Resource Efficient Computing for Warehouse-scale Datacenters

Christos Kozyrakis
Computer Systems Laboratory
Stanford University
kozyraki@stanford.edu

***Abstract*—An increasing amount of information technology services and data are now hosted in the cloud, primarily due to the cost and scalability benefits for both the end-users and the operators of the warehouse-scale datacenters (DCs) that host cloud services. Hence, it is vital to continuously improve the capabilities and efficiency of these large-scale systems. Over the past ten years, capability has improved by increasing the number of servers in a DC and the bandwidth of the network that connects them. Cost and energy efficiency have improved by eliminating the high overheads of the power delivery and cooling infrastructure. To achieve further improvements, we must now examine how well we are utilizing the servers themselves, which are the primary determinant for DC performance, cost, and energy efficiency. This is particularly important since the semiconductor chips used in servers are now energy limited and their efficiency does not scale as fast as in the past. This paper motivates the need for resource efficient computing in large-scale datacenters and reviews the major challenges and research opportunities.**

I. Introduction

Computing has become an essential tool and a catalyst for innovation in all aspects of human endeavor, including healthcare, education, science, commerce, government, and entertainment. An increasing amount of computing is now performed in the cloud [1]. Warehouse-scale datacenters (DCs) host popular online services such as search, social networking, webmail, video streaming, enterprise management tools, online maps, automatic translation, big-data analytics, open online courses, and general-purpose cloud computing and storage platforms. We have come to expect that these services provide us with instantaneous, personalized, and contextual access to petabytes of data.

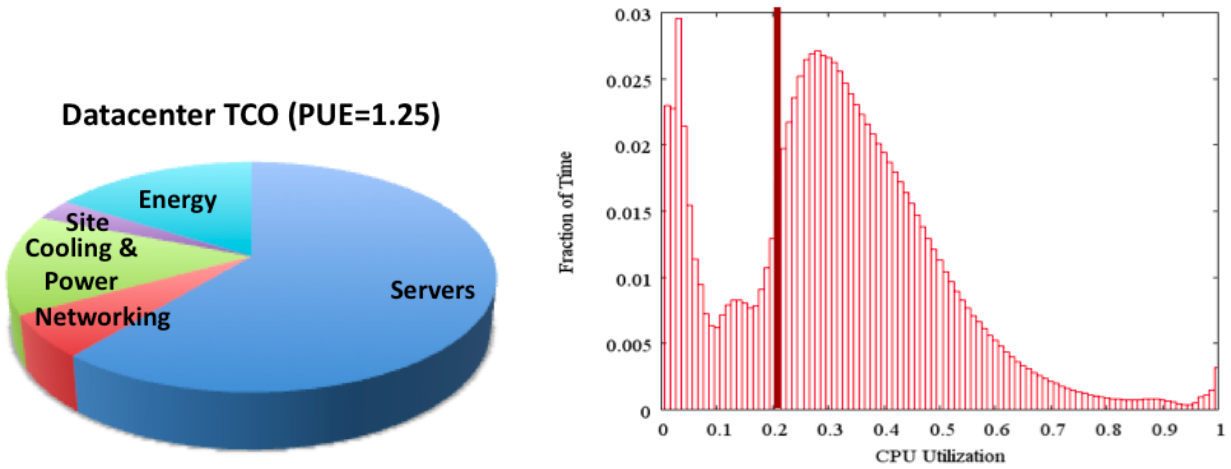
The primary advantages of cloud computing are the *scalability and cost benefits* for both the end-users and the DC operators. In the past ten years, operators have scaled the capabilities of cloud services by building larger datacenters that can host tens to hundreds of thousands of multi-core servers [2]. The servers are connected by networks with high-speed links (e.g., 10Gbps Ethernet) and advanced topologies that support high bandwidth between any two servers [3], [4]. At the same time, operators have increased cost effectiveness by using commodity, lower-cost servers and by reducing the cost and energy overheads of the power delivery and cooling infrastructure [2]. While a few years ago the power usage effectiveness (PUE) of datacenters was as high as 3.0, the PUE of modern facilities is as low as 1.1.

