2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum

Beyond DVFS: A First Look at Performance Under a Hardware-Enforced Power Bound

Barry Rountree*, Dong H. Ahn*, Bronis R. de Supinski*, David K. Lowenthal†, Martin Schulz*
*Lawrence Livermore National Laboratory
†Department of Computer Science, The University of Arizona

***Abstract*—Dynamic Voltage Frequency Scaling (DVFS) has been the tool of choice for balancing power and performance in high-performance computing (HPC). With the introduction of Intel's Sandy Bridge family of processors, researchers now have a far more attractive option: user-specified, dynamic, hardware-enforced processor power bounds. In this paper we provide a first look at this technology in the HPC environment and detail both the opportunities and potential pitfalls of using this technique to control processor power.**

As part of this evaluation we measure power and performance for single-processor instances of several of the NAS Parallel Benchmarks. Additionally, we focus on the behavior of a single benchmark, MG, under several different power bounds. We quantify the well-known manufacturing variation in processor power efficiency and show that, in the absence of a power bound, this variation has no correlation to performance. We then show that execution under a power bound translates this variation in efficiency into variation in performance.

I. Introduction

Power has now become the primary performance problem in high-performance computing (HPC). Up to this point, Dynamic Voltage/Frequency Scaling (DVFS) has been the method of choice for investigating the tradeoff between power and performance in HPC applications. Running the processor at a lower clock frequency requires less voltage, but the impact on performance and the amount of power and energy saved is highly application dependent. While research has made great strides in modeling these effects, to our knowledge no machine in the Top 500 list of supercomputers makes use of DVFS to save power or energy.

Power clamping provides a potentially compelling alternative to DVFS. Instead of managing the processor's frequency directly, the user simply specifies a time window and a power bound and the hardware guarantees that the average power will not exceed the specified bound over each window. Both the window size and bound may be modified at runtime. This mechanism enables system designers and operators to control the exact amount of power that each processor consumes across the entire system.

However, we account for several subtleties in order for power clamping to gain widespread acceptance in the HPC community. In this paper, we explore the variations in power efficiency across processors and how this variation is translated into variations in performance while under a power bound. Small variations in processor power consumption inevitably introduced in the manufacturing process are well-known [7] and do not affect processor speed. Placing a power bound on a processor moves the variation from power (the processor now operates at a specified number of watts) to performance (less-efficient processors run more slowly at the specified power bound).

This perfromance impact is of particular interest in the HPC domain. Up to this point, machines have been specified with the assumption that variation in power can be tolerated while variation in performance cannot: a few inefficient processors do not exceed the power provisioning of a machine room, but a single slow processor can constrain the performance of an entire application. Under a power bound, those few inefficient processors are transformed into slow processors. This paper quantifies that effect on real hardware for several of the NAS Parallel Benchmarks [15].

II. Overview

Until now, most research in power-aware supercomputing has focused on trading a loss of performance for energy savings. While an interesting problem in its own right, it did not match well with the goal of supercomputer stakeholders, which is to make an existing machine run as fast as possible. In effect, these stakeholders were asked to consider doing less work per unit time in order to save some amount of someone else's money. These approaches have not gained any traction in the wider community.

However, for the largest supercomputers, the amount of electricity that can be brought into the machine room is becoming the limiting factor on their capability and, thus, the amount of work that they can perform in a fixed time period. As a result, we can no longer simply purchase additional homogeneous nodes to increase performance because no power is available to bring them online.

Copyright 2012 IEEE. IEEE acknowledges that this contribution was authored or co-authored by a contractor or affiliate of the U.S. Government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only. This work was partially performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. HPPAC May 2012, Shanghai, China.

The processor architecture community has already reached this point and we propose to adopt their strategy. The most recent processors from both AMD and Intel are *overprovisioned* with respect to power: not all cores can run simultaneously at the highest possible frequency, and the user effectively buys capacity that will always remain unused. What the user buys instead is *flexibility*, either to run all cores at a slower frequency or a handful of cores at a faster frequency.

