A Comparison of SPECjAppServer2002 and SPECjAppServer2004

Authors:

1. Lixin Su (corresponding author)
   - Affiliation: ECE Dept. Univ. of Wisconsin - Madison
   - Address: 4614 Engineering Hall, 1415 Engineering Dr, Madison, WI 53706
   - E-mail: lsu@cae.wisc.edu
2. Kingsum Chow
   - Affiliation: Software Solutions Group, Intel Corporation
   - Address: M/S: JF1-239, 2111 NE 25th Ave, Hillsboro, OR 97124
   - E-mail: kingsum.chow@intel.com
3. Kumar Shiv
   - Affiliation: Software Solutions Group, Intel Corporation
   - Address: M/S: JF1-239, 2111 NE 25th Ave, Hillsboro, OR 97124
   - E-mail: kumar.shiv@intel.com
4. Ashish, Jha
   - Affiliation: Software Solutions Group, Intel Corporation
   - Address: M/S: JF1-239, 2111 NE 25th Ave, Hillsboro, OR 97124
   - E-mail: ashish.jha@intel.com

A Comparison of SPECjAppServer2002 and SPECjAppServer2004

Lixin Su
ECE, Univ. of Wisconsin - Madison
lsu@cae.wisc.edu

Kingsum Chow, Kumar Shiv, and Ashish Jha
Intel Corp.
{kingsum.chow, kumar.shiv, ashish.jha}@intel.com

Abstract

Multi-tier and mutli-threaded commercial workloads evaluating middle tier performance are playing an important role in influencing computer server designs. Many middle tier servers are actually J2EE application servers. The SPECjAppServer benchmarks from SPEC have been some of the standard J2EE application server evaluation workloads since 2001. This paper attempts to compare and contrast workload behaviors between SPECjAppServer2002 (the retiring version) and SPECjAppServer2004 (the future version) as characterized by performance measurements. We set up SPECjAppServer2002 and SPECjAppServer2004 on the same hardware and software configurations. We characterized the workload differences in three different layers – the system behavior, the execution profile, and the microarchitecture performance characteristics. Early data indicates that SPECjAppServer2004 appears to demand more memory, disks and network bandwidth, along with a higher demand on the Java virtual machine instead of the Java code. The behavior shift is consistent with the new features (e.g. HTTP servlets) introduced in the benchmark.

1. Introduction

Computer servers represent a significant segment of the computer system market. The development of computer servers has been relying on technology breakthroughs and application demands. On the application side, CPU benchmarks, scientific computing applications, and commercial workloads have been the major forces that have been pushing the better computer server designs. The rapid growth of Internet and E-commerce has put commercial workloads into a very important role in defining the design of the next generation of computer servers. Commercial workloads are usually multi-tier and multi-threaded. They test the performance of the Internet backend, the middle tier, and the client side. The middle tier performance is a very important system performance metric that influences many computer server purchases since the middle tier connects the backend (database) and the clients and plays a key role in the overall performance.

The SPECjAppServer[1] series workloads from SPEC have become the standard applications for evaluating middle tier performance. SPECjAppServer2001 (based on ECperf) [17] was the first release of the SPECjAppServer series and the current version of the workload is SPECjAppServer2002 [16]. Due to the fast improvement of the middle tier and the J2EE application server standards, SPEC will soon retire SPECjAppServer2002 and use SPECjAppServer2004 [15] as the standard benchmark for evaluating middle tier performance.

Our work presents an evaluation of SPECjAppServer2004. The system behavior, the execution profile and the microarchitectural characteristics are discussed for SPECjAppServer2004. A comparison with the current SPECjAppServer2002 is also provided. The same hardware configuration and software stack are employed to run the workloads to ensure a fair comparison between SPECjAppServer2002 and SPECjAppServer2004. An effective system tuning methodology [5][6] is used to tune the performance for both workloads.

We found that at the system level, SPECjAppServer2004 utilizes more system resources such as heap memory, disks, and network bandwidth than SPECjAppServer2002 when the application server achieves the same CPU utilization for both workloads. The two workloads show very similar execution profiles with the new workload having a bigger portion of the JVM execution. At the microarchitecural level SPECjAppServer2004 has a much higher path length and poorer branch behavior. However, it experiences fewer last-level cache misses and TLB misses.

