Simulation Environment for a Fuzzy Controller based Autonomic Computing System

Harish S. V.,
Reader,
Dept. of Computer Science & Engg.,
Manipal Institute of Technology,
Manipal.
harish.sv@manipal.edu

K. Chandra Sekaran,
Professor,
Dept. of Computer Engg.,
National Institute of Technology,
Karnataka, Surathkal.
kch@nitk.ac.in

Abstract

eCommerce is an area where an Autonomic Computing system could be very effectively deployed. eCommerce has created demand for high quality information technology services and businesses seek ways to improve the quality of service in a cost-effective way. Properly adjusting tuning parameters for best values is time-consuming and skills-intensive. This paper describes a simulation environment to implement an approach to automate the tuning of MaxClients parameter of Apache web server using a fuzzy controller and knowledge of the affect of the parameter on quality of service. This is an illustration of the self-optimizing characteristic of an autonomic computing system.

1. Introduction

The advent and evolution of networks and Internet, which has delivered ubiquitous service with extensive scalability and flexibility, continues to make computing environments more complex [1]. Along with this, systems are becoming much more software-intensive, adding to the complexity. There is the complexity of business domains to be analyzed, and the complexity of designing, implementing, maintaining and managing the target system. I/T organizations face severe challenges in managing complexity due to cost, time and relying on human experts.

All these issues have necessitated the investigation of a new paradigm, Autonomic computing [1], to design, develop, deploy and manage systems by taking inspiration from strategies used by biological systems. eCommerce is one area where an Autonomic Computing system could be very effectively deployed. eCommerce has created demand for high quality information technology (IT) services and businesses seek ways to improve the quality of service (QoS) in a cost-effective way. As an example, performance of an Apache web server [17] is heavily influenced by the MaxClients parameter, but the optimum value of the parameter depends on system capacity and workload. Properly adjusting tuning parameters for best values is time-consuming and skills-intensive. This paper describes a simulation environment to implement an approach to automate the tuning of MaxClients parameter of Apache web server using a fuzzy controller.

From [2], we see that the autonomic computing architecture provides a blue print for developing feedback control loops for self-managing systems. This observation suggests that control theory will be of help in the construction of autonomic managers.

2. Related Work

Control theory has been applied to many computing systems, such as networks, operating systems, database management systems, etc. The authors in [3] propose to control web server load via content adaptation. The authors in [5] extend the scheme in [3] to provide performance isolation, service differentiation, excess capability sharing and QoS guarantees. In [4][8] the authors propose a relative differentiated caching services model that achieves differentiation of cache hit rates between different classes. The same objective is achieved in [6], which demonstrates an adaptive control methodology for constructing a QoS-aware proxy cache. The authors in [7] present the design and implementation of an adaptive architecture to provide relative delay guarantees for different service classes

978-0-7695-3516-6/08 $25.00 © 2008 IEEE
DOI 10.1109/ICACC.2009.105

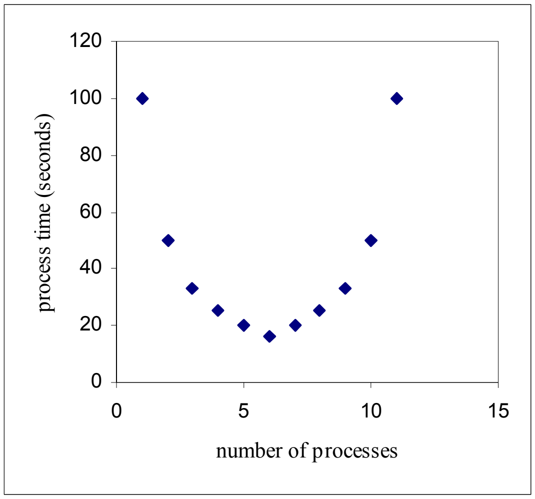

Figure 1. Process time curve

on web servers.

MIMO techniques are used in [9][10] to control the CPU and memory utilization in web servers. Queuing theory is used in [11] for computing the service rate necessary to achieve a specified average delay given the currently observed average request arrival rate. Same approach is used to solve the problem of meeting relative delay guarantees in [12].

The authors in [13] present a framework that monitors client perceived service quality in real-time with considerations of both network transfer time and server-side queuing delays and processing time. The authors in [14], present a fuzzy controller to guarantee absolute delays.

