Adaptive Resource and Job Management for limited power consumption

Yiannis Georgiou
BULL
Yiannis.Georgiou@bull.net

David Glesser
BULL
and Univ. Grenoble-Alpes
David.Glesser@bull.net

Denis Trystram
Univ. Grenoble-Alpes
Institut Universitaire de France
trystram@imag.fr

***Abstract*—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 high performance computing systems. 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.**

We introduce in this paper a new scheduling strategy that can adapt the executed workload to a limited power budget. The originality of this approach relies upon a combination of speed scaling and node shutdown techniques for power reductions. It is implemented into the widely used resource and job management system SLURM. Finally, it is validated through large scale emulations using real production workload traces of the supercomputer *Curie*.

Keywords: Resource Management; powercap; power-aware scheduling;

I. Introduction

Energy consumption has become one of the most crucial issues in the evolution of High Performance Computing systems. The increase in computational performance of such platforms has come with an even greater increase in energy consumption, turning energy into 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 HPC center is directly related to the energy consumption, such centers have additional motivation to regulate the energy usage. However, controlling the energy for a certain duration demands novel approaches for adjusting the instantaneous power on a daily basis.

Research efforts in all different abstraction layers of Computer Science, from hardware, to middleware up to applications, strive to improve energy consumption 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 have been widely used for energy reduction [1,2]. Nevertheless, the regulation of power needs to pass from a centralized system middleware 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 is aware of both the hardware component's state and information, along with details about the users' workloads and the executed applications.

The main contribution of this work is to propose a new powercapping mechanism based on a combination of both DVFS and node shutdown power-reduction techniques. We study a generic model which 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 impact of DVFS has been studied on several actual applications along with the effects of using grouped shutdown of nodes. The new scheduling algorithms have been implemented in the widely used open source workload manager SLURM. As our aim is to integrate this mechanism into large-scale supercomputers such as *Curie*, we have validated our model, algorithms and implementations using large-scale experimentations based upon real production workload traces of the *Curie* petaflopic supercomputer.

The content of the paper is as follows. We start with an overview of the most relevant approaches used so far for reducing the energy consumption and powercapping in Section II. Section III presents our model for determining the best combination between DVFS and shutdown for power limitations. The scheduling algorithm is presented in Section IV, while Section V presents the adaptation of the mechanisms using the real supercomputer's characteristics. We evaluate our algorithms and report the results of our experiments in Section VI. Finally, conclusions and perspectives are discussed in Section VII.

II. Related Works

A. Controlling the power

Dynamic Voltage and Frequency Scaling (*DVFS*) 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 theoretical works oriented on speed-scaling, 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-cores and extended in [3] for more general platforms. Schöne and Hackenberg [4] used register measurements for determining the frequency. Kimura et al. [5] provided also a new mode of control of DVFS through a code-instrumented DVFS control. Then, Gandhi et al. [6] 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. [7] developed a performance prediction model outperforming previous models. Etinski et al. [8] studied how to improve the trade-off energy versus completion time on applications. Freeh et al. [9] provided a huge number of experiments for measuring the Energy over 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 (also called *shutdown*). Lawson et al. [2] proposed an opportunistic mechanism, which switches off a node after a significant idle period. Aikema et al. [10] studied the energy from the view point of a cost function where a node becomes idle is considered as a zero-cost in term of energy. Under the assumption of under-loaded cluster, Hikita et al. [11] presented a batch scheduler that minimizes the number of active nodes while keeping the same performances. In [12], 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.

B. Powercapping

In the following paper, we are focusing on powercap as the main topic. Some papers are considering powercapping at the node level, for instance [**?**] used a new feature available in Intel processors to achieve a local powercap and [14] packed threads together in order to tune the DVFS. Powercap has been studied by Etinski et al. in [17] where DVFS is the only mechanism used to achieve powercapping while keeping good performances. We consider here a more sophisticated mechanism including also shutdown.

In the context of clouds, Geronimo et al. proposed a virtual machine manager that can use DVFS, update the virtual machine resources, migrate them and shutdown opportunistically some processors [18]. Fan et al. defined in [19] a methodology to reduce the global cost of data-centers by buying more processors and capping their power consumption. Our work is dedicated to HPC supercomputers where the constraints are very different. To the best of our knowledge, there is no similar work which consider DVFS and shutdown simultaneously in order to adapt the power consumption of a HPC cluster. Pierson and Casanova proposed a theoretical approach based on mixed Integer Linear Programs restricted to a single application [20]. We propose here a generic mechanism which selects the best strategy among DVFS, shutdown or both together.