978-3-9815370-0-0/DATE13/©2013 EDAA

Unfortunately, we have reached the end of the road for these scaling techniques. Datacenters are already consuming tens of MWatts, stressing the capabilities of power generation facilities and making it difficult to continuously increase the number of servers per datacenter [2], [5]. At the same time, the end of voltage scaling for the semiconductor chips used in servers means that we cannot rely on chips to offer exponentially increasing performance at constant energy consumption [6]–[8]. Both server chips and DC facilities are energy limited. Building additional datacenters is also expensive as each new facility costs hundreds of millions of dollars [2]. Finally, PUE improvements have reached the point of diminishing returns as well, since the overhead of power delivery and cooling can be reduced to 10% over the power consumption of the servers.

To achieve further improvements in DC capability and cost effectiveness, we must now focus on the following two questions [9], [10]. First, *are we using efficiently the servers available in these large-scale datacenters*? Second, *are we building the right servers to begin with*? Figure 1 shows the breakdown of the total cost of ownership (TCO) of a DC facility with $PUE = 1.25$ [5], [11]. The capital expenses for procuring the servers represent 61% of TCO, while the energy consumed by servers is responsible for another 16%. Figure 1 also shows the distribution of CPU utilization in DC servers [1]. Utilization is quite low, typically ranging at 10% to 20%. Hence, an obvious path towards improving both capability and cost efficiency is to increase the utilization of DC servers. High server utilization is also beneficial for energy efficiency. Since most servers are not energy proportional, consuming 40% to 60% of their peak power when idling, they operate most efficiently at high utilization [1], [9], [12]–[14].

Nevertheless, there are several challenges towards achieving resource efficiency through higher server utilization. First, DC operators must plan for diurnal usage patterns, unexpected spikes in user demands, and new workloads. Second, it is impossible to configure servers so that per machine resources (cores, memory, storage, I/O) perfectly match the requirements of all cloud services hosted in a particular facility [10]. Third, and most important, increasing server utilization by scheduling



(a) (b)

Fig. 1. (a) Breakdown of the total cost of ownership (TCO) of a 10MW datacenter assuming 3-year server amortization and 15-year site amortization [5], [11]. (b) The distribution of CPU utilization of 5,000 Google servers over a period of 6 months [1].

multiple services on each server leads to *performance loss due to interference*. Even if they use different processor cores, co-scheduled applications can interfere on other shared resources, such as caches, memory channels, storage and networking devices [15]–[17].

Interference is particularly detrimental for *latency-critical, user-facing services with strict quality-of-service (QoS) guarantees*. For instance, updating a social network's news feed involves queries for the user's connections and recent postings; ranking, filtering, and formatting updates; retrieving media files; and selecting and formatting relevant advertisements and recommendations. Since tens of servers are involved in each user query, low average latency is not sufficient. The requirement is for low tail latency (e.g., low 95th or 99th percentile), so that latency variability does not impact a significant percentage of user requests. Assigning further workloads to each server in order to raise utilization typically leads to higher latency and higher variability since requests may be queued for milliseconds if other tasks occupy processing cores. But even if cores are available, the latency-critical requests may underperform due to interference and contention on shared resources such as caches, memory, storage, and networking. Hence, it is common for latency-critical services to be deployed on dedicated servers, which are underutilized for the majority of time.

