Knowledge-Based Self- Management of Apache Web Servers

Kemal A. Delic, Jeff A. Riley, Claudio Bartolini
Hewlett-Packard Company
France/ Australia/USA
Technology Services Group – HP Labs
name.surname@hp.com

Adnan Salihbegovic
Faculty of EE
University of Sarajevo
asalihbegovic@etf.unsa.ba

***Abstract* - Today, more than 70% of all Internet sites are using Apache web servers. Furthermore, some indications are given that 40% of all web delays are due to badly tuned web servers. Consequently, we consider the tuning of web-servers to be a very relevant, practical problem. In this paper we outline an experimental set-up (graduate thesis) aiming at self-tuning of Apache web servers. First, we describe the problem domain, and then we give an overall architecture in the second section. Short analysis of related work and research is given in the third section. We conclude with projections and expectations from our research and experiments.**

Index Terms – web server, self-management, knowledge-based, self-tuning, automation

I. Introduction

With more than 70% of web sites on the Internet using the Apache web server [1], it is the most popular and widely used web server on the Internet. Furthermore, Apache is a very stable application and thus failures are very rare (indicative mean-time-between-failure (MTBF) rates of five years). Consequently we will focus in this paper on the knowledge-based self-management, including constant, dynamic tuning and re-tuning, of Apache web servers.

Since it is estimated that approximately 40% of all delays on the World Wide Web are attributable to failed or badly tuned web servers [2] and in many cases web servers are left to run the standard "out-of-the-box" configuration, automated tuning and error recovery in web servers represents an important problem to be addressed. A lot of work has been done on control systems theory approaches to tuning and maintenance of Quality of Service (QoS) [3], but we are not aware of any work which applies explicitly a knowledge-based approach to two distinct server operation modes:

- constant, dynamic tuning and
- recovery from fatal errors

System administrators are typically responsible for the manual tuning of web servers. These are 'skill-heavy', error-prone and highly repetitive operations. As such, they are the suitable candidates for automation. We would expect cost reduction, improved tuning quality and better reaction and reduced latency times to result from proper knowledge-based automation.

In this paper we outline the project for a potential student thesis which will both address an important practical problem and make a theoretical contribution to the field. We assume that experimental results will prove or disprove some of our initial assumptions, and will demonstrate the feasibility of a knowledge-based approach to web server tuning.

We start by describing the broader usage context and positioning the web server within the typical, generic enterprise application stack. Then, we briefly sketch an overall knowledge-based architecture and follow with decomposition of the knowledge-based tuning system into three layers: reflective, instinctive and learning layers [4]. We indicate the principal sources of knowledge as the key ingredients of this approach and discuss briefly the knowledge-capturing technology and knowledge-maintenance process. After we describe some previous work in this domain we conclude with a description of the experimental set-up which will demonstrate the soundness of our approach. We hope that elucidated principles and learned lessons are reusable across the entire enterprise application stack.

2. Problem Domain Outlined

Tuning of web servers is of prime importance for the generic enterprise applications (stacks consisting of web, application and database servers) as it impacts directly on the perception and end-user experiences and contributes to the performances as the quality-of-service (QoS). We could think about large-scale consumer web site or highly visited web news site as the typical enterprise application.

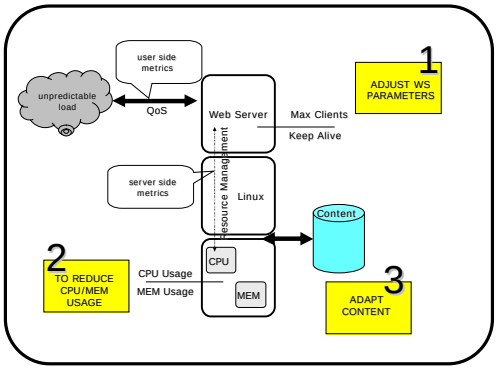

Fig. 1 Web Server Tuning Set-up

Generally speaking, performances are expressed either as the client-side metrics (response time) or server side metrics (CPU and memory usage). Tuning chores are executed daily by the system administrators who typically tune server configurations aiming to serve maximal number of users with minimal usage of CPU cycles and memory space. They deal with unpredictable behavior of web site visitors and users and constant changes of the site content using their practical, hands-on experiences. We aim at the automation of these tuning chores via knowledge-based system. Tuning heuristics can be obtained either from system designers, system operators or extracted automatically from the web-server logs. We would like to combine those various sources of knowledge into coherent tuning control strategy [5], which is also the principal contribution of this paper. Others have applied various analytical models for this, but we are not aware of any work being identical to our proposal suggesting combination of several analytical models.

