Metrics, Models and Methodologies for Energy-Proportional Computing

Balaji Subramaniam

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Application

January 16, 2015
Blacksburg, Virginia

Keywords:
Energy Proportionality, Resource Provisioning, Power Provisioning, Running Average Power Limit (RAPL), Scale-Out Workloads, Enterprise Workloads, Green Computing

Metrics, Models and Methodologies for Energy-Proportional Computing

Balaji Subramaniam

ABSTRACT

Massive data centers housing thousands of computing nodes have become commonplace in enterprise computing, and the power consumption of such data centers is growing at an unprecedented rate. 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 realize effective strategies to control the power consumption of the server and improve the energy efficiency of data centers. Adding to the problem is the inability of the servers to exhibit *energy proportionality* which diminishes the overall energy efficiency of the data center. Therefore in this dissertation, we investigate whether it is possible to achieve *energy proportionality* at the server- and cluster-level by efficient power and resource provisioning. Towards this end, we provide a thorough analysis of energy proportionality at the server and cluster-level and provide insight into the power saving opportunity and mechanisms to improve energy proportionality.

Specifically, we make the following contributions at the server-level using enterprise-class workloads. We analyze the average power consumption of the full system as well as the subsystems and describe the energy proportionality of these components, characterize the instantaneous power profile of enterprise-class workloads using the *on-chip energy meters*, 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 and then present the effects of our runtime system on energy proportionality, average power, performance and instantaneous power consumption of enterprise applications. We then make the following contributions at the cluster-level. Using data serving, web searching and data caching as our representative workloads, we first analyze the component-level power distribution on a cluster. Second, we characterize how these workloads utilize the cluster. Third, we analyze the potential of power provisioning techniques (i.e., active low-power, turbo and idle low-power modes) to improve the energy proportionality. We then describe the ability of active low-power modes to provide trade-offs in power and latency. Finally, we compare and contrast power provisioning and resource provisioning techniques.

This thesis sheds light on mechanisms to tune the power provisioned for a system under strict performance targets and opportunities to improve energy proportionality and instantaneous power consumption via efficient power and resource provisioning at the server- and cluster-level.

ACKNOWLEDGEMENTS

My life at Virginia Tech has been impacted by a number of people. I'm extremely grateful to each and everyone of them.

First, I would like to thank my advisor, Prof. Wu-chun Feng, for his guidance during my Ph.D. study. I'm grateful to him for introducing me to the rigors of research and the importance of scientific communication. He has provided me invaluable support, encouragement and opportunities throughout my years at Virginia Tech. I would also like to thank my other dissertation committee members: Prof. Peter Athanas, Dr. Mark Gardner, Prof. Micheal Hsiao, Dr. Chung-Hsing Hsu and Prof. Eli Tilevich for their valuable comments, suggestions and feedback.

I'm especially grateful to Eli for his contributions to my personal and professional well-being. He has been a wise mentor and a great friend over these five years I have known him. His advice – always delivered with his unique sense of humor, and immeasurable patience to listen to my queries and answer them – have been instrumental in my development as a researcher.

The SyNeRGy lab has been helpful in improving my work all these years. I would like to thank Ashwin Aji, Umar Kalim, Thomas Scogland, Jeremy Archuleta, Carlo del Mundo, Nabeel Mohamed, Nataliya Timoshevskaya and Vignesh Adhinarayanan for their valuable feedback.

I consider Umar and Ashwin as my academic brothers and this Ph.D. endeavor would not be complete without their friendship. The discussions with Umar have been a great stress-reliever. I have always enjoyed the company of Umar and Mariam, and I'm thankful for the hospitality they extend. The countless hours of discussions and interactions I've had with Ashwin during our "coffee group" and elsewhere have helped my growth as a researcher and generated many new ideas. I have absorbed a great amount of wisdom from Umar and Ashwin just by being alongside them during their Ph.D. study.

I'd not have embarked on this Ph.D. journey without Prof. Venkateswaran Nagarajan. I thank him for guiding me during my undergraduate studies and introducing me to the nuances of research.

This thesis would not be possible without my friends. I would like to thank Nabanita, Asmita, Gaurav, Jaideep, Anushree, Varun, Abhijit, Sudeeksha, Kartik, Saloni, Keyur, Krishna Raj, Shankha, Vyas, Bharath, Hari, Sony, Suruchi, Krishnan, Sapna and many many others for making this experience delightful. They have all been responsible, in one way or the other, to keep my sanity in check. Nabanita has been a great source of support and comfort in the final mile of my Ph.D. journey. I'm forever indebted to Asmita and Gaurav, for their camaraderie and care. My bromance with Jaideep has led to many casual conversations and deep philosophical exchanges, which made my Ph.D. journey more enjoyable. Vyas has been crucial in polishing my thoughts as a human being as well as a researcher and has streamlined me time and again. The arguments and discussions at home, ranging from politics to philosophy to astronomy, have all been vital in completing my all-round experience and development as a human being and a researcher.

Finally, I would like to thank my mom, Rajeswari, my sister, Akhila, and my brother-in-law, Sathish for their unconditional love and support. They sowed the seeds of reasoning and Science in me and provided me with an environment to follow my aspirations. Words cannot express my gratitude to them.

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

CDF Cumulative Distribution Functions. 38–40, 99, 100

DRAM Dynamic Random-Access Memory. 15, 18, 21, 25, 40–43, 51, 65–67, 79, 81, 82, 96

DVFS Dynamic Voltage Frequency Scaling. 94

EP Energy Proportionality, Energy-Proportional. 2, 28–30, 54, 77, 79, 80, 108, 141

EPG Energy Proportionality Gap. 28, 30–32, 35, 79

LD Linear Deviation. 31–33, 37, 56, 80, 81

LDG Linear Deviation Gap. 29, 31, 32, 80, 81

MSR Model-Specific Register. 15–18, 25

PG Proportionality Gap. 30–33, 37

PKG Package. 15, 17, 18, 25, 34, 79, 81

PP0 Power Plane 0. 15, 41–43, 51, 54, 56, 65, 66, 79–82

RAMFS Random-Access Memory File System. 27, 36

RAPL Running Average Power Limit. 4, 10, 15, 16, 18, 20, 28, 32, 33, 35, 37, 41, 83, 94, 105, 109, 111, 112, 124

RMSE Root Mean Squared Error. 65, 66

SLO Service-Level Objective. 88, 111, 114, 133, 137, 138, 141, 142

YCSB Yahoo! Cloud Serving Benchmark. 91, 96, 118

Chapter 1

Introduction

The number of large-scale data centers continues to grow rapidly to accommodate the ever-increasing resource demand of applications, e.g., [7, 35]. 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 [31, 40]. In addition, these data centers are under-utilized due to the inherent nature of the workloads [22, 24, 31] and the need to provide isolation to mitigate serious violation of performance agreements (i.e., throughput and response time constraints) [40]. Figure 1.1 shows the rampant under-utilization via the CPU utilization profile of cluster of over 20000 Google servers [22]. The Y-axis represents the fraction of time spent at a CPU utilization, and the X-axis represents the CPU utilization of the cluster. As observed, the servers spend negligible amount of time at CPU utilizations greater than 60%. The majority of time is spent between 10-50% CPU utilization. However, traditional processors are designed to operate at maximum energy efficiency when the processor is maximally utilized. There is a mismatch between CPU utilization profile of the workloads and the most energy-efficient operation regions of the processors [22]. As a result, there is a desire to achieve Energy Proportionality, Energy-Proportional (EP) (i.e., achieve energy-efficient execution under all levels of utilization). The lack of energy proportionality diminishes the overall energy efficiency of the data center. This thesis describes our investigation into whether we can achieve energy-proportional computing using power and resource provisioning at the server- and cluster-level.

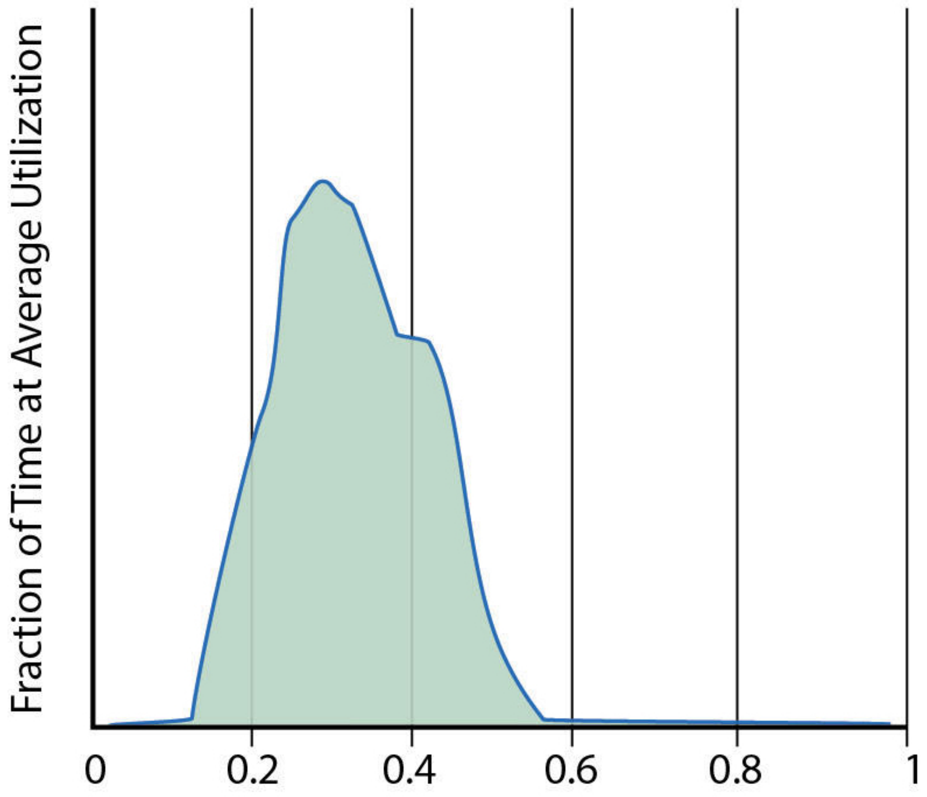


Figure 1.1: Average Utilization of a Google Cluster Between January and March 2013 [22]. The Cluster has Over 20000 Servers. X-axis = CPU utilization.

1.1 Analysis of Server-Level Energy Proportionality

Figure 1.2 shows the power consumption of a compute server running SPECpower under different load-levels and the hypothetical linear and *energy-proportional* (EP) power curves. The Y-axis represents the average power consumed by the system, and the X-axis represents the load-level. The load-level represents the throughput (i.e., performance) of the SPECpower benchmark. The maximum throughput of SPECpower on the server is represented as 100% load-level. The hypothetical EP curve (green line that connects circles) at 40% load-level consumes 40% of the power consumed at 100% load-level; at 60% load-level, it consumes 60% of the power consumed at 100% load-level and so on. Achieving the EP power profile provides same energy efficiency at all load-levels. For example, if SPECpower at 100% load-level achieves a throughput of 1000 units and consumes 100 watts of power, then it has an energy efficiency of 10 throughput-units/watt. If we are able to achieve an energy proportional power profile, 10 throughput-units/watt will be achieved at every load-level. At 30% load-level, we can achieve 300 throughput-units/30 watts = 10 throughput-units/watt; and at 70% load-level, we can achieve 700 throughput-units/70 watts = 10 throughput-units/watt.

As a result, the deviation of the power curve of the system from the energy-proportional power curve is of particular interest to us. We want the area between the system and the EP power curve to be as small as possible. However, as evident from Figure 1.2, there is room to improve the non-peak power efficiency of the server with respect to both the EP as well as linear power curves.

Figure 1.3 shows the cumulative distribution function (CDF) of the instantaneous power con-

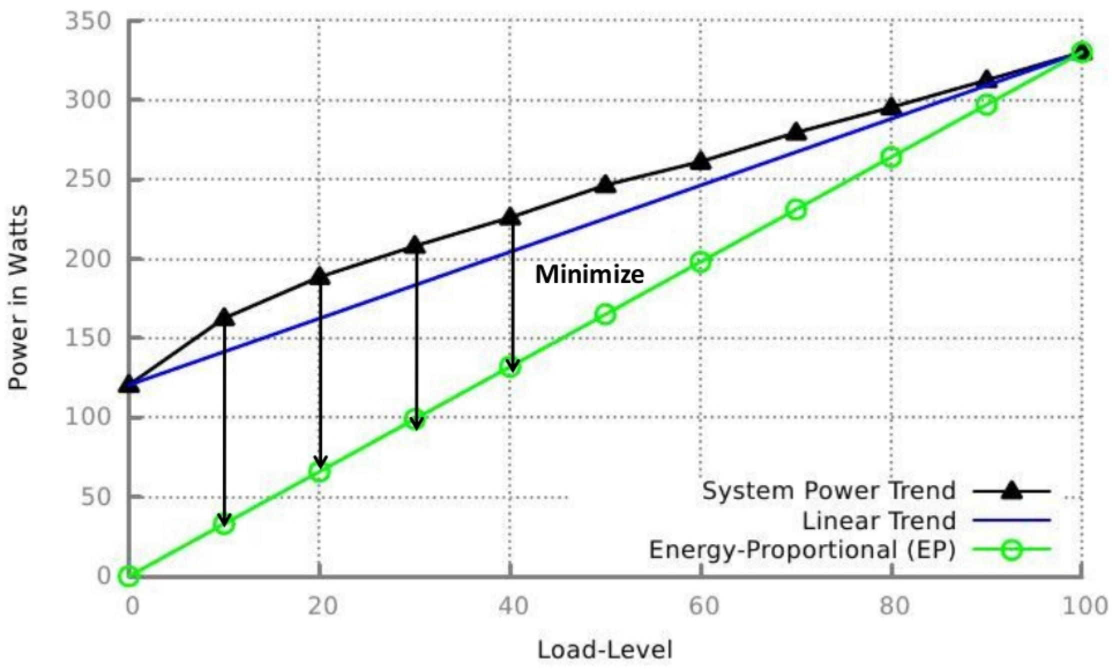


Figure 1.2: Illustration of Non-Peak Power Efficiency.

sumption for the processor package and memory subsystems running an enterprise transaction processing application (SPECpower_ssj2008 benchmark[1] [13]) at three load-levels on a dual-socket server. The plot shows the fraction of time spent at or under a particular percentage of peak power limit. For an illustration of how to interpret the plot, consider SPECpower at 20% load-level. It spends 60% of the time (based on Y-axis) at or below 35% of the peak power limit (based on X-axis). As observed, 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.

Power-capping mechanisms, such as Intel's Running Average Power Limit (RAPL) [8, 27, 46, 51], can be used to set power limits on subsystems (e.g., processor package and memory) and

[1]Hereafter, referred as SPECpower.

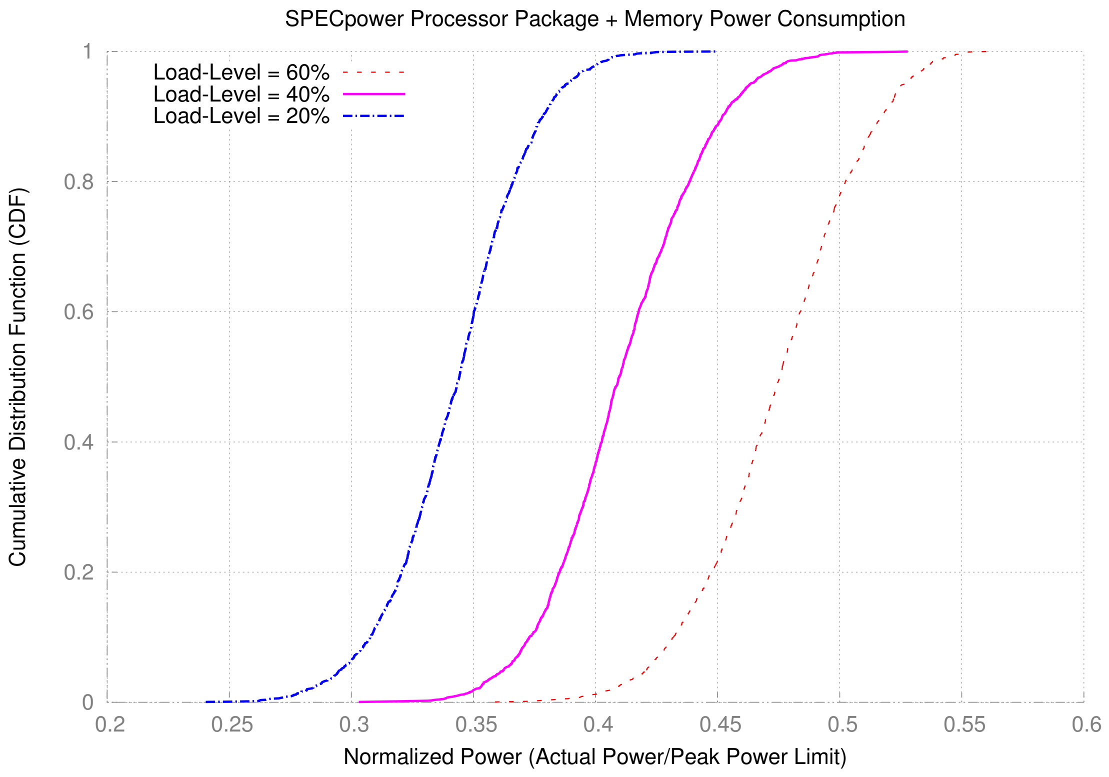


Figure 1.3: Illustration of the Opportunity to Limit Power Consumption.

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

Towards enabling power provisioning at the server-level, we propose a methodology to efficiently cap the power consumption of applications while meeting strict performance targets [51, 52, 55, 56] in Chapter 3.

1.2 Analysis of Cluster-Level Energy Proportionality

The proliferation of cloud computing has led to the emergence of scale-out workloads [4, 32]. These scale-out workloads are a part of several popular Internet services. For example, Netflix uses data-serving mechanisms to access vast amount of media [1], Google uses web search engines to index the public Internet in order to respond to search queries [23] and Facebook uses data-caching mechanisms to access and update very popular shared content [44].

Such scale-out workloads are user-driven. As a result, they are required to meet strict service-level objectives (SLOs), usually in terms of sub-second responsiveness. In order to meet the SLO, these scale-out workloads can span several hundred to thousands of servers to provide efficient access to massive computational resources and huge volumes of data [22]. Such installations consume a significant amount of energy, and in turn, contribute to the operational costs. Moreover, the operational cost is exacerbated by the lack of energy proportionality. Figure 1.4 shows the normalized power consumption of a four-node cluster running scale-out workloads and the hypothetical energy-proportional power curve under different load-levels. It clearly illustrates the lack of energy proportionality in these workloads.

To address these issues, *power provisioning* techniques [28,29,39,51,52] have been shown to improve the energy proportionality. These techniques take advantage of low utilization periods or short idle periods to assign (either active or idle) low-power states to subsystems such as the CPU and memory. Other researchers have used *resource provisioning* techniques [60] to improve the energy proportionality. These techniques use workload consolidation to minimize the number of

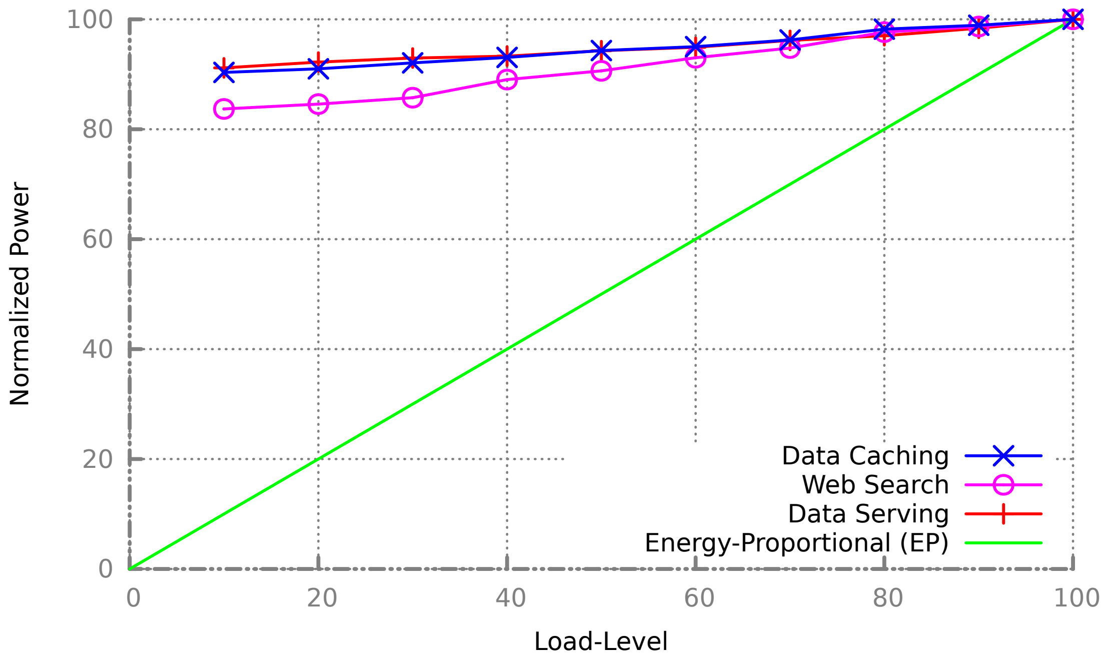


Figure 1.4: Energy Proportionality of Scale-Out Workloads.

servers required to sustain a desired throughput and reduce energy consumption by turning off the servers not in use.

As Internet services such as Netflix, Google and Facebook become even more ubiquitous, scale-out workloads will need increasingly larger cluster installations. Improvements in the energy proportionality of such installations will need to come from dynamic power management systems. Thus, our aim in this thesis is to study the energy proportionality of clusters in the context of scale-out workloads. Specifically, we analyze the effectiveness of different software-controlled (hardware-enforced) power management techniques, such as *power and resource provisioning*, to improve the energy proportionality of scale-out workloads in Chapter 4 [53, 54].

1.3 Thesis Organization

The rest of this thesis is organized as follows. In Chapter 2, we present the details of the related work in this area (Section 2.1). We also provide an overview of workloads, experimental setup, workload configuration, the power measurement interfaces and the power management techniques used in this thesis.

In Chapter 3, we describe our efforts to improve server-level energy proportionality. Section 3.2 describes our analysis and characterization of average and instantaneous power consumption. It presents details on the energy proportionality of full system as well as subsystems in Section 3.2.1 and Section 3.2.2 presents the instantaneous power profile of all subsystems and the observations from these experiments. In Section 3.3, we describe the potential of power limiting to improve energy proportionality. Section 3.4 describes our methodology to model the performance under subsystem-level power limits. In Section 3.5, 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 3.6.

Our efforts to analyze and improve the energy proportionality at the cluster-level is described in Chapter 4. We describe the component-level power distribution of the scale-out workloads in Section 4.2. Section 4.3 presents our characterization of the CPU utilization by the scale-out workloads. We show the effects of power provisioning techniques on energy proportionality, latency and power savings in Section 4.4. In Section 4.5, we give an overview of the power-performance trade-off space exposed by power provisioning techniques. We compare and contrast power and resource provisioning in Section 4.6.

Chapter 2

Background

In this chapter, we present the details of the related work in this area. We also provide an overview of workloads, experimental setup, workload configuration, the power measurement interfaces and the power management techniques used in this thesis.

2.1 Related Work

In this section, we give an overview of the work related to this thesis.

2.1.1 Energy Proportionality and Energy Efficiency

Lo et al. [39] analyze the energy proportionality of web search and in-memory key-value data store on a large-scale cluster. They propose the *Pegasus* runtime system which uses Intel's RAPL interfaces[1] to improve the energy proportionality of these workloads. Meisner et al. [43] 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 power models and simulation framework. Our work complements their papers by providing more analysis with respect to CPU utilization, turbo and active low-power modes, trade-offs in power and latency and the comparison of power and resource provisioning. Furthermore, we provide details of the energy proportionality at the subsystem-level. This thesis also deals with how we can prevent power spikes in servers using RAPL power limiting. Finally, this work provides empirical evidence for the existence of power-performance trade-offs in scale-out workloads on a *real* system.

Wong et al. [59, 60] provide the KnighShift 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. In addition, they compare cluster-level packing techniques (resource provisioning) and server-level low power modes to identify if one of these technique is better with current generation of processors using simulation and power modeling. The results in this thesis are applicable to commodity servers, ones that do not require any additional hardware setup. We study the effects of RAPL power limiting on the performance, energy proportionality, and power efficiency of enterprise workloads at server-level. We also discuss the energy proportionality of

[1]See Section 2.2 for more details on RAPL interfaces.

different subsystems. We then study the effects of power and resource provisioning on the energy proportionality, performance (in terms of response time) and power savings of scale-out workloads. We also provide empirical results for the trade-offs involved in power and resource provisioning on a *real* system.

Tsirogiannis et al. [57] provide a comprehensive study of power consumption of relational databases on a single node. They analyze the energy efficiency of database servers using different hardware and software knobs such as CPU frequency, scheduling policy and inter-query parallelism. They conclude that the most energy-efficient operating point is also the highest performing configuration. Willis et al. [37] study the trade-offs between performance scalability and energy efficiency for relational databases. They identify hardware and software bottlenecks that affect performance scalability and energy efficiency. In addition, they provide guidelines for energy-efficient cluster design in the context of parallel database software. Our research complements theirs by addressing the energy proportionality of non-relational (a.k.a. NoSQL) databases. Moreover, we provide details of subsystem-level power consumption for NoSQL databases, an analysis on the effect of different power provisioning techniques on NoSQL data stores and compare the trade-offs involved in resource and power provisioning.

2.1.2 Power Provisioning

Pelley et al. [45] proposed *power routing*, which assigns servers to power distribution units (PDUs) using a scheduling algorithm that is based on a linear programming formulation. They 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 draw across different AC phases. Govidan et al. [34] 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 caps at different hierarchies of power distribution in a datacenter. In contrast, this thesis focuses on how efficient power provisioning improves the *instantaneous* power consumption, thereby allowing datacenters to bear more load or resources under a given power budget.

Fan et al. [31] 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 datacenter level. This thesis 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. We also provide further insights into the characteristics of scale-out workloads and the effects of power and resource provisioning on their energy proportionality.

2.1.3 Subsystem-Level Power Management

Deng et al. [28, 29] propose the CoScale framework, which dynamically adapts the frequency of the CPU and memory while respecting a certain application performance degradation target. They also take per-core frequency settings into account. Li et al. [38] 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. Sarood et al. [48] present an interpolation scheme to optimally allocate power for CPU and memory subsystems in an over-provisioned high-performance computing cluster for scientific workloads. This thesis 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 [24,31,40].

2.1.4 Power Limiting

Several mechanisms to *cap* the power consumption of the system have been studied [25,33]. However, we study the use of RAPL power limiting in this thesis. David et al. [27] proposed RAPL and evaluated RAPL for the memory sub-system. They present a model that accurately predicts the power consumed by the DIMMs and use RAPL to *cap* the power consumption. Rountree et al. [46] use RAPL power limiting to study the behavior of performance for benchmarks in the NAS parallel benchmark suite. Specifically, they are interested in the performance of various compute nodes under a power bound. Weaver et al. [58] have have exposed RAPL energy meters through PAPI. This thesis uses RAPL power limiting as a tool to provision power in order to achieve energy proportionality.

2.2 Intel’s Running Average Power Limit (RAPL) Interfaces

RAPL was introduced in Intel Sandy Bridge processors. The RAPL interfaces provide mechanisms to enforce power consumption limits on a specific subsystem. The only official documentation available for these interfaces is section 14.7 of the Intel software developer’s manual [8]. Our experiments deal only with the Sandy Bridge server platforms.

The RAPL interfaces can be programmed using the Model-Specific Register (MSR). 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.” The *msr* kernel module can be used for accessing MSRs from *user space* in Linux environments. When loaded, the *msr* module exposes a file interface at */dev/cpu/x/msr*. This file interface can be used to read from or write to any MSR on that CPU.

According to the Intel documentation, RAPL interfaces operate at the granularity of a processor socket. The server platforms provide control over three domains (i.e., subsystems):[2] (1) Package (PKG), (2) Power Plane 0 (PP0), and (3) Dynamic Random-Access Memory (DRAM). PKG, PP0 and DRAM represents the processor package (or socket), the *core* subsystem, and memory DIMMs associated with that socket, respectively. The *core* subsystem is a terminology used by Intel. It represents any component within PKG which is directly involved in computation such as ALUs, FPUs, L1 caches and L2 caches [9]. Intel uses *uncore* to refer to the components indirectly involved in the computation such as memory controller, I/O controller and coherence engine [9].

[2]Note: We use RAPL domain and subsystem interchangeably in rest of the thesis.

Both the *core* and the *uncore* subsystems are a part of the processor package subsystem. An illustration of the subsystems is shown in Figure 2.1. The *MSR_RAPL_POWER_UNIT* register contains the units for specifying time, power, and energy, and the values are architecture-specific. For example, our testbed requires and reports time, power, and energy at increments of 976 microseconds, 0.125 watts, and 15.3 microjoules, respectively. Each domain consists of its own set of RAPL MSR interfaces. On a server platform, RAPL exposes four capabilities:

1. Power limiting – Interface to enforce limits on power consumption.
2. Energy metering – Interface reporting actual energy usage information.
3. Performance status – Interface reporting performance impact due to power limit.
4. Power information – Interface which provides value range for control attributes associated with power limiting.

2.2.1 Power Limiting

RAPL maintains an average power limit over a sliding window instead of enforcing strict limits on the instantaneous power. The advantage of having an average power limit is that if the average performance requirement is within the specified power limits the workload will not incur any performance degradation even if the performance requirement well exceeds the power limit over short bursts of time. The user has to provide a power bound and a time window in which the limit has to be maintained. Each RAPL domain exposes a MSR which is used for programming

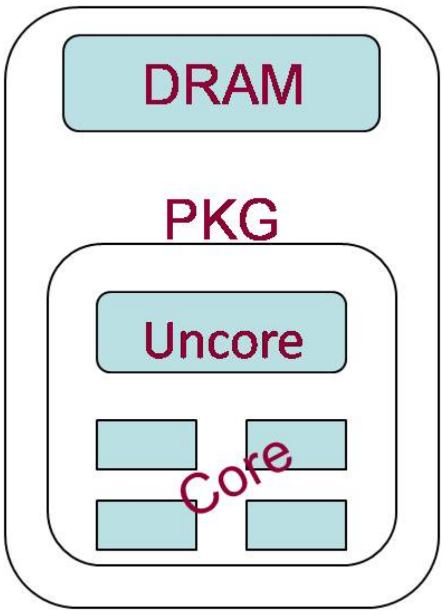


Figure 2.1: Illustration of RAPL Domains

these values. The PKG domain provides two power limits and associated time window for finer control over the workload performance whereas other domains provide only one power limit. The interface provides a *clamping* ability, which when enabled, allows the processor to go below an OS-requested P-state.

2.2.2 Energy Metering

Each domain exposes a MSR interface that reports the energy consumed by that domain. On a server platform, (1) *energy(PKG) = energy consumed by the processor package*, (2) *energy(PP0) = energy consumed by the core subsystem*, and (3) *energy(uncore subsystem) = energy(PKG) − energy(PP0)*.

2.2.3 Performance Status

This MSR interface reports the total time for which each domain was throttled (i.e., functioning below the OS-requested P-state) due to the enforced power limit. This information will be useful in understanding the effects of power limiting on a particular workload.

2.2.4 Power Information

The PKG and DRAM domains expose a MSR interface that provides information on the ranges of values that can be specified for a particular RAPL domain for limiting its power consumption. This includes maximum time window, maximum power, and minimum power.

Chapter 3

Analysis of Server-Level Energy Proportionality

Towards enabling efficient power provisioning, we propose a methodology to efficiently cap the power consumption of servers running enterprise-class workloads while meeting strict performance targets [51, 52, 55, 56] in this chapter. Specifically, our contributions include the following:

- A rigorous quantification of the energy proportionality of each subsystem within a server node via an analysis of power consumption profiles of the different subsystems when running SPECpower and SPECweb at different load-levels.
- An analysis and characterization of the instantaneous-power profiles at different load-levels of SPECpower and SPECweb to understand whether power limiting will enable us to improve the energy efficiency of these benchmarks.

- 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, average power, and instantaneous power consumption of enterprise applications.

Through our contributions, we make the following observations and conclusions on the power management of the SPECpower and SPECweb benchmarks using RAPL interfaces. First, the processor *core*[1] is the most energy-proportional subsystem and the processor *uncore*[2] is the least. Second, better power-management mechanisms are required to achieve energy proportionality at the *uncore* subsystem-level. Third, there is ample opportunity for limiting the power consumption of processor package and memory subsystems. Fourth, 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. Fifth, subsystem-level power caps can be tuned to meet strict performance targets. Sixth, the opportunity to achieve energy-proportional non-peak power efficiency

[1]The *core* subsystem includes components such as the ALUs, FPUs, L1, and L2 caches [9].

[2]The *uncore* subsystem includes components such as the memory controller, integrated I/O, and coherence engine [9].

reduces as system utilization decreases. Seventh, power limiting at the level of the *core* subsystem is the best option for improving energy efficiency and achieving energy proportionality. Whereas, power limiting of the memory subsystem does not contribute much to the improvement of non-peak power efficiency. Eighth, though we were not able to achieve energy proportionality at the full system level, i.e., entire compute node, we show that energy-proportional operation or better is possible at the granularity of subsystems over which we have control via RAPL power limiting (i.e., *core* subsystem, processor package, and DRAM). Ninth, 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.

3.1 Background

In this section, we provide a brief description about the workloads, their configuration and the experimental setup used for analyzing server-level energy proportionality.

3.1.1 Overview of SPECpower Benchmark

SPECpower [13] is an industry-standard benchmark that measures both the power and performance of a server node. The benchmark mimics a server-side Java transaction processing application and is based on the SPECjbb2005 benchmark [12]. It stresses the CPU, caches, and memory hierarchy and tests the implementations of the Java virtual machine (JVM), just-in-time (JIT) compiler, garbage collection, and threads. The benchmark requires two systems: (1) the system under test or SUT and (2) the control and collection system (CCS) [15] with communication between the systems established via Ethernet [16],[3] as shown in Figure 3.1. The SUT runs the workload and is connected to a power meter. The power meter, in turn, is connected to the CCS. The CCS collects the performance and power data passed to it by the SUT and power meter, respectively.

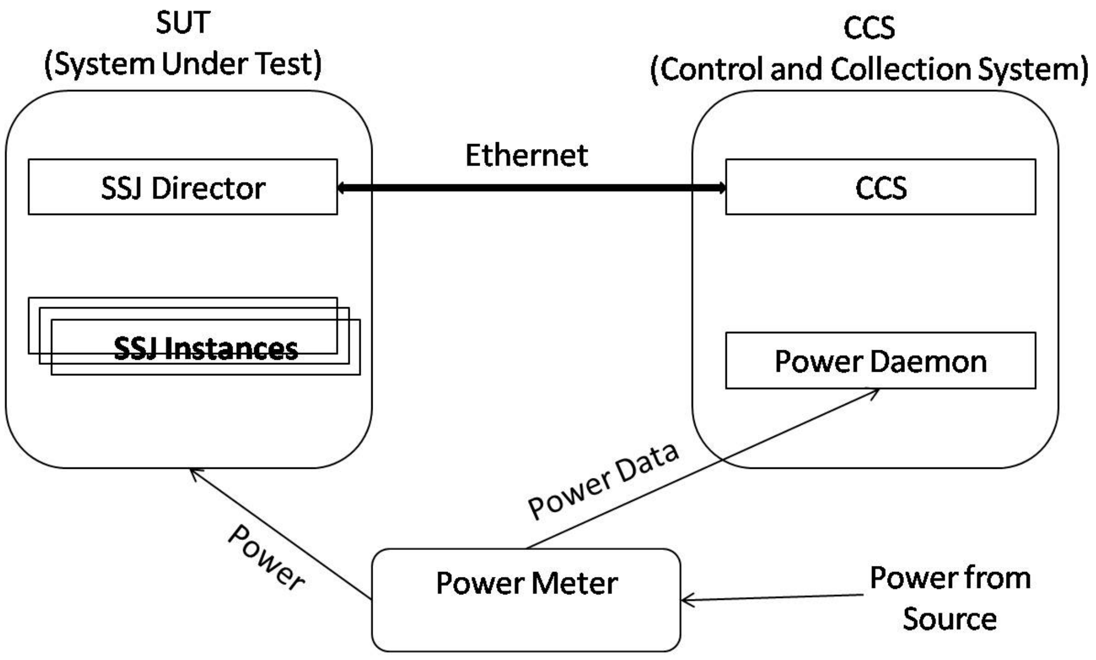


Figure 3.1: Setup for SPECpower

The SPECpower benchmark is designed to produce consistent and repeatable performance and power measurements. It executes different type of transactions and the transactions are grouped together in *batches* for scheduling purposes. Each *load-level* is achieved by controlling delay between the arrival of batches.

[3]SUT and CCS can be the same system. Communication is established via Ethernet only if the systems are different.

More specifically, 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 [18]. The benchmark starts with a calibration phase, which determines the maximum throughput. The calibrated throughput is set as the throughput target for 100% load-level. The throughput 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, 40% is 40,000 and so on. The throughput is measured in server-side Java operations per second (ssj_ops).

The benchmark supports a set of configurable parameters.[4] For example, the maximum target throughput and the batch size can be manually configured. We refer the reader to [17] for further information on configurable parameters. The flexibility, coupled with the consistency and repeatability of SPECpower, allows us to evaluate the applicability of newer power-management interfaces, such as RAPL, to enterprise-class server workloads.

3.1.2 Overview of SPECweb Benchmark

SPECweb is an industry-standard benchmark for measuring the front-end performance of a web server. 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, as shown in Figure 3.2:

[4]Only a subset of these parameters can be changed for compliant runs.

1. Client: It runs the application program that sends HTTP requests to the server and receives the corresponding response from the server.

2. Web Server: It handles the requests issued by the client. This is also the system under test (SUT) for this benchmark.

3. Back-End Simulator (BeSim): It emulates a back-end application server. The web server communicates with BeSim to retrieve specific information required to complete a request from one of the clients.

4. Prime Client: It initiates and controls the clients and also initializes the web server and BeSim. It collects performance results for the benchmark.

