ITEA
INFORMATION TECHNOLOGY
FOR EUROPEAN ADVANCEMENT

(ITEA2 09011)

Optimize HPC Applications on Heterogeneous Architectures

Deliverable: D5-2.3.5 (L2.3.5)

Prototype of an adaptive scheduling strategy to allocate resources based on power consumption

Version: V1.2

Date: January 2015

Authors: Bull SAS / Yiannis Georgiou

Status: Final

Visibility: Public

HISTORY

Document version #	Date	Remarks	Author
V1.0	June 2014		Y.Georgiou
V1.1	January 2015	After internal review	Y.Georgiou
V1.2	January 2015	After CEA Review	Y.Georgiou

TABLE OF CONTENTS

1. Executive Summary

The last decades have been characterized by an ever growing requirement in terms of computing and storage resources. This tendency has recently put the pressure on the ability to efficiently manage the power required to operate the huge amount of electrical components associated with state-of-the-art computing and data centers.

The power consumption of a supercomputer needs to be adjusted based on varying power budget or electricity availabilities. As a consequence, Resource and Job Management Systems have to be adequately adapted in order to efficiently schedule jobs with optimized performance while limiting power usage whenever needed.

This deliverable introduces a new power consumption adaptive scheduling strategy that provides the capability to autonomously adapt the executed workload to the available or planned power budget. This originality of this approach relies on a combination of DVFS (Dynamic Voltage and Frequency Scaling) and node shutdown techniques.

It is implemented into the widely used resource and job management system SLURM. Finally, it is validated through large scale emulations using workload traces of the petaflopic HPC cluster Curie and models of actual benchmarks executions.

2. Introduction

Energy consumption has become one of the most crucial issues in the evolution of High Performance Computing systems. The increase in computation performance of such platforms has come with an even greater increase in energy consumption, turning energy an undisputed barrier towards the exascale challenge.

The power demand of such HPC systems has made electricity one of the largest cost component for the lifetime of these systems and it is important, almost critical, to improve electrical system utilization and operation profiles. Since the operational cost of a computing center for HPC systems is directly related to the energy consumption, such centers have additional motivations to regulate the energy usage. However, controlling the energy for a certain duration demands novel approaches to adjust the instantaneous power on a daily basis.

We are interested here in proposing a mechanism that limits the instantaneous power consumption of an HPC platform and adjust it to a varying power budget or electricity availability. We will also show that reducing the power consumption does not always lead to reducing the global consumed energy. Research efforts upon all different abstraction layers of Computer Science, from hardware, to middleware up to applications, strive to improve energy conservation and provide efficient power management. The advances at the hardware layer need to be followed by evolutions on systems software and middleware in order to provide efficient results.

Various techniques such as speed scaling (DVFS) and node shutdown in case of node idleness have been broadly used for energy reductions (see references [1], [2] at the end of this document).

Nevertheless, the regulation of power needs to pass from a centralized system middleware that is able to monitor power consumption and adjust it through coordinated actions across the whole infrastructure. The most adequate software for this type of coordination is the Resource and Job Management System (RJMS) which plays a crucial role in the HPC software stack since it has knowledge of both the hardware components state and information, along with details upon the users workloads and the executed applications.

We propose a power capping mechanism implemented upon the widely used resource and job management system SLURM based on a combination of offline and online job scheduling approaches and making use of both DVFS and node shutdown mechanisms.

In particular, this work shows that in some cases, the best solution is to mix both techniques. As a consequence, the RJMS has to be adequately adapted in order to efficiently schedule the jobs with optimized performance while limiting power usage whenever needed. The main contribution is to introduce a new power consumption adaptive scheduling strategy that provides the capability to autonomously adapt the executed workload to the available or planned power budget. It is validated with large scale emulations using workload traces of the HPC cluster CURIE and models of actual benchmarks executions. These features have been integrated as a plug-in within the official source code of SLURM under GPL license.

The content of the paper is as follows. We start by describing the target HPC batch scheduler SLURM followed by an overview of the most relevant approaches used so far for reducing the energy consumption and power capping in Section 3. Section 4 presents the new approach for solving the problem and its analysis. The scheduling algorithm is presented in Section 4.2. We present the implementation of the proposed algorithm in Section 5. We report the results of experiments in Section 5.3. Finally, some perspectives are discussed in Section 6.

3. Background

3.1 Target HPC batch scheduler: SLURM

SLURM [3] is a widely used open-source resource manager and batch scheduler. It enables to efficiently manage large workloads on large systems, among the largest currently in production. In a nutshell, SLURM is designed as a client-server distributed application: a centralized server daemon slurmctld, also known as the controller, communicates with a client daemon slurmd running on each computing node.
Users can request the controller for resources to execute interactive or batch applications, referred as jobs. The controller dispatches the jobs on the available resources, whether full nodes or partial nodes, according to a configurable set of rules.
A power saving logic enables to define a set of nodes, scripts, timeouts and ratios, enabling to perform automated suspend/resume operations on idle nodes not yet necessary. In spite of being interesting to reduce the total power bill of a system, this mechanism does not enable to enforce a particular powercap. Furthermore, on systems where switching-off nodes on demand is not possible or simply not desired, this logic has a limited benefit in terms of real power savings.

SLURM offers a large variety of features but provides only limited support of energy saving mechanisms within a cluster. The power awareness of SLURM has recently been enhanced by introducing the capability to regularly capture the instantaneous consumed power of nodes [4].
Coupled with the introduction of a speed scaling logic, enabling to modify the maximum frequencies of the cores involved during the execution, this new feature helps to identify the behavior of applications in terms of power consumption when varying the frequency. Thus, the users can experiment various frequency values to evaluate the behavior of their jobs and optimize their energy efficiency. In the report of June 2013, SLURM is used in half of the 10 most powerful clusters registered in the Top500 and about two thirds of computers of the complete list. We present in the following subsections a brief overview of the different techniques used for controlling the power.

3.2 Controlling the power

Dynamic Voltage and Frequency Scaling (DVFS in short) is the most popular mechanism used so far for controlling the power in computing systems and as a consequence, reduce the energy. There exist a lot of works oriented toward theoretical results on speed-scaling (for instance continuous speeds, ...). We are targeting here more realistic issues. The first series of papers intended to determine the right frequency. Energy-centric DVFS controlling method was proposed in [1] for single CPU multi-core platforms. The idea was to monitor the traffic of data from the cores to the memory and to update the DVFS accordingly (i.e. reduce it if the traffic is high or expand if it is low). This was extended in [5] for more general platforms.

