On the impact of energy-efficient strategies in HPC clusters

Fábio D. Rossi, Miguel G. Xavier, Yuri J. Monti, César A. F. De Rose
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre – RS – Brazil
fabio.diniz@acad.pucrs.br

***Abstract*—Energy-aware management strategies are a recent trend towards achieving energy-efficient computing in HPC clusters. One of the approaches behind those strategies is to apply energy-saving states on idle nodes, alternating them among different sleep states that reflect on many power consumption levels. This paper investigated the way such energy-efficient strategies affected the job turnaround time - the elapsed time between when the job is submitted and when the job is completed, including the wait time as well as the job's actual execution time - in these clusters. Based on the results we proposed a Best-Fit Energy-Aware Strategy that switches the nodes to a sleep state, depending on the throughput of the resource manager's job queue. We simulated the proposed strategy using the SimGrid simulator. Our preliminary results showed a reduction of up to 19% in the overall energy consumption and give us a better understanding of the trade-offs involved in using energy-efficient strategies.**

***Keywords*—Energy-aware management; HPC cluster; performance evaluation; power consumption; sleep states.**

I. Introduction

HPC clusters are composed of multiple computers connected by a network, working together to solve heavy problems in units of time which are impossible in conventional machines. In such clusters, speedup is the main performance metric adopted in order to measure how much a parallel application is faster than a corresponding sequential application. However, with the growth of exascale machines, other metrics regarding energy saving have emerged in the past few years, since the energy costs for cooling servers have been a major concern in large-scale data-centers in developed countries.

In such a way, there has been several studies with the purpose of identifying the trade-off between performance and power consumption [1]. Such studies are focused on the waste of unused resources which is commonly observed in educational institutions clusters [2] [3]. A closer inspection on such clusters reveals a usage rate that goes from 13.8% to 36.3%. As we can see, there are idle nodes in periods of time consuming power needlessly, which could be better managed by using energy-efficient strategies.

Although these strategies allow saving energy when dealing with HPC environments, other metrics should be taken into consideration. Strategies using sleep states incur additional latency in resource manager's job queues, since the requested resources may take longer to become available for allocation. The total time taken between the submission of a job to the queue until it is scheduled and sent back to the user is called job turnaround time and it is a key metric used by resource managers such as PBS/TORQUE[1].

[1]http://www.adaptivecomputing.com/products/open-source/torque/

Most studies in this direction have been performed on low-throughput clusters because they use a large amount of idle nodes allowing the implementation of multiple power-saving strategies. On the other hand, we suppose that the job turnaround time suffers a direct impact. This impact reflects on resource managers of high-throughput clusters, since the job queue might rapidly increase due to the time needed to turn the nodes on. Furthermore, high-throughput clusters tend to increase the energy consumption when their nodes need to be frequently turned off/on. We believe that a Best-Fit Energy-Aware Strategy could choose the best energy state to be applied on the cluster, depending on its usage rate. In addition, previous works do not consider job queues. This is an important point that should be taken into account to avoid wastage in the changes of states, in both time and power.

This paper exposes the advantages of using an energy-efficient strategies for clusters, verifying jobs in a queue which will be submitted to the nodes before turning them off, avoiding reverting states. The findings included in this work are: (1) a comparison of all studied sleep states found in literature, with costs of power and time to change or support these states; (2) an analysis of the impact of these states in job turnaround time and power saving for different workloads; and (3) a new strategy which checks the job queue before putting the idle nodes into a sleep state and chooses the best strategy to use, depending on the cluster usage.

This paper is organized as follows: Section II shows ACPI power-saving states; Preliminary evaluations that served to sustain the metrics analyzed in this paper and the testbed used are discussed in Section III; The Best-Fit Energy-Aware Strategy proposed is presented in Section IV; In Section V is presented the evaluations of proposed strategy; We finish this work in the Section VI with our conclusions and future work.

II. Background

This section presents the states used to save energy in cluster environments. Furthermore, we present related works that used these states in their research.

A. Power Consumption of Machine States

The management of energy states is performed by the operating system (OS) through Advanced Configuration and Power Interface (ACPI). The communication between the OS and the hardware platform is performed by a device driver. Likewise, power management is done by the ACPI driver through a communication between the OS and the hardware platform.

