Profiling Memory Subsystem Performance in an Advanced POWER Virtualization Environment

Diana Villa, Mitesh Meswani and Patricia Teller
Department of Computer Science, The University of Texas at El Paso
{demarquez, mmeswani, pteller}@utep.edu

Bret Olszewski
IBM Corporation-Austin
breto@us.ibm.com

Abstract

Processor virtualization allows concurrent operating system execution environments to co-exist and share a fixed set of hardware resources. This allows higher system utilizations than possible by dedicating resources to particular environments. One of its many advantages is that this technology facilitates server consolidation, reducing both operating costs and power consumption. However, virtualization incurs a performance penalty due to added complexity and the conflicts arising from sharing a fixed set of hardware resources. It is important to identify and understand the sources of the overheads in order to guide tuning/optimization efforts to reduce these performance costs. In this research, we outline and use a performance evaluation framework and methodology that uses sampled event traces to identify and understand the virtualization overheads with respect to the memory subsystem performance.

1. Introduction

Processor virtualization facilitates the time-sharing of a fixed set of hardware resources by a number of different operating system execution environments. Virtualization technology dates back to the 1970s when virtual machine monitors (VMM), a.k.a hypervisors, became popular as part of the IBM 370 mainframe series of servers [4]. Since that time, the perceived value of virtualization declined with the introduction of low-cost mini- and personal computers. Currently, virtualization technology is re-emerging in new environments, such as UNIX-based systems, as a response to the challenge of reducing data center costs.

In this work, we study the POWER virtualization environment and the POWER hypervisor [12]. The POWER hypervisor resides in flash memory and provides the resource isolation that allows the physical hardware to be shared by independent execution environments. It supports up to 254 concurrent partitions, each of which can be running any of the supported operating systems [5]. This allows multiple, concurrent execution environments to co-exist and multiplex on the same set of hardware resources.

The POWER hypervisor supports two different kinds of partitioning: dedicated partitioning and micro-partitioning. The former assigns physical processors to a partition and does not allow the sharing of processors across partitions. It is supported by POWER4-based servers, while POWER5-based servers additionally support micro-partitioning, which assigns virtual processors (VPs) to a partition and are mapped to physical processors by the POWER hypervisor. Micro-partitioning supports the sharing of processing units across partitions and, therefore, allows more active micro-partitions than physical processors. On POWER5-based servers, I/O virtualization is possible using either type of partitioning.

One of the major benefits of virtualization is server consolidation. Consolidating multiple execution environments on a single hardware resource cuts down operating costs and reduces power consumption. This consolidation also allows increased security and higher availability [4].

Although beneficial in many ways, virtualization has an inherent performance penalty associated with it. This is largely due to contention for a fixed set of hardware resources, such as system caches, by the concurrent environments. Furthermore, the addition of a hypervisor layer to the system architecture increases complexity in that the hypervisor runs below each operating system and must provide isolation among the partitions. Though many aspects of virtualization have

direct hardware support, such as requiring that certain instructions can only be executed by the hypervisor, other resources such as interrupt management require direct participation of the hypervisor.

It is well documented that the memory subsystem is the major bottleneck in application performance [6]. For symmetric multiprocessor systems (SMPs), it often governs the performance of the whole system. Accordingly, this research looks at identifying and understanding the memory performance overhead incurred due to executing an application in a virtualized environment.

2. Related Research

Event tracing is a well known approach for obtaining detailed data about application execution. Lu and Reed [9] use a methodology based on curve fitting to obtain application signatures from event traces. This approach allows comparison of performance metrics across different hardware and software platforms. Performance effects of architectural modifications are studied by Barroso, et al. [2] using tools like IPROBE and DPCI (Dynamic Continuous Profiling Infrastructure) [1, 3] to capture event traces of applications executed on a four-processor AlphaServer 4100 using Oracle 7.3.2. Behavior of an OLTP workload is studied by Keaton et al. [8] using performance monitors on a four-processor Pentium Pro-based server. Barroso et al. use source code instrumentation and simulation methodologies to characterize workloads, whereas, Keeton, et al. characterize workloads by physically changing the hardware. Desikan, et al. [3] also use DPCI tool to quantify the reliability of an Alpha 21264 simulator by sampling certain events that are used to derive performance measurements for a Compaq DS-10L workstation.