III. Energy and Power analysis

In this section, we describe a new model that enables to determine when to switch off nodes and to determine the right frequency.

A. Tradeoff Switch-off between DVFS

Let us define W as the maximum amount of computations that could be performed during a given period of time T:

$$W = T \times (\frac{N - N_{off} - N_{dvfs}}{1} + \frac{N_{dvfs}}{deg_{min}}) \qquad \text{(C1)}$$

Where N is the total number of nodes; N_{off} is the total number of nodes which are switched off and N_{dvfs} is the total number of nodes whose frequency has been decreased.

deg_{min} represents the percentage of *degradation* of the completion time at the minimum frequency compared to the maximum one. The justification to take deg_{min} at the minimum frequency is to consider the maximum possible impact on applications. This choice will be discussed in Section V. Without loss of generality, T will be set to 1.

Obviously,

$$N_{dvfs} + N_{off} \leq N \qquad \text{(C2)}$$

The consumed power should be lower than the powercap:

$$\begin{aligned} &N_{off}.P_{off} + N_{dvfs} \times P_{min} + \\ &(N - N_{off} - N_{dvfs}) \times P_{max} \leq P \end{aligned} \qquad \text{(C3)}$$

Where P_{off}, P_{min} and P_{max} are respectively the power consumed by a switched-off node, by a node in the lowest frequency and by a node in the maximum frequency. P is the powercap, *i.e.* the global available power at this moment.

The previous constraints C1 and C3 correspond respectively to a plane and an half-space in the 3D space (W, N_{off} and N_{dvfs}). We are looking for points that maximize W within the previous constraints. The intersection of the two previous surfaces is a segment. We then consider the point that maximizes W on this segment, there are four cases:

1) There is only some switched-off nodes (the best point is on located on the plane (W, N_{off}).
2) There is only use of DVFS on some nodes (the best point is on located on the plane (W, N_{dvfs}).
3) Both previous options lead to the same maximum load (all the points of the segment are eligible)
4) The powercap is too low, both mechanisms should be used to reach the maximal W (the best point is at the intersection of the segment and the constraint C2).

The value for the two first cases can be easily computed thanks to the following formulas:

$$\begin{cases} N_{off} = \frac{P - N \times P_{max}}{P_{off} - P_{max}} \\ N_{dvfs} = 0 \end{cases} \quad \text{or} \quad \begin{cases} N_{off} = 0 \\ N_{dvfs} = \frac{P - N \times P_{max}}{P_{min} - P_{max}} \end{cases}$$

Let W_{dvfs} be available computational load available using only DVFS (similarly W_{off} for switch-off). DVFS is better to use when $W_{dvfs} > W_{off}$. Which is equivalent to $\rho > 0$, where $\rho = 1 - \frac{1}{deg_{min}} - \frac{P_{max} - P_{dvfs}}{P_{max} - P_{off}}$. In the third case, both mechanisms give the same results. Thus, for the three first cases, it is easy to determine which mechanism to use depending on ρ. Let us focus on the last case when the powercap is too low. The powercap is too low when $P < N \times P_{off}$ or $P < N \times P_{min}$. The first expression can not happen practically since the powercap will be less than when the cluster is completely switched-off. Let us define $P = \lambda N P_{max}$, where λ is the powercap normalized by the maximum power consumption. The second expression becomes $\lambda < \frac{P_{min}}{P_{max}}$, which means that the powercap can not be less than $\frac{P_{min}}{P_{max}}$ if DVFS is the only mechanism used to control power. In this case, the best choice for N_{off} and N_{dvfs} is:

$$\begin{cases} N_{off} = N - N_{dvfs} \\ N_{dvfs} = \frac{P - N \times P_{off}}{P_{min} - P_{off}} \end{cases}$$

We are now able to determine which mechanism to use depending on job, cluster and powercap. We present in the following section an optimization of the switch-off mechanism.

B. Power Bonus when switching off hardware components

In HPC clusters, there are several hardware levels from a power consumption point of view. A level is defined as a group of different hardware components that can be switched-off simultaneously. The extra power consumption gained by the network switches when powered-off is called a 'power bonus'. It allows us to reduce even more the power consumed by a cluster when disabling part of its compute power. In modern architectures, typical HPC clusters will have different 'power bonus' related to the different levels of components aggregation. Correctly selecting the computing elements to switch-off when coping with a power constraint will thus enable to sum up the different bonus at the different levels and maximize the power available to the active compute elements and their hardware dependencies. The lowest level considered in the our model is the multicore node. No actual power bonus is currently provisioned at this level. Individual cores and sockets can not be switched-off individually.