3. SOLUTION: KNOWLEDGE-BASED TUNING ARCHITECTURE

To illustrate possible uses of human knowledge in web-server tuning, we evoke the example of the CNN web site bringing breaking news. We observe a very sharp increase in the number of users and accesses, and know that human operators react by compressing pictures on web sites and reduce the number of URLs available. Formalized, this situation could be described by the following (stylized) rule:

If access_rate=very_high
Then CompressImages And/Or RemoveURLs

Early knowledge-based systems were typically rule-based and as such worked well but have never reached maturity of being robust, easy to maintain or grow smoothly. Also, they have been too simple to be able to implement sophisticated control strategies in the closed-control-loop systems and set-ups. Later, neural nets have been seen as better models for capturing of tuning heuristics while lacking explanation facilities and being totally non-transparent. Decision Trees have been further used to emulate decision making by humans in the feedback control systems.

Within the scope of self-management we deal with configuration, recovery, optimization and tuning. This decomposition is the first step towards creating a taxonomy of problems hinting at possible sources of knowledge. Knowledge-based systems acquire knowledge in one of two modes:

- on-line, real-time, unsupervised learning, or
- off-line, supervised learning.

In both cases (on-line unsupervised learning and off-line supervised learning), the sources of knowledge are either human(s) – tacit knowledge – or some trace collections from which we can extract knowledge automatically.

A variety of machine learning techniques can be applied in this domain. For example, as shown in the illustration above, a programmer's knowledge can be represented by a rule-based expert system [6], an IT operator's knowledge by decision trees [7] and knowledge acquired from logs by neural networks [8]. It is important to note that in each case the knowledge could be extracted and represented as rule-sets.

Inspired by thinking of Sloman [4] we postulate here that we need (at least) three layers of control being hierarchically embedded: 'reflective' control, 'intuitive' control and 'learning' control (Fig. 2). The first closest loop should be implemented as the rule based system, the next, intuitive loop will be implemented as decision-tree and outer loop is embodied in neural network – as the learning layer.

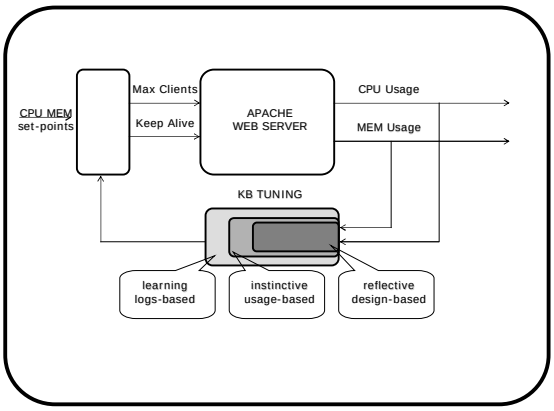

Fig. 2 Three Layers Control System

We further propose that the designer knowledge will be best captured in a rule-based system. System administrator knowledge is best expressed with decision trees, as a natural extension of troubleshooting diagrams. Finally, since system logs contain historical traces of web server operations, we propose that the learning layer employ neural nets to capture this knowledge.

Harvesting knowledge from the trustable sources and ensuring its quality and renewal at bearable cost represents an important engineering challenge. We will give some

cost and benefit estimates.We will review the work of others and point at the key findings from the key researchers in the field.

We will point at the importance of our hybrid approach (combining knowledge from different sources and utilizing machine learning techniques appropriate to the data sources) and will address the following research question:

What can we learn from
(1) error logs only
(2) access logs only
(3) both error and access logs combined

in order to develop the knowledge necessary for a web server to self-manage?

The theoretical contribution of this work should deal with this control set-up, will detail how those three layers function coherently and what would be theoretical advantage of such architecture. We will demonstrate the use of neural network technology for the automatic knowledge extraction from server logs.

4. RELATED WORK

We have studied the work of researchers from industry, academic research and student projects in order to define appropriately this project. Diao et al. describe an agent-based tuning system [9] which was later expanded into a complete toolkit [10] that can be used for the creation of various autonomic systems. These reports describe an overall architecture and give enough design details that their experiments can be repeated and results checked.

Then, we have used academic references [11, 12] to educate ourselves about control theoretic aspects of our approach and clarify our experiment choices. Along these lines we studied three student papers [13, 14, 15] addressing the same or similar domain with slightly different angle. Those helped us to position better our approach, set-up experiment plans and enable (eventual) comparisons.

It was our feeling that the choice of load generators emulating real-world traffic loads/patterns as much as possible was critical for sound judgment of experimental results. To this purpose, we explored several reports describing web server load generators [16, 17, 18].

5. EXPECTATIONS AND FUTURE WORK

We expect that previous work in this domain should be carefully studied and that certain experiments reported in related papers will be repeated and double-checked. This will lead to valuable hands-on experience and the ability to devise some innovative solutions in this domain.

We plan to set-up a test environment which will include 4,5 load generating computers (Linux based machines) connected via 100 MBits/sec Ethernet and exercising one machine running Apache web server on Linux (appropriate versions chosen). The student conducting the work should have sufficient skills to modify Apache source code in order to set-up experiments properly. In particular, it would be of ultimate importance to gather and analyze data sets from the real-world environment. This will clarify choices about load generators aiming to emulate real-world traffic handled by test web server.

