

MASTER'S THESIS

Mechanisms for provisioning quality of service in web servers

Chan, Ka Ho

Date of Award:
2008

[Link to publication](#)

General rights

Copyright and intellectual property rights for the publications made accessible in HKBU Scholars are retained by the authors and/or other copyright owners. In addition to the restrictions prescribed by the Copyright Ordinance of Hong Kong, all users and readers must also observe the following terms of use:

- Users may download and print one copy of any publication from HKBU Scholars for the purpose of private study or research
- Users cannot further distribute the material or use it for any profit-making activity or commercial gain
- To share publications in HKBU Scholars with others, users are welcome to freely distribute the permanent URL assigned to the publication

Mechanisms for Provisioning Quality of Service in
Web Servers

CHAN Ka Ho

A thesis submitted in partial fulfillment of the requirements
for the degree of
Master of Philosophy

Principal Supervisor: Dr. Xiaowen Chu

Hong Kong Baptist University

May 2008

Abstract

Since the number of web users is ever-increasing and the types of web services keep on growing, the ways to boost web server performance and server capacity are very important in nowadays commercial world. As the hardware cost keeps dropping, users can purchase a computer server with high computation power at a low price. Therefore, the price limitation of the server machine is eased out. It is obvious that even the most powerful server may still fail to provide timely services for every user. There are two ways to solve the problem: (1) the introduction of service differentiation techniques to provide better services for the premium users; (2) the application of cluster-based web servers.

In order to provide service differentiation, Proportional-Integral (PI) controllers and fuzzy controllers are attractive candidates. However, PI controllers require an accurate model and assume a linear characteristic. With such limitations, PI controllers cannot work well for the web server application. On the other hand, fuzzy controllers, based on complicated fuzzy sets, can work well under bursty environment. However, there are large amount of parameters needed to be tuned which are very difficult to make initial approximate adjustment since there are no cookery book to do it. Moreover, these parameters settings usually suffer from nonzero steady state error. In order to combine the advantages of the PI controllers and the fuzzy controllers, we have proposed the fuzzy PI controller. There are only simple fuzzy set definitions. With the fuzzy controllers' characteristics, the settling time is highly reduced. At the same time, it performs well in steady state just like the PI controllers.

Some people may argue that employing a Content-Delivery Network (CDN) can

be a good way to handle the loading problem of the web servers. However, large amount of investment is required. For most of the enterprises, this is not a reasonable and applicable method. Therefore, we would like to employ our proposed fuzzy PI controllers on top of the cluster-based web servers. In order to handle the system uncertainties and load balancing problem, we would like to dispatch the connections in a weighted round robin manner with a preemptive mechanism. It is manifest that the cluster environment is more chaotic. Therefore, a fuzzy PI controller with preemptive nature is a good way to handle the uncertainties. Thus, better services can be provided for the premium users. At the same time, the total number of clients is governed by the number of the back-end server nodes. In short, the fuzzy PI controllers can be scaled up and the investment is only limited which are beneficial for small to medium scale enterprises.

On the other hand, the computation power of the server machines is increasing conspicuously. However, the server capacity has not been enhanced by much. A number of researchers have proposed the application of high performance computation methods to boost the server capacity. However, one of the well known web servers, Apache, cannot support large amount of concurrent clients. The advantages of the Apache web server application are (1) the large amount of built-in modules; (2) the ease of building new supporting modules. In order to support high performance computing by the Apache web server, a high performance thin client proxy server, which serves as the middle-tiered between the clients and the Apache web server, is developed. The proxy server can support a large amount of client connections. Therefore, the overall server capacity can be enhanced.

Table of Contents

Declaration	i
Abstract	ii
Acknowledgements	iv
Table of Contents	v
List of Tables	viii
List of Figures	xi
1 Introduction	1
1.1 Background of Web Servers	2
1.1.1 Apache Web Servers	2
1.1.2 Latency of Downloading Web Pages	3
1.1.3 Quality of Services (QoS)	5
1.1.4 Enhancing Server Capacity	6
1.2 Architecture	7
1.2.1 Standalone Web Servers	7
1.2.2 Cluster-Based Web Servers	7
1.2.3 Server Capacity Improvement	9
1.3 Contributions of the Thesis	9
1.4 Outline of the Thesis	10

2	Fuzzy PI Controller	11
2.1	Related Works	11
2.1.1	Semantics of QoS	11
2.1.2	Classic Proportional Integral (PI) Controllers	13
2.1.3	Fuzzy Controllers	13
2.2	Architecture	14
2.3	Design of Sampling Time	15
2.4	Design of the Proposed Fuzzy PI Controller	16
2.4.1	Computation of Proportional and Integral Gains	18
2.4.2	Design of Fuzzy PI Controller	20
2.5	Cluster-Based Web Controllers	24
2.5.1	Related Works	24
2.5.2	Dispatching Controller Design	26
2.6	Performance Evaluation	31
2.6.1	Workload Generator	31
2.6.2	Performance Metrics	33
2.6.3	Experimental Results on standalone server	35
2.6.4	Experimental Results on cluster-based server	40
2.7	Chapter Summary	44
3	Preemptive Controllers	45
3.1	Related Works	45
3.2	Design and Implementation of Preemptive Mechanism	47
3.2.1	Difficulties of Choosing Preemptive Candidates	48
3.2.2	Design of the Preemptive Mechanism	49
3.3	Design of Sampling Time	53
3.4	Performance Evaluation	54
3.4.1	Cluster-Based Web Controllers	54
3.4.2	Standalone Web Controllers	60
3.5	Chapter Summary	69

4	Scaling Up Web Servers	70
4.1	Related Works	70
4.2	Architectures	71
4.3	Experimental Results	73
4.3.1	Requests Generator	73
4.3.2	Performance Metrics	74
4.3.3	Web Server Performance Evaluation	75
4.3.4	Enhanced Web Server Performance Evaluation	79
4.4	Chapter Summary	82
5	Conclusion and Future Work	83
	Appendix	88
	Curriculum Vitae	104