks ;
- each task of a workflow executes in a separate VM ;
- each task can exploit all the cores available in its VM, whatever the number of cores. It is the users' responsibility to implement tasks that automatically adapt their execution to use all the available cores ;
- a VM has always enough disk space and memory for the task to execute, even when the *Little* execution mode is selected ;
- a user application starts to be executed as soon as enough resources are available (i.e. idle or switched off servers or unused resources in servers already running VMs) to execute it ;

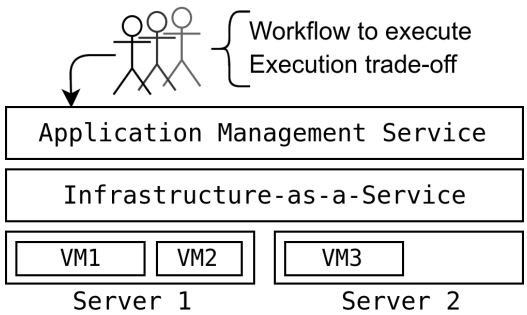

Figure 1 The users send their applications description, with the selected execution mode for each of them, to the cloud infrastructure which contains the servers hosting the users' VMs.

- cloud servers hosting the VMs have dynamic frequency scaling feature (DVFS) enabled ;
- cloud servers do not consider CPU over-commit policies as it can disturb the execution of CPU intensive applications.

To sum up, users' applications are scientific workflows that are composed of several VMs, each with a size calibrated according to the task it needs to execute. Here, we do not consider VM elasticity (i.e. dynamic resizing) as it is already implemented in a static way through the definition of the applications workflow.

2.2 System

The system architecture is presented in Figure 1. The user, at the top, sends a request to execute her workflow application. The request contains the workflow structure (number of steps and parallel tasks) and the amount of CPU, memory and disk space required by default by each step. She also indicates the execution mode for the run of her application.

Inspired by the ARM big.LITTLE (which is a heterogeneous processor) Villebonnet et al.

(2015) introduce the *Big*, *Medium* and *Little* (BML) infrastructure. Their idea consists in reaching energy proportionality by using heterogeneous processors for variable workloads: if the workload is low, it is executed on the Little processor, while when it is high, it smoothly migrates to the Big processor. Similarly, in our system the VMs' sizes for executing a workflow are chosen according to an energy/performance trade-off depending on the execution mode selected by the user. This is why in the rest of the paper, we opt for the same terminology which is easier to handle and it highlights the main variable of our system: the VM size (to avoid confusion between the *Medium* mode and the *medium* flavor, the modes always start with a capital letter).

The size of the VM for a given task is selected according to the specified amount of CPU, memory and disk space required (Section 3.3 details how the resource amount is defined). The VM flavor with just enough resources is the one selected for the *Medium* execution mode. The *Big* execution mode selects the VM flavor one size larger and for the *Little* mode, it selects the VM flavor one size smaller. For example, an application asking for an amount of resources matching the *medium* flavor will be assigned the *large* VM flavor in the *Big* execution mode and the *small* VM flavor in the *Little* execution mode.

A VM placement algorithm creates the VMs on specific servers in order to favour the consolidation of the whole system and reduce the global energy consumed. A simple *greedy* algorithm (Yue 1991) implementation is used to solve this complex bin packing problem. The servers are sorted in ascending order of available resources and the first one suitable for the VM creation is selected. This algorithm avoids the fragmentation of VMs across servers.

3 Experimental Setup

An evaluation of the energy consumed in our cloud system with different user profile distributions has been conducted using simulation. In order to have a simulator as realistic as possible, we took a job arrival trace from an existing HPC centre, we selected real scientific applications that we ran on a cloud infrastructure to get execution logs and finally we designed the simulated infrastructure based on the hardware configuration of the real cluster we used. This is described in more detail in the remainder of this section.

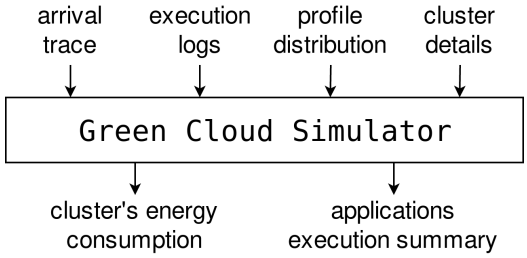

Figure 2 Simulator's architecture with its inputs and outputs.

