Performance Impact of Virtual Machine Placement in a Datacenter

Indrani Paul, Sudhakar Yalamanchili
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, USA
e-mail: {indrani, sudha}@ece.gatech.edu

Lizy K. John
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, USA
e-mail: ljohn@ece.utexas.edu

ABSTRACT - In virtualized systems, several Virtual Machines (VM) running on a single hardware platform share and compete for the hardware resources such as memory, disk and network IO to meet a certain Quality of Service (QoS) requirements. It is critical to characterize and understand how the different workloads running in the VMs interact and share such resources to be able to map them efficiently onto processor cores and server hosts for optimal performance. This is especially important for resources such as memory controllers or the on-chip or inter-socket networks for which there is currently no software control.

In this paper, we present a measurement-based performance analysis of server virtualization workloads from a real system using virtual machines that are part of the popular industry standard VMmark benchmark, a server consolidation benchmark. First, we characterize the relative resource contention and interference impact of VMs when multiple virtual workloads are run together. Second, we study the effects of co-locating different types of VMs under various VM to core placement schemes and discover the best placement for performance. We observe performance variations from 25-65% for Database servers and from 7-40% for File servers when compared to standalone VM depending on the placement of these VMs onto cores and the degree of sharing of resources. Finally, we propose an interference metric and regression model for the worst set of co-located VMs in our study. Based on different VM placement schemes we show that the overall server consolidation performance in a virtualized host can be improved by 8% when the VMs are placed effectively.

Keywords - Virtual Machine Placement, VM Scheduling, VM Performance, Workload Characterization, VM Mapping

1. INTRODUCTION

Virtualization technology adoption continues to grow in the enterprise segments across all types of workloads such as web hosting, data centers and even desktop computing. The emergence of chip multiprocessors (CMP) and the continuous increase in the number of cores in today's CMP architecture provide more hardware parallelism in a server platform making server consolidation very attractive to improve resource efficiency [3][12]. In addition, increased memory capacity per node has extended consolidation opportunities to workload mixes previously considered infeasible, for example database workloads. Consequently, virtualization is a key technology for cloud computing.

With virtualization several virtual machines (VM) are consolidated to run on a single machine (called a host) with the aid of a Hypervisor or Virtual Machine Monitor (VMM) layer. The hypervisor has access to all the underlying hardware resources and it multiplexes those across the VMs in a way such that the guest OS running in VMs has the illusion of running on its own physical machine. In such an environment those VMs share and compete for the hardware resources such as CPU, memory, disk and network IO. It is critical to characterize and understand how the different workloads running in the VMs interact and share the resources to be able to place them efficiently. This is especially important for resources such as memory controllers or the on-chip or inter-socket networks for which there is currently no software control. This knowledge naturally extends to decisions around which "types" of workloads to consolidate on a single machine to achieve the best performance.

Application interference is highly present in today's datacenter and is only going to get worse. So far there have been a few limited studies in characterizing the implications of cache sharing and cache size sensitivity [1][2] for individual VMs from the perspective of server consolidation workloads. There is also prior work around characterizing datacenter applications [8][25]. However, there has been a dearth of measurement-based studies that can provide some understanding of the potential interference between classes of VMs. In this paper we characterize and examine real-life large-scale applications running on existing hardware with a hypervisor layer to measure such interferences in a virtualization system and provide practical insights on how to place VMs on deployed systems. This cannot be accomplished by simulations where majority of the prior studies have focused. As part of this study we use the industry standard VMmark benchmark [14] – a consolidation benchmark.

Our paper specifically makes the following contributions:

- We present a detailed measurement based performance characterization of the typical server virtualization workloads.
- We show VM workload interactions and interference under different degrees of resource sharing.
- Based on the preceding observations, we show how workloads can be consolidated to improve performance in a shared environment such as shared host or datacenter.
- We discuss potential hypervisor and hardware optimizations for improving VM placement by detecting and reducing interference due to shared resources.

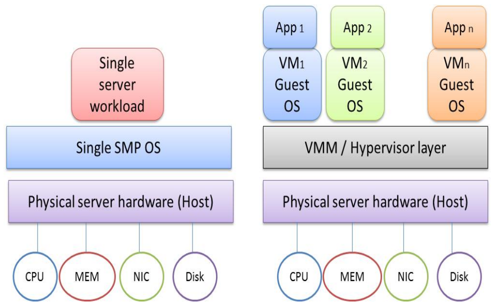

Figure 1: Virtualization and Consolidation Architecture

The rest of the paper is structured as follows: Section 2 discusses related work; Section 3 provides virtualization overview and discusses the resource sharing issues in modern CMP architecture. In Section 4 we define placement of a VM, in Section 5 we describe the experimental setup and measurement methodology. Section 6 presents results and Section 7 provides conclusion.

2. Related Work

Underutilization of multi-core resources has always been a problem in single workload driven traditional OS environment. To improve resource utilization, virtualization of multiple VMs and workloads onto the same host with the aid of Hypervisor has been the recent trend [3] [12].

