No "Power" Struggles: Coordinated Multi-level Power Management for the Data Center

Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, Xiaoyun Zhu

Enterprise Systems and Software Laboratory
HP Laboratories Palo Alto
HPL-2007-194
December 20, 2007*

data center, power management, coordination, efficiency, capping, virtualization, control theory.

Power delivery, electricity consumption, and heat management are becoming key challenges in data center environments. Several past solutions have individually evaluated different techniques to address separate aspects of this problem, in hardware and software, and at local and global levels. Unfortunately, there has been no corresponding work on coordinating all these solutions. In the absence of such coordination, these solutions are likely to interfere with one another, in unpredictable (and potentially dangerous) ways. This paper seeks to address this problem. We make two key contributions. First, we propose and validate a power management solution that coordinates different individual approaches. Using simulations based on 180 server traces from nine different real-world enterprises, we demonstrate the correctness, stability, and efficiency advantages of our solution. Second, using our unified architecture as the base, we perform a detailed quantitative sensitivity analysis and draw conclusions about the impact of different architectures, implementations, workloads, and system design choices.

Internal Accession Date Only Approved for External Publication
To be published and presented at *ASPLOS'08* March 1–5, 2008, Seattle, Washington, USA.

No "Power" Struggles: Coordinated Multi-level Power Management for the Data Center

Ramya Raghavendra[*], Parthasarathy Ranganathan[†], Vanish Talwar[†], Zhikui Wang[†], Xiaoyun Zhu[†]

[*]University of California, Santa Barbara
ramya@cs.ucsb.edu

[†]HP Labs, Palo Alto
{partha.ranganathan, vanish.talwar, zhikui.wang, xiaoyun.zhu}@hp.com

Abstract

Power delivery, electricity consumption, and heat management are becoming key challenges in data center environments. Several past solutions have individually evaluated different techniques to address separate aspects of this problem, in hardware and software, and at local and global levels. Unfortunately, there has been no corresponding work on coordinating all these solutions. In the absence of such coordination, these solutions are likely to interfere with one another, in unpredictable (and potentially dangerous) ways. This paper seeks to address this problem. We make two key contributions. First, we propose and validate a power management solution that coordinates different individual approaches. Using simulations based on 180 server traces from nine different real-world enterprises, we demonstrate the correctness, stability, and efficiency advantages of our solution. Second, using our unified architecture as the base, we perform a detailed quantitative sensitivity analysis and draw conclusions about the impact of different architectures, implementations, workloads, and system design choices.

Categories and Subject Descriptors C.0 [**Computer Systems Organization**], D.4 [**Operating Systems**]

General Terms Algorithms, Design, Management, Measurement, Performance.

Keywords data center, power management, coordination, efficiency, capping, virtualization, control theory.

1. Introduction

Power and cooling are emerging to be key challenges in data center environments. A recent IDC report estimated the worldwide spending on enterprise power and cooling to be more than $30 billion and likely to even surpass spending on new server hardware. The rated power consumptions of servers have increased by 10X over the past ten years [23]. This has led to increased spending on cooling and power delivery equipment. A 30,000 square feet 10MW data center can need up to five million dollars of cooling infrastructure; similarly, power delivery beyond 60 Amperes per rack can pose fundamental issues [23]. The increased power also has implications on electricity costs, with many data centers reporting millions of dollars for annual usage. From an environmental point of view, the Department of Energy's 2007 estimate of 59 billion KWhrs spent in U.S. servers and data centers translates to several million tons of coal consumption and greenhouse gas emission per year. The U.S. Congress recently passed Public Law 109-431, directing the Environmental Protection Agency (EPA) to study enterprise energy use, and several industry consortiums such as the GreenGrid [16] have been formed to address these issues. In addition, power and cooling can also impact compaction and reliability.

In response to the increased importance of this area, there has been a large body of recent work on enterprise power management [2-7][9-14][17][20-22][25-28][31][36]. Given the multifaceted nature of the problem, the solutions have correspondingly focused on different dimensions. For example, some studies have focused on average power reduction for lower electricity costs while others have examined peak power management for lower air conditioning and power delivery costs. Previous studies can also be categorized based on (1) the approaches used (e.g., local resource management, distributed resource scheduling, virtual machine migration), (2) the options used to control power (e.g., processor voltage scaling, component sleep states, turning systems off), (3) the specific levels of implementation – chip, server, cluster, or data center level – hardware, software, or firmware, and (4) the objectives and constraints of the optimization problem – for example, do we allow performance loss? Do we allow occasional violations in power budgets?

While these previous solutions individually address aspects of the enterprise power and cooling problem in isolation, deploying them together has the potential for synergistic interactions and can better address the dynamic and diverse nature of workloads and systems in future enterprises. However, this requires a carefully-designed coordination architecture. The need for federation, the full or partial overlap in the objective functions and the use of the same or interrelated knobs for power control across the different solutions, often at different time granularities, makes this a hard problem. In the absence of such coordination, however, the individual solutions are likely to interfere with one another, in unpredictable, and potentially dangerous, ways.

Several open questions exist for the design of a coordinated solution. How should the overall architecture be designed for individual controllers to interact with each other to ensure *correctness* (no excessive power budget violations), *stability* (no large oscillations), and *efficiency* (optimal tradeoff between power and performance)? How do we combine tracking, capping, and optimization solutions? How do we address the lack of visibility into other controllers and minimize the need to exchange global information? Furthermore, given such a coordinated scenario, there are several implications on the design of the solution. Are all solutions equally important? Do the policies and mechanisms at the individual levels need to be

ASPLOS'08 March 1–5, 2008, Seattle, Washington, USA.

	Solution	Efficiency controller (EC)	Server manager (SM)	Enclosure manager (EM)	Group manager (GM)	VM controller (VMC)
1	**Average/Peak**	Average	Peak	Peak	Peak	Average
2	**Res. management**	Local	Local	Distributed	Distributed	Distributed
3	**Actuator scope**	Local	Local	Local	Local	Global
4	**Time Constant**	millisecs-secs	millsecs-secs	millsecs-secs	secs-mins	mins-hrs
5	**Problem**	tracking	capping	cap/opt	cap/opt	optimization
6	**Implementation**	HW or SW	HW or SW	HW or SW	SW	SW
7	**Actuator**	P-state	P-state	P-state	P-state	consolidation + power off
8	**Input**	res util, power	per-server pwr	per-server pwr	per-server pwr	per-server util

Figure 1: Illustrating the "power" struggle. *The table summarizes five representative, and currently available, power management solutions and their interactions.*

revisited in the context of their interactions with other controllers? How sensitive are the answers to the nature of the applications and systems considered?

To the best of our knowledge, our work is the first to address these questions. We make two key contributions. We propose and evaluate a coordinated architecture for peak and average power management across hardware and software for complex enterprise environments. Our work leverages a *feedback* mechanism to federate multiple power management solutions and builds on a *control-theoretic* approach to unify solutions for tracking, capping, and optimization problems, with *minimal interfaces* across controllers. Simulation results, based on server traces from real-world enterprises, demonstrate the correctness, stability, and efficiency advantages of our solution. Second, we perform a detailed quantitative evaluation of the sensitivity of such a coordinated solution to different architectures, implementations, workloads, and system design choices. Our results illustrate interesting insights and tradeoffs for future enterprises.

The rest of the paper is organized as follows. Section 2 provides the background and defines the problem. Section 3 describes our proposed coordination architecture. Sections 4 and 5 describe the implementation, evaluation methodology, and simulation results. Section 6 discusses extensions and related work, and Section 7 concludes the paper.

2. Problem Statement

2.1 The diversity in power management

Solutions focusing on *average power* optimize the electricity consumed by minimizing the power needed to achieve the required performance. This is typically a *tracking* problem where the consumed power needs to track the resource demands of the applications. Solutions concerning *peak power*, on the other hand, optimize the provisioning of power delivery and cooling in data centers. This is a *capping* problem to ensure that the system does not violate a given power budget. The power budget usually corresponds to the "capacity of the fuse" in power supplies or the heat extraction capacity of the fans and air conditioners. One available leeway in thermal power budgeting is that transient violations of power budgets are allowable, as long as they are bounded. This leverages the observation that thermal failover happens only when the power budget is violated long enough to create enough heat to increase the temperature beyond normal operational ranges. Controlling power in most systems involves changing the *performance* as well. For example, the ACPI industry standard [19] specifies P-states or power states operating at different power-performance tradeoffs. Other options to control power such as using sleep states, or turning systems off also impact the performance. This leads to a potential performance loss with power management. When performance is added as a constraint, especially across a collection of systems, the power management problem becomes an *optimization* problem to ensure that performance loss is minimized and power savings are maximized.