3.1 Simulator

A cloud simulator has been developed instead of extending an existing one because we did not find any simulator including the utilization of applications' power profiles. Our simulator, written in Python, reproduces the behavior of a cloud system that takes the users into consideration in order to optimise the energy efficiency of the whole system. This simulator takes the following inputs:

- an arrival trace of request submissions (workload) based on real data in order to have a realistic use case ;
- a panel of execution logs of scientific applications measured on a real cloud infrastructure that ran with the three execution modes ;
- profile distribution probability represented by a percentage parameter to configure the amount of applications to execute in the different modes ;
- information about the servers to use in the simulated data centre such as the hardware resources (CPU, disk and memory) and the power consumption of a single node in idle and off states based on real measurements on the machines we used to run the scientific applications.

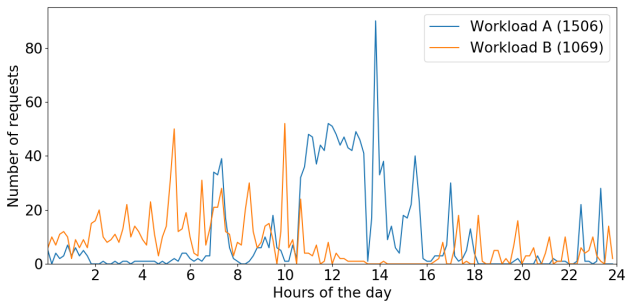

Figure 3 Workloads based on a real trace from the utilisation logs of the MetaCentrum Czech National Grid.

Each job submission in the workload simulates the execution of an application starting at a specific time (date during the day) using the arrival trace and is attributed an execution mode with respect to the profile distribution probability given as parameter.

The output of the simulator is the energy consumed during a whole day by the workload run on the simulated cloud infrastructure. It also generates the complete simulation log details for debugging purpose.

3.2 Arrival Trace

We used a realistic job submission trace as an input of our simulator. The original trace (*The MetaCentrum 2 log* 2013) is 2 years long and comes from the utilisation records of the MetaCentrum Czech National Grid (`www.metacentrum.cz`). The grid is composed of 30 Linux clusters, each with several multiprocessor machines, for a total greater than 12000 cores. The very detailed logs provide for each job submission: job ID, user ID, CPU and memory used, arrival time, start and end times, job duration and others.

We only analysed the submission time of each job, grouped the data by day and used a k-mean algorithm to group the days with similar job submission distributions. An analysis with 6 groups (k-mean with $k = 6$) gives meaningful results. From these groups, two typical candidates have been retained and are presented in Figure 3. The k-mean plot of the third group of this series rises at around 8AM and starts to decrease at 5PM, this group represents a typical working day job submissions distribution. On this group, the retained candidate (a day in the two year archive) has 11855 job submissions. In order to fit our targeted cluster, we generated a reduced version of this candidate (random selection) with 1506 submissions which is presented in Figure 3 under the name *Workload A*. This distribution reveals a normal working day with a submission peak at 7AM, a constant job submissions from 10:30AM until lunch time, another peak after lunch at 2PM and no more submissions after 6PM. The second candidate presented under the name *Workload B* is a plot of the sixth group given by k-mean. It has 8293 submissions which is reduced to 1069 with the same calculation as the first candidate. It represents a less loaded day with a smoother distribution over the day. In contrast with *Workload A*, the latter refers to an infrastructure that is not only used locally but rather at a worldwide scale (arrival of jobs is distributed regardless the time of the day).

Table 2 Flavors used for each step of the workflows in the three different execution modes.

Montage	Big	Medium	Little
DSS2*	large	medium	small
PNG	large	medium	small
Blast	Big	Medium	Little
Blastn	xlarge	large	medium
Palmtree	Big	Medium	Little
Stream #*	xlarge	large	medium

3.3 Execution Logs

The simulator utilises execution logs of workflow applications that we ran on a real cloud infrastructure. Three scientific applications from completely different research domains have been carefully selected in order to represent the computations we can find in data centres, such as memory-intensive, data-intensive and CPU-intensive tasks.

3.3.1 Montage Workflow

Montage (2017) is an engine to build astronomical image mosaics for astronomers. Its workflow structure is shown on the left of Figure 4. The first step downloads large

