What Computer Architects Need to Know About Memory Throttling

Heather Hanson, Karthick Rajamani

What Computer Architects Need to Know About Memory Throttling

Heather Hanson (hlhanson@us.ibm.com)
Karthick Rajamani (karthick@us.ibm.com)
IBM Research

Abstract

Memory throttling is one technique for power and energy management that is currently available in commercial systems, yet has has received little attention in the architecture community. This paper provides an overview of memory throttling: how it works, how it affects performance, and how it controls power. We provide measured power and performance data with memory throttling on a commercial blade system, and discuss key issues for power management with memory throttling mechanisms.

1 Memory throttling

1.1 Overview

Memory throttling is a power management technique that is currently available in commercial systems and incorporated in several proposed power and thermal control schemes, yet the underlying mechanisms and quantitative effects on power and performance are not widely known.

In a nutshell, memory throttling restricts read and write traffic to main memory as a means of controlling power consumption. A significant fraction of memory power is proportional to read and write bandwidth, and restricting bandwidth creates an upper bound for memory power.

Computer architects should be aware of memory throttling because it manipulates the instruction and data streams that feed the processor cores and can create both performance bottlenecks due to bandwidth overconstriction, as well as opportunities for improving performance through judicious power budgeting.

Simple control with a fixed memory throttle setting allows the full capacity of the memory to be available, with a regulated rate of access to limit power consumption, in enterprise systems with large memory configurations that if left unchecked would exceed the system's power budget.

Altering memory throttling dynamically tailors the access rate to workload demands and variable power allocation, useful for regulating DIMM temperature [1] [2] [3], providing memory power control beyond power-down modes [4], and optimizing system-wide performance or power with techniques such as power shifting [5].

Enforcing a power cap on the memory subsystem will become increasingly important for servers with large memory configurations to support data-intensive applications, and to support virtual systems that pack multiple software stacks–and their bandwidth requirements–into a compact number of processing elements.

Even in systems where memory power is a small fraction of the total, memory throttling provides an essential function: in power-limited situations, every Watt trimmed from the memory budget is a Watt gained for use in processor cores and other components.

1.2 Comparison to CPU clock throttling

At a high level, memory throttling is analogous to the more familiar processor clock throttling. Clock throttling creates a duty cycle for the processor core's clock signal, where a time period is divided into distinct run and hold intervals. During the *run* interval, the clock signal runs freely. During the *hold* interval, the clock signal is gated to remain still, and computation halts.

Figure 1 illustrates run-hold duty cycles for clock throttling, not to scale. The period of one run-hold cycle is typically on the order of thousands of cycles. With long (relative to processor pipelines) duty-cycle periods, clock throttling behavior is a repeating sequence of a burst of activity followed by an idle period. During the burst of activity, processor cores operate normally; during the idle period, computation and memory requests halt and power consumption drops accordingly. In this way, the clock throttling mechanism acts as a governor for the power and performance of the system, with a direct control of the processor and indirect control of the memory system.

Similar to clock throttling's limits on processor cycles within a time interval, memory throttling regulates reads and write accesses within a time interval. There are several approaches to implementing memory throttling features. One technique is similar to clock throttling with run-hold duty cycles, where memory accesses pass through at the requested rate untouched during the run portion of a time interval, then are halted during the hold portion [4] [5]. Another memory throttling mechanism is to periodically insert one or more idle clock cycles between every N^{th} and $(N+1)^{th}$ memory accesses [1], spreading out the accesses and lowering peak bandwidth. Yet another mechanism uses a bit mask to allow or block memory accesses, optimized to minimize interruptions to

