Adaptive Energy-Efficient Resource Sharing for Multi-threaded Workloads in Virtualized Systems

Can Hankendi Ayse K. Coskun

Electrical and Computer Engineering Department
Boston University, Boston, MA 02215
{hankendi, acoskun}@bu.edu

***Abstract*— Computational demand on today's data centers is continuously increasing, as computing trends are shifting towards the cloud. The corresponding increases in energy consumption and and management complexity remain as major challenges for data centers. Server virtualization provides opportunities to improve energy efficiency by reducing the number of physical servers through workload consolidation. Efficient consolidation of multi-threaded workloads requires a detailed understanding of various application characteristics such as performance scaling, inter-thread communications and memory access patterns. This paper proposes an efficient consolidation technique for multi-threaded workloads through adaptive resource sharing on virtual environments. We present a virtual machine reconfiguration algorithm that improves the overall throughput-per-watt of a real-life multicore system by up to 25% in comparison to existing consolidation methods.**

I. Introduction

Energy-related costs are among the biggest contributors to the total cost of ownership for today's modern data centers and high performance computing (HPC) clusters. Therefore, energy efficiency is one of the major goals for achieving sustainability in data centers. One of the main reasons for low energy efficiency in current clusters is under-utilized server nodes. Most server nodes are utilized between 10% to 50% on modern data centers [1]. Under-utilized server nodes indicate a large amount of idle power spent without utilizing the available hardware resources efficiently, leading to lower energy efficiency. Another major challenge for data centers is the increasing server management and administration cost. As number of servers increase, data center management becomes increasingly complex. Server management costs are projected to increase by 3 times from 2000 to 2012 [2].

Server virtualization provides opportunities to improve the energy efficiency of server nodes through consolidation. Consolidating multiple workloads on the same physical node reduces the number of active servers as well as increasing the resource utilization of individual nodes. However, energy efficiency of a consolidated system can dramatically vary depending on various types of resource contentions on the physical node. Thus, reducing the resource contention through co-scheduling techniques have been studied in recent years to improve the energy efficiency [3], [4].

Virtualization also provides an effective centralized management for a large number of server nodes. As data center management is becoming increasingly complex and costly, developing adaptive management techniques is essential to achieve efficient operation. Resource sharing across virtual machines (VM) that reside on the same physical node is an important knob to control the energy efficiency and performance of the server nodes. VMs can be resized or migrated to other server nodes to achieve better performance and energy efficiency [5]. While practices for virtualized resource sharing in enterprise computing have considerably advanced in recent years, consolidation of high-performance multi-threaded loads is an open problem. Resource sharing techniques for multi-threaded workloads is essential to be able to continue improving the energy efficiency, as multi-threaded workloads occupy more of the application domain of HPC clusters and data centers. However, with increasing number of VMs per node, energy-efficient VM reconfiguration remains to be a challenge.

In this paper, we first present an experimental infrastructure that enables accurate performance and power evaluation of consolidated multi-threaded workloads. We then discuss the performance impact of co-scheduling on virtualized systems. We show that the performance of individual VMs can be isolated from each other on a virtualized system by controlling resource affinities. We present an application classification technique that is able to categorize benchmarks according to their power efficiency levels with 97% accuracy. We propose an adaptive resource sharing technique that performs VM reconfiguration for multi-threaded workloads based on benchmark classification to improve energy efficiency of individual server nodes in a cluster. We evaluate the *throughput-per-watt* for our adaptive resource sharing technique for randomly generated 50 workload sets based on the PARSEC suite [6]. We consider *throughput-per-watt* for our evaluations, as it considers both the useful work done (i.e., throughput) and the power consumption. We show that the proposed technique provides 12% higher throughput-per-watt on average and improves the efficiency of the server by up to 25% in comparison to state-of-the-art co-scheduling policies.

The rest of the paper is organized as follows. Section II gives an overview of the prior work. In Section III, we present our methodology to evaluate the impact of co-scheduling. In Section IV, we explore various co-scheduling and application selection strategies on the virtualized system. Section V presents our adaptive resource sharing technique based on benchmark classification. In Section VI, we present the benefits of our resource sharing technique on a real-life system and Section VII concludes the paper.

