Received 19 Mars 2023 | Revised 20 May 2023 | Accepted 25 May 2023
Vol. 04, No. 01, 08 pages

RESEARCH ARTICLE Tobruk University Journal of Engineering Sciences

Performance Analysis of Multi-Thread Web Server

Samera H Mahmud Hamad

Computer Department, Omar Al-Mukhtar University, Al Bayda , Libya

samera.hamad@omu.edu.ly

Abstract. A web server is a computer with special software to host web pages and web applications. A computer that provides Web services and pages to intranet and internet users.
A Web server is a program that, using the client/server model and the World Wide Web's Hypertext Transfer Protocol (HTTP), serves the files that form Web pages to Web users (whose computers contain HTTP clients that forward their requests). Client starvation is one of the problems which will affect the response time for the client when issuing a request. A web server serves web pages to clients across the Internet or an Intranet. The web server hosts the pages, scripts, programs, and multimedia files and serves them using HTTP, a protocol designed to send files to web browsers and other protocols. Multithreading is a powerful tool for improving the performance of web servers. It allows a single processor to execute multiple tasks simultaneously, thus increasing the speed and efficiency of the server. By utilizing multiple threads, web servers can process multiple requests at the same time, resulting in faster response times and improved performance. Multithreading also allows web servers to handle more requests at once, which can lead to improved scalability.
This study presents to show the difference between Multi- Thread and None Multi-Thread Systems using a web server program representing the client that generates a request to the server. The Server that responsible for accepting and handling clients' requests. The results analysis and discussions are provided in section 5.

Keywords: Web server, Multi-Thread system (MTS).

1 Introduction

In the past few years, the World Wide Web has experienced phenomenal growth. Not only are millions browsing the Web, but hundreds of new Web sites are added each day [1].
A web server is a computer with special software to host web pages and web applications. A computer that provides Web services and pages to intranet and Internet users. A web server serves web pages to clients across the Internet or an Intranet. The web server hosts the pages, scripts, programs, and multimedia files and serves them using HTTP, a protocol designed to send files to web browsers and other protocols.
Web servers approach this boundary response times to increase suddenly toward infinity, disabling the server; but limiting the server's simultaneous connections prevents this problem [2].
Threads: A piece of code that runs concurrently with other threads. Each thread is a statically ordered sequence of instructions. Threads are being extensively used express concurrency on both single and multiprocessor machines.
Programming a task having multiple threads of control Multithreading or Multithreaded Programming.

Threads can be helpful within servers: Concurrent processing of client requests can reduce the tendency for servers to become a bottleneck. • E.g. one thread can process a client's request while a second thread serving another request waits for disk access to complete.
Multithreading is a process of executing multiple tasks simultaneously on a single processor. It is a way of dividing a single task into multiple threads, which can then be executed in parallel. Each thread is assigned a specific task, and the processor can switch between threads to execute them in parallel. This allows the processor to execute multiple tasks at the same time, resulting in improved performance.
Multithreading can provide a number of benefits to web servers. It can improve the performance of the server by allowing it to process multiple requests at the same time. This can result in faster response times and improved scalability. Additionally, multithreading can help reduce the amount of memory and resources used by the server, resulting in improved efficiency.
Multithreading can also help improve the user experience. By allowing the server to process multiple requests at the same time, users can experience faster response times and improved performance. This can lead to a better overall user experience, as users will be able to access the content they need more quickly.
When implementing multithreading on a web server, it is important to consider the performance and scalability of the server. It is also important to ensure that the code is written in a way that takes advantage of the multithreading capabilities. Additionally, it is important to ensure that the server is configured correctly to support multithreading.
Overall, multithreading can be a powerful tool for improving the performance of web servers. By utilizing multiple threads, web servers can process multiple requests at the same time, resulting in faster response times and improved performance. Additionally, multithreading can help reduce the amount of memory and resources used by the server, resulting in improved efficiency.

