

Ensemble-level Power Management for Dense Blade Servers

Parthasarathy Ranganathan, Phil Leech, David Irwin*, and Jeffrey Chase*
Hewlett Packard

partha.ranganathan@hp.com, phil.leech@hp.com irwin@cs.duke.edu, chase@cs.duke.edu

Abstract

One of the key challenges for high-density servers (e.g., blades) is the increased costs in addressing the power and heat density associated with compaction. Prior approaches have mainly focused on reducing the heat generated at the level of an individual server. In contrast, this work proposes power efficiencies at a larger scale by leveraging statistical properties of concurrent resource usage across a collection of systems (“ensemble”). Specifically, we discuss an implementation of this approach at the blade enclosure level to monitor and manage the power across the individual blades in a chassis. Our approach requires low-cost hardware modifications and relatively simple software support. We evaluate our architecture through both prototyping and simulation. For workloads representing 132 servers from nine different enterprise deployments, we show significant power budget reductions at performances comparable to conventional systems.

1. Introduction

The increasing power density of servers poses a key challenge in enterprise data center environments. For example, the rated power consumption of a typical server is estimated to have increased by nearly a factor of 10 over the past ten years [7]. Such increased power densities can lead to a greater probability of thermal failover, impacting the availability of these systems. Additional cooling is required to avoid thermal failover, leading to a dramatic increase in facility costs for cooling. For example, a 30,000 square foot 10MW data center can easily spend \$2-\$5 million for the cooling infrastructure [16]. Additionally, cooling can also require significant recurring costs. Every watt of power consumed in the compute equipment needs an additional 0.5 to 1W of power to operate the cooling system [16]. That adds another \$4-\$8 million in yearly operational costs. Similarly, there have been increases in the costs for cooling at an individual server as well.

The increasing power density also poses significant challenges in routing the large amounts of power needed per rack for future systems. For example, the power delivery in typical data centers is near 60 Amps per rack. Even if the cooling problem can be solved for future higher density systems, it is highly likely that delivering current to these configurations will reach the power delivery limits of most data centers.

Beyond power delivery and cooling, increased power also has implications on the electricity costs for the compute equipment. For a 10MW data center, this can range in the millions of dollars per year [16]. Increasing energy consumption also has an environmental impact (e.g., 4 millions tons of annual carbon-dioxide emissions) and environmental agencies worldwide are considering standards to regulate server and data center power (e.g., EnergyStar, TopRunner).

These problems are likely to be exacerbated by recent trends towards consolidation in data centers and adoption of higher-density computer systems [18]. Blade servers in particular, have been roadmapped to consume up to 55KW/rack – more than a five-fold increase in power density compared to recently announced 10KW/rack systems [15].

Traditionally, power density and heat extraction issues are addressed at the facilities level through changes in the design and provisioning of power delivery and cooling infrastructures (e.g., [16, 19]). However, these involve greater capital investment and/or additional transitioning costs. Furthermore, it is unlikely that future increases in power densities can be addressed purely at a facilities level.

At the systems level, there has been relatively little work in the area of enterprise power management, with most prior work focusing on mobile battery life issues. There has been some work [4, 6, 17] on algorithms to power-off or power-down servers when they are not in use, but these focus mainly on the average electricity consumption of individual servers.

In contrast to these approaches, our work proposes a new approach based on power management across a broader collection of individual servers. Our work leverages observations culled from analyzing several months of resource (and power usage) trends over more than a hundred servers in several real-world enterprise deployments. We find that enterprise systems are typically underutilized. Across collections of systems, there are large inefficiencies from provisioning power and cooling resources for a worst-case scenario, involving concurrent occurrence of individual power consumption spikes, *which almost never happens in practice*.

We leverage these common-case trends to propose a new power budgeting approach across an “ensemble” of systems. As a specific example, we discuss a new blade architecture where the power is managed and enforced at the level of the *enclosure* (or the chassis). Such an architecture recognizes trends across multiple systems and extracts power efficiencies at a larger scale. This leads to significant reductions in the requirements for power delivery, power consumption, and cooling in the system. As a side benefit, this approach also enables more flexibility in the choice of component power budgets and allows for improved low-cost designs for power-supply redundancy.

We discuss the high-level architecture of such a solution and the specific implementation details of the design. Overall, our approach requires low-cost hardware modifications and small changes to the software. We evaluate our design through both prototyping and simulation. For the 132 real-world enterprise server traces, our results show significant power budget reductions – up to 50% in the processor component and up to 20% in the overall system power – with workload slowdown close to 0% in most cases.

The rest of the paper is organized as follows. Section 2 presents detailed information on enterprise resource usage and power consumption to motivate our approach. Section 3 discusses the design and implementation of our architecture, and Section 4 provides an evaluation of the effectiveness of the design and the various trade-offs. Section 5 presents a qualitative discussion of other benefits from our approach, and Section 6 discusses related work. Section 7 concludes the paper.

2. Real-world Trends

In this section, we present detailed information on resource usage in enterprise environments, including long-term data over a spectrum of real-world

deployments. Our primary goal is to motivate and quantify the key trends that we leverage in our power management solution discussed next.

Resource usage as proxy for power consumption:

One of the challenges of focusing on “live” real-world enterprise infrastructure is the lack of existing support for fine-grained power monitoring. Given the ongoing use of these servers in business-critical functions, we could not shut down the machines to add the necessary metering either. However, these environments either already had rich support for measuring system resource usage or allowed simple software scripts to enable it in real time. Therefore, for the discussions in this paper, we use the resource usage trends, specifically, that of the processor, as a first-order proxy for power consumption trends.

