Design and Performance Analysis of Web Caching with Integrated File System on FPGA

Xue Kong, Yongxin Zhu, Xuetao Yu, Tian Huang, Gang Cheng, Jibo Yu
{kongxue, zhuyongxin, yuxuetao, huangtian, chenggang, yujibo}@ic.sjtu.edu.cn
School of Microelectronics
Shanghai Jiao Tong University
Shanghai, China

Abstract- **In the cloud computing era, internet resources are becoming ample and abundant; hence the rising gap between remote application users and the virtualization resources. Considerable efforts have been invested in distributed file system and web caching with different optimizing strategies to shrink and ultimately eliminate the gap, but few existing studies have combined the two together to enhance the performance of the web server systems. In this paper, we make a solid effort to reveal the feasibility of the integration. After challenging ourselves with significant difficulties in design, we manage to prototype an evaluation system which could bring higher throughput, lower response delay as well as integrated functionalities of file system, web caching mechanism and web services. Our experience in exploring the method of web accelerations will be taken as useful reference for researchers working on high performance web services and FPGA applications.**

Key words: cloud computing; architecture; web server; file system; web caching; FPGA; internet technology

I. INTRODUCTION

With the rapidly development of network services, the data volume of the WEB application systems increase unceasingly, while the internet users put forward higher request to data throughput rate and stability. With the advent of the cloud computing, this requirement could be even more urgent. Therefore, how to effectively balance the requirements of the application users and the internet source is a most compelling problem to Network Operators. Many intelligent ideas emerge as the times require to solving the problem, including web caching method and distribute file system.

Web caching has been recognized as an effective approach to improve performance by storing frequently-accessed documents from originating servers at storage sites “closer” to requesting clients in order to reduce the load on the network bandwidth which in turn can reduce user response time [1]. Web caching techniques become more important to meet the ever increasing internet service demands. With web caching techniques, we can reduce the response time and workload of the web server - the user request is processed on the internal memory and local area network which results in a workload reduction to the web server and a traffic reduction on the network [2]. The objective of web caching is to make efficient use of internet resources and to increase the speed of the delivery of content to the end application users.

Meanwhile, file system is a means to organize data expected to be retained after a program terminates by providing procedures to store, retrieve and update data, as well as manage the available space on the device(s) which contain it. A file system organizes data in an efficient manner and is tuned to the specific characteristics of the device [3]. The primary role of the file system is to organize the sequential fixed-size disk sectors into a collection of variable-sized files [4]. File system are typically implemented in software as fundamental part of the operating system running on the processor. In the network application cases, the importance of file system is ever more. Distributed File System is a key component of web server that is designed to store mass data on commodity hardware with highly access bandwidth across the cluster.

Considerable efforts have been invested in studying web caching and distributed file system respectively. Many interesting and effective approaches have been proposed and implemented. According to our survey about related work, previous studies mainly include improving web caching mechanism, presenting novel architecture or algorithm of web caching, and modified file system. On the other hand, less attention has been paid to develop an application specific file system with web caching mechanism used for web server systems. In consideration of there is a limit of improving the performance of one aspect only, we try to combined the two together to improve the performance of the web server systems.

As our previous efforts have primarily concerned about the architecture of the web server system, which is a hardware-based architecture of web server named Web Processing Module (WPM) [5], we would illustrate our novel file management system method with web caching based on the architecture in this paper.

WPM is a hardware module which implements MAC, TCP/IP and HTTP protocol instead of traditional TCP/IP protocol stack. WPM consists of five sub modules, namely TCP packet decomposer, URL parser, file splitter, TCP/IP processing and Timing service. Since HTTP GET is a typical HTTP request from Web clients, it is selected as the only request type we process in the prototype design [5]. In consideration of the distinctiveness and innovativeness of WPM, a novel and high efficiency file system and web caching mechanism is needed to correspond with it, and then we propose a HW/SW co-design File Management System

Unit (FMSU). There are two important partitions in FMSU. The software partition is primarily consisted of an Application Specific Embedded File System (ASEFS), while the hardware partition is basically composed of Hardware Transfer Unit. The software partition is mainly managed on the Application Layer, and the hardware partition is mainly managed on the Physical Layer. With the advantages of both, the web server system could provide a rapid and stable web access to users.

