RAPL: Memory Power Estimation and Capping

Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna and Christian Le
Intel Corporation, 2111 NE 25th Ave.
Hillsboro, OR 97124, USA
(howard.david, eugene.gorbatov, ulf.r.hanebutte, rahul.khanna, christian.le)@intel.com

ABSTRACT

The drive for higher performance and energy efficiency in data-centers has influenced trends toward increased power and cooling requirements in the facilities. Since enterprise servers rarely operate at their peak capacity, efficient power capping is deemed as a critical component of modern enterprise computing environments. In this paper we propose a new power measurement and power limiting architecture for main memory. Specifically, we describe a new approach for measuring memory power and demonstrate its applicability to a novel power limiting algorithm. We implement and evaluate our approach in the modern servers and show that we achieve up to 40% lower performance impact when compared to the state-of-art baseline across the power limiting range.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design Studies

General Terms

Measurement, Performance, Design, Experimentation

1 Introduction

Growth in server performance and ever increasing densities have created an upsurge of power consumption in data-centers [1, 2]. This growth has put considerable pressure on cooling and power delivery capacity which has driven up fixed infrastructure costs and operational expenses. As a result of these escalating trends, server and data-center energy efficiency has been pushed to the forefront of systems research and product design and architecture [3, 4, 5].

Several studies have explored using power limiting as a mechanism for managing power and thermal capacity in the data-center [6, 7]. Our work is motivated by statistical properties of server workloads whose average power is appreciably lower than their peak power. We exploit these properties and increase data-center density by provisioning power and cooling capacity according to worst case average rather than peak usage. System power limiting is used to manage excursions and maintain acceptable power and thermal load at the facility. Limiting system power can introduce performance inversions and significantly impact Service Level Agreements (SLA) at the datacenter level. Challenges in power limiting, power management and service level agreements postulate real-time visibility into system power consumption and efficient algorithms to enforce power limits while maintaining acceptable levels of system performance.

Prior studies have investigated various strategies to measure and limit processor power [8, 9, 10, 11]. While processor power limiting is paramount to high-performance multi-core designs used in server systems, the rapid growth in higher memory capacity and bandwidth driven by these designs has made memory a key target for power limiting [12, 13]. With processors maintaining same power envelope and memory accounting for 25% of active system power in the modern servers, this trend is expected to continue well into the future.

This paper presents a novel approach to memory power limiting. We provide a comprehensive definition and evaluation of memory power estimation and limiting algorithm that significantly improves sensing accuracy, power limit enforcement and system performance. We discuss a mechanism for measuring DIMM power using a set of observable activity variables that share a tight correlation to the power consumption. Furthermore, we describe and evaluate a novel Running Average Power Limiting (RAPL) scheme for memory sub-system that can simultaneously enforce multiple memory power limits.

Section (2) describes the related work on memory power measurement and limiting, section (3) introduces the basics of memory power/performance and discusses the examples of memory power consumption and bandwidth in server systems, section (4) describes new and traditional approach for measuring memory power in servers, section (5) discusses various memory power limiting techniques applied in this paper, section (6) analyzes the experimental results.

2 Related Work

Recent studies have addressed several facets of processor power limiting and budgeting. One of the earlier studies by Bellosa et. al [20] demonstrate strong correlations between performance events (IPC, memory references, cache references) and power consumption in the Pentium II. Bircher et. al. [14] proposes the use of microprocessor performance counters for on-line measurement of a complete system power consumption by taking advantage of the ”trickle-down” effect of performance events in a microprocessor. Kontorinis, et al [8] propose an adaptive mechanism that configures different components of the CPU to meet peak power constraints while minimizing performance degradation. In [9], the authors propose to use analytical models based on DVFS and cache adaptation to predict and control processor power and performance. Isci, et al [10] describes a mechanism for meeting a global power budget in a multicore processor while minimizing performance loss. They rely on power sensors to measure processor power and adjust its execution characteristics to meet global power constraints. Our work differs from these contributions in that we employ memory power limiting approach and expose it as a first-order control parameter that can be used by system software or platform infrastructure to set different power limits that correspond to associated physical constraints.

Several studies have shown the importance of limiting memory power in enterprise server systems [15, 16]. Diniz et al [12] propose a set of throttling techniques that limit memory power consumption by adjusting the power states of memory devices based on bandwidth and command power estimates. Their power control algorithms aim at limiting instantaneous memory power. We believe these schemes can be too restrictive in their control of memory power and do not represent physical

ISLPED'10, August 18–20, 2010, Austin, Texas, USA.
Copyright 2010 ACM 978-1-4503-0146-6/10/08 ...$10.00.

