On the Energy Proportionality of Distributed NoSQL Data Stores

Balaji Subramaniam and Wu-chun Feng

Department. of Computer Science, Virginia Tech
{balaji, feng}@cs.vt.edu

Abstract. The computing community is facing several *big data* challenges due to the unprecedented growth in the volume and variety of data. Many large-scale Internet companies use distributed NoSQL data stores to mitigate these challenges. These NoSQL data-store installations require massive computing infrastructure, which consume significant amount of energy and contribute to operational costs. This cost is further aggravated by the lack of energy proportionality in servers.
Therefore, in this paper, we study the energy proportionality of servers in the context of a distributed NoSQL data store, namely Apache Cassandra. Towards this goal, we measure the power consumption and performance of a Cassandra cluster. We then use *power and resource provisioning* techniques to improve the energy proportionality of the cluster and study the feasibility of achieving an energy-proportional data store. Our results show that a *hybrid (i.e., power and resource) provisioning* technique provides the best power savings — as much as 55%.

1 Introduction

The computing community is facing a data deluge. Software developers have to deal with large volumes and variety of data (a.k.a. *big data*). NoSQL data stores, such as Cassandra [11], Bigtable [5] and DynamoDB [17], have emerged as a viable alternative to the traditional relational databases to handle *big data*. They provide fast and scalable storage with unconventional storage schemas. The entire set of data is partitioned and stored in many different servers, and a key-value store is used to respond to queries from clients. In order to meet service-level objectives (SLOs), these distributed data stores can span several hundred servers (or a cluster) to provide efficient access to huge volumes of data. For example, Netflix uses Cassandra installations that span 2500 servers and stores 300 terabytes of data.

Such cluster installations consume a significant amount of energy, and in turn, contribute to their operational costs. Moreover, the operational cost is exacerbated by the lack of energy proportionality in servers. To address these issues, *power provisioning* techniques [19,18,7,9] have been shown to improve the energy proportionality of such servers. These techniques take advantage of low utilization periods or short idle periods to assign low-power states to subsystems, such as the CPU and memory, using mechanisms such as dynamic voltage-frequency

scaling (DVFS) or Intel's running average power limit (RAPL). Other researchers have provided solutions to improve the energy proportionality by using *resource provisioning* techniques. These techniques use workload consolidation to minimize the number of servers required to sustain a desired throughput and reduce energy consumption by turning off the servers not in use [22].

With the volume of data growing at a rapid pace and the variety of data continuing to evolve and change, distributed NoSQL systems will need increasingly larger cluster installations. Improvements in the energy proportionality of such installations will need to come from both hardware- and software-controlled power management. Thus, our aim in this paper is to study the energy proportionality of clusters in the context of distributed NoSQL data stores. Specifically, we analyze the effectiveness of different software-controlled power management techniques, such as *power and resource provisioning*, to improve the energy proportionality of NoSQL data store installations. Using Cassandra as our distributed NoSQL data store, we make the following contributions:

- *A detailed study of the power consumption and energy proportionality of a Cassandra cluster, including power measurements of individual components within the cluster.* In short, we find that the idle power consumption is very high in such distributed NoSQL installations and that the CPU contributes most to the dynamic power range of the cluster.
- *An investigation into the effects of different power management techniques on the energy proportionality of a Cassandra cluster.* Our results show that significant power savings (up to 55%), and in turn, improvements in energy proportionality can be achieved at low load-levels by taking advantage of the difference between measured latency and SLO. We also find that a hybrid (i.e., power and resource) provisioning technique provides the best power savings, closely followed by resource provisioning.

The rest of the paper is structured as follows. A brief overview of the workload generator, Cassandra, the power management interface and the experimental setup is described in Section 2. We present the baseline power and performance measurements in Section 3. The trade-offs between latency, power savings and energy proportionality using different power management techniques is described in Section 4. A discussion of the related work is presented in Section 5. Section 6 concludes the paper.

2 Background

In this section, we present the following background information to provide context for our work: (1) the workload generator, (2) the distributed NoSQL data store, (3) the power management interface and (4) the experimental set-up.

2.1 Workload Generator: YCSB

To generate the workload for our experiments, we use the Yahoo! Cloud Serving Benchmark (YCSB) [3]. YCSB is a benchmarking framework to evaluate the

performance of cloud data-serving systems. The framework consists of a load-generating client and a set of standard workloads, such as read-heavy or write-heavy workloads, which helps in stressing important performance aspects of a data-serving system. YCSB also allows the user to configure benchmarking parameters such as number of client threads and the number of record counts.