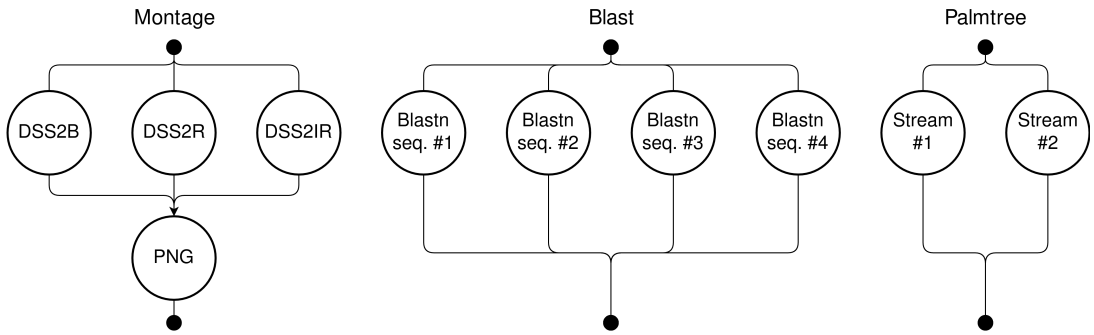

Figure 4 Structure of the workflows used in our system. From left to right: Montage has 3 parallel tasks for the 1st step and 1 task for the 2nd one, Blast has 4 parallel tasks for its 1st step and Palmtree has 2 parallel tasks for its 1st step.

amounts of data of a specific area of space and then runs calculations on the data to generate intermediate data. This important task is split into 3 parallel tasks. Then the second step, composed of a single task, creates the final mosaic (a PNG file) thanks to the 3 intermediate data given by the first step. In our case the workflow calculates the mosaic of the Pleiades location for a 2 angular degrees wide area. The input data represents about 5.5GB and 67MB for the output file which makes the workflow mainly IO-intensive and CPU-intensive during the calculation.

The number of hardware resources given to each task has been selected by experimentation in order to have a *Medium* execution that runs for less than an hour. The tasks of the first step need 2 cores, 2 GB of RAM and 10 GB of disk space. The second step requires 1 core, 4 GB of RAM and 20 GB of disk space. Table 2 shows that for an execution with the *Medium* mode, the tasks execute in *medium* sized VMs.

3.3.2 Blast Workflow

Basic Local Alignment Search Tool (Blast) (2017) is a program that compares nucleotide or protein sequences to sequence databases and calculates the statistical significance of matches. The figure in the middle of Figure 4 shows the structure of the Blast workflow. It has 4 parallel tasks, each searching for a match from a file containing 10 000 nucleotide sequences into the complete nucleotide sequences database of mouse. The execution of the workflow has a cyclic use of the memory and constantly uses the CPUs, making it a memory-intensive and a CPU-intensive application.

Again, the required hardware resources have been selected by experimentation. Each task asks for 4 cores, 2 GB of RAM and 10 GB of disk space and executes for about an hour. The selected VM flavors are *large* when the application runs with the *Medium* execution mode. VM flavors used for all execution mode are listed in Table 2.

3.3.3 Palmtree Workflow

Palmtree (Lenôtre 2016) is a library for the parallelisation of Monte Carlo methods where the challenge is the proper management of the random numbers. The workflow structure presented on the right of Figure 4 is composed of 2 parallel tasks. Each task is running 100 000 simulations with an accuracy of 0.0001 step. Its execution is CPU-intensive only.

Experimentation on this workflow showed us that assigning 4 cores, 2 GB of RAM and 10 GB of disk space to each task gives a good execution trade-off. Table 2 lists the given VM flavors for each execution mode. With the *Medium* execution mode, VMs with the *large* flavor are given to each task of the workflow.

3.4 Power Consumption Measurement

The three workflow applications in our benchmark have been executed on servers equipped with fine-grained watt-meters of the Lyon site of Grid'5000 in order to measure the energy consumed by their run in each execution mode.

At first we measured the energy consumption of the workflows by running one after the other. Thus the servers were almost never fully used

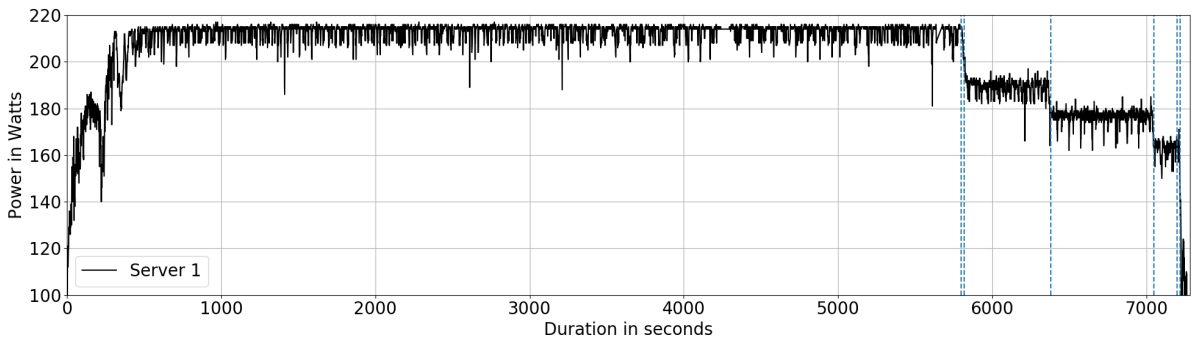

Figure 5 Power measurement realised with the Grid'5000 watt-meters of a server running the Blast workflow in *Little* mode and with 2 additional VMs in order to completely load the server.