In the rest of the paper, section 2 explains the differences between SPECjAppServer2002 and SPECjAppServer2004. Section 3 describes the system behavior comparison, section 4 shows the execution profiles, and section 5 presents the microarchitecture performance characteristics. Section 6 surveys the related work and it is followed by the conclusion in section 7.

[1] SPECjAppServer is a trademark of the Standard Performance Evaluation Corp. (SPEC).

2. SPECjAppServer2002 and 2004

The series of SPEC's jAppServer (Java Application Server) benchmarks are multi-tier benchmarks for measuring the performance of Java 2 Enterprise Edition (J2EE) technology-based application servers. It's designed to exercise the Java Enterprise Application Server (J2EE), the Java Virtual Machine (JVM), as well as the server Systems Under Test (SUT). The benchmarks are not focused on the front-end clients or the back-end database machines.

SPECjAppServer2002 is the second member in the SPECjAppServer benchmark family. It is ported from SPECjAppServer2001, which is the first of the series, and it was basically a repackage of the ECperf benchmark. Going back in time, ECperf was developed under the Java Community Process (JCP), designed to meet the J2EE 1.2 standard specification. SPECjAppServer2002 focuses on the performance of Enterprise Java Bean (EJB) 2.0 compliant J2EE 1.2 application servers. Standard EJB 2.0 features such as local interfaces, the EJB-QL query language, and Container Managed Relationships (CMRs) are fully tested in SPECjAppServer2002. In addition, the application servers' abilities such as memory management, connection pooling, passivation/activation, object persistence, and caching are also tested in SPECjAppServer2002. The benchmark emulates a heavyweight manufacturing, supply chain management (SCM) and an order/inventory system representative of those used at a Fortune 500 company. The emulation is based on the Remote Message Interface (RMI) driven by the drivers (the clients in the multi-tier environment). The emulated business model includes four domains – customer, manufacturing, supplier, and corporate.

SPEC's latest benchmark is SPECjAppServer2004, which is an enhanced version of the SPECjAppServer2002 benchmark that includes a modified workload and features that stress all major J2EE technologies implemented by compliant J2EE 1.3 application servers. The newly added features include the Java Message Service (JMS), Message Driven Beans (MDBs), and the Web container including Servlets and JSPs. In addition to the application servers' abilities tested in SPECjAppServer2002, the new benchmark stresses new abilities like Web page generation and message queuing. The inclusion of the Web layer on the application server requires more powerful drivers to parse the HTTP messages. The inclusion of JMS and MDBs provides asynchronous communication across the various domains like manufacturing and ordering to address the scaling issues seen in SPECjAppServer2002. Compared with SPECjAppServer2002, the new benchmark tries to increase the memory footprint on the application server in order to stress the JVM, which is at the core of the J2EE server. In the business domain perspective, SPECjAppServer2004 emulates an automobile manufacturing company and its associated dealerships. The emulated business model adds a new domain called dealer. The dealers interact with the system using Web browsers (simulated by the driver) while the actual manufacturing process is accomplished via RMI (also driven by the driver). SPECjAppServer2004 also increases the database cardinality in order to remove the inherent scaling issues in SPECjAppServer2002 database, in order to focus on the Application Server performance. Apart from other differences, it also removes the price/performance metric that exists in both SPECjAppServer2001 and SPECjAppServer2002.

At the end, even though these SPECjAppServer series of benchmarks are designed on the same philosophy to measure the performance of Java 2 Enterprise Edition (J2EE) technology-based application servers, their performances are not comparable to each other due to different optimization opportunities and constraints imposed for each of them.

3. Experiment Methodology

This section describes our experiment methodology to characterize the workload differences between SPECjAppServer2002 and SPECjAppServer2004. The comparison is not focused on benchmark scores. According to SPEC, the two benchmarks are not directly comparable in terms of their scores. Our comparison also does not suggest that one benchmark be better than the other. The main purpose of this comparison is to find the system performance impacts on both software and hardware by the two benchmarks and the insights we can draw from the evolutions of J2EE application server benchmarks in terms of software and hardware designs.

HW	4P 1.4GHz Itanium® 2 processors with 4MB L3 Cache, 16GB memory, Intel Pro1000 XT server adapter
SW	Windows Server 2003, JDBC driver, a leading JVM, a leading J2EE application server

Table 1. Hardware and software configurations for the application server.

