Implications of Cache Asymmetry on Server Consolidation Performance

Padma Apparao, Ravi Iyer and Don Newell
Hardware Architecture Laboratory, Intel Corporation
Contact: ravishankar.iyer@intel.com

Abstract

Today's CMP platforms are designed to be symmetric in terms of platform resources such as shared caches. However, it is becoming increasingly important to understand the performance implications of asymmetric caches for two key reasons: (a) multi-workload scenarios such as server consolidation are a growing trend and contention for shared cache resources between workloads causes logical cache asymmetry, (b) future CMP platforms may be designed to be physically asymmetric in hardware due to die area pressure, process variability or power/performance efficiency. Our focus in this paper is to understand the performance implications of both logical as well as physical asymmetric caches on server consolidation. Based on real measurements of a state-of-the-art CMP processor running a server consolidation benchmark (vConsolidate) we compare the performance implications as a function of (a) symmetric caches, (b) virtually asymmetric caches, (c) physically asymmetric caches and (d) a combination of logically and physically asymmetric caches. We analyze the performance behavior in terms of (i) performance of each of the individual workloads being consolidated and (ii) architectural components such as CPI and MPI. We believe that this asymmetric cache study is the first of its kind and provides useful data/insights on cache characteristics of server consolidation. We also present inferences on future optimizations in the VMM scheduler as well as potential hardware techniques for future CMPs with cache asymmetry.

I. Introduction

The emergence of chip multiprocessors (CMP) has enabled the industry with a significant increase in hardware parallelism on a server platform. Such CMP platforms [1, 12, 13, 10, 19] can be taken advantage of in two forms: (a) improving the performance of a parallel application running on the platform and (b) improving the performance of many (potentially parallel) applications running simultaneously on the platform. In the last few decades, several researchers [4, 14, 17] have characterized the performance of individual highly parallel applications (e.g. OLTP, Java, ERP, etc) running on a server platform (single-core as well as multi-core). However, the rapid growth of virtualization and server consolidation [11, 25, 26, 29] emphasizes the need to study the performance behavior of server consolidation workloads on current and future CMP architectures. In this paper, we will focus our attention on characterizing the performance of consolidation workloads. In particular, our goal in this paper is to provide a detailed understanding of the cache performance behavior of server consolidation workloads running on CMP platforms.

Since most server workloads are memory-intensive, understanding and optimizing cache performance is critical in server platforms. Most CMP platforms today employ shared last-level caches between one or more cores within the processor socket. For example the latest quad-core Intel Xeon Processor 5400 series [10] consist of four cores, with each pair of cores sharing a 6MB last-level cache. Although the cache size is symmetric in the hardware within each processor and across processors in the platform, any given virtual machine running within a consolidated workload on a server will contend for this shared cache resource with other virtual machines running simultaneously. This results in logical cache asymmetry and ends up affecting the performance of the virtual machine. So far, there have been very limited studies [2, 8, 21] characterizing the implications of shared cache contention on server consolidation performance. In this paper, we present detailed measurement-based analysis of the implications of virtual cache asymmetry on the performance of virtual machines that are part of vConsolidate (a consolidation benchmark).

Researchers in the past have also considered asymmetric architectures to improve the power/performance efficiency of future CMP platforms. In addition to core asymmetry [20], it is also important to consider hardware cache asymmetry since cache occupies a significant fraction of the processor die (40% or more) and it may be worthwhile to support physical cache asymmetry in future CMP platforms (for process variability or better power/performance efficiency). In this paper, we also study the implications of physical cache asymmetry in the context of server consolidation. Using the vConsolidate server benchmark, we show how each of the individual virtual machines change in overall performance as well as cache performance as they run on a platform with symmetric cache sizes versus a platform with asymmetric cache sizes. To our knowledge, this study is the first measurement-based evaluation of physical cache asymmetry on a server consolidation workload.

Overall, this paper makes the following major contributions:
(a) We present a detailed measurement characterization of a server consolidation workload using a recent benchmark (vConsolidate) on the latest Intel quad-core Xeon platform.
(b) We show the performance implications of virtual cache asymmetry on virtual machine performance within a consolidation environment.
(c) We show the performance implications of physical cache asymmetry on virtual machine performance within a consolidation environment.

978-1-4244-2778-9/08/$25.00 ©2008 IEEE

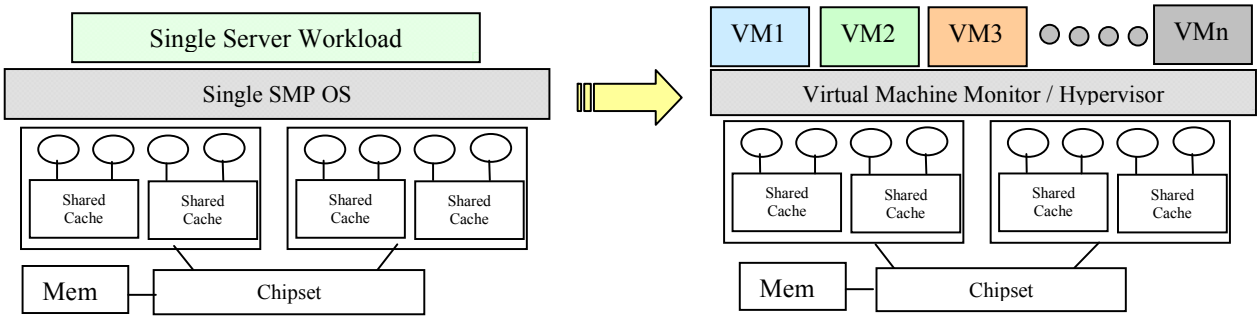


Figure 1: Towards Virtualization and Consolidation on CMP Servers

(d) We discuss the potential optimizations needed in systems software (VMM scheduler) and or hardware architecture when running consolidation workloads on asymmetric cache architectures.

