An Approach in using Differentiated Services to Maximize Profit in an Autonomic Computing System

Harish S. Venkatarama
Reader, Dept. of Computer Science & Engg.,
Manipal Institute of Technology,
Manipal, India

Chandrasekaran Kandasamy
Professor, Dept. of Computer Engg.,
National Institute of Technology Karnataka,
Surathkal, India

ABSTRACT

Ecommerce is an area where an Autonomic computing system could be very effectively deployed. The growth of ecommerce has created demand for services with financial incentives for service providers. Revenues accrue if the admitted requests are processed within the specified deadline and costs are incurred otherwise. In case of heavy load, it will not be possible to process all requests within the deadlines. It is beneficial to concentrate on those requests with which larger profits are associated. This paper describes an approach wherein a fuzzy controller is used which automatically allocates resources for priority requests in proportion to the number of priority requests. This is an illustration of the self-optimizing characteristic of an autonomic computing system.

General Terms

Autonomic computing

Keywords

Autonomic computing, E-commerce, Differentiated, Fuzzy control

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. The growth of ecommerce has created demand for services with financial incentives for service providers. Revenues accrue if the admitted requests are processed within the specified deadline and costs are incurred otherwise. In case of heavy load, it will not be possible to process all requests within the deadlines. It is beneficial to concentrate on those requests with which larger profits are associated. This paper describes an approach wherein a fuzzy controller is used which automatically allocates resources for priority requests in proportion to the number of priority requests. When the number of priority requests increase, resources allocated for processing these requests is proportionately increased and vice versa. This ensures that ordinary requests do not needlessly suffer. This is an illustration of the self-optimizing characteristic of an autonomic computing system.

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 on web servers.

Real-time scheduling theory makes response-time guarantees possible, if server utilization is maintained below a pre-computed bound. Feedback control is used in [9] to maintain the utilization around the bound. The authors in [10] [11] demonstrate the power of a control theoretic analysis on a controller for doing admission control of a Lotus Notes workgroup server.

MIMO techniques are used in [12] [13] to control the CPU and memory utilization in web servers. Queuing theory is used in [14] 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 [15].

The authors in [16] 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 [17], present a fuzzy controller to guarantee absolute delays.

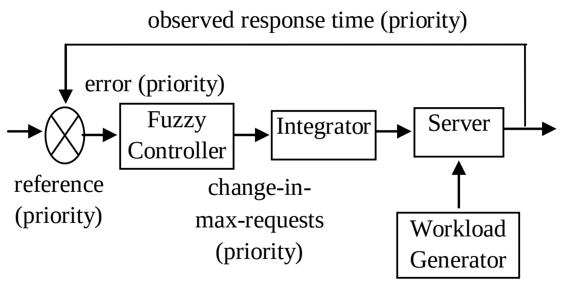

Figure 1. Fuzzy control system

The authors in [18] present a Linear-Parameter-Varying approach to the modeling & design of admission control for Internet web servers. The authors in [19] [20] study the performance/power management of a server system.

The authors in [21] propose an approach to automate enforcement of service level agreements (SLAs) by constructing information technology (IT) level feedback loops that achieve business objectives, especially maximizing SLA profits. Similarly, the authors in [22] 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. The authors in [23] describe an approach to automate parameter tuning using a fuzzy controller that employs rules incorporating qualitative knowledge of the effect of tuning parameters.

This paper presents an approach similar to the relative differentiated service, but with a difference. The proportion of resources allocated to process the priority requests versus the ordinary requests depends on the proportion of priority requests versus ordinary requests. The controller ensures that priority requests are processed within the deadline, while no guarantees are offered for the ordinary requests. However, the resource allocation is monitored at regular intervals by the controller to ensure that ordinary requests do not needlessly suffer.

