Analyzing Energy-Time Tradeoff in Power Overprovisioned HPC Data Centers

Akhil Langer, Harshit Dokania and Laxmikant V. Kalé
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801-2302
Email: {alanger, hdokani2, kale}@illinois.edu

Udatta S. Palekar
Department of Business Administration
University of Illinois at Urbana-Champaign
Urbana, IL 61820
Email: palekar@illinois.edu

***Abstract*—Minimizing energy and power consumption of large scale data centers is one of the biggest challenges faced by the high performance computing community. In an overprovisioned data center, nodes are power capped to run below their Thermal Design Power (TDP) value and therefore, an overprovisioned data center has more nodes than a conventional data center with the same power budget. In this work, we study the energy versus time trade-off in a power overprovisioned HPC data center. We show that overprovisioning with the goal of maximizing performance can lead to excessive energy consumption. However, careful selection of configuration, that is number of nodes and power cap, can lead to significant savings in energy consumption with very small penalty on execution time of the application. We achieve up to 15% savings in energy consumption with only 2.8% increase in execution time as compared to a configuration that yields the best execution time.**

I. Introduction

Reducing power and energy consumption of the data centers is one of the major challenges faced by the High Performance Computing (HPC) community. As size of data centers is increasing, their energy consumption is also increasing. Power is the instantaneous rate at which energy is provided to the data center, while energy is the total power consumption over a period of time. In this work, we focus on optimizing energy consumption of power-budgeted data centers while minimizing its impact on application performance.

Traditionally, equipment cost has been a limiting factor in achieving high performance in data centers. However, as we move towards exascale machines, the thrust is shifting from equipment cost to power/energy cost. In other words, future large-scale HPC data centers will be power limited as compared to equipment limited. This is because high power requirements mean huge infrastructure required to supply large amounts of instantaneous power. Therefore, DoE has set 20MW as the power budget for exascale machines. Recent research has focused on making efficient utilization of the available power budget. Power capping makes it possible to constrain CPU power consumption of the nodes to value below the CPU Thermal Design Power (TDP) value, where TDP is the maximum amount of power that a node can draw. Power capping feature is supported by some of the recent processor architectures, such as Intel SandyBridge [1], IBM Power6 [2], Power7 [3] and AMD Bulldozer [4] architectures. Power capping makes it possible to add more nodes to the data center, each node running below its TDP value, while staying within the overall power budget of the data center. This is also called as an overprovisioned data center. Since application performance does not scale proportionately with increase in CPU frequency, appropriate power capping can lead to significant power savings without significantly affecting execution time of the application. In fact, performance can be improved by using the saved power to turn-on more nodes and scaling-up the application. Previous work [5], [6] shows that significant speedups can be obtained by using such overprovisioning of the data centers when compared with a traditional/conventional data center with the same power budget. Therefore, overprovisioning is a promising solution to achieve exascale with a given power budget, as also proposed by [7], [8]. The performance benefits due to overprovisioning vary from application to application. Some applications are computationally intensive and benefit from using the power to increase CPU frequency, while other applications are memory intensive and benefit from using the power to turn-on more nodes each running at below their maximum frequency.

Even though overprovisioning of data centers gives significant improvement in performance for a given power budget, it can also lead to increased energy consumption because of the addition of nodes. Therefore it is also important to control the energy consumption in order to reduce the electricity costs. In this work, we analyze the energy consumption of applications in an overprovisioned system. We study how careful selection of the nodes and power cap can lead to significant savings in energy with minimal penalty in execution time.

The work is divided in to 6 sections. In Section II, we do a literature review of work on energy consumption optimization of data centers using hardware features such as DVFS. We also review the work so far that utilizes the power capping feature supported by recent hardware architectures. Section III gives background and motivation for this work. Section IV, Section V explain the experimental setup and results, respectively. Finally, the conclusions are given in Section VI.

