# Modeling Virtual Machine Performance: Challenges and Approaches

Omesh Tickoo
Intel Labs, Intel Corporation
omesh.tickoo@intel.com

Ravi Iyer
Intel Labs, Intel Corporation
ravishankar.iyer@intel.com

Ramesh Illikkal
Intel Labs, Intel Corporation
ramesh.g.illikkal@intel.com

Don Newell
Intel Labs, Intel Corporation
donald.newell@intel.com

## ABSTRACT

Data centers are increasingly employing virtualization and consolidation as a means to support a large number of disparate applications running simultaneously on server platforms. However, server platforms are still being designed and evaluated based on performance modeling of a single highly parallel application or a set of homogenous workloads running simultaneously. Since most future datacenters are expected to employ server virtualization, this paper takes a look at the challenges of modeling virtual machine (VM) performance on a datacenter server. Based on vConsolidate (a server virtualization benchmark) and latest multi-core servers, we show that the VM modeling challenge requires addressing three key problems: (a) modeling the contention of visible resources (cores, memory capacity, I/O devices, etc), (b) modeling the contention of invisible resources (shared microarchitecture resources, shared cache, shared memory bandwidth, etc) and (c) modeling overheads of virtual machine monitor (or hypervisor) implementation. We take a first step to addressing this problem by describing a VM performance modeling approach and performing a detailed case study based on the vConsolidate benchmark. We conclude by outlining outstanding problems for future work.

## Keywords

Virtualization, Consolidation, CMP, servers, performance analysis, measurement, modeling

## 1. INTRODUCTION

Traditionally, server platforms have been designed and evaluated based on individual parallel applications or benchmarks (TPC-C, TPC-E, TPC-W, SPECjbb, SPECjappserver, etc) as the focus. Developing the performance analysis capability for such commercial server workloads was by itself a significant challenge since the benchmarks had complex behaviors, required multiple client/server systems and were difficult to run on full-system simulators. However, industry and academic researchers coped with this problem by developing scaled-down execution-driven simulations [17] for these workloads, by trace-driven simulations [16] [18] [19] [25] where possible and also by developing analytical models driven by measurement/simulation experiments. Now, the recent emergence of virtualization [3] [23] [24] introduces another level of complexity to the problem of server modeling and performance analysis. As datacenters rapidly adopt virtualization [22] as a means to consolidate multiple applications on a server, it becomes critical that the performance behavior of virtual machines is well understood. In addition, it also becomes important that we develop the ability to estimate virtual machine performance on a datacenter server. In this paper, we discuss the challenges to virtual machine performance estimation and introduce potential approaches for appropriate metrics and modeling techniques for this purpose. The first challenge of virtualization was addressing the lack of a virtualization benchmark that can be used for consistent and repeatable server performance analysis. Recently, there have been specific industry benchmarks that have been developed (VMmark [10], vConsolidate [20]) to address this issue. In addition, there is also a SPEC committee [6] that is working on defining its first virtualization benchmark expected to release sometime in the future. In this paper, we focus our case studies on vConsolidate to show the challenges and the potential approach of server virtualization modeling.

The key considerations when modeling the performance of virtual machines (VMs) can be summarized as follows:

1. VM performance is not only dependent on its own characteristics, but also dependent on the interference caused by the other virtual machines running on the same platform with it. We need a method to capture the effect of these interactions.

2. The above interference can affect the use of (i) shared resources (e.g. core, memory capacity) that are visible to the operating system or virtual machine monitor directly or through performance counters and (ii) shared resources (cache space, memory bandwidth, etc) that are invisible to the operating system since they are transparent resources managed by the hardware. The modeling approach needs to be aware of both visible and invisible resource interference.

3. the specifics of virtualization technology (both hardware virtualization and software virtualization) and the scheduling disciplines adopted by the virtual machine monitor could be quite different on any given platform. The modeling approach needs to take into account the virtualization technology as well as the scheduling heuristics required (Figure 1).

