Web Search Using Mobile Cores: Quantifying and Mitigating the Price of Efficiency

Vijay Janapa Reddi
Harvard University
vj@eecs.harvard.edu

Benjamin C. Lee
Stanford University
bcclee@stanford.edu

Trishul Chilimbi
Microsoft Research
trishulc@microsoft.com

Kushagra Vaid
Microsoft Corporation
kvaid@microsoft.com

ABSTRACT

The commoditization of hardware, data center economies of scale, and Internet-scale workload growth all demand greater power efficiency to sustain scalability. Traditional enterprise workloads, which are typically memory and I/O bound, have been well served by chip multiprocessors comprising of small, power-efficient cores. Recent advances in mobile computing have led to modern small cores capable of delivering even better power efficiency. While these cores can deliver performance-per-Watt efficiency for data center workloads, small cores impact application quality-of-service robustness, and flexibility, as these workloads increasingly invoke computationally intensive kernels. These challenges constitute the price of efficiency. We quantify efficiency for an industry-strength online web search engine in production at both the microarchitecture- and system-level, evaluating search on server and mobile-class architectures using Xeon and Atom processors.

Categories and Subject Descriptors

C.0 [**Computer Systems Organization**]: General—*System architectures*; C.4 [**Computer Systems Organization**]: Performance of Systems—*Design studies, Reliability, availability, and serviceability*

General Terms

Measurement, Experimentation, Performance

1. INTRODUCTION

Internet services are experiencing a paradigm shift in which computation and data migrates from clients to a cloud of distributed resources located in data centers. The commoditization of hardware, data center economies of scale, and Internet-scale workload growth all demand greater power efficiency to sustain scalability. Advances in chip multiprocessors comprised of small cores provide opportunities for power efficiency. Piranha and Niagara propose integrating simple cores to lower design effort and improve throughput, respectively [1, 7, 9]. These smaller cores deliver throughput with better power efficiency when compared to their low-latency, high-performance counterparts. Recent advances in mobile computing provide small cores with even greater power efficiency. Traditional enterprise and online transaction processing (OLTP) applications are amenable to execution on small cores because they are typically memory or I/O bound. Consequently, platform and overall system design, rather than microarchitectural design, determine whether the application's service requirements are met.

Emerging data center workloads increasingly invoke computationally intensive and performance-critical kernels, for which small cores may weaken service requirements and guarantees. Relative to the traditional sort, these workloads are comprised of computationally intensive and performance-critical kernels that have latency requirements as part of their service level agreement (SLA). Microarchitecture design for small cores will increasingly determine whether service requirements are satisfied for these emerging and more compute-prone workloads. Although small cores are advantageous for power efficiency and throughput computing, 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 from small cores.

We quantify the price of small-core power efficiency for industry-strength web search, Microsoft Bing. Unlike traditional enterprise workloads, our version of web search relies on machine learning kernels at its core, which significantly increases computation at index serving nodes in a way that contravenes conventional wisdom regarding small cores for enterprise workloads. We demonstrate the computational intensity of our search engine with respect to other workloads in Figure 1.[1] Our search engine experiences significantly more instruction level parallelism (ILP) and is the only workload in Figure 1 exhibiting an un-normalized IPC greater than one, which surpasses prior search engines that experience an IPC similar to traditional enterprise workloads [2, 3]. This demand for ILP indicates a need for more performance than is typically provided by simple, in-order cores. Consequently, the strategy of small cores for greater power efficiency has deep implications for our Internet-scale workload. We explore these implications and make the following contributions:

[1]All workloads are configured as per industry-standard settings[23] and are operating under typical load conditions, running natively on a 4-way superscalar processor (Table 1).

ISCA'10, June 19–23, 2010, Saint-Malo, France.

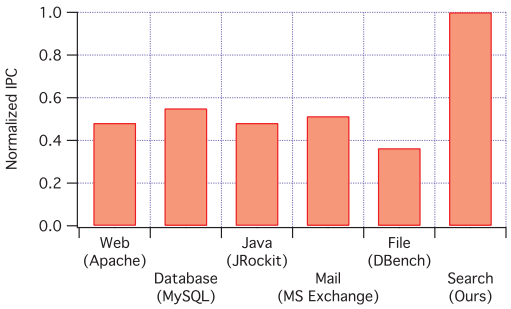


Figure 1: The computational needs of our search engine exceed that of traditional workloads with an un-normalized IPC noticeably greater than one.

- **Search:** We evaluate an online web search engine operating in an enterprise environment that uses machine learning kernels at the index serving nodes to service user queries. (Section 2)
- **Efficiency:** We compare the power efficiency of the server-class Xeon and mobile-class Atom microarchitectures for Search, relying on hardware counter data and multimeter power measurements. Search is 5× more efficient on Atom than on Xeon. (Section 3)
- **Price of Efficiency:** Search efficiency on Atom comes at the expense of robustness and flexibility. Quality-of-service guarantees become less robust as per query latency increases and becomes more variable. Computational intensity increases and microarchitectural bottlenecks impact search computation. (Section 4)
- **Mitigating the Price of Efficiency:** We address the price of efficiency through effective system design strategies, greater integration and enhanced microarchitectures. These strategies also address the total cost of ownership (TCO) at the data center level due to inefficiencies at the platform level, which translates to lower performance per dollar.

Collectively, the results in this paper contravene conventional wisdom with respect to small cores and data centers for online web search. The microprocessor industry offers a choice of two strategies to achieve the economies of scale necessary for this evolving workload: (1) start with big, high performance cores and improve efficiency or (2) start with small, low power cores and improve performance. We compare these two strategies and favor the latter for search.

2. SEARCH

Search is representative of a broad class of data center workloads that perform distributed and expensive computation. The workload requires thousands of processors to service queries under strict performance, flexibility and reliability guarantees. A single query might require tens of billions of processor cycles and access hundreds of megabytes of data [3]. Traditionally, parallel indexing affords substantial improvement in performance. However, as search engines continue to evolve and machine learning techniques become integral, per core performance is becoming more important again.

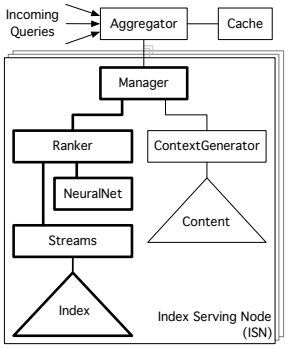


Figure 2: Overview of our specific Search instantiation. In this paper, we target the darkened subset within each index serving node (ISN).

2.1 Structure

Figure 2 outlines the structure of the search engine. Search queries enter the system through a top-level Aggregator. If the aggregator cannot satisfy a query from its set of frequently asked queries (i.e., Cache), it distributes the query to the Index Serving Nodes (ISNs). The ISNs operate in a distributed manner. Examining its subset of the document index, each node uses neural networks to identify and to return the top N most relevant pages and their dynamic ranks. Each ISN is responsible for ranking pages, as well as generating descriptions of relevant pages for the user [5].

Upon receiving a query, an ISN's Manager invokes the Ranker on the ISN's local subset of the index. The ranker parses the query and invokes Streams that use query features to identify a matching list of pages from query features. The ranker then uses statistical inference engines to compute a relevance score for each page in the matching list. Using these scores, the ranker identifies the top N results and passes these results to the aggregation layer, which in turn decides upon the set of results to return. After aggregating and sorting relevance scores, the aggregator requests captions from the ISNs. Each caption is comprised of a title, URL, and context snippets. Captions are generated by the ContextGenerator based on the content distributed to each particular ISN. Captions for the top N results across the ISNs are returned to the user in response to the query.

2.2 Requirements

Robustness. Search performance is quantified by a combination of Quality-of-Service (QoS), throughput, and latency. The application defines a QoS metric by the minimum percentage of queries handled successfully. For example, a QoS metric 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 QoS target constraint, we might consider a platform's sustainable throughput, which quantifies the maximum number of queries per second (QPS) that can arrive at a node without causing the node to violate the QoS constraint. If QPS exceeds the sustainable throughput, QoS 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. Architectures that exhibit gradual rather than sharp QoS degradations as load increases in absolute terms would provide greater 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, 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.