II. Literature Review

Performance of HPC applications does not increase proportionately with increase in CPU frequencies. On the other hand, dynamic power consumption of the CPU is proportional to cubic power of the CPU frequency. Therefore, previous research has suggested reducing CPU frequencies to decrease the energy consumption while having acceptable penalty on the execution time of the jobs [9], [10]. This is achieved using Dynamic Voltage and Frequency Scaling (DVFS). Optimal frequency selection in DVFS-based energy consumption minimization has been studied by Rizvandi et al [11]. Wange et al [12] propose heuristics to identify slack in non-critical tasks, They use this information to reduce the frequencies of processors running non-critical tasks without affecting the overall execution time of the job. In the context of DVFS, Freeh et al [13] have shown that for some NAS benchmarks, it is possible to save energy and to reduce time by running the application on more nodes at lower frequency rather than running on fewer nodes at higher frequency. In this work, we focus on HPC data centers with strict power budget, and use power capping capability to optimize energy consumption while minimizing its impact on execution time of HPC applications.

Recent research has focused on maximizing performance under a fixed power budget. As we move towards exascale computing, adding new nodes to the data center will not be as constraining as compared to cost and infrastructure required to supply power to run all the nodes. Kontorinis et al [14] propose an architecture for distributed UPSs (Uninterrupted Power Supplier) that store energy during low activity periods (low power requirements) and supply energy during high activity period, thus allowing installation of more servers within the same power budget. In this work, we use power capping that makes it possible to constrain power consumption of the nodes to below their TDP value. For a given power budget, this makes it possible to turn-on more nodes as compared to a conventional data center where all the nodes are allocated TDP amount of power. This is also called as overprovisioning. Sarood et al [15] show that up to 5.2x improvement in throughput can be obtained in an overprovisioned data center as compared to a conventional data center with the same power budget. Balaji et al [16] study the energy proportionality of enterprise benchmarks using power capping of different subsystems. In contrast, our work is focused on scientific workloads. Laors et al [17] have developed a new vendor-neutral API for power management and control of large systems.

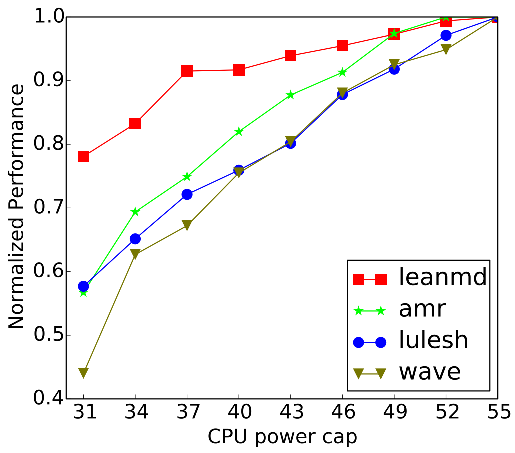
Modeling application performance given a power cap has also been studied. These models can be used to determine optimal power cap for the desirable user objective. Balaji et al [18], propose non-linear models to capture the relationships between the throughput, response time, and subsystem-level power limits for SPECpower and SPECweb enterprise benchmarks [19]. Storlie et al [20] propose a statistical model for modeling the power draw of jobs that can be used to optimally allocate power to jobs according to a given criterion, for example, maximizing data center throughput, user priorities, etc. They study these models for jobs submitted to a HPC data center. In this work, we do not focus on performance modeling for HPC applications under a given power budget, but modeling ideas as proposed in [6], [15] can be used for automated selection of configurations with desirable properties.

III. Background and Motivation

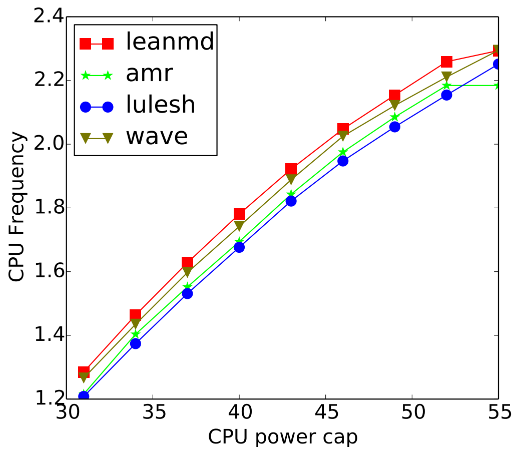
Modern processors have different performance states, also called as P-states. A P-state corresponds to the processor's frequency and voltage. For example, Intel Xeon 5160 processor has four P-states - P0, P1, P2, and P3 corresponding to core frequency of 3.0, 2.66, 2.33, 2 GHz, respectively. P0 is the highest performance state with maximum frequency but also higher power consumption. Higher P-state numbers represent lower frequency and lower power consumption. P-state of the processor can be changed to lower its power consumption. Clock throttling is another method for lowering power consumption of processor. In clock throttling, processor is forced to be idle a fixed number of cycles per second. This lowers processor frequency and reduces thermal effects. Since voltage is not changed in clock throttling, it has little effect in reducing the power consumption. However, clock idling significantly increases the execution time of the application, leading to higher energy costs. Power capping is achieved by using P-states and/or clock throttling. P-states can lower the power consumption only to a certain point, after which clock throttling is used to further lower the power consumption. Power capping guarantees that power consumption of processor will not exceed a user-specified threshold. The power management system carefully observes the power consumption to keep it below the user-specified power cap.