2.2 NoSQL Data Store: Apache Cassandra

We use Cassandra [11,1] as our distributed NoSQL data store. Cassandra aims to manage large amounts of data distributed across many commodity servers. It provides a reliable, high-availability service using a peer-to-peer architecture. The data is split across each node in the cluster using consistent hashing. Specifically, a random value within the range of the hash-function output is assigned to each node in the system and represents its position in the ring. Each data item identified by a key is assigned to a node by hashing the data item's key to yield its position on the ring and then walking the ring clockwise to find the first node with a position larger than the item's position.

To improve availability, each data item can be replicated at N different hosts, where N is the replication factor. Cassandra uses a gossip protocol to locally determine whether any other nodes in the cluster have failed. A Cassandra cluster can be provisioned with extra resources easily by providing the newly added node with information about the *seed* node (initial contact points) in the already existing cluster. All of the above features make Cassandra not only tolerant against single points of failure but also scalable. To evaluate Cassandra, we use the aforementioned YCSB workload generator.

2.3 Power Management Interface

To study energy proportionality in the context of a distributed NoSQL data store, namely Cassandra, we use Intel's Running Average Power Limit (RAPL) [2,6] interfaces for power management. RAPL, which debuted in Intel Sandy Bridge processors, provides interfaces to mechanisms that can measure the energy consumption of specific subsystems and enforce power consumption limits on them. The RAPL interfaces can be programmed using the model-specific registers (MSRs).

2.4 Experimental Setup

With respect to hardware, we use a four-node cluster as our evaluation testbed. Each node consists of an Intel Xeon E5-2620 processor, 16 GB of memory, and a 256-GB hard disk. A separate server runs the YCSB client, which sends data serving requests to the Cassandra cluster.

For configuration, we load 10-million records into the data store with a replication of three so that the Cassandra cluster can sustain multiple node failures. For the workloads, we evaluate with a read-only workload and an update-only workload. The requests follow a Zipfian distribution.

For data collection, the YCSB client reports the performance achieved in terms of throughput and latency, specifically average latency and latencies at the 95th and 99th percentile. To collect power numbers, a *Watts Up* power meter recorded full-system power measurements while the RAPL interfaces collected subsystem-level power measurements.

3 Baseline Measurements for Power and Latency

Here we measure the performance and corresponding power consumption of the Cassandra cluster. The goals are two-fold: (1) to improve our understanding of the relationship between power and performance for a distributed NoSQL data store and (2) to identify any potential for power savings.

Figure 1 shows the power distribution for the read-only workload and the update-only workload across the entire cluster. The values reported in Figure 1 are based on the sum of the power consumption from each node in the cluster and averaged over multiple runs. System components other than the processor and memory are represented as *"Others"* [1] in legend of the figure.

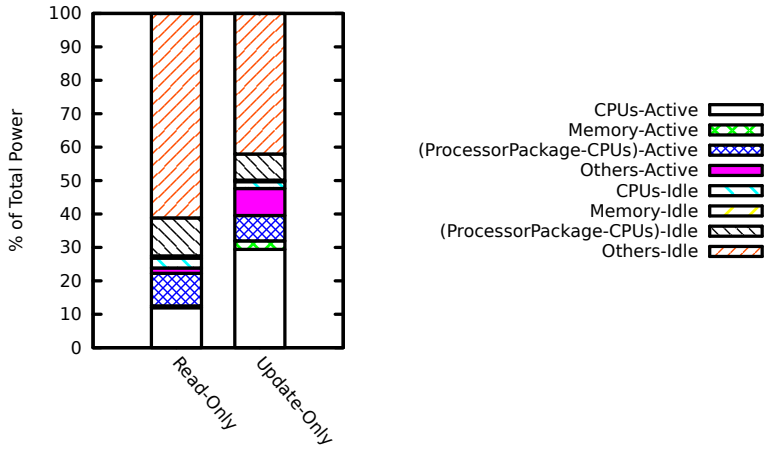

Fig. 1. Component-Level Power Distribution.

The other components of the system consumed a significant portion of the power when idling (i.e., 61% and 42% of the total power for the read-only and update-only workloads, respectively). However, they only add 2% and 8% more to the power consumption when the system is under load for the two workload cases, respectively. In contrast, the processor package adds another 22% and

[1] The other components, denoted by "Others," also include the power consumption of the hard disk.