From a user-visible level, the system can be thought of as being in one of the states of the Figure 1. ACPI specifies different levels of states, which are: global states, sleep states, device states, and processor states. Some of these levels comprehend IT resources, such as machines, switches, etc, whereas others represent peripherals, such as the processors.

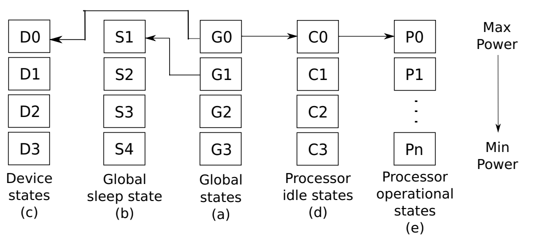


Fig. 1. ACPI States

TABLE I. ACPI STATES

Level	Description
	Global States
G0	threads (system/user) are running (*run*)
G1	the system consumes a small amount of power, user mode threads are not running, the system appears as if turned off, and the system context is saved (*idle*)
G2	the system consumes a minimal amount of power, user mode threads and system processes are not running, and the system context is not saved (*poweroff*)
	Sleep States
S3	CPU, system cache, and chip set context are lost (*standby*)
S4	powered off all devices (*hibernate*)
	Processor Operational States
Pn	CPU frequency scaling (*DVFS*)

The global states (Figure 1 (a)) denote the entire system and are visible only to the user. Sleep states (Figure 1 (b)) are types of states derived from the global states (G1) and are visible only to the system. For example, when the user has pressed the power button. The power states of a particular device (Figure 1 (c)) are usually not visible to the user. Devices may be turned off while the system keeps working, for instance. Processor states (Figure 1 (d)) are states of power consumption within the overall working state (G0). Besides these mentioned states, Dynamic Voltage and Frequency Scaling (DVFS) is the name given by the industry to P-States (Figure 1 (e)). Each level denotes one of all available modern processors' frequencies which in conjunction with the ACPI-based firmware allows adjustment on-the-fly based on the CPU load. Table I shows the levels of depth of all states used in this work, as well as their descriptions. The more deep is the state, the lower is the power consumption and the higher is the latency to return to a working state.

It is remarkable that there is a trade-off between the latency for going into a state and the overall power consumption. The impact of this relationship on real-world scenarios become clearer now. An in-depth study reveals that the contemporary energy-saving strategies do not consider this trade-off and that there are environments where these transitions would have a huge influence on power consumption and user's experience, such as those that the turnaround time is a critical key and should never be exceeded. We claim that these influences would be better comprehended through simulation-based evaluations which enable an analysis of results without any disturbance of the environment.

B. Related Work

Freeh et al. [4] presents a work which uses techniques to reduce the processors' frequency in a cluster to reduce power consumption. The study applied the NAS benchmark suite as workloads to check the power consumption. The results showed an energy saving of about 8%, even though they increased the job execution time in 2.6%. In another work presented by Ge et al. [5], the authors proposed a framework to control the frequency of processors. To confirm its strategy, the NAS benchmark suite was also used. However, the work shows variations in the jobs granularity, reaching 36% reduction of energy consumption in the FT benchmark and 5% of performance overhead in the testbed environment.

The work proposed by Alvarruiz et al. [6] called CLUES uses only the power off state to replace the idle. Their experiments were performed over a period of six months, in a cluster that was idle for more than 80% of the time. When the nodes were in the idle state, there were savings of 3.42%. When these nodes were placed in a power off state, it showed a energy saving of 66.67%. This solution is valid for the cluster used, where the processing load is relatively low with up to 20% of use.

As can be seen, all the states presented provide power-savings, but they directly impact the job turnaround time. No work so far has presented a comparison of all states to energy savings in HPC clusters and its impacts on the jobs execution time.

III. PRELIMINARY EXPERIMENTS

Preliminary experiments were conducted to find out what the behavior of the job turnaround while applying the sleep states on idle nodes, considering different cluster usage rates. To do so, evaluations were performed with usage rates varying from 10% up to 90% using all states presented in Section II. By these rates, it was possible to find the trade-off between power consumption and resource manager performance. An analysis of this behavior derives information for the development of the new energy-aware strategy.