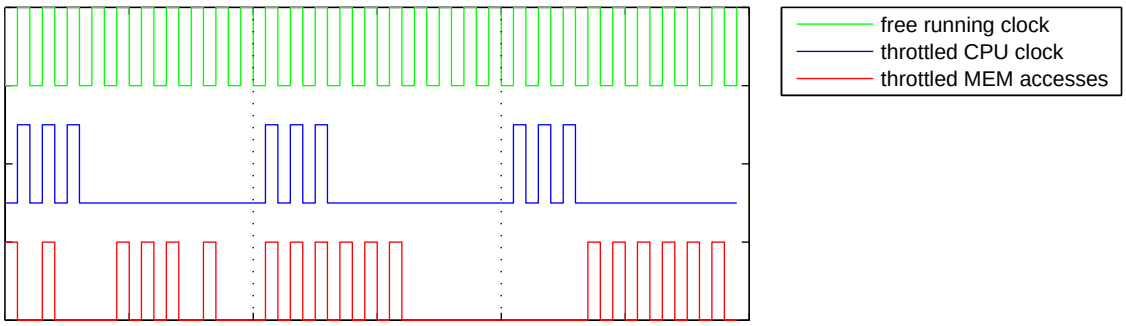

Figure 1: Clock throttling maintains the original frequency during a portion of the time interval; memory throttling allows memory accesses within the time interval up to a quota. In this example, six memory accesses are allowed in each interval, intermittent in the first interval and sequential until the quota in the second and third intervals.

real-time streaming data [6].

In this paper, we focus on an implementation within the memory controller that allows up to N transactions within a time period. The memory throttling time period in this implementation corresponds to a window of 32 frames, approximately 50 ns at a rate of one frame per 667 MHz bus cycle. Once the quota of N reads and writes is reached in a time period, any additional requests must wait for a future time period to proceed. Instructions waiting for the over-quota memory transactions must stall until they are processed. The POWER6 system used in these experiments uses a single memory throttle value universally applied to all DIMMS. Other implementations operate on finer granularities of unique throttle values, such as one memory throttling option on commercially available POWER7 systems that supports unique memory throttle values for each of two channel pairs within a memory controller.

1.3 Power and Performance

There are a few important distinctions to consider for memory throttling with respect to power and performance. First, memory throttling does not alter the DRAM clock in the manner that clock throttling alters the CPU clock signal. Memory throttling is usually implemented within the memory controller (processor chip or chipset) to restrict the memory request rate, and the DRAM itself is unchanged, unlike memory low-power states and power-down modes.

With quota-style memory throttling, the amount of additional latency due to throttling restrictions is a function of requested bandwidth rather than a fixed amount of busy time per interval. A workload whose bandwidth fits within the quota will proceed unchanged by the memory throttle mechanism. A steady workload with bandwidth needs beyond the quota will slow down as it waits for memory transactions to complete at a slower pace. A bursty workload with bandwidth spikes above the quota will effectively be smoothed out as excess transactions are postponed and fulfilled during a later interval; latency for some individual transactions will increase though the average throughput may or may not be affected. Thus, the performance impact of memory throttling depends upon both the bandwidth and the time-varying behavior of workloads.

To the first order, memory power is linearly proportional to the sum of read and write bandwidth. In situations where memory bandwidth requirements fall below the throttle-enforced quota, bandwidth–and thus, power consumption–are unaffected by memory throttling. Rather, the memory throttling mechanism enforces an upper bound on memory power consumption, creating an effective tool for power budgeting.

2 Infrastructure

2.1 System

In this study, we characterized memory throttling with an IBM JS12 blade system [7] with one dual-core POWER6 processor with a 3.8 GHz clock rate, hosting a SLES10 linux operating system. A single memory controller on the POWER6 processor orchestrates reads and writes between both cores and main memory. The memory subsystem has a 16 GB capacity, configured as eight DDR2 667 MHz 2 GB (1Rx4) DIMMS.

2.2 Workloads

We characterized the system response to memory throttling with a set of micro-benchmarks with specific characteristics. Each benchmark maintains steady operation during execution, providing a single point for comparison, without complicating the analysis with phases or other time-varying behavior. This small suite of micro-benchmarks covers a wide range of memory characteristics.

