Analyzing the Performance of a Multithreaded Application Server and its Limitations in a Multiprocessor Environment

Josep Oriol Fitó
fito@ac.upc.edu

Jordi Guitart
jguitart@ac.upc.edu

Computer Architecture Department - Technical University of Catalonia
C/ Jordi Girona 1-3, Campus Nord UPC, Mòdul C6, E-08034 Barcelona (Spain)

Abstract

The presence of application servers is more common than ever due to the quick evolution of e-Business and e-Commerce services. These software engines are able to host multiple web applications, which could receive dissimilar workloads, in a unique server machine. This means that diverse web applications have to share the limited server's resources. Furthermore, those applications differ in both resource requirements and performance goals. In order to properly exploit their capabilities, a good and accurate study must be performed to better understand the benefits and limitations of hosting various types of web applications in a single server.

In this paper, we present an in-depth study of the performance of an application server in a multiprocessor environment which process representative workloads of nowadays. We evaluate its performance using a fine Web server benchmark, SPECweb2005, which characterize three real web usage patterns. For each case, we expose the obtained results from the performed experiments on multithreaded java application server. Specifically, we have performed a study of the server's scalability in this multiprocessor environment when running with different number of processors. Afterwards, we examine which are the underlaying bottleneck resources for the server's performance for each one of the afore-mentioned workloads.

Keywords: application server, multiprocessor environment, performance scalability, SPECweb2005, workload characterization

1 Introduction

Nowadays, the Internet trend towards "everything-as-a-service" has an obvious side effect: the computing capacity is required in 'The Cloud', the server's area ([18], [20]). Moreover, is expected that these servers provide a multitude of disparate services at the same time. In turn, these services could be provided by dynamic web applications that are hosted on those application servers. This applications, which coexist in a single machine, have dissimilar both performance targets and Service Level Agreements (SLAs). Furthermore, the most important for us is that they also differ in terms of resource requirements. For this simple reason, and pursuing the same objective that the well-known consolidation strategy, it is necessary that these machines are not underutilized servers for two motives. Firstly, to reduce the number of servers needed to provide the afore-mentioned services. Secondly, with the aim to improve the energy efficiency of the IT equipment. This will result in reducing the wasted energy and consumed power and, consequently, in reducing system costs. In fact, these two objectives are gaining strength in the design, construction and operation of servers.

Following these goals, a good study of the capabilities of the target machine in question is essential and will be very useful. It can help us to better understand how the target server acts in front of this type of workloads. As well, it could be used to find the more adequate consolidation strategy that we should perform if we take into account both resources demands and performance that is expected of it.

In the case that occupies us, the application servers, we must evaluate its behavior considering the most current usages. Hence, in this scenario, is very convenient to use the SPECweb2005 benchmark [15]. It has been designed with three different workloads that characterize the most common usage of Web users of today and tomorrow: Banking, E-commerce and Support, which emulate an online banking, an e-commerce site and a vendor support site providing downloads, respectively.

This paper presents a deep study around the performance of an application server which receives representative workloads of today. We use Apache Tomcat [2] as the multithreaded java application server to deploy three web applications corresponding to each type of workload. We show the achieved results from the performed tests. Furthermore, we explain which are the interesting performance metrics that can be extracted from this and how we should treat it to obtain graphics with enough representativeness.

Recently, the scalability of web servers has become a key issue in order to support the maximum number of concurrent clients demanding dynamic content. For this reason, we perform a study of the server's scalability in a multiprocessor environment when running with different number of processors: from one to four processor units.

Additionally, we present an examination about what are the underlaying hardware resources that limits the performance of the server. Therefore, the main goal is to detect what are the resource bottlenecks in this multiprocessor environment for each one of the SPECweb2005 workloads.

The rest of the paper is organized as follows: in section 2 we discuss about the motivation of this work and the previous research related with it. Section 3 presents the experimental environment that we used. Section 4 expounds the evaluation of the obtained results from the performed tests and describes the analysis of the performance bottlenecks. Finally, Section 5 presents the conclusions of this paper and introduces the future work.

2 Motivation and Related Work

The utilization of benchmarks is very common, useful and necessary to evaluate the performance of hardware and software components/systems. In fact, a benchmark is the act of running a computer program in order to evaluate the relative performance of an object, normally by running a number of standard tests against it. The most important feature is that benchmarks are designed to act like a particular type of workload on a component or system. There are many types of benchmarks, but the most widely-used are synthetic benchmarks and application benchmarks. The first one consists in specially created programs that impose the workload on the desired component. The other one, called application benchmarks, run real-world programs on the target systems. Hence, application benchmarks usually give a much better measure of real-world performance on a given system. In the other hand, synthetic benchmarks are convenient for testing individual components, like a hard disk or networking device. Due to this great variety and possibilities, the use of benchmarks is highly prevalent.

Nowadays, the benchmarks created to evaluate the performance of web and application servers are gaining a remarkable protagonism. Here, in this paper, we use what we consider the best benchmark to evaluate these servers: SPECweb2005. We have found that there are some works dealing with SPECweb2005's issues.

Hariharan and Sun [12] make a brief but thorough summary about SPECweb2005 and its main characteristics. They present the workloads one by one and give the reader many technical aspects of them.

Mahadevan [14] utilizes a network processor for application-level offloading in order to improve Web server performance. He describes the working of Web servers and proxy servers, and their performance parameters. With the aim to demonstrate the improvements achieved, he provides a performance evaluation between this approach and the traditional host-only approach. This evaluation was performed using the SPECweb2005 benchmark.

Warner and Worley [19] examine various systems using ISS as the web server and PHP to assist dynamic content. This analysis is performed using SPECweb2005 benchmark and they provide results for a representative "real world" web server configuration. Nevertheless, they offer a comparison between these servers based on the SPECweb2005 overall score and overlook the most significant performance metrics that the benchmark gives us, such as throughput and response time.

On the other hand, evaluating the bottleneck resources for an application or web server running in a given environment is a task that have been already performed in some works. Carrera et al. [6] present the eDragon environment: a research platform created to perform complete performance analysis of new Web-based technologies. In fact, they describe the design and implementation of this platform and highlight some of its most important capabilities. Then, and using this platform, Guitart et al. [7] present a characterization of secure dynamic web applications scalability. In fact, they divide the work in two parts. First, they measure the vertical scalability of the server running with different number of processors. Second, they perform an in depth analysis of the server behaviour using the afore-mentioned platform. This analysis help them to determine the causes of the server overloading.

