Performance Analysis of Web based Applications on Single and Multi Core Servers

Gitika Khare, Diptikant Pathy, Alpana Rajan, Alok Jain, Anil Rawat
Raja Ramanna Centre for Advanced Technology
Department of Atomic Energy
Indore, India
e-mail: gitika@rrcat.gov.in

***Abstract*— Performance is one of the most important business requirements for web based applications. The suitability of better platform for any web based application depends on various factors attributable to processor architecture, memory, local/ global network, input/output characteristics etc. Performance testing is an important part of any distributed or web application testing plan. Inclusion of performance estimates into planning and development cycles ensures that the application delivered to the user satisfies sustained performance even at high load, availability and scalability requirements. In this paper we have thrown light on performance comparison of web applications on single and multi core servers.**

The tests have been carried out on single processor single core server and also on dual processor quad core Xeon based server. The performance comparison was carried out based on response time, throughput (No. of hits v/s response time) and connection pooling v/s no connection pooling. It was found that performance of Java applications running on the multi-core servers is far better than those running on single core servers. Web applications on quad-core server give better performance, high availability and scalability.

Apache JMeter was used to load test the functional behaviour and measure performance based on throughput (No of hits v/s response time) for dynamic querying of data using Java servlets on Tomcat web server. Based on the analysis of performance of web server under varying load of no. of users and threads, it was found that applications on quad core server give more throughput and less response time as compared to single core server. Connection pooling increases the performance of web applications by reusing active database connections instead of creating a new connection with every request. Performance was compared by running the tests using Java servlets for connection pooling and also by acquiring connections directly from the JDBC driver without connection pooling. Based on the test results it was found that in case of quad core server, throughput i.e the number of requests served per second is much higher and reuse of connections from pool of connections showed significant improvement in application performance.

This paper describes the results of tests carried out for comparing performance (based on response time) between web applications running on single core server and on quad core server. Tests have also been carried out to analyze advantage of connection pooling over no connection pooling. The test results have been shown graphically.

***Keywords* — Performance Comparison, throughput, connection pooling, Apache JMeter**

I. INTRODUCTION

Enterprise applications are widely popular and their performance and scalability are becoming even more critical as the number of online transactions and CPU cores continue to increase over time. Enterprise applications are often multi-tier and usually consist of a middle tier where the applications are deployed and a database backend tier where data are stored and retrieved to perform business functions. While every tier plays an important role in performance of application, the role of middle tier is particularly important as it is often responsible for performing complex business logic.

Performance evaluation focuses on the analysis of dynamic behaviour of a system and the prediction of indices or measures such as its throughput, utilization or response time. In this paper, we report performance comparison results between single processor single core server HP Proliant ML 370 G4 having single core Xeon @ 3.0 GHz processor with 4 GB RAM and dual processor quad core server having Xeon Nehlam processor X5570 @ 3.4 GHz with 16 GB RAM.

Performance testing of an application is basically the process of understanding how the application and its operating environment behave at various user loads. In general, it is performed by simulating virtual users to determine / validate the scalability, availability, robustness, hardware & software resource utilization of the application thus ideally paving the way for effective capacity planning of the system for future usage. Performance testing is intended to determine the responsiveness, throughput, reliability, and/or scalability of a system under a given workload.

II. NEED FOR PERFORMANCE ANALYSIS

Intranet of Raja Ramanna Centre for Advanced Technology (RRCAT) nicknamed as 'RRCATInfonet' is a very popular portal for dissemination of information among the employees using two access schemes – open access and authenticated access. It provides seamless flow of information from various information management systems apart from providing many other services to its users.

The reliability and availability of RRCATInfonet server has become critical to the organization due to the deployment

and wide usage of online applications like Project Monitoring Software, Budget Monitoring Software, Indent Preparation Module, Purchase Information Module, software packages for Annual Report Evaluation and Electronic Assessment etc. These application software packages have wide user base, thus they have direct performance value and time window limit associated for any downtime and unavailability.

Performance analysis of RRCATInfonet server on single core single processor server vs. dual processor quad core server was carried out to analyze and address the needs to manage the availability, scalability, better performance and response of web server.