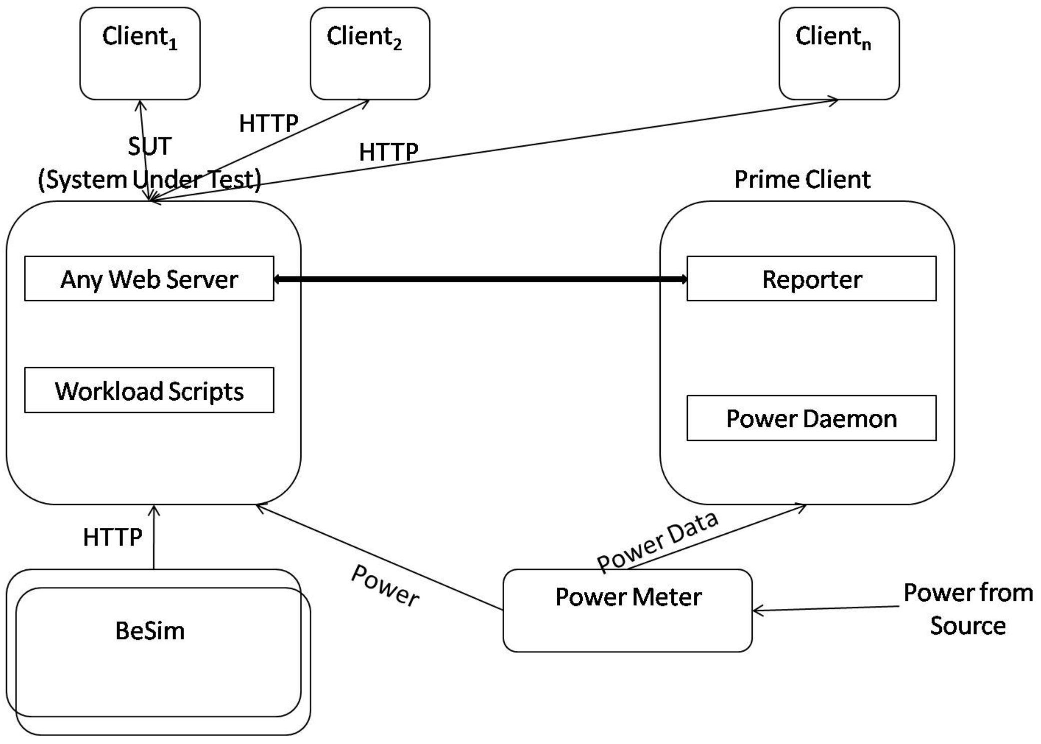


Figure 3.2: Setup for SPECweb

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 performance metrics for response time, 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. 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 [49] for a full list of configurable parameters.

3.1.3 Experimental Setup

The SUT for our experiments is a dual-socket node with Intel Xeon E5-2665 processors (Intel Romley-EP). The node has a total of 16 cores and 32 cores when hyperthreading is ON. It has 256 GB of memory and runs a Linux kernel version 3.2.0. We used a Yokogawa WT210 power meter for full system power measurements. We use RAPL interfaces to measure power consumption of subsystems.

The PKG and DRAM domains expose a MSR interface that provides information on the ranges of values that can be specified for a particular RAPL domain for limiting its power consumption. This includes maximum time window, maximum power, and minimum power. The range of per-socket values on our experimental platform is given in Table 3.1.

Table 3.1: Per-Socket Parameter Range (MTW = Maximum Time Window, MaxP = Maximum Power, MinP = Minimum Power). Multiply by two for Full Two-Socket System.

Domain/Range	MTW	MaxP	MinP
PKG	45.89 ms	180 watts	51 watts
DRAM	39.06 ms	75 watts	15 watts

Setup for SPECpower

The CCS has an Intel Xeon E5405 processor with dual quad-core CPUs and 8 GB of RAM. The CCS runs Linux kernel version 2.6.32. The CCS and SUT were connected through a gigabit Ethernet network. We used all the cores in SUT for our experiments. Eight JVMs with four threads for each JVM were used as the configuration for SPECpower. The four threads in each JVM were pinned to two adjacent physical cores on the SUT using *numactl*. 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 throughout our experiments, we configured the *input.load_level.target_max_throughput* parameter to achieve the same performance for each run. 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. This value was determined by averaging 10 calibration runs. We changed the runtime for each load-level to 120 seconds using the *input.load_level.length_seconds* parameter and the pre- and post-measurement interval to 15 seconds in order to reduce the total runtime of the benchmark. We use 1000 as our batch size as there is minimal to no effect on power due to batch sizes (see Appendix A). On an average, the SUT consumes 120 watts at idle and 330 watts at 100% load-level of SPECpower. We would like to stress that the system consumes 36.51% of peak power[5] even when idling.

Setup for SPECweb

We used 26 clients, one prime client and two BeSim for our experiments. The prime client is an Intel Xeon E5405 processor with two quad-core CPUs and 8 GB of RAM. The BeSims had two dual-core AMD Opteron 2218 processors with 4 GB of RAM. In this thesis, we benchmark only the SPECweb_PHP_Ecommerce workload. We used an 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 up to 2 Gbps. Note that the bonded Ethernet link is only set up for SPECweb. In our experiments for SPECweb, 100% load-level corresponds to 13000 SUS. This value was determined using empirical analysis (see Appendix B). In addition to the sessions, all our experiments also maintain the response-time criteria. In our case, 95% (*TIME_GOOD* parameter) and 99% (*TIME_TOLERABLE* parameter) of the requests need to have response times less than three and five seconds, respectively. These response-time constraints are default values and used in the compliant runs. The load-level is changed by manually modifying the SIMULTANEOUS_SESSIONS parameter in the input configuration. We modified the RUN_SECONDS input parameter to 420 seconds to reduce the runtime of the benchmark. Since we focus only on the processor package and memory power management, we load all the data associated with the Ecommerce workload into Random-Access Memory File System (RAMFS) to keep the data set in memory and minimize the involvement of disks. On an average, the SUT consumes 120 watts

[5]Power consumed at 100% load-level.

when idling and 219 watts at 100% load-level of SPECweb. In case of SPECweb, the system consumes 54.88% of peak power when idling.

3.2 Analysis of Power Consumption and Energy Proportionality for Enterprise Applications

In this section, we characterize the power consumption of the SPECpower and SPECweb benchmarks and analyze energy proportionality from the perspective of the entire system as well as each RAPL domain (i.e., subsystem). We also present our results for the instantaneous power profile analysis of the SPECpower and SPECweb benchmarks.

3.2.1 Average Power Consumption Analysis

As discussed earlier, we seek to analyze the energy proportionality of the system. The deviation of the power curve of the system from the energy-proportional power curve is of particular interest to us. To illustrate with an example, Figure 3.3 shows the average power consumed by the testbed running SPECpower benchmark at each load-level, the hypothetical linear power trend and the hypothetical energy-proportional (EP) power curve for the system. We would like the area (shown in Figure 3.4) between the system and the EP power curve to be as small as possible. Henceforth, this 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 (show in Figure 3.5) between the system power trend and linear curve. Henceforth, this area will be referred to as Linear Deviation Gap (LDG).

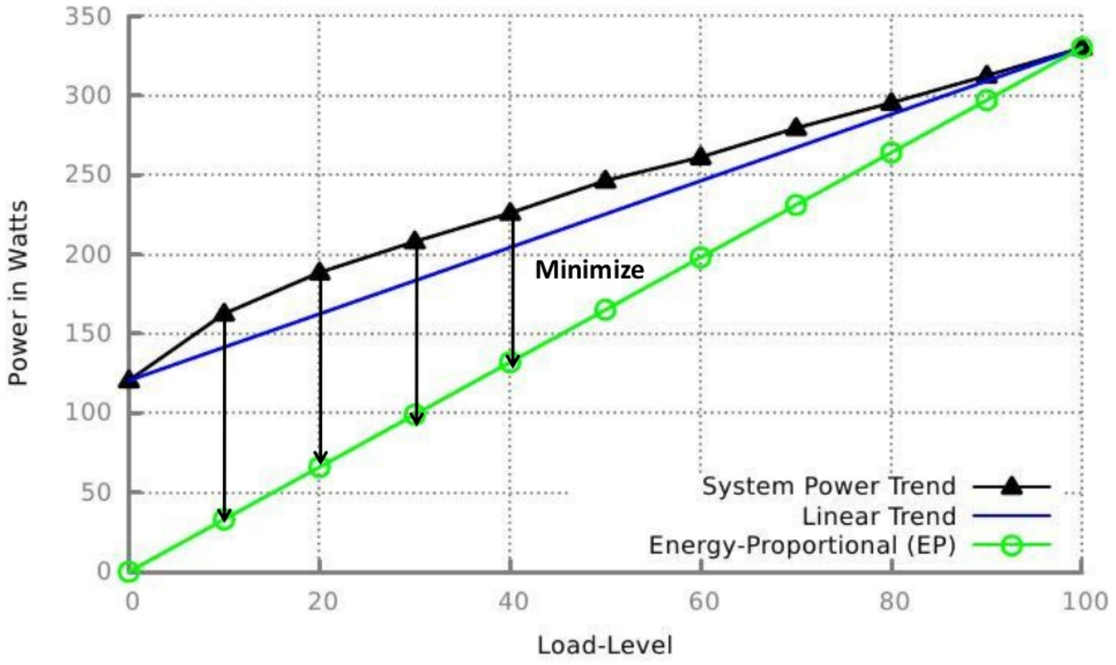


Figure 3.3: Illustration of Energy Proportionality and Linear Deviation

Properties of Energy-Proportional Systems

Barroso et al. [24] advocated the design of energy-proportional systems by addressing power characteristics of the server and the behavior of enterprise-class server workloads. They proposed two properties of energy-proportional systems – low idle power and wide dynamic power range. These two properties are particularly illustrated by the EP curve in Figure 3.3. The EP curve consumes zero power when idling (i.e., at 0% load-level) and has a wide dynamic power range. In this thesis, we will quantify the idle power and the dynamic power range as percentage of peak power (i.e., the power consumed at 100% load-level).

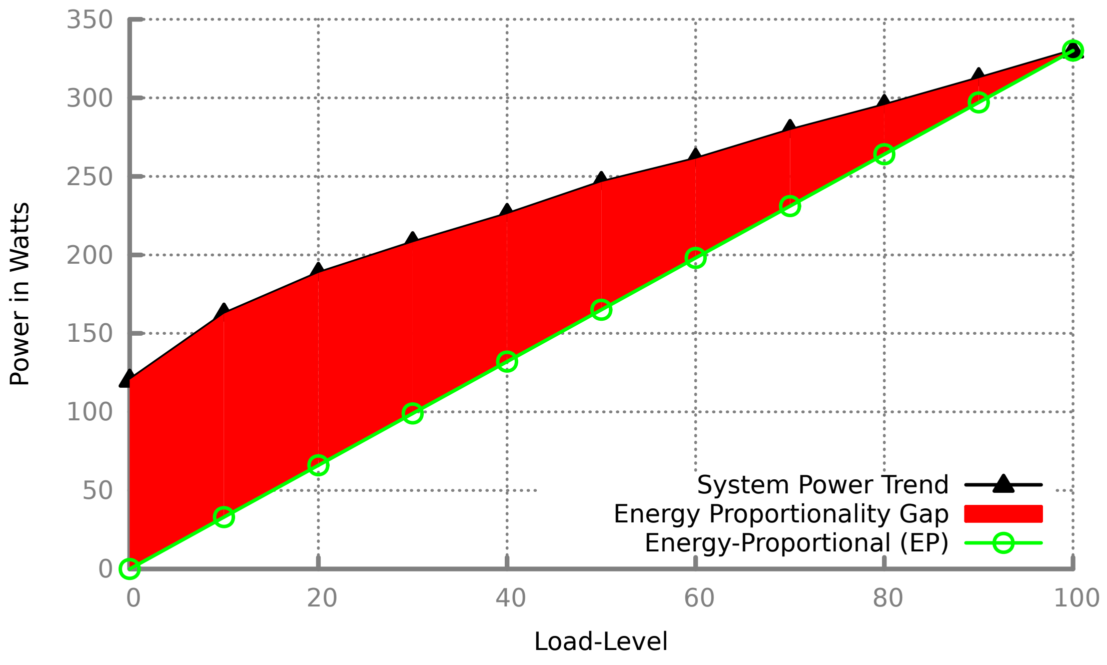


Figure 3.4: Illustration of Energy Proportionality Gap

Energy Proportionality Metric

We quantify the EPG using two different metrics: (1) the EP metric [47] and (2) the Proportionality Gap (PG) metric [59]. Each of these metrics serve different purposes and quantifies the energy proportionality of the system along different dimensions. The EP metric is calculated as shown in Equation (3.1) where $Area_{System}$ and $Area_{EP}$ represent the area under the system and EP power curves, respectively. A value of one for the metric represents an energy-proportional system. A value of zero 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 (see Section C.2).[6] The EP metric gives a perspective of the energy proportionality of the system at the

[6]Originally, the EP metric proposed in [47] varied only between 0 and 1 (i.e., it did not account for better than energy-proportional systems). However, in this thesis we extend EP metric to account for better than energy-proportional system (i.e., $0 <$ EP metric < 2).

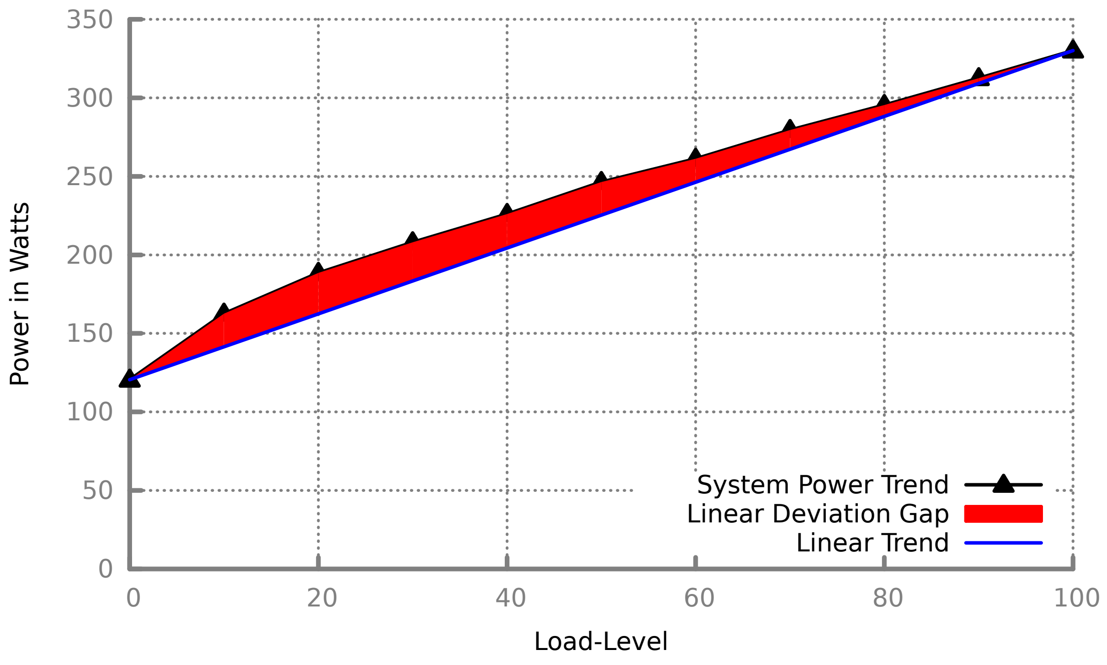


Figure 3.5: Illustration of Linear Deviation Gap

full system-level.

$$EP = 1 - \frac{Area_{System} - Area_{EP}}{Area_{EP}} \tag{3.1}$$

The PG metric[7] is calculated, as shown in Equation (3.2) where $X\%$ represents $X\%$ load-level. As observed, the PG metric defines the EPG at individual load-levels. For an energy-proportional server, the PG for all utilization should be zero .

$$PG_{X\%} = \frac{Power_{System@X\%} - Power_{EP@X\%}}{Power_{System@100\%}} \tag{3.2}$$

The LDG is quantified using the Linear Deviation (LD) metric [59]. The LD metric[8] is calcu-

[7]PG stands for proportionality gap [59].
[8]LD stands for linear deviation [59].

lated using Equation (3.3). For an linear energy-proportional system, the LD metric will be zero. LD metric > 1 or < 1 indicate superlinear or sublinear energy-proportional systems, respectively.

$$LD = \frac{Area_{System}}{Area_{Linear}} - 1 \tag{3.3}$$

We will use the properties described in Section 3.2.1 along with the EP metric, PG metric and LD metric to quantify the energy proportionality. We will also look at the EPG and LDG both at full system- and subsystem-levels in the rest of the sections.

Methodology

We used the energy meters exposed in each RAPL domain to determine the power dissipated in each domain. In all our results, we report the mean of the average power[9] over ten runs for the domain-level power consumption. For full-system power measurement, we have followed the power measurement methodology specified and developed by the SPEC organization for the SPECpower and SPECweb benchmarks [14].

Analysis of System- and Subsystem-Level Energy Proportionality

In this section, we present the details on the power consumption of SPECpower and SPECweb at a subsystem-level. We were able to profile the benchmark at a granularity that has not been possible until the advent of Intel Sandy Bridge by using the on-chip energy meters exposed by the RAPL

[9]Average power is calculated as (final energy reading - initial energy reading)/time.

interfaces. Our results provide insight into the energy proportionality of a system as a whole as well as at the RAPL domain-level.

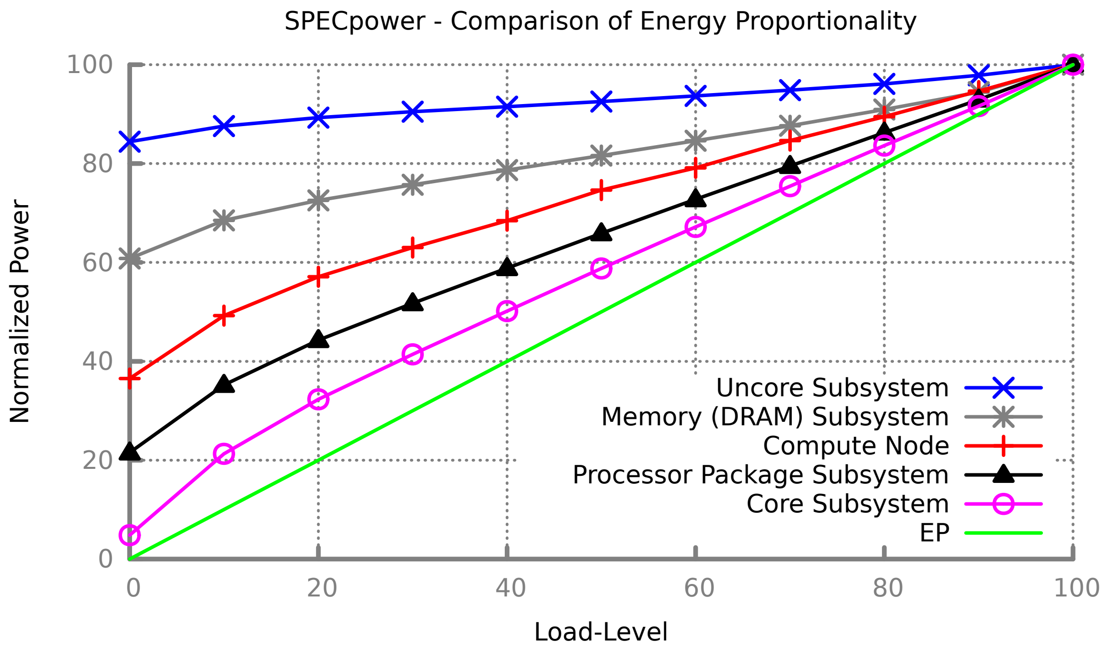


Figure 3.6: Analysis of SPECpower Energy Proportionality

Figures 3.6 and 3.7 describe the energy proportionality of the full system and different subsystems for SPECpower and SPECweb, respectively. The Y-axis represents the percentage of peak power consumed by the system or subsystem, and the X-axis represents the load-level. As a result, the EP curve (green line which connects circles) consumes 40% of peak power at 40% load-level, 60% of peak power at 60% load-level and so on. Figures 3.6 and 3.7 are also a compact comparison of the energy proportionality of different components of the system. As mentioned earlier, we will quantify the energy proportionality using the EP, PG and LD metrics and the desired properties of an energy-proportional system.

Full-System Energy Proportionality The energy proportionality of the full system is represented by the red line in Figures 3.6 and 3.7 for the SPECpower and SPECweb benchmarks, respectively. The EP metric for the full system is 0.54 and 0.29 for SPECpower and SPECweb respectively. The full system idles at 36.51% and 54.88% of peak power for SPECweb and SPECpower. Therefore, it is impossible to achieve energy-proportional operation for load-levels less than 36% in the case of SPECpower and 54% in the case of SPECweb. The dynamic power range is 63.48% for SPECpower and 45.11% for SPECweb.[10]

Package (PKG) Energy Proportionality The EP metric value for the PKG subsystem is 0.70 and 0.44 for SPECpower and SPECweb, respectively. It is also worth noting that the power profile of the package and full system follow a similar trend for both the benchmarks, indicating a strong correlation between them.[11] The package subsystem idles at 21.55% and 34.47% for SPECpower and SPECweb, respectively. The dynamic power range for SPECpower is 78.44% and SPECweb is 65.52%. In general, due to its better EP metric, the lower idle power and high dynamic power range of the package subsystem results in better energy-proportionality than the package subsystem.

***Core* (PP0) Energy Proportionality** The line that connects the circles in Figures 3.6 and 3.7 describes the energy proportionality of the PP0 domain. We observe that this subsystem has near energy-proportional power profile for the SPECpower benchmark. However, it is less energy-proportional in the case of the SPECweb benchmark. It has a EP metric value of 0.85 in the case of SPECpower and 0.63 in the case of SPECweb. This subsystem idles at 5.74 watts (4.83%

[10]Dynamic power range is calculated as power consumed at 100% load-level - 0% load-level.
[11]The Pearson correlation is greater than 0.99.

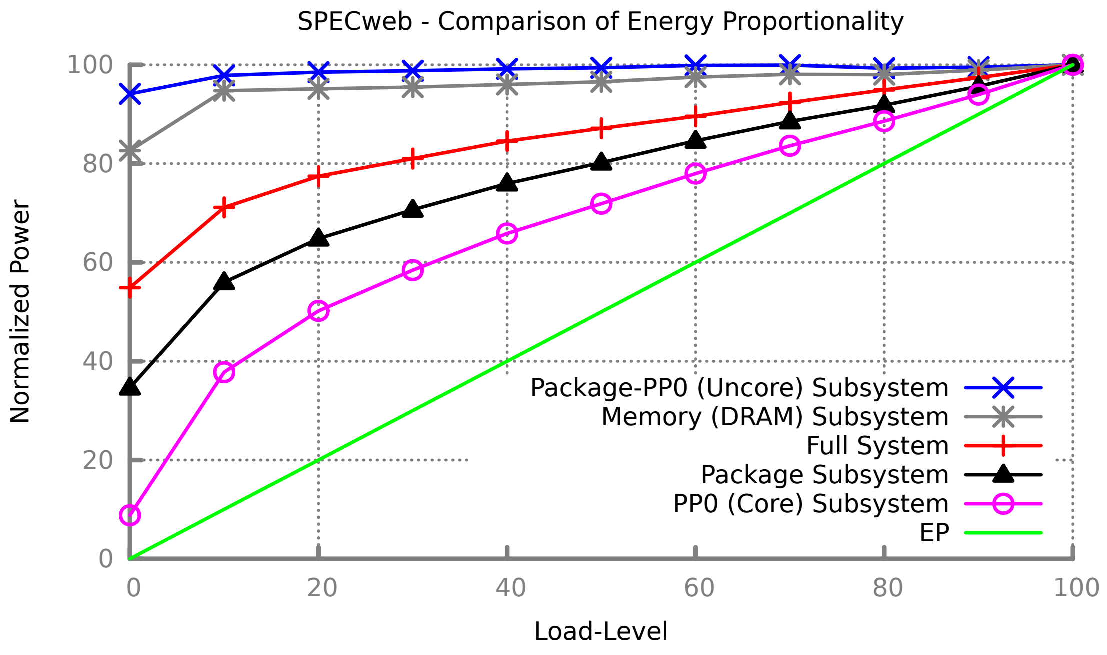


Figure 3.7: Analysis of SPECweb Energy Proportionality

of peak power) and has a dynamic power range of 95.16% of peak power for SPECpower. The idle power and dynamic power range are 8.80% and 91.19% of peak power for the SPECweb benchmark, respectively. The low idle power coupled with the high dynamic power range makes this subsystem suitable to be operated at different power-performance trade-offs.

***Uncore* (Package-PP0) Energy Proportionality** The *uncore* subsystem's power consumption remains almost constant irrespective of the load-level with an EP metric value of 0.14 for SPECpower and 0.02 for SPECweb. The *uncore* subsystem has the greatest EPG, and as a result, exhibits the worst power consumption trend among the full system and RAPL domains from the perspective of energy-proportional power scaling. It idles at 84.41% and 94.13% of peak power for SPECpower and SPECweb, respectively. It has the least dynamic power range among all systems and subsystems at 15.58% for SPECpower and 5.86% for SPECweb.

Memory (DRAM) Energy Proportionality The memory subsystem has an EP metric value of 0.36 for SPECpower and 0.07 for SPECweb. In the case of SPECweb, the memory power trend closely follows the uncore power trend which makes it less energy-proportional. The lack of energy proportionality in the memory subsystem for the SPECweb benchmark can be attributed to the use of RAMFS to house the data required by the web server. This subsystem idles at 60.80% and 82.62% for SPECpower and SPECweb, respectively.

To summarize, Table 3.2 describes our results on the energy proportionality of full system and its associated subsystems.

Table 3.2: Summary of Full System- and Subsystem-Level Energy Proportionality Analysis. Note: Idle Power and Dynamic Power Range are Represented as Percentage of Peak Power.

Subsystem	**Benchmark**	**EP Metric**	**Idle Power (%)**	**Dynamic Power Range (%)**
Full System	SPECpower	0.54	36.51	63.48
	SPECweb	0.29	54.88	45.11
Package (PKG)	SPECpower	0.70	21.55	78.44
	SPECweb	0.44	34.47	65.52
Core (PP0	SPECpower	0.85	4.83	95.16
	SPECweb	0.63	8.80	91.19
Uncore (Package-PP0)	SPECpower	0.14	84.41	15.58
	SPECweb	0.02	94.13	5.86
Memory (DRAM)	SPECpower	0.36	60.80	39.19
	SPECweb	0.07	82.62	17.37

Analysis of Load-Level Energy Proportionality

The PG metric allows us to look at the energy proportionality of a server at each load-level. Figure 3.8 shows the PG metric at each load-level for the SPECpower and SPECweb benchmarks. Similar to EP metric, the *uncore* and *core* subsystem have the worst and best PG metric for all load-levels for both the benchmarks.

The *uncore* subsystem's PG decreases linearly from 0% to 100% load-level which again shows that the subsystem's power remains nearly constant, irrespective of the load-level. In case of the PP0 subsystem, there is an increase in the PG metric when the load-level increases from 0% to 10%. This trend shows that the energy proportionality gap at 0% load is better than low utilization levels for both the benchmarks. The proportionality gap becomes better than at the 0% load-level only at the 70% and 80% load-levels for SPECpower and SPECweb benchmarks, respectively, for the PP0 subsystem. Such trends can be seen for the package subsystem and full system as well.

In summary, the core is the most energy-proportional subsystem and the uncore is the least energy-proportional subsystem.

Analysis of Linear Deviation

Table 3.3 shows the LD metric for both benchmarks at each RAPL domain and full system. The LD metric for all subsystems is always positive as none of them have a sub-linear energy proportionality trend. This observation also provides evidence that there is opportunity to improve the energy proportionality by improving (i.e., decreasing) the LD metric (see Section 3.3).

Table 3.3: Summary of Full System- and Subsystem-Level Linear Deviation Analysis.

Subsystem	**Benchmark**	**LD Metric**
Full System	SPECpower	0.067
	SPECweb	0.101
Package (PKG)	SPECpower	0.066
	SPECweb	0.151
Core (PP0	SPECpower	0.095
	SPECweb	0.254
Uncore (Package-PP0)	SPECpower	0.008
	SPECweb	0.022
Memory (DRAM)	SPECpower	0.017
	SPECweb	0.053

3.2.2 Instantaneous Power Consumption Analysis

In this section, our main goal is to visualize the opportunities for power limiting. We collected instantaneous power profiles for five load-levels.

Methodology

Our results are shown as Cumulative Distribution Functions (CDF). The CDFs present the percentage of time spent at or below a given percentage of the maximum power limit. We refer the reader to Table 3.1 for the maximum power limit for each subsystem. We collect the instantaneous power profile of the package and memory subsystems at 50 ms resolution.[12] The results are normalized to their respective maximum power limit.

[12]We profiled the instantaneous power at lower than 50 ms time resolution as well. However, 50 ms is where we start to see negligible overhead due to the profiling.

Instantaneous Power Analysis for Package (PKG) Subsystem

Figures 3.9 and 3.10 show the instantaneous power consumption for the package subsystem for five different load-levels of SPECpower and SPECweb. We observe that the maximum power consumed by 100% load-level of SPECpower as indicated by its CDF is lower than 0.5 normalized power. This indicates that the maximum power consumed while executing SPECpower is less than 50% of maximum power limit. The upper limit for the package power consumption for the SPECweb benchmark is also less than 50% of the maximum power limit. The lowest point in the CDF of each workload corresponds to the minimum power consumed. At no point during the execution of that load-level, the subsystem consumes lesser power than the lowest point in the CDF. For example, 100% and 60% load-levels of SPECpower do not consume less than 40% and 20% of normalized power respectively. The shape of the curves indicate that each load-level spends most of the time consuming a narrow range of power. For instance in the case of SPECpower, 80% load-level spends most of the time consuming power between 34% and 46% of maximum power limit. For some load-levels of both the benchmarks, we see that the CDF curve intercepts 100% line at flat slopes. This indicates that at few intervals, the instantaneous power consumption spikes. For both the benchmarks, we will benefit by removing the relatively few intervals where the workload has a power spike. Power limiting can help in such cases to remove these intervals. We also observe that the power range decreases with increase in load-level. In the case of SPECpower, 100% load-level has a power range from 0.40 to 0.50 of the normalized power whereas 20% load-level has a power range from 0.10 to 0.32 of the normalized power.

Instantaneous Power Analysis for Memory (DRAM) Subsystem

Figures 3.11 and 3.12 describe the instantaneous power consumption for the memory subsystem for five different load-levels of SPECpower and SPECweb, respectively. We observe CDF curves similar to the package subsystem for both the benchmarks. The minimum normalized power consumed at each load-level is higher than the corresponding observation for the package subsystem. This is an expected behavior as memory subsystem idles at a higher percentage of peak power than the package subsystem (see Table 3.2). Similar to the package subsystem, each load-level spends most of the time consuming a narrow range of power. The 100% load-level for SPECpower consumes 87% to 93% of the peak power limit leaving less opportunity for memory power management than other load-levels. The memory power consumption for the SPECweb benchmark is narrower than SPECpower as all load-levels of SPECweb consume power between 55% to 80% of the peak power limit. In general, there is less opportunity to limit the power consumption of memory than the package subsystem.

In summary, there is opportunity to limit the power consumption of SPECpower and SPECweb at different load-levels below the 50-ms resolution for the package and DRAM subsystems.

3.3 The Potential of Power Limiting

In this section, we discuss the effects of power limiting on the performance and power of SPECpower and SPECweb benchmarks. Specifically, we investigate whether we can achieve energy-proportional operation for these benchmarks by leveraging the RAPL interfaces. Through our results, we will show that most of the power savings comes from the PP0 domain. Our results also indicate that memory subsystem power limiting provides the least opportunity to save power.

3.3.1 Methodology

We run both the SPECpower and SPECweb benchmarks at five different load-levels (from 20% to 100% in steps of 20) under a power limit. Our experiments focus on PP0 and DRAM power limiting. We do not focus on power limiting of the processor package as the *uncore* subsystem contributes little to no power savings at any load-level, and all the power savings came from the PP0 domain while we experimented with processor package power limiting [51]. Our experiments present results for three different power-limiting scenarios:

- *CPUOnly* policy: Performance when only the PP0 subsystem is power limited.
- *MemOnly* policy: Performance when only the DRAM subsystem is power limited.
- *CPU+Mem* policy: Performance when both PP0 and DRAM subsystems are power limited.

In our experiments, we manually configure the power limit using the RAPL interfaces. For the CPUOnly and MemOnly policies, we manually set 15 different power limits below the average power consumption of the corresponding subsystem. These 15 different power limits start from the average power consumption down to 28 watts under the average power consumption at increments of two watts at a time. For the CPU+Mem policy we look at all possible power limits for a total of 225 combinations for each load-level. In this section, we only present the best possible power savings *without* performance degradation for the benchmarks. We only present runs that achieve performance within 1% of the target load-level for SPECpower. In the case of SPECweb, we present results that achieve the target load-level and maintain *TIME_GOOD* and *TIME_TOLERABLE* constraints (see Section 3.1.3). We also use the smallest possible value for the time window for power limiting (i.e., 976 microseconds).

3.3.2 Impact of Power Limiting

Figures 3.13 and 3.14 show the normalized power consumption of five different load-levels of SPECpower, respectively. The values are normalized against the power consumption of a vanilla run when running at 100% load-level.[13] We show the power consumption for the full system and processor package, PP0 and DRAM subsystems. Viewing power consumption in this way allows us to identify whether we achieve energy proportionality at a particular load-level or not.

From the figures, we observe that we achieve energy proportionality for the full system only for 100% and 80% load-levels. However, power limiting reduces the power consumption for other load-levels even though we are not able to achieve energy-proportional operations. Why is energy proportionality unachievable at low load-levels? This is primarily due to the fact that the system consumes 36.51% of peak power even when idling. The CPU+Mem and CPUOnly policies achieves the best power reduction. The MemOnly policy achieves negligible power reduction. In case of power consumption for the processor package, we are able to achieve energy-proportional

[13]Vanilla run is the baseline run without RAPL power limiting.

operation for all loads-levels except 20% load-level. Moreover, the reduction in power consumption is more when compared to the full system. This is an expected outcome as we only have power limiting control over the processor package, PP0 (which is a part of the processor package) and DRAM subsystems. Through power limiting, we achieve energy-proportional operation for all load-levels when we look at the PP0 domain in isolation. As mentioned earlier, we achieve negligible power reduction from the DRAM subsystems while meeting the performance constraints of the benchmarks. In case of the subsystems, the different power limiting policies have the same effect as seen for the full system (i.e., using CPU+Mem and CPUonly results in the best possible power reduction whereas using MemOnly results in negligible power savings).