Veal and Foong [17] identify the major bottlenecks to scalability for a reference server workload on a commercial server platform, that is, SPECweb2005 Support workload. Their results show that the operating system, TCP/IP stack, application exploited parallelism, load imbalance and shared cache affected performance little. Leaving aside these potential bottlenecks, they determined that performance scaling was limited by the capacity of the address bus, which became saturated.

Further, Bosque et. al [4] present a work in which the performance of Apache web server is characterized on multicore chips using SPECweb2005. Their conclusions are around limitations related with data caches and main memory accesses.

Amza et al. [1] describe three benchmarks for evaluating the performance of Web sites with dynamic content. Furthermore, they present a performance evaluation of their implementations on contemporary commodity hardware. Their evaluation is focused on finding and explain the bottleneck resources in each benchmark.

Despite the above contributions, we think that it is interesting to provide an article that deepens in using SPECweb2005 and shows key performance metrics and assessments that can be extracted from it. Additionally, this paper contributes both to characterize the SPECweb2005 workloads and to find their resources demands in a multi-processor environment.

3 Experimental Environment

We can highlight the following three elements that make up the experimental environment.

3.1 Tomcat Servlet Container

We use Apache Tomcat v.6.0.14 [2] as multithreaded java application server. Apache Tomcat is an open-source servlet container developed at the Apache Software Foundation. It implements the Java Servlet 2.5 and the JavaServer Pages 2.1 (JSP) specifications from the Java Community Process, and makes available a Java HTTP web server environment. As is well-known, it is an interesting context for develop and deploy web applications and web services. It can operate as standalone server, serving both static and dynamic content, or as a complement for a web server, offering only dynamic content. In order to meet SPECweb2005 requirements, we use Tomcat as standalone server (i.e web server). Initially, we have configured it with two top-level entry points, non-SSL and SSL HTTP/1.1 connectors, with a maximum number of 100 HttpProcessors in the shared thread pool and a connection timeout of 10 seconds. In fact, it is important to note that this timeout dynamically varies depending on the input load received by the server. Finally, we have statically deployed three web applications which are used by SPECweb2005.

3.2 SPECweb2005 Benchmark

SPECweb2005 [15] is the next-generation SPEC benchmark for evaluating the performance of World Wide Web Servers. Like its predecessors, SPECweb99 and SPECweb99_SSL, SPECweb2005 continues the SPEC tradition of giving Web users the most objective and representative benchmark for measuring a system's ability to act as a web server. It is know that the usage characteristics of web applications are changing and that is why it is necessary to update the benchmarks to reflect them. For this reason, the SPECweb2005 benchmark includes many sophisticated and state-of-the-art enhancements to meet the modern demand of Web users of today and tomorrow:

- User think time between page requests is now modeled.
- Client-side flexibility has been greatly increased. Workload's modifications can be easily made via configuration files.
- The clients send conditional GETs requests to simulate browser caching effects.
- It was written in Java in order to give the users a cleaner and portable code.
- The application server dynamic content is implemented in PHP and JSP.

His working method is to emulate users sending browser requests over broadband Internet connections to a web server. In fact, it simulates a web server using dynamic web pages and a backend database. It is based on downloading pages, which are dynamically created files, followed by requests for embedded images within the files. Furthermore, it provides three new workloads based on analysis of web server logs and studies of sites from the web browser side. This is so due to the high variability in the security requirements and differences in the dynamic content in various web server workloads. As a result, it is becoming impossible to characterize the web performance of any server with any single workload. Next, there is a brief explanation of each of those already mentioned workloads:

1. **Banking**, based on Internet personal banking. Consequently, the typical requests of an online bank were simulated, such as login/logout, bank balance inquiry, money transfers, look at and modify the user profile, etc. All of these requests are based on SSL. Since in the first (and mandatory) transaction an SSL connection is established, all the above mentioned requests are based on SSL. Furthermore, the possible operations include both POST and GET HTTP methods. Continuing inside the design of this workload, it is important to note that it is the only one in which the session timeout is implemented. Specifically, about 20% of the incoming users do not logout, thus allowing the session timeout. Besides, and after each page request, there is a likelihood that the user goes through a think time, which averages about 9.98 sec (data extracted from [16]).

2. **E-commerce**, based on the workload characteristics of an e-commerce site. It was designed to simulate a Web server that sells computer systems; this includes allowing users to search, browse, customize, and purchase products. In fact, a customer visiting the emulated site passes through three distinct phases. The first is when the customer browses the website, looking for a product. The second is the customization phase, were the end user customizes the desired product. Finally, if the customer wants to check out or buy the product, then he must use requests based on SSL. For this reason, and unlike the above explained workload, only about two thirds of the user sessions use SSL connections. As well, as in the case of Banking workload, the transition between states is driven via a Markov Chain and user's think time is used in many user sessions. To conclude, it is valuable to note that in this workload are used both HTTP and HTTPS connections.

3. **Support**, based on the characteristics seen in sites that were used to download patches for computer support. Typically, this downloads are very large and the purpose of this workload is to stress the server I/O. This workload does not use SSL connections. The web client emulated enters in the system, searches for the right patch file and then begin to download this file. Also, the think time value used for this workload is about 5 seconds.

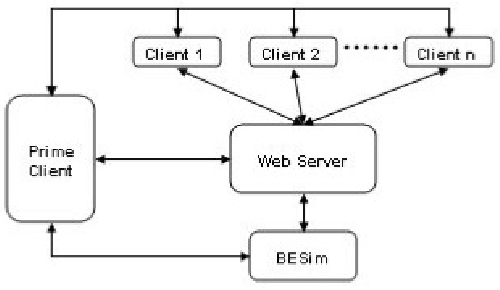


Figure 1: Logical components of SPECweb2005

Besides, and talking about the benchmark architecture, it has four major logical components as illustrated in Figure 1 (Source: [16]):

- **Client.** The benchmarks clients run the application program that sends HTTP requests to the server and receives HTTP responses from this one. This application program has been written in Java for portability.
- **Prime client.** This is the component that initializes and controls the behavior of the clients. Therefore, its main operations are to run initialization routines against web server and BeSim (Back-End Simulator), and collects and stores the results of the benchmark tests. It may be on the same physical system as one of the clients.
- **Web server or System Under Test.** Is the collection of hardware and software that handles the requests issued by the clients. Note that the application server is also called System Under Test (SUT) and henceforth we will use this terminology.
- **Back-End Simulator.** It is the logical component that is intended to emulate a back-end application server that the web server must communicate with in order to get back specific information needed to complete an HTTP response.