The rest of this paper is organized as follows. In the next section, a brief description of the related work about distributed file system and web caching mechanism. In Section III, the main architecture of the proposed web server system is described. In Section IV, its functionality, HW/SW portion and interfaces are presented. The performance analysis is reported in Section V. The paper concludes with a brief summary and future direction in Section VI.

II. RELATED WORK

Cloud computing is quickly becoming one of the most popular and trendy phrases being tossed around in today's technology world [6]. With the advent of cloud computing, many more web based applications will be developed. And the workload of web servers will grow rapidly which may result in client's perceived latency for a web page request [5]. In order to meet the rapidly growing demands of data processing needs, Google designed and implemented the Google File System (GFS), a scalable distributed file system for large distributed data-intensive applications. KFS (Kosmos File System) [7] and HDFS (Hadoop Distribute File System) [8] is an open-source implementation of GFS [9]. A GFS cluster consists of a single master and multiple chunk servers and is accessed by multiple clients [10]. The distributed file system virtualizes the scattered internet resources as a unified data to provide a consolidated service to internet users.

A number of research projects have approached the task of optimizing the web caching for web server, mainly including improving web caching mechanism, presenting novel architecture or algorithm of web caching. The author of [11] proposes the delayed caching mechanism which uses the delayed caching in order to improve system reliability and provide a quick service to users' service requests. The author of [12] presents a novel architecture for cooperative web caching based on a two-tier cluster-based lookup process. The author of [13] presents an innovative cache replacement algorithm. They implemented their projects through modifications of the SQUID cache server or proxy server.

Though these studies can meet the respective requirements and have their own application of the occasion, we are not aware of any prior work that systematically offer file system support directly to web caching as our work does.

In traditional web caching systems, Proxy server determines the possibility of the document accessed again according to the information of operation and the property of web documents, in order to cache the mostly repeat accessed document as many as possible. However, every web users have their own interests. An important method is to reduce the access delay of the web users' interested content. Based on this theory, we propose an Application Specific Embedded File System (ASEFS) to manage and organize the web content, a Hardware Transfer Unit to achieve the data transferring function in the caching architecture.

III. MAIN ARCHITECTURE

To verify the basic function and performance of FMSU, we embed the FMSU module into WPM. The overall architecture of the web server system with FMSU is shown in Figure 1. A MicroBlaze soft-core processor runs software, which includes the system initialization and the software partition of the FMSU. Web processing module (WPM), what is the hardware partition implementing a simple web server, is connected by a register bank to other system. File Management System Unit (FMSU), what is connected to MicroBlaze and WPM, is mainly in charge of scheduling and composing the web data to the WPM. Both the software and hardware partitions of the system can access the MEM, which include BRAM and DDR RAM, but not limit to this. In future study, Solid State Disk and Hard Disk would be appended to this architecture. The SSD could take the place of BRAM, and the hard disk could take the place of DDR.

Figure 2 shows the work flow of File Management System Unit (FMSU). The software partition includes configure file, initialization and an application specific embedded file system, which runs on MicroBlaze soft-core processor. The hardware partition mainly includes MicroBlaze, WPM, Hardware Transfer module, which could schedule and move web data in one memory or across different memory, and other components.

In order to better improve the performance of the web server system, three modes of File Management System Unit (FMSU) architecture are supported, which includes Local Mode, Pseudo-Cluster Mode, and Cluster Mode. There is an evolutionary relationship among those three modes. Local Mode and Pseudo-Cluster Mode runs on one chip of FPGA, while Cluster Mode could run on 2 chips of FPGA or more. In Local Mode, the web data and metadata are all stored in DDR RAM, while in Pseudo-Cluster Mode, the web data could be moved to Block RAM. In Cluster Mode, web data and metadata are distributed in the DDR RAM on the platform and each WPM could accesses the web data slices at other web server systems' DDR channel. Figure 3 show the 3 modes of FMSU.

