

# Development of a Kernel Thread Web Accelerator

Jonggyu Park, Hanna Lim, Soungjun Ko, and Hagbae Kim  
 Department of Electrical and Electronic Engineering, Yonsei University, Seoul, Korea  
 (Tel: 82-2-2123-2778 ; Fax : 82-2-362-2780 ; E-mail : hbkim@yonsei.ac.kr)

## 1. Introduction

In this paper, we suggest a kernel level multi-thread web accelerator (called the SCALA-AX), which significantly improves the performance of the web server. Specifically, the SCALA-AX runs on the kernel level of a web server and is based on the newest caching techniques. Moreover, the SCALA-AX supports the HTTP 1.1 protocol and allows the dynamic pages as well as static pages to be processed[1].

## 2. Architecture of the SCALA-AX

The SCALA-AX can be applied for any linux platform over 2.4.0 kernel version without complicated reconfiguration, since it is designed in a kernel module. Fig. 1 shows the generic structure of the SCALA-AX. It consists of two parts that are the interrupt part and the kernel part according to the location where to be implemented. The interrupt part has the handler of the SCALA-AX and intercepts the input packets of a TCP layer.

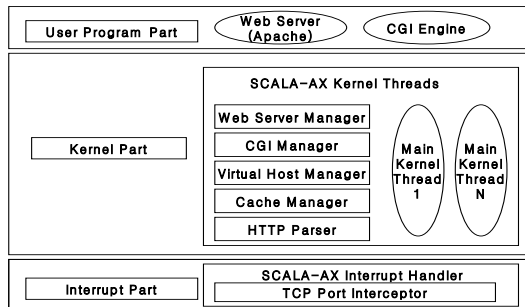


Fig. 1 Structure of the SCALA-AX

The kernel part consists of five components. These are summarized by (i)the http parser that analyses the http request and performs the main threads that are the same number as the CPU, (ii)cache manager that controls the requested data on the memory, (iii)virtual host manager that supports the virtual hosting method, (iv)web server manager that administers a web server of a user program part, and (v)CGI manager that allows the CGI to be treated on the kernel level. Moreover, it operates together with the web server of the user program part.

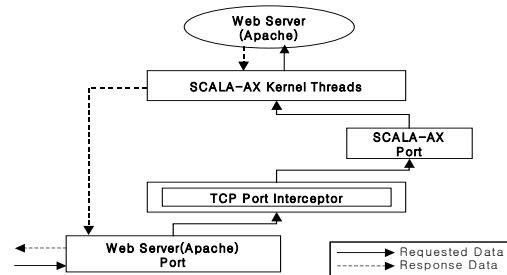


Fig. 2 Data transmission of the SCALA-AX

Fig. 2 shows generic data transmission in the SCALA-AX. A client sends the requests to the TCP 80 port generally used for a web services. The TCP interceptor of the interrupt part snatches the requests and then transfers it to the TCP port which has been already opened by the SCALA-AX. After the SCALA-AX takes correct classification and immediate action, it responds to a client instead of the apache web server. Through this process, the SCALA-AX is able to handle all clients' requests. In some cases, the SCALA-AX makes over the requests that cannot be treated by the SCALA-AX to the apache web server, but the response is completed by the SCALA-AX. The SCALA-AX focuses on thoroughly transparent behaviors to the apache web server, while handling those requests ahead of the apache web server.

## 3. Technical Features

The SCALA-AX is physically integrated in the linux kernel of the web server to reduce the functional overheads of the application level processes. Moreover, the SCALA-AX brings services directly to a client from a cache, which stores the cached pages without retrieving from the web server disk[2,3]. Fig. 3 shows the utilized caching algorithm. When the requests were transmitted into a server, the SCALA-AX distinguishes whether the response data can be stored to the memory or not. Then if it is possible, the SCALA-AX confirms whether it has been cached or not. For the cached data, the SCALA-AX serves to the client directly using that information. However, for the uncached data, the SCALA-AX accesses the disk and caches the data to the caching table.