Schöne and Hackenberg ([6]) used register measurements for determining the frequency. Kimura et al.[7] provided also a new mode of control of DVFS through a code-instrumented DVFS control. Then,

Gandhi et al.[8] considered a mechanism that switch between the maximum DVFS to the idle state, in order to minimize the energy consumption.

DVFS has also been studied to predict its impact on the whole system. Rountree et al. [9] developed a performance prediction model outperforming previous models. Etinski et al. [10] studied how to improve the trade-off energy versus completion time on applications. Freeh et al. [11] provided a huge number of experiments for measuring the Energy versus Time.

An interesting feature was studied in the case where a node is removed from the system (moldable jobs). Another complementary mechanism consists in switching off some nodes ('shutdown'). Lawson et al. [2] proposed an opportunistic shutdown mechanism, which switches off a node after a significant idle period. Aikema et al.[12] studied the energy from the view point of a cost function. Here, a node which becomes idle is considered as a zero-cost in term of energy.

Under the assumption of under-loaded cluster, Hikita et al.[13] presented a batch scheduler that minimizes the number of active nodes while keeping the same performances.
In [14], Demaine et al. took into account both the cost of energy and of switching (off/on) the processors. They proposed a theoretical algorithm that minimizes the number of such switches.
Despite the two first mechanisms, other possibilities have been studied including network frequency scaling [15] or temperature-aware scheduling [16]. An incentive mechanism for reducing energy has also been proposed in [17].

3.3 Powercapping

In the following of this paper, we are focusing on powercap as the main topic.
Some papers are considering powercapping at the node level, for instance [18] used a new feature available in Intel processors to achieve a local powercap and [19] packed threads together in order to tune the DVFS. Fan et al. defined in [20] a methodology to reduce the global cost of data-centers by buying more processors and capping their power consumption.

In cloud computing, Geronimo et al. proposed a virtual machine manager that can use DVFS, update the virtual machine resources, migrate them and switch-off opportunistically some processors [21].

Powercap has been studied by Etinski et al. in a series of papers ([22],[23],[24]) where DVFS is the only mechanism used to achieve powercapping while keeping good performances. We consider in this work a more sophisticated mechanism including also shutdown.

4. Scheduling under power control

4.1 Preliminaries on scheduling features

From a high level point of view, scheduling in a Resource and Job Management System can be decomposed into two successive phases:

- First, the jobs should be selected after prioritization among the group of pending jobs,
- Then, the resources should be selected upon which the job will be dispatched.

In more detail, the first phase may involve various mechanisms to select the next job to be treated. For instance, the usual backfilling [25] may be enriched with multifactor priorities such as job age and job size or even more sophisticated features like fair-sharing and preemption.

The second phase is related to the actual allocation of resources according to their characteristics such as internal node topology, network topology, usage of accelerators. The proposed powercapping algorithm takes place during this second phase of scheduling.

One of the main building blocks of this algorithm relies on the fact that power is treated as a new type of resources characteristics. According to its state (PowerDown, Idle, Busy, etc.), the resource will consume a different amount of power. At any moment, the RJMS keeping the state of each resource internally can deduce the power consumption of the whole cluster.

The characteristic of power can be related to any different component of the cluster but for the sake of clarity, we consider here the power of a whole node. The calculation of the power consumption of the cluster is simply obtained by summing up the power consumptions of each node. For instance, the nodes that are executing jobs will be counted with the maximum power consumption (except in cases of DVFS usage); the nodes that are kept idle will be counted with the minimum power consumption and those that have been switched-off are counted with no power consumption (it could be non-zero in case where the BMC (Baseboard Management Controller) is still on).

The algorithm goes one step further by considering the setting of the different CPU frequencies as different power states. Hence, the power consumption of each node will change depending on the CPU frequency at which the job is running.

The power values related to the state of each node can either be measured or be given by the constructor (this information can be configured and kept internally into the RJMS).

4.2 Scheduling algorithm for powercapping

The powercapping algorithm that we propose is composed of two successive parts:

A first offline part where the decisions for power management are taken in advance (to better prepare the future actions) followed by an online part where the power distribution and management take place directly.

The algorithm is activated as soon as a powercap threshold is provided. This powercap value can be either set for 'now' (i.e. the moment when the command is launched) with no time restriction/limitation or as a reservation for a certain time window in the future.

When a new job arrives during the allocation phase, the algorithm examines if there is any powercap limit for the time being or if the job may overlap with any future reservation of power at some point in the future. If any of these cases holds, the power consumption of the cluster is computed by considering as busy the nodes that the job will use. Then, the different values of the a-posteriori power consumption of the cluster are examined by measuring the power with all the different CPU frequencies where the allocated nodes may run.

If there is no CPU frequency to allow the future power consumption of the cluster to be less than the power budget that has been set then the job remains pending. In the opposite case, the job is executed at the maximum CPU frequency that allows the job to be executed keeping the cluster in the power budget. One of the important parts of the algorithm is the selection of the CPU frequency. Selecting a value close to the maximum will make the power consumption of the cluster to increase faster, getting obliged to keep some nodes idle, producing starvation of following jobs and dropping the utilization of the system.

Considering that jobs may run at a lower CPU frequency (which means that nodes will consume less power) gives us extra flexibility for scheduling more jobs. However, since only one job is treated at a time and we cannot know how many jobs will follow, we need to select the best possible value of CPU frequency whenever the powercapping is activated. The optimal CPU frequency is the maximum allowed frequency that all idle nodes could run such that the total power consumption of the cluster remains below the powercap value. Since the number of idle nodes may change, the optimal CPU frequency may also change from one job to another.

If the powercap value is set for 'now' then there could be a problematic scenario where the cluster is currently above the powercap. In this case, by default, no extreme actions are taken with the running jobs. This means that no additional jobs will be scheduled and the scheduler will wait until some jobs are completed to eventually have the power consumption of the cluster dropped to a value lower than the powercap. However, we argue that the above default way may not be accepted by some sites that may want to have extreme actions when the powercap value is set. In this case, the necessary number of jobs will be killed until the power consumption of the cluster drops instantaneously. The adequate energy saving mechanism is chosen in the offline step. System administrators can force one or another mechanism. We defined three policies, namely idle only, switch-off and DVFS.