To the best of our knowledge, event tracing has not yet been used to study the performance of virtualized environments. And, virtualization in its present form is a new area of research and its emergence is discussed by Figueiredo, et al. [4]. This paper describes our work in this area, i.e., the use of event tracing to study the performance of virtualized environments. This endeavor builds upon our previous work [10, 11], which explores the use of sampled event traces for profiling memory subsystem performance of workloads executed on IBM eServer pSeries p690 systems.

3. Data Collection

The performance data collected in this study are traces of sampled events associated with data-load hits generated by TRADE3 while executing on two different configurations of a four-processor IBM eServer pSeriers 570 (p570) systems. Below, in addition to discussing the TRADE3 workload and the experimental platforms, we describe the monitored events and the methodology used to collect and analyze the event traces.

3.1 Experimental Platform: IBM eServer pSeries 570

The research and data collection in this study were performed on a four-processor p570 [7], a symmetric multiprocessor (SMP) architecture. The basic building block of the p570 is a Dual Chip Module (DCM), which contains one POWER5 chip and an L3 cache chip. Each POWER5 chip is comprised of two 1.65 GHz CPUs. Each CPU has a 32KB level-one (L1) instruction cache and a 64KB L1 data cache; the two CPUs on a chip share a 1.9MB unified level-two (L2) cache. Each DCM has a 36MB unified level-three (L3) cache and 8GB of memory, giving a total of 16GB for the machine. The p570 system used in this study has four processors and two DCMs; it runs the AIX version 5.3 operating system. The conceptual layout of a p570 is given in Figure 1.

The POWER5 storage structure is distributed memory architecture. Each processor can address all memory and sees a single shared-memory resource. As such, a single DCM and its L3 cache and memory are packaged on a single card. Access to memory associated with another DCM is accomplished through the fabric buses.

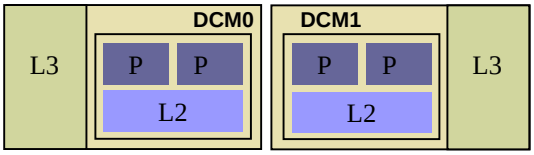

Figure 1. Two-DCM p570 configuration

The experiments discussed in this paper were carried out in the two different machine configurations described below:

- One Partition (1P) Configuration: In this case, the application has a dedicated partition with all four processors allocated to this partition. This is the non-virtualized environment.
- Five Partition (5P) Configuration: In this case, the p570 is micro-partitioned into five partitions; each partition is assigned two virtual CPUs and runs one copy of the TRADE3 benchmark.

3.2 Workload: TRADE3

TRADE3 is the third generation of the WebSphere end-to-end benchmark and performance sample

application. It models an online stock brokerage application and uses a real world workload that drives WebSphere's implementation of J2EE 1.3, Web Services, and other key Web Application Server components. TRADE3 execution involves three components: a client application that generates the workload, a database (in this case DB2), and a Web Application Server (in this case WebSphere). In this research the database and Web Application Server run concurrently in each partition, and the client application drivers are independent external systems and not part of the measured environment.

3.3 Monitored Events

The p570 has three levels of cache and a shared main memory. A data-load miss in an L2 cache can be serviced by the L2 cache of another DCM (L2.75 hit), the L3 cache on its (the miss-generating) DCM (L3 hit), the L3 cache on another DCM (L3.75 hit), or main memory. Memory hits can be further classified as either local memory (LMEM) hits or remote memory (RMEM) hits, depending upon the location of the memory module where the referenced data was found.

Cache hits can be further classified by the state of the cache line. Accordingly, a cache hit is either a shared hit or a modified hit.

The data collected for this study includes the following events:

- L2 data-load hits
- L2.75 shared data-load hits (L275_SHR)
- L3 data-load hits
- L3.75 shared data-load hits (L375_SHR)
- L3.75 modified data–load hits (L375_MOD)
- Local memory data-load hits (LMEM)
- Remote memory data-load hits (RMEM)

The approximate latencies associated with each event are found in Table 1 below.