We foresee future clusters designed the same way. For problems that benefit from the largest number of processors, nodes will run at either a low CPU frequency or at a low power bound. For problems that perform best given a smaller number of faster nodes, the user or operator will schedule a smaller number of nodes with a higher power draw and turn off the remaining nodes. Utilization will no longer be measured as a percentage of node-hours but rather as a percentage of maximum kilowatts.

Making this approach a reality requires solving problems far more difficult than the relatively straightforward problem of saving energy. The user is presented with what is effectively a dynamically reconfigurable, homogeneous cluster and must determine not only the optimal number of nodes but also how much power should be assigned to each node, and for how long. In short, we have moved from an *energy savings* problem to a *power scheduling* problem.

As we detail in the discussion section, both DVFS and power clamping have their strengths when brought to bear on the scheduling problem: power clamping provides a hard, tight bound on power but performance modeling under the power bound becomes much more difficult. DVFS provides a loose, soft bound, but performance under user-controlled DVFS is now well-understood. The experimental results that follow are the first necessary steps to understand performance under a power bound.

III. Intel's Running Average Power Limit (RAPL)

With the Sandy Bridge family of processors, Intel introduced both onboard power meters and power clamping. In this section we provide a technical overview that is informed by our practical experience with these tools. To the best of our knowledge, the only documentation for these features is in chapter 14.7 of Intel's *Software Developer's Manual*[11]. We stress that we are experimenting with new processors in a pre-production environment and issues raised here may be resolved in the near future.

A. Interface

Users measure and control processor power using several *model-specific registers*, or MSRs. Intel provides two privileged instructions, `readmsr` and `writemsr`, as the interface to these registers. Instead of writing a specialized kernel driver, users and developers on Linux can use the *msr* kernel module. This module exports a file interface at `/dev/cpu/N/msr` that, given suitable file permission, can be used to read and write any MSR on the node. This approach has significant security implications and should only be used for development in a trusted environment.

B. Architectures

Intel separates the Sandy Bridge family into two classes: client (family=0x06, model=0x2A); and server (family=0x06, model=0x2D). The server-class processor receives the *Xeon* designation. The two architectures share a subset of RAPL features. We only use Xeon processors in this work.

C. Domains

The Sandy Bridge architecture supports three power domains on each architecture. Both architectures support *package* (PKG) and *Power Plane 0* (PP0) domains, while the server adds a separate *DRAM* domain and the client adds a second power plane (PP1). The documentation provides little information to differentiate the circuitry that each domain covers. For example, "PP1 may reflect to uncore" [the *unified core* abstraction of last-level cache] and "Generally, PP0 refers to the processor cores" exhaust the descriptive documentation of these two domains.

Our testbed does not support measurement or control of the DRAM domain. Across the NAS Parallel Benchmark suite, the power ratio between the PKG and PP0 domain remained nearly constant. In this work we limit our experiments to measurement and control of the PKG domain.

D. Units

We have not found any documentation that describes the accuracy of the time, power and energy measurements. Precision is architecture-specific and is provided by reading the `MSR_RAPL_POWER_UNIT` register. Our architecture reports power clamping will be performed in units of 0.125W over time windows with units of 0.977 milliseconds. Energy measurements are reported in units of 0.0152 milliJoules.

E. The PKG Domain

The `POWER_LIMIT` set of MSRs reports the architecture-specific power envelopes and maximum clamping time window for each domain. Our PKG domain supports power bounds as low as 51W. Our *thermal spec power* is rated at 115W and our maximum power is 180W. The maximum time window for power clamping is 0.0459 seconds.

F. Power Clamping

The Sandy Bridge processor does not provide a power bound in the strictest sense. Instead, the user specifies a time window and a maximum average power for that window and the processor guarantees that it will not exceed this average. Intuitively, longer windows may allow better performance for applications that utilize the CPU in bursts; if the burst exceeds the window size, the processor will have to be throttled. Our future research will determine how the size of this window affects machine room power provisioning, particularly for parallel scientific applications, which tend to have highly synchronized load spikes.

The PKG domain provides for two separate clamping windows. A user can provide a higher bound for a smaller window and a lower bound for a larger window, which may provide finer control over application performance. A lower bound for the smaller window would make the larger window superfluous. We use a single window of the smallest possible size (0.000977 seconds) for all experiments in this paper. Smaller windows best avoid potential power spikes in the absence of application-specific knowledge.