3. EFFICIENCY

Search exercises the datapath as it uses inference engines (e.g., Neural Nets) to service queries. Therefore, the workload is traditionally run using high-end server processors. But in this section, we seek to understand the trade-offs between using high-end processors and small-core designs. After explaining our experimental infrastructure, we compare search running on Xeon versus Atom, quantifying the microarchitectural, power and cost efficiency across both designs in terms of delivered search application performance.

3.1 Experimental Setup and Methodology

Microarchitectural Extrema. Considering the spectrum of commodity x86 microprocessors, we observe high-performance, server-class microprocessors at one end and low-power, mobile-class processors at the other end. Table 1 summarizes the microarchitectures we consider.

The Xeon is representative of modern, high-performance microarchitectures. We consider the Harpertown, a dual-die, quad-core processor comprised of Penryn cores [8, 14]. This processor implements several power optimizations. Implemented at 45nm using high-K dielectric and metal gate transistors, the process technology reduces leakage power by 5-10× and switching power by 30 percent. Moreover, the L2 cache is organized into 1MB slices, allowing cache resizing in 1MB increments for power reduction. Dynamic voltage and frequency scaling supports system-driven power mitigation. Finally, we consider a particularly low-power Harpertown part, the L5420, which operates at 2.5 GHz. The choice of a power-optimized Harpertown provides an optimistic baseline, which favors server-class architectures.

	Xeon	**Atom**
	Harpertown	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	Unavailable
Data TLB	256-entry, 4-way	Unavailable
L1 Inst/Data Cache	32/32KB	32/24KB
L2 Cache (per die)	12MB, 24-way	1MB, 8-way
FSB	1066MHz	533 MHz

Table 1: Microarchitectural extrema. We evaluate search on Xeon-Harpertown [8, 14, 21] and Atom-Diamondville [10, 14] processors that represent endpoints in the spectrum of x86 commodity processors.

The Atom is representative of modern, low-power microarchitectures. We consider a dual-core Diamondville [10, 14]. Each core is designed to operate in the sub-1W to 2W range. Diamondville cores implement an in-order pipeline with power efficient instruction decode and scheduling algorithms. The Diamondville datapath is also narrower than that of a Harpertown, issuing only two instructions per cycle. Furthermore, the Atom design avoids specialized execution units and favors general-purpose logic that can provide multiple functionality. For example, the SIMD integer multiplier and floating point divider are used to execute instructions that would normally execute on separate, dedicated scalar equivalents [10]. Such strategies are intended to reduce power, but also may have implications for performance as will be described in Section 4.

Workload Setup. While search is a distributed activity across several nodes, in this paper we specifically target its activity within an ISN. We use the application currently in production and drive it with real queries from user activity. Of the search components illustrated in Figure 2, we specifically examine the darkened subset that ranks pages online and returns the sorted results to the aggregator.

The quality of search or the relevance of pages depends upon ISN performance. CPU activity at an ISN ranges between 60 to 70 percent, indicating ISNs are usually under heavy load. Consequently, an ISN's performance depends on the capabilities of the underlying microarchitecture, which motivates our evaluation of the leaf-nodes within search. We neglect the aggregator, as well as the caption generators. From hereon, we loosely refer to the term "search engine" as the parts we are investigating.

The ISN computes page ranks for forty thousand queries after warmup. Input queries are obtained from production runs. The query arrival rate is a parameter within our experiments. We sweep this rate to identify the maximum QPS an architecture can sustain without violating the QoS target. For each query, the ISN computes overall ranks for pages that match the query for a 1 GB production index, a subset of the global index distributed across several nodes.

The index fits in memory to eliminate page faults and minimize disk activity. In effect, we consider the first tier of a tiered index, which resides in memory. Subsequent tiers require accessing disk, but the tiers are organized such

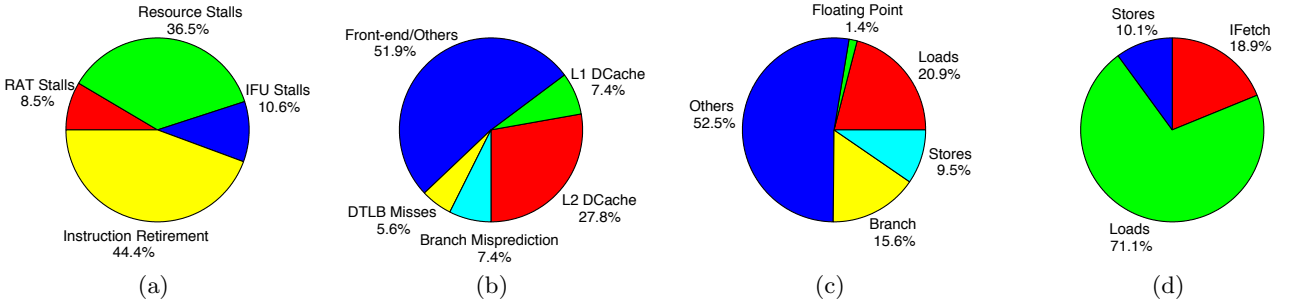


Figure 3: Microarchitectural synopsis of search on Xeon. (a) Execution versus stall time. (b) Breakdown of stall activity. (c) Instruction mix of search. (d) Sources of L2 cache accesses.

that this is extremely rare in practice. Therefore, it is the microarchitecture that determines application performance and not system-level configuration.

Additionally, to ensure fair comparison between Atom and Xeon, the total number of application threads servicing queries matches the number of hardware contexts available. More specifically, simultaneous multithreading (SMT) is enabled on Atom and results are normalized to a single Harpertown socket unless stated otherwise. System software (application and server OS) is also configured such that our setup is representative of a typical deployment strategy, allowing core-level comparison.

Microarchitectural and Power Measurements. We analyze the performance of the microarchitecture using details collected via VTune [13], a toolbox that provides us an interface to hardware counters on the Xeon and Atom. These counters provide detailed insight into microarchitectural activity as various parts of the search engine execute. To relate this microarchitectural activity to energy consumption, we quantify power dissipated by the processor at the voltage regulator module. We identify the 12V lines entering the regulator, apply a Hall-effect clamp ammeter (Agilent 34134A), and collect power measurements at 1KHz using a digital multimeter (Agilent 34411A).

3.2 Microarchitecture

Let X_θ be Xeon's sustainable throughput given the search application's quality of service target θ. Figure 3 illustrates microarchitectural events as search executes at X_θ queries per second on the Xeon. As shown in Figure 3(a), 55.6 percent of execution time is spent stalled for either the register alias table (RAT), the front end (IFU), or other resources (e.g., cache and memory). These stalls suggest the datapath is not fully utilized with only 44.4 percent of execution time spent retiring instructions due to structural conflicts in the front end (fetch or decode) or long latency memory instructions blocking instruction retirement. Figure 3(b) breaks down stall activity further.

Stalls during instruction fetch arise from branches and instruction cache effects. As illustrated in Figure 3(c), substantial branch activity of 15.6 branches per 100 instructions makes the branch predictor a bottleneck. Moreover, the datapath sees a seven cycle penalty when the number of in-flight speculative branches exceeds the capacity of the branch predictor (e.g., branch history table) [13]. Furthermore, the instruction fetch often stalls for L2 cache activity with 18.9 percent of L2 caches accesses attributed to instruction cache misses (Figure 3(d)).

Other resource stalls may be attributed to memory activity. According to Figure 3(c), a total of 30.4 percent of instructions either load or store data, which leads to pressure on the cache hierarchy. Consequently, Figure 3(d) shows that loads and stores account for 81.2 percent of all cache accesses. L2 cache activity often translates into memory requests with 67.0 percent of bus activity attributed to memory transactions (not shown). Application activity on Atom demonstrates similar behavior.

