Adrenaline: Pinpointing and Reining in Tail Queries with Quick Voltage Boosting

Chang-Hong Hsu*, Yunqi Zhang*, Michael A. Laurenzano*,
David Meisner†, Thomas Wenisch*, Jason Mars*, Lingjia Tang*, Ronald G. Dreslinski*

Clarity Lab
*University of Michigan - Ann Arbor, MI
{*hsuch,yunqi,mlaurenz,twenisch,profmars,lingjia,rdreslin*}*@umich.edu*

†Facebook, Inc, Menlo Park, CA
meisner@fb.com

***Abstract*—Reducing the long tail of the query latency distribution in modern warehouse scale computers is critical for improving performance and quality of service of workloads such as `Web Search` and `Memcached`. Traditional turbo boost increases a processor's voltage and frequency during a coarse-grain 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, 10's 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; and 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 2.50x tail latency improvement for `Memcached` and up to a 3.03x for `Web Search` over coarse-grained DVFS given a fixed boosting power budget. When optimizing for energy reduction, Adrenaline achieves up to a 1.81x improvement for `Memcached` and up to a 1.99x for `Web Search` over coarse-grained DVFS.

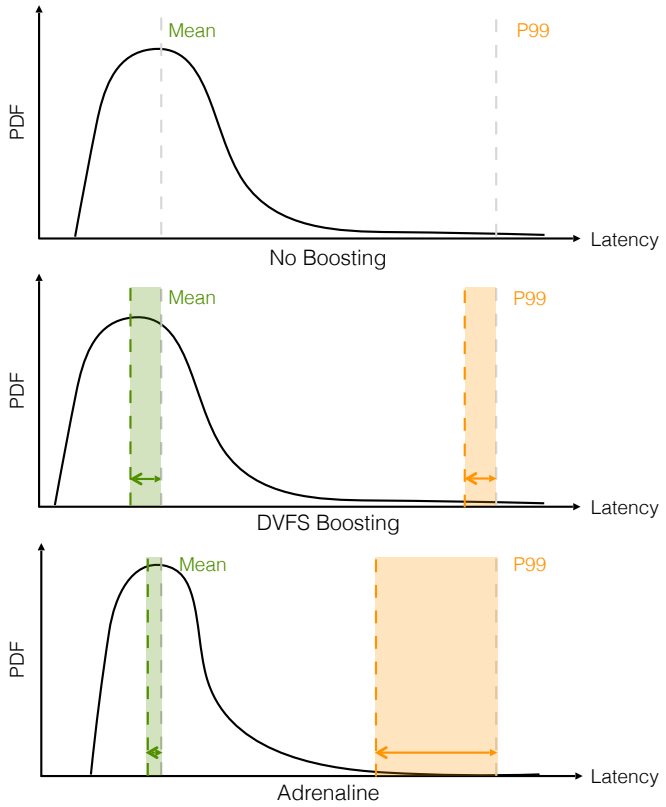

Fig. 1. Query latency distributions without boosting (top), with conventional DVFS boosting (center), and with Adrenaline (bottom).

I. Introduction

Managing high-percentile tail latency is one of the chief performance concerns for web services in modern *Warehouse Scale Computers* (WSCs) [1]; These online data intensive (OLDI) services traverse multi-terabyte data sets for user-facing latency-sensitive queries by dividing ("*sharding*") their datasets over thousands of servers ("*leaf nodes*") acting in concert [2]. A complete query response is formed by aggregating the responses from the individual leaf nodes. However, at hundred- to thousand-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 one-second response time, when aggregating parallel requests to 100 nodes, 63% of queries will take longer than one second [1]. 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 [1].

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 [3] 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 [4], 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-grain 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 [2, 5], the emerging class of fine-grain (10s of nanoseconds) voltage boosting (i.e., *quick boosting*) techniques [6–10] 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:

1) *Investigating whether tail queries are amenable to frequency/voltage boosting.* If tail queries in representative OLDI workloads are not bottlenecked on computation, 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, we can use these characteristics to develop per-query indicators to pinpoint these queries for boosting.
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 paper, we investigate the potential of query-level quick boosting and design **Adrenaline**, a technique to address the limitations of traditional DVFS and achieve tail-sensitive query-level voltage boosting to significantly reduce the tail latency with high energy efficiency. Adrenaline leverages the recently proposed *Short Stop* [8], 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 `Web Search` keywords).

In addition to reducing the tail latency when needed, Adrenaline's fine-grain 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-gain 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 paper 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-grain V/f scaling (10s 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 2.50x tail latency improvement for `Memcached` and up to a 3.03x for `Web Search` over coarse-grained DVFS given a fixed boosting power budget.
- **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 1.81x improvement for `Memcached` and up to a 1.99x for `Web Search` over DVFS.

II. 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 V).

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 1.2 GHz to 2.4 GHz 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. `SET`s' request latency is in general around 2x greater than `GET`s or `DEL`s, 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 1.2 GHz to 2.4 GHz improves SETs' 90-th 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, For Web Search (Nutch [11]) 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 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 85-th percentile). As shown in the figure, short queries (queries with fewer, e.g., 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.

