Energy-Efficient Heterogeneous Computing of Parallel Applications via Power Capping

Kishwar Ahmed
University of South Carolina Beaufort
Bluffton, SC, USA
ahmedk@uscb.edu

Samia Tasnim
Florida A&M University
Tallahassee, FL, USA
samia.tasnim@famu.edu

Kazutomo Yoshii
Argonne National Laboratory
Lemont, IL, USA
kazutomo@mcs.anl.gov

***Abstract*—High-performance computing (HPC) systems require significant energy to run their operations. A typical HPC system contains various heterogeneous hardware components, including CPUs and GPUs. Power capping is a widely-used feature in processors to achieve an upper limit of power allocation. In this paper, we exploit power capping capability in modern processors to achieve energy-efficiency in heterogeneous computing system. We first develop an optimal power cap allocation model considering heterogeneous computing platform. Next, we develop a simulator based on a parallel discrete-event simulation engine to simulate our proposed power capping allocation model. Finally, we perform trace-based simulation experiments to demonstrate effectiveness of our model. Experiments demonstrate that our proposed model is capable of achieving various level of energy-to-solution reduction for different parallel applications considering heterogeneous computing platform.**

***Index Terms*—Power Capping, High Performance Computing, Energy Efficiency.**

I. Introduction

Rising demand for scientific parallel application has been contributing to increase in demand for large-scale computing, such as high-performance computing (HPC). As such, new HPC systems with more large size and computation power are being introduced. With such increase in size and capability of HPC systems, their energy consumption has also been increasing dramatically over the years. The current top-ranked Fugaku supercomputer in Japan consumes approximately 29MWs of power [1], sufficient to power a small town of 30,000 homes. The 200-petaflops Summit system in the United States consumes more than 10MWs power [1]. All the supercomputers in the top 20 ranks currently consume power in the megawatts range.

To reduce the overall energy consumption, different energy saving techniques have been proposed for HPC systems. They include energy-efficient design for hardware components, including CPU, memory, and interconnection network. Saving energy generally interprets to reducing power consumption, runtime, or both. The methods in the area can be classified into three categories: reduce time and power, reduce time but allow an increase in power, and reduce power while allowing an increase in time. Dynamic voltage and frequency scaling (DVFS) is a widely-studied resource allocation technique for HPC systems [2]–[5]. A more recent effort on DVFS by Bao et al. [6] automatically selects the optimal frequency and core count at compile-time to achieve lower energy. There are also strategies to exploit variations in electricity price, carbon intensity and renewable energy [7]. Different job scheduling approaches have also been proposed in the literature to achieve lower energy consumption [8]–[12]. In this paper, we exploit power capping capability to achieve energy-efficiency in heterogeneous HPC platform.

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. Power-capping capability is becoming a standard feature for modern processors through various programming interfaces, such as Intel's running average power limit, AMD's advanced power management link, NVIDIA's NVIDIA management library, HP dynamic power capping, among others. A number of power allocation methods have been developed to achieve optimized power and performance for HPC systems (e.g., [8], [13]–[16]). In this paper, we develop power capping allocation model and simulator to achieve energy-efficiency for HPC systems considering heterogeneous computing platform. More specifically, our contributions can be summarized as follows:

- We present an optimal energy-efficient power cap allocation model for HPC platform consisting of CPUs and GPUs. We develop a simulator based on a parallel discrete-event simulation engine. The parallel capability of our simulator can be used scale to large number of job simulation.
- We perform trace-based simulation study to show effectiveness of our proposed model. We collect various power and performance data for CPU and GPU node running various parallel application. The simulation results demonstrates capability of achieving energy-efficiency in an HPC platform using our proposed model.

The rest of this paper is organized as follows. In Section II, we present the related studies that are most pertinent to this work. Section III presents the proposed model for energy-efficient power-capping of parallel applications considering heterogeneous computing platform. In the same section, we present a job scheduling simulator based on parallel discrete event simulation engine. In Section IV, we present a trace-based simulation study. Section V concludes our paper.

II. Related Work

In this section, we present related work in the following areas: energy saving methods in HPC, power allocation approaches, and various job simulation approaches in HPC.

