Reining in Long Tails in Warehouse-Scale Computers with Quick Voltage Boosting Using Adrenaline

CHANG-HONG HSU, YUNQI ZHANG, and MICHAEL A. LAURENZANO,
University of Michigan, Ann Arbor, MI
DAVID MEISNER, Facebook Inc., Menlo Park, CA
THOMAS WENISCH, RONALD G. DRESLINSKI, JASON MARS, and LINGJIA TANG,
University of Michigan, Ann Arbor, MI

Reducing the long tail of the query latency distribution in modern warehouse scale computers is critical for improving performance and quality of service (QoS) of workloads such as Web Search and Memcached. Traditional turbo boost increases a processor's voltage and frequency during a coarse-grained sliding window, boosting all queries that are processed during that window. However, the inability of such a technique to pinpoint tail queries for boosting limits its tail reduction benefit. In this work, we propose *Adrenaline*, an approach to leverage finer-granularity (tens of nanoseconds) voltage boosting to effectively rein in the tail latency with query-level precision. Two key insights underlie this work. First, emerging finer granularity voltage/frequency boosting is an enabling mechanism for intelligent allocation of the power budget to precisely boost only the queries that contribute to the tail latency; second, per-query characteristics can be used to design indicators for proactively pinpointing these queries, triggering boosting accordingly. Based on these insights, Adrenaline effectively pinpoints and boosts queries that are likely to increase the tail distribution and can reap more benefit from the voltage/frequency boost. By evaluating under various workload configurations, we demonstrate the effectiveness of our methodology. We achieve up to a 2.50× tail latency improvement for Memcached and up to a 3.03× for Web Search over coarse-grained dynamic voltage and frequency scaling (DVFS) given a fixed boosting power budget. When optimizing for energy reduction, Adrenaline achieves up to a 1.81× improvement for Memcached and up to a 1.99× for Web Search over coarse-grained DVFS. By using the carefully chosen boost thresholds, Adrenaline further improves the tail latency reduction to 4.82× over coarse-grained DVFS.

CCS Concepts:• **Applied computing → Data centers;**

Additional Key Words and Phrases: Datacenters, latency-critical workloads, tail queries, energy efficiency, fine-grained dynamic voltage/frequency scaling

ACM Reference Format:
Chang-Hong Hsu, Yunqi Zhang, Michael A. Laurenzano, David Meisner, Thomas Wenisch, Ronald G. Dreslinski, Jason Mars, and Lingjia Tang. 2017. Reining in long tails in warehouse-scale computers with quick voltage boosting using adrenaline. ACM Trans. Comput. Syst. 35, 1, Article 2 (March 2017), 33 pages.
DOI: http://dx.doi.org/10.1145/3054742

This article extends the version "Adrenaline: Pinpointing and Reining in Tail Queries with Quick Voltage Boosting" published at HPCA 2015 [Hsu et al. 2015]. This research was supported by Google and by the National Science Foundation under grants CCF-SHF-1302682 and CNS-CSR-1321047.
Authors' addresses: C.-H. Hsu, Y. Zhang, M. A. Laurenzano, T. Wenisch, R. G. Dreslinski, J. Mars, and L. Tang, Computer Science and Engineering Department, University of Michigan at Ann Arbor; emails: {hsuch, yunqi, mlaurenz, twenisch, rdreslin, profmars, lingjia}@umich.edu; D. Meisner, Facebook Inc.; email: meisner@fb.com.

DOI: http://dx.doi.org/10.1145/3054742

1. INTRODUCTION

Managing high-percentile tail latency is one of the chief performance concerns for web services in modern *Warehouse Scale Computers* [Dean and Barroso 2013]. These online data intensive (OLDI) services traverse multi-terabyte datasets for user-facing latency-sensitive queries by dividing ("*sharding*") their datasets over thousands of servers ("*leaf nodes*") acting in concert [Barroso et al. 2003]. A complete query response is formed by aggregating the responses from the individual leaf nodes. However, at 100- to 1,000-node scale, the overall latency distribution to respond to the user is often dominated by a single straggling leaf node. For example, if an individual leaf node has only a 1-in-100 chance of exceeding a 1s response time, when aggregating parallel requests to 100 nodes, 63% of queries will take longer than 1s [Dean and Barroso 2013]. Due to this service-level sensitivity to leaf-node tail latency, optimizations to reduce the long tail are paramount. Indeed, sacrificing mean latency for improved tail latency is encouraged [Dean and Barroso 2013].

Voltage and frequency boosting techniques, such as Intel's Turbo Boost, enable the processor to run above its base operating frequency and can be used to reduce computation time and thus shorten the long tail in the latency distribution. However, existing hardware-managed control policies are application oblivious; they adjust frequency and voltage based on the number of active cores [Emurian et al. 2014] and are unable to distinguish between low-latency queries and those responsible for the long tail. During a boosting period, all queries are processed at a higher voltage/frequency (V/f) irrespective of their latency profile or whether they benefit from boosting. Recently proposed workload- and latency-aware mechanisms, such as PEGASUS [Lo et al. 2014], adjust voltage/frequency to the large diurnal variations in service load to conserve energy while respecting a tail latency constraint. Such approaches similarly fail to distinguish between typical and tail queries instead modulating performance of the entire query latency distribution to meet the tail latency constraint. Coarse-grained uniform boosting leads to energy-inefficient tail reduction; energy is wasted accelerating queries that are not in the tail.

Considering that the query latency for many OLDI services is in the range of milliseconds and microseconds [Lim et al. 2013; Barroso et al. 2003], the emerging class of fine-grained (tens of nanoseconds) voltage boosting (i.e., *quick boosting*) techniques [Kim et al. 2008, 2011; Pinckney et al. 2013; Miller et al. 2012; Godycki et al. 2014] has the potential to enable precise query-level boosting approaches. Given an energy budget, an intelligent quick boosting strategy could precisely pinpoint and boost queries that contribute to the tail as well as those whose latency is more likely to benefit from frequency/voltage boosting. Figure 1 illustrates the limitation of prior work and our insight. The top graph illustrates a typical heavy-tailed latency distribution of a leaf node. Coarse-grained boosting techniques such as those mentioned above accelerate all queries, compressing the entire latency distribution until the 99% latency meets the target, unnecessarily accelerating the bulk of requests near the mean. Instead, by applying fine-grained boosting to queries that are both sensitive to frequency and lie in the tail, our approach skews the distribution, significantly reducing the tail latency. However, several open questions remain to realize this approach, including the following:

(1) *Investigating whether tail queries are amenable to frequency / voltage boosting.* If tail queries in representative OLDI workloads are not bottlenecked on computation, then V/f boosting will not be effective in reducing their time to completion and thus would not be able to pull in the tail.
(2) *Determining whether tail queries are predictable.* If the queries that push out the long tail share common characteristics, then we can use these characteristics to develop per-query indicators to pinpoint these queries for boosting.

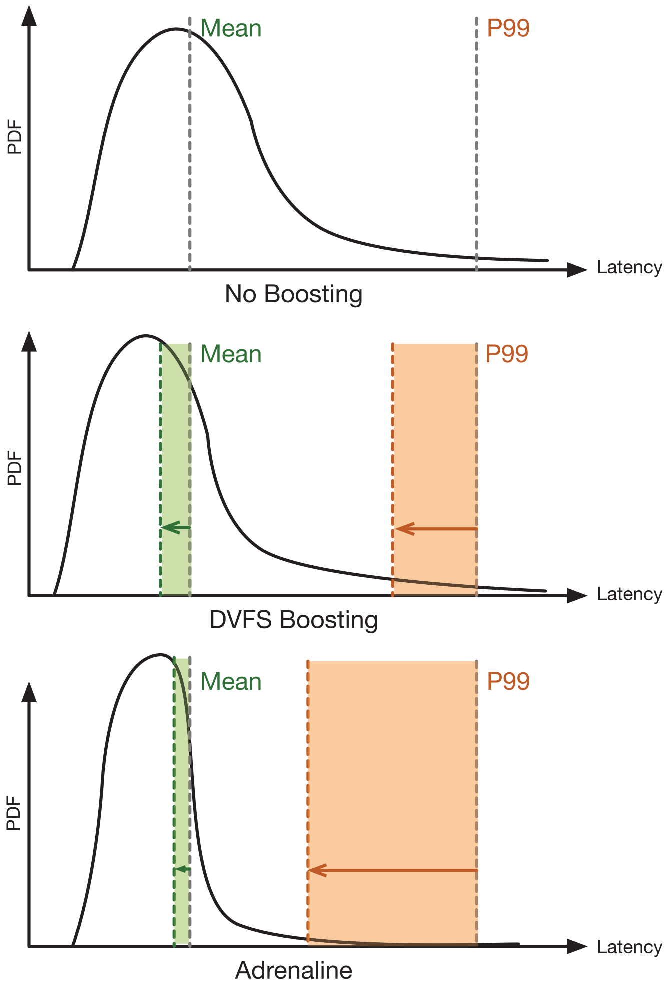

Fig. 1. Query latency distributions without boosting (top), with conventional DVFS boosting (center), and with Adrenaline (bottom).

(3) *Identifying an effective system design to pinpoint tail queries and precisely boost them.* To realize quick pinpointed boosting of tail queries, a mechanism must be in place to enable the identification and boosting of likely tail queries.

In this article, we investigate the potential of query-level quick boosting and design *Adrenaline*, a technique to address the limitations of traditional dynamic voltage and frequency scaling (DVFS) and achieve tail-sensitive query-level voltage boosting to significantly reduce the tail latency with high energy efficiency. Adrenaline leverages the recently proposed *Shortstop* [Pinckney et al. 2013], a circuit design for fine granularity voltage/frequency scaling to design the *Adrenaline Runtime Engine*, a software system approach for query-level quick boosting. The Adrenaline runtime engine effectively identifies queries that are likely to increase the tail distribution and can reap more benefit from the V/f boost. The runtime engine then adjusts the V/f at a query level accordingly. The various indicators Adrenaline relies on include query types (e.g., `SET`, GET, and DEL for Memcached) and query characteristics (e.g., number of extttWeb Search keywords).

In addition to reducing the tail latency when needed, Adrenaline's fine-grained query-level V/f boosting can significantly improve energy efficiency by scaling down voltage and frequency when tail latency is lower than the latency target. Adrenaline achieves energy efficiency improvements over coarse-grained V/f boosting by only lowering V/f for queries that are less likely to increase the tail or less affected by the V/f scaling.

This article makes the following contributions:

—**Identify Per-Query Indicators -** We characterize the query latency distributions for latency-sensitive datacenter applications, including Nutch Web Search and Memcached, on real systems and how the distributions are impacted by the core frequency scaling. We then identify application specific query-level indicators for pinpointing queries to apply appropriate V/f settings.

—**Adrenaline: Query-Level Voltage/Frequency Scaling -** We present the Adrenaline framework to achieve query-level fine-grained V/f scaling (tens of nanoseconds) and design heuristics based on the query-level indicators to scale the V/f based on per-query characteristics.

—**Rein in the Tail Latency -** Using Adrenaline, we demonstrate the effectiveness of applying query-level quick boosting for tail latency reduction. We achieve up to a 2.50× tail latency improvement for Memcached and up to a 3.03× for Web Search over coarse-grained DVFS given a fixed boosting power budget. With a carefully chosen boost threshold, Adrenaline can even achieve maximally 4.82× tail latency improvement for Memcached over coarse-grained DVFS.

—**Improving Energy Efficiency -** In addition to reining in tail latency, we demonstrate the effectiveness of applying Adrenaline to improve energy efficiency. We achieve up to a 1.81× improvement for Memcached and up to a 1.99× improvement for Web Search over DVFS.

2. MOTIVATION AND OPPORTUNITIES

We begin by examining the query latency distributions of two representative web services and exploring how frequency impacts query latency. A key insight underlying Adrenaline is that only a subset of queries need boosting to pull in the tail latency. Two key factors determine the subset that should be prioritized: (1) queries that fall in the tail distribution; (2) queries that can benefit more from the boosting. To understand how we might identify queries that exhibit these factors, we characterize the query distributions of Web Search and Memcached workloads on a state-of-the-art Intel Xeon server (described in Section 5).

Figure 2 presents the cumulative distributions of request latency for the three most common request types for Memcached (SET, GET, and DEL), collected at seven CPU frequency steps ranging from 1.2GHz to 2.4GHz on the Intel Xeon server. As shown in the figure, although all three query types have long tails, the latency distributions of these three types and how they are affected by frequency scaling vary. SETs' request latency is in general around 2× greater than GETs or DELs, indicating that the SET requests may contribute more to the tail latency, especially at low to medium load levels. In addition, higher frequency can significantly improve SETs' latency. Increasing the frequency from 1.2GHz to 2.4GHz improves SETs' 90th percentile latency from 13μs to 7μs. This indicates that to maximize the benefit of tail reduction using voltage/frequency boosting under a power budget, boosting SET requests should be prioritized.

In contrast to Memcached, Web Search (Nutch [Ferdman et al. 2012]) does not have multiple query types that can be directly used to classify queries. After investigating multiple query characteristics and the effectiveness of using those characteristics to

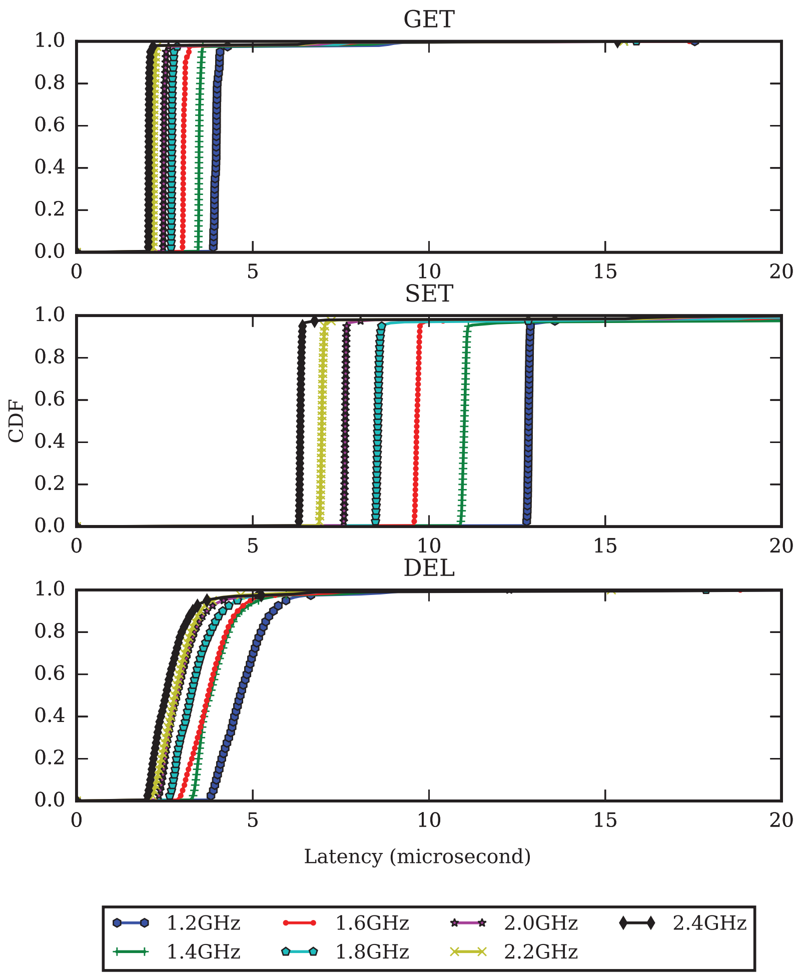

Fig. 2. The cumulative distributions of query latency in Memcached for `GET`, `SET`, and `DEL` requests.

predict the query latency distribution and the impact of frequency of scaling, we have identified that the query length (the number of search key words in a Web Search query) is a fairly effective indicator. Figure 3 presents the cumulative distributions of query latency for three different query lengths at seven frequencies, focusing on the tail part of the latency (beyond 85th percentile). As shown in the figure, short queries (queries with fewer, for example, 1-5, search key words) in general experience longer latency than long queries (queries with more search terms). Whereas it may seem counter-intuitive that adding terms reduces query latency, Nutch returns only documents that contain all search terms. Hence, additional terms reduce the number of documents that must be considered in scoring. In addition, the latency of short queries is much more improved when we increase the frequency compared to the medium and long queries. For example, the 95th percentile latency of queries with 1–5 keywords (short) is around 1100ms at 1.2GHz and lower than 700ms at 2.2GHz. On the other hand, the queries with 11–18 search keywords (long) are not affected by the frequency as much. This indicates that queries with fewer terms (short) should be prioritized for boosting to pull in the tail.

