Simple Power-Aware Scheduler to limit power consumption by HPC system within a budget

Deva Bodas, Justin Song, Murali Rajappa, Andy Hoffman
Intel Corporation
Deva.Bodas, Justin.J.Song, Muralidhar.Rajappa, Andy.Hoffman@intel.com

***Abstract*— Future Exascale systems are projected to require tens of megawatts. While facilities must provision sufficient power to realize peak performance, limited power availability will require power capping. Current approaches for power capping limit CPU power state and are agnostic to workload characteristics. Injudicious use of such mechanisms in HPC system can impose a devastating impact on performance. We propose integrating power limiting into a job scheduler. We will describe a power-aware scheduler that monitors power consumption, distributes the power budget to each job, and implements a "uniform frequency" mechanism to limit power. We will compare three implementations of uniform frequency. We will show that power monitoring improves the probability of launching a job earlier, allows a job to run faster, and reduces stranded power. Our data shows that "auto mode" for uniform frequency operates at 40% higher frequency than a fixed frequency mode.**

Keywords Resource manager, scheduler, energy efficiency, power manager, power limiting, HPC, Exascale, IPMI

I. INTRODUCTION

Future HPC systems are expected to deliver Exascale performance with limited power and energy budgets [8]. To address these new challenges, future systems need to operate within a power budget while delivering maximum efficiency and performance. A system resource manager (SRM) that schedules hardware resources for a job will need to consider power as a resource and maximize efficiency of all resources.

A power-aware scheduler will need to maintain system power within limits, maximize energy efficiency, and control the rate of change of system power consumption, while performing all the other functions a scheduler does today. The scheduler will need to forecast how much power the job will need and take corrective action when actual power differs from estimation. Corrective actions cannot be so drastic as to degrade performance or cause wild swings in power consumption. This is a very complex problem to solve [2] especially today when there is little data about power and energy characteristics of HPC workloads. We expect HPC systems to start using a simple power-aware scheduler and evolve in sophistication over the years. We developed such a power-aware scheduler for SLURM[1] that allowed us to study the cost and benefit of various features. We addressed several questions that power-aware schedulers face, and highlighted several issues that need further investigation.

A demand/response interface that is being defined will determine power availability for the entire system. We will need to use this interface to reserve power for jobs that have started and cannot be suspended. The scheduler must possess the ability to manage jobs with and without power limits. A power-aware scheduler needs a way to estimate the power required to run a job. Power-performance calibration of nodes will help develop such an estimate. The estimate is expected to be based upon power-performance data collected on sample workloads or past runs of the job. Although the estimate may have a built-in guard band, actual power consumption of the job is likely to be different. Job-level power monitoring assesses differences between the estimate and actual power consumption. Such assessments create opportunities to fine-tune power allocations to each job. We will show the importance of dynamic monitoring of power consumption on a job-by-job basis.

A power policy is a control mechanism used to ensure that the power consumed by a job stays within its allocation. Power monitoring influences the power policy. Lack of power monitoring requires heavy power allocation guard bands so that the job will never consume more power than the allocation. This heavy allocation will need to be equal to or greater than the maximum power for a worst case (power virus) workload. Heavy allocations have multiple issues. In many cases, fewer jobs will be able to run because the actual power consumption is likely to be far lower than the "power virus" allocation. Therefore, much of the power allocation will be stranded: unused and unavailable. The lack of dynamic power management also hurts performance. Without power monitoring, one will have to assume "power virus" levels for the operating frequency. The selected frequency will be lower than a frequency based upon more realistic power consumption.

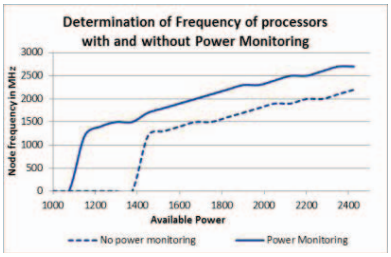

Fig. 1. Impact of power monitoring on startup and operation of a job

1 Other names and brands may be claimed as the property of others.

978-1-4799-7036-0/14 $31.00 © 2014 IEEE
DOI 10.1109/E2SC.2014.8

The example shown in Fig. 1 illustrates these points. The solid line represents frequency capability when power monitoring is used. The dashed line indicates frequency capability when power monitoring is not used. Zero frequency means that the job cannot run. With worst case allocation without monitoring, a job may not be able to start unless available power exceeds 1400 Watts. Power monitoring would provide a more realistic estimate, especially dynamic monitoring which would allow corrective action. In this example the job may start when available power is greater than 1100 Watts. At 1800W availability without monitoring forces 1.5 GHz while monitoring allows 2.2 GHz. Both curves employ a uniform frequency across all the nodes. The solid curve enjoys the benefit of power monitoring while the dashed curve does not. Again, the lack of power monitoring will impose over-estimation. This over-estimation may delay the start of the job until all the estimated power is available. When the job starts, the conservative estimate will force use of a lower frequency of operation. In most cases actual power consumed by a job will be significantly lower, resulting in stranded power.