A. Energy Saving Methods in HPC

Different analytical models have been developed to achieve energy-efficiency of HPC systems. Song et al. [17] developed one of the first analytical models to help users explore different system parameters (e.g., processor count, processor frequency, workload size) for energy-efficiency of parallel applications. The iso-energy-efficiency model in the paper is used to derive essential machine and application dependent parameters for accurate total system energy consumption, as well as maintain energy-efficiency of the system. An execution-cache-memory (ECM) based model is proposed and developed by Hoffman et al. [18]. The model combines both code-level analysis and hardware-level optimization to achieve energy reduction at both single-core and multi-core processor chips. Endrei et al. [19] developed a statistical model for to predict the Pareto optimal performance parameters and energy efficiency trade-off options that users can directly control. The model has shown capable of accurately capturing the performance-energy trade-offs, improving the overall energy-efficiency of the system, while incurring moderate performance loss.

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 [16]. 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 [8], 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. DVFS has been widely explored as a resource allocation technique for parallel applications. There are early efforts for DVFS-based HPC energy-efficiency (e.g., [3], [5]), as well some more recent efforts(e.g., [6], [20]).

B. Power Capping Allocation Model

Modern processors are equipped with power control systems to ensure that the processor operates within a power cap limit. Various power capping based software solutions have been proposed in the literature [13]–[15], [21]–[26]. Cochran et al. [23] proposed Pack & Cap, a technique for increasing the application performance within the power cap limit for multithreaded workloads on multicore processors. Thread packing is utilized to explore feasible power caps and to enable fine-grained dynamic control of power consumption. A thermal-aware power-capping allocation model for HPC application specific to homogeneous computing architecture has been developed in the paper [26]. RAPL, an implementation of hardware-level power capping, was first introduced in Sandy Bridge processors [13]. Patki et al. [14] 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 [15], 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. In this paper, we develop a power capping allocation model considering heterogeneous computing architecture, and develop a simulator for model simulation.

C. Job Scheduling Simulation

There exist many job scheduling and resource allocation simulators particularly focusing on HPC systems. For example, PYSS (Python Scheduler Simulator) is an open-source HPC workload scheduling simulator written in Python [27]. The simulator was developed by the Experimental System Lab at the Hebrew University, and has been used to study various scheduling algorithms (e.g., [28], [29]). The simulator implements a number of scheduling algorithms, including several backfilling algorithms. The simulator does not really model the target HPC system in detail. CQSim is another event-based simulator to study the detailed queuing behavior of job schedulers using real system workload. The simulator was developed by Illinois Institute of Technology and has been used to evaluate fault-aware utility-based job scheduling [30], adaptive metric-aware job scheduling [31], and so on. Current HPC simulators provide only limited capabilities for studying job scheduling. For example, SST/Macro contains only limited support for running multiple jobs via trace replay [32].

Our job scheduler simulator is developed based on Simulus, which is an open-source, process-oriented, parallel discrete-event simulation engine [33]. Simulus supports both event-driven and process-oriented simulation world-views. It is implemented in Python with several advanced features to ease modeling and simulation tasks with both events and processes.

III. Model

In this section, we present a power capping allocation model considering heterogeneous computing architecture. We first outline the job scheduling and resource allocation model, and then present a problem formulation along with a algorithm framework to solve the problem. Finally, we present a brief description of the simulator we develop based on a parallel discrete-event simulation engine. The simulator is later used to simulate our proposed power cap allocation model.

A. Job Scheduling and Resource Allocation Model

We assume that an HPC user submits job j requesting n_j nodes, consisting of both CPUs and GPUs. We focus on the node power consumption, while the power consumption of non-IT parts of the HPC center such as cooling and power supply system is captured using the factor of power usage effectiveness (PUE). We primarily use the first-come-first-serve (FCFS) policy, although other job scheduling policies can be applied as well.