2 Web Server Architectures

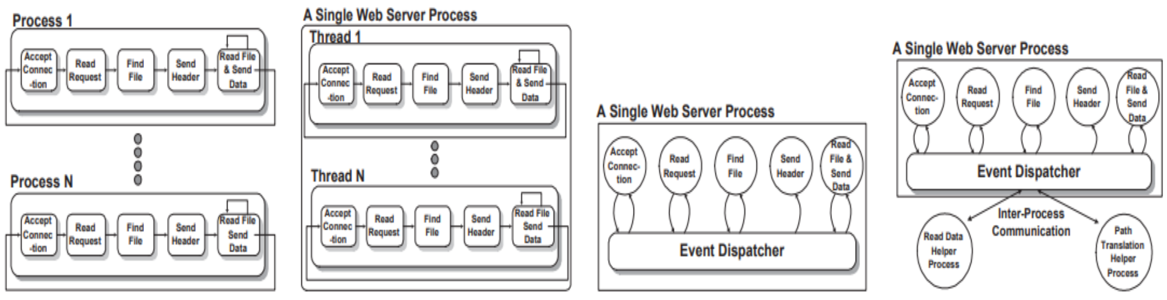
Generally, an HTTP server consists of six request processing steps. The first step is the accept client connection, which accepts an incoming connection from a client based on the socket operations. Second, the read request operation reads and parses an HTTP request from the client's connection. Third, the find file operation checks whether the requested file exists in the file system, and the client has appropriate permissions. Fourth, the send response header step sends an HTTP response header to the client through a socket connection. Next, the read file operation reads the requested data from the file system or from the memory cache. Finally, the send data step transmits the requested content to the client. Especially, for larger files, the read file and send data steps are repeated (shown by the self loop in Fig.1) until all of the requested contents are transmitted [3].

Four HTTP server architectures have been proposed in the literature as shown in Fig.1. The Multi-Process (MP) model, as shown in Fig.1 (a), has a process pool and each process is assigned to execute the basic steps associated with servicing a request. Since multiple processes are employed, many HTTP requests can be served concurrently. However, the disadvantage of this model is the difficulty to share any global information (e.g.: shared cache information) among the processes, since each process has its own private address. An MP-based Web server needs more memory to maintain the same cache size per process compared to other server models. Thus, the overall performance of this model is expected to be lower than that of other models [4, 5].

The Multi-Thread (MT) model, in the other hand, consists of multiple kernel threads with a single shared address space. In Fig.1 (b), each thread takes care of a client's request and performs the request processing steps independently. The advantage of this model is that the threads can share any global information. Especially, the data cache is shared among all threads. However, not all Operating Systems (OSs) support kernel threads, and sharing the data cache information among many threads may lead to high synchronization overhead. The widely used Apache Web server was originally designed as an MP model. Later, it is enhanced to support both MP and MT models, since the MT model tends to yield better performance than the MP model [6].

Next, Fig1 (c) shows a Single-Process Event-Driven (SPED) Web server architecture that uses non-blocking I/O operations. SPED can avoid context-switching and synchronization overheads among threads or processes, because it is a single Web server process. This model is implemented by the Zeus Technology [7]. However, the non-blocking I/O operations in this model are actually blocked [3] when it performs disk-related operations due to the limitations of current OSs. This is the reason why SPED doesn't show better results than the MT model for disk-bound workload [3].

The last architecture is the Asynchronous Multi-Process Event-Driven (AMPED) model [3], which has been proposed to alleviate the weakness of the SPED model. Fig1 (d) depicts the AMPED server architecture, which consists of one main Web server process and multiple helper processes to mainly handle I/O operations. As this model has multiple helper processes to serve disk-oriented requests, the main Web server process only serves cache-hit requests. If there is a cache miss, the main process forwards the request to a helper process, and then, the helper process fetches the data from the disk and sends it back to the main process by Inter-Process Communication (IPC). Especially, using the mmap operation in the AMPED model, additional data copying between a Web server and the helper processes can be removed.



(a) Multi-Process (MP) Model

(b) Multi-Thread (MT) Model

(c) Single-Process Event-Driven (SPED) Model

(d) Asynchronous Multi-Process Event-Driven (AMPED) Model

Fig.1. Web Server Architectures

Table 1, illustrates compute the cache overhead to maintain the cache information in each server architecture when system memory size is known[8].