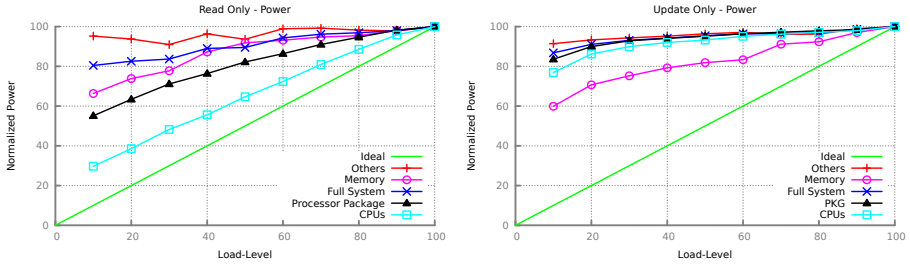

Fig. 2. Energy Proportionality

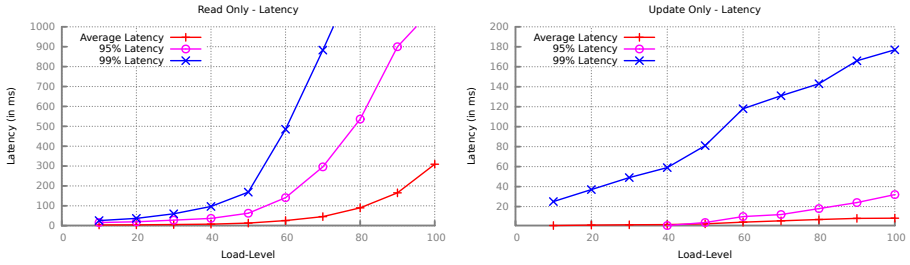

Fig. 3. Latency Profile

37% to the power consumption distribution for the workloads under evaluation. This shows that the processor contributes significantly to the dynamic power range of the cluster under test. The plot shows only the distribution of power consumption at the highest load on the system (i.e., 100% load-level). However, we are also interested in analyzing the energy proportionality (i.e., the power consumption at different load-levels) of the system, which leads us to Figure 2.

Figure 2 shows the normalized power consumed by the system at different load-levels. We normalize power relative to the power consumed at 100% load-level by that component. For example, each value in the CPU power trend is normalized to the power consumed by the CPU at 100% load-level. We also show the ideal energy proportional case for comparison purposes. Several insights can be gleaned from these figures. As evident from the figure, the system exhibits poor energy proportionality for both the workloads as it varies between 80% and 100% of normalized power. The read-only workload, however, exhibits better energy proportionality on the cluster than the update-only workload. For example, the CPUs have a linear increase in power consumption and varies between 30% and 100% for the read-only workload. However, the power consumption of the CPU varies only between 75% and 100% for the update-only workload even though the CPUs consume a higher percentage of the overall power, as shown in Figure 1. Later in this paper, we analyze whether existing power-management techniques can help to improve the energy proportionality of the cluster system.

Figure 3 shows the latency profile for the two workloads. We present the average latency as well as the latencies at the 95th and 99th percentile at different

load-levels for each workload. The performance targets (or SLOs) are typically based on either the 95th or 99th percentile rather than by the average. These SLOs are fixed at a particular value by the service provider and do not depend on the load-level of the system. SLOs provide us with the opportunity to trade latency for power under certain load-levels.

If the cluster achieves lower latencies at low load-levels, power-management techniques can be used to improve the efficiency of the cluster by provisioning power or provisioning server resources. For example, if the SLO is set as 160ms on the 99th-percentile latency for the update-only workload, there exists headroom between the measured latency and the SLO for any load-level less than 90%. We can use this headroom to improve the power consumption of the cluster, thus improving energy proportionality. In rest of the paper, we examine this power versus latency trade-off using different power-management techniques.

4 Evaluation of Power-Management Techniques for NoSQL Data Store

In this section, we evaluate the effect of different power-management techniques on the power consumption and energy proportionality of the Cassandra cluster. In this paper, we evaluate three different techniques: *power, resource, and hybrid (i.e., power and resource) provisioning*. The power-management techniques are applied while meeting the SLOs. Two different SLOs on latency, one on the 95th percentile and the other on the 99th percentile, are evaluated for each of the workload. For the read-only workload, we fix the SLOs at 600 ms for the 95th-percentile latency and 1000 ms for the 99th-percentile latency. For the update-only workload, the SLOs are fixed at 25 ms for the 95th-percentile latency and 160 ms for the 99th-percentile latency.