Event	Load Latency
L2	14 cycles
L275	121 cycles
L3	91 cycles
L375	205 cycles
LMEM	281 cycles
RMEM	307 cycles

Table 1. Approximate Data-load Latencies

3.4 Sampled Event Traces

The POWER5 microprocessor includes Performance Monitoring Unit (PMU) counters that permit up to six concurrent events to be monitored. The PMU is capable of storing aggregate counts and capturing event records, which include instruction and data addresses associated with events.

In this work, an in-house IBM tool called eprof, which uses the AIX operating system's time-based profiling tool, tprof, is used to program the PMU to sample hardware countable events at a defined target rate. The performance monitor allows samples to be collected using a time-based approach, or based on counts of specific events.

Sample information is recorded upon the periodic occurrence of the monitored event. The information gathered indicates the timestamp of the event, the effective instruction address and for load/store operations, the data address of the instruction, the process and thread IDs of the software entity that triggered the event.

Time based profiling is accomplished if the event being monitored is processor cycles. In contrast, if the event is a variable event, such as cache misses, and if the rate of occurrence is greater than the sampling rate, then eprof adjusts the sampling rate to approximate the specified sampling rate. Accordingly, a different number of samples may be collected for different types of events.

When an event is sampled, the sample information and an AIX trace hook, which identifies the trace record, is used to generate a trace record and write the record to a file. A trace formatting tool, called trcrpt, creates a time-stamped text file of events.

For this research, 100 events per second per CPU were collected for a 120-second interval of the steady-state execution of TRADE3. The size of the data set collected for the monitored events given in Table 2.

Event	Sample Count	
	1 Partition	5 Partitions
L2	64139	30301
L275_SHR	21774	3486
L3	60772	28114
L375_MOD	23653	7280
L375_SHR	7843	10987
LMEM	7221	13427
RMEM	7888	13580

Table 2. Event Sample Counts

3.5 Performance Analysis Toolkit

Figure 2 depicts our Performance Evaluation Framework, which includes a toolkit implemented in Java that is used to process sampled events and store them in a MySQL database according to the workload being monitored, the number of processors, and the event being sampled. Each database has 12 tables that store information related to an experiment. Once an experiment's events are loaded into a database, a second set of Java tools is used to query the database and generate reports in text format. Text reports are

imported into a spreadsheet application, and graphs are generated.

Storing an experiment's events in a database facilitates data analysis and provides an easy-to-use interface to explore the data. The results presented in this paper are just a sample of the information that can be obtained using this toolkit. The potential of our performance evaluation framework is exemplified in [10] and [11].

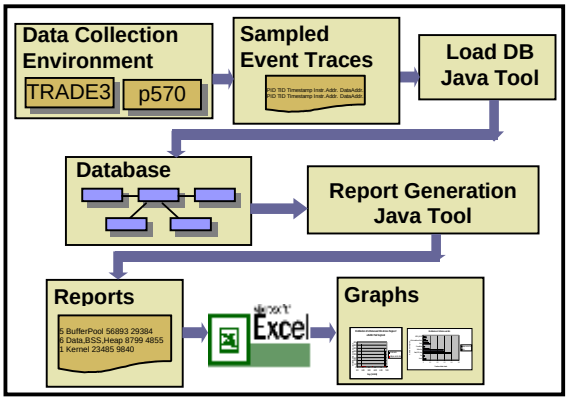

Figure 2. Performance Evaluation Framework

4. Results

The analyses and results presented in this section compare two different sets of sampled event traces, which were collected using the two different machine configurations described in Section 3.1. The first set, called 1P, is the set of event records captured during the execution of TRADE3 on one dedicated p570 partition. This data set represents our baseline data in that TRADE3 is executing exclusively on the 4-processor p570, i.e., hardware is not being shared with any other application or execution environment. In contrast, the second set, 5P, captures the execution of TRADE3 in a micro-partitioned environment; it consists of traces captured while monitoring one of the active micro-partitions. In this case, there are five active micro-partitions executing concurrently on the p570. Each is assigned two virtual processors (VPs) and is executing its own copy of the TRADE3 application.

