Mobile Processors for Energy-Efficient Web Search

VIJAY JANAPA REDDI, University of Texas at Austin
BENJAMIN C. LEE, Duke University
TRISHUL CHILIMBI, Microsoft Research
KUSHAGRA VAID, Microsoft Corporation

As cloud and utility computing spreads, computer architects must ensure continued capability growth for the data centers that comprise the cloud. Given mega-Watt scale power budgets, increasing data center capability requires increasing computing hardware energy efficiency. To increase the data center's capability for work, the work done per Joule must increase. We pursue this efficiency even as the nature of data center applications evolves. Unlike traditional enterprise workloads, which are typically memory or I/O bound, big data computation and analytics exhibit greater compute intensity. This article examines the efficiency of mobile processors as a means for data center capability. In particular, we compare and contrast the performance and efficiency of the Microsoft Bing search engine executing on the mobile-class Atom processor and the server-class Xeon processor. Bing implements statistical machine learning to dynamically rank pages, producing sophisticated search results but also increasing computational intensity. While mobile processors are energy-efficient, they exact a price for that efficiency. The Atom is 5× more energy-efficient than the Xeon when comparing queries per Joule. However, search queries on Atom encounter higher latencies, different page results, and diminished robustness for complex queries. Despite these challenges, quality-of-service is maintained for most, common queries. Moreover, as different computational phases of the search engine encounter different bottlenecks, we describe implications for future architectural enhancements, application tuning, and system architectures. After optimizing the Atom server platform, a large share of power and cost go toward processor capability. With optimized Atoms, more servers can fit in a given data center power budget. For a data center with 15MW critical load, Atom-based servers increase capability by 3.2× for Bing.

Categories and Subject Descriptors: C.0 [**Computer Systems Organization**]: General - Systems architectures

General Terms: Measurement, Experimentation, Performance

Additional Key Words and Phrases: energy-efficient datacenters, mobile hardware architectures, web search

ACM Reference Format:
Reddi, V.J., Lee, B.C., Chilimbi, T, Vaid, K. 2011. Mobile Processors for Energy-Efficient Web Search. ACM Trans. Comput. Syst. 9, 4, Article 39 (March 2011), 40 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

This article extends V.J. Reddi, B.C. Lee, T. Chilimbi, and K. Vaid, "Web search using mobile cores: Quantifying and mitigating the price of efficiency," *Proceedings of the 37th International Symposium on Computer Architecture* [Reddi et al. 2010].

Author's addresses: V.J. Reddi, Engineering and Applied Sciences, Harvard University; B.C. Lee, Electrical and Computer Engineering, Duke University; T. Chilimbi, Microsoft Research; K. Vaid, Microsoft Corporation. This work was done, in part, while V.J. Reddi and B.C. Lee were affiliated with Microsoft Research.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

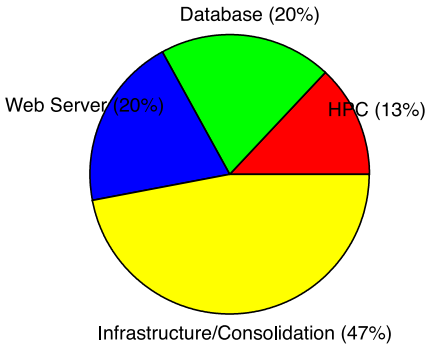


Fig. 1. Workloads constituting the cloud server market are growing in diversity [Eastwood et al. 2009].

1. INTRODUCTION

Internet services are experiencing a paradigm shift in which computation and data migrate from clients to a cloud of distributed resources located in data centers. Today's large-scale data centers already run diverse workloads. Cloud computing services, such as Amazon EC2 and Google App Engine, host a variety of workloads from e-commerce, gaming, enterprise information technology (IT), news, and more. Web search is a first-class Internet service as users turn to search engines for rapid access to the web's wealth of information. Social media workloads, such as Facebook and YouTube, are also on the rise. But compared to search, these workloads are more interactive and dynamic, consisting of environments that are rich with various types of media. Figure 1 shows that Internet services (Infrastructure/Consolidation) comprise a large fraction of the server market. The increasing prominence of Internet services greatly increases the diversity of server hardware and of their uses in datacenters.

Consider data center capability as the amount of work done within a fixed power budget. As demand for computational infrastructure and consolidation increases, the capability of data centers must increase. The demand for capability is driven by data growth enabled by Internet services. This growth is exponential and faster than our growth in ability to store and compute on that data. Over the past six years between 2005 and 2011 data volumes grew at 50 percent per year [Gantz et al. 2008]. Over that same period, storage capacities grew at 40 percent per year. For example, YouTube users upload over 20 hours of video in under a minute [Junee 2009].

Not only must we store big data, we must compute on that data to extract value. Big data domains motivate exascale capability [Kogge et al. 2008]. Just as molecular dynamics for protein folding and fluid dynamics for climate modeling are big data domains, so are web search, data mining, and business analytics. To support a massive number of concurrent requests, Google is estimated to operate over one million servers, which is approximately two percent of the world's servers. Emerging social media Web 2.0 workloads now match the demands of web search. Facebook and Google together account for 14 percent of all US Internet traffic [Heather Dougherty 2010].

Greater data center capability requires greater energy efficiency from commodity computing hardware. With data center power budgets already at mega-Watt scales, cloud computing cannot realize greater capability with larger or more data centers. Rather, data centers should deploy more computing hardware within the same budget. To do so, we must look beyond conventional techniques that manage power but do not provide capability. For example, reducing processor frequency reduces power but also reduces capability proportionally. Because we want both capability and efficiency, we must turn to architectures that complete more work per Joule consumed. Moreover,

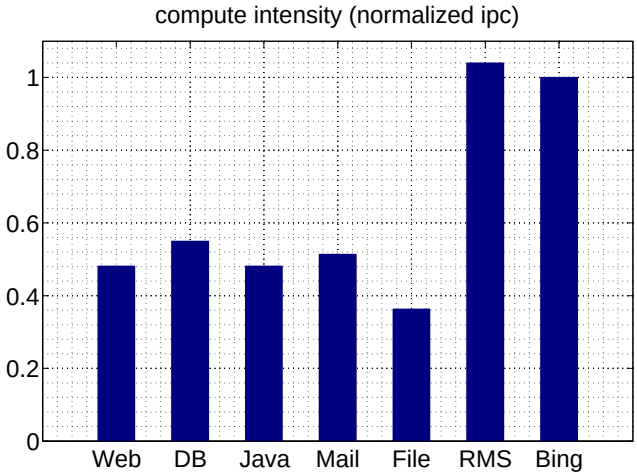


Fig. 2. Computational intensity measured in instructions executed per cycle (IPC). Web search (Bing) is compared against traditional enterprise computing applications as well as recognition, mining, and synthesis (RMS), as benchmarked by the PARSEC suite [Bienia et al. 2008]. IPC is normalized with respect to Bing IPC, which is notably greater than one. All workloads are configured as per industry standards [VMware 2009]. These workloads are run natively on bare metal hardware under typical load conditions on a high-performance server processor (Table I).

we must achieve this efficiency with commodity architectures. By exploiting declining prices and increasing performance in commodity systems, infrastructure providers improve return on investment through horizontal scaling of compute resources.

To improve data center efficiency, we must address processor efficiency, which accounts for the largest share of server power. Small processor cores, in particular, offer dramatic power and energy savings. Traditionally, small cores have been proposed for chip multiprocessors to target throughput-oriented workloads [Barroso et al. 2000; Davis et al. 2005; Kongetira et al. 2005]. Such workloads are characterized by many independent tasks that are constrained by memory, network, or other I/O latencies. In such an environment, computational latency is a second-order effect that might be compromised without much user-perceived impact when using small cores [Lim et al. 2008]. Smaller cores deliver throughput with lower design and power cost when compared to their low-latency, high-performance counterparts.

However, data center applications are in transition. In big data domains, synthesizing information from data requires greater computation. Indeed, Figure 2 considers the Microsoft Bing search engine and its computational intensity as measured by the number of instructions executed per cycle (IPC). By using techniques in statistical machine learning to compute dynamic page ranks, Bing IPC is greater than one and is 2-3$\times$ greater than that of traditional enterprise workloads. Indeed, Bing computational intensity resembles that of recognition, mining, and synthesis (RMS) benchmarks [Bienia et al. 2008]. Bing computational intensity is also greater than measurements previously published for other search engines [Barroso et al. 2003; Barroso 2005].

Although small cores are advantageous for efficiency, they are less capable of handling increases in computational intensity and might jeopardize application quality-of-service and latency constraints. These challenges constitute the price of efficiency exacted by small cores. We quantify this price for the Microsoft Bing search engine and the Intel Atom mobile-class processor. In particular, we perform the following, vertically integrated study:

— **Efficiency (Section 2):** Running the Microsoft Bing search engine, the Intel Atom mobile-class processor is $5\times$ more energy-efficient than the Intel Xeon server-class processor. We quantify efficiency in queries per Joule.

— **Price of Efficiency (Section 3):** Search queries on Atom encounter $3\times$, different page results for 1-3 percent of queries, and diminished robustness for complex queries using advanced search features. However, quality-of-service is maintained for most, common queries.

— **Mitigating the Price of Efficiency (Section 4):** Different computational phases encounter different bottlenecks. We discuss implications for microarchitectural enhancements (e.g., larger caches), application tuning (fewer branches), and system architecture (e.g., heterogeneous multiprocessors).

— **Platform Effects (Section 5):** To realize Atom efficiency, server platforms must be reorganized to reduce power overheads. By integrating multiple Atom cores per chip, we amortize overheads over more cores. By reducing and multiplexing motherboard components, we reduce overheads. After accounting for platform overheads, Atom-based servers can be 1.4-2.1$\times$ more energy efficient than Xeon-based servers.

— **Data Center Effects (Section 6):** With power-efficient Atoms, more servers fit in a given data center power budget. We compute total cost of ownership. Of every dollar spent, more goes to capability and less goes to overhead. For a data center with 15MW critical load, Atom-based servers increase capability by $3.2\times$.

Collectively, the results in this article illustrate a strategy to enhance data center capability through energy-efficient, mobile processors. The microprocessor industry offers a choice of two strategies for efficiency: (1) start with a big, high-performance core and improve efficiency or (2) start with a small, low-power core and improve performance. We compare these two strategies and the data favors the latter for Microsoft Bing.

2. WEB SEARCH EFFICIENCY

Web search is representative of a broad and emerging class of datacenter workloads that extract value from the wealth of data in the web. Given queries, search must identify and return relevant pages to the user [Brin and Page 1998]. Inference engines and machine learning techniques will play an increasingly large role as estimates of relevance become more sophisticated. In this application environment, we compare server- and mobile-class architectures for energy efficiency.

2.1. Web Search Overview

Indexed web pages are distributed across server nodes and each node is responsible for serving queries to its subset of the web. Each indexed page has a static page rank, which quantifies its relevance independent of any query. Upon query arrival, a node computes a dynamic page rank as a function of static rank and query details. Computing these page ranks may require tens of billions of processor cycles and access hundreds of megabytes of data [Barroso et al. 2003].

Figure 3 outlines the structure of the Microsoft Bing search engine. Search queries enter the system through a top-level aggregator. If the aggregator cannot satisfy the query from its cached set of frequently asked queries, it distributes the query to index serving nodes (ISNs). The ISN ranker parses the query and streams through the index to identify a matching list of pages from query features.

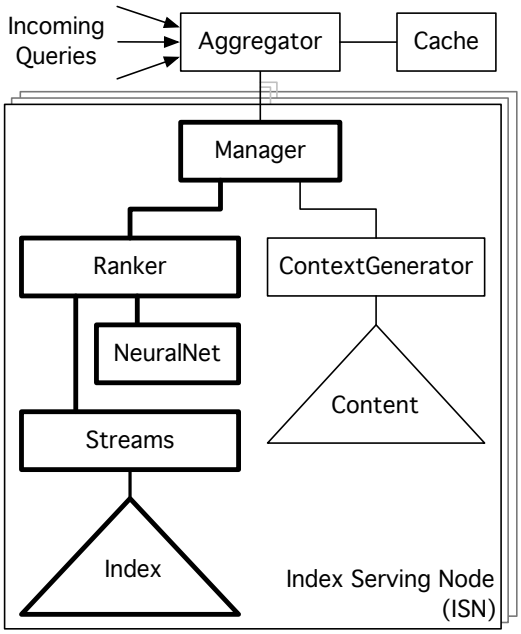


Fig. 3. Overview of Microsoft Bing and the page rank computation within the index serving node.

For these matching pages, the ranker computes the dynamic page ranks. Bing computes this rank using a neural network. Given rank-ordered pages, the ISN returns the top N most relevant pages and their dynamic ranks to the aggregator. Merging results from multiple ISNs, the aggregator identifies and requests metadata for the most relevant results.

If requested, an ISN provides captions, which comprise query results viewed by the user. Each caption includes a title, URL, and snippets of the page for context. These captions are produced by the context generator based on an ISN's subset of pages. Thus, the aggregator identifies the top N results across ISNs and returns the associated captions in response to the query.

Traditionally, the natural data parallelism of web search facilitated data center capability scaling. Such data parallelism often diminished the importance of individual server node performance. However, search engines have evolved beyond the simple page rank algorithms that count link popularity. As web search deploys more sophisticated machine learning techniques to rank pages, server capability becomes more important.

2.2. Web Search Requirements

Microsoft Bing is characterized by broad performance requirements, which fall into three broad categories of robustness, flexibility, and reliability.

Robustness. Search performance is quantified by a combination of quality-of-service, throughput, and latency. Web search defines quality-of-service by the minimum percentage of queries handled successfully. For example, a service target of θ percent requires a minimum of θ successful queries for every 100. The other 100-θ queries might time-out due to long latencies for expensive query features or might be dropped due to fully occupied queues. Given a quality-of-service target constraint, we might consider a platform's sustainable throughput, which quantifies the maximum number of queries

per second that can arrive at a node without causing it to violate θ. If the query arrival rate exceeds sustainable throughput, quality-of-service degrades.

Query processing must also observe latency constraints. The average response time of queries must fall within a certain number of milliseconds, with additional constraints for the 90-th percentile of queries. Latency directly impacts relevance (i.e., documents corresponding to a specific query) by affecting the number of iterative refinements made to a search result. Given a latency constraint, the ranker checks for remaining time before, for example, checking the next tier in a tiered index. Lower query processing latencies allow for additional refinements to improve relevance.

Flexibility and Absolute Load. The search engine operates in a highly distributed system under a variety of loads and activity patterns. Not only must such a system be scalable, it must also be flexible to changes in activity. For example, activity patterns are often periodic and correlated with time of day. Moreover, even a modest spike in complex queries may generate sudden activity spikes measured in absolute terms, as complex queries cannot be broken down into simpler queries for redistribution across multiple nodes. Every ISN must handle its own incoming complex query load. Therefore, the underlying architecture must be robust enough to tolerate these *absolute* spikes and, ideally, would exhibit gradual rather than sharp QoS degradations as load increases. Such architectures would provide greater query flexibility with less disruption.

Reliability and Relative Load. Hardware failures are to be expected within large-scale data centers. To ensure reliability and robustness in the presence of failures, ISNs must operate with spare capacity to bear additional load when a fraction of index serving nodes fail, since work is then dynamically re-balanced across the remaining nodes. Each node experiences a fractional increase of a failed node's sustainable throughput, which is an activity spike measured in *relative* terms. Architectures that exhibit gradual and minimal QoS degradations as load increases in relative terms would provide greater reliability with less disruption.

2.3. Web Search and Mobile Architectures

Web search efficiency depends on the application and its interactions with the underlying hardware architecture. Many traditional enterprise applications run on high-performance, server processor architectures. However, processors for mobile or embedded platforms often require less energy per operation [Dally et al. 2008; Grochowski and Annavaram 2006]. We identify the opportunities and challenges for efficient web search on these more efficient processors. In particular, we compare web search running on Xeons and Atoms, which are Intel architectures targeting server- and mobile-class platforms, respectively.