The processor provides two modes for power clamping: *enabled* and *clamping*. Setting the former causes the processor to respect the minimum performance level request by the operating system. Setting both allows the processor to override the OS if necessary to meet the power bound. In our experience, setting only the *enabled* bit did not change the power profile of any benchmark. All experiments reported in this paper set both bits high.

G. Other Interfaces

Each domain has a separate, read-only, 32-bit energy meter. As the unit of joules is so small, this meter rolls over every few hours. The PP0 and PP1 domains expose a *policy* interface that may only be useful on client architectures. The PKG and DRAM domains expose a counter that records the number of seconds spent below the performance level requested by the operating system. This measurement could be useful when the power bound is only expected to be reached sporadically. In our work we expect that the clamping will operate more or less continuously, and so we have not investigated these counters further.

IV. Experimental Results

We performed these experiments on a dedicated 32-node partition of the Zinfandel TLCC2 cluster and the general-use 137-node partition on the Merlot TLCC2 cluster, both at Lawrence Livermore National Laboratory. Each node contains two Sandy Bridge 8-core processors. Benchmarks were compiled using GCC 4.4.6 and MVAPICH2 1.7 and executed under a Red Hat Linux derivative using the 2.6.32 Linux kernel. We configured the NAS Parallel Benchmarks to use the class C problem size and 8 MPI ranks. We explored several representatives of the suite initially and then focus on the MG benchmark. We choose this benchmark because it consumes the most amount of power of the NAS Benchmarks, and because it executes for a reasonable length of time while keeping all eight cores busy. We configured experiments that use a hardware power bound to use a 1 millisecond window on the PKG domain with clamping enabled.

Figure 1 shows power variation across the 64 processors in our testbed partition. We measure average power on each processor for selected NAS Parallel Benchmarks. We order the processors by maximum power draw. Three processors are annotated: the two least-efficient processors are on nodes 49 and 50, and the most-efficient processor is found on node 48. The MG benchmark consumes the greatest amount of power, with roughly ten watts separating the least- and most-efficient processors.

Figure 2 shows the results for the MG benchmark. We compile the MPI version of the benchmark to use eight MPI ranks, one for each core in the machine. We use an MPI profiling library to set up the necessary MSRs immediately after the program returns from `MPI_Init` and measure and reset the MSRs immediately before calling `MPI_Finalize`. We read total joules from the PKG domain and divide this over elapsed time to calculate average watts.

We run the benchmark 34 times on each processor, once without any power bound and once at each of the 33 power bounds ranging from 51W to 83W. Each point on the graph represents a single run and its associated power (given in average watts) and performance (given in seconds of execution time). We highlight the following special cases:

Unbounded power:: The bottom right corner of the graph has a cluster of black crosses, a red empty circle and a blue empty triangle. These points represent each of the 64 processors in normal operation, *i.e.*, with no user-specified power bound. These points are distributed horizontally: each processor takes roughly the same amount of time to execute the benchmark. However, these data points are spread out over the power dimension, ranging from 77.74W for the most-efficient processor (the blue empty triangle) to 85.36W for the least-efficient processor (the empty red circle).

We emphasize the *lack* of correlation between efficiency (as measured by power consumption) and performance (as measured by wall clock time). All 64 processors operate at the same clock frequency and have the same execution rate. However, some processors use more power than others.

Bounded power:: Starting from the top left of the graph we show average power and execution time while running under a user-specified power bound. We track the processors identified as most and least efficient based on non-bounded execution. The same processors are usually the most and least efficient at nearly all measured bounds. However, we now express efficiency in terms of time. Execution times range from 16.26 seconds for the most-efficient processor to 17.23 seconds for the least efficient. We again emphasize the *lack* of correlation between efficiency and power consumption. Regardless of how efficient a processor may be, if it is operating under a user-specified power bound, its power consumption matches the bound precisely.