The overall goal of our analysis is to observe differences between the 1P and 5P data sets in terms of the TRADE3 data-load behavior across levels of the p570 memory hierarchy. Such differences could represent the performance overhead incurred when executing an application in a virtualized environment.

Because our data sets are sampled event traces, we do not capture every occurrence of a monitored event. Previous work, [10] and [11], demonstrates the capability of sampled event traces to capture representative application behavior; however, the event sample counts in that work are substantially larger than those associated with this work. Therefore, to ensure the representativeness of our data sets, we compare the aggregate event counts obtained with performance counters to those obtained in our sampled event traces.

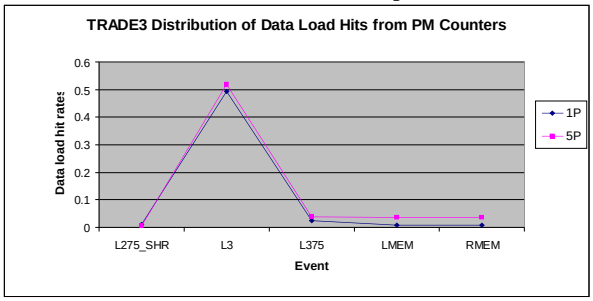

Figure 3. Aggregate Event Counts from Performance Counters

Figure 3 presents the aggregate event counts obtained from performance counters for both the 1P and 5P executions of TRADE3. As shown, for both the 1P and 5P cases L2-cache data-load misses are resolved predominantly in the L3 cache associated with the miss-generating DCM, i.e., approximately 50% of data-load hits are associated with the L3 event. Additionally, the 5P data set, as compared to the 1P data set, shows a notable increase in the number of L2-cache data-load misses resolved in local and remote memories, i.e., memories associated with the miss-generating DCM (LMEM event) and memory associated with a different DCM (RMEM event).

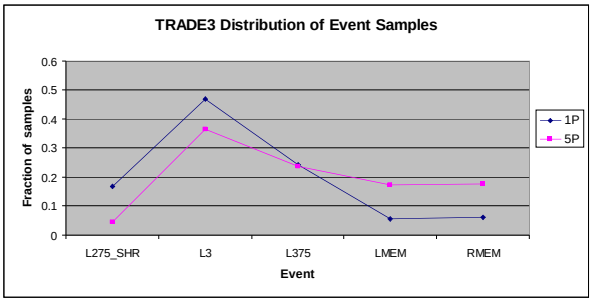

Figure 4. Event Sample Counts from Sampled Event Traces

Figure 4 presents the event sample counts associated with the 1P and 5P sampled event traces. Comparing Figures 3 and 4, the same trends can be seen, i.e., Figure 4 shows that the fraction of samples associated with each event is similar to the related performance event counts; again, the majority of data-load hits are associated with the L3 event and for the 5P cases, as compared to the 1P case, there is a noticeable increase in the fraction of data-load hits associated with the LMEM and RMEM events. This comparison indicates that the sampled event traces do capture representative

application data-load behavior. Recall that the 5P data set is comprised of fewer samples than the 1P data set (see Table 2) due to the fact that the 5P data set is comprised of events captured on two VPs running on one physical processor, while the 1P data set is comprised of events captured on four physical processors. This could explain the larger deviation of the 5P sampled event counts from the aggregate event counts obtained from performance counters.

The sampled event traces capture data-load behavior for seven different events in two separate data sets. In order to attempt to identify the performance overhead associated with virtualization, the first target of the analysis is the identification of events with behavior that significantly differs between the 1P and 5P cases. As previously mentioned, the LMEM and RMEM events experience a noticeable increase in the 5P case and given their associated latencies (281 cycles for LMEM and 307 cycles for RMEM), this translates into an average 331% increase in latency per instruction. Furthermore, the DB2 and WebSphere components account for over 90% of the execution time captured in the sampled event traces. Therefore, the remainder of our analysis will focus on L2-cache data-load misses associated with DB2 and WebSphere that are resolved in both LMEM and RMEM. Note that, in the interest of space, the LMEM and RMEM events are combined to form a single MEM event.