To validate this assumption, we performed some experiments on configurations similar to the environments that we considered. Our results showed that, for these cases, the trends in resource utilization were a good first-order proxy for the overall power (albeit with some attenuation to account for some constant factors). Furthermore, the processor power consumption was the dominant (40-75%) and most variable component of total server power; this conclusion is consistent with earlier studies (e.g., [4]). Therefore, we focus our discussion on processor utilization. We also collected data on the memory, network, and disk usage, and though not reported here for lack of space, the trends discussed below apply qualitatively to those as well. (Note that in Section 4, where we discuss the results from our prototype – on an environment that *we* control – we present data measuring the entire system power that validates this assumption further.)

Data collection: We studied the variation in the CPU utilization for 132 servers from nine “live” enterprise environments (including HP, Walmart, and others who requested anonymity). These deployments run a variety of application environments such as enterprise resource planning, online transaction processing, data warehousing, collaborative applications, IT and web infrastructure workloads, backend client processing, and application development and simulation workloads. The data includes both traces that we collected ourselves or had access to, as well as one public trace that provided this information [2]. The traces were collected over 3 to 10 weeks, at sampling time periods ranging from 15 seconds to 5 minutes.

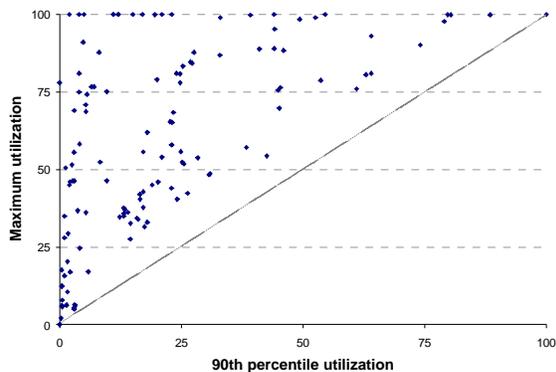


Figure 1: Summary data on individual utilization trends of 132 enterprise servers. Each point represents a server.

Figure 1 summarizes the resource consumption behavior of individual servers. Each point in the scatter plot represents a server and shows the 90th percentile of utilization with respect to the maximum utilization. Figure 2 presents the cumulative resource usage statistics for the nine sites across all the servers. For each site, we pick a representative 7-day trace when all the servers are active, and at each time stamp, add the CPU utilization of each server (between 0-100%) to obtain an overall resource utilization trace for the site. We present the average, 90th percentile, and maximum value for each case. The “sum-peaks” column presents the value obtained from summing the peaks from the individual server utilizations (from Figure 1). The “worst” column shows the actual utilization that the system is provisioned for. The “savings” column shows the differences between the actual provisioning and the maximum utilization.

Trends: Figures 1 and 2 summarize two key trends relevant to our discussion.

Bursty, small-duration spikes: At an individual server level, Figure 1 shows how resource utilization is low and bursty, with spikes being relatively infrequent and of small duration. For example, Figure 1 shows that the 90th percentile of resource usage is often significantly lower than the maximum utilization. This has also been documented in several previous studies and stems from seasonal variations in access patterns, and common resource deployment practices.

Non-synchronized spikes: More interestingly, across a large collection of servers, such as in a data center or blade cluster, the probability of synchronized spikes on all the servers at the same time is rather low. For example, a server used for ATM transactions may spike on Friday versus a server used for airline

transactions that spikes on Thursdays. Similarly, payroll servers have increased utilizations at the end of the month, which may not be concurrent with asynchronous spikes of other servers timed with advertising launches or product tape-outs. Time-zone differences across different groups in a global organization also shift the peak usage times. In all the nine enterprise deployments we study, the sum of the individual peak resource usages is significantly higher than the peak of the total resource utilization. For example, for Site 1, the peak of the entire solution is a total CPU utilization (over 26 servers) around 300%. In contrast, the sum of the peak utilizations of the individual servers is 1100%, and the actual provisioning is 2600%. The magnitude difference between the provisioned worst case and the actual worst-case utilizations is 88%.

Our data shows that these trends are relatively general, and not limited to an individual site or mix of workloads. For example, the world-cup infrastructure represents usage patterns consistent with a one-time event while the e-commerce infrastructure shows more regular long-term usage patterns. Similarly, some sites have servers with independent workloads (e.g., backend client) while others have multi-tier interrelated workloads (e.g., e-commerce). However, the trends are qualitatively similar in all the cases.

While we summarize instrumentation data gathered from these sites, we also have anecdotal and less detailed data from several other enterprise deployments that match these trends. We do not claim that *all* enterprise workloads share these trends; however, our goal is to show that a large existing base of enterprise workloads do share these trends.

Implications for power management: Given that power consumption closely tracks resource usage, the same trends exist for power consumption behavior in these workloads, as well.

Current practice, however, is to design the power budget for the worst-case individual system scenario. This affects the provisioning of cooling (fans) and power delivery (power supplies) in the server. Since worst-case power spikes happen infrequently, this leads to inefficient overprovisioning in the cooling and power delivery at the system level.

When these systems operate in the context of a larger collection of systems, such as a data center, the inefficiencies are compounded. The total power rating of the collection of systems is typically computed as the sum of the individual worst-case ratings. Given

Site	Workload and trace length	Servers	Avg	90th %	Max	sum-peaks	Worst	Savings
1	Backend of pharmaceutical company	26	87	138	307	1128	2600	88%
2	Web hosting infrastructure for worldcup98 web site [2]	25	256	481	1166	1366	2500	53%
3	SAP-based business process application in large company	27	585	691	919	1654	2700	66%
4	E-commerce web site of a large retail company	15	83	166	234	591	1500	84%
5	Backend for thin enterprise clients - company 1	10	138	184	298	729	1000	70%
6	Backend for thin enterprise clients - company 2	14	102	159	287	1253	1400	80%
7	Front-end customer facing web site for large company	8	119	187	255	467	800	68%
8	Business processing workload in small company	3	78	132	225	278	300	25%
9	E-commerce web site of small company	4	90	136	197	228	400	51%
	All sites	132	1540	1872	2682	7694	13200	80%

Figure 2: Cumulative resource utilization behavior for the nine sites. The last column summarizes the potential savings in processor resource (and power) provisioning from ensemble-level management.

that the chances of synchronized power peaks are low (as with the CPU utilization), this leads to even greater differences between the estimated worst-case power and the actual peak power at this level. Further, this estimated worst-case power is used when planning the cooling and power delivery at these higher levels (e.g., air-conditioning units, power distribution units), and consequently, these also end up being overprovisioned.