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-grain DVFS boosting. As illustrated in the figure, coarse-gain 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.

III. 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. [6, 7]. Per-Core DVFS integrates a voltage regulator on-die for each core in the system, allowing individual control and nanosecond scale voltage transition times. The Per-Core DVFS technique comes at the expensive of on-chip inductors and reduced regulator efficiency. Intel's TurboBoost [12] enables microsecond scale voltage transitions to allow, for example,

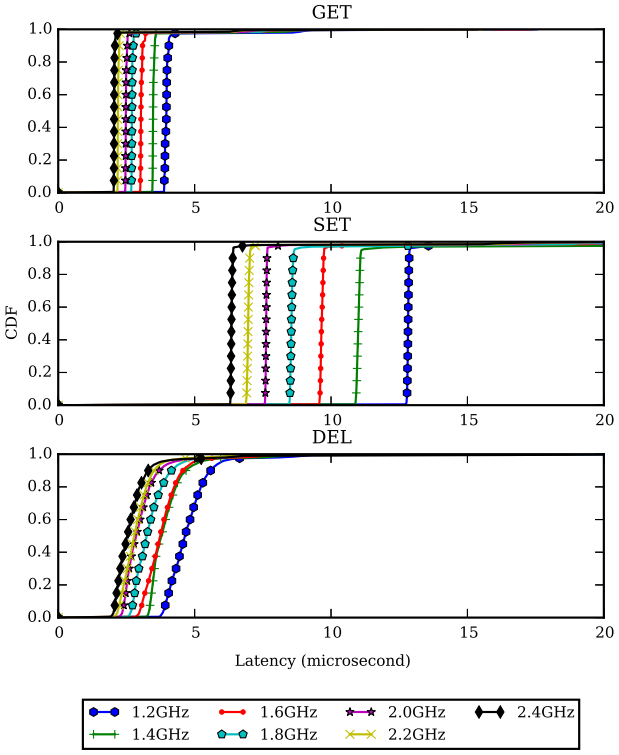

Fig. 2. The cumulative distributions of query latency in Memcached for GET, SET and DEL requests.

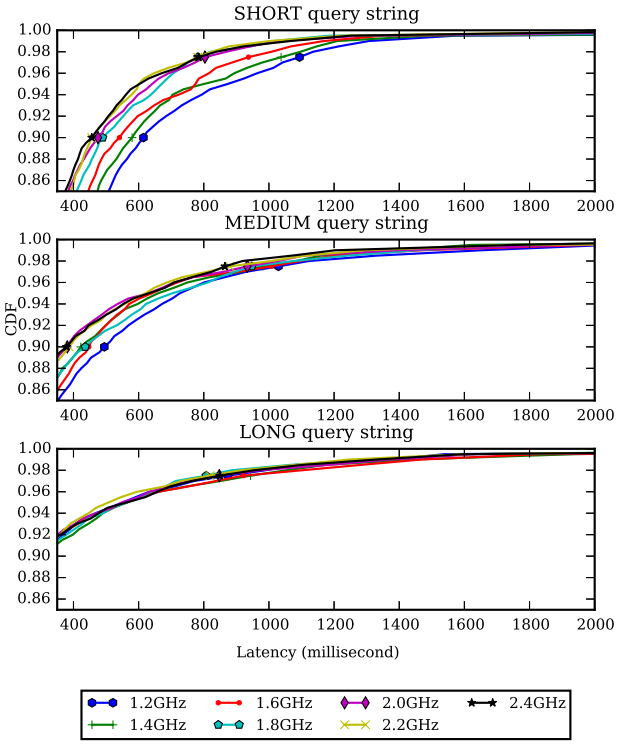

Fig. 3. The cumulative distributions of query latency in Web Search for queries with different numbers of search keywords.

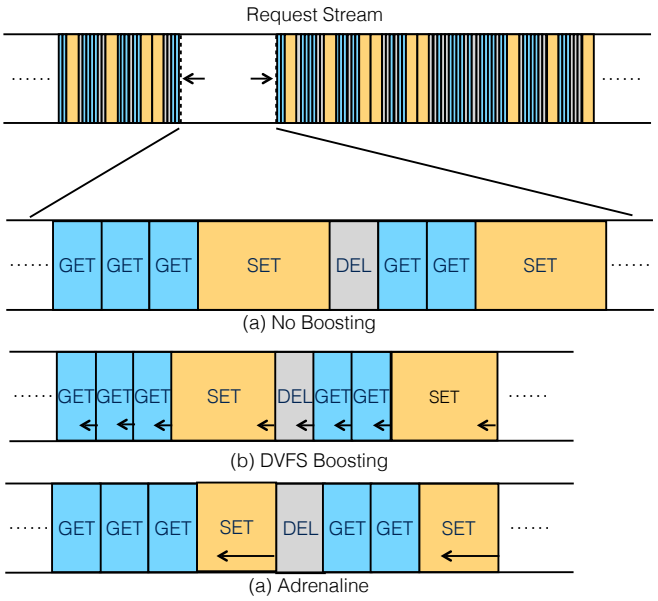

Fig. 4. Adrenaline's Query Level V/f scaling vs. Coarse-grain Sliding Window based V/f scaling.