For power provisioning, we use the power-limiting interface of RAPL. RAPL maintains an average power limit over a sliding window instead of enforcing strict limits on the instantaneous power. The advantage of having an average power limit is that if the average performance requirement is within the specified power limits, the workload will not incur any performance degradation even if the performance requirement surpasses the power limit over short bursts of time. (The user has to provide a power bound and a time window in which the limit has to be maintained.)

In this paper, we use only CPU power limiting as we have shown that it contributes most to the dynamic power range of the system (see Figure 1). We run the workload at a particular load-level and manually change the CPU power limit in order to find the best power limit for the CPU which satisfies the SLOs. The evaluation of resource provisioning is done by manually hibernating nodes in the cluster. Hibernating nodes saves approximately 40 watts per node. We manually find the optimal number of nodes to run Cassandra to satisfy SLOs at a given load-level. We also evaluate a hybrid version of power and resource provisioning. First, we run the workload on the optimal number of nodes and

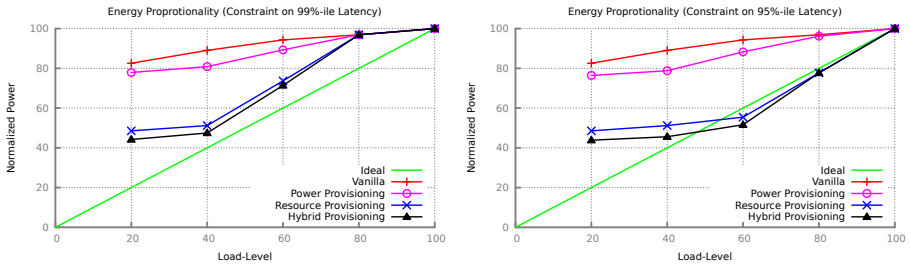

Fig. 4. Read Only Workload - Full System Energy Proportionality

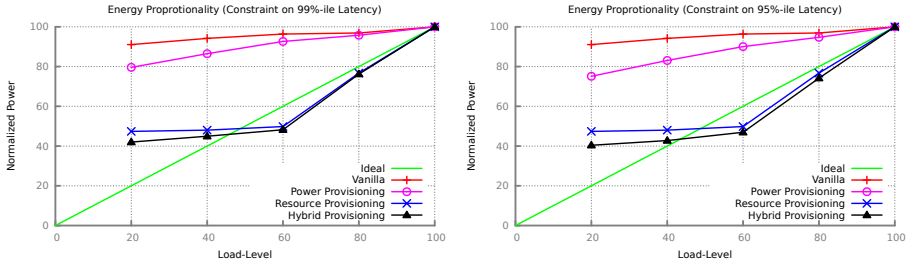

Fig. 5. Update Only Workload - Full System Energy Proportionality

then find the best possible CPU power limit on those nodes that satisfy the SLOs.

4.1 Energy Proportionality

Our main goal in this section is to understand the effects of the different power-management techniques on the energy proportionality of the system. Figures 4 and 5 show the effects of different power-management techniques on the energy proportionality of the read-only and update-only workloads, respectively, under different SLO targets. Energy proportionality is improved in every case.

Power provisioning is the least effective technique. However, it still saves power even at low load-levels. Resource provisioning and hybrid provisioning achieve better than energy-proportional operation at certain load-levels for both the workloads. Resource provisioning in certain cases provides higher energy-proportionality improvements when the SLO target is relaxed. For example at 80% load-level in the read-only workload case, better energy proportionality is achieved when the SLO is changed from the 99th percentile to the 95th. In this case, we achieve better energy proportionality because the 95th-percentile SLO can be maintained with only three nodes when compared to the four nodes used for satisfying the SLO for the 99th percentile.

We quantify energy proportionality using the energy-proportionality (EP) metric [15]. The EP metric is calculated, as shown in Equation (1), where $Area_{System}$ and $Area_{Ideal}$ represent the area under the system and ideal power

curve, respectively. A value of 1 for the metric represents an ideal energy-proportional system. A value of 0 represents a system that consumes a constant amount of power irrespective of the load-level. A value greater than 1 represents a system which is better than energy proportional.

$$EP = 1 - \frac{Area_{System} - Area_{Ideal}}{Area_{Ideal}} \tag{1}$$

Figure 6 shows the EP metric for the power-management techniques under different SLOs. In general, the power management techniques under evaluation improve the energy proportionality of the update-only workload better than the read-only workload. In certain cases for the update-only workload, EP > 1 is achieved.