The previous experimental results measure power and execution time over the entire execution of the benchmark. In Figure 3 we zoom in to measure the effects of bounding power at a much finer scale. We run the class-E version of MG over 256 cores (16 nodes) and five power bounds

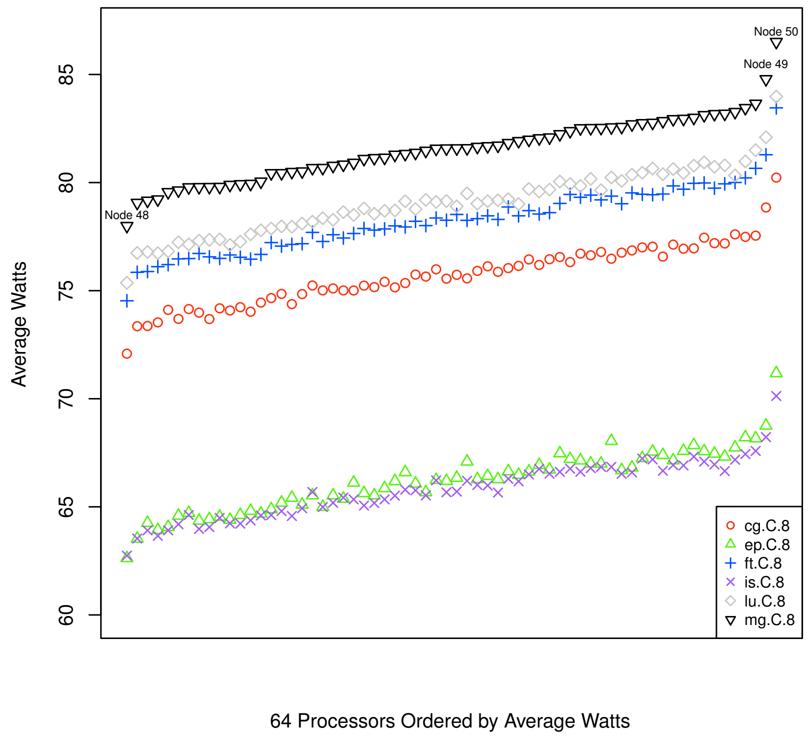

Figure 1. 64 processors ordered by average power consumption over selected NAS Parallel Benchmarks.

ranging from the lowest supported by the PKG domain (51W) to a bound that is high enough not to affect execution time (91W). Our instrumentation records timestamps and accumulates energy at the beginning and end of each MPI library call, thus distinguishing between time and energy spent during computation and communication tasks. From the over 25,000 measurements that result, we show only those computation tasks that take more than 100 milliseconds at the 91W bound.

For this particular instance of this particular benchmark, power varies over a narrow range at the 91W bound. At the 71W bound nearly every task is slowed with the greatest variability in slowdown occuring at the 51W bound. This test reflects the behavior of a single processor (node zero); observed variation in these experiments is mostly likely due to variation in execution rather than variation in hardware.

V. Discussion

In this paper we have laid out a vision for a new approach to power-aware supercomputing and demonstrated a new tool that may help to achieve that vision. In this section, we sketch out some less obvious implications of computing under a power bound.

A. *Processor-level modeling*

The combination of high-resolution timers and onboard power meters makes processor-level performance modeling far more tractable. For a given static computational load, we can easily plot performance under an arbitrary power bound. If we are only going to use a subset of the processors in a system than an obvious optimization is to select first from the most-efficient processors.

Performance modeling becomes far more interesting when we no longer assume that loads are static. For example, running a load-imbalanced application under a power bound could produce dramatic swings in execution rate: a state in which all cores are busy and slowed to meet the power bound can suddenly become a state in which a subset of cores are idling and their power has been effectively contributed to the remaining cores, which causes them to execute much faster. This execution profile would be a challenging problem for single-processor applications; modeling HPC applications would also have to account for how these local effects influence the behavior of remote processors.

