Autonomous Resource Sharing for Multi-Threaded Workloads in Virtualized Servers

Can Hankendi*
hankendi@bu.edu

Ayse K. Coskun*
acoskun@bu.edu

Electrical and Computer Engineering Department—Boston University

Abstract

Multi-threaded applications from many application domains have started to appear on the cloud resources. Multi-threaded applications bring additional challenges to the management of the cloud resources, due to characteristics such as inter/intra-thread communication. In tandem, as the energy spent on computing continues to increase, the ability to provide energy-performance tradeoffs has become essential for both users and the data center administrators.

This paper proposes an autonomous resource sharing technique for multi-threaded workloads with the goal of creating tunable energy cost-performance tradeoffs for cloud administrators and users. The proposed technique adjusts the resources allocated to each virtual machine (VM) based on the energy efficiency of the applications running on the VMs, while providing the desired performance guarantees. The success of the proposed technique is evaluated on commercial multi-core servers. The final results show that the proposed technique improves the energy efficiency of the virtualized servers by up to 21% in comparison to existing methods.

1. Introduction

Energy-related costs are among the major contributors to the total cost of ownership for today's data centers and high performance computing (HPC) clusters [9]. Thus, energy-efficient management of the cloud resources has become one of the main prerequisites of achieving sustainable computing. Due to the increasing number of servers installed in data centers, management of the cloud resources has become increasingly complex and costly [9]. Virtualization enables elastic management of a large number of physical resources and provides isolated execution environment for individual VMs. As a result, the number of virtualized servers has exceeded the number of native (not virtualized) servers [4].

In recent years, multi-threaded applications also start to emerge on cloud resources from various application domains, such as HPC applications (e.g., molecular dynamics) and scale-out applications (e.g., Hadoop). Multi-threaded workloads have several distinct characteristics that distinguish them from traditional data center workloads. Unlike most traditional data center loads, multi-threaded workloads highly utilize the servers, whereas traditional enterprise loads utilize the systems at lower levels [13]. Another challenge with multi-threaded workloads is the significance of application-specific needs and constraints while determining the resource allocation and consolidation strategies. Resource requirements exhibit significant variations across and within application domains. Therefore, runtime monitoring and adaptive techniques would enable (1) workloads to operate at more efficient settings, and (2) the administrators to offer users various levels of performance-energy cost pairs to operate on.

In this paper, we propose a novel, autonomous resource-sharing strategy for consolidating multi-threaded workloads on multi-core servers. While some of the prior approaches targeting multi-threaded loads select which applications to consolidate together for reducing resource contention (e.g., [1] [3]), we find that performance isolation in VMs reduces the significance of co-runner application selection. We argue that we can improve the overall server throughput and energy efficiency by shifting resources from applications that cannot utilize the processors efficiently towards applications that make more efficient use of the resources. As shrinking CPU resources might cause performance degradation, the proposed technique also enables the user to request performance guarantees for individual VMs, and meets the desired performance by utilizing a feedback mechanism to guide the resource allocation decisions.

Our technique uses runtime performance monitoring to identify application characteristics and to guide the resource allocation decisions by utilizing VM control knobs (i.e., CPU resource limits) and virtualized performance counters that are available on the ESXi hypervisor. We implement and demonstrate the benefits of our technique on two commercial multi-core based virtualized servers and show that our proposed technique improves the energy efficiency up to 21% in comparison to previously proposed consolidation strategies.

The rest of the paper is organized as follows. Section 2 presents our experimental methodology. Section 3 explains the proposed resource allocation technique. Section 4 discusses the experimental results, and Section 5 provides an overview of the related work. Section 6 concludes the paper.

* This work has been partially funded by VMware, Inc.

2. Methodology

We perform all experiments on two typical commercial servers found in data centers. One of the servers includes an AMD Opteron 6172 (Magny Cours) single-chip processor that comprises two 6-core dies attached side by side (i.e., 12 cores in total). Each core has a 512 KB private L2 cache. Each 6-core die has one local NUMA node and a 6 MB shared L3 cache. All cores share 16 GB off-chip memory. The other server is a multi-socket system that includes two 4-core Intel Xeon E5-2603 processors (i.e., 8 cores in total). Each core has 32 KB of private L1 and 256 KB of private L2 cache, each processor (i.e., 4 cores) shares 10 MB of L3 cache, and all processors (i.e., 8 cores) share 32 GB off-chip memory. We virtualize both servers with VMware ESXi™ 5.1 hypervisor and create VMs with Ubuntu Server 12.04 guest OS.