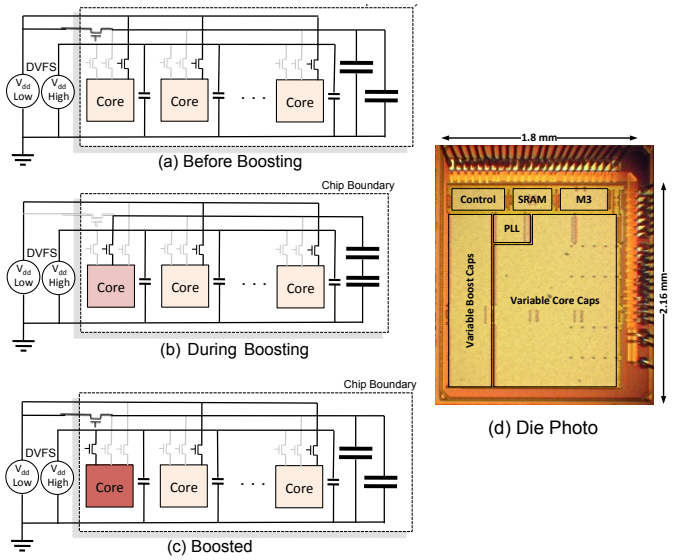

Fig. 5. Short Stop's Dual-V_{dd} chip configurations. (a) shows normal operation, where all cores are connected to the low voltage network and the cap networks are in parallel. (b) shows the cores in boost transition where the core boosting is connected to the output of the serially connected cap network. (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. (d) shows the die photo of the 28nm Short Stop test chip [8].

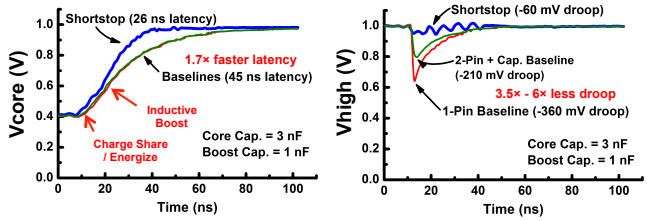

Fig. 6. Measured chip results [8] of Short Stop compared to a standard dual-rail baseline.

the system to exceed thermal budgets for short periods of time. TurboBoost also includes an integrated Power Control Unit (PCU) which makes boosting decisions in hardware based on sensors and other hardware performance counters, eliminating the long latencies associated with OS management. Recently, Godycki et al. have introduced Reconfigurable Power Distribution Networks (RPDN) [10] as a method to improve voltage transition times with the use of a configurable on-chip switch-cap based voltage regulator. Finally, Short Stop [8] and Booster [9] 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 Short Stop 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 Power Control Unit (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.

A. Short Stop

Short Stop [8] 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}. 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 either the V_{ddHigh}, V_{ddLow}, 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 Short Stop 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 [13], 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 ~10-16%. Finally, short stop can be used to perform both fine- and coarse-grained voltage control. Through boosting Short Stop provides nanosecond

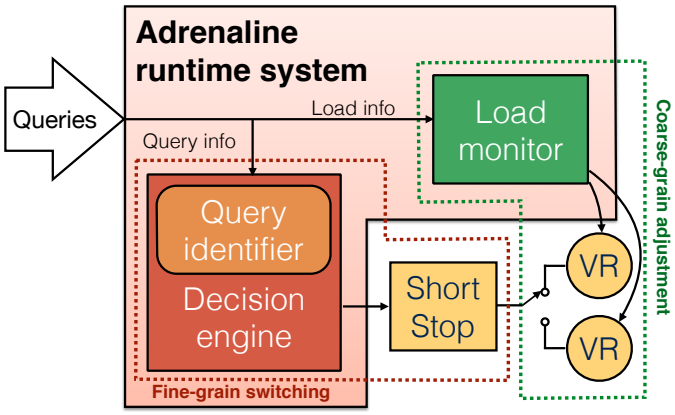

Fig. 7. The Adrenaline framework. Adrenaline makes fine-grain boosting decisions based on the characteristics of each incoming query, and controls the Short Stop circuit to switch between the high/low voltage rails (VRs) quickly. Meanwhile, Adrenaline also monitors long-term loading information, and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

scale voltage transitions to adjust to fine-grained changes in behavior (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).

Short Stop was fabricated and tested in a 28nm technology by Pinckney et al. [8], 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. Short Stop provides nanosecond scale transitions that are 1.7x faster with a 3.5-6x smaller voltage droop. The original paper also provides results showing the scalability of the approach across different core sizes/capacitances.

IV. Adrenaline Framework

This section describes the design of the Adrenaline runtime system, which takes advantage of the fast V/f switching capability of the Short Stop 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 Short Stop 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.

A. 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:

- **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 a 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 number of words.

B. Defining the Boosting Policy

Query type and query composition are used to develop policies that determine when the Short Stop 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 boosting 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 6 keywords are similarly amenable to boosting and have high latency.