Consider the spectrum of x86 processors. We observe high-performance, server-class architectures at one end and low-power, mobile-class architectures at the other end. Processor architects have two strategies for energy efficiency. Architects might start with a high-performance design and optimize efficiency by removing features that deliver small performance gain at large power cost. Alternatively, architects might start with a low-power design and optimize by adding features or accelerators that deliver high performance gain at modest power cost. Comparing web search efficiency at both starting points helps us choose between these strategies. Table I summarizes the architectures considered in this study.

Server-class Architecture. The Xeon is a product class built around modern, high-performance processor architectures. We study the Harpertown, which contains eight Penryn cores, organized into two dies with four cores each [George et al. 2007; Intel Corporation 2009b]. Penryn implements several power optimizations. Implemented at 45nm with high-K dielectric and metal gate transistors, the process technology re-

Table I. Server- and mobile-class x86 processor architectures. We evaluate web search on Xeon-Harpertown [George et al. 2007; Intel Corporation 2009b] and Atom-Diamondville [Gerosa et al. 2008; Intel Corporation 2009b] processors, which represent endpoints in the spectrum of x86 commodity processors.

	Xeon	**Atom**
	Harpertown-Penryn	Diamondville
Processors	1	1
Cores	4	2
Process	45nm	45nm
Frequency	2.5GHz	1.6 GHz
Pipeline Depth	14 stages	16 stages
Superscalar Width	4 inst issue	2 inst issue
Execution	out-of-order	in-order
Reorder Buffer	96 entries	n/a
Load/Store Buffer	32/20 entries	n/a
Inst TLB	128-entry, 4-way	Unknown
Data TLB	256-entry, 4-way	Unknown
L1 Inst/Data Cache	32/32KB	32/24KB
L2 Cache (per die)	12MB, 24-way	1MB, 8-way
FSB	1066MHz	533 MHz

duces leakage power by 5-10×. Thus, compared to prior high-end core designs, Penryn consumes much less energy when idle. Process technology also enables more efficient active computation as switching power falls by 30 percent.

Adaptive architectures adjust hardware resources to match application behavior and to reduce power. For example, each Penryn die includes four cores sharing a 12MB L2 cache. The cache is organized into 1MB slices, which allows it to dynamically adapt capacity. Powering down the cache in 1MB increments reduces power.

Dynamic voltage and frequency scaling also manages power. However, such scaling has a modest effect for a platform already optimized for power efficiency, such as the Harpertown L5420. As shown in Figure 4, the L5420 already operates at near minimum voltages and frequencies. Switching between the highest and lowest performance states (i.e., P-states), the architecture implements only a negligible voltage reduction. As we increase processor load by running computationally intensive floating-point loops [Mienik 2000], we observe a 15 percent power savings between the high and low P-state. Since operating voltages are not changing, the majority of the power reduction is due to a proportional reduction in clock frequency.

The choice of a power-optimized Harpertown L5420 with Penryn cores provides a robust baseline for our study. Since our original study, Intel's server-class architecture has evolved from the Penryn to the Nehalem [Kumar and Hinton 2009]. Nehalem further improves energy efficiency, emphasizing features that increase performance by more than one percent for every one percent increase in power. Further improving efficiency, Nehalem includes a dedicated controller that monitors the operating environment to set performance states, voltages, and frequencies. Among these settings is the ability to power gate processor cores, eliminating both dynamic and static power. Many of these optimizations improve energy proportionality when the processor is less than fully utilized. While Nehalem is significantly more power-efficient than Penryn,

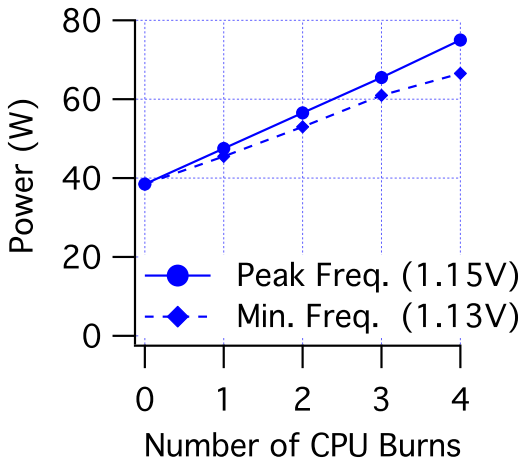


Fig. 4. Effects of dynamic voltage and frequency scaling on the Xeon Harpertown L5420 as the processor is subject to a power virus—CPU Burn [Mienik 2000].

its evolutionary efficiency is insufficient to close the large power gap between server- and mobile-class processors.

Mobile-class Architecture. The Atom is representative of modern, low-power processor architectures. We study the Diamondville, which contains two cores [Gerosa et al. 2008; Intel Corporation 2009b]. Each core is designed to operate in the sub-1W to 2W range. While modern high-performance architectures dynamically re-order instructions to improve instruction-level parallelism, Diamondville cores implement an in-order pipeline with power efficient instruction decode and scheduling algorithms.

The Diamondville datapath is narrower than that of the Harpertown, issuing only two instructions per cycle. These instructions issue to general-purpose logic that provides multiple functionality rather than specialized execution units. For example, the SIMD integer multiplier and floating-point divider are used to execute instructions that would normally execute on separate, dedicated scalar equivalents [Gerosa et al. 2008]. Such strategies may reduce power, but also have implications for performance as SIMD execution units compute on scalar operands.

2.4. Experimental Methodology

Web search distributes queries across several index serving nodes. In this article, we specifically examine its activity within a node. On this node, we install production-quality web search and drive it with queries traced from real user activity. Of the components illustrated in Figure 3, we specifically examine the subset that performs dynamic page ranking and returns the sorted results to the aggregator. We do not account for the context generator.

The quality of web search is defined by the relevance of pages returned in response to a query. Relevance depends on server node performance. Nodes experience heavy computational load. In particular, under typical operating conditions, processor activity ranges between 60 and 70 percent. This intensity motivates our study of leaf nodes in distributed web search computation. We neglect the aggregator and the query trace only considers activity that misses in the cache. Hereafter, “search engine” refers to the dynamic page ranker, which we investigate.

The index serving node computes page ranks for a trace of forty thousand queries after warm-up. Input queries are obtained from production runs. Our experiments pa-

rameterize query arrival rates, measured in queries per second (QPS). Recall from Section 2.2 that search defines quality-of-service as the percentage of successfully processed queries (θ). To determine an architecture's maximum sustainable throughput, we sweep query arrival rates and maximize QPS without violating θ.

The index serving node services queries for a 1GB production index, which is a subset of the global index distributed across several nodes. Given a query, each node computes ranks for pages in its index. Each indexed page is characterized by a static rank, which determines its potential relevance before a query is known. After a query arrives, a dynamic rank is calculated as a function of static rank and query features. The size of a node's local index is determined by memory capacity. The index is designed to reside in memory, minimizing page faults and disk activity. In such a system, processor architecture is a primary determinant of performance.

To ensure a fair comparison, we normalize results by the number of sockets; Harpertown consists of two Penryn processors, each with four cores. Moreover, we often normalize by the number of cores as well; Penryns have four cores whereas Atoms have two. Also for a fair comparison, we match the total number of application threads, which service queries, to the number of hardware thread contexts. For Atom, this means enabling simultaneous multithreading (SMT). System software is configured such that our setup is representative of a typical deployment strategy in Microsoft datacenters.

We analyze microarchitectural performance using measurements collected via VTune [Intel Corporation 2008a], a toolbox that provides an interface to hardware counters on both the Xeon and the Atom. These counters provide detailed insight into microarchitectural activity that can be attributed to specific computational phases in the search engine. To relate microarchitectural activity to energy consumption, we measure power dissipated by the processor. In particular, we identify the 12V lines entering the voltage regulator module. Applying a Hall-effect clamp ammeter (Agilent 34134A) to this line, we collect power measurements at 1KHz using a digital multimeter (Agilent 34411A).

We present much of the data in relative terms to illustrate trends and trade-offs. By normalizing data, we safeguard the absolute performance numbers for the Microsoft Bing search engine.

2.5. Efficiency Analysis

To determine web search efficiency on server- and mobile-class processors, we must compare the rate of work against the rate of energy consumed. In particular, we measure throughput and power, which quantify queries per second and Joules per second. Dividing throughput by power, we obtain queries per Joule.

Throughput. Let X_θ be the maximum number of queries per second sustained by the Xeon without violating the quality-of-service target θ. Similarly, define A_θ for sustainable Atom throughput. The quality-of-service guarantee is robust if target θ is satisfied despite fluctuations and temporary increases in query load. Figure 5 illustrates quality-of-service trends for Xeon and Atom processors. The horizontal axis quantifies query arrival rate, normalized to Xeon's sustainable throughput X_θ. The vertical axis quantifies quality-of-service as the percentage of successfully processed queries.

At the same θ, Xeon sustains 3.9$\times$ the throughput sustained by an Atom (X_θ = 3.9$\times A_\theta$). On a per core basis, each of the four Xeon cores sustain 2.0$\times$ the throughput sustained by each of the two Atom cores. Xeon processes the additional queries more robustly. Degradations in quality-of-service are modest and gradual. Atom is unable to absorb a large number of additional queries and quickly violates its quality-of-service target. We sweep query arrival rates to extreme values (e.g., $2X_\theta$) to identify the maximum number of queries per second that an architecture can sustain. This sweep also

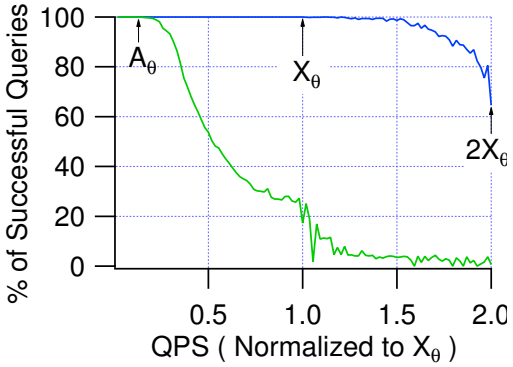


Fig. 5. Xeon and Atom quality-of-service with varying query arrival rates (QPS). Query arrival rates are normalized to X_θ.

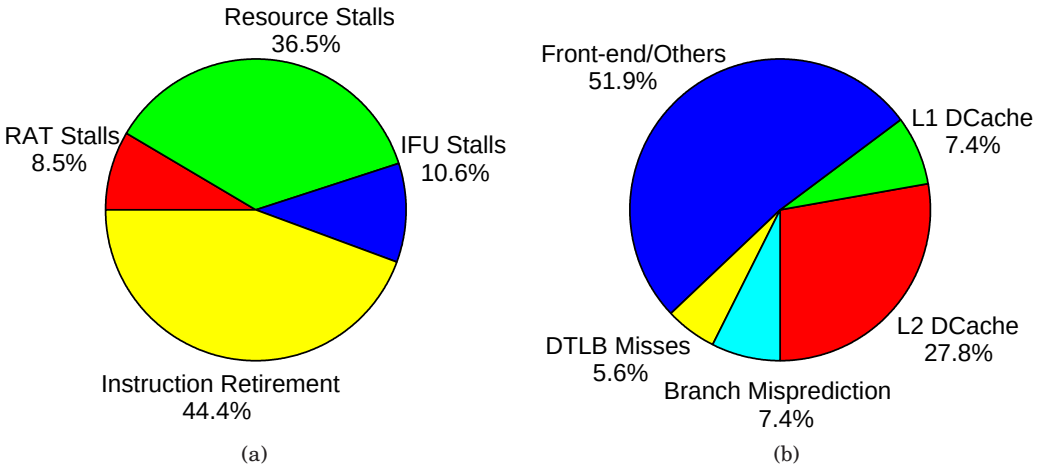


Fig. 6. Search and Xeon (a) execution versus stall time, (b) stall time breakdown.

characterizes robustness and trends. We do not assume sustainable operation under such loads in a production environment.

Xeon Architectural Activity. Figures 6-7 illustrates microarchitectural activity on the Xeon under its maximum sustainable query load X_θ. As shown in Figure 6(a), 55 percent of execution time is spent stalled for either the register alias table (RAT), the instruction fetch front-end (IFU), or data communication (e.g., cache and memory). Structural conflicts or long latency memory instructions block instruction retirement. Collectively, these stalls reduce datapath utilization such that only 44 percent of processor cycles retire instructions. Figure 6(b) further highlights the source of stalls.

Stalls during instruction fetch arise from branches and instruction cache effects. As illustrated in Figure 7(a), substantial branch activity of 16 branches per 100 instructions makes the branch predictor a bottleneck. Moreover, the datapath sees a seven cycle penalty when the number of unresolved and in-flight branches exceeds the capacity of the branch predictor (e.g., branch history table) [Intel Corporation 2008a]. Furthermore, as shown in Figure 7(b), the instruction fetch unit often stalls for L2

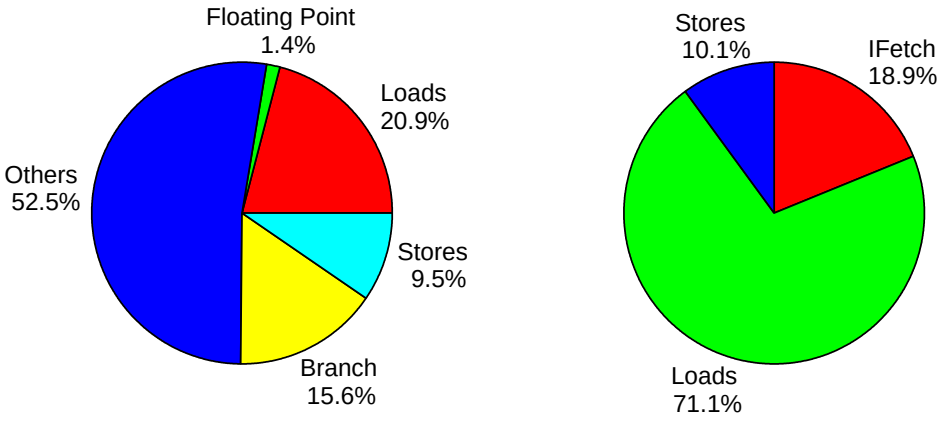


Fig. 7. Search and Xeon (a) instruction mix, (b) source of L2 cache accesses.

cache activity with 19 percent of L2 cache accesses attributed to instruction cache misses.

Other resource stalls may be attributed to memory activity. According to Figure 7(a), a total of 30.4 percent of instructions either load or store data, which leads to pressure on the cache hierarchy. Consequently, Figure 7(b) shows that data loads and stores account for 81.2 percent of all L2 cache activity. Such L2 cache activity often translates into memory requests with 67 percent of bus activity attributed to memory transactions (not shown). This L2 and memory activity arises from the nature of the search engine, which streams through the server node's indexed pages to compute dynamic page ranks.

Atom Architectural Activity. Figure 8 compares Atom microarchitectural activity to that of the Xeon when both architectures compute on their respective maximum sustainable query loads. Relative to the Xeon, the Atom implements a simpler and deeper pipeline. The high frequency of branches and Atom's deeper pipeline depth increases the number of cycles penalized by branches. Moreover, despite its deeper pipeline, the Atom incurs additional penalties as the number of in-flight branches exceeds the maximum allowed.[1] Collectively, these effects lead to a $10\times$ increase in cycles spent penalized by branch activity.

The divider is another architectural bottleneck in the Atom. The number of Atom cycles executing divide instructions increases by $48\times$ relative to Xeon divider activity. These performance effects may arise from a decision to favor generality over specialization for execution units. Specifically, "the use of specialized execution units is minimized. For example, the [single instruction multiple data] integer multiplier and floating point divider are used to execute instructions that would normally require a dedicated scalar integer multiplier and integer divider respectively" [Gerosa et al. 2008]. While this design decision may reduce static power that would be dissipated

[1]We see a large increase in BACLEAR events reported by VTune. According to the user manual [Intel Corporation 2008a], BACLEAR "counts the number of times the front end is re-steered, mainly when the Branch Prediction Unit cannot provide a correct prediction and this is corrected by the Branch Address Calculator at the front end. This can occur if the code has many branches such that they cannot be consumed by the BPU. Each BACLEAR generates approximately an 8 cycle bubble in the instruction fetch pipeline. The effect on total execution time depends on surrounding code."