One option to address these overprovisioning inefficiencies is to move the power budget management to a higher level – at a broader collection of systems (“ensemble”). The key idea is to set the power budget at the ensemble level to avoid excessive overprovisioning. Individual bursty workloads can still be handled within this overall power budget by dynamically redistributing power budget to that server, from other servers not currently requiring as much power. In the cases when this is not possible, performance throttling can be used to reduce power to avoid redlining (temperature increase beyond a critical threshold). The challenges involve careful design of the hardware hooks as well as implementing the policies that manage and enforce the budget.

3. Ensemble-level Power Budget Management

Below, we discuss the architecture for ensemble-level power budget management. In this paper, we focus primarily on blade servers, since their inherent design provides for multiple servers housed in the same enclosure with a common control point (in the enclosure manager); however, our approach can be easily adapted to non-blade servers as well.

3.1 High-level description

Functional architecture: Figure 3 presents a conceptual diagram of our approach. The key components are (1) a controller at the ensemble level, and (2) a management agent at each blade. The

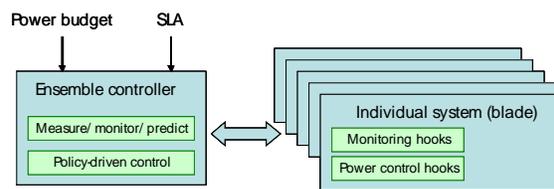


Figure 3: Ensemble-level power management. The key is to consider power budget management across a broader collection of systems.

management agent provides *local power monitoring and control* per server. The controller collects all the local readings and estimates total power consumption at the ensemble level. This information is then fed to a policy-driven control engine that issues directives to the individual blades on the next steps for power control. For example, if the total power exceeds a *pre-determined power budget*, the controller directs the individual servers to *throttle* the power consumption to bring the overall power back under the threshold (e.g., through voltage scaling). The policy heuristics can be implemented to minimize the impact on performance for the end user, and may be used in concert with higher-level *service-level agreements (SLA)* for different workloads.

Benefits: This approach enables us to provision the power budget of the ensemble to a value much lower than the sum of the worst-case power for each of the individual servers. This allows significant reductions in the requirements for power delivery, power consumption, and heat extraction in the system. This, in turn, can lead to designs that use power supplies of lower ratings (lower costs and better efficiencies), consume less electricity (lower costs and better environmental friendliness), and require reduced investment in cooling equipment like fans and air-conditioning (lower costs).

As an illustrative example, let us consider the cooling requirements for a 500W blade enclosure with each

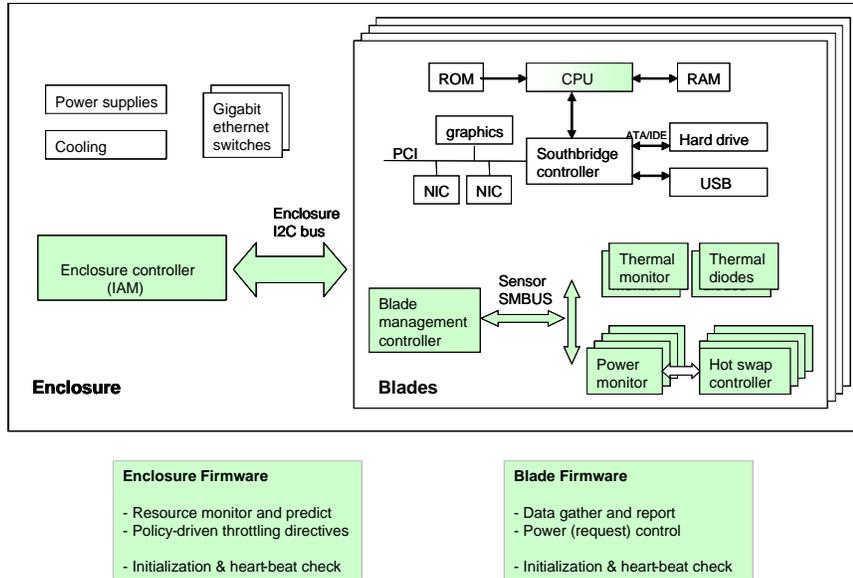


Figure 4: Implementation of enclosure-level power management.

blade rated at 20W. On current systems, this requires provisioning each blade with support for heat dissipation up to 20W (heat sinks, etc) and support at the blade enclosure level for heat dissipation of 500W (fans, etc), and support at the data center level for heat dissipation of 10KW per rack (air conditioning, etc.).

In contrast, consider the scenario where we implement ensemble-level power management at the level of the blade enclosure or chassis, and the power is set based on the peak of the cumulative resource utilization. As seen in Figure 2, this value is typically 25-90% lower than the worst-case provisioning. The enclosure power and cooling budget can be reduced significantly without affecting any properties of the solution.

A more aggressive approach would set the power budget to an even smaller value – say, the 90th percentile of the cumulative power usage. Given the bursty nature of the usage, this can achieve even more savings (Figure 2 shows that this is an additional 25-60% lower.), but at the expense of scenarios where performance needs to be throttled to bring the system within budget. The workload slowdown from such throttling can be minimized by intelligently selecting the blades to throttle. Additionally, one can judiciously allow some spikes over budget as long as the heat can be dissipated before redlining. This is further discussed in Section 4.