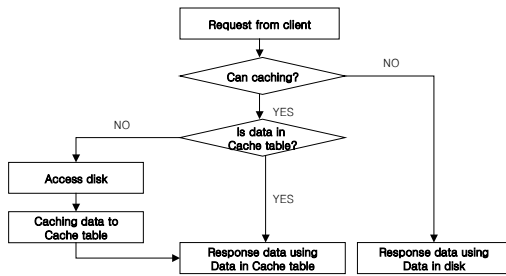


Fig. 3 Caching algorithm of the SCALA-AX

The SCALA-AX supports persistent connections and requested pipeline of the HTTP 1.1 protocol completely. Therefore, it reduces the latency and the bandwidth incurred from TCP connection. As mentioned earlier, since the SCALA-AX is built to be embedded in the web server, the caching server placement and the content-freshness problem can be avoided[4]. In addition, the SCALA-AX processes dynamic pages as well as static pages to reduce the accessing time significantly, because it has an embedded CGI API. Since, most processes are executed in kernel level, the SCALA-AX improves the processing speed of the CGI data as well. The SCALA-AX also supports a virtual hosting which is to manage multiple hosts in the same machine. Furthermore, it is able to extend the host name and support the binary format generating log file in order to minimize both time and space. This format requires narrower I/O bandwidth and less disk space than conventional W3C ASCII formats. Therefore, the SCALA-AX makes the log file less than 50 percent of the case using the ASCII format.

#### 4. Performance Evaluation

To validate the effectiveness of the SCALA-AX, we conducted a performance evaluation using the ZDNet's WebBench 3.0 on a test bench as follows. A server, which has dual Pentium III 800Mhz processors and 512MB RAM, provides the web service for the requests of the 24 Pentium II processor PCs as the clients. To eliminate the effects of network congestion, we adopt a gigabit ethernet environment.

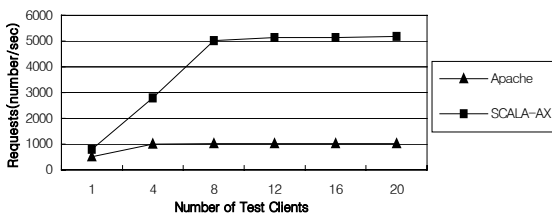


Fig. 4 Request of the SCALA-AX and the Apache

Fig. 4 shows the evaluation results of the web server with and without SCALA-AX, respectively. Without the SCALA-AX, the server responses about 1000 requests per second. However, with the SCALA-AX, it can handle about 5000 requests per second to achieve about five times improved performance.

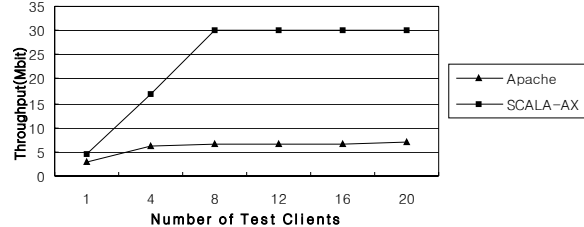


Fig. 5 Throughput of the SCALA-AX and the Apache

Fig. 5 again shows the performance enhancement of the server which contains the SCALA-AX compared to the server without it. The web server without the SCALA-AX faces the saturation of throughput growth around 5Mbit/sec. However, the web server embedded with the SCALA-AX has the maximum throughput around 30Mbit/sec.

#### 5. Conclusion

We developed a kernel thread web accelerator called the SCALA-AX which significantly maximizes web server performance without upgrading H/W components or a bandwidth. Migrating web server functionality into the kernel may have such risks as difficult kernel handling tasks and complicated debug, since the kernel functions are tightly interrelated. However, the benchmarking results validate that the SCALA-AX optimizes the service speed of the web server, saves the bandwidth, and presents faster response time to client by reducing the server overhead.

#### References

- [1] J. Challenger, A. Iyengar, and P. Dantzig, "A scalable system for consistently caching synamic web data," Proc. of the IEEE INFOCOM'99, March 1999
- [2] P. Cao and S. Irani, "Cost-aware WWW proxy caching algorithms," Proc. of the USENIX Symposium on Internet Technologies and Systems, December 1997.
- [3] A. Chankhunthod et al, "A hierarchical internet object cache," Proc. of the 1996 USENIX Technical Conference, pp 153-163, January 1996.
- [4] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache : a scalable wide-area web cache sharing protocol," SIGCOMM'98, 1998.