2011 IEEE International Conference on High Performance Computing and Communications

Performance Modeling of a Consolidated Java Application Server

Hitoshi Oi and Kazuaki Takahashi
Department of Computer Engineering
The University of Aizu
Aizu-Wakamatsu, JAPAN
Email: {oi,dianthudia}@oslab.biz

***Abstract*—System-level virtualization enables multiple servers to be consolidated on a single hardware platform and to share its resources more efficiently. We are currently developing a performance model of a consolidated multi-tier Java application server. The model breaks down the CPU utilization of the workload into servers and transaction types, and use these service time parameters in the network of queues to predict the performance. For the target of initial development, we use SPECjAppServer2004 running on a quad-core server consolidated by Xen.**

In this paper, we present the current status of performance model development. We have found that the measured CPU utilization seems lower than the actual system saturation level. As a result, the performance model is saturated at a larger system size. We also have found that while the behavior of Manage transactions is most sensitive to the system size, its service times are lower than other transactions. When the CPU utilization of 4-core execution is predicted by the data from 1 to 3-core executions, the prediction errors range from -3.6 to 43.4%, with the largest error occurring in the database domain.

Keywords: Workload Analysis, Virtualization, Server Consolation, Performance Modeling, Measurement, SPECjAppServer2004, SPECjEnterprise2010

I. Introduction

By means of system virtualization, multiple independent systems can be consolidated on a single physical platform. Advantages of consolidation include increased hardware resource utilizations, reduced cost of servers and peripheral devices, and reduced power supply and cooling costs. One possible target of consolidation is a multi-tier system, which consists of multiple server applications interacting with each other. Such a consolidated system still can employ, for example, the administrative policies and software versions of the original (unconsolidated) system. On the other hand, there are also disadvantages in the server consolidation. First, virtualization itself consumes system resources [1]. Second, each virtualized system (or virtual machine, VM) has its own resource demand which changes dynamically during operation and sometimes VMs compete for resources with each other. This can make the system administration and tuning difficult.

To understand the behavior of consolidated multi-tier systems, we are currently developing a queuing network based performance model of such systems that takes into account the virtualization overhead and the effect of multi-core processors. For the target of initial development, we use SPECjAppServer2004 [2] running on a quad-core server consolidated by Xen. The CPU utilization of each virtual machine (or domain in Xen) running a layer of a multi-tier system is broken down into transaction types, and they are used as the service time parameters in the queuing network. In this paper, we present the current status of performance model development and the issues we have found thus far, including the discrepancies between the measured CPU utilization and the system behavior. We also investigate the feasibility of approximating multi-core performance from the measurement results.

The rest of this paper is organized as follows. In the next section, a brief introduction to SPECjAppServer2004 and methodologies of the performance model construction are presented. In Section III, the measurement results are analyzed and compared against the output of the model. In Section IV, we introduce related work and the paper is concluded in Section V.

Disclosure

SPECjAppServer and SPECjEnterprise are trademarks of the Standard Performance Evaluation Corp. (SPEC). The SPECjAppServer2004 and SPECjEnterprise2010 results or findings in this publication have not been reviewed or accepted by SPEC, therefore no comparison nor performance inference can be made against any published SPEC result. The official web sites for SPECjAppServer2004 and SPECjEnterprise2010 are located at [2] and [3], respectively.

II. Performance Modeling

In this section, we first describe the target system of initial model construction, SPECjAppServer2004, and then present the methodologies of the performance model construction.

A. SPECjAppServer2004

SPECjAppServer2004 is a benchmark suite to evaluate the performance of Java Enterprise Edition (JavaEE[1]) which is modeled after the business of the automobile manufacturer [4].

[1]formerly called Java 2 Platform Enterprise Edition, or J2EE when SPECjAppServer2004 was published.

978-0-7695-4538-7/11 $26.00 © 2011 IEEE
DOI 10.1109/HPCC.2011.118

There are five application domains: Dealer (selling automobiles), Customer (managing orders from Dealers), Supplier (handling parts orders to external vendors), Manufacturing (managing vehicle manufacturing) and Corporate (managing Dealers and their orders).