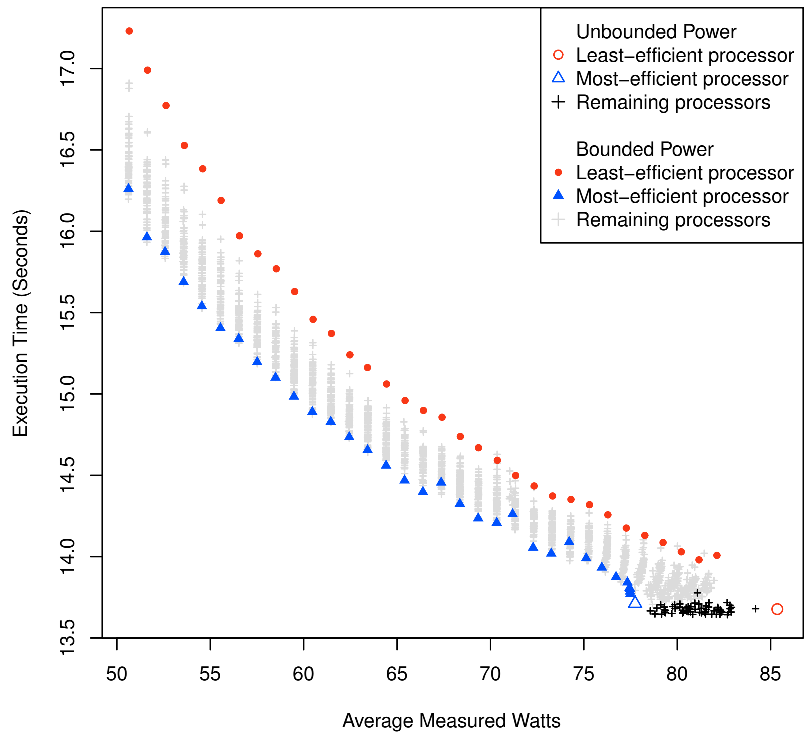

Figure 2. Measured power and performance of the single-processor mg.C.8 benchmark over 64 processors and several power bounds.

B. *Node-level modeling*

Most HPC systems use nodes with multiple sockets. Assuming that the available processors will fall within a range of efficiencies, is it more advantageous to install processors of similar efficiencies on a node or to arrange them so that one processor will be significantly more efficient than the others? The answer will depend on the cluster configuration as well as the characteristics of the applications that are executed on the cluster. For example, application that should maximize memory first and processor counts second would benefit from having unused, less-efficient processors distributed throughout the nodes. Applications that benefit from smaller numbers of faster processors would likely benefit from having the most-efficient processors concentrated in the smallest number of nodes.

C. *System modeling*

If the system has heterogeneous nodes (with regard to processor efficiency), how are those nodes best distributed throughout the system? Segregating nodes by efficiency may provide superior network performance for small-node-count, high-node-power jobs but inferior performance for large-node-count, low-node-power jobs. Rack capacity could also be an issue: a rack designed to match the expected median power draw may not performance as well if it is populated entirely with low-efficiency processors.

D. *Runtime modeling*

Once a system design has been implemented, the user must decide the best way in which to use the system. To do so, the user must not only choose the best number of nodes, but also select which nodes to use, which processors on the nodes, and finally how to go about balancing power consumption throughout the duration of the program. These decisions must be informed by the interplay of power and performance at the system, node and processor levels.

VI. Related Work

Power clamping (or "capping") is by now a well-established if little-used processor feature. In addition to the Intel processor families, power capping is available in both the IBM Power6 and Power7 architectures [3], [4]. The AMD Bulldozer architecture implements power capping by allowing the user to specify a *thermal design power* limit for the processor as well independent DRAM power capping [1].