II. Related Work

Energy and resource management techniques for both virtualized and non-virtualized systems have been widely studied in the literature. One line of work focuses on cluster-level power and resource management techniques that target improving the overall efficiency of the data center. Fan *et al.* study power provisioning strategies for large scale data centers [7]. Wang *et al.* propose a power management technique through a feedback controller to optimize the cluster-level performance while meeting the power constraints [8]. Nathuji *et al.* propose a VM-aware power allocation technique to improve performance under power constraints [9]. Proposed technique allocates power budgets proportionally across virtual machines by favoring applications that are SLA (service level agreement) critical.

Cluster-level resource management techniques are mainly divided into two groups: VM migration and consolidation techniques. Common goal for cluster-level resource management techniques is to provide service availability guarantees. VM migration techniques mostly focus on resource utilization levels of the cluster to provide resource availability [5], [10]. Kusic *et al.* propose a dynamic resource provisioning framework based on lookahead control for virtualized server environments [11]. In order to improve the clusters that heavily utilizes the disk, Romosan *et al.* propose co-scheduling algorithms based on load balancing frequently used files [12]. Zheng *et al.* present an experiment-based management infrastructure for data center management [13]. Their proposed infrastructure allocates a server node (i.e., sandbox) to experimentally derive the energy/performance tradeoffs. Bonvin *et al.* propose a dynamic resource allocation algorithm to meet SLA performance and availability guarantees by adding or removing new resources (i.e., allocation of cores or new server nodes) [14]. However, their proposed work does not consider power and energy aspects.

Although cluster-level techniques are valuable, node-level analyses provide more insights about the underlying reasons for lower energy efficiency of data centers. Node-level techniques focus on workload analysis and the impact of consolidation on individual server nodes. For MPI (message passing interface) based parallel applications, Frachtenberg *et al.* propose a co-scheduling technique based on monitoring MPI calls to identify frequently communicating processes [15]. McGregor *et al.* present scheduling algorithms that determine best thread mixes to improve the performance of multi-threaded applications [16]. Meng *et al.* propose a joint-VM provisioning technique based on workload pattern analysis [3]. Proposed technique selects VM combinations with complementary workload patterns to improve the energy efficiency.

Bhadauria *et al.* propose co-scheduling algorithms based on application characteristics such as cache misses and bus contentions [17]. Proposed algorithms determine the time and space (e.g., number of cores) share of the co-scheduled applications. Authors consider applications that do not scale linearly and propose algorithms to find the best time and space sharing by comparing energy-delay measurements. Dhiman *et al.* propose VM scheduling technique based on application characteristics to improve the energy efficiency [4]. Authors propose a method to estimate VM-level CPU and memory usage based on system-level metrics to make scheduling and migration decisions. Success of the proposed technique depends on identifying the workloads that have complementary characteristics. However, the proposed technique does not consider adjusting resource allocations for co-scheduled applications.

Our proposed resource sharing technique differentiates from the previous work in the following aspects. First, we present an experimental framework to accurately evaluate energy/performance tradeoffs of co-scheduling multi-threaded applications on virtualized systems. We then explore the effect of application selection on energy efficiency. We show that performance degradation due to resource contention can be minimized by setting memory and NUMA affinities for consolidated VMs. Based on our analysis, we propose an adaptive VM reconfiguration algorithm based on power efficiency characteristics of multi-threaded workloads. We demonstrate that the proposed resource sharing technique outperforms the state-of-the-art co-scheduling techniques on a real-life multicore system.

III. Methodology

In this section, we present the details of our experimental setup and explain our methodology. All experiments are performed on an AMD 12-core Magny Cours (6172) server, virtualized by VMware vSphere 5.0 ESXi hypervisor. Magny Cours is a single-chip processor that comprises two 6-core dies (similar to AMD Istanbul architecture) attached side by side. Each core has a 1 MB private L2-cache and each 6-core die has a 6 MB shared L3-cache. All cores share a 16 GB off-chip memory.

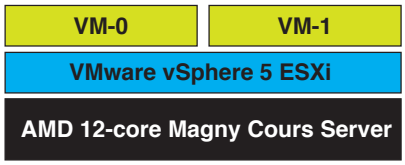

Fig. 1. Virtual server setup.

In Figure 1, we show the overall virtualization setup. We create two virtual machines (VM), *VM-0* and *VM-1*, on top of the hypervisor. Each VM runs Ubuntu Server 11.04 as its operating system. We allocate 8 GB of memory for each VM and allow VMs to add/remove virtual CPUs during runtime (i.e., hot-plugging).