The target system of SPECjAppServer2004 consists of three layers of servers: web, application and database. The web server works as an interface to the transactions from the Dealer domain. The application server acts as a middleware handling transaction management. The database server (DB) holds information on orders, customers (Dealers) and their inventories.

There are five transactions processed at the target system. Browse (browsing automobile catalogs), Manage (managing orders and inventories) and Purchase (ordering automobiles): These three types of transactions are issued from Dealers via the Web server. The transactions from the Manufacturing domain are called WorkOrders which model the activities of the vehicle production lines. Based on the quantity of the cars, they are classified into two types, Planned line and LargeOrder. The former is for orders with smaller quantities (14 cars on average) and issued periodically, while the latter is for larger orders (140 cars on average) and issued immediately as the results of Purchase transactions from the Dealer domain.

The system sizes, such as number of emulated clients, are proportional to the scaling factor (SF)[2] and so are the transaction issue rates from the Dealer and Manufacturing domains (TABLE I).

TABLE I
TRANSACTION RATE IN TERMS OF SCALING FACTOR (SF). TRANSACTION TYPES ARE P (PURCHASE), M (MANAGE), B (BROWSE) W (WORKORDER) AND L (LARGEORDER). THE WORKORDER RATE IS INCLUSIVE OF LARGEORDER.

Transaction	P	M	B	W	L
Rate (*SF/Sec)	.25	.25	.5	.67	.067

B. Performance Modeling Methodologies

To model the performance of a consolidated server running SPECjAppServer2004, we first breakdown the workload into the service time for each transaction type and domain (virtual machine), $ST_{i,j}$, where i is a transaction type (Purchase, Manage, Browse, WorkOrder and LargeOrder, or $TX = \{p, m, b, w, l\}$) and j is a virtual machine (Web, App DB and Dom0, or $VM = \{W, A, D, 0\}$). This methodology is used in [5] with the following differences. On the system we used for the measurements, the disk utilization was low (1.3% at $SF = 30$ at which the highest throughput with valid response times was achieved) and we did not include the queuing delay of the disk access. Therefore, we only take into account the CPU utilization as the system resource.

Next, since our target platform is a consolidated server on which a privileged domain (Dom0) works as the interface between Xen and guest domains, it consumes CPU time and we include it to represent the (part of) virtualization overhead. Transaction flows and mapping of transaction processing to the physical CPU are shown in Fig. 1. Similar to [5], transaction flows are simplified so that each transaction visits each server domain only once.

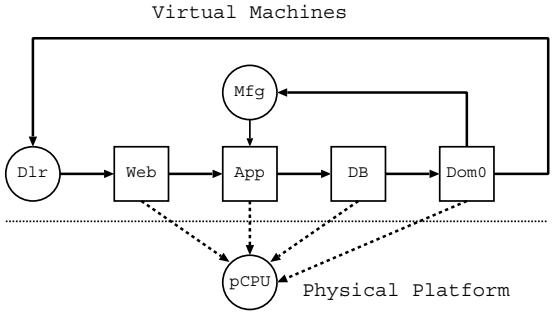

Fig. 1. Transaction Flow and Mapping to Physical CPU. The squares above the horizontal line are the servers running on the virtual machines (Web, App, DB and Dom0). The circles above the line are the transaction drivers (Dlr and Mfg). The solid arrows are the flows of transactions and the dashed lines are mapping of transaction processing to the physical CPU (pCPU).

From the scaling factor (SF), we calculate the issue rate of each transaction (TP_i where $i \in TX$), and using the transaction service times $ST_{i,j}$ from the measurement, we can obtain the CPU utilization in each virtual machine (U_j, where $j \in VM$):

$$U_j = \sum_{i}^{TX} TP_i \times ST_{i,j} \quad (1)$$

When the total CPU utilization $Util = \sum_{j \in VM} U_j \leq 1$, the system is not saturated and we calculate the response time of each transaction type (RT_i, where $i \in TX$) using the processor-sharing model [6] as follows:

$$RT_i = \sum_{j}^{VM} ST_{i,j}/(1 - Util) + C_i \quad (2)$$

C_i is the constant factor in the response time for each transaction type i which represents the latency of the components not included in out model (network interface, HDDs). In the experiments in Section III-B, we use the response time at a small scaling factor (RT_i at $SF = 5$) minus $\sum_j ST_{i,j}$, where $j \in VM$ for C_i.