IV. Scheduling under power control

A. 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 selection of jobs after prioritization among the group of pending jobs, then, the selection of the adequate resources. The first phase may involve various mechanisms to select the next job to be treated. For instance, the usual backfilling [21] 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. The main idea is to consider that power is treated as a new type of resource. According to its state (PowerDown, Idle, Busy, etc.), the resource will consume a different amount of power. At any moment, the RJMS can deduce the power consumption of the whole cluster keeping the state of each resource internally. The characteristic of power can be related to any components 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 Baseboard Management Controller is still on). The algorithm goes one step further by considering the setting of the various CPU frequencies as different power states. Hence, the power consumption of each node will change depending on the 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.

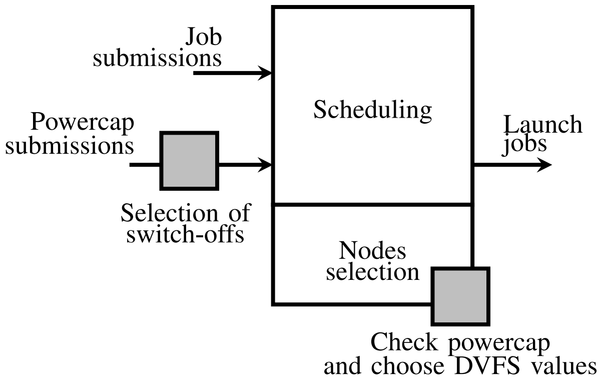


Fig. 1: SLURM architecture. In grey, modified part.

B. Scheduling algorithm for powercapping

Our powercapping algorithm is composed of two successive parts: A first part where the decisions for power management are taken in advance (to better prepare future actions) followed by the power distribution. The algorithm is activated as soon as a powercap reservation 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 power reservation. If any of these cases hold, 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 frequencies where the allocated nodes may run. If no frequency allows the execution within the power budget then the job remains pending. In the opposite, the job is executed at the maximum CPU frequency that allows the job to be executed keeping the cluster into the power budget.

```
Input: The user creates a powercap reservation,
       indicating the interval time and the value of the
       powercap ($P$).
if $P < N.P_{min}$ then
    $N_{dvfs} = \frac{P - N \times P_{off}}{P_{min} - P_{off}}$
    $N_{off} = N - N_{dvfs}$
    Make a switch-off reservation of $N_{off}$ nodes during
    the powercap.
else
    $\rho = 1 - \frac{1}{deg_{min}} - \frac{P_{max} - P_{dvfs}}{P_{max} - P_{off}}$
    if $\rho <= 0$ then
        $N_{off} = \frac{P - N \times P_{max}}{P_{off} - P_{max}}$
        Make a switch-off reservation of $N_{off}$ nodes
        during the powercap.
    end
end
```

Algorithm 1: The selection of switch-offs algorithm to control the nodes switch-off reservations.

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 (some nodes will have to be kept 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 remain within the powercap value. Since the number of idle nodes may change, the optimal CPU frequency may also change from one job to another.

```
Input: The job to be scheduled.
job.DVFS = highest possible DVFS
while $currentPower + N_{job.DVFS} \times job.requiredNodes > P$ do
    if $job.DVFS == minimumDVFSpossible$ then
        return Impossible to schedule the job now.
    end
    job.DVFS = a slower value of job.DVFS
end
```

Algorithm 2: Simplified algorithm of second part (check of powercap and choose DVFS values of jobs).

The adequate energy saving mechanism is chosen in the selection of switch-offs step. System administrators can force one or another mechanism. We defined three policies *SHUT*, *DVFS* and *MIX*. *SHUT* 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 can force jobs to be executed at lower CPU frequencies. Finally, *MIX* is a mixed *DVFS* and *SHUT* strategy, which considers both possibilities of saving power. In *DVFS*, *SHUT* or *MIX* modes, the system will decide which mechanism is the most suitable one using the equations introduced in Section III-A.

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.

This scheduling algorithm has been implemented upon the open-source resource and job management system SLURM [22]. All implemention details may be found in the SLURM release 15.08.

V. Adapting powercapping logic for Curie

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.

A. Presentation of Curie

Curie[1] is owned by GENCI[2], it is the first french Tier-0 system opened to scientists through the participation into the PRACE[3] research infrastructure. Since its upgrade in February 2012, Curie consists of 280 Bullx B chassis housing 5,040 Bullx B510 nodes, each with two 8-core Intel Sandy Bridge processors for a total of 80640 cores. Curie was ranked 26th among the 500 most powerful supercomputers on June's 2014 Top500 list[4].