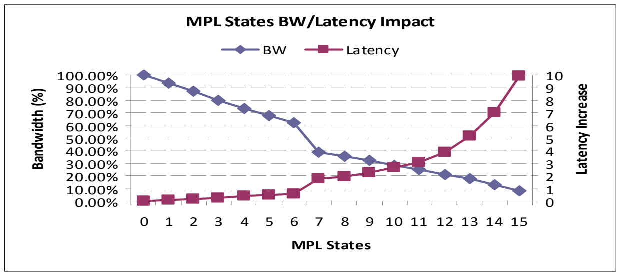

Figure 1: MPL States Bandwidth/Latency Impact.

constraints that arise in real server deployments and datacenters. The overly aggressive power limiting can also have a negative impact on system performance. Felter et al. [11] proposed a method to dynamically allocate available power between the processor and memory subsystems by controlling the number of processor instructions and memory accesses. Similar to our approach, the authors propose a mechanism that limits the number of operations performed by each subsystem over an interval of time. Our work differs from their approach in three fundamental ways:

(1) We expose the time interval to software and platform making it dynamically configurable. Furthermore, we allow multiple limits to be set simultaneously to meet different thermal and power constraints that arise in real physical deployments. Setting the time interval statically and choosing the lowest common denominator can either cause power excursions or unnecessary performance degradation.

(2) We propose a novel running average power limiting algorithm that controls memory power while maximizing bandwidth and smoothing the effects of bandwidth limiting.

(3)We evaluate our power limiting algorithm on the latest generation platform, its characteristics and performance benefit in the context of complex system interactions that cannot be easily modeled through simulation.

3 Background

DRAM power consumption comprises of background power, operation power, read/write power and I/O power. Background power describe DRAM chip consumption with or without memory accesses. Modern DRAMs support several low power states that can reduce the background power when a chip is idle or no operations scheduled for it. The DDR3 background power has three state with decreasing power consumption: standby, power-down and self-refresh. A control signal CKE (Clock Enable) is de-asserted in power-down and self-refresh states where no commands can be issued to the chip.

Operation power is consumed when a chip is active and performs precharge operations. Read/write power is consumed when data is read out or written into a chip. Finally, I/O power is the power used for driving the data bus and terminating signals from other ranks if necessary. Limiting memory power entails restricting its operation power, read/write power and potentially lowering its background power. Restricting operation and read/write power is done by controlling memory bandwidth by either increasing command timing parameters or regulating the number of activates, reads and writes going through memory. Reducing background power is accomplished by putting DRAM memory into lower power states when idle interval is sufficiently long. For example, if idle interval is on the order of hundreds of clocks memory can be put into a power down mode while if it is on the order if hundreds of microseconds it can be put into self-refresh.

As Table 1 shows we have defined sixteen Memory Power Limiting (MPL) states for evaluating different power limiting algorithms in this paper. The states provide robust dynamic range for memory power while staying within the constraints of current memory controller design and implementation. The first seven states (MPL0 - MPL6) adjust back-to-back CAS timing to achieve specific worst case memory power targets. The last eight states (MPL7-MPL15) delays the issuance of commands by the memory controller, thereby forcing DRAM idle windows during these intervals. Table 1 shows all MPL states with their relative worst case power and percentage of maximum bandwidth. Increasing memory timing or delaying commands from going to memory can have a significant impact on memory bandwidth and latency. To illustrate this point, Figure 1 shows sensitivity of memory latency and bandwidth for a synthetic memory micro benchmark. As can be seen from the figure the impact on this memory intensive workload is quite dramatic with MPL15 only providing 10% of bandwidth and increasing latency by a factor of 10 when compared to the unconstrained case (MPL0).

MPL STATES	BANDWIDTH (%)	POWER (W)
MPL0	100	11.9
MPL1	80	10.5
MPL2	66.67	9.46
MPL3	57.14	8.70
MPL4	50.0	8.12
MPL5	44.44	7.67
MPL6	40.0	7.30
MPL7	36.16	6.31
MPL8	33.33	6.01
MPL9	30.25	5.69
MPL10	27.07	5.35
MPL11	23.58	4.99
MPL12	20.0	4.61
MPL13	16.34	4.23
MPL14	12.33	3.80
MPL15	8.24	3.37

Table 1: Memory Power Limiting (MPL) states for evaluating different power limiting algorithms

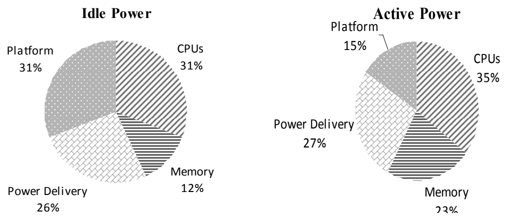

Figure 2: System power Distribution for Idle Power and Active Power

As will be discussed in the next section the goal of a robust power limiting algorithm is to lessen or potentially eliminate this impact. As the chart in Figure 2 shows memory accounts for only 12% of system idle power in the modern server platform but can contribute as much as 25% in an active server system. Given that CPU and memory are the only two components that provide high dynamic range and expose control interfaces to limit their power consumption it is important to address both when designing platform power limiting.