3. SYSTEM BACKGROUND

The system studied here is the Apache web server. In Apache version 2.2 (configured to use Multi-Processing Module prefork), there are a number of worker processes monitored and controlled by a master process [24]. 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. Thus the worker is idle between consecutive requests from its connected client. 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.

The client server architecture is simulated here as a M/M/1 queue. The parameter total-max-requests used here is assumed to be analogous to MaxClients. Parameter total-max-requests consists of priority-max-requests and ordinary-max-requests. The controller adjusts priority-max-requests at regular intervals.

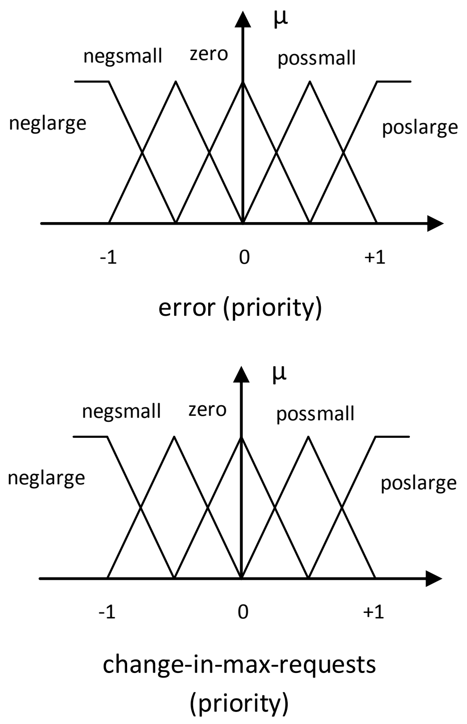

Figure 2. Membership functions

4. DESIGN OF FUZZY CONTROL SYSTEM

The block diagram of the fuzzy control system is shown in figure 1. The simulation environment consists of a workload generator program to generate requests, a server program to service the requests, a fuzzy controller program and an integrator routine.

The workload generator generates requests such that the time between generations of consecutive requests is exponentially distributed. For each request received by the server, the parent process creates a child process which sleeps for a time which is exponentially distributed. Thus, the client server architecture is simulated here as an M/M/1 queue. The workload generator generates two types of requests, priority requests and ordinary requests. Priority requests are those requests for which the waiting time in the queue is zero or at most equal to a specified reference value. In the ideal case, priority requests should have zero waiting time. However, it may not be possible to achieve this goal always, since the proportion of priority requests to ordinary requests can change rapidly. Thus, a separate queue is maintained in the server for priority requests. The number of priority requests accepted by the server, is limited by the parameter priority-max-requests, which is updated by the integrator at the beginning of every measurement interval. Simulation readings are recorded after every interval, called measurement interval.

Any fuzzy control system involves three main steps, that is, fuzzification, inference mechanism and defuzzification. Figure 2 shows the triangular membership functions used for the

Table 1. Fuzzy Rules

Rule	IF	THEN
	error (priority)	change-in-max-requests (priority)
1	neglarge	poslarge
2	negsmall	possmall
3	zero	zero
4	possmall	negsmall
5	poslarge	neglarge

fuzzification of the input and defuzzification of the output. In each case, the parameter is divided into 5 intervals called neglarge, negsmall, zero, possmall and poslarge. Neglarge is an abbreviation for "negative large in size". Similarly negsmall, possmall and poslarge are abbreviations. Zero is the name of the interval denoting small changes. The measured numeric values will be multiplied by factors known as the normalized gains. That is why the x-axis shows -1 and 1 for all the membership functions. The output value, change-in-max-requests (priority), obtained will be denormalized by dividing by the normalized gain to obtain the actual output value. The fuzzy rules describing the working of the controller is shown in Table 1.

5. IMPLEMENTATION

The workload generator is a program which continuously generates requests such that the time between generation of consecutive requests is exponentially distributed with mean = 0.2 seconds. That is, it generates 5 requests per second on the average.