such as during the execution of the second step of Montage (only 1 VM). The low resources usage results in an energy consumed with a major part of idle energy consumption. The latter is usually shared with concurrent VMs running on the same server but in this case the VM was running alone on its server. The energy attributed to this VM was incorrect because it didn't take into account concurrency and high resource usage. The energy logs obtained with this measurement setup resulted on a simulator showing servers consuming almost double the watts of the maximum power that a real server consumes.

In our second measurement we loaded as much as possible the servers. When used at its maximum capacity, the server has the energy consumed by the electrical components shared with all the VMs it is hosting. In order to have a fair energy sharing, the VMs had to be identical. Indeed, a large VM running important CPU-intensive calculations should have a bigger part of this consumption compared to a small VM doing low CPU-intensive tasks.

Thus our measurement is as follow: the VMs of a specific step are created on the servers with a distribution based on a *greedy* algorithm ; thus the last used server may not be fully loaded ; in this last server, we duplicate the VM (in other words, add a parallel task in the step) in order to make the server fully used so it is not possible to duplicate the VM another time.

With the same previously explained example, the second step of Montage only demands for 1 VM creation and depending on the selected execution mode, the flavor can be large, medium or small (see Table 2). On the servers presented in Section 2, a server can host a maximum of 3 large VMs or 6 medium VMs or 12 small VMs. So, with the *Medium* mode, the medium VM of the Montage PNG task can be duplicated 5 times and the energy consumed by a single VM is $\frac{1}{6}$ of the global energy measured on the server.

Blast has 4 parallel medium VMs when executed with the *Little* mode. They can be hosted on a single server which is not fully loaded after the VM creations. To completely load the server, 2 other identical VMs have to be instantiated. The power consumption measurement of this server, presented in Figure 5, corresponds to the execution of 6 VMs: 4 Blast parallel tasks + 2 duplicates. The maximum power consumption reached is 217W and we can see the power falling each time a VM terminates its execution until the last one where the power consumption falls to 100W (the server's idle power consumption).

Finally, power measurements were conducted on the same servers to obtain the power profile of switching on and off sequences. These measurements show that turning off a server takes about 6 seconds which is insignificant compared to the booting time (2.5 minutes) and to execution duration of the 3 scientific applications (from 6.75 minutes to 117.4 minutes depending on the application and configuration). This behavior is even exacerbated from the energy point of view because turning off a server does not produce significant power peaks as the booting process does. The power profile of the booting sequence is integrated within the simulator in order to

Table 3 Details of the execution of all the workflows on the three execution modes with the type and number of instances used, the required number of hosts, the execution time and the amount of energy consumed.

Montage	Runtime	Energy	Hosts	Instances
Big	1278 sec	104 193 J	1	$4 \times large$
Medium	2018 sec	101 818 J	1	$4 \times medium$
Little	3704 sec	99 514 J	1	$4 \times small$
Blast	Runtime	Energy	Hosts	Instances
Big	2213 sec	572 577 J	4	$4 \times xlarge$
Medium	3520 sec	516 383 J	2	$4 \times large$
Little	7043 sec	502 465 J	1	$4 \times medium$
Palmtree	Runtime	Energy	Hosts	Instances
Big	404 sec	59 802 J	2	$2 \times xlarge$
Medium	653 sec	49 821 J	1	$2 \times large$
Little	1253 sec	49 457 J	1	$2 \times medium$

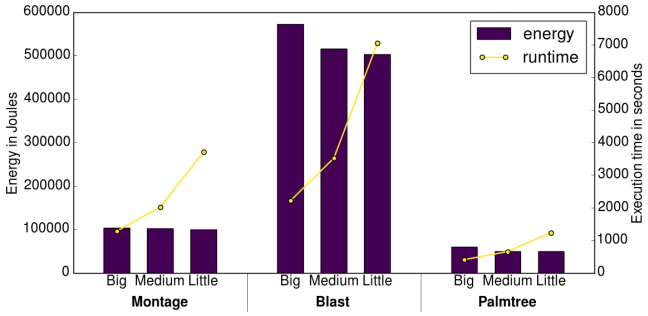

Figure 6 Energy consumption and execution time of each workflow in each execution mode.