Exascale centers are expected to have huge power requirements. Special infrastructure is required to support such large amount of instantaneous power to the data centers. Therefore, the data centers will have a power budget that the infrastructure can support. Power capping allows us to constrain power consumption of nodes, and add more nodes to the data center while remaining within the power budget of the data center. Having large number of nodes running at lower power/frequency gives better execution time for many applications than running them on fewer nodes each with higher power/frequency. Therefore, overprovisioning improves the overall performance of the data center. Power capping ensures that the power consumption remains below the power budget and hence prevent circuit breakers from tripping. CPU power management can control power consumption at the granularity of milliseconds which is sufficient because the capacitance on the mother board and the power supply smoothes out the power draw at a much greater granularity.

Currently, users/customers of data centers are charged computational power in the units of node hours or core hours. Large data centers will have very high electricity costs. For example, if we assume that electricity costs USD 0.07 per KWh [21], an exascale data center with 20MW power consumption will cost about USD 1 million per month in electricity costs only. Therefore, in the future, it is very likely that users will be charged in the units of energy consumption as compared to the traditional way of charging in the units of core hours. This will also encourage users to run their jobs in an energy-efficient manner rather than running for the best execution time. Injudicious overprovisioning can lead to very high energy costs. Therefore, selecting the right configuration (number of nodes, and power cap) is very important for desirable energy-vs-time trade-off. We study the performance and energy consumption of various HPC mini-applications for different configurations and show how significant energy savings can be made by intelligent selection of configuration.



(a)



(b)

Fig. 1: (a) Impact of power capping on performance of the four applications running on 4 nodes. Performance is normalized with respect to performance at the highest power cap (55W), (b) CPU frequencies at different power caps for the four applications running on 4 nodes.

IV. Evaluation Setup

A. *Computational Testbed*

Our experiments were performed on a 38-node Dell PowerEdge R620 cluster. It is installed in a room with a computer room air conditioning (CRAC) unit that monitors and maintains the temperature, air distribution and humidity in the room. CRAC temperature is set to 23.3 °C. Each node in the cluster contains an Intel Xeon E5-2620 Sandy Bridge server with 6 physical cores running at 2GHz, 2-way SMT with 16GB of RAM. These servers support on-board power measurement and power capping. We use the Runtime Average Power Limit (RAPL) interface [22] to measure and constrain CPU power.

In this testbed, CPU power can be capped in the range [25-95]W. Our empirical observations show that CPU power consumption for the focal applications never goes beyond 55W. When the CPU is power capped below 30W, execution time for the applications increases significantly even with small reduction in power cap. This is because at these low power caps, the power management has exhausted the P-states and starts using CPU throttling to reduce the power consumption. Power caps less than 30W will therefore not be used for most practical purposes. In our experiments, we apply power caps from the set [31, 34, 37, 40, 43, 46, 49, 52, 55]W to all the applications. In all our experiments, memory power consumption never exceeds 15W, while the TDP value of memory is 35W. In this work, we only focus on CPU power capping and do not apply power capping to memory. Idle power or the base power consumption of the node, excluding the CPU and memory power consumption, is 38W. Using CPU and Memory's TDP values, the maximum power a node can consume is (38 + 35 + 95)W = 168W.

