14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Chicago, IL, 2014.

Enabling Efficient Power Provisioning for Enterprise Applications

Balaji Subramaniam and Wu-chun Feng
Department of Computer Science
Virginia Tech
{balaji, feng}@cs.vt.edu

***Abstract*—The increasing demand for computation and the commensurate rise in the power density of data centers have led to increased costs associated with constructing and operating a data center. Exacerbating such costs, data centers are often over-provisioned to avoid costly outages associated with the potential overloading of electrical circuitry. However, such over-provisioning is often unnecessary since a data center rarely operates at its maximum capacity. It is imperative that we maximize the use of the available power budget in order to enhance the efficiency of data centers. On the other hand, introducing power constraints to improve the efficiency of a data center can cause unacceptable violation of performance agreements (i.e., throughput and response time constraints).**

As such, we present a thorough empirical study of performance under power constraints as well as a runtime system to set appropriate power constraints for meeting strict performance targets. In this paper, we design a runtime system based on a load prediction model and an optimization framework to set the appropriate power constraints to meet specific performance targets. We then present the effects of our runtime system on energy proportionality, average power, performance, and instantaneous power consumption of enterprise applications. Our results shed light on mechanisms to tune the power provisioned for a server under strict performance targets and opportunities to improve energy proportionality and instantaneous power consumption via power limiting.

I. Introduction

The number of large-scale data centers continues to grow rapidly to accommodate the ever-increasing resource demand of applications, e.g., [1], [16]. This, in turn, has exposed power consumption as a first-order design constraint with a commensurate increase in the cost of building infrastructure capable of powering such massive data centers and the recurring energy costs to keep such data centers operational. Such data centers are typically over-provisioned to avoid unexpected outages associated with the potential overloading of electrical circuitry [14], [18]. But data centers rarely operate at their peak power capacity, making over-provisioning unnecessary.

Figure 1 shows the cumulative distribution function (CDF) of the instantaneous power consumption for the processor package and memory subsystems running an enterprise transaction processing (SPECpower_ssj2008 benchmark[1] [4]) application at three different load-levels on a dual-socket server. The workload's power consumption ranges between 25% and 55% of the peak power limit. Such characteristics provides us opportunity to support more resources under the same power budget if we can cap the power consumption.

[1]Hereafter, referred as SPECpower.

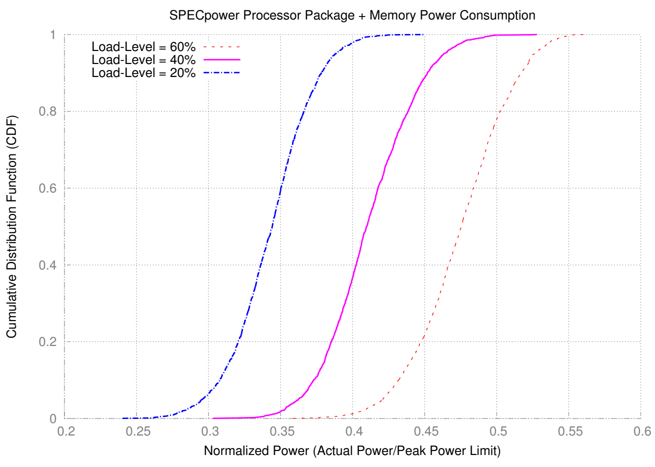

Fig. 1. Illustration of the Opportunity to Limit Power Consumption.

In addition, these data centers are under-utilized due to the inherent nature of the workloads [8], [14] and the need to provide isolation to mitigate serious violation of performance agreements (i.e., throughput and response time constraints) [18]. Unfortunately, power efficiency is traditionally measured only when a server is maximally exercised, and effective techniques to improve non-peak power efficiency is an active research area.

Figure 2 shows the power consumption of a compute server running SPECpower under different load-levels and the hypothetical linear and ideal (i.e., *energy-proportional*) non-peak power curves. As evident from the figure, there is room to improve the non-peak power efficiency of the server with respect to both the ideal as well as linear power curves.

Power-capping mechanisms, such as Intel's Running Average Power Limit (RAPL) [2], [9], [22], [26], can be used to set power limits on subsystems (e.g., processor package and memory) and support more resources under the same power budget. However, introducing arbitrary power limits can cause serious violations of performance and the interactions between subsystem-level power limits is not yet well understood. What we desire is a thorough empirical study of mechanisms to improve non-peak power efficiency, along with its effects on the performance (both throughput and response time) of an

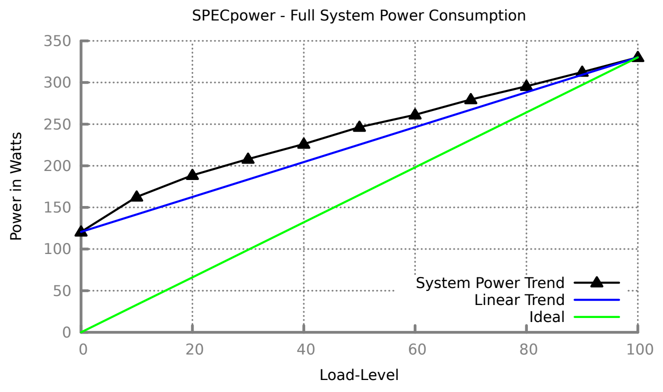

Fig. 2. Illustration of Non-Peak Power Efficiency.

application, and a methodology to place appropriate power caps on the system while meeting strict performance targets.

Towards enabling efficient power provisioning, we propose a methodology to efficiently cap the power consumption of applications by improving the non-peak power efficiency while meeting performance targets. Specifically, our contributions include the following:[2]

- Accurate models capturing the performance of an application under power limit, while taking into account the interaction between subsystem-level power limits.
- Prediction of current load on the system through statistical models and a multi-dimensional optimization framework to determine power limits on subsystems to maintain throughput while meeting response time constraints.
- A runtime system that leverages the load prediction model and the optimization framework to place appropriate power limits on subsystems.
- An analysis of the effects of a power-constraining runtime system on the energy proportionality, power, and instantaneous power consumption of enterprise applications.

Leveraging the above contributions and using an enterprise transaction-processing application and a web-serving application as representative workloads, we draw *six key inferences*. First, simple, but well defined and studied, basis functions can be used to accurately model the non-linear relationship between performance and subsystem-level power limits. Second, subsystem-level power caps can be tuned to meet strict performance targets. Third, the opportunity to achieve ideal non-peak power efficiency reduces as system utilization decreases. Fourth, power limiting of the memory subsystem does not contribute much to the improvement of non-peak power efficiency. Fifth, mixed workload deployment allows for improvement in energy proportionality at low utilization-levels relative to single workload deployment. Finally, power limiting narrows the range of instantaneous power consumption, allowing us to support more resources under same power budget.