In order to analyze the workload characteristics, we collect per-core performance counter data. We utilize `vmkperf` utility to poll the following performance counter data from the physical CPUs at every 1 second: CPU cycles, retired instructions, and L3-cache misses. We use `esxtop` utility to collect VM-level resource utilization data such as CPU, memory and disk utilization at every 2 seconds, which is the minimum sampling rate provided. In addition to performance related data, we measure system power by using *Wattsup PRO* power meter with a 1 second sampling rate, which is the minimum sampling rate provided.

Parallel applications typically consist of serial I/O stages and a parallel phase, i.e., region-of-interest (ROI). As ROI is the power and performance hungry portion of parallel applications, it is important to consider only the ROI phase for evaluating the energy and performance tradeoffs during consolidation. We run PARSEC [6] multi-threaded benchmarks in our experiments. As the start and end points of ROI phase vary across different applications, we implement a consolidation management interface, `consolmgmt`, that synchronizes the ROIs of co-scheduled applications on top of the default benchmark management interface in PARSEC, `parsecmgmt`. In order to communicate across VMs, we configure a shared network file system (NFS) across VMs. VM that first reaches the ROI phase waits for the other VM to reach its own ROI. As soon as both of the VMs enter the ROI, VMs send appropriate triggering interrupts 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 ROI phase.

PARSEC benchmarks include a wide-range of HPC type applications. There are 13 parallel workloads in the PARSEC benchmark suite. We do not evaluate `fluidanimate`, as it is not possible to run `fluidanimate` with 6 threads. We also do not run `ferret` and `raytrace`, as their internal interrupts disrupt the `consolmgmt` synchronization flow. We use the default native input set that is provided with the PARSEC suite.

IV. Performance Impact of Co-scheduling

In this section, we explore the impact of co-scheduling on application performance and evaluate various application selection strategies.

Depending on the resource requirements of the applications, performance impact of co-scheduling varies dramatically. In this work, we focus on multi-threaded HPC applications most of which require significant CPU resources. Thus, we configure the total number of vCPUs equal to the total number of physical CPUs (pCPU) (i.e., total 12 vCPUs for 2 VMs).

In this setup, main sources of contention are *memory, bus bandwidth*, and *shared caches*. Performance of CPU-bounded applications are expected to be affected less from co-scheduling, as the contention on CPU is eliminated by assigning a different pCPU to each vCPU. On the other hand, applications that have higher memory accesses generate higher bus traffic and increase cache and memory contentions.

In Figure 2, we compare the impact of resource contention on non-virtualized (native) and virtualized system. Non-virtualized system runs CentOS Linux distribution with 2.6.38 kernel. We evaluate the throughput of `streamcluster` benchmark, which is a memory-bounded application from PARSEC benchmark, when co-scheduled with other PARSEC benchmarks. In each experiment, only two PARSEC benchmarks are co-scheduled on the system. For both of the virtualized and native systems, we execute each benchmark with 6 threads running on 6 physical cores and measure the average throughput, which is the number of retired instructions per second. Throughput of `streamcluster` varies significantly on the non-virtualized system, depending on the co-runner application. For instance, running two instances of `streamcluster` decreases the throughput by almost 50% due to increased resource contention. However, on the virtualized system throughput varies marginally regardless of the application pair that is co-scheduled. Virtualized system provides more stable co-scheduling performance due to optimizing memory and NUMA (non-uniform memory access) node affinities according to CPU affinities. On the other hand, on a non-virtualized system, application pairs compete for shared resources and OS does not always optimize memory or NUMA node affinities as hypervisor does. Performance variations due to co-runner application also appear on the virtualized system when there are no memory and NUMA affinities. VMware ESXi hypervisor allows user to assign specific NUMA nodes to each VM. After each VM is assigned to the specific NUMA node and memory spaces, the main sources of contention are eliminated with a minimal performance degradation in comparison to the non-virtualized system.

In Figure 3, we show the average performance deviation of 10 PARSEC benchmarks when each benchmark is separately co-scheduled with the other benchmarks. Again, only two benchmarks are co-scheduled on the system at a time, and each benchmark runs on 6 vCPUs in each experiment. Lower performance deviation implies that the performance of the application is not affected by the co-runner application. On the virtualized system, where memory and NUMA affinities are optimized, performance deviation is consistently lower in comparison to non-virtualized system. `vips` is the only application that has significant deviation on the virtualized system. `vips` is an image processing application with 18 pipeline stages, which have varying performance characteristics. In the `consolmgmt` setup, applications are terminated when one of the co-runner applications complete its ROI. Thus, `vips` is executed for a different number of phases depending on the ROI completion time of the co-runner application, which causes the higher performance deviation across co-scheduling pairs.