3.3 Hardware and Software Platform

In order to reproduce the desired situation with the utmost reality, we decided to use a total of up to 9 machines. The Tomcat application server, also known as System Under Test (SUT), runs on a 4-way Intel Xeon 3.16GHz with 16 GB of RAM memory. This application server was enveloped in the Sun Java Virtual Machine for Linux and the initial and maximum size of Java heap was set up to 1 GB. Moreover, Tomcat sends requests to the BackEnd Simulator (BeSim) and receives response back. This simulator, which is encapsulated into an Apache Web server, runs on a 2-way Intel Xeon 2.4GHz with 2 GB RAM. Another important logical component is the prime client. It runs, along with a simple client, on a machine like the previous one. And, additionally, we use five more clients (six in total) that send HTTP/HTTPS requests to the server. These clients run on the five remaining machines, which have the following CPU and memory capacity: 4-way Intel Xeon 1.4GHz with 2 GB RAM, 4-way Intel Xeon 3GHz with 4 GB RAM, 4-way Intel Xeon 2.66GHz with 2GB RAM and, lastly, two equal machines with 2-way Intel Xeon 2.4GHz alongside 2GB of RAM memory. All of this machines are connected using a 1Gbps Ethernet interface and run Linux Operating System (with kernel 2.6).

For each one of the performed tests, the benchmark was configured to emulate the desired workload expressed in number of simultaneous user sessions. Then, all the clients emulate these simultaneous sessions against the SUT during a run of 30 minutes and three times per test.

4 Evaluation

In this section we present the evaluation of the obtained results. We represent the server's performance using diverse metrics, such as throughput, response time and Quality of Service (QoS). Firstly, we perform a characterization of the SUT's scalability for each workload, one by one. Later, in the second part, we present an examination around the bottleneck resources that limit the performance of the server.

4.1 Scalability of the Server's Performance

In this first part we describe the scalability characterization of the SUT when receiving each one of the SPECweb2005 workloads in an individual way. We perform a vertical scalability of the multithreaded web server when running with different number of available processors, thereby determining the consequences of allocating more processors to it. As can be seen in the upcoming graphics, the caused effects are different in each of the workloads and later we will try to demonstrate and justify this fact.

The information obtained after the completion of each test allows us to extract the most representative metrics regarding with the performance of the server:

- **Throughput**. It express the ratio of requests per second (rps) served by the server.
- **Response time**. It refers to the amount of time server takes to return the results of a request to the user.

- **Quality of Service**. It comprises requirements on all the aspects of a connection, such as service response time, frequency response, etc.

4.1.1 Throughput

Figures 2, 3 and 4 show the Tomcat throughput for Banking, E-commerce and Support workload, respectively. The throughput is expressed in requests per second, and as a function of the simultaneous user sessions within the server when running with different number of processors (one to four).

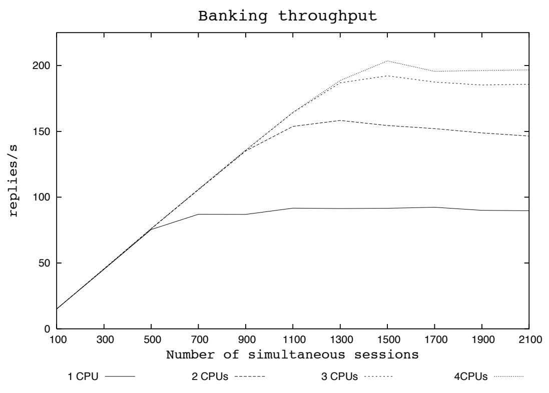


Figure 2: Banking throughput when running with different number of processors

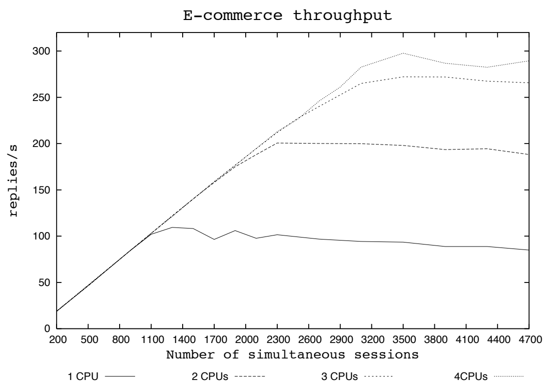


Figure 3: E-commerce throughput when running with different number of processors

All the Banking workload's requests are based on secure connections. For this reason, and considering the probable hypothesis that the CPU is the critical resource, we can see how the server's throughput scale according to the number of available CPUs. Nevertheless, it does not scale linearly

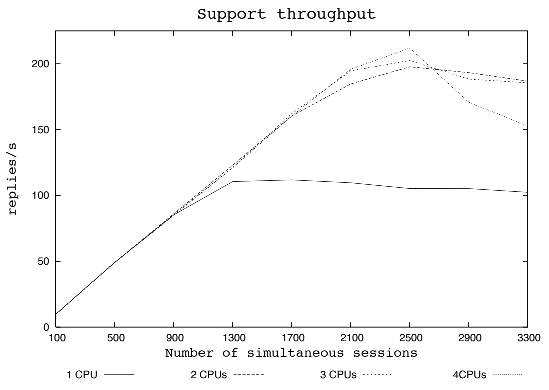


Figure 4: Support throughput when running with different number of processors

according to the number of allocated CPUs. For instance, considering 1500 simultaneous user sessions, the throughput obtained with 1 CPU is 91.5 rps and with 4 CPUs is 203.5 rps. Thus, it supposes an escalation of 4x in the critical resource and 2.2x in the performance. Further study is needed to see what is really happening.

Taking into account the characteristics of the E-commerce workload, in terms of SSL connections, this one is not fully based on this kind of connections. In fact, a high percentage of requests (90%) are performed in a secure context. For this cause, the shape of the Figure 3 is similar to Figure 2, but the performance takes longer to differentiate between having or not more CPUs.