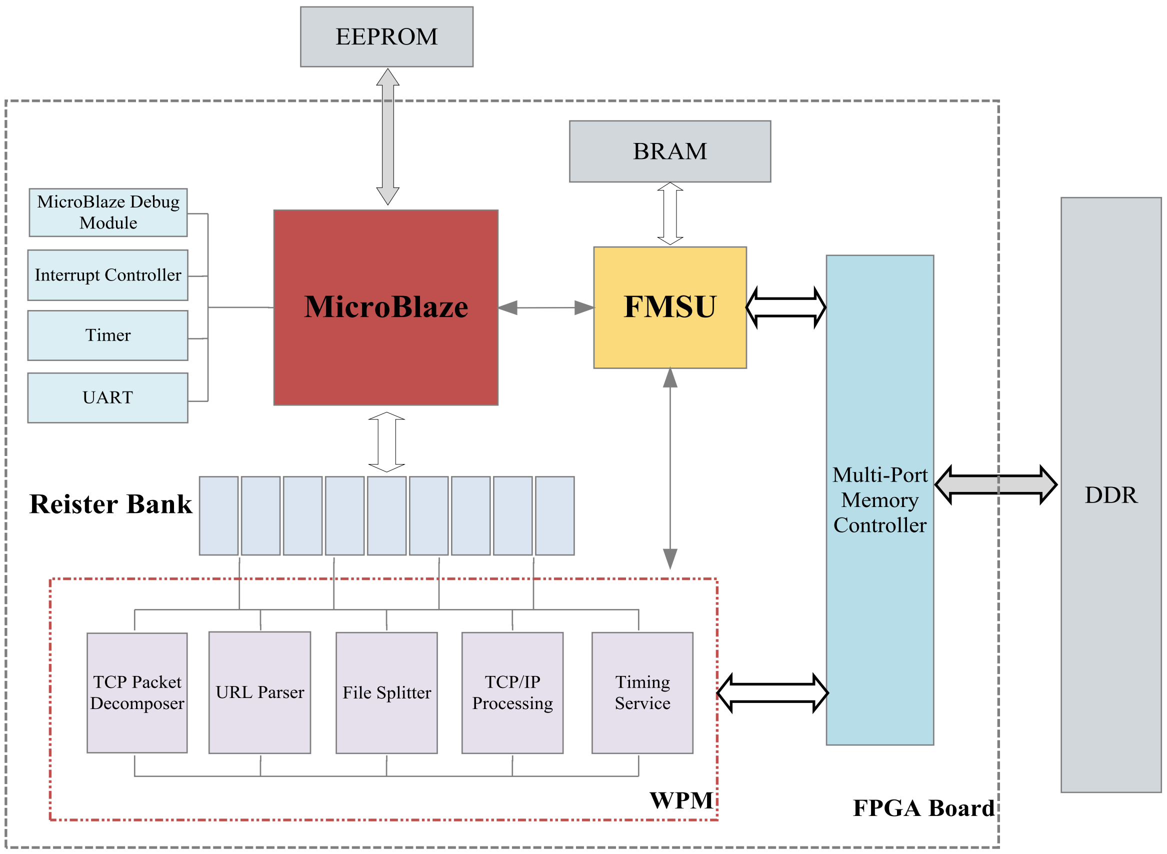


Figure 1. Basic architecture of the web server system

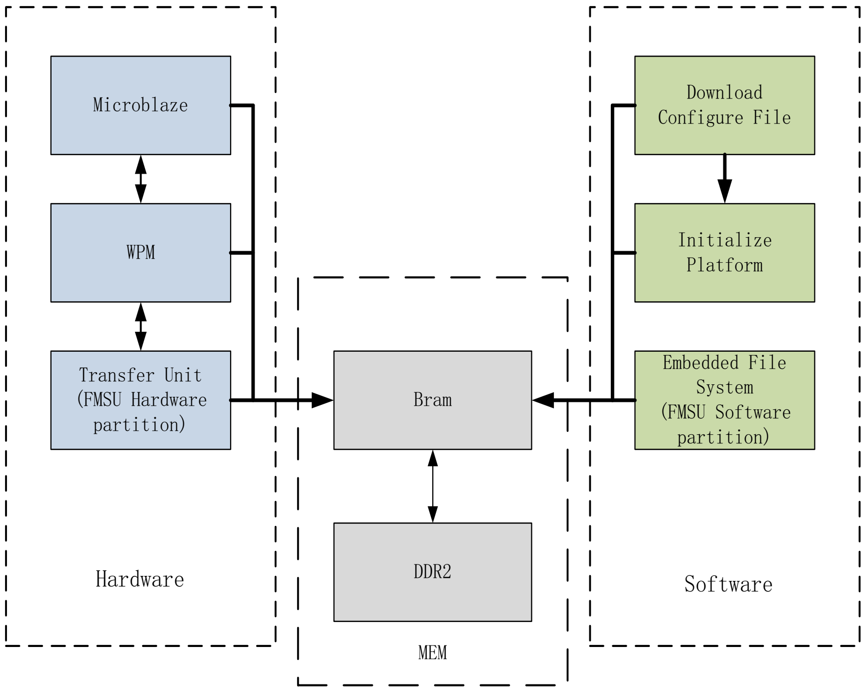
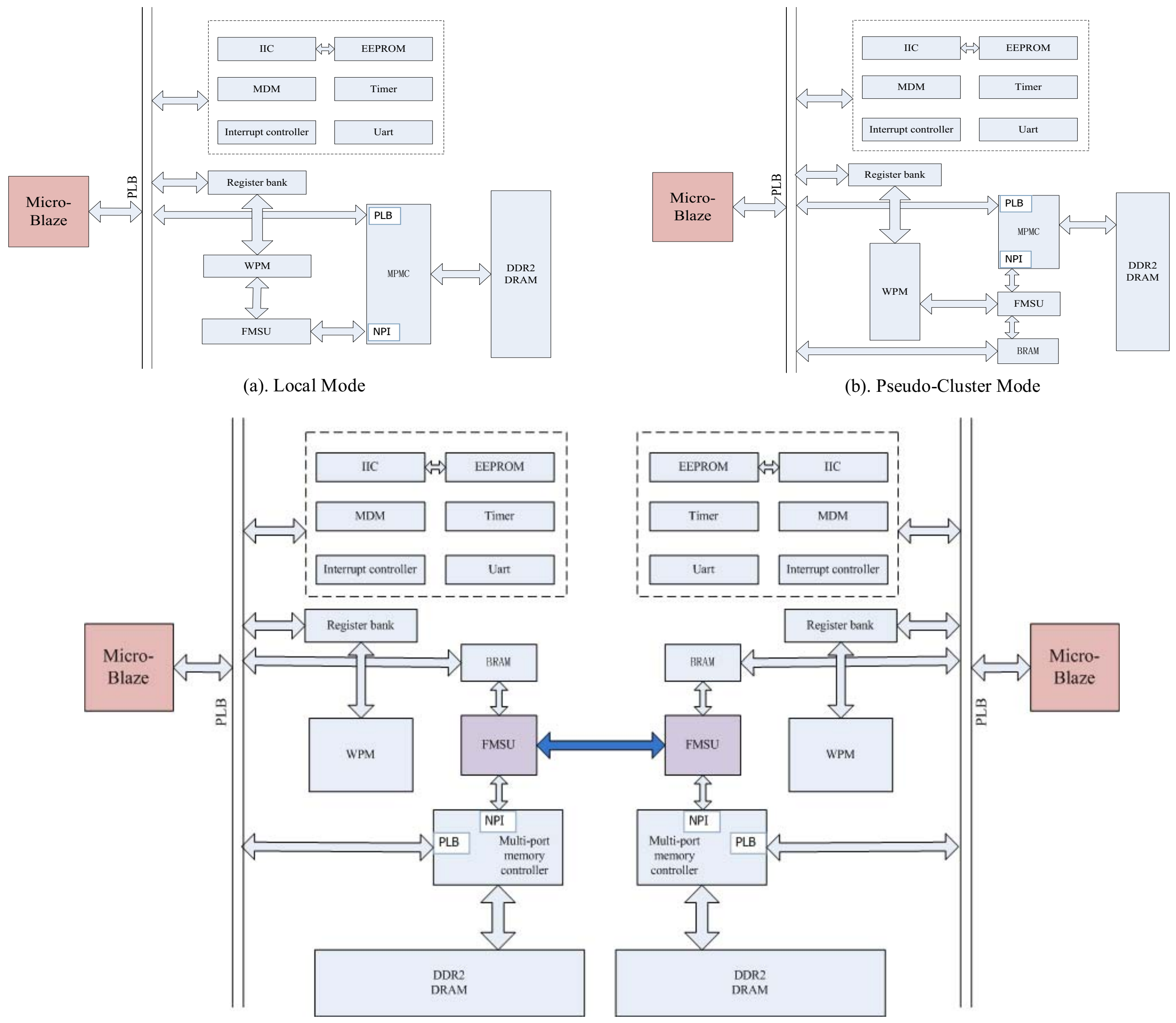

Microblaze
WPM
Transfer Unit (FMSU Hardware partition)
Hardware
Bram
DDR2
MEM
Download Configure File
Initialize Platform
Embedded File System (FMSU Software partition)
Software