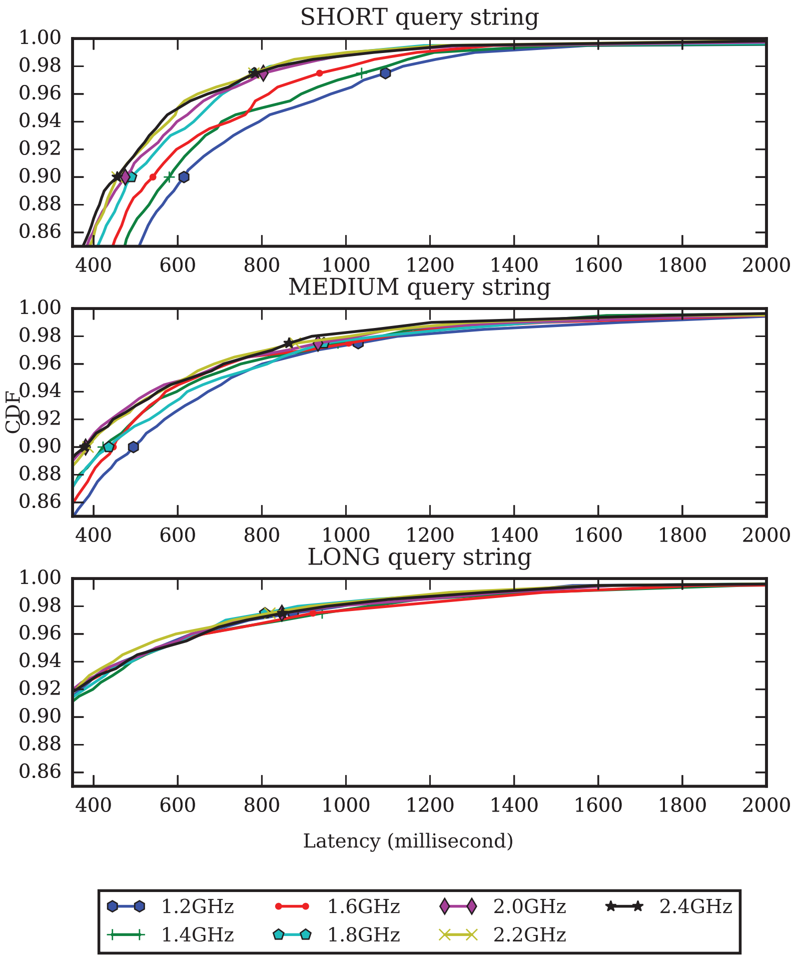

Fig. 3. The cumulative distributions of query latency in Web Search for queries with different numbers of search keywords.

In summary, Figures 2 and 3 demonstrate that per-query indicators (e.g., query types and query properties) can help pinpoint a subset of queries that contribute more to the tail and/or more impacted by the frequency and thus need to be prioritized to effectively pull in the tail. Based on this, we propose query-level boosting. Figure 4 illustrates the difference of our system, Adrenaline, compared with no boosting and traditional coarse-grained DVFS boosting. As illustrated in the figure, coarse-grained approaches apply V/f boosting to a sliding window, boosting all queries within a window. Adrenaline takes advantage of fast boosting (sub 20ns) and uses per-query indicators to pinpoint individual queries that should be boosted (for example, SETs as illustrated) to conduct query-level boosting only for these queries to achieve effective tail reduction with high energy efficiency.

3. QUICK V/F BOOSTING: AN ENABLING TECHNOLOGY

There are existing technologies that enable fast voltage transitions. Per-Core DVFS was a technique explored by Kim et al. [2008, 2011]. Per-Core DVFS integrates a voltage regulator on-die for each core in the system, allowing individual control and nanosecond

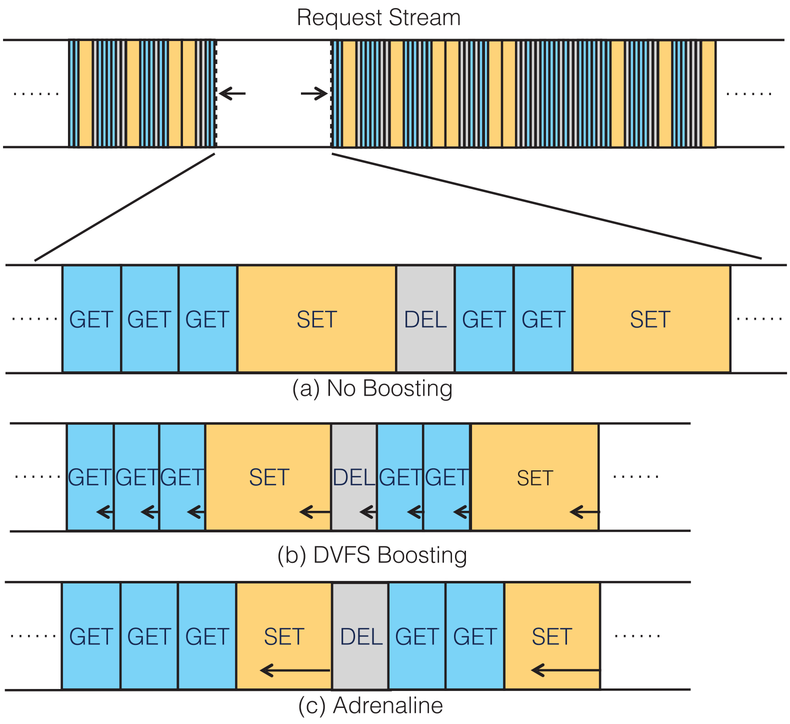

Fig. 4. Adrenaline's query-level V/f scaling vs. coarse-grained sliding window-based V/f scaling.

scale voltage transition times. The Per-Core DVFS technique comes at the expense of on-chip inductors and reduced regulator efficiency. Intel's TurboBoost [Corporation 2008] enables microsecond scale voltage transitions to allow, for example, the system to exceed thermal budgets for short periods of time. TurboBoost also includes an integrated Power Control Unit (PCU) that makes boosting decisions in hardware based on sensors and other hardware performance counters, eliminating the long latencies associated with operating system (OS) management. Recently, Godycki et al. have introduced Reconfigurable Power Distribution Networks [Godycki et al. 2014] as a method to improve voltage transition times with the use of a configurable on-chip switch-cap based voltage regulator. Finally, Shortstop [Pinckney et al. 2013] and Booster [Miller et al. 2012] use dual-rail voltage systems to enable fine-grained boosting. They use off-chip voltage regulation for better efficiency and to remove the need for on-die inductors.

For this work, we evaluate Shortstop as the voltage regulation methodology due to its short transition latency, lack of on-die inductive elements, and better regulator efficiency. However, the other approaches could be adapted as well to support Adrenaline. For example, the PCU of Turbo Boost could be augmented to accept query indicators to support the Adrenaline decision engine at the expense of a longer voltage transition latency.

3.1. Shortstop

Shortstop [Pinckney et al. 2013] is a dual-V_{dd} system with two distribution networks for V_{dd} supplied by two independent external voltage sources, V_{ddHigh} and V_{ddLow}, as shown in Figure 5. In addition, there is an internal V_{boost} supply to aid in the transition of a core from the low to the high supply. Each core in the system is connected via multiple power gating transistors (shown as a single transistor in the diagram) to the V_{ddHigh}, V_{ddLow},

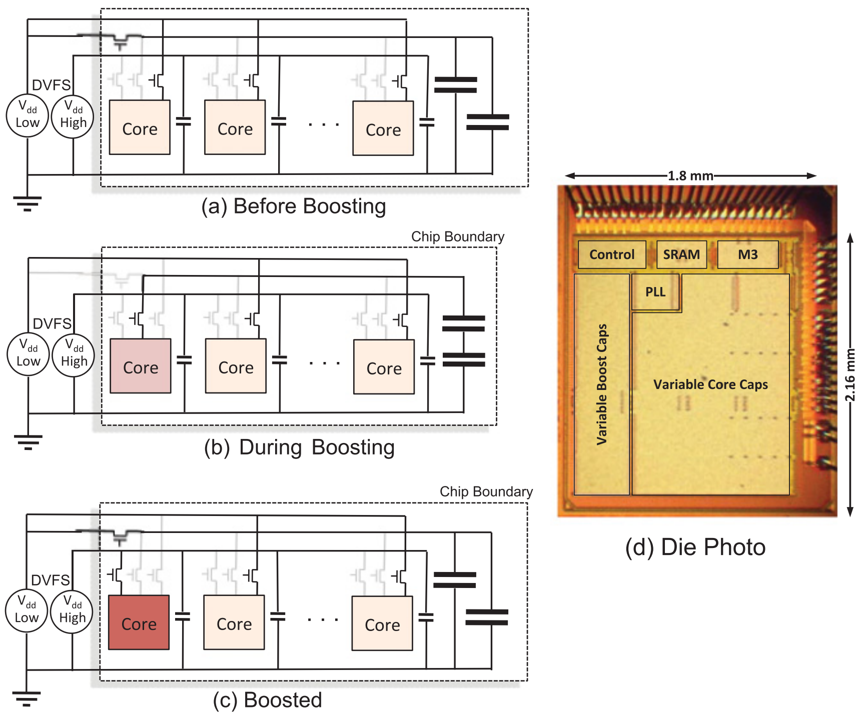

Fig. 5. Shortstop's Dual-V_{dd} chip configurations. Panel (a) shows normal operation, where all cores are connected to the low-voltage network and the cap networks are in parallel. Panel (b) shows the cores in boost transition where the core boosting is connected to the output of the serially connected cap network. Panel (c) shows the system once the transition stabilizes where the cap network returns to parallel, and the boosted core runs off the external high voltage. Panel (d) shows the die photo of the 28nm Shortstop test chip [Pinckney et al. 2013].

or V_{boost} voltage rails. Decoupling capacitors (decaps) are placed between the high supply network and the ground node to reduce ripples on the node during transitions. In addition, the V_{boost} supply has a set of reconfigurable decoupling capacitors to aid in transitioning the core quickly. This system keeps the voltage regulators off-chip, removing the TDP overheads required for on-chip conversion.

Using Shortstop has several advantages. First, since the boosting decaps are on chip, they can act quickly and, through charge re-distribution, provide for a rapid transition. Second, since the boosting decaps are shared by all the processors, their area overhead is amortized over all the cores. In addition, while the boosting network does require the distribution of a third supply rail (V_{boost}), this rail does not need to have a high level of signal integrity, meaning it can be more sparse. The overall overhead of adding a boosting rail with reconfigurable decaps is 11%. When considering advanced technologies, such as deep trench capacitors [Wang et al. 2009], this overhead can be reduced to less than 5%. Third, since the boosting network brings the voltage of the transitioning core to nearly V_{ddHigh}, the voltage droop on V_{ddHigh} is not nearly as large as when no boosting network is used. Extra decaps on the high supply further suppress the droop to a level that is acceptable. The area overhead of the V_{ddHigh} rail is just 5%. So the overall area overhead of Shortstop is $\sim$10–16%. Finally, Shortstop can be used to perform both fine- and coarse-grained voltage control. Through boosting, Shortstop provides nanosecond-scale voltage transitions to adjust to fine-grained changes in behavior

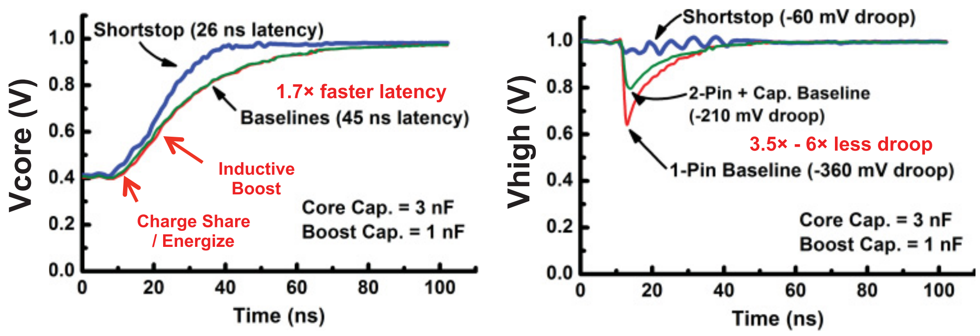

Fig. 6. Measured chip results [Pinckney et al. 2013] of Shortstop compared to a standard dual-rail baseline.

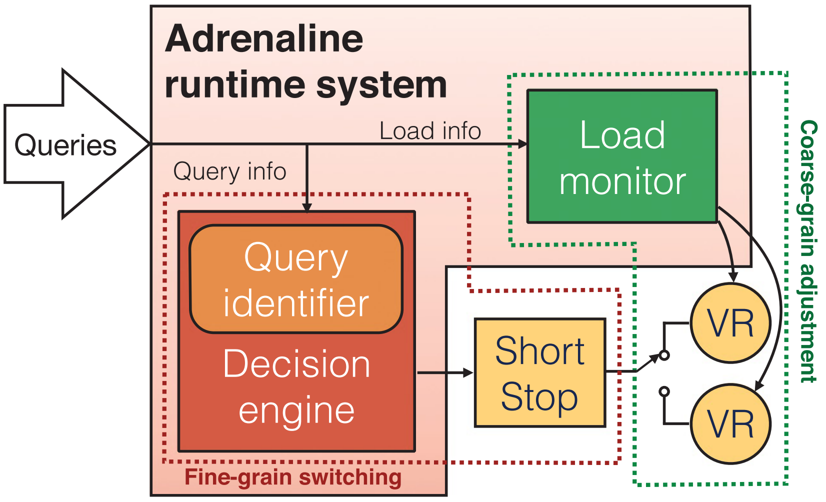

Fig. 7. The Adrenaline framework. Adrenaline makes fine-grained boost decisions based on the characteristics of each incoming query and controls the Shortstop circuit to switch between the high/low voltage rails (VRs) quickly. Meanwhile, Adrenaline also monitors long-term loading information and makes proper adjustments to the voltage on the high/low VRs periodically.