A. Testbed

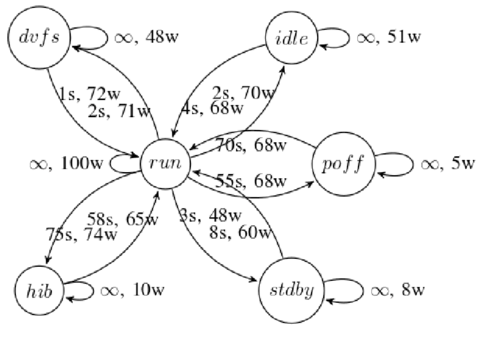
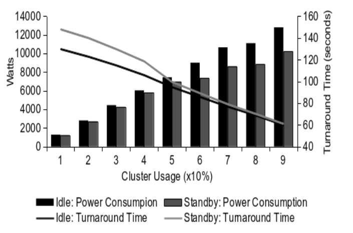
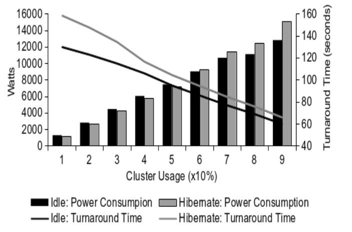


Fig. 2. Power State Transitions

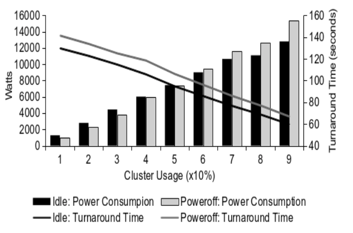
In order to check the trade-off between time and power consumption from all power-saving states transitions, we developed an energy module in SimGrid simulator [7]. We simulated a cluster composed of 128 nodes. The base server node for the simulation consists of two Intel Xeon 2.2Ghz (each processor



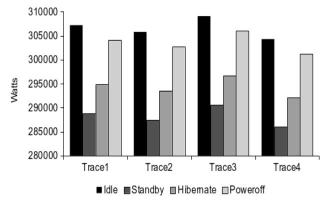
(a) Idle x Standby - Synthetic Workload



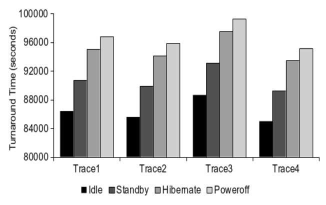
(b) Idle x Hibernate - Synthetic Workload



(c) Idle x Poweroff - Synthetic Workload



(d) Real Workload - Energy Consumption



(e) Real Workload - Turnaround Time

Fig. 3. Preliminary Evaluations

has 2 cores), 2GB of RAM and one Gigabit ethernet adapter. Cluster resources were managed by Conservative Backfilling (CBF) [8]. The difference here lies in the addition of the new power consumption module. Some metrics were taken into account during the building of the new energy consumption module. Such metrics and its transitions are presented in Figure 2. As a comparison factor, the node consumes 100 watts at peak (running state - run). This will be the factor of power consumption to be considered against the measures undertaken to sleep states. Various states of power savings were measured, such as idle ($idle$), the lowest DVFS P-State ($dvfs$), standby ($stdby$), hibernate (hib), and poweroff ($poff$). The values show the required time in seconds for each state to be reached, and the correspondent power consumption. Moreover, in each state, the power consumed while the node remains in this state may also be observed.

B. Results

To simulate a stream of job submissions by users in a HPC cluster, we have applied two different approaches: a synthetic workload based on a widely used model by Lublin et al. [9], which is one of the most comprehensive and validated batch workload models in literature. We chose to use ranges among 10% and 90%, which would enable us to visualize the points of variation in power consumption; and real-world workload traces obtained from the Parallel Workloads Archive[2]. We chose traces from the San Diego Supercomputer Center SP2 (SDSC SP2) [10], which is a well-known and widely studied workload. SDSC SP2 workload has 128 nodes and 73.496 MPI jobs, spanning 2 years from July 1998 to December 2000. In the evaluation, we use four slices of 24 hours each (Trace 1, Trace 2, Trace 3, Trace 4 of SDSC workloads), with usage rates of 55% of the cluster.

Analyses of power consumption and turnaround time were conducted. In our evaluations, we decided to ignore DVFS, due to the proximity of the values of time and power shown between DVFS and idle state in Figure 2. Moreover, this decision is also due to we are simulating an HPC cluster, which should keep the nodes with the highest possible performance. Thus, we take into account only the states of standby, hibernate and poweroff.