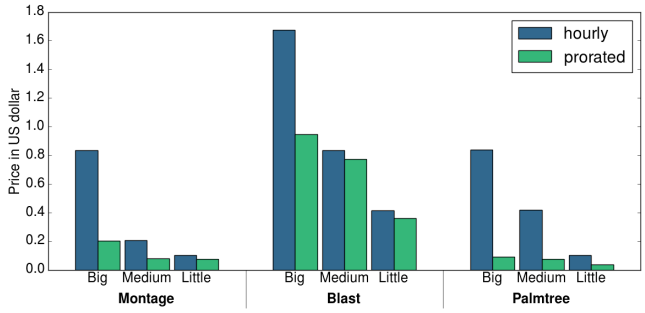

Figure 7 EC2 hourly pricing and the prorated pricing of each workflow in each execution mode.

account for the cost of switching on a server in terms of energy consumption and time duration.

3.5 Performance versus Cost Trade-off

Details on the execution of the workflows are listed in Table 3. It shows the maximum execution time, the total energy consumed in Joules, the number of servers required to host the VMs and the number and the type of VMs instantiated for each workflow in each execution mode. The number of servers required to run the workflows increases when the *Big* mode is selected which explains the energy consumption increasing. Thus, the smaller the VMs, the smaller the amount of Joules consumed by the run of the given workflow. On the other hand, the execution time increases by a factor of 3 and more when the *Little* mode is selected. A summary of the execution time versus the energy consumption of each workflow in each execution mode is given in Figure 6.

Figure 7 gives an idea about how much it would cost to run these workflows on the Amazon EC2 platform. On this platform users pay the access to their instances by hour even if the instances are not used a complete hour. The figure presents the EC2 hourly pricing but also the price if a prorated pricing were available. It shows the *Big* execution mode costs more than the *Medium* mode that also costs more than the *Little* mode.

4 Experimental Validation

Table 4 presents our simulation results. We simulate a full day and a cluster with 330 servers (minimum number of servers required to be able to respond to the demand in the highest demand case). The table at the top contains the results for a simulation with the workload A and the second table is for the workload B. Each row presents the results for a profile distribution following the percentages

Table 4 The simulation results give the energy consumption of a whole cluster used during 24h from 2AM to 2AM the next day and the maximum number of hosts used with various profiles of job execution modes. The table at the top is with workload A and the second one is with workload B.

Big	Medium	Little	Energy (KWh)	Std dev energy	Hosts used	Std dev hosts	Energy saved
100	0	0	632.489	16.277	282	7.909	0.00 %
100	0	0	292.941	3.690	292	16.806	53.68 %
0	100	0	234.122	4.882	168	6.363	62.98 %
0	0	100	231.921	3.840	143	3.187	63.33 %
80	0	20	273.205	6.021	236	16.117	56.80 %
60	0	40	269.969	3.497	208	11.071	57.32 %
40	0	60	258.138	3.980	190	14.935	59.19 %
20	0	80	246.996	3.701	170	6.610	60.95 %
20	20	60	246.590	5.482	167	9.843	61.01 %
20	60	20	242.464	4.013	171	9.243	61.67 %

Big	Medium	Little	Energy (KWh)	Std dev energy	Hosts used	Std dev hosts	Energy saved
100	0	0	474.613	28.942	163	15.492	0.00 %
100	0	0	228.022	4.537	167	10.595	51.96 %
0	100	0	184.073	3.139	89	3.847	61.22 %
0	0	100	181.031	2.833	61	2.377	61.86 %
80	0	20	214.511	3.338	136	11.020	54.80 %
60	0	40	206.305	2.171	115	5.653	56.53 %
40	0	60	203.497	4.558	105	8.821	57.12 %
20	0	80	193.719	2.831	81	3.976	59.18 %
20	20	60	193.185	3.739	82	5.665	59.30 %
20	60	20	193.356	3.644	90	6.651	59.26 %

given in the 3 first columns. All results are the average of 10 simulations and contain the energy consumption in KWh of the whole cluster, the maximum number of hosts required to execute the workload and the standard deviations.

The grey row of each table corresponds to a simulation on a usual cloud infrastructure without any energy optimisation. The unused servers are not powered down and all users select the *Big* execution mode because it reflects a common behaviour when users want results as soon as possible. The last column in both tables is the percent of energy saved compared with the scenario of the first row. A scenario with a 50% energy saving means its execution consumes half of the execution with the scenario of the first row.