In order to make a fair comparison, we use the same hardware configuration and software stack for both benchmarks. The database and drivers are guaranteed powerful enough for both SPECjAppServer2002 and SPECjAppServer2004. See table 1 for the details in the hardware and software used in the application server. The JVM uses a 12GB heap and the best know memory optimization parameters for this JVM. We also make the application server CPU utilization comparable between the two benchmarks. In SPECjAppServer2002 and 2004, many factors such as database performance, thread queues on the driver, and the response times, can affect the application server CPU utilizations. An effective performance tuning methodology [5][6]was employed to achieve the high CPU utilizations on the application server in the experiments.

The results[2] from both benchmarks met the response time requirements for the benchmarks [15][16] to ensure a reasonable transaction mix in the workloads. The comparison focuses on the application server instead of the database and the driver since both benchmarks are J2EE application server benchmarks.

The following three sections compare the two workloads in terms of the system behavior, the execution profiles, and the microarchitecture performance.

4. System Behavior

Section 4 focuses on the application system behavior comparison between SPECjAppServer2002 and SPECjAppServer2004. The behaviors of major system components – CPU, memory, disks and network are shown for both SPECjAppServer2002 and SPECjAppServer2004. All the measured stats shown in this section are per transaction based stats. One must be careful with the comparison based on per transaction data as one Sjas02 transaction is not equivalent to one Sjas04 transaction as they are different workloads. Other approaches, such as normalization of measurements per unit of time and assembly instructions, were considered. However, a transaction is a unit of work defined in the workloads and thus per transaction based stats better illustrate the workload evolution.

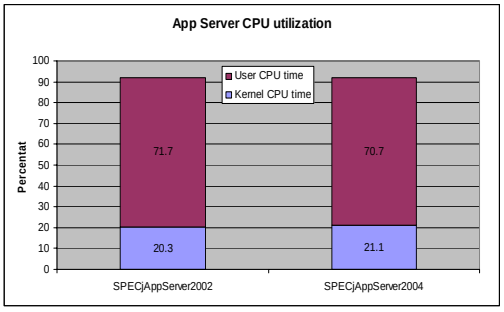

Figure 1. The CPU utilization breakdown on the application server for both SPECjAppServer2002 and SPECjAppServer2004.

Figure 1 shows the application server CPU utilization for the two benchmarks. The application server CPU utilizations are around 92% for both SPECjAppServer2002 and SPECjAppServer2004. The kernel and user CPU time breakdowns are very similar between the two benchmarks. SPECjAppServer2004 has a slightly higher kernel usage, which is probably due to the increased network activities. In addition, the two workloads have different response time requirements and transaction queues. The response times and the transaction queues can affect the kernel/user execution time breakdown.

	Sjas02	Sjas04
# GC per tran	0.0000068	0.000057
GC pause time per tran	2.6µs	77.4µs
Heap after GC	519MB	1,682MB
# large objects allocated per tran	2.9	1.6
# bytes allocated for large obj per tran	17,404	12,034
#bytes allocated for small obj per tran	25,768	175,077

Table 2. Heap memory usages by SPECjAppServer2002 and SPECjAppServer2004.

The memory usage is mainly due to the heap usage by Java. Table 2 describes the main characteristics of the heap usage by two benchmarks. In the table Sjas02 stands for SPECjAppServer2002 and Sjas04 represents SPECjAppServer2004. In the rest of the paper, we use the same naming conventions for SPECjAppServer2002 and SPECjAppServer2004. For each transaction, SPECjAppServer2004 executes garbage collections 7 times more than SPECjAppServer2002. SPECjAppServer2004 also spends about 30 times more time in the garbage collection for each transaction. GC frequency and GC pause time vary according to the JVM, the GC algorithm, and the heap size. In our experiment, we use a leading JVM, 12GB heap memory, and a parallel GC algorithm. SPECjAppServer2004's after GC heap size is more than 2 times bigger than SPECjAppServer2002's. SPECjAppServer2004 allocates many more bytes for small objects in each transaction than SPECjAppServer2002 while the bytes allocated for large objects in each transaction slightly decrease.

