Enabling Demand Response for HPC Systems Through Power Capping and Node Scaling

Kishwar Ahmed, Jason Liu
School of Computing and Information Sciences
Florida International University
Emails: {kahme006,liux}@cis.fiu.edu

Kazutomo Yoshii
Mathematics and Computer Science Division
Argonne National Laboratory
Email: kazutomo@mcs.anl.gov

***Abstract*—Demand response is an increasingly popular program ensuring power grid stability during a sudden surge in power demand. We expect high-performance computing (HPC) systems to be valued participants in such program for their massive power consumption. In this paper, we propose an emergency demand-response model exploiting both power capping of HPC systems and node scaling of HPC applications. First, we present power and performance prediction models for HPC systems with only power capping, upon which we propose our demand-response model. We validate the models with real-life measurements of application characteristics. Next, we present models to predict energy-to-solution for HPC applications with different numbers of nodes and power-capping values, and we validate the models. Based on the prediction models, we propose an emergency demand response participation model for HPC systems to determine optimal resource allocation based on power capping and node scaling. Finally, we demonstrate the effectiveness of our proposed demand-response model using real-life measurements and trace data. We show that our approach can reduce energy consumption with only a slight increase in the execution time for HPC applications during critical demand response periods.**

***Index Terms*—Demand Response; High-Performance Computing; Power Capping; Energy Efficiency.**

I. Introduction

Demand response aims at energy reduction during peak electricity periods or other emergency events, and as such provides financial incentives to its participants. Various demand response programs are offered by energy service providers to encourage energy reduction from participants. Among these programs, emergency demand response is most widely adopted, taking up 87% of all the demand response capabilities across the U.S [1]. Emergency demand response requires participants to reduce the energy consumption to requested levels when supply shortage situations or emergency conditions occur (e.g., extremely cold/hot weather, natural disasters). Many electricity markets (e.g., PJM, NYISO, and ISO-NE) serving major states in the United States contribute to power grid stability through demand response participation [2]. The National Institute of Standards and Technology (NIST) and the U.S. Department of Energy (DoE) have both identified demand response as one of the priority goals to achieve power grid efficiency [3], [4]. Many countries, including many EU countries [5], already have a demand response participation program in place [6]. The literature includes a large body of research and field studies of demand response for various sectors, such as data centers and smart buildings (e.g., [7], [8]).

High-performance computing (HPC) systems are generally large infrastructures containing thousands of nodes with a low latency interconnect and distributed file system. Scientific applications with high computation and communication requirements are generally executed on HPC systems. HPC systems can consume an enormous amount of energy during their operation. In the U.S., the Department of Energy has set a limit in the total power consumption of an upcoming exascale system to be within 20 MW. It is projected nevertheless that future HPC systems in many other countries can easily exceed this amount; some systems may even consume hundreds of megawatts of electricity [9]. Apparently, the energy cost is a major component of the overall cost of operation of an HPC system. Any reduction in the electricity bill can be a significant benefit for HPC facilities. Furthermore, the energy consumption of HPC systems can fluctuate drastically due to workload diversity, temperature fluctuation, and dynamic power saving technologies such as clock gating and power gating. Being able to predict and control the massive energy demand can be important for maintaining stability of the energy provider. We argue that by participating in the demand-response program and earning rewards from such participation, HPC systems can both reduce the overall cost of operation and contribute to the power system stability.

To cope with the variations in processing, modern processors are becoming adaptive, providing hardware-level power-capping capabilities that can opportunistically adjust their core frequency based on thermal and energy constraints (e.g., Intel's Turbo Boost Technology). Power-capping capability is becoming a standard feature for modern processors through various programming interfaces, such as Intel's running average power limit (RAPL) [10], AMD's advanced power management link (APML) [11], and NVIDIA's NVIDIA management library (NVML) [12]. New intelligent features have been introduced in processors to achieve energy efficiency through power capping at various locations in the system hierarchy. For example, Intel's Intelligent Power Node Manager and Data Center Manager allow power capping at the node level and component level (e.g., processor, memory) to achieve energy efficiency with different granularity [13].

In this paper, we propose an emergency demand-response model for HPC systems based on power capping and node scaling. Our contributions can be summarized as follows:

- We exploit the power-capping capability in the modern processors to enable HPC system emergency demand response participation. We present prediction models for power and performance prediction with respect to different power-capping values. We propose a demand response participation model for HPC systems based on the prediction models with power capping.
- We extend the HPC system demand-response model by exploiting job malleability. We incorporate an energy-to-solution prediction model to the demand-response model in order to determine the optimal job size and power-capping values.
- We perform experiments using real-life scientific applications on an existing HPC cluster to measure application performance and power usage under different power-capping values. Using these measurements, we use trace-based simulation to show the effectiveness of our proposed demand-response model and compare it with power-capping policies implemented in processors.

The rest of this paper is organized as follows. Section II describes the background and the motivation behind our work, and present work in related areas. Section III describes the energy and performance prediction models and proposes the model for HPC demand response participation exploiting the power-capping property in processors. In Section IV, we present demand-response models that exploit node scaling and power-capping properties. In Section V, we use real-life measurements to compare our approach with traditional methods. Section VI presents our conclusions.