The last situation, Support workload, is also quite particular. Looking at the scalability of the server, in terms of CPU resource, it looks like that this resource may not be critical as in the other two cases. In fact, with a single CPU the performance seems that is limited by it. On the other hand, with 2, 3 or 4 CPUs the throughput is very similar. Afterwards, in the next section, we present an examination which allows us to detect what is the critical resource.

As a summary, Table 1 shows the maximum achievable, and non-comparable, throughput for each of the workloads, differentiating between 1, 2, 3 or 4 CPUs allocated to the System Under Test:

Maximum achievable throughput (rps)			
CPUs	Banking	E-commerce	Support
1	92.2	109.4	111.8
2	158.2	200.6	197.7
3	192.2	272.2	202.5
4	203.5	297.6	211.9

Table 1: The maximum achievable throughput for each of the SPECweb2005 workloads

4.1.2 Response Time

Another typical metric that the community use to quantify the performance of servers is the average response time: the time taken to attend the client request and send the response to it. Notice that we have used the second as the time unit to express this metric. Also, and like the above metric, we represent the average response time for each case, illustrating the performance obtained with different processor units allocated to Tomcat.

In fact, the obtained graphics regarding with this metric give us a similar information than the graphics about throughput. The next Figures 5, 6 and 7, show the server's average response time as a function of the number of simultaneous user sessions within the server when running with different number of processors. In every one of the workloads, it is possible to be seen that the response time increases, more or less exponentially, when the server is overloaded. In fact, the response time begins to grow in an undesirable way at the same point where the throughput begins to decline.

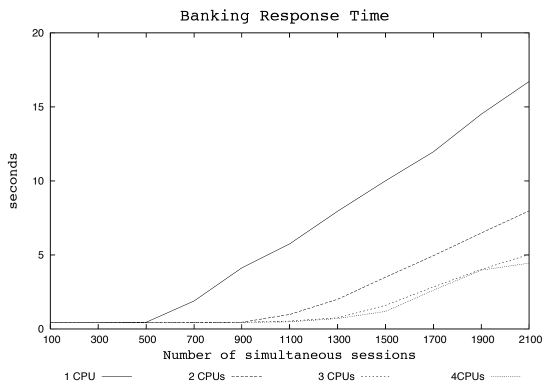


Figure 5: Banking response time when running with different number of processors

From here we can extract one decisive conclusion: if we want to give good sensations to users, we can not allow that the server's response time is inside the range of exponential growth. Accordingly, we have the option, for example, to implement an overload control strategy in order to avoid overloading the server. This strategy is usually based on connection differentiation and an admission control mechanism, as Guitart et. al did in [8] and [9].

What is interesting about this metric is that it has more influence with the clients, because the response time is perceived by them. This fact does not happen with the overall throughput of the server. Therefore, the response time offered is a significant metric to keep in mind when we are optimizing the performance of a given application server.

Also is noteworthy that response time is a very important point in SLA contracts. In fact, the common metrics of

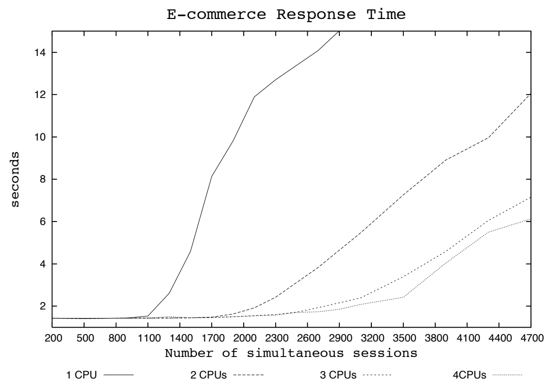


Figure 6: E-commerce response time when running with different number of processors

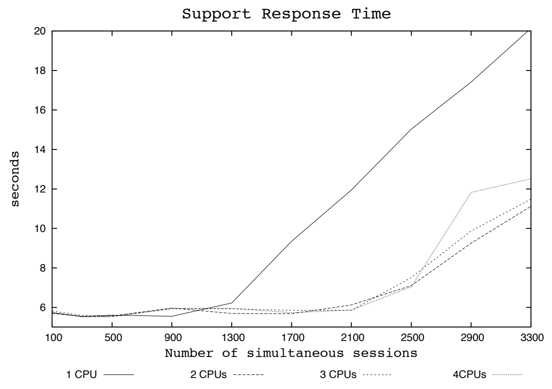


Figure 7: Support response time when running with different number of processors

these service contracts include both Average Speed to Answer (ASA) and Time Service Factor (TSF), which are another metrics extremely related with server's response time.

4.1.3 Quality of Service

It is one of the most notable changes in SPECweb2005 from its predecessors: the change from connection-based QoS to web page-based QoS. For all except the Support workload's "download" request, a time-based QoS is used. Specifically, QoS is based on the amount of time that elapses between a web page request and the receipt of the complete web page, including any image files. For the support workload's download state, a more appropriate byte rate-based QoS is applied. Also, the output results of any SPECweb2005 test has the called "Aggregate QoS compliance". It consists in calculate the total amount of requests (expressed as a percentage of the requests) that are within each of three different ranges:

good, tolerable and fail. These percentages give us a quantitative overview of what level of QoS has been fulfilled.

The used procedure by SPECweb2005 to calculate the implemented QoS criteria is the following: for each page requested by a load-generating thread of the benchmark, a timer is launched before sending the page request to the server. Subsequently, it is stopped as soon as the last byte of the response for that page is received. Then, valid responses will have their aggregate page response time checked against their respective QoS values of the workload, and the value for the corresponding QoS field (good time, tolerable time and fail time) will be incremented. At the end of the run, the prime client aggregates the run data from all the clients and determines whether the run met the benchmark QoS criteria. Table 2 clarifies the time limits (in milliseconds) of each of the considered ranges:

QoS criteria			
	Banking	E-commerce	Support
Good time	< 2000	< 3000	< 3000
Tolerable time	< 4000	< 5000	< 5000
Fail time	> 4000	> 5000	> 5000

Table 2: The time limits (in milliseconds) of every QoS range of SPECweb2005

With the obtained results from the tests, we have been able to represent the following nine graphs (Figures 8, 9 and 10) that illustrate the percentage of users requests that are within the already commented bounds. Once again, we present the results one by one for each workload. Specifically, we present the graph corresponding to the amount in each hundred requests that have been answered within a 'good compliance'. In the same way, we display the percentage of those that have been replied in a 'tolerable compliance', and the requests percentage whose the response time needed by the server has surpassed the last boundary ('fail compliance').