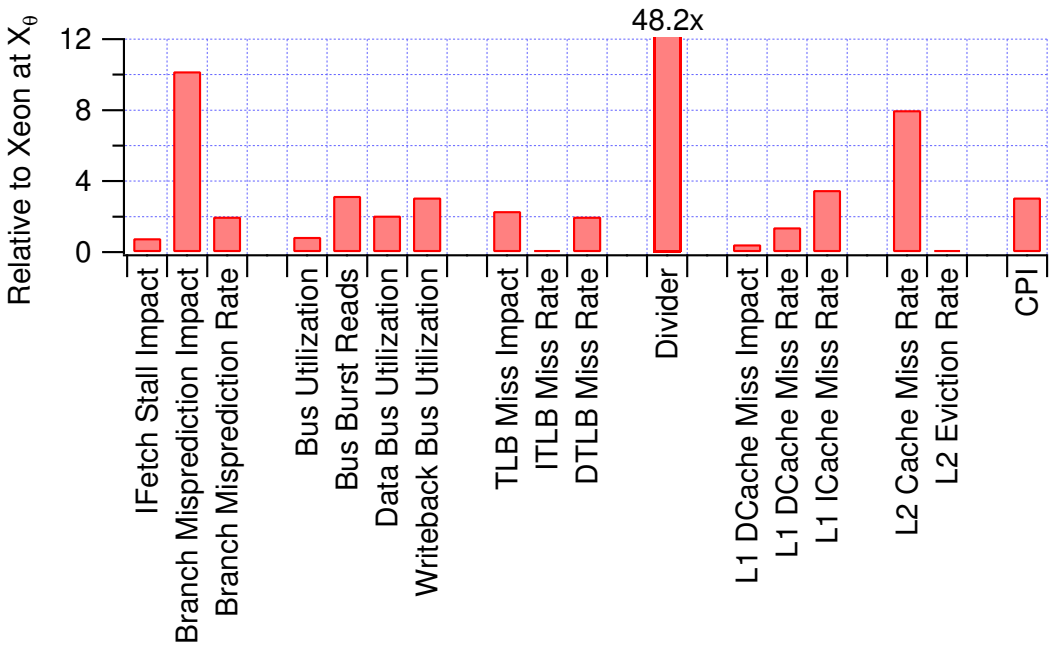


Fig. 8. Atom microarchitectural activity with respect to Xeon.

by infrequently used logic, performance penalties may arise as SIMD units execute on scalar operands.

Per core, the Atom implements a much smaller cache hierarchy. As summarized by Table I, the Atom L1 data and L2 unified caches are 25 and 66 percent smaller than their Xeon counterparts. This smaller cache capacity translates into $1.5\times$ and $8.0\times$ the number of data cache and L2 cache misses.

Collectively, these microarchitectural effects lead to a $3\times$ increase in cycles per instruction. This increase impacts the mean and variance in per query latency. Note the distinction between microarchitectural latency, per query latency, and system query throughput. Switching from the Xeon to the Atom impacts cycles per instruction by $3\times$ but only impacts query throughput by $0.5\times$. The impact on system throughput can be less than the impact on microarchitectural latency because some processing is wasted on failed, timed-out queries. Thus, Section 3.1 will show that microarchitectural measures of per instruction latency are most useful when paired with system measures of per query latency.

Power. Figure 9 illustrates the power time series for both Xeon and Atom as they run web search at their respective maximum query loads. The Xeon operates with an idle power component of 38 W. This substantial idle power is particularly problematic given that the processor is stalled for 56 percent of its cycles. The Xeon exhibits a dynamic power range of 38 to 75 W with a 95 percent difference between idle and peak power.

In contrast, Atom has a low idle power component of 1.4 W with a 168 percent difference between idle and peak power. The Atom realizes a dramatic power reduction; the Xeon dissipates $10\times$ more power. Atom's low idle power and large dynamic range is particularly attractive in the pursuit of energy proportional computing [Barroso and Hölzle 2007]. Moreover, relative to a Xeon core, an Atom core contributes less to query throughput. As Atom cores are idled or powered down, datacenter capability falls at finer granularities, which also favors energy proportionality.

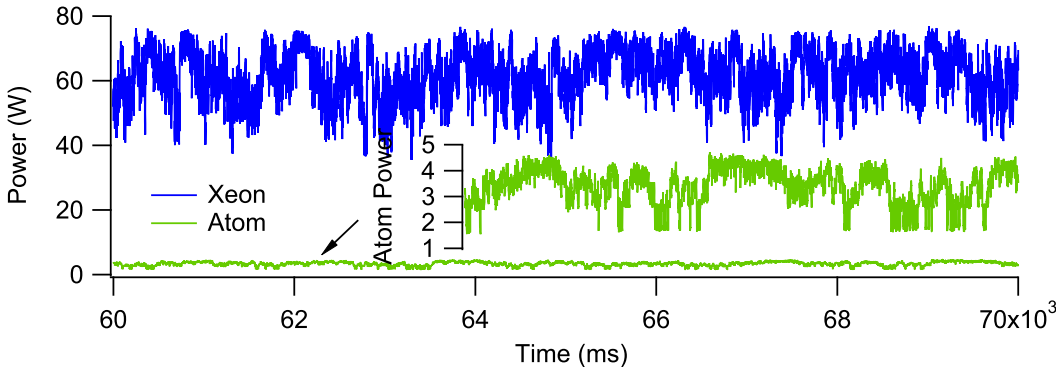


Fig. 9. Xeon consumes ∼62.5W, whereas the low-power Atom consumes ∼3.2W on average.

Table II. Web search on Xeon versus Atom. Performance is sustainable query throughput, measured in queries per second (QPS) at the same quality-of-service target θ. Average power is measured for processors operating at sustainable throughput. Area is cited per processor. Price is reported per unit purchased in orders of one thousand units. QPS is reported normalized to Atom performance A_θ. Measures of efficiency propagate this normalization.

	Per Processor			**Per Core**		
	Xeon	Atom	$\Delta_{X/A}$	Xeon	Atom	$\Delta_{X/A}$
Performance (QPS)	$3.9A_\theta$	A_θ	3.9×	$1.0A_\theta$	$0.5A_\theta$	1.9×
Power (W)	62.5	3.2	19.5×	15.6	1.6	9.8×
Power Efficiency (QPS / W)	$6.2A_\theta$	$31.3A_\theta$	0.2×	$6.2A_\theta$	$31.3A_\theta$	0.2×
Area (T, $\times 10^6$)	820	94	8.7×	n/a	n/a	n/a
Area (mm^2)	214	50	4.3×	n/a	n/a	n/a
Area Efficiency (QPS / mm^2)	$1.8A_\theta$	$2.0A_\theta$	0.9×	n/a	n/a	n/a
Price ($)	380	45	8.4×	95	22.5	4.2×
Price Efficiency (QPS / $)	$1.0A_\theta$	$2.2A_\theta$	0.5×	$1.0A_\theta$	$2.2A_\theta$	0.5×

Efficiency. Throughput is measured in queries per second. Power is measured in Joules per second. Dividing throughput by power, we measure energy efficiency in queries per joule. On a per core basis, the Atom is $5\times$ more efficient than the Xeon. Table II compares the Xeon and the Atom, indicating the large power differential of $20\times$ ($10\times$) per processor (per core) dominates the performance differential of $4\times$ ($2\times$) per processor (per core). Thus, the large power cost of the Xeon is not justified by the relatively modest advantage in sustainable query throughput.

Although this article focuses on energy efficiency, Table II mentions area and price efficiency for comparison. Xeon and Atom area efficiency are comparable, indicating Xeon area overheads from dynamic instruction scheduling, out-of-order execution, and larger caches produce a proportional improvement in search query throughput. Regarding price efficiency, however, a Xeon core is priced more than $4\times$ higher than the price of an Atom core. At this higher price, the Xeon core sustains nearly $2\times$ the query throughput. In effect, every dollar spent on an Atom core leads to $2\times$ the query throughput. Note that we consider the price seen by data center operators and not the cost seen by processor manufacturers. Manufacturers may target higher profit margins on server-class processors. Moreover, this price analysis considers only processor prices, neglecting platform prices and total cost of ownership. Peripheral components,

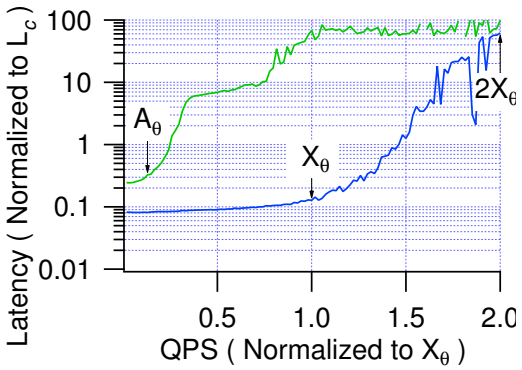


Fig. 10. Xeon and Atom latency with varying query arrival rates (QPS), which are normalized to X_θ.

such as motherboards and memories, will impact the analysis. Section 4 will further assess the sensitivity of these effects.

3. PRICE OF EFFICIENCY

We quantify the price of exploiting energy efficiency with mobile-class processors. As data center workloads evolve toward big data domains that require more analytical computation, understanding the interactions between architecture and application is particularly important. For web search, we find quality-of-service is maintained for most, common queries. However, mobile processor cores are less robust to increases in query load, which impact per query latency as well as latency variance.

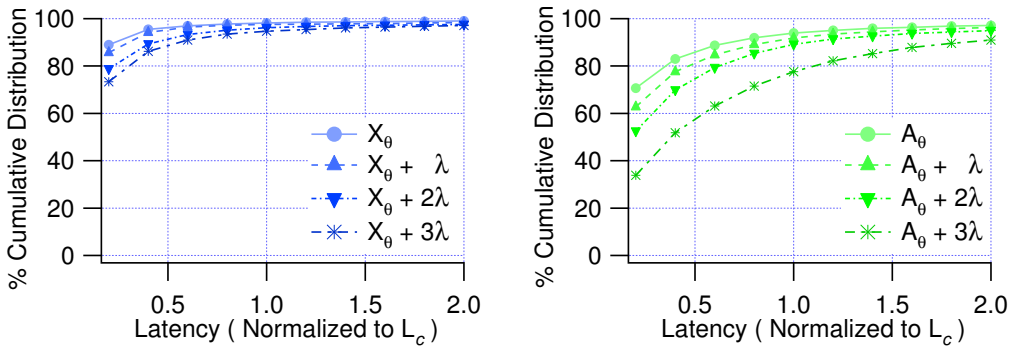
Higher latencies impact the relevance of search results. The search engine allots time spent on any given query. If the allotted time is exhausted, potentially incomplete results computed up to that time are returned. Thus, if the rate of computation slows, relevance may suffer as fewer opportunities exist to refine search results. This impact on relevance is particularly apparent for complex queries.

3.1. Latency

Computational latency is often neglected in online and web services. Often, network latency is assumed to dominate computational latency in the response time perceived by the user. For web search, however, computational latency impacts the substance, in addition to the speed, of results. In particular, each query is subject to a cut-off latency L_C, which defines the allotted time for computation. The search algorithm uses multiple strategies to refine search results as long as query latency has not yet exceeded the cut-off latency.

Figure 10 illustrates average query latency trends as query arrival rate increases. On the logarithmic, vertical axis, latency is normalized to the cut-off. Consider both architectures at their maximum sustainable loads A_θ and X_θ. In this case, query latencies on the Atom are $3\times$ greater than those on the Xeon ($0.33L_C$ at A_θ versus $0.12L_C$ at X_θ). These latencies appear fundamental to the architecture as we observed a similar $3\times$ increase in cycles per instruction (Figure 8). Moreover, these latencies appear to arise from architecture and not system organization (e.g., queuing delays) since the latency gap persists even as query load falls toward the minimum values of Figure 10.

Latency Distributions. In addition to characterizing average latency, we also characterize latency variance. Figure 11 illustrates the cumulative distribution function for latency, which is normalized to cut-off. The experiment allows queries to exceed L_C and tracks the time required to process each query to completion. Moreover, we track



(a) Xeon (b) Atom

Fig. 11. Per query latency distribution. Allow queries to exceed cut-off latency L_C and track per query latencies.

the impact of increased query activity, measured in multiples of λ. This analysis allows us to compare the robustness of architectures to activity spikes.

The Xeon, processing queries at X_θ, satisfies 89 percent of queries in less than $0.2L_C$. 98 percent of queries are satisfied before the cut-off latency L_C and less than 1 percent of queries require more than $2L_C$. Moreover, on the Xeon, these trends are modestly sensitive to increased query loads. For an increased load of X_θ+3λ, 83 and 96 percent of queries are still satisfied in less than $0.2L_C$ and L_C, respectively.

Processing queries at A_θ, search on Atom exhibits greater latencies and variance. Although Atom satisfies 93 percent of its queries before cut-off, we find much greater variance in the latency distribution. Only 68 percent of its queries are satisfied in less than $0.2L_C$. And nearly 3 percent of its queries require more than $2L_C$. Thus, compared to Xeon latency distributions, Atom latency distributions are characterized by a much larger spread between the minimum and maximum latencies. Furthermore, Atom latency distributions are highly sensitive to activity increases. For an increased load of A_θ + 3λ, only 33 and 78 percent of queries are satisfied in less than $0.2L_C$ and L_C, respectively.

Latency and Architecture. Latency effects appear to arise from differences in the Xeon and Atom processor architecture; they persist even under minimum query loads. Figures 12-13 activity to provide deeper insight. As query load increases, measured activity is normalized to values observed at each architecture's respective sustainable loads. Increases in query load minimally impact Xeon microarchitectural activity. Aside from increased memory bus utilization, we find little noticeable change in activity. The net effect on the number of cycles per instruction is modest.

In contrast, Atom microarchitectural activity increases dramatically with additional query load. Increases of λ, 2λ and 3λ queries per second beyond A_θ stress architectural resources and increase the number of cycles per instruction by 7.5, 13.2, and 16.5 percent, respectively. This performance degradation arises primarily from increasing contention in the cache hierarchy. The small 1 MB, 8-way L2 cache becomes a constraint. The L2 cache miss rate increases by up to 22 percent and the L2 cache eviction rate increases by up to 100 percent. The increased eviction rate results in much higher bus utilization; writebacks increase linearly with additional query load. As the memory subsystem becomes a bottleneck, the pipeline is more often stalled waiting for data. Thus, the compute-to-memory intensity is impacted and divider utilization falls by 8 percent with an extra load of 3λ queries per second.

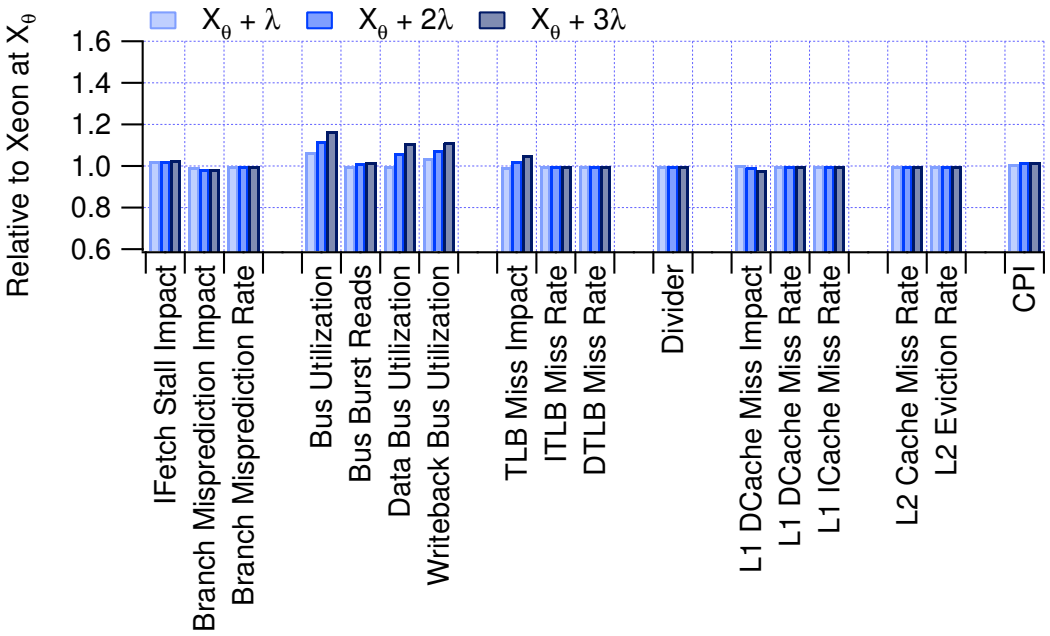


Fig. 12. Xeon microarchitectural activity as load increases beyond sustainable throughput X_θ.

