Towards Better Performance Per Watt in Virtual Environments on Asymmetric Single-ISA Multi-core Systems

Viren Kumar
Simon Fraser University
8888 University Dr
Vancouver, Canada
vka4@sfu.ca

Alexandra Fedorova
Simon Fraser University
8888 University Dr
Vancouver, Canada
fedorova@sfu.ca

ABSTRACT

Single-ISA heterogeneous multicore architectures promise to deliver plenty of cores with varying complexity, speed and performance in the near future. Virtualization enables multiple operating systems to run concurrently as distinct, independent guest domains, thereby reducing core idle time and maximizing throughput. This paper seeks to identify a heuristic that can aid in intelligently scheduling these virtualized workloads to maximize performance while reducing power consumption.

We propose that the controlling domain in a Virtual Machine Monitor or hypervisor is relatively insensitive to changes in core frequency, and thus scheduling it on a slower core saves power while only slightly affecting guest domain performance. We test and validate our hypothesis and further propose a metric, the Combined Usage of a domain, to assist in future energy-efficient scheduling. Our preliminary findings show that the Combined Usage metric can be used as a starting point to gauge the sensitivity of a guest domain to variations in the controlling domain's frequency.

Categories and Subject Descriptors

D.4.1 [**Operating Systems**]: Process Management—*multiprocessing, scheduling*

General Terms

Measurement, Performance, Experimentation

Keywords

virtualization, performance-asymmetric multicore architectures, performance per watt

1. INTRODUCTION

Asymmetric single-ISA (ASISA) multicore processors [21][12] (also known as *heterogeneous*) can potentially deliver a greater performance per watt than homogeneous multicore processors. As power consumption in data centers becomes a growing concern [3], deploying ASISA multicore systems is an increasingly attractive opportunity. These systems perform at their best if application workloads are assigned to heterogeneous cores in consideration of their runtime properties [4][13][12][18][24][21]. Therefore, understanding how to schedule data-center workloads on ASISA systems is an important problem. This paper takes the first step towards understanding the properties of data center workloads that determine how they should be scheduled on ASISA multicore processors. Since virtual machine technology is a *de facto* standard for data centers, we study virtual machine (VM) workloads.

ASISA multicore systems will consist of several cores exposing the same ISA but differing in features, complexity, power consumption, and performance. Most likely, they will feature a handful of complex cores running at high frequency and consuming a relatively high amount of power and a larger number of simple cores running at low frequency and consuming relatively little power [1][13]. The motivation behind ASISA processors is to deliver a higher performance per watt ratio. While some workloads experience an increase in performance proportional to the increase in the power consumption when they run on a high-frequency core as opposed to a low-frequency core, other workloads experience only a marginal gain in performance while consuming a lot more power. To maximize performance per watt, workloads must be assigned to heterogeneous cores in ASISA systems in consideration of their runtime properties. Workloads whose performance is the most sensitive to frequency of the core (i.e., those that experience the largest relative performance gain on a high-frequency core vs. a low-frequency core) should be assigned to run on high-frequency cores, while workloads that are less sensitive should run on low-frequency cores. Several recent studies have demonstrated that this scheduling strategy improves performance per watt [4][13][12][18][24][21].

In this paper, we study properties of virtualized workloads that shed insight into how these workloads should be scheduled on ASISA systems. We hope to use our findings to design new scheduling algorithms in virtual machine hypervisors.

