Minimizing Response Time in an Autonomic Computing System Using Proportional Control

Harish S. Venkatarama[1] and Kandasamy Chandra Sekaran[2]

[1] Reader, Computer Science & Engg. Dept.,
Manipal Institute of Technology, Manipal, India
harish.sv@manipal.edu
[2] Professor, Dept. of Computer Engg.,
National Institute of Technology Karnataka, India
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 are seeking quality of service guarantees from their service providers. These guarantees are expressed as part of service level agreements. Properly adjusting tuning parameters for enforcement of the service level agreement is time-consuming and skills-intensive. Moreover, in case of changes to the workload, the setting of the parameters may no longer be optimum. In an ecommerce system, where the workload changes frequently, there is a need to update the parameters at regular intervals. This paper describes an approach to automate the tuning of MaxClients parameter of Apache web server using a proportional controller based on the required response time and the current workload. This is an illustration of the self-optimizing characteristic of an autonomic computing system.

Keywords: autonomic computing, ecommerce, proportional 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. Ecommerce has created demand for high quality information technology (IT) services and businesses are seeking quality of service (QoS) guarantees from their service providers (SPs). These guarantees are expressed as part of service level agreements (SLAs). As an example, performance of an Apache web server [16] is heavily influenced by the MaxClients parameter, but the optimum value of the parameter depends on system capacity, workload and the SLA. Properly adjusting tuning parameters for enforcement of the SLA is time-consuming and skills-intensive. Moreover, in case of changes to the workload, the setting of the parameters may no longer be optimum. In an ecommerce system, where the workload changes frequently, there is a need to update the parameters at regular intervals.

The simplified architecture for autonomic computing is shown in fig. 1. Adding an autonomic manager makes the resource self-managing [2]. The manager gets required data through the sensors and regulates the behavior of the resource through effectors. This shows how self-managing systems are developed using feedback control loops. This observation suggests that control theory will be of help in the construction of autonomic managers.

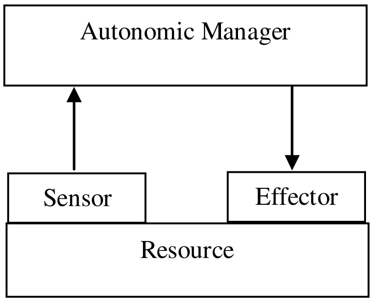

Fig. 1. Autonomic computing architecture

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].

This paper describes an approach to automate the tuning of MaxClients parameter of Apache web server using a proportional controller. The controller maximizes the number of users allowed to connect to the system subject to the response time constraint as given in the SLA. This is an illustration of the self-optimizing characteristic of an autonomic computing system.

2 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 [16]. 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. MaxClients should be large enough so that more clients can be served simultaneously, but not so large that response time constraints are violated. If MaxClients is too small, there is a long delay due to waits in the queue. If it is too large resources become over utilized which degrades performance as well. The optimal value depends on server capacity, nature of the workload and the SLA.

3 Modeling and System Identification

Fig. 2 shows the scheme used for system identification. The simulation environment consists of a workload generator which generates requests and a server program which services the requests.

In this model, parameter max-requests is varied from 200 in steps of 10 and the corresponding response time values are noted. A first order ARX model is used to describe the relationship between inputs and outputs.

$$y(k+1) = a*y(k) + b*u(k) \tag{1}$$

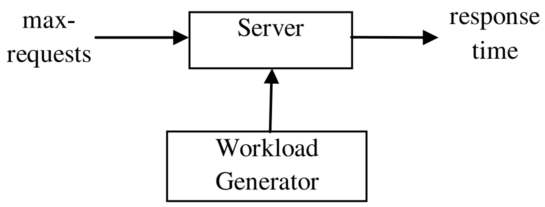

Fig. 2. Modelling the system

Here, u is the input or actuating signal, y is the output signal and a and b are scalars. Since a discrete signal has value only at specific instants of time, an integer k is used to index these instants. Using least squares regression, values for a and b are estimated as a = 0.1 and b = -0.36. That is, we arrive at the model

$$y(k+1) = 0.1*y(k) - 0.36*u(k) \tag{2}$$

4 Controller Design

We use the proportional control law.

$$u(k) = K_P*e(k) \tag{3}$$

Here K_P is a constant called gain of the controller. The actuating signal is proportional to the present error signal. It is not dependent on the past values of the error. Taking Z transform of equation (2) and manipulating, we get the open loop transfer function.

$$G(z) = Y(z) / U(z) = -0.36 / (z-0.1) \tag{4}$$

Closed loop transfer function is as follows.

$$F_R(z) = Y(z) / R(z) = K_P*G(z) / (1+K_P*G(z))$$

Solution of the characteristic equation, $1+K_P*G(z) = 0$ gives the poles. For the system in question, there is only 1 closed loop pole, given by the following equation.

$$p1 = 0.1 + 0.36*K_P$$

For stability, we need to have,

$$| 0.1 + 0.36*K_P | < 1 \text{ or } -3.1 < K_P < 2.5$$

5 Implementation

Fig 3 shows the system for proportional control which is used for the implementation. In terms of fig. 1, server is the resource and controller is the autonomic manager. Response time is converted to error signal, which corresponds to input to the manager from the sensor. Just as the behavior of the resource is influenced by the effector, the server is influenced by max-requests.

