Project Report On

Multi-threaded Web Server

Submitted By

Chinmayi C. Ramkrishna (181IT113)

Mansi Saxena (181IT126)

Meghna Kashyap (181IT127)

K. Keerthana (181IT221)

IV Sem BTech (IT)

Under the guidance of

Dr. B. Neelima

Dept of IT, NITK Surathkal

In partial fulfillment for the award of the degree

Of

Bachelor of Technology

in

Information Technology

At

Department of Information Technology

National Institute of Technology Karnataka, Surathkal

TABLE OF CONTENTS

ABSTRACT

Web browsers and web servers interact using a text-based protocol called HTTP (Hypertext Transfer Protocol). A web browser opens an Internet connection to a web server and requests some content with HTTP. The web server responds with the requested content and closes the connection. The browser reads the content and displays it on the screen.

Single-threaded web servers suffer from a fundamental performance problem in that only a single HTTP request can be serviced at a time. Thus, every other client that is accessing this web server must wait until the current http request has finished; this is especially a problem if the current http request is a long-running CGI program or is resident only on disk.

A multi threaded web server is one that handles each request with a new thread, as opposed to handling each request with a new process. For a web server that receives multiple requests, multithreading is useful.

As a part of this project, a multithreaded web server will be implemented making the use of C language.

INTRODUCTION

A web server is software or hardware that uses HTTP and other protocols to respond to client requests. Web server software controls how a user accesses hosted files. The web server process is an example of the client/server model. All computers that host Web sites must have web server software.

This project aims to implement a web server which can handle multiple requests efficiently. It aims to build a multi-threaded web server that is capable of processing multiple simultaneous service requests in parallel. Implementation includes a multithreaded web server with appropriate synchronisation.

The implementation of the web server as a part of this project will be multi-threaded, where the processing of each incoming request will take place inside a separate thread of execution. This allows the server to service multiple clients in parallel, or to perform multiple file transfers to a single client in parallel.

OBJECTIVES

1. To understand the concept behind threads and the working of web servers and HTTP requests.
2. To learn how to code and implement multithreaded web servers using C/C++.
3. To learn how to create and synchronize multi threads in Unix.
4. To gain exposure on how a basic web server is structured.

METHODOLOGY

The project follows a simple methodology involving starting off with by obtaining a complete understanding of the basics which will be followed by understanding of the basics of the functions and libraries involved. Once done, code will be written in C language and executed on the Ubuntu terminal.

