Multi–thread Download Accelerator

Badal Karki

16th April 2018

1 Introduction

The rise in technology is helping people to do different things easily and more conveniently. Advancement in modern technology allows people to complete their tasks as soon as possible. In this digital world everyone wants to have a fast and accurate processor that can perform multiple tasks simultaneously and also in a very short period of time with minimal errors. There are multiple sectors in computerized environment where people want their machine to process quickly, like internet speed, playing games, downloading files, etc. So, this paper will presents some ideas about a download accelerator which helps to analyze the process of a multi-thread file download accelerator. Download accelerators helps to increases the speed, creating multiple connections of different file segments which speed up the downloads.

2 Background

The multi-thread download accelerator is a software that is actually built in Java programming language to download a file containing a large amount of data in a short period of time. Java was a suitable language for this program because the program consists of a connection between two users and Java has a built in package for sockets to connect between client and server. The program consist of two classes where one acts as a client and the other one as a server. Both client and server are connected via five different server sockets. Server sockets actually provide a communication mechanism between two users in Transmission Control Protocol(TCP). TCP connection was implemented instead of User Datagram Protocol(UDP) because even though UDP is fast, it is an unreliable connection

which does not make sure if all the data is being sent over the connection between users. Once the connection is established between client and server on the basis of IP address and Port number, the client side of the program requests a file to be downloaded. Before the requested file is sent to the client, the data of the given file was read using DataInputStream instead of BufferedReader because DataInputStream works with the binary data while BufferedReader works with character data so, DataInputStream consumes less memory space as it is a binary stream, where as BufferedReader consumes more memory space as it is character stream. Afterwards, the sever side of the program splits the requested file into five different small chunks and sent it to client through five different socket. The whole program is built in a waterfall process.

3 Analysis

The download accelerator software went through a series of testing to check its reliability. Five different sizes of file(100Mb, 200Mb, 400Mb, 750Mb and 1GB) were created using FileWriter class. Java FileWriter class is used to write character-oriented data into a file. Afterwards, each file was sent from the server side of the program to the client through five different sockets and the time taken to receive each file on client side was tracked. The test for tracking the time elapsed was done under three different conditions:

- Multi–thread with Merge

  The given size of file was split within five equivalent small chunks and the time taken to download a file was tracked including the time consumed to merge the partitioned file at client side of the program.

- Multi–thread with without merge

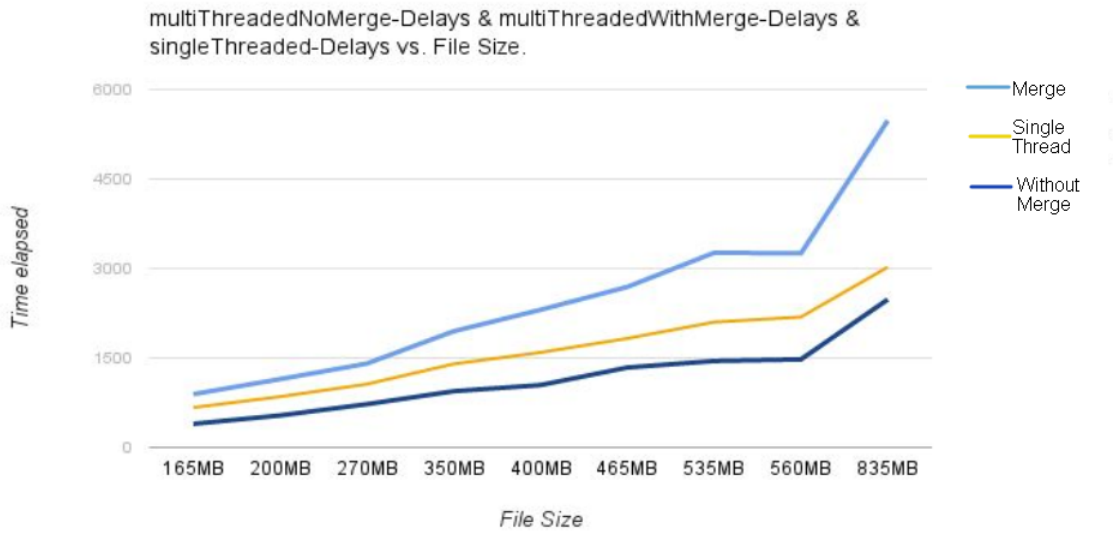
  The given size of file was split within five equivalent small chunks and the time taken to download a file was tracked excluding the time consumed to merge the partitioned file at client side of the program.

- Single thread

  The given size of file was sent as a single file without splitting from server to client and the time taken to download a whole file at client side was recorded.

4 Conclusion

Initially, there was not a big difference between these three tests but when the size of file used for testing was increased it was found that, the multi-thread without merge consumed significantly less amount of time to download a file compared to other two tests. The multi-thread with merge tended to consumed the highest amount of time to download a file and the single thread was the average among these three tests. So, the files download faster when partitioned and sent individually to the client. If files are downloaded without splitting, downloading process completes at a time faster than time calculated files after merging, slower than the time calculated when files downloaded before merging. Therefore we can conclude that, if this program was actually a software as download accelerator, the files would download faster than the regular time by splitting the file into a number of sections and downloading the sections more or less simultaneously. This approach would be a huge help with large files.



The above chart shows time elapsed while running the program with different size of file under three different tests. As a result, the evidence shows the multi–thread without merge consumed less amount of time for the file to be downloaded.