Power Management of Online Data-Intensive Services

David Meisner[†], Christopher M. Sadler[‡], Luiz André Barroso[‡],
Wolf-Dietrich Weber[‡] and Thomas F. Wenisch[†]

[†]The University of Michigan [‡]Google, Inc.

ABSTRACT

Much of the success of the Internet services model can be attributed to the popularity of a class of workloads that we call Online Data-Intensive (OLDI) *services. These workloads perform significant computing over massive data sets per user request but, unlike their offline counterparts (such as MapReduce computations), they require responsiveness in the sub-second time scale at high request rates. Large search products, online advertising, and machine translation are examples of workloads in this class. Although the load in OLDI services can vary widely during the day, their energy consumption sees little variance due to the lack of energy proportionality of the underlying machinery. The scale and latency sensitivity of OLDI workloads also make them a challenging target for power management techniques.*

We investigate what, if anything, can be done to make OLDI systems more energy-proportional. Specifically, we evaluate the applicability of active and idle low-power modes to reduce the power consumed by the primary server components (processor, memory, and disk), while maintaining tight response time constraints, particularly on 95th-percentile latency. Using Web search as a representative example of this workload class, we first characterize a production Web search workload at cluster-wide scale. We provide a fine-grain characterization and expose the opportunity for power savings using low-power modes of each primary server component. Second, we develop and validate a performance model to evaluate the impact of processor- and memory-based low-power modes on the search latency distribution and consider the benefit of current and foreseeable low-power modes. Our results highlight the challenges of power management for this class of workloads. In contrast to other server workloads, for which idle low-power modes have shown great promise, for OLDI workloads we find that energy-proportionality with acceptable query latency can only be achieved using coordinated, full-system active low-power modes.

ISCA'11, June 4–8, 2011, San Jose, California, USA.

Categories and Subject Descriptors

C.5.5 [**Computer System Implementation**]: Servers

General Terms

Design, Measurement

Keywords

Power Management, Servers

1. INTRODUCTION

Modern data centers' energy efficiency is undermined by server hardware that fails to provide good power-performance trade-offs for data center workloads that have periods of low utilization. Ideally systems would exhibit *energy-proportionality*, wherein servers consume power in proportion to their load. Current servers are far from energy proportional—it is common for a server with only 10% load to draw 50-60% of its peak power consumption. Energy-proportional operation could increase server efficiency by a factor of two [5] as well as increase the efficacy of data center techniques like power capping [14].

In this paper we examine, for the first time, power management for a class of data center workloads, which we refer to as *Online Data-Intensive* (OLDI). This workload class would benefit drastically from energy proportionality because it exhibits a wide dynamic load range. OLDI workloads are driven by user queries/requests that must interact with massive data sets, but require responsiveness in the sub-second time scale, in contrast to their offline counterparts (such as MapReduce computations). Large search products, online advertising, and machine translation are examples of workloads in this class. As shown in Figure 1, although the load on OLDI services varies widely during the day, their energy consumption sees little variance due to the lack of energy proportionality of the underlying machinery.

Previous research has observed that energy-proportional operation can be achieved for lightly utilized servers with full-system, coordinated *idle low-power modes* [20]. Such a technique works well for workloads with low average utilization and a narrow dynamic range, a common characteristic of many server workloads. Other work observes that cluster-level power management (e.g., using VM migration and selective power-down of servers) can enable energy-proportionality at the cluster level even if individual systems are far from energy proportional [27].

As we will show, full-system idle low-power modes fare poorly for OLDI services because these systems have a large

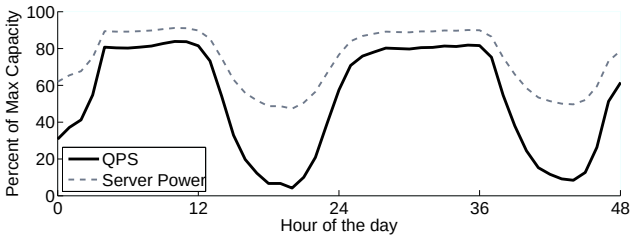


Figure 1: Example diurnal pattern in queries per second (QPS) for a Web Search cluster: Non-peak periods provide significant opportunity for energy-proportional servers. For a perfectly energy proportional server, the percentage of peak power consumed and peak QPS would be the same. Server power is estimated for systems with 45% idle power.

dynamic range and, though sometimes lightly loaded, are rarely fully idle, even at fine time scales. Cluster-grain approaches that scale cluster size in response to load variation are inapplicable to OLDI services because the number of servers provisioned in a cluster is fixed. Cluster sizing is determined primarily based on data set size instead of incoming request throughput. For a cluster to process an OLDI data set for even a single query with acceptable latency, the data set must be partitioned over thousands of nodes that act in parallel. Hence, the granularity at which systems can be turned off is at cluster- rather than node-level.

Fundamentally, the architecture of OLDI services demands that power be conserved on a per-server basis; each server must exhibit energy-proportionality for the cluster to be energy-efficient, and the latency impact of any power management actions must be limited. We find that systems supporting OLDI services require a new approach to power management: coordination of *active low-power modes* across the entire utilization spectrum. We demonstrate that neither power management of a single server component nor uncoordinated power management of multiple components provide desirable power-latency tradeoffs.

We report the results of two major studies to better understand the power management needs of OLDI services. First, we characterize a major OLDI workload, Google Web Search, at thousand-server, cluster-wide scale in a production environment to expose the opportunities (and non-opportunities) for active and idle low-power management. We introduce a novel method of characterization, *activity graphs*, which enable compact representation of the activity levels of server components. Activity graphs provide designers the ability to identify the potential of per-component active and idle low-power modes at various service load levels. Second, we perform a study of how latency constrains this potential, making power management more difficult. We construct and validate a performance model of the Web Search workload that predicts the 95th-percentile query latency under different low-power modes. We demonstrate that our framework can predict 95th-percentile latency within 10% error. Using this framework, we explore the power-performance tradeoffs for available and future low-power modes.

We draw the following conclusions about power management for major server components:

1) **CPU active low-power modes provide the best *single* power-performance mechanism, but are not sufficient for energy-proportionality.** Voltage and frequency scaling (VFS) provides substantial power savings for small changes in voltage and frequency in exchange for moderate performance loss (see Figure 15). Looking forward, industry trends indicate that VFS power savings will be reduced in future technology generations as the gap between circuits' nominal supply and threshold voltages shrink [6], suggesting that power savings may not be realized from VFS alone. Furthermore, we find that deep scaling yields poor power-performance tradeoffs.

2) **CPU idle low-power modes are sufficient at the core level, but better management is needed for shared caches and on-chip memory controllers.** We find that modern CPU cores have aggressive clock gating modes (e.g., C1E) that conserve energy substantially; power gating modes (e.g., core C6) are usable, but provide little marginal benefit at the system level (see Figure 16). However, we observe that non-core components such as shared caches and memory controllers must remain active as long as *any* core in the system is active. Thus, we find opportunity for full socket idle management (e.g, socket C6) is minimal.

3) **There is great opportunity to save power in the memory system with active low-power modes during ample periods of underutilization.** We observe that the memory bus is often highly underutilized for periods of several seconds. There is a great opportunity to develop active low-power modes for memory (e.g., [10]) and we demonstrate that these would provide the greatest marginal addition to a server's low-power modes. Because the memory system is so tightly coupled to CPU activity, it is rare for DRAM idle periods to last long enough to take advantage of existing idle low-power modes (e.g., self-refresh) (see Figure 7).

4) **Unlike many other data center workloads, full-system idle power management (e.g., PowerNap) is ineffective for OLDI services.** Previous research has demonstrated that energy-proportionality can be approached by rapidly transitioning between a full-system high-performance active and low-power inactive state to save power during periods of brief idleness [20]. Whereas such a technique works well for many workloads, we demonstrate that it is inappropriate for the ODLI workload class. Because periods of full-system idleness are scarce in OLDI workloads, we evaluate batching queries to coalesce idleness, but find that the latency-power trade-offs are not enticing.

5) **The only way to achieve energy-proportional operation with acceptable query latency is coordination of full-system active low-power modes.** Rather than requiring deep power savings out of any one component, we observe that systems must be able to leverage moderate power savings in all their major components. Since full-system idleness is scarce, power savings must be achieved with active low-power modes. Furthermore, system-wide coordination of power-performance states is necessary to maintain system balance and achieve acceptable per-query latency.