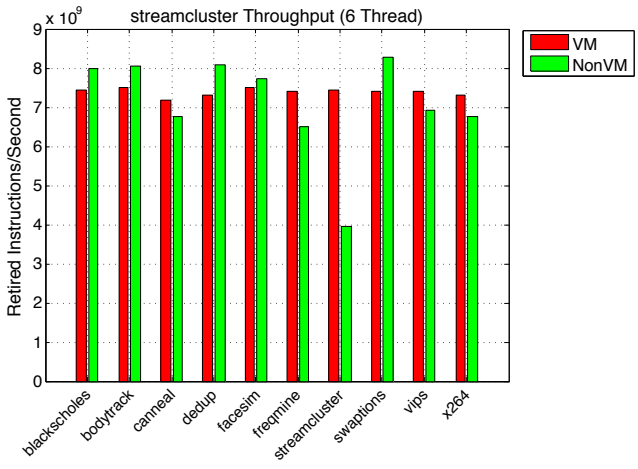

Fig. 2. Performance impact of co-scheduling `streamcluster` with the other PARSEC benchmarks on virtualized and non-virtualized systems. Virtualized system provides stable performance regardless of the co-runner application.

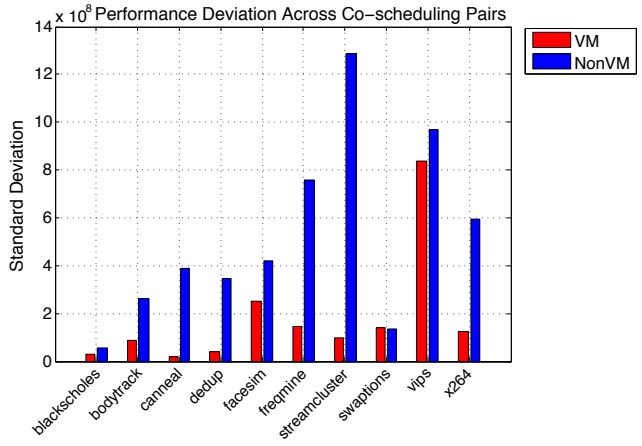

Fig. 3. Performance deviation for each PARSEC benchmark due to co-scheduling.

Previous studies show that co-scheduling the application pairs that have contrasting performance characteristics (i.e., high/low IPC) improves the energy efficiency significantly, as it leads to more balanced resource usage [4]. However, we observe that it is possible to eliminate the performance variation due to co-scheduling by optimizing memory and NUMA node affinities. Thus, co-scheduling policies that are based on application selection have minimal improvement on the energy efficiency of the systems that can provide performance isolation. In order to quantify our observation, we generate 50 workload sets, each consisting of randomly selected 10 applications from the PARSEC suite. We evaluate the throughput-per-watt of the overall workload sets. We evaluate co-scheduling policies that are similar to previously proposed techniques [4], [17]. These techniques first rank the applications according to the selected metric and then co-schedule the highest ranked benchmark with the lowest one, and proceed through the ranked list in a similar fashion. Through application selection, these policies try to balance the resource usage by co-scheduling applications that have contrasting characteristics. We evaluate MPC (memory accesses per cycle), IPC (instructions per cycle), MPC*CPU Utilization, and IPC*CPU Utilization as the metrics used in co-scheduling policies. We also evaluate the throughput-per-watt of the workload sets when applications are co-scheduled randomly. In Figure 4, we show minimum, maximum and

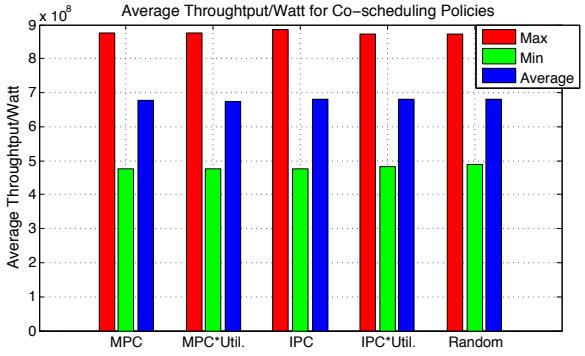

Fig. 4. Maximum, minimum and average throughput-per-watt for 50 randomly generated workload sets when co-scheduling policies are applied.

average throughput-per-watt across the 50 workload sets for various policies. As Figure 4 shows, random application selection does not hurt the throughput-per-watt more than 1% in comparison to previously proposed policies. This observation motivates that rather than choosing the application pairs, it is more important to consider how much physical resources should be allocated for each co-runner application to improve the energy efficiency of a virtualized system.