	Sjas02	Sjas04	Ratio
Rd Bytes/Tran	0.0047	4.7	1,000
Wrt Bytes/Tran	9.0	5,139	573
Rds/Tran	0.0000006	0.00057	989
Wrts/Tran	0.00084	0.6868	819

Table 3. Disk activities of SPECjAppServer2002 and SPECjAppServer2004.

Disk performance is an important factor that affects the overall system performance for many server workloads. However, there are few disk accesses in properly configured SPECjAppServer2002 and SPECjAppServer2004 even though the increase of the disk activity is the general trend of the J2EE application server workloads. Table 3 shows that SPECjAppServer2002 hardly has any disk reads/writes in a transaction. SPECjAppServer2004 has significantly more disk accesses but the average number of disk access per transaction is still less than 1. For both workloads, write traffic dominates the read traffic. The significantly

[2] The SPECjAppServer results in this publication have not been reviewed or approved by SPEC. No comparison nor performance inference should be made against any published SPEC result.

increased disk traffic is due to the increased pressure of EJB beans in SPECjAppServer2004 and also the addition of MDBs. We suspect that the database tables become bigger and there is a higher probability of fetching different records from the database tables by the application server in SPECjAppServer2004. Accordingly, the application server needs more EJB beans to track the database records and hence more activation and passisvation of EJB's. This results in the higher disk utilization. In addition, the MDBs write their messages to the disk and it is another source of the increased disk traffic.

	Sjas02	Sjas04	Ratio
Bytes sent per tran	2,370	117,430	49.6
Bytes received per tran	3,170	12,076	3.8
Packets sent per tran	17	79	4.6
Packets received per tran	17	70	4.1

Table 4. Network traffic of SPECjAppServer2002 and SPECjAppServer2004.

Table 4 shows the inbound and outbound traffic of the application server. The last column gives the ratios between the third column and the second column. All the ratios are the ratios of per transaction based stats. In the outbound traffic, the number of packets sent stays almost unchanged while the average packet size (bytes sent per tran / packets sent per tran) increases by more than an order. We think that the increase of the outbound packet size is mainly due to the service of HTTP request of the driver, which is an added feature in SPECjAppServer2004.

SPECjAppServer2004 stresses the application system differently from SPECjAppServer2002. When the same CPU utilization is achieved, SPECjAppServer2004 puts more stresses on the major system components such as memory and network. While the disk activity is increasing dramatically in SPECjAppServer2004, it still appears to be small and does not impact the performance very much. The changes in the utilizations of different system components are introduced by the newly added workload designs in SPECjAppServer2004.

5. Execution Profiles

The software stack running on the application server includes a variety of components – the OS, the Java Application Server, the JVM, and other small components. All the software components affect the workload execution and the overall system behavior. To show the contributions of different software components to the execution time, the retired instruction, and the CPI, we used Intel VTune Performance Analyzer [23] to collect the execution profiles. VTune data is based on the real-time sampling and the sampling data might show variances from run to run due to both sampling randomness and workload fluctuations. The data is collected off the GC pause time to avoid the GC's impact on the execution profile. The execution profiles off and on the GC time behave differently but the analysis of the execution profile on the GC time can be performed in a similar way.

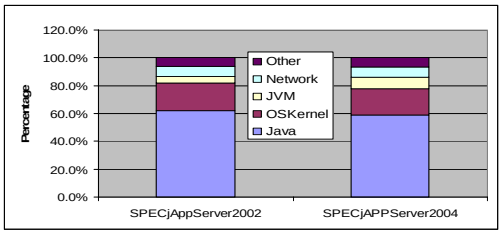

Figure 2. The execution time breakdown among Java, the OS kernel, the JVM, the network and other small software components. .

Figure 2 indicates that both benchmarks have a very similar execution time breakdown among the major software components. The breakdown helps understand what software components really take the execution resource and can be the possible optimization targets. Java accounts for a significant portion of the execution time, 61.8% for SPECjAppServer2002 and 58.8% for SPECjAppServer2004. Java includes both the workload itself and the J2EE application server code. The OS kernel also registers for a big chunk of the execution time, 19.9% for SPECjAppServer2002 and 19.0% for SPECjApplication2004. The JVM execution times are 5.1% for SPECjApplication2002 and 8.1% for SPECjApplication2004. Our data is collected off the GC time. If the data is collected during GC, the JVM's execution time will increase to about 10% with SPECjAppServer2004 still having a larger JVM execution time. The network related software execution times for SPECjAppServer2002 and SPECjAppServer2004 are 7.1% and 7.5% respectively.