The rest of this paper is organized as follows. In Section 2, we discuss the unique aspects of OLDI services, how these impact power management, and prior work. To identify opportunities for power management, we present our cluster-scale Web Search characterization in Section 3. In Section 4, we develop and validate a Web Search performance model to determine which of these opportunities are viable from a latency perspective, and draw conclusions from this model in Section 5. Finally, in Section 6, we conclude.

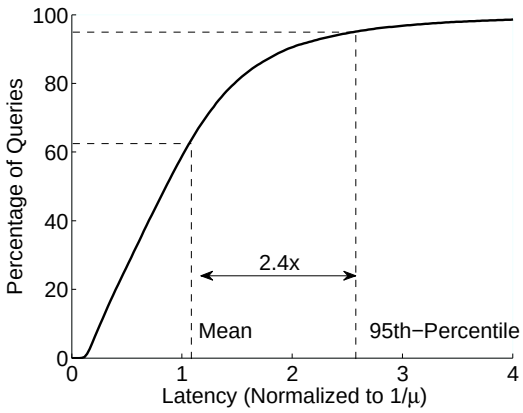


Figure 2: Example leaf node query latency distribution at 65% of peak QPS.

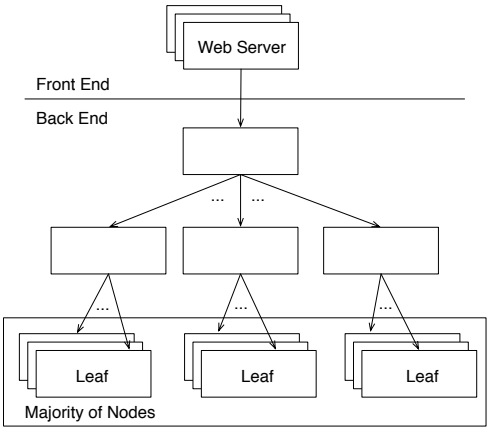


Figure 3: Web Search cluster topology.

2. BACKGROUND

Online data-intensive services are a relatively new workload class and have not undergone an extensive characterization study in the systems community. A defining feature of these workloads is that latency is of the utmost importance. User experience is highly dependent on system responsiveness; even delays of less than 400 ms in page rendering time have a measurable impact on web search queries per user and ad revenue [24]. Therefore, search operators focus not only on the mean but also on the tail of the latency distribution, and express targets for the 95th or 99th latency percentile of the distribution. Achieving such tail latency targets is challenging since queries have varying computing demands. Figure 2 shows a cumulative distribution of query latency (normalized to the mean) for a web search cluster operating at 65% of its peak capacity. There is a 2.4x gap between mean and 95th-percentile latency. The relevance of tail latencies is an additional challenge for power management schemes as many of them affect tail latencies more than they affect the mean.

2.1 Understanding OLDI Services

Cluster Processing in Web Search. The sheer size of the Web index and the complexity of the scoring system requires thousands of servers working in parallel to meet latency requirements. Figure 3 depicts the distributed, multi-tier topology of a Web search cluster [4]. Queries arrive at a Web server front-end, which forwards them to a large collection of *leaf* nodes using a distribution tree of intermediary servers. The index is partitioned (i.e., sharded) among all the leaf nodes, so that each performs the search on its index shards. There may be some shard replication across machines but it tends to be limited given DRAM costs. As a result, every leaf node is involved in processing each search query to maximize result relevance. The query distribution tree aggregates all results and returns the highest-scoring ones back to the front-end server.

Leaf nodes are the most numerous in a search cluster. Because of the high fan-out of the distribution tree, leaf nodes account for both the majority of query processing and cluster power consumption. Hence, we restrict our study to power savings for these servers. Note, however that the importance of the 95% latency becomes even more relevant when studying performance at the leaf-node level, since the entire service latency can be dominated by the latency of a single slow leaf. Although the effect of slow leaf nodes can be limited using latency cutoffs, in practice these tend to be conservative—prematurely terminating a search can compromise query result relevance.

Though other data-intensive workloads share similarities with OLDI services, there are some critical differences that make OLDI a particularly challenging target for power and energy optimizations. For example, personalized services such as Web mail and social networking also manipulate massive amounts of data and have strict latency requirements for user interaction. A key difference between these workloads and OLDI services is the fraction of data and computing resources that are used by every single user request. For a Web search cluster, a query typically touches all of the machines in the cluster. (Efficient index structures tend to be partitioned in the document identifier space. Therefore a query term can find document results in every single partition, requiring queries to be broadcast to every leaf node.) By contrast, a Web mail request will access primarily the users' active mailbox and a small fraction of the machines in the serving cluster. MapReduce-style computations can touch all of the machines and data in a cluster, but their focus on throughput makes them more latency tolerant, and therefore an easier target for energy optimizations, which tend to increase latency.

Diurnal Patterns. The demands of an OLDI service can vary drastically throughout the day. The cluster illustrated in Figure 1 is subject to load variations of over a factor of four. To simplify our discussion, we will focus on three traffic levels based on percentage of peak achievable traffic: low (20%), medium (50%) and high (75%). We choose 75% of peak as our high traffic point because, although clusters can operate near 100% capacity when needed, it is common for operators to set normal maximum operating points that are safely below those peaks to ensure service stability.

2.2 Taxonomy of Low-Power Modes

Component-level low power modes fall into two broad classes, *idle low-power modes*, and *active low-power modes*. We briefly discuss each and introduce a generic model for describing an idealized low-power mode that encompasses both classes. Figure 4 illustrates a wide variety of active and idle low-power modes and classifies each according to both the spatial (fraction of the system) and temporal (activa-

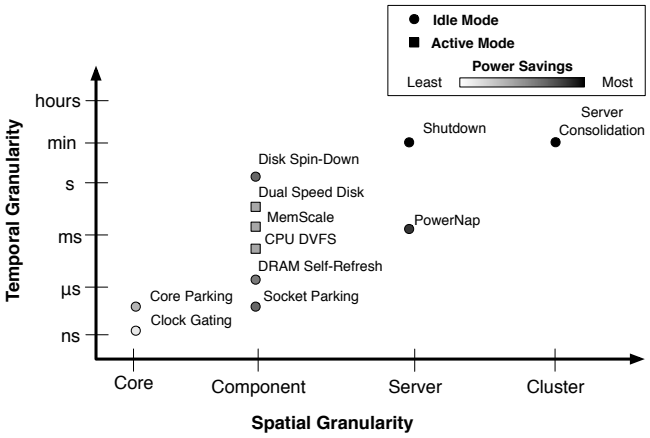


Figure 4: Low-power mode taxonomy. Modes with the greatest power savings must be applied at coarse granularity. Modes that apply at fine granularity generally yield less savings.

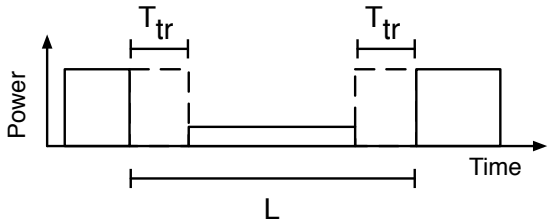


Figure 5: Idealized low-power mode. L is the length of the idle period and T_{tr} is the time required to transition in and out of the low-power state.

tion time scale) granularity at which each operates. Darker points are used for modes that provide greater relative power savings. Coarser modes tend to save more power, but are more challenging to use because opportunity to use them is more constrained.

Idle Low-Power Modes. Idle low-power modes save power during periods of full inactivity (e.g., ACPI C-states, DRAM power-down, and disk spin-down); no processing can proceed while they are in use. Because these modes can power off support circuitry (e.g., clocks, buses, etc.) that must remain powered when the components are active, they often have the greatest power savings. For example, PowerNap [20] reduces a server's power consumption by as much as 95%. The main challenge in using these modes is finding enough long periods of inactivity. Idle low-power modes are characterized by a transition latency to enter and leave the mode. Some components offer multiple modes with increasing transition latency and power savings (e.g., ACPI C-states).