Two micro-benchmarks use the same floating-point workload, DAXPY, with distinct memory footprints. The small data set for DAXPY-L1 fits within the level-1 data cache, while the DAXPY-DIMM footprint is 8 MB,

forcing off-chip memory accesses. By performing the same computation with different data set sizes, we are able to isolate effects due to the memory subsystem behavior.

Computation within the FPMAC kernel is similar in nature to DAXPY; the primary difference is that the floating-point multiply and accumulate algorithm in FPMAC computes and stores with a single array while the DAXPY implementation uses two arrays, providing a different flavor of memory access pattern with the same data set size (8 MB) and similar compute load.

The RandomMemory-DIMM micro-benchmark generates random address locations for reads and writes within an 8 MB memory footprint. The memory access patterns defeat prefetching that would benefit FPMAC and DAXPY kernels' regular access patterns, exposing the full effects of memory latency at each throttle point. The FPMAC, DAXPY, and RandomMemory kernels are short C programs with heavy computational load or intensive memory accesses and very little overhead.

We also use a single calibration phase of the Java benchmark SPECPower_ssj2008 that continuously injects transactions as fast as the system can process them, executing for a fixed period of time to provide insight into the effects of memory throttling on transactional workloads.

2.3 Measurements

The POWER6 Blade system used for the experiments is instrumented with on-board power sensors including memory power, and event counters for throughput in units of instructions per second (IPS), memory reads, and memory writes (among others).

The blade power management controller obtains event counter data via a dedicated I2C port on the processor chip [8], averages the data over 256 ms, and sends measurements via an Ethernet connection to a separate workstation, where a monitoring program dumps a trace file for our analysis, without interfering with the memory characterization experiments.

2.4 Throttling Characterization

We characterized the blade system's response to memory throttling by executing four copies of a micro-benchmark (two cores, two threads each) with a fixed quota-style throttle while we recorded power and performance data, then changed the throttle setting and re-ran the same workload. Throttle values range from 1 to 32, out of a window of 32 accesses. A 100% throttle setting of 32/32 is unthrottled, 16/32 is 50% throttled, etc. As memory throttle settings are successively lower, memory bandwidth is more extensively throttled.

The amount of work performed by DAXPY-L1, DAXPY-DIMM, FPMAC-DIMM, and RandomMemory-DIMM at each throttle setting remained constant, and execution time varied. The single calibration phase of the SPECpower_ssj2008 benchmark executed for a fixed time duration, and the amount of work completed varied with memory throttle setting.

We summarized memory characterization data by calculating the median value for memory bandwidth, throughput (IPS) and memory power over the observed intervals for each permutation of workload and memory throttle setting.

3 Bandwidth

Figure 2 charts memory traffic normalized to the peak traffic observed over this suite, in DAXPY-DIMM. The curves show three distinct regions of operation: *bandwidth-limited* where the bandwidth is a linear function of memory throttle, a *bandwidth-saturated* region, and a *transitional* portion between the limited and saturated regions.

3.1 Bandwidth-limited

In the bandwidth-limited region, increasing the available bandwidth by increasing the throttle value translates directly to higher memory traffic. Changing the throttle value within this region directly affects bandwidth, and thus has a direct effect on both performance and power. The bandwidth-limited region may include only very-throttled operation, such as the case of SPECPower_ssj2008 phase, or a wider range of throttle values, as in the case of the DAXPY-DIMM and FPMAC-DIMM benchmarks with large memory footprints.

3.2 Transition

The transition region is critical for power and performance management with memory throttling. In this region, changing memory throttle setting does affect memory traffic, but in a more complex manner than in the bandwidth-limited region. The uncertainty in the relation between memory throttle and bandwidth within this region, and in the extent of the region itself, create a challenge for managing power and performance.

Memory throttle values that bound transition regions vary by benchmark, with transitions at lower throttle values for less-demanding workloads and higher throttle values for memory-intensive workloads.