Adrenaline also watches the condition in which a query tend 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 boosting 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 boosting decisions for this query. We found by experiment that setting the threshold to 0.5x of the QoS target of the benchmark gives us good results.

C. 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 NICs 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 boosting decision; prototype systems have demonstrated packet delivery latencies below 1.5us and commercial offerings from vendors like Mellanox have similar performance characteristics [14]. 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.

D. 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, the runtime signals the Short Stop hardware to insert the core into a boosting queue. Because Short Stop only allows one core to transition at a time, the queue is serviced in a FIFO fashion. Note the worst case time spent in the queue will be short, as the transition times for Short Stop are on the order of 10's 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, the Adrenaline runtime tracks the additional request. Once all the requests finish, the Adrenaline runtime signals Short Stop to unboost the core.

E. Clock Distribution

Clock distribution is another key consideration in this design. Our proposed solution is to distribute a chip-wide 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 (DLLs) would not need to re-lock during boosting transitions since the voltage and latency of the clock tree does not change.

F. 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). Similar 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 V.

G. Responding to Load Changes

The Short Stop 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.

V. 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.

A. Evaluation Methodology

To accurately evaluate Adrenaline we use the BigHouse simulator [15], 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.

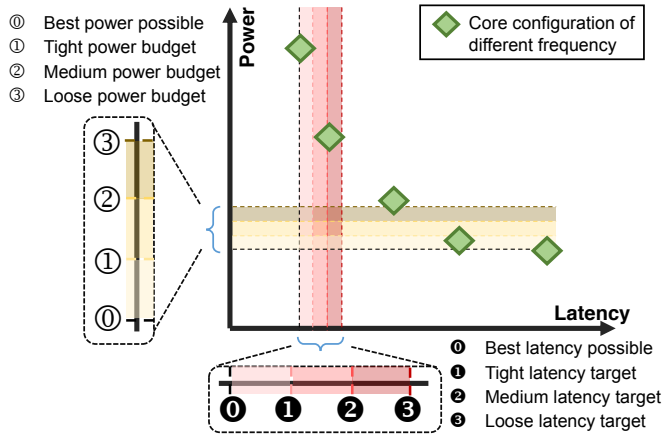

Fig. 8. Selection ofTight/Medium/Loose power budget and Tight/Medium/Loose latency target for our evaluation.

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 [11] 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 we send requests to the instrumented server one-at-a-time so that 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 2 sockets with 4 cores per sockets. The size of the main memory is 136 GB. 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, etc. 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 [16], 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 [16] 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 [16] 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.

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

Different 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 [17], which suggests to use an exponential distribution as an approximation according to the empirical measurement from the production Google `Web Search` server node. And we use a Generalized Pareto distribution for `Memcached` as suggested in prior work [16], which characterizes the production `Memcached` traffic at Facebook.

2) Power modeling: The power equation we are using is adapted based on previous work [18]. 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 IV-E. The 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 power consumption measured on a real machine, and $P_{sta,0}$ is the static part of P_0 and $P_{dyn,0}$ is the dynamic counterpart; f_0 is the lowest frequency step the machine can run at. P and f are the power and the frequency the CPU is currently operating at.

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. Upon 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

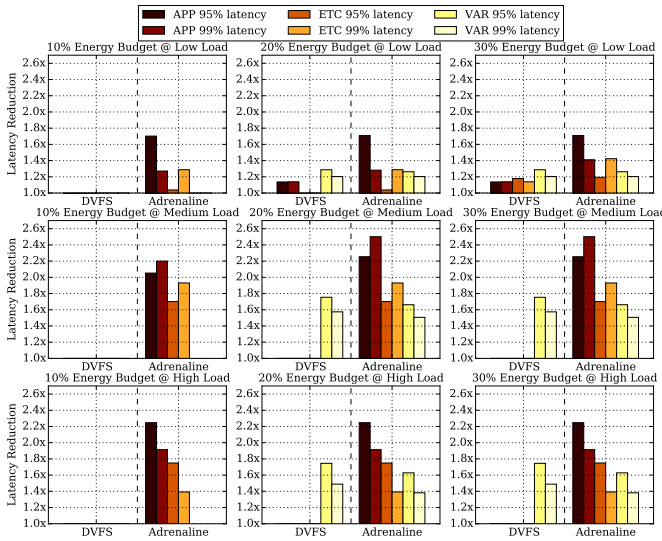

Fig. 9. Tail latency reduction for `Memcached` using coarse-grain DVFS vs. Adrenaline. The row presents three load levels and the column represents three energy budgets for boosting. Adrenaline achieves much higher tail latency reduction at various load levels, across workload compositions and energy budgets.