Level	Consumption	Acummulated bonuses	Acummulated 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 I: Power consumption and the possible saved watts when various levels of the cluster are switched-off.

We have seen in Section III-B 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, there are four levels that can be switched-off. Table 2 gives the power consumption at each level.

B. Control and Measure power

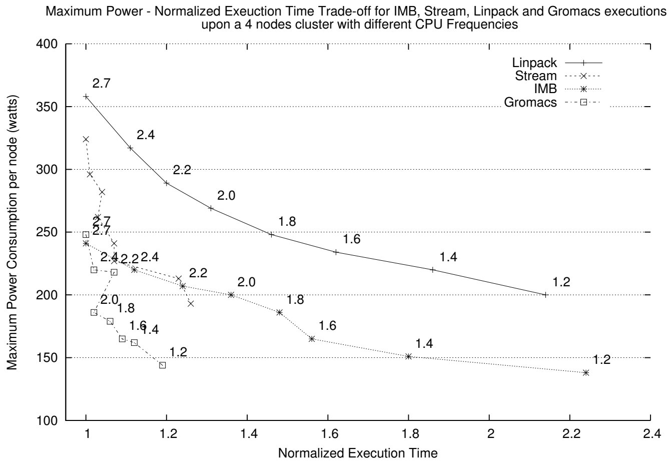


Fig. 2: Maximum Power - Execution Time Tradeoffs for Linpack, Stream, IMB and Gromacs benchmarks at different CPU frequencies

The power consumption of the different states of a node may be given by the constructor or calculated through experimentations. We have run three different widely used benchmarks and one application to measure the characteristics of Curie cluster. We have chosen a first benchmark to stress all computing resources (Linpack [23]), a second one targeting memory (Stream [24]), one focused on network (IMB [25]) and the last one is a widely used application for molecular

[1]http://www-hpc.cea.fr/en/complexe/tgcc-curie.htm
[2]http://www.genci.fr/en/our-computers
[3]http://www.prace-project.eu/
[4]http://www.top500.org

dynamics simulation (GROMACS [26]). 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 [27]. Figure 3 gives the evolution of the completion time and the maximum watts consumed at different DVFS values.

Table 4 presents the maximum power consumption observed on a node for each DVFS value based upon the results of all 4 applications. These measurements set the maximum Watts that a node can consume. There is huge gap between switched-off and idle nodes. Both states do not produce computational power, but a switched-off node consumes an 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 II: Table of the maximum power consumption of a Curie node in different states.

Also, from these benchmarks we compute the performance impact of DVFS. For simplicity, only the maximum and minimum frequencies are considered. Thus, in the following the *degradation* of performance is between the maximum and minimum DVFS values. As seen in Section II, the *degradation* of performances has already been studied. The *degradation* has been measured for the NAS benchmark [9], the SPEC float and integer benchmarks. A *degradation* of 163% is assumed to be a good approximation [15]. Table 5 summarizes the data obtained on Curie for various benchmarks. Common values of *degradation* clearly show that shutdown is the best mechanism to use for reasonable values of powercap.

Benchmark	deg_{min}	ρ	Best mechanism
NA	2.27	0	-
linpack	2.14	-0.027	Switch-off
IMB	2.13	-0.029	Switch-off
SPEC Float [9]	1.89	-0.088	Switch-off
SPEC Integer [9]	1.74	-0.134	Switch-off
Common value [15]	1.63	-0.174	Switch-off
NAS suite [9]	1.5	-0.225	Switch-off
STREAM	1.26	-0.350	Switch-off
GROMACS	1.16	-0.422	Switch-off

TABLE III: Comparison between DVFS and switch-off in Curie for various benchmarks.

In our context, the *SHUT* policy appears to be the best one. Hence, the selection of switch-offs algorithm would never mix *DVFS* and *SHUT* modes. However, we observe in the benchmarks' results that, unlike the power/performance trade-off, the energy/performance trade-off is not monotonic. The best values are between 2.7 GHz and 2.0 GHz. As a consequence, we consider *MIX* with higher DVFS values. The aim of this *MIX* is to improve performance and energy consumption while remaining under the power budget. This algorithm is the same as the one previously described but the minimum DVFS frequency is 2.0 GHz instead of 1.2 GHz. Both mechanisms should be used together when the powercap is less than 75% of the maximum power. In the remainder of the paper all references to *MIX* consider always the high DVFS values (2.0-2.7GHz).

In case we cannot switch-off nodes, *SHUT* can be implemented by keeping nodes idle. ρ becomes positive for all *degradation* values of benchmarks. Thus, *DVFS* turns out to be the best policy in all cases.