Active Low-Power Modes. Active low-power modes throttle performance in exchange for power savings, which potentially extends the time necessary to complete a given amount of work, but still allows processing to continue. Their power savings potential is often limited by support circuitry that must remain powered regardless of load. Some components offer multiple active low-power modes with increasing performance reduction and power savings (e.g., ACPI P-states).

Abstract Power Mode Model. Figure 5 illustrates the abstract model we use to determine upper bounds on power savings. A power mode can be activated for a period L in which utilization is less than or equal to a threshold U. For idle low-power modes, U is zero. For active low-power modes, U is bounded by the slowdown incurred when operating under the low power mode. For example, a low power mode that incurs a factor of two slowdown may not be used when $U > 50\%$ as the offered load would then exceed the processing capacity under the mode. To save power, a component must transition in/out of a mode with latency T_{tr}.

2.3 Related Work

Numerous studies examine power management approaches for processors [12, 13, 23, 25], memory [9, 10, 11, 13], and disk [7, 15, 22]. A detailed survey of CPU power management techniques can be found in [18]. Others consider system-level mechanisms to mitigate the lack of component energy-proportionality [20, 27]. Our study builds on this literature by quantifying the opportunity (or lack thereof) for these classes of techniques in OLDI services, which, because of their latency sensitivity, bring a unique set of power management challenges.

Many authors have examined cluster-level approaches, such as dynamic resizing, load dispatching, and server/virtual machine consolidation, to achieve energy-proportionality by powering down unused systems (e.g., [8, 16, 27]). As noted in Section 2, the distributed data sets and latency constraints of OLDI services make such approaches difficult or impossible to apply to this workload class.

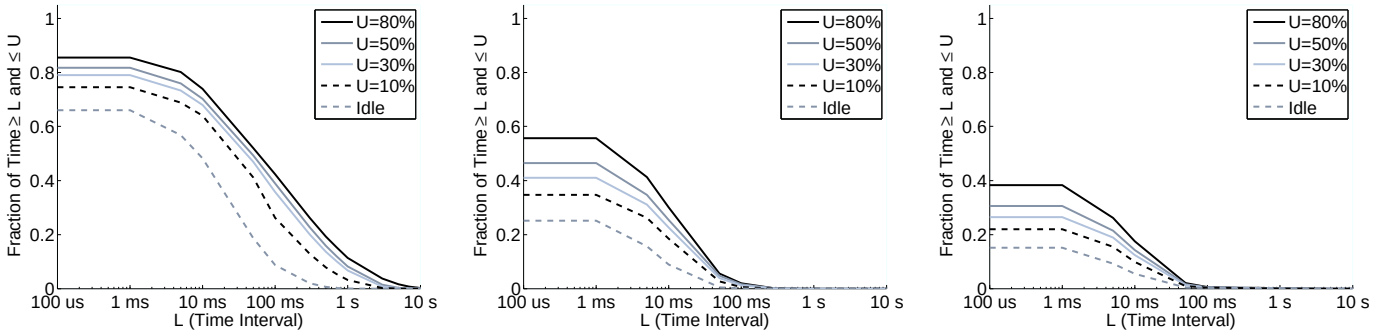
Recently, many studies have looked at using low-power, low-cost server components to improve energy-efficiency for data center workloads [19, 28, 29]. We take an approach that is agnostic of "small" or "big" cores. Previous studies have focused on improving the *peak* efficiency of server systems. Instead we seek server-level energy-proportionality through low-power modes to save power during non-peak periods.

3. CLUSTER-SCALE CHARACTERISTICS

In this section, we characterize the Web Search workload at 1000-node cluster scale. Opportunities for power savings often occur at very fine time-scales, so it is important to understand the magnitude and time-scale at which components are utilized [20]. We present our characterization, and analyze its implications on opportunity for power management, using a new representation, which we call *activity graphs*. Our characterization allows us to narrow the design space of power management solutions to those that are applicable to OLDI services. In Section 5 and 6, we further analyze promising approaches from a latency perspective.

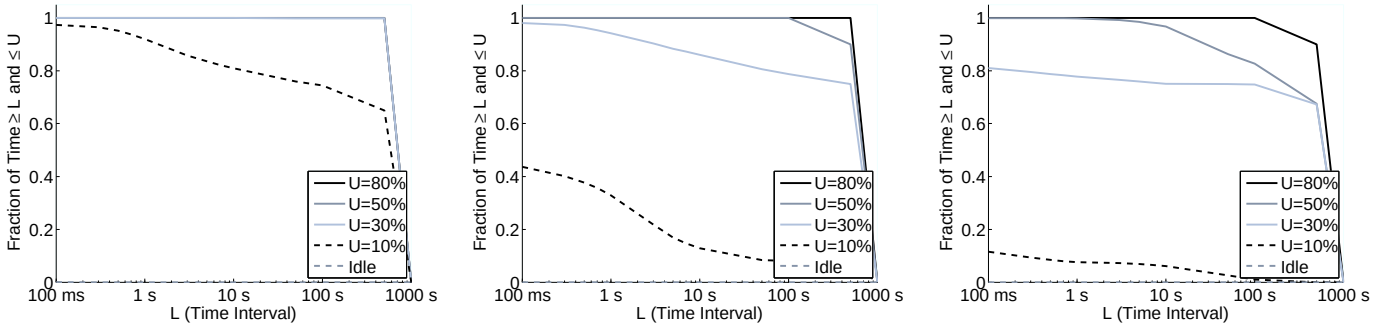
3.1 Experimental Methodology

We collected fine-grained activity traces of ten machines from a Web search test-cluster with over a thousand servers searching a production index database. The test workload executes a set of anonymized Web Search queries, with the query injection rate tuned to produce each of the QPS loads of interest (low, medium, and high). The behavior of the test runs was validated against live production measurements, however the latency and QPS levels reported here



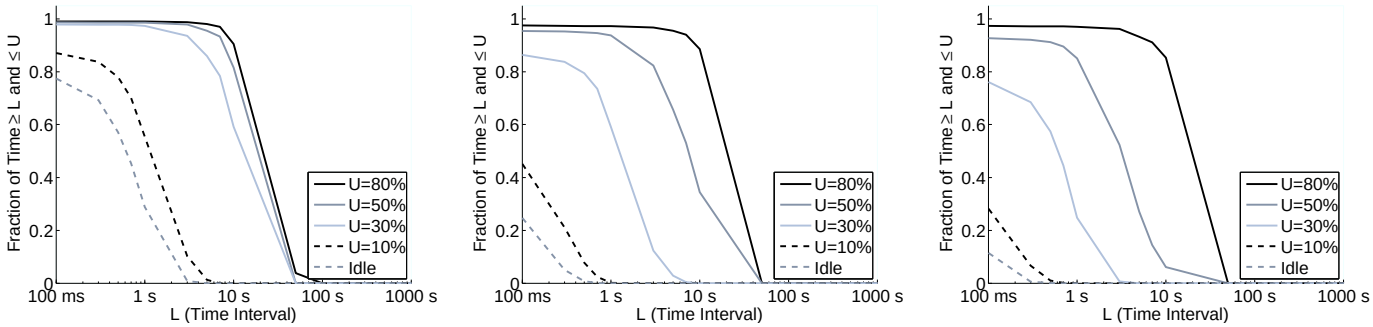
(a) 20% QPS (b) 50% QPS (c) 75% QPS

Figure 6: CPU activity graphs. Opportunities for CPU power savings exist only below 100 ms regardless of load.



(a) 20% QPS (b) 50% QPS (c) 75% QPS

Figure 7: Memory activity graphs. Memory bandwidth is undersubscribed, but the sub-system is never idle.



(a) 20% QPS (b) 50% QPS (c) 75% QPS

Figure 8: Disk activity graphs. Periods of up to tens of seconds with moderate utilization are common for disks.

do not correspond exactly to any particular Google search product—there are multiple production search systems at Google and their exact behavior is a result of complex trade-offs between index size, results quality, query complexity and caching behavior. We are confident that the parameters we chose for our test setup faithfully represents the behavior and energy-performance trade-offs of this class of workloads.

We collect raw utilization metrics for CPU, memory bandwidth, and disk I/O bandwidth with in-house kernel and performance counter-based tracing tools that have a negligible overhead and do not have a measurable impact on application performance. CPU utilization is aggregated across all the processors in the system. Our memory and disk instrumentation computes utilization as the fraction of peak bandwidth consumed over a given window; our resolution is limited to 100 ms windows.