Studying the interference of applications under shared resources was done in [13] but their approach differed from ours in two aspects: 1) it did not address VM mapping and placement; 2) it was not done on a virtualization benchmark which is more reflective of the consolidation workloads typically found in a datacenter. Apparao et al [1] and Enright et al [7] looked at cache contention of consolidation workloads to understand the impacts of various cache geometries on a workload and OS scheduling schemes. Along those lines there have been a number of studies in modeling cache utility and cache contention under virtualization [9].

Recently Tang et al conducted a study [25] on applications performance in a datacenter and their objective was to characterize the memory resource interference for optimal thread to core mapping. Other studies that characterize the interaction between emerging datacenter workloads and underlying architectures include research by Reddi et al. [20] and Soundararajan et al. [21]. Similar research is conducted in [10][16][17], however they do not characterize architectural interference. Hardavellas et al. [8] investigate sharing characteristics for database workloads and web-service workloads. Our paper is similar to these with respect to characterizing system level resource interference among workloads. However, we differ in that we examine virtualized workloads. Finally, resource sharing at a fine grain level in CMPs has been studied via simulation [4][6][11][18]. While insightful, we argue that these studies cannot directly address the problem of VM placement in modern data centers.

3. Server Consolidation Overview

Server consolidation is the process of encapsulating single server workloads into VMs and running them in a shared hardware platform via hypervisor – Figure 1. A recent survey [14] of datacenter applications show that some of the most common workloads targeted for virtualization are parallel computing applications, Databases, Web hosting, Mail Exchange and File hosting. These workloads are multi-threaded using some CPU, memory and IO.

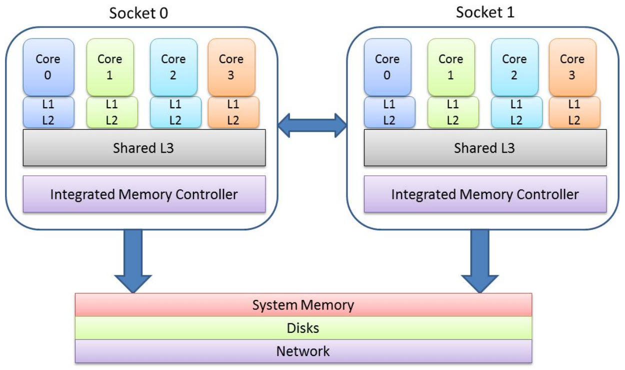

Figure 2: CMP Architecture of a 2 socket 4-core Intel Nehalem System

3.1 Sources of contention in CMP System Architecture

In modern CMP architectures we have multiple cores inside a processor socket. Typically each core has its own L1 and L2 caches whereas L3 (referred to as the Last Level Cache or LLC) is shared among all cores in the same socket. There are also other cache hierarchy variations such as private L2, L2 LLC or a shared L2 and L3 scheme. Figure 2 illustrates the CMP system architecture of a 2 socket 4 core Intel Nehalem [15] based server with shared cache, memory, network and disks. The source of interference could be from any component in the hierarchy such as Last-Level Cache, memory controller or network/IO

4. VM Placement

In the context of our paper we consider two types of host level VM placement decisions: 1) Hypervisor scheduling decisions of VMs onto physical cores/sockets in a host, 2) Datacenter placement decision around which workloads to consolidate together onto a given host

4.1 Hypervisor placement and scheduling of VMs

Modern Hypervisors like VMware ESX [26], Xen [3] and KVM [12] make effective usage of the multi-core architecture by scheduling workloads across several cores and sockets to improve overall system performance. Hypervisor scheduler places Virtual CPUs (vCPU) of a VM onto physical CPU core (PCPU). When different types of workloads are being scheduled onto the same host, the scheduler needs to decide how to place them for optimal VM performance. Such scheduling policies need an accurate understanding of interference patterns between VMs due to access to shared resources.

4.2 Datacenter Placement of VMs

The other aspect of VM placement that we address in our paper is which types of workloads can be consolidated together onto a single host for achieving the best performance. This decision is made by the Datacenter's IT infrastructure and is often based on categorization of server workloads based on resource requirements. For example, all IO centric VMs may be placed together. In our paper, we provide an analysis of the resource utilization and corresponding contention between virtual workloads to guide IT application optimizations for consolidation.

5. Experimental Setup and Methodology

In this section, we present i) our evaluation methodology, ii) the system setup we used, and iii) the virtualization benchmark and tools we used to gather data.

5.1 Platform Setup

We use a Dell PowerEdge R710 [19] mainstream dual-socket server based on Intel Xeon 5500 series processors (codenamed Nehalem-EP) [15] populated with total 32GB DDR3 1066MHz memory as an evaluation platform. Table 1 details the CPU/cache parameters and platform settings.