As our technique utilizes runtime performance monitoring to guide the resource allocation decisions, we utilize the virtualized performance counters that have become available with the latest release of the ESXi hypervisor. Virtualized performance counters enable measuring hardware events at each guest OS [17]. We monitor retired instructions, CPU cycles, and last level cache misses for each VM every second using the default performance counter monitoring tool, *perf*, at the guest OS level. We use *esxtop* to collect VM-level CPU usage data every 2 seconds. To be able to evaluate the energy efficiency of the server node, we measure the total system power by using a *Wattsup PRO* power meter with a 1-second sampling rate, which is the minimum sampling rate provided for this meter. As total system power determines the electricity cost of a server, we choose to evaluate system power rather than individual component power (i.e., processor, disk, etc.).

We run seven benchmarks from the PARSEC [8] multi-threaded benchmark suite (*blackscholes, bodytrack, canneal, dedup, streamcluster, vips* and *x264*) and three benchmarks from Cloudsuite [7] *(hadoop, cassandra* and *faban*) in our experiments as a representative set of multi-threaded workloads in the cloud. We run each benchmark with 12 threads using the native input sets.

Parallel applications typically consist of serial I/O stages and a parallel phase, i.e., region-of-interest (ROI). As parallel phases of the multi-threaded workloads occupy most of the compute cycles of the processors in real-life data centers, it is important to focus on the ROI of the parallel workloads. As the start and end points of the ROI phases vary across different applications, we implement a consolidation management interface, *consolmgmt*, that synchronizes the ROIs of the co-scheduled applications. We implement the ROI-Synchronization routine inside the existing PARSEC HOOKS library, which marks the start and end points of the application ROIs. The VM that first reaches the ROI phase sleeps until the second VM reaches its own ROI. The second VM sends interrupts upon reaching its own ROI to resume the execution and start data logging. We stop data collection and terminate the applications after one of the applications reaches the end of its ROI phase.

In order to evaluate our technique on a larger set of consolidation cases, we randomly generate 10-workload sets, each of which consists of 10 benchmarks that are chosen from seven PARSEC and three Cloudsuite benchmarks. For example, each workload set consists of five application pairs running on five servers when two VMs are consolidated at a time.

For performance evaluation, we use two metrics: (1) throughput (retired instructions per second) and (2) Application Heartbeats [6]. Throughput is an accurate metric to measure the application progress for the PARSEC and Cloudsuite benchmarks we experiment with. We also evaluate our technique using application-specific performance metrics, such as *frames-per-second* (fps) for an image processing applicationß by utilizing the *Application Heartbeats* framework.

3. Autonomous Resource Sharing under Performance Constraints

Resource allocation strategies have a significant impact on the energy efficiency of the server nodes. As applications utilize the available hardware resources in different ways, allocating equal resources for consolidated applications is not the most energy-efficient way to distribute the limited amount of hardware resources. In order to improve the energy efficiency of the system, we propose to proportionally allocate the resources depending on the energy efficiency level of each of the consolidated multi-threaded applications by utilizing the VM resource management knobs (i.e., CPU resource limits).

Figure 1 shows the throughput of the PARSEC and Cloudsuite benchmarks as a function of CPU resource limits for our AMD system. Depending on the application characteristics, the impact of the CPU resource limits on the performance varies significantly. Therefore, favoring the applications that are more efficiently utilizing the CPU resources improves the energy efficiency of the server nodes. For instance, allocating more resources to *vips*, when it is consolidated with *canneal*, would improve the throughput and the energy efficiency of the server, as *vips* utilizes the hardware resources more efficiently.

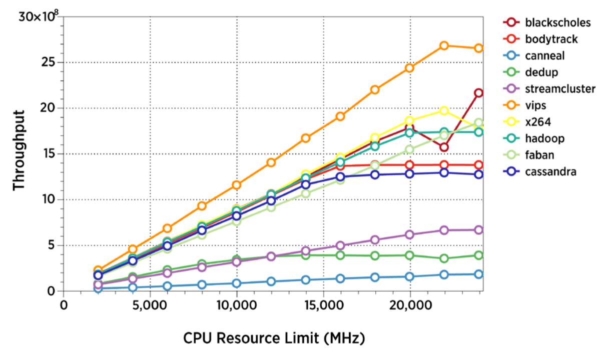

Figure 1. Throughput of the PARSEC and Cloudsuite benchmarks as a function of CPU resource limits for AMD Opteron based server. Throughput of the system exhibits significant variations depending on the application

Favoring applications that are more efficiently utilizing the available hardware resources might cause degradation on the performance of the co-runner VMs. Therefore, it is also essential to provide additional knobs to the users, such that the users can request performance guarantees. Our runtime policy continuously monitors the performance of the individual applications and utilizes a closed-loop controller to satisfy the user performance requirements.