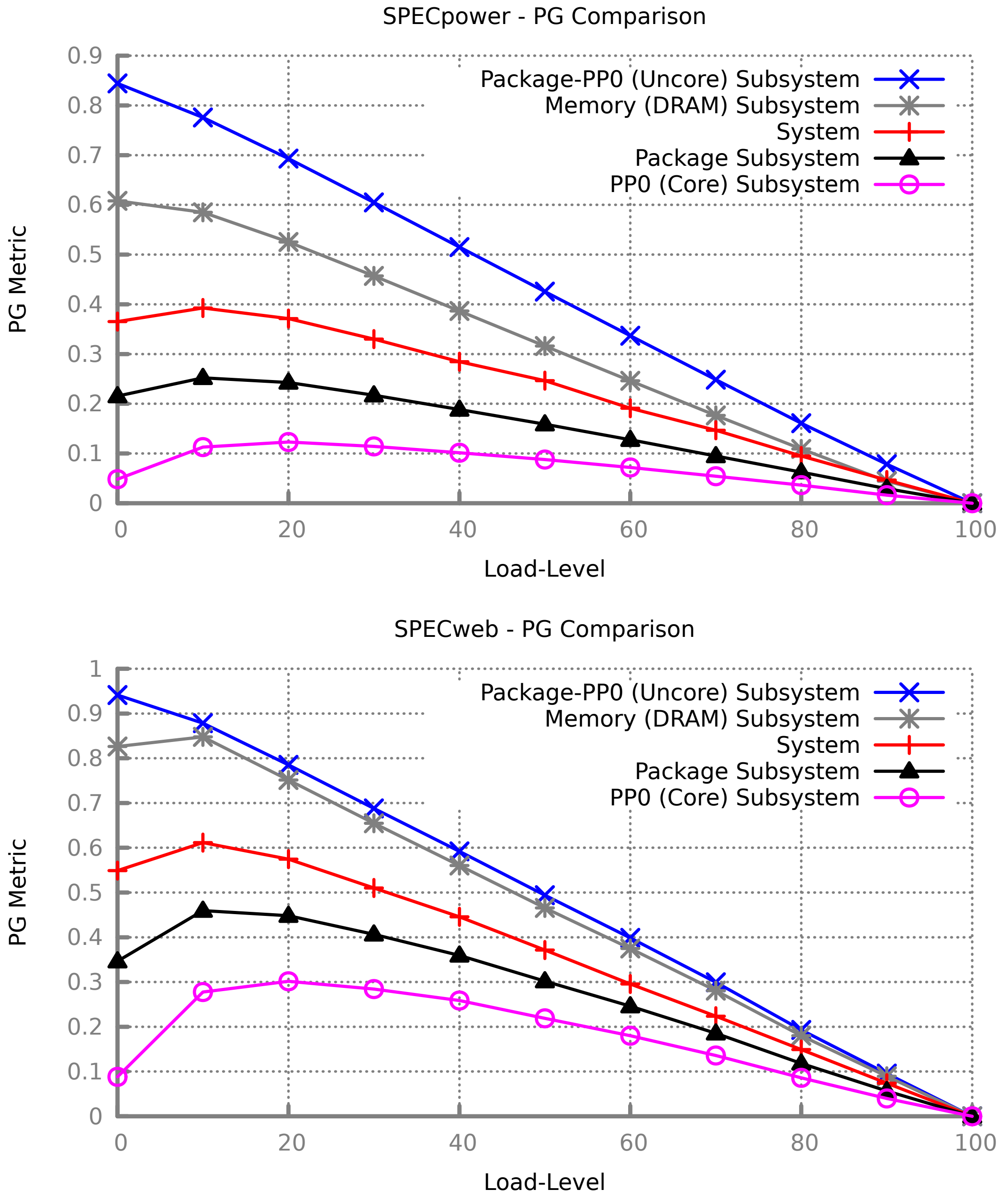


Figure 3.8: Analysis of Load-Level Energy Proportionality

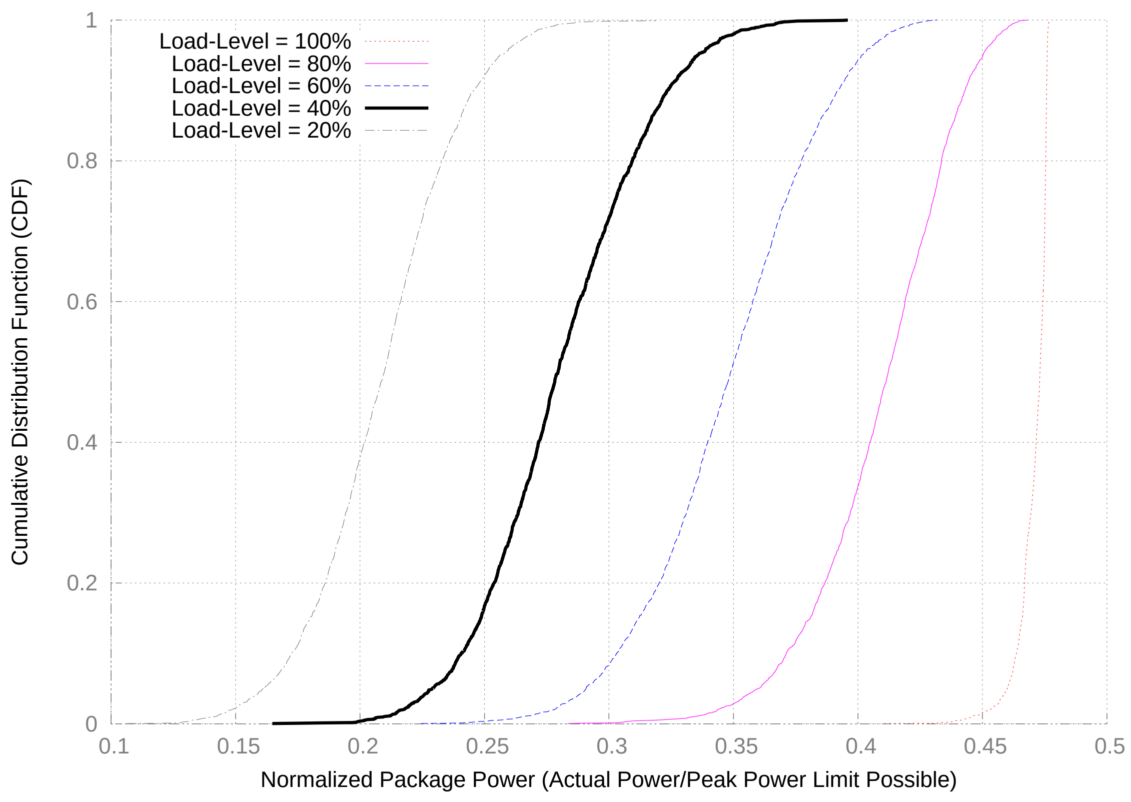


Figure 3.9: Analysis of SPECpower Instantaneous Power Consumption For Package (PKG) Subsystem

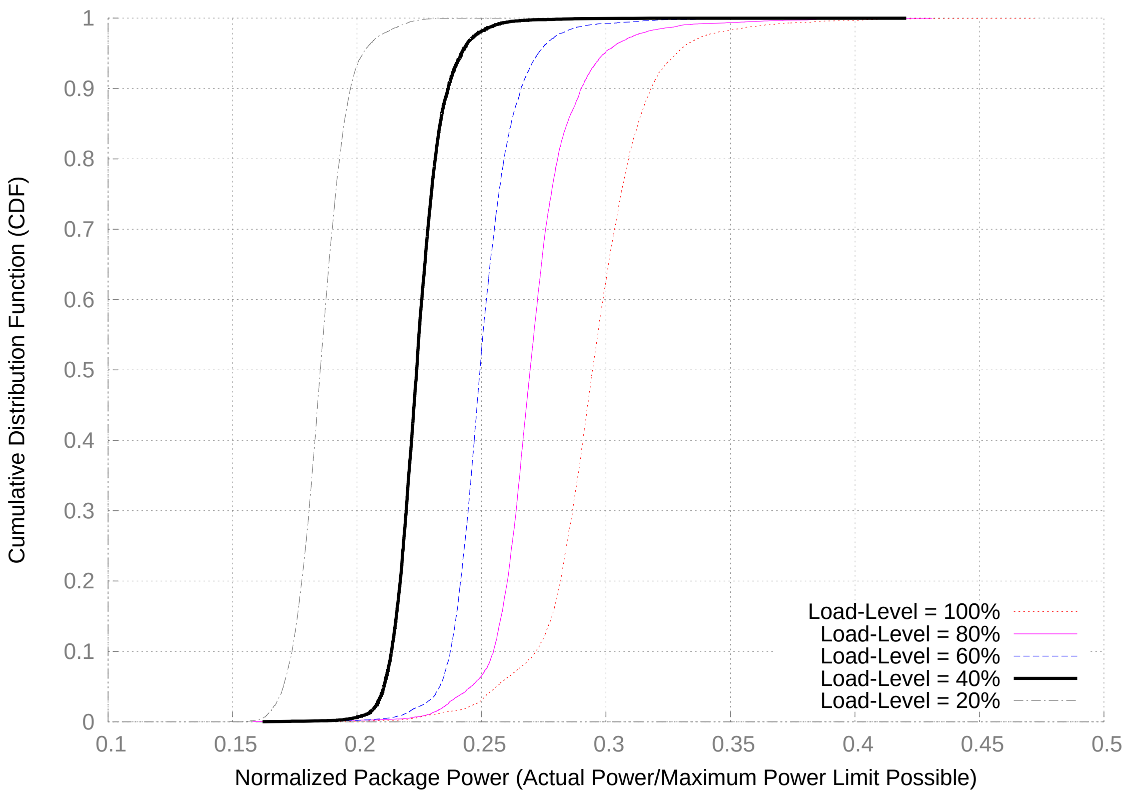


Figure 3.10: Analysis of SPECweb Instantaneous Power Consumption For Package (PKG) Subsystem

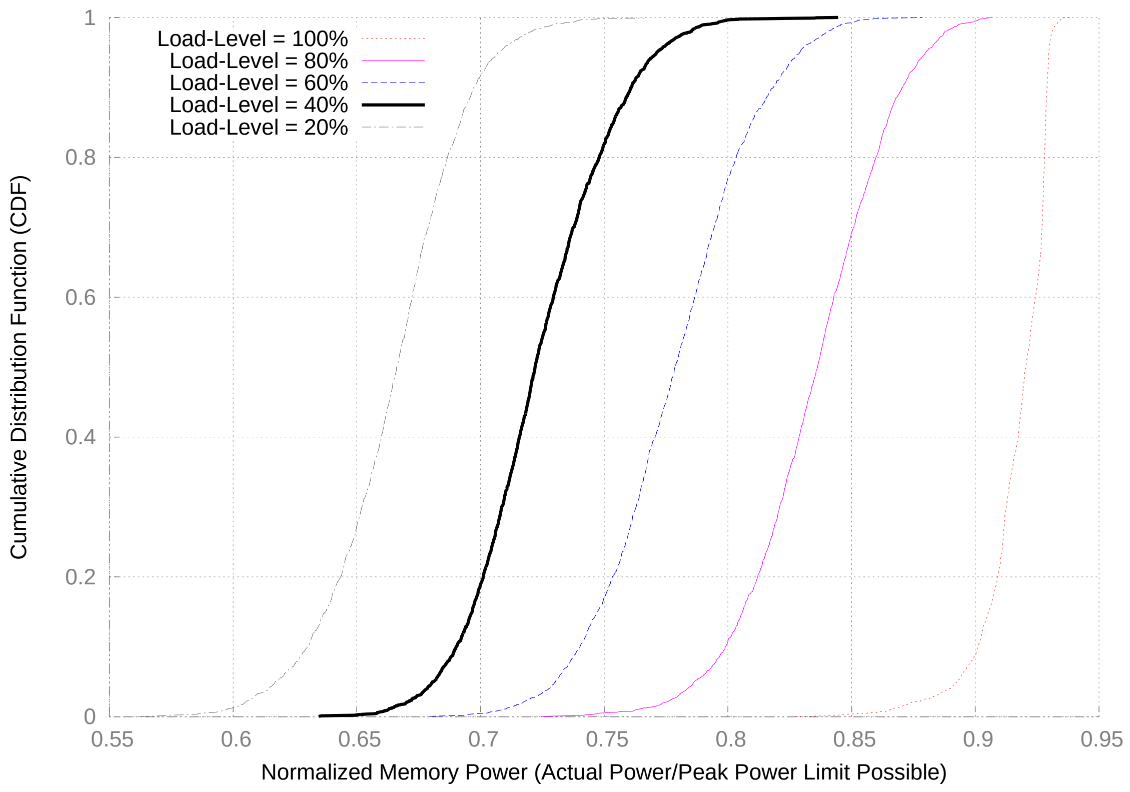


Figure 3.11: Analysis of SPECpower Instantaneous Power Consumption For Memory (DRAM) Subsystem

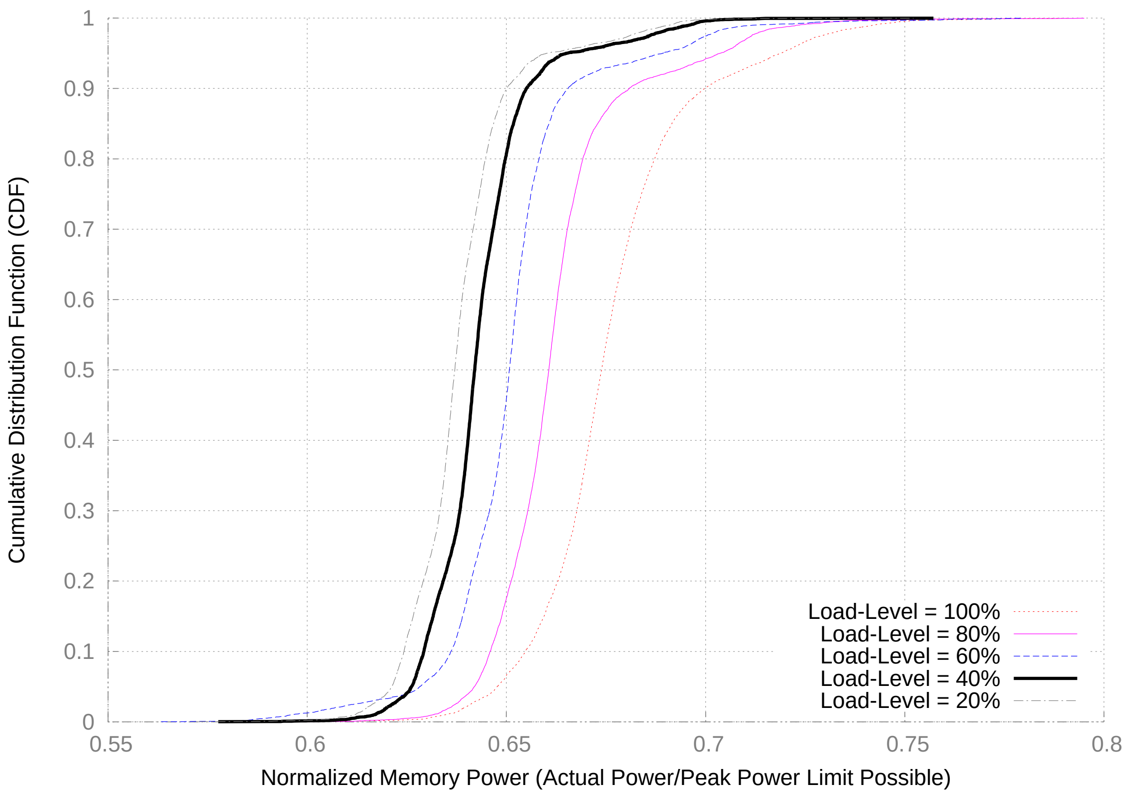
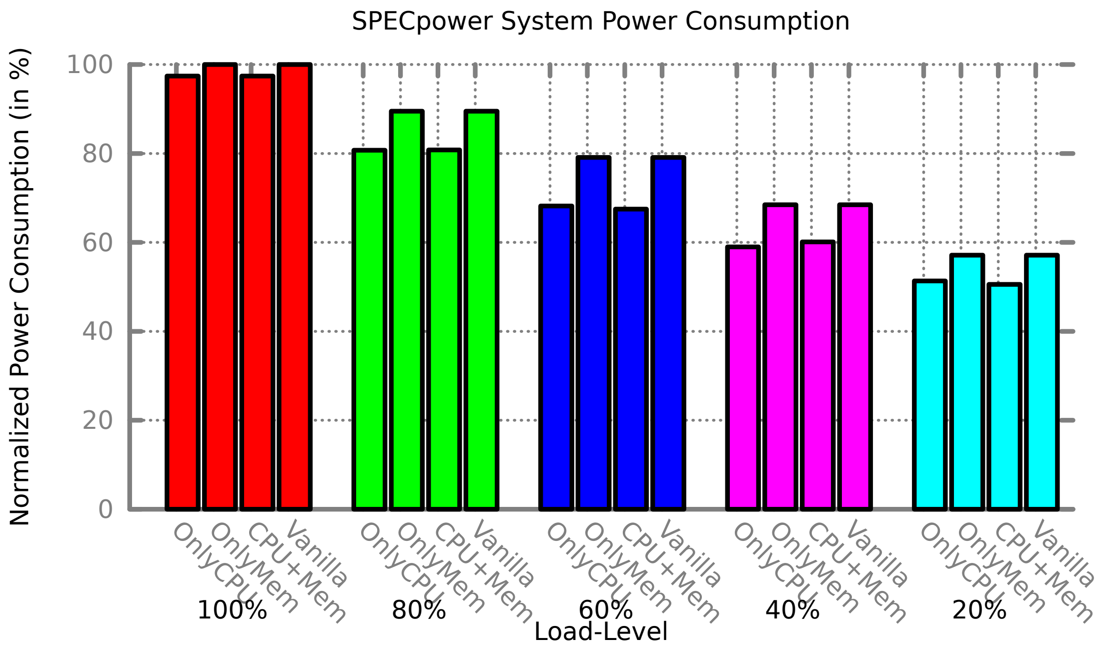


Figure 3.12: Analysis of SPECweb Instantaneous Power Consumption For Memory (DRAM) Subsystem



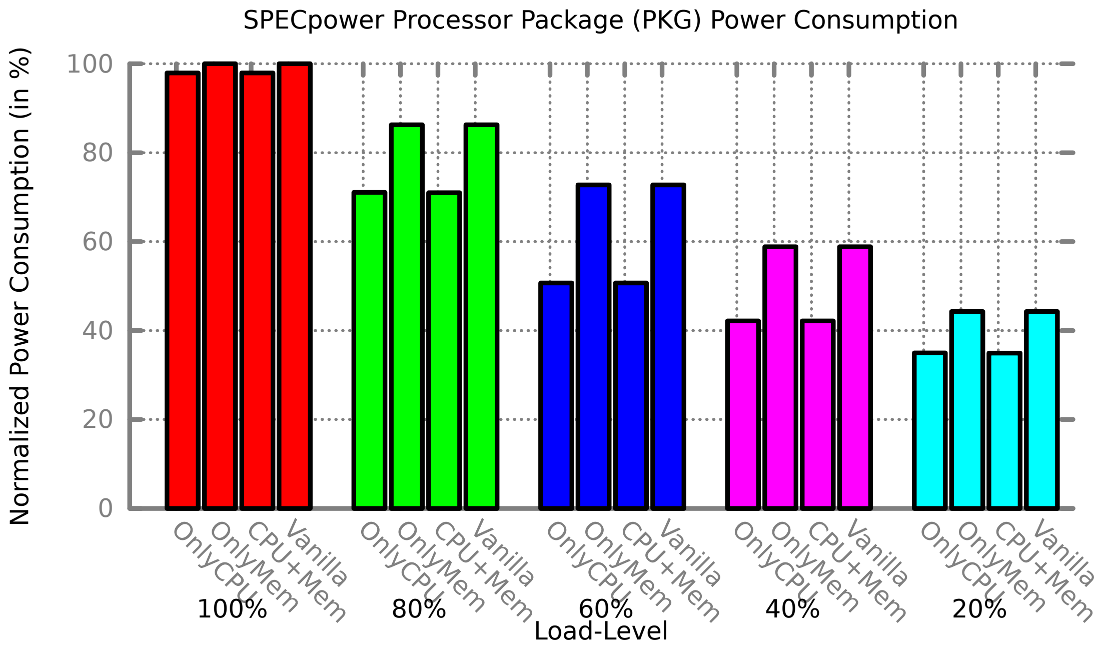


Figure 3.13: Impact of Power Limiting on SPECpower (System and PKG)

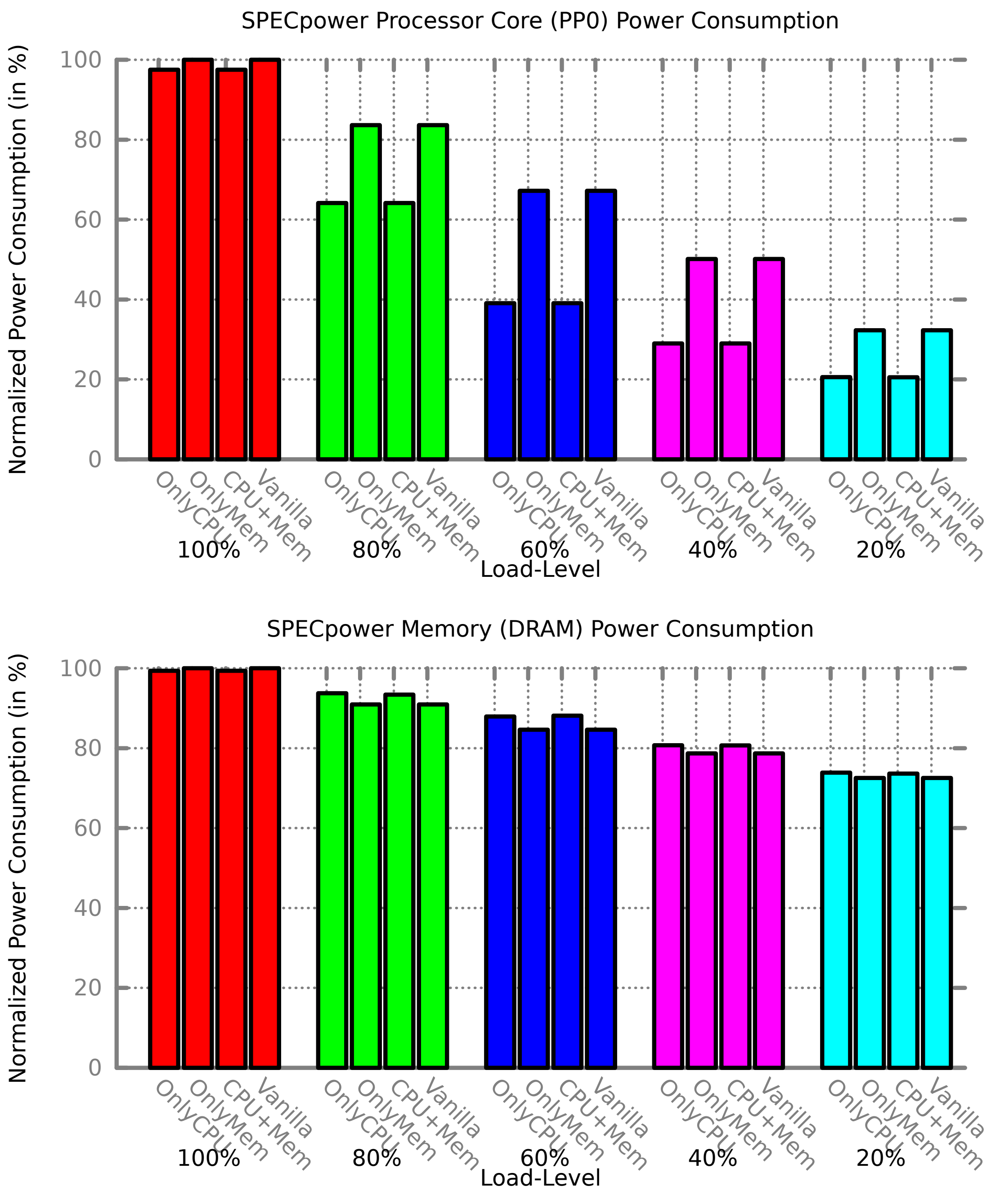
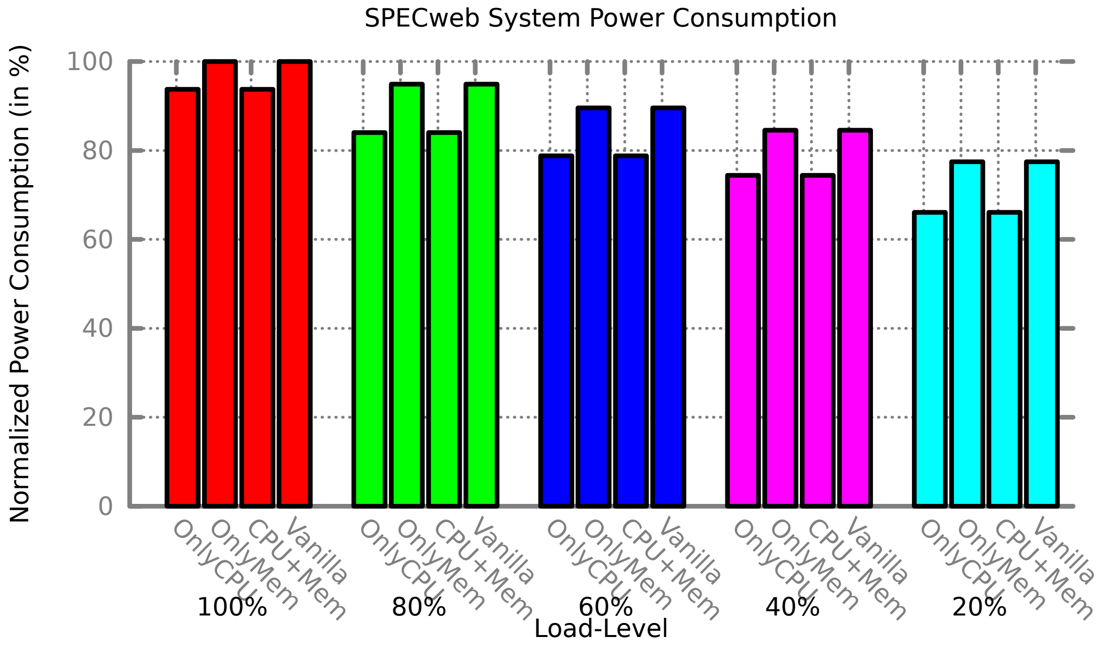


Figure 3.14: Impact of Power Limiting on SPECpower (PP0 and Memory)

Figures 3.15 and 3.16 show the normalized power consumption of five different load-levels of the SPECweb benchmark, respectively. Similar to SPECpower, we show the power consumption for the full system as well as the processor package, PP0 and DRAM subsystems. We achieve energy-proportional operation only for the 100% load-level in the case of SPECweb. However, like SPECpower, we achieve power savings for other load-levels. The power reduction for SPECweb is less than the power reduction seen for SPECpower as SPECpower is *more* energy-proportional than SPECweb (see Section 3.2.1) on our system under test. Similar also to SPECpower, we note that the SPECweb benchmarks idle at 54.88% of its peak power. When looking at the power consumption of processor package in isolation, we achieve energy-proportional operation for the 100% and 80% load-levels, respectively. The PP0 domain provides the most power reduction and achieves energy-proportional operation for all the load-levels except 20% load-level. The memory subsystem does not contribute much to the power reduction. The CPU+Mem and CPUOnly policies delivers the most power reduction possible.



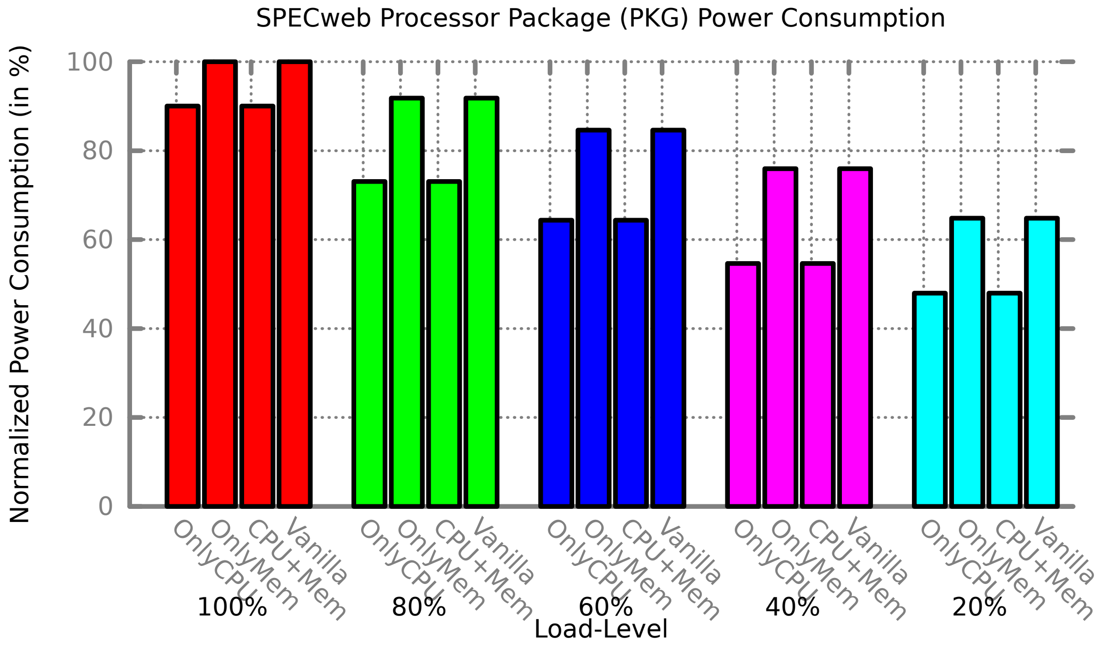
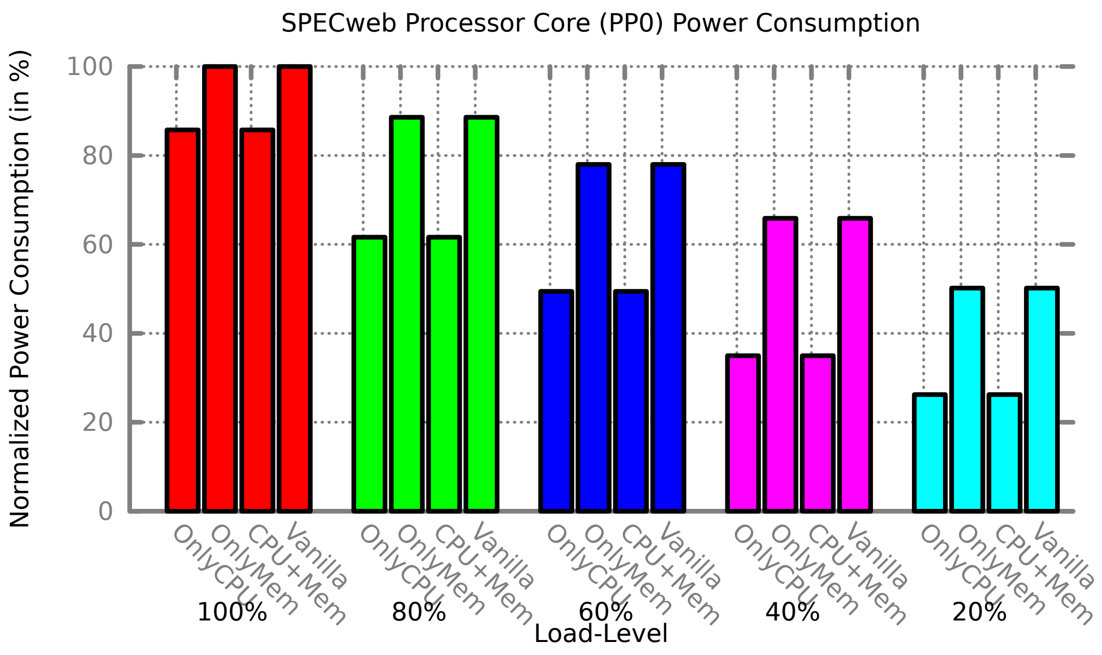


Figure 3.15: Impact of Power Limiting on SPECweb (System and PKG)



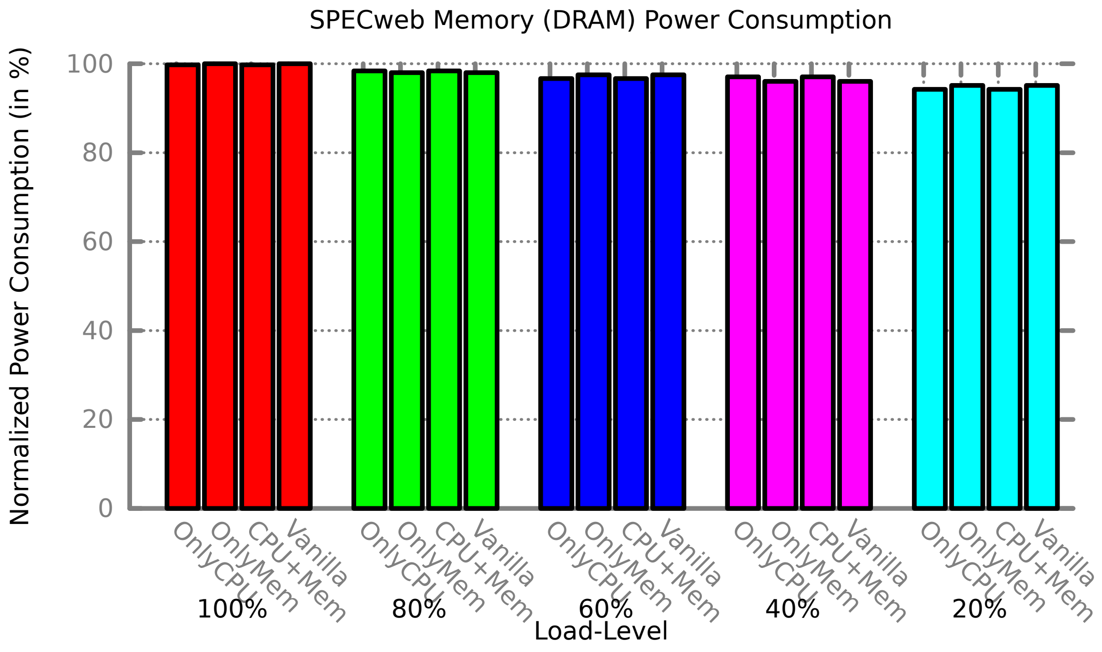


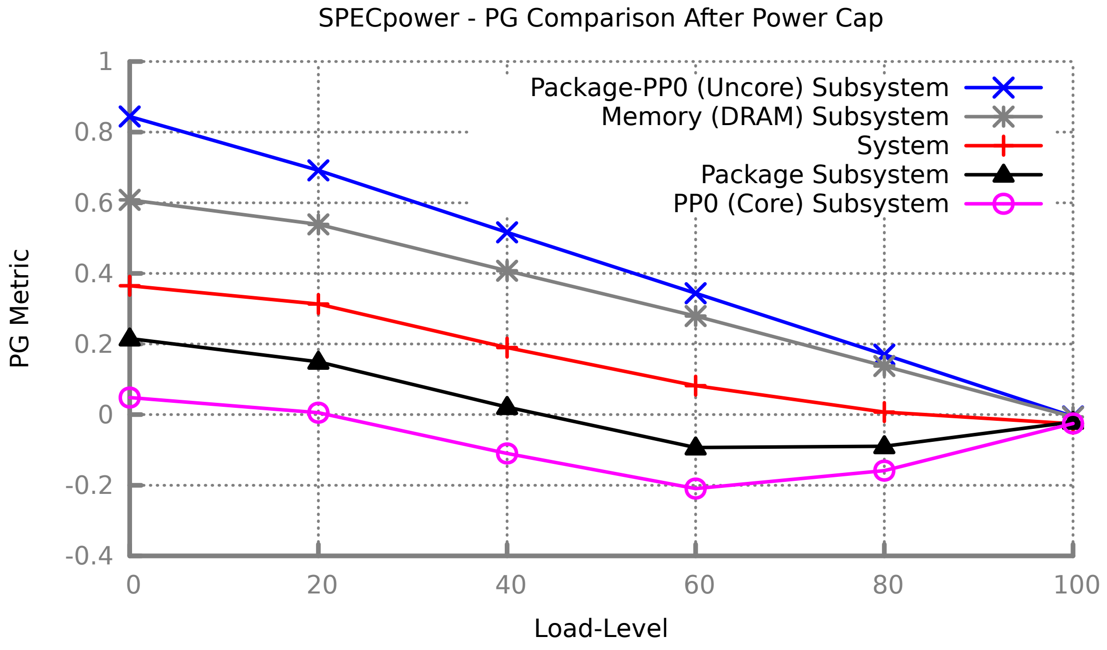
Figure 3.16: Impact of Power Limiting on SPECweb (PP0 and Memory)

Table 3.4: Summary of Full System- and Subsystem-Level Energy Proportionality and Linear Deviation Before and After Power Caps.

Subsystem	**Benchmark**	**EP Metric (Before)**	**EP Metric (After)**	**LD Metric (Before)**	**LD Metric (After)**
Full System	SPECpower	0.54	0.69	0.067	-0.044
	SPECweb	0.29	0.48	0.101	-0.024
Package (PKG)	SPECpower	0.70	0.96	0.066	-0.149
	SPECweb	0.44	0.79	0.151	-0.101
Core (PP0)	SPECpower	0.85	1.18	0.095	-0.221
	SPECweb	0.63	1.12	0.254	-0.192
Uncore (Package-PP0)	SPECpower	0.14	0.16	0.008	0.004
	SPECweb	0.02	0.03	0.022	0.019
Memory (DRAM)	SPECpower	0.36	0.37	0.017	0.013
	SPECweb	0.07	0.09	0.053	0.045

Table 3.4 shows the EP metric for the configuration that achieves the best power savings of the SPECpower and SPECweb benchmarks. We see components with an EP metric > 1, indicating power consumption that is better than energy proportional. As expected, the EP metric for the PP0 subsystem substantially. For the PP0 domain, the metric increases from 0.85 to 1.18 and 0.63 to 1.12 for the SPECpower and SPECweb benchmarks, respectively. The memory and the *uncore* subsytem do not see any substantial improvement in the EP metric.

Figure 3.17 shows the PG metric for the configuration that achieve best power savings. For the Package and PP0 subsystems, the PG metric is negative for some load-levels suggesting that we achieve better than energy-proportional operation. As observed both the memory and *uncore* subsystems are not amenable to operating at different power-performance trade-off points as the PG metric trend decreases linearly for those subsystems.



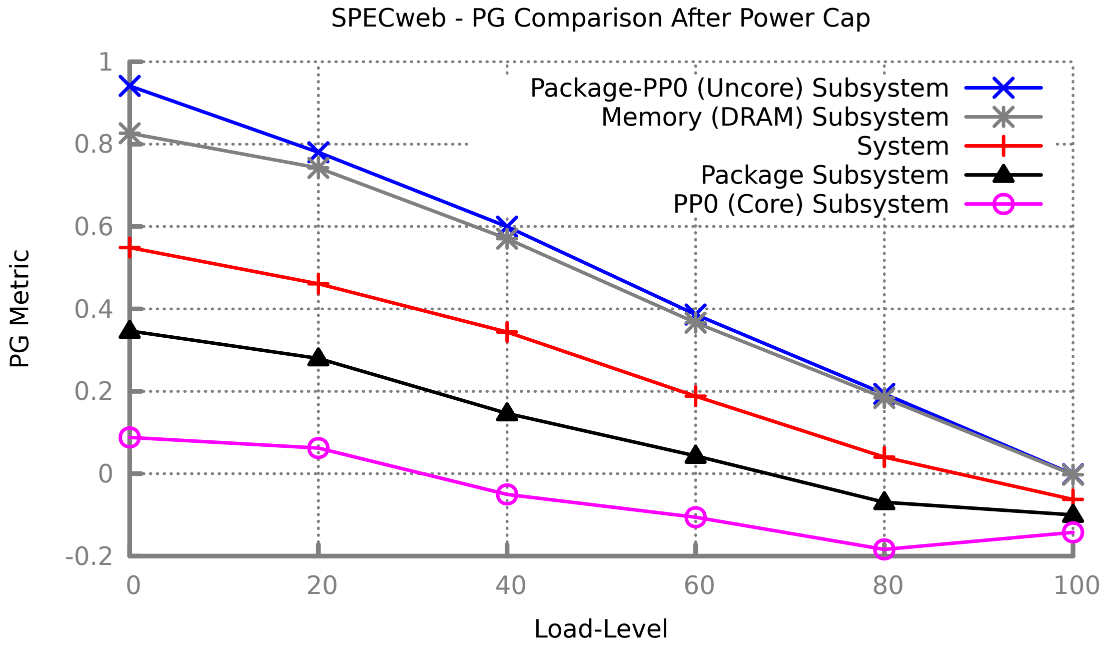


Figure 3.17: PG Metric after Power Cap

Table 3.4 also shows the LD metric for the run with the most power savings. We are able to shift the linear deviation from positive to negative for the full system, Package and PP0 subsystems. Our approach improves the energy proportionality of the server by improving the linear deviation of the subsystems.

3.3.3 Power Savings

Figure 3.18 shows the power savings for the SPECpower and SPECweb benchmarks. Power limiting conserves between 3% to 15% of power at the full system-level. We would like to stress that the subsystems over which we do not have power limiting control consume between 11% and 17% power of the full system depending upon the load-level. The power savings at 100% load-level for SPECpower is less than SPECweb as the former is a CPU-intensive benchmark and most of our power savings comes from the PP0 domain. We observe that the memory subsystem provides negligible power savings. In the case of the PP0 domain we conserve between 3% and 30% for SPECpower and 14% to 45% for SPECweb depending upon the load-level.

3.3.4 Impact on Energy Efficiency

Figure 3.19 shows the energy efficiency of SPECpower and SPECweb at five different load-levels for the CPU+Mem and vanilla runs. The energy efficiency of SPECweb and SPECpower are represented as *ssj_ops/watt* and *SUS/watt* respectively. The improvement is calculated as the ratio of difference between the energy efficiency under power limit and vanilla run over the vanilla run.

We improve energy efficiency improvements in all cases. SPECpower and SPECweb produce up to 16% and 17% energy efficiency improvement, respectively, due to power limiting.