The rest of the paper is structured as follows. We start in Section II by giving an overview of the workloads and the RAPL interface used for power management. Section III describes our methodology to model the performance under subsystem-level power limits. In Section IV, we present our runtime system to minimize power while meeting strict performance targets. The runtime system leverages a load prediction model and an optimization framework. We describe our empirical results on the effects of subsystem-level power limits on energy proportionality, power efficiency, and instantaneous power consumption in Section V. Related research is described in Section VI, and Section VII concludes the paper.

II. Background

We provide an overview of the transaction processing application (SPECpower benchmark), the web serving application (SPECweb2009 benchmark[3] [6]) and RAPL interfaces. Specifically we discuss the details of the benchmark and the features of the RAPL interfaces.

A. Overview of SPECpower Benchmark

SPECpower is an industry-standard benchmark that measures both the power and performance of a server node. The workload captures a server-side Java transaction processing application, which is based on the SPECjbb2005 benchmark. The system setup consists of two machines: the system under test (SUT) and the control and collection system (CCS).

The SPECpower benchmark is a graduated workload, i.e., it runs the workload at different *load-levels* and reports the power and performance at each load-level [5]. The benchmark uses a calibration phase to determine the maximum throughput possible and it is set as the throughput target for 100% load-level. The target for the rest of the load-levels is calculated as a percentage of the throughput target for 100% load-level. For example, if the throughput target for 100% load-level is 100,000, then the target for 70% load-level is 70,000, 60% is 60,000, and so on. The throughput is measured in server-side Java operations per second (ssj_ops).

B. Overview of SPECweb Benchmark

SPECweb is an industry standard benchmark for measuring front-end web server performance. It allows the user to measure performance based on the request handling capability and response time maintained by a server node. The benchmark consists of four different components: clients, web server (system under test – SUT), back-end simulator (BeSim), and prime client.

The main performance and power metric for the benchmark is simultaneous user sessions (SUS) and SUS/watt respectively. In addition to SUS, the SPECweb benchmarks adds two different response time performance metrics, namely *TIME_GOOD* and *TIME_TOLERABLE*. By default, 95% and 99% of the requests should have response time less that *TIME_GOOD* and *TIME_TOLERABLE* respectively. Because response time is important for web serving applications, the constraints are based on the 95th or 99th percentile response

[2]This is a follow-up paper to our previous work described in [26]

[3]Hereafter, referred to as SPECweb.

time (instead of the average). Meeting such strict response time constraints is challenging from the perspective of power management. Similar to SPECpower, we can control the benchmark parameters to execute the benchmark at different *load-levels* i.e., different SUS. This benchmark also allows us tweak a set of input parameters. We refer the reader to [25] for a full list of configurable parameters.

C. Intel's Running Average Power Limit (RAPL) Interfaces

RAPL debuted in Intel Sandy Bridge processors. The RAPL interfaces provide mechanisms to enforce power consumption limits on a specific subsystem. The RAPL interfaces can be programmed using the model-specific registers (MSRs). MSRs are used for performance monitoring and controlling hardware functions. These registers can be accessed using two instructions: (1) *rdmsr*, short for *read model-specific registers* and (2) *wrmsr*, short for write model-specific registers.

RAPL interfaces operate at the granularity of a processor socket. The server platforms provide control over three domains (i.e., subsystems): (1) package (PKG), (2) power plane 0 (PP0),[4] and (3) DRAM. Each domain consists of its own set of RAPL MSR interfaces. On a server platform, RAPL exposes four capabilities: power limiting,[5] energy metering,[6] performance status, and power information. We refer the reader to the Intel software developer's manual [2] and existing literature [22], [26] for more information on RAPL interfaces.

III. Modeling Performance Under Power Limit

It is unclear how subsystem-level power limits will affect the throughput and the response time of the benchmarks, particularly satisfying the constraints based on 99th-percentile response times. Therefore, in this section, we model the relationship between performance (i.e., throughput and response time) and subsystem-level power limits to facilitate the design of a runtime system. In case of the SPECweb benchmark, we take the response time constraints into account (i.e., *TIME_GOOD* and *TIME_TOLERABLE*) in addition to the throughput.

First, we provide details of our experiment setup and the configurations used for each workload. Next, we provide an overview of non-linear models and their mathematical forms. Finally, we discuss our experiences in modeling the relationship between performance and subsystem-level power limits. We then use these models to design a runtime system using a load prediction model and an optimization framework in Section IV and evaluate our framework with different performance requirements on a real system in Section V.

A. Experimental Setup

The SUT is a dual-socket and dual-memory node setup with two Intel Xeon E5-2665 processors for a total of 32 cores when hyperthreading is ON. It has 256 GB of memory. Watts Up power meter is used for full system power measurements. In all our experiments, we use the least possible value as the sliding (time) window for power limiting (i.e., 976 microseconds).

1) Setup for SPECpower: The CCS has an Intel Xeon E5405 processor with dual quad cores and 8 GB of RAM. We used all the cores in SUT for our experiments. Eight JVMs with four threads each were used. The four threads in each JVM were pinned to two adjacent physical cores on the SUT. To further enhance the performance of the SUT, we enabled large page memory (HugeTLB) support and set aside 32 GB for huge page allocation. Note that HugeTLB support is enabled only for SPECpower. In order to provide consistent performance results in all our experiments, we fix the *input.load_level.target_max_throughput* parameter to achieve the same performance. It was set to 140,000 ssj_ops for each JVM for a total of 1,120,000 ssj_ops for the entire run. In all our experiments, 100% load-level corresponds to 1,120,000 ssj_ops.[7] We changed the runtime to 120 seconds using the *input.load_level.length_seconds* parameter.

2) Setup for SPECweb: We used 26 clients, 1 prime client and 2 Besim for our experiments. The prime client is an Intel Xeon E5405 processor with two quad cores and 8 GB of RAM. The Besims had two dual core AMD Opteron 2218 processors with 4 GB of RAM. In this paper, we benchmark only the SPECweb_PHP_Ecommerce workload. We used a Apache installation with php module as our web serving application. We setup a bonded Ethernet link with the available ports on the SUT to enable data transmission upto 2 Gbps. Note that the bonded Ethernet link is only setup for SPECweb. Because the network bandwidth is saturated at 13000 SUS, 100% load-level corresponds to 13000 SUS in all our experiments with SPECweb. In addition to the load-level, we also maintain the response time constraints. By default, 95% (*TIME_GOOD* parameter) and 99% (*TIME_TOLERABLE* parameter) of the requests need to have response times less than 3 and 5 seconds, respectively. These response time constraints are used in the compliant runs. The load-level is changed by manually modifying the SIMULTANEOUS_SESSIONS parameter. We modified the RUN_SECONDS input parameter to 420 seconds. Since we focus only on the processor package and memory power management, we load all the data associated with the Ecommerce workload into RAMFS to keep the data set in memory and minimize the involvement of disks.