As we can see in the simulation results, a cloud system that turns off unused servers consumes less than 50% of usual cloud systems. The simulations in which the most energy has been saved are when 100% of the users selected the *Medium* and the *Little* execution modes. In these two cases, in both studied workloads, the energy saving varies from 61.22% to 63.33% in comparison with the consumption of the first row scenario. The simulation costing the most in terms of energy is the scenario where 100% of the users select the *Big* execution mode, corresponding to an energy saving of 53.68% in workload A and 51.96% in workload B. It shows we can save important amounts of energy by avoiding the *Big* mode. Informing users about how much more their application consumes compared to another mode may encourage them to select a more energy-efficient execution mode and thus, motivates the implementation of an incentive mechanism. It also shows that the gap between the *Little* and *Medium* modes is very small. Selecting the *Little* mode is not always the best performance and energy trade-off. Indeed, the energy may be very similar between the two modes and the execution time much longer in the *Little* mode compared with the *Medium* mode.

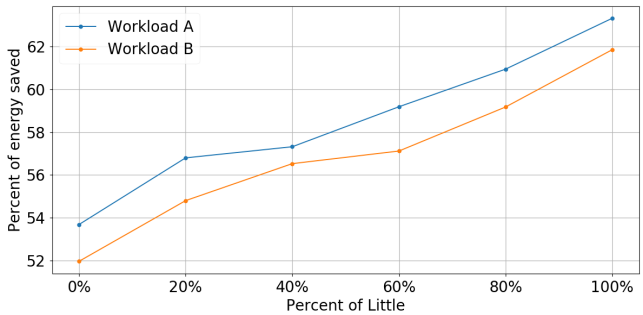

Figure 8 Energy savings increase when the percent of applications running with the *Little* mode increases (*Medium* mode kept at 0%).

When 100% of the workload is using the *Little* mode, we can see that workload A uses a maximum of 143 servers and only 61 servers on workload B out of the 330 servers available in the simulated cluster. Fewer servers turned on means a lower energy cost on the cooling system of the data centre. It also means the cloud provider could buy less servers and thus, fewer ones to recycle after their lifespan. From another point of view, the low server utilisation means this system can handle a higher number of users if most of them continue to use the *Little* mode.

In a realistic situation, users won't be 100% using the same execution mode but rather a few percent in each of them. For table dimension reasons, Table 4 does not contain all possible distribution configurations but still reveals a link between the user profiles and the energy consumed. If we sort the table by descending order of *Big* users, we can see the energy consumption and the number of used hosts decreasing. This behaviour can be seen in Figure 8 where the *Medium* mode is fixed to 0% and the amount of applications running with the *Little* mode increases from 0 to 100% by steps of 20%. In both workloads the energy savings increase when the percentage of *Big* mode decreases and the percentage of *Little* mode increases. It shows that even with a small amount of applications using the *Little* execution mode, we can achieve energy savings.

However, as shown by the two last rows in each table, for a fixed amount of *Big* users, the percentage variation of *Medium* and *Little* users has a small impact on the energy consumption. Thus, the system does not save much more energy with more *Little* users than *Medium* but globally allows the system to run the workload on fewer servers.

5 Discussion

A number of optimisations of the green cloud system we present in this paper can be made. We decided to give to the user 3 execution modes following the BML profiles presented by Villebonnet et al. (2015). Maybe a number of 3 modes is not enough and giving a larger number of choices to the user may allow a finer-grained control on the performance and energy consumed by the execution of an application.

A routine turns off the unused servers each time a VM instance is deleted. So the servers are turned off directly when they do not host any VMs. It may be better to wait for a short amount of time in order to avoid to turn on a just powered down server because it received a VM creation request few seconds after the turning off action. This optimisation may be used when calculating the trade-off between the energy to turn on and off a server and the energy to keep it powered on along a prediction algorithm to estimate the arrival of the next VM creation request. Switching on a server may take several minutes (2.5 minutes from our measurements), and consequently, the execution of a given VM may be delayed. Yet, aggressive shut down techniques have been proved energy efficient and may offer an acceptable trade-off between energy savings and platform reactivity in the context of scientific testbeds Rais et al. (2016).

Based on previous execution logs, we could inform the user about how much her application consumed in the past and how much she can save in energy by selecting a more energy-efficient execution mode. This incentive system should also take into consideration the gain in energy versus the execution time between the *Medium* and *Little* modes to avoid a longer run where no additional energy is saved.

The consolidation mechanism implemented in our system only does spatial optimisation. The

VMs are created in servers in order to have a maximum number of servers fully used in terms of hardware resources. Including the temporal consolidation mechanism presented by Orgerie et al. (2008) may allow to save even more energy but would require users to accept to delay the start time of their applications.