V. Adaptive Resource Sharing for Multi-threaded Workloads

In this section, we present an adaptive resource sharing technique for parallel workloads through workload classification. In order to maximize the energy efficiency of a server node, it is important to allocate CPU resources depending on the performance and power characteristics of the applications. Increasing the number of cores allocated to a multi-threaded workload improves the performance in all cases. However, depending on the performance scaling of the applications, performance improvements do not always justify the increasing power consumption. Thus, the energy efficiency of the system is significantly affected depending on the performance scaling and the energy proportionality of the applications. Therefore, adjusting the resource shares across VMs can improve the overall energy efficiency of the system by favoring energy proportional applications. To achieve efficient resource sharing across VMs, we first present a clustering-based application classification technique to identify the energy proportionality of the applications. We then propose a resource sharing algorithm across two VMs through dynamically adjusting number of vCPUs for each VM.

A. Metric Selection

Application characteristics dramatically affect the efficiency of the system. In order to improve the overall efficiency of a consolidated system, it is important to allocate more resources to applications that are more efficient. Thus, it is important choose a metric that reflects the power efficiency of the applications accurately.

IPC and CPU utilization are commonly used metrics to evaluate the performance and power characteristics of applications. However, none of these two metrics capture the overall characteristics of the application alone. A high-IPC application might utilize the CPU at lower rates, as IPC is measured over unhalted CPU cycles, which hides the effect of cycle stalls. Similarly, an application that highly utilizes the CPU might have lower IPC rates due to CPU resource stalls such as stalls caused by a busy floating point unit. In order to quantify the accuracy of each metric, we perform linear regression analysis for each metric. For each metric, we regress the metric value together with a constant term to predict the throughput-per-watt of the application. In Figure 5, we show the average prediction error of each metric for 10 PARSEC benchmarks. *IPC*CPU Utilization* outperforms both IPC and CPU Utilization metrics with a 6% average error rate for predicting power efficiency of applications.

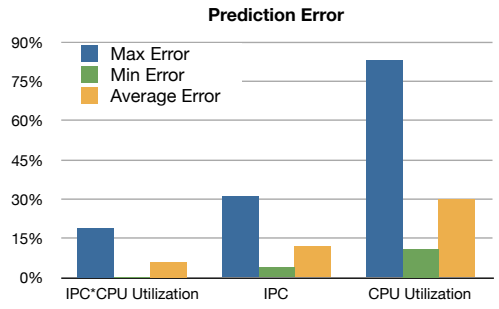

Fig. 5. Maximum, minimum and average prediction errors for candidate metrics.

*IPC*CPU Utilization* metric has been previously proposed to estimate the VM-level IPC [4]. In this work, we utilize *IPC*CPU Utilization* metric to capture the power efficiency characteristics of the applications rather than to derive a VM-level metric. As we assign different set of cores for each VM, we can directly associate hardware events to specific VMs. Thus, our use of *IPC*CPU Utilization* is different than previously proposed approaches. In Figure 6, we show the strong correlation between the power efficiency and the *IPC*CPU Utilization* metric. As *IPC*CPU Utilization* is an accurate measure of application power efficiency, we use *IPC*CPU Utilization* to classify benchmarks.

B. Density Based Clustering Classification

To classify the benchmarks according to their characteristics, we utilize a cluster based classification scheme based on the chosen metric, IPC * CPU Utilization. Commonly used clustering algorithms such as k-mean or fuzzy c-mean requires *a priori* knowledge of number of clusters. Therefore, we utilize DBSCAN (Density-based spatial clustering of applications with noise) clustering algorithm to classify benchmarks, which does not require *a priori* knowledge of number of clusters, as it discovers the clusters on the fly based on a density reachability threshold, ε [18]. Neighbour node, q, is *density reachable* from node p, if the distance between q and p is less than the density reachability threshold, ε. *Density reachability test* essentially determines whether two nodes belong to the same cluster or not, based on their distance.