III. OBJECTIVE OF PERFORMANCE/ LOAD TESTING

Performance testing deals with continually changing technologies, large variety of user operating environments, collection and analysis of large volumes of data etc. Also there is architectural complexity associated with N-tier applications, which are made up of a large number of components residing on multiple machines. This increases the scope and complexity of testing. By testing the performance of an application early in the development lifecycle, organizations can find and eliminates bugs and bottlenecks, and expose potential scalability and performance glitches at substantially lower cost.

A load test emulates real-life interaction between users and server systems. A load test monitors and measures the Performance and reliability of servers and applications, Performance of an application under expected and peak user load conditions, Stability, Responsiveness and Throughput of the system in a realistic test environment. Performance testing is based on the following factors :

- In case of users – whether the users access the application inside or outside of the firewall, the connection speed that the users will use, will the client application cache the data etc.
- In case of servers - monitoring the hits, data sent/received to and from the web/application server/ database server, attributes of proxy server and what type of content will be blocked, attributes of firewall and permissions level, hardware (system/ network), how powerful is the CPU on the server and client machines, physical memory on each machine and network bandwidth.

IV. CHARACTERIZING PERFORMANCE AND QUALITY OF SERVICE (QOS) OF WEB APPLICATIONS

Most web environments are complex software systems. They are composed of firewalls, dynamic services and security techniques. Various options should be taken into account in order to carry out their performance analysis. This may depend on the kind of web application (distributed object-oriented vs service-oriented) and implementation technologies used in their development (e.g. Enterprise Java Beans or ActiveX controls) or design patterns used in their conception. All of these choices directly affect their performance as well as their QoS attributes.

The goal of load testing is to determine both the *performance* and *scalability* of a Web application. The most important performance parameters commonly used to measure web applications are:

- Response time : It is the time a system takes to respond to various types of requests. It is a function of load intensity, which can be measured in terms of arrival rates (such as request per second) or number of concurrent requests.
- Throughput : It is the rate at which a system or service can process requests, i.e. the request is completely processed by the server. It is normally measured in transaction per second (tps). The throughput is a function of the load assigned to a system and of the maximum capacity of the system to process work.
- Availability : It is the fraction of time that any system is operating and available to its users. The two main reasons for systems to be unavailable are failures and overloads.

Besides the above mentioned performance metrics, there are other qualities or properties that characterize QoS, such as:

- Security properties include the existence and type of authentication mechanisms, confidentiality and data integrity of messages exchanged; non-repudiation of requests or messages, and resilience to denial-of-service attacks.
- Reliability of a system is the probability that it works properly and continuously over a fixed period of time. When the time period during which the reliability is computed becomes very large, then reliability tends to the availability.
- Scalability of a system is considered to be good when its performance does not degrade significantly as the number of users, or equivalently, the load on the system increases.
- Extensibility is the property of a system to easily evolve to cope with new functional and performance requirements.

V. COMMON ATTRIBUTES AFFECTING PERFORMANCE OF APPLICATIONS

- Too Many Database Calls - This may be because more data is requested than actually required in the context of the current transaction, e.g. requesting all account information instead of that needed to display on the current screen. Multiple queries are executed to retrieve a certain set of data instead of using stored procedures to retrieve the data in one batch.
- Synchronization is necessary to protect shared data in an application. If excessively large code sequences are synchronized, under low load (on the local developers workstation) performance will not be a problem. But in high-load or production environment over synchronization results in severe performance and scalability problems.
- Memory Leaks - Managed runtime environments such as Java and .NET have the advantage of helping with

memory management by offering Garbage Collectors. A Garbage Collector, however, does not prevent memory leaks. Objects that are "forgotten" will stick around in memory and ultimately lead to a memory leak that may cause an OutOfMemoryException. It is important to release object references as soon as they are no longer needed.

- Using existing 3rd party libraries increases performance risks introduced by these components.
- Wasteful handling of resources such as memory, CPU, I/O or database connections results in lack of access to these resources and ultimately leads to performance and scalability issues. The database connections must only be used for the time period they are really needed, and then returned to the connection pool. Connections requested early and not released till the end leads to bottleneck situation.