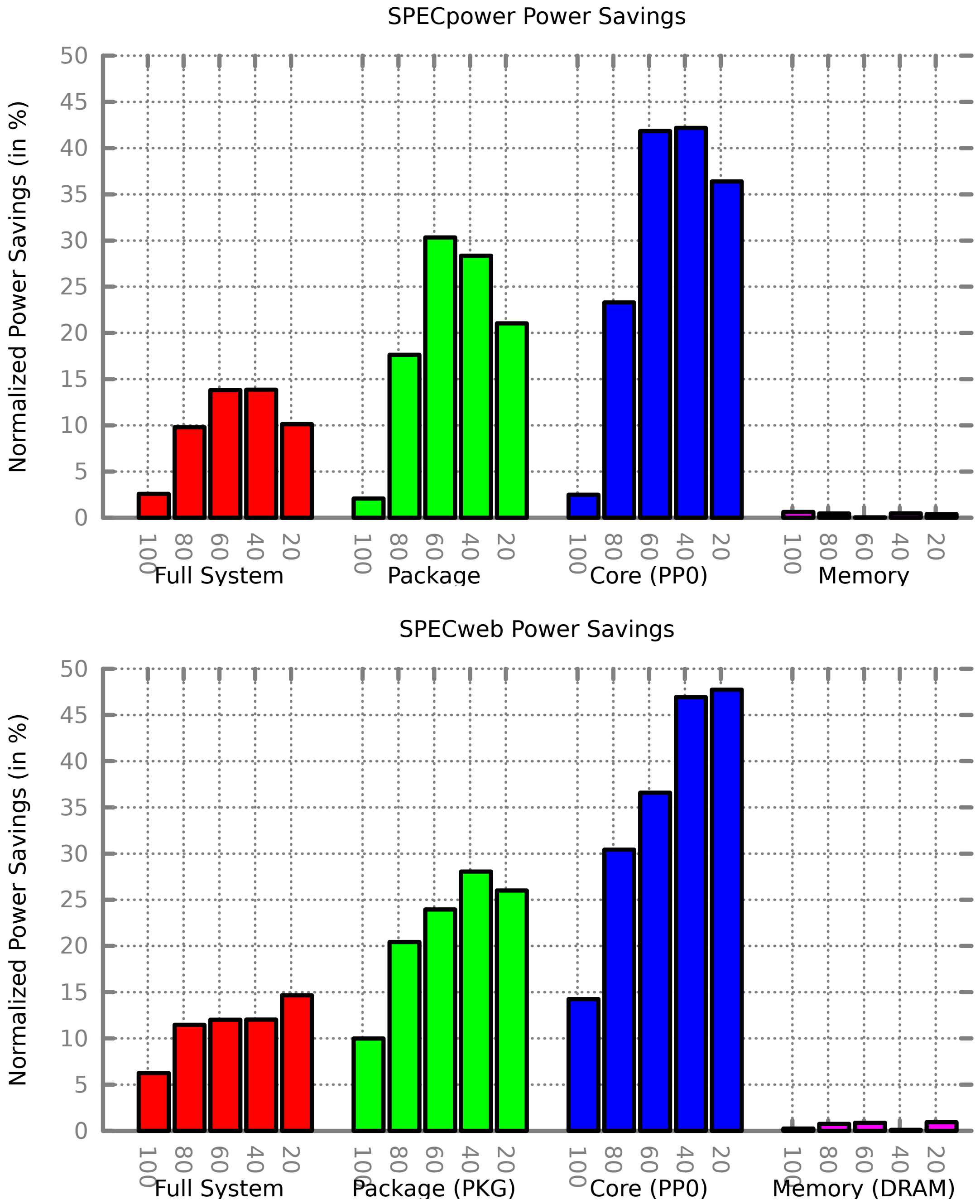


Figure 3.18: Power Savings

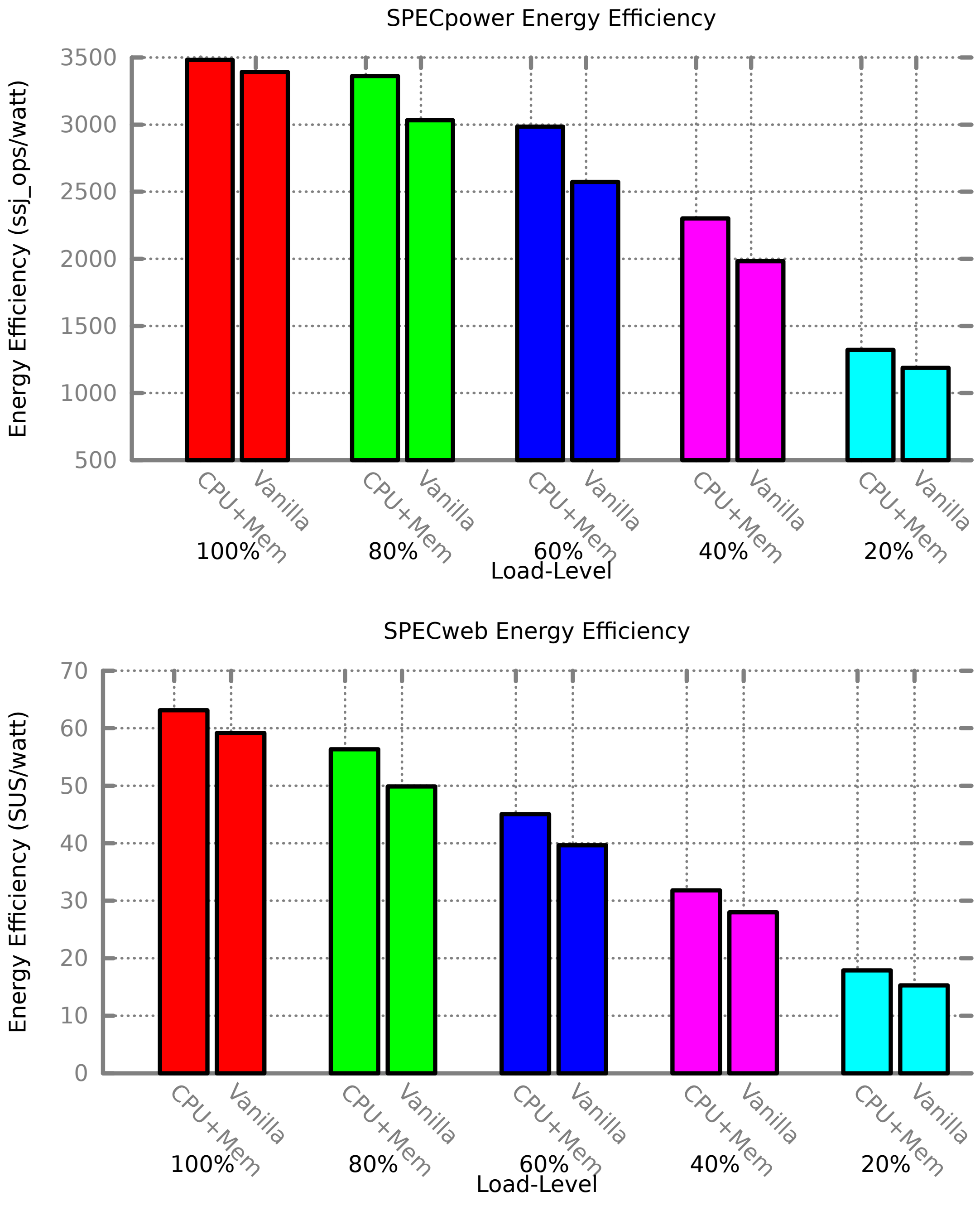


Figure 3.19: Energy Efficiency

3.3.5 Impact on Instantaneous Power Consumption

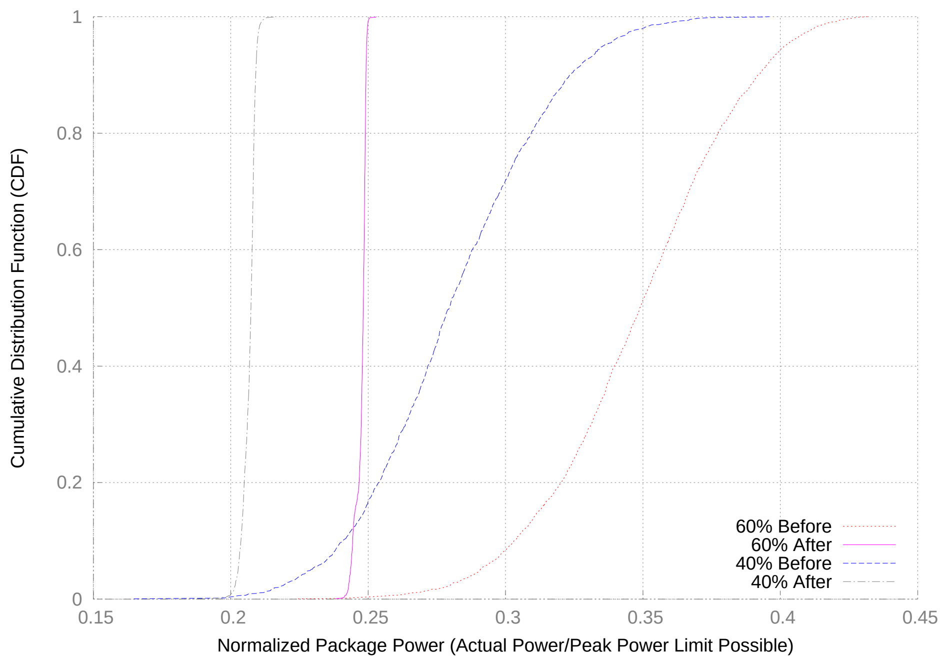


Figure 3.20: Instantaneous SPECpower Package Power Consumption (Before and After Applying Power Caps)

The maximum instantaneous power consumed by the subsystems is an important factor in determining the power budget for a system. Determining the optimal power-provisioning strategy requires an understanding of the instantaneous power profile of the system. Towards this end, the instantaneous power profile is discussed in this section. We describe the differences in instantaneous power profiles between two different load-levels (40% and 60%) and with and without power cap for both the SPECpower and SPECweb benchmarks. The instantaneous power profile for the configuration which achieved best power savings is shown. The power profile of the package and memory subsystems is collected at 50-ms resolution in all cases.

Figures 3.20 and 3.21 show the instantaneous power profile of the package subsystem at 40%

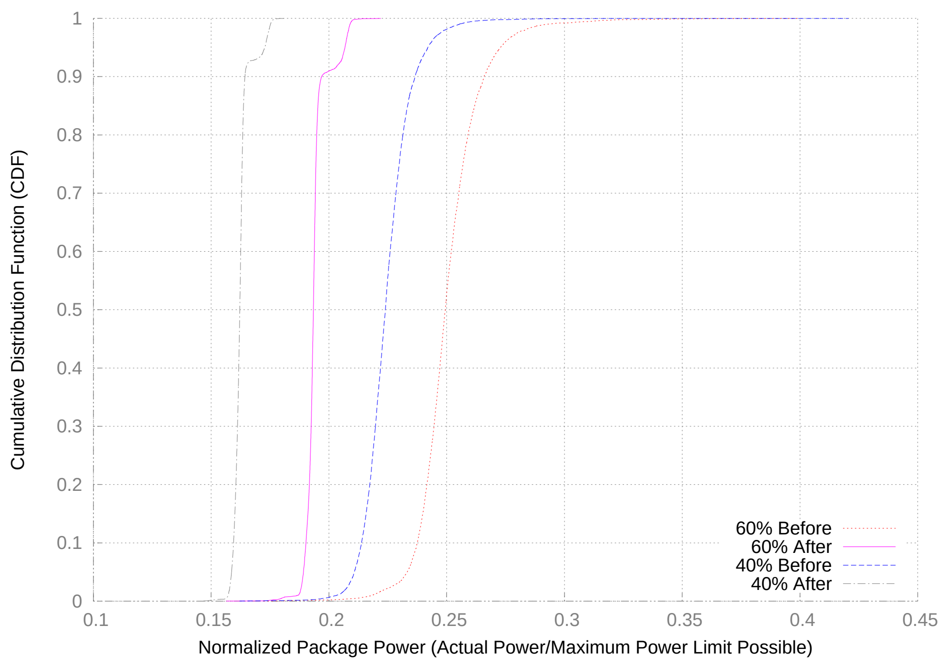


Figure 3.21: Instantaneous SPECweb Package Power Consumption (Before and After Applying Power Caps)

and 60% load-level with and without the power cap for SPECpower and SPECweb, respectively. Power limiting works as expected for both the SPECpower and SPECweb benchmarks. The range of instantaneous power consumption is narrowed due to power capping. Moreover, the power limiting removes the few power spikes indicated by the flat lines at 100% (see Section 3.2.2). Such power-limiting mechanisms are useful for power provisioning without impacting the performance of the application.

Figures 3.22 and 3.23 show the instantaneous power profile of the memory subsystem at 40% and 60% load-level with and without the power cap for both the benchmarks, respectively. The few power spikes in the memory subsystem for both the benchmarks are removed via power limiting. Even though we do not achieve considerable power savings at the memory subsystem-level due to power limiting, applying appropriate power limits such that the impact on performance is

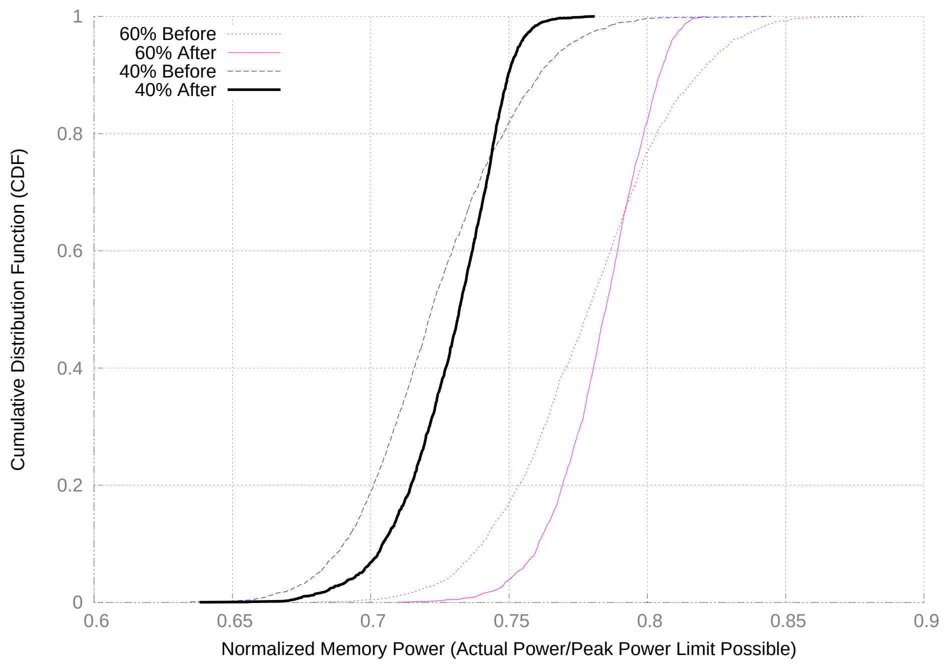


Figure 3.22: Instantaneous SPECpower Memory Power Consumption (Before and After Applying Power Caps)

controlled to the desired level can help make power-provisioning decisions easier and increase the efficiency of the server.

3.4 Modeling Performance Under Power Budget

It is unclear how subsystem-level power limits will affect the throughput and the response time of the benchmarks, particularly satisfying the constraints that are based on the 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 run-time system. In the 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.

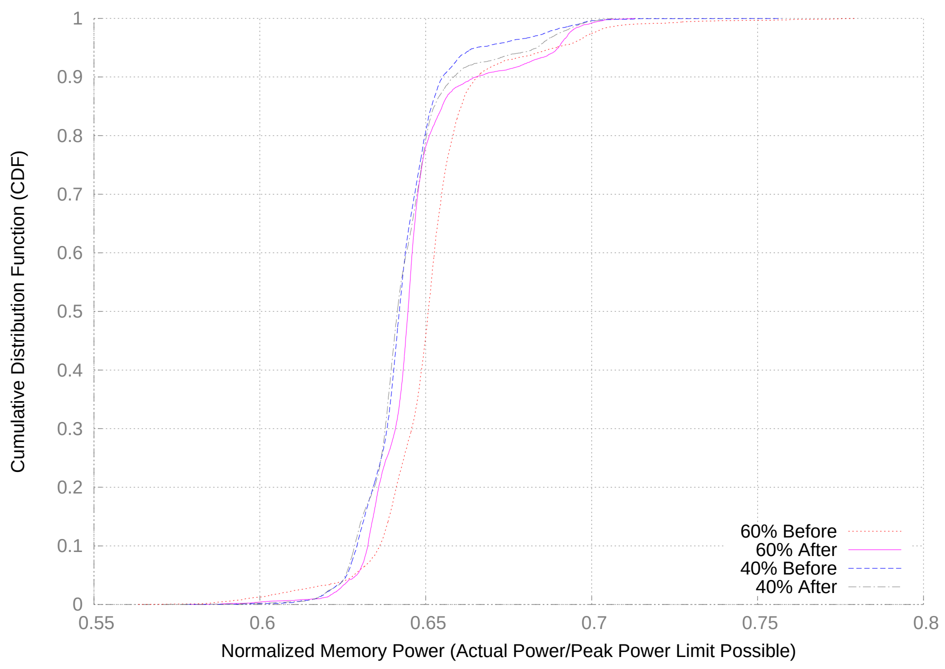


Figure 3.23: Instantaneous SPECweb Memory Power Consumption (Before and After Applying Power Caps)

In this section, we first provide an overview of non-linear models and their mathematical forms. Then we discuss our experiences in modeling the relationship between performance and subsystem-level power limits.

3.4.1 Overview of Non-linear Models

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

Through our experimental data, we determined that two non-linear forms (Gompertz's and

Power-Law model) are sufficiently capture the relationships required to design an optimization framework. Table 3.5 presents the mathematical forms of these models. We chose the most basic form of these models for simplicity. In Sections 3.4.2, we use these basic forms to capture the non-linear relationship between throughput, response times and power limits.

Table 3.5: Models and their Mathematical Forms

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

3.4.2 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 3.6: 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 3.6. 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 the 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 section because our experiments reveal that all the power savings comes from the PP0 subsystem when we limit the power consumption of the package subsystem [51]. The relationship between performance and the 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 the case of SPECweb, we use five different combinations of data sets, as shown in Table 3.6. Each of the combination corresponds to running SPECweb at load-levels between 100% and 20% in decreasing 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 [20]. Specifically, we use the nls2() package to perform non-linear regression.

The Root Mean Squared Error (RMSE) is used to determine the quality of fit of our models. By definition, the residual is the difference between the observed value and predicted value, as shown in Equation (3.4), 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 (3.5), where N is the number of predictions.

$$Residual = (Y_i - f(X_i)) \tag{3.4}$$

$$RMSE = \frac{\sqrt{\sum Residual^2}}{N} \tag{3.5}$$

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 to understand the effects of simultaneously changing the power limits on two different subsystems. Figures 3.24, 3.25, 3.26, 3.27, 3.28 and 3.29 show our non-linear models for subsystem-level power limiting of SPECpower and SPECweb. In each figure, the X-axis represents the predictor variable and the Y-axis represents the predicted variable. The figures also show how we use the data set in Table 3.6 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 3.24. 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 PPO and DRAM power limit ratio. These bounds were chosen as they are sufficient to create our optimization framework. Each of the terms is modeled separately to keep the basic forms of the models used for non-linear regression simple.

For illustration, consider Figure 3.24. The figure shows the model for PP0 power limit bound of the SPECpower benchmark. The Y-axis represents load-level achieved (predicted variable) and the X-axis represents the PP0 power limit ratio (predictor variable). The points in the figure are the training data set (PP0 power limit ratio) and the curve represents the model. We use Gompertz model with the basic form shown in Table 3.5 to derive the parameters for the model. The values for each parameter of this model is summarized in Table 3.7. In Figure 3.25, we show the model for DRAM power limit bound of the SPECpower benchmark. In this case, the Y-axis is the load-level achieved (predicted variable) and the X-axis represents the DRAM power limit ratio (predictor variable). Similar to Figure 3.24, the points represent the training data set and the curve represents the model. We use power-law model with the basic form shown in Table 3.5 to derive the model shown in the figure. The derived parameters are shown in Table 3.7. We show the model for the interaction term of the SPECpower benchmark in Figure 3.26. The Y-axis represents the load-level achieved (predicted variable) and the X-axis represents the interaction term (predictor variable). We use the Gompertz model in this case and the parameters derived for the model is shown in Table 3.7. Similarly, Figures 3.27, 3.28 and 3.29 show the training data set (points), models (curves), predicted variable (Y-axis) and the predictor variable (X-axis) for the PP0 power limit ratio, DRAM power limit ratio and the interaction term, respectively, for the SPECweb benchmark. Table 3.7 summarizes our models, their parameters, and quality of fit for both the benchmarks.

The models described in this section will be used as upper-boundary conditions to determine subsystem-level power limits to minimize the power consumption given performance constraints using an optimization framework in Section 3.5.2.

Table 3.7: Models and their Parameters for Performance Modeling Under Power Limit. See Table 3.5 for Basic Forms of the Models. RMSE = Root-Mean-Square Error. Lower RMSE is Better.

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

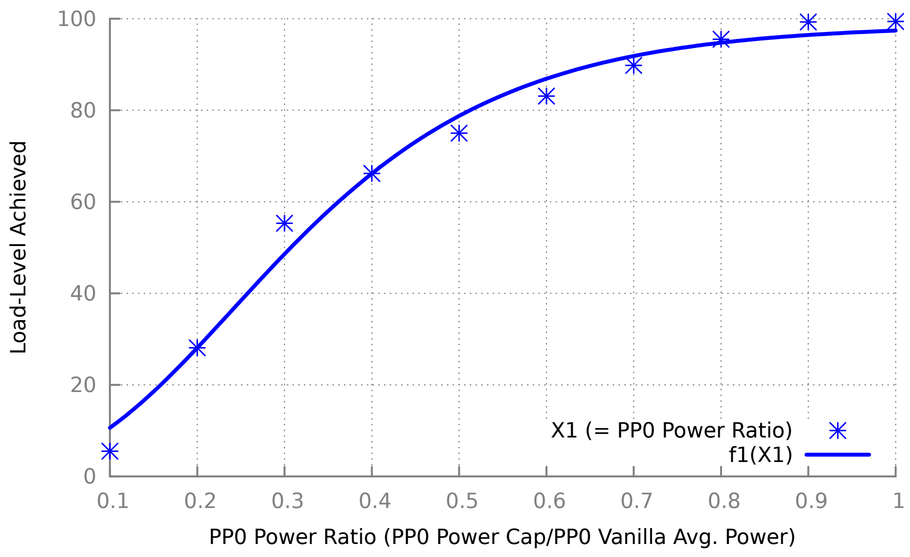


Figure 3.24: Models for SPECpower – PP0 Power Limit Bounds. RMSE = 1.08.

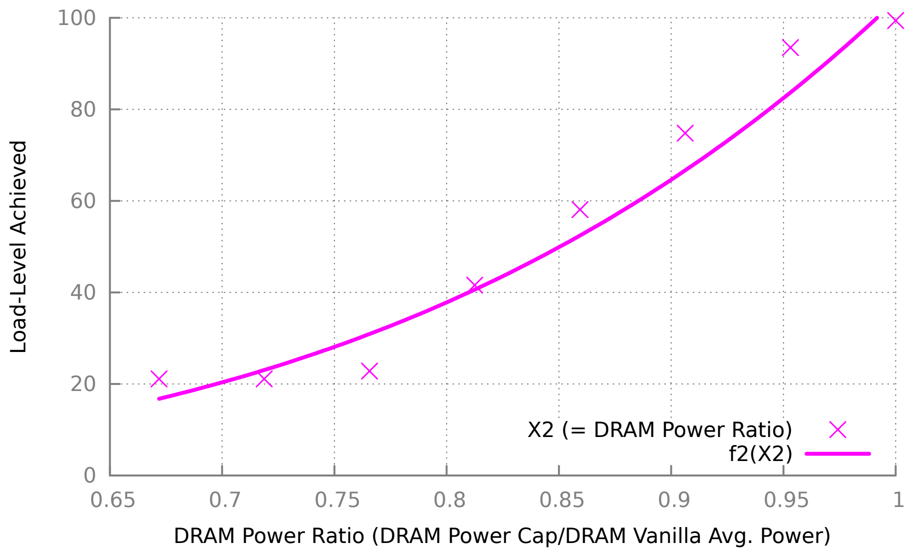


Figure 3.25: Models for SPECpower – DRAM Power Limit Bounds. RMSE = 1.41.

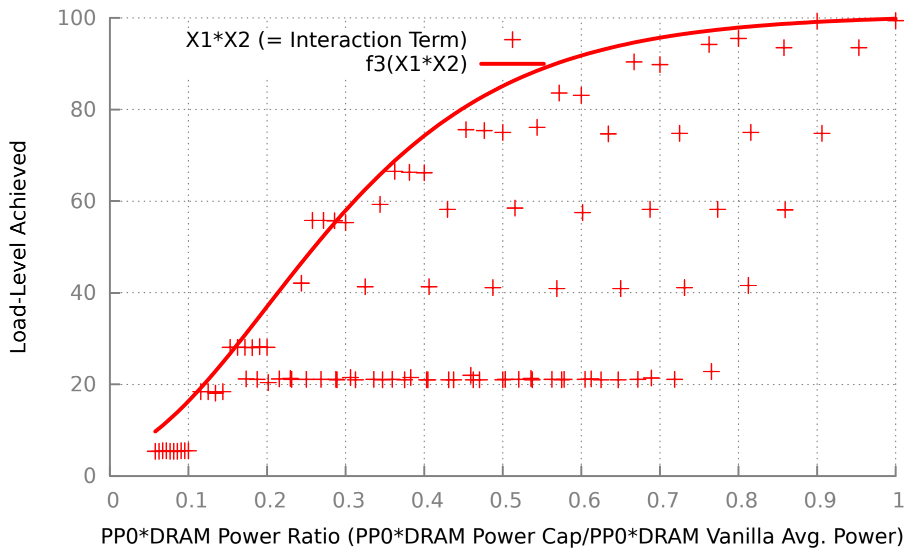


Figure 3.26: Models for SPECpower – Interaction Term Bounds. RMSE = 1.75.

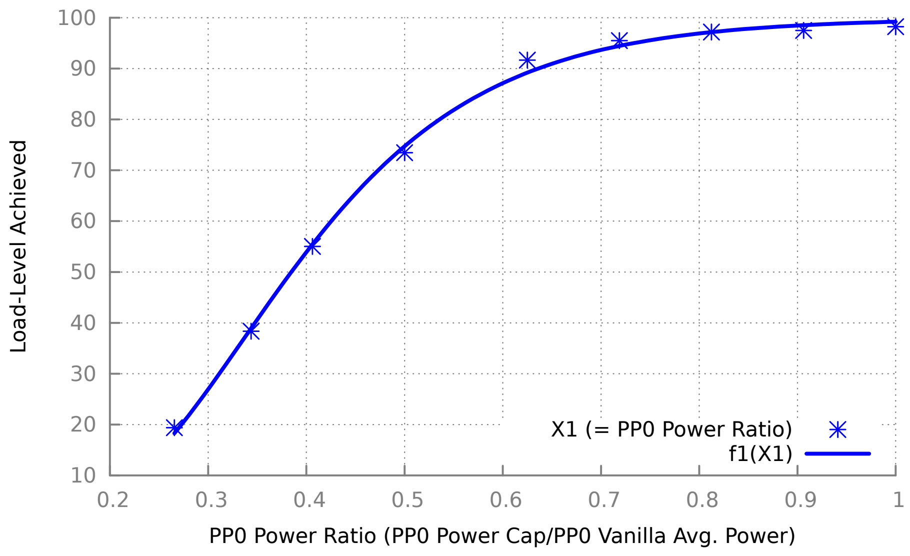


Figure 3.27: Models for SPECweb – PP0 Power Limit Bounds. RMSE = 0.39.

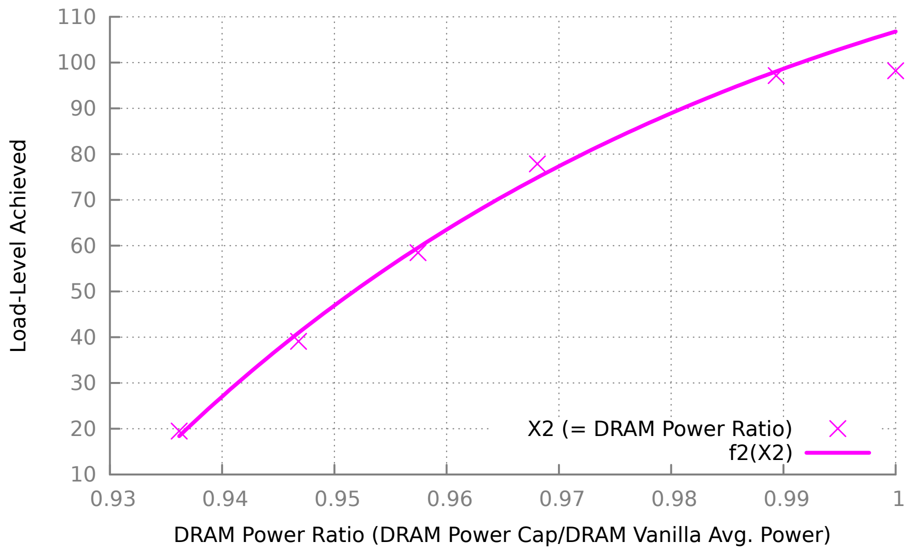


Figure 3.28: Models for SPECweb – DRAM Power Limit Bounds. RMSE = 1.05.

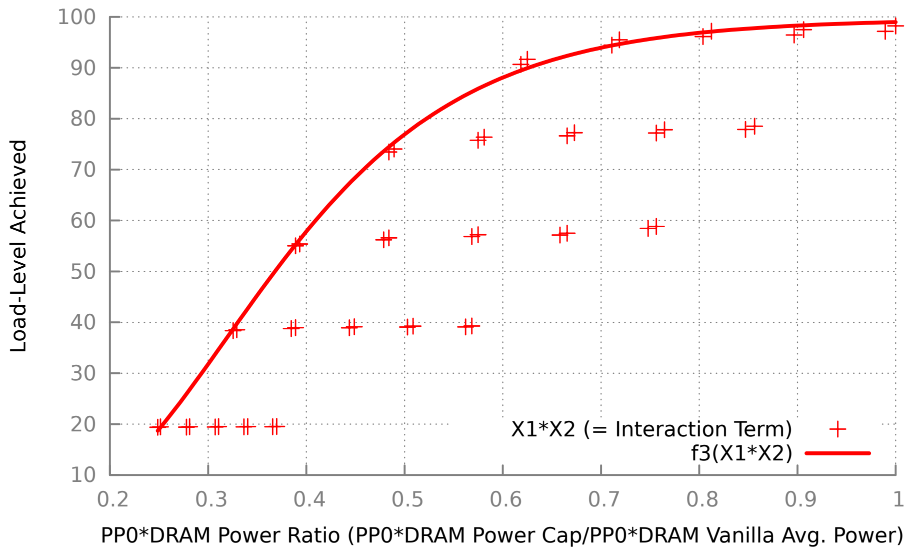


Figure 3.29: Models for SPECweb – Interaction Term Bounds. RMSE = 0.28.

3.5 Runtime System for Power Management

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

3.5.1 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 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 [26, 30, 41]. The relationship between LLCM/S and the load-level is modeled like the models presented in Section 3.4. Figure 3.30 shows our models. We use linear regression[14] for modeling the relationship in the case of SPECpower and non-linear regression (Power-Law model) in the 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 3.8.

Table 3.8: Models for Load-Level Detection

Benchmark	**Model Name**	α	β	γ	**RMSE**
SPECpower	Linear	103.69	7.63	-	0.45
SPECweb	Power Law	101.42	2.51	5.69	0.33

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

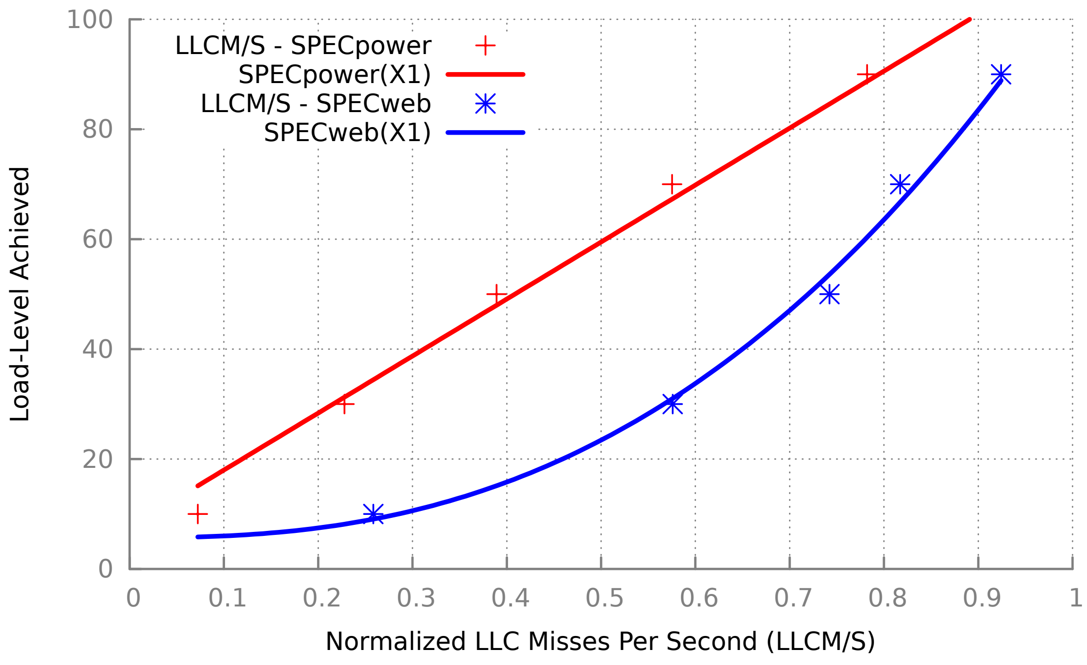


Figure 3.30: Models for Load-Level Detection

3.5.2 The Optimization Framework

The models proposed in Section 3.4 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 (3.6) 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 Y). As already discussed, we model only the boundaries. As a consequence, we have inequalities in Equation (3.6). $f1(X1)$, $f2(X2)$, and $f3(X1 * X2)$ from Figures 3.24, 3.25 and

3.26, respectively, are used as upper-bound constraints on performance for the SPECpower benchmark. $f1(X1)$ from Figure 3.27, $f2(X2)$ from Figure 3.28 and $f3(X1 * X2)$ from Figure 3.29 are used as upper-boundary conditions on performance for the SPECweb benchmark.

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.

$$
\begin{aligned}
&Minimize : X1 + X2 \\
&Subject\, to : \\
&\qquad f1(X1) \geq Y \\
&\qquad f2(X2) \geq Y \\
&\qquad f3(X1 * X2) \geq Y \\
&\quad Var : PP0R_{LB} \leq X1 \leq PP0R_{UB} \\
&\quad Var : DRAMR_{LB} \leq X2 \leq DRAMR_{UB}
\end{aligned}
\tag{3.6}
$$

3.5.3 Runtime Algorithms

We propose runtime power management schemes that leverage the load-level detection model and the optimization framework described in Sections 3.5.1 and 3.5.2, 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 need 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 to address 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 3.31. The power-management scheme monitors the LLC misses for first N seconds and places appropriate power limits on the subsystems based on offline models.

```
Input: Workload --> W1.
Launch workload W1.
Calculate LLCM/S over first N seconds.
Determine the load-level using the appropriate load-level detection model shown in Section
 3.5.1.
Set appropriate PP0 and DRAM power limits on all sockets using the optimization framework shown
in Section 3.5.2.
```

Figure 3.31: Algorithm Without Workload Consolidation

The algorithm with workload consolidation, used for second case, is described in Figure 3.32. In the second scheme, we consolidate the workload onto a single socket and memory node and launch another workload on 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 [31]. As we will show in Section 3.6, deploying mixed workloads can help improve the energy proportionality of the system.

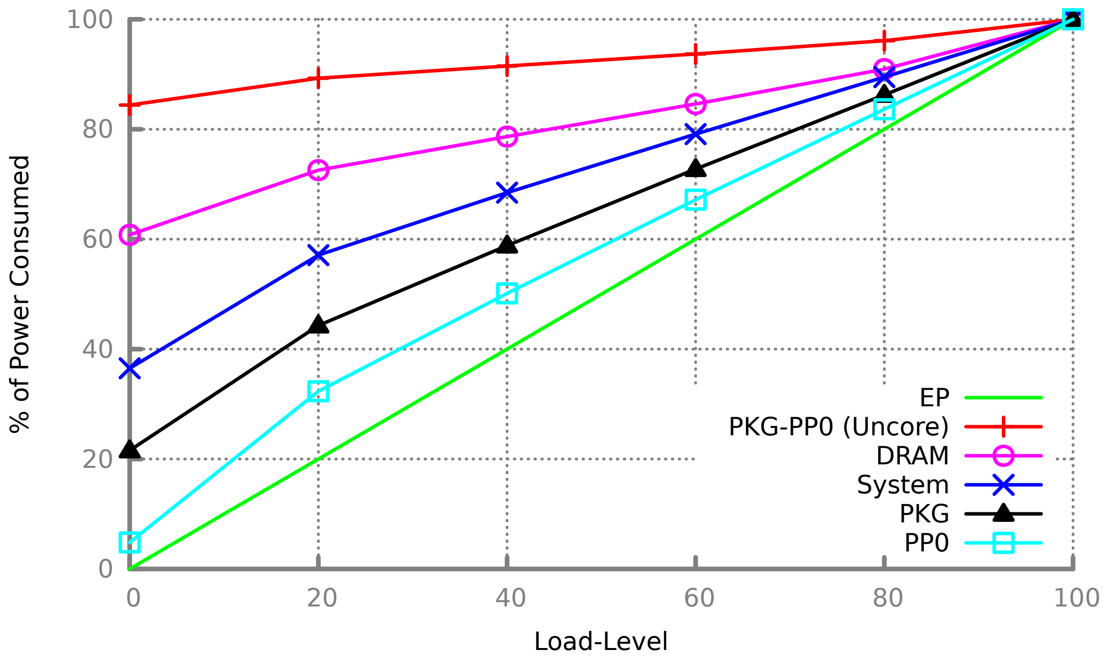
```
Input: Workloads --> W1 and W2
Lines 2 through 4 in Figure 3.31.
if load-level < 50%
then
   Consolidate the workload W1 on to a single socket and memory node.
   Launch workload W2 and confine it to the second socket and memory node.
   for each socket
   do
      Calculate LLCM/S over first N seconds.
      Determine the load-level using the appropriate load-level detection models shown in
      Section 3.5.1.
      Set appropriate PP0 and DRAM power limits on the socket using the optimization framework
      shown in Section 3.5.2.
   done
endif
```

Figure 3.32: Algorithm With Workload Consolidation

3.6 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 the algorithm shown in Figure 3.31), SPECweb without consolidation (SW) (using the algorithm shown in Figure 3.31) and SPECpower+web with consolidation (SPW[15]) (using the algorithm shown in Figure 3.32). 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 3.1.3.