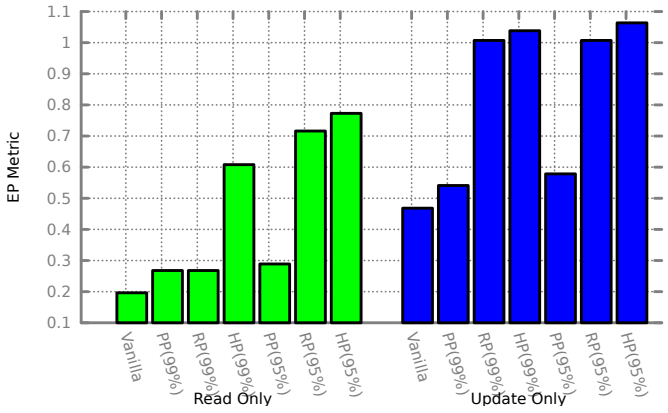

Fig. 6. EP Metric (PP = Power Provisioning, RP = Resource Provisioning, HP = Hybrid Provisoning)

4.2 Power Savings

Figure 7 shows the power savings resulting from the different power-management techniques. The savings range from 5% to 45% for the read-only workload and 15% to 55% for the update-only workload. In each case, hybrid provisioning provides the most power savings, but it is only marginal power savings over resource provisioning. We also observe that if the same number of nodes are used, relaxing the SLO target only provides marginal power savings. For example, power savings in the case of power provisioning under both the SLOs for the workloads provide similar power savings.

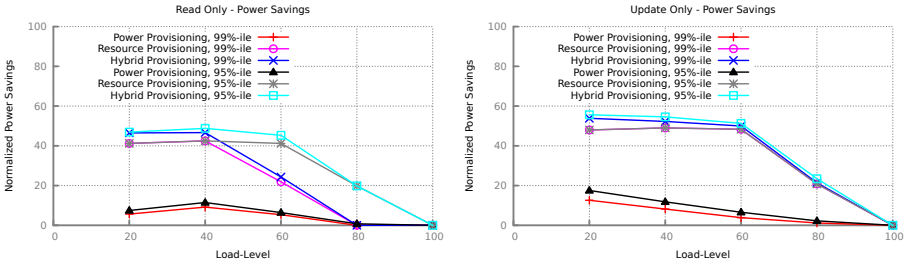

Fig. 7. Power Savings

5 Related Work

Dimitris et al. [20] provide a comprehensive study of component-level power consumption of relational databases on a single node. They analyze the energy efficiency of database servers using different hardware and software knobs such as CPU frequency, scheduling policy and inter-query parallelism. They conclude that the most energy-efficient operating point is also the highest performing configuration. Willis et al. [12] study the trade-offs between performance scalability and energy efficiency for relational databases. They identify hardware and software bottlenecks that affect performance scalability and energy efficiency. In addition, they provide guidelines for energy-efficient cluster design in the context of parallel database software. Our research complements theirs by addressing with the energy proportionality of non-relational (a.k.a. NoSQL) databases.

In our previous work [19,18], we studied the effects of RAPL power limiting on the performance, energy proportionality and energy efficiency of enterprise applications. We also designed a runtime system to decrease the energy proportionality gap. To design this runtime system, we used a load-detection model and optimization framework that uses statistical models for capturing the performance of an application under power limit. Wong et al. [22,21] provide an infrastructure for improving the energy proportionality using server-level heterogeneity. They combine a high-power compute node with a low-power processor essentially creating two different power-performance operation regions. They save power by redirecting requests to the low-power processor at low request rates thereby improving energy proportionality. In addition, they compare cluster-level packing techniques (resource provisioning) and server-level low power modes to identify if one of these technique is better with current generation of processors. Fan et al. [10] study the improvements to peak power consumption of a group of servers due to the improvements in non-peak power efficiency using their power model. They provide analytical evidence that shows energy-proportional systems will enable improved power capping at the data-center level. In this paper, we complement the existing literature by studying the effects of power and resource provisioning on the energy proportionality of a NoSQL cluster installation. Our

paper is also the first step towards a runtime system for power management of such installations.