1) Separate CPU and GPU power-cap model: In this model, we assume that power capping can be allocated separately for CPU and GPU components of a node. We assume that a job j submitted by user runs on GPU at power capping value $P_{j,gpu}$. We denote the GPU's power consumption for running job j as $p_{j,gpu}$. We determine $p_{j,gpu}$ using the following third-order polynomial function:

$$p_{j,gpu} = a_{gpu} + b_{gpu} \cdot P_{j,gpu} + c_{gpu} \cdot P_{j,gpu}^2 + d_{gpu} \cdot P_{j,gpu}^3, \quad (1)$$

where a_{gpu}, b_{gpu}, c_{gpu}, and d_{gpu} are constants determined from empirical analysis of average power relation with different power-capping values. The power-capping value assigned to a GPU should be within a minimum and maximum power-capping limit. The constraint can be represented as follows:

$$P_{gpu}^{min} \leq P_{j,gpu} \leq P_{gpu}^{max}, \quad (2)$$

where P_{gpu}^{min} and P_{gpu}^{max} are the minimum and maximum power-capping values that can be assigned to a GPU. In a similar fashion, we model the CPU's power consumption $p_{j,cpu}$ running at power capping value $P_{j,cpu}$ using the following third-order polynomial function:

$$p_{j,cpu} = a_{cpu} + b_{cpu} \cdot P_{j,cpu} + c_{cpu} \cdot P_{j,cpu}^2 + d_{cpu} \cdot P_{j,cpu}^3, \quad (3)$$

where a_{cpu}, b_{cpu}, c_{cpu}, and d_{cpu} are constants determined from empirical analysis of average power relation with different power-capping values. Moreover, the power-capping value assigned to a CPU should be within a minimum and maximum power-capping limit. The constraint can be represented as follows:

$$P_{cpu}^{min} \leq P_{j,cpu} \leq P_{cpu}^{max}, \quad (4)$$

where P_{cpu}^{min} and P_{cpu}^{max} are the minimum and maximum power-capping values that can be assigned to a CPU. Total dynamic power consumption of the node can be represented as follows:

$$p_{j,node} = p_{j,gpu} + p_{j,cpu}. \quad (5)$$

We assume the time taken for the job to execute is represented by t_j. The total energy consumption of the job can be determined as follows:

$$e_{j,node} = n_j \cdot p_{j,node} \cdot t_j. \quad (6)$$

We determine the separate power-cap values for CPU and GPU based on the following optimization objective.

$$\begin{gathered} \text{Minimize} \sum_{j=1}^{N} e_{j,node} \\ \text{Subject to constraints (2) and (4).} \end{gathered} \quad (7)$$

2) Entire node power-cap model: In this model, we assume that same power capping value is allocated to the entire node. We assume that a job j submitted by user runs on a node consisting of both CPU and GPU. The node runs at power capping value P_j. We denote the node's power consumption for running job j as p_j. We determine p_j using the following third-order polynomial function:

$$p_j = a + b \cdot P_j + c \cdot P_j^2 + d \cdot P_j^3, \quad (8)$$

where a, b, c, and d are constants determined from empirical analysis of average power relation with different power-capping values. The power-capping value assigned to a GPU should be within a minimum and maximum power-capping limit. The constraint can be represented as follows:

$$P^{min} \leq P_j \leq P^{max}, \quad (9)$$

where P^{min} and P^{max} are the minimum and maximum power-capping values that can be assigned to the node.

We assume the time taken for the job to execute is represented by t_j. The total energy consumption of the job can be determined as follows:

$$e_j = n_j \cdot p_j \cdot t_j. \quad (10)$$

We determine optimal power capping values for the entire node based on the following optimization objective.

$$\begin{gathered} \text{Minimize} \sum_{j=1}^{N} e_j \\ \text{Subject to constraint (9).} \end{gathered} \quad (11)$$

We resort to standard optimization algorithm (e.g., sequential least squares programming algorithm using the Han-Powell quasi-Newton method) to solve the optimization problems given in Eqs. 7 and 11.

B. Simulation Model

We use simulation to study the effect of the proposed job scheduling and resource allocation model. Our trace-driven simulator is developed based on Simulus, which is an open-source, process-oriented, parallel discrete-event simulation engine [33]. Simulus has several unique design features that make it more attractive for us to build our scheduler simulator. Simulus has a very simple application programming interface (API). The simulator adopts a minimalistic design with only a handful of core functions. Simulus also supports process-oriented world view for easy model development. Simulus is developed using interpreted languages, such as Python. Simulus has previously been used for trace-driven simulation study of HPC applications [34].

The job scheduler simulator consists of five major components: a job dispatcher, a job executioner, scheduling policies, application models, and a resource manager. The job dispatcher takes four different types of events: job arrival, job departure, job eviction (when an executed job is interrupted and removed in the middle of the run), and power demand change (when the power service provider of the HPC center changes the current power limit due to any emergency event). When a job is submitted, it enters the job waiting queue and invokes the job dispatcher. The job dispatcher determines whether the job is eligible to run according to the application models (that describe the job's power and performance characteristics) and the current available resources from the resource manager. The job dispatcher processes the jobs from the waiting queue according to the scheduling policies. For this study, we use the FCFS policy.