These figures allow us to know the number of simultaneous user sessions that the server can respond if we want that a percentage of requests X are answered in less time than Y. Nowadays, this is crucial for the hosting companies since this type of fulfillment is, probably, the most important clause of SLA contracts.

4.2 Performance Bottlenecks of the Server for each Workload

Notice that, for a given number of processors, the performance of the server increases linearly with respect to the simultaneous sessions until a determined number of it hit the server. This points are known as knee points. This section presents a detailed analysis about the limitations of the server in front of the SPECweb2005's workloads. There are

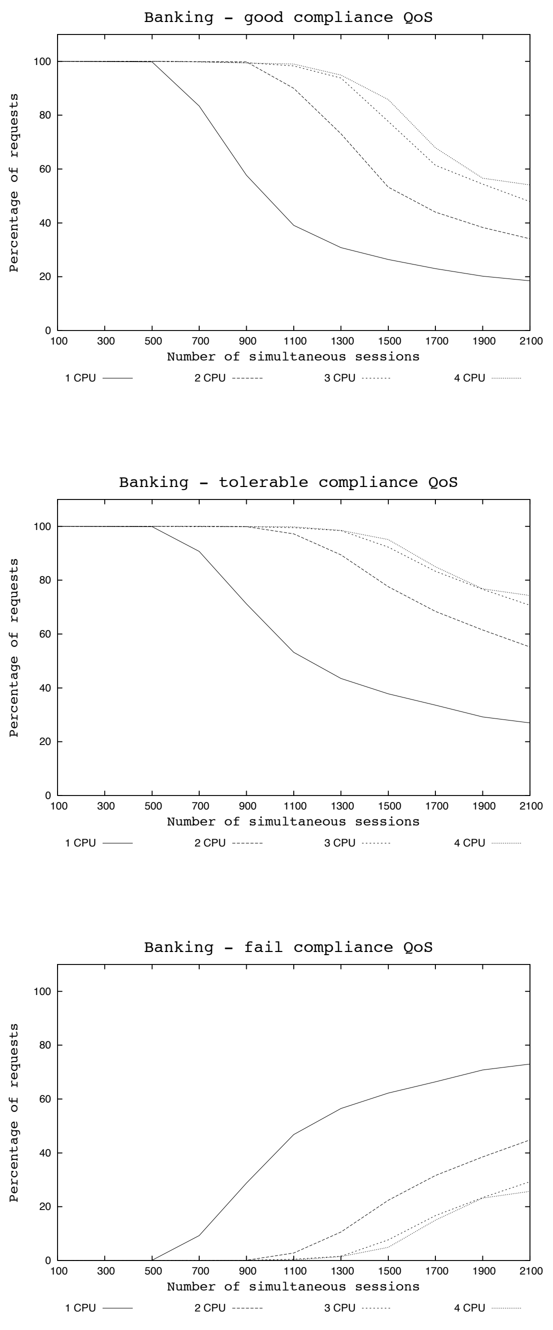
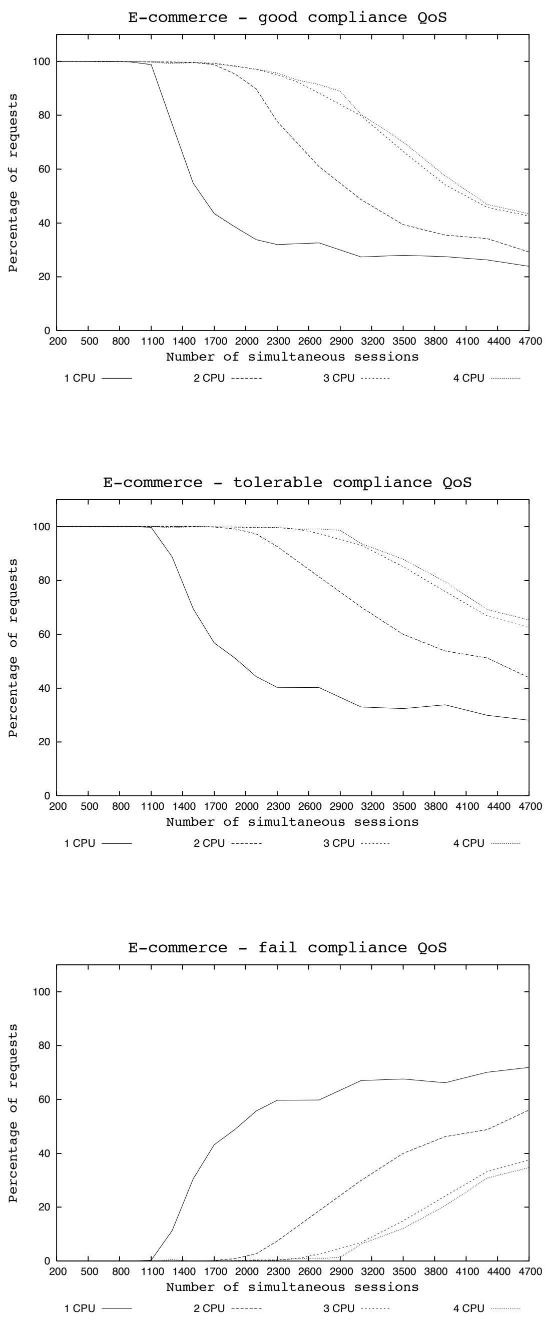


Figure 8: Banking QoS compliance when running with different number of processors