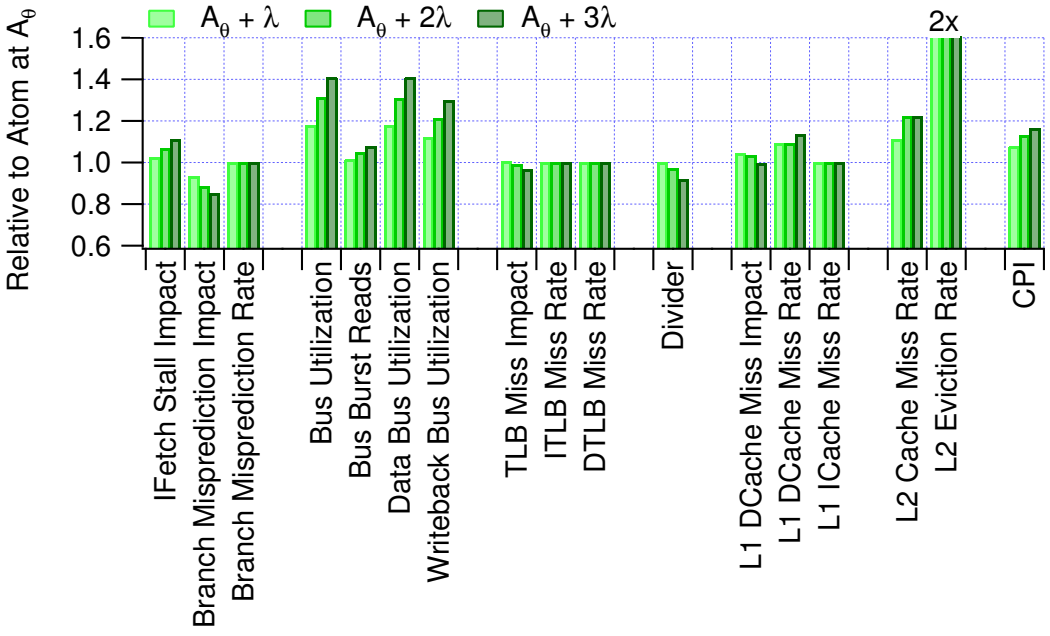


Fig. 13. Atom microarchitectural activity as load increases beyond sustainable throughput A_θ.

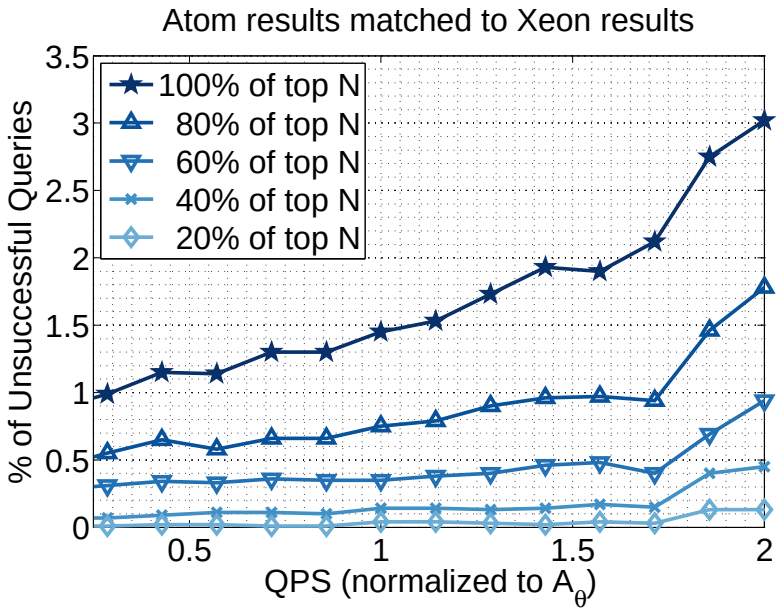


Fig. 14. Bottlenecks on Atom limit the quality of page hits.

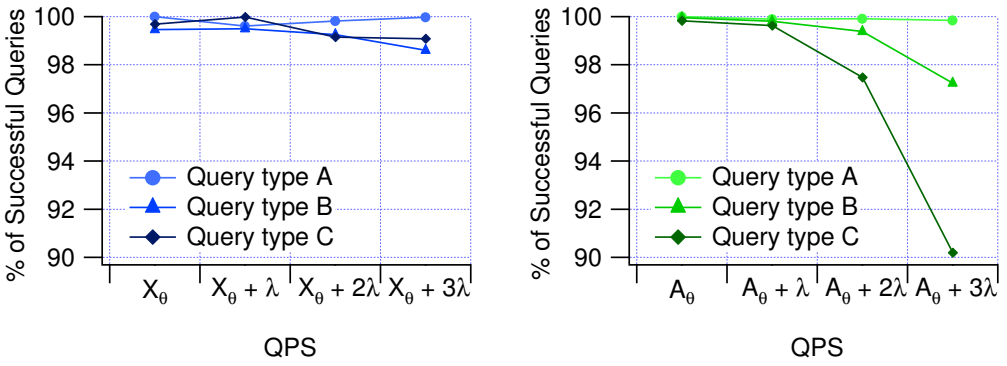
3.2. Relevance

Although network latency dominates user perceived response times, computational latency impacts the relevance of computed results. Search algorithms may have multiple strategies for refining results and the load or cut-off latency L_C would determine opportunities for refinement. In a tiered index, pages indexed in the first tier are always ranked but indices in subsequent tiers may only be ranked if L_C has not been exceeded. Moreover, the fraction of pages that are ranked might be dynamically tuned to trade-off computation energy and user perceived relevance [Baek and Chilimbi 2010]. In these scenarios, higher rates of computation allow for multiple iterative refinements of search results to improve relevance. Lower latencies also provide timing slack, which algorithm designers can consume to improve search heuristics.

Given its higher latencies, search on Atom has fewer opportunities to refine results. Figure 14 illustrates the effect. For each query, we take the top N page results returned by the Xeon as the golden, most relevant results. We repeat each query on the Atom and compare the top N page results against those from the Xeon. In the best case, 100 percent of these N pages match. In the worst case, none of these pages match. We perform this analysis for each of our forty thousand traced queries and determine how many of these queries fail to meet matching criteria and return different results on the Atom. These effects are then examined at varying query loads.

This measure of relevance is conservative since it assumes Xeon results are most relevant and any result mismatch degrades user perceived relevance. In practice, any change in the search engine algorithm or architecture would require assembling a user study group. In some cases, mismatched results may not be noticed by users. However, in the absence of a user study, we apply matching criteria to quantify the impact on relevance.

At maximum sustainable Atom load A_θ, nearly all queries satisfy relaxed matching constraints. In particular, search on Atom easily matches 20 percent of the top N



(a) Xeon (b) Atom

Fig. 15. Quality-of-service by query complexity. Query types A, B, and C are characterized by increasing complexity.

queries. However, more stringent constraints expose differences. At A_θ, 1.4 percent of queries fail to match all N pages returned by the Xeon. The number of such differences increases to 3.0 percent as load increases.

Some of these differences in page results arise from architectural effects. Even at a low query load, results on the Atom differ for 1 percent of all queries. At such low query loads, software queuing delays are negligible and architectural latencies likely account for the difference. Since Atom's per query latencies are $3\times$ those of the Xeon, page rank refining algorithms have a third of the time to complete their work. Thus, we observe different results, even if hardware resources are under-subscribed.

3.3. Complexity

A search engine may experience user-generated activity spikes. The underlying architecture must be capable of adapting to rapid and significant shifts in search activity. These shifts are particularly pronounced for complex queries. Search features (e.g., query length, language specification, and logical expression) determine query complexity and its computational demands. Complex queries arrive at the aggregator and are distributed to index serving nodes. After arriving at a particular node, the complex query is broken into multiple simpler queries. The search engine does not allow these simpler queries to be re-distributed; each index serving node must handle the additional load associated with query complexity.

Even a modest spike in complex queries causes a significant activity spike with an index serving node. Nodes sensitive to activity spikes must be over-provisioned by operating below sustainable throughput by a safety margin. To understand this sensitivity, we evaluate how query latency distribution changes as load increases in the presence of different types of queries. From the trace of forty thousand mixed queries, we separate queries according to complexity and label them into three groups in order of increasing complexity: A, B, and C as illustrated in Figure 15. Query type A has no complex search criteria and is therefore fast to process. Types B and C increase complexity, characterized by longer queries and advanced search engine features.

Figure 15 compares and contrasts the Xeon and the Atom. Atom can match Xeon's quality-of-service for queries of type A, even as query load increases to A_θ+3λ. Thus, Atom provides energy-efficient web search for the simplest queries, which are also the common case. However, the Atom becomes uncompetitive for query types B and C as load increases beyond A_θ. At A_θ+3λ, the percentage of successful queries is only 90

percent for a stream of type C queries. These queries increase the effective query load by splitting complex queries into simpler ones. With this extra load, queues begin to fill and new, incoming queries are dropped until this extra load is processed or until the query exceeds cut-off latencies. In contrast, the Xeon does not suffer from this problem. As query complexity varies, Xeon is able to absorb activity spikes more smoothly.

4. MITIGATING THE PRICE OF EFFICIENCY

For simple and common queries, the Atom offers dramatic energy efficiencies without compromising quality-of-service. For more complex queries, architectural bottlenecks and latency may impact the relevance of results. These effects might be mitigated through architectural enhancements, system organizations, or application tuning.

4.1. Architectural Enhancements

Compared to the Xeon, the Atom can sustain fewer queries per second. System organization can mitigate this disadvantage by over-provisioning processors and reducing query load on index serving nodes. However, the Atom architecture also limits web search performance. As discussed in Section 3, even at low query loads, the Atom exhibits longer per query latency, returns different page results, and handles complex queries less effectively. Thus, system strategies to manage query load may be insufficient. Enhancements to the core architecture may be needed.

Bottlenecks are limited to a few functions within each phase of search computation. Assembling a list of the top twenty functions ranked by share of execution time for Xeon and comparing against a similar list for Atom, we find significant overlap. An important function is important regardless of the architecture. From this list of twenty functions, we identify a representative function in each major computational phase (i.e., manager, neural network, ranker, and streams). Profiling Atom architectural activity for these representative functions, Figure 16 indicates the diversity of microarchitectural bottlenecks across computational phases. No single architectural bottleneck accounts for all of Atom's latency gap. Each function exercises different parts of the microarchitecture.

— The manager coordinates the movement of index files to and from memory. The smaller L2 cache limits opportunities to exploit locality and produces a 14× increase in misses. The smaller cache also increases memory subsystem activity by 20 to 22×, thus causing a 4.6× increase in cycles per instruction.

— The neural network stresses the divider and L2 cache. This function exhibits a 64× increase in division time, which seems to arise from a design decision regarding SIMD versus scalar dividers; scalar division is performed in a SIMD execution unit [Gerosa et al. 2008]. Atom's small 1MB, 8-way L2 cache leads to a 14× increase in L2 cache misses. The effect on neural network computation is a 4.8× increase in cycles per instruction.

— In contrast, other parts of the ranker stress the branch predictor and L1 data cache. The performance impact of branch misprediction increases by 142× from a very low baseline on the Xeon. Such penalties may arise from the ranker's algorithmic components, which apply different strategies depending on query features and cut-off latencies. Also from a low Xeon baseline, the impact of L1 data cache misses increases by 79×. These two effects contribute to a 2.9× increase in cycles per instruction.

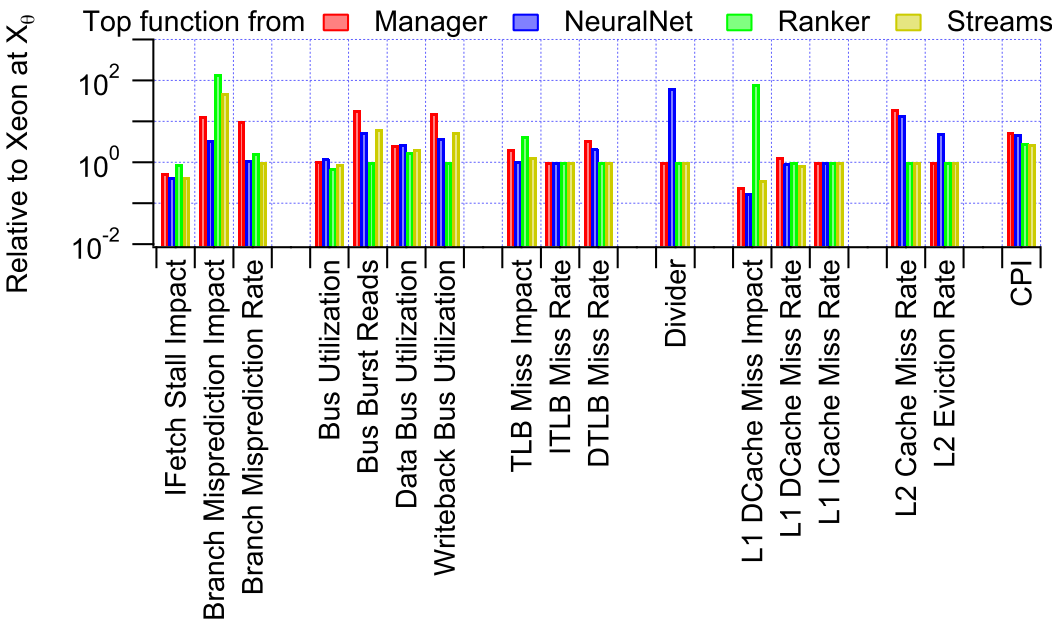


Fig. 16. Identifying bottlenecks across phases of search using representative functions.

— Also stressing the branch predictor, streams manipulate an iterator data structure containing indices that match words within the query. Finding a particular element within the iterator requires non-trivial control flow, which exercises the branch predictor with a 49× penalty relative to Xeon. The number of cycles per instruction increases by 2.8×.

Although we observe performance degradations between 2.8× and 4.8× in representative functions across the four major phases of computation in search, the microarchitectural bottlenecks differ significantly.

Despite the near-term shortcomings of the Atom, the ideal efficient microprocessor seems closer to the mobile-class end of the microarchitectural spectrum. For example, as indicated in Figure 12, Xeon datapath resources (e.g., functional units) are overprovisioned since activity spikes do not have any noticeable effect on most hardware structures. In adopting simpler Atom cores, area falls by 2× as the datapath and caches shrink. Some of these area savings could be re-directed towards larger caches. Such an approach would lead to an unconventional small core designs where high-performance cache hierarchies are paired with an in-order, narrow-issue datapath. Such a strategy would address specific limitations and may be more efficient than architecting a datapath for speculative, out-of-order instruction execution.

4.2. Application Tuning

This article consider a particular implementation of Microsoft Bing. All experiments are performed on compiled binaries and source code analysis is beyond the scope of this article. However, application tuning may circumvent Atom's architectural limitations. To lay the groundwork for such tuning, we consider the computational phases of the search engine and their contribution to overall performance.

Computational Phases. Figure 17(a) illustrates the distribution of execution time across four phases. While ranker and neural network computation is of particular in-

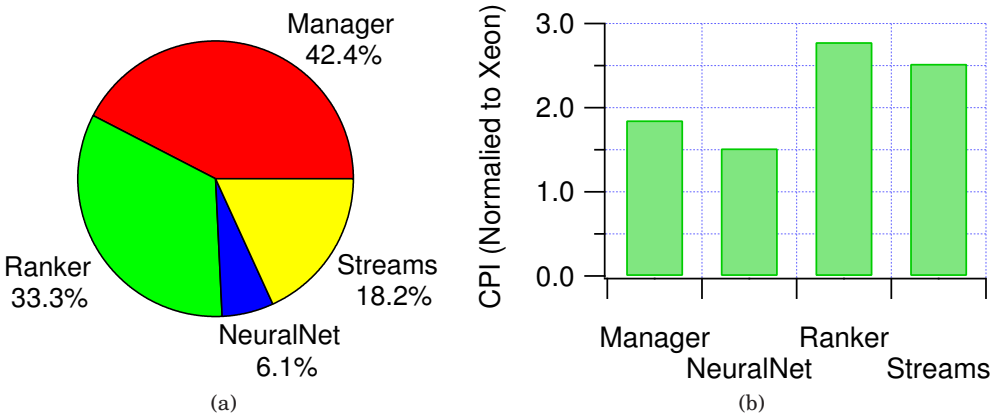


Fig. 17. Computational phases and (a) Atom execution time breakdown, (b) Atom performance penalties measured in cycles per instruction relative to Xeon.

terest, they account for less than 40 percent of the computation. Manager and stream computation must manage the flow of data to feed this ranking computation. To close a latency gap of $3\times$, we need to improve the performance of all computational phases.