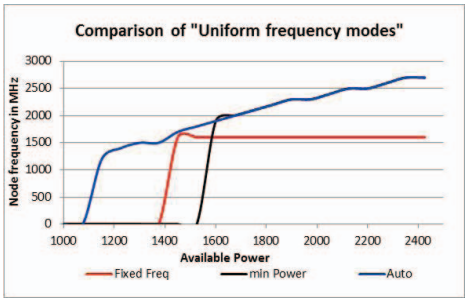

Fig. 2. Characteristics of various uniform frequency modes on scheduling and operation

A key function of a power-aware scheduler is to maintain system power within a limit. There can be any number of schemes to limit power consumption. As a starting point, we implemented a set of schemes we call “uniform frequency”: all processors running the same job operate at the same frequency. Most HPC systems already operate this way. The industry has characterized and optimized many workloads running at a uniform frequency [5]. Uniform frequency power-limiting can be exercised different ways depending on how the frequency is selected and whether the same frequency is maintained throughout the job. We implemented three schemes in SLURM and IPMI daemon for executing a job at a uniform frequency: a) the user selects a frequency of operation for the duration of a job (called fixed frequency mode), b) the user specifies the minimum power level that must be allocated to a job (called "min power" mode), and c) the best frequency is automatically selected based upon available power (called "auto" mode). As Fig. 2 shows, auto mode allows a job to start with less available power. With the auto and min-power modes, a workload manager is allowed to adjust the uniform frequency based upon power headroom.

The rest of this paper will be devoted to discussing the pros and cons of these three schemes.

We believe power-aware scheduling to budget power for jobs will be essential, and dynamic power monitoring is necessary to minimize guard bands. Auto mode relieves users from the burden of choosing an appropriate frequency or power level. By dynamically adjusting frequency based upon power headroom, auto mode can deliver the best uniform performance and minimize stranded power.

II. Related Work

HPC systems have demonstrated limited deployment of power-aware scheduling or methods for limiting system power. In [10] we see a collection of research issues that need to be addressed related to energy management. Simulation studies have been conducted which compare a variety of energy efficient scheduling algorithms to substantiate the benefit of including energy efficiency in scheduling [10]. In some systems, the user submitting a batch job to a work queue is provided an option to select a frequency [13], and in others, user-specified policies for performance based upon a published table of power and performance for each operational frequency are used for energy-aware scheduling [3]. These approaches do not directly address the problem of scheduling jobs within an overall system power limit. Alvarruiz et. al. [4] discusses a framework to turn off idle resources independent of a resource manager. This approach provides a coarse level of power control. It does not address dynamically determining a run-time frequency. A dynamic DVFS mechanism like Intel[2]®’s Enhanced Intel Speedstep Technology [8] could be employed for the jobs that have been selected to run. There have been several studies on the impact of Intel[2]®’s Running Average Power Limit (RAPL) on HPC workloads. For example, Rountree et. al. [12] has documented the performance degradation that arises due to power capping multiple processors while accounting for manufacturing variations. Additionally, specifying a processor power limit leaves the frequency decision in "the wrong hands". It could result in nodes running at different frequencies, incurring performance variation, and hence, degradation. A presentation by Matthieu Hautrexus describes a power-capping exploration for SLURM [10]. His main focus is to shut down nodes instead of leaving them idle to create more power headroom.

III. Simple Power-Aware Scheduler

The future resource manager will not only manage jobs, but also maintain a power budget and manage power-constrained energy efficiencies in real time. The key functions of a power-aware resource manager are:

- Run jobs while maintaining the average power consumption of the system at or slightly below the provisioned power level;
- Maximize performance and energy efficiency of a job by using all of the allocated power;

2 Intel, the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

- Reduce waste by operating unused resources in a sleep state;
- Manage power consumption ramps to a facility specification.

We focused our study on the first aspect: operate jobs while maintaining average system power within a budget. In this section we will describe various terms used in this paper and introduce the essential elements of a power-aware scheduler.

A. Description of terms used in this paper

Throughout this paper we use terms that are expected to convey uniform understanding.

1) Provisioned power to a system (P_{SYS})

P_{SYS} is the power allocation consisting of compute, IO, and OS nodes, network switches, and a storage system. The demand/response interface will eventually determine P_{SYS}. For this paper we control P_{SYS} manually or with a script.

2) Available power for a system

A power-aware scheduler is expected to distribute P_{SYS} among various jobs. Power available for distribution depends upon pre-allocated power and monitoring. Without monitoring: Available power = (P_{SYS} - allocated power). With monitoring: Available power = (P_{SYS} – power consumed by the system – guard band).

3) Platform Max Power (PMP)

When monitoring is not used, the scheduler is forced to allocate power based on the maximum power any job could use. This maximum job power is based on the node's Platform Maximum Power (PMP). PMP is measured by running a “power virus”.

4) Startup power for a job

Any job will need a minimum power allocation, or startup power, to start or resume from suspension. The scheduler estimates the startup power. Without monitoring, the startup power has to be PMP. With monitoring, the startup power can be based upon calibration (see section C.3)a)). When available power is less than startup power, the job cannot start.

5) Minimum required power (MRP) for a job

A scheduler may not be able to suspend or kill certain jobs due to inadequate power. There are two categories of such “special" jobs: jobs with no power limit and jobs that cannot be suspended. A scheduler has to reserve power for such jobs before distributing the remaining power to rest of the jobs. The amount of reserved power for each “special" job is called the Minimum Required Power (MRP). For jobs with a power limit, MRP is either PMP or workload max power. For a job that cannot be suspended the MRP is the power necessary to operate the job at the lowest frequency. MRP is zero for all other jobs. Jobs that run without a power limit are never affected by a reduction in P_{SYS}. Jobs that can be suspended may get suspended when P_{SYS} reduces. Jobs that cannot be suspended may drop to the lowest frequency. P_{SYS} may even drop to such a low level that the system cannot continue to run the “special” jobs. This could happen when the utility reduces its power allocation or a failure occurs in the power-delivery or cooling infrastructure. These can be avoided by using the demand/response interface to communicate the MRP for the system while ensuring high reliability and availability of the infrastructure.