(query level) and can adapt the off-chip voltage regulators over longer periods of time to adjust to coarse-grained changes in workload behavior (request rate level).

Shortstop was fabricated and tested in a 28nm technology by Pinckney et al. [2013], proving the feasibility of the approach. Figure 6 provides a summary of the measured data on transition times of that chip. The baseline for comparison was a standard dual-rail approach. Shortstop provides nanosecond-scale transitions that are 1.7× faster with a 3.5–6× smaller voltage droop. The original article also provides results showing the scalability of the approach across different core sizes/capacitances.

4. ADRENALINE FRAMEWORK

This section describes the design of the Adrenaline runtime system, which takes advantage of the fast V/f switching capability of the Shortstop circuit to rein in tail latency by boosting precisely those queries that contribute to the tail.

Adrenaline Overview. An overview of the Adrenaline runtime system is presented in Figure 7. Adrenaline is composed of two components, a decision engine and a load monitor. First, the decision engine chooses from between the low or high V/f settings at the individual query level to boost the speed of precisely those queries that contribute to tail latency, based on the query characteristics analyzed by the equipped query identifier, reducing the latency of those queries to shorten the tail of the query latency distribution. This is in contrast to conventional V/f scaling techniques that monitor the query load and distribution over long timescales (usually several seconds) and adjust the V/f settings for large clusters of queries.

Second, Adrenaline's load monitor performs lightweight accounting on incoming queries to measure the load to the system (e.g., to measure queries per second), which in turn is used to drive a coarse-grained tuning policy that changes the supply voltage parameters to the off-chip voltage regulators. This gives Adrenaline the capacity to adapt the Shortstop circuit to address longer-term (milliseconds or longer) changes in the characteristics of the query profile, such as fluctuations in user demand (queries per second) or the type and composition of queries.

4.1. Decision Engine

The purpose of the decision engine is to rapidly identify the queries that should be boosted to have the largest impact on reducing tail latency. First, queries that are amenable to boosting, because they are compute bound but are not in the tail, can have a significant impact on tail latency via the indirect mechanism of reducing queuing delay for queries that are in the tail of the latency distribution. Second, queries that are themselves in the tail of the distribution have a direct impact on tail latency. We therefore design the decision engine to use the following two guiding principles to determine which queries to boost:

(1) **Boostability –** Only those queries that are amenable to boosting should be boosted. Power spent boosting queries that do not speed up is wasted. Furthermore, queries that are more amenable to boosting confer a larger benefit to subsequent queries that may be headed for the tail of the distribution.
(2) **Long-running Queries –** As long as they are boostable, queries that are destined to be in the tail of the latency distribution have the most direct impact on tail latency and therefore should be boosted.

These characteristics may be difficult to directly reason about, particularly at the short timescale necessary to boost a query quickly after arriving at the host server. Fortunately, the presence of one or both of these characteristics are often indicated by other easily identifiable proximate characteristics. Such characteristics include the following:

—**Query Type –** Different query types, which are easily identifiable as part of the query metadata, often have very different latency profiles. For example, Figure 2 shows that Memcached `GET` queries have around one-third of the latency of `SET` queries.
—**Query Composition –** The same type of query may have other easily identifiable characteristics that correlate well with query latency. For example, Figure 3 shows that Web Search has very different latency profiles for queries with different numbers of words.

4.2. Defining the Boost Policy

Query type and query composition are used to develop policies that determine when the Shortstop circuit changes rails, lowering and raising the supply voltage at the query level to bring down the latency of critical queries. These policies are developed offline using a short, targeted profiling phase on the application to characterize the relationship between query latency and the high-level characteristics of the query.

Adrenaline's boost policies are guided by how different types of requests can affect the tail the most under different V/f settings. For each application, we first analyze the behavior of requests across varying characteristics, load levels, and chip V/f settings. The characteristics chosen for the analysis are particular to the application under study and are chosen such that they are characteristics that can be rapidly identified (e.g., by checking a few bits in the header or the size of the application query packet). For Memcached, we characterize requests across the different query types: `GET`, `SET`, and `DEL`. For Web Search, we characterize across a series of lengths, which correlates well to the size of the query packet and the latency of the query. For Memcached, we find that `SET` queries are highly amenable to boosting the bulk of the high-latency queries. For Web Search, we find that queries with fewer than six keywords are similarly amenable to boosting and have high latency.

Adrenaline also watches the condition in which a query tends to fall in the tail of the latency distribution, even if that query does not belong to the types mentioned above. Adrenaline's policy is to keep track of the time a query has spent in the system and check if it exceeds a predefined **boost threshold**. When the threshold is reached, the query is considered *long running* and has a high probability to end up falling in the tail; hence, Adrenaline will make boost decisions for this query. We found by experiment that setting the threshold to $0.5\times$ of the quality of service (QoS) target of the benchmark gives us good results.

4.3. Rapidly Identifying Query Characteristics

Adrenaline is implemented as a runtime engine in the first layer of software that processes incoming request packets to take advantage of features of advanced network interface controllers and low-latency networking stacks, such as OS-bypass, zero-copy, and direct cache access. These features facilitate extremely low-latency packet delivery to the Adrenaline runtime engine, which can then rapidly examine packet headers to make a per-query boost decision; prototype systems have demonstrated packet delivery latencies below 1.5μs, and commercial offerings from vendors like Mellanox have similar performance characteristics [Gordon et al. 2012]. In addition, by fixing the header lengths of the link, network, and transport layers, even shorter times can be achieved. In the case of Memcached, the request type field can be attained by inspecting the corresponding bits in the application layer of a binary encoded Memcached packet. For the Web Search workloads, a table of sequence numbers and corresponding packet sizes can be used to determine the total request length, which is used as a proxy for the size of the search.

4.4. Boosting and Unboosting the Core

Once a query type is identified, the Adrenaline runtime decision engine makes a boosting decision based on the profile characteristics obtained from the analysis of the request behavior. If a decision to boost a core not already boosted is made, then the runtime signals the Shortstop hardware to insert the core into a boosting queue. Because Shortstop only allows one core to transition at a time, the queue is serviced in a first-in, first-out fashion. Note the worst-case time spent in the queue will be short, as the transition times for Shortstop are on the order of tens of nanoseconds and the number of cores on a chip is relatively small. If a decision to boost a core is made while the core is already boosted, then the Adrenaline runtime tracks the additional request. Once all the requests finish, the Adrenaline runtime signals Shortstop to unboost the core.

4.5. Clock Distribution

Clock distribution is another key consideration in this design. Our proposed solution is to distribute a chipwide clock at the high frequency but at low voltage. At each node, the clock is divided down to the required frequency before entering each core. In a typical system, the majority of the clock power is consumed at the bottom of the tree (i.e., flip flops), meaning that running the clock globally at high frequency has minimal impact on the total power. In boost mode, the local clock tree remains at low voltage and is level converted to V_{ddHigh} only in the last driving stage. With this approach, clock tree synchronizing mechanisms such as delay-locked loops would not need to re-lock during boosting transitions, since the voltage and latency of the clock tree does not change.

4.6. Adrenaline for Energy Efficiency

In addition to reducing the tail latency, when needed, Adrenaline can be configured to scale down the voltage and frequency at the query level when the tail latency is much lower than the latency target specified in the service level agreement (SLA). Similarly to improving the tail latency, this mechanism works by leveraging query characteristics to identify those queries that are unlikely to have a large impact on tail latency and throttle the voltage during such queries. Adrenaline is flexible enough to adjust policies dynamically based on high-level characteristics of user load, such as using different latency targets or optimization targets for different levels of load or mixes of query types. Along with the tail reduction approach, we evaluate using Adrenaline to target energy efficiency in Section 5.

4.7. Responding to Load Changes

The Shortstop circuit can be configured to use different supply voltages on its two voltage rails with a penalty on the order of tens of microseconds. To take advantage of this feature, we employ a load monitor in Adrenaline to monitor the query traffic over time and use this to switch between supply voltage configurations every few seconds to provide maximum benefit to the current query traffic pattern. Note that the fundamental activity of the decision engine is not affected by this tuning, as it continues to make decisions about which queries should be boosted. Since this tuning is done in a much coarser granularity, although it imposes a short pause for changing the voltage on the two voltage rails, it has little impact on the overall distribution of query latency.

5. EVALUATION

In this section, we evaluate the effectiveness of Adrenaline in both reining the tail latency as well as saving energy. We conduct our evaluations at single-server level, as well as cluster level for a cluster composed of thousands of servers.

5.1. Evaluation Methodology

To accurately evaluate Adrenaline, we use the BigHouse simulator [Meisner et al. 2012], which is a datacenter simulation infrastructure that takes workload characteristics to synthesize an event trace to drive a discrete event simulation. We implement Adrenaline and a conventional DVFS baseline in BigHouse. We evaluate both using various configurations.

5.1.1. Real System Performance Characterization. In the BigHouse simulator, two distributions are used to represent a workload: the service distribution and the inter-arrival distribution. We collect these distributions from real system measurements. We use Memcached and Web Search from Cloudsuite [Ferdman et al. 2012] as our workloads.

The service time distribution describes how fast the server can process each individual request without counting the queueing latency. To obtain this distribution, we first instrument the Web Search and Memcached server-side software to record the time at the beginning and the end of processing to measure the service time distribution. Then

Table I. Request Type Composition of Memcached

Composition	GET%	SET%	DEL%
APP	83.8	4.7	11.5
ETC	68.6	2.7	28.7
VAR	18.3	81.7	0.0

Table II. Request Type Composition of Web Search

Composition	SHORT%	MEDIUM%	LONG%
SHR	53.1	28.7	18.2
LNG	12.7	34.4	52.9
ORG	92.0	5.5	2.5
UNI	34.0	33.0	33.0

we send requests to the instrumented server one at a time so no queueing will occur within the server and collect the latency statistics as the service time distribution.

The machine we use for our service-time distribution measurement features an Intel Xeon CPU E5-2407 v2 @ 2.40GHz CPU, which has two sockets with four cores per sockets. The size of the main memory is 136GB. The kernel of the underlying operating system is Linux 3.11.0-15-generic. We carefully deploy the server side of the benchmarks and the client-side counterpart in a controlled environment to minimize noise due to resource racing and the varying communication over the network, and so on. We collect the service-time distribution at each of the frequency steps of the core.

Many workloads have multiple request types rather than a single type. As prior work suggests [Atikoglu et al. 2012], there is a big variability in terms of request type composition in production systems and it has a significant impact on the overall performance. Thus we capture this by using the same request type composition as reported in prior work [Atikoglu et al. 2012] for Memcached and classifying the requests according to length for Web Search due to the lack of published characterization. Specifically, APP, ETC, and VAR refer to the same compositions as described in prior work [Atikoglu et al. 2012] for Memcached. For Web Search, ORG refers to the composition hard-coded in the Web Search client loader, and SHR, LNG, and UNI refers to synthetic compositions of requests that are short-query-heavy, long-query-heavy, and uniform, respectively. The detailed breakdown of each composition is listed in Table I and Table II.

We collect the service time distribution for each type of request separately and evaluate our system under various composition configurations.

Differing from service time distribution, inter-arrival distribution heavily depends on the specific configuration and workload. Thus we use the characteristics published by large companies that run these services in production. For Web Search, we follow the guideline from prior work [Meisner et al. 2011], which suggests to use an exponential distribution as an approximation according to the empirical measurement from the production Google Web Search server node. We use a Generalized Pareto distribution for Memcached as suggested in prior work [Atikoglu et al. 2012], which characterizes the production Memcached traffic at Facebook.

5.1.2. Power Modeling. The power equation (Equation (1)) we are using is adapted based on previous work [Meisner et al. 2009]. We change the exponent term in the equation from 3.0 to 2.7 to consider the effect of imperfect components such as the overhead of the global clock distribution network discussed in Section 4.5. The CPU power consumption is described in the following formula:

$$\begin{aligned} P_0 &= P_{sta,0} + P_{dyn,0} \\ P_{sta,0} &= 0.2 \times P_{dyn,0} \\ P_{dyn} &= P_{dyn,0} \times (f/f_0)^{2.7} \\ P_{sta} &= P_{sta,0} \times (f/f_0), \end{aligned} \tag{1}$$

where P_0 is the CPU power consumption measured on a real machine, $P_{sta,0}$ is the static part of P_0 and $P_{dyn,0}$ is the dynamic counterpart, and f_0 is the lowest frequency step at which the machine can run. P and f are the power and the frequency at which the CPU is currently operating.

We instrumented BigHouse to simulate the Adrenaline framework by leveraging multiple service-time distributions in BigHouse on the fly. In addition, we model the boosting and decision latencies as well as the requirement to only transition one core from low to high voltage at a time. On starting to service a request, our instrumented BigHouse picks the corresponding service-time distribution based on the type of request and the voltage level (always the low-voltage mode at the beginning of a request) and sets the instantaneous power number to be the number that is corresponding to that voltage level. When BigHouse reaches the point of boosting a request, our instrumented BigHouse will calculate the remaining amount of work of a request plus the switching overhead and use the boosted version of the service-time distribution file to determine the processing time for the rest of the work; the energy consumption of the elapsed period of time will be summed up, and the instantaneous power will be updated to the new (boosted) version at the same time. For example, if BigHouse decides to boost from 1.2GHz to 1.8GHz at 50% of a request, it calculates the energy used for the first half of the request at the 1.2GHz power number and uses the 1.8GHz service time to process the second half of the request; on the request completion, it calculates the energy consumption of the second half of the request using the 1.8GHz power number and switches back to using the 1.2GHz service-time distribution for the next incoming request.

5.2. Reining in the Tail

In this section, we evaluate the effectiveness of Adrenaline and the conventional coarse-grained sliding window-based DVFS in optimizing the tail latency. Starting from the low frequency as the baseline (e.g., core configuration at bottom right as illustrated in Figure 8), which consumes the lowest energy while generating the highest latency, we gradually increase our energy budget and measure the tail latency reduction at both 95% and 99%.