Each workload has a gap between the maximum available and *consumed* bandwidth in the transition region, and the extent of the gap varies. For example, at a 30% throttle, RandomMemory-DIMM has not reached its saturation level, yet it consumes less bandwidth than other workloads at the same throttle setting. The knee of the curve is sharper for some workloads than other: the FPMAC-DIMM micro-benchmark has a particularly sharp transition, while the SPECPower_ssj2008 phase has a much more gradual transition. Workloads with sharper transition are able to use more of the available bandwidth at a given throttle setting, up to the point of saturation. Workloads with more gradual bandwidth

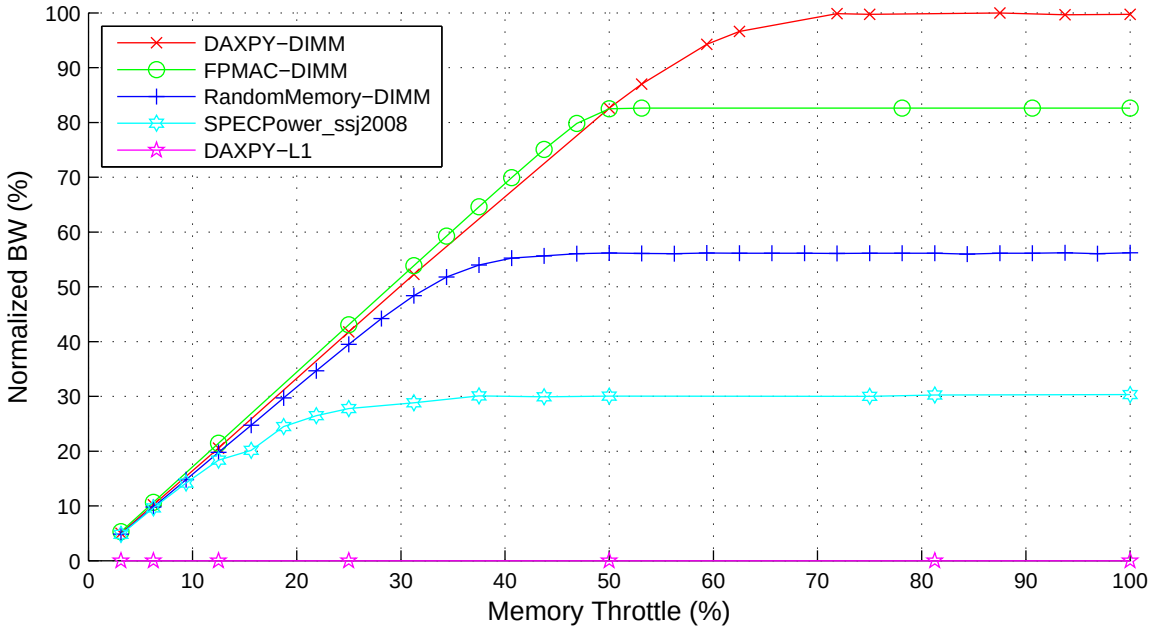

Figure 2: Memory traffic: total read and write accesses normalized to peak observed traffic, unthrottled DAXPY-DIMM. Each workload has unique characteristics for the three throttle regions: bandwidth-limited, transition, and bandwidth-saturated.

roll-off have other bottlenecks that also factor into limiting the rate of memory requests.

While the bandwidth regions are clearly visible in off-line analysis of the full throttle range, it is difficult at run time to discern whether the current workload(s) are in the transitional area. For example, at the 50% throttle level in Figure 2, the observed bandwidth for DAXPY-DIMM and FPMAC-DIMM workloads are nearly identical, yet DAXPY-DIMM is in the linear bandwidth-limited region and FPMAC-DIMM is at a very sharp transition point. Without knowing the bandwidth trends from neighboring throttle points, a controller would not know whether to expect a linear, non-linear, or no change in bandwidth for an incremental change in throttle value.