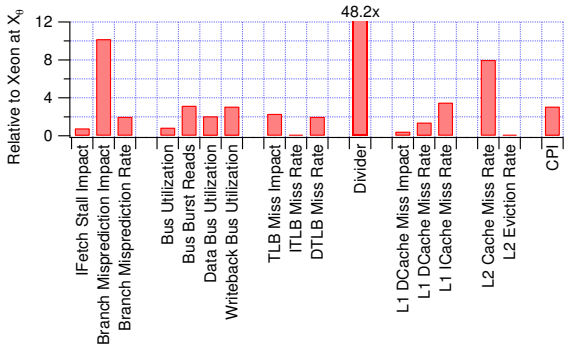


Figure 4: Atom with respect to Xeon.

Relative to Xeon, the Atom implements a simpler and deeper pipeline. As Figure 4 demonstrates, microarchitectural effects are aggravated in the low-power Atom. This data corresponds to Atom at A_θ, the Atom's sustainable throughput for quality-of-service target θ. The high frequency of branches and Atom's deeper pipeline depth leads to a 10× increase in performance penalties from mispredicted branches. Atom divide time is 48.2× greater than that of the Xeon. These performance effects may arise from the decision to favor generality over specialization for execution units. Specifically, "the use of specialized execution units is minimized. For example, the SIMD integer multiplier and Floating Point divider are used to execute instructions that would normally require a dedicated scalar integer multiplier and integer divider respectively." [10]

Moreover, on a per core basis, the Atom implements a much smaller cache hierarchy. According to Table 1, the Atom data and L2 caches are 25 and 66 percent smaller per core than their Xeon counterparts. This smaller cache hierarchy translates into 1.5× and 8.0× the number of data and L2 cache misses. Collectively, these microarchitectural effects lead to a 3× increase in cycles per instruction.

Table 2 compares throughput for search at its quality-of-service target θ. Let X_θ and A_θ be the sustainable throughput of the Xeon and Atom in queries per second (QPS). At the same θ, Xeon sustains 3.9× the throughput sustained by an Atom ($X_\theta = 3.9 \times A_\theta$). On a per core basis, each of

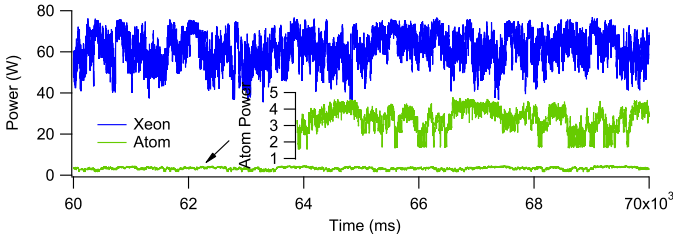


Figure 5: Xeon consumes ∼62.5W, where as the low-power Atom consumes ∼3.2W on average.

the four Xeon cores sustain 2.0× the throughput sustained by each of the two Atom cores. Sustainable throughput differs from absolute CPI because some processing is wasted on failed queries (time-outs), therefore system-level throughput gains can be less than microarchitectural gains.

3.3 Power

Figure 5 illustrates the power time series for both Xeon and Atom as search is running. The Xeon operates with an idle power component of 38.5W. The substantial idle power dissipated is particularly problematic given that the processor is stalled for 56.6 percent of its cycles. The Xeon exhibits a dynamic power range of 38.5 to 75W with a 95 percent difference between the idle and peak power. In contrast, Atom has a low idle power component of 1.4W with a 168 percent difference between idle and peak power. The Atom gets a drastic power reduction of 9.8× relative to Xeon. Atom's low idle power and large dynamic range is particularly attractive in the pursuit of energy proportional computing [4].

3.4 Performance

On a per core basis, the Atom is 5× more efficient than the Xeon. We compute performance and power efficiency as sustainable QPS per watt. Table 2 compares the Xeon and Atom over a variety of metrics. The power differentials (19.5× per processor, 9.8× per core) dominate the performance differentials (3.9× per processor, 1.9× per core). The large power cost of the Xeon is not justified by relatively modest increases in sustainable throughput.

Although this paper focuses primarily on power efficiency, we also mention area and price efficiency for comparison. Area efficiency is comparable between the two architectures, indicating the Xeon area overheads from dynamic instruction scheduling and out-of-order execution produce a linear improvement in throughput for search. However, Xeon price efficiency is 0.45× Atom price efficiency. Xeon prices are 8.4× Atom prices, but each Xeon sustains only 3.8× the number of queries per second. Note we consider price seen by the data center, not cost seen by processor manufacturers; manufacturers may target higher profit margins on server-class processors. Moreover, this analysis considers only equipment prices, not total cost of ownership. Peripheral components (e.g., motherboard, network interface card) will also impact the analysis. Section 5 will further assess the sensitivity of these effects.

4. PRICE OF EFFICIENCY

To understand the role of mobile-class architectures in data centers as workloads continue to emerge and evolve, we must identify and understand the price of exploiting small core efficiency. For search, we find quality-of-service becomes less robust as small cores are less capable of absorbing even modest increases in query load. Greater variance in the per query latency distribution indicates more queries experience higher latencies, which limit the time available to perform additional computation and improve search result relevance and ranking. Both quality-of-service and latency effects depend, in part, on the shifts in computational intensity and resource bottlenecks when using small cores.

4.1 Robustness

Quality-of-Service(QoS). Given a quality of service target θ, Xeon and Atom sustain a throughput X_θ and A_θ, respectively. We consider QoS guarantees robust if target θ is met despite fluctuations or temporary increases in query load. Robust QoS guarantees are a critical application requirement for search and data center applications. Robustness also determines the extent to which any one node can be utilized. If index server nodes can absorb activity spikes with little QoS degradation, they can operate closer to peak sustainable throughput.

For a given increase in query load, Figure 6 indicates robustness trends. The horizontal axis quantifies QPS normalized to Xeon's sustainable throughput X_θ. The vertical axis quantifies QoS in terms of the percentage of successfully processed queries. 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 QoS target.[2]

Latency. Increasing query load not only degrades the success rate in a quality-of-service metric, it also increases the latency at which that service is delivered. The search algorithm uses multiple strategies to refine search results if the query latency has not yet exceeded a cutoff latency L_C. Figure 7 illustrates latency trends with latencies normalized to the cutoff latency L_C. Using a logarithmic vertical axis, the figure indicates super-exponential increases in mean latency as query load increases on both Xeon and Atom. At their respective sustainable throughputs, Atom's latency is nearly 3× greater than that of the Xeon ($0.33L_C$ versus $0.12L_C$).

Relevance. QoS and latency shortcomings have a direct impact on search query relevance, which refers to a user's utility from a set of search results. Search algorithms often have multiple strategies for refining their results. However, load and cutoff latency L_C determine room 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. In such scenarios, lower per query latencies allow for multiple iterative refinements of search results to improve page relevance. Moreover, lower latencies provide opportunities to search algorithm designers for exploring latency-relevance trade-offs.

Since the average latency on Atom at A_θ is higher than on the Xeon at X_θ (see Figure 7), search on the Atom is unable to refine pages in a manner equivalent to the Xeon. Figure 8 shows how page relevance on the Atom compares to the Xeon at X_θ as load increases. The vertical axis compares matching pages returned by search on Xeon and Atom, defining baseline relevance with Xeon search results. For example, 100 percent means 100 percent of Atom queries return search results that perfectly match those from Xeon

[2]We study Xeon and Atom beyond sustainable throughput to characterize robustness. We do not assume sustainable operation under such loads.