For example, as shown in Figure 2, processors consume a little over 30% of the system power while processors and memory together account for roughly 60% of the total power. If we assume that processors support 50% dynamic range for power limiting, system dynamic range would be 15% if processors alone were used to limit platform power. When we extend power limiting to memory, system dynamic range grows to 25% or around 100W in our system, a substantial increase that gives larger controllable range at the system and datacenter. Following sections describe memory power measurement and limiting algorithms for memory power limiting control.

4 Measuring Memory Power

This section describe our power measurement methodology. Fundamentally, we can obtain real time memory power measurements in two ways: (1) a direct approach utilizing circuit-based mechanism such as current sense resistors and (2) an indirect approach based on a memory power model which uses a set of activity counters and predefined weights. We first describe the direct method which provides a reference point against which activity based power models evaluated.

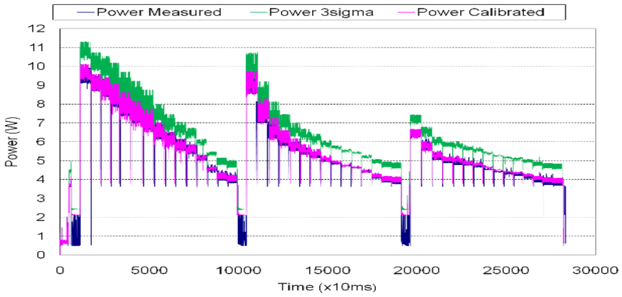

Figure 3: Memory Power Comparison Measured vs. 3 Sigma vs. Calibrated

4.1 Direct Measurement Methodology

The reference power measurement used in this paper employed a memory power riser to directly measure the DIMM power with an Agilent 3470 data acquisition system at a 10ms cadence. For precise and repeatable testing, we utilized a synthetic memory power stress tool to create different memory traffic patterns which covered full memory bandwidth range. Figure 3 shows power profiles for a dual rank x4 DDR3 1333 4GByte memory module. The first of the three graphs has been obtained by the direct measurement approach, while the two others are obtained by the power model as described in the following section.

The graphs in Figure 3 show three sets of experiments. The first one was measured using typical enterprise memory access pattern of 67% reads and 33% writes with a closed page policy. The second and third experiments were done using 100% reads with open and closed page policy respectively. These two experiments were used to fully characterize DRAM activation power across multiple conditions that capture both open and closed page behavior. In all of the cases hardware prefetchers were disabled and power was measured over the entire bandwidth range. As can be seen from the figure DIMM power ranges between 11W for the read/write case to 3.6W for the background or active standby power with clock enable high (CKE_High). During OS idle, Intel$^{®}$ $Xeon^{TM}$ 5500 processor resides in low power C6 state [17] with memory in self refresh consuming around .6 Watts per DIMM.

Not surprisingly, direct approach described above provides data of the highest quality with any inaccuracy coming only from the instrument's own noise and error. However, using such an approach in real systems comes at substantial cost and complexity that cannot be justified in a high volume server platform. To lower cost and limit design complexity state-of-art Voltage Regulators (VRs) provide current/voltage sensing mechanisms that are accurate only at the high end of the load line where memory bandwidth is close to its peak. Since most server benchmarks operate at much lower bandwidth, the accuracy of the power sensing mechanisms degrades in these designs requiring additional guard-bands at the low end of the load line. In addition, the need for a high speed interface to a VR or other sensing mechanism further adds to system design complexity. To deal with these challenges this paper evaluates an alternative approach for memory power sensing based on a power model and activity counters that can be used in place of instrumented power measurements.

4.2 Memory Power Model Approach

Measuring memory power using a power model based on activity counters and predefined weights provides several key advantages. First, it can be easily integrated into existing memory controller hardware with minimal cost and impact on area and power. Second, it enables high resolution sampling and measurement frequency with minimal design complexity. Finally, we can measure at very fine granularity starting at the individual bank or rank level and going up to the DIMM, channel or memory controller level. The challenge in developing such models is ensuring their accuracy and addressing corner cases that are not easily captured by the model.

