Massey University

J2EE Application for Clustered Servers

--- Focus on balancing workloads among clustered servers

A thesis presented in partial fulfillment of the requirements for the degree of

Master of Information Science

in

Computer Science

at Massey University, Albany,

New Zealand.

Supervised By: **Dr Chris Messom**

Xi Chen

[2006]

Acknowledgment

Special thanks go to my supervisor **Dr. Chris Messom** for his guidance, enthusiasm and technical support during the year. He impressed me with his broad knowledge, background and attitude on every issue. He teaches me not only the latest technologies but also solving problem skills. Thanks a lot.

I also would like to thank the system administrator James Chai for his support to make my project run smoothly, e.g. increasing shared memory size and resetting network routes etc.

Thanks also go to all the developers who have contributed to open source software.

Meanwhile, I appreciate my parents who gave me full support and took care of my sons during my study.

Abstract

J2EE has become a de facto platform for developing enterprise applications not only by its standard based methodology but also by reducing the cost and complexity of developing multi-tier enterprise applications. J2EE based application servers keep business logic separate from the front-end applications (client-side) and back-end database servers. The standardized components and containers simplify J2EE application design. The containers automatically manage the fundamental system level services for its components, which enable the components design to focus on the business requirement and business logic.

This study applies the latest J2EE technologies to configure an online benchmark enterprise application – MGProject. The application focuses on three types of components design including Servlet, entity bean and session bean. Servlets run on the web server Tomcat, EJB components, session beans and entity beans run on the application server JBoss and the database runs on the database server PostgreSQL. This benchmark application is used for testing the performance of clustered JBoss due to various load-balancing policies applied at the EJB level.

This research also focuses on studying the various load-balancing policies effect on the performance of clustered JBoss. As well as the four built-in load-balancing policies i.e. *FirstAvailable, FirstAvailableIdenticalAllProxies, RandomRobin* and *RoundRobin*, the study also extend the JBoss *LoadbalancePolicy* interface to design two dynamic load-balancing policies. They are *dynamic* and *dynamic weight-based* load-balancing policies.

The purpose of dynamic load-balancing policies design is to ensure minimal response time and obtain better performance by dispatching incoming requests to the appropriate server. However, a more accurate policy usually means more communications and calculations, which give an extra burden to a heavily loaded application server that can lead to drops in the performance.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Summary: The chapter describes some brief reasons why the J2EE platform was selected. And then presents the purpose of the study and aims of the study. Finally, the overall structure of the thesis and main contents in each chapter are listed.

1.1 Why Choose J2EE

Since the 1990s, middleware technology i.e. middle-tier software has developed to simplify distributed assembly of components, which is a collection of services including managing communication, security and threads etc. to enable multiple processes on different working machines to interact across the network. For a distributed enterprise application, usually we use middleware to connect separate applications such as linking between web applications to a database system, the technique provides an abstraction capability to simplify the construction of a distributed enterprise system and allows the application developers to only focus on business logic [52]. The most attractive middleware today are CORBA, J2EE and .NET. All of them are deployed as standard components / objects. But CORBA (Common Object Request Broker) provides only middleware techniques to model a standard and consistent component architecture framework such as clients' request for services from servers via well-defined interfaces across network. J2EE and .NET are the two most popular software development platforms to design server-side enterprise applications.

Both J2EE and .NET are emerging and competing platforms that contribute to simplifying writing enterprise applications. .NET is restricted to MS Windows-based platforms although some limited open source multi-platform .NET systems are in development, such as the Mono Project [16]. This research will utilize Massey Beowulf Clustered computers that are running the Linux operating system. In addition, the advantage of J2EE is that it is programmed in Java that can be deployed cross-platform. Moreover, this research extends a previous study, Zhou's Master's research that used SUN ECPerf --- a benchmark J2EE application as the testing application to study "A scalable application server on Beowulf Cluster" [1]. As a result, the J2EE platform is a natural choice for the study.

1.2 The Purpose of the Study

This research addresses techniques that support the development of distributed enterprise applications particularly its non-functional requirements including security, scalability, fault tolerance and load-balancing etc but focuses specifically on load-balancing policies. Due to the dramatically dropping hardware price, using more than one server has become affordable for more and more companies, including most small companies. Moreover, the Internet is providing a potential e-market for enterprises since the network has effectively shortened distances for international or national trade. The quality of e-commerce services, such as securing customer information, ensuring minimal online waiting time to increase returning customers has been a priority. Improving the performance of servers, particularly clustered servers are now being addressed. Load balancing is one of the key technologies behind clustering that affects the performance of the clustered servers when the servers face heavy loads.