Figure 3 presents the preliminary tests we performed in order to view the execution time and power consumption of selected states, in relation to the cluster usage. The first three figures used a workload statistical model based on Lublin et al. [9], while the last two figures used four real traces of SDSC. The standby state shown in Figure 3(a) shows the best relationship between power consumption and execution time, when the use of the cluster is above 50%. This explains why, if we have a high usage rate, the entry of new jobs is more intensive, forcing the nodes at the cluster to change the state faster. As the ratio of power consumption and time to going to and leaves of each state is lower in the standby state, the results are satisfactory above this limit. The poweroff state can be seen in Figure 3(c), and it presents a better relationship between power consumption and execution time when the cluster is at low use, up to 30%. As clusters with low use keep a large amount of idle nodes and the entry of new jobs is not so intensive, the act of turning off the nodes can save enough power, while the time to restart these nodes is not as impactful on total time execution of jobs. The hibernate state is positioned between these two thresholds shown, showing a better relationship between power consumption and execution time of 30% to 50%. As a cluster of average use, this state can balance the best way, issues such as higher entry jobs in the cluster take into account idle nodes in the same time. Figures 3(d) and 3(e) show the four real traces rate of 55% of use. The real traces confirmed the tests with synthetic traces, exhibiting the same behavior for this track usage of the cluster. As the results show well-defined thresholds, we can use these values to create a best-fit energy-aware strategy that allows consuming less power, with less impact on the execution time of jobs. We can summarize these thresholds in Table II.

IV. Best-Fit Energy-Aware Strategy

This section is divided into two parts: the first algorithm shows the way our strategy allocates resources and uses the

[2]http://www.cs.huji.ac.il/labs/parallel/workload/logs.html

TABLE II. USAGE PERCENTAGE OF ENERGY STATES

States	Usage Rate
poweroff	30%
hibernate	50%
standby	over 50%

states of sleep mode; the second algorithm presents the second part of the strategy proposed in this paper, which uses all previous states together.

The Algorithm 1 was developed taking into account the behavior of Conservative Backfilling (CBF) [8] scheduler. The CBF algorithm enables backfilling and it is a well-know representative algorithm running on deploying RMS schedulers today. The main idea of CBF is that an arriving job is always inserted in the first free slot available in the scheduler's queue, which offers an upper-bound to the job start time. Every time a new free slot appears, the scheduler sweeps the entire queue looking for jobs that can be brought forward without delaying the start of any other job in the queue. At this time, the strategy operates by checking if there are enough nodes to support the job execution. If there are enough nodes, the job is put into execution, and the unused nodes are put into a sleep state. If the strategy detects that there are not enough nodes to meet job execution, enough nodes to meet the job request are awake.

Algorithm 1: CBF Improvement

```
$numNodes \longleftarrow nodes\_requested\_by\_a\_new\_job$
$freeSlotList \longleftarrow getFreeSlots()$
for $\forall$freeSlot $\in$ freeSlotList do
    $slotDuration \longleftarrow getSlotDuration(freeSlot)$
    $execTime \longleftarrow estimateJobExecutionTime()$
    if execTime $<=$ slotDuration and numNodes $<=$ freeSlotList then
        scheduling job
        sleeping unused nodes
    if execTime $>=$ slotDuration or numNodes $>=$ freeSlotList then
        awakening enough turned off nodes
        scheduling job
```

The Algorithm 2 presents the choice of which sleep state is the most suitable for the unused nodes, depending on the usage rate of the cluster. The strategy proposed can handle individually, each of the states presented, or manipulate the best-fit energy-aware strategy. We can classify the states into sleep mode levels of intrusiveness, the least intrusive that keeps the node with only some minimal components connected to the total shutdown of the node. This classification is based on the behavior of the states presented in Section III. This means that more intrusive states can save more power when there is a low usage rate of the cluster. However, when the usage rate of the cluster increases, the time wasted to change the idle state to one of the sleep states, plus the power consumption associated with this event, do not allow its use. Therefore, in clusters with a high usage rate, less intrusive states are best. We can classify intrusiveness in the following order, from least intrusive to most intrusive: standby, hibernate and poweroff.