Power management solutions can be implemented in hardware or in software. The key differences are in the *access to information* and in the *time constants*. Typically, the software solutions have more high-level application information and operate at coarser granularities (seconds to hours) whereas the hardware solutions have more access to low-level hardware information and can operate at finer granularities (milliseconds to seconds). Finally, the scope the solution operates at can be limited to a component, a platform, a cluster, or an entire data center. Typically this translates to whether the solution is optimizing a *local* metric or a *global* metric and whether we have a *local* resource management or a *distributed* resource management optimization.

2.2 A "representative" subset of diversity

The above discussion points to four key high-level axes to divide power management solutions – (1) the objectives and constraints, (2) the scope and time granularities, (3) the approach used, and (4) the specific option used to control power. However, the combinatorial space can be quite large. For example, for (1), the solution can optimize average or peak power with or without additional constraints on performance, with or without additional leeway in budget violations. For (2), the solution can be limited to just a processor, an entire server, a blade enclosure (with multiple servers sharing common resources), or a data center. The implementation can be at the hardware, firmware, VM, OS, or application layer with associated differences in the granularity of operation and the access to information. For (3), the different approaches used may include local resource management, distributed scheduling, or virtual machine consolidation. And for (4), the knobs to control power can include voltage and frequency scaling, sleep states, system shut-down, consolidation, etc.

Rather than trying to address this huge space, in this paper, we focus on five individual solutions that are representative for their diversity and are currently available commercially. (Section 6 discusses how our approach applies for other solutions.) An *efficiency controller* (EC) optimizes per-server average power consumption. The controller monitors past resource utilization and adjusts the processor P-state to match estimated future demand. A *server manager* (SM) implements thermal power capping at the server level. It monitors the per-server power consumption and reduces the P-state if a given power budget is violated. An *enclosure manager* (EM) and a *group manager* (GM) implement (thermal) power capping at the blade enclosure and rack or data center levels, respectively. They monitor individual power consumptions across a collection of machines and dynamically re-provision power

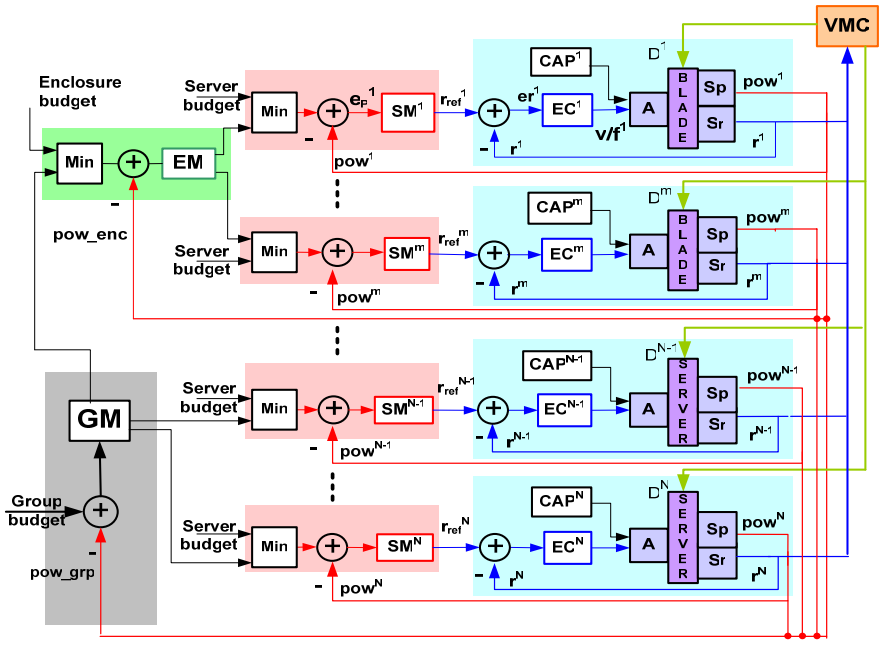

Figure 2: A coordinated power management architecture. *Our proposed architecture coordinates different kinds of power management solutions (multiple levels, approaches, time constants, objective functions, and actuators). Key features of our solution include (a) the use of a control-theoretic core to enable formal guarantees of stability, (b) intelligent overloading of the control channels to include the impact of other controllers, and reducing the number of interfaces and access to global data.*

across systems to maintain a group power budget. These power budgets can be provided by system designers or data center operators based on thermal budget constraints, or determined by high-level power managers. Finally, a *virtual machine controller* (VMC) seeks to reduce the average power consumed across a collection of machines by consolidating workloads and turning unused machines off. Figure 1 summarizes these solutions and illustrates their diversity.

2.3 State-of-the-art: "Power" struggles

The rich diversity in power management discussed above can lead to problems if all the solutions are deployed at the same time. For example, the EC and the SM both operate on the same knob (P-state) but for different metrics. If uncoordinated, the EC can potentially overwrite the SM leading to power budget violations and eventual thermal failover. As another example, in the absence of information about the local power capper's actions, the global power capping algorithm can incorrectly conflict with the local capper leading to increased per-server budget violations or reduced performance. Both are serious correctness issues. As a third illustrative example, if the VMC and group cappers are uncoordinated, the VMC can consolidate more capacity onto a collection of servers than allowed by the group power budget. In addition to excessive performance violations (inefficiency), the VMC can potentially react to the lower utilization (because of power capping) and pack even more workloads onto the server, leading to a vicious cycle and system instability.

As we can see, lack of coordination can lead to problems of correctness, stability, and efficiency. Overall, the issues motivating the need for coordination can be classified as follows – (1) overlap in objective functions – peak versus average, local vs. global, etc., (2) overlap in actuators, (3) different time constants, and (4) different problem formulations. These are summarized in Figure 1. Among these issues, overlap in actuators is the most insidious since it can pose a serious problem of correctness (as in the first two examples above).

However, given the growing challenge from power and cooling, future data centers will likely deploy multiple power management solutions at the same time, and federation[1] of these solutions is desirable. It therefore becomes important to consider a solution that coordinates different power solutions across the various axes of the taxonomy. Two key sets of questions exist in the context of such an architecture. The first pertains to the design of such a coordinated architecture. How should individual controllers interact with each other to ensure correctness, stability, and efficiency? In particular, how do we federate the individual controllers to be aware of one another, but without requiring global knowledge of all the properties at each of the individual controllers? Furthermore, given the dynamism in future enterprise environments, how do we design the solution to respond to changes in the number and nature of controllers participating in the overall architecture, and to changes in the nature of systems and applications deployed?

The second set of questions pertains to the implications of such a unified solution on the design and deployment of individual power management solutions. Are all solutions equally important? Does the coordinated architecture allow for functionality of one controller to be simplified, or even subsumed in another controller, to enable an overall simpler design? Do the policies and mechanisms at the individual level need to be revisited in the context of their interactions with other controllers? How sensitive are the answers to the above

[1] Of course, a centralized solution that implements all individual solutions at one place would solve the challenges discussed, but given the business aspects around different solutions from multiple vendors and the technical aspects around isolation, abstraction, and access to information, we don't believe this approach to be pragmatic.

questions to the nature of applications and systems considered? In this paper, we will answer those questions through design, evaluation and analysis.

3. Proposed Solution

3.1 Functional architecture

Figure 2 shows our proposed coordinated solution. We use a nested structure of multiple feedback controllers at various levels that can be implemented in a distributed fashion. We discuss our functional architecture below. Specific details of the implementation are discussed in Section 4.