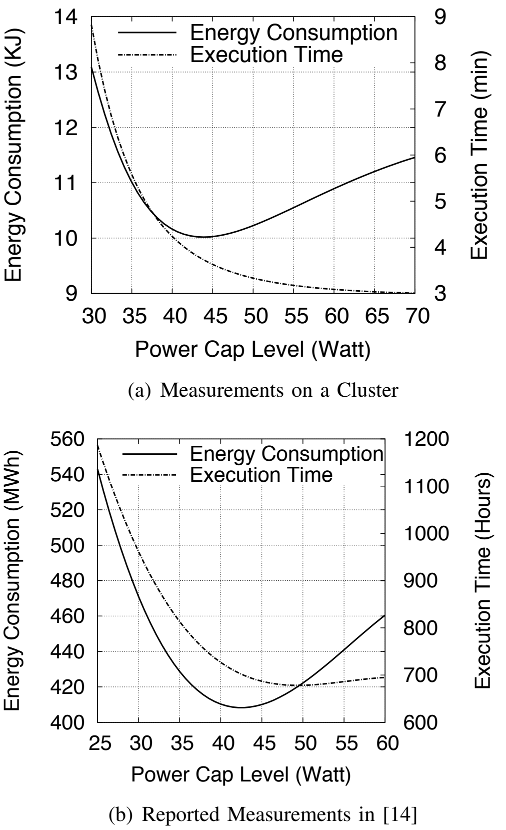
II. Related Work

In this section, we present the background behind our work. We also present related work in the following areas: power allocation methods in HPC systems for performance optimization, demand response in various sectors, and energy prediction in HPC systems.

A. Background

a) Power Capping: Power capping is the allocation of power to nodes mainly to achieve an overall HPC cluster power limit. Power capping not only helps achieve power reduction in the node but also optimizes application performance within a power budget. In this paper, we optimally allocate power to nodes in order to enable the participation of HPC systems in emergency demand-response program.

Power capping can be optimized to control performance and power of applications. With changes in the power limit, application execution time and average power consumption may change, which can impact the energy-to-solution of an application. Fig. 1 shows an example application behavior under different power-capping values. Fig. 1(a) presents our measured energy consumption and execution time for running an HPC application (Intel's DGEMM application from the HPCC suite [15]) on a cluster. As can be observed in the figure, energy consumption has a convex characteristic that can be exploited to achieve optimal energy consumption. We exploit the property to enable demand response participation in this study. Such convexity can also be observed in the literature. For example, Fig. 1(b) presents application characteristics reported in [14], where the scientific applications were collected from the Rodinia benchmark suite and the NPB benchmark suite. As evident from this figure, energy consumption has a convex relation with changes in the power-capping values.



(a) Measurements on a Cluster

(b) Reported Measurements in [14]

Fig. 1. Impact of power capping on application characteristics.

b) Demand Response: Demand response is a program that anticipates customers to reduce energy consumption upon requests from the power utility companies during the time periods of high demand power usage or temporary shortage in power supply. Customers are willing to participate in demand-response programs in expectation to receive financial or operational benefits from the utility companies. Demand response can be broadly categorized into two types: economic demand response and emergency demand response. In economic demand response, participants voluntarily enroll in the programs (without the need of prior commitment) and willingly reduce the load based on economic incentives offered by the supplier. Emergency demand response requires prior commitment from the participants; once enrolled, it is mandatory for the participants to shed loads and collectively prevent the power grid from getting into blackouts, potentially saving billions of dollars' loss. Thus, many demand response resources, e.g., data centers, office buildings, and residential customers, are emerging and sought to participate in emergency demand-response program. Overall, demand response has become increasingly popular among power utility companies. Revenue earnings from demand response has increased significantly in recent years. For example, a report from PJM Interconnection (a large utility company servicing many states in the U.S.) shows that it has achieved an earning of \$650 million from various demand response participation in 2016, a significant increase from only about \$50 million in 2006 [16].

B. Related Work in Power Allocation

Various power-capping-aware methods have been proposed in the literature to optimize power, performance, and energy in HPC systems. Before the emergence of hardware-level power-capping mechanisms, dynamic voltage frequency scaling, idle cycle injection, and clock cycle modulation were popular approaches to limit the power consumption of processors. Hardware-level power capping is the most lucrative choice, since it gives the highest opportunity to save energy [17]. RAPL, an implementation of hardware-level power capping, was first introduced in Sandy Bridge processors [18]. Patki et al. [19] performed an extensive study of application performance for an entire cluster while limiting the power usage at the node level. An optimal power allocation scheme was proposed with consideration of application parallel efficiency and memory intensity to achieve the best application performance. Recently, Liu et al. proposed FastCap [20], a system-wide power-capping approach based on both CPU and memory DVFS to achieve optimal system performance within a given budget for systems with a large number of cores.

Recently, much interest has been shown for job scheduling and resource allocation to HPC jobs in power bound HPC systems. Sarood et al. explored the possibility of exploiting power capping capability of processor and memory subsystems in HPC systems to achieve optimal application execution time within the power budget [21]. They presented an interpolation scheme to estimate application execution time at various processor and memory power levels and later used the prediction to optimize number of nodes and power allocation to nodes by exploiting RAPL capability. In [22], Sarood et al. presented an optimal dynamic resource allocation scheme in HPC systems within a power budget. More specifically, the scheme allocated power and nodes to HPC jobs exploiting resource overprovisioning, power capping, and job malleability properties to maximize job throughput. Unlike these methods, our focus is on enabling emergency demand response participation for HPC systems, although we do exploit the same set of capabilities (power capping and job malleability).