The goal of load balancing among clustered servers is to ensure minimal response time and obtain better performance by dispatching incoming requests to the appropriate server. When Zhou [1] studied the scalability of Beowulf clustered servers using JBoss, he found the different load-balancing policies could get different performance results. His study indicated that a better scalability result could be achieved by using First Available Load-Balancing policy than using default Round Robin Load-balancing policy, but both of them became a bottleneck under heavy workloads [1]. In order to improve this situation, this research will focus on studying the load-balancing mechanism. The study will extend the JBoss *LoadbalancePolicy* interface to design a more appropriate load-balancing policy --- dynamic and dynamic weight-based load-balancing policies.

In order to test the performance of clustered servers using various load-balancing policies, this study also includes building a benchmark J2EE application. It consists of:

- Client side: client view of dynamically generated HTML pages using server side technology Servlets that is implemented on Tomcat, a servlet container.
- Server-side: refers to both web server and application server. The web server (Tomcat) focuses on user interface design via web component Servlets and the application server (JBoss) which handles business processes via designing J2EE enterprise Java beans components entity beans and session beans
- Database: used for store business persistent data that is implemented on the open

source database application PostgreSQL

1.3 How the Thesis Is Organized

The thesis focuses on the J2EE application and load-balancing policies design. The entire thesis will be focused on these two main topics and organized into eight chapters. The overall structure is as follows:

- ***Chapter 1*** describes some brief reasons why the J2EE platform was selected. And then presents the purpose of the study and aims of the study. Finally, the overall structure of the thesis and main contents in each chapter are listed.
- ***Chapter 2*** overviews the knowledge background and terminology needed to understand this thesis. J2EE is introduced including the architecture of the J2EE platform, containers and APIs and the structure of deployment of J2EE applications. Secondly the well-known benchmark J2EE application – ECperf design is introduced. And finally the n-tier clustered distributed system and common load-balancing policies in use are presented.
- ***Chapter 3*** covers the hardware system for this study - the clustered systems built in Massey University Albany campus. The detail of the open source software chosen for this study including J2EE application design IDE, web server Tomcat, application server JBoss, database PostgreSQL and performance test tool Jmeter are discussed.
- ***Chapter 4*** details the entire benchmark J2EE application – MGProject design, including database, EJB and web components design. A brief introduction of the EJB and Servlet design, including the life cycle of the EJBs and Servlets, including the main concerns on designing these components is given. In addition, example coding of the design to explain the components are given. Moreover, how the web component and EJB components interact with each other in the application and how to use XDoclet in the NetBeans IDE to generate JBoss specific deployment descriptors is presented.
- ***Chapter 5*** covers the detailed design of the dynamic and dynamic weight-based load-balancing policies for clustered JBoss. The policies will extend JBoss *LoadBalancePolicy* interface and utilizes EJB timer service and JBoss T*reeCache*. The *LoadBalancePolicy* interface exposes the cluster members' information. The EJB timer service performs tasks in a regular period and JBoss

TreeCache provides shared storage for all timer session beans located in different machines. In addition, the chapter also lists the pseudo code design.

- ***Chapter 6*** The test plan in this chapter has two meanings. One refers to establishing a test plan in Jmeter. The plan will simulate a large number of independent clients performing online actions to interact with back-end servers. Another means to plan a test procedure for testing the performance of JBoss under various load-balancing policies. This plan should provide a sequence of testing steps to achieve the testing goal. Meanwhile, techniques to ensure the benchmark application runs as smoothly as possible under the very heavy loads are discussed.
- ***Chapter 7*** presents the details of the tests implementation on the Sisters cluster. The main tests are classified into two kinds – scalability tests and load-balancing policy comparison tests. The scalability refers to a single JBoss and clustering two or more JBoss performance test. The load-balancing policy tests will apply various load-balancing policies on the EJB level to identify if each of them affects the final performance of clustering JBoss.
- ***Chapter 8*** gives the final conclusions of the thesis. The conclusions include a critique of the J2EE application design and the effect of the various load-balancing policies on the system performance. Finally, the chapter also presents potential further work.