Figure 2. Work flow of File Management System Unit (FMSU)



(a). Local Mode

(b). Pseudo-Cluster Mode

(c) Cluster Mode

Figure 3. Three types Work Mode of FMSU

IV. FUNCTIONALITY AND EXPERIMENT

The functions of WPM web server are demonstrated on a BEE3 prototyping board from BEEcube Incorporation. And FMSU could be coupled with WPM tightly on this board. There are four FPGAs on the board, and each of which has two independent DDR2 channels. The other components, such as MicroBlaze Debug Module (MDM), interrupt controller, timer, IIC, UART and MPMC, are IPs provided by Xilinx Embedded Development Kit. A Parallel Cable IV (PC4) JTAG cable, a null modem serial cable and RJ45 cable is also needed. The design tool we used is Xilinx ISE Design Suite 12.2 [14].

A. Application Specific Embedded File System (ASEFS)

This Application Specific Embedded File System (ASEFS) is responsible for extracting pure web data from MFS files, managing and organizing web content on a nonvolatile storage medium, such as Block RAM, DDR RAM, SSD and SATA hard disk. As files stored in RAM/ROM/Flash memory are in the Xilinx® Memory File System (MFS) format, we should extract the pure useful web data from the MFS files first. When compared to the general meaning of the file system, ASEFS is independent outside in the operating system. So this file management system has a relatively high independence and high efficiency. The main function of ASEFS is dissecting the MFS raw files, dividing web content data into small slices, and storing the relevant content slices in one memory or different memory. The slices contain the partial web content data and the number and size of the slices could be set preliminary. The metadata of the web content is stored in the same memory or other else. When web users launch the access request, ASEFS could provide the assembled web content to WPM.

The web data moving based on a data shifting mechanism, which could judge the HotFiles and ColdFiles. A dedicated log file is responsible for recording the web content visits. The ASEFS will judge the frequently visited content as HotFiles (such as TOP 10), conversely ColdFiles, dividing the HotFiles content data into some small slices, and then shifting the HotFiles slices to BRAM as many as possible. When needs to moving data slices, the ASFES would trigger the Hardware Transfer Unit to complete the shifting task.

B. Hardware Transfer Unit

The components in the hardware partition of this system are as follows: IIC to collect local time information from EEPROM, MDM to provide a debug interface for software, Interrupt controller to receive interrupt signals from timer and UART and send a interrupt request to MicroBlaze, Timer to provide an interrupt signal every one second, UART to communicate between FPGA and the host computer, MPMC to provide interfaces for accessing DRAM, Transfer unit to move data in or across memory. The Transfer unit consists of controlling State Machine and a Datapath. The state machine achieves the data moving operations. The datapath consists of Block RAMs, DDR RAM. Block RAMs or DDR RAM is used as buffers for storing web data and metadata. In future research, Solid State Disk and SATA Hard Disk could be appended to hardware architecture seamlessly. The BRAM could be replaced by SSD as a main buffer in the caching architecture, and the DDR could be replaced by SATA disk as the main documents storage.

C. Interfaces

In the software partition, the Application Specific Embedded File System (ASEFS) is running on the embedded processor MicroBlaze. There are dedicated APIs for system software to call, such as *r_fsinit, r_fopen, dev_read, dev_write*. The software partition can access BRAM and DDR RAM via Processor Local Bus (PLB) interface.

The hardware partition can access DDR RAM via Native Port Interface (NPI) through MPMC. We expand an exclusive NPI port to store and load web page file data from DDR RAM. Each FPGA on BEE3 board is allowed to enable two independent DDR2 channels, so we can use both of them to access DDR RAM simultaneously. The hardware partition connects BRAM directly. MDM, interrupt controller, timer, IIC, UART and the register bank are all connected by MicroBlaze via PLB.

V. PERFORMANCE EVALUATION

A. Experimental Setup

To test the timing and performance metrics described above, we used ModelSim and Xilinx ISE for simulation and synthesis purposes. The Xilinx Synthesis Tool (XST) manual served as a guideline for writing synthesizable Verilog code. The Verilog design description was synthesized using the XST available in the Xilinx ISE design suite, version 12.2, for the target device XC5VLX155T-2ff1136 from the Virtex-5 family to generate the Xilinx specific NGC files. The software application is debugged by Xilinx EDK, version 12.2, on Linux environment. ModelSim verification environment, version 6.3c [15], running on a Linux Workstation was used for simulation and debugging. The web server system is evaluated with web test equipment, i.e. Avalanche 2900 and Spirent TestCenter 3.51. The tests mainly focus on the minimum TCP response time of the system. The response time of web server refers to the span from the client launches a visit request to the client receives a return of the web server. Results are compared with Apache2.2 and Nginx0.7.61 which are run on a main stream quad-core processor: Intel Xeon 5520. All the web pages under tests are present in the DDR memory of the testing system as well as the main memory of the reference Xeon 5520 platform.