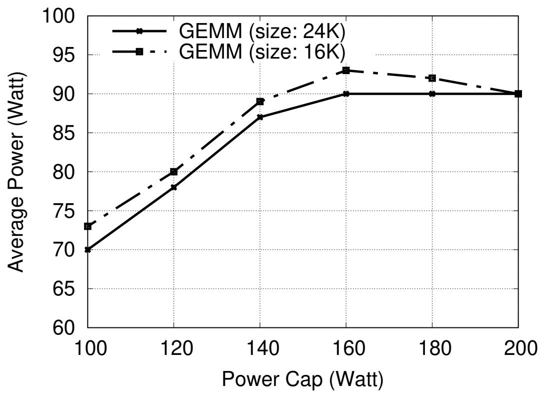
When a job is scheduled to run, the job dispatcher removes the job from the waiting queue and put it in the list of running jobs. The job executioner allocates the resources using the resource manager to represent the occupied processors (at the specified power-capping values) with associated power consumption for running the job. The job executioner then simulates the job's execution accordingly. We simulate using the job's execution time and power consumption according to the estimates from the power and performance models. When a job completes its execution, the job executioner removes the job from the list of running jobs, reclaims the resources occupied by the job, and then invokes the job dispatcher to select new eligible jobs to run.

IV. Experiments

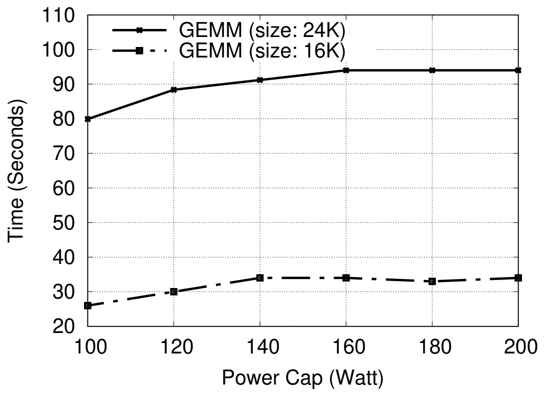
We collect and use jobs' power-related information from literature. More specifically, we choose the published data from an existing study [35]. We collect the power and performance data for various parallel application at different power capping values. The testbed used for the experiments contains NVidia GeForce GTX 1070 GPU and Intel Core i7-7700 CPU. The applications used were general matrix multiplication (GEMM) and NAS parallel benchmark (NPB). GEMM was executed for various matrix sizes, including sizes 16384×16384 and 24576×24576. The NPB benchmark included the following kernels: tri-diagonal solver (BT), scalar Penta-diagonal solver (SP) and lower-upper Gauss-Seidel solver (LU). We collect the execution time, average power data for various power-capping values.

Fig. 1 presents the results reported for various power-capping values for GEMM application. The reported value of power-capping was changed from 100W to 200W at a 20W increase. Two variations of the applications are shown in the figure based on two variation of matrix sizes. Fig. 1(a) presents the average power measurements for the application. With increase in power capping limit, the average power consumption increase until it becomes saturated with increase in power capping value. Fig. 1(b) presents the execution time measurements for the application.

Fig. 2 presents the results reported for different power-capping value for the application NPB. The reported value of power-capping was also changed from 100W to 200W at a 20W increase for this application. Three kernels of the applications are reported in the figure: SP, BT, and LU. Fig. 2(a) presents the average power measurements for the application for different kernels. With increase in power capping limit the average power consumption also increases. Fig. 1(b)



(a) Average Power



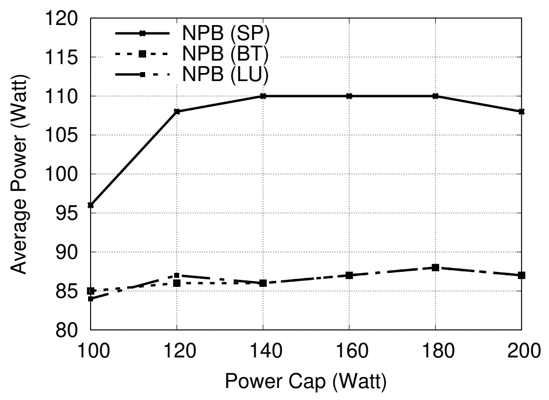
(b) Execution Time

Fig. 1. GEMM application performance on NVIDIA GeForce GTX 1070.