Idle only means that if a power cut is needed, the system can only keep nodes in an idle state in order to save energy. Switch-off means that the system will have the possibility to switch-off nodes and keep

others in an idle state if needed. DVFS policy means that the system will have the possibility to oblige jobs to be executed at lower CPU frequencies.

4.3 Implementation upon SLURM

The above scheduling algorithm has been implemented upon the open-source resource and job management system SLURM [3]. To achieve the targeted goal of dynamic powercapping, a new parameter, PowerCap is added to the controller's set of states. It represents the allowed power budget in watts of the cluster. To compute the maximum power amount required to operate a cluster, new parameters are associated to the compute nodes definition to provide the different amounts of watts required to operate them at different states.

Thus, the IdleWatts, MaxWatts, DownWatts will respectively correspond to the amounts of watts required to operate a node in idle, fully used and down states. The down state corresponds to the state the controller uses to characterize a node not being currently accessible within SLURM. Furthermore, other parameters such as CpuFreqXWatts may be used to characterize a node that its CPUs runs at a specific X Frequency. For example, CpuFreq12Watts and CpuFreq27Watts will respectively correspond to the amounts of watts that the node will consume when all its CPUs operate with 1.2 and 2.7 GHz of Frequency.

While computing the instantaneous maximum amount of power of the cluster, the controller will use the known states of the nodes in order to sum up the individual maximum amounts of watts and produce a single power value for the whole cluster. The offline part of the scheduling algorithm is implemented as particular types of reservations in SLURM. SLURM reservation characteristics are extended by a new Watts parameter in order to specify a particular amount of power reserved for a particular time slot: this enables to define power reservations corresponding to the time windows that particular power cuts may be needed.

The online part is implemented in the central part of SLURM scheduling mechanism upon the controller. When evaluating the impact of the start of a pending job, the controller will temporarily alter the states of the candidate nodes, compute the resultant threshold and compare it with the defined and planned caps. By planned caps, we mean the values corresponding to the current cap of the cluster minus the different power reservations encountered during the estimated execution time of the job. In case of power budget overflow, the evaluated job will stay pending and the next one of the list will be considered instead. Thus, only the first n jobs of the pending list enabling to respect the budget will be executed. As soon as jobs finish, associated nodes may return to the idle state, resulting in a new power capacity available for other pending jobs.

Note that the current prototype does not make any difference in power requirements whether nodes are fully or partially used. The evaluation of new jobs only filling partially used nodes will always pass the powercapping criteria as no extra power will be required. As a result, the scheduler will tend to fill the compute nodes up to the targeted power budget.

5. Experimentations - Adapting powercapping logic for a large scale cluster

The activation of the adaptive power control of SLURM for a certain infrastructure requires an initial configuration where the maximum power consumptions of each implicated components, along with other important parameters are defined.
In this section we present the study made for the adaptation of SLURM powercapping logic for Curie supercomputer.

5.1 Details on Curie

Curie[1] is operated by the French Atomic Energy Research Center CEA[2]. It is the first French Tier-0 system opened to scientists through the participation into the PRACE[3] research infrastructure. Curie was ranked 11th among the 500 most powerful supercomputers on November's 2012 Top500 list[4]. Configuration details and workload traces of Curie have been extracted at some points of its early lifetime and are used in this study to specify hardware characteristics and evaluate the behavior of the proposed mechanisms. We have seen previously that the architecture of an HPC cluster plays an important role when considering which nodes to switch-off in order to maximize the 'power bonus'.
In Curie, we distinguish 4 hierarchical levels that can be switched-off. The following table 1 gives the power consumption of each level.

- The first one is the node level. A node is composed of 2 sockets and each socket contains 8 processors. When a node is powered-off the BMC is kept active so that the node can be powered back on through the network. This explains the consumption of 14W in the following table when a node is down.The power of a node depends on its state (on/off) but also on the DVFS of the processors. In this case we only consider the two node states of down and maximum utilization. The gained accumulated power if we switch off one single node would then be the difference between the down and max states which is 344W.
- The second level is the chassis level. Each Chassis contains 18 nodes and the power bonus is composed by global cooling fans, network switches installations such as Ethernet and Infiniband switches, optical cables, network ports. These hardware components consume an extra amount of power of 248 Watts with a power bonus of 500 Watts as we see in the following table 1.
- Rack level is the third one. It is composed of 5 chassis, which contain fans and a cold door related to the liquid cooling equipment. The power consumption of these components is 900 Watts and the bonus when switching off a complete rack will be 3400 Watts.
- Finally, the last level is the whole cluster, which is composed of 56 racks.

1 http://www-hpc.cea.fr/en/complexe/tgcc-curie.htm
2 http://www.cea.fr/english-portal
3 http://www.prace-project.eu/
4 http://www.top500.org

The power bonus values of the table 1 imply that if we switch off a complete chassis this will allow us to completely use at least 1 extra node where-as if we switch-off a complete rack this will allow us to use at least 9 extra nodes.

Level	Power Consumption	Power Bonus	Accumulated Power
Node (down)	14 W	-	-
Node (max)	358 W	-	344 W
Chassis (18 nodes)	248 W	248+18*14=500 W	344*18+500=6692 W
Rack (5 chassis)	900 W	900+500*5=3400 W	6692*5+900=34360 W

Table 1: Power consumption and the possible saved watts when various levels of the cluster are switched-off.

For example, if our model needs a power reduction of 6600 Watts and we consider that when a node is powered-on it will be used with the maximum consumption of 358 Watts then we need to switch-off 20 single nodes to reduce the power for 6880 Watts (=344*20). However if we make sure that we power off the 18 nodes of a complete chassis this will allow us to make use of the bonus and reduce the power for 6692 Watts which is even more than what we actually need. This allows us to use 2 extra nodes.

In order to take advantage of the power bonus and keep more nodes powered-on, we need to prepare an efficient grouping of nodes to switch-off. Hence that is why the choice of which nodes will be switched off takes place during the offline part of the algorithm.