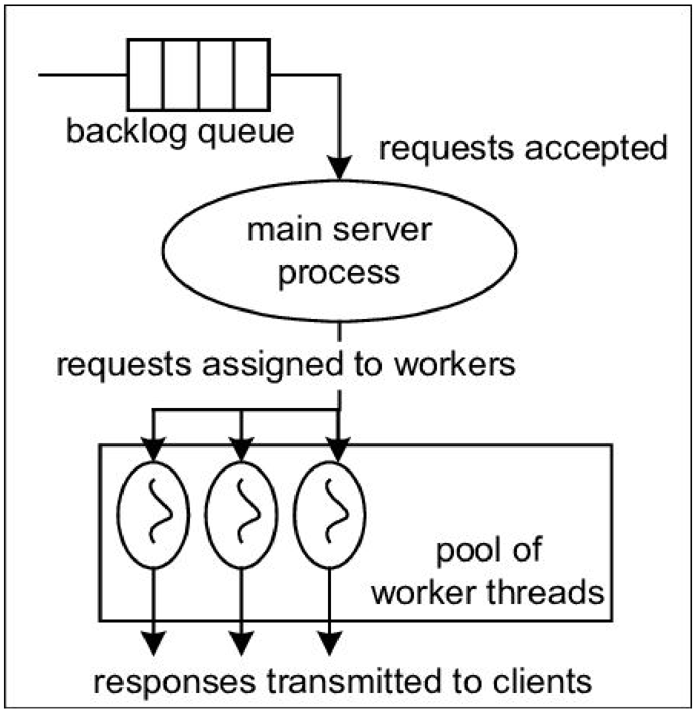
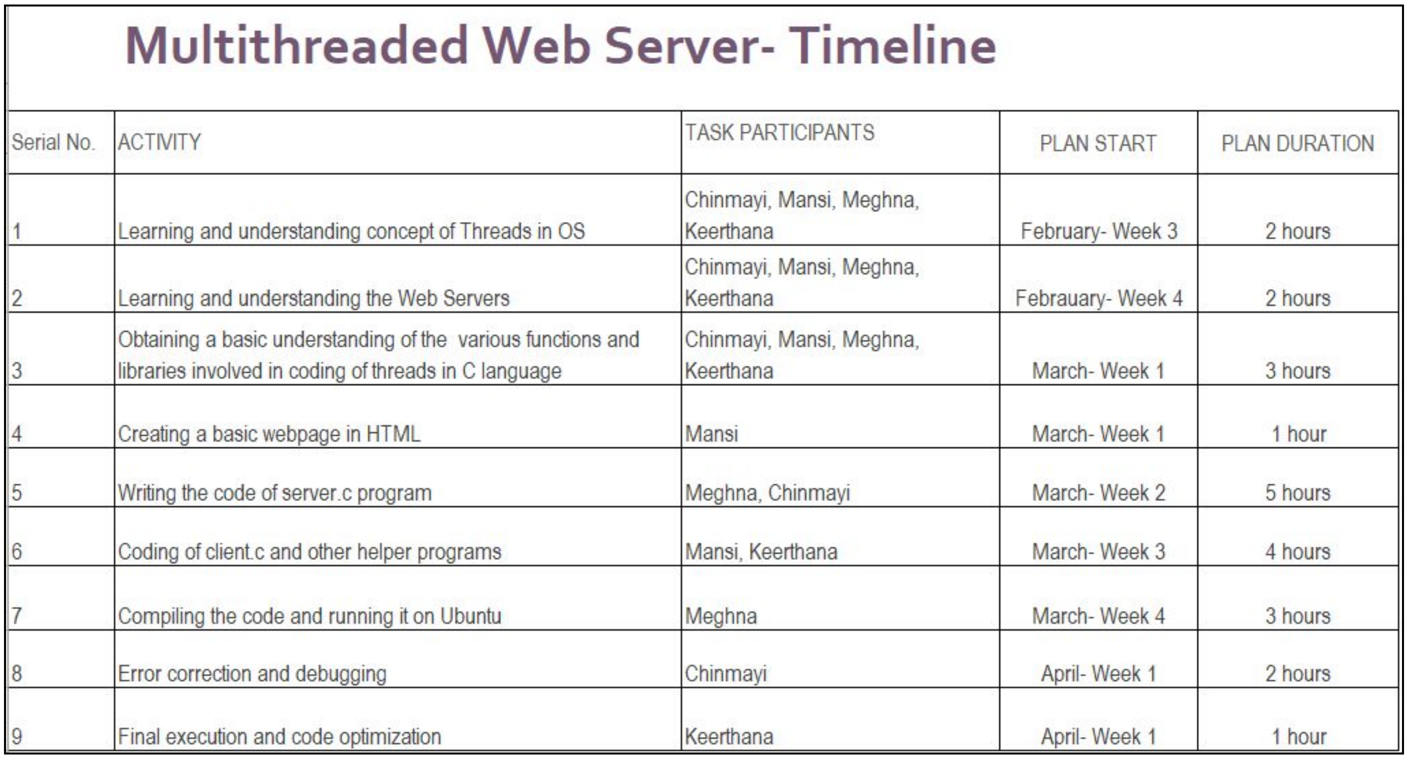


Figure 1: Multithreading flowchart

A simple approach to build a multi-threaded server for this project would be to spawn a new thread for every new http request. The OS will then schedule these threads according to its own policy. The advantage of creating these threads is that now short requests will not need to wait for a long request to complete; further, when one thread is blocked (i.e., waiting for disk I/O to finish) the other threads can continue to handle other requests.
However, the drawback of the one-thread-per-request approach is that the web server pays the overhead of creating a new thread on every request.

Therefore, the preferred approach for this project for a multi-threaded server is to create a fixed-size pool of worker threads when the web server is first started.
With the pool-of-threads approach, each thread is blocked until there is an http request for it to handle. Therefore, if there are more worker threads than active requests, then some of the threads will be blocked, waiting for new http requests to arrive; if there are more requests than worker threads, then those requests will need to be buffered until there is a ready thread.

TIMELINE



Multithreaded Web Server- Timeline

Serial No.	ACTIVITY	TASK PARTICIPANTS	PLAN START	PLAN DURATION
1	Learning and understanding concept of Threads in OS	Chinmayi, Mansi, Meghna, Keerthana	February- Week 3	2 hours
2	Learning and understanding the Web Servers	Chinmayi, Mansi, Meghna, Keerthana	Febrauary- Week 4	2 hours
3	Obtaining a basic understanding of the various functions and libraries involved in coding of threads in C language	Chinmayi, Mansi, Meghna, Keerthana	March- Week 1	3 hours
4	Creating a basic webpage in HTML	Mansi	March- Week 1	1 hour
5	Writing the code of server.c program	Meghna, Chinmayi	March- Week 2	5 hours
6	Coding of client.c and other helper programs	Mansi, Keerthana	March- Week 3	4 hours
7	Compiling the code and running it on Ubuntu	Meghna	March- Week 4	3 hours
8	Error correction and debugging	Chinmayi	April- Week 1	2 hours
9	Final execution and code optimization	Keerthana	April- Week 1	1 hour