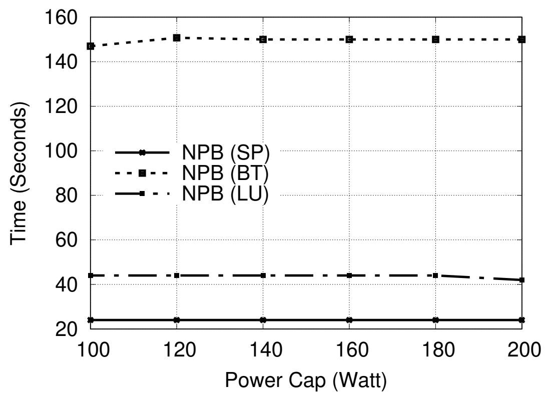
presents the execution time measurements for the application at different power-capping limit.

Fig. 3 presents a comparison between two benchmarks: *Optimal* and *Non-Optimal*. In the algorithm *Optimal*, we allocate optimal power-capping value to the node based on the optimization problem formulation in Eq. 11. In the algorithm *Non-Optimal*, maximal power-capping value is allocated to each application. As can be seen in the figure, various level of energy consumption reduction is possible for all the applications when considering optimal power-capping allocation. More specifically, for the application GEMM, 38.9% and 33.7% energy saving is possible for matrix size 16K and 24K, respectively. Therefore, selecting an appropriate level of power-cap value on a heterogeneous computing architecture can be energy-efficient for various HPC applications.

Next, we collect application power and performance measurement data from existing study [35] for a different node. The node contains Intel Xeon processor and NVIDIA GeForce RTX 2080 GPU. The data for GEMM application for various matrix sizes (16K, 24K and 32K) are shown in Fig. 4. For this reported data, the power capping values were changed from 140W to 240W at 20W granularity. Fig. 4(a) shows the

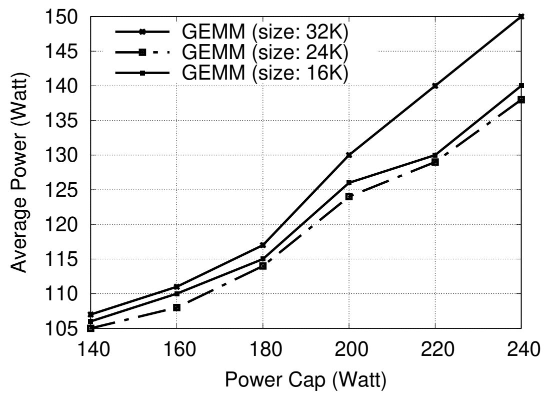


(a) Average Power

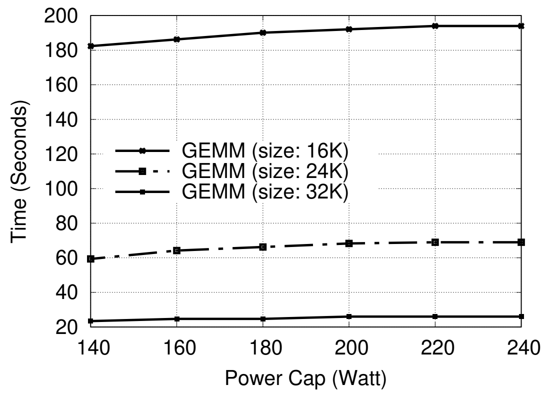


(b) Execution Time

Fig. 2. NPB application performance on NVIDIA GeForce GTX 1070.



(a) Average Power



(b) Execution Time

Fig. 4. GEMM application performance on NVIDIA GeForce RTX 2080.