3.1. Runtime Policy

The goal of our runtime policy is to allocate CPU resources proportionally with the throughput of each consolidated VM, for utilizing the available hardware resources more efficiently. In order to find the amount of CPU resources that should be allocated to a VM, we monitor the throughput of the applications using the virtualized performance counters. We distribute the total amount of available CPU resources, R_{tot}, by computing a weight for each VM, which is the ratio of the throughputs of the consolidated VMs. The amount of CPU resources, $\mathrm{R_i}$ that is allocated to VM_i is equal to $w_i * R_{tot}$. On the ESXi environment, the amount of computational capacity is represented in units of frequency (MHz), for each server, and R_{tot} is calculated by multiplying the number of cores and the frequency of each core. Therefore, we use two different R_{tot} values for our AMD and Intel systems. Our policy allows the users to set minimum throughput or Application Heartbeats constraints, which are then used in adjusting the resource allocation decisions.

3.2 Implementation of the Runtime Policy on VMware vSphere® Environment

We evaluate our autonomous resource sharing technique on Intel and AMD based servers. Our implementation consists of a management node (VMware vCenter™ terminal) and the virtualized server(s). Figure 2 shows the architecture of our implementation. We use

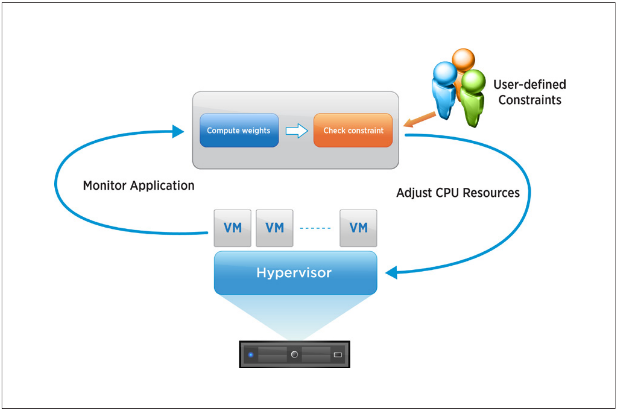

Figure 2. A feedback control mechanism monitors the performance of the consolidated applications and makes resource allocation decisions to improve the energy efficiency while meeting user-defined performance constraints.

an Intel i3 dual-core processor based machine as the vCenter management node. The runtime monitor polls the VM-level performance counter readings (i.e., retired instructions, clock cycles) every second; then, the vCenter management node makes resource allocation decisions and adjusts the CPU resource limits of the VMs. The resource allocation routine communicates with the ESXi through the vSphere SDK to perform administrative tasks (i.e., VM reconfiguration) on VMs [16]. For meeting the user performance constraints, we design a closed-loop feedback mechanism, which periodically evaluates the throughput losses and gains due to resource allocation decisions. We first store the *baseline throughput* for each application, which is the throughput when VMs are allocated equal amount of resources. We compare the current throughput of the application with the baseline throughput to compute the throughput gains and losses. We then adjust the resource allocations if the user performance constraints are violated.

3.3. Application Selection for Consolidation

As consolidating multiple VMs on the same physical resources causes increased resource contention, previous studies propose to reduce the resource contention by consolidating applications that have complementary characteristics (e.g., consolidating the most memory-intensive benchmark with the least memory-intensive one) [2]. However, on a highly isolated execution environment, the impact of co-runner VMs on each other's performance is minimal.