III. EVALUATION

In this section, we first present the specifications of the measurement platform. Next, we examine the prediction results obtained from the performance model and also investigate the feasibility of approximating multi-core performance with the model.

[2]We use the term Scaling Factor, or SF, to specify the system size instead of the official term Injection Rate in [4].

A. Measurement Environment

TABLE II shows the specifications of the hardware and software used for measurements (top) and the configurations of the virtual machines and applications (bottom). To obtain the per domain per transaction service time ($ST_{i,j}$ in Section II-B), we change the mixture of the transactions issued from the driver. Also, for the measurement of WorkOrder transactions, we use the customized driver by Kounev [5]. To obtain the CPU utilization of each domain, we use xentop.

TABLE II
MEASUREMENT PLATFORM SPECIFICATIONS (TOP) AND VIRTUAL MACHINE CONFIGURATIONS (BOTTOM).

Component	Specification
CPU	Xeon (E5310, Quad-Core, 1.86GHz)
Memory	6GB
OS	CentOS 5.3
VMM	Xen 3.1.2

VM	Config		Application
	vCPU	Mem	
Web	2	0.5GB	apache v2.2.14
App	4	3GB	JBoss v4.0.0
DB	2	1.5GB	PostgreSQL v8.3.5
Dom0	2	0.5GB	

On this platform, the highest throughput with the valid response times (≤ 2 and $5 \leq$ seconds for the 90% of Dealer and Manufacturing transactions, respectively) is obtained at $SF = 30$. TABLE III shows the utilization of CPU time in each domain at $SF = 30$. The largest fraction of CPU time is allocated to the application domain and DB and web domains follow. Dom0 takes about 10% of the CPU time.

TABLE III
CPU UTILIZATION AT $SF = 30$

VM	Web	App	DB	Dom0
Util (%)	5.1	50.1	21.6	10.2

B. Scalability and CPU Utilization

From the measurement results described in the previous section, we obtained the service time of each transaction at each domain (TABLE IV). The response time behavior predicted by the performance model with these service time parameters shows several discrepancies against the measurement results (Fig. 2).

TABLE IV
TRANSACTION SERVICE TIME (MSEC)

Transaction	Web	App	DB	Dom0
Purchase	1.42	13.57	7.55	2.91
Manage	0.23	1.39	0.81	0.54
Browse	1.91	7.87	0.61	2.39
WorkOrder	0.05	9.18	5.96	1.63
LargeOrder	4.42	50.21	17.58	4.96

First, it seems that the actual system is saturated at a smaller system size ($SF > 32$) than that of the performance model ($SF > 35$). In other words, the sum of CPU utilizations in guest domains reported by xentop is smaller than the actual CPU utilization by some factor and it should be this factor that represents the overhead of virtualization.

Fig. 2 compares the measured and predicted average response times of Purchase, Manage (Dealer domain) and WorkOrder (Manufacturing domain) transactions. For the system sizes below saturation, the actual response times increase faster than those of the performance model. For example, for $SF = 5 \rightarrow 25$, while the response time of WorkOrder transaction is increased by 58% on the actual system, that of the performance model is increased only by 14%. Another point of discrepancy is the type of Dealer transaction that is most sensitive to the system scaling up. On the actual system, the response time of the Manage transaction violates the quality-of-service (QoS) requirement ($\leq$ 2seconds at 90 percentile) before other two transactions. However, it is the Purchase transaction on the performance model (Fig. 2). Actually, we can also find this discrepancy in TABLE IV: the service times of Manage transaction are lower than those of Purchase transaction in all domains, which result in the slower response time growth on the queuing network-based performance model.