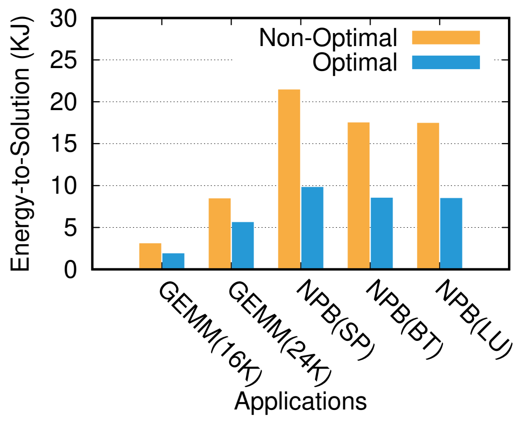


Fig. 3. Energy-to-Solution for applications on GTX node.

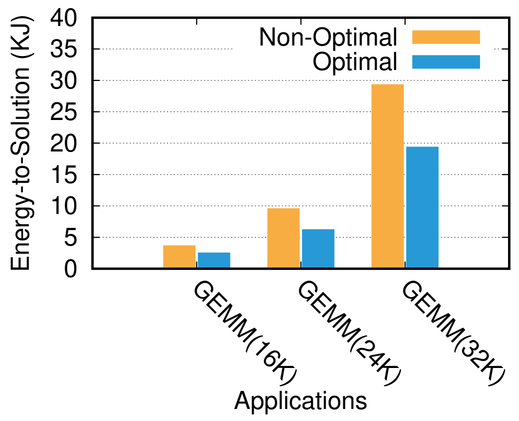


Fig. 5. Energy-to-Solution for applications on RTX node.

average power for different power cap values, while Fig. 4(b) presents the application time execution for various power cap values. Fig. 5 presents the comparison of the *Optimal* and *Non-Optimal* benchmarks for these measurements. As evident from the figure, our proposed model can achieve reduced energy-to-solution for the application while considering a different set of node consisting of CPUs and GPUs.

V. Conclusions

In this paper, we develop a power cap allocation model for parallel application execution on heterogeneous computing platform, consisting of CPU and GPU. We develop a simulator based on a parallel discrete simulation engine. The simulator includes our proposed power cap model. We collect various trace data to demonstrate effectiveness of our model. Our collected data consist of number of HPC application running on heterogeneous compute node. Simulation experiments show that our model can achieve energy-to-solution reduction at various level for different parallel applications on heterogeneous computing platform.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation EPSCoR Program under NSF Award # OIA-1655740 and an ASPIRE grant from the Office of the Vice President for Research at the University of South Carolina.

REFERENCES

[1] TOP500.org, "Top 500 list," https://www.top500.org/, 2019, accessed 14th July 2020.

[2] R. Ge, X. Feng, W.-c. Feng, and K. W. Cameron, "CPU MISER: A performance-directed, run-time system for power-aware clusters," in *Parallel Processing, 2007. ICPP 2007. International Conference on*. IEEE, 2007, pp. 18–18.

[3] J. Li and J. F. Martinez, "Dynamic power-performance adaptation of parallel computation on chip multiprocessors," in *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*. IEEE, 2006, pp. 77–87.

[4] B. Rountree, D. K. Lownenthal, B. R. De Supinski, M. Schulz, V. W. Freeh, and T. Bletsch, "Adagio: making dvs practical for complex HPC applications," in *Proceedings of the 23rd international conference on Supercomputing*. ACM, 2009, pp. 460–469.

[5] V. W. Freeh, F. Pan, N. Kappiah, D. K. Lowenthal, and R. Springer, "Exploring the energy-time tradeoff in MPI programs on a power-scalable cluster," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*. IEEE, 2005, pp. 10–pp.

[6] W. Bao, C. Hong, S. Chunduri, S. Krishnamoorthy, L.-N. Pouchet, F. Rastello, and P. Sadayappan, "Static and dynamic frequency scaling on multicore CPUs," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 13, no. 4, p. 51, 2016.

[7] D. Aikema and R. Simmonds, "Electrical cost savings and clean energy usage potential for hpc workloads," in *Sustainable Systems and Technology (ISSST), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 1–6.

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

[9] T. Patki, D. K. Lowenthal, A. Sasidharan, M. Maiterth, B. L. Rountree, M. Schulz, and B. R. de Supinski, "Practical resource management in power-constrained, high performance computing," in *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*. ACM, 2015, pp. 121–132.

[10] M. Etinski, J. Corbalan, J. Labarta, and M. Valero, "Parallel job scheduling for power constrained HPC systems," *Parallel Computing*, vol. 38, no. 12, pp. 615–630, 2012.