B. Overview of Non-linear Models

Non-linear models are widely used in a variety of scientific fields. In our case, we use non-linear models to capture the relationships between the throughput, response time, and subsystem-level power limits for SPECpower and SPECweb benchmarks.

Through our experimental data, we determined that two non-linear forms (Gompertz's and Power-Law model) are sufficient to capture the relationships required to design an

[4]PP0 subsystem includes components such as ALUs, FPUs, L1 and L2 caches [3].

[5]RAPL maintains an average power limit over a sliding window instead of enforcing strict limits on the instantaneous power.

[6]The DRAM RAPL domain's energy consumption includes the DRAM modules' energy consumption.

[7]This value was determined by averaging 10 calibration runs.

optimization framework.[8] Table I presents mathematical forms of these models. We chose the most basic form of these models for simplicity. In Sections III-C, we use these basic forms to capture the non-linear relationship between throughput, response times and power limits.

TABLE I
MODELS AND THEIR MATHEMATICAL FORMS

Model Name	Function
Gompertz Model	$\alpha(e^{\beta e^{\gamma x}})$
Power-Law Model	$(\alpha x^{\beta}) + \gamma$

C. Performance Under Power Limit

To model performance under a power limit, we use a non-linear regression approach in which a non-linear mathematical model is used to describe the relationship between the response variable and the predictor variables. In general, modeling involves the collection of a data set, followed by the creation of the model.

TABLE II
DATA SET USED

Benchmark	Power Limit Ratios $\{PP0_1, PP0_2...\}X\{DRAM_1, DRAM_2...\}$
SPECpower	{1.00, 0.90, 0.80, 0.70, 0.60, 0.50, 0.40, 0.30, 0.20, 0.10} X {1.00, 0.95, 0.90, 0.85, 0.81, 0.76, 0.71, 0.67, 0.62, 0.57}
SPECweb	[{1.00, 0.90, 0.81, 0.71, 0.62} X {1.00, 0.98}], [{0.87, 0.78, 0.68, 0.59, 0.50} X {.97, .96}], [{0.78, 0.68, 0.59, 0.50, 0.40} X {0.96, 0.95}], [{0.59, 0.53, 0.46, 0.40, 0.34} X {0.95, 0.94}], [{0.39, 0.35, 0.32, 0.29, 0.26} X {0.94,0.93}]

A description of the data set is given in Table II. Note that the data is presented as Cartesian products (i.e., AXB is set of all pairs of (a, b) where $a \in A$ and $b \in B$). In our experiments, the data set for SPECpower uses 10 different power limits for both PP0 and DRAM subsystems. For SPECweb, we use 25 PP0 power limits and 10 DRAM power limits. We do not include the results of the power limiting of the package subsystem in this paper because our experiments reveal that all the power savings comes from the PP0 subsystem when we limit the power consumption of the package subsystem [26]. The relationship between performance and power limit ratio is modeled instead of using the actual power limit, as ratios act as a good system independent metric. These ratios are calculated as follows:

$$\text{Power Limit Ratio} = \frac{\text{Power Limit on the Subsystem}}{\text{Highest Power Limit Used}}$$

For SPECpower, we collect the data at 100% load-level using the *load_level.target_max_throughput* parameter and running it through every possible combination of power limits for a total of 100 data points. In case of SPECweb, we use five different combinations of data sets as shown in Table II. Each of the combination corresponds to running SPECweb at load-levels between 100% and 20% in steps of 20. We make sure that the response time metrics are satisfied while collecting the data since SPECweb is a latency-sensitive application. The non-linear models are built using R [7]. Specifically, we use the nls2() package to perform non-linear regression.

[8]We investigated creating models using several non-linear forms such as exponential, power law, logistic, monomolecular and Gompertz model. However, we only discuss the models that best fit our data in this paper.

Root mean square error (RMSE) is used to determine the quality of fit of our models. By definition, residual is the difference between the observed value and predicted value, as shown in Equation (1), where Y_i is the observed value, $f()$ is the model, and $f(X_i)$ represents the predicted value as estimated by the model. The RMSE for a model is calculated as shown in Equation (2), where N is the number of predictions.

$$Residual = (Y_i - f(X_i)) \tag{1}$$

$$RMSE = \frac{\sqrt{\sum Residual^2}}{N} \tag{2}$$

We are interested in modeling the upper bounds as these upper boundary conditions will later be used to determine subsystem-level power limits to minimize the power consumption given performance constraints using an optimization framework. We also model the interaction term for understanding the effects of simultaneously changing the power limits on two different subsystems. Figures 3 and 4 show our non-linear models for subsystem-level power limiting of SPECpower and SPECweb, respectively. The figures also show how we use the data set in Table II to identify the upper bounds. The curves f1(x), f2(x), and f3(x) represent our model for the boundaries. For SPECpower, throughput is indirectly modeled using load-levels achieved (i.e., as a percentage of maximum throughput possible). The throughput of SPECweb is indirectly modeled as a percentage of maximum SUS possible. The power constraints placed on subsystems limit the maximum possible operations processed by the SPECpower benchmark, which is clearly evident in Figure 3. Similar behavior can be seen for SPECweb. There are three parts in the model: PP0 power limit ratio – f1(), DRAM power limit ratio – f2(), and the interaction term – f3(), which is the product of both those ratios. Each of the terms is modeled separately to keep the basic forms of the models used for non-linear regression simple. Table III summarizes our models, their parameters, and quality of fit.

IV. RUNTIME SYSTEM

The runtime consists of three parts: load-level detection, the optimization framework and the algorithm for the runtime system. We describe each of these components in rest of the section.

A. Load-Level Detection

The runtime system should detect the load-level of the application in order to set appropriate power limits. To facilitate this process, we used last-level cache misses (LLCM) per

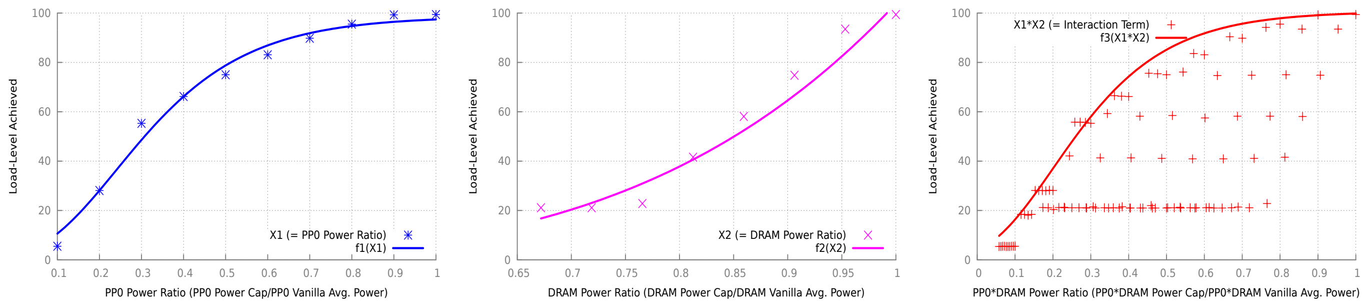

Fig. 3. Models for SPECpower – Left: PP0 Power Limit Bounds, Center: DRAM Power Limit Bounds, and Right: Interaction Term Bounds