Typical feedback loop terminology: Before describing how the individual solutions are designed, let us first consider the basic feedback control loop at the core of the solution (Figure 3). The

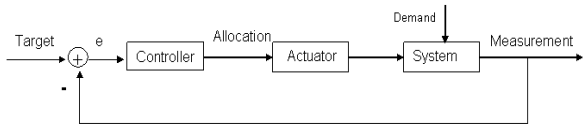

Figure 3: Base feedback control loop. *Our solution overloads the variables and interfaces in the classical control loop above to enable coordination.*

system measures the output *metric* of interest, and compares it to a specified *target or reference.* Based on the error between the two, the controller manipulates some *actuators* in the system so that the measured output value can track the reference. To determine how to operate the actuator, the controller typically includes a *model* that characterizes the input-output relationship of the system being controlled.

Efficiency controller: The innermost level of our solution is the efficiency controller (EC). To implement this controller, we consider the system as a "container" that needs to be used at a desired fraction of its capacity, notated as the reference (r_{ref}) to the controller. This value is compared to the actual utilization of the container (r_i) measured at the Sensor Sr (e.g., operating system calls). Regulating resource utilization around its reference drives the efficiency controller to dynamically "resize the container" by varying the clock frequency through P-states (Actuator A). This allows the power consumed to adapt to the total resource demand the workloads place on the server in real time. For example, if r_{ref} is set to 75%, and a server has a CPU utilization of only 10% due to light workload, the controller determines that there is a tracking error and resizes the container by gradually transitioning the processor from state P0 (highest clock frequency) possibly towards state P4 (lowest clock frequency) leading to higher utilization, and lower power consumption.

Local power capping: Power capping at the server level is implemented as a second controller (SM) nested on the EC. This controller measures the per-server power consumption (Sensor Sp) and compares it with the reference of its power budget. A key aspect of our design is that we use r_{ref} as the actuator rather than directly changing P-states as in a conventional design. In the event of a power budget violation, the controller increases the r_{ref} input to the EC, which in turn responds by going to lower P-states, enabling the power budget to be met. Using r_{ref} as the communication channel between the EC and the SM reduces the need for global data structures or centralized arbitrators. Working in a reactive way, this approach may lead to transient budget violations, but the controller bounds the time on such violations. As discussed earlier, this is acceptable in a thermal power capper. An optional electrical power capper (CAP) can be implemented in parallel to the EC as shown in Figure 2.

Enclosure and group power capping: The enclosure manager (EM) implements enclosure-level power capping. For each epoch, the EM controller monitors the total power consumption of the blade enclosure and compares it with an enclosure-level power budget. Based on the comparison, the controller assigns power budgets for the next epoch to all the individual blades in the enclosure. The SM controller in each blade uses the minimum of the power budget recommended by the EM and its own local power budget as its input reference. The actual division of the total enclosure power budget to individual blades is policy-driven and different policies (e.g., fair-share, FIFO, random, priority-based, history-based) can be implemented. Essentially, the communication between the layers happens through the power budget settings and the measurement of the consumed power.

The group-level power capping, implemented by the group manager (GM), works fairly similarly, but at either the rack level or the data center level, and with different time constants. The actual power consumption of the group is compared to the group power budget, based on which the power budgets are assigned to all the next-level servers and blade enclosures. As before, within the SM and the EM, the minimum is chosen between the GM's recommendation and the local budget values.

Virtual machine controller: The final element of our architecture is the virtual machine controller (VMC). It reads as input the resource utilizations of the individual VMs (Sensor Sr) and implements an optimizer that creates a new VMs-to-servers mapping to minimize the aggregate power for the whole rack or data center. Given that the new mapping affects the demand (D) that the other controllers see, there is already one implicit feedback channel for coordination. However, two other key changes are needed in the context of the coordinated solution.

First, the resource utilization values read by the VMC need to be adjusted for local power management. For example, two servers with 100% utilization are not comparable if one of them is at the highest power state and the other is at a lowest power state; the latter is a potential candidate for consolidation while the former is not. We address this by having the VMC consider the *real* utilization instead of the *apparent* utilization. Simple models (such as those in Section 4) can be used to translate apparent utilization to real utilization when the power state is known.

Second, the VMC controller needs to be aware of the approximate budget caps at the various levels. Otherwise, a conventional design can aggressively pack workloads onto a server, which in turn can compromise the statistical load variations that the SM, EM, and GM expect, leading to more aggressive performance throttling. On the other hand, given the saturating nature of resource utilization metrics, the throttled performance can be misinterpreted by the VMC as extra space for consolidation, leading to a vicious cycle. We address this problem by having the VMC (1) be aware of the approximate power budgets at the various levels and use them as constraints in its optimization, and (2) be aware of power budget violations at individual levels and use them to vary the aggressiveness of consolidation. Getting information on the former is fairly

Level	Changes to enable coordination
EC	Expose API to SM to change r_ref
SM	Expose API to EM and GM to change power budget
EM	Expose API to GM to change power budget Expose power budget violations to VMC
GM	Expose power budget violations to VMC
VMC	Use "real utilization"; use power budgets as constraints; explicit feedback to violations

Figure 4: Changes to individual controllers for coordination. *Given our overloading of classical control interfaces, our solution only requires a few explicit changes for coordination.*

straightforward – either machine specifications or approximate estimates can be used. For the latter, we require the individual capping controllers to expose information on their power budget violations externally. This is still reasonable, and can be done by extending current CIM models exposed through DMTF interfaces [8]. (An alternate approach is to determine a proxy for the power budget violations using P-states and performance violations, but this approach is likely to have more hystereses compared to using CIM interfaces.)

3.2 Discussion

A common guiding principle in our design is to enable coordination, wherever possible, by connecting the actuation at one layer to the inputs at another layer. This allows the feedback controller to react to (and learn from) interactions across controllers, (e.g., through the changes of its reference value), the same way as it would react to changes in workload behavior. This enables several benefits.

Minimal interfaces: First, it allows us to minimize the number of explicit changes in the individual controllers for coordination. Figure 4 summarizes the changes needed to enable coordination for the traditional implementations of the individual controllers discussed in Figure 1. As we can see, fairly minimal interface changes are required. This avoids the performance issues around global information exchange or the availability issues around a centralized arbitration model.

Formal rigor: Second, the same mathematical analysis that control theory enables for stability and performance in the face of changing workload demand can be used in the context of interacting controllers. Space and scope constraints prevent us from providing a full mathematical analysis of our architecture, but Appendix A sketches an illustrative proof for stability in one example scenario for one set of controller algorithms and system model assumptions. Such analysis can also be used to tune and bound the gain parameters of the individual controller equations.

Flexibility: Our architecture is also flexible and allows different deployment scenarios and works well with the dynamic nature of enterprise data centers. Changes to workload behavior, changes to system models, changes in controller policies, changes in time constants, etc. can all be accommodated.

Extensibility: The five power management solutions we consider in this paper represent a large class of existing approaches. However, the architectural principles we use enable our design to be easily extended to other classes of controllers and other specific implementations. Section 6 provides additional details on some such extensions.

Federation: Our approach to connecting control parameters across individual solutions has the side benefit of providing better federation in the presence of different time constants and

Metrices and knobs		Notation	Base value
Server	static power budget	*CAP_LOC*	10% off server max
	dynamic power budget	*cap_loc*	tuned by EM or GM
	power consumption	*pow*	measured for SM/EM/GM
	target utilization	*r_ref*	tuned by SM
	measured utilization	*r*	measured for EC
	P-states	*p0, p1, …*	p0,…,p4, tuned by EC
	desired clock frequency	*f*	Hz
	quantized frequency	*f_Q*	[1G, 833M, 700M, 600M, 533M] Hz
	performance	*perf*	work done
Enclosure	static power budget	*CAP_ENC*	15% off enclosure max
	dynamic power budget	*cap_enc*	tuned by GM
	power consumption	*pow_enc*	measured for EM and GM
Group	power budget	*CAP_GRP*	20% off group max
	power consumption	*pow_grp*	measured for GM
Virtual Machine	virtualization overhead	α_V	10% of VM utilization
	migration overhead	α_M	10% of VM utilization
	constraints buffers	*b_loc, b_enc, b_grp*	tuned based on budget violations
Workload	number of workloads	*n*	180 enterprise traces
	demand for capacity	*D*	in utilization
	placement on servers	*X*	matrix with 0/1 elements
System Property	number of servers	*m*	180
	number of enclosures	*l*	20
	relationship between servers & enclosures	*M*	matrix with 0/1 elements
Control Interval	efficiency control (EC)	*T_ec*	1
	server manager (SM)	*T_sm*	5
	enclosure manager (EM)	*T_em*	25
	group manager (GM)	*T_grp*	50
	VM Controller (VMC)	*T_vmc*	500
Controller Gain	efficiency control (EC)	λ	0.8
	server manager (SM)	β_loc	1