VI. Experimental Evaluations

Our choices for experimental evaluations were to: i) use the real workload trace of Curie for approximating production executions of a large-scale supercomputer, ii) take into account the real power consumption data of Curie as discussed in Section III and iii) evaluate the powercapping scheduling algorithm as implemented upon SLURM iv) make use of an emulation technique to study SLURM by using only a small fraction of physical machines.

A. Platform and Testbed

The experiments have been performed upon Nova2 platform which is an internal BULL cluster dedicated for experimentations. The cluster is composed by Intel Sandy Bridge processors with 65 GB of Memory and Infiniband network.

In order to enable real-scale experiments of Curie's workloads with SLURM, we need to deploy a configuration of the same size. This is done by making use of an internal SLURM emulation technique called *multiple-slurmd*. This technique is described and validated in [28]. In our context, we deploy 5040 nodes of Curie with only 16 physical nodes of our experimental platform Nova2.

We propose to replay time intervals extracted from a real workload trace of Curie supercomputer in 2012[6]. We selected three intervals of 5 hours and one interval of 24 hours with high utilization, big number of jobs in the queue and short inter-arrival time. The intervals used in the following experiments are: i) *medianjob*, with jobs that are representative of the whole workload, ii) *smalljob*, with more small jobs than in *medianjob*, iii) *bigjob*, with more big jobs than in *medianjob*, iv) *24h*, with jobs that are representative of the whole workload. In the extracted traces, Curie is overloaded. Most of the jobs are small compared to the Curie size: 69% are jobs with less than 512 cores and ran for less than 2 minutes. 0.1% of jobs are huge, these jobs use more than the half of the whole cluster. It is important to notice that in these particular traces, the users estimate badly the runtimes. The replay methodology is not described here due to a lack of space. We kindly invite the readers to read [28] for more details.

In the experiments reported in the next section, we are evaluating the three policies *DVFS*, *SHUT* and *MIX*. They are tested under three powercap scenarios reserving respectively 80%, 60% and 40% of the available power budget for one hour in the middle of the replayed interval. Powercap reservations

[5]IPMI (Intelligent Platform Management Interface)

[6]http://www.cs.huji.ac.il/labs/parallel/workload/l_cea_curie/index.html

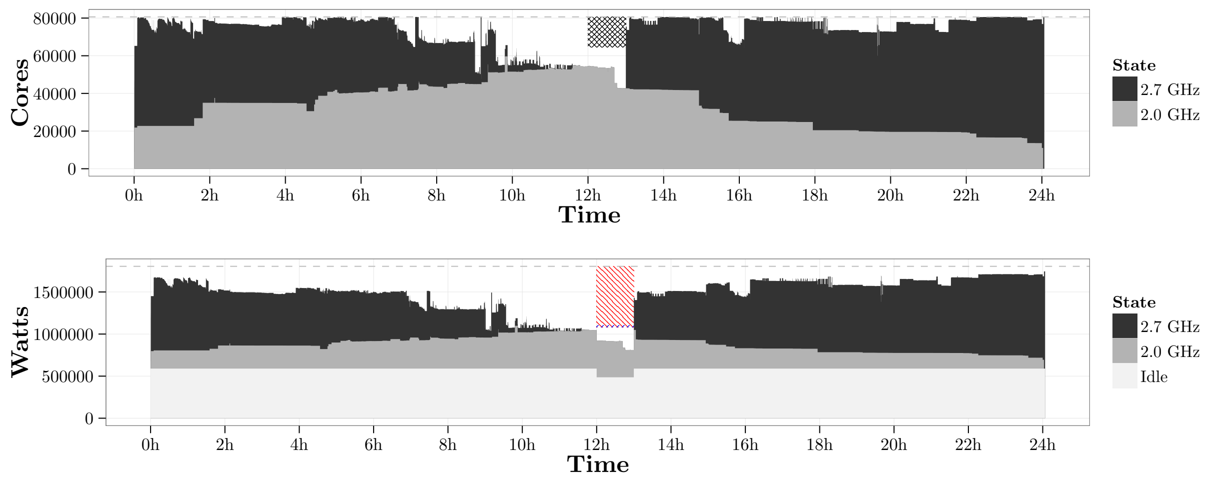
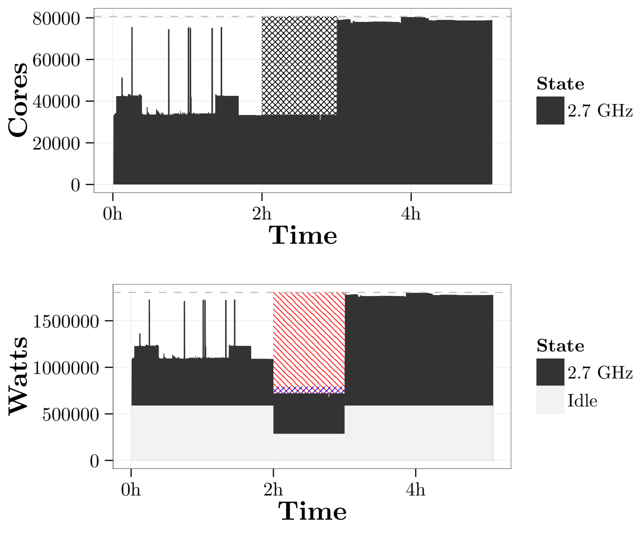
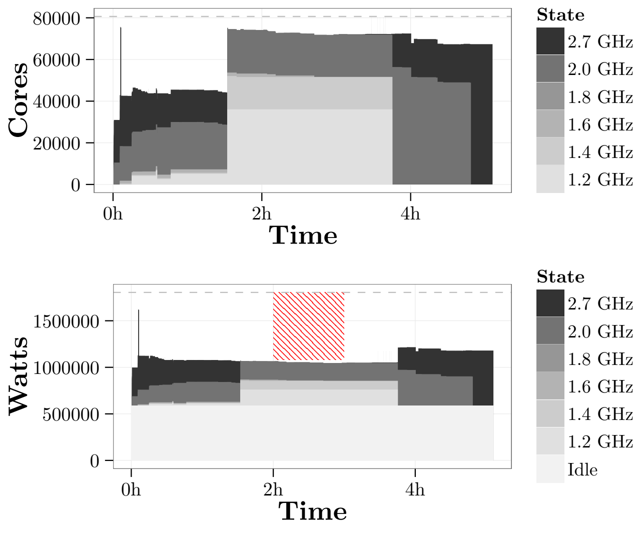