Memcached - Figure 9 shows the tail latency reduction for Memcached achieved by both coarse-grained DVFS and Adrenaline. In this figure, each row corresponds to a load level (low, medium, and high load), and each column corresponds to an energy budget for boosting (low, medium, and high budget). The low, medium, and high energy budgets are 10%, 20%, and 30% of energy increase from the baseline energy, respectively. In each sub-figure, we present the tail reductions achieved by both coarse-grained DVFS (the left cluster of bars) and Adrenaline (the right cluster of bars). Each bar in a cluster represents three real-world workload compositions described in Table I and the latency reductions for each composition at two percentiles, 95% and 99%. As shown in the first column, given a low energy budget (e.g., 10% energy increase), the coarse-grained DVFS fails to reduce the tail latency, while Adrenaline is able to improve the 95% latency by up to 2.20× and the 99% latency by up to 2.25×. This is due to the fact that the 10% energy budget is too small for the coarse-grained DVFS to boost to the next frequency step, while Adrenaline can take advantage of the fine-granularity nature and boost a small percent of the requests at the tail without consuming much energy. As we increase the energy budgets to 20% and 30%, coarse-grained DVFS starts

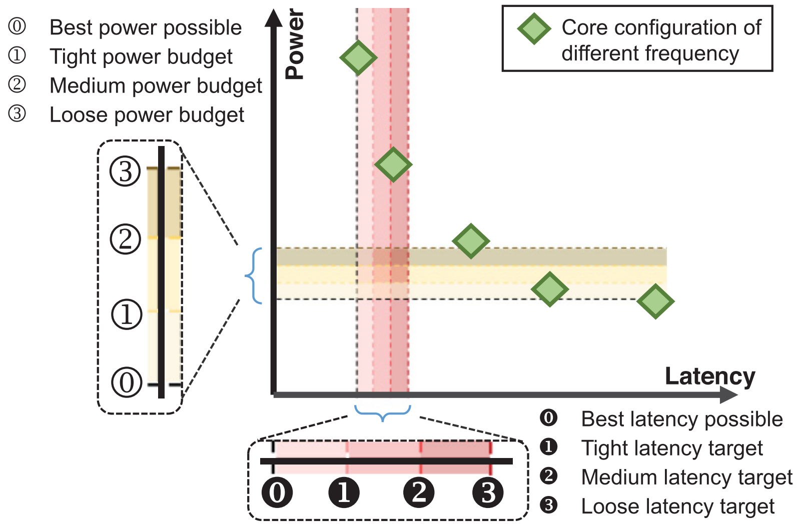

Fig. 8. Selection of Tight/Medium/Loose power budget and Tight/Medium/Loose latency target for our evaluation.

to show limited improvement on the tail latency. It slightly outperforms Adrenaline for workload `VAR`, because `GET` and `DEL` requests are usually surrounded by groups of `SET` requests in `VAR`. Adrenaline still tries to react to these short-term changes by first switching the core to a lower frequency when the core starts to process these `GET` and `DEL` requests. However, since `GET` and `DEL` requests now have high probabilities to queue behind one or more `SET` requests, they experience long waiting time, and thus become highly likely to be at the tail. Therefore, while Adrenaline does not rein in these `GET`s and `DEL`s promptly, coarse-grained DVFS luckily benefits from its inertia of switching by running at the higher frequency for the entire epoch of requests. However, its improvement on the tail latency is rather limited for the other workload compositions. In general, Figure 9 shows that, across various request compositions, load levels, and energy budgets, Adrenaline almost always improves the tail latency by a larger amount than the coarse-grained DVFS.

Web Search - Similarly, Figure 10 presents the tail latency reduction for Web Search at various load levels and across four request composition configurations (specified in Table II). As demonstrated in the sub-figures, Adrenaline shows an edge over coarse-grained DVFS across almost all different workload compositions and energy budgets, especially when the system is at medium and high load levels (the last two rows) where tail reduction is critical. The advantage becomes even more significant when the system is given a higher energy budget. Adrenaline achieves up to a 3.03× tail reduction when given a 20% budget. Given a 30% energy budget, while coarse-grained DVFS starts to take advantage of the large energy head room and show comparable tail latency reduction for `ORG`, Adrenaline still achieves better reduction for the same composition and consistently outperforms coarse-grained DVFS across all other workload compositions.

Latency Distribution - To better understand the impact of coarse-grained DVFS and Adrenaline on the overall latency distribution, Figure 11 compares the distributions achieved by both approaches under the same power budget for boosting. In this figure, the request type composition is `APP` (Table I), composed of 4.7% `SET`, 11.5% `DEL`, and 83.8% `GET` requests. This figure presents two probability density functions (PDFs) of the measured request latency of all requests: one measured when using coarse-grained DVFS (red) and the other one using Adrenaline (blue). The dotted lines indicate the

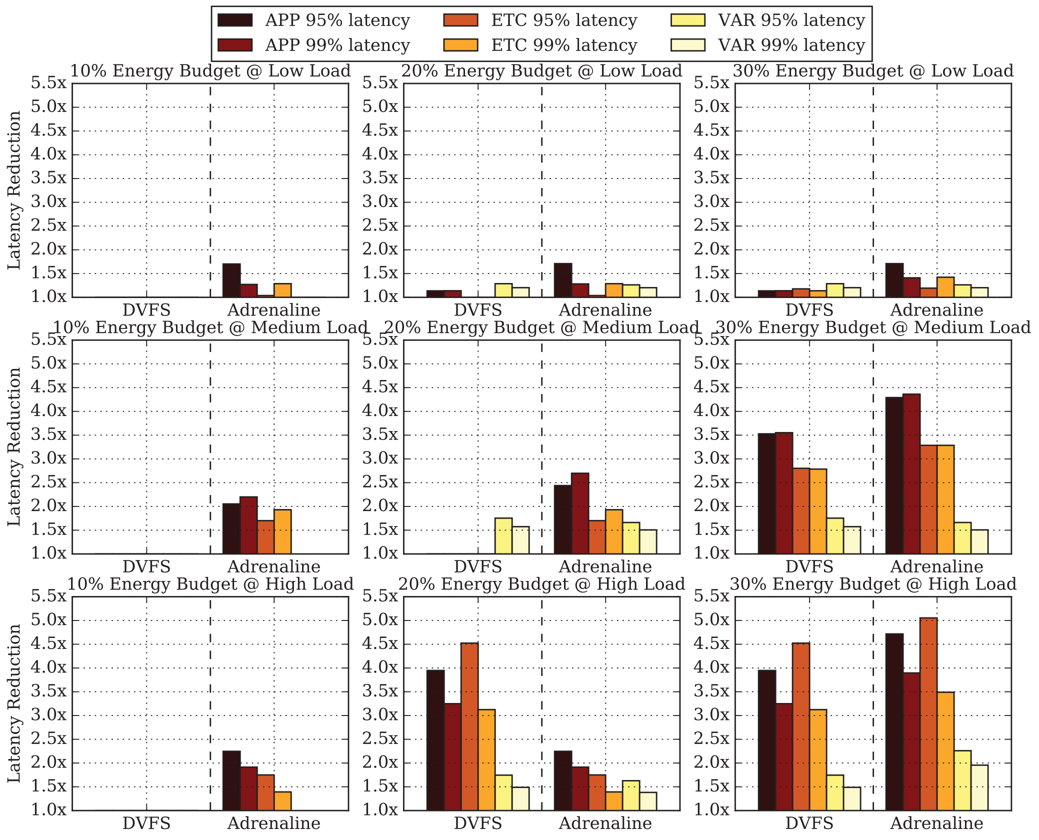

Fig. 9. Tail latency reduction for Memcached using coarse-grained DVFS vs. Adrenaline. The row presents three load levels, and the column represents three energy budgets for boosting. Adrenaline achieves much higher tail latency reduction at various load levels, across workload compositions and energy budgets. (Rows 1 to 3: low, medium, and high load. Columns 1 to 3: 10%, 20%, and 30% energy budget.)

95% and 99% latency of each distribution. For the distributions, both coarse-grained DVFS and Adrenaline show that GET and DEL requests are tightly clustered on the left, whereas SET requests have a much higher latency and compose the long tail.

Compared to the no-boosting scenario, the coarse-grained DVFS shifts the entire distribution to the left, reducing both the mean and 99% latency by a small amount. However, Adrenaline spends the limited power budget for boosting in a more effective way. Instead of boosting all requests, including both the fast requests and the tail requests, it only boosts the requests that have high probability to be in the tail. The mean latency Adrenaline achieves is slightly slower than that achieved by the coarse-grained DVFS, but it is still faster than the no-boosting scenario. More importantly, Adrenaline achieves a much more significant reduction on the tail latency than the coarse-grained DVFS.

Figure 12 shows a Pareto-optimal curve for the power versus 99% latency tradeoff of different V/f configurations for both DVFS and Adrenaline. At extremely tight SLA targets, the DVFS approach is necessary, but as soon as the latency target is loosened, the Adrenaline approach offers either significant reductions in tail latency (further to the left at a given power budget) or a gain in power efficiency (lower in the graph for a given target latency).

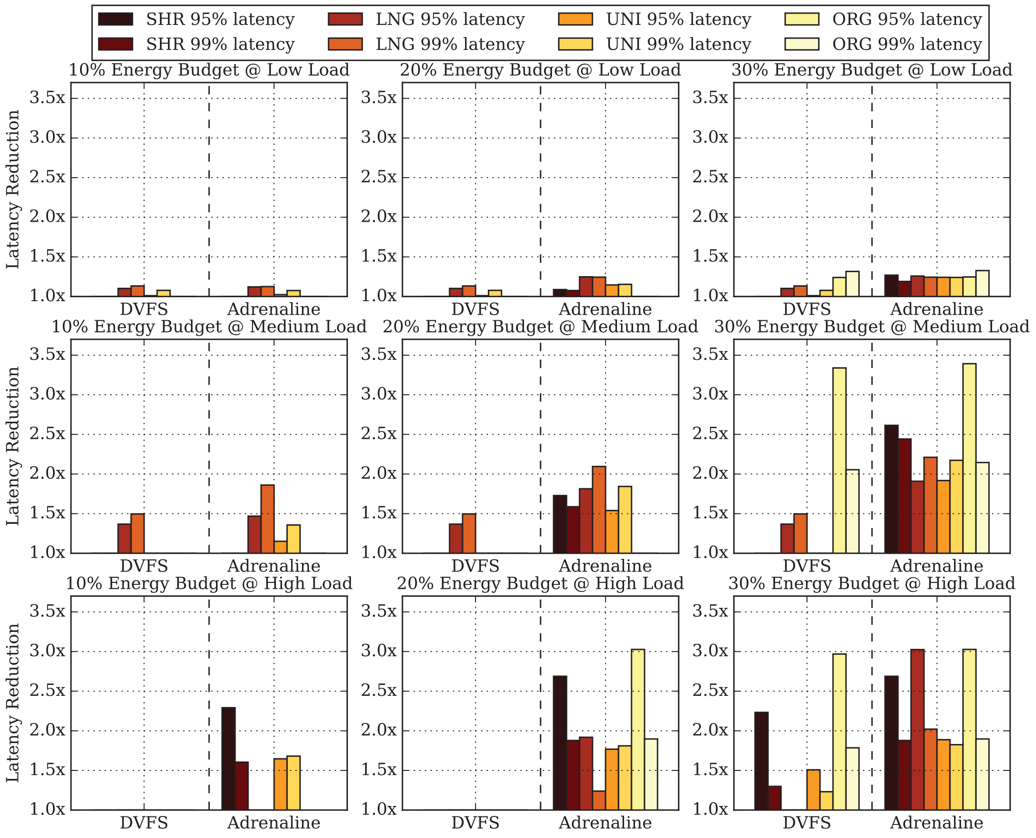

Fig. 10. Tail latency reduction for Web Search by relaxing energy budget at various load levels using Adrenaline and DVFS. (Rows 1 to 3: low, medium, and high load. Columns 1 to 3: 10%, 20%, and 30% energy budget.)

5.3. Energy Saving

In addition to reining the tail latency by fine-grained boosting, Adrenaline can also be used to take advantage of the latency slack, especially at low load, to improve energy efficiency. As illustrated in Figure 8, we start at the highest CPU frequency, which generates the lowest possible latency, and evaluate the effectiveness of coarse-grained DVFS and Adrenaline in improving the energy efficiency as we gradually relax the tail latency target. Specifically, we choose 10% latency lack as the strict target, 20% as the moderate target, and 30% as the relaxed target in our experiments.

Figure 13 presents the energy savings for Memcached achieved by coarse-grained DVFS and Adrenaline, respectively. As demonstrated in the figure, given the same latency target, Adrenaline can significantly reduce the energy consumption, especially at low load. It achieves up to a 2.12× energy savings over the no-DVFS baseline, whereas DVFS can only save 1.56× in the best scenario. At high load, both Adrenaline and coarse-grained DVFS cannot achieve much energy savings. This is to be expected, because at the high load, there is not much latency slack due to the queuing delay. It is very hard to achieve energy savings unless we are willing to sacrifice a great amount of tail latency slack. Similarly, Figure 14 shows that Adrenaline can achieve significant energy savings across all three different load levels, especially at low and medium load levels, for Web Search. In addition, Adrenaline often achieves greater energy

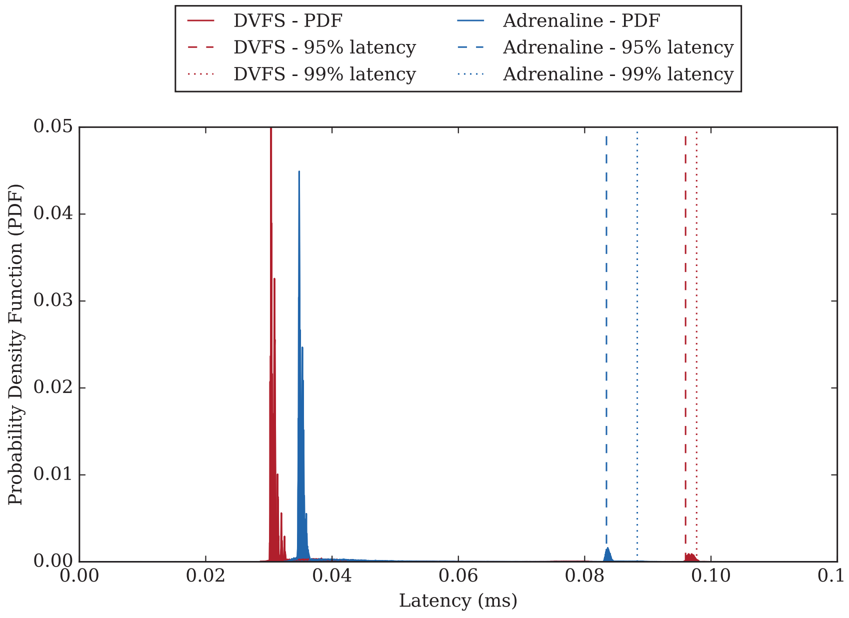

Fig. 11. Latency probability density function of Memcached when applying coarse-grained DVFS and Adrenaline, respectively, to the same stream of queries.

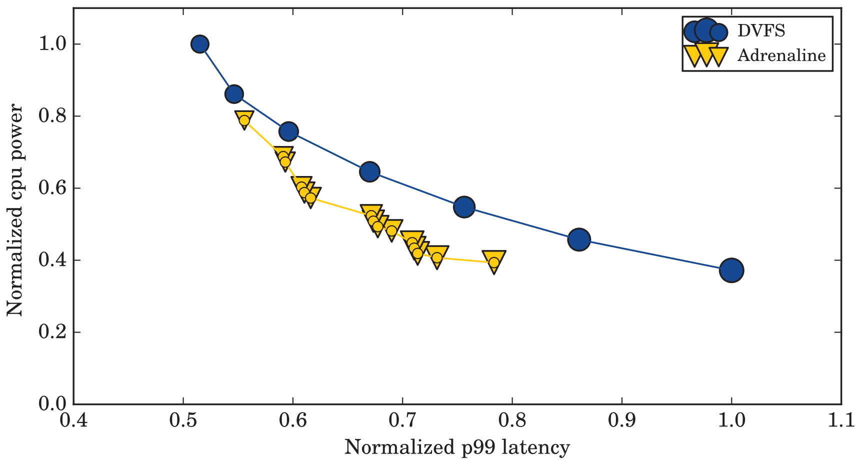

Fig. 12. The scatter plot of Memcached, in which each data point represents the normalized power/latency performance of a single CPU V/f configuration with load composition APP and low traffic.

savings compared to coarse-grained DVFS. This is because Adrenaline leverages the observation demonstrated in Figures 2 and 3 and prioritizes the boosting of those requests which give the most benefits, which effectively prevents Adrenaline from wasting energy on those requests with low boostability.