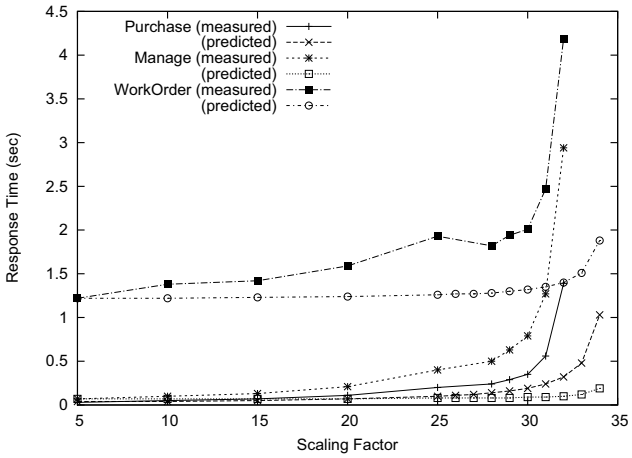

Fig. 2. Response Time Comparison

C. Prediction of the Multi-Core Performance

As the industry trend in high-performance processors has switched from higher clock frequency or higher degree of superscalar to increasing the number of cores [7], one of the features we wanted to incorporate into the performance model is to predict the performance gain when the number of cores is increased. A possible and straightforward approach is to approximate the service time for n-core by the formula $a/n+b$ in Amdahl's law [8]. Based on the workload measurements of SPECjAppServer2004 for $n = 1 \ldots 4$, we will discuss the feasibility of this approximation.

TABLE V shows the maximum scaling factor and CPU utilization in each domain when the number of cores is

TABLE V
CPU UTILIZATION (%) AND MAXIMUM SCALING FACTOR (SF) FOR 1 TO 4 CORES

	No. Cores			
	1	2	3	4
SF	9	15	22	30
Web	7.5	5.9	5.7	5.1
App	39.1	43.2	46.6	50.1
DB	21.4	24.0	24.3	21.6
Dom0	15.2	11.6	10.6	10.2
Total	83.3	84.7	87.2	87.1

increased from 1 to 4. The scaling factor (SF) is maximum in the sense at which the highest throughput with valid response time[3] is achieved. As mentioned in Section III-B, the sum of CPU utilization of each domain reported by xentop is significantly lower than 100%. Please note that the SFs in TABLE V are before saturation. By ignoring the response time conditions, the total CPU utilizations are slightly increased, but they are still below 100%. For example, in the case of 4-core, the total CPU utilization can be 91.2% at $SF = 32$.

TABLE VI
CPU UTILIZATION NORMALIZED TO SCALING FACTOR (%).

	No. Cores			
VM	1	2	3	4
Web	0.83	0.40	0.26	0.17
App	4.35	2.88	2.12	1.67
DB	2.38	1.60	1.11	0.72
Dom0	1.69	0.77	0.48	0.34

Next, we normalize the CPU utilization in each domain by SF (TABLE VI). Please note that for each SF in TABLE V, the system is not saturated and the ratios between transaction types are constant (TABLE I). Therefore, TABLE VI should indicate the CPU utilizations for the same amount of work for the varying number of cores. We fit the CPU utilizations of 1 to 3-core executions into $a/n+b$, where n is the number of cores by the least square method and predict the CPU utilizations in 4-core executions. The prediction results and their errors are shown in TABLE VII. The prediction errors range from -3.6 to 43.4%. These large errors, especially in the DB, indicate the necessity of further investigation and modeling methodology refinement.

TABLE VII
4-CORE CPU UTILIZATIONS PREDICTED BY 1 TO 3-CORE EXECUTIONS. TOTAL IS THE SUM OF ALL FOUR DOMAINS MULTIPLIED BY THE MAXIMUM SCALING FACTOR (30).

VM	Web	App	DB	Dom0	Total
Util (%)	0.18	1.94	1.03	0.33	104.53
Error (%)	6.9	16.1	43.4	-3.6	20.0

[3]Manage and Browse transactions violate the response time requirements first beyond these SFs, in 2 to 4-core and 1-core executions, respectively.

D. SPECjEnterprise2010

SPECjEnterprise2010 is the latest benchmark suite from SPEC to evaluate the application servers based on Java EE [3]. On November 30, 2010, SPECjEnterprise2010 replaced SPECjAppServer2004 and our effort of performance model development continues with SPECjEnterprise2010 as the target system. In addition to the version up of the base JAVA EE technology (from J2EE 1.3 to Java EE 1.5), SPECjEnterprise2010 is designed to use JMS and MDB more extensively. We are currently migrating the target system to SPECjEnterprise2010 and we present the result of initial measurement in this Section. TABLE VIII shows the specifications of the measurement platform (top) and virtual machine configurations (bottom).