3.2 Activity Graphs

Activity graphs compactly describe the opportunity for power savings at a range of utilizations and time-scales of interest. Figures 6-8 show the activity graphs for a Web search workload at multiple traffic levels (denoted by queries-per-second or QPS). These graphs depict a function $A(L, U)$ that gives the fraction of time a component spends at or below a given utilization U for a time period of L or greater:

$$A(L, U) = Pr(l \geq L, u \leq U) \quad (1)$$

This formulation allows us to determine the fraction of time

where any particular power mode might be useful. Perhaps as importantly, it lets us quickly rule out low-power modes that are inapplicable. We assume that an oracle guides transitions in and out of the mode at the start and end of each period L. Most modes do not save power during this transition and may halt processing during transitions; accordingly, for our study we assume that T_{tr} must be an order of magnitude less than L for a low-power mode to be effective.

Example: Reading Figure 6(a). We now demonstrate how to interpret the results of an activity graph, using Figure 6(a) as an example. Suppose we wish to evaluate a CPU active low-power mode that incurs a 25% slowdown (i.e., it can only be used below $1/1.25 = 80\%$ utilization) and has a transition time of 1 ms (i.e., it is only useful for L of about 10 ms or greater). We can see that at 20% QPS, this mode is applicable nearly 80% of the time. Conversely, if transitions latency limited this mode to periods of 1 second or greater, there would be almost no opportunity for the mode.

3.3 Characterization Results

We now discuss the activity graphs shown in Figures 6, 7, and 8, for CPU, memory and disk, respectively.

CPU. Figure 6 show activity graphs for CPU. There is almost no opportunity for CPU power management at a time-scale greater than one second even at low loads; instead power modes must be able to act well into the sub-second granularity to be effective. Perhaps most interestingly, the 1 ms time scale captures nearly all the savings opportunity regardless of the utilization threshold, suggesting that it is unnecessary to design modes that act on granularities finer than 50-100μs. Additionally, while increased QPS reduces the overall opportunity for power management at each utilization level, it does not change the time-scale trends.

Memory. Figure 7 presents the activity for memory. Two features are prominent. First, we observe that the memory bus is greatly underutilized, with many long periods during which utilization is below 30%. Hence, active low power modes that trade memory bandwidth for lower power (e.g., [10]) are applicable even if they have relatively large transition times. Second, the memory bus is never idle for 100ms or longer. Unfortunately, our instrumentation cannot examine sub-100ms granularities. However, by estimating finer-grained memory idleness from our CPU traces (by assuming the memory system is idle when all CPUs are idle), we find that idle periods longer than 10 μs are scarce even at low QPS values. We conclude that DRAM idle low-power modes require sub-μs transitions times to be effective for these workloads, which is an order of magnitude faster than currently available DRAM modes.

Disk. Finally, Figure 8 shows the activity graphs for disk. The activity trends for disk differ from CPU and memory because varying QPS shifts the time-scales over which utilization levels are observed. Whereas the majority of time spent at or below 30% utilization is captured in 10-second intervals at 20% QPS, this opportunity shifts to an order of magnitude finer time-scale when load increases to 75% QPS. As with memory, disk bandwidth is not heavily utilized even at high QPS; the majority of time is spent below 50% utilization even at a 1 second granularity. Unlike memory, disk exhibits periods of idleness at 100 ms to 1 s time-scales, especially for low QPS.

A particularly interesting property of disk activity patterns is that disks rarely see peak utilization—moderate utilization levels persist for long periods. This observation suggests that active low-power modes that provide less than 2-3x performance degradation (i.e., that can be used in periods of 30-50% utilization) at time scales of a few seconds may be quite useful. Unfortunately, most available or proposed power modes for disks act on time scales substantially greater than one second [15, 22]. Accordingly, this data suggests that the main challenge in saving disk power lies in reducing power mode transition time. Alternatively, a non-volatile storage solution that could trade bandwidth for power would be useful.

Table 1: Low-power mode characteristics.

Power Mode	T_{tr}	$u_{\text{threshold}}$	$\frac{\Delta P_{\text{Mode}}}{P_{\text{Nominal}}}$	Ref.
C1E $\rightarrow$ ACPI C3	10 μs	Idle	2%	[2]
C1E $\rightarrow$ ACPI C6	100 μs	Idle	44%	[2]
Ideal CPU V_{dd} Scaling	10 μs	50%	88%	[2]
Ideal Mem. V_{dd} Scaling	10 μs	50%	88%	[17]
Dual-Speed Disk	1 sec	50%	59%	[22]
Disk Spin-Down	10 sec	Idle	77%	[7]

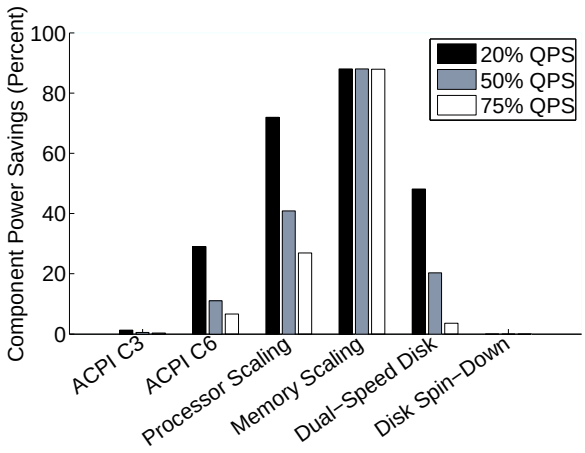
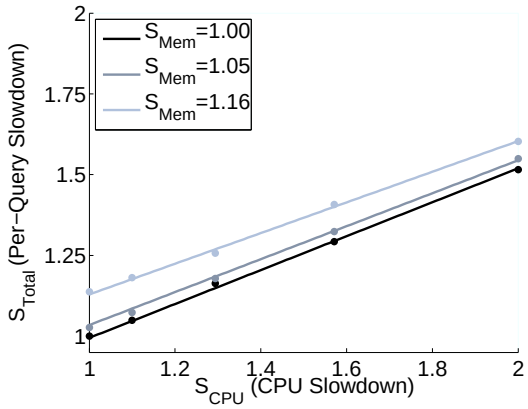


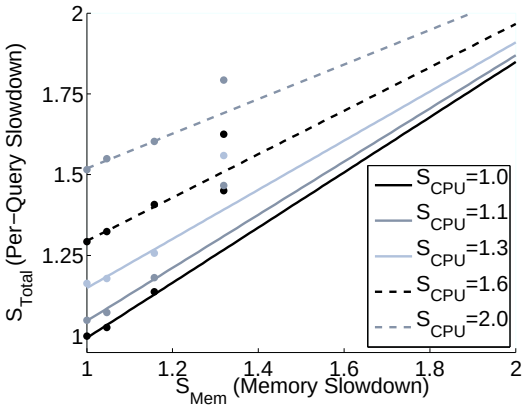
Figure 9: Power savings potential for available low-power modes.

3.4 Analysis: Evaluating Low-Power Modes

Using the activity graphs, we now examine power-savings effectiveness, that is, the product of applicability and power reduction, for existing and proposed power modes. Table 1 lists the idle and active power modes we consider for CPU, memory, and disk, as well as our assumptions for transition time (T_{tr}), utilization threshold ($u_{\text{Threshold}}$), and normalized power savings, while using a power mode ($\frac{\Delta P_{\text{Mode}}}{P_{\text{Nominal}}}$). We use these parameters to derive an estimate of the potential per-component power savings at each QPS level. Here P_{Nominal} is the power consumed under normal operation and P_{Mode} the power when the mode is active. For the ideal V_{dd} scaling modes, we assume that $\frac{P_{\text{Mode}}}{P_{\text{Nominal}}} = \left(\frac{f}{f_{\text{Max}}}\right)^3$, which is an optimistic upper bound on the potential of frequency and voltage scaling. Note that, to our knowledge, no current memory device can scale supply voltage during operation; however, we include it as a proxy for other potential memory active low-power modes. Disk active mode savings are taken from [22]. The quantity $\frac{P_{\text{Mode}}}{P_{\text{Nominal}}}$ for idle power modes is simply the ratio of the idle mode power to the power at



(a) Average query slowdown as a function of CPU slowdown (holding memory slowdown constant).