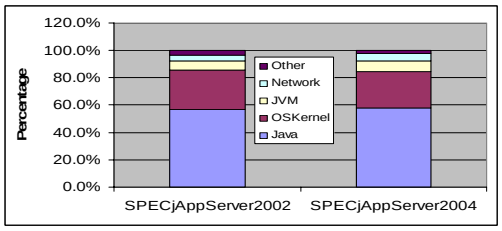

Figure 3. The retired instruction breakdown among Java, the OS kernel, the JVM, the network and other small software components.

The retired instruction breakdown provides a complementary picture of the execution time breakdown in terms of understanding the execution profile of a workload. Figure 3 presents the retired instruction breakdowns for SPECjAppServer2002 and SPECjAppServer2004. Java retires 56.7% of the overall instructions for SPECjAppServer2002 and 57.6% of the total instructions

for SPECjAppServer2004. The OS kernel is the second largest component retiring instructions and it retires 29% and 26.7% for SPECjAppServer2002 and SPECjAppServer2004 respectively. The number of retired instructions by the JVM increases from 6.4% for SPECjAppServer2002 to 7.9% for SPECjAppServer2004. The network related software retires 4.2% of the instructions for SPECjAppServer2002 and 5.7% of the instructions for SPECjAppServer2004.

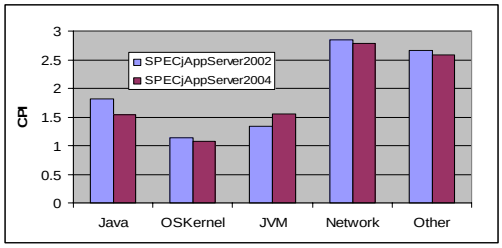

Figure 4. The CPIs of different software components Java, the OS kernel, the JVM, the network and other small software components.

The CPIs of different components provide an insight about what software components slow down the workload. Combined with the execution and the retired instruction breakdowns, they indicate what software components should be the optimization focuses. Figure 4 presents the CPIs of different software components for SPECjAppServer2002 and SPECjAppServer2004. The OS kernel has a relatively low CPI (1.144 in SPECjAppServer2002 and 1.073 in SPECjAppServer2004). The network software and Java have high CPIs and they also contribute to a large portion of the execution time. Therefore, optimizations of the application server and the network software can dramatically improve the performance. The JVM is the only software that experiences a CPI increase from 1.333 in SPECjAppServer2002 to 1.555 in SPECjAppServer2004. From the execution time breakdown, we know that the JVM execution time also increases in the newer version of the benchmark, about 8 per cent of the off-GC execution time. The JVM optimization is an opportunity to further improve the SPECjAppServer2004 performance.

The execution profiles of SPECjAppServer2002 and SPECjAppServer2004 show that the Java application server, the OS kernel, and the JVM account for big portions of the execution time and the retired instruction. The Java application server and the JVM have significantly higher CPIs than the OS kernel. They should be the major optimization focuses. In SPECjAppServer2004, the JVM should receive attention since it now accounts for more execution time and shows a higher CPI.

6. Microarchitecture Performance Characteristics

In addition to software optimizations, microarchitecture is an important factor of the application server performance. Understanding the microarchitecture performance characteristics can help understand the strengths and the weaknesses of the microarchitecture and design better future microprocessors. Section 6 examines and compares the processor-independent microarchitecture performance characteristics of SPECjAppServer2002 and SPECjAppServer2004.

	Ratio
PL	4.57
Kernel PL	4.04
User PL	5.05

Table 5. The path length ratios between SPECjAppServer2004 and SPECJAppServer2002. In the table PL stands for path length.

Table 5 shows the ratios of SPECjAppServer2004's path lengths and SPECjAppServer2002's path lengths. The path length is the number of instructions retired per transaction by the application server. The kernel path length is the number of kernel instructions retired per transaction by the applications server. The user path length is the number of user instructions retired per transaction by the application server. We can see that SPECjAppServer2004 has a much larger path length in both kernel and user path lengths. In addition to the data shown in table 5, there are many NOPs for both workloads. The percentages of NOPs among all the instructions executed are 25.7% for SPECjAppServer2002 and SPECjAppServer2004. The newer benchmark has a slightly higher NOP percentage. This again suggests opportunities for JVM vendors.