B. *Applications*

Four applications were used. These applications represent the kernels of different high performance computing applications typically used on many of the world's largest supercomputers. We call these programs as mini-applications. They vary from computationally intensive to memory intensive applications. The description of these applications and their input parameters used in our experiments are as follows:

1) Wave - This program solves a wave equation of the following form,

$$\frac{\partial^2 p}{\partial p^2} = c^2 \nabla^2 p \tag{1}$$

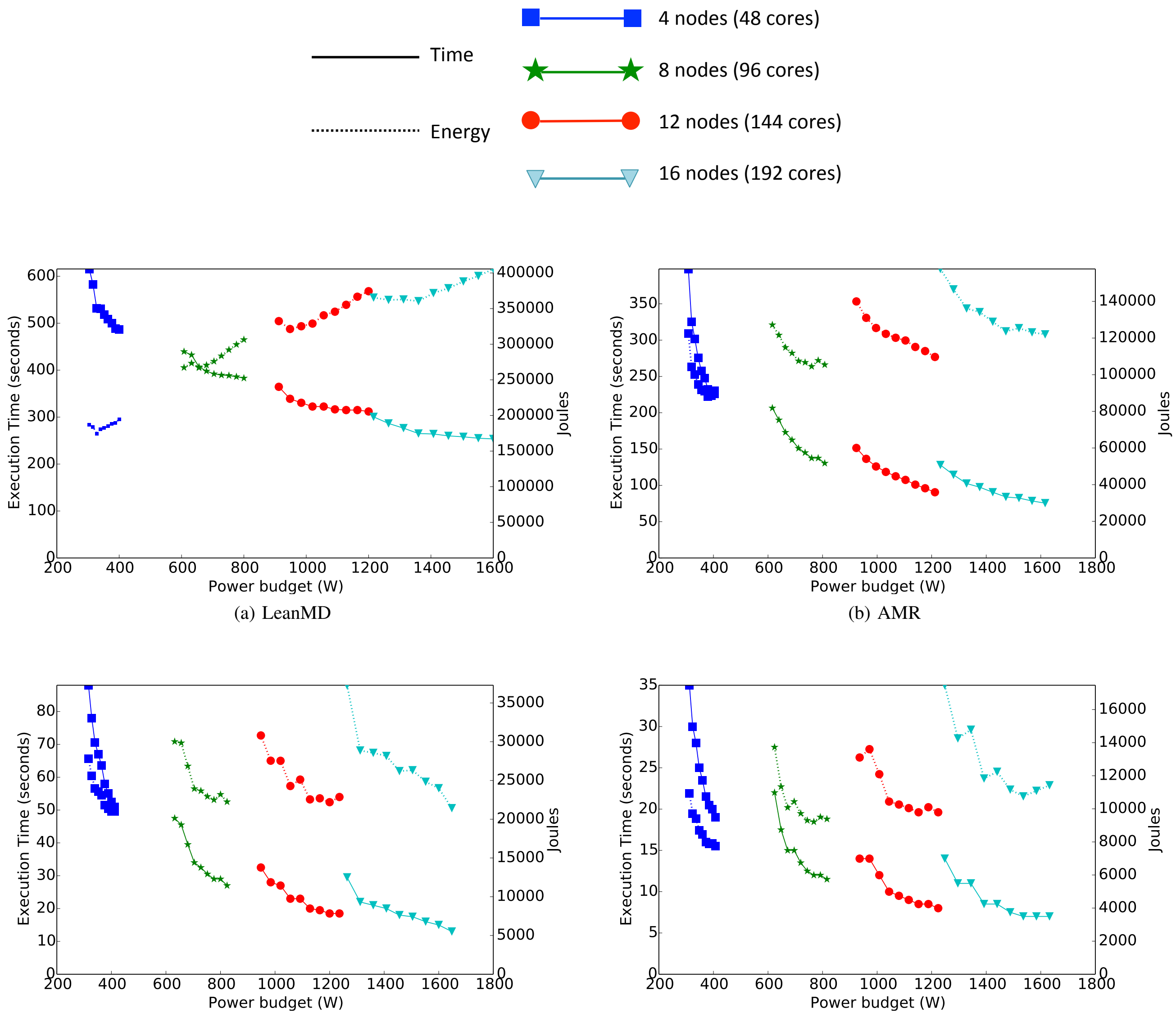
using a finite difference scheme over a 2D mesh. It calculates the pressure in the grid resulting from an initial set of perturbations. Our experiments were conducted on a mesh of size 20000 $\times$ 20000, with 5 initial perturbations simulated for 40 iterations.

2) Lulesh - It is a shock hydrodynamics application [23], [24] that represents the numerical algorithms, data motion, and programming style typical in scientific C/C++ based applications. Lulesh is one of the five challenge problems in the DARPA UHPC program, and was defined and implemented by Lawrence Livermore National Laboratory (LLNL). For our experiments, we run LULESH on a grid of size 512 $\times$ 512 $\times$ 512.