6) Allocated power for a job

The resource manager will allocate a power budget for a job. This allocation does not enforce any hardware power limit. The allocation is strictly used for two purposes: a) to determine available power for the system, and b) to take action for those cases when consumed power significantly differs from allocated power.

7) Stranded Power

Ideally, a job uses the entire power budget for computation. In reality, the consumption may be less. When power is allocated to a job, it is unavailable for other jobs. The difference between allocated and actual consumption is stranded power. Stranded power is unused, unavailable, and wasted. The scheduler must minimize stranded power.

B. Assumptions

Our goal was to explore issues, implementation challenges, and factors that influence various scheduler decisions. For simplicity, we made two assumptions: a) P_{SYS} is provided manually or with a script, and b) compute nodes are single tenant, i.e. all processors in a node run only one job at a time. In a production system, P_{SYS} will need to account for: a) demand/response negotiations, b) losses incurred from inefficiencies, and c) networking and storage. Although a fully-featured production scheduler must consider energy used by all the system elements, we limit our scope to compute nodes. Although multiple jobs run simultaneously, they may have different expectations in terms of importance, priority, run-time, etc. Comprehending these restrictions increases complexity, we chose to limit the variables.

We focused on managing jobs with and without a power limit. We explored the value of job-level dynamic power monitoring. We compared and analyzed the implications of static and dynamic allocations. In addition, we explored effects of jobs that needed to run to completion. We deferred all other improvements and capabilities to future analysis.

C. Essential elements to operate jobs while maintaining a power budget

A power-aware scheduler needs to support a few basic functions mentioned in the sections below and illustrated in Fig. 3.

1) Power scheduling policy

A power-aware scheduler must be guided by an over-arching policy that defines its operation. We crafted the following “max resource utilization” policies for our development. A system administrator could develop other policies that the administrator uses at various times.

a) Maximize utilization of all hardware and software resources. Instead of running fewer jobs at high power and leaving resources unused, we chose to run as many jobs as possible and use all the resources.

b) A job with no power limit is given the highest priority among all running jobs.

c) Suspended jobs are at higher priority for resumption.

2) Determining power for a system (P_{SYS})

In a real implementation, P_{SYS} can be derived from the power allocation to the facility, distribution losses within the facility, voltage conversion losses outside of the server, and power needed to cool the system. For our experiments, the value of P_{SYS} is provided as a scheduler parameter.

3) Estimation of power needed to run a job

Before a scheduler can start a job, it will need to estimate how much power is needed. The estimate is governed by: a) power and performance calibration of a node, b) the ability to monitor job level power, and c) a user- selection of a power policy to limit power for a job.

a) Node calibration

Although an HPC system may use thousands of "identical" nodes, the power and performance characteristics may vary widely between nodes [12]. Variations in hardware and the environment could result in different levels of power consumption of otherwise identical nodes running the same job at the same frequency. Conversely, when hardware power limiting mechanisms force the consumption of each node to be the same, the performance of those nodes may differ. Node-level power and performance calibration enables the scheduler to generate less conservative power estimates for better decisions. Two types of characterizations were performed in our experiments. First, we ran a "power virus" and measured the PMP of each node at each operating frequency. Second, we varied the processor frequency across five representative mini-applications. For each frequency we logged into a database average power, maximum power, power deviation, and time to completion. Node calibration was performed manually. In the future, we expect an application to perform node calibration.

b) Job power estimate

A node calibration database is used to estimate job power. Without dynamic power monitoring, the scheduler has to presume that the job requires PMP. Monitoring provides closed-loop control and a power estimate less than PMP. With power monitoring, even an inflexible policy with limited control knobs can base an estimate on workload maximum power. Flexible controls enable dynamic job power management. Startup power becomes workload average power. The scheduler will also need to estimate the minimum required power (MRP) for jobs that cannot be suspended. The scheduler simply sums the estimates for each node to generate the job estimate. This process can be refined in the future by considering differences between sample and actual workloads.

4) User preference for job power allocation

As users specify job priorities, they also need to specify power and energy policies. In this paper, we are only concerned with power policies. The choices must be self-explanatory and should not require extensive consultation of specifications, manuals, or coding. These are the choices we considered: a) whether or not a job should be subjected to a power limit; b) whether or not the job can be suspended; and c) for a job with a power limit, the user also selects one of three modes to enforce the limit. We will discuss the modes in section IV.

5) Methods to maintain job power within a limit

The user indicates whether job power can be limited. There can be any number of policies and mechanisms to do so. Today many systems run all nodes at an "identical" frequency. We evolved the "identical" or "uniform" method to limit power. The three modes we developed for uniform frequency are explained in section IV. We expect significant enhancement in algorithms and methods over time as the industry gathers and analyzes data from actual runs.

D. How it works

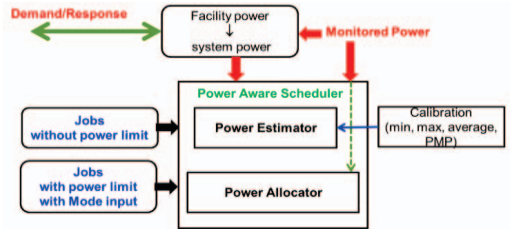

Fig. 3. Overview of power-aware scheduler

Based on user input, the scheduler develops an estimate of power required to start the job (startup power). This estimate is based on node calibration and whether the job is allowed to be suspended. The scheduler then checks for available power. The scheduler starts the job if available power is equal to or greater than startup power. When dynamic monitoring is available, the uniform frequency used by all nodes may be changed periodically based on power headroom. We used a policy that job that started earlier in time has higher priority in using the additional power headroom. Re-evaluations of power budgets and uniform frequency are performed periodically during runtime.