The rest of this paper is organized as follows. Section II presents an overview of server consolidation along with related work in this area. Section III introduces the different forms of asymmetric caches and discusses related work in this area. Section IV presents the measurement-based methodology and tools required to perform the detailed characterization. Section V presents the results and analysis. Section VI presents inferences and case studies highlighting some future optimizations in the context of asymmetric caches. Section VII concludes the paper by summarizing the contributions and presenting a direction for future research.

II. Server Consolidation on CMP Platforms

In this section, we will present an overview of the different types of cache asymmetry and discuss it in the context of individual workloads and server consolidation. We start by presenting an overview of server consolidation and some background on related work.

A. Server Consolidation Overview

Within the last decade, data centers have started employing virtualization solutions [30, 32] to consolidate multiple server applications on the same platform. This was motivated by the need to (a) reduce complexity of managing and interconnecting multiple platforms in the datacenter, (b) improving the overall resource utilization and total cost of ownership by sharing resources and (c) allowing more flexibility to migrate applications and deploy newer ones. Within the next decade, it is expected that more than 50% of the servers in the marketplace will be running consolidated workloads as opposed to individual applications.

Figure 1 illustrates the transition from an individual image (single OS) and single server application running on a platform to a virtual machine monitor (VMM) or hypervisor running multiple images (i.e. virtual machines -- VMs) on a server platform. This transition becomes especially important as the number of cores integrated into CMP processors continues to increase and datacenter-on-chip usage models start to become real. As this transition gains momentum, architects need to start considering the performance implications of server consolidation when making decisions on future platforms. While CMP platforms provide abundant hardware parallelism to achieve server consolidation, running multiple workloads simultaneously introduces contention for shared resources in the platform. A typical CMP processor today consists of multiple cores with one or more shared last-level caches within the processor. Contention for shared cache resources and shared memory bandwidth provides better resource usage efficiency, but also introduces performance isolation concerns for the system administrator and data center manager. In this paper, we focus our attention on one of the critical shared resources in the platform – namely the last-level cache.

B. Related Work on Server Consolidation

There have been only a few recent studies [2, 8, 21] that explore shared cache characteristics of consolidation environments. Jerger et al [8] presented a simulation-based study that showed the performance implications of private and shared caches on a mix of server workloads to mimic server consolidation. While this study is a unique first step towards evaluating server workloads, it does not employ a virtualization benchmark and the use of simulation-based evaluation introduces some uncertainty in the data. Apparao [2] et al present a modeling and analysis approach for server consolidation and highlight that understanding shared cache interference is a cruicial component of performance modeling. Marty [21] study the benefits of a virtual coherence hierarchy in the context of server consolidation. They employ a simulation-based approach with a mix of server workloads to mimic server consolidation.

In this paper, we present a detailed measurement characterization of cache behavior of a recently defined server consolidation benchmark (vConsolidate). The focus of our study is to understand the implications of both virtual asymmetry (caused by interference) as well as physical asymmetry in the hardware. In the next section, we describe these degrees of cache asymmetry in more detail and elaborate on its potential implications on server consolidation performance.

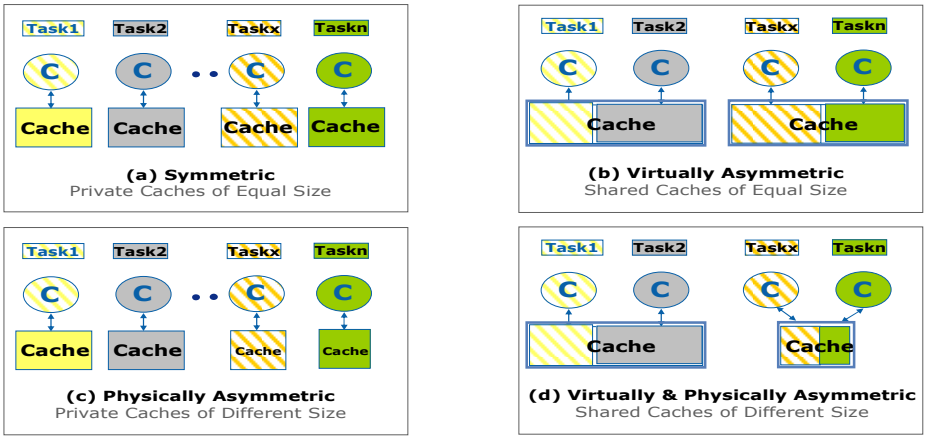


Figure 2: Illustrating Virtual and Physical Asymmetry in Cache Architectures

III. CACHE ASYMMETRY IN CMP PLATFORMS

In this section, we describe the degrees of asymmetry in cache architectures as a function of platform design as well as the consolidation of workloads running on the platform.

A. Asymmetry in Cache Architectures

We classify symmetric/asymmetric cache architectures into the following four major categories:

(a) Symmetric Caches: Figure 2(a) illustrates a symmetric cache architecture where every core has a private last-level cache of equal size. In such a symmetric cache architecture, a task can run on any core and have equal amount of cache space at its disposal.

(b) Virtually Asymmetric Cache: Figure 2(b) illustrates a virtually asymmetric cache configuration where a core shared the last-level cache with one or more other cores and all the shared last-level caches are of equal size. Such a cache configuration is classified as virtually asymmetric because a task running on any core cannot deterministically have a pre-defined amount of cache at its disposal. In other words, the amount of cache space that this task ends up with depends heavily on the other tasks running on (sibling) cores that share the same last-level cache. At one end, a task can have the entire shared last-level cache at its disposal if no other task runs on the sibling cores. On the other end, the task could get very minimal share of last-level cache if it runs with cache-intensive tasks on the other sibling cores.

(c) **Physically Asymmetric Cache:** Figure 2(c) illustrates a physically asymmetric cache organization where there is a private last-level cache dedicated to each core, but each of private last-level cache may be different in size. In such a physically asymmetric cache configuration, a task can get more or less cache to its disposal depending on which core it runs on. As architects explore future asymmetric platforms, asymmetry in cache could be one of the attributes in the platform due to die area pressures and the need for better power/performance efficiency. In such configurations, it is important that sufficient care is taken in determining how task scheduling is accomplished.