II. Resource Efficient Computing

We set as our goal to *increase the capability and cost effectiveness of large-scale datacenters by improving their resource efficiency*. Specifically, we want to reconcile the apparent conflict between the need to maintain high quality-of-service (QoS) for latency-critical, high-priority services and the desire to increase hardware utilization by scheduling multiple workloads per server. Towards this goal, we need advances in both *isolation mechanisms* that reduce interference between co-scheduled services and *scheduling policies* that manage resource usage and isolation mechanisms to increase resource utilization without sacrificing QoS guarantees. The combination of strong isolation mechanisms and proper scheduling policies can allow us to increase utilization for a wide range of operational scenarios. For instance, during periods of low or medium user traffic, we can schedule background analytics tasks on servers provisioned for latency-critical services without an impact on their QoS. Similarly, we can run low-latency storage services on any server to export unused memory or SSD capacity to workloads running on other servers that are resource constrained. Finally, we can aggressively pack workloads on cloud computing servers without performance and efficiency losses due to interference.

A significant improvement in resource efficiency requires a coordinated, *cross-layer research approach* that spans hardware architecture, operating systems, and cluster management. Architecture research can provide the basic mechanisms for efficient sharing and isolation of hardware resources including cores, caches, memory channels, networking and storage devices. Moreover, it can lead to new server, rack, or cluster organizations that lend themselves to higher degrees of resources sharing between concurrent workloads. Operating systems research can provide algorithms that manage the granularity of resource sharing between workloads and control hardware isolation mechanisms. Cluster management research is necessary because most cloud services use multiple tiers to process any user request. Moreover, the cluster scheduler determines how services are distributed across the available servers. It is vital to understand the interactions and coordinate isolation mechanisms and scheduling policies across the system layers reviewed in the following subsections.

A. Datacenter Workload Analysis

While there are numerous studies of cloud workloads, relatively few have focused on interference and resource usage issues [15], [18], [19]. Early studies have shown that the

performance of Google workloads varies by up to 40% due to cache and memory interference [18]. We have measured even larger performance drops due to interference on latency-critical, memory-based storage services. Nevertheless, we need a stronger understanding of the issues to drive hardware and software research on resource efficiency. Specifically, we need to understand the impact of varying levels of interference on hardware and software resources with respect to performance and QoS guarantees. We must also quantify the relationship between performance and resource allocation within and across servers. Finally, we must analyze whether application-level techniques for load balancing and fault tolerance can also mitigate interference and performance variability.

B. Hardware Isolation Mechanisms

At the architecture level, we need isolation and priority mechanisms for hardware resources, such as multithreaded cores, private and shared caches, memory channels, network interfaces, and high-speed storage devices. Unlike existing mechanisms for hardware QoS that focus primarily on fairness, these new mechanisms must support fine-grain resource allocation between multiple latency-critical and throughput tasks of various priority levels and must have reaction times at the microsecond level in order to be useful for latency-critical services. They must provide strict guarantees on resource allocation, since best effort schemes introduce the risk of significant interference in unexpected workloads scenarios. Finally, the control interface for hardware mechanisms must be compatible with the scheduling, priority, and allocation abstractions used by the server OS and the cluster manager.

Recent research has produced promising results towards these goals. Scalable partitioning techniques can enable an arbitrary number of workloads to share caches with strict guarantees on partition sizes and performance [20]. QoS mechanisms can define classes of service and priority for the use of cache and memory bandwidth [21]. Emerging network adapters can support hundreds of send and receive queues that the OS can associate with different workloads or connections. Under software control, the adapter can enforce priorities and rate limits on each queue. We need to extend these concepts across all hardware resources, study implementation alternatives and their respective cost, and define a uniform set of APIs that allow software control of isolation and priority mechanisms.

C. OS Support for Latency-critical Services

At the OS level, we need to enhance the scheduling algorithms for the sharing scenarios that latency-critical services might encounter. Decades of OS development have led to scheduling algorithms that fully utilize processing cores through multitasking of throughput-bound workloads, while providing human users with fluid-appearing interactions. Unfortunately, these algorithms are tuned for human visual sensitivity, which is in the order of several milliseconds. Multi-tier, latency-critical services require that individual servers respond within hundreds or even tens of microseconds.