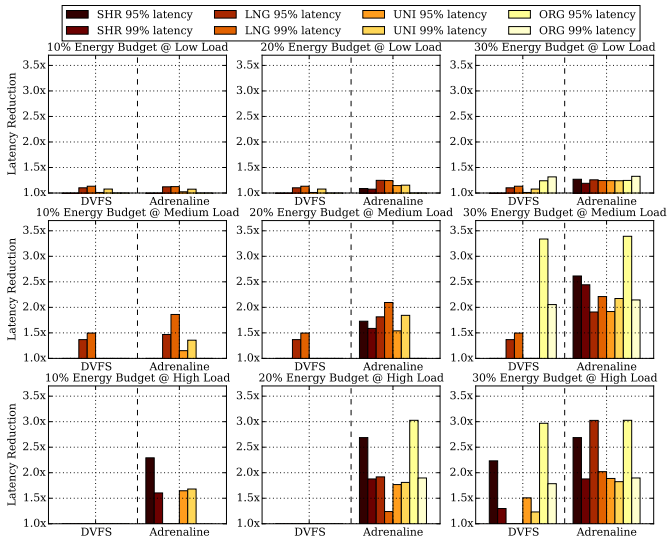

Fig. 10. Tail latency reduction for `Web Search` by relaxing energy budget at various load levels using Adrenaline and DVFS.

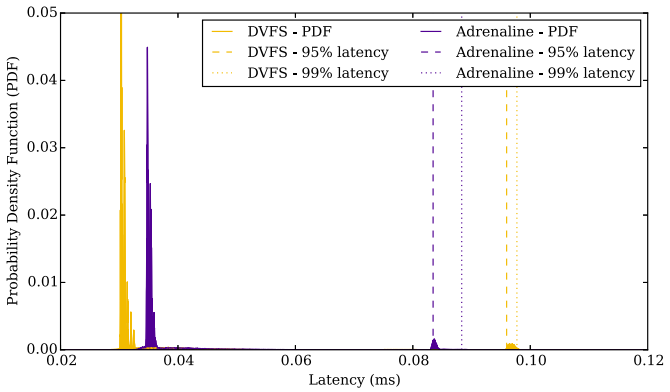

Fig. 11. Measured latency probability density function of `Memcached` when applying DVFS and Adrenaline respectively.

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 1.2 GHz to 1.8 GHz at 50% of a request, it calculates the energy used for the first half of the request at the 1.2 GHz power number, and uses the 1.8 GHz service time to process the second half of the request; upon the request completion, it calculates the energy consumption of the second half of the request using the 1.8 GHz power number, and switches back to using the 1.2 GHz service-time distribution for the next incoming request.

B. Reining in the Tail

In this section, we evaluate the effectiveness of Adrenaline and the conventional coarse-grain 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 the 95%-tile and 99%-tile.

Memcached - Figure 9 shows the tail latency reduction for `Memcached` achieved by both coarse-grain 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-grain 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-grain DVFS fails to reduce the tail latency, while Adrenaline is able to improve the 95%-tile latency by up to 2.20x and the 99%-tile latency by up to 2.25x. This is due to the fact that the 10% energy budget is too small for the coarse-grain 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-grain DVFS starts 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 GETs and DELs promptly, coarse-grain 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-grain 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-grain 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 3.03x tail reduction when given a 20% budget. Given a 30% energy budget, while coarse-grain 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-grain DVFS across all other workload compositions.

Latency Distribution - To better understanding the impact of coarse-grain 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 (yellow), and the other one using Adrenaline (purple). The dotted lines indicate the 95% and 99%-tile latency of each distribution. For the distributions both coarse-grain DVFS and Adrenaline show 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-grain DVFS shifts the entire distribution to the left, reducing both the mean and 99%-tile 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-grain 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-grain DVFS.

Figure 12 shows a Pareto-optimal curve for the power versus 99% latency tradeoff of different voltage 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).

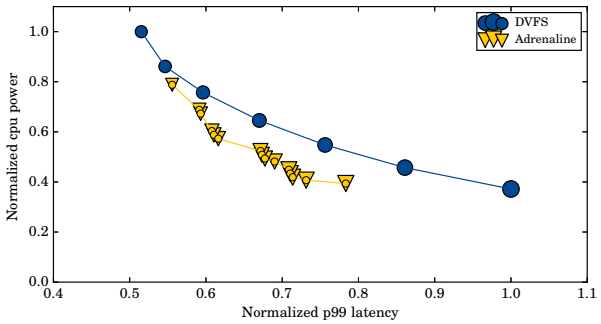

Fig. 12. The scatter plot of Memcached, in which each data point represents the normalized power/latency performance of a single CPU configuration with load composition APP and low traffic.

C. Energy Saving

In addition to reining the tail latency by fine-grain 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-grain DVFS and Adrenaline in improving the energy efficiency as we gradually relax the tail latency target. Specifically, we refer 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-grain 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 2.12x energy savings over the no-DVFS baseline, whereas DVFS can only save 1.56x in the best scenario. At high load, both Adrenaline and coarse-grain 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 savings compared to coarse-grain DVFS. This is because Adrenaline leverages the observation demonstrated in Figure 2 and Figure 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.

D. 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 always performs better than coarse-grain DVFS. When optimizing for tail 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.