(d) **Virtual+Physical Cache Asymmetry:** Figure 2(d) illustrates a cache configuration which is virtually as well as physically asymmetric. This is essentially a combination of the previous two configurations. The physical asymmetry occurs because the shared last-level caches are of different size. The virtual asymmetry occurs because we employ shared caches between cores and the task running on these cores can interfere with each other and end up with very different amounts of cache space. In such configurations, it is important to ensure that task scheduling takes into account both forms of asymmetry by scheduling tasks that are cache-friendly (or more important) on cores attached to larger shared caches and tasks that are less cache-friendly (or less important) on cores attached to smaller shared caches. In addition, it is important to ensure that tasks co-scheduled on the sibling cores behave well with each other in terms of shared cache usage.

B. Related Work on Cache Asymmetry

In recent years, there have been several studies [6, 18, 15, 16, 23, 24] on understanding shared cache interference and identifying appropriate techniques to improve QoS, fairness and efficiency of shared caches. For example, Chandra et al [6] described mechanisms to predict inter-thread cache contention in CMP architectures. Iyer et al [15, 16] described QoS mechanisms to provide performance differentiation in shared caches due to the inter-thread interference. Qureshi et al [23] defined utility-aware cache partitioning techniques to improve cache efficiency and improve overall performance. Most of these studies were done in the context of simple applications (usually SPEC) running simultaneously and the evaluation was all simulation-based. In addition, they did not consider physically asymmetric caches in the study. We believe our study is the first to discuss the entire spectrum of cache asymmetry and especially evaluate its implications in the context of a server consolidation benchmark (vConsolidate).

IV. MEASUREMENT-BASED METHODOLOGY & TOOLS

In this section, we present our evaluation methodology, the consolidation workload and tools we used to gather data.

A. Measurement Platform

Our measurement-based evaluation was conducted on a dual-socket Intel server platform that has two processor sockets (the latest Intel Xeon 5400 series) and is populated with 16GB of memory. Within each processor socket, there are 4 cores running at 2.8 GHz. Each pair of cores share a 6MB cache, adding up to a total of 12MB on each socket. To mimic asymmetric platforms, we employed processors of differing cache sizes (12MB, 8M, 6M and 4MB) in the platform. Figure 3 illustrates the symmetric platform and the asymmetric cache platform.

B. vConsolidate Benchmark

The vConsolidate (vCon) benchmark [5] consists of four key virtual machines (VMs): (a) a compute intensive workload/application, (b) a database workload, (c) a web server workload, and (d) a mail server workload. To emulate a real world environment, an idle virtual machine is added to the mix since datacenters are not fully utilized all the time. The compute intensive VM runs a modified version of SPECjbb2005 [27]. Typically SPECjbb2005 is a cpu intensive workload that consumes as much cpu as it possibly can. However, in this environment, namely vCon, SPECjbb has been modified to consume roughly 75% of the cpu or so, by inserting in random sleeps every few millisecs. The database virtual machine runs Sysbench [28], which is a OLTP workload executing transactions against a MySQL database. The web server VM runs Webbench [31] which uses Apache webserver. The mail server virtual machine runs Microsoft Exchange workload that executes transactions on Outlook with 500 users logged in simultaneously.

A vConsolidate configuration as described above with four active VMs and an idle VM comprises a Consolidated Stack Unit (CSU). Figure 4 illustrates a single CSU configuration. The vCon benchmark also defines various profiles – we have chosen a profile that is a mix of 32-bit and 64-bit VMs including 3 Linux VMs and 2 Windows VMs. The SPECjbb VM (linux 64 bit) has 2 vcpus and 2GB memory, Sysbench VM (linux 64 bit) has 2 vcpus and 1.5GB memory, Webbench (32 bit linux) is assigned 2 vcpus and 1.5GB memory and Mail (32 bit Windows) VM has 1vcpu and 1GB memory. The idle VM is given 1 vcpu and 0.4GB memory. The entire configuration is on a private switched subnet with two clients generating traffic for the Webserver workload and one client generating traffic for the Exchange/Mail workload. All VMs run on platforms with Intel Virtualization technology [11, 29] enabled.