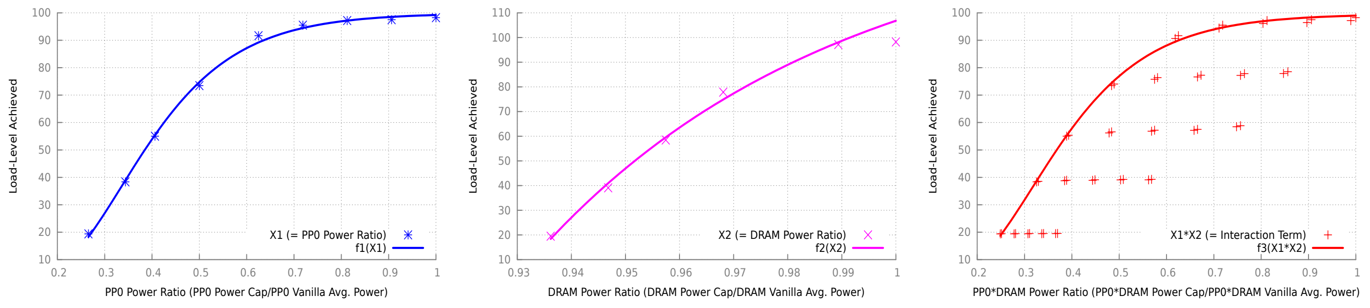

Fig. 4. Models for SPECweb – Left: PP0 Power Limit Bounds, Center: DRAM Power Limit Bounds, and Right: Interaction Term Bounds

TABLE III
MODELS AND THEIR PARAMETERS FOR PERFORMANCE MODELING UNDER POWER LIMIT. SEE TABLE I FOR BASIC FORMS OF THE MODELS. RMSE = ROOT MEAN SQUARED ERROR.

Predictor	**Model Name**	α	β	γ	**RMSE**
		SPECpower			
PP0 power limit ratio–$f1(x)$	Gompertz	98.66	-3.95	-5.95	1.08
DRAM power limit ratio–$f2(x)$	Power Law	105.11	4.42	-1.31	1.41
Interaction term –$f3(x)$	Gompertz	100.67	-3.29	-5.95	1.75
		SPECweb			
PP0 power limit ratio –$f1(x)$	Gompertz	99.87	-12.58	-7.53	0.39
DRAM power limit ratio–$f2(x)$	Power Law	-45.80	-16.30	152.6	1.05
Interaction term –$f3(x)$	Gompertz	99.59	-10.62	-7.43	0.28

TABLE IV
MODELS FOR LOAD-LEVEL DETECTION

Benchmark	**Model Name**	α	β	γ
SPECpower	Linear	103.69	7.63	-
SPECweb	Power Law	101.42	2.51	5.69

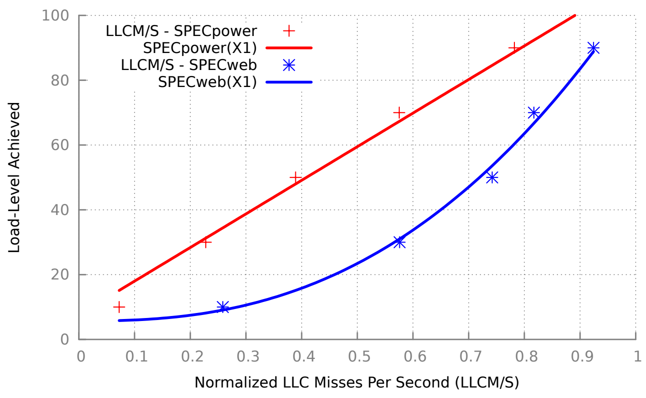

Fig. 5. Models for Load-Level Detection

second (LLCM/S) as an indicator to determine the load-level. LLCM is a good indicator of performance for a variety of applications as shown in [13], [19]. The relationship between LLCM/S and the load-level is modeled similar to the models presented in Section III. Figure 5 shows our models. We use linear regression[9] for modeling the relationship in case of SPECpower and non-linear regression (Power-Law model) in case of SPECweb. The points in the figure are the training data set and the lines and curves are the models. The regression parameters for our models are given in Table IV.

[9]The linear regression is of the form: Load-Level = α LLCM/S + β.

B. The Optimization Framework

The models proposed in Section III are used in our optimization framework. We define and solve these non-linear optimization problems to minimize power for a given performance Y (i.e., throughput for SPECpower and throughput and response time constraints for SPECweb). The variables in our optimization framework are PP0 power ratio (X1), DRAM power ratio (X2), and the interaction term (X1*X2).

Equation (3) presents the optimization problem. In general, for subsystem-level power limiting, we find the smallest power ratio (i.e., $X1$ and $X2$) given a performance constraint (i.e., constraint on Y1). As already discussed, we model only the boundaries. As a consequence, we have inequalities in Equation (3). $f1(X1)$, $f2(X2)$, and $f3(X1*X2)$ from Figures 3 and 4 are used as upper-bound constraints on performance. The set $\{(PP0R_{LB}, PP0R_{UB}), (DRAMR_{LB}, DRAMR_{UB})\}$ are the upper bound and lower bound for the PP0 and DRAM power limit ratios. The set for SPECpower and SPECweb are $\{(1.00, 0.10), (1.00, 0.57)\}$ and $\{(1.00, 0.26), (1.00, 0.93)\}$ respectively. In all our experiments, we solve these non-linear optimization problems using the MINOS solver.

$$\begin{aligned}
&Minimize : X1 + X2\\
&Subject\,to :\\
&\qquad\qquad f1(X1) \geq Y\\
&\qquad\qquad f2(X2) \geq Y\\
&\qquad\qquad f3(X1*X2) \geq Y\\
&\qquad Var : PP0R_{LB} \leq X1 \leq PP0R_{UB}\\
&\qquad Var : DRAMR_{LB} \leq X2 \leq DRAMR_{UB}
\end{aligned} \tag{3}$$