In particular, Figure 17(b) indicates the performance degradation across all major phases of computation on the Atom. Atom latency, measured in cycles per instruction, for these phases increases between $1.5\times$ and $2.8\times$ relative to Xeon latency. Moreover, given the difference in clock frequency, each Atom cycle is 50 percent longer. This analysis highlights the challenges when no single function or phase of computation can be targeted to close the performance gap.

Hot Functions. Because each phase comprises a large fraction of total execution time and all phases see a performance penalty, the importance of each computational phase is similar for both the Xeon and the Atom. However, if we consider the importance of individual functions, we observe more significant shifts. The different hardware requirements of individual functions cause some functions to become more important as they encounter specific Atom bottlenecks.

Figure 18 illustrates this effect for the top twenty functions, when rank ordered by share of total execution time. A list of the top twenty is compiled for search subroutines. Although we cannot present specific function names, we identify their association to broader computational phases (e.g., Manager). Each function in the two lists are cross-referenced, allowing us to compare and contrast their relative importance in search on the Xeon and the Atom.

Examining the gradient of arrows linking the two lists, we find neural network functions exhibit the largest increase in share of execution time. On Xeon, neural networks occupy the 10th and 27th position in a list of twenty. These same functions occupy the 3rd and 7th position on the Atom. This shift illustrates Atom's architectural impact on neural network computation. The neural network helps compute dynamic page ranks and is on the critical path for query processing. When switching to the Atom, five new functions enter the top twenty: one NeuralNet, one Stream, two Manager, and one Ranker function.

This analysis highlights the need for future work in application tuning as the search engine deploys on a new architecture. As new functions become hot, the software tuning strategies must be aware of underlying hardware resources. At the function level,

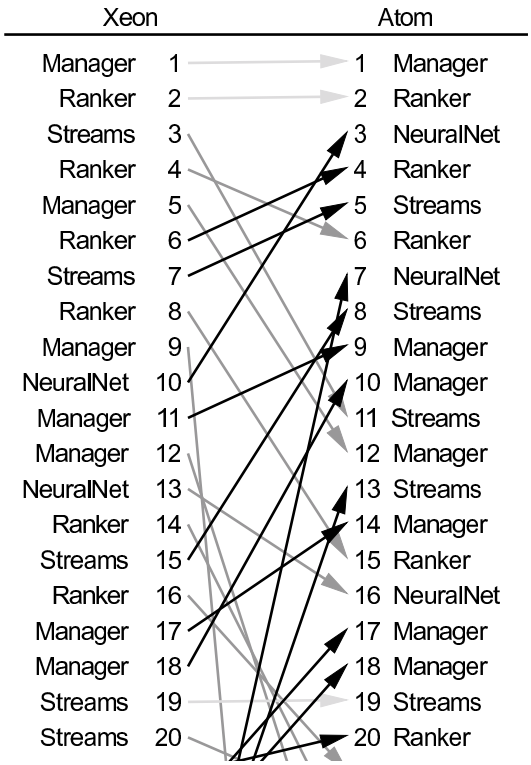


Fig. 18. Hot functions, labeled by computational phase, on Xeon and Atom. Arrows cross-reference functions across two architectures.

for example, re-writing code to reduce the degree of conditional control flow may relieve pressure on the branch predictor.

At the application level, web search may dynamically adjust the number of pages that must be ranked before returning answers. In many cases, a server node might not need to rank all pages in its index since its results are aggregated with other serving nodes' results. In this aggregation, the set of top N pages from a particular node may not propagate into the set of top N pages after aggregation across all nodes. Exploiting this fact, algorithm designers might develop heuristics to determine what fraction of the index requires ranking [Baek and Chilimbi 2010].

A coordinated hardware-software design strategy is needed to identify Atom limitations that should be addressed by architectural enhancements in silicon and those that should be addressed by application re-writes in software. In practice, however, business realities and separation of interests pose challenges to coordination. In particular, data center application developers are accountable for the performance of their deployed service but do not observe costs associated with procuring and running hardware. As a result, there is little incentive to jeopardize application performance to accommodate hardware capabilities. Moreover, application quality impacts end-user experience and market share. Thus, further work is needed to span the hardware-software divide.

4.3. Heterogeneous Chip Multiprocessors

Looking further beyond conventional architectures, hardware heterogeneity can improve efficiency and performance by tailoring accelerators for computational bottle-

necks. Prior research in heterogeneous chip multiprocessors proposes integrating large and small cores onto a single chip (Section 7). However, consider the divisions between computational phases (Manager, NeuralNet, Ranker, Streams) and the assignment of these phases to large and small cores. A preliminary analysis suggests such a solution may be energy inefficient for Microsoft Bing. In particular, we optimistically estimate performance gains and conservatively estimate power costs. If heterogeneity is not energy-efficient in this analysis, it will not be efficient in a real implementation.

Consider the four phases of computation: Manager, NeuralNet, Ranker, Streams. We quantify potential performance gains from accelerating this computation using a Xeon core instead of an Atom core. If we accelerate the most important function in each phase, Figure 16 indicates benefits between 2.8× and 4.8×. If we accelerate the broader phases of computation, Figure 17(b) indicates benefits between 1.5× and 2.8×. Grouping functions into phases, instead of targeting a single top function, dilutes performance gains. In both cases, these performance gains are optimistic because they do not account for costs of carving out computation to execute on a different processor core.

We also quantify potential power costs from accelerating computation on a large Xeon core. Figure 9 indicates a two-core Atom dissipates 4.3W (2.2W per core) at its measured peak while a four-core Xeon dissipates 38.5W (9.6W per core) at its measured trough. Thus, the minimum power cost of assigning computation on the Xeon instead of the Atom is 4.3×. By comparing Atom peaks to Xeon troughs, this power cost is optimistic because they do not account for Xeon core activity, which would further increase power cost.

Even under optimistic performance benefits and conservative power costs, costs are clearly larger than the benefits. This data highlights the difficult of efficient acceleration with general-purpose processor cores of varying sizes. While it may be possible to exploit heterogeneity at a granularity finer than computational phases, doing so requires significant re-engineering of the search engine, which is a complicated and costly task. It may also be possible to exploit heterogeneity at a coarser, directing complex queries to Xeon cores while directing simpler queries to Atom cores. This would ameliorate the quality-of-service penalties associated with Atoms by using Xeons. Even in this case, however, Joules per query would increase.,

In the future, more efficient acceleration may arise from application-specific accelerators targeting computational kernels (e.g., neural network). Such accelerators may incur low power costs, while recovering performance lost by low-power, mobile-class architectures.

5. PLATFORM ARCHITECTURE AND RELIABILITY

Although the Atom processor core is energy efficient, its effect on data center efficiency depends on platform organization. Organization, in turn, often interacts with reliability constraints. We consider the impact of node failures as query load on the failed node shifts to remaining nodes. To maintain quality-of-service, node over-provisioning may be needed. The cost of over-provisioning depends on platform power.

5.1. Reliability

Hardware- or software-based failures are often threats in a data center. Therefore, we must understand how Atom processors perform when query load is re-balanced to accommodate node failures. Such load re-distribution leads to fractional or relative increases in query load for a given processor. For instance, a processor will experience a relative load of $1.2A_\theta$ when a node fails within a group of five nodes. Given the same query load, Atoms may provide reliability more easily or gracefully than Xeons. If the

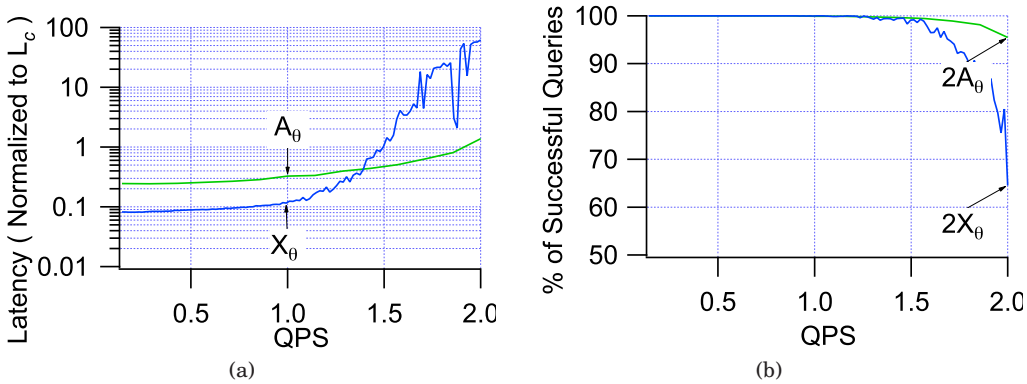


Fig. 19. Comparing quality-of-service and latency for fractional increases in query load. Horizontal axis quantifies queries per second normalized to Atom and Xeon's respective maximum query throughputs, A_θ and X_θ.

same absolute load is distributed across many more Atom-based nodes than Xeon-based nodes, the increase in relative query load is smaller when a node fails.

Figure 19 compares the effect of fractional load increases on quality-of-service and average query latency. The horizontal axis quantifies queries per second normalized to Atom and Xeon's respective maximum sustainable query throughputs, A_θ and X_θ. At loads beyond sustainable throughput (1.0 on the normalized horizontal axis), Xeon query latency rises more quickly than Atom query latency. At query loads of $1.5X_\theta$, Xeon query latencies exceed L_C. These Xeon queries are at risk of time-outs in software queues. The aggregator assumes the index serving node cannot produce the required pages. In contrast, Atom query latencies of $0.5L_C$ are still tolerable at a query load of $1.5A_\theta$.

Higher latencies produce a corresponding drop in quality-of-service as more queries time-out in software queues. Atom quality-of-service degrades to 95 percent at $2A_\theta$whereas Xeon quality-of-service degrades to 64 percent at $2X_\theta$. Atom quality-of-service degrades more gradually because the same fractional increase in load (e.g., $1.2\times$) on Atom and Xeon corresponds to a smaller increase in absolute load on Atom; A_θ is smaller than X_θ.

Figure 20 shows microarchitectural activity for both the Xeon and the Atom as load increases by 1.2 to $2.0\times$ their respective sustainable query throughputs. The Xeon is unable to scale to higher loads because architectural resources are saturated at query loads beyond $1.5X_\theta$. Branch misprediction rates, memory bus utilization, TLB activity, cache activity reach maximum levels. At 50 percent additional load, the number of cycles per instruction increases by 11 percent.

In contrast, Figure 21 indicates the Atom architecture has more room for additional fail-over load. Activity increases and we do not observe plateaus. Thus, while the Xeon is capable of sustaining higher throughput, it must run significantly under this peak to anticipate and to handle fail-overs. By adding this safety margin, Xeon query throughput falls. Moreover, energy efficiency falls as high Xeon power is amortized over fewer queries per second.

The costs of over-provisioning depend on the costs of the processors and their platforms. To understand these effects, we need to account for other platform component and their costs, both in dollars and power. To provide a holistic analysis, platform costs should be integrated to quantify total cost of ownership (TCO). In addition to capital costs associated with computing hardware, TCO accounts for facility capital costs as

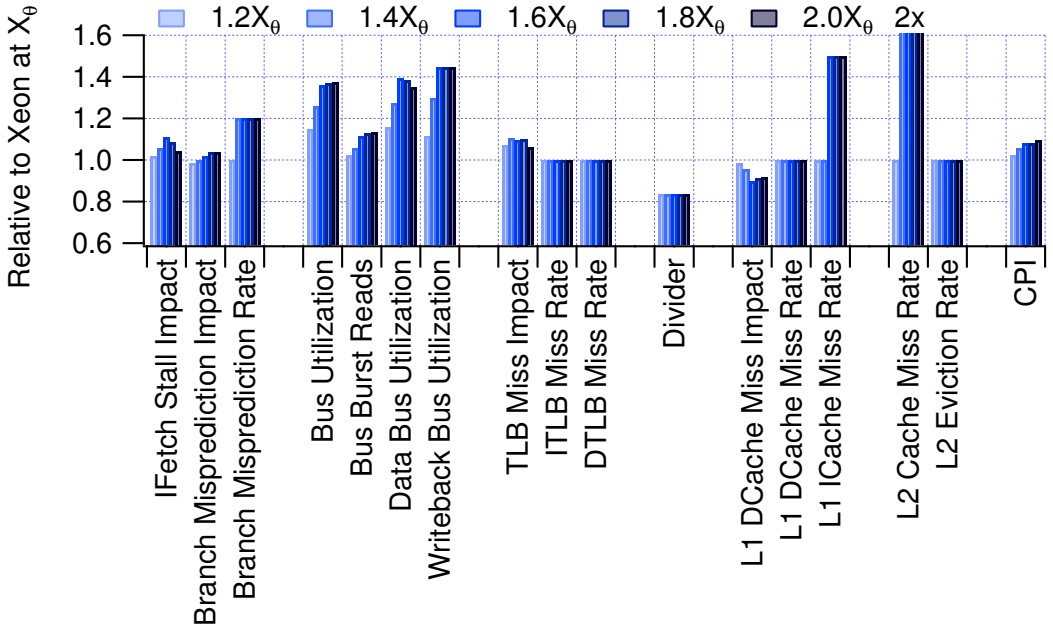


Fig. 20. Xeon microarchitectural activity during load re-distribution and fail-overs.

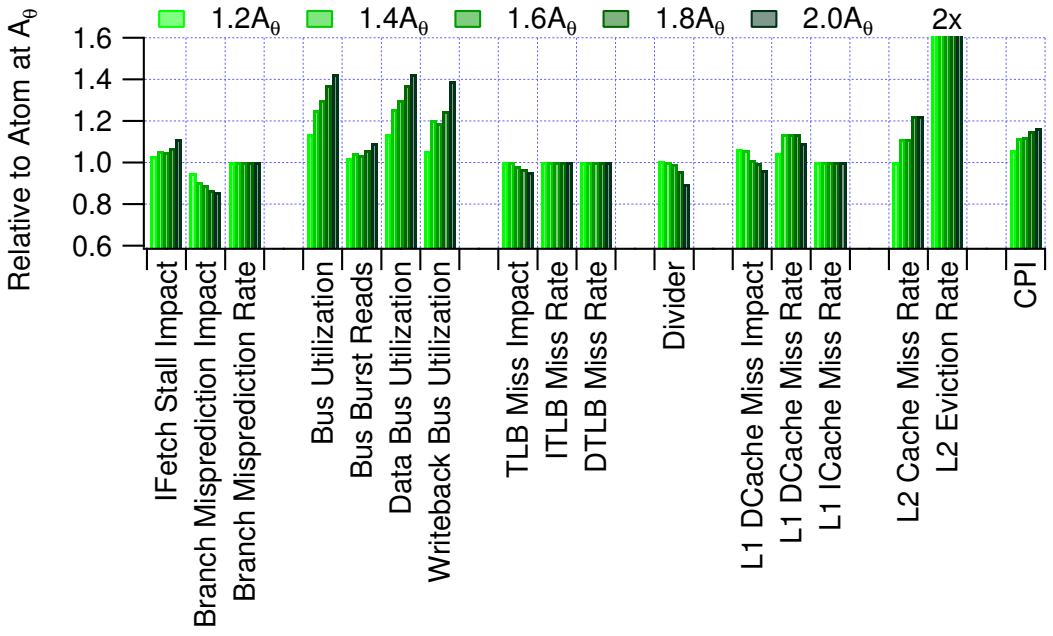


Fig. 21. Atom microarchitectural activity during load re-distribution and fail-overs.