3) Adaptive Mesh Refinement (AMR) - This is a mini-application for oct-tree based structured adaptive mesh refinement simulations. It benchmarks motion of a circular fluid advected by a constant velocity fluid and is described by the following hyperbolic partial differential equation,

$$\frac{\partial u}{\partial t} + v \nabla u = 0 \tag{2}$$

In AMR simulations, works is dynamically created and destroyed during runtime and load balancing is done every few iterations to balance the load across the processors. More details about the algorithms used for implementing AMR can be found in [25],



(a) LeanMD (b) AMR (c) LULESH (d) Wave

Fig. 2: Execution time and energy consumption for different configurations plotted against the power budget for four different applications. Solid lines (——) correspond to execution time, while the dotted lines (. . . . . .) correspond to energy consumption. Different colors correspond to the number of nodes in the configuration. Legend for colors is shown at the top of the figure. Markers on each line correspond to the CPU power caps in the set [31, 34, 37, 40, 43, 46, 49, 52, 55]W. To determine the best performing configuration for a given power budget draw a vertical line at the power budget and chose the configuration to the left that has the minimum execution time.

[26]. In our simulations, the initial mesh size was $256 \times 256 \times 256$, with maximum allowed depth of 9. The simulation is run for 15 iterations, with mesh restructuring after every 3 iterations, and load balancing after every 6 iterations.

4) LeanMD - It is a parallel molecular dynamics simulation program that simulated behavior of atoms based on Lennard-Jones potential. It involves calculating forces in all atoms during each time-step, and updating the position, velocities and acceleration of the atoms using these forces. It is implemented in Charm++ [27] and has force structure very similar to NAMD [28]. We simulate a total of $36 \times 36 \times 36$ particles, for a total of 10 time steps.

V. Results

In this section, we present our observations on impact of power capping on the focal applications, and we analyze the energy-vs-time trade-off for these applications.

Depending upon their CPU and memory sensitivity, performance of different applications is affected differently by power capping. Figure 1a shows the performance of the four applications running on 48 cores (4 nodes) at various power caps (all nodes are power capped at the same value). The performance is normalized with respect to the performance at the highest power cap, i.e. 55W. Performance of LeanMD is affected the least as power cap is decreased, while the performance of Wave is affected the most. Correlation between CPU frequencies and application sensitivity to power capping is not direct (Figure 1b). LeanMD is least affected by power capping and has the highest frequency of all the applications. AMR, which is the next application that is affected the least by power capping, has very low frequency. Wave and Lulesh, which have similar sensitivity to power capping, have very different frequencies. This implies that power is a complex interplay of power consumption by the cores and by other on-chip activity like cache accesses, memory controller, etc.

We define a configuration, (n, p), as a combination of number of nodes (n) and CPU power cap per node (p). Experiments were performed for configurations with $n \in [4, 8, 12, 16]$, and $p \in [31, 34, 37, 40, 43, 46, 49, 52, 55]W$. Power budget for a configuration is computed as $n * (p + 15 + 38)$W, where 15W is the maximum memory power consumption and 38W is the maximum idle power of a node. Total energy consumption of a configuration for an application is calculated by measuring the actual memory and CPU power consumption integrated over the execution period of the application. Figure 2 shows the energy consumption and execution time of the four applications for various configurations. They are plotted against the power budget of the configurations. Colors correspond to the different values of n. Solid lines represent execution times, and dotted lines represent energy consumption. A line corresponds to all configurations with the same number of nodes. Markers along a line correspond to the different power caps of the configuration with the leftmost marker on a line corresponding to the smallest power cap (i.e. 31W).