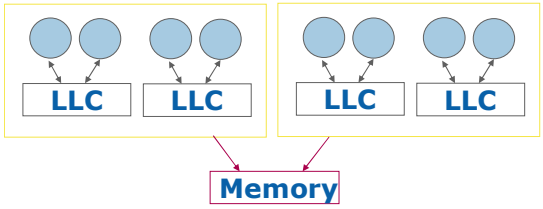


Figure 3: Measurement Platform Architecture

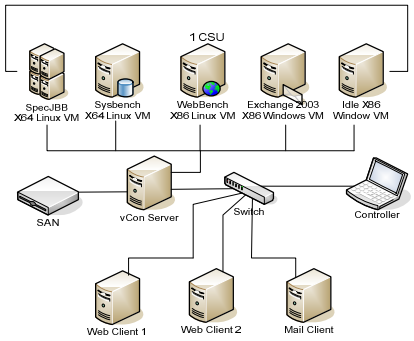


Figure 4: vConsolidate Benchmark Components

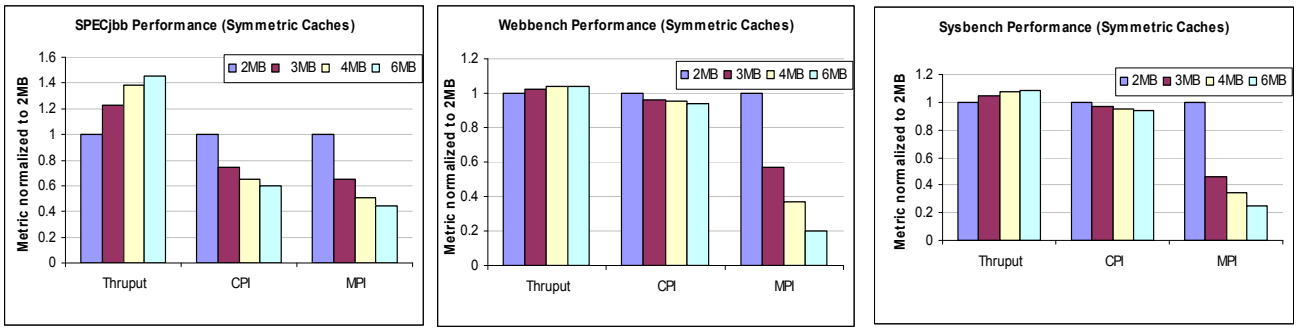
C. Evaluation Tools and Configurations

We run vConsolidate on top of Xen 3.1 [3, 32] which has support for HVM guests and also has tools that support profiling. Commonly available tools [7, 9] with Xen such as sar, xentop, xm info, xentrace have been used to get information such as cpu usage and network/IO traffic from each VM. We used xm vcpu-pin and vcpu-set to affinitize the VMs to different cores in order to evaluate asymmetric caches. To get architectural metrics such as CPI (cycles per instruction), L2 MPI (misses per instruction) we used a tool that can read processors performance counters and associate the cpu cycles, the L2 misses, and the instructions to each of the VMs running during that period of time. Our measurement configurations consist of the following:

(a) running each virtual machine individually by affinitizing the two virtual cpus within the VM to two physical cores that are attached to different caches. To emulate a symmetric cache, we affinitize to cores attached to same size caches. To emulate an asymmetric cache, we affinitize to cores attached to different sized caches.

(b) running the vCon benchmark by affinitizing all of the virtual machines to only four cores. This results in a >2X oversubscription (since there are more than 8 vcpus requested, but only 4 cores available). Again, depending on symmetric cache or asymmetric cache evaluation, we affinitize the virtual machines to cores attached to either same sized caches or differently sized caches.

(c) We also run experiments where we affinitize a virtual cpu to a specific cache size to understand the benefits of scheduler optimizations in the future that take into account the asymmetry in the platform and attempt to optimize to improve overall throughput or quality of service (QoS).



(a) SPECjbb Performance **(b) Webbench Performance** **(c) Sysbench Performance**

Figure 6: Virtual Machine Performance in Symmetric Cache Configurations

V. Results and Analysis

In this section we analyze the measurement data collected for four configurations: (a) symmetric caches running individual virtual machines, (b) virtually asymmetric caches when running consolidation in a shared cache, (c) physically asymmetric caches when running individual virtual machines on asymmetric cache sizes, and (d) virtual+physical asymmetry where consolidation is run on a physically asymmetric cache. We start with our baseline configuration (a), which we will compare against in later subsections on asymmetric caches.

A. Symmetric Caches and Individual Workloads

In this case, we run each virtual machine individually in a symmetric cache configuration. However, we individually choose the performance optimal configuration by observing that some virtual machine workloads prefer to share cache between its virtual CPUs, whereas other workloads do not. For example, SPECjbb performs the best when the two vcpus of the SPECjbb VM are affintized to cores attached to different caches, thereby sharing no cache at all between the vcpus. Thus each vcpu gets the full cache to itself. On the other hand, webbench and sysbench individually perform better when the two vcpus of each workload are on cores connected to the shared cache (i.e. they share the last-level cache). Figure 5 illustrates the two symmetric configurations.

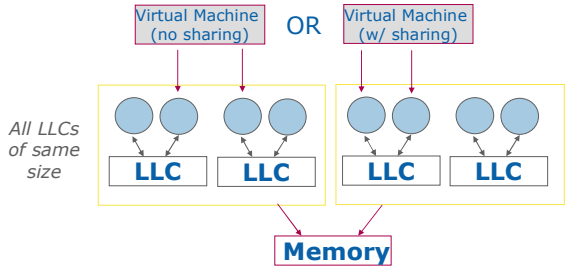


Figure 5: Individual Workloads on Symmetric Caches

Figure 6 shows the performance for the different workloads when run individually in their best symmetric configurations. In each figure, we show a comparison of the overall performance, the cycles per instruction (CPI) and the misses per instruction (MPI) as a function of the last-level cache (LLC) size (varied as 6MB, 4MB, 3MB, 2MB). Of the three workloads, SPECjbb is the most cache sensitive workload with the overall performance increasing by as much as 50% as cache size is increased from 2MB to 6MB. The performance improvement is reflected in CPI reduction (close to 40%) and the MPI reduction (by as much as 55%). The data also shows (as expected) that the benefits of increasing cache size is higher when going from 2MB to 3MB as opposed when increasing from 4MB to 6MB.

The other two workloads show lower sensitivity to cache size increase. The overall throughput and CPI changes by less than 10% when increasing cache size from 2MB to 6MB. It should be noted that the misses per instruction (MPI) however does reduce significantly when increasing cache size (by 80% for Webbench and 75% for Sysbench). However, this does not affect overall performance much because the misses per instruction are smaller (compared to SPECjbb2005) to begin with and the memory stall time component of CPI is also a much smaller fraction. This also occurs because these workloads benefit more from shared cache effects as compared to cache capacity.