C. Runtime Algorithms

We propose runtime power management schemes which leverage the load-level detection model and the optimization framework described in Section IV-A and IV-B, respectively. Our runtime systems can be used in two cases. In the first case, a single application is run on the entire system and in the second case two different applications that needs to run at load-levels less than 50% are executed. In both cases, appropriate subsystem-level power limits required are identified and set on the system. We propose two different algorithms addressing each case: algorithm without workload consolidation and algorithm with workload consolidation. The algorithm without workload consolidation, used for the first case, is described in Figure 6. The power management scheme monitors the LLC misses for first N seconds and places appropriate power limits on subsystem based on offline models.

```
1 Input: Workload --> W1.
2 Launch workload W1.
3 Calculate LLCM/S over first N seconds.
4 Determine the load-level using the appropriate load-
  level detection model shown in Section IV-A.
5 Set appropriate PP0 and DRAM power limits on all sockets
   using the optimization framework shown in Section IV-B.
```

Fig. 6. Algorithm Without Workload Consolidation

The algorithm with workload consolidation, used for second case, is described in Figure 7. In the second scheme, we consolidate the workload on to a single socket and memory node and launch another workload in the other socket. We also place appropriate subsystem-level power limits on the socket depending on the workload. This scheme was developed as previous work has shown that deploying mixed workloads can smooth the instantaneous power profile of the system and such deployment can reduce the difference between average and peak power [14]. As we will show in Section V, deploying mixed workloads can help improve the energy proportionality of the system.

```
1  Input: Workloads --> W1 and W2
2  Lines 2 through 4 in Figure 6.
3  if load-level < 50%
4  then
5     Consolidate the workload W1 on to a single socket and
       memory node.
6     Launch workload W2 and confine it to the second
      socket and memory node.
7     for each socket
8     do
9        Calculate LLCM/S over first N seconds.
10       Determine the load-level using the appropriate
         load-level detection models shown in Section IV-A.
11       Set appropriate PP0 and DRAM power limits on the
         socket using the optimization framework shown in
         Section IV-B.
12    done
13 endif
```

Fig. 7. Algorithm With Workload Consolidation

V. Evaluation

Our runtime system which leverages the models and optimization framework allows us to run a workload at the optimal power limit possible. The runtime system is used to determine the power limit for each subsystem for a particular load-level, the subsystem-level power limit is set on our evaluation system and results are reported. We show results for three workload cases: SPECpower without consolidation (SP) (using algorithm shown in Figure 6), SPECweb without consolidation (SW) (using algorithm shown in Figure 6) and SPECpower+web with consolidation (SPW[10]) (using algorithm shown in Figure 7). In this section, we look at the impact of our runtime system on energy proportionality, power savings and instantaneous power consumption in each workload case. We use the same experiment setup described in Section III-A.

A. Energy Proportionality

We are interested in analyzing the energy proportionality of the system. The deviation of the power curve of the system from the ideal power curve is of particular interest to us. To illustrate with an example, Figure 2 shows the average power consumed by the testbed running SPECpower benchmark at each load-level, the hypothetical linear power trend and the hypothetical ideal energy proportional power trend for the system. We would like the area between the system and the ideal power trend to be as small as possible. Henceforth, this

[10]In the SPW case: (1) SPECweb is launched first, followed by SPECpower, (2) The runtime for SPECpower is changed to 420 seconds, (3) X% load-level of SPECpower + X% load-level of SPECweb is reported as 2∗X% load-level for the system. For example, 10% load-level of SPECpower and SPECweb is reported as 20% load-level for the system and (4) The maximum load-level we could achieve on a single socket and memory node while maintaining performance was 43% for SPECpower and 46% for SPECweb. Therefore, we show results only to a maximum of 80% load-level for SPW (i.e., 40% SPECpower + 40% SPECweb).