5.2 Control and Measure Power

The power consumption of the different states of a node may be given by the constructor or calculated through experimentations. In our case we perform experiments of various known benchmarks using the exact same models of computing nodes and hardware components that are used on Curie.
We have run 3 different widely used benchmarks and 1 application to measure the characteristics of Curie cluster. We have chosen a first benchmark to stress all computing resources (Linpack [27]), a second one targeting memory (Stream [28]), one focused on network (IMB [29]) and the last one is a widely used application for molecular dynamics simulation (GROMACS [30]).
Measurements have been done through the IPMI[5] interface of SLURM power profiling mechanisms which have been shown to provide precise results for the consumption at the node level [4].DVFS is a way to obtain a trade-off between power and completion time. It has been widely studied in the literature (see Section 4). Figure 1 gives the evolution of the completion time and the maximum watts consumed at different DVFS values.

Illustration 1: Maximum Power - Execution Time Tradeoffs for Linpack, Stream, IMB and Gromacs benchmarks at different CPU frequencies

5

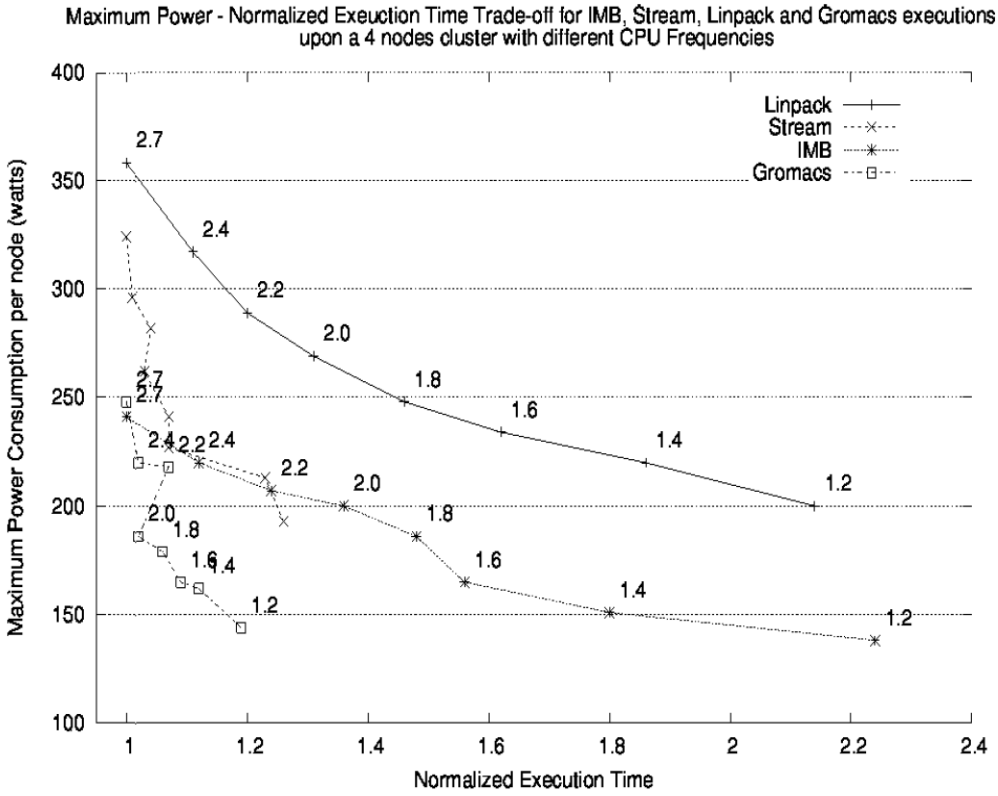


Table 2 presents the maximum power consumption observed on a node for each DVFS value based upon the results of all 4 applications. We have also added the observed power consumption of switched-off and idle states. These measurements set the maximum Watts that a node can consume. There is huge gap between switched-off and idle nodes. Both of them do not produce computational power but a switched-off node consumes one order of magnitude less power.

Node state	Maximum power consumption
Switch-off	14 W
Idle	117 W
DVFS 1.2 GHz	193 W
DVFS 1.4 GHz	213 W
DVFS 1.6 GHz	234 W
DVFS 1.8 GHz	248 W
DVFS 2.0 GHz	269 W
DVFS 2.2 GHz	289 W
DVFS 2.4 GHz	317 W
DVFS 2.7 GHz	358 W

Table 2: Table of the maximum power consumption of a node in different states.

6. Experimental Evaluations

In order to assess the results of our new scheduling algorithms, validate the implementation choices and evaluate the performance of each policy we need to test SLURM as one complete system through real-scale experimentations.
Our choices for experimental evaluations have been to:

- Use the real workload trace of Curie supercomputer for approximating real production executions,
- Take into account the real power consumption data of Curie supercomputer as discussed in Section 5
- Make use of an emulation technique to study SLURM as a complete system and allow the reproduction of Curie real-scale experiments by using only a small fraction of physical machines.

6.1 Platform and Test bed

The experiments have been performed upon the Nova2 platform which is an internal BULL cluster dedicated to experiments. The nodes used for our experiments are composed by Intel Sandy Bridge processors with 65 GB of Memory and Infiniband network.
In order to enable Curie real-scale experiments of SLURM we make use of an internal SLURM emulation technique called 'multiple-slurmd;. This technique allows the execution of multiple compute daemons slurmd upon the same physical machine using the same IP address but different communication ports.

The central controller will have to deal with different slurmd daemons and as far as the controller's internal mechanisms are concerned, they will run as if slurmd were deployed on real physical machines. The only drawback is that since various slurmd deamons are sharing the hardware components of the nodes we cannot execute real applications, thus only simple sleep jobs are used in the experiments. Nevertheless, this emulation technique allows us to make reproducible experiments with SLURM internal functions of the controller in large scale regardless the real number of physical machines.
In our context, we deploy 5040 nodes of Curie with only 16 physical nodes of our experimental platform Nova2.

6.2 Methodology

We propose to replay a time interval extracted from a real workload trace of Curie in 2012[6].In more detail, we select an interval of 5 hours with high utilization, big number of jobs in the queue, short inter-arrival time and with jobs that are representative of the whole workload. As already mentioned in

6

previous section, the users reserve in average 12670 times more time than need for the execution of their jobs.