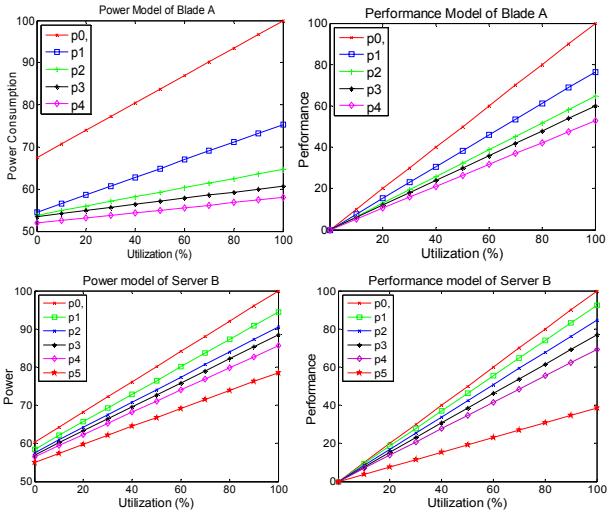

Figure 5: Design parameters and implementation assumptions. *The individual solutions, the system models, and the workload traces, each have multiple variable parameters, leading to a combinatorial explosion in the design space. The last column highlights the base values for all the parameters involved. Section 4 discusses the rationale for these baselines, and Section 5 examines sensitivity to most of these parameters.*

different granularities of information. For example, the solutions that operate less frequently and have access to larger windows end up providing first-order guidelines for actuation that are in turn refined by other controllers that operate more frequently.

4. Implementation and Evaluation

Figure 5 summarizes key elements of our assumptions and introduces terminology for the discussion below. There is a

combinatorial explosion in the design space from the choices around the controller implementations and their tunable parameters, and the choices of the systems and workloads. Below, we first discuss our controller implementations and then discuss our evaluation methodology and the configurations we evaluate.

4.1 Implementation of the controllers

Space constraints prevent us from a detailed exposition of the formulation of the various controllers. Figure 6 provides a mathematical summary that we briefly describe below.

Power/performance models: A key element of our implementation is the use of performance-power models based on CPU utilization. For each system, the models are calibrated on the actual hardware by running workloads at different utilization levels and measuring the corresponding power and performance (in percentage of work done). We then use linear models obtained through curve-fitting in our simulation. The linear models are shown in Equations (Models) in Figure 6, where the index p represents the P-state. Figure 5 shows these models visually for the two real systems we studied (detailed more in Section 4.3). Note that these models also highlight the monotonicity in variation for the dependence between the various parameters (utilization, performance, power, and frequency), that are key assumptions to the design of the controllers discussed next.

Efficiency Controller and Server Manager: The efficiency controller as shown in Equation (EC) polls the average resource utilization, and tunes the clock frequency based on an integral control law, where the change of the frequency is proportional to the error in utilization. The integral gain that determines the aggressiveness of the controller is self-tuning, and stability is guaranteed by posing an upper bound on the scaling parameter λ. Local power budget is guaranteed by the server manager, by tuning the utilization target of the EC as shown in Equation (SM). The utilization target r_{ref} is increased when the measured power consumption exceeds the local power budget *cap_loc*. As discussed earlier, this in turn causes the EC to reduce the clock frequency lowering the power consumed. Similarly, controller stability can be guaranteed by imposing an upper bound on the gain parameter β_{loc}, which can be computed from given power and performance models. We set a lower bound of 75% on r_{ref} to ensure reasonably high resource utilization in the server even when the power consumption is below the local budget. (See Appendix A for a stability proof of the EC and the SM.)

Enclosure and Group Manager: The enclosure or the group manager operates similarly to enforce the enclosure-level or group-level power budget. Equations (EM) and (GMs) show the implementation of a proportional-share policy. In each interval, the *overall* power budget is reallocated to the components of the enclosure/group, proportional to its power consumption in the last interval. This simple policy can guarantee a fair share of the budget among the components, and can adapt the allocations to the changing demands in the individual components.

VM Controller: In every epoch, the VM controller solves a constrained optimization problem as described in Equations (VMCs). Specifically, the decision variable is a matrix "X" that maps *n* VMs to *m* servers. The goal is to minimize an objective function that includes the total power consumption and the migration overhead (weighted by the term α_M in Equation (1)) while meeting server capacity constraints (Equation (2)) as well as local, enclosure and group level power budget constraints (Equations (3-5)). To tune the aggressiveness of consolidation, buffers of the power budgets are tuned based on feedback on budget violations in the three levels respectively. Many algorithms are available to solve this 0-1 integer program. In our evaluation, we use a greedy bin-packing algorithm to search for a new placement solution that satisfies all the constraints, which is an approximation of the optimal solution.

As we can see from the discussion above, even with five controllers, the implementation can get fairly complicated with a lot of decisions needed at each level. Figure 5 summarizes the baseline values that we used for this paper for the other parameters.

4.2 Evaluation Methodology and Metrics

Challenges: Ideally, we would like to evaluate our coordinated solution at the data center level in a real implementation. However, this is impractical for several reasons – (1) It is hard to get access to a data center or a sufficiently large collection of machines; (2) Such a collection needs to be fully populated with relatively new servers with support for multiple power states and only a few systems can be studied; (3) All the individual controllers need to be set up and tuned. In addition to the effort needed, this allows only implementations specific to the idiosyncrasies of the systems considered; (4) Even if we did all this, we would need to set up the test bed with complex enterprise applications and exercise them to model real-world usage. The alternate approach of using full-system simulation (e.g., M5, Simics, GEMS) suffers from drawbacks (3) and (4) above, and additionally, simulation speeds and complexities of modeling clusters make this impractical.

Utilization-based large-scale simulation: Given these challenges, in this paper, we use a trace-driven simulation approach for data center environments [27][29]. This approach uses real-world traces from actual enterprise deployments to drive individual server simulations. High-level models like those in Figure 5 are used to correlate resource utilization and the impact of changing specific actuators to system metrics like power and performance. This approach enables the workload behavior and system characteristics to be modeled expediently while allowing detailed evaluation of tradeoffs at the policy and system-parameter levels. Previous studies have used and validated this approach in the context of individual power management solutions (including some considered in this paper) [27][28].

Metrics: In this paper, we only report aggregate power savings, performance loss, and power budget violations at the server, enclosure and group levels as the metrics to evaluate the architecture. The metrics do not include peak power savings since they are used as configuration parameters for the SM, EM, and GM, in the form of power budgets at the various levels. For example, 20-15-10 indicates peak power savings of 20%, 15%, and 10% at the group, enclosure, and local levels, respectively. No queuing process is assumed when the demand of a workload exceeds the capacity. So when the workload demand is increased, or the capacity of the server is reduced due to power capping, performance loss could happen as the excessive demand is not carried over.

4.3 Evaluation Parameters

Figure 5 summarizes the baseline values of all the parameters used in our evaluation.

Workloads: The advantage of our methodology is that it allows us to use actual utilization traces from real-world enterprises. We specifically consider 180 traces representing individual server utilization from nine different enterprise sites for several classes of individual and multi-tier workloads (database servers, web servers, e-commerce, remote desktop infrastructures, etc). To better study the variability in workloads, we study four mixes – one incorporating all the 180 workloads (`180`), and others focusing on specific mixes of 60 workloads (`60L, 60M, 60H`). Most of our workload traces, as is common with most real-world deployments, show relatively low utilization (15-50% in most cases). To better illustrate more resource-intensive workloads, we created "synthetic" workloads (`60HH, 60HHH`) that stacks multiple workloads from our real-world traces to create higher utilization.