TABLE VIII
SPECJENTERPRISE2010 MEASUREMENT PLATFORM SPECIFICATIONS (TOP) AND VIRTUAL MACHINE CONFIGURATIONS (BOTTOM).

Component	Specification
CPU	Xeon 2.13GHz (X3210)
Memory	6GB
OS	CentOS 5.5
VMM	Xen 4.0.1

VM	Config		Application
	vCPU	Mem	
App	4	2GB	Glassfish v3.0.1
DB	2	1.5GB	MySQL v5.1.50
Dom0	2	0.5GB	

Fig. 3 shows the CPU utilization of SPECjAppServer2004 and SPECjEnterprise2010 at $SF = 20$. Please note that the system setup and parameter turning of SPECjEnterprise2010 are still underway. Also, in SPECjEnterprise2010, the web server is not in a separate VM; rather Dealer transactions are routed to the application server (Glassfish) through its http port. For these reasons, it is not possible to draw definitive conclusions from this early stage of measurement. However, we see that the CPU utilization of the application server is significantly higher in 2010, while that of the DB server is decreased. This observation coincides with the design of SPECjEnterprise2010, in which some functionality has been moved from the DB server to the application server (A52 in [9]).

IV. RELATED WORK

Kounev measured the workload of SPECjAppServer2004 and broke down the CPU utilization of each server to transaction types. Using these workload parameters, he build a performance model by the Queuing Petri Net (QPN) and validated it on the machines with varying performance [5]. Our modeling methodology is based on Kounev's and we aim at extending and validating the model under the conditions such as the varying number of CPU cores and different transactions mix.

In [10], authors defined the parameters to represent the resource usage of VMs (e. g. CPU or memory), and refined them using the techniques of artificial neural network training. In [11], authors formulated the performance interferences

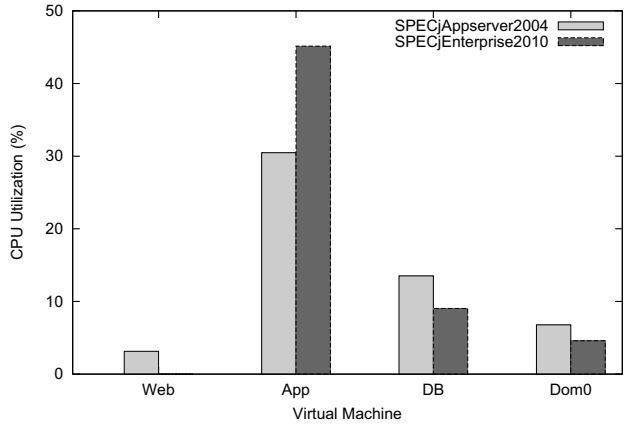

Fig. 3. SPECjAppServer2004 and SPECjEnterprise2010 CPU Utilization ($SF = 20$).

between VMs using the workload parameters, such as CPU utilization or the number of cache hits. The target platform had two VMs running relatively simple synthetic benchmark and utility programs. In [12], authors modeled the interferences between VMs on a consolidated CMP server using the similar parameters as [11]. One of differences between this paper and [11], [12] is, in this paper, VMs execute different layers of a single system, while in [11], [12], VMs execute different applications independently.

SPEC and several vendors proposed benchmark suites for the performance of virtualized data center servers [13], [14], [15]. They assume that the virtualized servers are configured as 'tiles' : Each title consists of a set of VMs and each VM on a title runs a common commercial application such as web server, mail server. One of advantages of virtualized systems is the capability of changing resource allocations dynamically. AutoParam is one of attempts to control the resource allocation in a consolidated 3-tier application server by feeding back the application QoS [16]. Virtualization technologies are mostly used for desktop and server systems. Ye et. al, investigated the overhead of virtualization in high-performance applications [17].