Reduced enclosure power budgets also translate to reduced rack-level cooling requirements in the data center. The per-server cooling can also be reduced as

long as it is adequate to dissipate the heat under the transient bursts the server encounters for the specific policy adopted by the enclosure controller.

A similar discussion is applicable to the power delivery to the system. However, the policies are more constrained relative to the cooling case since transient spikes over budgeted power will trip the fuse and need to be prevented.

3.2 Enclosure-level Implementation

Below, we discuss the implementation of our approach for a blade server enclosure. Figure 4 summarizes the various elements of the architecture that we consider. The enclosure is a rack-mountable chassis and contains 20 blade bays, two gigabit Ethernet switches, and an embedded enclosure management controller called the Integrated Administrator Module (IAM). The individual blades include the processor, chipset, memory, hard drive, and network interfaces. In addition, each blade also includes an ASIC that functions as a blade management controller and manages and controls the hardware, responds to events, and communicates with the enclosure manager. The changes for implementing our solution consist of a few relatively straightforward hardware additions at the blade level and changes to the firmware at both the blade and enclosure levels. Below, we discuss the key implementation issues in more detail.

Choosing and enforcing the power budget: As discussed earlier, the sum of the worst-case power

ratings of the individual blades in the enclosure gives an upper bound on the power budget. The lower bound of the enforceable power budget can be determined by examining the maximum reduction in power possible at the subcomponents from power throttling (discussed next). The specific power budget is chosen within this range, based on whether we are targeting savings in the cooling or power delivery subsystem, and specific constraints in the implementation of those. For example, the rated value of an available power supply or the heat extraction capacity of a specific cooling arrangement can determine the chosen power budget. Similarly, in terms of strictness, occasional transients over budget can be better tolerated when optimizing the cooling than when optimizing the power delivery.

Power monitoring and control at the blade level: A key aspect of the architecture is the power monitoring at the individual blades. On-board thermal sensors available in many current systems can detect thermal redlining; however, measuring the current and power consumed is necessary to optimize power delivery, and can provide finer control on heat dissipation. These can be obtained with relatively low-cost circuits to measure the voltage drop across a sense resistance and the rail voltage at the blade input. The measured values can be communicated by the hot swap controller/power monitoring device through an interface such as sensor SMBUS to the blade management controller. If needed, data can also be collected at each power plane to provide a finer breakdown of power consumption. Additional higher-level system resource utilization metrics, if exposed to the drivers, can also be communicated the same way.

From a power control point of view, most current systems already have some hooks for power reconfiguration. For example, these could be in the support of the different power states (P-states) in the ACPI specification [1]. Voltage and frequency scaling is either already supported or is planned to be supported for server processors (e.g., AMD, Intel). Previous work has also evaluated power control of other components such as memory and disk [11, 12]. The BIOS and embedded controller on the blade can be used to implement whatever power control mechanisms are chosen. Alternatively, power control can also be effected at a system level by turning off servers [6] or by using a choice of servers in heterogeneous systems to provide power control [10].

Communication between the blade and enclosure: The communications between the enclosure

administrator and the individual blades can be performed on top of standard interfaces such as I2C or SMBUS. Data collection can be done through both polling and interrupts. For example, the enclosure manager can periodically poll every blade in the enclosure and gather power data. Alternatively, the blade controller can communicate pre-defined power and thermal events on the blade through alert signals (such as SMBALERT) to the enclosure manager.

Policy choices in enclosure controller: There are two broad classes of policies to control the individual blades to meet the specified power budget – pre-emptive and reactive.

Pre-emptive management proceeds through pre-defined budget allocation to each blade server by the enclosure manager. If a blade server requires more power, the blade must request power from the enclosure manager to be allowed to proceed to the next power level. The embedded controller sets the blade power level to the allocated power level and only allows a higher power level when permission is granted from the enclosure. (*Don't assume you can use more power, always ask.*)

Reactive management, on the other hand, proceeds by having the enclosure monitoring the power levels of the entire enclosure and responding only in the event of a threshold violation. This has the advantage of being less conservative about throttling, but incurs the possibility of transient power spikes over the threshold. (*Use as much power as needed until told you can't.*)

Consequently, while pre-emptive policies can be used in all applications of power budget control, reactive policies cannot be used in situations like power supply optimizations which need stricter budget enforcement.

Figures 5(a) and 5(b) summarize the pseudocode for both the algorithms. Note that several variants of these algorithms are possible. Figure 5(c) provides a brief high-level taxonomy of the design space. An exhaustive study of this policy space is outside the scope of this paper; however, in Section 4, we discuss the variation of several of these parameters.

3.3 Operation

The operation of the system has three phases:

Initialization and setup: When a blade first turns on in an enclosure, as part of the power-on-self-test, it reports the power it uses in the different p-states, to the enclosure manager. The BIOS and blade controller perform calibration experiments varying the

```

Start with all servers throttled

At each control period or on interrupt
  Compute total power consumption

  Identify servers with "low" utilization
  Prioritize which servers to throttle
  Throttle each server to decided level

  Check if room in power budget
  If yes
    Identify servers with "high" utilization
    Prioritize which servers to unthrottle
    Unthrottle each server to decided level
    Stop if power budget likely exceeded
  If no
    Stop

```

(a) Preemptive algorithm

```

Start with all servers unthrottled

At each control period or on interrupt
  Compute total power consumption

  Check if power above threshold
  If yes
    Prioritize which servers to throttle
    Throttle each server to decided level
    Stop when power budget below threshold
  If no
    Prioritize which server to unthrottle
    Unthrottle each server to decided level
    Stop if power budget likely exceeded

```

(b) Reactive algorithm

Parameter	Options
When to assign power budgets	Pre-emptive or reactive
Which server to (un)throttle	Round-robin, random, p-state, power, performance, fair-shared based on past-history, customer-service requirements
What (un)throttle knob to use	Processor voltage/frequency scaling, memory, disk, turn blades off/on, heterogeneity, others...
What level to (un)throttle to	Next P-state, lowest/highest P-state, per-blade policy
How to predict future power	Use past history as future indicator, other resource prediction models, conservativeness of prediction
Event trigger	Interrupt-driven on power event, polling
How often to monitor/control	Polling frequency, number of blades to poll per second, interrupt service times, hysteresis times

(c) Policy design space

Figure 5: Policies to manage enclosure power budget.

processor frequency and voltage (and more broadly the various power control options) and executing code to exercise the system. During these experiments, the power consumption data is gathered from the hot swap controllers, and reported to the enclosure manager. The enclosure manager then uses this power data to make later decisions about which power level a blade may need to operate.