Table III. Platform cost and power for Xeon and Atom baselines. Efficiency normalized to Xeon Harpertown baseline. See Appendix A for sources and assumptions.

	Xeon Harpertown 4-core, 2-socket		Atom Diamondville 2-core, 1-socket	
	Cost ($)	Power (W)	Cost ($)	Power (W)
Processor	760	125	45	3.2
Motherboard	200	30	80	30
Network Interface	0	5	0	5
Memory (16GB)	300	20	300	20
Storage (HDD)	100	10	100	10
Total Server Cost	1360	190	525	68.2
Normalized Efficiency	1.00 QPS/$	1.00 QPS/W	0.22 QPS/$	0.23 QPS/W

well as operating costs. Performing this analysis for both Xeons and Atoms will illustrate the sensitivity of Atom efficiency to platform-level costs.

5.2. Platform Cost and Power

Table III summarizes the cost and power of various platform components. See Appendix A for sources and assumptions. Although the Atom processor dissipates between 1.5 to 4.5W when running web search, peripheral and other platform components also contribute power to the total. Of particular concern is the commodity motherboard, which dissipates 30W. In the Xeon Harpertown, motherboard power is modest in comparison to that of the processor. However, as processor power falls, motherboard overheads become much more significant in the Atom Diamondville.

The fraction of cost and power attributed to processors is important. Processors contribute to data center capability and search engine throughput. In the Xeon Harpertown, processor cost and power accounts for 56 and 65 percent of the total. However, the processors shrink dramatically in the Atom Diamondville and account for only 8 and 4 percent of platform cost and power. Worse, as discussed in Section 2, every Atom processor core contributes only 0.5$\times$ the query throughput of a Xeon processor core. Thus, for every dollar spent or Watt dissipated, the Atom processor delivers fewer queries per second than the Xeon processor.

To address these challenges, Table IV considers two alternative platforms built from Atom processors.

— *Integrated Atom*: Consider a platform with greater multi-core integration. Determine the number of Atom cores that fit in the area of the Xeon chip. As noted in Table II, an Atom core is approximately half the size of a Xeon core. Thus, we consider a 200 mm^2 chip multiprocessor with eight Atom cores. Moreover, we consider a two-processor motherboard to produce a server with sixteen Atom cores. This strategy reflects a trend toward greater integration for x86 cores [Seiler et al. 2008].

— *Optimized Atom*: Consider 50 mm^2 chip multiprocessors with two Atom cores each. Deploy an eight-processor motherboard to produce a server with sixteen Atom cores. Furthermore optimize the motherboard and peripherals. Except for processors and memory, eliminate all components from the motherboard. These components are then consolidated and multiplexed across a large number of processors, thereby eliminat-

Table IV. Platform cost and power for Atom platform alternatives. Efficiency normalized to Xeon Harpertown baseline of Table III. See Appendix A for sources and assumptions.

	Integrated Atom 8-core, 2-socket		Optimized Atom 2-core, 8-socket	
	Cost ($)	Power (W)	Cost ($)	Power (W)
Processor	760	25.6	360	25.6
Motherboard	200	30	1350	3
Network Interface	0	5	0	0.5
Memory (4GB)	300	20	400	20
Storage (HDD)	100	10	100	10
Total Server Cost	1360	90.6	2110	59.1
Normalized Efficiency	0.67 QPS/$	1.40 QPS/W	0.43 QPS/$	2.14 QPS/W

ing a large fraction of motherboard overheads. SeaMicro demonstrates this approach to eliminate 90 percent of motherboard components [Rao 2010].

These two strategies reflect two perspectives on deploying mobile processors in datacenters. The first strategy assumes the cooperation of processor architects. The second strategy assumes fixed processor designs and organizes a system around the processors.

Table IV illustrates the advantages of platform engineering. Chip multiprocessor integration increases processor cost and power as a share of server totals. With a total of sixteen Atom cores, the Integrated Atom platform amortizes motherboard overheads over a larger number of cores, which means processor power comprises 28 percent of the total and more of each Watt dissipated contributes to query throughput. Relative to the Xeon baseline, energy efficiency improves by 1.4×. Additional processor cores increase throughput more than they increase power. The power of other platform components dilutes the power impact of additional cores. However, larger chips and more processors per motherboard will increase cost. Query throughput per dollar falls by 0.7×.

The Optimized Atom in Table IV illustrates the effects of customized motherboards. Increasing the number of processor cores per server amortizes platform overheads. However, if such chip-level integration is not on the processor architect's roadmap, platform-level integration is an alternative. In the latter scenario, place eight processors onto a single motherboard. The motherboard is customized to eliminate peripheral overheads so that processor power comprises 43 percent of the total. Like the Integrated Atom platform, more of each Watt dissipated contributes to query throughput. Relative to the Xeon baseline, energy efficiency improves by 2.1×.

However, this strategy increases net server cost. Building a sixteen-core platform from $45, dual-core Atom processors may be less expensive than integrating cores into a large chip multiprocessor. However, motherboard customization may be very expensive. We estimate the price that data center operators might be willing to pay for the power savings of the Optimized Atom platform. In particular, assuming the owner is unwilling to see an increase in the cost of operating a data center, we sweep a range of motherboard prices to identify the maximum price the owner would be willing to pay for customization. As we sweep the price of motherboard customization upwards, we find the point where the marginal increase in server cost exceeds the

marginal decrease in power cost. In this case, a custom motherboard costs $1,350. See Appendices A-B for details and assumptions.

The switch to energy-efficient, mobile processors likely increases server cost. For Integrated Atom and Optimized Atom platforms, we observe a drop of $0.7\times$ and $0.4\times$ in query throughput per dollar spent on servers. However, server cost is an incomplete analysis of cost. Accounting only for capital cost of computing hardware, it neglects electricity savings and other operating costs. For this reason, we also perform a total cost of ownership analysis. Measured holistically, we find increases in total cost of ownership are accompanied by proportional increases in capability.

6. DATA CENTER ARCHITECTURE AND COST

Although the Integrated Atom and Optimized Atom platforms are energy-efficient, they likely increase server cost. For Internet-scale applications, both performance and cost play an important role in determining the type and size of compute clusters. Balancing the cost with the service experience is also important. To determine the net impact on data center capability, we quantify server cost in terms of data center total cost of ownership (TCO). In particular, we examine two measures of cost efficiency: total cost per server and total cost per capability. In both cases we study the systems operating at sustainable throughput.

6.1. Total Cost of Ownership

We quantify TCO in dollars per server per month. See Appendix B for details and assumptions. TCO is a holistic metric that accounts for capital and operating costs. For the data center, capital costs are attributed to facility construction and hardware procurement. Operating costs are attributed to power usage. In our TCO analysis, we consider a data center with 15MW critical load, which defines the power budget allowed for computing hardware. Assuming a power usage efficiency (PUE) of 1.7, the data center dissipates a peak of 25.5 MW. This PUE multiplier accounts for electricity overheads in delivery and in the facility. See Appendix B for additional detail and assumptions.

Tables V-VI quantifies total cost of ownership and cost efficiency. Given a fixed data center power budget (15MW critical load, 25.5mW total), we examine trade-offs between the Xeon baseline and the three Atom alternatives. Since critical load is fixed, operating costs due to power usage do not change as we consider different platforms. However, as indicated in Tables III-IV, these platforms differ in cost and power. These differences affect server capital costs and the number of servers that fit in the critical load, respectively. Ultimately, the number of servers affects data center capability.

Consider the Xeon Harpertown and the Atom Diamondville. According to Table III, Atom Diamondville costs $0.38\times$ and $0.36\times$ less than Xeon Harpertown, in server price and power dissipated, respectively. As a result, $2.7\times$ the number of Diamondville servers fit into the 15MW critical load. Although the number of servers increases dramatically, price per server is also much lower. The net effect is only a modest impact on total cost of ownership ($6.1M versus $5.8M). Thus, with Atom Diamondvilles, the number of servers increases and the TCO per server falls.

However, the number of servers do not provide a complete assessment since each Diamondville server is capable of sustaining fewer queries per second. Indeed, Xeon Harpertown supplies eight cores per server for Atom Diamondville's two. On top of this $4\times$ gap in core count, each Xeon core is capable of $2\times$ Atom core throughput. Collectively, this means each Harpertown server is capable of $8\times$ the throughput of a Diamondville server. A $3\times$ increase in the number of Diamondville servers cannot overcome this disadvantage and the net result is a $0.4\times$ fall in capability. Since both

Table V. Total cost of ownership ($ per month) for Xeon Harpertown and Atom Diamondville. See Appendix B for details and assumptions.

	Xeon Harpertown 4-core, 2-socket	Atom Diamondville 2-core, 1-socket
Server (capital, $M)	3.22	3.46
Facility (capital, $M)	1.30	1.30
Other (capital, $M)	0.28	0.28
Power (operating, $M)	1.04	1.04
Total ($M)	5.84	6.08
Servers (Count, $\times 10^3$)	78.95	219.94
TCO / Server ($)	74.00	27.65
Capability (QPS, $\times X_\theta \times 10^6$)	0.63	0.22
TCO / Capability ($ per X_θ)	9.25	27.65

Table VI. Total cost of ownership ($ per month) for Integrated Atom, which integrates eight cores per processor and two processors per server, and Optimized Atom, which eliminates motherboard components to reduce power overheads. See Appendix B for details and assumptions.

	Integrated Atom 8-core, 2-socket	Optimized Atom 2-core, 8-socket
Server (capital, $M)	6.75	16.1
Facility (capital, $M)	1.30	1.30
Other (capital, $M)	0.28	0.28
Power (operating, $M)	1.04	1.04
Total ($M)	9.37	18.67
Servers (Count, $\times 10^3$)	165.56	253.81
TCO / Server ($)	56.60	73.57
Capability (QPS, $\times X_\theta \times 10^6$)	1.32	2.03
TCO / Capability ($ per X_θ)	7.08	9.20

platforms incur comparable TCO, the net effect is higher cost per unit throughput. For this reason, naively deploying Atom platforms is ineffective.

A strategy with Integrated Atom is more attractive. By using Atom cores, this platform requires $0.5\times$ the power of a Xeon Harpertown, which means we can deploy approximately $2\times$ the number of servers within the 15MW critical load while TCO only increases by $1.6\times$ ($9.4M versus $5.8M). The net effect is a fall in TCO per server. Moreover, with eight cores per socket and two sockets per server, each Integrated Atom server contributes more substantial throughput than the Atom Diamondville and even exceeds the throughput of the Xeon Harpertown. By deploying highly integrated Atom multiprocessors, data centers can increase peak capability and reduce cost per unit throughput.

This strategy is confirmed for the Optimized Atom, which deploys eight dual-core Atom processors on a highly optimized motherboard. By using Atom cores and by eliminating motherboard power overheads, this platform requires $0.3\times$ the power of a Xeon

Harpertown, which means we can deploy approximately 3× the number of servers, which produces a proportional increase in peak query throughput.

TCO per server is comparable for Xeon Harpertown and Optimized Atom. This similarity is an artifact of our motherboard price estimate (Appendix A). In particular, given uncertainty about motherboard prices, we estimate the maximum price a data center would pay for a more power-efficient motherboard by pricing the motherboard such that Optimized Atom TCO per server is no more than Xeon Harpertown TCO per server. If the Optimized Atom were more expensive than estimated in Table IV, TCO would increase and the Optimized Atom would be less cost-efficient than the Xeon Harpertown.

6.2. Capability

Power efficiency increases data center capability. As each server dissipates less power, more servers can be deployed within the same footprint. As demand for data center capability increases, deploying more efficient computing hardware is likely preferable to simply building more data centers. The total cost of ownership analysis highlights the advantages of the former. In particular, if processors provide capability, we would prefer processors to comprise a large fraction of server power and servers to comprise a large fraction of data center cost. The Integrated Atom and the Optimized Atom achieves both goals.

Processors in the Server. Figure 22 illustrates server power and the breakdown across various components. Although processors comprise more than 65 percent of the 190W dissipated by the Xeon Harpertown, absolute power costs are high. To address processor power costs, we consider deploying Atom Diamondvilles. However, the platform is sub-optimal as processor power comprises a small fraction of total power. Although absolute power costs are low, only a small fraction of dissipated power is going toward computation due to motherboard overheads. To increase processor power as a fraction of the total, the Integrated Atom and Optimized Atom amortize overheads over sixteen processor cores by integrating more cores per socket or integrating more sockets per server. To reduce absolute power costs, motherboard optimizations are necessary.

The final result is an architecture that reduces absolute power costs from 190W to 60W. Moreover, these power costs are dissipated on the processor and capability, not on overheads. Processors dissipate 65 percent of Xeon Harpertown power, they dissipate a sizeable 43 percent of Optimized Atom power. Once processor power has been optimized, server architects might consider optimizing memory and storage.

Servers in the Datacenter. Figure 23 illustrates the breakdown of TCO. TCO is comprised of capital costs for server and the facility; these costs are amortized over their respective 3 and 15 year lifetimes. Power is the largest component of operating cost for both servers and facility. In the baseline Xeon Harpertown and Atom Diamondville, of every dollar spent, only 55-57 percent goes toward servers and computing capability. The remainder goes toward overheads, whether in facility capital costs or utility operating costs. By deploying more efficient hardware, the Integrated Atom and Optimized Atom platforms spend 72-86 percent of every dollar on servers. This shift occurs as the number of servers increase while data center overheads remain unchanged. Thus, we achieve greater capability by exploiting infrastructure more effectively while avoiding the need to expand or to build new data centers with higher budgets.

As Tables V-VI indicate, however, extra capability increases total cost of ownership. As more power efficient processors are deployed to fill the same data center's critical load, server capital costs must necessarily increase. However, by quantifying TCO per server and TCO per capability, we find the benefits of additional servers increase at

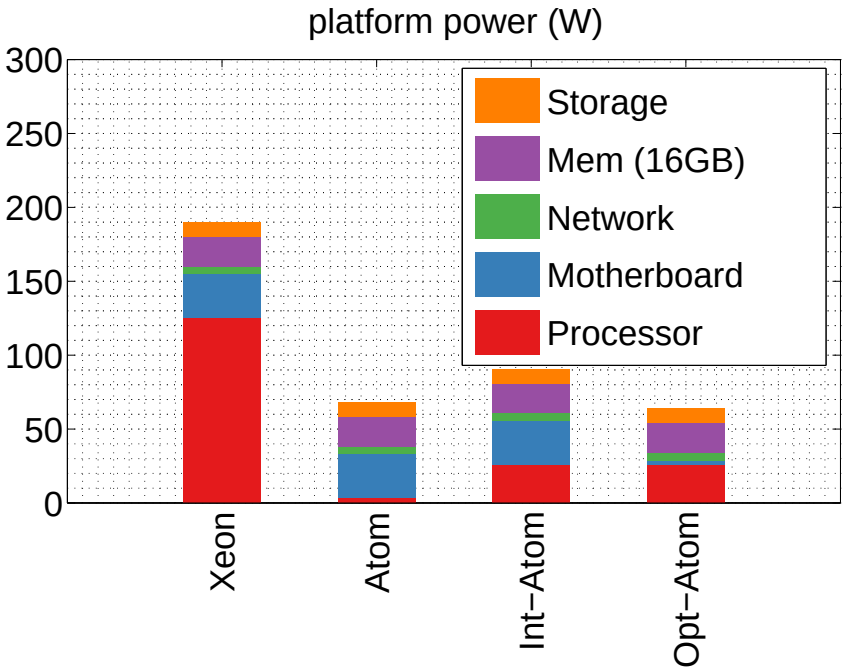
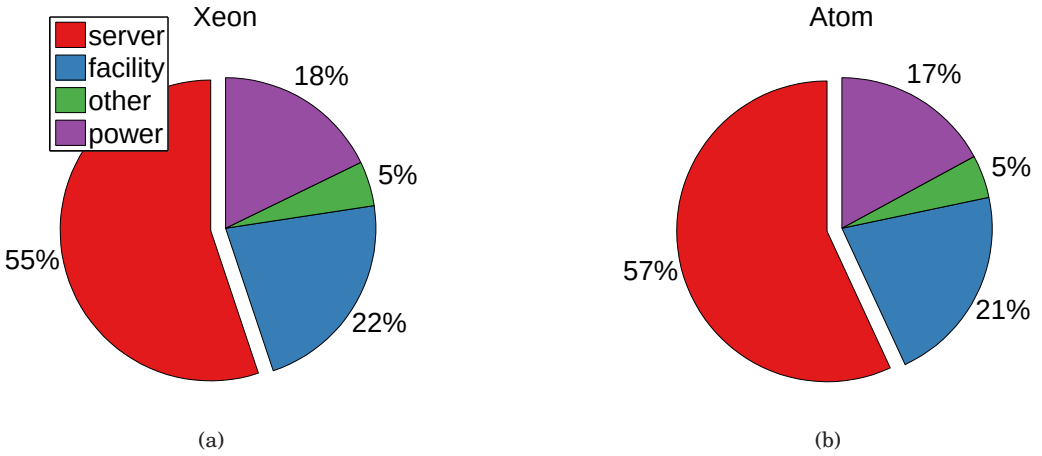
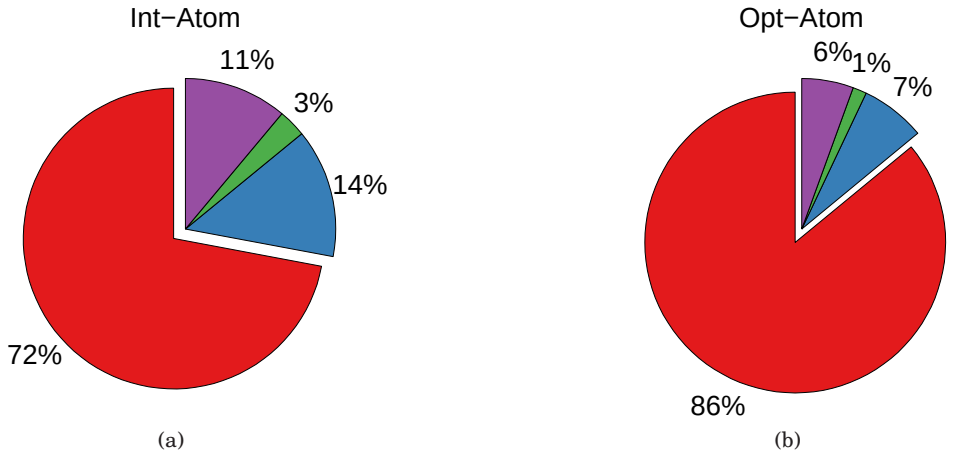