DBSCAN starts from an arbitrary point, p, and discovers all neighbor nodes that are density-reachable. Distance between

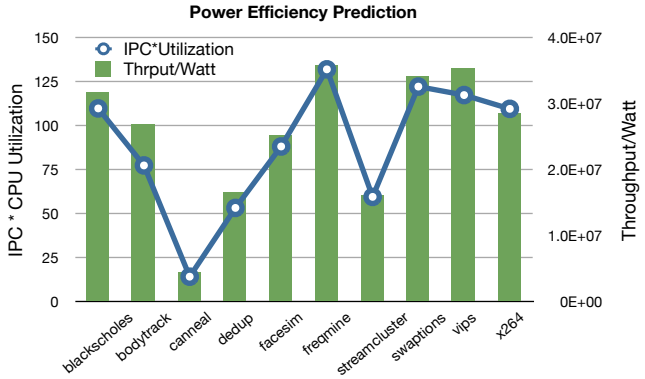

Fig. 6. Correlation between IPC*CPU Utilization (left axis) and application power efficiency (right axis).

clusters, S_1 and S_2, (i.e., set of points) is given as the minimum distance across all member points, p, q, where $\forall p \in S_1$, $\forall q \in S_2$. Clusters are expanded or merged only if:

$$\forall p, q : dist(p, q) < \varepsilon \quad (1)$$

Based on our experimental analysis, we choose ε=20 as the minimum distance between two clusters. In Figure 7, we show the cluster classes (i.e., *high*, *medium*, *low*) and the members (i.e., benchmarks) of each cluster class. Benchmarks that belong to *high* class are the most power efficient benchmarks, where as the *low* class corresponds to the lowest power efficiency class.

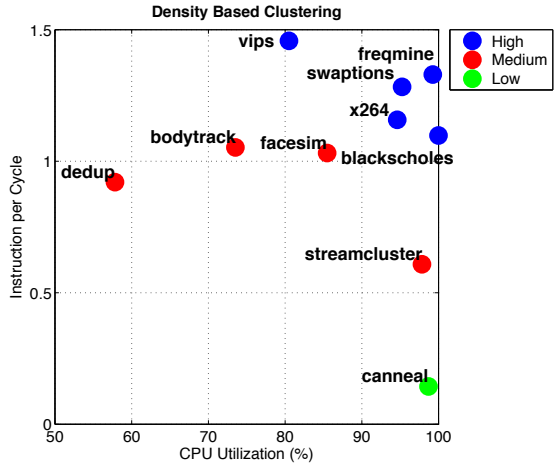

Fig. 7. Benchmark classification through density based clustering.

C. Adaptive Resource Sharing

Based on the benchmark classes that are derived from the DBSCAN algorithm, we allocate CPU resources to each benchmark by favoring the most power efficient ones. Offline benchmark classification is used as a lookup table to adaptively adjust the resource sharing across VMs during runtime. We determine the application classes according to the IPC and CPU utilization metrics. Our runtime technique is also able to adaptively reconfigure the resource sharing of VMs in case of potential phase variations within applications. At runtime, we monitor IPC and CPU utilization of each applications and adjust their resource share accordingly.

As long as the initial training set covers a representative and wide range of applications that have different runtime characteristics, the offline classification scheme will work for *unknown* applications. Moreover, for applications that do not fit within the current classification scheme (i.e., outliers), classification can be re-computed at runtime. As the DBSCAN algorithm has $O(nlogn)$ average complexity, runtime overhead for reclassification is low [18]. Running the reclassification algorithm on the hypervisor level does not affect the performance of VMs, since hypervisor uses its own computational resources that are strictly isolated from computational resources of the VMs.

It is possible reconfigure the resource allocation of VMs by either CPU hot-plugging or adjusting the CPU usage limits. VMware ESXi hypervisor allows CPU hot-plugging, thus

number of vCPUs can be dynamically adjusted without restarting the VMs. Adding and removing vCPUs to VMs impose negligible overhead on the performance of the applications and VM reconfiguration takes effect within 0.5 to 0.8 seconds. Initially, we execute each benchmark with 8 threads. We pack the threads onto a small number of vCPUs, if number of vCPUs is configured to have less than 8 vCPUs. For instance, for a VM with 4 vCPUs, we pack 8 threads onto 4 vCPUs. Thread packing is previously shown to be an effective runtime technique to reduce the number of active cores without causing performance degradation [19]. We assume 3 different VM configurations: 6 vCPUs for each VM; 4 vCPU for VM0, 8 vCPU for VM1; 8 vCPU for VM0, 4 vCPU for VM1. If benchmarks that belong to same power efficiency classes co-scheduled together, we allocate equal number of vCPUs to each VM. Higher power efficiency classes are favored by increasing the number of vCPUs to 8 and the VM running a lower class application is set to have 4 vCPUs. As all of the parallel workloads have fairly good performance scaling up to 4 threads, we do not consider VM configurations less than 4 vCPUs.