Table 1: Processor and Platform configuration parameters

CPU/cache parameters	Value	Platform Configuration	Value
Number of cores per processor	4	Hyper-threading	Disabled
Frequency	2.26GHz	Intel Vt (Virtualization Technology)	Enabled
L1 Instruction cache	32KB, 4 way	NUMA	Enabled
L1 Data cache	32KB, 8 way	Memory Configuration	Independent channel mode (mirroring disabled)
L2 cache	256KB, 8 way	Memory	32GB (4 8GB Hynix DDR3 1066MHz DIMMs)
Shared L3 cache (LLC)	8MB, 16 way	Storage	2 73GB SAS drives in RAID1 for ESXi
QPI	5.86 GT/s		3 300GB SAS drives in RAID5 for VMs
Integrated Memory Controller	3 DDR3 channels	Power/Performance Policy	Maximum Performance

Table 2: One VMark Tile

Virtual Machine	Guest OS	Application	# of vCPU	Memory	Disk Space
Java Server	W2K3 64 bit	SPECjbb2005 - based	2	1024MB	8GB
Database Server	SLES 10 64 bit	MySQL	2	2048MB	10GB
Web Server	SLES 10 64 bit	SPECWeb2005-based	2	512MB	8GB
File Server	SLES 10 32 bit	Dbench	1	256MB	8GB
Mail Server	W2K3 32 bit	Exchange 2003	2	1024MB	24GB
Standby Server	W2K3 32 bit	None/Idle	1	256MB	4GB

5.2 Virtualization Benchmark and VM Setup

We run the latest release of VMware ESXi 5.0 [26] for our VMM/Hypervisor layer to host all the virtual machines. For our workloads we use industry standard *VMmark* virtualization benchmark, specifically VMmark 1.1.1 [14]. VMmark tests system performance by adding fixed load virtual machines instead of scaling up a single application to system saturation. Unlike traditional single system benchmarks that tune up their load generation against a single instance, VMmark increases the number of virtual machines until the system is capable of no more work. A single unit of workload in VMmark is called a tile comprising of 6 virtual machines as shown in Table 2.

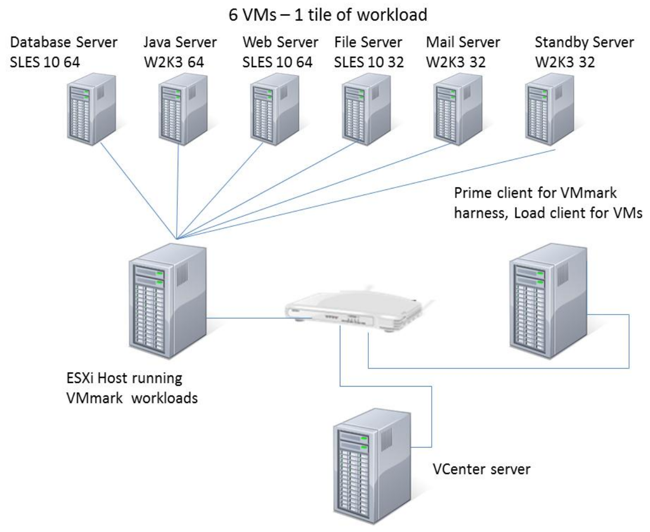
6 VMs – 1 tile of workload
Database Server SLES 10 64
Java Server W2K3 64
Web Server SLES 10 64
File Server SLES 10 32
Mail Server W2K3 32
Standby Server W2K3 32
Prime client for VMmark harness, Load client for VMs
ESXi Host running VMmark workloads
VCenter server

Figure 3: VMmark setup and components

The compute intensive Java VM runs a modified *SPECjbb2005* [23] to generate a steady load by setting the database size to a maximum (8 warehouses). In VMmark, resource consumption of SPECjbb is throttled by introducing a non-standard think time between transactions. The Database server uses MySQL as the underlying database whose size is fixed at 2.5GB. *Sysbench* [24] is used as the database workload that models users executing pre-defined transactions on the database. Web server runs a modified version of *SPECWeb2005* [22] with the Ecommerce test profile. File server uses the Dbench application to service requests from clients. *Dbench* is derived from the industry standard *NetBench* benchmark. Mail server is based on Microsoft Exchange 2003 running a load generating utility called *LoadSim* that simulates the users of the Exchange mail server. VMmark uses a fixed load by limiting the mail server configuration to 1000MMB3 (*MAPI Messaging Benchmark 3*) users. VMmark also introduces a standby server. This server is idle, does not run any workload and is not used towards any benchmark scores. However, it represents idle VMs ready to be deployed that are very commonly found in a datacenter. Even though they are sitting idle they still consume some minimal amount of resources which might impact other VM workloads.

Our objective is to analyze interference between VMs. Therefore we use one tile of VMs. The load generating client for these VMs is a Dell PowerEdge 1950 connected to the ESXi host through 1Gbps Ethernet. The same client is also used to run the benchmark harness software that can start, stop and measure the performance of the workloads – typically known as the "Prime client". Figure 3 shows the setup. For monitoring and management of the VMs we use VMware Vcenter server [26].

5.3 Evaluation Tools

For performance analysis we use *Vmkperf* included in the ESXi release. Vmkperf [5] is a command-line library and tool that provides an interface to access the hardware performance counters in the processor. We use Vmkperf to read architectural metrics such as unhalted clock cycles, instructions retired, last-level L3 cache miss etc. For CPU, network/IO, memory and vCPU usage we used a standard profiling tool *ESXtop* that is included in ESXi. We did not use the built-in poll functions in vmkperf because vmkperf requires the minimum poll rate to be 1 second. In the context of our work, this granularity was not sufficient to provide useful architectural insights into workload interactions and their impact on architectural metrics. So we implemented additional instrumentation in ESXi to poll the counters using vmkperf at a much faster rate of 30ms for data collection and logging. We came up with the poll rate empirically such that the fast poll rate does not introduce any performance overhead on the Hypervisor but at the same time provides useful trends of resource interference.