Systems and virtual machines: We study two different kinds of enterprise systems – a low-power blade server, Blade A, and an entry-level 2U server, Server B. The processor of Blade A has 5 P-states, with frequencies of 1GHz, 833MHz, 700MHz, 600MHz, 533MHz. The processor of Server B has 6 p-states, with frequencies 2.6GHz, 2.4GHz, 2.2GHz, 2.0GHz, 1.8 GHz and 1.0GHz. The performance-power models for these are shown in Figure 5. We assume that the baseline is also virtualized. For virtual machine migration, we assume a pre-copied migration process [34] and model the migration overhead as 10% performance loss during the migration process. We also study the impact of varying this.

$$(Models): pow = g_p(r) = c_p r + d_p, \quad p = 0, 1, 2, \ldots$$
$$perf = h_p(r) = a_p r, \quad p = 0, 1, 2, \ldots$$

$$(EC): f(k) = f(k-1) - \lambda f_Q(k-1) r(k-1) / r_{ref} \; (r_{ref} - r(k-1)).$$

$$(SM): r_{ref}(\hat{k}) = r_{ref}(\hat{k}-1) - \beta_{loc}(cap\_loc - pow(\hat{k}-1)).$$

$$(EM): cap\_loc = \min(CAP\_LOC, \frac{pow\_loc}{pow\_enc} \times cap\_enc).$$

$$(GMs): cap\_enc = \min(CAP\_ENC, \frac{pow\_enc}{pow\_grp} \times CAP\_GRP).$$

$$cap\_loc = \min(CAP\_LOC, \frac{pow\_loc}{pow\_grp} \times CAP\_GRP).$$

$$(VMCs)$$

$$(1) \quad \min \sum_{i=1}^{m} pow_i + \sum_{i=1}^{m}\sum_{j=1}^{n} \alpha_M \left| X_{ij} - X_{ij}^0 \right|$$

$$(2) \quad s.t. \sum_{j=1}^{n} X_{ij} r_j (1 + \alpha_V) \le \bar{r}$$

$$(3) \quad pow_i \le (1 - b\_loc) CAP\_LOC_i, \quad i = 1, 2, ..., m$$

$$(4) \quad \sum_{i=1}^{m} M_{iq} pow_i \le (1 - b\_enc) CAP\_ENC_q, \quad q = 1, 2, ..., l$$

$$(5) \quad \sum_{i=1}^{m} pow_i \le (1 - b\_grp) CAP\_GRP$$

$$(6) \quad \sum_{i=1}^{m} X_{ij} = 1, \quad j = 1, 2, ..., n$$

$$(7) \quad X_{ij} = \{0,1\}, \quad i = 1, 2, ..., m, \quad j = 1, 2, ..., n$$

Figure 6: Mathematical formulation of various controllers.

Cluster/Datacenters: For the 180-workload evaluation, we assume a cluster of 180 servers. This is organized as six 20-blade enclosures and sixty individual servers. For the 60 workload evaluations, we assume a cluster of 60 servers: two 20-blade enclosures and twenty individual servers.

Power budgets: We study three different kinds of power budget values – (1) 20-15-10 representing group, enclosure, and local power budget caps that are respectively 20%, 15%, and 10% off from their maximum possible power consumption, (2) 25-20-15 representing caps that are 25%, 20%, and 15% off their maximum possible power consumption, and (3) 30-25-20 representing caps that are 30%, 25%, and 20% off their maximum.

Architectural alternatives: We study sensitivity of the architecture through a few alternatives, for instance, the time constants. In the baseline, the constants of EC/SM/EM/GM/VMC are set to 1/5/25/50/500 respectively. Other alternatives include variants of the models with different idle power, p-state groups, different coordination architectures, different policies, etc. These are detailed in the discussion of results in Section 5.

5. RESULTS

We next discuss the evaluation results. We first present results showing how coordination can improve correctness and efficiency vis-à-vis an uncoordinated architecture and then discuss the impact of different architectures, implementations, system design choices, and workloads.

5.1 Base results

In the first set of experiments, we use a system where no controllers for power management are turned on as the *baseline*, and compare two distinct solutions - (1) our proposed *coordinated* architecture, using the base parameter values in Figure 5; (2) an *uncoordinated* solution where the five individual power management solutions work independently of one another, as described in Section 2.2. Figure 7 shows the results for both the coordinated and the uncoordinated solutions as they are compared against the baseline results. Four configurations are included, representing two types of systems and two sets of workloads[2]. For each configuration, we present a family of four bars – three bars for power budget violations, at the group, enclosure, and local levels, one bar for performance degradation. To visually illustrate the negative ramifications of budget violations and performance loss, we show these as negative numbers.

Benefits from coordination: The top-left graph in Figure 7 shows the results for the base 180-server configuration for Blade A. Compared to the baseline, our coordinated solution achieves a 64% reduction in power consumed (not graphed), translating to savings in electricity costs, with negligible (3%) performance degradation and (5%) power budget violations, as seen in the Figure. Recall that this configuration has additional savings of 10%, 15%, and 20% in the peak power budgets at the local, enclosure, and group levels which translate to capital savings for the cooling equipment. In comparison, the uncoordinated solution results in greater performance loss (12%) and higher power budget violations (7%). These observations are consistent across the four scenarios shown in Figure 7, and are more

[2] The discussions in this paper represent simulations of more than 800 individual configurations. In the interests of space, we will talk to the trends that we saw in the overall data, but plot only a subset. More data is available in [26].

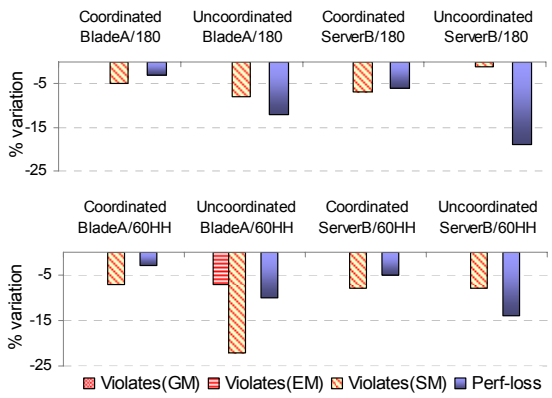

Figure 7: Results. *The figure presents a comparison of an uncoordinated deployment and our proposed coordinated solution for four different configurations. All results are normalized to a baseline where no controllers for power management are turned on. The three left bars show violations in group, enclosure, and server power budgets; the last bar shows performance loss.(Note that empty bars mean no violations in the GM or EM levels.) In general, the uncoordinated architecture has higher performance degradation and power budget violations.*

pronounced in the bottom two scenarios with high activity workloads.

Though these results illustrate the correctness and efficiency benefits of the coordinated solution relative to the uncoordinated solution, the results are not as dramatic (it is hard to graphically show a thermal failover) because of the inherent randomness in the uncoordinated controller and the relatively low average utilization in our traces. As additional validation, we implemented a simple prototype implementation of an uncoordinated deployment of the EC and SM on a server in our lab, and even with one machine, over sustained high loads, the uncoordinated solution went into thermal failover.

Variation for different systems: Figure 7 also illustrates the sensitivity to different system models. As discussed earlier, Server B has 6 P-states relatively uniformly clustered, but with a smaller range in power, compared to the five non-uniformly clustered, but higher range, P-states of Blade A. This typically manifests itself in reduced absolute power savings results for Server B compared to Blade A. It indicates that the range of power control is likely more important than the granularity of control for these configurations.

Variation for different workloads: As discussed earlier, in addition to the 180-workload configuration, we study other workload sets with different levels of activity. The benefits from coordination are qualitatively similar for all classes of workloads. However, as one would expect, the actual power savings for the low utilization workload relative to the baseline is higher than that for the high utilization workload while the relative improvements over the uncoordinated solution is higher with higher utilizations.

5.2 Architectural Choices

This section seeks to answer interesting questions on the relative importance of various controllers in the context of a coordinated architecture and the impact of specific interfaces.