Hence, we need to examine the amenability of conventional and real-time scheduling algorithms for such workloads, properly expand and tune their parameters, or develop alternative algorithms that address any shortcomings. Our early experiments suggest that neither the fair scheduling nor the real-time scheduling algorithms in modern operating systems can efficiently handle all sharing scenarios in the presence of latency-critical workloads. Their primary shortcoming is the lack of decoupling between latency and throughput guarantees. Moreover, the OS scheduler must manage not just processing cores but the allocation of all hardware resources in a coordinated manner. Finally, to make informed scheduling decisions at the cluster level or at the various application tiers, we need new interfaces that allow the OS in each server to communicate resource allocation, interference, and QoS information.

D. Cluster-level Management

Managing services in a large datacenter has several challenges. First, resource requirements within and across services are very dynamic. The load of user-facing applications, such as search, or webmail can vary widely within a day. Cloud computing frameworks, such as EC2, must also deal with previously unknown workloads submitted at high rates, making it difficult to make a priori decisions for resource use. In environments where services express resource reservations, it is common for developers to exaggerate reservations by integer factors in order to side-step interference issues. Hence, the cluster manager rarely knows the true needs of a service at admission point. Second, services have different structures, priorities and require different latency and performance constraints. Third, each service exhibits different degrees of sensitivity to interference on shared resources by co-scheduled workloads and can similarly inflict different amounts of interference to them. Fourth, since datacenters are provisioned over 15-year periods with servers gradually installed and replaced [2], there is significant heterogeneity in server configuration, further complicating resource efficient cluster management.

Given these challenges, there are three questions that the cluster manager must address. First, *"how many resources" should the application be allocated*, i.e., what are the real resource requirements the workload needs to preserve its QoS constraints? Second, *"where" should an application be scheduled*, meaning, which of the tens of thousands of shared servers are the most appropriate for the workload's demands given the current system state and available resources? Third, *"when" should the application be scheduled* if its QoS constraints allow scheduling delays or execution with fewer than ideal resources?

In recent work, we showed that we can address the second question ("where") in the presence of unknown workload needs, varying interference sensitivity, and heterogeneous server configurations [22]. We developed the Paragon framework that, first, performs application classification to understand the workloads' interference characteristics and how well they behave on the different server configurations.

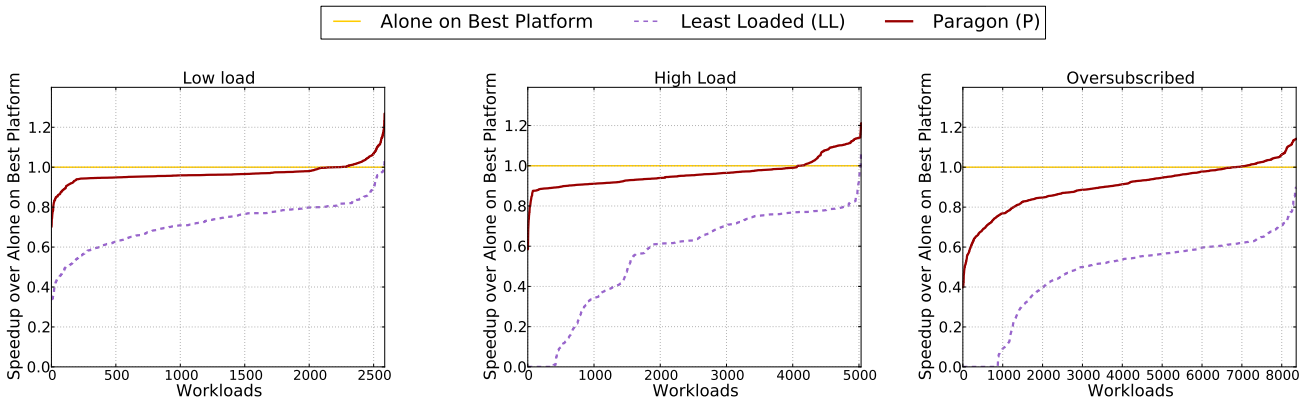


Fig. 2. Performance comparison when scheduling workloads on 1,000 shared servers with 14 hardware configurations [22]. Performance is normalized to executing each workload in isolation on the best possible server (best case QoS). The (LL) scheduler packs workloads to the least-loaded server at each point taking into account their core and memory requirements. Paragon uses its classification engine to extract interference and heterogeneity information. All classification and scheduling overheads are included. The three scenarios shown differ in workload number and arrival rates. Experiments were conducted using exclusive, reserved, EC2 servers.

