A Practical Approach to Automatic Parameter-Tuning of Web Servers

Akiyoshi Sugiki[1], Kenji Kono[2], and Hideya Iwasaki[1]

[1] Department of Computer Science,
The University of Electro-Communications,
1-5-1 Chofugaoka, Chofu, Tokyo, Japan
sugiki@zeus.cs.uec.ac.jp, iwasaki@cs.uec.ac.jp
[2] Department of Information and Computer Science,
Keio University, 3-14-1 Hiyoshi,
Kohoku-ku, Yokohama, Japan
kono@ics.keio.ac.jp

Abstract. This paper presents a practical approach to automatically tuning the parameters of the Apache Web server. In particular, two significant parameters, `KeepAliveTimeout` and `MaxClients`, are dealt with. The notable features of our approach are twofold. First, it is easy to deploy because no modifications to Apache or the underlying operating system are required. Second, our approach is based on the detailed analysis on how each parameter affects the server's behavior. Experimental results demonstrate that our prototype works well on different workloads; it can discover almost optimal values and quickly adapt to workload changes.

1 Introduction

Modern Internet servers are growing rapidly in size and complexity. To maintain good performance and availability, an administrator is confronted with a huge amount of time consuming tasks to tune the server's ever-changing parameters. Performance parameters are especially difficult to tune *manually* for three reasons. First, it is not obvious what value is proper for each performance parameter. This is because the proper value largely depends on the execution environment. Second, it is time-consuming to find the proper values because tedious trials and errors must be repeated. Third, the proper value may change over time because the execution environment changes.

In this paper, we present a practical approach to automatically tuning the parameters of the famous Apache Web sever [1]. In particular, this paper deals with two major performance parameters, `KeepAliveTimeout` and `MaxClients`, because much of the literature [2, 3] has pointed out these greatly influence on Apache performance. Although we present our methodology within the context of Apache, we believe it could be applied to other Web servers.

The notable features of our methodology can be summarized as follows:

– **Easy to deploy:** Our mechanism does not require any modifications to Apache and the underlying operating system. To incorporate our mechanism, it is sufficient to restart Apache after defining a special environment variable. Since Apache is so widely used, it would help many administrators to tune their Web servers and thus have a practical impact.

- **Based on parameter analysis:** We exploit parameter-specific features to adjust `KeepAliveTimeout` and `MaxClients`. We analyzed the effects each parameter had on the behavior of Web servers to develop the tuning algorithm, and derived it to discover the proper value for each parameter. To derive both algorithms, HTTP-request intervals were assessed for `KeepAliveTimeout`, and resource contention was investigated for `MaxClients`.

Our approach based on parameter-specific analysis is effective for the performance-influenced parameters of widely used servers such as Apache. Although some might think our approach is not generic, our claim is that `KeepAliveTimeout` and `Max Clients` deserve to special consideration.

Our prototype system was implemented on Linux 2.4.20 and ran with Apache Web server 2.0.49. Experimental results demonstrated that our approach could successfully tune both `KeepAliveTimeout` and `MaxClients` to nearly optimal and manually tuned values. By automatically tuning `KeepAliveTimeout`, the throughput was improved by 27.5 – 368.5% compared to the default setting. Automatically tuning `MaxClients` also resulted in throughput close to the nearly optimal, hand-tuned level of Apache.

The rest of this paper is organized as follows. Section 2 analyzes the effects performance parameters had on the Apache. Section 3 introduces our mechanism. Section 4 describes the implementation of our prototype system. Section 5 presents the experimental results. Section 6 discusses related work. Finally, we conclude with a summary in Sect. 7.

2 Performance Effects on Parameters

It is important to adjust both `KeepAliveTimeout` and `MaxClients` properly, since they significantly affect server performance. To confirm whether this was the case, we measured the performance of the Apache Web server [1] using a standard Web benchmark (for details, refer to Sect. 5). Figure 1 plots the server throughput and average response time for various `MaxClients` and `KeepAliveTimeout` values.

We can see that `KeepAliveTimeout` dramatically affects server performance. When `KeepAliveTimeout` is set to 400 ms, the throughput improves up to 150.2% and the response time improves up to 13.3%, compared to the default `KeepAlive Timeout` value (i.e. 15 sec) of the Apache Web server. When this value is changed to 200 ms or 600 ms, the server throughput and response time both degrade down to 20.3% and 23.0% respectively, compared to 400 ms. This implies that `KeepAlive Timeout` is difficult to adjust manually; a slight difference in `KeepAliveTimeout` affects server performance.

Figure 1(b) shows that `MaxClients` also affects server performance. With this benchmark workload, the server yields the best throughput and moderate response time if its `MaxClients` is set to 700. Compared to the default `MaxClients` (150), the server throughput improves up to 297.9%. When the `MaxClients` is changed to 600