Fig. 22. Un-optimized Atom motherboards produce prohibitively high power overheads. Integrate multiple cores (e.g., 8 per chip) to amortize overheads over more cores. Optimize and share motherboard peripherals (e.g., Seamicro) to further reduce overheads.



(a) (b)

Fig. 23. Total cost of ownership for baselines, Xeon Harpertown and Atom Diamondville.



(a) (b)

Fig. 24. Total cost of ownership for integrated Atom (8 cores per chip) and optimized Atom (90 percent of motherboard peripherals eliminated).

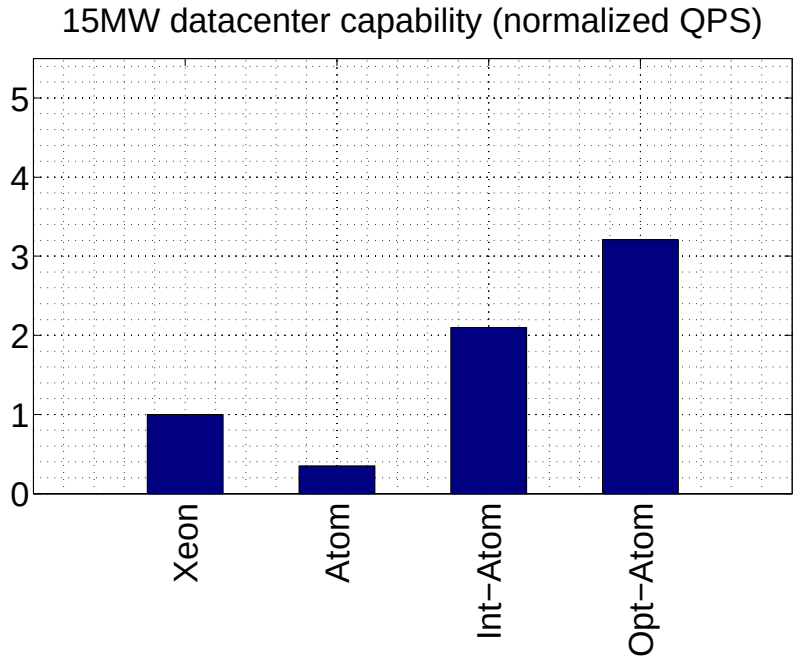


Fig. 25. Data center throughput increases as Atom energy efficiency allows a larger number of servers in a datacenter's fixed power budget (15MW).

least as fast as cost. Compared to Xeon Harpertown, we observe lower TCO per server and lower TCO per capability for both Integrated Atom and Optimized Atom.

Figure 25 summarizes the net effect on data center capability across the four server architectures considered. If poorly architected, Atom processors produce a net decrease in capability as overhead power dominates server power, limiting the number of deployed servers within the 15MW critical load. However, as server overheads are re-

duced either by greater core integration or platform optimization, more servers can be deployed within the same power budget, to increase capability by as much as $3.2\times$. Thus, data center power efficiency is inherently linked to data center capability.

7. RELATED WORK

Web search must identify and return relevant pages to the user [Brin and Page 1998]. Evolving beyond simply returning web pages, search engines increasingly provide additional information to help users make complex decisions [Microsoft Corporation 2009]. Moreover, relevance is computed with increasingly sophisticated techniques in statistical machine learning, which in our case takes the form of neural networks. As a result, web search becomes more similar to other analytic workloads that recognize, mine, and synthesize information from data [Dubey 2005]. Indeed, we find similar computational intensity when comparing analytic workloads, benchmarked by the PARSEC suite [Bienia et al. 2008], with Microsoft Bing. Moreover, these analytic workloads may increasingly deploy in the context of online web services. As online services, such analytical computation must meet specific quality-of-service targets [Ranganathan and Jouppi 2005].

In this work, we understand the characteristics of a more computationally intensive search engine. The results of this study may also be relevant for other analytics workloads under quality-of-service constraints.

7.1. Small Processor Cores

Commercial Computing. Small processor cores are proposed, initially, for chip multiprocessors. Targeting application domains with abundant thread-level parallelism, Piranha proposes eight simple Alpha processor cores on a single chip [Barroso et al. 2000]. Compared to more complex cores, each simple core is less capable. However, in aggregate, chip-level performance improves. Moreover, the design and deployment of simpler cores incur lower cost. Similarly, Niagara deploys simple cores with many threads, swapping out threads as they stall for memory or I/O [Geppert 2005; Kongetira et al. 2005]. Collectively, these architectures observe that several small cores in a chip multiprocessor can provide greater throughput than a single large core [Davis et al. 2005]. Similar arguments apply to commodity chip multiprocessors for web search [Barroso et al. 2003; Barroso 2005].

Also in commercial computing, FAWN deploys a fast array of wimpy nodes for a key-value storage system [Andersen et al. 2009]. FAWN-KV, is a consistent, replicated, highly-available, high-performance storage system built on this array. Back-end nodes deploy PCEngine Alix 3c2 devices, which target embedded computing. This architecture exploits the fact that key-value storage systems are I/O intensive and massively parallel [Vasudevan et al. 2010].

Chip multiprocessors target conventional applications in enterprise data centers. In these applications, response times are determined by network latency and are often assumed insensitive to computational latency at the server node. In contrast, we examine Microsoft Bing, in which computational latency impacts the quality of search results. Thus, we must analyze quality-of-service as well as energy efficiency. Small core latency effects may generalize to other applications in data analytics and mining.

Scientific Computing. Small cores also find application in high-performance, scientific computing. Exascale computing requires three orders of magnitude more capability within the same power footprint [Kogge et al. 2008]. Small cores are attractive for their power efficiency. IBM architects Blue Gene with PowerPC cores, which are designed for embedded computing. Accelerators for communication and floating-point calculation complement the general-purpose core and deliver performance. D. E. Shaw Research architects Anton with Tensilica cores, which provide a parameterized in-

struction set architecture and target embedded computing [Shaw et al. 2007; Kuskin et al. 2008; Larson et al. 2008]. Accelerators improve performance for molecular dynamics computation. Lawrence Berkeley National Laboratory deploys Tensilica cores for power-efficient climate modeling in the Green Flash system [Wehner et al. 2008].

In each of these cases, a simple core is used because power-efficiency is necessary to achieve qualitative improvements in capability. Blue Gene achieves petascale capability. Anton simulates molecular dynamics over milliseconds instead of microseconds. Green Flash models weather at the scale of kilometers instead of thousands of kilometers. Architects improve capability and efficiency in these domains by relying on accelerators. In future, this strategy may extend to commercial computing.

7.2. Servers and Atom Processors

SeaMicro is commercializing the concept of Atoms for servers. Unlike previous small core architectures, Atom's x86 instruction set architecture can natively support a broad range of applications. Online service providers may be characterized by a small number of very large applications (e.g., web search). As a result, such applications may be re-optimized and re-built to exploit the efficiency of another architecture [Barroso and Hölzle 2009]. However, efficient x86 processors will enable efficiency for a broader class of applications, which will benefit more general cloud and utility computing.

SeaMicro inspires the Optimized Atom platform in our study. Without control over processor design, SeaMicro organizes dual-core Atoms into a highly optimized server platform. Motherboard components are eliminated and peripheral components are multiplexed across many processor cores, thereby amortizing power overheads more effectively [Rao 2010]. The SeaMicro system provides greater compute density with 512 Atom cores in their SM10000 [SeaMicro 2011; Rao 2011].

Given our experiences with Microsoft Bing on the Atom, SeaMicro may significantly improve energy efficiency for carefully chosen applications. Our study highlights the trade-offs between energy efficiency and a variety of performance metrics. Other applications may experience different trade-offs. By building Atom servers, SeaMicro benefits from application portability. In contrast, severs using mobile processors that implement different instruction sets may require trade-offs between efficiency, performance, and portability.

7.3. Heterogeneity and Chip Multiprocessors

Heterogeneous chip multiprocessors are often proposed to realize both energy efficiency and high performance. Such architectures would combine large cores (e.g., Xeon) with small cores (e.g., Atom) and assign computation to the core best suited for it. Kumar et al. propose heterogeneous multiprocessors with a common instruction set architecture to facilitate workload scheduling and migration [Kumar et al. 2003; Kumar et al. 2004]. Because of the difficulty of optimizing the choice of heterogeneous cores, Kumar proposed cores already designed across Alpha generations. Addressing this challenge, Lee and Brooks apply statistical inference and clustering heuristics to identify efficient heterogeneous cores [Lee and Brooks 2007].

While early studies in heterogeneity considered cores for arbitrary workloads, heterogeneity can also target specific computation. Mogul et al. examine small cores for energy-efficient operating system computation [Mogul et al. 2008]. Small cores are attractive as operating system codes see modest benefits and large costs when running on complex, high-frequency cores. In contrast, Suleman et al. examine large cores for executing the critical section in parallel applications to mitigate Amdahl's Law [Suleman et al. 2009]. In this framework, highly parallel applications benefit from the energy-efficiency of small cores and incurs the power cost of high-performance cores only when necessary.

This study finds that fine-grained heterogeneity in chip multiprocessors incurs a high power cost for modest performance gains. However, a large and inefficient core may be needed if quality-of-service for complex queries is required. In this environment, heterogeneous chip multiprocessors may be relevant.

7.4. Data Center Design and Management

Data centers are effectively large computers with many inter-related components [Barroso and Hölzle 2009]. Data centers are often optimized for energy proportionality and energy efficiency. An energy proportional data center will scale power dissipation with the rate of computation [Barroso and Hölzle 2007]. In practice, proportionality is difficult as most servers dissipate a large fraction of its power when idle. While component-level energy proportionality is challenging, system-level proportionality might be achieved by turning off under-used servers and consolidating load. Mobile processors, such as the Atom, may improve system-level proportionality as power and capability can now be managed at finer granularities.

An energy-efficient data center will deploy more efficient hardware components. Energy-efficient cores must be deployed in an integrated fashion and controlled as an ensemble [Ranganathan et al. 2006]. For example, adopting mobile cores may also motivate low power memory and storage [Lim et al. 2008]. In particular, Lim et al. propose non-traditional hardware components for data centers. They propose embedded processors, disaggregated servers for memory, and Flash caches for storage. Deploying low-cost, low-power processors, Lim et al. find small performance penalties as they consider a range of workloads from mail, search, and media streaming that are not computationally intensive. With a small performance penalty, embedded processors reduce power usage and reduce TCO for a net gain in cost efficiency.

In contrast, this work considers a more computationally intensive search engine, which requires a deeper performance analysis. In particular, we study throughput, latency, quality-of-service, and end-user impact. We also study total cost of ownership. While Lim et al. report lower cost of ownership for the same number of servers, we examine the potential to increase the number of servers, thereby increasing data center capability. The underlying conclusions are the same; low-power processors reduce cost as long as performance lost is modest compared to energy efficiency gained.

8. CONCLUSION

While we study a particular mobile processor for a particular data center application, the framework we apply is broadly applicable to understanding emerging processor architectures and their interactions with emerging data center applications. It is not possible to claim mobile processors are effective for all application classes, but the results of this article suggest they may be effective for some applications not traditionally targeted by small processor cores, such as web search. We demonstrate a holistic and vertically integrated framework, which will help identify other applications amenable for execution on mobile processors. This framework starts with a detailed microarchitectural analysis and ends for a data center capability analysis. Throughout, we link hardware activity with software behavior. An application precisely tailored for an architecture, and vice versa, will always be more energy-efficient than the general-purpose alternative.

A. APPENDIX: PLATFORM COST AND POWER

Tables III-IV estimate the dollar and power cost for a variety of server components. This appendix details sources and assumptions.

A.1. Processor

We obtain processor dollar cost for the Xeon Harpertown L5420 and the Atom Diamondville 330, effective as of Sept 21, 2008. When bought in lots of one thousand, the L5420 processor is priced at $380 per processor and the 330 is priced at $45 per processor. Intel processor pricing sheets are available through Intel Corporation investor relations [Intel Corporation 2008c]. We measure processor power costs using a clamp ammeter while the processor runs Microsoft Bing (Section 2.5).

For the Integrated Atom platform, we estimate the price of an eight-core Atom chip multiprocessor based on area. Eight Atom cores fit in 200 mm^2. Because chip size impacts manufacturing yield, we assume price correlates with chip size. At 200 mm^2, the eight-core Atom multiprocessor occupies the same area as a four-core Xeon, which is priced at $380 per socket [Intel Corporation 2008c]. With two sockets, processors on the Integrated Atom cost $760. Power costs scale linearly with the number of cores.

For the Optimized Atom platform, we scale the price of a two-core Atom chip multiprocessor by $8\times$ to put a total of sixteen cores on a single motherboard at a cost of $360. Power costs scale linearly with the number of cores.

A.2. Motherboard

Xeon motherboard cost and power is quoted for the Intel S5000PAL [Intel Corporation 2009a; 2006]. A server motherboard supports two processor sockets and optimizes the form factor for high-density data centers. Atom motherboard cost and power is quoted for the Intel D945GCLF2 [Intel Corporation 2008b]. This desktop motherboard supports one processor socket.

For the Integrated Atom platform, we require two processor sockets to provision sixteen cores per server. We estimate motherboard price by assuming server motherboards are used instead of those for desktops.

For the Optimized Atom, we assume SeaMicro motherboard customization strategies, which remove all components, except for processors and memory, from the motherboard [Rao 2010]. These components are then made common and shared across hundreds of processors, thereby eliminating many of the power overheads that arise for duplicated motherboard components residing in each server. SeaMicro removes 90 percent of motherboard components from the server and we scale power accordingly.

Precisely estimating motherboard price for the Optimized Atom is difficult. For $148,000, SeaMicro ships the SM10000-64 with 256 dual-core Atom processors, one terabyte of DRAM, storage, and Ethernet uplinks [Rao 2011; SeaMicro 2011]. However, there is insufficient information to separate prices for different components. For this reason, we consider pricing motherboard in a way that does not impact total cost of ownership (Appendix B).