To study the performance impact of resource sharing in a controlled and isolated fashion, we compare the performance differences in an application under different VM to CPU static binding schemes as opposed to using dynamic allocation and migration schemes by hypervisor. This sheds light on how sharing of each type of resource impacts performance of various applications with different data sharing patterns. Example: the performance difference between {Web, Database} under inter-socket mapping reflects IO contention, whereas {Web,Database} under intra-socket mapping reflects cache, memory controller contention. We also bind the VM's memory usage to only local memory such that all the memory pages of a VM are physically located on the memory module connected to that core/socket. This was done using *Vcenter Server*.

Table 3: Application Metric used for each VM

Workload	Metric
Mail server	Actions/minute
Java server	New orders/second
Standby server	None
Web server	Accesses/second
Database server	Commits/second
File server	MB/second

5.4 Application Metric

Since the goal of our study is to characterize the workload's throughput profile over the entire benchmark execution in addition to the final throughput, we monitor the raw application metrics included in VMmark for each of the VMs periodically. This helps in correlating performance variations to dynamic changes in resource sharing. More details on VMmark metric calculations can be found in [14]. **Table 3** shows the application metric.

5.5 Measurement Configurations

We use the following test configurations in our analysis:

1) First, all workloads in one tile are run together under different VM placement/affinity schemes and their performance profiles are observed.
2) Next each individual VM workload is run by itself to obtain the workload's standalone performance metric. During this run all other VMs were shut down so that they were not consuming any resources.
3) After that we run VM workloads in pairs by affinitizing them to various combinations of cores and system memory to introduce different degrees of resource sharing for characterizing contention and interference.

We had some setup issues with Mail server due to which mail server will be omitted from our analysis.

6. Results and Analysis

Performance and architectural counters under VM workload consolidation are compared against the same when those workloads are run individually (standalone) on an ESXi host. We chose to compare with standalone virtualized performance instead of standalone native (non-virtualized) performance in order to maintain a common baseline (i.e.: workloads running on top of ESXi host) and also account for Hypervisor overheads. All experiments were run 5 times to ensure data consistency and repeatability. We group the analysis into two main sections. The first one discusses the overall performance impact from consolidation under different workload placement. The second section details the workload interactions, architectural responses and various degrees of interference under different placement of VMs. Lastly we present a summary of the main insights and observations in our study and propose an interference metric regression model.

6.1 Consolidation Performance Analysis

Here we compare consolidation performance where all the VMs are run together, with standalone VM performance when other VMs are shut down. We choose three different consolidation configuration placements for the VMs based on intra-socket and inter-socket pairing of the workloads as shown in Figure 4, to create different types of contention. When two VMs are placed in the same socket there is cache and memory interference. When two VMs are running in two different CPU sockets we mostly eliminate cache or memory contention (through affinities) but there is still system level network and disk IO resource conflict.

The application throughput for VM under these different placement schemes is shown in Figure 5. SPECJBB Java server's (JV) performance degradation is minimal when run in a consolidated workload environment (~1%). SPECWeb suffers around 3-4% performance degradation with consolidation. Database (DB) suffers 25%-65% performance degradation depending on workload placement. File server (FS) suffers 7-45% performance degradation depending on workload placement. FS performs the worst when paired with Web server (45% decrease), best when paired with Java (7% decrease). This shows that consolidation workloads can be placed intelligently to minimize performance degradation by understanding the effects of interference and resource (cache, memory, IO) contention. We also observe that the overall consolidation performance improves by 8% when Consolidation config. 1 placement profile is chosen over the other two consolidation schemes – Figure 6.

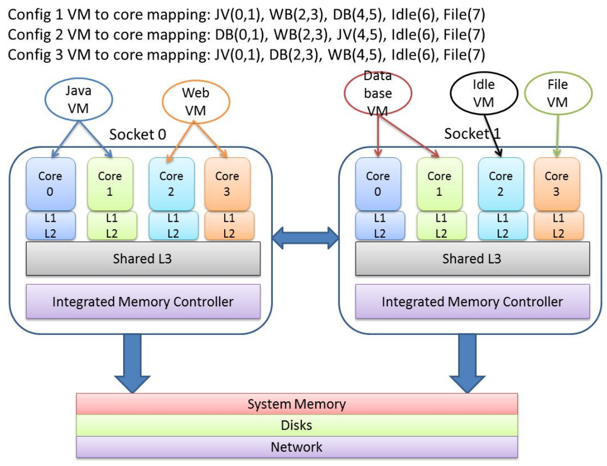
Config 1 VM to core mapping: JV(0,1), WB(2,3), DB(4,5), Idle(6), File(7)
Config 2 VM to core mapping: DB(0,1), WB(2,3), JV(4,5), Idle(6), File(7)
Config 3 VM to core mapping: JV(0,1), DB(2,3), WB(4,5), Idle(6), File(7)
Java VM
Web VM
Data base VM
Idle VM
File VM
Socket 0
Socket 1
Core 0
Core 1
Core 2
Core 3
L1 L2
Shared L3
Integrated Memory Controller
System Memory
Disks
Network