Table 1: cache overhead in Web Server Architectures

Web Server Architecture	**cache overhead (Bytes)**
Multi-Process Model	file entries × 850 × N × P
Multi-Thread Model	file entries × 850 + T × 25K
Single-Process Event-Driven Model	file entries × 850 × N
Asynchronous Multi-Process Event-Driven Model	file entries × 850 × N + 3M × N

where file entries is the number of cached files, N is the number of processing elements, P is the number of Web server processes per processing element, and T is the total number of threads.

3 Literature Review

Web server vendors are quite happy to extol the performance virtues of their products, and industry professionals abound with theories about how to serve data faster; but these virtues and theories are generally based upon anecdotal evidence, gut instinct, or narrow empirical evidence that has little general utility[2]. Generally, an HTTP server consists of six request-processing steps. The first step is the accept client connection, which accepts an incoming connection from a client based on the socket operations. Second, the read request operation reads and parses an HTTP request from the client’s connection. Third, the find file operation checks whether the requested file exists in the file system, and the client has appropriate permissions. Fourth, the send response header step sends an HTTP response header to the client through a socket connection. Next, the read file operation reads the requested data from the file system or the memory cache. Finally, the send data step transmits the requested content to the client. Especially, for larger files, the read file and send data steps are repeated [3]. In Fig.2, each thread takes care of a client’s request and performs the request processing steps independently. The advantage of this model is that the threads can share any global information[6].

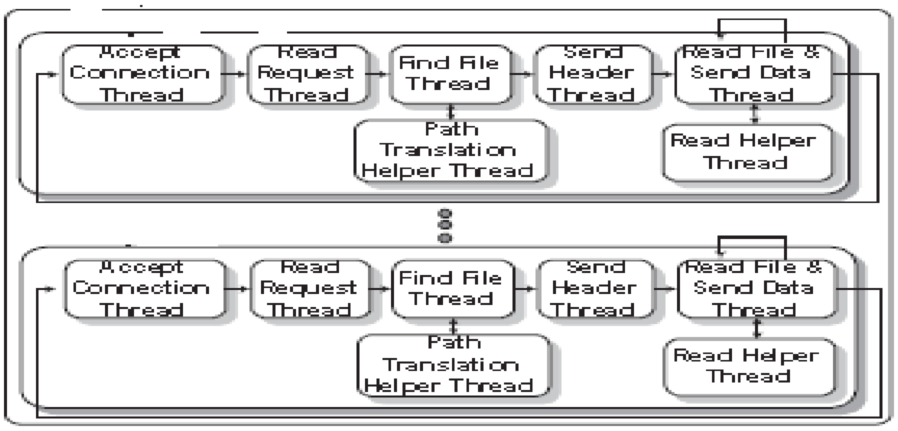


Fig.2. Multi-Thread (MT) Model

Many programming languages e.g. C & C++, Java [9,10] Python have been supporting multi- Thread solutions for a long time. Java has built-in thread support for Multithreading (Synchronization -Thread Scheduling). Java provides built in multithreading . Fig.3 illustrate diagram showing the Life cycle of a thread.
A thread-safe library guarantees to solve the race condition problem even when it is used by multiple threads concurrently [11].