VM migration versus Local Power Control: Figure 8 summarizes the power savings for Blade A and Server B running the 6 workload types discussed earlier. For each configuration, three bars are shown representing (1) the coordinated solution, (2) NoVMC where the VM controller is turned off, and (3) VMCOnly - where only the VMC controller is turned on. As the results show, for our base system models and workloads, most of the average power reductions are from the VMC controller. For example, for the 180-workload configuration, on Blade A, the power savings for Coordinated, NoVMC, and VMCOnly are 64%, 23% and 48%, respectively. The Server B configuration, with its limited P-states support, gets equivalent savings of 57%, 4%, and 54% respectively. An interesting trend is seen as workload utilization is increased. Though the power savings percentages decrease, a greater fraction of the savings now comes from the local power management compared to the VM consolidation. This tracks our intuition that benefits from VM consolidation will decrease if the base workloads have high utilization. The contribution of both EC and VMC to power savings at different operating points highlights the importance of a coordinated solution where both approaches are deployed. In all the scenarios, the coordinated solution continues to behave better compared to the uncoordinated one.

Coordination alternatives: Figure 9 presents a table summarizing the budget violations, performance loss, and power savings for Blade A and Server B for five other alternative coordination solutions with one or more of the interfaces in Figure 4 disabled. The results show that each one of these alternative solutions suffers from some drawbacks in terms of increased performance loss, reduced power savings, or increased budget violations. The drawbacks get exacerbated with changes in system configurations (not shown in table). This indicates that each aspect of our proposed solution is important to ensure general-purpose applicability. The results also illustrate the drawbacks with piecemeal naïve coordination policies and reinforce our arguments earlier for a carefully-designed coordination architecture.

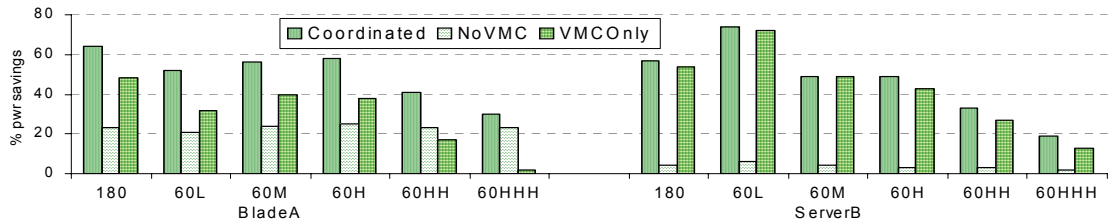

Figure 8: Isolating impact of different controllers. *For our base systems, in general, the VMC is responsible for a larger fraction of the power savings.*

System under control	Power violations GM	EM	SM	perf loss	pwr save
Blade A					
Coordinated	0	0	-5	-3	64
Uncoordinated	0	0	-8	**-12**	72
Coordinated, appr util	0	0	-3	-2	**56**
Coordinated, no feedback	0	0	**-7**	-4	69
Coordinated, no budget limits	0	**-5**	**-23**	**-8**	76
Uncoordinated, min Pstates	0	0	0	**-13**	71
Server B					
Coordinated	0	0	-7	-6	57
Uncoordinated	0	0	-1	**-19**	63
Coordinated, appr util	0	0	-3	-3	**44**
Coordinated, no feedback	0	0	**-13**	-7	66
Coordinated, no budget limits	0	**-15**	**-18**	**-12**	72
Uncoordinated, min Pstates	0	0	0	**-19**	50

Figure 9: Characterizing different coordination interfaces. *The results show that each one of the assumptions made in our proposed coordination architecture is important.*

System under control	Power violations GM	EM	SM	perf loss	pwr save
Blade A					
Coordinated 20-15-10	0	0	-5	-3	64
Coordinated 25-20-15	0	0	-4	-3	58
Coordinated 30-25-20	0	0	-2	-2	46
Uncoordinated 20-15-10	0	0	**-8**	**-12**	72
Uncoordinated 25-20-15	0	0	**-16**	**-12**	72
Uncoordinated 30-25-20	0	**-11**	**-28**	**-13**	73
Server B					
Coordinated 20-15-10	0	0	-7	-6	57
Coordinated 25-20-15	0	-1	-6	-7	52
Coordinated 30-25-20	0	-9	-6	-8	48
Uncoordinated 20-15-10	0	0	-1	**-19**	60
Uncoordinated 25-20-15	0	0	-4	**-18**	61
Uncoordinated 30-25-20	0	**-23**	**-7**	**-17**	61

Figure 10: Impact of different power budgets. *The results show that our controller is effective at responding to changes in the power budgets, while the uncoordinated solution progressively gets worse.*

5.3 System Design Choices

Below, we discuss the impact of a few hypothetical scenarios in terms of different system designs.

Different power budgets: We studied three different power budget configurations – (1) 20-15-10, (2) 25-20-15, and (3) 30-25-20. Note that, from (1) to (3), the peak power savings are increased, and the various power budgets are decreased. Figure 10 shows how our coordination solution responds effectively to the reduced power budgets. The total average power savings are lower with lower power budgets since the VMC is now more conservative about consolidating workloads to avoid violating the reduced power budgets. Our results comparing coordinated with uncoordinated solutions indicate that the need for coordination is increased with more stringent peak power requirements.

Number of P-states: We also studied the impact of the number of P-states for our two systems. Our results showed that all the P-states are not needed in the context of a coordinated solution. In particular, we find that having the two extreme P-states (P0 and P4 in Blade A and P0 and P5 in Server B) can get behavior close to that when all the P-states are considered. The results illustrate how the design of individual control knobs can be simplified in the context of a coordinated architecture. In particular, a processor with two P-states is significantly less complex to test and ship than one with a higher number of P-states. It is also interesting to note that the relative differences between the coordinated and uncoordinated architectures are more pronounced with two P-states than with four. The results show that well-designed coordination is more relevant as the choices for control get more constrained.

5.4 Implementation Choices

Avoiding turning machines off: Our VMC solution assumes that *idle* machines are turned off when workloads are consolidated. However, some users might be nervous about intermittently turning on and off machines. We therefore performed some experiments where we assumed that the option to turn off machines was not available. As expected, our results show significant drop in the net power savings. Compared to the 64% savings the Blade A configuration used to obtain, we now get only 23%. The Server B configuration gets even lesser savings (~5%). It is interesting, however, to note that our coordinated solution automatically adapted to the changed assumption and moved to more aggressively controlling power at the local levels compared to VM consolidation.

Sensitivity to migration overhead: In addition to the baseline, we studied two other configurations, with migration overheads of 20% and 50% during the migration period. Our results showed that the performance degradations increased, but were still less than 10% in all cases for the coordinated solution.

Sensitivity to time constants: We also performed experiments where we varied the time constants of the individual controllers (EC – 1, 2, 5, 10; SM: 1, 2, 5, 10; GM: 50, 100, 200, 400; VMC 100, 200, 300, 400, 500). Our results were relatively invariant to changes in frequency of operation for the EC, SM, and GM. For the VMC, however, increased frequency of operation led to a reduction in power savings. More detailed examination of these results indicated that this was due to the increased aggressiveness in the feedback parameter with increased frequency of operation leading to more conservative workload consolidation. Experiments disabling feedback validated this observation.

Policy choices: We also examined alternate policy choices at the EM, GM, SM, and EC levels. Our results showed no significant variation in the results across the different systems and different classes of workloads. The results indicate the robustness of our architecture to change in individual policy decisions.

6. Discussion

6.1 Extensibility of our approach

The five solutions that we considered in our proposed architecture are representative of the key attributes and challenges in previously proposed power management solutions, e.g., average versus peak, local versus global, per-server versus cluster, power versus performance, and fine-grained versus coarse-grained. Below, we briefly address how our solution can be extended to address other likely deployment scenarios – (1) *Coordination of controllers at the component and platform levels* (e.g., CPU and server power management): we expect the solution be similar to the platform-cluster coordination across EM and GM, (2) *electrical power capper* (e.g., power capper faster than the efficiency loop): as discussed earlier, this can be addressed with an alternative overwriter block implemented in parallel to the nested controller directly adjusting P-states, (3) *multiple actuators at a given level* (e.g., CPU, memory, and disk

power controllers interacting at the platform level): this may be addressed with the use of multi-input-multi-output controllers, (4) *VM-platform level coordination* (e.g, multiple ECs implemented at the VM level): this can be addressed with an arbitration interface similar to the <min> interface used for SM/EM/GM interactions, though likely more generalized, (5) *Heterogeneity in system types*: This can be easily addressed by including a range of different models (like in Figure 5) in the controllers, (6) *energy efficiency and energy-delay objective functions* (different tradeoffs between power and performance): at the higher levels (e.g., VMC), this is a straightforward change to the linear programming optimization problem; at the lower levels (e.g., EC), this can be implemented as a redesign of the controller algorithm, (7) *different hardware/software implementations:* these are in most cases just varying time constants and overheads in the solution.