Figure 4: Three different server consolidation configurations

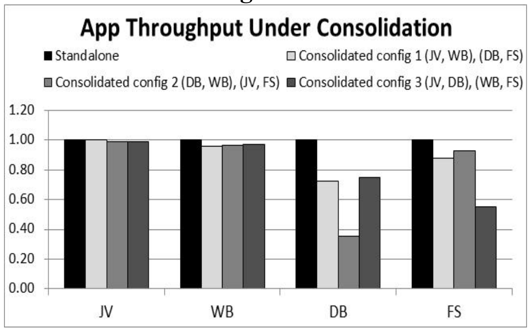

Figure 5: Application performance under consolidation

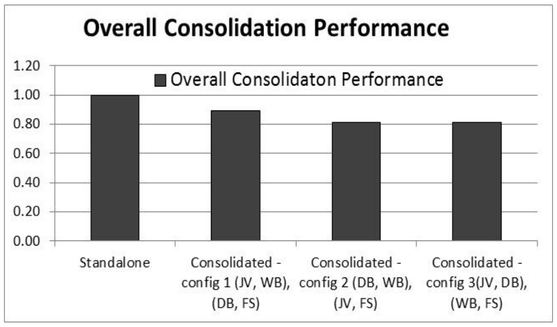

Figure 6: Overall consolidation performance under different placement profiles

The application throughput loss is further corroborated with their respective architectural metrics like L3 Miss per 1000 Instructions (MPKI) and Cycles per Instruction (CPI) in Figure 7. We find that for Java workload MPKI does not change significantly under the various consolidation profiles considered in our experiments as compared to a standalone Java VM run, resulting in minimal performance degradation under consolidation. Database workload Sysbench on the other hand has significant impact on its MPKI and CPI with consolidation. Under 2nd consolidation profile, Sysbench incurs a 74% increase in MPKI and 21% increase in CPI resulting in a 65% performance loss. This shows that the performance of server consolidation depends on many factors such as VM placement, which other VM workload it is sharing resources with, how much destructive interference is coming from the other VMs etc.

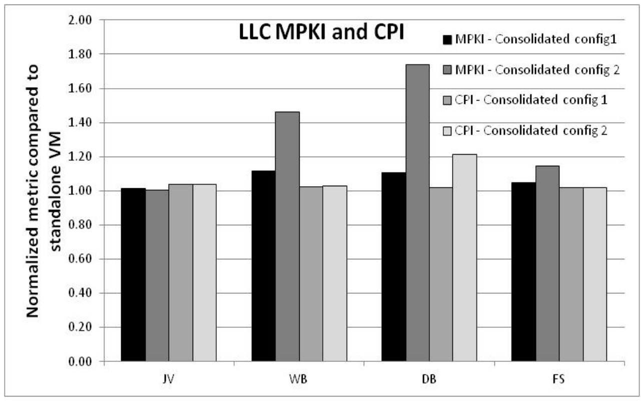

Figure 7: MPKI and CPI under consolidation

6.2 VM Interaction Analysis

In this section we present detailed analysis of the interference and contention encountered when two different VMs interact. For this we run VM workloads in pairs by affinitizing them to various combinations of cores and system memory to introduce different degrees of resource sharing. Several combinations of placement both intra socket and inter-socket were tried out. Due to space constraints we present a representative subset.

6.2.1 Java and Web Server

Java server running SPECjbb is a compute intensive VM with approximately 75-80% CPU utilization in this environment. It does not issue any significant IO accesses. Web Server running SPECWeb is a network intensive, latency sensitive VM with high CPU utilization. For this experiment we first affinitize or place SPECjbb and SPECWeb workloads onto cores from the same CPU socket. We also fix the memory affinities such that VMs use only local memory attached to its CPU.

We find that L3 access rate for Java with Web remains the same as a standalone Java run but L3 miss per 1000 cycles increases by about 20% because of the L3 cache interference caused by the two VMs. This causes a minor increase in Java's CPI and L3 MPKI (<1%), because even if L3 miss rate is high memory related stall time is only a small fraction due to the compute bound nature of this workload. The impact on application throughput over the actual execution period is therefore insignificant (<0.04%). Figure 8 shows the average profile of various architectural metrics for SPECjbb when run with Web VM during the stable period of the workload execution.

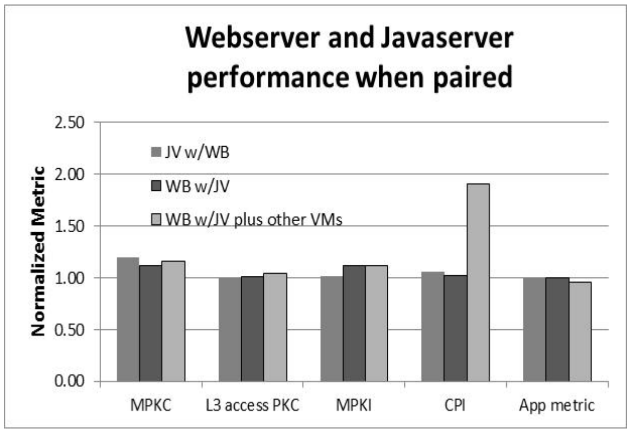

Figure 8: Architectural performance counters and Application Throughput of Java-Web pair

Similarly, for Web server we achieve good performance isolation. This is further correlated by minimal CPI change in Web server (<1%) when compared to a standalone Web VM run as seen in Figure 8.