(b) Average query slowdown as a function of memory slowdown (holding CPU slowdown constant). Data points at $S_{mem} = 1.3$ exhibit test-harness-specific memory bandwidth effects and are excluded from the regression.

Figure 10: Per-query slowdown regression. Dots represent measured values; lines show values predicted by the regression model.

idle without the mode. We calculate the normalized per-component power savings using the following model:

$$\frac{P_{\text{Saving}}}{P_{\text{Nominal}}} = \frac{P_{\text{Nominal}} - (1 - A(L,U)) \cdot P_{\text{Nominal}} + A(L,U)) \cdot P_{\text{Mode}}}{P_{\text{Nominal}}}$$
$$= 1 - ((1 - A) + A \cdot \frac{P_{\text{Mode}}}{P_{\text{Nominal}}}) = A \cdot \frac{\Delta P_{\text{Mode}}}{P_{\text{Nominal}}} \quad (2)$$

Figure 9 reports the power savings opportunity for each low-power mode at each QPS level. The ACPI C3 state provides minimal opportunity because it does not lower CPU power much below the savings already achieved by C1E, which current processors automatically enter upon becoming idle [2]. On the other hand, ACPI C6 (also referred to as power-gating or core-parking for the Nehalem architecture) can provide moderate savings, especially at low-utilization. Using scaling for processor and memory provides substantial opportunity for benefit. Though the opportunity for CPU scaling decreases with increasing load, the opportunity for memory scaling remains large. Disk spin-down is inapplicable to this workload due to its prohibitive transition latency. However, a dual-speed disk has moderate power savings potential, especially at low QPS. Improving transition latency could further increase the opportunity for dual-speed disks, but we leave this study for future work.

4. LEAF NODE PERFORMANCE MODEL

The activity graphs reveal the potential of low-power modes with respect to throughput constraints, but do not consider whether their impact on service latency will be acceptable. As noted in Section 2, OLDI services place tight constraints on 95th-percentile latency. To predict impacts on query latency, in this section, we develop a performance model that relates per-component slowdowns to the overall query latency distribution. Our validation against hardware measurements shows that our model estimates 95% query latency to within 9.2% error. In Section 5, we will apply this model to estimate the power-performance pareto frontiers of a variety of current and hypothetical CPU and memory low power modes for various *service level agreement* (SLA) targets on 95th-percentile query latency. We elected not to explore disk power/performance trade-offs based on the results in Section 3.3.

Query latency is composed of time a query spends in service and the time it must wait to receive service:

$$L_{\text{Query}} = L_{\text{Service}} + L_{\text{Wait}} \quad (3)$$

We first construct a model to predict the impact of reduced component performance on L_{Service}, the expected time to execute a single query without queuing. We then incorporate the service model into a queuing model to estimate L_{Wait} and determine the total latency, L_{Query} as a function of low-power mode settings and contention.

4.1 Modeling L_{Service}

Rather than model specific low power modes, we instead capture directly the relationship between per-component performance and query service time; our model can then be applied to existing/hypothetical per-component low power modes by measuring/assuming a specific power-latency trade-off for the mode. We develop the query service time model empirically by measuring the impact of per-component slowdowns in a controlled test environment. We then perform a linear regression on the measured performance results to derive the overall service time relationship.

We replicate the behavior of a production leaf node on a 16-core x86 server with 32GB of memory. Our test harness reproduces an arrival process, query set, and web index comparable to production environments. Our test server allows us to vary the CPU and memory performance and measure their impact on average query execution time. We vary processor performance using frequency scaling; we scale all cores together at the same clock frequency.

The most effective method to vary memory latency in our test environment is to decrease the interconnect speed between the processor and DRAM. Varying interconnect speed has the potential to introduce both latency and bandwidth effects. To construct a platform agnostic model, we wish to exclude bandwidth effects (which are specific to the memory system implementation of our test harness) and capture

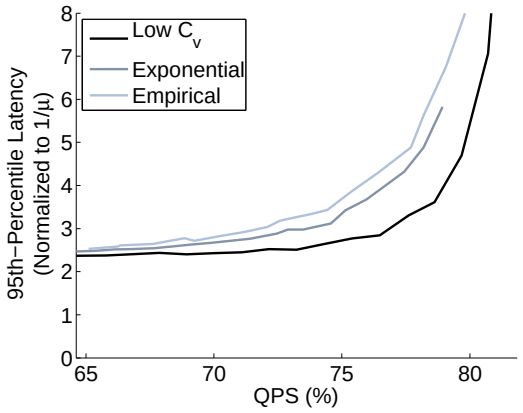


Figure 11: Arrival process greatly influences quantile predictions.

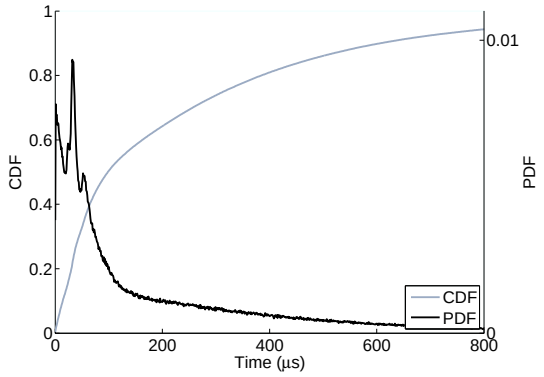


Figure 12: Distribution of time between query arrivals.

only latency effects in our model; the latency model can be augmented with bandwidth constraints as appropriate. Bandwidth effects are generally small at low utilization [26]. However, for this workload and test system, we have determined that memory bandwidth becomes a constraint at the slowest available interconnect speeds. Hence, we have chosen to exclude these settings from our regression, but report the measured values in our figures for interested readers. We also include measurements from one of the excluded settings in our validation to expose these bandwidth-related queuing effects.

Given component slowdowns S_{CPU} and S_{Mem} (the relative per-query latency increase in the processor and memory) we would like to determine the function $S_{\text{Total}} = f(S_{\text{CPU}}, S_{\text{Mem}}) = \frac{L_{S_{\text{CPU}}, S_{\text{Mem}}}}{L_{\text{Nominal}}}$ where S_{Total} is the overall per-query slowdown.

First, we measured the latency of queries delivered to the leaf node one at a time (hence eliminating all contention effects). This experiment confirmed our intuition that the relationship between CPU and memory slowdown and per-query slowdown is linear, providing a sanity check on our use of a linear regression model.

Next we measured the $S_{\text{Total}}, S_{\text{CPU}}, S_{\text{Mem}}$ triple for each available processor and memory setting while loading the leaf node to capacity, but only issuing new queries once outstanding queries complete. This experiment captures the interference effects (e.g., in on-chip caches) of concurrent queries while isolating service latency from queuing latency. Because of the previously-discussed bandwidth effects, we can use only three memory-link-speed settings to approximate the slope of the S_{Mem} relationship. Whereas the absolute accuracy of S_{Total} estimates will degrade as we extrapolate S_{Mem} beyond our measured results, we nevertheless expect that general trends will hold.

We empirically fit the measured data using a linear regression on an equation with the form:

$$S_{\text{Total}} = \mathbf{S}\boldsymbol{\beta}, \quad \boldsymbol{S} = [1,\ S_{\text{CPU}},\ S_{\text{Mem}},\ S_{\text{CPU}} \cdot S_{\text{Mem}}] \\ \boldsymbol{\beta} = [\beta_0,\ \beta_1,\ \beta_2,\ \beta_3]^T \tag{4}$$

Using a least-squares regression, we find that the $\boldsymbol{\beta}$ vector has the values:

$$\boldsymbol{\beta} = [-0.70,\ 0.84,\ 1.17,\ -0.32]^T \tag{5}$$

This formulation predicts S_{Total} with an average error of 1% and a maximum error of 1.2%.

The values of the β parameters yield intuition into the workload's response to component slowdown. The β_1 and β_2 values show a strong linear relationship between application slowdown and CPU and memory slowdown. Interestingly, the negative value of β_3 indicates that slowing CPU and memory together may be beneficial.

Figures 10(a) and 10(b) provide two views of how CPU and memory slowdown affect average query latency. They further illustrate how our regression predictions (represented by lines) compare to measured values (represented by points).