Available power may drop so much that all jobs cannot continue running. In that case the scheduler picks a job at the lowest priority from a list of jobs that can be suspended. Details of this can be found in section E.2).

When the allocated power increases after suspension of any job, we first try to resume a suspended job. This is consistent with our power scheduling policy described in section C.1).

E. Impact of key options in a design

Certain capabilities and features of HPC systems can influence scheduling sophistication and accuracy, as well as subsequent management of job power.

1) Power monitoring for a job

Most HPC clusters today have limited power monitoring. Few system resource managers or workload managers have job power monitoring [3][13]. In the future we expect wide-scale deployment of power monitoring technologies to track and profile job power. The value of power monitoring will influence the rate of adoption. To understand the value, we operated power limiting methods with and without monitoring. Without monitoring, the integrity of the power and cooling system can only be ensured by allocating enough power to a job so that actual consumption will never exceed some hard limit. For such a scenario, estimated job power is based upon PMP. Such a high power allocation exposes a number of efficiency issues. Actual power consumption by nodes in most cases is expected to be lower than PMP. Since the nodes and cluster will consume less than PMP the unused power is stranded. Stranded power is a huge barrier to effectively controlling rate of change of power consumption [9].

2) Ability to pause, suspend and resume a job

The user may specify that a job must run to completion un-interrupted: it must not be suspended. Jobs that do not implement check-pointing generally fall into this category. The scheduler needs to leverage an additional technique to manage such a job. It must estimate MRP for continuous operation. Available power needs to account for MRP. An aggregate of required power in a system must be tracked and communicated via the demand/response interface so that P_{SYS} never falls below the aggregated MRP.

F. Features not considered at this time

Power allocation to jobs, in the future, will be influenced by several other features which will be discussed in subsequent sections. These are areas for future enhancements.

1) Job Priority

When power is scarce, the allocation will need to follow some sort of job priority. A job that was started or scheduled first is assumed to have a higher priority. Clearly this needs to be improved in the future in order to match the job priority specified by the user.

2) Controlling rate of power consumption

Many facilities that host HPC systems prefer that the rate of change of power consumption stay within bounds. Facility managers prefer that all of the allocation be consumed and that stranded power be kept to a minimum. It is also desirable to limit fluctuations in power consumption within specified bounds. A scheduler and workload manager can play an important role in controlling fluctuations.

3) Improving energy efficiency of the system

The scheduler should reduce power used by idle resources to improve energy efficiency and increase available power for computation.

4) Deep examination of job queues

The scheduler should deeply examine the job queue to identify and schedule the best candidates to fit within P_{SYS}. This bolsters our principle of attempting to schedule as many jobs as possible.

IV. THE UNIFORM FREQUENCY POWER LIMIT

When a scheduler starts a job comprised of a fixed set of compute nodes, the job may be subject to minimum and maximum power limits. A workload manager will need to ensure that the job's power consumption stays within the prescribed limits. While power limiting degrades performance, the technical challenge is to keep this degradation minimal or hidden. One can develop an optimal algorithm for managing power and performance for a specific workload, usage model, and constraints. The challenge is to develop an algorithm that applies to a wide range of workloads, usages, and constraints.

Today, most HPC jobs run without any power bound. Furthermore, many workloads are tuned for best performance when all compute nodes operate at a uniform frequency. We focused on an evolutionary approach that enforces a power limit while maintaining a uniform frequency for all processors in a job. There could be a number of ways to determine and select the uniform frequency. Further, the frequency may be static throughout the job, or can be changed dynamically. Based upon these options we developed three methods for comparison. These three methods represent three control mechanisms.

Control systems works best when there is feedback such as power monitoring. Precise control requires sensor accuracy and control resolution. Precise high-resolution sensors allow a reduction in guard-banding. Guard-banding used in power limiting results in stranded power and lower energy efficiency. Any scheduler or workload manager will need to manage a mix of jobs subjected to various (including "no") power limits, and may use different approaches to enforce them. In this section we will detail approaches we used for comparison.

A. Jobs not subjected to a power limit

The user may designate "special" jobs that are not power limited. The scheduler will need to estimate the maximum power the job could consume, and only start the job when the power is available. The workload manager may be redistributing power among "normal" jobs in order to reduce stranded power and maximize efficiency. But even if P_{SYS} falls, the workload manager must ensure that these "special" jobs' power allocations remain intact.

B. Fixed-frequency mode

A user may specify the frequency. User selection may be based upon a table that indicates degradation in performance and reduction in power for each frequency. This approach is used by some systems to find the frequency that delivers the best energy efficiency. A system may use an ACPI P-state table [1] for such a purpose. Once a job starts in this mode, the frequency is never altered. The job does not incur overhead associated with a frequency-shift and is therefore scalable. However, this approach has a number of drawbacks.

1. If the user selects frequency based upon a predefined table purportedly applicable to all nodes, power variations due to manufacturing variances, environmental conditions, and built-in guard bands renders the table inaccurate.
2. The user may select a frequency based upon available power when the job is submitted. Available power at launch is likely to be very different. In a computing environment subject to shifting power availability and other variable conditions, the user lacks the information necessary to select the best frequency.
3. If the available power is lower than estimated power for the selected frequency, the job cannot be started. In some cases it may be possible to run the job at a lower frequency. When the job cannot be started and there are no other jobs in the queue, resources are underutilized. Conversely, if the available power is greater than the estimated power associated with the user selected frequency, the system may waste power as excess power cannot be used to operate the job at a higher frequency than the one selected by the user.