[11] X. Yang, Z. Zhou, S. Wallace, Z. Lan, W. Tang, S. Coghlan, and M. E. Papka, "Integrating dynamic pricing of electricity into energy aware scheduling for HPC systems," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. ACM, 2013, p. 60.

[12] Y. Fan, Z. Lan, P. Rich, W. E. Allcock, M. E. Papka, B. Austin, and D. Paul, "Scheduling beyond cpus for hpc," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, 2019, pp. 97–108.

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

[14] 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.

[15] 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.

[16] 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.

[17] S. Song, C.-Y. Su, R. Ge, A. Vishnu, and K. W. Cameron, "Iso-energy-efficiency: An approach to power-constrained parallel computation," in *2011 IEEE International Parallel & Distributed Processing Symposium*. IEEE, 2011, pp. 128–139.

[18] J. Hofmann and D. Fey, "An ecm-based energy-efficiency optimization approach for bandwidth-limited streaming kernels on recent intel xeon processors," in *2016 4th International Workshop on Energy Efficient Supercomputing (E2SC)*. IEEE, 2016, pp. 31–38.

[19] M. Endrei, C. Jin, M. N. Dinh, D. Abramson, H. Poxon, L. DeRose, and B. R. de Supinski, "Energy efficiency modeling of parallel applications," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2018, pp. 212–224.

[20] E. Calore, A. Gabbana, S. F. Schifano, and R. Tripiccione, "Evaluation of dvfs techniques on modern hpc processors and accelerators for energy-aware applications," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 12, p. e4143, 2017.

[21] C. Lefurgy, X. Wang, and M. Ware, "Power capping: a prelude to power shifting," *Cluster Computing*, vol. 11, no. 2, pp. 183–195, 2008.

[22] K. K. Rangan, G.-Y. Wei, and D. Brooks, "Thread motion: fine-grained power management for multi-core systems," *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3, pp. 302–313, 2009.

[23] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda, "Pack & cap: adaptive dvfs and thread packing under power caps," in *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2011, pp. 175–185.

[24] S. Reda, R. Cochran, and A. K. Coskun, "Adaptive power capping for servers with multithreaded workloads," *IEEE Micro*, vol. 32, no. 5, pp. 64–75, 2012.

[25] K. Ahmed, J. Liu, and K. Yoshii, "Enabling demand response for hpc systems through power capping and node scaling," in *Proceedings of the 2018 IEEE 16th International Conference on High Performance Computing and Communications (HPCC)*. IEEE, 2018, pp. 789–796.

[26] K. Ahmed, K. Yoshii, and S. Tasnim, "Thermal-aware power capping allocation model for high performance computing systems," in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2019, pp. 1488–1493.

[27] Parallel Systems Lab, "Python scheduler simulator," https://code.google.com/archive/p/pyss/, 2010, accessed 10th April 2017.

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

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

[30] W. Tang, Z. Lan, N. Desai, and D. Buettner, "Fault-aware, utility-based job scheduling on blue, gene/p systems," in *Proceedings of the 2009 IEEE International Conference on Cluster Computing and Workshops*. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc., 2009, pp. 1–10.

[31] W. Tang, D. Ren, Z. Lan, and N. Desai, "Adaptive metric-aware job scheduling for production supercomputers," in *Proceedings of the 2012 41st International Conference on Parallel Processing Workshops*, 2012, pp. 107–115.

[32] C. L. Janssen, H. Adalsteinsson, S. Cranford, J. P. Kenny, A. Pinar, D. A. Evensky, and J. Mayo, "A simulator for large-scale parallel computer architectures," *Technology Integration Advancements in Distributed Systems and Computing*, vol. 179, 2012.

[33] Liu, Jason, "Simulus - a discrete-event simulator in python," https://simulus.readthedocs.io/en/latest/#, 2019, accessed 15th November 2019.

[34] K. Ahmed, S. Tasnim, and K. Yoshii, "Simulation of auction mechanism model for energy-efficient high performance computing," in *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 2020, pp. 99–104.

[35] A. Krzywaniak and P. Czarnul, "Performance/energy aware optimization of parallel applications on gpus under power capping," in *International Conference on Parallel Processing and Applied Mathematics*. Springer, 2019, pp. 123–133.