As we are only interested in the RJMS internal decisions, hence the jobs are replaced by simple ``sleep'' commands. On the original workload, all the jobs were run at maximum DVFS. If our powercap scheduling algorithm decides to run a job at a lower speed, the emulated job will be executed slower.
We choose to use a performance degradation of 1.63 for all jobs, as our experiments and the literature agree that it is a reasonable value (see Control & measure power section). This performance degradation is computed with the maximum speed (2.7 GHz) and minimum speed (1.2 GHz), the intermediate values have been linearly interpolated. The replay of the time interval is based on the four following phases:

- the environment setup: SLURM is set in the closest state of the original run. We put in place the original SLURM configuration of Curie and modified only the parameters that allow our replay (node names, characteristics and power values).
- the interval initial state setup: runtime characteristics are put in place (queued and running jobs, fairshare values for each user)
- the actual workload replay: jobs are submitted with the same characteristics as they were on the original run (simple 'sleep' jobs, not real executions).
- data post-treatment: once the replay of the time interval end, we stop SLURM and then collect and gather information about the replay by the end of the 5 hours interval (jobs state, outputs and characteristics).

This methodology has some limitations. The initial placement of jobs is not always respected, and we do not replay node failures. Furthermore, job submissions depend of the response time experienced by users [31].

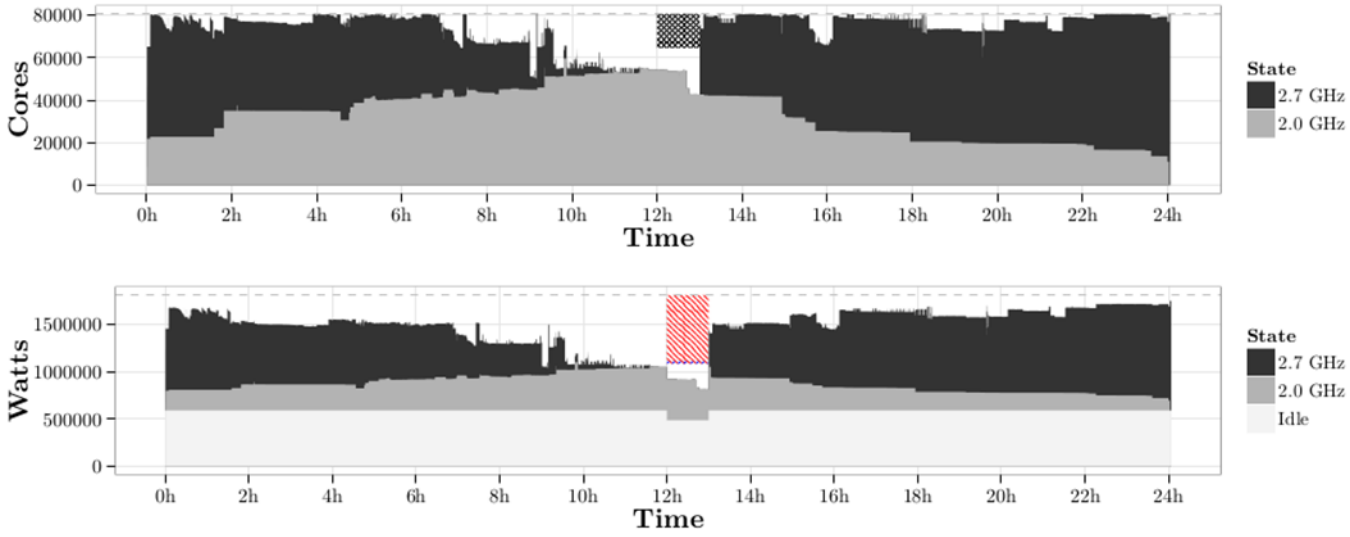
These limitations imply that comparisons to the original traces can not be conducted, but, as the replay is deterministic, we can compare the different replays.

In the experiments reported in the next subsection, we are evaluating the three different policies IDLE, SHUT and DVFS which enable respectively only idleness, only DVFS and only switch-off policies described in Section 5. The policies are tested under two powercap scenarios reserving respectively 60% and 40% of the available power budget for one hour in the middle of the replayed interval. All experimental results are compared between them along with a simple run where no powercapping takes place. Our goal is to compare system utilization in terms of CPUs and power usage along with stretch factors and effective work for each use case scenarios.

6.3 Analysis of results

We have shown experimentations and evaluation results that validated the initial developments in June's report. In this section we show some experimentations that enable the usage of both DVFS and shutdown technique in the same time called MIX and also some additional experiments with DVFS enabling the usage of the new functionality.

Figure 6 shows the system utilization (top) and power consumption (bottom) during the replay of the 24h workload using the MIX policy. The grey area in the top figure represents the system utilization of jobs executed upon cores with CPU Frequency of 2.0 GHz whereas the black area represents those running with 2.7 GHz. In the bottom figure the light grey area represents the minimum power consumption of the system if all nodes are idle and no jobs are executed, the grey area represents the additional power consumption of jobs whom the cores compute with 2.0 GHz and the black area reflects the additional power consumed by jobs that compute with 2.7 GHz.