Unlike previous work that uses time-consuming, offline per-application profiling to analyze interference and heterogeneity requirements, the classification engine in Paragon operates like a recommendation engine for online shopping. Given a minute's worth of profiling data from a couple of servers, Paragon uses collaborative filtering methods, such as Singular Value Decomposition (SVD) and PQ-reconstruction, to compare the current workload to previously scheduled workloads and derive detailed and accurate information about its interference and heterogeneity characteristics. Figure 2 shows the improvements in resource efficiency when using Paragon to schedule thousands of workloads on 1,000 EC2 servers (SC4 sharing scenario). Compared to scheduling workloads accounting only for the core and memory requirements (LL lines), Paragon improves workload performance by 25% to 4x by assigning them to servers in a manner that matches workloads to server configurations and minimizes interference between co-scheduled applications. The gains from Paragon are bigger when the cluster is under heavy load (higher workload arrival rates). Faster execution times with Paragon imply that application QoS requirements are met. Moreover, they imply that resources are used and released faster. A subset of the cluster is sufficient for the given workloads and the remaining resources can be used for additional workloads of any type (improved capability at constant TCO).

Further research is necessary to address the other two questions for cluster management. The classification-based approach can be useful for resource allocation as well, providing valuable insights during application development and application deployment. To properly manage shared resources for all scenarios, some additional infrastructure is necessary. We need APIs and mechanisms to communicate performance and QoS information between the OS, the cluster manager, and the different service tiers. For instance, if the hardware, the OS, or the application itself observes behavior different from what the classification predicted, we may either have a classification error or a change in workload behavior (phase change or change in user load). In any case, the cluster manager must be notified to take further scheduling actions. Similarly, if a language runtime system, such as the JVM, is about to perform expensive garbage collection or if a certain tier observes its queue getting larger, they should pass this information to schedulers and load balancers to avoid performance variability. Finally, for some workload scenarios, we may need to rate limit the additional workload so that it utilizes spare resources without exceeding certain bounds.

E. Resource Efficient Hardware

In addition to increasing server utilization, we can achieve further improvements in resource efficiency by revisiting the basic structure of DC servers and the components these servers employ.

At the macro level, blade and microserver enclosures have the potential for significant improvements over the commonly used, two-socket servers [23], [24]. Using high-bandwidth, low-latency interconnect fabrics, these enclosures enable tens of processor chips to share resources such as memory, storage devices, and network adapters. Resource sharing leads to reduced component counts, better component utilization, better amortization of expensive components (e.g., DRAM or Flash memory), and new opportunities for load balancing and power management. The optimal balance and resource management policies for such enclosures is still an open question.

At the micro level, there is an active debate on the use of wimpy cores in DC chips. While simple cores offer more performance per unit of area and power compared to the high-end cores in current servers, they lag in single-thread performance and often cause QoS and performance variability problems for latency-critical workloads [25], [26]. Nevertheless, there are many opportunities for resource efficiency gains, regardless of the choice of cores. Custom accelerators for common tasks can provide large performance increases at small power and area overheads. Some server chips and I/O controllers already