The authors in [15] propose an approach to automate enforcement of service level agreements (SLAs) by constructing information technology level feedback loops that achieve business objectives, especially maximizing SLA profits. Similarly, the authors in [16] [17] propose a profit-oriented feedback control system that automates the admission control decisions in a way that balances the loss of revenue due to rejected work against the penalties incurred if admitted work has excessive response times.

3. System Background

The system studied here is the Apache web server. In Apache version 1.3, there are a number of worker processes monitored and controlled by a master process [18]. The worker processes are responsible for handling the communications with the web clients. A worker process handles at most one connection at a time, and it continues to handle only that connection until the connection is terminated.

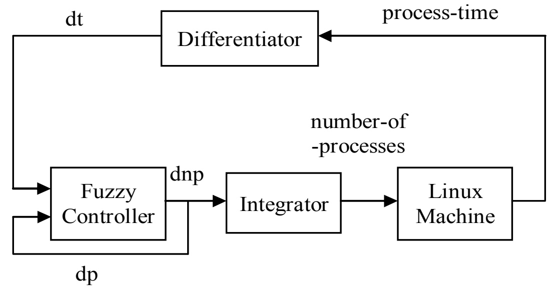

Figure 2. Block diagram of the fuzzy control system

A parameter termed *MaxClients* limits the size of this worker pool, thereby providing a kind of admission control in which pending requests are kept in the queue. MaxClients should be large enough so that more clients can be served simultaneously, but not so large that resource contention occurs. The optimal value depends on server capacity and the nature of the workload. The combined effect is that the response time is a concave upward function of MaxClients.

Figure 1 shows a typical curve to model the response type behavior of a typical Apache server. Here process time denotes the time taken by a process to run to completion. As shown in figure 1, if there is only 1 process it takes about 100 seconds to complete, if there are 2 processes, each of them take 50 seconds, and so on. The process time is minimum (about 16 seconds) when 6 processes are running. This corresponds to the optimum value of MaxClients in an Apache server.

4. Design of Fuzzy Controller

The block diagram of the fuzzy control system is shown in figure 2. The system being controlled is a linux machine. A number of processes will be running on the machine, the exact number depends on a parameter *number-of-processes*. The time taken by the processes are measured and input to a differentiator whose output is the *change-in-process-time* (dt) between current and previous intervals. The fuzzy controller has two inputs: change-in-process-time (dt) and *change-in-number-of-processes* (dp) between intervals. The controller's output is *next-change-in-number-of-processes* (dnp), whose value is taken as the value of change-in-number-of-processes for the next interval. An integrator converts this value into number-of-processes.The approach is similar to the one presented in [17].

Figure 3 shows the triangular membership functions used for the fuzzification of the inputs and defuzzification of the output. The measured numeric

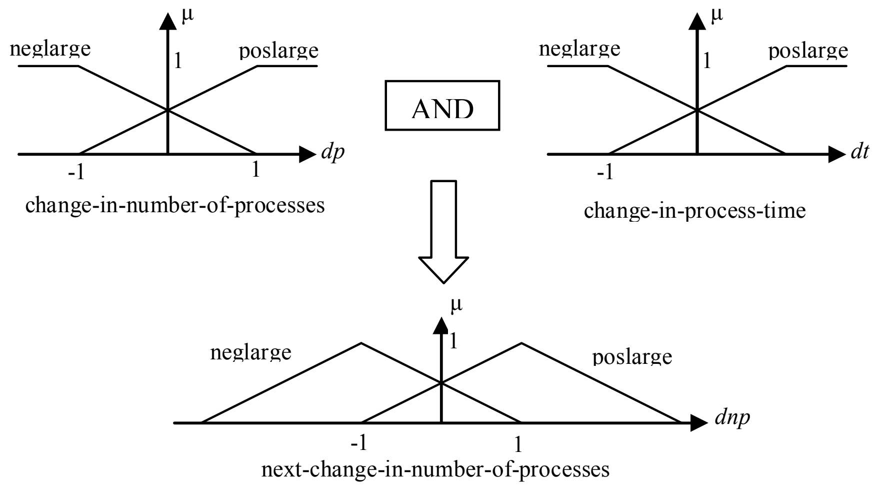

Figure 3. Membership functions and fuzzy inference

Table I. Fuzzy rule base

Rule	IF			THEN
	change-in-number -of-processes	AND	change-in-process-time	next-change-in-number-of-processes
1	neglarge	AND	neglarge	neglarge
2	neglarge	AND	poslarge	poslarge
3	poslarge	AND	neglarge	poslarge
4	poslarge	AND	poslarge	neglarge