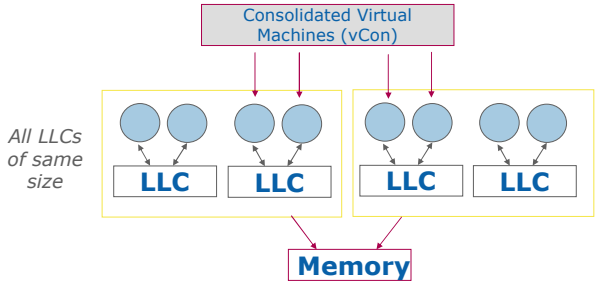
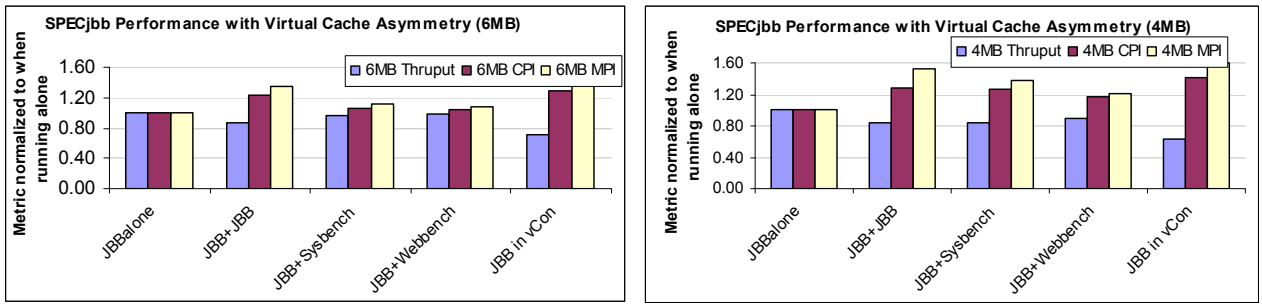


Figure 7: vCon Consolidation & Virtual Asymmetry

B. Virtual Cache Asymmetry with Consolidation

In this configuration, we ran the vConsolidate benchmark with all four active virtual machines (SPECjbb2005, Sysbench, Webbench,with two vcpus each and Mail with 1 vcpu) and the idle virtual machine on the same platform affinitized to four of the eight physical cores. The four cores were picked such that they shared two last-level caches of equal size. This was chosen both to study virtual cache asymmetry (since multiple workloads will be contending for



(a) 6MB Last-Level Caches (b) 4MB Last-level Caches

Figure 8: SPECjbb Performance in Virtual Asymmetric Cache Architectures

the shared cache) as well as core interference (since there are more virtual cpus (>8) as compared to physical cores). Figure 7 illustrates the configuration chosen. It should be noted that two cores from each socket was chosen (although we repeated the study with four cores from the same socket and found the results to be almost identical). In addition, we also ran the pairs of virtual machines to show how virtual asymmetry changes depending on the number of workloads being run simultaneously.

Figure 8 presents the data for SPECjbb2005 and shows how the performance, CPI and MPI change from when it runs alone on a dedicated cache (symmetric cache) to when it runs with one other workload (pairwise) versus when it runs with all other virtual machines in consolidation. In figure 8(a), the shared last level caches are 6MB in size each. In figure 8(b), the shared last level caches are 4MB in size each. Let's start by analyzing the Figure 6(a).

When SPECjbb2005 virtual machine runs with another SPECjbb2005 virtual machine, we find that each virtual machine suffers about a 16% loss in performance because it contends for shared cache (but not for cores). When SPECjbb2005 runs with Sysbench or with Webbench, the effect is lower in significance. However, when it runs all virtual machines within vConsolidate, we find that it loses more than 30% in performance due to both core and cache interference. To isolate the impact of cache interference alone, we can look at the CPI increase, where find that the SPECjbb2005 CPI increases by 25% (implying a performance loss of 20%) due to consolidation. Figure 6(b) shows similar effects on a 4MB cache. Table 1 shows the additional impact of core interference based on the cpu utilization loss during consolidation. The core interference contributes to about 7 to 9% of the performance loss.

cpu%	Alone	in vCon	Delta
JBB (6M)	133	121	9%
JBB (4M)	137	127	7%
JBB (3M)	144	130	9%
JBB (2M)	154	140	9%

Table 1: Core Interference Impact

We have also done similar experiments with Sysbench and Webbench to understand the effects of virtual cache asymmetry (but do not present the data for brevity). Overall, we find that virtual asymmetry due to cache interference has significant impact on each VM's performance.

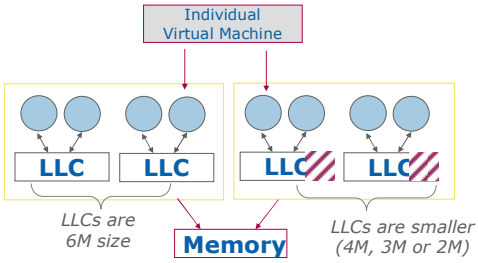
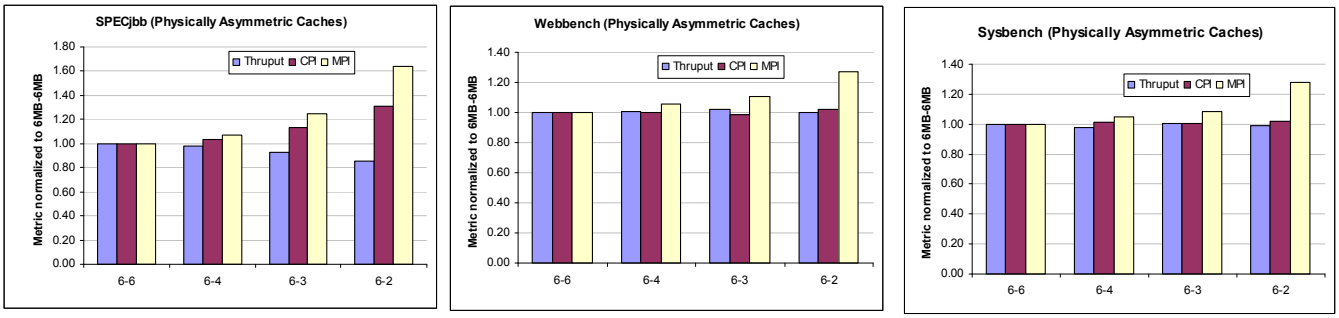


Figure 9: Individual VMs on Physical Cache Asymmetry

C. Physical Cache Asymmetry with Individual Workloads

In both previous sections, we evaluated configurations with shared last-level cache sizes of equal size in the hardware. In this section, we analyze the impact of physical cache asymmetry on a single virtual machine running one workload. To achieve this, we used one of the processor sockets with a different cache size in the platform and then affinitized the vcpus of an individual virtual machine to the cores attached to caches of different sizes. Figure 9 illustrates the physically asymmetric cache configuration and the VM mapping.