As our experimental virtual platform we use the Xen hyper-

Table 1: Experimental System

Processors	4 quad-core AMD Barcelona
L1 Cache	64 KB per core
L2 Cache	512 KB per core
L3 Cache	2 MB (Shared)
domU clock speeds	2.3 GHz, 2.0 GHz, 1.7 GHz, 1.4 GHz, 1.15 GHz
dom0 clock speed	2.3 GHz
Physical Memory	64 GB
domU Memory	1.0 GB
OS	Gentoo Linux 2.6.21
Xen version	3.2.1

visor [2], a popular system used in data centers. Xen has a single controlling domain, henceforth also referred to as dom0, and many guest domains, also referred to as domUs. The controlling domain in Xen executes code on behalf of the guest domains, usually when the guest domains need access to devices, such as the network interface (NIC) or disk. Frequent device access in dom0 implies several properties about workloads dependent on dom0: (1) in many cases the device, and not the CPU, is the bottleneck, so running this workload on a slower CPU may not hurt overall performance as much as for other workloads; (2) device access involves frequent execution of system code and handling of interrupts – according to previous work this workload is less sensitive to changes in CPU performance than other workloads [18]. To this end, we hypothesize that the workloads executed by dom0 are less sensitive to changes in CPU frequency than domU (CPU-bound) workloads. The main contribution of our work is the test of this hypothesis.

We measured the performance of a large number of virtual workloads running on Xen on cores running at various frequencies and computed sensitivity to frequency changes. To vary the frequency of the cores, we used dynamic voltage and frequency scaling (DVFS) available on our AMD Barcelona system. We found that our hypothesis holds: in general, the performance of workloads that depend on dom0 is less sensitive to variations in CPU performance than workloads that do not depend on dom0. However, the degree of sensitivity depends on the nature of the guest workload that drives dom0. By accurately identifying dom0-dependent workloads that exhibit the lowest sensitivity, we can schedule them on low-frequency cores and maximize the performance per watt ratio. Therefore, another contribution of this work is the discovery of heuristics that help us identify the least sensitive dom0-dependent workloads.

The rest of the paper is structured as follows. Section 2 details our methodology and experimental environment. Section 3 presents the experimental results. Section 4 discusses related work. Section 5 summarizes our contributions and discusses future work.

2. METHODOLOGY AND EXPERIMENTS

To test our hypothesis we selected a range of benchmarks that exercised dom0 and benchmarks that did not depend on dom0 (CPU-bound benchmarks). We measured performance of these benchmarks at various core frequencies and computed sensitivity to changes in frequency. To measure performance of CPU-bound workloads, we measured their runtime in the guest domain. We measured performance of the *guest* domain that exercised dom0: the frequency of the guest domain remained fixed, so any performance variation was due to performance variation in the dom0. Since the dom0-intensive workloads executed almost exclusively in dom0, this measurement technique was a good way to approximate the measurement in dom0 – the reason for not measuring performance in dom0 directly was the lack of access to application-level performance metrics in dom0.

For CPU-bound benchmarks that do not depend on dom0 we chose the industry standard benchmarks: SPEC CPU2000, SPEC CPU2006, and SPEC JBB2005 [22]. The process of selecting dom0-intensive benchmarks was more involved.

While there are a large number of dom0-intensive benchmarks, not all of them provide good potential for power savings on ASISA processors. For example, those dom0-intensive workloads that have low CPU utilization in dom0, due to being highly I/O bound, cannot be used to deliver substantial savings in CPU power consumption simply because they leave the CPU idle most of the time. Therefore, we focused on those workloads that produce high CPU utilization in dom0. This could be accomplished either by choosing benchmarks that naturally produced high CPU utilization in dom0 or by running multiple guest domains that individually produce low CPU utilization but collectively produce moderate to high CPU utilization.

To select the suitable dom0-intensive workloads, we ran a wide range of I/O intensive workloads in the guest domain and measured the resulting CPU utilization in the dom0. We ran the following workloads that utilize disk and network: *dbench* [25], *httperf* [19], *sysbench (file)* [11], *lmbench (pagefault)* [23], *stress* [26], *bonnie* [5], *bonnie++* [6], *iozone* [20], *piozone* [7], *hdparm* [15], *tiobench* [16], *volanomark* [14], *ab (Apache benchmark)* [8], and *netperf* [10]. We ran them using anywhere from one to four guest domains. For many of these benchmarks, CPU utilization in dom0 was low (below 10%) even when multiple guest domains were used. In particular, all disk-bound benchmarks had low dom0 CPU utilization. We suspect that if we had a powerful storage system (as opposed to a single disk), we would see higher CPU utilization for these workloads: in our present testing conditions, where we had only one spindle, the disk was overloaded and so CPU utilization was low. The benchmarks that exhibited moderate to high CPU utilization were the following:

- *ab* with four guest domains (91% CPU utilization in dom0)
- *netperf TCP* with one guest domain (79% CPU utilization in dom0)
- *netperf UDP* with one guest domain (65% CPU utilization in dom0)

In the rest of the paper we focus on these benchmarks. Apache Benchmark (ab) sends a fixed number of requests to a running instance of the Apache web server and then calculates the number of requests per second the web server is ca-

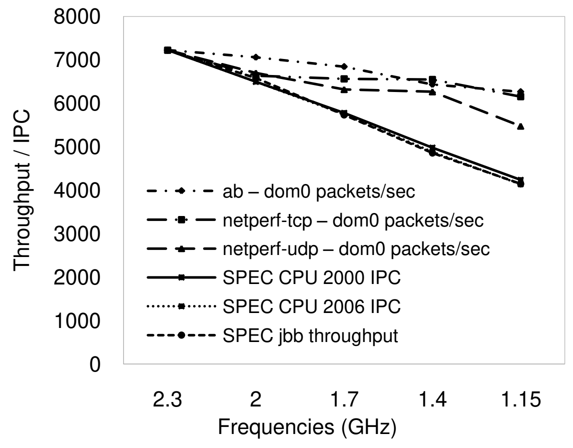


Figure 1: Sensitivity of Workloads

pable of handling. *Netperf* is a networking benchmark that consists of many tests, the most commonly used ones being TCP_STREAM to measure TCP bandwidth and UDP_STREAM to measure UDP bandwidth. The bandwidth is measured in megabits/second.

We ran our experiments on a Dell PowerEdge R905 server powered by four quad-core AMD Barcelona processors capable or running at five different frequencies. We used Xen 3.2.1 running Gentoo Linux 2.6.21 as guest domains and in the dom0. The details of our experimental system are shown in Table 1.

To measure performance in the controlling domain, we used *sysstat* [9], a system-wide performance monitoring tool available for Linux. *Sysstat* gave us the metrics we sought, such as the number of packets processed per second for *ab* and *netperf*.

Xenoprof [17], the *oprofile*-based Xen profiling tool, was used to measure the controlling domainŠs CPU usage during the workload execution phase.

3. EXPERIMENTAL RESULTS

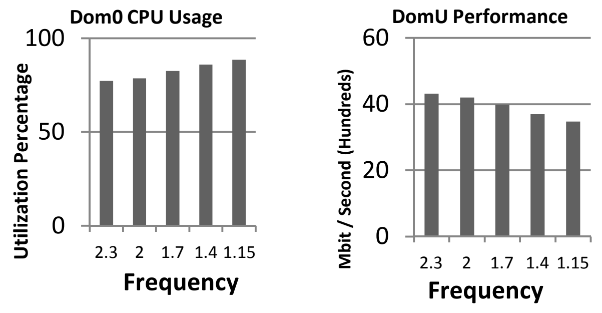
3.1 Sensitivity

Figure 1 shows the relative sensitivity of dom0 performance compared to the CPU-bound benchmarks. Measuring the performance of diverse workloads requires different metrics, and we chose the standard Instructions Per Cycle (IPC) metric for the CPU-bound SPEC workloads. The networking workloads' performance is measured with a throughput metric, i.e. the number of packets that are processed by the dom0 per second. Workloads that are CPU-bound and spend all their time in the guest domain are penalized the most by frequency changes. At the lowest dynamic frequency and voltage setting, SPEC CPU2006's IPC is 58% of the IPC at the highest setting. On the other hand, workloads dependent on dom0 do not suffer as much when the dom0's CPU frequency is adjusted. However, *ab* is not affected as strongly. With *ab*, the rate of processing packets at the lowest frequency is 86% that of the rate at the highest frequency. *Netperf-tcp* and *netperf-udp* also suffer less significantly than CPU-bound workloads: *netperf-TCP*'s performance loss is only 15% and *netperf-UDP*'s loss is 25% compared to the highest-frequency scenario.