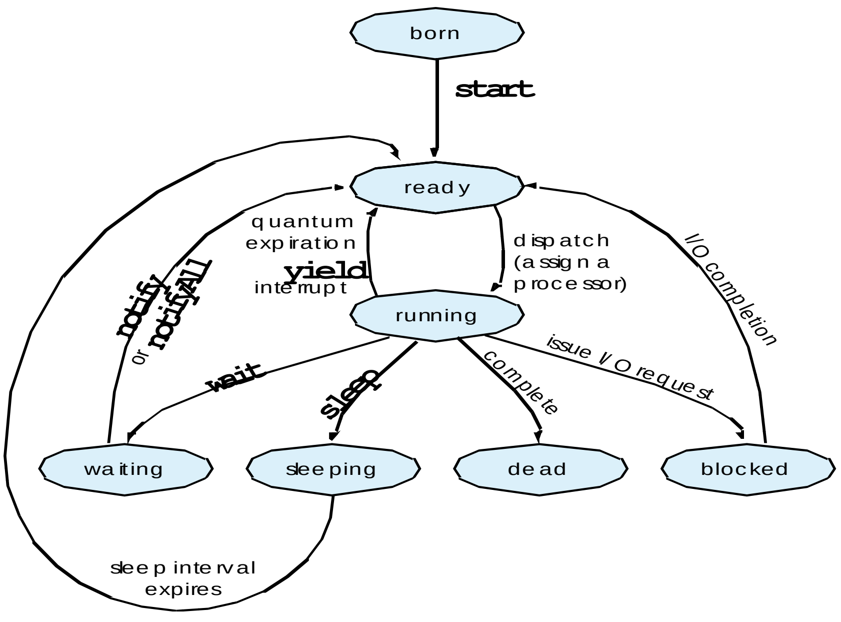


Fig. 3. State diagram showing the Life cycle of a thread

With multi-core CPUs becoming widespread, one way to increase performance is to split the workload between multiple threads and achieve parallel computing [12].

The authors concluded that the best method to determine the ideal number of workers is to use performance monitoring that can detect real time changes in shared resources and if workers are overloading the CPU [13]. Consist of a variety of different Web browsers running on various hardware platforms and connected to the Internet at several different speeds [14,15].

Evaluating and measuring the performance of web servers has been the subject of much recent research. Banga and Druschel [16] proposed a method to generate heavy concurrent traffic that has a temporal behavior similar to that of real Web traffic. They also measured the overload behavior of a web server and throughput under bursty condition. Arlitt and Williamson [17] characterized several aspects of web server workloads such as reference locality, request file type and size distribution. Almeida and Yates [18] provided a tool called Web Monitor to measure activity and resource consumption both within the kernel and in HTTP processes running in user space.

Fundamental to the goal of improving web server performance is a solid understanding of behavior of web servers. A number of performance studies on web server performance have been recently reported in the literature [16,17,18,19].

4 Methodology:

4.1 System Architecture

This system consists of two Multi-Thread programs.
• The First one represents the client that generates a request (50,100,200, and 500) to the server.
• The Server that is responsible for accepting and handling client requests.
• Do the same traffic load on a system without threading.

4.2 Performance Metrics:

4.2.1 **Execution Time** : The Time needed to receive all files requested by the Clients.

4.2.2 **CPU Time**: is the amount of time for which a central processing unit (CPU) was used for processing instructions of a computer program or operating system.

4.2.3 **Response Time:** includes the time taken to transmit the inquiry, process it by the computer, and transmit the response back to the terminal. (It refers to the amount of time the server takes to return the results of a request to the user).

5 Results and Discussion

The performed experiment on a Multi-Thread java application server. Java was one of the first languages to make multithreading easily available to developers. Java had multithreading capabilities from the very beginning.

The results from Fig.4 , Fig.5 and Fig. 6 show the results between (Multi-Thread and None Multi-Thread Systems) represent the client that generates a request (50,100,200, and 500) to the server. In Fig.4 show The Execution Time is expressed in requests per second, and as a function of the simultaneous user sessions within the server when running with different number of requests. also Multi-Thread needs more time for receive all requests.

Noticed how the server's CPU time scale according to the number of available Requests. Results in Fig.5 show when the number of request increases the CPU time increased in both (None Multi Thread than Multi-Thread), also Multi-Thread system needs more CPU time. In Fig.6 show response time: the time taken to attend the client request and send the response to it. Notice that the time unit used to define this measure is the second. Also, and like the above metric, represent the response time for each case, illustrating the performance obtained with different requests. In Fig.6, the response time begins to grow When the number of request increases in both (None Multi-Thread than Multi-Thread).