[15]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).



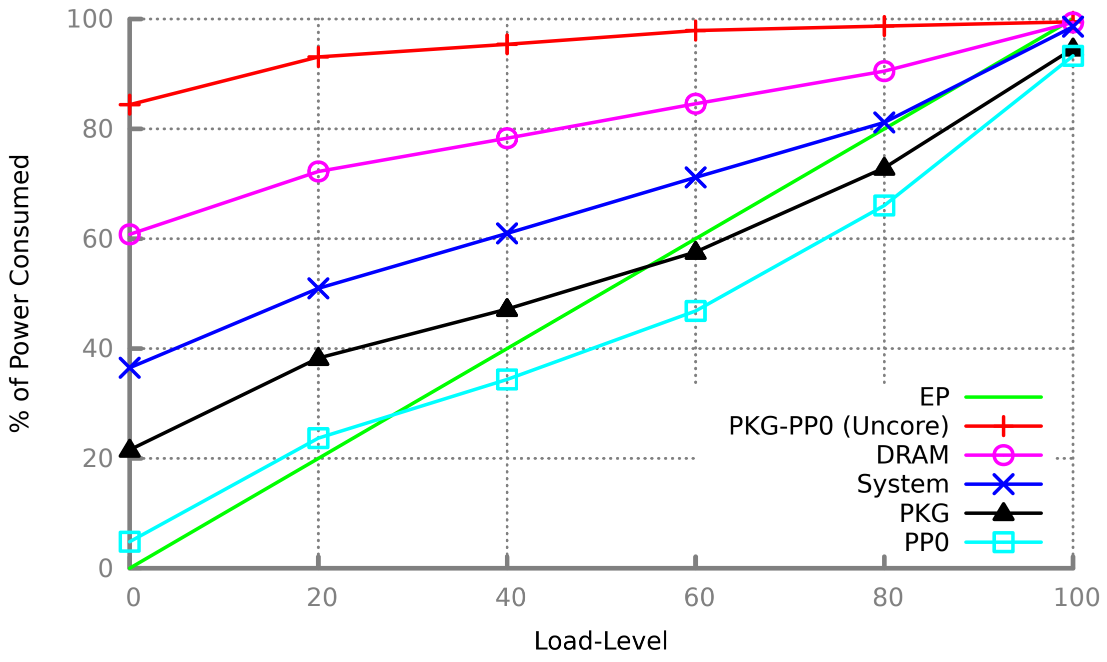
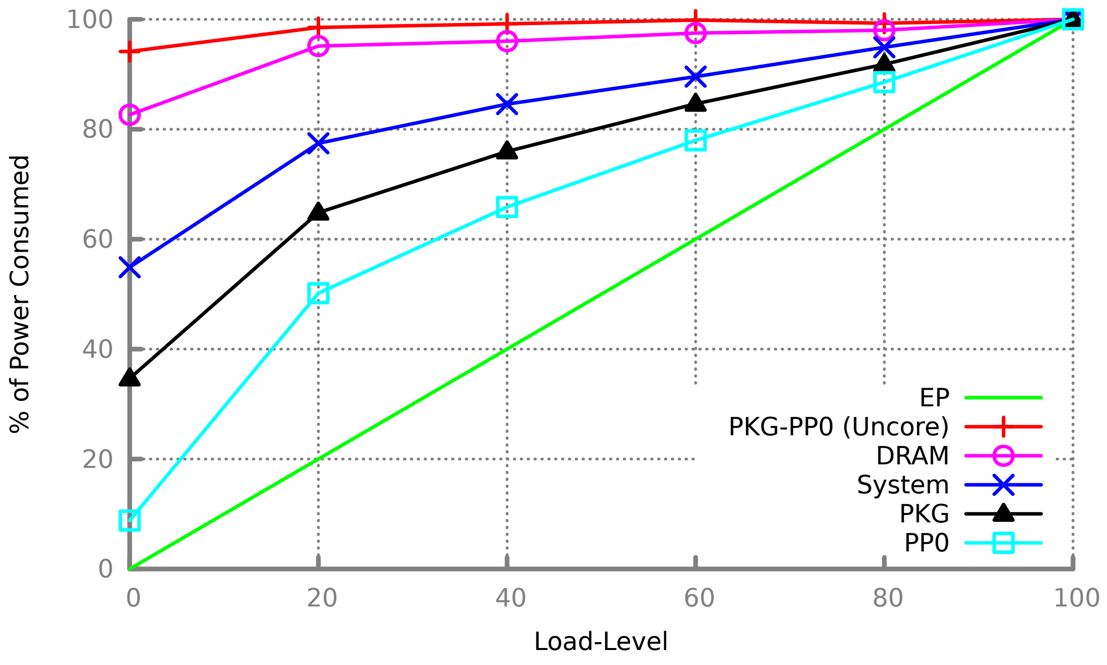


Figure 3.33: Energy Proportionality – Top: SP, Bottom: SP With Runtime System



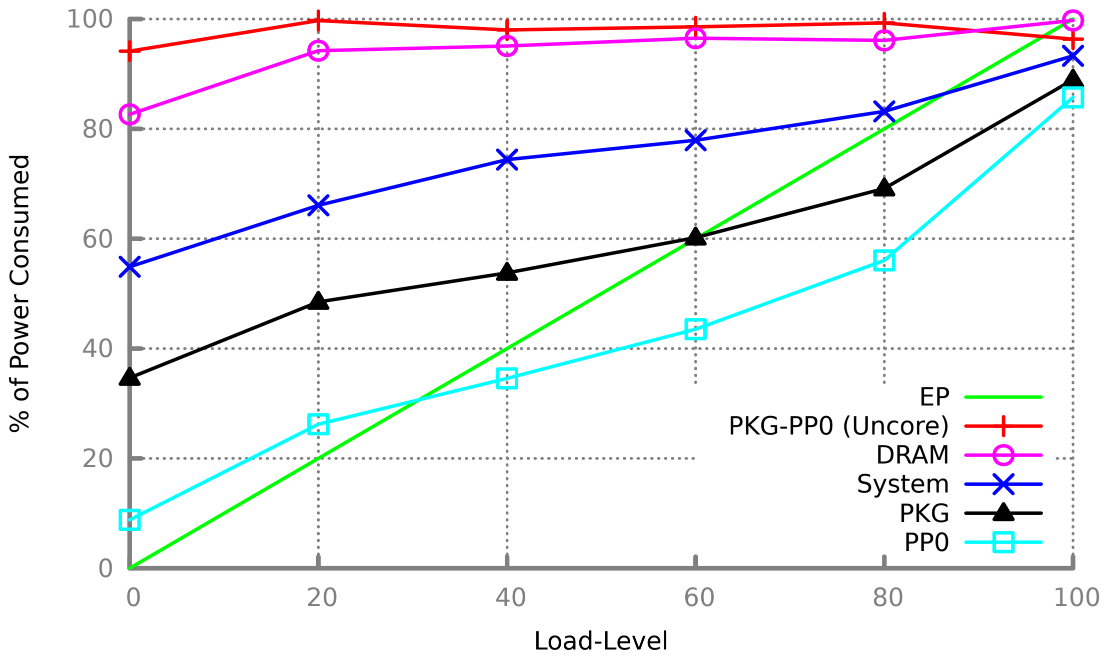
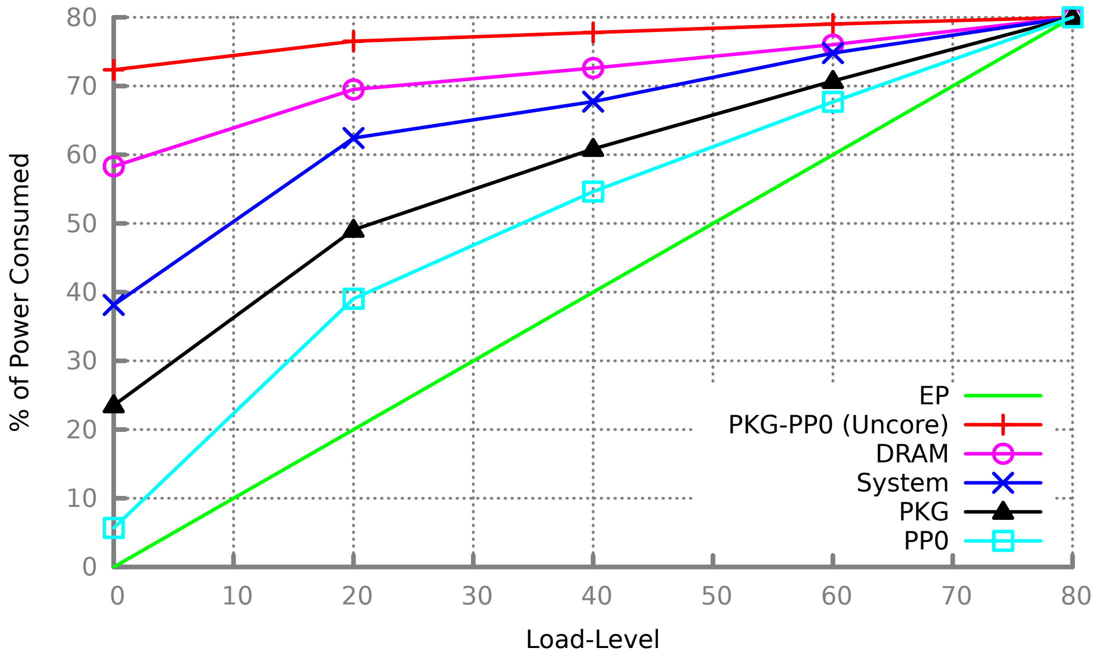


Figure 3.34: Energy Proportionality – Top: SW, Bottom: SW With Runtime System

3.6.1 Energy Proportionality

Here we analyze the energy proportionality of the system. The deviation of the power curve of the system from the EP power curve is of particular interest to us. To illustrate with an example,



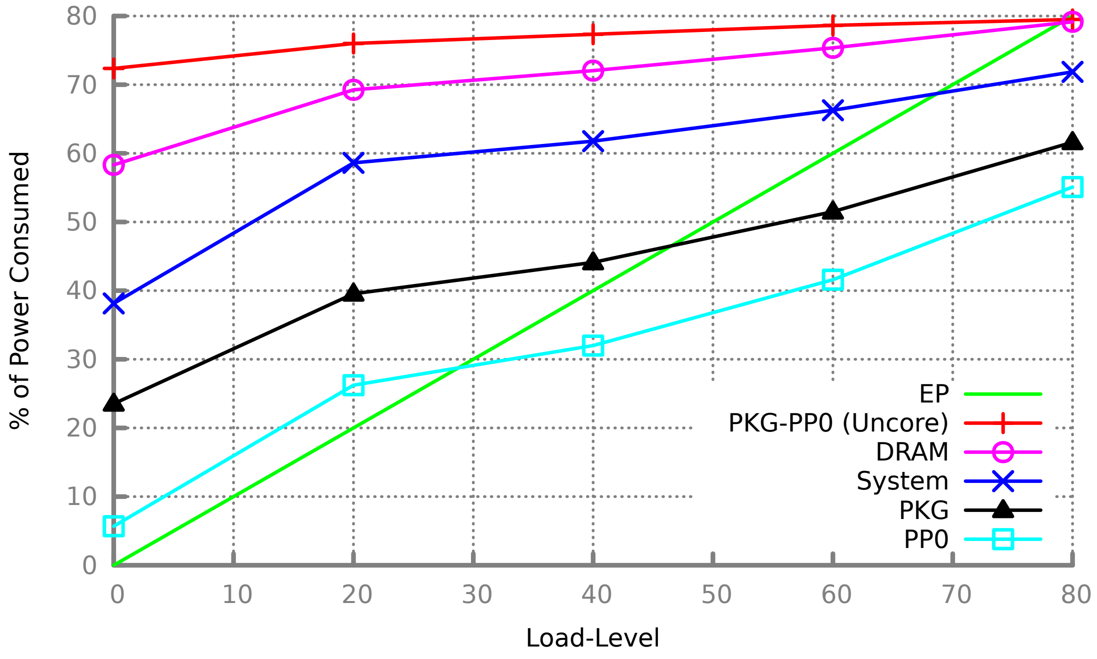


Figure 3.35: Energy Proportionality – Top: SPW, Bottom: SPW With Runtime System

Figure 1.2 shows the average power consumed by the testbed running SPECpower benchmark at each load-level, the hypothetical linear power trend and the hypothetical energy-proportional power trend for the system. We would like the area between the system and the EP power trend to be as small as possible.

Figures 3.33, 3.34 and 3.35 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[16] consumed by the system or subsystem and X-axis represents the load-level. The EP case (green line in color and gray line in black and white) 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) [9]. 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 system consumes 120 watts when idling which is 36.51% and 54.88% of peak power for SPECpower and SPECweb, respectively. However, there is improvement in energy proportionality even at low load-levels. The 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 energy-proportional power consumption reduces as the load-levels decrease.

We quantify the EPG using the EP metric [47]. A value of one for the metric represents

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

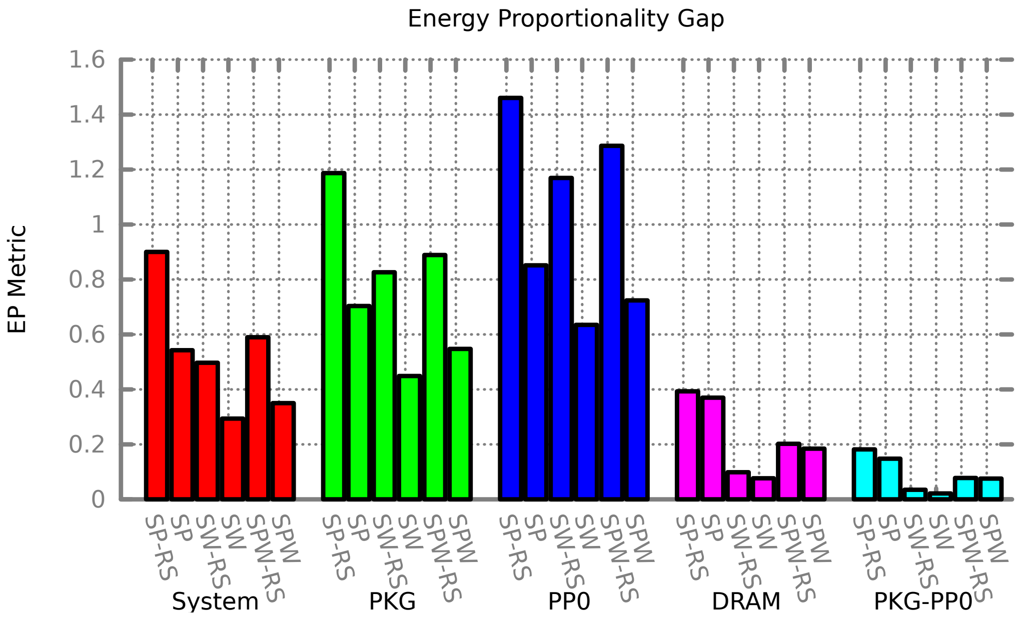


Figure 3.36: EP Metric (RS - With Runtime System)

an energy-proportional system. A value of zero represents a system that consumes a constant amount of power, irrespective of the load-level. A value greater than one represents a system which is better than energy-proportional. Figure 3.36 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 the case of PP0 subsystem, we achieve an 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 the 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 [59]. For an linear energy-proportional system, the LD metric will be 0. An LD metric > 0 and < 0 indicate superlinear and sublinear energy proportional systems, respectively. Figure 3.37 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 the workload cases. In the 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.

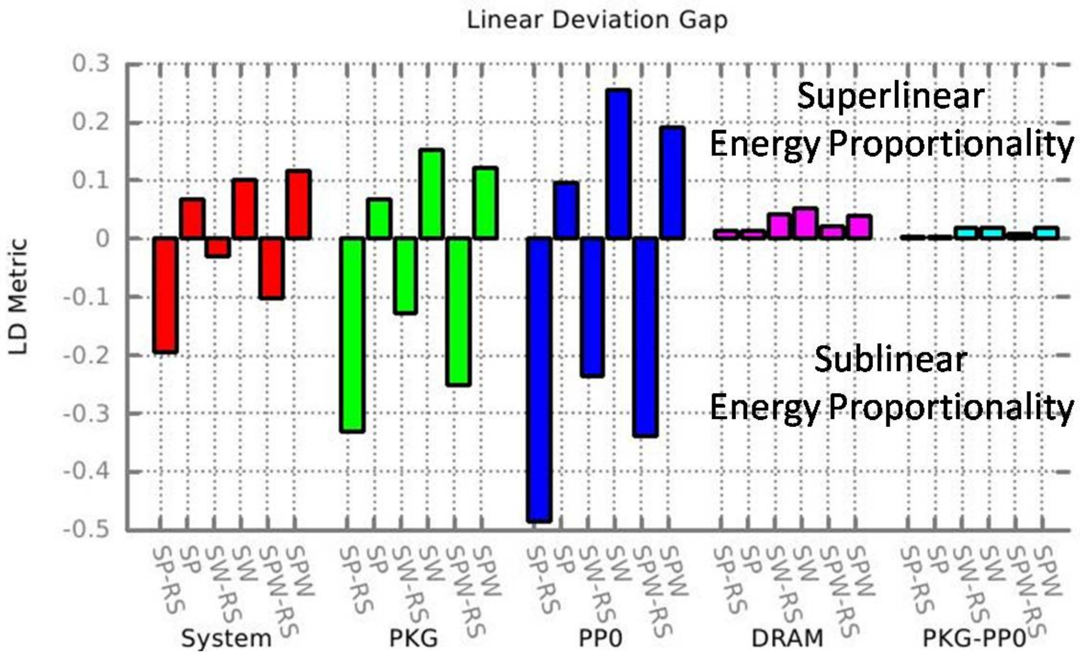


Figure 3.37: LD Metric (RS - With Runtime System)

3.6.2 Power Savings

Here we report the power-saving results for the three different workload cases. Figure 3.38 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 up to 32%, 48% and 42% of PP0 power for SP, SW and SPW, respectively. The rest of the subsystem (including DRAM) provided little to no 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 the DRAM subsystem, has advantages with respect to improving the instantaneous power profile of the server.

3.6.3 Instantaneous Power Profile

Power provisioning is directly dependent on the instantaneous power consumption of the servers. Figure 3.39 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 [8, 51].

In each workload case, our runtime system narrows the range of instantaneous power used by the server. A 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 can 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.

3.7 Comparison with Related Work

In this section, we compare the work presented in this chapter with Pegasus [39] and KnightShift [59, 60], as shown in Table 3.9. As mentioned in Section 2.1, this chapter complements [39] by providing details of the energy proportionality at the subsystem-level and how we can prevent power spikes in servers using RAPL power limiting. When compared to [59], the results in this chapter are applicable to commodity servers, one that does not require any additional hardware setup. We study the effects of RAPL power limiting on the performance, energy proportionality, and power efficiency of enterprise workloads at server-level. We also discuss the energy proportionality of different subsystems.

Table 3.9: Comparison with Related Work

Features	Chapter 3	Lo et al. [39]	Wong et al. [59, 60]
Real System	✓	✓	×
Commodity Infrastructure	✓	✓	×
Cluster Analysis	×	✓	✓
System-Level Energy Proportionality Analysis	✓	✓	✓
Subsystem-Level Energy Proportionality Analysis	✓	×	×
Subsystem-Level Instantaneous Power Analysis	✓	×	×
Runtime System	✓	✓	✓
Effects of the Runtime System on System-Level Energy Proportionality	✓	✓	✓
Effects of the Runtime System on Subsystem-Level Energy Proportionality	✓	×	×
Effects of the Runtime System on Instantaneous Power	✓	×	×
Production Application User Trace	×	✓	✓

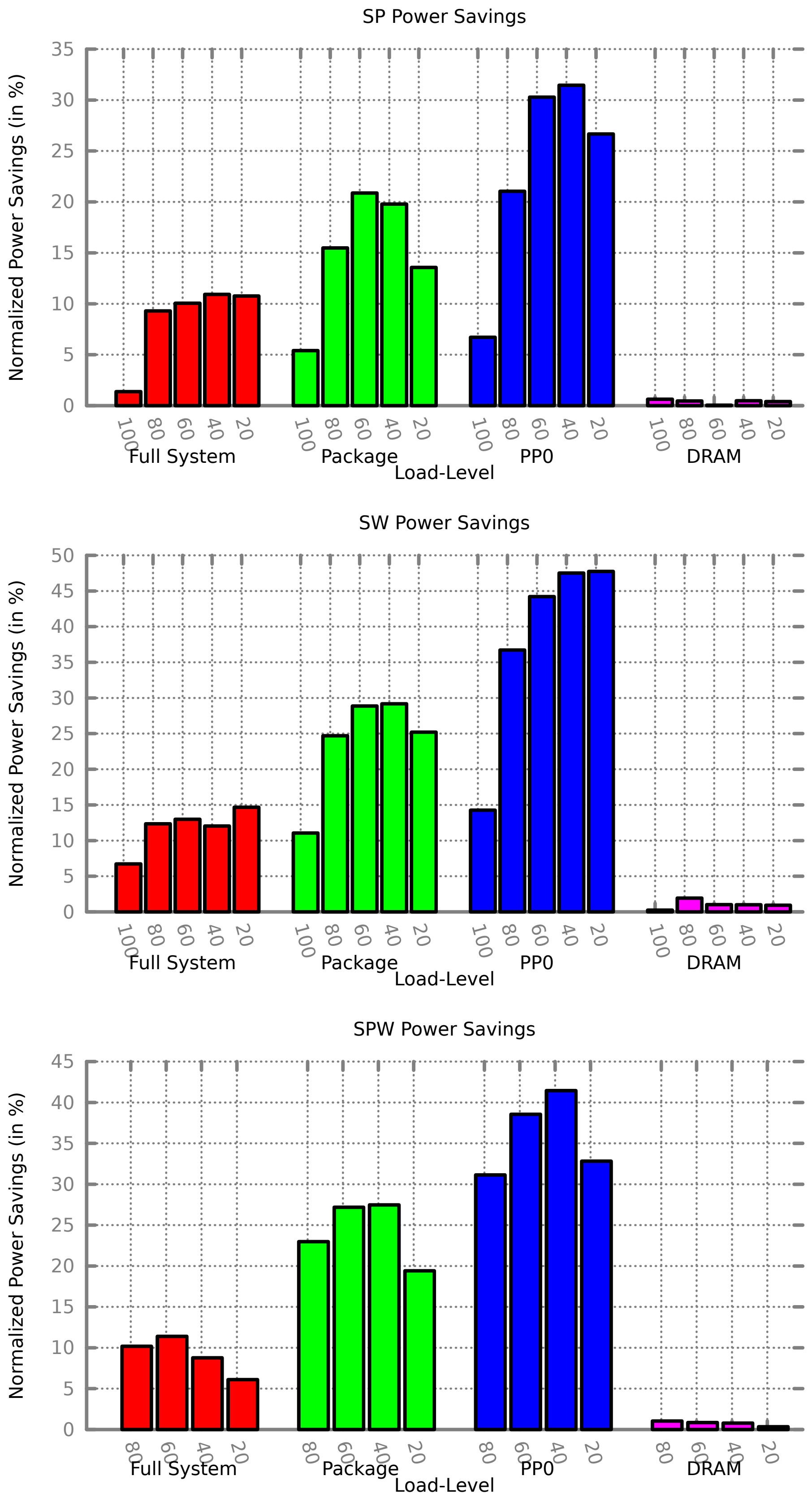
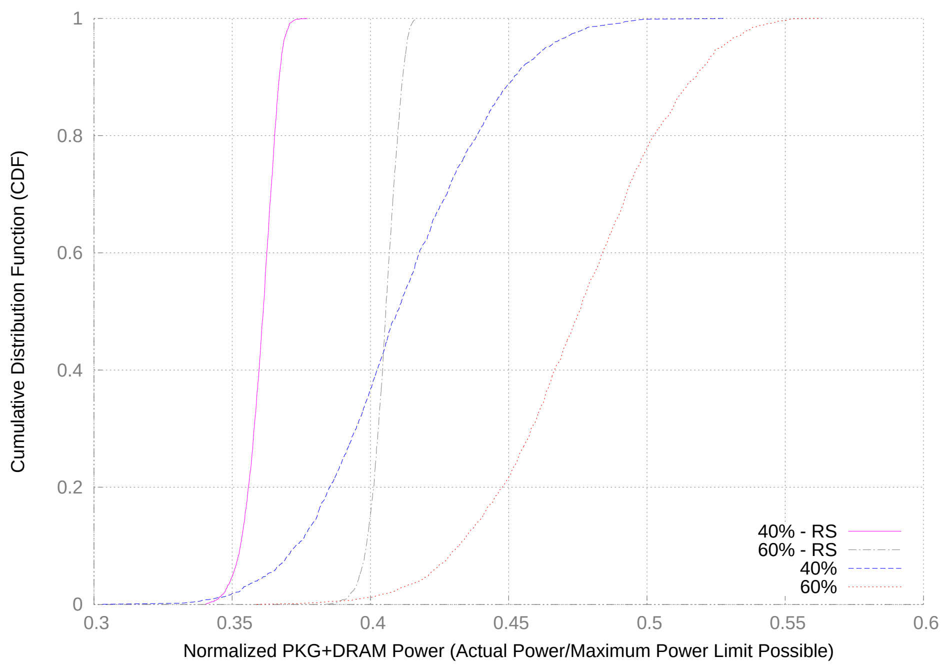
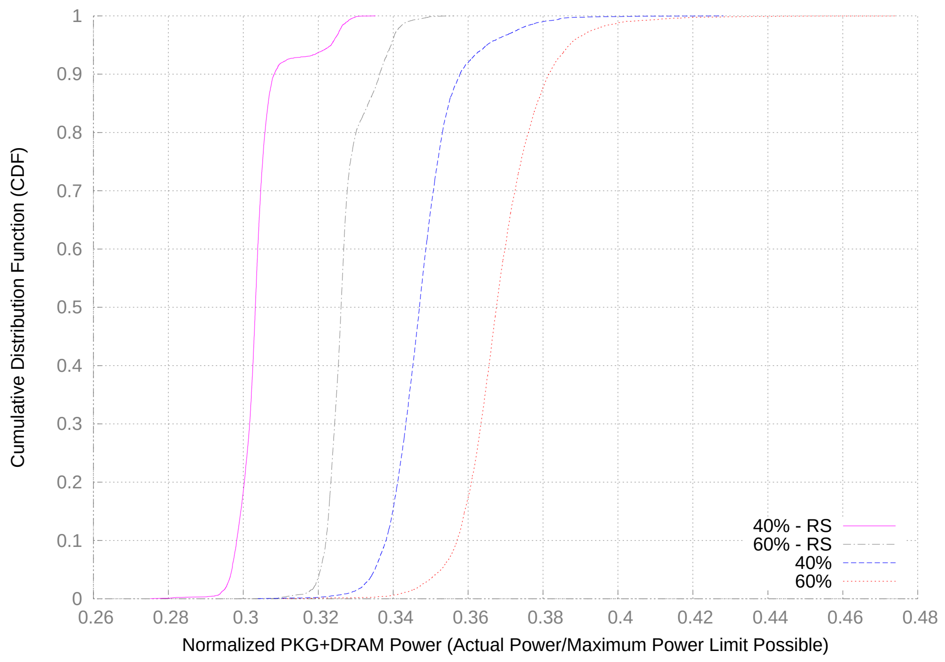
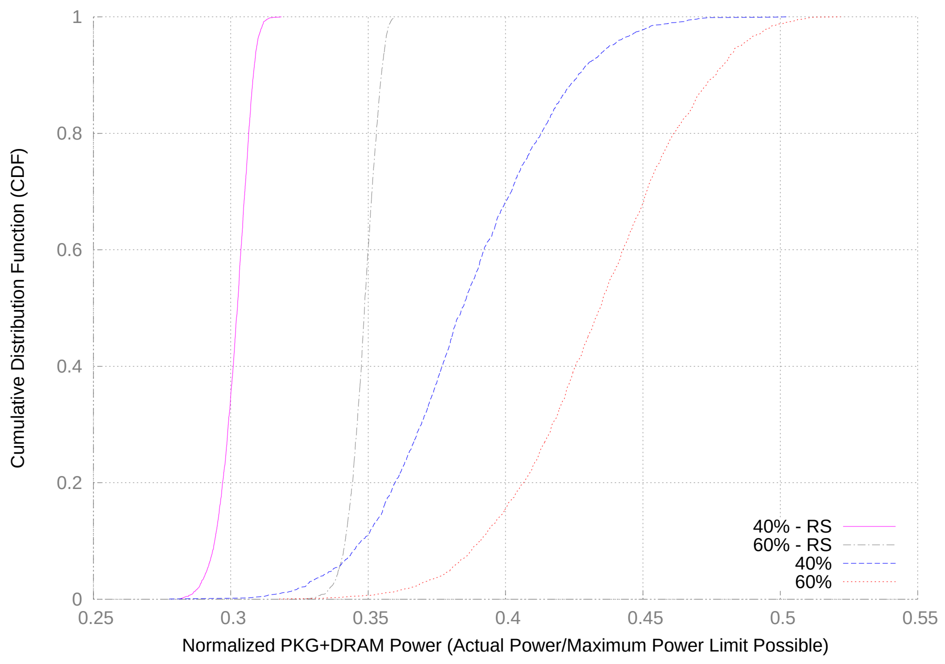


Figure 3.38: Power Savings Achieved Due to the Runtime System – Top: SP, Middle: SW and Bottom: SPW


Cumulative Distribution Function (CDF)
Normalized PKG+DRAM Power (Actual Power/Maximum Power Limit Possible)
40% - RS
60% - RS
40%
60%
0
0.2
0.4
0.6
0.8
1
0.3
0.35
0.4
0.45
0.5
0.55
0.6


Cumulative Distribution Function (CDF)
Normalized PKG+DRAM Power (Actual Power/Maximum Power Limit Possible)
40% - RS
60% - RS
40%
60%
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.26
0.28
0.3
0.32
0.34
0.36
0.38
0.4
0.42
0.44
0.46
0.48


Cumulative Distribution Function (CDF)
Normalized PKG+DRAM Power (Actual Power/Maximum Power Limit Possible)
40% - RS
60% - RS
40%
60%
0
0.2
0.4
0.6
0.8
1
0.25
0.3
0.35
0.4
0.45
0.5
0.55

Chapter 4

Analysis of Cluster-Level Energy Proportionality

We analyze the effectiveness of different software-controlled (hardware-enforced) power management techniques, such as *power and resource provisioning*, to improve the energy proportionality of scale-out workloads in this chapter. Using a data serving, web search and data caching workload, we make the following contributions [53, 54]:

- *A study of the power consumption including power measurements of individual components within the cluster.* We find that power consumption is still dominated by the processor. For the scale-out workloads under consideration, our results show that the processor consumes 45-70% of the system power depending upon the load-level.
- *An analysis of the CPU utilization of a cluster executing scale-out workloads.* We show that

the potential to save power decreases with increase in time resolution and load-level and there is ample opportunity to save power even at sub-millisecond granularity.

- *An investigation into the energy proportionality improvements, the associated performance costs (in terms of response time/latency) and power savings achieved using different power provisioning techniques.* Our results show that energy-proportional operation is not possible under all load-levels. However, improvements to the energy proportionality are possible. We also show that idle and active low-power together provide the best energy proportionality. We save up to 47% and 77% of power at the system- and processor-level, respectively.

- *An analysis of the power-performance trade-off space exposed by the use of active low-power modes for scale-out workloads.* We show that by creating a power-performance trade-off space, active low-power modes give us an opportunity to operate a given workload in a power saving configuration while meeting strict Service-Level Objective (SLO).

- *A comparison between power and resource provisioning techniques including the analysis of the associated performance achieved.* We expose the trade-off in power and resource provisioning. We show that using resource provisioning at low load-levels provides the best energy proportionality as idle power becomes a large portion of the cluster power consumption. However, the best power saving gradually shifts from resources provisioning (at low load-levels) to power provisioning (at high load-levels).

4.1 Background

In this section, we present the following background information to provide context for our work: (1) a description of the scale-out workloads under investigation, (2) the experimental setup including the cluster and workload configuration, (3) the power management interface and (4) a description of the power management techniques used in this thesis.

4.1.1 Scale-Out Workloads

Here we describe the scale-out workloads under investigation.

Data Serving

NoSQL data stores are popularly used as a data-serving application, particularly to handle the vast amount of data produced in large-scale web applications. These data stores provide fast and scalable storage with unconventional storage schemas. The entire set of data is partitioned and stored in many different servers. A key-value store is used to respond to queries from clients. A middleware layer handles the aggregation of the data required for a single client query and the servers respond to the middleware independently.

Web Search

A typical search engine maintains search indexes that are distributed among several compute nodes (or index-serving nodes) with each node containing a part of the index from the Internet. The index-serving nodes are responsible for processing requests on its own part of the search index. A master node receives requests from a client, sends requests to all the index-serving nodes, collects all the responses from them, and sends a appropriate response back to the client.

Data Caching

Data caching seeks to alleviate the load on databases that are used by large-scale web applications. Performance is a key concern when thousands of client requests must be supported simultaneously. A data cache can improve the performance by decreasing the look-up time for frequently accessed information. Spare memory in servers is aggregated to store frequently accessed results of database queries.

4.1.2 Experimental Setup

A four-node cluster is used as the evaluation testbed. Each node consists of an Intel Xeon E5-2620 processor, 16 GB of memory and a 256-GB hard disk. Each node runs a Linux OS (kernel version 3.16.5). For all the workloads, a separate server runs the workload generator or the client. We used the workloads from CloudSuite – a benchmark suite for emerging scale-out workloads [4]. Also, we used the latest versions of the software when possible. In rest of this section, we describe the software and configurations used for each of the scale-out workload under investigation.

Data Serving Workload

We use Cassandra (version 2.0.7) [1, 36] as our distributed NoSQL data store. Cassandra aims to manage large amounts of data distributed across many commodity servers. It provides a reliable, high-availability service using a peer-to-peer architecture. The data is split across each node in the cluster.

To generate the workload for our experiments, we use the Yahoo! Cloud Serving Benchmark (YCSB) (version 0.1.4) [21]. YCSB is a benchmarking framework to evaluate the performance of cloud data-serving systems. The framework consists of a load-generating client and a set of standard workloads, such as read-heavy or write-heavy workloads, which helps in stressing important performance aspects of a data serving system.

Configuration: We load 25 million records into the data store with a replication factor of three to simulate a realistic setup. The total data stored was approximately 80 GB in size (20 GB per server). We evaluate this setup using a predefined read-modify-write workload from YCSB. The data access pattern follows a Zipfian distribution. The arrival rate of requests follows a exponential distribution.[1] The YCSB client reports the performance achieved in terms of throughput and latency, specifically average latency and latencies at the 95th and 99th percentile. At 100% load-level with no power management, the system consumes 468 watts of power.

[1]We use exponential or negative exponential distribution for arrival rate of requests for all the scale-out workloads. These are the recommended distributions for the arrival rate of requests when real user trace is not available [43].

Web Search Workload

We use Apache Nutch (version 1.2) [2] as our web search workload. Nutch is an open-source web crawler and searcher. It provides a framework for distributed indexing and search. For the front-end, Apache Tomcat (version 7.0.23) [3] is used.

We use Faban toolkit (version 1.0.1) [5] as our workload generator. The Faban toolkit is a Java-based driver framework which allows for easy definition and load generation for new benchmarks. This framework allows the user to specify load parameters such as the ramp-up time, steady state time, ramp-down time, number of search users and number of client threads.

Configuration: We crawl the public Internet and use an index and segment size of approximately 1 GB and 14 GB, respectively, per server. The Faban driver uses a negative exponential distribution for the arrival rate of requests [6]. The Faban driver uses latency and throughput as the performance metric. It reports 90th and 99th percentile latencies. At 100% load-level with no power management, the system consumes 596 watts of power.

Data Caching Workload

We use Memcached [10] as our data caching workload. Memcached is a distributed in-memory key-value store for generic data. It is used to store popularly used data from database queries or page accesses.

The Memcached client provided with the CloudSuite is used as our workload generator. This client allows the user to specify various parameters such as ratio of get and set operations.

Configuration: We store approximately 15GB of data per node. A scaled version of the Twitter dataset provided with the CloudSuite is used. We use an exponential arrival rate distribution. The Memcached client reports performance in terms of throughput and latency (90th, 95th and 99th percentile latencies are reported). At 100% load-level with no power management, the system consumes 466 watts of power.

4.1.3 Power Measurement Interface

To study energy proportionality in the context of scale-out workloads, we measure the power consumption while executing these workloads at different load-levels. We use a *Watts Up* power meter to measure the cluster power consumption. Intel's Running Average Power Limit (RAPL) [8, 27] interfaces is used to measure power of components within a system.

4.1.4 Power Management

We study four types of power management techniques in this thesis. These techniques can be classified into two broad categories: power provisioning and resource provisioning. A brief overview of these categories is given below.

Power Provisioning

Power provisioning can be further divided into three power management techniques. We describe these mechanisms here.

Active Low-Power Modes: These low-power modes are the most popularly used form of Dynamic Voltage Frequency Scaling (DVFS). It is also referred as P-state (or performance state). They are voltage-frequency pairs which allow the processor to be operated at lower than the rated frequency. When the frequency along with voltage is reduced, the operational speed and the power consumption of the processor reduces. In this thesis, we use Intel RAPL to manipulate active low-power modes. RAPL interfaces provide mechanisms to limit the power consumption of the processor.[2] RAPL interfaces can be programmed using model specific registers (MSRs). We refer the reader to Section 2.2 for more information on RAPL interfaces. The rated and minimum frequency of the processors in our experimental setup is 2GHz and 1.2GHz, respectively. The rated and the minimum frequency are referred as $P1$ and Pn states ($n = 9$ in our experimental setup) [11]. In other words, there are nine voltage-frequency pairs at which the processor can operate.