We utilize NUMA scheduling capabilities of the ESXi hypervisor [15] to create highly isolated execution environment to reduce the negative performance impact of the co-runner VMs. On the virtual environment, we dedicate one NUMA node to each VM to reduce the contention on the memory. We compare native and virtualized executions to investigate the impact of performance isolation. For all experiments on the native OS, we co-schedule two applications at a time, each of them running with six threads. For the virtual system, we create 12 vCPUs (12 threads) per VM and distribute the total CPU resources equally across the VMs. Figure 3 shows the

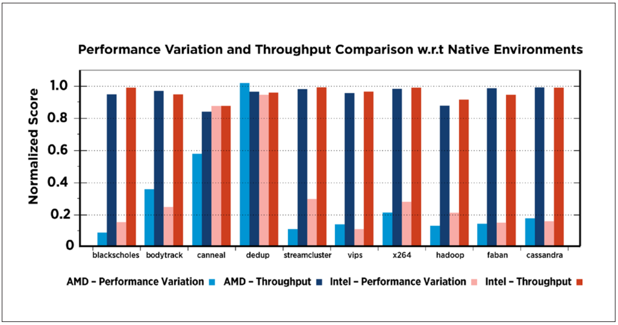

Figure 3. Performance isolation and throughput comparison normalized with respect to the results in native environment. Virtualized AMD and Intel servers provide better performance isolation (i.e., lower performance variation) and comparable throughput to the native case.

performance variation and throughput for the PARSEC and Cloudsuite benchmarks, when each application is consolidated with all the other benchmarks in pairs of two. We report the average of all consolidation pairs and normalize the performance variation and throughput with respect to the native environment. Higher performance variation implies that the performance of the application is significantly affected by the co-runner VM, and therefore indicates poor performance isolation. Note that, for consolidated servers, it is preferable to have low performance variation and high performance. For both Intel and AMD servers, virtualized environment provides 65% lower performance variation with less than 3% performance (i.e., throughput) degradation on average. These results are also in line with the findings of prior work [10]. We observe that the impact of the co-runner VMs is minimal for most of the benchmarks on a virtualized environment with high performance-isolation capabilities. Thus, the benefits of consolidation techniques that focus on selecting which VMs (i.e., applications) to co-schedule are expected to be limited for our environment running multi-threaded loads. As resource-sharing decisions have a bigger impact on the energy efficiency of the virtualized servers that provide highly isolated execution environments, utilizing the proposed resource sharing technique with application selection policies would further improve the energy efficiency.

4. Experimental Results

Our experimental work focuses on evaluating the proposed resource allocation technique for two main aspects, (1) ability to meet the performance constraints, and (2) energy efficiency improvements for randomly generated workload sets. We perform all of our experiments on two commercial multi-core servers, and we compare our technique with previously proposed consolidation techniques for randomly generated 10-workload sets, as explained in Section 3. In order to measure the energy efficiency of the server, we use throughput-per-watt, as it measures the useful work done per watt.

4.1. Runtime Behavior under Performance Constraints

We evaluate our technique under various performance constraints from the users. We evaluate both the throughput and an application-specific performance metric (i.e., frames-per-second (FPS)) for two image processing applications (i.e., *bodytrack* and *x264*). Figure 4 shows the runtime behavior of our technique under performance constraints. We show both the performance (top) and the resource adjustment decisions for individual VMs (bottom). On the AMD system, we consolidate two VMs that are running *blackscholes* and *dedup* and on the Intel system we consolidate *bodytrack* and *x264*. We test our technique with the minimum performance constraint of 70% for dedup, and 4 FPS for bodytrack. At t=14, the performance of *dedup* falls below the 70% of its baseline throughput, which is the throughput when all VMs are allocated equal resources. Thus, the resource allocation routine responds to the performance constraint violation by increasing the amount of resources allocated to *dedup*. Similarly, for *bodytrack*, resource allocation routine readjusts the CPU resource limits at t=36 to satisfy the performance constraints.

4.2 Energy Efficiency Evaluation for Workload Sets

We compare our technique with previously proposed consolidation techniques that target reducing the resource contention by matching application pairs that have complementary characteristics [1]. We evaluate two previously proposed metrics—memory operations per cycle (MPC) and instructions per cycle (IPC)—to guide the application selection policy. In order to choose the application pairs, we first rank the applications according to the selected metric (i.e., IPC or MPC), then pair the application at the top of the list with the application at the bottom of the list, and repeat the same process for the other applications in the list. This way, applications that have complementary characteristics are consolidated together, and we allocate equal resources to each VM. We report our results for the cases where we first use the application-selection policies to determine the application pairs, and then apply our resource-sharing policy at runtime to dynamically adjust the CPU resources allocated to each VM. Figure 5 shows the throughput-per-watt for various policies. We normalize the throughput-per-watt values with respect to the baseline case, where VMs have no CPU resource limits and consolidated VMs are selected randomly. Consolidating more