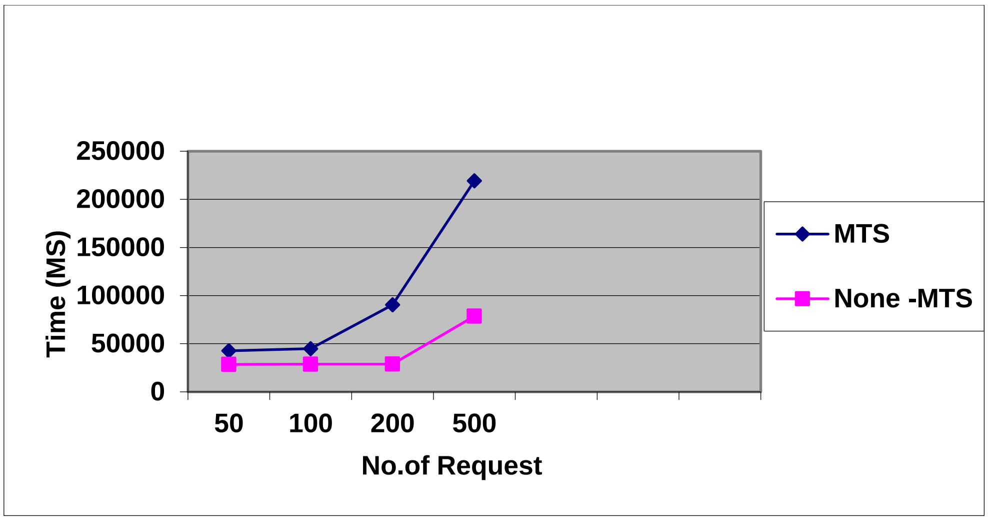


Fig. 4. Execution Time for Multi-Thread and None Multi-Thread

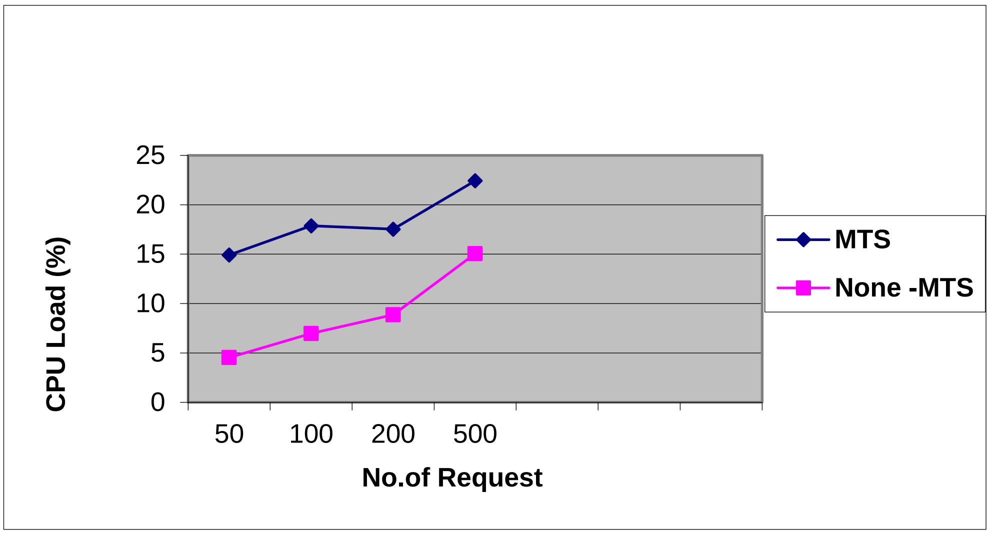


Fig. 5. CPU time for Multi-Thread and None Multi-Thread

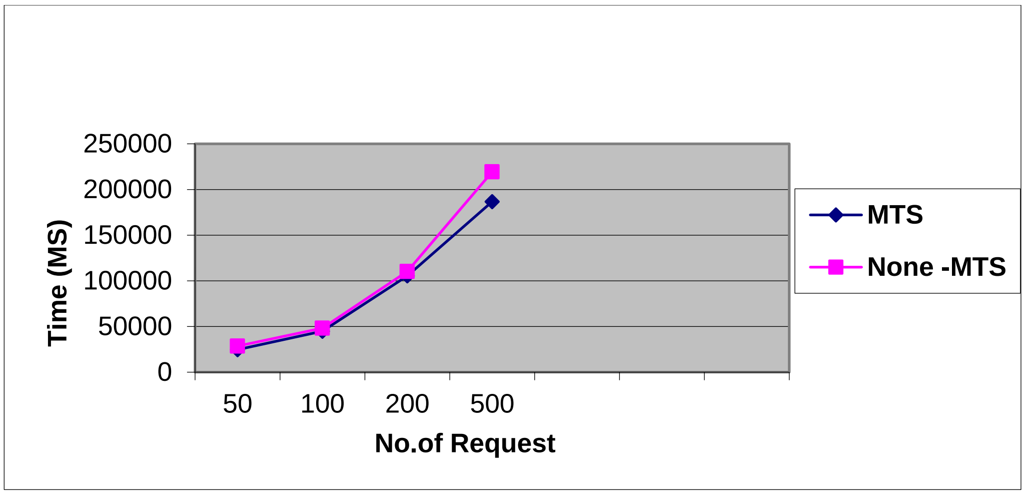


Fig. 6 . Response time for Multi-Thread and None Multi-Thread

6 Conclusion and Future Work6

Above results illustrate for this study can be summarized as follows:

- Multi-Thread system needs more time receive all files requested by the Clients.
- Multi-Thread system needs more CPU time.
- Multi-Thread can improve the response time of the clients .

Future work should focus on using other Performance metrics (e.g. Memory Utilization ,IO Utilization) and test the System on Heterogeneous environment .

Conflict of Interest

This statement is to certify that the author have seen and approved the manuscript being submitted and there is no competing interest.

References

1. Wiederspan, J. and C. Shotton, Planning and Managing Web Sites on the Macintosh, Addison-Wesley, 1996.
2. Louis P. Slothouber," A Model of Web Server Performance", June, 1995.
3. V. Pai, P. Druschel, and W. Zwaenepoel. Flash: An Efficient and Portable Web Server. In Proceedings of the USENIX 99 Annual Technical Conference, June 1999.
4. A. Bestavros, R. L. Carter, M. E. Crovella, C. R. Cunha, A. Heddaya, and S. A. Mirdad. Application-level Document Caching in the Internet. In Proceedings of the 2nd International Workshop on Services in Distributed and Networked Environments, page 166, 1995.
5. E. P. Markatos. Main Memory Caching of Web Documents. In Proceedings of the fifth international World Wide Web conference on Computer networks and ISDN systems, pages 893–905, 1996.
6. The Apache Software Foundation. The Apache HTTP Server Project, 2003. http://httpd.apache.org
7. Zeus Technology Limited. Zeus Web Server, 2003. Available from http://www.zeus.com/.
8. Gyu Sang Choi, Jin-Ha Kim, Deniz Ersoz, Chita R. Das : A multi-threaded PIPELINED Web server architecture for SMP/SoC machines. WWW 2005: 730-739.
9. Chaitanya Singh. Threads, 2013. https://beginnersbook.com/2013/03/java-threads/ [2020-05-07].
10. Souvik Banerjee. Top 10 programming languages for web development. RS Web Solutions, 2015. https://www.rswebsols.com/tutorials/programming/ top-10-programming-languages-web-development [2020-02-25].
11. Oracle and/or its affiliates. Thread Safety, 2019. https://docs.oracle.com/cd/E37838_01/html/E61057/compat-14994. html#scrolltoc [2020-04-17].
12. MDN. Intensive JavaScript, 2019. https://developer.mozilla.org/en-US/docs/Tools/Performance/Scenarios/Intensive_JavaScript [2020-04-17].
13. J. Verdú and A. Pajuelo. Performance scalability analysis of javascript applications with web workers. IEEE Computer Architecture Letters, 15(2):105–108, 2016. https://ieeexplore.ieee.org/abstract/document/7307120 [2020-05-07]
14. Pitkow, James E. and Colleen M. Kehoe, Results from the Third WWW User Survey, The World Wide Web Journal, Vol. 1, No. 1, 1995.
15. Pitkow, James E. and Colleen M. Kehoe, "The Fourth GVU Center WWW User Survey", http://www.cc.gatech.edu/gvu/user_surveys/, 1995.
16. G. Banga and P. Druschel, "Measuring the Capacity of a Web Server," Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, Dec. 1997
17. M.F Arlitt and C. L. Williamson, "Web Server Workload Characterization: The Search for Invariants," In Proceeding of the ACM SIGMETRICS Conference on the Measurement and Modeling of Computer Systems, 1996, pp. 126-137.

18. M. Almeida, V. Almeida, and D. J. Yates, “Measuring the Behavior of a World-Wide Web Server,” Seventh IFIP Conference on High Performance Networking (HPN), White Plains, NY, Apr. 1997, pp. 57-72.
19. T. Bray, “Measuring the Web,” In Fifth International World Wide Web Conference, Paris, France, May 1996.