values will be multiplied by factors known as the normalized gains, denoted by gdp and gdt. That is why the x-axis shows -1 and 1 for all the membership functions. The output value obtained will be denormalized by dividing by the normalized gain, gnp, to obtain the actual output value. Figure 1 illustrates that process time is a concave upward function of the number of processes. Hence, a gradient descent procedure is used to minimize process times. This is described using fuzzy rules shown in table I. Since the value of number of processes that minimizes the process time is not known, these rules are described in terms of changes to number of processes and process times values.

5. Implementation Details and Results

Fedora 5 running on a 2.26 GHz Intel Celeron desktop is used as the platform for running the simulations. The simulation environment consists of

- A load program to create processes
- A differentiator routine, which finds the difference between process times
- A fuzzy controller program, which finds the optimum value of the number of processes and
- An integrator routine, which obtains the value of required number of processes from change in number of processes.

Simulation readings are recorded after every interval, called *measurement* interval.

The load program reads it's input at the beginning of every measurement interval. The parent process in the load program creates and maintains that many child processes. Creation of each child process corresponds to the arrival of a client in Apache server. Hence, before creating a child process, a parent waits for a random duration. The time taken by the Apache server to service a client is simulated by means of a delay routine. This delay routine is invoked within each child process and the quantum of delay depends on number-of-processes so that the relation between the latter and process time is as shown in figure 1. Each child process, just before terminating sends the time taken to the differentiator.

The measurement interval should be large enough to reduce the effect of transients and also small enough so that the controller is able to quickly respond to changes. A measurement interval of 3 minutes was used. After waiting 2 minutes for the transients to reduce, 5 readings of process time are taken with a gap of 10 seconds between consecutive readings. The

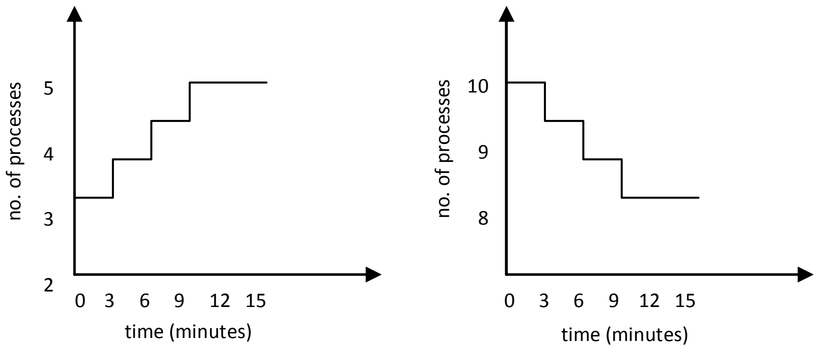

Figure 4. Number of processes reaching optimal value

Table II. Values of input and output variables

II(a) Processes increasing towards optimal value				
dp (normalized)	dt (normalized)	dnp (normalized)	dnp	no.-of-processes
-	-	-	-	2
0.500	-10.4	0.421	0.842	3
0.421	-3.4	0.352	0.703	4
0.352	-1.8	0.292	0.583	5
0.292	-1.0	0.240	0.480	5
0.240	0.0	0.0	0.0	5
II(b) Processes decreasing towards optimal value				
dp (normalized)	dt (normalized)	dnp (normalized)	dnp	no.-of-processes
-	-	-	-	10
-0.500	-8.8	-0.421	-0.842	9
-0.421	-3.4	-0.352	-0.703	8
-0.352	-1.6	-0.290	-0.580	7
-0.290	-1.2	-0.240	-0.480	7
-0.240	0.0	0.0	0.0	7

median of these 5 values was used to further reduce the effect of the transients. For the normalizing gains, large values increase the speed of the controller, but too large values will cause the system to oscillate. After experimenting with a few values, the values selected were gdp = gdnp = 1/2 and gdt = 1/5. This means a change of 2 in the number of processes or a change of 5 seconds in process time is considered to be large.