We also test adjusting CPU usage limits for VMs as a means of implementing adaptive resource sharing. Adjusting usage limits takes effect within 0.5 second with negligible performance overhead on benchmark execution. Both of the VM reconfiguration techniques can be performed by utilizing vSphere Command Line Interface (vCLI), which allows user to perform administrative actions on VMs. Our adaptive policy runs on the hypervisor level and makes VM reconfiguration decisions by monitoring the IPC and CPU utilization metrics.

VI. Experimental Results

In this section, we evaluate the proposed resource sharing technique on a real-life system. We execute each application with all other possible co-runner applications and utilize our benchmark classification scheme to guide resource allocation decisions.

In our experiments, we perform VM reconfiguration once per co-scheduled benchmark pair. It is also possible to reconfigure the VMs multiple times during the execution. We first evaluate the success of our benchmark classification scheme. For all potential co-scheduling combinations for 10 benchmarks, we first find the optimal resource allocation which maximizes the throughput-per-watt and then compare the optimal solution to our resource allocation technique based on benchmark classification. Within $\pm 3\%$ error range, our resource allocation technique is able to find the optimum solution with 97% accuracy. We define the error range as the percentage of throughput-per-watt difference between the optimum and the proposed resource sharing, where optimum solution is the resource allocation decision that maximizes the throughput-per-watt of the system. As the selected metric, *IPC*CPU Utilization*, is able to predict the power efficiency of benchmarks with high accuracy, our resource allocation technique achieves close-to-optimal accuracy for making resource allocation decisions.

In Figure 8, we show throughput-per-watt improvements for each benchmark in comparison to the baseline case, where all benchmarks share equal resources (i.e., 6 vCPUs each). We report the average, minimum and maximum throughput-per-watt improvements for the cases when benchmarks are co-scheduled separately with other benchmarks. Throughput-per-watt improvements reach up to 25%, with an average of 9% across all benchmarks.

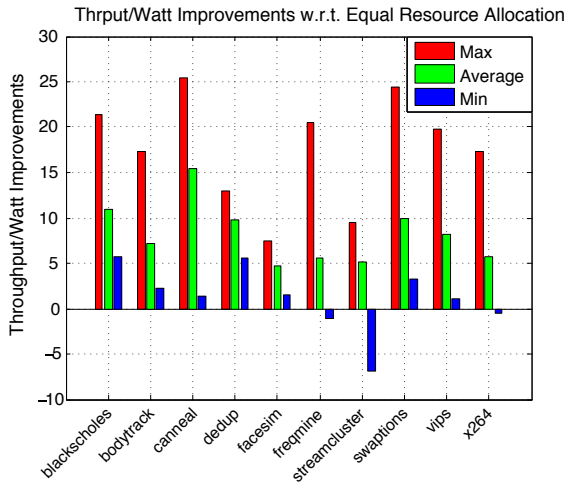

Fig. 8. Throughput-per-watt improvements in comparison to equal resource sharing.

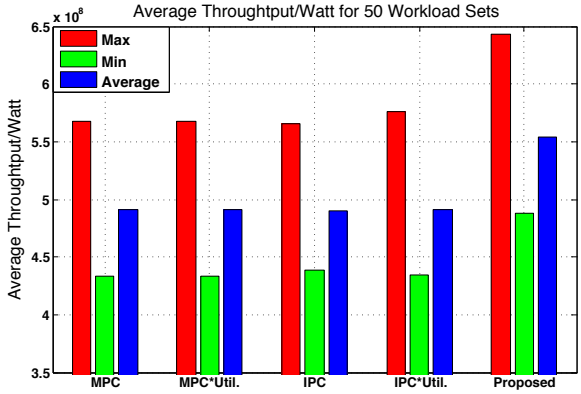

Fig. 9. Throughput-per-watt comparison of previous co-scheduling policies and proposed technique.