Figure 10 shows the impact of physical cache asymmetry on the performance, CPI and MPI of individual virtual machines (SPECjbb2005, Webbench and Sysbench) running alone. The first set of bars in every chart shows the data for the base symmetric cache configuration (with two 6MB caches), whereas the subsequent sets of bars shows the data when the two vcpus of a VM are on cores attached to asymmetric caches differing in size. For example, the SPECjbb VM has two vcpus, one of which is affinitized to a 6MB and the other to either 4MB, 3MB or 2MB cache (depending on the sets). In Figure 10(a) we observe that the performance of SPECjbb reduces by more than 15% going from a 6-6 configuration to a 6-2MB configuration. This is around half of what we lost in the symmetric cache where the 2MB performance loss was 31% from the 6MB. Similarly at 6-4MB we lose 7 % (15%



(a) SPECjbb Performance **(b) Webbench Performance** **(c) Sysbench Performance**

Figure 10: Implications of Physical Cache Asymmetry on Individual Virtual Machines

in the symmetric case) and 2% in the 6-4MB case (5% in the fully 4MB symmetric cache). Webbench and Sysbench are less sensitive to asymmetric caches as compared to SPECjbb2005. To clearly show the performance difference between asymmetric caches, Table 2 shows the difference in the CPI and MPI of different vcpus within the same workload. Since one vcpu (vcpu0) was attached to a large cache (6MB) and the other vcpu (vcpu1) is attached to a smaller last-level cache (varied from 6MB to 2MB). For example, for SPECjbb2005, the CPI difference between vcpu0 (6MB) and vcpu1 (2MB) is as high 62%.

	6MB	4MB	3MB	2MB
JBB	**vcpu0**	**vcpu1**	**vcpu1**	**vcpu1**
CPI	1.00	1.06	1.24	1.62
MPI	1.00	1.14	1.47	2.27
	6MB	**4MB**	**3MB**	**2MB**
Sysbench	**vcpu0**	**vcpu1**	**vcpu1**	**vcpu1**
CPI	1.00	0.99	1.01	1.04
MPI	1.00	1.11	1.21	1.65
	6MB	**4MB**	**3MB**	**2MB**
Webbench	**vcpu0**	**vcpu1**	**vcpu1**	**vcpu1**
CPI	1.00	1.09	1.07	1.14
MPI	1.00	1.20	1.28	1.76

Table 2: Difference between vCPUs

It should be noted that this behavior of consolidated workloads, where some workloads are cache-sensitive and others are cache insensitive makes asymmetric caches attractive as long as the VMM scheduler is optimized to be aware of cache asymmetry and VM resource usage.

D. Physical+Virtual Cache Asymmetry

In this subsection we will discuss the impact of virtual and physical cache asymmetry. The hardware was configured exactly the same way as in the previous subsection (with two sockets containing shared last-level caches of different sizes). We run the vConsolidate workload on 4 cores (2 cores sharing a last-level cache from each socket). Running the consolidated workload introduces virtual asymmetry while the different cache sizes in hardware introduce physical asymmetry. We have not used any special affinitization of the individual workloads, just ensuring that all the vCon workloads run on only 4 cores of the system.

Figure 11 illustrates this configuration with physical+virtual cache asymmetry.

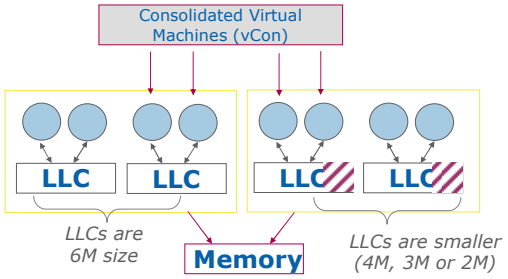


Figure 11: Consolidation & Virtual+Physical Asymmetry

For the charts below (Figure 12) , we observe that SPECjbb and Sysbench lose ~20% when run on a 6-2MB configuration as compared to a 6-6 configuration (both in consolidation mode). Webbench on the other hand does not lose much (<5%) at different cache sizes when it is run in consolidation on physically asymmetric caches. Figure 12 also shows the CPI and MPI behavior of these workloads as well. For SPECjbb, the CPI and MPI increases significantly when the level of asymmetry increases, whereas they increase moderately for Sysbench and Webbench.

As previously mentioned, this disparate cache usage behavior points to the potential for improving cache efficiency and die area by adopting asymmetric architectures. This, however, will require schedulers to be optimized to schedule cache-friendly workloads on large caches and cache-insensitive workloads to small caches. Cache monitoring hooks such as [33] that provide visibility into cache usage are highly useful for these purposes. In the next section, we will delve deeper into the inferences drawn from virtual and physical cache asymmetry in terms of the hardware and software optimizations required to support asymmetric architectures efficiently for both performance as well as quality of service.