4.2 Modeling L_{Wait}

To capture query traffic and contention effects, we augment our per-query service time model with a queuing model describing the overall behavior of a leaf node. We model the leaf node as a G/G/k queue—a system with an arbitrary interarrival and service time distribution, average throughput λ (in QPS), average service rate μ (in QPS), k servers, average load $\rho = \frac{\lambda}{\mu \cdot k}$ and a unified queue. Whereas this model does not yield a closed-form expression for latency, it is well-defined and straight-forward to simulate. We use the approach of [21] to simulate the leaf-node queuing system.

Interarrival Distribution. We have found that the distribution of query arrival times at the leaf node can greatly influence 95th-percentile latencies. Naïve loadtesting clients tend to send queries at regular intervals, resulting in fewer arrival bursts and fewer, longer idle intervals than what is seen with actual user traffic. Figure 11 shows how different assumptions for interarrival distribution influence predictions for 95th-percentile latency; the difference is most pronounced at moderate QPS. We have found that actual user traffic tends to show higher coefficient of variation (C_v = 1.45) than an exponential distribution and considerably higher than a naïve (Low C_v) arrivals. We have modified our loadtesting client to replay the exact distribution seen by live traffic and were able to validate these effects. Our validation demonstrates the perils of using naïve loadtesting clients, and suggests that in absence of real user traces, exponential distributions could be used with reasonably small errors.

Figure 12 depicts the production system arrival distribution at a leaf node. The interarrival distribution has distinguishing notches at $.007(1/\mu)$ and $.01(1/\mu)$. This shape is an

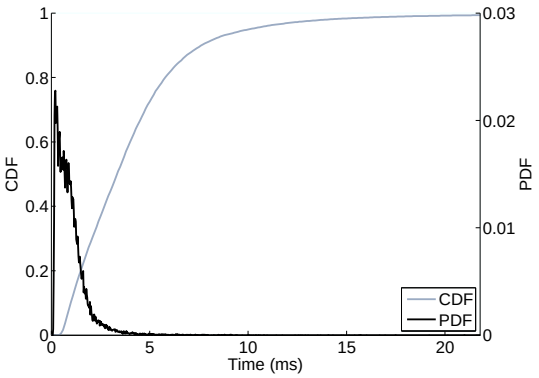


Figure 13: Distribution of query service times.

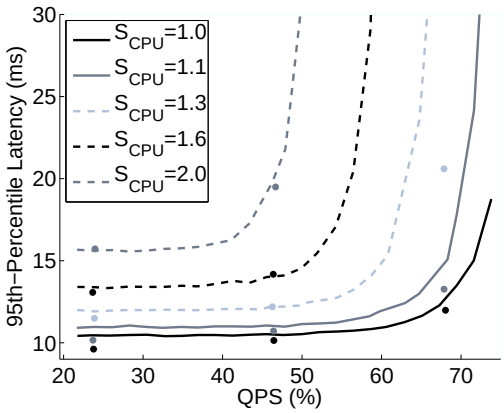


Figure 14: Performance model validation: Dots and lines repesent measured and modeled values, repsectively.

artifact of the multi-tier server topology shown in Figure 3. We have also observed that the shape of the interarrival distribution does not change significantly with varying QPS, only its scale. This property allows us to use the distribution measured at a single QPS and scale it to achieve an arbitrary throughput for our model.

Service Distribution. The service time distribution describes the amount of time spent executing each query. It is parameterized by μ, the average service rate (in QPS); $1/\mu$, the average query execution time, is given by the L_{Service} model described in Section 4.1. Since multiple queries are serviced concurrently, the aggregate service rate is $k \cdot \mu$. In our case, k is 16.

The service time distribution measured is shown in Figure 13. Though the variance of this distribution is not particularly high (C_v=1.12), the 95th-percentile of the distribution is nearly 3 times greater than the mean. This variability is largely intrinsic to the computing requirements of each individual query.

The service distribution shape, as with the interarrival distribution, does not change significantly with average query latency changes, allowing us to model power modes' effects as a service rate modulation, $\mu' = \mu/S_{\text{Total}}$.

Autocorrelation. Our simulation approach [21] generates interarrival and service times randomly according to the scaled empirical distributions, which assumes that the arrival/service sequences are not autocorrelated (i.e., consecutive arrivals/services are independent and identically-distributed). We have validated this assumption in traced arrival and service time sequences (there is less than 5% autocorrelation on average). This independence is important to establish, as it allows us to generate synthetic interarrival/service sequences rather than replaying recorded traces.

4.3 Validation

To validate our performance model, we measured the performance of the Web Search workload at the three QPS levels of interest and all applicable performance settings. Figure 14 compares the 95th-percentile latency predicted by our model (lines) against measured values on our test node (points). At low QPS, the predicted latency is primarily a function of the L_{Service} performance model, as queuing rarely occurs. As QPS reaches the maximum level, queuing effects and our model for L_{Wait} increase in importance; our overall model predicts 95th-percentile latency accurately in both regions, achieving an average error of 9.2%.

Table 2: Typical Server Power Breakdown.

Power (% of Peak)	CPU	Memory	Disk	Other
Max	40%	35%	10%	15%
Idle	15%	25%	9%	10%

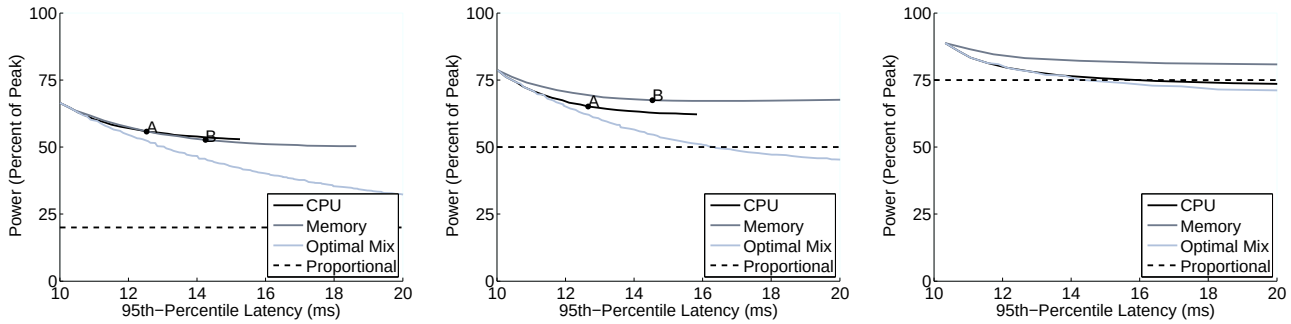
5. LATENCY-POWER TRADEOFFS

Our performance model allows us to quantify the tradeoffs between power and latency for both proposed and currently available power modes for our OLDI workload. In this section, our analysis will draw the following conclusions: (1) Full-system, coordinated low-power modes provide a far better latency-power tradeoff than individual, uncoordinated modes. (2) Idle low-power modes do not provide significant marginal power savings over C1E. (3) There is a need to develop full-system, coordinated active low-power modes, because full-system idleness is difficult to find.

We assume similar server power breakdown and power scaling as in [3]; these assumptions are summarized in Table 2. In this analysis, we consider processor and memory voltage and frequency scaling, processor idle low-power modes, and batching queries to create and exploit full-system power modes (i.e., PowerNap [20]).

5.1 Active Low-Power Modes

First, we investigate the applicability of active low-power modes for CPU and memory; in particular, we assess the power savings of frequency and voltage scaling. In an ideal CMOS processor, voltage scales proportionally to frequency. Accordingly, power consumption is reduced cubically with respect to processor frequency ($P \propto f^3$). In reality, however, reducing processor frequency does not allow an equivalent reduction in processor voltage [6, 25]. Divergence from the ideal model is due to the inability to scale voltage with the required ideal linear relationship to frequency, and the growing static power consumed by modern processors. This divergence is much more pronounced in high-end server processors than low-end processors, because by definition they are optimized for performance (low threshold voltage and



(a) 20% QPS. (b) 50% QPS. (c) 75% QPS.

Figure 15: System power vs. latency trade-off for processor and memory scaling ($P \propto f^{2.4}$): The point "A" represents $S_{\text{CPU}} = 1.5$ and "B" represents $S_{\text{Mem}} = 1.5$. "A" and "B" do not appear in graph (c) because the latency exceeds 20 ms.

high supply voltage) rather than power (high threshold voltage and low supply voltage).