Data gathering and heartbeat checking: Data gathering takes place after the blade server has reported its power levels to the enclosure manager. During data gathering, the enclosure manager polls each blade in the enclosure periodically and gathers power data from the blade. The polling also lets the

blade know that the enclosure manager is still operating properly. A watchdog timer in the blade management controller also monitors for the enclosure manager heartbeat. In the event of two consecutive intervals without an enclosure response, all blades transition to the lowest operational level to ensure no power budget violation. Depending on the communication latencies of the design, one of the blades can then optionally take on the role of the power management controller.

Responding to events: Power and thermal events are triggered when the polling and power estimation models determine a possible violation of the power budget. Alternatively, power and thermal events can be triggered when the thermal diode on the blade server exceeds a pre-programmed limit or when the blade power exceeds or falls below pre-defined limits. As discussed above, these power and thermal events trigger specific recommendations from the policy engine in the enclosure manager and corresponding actions by the blade management controller.

4. Evaluation

Our architecture optimizes for *common-case* usage behavior likely to be seen in actual deployments over *multiple servers*. Validating this solution poses several challenges.

Ideally, we would like to develop a prototype implementation and deploy it in a live enterprise environment spanning multiple servers. However, building a prototype blade enclosure from scratch requires significant resources. Getting enterprises to trust their business applications to prototype hardware is also a difficult proposition. Even if we were to mirror some of the software setup on these enterprise platforms, exercising our setup to show behavior akin to a live deployment is even more challenging. Additionally, it is hard to perform a detailed design space exploration with the one specific hardware implementation a prototype would represent.

However, the alternate approach of using a simulator is also challenging. A detailed full system simulation has all the disadvantages of the prototype and is additionally several orders of magnitude slower than the real system.

Given these challenges, we choose a hybrid approach to validate our solution, based on real prototyping *and* simulation. We build a prototype and use it to validate the implementation details discussed earlier, as well as measure actual performance and power data on toy workloads modeling typical enterprise usage. We

supplement this with results from simulating the workloads representing the 132 servers from the nine real-world enterprise sites discussed earlier. We validate the simulator models against the prototype. We also use the simulator to evaluate synthetic workload traces of varying concurrency and utilization, and to perform a design space exploration of tradeoffs in various hardware and software parameters.

4.1 Methodology

Prototype: Our prototype implementation builds on an existing blade design from an earlier project. Each blade includes a 1GHz Transmeta Efficeon (TM8000) processor, 256 MB SODIMM memory, and 40 GB storage (Seagate 5400 RPM). The blade management controller architecture includes a super I/O chip and an integrated 8051 controller. At a firmware level, additional software modules are added to the BIOS, and the blade and enclosure controllers. These implement the initialization/setup routines, data gathering and measurement, heartbeat timers, and the pre-emptive and reactive power management policies discussed earlier. Software in the ROM/BIOS handles configuration and support for power control at the individual blade level. The only power control mechanism available in our prototype is the use of voltage and frequency scaling in the processor. There were five voltage and frequency settings: (1) 533MHz at 0.8V, (2) 600 MHz at 0.925V, (3) 700 MHz at 1V, (4) 833MHz at 1.1V, and (5) 1000 MHz at 1.25V. All the blades run Windows XP and our enclosure-level testbed includes 8 blades.

For the results reported on the prototype, we log power consumption as measured by a power meter connected to the actual hardware and report the average power for the experiment.

Simulator: Our simulator models high-level properties of the prototype blade enclosure. It takes resource utilization traces as input and models the impact of different enclosure controller policies on performance and power under various load and policy conditions. The basic operation of the simulator is as follows. The main simulator loop operates on a timer that matches the time of the simulated system. When reading the input trace file, the simulator uses this timer to access resource utilization data at the corresponding time stamp associated with it.

However, there are several challenges with using traces that only capture resource utilization information. First, we need to determine the variation

in resource utilization when the system is changed, for example, with voltage and frequency scaling. Second, we need to be able to correlate these changes in resource utilization to performance, to better understand the impact on application latencies and throughputs. Finally, we need to correlate resource utilization to the power consumed in the system.

To address these challenges, we run experiments on the prototype using `gamut` [13], a synthetic load generator, to execute a pre-determined synthetic stress kernel, while controlling the resource utilization in progressive steps. For example, we can exercise the processor with an compute-intensive loop at CPU utilizations from 0% to 100% in small increments. At each of these data points, we measure the power (using a power meter) and the performance (in terms of the useful work done, as reported by `gamut`). We repeat this experiment for all the different states of the system (in our prototype, the five voltage and frequency settings). Based on these experiments, we create high-level models that correlate resource utilization to power consumption, identify changes in resource utilization with changes in frequency, and convert changes in utilization to the corresponding impact on actual work done. Using this approach, we validated high-level properties of the simulator with the prototype and found good correlations.