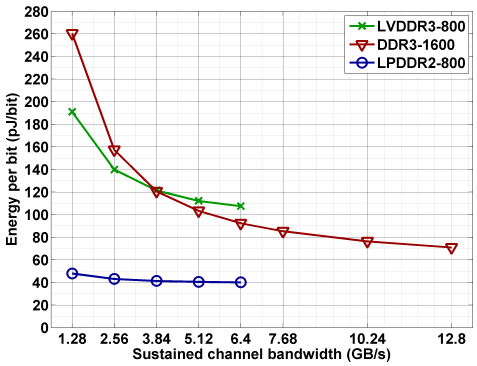


Fig. 3. Energy per accessed bit with varying channel utilization for DDR3, low-voltage DDR3, and LPDDR2 technologies. We assume four ranks per memory channel and 3:1 read to write operations ratio.

provide specialized accelerators for low-level tasks, such as encryption and compression. As communication, storage, and coordination protocols in DC frameworks are getting standardized, we will likely see specialized accelerators for higher-level functionality as well. The accelerators may be integrated in the processor chips or in separate I/O controllers, shared within a larger enclosure.

It is also critical to boost the efficiency of memory systems, which have been improving at a significantly slower pace compared to processing systems. With online services demanding fast access to ever increasing datasets, memory accounts for 25% to 40% of datacenter energy consumption and a large fraction of the total cost. The high-bandwidth interfaces of the commonly-used DDR3 technology consume significant energy even when memory is idle or lightly utilized. In recent work, we observed that emerging DC workloads exhibit capacity and bandwidth demands that do not match the features of DDR3 technology [10]. Applications, such as web search, MapReduce, and distributed memory caching stress memory capacity and latency but not bandwidth. Hence, we turned to memory technology originally designed for mobile platforms, LPDDR2. Mobile-class memory forgoes the expensive interface circuitry of DDR3, sacrificing some of the peak bandwidth offered for a 2x – 5x energy improvement over DDR3 technology across all utilization points, as shown in Figure 3. We used stacked LPDDR2 devices to architect high capacity memory systems for DC servers that reduce memory power consumption by 3x – 5x and impose negligible performance penalties for latency-critical workloads [27]. More important, the LPDDR2-based memory systems lead TCO improvements of up to 30% for large-scale DC facilities.

III. Conclusion

We can improve the capability and cost effectiveness of online services by systematically increasing the resource utilization in large-scale datacenters. Queuing theory tell us that when resource utilization approaches 100%, the impact on latency will be severe. However, given the current utilization levels of 10%-20%, there is significant room for improvement. A cross-layer approach that spans hardware architecture, operating systems, and cluster management can lead to resource efficiency gains of 2x – 5x across a wide range of operational scenarios and contribute towards the goal of improving the scalability and cost effectiveness of warehouse-scale computing.

Acknowledgment

The author would like to thank Adam Belay, Christina Delimitrou, Jacob Leverich, and David Lo at Stanford University for contributing to the ideas discussed in this manuscript.

References

[1] U. Hoelzle and L. Barroso, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 2009.

[2] J. Hamilton, "Internet-Scale Service Infrastructure Efficiency," in *Proceedings of the 37th Intl. Symposium on Computer architecture*, Jun. 2009.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the ACM SIGCOMM Conference on Data communication*, Aug. 2008.

[4] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a Scalable and Flexible Data Center Network," *Communications of the ACM*, vol. 54, no. 3, pp. 95–104, Mar. 2011.

[5] K. Vaid, "Datacenter Power Efficiency: Separating Fact from Fiction," Invited talk at the 2010 Workshop on Power Aware Computing and Systems, Oct. 2010.

[6] M. Horowitz, "Scaling, Power and the Future of CMOS," in *Proceedings of the 20th Intl. Conference on VLSI Design*, Jan. 2007.

[7] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark Silicon and the End of Multicore Scaling," in *Proceedings of the 38th annual Intl. Symposium on Computer Architecture*, Jun. 2011.

[8] S. Keckler, "Life After Dennard and How I Learned to Love the Picojoule," Keynote at the he 44th Intl. Symposium on Microarchitecture, Dec. 2011.

[9] L. Barroso, "Warehouse-Scale Computing: Entering the Teenage Decade," in *Proc. of the 38th Intl. Symposium on Computer Architecture*, Jun. 2011.