C. Related Work in Demand Response

Demand response participation in various sectors (e.g., data centers, smart buildings) is a well-studied topic. Being a large energy consumer and also being flexible to workload serving requirements, data centers have proved capable of contributing to demand-response programs effectively and quickly (e.g., [7] and references therein). Data center demand response exploits different workload scheduling (e.g., load shifting in time and load balancing among geographical locations) and resource provisioning (e.g., server consolidation, speed scaling) to enable demand response participation. There are also studies enabling demand response participation from smart buildings [8]. However, these approaches are applicable for different types of workloads (such as Internet transaction-based workload) but not for HPC applications. Unlike data center jobs, most HPC applications can tolerate certain delays. Moreover, HPC applications have a unique power and performance signature, whereas data center jobs do not necessarily demonstrate such trait. Therefore, data center demand-response models cannot be directly applied to HPC systems. Recently, we proposed a demand-response-aware model considering frequency scaling for HPC systems [23]. In this paper, we explore HPC system demand response participation using power capping and job malleability properties. In addition, we study power, performance, and energy prediction models for power capping at the per node level and with different job sizes.

D. Related Work in Energy Prediction

A number of analytical models have been introduced to capture the relation between power, energy, performance, etc., for different job sizes. Hager et al. presented an analytical model based on the Execution-Cache-Memory (ECM) model (derived from roofline model) to represent power and performance of multi-core processors [24]. Their approach requires specific information about the applications (such as instruction type) and the platform (e.g., the number of execution ports) in order to perform the prediction. Shoukourian et al. [25] proposed an analytical model, called the Adaptive Energy and Power Consumption Prediction (AEPCP), for application-specific power and energy prediction. Based on historical energy usage by specific applications, their model predicts future power and energy usage. The model can also adapt prediction accuracy continuously with further execution of the applications. Shoukourian et al. extended the AEPCP model and proposed the Lightweight Adaptive Consumption Prediction (LACP) model to predict application execution time, power, energy for different number of nodes and CPU frequency [26]. The LACP model, however, does not predict application characteristics for different power-capping values. Our demand-response model uses HPC application performance prediction models for different power-capping values and node scaling.

III. Demand Response Through Power Capping

In this section, we present the performance prediction models we consider and the optimization problem we implement for demand response participation of HPC systems. We leverage the power-capping property in each node to enable demand response participation.

A. Power and Performance Prediction Models

We first present the power-capping prediction model that we use to predict application behavior under different power-capping values. The average power consumption of job j

running on a processor at power-capping level P can be estimated by the following polynomial function:

$$p(j, P) = a_j + b_j \cdot P + c_j \cdot P^2 + d_j \cdot P^3 \qquad (1)$$

where a_j, b_j, c_j, and d_j are constants determined from empirical analysis of the average power relation with different power-capping values. In particular, a_j represents the static power consumption while running the application.

In a similar approach, we can determine the execution time of job j at power-capping level P using the following equation:

$$t(j, P) = \alpha_j \cdot e^{\beta_j \cdot P} + \gamma_j \qquad (2)$$

where α_j, β_j, and γ_j are regression coefficients determined from polynomial fitting function using empirical data.

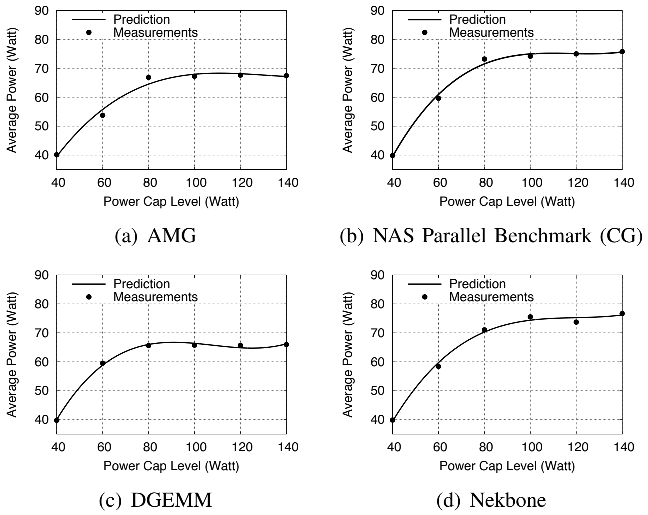
To gain confidence in the proposed models, we present a validation study of the power and execution time prediction models based on real-life measurements of application running on a cluster.

We used a system monitoring/controlling tool, called `pycoolr` [27], to sample per-CPU core temperatures and CPU/DRAM power consumption. The tool uses the Intel RAPL interface to take measurements and reports the results in the JavaScript Object Notation (json) format for later analysis. The tool can also be used to set the upper limit of CPU power consumption. In the validation study, we used `pycoolr` to set the processor's power limit and to characterize the behavior of HPC applications, particularly focusing on performance, temperature change, and actual power consumption with various power-capping values.

Our testbed is an Intel Sandy Bridge node with two Xeon E5-2670 processors, 8 cores, and 16 hardware threads with hyper-threading, which were run at 2.6 GHz, up to 3.3 GHz with the Intel Turbo Boost Technology. The two processors are connected via two Intel Quick Path Interconnect links, which form a cache-coherent NUMA node.