Turbo Mode: This is another form of DVFS. This mode overclocks the processor above the rated frequency over a short duration to improve performance. This mode is hardware-controlled and the frequency[3] at which the processor operates depends upon the thermal and power headroom available to the processor. In our cluster, the processor can operate at a maximum frequency of 2.5GHz when in turbo mode. This maximum turbo frequency is referred as $P0$ state [11].

We disable and enable turbo mode by setting and clearing bit 32 of the *IA32_PERF_CTL* MSR. We also ensure that the appropriate P-states are used by writing the corresponding frequencies to the */sys/devices/system/cpu/cpux/cpufreq/scaling_max_freq* and */sys/devices/system/cpu/cpux/cpufre-*

[2]Internally RAPL uses DVFS [50].
[3]This frequency is above the rated frequency.

q/scaling_min_freq.

Idle Low-Power Modes: These modes are applied when the processor is idling (i.e., no instruction is being executed). It is also referred as C-state. Each C-state selectively shuts down supporting circuitry in the processor in order to save more power. $C0$ state is the active state and $C1$ to Cn (where, $n \in 1, 1E, 3, 6, 7$) are idle low-power modes. $C7$ state saves the most power. Each mode has an exit latency (i.e., the time taken to bring back the processor to $C0$ state from any Cn state) associated with it. Entering a deeper idle low-power mode incurs more cost in terms of exit latency as shown in Table 4.1.

Table 4.1: C-State Exit Latencies on Our Experimental Setup

C-State	C1	C1E	C3	C6	C7
Exit Latency (μs)	2	10	80	104	109

To dynamically control C-states, we have to write the maximum allowable exit latency to the file */dev/cpu_dma_latency*. As long as the file */dev/cpu_dma_latency* is kept open, C-states with transition latencies higher than the specified exit latency value will not be used. For example, writing a maximum allowable latency of 12 μs will keep the processors only in C0, C1 or C1E state.

Resource Provisioning

Resource-provisioning techniques aim to match the number of active servers to the load-level on the cluster. Idle servers can be simply turned off or provisioned to another application. However, there is an associated cost with bringing the servers back to active state as the load on the cluster increases. The evaluation of resource provisioning is done by manually hibernating nodes in the cluster. In other words, we manually match the number of nodes to the load-level of the scale-out workload.

4.2 Component-Level Power Distribution

In this section, we measure the component-level power consumption of the scale-out workloads under investigation. The main goal is to understand the contribution of each component to the energy proportionality of these workloads.

Figure 4.1 shows the power distribution for the data serving,[4] web search and data caching workloads for the entire cluster. The power consumption reported here corresponds to the processor P1 state with turbo and idle low-power modes disabled. In rest of the thesis, this configuration is referred as no management (NM). The values reported in Figure 4.1 are based on the sum of the power consumption from each node in the cluster and averaged over multiple runs. System components other than the processor and memory are represented as *Others*[5] in these figures.

In the NM configuration, the power consumption of each component does not vary across load-levels. Such constant power consumption, irrespective of the load-level, is the main reason for poor energy proportionality of scale-out workloads. In general, the processor power consumption contributes most to the system power. The processor consumes 45-50%, 65-70% and 50-65%

[4]Note: An analysis of other predefined workloads from YCSB is described in Appendix C.

[5]The other components, denoted by "Others," also include the power consumption of the hard disk. This power consumption is calculated as *system power - (processor power + DRAM power)*

for the data serving, web search and data caching workloads respectively. DRAM consumes less power when compared to the processor across all the workloads. The power consumption of *other* components varies depending upon the workload; however, their aggregate power consumption is always either similar (e.g., data serving) or less than the power consumption of the processor (e.g., web search).

Since the processor consumes a large portion of the system power, this thesis focuses on understanding the energy proportionality from the perspective of the processor. In the rest of this thesis, we study the utilization of the CPU and the effects of active low-power, turbo and idle low-power modes of the processor on the energy proportionality of the system for scale-out workloads. We also investigate the trade-offs involved in using resource provisioning and power provisioning.

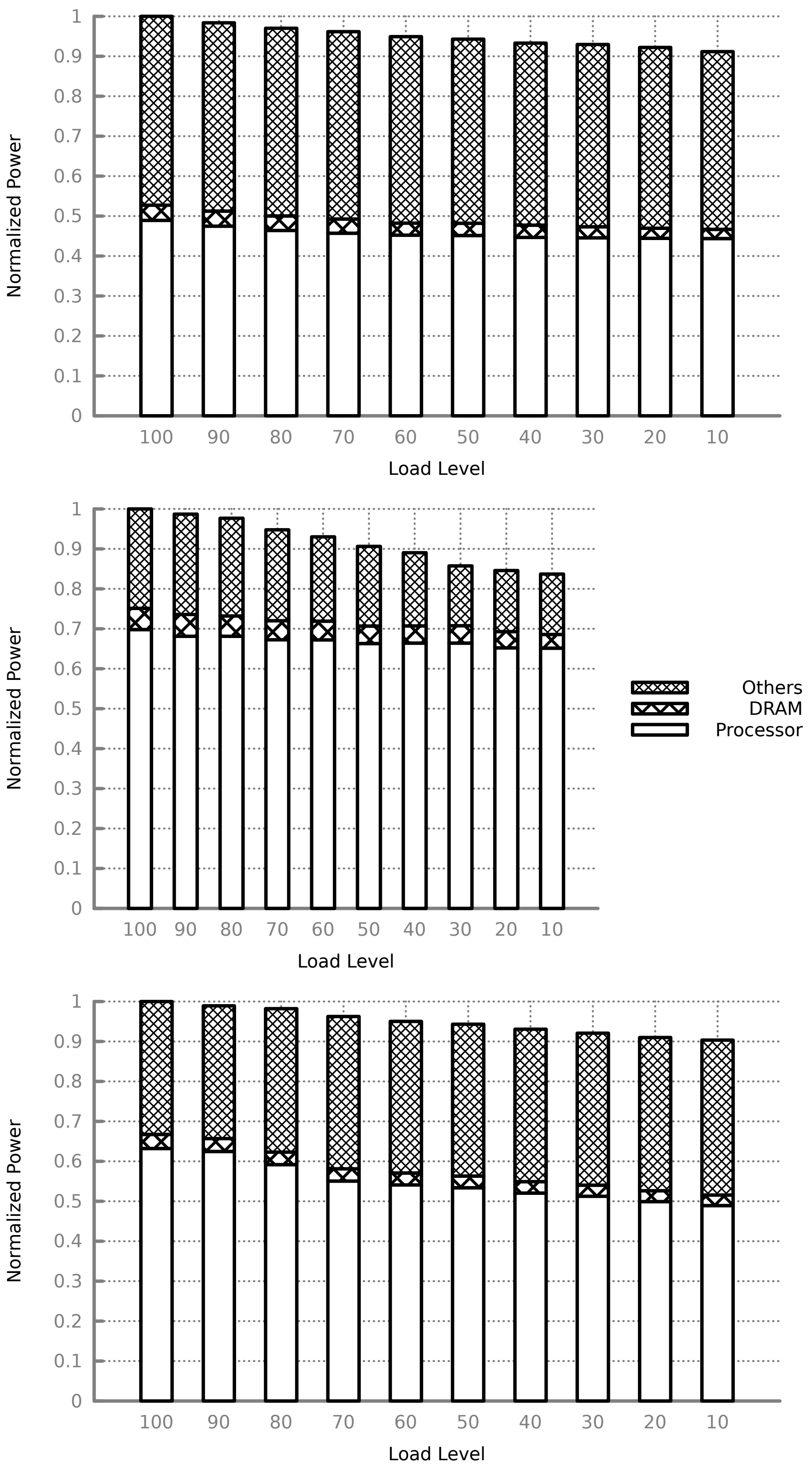


Figure 4.1: Component-Level Power Distribution. Top: Data Serving (Cassandra), Middle: Web Search (Nutch), Bottom: Data Caching (Memcached). Y-axis is Normalized to Power Consumption at 100% load-level.

4.3 Analysis of CPU Utilization

Here we present the CPU utilization traces of the scale-out workloads for three load-levels at different time granularities. Using these results, we seek to understand whether power management techniques will have any effect on these workloads. Moreover, these experiments provide a view into the time granularity at which any power management technique should be applied. From the perspective of processor usage, these results also show the similarity and diversity of the workloads used in this thesis. In this section, we will show the following: the potential to save power decreases with increase in time resolution and load-level and there is ample opportunity to save power at sub-second and sub-millisecond granularity at each load-level.

4.3.1 Methodology

The analysis of the CPU utilization is presented as CDF. These CDFs provide a succinct view of the fraction of time for which a workload at a particular load-level spent its execution at or below a CPU utilization. We also show CDFs at time granularities of 100 μs, 500 μs, 1 ms, 5 ms and 10 ms. This allows us to visualize the change in CPU utilization as we change time resolution at which we monitor it. These fine-grained CPU utilization traces are collected using SystemTap [19].

4.3.2 Discussion

Figures 4.2, 4.3 and 4.4 show the CPU utilization profile of the data serving, web search and data caching workloads, respectively, for three load-levels. Consider the 100-μs and 1-ms CDFs for data serving workload (see Figure 4.2). The time spent at 0% CPU utilization (i.e., idling) for both CDFs decrease when moving from 30% load-level to 70% load-level. It also decreases when you move from 100 μs to 1 ms within a load-level. In general, the CDFs show that opportunity to save power and the amount of time spent idling decreases with increase in load-level and time resolution within a load-level.

The fraction of time spent idling (i.e., 0% utilization) at a time resolution (especially at sub-millisecond resolution) provides insights into the effectiveness of idle low-power modes on these workloads. At 5-ms and 10-ms resolution, the data serving workload spends less time idling. Whereas, the other two workloads spend more time idling than the data serving workloads even at this resolution. It is expected that more time spent idling allows the processor to enter low power modes frequently and thus save power. These plots also provide insights into the time resolution at which to apply dynamic power management mechanisms. For example, in all cases the best time resolution to apply power management mechanism is 100 μs as it provides the best opportunity for power savings due to the high fraction of idle time. Moreover, if the CDFs of two time resolutions follow the similar curve it is better to apply the power management mechanism at the larger time resolution in order to avoid unnecessary overhead. For example, the 5-ms and 10-ms CDFs for the 70% load-level of the data serving and data caching workload follow a similar curve. As a result, we can infer that there is little opportunity to save much additional processor power when using a 5-ms time resolution than when using a 10-ms time resolution.

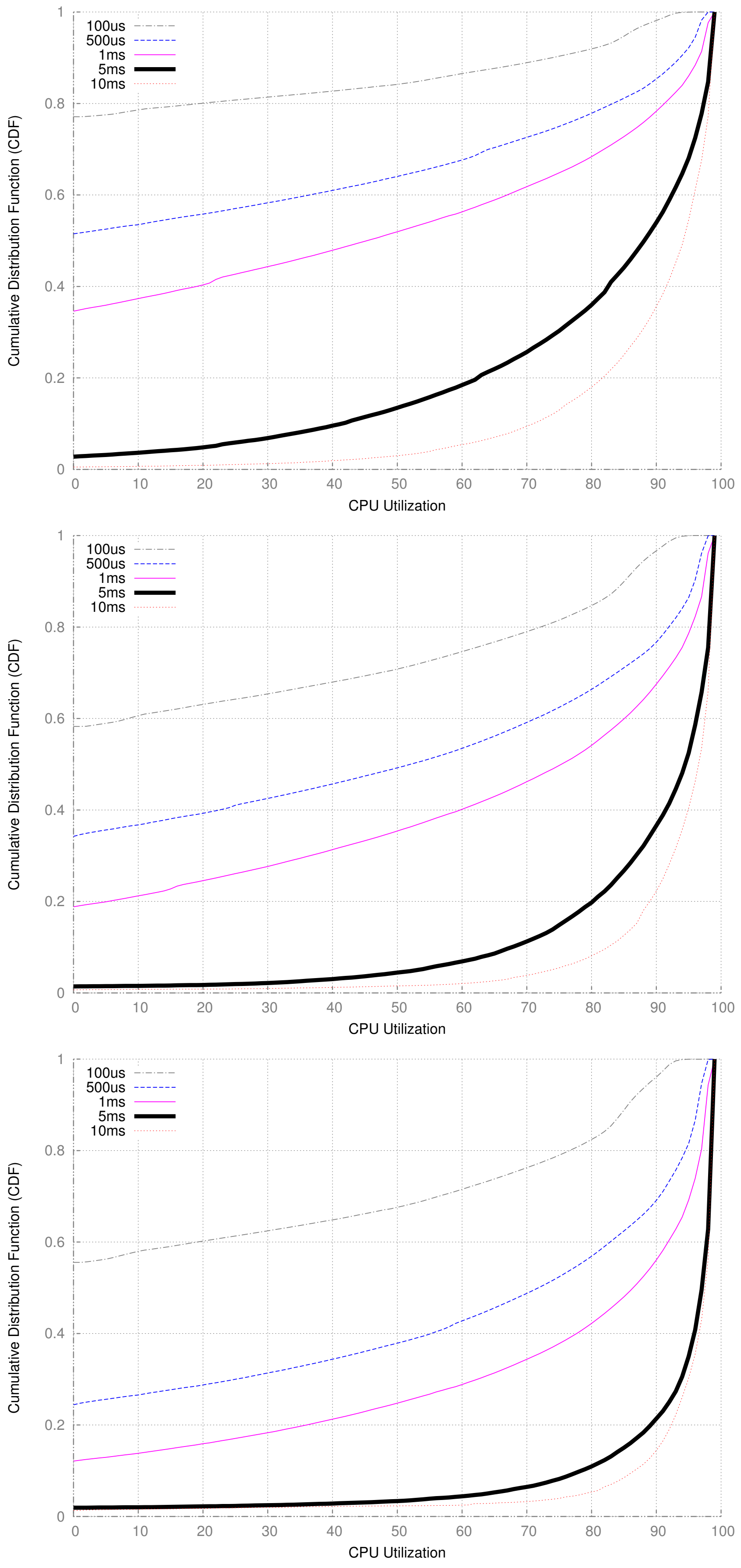


Figure 4.2: CPU Utilization of Data Serving (Cassandra) At Different Time Resolution. 30% (Top), 50% (Middle) and 70% (Bottom) Load-Levels

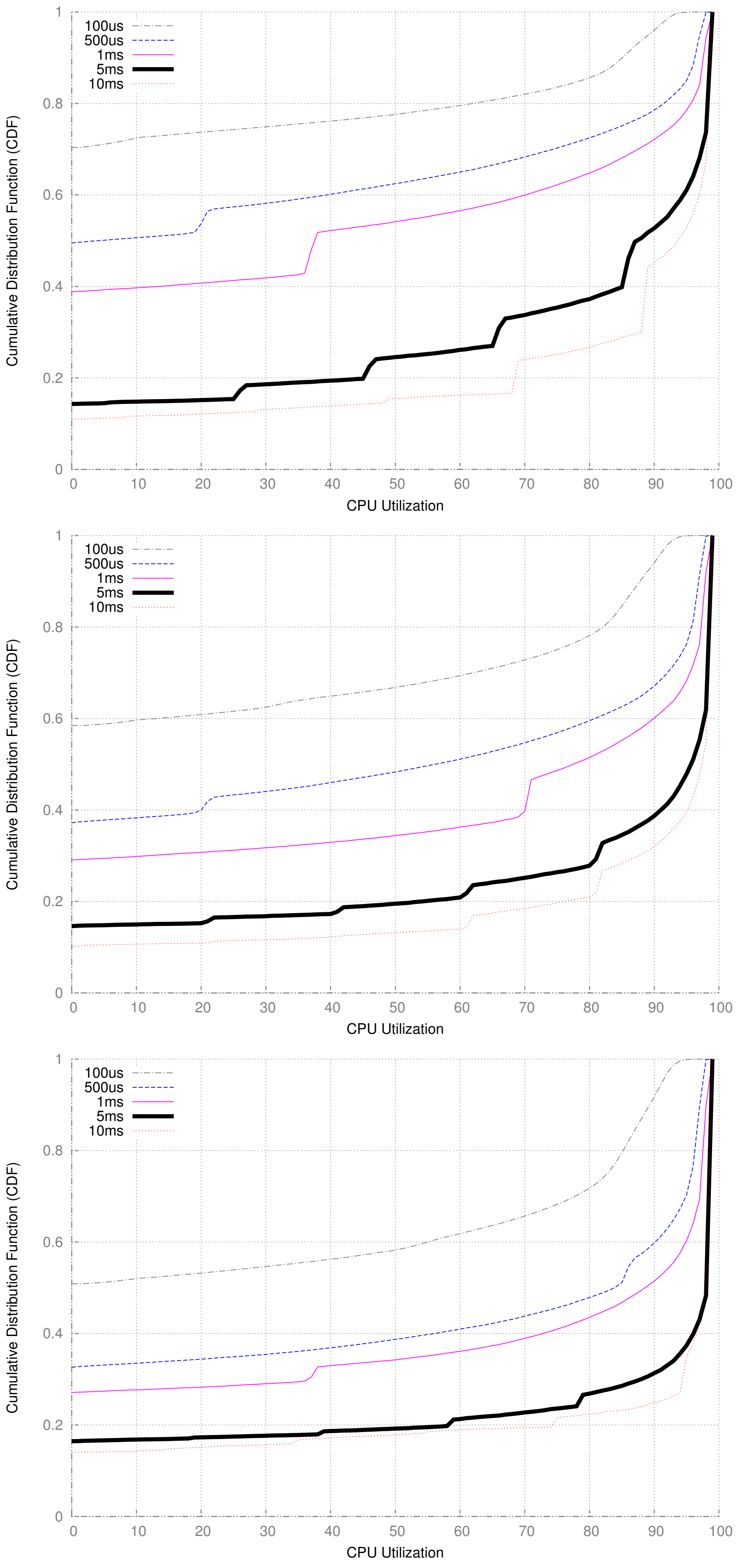


Figure 4.3: CPU Utilization of Web Search (Nutch) At Different Time Resolution. 30% (Top), 50% (Middle) and 70% (Bottom) Load-Levels

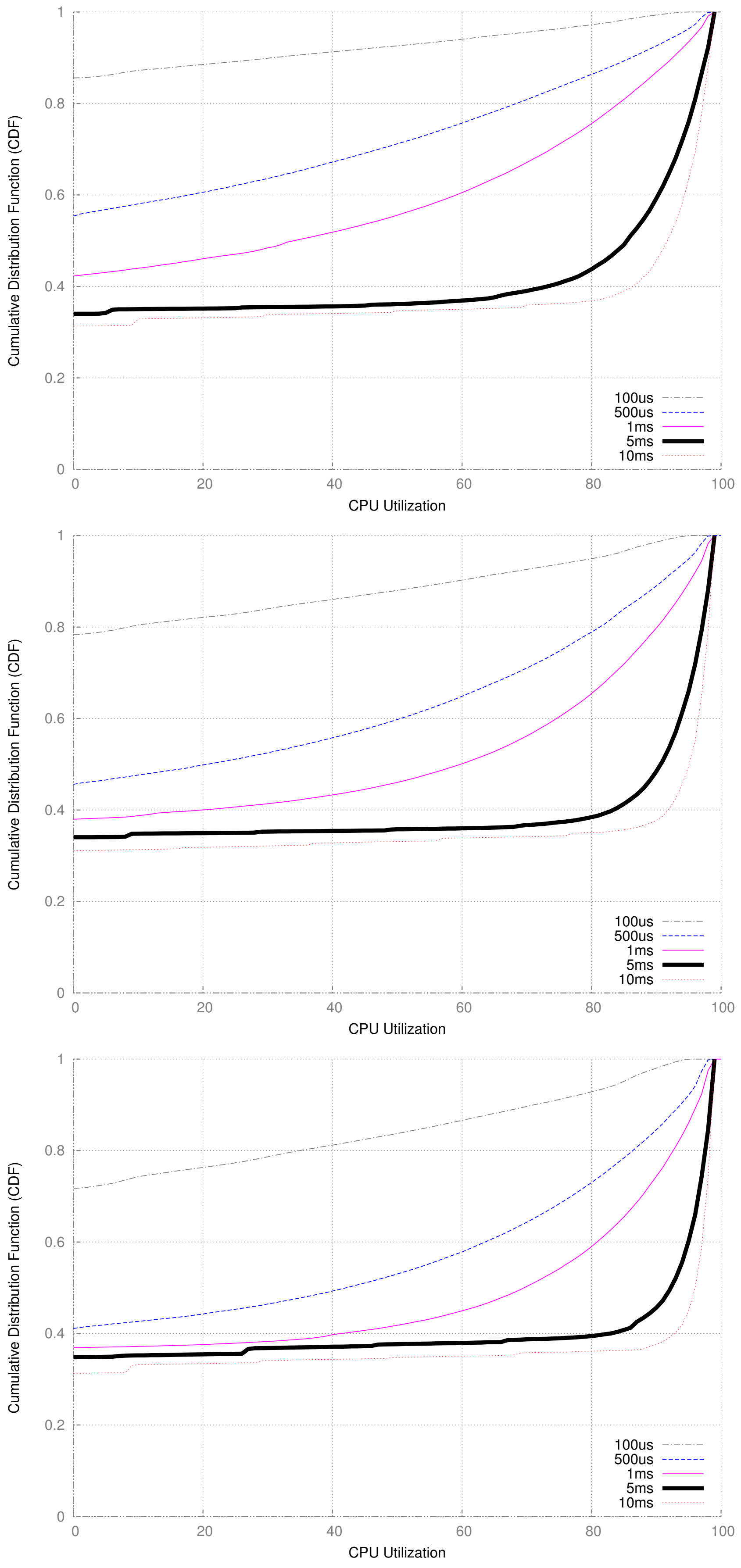


Figure 4.4: CPU Utilization of Data Caching (Memcached) At Different Time Resolution. 30% (Top), 50% (Middle) and 70% (Bottom) Load-Levels

4.4 Effect of Power Provisioning

In this section, we describe the effects of the power provisioning techniques (i.e., active low-power, turbo and idle low-power modes) on the energy proportionality, corresponding response times (latency) and power saving potential for the scale-out workloads.

We describe these effects using Figures 4.5, 4.6, 4.7, 4.8 and 4.9. The No Management (denoted as NM) line in each figure corresponds to the power consumed by each workload when the system is restricted to the P1 state (i.e., 2.0GHz) and C0 state with turbo mode disabled. For studying the effects of idle low-power modes, we restrict the deepest possible C-state that can be entered by writing appropriate latency values to the */dev/cpu_dma_latency* device file as described in Section 4.1.4. The effects of turbo mode on these workloads are studied using the procedure described in Section 4.1.4. To understand the potential of active low-power modes, we run through all possible RAPL processor power limits possible on each load-level such that the throughput is maintained. In this section, we present only the results from the best possible RAPL configuration (i.e., the configuration which achieves the best power savings while meeting throughput constraints). We also show the effects of RAPL in conjunction C7 state (denoted as RAPL+C7). We do not show results which restricts the deepest possible C-state to C1 and C6 because C1E and C7 had identical effects to C1 and C6, respectively, for all the workloads. In all cases, using only turbo mode resulted in power consumption that exceeded the power consumption of each workload at 100% load-level. Hence, we do not show results corresponding to only turbo mode. The results corresponding to turbo mode in conjunction with C7 state (denoted as Turbo+C7) is shown

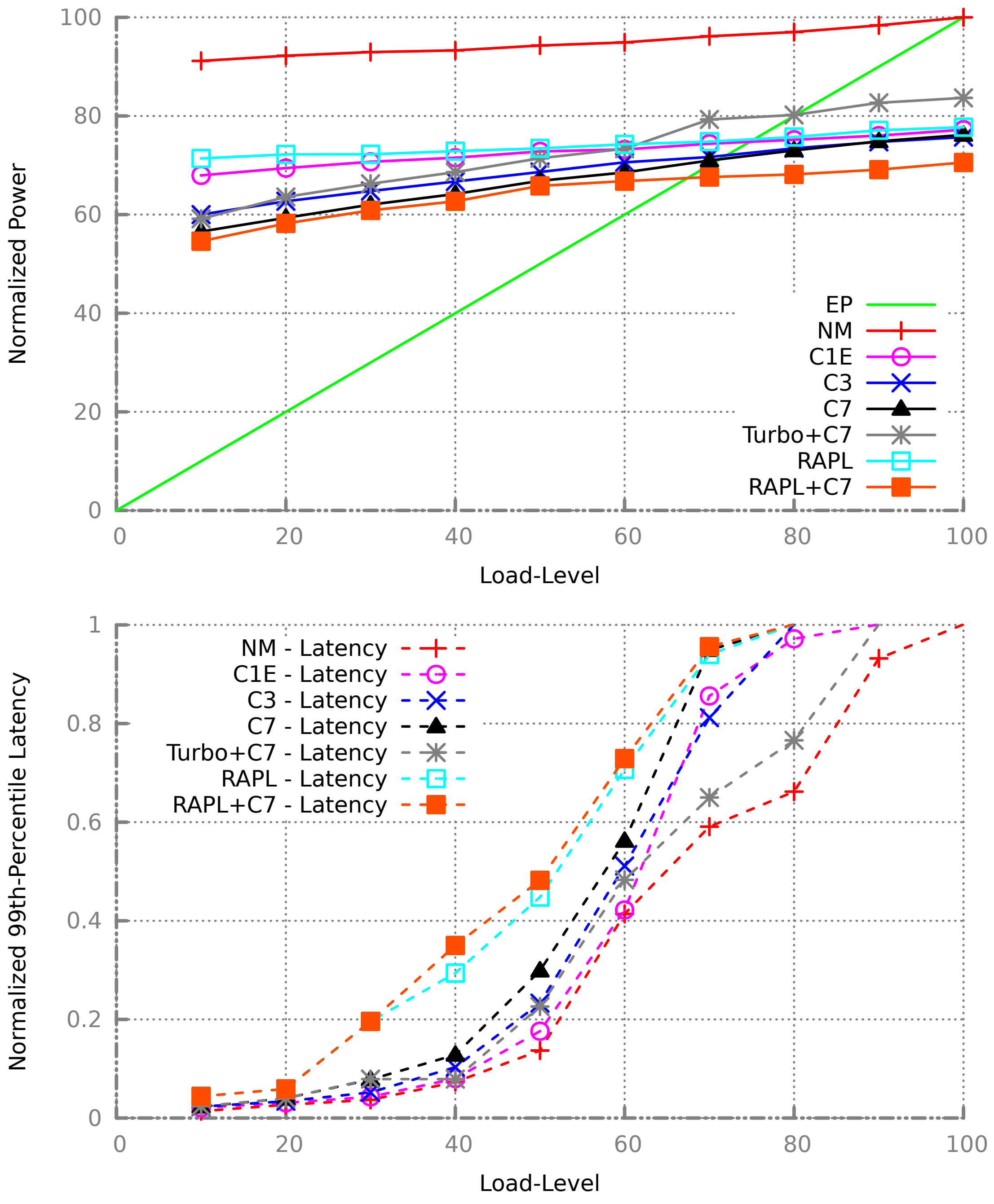


Figure 4.5: Data Serving (Cassandra) Workload. Energy Proportionality (Top) and Latency (Bottom). NM = No Management.

instead.

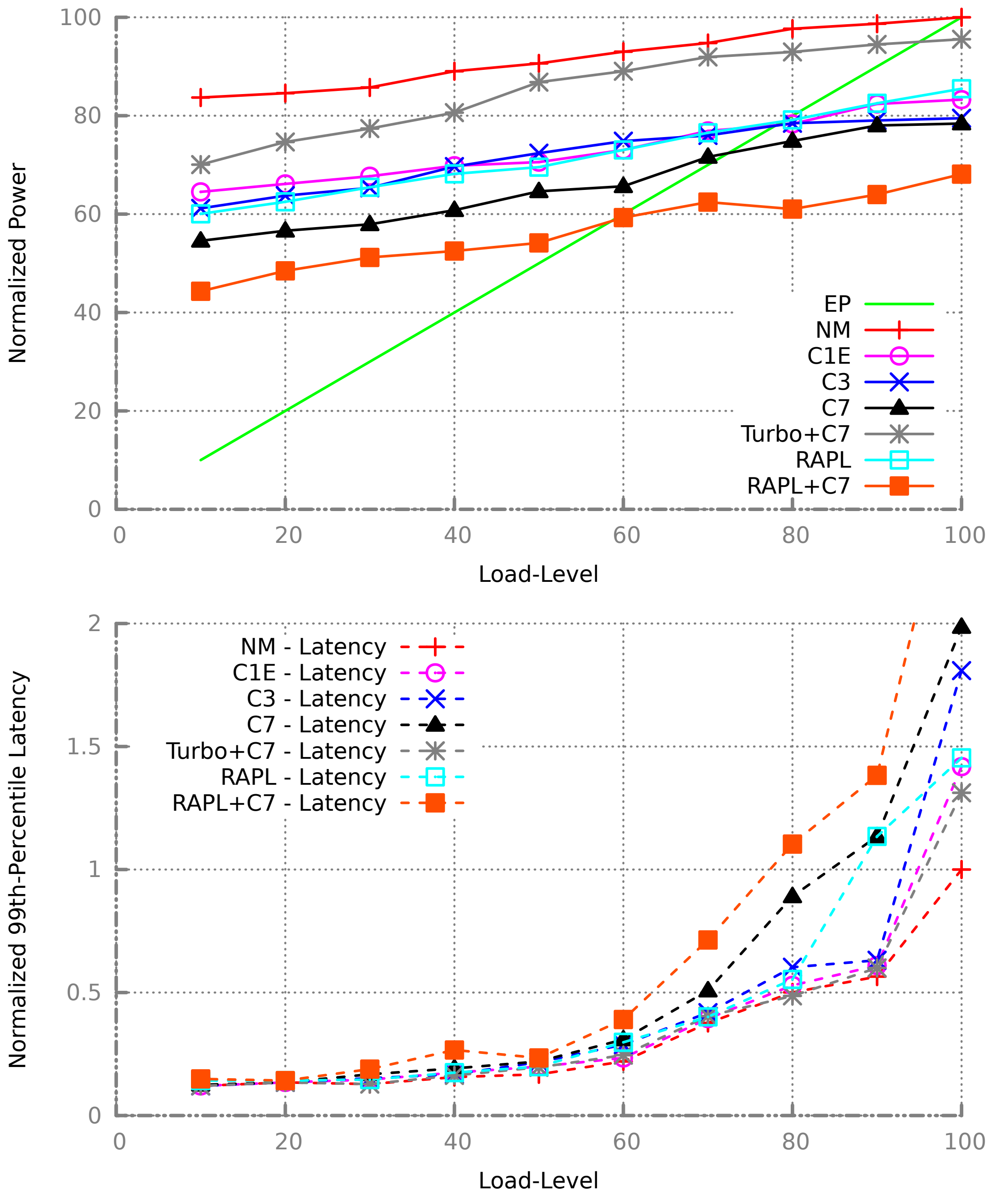
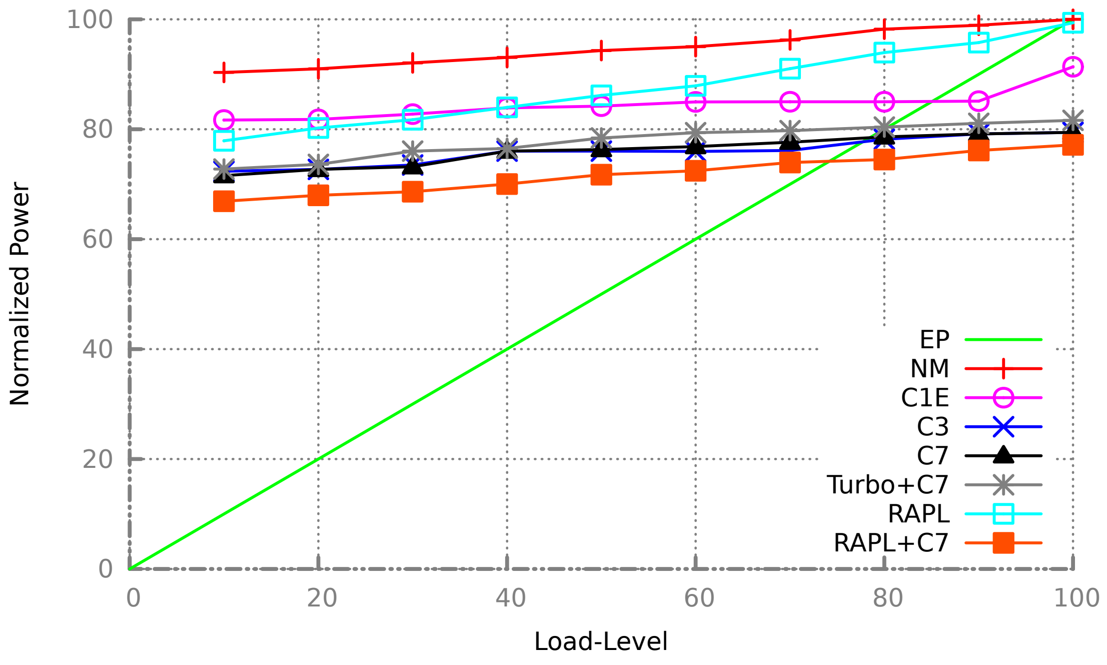


Figure 4.6: Web Search (Nutch) Workload. Energy Proportionality (Top) and Latency (Bottom). NM = No Management.



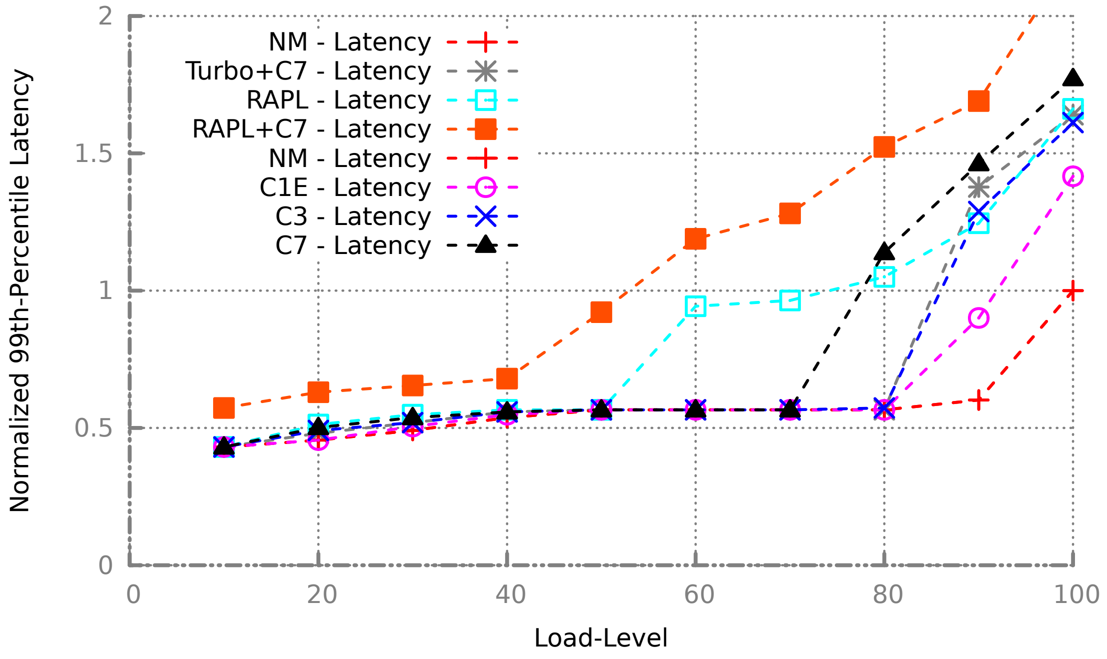


Figure 4.7: Data Caching (Memcached) Workload. Energy Proportionality (Top) and Latency (Bottom). NM = No Management.

4.4.1 Effects on Energy Proportionality

The top plots in Figures 4.5, 4.6 and 4.7 show the effect of power provisioning on the energy proportionality of scale-out workloads. The Y-axis represents the normalized power (normalized to the power consumed at 100% load-level) by the system and X-axis represents the load-level. As a result, the energy-proportional curve (denoted by EP) consumes 40% of power at 40% load-level, 60% of power at 60% load-level and so on.

As observed, better than energy-proportional operation is not possible under all load-levels for these workloads. However, energy-proportional operation can be achieved for certain load-levels depending upon the workload. For example, better than energy-proportional operation can be achieved for load-levels greater than 70%, 60% and 80% load-level for data serving, web search and data caching workloads, respectively, using RAPL+C7. Furthermore, in every case, energy proportionality is improved.