To report performance degradation (workload slowdown) in this section, we compare the total work done across all blades in the enclosure across different policies. This metric represents the degradation for the entire solution. However, in some cases, a per-server (associated with a specific user or application) metric might be more appropriate. Consequently, we also study per-blade workload degradation averaged across the individual blades in the system. We do not call this metric out in our results since the trends were similar to the enclosure workload degradation.

We also studied the degradation in clock frequency as opposed to work done (this gives a sense of peak MIPS degradation) and also looked at metrics that averaged the percentage of time a blade is throttled and the number of blades throttled per second. In addition, the simulator allows us to monitor and analyze trends on a host of other statistics. These include the variation in power and performance from different perspectives, at a temporal level, at a per-blade level, at an enclosure level, and at the level of specific phases in workloads (e.g., peaks vs idle). In addition we can monitor parameters at the hardware level such as the usage of multiple power states, the effect of delays in servicing

Power budget	Algorithm	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6	Site 7	Site 8	Site 9
400W	Preemptive	1%	1%	4%	1%	1%	2%	2%	5%	4%
425W		1%	1%	3%	1%	1%	2%	2%	5%	3%
450W		1%	1%	3%	1%	1%	2%	1%	4%	3%
400W	Reactive	0%	0%	0%	0%	0%	0%	0%	1%	0%
425W		0%	0%	0%	0%	0%	0%	0%	0%	0%
450W		0%	0%	0%	0%	0%	0%	0%	0%	0%

Figure 6: Summary of workload slowdown for the 9 enterprise traces. The power budgets of 450W, 425W, and 400W represent CPU power reductions of 25%, 38%, and 50%, compared to the base power budget of 500W.

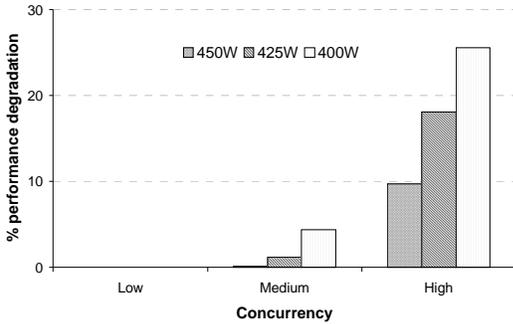


Figure 7. Sensitivity to concurrency (synthetic traces).

the interrupts, etc. We use these to further validate our intuition behind the results.

4.2 Results

Simulation Results: Figure 6 summarizes simulation results for the 9 enterprise sites discussed earlier. We consider three different values of the power budgets – 450W, 425W, and 400W – corresponding to enclosure-level power budget reductions of 10%, 15%, and 20%. Note that this corresponds to equivalent reductions of approximately 25%, 37%, and 50% of the total processor power budget.

As seen from Figure 6, overall, enclosure-level power management can achieve power budget reductions with marginal reductions in total solution performance. Even when the power budget is set to 400W, the performance reductions are less than 5%. The backend desktop trace (site 8) has the maximum slowdown; the higher numbers in this case are due to the occurrence of several utilization traces that stay at 100% for a long period of time. The pre-emptive algorithm has slightly higher performance loss compared to the reactive algorithm because of its conservative budgeting policy. Results with other performance metrics are all qualitatively similar. Note that though the net enclosure power reduction is 20%, the CPU power reduction is almost 50%. Power control at the other components including memory, disk, and switching fabric, would have likely achieved higher power reductions at the overall system level. Further, though not quantified here, power savings at

the enclosure level can lead to corresponding benefits in cooling and power delivery at the data center level, as well.

Interestingly, in all cases, the fraction of time the reactive algorithm exceeds the power budget is less than 0.5%. This is because our algorithm makes a number of assumptions biased towards performance loss over power budget violations (e.g., 25W budget headroom, unthrottle hysteresis). Note that as discussed earlier, pre-emptive algorithms, by design, can never exceed the power budget.

Sensitivity to workload, policy, implementation:

The low performance degradation stems from the low utilization and limited concurrency of spikes in the real-world traces (Section 2). To assess the sensitivity of our results to workload concurrency, we evaluated three synthetic traces, with fine-grained utilization variation, and low (25%), medium (50%), and high (75%) concurrencies. For example, a trace with 25% concurrency has synchronized peak utilization across all the blades 25% of the time. Figure 7 summarizes the performance degradation for the reactive algorithms with the three power budget thresholds as before. As expected, the performance degradation increases with the concurrency of the workload. The fraction of time that the system exceeds the power budget increases as well, but even at the high-concurrency value, this is still less than 5%.

We also evaluated several different policies. Specifically, we considered two different approaches for which servers to throttle (and unthrottle): one where priority was given to servers at the highest power and one where priority was given to servers at the lowest utilization. Similarly, we considered two different approaches on the level to which to throttle the servers: either incrementally transition to the next power-state, or transition to the lowest or highest power state. For the nine real-world enterprise traces, these algorithmic variations made little difference, because of the low utilizations and concurrencies. For the synthetic workloads with higher concurrencies, in general, deep throttling of a few servers consuming

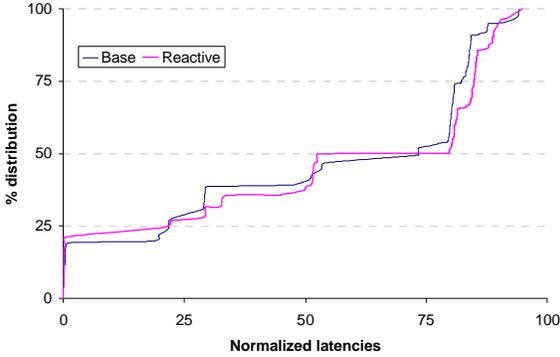


Figure 8: Variation in latencies for `vncplay` on prototype.

the highest power was desirable to throttling a large number of servers at low utilization. Our experiments varying the polling frequency and the interrupt service time showed that the algorithms are fairly robust at adapting to changes in these parameters. It is natural to use a proportional sharing model, in which the power budget is allocated to the contending workloads proportionally to weights assigned to each workload.