5.4. Overall Comparison

Figures 15 and 16 present Adrenaline's improvement of tail latency and energy savings over DVFS, averaged across all workload compositions. Overall, Adrenaline outperforms coarse-grained DVFS in almost all the situations. When optimizing for tail

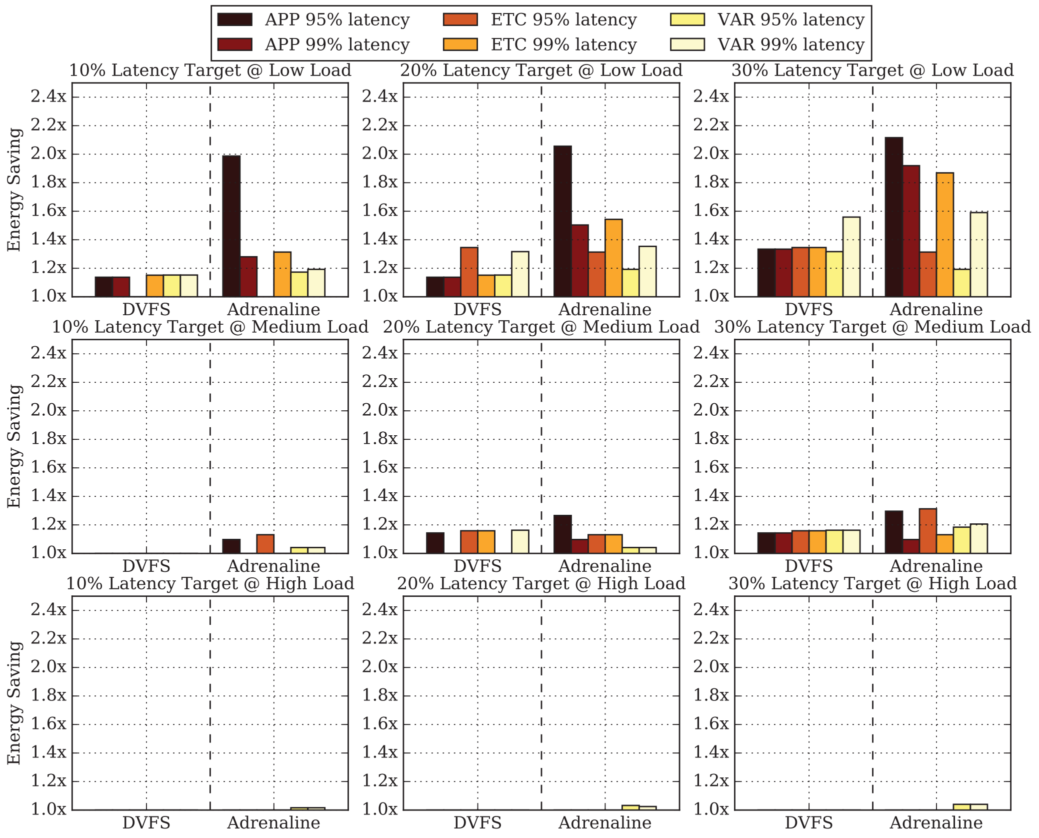

Fig. 13. Energy saving for Memcached by relaxing the tail latency target at various load levels using Adrenaline and coarse-grained DVFS. (Rows 1 to 3: low, medium, and high load. Columns 1 to 3: 10%, 20%, and 30% latency budget.)

latency, Adrenaline achieves up to 22.7% improvement over DVFS by dynamically scaling the frequency at a finer granularity to target only the most important requests.

5.5. The Impact of Boost Threshold

We analyzed the time non-candidate queries spent in the system and use a **boost threshold** to predict long-running queries that should be boosted. This boost threshold is the product of the QoS target and a coefficient, as shown in Equation (2) as follows:

$$T_{threshold} = c_t \times T_{QoS}. \tag{2}$$

For example, for the previous studies in Section 5.2 and Section 5.3, our choice of boost threshold was 0.5× of the QoS target. This choice translates into a threshold coefficient c_t of 0.5.

While our results shown in the previous sections show that we achieve significant improvement in both tail latency reduction and energy saving with a straightforward selection of $c_t = 0.5$, we observe that the choice of c_t can have significant impacts on performance and efficiency of Adrenaline for benchmarks with different characteristics. To understand the effects, we evaluate Adrenaline under a range of threshold coefficient settings and study their impact on the reduction of the 95th percentile latency given certain energy consumption constraints. Note that the coefficient c_t determines only

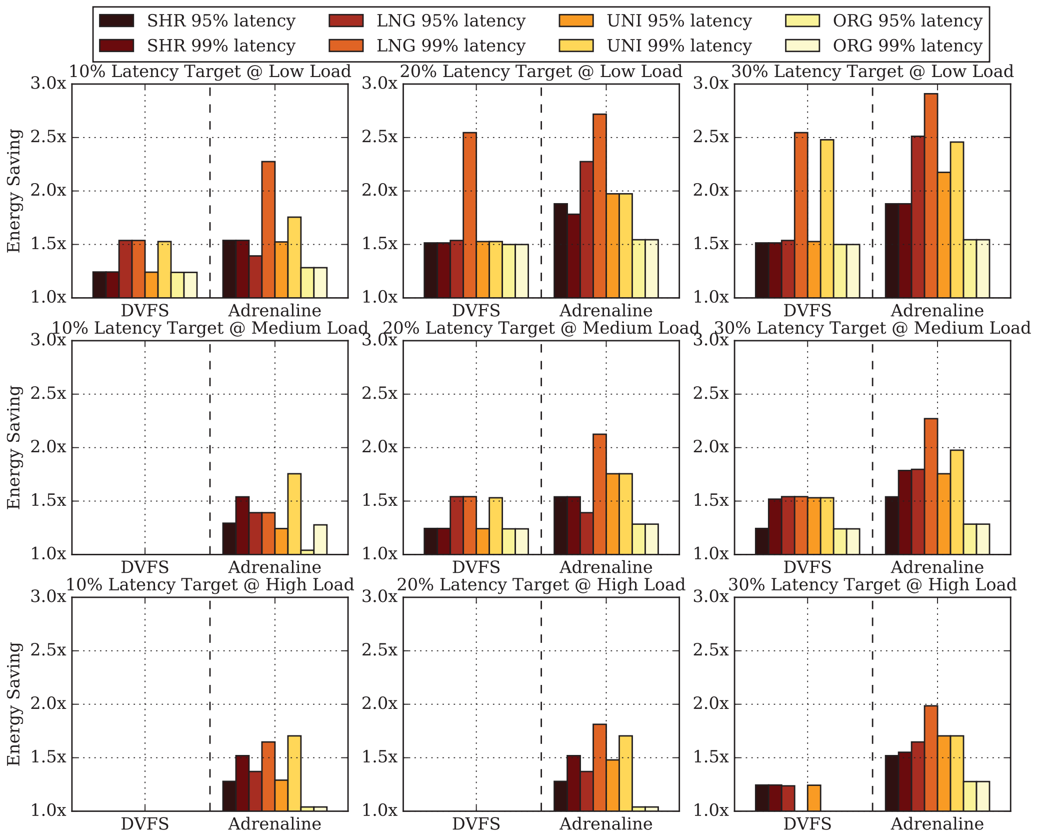

Fig. 14. Energy saving for Web Search by relaxing the tail latency target at various load levels using Adrenaline and DVFS. (Rows 1 to 3: low, medium, and high load. Columns 1 to 3: 10%, 20%, and 30% latency budget.)

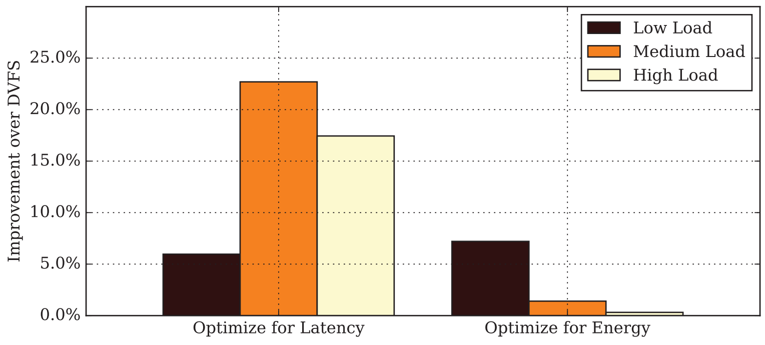

Fig. 15. Improvement of Adrenaline over coarse-grained DVFS for Memcached.

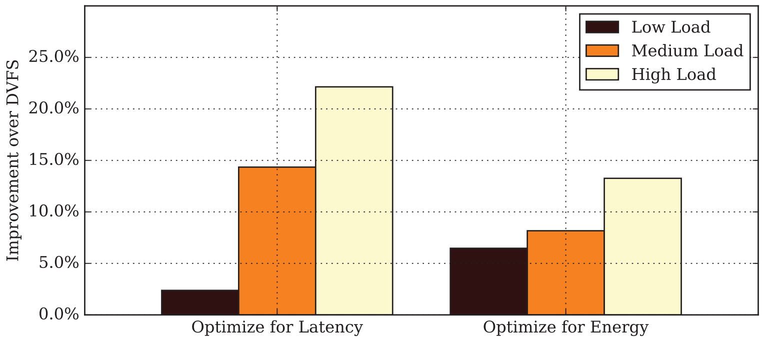

Fig. 16. Improvement of Adrenaline over coarse-grained DVFS for Web Search.

the boost threshold for non-candidate types of queries (i.e., `GET` and `DEL` for Memcached and `MEDIUM` and `LONG` for Web Search), whereas the candidate types of queries (i.e., `SET` for Memcached and `SHORT` for Web Search) will always be boosted at the onset of core processing.

5.5.1. A General Principle. The goal of Adrenaline is to rein in the tail queries to meet the QoS target. To accomplish this, it is desirable for the boost threshold to be smaller than the QoS target. This allows tail queries to be identified before they miss the target. But one challenge here is that rather than ignoring those queries that are fast enough, when the threshold is too small, Adrenaline boosts most of the queries. This failure to consider query characteristics leads to energy waste.

5.5.2. Memcached vs. Threshold Coefficient. To study the impact of boost threshold on the performance of Memcached, we sweep c_t within the range of 0.0001 to 0.5, which translates to a range from 1μs to 5ms for the boost threshold $T_{threshold}$. Figure 17 shows the effect of choosing different c_t on tail latency reduction for Memcached, given certain energy budgets. We prefetch and conclude our major findings as follows:

—**Finding 1:** Adrenaline's performance is sensitive to the choice of c_t. The performance variation, with the same workload, in tail latency reduction and energy saving can be as large as 5$\times$ and 1.78$\times$, respectively.
—**Finding 2:** Adrenaline configured with optimal c_t can extract up to 4.82$\times$ improvement over coarse-grained DVFS in the reduction of the 95th-percentile latency, even under the tightest energy budget.
—**Finding 3:** The optimal choice of c_t is within the range of 0.0005 and 0.001. This optimal choice range remains almost constant across different loads and energy/latency budgets.

Choosing an optimal c_t that minimizes the tail latency while remaining within the energy budget can be challenging. Too low a threshold wastes considerable amounts of energy to boost queries that are natively fast enough. Too high a threshold fails to effectively reduce query latency due to the limited percentage of the execution that is boosted.

To explain the importance of a proper threshold, we refer to both Figures 17 and 18. Figure 17 shows Adrenaline's impact on tail latency reduction with different choices of c_t across different load levels and energy budgets. Figure 18 gives the percentage of boosted non-candidate queries. When c_t is large, Adrenaline boosts very few or no non-candidate queries, generating little positive effect on tail latency. As we decrease c_t, Adrenaline boosts an increasing amount of non-candidate queries and therefore yields better performance. When $c_t = 0.0005$ and $c_t = 0.001$, Adrenaline achieves the

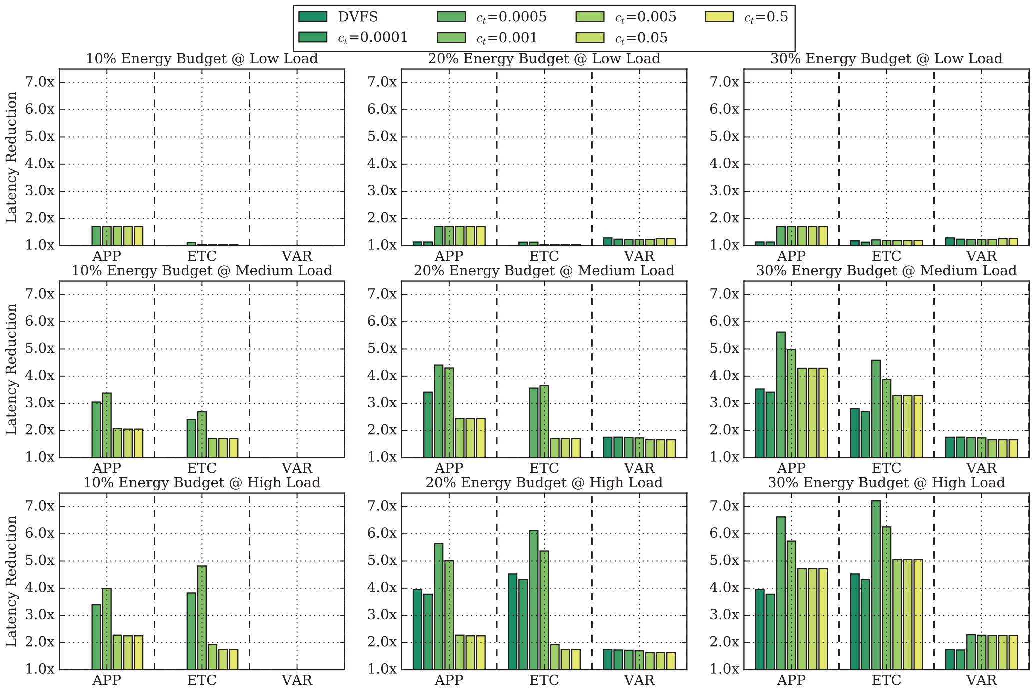

Fig. 17. Sensitivity analysis of the effect of different boost thresholds on tail latency reduction of Memcached. Rows 1 to 3 are the results of sensitivity analysis with low, medium, and high load, respectively. Columns 1 to 3 are the results with 10%, 20%, and 30% energy budget, respectively.

optimal improvement across different load levels and energy budgets. As can be seen in Figure 18, Adrenaline properly boosts a large portion, but not all, of the queries. This indicates that the small threshold coefficients allow Adrenaline to boost queries early enough and therefore reduce the probability of forming a long queue, resulting in shorter queueing delays for queries in the queue.

This outcome is highly correlated to the characteristics of different request types of Memcached shown in Figure 2. While the service times of all types of Memcached queries are generally short and the coefficients of variation are small, there is a clear distinction between the latency distribution of the candidate type (`SET`) and the distributions of the two non-candidate types (`GET` and `DEL`). If we set c_t to 0.0005 or 0.001, then Adrenaline can save energy by using the lower voltage to serve the queries when the queue is short and is composed of mostly non-candidate queries; when the queueing delay becomes critical, Adrenaline can agilely enter the boost mode. If we further lower c_t to 0.0001, then the benefit on tail latency reduction disappears. We can conclude from Figure 18 that Adrenaline with $c_t = 0.0001$ boosts almost 100% of the queries. Under this condition, the energy overhead incurred by running the majority queries with high-performance configuration is simply too high for the energy budget to afford. The choices of supply voltage on the two voltage rails in Shortstop are, therefore, restricted to the lower ones, which limits the speedup Adrenaline can achieve.