Power model accuracy is a function of activity counters and their associated weights. Typically, it involves the definition of the power model, the selection of the activity counters and the methodology for creating the weights in order to achieve accurate power estimates which would closely track direct measurements. Prior work has looked extensively at using activity based modeling for processor power estimation, with its benefits and shortcomings well studied and documented [19, 20]. While estimating processor power is particularly challenging due to its complex architecture, interaction and dependencies between different components and the presence of multiple corner cases, memory is significantly more amenable to activity based power modeling. Memory provides a well defined and isolated set of activities shown in Table 2 that exercise particular components and interfaces in the DRAM design. The memory controller we use in our experimental setup provides counters for every activity shown in Table 2.

The weights for the power model can be determined in several different ways. One approach is to pre-compute the weights offline for each activity based on the information provided by different DRAM vendors. In order to account for vendor to vendor and manufacturing process variations and guarantee model robustness, statistical correction needs to be applied to take into account 3-sigma variations in the weights [21]. The advantage of this approach is that it does not require a power sensing mechanism, significantly reducing system cost and design complexity. The biggest drawback is its use of 3-sigma values for the weights that results in very large guard-bands that make measurements less accurate for most DIMM configurations and DRAM activity patters.

The approach we propose in this paper uses an alternative mechanism for computing the weights. It relies on a power sensor to calibrate the weights during system boot time. When system memory is initialized, BIOS runs a set of synthetic memory tests that generate continuous memory traffic for each of the activities used in the power model. Note that the design of the power sensing mechanism can be relatively simple since it needs to be accurate only at high bandwidth and the sensing frequency can be coarse due to the static nature of these synthetic tests. Once memory power is measured for each of the activities the weights are selected to minimize the overall error of the power model.

ACTIVITY	WEIGHTS	Units
Activate (A)	36.3	nj/Activate
Read (R)	22.8	nj/Read
Write (W)	21.7	nj/Write
CKE=HIGH adder	2112	mW
CKE=LOW baseline	1663	mW

Table 2: Memory Power Model Weights

There are several approaches to designing a calibration model that can be used to determine the activity weights. This paper uses Genetic Algorithms (GA) [18] to build parametric model for identification and calibration of activity based power estimation. We have selected GA framework due to its ability to quickly converge to a local minima in the presence of large and complex datasets. The memory energy weights were fitted with GA for all activities shown in Table 2. The calibrated weights for activate/pre-charge, read and write were 36, 23 and 22 nJ/sec respectively (Table 2). These calibrated weights and activity counters are fed into a power model (Equation 1) to estimate memory power (along with baseline power K) during system operation:

$$\begin{aligned} Power &= W_{act}\cdot A + W_{read}\cdot R + W_{write}\cdot W \\ &+ W_{CKE\_H}\cdot CKE\_H_{adder} + K \end{aligned} \tag{1}$$

Figure 3, shows the accuracy that can be achieve with a power model based on Calibrated Weights (CW). While 3-sigma model shows significant error when compared to direct power measurements, the CW model tracks real measurements very closely across the entire bandwidth range. More specifically, the calibrated estimated power is within 1% with a standard deviation of 1.1% of the reference measurement, while for the 3-σ model the average error was computed to be 16.0% with a standard deviation of 3.0%.

5 Running Average Power Limit

Running Average Power Limit (RAPL) provides a standard interface for limiting memory power by HW, OS, server applications or platform/datacenter manageability agent. Unlike previously proposed mechanisms that maintain instantaneous power limits, RAPL maintains an average power limit over a sliding time window. The power limit and the time window ($Power$, $TimeWindow$) form essential parameters of RAPL memory interface. Multiple limits can be set simultaneously for different power and thermal constraints.

Specifying power limit as an average over a time window represents physical power and thermal constraints present at the system and datacenter levels. The exact window size is a function of the underlying physical constraint and together with the power limit is set to maintain robust, efficient and safe operation. In practice, the window size can vary between milliseconds to satisfy memory power delivery constraints to seconds for managing memory thermals.

Enforcing power limits over a time widow reduces performance impact in highly dynamic and transient datacenter workloads. Rather than setting instantaneous limits, RAPL maintains energy credits, which are traded to fulfill memory performance demands and accumulated when that demand is low. If average workload memory bandwidth requirements are within the specified power limits the system will not experience any performance degradation even though its memory demand over short periods of time may well exceed the average power limit.

Memory RAPL architecture comprises of three principal components: power measurement logic, power limiting algorithm and memory power limiting control. Power measurement logic provides an accurate mechanism for measuring memory power. We use calibrated weights (section 4.2) to implement memory power measurement logic. Power limiting algorithm tracks memory energy consumption over a sliding time window and determines available power budget for the next interval. The algorithm aims to deterministically maintain a power limit while maximizing memory bandwidth and performance.