VI. PERFORMANCE ANALYSIS METHODOLOGY

This paper examines the performance comparison of web based applications running on single processor single core server and dual processor quad core server. The performance comparison was carried out to determine the response time, throughput (No of hits v/s response time) and advantage of connection pooling over no connection pooling. There is a significant improvement in performance in case of quad core server.

System characteristics of the two setups are summarized in the following table :

System Characteristics		
Specification	***Single processor Single core server***	***Dual processor Quad core server***
Make and Model	HP Proliant ML 370 G4	Dell Power Edge R710
Processor	One Intel Xeon @ 3.0 GHz	Two quad core Xeon Nehlam Processors X5570 @ 3.4 GHz , 6.4 GT/s QPI
Memory	4 GB ECC DDR Unbuffered RAM	16 GB DDR-3 1333 MHz RAM
Hard drives	Two no. of 145.6 GB (10K rpm)	Five no. of 300 GB (15K rpm)
Operating System	Red Hat Enterprise Linux 4.2	Red Hat Enterprise Linux 4.2
Web Server	Apache Tomcat 6.0.18	Apache Tomcat 6.0.18
JDK	JDK 1.5	JDK 1.5
JVM Memory Size	512 MB	2048 MB

The testing configuration consists of Open Source Performance testing tool Apache JMeter, Tomcat 6.0.18 web container and Oracle 10g database instance. Tomcat and Oracle 10g database are running on separate severs connected over 1 Gbps switch.

Performance comparison was carried out based on the following parameters:

A. Throughput and Response Time

Apache JMeter tool was used to test the functional behaviour and measure performance based on throughput. Apache JMeter is an open source Java desktop application designed as a load testing tool for analyzing and measuring the performance of web applications. It can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types. It provides a graphical analysis of performance and tests the server/script/object behaviour under heavy concurrent load.

A JMeter test plan was created for dynamic querying of data using Java servlets on Tomcat web server. The performance was tested by varying the number of users and number of threads. It was used to simulate a load of 1000 users on the database server to analyze overall performance under different load types.

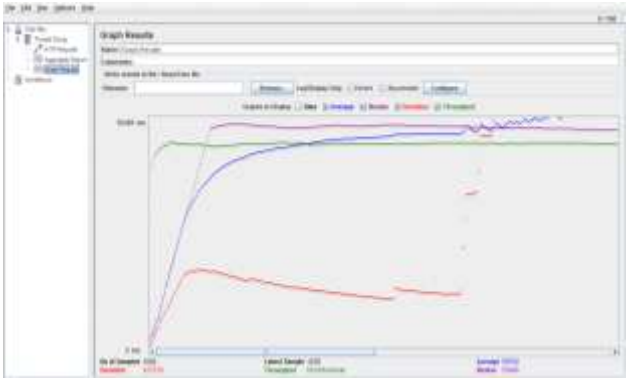

Figure 1. Throughput on Single Core Server

Figure 2. Throughput on Quad Core Server

The above graphs illustrate the test results with number of threads = 100, ramp-up time = 5 and loop count = 10. The results are plotted between No. of samples vs. response time in millisecs. The parameters displayed on the charts are throughput (green line), median (purple line), deviation (red line) and average (blue line).

- Throughput is the number of requests per minute the server has processed.
- Average is the total time running divided by number of requests sent to the server.
- Median is the number that represents the time, where half of server response time is lower than this number and half is higher.
- Deviation shows how much the server response time varies, a measure of degree of dispersion, or, in other words, how spread the data are.

Based on the graphical analysis of performance of web server, the throughput of 1000 concurrent requests on quad core server was 180.913 per minute as compared to 69.065 per minute on single core. This shows that throughput of dual processor quad core server is 3 times more than that of single core server for 1000 concurrent users. Applications running on quad core server give more throughput and less response time as compared to single core server.

B. Connection pooling vs. No Connection pooling

Connection pooling provides significant benefits in terms of application performance, concurrency and scalability. Connection pooling increases the performance of Web applications by reusing active database connections instead of creating a new connection with every request. It reduces database-related overhead and enables an application to use a connection from a pool of connections that need not be re-established for each use.