Next we keep Java and Web mapped to the same socket, however to induce IO sharing we place Database and File Server in the other CPU socket to co-exist with Web (Configuration: Java: core 0, 1 and Web: core 2, 3 running on CPU socket 0, Database: core 4, 5 and File: core 7 running on CPU socket 1). We find that an intra-socket pairing of Web with Java in the presence of other workloads running on the 2nd CPU socket results in at least a 4% performance degradation in Webserver's performance due to shared network IO conflicts, as seen in Figure 8.

Summary: When a compute bound VM such as Java and an IO intensive VM such as Web are co-located or placed onto the same CPU socket they achieve good performance isolation. These two types of VMs are not sensitive to each other and do not create significant resource interference. Datacenter infrastructure and Hypervisor scheduler can deploy these workloads onto the same processor socket without significant performance degradation. However when this VM-pair co-exists with other IO intensive VMs in the same host there is IO resource contention and thereby throughput degradation is incurred for the IO intensive VM.

6.2.2 Web and Database Server

Database is heavy on memory and disk IO, whereas Webserver is intensive on network IO. We map Database and Web to induce memory sharing by placing them onto cores from the same CPU socket.

When Web and Database co-exist in the same host, specifically mapped onto the same CPU socket sharing cache and memory resources, there is heavy interference. In fact Web-Database pair has the worst increase in L3 misses (Figure 9) among all the other VM workload pairs in our study. The total LLC miss rate generated by this workload pair from the four cores is significantly higher than sum of the individual VM's miss rate in a standalone configuration. Web server's CPI increases by 3%, performance degrades by 4% compared to its standalone performance. Database server's CPI increases by 22%, performance degrades by 65%.

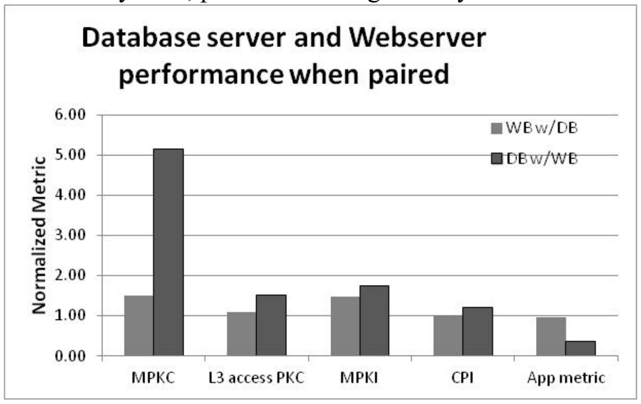

Figure 9: Performance counters and Throughput of Web when run with Database

Summary: Our study indicates that the Web and Database VM pair generates a lot of LLC cache pollution and destructive interference resulting in heavy memory accesses in addition to the shared IO. Hence this pair is not a suitable candidate for consolidation. In section 6.4 we do further characterization and regression of this heavy LLC interference. If the datacenter infrastructure needs to consolidate these two types of workloads onto the same host, Hypervisor should always schedule them on two different CPU sockets to minimize resource contention.

6.3 Observations and Insights

We find out from this study that VMs can be placed intelligently onto cores to minimize performance degradation by characterizing and measuring the effects of interference and resource contention. Some VM pairs are sensitive to each other creating destructive interference whereas we can achieve performance isolation with other VM pairs. This section details the summary of our analysis and lists some of the key insights.

***Observation 1**:* Our results show that compute intensive workloads such as Java scale very well in a virtualized infrastructure because they are less sensitive to the other VMs and rely mostly on the processing power of the cores. Memory related stall time is less for these VMs and good performance isolation is achieved by Hypervisor scheduler as long as CPU cores are not oversubscribed. These types of workloads can be easily consolidated in a datacenter.

***Observation 2**:* In a server workload consolidated infrastructure it is good to place CPU intensive VMs with IO intensive VMs on the same CPU socket or host to minimize resource sharing and contention. This can be seen from performance gains achieved by an IO intensive VM such as Database, File or Web when placed with Java, Figure 10 for example. This is often not practiced in a virtualized environment because the tendency is to place "like" workloads together for ease of deployment, manageability and chargeback. This method can also be used towards making decisions around load balancing and live migration such that resources are well distributed.