Once the power limit is established RAPL uses memory power limiting control mechanism to enforce the limit. In this paper we use sixteen memory power states (Table 1) that we call M states to control memory bandwidth and limit its operational power. At the beginning of each time interval RAPL takes available power budget determined by the power limiting algorithm described in the next section and maps it to a an M state that doesn't exceed this budget. We next describe the algorithm details and the process for selecting M states.

5.1 RAPL Algorithm

RAPL algorithm determines the power budget based on memory bandwidth, specified power limit and time window. The algorithm operates on a fundamental tick that defines the time interval over which the power is measured and the limit is enforced. In this paper, we use the tick rate to 1 KHz (1 ms time interval). The window size N is the number of these time intervals in sliding time window specified in the RAPL interface.

RAPL maintains an energy budget and a history of power consumption over the time window. Starting at each interval it computes the budget by subtracting the power consumed over the last N-1 intervals from the available power budget:

$$PwrBudget_{hard} = N \cdot PwrLimit - \sum_{i=1}^{N-1} MemoryPwr_i \quad (2)$$

Upon computing power budget, RAPL searches M-state table to determine the state that will not exceed available power budget. Note, that the power levels shown for each M-state in Table 1 represent worst case power based on worst case memory bandwidth and access pattern allowed in this state. The M-states provide hard guarantees on never exceeding their power limits. Once the M-state is set, memory controller monitors memory bandwidth and limits it when it exceeds specified level.

While the algorithm above enforces the power limit and always tries to allocate all of available bandwidth to increase system performance it suffers from several drawbacks. First, it uses worst case power in determining M-state for the next interval. Worst case power corresponds to worst case memory traffic in terms of bandwidth utilization, page hits and read/write distribution and is unlikely to represent any realistic workload or its true memory power consumption. Second, the algorithm uses the entire window to compute the budget for the next interval. We call this hard power budget since it provides strict power limiting guarantees. While hard budgets ensure power limits, they don't represent the most recent workload phase behavior and tend to allocate the entire available budget in single intervals resulting in significant memory bandwidth swings over time. This is a highly undesirable effect for many multithreaded server workloads in multi-core systems that would like to provide stable and repeatable execution across multiple threads over time. The goal of an efficient RAPL algorithm would be to smooth the effects of memory power limiting and allow for more graceful impact on memory bandwidth.

To address these issues we propose the RAPL algorithm that maintains two types of limits: soft and hard limits. Hard limit are used to provide deterministic guarantees and are computed using Equation 2. Hard limits use the entire window N of past power consumption history and allocate all available budget to the next time interval. Soft limits shift the time window by M intervals to capture the most recent workload behavior and smooth the effects of power limiting by predicting average bandwidth demand over the next M time intervals:

$$PwrBudget_{soft} = \frac{N \cdot PwrLimit - \sum_{i=M}^{N-1} MemoryPwr_i}{M} \quad (3)$$

Further, instead of using worst case power for determining the next M-state, soft limits use Weighted Running Average (WRA) of measured memory power for each of the M states. Soft limits shadow values shown in Table 1 and update them at the end of each time interval using the following formula:

$$WRA^{new}[M] = \alpha \cdot MemoryPwr + (1 - \alpha) \cdot WRA^{curr}[M] \quad (4)$$

After computing hard and soft limits and updating the corresponding WRA entry, RAPL searches the shadow M-state table to find the M state that does not violate that limit. This selection represents the best and most graceful memory working point based on the latest workload behavior. However, this M-state selection based on soft limits does not guarantee that overall power limit will not be exceeded. For this, RAPL references the M-state table to ensure that the worst case power will not exceed the hard limit. If the hard limit is exceeded, RAPL will walk the M-state table down until it finds a power state that meets hard limit requirements. Note, that in most scenarios soft limits are sufficient to meet the overall power budget given their tendency to spread available bandwidth over multiple intervals. This ensures graceful and efficient operation of the RAPL algorithm while adhering to specified power limits.

6 Evaluation

In this section, we evaluate the RAPL algorithm and demonstrate the benefit of this algorithm vs. a static power limiting approach. We describe our methodology in Section (6.1) and present results based on our experimental setup (section 6.2).

6.1 Methodology

To evaluate memory power limiting we developed a Software (SW) based experimental setup utilizing a standard $Xeon^{TM}$ 5500 platform. The SW is not only used to run the power limiting algorithm, but also for data sampling of pertinent performance and state counters. Furthermore, we implement the Baseline and RAPL algorithms such that equal computational steps and identical number and type of driver function calls are executed. The baseline algorithm selects a fixed MPL state that guaranties that the worst case power condition is below the power limit. In contrast, the RAPL algorithm dynamically selects one of the 16 MPL states as described earlier.