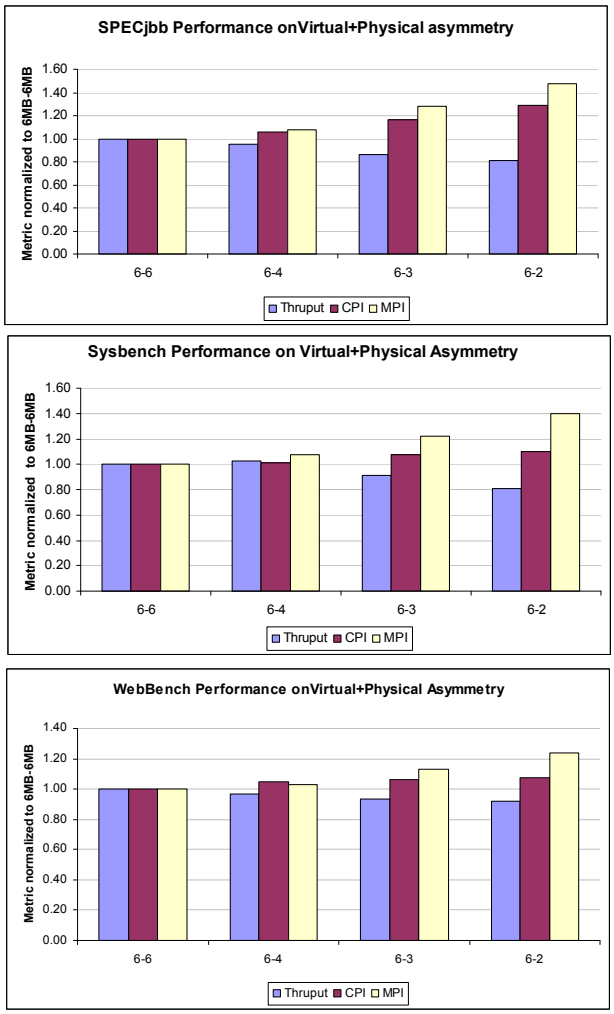


Figure 12: Virtual+Physical Asymmetry Implications

SPECjbb Thruput	6MB	4MB	3MB	2MB
Virtual	100%	85%	75%	67%
Physical	100%	98%	93%	85%
Virtual+ Physical	100%	95%	86%	81%

SPECjbb CPI	6MB	4MB	3MB	2MB
Virtual	100%	119%	137%	169%
Physical	100%	103%	113%	131%
Virtual+ Physical	100%	106%	117%	128%

SPECjbb MPI	6MB	4MB	3MB	2MB
Virtual	100%	131%	166%	209%
Physical	100%	107%	125%	164%
JBB Virtual+	100%	108%	128%	148%

Table 3: Comparing Virtual & Physical Asymmetry

VI. ASYMMETRIC CACHE INFERENCES

In this section, we discuss the inferences drawn from the two forms of cache asymmetry: physical cache asymmetry and virtual cache asymmetry. Table 3 sumarizes the performance differences observed when running SPECjbb in virtual and physically asymmetric cache architectures. While the inferences are drawn specifically in the context of server consolidation, it can be generalized in some cases.

A. Virtual Cache Asymmetry

In virtual cache asymmetry, we showed that running vCon on a set of symmetric shared caches can affect cache-sensitive workloads significantly via performance loss while cache insensitive workloads perform pretty much the same. In a virtualized data center scenario, where a number of applications are run in consolidation, it is important for the administrator to ensure service level agreements (SLA) which in turn requires performance isolation or performance differentiation between virtual machines running simultaneously on the same platform. One obvious method of meeting SLA's is providing the performance sensitive application with the resources it needs. This can be done either by modifying the underlying scheduler to be aware of interfence effects and ensuring that the performance-sensitive application is not running with applications that do not use much cache space or by introducing hardware quality of service techniques [15, 16, 24] described in recent literature. It is also important that the VMM get some visibility into the cache usage of the virtual machines much like what was proposed by Zhao et al. in [33].

JBB	vcpu0 (affinitized to 6MB)	vcpu1 (floating)	% benefit
CPI	1.51	1.80	19%
MPI	0.0051	0.0070	39%
Sysbench	vcpu0 (affinitized to 6MB)	vcpu1 (floating)	% benefit
CPI	2.51	2.96	18%
MPI	0.0016	0.0020	25%
Webbench	vcpu0 (6MB cache)	vcpu1 (floating)	% benefit
CPI	2.59	2.88	11%
MPI	0.0023	0.0026	11%

Table 4: Asymmetry-Aware Scheduler Optimizations

B. Physical Cache Asymmetry

In physically asymmetric cache architectures, it is important that the VMM scheduler be aware of the asymmetry as well as the cache usage behavior of workloads. To achieve this, we require both hardware and software support. The hardware support can be similar to mechanisms proposed by Zhao et al in [33] where occupancy in the cache is provided back to the VMM for each of the individual applications. The software support is the changes to the scheduler to take advantage of the resource usage information and appropriately schedule cache-sensitive workloads on core attached to larger caches and cache-insensitive workloads to cores attached to smaller caches. In order to highlight the benefits of this approach, we ran the consolidated workload on physically asymmetric cache architecture (with 6MB and 3MB shared last-level caches). Since SPECjbb2005 is known to be cache-sensitive, we affinitized one of the vcpus of SPECjbb2005 to the core attached to the 6MB cache and left the other vcpu floating amongst any of the other cores in the platform. Table 4

summarizes the effects of this affinitization and shows that vcpu0 (affinitized to 6MB) performs 19% better than vcpu1 (floating). We ran similar experiments with Sysbench and Webbench which are also listed in the Table.

VII. Conclusions and Future Work

In this paper, we evaluated different forms of cache asymmetry and its implications on workload performance. Furthermore, since server consolidation is a rapidly growing trend, we chose a recently defined server consolidation workload (vConsolidate) to evaluate the symmetric and asymmetric cache architectures.

We presented virtually and physically asymmetric caches architectures based on the sizes of caches as well as the sharing of caches between workloads. First, we showed that cache asymmetry can occur even in architectures that have symmetric shared caches in the hardware. The interference between multiple workloads running simultaneously in the cache causes virtual cache asymmetry and thereby introduces indeterminism in performance. Second, we evaluated physically asymmetric cache architectures that consist of caches of different sizes. Depending on the workload's use of cache space, we showed that the use of physically asymmetric caches may or may not affect performance. This observation motivates the potential of asymmetric cache architectures from an area/power/performance perspective.