Figure 18 reveals some cases where the result of $c_t = 0.0001$ is missing. For example, the results of $c_t = 0.0001$ of all the three query compositions with 10% energy budget at low load, of `APP` and `ETC` with 10% energy budget at medium load, and of ETC with 20% budget at medium load are absent. The absence of these results indicates that the low threshold makes Adrenaline boost queries too early, largely increasing the overall

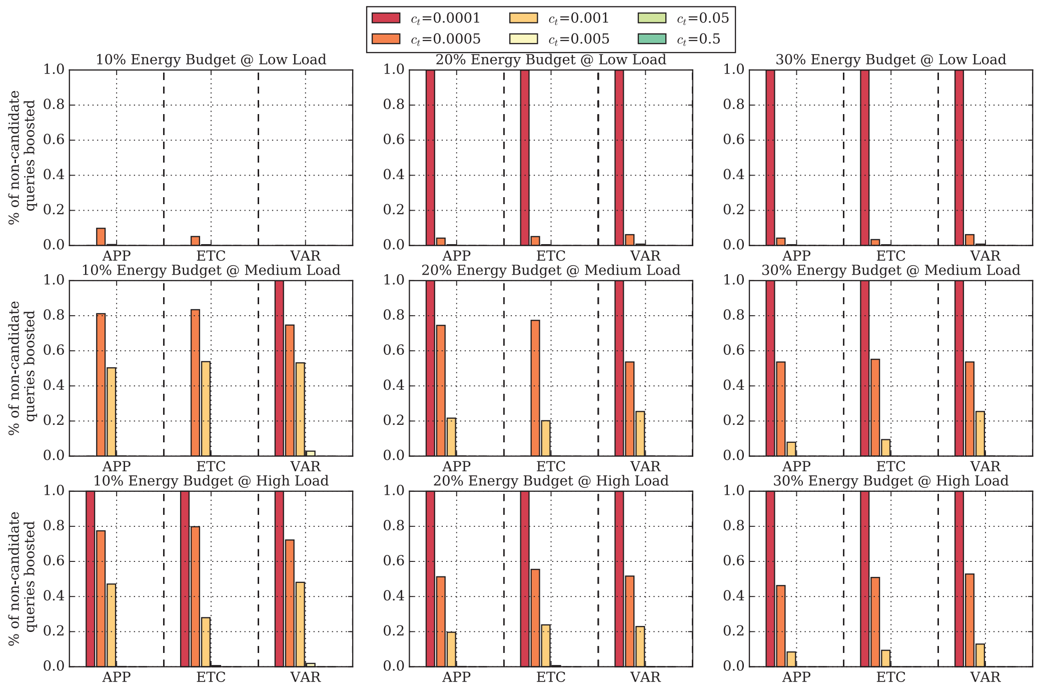

Fig. 18. Percentage of non-candidate Memcached queries (GETs and DELs) boosted when optimizing for tail latency. (Rows 1 to 3: low, medium, and high load. Columns 1 to 3: 10%, 20%, and 30% energy budget.)

energy consumption, making Adrenaline violate the given energy budget no matter which CPU V/f configuration is used.

Figures 19 and 20 show boost threshold's impact on energy saving. We see from these two figures that when the system is under medium and high load, there is a trend similar to what we have seen in the previous experiments. However, one counter-intuitive finding for workload ETC at low load is that, as we decrease c_t from 0.001 to 0.0005 and Adrenaline is given a 20% latency target, the energy saving becomes more significant even though the percentage of boosted non-candidate queries increases. This is due to the fact that Adrenaline is using a different of supply voltages for the Shortstop voltage rails. When $c_t = 0.001$, because the threshold is high and the workload ETC contains only a very small percentage of SET queries, only a few queries get boosted; under this condition, Adrenaline is not able to meet the 20% tail latency budget without using the highest V/f setting for its boost mode. But once we decrease c_t to 0.0005, Adrenaline enters boost mode much earlier, which mitigates the queueing effect; a V/f configuration with a lower boost-mode frequency (i.e., the frequency used when Shortstop is supplied by its high-voltage rail) is now sufficient for Adrenaline to meet the tail latency target.

We also see from the two figures that larger energy budgets sometimes lead to fewer boostings compared to smaller energy budgets at the same load. This is because the larger energy budget gives Adrenaline the flexibility to use a more aggressive V/f configuration, which enables a higher normal-mode frequency (i.e., the frequency used when Shortstop is supplied by its low-voltage rail). This higher normal-mode frequency benefits all of the queries in the incoming query stream and shortens their latency, leading to fewer needs for boostings.

5.5.3. Web Search vs. Threshold Coefficient. We conduct a similar sensitivity analysis for Web Search. For Web Search, we sweep the threshold coefficient c_t from 0.001 to 0.5,

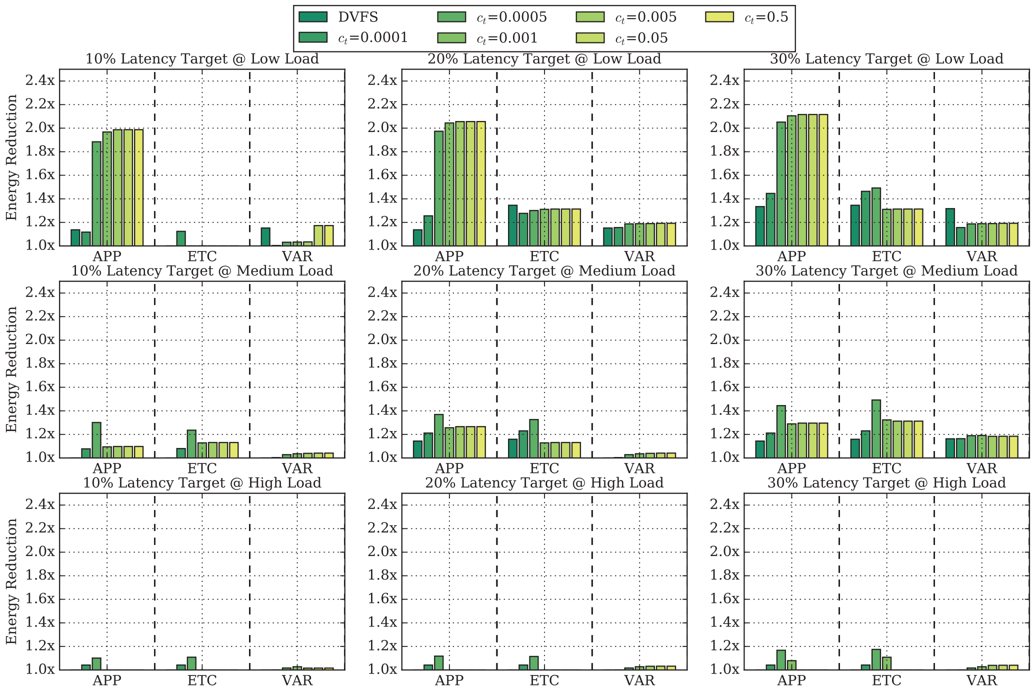

Fig. 19. Sensitivity analysis of the effect of different boost thresholds on energy reduction of Memcached. (Rows 1 to 3: low, medium, and high load. Columns 1 to 3: 10%, 20%, and 30% latency budget.)

which translates to a range from 50μs to 250ms for the boost threshold $T_{threshold}$. We again prefetch our findings and present them as follows:

—**Finding 1:** For Web Search, Adrenaline's impact on latency reduction and energy saving is also sensitive to the choice of c_t. For a same workload, the performance variation across different choices of c_t can be up to 3.17× in tail latency reduction and up to 1.62× in energy saving.

—**Finding 2:** The optimal choice of c_t for Web Search is within the range of 0.1 to 0.5, which is much larger than that for Memcached. This distinction results from the different characteristics the Web Search queries have. However, the optimal choice still remains almost constant across different loads and different energy/latency budgets.

As can be seen in Figure 21, Adrenaline, again, provides significant improvement over coarse-grained DVFS for Web Search at medium load and high load. When we study Adrenaline's effectiveness in tail latency reduction across the different choices of c_t, however, we find that the system behavior drastically differs from what we have seen in the Memcached experiments. As shown in Figure 21, the larger c_t's such as 0.1 and 0.5 now have a better chance to provide significant tail latency reduction over coarse-grained DVFS. The optimal c_t for Web Search is higher compared to Memcached, because Web Search service queries usually need a much longer time to process, as shown in Figure 3. The non-candidate Web Search queries, and especially the `LONG` queries, are very insensitive to the change of the frequency boosting. This means that when Adrenaline enters the boost mode too early when processing non-candidate queries, it wastes significant amounts of energy but only yields a small latency reduction. So c_t should be set to a higher value so the cores stay at the normal mode and boost only

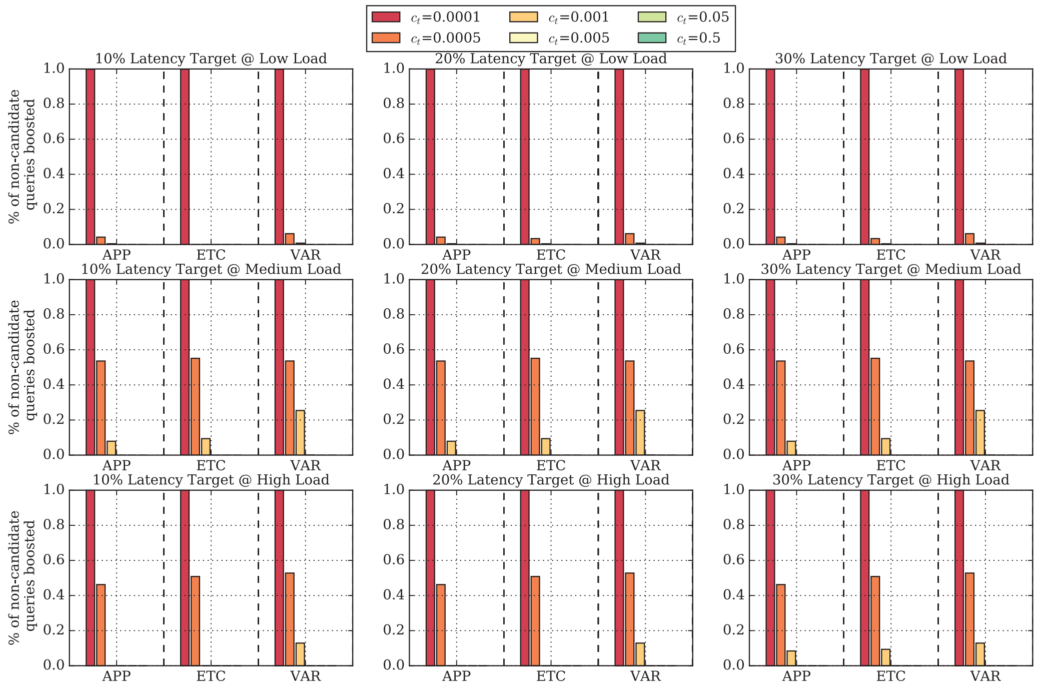

Fig. 20. Percentage of non-candidate Memcached queries (GETs and DELs) boosted when optimizing for energy saving. (Rows 1 to 3: low, medium, and high load. Columns 1 to 3: 10%, 20%, and 30% latency budget.)

when urgent conditions occur. Similar findings are made when we use Adrenaline to optimize for energy saving. Figures 23 and 24 show that, under the same load level and the same tail latency target, Adrenaline achieves more significant energy saving when using larger c_t's.

Seeing this significant improvement, one might find it surprising, in Figure 22, that the boost percentages of higher c_t's are low. The reason why such low boost percentages can lead to significant tail latency reduction is that Adrenaline is using higher voltage values for Shortstop's high-voltage rail. With these higher boost-mode voltages, Adrenaline is able to use higher boost-mode frequencies to effectively shorten the latency of candidate queries and some of the long-running non-candidate queries, which compensates for the low boost percentages. Due to the low boost percentage, even though Adrenaline uses a higher boost-mode voltages, the energy budget constraints are not easily violated.

5.5.4. Strategy for Choosing the Right Boost Thresholds and V/f Configurations. In this section, we discuss the strategy for choosing the right boost thresholds and voltage configurations at runtime when deploying Adrenaline for an application.

During runtime, Adrenaline makes two types of decisions for different purposes. The first type of decision is the per-query boost decision. This type of decision, as we discussed in the previous sections of the article, aims to determine whether to boost processor performance for each query. Since user-facing applications have strict latency requirement and have an enormous amount of user queries hitting the servers every second, the fine-grained decision engine needs to be low overhead. Any significant extra latency added by this engine builds up non-negligible delays and slows down all the queries drastically in a loaded system. Adrenaline achieves low overhead by looking for simple query-level indicators to make fast decisions that switch the underlying

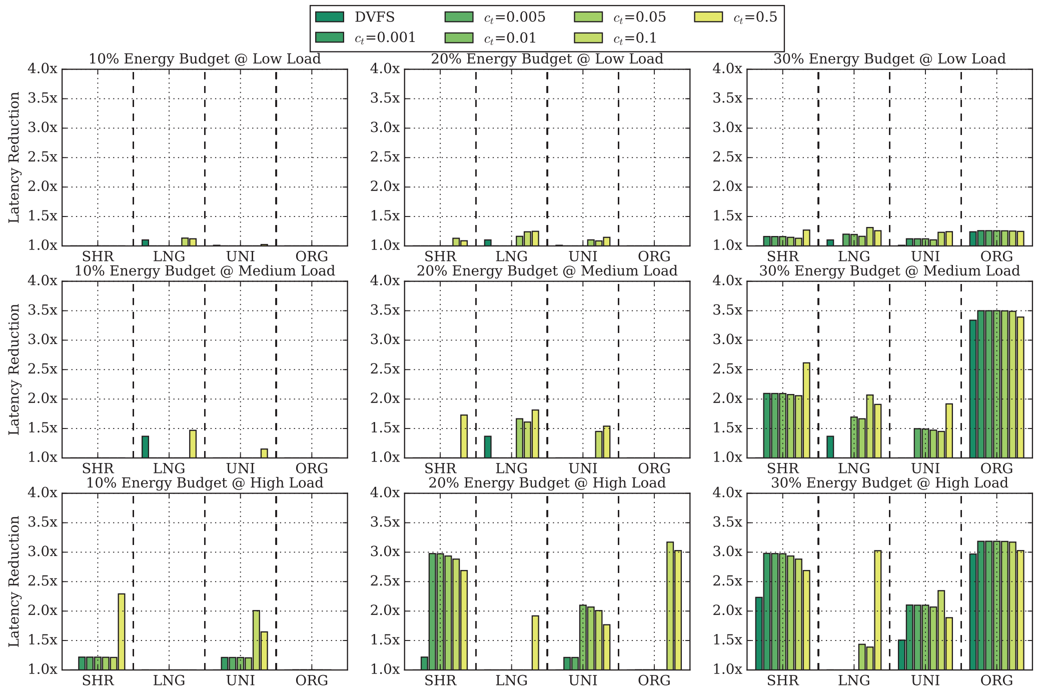

Fig. 21. Sensitivity analysis of the effect of different boost thresholds on tail latency reduction of Web Search. (Rows 1 to 3: low, medium, and high load. Columns 1 to 3: 10%, 20%, and 30% energy budget.)