Prototype results: In addition to running emulated production workload traces on the prototype, we evaluated two other simple workloads. The first models interactive workloads and is based on the `vncplay` tool developed by Zeldovich et al [22]. This tool records a user’s interactive session with a system and allows it to be replayed multiple times under different configurations. This benchmark allows us to measure the impact of our approach in an interactive GUI environment.

When comparing the total execution time of the session with and without enclosure-level power management, our results showed relatively little variation (less than 1% even with 20% reduction in the enclosure power budget). This is consistent with the simulation data for similar benchmarks with equivalent utilization (site 6). However, in interactive applications, rather than throughput, it is more important to capture the impact on latency for user events. Consequently, following the methodology adopted by Zeldovich et al [22], we also study cumulative distribution functions of the latencies. An example is summarized in Figure 8. As can be seen from the figure, for the different latencies seen by user events in the benchmark, the variation between the base system and the system with enclosure-level power budgeting is quite small.

The second benchmark models batch workloads that stress the CPU. A simulation program that consumes

100% CPU resources is interspersed with smaller periods of idle time, to model a high-concurrency benchmark. As expected, the performance degradations are higher for this workload – 3%, 7%, and 35% for power budget settings of 450, 425, and 400W respectively. Again, this is consistent with the results we observe in our simulations of medium-to-high concurrency workloads.

5. Other Enclosure-level Benefits

While our primary focus in this paper was on reducing the peak power budget to get the associated benefits in power delivery and cooling, enclosure-level power management can have other advantages.

Cost benefits from component choice flexibility:

Based on volumes of chips shipped, low-power processors such as those used in blade systems can sometimes be more expensive than higher-power (possibly even higher-performance) processors using a different technology for a different market segment. Enclosure-level power management allows individual components in a blade (and individual blades in the enclosure) to exceed their local budgets as long as the overall system budget is enforced. This effectively means that power-budget-imposed limits on the choice of components can be relaxed, allowing the use of cheaper higher-volume components even if they are rated for higher power. In some cases, having the option for higher-performance peaks can also provide better single-thread performance during bursty cases.

Cost reduction through reduced redundancy:

Most enterprise servers provide redundancy in the power delivery with two or more power supplies per system. In these designs, all the power supplies are each rated at the peak capacity of the system. Our approach, however, allows for an alternative design point where the secondary power supplies can potentially be rated at lower capacities. Even if the primary supply were to fail, the enclosure can detect that and change the power threshold to that of the secondary supply’s rating. (Most power supply specifications support the transient overloading likely during this transition.) The system would continue to stay operational with the secondary supply, with potentially lower performance, until the problem is fixed. Given the rarity of these failure events, our approach can help reduce costs that are otherwise not recovered during the server lifetime.

Average power reduction: The main motivation (and focus) of our approach is the peak power. However, one could retain the overprovisioning in power and

cooling and still use this approach to extract average power consumption efficiencies. Enclosure-level throttling, say for the 90th percentile cumulative utilization, is likely to be much simpler to implement than equivalent 90th percentile utilizations at the local level where each server threshold is potentially different. Further, enclosure-level power management can also extend local-level power throttling with knowledge of enclosure-level trends (resource consumption, efficiency) to minimize global performance degradation.

6. Related Work

To the best of our knowledge, our work is the first to manage and enforce a peak power budget across a server ensemble, e.g., a blade enclosure. As part of this, we discuss algorithms to redistribute the power budget in two contexts: strict constraints for power delivery and looser constraints for heat dissipation.

At a single-server level, Brooks et al proposed setting a thermal threshold and enforcing it from a cooling point of view [5] while Felter et al suggest dynamic shifting of power within the processor and memory components of a single server [8]. Similar to these studies, we also use the notion of setting and enforcing a power budget, but our work differs from these in its focus on trends across multiple systems, and, indeed, all these optimizations can (and probably will) be used together on future systems.

At a cluster level, Femal and Freeh [9] discuss how, for a given cluster power budget, one can choose different permutations of the quantity and size of individual nodes to better improve throughput, by optimizing for the different power-performance efficiency curves. Individual nodes are responsible for determining their power limits and the environment assumes an explicit trust model between nodes. Other previous work has evaluated algorithms to turn off or turn-down individual servers when they are not used [4, 6, 17]. However, these have been mainly focused on reducing electricity consumption in such environments.

Many papers have studied the resource demand profiles of competing workloads through time in order to evaluate the performance impact of resource sharing. For example, one recent study examines several workloads and concludes that overbooking resources for a shared hosting platform may increase a hosting provider’s revenue while meeting probabilistic service level agreements [20]. Our work uses a similar idea to constrain the power budget for a shared

ensemble of servers, with little performance impact. Our use of dynamic voltage scaling to throttle CPU power consumption dynamically is similar to GRACE-OS [21], which profiles CPU usage in conjunction with soft real-time CPU scheduling to conserve battery power while bounding missed deadlines. They leverage a similar insight at the OS level: all processes will not demand cycles at the same time.

7. Conclusions

In this paper, we address the increasing power density challenges in enterprise servers. We propose the notion of “*ensemble-level*” *power management* to leverage concurrent resource usage trends across collections of systems, for power savings beyond that possible from optimizing a single system in isolation.

We present resource utilization data across a large collection of servers from several live enterprise deployments and identify the potential benefits from such an approach. We discuss an implementation of this approach at the blade *enclosure* level to monitor and manage the power across the individual blades in a chassis. Our proposed design requires low-cost hardware additions and simple software support to conventional systems.