3.2 Analysis of Sensitivities

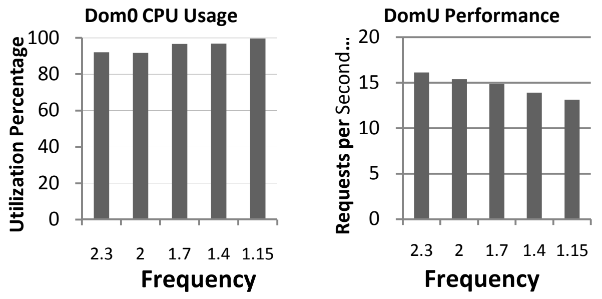
In this section we analyze the sensitivity results. While network saturation in the domUs is not very common in practice, we assume full network utilization in our domUs.

Figures 2, 3, and 4 show the CPU utilization in dom0 and the corresponding performance of the benchmarks at the different frequencies, for *ab*, *netperf-TCP*, and *netperf-UDP* respectively. While *ab* and *netperf-TCP* lose 19% and 20% of performance compared to the highest frequency setting, *netperf-UDP* experiences a slightly more significant loss of



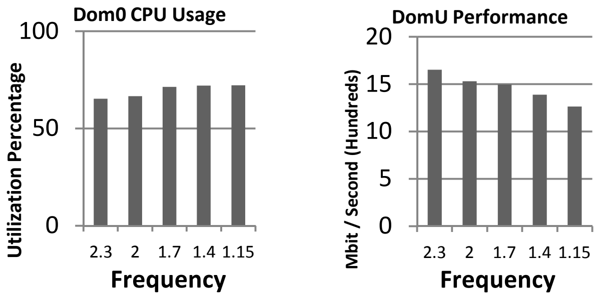
(a) (b)

Figure 3: netperf-TCP Workload



(a) (b)

Figure 2: ab Workload



(a) (b)

Figure 4: netperf-UDP Workload

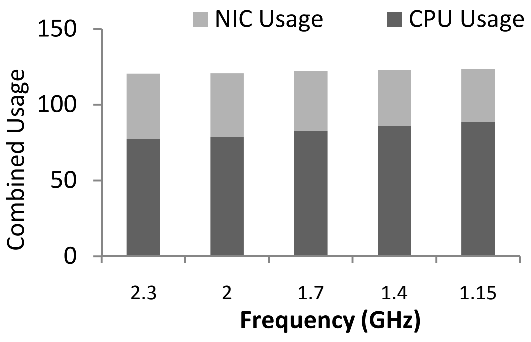


Figure 5: Combined Usage for netperf-TCP

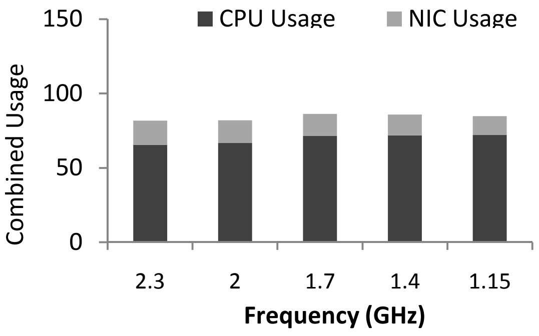


Figure 7: Combined Usage for netperf-UDP

25%. We attempt to understand this difference in sensitivities in order to develop heuristics for scheduling that distinguish more sensitive workloads from less sensitive ones.