Furthermore, for chosen data sets, complete and accurate logs (access, error) should be provided as they represent the basic source of data for the machine learning algorithm(s). This will include preparation, cleansing and checks of the integrity of the logs.

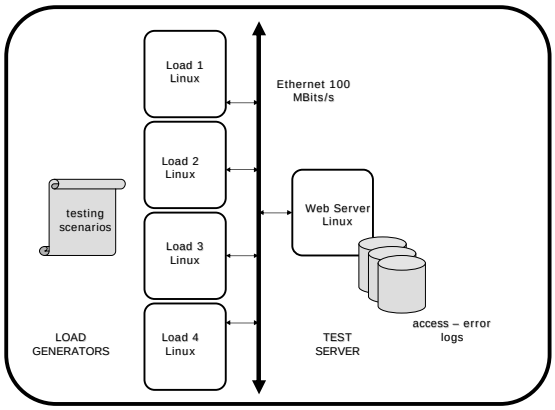

Fig. 3 Experimental Set-up

The ultimate result is seen as the web server's 'self-tuning' or the ability of the server to deal not only with unpredictable workloads but also with constant changes of published content. Principles and insights from control systems theory should be used to check for stability and optimality of knowledge-based solution. Experiments could be devised demonstrating eventual advantages of our approach. The practicality of the results should be judged against eventual industrial deployment.

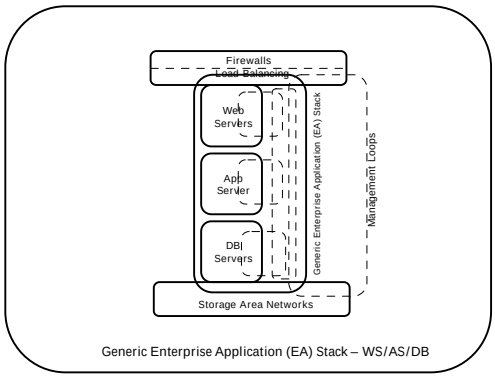

Fig. 4 Generic Enterprise Application (Stack)

Finally, we hope that the lessons learned from web-server experiments can be reused for application and database servers (even in a limited sense). Moreover, the same principles could be hopefully applied to entire stack of servers extended for load balancers and storage networks. Thus the results of this work will have wider meaning and give larger benefits (Fig. 4).

ACKNOWLEDGMENT

We would like to thank Richard Lloyd from Hewlett-Packard Co. for helpful comments and suggestions leading to improvements of our style and language.

REFERENCES

[1] November 2005 Netcraft Web Server Survey, http://news.netcraft.com
[2] Arlitt M. et al., Characterizing the scalability of a large Web-based shopping system, ACM Tr. On Internet Technology, Vol 1, No 1, Aug 2001, pp. 44-69
[3] Zhang R et al., ControlWare: A Middleware Architecture for Feedback Control of Software Performance, IEEE ICDCS'02
[4] A. Sloman , Damasio, Descartes, Alarms and Meta-Management, Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Diego, CA (1998), pp. 2652-2657
[5] Hellerstein J.L. et al., Feedback Control of Computing Systems, John Wiley & Sons, 2004.
[6] J. Giarratano and G. Riley, "Expert Systems: Principles and Programming", PWS-Kent, Boston, 1989
[7] J. R. Quinlan. "C4.5: Programs for Machine Learning", Morgan Kaufmann, San Mateo, California, 1993.
[8] J. A Freeman, D. M. Skapura, "Neural Networks: Algorithms, Applications and Programming Techniques", Addison-Wesley, 1991.
[9] Y. Diao et al., Managing Web server performance with AutoTune agents, IBM Systems Journal, Vol. 42, No. 1, 2003.
[10] J.P.Bigus et al., ABLE: A toolkit for building multiagent autonomic systems, IBM Systems Journal, Vol. 41, No3, 2002.
[11] T.F. Abdelzaher et al., Feedback Performance Control in Software Services, IEEE Control Systems Magazine, June 2003, pp-74-90.
[12] T.F. Abdelzaher et al., Performance Guarantees for Web Server End-Systems: A control-Theoretical Approach, IEEE Tr. On Parallel and Distributed Systems, Vol. 13, No. 1, January 2002.
[13] J.A. McCann, G. Jawaheer, The Patia Autonomic Webserver: Feasability Experimentation, 13th IEEE DEXA'03 Conference
[14] J.A. McCann et al., Patia: Adaptive Distributed Webserver, in Proceedings of 6th IEEE ISADS'03 Symposium
[15] R. Sterritt et al., PACT: Personal Autonomic Computing Tools, in Proceedings of 12th IEEE ECBS'05 Conference
[16] P. Barford, M. CrovellaM. Generating Representative Web Workloads for Network and Server Performance Evaluation, in SIGMETRICS'98
[17] D. Mosberger, T. Jin httperf – A Tool for Measuring Web Server Performance, HP Labs Report, 1998
[18] T. Voigt, Overload Behaviour and Protection of Event-driven Web Servers, Young, *2001*