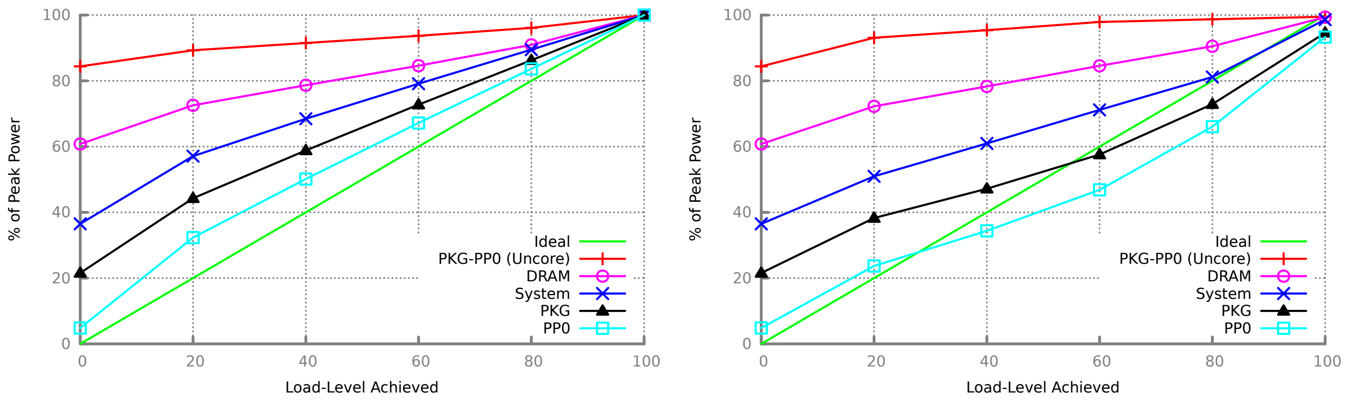

Fig. 8. Energy Proportionality – Left: SP, Right: SP With Runtime System

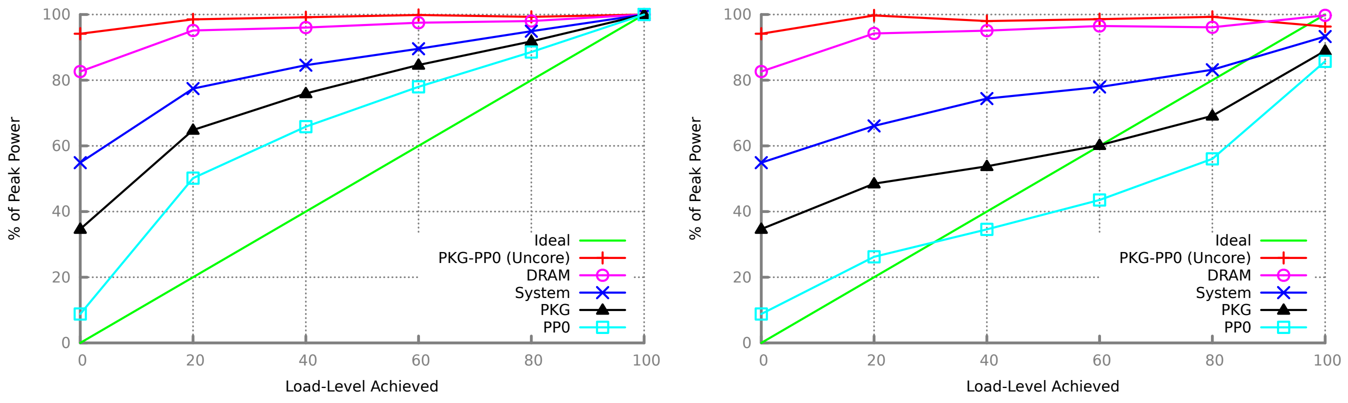

Fig. 9. Energy Proportionality – Left: SW, Right: SW With Runtime System

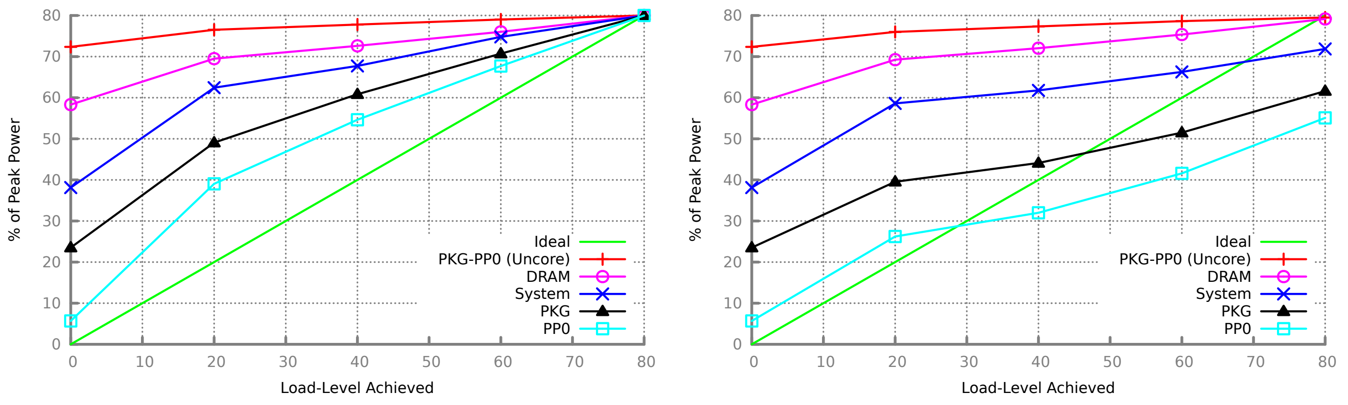

Fig. 10. Energy Proportionality – Left: SPW, Right: SPW With Runtime System

area will be referred to as *energy proportionality gap (EPG)*. We are also interested in the linearity of the system power curve which is the area between the system power trend and linear trend. Henceforth, this area will be referred to as *linear deviation gap (LDG)*.

Figures 8, 9 and 10 show the energy proportionality at different load-levels for the three workload cases with and without the runtime system. The Y-axis represents the percentage of peak power[11] consumed by the system or subsystem and X-axis represents the load-level. The ideal case (green line) consumes 40% of peak power at 40% load-level, 60% of peak power at 60% load-level and so on. We show the energy proportionality for the full system, PKG, DRAM, PP0, and *Uncore* (PKG-PP0) [3]. We emphasize that all the results shown are measurements from running the three workload cases on a *real system*. The target throughput is achieved within a range of 2% and the response time constraints are maintained for SPECweb benchmark in all the cases reported. As observed, we achieve energy proportionality for the full system only at 80% or higher load-levels. We want to stress that the

[11]Peak power is the average power consumed at the 100% load-level of the vanilla run of that workload.

system consumes 120 watts when idling which is 36.51% and 54.88% of peak power for SPECpower and SPECweb, respectively. However, there is energy proportionality improvement even at low load-levels. PP0 subsystem followed by PKG achieves the best energy proportionality improvement. PP0 and PKG achieve better-than-energy-proportional operation for load-levels 40% and 60% or higher respectively. The DRAM subsystem achieves only a negligible improvement in energy proportionality. Overall, we observe that the opportunity to achieve ideal energy-proportional power consumption reduces as the load-levels decrease.

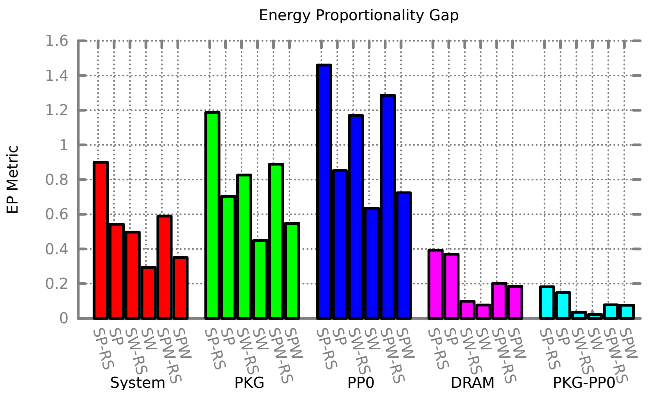

Fig. 11. EP Metric (RS - With Runtime System)

We quantify the EPG using the EP metric [23]. The EP metric is calculated as shown in Equation 4 where $Area_{System}$ and $Area_{Ideal}$ represent the area under the system and ideal power curve respectively. A value of 1 for the metric represents an ideal energy-proportional system. A value of 0 represents a system that consumes a constant amount of power irrespective of the load-level. A value greater than 1 represents a system which is better than energy-proportional.

$$EP = 1 - \frac{Area_{System} - Area_{Ideal}}{Area_{Ideal}} \tag{4}$$

Figure 11 shows the EP metric value for full system and subsystems for the three different workload cases. As evident from the figure, we improve the energy proportionality in all cases. In case of PP0 subsystem, we achieve EP metric > 1 in all the three workload cases. At the full system-level, SP has the best energy proportionality and SW has the worst. SPW achieves energy proportionality that is greater than SW and less than SP. Similar behavior is observed even in the case of subsystems. This makes an interesting case for co-running two different workloads in order to achieve better energy proportionality.

The LDG is quantified using LD metric [27]. The LD metric is calculated using Equation 5. For an linear energy-proportional system, the LD metric will be 0. LD metric > 0 and < 0 indicate superlinear and sublinear energy proportional systems.

$$LD = \frac{Area_{System}}{Area_{Linear}} - 1 \tag{5}$$

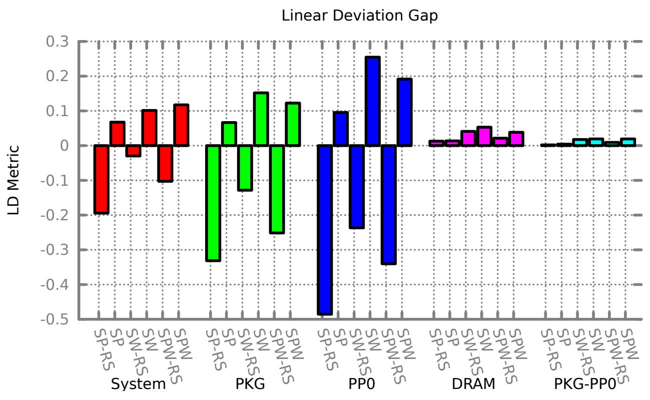

Fig. 12. LD Metric (RS - With Runtime System)

Figure 12 shows the LD metric value for full system and subsystems for the three different cases. At the full system-level, our runtime system changes the LD metric to negative in all workload cases. In case of the subsystems, the LDG for DRAM and *Uncore* see negligible improvement. However, the LD metric for PP0 and PKG subsystem become negative due to our runtime system in all cases.