In addition, limited voltage scaling and automated idle power modes (e.g., C1E) lead to essentially constant idle power, no matter what voltage/frequency mode is selected. To provide an optimistic scenario, we evaluate the expected power savings assuming ($P \propto f^{2.4}$), which was derived from an embedded processor [6] and matches theoretical predictions of practical scaling limits [30]. For this case, we also optimistically assume that the ratio of static power to active power remains constant. Data from today's high-end servers [1] suggests much more modest power savings associated with DVFS, with energy savings approaching zero for some DVFS modes.

Although full-featured voltage and frequency control (e.g., DVFS) is not currently available for modern memory systems, the power consumption of these devices follow similar relationships with respect to voltage and frequency [17]. We therefore hypothesize a memory power mode that resembles the optimistic processor frequency and voltage scaling scenario. Recent work [10] has already begun exploring active low-power modes for main memory.

Figure 15 shows total server power as a function of 95th-percentile latency for each QPS level. Three scenarios are considered: CPU scaling alone ("CPU"), memory scaling alone ("Memory") and a combination of CPU and memory scaling ("Optimal Mix"). Since there are many permutations of CPU/memory settings, we show only pareto-optimal results (best power savings for a given latency).

In isolation, the latency-power tradeoff for CPU scaling dominates memory scaling. Memory slowdown impacts overall performance more and saves less power at higher QPS. However, at 20% one can see that memory scaling may do better at times. Different server configurations may shift this balance; in particular, recent trends indicate that the fraction of power due to memory will increase. Our results show that coordinated scaling of both CPU and memory yields significantly greater power reductions at all target latencies than scaling either independently. This observation underscores the need for coordinated active low-power modes: components must scale together to maintain system balance. With our optimistic scaling assumption, we can achieve better–than–energy-proportional operation at 50% and 75% QPS. At 20% QPS, power consumption becomes dominated by other server components with large idle power (e.g., disk, regulators, chipsets, etc.).

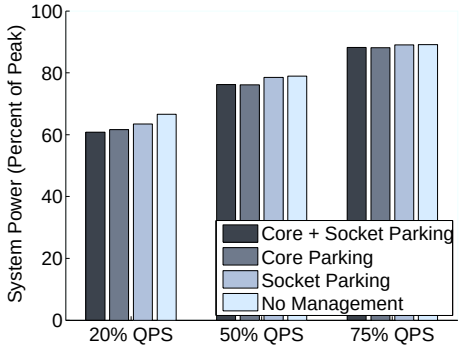


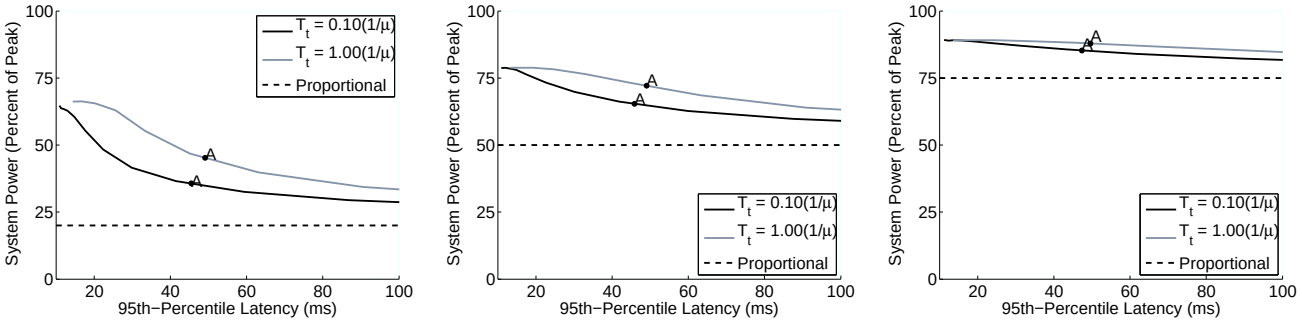
Figure 16: System power savings for CPU idle low-power modes: Core parking only yields marginal gains over C1E. Power savings from socket parking is limited by lack of per-socket idleness.

Table 3: Processor idle low-power modes.

	Power (% of Peak)			
	Active	Idle (HLT)	Parking	
			Core	Socket
Per-Core (x4)	20%	4%	0%	0%
Uncore	20%	20%	20%	0%

5.2 Processor idle low-power modes

Next, we explore idle CPU power modes, where the CPU enters a low-power state but other components remain active. Modern CPUs already use the ACPI C1 or C1E state whenever the HLT instruction is executed. We would like to understand the benefit of using deeper idle low-power modes. For comparison, we evaluate using ACPI C6, which can be applied either at the core or socket level. At the core level, ACPI C6 or "Core Parking" uses power gating to eliminate power consumption for that core. It has a transition time of less than 100 μs [2]. At the socket level ("Socket Parking"), we assume that an integrated memory controller must remain active but all caches may be powered down yielding a 50% reduction in non-core power consumption. This mode requires socket-level idleness and incurs a 500 μs transition time. For both modes, we assume the mode is invoked immediately upon idleness and that transition to active occur immediately upon a query requiring the resource. Socket power consumption in each state is summarized in Table 3.



(a) 20% QPS. (b) 50% QPS. (c) 75% QPS.

Figure 17: System power vs. latency trade-off for query batching with PowerNap: "A" represents a batching policy that holds jobs for periods equal to 10x the average query service time.

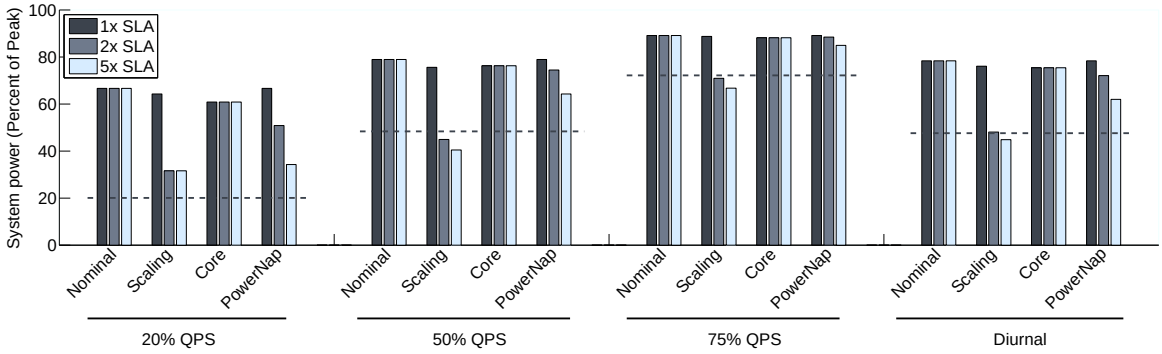


Figure 18: Power consumption at each QPS level for a fixed 95th-percentile increase: The dotted line at each QPS level represents the power consumption of an energy proportional system."Diurnal" represents the time-weighted daily average from Figure 1. An energy-proportional server would use 49% of its peak power on average over the day.

Figure 16 depicts the power savings opportunity for processor idle low-power modes. The transition time of these modes is sufficiently low that the 95th-percentile is not affected significantly. We observe that ACPI C6 applied at the core level ("Core Parking"), socket level ("Socket Parking") or in combination ("Core + Socket Parking") does not provide a significant power reduction. Since modern processors enter C1E at idle (i.e., the HLT instruction), the idle power of the processor is not high. Therefore, compared to a baseline scenario ("No Management"), using core or socket parking does not provide much benefit. Larger power savings are likely possible if work is rescheduled amongst cores to coalesce and thus extend the length of the idle periods, enabling the use of deeper idle power modes.

5.3 Full-system Idle Power Modes

Finally, we investigate using a full-system idle low-power mode to recover power consumed by all server components. We evaluate PowerNap, which provides a coordinated, full-system low-power mode with a rapid transition time to take advantage of idle periods and has shown great promise for non-OLDI server applications [20]. We assume that while in PowerNap mode, the server consumes 5% of peak but cannot process queries.

It is extremely rare for full-system idleness to occur naturally for our workload, even at low QPS values. Instead, we coalesce idle periods by batching queries to the leaf node. Queries are accumulated at a higher level node in the search tree and released after a given timeout to the leaf node. Increasing the timeout allows for longer periods in the PowerNap mode. Our approach is similar to that of [12].