We selected various HPC applications from different sources, such that they can be representative of different application characteristics (e.g., compute-intensity vs. communication-intensity). In particular, we chose the applications from the CORAL benchmarks [28], the NAS Parallel Benchmarks (NPB) [29], and the HPCC Suite [15].

The CORAL initiative is a collaboration among Lawrence Livermore National Laboratory, Oak Ridge National Laboratory, and Argonne National Laboratory and contains a number of HPC benchmarks, representing various DOE applications. We selected applications from the following divisions: scalable science benchmarks (applications expected to run at full scale on the CORAL systems), throughput benchmarks (applications representing large ensemble runs), data-centric benchmarks (applications representing data-intensive workloads, such as integer operations, instruction throughput, and indirect addressing), and skeleton benchmarks (proxy applications that investigate various platform characteristics including network performance, and multithreading overheads.) The following applications were chosen in particular from CORAL: (1) Nekbone, a compute-intensive application supporting various operations (such as MPI Allreduce, vector operations, and matrix-matrix multiplication); (2) LULESH, an application that models hydrodynamics for unstructured meshes and solves a simple Sedov blast problem; (3) Hash, an application that evaluates the performance of architecture integer operations; and (4) XSBench, an application that stresses system through memory capacity.



(a) AMG

(b) NAS Parallel Benchmark (CG)

(c) DGEMM

(d) Nekbone

Fig. 2. Power regression model for different applications.

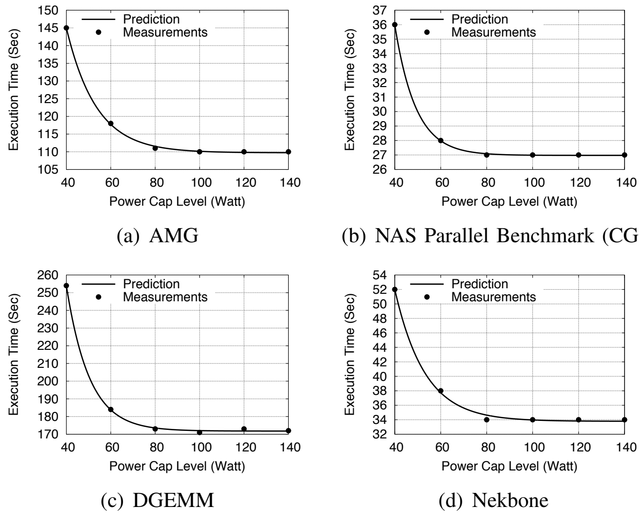
The NPB suite includes a small set of programs designed to help evaluate the performance of parallel applications. From this suite, we selected the CG application (class C), which uses a conjugate gradient algorithm for solving particular systems of linear equations. From the HPCC suite, we selected Intel's DGEMM application for our study. DGEMM is a double-precision general dense-matrix multiply routine in the Intel MKL library. The application is designed to measure the sustained, floating-point computational rate of a single node.

For our validation study, we varied the power-capping level from 40 W to 140 W at the increment of 20 W. We measured the average power usage and execution time of the applications using the `pycoolr` tool. Fig. 2 shows the average power consumption of running different applications: AMG, CG, DGEMM, and Nekbone, respectively. We plot both the measured experiment data and the fitted model data in the same figures. We observe that the prediction model matches with the application power usage generally well. Fig. 3 shows the execution time of different applications with different power-capping values. Similar to the power prediction model, the execution time prediction model is reasonably accurate, as evident from the figure. We later use the measured data and prediction models to demonstrate the effectiveness of our proposed HPC system demand-response model.

B. Determining Optimal Power Cap

Next, we present an optimal power cap allocation algorithm for HPC system demand response participation.

During normal operating time, we set the power-capping value to the maximum limit. This is to ensure that the applica-



(a) AMG

(b) NAS Parallel Benchmark (CG)

(c) DGEMM

(d) Nekbone

Fig. 3. Run-time regression model for different applications.

tions are run with maximum performance, such that demand response participation from HPC system does not impact the original target: to achieve high-performance capability for running the applications. When a demand response event happens, we exploit the power-capping property and select an appropriate power-capping value to reduce energy consumption.

For simplicity, we assume that a job j runs with the same power cap P on all n_j processors, where n_j is the job size. The total energy consumption of the job can be determined as follows:

$$e(j, P) = n_j \cdot p(j, P) \cdot t(j, P) \tag{3}$$

Let P_{min} and P_{max} denote the minimum and maximum power cap allowed by the HPC processor architecture, respectively. We determine the power cap of the processors P_j for job j such that

$$P_{min} \leq P_j \leq P_{max} \tag{4}$$

During the demand response period, we resort to the energy conservation mode by selecting the appropriate power cap values of all running jobs so that we can minimize the overall energy usage. The power cap limit allocation problem can be formulated as an optimization problem, as follows:

$$\begin{aligned} &\text{Minimize:} \sum_{j \in R} e_R(j, P_j) \\ &\text{subject to constraint (4),} \end{aligned} \tag{5}$$

where $e_R(j, P_j)$ denotes the remaining energy expected to be consumed if running job j at power cap P_j, which can be calculated as follows:

$$e_R(j, P_j) = (1 - \alpha_j) \cdot n_j \cdot p(j, P_j) \cdot t(j, P_j) \tag{6}$$

where α_j is the percentage of job j that has been completed thus far. As outlined earlier, the energy consumption of applications at different power-capping values has been in general shown to be convex. We can therefore solve the optimization problem using standard optimization solver and determine optimal power-capping values for each job.