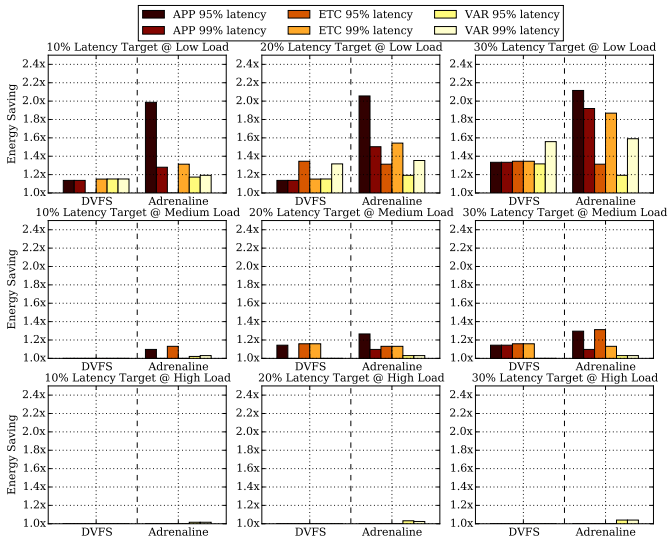

Fig. 13. Energy saving for `Memcached` by relaxing the tail latency target at various load levels using Adrenaline and coarse-grain DVFS.

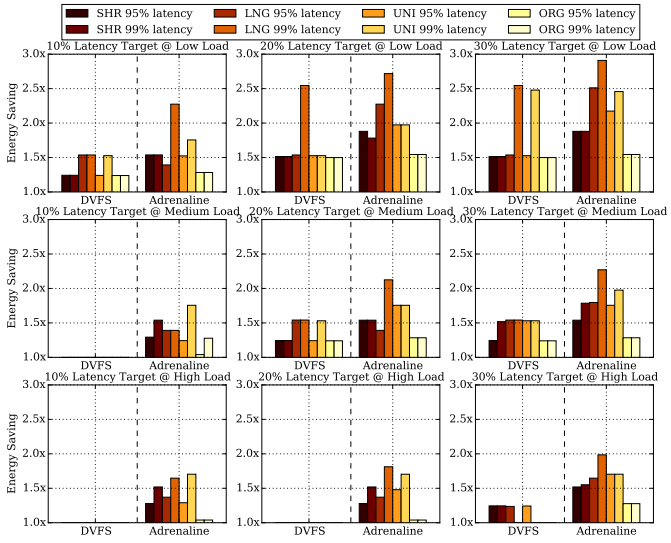

Fig. 14. Energy saving for `Web Search` by relaxing the tail latency target at various load levels using Adrenaline and DVFS.

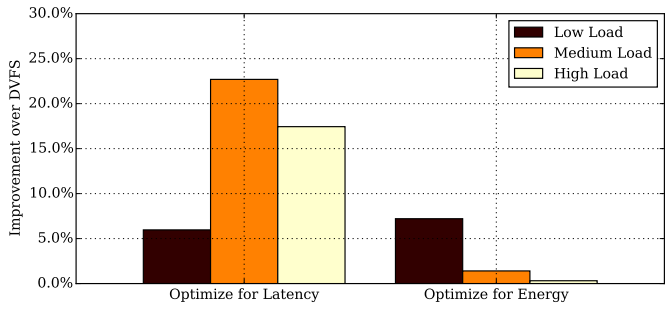

Fig. 15. Improvement of Adrenaline over coarse-grain DVFS for `Memcached`.

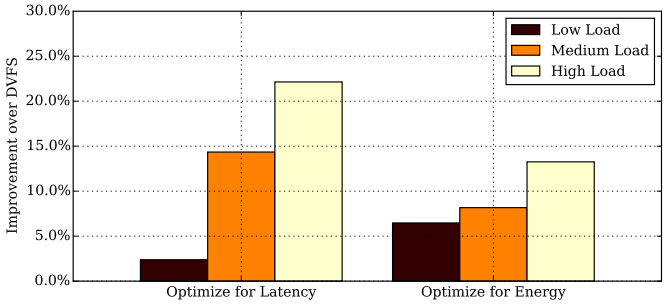

Fig. 16. Improvement of Adrenaline over coarse-grain DVFS for `Web Search`.

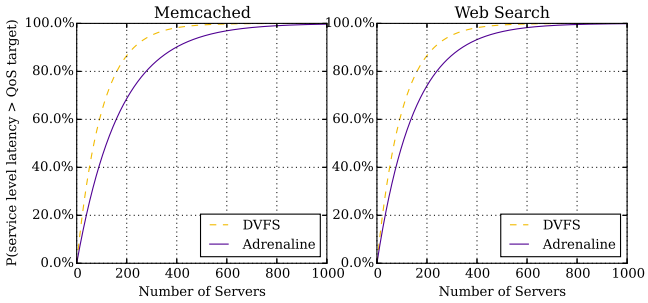

Fig. 17. Effectiveness of Adrenaline in reducing the tail latency at service level comparing to DVFS.

E. 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 [2, 19], 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 Quality of Service target (e.g., request latency target) grows significantly [1].

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-grain DVFS versus Adrenaline. Figure 17 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-grain 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`.