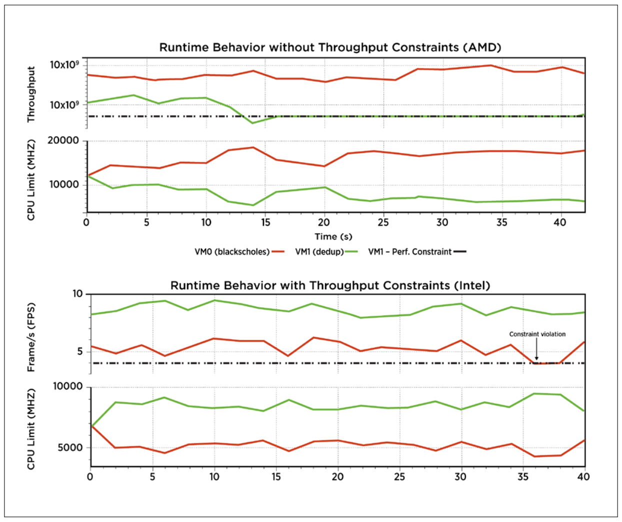

Figure 4. Runtime behavior of the proposed technique on two different systems under user-defined performance constraints.

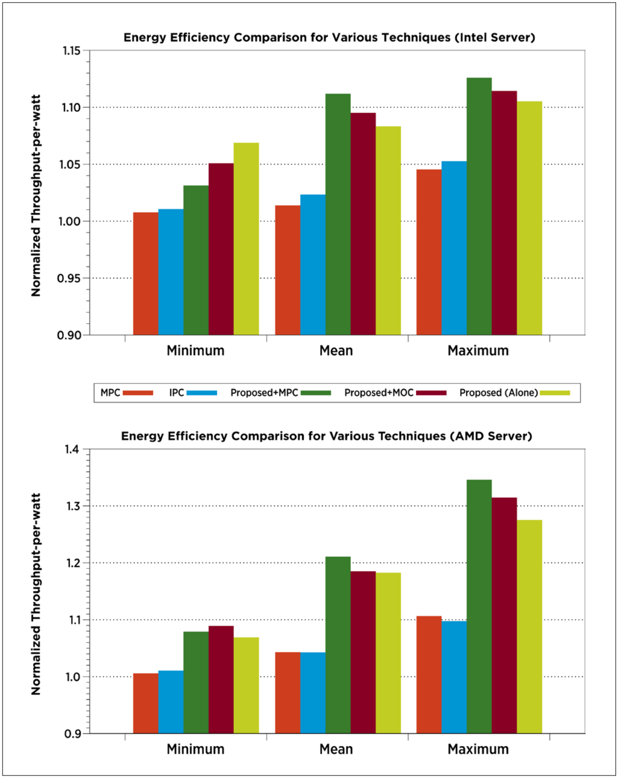

Figure 5. Energy efficiency comparison for various consolidation techniques. The proposed technique provides the highest improvements when jointly utilized with the application-selection policies on both Intel and AMD servers.

than two VMs causes significant performance degradation for multi-threaded applications when compared to running them alone. Therefore, we evaluate our technique for the case where we consolidate two applications at a time, which is a feasible design choice considering the performance requirements.

For randomly generated 10-workload sets, the proposed policy consistently improves the average throughput-per-watt of the system. Utilizing our technique jointly with application-selection policies further improves the energy efficiency. The energy efficiency improvements reach up to 21% for the AMD server and 9% for the Intel server. As the power range, which is the difference between the peak power consumption and the idle power, is much larger on the AMD server, the energy efficiency improvements are significantly higher than the Intel server.

5. Related Work

A number of the existing energy and resource management techniques target optimization of a larger-scale computing cluster (i.e., cluster-level management). Cluster-level resource management techniques can be mainly divided into two groups: VM migration and consolidation techniques. The goal of cluster-level resource management techniques is mainly to provide service availability guarantees and reduce the number of physical servers by turning off the idle nodes (e.g., [4] [5] [14]). VM resource management policies that utilize the history of the VMs to make resource allocation decisions have been also proposed to reduce the overhead of dynamic management [2].