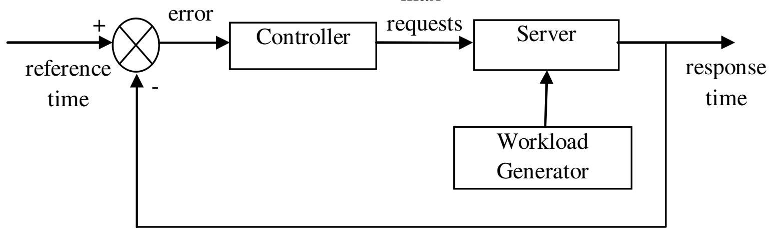

Fig. 3. System for Proportional control

The incoming request from the workload generator is first put into a queue in the server. When the server becomes free, the first request in the queue is dequeued. The time spent by the request in the queue is called the response time. The workload generator generates requests such that the time between generations of consecutive requests is exponentially distributed. Also, the time taken by the server to process each request is exponentially distributed. Thus, the client server architecture is simulated here as an M/M/1 queue.

Workload generator is set to generate requests such that the time between arrivals of consecutive requests on an average (mean interarrival) is 0.2 second. That is 300 requests per minute on an average. Mean service time is set to 60 seconds. Readings are noted every 3 minutes. To ensure that transients do not affect the readings, readings are taken for the last 1 minute of the 3 minute interval. Response time values of the requests which entered service in the last 1 minute are noted and the average is calculated. In this simulation, MaxClients is simulated by max-requests.

The controller tries to drive the error signal to 0 by adjusting the value of max-requests at regular intervals. That is, it tries to make the response time equal to the reference time.

The simulation is carried out using C-based simulation language "simlib" [17]. The experiment was repeated for different values of reference times and different values of gains. Each simulation was run for 60 minutes.

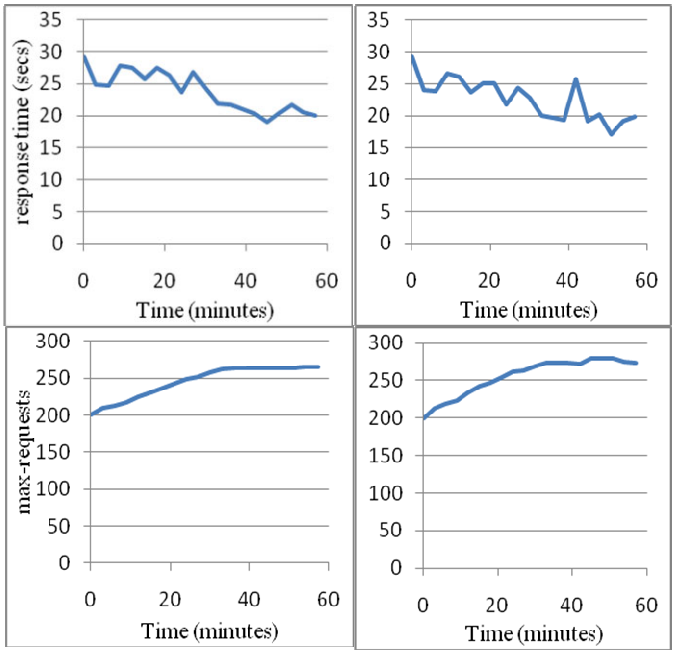

Fig. 4. With reference time = 20 secs, K_P = -1.0 (left hand side) and K_P = -1.5 (right hand side)

6 Results

Fig. 4 shows the results for reference time = 20 seconds with gain values equal to -1.0 (top and bottom figures on left hand side) and -1.5 (top and bottom figures on right hand side) respectively. The plots at the top of the figure show the variation of response time, while the plots at the bottom show the variation in max-requests. It is seen that when the gain is larger, the controller tries to correct the error more aggressively, but it results in undershoot. There is very little undershoot when the gain is less. Fig. 5 shows the results for reference time = 25 seconds with gain values equal to -1.0 (left hand side) and -1.5 (right hand side) respectively. Here also, undershoot is observed in case of larger gain.

For verifying the stability of the system, the simulation was run for durations upto 18,000 seconds (5 hours). For gain values shown above, the system was stable for the entire duration. However, for larger values of the gain, it was observed that the system was becoming increasingly unstable. Hence, results shown are only for those values of gains for which the system was stable.