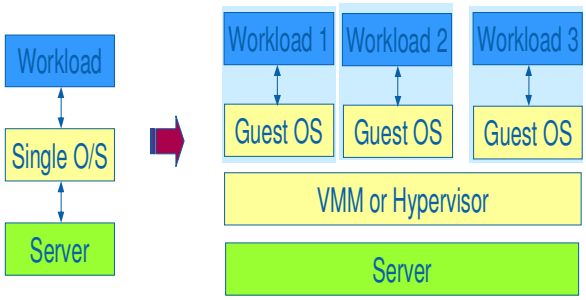


**Figure 1: From Dedicated Workloads to Consolidated Workloads.**

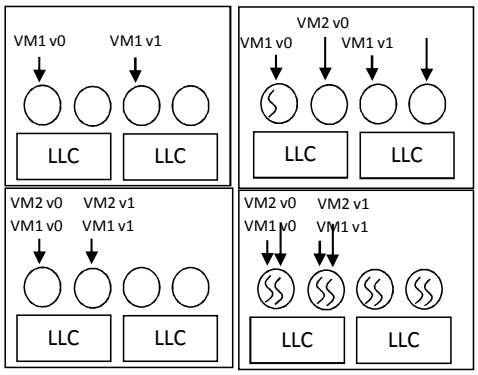


**Figure 2: VM Performance Scenarios**

In this paper, we will expand primarily on the resource interference effects and describe how a modeling approach can potentially take into account both visible and invisible resource contention effects on performance. Specifically, the contributions of this paper are as follows:

1. Using a virtualization/consolidation benchmark, we will show how VM performance depends heavily on visible and invisible resource interference caused by other VMs.
2. We will de scribe offline and online monitoring techniques needed to accurately characterize the behavior of multiple VMs sharing resources.
3. We will describe a potential modeling approach that employs either online or offline monitoring mechanisms to estimate the performance of the virtual machine.
4. We will outline a detailed list of next steps required to achieve a complete model for VM performance estimation.

## 2. VM PERFORMANCE EFFECTS

One of the key challenges when modeling the performance of virtual machines is that the application no longer has platform resources dedicated solely for its consumption. Figure 1 illustrates the transition from a dedicated execution (where one application used to run on a server previously) to a shared execution (where multiple guest OSes or VMs now run on the same server). Figure 2 illustrates a typical

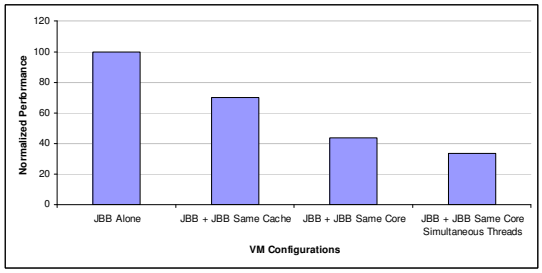


**Figure 3: SpecJBB Performance (1 copy vs 2 copies)**

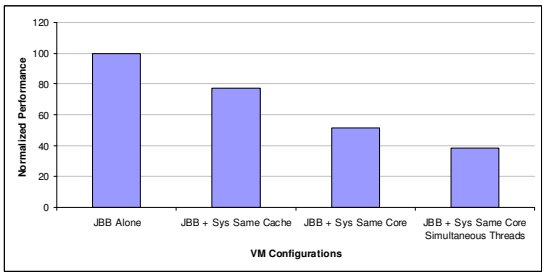


**Figure 4: SpecJBB Performance (Alone vs with Sysbench)**

server platform with multiple cores and the following four scenarios for the execution of a virtual machine (say VM1) with two virtual cpus (v0 and v1):

1. VM1 runs in dedicated mode with v0 running on c0 and v1 running c2. Since c0 and c2 have last-level caches of their own, these two virtual cpus do not share cache resources.
2. VM1 runs in shared mode with VM2 running on sibling cores (virtual cpus of these VMs share cache). This causes cache contention between the VMs and affects performance.
3. VM1 runs in shared mode with VM2 scheduled to run on the same core (time sliced scheduling). This causes core as well as cache contention between the two VMs and affects performance.
4. VM1 runs in shared mode with VM2 scheduled to run on the same core but on simultaneous threads made possible by hyperthreading [30]. This causes core contention (because of thread contention within the core for shared pipeline resources) as well as cache contention between the two VMs and affects performance.