Node-level (i.e., server-level) management techniques provide fine-granularity management for the data centers. Common consolidation techniques at the server level target reducing the resource contention by pairing contrasting applications or threads to work on the same physical resources (e.g., [1]). Application characteristics such as cache misses are utilized to guide the consolidation policies [3]. The success of these techniques relies on the fact that the co-runner VMs affect each other's performance, which implies poorly isolated execution environment for consolidated VMs. One line of work in consolidation focuses on reducing the impact of interference across consolidated applications [11] [12]. However, the interference impact varies depending on the application domain and the target architecture. In this work, we show that the recent advancements in virtualization technologies provide highly isolated execution environments, which limits the benefits of consolidation techniques for multi-threaded workloads. Our resource sharing technique utilizes multi-threaded application characteristics and guides the resource allocation decisions. In addition, the proposed technique can be utilized together with application-selection policies to further improve the energy efficiency.

6. Conclusions and Future Work

Energy efficiency is one of the major challenges for future data centers. The emergence of multi-threaded applications on the cloud resources introduces new challenges for energy-efficient management of hardware resources. In this work, we propose an adaptive solution to energy-efficient management of multi-core servers. The proposed technique takes the energy efficiency of the consolidated applications into account to adjust the amount of CPU resources that are allocated to each VM. We implemented our technique on a vCenter management node and evaluated two commercially available Intel- and AMD-based servers. Our results show that our policy can be jointly utilized with previously proposed consolidation techniques to further improve the energy efficiency by up to 21%.

Open problems in this area include designing scalable management techniques for larger-scale computing clusters, understanding the interference impact of applications from various application domains, and optimizing the performance of power-constrained systems. We plan to extend our work to be more applicable to larger-scale systems by introducing additional capabilities (e.g., VM migration) to our current technique. We also plan to investigate the performance-energy tradeoffs across a wider range of application domains.

References

1 G. Dhiman, G. Marchetti, and T. Rosing, "vGreen: A System for Energy-efficient Computing in Virtualized Environments," in *Proceedings of International Symposium on Low Power Electronics and Design (ISLPED)*, 2009, pp. 243–248.

2 N. Vasic, D. Novakovic, S. Miucin, D. Kostic, and R. Bianchini, "Dejavu: Accelerating Resource Allocation in Virtualized Environments," in *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2012, pp. 423–436.

3 M. Bhadauria and S. A. McKee, "An Approach to Resource-aware Co- scheduling for CMPs," in *ACM International Conference on Supercomputing*, 2010, pp. 189–199.

4 N. Bobroff, A. Kochut, and K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations," in *International Symposium on Integrated Network Management*, 2007, pp. 119 –128.

5 A. Beloglazov and R. Buyya, "Adaptive Threshold-based Approach for Energy-efficient Consolidation of Virtual Machines in Cloud Data Centers," in *International Workshop on Middleware for Grids, Clouds and e-Science*, 2010, pp. 1–6.

6 H. Hoffmann, S. Sidiroglou, M. Carbin, S. Misailovic, A. Agarwal, and M. Rinard, "Dynamic Knobs for Responsive Power-aware Computing," in ASLPOS, 2011, pp. 199–212.

7 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 *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2012, pp. 37–48.

8 C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *International Conference on Parallel Architectures and Compilation Techniques (PACT)*, October 2008.

9 M. Bailey, "The Economics of Virtualization: Moving Toward an Application-Based Cost Model," *International Data Corporation (IDC)*, Whitepaper, December 2009.

10 J. Kim, M. Ruggiero, D. Atienza Alonso and M. Ledergerber. "Correlation-Aware Virtual Machine Allocation for Energy-Efficient Datacenters," in *IEEE/ACM Design Automation and Test in Europe Conference (DATE), Grenoble, France, March 18-22, 2013*.

11 M. Kambadur, T. Moseley, R. Hank, M. A. Kim. "Measuring Interference Between Live Datacenter Applications," in *International Conference on Supercomputing (SC)*, 2012.

12 Christina Delimitrou and Christos 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)*, 2013.

13 L. A. Barroso and U. Holzle, "The Case for Energy-Proportional Computing," in *IEEE Computer*, pp. 33–37, 2007.

14 "Resource Management with VMware DRS", http://vmware.com/pdf/vmware_drs_wp.pdf

15 "vSphere Resource Management", http://pubs.vmware.com/vsphere-50/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-50-resource-management-guide.pdf

16 "vSphere SDK for Perl Documentation", http://www.vmware.com/support/developer/viperltoolkit/

17 "Using Virtual CPU Performance Monitoring Counters", http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=2030221