In particular, compute the data center total cost of ownership for the Xeon baseline. This cost accounts for amortized hardware and facility costs as well as electricity prices. For example, given Xeon server cost and power, we estimate Xeon TCO to be $74 per server per month. This estimate reflects the cost of the server and the number of servers that can be provisioned within a data center's 15MW critical load.

For the Optimized Atom platform, we estimate power for all components and prices for all components except motherboard. We then compute total cost of ownership for various motherboard prices until we identify the one that produces an Optimized Atom TCO of $74 per server per month, matching the TCO of the Xeon baseline. As indicated in Table IV, we find this price to be $1,350.

This approach quantifies the dollar savings that arise from less expensive Atom processors and from lower power costs per server. As motherboard price increases, the data center operator is willing to pay for a more expensive motherboard until its price

exceeds the cost savings from using the Optimized Atom platform. Thus, we estimate $1,350 is the maximum price a data center operator would pay to realize the motherboard customizations.

A.3. Network

We assume the network interface cost is negligible, either due to an integrated network interface or low bandwidth requirements. This assumption may not be valid if the data center requires high cross section bandwidth. However, unlike scientific computing, web search exhibits few computational dependences between index serving nodes. Commercial data center network traffic is often low and studies of such networks often use synthetic traffic patterns that scale up load for interesting studies [Abts et al. 2010; Heller et al. 2010]. Power estimates for the network interface card are based on those from Cisco [Cisco Systems 2009].

A.4. Memory and Storage

Memory prices are highly variable. We assume $300 for sixteen gigabytes. Power costs are estimated from Micron data sheets and system power calculator [Micron 2005]. More generally, 1.25W per gigabyte appears a reliable guideline.

Storage prices are also variable. In practice, the search engine does not require large capacity node storage and it very rarely accesses disk. Typical deployments include two to four SATA disks and use less than two percent of available disk bandwidth [Kozyrakis et al. 2010]. Power estimates for disks are based on those from Seagate [Seagate 2010].

B. APPENDIX: TOTAL COST OF OWNERSHIP

Sections 5-6 compare and contrast total cost of ownership for various platform alternatives. This appendix details assumptions.

B.1. Capital and Operating Costs

Section 6 computes total cost of ownership for a variety of server architectures and platforms using Hamilton's model and assumptions [Hamilton 2008]. In particular, we compute the cost per server per month. This cost includes both capital and operating costs. We consider a data center with a 15MW critical load, which defines the power budget attributed to computing hardware.

Capital costs include the cost of building the data center facility. The model assumes $200 million amortized over the facility's lifetime of 15 years (i.e., 180 months). In this analysis, we consider cooling and power infrastructure as "facility" while other data center related capital costs are classified as "other." The data center must be populated with servers. We estimate the total number of servers by dividing the 15MW critical load by the power dissipated by each server. As server efficiency improves, the number of servers increases. Server power and cost estimates are outlined in Tables III-IV. Server hardware is amortized over its lifetime of 3 years (i.e., 36 months). Amortizing the cost of facilities and servers over their respective lifetimes, we estimate monthly capital cost.

Operating costs depend on power usage. Since we assume a 15MW critical load, the power dissipated by servers is fixed; only the number of servers vary with server efficiency. In addition to critical load, we assume a power usage effectiveness of 1.7, which acts a multiplier to account for facility overheads. Thus, the data center peak power dissipation is 25.5MW. We assume the cost of power is $0.07/kWh. Multiplying these numbers, we estimate power cost per hour. We then scale by the average power used, which is assumed to be 80 percent of peak, and convert hours to months. Thus, we estimate the monthly operating cost.

B.2. Cost Efficiency

Total cost of ownership is the sum of amortized capital cost and operating cost. We consider two measures of cost efficiency. TCO per server is measured in units of dollars per server per month. The TCO analysis provides dollars per month for the datacenter with 15MW critical load. Divide this TCO by the number of servers that fit in this critical load to calculate TCO per server.

TCO per capability is measured in units of dollars per unit throughput. The TCO analysis provides dollars per month for the data center. To determine capability, consider the number of servers that fit within a 15MW critical load. Then discount Atom-based platforms, observing that an Atom core is only capable of sustaining half the query throughput that is sustained by a Xeon core. Thus, we measure capability in units of X_θ queries per second.

ACKNOWLEDGMENTS

We thank Preet Bawa, William Casperson, Utkarsh Jain, Paul England, Mark Shaw, CJ Williams, Tanj Bennett, David Brooks, Qiang Wu, Dan Connors, and Michael Smith for their feedback on this work. This work is funded, in part, by the National Science Foundation under Grant #0937060 to the Computing Research Association for the CIFellows Project. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessary reflect the views of the National Science Foundation or the Computing Research Association.

REFERENCES

ABTS, D., MARTY, M., WELLS, P., KLAUSLER, P., AND LIU, H. 2010. Energy proportional datacenter networks. In *Proceedings of the 37th International Symposium on Computer Architecture (ISCA)*. ACM, New York, NY, USA, 338–347.

ANDERSEN, D. G., FRANKLIN, J., KAMINSKY, M., PHANISHAYEE, A., TAN, L., AND VASUDEVAN, V. 2009. FAWN: A fast array of wimpy nodes. In *Proceedings of the Symposium on Operating Systems Principles (SOSP)*. ACM, New York, NY, USA, 1–14.

BAEK, W. AND CHILIMBI, T. M. 2010. Green: A framework for supporting energy-conscious programming using controlled approximation. In *Proceedings of the Conference on Programming Language Design and Implementation (PLDI)*. ACM, New York, NY, USA, 198–209.

BARROSO, L. 2005. The price of performance. *ACM Queue 3*, 48–53.

BARROSO, L., DEAN, J., AND HÖLZLE, U. 2003. Web search for a planet: The Google cluster architecture. *IEEE Micro 23*, 22–28.

BARROSO, L., GHARACHORLOO, K., MCNAMARA, R., NOWATZYK, A., QADEER, S., SANO, B., SMITH, S., STETS, R., AND VERGHESE, B. 2000. Piranha: A scalable architecture based on single-chip multiprocessing. In *Proceedings of the 37th International Symposium on Computer Architecture (ISCA)*. ACM, New York, NY, USA, 282–293.

BARROSO, L. AND HÖLZLE, U. 2007. The case for energy-proportional computing. *IEEE Micro 40*, 33–37.

BARROSO, L. AND HÖLZLE, U. 2009. *The Datacenter as a computer - An introduction to the design of warehouse-scale machines*. Morgan & Claypool.

BIENIA, C., KUMAR, S., SINGH, J. P., AND LI, K. 2008. The PARSEC benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. ACM, New York, NY, USA, 72–81.

BRIN, S. AND PAGE, L. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International Conference on World Wide Web 7 (WWW)*. Elsevier Science Publishers B. V., Amsterdam, The Netherlands, 107–117.

CISCO SYSTEMS. 2009. Power and cooling savings with unified fabric. Tech. Rep. C11-497077, Cisco Systems.

DALLY, W. J., BALFOUR, J., BLACK-SHAFFER, D., CHEN, J., HARTING, R. C., PARIKH, V., PARK, J., AND SHEFFIELD, D. 2008. Efficient embedded computing. *IEEE Computer 41*, 27–32.

DAVIS, J., LAUDON, J., AND OLUKOTUN, K. 2005. Maximizing CMP throughput with mediocre cores. In *Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. IEEE Computer Society, Washington, DC, USA, 51–62.

DUBEY, P. 2005. A platform 2015 workload model: Recognition, mining, and synthesis moves computers to the era of tera. Tech. rep., Intel Corporation.

EASTWOOD, M., HARRINGTON, D., AND BRODERICK, K. 2009. Server Workloads 2009: Understanding the Applications Behind the Deployment.

GANTZ, J. F., CHUTE, C., MANFREDIZ, A., MINTON, S., REINSEL, D., SCHLICHTING, W., AND TONCHEVA, A. 2008. *The Diverse and Exploding Digital Universe*. IDC.

GEORGE, V., JAHAGIRDAR, S., TONG, C., SMITS, K., DAMARAJU, S., SIERS, S., NAYDENOV, V., KHONDKER, T., SARKAR, S., AND SINGH, P. 2007. Penryn: 45-nm next generation Intel Core 2 processor. In *Asian Solid-State Circuits Conference (ASSCC)*. 14–17.

GEPPERT, L. 2005. Sun's big splash. *IEEE Spectrum*.

GEROSA, G., CURTIS, S., D'ADDEO, M., JIANG, B., KUTTANNA, B., MERCHANT, F., PATEL, B., TAUFIQUE, M., AND SAMARCHI, H. 2008. A sub-1W to 2W low-power IA processor for mobile internet devices and ultra-mobile PCs in 45nm hi-K metal gate CMOS. In *International Solid-State Circuits Conference - Digest of Technical Papers (ISSCC)*. 256–257.

GROCHOWSKI, E. AND ANNAVARAM, M. 2006. Energy per instruction trends in Intel microprocessors. *Technology@Intel Magazine*, 1–8.

HAMILTON, J. 2008. Cost of power in large-scale data centers. In *http://perspectives.mvdirona.com*.

HEATHER DOUGHERTY. 2010. Facebook Reaches Top Ranking in US. http://weblogs.hitwise.com/heather-dougherty/2010/03/facebook_reaches_top_ranking_i.html.

HELLER, B., SEETHARAMAN, S., MAHADEVAN, P., YIAKOUMIS, Y., SHARMA, P., BANERJEE, S., AND MCKEOWN, N. 2010. ElasticTree saving energy in data center networks. In *Proceedings of the 7th Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association, Berkeley, CA, USA, 17–17.

INTEL CORPORATION. 2006. Intel 5000 series chipset memory controller hub (MCH). Tech. Rep. 313067-001.

INTEL CORPORATION. 2008a. 45nm Intel Core 2 Duo Processor: BAClears. *Intel VTune Performance Analyzer 9.1 Help*.

INTEL CORPORATION. 2008b. Intel desktop board D945GCLF2: Technical product specification. Tech. Rep. E45013-002US.

INTEL CORPORATION. 2008c. Intel processor pricing. Tech. Rep. September 21, 2008.

INTEL CORPORATION. 2009a. Intel server board S5000PAL/S5000XAL: Technical product specification. Tech. Rep. D31979-010.

INTEL CORPORATION. 2009b. Volume 1 basic architecture. *Intel 64 and IA-32 Architectures: Software Developers Manual*.

JUNEE, R. 2009. Zoinks! 20 hours of video uploaded every minute! In *http://youtube-global.blogspot.com/2009/05/zoinks-20-hours-of-video-uploaded-every_20.html*.

KOGGE, P. ET AL. 2008. *Exascale computing study: Technology challenges in achieving exascale systems*. Information Processing Techniques Office, Defense Advanced Research Projects Agency (DARPA).

KONGETIRA, P., AINGARAN, K., AND OLUKOTUN, K. 2005. Niagara: A 32-way multithreaded Sparc processor. *IEEE Micro 25*, 21–29.

KOZYRAKIS, C., KANSAL, A., SANKAR, S., AND VAID, K. 2010. Server engineering insights for large-scale online services. *IEEE Micro 30*, 8–19.

KUMAR, R., FARKAS, K., JOUPPI, N., RANGANATHAN, P., AND TULLSEN, D. 2003. Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction. In *Proceedings of the 36th International Symposium on Microarchitecture (MICRO)*. IEEE Computer Society, Washington, DC, USA, 81–.

KUMAR, R. AND HINTON, G. 2009. A family of 45nm IA processors. In *International Solid-State Circuits Conference - Digest of Technical Papers (ISSCC)*. 58–59.

KUMAR, R., TULLSEN, D., RANGANATHAN, P., JOUPPI, N., AND FARKAS, K. 2004. Single-ISA heterogeneous multi-core architectures for multithreaded workload performance. In *Proceedings of the 31st International Symposium on Computer Architecture (ISCA)*. IEEE Computer Society, Washington, DC, USA, 64–.

KUSKIN, J. S. ET AL. 2008. Incorporating flexibility in Anton, a specialized machine for molecular dynamics simulation. In *Proceedings of the 14th International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, New York, NY, USA, 343–354.

LARSON, R. H. ET AL. 2008. High-throughput pairwise point interactions in Anton, a specialized machine for molecular dynamics simulation. In *Proceedings of the 14th International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, New York, NY, USA, 331–342.

LEE, B. C. AND BROOKS, D. M. 2007. Illustrative design space studies with microarchitectural regression models. In *Proceedings of the 13th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE Computer Society, Washington, DC, USA, 340–351.

LIM, K., RANGANATHAN, P., CHANG, J., PATEL, C., MUDGE, T., AND REINHARDT, S. 2008. Understanding and designing new server architectures for emerging warehouse-computing environments. In *Proceedings of the 35th International Symposium on Computer Architecture (ISCA)*. IEEE Computer Society, Washington, DC, USA, 315–326.

MICRON. 2005. Calculating memory system power for DDR2. Tech. Rep. TN-47-04.

MICROSOFT CORPORATION. 2009. Microsoft's new search at Bing.com helps people make better decisions. Tech. Rep. May 28, 2009.

MIENIK, M. 2000. CPU burn-in homepage. In *http://users.bigpond.net.au/CPUburn*.

MOGUL, J., MUDIGONDA, J., BINKERT, N., RANGANATHAN, P., AND TALWAR, V. 2008. Using asymmetric single-ISA CMPs to save energy on operating systems. *IEEE Micro 28*, 26–41.

RANGANATHAN, P., IRWIN, P. L. D., AND CHASE, J. 2006. Ensemble-level power management for dense blade servers. In *Proceedings of the 33rd International Symposium on Computer Architecture (ISCA)*. IEEE Computer Society, Washington, DC, USA, 66–77.

RANGANATHAN, P. AND JOUPPI, N. 2005. Enterprise IT trends and implications for architecture research. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE Computer Society, Washington, DC, USA, 253–256.

RAO, A. 2010. SeaMicro technology overview. Tech. rep., SeaMicro.

RAO, A. 2011. SeaMicro SM10000-64 system overview. Tech. rep., SeaMicro.

REDDI, V., LEE, B., CHILIMBI, T., AND VAID, K. 2010. Web search using mobile cores: Quantifying and mitigating the price of efficiency. In *Proceedings of the 37th International Symposium on Computer Architecture (ISCA)*. ACM, New York, NY, USA, 314–325.

SEAGATE. 2010. Barracuda product manual. Tech. Rep. 100636864.

SEAMICRO. 2011. SeaMicro now shipping the world's most energy efficient x86 server with new 64-bit Intel Atom N570 processor. Tech. Rep. February 28, 2011.

SEILER, L., CARMEAN, D., SPRANGLE, E., FORSYTH, T., ABRASH, M., DUBEY, P., JUNKINS, S., LAKE, A., SUGERMAN, J., CAVIN, R., ESPASA, R., GROCHOWSKI, E., JUAN, T., AND HANRAHAN, P. 2008. Larrabee: A many-core x86 architecture for visual computing. *ACM Transactions on Graphics 27*, 18:1–18:15.

SHAW, D. E. ET AL. 2007. Anton: A special-purpose machine for molecular dynamics simulation. In *Proceedings of the 34th International Symposium on Computer Architecture (ISCA)*. ACM, New York, NY, USA, 1–12.

SULEMAN, M. A., MUTLU, O., QURESHI, M. K., AND PATT, Y. N. 2009. Accelerating critical section execution with asymmetric multi-core architectures. In *Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ACM, New York, NY, USA, 253–264.

VASUDEVAN, V., ANDERSEN, D., KAMINSKY, M., TAN, L., FRANKLIN, J., AND MORARU, I. 2010. Energy-efficient cluster computing with FAWN: Workloads and implications. In *Proceedings of the 1st International Conference on Energy-Efficient Computer and Networking*. ACM, New York, NY, USA, 195–204.

VMWARE. 2009. Vmmark benchmark. In *www.vmware.com/products/vmmark*.

WEHNER, M., OLIKER, L., AND SHALF, J. 2008. Towards ultra-high resolution models of climate and weather. *International Journal of High Performance Computing Applications 22*, 149–165.

Received February 2011; revised March 2011; accepted April 2011