B. Results and Performance Analysis

The WPM web server system implemented on FPGA V5 could work at 125MHz. The speed of the physical Ethernet port on the FPGA board is 1 Gbps. As the prototype web server does not connect SSD and SATA disk to BEE3 platform, we would give a reasonable performance deduction of FMSU based on the hand tested data under suitable assumptions.

Figure.4 indicates WPM has a substantially short minimum TCP response time than other systems. Apache and Nginx do not both necessarily shorten the response time, especially when the size of web page is becoming bigger.

Researchers have found in the log file of proxy web server over 10% accesses are “not modified” (304 status code), which means if the client has done a conditional GET and access is allowed, but the document has not been modified since the date and time specified in If-Modified-Since field, the server responds with a 304 status code and does not send the document body to the client. Proxy server can be designed to allow for a distribution model to validate cache contents and hence “conditional Get” will not be necessary [16]. In FMSU, web data and metadata are distributed in the storage on the platform and each WPM could accesses the web data slices. Assuming these “conditional Get” takes 10% of response time, we can shorten the response time by at least 10% compared to WPM if we redesign the proxy server to get rid of “conditional Get” requests with integrated FMSU based on the distribution model.

Meanwhile, as shown in the formula [17] below the delay of a packet in a limited-bandwidth network pipe model between a client and the server, is computed as the sum of three components:

$$\text{delay}_{\text{packet}} = \text{latency}_{\text{I/O}} + \frac{\text{size}_{\text{packet}}}{\text{bandwidth}_{\text{path}}} + \text{latency}_{\text{path}} \quad (1)$$

The transmission route between client and server is internet path. The first component in the formula is the I/O latency of the disk when web server read the data on the hard disk. The second is the transmission delay, which models the time required to transfer a packet given a certain path bandwidth. The last is a propagation delay, which models the latency of the path.

We assume the performance of FMSU is good enough and the local BRAM could buffer the whole web data. On Xilinx FPGA, there is 1 cycle latency when BUS accesses the BRAM. But in the case of DDR accesses, the access latency could be much higher. According to the user guide of Multi-Port Memory Controller (MPMC), the latency is uncertain and relates to the actual situation. In common actual practical applications, the DDR latency is usually about 10-30 cycles as we debug by grabbing the data on the BUS through Xilinx ChipScope Pro. Therefore the I/O latency of FMSU could be reduced about 1/10 of WPM, assuming FMSU with whole data buffered.

FMSU divides web content data into small slices, and the size of web data packet is reduced. So the transmission delay could be shortened at the similar bandwidth environment. But web data could not be divided unlimitedly, because the propagation delay could be increased if the number of slices is too much. Assuming the number of web data is divided into an optimal amount, the transmission delay would be linear reduced at some extent.

On these premises, we could deduce that the response time of FMSU with whole data buffered can be reduced by 10-20 percent in the WPM basis. For example, when the web page size is 100KB, the access response time of WPM is 0.024ms as we test. In a FMSU with whole data buffered, the time could be 0.02ms with a 15% reduction through our analysis above. The performance of FMSU could have a definite enhancement when the number of slices is critical and the work module is matching.