3.3 Bandwidth-saturated

In the bandwidth-saturated region, the flat portions of each curve in the graph, memory traffic does not change with memory throttle settings. Each workload settles to a unique saturation level. Other bottlenecks and the workload's data footprint limit the memory request rate.

On the blade system used to collect the data shown in Figure 2, the memory bus limits the available bandwidth for memory throttle settings 75% and above, meaning that throttle settings between 75-100% provide essentially the same amount of available bandwidth.

The DAXPY benchmarks illustrate two ends of the saturation spectrum. Bandwidth consumed by DAXPY-DIMM approaches the architectural limit of the memory bus; the other benchmarks in this collection are limited by other factors. The cache-resident dataset of DAXPY-L1 naturally limits its memory request rate, and the consumed bandwidth is so low that it is independent of the memory throttle setting, essentially in the bandwidth-saturated region throughout the entire range.

Increasing memory throttle settings beyond the saturation level has a negligible effect on bandwidth. It follows that increasing memory throttle settings beyond the saturation level will not improve performance, or draw more power. For example, no memory throttle setting would have any bearing on DAXPY-L1, nor would modulating memory throttle settings between 40-100% for the SPECPower_ssj2008 calibration phase recorded in Figure 2.

4 Performance

Figure 3 plots performance (IPS) as a function of the memory throttle setting. Data are normalized to the peak throughput of individual benchmarks to factor out the effect of disparate throughput levels among benchmarks in the suite.

One advantage of power-cap control rather than continuous control is that when memory traffic is less than the limit imposed by memory throttling, performance is unchanged. Cache-resident DAXPY-L1 throughput is unaffected by memory throttling.

At the opposite end of the spectrum, DAXPY-DIMM shows noticeable throughput loss for throttling up to 65%. Remember that at 75%, the memory bus becomes the dominant bandwidth-limiting factor. DAXPY-DIMM (and other workloads with similar characteris-

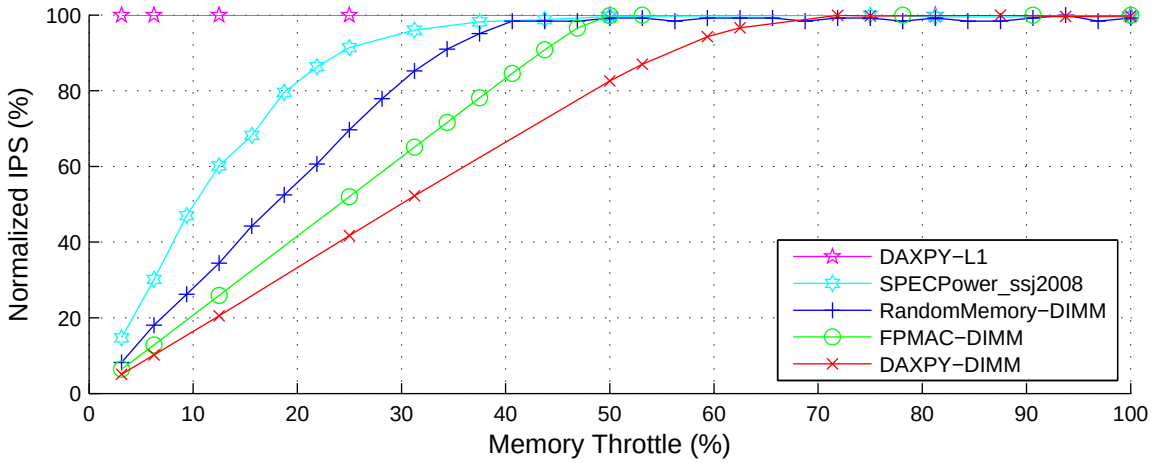

Figure 3: Throughput (instructions per second), each benchmark normalized to its own peak.