6.2 Related work

Several previous studies have addressed some coordination issues, but in limited contexts. Chen *et al.* [7] study server provisioning and P-states control for average power, Donald *et al.* [10] study P-states and shut-down for processor thermal capping at the local and global levels. Patel *et al.* [24] discuss a co-efficient-of-the-ensemble to address cooling inefficiencies at multiple levels of the data center. Two recent studies have addressed the interactions between multiple VMs changing the P-states of the same platform [22] and the interactions between average and peak power [9]. The MilliWatt [37][38] and GRACE [30] projects have examined cross-cutting issues in power management across the OS-applications and hardware-software layers respectively. Other work has examined similar issues, albeit on a small scale for CMPs [20][36]. In contrast to these studies, our work is the first to propose a general architectural solution for the problem of coordination of different power management solutions using different techniques and actuators to optimize different objective functions, at different levels, and across hardware and software. We are also unaware of any prior work that has done a detailed quantitative analysis of the tradeoffs in this area with real-world enterprise traces.

There is a huge body of work on individual power and cooling management solutions for the enterprise; Patel and Ranganathan present a good overview in their tutorial [23]. The five individual solutions we study as part of our coordinated solution are inspired by [6][14][28][12][34] respectively. The control algorithm in our efficiency controller is based on the adaptive utilization controller in [35], and similar proof for stability can be provided. Several previous studies have used control theory for power management (e.g., [7][10]), but we are unaware of any previous work that has leveraged connections across the actuators and inputs across multiple controllers to simplify interfaces for coordination.

7. Conclusions

The past few years has seen a surge in interest in enterprise power management with several solutions that individually address different aspects of the problem. Going forward in the future, many (or all) of these solutions are likely to be deployed together for better coverage and increased power savings. Currently, the emergent behavior from the collection of individual optimizations may or may not be globally optimal, or even stable, or correct! A key need, therefore, is a carefully-designed coordination framework that is flexible and extensible and minimizes the need for global information exchange and central arbitration.

In this paper, we propose a coordination solution that addresses this need. Our design is based on carefully connecting and overloading the abstractions in current implementations to allow the individual controllers to learn and react to the effect of other controllers the same way they would respond to changes in workload demand variations. This enables formal mathematical analysis of stability, and provides flexibility to dynamic changes in the controllers and system environments. We demonstrate a specific coordination architecture for five individual solutions using different techniques and actuators to optimize for different goals at different system levels across hardware and software. Using simulations based on close to 200 server traces from real-world enterprise deployments, we demonstrate the effectiveness of our coordination architecture.

We also perform a detailed sensitivity analysis to evaluate several interesting variations in the architecture and implementation, and in the mechanisms and policies space. Our results indicate that effective coordination is likely to be more important in future environments with richer diversity in workloads and increased emphasis on power reduction. Our results also illustrate the relative benefits from individual solutions. Specifically, we find that for current systems with high baseline idle power consumptions, virtual machine consolidation can be a more effective way to save power in spite of its additional overhead, but local power management can still be effective for high-activity workloads. Finally, we also identify interesting insights for future designs. We find that the redundancy in power optimization across multiple levels in a coordinated solution can enable systems to be much simpler by supporting a few widely separated power states (as compared to existing approaches of providing a finer (hard-to-test) spectrum of multiple power states). We find the possibility for similar simplification of policies for the individual controllers. Our results also motivate the need to reduce the baseline idle power for future systems but note interesting advantages from virtual machine consolidation even in those cases.

We believe our work lays the foundation for more work in this space. In particular, we are currently extending our evaluation to consider other power management solutions, but are particularly interested in extending our architecture to include coordination with the equivalent spectrum of solutions in the *performance* and *cooling* domains. Though our work focuses on power management, it is representative of a broad class of problems typified by "intersecting control loops" and it would be interesting to see how our results generalize to the broader resource management domain. Overall, as the complexity of management continues to increase, with multiple players at multiple levels optimizing for multiple objectives, approaches like ours that focus on coordination across these multiple levels are likely to be a critical part of future enterprise architectures.

8. Acknowledgements

We would like to thank the anonymous reviewers as well as acknowledge the feedback and support from Alan Goodrum, Phil Leech, Chandrakant Patel, Sharad Singhal, John Sontag, Niraj Tolia, Tom Vaden, Donald Young and Kumar Goswami.

9. References

[1] L. Barroso. The price of performance. *ACM Queue*, 3(7), Sept. 2005.

[2] P. Bohrer *et al.* The case for power management in web servers. In *Power Aware Computing (PACS)*, 2002.

[3] D. Brooks and M. Martonosi. Dynamic thermal management for high-performance microprocessors. In *7th International Symposium on High-Performance Computer Architecture*, 2001.

[4] E. V. Carrera, E. Pinheiro, and R. Bianchini. Conserving disk energy in network servers. *In 17th International Conference on Supercomputing*, 2003.

[5] J. Chase *et al.* Managing energy and server resources in hosting centers. In *18th Symposium on Operating Systems Principles (SOSP)*, 2001.

[6] J. Chase and R. Doyle. Balance of power: Energy management for server clusters. In *8th Workshop on Hot Topics in Operating Systems*, May 2001.

[7] Y. Chen *et al.* Managing server energy and operational costs in hosting centers. In *ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, June 2005.

[8] CIM Specification, DMTF industry group, www.dmtf,org

[9] B. Diniz *et al.* Limiting the power consumption of main memory. In *34th International Symposium on Computer Architecture (ISCA)*, June 2007.

[10] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *Proc. of the 33rd International Symposium on Computer Architecture,* 2006.

[11] M. Elnozahy, M. Kistler, and R. Rajamony. Energy-efficient server clusters. In *Power Aware Computing Systems (PACS)*, February 2002.

[12] X. Fan *et al*. Power provisioning for a warehouse-sized computer, In *34th ACM International Symposium on Computer Architecture*, CA, June 2007.

[13] W. Felter *et al.,* A performance-conserving approach for reducing peak power consumption in server systems. In *19th International Conference on Supercomputing*, 2005.

[14] M. Femal and V. Freeh. Safe over-provisioning: Using power limits to increase aggregate throughput. In *Power-Aware Computing Systems (PACS)*, December 2004.

[15] P. Gelsinger. Intel Developer Forum, Keynote, April 2006.

[16] The Green Grid™, http://www.thegreengrid.org/home

[17] T. Heath *et al*. Self-configuring heterogeneous server clusters. In *Workshop on Compilers and Operating Systems for Low Power (COLP)*, 2003.

[18] Hewlett Packard. HP Power Regulator for Proliant. Online. http://h18004.www1.hp.com/products/servers/management/ilo/powerregulator.html.

[19] Intel Corporation, Motorola Corporation, and Toshiba Corporation. Advanced configuration and power interface specification, December 1996. http://www.teleport.com/acpi.

[20] P. Juang *et al*. Formal coordinated, distributed energy management of chip multiprocessors, *International Symposium on Low Power Electronics and Design (ISLPED-05)*, August, 2005

[21] C. Lefurgy *et al.* Energy management for commercial servers. In *IEEE Computer,* pp. 39-48, December 2003.

[22] R. Nathuji and K. Schwan. VirtualPower: Coordinated power management in virtualized enterprise systems. In Proc. of the *21st Symposium on Operating Systems Principles (SOSP)*, October 2007.

[23] C. Patel and P. Ranganathan. Enterprise power and cooling. *ASPLOS Tutorial*, October 2006.

[24] C. Patel *et al.* Energy flow in the information technology stack: Introducing the coefficient of the ensemble at its impact on total cost of ownership. In *HP Labs Technical Report HPL-2006-55*, 2006.

[25] E. Pinheiro *et al.* Load balancing and unbalancing for power and performance in cluster-based systems. In *Proc. of the Workshop on Compilers and Operating Systems for Low Power (COLP)*, 2001.

[26] R. Raghavendra *et al.* "No power struggles: Coordinated multi-level power management for the data center," *Hewlett Packard Technical Report, HPL-TR-2007-194*, December 2007.