Finally, the next step of this work would be for the cloud provider to propose different VM sizes that would be dynamically calculated according to the current job submission rate and the workload characteristics in order to achieve a good energy-performance trade-off. This autonomic VM size adjustment depending on the workload could benefit from learning techniques to predict the future job arrivals and from classification techniques to profile typical user applications and to determine their Medium size of VM.

6 Related Work

The energy-efficient cloud system we present in this paper uses consolidation as a lever to save energy but other solutions exist. Martinovic & Krpic (2011) list the existing green solutions for HPC and cloud computing infrastructures from the hardware level to the software level and present new energy metrics such as the PUE (Power Usage Effectiveness) which gives a value of the energy-efficiency of a system. To name a few existing techniques: Vary-On/Vary-Off controls the servers' state in order to turn off a maximum of servers and thus reduce the consumption at the cluster's level, Dynamic Voltage and Frequency Scaling (DVFS) reduces the energy consumption at the CPU level. The latter is used in the study of De Matteis & Mencagli (2016) alongside with scaling strategies in order to achieve energy savings in the context of elastic data stream processing. Their ability to change configuration on-the-fly in response to the fluctuation of the workload is also provided by virtualisation. The elasticity offered by virtualisation allows to optimise the energy consumption from the software level. A complete list of the techniques is presented by Orgerie et al. (2014).

In the study of S.K. Tesfatsion et al. (2014), they propose a cloud mechanism including the user to save in energy consumption. It combines horizontal and vertical scaling and dynamic frequency scaling where the user has a control on the default configuration. However the parameters are complex to tune for a lambda user and thus the system is not user-friendly. A different system including the user to save energy is presented in the paper of Orgerie et al. (2008). It uses temporal consolidation of jobs in large scale distributed systems. The user controls how green will be the execution of her job. This control has an impact on how elastic can be the start time of the job. A job allowing to be delayed allows a better temporal consolidation. As a similar user-oriented temporal consolidation system, there is EARH, which is a scheduling algorithm proposed by Zhu et al. (2013) doing automatic consolidation to save energy in cloud data centres. It uses the elasticity offered by cloud systems and the applications' deadline which is controlled by the users in order to optimise the infrastructure utilisation and thus decreases the global energy consumption. Unlike the spatial consolidation control we give to the users, these studies provide a different control based on temporal consolidation.

7 Conclusion and future work

In this paper we present a simulation-based evaluation on how much an energy-aware cloud system could save in energy consumed by involving users in the energy conservation. In this system users can select an execution mode for running their applications. An execution mode controls the size of the application's VMs. The higher the mode, the larger the VMs and vice versa. A consolidation algorithm packs the VMs into a minimum number of servers in order to have a maximum of servers powered down. The smaller the VMs, the better the consolidation and the lower the global energy consumption of the infrastructure.

We simulated two typical daily uses of a data centre running 3 real scientific applications

and varied the amount of applications in each execution mode. The simulation results show a saving of energy of more than 50% whatever the selected mode compared with cloud infrastructures where the servers are not turned off when not used. Scenarios where the most energy is saved are when 100% of users select the *Medium* and *Little* execution modes. However, cloud users tend to over-commit their job reservations (Ghosh & Naik 2012) and they end up selecting the *Big* mode while the *Medium* mode is sufficient. The simulator results show the importance of reducing the amount of users using the *Big* mode and also that selecting the *Little* mode is not always the best practice because energy savings may be low and the application execution time much higher than with the *Medium* mode.

As a future work, we would like to inform the user about how much her previous executions consumed and how much she can save by selecting a more energy-efficient execution mode. More generally, we want to implement a smart incentive system where users are informed about their energy impact on the infrastructure they are using and to motivate them to reduce this energy consumption with a low reduction of their applications' performance.

Acknowledgment

Experiments presented in this paper were carried out using the Grid'5000 experimental test-bed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see `https://www.grid5000.fr`).

References

Amazon EC2 (2017).
URL: *https://aws.amazon.com/ec2/*

Andrae, A. S. & Edler, T. (2015), 'On Global Electricity Usage of Communication Technology: Trends to 2030', *Challenges* **6**(1), 117–157.

Basic Local Alignment Search Tool (Blast) (2017).
URL: *https://blast.ncbi.nlm.nih.gov/Blast.cgi*

De Assuncao, M. D., Gelas, J.-P., Lefevre, L. & Orgerie, A.-C. (2012), The Green Grid5000: Instrumenting and using a Grid with energy sensors, *in* 'Remote Instrumentation for eScience and Related Aspects', Springer, pp. 25–42.