B. Power Savings

Here we report the power-saving results for the three different workload cases. Figure 13 shows the power savings due to our runtime system for different load-levels for the system and the subsystems. The best power savings came from the SW workload for the full system. We save up to 11%, 15% and 12% for SP, SW and SPW respectively. We observe that the highest power savings is achieved at different load-levels for each workload case.

At the subsystem-level, all of the power savings came from the PP0 subsystem. Moreover, The best power savings across all workloads and load-levels is consistently achieved for the PP0 subsystem. We save upto 32%, 48% and 42% of PP0 power for SP, SW and SPW respectively. The rest of the subsystem (including DRAM) provided no or negligible power savings. However, our runtime system was able to determine the optimal DRAM-level power limit to meet the performance targets. Placing appropriate power limits, even though we get negligible power savings from DRAM subsystem, has advantages with respect to improving the instantaneous power profile of the server.

C. Instantaneous Power Profile

Power provisioning is directly dependent on the instantaneous power consumption of the servers. Figure 14 shows the cumulative distribution functions (CDFs) of instantaneous power profile of the three workload cases. The CDFs present the percentage of time spent at or below a given percentage of the maximum power limit possible for PKG+DRAM subsystems. The maximum power limit possible for our dual-socket server for the PKG+DRAM subsystems is 510 watts [2], [26].

In each workload case, our runtime system is capable of narrowing the range of instantaneous power used by the server.

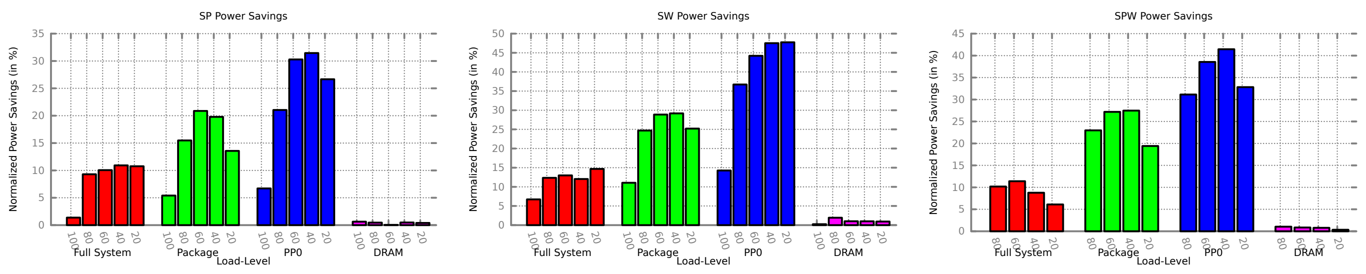

Fig. 13. Power Savings Achieved Due to the Runtime System – Left: SP, Center: SW amd Right: SPW

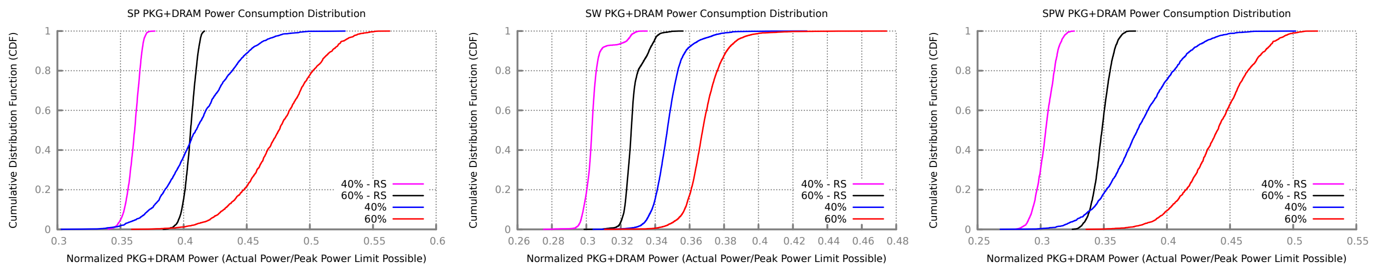

Fig. 14. Instantaneous Power Consumption Distribution – Left: SP, Center: SW amd Right: SPW (RS = With Runtime System)

Such narrow power range provides two advantages: (1) by removing the small number of instances in which the vanilla run's CDF intercepts the 100% line, we will able to support more resources under the same power budget and (2) we reduce the difference between average and the instantaneous peak power drawn. The SPW workload case with the runtime system, similar to energy proportionality, has a distribution which is narrower than SW. Such characteristics provide strong arguments for co-running workloads to improve the energy proportionality of servers at low utilization and help implement better power provisioning strategies.

VI. Related Work

A. Power Provisioning

Pelley et al. [21] proposed *power routing* which assigns servers to power distribution units (PDUs) using a scheduling (linear programming) algorithm. The organize the PDUs into a shuffled topology instead of the traditional primary and secondary PDU organization. Using power routing they dynamically assign servers to PDUs and balance the power drawn across different AC phases. Our work focuses on how local (server-level) power capping can help reduce peak power consumption thereby allowing us to bear more resource under a given power budget.

Fan et al. [14] study the improvements to peak power consumption of a group of servers due to the improvements in non-peak power efficiency using their power model. They provide analytical evidence that shows energy-proportional systems will enable improved power capping at the data-center level. This paper complements Fan et al. by showing empirical evidence on the improvement in instantaneous power consumption of enterprise applications by way of non-peak power efficiency enhancements.

Govidan et al. [15] have developed provisioning techniques based on statistics of application power usage. They identify and exploit statistical properties of power usage for systematic under-provisioning and *over-booking* of power. The work also introduces the notion of short fuses which deals with maintaining power cap at different hierarchies of power distribution in a data center. Our work deals server-level power capping and it effects on performance of latency-sensitive applications. The effects of such power capping at the data center-level is an avenue for future research.

B. Energy Proportionality of Enterprise Applications

Wong et al. [27] provide an infrastructure for improving the energy proportionality using server-level heterogeneity. They combine a high-power compute node with a low-power processor essentially creating two different power-performance operation regions. They save power by redirecting requests to the low-power processor at low request rates thereby improving energy proportionality. While both of our work looks at increasing the energy proportionality by improving linear deviation, this work does not require additional hardware setup and operates on commodity servers.