To measure the performance in various scenarios, we ran experiments using virtual machines within the vConsolidate benchmark (consisting of SPECjbb [7], Sysbench [8], Webbench [11] and a mail server). We chose SPECjbb as VM1 as well as VM2 to collect the data shown in Figure 3. We also chose SPECjbb as VM1 and Sysbench as VM2 for the data collected in Figure 4. The first three configurations (illustrated in Figure 2) were collected on a Xen hypervisor [13] [12] running on a Core 2 Duo - based server platform [4] with two cores on each die sharing a last-level cache. The performance effect of the last configuration (effect of sharing simultaneous threads) was measured on the

latest Intel Nehalem-based server platform since the Core 2 Duo processor does not support threads, whereas future platforms are expected to. As shown in the Figure 3and Figure 4, the performance of the virtual machine is significantly affected by the sharing of cache and core resources. The performance can drop by as much as 20 to 30% due to cache contention alone. The performance further drops by another 30% when sharing cores. The performance can deteriorate further by 25% when multiple threads are being used on the same core. Overall, the performance of a virtual machine can vary from its performance during dedicated execution to its performance when running with other virtual machines either on a sibling core, a sibling thread, the same core or even the same thread. It should be noted that some of these resources are visible (e.g. core) and some are invisible (cache space, pipeline resources shared by simultaneous threads). Overall, the combined performance effect can be as high as 3X reduction due to resource contention. In addition, the performance depends on which virtual machine monitor you use.

# 3. VM PERFORMANCE MODELING APPROACHES

In the previous subsection, we showed how virtual machine performance can be significantly affected by resource contention with other virtual machine running along with it. In order to model the VM performance, it is important to accurately characterize all shared resource contention effects [14] [18] [21] in the platform. In this section, we will describe how this can be achieved for two modeling modes:

1. **Offline modeling:** This assumes that the workload performance can be measured on several platform configurations alone as well as pairwise with other virtual machines. It also assumes that the workload (each VM at a minimum) can be traced to run through core and cache simulators to collect behavioral information. Here, the intent is to fully calibrate a model in order to predict the performance of a virtual machine on a future server architecture or configuration.

2. **Online modeling:** This assumes that no offline analysis is available and the characterization of the VM performance effects needs to be done online (on production servers with perhaps some brief initial runs on test configurations within the datacenter). Here, the intent is to develop a model that will predict the performance of a VM on a different platform (if it was migrated to that platform within the datacenter).

In addition, it is important to consider the following key visible and invisible resources that will be contended for:

1. **Visible Resources:** Platform resources that are currently exposed (or visible) to the OS/VMM include core, memory capacity and potential I/O devices. In this paper we focus primarily on core usage since that is a dominant factor out of these three.

2. **Invisible Resources:** Platform resources that are currently transparent (or invisible) to the OS/VMM include shared cache space and core pipeline resources. In other words, the cache space occupied by one VM or another cannot be monitored today by an OS/VMM. Similarly, the pipeline resources used by one thread (running one VM) versus another thread (running another VM) cannot be observed today. The effect of cache and core contention (i.e. misses per instruction (MPI) and instructions per cycle (IPC) can however be obtained through performance monitoring counters).

The proposed approach for the VM modeling effort is the resource estimation of a virtual platform architecture that each virtual machine ends up with when running on a physical server platform. A virtual platform architecture (VPA) is defined as the set of resources used by (or solely made available to) the virtual machine of interest. The VPA resources include key resources such as number of cores, the core frequency and utilization, the cache space at each level and the memory frequency and bandwidth. In this paper, we focus on a VPA proof of concept that consists of one visible resource (the core) and one invisible resource (shared last-level cache space). In other words, the estimated VPA will produce (a) the number of cores and core utilization that a VM is given, (b) the cache space that a VM is given. Once the VPA core usage and the cache usage are available, the problem of estimating the performance of the virtual machine is mapped back to the simple problem of estimating the performance of a application running on a dedicated platform (with VPA as the dedicated platform).

### A. VPA Core Utilization Estimation

Since the core is a visible resource, the VMM can easily track the VPA core utilization on the platform for every VMM. The cost of tracking the core utilization [15] is negligible and provides an accurate accounting. However, in order to predict the core utilization on a different platform configuration, a simplistic approach to start with is as follows:

**(i) Monitor a VM's core utilization when running alone**(VMxAloneUtil)

- In an offline mode, this requires running the VM by itself on a platform and monitoring the core utilization using performance counters.

- In an online mode, this requires monitoring the VM core utilization and keeping track of the maximum core utilization (especially when no other VM is running).