De Matteis, T. & Mencagli, G. (2016), Keep Calm and React with Foresight: Strategies for Low-Latency and Energy-Efficient Elastic Data Stream Processing, *in* 'Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming', ACM, p. 13.

Feitelson, D. G., Tsafrir, D. & Krakov, D. (2014), 'Experience with using the Parallel Workloads Archive', *Journal of Parallel and Distributed Computing* **74**(10), 2967 – 2982.
URL: *http://www.sciencedirect.com/science/article/pii/S0743731514001154*

Ghosh, R. & Naik, V. K. (2012), Biting Off Safely More Than You Can Chew: Predictive Analytics for Resource Over-Commit in IaaS Cloud, *in* 'IEEE 5th International Conference on Cloud Computing (CLOUD)', pp. 25–32.

Gupta, A., Kale, L. V., Gioachin, F., March, V., Suen, C. H., Lee, B.-S., Faraboschi, P., Kaufmann, R. & Milojicic, D. (2013), The Who, What, Why and How of High Performance Computing Applications in the Cloud, *in* 'IEEE International Conference on Cloud Computing Technology and Science (CloudCom)', p. 12.

Gupta, A. & Milojicic, D. (2011), Evaluation of HPC Applications on Cloud, *in* 'Open Cirrus Summit (OCS)', pp. 22–26.

Guyon, D., Orgerie, A.-C. & Morin, C. (2015), Energy-efficient User-oriented Cloud Elasticity for Data-driven Applications, *in* 'IEEE International Conference on Green Computing and Communications (GreenCom)', pp. 376–383.

insideHPC (2012), 'What is high performance computing?'.
URL: *http://insidehpc.com/hpc-basic-training/what-is-hpc/*

Kivity, A., Kamay, Y., Laor, D., Lublin, U. & Liguori, A. (2007), kvm: the Linux Virtual Machine Monitor, *in* 'Proceedings of the Linux symposium', Vol. 1, pp. 225–230.

Lenôtre, L. (2016), A Strategy for Parallel Implementations of Stochastic Lagrangian Simulation, *in* 'Monte Carlo and Quasi-Monte Carlo Methods', Springer, pp. 507–520.

Martinovic, G. & Krpic, Z. (2011), Towards Green HPC Blueprints, *in* 'Proceedings of the Second International Conference on Cloud Computing, GRIDs, and Virtualisation', pp. 113–118.

Montage (2017).
URL: *http://montage.ipac.caltech.edu/*

Orgerie, A.-C., Assuncao, M. D. d. & Lefevre, L. (2014), 'A Survey on Techniques for Improving the Energy Efficiency of Large-scale Distributed Systems', *ACM Comput. Surv.* **46**, 47:1–47:31.

Orgerie, A.-C., Lefevre, L. & Gelas, J.-P. (2008), 'Save Watts in Your Grid: Green Strategies for Energy-Aware Framework in Large Scale Distributed Systems', *International Conference on Parallel and Distributed Systems (ICPADS)* pp. 171–178.

Rais, I., Orgerie, A.-C. & Quinson, M. (2016), Impact of Shutdown Techniques for Energy-Efficient Cloud Data Centers, *in* 'ICA3PP: International Conference on Algorithms and Architectures for Parallel Processing', Granada, Spain, pp. 203–210.

S.K. Tesfatsion, E. Wadbro & J. Tordsson (2014), 'A combined frequency scaling and application elasticity approach for energy-efficient cloud computing', *Sustainable Computing: Informatics and Systems* **4**(4).

The MetaCentrum 2 log (2013).
URL: *http://www.cs.huji.ac.il/labs/parallel/workload/l_metacentrum2/index.html*

Villebonnet, V., Da Costa, G., Lefevre, L., Pierson, J.-M. & Stolf, P. (2015), 'Big, Medium, Little: Reaching Energy Proportionality with Heterogeneous Computing Scheduler', *Parallel Processing Letters* **25**.

Yue, M. (1991), 'A simple proof of the inequality FFD (L) 11/9 OPT (L) + 1, L for the FFD bin-packing algorithm', *Acta Mathematicae Applicatae Sinica* **7**(4), 321–331.

Zhu, X., Chen, H., Laurence T. Yang & Yin, S. (2013), Energy-Aware Rolling-Horizon Scheduling for Real-Time Tasks in Virtualized Cloud Data Centers, *in* 'IEEE International Conference on High Performance Computing and Communications (HPCC)', pp. 1119–1126.