[27] P. Ranganathan and P. Leech. Simulating complex enterprise workloads using utilization traces. In *10th Workshop on Computer Architecture Evaluation using Commercial Workloads (CAECW)*, February 2007.

[28] P. Ranganathan *et al.* Ensemble-level power management for dense blade servers. In *Proc. of the 33rd International Symposium on Computer Architecture (ISCA)*, 2006.

[29] J. Rolia *et al.* Statistical service assurances for applications in utility grid environments. In *10th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, October 2002

[30] D. G. Sachs *et al.* Grace: A cross-layer adaptation framework for saving energy. In *IEEE Computer, special issue on Power-Aware Computing,* December 2003.

[31] V. Sharma *et al*. Power-aware QoS management in web servers. In *Proc. of the Real-Time Systems Symposium*, December 2003.

[32] United States Environmental Protection Agency (EPA). Enterprise server and data center efficiency initiatives. http://www.energystar.gov/index.cfm?c=prod_development.server_efficiency.

[33] A. Vahdat *et al.* Every joule is precious: The case for revisiting operating system design for power efficiency. In *9th ACM SIGOPS European Workshop*, 2000.

[34] VMware. Vmotion: Virtual machine migration. http://www.vmware.com/products/vi/vc/vmotion.html.

[35] Z. Wang, X. Zhu, and S. Singhal. Utilization and SLO-based control for dynamic sizing of resource partitions. In *16th IFIP/IEEE Distributed Systems: Operations and Management*, Oct. 2005.

[36] Q. Wu *et al*. Formal control techniques for power-performance management, IEEE Micro, Vol. 25, No. 5, September, 2005, pp. 52-63.

[37] H. Zeng *et al*. Ecosystem: managing energy as a first class operating system resource. In *Proc. of the ASPLOS-X*, pp. 123–132. ACM Press, 2002.

[38] H. Zeng *et al*. Currentcy: A unifying abstraction for expressing energy. In *Proc. of the Usenix Annual Technical Conference*, June 2003.

Appendix A: Guaranteeing Stability

In this paper, we rely on control theory to provide formal guarantees for some desirable properties of our control loops, including stability, zero tracking error, as well as adaptivity to changes in the workload. A formal proof for these properties requires the use of mathematical models to describe the system behavior. It also depends on the specific controller algorithms used, and the assumptions made about the system. The analysis becomes more challenging due to the possible interactions among the multiple variables including power, performance, and P-states. Fortunately, our hierarchical architecture design and the use of different time scales in different controllers make it possible to provide at least qualitative arguments for stability.

Below, we sketch an illustrative proof for both stability and zero tracking error in one example scenario. Specifically, we consider the case where the power efficiency (EC) and the power capping (SM) controllers are nested as shown in Figure 2, and prove the following two results: (i) The EC controller can make the CPU utilization track a specified utilization target by dynamically tuning the clock frequency, in spite of slow changes in the workload demand; (ii) The SM controller can make the server power consumption track a given local power cap, possibly set by the upper layer controllers such as the EM or the GM, by dynamically tuning the utilization target fed into the EC controller. Note that, in (i), by "slow" changes we refer to the situation where the workload demand changes at a time scale much longer than the time scale of the efficiency controller.

We first consider the stability of the efficiency control loop. For analysis purposes, we represent the CPU capacity on a server using its clock frequency f, and similarly represent the CPU demand of all the workloads on the server as f_D. The actual measured CPU consumption, denoted as f_C, is upper bounded by both the available capacity and the workload demand, i.e., $f_C = \min(f, f_D)$. We assume that excessive demand in one control interval is not carried over to the next control interval. Moreover, we ignore the quantization that converts continuous clock frequencies to discrete P-states, and assume that the clock frequency could be tuned continuously. We can then define the CPU utilization on a server in an interval k as the ratio between the measured consumption and the available capacity, that is,

$$r(k) = \frac{f_C(k)}{f(k)} = \min\left(1, \frac{f_D(k)}{f(k)}\right). \quad (1)$$

Proposition A: For a given utilization target $r_{ref} < 1$, the CPU utilization of the server, r(k), converges to r_{ref} asymptotically, in spite of slow changes in the workload demand f_D, using the efficiency controller that implements the following control law:

$$f(k) = f(k-1) - \frac{\lambda f_C(k-1)}{r_{ref}}\left(r_{ref} - r(k-1)\right) \quad (2)$$

where $$0 < \lambda < 1 / r_{ref} . \quad (3)$$

Proof: With the assumption of slow changes, we can assume f_D to be a constant for the purpose of this proof. Equation (2) shows that $r(k) = r_{ref}$ if and only if $f(k+1) = f(k)$. Hence, we only need to prove that the clock frequency $f(k)$ converges to some steady state $f_0 = f_D / r_{ref}$. Rewrite Equation (2) as

$$f(k) - \frac{f_C(k-1)}{r_{ref}} = \left(1 - \frac{\lambda f_C(k-1)}{f(k-1)}\right)\left(f(k-1) - \frac{f_C(k-1)}{r_{ref}}\right). \quad (4)$$

Note that $f_C(k-1) <= f_D$, for all k. In the case where $f_C(k-1) < f_D$, we know that $f(k-1) = f_C(k-1)$, and from Equation (2), $f(k) > f(k-1)$ since $r(k) = 1 > r_{ref}$. That is, both $f(k)$ and $f_C(k)$ will increase monotonically until f_C reaches f_D. So next we only consider the case where $f_C(k-1) = f_D$, when Equation (4) becomes

$$f(k) - f_0 = \left(1 - \frac{\lambda r_{ref} f_0}{f(k-1)}\right)\left(f(k-1) - f_0\right). \quad (5)$$

Now consider the following two cases:

a) If $f(k-1) \geq f_0$, then $0 < \frac{\lambda r_{ref} f_0}{f(k-1)} \leq \lambda r_{ref} < 1$. Therefore, $0 < 1 - \frac{\lambda r_{ref} f_0}{f(k-1)} < 1$. This implies that $f(k) - f_0 \to 0$ guaranteeing that the clock frequency will converge to f_0;

b) If $f(k-1) < f_0$, then $f(k) > f(k-1)$ based on equation (2) because $r(k-1) > r_{ref}$. If $f(k) \geq f_0$, then convergence of $f(k)$ is guaranteed from condition a). If, however, $f(k)$ remains below f_0, it has to converge to a constant. But Equation (5) shows this constant has to be f_0. □

So far we have proved the global stability of the EC controller as in Proposition A. If we only consider local stability, the condition could be extended to $0 < \lambda < 2 / r_{ref}$ (see [35]).

For the stability analysis of the local power capper (SM), we assume that the SM controller is sufficiently slower than the EC controller such that the server utilization $r(\hat{k})$ has enough time to converge to every new target $r_{ref}(\hat{k})$. In this case, the server power consumption $pow(\hat{k})$ is a nonlinear decreasing function of $r_{ref}(\hat{k})$, which can be linearized locally as

$$pow(\hat{k}) = -c\, r_{ref}(\hat{k}) + d, \quad c > 0,\ d > 0, \quad (6)$$

and the slope c depends on the operating point $r_{ref}(\hat{k})$. Hence,

$$pow(\hat{k}) - pow(\hat{k}-1) = -c(r_{ref}(\hat{k}) - r_{ref}(\hat{k}-1)). \quad (7)$$

Note that the SM controller works under the following model:

$$r_{ref}(\hat{k}) = r_{ref}(\hat{k}-1) - \beta_{loc}(cap\_loc - pow(\hat{k}-1)). \quad (8)$$

Substituting Equation (8) into Equation (7), we get

$$pow(\hat{k}) = (1 - \beta_{loc} c)\, pow(\hat{k}-1) - \beta_{loc}\, cap\_loc. \quad (9)$$

This function is stable if and only if $|1 - \beta_{loc} c| < 1$, or $0 < \beta_{loc} < 2/c$. Let $c_{\max}$ be the upper bound on the slope c, then $0 < \beta_{loc} < 2/c_{\max}$ provides a sufficient condition for global stability for all $0 < r_{ref}(\hat{k}) < 1$ for the SM controller.