Java servlets are written to use either pooled or non pooled database connections depending on the query string passed in its URL. The servlet was dynamically instructed by the load tester to use (or not use) connection pooling in order to compare throughput in both modes. The servlet created pooled connections using a Tomcat JDBC connection pool and non pooled connections directly from Oracle's thin JDBC driver.

A JMeter test plan was created to continuously run the servlet in pooled and non pooled connection modes with number of threads = 100, ramp-up time = 5 and loop count = 10. The total elapsed time for each run was measured with connection pooling and no connection pooling.

When connection pooling is used, the first connection takes the longest time because a new physical connection had to be created and the pool manager had to be initialized. Once the connection existed, the physical connection was placed in the pool and was reused to create a handle for each subsequent connection. In case of connection pooling, all subsequent connections were reused because they were used for the same user and pool cleanup had not occurred.

On comparing the pooling results at each iteration checkpoint with the non-pooling results, we see that Connection pooling provides a significant improvement on performance by reusing connections rather than creating a new connection for each connection request.

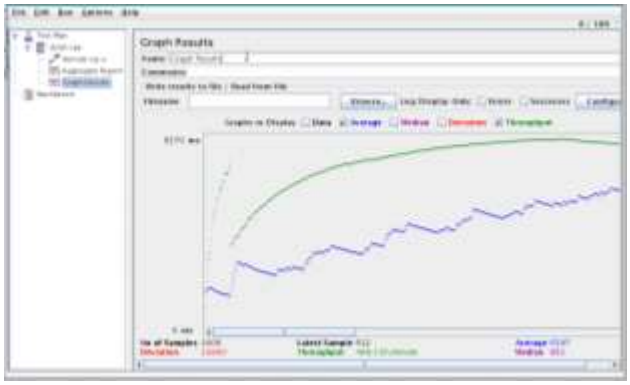

Figure 3. Pooled connections throughput

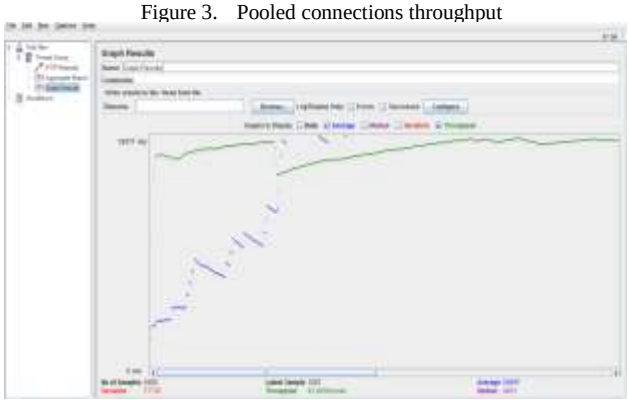

Figure 4. Non pooled connections throughput

The results are plotted between No. of samples vs. response time in millisecs. Based on the above JMeter results graph it was observed that the throughput of 1000 concurrent requests using pooled connections was 468.315 per minute as compared to 83.969 per minute using non pooled connections.

This very clearly indicated that throughput using pooled connections is almost six times faster than non pooled connections and web applications realize significant performance improvements by using connection pooling over no connection pooling.

VII. CONCLUSION

Performance analysis is very important as it can be used to study various parameters such as server response time, maximum simultaneous users, duration of each service invocation, interval and wait times, session usage, session expiry, and memory usage.

Performance testing helps to identify bottlenecks in a system, establish a baseline for future testing, support a performance tuning effort, and determine compliance with performance goals and requirements. Including performance testing early in the development life cycle can add significant value to the project. Also performance tuning can reduce the amount of resources being used and/or further improve system performance.

Based on the performance comparison carried out during our study, it was found that applications on quad core server give more throughput and faster response time as compared to

single core server. Connection pooling increases the performance of web applications by reusing active database connections. Therefore deployment of RRCATInfonet applications on quad core server will definitely address the needs for high availability, scalability, better performance and response of the web server to a great extent.

REFERENCES

[1] Java - http://www.java.com
[2] Apache Tomcat – http://tomcat.apache.org/
[3] Apache JMeter - http://jakarta.apache.org/jmeter/
[4] Oracle – http://www.oracle.com
[5] Open source performance testing tools - http://www.opensourcetesting.org/performance.php