IV. Demand Response Through Power Capping and Node Scaling

In this section, we extend our demand-response model to incorporate node scaling for jobs that can vary the job size (i.e., with job malleability). In the preceding section, we considered only power capping. We do so when one cannot change the job size, that is, when the jobs are specified with a fixed number of nodes upon arrival. In this section, we relax this constraint for malleable jobs and exploit both node-scaling and power-capping capabilities for HPC system demand response participation.

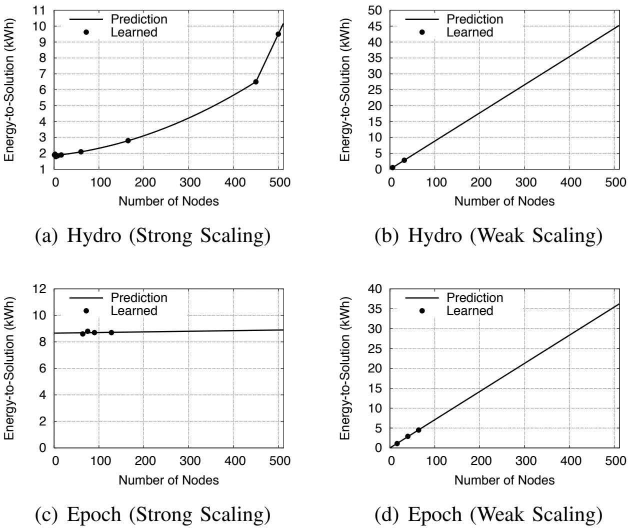
A. Energy-to-Solution Prediction Model

We first present a prediction model that incorporates the effect of both node scaling and power capping for demand response. We consider various regression prediction models (e.g., linear interpolation, spline interpolation) for predicting the energy-to-solution (EtS) of the same HPC applications. We determine the type of predictor to be used in the demand-response model based on the root-mean-square-error value determined from the learned data and predicted data.

We implemented the following five predictor functions: (1) Linear, which captures the linear behavior of predicted function; (2) Spline, which captures the non-linear behavior of the prediction; (3) Linear + Spline, which is a combination that first captures the linear and then the non-linear behavior of the prediction; (4) Spline + Linear, which is a combination that first captures the non-linear behavior of the prediction and then the linear behavior of prediction; and (5) Linear + Spline + Linear, which is a combination that captures the nonlinear behavior in-between the linear behavior predictions at the beginning and at the end. For the combined cases, we determine the boundary value between different prediction regimes (i.e., the points at which transition is made from linear to nonlinear or vice versa). After determining the appropriate prediction function and boundary values, we are able to predict the application behavior for unknown numbers of nodes. Then, we interpolate across known power-capping values to predict the application characteristics for unknown power-capping values.

To validate the EtS prediction model for different job sizes, we collected and used the values reported in [25] for two HPC applications: Hydro [30] and EPOCH [31]. We first consider a strong-scaling scenario for the application Hydro and use the values from [25]. For this case, the data available were for node numbers 1, 2, 4, 8, 16, 165, 450, and 500. Fig. 4(a) shows the measured data points and predicted results for this scenario. We observe that the prediction model correctly predicts a combination of nonlinear and linear predictor functions to be used for prediction, along with the appropriate boundary value (i.e., 450 nodes in this example). The prediction error is low. We make similar observations for Hydro under weak scaling, as shown in Fig. 4(b): the prediction error is reasonably low for this application.

Next, we consider a strong-scaling scenario for the application EPOCH. Fig. 4(c) presents the prediction for EPOCH application with strong scaling. We use the values from [25].



(a) Hydro (Strong Scaling)

(b) Hydro (Weak Scaling)

(c) Epoch (Strong Scaling)

(d) Epoch (Weak Scaling)

Fig. 4. Node scaling model for different applications.

For this case, the data available were for node numbers 64, 75, 90, and 128. Fig. 4(d) presents the measured data points (for node numbers 16, 40, and 64) and predicted results for this scenario. As can be seen from the figures, the prediction model correctly predicts based on the measured data.

B. *Determining Optimal Power Cap and Node Number*

Now, we formulate an optimization problem to determine the optimal power cap and node number for the HPC system emergency demand response participation. The optimal number of nodes and optimal power cap determination problem during demand response periods can be formulated as follows:

$$\begin{aligned}&\text{Minimize:} \sum_{j \in R} e_R(j, P_j, n_j)\\&\text{subject to constraint (4)}\end{aligned} \tag{7}$$

where $e_R(j, P_j, n_j)$ denotes the remaining energy expected to be consumed if running job j at power cap P_j on n_j number of nodes, which can be calculated as follows:

$$e_R(j, P_j, n_j) = (1 - \alpha_j) \cdot n_j \cdot p(j, P_j) \cdot t(j, P_j) \tag{8}$$

We determine n_j and P_j for job j to optimize (7).

V. Performance Evaluation

In this section, we evaluate the performance of our proposed power-capping-based demand-response model. We first present the job scheduler simulator we use for the performance study. Next, we present a performance study of our demand response method with only power capping. Then, we present a performance study with both power-capping and node-scaling capabilities.