Fig. 3: System utilization for the *MIX* policy in terms of cores (top) and power (bottom) during the 24 hours workload with a reservation (hatched area) of 1 hour of 40% of total power. The switched-off nodes are represented by the crosshatched area.



(a) Powercap of 60% with mainly big jobs and *SHUT* policy



(b) Powercap of 40% with mainly small jobs and *DVFS* policy

Fig. 4: System utilization for the different runs in terms of cores (top) and power (bottom) during 5 hours workload with a powercap reservation (dash area) of 1 hour of 60% or 40% of total power.

are made at the beginning of the workload replay. The power consumption of jobs and the whole cluster are calculated theoretically based on the values of Table 4. 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 the effective work for each scenario.

B. Analysis of results

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 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. At the bottom, 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 at 2.0 GHz and the black area reflects the additional power consumed by jobs at 2.7 GHz. The reserved power, is represented by the hatched area in the bottom, thus the allowed powercap budget is the remaining power below that area. The powercap is triggered at the beginning of workload that is why we observe that jobs are launched with lower CPU frequency directly from the start. Since the policy is *MIX* the first 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 during the period of powercap. The small cross hatched area below the powercap reservation represents the bonus gained by grouping shutdown of continuous nodes.

It is interesting to observe how while approaching the

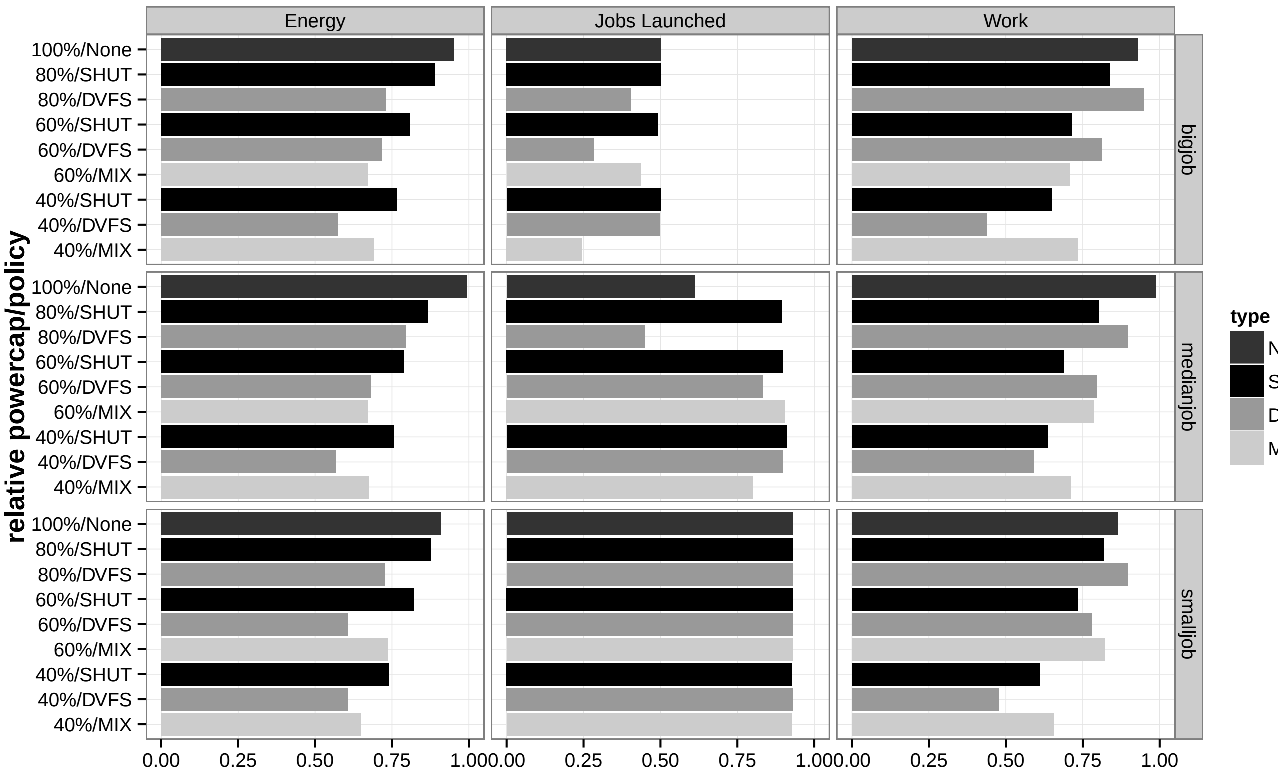


Fig. 5: Comparison of different scenarios of policies and powercaps based on normalized values of total consumed energy, launched jobs and accumulated cpu time during the 5 hours workload interval.

powercap reservation the system prepares itself for limited power usage by launching more jobs at 2.0 GHz. Similarly after the powercap, the utilization of 2.7 GHz cores increases because new jobs are launched at maximum frequency, while older jobs still remain at 2.0 GHz but gradually decrease. After the powercapped period, the system utilization increases directly to nearly 100% in terms of cores. A large job with more than 40000 cores was scheduled directly after the powercap. This large job is blocking other smaller jobs that follow and backfilling does not 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 notice that the energy consumption is the lowest in the *MIX* mode.