The server is a program which services the requests such that the time taken for servicing each request is exponentially distributed with mean = 60 seconds. This means, 1 process running on the server, can service 1 request per minute on an average. With 5 requests being generated per second, a reasonable value for total-max-requests would be 300. For this simulation, it is fixed at 280. The simulation is run for 4000 seconds.

The fuzzy controller program takes as input error (priority), which is observed-response-time (priority) subtracted from the reference (priority) value. The controller calculates the adjustment required for priority-max-requests, i.e., priority-max-requests-change for the next measurement interval. This value is sent to the integrator, which calculates the value of priority-max-requests for the next interval. Value of ordinary-max-requests is obtained by subtracting priority-max-requests from total-max-requests.

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, waiting times of requests that entered service in the last 1 minute are noted. The average of these values is taken as the observed-response-time (priority).

6. RESULTS AND VALIDATION

Table 2 shows the results for a reference value of 1 second. After an initial adjustment to priority-max-requests, no further adjustment is required as the observed-response-time is well within the reference value. Though the deadline for processing the priority request is 1 second, it is seen that almost all requests are processed instantaneously on arrival. It may be possible to design a more sensitive controller which would, perhaps, reduce the resources allocated to priority requests to some extent.

Table 3 shows the results for a reference value of 2 seconds. As the reference value is more, the controller, at many occasions deallocates the extra resources allocated to the priority requests. Here it is observed that on two occasions, the observed value of response time exceeds the reference value. The reason for this is that there is high variance in input, since both the interarrival times and the service times are exponentially distributed.

Figure 3 shows the plots of response times of priority requests (top) and ordinary requests (bottom) for a reference value of 2 seconds. The figures on the left top and bottom are the plots with the controller enabled, while those on the right top and bottom are without the controller. A dramatic difference in the response times of priority requests can be observed, while no such thing is seen in case of ordinary requests. This is, as expected, since the controller is only concerned with priority requests. Thus, the proposed fuzzy model is validated.

7. CONCLUSIONS

This paper describes an approach to minimize response time for priority requests in an ecommerce system using fuzzy control. This is an illustration of the self-optimizing characteristic of an autonomic computing system. Specifically, the system studied here is the allocation of MaxClients parameter of the Apache web server for requests of different classes. The workload and server are simulated as an M/M/1 queue. The controller attempts to optimize priority-max-requests, which decides the number of processes for servicing priority requests. It is easily seen from the results, that a single fixed value of priority-max-requests will not be optimum for all cases. Since the proportion of priority requests of a server can change rapidly, it is of immense benefit to have a controller which updates the value of priority-max-requests at regular intervals.

Table 2. With reference (priority) = 1 second

max-requests		observed time		error	max-req -change
(priority)	(ordinary)	(priority)	(ordinary)	(priority)	(priority)
10	270	11.2	0.8	-10.2	5
15	265	0.0	14.0	1.0	0
15	265	0.0	24.2	1.0	0
15	265	0.0	21.3	1.0	0
15	265	0.0	19.9	1.0	0
15	265	0.0	24.2	1.0	0
15	265	0.0	23.2	1.0	0
15	265	0.0	20.6	1.0	0
15	265	0.0	20.3	1.0	0
15	265	0.0	19.4	1.0	0
15	265	0.0	21.4	1.0	0

Table 3. With reference (priority) = 2 seconds

max-requests		observed time		error	max-req -change
(priority)	(ordinary)	(priority)	(ordinary)	(priority)	(priority)
10	270	11.2	0.8	-9.2	5
15	265	0.0	14.0	2.0	-1
14	266	0.0	24.2	2.0	-1
13	267	0.0	21.3	2.0	-1
12	268	0.0	20.3	2.0	-1
11	269	2.6	24.4	-0.6	1
12	268	2.3	23.3	-0.3	0
12	268	1.0	20.5	1.0	0
12	268	1.6	20.2	0.4	0
12	268	0.0	18.9	2.0	-1
11	269	0.0	21.4	2.0	-1