In our previous work [26], we have studied the effects of RAPL power limiting on the performance, energy proportionality and energy efficiency of the SPECpower benchmark. In this paper, we improve upon our previous work by creating a runtime system to decrease the energy proportionality gap. To design this runtime system, we use a load-detection model and optimization framework which uses statistical models for capturing the performance of an application under power limit. We also discuss the effects of subsystem-level power limiting on the instantaneous power consumption augmenting our previous work.

C. *Subsystem-level Power Management*

Deng et al. [10]–[12] propose the CoScale framework which dynamically adapts the frequency of the CPU and memory respecting a certain application performance degradation target. They also take per-core frequency settings into account. Li et al. [17] study the CPU microarchitectural adaptation and memory low power states to reduce energy consumption of applications bounding the performance loss by using a slack allocation algorithm. Meisner et al. [20] characterize online data-intensive services (OLDI) to identify opportunities for power management, design a framework that predicts the performance of OLDI workloads and investigate the power and performance trade-offs using their simulation framework. Our work provides empirical models for performance (i.e., throughput and response time) of an application under subsystem-level power limit and evaluates our framework on a *real system*. Sarood et al. [24] present an interpolation scheme to optimally allocate power for CPU and memory subsystems in an over-provisioned high-performance computing cluster for scientific workloads. Our work deals with enterprise applications. Moreover, this paper deals with improving energy efficiency of the compute nodes across different levels of utilization (and not just at the peak utilization levels) as data centers running even well-tuned applications spend a significant fraction of their time below peak utilization levels [8], [14], [18].

VII. Conclusion

Improving non-peak power efficiency has the potential to significantly enhance the efficiency of a data center and allows us to host more resources under a given power budget.

In this paper, we use RAPL interfaces to analyze and model the performance (both throughput and response time) of SPECpower and SPECweb benchmarks under subsystem-level power limits. We show that performance under a subsystem-level power limits can be modeled using simple and well-studied non-linear models. We then leverage a load prediction model and an optimization framework to create a runtime system for power management of enterprise application. Our study shows that effective subsystem-level power capping improves the energy proportionality and instantaneous power characteristics even at low utilization-levels, thereby allowing us to support more resources under the same power budget.

Acknowledgement

This work was supported in part by an equipment donation from Supermicro. The authors would like to thank Nataliya Timoshevskaya, Umar Kalim and the anonymous reviewers for their consultation and feedback.

References

[1] Google Details, and Defends, Its Use of Electricity. http://www.nytimes.com/2011/09/09/technology/google-details-and-defends-its-use-of-electricity.html.

[2] Intel 64 and IA-32 Software Developer Manuals - Volume 3. www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html.

[3] Intel Xeon Processor E5-2600 Product Family Uncore Performance Monitoring Guide. http://www.intel.com/content/dam/www/public/us/en/documents/design-guides/xeon-e5-2600-uncore-guide.pdf.

[4] SPECpower Benchmark. http://www.spec.org/power_ssj2008.

[5] SPECpower Benchmark – SSJ Workload Design Document. http://www.spec.org/power/docs/SPECpower_ssj2008-Design_ssj.pdf.

[6] SPECweb Benchmark. http://http://www.spec.org/web2009.

[7] The R Project for Statistical Computing. http://www.r-project.org.

[8] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 40(12), 2007.

[9] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le. RAPL: Memory Power Estimation And Capping. In *International Symposium on Low Power Electronics and Design*, ISLPED, 2010.

[10] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini. CoScale: Coordinating CPU and Memory System DVFS in Server Systems. In *International Symposium on Microarchitecture*, MICRO, 2012.

[11] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini. MultiScale: Memory System DVFS with Multiple Memory Controllers. In *International Symposium on Low Power Electronics and Design*, ISLPED, 2012.

[12] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini. MemScale: Active Low-Power Modes for Main Memory. 2011.

[13] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan. Full-System Power Analysis and Modeling for Server Environments. In *Proceedings of the Workshop on Modeling Benchmarking and Simulation*, MOBS, 2006.

[14] X. Fan, W.-D. Weber, and L. A. Barroso. Power Provisioning For a Warehouse-Sized Computer. In *International Symposium on Computer Architecture*, ISCA, 2007.

[15] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini. Statistical Profiling-Based Techniques for Effective Power Provisioning in Data Centers. In *European conference on Computer systems*, EuroSys, 2009.

[16] J. Koomey. Growth In Data Center Electricity Use 2005 To 2010. http://www.mediafire.com/file/zzqna34282frr2f/koomeydatacenterelectuse2011finalversion.pdf.

[17] X. Li, R. Gupta, S. V. Adve, and Y. Zhou. Cross-Component Energy Management: Joint Adaptation of Processor and Memory. *ACM Transactions on Architecture and Code Optimization*, 2007.

[18] J. Mars, L. Tang, R. Hundt, K. Skadron, and M. L. Soffa. Bubble-Up: Increasing Utilization in Modern Warehouse Scale Computers via Sensible Co-Locations. In *International Symposium on Microarchitecture*, MICRO, 2011.

[19] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta. Evaluating the Effectiveness of Model-Based Power Characterization. In *Proceedings of the USENIX Annual Technical Conference*, USENIX ATC, 2011.

[20] D. Meisner, C. M. Sadler, L. A. Barroso, W. Weber, and T. F. Wenisch. Power Management of Online Data-Intensive Services. In *International Symposium on Computer Architecture*, ISCA, 2011.

[21] S. Pelley, D. Meisner, P. Zandevakili, T. F. Wenisch, and J. Underwood. Power Routing: Dynamic Power Provisioning in the Data Center. 2010.

[22] B. Rountree, D. Ahn, B. de Supinski, D. Lowenthal, and M. Schulz. Beyond DVFS: A First Look at Performance Under a Hardware-Enforced Power Bound. In *International Parallel and Distributed Processing Symposium Workshops and PhD Forum*, IPDPSW, 2012.

[23] F. Ryckbosch, S. Polfliet, and L. Eeckhout. Trends in Server Energy Proportionality. *IEEE Computer*, (9):69–72, 2011.

[24] O. Sarood, A. Langer, L. Kale, B. Rountree, and B. Supinski. Optimizing Power Allocation to CPU and Memory Subsystems in Overprovisioned HPC Systems. In *Proceedings of IEEE Cluster*, 2013.

[25] SPEC. SPECweb2009 Benchmark – User Guide, 2009. Available at http://www.spec.org/web2009/docs/usersguide.html.

[26] B. Subramaniam and W. Feng. Towards Energy-Proportional Computing for Enterprise-Class Server Workloads. In *Proceedings of the International Conference on Performance Engineering*, ICPE, 2013.

[27] D. Wong and M. Annavaram. KnightShift: scaling the energy proportionality wall through server-level heterogeneity. In *Proceedings of the International Symposium on Microarchitecture*, MICRO, 2012.