**(ii) Predict a VM's core utilization on a different consolidated platform** by scaling down the above VMx core utilization by the ratio of number of available physical CPUs over the total utilization of all VMs on the target platform (VMxConsUtil)

$$VM_xConstUtil =$$

$$min[VM_xAloneUtil, \frac{VM_xAloneUtil * PhysCPUs}{VM_{all}AloneUtil}]$$

This simple model allows for reasonable estimation of the VPA core utilization of a VM. In the future, we plan to extend this simple model by taking VMM scheduling heuristics into account. For the vConsolidate measurement, we applied the estimation model to platforms configured with different cache sizes (1MB, 2MB, 4MB) as described earlier. We also

**Table 2: Execution Probability of VMs running cores sharing a L2 Cache with SPECjbb**

| L2 size, | SPECjbb(%) | sysbench(%) | webbench(%) | MMB(%) | Domain-0(%) |
|---|---|---|---|---|---|
| 4MB | 18 | 38 | 32 | 3 | 9 |
| 2MB | 37 | 19 | 31 | 3 | 10 |
| 1MB | 36 | 17 | 32 | 4 | 12 |

**Table 1: SPECjbb VPA Core Utilization**

| SPECjbb, | 4MB | 2MB | 1MB |
|---|---|---|---|
| Estimated(%) | 115 | 124 | 126 |
| Measured(%) | 122 | 124 | 121 |
| Error(%) | -5.58 | 0.39 | 3.91 |

instrumented the Xen VMM to measure the actual utilization of the SPECjbb virtual machine and compare it to the estimated utilization. Table 1 shows the utilization (in %) as well as the accuracy (computed as relative error). For example, the estimated utilization is 1.15 cores (115% utilization) for a 4MB configuration, whereas the measured utilization is 1.22 cores (122%) for the same configuration. With such a simplistic model, we find that the error is around 6% or less. The error is, in part, attributed to software scheduling that is not considered in the model above.

### B. VPA Cache Space Estimation

Since the shared last-level cache is an invisible resource and cannot be directly monitored by OS/VMMs, we propose the following approach to estimating VPA cache space (per VM) as follows:

**(i) Profile the execution of a VM by capturing the percentage of time it runs with other VMs on sibling cores (sharing the same cache).**

- One approach to this is to instrument the VMM at schedule/deschedule points to keep track of where a VM was running. However, this is useful only in hindsight for understanding the performance effects of other VMs in a platform configuration already established.
- In order to predict the performance on a future platform configuration (with different VMs), it is possible to statistically approximate the percentage of time that a VM runs with other VMs.

$$P(VM_x{+}VM_y) = \frac{VM_yUtil}{(VM_{all}Cons - Util - VM_xCons - Util)}$$

**Table 3: Overall Cache Contention Effect for Consolidation**

| Benchmark, | Estimated MPI | Measured MPI |
|---|---|---|
| SpecJBB | 0.0077 | 0.0085 |
| Sysbench | 0.0041 | 0.0043 |

Table 2 presents the fraction of SPECjbb VM's execution time that it spends running with another virtual machine contending on a shared cache. For example, a SPECjbb virtual cpu runs with a SysBench virtual cpu sharing the same shared cache for an estimated 38% of its overall execution

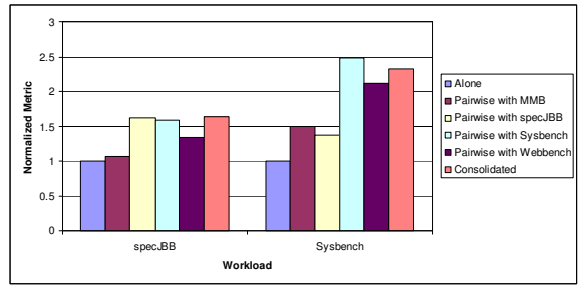


**Figure 5: MPI Increase in Pairwise/Consolidated Execution**

in the platform configuration with 4MB of shared cache. On the other hand, it spends an estimated 18% of its overall execution time running with the other SPECjbb virtual cpu in the same configuration. It should be noted that we have validated these fractions (and found them to be within 2% error) by profiling the execution of vConsolidate on Xen.