	Sjas02	Sjas04	Ratio
Total br (BPI)	0.097	0.1093	1.127
IP relative br (BPI)	0.0755	0.0829	1.098
Return br (BPI)	0.0141	0.0154	1.092
Indirect br (BPI)	0.0045	0.0048	1.067
IP relative br ratio	0.778	0.759	0.974
Return br ratio	0.146	0.141	0.969
Indirect br ratio	0.046	0.044	0.947
Branch mispredic ratio	0.137	0.163	1.189
Taken branch ratio	0.655	0.589	0.901

Table 6. Branch information for SPECjAppServer2002 and SPECjAppServer2004. BPI stands for branches per instruction.

Table 6 presents the branch statistics for SPECjAppServer2002 and SPECjAppServer2004. SPECjAppServer2002 has 0.097 branches per instruction

while SPECjAppServer2004 has an increase of 12.7% of branch instructions at 0.1093 branches per instruction. The increase of branches is probably due to the workload design changes and the increased pressure on the JVM. The distributions of IP relative branches, returns, and indirect branches are fairly similar between the two benchmarks. Both benchmarks have a higher misprediction ratio compared with most of the SPEC CPU2000 benchmarks. SPECjAppServer2004 has an even higher branch misprediction ratio at 16.3 % in comparison with 13.7% for SPECjAppServer2002. The higher branch misprediction ratio in SPECjAppServer2004 indicates that better branch predictors are always welcome for future processor designs.

	Sjas02	Sjas04	Ratio
I cache miss rate	0.07	0.0717	1.024
D cache miss rate	0.034	0.0335	0.985
L2 cache miss rate	0.0303	0.0257	0.848
L3 cache miss rate	0.0034	0.003	0.882

Table 7. Cache miss information for SPECjAppServer2002 and SPECjAppServer2004. The cache miss rate stands for the number of cache misses per instruction.

Table 7 gives the miss rates of different caches. In both benchmarks, the instruction cache miss rate doubles the L1 data cache miss rate. The cache miss rate of the second level cache is an order magnitude larger than that of the third level cache. The L1 caches' miss rates remain nearly unchanged between SPECjAppServer2002 and SPECjAppServer2004. However, the miss rates of L2 and L3 caches are more than 10% less in SPECjAppServer2004. The lower last level cache miss rates show that SPECjAppServer2004 has a better instruction/data access locality than SPECjAppServer2002 when large caches are present. SPECjAppServer2004's instructions and data fit large caches better.

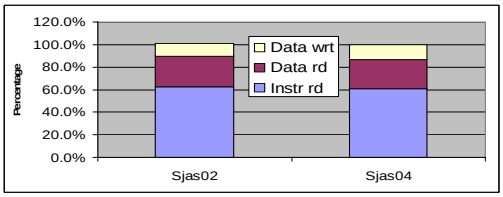

Figure 5. L3 access breakdown for SPECjAppServer2002 and SPECjAppServer2004.

Accesses to L3 can be caused by instruction reads, data reads, and data writes. The L3 access breakdown is shown in figure 5. The breakdown shows a similar behavior between SPECjAppServer2002 and SPECjAppServer2004. The percentages of L3 accesses caused by instruction reads, data reads, and data writes are 62.6%, 26.9%, and 11.4% in SPECjAppServer2002. The breakdown for SPECjAppServer2004 is 61.1%, 25.7%, and 13.2. The writes' contribution to L3 accesses is slightly higher in SPECjAppServer2004.

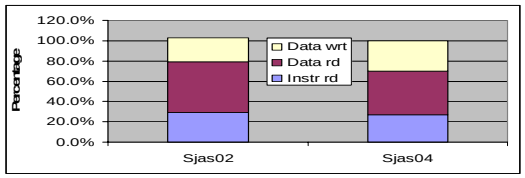

Figure 6. L3 miss breakdown for SPECjAppServer2002 and SPECjAppServer2004.