A. *Scheduler Simulator*

We developed a job scheduler simulator to simulate HPC jobs and demonstrate effectiveness of our proposed approaches. The scheduler simulator has trace-driven capability to run large numbers of jobs. The simulator provides the flexibility to incorporate new scheduling functions, power-aware methods, and so on. We developed our scheduler simulator based on Simian [32], an open-source, process-oriented parallel discrete-event simulation (PDES) engine. Simian has a minimalistic design, with only 500 lines of code in Python. For some models, Simian has outperformed simulators using compiled languages such as C or C++. Moreover, recently several simulation models have been developed based on Simian, demonstrating the effectiveness of the PDES engine [33], [34], [35], [36]. We validated our scheduler simulator against PYSS, a Python-based scheduler simulator for HPC workloads, which has been used to study various scheduling algorithms in HPC systems and has been used widely in the literature [37], [38].

B. *Performance Evaluation*

Now, we present experiments and results to show the effectiveness of our demand-response model exploiting the power-capping capability. We used two benchmarks:

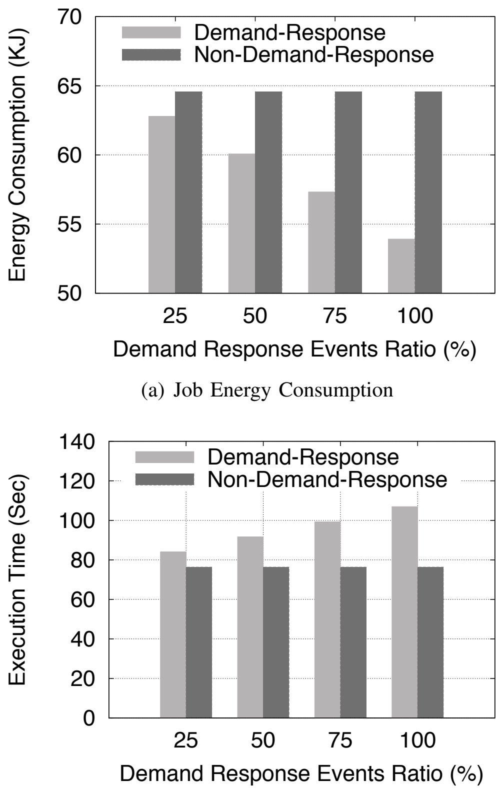
• Demand-Response, which determines the optimal power-capping value based on the optimization problem and solution given in (5) during a demand response period. It chooses the maximum power-capping limit during the normal operating time.

• Non-Demand-Response, which always chooses the maximum power-capping limit to ensure best application performance. We choose this benchmark since it ensures the high-performance requirements of HPC applications. Such policy is also denoted as the default method for power allocation to processors [39].

To evaluate our design, we used a real-life workload trace from the parallel workload archive [40]. The trace was collected at the San Diego Supercomputer Center (SDSC SP2), which contains 5,000 jobs. This trace has been widely used in various studies [41], [42]. The trace includes information about job start time, job run time, job wait time, requested number of processors, and so on. The workload trace, however, does not contain any power-related information. For that, we ran real-life applications on the cluster to measure the power consumption. The details of the applications' power and execution time are given in Section III.

Fig. 5 compares the two benchmarks with different demand response events ratio. We vary the demand response events ratio from 25% (i.e., a demand response event lasts 25% of the entire operation duration) to 100%. As can be seen in Fig. 5(a), Demand-Response achieves reduced per job average energy consumption compared with the Non-Demand-Response benchmark. The energy saving is more pronounced as the demand response events ratio increases. Demand-Response incurs only a moderate increase in the per job average average execution time compared with that of the performance-mode. Since Demand-Response reduces the energy consumption during the critical demand response periods, the increase in application execution time is understandable.

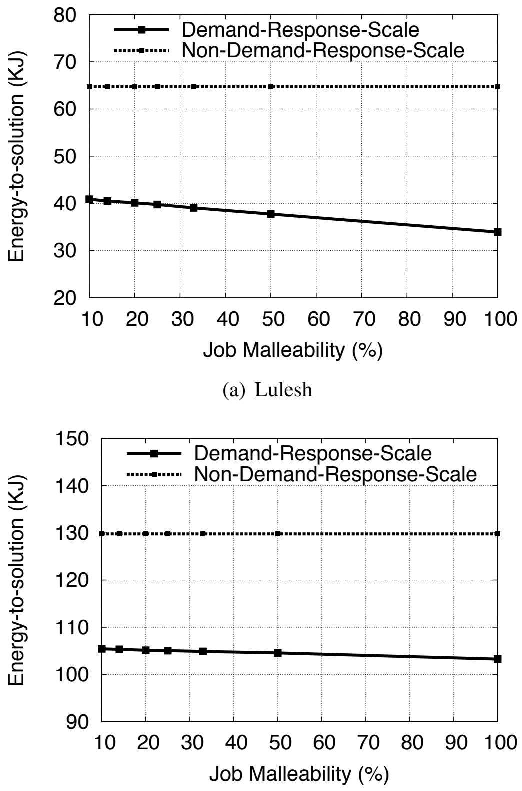
In the rest of the section, we present an experiment to show the effectiveness of the demand-response model with both



(a) Job Energy Consumption

(b) Job Execution Time

Fig. 5. Benchmark comparison with power capping.



(a) Lulesh

(b) AMR

Fig. 6. Benchmark comparison with power capping and node scaling.

power-capping and node-scaling capabilities. We compare the following two benchmarks:

• Demand-Response-Scale, which determines optimal power-capping value and node allocation that reduces EtS. The benchmark solves (7) to determine the optimal resource allocation during the demand response period. The benchmark allocates all available nodes and chooses the maximum power-capping limit on all nodes during the normal operation time.