	Per Processor			Per Core		
	Xeon	Atom	$\Delta_{X/A}$	Xeon	Atom	$\Delta_{X/A}$
Performance (QPS)	$3.86A_\theta$	A_θ	3.86×	$0.96A_\theta$	$0.5A_\theta$	1.93×
Power (W)	62.50	3.2	19.53×	15.63	1.60	9.77×
Power Efficiency (QPS/W, $\times 10^{-2}$)	$6.17A_\theta$	$31.25A_\theta$	0.20×	$6.17A_\theta$	$31.25A_\theta$	0.20×
Area (T, $\times 10^6$)	820	94	8.72×	n/a	n/a	n/a
Area (mm^2)	214	50	4.28×	n/a	n/a	n/a
Area Efficiency (QPS/mm^2, $\times 10^{-2}$)	$1.80A_\theta$	$2.00A_\theta$	0.90×	n/a	n/a	n/a
Price ($)	380	45	8.44×	95	22.50	4.22×
Price Efficiency (QPS/$, $\times 10^{-2}$)	$1.02A_\theta$	$2.22A_\theta$	0.45×	$1.02A_\theta$	$2.22A_\theta$	0.45×

Table 2: Search on Xeon versus Atom. Performance is quantified by sustainable throughput, which is the maximum number of queries arriving per second (QPS) that can be serviced successfully at the target quality-of-service θ. QPS is reported relative to Atom performance A_θ. Average power is used to compute efficiency. Area is reported per processor. Price is reported per unit purchased in orders of one thousand units.

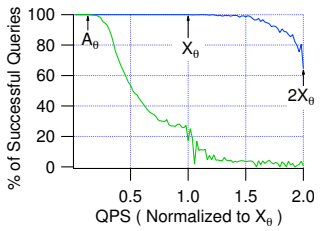


Figure 6: QoS on Xeon and Atom normalized to X_θ.

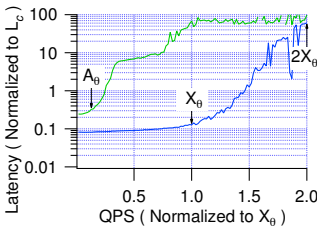


Figure 7: Latency on Xeon and Atom normalized to X_θ.

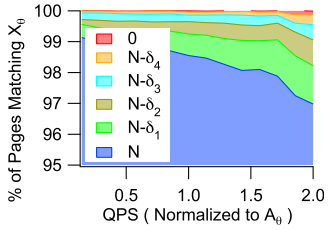


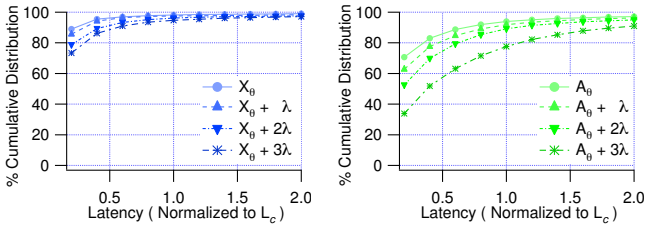
Figure 8: Bottlenecks on Atom limit the quality of page hits.

search. At 98 percent, the vertical axis indicates 2 percent of Atom queries produce different results relative to Xeon. A query's results match perfectly across architectures if the top N results match. The magnitude of the difference is quantified by $\delta_1 < \delta_2 < \delta_3 < \delta_4 < N$ where a reduced subset of $N - \delta_i$ of results match. In the worst case, none of the results match. This measure of relevance is highly conservative since it assumes Xeon defines the best relevance and any result mismatch degrades user perceived relevance. However, this metric facilitates the relevance analysis by quantifying search result differences.

Even considering the case when the load is at an absolute minimum on the Atom, approximately 1 percent of queries produce different results on Atom. Query latency on Atom at minimum load (e.g., 10 queries per second) is still higher than the latency on Xeon at X_θ (see Figure 7). Consequently, page refining algorithms have a third of the time to complete their work, leading to different search results from those on the Xeon. At A_θ about 1.5 percent of queries produce different results and the number of differing queries increases to 3 percent as load approaches $2A_\theta$.

4.2 Flexibility

A search engine frequently experiences user-generated activity spikes measured in absolute terms. Therefore, the underlying microarchitecture must be capable of adapting to rapid and significant shifts in search activity. This is especially the case in the presence of complex queries. Complex queries consist of multiple simpler queries that cannot be broken down and distributed across multiple nodes; each index serving node (ISN) must process the complex query. Even a modest spike in complex queries causes a significant absolute activity spike within an ISN. Therefore, nodes sensitive to activity spikes must be overprovisioned by operating below sustainable throughput to provide a safety margin. To understand this sensitivity, we evaluate how query latency



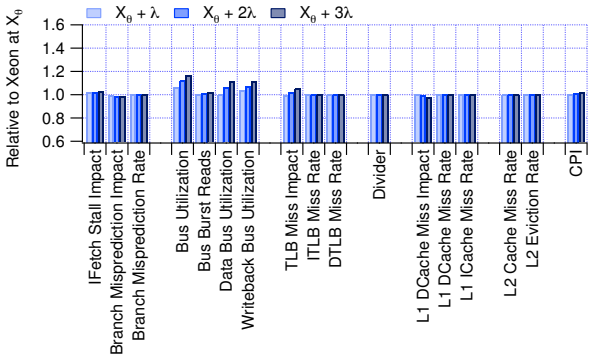
(a) Xeon (b) Atom

Figure 9: Query latency distribution. The experiment allows queries to exceed L_C and tracks the frequency in order to understand how the two processors handle sudden bursts in activity spikes.

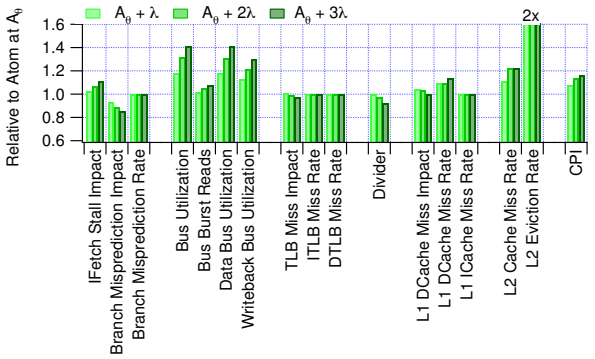
distribution changes as load increases in the presence of different types of queries. We explain the behavior through microarchitectural effects.

Cutoff Latencies. Figure 9 illustrates the latency distribution with latencies normalized to the cutoff latency L_C. The experiment allows queries to exceed L_C and tracks the frequency to understand how the two processors handle sudden bursts in activity spikes. Processing queries at X_θ, 89.4 percent of queries are satisfied in less than $0.2L_C$, 98.2 percent of queries are satisfied before the cutoff latency L_C, and less than 1.0 percent of queries require more than $2L_C$. Moreover, on the Xeon, these trends are modestly sensitive to activity spikes, which we quantify using λ. For an increased load of $X_\theta + 3\lambda$, 82.7 and 96.4 percent of queries are still satisfied in less than $0.2L_C$ and L_C.

Atom per query latency exhibits much greater variation and sensitivity to activity spikes (Figure 9(b)). Processing queries at A_θ on the Atom, only 68.2 percent of queries are satisfied in less than $0.2L_C$. At 93.4 percent, the number of queries completing before L_C on the Atom is comparable to that on the Xeon. However, nearly 3.0 percent of queries



(a) Xeon



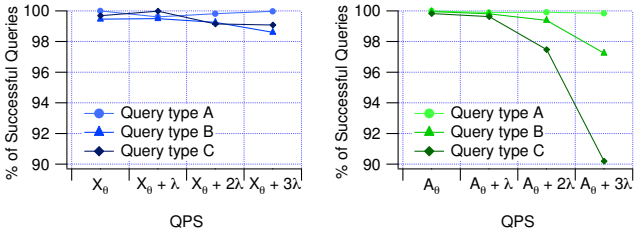
(b) Atom

Figure 10: Activity beyond sustainable query load.

require more than $2L_C$. Thus, compared to Xeon latency distributions, we observe a much larger spread in Atom's minimum and maximum per query latencies. Furthermore, Atom latency distributions are highly sensitive to activity spikes. For an increased load of $A_\theta + 3\lambda$, only 33.0 and 77.6 percent of queries are satisfied in less than $0.2L_c$ and L_C.