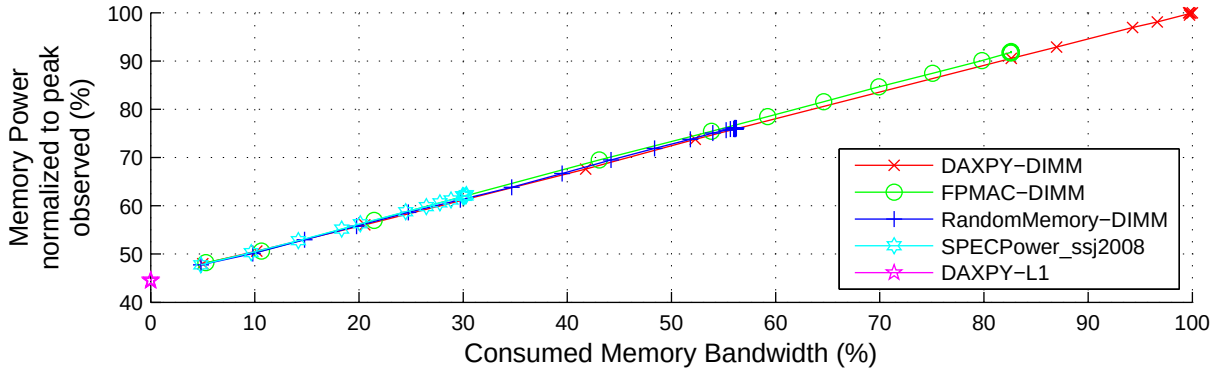

Figure 4: Memory power is a linear function of bandwidth. In this system, about 60% of memory power is controlled by memory throttling.

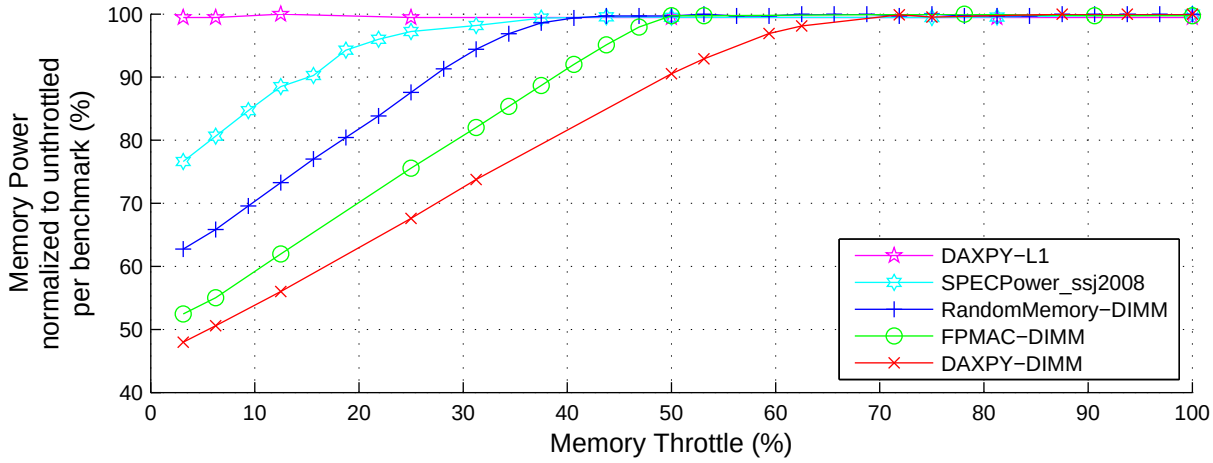

Figure 5: Relationship between memory power and throttle varies by workload.

tics) are sensitive to the majority of memory throttling values in the useful range up to 75%, and almost any power control via memory throttling would directly degrade performance.

SPECPower_ssj2008 tolerates memory throttling without serious performance loss down to about 40% throttled, below which it has non-linear performance loss with memory throttling throughout its wide transition region. Kernels like FPMAC-DIMM with very short transition regions are dominated by linear performance loss in the bandwidth-limited region.