**(ii) Estimate the pairwise VPA cache space for a VM when it runs with another sibling VM (assuming two CPUs sharing a cache)**

- In the offline mode, it is possible to capture a trace for each VM and then run every pair of traces (for different VMs) through a shared cache model to create a table of VPA cache space usage per VM.
- In the online mode, it is not possible today to monitor the cache space usage directly. However, proposals like [25] that provide accurate cache space monitoring in future architectures might address this problem in the future. Instead, the VMM can monitor the effect of cache interference by keeping track of the misses per instruction (collected through performance counters) as a function of the other VMs running on sibling cores. Even here, it is required that this estimation be done on a test configuration where all pairs of VMs are run simultaneously and measured.

Figure 5 shows the effect on normalized MPIs (cache Misses-per-Instruction) of SPECjbb and sysbench from pairwise workload runs. The normalization is done with respect to the MPI values when the workload was running alone. As shown in Figure 5, for a 4MB cache configuration, one SPECjbb virtual CPU's MPI increases by as much as 62% when running with another SPECjbb virtual CPU, whereas it increases by 58% and 34% when running with Sysbench and Webbench virtual CPUs respectively.

**(iii) Estimate the consolidated VPA cache space that is a weighted average of the pairwise VPA cache space (with weights being the fraction of execution time).**

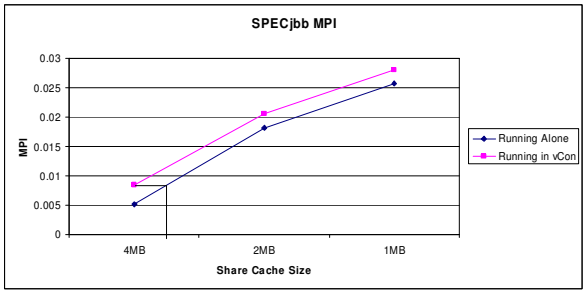


**Figure 6: SPECjbb cache occupancy analysis**

Table 3 shows that the estimated MPI (for SPECjbb and sysbench) comes reasonably close to the measured MPI during consolidation. The cache contention effect can also be translated to prediction of effective cache size by looking up the cache size based on offline MPI profiling experiments. For example, Figure 6 shows the MPI of SPECjbb running alone and in consolidated mode as a function of cache size. By looking up the miss rate curve (for 4MB), we can find the effective cache size ( 3.5MB).

**(iv) Use the estimated MPI to derive the CPI for the virtual machine.**
The cache contention effects can now be translated into estimated CPI values (cycles per instruction) for each virtual machine in the consolidated environment using the performance data and based on the VMA core utilization. Due to space constraints we skip the exact procedure here. Table 4 shows that the estimated CPI values match the measured values with low error ( 5%).

**Table 4: CPI Estimation for SPECjbb**

| Metric, | SPECjbb Single VM | SPECjbb in vCon |
|---|---|---|
| SpecJBB | 1.549 | 2.231 |
| Sysbench | 1.543 | 2.234 |

**(v) Estimate performance loss from cache contention (as a multiplier) as follows:**

$$CacheContentionPerfMultiplier = \frac{VM_x - Alone - CPI}{VM_x - Cons - CPI}$$

The factor calculated above represents the performance loss due to cache contention.

# 4. CONCLUSIONS AND FUTURE WORK

In this paper we presented the VM performance modeling challenges and highlighted that visible and invisible resource interference cause a significant performance degradation and need to be considered when modeling VM performance. We then proposed VPA estimation as an approach to modeling VM performance. Using a case study of vConsolidate (a server virtualization benchmark), we showed that VPA core and cache usage estimation is possible with offline models. With online models, only VPA core usage estimation is possible and cache usage estimation needs to be approximated as cache performance effects. Future work in this area is as follows. We plan to extend the VPA modeling approach to include other resources (core pipeline resources, memory bandwidth, etc). We also plan to test out the modeling approach on a wider set of configurations and put together a analytical model that can be calibrated for this purpose. Last but not least, we would also like to study other virtualization benchmarks and VMMs to ensure that this approach proves valid for a wider range of workload, VMM implementations and cloud computing scenarios [1] [9] [2] [5].

