Future Generation Computer Systems 28 (2012) 1110–1120

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A weighted-fair-queuing (WFQ)-based dynamic request scheduling approach in a multi-core system

Guohua You, Ying Zhao *

College of Information Science and Technology, Beijing University of Chemical Technology, 100029 Beijing, PR China

ARTICLE INFO

Article history:
Received 18 November 2010
Received in revised form
15 April 2011
Accepted 13 July 2011
Available online 3 November 2011

Keywords:
Dynamic requests
Scheduling
Web server
Multi-core
Threads
Hard affinity

ABSTRACT

A popular website is expected to simultaneously deal with a large number of dynamic requests in the reasonable mean response time. The performance of websites mainly depends on hardware performance and the processing strategy of dynamic requests. In order to improve the hardware performance, more and more web servers are adopting multi-core CPUs. Moreover, the scheduling algorithm of requests on the first-come–first-served (FCFS) basis is still utilized. Although FCFS is a reasonable and fair strategy for request sequences, it takes into account neither the distribution of the dynamic request service times nor the characteristics of multi-core CPUs. In the present paper, in order to solve the above-mentioned problems, a new dynamic request scheduling approach is proposed. The new scheduling approach, according to the distribution of the dynamic request service time, schedules the dynamic requests based on a weighted-fair-queuing (WFQ) system, and exploits the performance of multi-core CPUs by means of the hard affinity method in the O/S. Simulation experiments have been done to evaluate the new scheduling approach, and the results obtained prove that the new scheduling approach could eliminate the ping-pong effect and efficiently reduce the mean response time.

1. Introduction

With the huge development of the Internet industry, people are more like to rely on the web for their daily activities such as e-commerce, online banking, stock trading, reservation, and product merchandising. Consequently, popular web sites are expected to deal simultaneously with large numbers of requests without noticeable reduction of the response time performance. Moreover, dynamic and personalized content delivery has been increased sharply with the application of server-side scripting technologies. Web pages incorporating the latest customized information are generated dynamically, but they are not cacheable. So the generation of these dynamic web pages is heavy to load on the web server. Furthermore, with the progress of broadband communication technology, web servers tend to become performance bottlenecks. The performance of web servers mainly depends on the hardware and the scheduling strategy of requests.

At the same time, with the development of multi-core technology, web servers have mostly adopted multi-core CPUs to improve the hardware performance in the past few years. A multi-core system integrates two or more processing cores into one silicon chip [1–3]. In this type of design, every processing core has its own private L1 cache and shared L2 cache [4,5]. All the processing cores share the main memory and the system bandwidth. Fig. 1 shows the architecture in a multi-core system. When web servers adopt multi-core CPUs, there will be some new problems.

There is usually one or more thread pool in the web server, in which threads are usually in the blocking state. When a request arrives at the thread pool, the web server fetches a blocked thread from the thread pool, and then assigns the request to the thread and finally executes the thread. The processing result is saved into the I/O buffer queue, and then sent to the network under scheduling by the I/O management module. This is the procedure of the web server, which is modeled in the light of queuing network theory [6]. The procedure is shown in Fig. 2.

Obviously, the web server is a service application including multiple threads. To improve the service performance in multi-core web servers, the scheduling strategy of multiple threads must take into account the characteristics of the multi-core CPUs. If there are multiple threads in the multi-core system, the O/S will usually assign these threads to different cores due to consideration of performance improvement and load balance [7]. But, in some cases, this would not result in good performance. Generally, when a thread is running, the O/S will transfer its data from main memory or the L2 cache to its private L1 cache. If both threads have shared data and the O/S assigns the threads to different cores, the O/S will continually transmit the shared data of the threads back and forth between the private L1 caches of the cores during the execution of the threads. This is the ping-pong effect, which will greatly degrade the performance of the multi-core system. Usually, there are a lot of dynamic requests that ask for the same dynamic page in a

* Corresponding author.
E-mail address: guohua.yau@gmail.com (G. You).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.07.006

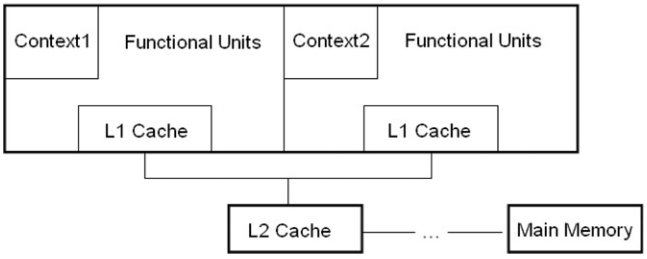

Fig. 1. The architecture of multi-core CPUs.

multi-core web server. When the threads that deal with these dynamic requests are assigned to different cores, this easily gives rise to the ping-pong effect.