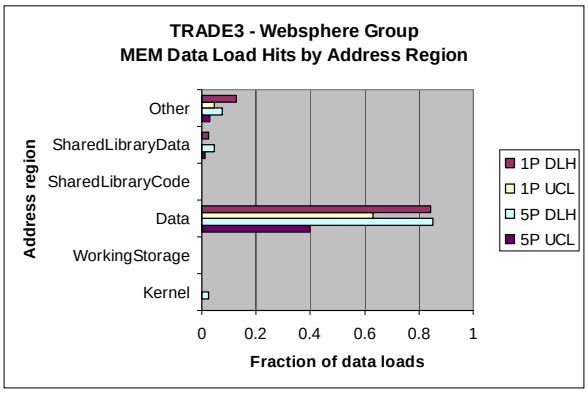

Figure 5. Data-Load Hits in Memory by Address Region for Websphere

Figure 5 profiles the distribution of the WebSphere component's data-load hits in MEM across regions of the application address space. For both the 1P and 5P data sets, each address region contains an associated data-load hits bar (DLH) and unique cache line (UCL) bar. The DLH bar corresponds to the fraction of data-load hits associated with the address region, while the UCL is an indication of the unique data that is accessed, i.e., locality of reference for the associated data-load hits. As shown, for both the 1P and 5P cases, over 80% of the L2-cache data-load misses resolved in MEM are associated with the Data region. However, there is a radical difference in the locality of reference for these data-load hits. In the 1P case, the UCL is a large percentage of the DLH indicating that a majority of data-load hits access unique data. This demonstrates good application data-load behavior; i.e., in general, data is accessed from memory only once, rather than repeatedly. In contrast, for the 5P data, the UCL is a smaller percentage of the DLH, indicating an increased locality of reference for the data-load hits associated with the Data region. This indicates that data is being prematurely evicted from the higher levels of the memory hierarchy, causing repeated data references to memory. Capacity and/or conflict misses due to contention for cache space could be the cause of this increased locality of reference, which is an indication of the memory performance overhead associated with virtualization.

Using the performance evaluation framework, we further profiled data-load hits in MEM by the individual processes that triggered each event. In the interest of space, the complete results of this study are not presented here. However, note that the increase in data-load hits at MEM for the 5P case is not attributable to any particular process group (i.e., kernel, DB2, WebSphere, or other processes); instead, there is a uniform increase across process groups.

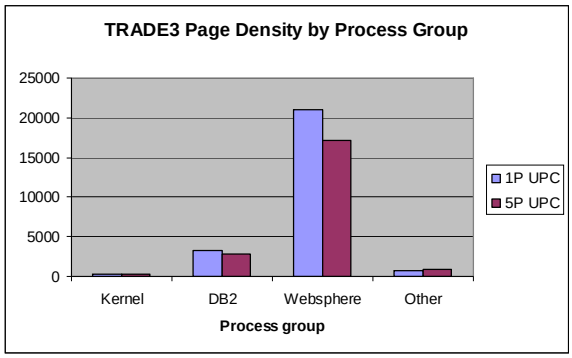

Figure 6. Overall Page Density

In an effort to understand the cause of the performance overhead described above, the page density associated with each data set is analyzed. Figure 6 represents the unique page count (UPC) per process group for both the 1P and 5P data sets. The UPC indicates the number of unique pages touched throughout the monitoring interval; it gives an indication of the memory footprint per process group. Notice that the 5P data set accesses a smaller number of unique pages than the 1P case. However, as described above, the 1P data exhibits better performance in terms of L2-cache data-load hits resolved in MEM. A possible reason for this discrepancy could be that the 5P data set represents the

data-load activity for only one of five active partitions, each of which accesses a similar number of unique pages. In this case, the hardware is actually managing a much higher UPC than is recorded in the 5P data set.

Additionally, note that the UPC associated with Kernel processes is the same for both the 1P and 5P cases. Therefore, regardless of the execution environment, the memory footprint for the operating system remains unchanged. This helps focus optimization efforts in that the performance overhead is associated with the components of the application rather than the operating system itself.