We compare our resource sharing technique with co-scheduling policies that are similar to previously proposed approaches. We randomly generate 50 workload sets, each consisting of 10 randomly selected PARSEC benchmarks. For each workload set, we co-schedule the benchmarks according to their rank for various metrics. We balance the selected metric across 10 benchmarks by co-scheduling the higher ranked benchmarks with lower ranked ones. We report throughput-per-watt results for policies based on various metrics and the proposed resource sharing technique in Figure 9. For previously proposed policies, we co-schedule the benchmarks with equal resources. Proposed resource sharing technique outperforms the best previous co-scheduling technique by 12% on average.

VII. Conclusions

Energy efficiency remains to be a major challenge for modern data centers. As multi-threaded workloads dominate the application space of HPC clusters and date centers, proportional resource sharing across multi-threaded workloads provide opportunities to improve the energy efficiency of the system. With increasing number of server nodes, energy-efficient management of data center resources is an important and a challenging problem to be solved.

In this paper, we evaluate the existing co-scheduling techniques that are based on application selection and show that rather than choosing the applications to co-schedule, it is more important to adjust the allocated resources depending on the power efficiency of the applications. As it is possible to improve the degree of performance isolation across VMs, application selection policies have smaller impact on the energy efficiency when compared to proportional resource sharing. We present a benchmark classification technique to classify benchmarks according to their power efficiencies and utilize the classification technique to make design an adaptive resource sharing technique. We show that the proposed resource sharing technique is able to achieve 12% higher throughput-per-watt in comparison to state-of-the-art co-scheduling techniques.

References

[1] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE Computer*, pp. 33–37, 2007.

[2] M. Bailey, "The economics of virtualization: Moving toward an application-based cost model," *International Data Corporation (IDC), Whitepaper*, December 2009.

[3] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *Proceedings of the 7th International Conference on Autonomic Computing*, 2010, pp. 11–20.

[4] G. Dhiman, G. Marchetti, and T. Rosing, "vgreen: A system for energy efficient computing in virtualized environments," in *ISLPED*, 2009, pp. 243–248.

[5] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *10th IFIP/IEEE International Symposium on Integrated Network Management.*, 2007, pp. 119 –128.

[6] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, January 2011.

[7] X. Fan, W. dietrich Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *In Proceedings of International Symposium on Computer Architecture*, 2007, pp. 13–23.

[8] X. Wang and M. Chen, "Cluster-level feedback power control for performance optimization," in *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, feb. 2008, pp. 101 –110.

[9] R. Nathuji, K. Schwan, A. Somani, and Y. Joshi, "Vpm tokens: virtual machine-aware power budgeting in datacenters," *Cluster Computing*, vol. 12, pp. 189–203, June 2009.

[10] A. Beloglazov and R. Buyya, "Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers," in *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*, 2010, pp. 1–6.

[11] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," in *Autonomic Computing, 2008. ICAC '08. International Conference on*, june 2008, pp. 3 –12.

[12] R. Romosan, D. Rotem, A. Shoshani, and D. Wright, "Co-scheduling of computation and data on computer clusters," in *In Proceedings of the 17th International Conference on Scientific and Statistical Database Management (SSDBM)*, 2005, pp. 103–112.

[13] W. Zheng, R. Bianchini, G. J. Janakiraman, J. R. Santos, and Y. Turner, "Justrunit: experiment-based management of virtualized data centers," in *Proceedings of the 2009 conference on USENIX Annual technical conference*, ser. USENIX'09, 2009, pp. 18–18.

[14] N. Bonvin, T. Papaioannou, and K. Aberer, "Autonomic sla-driven provisioning for cloud applications," in *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, 2011, pp. 434 –443.

[15] E. Frachtenberg, D. G. Feitelson, F. Petrini, and J. Fern, "Adaptive parallel job scheduling with flexible coscheduling," *IEEE Trans. Parallel and Distributed Syst*, pp. 1066–1077, 2005.

[16] R. L. Mcgregor and C. D. Antonopoulos, "Scheduling algorithms for effective thread pairing on hybrid multiprocessors," in *In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*. IEEE Computer Society Press, 2005.

[17] M. Bhadauria and S. A. McKee, "An approach to resource-aware co-scheduling for cmps," in *Proceedings of the 24th ACM International Conference on Supercomputing*, ser. ICS '10, 2010, pp. 189–199.

[18] M. Ester, H. peter Kriegel, J. S, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *In Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*. AAAI Press, 1996, pp. 226–231.

[19] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda, "Pack & cap: adaptive dvfs and thread packing under power caps," in *MICRO*, 2011, pp. 175–185.