V. Conclusion and Future Work

In this paper, we presented our work-in-progress effort of developing a performance model of consolidated Java application servers. Using SPECjAppServer2004 as the initial target system, we extended the methodologies in [5] to a consolidated system. We evaluated its feasibility and also analyzed the workload parameters obtained from the measurements. Our observations are as follows. (1) It seemed that the CPU utilizations of VMs obtained from xentop were lower than those of the actual system. We expect that this difference represents the virtualization overhead. (2) The transaction response times increased gradually even before CPU saturation. This indicates that it is necessary to incorporate other factors than PS-queue modeling of CPUs into the transaction response times. (3) The observed response time and service times of Manage transaction showed discrepancy. This is the results of measured per-domain service time of Manage transaction. (4) When the number of CPU cores was varied, the above mentioned potential virtualization overhead ranged 13 to 17%. (5) The prediction error of 4-core CPU utilization using 1 to 3-core executions ranged from -3.6 to 43.4%. The largest error occurred in the DB domain.

Our current and future work include: (i) As mentioned in Section III-D, migrating the target system and measurement platform to SPECjEnterprise2010, (ii) measurement and analysis of per-transaction service time in 1 to 3-core executions to identify the source of the prediction error, (iii) analysis of workload in each VM and vCPU scheduling to identify the virtualization overhead more precisely.

Acknowledgments

The authors would like to thank Samuel Kounev of Karlsruhe Institute of Technology for his help in measurements of SPECjAppServer2004. We would also like to thank the SPEC support team for their technical support in using SPECjAppServer2004 and SPECjEnterprise2010.

References

[1] Aravind Menon, Alan Cox and Willy Zwaenepoel, "Optimizing Network Virtualization in Xen ," in *Proceedings of the 2006 USENIX Annual Technical Conference*, pp15–28, May–June 2006.
[2] SPECjAppServer2004, http://www.spec.org/jAppServer2004/
[3] SPECjEnterprise2010, http://www.spec.org/jEnterprise2010/
[4] SPECjAppServer2004 Design Document, http://www.spec.org/jAppServer2004/docs/DesignDocument.html
[5] Samuel Kounev, "Performance Modeling and Evaluation of Distributed Component-Based Systems Using Queueing Petri Nets," IEEE Trans. on Software Engineering, vol. 32 no. 7, pp. 486–502, 2006.
[6] Leonard Kleinrock, "Time-shared Systems: a theoretical treatment," Journal of the ACM (JACM), Vol. 14 Issue 2, pp242–261, April 1967
[7] David Geer, "Industry Trends: Chip Makers Turn to Multicore Processors," Computer, vol. 35, no. 5, pp. 11–13, May 2005.
[8] Gene M. Amdahl, "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities," in *Proceeding of the AFIPS Spring Joint Computer Conference*, April pp18–20, 1967.
[9] SPECjEnterprise2010 Frequently Asked Questions, http://www.spec.org/jEnterprise2010/docs/FAQ.html .
[10] Sajib Kundu, et. al., "Application Performance Modeling in a Virtualized Environment," in *Proceedings of the Sixteenth International Symposium on High-Performance Computer Architecture*, January 2010.
[11] Younggyun Koh, et. al., "An Analysis of Performance Interference Effects in Virtual Environments," in *Proceedings of 2007 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS 2007)*, pp200–207, April 2007.
[12] Padma Apparao, et al., "Towards Modeling & Analysis of Consolidated CMP Servers," in *ACM SIGARCH Computer Architecture News*, Vol. 36, Issue 2, May 2008.
[13] SPECvirt_sc2010, http://www.spec.org/virt_sc2010/
[14] "VMmark Virtualization Benchmarks," http://www.vmware.com/products/vmmark/
[15] Jeffrey P. Casazza, Michael Greenfield and Kan Shi, "Redefining Server Performance Characterization for Virtualization Benchmarking," Intel Technology Journal, Vol. 10, Issue 03, pp. 243–252, 2006.
[16] Zhikui Wang, et. al., "AutoParam: Automated Control of Application-Level Performance in Virtualized Server Environments," in *Feedback Control Implementation and Design in Computing Systems and Networks*, Munich, Germany, May 2007
[17] Kejiang Ye, et. al, "Analyzing and Modeling the Performance in Xen-Based Virtual Cluster Environment," in *Proceedings of 2010 IEEE 12th International Conference on High Performance Computing and Communications*, pp273–280, September 2010.