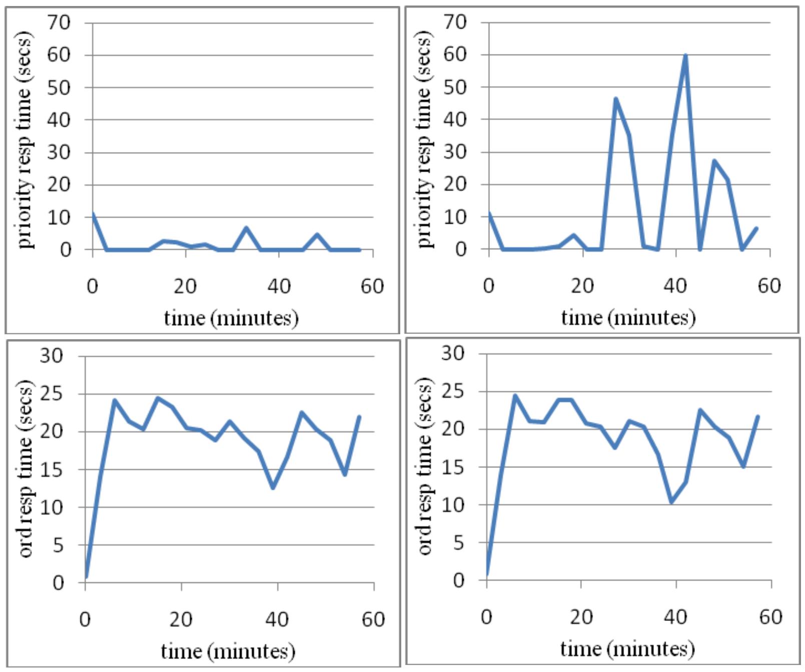

Figure 3. Graph showing plots of response times with the controller (left top and bottom) and without the controller (right top and bottom)

8. REFERENCES

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

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

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

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

[5] Abdelzaher, T. F., Shin, K. G. and Bhatti, N., “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] Lu, Y., Abdelzaher, T. F., Lu, C. and Tao, G., “An adaptive control framework for QoS guarantees and its application to differentiated caching services,” Proceedings of the International Conference on Quality of Service, May 2002.

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

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

[9] Abdelzaher, T. F. and Lu, C., “Modeling and performance control of internet servers,” IEEE Conference on Decision and Control, December 2000.

[10] Parekh, S., Gandhi, N., Hellerstein, J., Tilbury, D., Jayram, T. and Bigus, J., “Using control theory to achieve service level objectives in performance management,” IFIP/IEEE International Symposium on Integrated Network Management, May 2001.

[11] Gandhi, N., Tilbury, D. M., Parekh, S. and Hellerstein, J., “Feedback control of a lotus notes server : Modeling and control design,” Proceedings of the American Control Conference, June 2001.

[12] Diao, Y., Gandhi, N., Hellerstein, J. L., Parekh, S. and Tilbury, D. M., “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.

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

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

[15] Lu, Y., Abdelzaher, T. F., Lu, C., Sha, L. and Liu, X., “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.

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

[17] Wei, Y., Lin, C., Voigt, T. and Ren, F., “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.

[18] Qin, W. and Wang Q., “Feedback performance control for computer systems: an LPV approach,” Proceedings of the American Control Conference, June 2005.

[19] Qin, W., Wang, Q., Chen, Y. and Gautham, N., “A first-principles based LPV modeling and design for performance management of Internet web servers,” Proceedings of the American Control Conference, June 2006.

[20] Qin, W. and Wang, Q., “Modeling and control design for performance management of web servers via an LPV approach,” IEEE Transactions on Control Systems Technology, Vol. 15, No. 2, March 2007.

[21] Diao, Y., Hellerstein, J. L. and Parekh, S., “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.

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

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

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