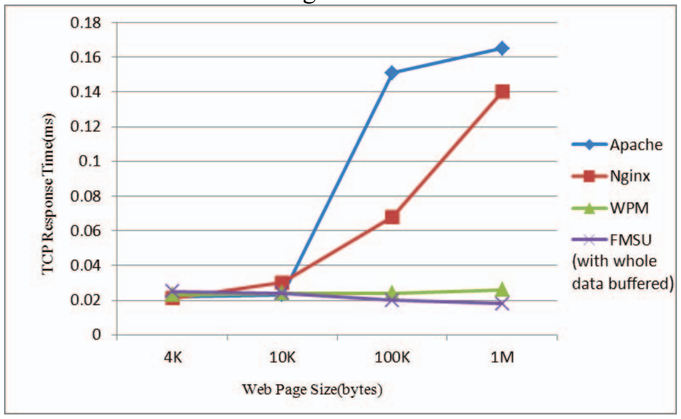


Figure 4. Minimum TCP Response Time of different systems

VI. CONCLUSION AND FUTURE RESEARCH

We present the design and performance analysis of a file management system tightly coupled with web caching mechanisms on a FPGA based web server. We evaluate the feasibility of integrating a file management system and web caching mechanisms in a web server through hardware and software co-design. The novel architecture proposed and evaluated in this project attempts to avoid the web data request bottleneck by giving the WPM hardware-based web server on FPGA the opportunity to bypass the operating system and MPMC entirely. It enables the WPM to get direct, high bandwidth access to large data sets. Finally, the web users could get the web resources more quickly and smoothly than before.

The design currently supports BRAM and DDR2 on the BEE3 platform and the Embedded File System partition works sequential accessing and scheduling to web page in or across memory. It can be enhanced to incorporate random access and schedule by adding new operation in the Embedded File System partition. It can also be extended to a full system with SSD and SATA disk array or other physical storage devices.

This design will serve as an important first step to develop and evaluate a parallel web server or CDN server which will co-ordinate file access from multiple disks across the cluster and present a closely “web resource image” for internet users.

ACKNOWLEDGMENT

This paper is partially sponsored by the National High-Technology Research and Development Program of China (863 Program) (No. 2009AA012201) and the Shanghai International Science and Technology Collaboration Program (09540701900).

REFERENCES

[1] R.T. Hurley and B.Y. Li, “A Performance Investigation of Web Caching Architectures,” Proc. C3S2E Conf., pp. 205-213, 2008.

[2] Van Jacobson, “How to kill the Internet” in SIGCOM’95 Middleware Workshop, August 1995.

[3] Wikipedia, en.wikipedia.org/wiki/File_system.

[4] A. A. Mendon and R. Sass, “A hardware filesystem implementation for highspeed secondary storage,” in 2008 IEEE International Conference on Reconfigurable Computing and FPGA’s, 2008.

[5] Jibo Yu, Yongxin Zhu, Liang Xia, Meikang Qiu, Yuzhuo Fu, Guoguang Rong, “Grounding High Efficiency Cloud Computing Architecture: HW-SW Co-Design and Implementation of a Stand-alone Web Server on FPGA”, Proc. of the Fourth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2011),Wisconsin, U.S.A., pp. 124-129, 2011.

[6] F. Hu, M. Qiu, J. Li, T. Grant, D. Tyloy, S. McCaleb, L. Butler, and R. Hamner, "A Review on Cloud Computing: Design Challenges in Architecture and Security", Journal of Computing and Information Technology (CIT), Vol. 19, No. 1, Page 25-55, Mar. 2011.

[7] KFS, kosmosfs.sourceforge.net

[8] HDFS, http://hadoop.apache.org/hdfs/

[9] Laboratory of Cryptology and Computer Security, http://loccs.sjtu.edu.cn/drupal/node/166

[10] S. Ghemawat, H. Gobioff, S. Leung, “The Google file system,” In Proc. of ACM Symposium on Operating Systems Principles, Lake George, NY, Oct 2003, pp 29–43.

[11] Daesung Lee, Kim, K.J., “A Study on Improving Web Cache Server Performance Using Delayed Caching”, Information Science and Applications (ICISA), 2010, 21-23 April 2010.

[12] Riccardo Lancellotti , Bruno Ciciani , Michele Colajanni, A Scalable Architecture for Cooperative Web Caching, Revised Papers from the NETWORKING 2002 Workshops on Web Engineering and Peer-to-Peer Computing, p.29-41, May 19-24, 2002.

[13] W.G. Teng, C.Y. Chang and M.S. Chen, Integrating Web caching and Web prefetching in client-side proxies. IEEE Trans Parallel Distributed Syst, 16 5 (2005), pp. 444–455.

[14] Xilinx, www.xilinx.com/

[15] ModelSim, www. http://model.com/

[16] B. Liu. Characterizing web response time. Master's thesis,Virginia State University, Blacksburg, Virginia, USA, April 1998.

[17] G. Banga and P. Druschel. Measuring the capacity of a Web server under realistic loads. World Wide Web Journal (Special Issue on World Wide Web Characterization and Performance Evaluation), 1999.