Deng et al. [7,8,9] propose the CoScale framework, which dynamically adapts the frequency of the CPU and memory while respecting a certain application performance degradation target. They also take per-core frequency settings into account. Li et al. [13] study the CPU microarchitectural adaptation and memory low-power states to reduce energy consumption of applications bounding the performance loss by using a slack allocation algorithm. Sarood et al. [16] present an interpolation scheme to optimally allocate power for CPU and memory subsystems in an over-provisioned high-performance computing cluster for scientific workloads. This paper deals with improving energy efficiency of the compute nodes across different levels of utilization (and not just at the peak utilization levels) as data centers running even well-tuned applications spend a significant fraction of their time below peak utilization levels [4,14,10].

6 Conclusion

In this paper, we analyze the power distribution and energy proportionality of a distributed NoSQL data store. We find that the idle power in such an installation is significant, and most of the power is consumed by the CPUs when the system is under load. We apply different power-management techniques to the cluster supporting the distributed NoSQL data store in order to investigate whether we can trade latency for power at low utilization of the cluster by taking advantage of the difference between the measured latency and SLO. Our results show that our hybrid-provisioning technique delivers the most power savings (up to 55%), followed by resource provisioning.

References

1. Apache Cassandra. Available at `http://cassandra.apache.org/`.
2. Intel 64 and IA-32 Software Developer Manuals - Volume 3. `www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html`.
3. Yahoo Cloud Serving Benchmark (YCSB). `https://github.com/brianfrankcooper/YCSB/wiki`.
4. L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 40(12), 2007.
5. F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *ACM Transactions on Computer Systems*, 2008.
6. H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le. RAPL: Memory Power Estimation And Capping. In *International Symposium on Low Power Electronics and Design*, ISLPED, 2010.
7. Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini. CoScale: Coordinating CPU and Memory System DVFS in Server Systems. In *International Symposium on Microarchitecture*, MICRO, 2012.

8. Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini. MultiScale: Memory System DVFS with Multiple Memory Controllers. In *International Symposium on Low Power Electronics and Design*, ISLPED, 2012.
9. Q. Deng, D. Meisner, L. Ramos, T. F. Wenisch, and R. Bianchini. MemScale: Active Low-Power Modes for Main Memory. 2011.
10. X. Fan, W.-D. Weber, and L. A. Barroso. Power Provisioning For a Warehouse-Sized Computer. In *International Symposium on Computer Architecture*, ISCA, 2007.
11. A. Lakshman and P. Malik. Cassandra: A Decentralized Structured Storage System. *SIGOPS Operating Systems Review*, 2010.
12. W. Lang, S. Harizopoulos, J. M. Patel, M. A. Shah, and D. Tsirogiannis. Towards Energy-Efficient Database Cluster Design. *arXiv:1208.1933 [cs]*, Aug. 2012.
13. X. Li, R. Gupta, S. V. Adve, and Y. Zhou. Cross-Component Energy Management: Joint Adaptation of Processor and Memory. *ACM Transactions on Architecture and Code Optimization*, 2007.
14. 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 *International Symposium on Microarchitecture*, MICRO, 2011.
15. F. Ryckbosch, S. Polfliet, and L. Eeckhout. Trends in Server Energy Proportionality. *IEEE Computer*, (9):69–72, 2011.
16. O. Sarood, A. Langer, L. Kale, B. Rountree, and B. Supinski. Optimizing Power Allocation to CPU and Memory Subsystems in Overprovisioned HPC Systems. In *Proceedings of IEEE Cluster*, 2013.
17. S. Sivasubramanian. Amazon dynamoDB: A Seamlessly Scalable Non-relational Database Service. In *In Proceedings of the International Conference on Management of Data*, SIGMOD, 2012.
18. B. Subramaniam and W. Feng. Towards Energy-Proportional Computing for Enterprise-Class Server Workloads. In *Proceedings of the International Conference on Performance Engineering*, ICPE, 2013.
19. B. Subramaniam and W. Feng. Enabling Efficient Power Provisioning for Enterprise Applications. In *Proceedings of the International Symposium on Cluster, Cloud and Grid Computing*, CCGRID, 2014.
20. D. Tsirogiannis, S. Harizopoulos, and M. A. Shah. Analyzing the Energy Efficiency of a Database Server. In *Proceedings of the International Conference on Management of Data*, SIGMOD '10, 2010.
21. D. Wong and M. Annavaram. KnightShift: scaling the energy proportionality wall through server-level heterogeneity. In *Proceedings of the International Symposium on Microarchitecture*, MICRO, 2012.
22. D. Wong and M. Annavaram. Implications of High Energy Proportional Servers on Cluster-Wide Energy Proportionality. In *Proceedings of the International Symposium on High Performance Computer Architecture*, HPCA, 2014.