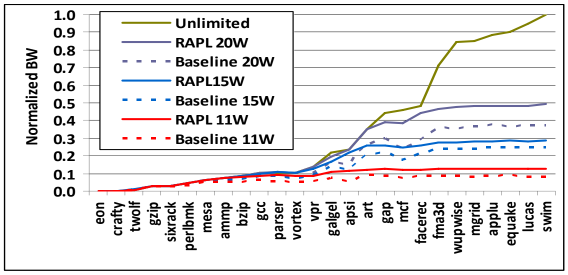

Figure 4: SPECCPU2000 normalized bandwidth at different power limits.

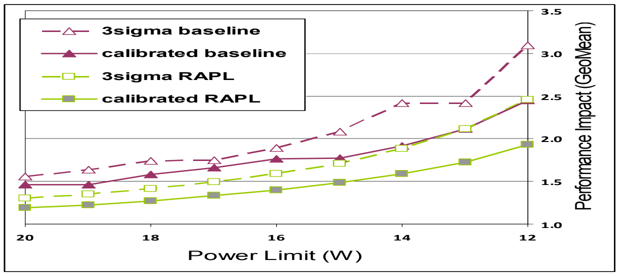

Figure 6: SPECCPU2000 performance impact as function of power limit.

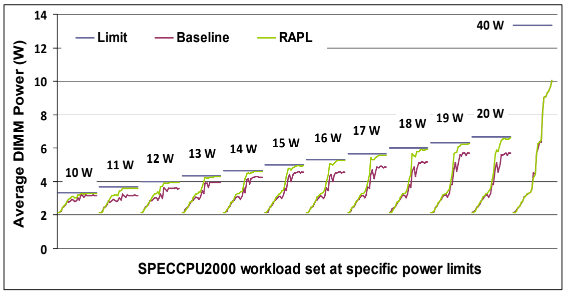

Figure 5: SPECCPU2000 average DIMM power for workload set at different power limits.

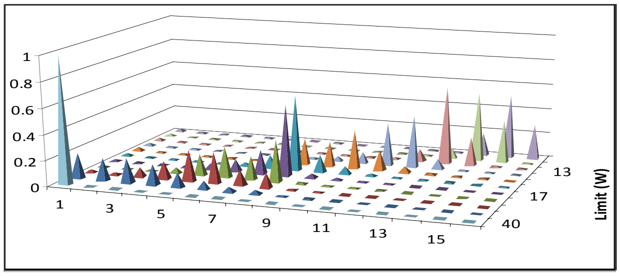

Figure 7: RAPL MPL distribution for SPECCPU2000 facerec.

6.2 Results

Experiments were carried out using SPEC-CPU-2000 and SPEC-jbb benchmark sets on a single socket Intel® $Xeon^{TM}$ Processor X5570 platform (2.93GHz, 4 cores, symmetric multithreading disabled) with 3 memory channels running the Linux operating system. For the memory configuration we choose a common 1 DIMM per channel configuration with 4GB DIMMS running 1333MHz frequency for a total of 12GB. In the following section the power limit is given for the 3 DIMMS combined, while power graphs show the per DIMM power. Each of the 26 sub benchmarks of SPECCPU2000 was executed in SPEC-rate fashion, i.e. 4 copies of the workload are submitted simultaneously for parallel execution on all 4 cores.

Figure 4 shows the normalized average bandwidth for each of the SPECCPU2000 sub-benchmarks at different power limits for baseline and RAPL algorithms using calibrated weights. For Figure 4 and 5 the workloads have been ordered according to average bandwidth when no limit is applied. For clarity, both graphs show the data for the 26 benchmarks as contiguous lines, even so the data points are independent of each other and a different ordering would result in a different shape of the graphs. The 26 sub benchmarks can be grouped into 3 sets. About 46% of the workloads require very low bandwidth, while about 23% are high bandwidth workloads with the remainder being in between. The power limiting does affect the workloads differently, with the high bandwidth workload being significantly reduced, while the bandwidth of the low bandwidth workload set is not affected even when an 11W RAPL limit is applied. The static limiting (dotted lines), compared to the RAPL limiting (solid lines), results in more severe bandwidth reduction across the board at all power limits.

Figure 5 shows the average DIMM power for each SPEC-CPU2000 sub-benchmark at different power limits for the baseline (red) and RAPL (green) algorithms using calibrated weights. Twelve sets of measurements are shows. When not limited, the average DIMM power values of the 26 workloads ranges from 2W (background power) to 10W (background + high operation power). Applying a power limit changes the average power curves differently. With increasingly lower power limit, more and more workloads have an average DIMM power consumption that hits the power limit. In general, the baseline algorithm results in lower average power compared to RAPL, i.e. not utilizing the provisioned power as well as RAPL.