C. min-Power mode

The user may specify the minimum amount of power a job must be allocated. The user may calculate this power level based upon a power-performance table and the requested number of nodes. The scheduler can start a job only if available power equals or exceeds the minimum power. Based upon available power, the scheduler selects the best frequency. When dynamic power monitoring is used, the workload manager may raise or lower the frequency while the job is running based upon increase or decrease in available power. If the available power falls below the specified minimum power threshold, the job will be suspended or terminated.

The main benefit of min-Power mode is that it reduces the burden on the user to guess the right frequency. Secondly, with dynamic power monitoring, the workload manager can improve performance by raising frequency opportunistically. The frequency can be altered based upon power consumed by the workload while running. However, to start a job, the scheduler will need to rely on calibration and estimation of power requirements. A key drawback of min-Power mode is that the user has to calculate minimum power needed for a job using inaccurate tables. A min-Power mode job will get suspended if available power falls below the min-Power level.

D. Auto mode

Auto mode eliminates the need for users to estimate the power or frequency to be used by their job. Uniform frequency selection is automated based upon available power. With dynamic power monitoring, a workload manager will adjust this uniform frequency periodically based upon power headroom. Auto mode allows a job to operate at all available frequencies. Since there is no user-defined minimum job power requirement, the job can start and continue as long as there is enough power to run the job at the lowest frequency. Auto mode reduces the probability of a job waiting for enough power or the job getting suspended due to a reduction in power availability. Auto mode increases resource usage and throughput. You can reduce the power limit and run more jobs to use all hardware resources. We will demonstrate that "uniform frequency" auto mode delivers the best performance and energy efficiency among the three modes we considered.

Besides the fixed frequency and min-Power modes there could be variations using minimum and/or maximum frequency, and minimum and/or maximum power. There could be modes which combine settings for frequency and power. All these require user calculation and experimentation. Auto mode removes this burden and delivers the best performance in a large number of scenarios.

E. Implementation

These policies were implemented using SLURM, an IPMI daemon, and access to processor Model Specific Registers (MSR's). We defined a period to maintain average power for a job, $T_{AVERAGE}$ and a control period, $T_{CONTROL}$. In each control period we re-evaluate power budget, power allocation and frequency selection. We decided that $T_{CONTROL}$ should be $1/10^{th}$ of $T_{AVERAGE}$. $T_{AVERAGE}$ is programmable. Assuming a facility has to maintain an average over 15 minutes, $T_{AVERAGE}$ for jobs ends up being 9 seconds. For min-Power and Auto mode, the control system evaluates every 900 milliseconds whether the uniform frequency should be changed.

The tables below describe various power levels used by the algorithms that implement these modes.

TABLE I. **POWER FREQUENCY SELECTION WITHOUT USE OF POWER MONITORING.**

	Job without power limit	**Fixed-frequency mode User selects F_S.**	**min-Power mode User select P_{MIN}**	**Auto Mode**
Available Power	$P_{AVAILABLE} = P_{SYS}$ – Allocated power			
Startup power	PMP	Workload max @ F_S	P_{MIN}	PMP @ Pn[(b)]
Frequency	Maximum	PMP @ F_S	F_O; Max frequency where PMP $\leq P_{AVAILABLE}$	
Min. required power[(c)]	PMP	PMP @ F_S	= 0 or = P_{MIN}	= 0 or = PMP at Pn.
Allocated power	PMP	PMP @ F_S	Max (Min. required power, PMP @ F_O)	
Dynamic adjustments	No	No	Yes	Yes

TABLE II. **POWER FREQUENCY SELECTION WITH USE OF POWER MONITORING**

	Job without power limit	**Fixed Frequency User selects F_S.**	**min-Power mode User select P_{MIN}**	**Auto mode**
Available Power	$P_{AVAILABLE} = P_{SYS} - P_{CONSUMED}$ – Guard Band			
Startup power	Workload Max + GB[(a)]	Workload max @ F_S	P_{MIN}	Workload Average @ Pn[(b)]
Frequency	Maximum	F_S	F_O; Max frequency where workload average ≤ $P_{AVAILABLE}$	
Min. required power[(c)]	Workload Max + GB	= 0 or = Workload Max at Fs	= 0 or = P_{MIN}	= 0 or = Workload max at Pn.
Allocated power	Workload Max + GB	Workload max @ F_S	Max (Min. required power, Workload average @ F_O)	
Dynamic adjustments	No	No	Yes	Yes

Notes:

[a.] GB: Guard band

[b.] Pn is lowest operation frequency for processors and nodes

[c.] For jobs that can be suspended the min. required power is zero. For others it is defined in the table

V. EXPERIMENTAL SET UP

A. Hardware configuration

We used a cluster of 8 compute nodes shown in Fig. 4 The compute nodes use dual socket Intel[3]® Xeon® E5-2690 (Sandy Bridge EP with a thermal design power (TDP) of 135W). The head node serves as the cluster controller hosting the SLURM resource/workload scheduler. It also serves as the front-end node that is used to start jobs. Infiniband was used to interconnect the nodes.