Figure 17 shows the power-latency tradeoff given a transition time, T_t, of both one tenth and equal to the average query processing time ($1/\mu$). (Note the larger scale of the horizontal axis in this figure relative to Figure 15). We find that in order to save an appreciable amount of power, PowerNap requires relatively long batching intervals incurring large latency penalties. The point ("A") represents a batching policy with a timeout period of 10x the average query service time. Even with a relatively short transition time, the latency-power trade-off is not enticing across QPS levels.

5.4 Comparing Power Modes

Out of the power modes explored thus far, we would like to identify the best mode given an SLA latency bound. For this evaluation, we define our initial SLA to be the 95th-percentile latency at 75% QPS. Figure 18 compares the power saving potential of system active low-power modes ("Scaling") from Section 5.1, processor idle low-power modes ("Core") from Section 5.2, and system idle low-power modes ("PowerNap") from Section 5.3 given a latency bound of 1x, 2x, and 5x of this initial SLA. At 20% QPS, none of the power management schemes can achieve energy-proportionality (a power consumption of 20% of peak) even at a 5x latency increase, although scaling falls just short. For 50% and 75% QPS, coordinated scaling can achieve slightly better than energy proportional power consumption for a 2x latency increase, but other techniques cannot regardless of SLA.

To understand the overall daily potential for power savings, we show the time-weighted average power consumption ("Diurnal") using the QPS variations shown in Figure 1. Once again, only scaling is able to achieve a power con-

sumption at or better than energy proportional operation for a 2x or greater latency increase. This result supports our claim that OLDI services can achieve energy-proportionality only with coordinated, full-system scaling active low-power modes; other power management techniques simply do not provide a suitable power-latency tradeoff for the operating regions of this workload class.

6. CONCLUSION

Our work explores the power management opportunities of OLDI services—an application class that is central to the success of online Internet services, and yet presents formidable power management challenges. These workloads are highly latency sensitive, and their data set usage does not scale down with traffic, making it infeasible to simply turn off machines during off-peak periods. Our two studies, a cluster-scale examination of per-component utilization and a validated model to examine the latency impact of low-power modes, provide a series of actionable conclusions to guide further development of server low-power modes. We find: (1) CPU active low-power modes provide the best *single* power-performance mechanism, but, by themselves, cannot achieve energy-proportionality; (2) there is a pressing need to improve idle low-power modes for shared caches and on-chip memory controllers, however, existing modes are sufficient at the cores; (3) there is substantial opportunity to save memory system power with active low-power modes during ample periods of underutilization, but there appears to be little, if any, opportunity for existing idle low-power modes; (4) even with query batching, full-system idle low-power modes cannot provide acceptable latency-power tradeoffs; and (5) coordinated, full-system active low-power modes hold the greatest promise to achieve energy-proportionality with acceptable query latency by maintaining system balance across the dynamic load range of the workload.

Acknowledgements. The authors would like to thank Paolo Ferraris, Dean Gaudet, Greg Johnson, Artur Klauser, Christian Ludloff, Mahesh Palekar, Jeff Raubitschek, Bianca Schroeder, Divyesh Shah, Arun Sharma, Sivagar Natarajan Sivagar, Gautham Thambidorai, and Tamsyn Waterhouse for their consultation and technical assistance in our experiments and the anonymous reviewers for their feedback. This work was supported in part by grants from Google and NSF grants CNS-0834403, CCF-0811320, and CCF-0815457.

7. REFERENCES

[1] "AMD Family 10h Server and Workstation Processor Power and Thermal Data Sheet Rev 3.15," 2010.
[2] "Intel Xeon Processor 5600 Series. Datasheet, Vol. 1," 2010.
[3] L. A. Barroso and U. Hölzle, *The Datacenter as a Computer*. Morgan Claypool, 2009.
[4] L. A. Barroso, J. Dean, and U. Hölzle, "Web search for a planet: The google cluster architecture," *IEEE Micro*, vol. 23, no. 2, 2003.
[5] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE Computer*, vol. 40 no. 12, 2007.
[6] D. Blaauw, S. Das, and Y. Lee, "Managing variations through adaptive design techniques," Tutorial at International Solid-State Circuits Conference, 2010.
[7] E. V. Carrera, E. Pinheiro, and R. Bianchini, "Conserving disk energy in network servers," in *Proc. International Conf. on Supercomputing*, 2003.
[8] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers," in *Symp. on Operating System Principles*, 2001.
[9] V. Delaluz, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam, and M. J. Irwin, "Hardware and software techniques for controlling DRAM power modes," *IEEE Trans. Comput.*, vol. 50, no. 11, 2001.
[10] Q. Deng, D. Meisner, T. F. Wenisch, and R. Bianchini, "MemScale: Active Low-Power Modes for Main Memory," in *Architectural Support for Programming Languages and Operating Systems*, 2011.
[11] B. Diniz, D. Guedes, W. Meira, Jr., and R. Bianchini, "Limiting the power consumption of main memory," in *International Symp. on Computer Architecture*, 2007.
[12] M. Elnozahy, M. Kistler, and R. Rajamony, "Energy conservation policies for web servers," in *Proc. USENIX Symp. on Internet Technologies and Systems*, 2003.
[13] X. Fan, C. S. Ellis, and A. R. Lebeck, "The synergy between power-aware memory systems and processor voltage scaling," in *Workshop on Power-Aware Computing Systems*, 2003.
[14] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *International Symp. on Computer Architecture*, 2007.
[15] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke, "DRPM: dynamic speed control for power management in server class disks," in *International Symp. on Computer Architecture*, 2010.
[16] T. Heath, B. Diniz, E. V. Carrera, W. Meira, Jr., and R. Bianchini, "Energy conservation in heterogeneous server clusters," in *Principles and Practice of Parallel Programming*, 2005.
[17] J. Janzen, "Calculating memory system power for DDR SDRAM," *Micron DesignLine*, vol. 10, no. 2, 2001.
[18] S. Kaxiras, M. Martonosi, *Computer Architecture Techniques for Power-Efficiency*. Morgan Claypool, 2009.
[19] K. Lim, P. Ranganathan, J. Chang, C. Patel, T. Mudge, and S. Reinhardt, "Understanding and designing new server architectures for emerging warehouse-computing environments," in *International Symp. on Computer Architecture*, 2008.
[20] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating server idle power," in *Arch. Support for Programming Languages and Operating Systems*, 2009.
[21] D. Meisner and T. F. Wenisch, "Stochastic Queuing Simulation for Data Center Workloads," in *Exascale Evaluation and Research Techniques Workshop*, 2010.
[22] E. Pinheiro and R. Bianchini, "Energy conservation techniques for disk array-based servers," in *International Conf. on Supercomputing*, 2004.
[23] K. Rajamani, C. Lefurgy, S. Ghiasi, J. Rubio, H. Hanson, and T. W. Keller, "Power management solutions for computer systems and datacenters," in *International Symp. on Low-Power Electronics and Design*, 2008.
[24] E. Schurman and J. Brutlag, "The user and business impact of server delays, additional bytes, and HTTP chunking in web search," *Velocity*, 2009.
[25] D. Snowdon, S. Ruocco, and G. Heiser, "Power Management and Dynamic Voltage Scaling: Myths & Facts," in *Power Aware Real-time Computing*, 2005.
[26] S. Srinivasan et al, "CMP memory modeling: How much does accuracy matter?" in *Workshop on Modeling, Benchmarking and Simulation*, 2009.
[27] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu, "Delivering energy proportionality with non energy-proportional systems – optimizing the ensemble," in *HotPower*, 2008.
[28] VJ Reddi, Benjamin Lee, Trishul Chilimbi, and Kushagra Vaid, "Web Search Using Mobile Cores: Quantifying and Mitigating the Price of Efficiency," in *International Symp. on Computer Architecture*, 2010.
[29] Willis Lang and Jignesh M. Patel and Srinath Shankar, "Wimpy Node Clusters: What About Non-Wimpy Workloads?" in *Workshop on Data Management on New Hardware*, 2010.
[30] F. Xie, M. Martonosi, and S. Malik, "Intraprogram dynamic voltage scaling: Bounding opportunities with analytic modeling," *Trans. Archit. Code Optim*, vol. 1, 2004.