[10] C. Kozyrakis, A. Kansal, S. Sankar, and K. Vaid, "Server Engineering Insights for Large-Scale Online Services," *IEEE Micro*, vol. 30, no. 4, pp. 8–19, Jul. 2010.

[11] J. Hamilton, "Cost of Power in Large-Scale Data Centers," http://perspectives.mvdirona.com.

[12] L. Barroso and U. Hölzle, "The Case for Energy-Proportional Computing," *IEEE Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[13] J. Leverich and C. Kozyrakis, "On the Energy (in)efficiency of Hadoop Clusters," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 1, pp. 61–65, Mar. 2010.

[14] D. Meisner, C. M. Sadler, L. A. Barroso, W.-D. Weber, and T. F. Wenisch, "Power ¡anagement of Online Data-intensive Services," in *Proceedings of the 38th Annual Intl. Symposium on Computer Architecture*, Jun. 2011.

[15] S. Govindan, J. Liu, A. Kansal, and A. Sivasubramaniam, "Cuanta: Quantifying Effects of Shared on-chip Resource Interference for Consolidated Virtual Machines," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, Oct. 2011.

[16] 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 Intl. Symposium on Microarchitecture*, Dec. 2011.

[17] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-Clouds: Managing Performance Interference Effects for QoS-Aware Clouds," in *Proceedings of of the EuroSys Conference*, Apr. 2010.

[18] L. Tang, J. Mars, N. Vachharajani, R. Hundt, and M. L. Soffa, "The Impact of Memory Subsystem Resource Sharing on Datacenter Applications," in *Proceedings of the 38th annual Intl. Symposium on Computer Architecture*, Jun. 2011.

[19] J. Mars, L. Tang, and R. Hundt, "Heterogeneity in "Homogeneous"; Warehouse-Scale Computers: A Performance Opportunity," *IEEE Comput. Archit. Lett.*, vol. 10, no. 2, pp. 29–32, Jul. 2011.

[20] D. Sanchez and C. Kozyrakis, "Scalable and Efficient Fine-Grained Cache Partitioning with Vantage," *IEEE Micro's Top Picks from the Computer Architecture Conferences*, vol. 32, no. 3, May-June 2012.

[21] R. Iyer, L. Zhao, F. Guo, R. Illikkal, S. Makineni, D. Newell, Y. Solihin, L. Hsu, and S. Reinhardt, "QoS Policies and Architecture for Cache/memory in CMP Platforms," in *Proceedings of the 2007 Intl. Conference on Measurement and mMdeling of Computer Systems*, Jun. 2007.

[22] C. Delimitrou and C. Kozyrakis, "Paragon: QoS-Aware Scheduling for Heterogeneous Datacenters," in *Proceedings of the 18th Intl. Conference on Architectural Support for Programming Languages and Operating Systems*, March 2013.

[23] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level Power Management for Dense Blade Servers," in *Proceedings of the 33rd Annual Intl. Symposium on Computer Architecture*, Jun. 2006.

[24] A. Dhodapkar, G. Lauterbach *et al.*, "SeaMicro SM10000-64 Server," in *The Technical Record of the 23rd Hot Chips Conference*, Aug. 2011.

[25] V. Reddi, B. Lee, T. Chilimbi, and K. Vaid, "Web search using mobile cores: Quantifying and mitigating the price of efficiency," in *Proceedings of the Intl. Symposium on Computer Architecture*, Jun. 2010.

[26] U. Holzle, "Brawny Cores Still Beat Wimpy Cores, Most of the Time," *IEEE Micro*, vol. 30, no. 4, Jul. 2010.

[27] K. T. Malladi, B. C. Lee, F. A. Nothaft, C. Kozyrakis, K. Periyathambi, and M. Horowitz, "Towards energy-proportional datacenter memory with mobile DRAM," in *Proceedings of the 39th Intl. Symposium on Computer Architecture*, Jun. 2012.