In addition to the L3 access breakdown, we also characterize the L3 miss breakdown. Figure 6 shows the L3 misses breakdown among instruction reads, data reads, and data writes. The sum of these three types of L3 access is larger than one because of the hardware counter variance during the experiment. Compared with their contributions to L3 acesses, data accesses account for bigger portions of L3 misses while instruction accesses lead to fewer L3 misses. It is not surprising that the data access is more irregular since the JVM does many object optimizations and tends to move objects around within the memory. SPECjAppServer2004 has a bigger contribution to the L3 misses by the writes. We suspect that the increased EJB bean pressure, the added MDBs, and the optimizations of more small objects lead to more L3 misses by the writes. The contributions to L3 misses by instruction reads, data reads, and data writes are 29.4%, 50%, and 23.5% for SPECjAppServer2002 while 26.7%, 43.3%, and 30.0% for SPECjAppServer2004.

	Ratio
L1 ITLB miss rate	1.149
L2 ITLB miss rate	0.941
DTLB miss rate	0.725

Table 8. The ratios of TLB miss rates between SPECjAppServer2004 and SPECjAppServer2002.

The changes in TLB behaviors from SPECjAppServer2002 to SPECjAppServer2004 are shown in table 8. The L1 ITLB miss rate is about 15% higher in SPECjAppServer2004 while fewer L1 ITLB misses lead to further L2 ITLB misses. The L2 ITLB miss rate is roughly 6% smaller. SPECjAppServer2004 has a significantly smaller DTLB miss rate – 27% smaller.

The comparison of the microarchitecture performance characteristics shows that SPECjAppServer2004 has a much larger path length due to the workload design. The new benchmark has a higher branch misprediction ratio even though the branch profile remains similar. SPECjAppServer2004 causes lower last-level cache miss rates and last level TLB miss rate. This seems to deviate from the trend that memory pressure is becoming more evident and larger caches are necessities for future processors. It is also possible that early implementations of

Sjas04 workloads are not as efficient as well tuned Sjas02 setups.

7. Related Work

This paper is an attempt to compare SPECjAppServer2002 [16] and SPECjAppServer2004 [15] workloads on one specific platform. The workloads belong to SPEC's multi-tier Java application server benchmark series. The predecessor of SPECjAppServer2002 is SPECjAppServer2001 [17], which is derived from ECperf. The SPECjAppServer workloads are important server benchmarks that assist in the design of servers. Many papers have been published about ECperf and SPECjAppServer2002 and their influences in the computer server designs. Chow et al. [2] studies the variance of ECperf-like workloads and the relationship between Java application workload implementations and CPU designs. A detailed architectural level analysis of ECperf [19] was performed by Karlsson et al. A real-time cache simulator [18] was developed to study the memory behavior of SPECjAppServer2002 [3]. Meanwhile, a sophisticated system performance tuning methodology for ECperf and SPECjAppServer2002 has been developed [4][5][6].

Java based server workloads have been gaining gradual attention from both academia and industry. Cain et al. [1] implemented Java-based TPC-W and studied its architectural behavior in both real systems and the simulator (PHARMsim [7]). Another Java based two-tier benchmark, VolanoMark, was studied by Luo et al [11]. The memory behavior and the architecture influence of SPECjvm98 were investigated in [9] and [12]. Marden et al. compared non-Java workloads with Java workloads in [20].

8. Conclusions

This paper presents a detailed comparison of a new workload – SPECjAppServer2004 and its predecessor – SPECjAppServer2002. Both workloads are complex multi-tier J2EE application server benchmarks. The two workloads are set up in exactly the same hardware and software configurations. The comparison is performed when both workloads achieve an application server CPU utilization of 92%. The analysis is based on three layers: the system behavior, the execution profiles, and the microarchitecture performance characteristics.

Our data shows that SPECjAppServer2004 demands more system resources. The after GC heap size increases by more than 2 times, the disk traffic grows by more than 100 times, and the network traffic rises by more than 10 times. The new workload also stresses the JVM more than the Java code. The JVM's execution time portion increases by roughly 20% from 6.4% to 7.9%. In the microarchitectural level, our data indicates that the new workload results in about 13% more branches per instruction and a 19% higher branch misprediction ratio. However, the last level cache miss rate lowers by about 12%, the last leve ITLB miss rate reduces by roughly 6%, and the DTLB miss rate decreases by 27% or so.