Another factor that influences the performance of a web server is the scheduling strategy of requests. The requests of a web server usually are of two types.

Static requests: these request a file (including media files) from the web server.

Dynamic requests: these request some kind of processing from the server web. The processing is usually programmed in the server using a script language (JSP, ASP, etc.) and the result is usually a generated dynamic page [8].

Usually, the processing of the static requests is simple. The procedure has two steps: reading a file from a disk or cache and transferring it through the network interface. The disk and network resources are the main bottlenecks for this type of web object. The service time of the static request is proportional to the size of the file [9]. But the processing of the dynamic requests is complicated. Many dynamic requests include some personalized information (such as location and personal data), so the contents of the dynamic requests cannot be known in advance and must be retrieved from the web servers. They must be generated dynamically each time and cannot be fully cached [10]. In general, many dynamic requests are very simple, and they do not require intensive server resources, such as sum of bill items, but some dynamic requests are very complex, and they require intensive use of web server resources, such as the content of an e-commerce secure site which require Secure Socket Layer Protocol (SSL) processing with intensive CPU use. So the service time of the dynamic requests differs greatly, and usually is a heavy-tailed distribution [8]. In the present paper, we mainly discuss dynamic requests.

Many request scheduling strategies have been proposed. Cherkasova [11] proposed using shortest-job first (SJF) scheduling for static requests. In 1998, the α scheduling strategy was developed at HP Labs [11]. Schroeder and Harchol-Balter [12] demonstrated an additional benefit using the shortest remaining processing time (SRPT) for static requests. Elnikety et al. [13] proposed preferential scheduling for dynamic requests in a transparent fashion. And they addressed the starvation question by using an aging mechanism to prevent starvation.

Actually, most servers, e.g. Apache [14], employ the first-come–first-served (FCFS) strategy. FCFS is fair and starvation free [15], but it is a traditional system-centric scheduling approach [16] and it cannot consider the characteristics of multi-core web servers and the distribution of the dynamic request service times. As a consequence, we propose a new dynamic request scheduling approach in a multi-core web server, which fully considers the distribution of the dynamic request service times and the characteristics of the multi-core web server, and improves the performance of the multi-core web server in an efficient manner.

The remainder of the paper is organized as follows. Section 2 introduces the related work. The new dynamic request scheduling approach is described in Section 3. Section 4 introduces the simulation experiments of the new approach and presents an evaluation of the performance. And finally, we present our conclusions and future work in Section 5.

2. Related work

2.1. Request scheduling strategy

In this section, some proposed request scheduling strategies are introduced. FCFS is a fair strategy, but SJF and SRPT have shorter average waiting time [17]. In addition, α scheduling and weighted fair queuing are also introduced.

(1) First-come–first-served (FCFS): In the FCFS strategy, requests are handled in the sequence of their arrival time. FCFS is fair, but it takes a long time to process the large files that are newly arriving. As a result, the overall average waiting time increases.

(2) Shortest-job first (SJF): In SJF, requests with small service time have precedence over requests with longer service time. In this way, the overall mean waiting time is reduced. However, SJF needs to know the service time of requests beforehand. Because requests with longer service time have lower priority, there will be starvation on long-term heavily loaded web servers, i.e., when there are many requests for small files in the web server.

(3) Shortest remaining processing time (SRPT) [18]: In SRPT, the request with the least remaining processing time is scheduled and processed with precedence over requests with a longer processing time. Each request is divided into sub-requests, only the first of which is scheduled at the request arrival time. The next sub-request is qualified to be scheduled only if its previous request has been completed. This scheme approximates round-robin scheduling. Like SJF, SRPT unfairly penalizes requests with longer processing time in order to give priority to requests with shorter remaining processing time.

(4) α scheduling: α scheduling is a scheduling strategy that is adjustable between (fair and starvation-free) FCFS and

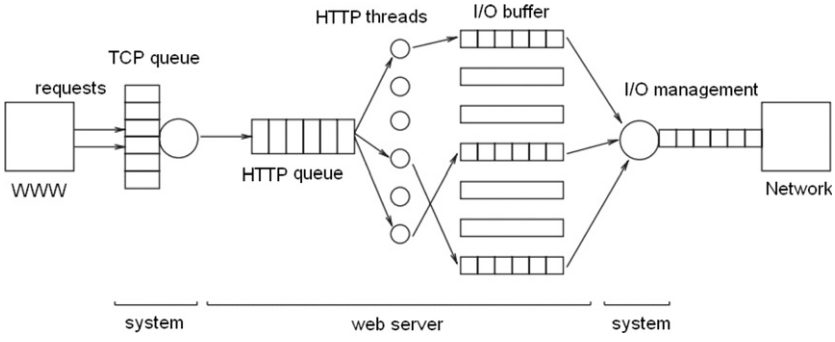

Fig. 2. Basic model of a web server [6].