Figure 10 examines microarchitectural activity to provide deeper insight into the flexibility of the Xeon processor and the inflexibility of the Atom processor under load. Data is normalized with respect to activity on each processor at its sustainable throughput. Increases in query load minimally impacts the Xeon because next to added bus activity we see no other noticeable changes in Figure 10(a). However, query load increases of λ, 2λ and 3λ QPS beyond A_θ on the Atom degrades microarchitectural performance (CPI) by 7.5, 13.2, and 16.5 percent, respectively. These performance degradations arise primarily from increasing contention in the cache hierarchy. The small 1 MB, 8-way L2 cache becomes a constraint with L2 miss rate increasing by up to 22.2 percent and L2 eviction rate increasing by up to 100 percent. The increased eviction rate results in much higher bus utilization; writebacks increase linearly with the additional query load. As the memory subsystem becomes a bottleneck, the pipeline is more often stalled waiting for data. The divider utilization falls by 8 percent with an extra load of 3λ QPS.

Query Complexity. Atom processor's susceptibility to absolute load spikes is a function of query complexity. Query latency on the Atom takes a significant hit at throughput rates higher than A_θ because certain types of queries require more computation than others. Search criteria (e.g., language specification, conditional statements like ANDs, ORs etc.) determine query complexity, and how long it takes



(a) Xeon (b) Atom

Figure 11: QoS degradation by query complexity. Query type A has no search constraints. Query types B and C have increasing amounts of complexity, requiring more processing capability.

to process a query. Complex queries are broken down into simpler multiple individual queries within the ISN, so the "effective" query load is higher for complex queries.

To demonstrate how query complexity affects processor behavior and sustainable throughput, we isolate queries with different search criteria from the pool of mixed queries we use for all other evaluation and label them as query types A, B and C in Figure 11. Query type A has no complex search criteria, and is therefore fast to process. Query types B and C impose search criteria. Query type C contains many more constraints than type B, and thus requires even more processing.

Atom is able to sustain QoS equivalent to that of the Xeon for query type A even under high activity spikes (A_θ+3λ QPS) because of query simplicity. But the processor is unable to stay competitive for query types B and C as load increases beyond A_θ. At A_θ+3λ QPS the percentage of successful queries is only 90 percent when a stream of type C queries arrive back to back. These queries take longer to process, and consequently search queues begin to fill up and new incoming queries are dropped until search completes some pending queries or exceeds cutoff latencies . In contrast, Figure 11(a) shows Xeon does not suffer from this problem. Regardless of query complexity, Xeon is able to absorb activity spikes smoothly.

4.3 Reliability

Hardware or software-based failures are often threats in a data center. Therefore, we must understand how a homogeneous deployment of servers comprising of only Atom processors performs when load is re-balanced to accommodate failures. Load redistribution due to node failures leads to fractional or relative increases of sustainable throughput for a given processor. For instance, a processor will experience a relative load of 1.2θ when a hypothetical rack consisting of five systems experiences one system failure. Since the same absolute load is more evenly distributed across many more Atom ISNs, as compared to a data center full of Xeons, we find that on a per-node basis Atom achieves higher sustainable fail-over throughput compared to a Xeon.

Atom is more reliable for deploying search. To compare how QoS degrades due to fractional load increases across Atom and Xeon, we present the QoS data of each system normalized to its sustainable throughput in Figure 12. At loads slightly beyond sustainable throughput θ (1.0 QPS on the normalized axis), the Xeon latency trend line increases more sharply than that of the Atom. Atom QoS degrades to 95 percent at $2A_\theta$whereas Xeon QoS degrades to 64.5 per-

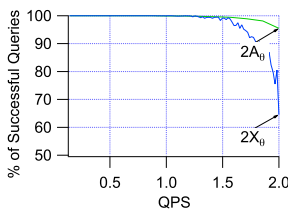


Figure 12: Comparing QoS normalized to each system's sustainable throughput.

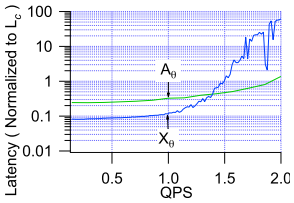


Figure 13: Comparing latency normalized to each system's sustainable throughput.

cent at $2X_\theta$. QoS degrades more gradually because the same fractional increase in load (e.g., 1.2×) on Atom corresponds to a smaller increase in the absolute number of queries on Xeon; $X_\theta$is larger than A_θ.

Xeon is unable to handle load increases robustly because of latency violations. Compare the latency on Xeon versus Atom with respect to each machine's sustainable throughput in Figure 13. Latency increases more gradually on the Atom than on the Xeon. Query latencies exceed L_C on the Xeon beyond $1.5X_\theta$. This means that even though the processor eventually locates pages corresponding to a query, the results are invalid because the aggregator terminates after L_C, assuming the ISN cannot produce the required pages. By comparison, latency on the Atom is still tolerable at $0.5L_C$ at A_θ, and thus its results are still valid.

Xeon is unable to scale to higher loads because the microarchitecture is saturated beyond $1.5X_\theta$. Figure 14 shows microarchitectural effects across the Xeon and Atom processors for relative load increases of 1.2×, 1.4×, 1.6×, 1.8×, and 2× beyond sustainable throughput. On Xeon, the processor's microarchitectural event activity plateaus from $1.6X_\theta$ onwards. Branch misprediction rates, bus utilization, TLB activity, cache activity all reach maximum levels. Therefore, the CPI increase is capped at ∼1.11x. While the Xeon system saturates at $1.6X_\theta$, Atom has more room for additional fail-over load, as indicated by increasing additional microarchitectural activity and CPI for the same relative loads.

Overall, these findings indicate that while Xeon is capable of sustaining higher throughput, it must run significantly under peak capacity to handle fail-overs, thus lowering throughput-per-watt further on top of already high power costs. As fail-over load per Atom server is smaller, and as its load is more distributed than in a data center comprising of fewer but higher capacity Xeons, Atom-based search is more reliable and energy-efficient. However, there are platform and system-level effects, in addition to flexibility sensitivity that impact any transition to Atoms.

5. MITIGATING THE PRICE OF EFFICIENCY

Strategies are necessary to mitigate the price of efficiency in using small cores for our web search. In this section, we propose holistic system-level strategies, as well as microarchitectural enhancements to individual cores to enhance robustness, flexibility and reliability guarantees.

5.1 Over-provisioning and Under-utilization

To improve QoS despite activity spikes, we might overprovision and underutilize Atom's. If we find an Atom node

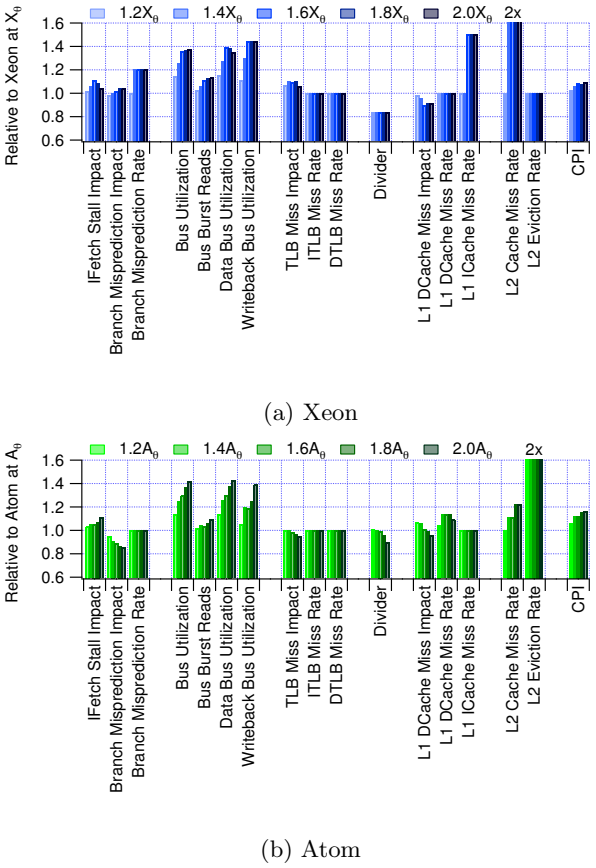


Figure 14: Microarchitectural activity during load redistribution because of fail-overs.

cannot tolerate a λ increase in QPS in a robust manner, we simply utilize the node at $(1-\lambda/A_\theta)$ percent. While this reduces throughput, the power difference between Atom and Xeon processors still favors mobile processors. Overprovisioning, however, incurs significant power overheads and has implications for the total cost of ownership (TCO). We break down TCO into server costs, their average power consumption, and the associated infrastructure for cooling and power distribution to provide insight into the system-level costs of Xeons and Atoms. We also assess the sensitivity of Atom's efficiency advantages to system-level costs.