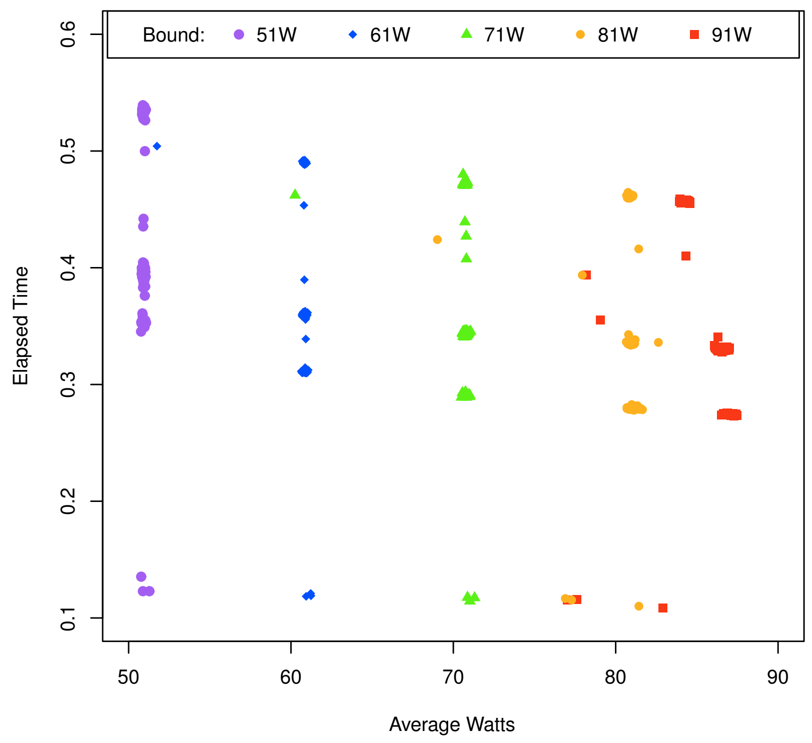

Figure 3. Selected power traces for node 0 over multiple power bounds. (mg.E.256)

Characterizing and mitigating processor power variation has sparked recent interest. Power variation usually changes as processors age, thus requiring continuing characterization during their lifetimes [2]. Rangan et al. [16] observe that variation across cores can be evidenced by variation in the highest frequency that each core can support. Rather than using the minimum frequency across all cores, they recommend classifying the processor using the average core frequency, thus effectively masking core heterogeneity.

Our proposed strategies for saving power by turning off individual processor components is very much in the spirit of existing work. In particular, Ahmed Youssef has designed a Dynamic Sleep Signal Generator that uses runtime traces to predict when functional units can safely be placed in a sleep state [20]. While hardware-enforced power bounds cannot yet be applied to individual cores, Satori and Kumar's work shows demonstrated that control at this level of granularity allowed a larger number of cores to be placed on a processor [17]. Follow-on work explored hierarchical, table-drivent and gradient-ascent techniques for power scheduling that mitigated power bound violations [18], [12].

Several teams have taken advantage of processor variation to create more efficient schedulers. Several teams have examined the combination of processor power variation and DVFS at processor granularity [14] (for the Pentium M architecture) and at the granularity of individual cores [19], [10] (on the AMD Opteron architecture), as well as on more exotic architectures [8], [6]. Herbert et al. [9] took the next step by combining DVFS with work shifting to prioritize use of the most-efficient cores.

Several alternative approaches exist for processor power control. Per-core power gating (effectively shutting off individual cores) has been explored for data center workloads [13]. Cebrain et al. [5] combine DVFS with several additional architectural-level techniques, such as instruction criticality analysis, pipeline throttling, and power-token throttling. Davis et al. [7] have examined the effects of variability in power models used to characterize large-scale clusters.

Two key differences distinguish this work from the foregoing. First, to the best of our knowledge we are the first to specifically target this technology to the high-performance computing domain. Second, for the first time, we show the effects of of power clamping across a significant number of processors. Our future work will apply the lessons learned from both the functional-unit scheduling and multicore scheduling detailed above in order to make the most efficient use of HPC assets.

VII. Conclusion

In this paper we have made the following contributions:

1) Quantification of the power envelope and variation of the most recent Intel server-class processor, the Xeon Sandy Bridge.
2) Explanation of how the Runtime Average Power Limit (RAPL) technology can measure and limit power.
3) Demonstration that a power bound converts variation in processor power to variation in performance.
4) Exploration of the potential of RAPL as a DVFS replacement.
5) Discussion of how RAPL could enable moving beyond power savings and into power scheduling in the high-performance computing domain.

We have only scratched the surface of these features. In particular, we look forward to being able to clamp and to measure power in the DRAM domain as well as to determine how best to use the two clamping windows in the PKG domain. We see RAPL as an enabling technology that will allow us to treat power as a schedulable resource. The greater efficiency realized will in turn allow more resources to be brought to bear for the same amount of power.

References

[1] Advanced Mico Devices. BIOS and kernel developer's guide (BKDG) for AMD family 15h models 00h-0fh processors, Jan. 2012.

