Vol.2011-EVA-36 No.1
2011/12/2

Performance Analysis of SPECjEnterprise2010

Sho Niboshi and Hitoshi Oi†1

We are currently developing a performance analysis model for SPECjEnterprise2010, a benchmark suite for a multi-tier Java EE server which is modeled after an automobile manufacturer. In this report, we present our recent measurement results and their analysis, including the scalability, the workload on each server and the response time of each transaction type.

1. Introduction

We have been developing a performance model for consolidated multi-tier Java application server[1]. As the initial target of the model, we have been using SPECjAppServer2004 (noted as 2004 here after)[2]. However, according to the retirement of 2004 on November 2010, we also have migrated our target system to SPECjEnterprise2010 (2010)[3]. While both benchmark suites are modeled after the business of the automobile manufacturer, 2010 has up-to-date version of the Java EE technology (from J2EE 1.3 to Java EE 1.5) and it is designed to use JMS and MDB more extensively.

Our modeling methodology is based on Kounev's[4] with extensions in (1) basing the platform on a consolidated system that includes virtualization overhead and (2) prediction of the performance gain with the increased number of cores. Using the workload parameters obtained by the measurement of 2004, we compared the actual measurements and the predictions by the model and identified discrepancies in (i) total (physical) CPU utilization and the transaction type that is most sensitive to the system scaling[1].

Disclosure

SPECjEnterprise is trademark of the Standard Performance Evaluation Corp. (SPEC). The 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.

†1 The University of Aizu. Authors are ordered alphabetically.

2. Tuning Problem in Consolidated Platform

After the initial measurements of 2010 on a Xen-consolidated system using a 4-core Xeon 2310 (reported in [5]), we migrated our measurement platform to a 6-core AMD Phenom II based system (now used as the application server in Table 1) and tackled the problems identified in[1]. On this measurement platform, we created three domains (VMs): Application Server (6 vCPU, 7GB Memory), DB Server (6 vCPU, 6 GB Memory) and Dom0 (6 vCPU and 0.5GB Memory). While conducting scalability measurement of this platform, we have encountered a problem in tuning the system and application parameters. The system saturated at around 70% of the total CPU utilization measured by `xentop`. From our experiences in[1], CPU utilization reported by `xentop` seems to be lower than the actual system status. However, a system saturated at 70% of CPU utilization looks suboptimal and requires further tuning. To properly tune the system and application parameters (and to understand the behavior of 2010), we have decided to conduct the tuning and measurements on non-consolidated (i.e. separate) servers first, and then return to the consolidated platform later.

3. Measurement

In this section, we present the measurement results obtained thus far. As mentioned in the previous section, we use two machines for application and database servers (Table 1). System tuning that are have been applied to the system include installation of `irqbalance` and use of `-XX:+DisableExplicitGC` option for the JVM. To represent the system size, we use the term Scaling Factor (SF) instead of the official term Injection Rate (IR).

Fig. 1 shows the CPU utilization of the App server (left) and the DB server (right). The vertical lines drawn at $SF = 177$ indicate the maximum system size by which all the transaction response times meet the requirement. First of all, it is noticed that the CPU utilization of the App server does not linearly grow against SF. The growth rate of the CPU utilization is reduced gradually. This is not a good sign for our modeling methodology, because it is based on the

Component	App Server	DB Server
CPU	AMD Phenom II X6 1065T	Xeon X3210 (4 Cores)
Memory	16GB	6GB
OS	Oracle Linux 6	
Software	Glassfish v3.0.1	MySQL 5.5.13

Table 1 Measurement Environment

assumption that each transaction type consumes a fixed amount of CPU time at each server[⋆1]. The source of this non-linearity needs to be identified by further investigations. On the contrary, the CPU utilization of the DB server grows linearly against SF. In addition, at the maximum SF of 177, the CPU utilization is close to 100% (93%) and it saturates for $SF > 177$. This implies that the DB server is the performance bottleneck of our measurement platform.

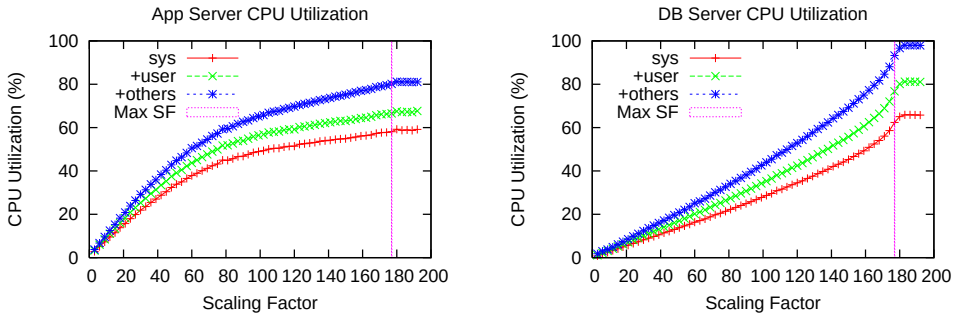

Fig. 1 CPU Utilization (Left: App Sever, Right: DB server)

Fig. 2 shows the average transaction response time (left) and the throughput against SF. For the system smaller than saturation point (at $SF = 177$), the response time of each transaction is quit low (less than 1 second) and the transaction throughput grows linearly. It is also noticed that the response times of Purchase and Browse transaction types are most sensitive to the system scaling.

⋆1 Kounev's model also assumes this property

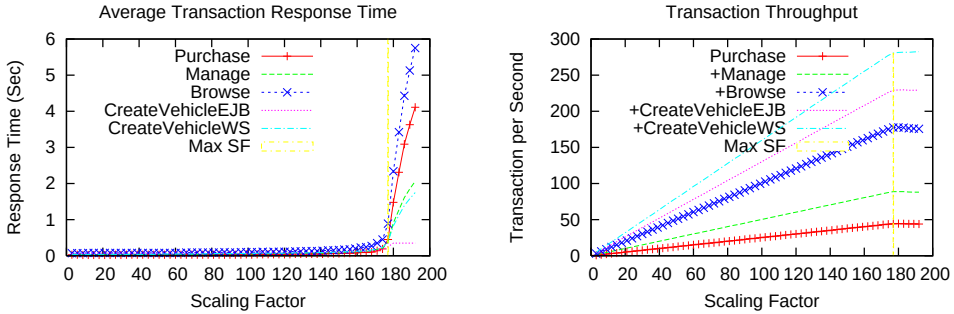

Fig. 2 Transaction Response Time (Left) and Throughput (Right)

4. Current Status

We are currently performing the measurement of each transaction type separately. As mentioned in Section 3, it is necessary to identify the reason of non-linearly in the CPU utilization of the App server against the increasing system size. According to the result of this investigation, we may have to alter the modeling methodology so that the model represents the system behavior of the multi-tier Java application server more accurately.

References

1) Hitoshi Oi and Kazuaki Takahashi, "Performance Modeling of a Consolidated Java Application Server," in *Proceedings of 2011 IEEE International Conference on High Performance Computing and Communications (HPCC-2011)*, pp834–838, Banff, Alberta, Canada, September, 2011.
2) SPECjAppServer2004, `http://www.spec.org/jAppServer2004/` .
3) SPECjEnterprise2010, `http://www.spec.org/jEnterprise2010/` .
4) 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.
5) Kazuaki Takahashi and Hitoshi Oi, "Measurement of SPECjEnterprise2010", IPSJ SIG Technical Report, Vol.2011-EVA-34 No.4, March 2011.