Thus, the Best-Fit Energy-Aware Strategy uses thresholds of the cluster use, to decide the best strategy that will be applied to each new time, depending on the cluster usage rate (by default, for each new job submitted). These thresholds can be seen in Table II. These percentages are based on the primary

Algorithm 2: Power state selection procedure

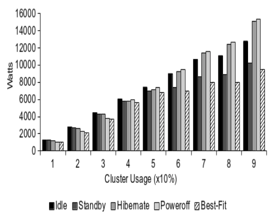
```
for $\forall$Nodes $\in$ Managed physical machines do
    $allNodesList \longleftarrow getAllNodes()$
    $freeNodesList \longleftarrow getFreeNodes()$
    for $\forall$freeNode $\in$ freeNodesList do
        if allNodesList - freeNodeList $<=$ 30% then
            apply poweroff to freeNode
        if allNodesList - freeNodeList $<$ 50% then
            apply hibernate to freeNode
        if allNodesList - freeNodeList $>=$ 50% then
            apply standby to freeNode
```

experiments presented in Section III, which showed the way a state saves more energy, depending on the cluster usage. The Algorithm 2 checks the total number of nodes (*allNodesList*), and the amount of free nodes (*freeNodesList*). Using these values, we can verify the rate of use of the cluster, on-the-fly. Based on this information, the different sleep states can be applied over unused nodes. We can see that the relationship between power consumption and the turnaround time of jobs within the thresholds shown in the table correspond to intervals with the highest usage rate for each strategy. These choices are based on the impact of each sleep state on the turnaround time. The greater the power savings (poweroff), the greater the impact on turnaround time due to the time required for the node is active again. With a low rate of use, this is a technique which fits, because the cluster retains a lot of idle nodes the majority of the time. Already in a cluster with high rate of usage, a technique that allows a rapid return of the nodes to the ready state is deciding factor on turnaround time. The work proposed in this paper differs from the previous as well as proposing a mixed use of energy-saving techniques, besides presenting the smallest possible impact on the turnaround time of jobs.

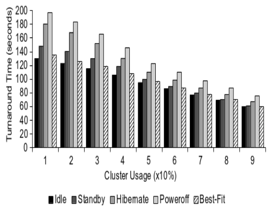
V. EVALUATIONS

When applying our power strategy on the previous scenario showed in Section III, we can see in Figure 4(a) that the results become more expressive in both power-savings and job turnaround time. Thus, when the job run time ends, the node is then released and it becomes available for the next job. It is not directly placed into one of the sleep states. The strategy checks in the job queue if there are workloads that are waiting for resources. In this case, there is no time loss or power consumption increased, to put the node in a sleep state and make it return to a ready state. Therefore, this strategy provides benefits for all states used. For states that are slightly intrusive, power-saving is increased, and for the states that are quite intrusive, allowing power-savings because it has fewer changes between states. Figure 4(b) shows the impact of these changes on the job turnaround time, and we note that in all situations that impact on users jobs time was decreased, because in this scenario, there are fewer context switches between the 'ready to run' state and sleep states. This is also due to the behavior of jobs under the states, because in this scenario, there is less time to be added to the job turnaround time, to shut down and restart nodes.

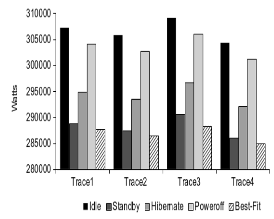
Although all states have taken advantage of the resources allocation, the best usage of this allocation was the Best-Fit Energy-Aware Strategy, which achieved the highest power-savings in all cases. As observed in Figure 4(a), the proposed strategy showed more power-saving than the poweroff while the cluster was under low usage. This happens due to the



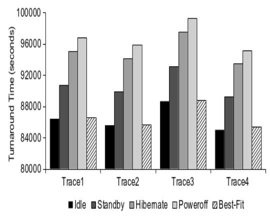
(a) Synthetic Workload: 10% - 90% power consumption