VI. Related Work

A. DVFS and Power Management

Dynamic voltage and frequency scaling (DVFS) has been widely studied to improve the energy efficiency of various scales of computational units over the past several decades [4, 20–26]. However, most of the prior works only look at voltage and frequency scaling decisions at coarse granularity. In [20–22], 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 efforts into coarse-grained DVFS decisions on modern multi-core processors [22, 24–26], 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 [27–30]. Related works on quick voltage boosting techniques are discussed in Section 3.

Recently, there is an increasing research interest on power management in datacenters [4, 18, 26, 31, 32] from different perspectives. Among them, PEGASUS [4] 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.

B. Tail Latency

As reported in prior work [1], 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 [33–40] 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 [41] proposes to exchange package information across multiple network layers to optimize the scheduling of package processing and distribute the network load evenly. [42, 43] move the network stack from kernel space to user space to avoid overhead, so that 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.

VII. Conclusion

In this paper, 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. We achieve up to a 2.50x tail latency improvement for `Memcached` and up to a 3.03x for `Web Search` over coarse-grained DVFS given a fixed boosting power budget. When optimizing for energy reduction, Adrenaline achieves up to a 1.81x improvement for `Memcached` and up to a 1.99x improvement for `Web Search` over DVFS.

VIII. Acknowledgement

We thank our anonymous reviewers for their feedback and suggestions. This research was supported by Google and by the National Science Foundation under grants CCF-SHF-1302682 and CNS-CSR-1321047.

References

[1] J. Dean and L. A. Barroso, "The Tail at Scale," *Commun. ACM*, vol. 56, pp. 74–80, Feb. 2013.

[2] L. A. Barroso, J. Dean, and U. Hölzle, "Web Search for a Planet: The Google Cluster Architecture," *IEEE Micro*, vol. 23, pp. 22–28, Mar. 2003.

[3] L. Emurian, A. Raghavan, L. Shao, J. M. Rosen, M. Papaefthymiou, K. Pipe, T. F. Wenisch, and M. Martin, "Pitfalls of Accurately Benchmarking Thermally Adaptive Chips," *Power (W)*, vol. 5, p. 10.

[4] D. Lo, L. Cheng, R. Govindaraju, L. A. Barroso, and C. Kozyrakis, "Towards energy proportionality for large-scale latency-critical workloads," in *Proceeding of the 41st annual international symposium on Computer architecuture*, pp. 301–312, IEEE Press, 2014.

[5] K. Lim, D. Meisner, A. G. Saidi, P. Ranganathan, and T. F. Wenisch, "Thin Servers with Smart Pipes: Designing SoC Accelerators for Memcached," in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA '13, (New York, NY, USA), pp. 36–47, ACM, 2013.

[6] W. Kim, M. Gupta, *et al.*, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pp. 123 –134, February 2008.

[7] W. Kim, D. Brooks, *et al.*, "A fully-integrated 3-level DC/DC converter for nanosecond-scale DVS with fast shunt regulation," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, pp. 268 –270, feb. 2011.

[8] N. Pinckney, M. Fojtik, B. Giridhar, D. Sylvester, and D. Blaauw, "Shortstop: An on-chip fast supply boosting technique," in *VLSI Circuits (VLSIC), 2013 Symposium on*, pp. C290–C291, IEEE, 2013.

[9] T. N. Miller, X. Pan, R. Thomas, N. Sedaghati, and R. Teodorescu, "Booster: Reactive core acceleration for mitigating the effects of process variation and application imbalance in low-voltage chips," in *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pp. 1–12, IEEE, 2012.

[10] W. Godycki, C. Torng, I. Bukreyev, A. Apsel, and C. Batten, "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, (New York, NY, USA), ACM, 2014.

[11] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the Clouds: A Study of Emerging Scale-out Workloads on Modern Hardware," in *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVII, (New York, NY, USA), pp. 37–48, ACM, 2012.

[12] I. Corporation, "Intel Turbo Boost Technology in Intel Core Microarchitecture (Nehalem) Based Processors." White paper, Intel Corporation, November 2008.

[13] G. Wang, D. Anand, *et al.*, "Scaling deep trench based eDRAM on SOI to 32nm and Beyond," in *Electron Devices Meeting (IEDM), 2009 IEEE International*, pp. 1 –4, dec. 2009.

[14] A. Gordon, N. Amit, N. Har'El, M. Ben-Yehuda, A. Landau, A. Schuster, and D. Tsafrir, "It's Time for Low Latency," in *ACM SIGARCH Computer Architecture News*, vol. 40, pp. 411–422, 2012.

[15] D. Meisner, J. Wu, and T. F. Wenisch, "BigHouse: A Simulation Infrastructure for Data Center Systems," in *Proceedings of the 2012 IEEE International Symposium on Performance Analysis of Systems & Software*, ISPASS '12, (Washington, DC, USA), pp. 35–45, IEEE Computer Society, 2012.

[16] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny, "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, (New York, NY, USA), pp. 53–64, ACM, 2012.