# 5. REFERENCES

[1] Amazon elastic compute cloud (ec2). http://www.amazon.com/ec2/.
[2] Hp utility computing. http://h71028.www7.hp.com/enterprise/cache/308072-0-0-0-121.html.
[3] Intel virtualization technology specification for the ia-32 intel architecture. http://www.intel.com/technology/platformtechnology/virtualization/.
[4] Intel xeon 5400 series. ftp://download.intel.com/products/processor/xeon/dc54kprodbrief.pdf.
[5] Microsoft live mesh. http://www.mesh.com.
[6] Spec. http://www.spec.org.
[7] Specjbb2005. http://www.spec.org/jbb2005/.
[8] Sysbench. http://sysbench.sourceforge.net/.
[9] Twenty experts define cloud computing. http://cloudcomputing.syscon.com/read/612375_p.htm.
[10] Vmware vmark. http://www.vmware.com/products/vmmark/results.html.
[11] Webbench. http://cs.uccs.edu/ cs526/webbench/webbench.htm.
[12] Xen, the xen virtual machine monitor. http://www.cl.cam.ac.uk/Research/SRG/netos/xen/architecture.html.
[13] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew 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.
[14] Dhruba Chandra, Fei Guo, Seongbeom Kim, and Yan Solihin. Predicting inter-thread cache contention on a chip multi-processor architecture. In *HPCA '05: Proceedings of the 11th International Symposium on High-Performance Computer Architecture*, pages 340–351, Washington, DC, USA, 2005. IEEE Computer Society.
[15] Ludmila Cherkasova and Rob Gardner. Measuring cpu overhead for i/o processing in the xen virtual machine monitor. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 24–24, Berkeley, CA, USA, 2005. USENIX Association.
[16] R. Iyer et al. Datacenter-on-chip architectures: Tera-scale opportunities and challenges. *Intel technology Journal*, Aug 2007. http://www.intel.com/technology/itj/2007/v11i3/6-datacenter/1-abstract.htm.
[17] Richard A. Hankins, Trung Diep, Murali Annavaram, Brian Hirano, Harald Eri, Hubert Nueckel, and

John P. Shen. Scaling and characterizing database workloads: Bridging the gap between research and practice. In *MICRO 36: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, page 151, Washington, DC, USA, 2003. IEEE Computer Society.

[18] Ravi Iyer. Cqos: a framework for enabling qos in shared caches of cmp platforms. In *ICS '04: Proceedings of the 18th annual international conference on Supercomputing*, pages 257–266, New York, NY, USA, 2004. ACM.

[19] Ravi Iyer, Li Zhao, Fei Guo, Ramesh Illikkal, Srihari Makineni, Don Newell, Yan Solihin, Lisa Hsu, and Steve Reinhardt. Qos policies and architecture for cache/memory in cmp platforms. In *SIGMETRICS '07: Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 25–36, New York, NY, USA, 2007. ACM.

[20] M. Greenfield J.P. Casazza and K. Shi. Redefining server performance characterization for virtualization benchmarking. Aug 2006.

[21] Seongbeom Kim, Dhruba Chandra, and Yan Solihin. Fair cache sharing and partitioning in a chip multiprocessor architecture. In *PACT '04: Proceedings of the 13th International Conference on Parallel Architectures and Compilation Techniques*, pages 111–122, Washington, DC, USA, 2004. IEEE Computer Society.

[22] Parthasarathy Ranganathan and Norman Jouppi. Enterprise it trends and implications for architecture research. In *HPCA '05: Proceedings of the 11th International Symposium on High-Performance Computer Architecture*, pages 253–256, Washington, DC, USA, 2005. IEEE Computer Society.

[23] Mendel Rosenblum and Tal Garfinkel. Virtual machine monitors: Current technology and future trends. *Computer*, 38(5):39–47, 2005.

[24] Rich Uhlig, Gil Neiger, Dion Rodgers, Amy L. Santoni, Fernando C. M. Martins, Andrew V. Anderson, Steven M. Bennett, Alain Kagi, Felix H. Leung, and Larry Smith. Intel virtualization technology. *Computer*, 38(5):48–56, 2005.

[25] Li Zhao, Ravi Iyer, Ramesh Illikkal, Jaideep Moses, Srihari Makineni, and Don Newell. Cachescouts: Fine-grain monitoring of shared caches in cmp platforms. In *PACT '07: Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques*, pages 339–352, Washington, DC, USA, 2007. IEEE Computer Society.