• Non-Demand-Response-Scale, which always chooses the maximum power-capping limit to ensure the best application performance. It also allocates the maximum number of nodes to the job. Note that, this allocation policy is also chosen in the literature as baseline case [21].

We collected the execution time and energy consumption data from [43] for the following two HPC applications: AMR (an application for adaptive mesh refinement simulations) and LULESH (a shock hydrodynamic modeling application for unstructured meshes). The applications were executed on 4, 8, 12, and 16 nodes. The power-capping values were set to be between 31 W and 55 W with 3 W intervals. We collected the measurements from [43] and used the prediction model in Section IV to predict the application characteristics (i.e., EtS) for unknown node numbers and power-capping values. We used the SDSC SP2 trace from the parallel workload archive [40] for other job-related information (e.g., job start time, job wait time).

Fig. 6 compares the results. Fig. 6(a) shows the effect of running the jobs using the power and performance data of the Lulesh application. Fig. 6(b) shows the effect of using the power and performance data of AMR application. As evident from the figures, the Demand-Response-Scale benchmark always achieves smaller EtS than the Non-Demand-Response-Scale benchmark. This is because Demand-Response-Scale considers optimization of energy consumption during the demand response event through both power capping and node scaling, while Non-Demand-Response-Scale does not consider such optimization. In the figures, job malleability denotes the percentage of jobs that are allowed to change the job size. We change the job malleability from 10% (in which case, only 10% of the jobs can be distributed on an optimal number of nodes) to 100%. As expected, with higher job malleability, more jobs can be flexible in choosing the job size for optimized performance, increasing the opportunity to reduce the energy consumption with our proposed algorithm.

VI. Conclusions

In this paper, we study HPC system demand response participation and propose an emergency demand-response model that leverages power-capping and node-scaling capabilities. We present power, performance, and energy prediction models for HPC applications with unknown power-capping values and job sizes. We validate our prediction models using real-life measurement of application characteristics (including both power and execution time) and compare our models with approaches in literature. We propose an HPC emergency demand-response model by selecting optimal power limit and job size. Using real-life measurements and trace-based data, we examine the effectiveness of our proposed approach and compare it with existing approaches. Our model can effectively reduce the HPC system's energy consumption during critical demand response periods and by doing so enable emergency demand response participation from HPC systems.

Acknowledgments

This material is based upon work supported by an NSF grant CNS-1563883, and by the U.S. Department of Energy Office of Science, under contract DE-AC02-06CH11357.

References

[1] K. Managan, "Demand response: A market overview," 2014.

[2] EnerNOC, "IEA workshop: Demand response," https://www.iea.org/media/workshops/2014/esapworkshopii/Jeff_Renaud.pdf, 2014.

[3] D. G. Holmberg, S. T. Bushby, and D. B. Hardin, "Facility smart grid interface and a demand response conceptual model," NIST, Tech. Rep. NIST Technical Note 1832, 2014.

[4] Federal Energy Regulatory Commission, "Assessment of demand response and advanced metering," https://www.ferc.gov/legal/staff-reports/2016/DR-AM-Report2016.pdf, 2016.

[5] P. Bertoldi, P. Zancanella, and B. Boza-Kiss, "Demand response status in EU member states," *JRC Science for Policy Report, European Comission*, 2016.

[6] S. Zhou, Z. Shu, Y. Gao, H. B. Gooi, S. Chen, and K. Tan, "Demand response program in Singapore's wholesale electricity market," *Electric Power Systems Research*, vol. 142, pp. 279–289, 2017.

[7] A. Wierman, Z. Liu, I. Liu, and H. Mohsenian-Rad, "Opportunities and challenges for data center demand response," in *IGCC*, 2014.

[8] E. Bilgin, M. C. Caramanis, I. C. Paschalidis, and C. G. Cassandras, "Provision of regulation service by smart buildings," *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1683–1693, 2016.

[9] A. Geist and D. A. Reed, "A survey of high-performance computing scaling challenges," *The International Journal of High Performance Computing Applications*, vol. 31, no. 1, pp. 104–113, 2017.

[10] Intel Corporation, "Intel 64 and IA-32 architectures software developers manual," *Combined Volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C*, 2014.

[11] P. G. Howard, "Six-core AMD Opteron processor Istanbul," *white paper, Microway Inc*, 2009.

[12] Nvidia Corporation, "NVML API reference manual," https://docs.nvidia.com/deploy/pdf/NVML_API_Reference_Guide.pdf, 2015.

[13] C. Sahin, F. Cayci, J. Clause, F. Kiamilev, L. Pollock, and K. Winbladh, "Towards power reduction through improved software design," in *Energytech*, 2012.

[14] K. Tang, D. Tiwari, S. Gupta, P. Huang, Q. Lu, C. Engelmann, and X. He, "Power-capping aware checkpointing: On the interplay among power-capping, temperature, reliability, performance, and energy," in *DSN*, 2016.

[15] P. R. Luszczek, D. H. Bailey, J. J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, and D. Takahashi, "The HPC Challenge (HPCC) benchmark suite," in *SC*, 2006.

[16] PJM Interconnection, "Demand response strategy," http://www.pjm.com/~/media/library/reports-notices/demand-response/20170628-pjm-demand-response-strategy.ashx, 2017.