[17] D. Meisner, C. M. Sadler, L. A. Barroso, W.-D. Weber, and T. F. Wenisch, "Power Management of Online Data-intensive Services," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, (New York, NY, USA), pp. 319–330, ACM, 2011.

[18] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: eliminating server idle power," *ACM SIGARCH Computer Architecture News*, vol. 37, no. 1, pp. 205–216, 2009.

[19] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani, "Scaling Memcache at Facebook," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, (Berkeley, CA, USA), pp. 385–398, USENIX Association, 2013.

[20] K. Choi, R. Soma, and M. Pedram, "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," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, no. 1, pp. 18–28, 2005.

[21] Q. Wu, M. Martonosi, D. W. Clark, V. J. Reddi, D. Connors, Y. Wu, J. Lee, and D. Brooks, "A dynamic compilation framework for controlling microprocessor energy and performance," in *Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture*, pp. 271–282, IEEE Computer Society, 2005.

[22] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "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*, pp. 347–358, IEEE Computer Society, 2006.

[23] S. Kaxiras and M. Martonosi, "Computer architecture techniques for power-efficiency," *Synthesis Lectures on Computer Architecture*, vol. 3, no. 1, pp. 1–207, 2008.

[24] J. Lee and N. S. Kim, "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*, pp. 47–50, IEEE, 2009.

[25] T. Kolpe, A. Zhai, and S. S. Sapatnekar, "Enabling improved power management in multicore processors through clustered DVFS," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2011*, pp. 1–6, IEEE, 2011.

[26] D. Lo and C. Kozyrakis, "Dynamic management of TurboMode in modern multi-core chips," in *20th IEEE International Symposium on High Performance Computer Architecture, HPCA 2014, Orlando, FL, USA, February 15-19, 2014*, pp. 603–613, 2014.

[27] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini, "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, (Washington, DC, USA), pp. 143–154, IEEE Computer Society, 2012.

[28] H. David, C. Fallin, E. Gorbatov, U. R. Hanebutte, and O. Mutlu, "Memory Power Management via Dynamic Voltage/Frequency Scaling," in *Proceedings of the 8th ACM International Conference on Autonomic Computing*, ICAC '11, (New York, NY, USA), pp. 31–40, ACM, 2011.

[29] Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini, "MemScale: Active Low-power Modes for Main Memory," in *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVI, (New York, NY, USA), pp. 225–238, ACM, 2011.

[30] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini, "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, (New York, NY, USA), pp. 297–302, ACM, 2012.

[31] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No "Power" Struggles: Coordinated Multi-level Power Management for the Data Center," *SIGARCH Comput. Archit. News*, vol. 36, pp. 48–59, Mar. 2008.

[32] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis, "Power management of datacenter workloads using per-core power gating," *Computer Architecture Letters*, vol. 8, no. 2, pp. 48–51, 2009.

[33] M. A. Laurenzano, Y. Zhang, L. Tang, and J. Mars, "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, (New York, NY, USA), ACM, 2014.

[34] Y. Zhang, M. A. Laurenzano, J. Mars, and L. Tang, "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, (New York, NY, USA), ACM, 2014.

[35] H. Yang, A. Breslow, J. Mars, and L. Tang, "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, (New York, NY, USA), pp. 607–618, ACM, 2013. Acceptance Rate: 19

[36] L. Tang, J. Mars, W. Wang, T. Dey, and M. L. Soffa, "ReQoS: Reactive Static/Dynamic Compilation for QoS in Warehouse Scale Computers," in *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, ASPLOS '13, (New York, NY, USA), pp. 89–100, ACM, 2013. Acceptance Rate: 23

[37] J. Mars, L. Tang, R. Hundt, K. Skadron, and M. L. Soffa, "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, (New York, NY, USA), pp. 248–259, ACM, 2011. Acceptance Rate: 21

[38] L. Tang, J. Mars, and M. L. Soffa, "Compiling for Niceness: Mitigating Contention for QoS in Warehouse Scale Computers," in *Proceedings of the Tenth International Symposium on Code Generation and Optimization (CGO)*, CGO '12, (New York, NY, USA), pp. 1–12, ACM, 2012. Acceptance Rate: 28

[39] J. Mars and L. Tang, "Whare-map: Heterogeneity in "homogeneous" warehouse-scale computers," in *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA)*, ISCA '13, (New York, NY, USA), pp. 619–630, ACM, 2013.

[40] C. Delimitrou and C. Kozyrakis, "Paragon: QoS-Aware Scheduling for Heterogeneous Datacenters," in *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, March 2013.

[41] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "DeTail: reducing the flow completion time tail in datacenter networks," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 139–150, 2012.

[42] G. Zellweger, S. Gerber, K. Kourtis, and T. Roscoe, "Decoupling Cores, Kernels, and Operating Systems," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, (Broomfield, CO), pp. 17–31, USENIX Association, Oct. 2014.

[43] A. Belay, G. Prekas, A. Klimovic, S. Grossman, C. Kozyrakis, and E. Bugnion, "IX: A Protected Dataplane Operating System for High Throughput and Low Latency," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, (Broomfield, CO), pp. 49–65, USENIX Association, Oct. 2014.