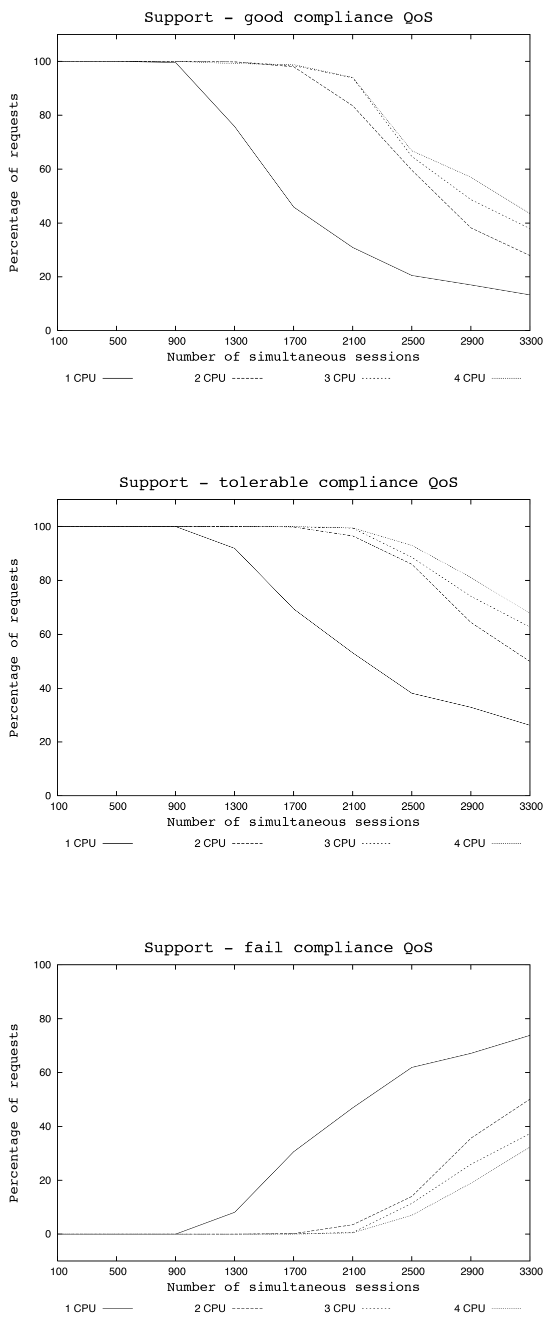


Figure 9: E-commerce QoS compliance when running with different number of processors

Figure 10: Support QoS compliance when running with different number of processors

several Unix monitoring tools that both gives us helpful information and allows us to monitor the performance of any application when it runs on top of any UNIX system. The goal of this analysis is to detect the causes of the server's performance degradation when running with different number of processors, that is, identify the bottleneck resource for each SPECweb2005 workload.

4.2.1 Tuning Complexity of Multithreaded Web Servers

As we already said, we opted for using a multithreaded Java web server. After performing the foregoing study of the server's scalability (presented in Section 4.1), we found that the configuration of the web server prevents the performance scalability of it. As you can see in Figures 2, 3 and 4, the throughput of the server does not scale according with the allocated processor units. We have checked the machine's resources utilization and in many cases we found that all the resources are underutilized. For this reason, we pose the possibility that the configuration of the multithreaded web server could be a potential performance bottleneck. Actually, we checked that configuring this type of web servers is a quite hard task. In addition, there are a lot of parameters that affect its performance. As Guitart et al. [11] [10] demonstrate, tuning these configuration parameters of application servers is a complex task to deal with because of the large complexity of this environment.

In fact, in [3] the authors show the complexity of optimally configuring this kind of servers for different workloads. They performed an exhaustive study around two key configuration parameters: the keep-alive timeout and the number of worker threads that are who process the incoming users requests. There are two main conclusions which we also have checked:

1. The optimal configuration of a multithreaded application server, that is, the ones with the server have better performance, is made up by one worker thread per user connection and an infinite timeout. Note that a typical and non-optimal configuration has some few hundreds of workers threads and a timeout of several seconds that the web server adapts appropriately with the input load. We found the afore-mentioned "optimal" configuration but it is clear that this is not an scalable setup and is highly dependent on the processed workload. In fact, we found that it is only viable when the server is not overloaded. For this reason, the data illustrated in this section has been obtained by modifying the server configuration for each test. Therefore, we configured the server with one thread per user connection until this setup was not feasible because of the overhead introduced by the creation and synchronization of them. From this point, we kept the setting with the possible maximum worker threads given the characteristics of both the analyzed machine and the Web application in question.
2. The hybrid application server architecture [5] (multithreaded and event-driven) is a feasible solution to simplify the web server tuning and, thus, we can obtain better performance.

We want to note that, although it is obvious that changing the configuration of the server depending on both the load and the Web application is not feasible in a real environment, we have used this idea in order to avoid that this become the performance bottleneck. Thus, we were able to reach our purpose: what is the underlaying resource bottleneck for each SPECweb2005 workload without having consideration about problems that depend on the type of server.

4.2.2 Banking Workload

This workload is mainly characterized by secure connections between the server and its clients. Because of the use of SSL protocol, which includes a handshaking method with high computational cost, is well known that the CPU is probably the resource bottleneck in secure scenarios. In order to check this assumption, we have measured the utilization percentage of this resource when the web server runs with one to four processors and with different simultaneous user sessions (Figure 11).

Indeed, from this Figure 11 we can conclude that the CPU is the resource bottleneck for the SPECweb2005 Banking workload. As we increase the number of concurrent users, the utilization of this resource grows until the maximum capacity of computation is reached. Then, the server can not scale its performance. Notice that the scalability of the server is not linear according with the number of processor units available to it. This fact is because with 3 and 4 CPUs we could not scale the optimal configuration of the server and, consequently, the overall performance does not scale. Really, the CPU utilization remains high in this two cases because of the threads management overhead.

4.2.3 E-commerce Workload

In the same manner as with the banking workload, we made the hypothesis that the processor unit should be the limiting resource in this case. Although this workload uses both non-secure and secure HTTP connections, we have considered the same assumption that in the Banking workload: the CPU should be the resource bottleneck. Again we have checked this assumption by monitoring the CPU utilization when the web server runs with different number of processor units and simultaneous user sessions (Figure 12).

Also we can affirm that the processor is the resource bottleneck for the SPECweb2005 E-commerce workload. As in the previous case, the scalability is not linear because from a certain point (3200 simultaneous user sessions) we can not scale the optimal server configuration.