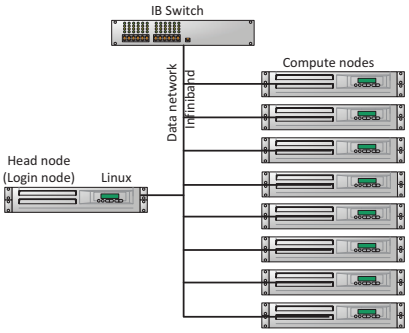

Fig. 4. Hardware configuration for the experiment

B. Software configuration

We used the SLURM resource manager [13], for our experiment (see Fig. 5). we enhanced the node level daemon (slurmd) to gather power information using IPMI. Additionally, it also gets/sets processor register values related to CPU frequency via the OS MSR module. The scontrol command was modified to implement the algorithm described in section IV.E. The modified "scontrol" command logs the power and other job related information needed for our study.

[3] Intel, the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

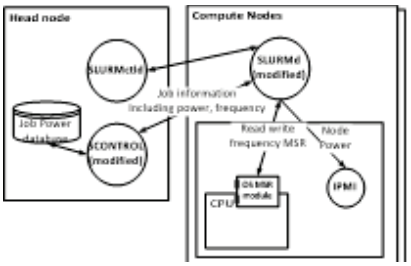

Fig. 5. Software configuration for the experiment

C. Workloads used for analysis

We used a set of HPC benchmarks listed in TABLE III. We selected the workload from list of DOE workloads. [5]. These workloads were chosen as representative of real applications used in traditional HPC environments.

TABLE III. **LIST OF WORKLOADS**

Workload	**Short description**
MCB	Monte Carlo simulation of particle transport
miniFE	Unstructured implicit finite element method
Qbox	Molecular Dynamics
Lulesh	Unstructured shock hydrodynamics

VI. RESULTS

We now examine the results of the performance data collected by running the jobs through the power-aware scheduler using several scenarios as per the modes described earlier with and without power monitoring.

A. Benefit of power monitoring

For the purpose of this analysis, we focus on the results from the MCB workload for illustrative purposes. (Similar results for other workloads will be discussed in subsequent sections.)

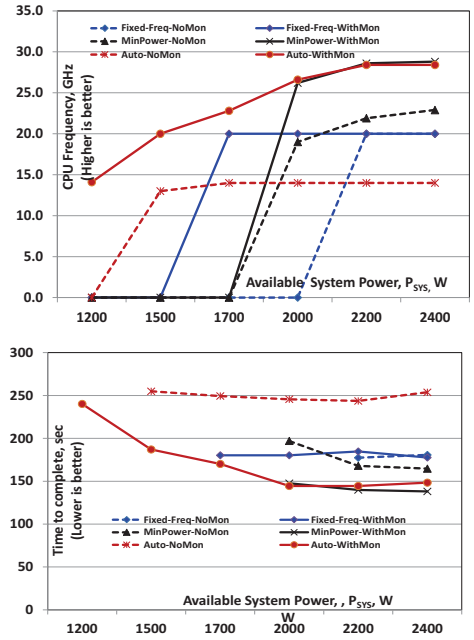

Fig. 6. Comparison of performance of MCB workload with and without power monitoring

In the graph above (Fig. 6), we plotted the performance of MCB against available system power (P_{SYS}) with and without power monitoring[4]. Note that since performance is measured as the wall clock time to complete the job, the lower the number, the better the performance. The solid lines show the performance with monitoring and the dotted lines show the performance without monitoring.

Since the solid lines (with monitoring) are better (lower) than the dotted lines (without monitoring), the resource manager gets better performance with monitoring at all power limits in all modes. The benefit can be up to 40% except in fixed frequency mode where, if the job runs at all, it has to run at the fixed frequency. Hence the two lines almost overlap.

Additionally, the solid lines in all three cases start closer to the Y-Axis than the corresponding dotted lines. This indicates that monitoring enables the scheduler to start jobs with lower system power limits.

With this ample evidence that power monitoring deliver much higher performance for rest of the paper we will show comparisons with power monitoring alone.

B. *Comparison of uniform frequency modes*

We compare the benefits of auto mode over fixed-frequency mode for the set of workloads mentioned in TABLE III. .

[4] Results have been estimated based upon internal Intel analysis and provided for information purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

Fig. 7 indicates ratio of frequency in auto mode to frequency used in fixed frequency mode for the same workload at for various values of P_{SYS}. The range of P_{SYS} was chosen to be 50% to 100% of the unconstrained workload power. We expect that users will not choose to run jobs at power constraints more severe than 50%. auto mode results in up to 40% increase in frequency.

There are no data points between 1200W and 1700W because in that range the jobs can start only in auto. Auto mode obviously outperformed fixed-frequency mode everywhere for all these workloads.

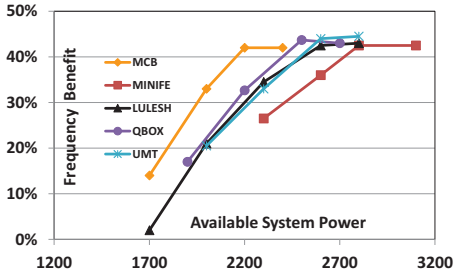

Fig. 7. Performance benefit of auto mode over fixed frequency for various workloads

C. *Mixed mode operation*

We ran three jobs at the same time in different modes. Job1 was a small job that finished first and Job2 and Job3 ran longer. We compared two cases (see TABLE IV.).