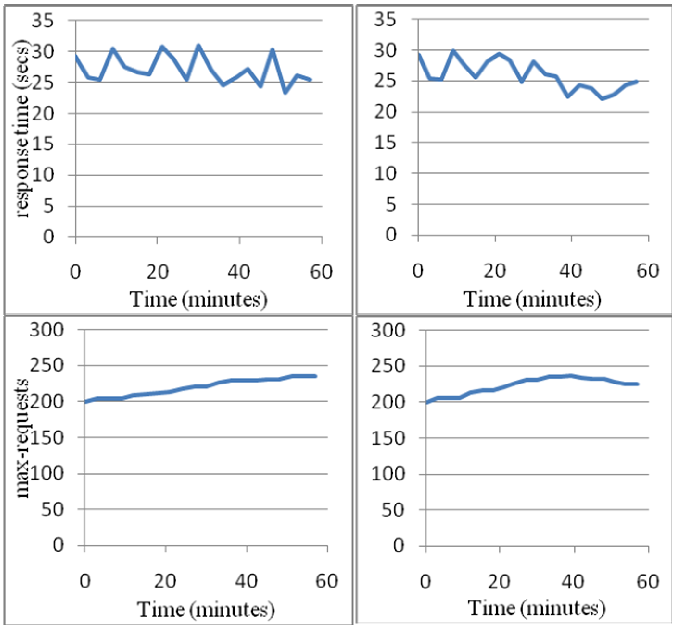

Fig. 5. With reference time = 25 secs, K_P = -1.0 (left hand side) and K_P = -1.5 (right hand side)

7 Conclusions

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

Though the controller is properly able to adjust value of max-requests, it takes some time for it to converge to the optimum value. Increasing the gain led to stability problems. Thus, as part of future work, it is intended to find ways to speed up the working of the system. It is also intended to test the functioning of the controller under different simulation environments like having an arbitrary (general) distribution for the service time, i.e., simulating the workload and server as an M/G/1 queue.

References

1. Salehie, M., Tahvildari, L.: Autonomic Computing: Emerging trends and open problems. In: Proc. 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., Phung, D.: A control theory foundation for self-managing computing systems. IEEE J. on Selected Areas in Communications 23(12) (December 2005)
3. Abdelzaher, T.F., Bhatti, N.: Web server Quality of Service management by adaptive content delivery. In: Intl. Workshop on Quality of Service (June 1999)
4. Lu, Y., Saxena, A., Abdelzaher, T.F.: Differentiated caching services - A control-theoretical approach. In: Proc. of the International Conference on Distributed Computing Systems (April 2001)
5. Abdelzaher, T.F., Shin, K.G., Bhatti, N.: Performance guarantees for web server end-systems: A control-theoretical approach. IEEE Trans. on Parallel and Distributed Systems 13(1) (January 2002)
6. Lu, Y., Abdelzaher, T.F., Lu, C., Tao, G.: An adaptive control framework for QoS guarantees and it's application to differentiated caching services. In: Proc. of the Intl. Conference on Quality of Service (May 2002)
7. Lu, C., Abdelzaher, T.F., Stankovic, J.A., Son, S.H.: A feedback control approach for guaranteeing relative delays in web servers. In: Proc. of the IEEE Real-Time Technology and Applications Symposium (June 2001)
8. Lu, Y., Abdelzaher, T.F., Saxena, A.: Design, implementation and evaluation of differentiated caching services. IEEE Transactions on Parallel and Distributed Systems 15(5) (May 2004)
9. Abdelzaher, T.F., Lu, C.: Modeling and performance control of internet servers. In: IEEE Conf. on Decision and Control (December 2000)
10. Parekh, S., Gandhi, N., Hellerstein, J., Tilbury, D., Jayram, T., Bigus, J.: Using control theory to achieve service level objectives in performance management. In: IFIP/IEEE Intl. Symposium on Integrated Network Management (May 2001)
11. Gandhi, N., Tilbury, D.M., Parekh, S., Hellerstein, J.: Feedback control of a lotus notes server: Modeling and control design. In: Proc. of the American Control Conference (June 2001)
12. Diao, Y., Gandhi, N., Hellerstein, J.L., Parekh, S., Tilbury, D.M.: Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache web server. In: Proc. of the IEEE/IFIP Network Operations and Management (April 2002)

13. Gandhi, N., Tilbury, D.M., Diao, Y., Hellerstein, J., Parekh, S.: MIMO control of an Apache web server: Modeling and controller design. In: Proc. of the American Control Conference (May 2002)
14. Sha, L., Liu, X., Lu, Y., Abdelzaher, T.F.: Queuing model based network server performance control. In: Proc. of the IEEE Real-Time Systems Symposium (2002)
15. Lu, Y., Abdelzaher, T.F., Lu, C., Sha, L., Liu, X.: Feedback control with queuing-theoretic prediction for relative delay guarantees in web servers. In: Proc. of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (2003)
16. Apache Software Foundation, `http://www.apache.org`
17. Law, A.M.: Simulation Modeling and Analysis. Tata McGraw Hill Publishing Company Ltd., New Delhi (2008)