9. Acknowledgement

We would like to thank Ricardo Morin, Suresh Srinivas, Ranajeet Talwalkar, and Wen-Hann Wang for the valuable feedbacks. We also thank Chris Elford and Mahesh Baht for insightful discussions.

References

[1] H. Cain, R. Rajwar, M. Marden, and M. Lipasti. "An Architectural Evaluation of Java TPC-W." In Proceedings of the Seventh International Symposium on HPCA, January 2001.

[2] K. Chow, M. Bhat, A. Jha, and C. Cummingham. "Characterization of Java Application Server Workloads." In IEEE 4th Annual WWC in conjunction with MICRO-34, pages 175-181, 2002.

[3] N. Chalainanont, E. Nurvitadhi, K. Chow, and S. Lu. "Characterization of L3 Cache Behavior of Java Application Server." 7th Workshop on CAECW, 2004.

[4] K. Chow and G. Deisher. "SPECjAppServer2002 Performance Tuning." WebLogic Developer's Journal, September 2003.

[5] K. Chow, R. Morin, K. Shiv. "Enterprise Java Performance: Best Practices." Intel Technical Journal, 2003 Q1.

[6] K. Chow, Z. Yu, L. Su, M. Jones, and H. Yan. "An Automation and Analysis Framework for Testing Multi-tiered Applications." PNSQC Conference, 2004.

[7] H. Cain, K. Lepak, B. Schwartz, M. Lipasti. "Precise and Accurate Processor Simulation." 5th Workshop on CAECW, 2002.

[8] K. Lepak, G. Bell, and M. Lipasti. "Silent Stores and Store Value Locality." In IEEE Trans. on Computers, Vol. 50, No. 11, November 2001.

[9] J. Kim and Y. Hsu. "Memory System Behavior of Java Programs: Methodology and Analysis." In Proceedings of the 2000 Int. Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS 2000), June 2000.

[10] T. Li, L. John, N. Vijaykrishnan, A. Sivasubramaniam, J. Sabarinathan, and A. Murthy. "Using Complete System Simulation to Characterize SPECjvm98 benchmarks." In Proc. of ACM Int. Conf. on Supercomputing (ICS 2000), May 2000.

[11] Y. Luo and L. John. "Workload Characterization of Multithreaded Java Servers." In Proceedings of the 2001 IEEE (ISPASS 2001, April 2001.

[12] R. Radhakrishnan, N. Vijaykrishnan, L. John, and A. Sivasubramaniam. "Architectural Issues in Java Runtime Systems." In Proceedings of the Sixth International Symposium on High Perf. Comp. Architecture (HPCA-VI), January 2000.

[13] B. Rychlik and J. Shen, "Characterization of Value Locality in Java Programs." In Proc. of the Workshop on Workload Characterization, ICCD, September 2000.

[14] K. Chow, M. Bhat, A. Jha, and C. Cummingham. "Characterization of Java Application Server Workloads." In IEEE 4th Annual WWC in conjunction with MICRO-34, pages 175-181, 2002.

[15] http://www.spec.org/jAppServer2004/
[16] http://www.spec.org/jAppServer2002/
[17] http://www.spec.org/jAppServer2001/
[18] N. Chalainanont, E. Nurvitadhi, R. Morrison, S. Lu, L. Su, K. Chow, and K. Lai, "Real-time L3 Cache Simulations Using the Programmable Hardware-Assisted Cache Emulator (PHA$E)," proc. of wwc-6, Autin, TX 2003.
[19] M. Karlsson, K. E. Moore, E. Hagersten, and D. A. Wood. "Memory System Behavior of Java-Based Middleware." In Proceedings of the Ninth HPCA, February 8-12, 2003
[20] M. Marden, S. Lu, K. Lai, and M. Lipasti. "Comparison of Memory System Behavior in Java and Non-Java Commercial Workloads." 5th Workshop on CAECW, 2002.
[21] P. Koka and M. Lipasti. "Characterization of an IMAP Server on a Shared-Memory Multiprocessor." 7th Workshop on CAECW, 2004.
[22] K. D. Safford, "A Framework for Using the Pentium's Performance Monitoring Hardware", *M.S. Thesis*, University of Illinois at Urbana-Champaign, 1997
[23] http://www.intel.com/software/products/vtune/