Figure 4 shows the number of processes reaching the neighborhood of the optimal value. In the first case, the number of processes starts from the left end of the process time curve, while in the second case, it starts from the right end. In both the cases, the number of processes is not able to reach the optimal value. The same data is shown in table II along with the input and output variables.

6. Conclusions

This paper describes a simulation environment to illustrate the self-optimizing characteristic of an autonomic computing system. Here quality of service is optimized by using fuzzy control. The simulation environment provides a framework to experiment with enhancements and modifications to the basic autonomic computing system used here.

Possible future work includes speeding up the controller (implementing it in hardware is one of the options) and also incorporating learning heuristics so that the fuzzy rules could be automatically derived and modified as and when required.

7. References

[1] M. Salehie and L. Tahvildari, “Autonomic Computing: Emerging trends and open problems,” in Proceedings of the Workshop on the Design and Evolution of Autonomic Application Software, 2005.

[2] Y. Diao, J.L. Hellerstein, S. Parekh, R. Griffith, G. E. Kaiser and D. Phung, “A control theory foundation for self-managing computing systems,” IEEE Journal on Selected Areas in Communications, Vol. 23, No. 12, December 2005.

[3] T. F. Abdelzaher and N. Bhatti, “Web server Quality of Service management by adaptive content delivery,” International Workshop on Quality of Service, June 1999.

[4] Y. Lu, A. Saxena and T. F. Abdelzaher, “Differentiated caching services - A control-theoretical approach,” Proceedings of the International Conference on Distributed Computing Systems, April 2001.

[5] T. F. Abdelzaher, K. G. Shin and N. Bhatti, “Performance guarantees for web server end-systems : A control-theoretical approach,” IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 1, January 2002.

[6] Y. Lu, T. F. Abdelzaher, C. Lu and G. Tao, “An adaptive control framework for QoS guarantees and it’s application to differentiated caching services,” Proceedings of the International Conference on Quality of Service, May 2002.

[7] C. Lu, T. F. Abdelzaher, J. A. Stankovic and S. H. Son, “A feedback control approach for guaranteeing relative delays in web servers,” Proceedings of the IEEE Real-Time Technology and Applications Symposium, June 2001.

[8] Y. Lu, T. F. Abdelzaher and A. Saxena, “Design, implementation and evaluation of differentiated caching services,” IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 5, May 2004.

[9] Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh and D. M. Tilbury, “Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache web server,” Proceedings of the IEEE/IFIP Network Operations and Management, April 2002.

[10] N. Gandhi, D. M. Tilbury, Y. Diao, J. Hellerstein and S. Parekh, “MIMO control of an Apache web server : Modeling and controller design,” Proceedings of the American Control Conference, May 2002.

[11] L. Sha, X. Liu, Y. Lu and T. F. Abdelzaher, “Queuing model based network server performance control,” Proceedings of the IEEE Real-Time Systems Symposium, 2002.

[12] Y. Lu, T. Abdelzaher, C. Lu, L. Sha and X. Liu, “Feedback control with queuing-theoretic prediction for relative delay guarantees in web servers,” Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium, 2003.

[13] J. Wei and C. Xu, “Feedback control approaches for Quality of Service guarantees in web servers,” Fuzzy Information Processing Society, 2005.

[14] Y. Wei, C. Lin, T. Voigt and F. Ren, “Fuzzy control for guaranteeing absolute delays in web servers,” Proceedings of the 2nd International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks, August 2005.

[15] Y. Diao, J. L. Hellerstein and S. Parekh, “A business-oriented approach to the design of feedback loops for performance management,” Proceedings of the 12th IEEE International Workshop on Distributed Systems : Operations and Management, October 2001.

[16] Y. Diao, J. L. Hellerstein and S. Parekh, “Using fuzzy control to maximize profits in service level management,” IBM Systems Journal, Vol. 41, No. 3, 2002.

[17] Y. Diao, J. L. Hellerstein and S. Parekh, “Optimizing Quality of Service using fuzzy control,” Proceedings of Distributed Systems Operations and Management, 2002 - Springer

[18] Apache Software Foundation. http://www.apache.org.