Overall, RAPL+C7 achieves the best energy proportionality. But in certain cases, the C7 state improves energy proportionality as much as RAPL+C7. However, in Section 4.3, we showed that there is ample idle time available for the idle low-power modes to save power even at sub-millisecond time granularity. To understand the reasons behind C7 state not performing as expected, we looked at the C-state residency (i.e., time spent in each C-state) shown in Figure 4.8. As observed, the processor does not spend an amount of time proportional to idle time in deep idle low-power states such as C3, C6 or C7 even though there is ample opportunity to do so in the 100us and 500us time granularities (see Section 4.3 for an illustration of power saving opportunity at varying time granularities). A huge fraction of time is spent in C0 state even though the CPU is idling. *From these results, we infer that there is a need for better system software to enter deep low-power states more aggressively in order to save power.*

The best idle low-power mode power savings come from allowing the processor to use the

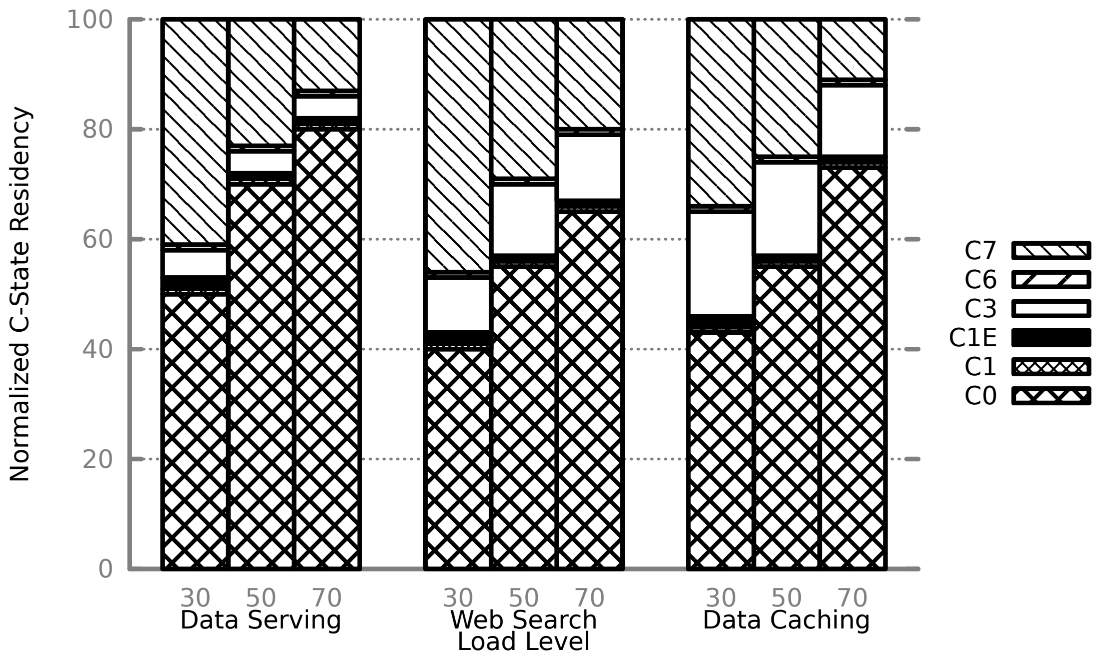


Figure 4.8: C-State Residency of Scale-Out Workloads

C7 state as expected. Using Turbo+C7 does not improve energy proportionality for high load-levels. But some improvement is seen when Turbo+C7 is used at low load-levels. However, it does not do better than using only C7. While these plots only describe the improvements in energy proportionality, the scale-out workloads are latency-sensitive. The latency of these workloads might be affected due to power management techniques even though the throughput (i.e., load-level) can be maintained. We discuss the effects of using these power management techniques on the latency of scale-out workloads in the next section.

4.4.2 Effects on Latency (Response Time)

The plots in the bottom of Figures 4.5, 4.6 and 4.7 show the effect of power provisioning on the 99th-percentile latency (i.e., response times) of scale-out workloads. The Y-axis shows the normalized 99th-percentile latency. All the latencies are normalized to the 99th-percentile latency at 100% load-level with NM. X-axis presents represents the load-level. These curves represents a typical response time curve for a scale-out workload. Each curve has a inflection point after which the response times rapidly rises. This inflection point for NM curve lies at 50%, 60% and 90% load-level for the data serving, web search and data caching workloads, respectively. It is clear from the plots that each power management technique has its own unique effect on the response times of the workload. For example, the inflection point is shifted when a more aggressive power management technique is used. We also observe that for certain load-levels all the latency curves are similar. In case of web search, all the latency curves are similar after 60% load-level suggesting that a more aggressive power management technique can be used without having any effect on the response times of this workload at or below 60% load-level. Similar inferences can be made for other workloads and power management techniques.

In practice, these workloads operate under strict SLOs. This SLO is placed on the 99th-percentile latency and fixed for a particular workload. Any violation of SLO is unacceptable. As such, a mechanism to trade-off some power savings for improvements in response times is desirable. In Section 4.5, we describe the potential of RAPL to provide a trade-off between power and latency. This makes dynamic power management using active low-power modes an attractive option for improving the energy proportionality of scale-out workloads while meeting strict SLOs.

4.4.3 Power Savings

Figure 4.9 shows the power savings achieved for different load-levels of the scale-out workloads. We show the power savings at the cluster- and processor-level. The processor-level power savings include the combined power consumption of all the processors in the cluster. This processor power consumption is measured using RAPL interface. We only show the processor-level power consumption as the turbo, active low-power and idle low-power modes affect only the processor power consumption. The power saving is calculated as $(power_{NM} - power_{mode})/power_{NM}$. At the cluster-level, we save the highest power using RAPL+C7 in all cases. The difference in power savings achieved due to RAPL+C7 when compared to C7 alone is substantial for each workload. We save up to 40%, 47% and 26% for the three different workloads at the cluster-level when compared to the power consumption with no management. While comparing the power saving at the processor-level, the difference between the power savings achieved with RAPL+C7 and C7 is even more pronounced. The maximum power savings achieved for the scale-out workloads at the processor-level are 77%, 76% and 67% for the three different workloads.

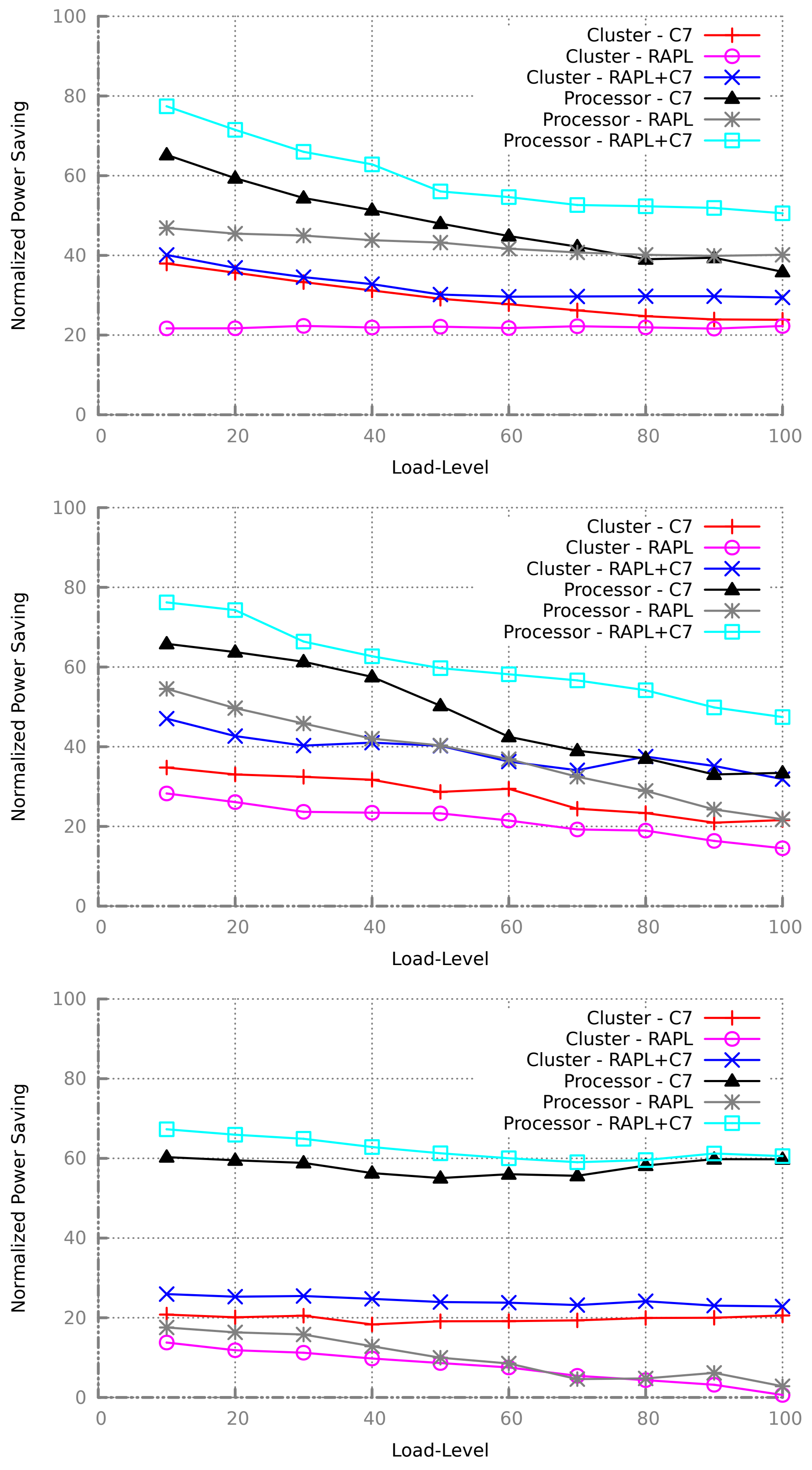


Figure 4.9: Power Savings. Data Serving (Top), Web Search (Middle) and Data Caching (Bottom).

4.5 Trade-Offs in Power and Latency

As mentioned earlier, active low-power modes not only provide the best power saving but also gives us an opportunity to operate a given workload in a configuration to meet strict SLOs by creating a power-performance trade-off space. In this section, we describe this space with respect to scale-out workloads. Our methodology involves the projection of power consumed for a particular latency while running the scale-out workload on our experimental testbed. We run through different possible RAPL processor power limits and present the impact of each configuration on the power and performance (latency) of these workloads.

Figures 4.10 and 4.11 show the power-performance trade-off space for the 30%, 50% and 70% load-level of the data serving and web search workload, respectively. In these figures turbo mode was disabled and the processor was restricted to the C0 state. Essentially, it is power-performance exhibited by only using RAPL.[6] The X-axis represents normalized 99th-percentile latency. All the latencies are normalized to the 99th-percentile latency at 100% load-level with no power management. The Y-axes represents normalized power and power savings, respectively. The latency and power savings are normalized with respect to no management (NM) mode at the same load-level. However, power is normalized to the power consumed at 100% load-level in order to illustrate energy proportionality. For 70% and 50% load-levels of the data serving workload, the power vs. latency curve follows a similar trend at the cluster-level. They provide best power savings on the left side of the graph where the latency penalty is less. However, as more aggressive power limiting

[6]We do not show results for the data caching workload since it does not exhibit any variation for latency in RAPL mode for 30% and 50% load-level as shown in Figure 4.7.

is applied, the power savings achieved stagnates.

In the case of the 30% load-level at the cluster-level and in the initial run through of the power limits, the latency increases more than the power savings it provides. In the center part of the plot, there is an inflection point which provides the best power savings for little increase in latency and then, the power savings achieved stagnates. For all load-levels at the processor-level, the power vs. latency curve follows a similar trend. Similar observations can be made for the power-performance trade-offs of the web search workload as shown in Figure 4.11. Moreover, the trade-off space exhibited by each workload depends upon the load-level. For example, the web search workload gives an opportunity to operate it anywhere in the $1 - 1.15x$ latency cost range depending upon the load-level. Such inferences on the relationships between power and latency are useful to design a dynamic power management runtime for scale-out workloads.

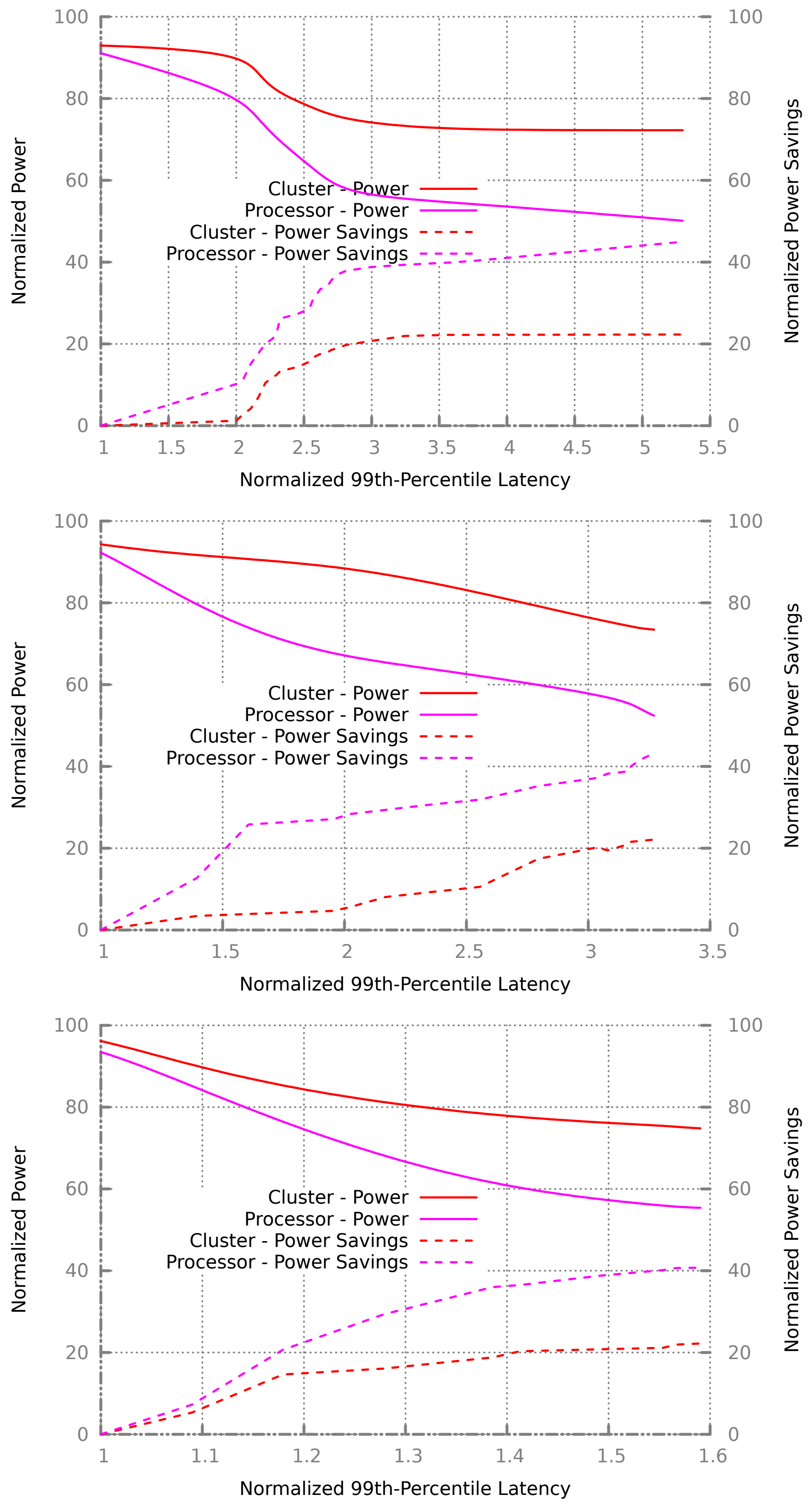


Figure 4.10: Trade-Offs in Power and Latency. Top: 30%, Middle: 50%, Bottom: 70% Load-Levels) of Data Serving Workload

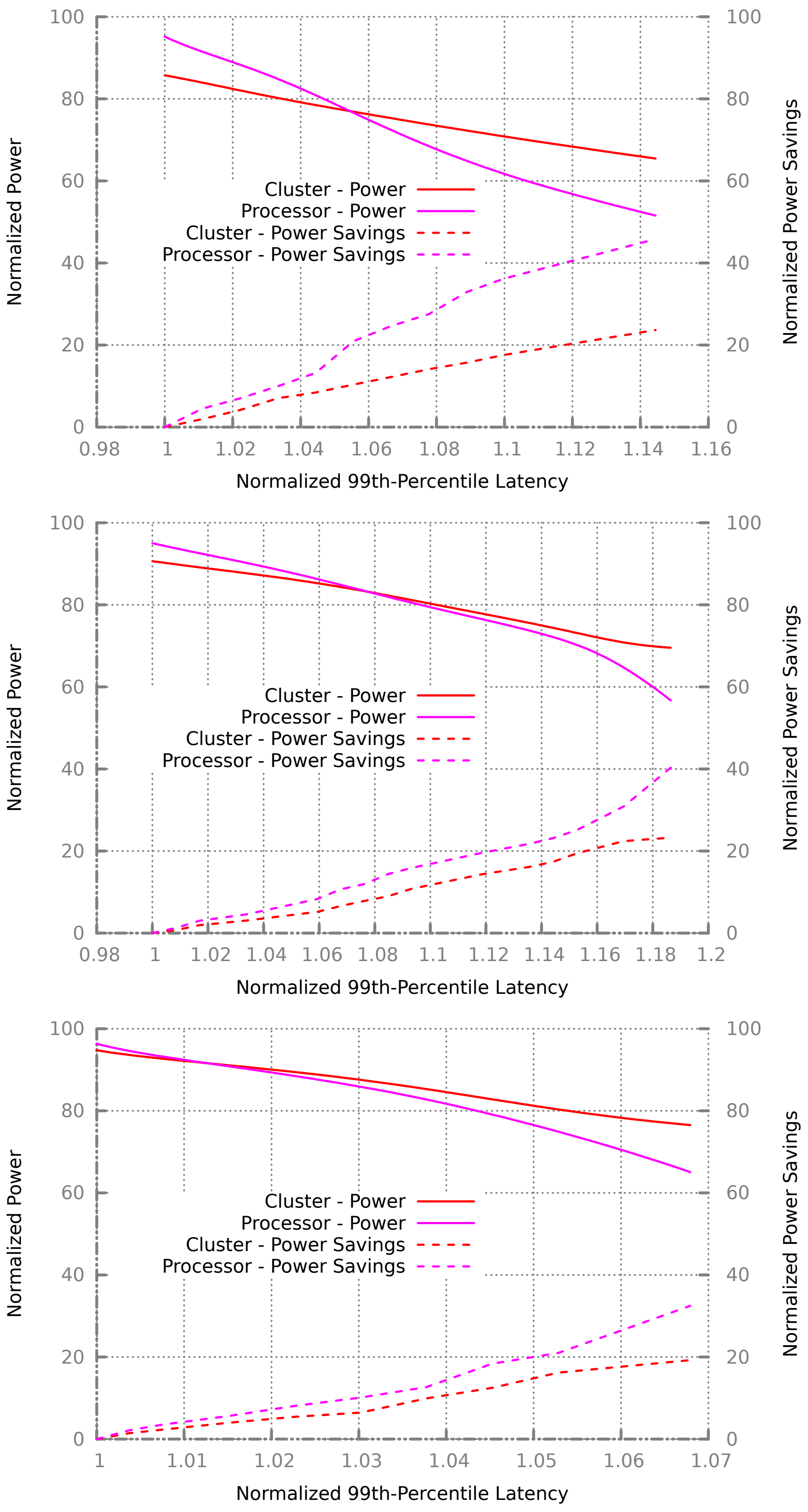


Figure 4.11: Trade-Offs in Power and Latency. Top: 30%, Middle: 50%, Bottom: 70% Load-Levels) of Web Search Workload

4.6 Power vs. Resource Provisioning

In this section, we evaluate the impact of resource and power provisioning techniques on the energy proportionality and power savings achieved. Our main goal is to compare and contrast the above two in the context of scale-out workloads.

4.6.1 Experimental Setup and Workload Configuration

For the experiments in this section, we use a different cluster – 32 nodes from the Shadowfax cluster housed at Virginia Tech. Each node consists of two Intel Xeon E5-2670 processors (each with eight physical cores), 64 GB of memory and a 1 TB hard disk. A separate server runs the workload generator or the client.

We evaluate only the data serving workload for the experiments in this section. Cassandra (version 2.0.7) is used as the NoSQL data store. To generate the workload for our experiments, Yahoo! Cloud Serving Benchmark (YCSB, version 0.1.4) is used. We load 300 million records into the data store with a replication factor of nine. The total data stored was approximately 2.85 TB in size (90 GB per node). We evaluate this setup using a predefined read-modify-write workload from YCSB. The data access pattern follows a Zipfian distribution. The arrival rate of requests follows an exponential distribution.

4.6.2 Methodology

To understand the effects of resource provisioning, we change the number of servers involved in our experiments at each load-level from 25 to 32 nodes and report the effects of each configuration on the power and performance (in terms of latency). We first start the client with all the 32 nodes on the Cassandra cluster operational. After the client starts receiving responses, we remove the nodes from Cassandra cluster. This methodology ensures that we use a realistic setup as we take the effects of removing a node from the Cassandra cluster on power and performance into account. To analyze the effects of power provisioning, we run through different possible RAPL processor power limits and present the impact of each configuration on the power and performance of the data serving workload.

4.6.3 Discussion

In this section, we run the data serving workload (Cassandra) at 30%, 40% and 50% load-levels. Figure 4.12 shows the power vs latency trade-off for the load-levels using power and resource provisioning. The left Y-axis shows the power normalized to the power consumed at the 100% load-level using all the 32 servers. We also show the power saving along the right Y-axis.

We are not able to achieve energy proportionality for the three load-levels presented in the figure. However, there is a good power saving potential depending upon how much performance (latency) we are willing to sacrifice.

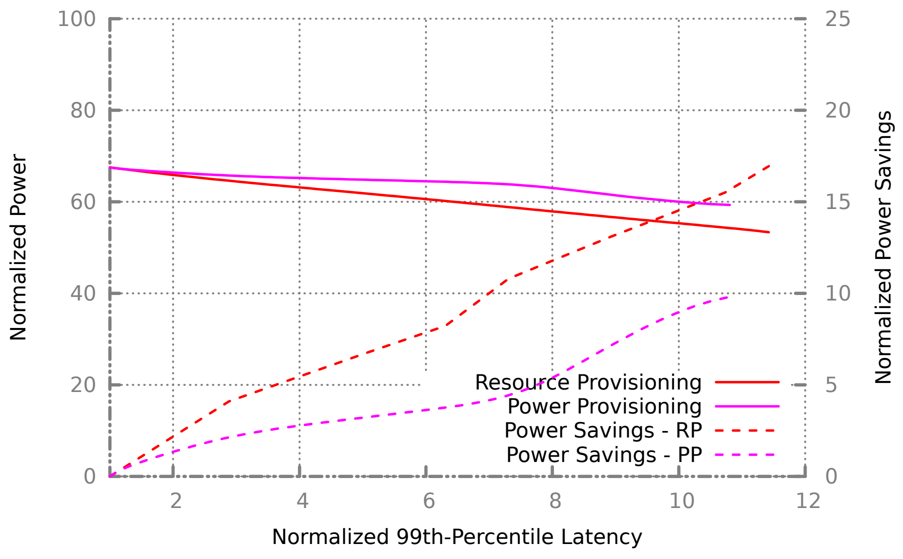
These plots can be used to make other interesting observations. For example, the plots show that the best power saving technique for a given latency depends on the load-level. The best power saving technique for the 50% load-level is power provisioning. However, for 30% load-level it is resource provisioning. In fact, the best power saving technique gradually shifts from power provisioning to resource provisioning when moving from the 50% to 30% load-level. The reason for such a trend is that at low load-levels the idle power becomes a large portion of the cluster power consumption and active low-power modes are not capable of improving the power savings further. At low load-levels, the only way to improve energy proportionality is to use resource provisioning to offset the idle power with an associated latency cost.

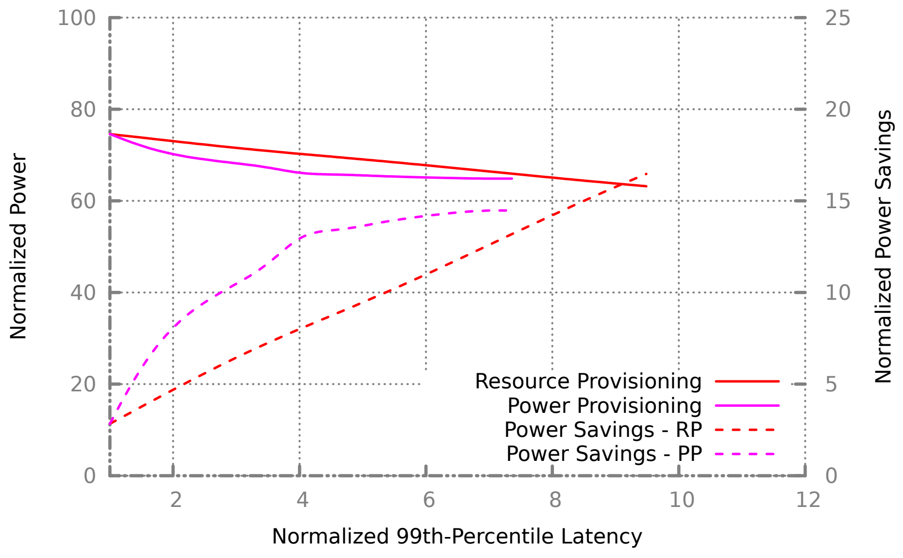
4.7 Comparison with Related Work

In this section, we compare the work presented in this chapter with Pegasus [39] and KnightShift [59, 60] as shown in Table 4.2. As mentioned in Section 2.1, this chapter complements [39] by providing more analysis with respect to CPU utilization, turbo and active low-power modes, trade-offs in power and latency and the comparison of power and resource provisioning. When compared to [59] and [60], we study the energy proportionality of scale-out workloads on a commodity cluster setup. We study the effects of power and resource provisioning on the energy proportionality, performance (in terms of response time) and power savings of scale-out workloads. We also provide empirical results for the trade-offs involved in power and resource provisioning on a *real* system.

Table 4.2: Comparison with Related Work

Features	Chapter 4	Lo et al. [39]	Wong et al. [59, 60]
Real System	✓	✓	×
Commodity Infrastructure	✓	✓	×
Cluster Analysis	✓	✓	✓
Cluster-Level Energy Proportionality Analysis	✓	✓	✓
Relationship Between Load-Level and CPU Utilization	✓	×	×
Impact of Active Low-Power Modes on Energy Proportionality	✓	✓	×
Impact of Idle Low-Power Modes on Energy Proportionality	✓	✓	×
Impact of Turbo Mode on Energy Proportionality	✓	×	×
Trade-offs in Power and Latency for Active Low-Power Modes	✓	×	×
Runtime System / Power vs. Resource Provisioning	✓	✓	✓





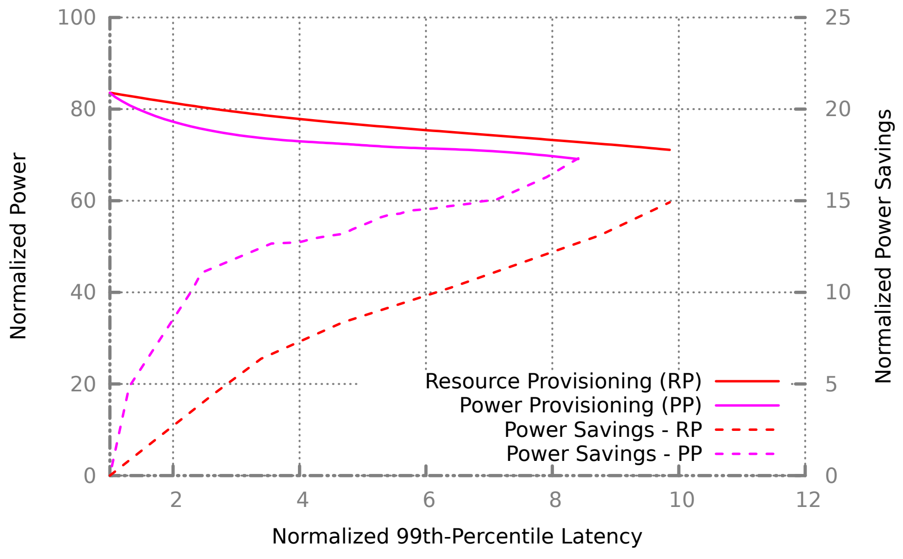


Figure 4.12: Comparison of Power and Resource Provisioning Using Power vs. Latency Trade-Off. Top: 30%, Middle: 40%, Bottom: 50% Load-Levels of Data Serving Workload)

Chapter 5

Conclusion and Future Work

In this chapter, we summarize the contributions made in this dissertation. We then provide a brief overview of the future work.

5.1 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. As such, energy proportionality can significantly improve the energy efficiency of data centers. This dissertation presented our investigation into improving energy proportionality at the server- and cluster-level.

5.1.1 Analysis of Sever-Level Energy Proportionality

In this thesis, we use RAPL interfaces to analyze and model the performance (both throughput and response time) of enterprise applications under subsystem-level power limits. We show that performance under a subsystem-level power limit can be modeled using 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.

5.1.2 Analysis of Cluster-Level Energy Proportionality

We then evaluate the potential of power and resource provisioning to improve the energy proportionality for scale-out workloads. Using data serving, web searching and data caching as our representative workloads, we show that processor is still the dominant power consuming component. We illustrate that there is ample opportunity to save power by characterizing the CPU utilization of scale-out workloads. We then present the potential of power provisioning techniques to improve the energy proportionality of scale-out workloads. The ability of active low-power modes to provide a power-performance trade-off space for scale-out workloads is described. We also compare and contrast power and resource provisioning techniques. Our study shows that effective power and resource provisioning improves the energy proportionality of scale-out workloads and exposes the trade-off involved in power and resource provisioning.

5.2 Future Work

In this section, we give a brief description of the possible extensions to this thesis. The future work is divided into two parts: (1) improving energy proportionality of enterprise class application using enhanced software and hardware approaches and (2) improving energy efficiency in high-performance computing (HPC) environments using power and resource provisioning.

5.2.1 Improving Existing Software and Hardware Approaches

Not all workloads benefit from using a single type of power management technique. For example, using turbo mode might only be beneficial for certain type of workloads and allowing deeper idle low-power states might violate service-level objectives for other types of applications. Moreover, there might be certain hours in a day where the data center might get varied amount of requests. As such, it would be beneficial to design a runtime system which can use active low-power, turbo and idle low-power mode to improve energy proportionality. This future work will involve the design of a runtime system with the ability to switch between active low-power, turbo and idle low-power mode in a coordinated fashion. This extension will allow us to improve energy proportionality for a variety of enterprise workloads.

On the hardware end, we seek to eliminate the mismatch between the workload profile and sever's energy efficiency by taking key operating regions into account. Traditional servers are designed to operate at maximum energy efficiency when the processor is maximally utilized. Identification of opportunities for hardware-software co-design to improve the energy efficiency of enterprise workloads will be key to bridging this gap. We want to explore methodologies to systematically identify hardware changes that will improve the energy proportionality for enterprise workloads.

5.2.2 Improving Energy Efficiency in HPC Environments

This thesis has dealt with improving energy proportionality for enterprise applications. However, there is also opportunity to improve the energy efficiency of HPC data centers using power and resource provisioning as shown in [48]. The main objective of this future work would be to investigate whether we can optimize the power consumed and the resource provisioned for an scientific application using statistical models. This extension will also involve the design and deployment of a runtime system for power management. As a part of this future work, we want to analyze the energy efficiency of scientific workloads at the cluster-level and gain insights into the opportunities for power savings. We then want to leverage our understanding of the energy efficiency of scientific application to create an optimization framework using statistical models to execute the workload on an optimal number of compute nodes in the cluster and set appropriate power limits at the subsystem-level. Finally, we want to use the knowledge gained by designing these models and optimization framework to create a runtime system for power management using simple system-level metrics.

Appendix A

SPECpower Batch Size has No Effect on Power Consumption

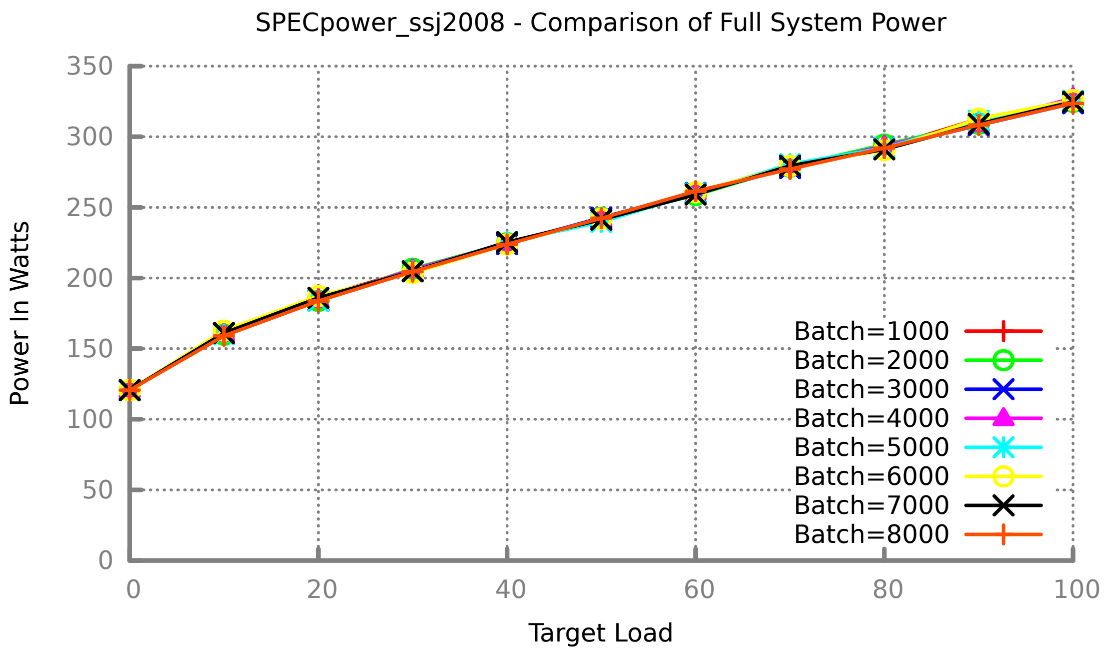


Figure A.1: Average Power Consumption of SPECpower

Figure A.1 shows the average power consumption of SPECpower benchmark at different load-levels. For full-system power measurement, we have followed the power measurement methodology specified and developed by the SPEC organization for the SPECpower benchmark [14]. The figure also shows the effect of changing the batch sizes in SPECpower. We wanted to quantify this effect as batching queries to exploit and create opportunities for power management is a well-researched area [42]. The number of transactions in each batch scheduled in SPECpower benchmark is calibrated using the *input.scheduler.batch_size* input parameter. We use eight different batching sizes from 1000 to 8000 in steps of 1000. Each data presented is the average of 10 runs. The standard deviation for the power consumed during the runs were less than $\pm 2\%$ of the mean.

Our results shed light on the repeatability of our experiments and the consistency of SPECpower benchmark. We observe that the lines in the plot overlap each other. Based on our experiments, the batch sizes have minimal to no effect on the power consumed by the benchmark. Similarly, changing batch sizes did not have any effect on the power consumption of subsystems (i.e., package, core and memory) as well.

Appendix B

SPECweb at 13000 SUS is Network Intensive

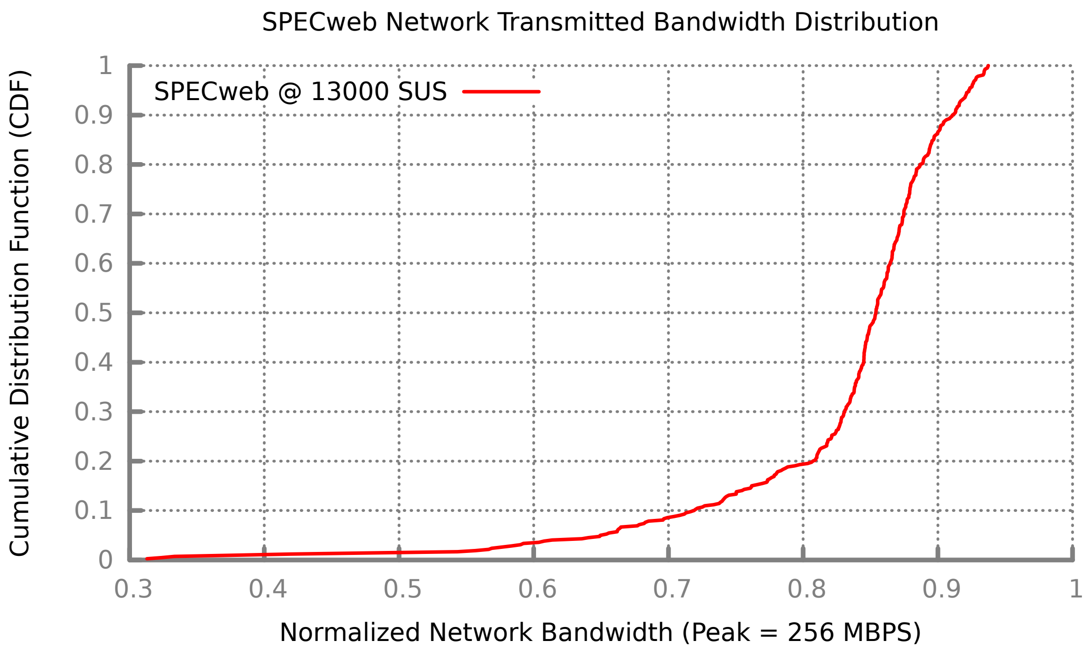


Figure B.1: CDF of Network Bandwidth - SPECweb at 13000 SUS

Figure B.1 shows the cumulative distribution function (CDF) for the transmitted network bandwidth while running SPECweb at 13000 SUS. This CDF presents the percentage of total time where the transmitted bandwidth was either at or below a certain percentage of peak bandwidth possible. In our case the peak bandwidth is 256 megabytes per seconds (MBPS) due to the bonded Ethernet connection on the testbed (see Section 3.1.3). We monitor the network bandwidth using the *sar* utility at a resolution of one second. We observe that the SPECweb benchmark at 13000 SUS is networking intensive. The benchmark spends 80% of the time consuming more than 80% of the network bandwidth. Moreover, it spends 50% of time consuming more than 85% of the network bandwidth. Through our experiments we also found that the system under test was not able to meet the response time constraints when we increased the SUS beyond 13000. Hence, our experiments use 13000 SUS as 100% load-level for SPECweb.

Appendix C

Other Analysis of Data Serving Workload

C.1 Baseline Measurements for Power and Latency

Here we measure the performance and corresponding power consumption of the Cassandra cluster. The goals are two-fold: (1) to improve our understanding of the relationship between power and performance for a distributed NoSQL data store and (2) to identify any potential for power savings.

Figure C.1 shows the power distribution for the read-only workload and the update-only workload across the entire cluster. The values reported in Figure C.1 are based on the sum of the power consumption from each node in the cluster and averaged over multiple runs. System components other than the processor and memory are represented as *“Others”* [1] in legend of the figure.

The other components of the system consumed a significant portion of the power when idling

[1]The other components, denoted by “Others,” also include the power consumption of the hard disk.