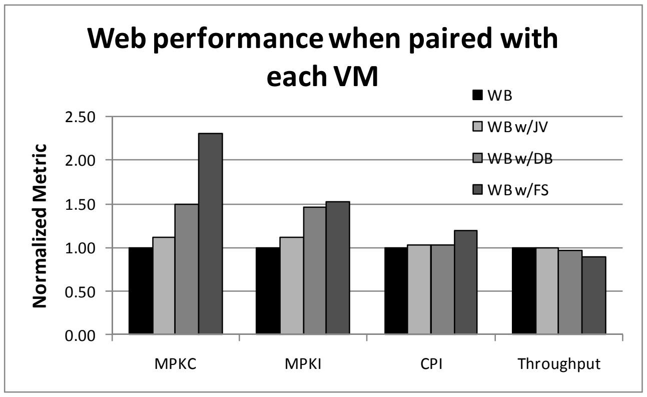

Figure 10: Web Server's overall performance when paired with each of the other VMs

Observation 3: We have found that L3 cache and memory controller contentions are much more critical to VM performance when compared to IO sharing. For example, in the presence of Webserver as its co-runner, Database performance improves to a large extent when it is placed such that memory contention between Web and Database is eliminated. There are two underlying reasons behind why Database server performs so poorly when co-located with Web server on the same CPU socket: 1) Web server creates destructive interference at LLC by causing large increase in L3 cache misses for Database server. This increases the memory latencies for Database server and hurts performance. 2) Memory controller sees a large increase in memory requests generated by the Web-Database pair (Figure 11). It has to schedule those requests contending for memory accesses. However, currently memory controller has no visibility to VM's QoS requirements, leading to unfairness, increase in memory stall-time and loss of throughput. Similar lack of hardware-software controls for the network-interconnects create poor network IO performance in the case of resource sharing. These indicate the need for future optimizations in Hypervisor and hardware to have better control on system level shared resources such as memory and IO

We also observe that Database server does relatively worse with consolidation compared to the other VM types. Hence attention should be paid to Database Server when virtualized to minimize resource interference.

To analyze the relationship between performance and cache/memory interference we look at the total number of LLC misses generated by each VM pair and the overall performance of the pair. Figure 11 shows the impact of a given VM pair on the total increase in LLC misses generated by all cores mapped to that VM pair. It confirms that when an IO intensive VM is paired with Java, the number of memory requests (LLC misses) is significantly smaller compared to when the same VM is paired with another IO intensive VM such as Web Server. It also shows that LLC interference and increase in memory accesses to the memory controller is a good indicator of VM's performance. The best performing pair Java-Web (overall performance is same as standalone performance) also has the least LLC interference. The worst performing pair Web-Database (overall performance is 59% compared to standalone) has the highest increase in LLC misses.

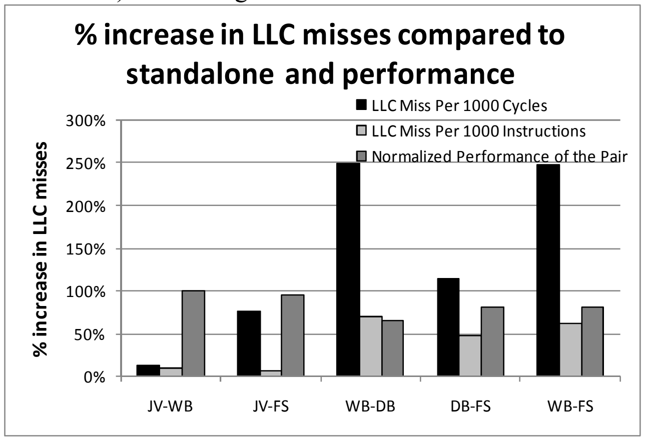

Figure 11: Overall performance and % increase in total LLC misses generated by all cores mapped to a VM pair, as compared to the same when running the VMs individually

6.4 Performance Interference Model

The previous section confirms that the Web-Database VM pair generates the worst case cache/memory interference among all the other VM-pairs. This results in the worst database performance loss of 65% when compared to running Database VM alone. However Web Server suffers only 4% performance degradation. Due to this unbalance in performance loss we term Database server as the *victim VM* and Web server as the *aggressor VM*. To further analyze this performance bottleneck in Web-Database pair, we perform linear regression on the measured dataset.

We propose an interference metric for the Database workload as a function of the L3 misses generated by the Web server when the pair is run together. Specifically, we define the metric such that L3 miss rate for Database with Web can be expressed as a time-varying factor added to the standalone L3 miss rate for Database. Mathematically this is expressed as below, where, victim is Database, aggressor is Web, and the interference function is *f(LLC_miss_rate_aggressor_VM)*.

$$LLC\_miss\_rate\_victim\_VM = LLC\_miss\_rate\_standalone + f(LLC\_miss\_rate\_aggressor\_VM)$$

Hence, *f(LLC_miss_rate_aggressor_VM) = LLC_miss_rate_victim_VM - LLC_miss_rate_standalone*

To solve this interference function we perform linear regression analysis of the L3 miss rates for the two workloads. We observe a very high degree of correlation between the two rates for these two VMs (with 0.91 R2 coefficient of determination). Figure 12 shows the regression model. Such performance interference metric will be useful for future offline and online optimizations in Hypervisor scheduling as well as hardware to predict performance of a VM pair under a given placement scheme and hence arrive at the optimal placement. The regression correlation can be improved in future by including other system level interference factors such as disk reads and writes, latencies, page swaps etc but our initial goal was to make the model easy to implement in hypervisor for online calculation of interference.