We evaluate our design on a prototype that uses voltage and frequency scaling for CPU power throttling, and also through simulation. Our results show significant power budget reductions (up to 50% reduction in the processor power and 20% in system power) with marginal (close to zero in most cases) impact on performance. The power reductions are likely to be higher on systems with support for power control of other components (e.g., memory, disk). These savings also have a cascading effect in the cooling and power delivery costs at other levels such as in the data center. Beyond power budget reductions, we also discuss how our approach enables lower-cost resiliency and per-component budget flexibility.

Beyond the policies we examined, a rich design space exists for other policies for ensemble-level power control, particularly in the context of geographically-dispersed servers. Another interesting area of future research is the applicability of our approach to high-performance technical computing and virtualized environments. While processor utilization in these environments is higher, opportunities may exist for ensemble-level control of other sub-components of the system. Finally, our approach allows the power budget to be varied on the fly as long as the power delivery and cooling can also be suitably varied. This might be

a promising avenue of research, especially when our ensemble-level control loop is interfaced with local per-server control [1] and broader data center level control [14, 16] of power and cooling. As part of ongoing work, we are currently evaluating these options further.

Overall, as trends towards consolidation and compaction exacerbate the power and heat management challenges, it will become critical to go beyond conventional approaches to solve these problems. We believe that approaches like ours – that optimize at the ensemble level, and for common-case behavior of commercial enterprise workloads – are likely to be an integral part of future solutions to address these challenges.

8. Acknowledgements

We would like to thank Hernan Laffitte, Charlie Shaver, Khaldoun Alzien, Chandra Patel, and John Sontag for support of this work. We are indebted to Martin Arlitt as well as the enterprise customers for the real-world traces. We would also like to thank the reviewers, Luiz Barroso, Mike Schlansker, Manohar Prabhu, and Norm Jouppi for their comments.

9. References

1. Advanced Configuration and Power Interface Specification. Intel Corporation, Microsoft Corporation, and Toshiba Corporation. <http://www.teleport.com/acpi>, December 1996.
2. M. Arlitt and T. Jin. Workload Characterization of the 1998 World Cup Web Site. <http://www.hpl.hp.com/techreports/1999/HPL-1999-35R1.html>, February 1999.
3. L. Barroso, J. Dean, and U. Hoelzle. Web Search for A Planet: The Architecture of the Google Cluster. In *IEEE Micro Magazine*, April 2003.
4. P. Bohrer, D. Cohn, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, R. Rajamony, F. Rawson, and E. V. Hensbergen. Energy Conservation for Servers. In *IEEE Workshop on Power Management for Real-time and Embedded Systems*, May 2001.
5. D. Brooks and M. Martonosi. Dynamic Thermal Management for High-performance Microprocessors. In *the 7th International Symposium on High-Performance Computer Architecture (HPCA)*, January 2001.
6. J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle. Managing Energy and Server Resources in Hosting Centers. In *18th Symposium on Operating Systems Principles (SOSP)*, October 2001.
7. Datacom Equipment Power Trends and Cooling Applications. *ASHRAE*, <http://www.ashrae.org>, 2005.
8. Wes Felter, Karthick Rajamani, Cosmin Rusu, Tom Keller. A Performance-Conserving Approach for Reducing Peak Power Consumption in Server Systems. In *Proceedings of 19th International Conference on Supercomputing (ICS)*, June 2005.
9. M. Femal and V. Freeh. Safe Over-provisioning: Using Power Limits to Increase Aggregate Throughput. In *Workshop on Power-Aware Computing Systems (PACS)*, December, 2004.
10. T. Heath, B. Diniz, E. V. Carrera, W. Meira Jr., and R. Bianchini. Energy Conservation in Heterogeneous Server Clusters. In *10th Symposium on Principles and Practice of Parallel Programming (PPOPP)*, June 2005.
11. D. Helmbold, D. Long, and B. Sherrod. A Dynamic Disk Spin-down Technique for Mobile Computing. In *Proceedings of the 2nd ACM International Conference on Mobile Computing (MOBICOM96)*, November 1996.
12. A. Lebeck, X. Fan, H. Zeng, and C. S. Ellis. Power Aware Page Allocation. In *9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX)*, November 2000.
13. J. Moore, J. Chase, K. Farkas, and P. Ranganathan. Data Center Workload Monitoring, Analysis, and Emulation. In *Eighth Workshop on Computer Architecture Evaluation Using Commercial Workloads*, February, 2005.
14. J. Moore, J. Chase, and P. Ranganathan. Making Scheduling “Cool”: Temperature-Aware Resource Assignment in Data Centers. In *Proceedings of the USENIX Annual Technical Conference*, April 2005.
15. J. Mouton. Enabling the Vision: Leading the Architecture of the Future. *Keynote at Server Blade Summit*, 2004.
16. C. D. Patel, C. E. Bash, R. Sharma, and M. Beitelmal. Smart Cooling of Data Centers. In *Proceedings of IPACK*, July 2003.
17. E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath. Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems. In *Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP)*, September 2001.
18. P. Ranganathan and N. Jouppi. Enterprise IT Trends and Implications for Architecture Research, In *11th International Symposium on High-Performance Computer Architecture (HPCA)*, February 2005.
19. R. Sullivan. Alternating Hot and Cold Aisles Provides more Reliable Cooling for Server Farms. In *Uptime Institute*, 2000.
20. B. Urgaonkar, P. Shenoy, and T. Roscoe. Resource Overbooking and Application Profiling in Shared Hosting Platforms. In *5th Symposium on Operating Systems Design and Implementation (OSDI)*, December, 2002.
21. W. Yuan and K. Nahrstedt. Energy-Efficient Soft Real-Time CPU Scheduling for Mobile Multimedia Systems. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, October, 2003.
22. N. Zeldovich and R. Chandra. Interactive Performance Measurement with VNCplay. In *Proceedings of the FREENIX Track: USENIX Annual Technical Conference*, April 2005.