Platform Overheads. Although the Atom core itself typically dissipates anywhere between 1.5 to 4.5W when running search, peripheral components on the motherboard contribute another 20 to 30W to platform power. To be successful, multiprocessor integration is necessary to help reduce these overheads. Through system engineering, Atom processors can deliver better platform-level performance-per-watt, since multiprocessor integration will amortize peripheral penalties over a larger number of processors. Based on our peripheral component analysis in Table 3, Atom-Diamondvilles exhibit only 0.4× the platform-level cost and power efficiency of the Xeon. Although the Atom processor is itself much more power-efficient, system engineering and optimization is required to reduce the overheads of peripheral components, such as the motherboard, network interface card, memory, and storage.

To reduce platform overheads, an amortization strategy is needed to integrate more Atom cores into a single processor

	Xeon		Atom			
	Harpertown 4-core, 2-socket		Diamondville 2-core, 1-socket		Hypothetical 8-core, 2-socket	
	Cost ($)	Power (W)	Cost ($)	Power (W)	Cost ($)	Power (W)
Processor	760	125	45	3.2	760	25.6
Motherboard	200	30	80	30	200	30
Network Interface	0	5	0	5	0	5
Memory (4GB)	150	8	150	8	150	8
Storage (HDD)	100	10	100	10	100	10
Total Server Cost	1210	178	375	56.2	1210	78.6
Normalized Efficiency	1.0 QPS/$	1.0 QPS/W	0.4 QPS/$	0.4 QPS/W	1.0 QPS/$	2.3 QPS/W

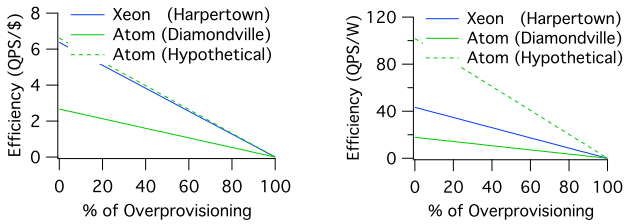
Table 3: Peripheral component costs. Xeon motherboard cost and power is quoted for the Intel S5000PAL [12] and Atom is quoted for the Intel D945GCLF2 [6]. Memory data is reported from Micron power specifications [16]. Storage data is reported from Seagate data sheets [19]. Processor and motherboard cost of the hypothetical Atom is based on the Xeon-Harpertown. Its power corresponds to the Atom-Diamondville.

die. We compute the cost of building a two socket system consisting of Atom-Diamondville processors by evaluating the number of Atom cores that fit in the area budget of a Xeon processor. Four dual-core Atom-Diamondville processors fit into the area of a single socket Xeon-Harpertown processor (50 sq-mm into 214 sq-mm). This hypothetical scenario is based on industry trends towards multiple in-order x86 CPU cores [20]. We assume processor and motherboard costs for the multiprocessor-based Atom are equivalent to the Xeon-Harpertown platform. As manufacturing costs are driven by die area and not by the number of cores for a given die size, we assume the same processor cost. Similarly, because of socket compatibility, we assume motherboard cost is the same. System memory and storage costs are independent of processor specifics, and therefore equivalent across all three platforms.

To test the benefits of multiprocessor integration, we compare the costs of a two-socket Xeon-Harpertown platform, a single-socket Atom-Diamondville platform (representative of current experimental systems), and a dual-socket Atom-Hypothetical platform (representative of projected integration). By integrating up to eight Diamondville cores into a single chip and building a system with two sockets, amortization once again highlights Atom efficiency. As per our computed cost in Table 3, an integrated low-power multiprocessors would be competitive with a cost and power efficiency advantage of 1.03× and 2.3× over the Xeon. In summary, our analysis prompts us towards better system integration to enable successful overprovisioning, while achieving much better performance-per-watt for approximately the same performance-per-dollar.

Efficiency Sensitivity. Depending upon application robustness, flexibility and reliability characteristics on a platform, a certain degree of overprovisioning is necessary. However, overprovisioning degrades efficiency by reducing the operational load on the processor. Therefore, while Table 3 reports efficiency differences at maximum sustainable load on Xeon and Atoms, there are cost and power efficiency trade-offs that require careful consideration while underutilizing and overprovisioning processors.

We present overprovisioning analysis of search in Figure 15. At peak throughput utilization, or 0 percent overprovisioning, the integrated Atom-Hypothetical is 1.0× as cost efficient and 2.3× as power efficient as the Xeon-Harpertown (see Figure 15(a) and Figure 15(b), respectively). However, when considering flexibility requirements, Harpertown efficiency improves. For example, if search requires a tolerance



(a) QPS-per-Dollar (b) QPS-per-Watt

Figure 15: Efficiency sensitivity to overprovisioning when processors are underutilized for robustness, flexibility and reliability service requirements.

of λ QPS against activity spikes, the Harpertown requires only 7.4 percent overprovisioning, whereas every core within the hypothetical Atom requires a 28.6 percent margin. Consequently, cost and power efficiency of Harpertown with respect to the hypothetical Atom improves by 25 percent and 12 percent to 1.25× and 0.55×, respectively. Based on the sensitivity analysis in Figure 15, we invite readers to identify target utilization levels that best fit production-level robustness, flexibility and reliability requirements.

Capital and Operational Costs. To study the costs of managing Xeons versus Atoms at the scale of an entire data center, we quantify TCO in terms of aggregate sustainable throughput-per-TCO dollar. We assume search is running on either the Xeon or Atom processors. We evaluate the Atom-Diamondville and Atom-Hypothetical.

We use a publicly available cost model to estimate data center capital and operational costs [11]. The model assumes $200M facility costs based on 15MW of critical power with a data center power usage efficiency (PUE) of 1.7. The model amortizes power distribution and cooling infrastructure costs over fifteen years and the purchase cost of servers over three years. The total cost of operating and managing the data center is presented on a monthly basis. The model categorizes the TCO into the following: power (electricity bill for running all servers), servers (purchase cost), cooling and power distribution (data center infrastructure power cost) and others (miscellaneous costs). To maintain our focus on platform overheads and power efficiency only, we constrain our TCO analysis by omitting discussion about network, software licensing, and personnel costs, assuming they are uniform across all deployments.

Figure 16 proportionally illustrates TCO costs associated with using Atom processors instead of Xeon-Harpertowns.

In going from the Xeon-Harpertowns (Figure 16(a)) to the Atom-Diamondvilles (Figure 16(b)), we suffer a 56 percent loss in performance-per-dollar spent managing the data center. Despite spending the same amount of money purchasing servers (51.4 percent on the Xeons versus 51.1 percent on the Atom-Diamondvilles), we get lower aggregate throughput from the Atom-Diamondvilles.

The loss in efficiency is mainly because of power inefficiencies at the system-level, which limit the number of Atom systems per data center. For a fixed critical power budget of 15MW, the data center houses 84,269 Xeons, which is far fewer than the 267,857 Atom-Diamondvilles possible within that same power envelope. But the Xeons generate 59 percent more throughput than the Atom-Diamondvilles. To match that throughput, the data center requires 650,000 Atom-Diamondvilles, far exceeding the data center's critical power budget by a factor of 3×.