Workloads with a sharp-knee characteristic would show no response to changes in memory throttling in their bandwidth-saturation regions, until an incremental step down in memory throttling level tipped them into a bandwidth-limited region and suddenly dropped in performance.

5 Power

Figure 4 confirms that memory power consumption is linearly proportional to memory bandwidth on our system. Data are normalized to the maximum observed memory power measurement, in DAXPY-DIMM. The near-zero bandwidth requirements of the cache-resident DAXPY-L1 show that about 40% of memory power is *not* under throttle control in this system.

Measured memory power data points normalized individually per benchmark, shown in Figure 5, demonstrate where opportunity for power control lies. Memory throttling offers essentially no power control for core-bound workloads such as DAXPY-L1, a small range of control for moderate-intensity workloads such as SPECPower_ssj2008, and a large swing for memory intensive workloads such as DAXPY-DIMM and FPMAC-DIMM that have larger unthrottled memory power consumption.

Since quota-style memory throttling enforces an upper bound on power consumption, actual memory power consumption will be in the range between the static power levels and the memory power cap, depending upon run-time bandwidth demands.

6 Conclusion

Memory throttling exists in various forms in commercial systems, yet it has garnered little attention in architecture studies to date. Memory throttling can be used to enforce memory power budgets, enabling large memory configurations that would violate power constraints if left unthrottled, and also supporting dynamic techniques such as power shifting.

As with nearly all power management options, memory throttling comes with the price of performance penalties in some situations. We point out the regimes of power control with no performance loss, and where more extensive power reduction does degrade performance.

This paper characterized the effects of memory throttling on throughput performance and memory power on a commercial blade server system. We demonstrated the three regions of bandwidth response: *bandwidth-limited*, *transition*, and *bandwidth-saturation*. Understanding these regions and the workload characteristics that determine the interaction between throttle settings and bandwidth restriction enables wise choices in memory power management design.

7 Acknowledgments

Thank you to our colleagues who contributed technical expertise and system administration support, especially Joab Henderson, Guillermo Silva, Kenneth Wright, and the power-aware systems department of the IBM Austin Research Laboratory.

References

[1] C.-H. R. Wu, "U.S. patent 7352641: Dynamic memory throttling for power and thermal limitations." Sun Microsystems, Inc., issued 2008.

[2] J. Iyer, C. L. Hall, J. Shi, and Y. Huang, "System memory power and thermal management in platforms built on Intel Centrino Duo Mobile Technology," *Intel Technology Journal*, vol. 10, May 2006.

[3] E. C. Sampson, A. Navale, and D. M. Puffer, "U.S. patent 6871119: Filter based throttling." Intel Corporation, issued 2005.

[4] I. Hur and C. Lin, "A comprehensive approach to DRAM power management," in *Proceedings of the 14th International Symposium on High Performance Computer Architecture (HPCA '08)*, August 2008.

[5] W. Felter, K. Rajamani, C. Rusu, and T. Keller, "A Performance-Conserving Approach for Reducing Peak Power Consumption in Server Systems," in *Proceedings of the 19th ACM International Conference on Supercomputing*, June 2005.

[6] O. Kahn and E. Birenzwig, "U.S. patent 6662278: Adaptive throttling of memory accesses, such as throttling RDRAM accesses in a real-time system." Intel Corporation, issued 2003.

[7] International Business Machines, Inc., "IBM BladeCenter JS12 Express blade product description.." Available at ftp://public.dhe.ibm.comcommon/ssi/pm/sp/n/bld03013usen/-BLD03013USEN.PDF, 2008.

[8] M. S. Floyd, S. Ghiasi, T. W. Keller, K. Rajamani, F. L. Rawson, J. C. Rubio, and M. S. Ware, "System power management support in the IBM POWER6 microprocessor," *IBM Journal of Research and Development*, vol. 51, pp. 733–746, November 2007.