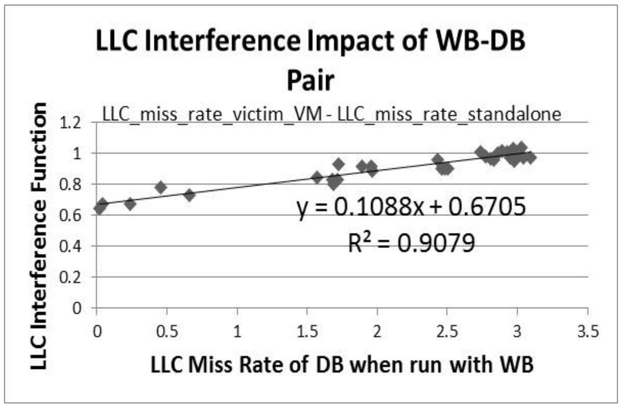

Figure 12: LLC Interference Model for WB-DB pair

7. Conclusions

Current technologies do not always provide performance isolation in a virtualized setup, which can have significant adverse effects on overall system performance. In this paper, we collected measurements of key performance characteristics of several standard virtualized workloads across different virtual machine placement schemes in a datacenter host. We analyzed the data closely to characterize the performance interference generated by these workloads and showed how workloads can be consolidated to improve performance. We show that while compute bound workloads are less sensitive to the presence of other VMs and easily virtualized, there are still resource bottlenecks in memory and IO leading to poor performance isolation of heavy memory, IO intensive VMs. In future, better controls are needed in Hypervisor and hardware layers to manage such shared resources.

8. REFERENCES

[1] P. Apparao, Ravi Iyer and D. Newell, "Implications of Cache Asymmetry on Server Consolidation Performance," *IISWC 2008.*

[2] P. Apparao, R. Iyer, X. Zhang, D. Newell, T. Adelmeyer: Characterization & analysis of a server consolidation benchmark. *VEE 2008.*

[3] 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, *Proceedings of the nineteenth ACM symposium on Operating systems principles*, page 177. *ACM, 2003*.

[4] J Chan, L. John, D. Kaseridis , Modeling Program Resource Demand Using Inherent Program Characteristics, *SIGMETRICS'11*

[5] J. Du, N. Sehrawat, W. Zwaenepoel, Performance Profiling in a Virtualized Environment. *Proc. HotCloud 2010*

[6] E. Ebrahimi, C. Lee, O. Mutlu, and Y. Patt. Fairness via source throttling: a configurable and high-performance fairness substrate for multi-core memory systems. *ASPLOS 2010*.

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

[8] N. Hardavellas, I. Pandis, R. Johnson, N. Mancheril, A. Ailamaki, and B. Falsafi. Database servers on chip multiprocessors: Limitations and opportunities. *Proc. CIDR*, 2007.

[9] R. Iyer, L. Zhao, F. Guo, R. Illikkal, S. Makineni, D. Newell, Y. Solihin, L. Hsu, and S. Reinhardt. Qos policies and architecture for cache/memory in cmp platforms. *In SIGMETRICS '07*.

[10] A. Beloglazov, Rajkumar Buyya, Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Datacenters, *CCPE* 2012

[11] D.Kaseridis, F.Iqbal, J.Stuecheli, L.John, MCFQ: Leveraging Memory-level Parallelism & Application's Cache Friendliness for Efficient Management of Quasi-partitioned LLC, *PACT 2011*

[12] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. KVM: the Linux virtual machine monitor, *Linux Symposium*, 2007.

[13] Y. Koh, R. Knuaerhase, P. Brett, M. Bowman, Z. Wen, C. Pu, An Analysis of Performance Interference Effects in Virtual Environments, *ISPASS 2007*

[14] V. Makhija, B. Herndon, P. Smith, L. Roderick, E. Zamost, J. Anderson, VMmark: A Scalable Benchmark for Virtualized Systems, *Technical Report VMware-TR-2006-002,* Sep 2006

[15] D. Molka, D. Hackenberg, R. Schone, M. Muller, Memory Performance and Cache Coherency Effects on an Intel Nehalem Multiprocessor System, *PACT'09*

[16] X. Meng et-al, Efficient Resource Provisioning in Compute Clouds via VM Multiplexing, *ICAC 2010*

[17] J. Xu, J. Fortes, A Multi-objective Approach to Virtual Machine Management in Datacenters, *ICAC 2011*

[18] M. Qureshi and Y. Patt. Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches, *MICRO 39*, Dec 2006.

[19] R710:http://www.dell.com/downloads/global/products/pedge/en/server-poweredge-r710-specs-en.pdf

[20] V. Reddi, B. Lee, T. Chilimbi, and K. Vaid. Web search using mobile cores: quantifying and mitigating the price of efficiency. *ISCA '10*

[21] V. Soundararajan and J. Anderson. The impact of management operations on the virtualized datacenter. *ISCA*, 2010.

[22] SPEC, http://www.spec.org

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

[24] Sysbench http://sysbench.sourceforge.net/

[25] L. Tang, J. Mars, N. Vachharajani, R. Houndt, M. Soffa, The Impact of Memory Subsystem Resource Sharing on Datacenter Applications, *ISCA 2011*

[26] http://www.vmware.com/products/vsphere/esxi-and-esx/index.html