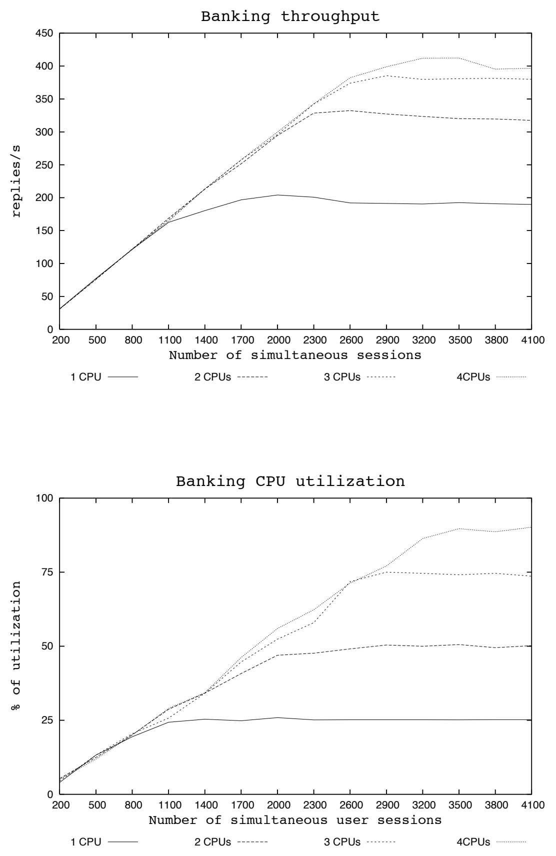


Figure 11: Web server throughput with ’optimal’ configuration and the average CPU utilization for banking workload

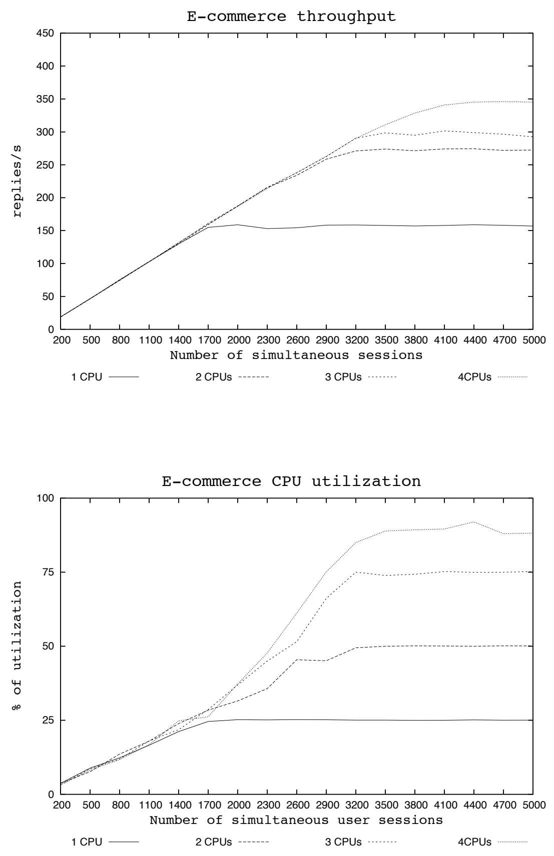


Figure 12: Web server throughput with ’optimal’ configuration and the average CPU utilization for e-commerce workload

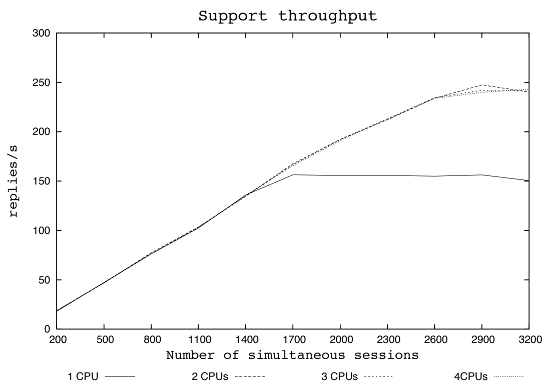
4.2.4 Support Workload

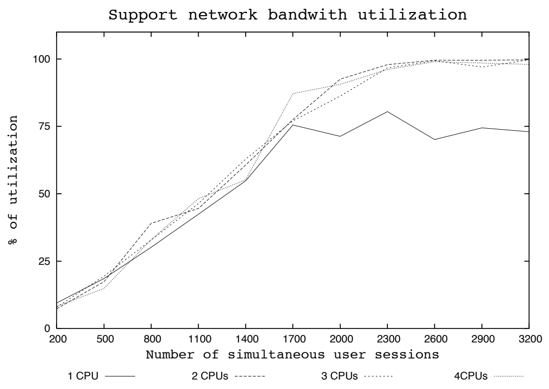
This third workload is quite different from the two previous: it was developed based on the characteristics seen in sites that were used to download, upgrade and fix related patches for computer support. Typically, these downloads are very large. For this reason, and taking into account the characteristics of this workload, we can predict that the possible resource bottlenecks could be the disk (i.e. I/O operations) or the network bandwidth. Once again, we have used some useful Unix monitoring tools to analyze and detect which are the special resource requirements of this workload. After a few tests analyzing the disk usage, we discarded this option and we centered in the network resource. Now we present the network bandwidth utilization (Figure 13) for each test case as we have done in the previous analysis.

As you can see, with two, three and four processor units, the utilization of the network bandwidth is around 100% and this fact limits the performance of the server. In the other hand, it is not the resource bottleneck with one processor unit available to the web server. For this reason, we have also checked the CPU utilization (Figure 13) and we can affirm that in the case of having only one CPU this has became the resource bottleneck.

In addition to the analysis of these resource bottlenecks, we have checked that many other resources are not limiting the server’s performance in any case:

- First of all, we checked that the JVM heap memory is never the resource bottleneck. Nevertheless, in the case of the Banking workload we had to increase the amount of JVM heap memory from 1GB to 2GB. In the others two workloads is enough with 1GB.
- Also, the overall memory of the server’s machine is not a limiting resource. In fact, it is obvious because the amount of memory that the machine has is proportionally larger than the computational resource.
- We could see that the BeSim system is not the bottleneck. If it were, there is an option to use multiple physical BeSim systems. We tried this option and it not resulted in improved performance.