With better system integration, however, an Atom-based data center can achieve approximately 1.5× the throughput-per-TCO dollar of Xeons. Figure 16(c) illustrates better throughput-per-dollar improvement through sheer increase in pie size with respect to the Xeons. Notice that this is a significant improvement at the data center-level in contrast to the system-level where the difference between Xeon-Harpertown and Atom-Hypothetical is negligible (see Table 3). The larger advantage at the data center-level arises from issues like power and cooling distribution. For instance, a multiprocessor-based Atom consumes approximately 3× less power than Xeons, and therefore its cooling requirements are lower, which benefits data center density.

System integration achieves better value per dollar spent managing the data center. Consider the distribution changes between the slices corresponding to Figure 16(a) and Figure 16(c). In the hypothetical Atom, server purchase cost is a significant portion of the monthly TCO, increasing from 51.4 percent for Xeons to 70.5 percent for Atoms. Despite this increase, using Atoms is opportune for two reasons. First, aggregate throughput using multiprocessor-based Atoms increases by over 2.3×. Second, considering the same critical power envelope of the data center, we are able to accommodate many more servers that are delivering higher aggregate throughput. By better integration, we void the need to expand or build newer data centers with higher power budgets, since we are capable of exploiting existing infrastructure more effectively. Moreover, compared to the purchase cost of additional servers, the overall operational and management costs of Atoms increase by only 60 percent relative to the Xeons. These trends, combined with lower expenditures for power and cooling infrastructure, indicate investment value will improve by transitioning to small cores.

5.2 Microarchitectural Enhancements

While overprovisioning addresses robustness QoS concerns at the system level, achieving lower latency per query is still essential on the Atom, especially for flexibility requirements in the presence of complex query load. Otherwise, the quality of search results will deteriorate as explained in Figure 8. To effectively achieve lower latency, we require microarchitectural enhancements to the Atom core. In Section 3, we found the Atom experiences particular stress on the divider, branch predictor, and cache hierarchy. Enhancing the Atom core based on those findings implies designing small cores using unconventional strategies in which, for example, a light-

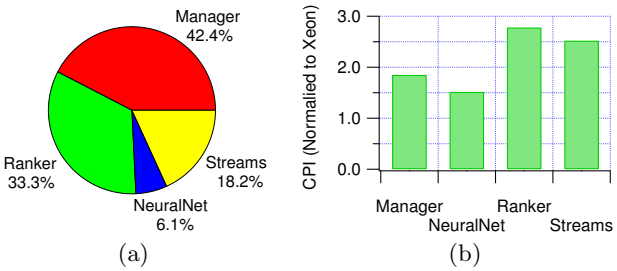


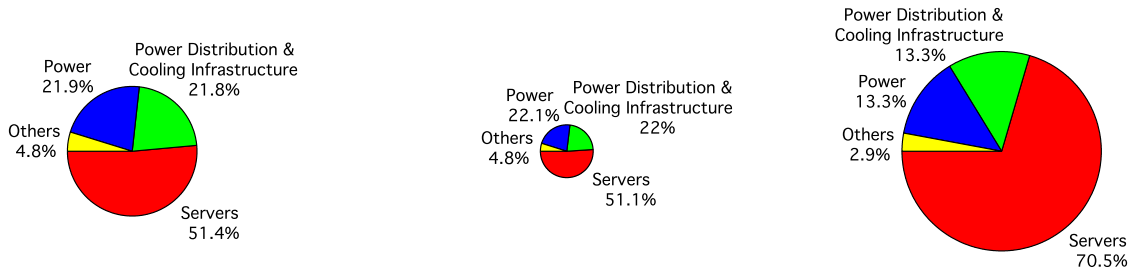
Figure 17: Search execution activity. (a) Execution time distribution across search phases. (b) Cycles-per-instruction (CPI) normalized to Xeon across the different phases of search computation on Atom.

weight datapath is paired with a disproportionately large cache hierarchy, at least for applications like search. These subtle modifications are likely to be more power efficient than transitioning to an out-of-order datapath.

To understand Atom bottlenecks and their root causes in the design, we must breakdown and analyze the different parts of search in isolation. Figure 17(a) illustrates that execution time is fairly distributed across all phases of search computation. This distribution gathered on a Xeon is identical to search on Atom. Additionally, Figure 17(b) suggests the most important functions exist throughout the phases of computation; all four major components of the search engine experience significant microarchitectural latency increases between 1.5× and 2.8× relative to Xeon performance. Atom performance degradation is broadly distributed and cannot be attributed to any one function or phase of computation.

Bottlenecks are limited to a few functions within each phase of search computation. We observe a significant overlap within the top 20 functions across both Xeon and Atom, indicating function importance depends more on the application and less on the architecture. Given the 20 most important functions and our knowledge of computation phases, we identify a representative function from each phase (Manager, NeuralNet, Ranker, and Streams). Figure 18 illustrates the microarchitectural events that affect the performance of those four representative functions. The events illustrate a broad range of performance limitations when adopting the small Atom cores. Just as no single function or single phase of computation accounts for Atom's performance limitations, no single microarchitectural event can be identified as the performance constraint.

Each function representing a different phase of search exercises different parts of the microarchitecture. Atom resources are constrained and particular stress is exerted on the divider, branch predictor, and cache hierarchy. 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 [10]. Atom's small 1MB, 8-way L2 cache leads to a 14× increase in L2 cache misses. The net effect is a 4.8× increase in CPI. 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 Xeon baseline. Such penalties may arise from the ranker's algorithmic components, which apply different strategies depending on the query and cutoff latencies. The



(a) Xeon (Harpertown) (b) Atom (Diamondville) (c) Atom (Hypothetical)

Figure 16: Total Cost of Operation (TCO) dollar. Each chart illustrates the breakdown of capital and operational expenses associated with sustainable throughput per monthly TCO dollar. With Xeon-based systems (a) as the baseline, pie charts (b) and (c) proportionally illustrate the return value per dollar spent managing the data center using either Atom-Diamondvilles, or our proposed integrated Atoms.

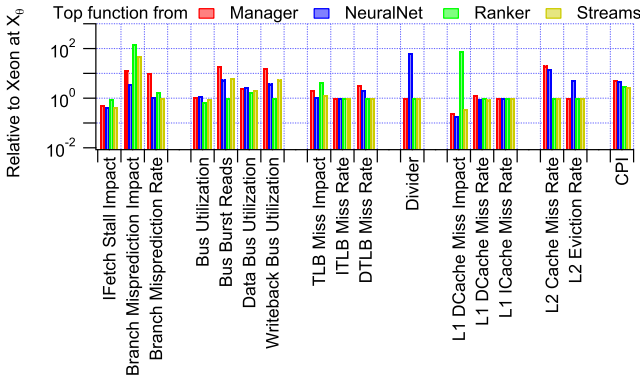


Figure 18: Identifying bottlenecks across phases of search using a representative functions.

impact of L1 data cache misses increases by 79× from a very low Xeon baseline. The net effect is a 2.9× increase in CPI.

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. Therefore, CPI increases by 2.8×. Lastly, 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 CPI. Overall, although we observe comparable 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, the ideal efficient microprocessor seems closer to the mobile-class end of the microarchitectural spectrum. For instance, reflecting back on the microarchitectural effects shown for activity spikes in Figure 10, we observe that the Xeon is significantly overprovisioned since activity spikes do not have any noticeable effect on the branch predictor, divider or the cache hierarchy. In migrating to simpler cores, some of that logic or area overhead can be redirected towards our proposal for (traditionally) disproportional structures like large caches and bus bandwidth for mobile cores to reduce latency.

5.3 Towards Heterogeneous Cores or Accelerated Multiprocessors

Looking further beyond microarchitectural enhancements, heterogeneous multiprocessors might provide accelerators to mitigate computational bottlenecks. Prior efforts have been made to combine large and small cores into a single chip multiprocessor (Section 6). However, when considering task-level division (Manager, NeuralNet, Ranker, Streams) across big and small cores, our analysis indicates that such a solution is inefficient for our version of web search.