Figure 6 compares the impact of power limiting on the workload performance for the 2 limiting algorithms (baseline and RAPL) and the 2 sets of weights (With 3-σ and calibrated) when memory power is being limit between 11W and 20W. The data represents the geometrical mean of the 26 sub benchmarks. Performance impact is defined as the ratio of runtime when limited to unlimited runtime. The 4 curves line up next to each other; RAPL lower impact than baseline; calibrated weights lower impact than With 3-σ weights. Thus, baseline algorithm using With 3-σ weights results in the highest performance impact and the RAPL algorithm with calibrated in the least. At 15W limit the RAPL algorithm with calibrated weights results in a geometrical mean of 1.5 compared to over 2.0 for baseline algorithm using With 3-σ weights. The With 3-σ weights overestimate the memory power and thus result in more severe limiting which in turn increases the performance impact. This effect is greater with lower power limits which one can see in the shape of the curves which are steeper toward the right.

The fundamental difference between the RAPL and baseline algorithm is dynamic selection of the MPL state in the RAPL algorithm vs. a static MPL state for the baseline algorithm. In Figure 7 and 8 the MPL distributions at different power limits for the RAPL algorithm with calibrated weight is shown for 2 SPECCPU2000 sub-benchmark, facerec and apsi respectively. The 3d graphs should be read as a stack of cards sorted by power limit (40W, 20W - 12W), i.e. the first card shows the distribution at 40W memory power, which is the unlimited case. For the unlimited case 100% of the time is spent in MPL 0 as the spike of length 1 shows. With stricter power limiting, as expected, more and more time is spent in higher limiting stages. It is important to noticed that significant time is also spent in lower MPL states and that the distributions are highly workload depenent. E.g. for apsi even at 18W power limit nearly all the time is spent in MPL 0, which results in very small performance impact. For apsi at 15W, RAPL algorithm is using the first 7 states equally and spents 3x that time in state 7 as Figure 8 shows.

A 10 second snapshot of the DIMM power at 10ms granularity is shown for apsi at 15W RAPL with calibrated weights is shown in Figure 9. The instantaneous power over the last 10ms is shown in light gray, while the running average over the last 100ms is shown in back. The power limit of 15 W, i.e.

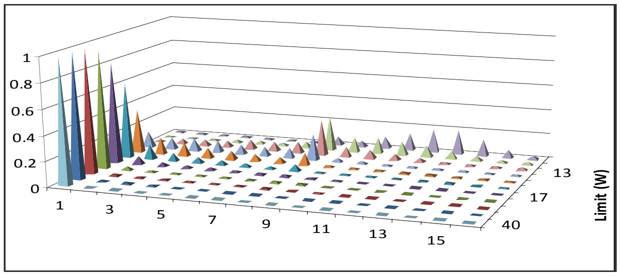

Figure 8: RAPL MPL distribution for SPECCPU2000 apsi.

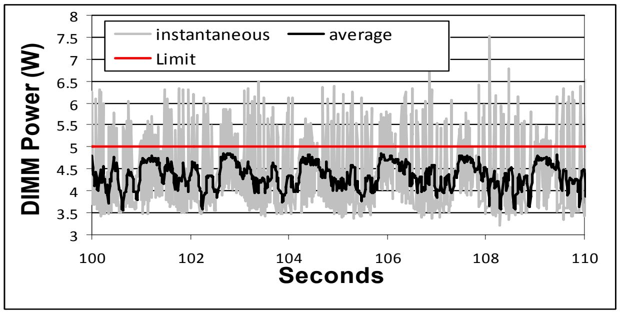

Figure 9: SPECCPU2000 apsi RAPL 15W limit; 10s snapshot.

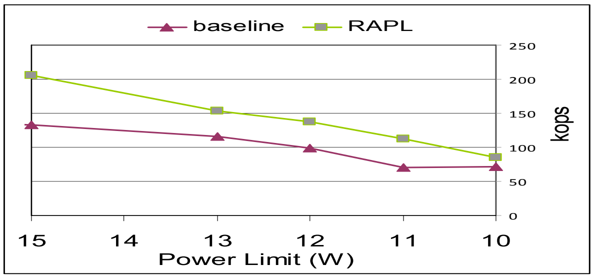

Figure 10: SPECjbb Performance.

5W per DIMM) is given in red. One can observe a periodicity of about 1.7 seconds. Periods of lower instantaneous power are followed by periods of higher instantaneous power, while the average power stays below the limit. This example shows, how the RAPL power budgeting.