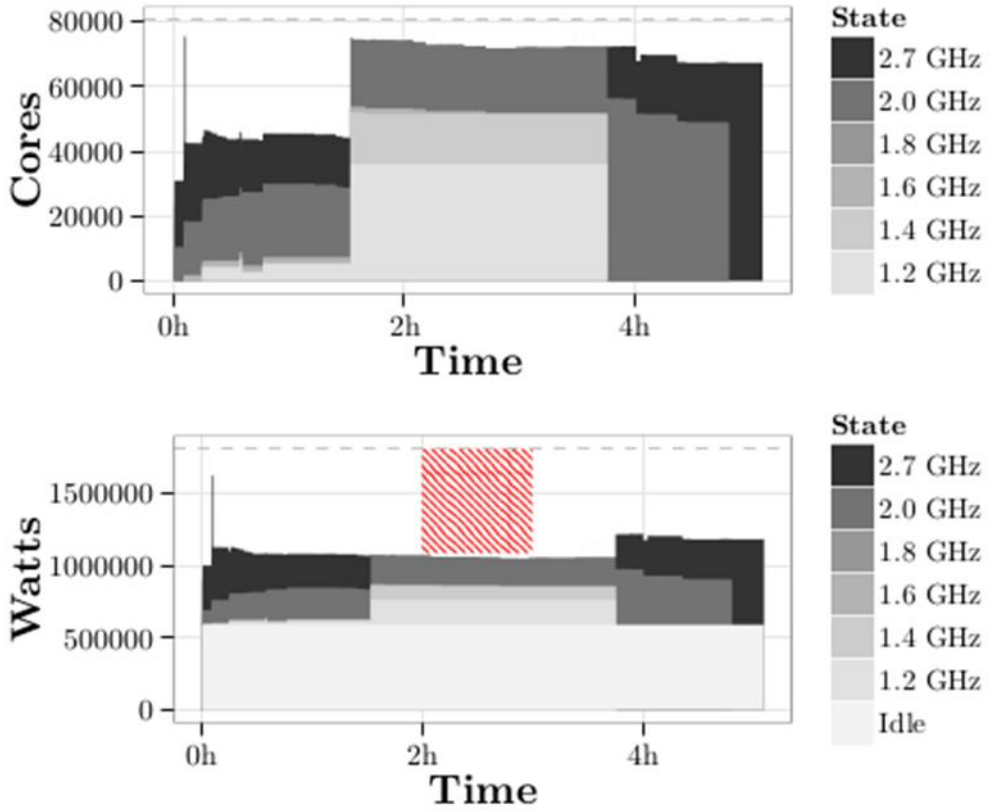


The reserved power, is represented by the hatched area in the bottom figure, thus the allowed powercap budget, for that period, is the remaining power below that area. The powercap is triggered in the beginning of workload that is why we observe that jobs are launched with lower CPU frequency directly from the start. Since we are in a MIX policy the offline part of the scheduling has reserved a certain number of nodes to shutdown. This can be viewed by the cross hatched area in the top figure during the period of powercap. The small blue cross hatched rectangle below the powercap reservation rectangle represents the bonus power gained back by the grouped shutdown of continuous nodes. This is actually gained power being used by the system for computations but it is plotted upon the reserved power hatched area to better reflect the proportions between them.

It is interesting to observe how while approaching the powercap reservation the system prepares itself for limited power usage by launching more jobs with 2.0 GHz. Similarly after the powercap has passed, utilization of 2.7 GHz cores increases because new jobs are launched with maximum frequency, while older jobs launched with 2.0 GHz, launched before or during the powercap, still remain but gradually decrease. After the powercapped period, we see that the system utilization in terms of cores increases directly to nearly 100%. It seems that a large job allocating more than 40000 cores was scheduled directly after the powercap period. This large job seems to be blocking other smaller jobs that follow and backfilling does not seem to work since thick gaps are witnessed during the powercap interval. Based on previous observations; backfilling is not efficient because of wrong walltime estimations. If we take a look at other 24h runs with a powercap of 40%,

DVFS and MIX show similar results: a work around 85% of the total possible work, while SHUT has a work of 94% of the total possible work. It is interesting to note that the energy consumption is a the lowest in the MIX mode.

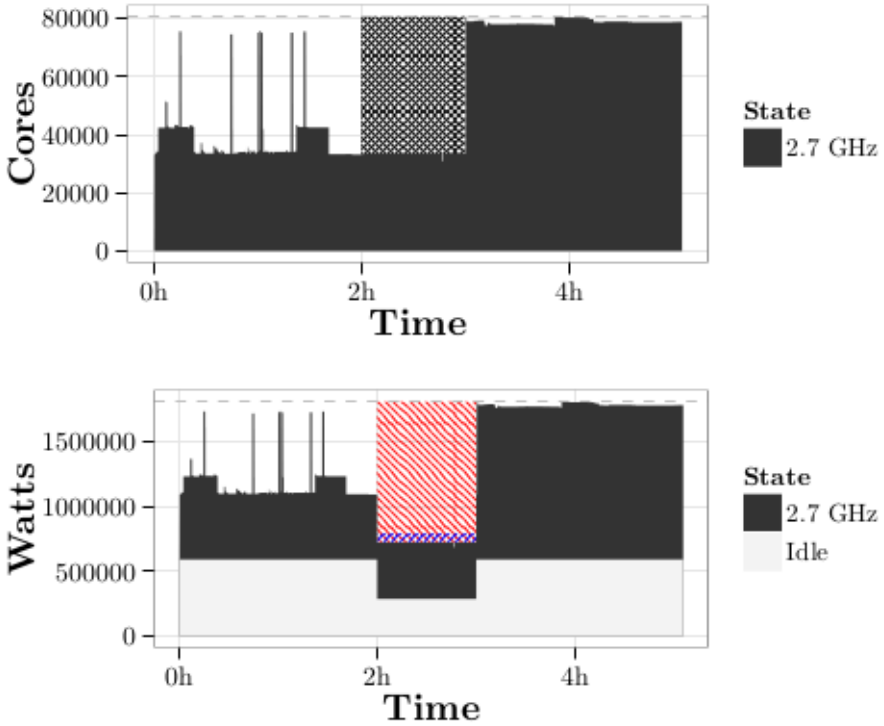
Figure 7a and 7b represent the system utilization for the smalljob and bigjob workloads with different use cases.



They are based on the same representations as the previous figure. The difference is that the left one provides results with SHUT policy whereas the right one with DVFS policy for 60% and 40% powercaps respectively. In the left figure we can observe how the shutdown of nodes makes big space in order to adapt the workload without wasting un-utilized cores.
In addition, we can see the value of power bonus due to the regrouping of nodes to be switched-off. Without the offline part of the scheduler this bonus would not be possible.