Figures 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 previously. The difference is that the left one provides results with *SHUT* whereas the right one is for *DVFS* for 60% and 40% powercaps respectively. In the left figure, we observe how the shutdown creates big space in order to adapt the workload without wasting un-utilized cores. It is interesting to see how backfilling is limited while preparing for the powercap period. This is due to the type of jobs which concern mainly big jobs along with the walltime problems as before. In the right figure, different tones of grey represent the different frequencies until the black area which is 2.7GHz. There are more and more low frequencies while approaching to the powercap period with a total disappearance of 2.7GHz frequencies close to the powercap interval. *DVFS* manages to obtain high 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 on the performances. Figure 8 provides the different runs executed to compare the performance of the various powercap modes for 5 hours workload. The columns report the total consumed energy, the number of launched jobs and the total work. The results are given for three groups based on *bigjob*, *medianjob* and *smalljob*. For each group, we considered 100%, 80%, 60% and 40% of powercap reservations. 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*'s work is always larger than in *SHUT* since the jobs are running at lower CPU frequencies and hence, the walltime is increased. *MIX* provides most of the time the best energy consumption, while keeping a work in the same order as the others.

In the *medianjob* workload, 100%/None and 80%/DVFS launched less jobs than other runs, while having a high utilization. In these runs, the algorithm chooses to schedule a huge job preventing a large number of others to be launched.

If we take a look at each mode independently, the work and energy decrease proportionally to the powercap diminution for every type of workload. Furthermore, *DVFS* seems to be decreasing more rapidly below 60% whereas *SHUT* and *MIX* appear to be more consistent. The switch-off mechanisms (*SHUT* and *MIX*) seem to be more efficient if we consider the tradeoffs energy/work. This is basically related to the préparation in-advance in the first part of the algorithm and the gained power due to the bonus.