In a Conventional Data Center (CDC), TDP amount of power is allocated to each node so that the circuit breakers do not trip on the occasion that the power consumption reaches TDP. Given a power budget of P Watts, a CDC will be able to use $\frac{P}{168}$ nodes, where 168W is the maximum possible power draw of a compute node (Section IV-A). An Overprovisioned Data Center (ODC) uses power capping and has more nodes than a CDC while remaining within the same power budget. An ODC with a configuration that maximizes the performance (pODC) will have very high energy consumption (Figure 2), and hence it is not very desirable. As can be observed in Figure 2, energy consumption can be minimized by running at a configuration with least number of nodes (four, in this case). However, such configurations have very large execution time and are therefore also not desirable. We examine various scenarios and show how careful selection of configuration (let us call it iODC) can lead to significant energy savings as compared to pODC with minimal impact on execution time:

- Given a total power budget of $P = 1450$W and AMR application, in pODC, configuration (16, 43) gives the best execution time of 90.5 seconds. This is 30% improvement in execution time with 22% increase in energy consumption over a CDC which can enable only 8 nodes for the given power budget. However, energy consumption of the ODC can be reduced by 14.9% by using the configuration (12, 55) with mere 0.001% increase in execution time as compared to the configuration (16, 43) with the best execution time. In other words, in iODC, by intelligent selection of the configuration, a 30% improvement in execution time can be achieved with only 4% increase in energy consumption as compared to a CDC setting.
- Given $P = 1200$W and LeanMD, configuration (12, 55) gives the best execution time. However, by running with configuration (12, 46), we can obtain 7.7% savings in energy while having only 1.4% penalty in execution time.
- Given $P = 1500$W and Lulesh, configuration (16, 43) yields the best execution time, while the configuration (12, 52) gives savings of 15.3% in energy with only 2.8% increase in the execution time.
- Given $P = 1550$W and Wave, configuration (16, 46) gives the best execution time, but a 12% savings in energy consumption can be obtained with 6% increase in execution time by using the configuration (12, 55).

In this analysis, our choices were limited by the sample configurations for which the application performance was empirically profiled. However, by modeling application performance with power cap and number of nodes, more optimal configurations can be selected using the model function. It requires collecting profile information for minimum number of samples required to build the model, and then using the model to predict the quality of any configuration. This has been left for future work.

VI. Conclusion

Although overprovisioned data centers have promise for maximizing power efficiency for a given power budget, it also has the disadvantage of excessive energy consumption. In this work, we analyzed the energy-vs-time trade-off for power overprovisioned HPC data centers. We showed that careful selection of configurations can lead to significant savings in energy consumption with little impact on the execution time. For some applications, running at fewer nodes and a higher power cap can lead to 15.3% savings in energy with only 2.8% increase in execution time as compared to a configuration that yields the best execution time.

Acknowledgment

Work of the authors, Akhil Langer, Udatta Palekar, and L.V. Kale is partially supported by MITRE Research Agreement Number 81990 with UIUC.

References

[1] 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 *IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW)*, 2012.

[2] B. Behle, N. Bofferding, M. Broyles, C. Eide, M. Floyd, C. Francois, A. Geissler, M. Hollinger, H.-Y. McCreary, C. Rath *et al.*, "IBM Energyscale for POWER6 Processor-based Systems," *IBM White Paper*, 2009.

[3] M. Broyles, C. Francois, A. Geissler, M. Hollinger, T. Rosedahl, G. Silva, J. Van Heuklon, and B. Veale, "IBM Energyscale for POWER7 Processor-based Systems," *white paper, IBM*, 2010.

[4] "Advanced Micro Devices. BIOS and Kernel Developer's guide (BKDG) for AMD Family 15h Models 00h-0fh Processors," January 2012.

[5] T. Patki, D. Lowenthal, B. Rountree, S. Martin, and B. d. Supinski, “Exploring Hardware Overprovisioning in Power-Constrained, High Performance Computing,” in *Proceedings of the 27th International Conference on Supercomputing, ICS (to appear)*, 2013.

[6] O. Sarood, A. Langer, L. V. Kale, B. Rountree, and B. de Supinski, “Optimizing Power Allocation to CPU and Memory Subsystems in Overprovisioned HPC Systems,” in *Proceedings of IEEE Cluster 2013*, Indianapolis, IN, USA, September 2013.

[7] N. P. Carter, A. Agrawal, S. Borkar, R. Cledat, H. David, D. Dunning, J. Fryman, I. Ganev, R. A. Golliver, R. Knauerhase *et al.*, “Runnemede: An architecture for ubiquitous high-performance computing,” in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*. IEEE, 2013, pp. 198–209.

[8] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, and D. Glasco, “Gpus and the future of parallel computing,” *IEEE Micro*, vol. 31, no. 5, pp. 7–17, 2011.

[9] R. Ge, X. Feng, and K. W. Cameron, “Improvement of power-performance efficiency for high-end computing,” in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*. IEEE, 2005, pp. 8–pp.