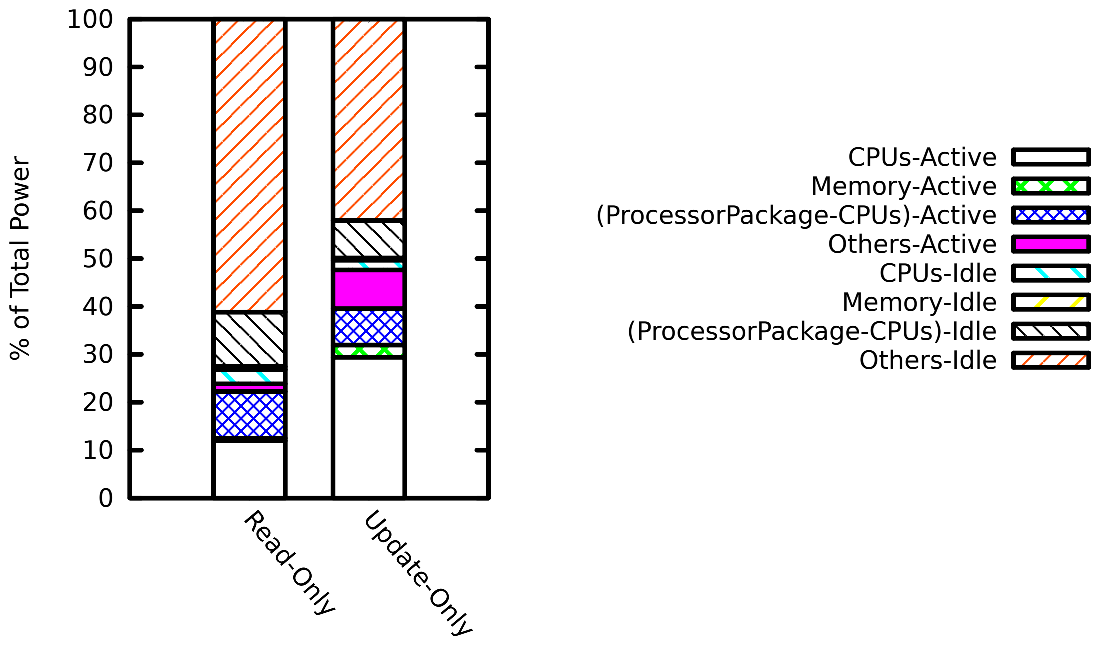


Figure C.1: Component-Level Power Distribution.

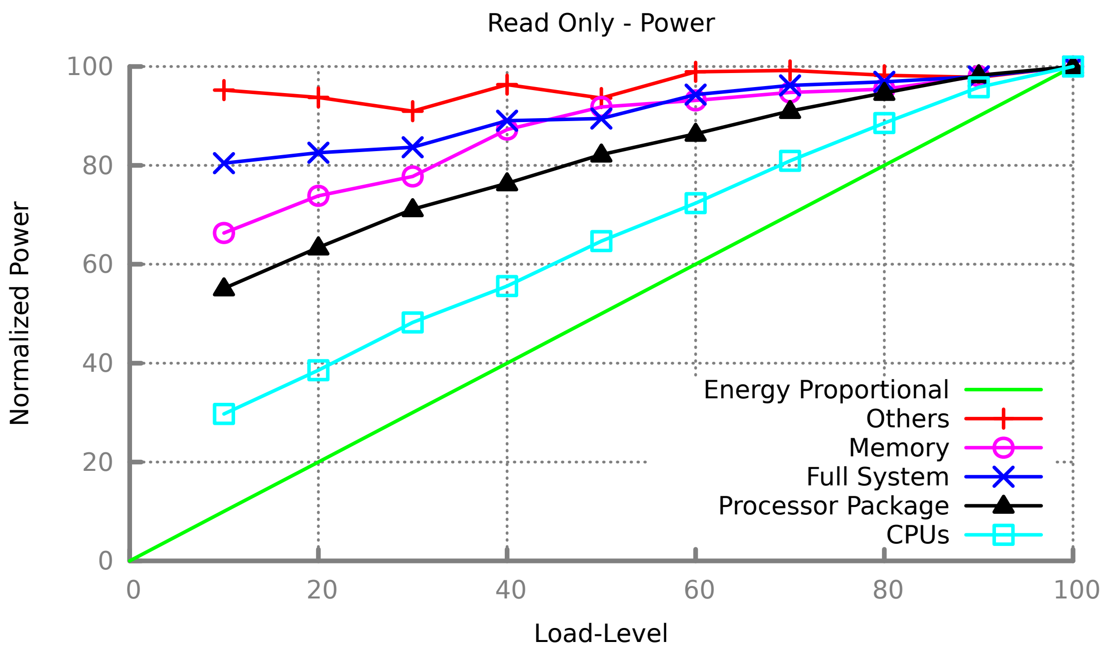
(i.e., 61% and 42% of the total power for the read-only and update-only workloads, respectively). However, they only add 2% and 8% more to the power consumption when the system is under load for the two workload cases, respectively. In contrast, the processor package adds another 22% and 37% to the power consumption distribution for the workloads under evaluation. This shows that the processor contributes significantly to the dynamic power range of the cluster under test. The plot shows only the distribution of power consumption at the highest load on the system (i.e., 100% load-level). However, we are also interested in analyzing the energy proportionality (i.e., the power consumption at different load-levels) of the system, which leads us to Figure C.2.

Figure C.2 shows the normalized power consumed by the system at different load-levels. We normalize power relative to the power consumed at 100% load-level by that component. For example, each value in the CPU power trend is normalized to the power consumed by the CPU at

100% load-level. We also show the energy proportional case for comparison purposes. Several insights can be gleaned from these figures. As evident from the figure, the system exhibits poor energy proportionality for both the workloads as it varies between 80% and 100% of normalized power. The read-only workload, however, exhibits better energy proportionality on the cluster than the update-only workload. For example, the CPUs have a linear increase in power consumption and varies between 30% and 100% for the read-only workload. However, the power consumption of the CPU varies only between 75% and 100% for the update-only workload even though the CPUs consume a higher percentage of the overall power, as shown in Figure C.1. Later in this appendix, we analyze whether existing power-management techniques can help to improve the energy proportionality of the cluster system.

Figure C.3 shows the latency profile for the two workloads. We present the average latency as well as the latencies at the 95th and 99th percentile at different load-levels for each workload. The performance targets (or SLOs) are typically based on either the 95th or 99th percentile rather than by the average. These SLOs are fixed at a particular value by the service provider and do not depend on the load-level of the system. SLOs provide us with the opportunity to trade latency for power under certain load-levels.

If the cluster achieves lower latencies at low load-levels, power-management techniques can be used to improve the efficiency of the cluster by provisioning power or provisioning server resources. For example, if the SLO is set as 160 ms on the 99th-percentile latency for the update-only workload, there exists headroom between the measured latency and the SLO for any load-level less than 90%. We can use this headroom to improve the power consumption of the cluster, thus improving energy proportionality.



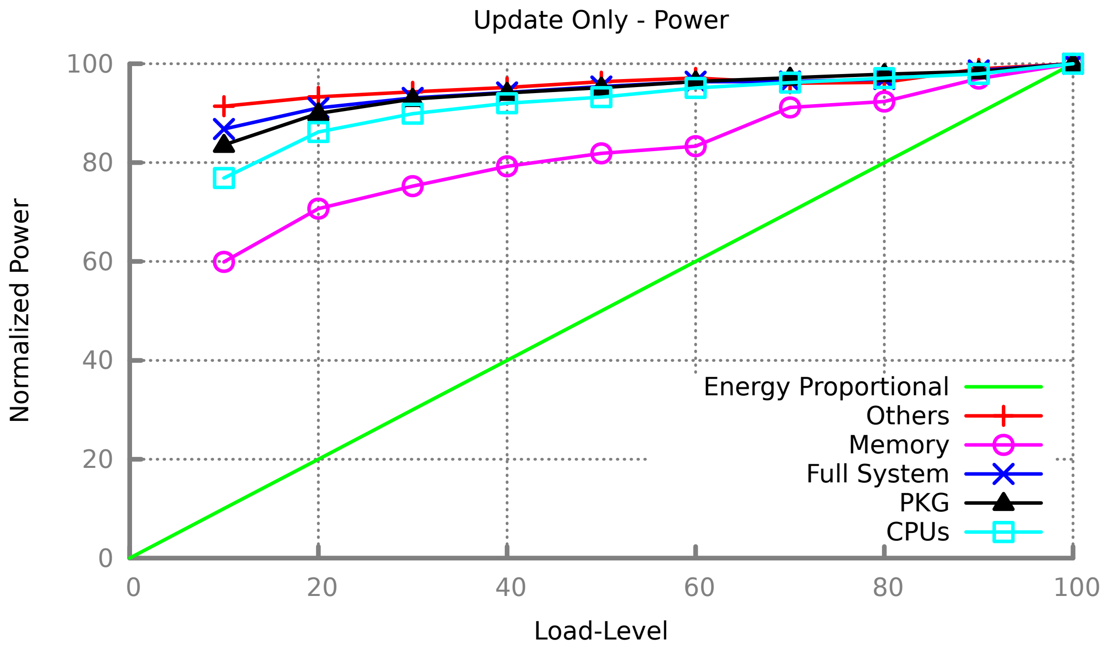
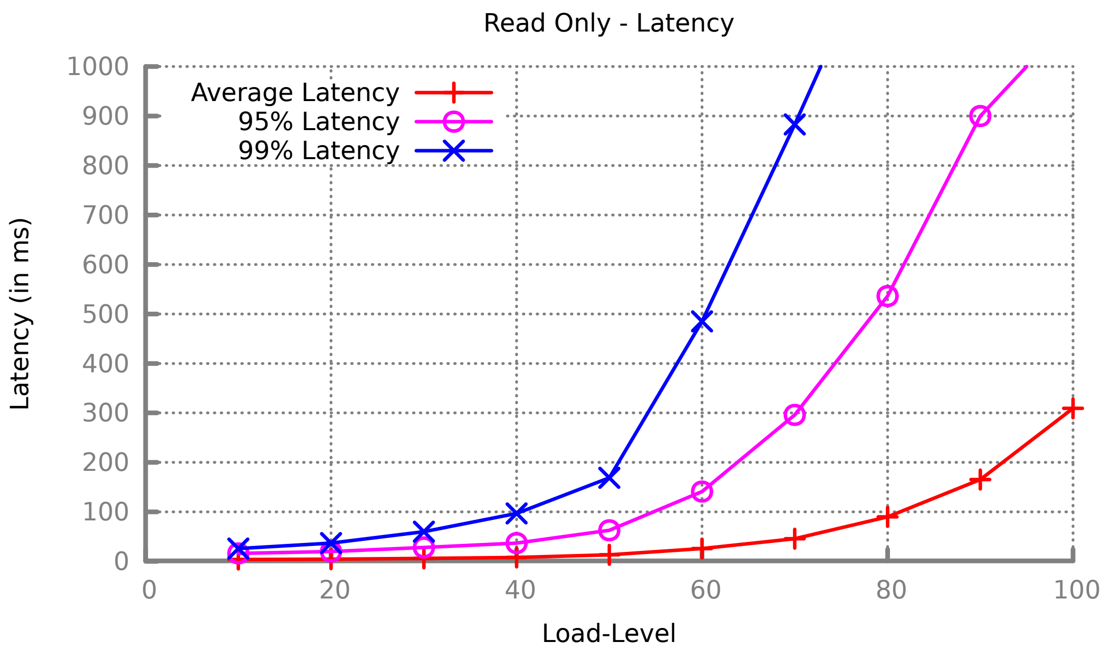


Figure C.2: Energy Proportionality



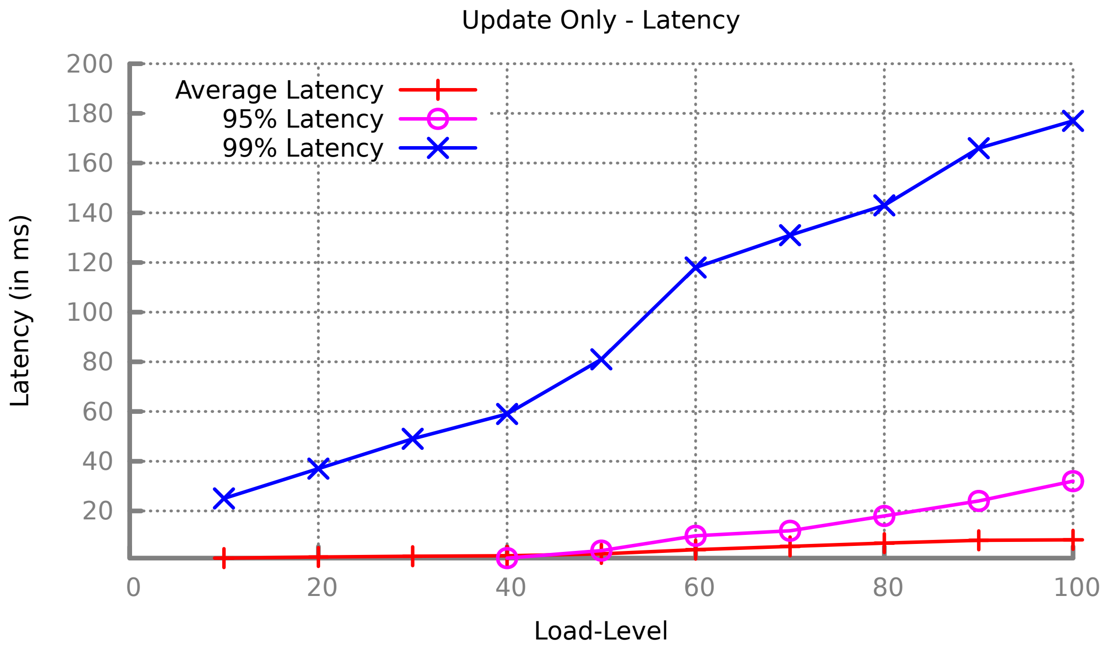


Figure C.3: Latency Profile

C.2 Evaluation of Power-Management Techniques for NoSQL Data Store

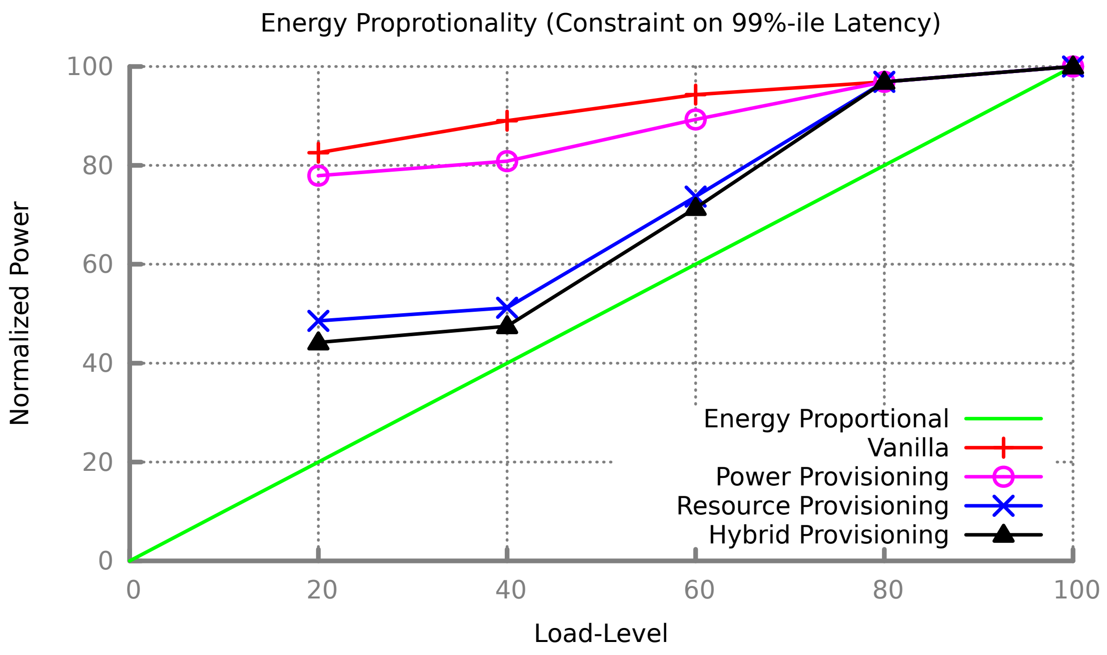
In this section, we evaluate the effect of different power-management techniques on the power consumption and energy proportionality of the Cassandra cluster. In this appendix, we evaluate three different techniques: *power, resource, and hybrid (i.e., power and resource) provisioning*. The power-management techniques are applied while meeting the SLOs. Two different SLOs on latency, one on the 95th percentile and the other on the 99th percentile, are evaluated for each of the workload. For the read-only workload, we fix the SLOs at 600 ms for the 95th-percentile latency and 1000 ms for the 99th-percentile latency. For the update-only workload, the SLOs are fixed at 25 ms for the 95th-percentile latency and 160 ms for the 99th-percentile latency.

For power provisioning, we use the power-limiting interface of RAPL. In this appendix, we use only CPU power limiting as we have shown that it contributes most to the dynamic power range of the system (see Figure C.1). We run the workload at a particular load-level and manually change the CPU power limit in order to find the best power limit for the CPU which satisfies the SLOs. The evaluation of resource provisioning is done by manually hibernating nodes in the cluster. Hibernating nodes saves approximately 40 watts per node. We manually find the optimal number of nodes to run Cassandra to satisfy SLOs at a given load-level. We also evaluate a hybrid version of power and resource provisioning. First, we run the workload on the optimal number of nodes and then find the best possible CPU power limit on those nodes that satisfy the SLOs.

C.2.1 Energy Proportionality

Our main goal in this section is to understand the effects of the different power-management techniques on the energy proportionality of the system. Figures C.4 and C.5 show the effects of different power-management techniques on the energy proportionality of the read-only and update-only workloads, respectively, under different SLO targets. Energy proportionality is improved in every case.

Power provisioning is the least effective technique. However, it still saves power even at low load-levels. Resource provisioning and hybrid provisioning achieve better than energy-proportional operation at certain load-levels for both the workloads. Resource provisioning in certain cases provides higher energy-proportionality improvements when the SLO target is relaxed. For example at 80% load-level in the read-only workload case, better energy proportionality is achieved when the SLO is changed from the 99th percentile to the 95th. In this case, we achieve better energy proportionality because the 95th-percentile SLO can be maintained with only three nodes when compared to the four nodes used for satisfying the SLO for the 99th percentile.



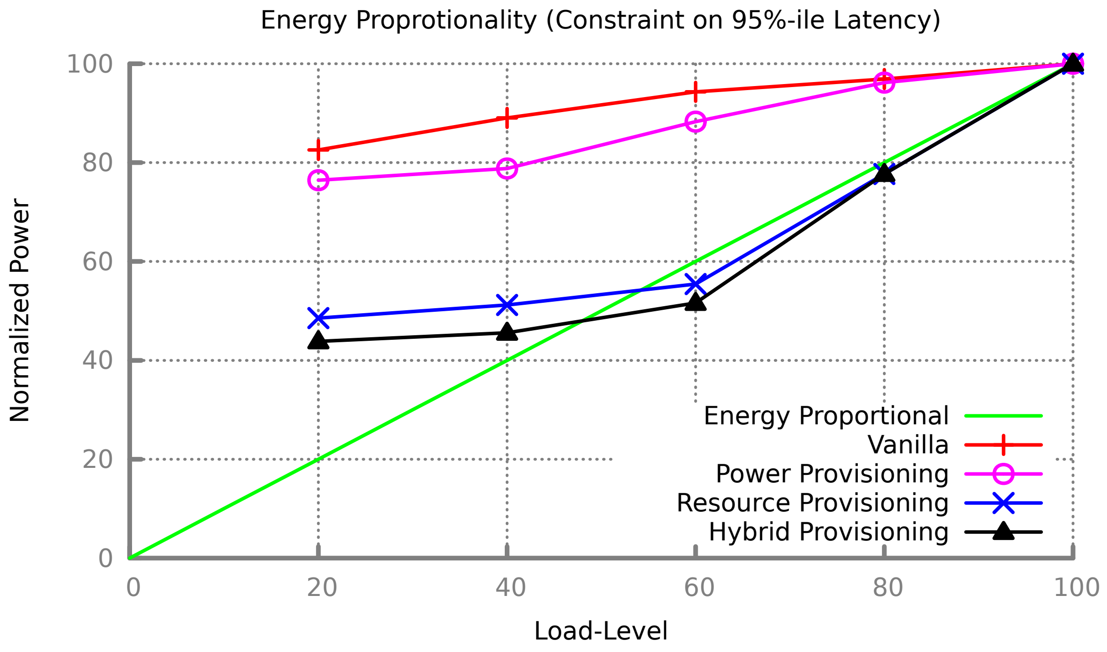
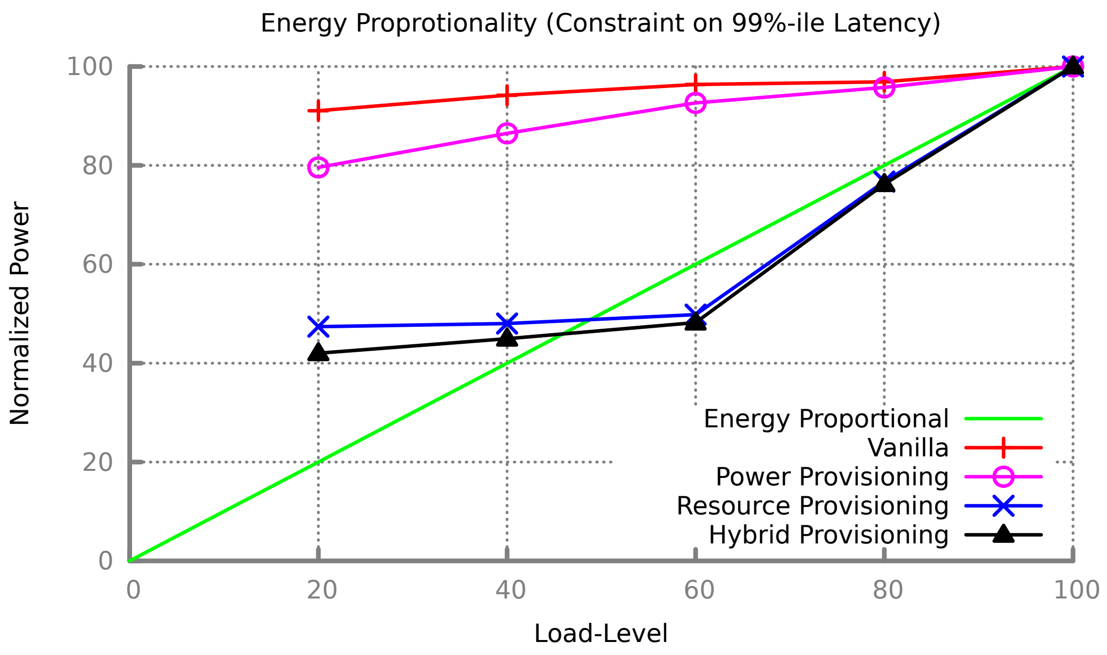


Figure C.4: Read Only Workload - Full System Energy Proportionality



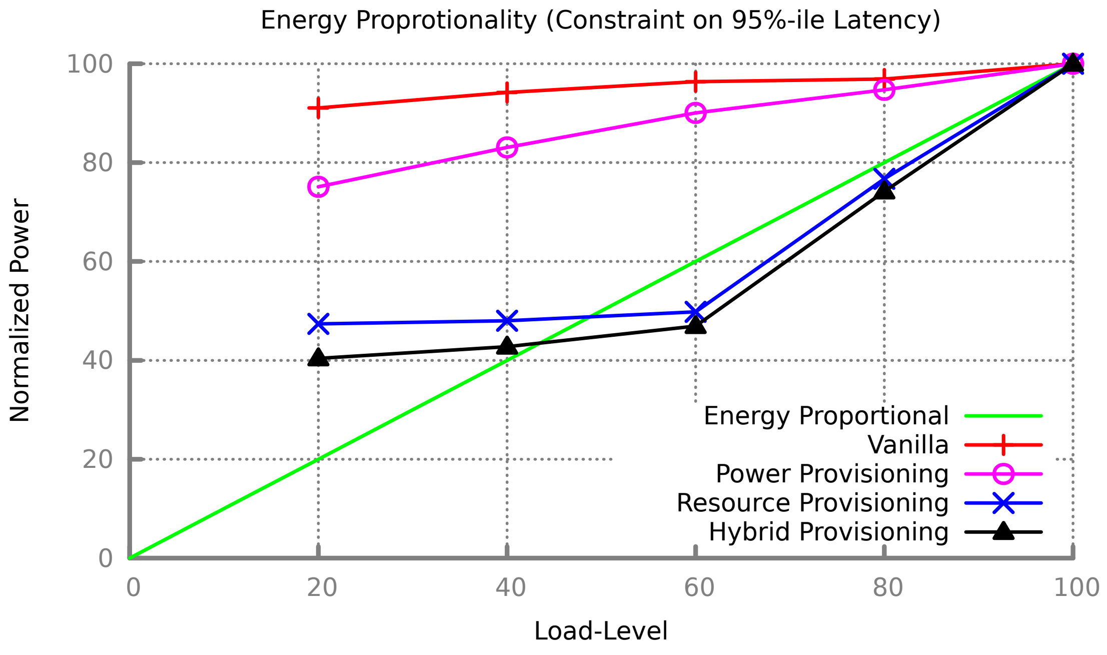


Figure C.5: Update Only Workload - Full System Energy Proportionality

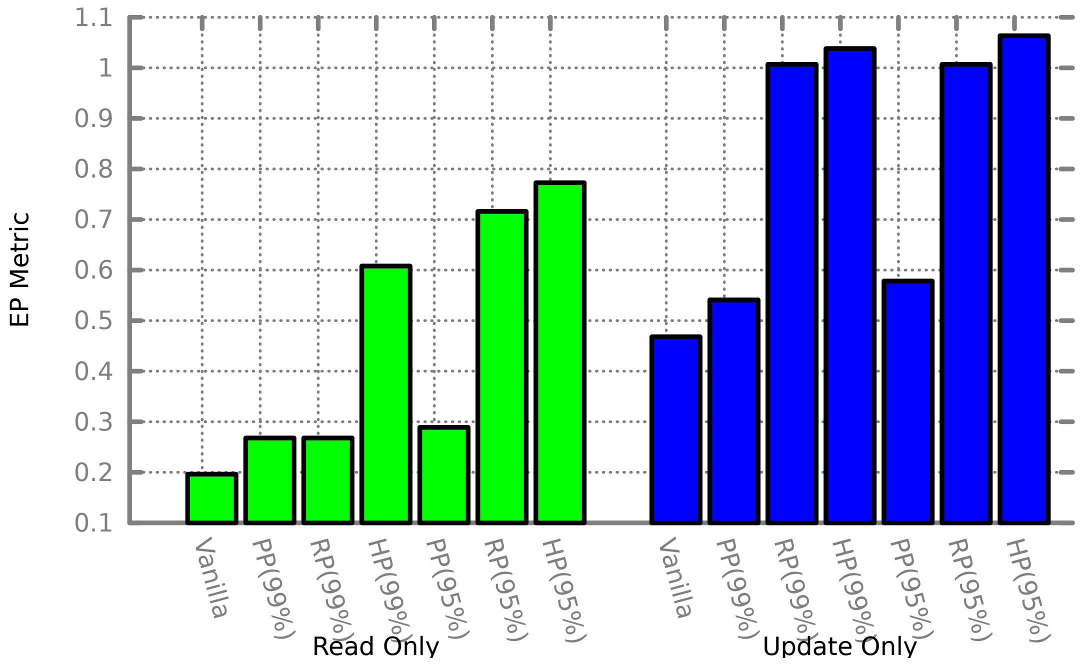
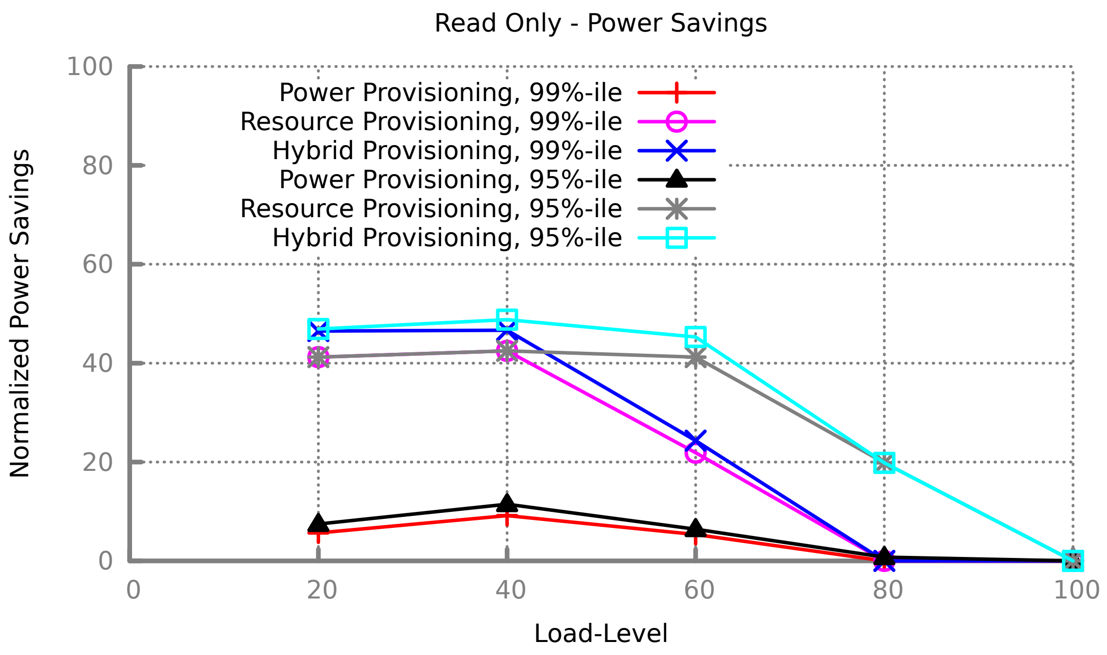


Figure C.6: EP Metric (PP = Power Provisioning, RP = Resource Provisioning, HP = Hybrid Provisoning)

We quantify energy proportionality using the energy-proportionality (EP) metric [47]. Figure C.6 shows the EP metric for the power-management techniques under different SLOs. In general, the power management techniques under evaluation improve the energy proportionality of the update-only workload better than the read-only workload. In certain cases for the update-only workload, EP > 1 is achieved.

C.2.2 Power Savings

Figure C.7 shows the power savings resulting from the different power-management techniques. The savings range from 5% to 45% for the read-only workload and 15% to 55% for the update-only workload. In each case, hybrid provisioning provides the most power savings, but it is only marginal power savings over resource provisioning. We also observe that if the same number of nodes are used, relaxing the SLO target only provides marginal power savings. For example, power savings in the case of power provisioning under both the SLOs for the workloads provide similar power savings.



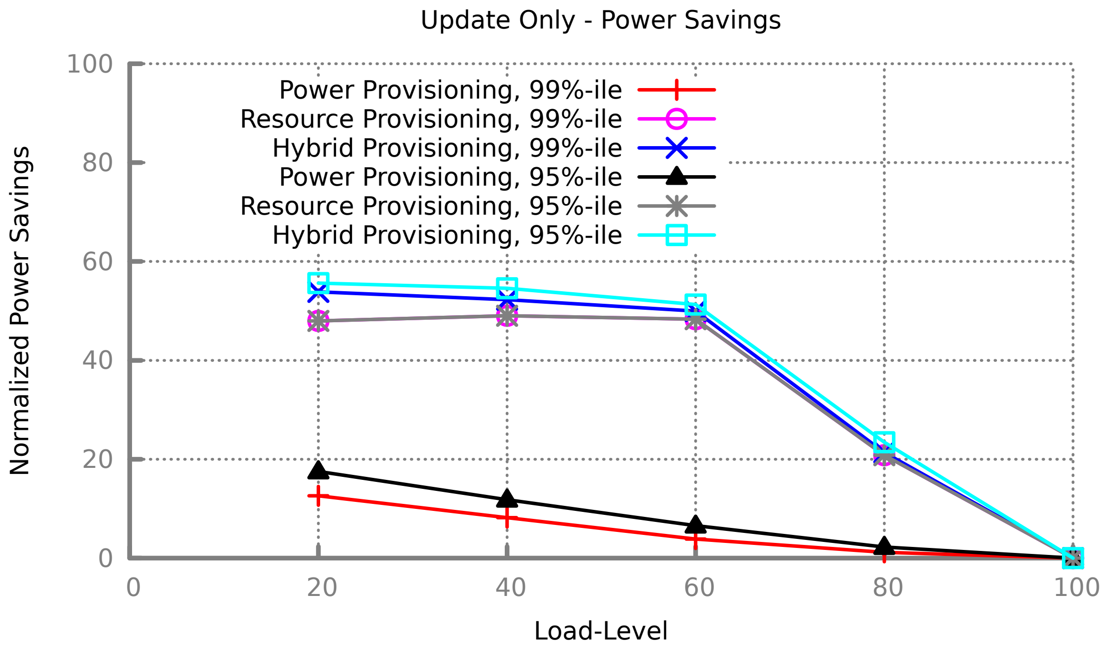


Figure C.7: Power Savings

Bibliography

[1] Apache Cassandra. Available at http://cassandra.apache.org/.

[2] Apache Nutch. http://nutch.apache.org.

[3] Apache Tomcat. http://tomcat.apache.org.

[4] CloudSuite. http://parsa.epfl.ch/cloudsuite/cloudsuite.html.

[5] Faban Toolkit. http://faban.org.

[6] Faban Workload Design. http://faban.org/docs/WorkloadDesign.pdf.

[7] 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.

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

[9] 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.

[10] Memcached. http://memcached.org.

[11] Power Management Architecture of the 2nd Generation Intel Core Microarchitecture, Formerly Codenamed Sandy Bridge. http://www.hotchips.org/wp-content/uploads/hc_archives/hc23/HC23.19.9-Desktop-CPUs/HC23.19.921.SandyBridge_Power_10-Rotem-Intel.pdf.

[12] SPECjbb Benchmark. http://www.spec.org/jbb2005/.

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

[14] SPECpower Benchmark – Benchmarking Methodology. http://www.spec.org/power/docs/SPEC-Power_and_Performance_Methodology.pdf.

[15] SPECpower Benchmark – CCS Design Document. `http://www.spec.org/power/docs/SPECpower_ssj2008-Design_ccs.pdf`.

[16] SPECpower Benchmark – Design Overview. `http://www.spec.org/power/docs/SPECpower_ssj2008-Design_Overview.pdf`.

[17] SPECpower Benchmark – Run Rules. `http://www.spec.org/power/docs/SPECpower_ssj2008-Run_Reporting_Rules.html`.

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

[19] SystemTap. `https://sourceware.org/systemtap/wiki`.

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

[21] Yahoo Cloud Serving Benchmark (YCSB). `https://github.com/brianfrankcooper/YCSB/wiki`.

[22] L. A. Barroso, J. Clidaras, and U. Hölzle. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second Edition. *Synthesis Lectures on Computer Architecture*, 2013.

[23] L. A. Barroso, J. Dean, and U. Hölzle. Web Search for a Planet: The Google Cluster Architecture. *IEEE Micro*, 2003.

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

[25] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda. Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps. In *Proceedings of the International Symposium on Microarchitecture*, MICRO, 2011.

[26] M. Curtis-Maury, A. Shah, F. Blagojevic, D. S. Nikolopoulos, B. R. de Supinski, and M. Schulz. Prediction Models for Multi-Dimensional Power-Performance Optimization on Many Cores. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*, PACT, 2008.

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

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

[29] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini. MemScale: Active Low-Power Modes for Main Memory. In *Proceedings of the Architectural Support for Programming Languages and Operating Systems*, ASPLOS, 2011.

[30] 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.

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

[32] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi. Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS, 2012.

[33] A. Gandhi, M. Harchol-Balter, R. Das, J. Kephart, and C. Lefurgy. Power Capping Via Forced Idleness. In *Proceedings of the Workshop on Energy-Efficient Design*, WEED, 2009.

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

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

[36] A. Lakshman and P. Malik. Cassandra: A Decentralized Structured Storage System. *SIGOPS Operating Systems Review*, 2010.

[37] W. Lang, S. Harizopoulos, J. M. Patel, M. A. Shah, and D. Tsirogiannis. Towards Energy-Efficient Database Cluster Design. *arXiv:1208.1933 [cs]*, Aug. 2012.

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

[39] D. Lo, L. Cheng, R. Govindaraju, L. A. Barroso, and C. Kozyrakis. Towards Energy Proportionality for Large-scale Latency-critical Workloads. In *Proceedings of the International Symposium on Computer Architecuture*, ISCA, 2014.

[40] 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 *Proceedings of the International Symposium on Microarchitecture*, MICRO, 2011.

[41] 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.

[42] D. Meisner, B. T. Gold, and T. F. Wenisch. PowerNap: Eliminating Server Idle Power. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS, 2009.

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

[44] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani. Scaling Memcache at Facebook. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, USENIX NSDI, 2013.

[45] S. Pelley, D. Meisner, P. Zandevakili, T. F. Wenisch, and J. Underwood. Power Routing: Dynamic Power Provisioning in the Data Center. In *Proceedings of the Architectural Support for Programming Languages and Operating Systems*, ASPLOS, 2010.

[46] 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 *Proceedings of the International Parallel and Distributed Processing Symposium Workshops and PhD Forum*, IPDPSW, 2012.

[47] F. Ryckbosch, S. Polfliet, and L. Eeckhout. Trends in Server Energy Proportionality. *IEEE Computer*, 2011.

[48] 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 Cluster*, 2013.

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

[50] B. Subramaniam. Personal Communication with Barry Rountree.

[51] 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.

[52] B. Subramaniam and W. Feng. Enabling Efficient Power Provisioning for Enterprise Applications. In *Proceedings of the International Symposium on Cluster, Cloud and Grid Computing*, CCGRID, 2014.

[53] B. Subramaniam and W. Feng. On the Energy Proportionality of Distributed NoSQL Data Stores. In *Proceedings of the International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, PMBS, 2014.

[54] B. Subramaniam and W. Feng. On the Energy Proportionality of Scale-Out Workloads. Arxiv, 2015.

[55] B. Subramaniam and W. Feng. Towards Energy-Proportional Computing Using Subsystem-Level Power Management. Arxiv, 2015.

[56] B. Subramaniam, W. Saunders, T. Scogland, and W. Feng. Trends in Energy-Efficient Computing: A Perspective from the Green500. In *Proceedings of the International Green Computing Conference*, IGCC, 2013.

[57] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah. Analyzing the Energy Efficiency of a Database Server. In *Proceedings of the International Conference on Management of Data*, SIGMOD, 2010.

[58] V. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore. Measuring Energy and Power with PAPI. In *Proceedings of the International Workshop on Power-Aware Systems and Architectures*, PASA, 2012.

[59] 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.

[60] D. Wong and M. Annavaram. Implications of High Energy Proportional Servers on Cluster-Wide Energy Proportionality. In *Proceedings of the International Symposium on High Performance Computer Architecture*, HPCA, 2014.