VII. Conclusions and Future Works

We presented here 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 resulted from the design of three powercap policies, namely *DVFS*, *SHUT* and *MIX* which respectively take advantage of CPU Frequency scaling, nodes shutdown and mixed capabilities in order to achieve power reductions whenever needed. The main objective 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. These new developments will be integrated into the main branch of SLURM in the upcoming version 15.08. As far as we know, this is the first work that considers powercapping at the level of job scheduling in HPC. The study allowed us to validate the algorithms and evaluate the different policies through real-scale emulation of an actual petaflopic supercomputer. 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 one 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 tradeoffs could be also determined through a particular profiling run as proposed in [29].

References

[1] S.-g. Kim, C. Choi, H. Eom, H. Yeom, and H. Byun, "Energy-centric DVFS controling method for multi-core platforms," in *SCC*, 2012.

[2] B. Lawson and E. Smirni, "Power-aware resource allocation in high-end systems via online simulation," in *SC*, 2005.

[3] D. C. Snowdon, S. Ruocco, and G. Heiser, "Power management and dynamic voltage scaling: Myths and facts," in *PARTS*, 2005.

[4] R. Schöne and D. Hackenberg, "On-line analysis of hardware performance events for workload characterization and processor frequency scaling decisions," in *ICPE*, 2011.

[5] H. Kimura, M. Sato, T. Imada, and Y. Hotta, "Runtime DVFS control with instrumented code in power-scalable cluster system," in *Cluster*, 2008.

[6] A. Gandhi, M. Harchol-Balter, R. Das, J. O. Kephart, and C. Lefurgy, "Power capping via forced idleness," in *WEED*, 2009.

[7] B. Rountree, D. Lowenthal, M. Schulz, and B. De Supinski, "Practical performance prediction under dynamic voltage frequency scaling," in *IGCC*, 2011.

[8] 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.

[9] 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," *TPDS*, 2007.

[10] D. Aikema, C. Kiddle, and R. Simmonds, "Energy-cost-aware scheduling of HPC workloads," in *WoWMoM*, 2011, pp. 1–7.

[11] J. Hikita, A. Hirano, and H. Nakashima, "Saving 200kw and $200 k/year by power-aware job/machine scheduling," in *IPDPS*, 2008.

[12] E. D. Demaine, M. Ghodsi, M. T. Hajiaghayi, A. S. Sayedi-Roshkhar, and M. Zadimoghaddam, "Scheduling to minimize gaps and power consumption," in *SPAA*, 2007.

[13] 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 *IPDPSW*, 2012.

[14] S. Reda, R. Cochran, and A. K. Coskun, "Adaptive power capping for servers with multithreaded workloads," *IEEE Micro*, 2012.

[15] M. Etinski, J. Corbalan, J. Labarta, and M. Valero, "BSLD threshold driven power management policy for HPC centers," in *IPDPSW*, 2010.

[16] ——, "Optimizing job performance under a given power constraint in HPC centers," in *IGCC*, 2010.

[17] ——, "Parallel job scheduling for power constrained HPC systems," *Parallel Computing*, 2012.

[18] G. A. Geronimo, J. Werner, R. Weingartner, C. B. Westphall, and C. M. Westphall, "Provisioning, resource allocation, and DVFS in green clouds," *IJANS*, 2014.

[19] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *SIGARCH*, 2007.

[20] J.-M. Pierson and H. Casanova, "On the utility of dvfs for power-aware job placement in clusters," in *Euro-Par*, 2011.

[21] A. Mu'alem and D. Feitelson, "Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling," *TPDS*, 2001.

[22] A. B. Yoo, M. A. Jette, and M. Grondona, "SLURM: Simple Linux utility for resource management," in *JSSPP*, 2003.

[23] "http://www.netlib.org/linpack/."

[24] J. D. McCalpin, "A survey of memory bandwidth and machine balance in current high performance computers," *IEEE TCCA Newsletter*, 1995.

[25] "https://software.intel.com/en-us/articles/intel-mpi-benchmarks."

[26] H. J. Berendsen, D. van der Spoel, and R. van Drunen, "Gromacs: A message-passing parallel molecular dynamics implementation," *Computer Physics Communications*, 1995.

[27] Y. Georgiou, T. Cadeau, D. Glesser, D. Auble, M. Jette, and M. Hautreux, "Energy accounting and control with slurm resource and job management system," in *ICDCN*, 2014.

[28] Y. Georgiou and M. Hautreux, "Evaluating scalability and efficiency of the resource and job management system on large hpc clusters," in *JSSPP*, 2012.

[29] A. Auweter, A. Bode, M. Brehm, L. Brochard, N. Hammer, H. Huber, R. Panda, F. Thomas, and T. Wilde, "A case study of energy aware scheduling on supermuc," in *ISC*, 2014.