[17] Q. GAO, "Investigation of power capping techniques for better computing energy efficiency," Ph.D. dissertation, Politecnico di Milano, 2014.

[18] B. Rountree, D. H. Ahn, B. R. De Supinski, D. K. Lowenthal, and M. Schulz, "Beyond DVFS: A first look at performance under a hardware-enforced power bound," in *IPDPSW*, 2012.

[19] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. De Supinski, "Exploring hardware overprovisioning in power-constrained, high performance computing," in *SC*, 2013.

[20] Y. Liu, G. Cox, Q. Deng, S. C. Draper, and R. Bianchini, "FastCap: An efficient and fair algorithm for power capping in many-core systems," in *ISPASS*, 2016.

[21] O. Sarood, A. Langer, L. Kalé, B. Rountree, and B. De Supinski, "Optimizing power allocation to CPU and memory subsystems in overprovisioned HPC systems," in *CLUSTER*, 2013.

[22] O. Sarood, A. Langer, A. Gupta, and L. Kale, "Maximizing throughput of overprovisioned HPC data centers under a strict power budget," in *SC*, 2014.

[23] K. Ahmed, J. Liu, and X. Wu, "An energy efficient demand-response model for high performance computing systems," in *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2017, pp. 175–186.

[24] G. Hager, J. Treibig, J. Habich, and G. Wellein, "Exploring performance and power properties of modern multi-core chips via simple machine models," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 2, pp. 189–210, 2016.

[25] H. Shoukourian, T. Wilde, A. Auweter, and A. Bode, "Predicting the energy and power consumption of strong and weak scaling HPC applications," *Supercomputing frontiers and innovations*, vol. 1, no. 2, pp. 20–41, 2014.

[26] H. Shoukourian, T. Wilde, A. Auweter, A. Bode, and D. Tafani, "Predicting energy consumption relevant indicators of strong scaling HPC applications for different compute resource configurations," in *Proceedings of the Symposium on High Performance Computing*. Society for Computer Simulation International, 2015, pp. 115–126.

[27] K. Yoshii, "Python script collection for COOLR," https://github.com/coolr-hpc/pycoolr, 2015.

[28] Oak Ridge National Laboratory, Argonne National Laboratory, Lawrence Livermore National Laboratory, "CORAL benchmark codes," https://asc.llnl.gov/CORAL-benchmarks/, 2014.

[29] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber *et al.*, "The NAS parallel benchmarks summary and preliminary results," in *Supercomputing*, 1991.

[30] P.-F. Lavallée, G. C. de Verdiere, P. Wautelet, D. Lecas, and J.-M. Dupays, "Porting and optimizing HYDRO to new platforms and programming paradigms-lessons learnt," *Technical report, PRACE*, 2012.

[31] T. Arber *et al.*, "Epoch: Extendable pic open collaboration," 2014.

[32] N. Santhi, S. Eidenzenz, and J. Liu, "The Simian concept: Parallel discrete event simulation with interpreted languages," in *Proceedings of the 2015 Winter Simulation Conference*, L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, and M. D. Rossetti, Eds., 2015.

[33] K. Ahmed, J. Liu, A.-H. Badawy, and S. Eidenbenz, "A brief history of hpc simulation and future challenges," in *Simulation Conference (WSC), 2017 Winter*. IEEE, 2017, pp. 419–430.

[34] G. Chapuis, D. Nicholaeff, S. Eidenbenz, and R. S. Pavel, "Predicting performance of smoothed particle hydrodynamics codes at large scales," in *WSC*, 2016.

[35] K. Ahmed, M. Obaida, J. Liu, S. Eidenbenz, N. Santhi, and G. Chapuis, "An integrated interconnection network model for large-scale performance prediction," in *Proceedings of the 2016 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. ACM, 2016, pp. 177–187.

[36] K. Ahmed, J. Liu, S. Eidenbenz, and J. Zerr, "Scalable interconnection network models for rapid performance prediction of HPC applications," in *2016 IEEE 18th International Conference on High Performance Computing and Communications (HPCC)*, Dec 2016, pp. 1069–1078.

[37] Y. Georgiou, D. Glesser, K. Rzadca, and D. Trystram, "A scheduler-level incentive mechanism for energy efficiency in HPC," in *CCGrid*, 2015.

[38] F. Liu and J. B. Weissman, "Elastic job bundling: An adaptive resource request strategy for large-scale parallel applications," in *SC*, 2015.

[39] G. Lawson, V. Sundriyal, M. Sosonkina, and Y. Shen, "Runtime power limiting of parallel applications on Intel Xeon Phi processors," in *Proceedings of the 4th International Workshop on Energy Efficient Supercomputing*, 2016.

[40] D. Feitelson *et al.*, "Parallel workloads archive," http://www.cs.huji.ac.il/labs/parallel/workload/, 2007.

[41] K. Deng, J. Song, K. Ren, and A. Iosup, "Exploring portfolio scheduling for long-term execution of scientific workloads in IaaS clouds," in *SC*, 2013.

[42] K. Kianfar, G. Moslehi, and R. Yahyapour, "A novel metaheuristic algorithm and utility function for QoS based scheduling in user-centric grid systems," *The Journal of Supercomputing*, 2015.

[43] A. Langer, H. Dokania, L. V. Kalé, and U. S. Palekar, "Analyzing energy-time tradeoff in power overprovisioned HPC data centers," in *IPDPSW*, 2015.