TABLE IV. **MIXED MODE WORKLOADS**

P_{SYS} = 1700W	**Job 1**	**Job 2**	**Job 3**
	Lulesh	**Qbox**	**MCB**
	2 Nodes	***2 Nodes***	***4 Nodes***
Case 1	No Power Limit	Fixed-Frequency (2.0 GHz)	Fixed-Frequency (2.1 GHz)
Case 2	No Power Limit	Fixed-Frequency (2.0 GHz)	Auto mode

The results are shown in Fig. 8. Job 3 runs in fixed frequency mode in case 1 and in auto mode in case 2. When Job 1 (Lulesh) completes at 38 seconds, power freed by Job 1 is allocated to Job 3 (MCB-auto) auto mode resulting in higher frequency of 2.9 GHz using additional power headroom. As a result Job3 (MCB-auto) finishes sooner than the same job (MCB-fixed-freq) running in fixed frequency mode as in Case 1.

The corresponding chart in the middle showing the power consumption over the entire run follows a pattern similar to the CPU frequency. The bottom chart shows the total power consumption and the resultant stranded power in the two cases. It should be noted that in case 1, the system does not consume all available power, with a stranded power of up to 620W and the job ends up finishing late. In case 2, by contrast, the system power consumption is steady and is closer to the P_{SYS} value of 1870W (with a stranded power around only 340W) causing the job to finish sooner.

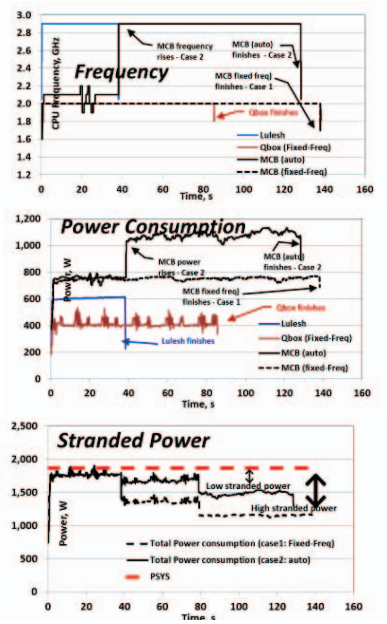

Fig. 8. CPU Frequency and power with multiple simultaneous workloads with mixed modes

D. *Managing time-varying P_{SYS}*

As mentioned in section III.A.1), the amount of total power available to an HPC system from the utility company varies dynamically. To study the behavior of the scheduler in such scenarios, the system power available to the cluster was changed dynamically.

1) *Use of increased available power*

We compared the time for completion of two jobs running simultaneously, one (MCB) running in fixed frequency mode at 2.0 GHz , another (MCB) running in auto mode, when subjected to time-varying P_{SYS} that starts at 1550W and changes between 1550W and 2000 in several steps.

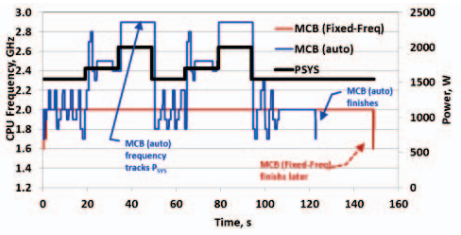

Fig. 9. Faster completion in auto mode with varying P_{SYS}

Fig. 9 shows that the frequency of the nodes running MCB in auto mode increases with additional system power. As a result MCB in auto mode finishes 20% sooner than MCB in fixed-frequency with almost same amount of consumption of energy.

2) *Benefit of not suspending when P_{SYS} decreases*

We ran 2 cases as shown in TABLE V. .Each case has 2 jobs running simultaneously. The difference between the two cases is that Job 2 ran at a fixed frequency of 2.0 GHz in case 1 and ran in auto mode in case 2.

TABLE V. CASES STUDIED FOR TIME-VARYING P_{SYS}

	Job 1	Job 2
	MCB	MCB
	4 Nodes	*4 Nodes*
Case 1	No Power Limit	Fixed-Frequency(2.0 GHz)
Case 2	No Power Limit	Auto mode

The results are shown in Fig. 10 and Fig. 11. The upper sets of charts illustrate case 1 and the lower set of charts illustrates case 2. When the system power is reduced, notice in Fig. 10 that the scheduler has the flexibility to decide the power allocation and frequency of the job only in case 2 (Job 2 running in auto mode). This shows as the benefit that Job 2 in case 2 continues running (at a lower frequency) and completes sooner whereas Job 2 in case 1 is suspended and takes longer to complete.

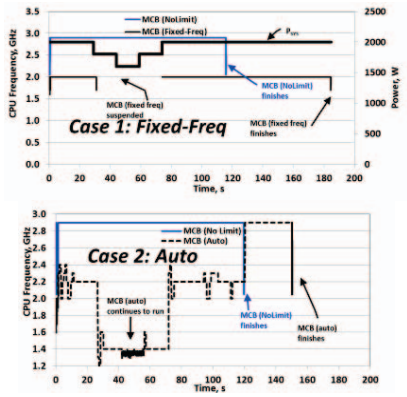

Fig. 10. Effect of time-varying P_{SYS} on frequency

Fig. 11 shows P_{SYS}, actual power consumed, and the stranded power. Note that in case 2, the actual power consumption is closer to the time-varying P_{SYS} and the stranded power also stays close to zero.

Also, note that the stranded power occasionally goes negative for short durations. P_{SYS} is average maintained over relatively long duration while we are monitoring power for much smaller duration. P_{SYS} is expected to be maintained only over a longer time period and small excursions above it are tolerated so long as the longer time-average stays below the specified system power limit.