The 26 sub-benchmarks of SPECCPU2000 provide a wide range of application and memory characteristics for a good evaluation of the RAPL algorithm. However, they are HPC focused. Figure 10 illustrate results for SPECjbb, which is a server side Java performance benchmark, thus a more representative data center application. An optimized JVM which utilizes large memory pages was selected for this study. To study the effects of memory power limiting, we fixed the transaction throughput target (SPEC kops) to a value which represented 80% of the maximal achievable target on our test system when run without any power limiting. Even in a highly utilized data center it is unlikely to see 100% platform utilization. Figure 10 illustrates the achieved performance as percentage of the performance target for the RAPL and baseline algorithms using calibrated weights. Even at 15W power limit, i.e. 5W per DIMM, the RAPL algorithm results in achieving 100% of target, while the static baseline algorithm reaches only 66% of target. Both algorithms converge for the severe limited case (10W) at around 40% of target performance.

7 Conclusions

This paper presented RAPL, a method for enforcing memory power limits in server platforms. We described a novel methodology for measuring memory power and evaluated RAPL algorithm on the latest generation server platform. Our results demonstrate that RAPL is capable of deterministically enforcing memory power limits across different workloads while minimizing performance impact and gracefully degrading memory bandwidth in the presence of power constraints.

8 References

[1] U.S. EPA, "Report to congress on server and data center energy efficiency,"*Tech. Rep.*, Aug. 2007

[2] J.G. Koomey, "Estimating Total Power Consumption by Servers in the U.S. and the World"

[3] P. Bohrer, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, and R. Rajamony, "The case for power management in web servers,"*Power Aware Computing*, Jan 2002.

[4] Intel, "First the Tick, Now the Tock:Intel Microarchitecture (Nehalem),"2009.

[5] L. Barroso and U. Hĺolzle, "The case for energy-proportional computing,"*IEEE Computer*, Jan 2007.

[6] Xiaobo Fan , Wolf-Dietrich Weber , Luiz Andre Barroso, "Power provisioning for a warehouse-sized computer,"*in Proc. of ISCA*, June 09-13, 2007.

[7] C. Lefurgy, X. Wang, and M. Ware, "Server-level power control,"*in Proc. of the IEEE International Conference on Autonomic Computing*, Jan 2007.

[8] V.Kontorinis, A.Shayan, R. Kumar and D.Tullsen, "Reducing Peak Power with a Table-Driven Adaptive Processor Core,"*International Symposium on Microarchitecture*, 2009

[9] K. Meng, R. Joseph, R. P. Dick, and L. Shang, "Multi-optimization power management for chip multiprocessors,"*In Proc. of PACT*, 2008.

[10] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget,"*In Proc. of the International Symposium on Microarchitecture*, 2006.

[11] W. Felter, K. Rajamani, T. Keller, and C. Rusu, "A Performance-Conserving Approach for Reducing Peak Power Consumption in Server Systems,"*In Proceedings of ICS*, 2005.

[12] B. Diniz, D. Guedes, J. Wagner Meira, and R. Bianchini, "Limiting the power consumption of main memory,"*In Proc. of the 34th Annual International Symposium on Computer Architecture*, pages 290-301, 2007.

[13] J. Lin, H. Zheng, Z. Zhu, E. Gorbatov, H. David, and Z. Zhang, "Software thermal management of DRAM memory for multicore systems,"*In Proc. of the 2008 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 337-348, 2008.

[14] W. Bircher and L. John, "Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events,"*in Proc. of 2007 IEEE Int'l Symp. on Perf. Analysis of Systems and Software*, April 2007, pp. 158-168.

[15] C. Lefurgy, K. Rajamani, F. L. Rawson III, W. Felter, M. Kistler, and T. W. Keller, "*Energy management for commercial servers, IEEE Computer*, 36(12):39-48, 2003.

[16] Bianchini R. and Rajamony R, "Power and Energy Management for Server Systems, "*Computer 37*, pp. 68-74, Nov. 2004.

[17] Intel Xeon Processor 5500 Series EMTS

[18] D. Goldberg, *Genetic Algorithm in Search, Optimization and machine learning*, Addison-Wesley Publishing Company, Inc., 1989.

[19] C. Isci and M. Martonosi, "Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data,"*In Proceedings of the ACM/IEEE International Symposium on Microarchitecture (MICRO-36)*, San Diego, CA, Dec. 2003.

[20] Frank Bellosa, "The Benefits of Event-Driven Energy Accounting in Power-Sensitive Systems,"*ACM SIGOPS European Workshop*, September 2000.

[21] Intel Memory 3-sigma Power Analysis Methodology, http://edc.intel.com/Platforms/Xeon-5500/#hardware

[22] Micron Technology, Inc. TN-41-01: Calculating Memory System Power For DDR3, Aug. 2007.