[2] L. Ahang, L. S. Bai, R. P. Dick, L. Shang, and R. Joseph. Process variation characterization of chip-level multiprocessors. In *46th Design Automation Conference (DAC)*, July 2009.

[3] B. Behle, N. Bofferding, M. Broyles, C. Eide, M. Floyd, C. Francois, A. Geissler, M. Hollinger, H.-Y. McCreary, C. Rath, T. Rosedahl, G. Silva, and C. Wang. IBM EnergyScale for POWER6 processor-based systems, Oct. 2009.

[4] M. Broyles, C. Francois, A. Geissler, G. Grout, M. Hollinger, T. Rosedahl, G. J. Silva, M. Vanderwiel, J. V. Heuklon, and B. Veale. IBM EnergyScale for POWER7 processor-based systems, Apr. 2011.

[5] J. M. Cebrián, J. L. Aragón, J. M. García, P. Petoumenos, and S. Kaxiras. Efficient microarchitecture policies for accurately adapting to power constraints. In *International Symposium on Parallel and Distributed Processing (IPDPS)*, May 2009.

[6] S. Chandra, A. Raghunathan, and S. Dey. Variation-aware voltage selection. *IEEE Transactions on Very Large Scale Integration Systems*, PP(99):1–12, July 2011.

[7] J. D. Davis, S. Rivoire, M. Goldszmidt, and E. Ardestani. Accounting for variability in large-scale cluster power models. In *Exascale Evaluation and Research Techniques Workshop (EXERT)*, Mar. 2011.

[8] S. Dighe, S. R. Vangal, P. Aseron, S. Kumar, T. Jacob, K. A. Bowman, J. Howard, J. Tschanz, V. Erraguntla, N. Borkar, V. K. De, and S. Borkar. Within-die variation-aware dynamic-voltage-frequency-scaling with optimal core allocation and thread hopping for the 80-core TeraFLOPS processor. *IEEE Journal of Solid-State Circuits*, 46(1):184–193, Jan. 2011.

[9] S. Herbert, S. Garg, and D. Marculescu. Exploiting process variability in voltage/frequency control. *IEEE Transactions on Very Large Scale Integrated Circuits*, PP(99):1, Aug. 2011.

[10] S. Herbert and D. Marculescu. Variation-aware dynamic voltage/frequency scaling. In *15th International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2009.

[11] Intel Corp. *System Programming Guide*, volume 3B-2 of *Intel 64 and IA-32 Architectures Software Developer's Manual*. Dec. 2011.

[12] V. Kontorinis, A. Shayan, R. Kumar, and D. M. Tullsen. Reducing peak power with a table-driven adaptive processor core. In *MICRO*, Dec. 2009.

[13] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis. Power management of datacenter workloads using per-core power gating. *Computer Architecture Letters*, 8(2):48–51, Dec. 2009.

[14] B. Lin, A. Mallik, P. Dinda, G. Memik, and R. Dick. User- and process-driven dynamic voltage and frequency scaling. In *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Apr. 2009.

[15] NASA Advanced Supercomputing Division. *NAS Parallel Benchmark Suite*. http://www.nas.nasa.gov/Resources/Software/npb.html, 2006. Version 3.3.

[16] K. K. Rangan, M. D. Powell, G.-Y. Wei, and D. Brooks. Achieving uniform performance and maximizing throughput in the presence of heterogeneity. In *17th International Symposium on High Performance Computer Architecture (HPCA)*, Feb. 2011.

[17] J. Sartori and R. Kumar. Proactive peak power management for many-core architectures. Technical Report CRHC-07-04, University of Illinois at Urbana-Champaign Center for Reliable and High-Performance Computing, 2007.

[18] J. Sartori and R. Kumar. Distributed peak power management for many-core architectures. In *Design, Automation and Test in Europe (DATE)*, June 2009.

[19] R. Teodorescu and J. Torrellas. Variation-aware application scheduling and power management for chip multiprocessors. In *35th International Symposium on Computer Architecture (ISCA)*, June 2008.

[20] A. Youssef. *Power Management for Deep Submicron Microprocessors*. PhD thesis, University of Waterloo, 2008.