Search has no single function or phase of computation that can be carved out to execute efficiently on a large core, such as the Xeon. Consider the four phases of computation (Manager, NeuralNet, Ranker, Streams) and their potential performance gains on Xeon (4.8×, 4.6×, 2.9×, 2.8×) from Figure 18. Furthermore, pessimistically assume the Atom operates at its measured peak (4.3W) and optimistically assume Xeon operates at its measured idle (38.5W). Thus, we derive the minimum power increase of 16.3× when performing computation on Xeon versus Atom; the power increase will always be larger in practice. Even in this optimistic scenario, the power differential of 16.3× dominates the performance differential of 2.8-4.8×.

Speeding up the Manager, NeuralNet, Ranker, and Streams with a Xeon core in an heterogeneous multiprocessor would lead to 0.29×, 0.17×, 0.18×, and 0.29× the Atom efficiency, indicating the difficulty of efficient acceleration with heterogeneous multiprocessors comprised of general-purpose architectures. While it may be possible to extract parallelism at a granularity finer than computational phases, doing so requires significant re-engineering of our production web search, which is a complicated and costly task. It might also be feasible to direct complex queries to Xeon instead of Atom, since Xeon can handle ISN-level internal absolute load spikes more gracefully than Atom (see Section 4.2). However, the cost to efficiency is high.

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.

6. RELATED WORK

The landscape of data center workloads is changing. Ranganathan and Jouppi discuss the changing workload mix and

usage patterns, motivating the need for integrated analysis of microarchitectural efficiency and application service-level agreements in their survey of enterprise information technology trends [17]. We complement their prior work by examining the role of power efficiency from small cores with in-depth microarchitectural analysis relating to application robustness, flexibility, and reliability constraints.

Power efficient data center design is an active area of research. Barroso et al. make the case for energy proportional computing in which data center components utilize energy proportional to the amount of computation [3]. Ranganathan et al. propose power management schemes across an ensemble of systems to mitigate costs associated with power and heat [18]. Efficient microarchitectures are integral components of such efforts and we further our understanding of microarchitectural effects in this work.

Piranha and Niagara make the case for small cores to improve design efficiency and throughput for memory and I/O bound workloads, such as transaction processing [1, 7, 9]. In contrast, we quantify limitations of small cores, specifically mobile processors, for more computationally intensive data center workloads, such as online web search. Lim et al. take a system view of warehouse-computing environments and propose unified systems, which transition from high-end server-class processors to mobile and embedded processors [15]. Similarly, Vasudevan et al., propose using fast arrays of wimpy nodes (FAWN) to achieve energy efficiencies in data intensive computing [22]. Both efforts emphasize low-power embedded processors for performance (queries per second) and power efficiency, using system power measured at the electrical socket. In contrast, we take a microarchitectural view of the differences between server- and mobile-class processors and measure, in addition to system power, core power by tracking current activity at the voltage regulator.

Making the case for chip multiprocessors, Barroso quantifies the economic price of high-performance, high-power platforms as compared to smaller cores [2]. While small cores reduce the economic price, they increase the application's perceived price of efficiency as we quantify in terms of quality-of-service robustness, flexibility, and reliability. As emerging data center workloads exercise the data path more extensively, the application perceived price of efficiency raises qualifications and caveats in the case for small cores in data centers. Once these limitations are understood, microprocessor and system architects can better navigate the inherent trade-offs between performance and power efficiency.

7. CONCLUSION

Emerging data center applications exercise the processor more significantly than traditional enterprise and online transaction processing (OLTP) applications. Thus, we deeply examine the role of small cores, specifically mobile processors, for an Internet-scale workload: web search. In particular, we study the challenges with respect to robustness flexibility and reliability of search. The price of efficiency has implications on system design, implementation and deployment strategies. For example, in the future, reducing platform power overheads associated with peripheral components is essential. In the longer term, microarchitectural enhancements, heterogeneous multiprocessors, or accelerators customized to mitigate bottlenecks may be instrumental in reducing the price of small core power efficiency.

8. ACKNOWLEDGMENTS

We are extremely grateful to several people for helping us with this research undertaking, including the reviewers. We would like to specifically thank Preet Bawa, William Casperson, Utkarsh Jain, Paul England, Mark Shaw, CJ Williams, Tanj Bennett, David Brooks, Qiang Wu, Dan Connors and Michael D. Smith. We are also grateful to the National Science Foundation under Grant #0937060 to the Computing Research Association for the CIFellows Project. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the National Science Foundation or the Computing Research Association.

9. REFERENCES

[1] L. Barroso, K. Gharachorloo, R. McNamara, A. Nowatzyk, S. Qadeer, B. Sano, S. Smith, R. Stets, and B. Verghese. Piranha: A scalable architecture based on single-chip multiprocessing. In *ISCA'00*.

[2] L. A. Barroso. The price of performance: An economic case for chip multiprocessing. *Queue, ACM*, 2005.

[3] L. A. Barroso, J. Dean, and U. Holzle. Web search for a planet: The Google cluster architecture. *Micro, IEEE*, 2003.

[4] L. A. Barroso and U. Holzle. The case for energy-proportional computing. *Computer, IEEE*, 2007.

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *WWW7*, 1998.

[6] I. Corporation. Technical product specification. *Intel Desktop Board D945GCLF2*, 2008.

[7] J. Davis, J. Laudon, and K. Olukotun. Maximizing CMP throughput with mediocre cores. In *PACT'05*.

[8] V. George, S. Jahagirdar, C. Tong, K. Smits, S. Damaraju, S. Siers, V. Naydenov, T. Khondker, S. Sarkar, and P. Singh. Penryn: 45-nm next generation intel core 2 processor. In *ASSCC'07*, 2007.

[9] L. Geppert. Sun's big splash: Niagara multiprocessor chip. *IEEE Spectrum'05*.

[10] G. Gerosa, S. Curtis, M. D'Addeo, B. Jiang, B. Kuttanna, F. Merchant, B. Patel, M. Taufique, and H. Samarchi. 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 *ISSCC'08*.

[11] J. Hamilton. Cost of power in large-scale data centers. In *http://perspectives.mvdirona.com*.

[12] Intel Corporation. Thermal/mechanical design guide. *Intel 5000 Series Chipset Memory Controller Hub (MCH)*, 2006.

[13] Intel Corporation. 45nm Intel Core 2 Duo Processor: BAClears. *Intel VTune Performance Analyzer 9.1 Help*, 2008.

[14] Intel Corporation. Volume 1 basic architecture. *Intel 64 and IA-32 Architectures: Software Developers Manual*, 2009.

[15] K. Lim, P. Ranganathan, J. Chang, C. Patel, T. Mudge, and S. Reinhardt. Understanding and designing new server architectures for emerging warehouse-computing environments. In *ISCA'08*.

[16] Micron. Technical note TN-47-04: Calculating memory system power for DDR2. In *www.micron.com*, 2006.

[17] P. Ranganathan and N. Jouppi. Enterprise IT trends and implications for architecture research. In *HPCA-11*, 2005.

[18] P. Ranganathan, P. Leech, D. Irwin, and J. Chase. Ensemble-level power management for dense blade servers. In *ISCA'06*.

[19] Seagate. Barracuda 7200.12 data sheet. In *www.seagate.com*, 2009.

[20] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan. Larrabee: a many-core x86 architecture for visual computing. *ACM Trans. Graph.*, 2008.

[21] R. Swinburne. Intel Core i7 - Nehalem architecture dive. In *www.bit-tech.net*, 2008.

[22] V. Vasudevan, J. Franklin, D. Anderson, A. Phanishayee, L. Tan, M. Kaminsky, and I. Morau. FAWNdamentally power-efficient clusters. In *HotOS-XII*, 2009.

[23] VMware. Vmmark benchmark. In *www.vmware.com/products/vmmark*, 2009.