The observations in this paper provide a basis for hardware optimizations (QoS techniques for virtually asymmetric caches) as well as software optimizations (scheduling techniques for asymmetric cache architectures). We showed simple case studies that highlight the potential for both hardware and software optimizations.

Future work in cache asymmetry is as follows. We plan to continue studying scheduler optimizations for physically asymmetric cache architectures. We also are studying QoS techniques for virtually asymmetric cache architectures. An overall solution with both scheduler optimizations as well as QoS techniques has the potential to improve cache efficiency as well as increase overall throughput and quality of service.

References

[1] AMD Inc., "AMD Multi-core Processors," *http://multicore.amd.com/en/*

[2] P. Apparao, R. Iyer, D. Newell, Towards Modeling and Analysis of CMP Server Consolidation, *2nd Workshop on Design, Architecture and Simulation of Chip-Multiprocessors (dasCMP 2007), Dec 2007.*

[3] P. Barham, et al. Xen and the Art of Virtualization. In *Proc. of the ACM Symposium on Operating Systems Principles (SOSP), Oct 2003.*

[4] L. Barrosso, K. Gharachorloo and E. Bugnion, "Memory System Characterization of Commercial Workloads," *ACM SIGARCH Computer Architecture News, June 1998.*

[5] J. P. Casazza, M. Greenfield, K. Shi, Redefining Server Performance Characterization for Virtualization Benchmarking, *Intel technology Journal, Volume 10, Issue 03*

[6] D. Chandra, F. Guo, et al. Predicting inter-thread cache contention on a chip multiprocessor architecture", *11th International Symposium on High Performance Computer Architecture (HPCA), Feb 2005.*

[7] L. Cherkasova and R. Gardner,"Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor," *Proceedings of the USENIX Annual Technical Conference, April 2005.*

[8] N. Enright Jerger, D. Vantrease, M. H. Lipasti, Evaluation of Server Consolidation Workloads for Multi-core Designs, *IISWC-2007*

[9] D. Gupta, R. Gardner,, L. Cherkasova; *XenMon: QoS Monitoring and Performance Profiling Tool*, http://www.hpl.hp.com/techreports/2005/HPL-2005-187.html

[10] Intel Xeon 5400 Series, ftp://download.intel.com/products/processor/xeon/dc54kprodbrief.pdf

[11] Intel Virtualization Technology Specification for the IA-32 Intel Architecture, April 2005 http://www.intel.com/technology/platformtechnology/virtualization/

[12] Intel Corporation. "Intel Dual-Core Processors," http://www.intel.com/technology/computing/dual-core/

[13] Intel Corporation, "World's first quad-core processors for desktop and mainstream processors," http://www.intel.com/quad-core/

[14] R. Iyer, M. Bhat, et al, *Exploring Small-Scale and Large-Scale CMP Architectures for Commercial Java Servers*, IEEE International Symposium on Workload Characterization (IISWC 2006), Oct 2006.

[15] R. Iyer. CQoS: A Framework for Enabling QoS in Shared Caches of CMP Platforms. In *Proc. of 18th Annual International Conference on Supercomputing (ICS'04)*, July 2004.

[16] R. Iyer, L. Zhao, et al., "QoS Policies and Architecture for Cache/Memory in CMP Platforms", *the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), June 2007*

[17] M. Karlsson, et al., "Memory System Behavior of Java-Based Middleware," *9th International Symposium on High Performance Computer Architecture (HPCA-9), Feb 2003*

[18] S. Kim, D. Chandra, and Y. Solihin. Fair Cache Sharing and Partitioning in a Chip Multiprocessor Architecture. In *Proc. of 13th Int'l Conf. on Parallel Arch. & Complication Techniques(PACT)*, Sept 2004.

[19] K. Krewell. Best Servers of 2004: Multicore is Norm. Microprocessor *Report, www.mpronline.com, Jan 2005*

[20] R. Kumar,D. Tullsen et al., Single-ISA Heterogeneous Multi-Core Architecturesfor Multithreaded Workload Performance, *In 31st International Symposium on Computer Architecture, June, 2004*

[21] M. R. Marty , M.D. Hill Virtual Hierarchies to Support Server Consolidation, *ISCA 2007*

[22] A. Menon et al. Diagnosing Performance: Overheads in the Xen Virtual Machine Environment. *In First ACM/USENIX Conference on Virtual Execution Environments (VEE'05), June 2005*

[23] M. K. Qureshi and Y. N. Patt. Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches *In Proc. of Annual Int'l Symposium on Microarchitecture (MICRO), June 2006.*

[24] N. Rafique, W.T. Lim and M. Thottethodi. Architectural Support for Operating System-Driven CMP Cache Management. *In Proc. of the 15th International Conference on Parallel Architectures and Compilation Technology (PACT 2006), Sept 2006*

[25] P. Ranganathan and N. Jouppi. Enterprise IT Trends and Implications on Architecture Research. *In Proc. of the 11th International Symposium on High Performance Computer Architecture (HPCA), Feb 2005*

[26] M. Rosenblum, T. Garfinkel. Virtual Machine Monitors: Current Technology and Future Trends. *IEEE Trans on Computers*, 2005.

[27] SPECjbb2005, *http://www.spec.org/jbb2005/*

[28] Sysbench *http://sysbench.sourceforge.net/*

[29] R. Uhlig, et al., "Intel Virtualization Technology," *IEEE Computer, 2005*

[30] VMware: www.vmware.com

[31] Webbench *http://cs.uccs.edu/~cs526/webbench/webbench.htm*

[32] Xen: The Xen Virtual Machine Monitor. *http://www.cl.cam.ac.uk/Research/SRG/netos/xen/architecture.html*

[33] L. Zhao et.al CacheScouts: Fine-Grain Monitoring of Shared Caches in CMP Platforms, PACT 2007