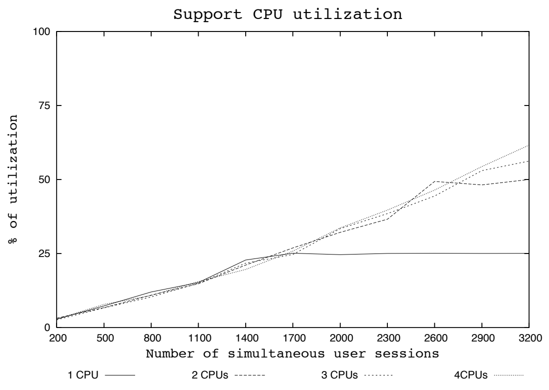


Figure 13: Web server throughput with 'optimal' configuration; average network bandwith utilization and average CPU utilization for support workload

- We tried to improve the overall performance by tuning several configure parameters of the Web server which emulates the Back-End Simulator. Once again, changing these configurations did not result in a remarkable performance improvement.
- Also we have examined that the clients (and also de prime client) are not overloaded during the performed tests.

5 Conclusions & Future Work

In this paper we have performed an exhaustive study of the performance of an application server in a multiprocessor environment with typical workloads of today. We get the measure of its performance using the next generation SPEC benchmark for evaluating the performance of World Wide Web servers: SPECweb2005. Firstly, we present the interesting metrics that the benchmark gives us, such as throughput, average response time and the offered quality of service. We represent the obtained results as a function of the simultaneous user sessions within the server when running with one to four number of processors. With it, we can know the behavior of the server in front of diverse workloads as well as their resources requirements. Furthermore, we use some Unix performance tools in order to better comprehend these critical resources demands which limit the scalability of the server's performance.

We endorse that a proper study, like the presented one, is primordial to know the performance of a server in front of characteristic workloads of nowadays. Moreover, it allows us to detect and be aware of the conditions that overload this server. This fact will be very useful in order to prevent this undesirable situation.

As a future work we are considering two different research directions: one that works in the same environment and the other that contemplates a different context. The first one would be focused on apply some ideas to the existing work in order to host and offer, at the same time, different services while maintaining a given Service Level Agreement for each one of them. In addition, we are considering to perform a similar evaluation around the performance scalability of the server, like the presented in this work, but using the hybrid application server architecture. On the other hand, we have the possibility to do the same work in other environments which let us to add / free computational resources depending on the requested load. Thus, *Cloud* are a suitable environment for this purpose. The work would go through the use of resources outsourcing with the aim to improve the server's scalability, as well as the availability and fault tolerance of the same.

Furthermore, we want to use some analysis tool in order to extract all the possible execution information from the Linux Kernel of the System Under Test, that is, trace and represent what are doing the worker threads of the application server. For example, we want to obtain information about the different thread states, in which CPU they run, the synchronization between them, etc. One possible tool which allows us to extract this desired information is *IBM Toolkit for Data Collection and Visual Analysis for Multi-Core Systems* [13]. In fact, it targets many common performance problems pertaining to Java applications running on multi-core or multi-processing platforms.

References

[1] C. Amza, E. Cecchet, A. Chanda, A. Cox, S. Elnikety, R. Gil, J. Marguerite, K. Rajamani, and W. Zwaenepoel. Bottleneck Characterization of Dynamic Web Site Benchmarks. In *Third IBM CAS Conference*, 2002.

[2] Apache Tomcat. `http://tomcat.apache.org`.

[3] V. Beltran, J. Torres, E. Ayguade, and S. Barcelona. Understanding Tuning Complexity in Multithreaded and Hybrid Web Servers.

[4] A. Bosque, P. Ibañez, V. Viñals, P. Stenström, and J. Llabería. Characterization of Apache web server with Specweb2005. *Proceedings of the 2007 workshop on Memory performance: Dealing with Applications, systems and architecture*, pages 65–72, 2007.

[5] D. Carrera, V. Beltran, J. Torres, and E. Ayguade. A Hybrid Web Server Architecture for e-Commerce Applications. In *ICPADS'05: Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS'05)*.

[6] D. Carrera, J. Guitart, J. Torres, E. Ayguade, and J. Labarta. Complete instrumentation requirements for performance analysis of Web based technologies. In *Performance Analysis of Systems and Software, 2003. ISPASS. 2003 IEEE International Symposium on*, pages 166–175, 2003.

[7] J. Guitart, V. Beltran, D. Carrera, J. Torres, and E. Ayguadé. Characterizing secure dynamic web applications scalability. *Parallel and Distributed Processing Symposium, International*, 1:108a, 2005.

[8] J. Guitart, D. Carrera, V. Beltran, J. Torres, and E. Ayguadé. Designing an overload control strategy for secure e-commerce applications. *Computer Networks*, 51(15):4492–4510, 2007.

[9] J. Guitart, D. Carrera, V. Beltran, J. Torres, and E. Ayguadé. Dynamic CPU provisioning for self-managed secure web applications in SMP hosting platforms. *Computer Networks*, 2008.

[10] J. Guitart, D. Carrera, J. Torres, E. Ayguadé, and J. Labarta. Successful Experiences Tuning Dynamic Web Applications using Fine-Grain Analysis. Technical report, Research Report UPC-DAC-2004-3/UPC-CEPBA-2004-2. January 2004.

[11] J. Guitart, D. Carrera, J. Torres, E. Ayguade, and J. Labarta. Tuning dynamic Web applications using fine-grain analysis. In *Parallel, Distributed and Network-Based Processing, 2005. PDP 2005. 13th Euromicro Conference on*, pages 84–91, 2005.

[12] R. Hariharan and N. Sun. Workload characterization of SPECweb2005. *SPEC Benchmark Workshop. SPEC*, 2006.

[13] IBM Data Collection and Visual Analysis for Multi-core Systems. `www.alphaworks.ibm.com/tech/dcva4j`.

[14] S. Mahadevan. *Performance Analysis of Offloading Application-layer Tasks to Network Processors*. PhD thesis, University of Massachusetts Amherst, 2007.

[15] SPECweb2005. `http://www.spec.org/web2005/`.

[16] SPECweb2005 Benchmark Design Document. `http://www.spec.org/web2005/docs/designdocument.html`.

[17] B. Veal and A. Foong. Performance scalability of a multi-core web server. In *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems*, pages 57–66. ACM New York, NY, USA, 2007.

[18] L. Wang, G. Von Laszewski, M. Kunze, and J. Tao. Cloud computing: A Perspective study. 2008.

[19] S. Warner and J. Worley. SPECweb2005 in the Real World: Using Internet Information Server (IIS) and PHP. In *2008 SPEC Benchmark Workshop*, 2008.

[20] A. Weiss. Computing in the clouds. 2007.