Our hypothesis was that *ab* and *netperf-TCP* are less sensitive to variations in frequency than *netperf- UDP*, because they better overlap network I/O with CPU computation, and thus slowing down the CPU has a smaller effect on the overall performance than for the UDP workload. To test whether this explains the difference in sensitivities, we compute the degree of overlap between the I/O and computation in dom0 as follows: we add CPU utilization and network utilization (computed as the ratio of the benchmark's actual bandwidth divided by the maximum theoretical bandwidth of 10 Gbits/second). If there is an overlap between I/O and computation, this overlap, i.e., the sum of utilizations, will be greater than 100% (recall that dom0 runs on a single CPU).

Figures 5, 6, and 7 show the results. Combined usage for *ab* and *netperf-TCP* are above 100% for all frequencies. Combined usage for *netperf-UDP*, on the other hand, is below 100%, showing that I/O and computation do not overlap. (The reason why these numbers do not add up to 100% is because some of the computation is done in the guest domain and in the hypervisor, whose CPU utilization we do not measure.) We suspect that low combined usage correlates with higher sensitivity for the following reason. When the combined usage is low, dom0 is not keeping the device busy. As a result, when CPU frequency is decreased, the rate at which packets are generated decreases even further, slowing down the device even more and decreasing the overall performance. This effect is less noticeable in the workloads where the combined usage and thus the overlap between I/O and computation are high.

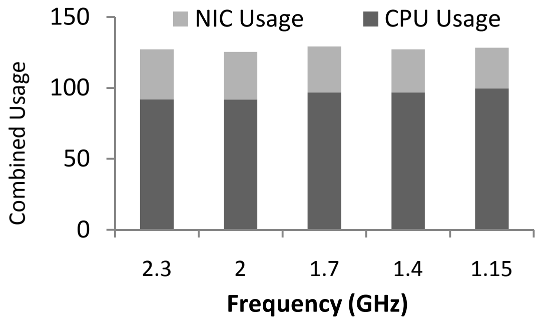


Figure 6: Combined Usage for ab

While combined usage seems like a plausible explanation for the differences in sensitivities, the results that support this theory are still preliminary. In order to thoroughly test the validity of this explanation, we need to run more tests with a wider range of workloads. If this theory is confirmed by additional experiments, combined usage may be used as a heuristic for scheduling of dom0-dependent workloads on ASISA systems.

4. RELATED WORK

Our work is similar to that of Mogul et al., who posit that an ASISA system should have slower cores dedicated to OS tasks [18]. Our work differs from theirs in that we take into account virtualization and run the entire controlling domain, i.e. dom0, on a slower core, as opposed to a few select system calls and interrupts. While the authors did briefly theorize that dom0 could run on a slower, "OS-friendly" core, our work provides the first concrete validation of their hypothesis.

Matching a workload to a certain core, based on the workload's characteristics, builds on the work of Kumar et al [13]. Our work is also similar to other work by Kumar et al. [12], which trades performance for power savings. Unlike them, we do not switch the entire workload to a different frequency but only switch a subset of the workload, the portion that runs in the controlling domain.

5. CONCLUSIONS AND FUTURE WORK

The goal of our work was to analyze performance sensitivity of virtual machine workloads to variations in CPU frequency. We tested the hypothesis that workloads are less sensitive to changes in frequency when they stress dom0. While our hypothesis was validated we found that dom0-dependent workloads vary in the degree of their sensitivity. In an attempt to explain this sensitivity, we proposed combined usage, a combined utilization of the CPU and devices, as the heuristic. While combined usage appears to be a plausible factor explaining the differences in sensitivities, more tests are needed to understand whether it has wide applicability. If wide applicability is confirmed, combined usage can be used as a heuristic for scheduling workloads. The hypervisor scheduler can measure the combined usage of virtual CPUs mapped to dom0, and, in case if it is found to be high, assign those virtual CPUs to low-power physical cores, improving overall performance per watt.

In the future, we plan to evaluate whether combined usage is a good heuristic for scheduling and, if so, implement and evaluate a scheduling algorithm that uses it.