Shortstop circuit between its high/low voltage rails, incurring only nano-second-level delays.

The second type of decisions reacts to the change of traffic level and query composition, determining the voltage configuration of the underlying Shortstop circuit and the boost threshold used for accelerating non-candidate queries. The decision engine needs to collect enough information over a sliding window before it can explore a suitable next configuration to adapt to. In addition, changing the voltage configuration incurs significant overhead; instead of switching quickly between the high- and low-voltage rails, executing this type of decision involves adapting the voltages on the two voltage rails via an off-chip regulator. The delay of such an operation is within the range of tens of microseconds [Eyerman and Eeckhout 2011], easily comparable or even exceeding the average service time of shorter queries, such as `GET`, `SET`, and `DEL` queries in Memcached. This type of decisions, therefore, should only be triggered at a coarser time granularity.

However, it is challenging to develop a generalized decision-making runtime algorithm suitable for multiple applications. As shown in Section 5.5, the optimal boost threshold and the optimal V/f configuration of Adrenaline vary significantly under different situations. Since the decision highly depends on the characteristics of queries of the target application, as well as the current traffic level and query composition, the construction of such a generalized algorithm might involve designing complicated control-theoretic mechanisms, which is out of the scope of this article.

In this work, we advice to construct such a decision engine on top of profiling results of the target application, which can be done as discussed in Section 5.5.2 and Section 5.5.3. We provide some guideline as follows.

Strategy for choosing boost threshold. From the results shown in the previous section, we observe that boost threshold is highly related to the target application's

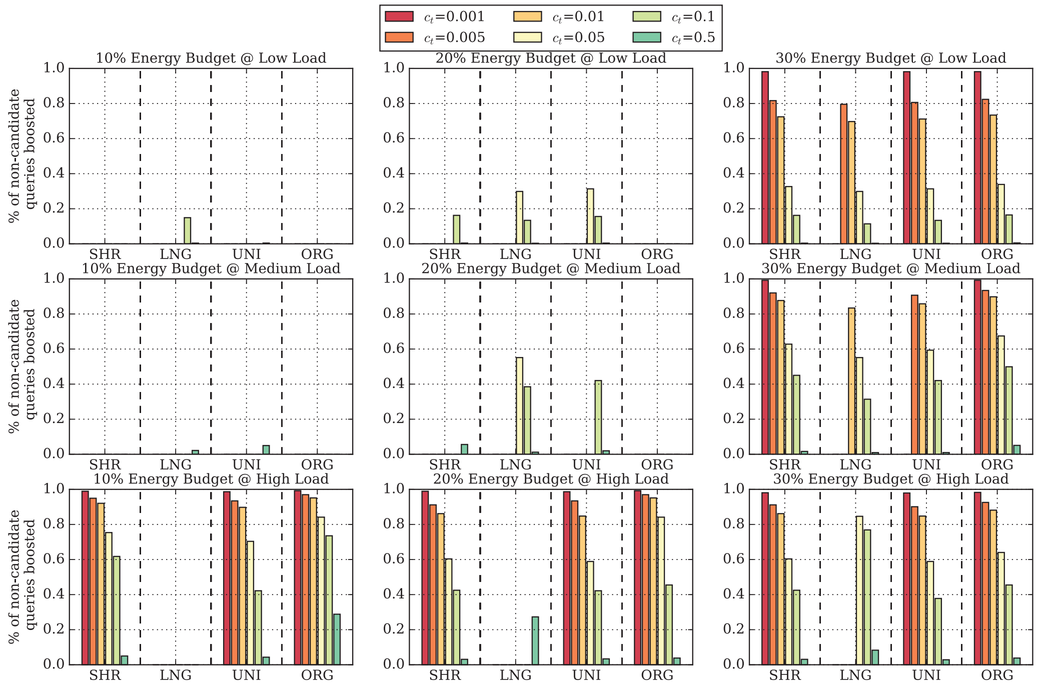

Fig. 22. Percentage of non-candidate Web Search queries (MEDIUMs and LONGs) boosted when optimizing for tail latency. (Rows 1 to 3: low, medium, and high load. Columns 1 to 3: 10%, 20%, and 30% energy budget.)

typical service time, typical query arrival rate (in terms of query-per-second, QPS), and non-candidate queries' responsiveness to core frequency. Boost threshold is rather insensitive to the change in query composition and traffic level. As shown in Figures 17 and 23, the optimal threshold coefficients for an application is almost always the same under different combinations of query compositions and traffic levels. For Memcached, $c_t = 0.0005$ or $c_t = 0.001$ is always the choice for threshold coefficient. For Web Search, $c_t = 0.5$ or $c_t = 0.1$ is almost always superior than other c_t's; in the cases where neither of them is the best, their performance still comes close to the optimal ones. Such a characteristic greatly simplifies the strategy for choosing a suitable boost threshold at runtime. For example, for applications similar to Memcached, which feature short query service time, high QPS, and non-candidate queries that are highly core-frequency-responsive, boost decisions should be made early enough to prevent long queues from building up. In this situation, smaller c_t's should therefore be prioritized over larger c_t's. On the other hand, applications similar to Web Search have long query service time, much lower QPS, and unresponsive non-candidate queries. Adrenaline takes advantage of this characteristic and uses a larger threshold that boosts non-candidate queries later, gaining significant energy saving without building up a long queue.

Strategy for choosing V/f configuration. After we conduct an off-line profiling on the target application, for each combination of traffic level and query composition, we identify the most beneficial V/f configuration and record this information in a lookup table. During runtime, we use a sliding window to monitor the most recent history of workload. According to the workload observed in the window, the decision engine chooses the next configuration by searching for the V/f configuration associated with the closest combination of traffic level and query composition in the lookup table.

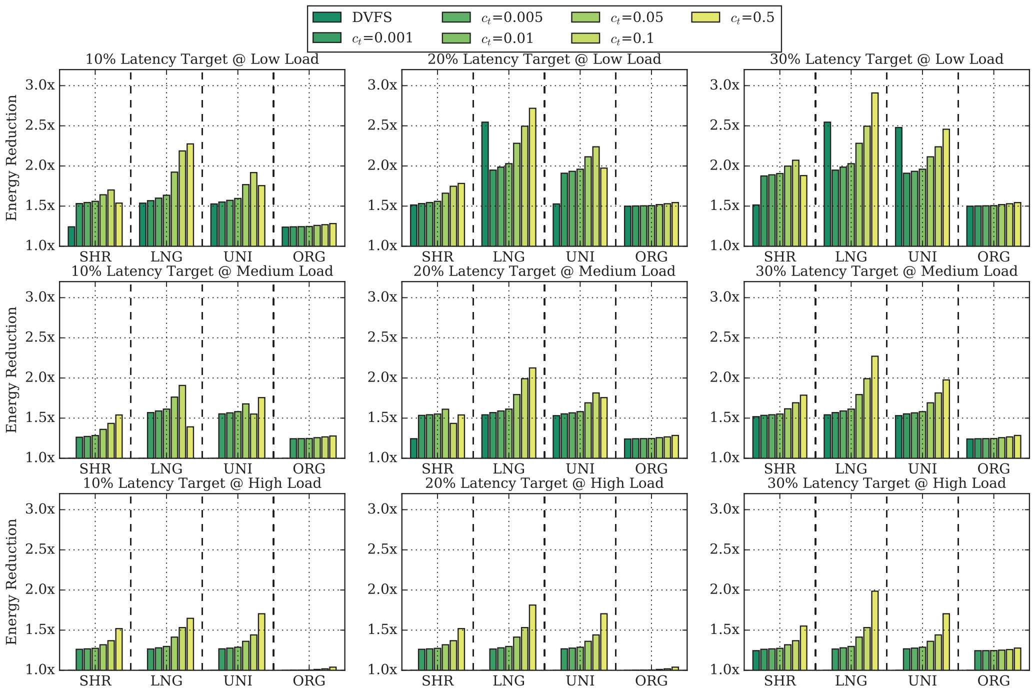

Fig. 23. Sensitivity analysis of the effect of different boost thresholds on energy saving of Web Search. (Rows 1 to 3 : low, medium, and high load. Columns 1 to 3: 10%, 20%, and 30% latency budget.)

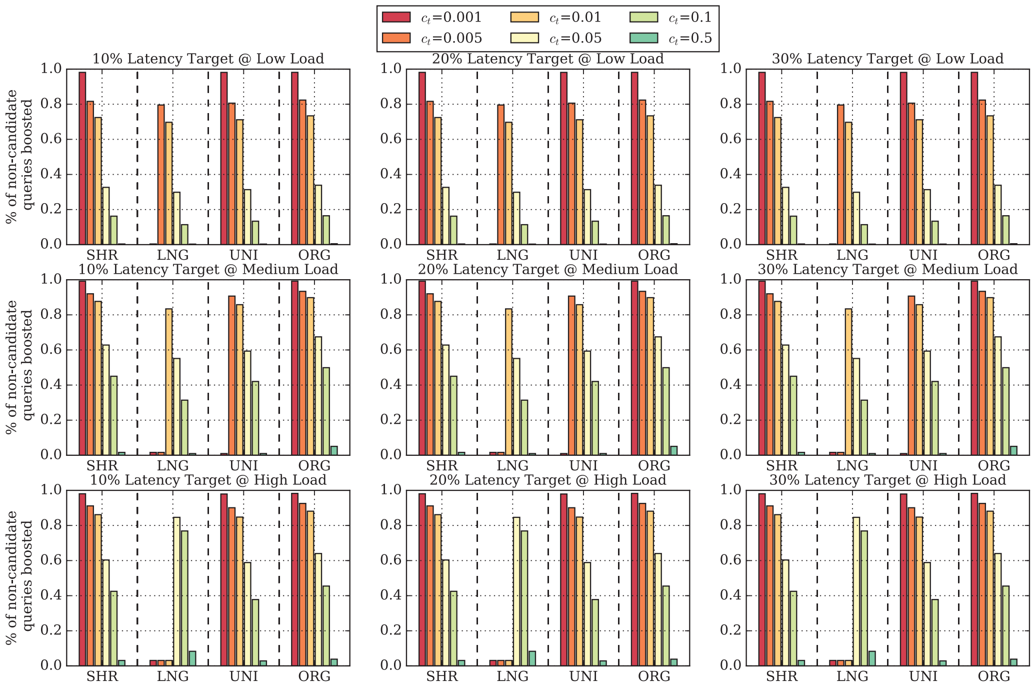

Fig. 24. Percentage of non-candidate Web Search queries (`MEDIUM`s and `LONG`s) boosted when optimizing for energy saving. (Rows 1 to 3: low, medium, and high load. Columns 1 to 3: 10%, 20%, and 30% latency budget.)

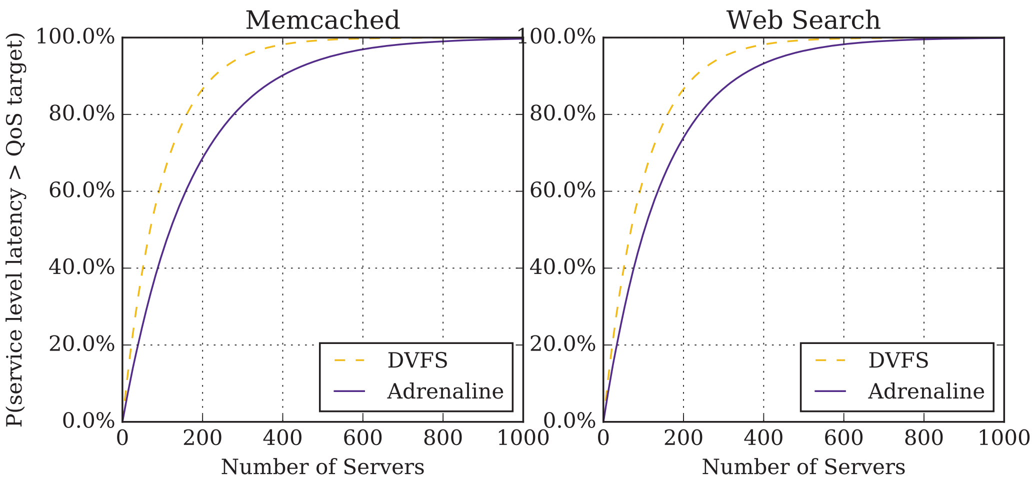

Fig. 25. Effectiveness of Adrenaline in reducing the tail latency at service level comparing to DVFS.

For user-facing applications, their traffic usually follows a diurnal pattern [Meisner et al. 2011] and is rather stable within a short period of time, meaning that the choice for the next decision is more predictable and reliable if given the recent history of traffic. For these applications, datacenter operators can quantize traffic and query compositions into many levels and create a fine-grained lookup table, which provides more precise V/f decisions. On the other hand, if the target application features a varying and unpredictable traffic pattern, datacenter operators can use a coarser-grained lookup table to prevent the decision engine from overfitting to sudden bursts of traffic.

5.6. Tail at Scale

Many large-scale web services, including Memcached and Web Search, use one or even many clusters of machines to serve user queries. As presented in prior work [Barroso et al. 2003; Nishtala et al. 2013], such web services tend to have a large amount of inter-communications and high fan-out. For example, a single user `HTTP` request can result in hundreds of Memcached data fetches and many round trips within a cluster at Facebook. As the number of servers involved in serving a user request increases, the probability of violating the QoS target (e.g., request latency target) grows significantly [Dean and Barroso 2013].

In this section, we evaluate the effectiveness of Adrenaline in reducing the service-level tail latency violations by reducing the tail latency at each leaf node. We use the 99% tail latency measured when the coarse-grained DVFS is enabled as the service-level latency target and compare the probabilities of one request in a fan-out cluster violating the service-level QoS target when using coarse-grained DVFS versus Adrenaline. Figure 25 presents the probability that at least one leaf node misses its QoS target as a function of the number of leaf nodes for a service that must wait for all leaves to respond. As shown, the probability of a request violating the service level QoS target increases drastically as the number of servers in the fan-out cluster increases. With 100 servers, the probability of one request violating the service-level target has increased to 63% for the coarse-grained DVFS. By enabling Adrenaline to reduce the tail latency at leaf nodes, we are able to reduce the probability with 100 servers significantly, by 19% for Memcached and by 11% for Web Search. With 200 servers, Adrenaline achieves an 18% improvement over DVFS for Memcached and a 13% improvement for Web Search.