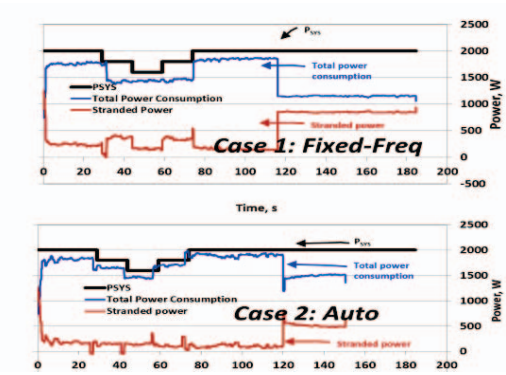

Fig. 11. Effect of time-varying P_{SYS} on system power

We conclude that the scheduler was able to adapt to varying P_{SYS} in auto mode by providing improved performance compared to fixed frequency mode. Based on the above results, we have demonstrated that auto mode with power monitoring provides several superior power-aware scheduling capabilities.

VII. Conclusion, Future Improvements

We demonstrated that job level power monitoring is beneficial to gain higher performance. Auto mode removes the burden of estimating the power consumption or frequency. Auto mode also enables a job to start at lowest available power compared to fixed frequency and min power modes. Auto mode's automatic uniform-frequency adjustment maximizes use of available power consumption. Finally, auto mode gives the fastest time to complete with optimal use of available power.

The tests run on the small cluster need to be expanded to ensure that the dynamic monitoring and controls are scalable to large HPC environments. One can refine control methods by developing schemes to optimize tuning periods for monitoring and control to minimize guard bands.

VIII. Acknowledgment

We would like to thank a number of people who contributed to this paper. We specially acknowledge Jimbo Alexander who gathered the data for several experiments for this paper. We also would like to thank Corey Gough, Sunil Mahawar, David Scott, Tom Spelce, and Matt Tolentino for reviewing this paper and providing critical feedback.

IX. References

[1] Alvarruiz, F., de Alfonso, C., Caballer, M. and Hernández, V. 2012. An Energy Manager for High Performance Computer Clusters. *ISPA '12 Proceedings of the 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*.

[2] Bhattacharya, A. 2013. *Constraints And Techniques For Software Power Management In Production Clusters*. Technical Report No. UCB/EECS-2013-110, Electrical Engineering and Computer Sciences, University of California at Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-110.pdf.

[3] Brehm, M. 2013. *Energy Aware Scheduling SuperMUC@ LRZ*. Application Support Group. Leibniz Supercomputing Centre. http://www.autotune-project.eu/system/files/Matthias_Brehm_Energietag.pdf.

[4] Cai, C., Wang, L., Khan, S. and Tao, J. 2011. Energy-aware High Performance Computing – A Taxonomy Study. *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*. (Tainan, Taiwan. December 07, 2009).

[5] Department of Energy. 2013. *CORAL procurement benchmarks*. LLNL-PRE-637694. (May 31, 2013). https://asc.llnl.gov/CORAL-benchmarks/CORALBenchmarksProcedure-v26.pdf.

[6] Etinski, M., Corbalan, J. and Labarta, J. *Power-Aware Parallel Job Scheduling*. Barcelona Supercomputing Center. http://nsfcac.rutgers.edu/GreenHPC/EEHiPC/eehipc_etinski.pdf.

[7] HP, Intel, Microsoft, Phoenix, Toshiba. 2011. *Advanced Configuration and Power Interface Specification Revision 5.0*. http://www.acpi.info/DOWNLOADS/ACPIspec50.pdf.

[8] Intel® Corp. 2014. *Intel® 64 and IA-32 Architectures Software Developer Manuals*. http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html.

[9] Lefurgy, C., Allen-Ware, M., Carter, J., El-Essawy, W., Felter, W., Ferreira, A., Huang, W., Hylick, A., Keller, T., Rajamani, K., Rawson F. and Rubio, J. 2011. Energy-Efficient Data Centers and Systems. *2011 IEEE International Symposium on Workload Characterization*. (Austin, Texas. November 6, 2011). http://researcher.watson.ibm.com/researcher/files/us-lefurgy/EEDCS_tutorial_IISWC2011.pdf.

[10] Mämmelä, O., Majanen, M., Basmadjian, R., De Meer, H., Giesler, A. and Homberg, W. *Energy-aware job scheduler for high-performance computing*. Computer Science-Research and Development 27, no. 4 (2012): 265-275.

[11] Matthieu, H. Power capping in SLURM. *Green days @ life*, (November 2013).

[12] Rountree, B., Ahn, D., de Supinski, B., Lowenthal, D. and Schulz, M. 2012. Beyond DVFS: A First Look at Performance Under a Hardware-Enforced Power Bound. *8th Workshop on High-Performance, Power-Aware Computing (HPPAC)*. (May 2012). https://e-reports-ext.llnl.gov/pdf/576372.pdf.

[13] 2013. *Slurm Workload Manager*. (November 2013). http://slurm.schedmd.com.

[14] Yoo, A., Jette, M. and Grondona, M. 2003. *SLURM: Simple Linux utility for resource management*. In, Feitelson, D., Rudolph, L. and Schwiegelshohn, U. editors. *Job Scheduling Strategies for Parallel Processing*. 9th Springer Verlag International Workshop. JSSPP 2003 (Seattle June 2003). Lect. Notes Comput. Sci. vol. 2862, pages 44–60.

[15] Zhou, Z., Lan, Z., Tang, W. and Desai, N. 2013. *Reducing Energy Costs for IBM Blue Gene/P via Power-Aware Job Scheduling*. Department of Computer Science, Illinois Institute of Technology; Mathematics and Computer Science Division, Argonne National Laboratory. JSSPP 2013. http://www.cs.huji.ac.il/~feit/parsched/jsspp13/zhou.pdf