Furthermore, we see how the system utilization increases directly after the powercap interval to 100%. It is interesting to see how backfilling does not take place a lot while preparing for the powercap period. This is due to the nature of type of jobs which is mainly big jobs along with the walltime problems that we explained before. In the right figure, different tones of grey represent the different frequencies until the black area which is 2.7GHz. We can see how the appearance of low CPU Frequencies increase while approaching to the powercap period with a total disappearance of 2.7GHz frequencies in close regions to the powercap interval. DVFS policy manages to obtain high system utilization with a low

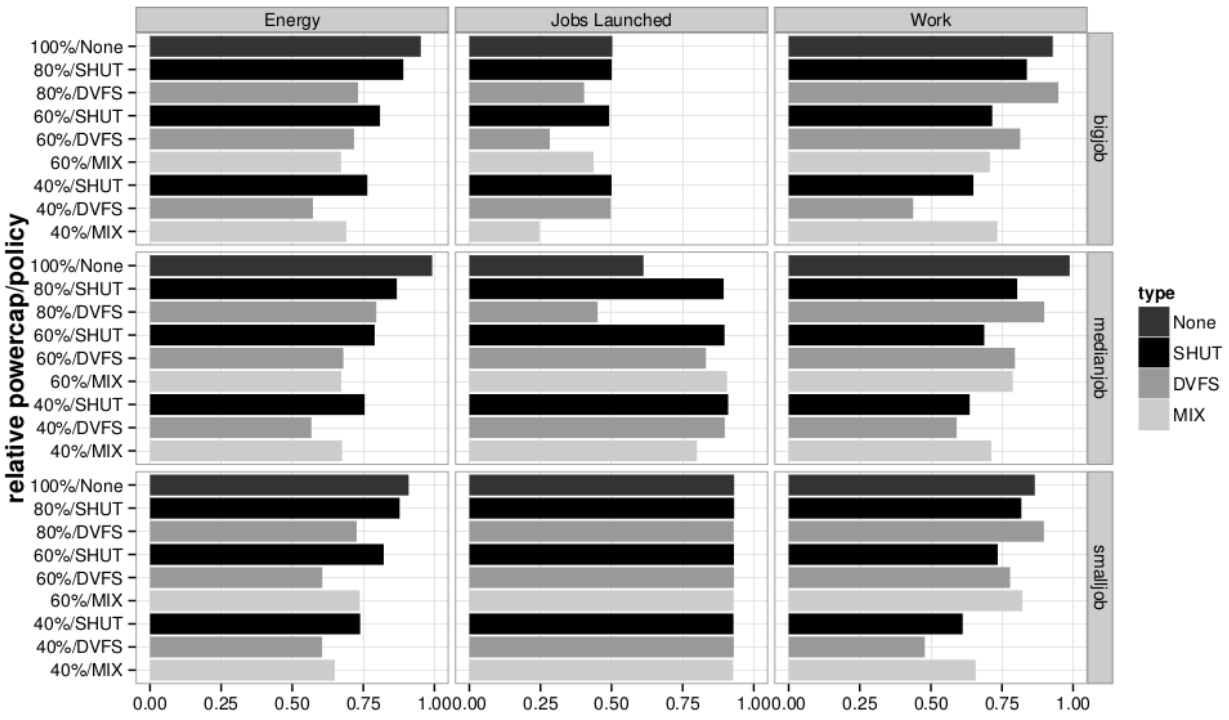
power consumption. We also have done several run with DVFS and switch-off mechanisms deactivated. The only solution for our algorithm is to let nodes idle. As expected, this solution has the worst work (about 40% lower than other modes), while keeping about the same energy consumption.



Let us now look at the impact of the policies for the performances. Figure 8 provides the different runs executed to compare the performance of the different powercap scheduling modes for 5 hours workload. Considering columns we observe the total consumed energy, the number of launched jobs and the total work. In terms of rows we have different groups. The groups based on the workload (left): bigjob, medianjob and smalljob along with the groups representing powercap reservations: 100%, 80%, 60% and 40% which reflect the system power which is allocated for computation. Furthermore distinctions between the different scheduling modes is also made in groups of particular rows in the figure.

Only jobs that were running during the replayed time interval are taken into account, and all measures are normalized to the maximal possible value. In the histograms we observe that DVFS mode's work is always larger than SHUT mode's work and that is because jobs run with lower CPU Frequency and hence the walltime is increased. The MIX mode provides most of the time the best energy consumption, while having a work in the same order of others modes. In the medianjob workload, 100%/None and 80%/DVFS runs launched less jobs than others run, while having a high utilization. It seems that in

these runs, the algorithm chooses to schedule a huge job preventing a large number of other jobs to be launched.If we take a look at each mode independently we can see that for every type of workload work and energy decrease proportionally to the powercap diminution.
Furthermore, DVFS mode seems to be decreasing more rapidly below 60% whereas SHUT and MIX modes appear to be more consistent. Switch- off mechanism (SHUT, MIX) seems to be more efficient if we consider the tradeoffs energy/work and this is basically related to the in-advance preparation in the offline part and the gained power due to the bonus.



7. Conclusion and future works

We presented in this paper a new scheduling algorithm for dealing with power limitations in large scale HPC clusters. The algorithm was developed for a resource and job management system and implemented upon SLURM. It is composed of two phases: an offline part where the planning takes place (choice of policy, selection of group of nodes to be switched-off, etc.) followed by an online part where the power reduction is applied.
The implementation upon SLURM resulted into the design of three powercap policies, namely DVFS, SHUT and MIX which respectively take advantage of CPU Frequency scaling, nodes shut down and mixed capabilities in order to achieve power reductions whenever needed.
One of our main objectives was to enable the scheduler to determine automatically the best powercap mechanism for the nodes and we showed how this depends on the architecture, the power consumption of the components and the actual workload.
The new developments will appear on the main branch of SLURM in the upcoming version 15.08. As far as our knowledge, this is the first work that considers power capping techniques in the level of job scheduling for a resource and job management system in HPC. The study allowed us to validate the algorithms and evaluate the different policies through real-scale emulation of a petaflopic supercomputer. In particular we performed experiments with emulation of Curie's characteristics and calculated power values using replay of a real workload trace collected from the production of Curie on 2012.

The experimental results validate the model and provide interesting initial insights. Switching-off nodes appear to be the most efficient policy in our use cases of less than 60% powercaps, mixed policy seems to be the more consistent on and finally frequency scaling provides better results with large powercaps of 80%.

The studies will continue to correlate the proposed model with application preferences concerning DVFS. Indeed, if an application is able to provide optimized DVFS values, this should be taken into account by the algorithm. Then, the global performance of the cluster will be improved while respecting the powercap. SHUT policy makes an offline selection of the group of nodes to be switched-off in order to take advantage of the power bonus. Nevertheless, this might increase the fragmentation of the system in case of un-homogeneous infrastructures. Hence, deeper studies are needed to compare the rigidity of the selection of the nodes to be switched-off with a more flexible selection of nodes.