6. RELATED WORK

6.1. DVFS and Power Management

DVFS has been widely studied to improve the energy efficiency of various scales of computational units over the past several decades [Choi et al. 2005; Wu et al. 2005; Isci et al. 2006; Kaxiras and Martonosi 2008; Lee and Kim 2009; Kolpe et al. 2011; Lo and Kozyrakis 2014; Lo et al. 2014]. However, most of the prior works only look at voltage and frequency scaling decisions at coarse granularity. In Choi et al. [2005], Wu et al. [2005], and Isci et al. [2006], a decision engine, which leverages measured runtime information, is implemented as regression models or dynamic compilation mechanisms to find the best voltage and frequency levels to optimize for energy efficiency. Researchers have also put effort into coarse-grained DVFS decisions on modern multi-core processors [Isci et al. 2006; Lee and Kim 2009; Kolpe et al. 2011; Lo and Kozyrakis 2014], in order to achieve better system throughput and energy efficiency. Similarly to DVFS on processors, some prior works also explore the opportunities of deploying DVFS on memory systems [Deng et al. 2011, 2012a, 2012b; David et al. 2011]. Related works on quick voltage boosting techniques are discussed in Section 3.

Recently, there has been increasing research interest on power management in datacenters [Raghavendra et al. 2008; Leverich et al. 2009; Lo and Kozyrakis 2014; Lo et al. 2014; Meisner et al. 2009] from different perspectives. Among them, PEGASUS [Lo et al. 2014] constantly monitors the workload and the performance statistics of the recent requests within a sliding window and leverages a feedback controller to minimize the power consumption without violating the QoS requirement. This work differs from theirs by identifying and selectively targeting only the requests that most likely will fall in the tail of the distribution.

6.2. Tail Latency

As reported in prior work [Dean and Barroso 2013], tail latency has become a major concern of modern datacenter applications, which has gained much research attention. Since many datacenter workloads have critical tail latency requirements, many works [Laurenzano et al. 2014; Zhang et al. 2014; Yang et al. 2013; Tang et al. 2013; Mars et al. 2011; Tang et al. 2012; Mars and Tang 2013; Delimitrou and Kozyrakis 2013] try to improve the performance predictability of such workloads to optimize datacenter utilization. This requires precise performance interference prediction and careful control of resource sharing, in order to make better scheduling decisions. Another class of optimization tries to reduce the tail latency. DeTail [Zats et al. 2012] proposes to exchange package information across multiple network layers to optimize the scheduling of package processing and distribute the network load evenly. Zellweger et al. [2014] and Belay et al. [2014] move the network stack from kernel space to user space to avoid overhead, so they can achieve lower query latency, as well as higher system throughput. Adrenaline differs from these works due to the fact that it specifically accelerates the queries that tend to appear in the tail, which makes the optimization more efficient than simply boosting the entire latency distribution.

7. CONCLUSION

In this article, we present the Adrenaline methodology that adjusts the voltage/ frequency at query-level granularity to rein in the tail latency as well as to save energy. By evaluating our methodology under various realistic workload configurations, we demonstrate the effectiveness of our methodology. With a naive boost threshold for non-candidate queries, we achieve up to a 2.50× tail latency improvement for Memcached and up to a 3.03× for Web Search over coarse-grained DVFS given a fixed boosting power budget. When optimizing for energy reduction, Adrenaline achieves up to a 1.81× improvement for Memcached and up to a 1.99× improvement for Web Search over DVFS. By using the carefully chosen boost thresholds, Adrenaline further improves the tail latency reduction to 4.82× over coarse-grained DVFS.

REFERENCES

Berk Atikoglu, Yuehai Xu, Eitan Frachtenberg, Song Jiang, and Mike Paleczny. 2012. Workload analysis of a large-scale key-value store. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'12)*. ACM, New York, NY, 53–64. DOI:http://dx.doi.org/10.1145/2254756.2254766

Luiz André Barroso, Jeffrey Dean, and Urs Hölzle. 2003. Web search for a planet: The google cluster architecture. *IEEE Micro* 23, 2 (Mar. 2003), 22–28. DOI:http://dx.doi.org/10.1109/MM.2003.1196112

Adam Belay, George Prekas, Ana Klimovic, Samuel Grossman, Christos Kozyrakis, and Edouard Bugnion. 2014. IX: A protected dataplane operating system for high throughput and low latency. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. USENIX Association, Broomfield, CO, 49–65.

Kihwan Choi, Ramakrishna Soma, and Massoud Pedram. 2005. Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 24, 1 (2005), 18–28.

Intel Corporation. 2008. Intel Turbo Boost Technology in Intel Core Microarchitecture (Nehalem) Based Processors. White paper, Intel Corporation. (November 2008).

Howard David, Chris Fallin, Eugene Gorbatov, Ulf R. Hanebutte, and Onur Mutlu. 2011. Memory power management via dynamic voltage/frequency scaling. In *Proceedings of the 8th ACM International Conference on Autonomic Computing (ICAC'11)*. ACM, New York, NY, 31–40. DOI:http://dx.doi.org/10.1145/1998582.1998590

Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM* 56, 2 (Feb. 2013), 74–80. DOI:http://dx.doi.org/10.1145/2408776.2408794

Christina Delimitrou and Christos Kozyrakis. 2013. Paragon: QoS-aware scheduling for heterogeneous datacenters. In *Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

Qingyuan Deng, David Meisner, Abhishek Bhattacharjee, Thomas F. Wenisch, and Ricardo Bianchini. 2012a. CoScale: Coordinating CPU and memory system DVFS in server systems. In *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-45)*. IEEE Computer Society, Washington, DC, 143–154. DOI:http://dx.doi.org/10.1109/MICRO.2012.22

Qingyuan Deng, David Meisner, Abhishek Bhattacharjee, Thomas F. Wenisch, and Ricardo Bianchini. 2012b. MultiScale: Memory system DVFS with multiple memory controllers. In *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED'12)*. ACM, New York, NY, 297–302. DOI:http://dx.doi.org/10.1145/2333660.2333727

Qingyuan Deng, David Meisner, Luiz Ramos, Thomas F. Wenisch, and Ricardo Bianchini. 2011. MemScale: Active low-power modes for main memory. In *Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XVI)*. ACM, New York, NY, 225–238. DOI:http://dx.doi.org/10.1145/1950365.1950392

Laurel Emurian, Arun Raghavan, Lei Shao, Jeffrey M. Rosen, Marios Papaefthymiou, Kevin Pipe, Thomas F. Wenisch, and Milo Martin. 2014. Pitfalls of accurately benchmarking thermally adaptive chips. *Power (W)* 5 (2014), 10.

Stijn Eyerman and Lieven Eeckhout. 2011. Fine-grained DVFS using on-chip regulators. *ACM Trans. Arch. Code Opt.* 8, 1 (2011), 1.

Michael Ferdman, Almutaz Adileh, Onur Kocberber, Stavros Volos, Mohammad Alisafaee, Djordje Jevdjic, Cansu Kaynak, Adrian Daniel Popescu, Anastasia Ailamaki, and Babak Falsafi. 2012. Clearing the clouds: A study of emerging scale-out workloads on modern hardware. In *Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS XVII)*. ACM, New York, NY, 37–48. DOI:http://dx.doi.org/10.1145/2150976.2150982

Waclaw Godycki, Christopher Torng, Ivan Bukreyev, Alyssa Apsel, and Christopher Batten. 2014. Enabling realistic fine-grain voltage scaling with reconfigurable power distribution networks. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO) (MICRO-47)*. ACM, New York, NY.

A. Gordon, N. Amit, N. Har'El, M. Ben-Yehuda, A. Landau, A. Schuster, and D. Tsafrir. 2012. It's time for low latency. In *ACM SIGARCH Comput. Arch. News*, Vol. 40. 411–422.

Chang-Hong Hsu, Yunqi Zhang, Michael A. Laurenzano, David Meisner, Thomas Wenisch, Jason Mars, Lingjia Tang, and Ronald G. Dreslinski. 2015. Adrenaline: Pinpointing and reining in tail queries with quick voltage boosting. In *Proceedings of the 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 271–282.

Canturk Isci, Alper Buyuktosunoglu, Chen-Yong Cher, Pradip Bose, and Margaret Martonosi. 2006. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 347–358.

Stefanos Kaxiras and Margaret Martonosi. 2008. Computer architecture techniques for power-efficiency. *Synth. Lect. Comput. Arch.* 3, 1 (2008), 1–207.

Wonyoung Kim, D. M. Brooks, and others. 2011. A fully-integrated 3-level DC/DC converter for nanosecond-scale DVS with fast shunt regulation. In *Proceedings of the 2011 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 268–270. DOI:http://dx.doi.org/10.1109/ISSCC.2011.5746313

Wonyoung Kim, M. S. Gupta, et al. 2008. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *Proceedings of the IEEE 14th International Symposium on High Performance Computer Architecture, 2008 (HPCA'08)*. 123–134. DOI:http://dx.doi.org/10.1109/HPCA.2008.4658633

Tejaswini Kolpe, Antonia Zhai, and Sachin S. Sapatnekar. 2011. Enabling improved power management in multicore processors through clustered DVFS. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*. IEEE, 1–6.

Michael A. Laurenzano, Yunqi Zhang, Lingjia Tang, and Jason Mars. 2014. Protean code: Achieving near-free online code transformations for warehouse scale computers. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO) (MICRO-47)*. ACM, New York, NY.

Jungseob Lee and Nam Sung Kim. 2009. Optimizing throughput of power-and thermal-constrained multicore processors using DVFS and per-core power-gating. In *Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE*. IEEE, 47–50.

Jacob Leverich, Matteo Monchiero, Vanish Talwar, Parthasarathy Ranganathan, and Christos Kozyrakis. 2009. Power management of datacenter workloads using per-core power gating. *Comput. Arch. Lett.* 8, 2 (2009), 48–51.

Kevin Lim, David Meisner, Ali G. Saidi, Parthasarathy Ranganathan, and Thomas F. Wenisch. 2013. Thin servers with smart pipes: Designing SoC accelerators for memcached. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA'13)*. ACM, New York, NY, 36–47. DOI:http://dx.doi.org/10.1145/2485922.2485926

David Lo, Liqun Cheng, Rama Govindaraju, Luiz André Barroso, and Christos Kozyrakis. 2014. Towards energy proportionality for large-scale latency-critical workloads. In *Proceeding of the 41st Annual International Symposium on Computer Architecuture*. IEEE Press, 301–312.

David Lo and Christos Kozyrakis. 2014. Dynamic management of TurboMode in modern multi-core chips. In *Proceedings of the 20th IEEE International Symposium on High Performance Computer Architecture (HPCA 2014). 2014*. 603–613.

Jason Mars and Lingjia Tang. 2013. Whare-map: Heterogeneity in "homogeneous" warehouse-scale computers. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA) (ISCA'13)*. ACM, New York, NY, 619–630. DOI:http://dx.doi.org/10.1145/2485922.2485975

Jason Mars, Lingjia Tang, Robert Hundt, Kevin Skadron, and Mary Lou Soffa. 2011. Bubble-Up: Increasing utilization in modern warehouse scale computers via sensible co-locations. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO) (MICRO-44)*. ACM, New York, NY, 248–259. DOI:http://dx.doi.org/10.1145/2155620.2155650 Acceptance Rate: 21% - Selected for IEEE MICRO TOP PICKS.

David Meisner, Brian T. Gold, and Thomas F. Wenisch. 2009. PowerNap: Eliminating server idle power. *ACM SIGARCH Comput. Arch. News* 37, 1 (2009), 205–216.

David Meisner, Christopher M. Sadler, Luiz André Barroso, Wolf-Dietrich Weber, and Thomas F. Wenisch. 2011. Power management of online data-intensive services. In *Proceedings of the 38th Annual International Symposium on Computer Architecture (ISCA'11)*. ACM, New York, NY, 319–330. DOI:http://dx.doi.org/10.1145/2000064.2000103

David Meisner, Junjie Wu, and Thomas F. Wenisch. 2012. BigHouse: A simulation infrastructure for data center systems. In *Proceedings of the 2012 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS'12)*. IEEE Computer Society, Washington, DC, 35–45. DOI:http://dx.doi.org/10.1109/ISPASS.2012.6189204

Timothy N. Miller, Xiang Pan, Renji Thomas, Naser Sedaghati, and Radu Teodorescu. 2012. Booster: Reactive core acceleration for mitigating the effects of process variation and application imbalance in low-voltage chips. In *Proceedings of the 2012 IEEE 18th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 1–12.

Rajesh Nishtala, Hans Fugal, Steven Grimm, Marc Kwiatkowski, Herman Lee, Harry C. Li, Ryan McElroy, Mike Paleczny, Daniel Peek, Paul Saab, David Stafford, Tony Tung, and Venkateshwaran Venkataramani. 2013. Scaling memcache at facebook. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI'13)*. USENIX Association, Berkeley, CA, 385–398.

Nathaniel Pinckney, Matthew Fojtik, Bharan Giridhar, Dennis Sylvester, and David Blaauw. 2013. Shortstop: An on-chip fast supply boosting technique. In *Proceedings of the 2013 Symposium on VLSI Circuits (VLSIC)*. IEEE, C290–C291.

Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. 2008. No "power" struggles: Coordinated multi-level power management for the data center. *SIGARCH Comput. Arch. News* 36, 1 (March 2008), 48–59. DOI:http://dx.doi.org/10.1145/1353534.1346289

Lingjia Tang, Jason Mars, and Mary Lou Soffa. 2012. Compiling for niceness: Mitigating contention for qos in warehouse scale computers. In *Proceedings of the 10th International Symposium on Code Generation and Optimization (CGO) (CGO'12)*. ACM, New York, NY, 1–12. DOI:http://dx.doi.org/10.1145/2259016.2259018 Acceptance Rate: 28% - Best Paper Award!

Lingjia Tang, Jason Mars, Wei Wang, Tanima Dey, and Mary Lou Soffa. 2013. ReQoS: Reactive static/dynamic compilation for QoS in warehouse scale computers. In *Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) (ASPLOS'13)*. ACM, New York, NY, 89–100. DOI:http://dx.doi.org/10.1145/2451116.2451126 Acceptance Rate: 23%.

G. Wang, D. Anand, and others. 2009. Scaling deep trench based eDRAM on SOI to 32nm and Beyond. In *Proceedings of the 2009 IEEE International Electron Devices Meeting (IEDM)*. 1–4. DOI:http://dx.doi.org/10.1109/IEDM.2009.5424375

Qiang Wu, Margaret Martonosi, Douglas W. Clark, Vijay Janapa Reddi, Dan Connors, Youfeng Wu, Jin Lee, and David Brooks. 2005. A dynamic compilation framework for controlling microprocessor energy and performance. In *Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 271–282.

Hailong Yang, Alex Breslow, Jason Mars, and Lingjia Tang. 2013. Bubble-flux: Precise online QoS management for increased utilization in warehouse scale computers. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA) (ISCA'13)*. ACM, New York, NY, 607–618. DOI:http://dx.doi.org/10.1145/2485922.2485974 Acceptance Rate: 19%.

David Zats, Tathagata Das, Prashanth Mohan, Dhruba Borthakur, and Randy Katz. 2012. DeTail: Reducing the flow completion time tail in datacenter networks. *ACM SIGCOMM Comput. Commun. Rev.* 42, 4 (2012), 139–150.

Gerd Zellweger, Simon Gerber, Kornilios Kourtis, and Timothy Roscoe. 2014. Decoupling Cores, Kernels, and Operating Systems. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. USENIX Association, Broomfield, CO, 17–31.

Yunqi Zhang, Michael A. Laurenzano, Jason Mars, and Lingjia Tang. 2014. SMiTe: Precise QoS prediction on real-system SMT processors to improve utilization in warehouse scale computers. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO) (MICRO-47)*. ACM, New York, NY.

Received January 2016; revised December 2016; accepted January 2017