5. Conclusions

The goal of the work presented in this paper is to demonstrate the potential of the performance evaluation methodology to identify and understand the performance overhead associated with virtualization. Sampled event traces were collected for TRADE3 executed in non-virtualized (1P) and virtualized (5P) environments on an IBM eServer pSeries 570. By comparing event sample counts associated with the traces to the event counts captured by performance counters, the sampled event traces are shown to be representative of actual application data-load behavior. Comparing the 1P and 5P data sets, performance overhead associated with virtualization is identified in terms of the distribution of resolution sites for L2-cache data-load misses across levels of the memory hierarchy. Beginning with a profile, by address space region, of data-load hits in MEM for one of TRADE3's three components, WebSphere, it is shown that the 5P case has an increased locality of reference in MEM. This indicates a possible increase in capacity and/or conflict misses for the 5P case, supporting an expected performance overhead associated with virtualization arising from contention for hardware resources, such as caches. Additionally, the page density for each data set is studied in an effort to identify the source of the performance overhead. This analysis shows that the components of the application are the source of the performance overhead, rather than the operating system. This helps target optimization efforts by eliminating the operating system as a significant contributor to performance degradation.

6. Future Work

One of the major goals of future work is to concurrently trace the data-load behavior of all active partitions. Recall that the 5P data set is the sampled event trace of only one of the five active micro-partitions. Having sampled event traces for all active micro-partitions will facilitate performance analysis of application data-load behavior across individual partitions. Additionally, the performance evaluation framework will be improved to assist in the study of data-load behavior in terms of hypervisor dispatching. When a partition is activated by the hypervisor, a spike in the number of cache misses is expected as the working set is brought into the memory subsystem. Ideally, the number of cache misses is expected to stabilize once the working set has been loaded. Deviations to this expected behavior could indicate additional performance issues. The study of application data-load behavior also will be expanded to different applications and execution environments.

Acknowledgements

This work was supported by the IBM Corporation through IBM Faculty awards, an NPSC/IBM Ph.D. Fellowship, and an IBM SUR grant. It also was supported by The University of Texas – El Paso.

REFERENCES

[1] J. Anderson, et al., "Continuous Profiling: Where have all the cycles gone?," ACM Transactions on Computer Systems, 15:4, November 1997, pp. 357-390.

[2] L. Barroso, et al., "Memory System Characterization of Commercial Workloads," Proceedings of the 25th ISCA, Spain, June 1998, pp. 3-14.

[3] R. Desikan, et al., "Measuring Experimental Error in Microprocessor Simulation", Proceedings of the 28th ISCA, Goteborg, Sweden, July 2001, pp. 266-277.

[4] R. Figueiredo, et al., "Guest Editors' Introduction: Resource Virtualization Renaissance," *IEEE Computer*, 38:5, May 2005, pp. 28 – 31.

[5] B. Gibbs, et. al., "Advanced POWER Virtualization on IBM eserver p5 Servers: Architecture and Performance Considerations", IBM Redbook, March 2005.

[6] J. Hennessy and D. Patterson, Computer Architecture: A Quantitative Approach, Morgan Kaufman, CA, 1996.

[7] R. Kalla, et al., "IBM Power5 Chip: A Dual-Core Multithreaded Processor," IEEE Micro, vol. 24, no. 2, March/April 2004, pp. 40-47.

[8] K. Keeton, et al., "Performance Characterization of a Quad Pentium Pro SMP Using OLTP Workloads," Proceedings of the 25th ISCA, June 1998, pp. 15-26.

[9] C. Lu and D.A. Reed, "Compact Application Signatures for Parallel and Distributed Scientific Codes," Proceedings of the 2002 ACM/IEEE Conference on Supercomputing, Baltimore, MD, November 2002, pp. 1-10.

[10] R. Portillo, et al., "Mining Performance Data from Sampled Event Traces," Proceedings of the 12th MASCOTS, Volendam, The Netherlands, October 2004.

[11] D. Villa, et al., "A Framework for Profiling Multiprocessor Memory Performance," Proceedings of the 10th ICPADSs, Newport Beach, CA, July, 2004, pp. 530-538.

[12] IBM Corp. (July 2004), IBM eServer p5 - AIX 5L Support for Micro-Partitioning and Simultaneous Multi-threading, http://www-1.ibm.com/servers/aix/whitepapers/aix_support.pdf