For future improvements on the code for DVFS, we will consider to dynamically change the CPU frequencies while the jobs are running, this will allow nodes to adjust the power consumption instantly whenever it is needed. This will eventually result into faster power decrease when a powercap period is approaching and lower jobs' turnaround time after a powercap period is over. The optimal DVFS choice for the best power/performance trade-offs could be also determined through a particular profiling run as proposed in [34]. In addition, exploring other techniques to limit power consumption upon utilized nodes, such as RAPL, will be also studied. Finally, we would like to adapt the power capping algorithm in order to consider the real-time power consumption measures of the nodes [26], instead of considering the static values defined during the initialization phase.

8. References

[1] S.-g. Kim, C. Choi, H. Eom, H. Yeom, and H. Byun, “Energy-centric DVFS controling method for multi-core platforms,” in High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:, 2012.

[2] B. Lawson and E. Smirni, “Power-aware resource allocation in high-end systems via online simulation,” in Proceedings of the 19th annual international conference on Supercomputing, 2005.

[3] A. B. Yoo, M. A. Jette, and M. Grondona, “SLURM: Simple Linux utility for resource management,” in Job Scheduling Strategies for Parallel Processing. Springer Verlag, 2003.

[4] Y. Georgiou, T. Cadeau, D. Glesser, D. Auble, M. Jette, and M. Hautreux, “Energy accounting and control with slurm resource and job management system,” 2014.

[5] 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, New Jersey, USA, 2005.

[6] R. Sch¨ ne and D. Hackenberg, “On-line analysis of hardware performance events for workload characterization and processor frequency scaling decisions,” in Proceedings of the 2Nd ACM/SPEC International Conference on Performance Engineering, 2011.

[7] H. Kimura, M. Sato, T. Imada, and Y. Hotta, “Runtime DVFS control with instrumented code in power-scalable cluster system,” in 2008 IEEE International Conference on Cluster Computing, 2008.

[8] A. Gandhi, M. Harchol-Balter, R. Das, J. O. Kephart, and C. Lefurgy, “Power capping via forced idleness,” 2009.

[9] B. Rountree, D. Lowenthal, M. Schulz, and B. De Supinski, “Practical performance prediction under dynamic voltage frequency scaling,” in Green Computing Conference and Workshops (IGCC), 2011 International, 2011.

[10] M. Etinski, J. Corbalan, J. Labarta, and M. Valero, “Understanding the future of energy-performance trade-off via DVFS in HPC environments,” Journal of Parallel and Distributed Computing, 2012.

[11] V. W. Freeh, D. K. Lowenthal, F. Pan, N. Kappiah, R. Springer,B. L. Rountree, and M. E. Femal, “Analyzing the energy-time trade-off in high-performance computing applications,” IEEE Transactions on Parallel and Distributed Systems, Jun. 2007.

[12] D. Aikema, C. Kiddle, and R. Simmonds, “Energy-cost-aware scheduling of HPC workloads,” in World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a, 2011, pp. 1–7.

[13] J. Hikita, A. Hirano, and H. Nakashima, “Saving 200kw and $200 k/year by power-aware job/machine scheduling,” in Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on, 2008.

[14] E. D. Demaine, M. Ghodsi, M. T. Hajiaghayi, A. S. Sayedi-Roshkhar, and M. Zadimoghaddam, “Scheduling to minimize gaps and power consumption,” in Proceedings of the Nineteenth Annual ACM Symposium on Parallel Algorithms and Architectures, ser. SPAA ’07. New York, NY, USA: ACM, 2007.

[15] L. Shang, L.-S. Peh, and N. K. Jha, “Dynamic voltage scaling with links for power optimization of interconnection networks,” in High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on, 2003.

[16] J. D. Moore, J. S. Chase, P. Ranganathan, and R. K. Sharma, “Making scheduling” cool”: Temperature-aware workload placement in data centers.” in USENIX annual technical conference, General Track, 2005.

[17] T. Singh and P. K. Vara, “Smart metering the clouds,” 2009.
[18]B. Rountree, D. H. Ahn, B. R. de Supinski, D. K. Lowenthal, and M. Schulz, “Beyond DVFS: a first look at performance under a hardware-enforced power bound,” in Parallel and Distributed Process-ing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International.
[19] S. Reda, R. Cochran, and A. K. Coskun, “Adaptive power capping for servers with multithreaded workloads,” IEEE Micro, 2012.
[20]X. Fan, W.-D. Weber, and L. A. Barroso, “Power provisioning for a warehouse-sized computer,” ACM SIGARCH Computer Architecture News, 2007.
[21] G. A. Geronimo, J. Werner, R. Weingartner, C. B. Westphall, and C. M. Westphall, “Provisioning, resource allocation, and DVFS in green clouds.”
[22] M. Etinski, J. Corbalan, J. Labarta, and M. Valero, “BSLD threshold driven power management policy for HPC centers,” in 2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010.
[23]——, “Optimizing job performance under a given power constraint in HPC centers,” in Green Computing Conference, 2010 International, 2010.
[24]——, “Parallel job scheduling for power constrained HPC systems,” Parallel Computing, 2012.
[25] A. Mu’alem and D. Feitelson, “Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling,” Parallel and Distributed Systems, IEEE Transactions on, vol. 12, no. 6, pp. 529–543, Jun 2001.
[26] D. Tsafrir, Y. Etsion, and D. G. Feitelson, “Modeling user runtime estimates.” Springer, 2005.
[27] “http://www.netlib.org/linpack/.”
[28] J. D. McCalpin, “A survey of memory bandwidth and machine balance in current high performance computers,” IEEE TCCA Newsletter, 1995.
[29] “https://software.intel.com/en-us/articles/intel-mpi-benchmarks.”
[30] H. J. Berendsen, D. van der Spoel, and R. van Drunen, “Gromacs: A message-passing parallel molecular dynamics implementation,” Computer Physics Communications, 1995.
[31] E. Shmueli and D. G. Feitelson, “Uncovering the effect of system performance on user behavior from traces of parallel systems,” in MASCOTS. IEEE Computer Society, 2007.