(b) Synthetic Workload: 10% - 90% turnaround time



(c) Real Workload: SDSC power consumption



(d) Real Workload: SDSC turnaround time

Fig. 4. Best-Fit Energy-Aware Strategy Evaluations

dynamicity of the proposed strategy, which always chooses the least intrusive state. When we used our strategy about the scenario of real workloads, we obtained satisfactory results. In Figure 4(c), states that show less intrusiveness have been able to increase power savings, and the states of greatest intrusiveness changed a behavior that wasted power to power-saving too. This shows that the Best-Fit Energy-Aware Strategy proposed in this work can really save power, using the sleep states wisely. As for the turnaround time, the biggest difference can be seen in Figure 4(d). Besides allowing less intrusive states, impacting even less execution time of jobs, it has enabled the turnaround time of more intrusive states fall by half. This shows that our proposed strategy actually performs its main goal, saving power while decreasing the impact on the turnaround time of jobs. The behavior of the Best-Fit Energy-Aware Strategy when we use a real trace presents energy savings and a great correlation with the job turnaround time. We can see in Figure 4(c) that the four tests with the SDSC trace, the Strategy presented the best energy saving. We can see an average savings of 19% on the idle state with and 12% more turnaround time. The results showed that the best state to save energy is in the standby mode. This state allows all hardware components (less memory) to be turned off and allow a quick return to a ready state for a new execution. Hibernate and poweroff are more intrusive states completely turning off the node, leaving only a small hardware implementation checking out a warning to its new restarting. This context switches between turning off and restarting nodes takes much time and power-consuming, so many context switches take precedence over the benefits on these power-saving states. The proposed strategy shows a greater influence precisely in these last two states, since it can make these states save energy even with a high usage rate of the cluster. The turnaround time, a very important factor for the user of an HPC cluster, was also affected by the use of the amount of states, which allowed to keep a low impact at this time. Moreover, the Best-Fit Energy-Aware Strategy proposed balances the power consumption of the cluster as a better strategy can adjust as the usage rate of the cluster. This provides a solution which adjusts the power consumption characteristics of the workload dynamically.

VI. Conclusion and Future Work

In this paper we investigated the impact of power-saving techniques on job turnaround time in HPC clusters, since such techniques might disturb the times while alternating the nodes among different sleep states and even turn machines off for some period of time. Based on these results, we proposed a new strategy which switches from one state to another depending on the workload and on the RMS's job queue. Our preliminary experiments with an enhanced version of the SimGrid simulation tool showed results up to 55% of energy savings with synthetic workloads and 19% with real workloads. As expected, these savings are strongly related to the cluster usage rate. With a light workload, cluster nodes remain idle for longer periods of time so that switching to more economic states bring more energy savings even with the high overhead to enter them. We strongly believe that with this work we contributed to a better understanding of the trade-offs involved in power-saving states after the same approach. In the future, we intend to confirm our experiments with this new strategy with a case study in one of our production clusters.

References

[1] N. Maillard, P. Navaux, and C. De Rose, "Energy-aware scheduling of parallel programs," in *Conferencia Latino Americana de Computación de Alto Rendimiento*, ser. CLCAR, 2010, pp. 95–101.

[2] G. Sabin and P. Sadayappan, "Unfairness metrics for space-sharing parallel job schedulers," in *Proceedings of the 11th international conference on Job Scheduling Strategies for Parallel Processing*, ser. JSSPP'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 238–256.

[3] E. Medernach, "Workload analysis of a cluster in a grid environment," in *Proceedings of the 11th international conference on Job Scheduling Strategies for Parallel Processing*, ser. JSSPP'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 36–61.

[4] V. W. Freeh, N. Kappiah, D. K. Lowenthal, and T. K. Bletsch, "Just-in-time dynamic voltage scaling: Exploiting inter-node slack to save energy in mpi programs," *J. Parallel Distrib. Comput.*, vol. 68, no. 9, pp. 1175–1185, Sep. 2008.

[5] R. Ge, X. Feng, and K. W. Cameron, "Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, ser. SC '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 34–.

[6] F. Alvarruiz, C. de Alfonso, M. Caballer, and V. Hern'ndez, "An energy manager for high performance computer clusters," in *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*, july 2012, pp. 231 –238.

[7] H. Casanova, A. Legrand, and M. Quinson, "Simgrid: A generic framework for large-scale distributed experiments," in *Proceedings of the Tenth International Conference on Computer Modeling and Simulation*, ser. UKSIM '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 126–131.

[8] A. Weil, "Utilization and predictability in scheduling the ibm sp2 with backfilling," in *Proceedings of the 12th. International Parallel Processing Symposium on International Parallel Processing Symposium*, ser. IPPS '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 542–.

[9] U. Lublin and D. G. Feitelson, "The workload on parallel supercomputers: modeling the characteristics of rigid jobs," *J. Parallel Distrib. Comput.*, vol. 63, no. 11, pp. 1105–1122, Nov. 2003.

[10] G. Utrera, S. Tabik, J. Corbalan, and J. Labarta, "A job scheduling approach for multi-core clusters based on virtual malleability," in *Proceedings of the 18th international conference on Parallel Processing*, ser. Euro-Par'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 191–203.