[10] F. Pan, V. W. Freeh, and D. M. Smith, “Exploring the energy-time tradeoff in high-performance computing,” in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*. IEEE, 2005, pp. 9–pp.

[11] N. B. Rizvandi, J. Taheri, and A. Y. Zomaya, “Some Observations on Optimal Frequency Selection in DVFS-based Energy Consumption Minimization,” *Journal of Parallel and Distributed Computing*, vol. 71, no. 8, pp. 1154–1164, 2011.

[12] L. Wang, S. U. Khan, D. Chen, J. Kołodziej, R. Ranjan, C.-z. Xu, and A. Zomaya, “Energy-aware Parallel Task Scheduling in a Cluster,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1661–1670, 2013.

[13] 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. 4a–4a.

[14] V. Kontorinis, L. E. Zhang, B. Aksanli, J. Sampson, H. Homayoun, E. Pettis, D. M. Tullsen, and T. Simunic Rosing, “Managing distributed ups energy for effective power capping in data centers,” in *Computer Architecture (ISCA), 2012 39th Annual International Symposium on*. IEEE, 2012, pp. 488–499.

[15] O. Sarood, A. Langer, A. Gupta, and L. V. Kale, “Maximizing throughput of overprovisioned hpc data centers under a strict power budget,” in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC ’14. New York, NY, USA: ACM, 2014.

[16] B. Subramaniam and W.-c. Feng, “Towards energy-proportional computing for enterprise-class server workloads,” in *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*. ACM, 2013, pp. 15–26.

[17] J. H. Laros III, D. DeBonis, R. Grant, S. M. Kelly, M. Levenhagen, S. Olivier, and K. Pedretti, “High performance computing-power application programming interface specification version 1.0,” *Sandia National Laboratories, Tech. Rep. SAND2014-17061*, 2014.

[18] B. Subramaniam and W.-c. Feng, “Enabling efficient power provisioning for enterprise applications,” in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*. IEEE, 2014, pp. 71–80.

[19] “SPECpower and SPECweb benchmarks,” http://www.spec.org/power_ssj2008, http://www.spec.org/web2009.

[20] C. Storlie, J. Sexton, S. Pakin, M. Lang, B. Reich, and W. Rust, “Modeling and predicting power consumption of high performance computing jobs,” *arXiv preprint arXiv:1412.5247*, 2014.

[21] “DoE Data on Pricing for Industrial Electricity,” http://www.eia.gov/electricity/data.cfm.

[22] Intel, “Intel-64 and IA-32 Architectures Software Developer’s Manual , Volume 3A and 3B: System Programming Guide, 2011.”

[23] I. Karlin, A. Bhatele, J. Keasler, B. Chamberlain, J. Cohen, Z. DeVito, R. Haque, D. Laney, E. Luke, F. Wang *et al.*, “Exploring Traditional and Emerging Parallel Programming Models using a Proxy Application,” in *International Parallel and Distributed Processing Symposium*, 2013.

[24] I. Karlin, J. Keasler, and R. Neely, “Lulesh 2.0 updates and changes,” Tech. Rep. LLNL-TR-641973, August 2013.

[25] A. Langer, J. Lifflander, P. Miller, K.-C. Pan, , L. V. Kale, and P. Ricker, “Scalable Algorithms for Distributed-Memory Adaptive Mesh Refinement,” in *Proceedings of the 24th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2012)*, New York, USA, October 2012.

[26] A. Langer, “An Optimal Distributed Load Balancing Algorithm for Homogeneous Work Units,” in *Proceedings of the 28th ACM International Conference on Supercomputing*, ser. ICS ’14. New York, NY, USA: ACM, 2014, pp. 165–165. [Online]. Available: http://doi.acm.org/10.1145/2597652.2600108

[27] B. Acun, A. Gupta, N. Jain, A. Langer, H. Menon, E. Mikida, X. Ni, M. Robson, Y. Sun, E. Totoni, L. Wesolowski, and L. Kale, “Parallel Programming with Migratable Objects: Charm++ in Practice,” in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC ’14. New York, NY, USA: ACM, 2014.

[28] J. Phillips, G. Zheng, and L. V. Kalé, “Namd: Biomolecular simulation on thousands of processors,” in *Workshop: Scaling to New Heights*, Pittsburgh, PA, May 2002.