6. REFERENCES

[1] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick. The Landscape of Parallel Computing Research: A View from Berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *SOSP '03: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, pages 164–177, New York, NY, USA, 2003. ACM Press.

[3] L. A. Barroso. The Price of Performance. *Queue*, 3(7):48–53, 2005.

[4] M. Becchi and P. Crowley. Dynamic Thread Assignment on Heterogeneous Multiprocessor Architectures. In *CF '06: Proceedings of the 3rd conference on Computing frontiers*, pages 29–40, New York, NY, USA, 2006. ACM.

[5] T. Bray. Bonnie Disk Benchmark. `http://www.textuality.com/bonnie/`.

[6] R. Coker. Bonnie++ Disk Benchmark. `http://www.coker.com.au/bonnie++/`.

[7] P. Eriksson. A Hard Disk Benchmarking Tool. `http://www2.lysator.liu.se/ pen/piozone/`.

[8] A. S. Foundation. ab - Apache HTTP Server Benchmarking Tool. `http://httpd.apache.org/docs/2.0/programs/ab.html`.

[9] S. Godard. Performance Monitoring Tools for Linux. `http://pagesperso-orange.fr/sebastien.godard/`.

[10] R. Jones. A Network Performance Benchmark. `http://www.netperf.org`.

[11] A. Kopytov. Sysbench Benchmarking Tool. `http://sysbench.sourceforge.net/`.

[12] R. Kumar, K. Farkas, N. Jouppi, P. Ranganathan, and D. Tullsen. Single-ISA Heterogeneous Multi-core Architectures: The Potential for Processor Power Reduction. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, 2003.

[13] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas. Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance. In *ISCA '04: Proceedings of the 31st Annual International Symposium on Computer Architecture*, page 64, Washington, DC, USA, 2004. IEEE Computer Society.

[14] V. LLC. VolanoMark benchmark. `http://www.volano.com/benchmarks.html`.

[15] M. Lord. A Hard Drive Performance and Benchmarking Utility. `http://sourceforge.net/projects/hdparm/`.

[16] J. Manning and M. Kuoppala. Threaded I/O Benchmark for Linux. `http://tiobench.sourceforge.net`.

[17] A. Menon, J. R. Santos, Y. Turner, G. J. Janakiraman, and W. Zwaenepoel. Diagnosing Performance Overheads in the Xen Virtual Machine Environment. In *VEE '05: Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, pages 13–23, New York, NY, USA, 2005. ACM.

[18] J. C. Mogul, J. Mudigonda, N. Binkert, P. Ranganathan, and V. Talwar. Using Asymmetric Single-ISA CMPs to Save Energy on Operating Systems. *IEEE Micro*, 28(3):26–41, 2008.

[19] D. Mosberger and T. Jin. httperf — A Tool for Measuring Web Server Performance. *SIGMETRICS Perform. Eval. Rev.*, 26(3):31–37, 1998.

[20] W. Norcutt. The Iozone Filesystem Benchmark. `http://www.iozone.org/`.

[21] D. Shelepov, J. C. Saez, S. Jeffery, A. Fedorova, N. Perez, Z. F. Huang, S. Blagodurov, and V. Kumar. HASS: A Scheduler for Heterogeneous Multicore Systems. *SIGOPS Operating Systems Review*, 43(2):55–75, April 2009.

[22] SPEC. SPEC CPU 2000. `http://www.spec.org`.

[23] C. Staelin and H.-P. Laboratories. lmbench: Portable Tools for Performance Analysis. In *In USENIX Annual Technical Conference*, pages 279–294, 1996.

[24] R. Strong, J. Mudigonda, J. C. Mogul, N. Binkert, and D. Tullsen. Fast switching of threads between cores. *SIGOPS Oper. Syst. Rev.*, 43(2):35–45, 2009.

[25] A. Tridgell. Dbench Benchmarking Tool. `http://freshmeat.net/projects/dbench/`.

[26] A. Waterland. Stress Workload Generator. `http://weather.ou.edu/~apw/projects/stress/`.