COL334 Assignment 2 Report Team pirate

Mrunal Kadhane - 2021CS10109 Kanishka Gajbhiye - 2021CS50131
Vamshi Vangala - 2020CS10404 Shiva Krishna Sagi - 2020CS50441

September 8, 2023

1 Algorithm for code and error checking for p2p

Our team has implemented a p2p exchange network through **master client** and **slave clients**. Each of the clients is connected to **"vayu.iitd.ac.in"** to read the lines and all the information exchange occurs individually between master and slave client. The master client acts as a central hub for receiving and sending the data for all the clients.

1.1 Slave Client :

The slave client runs **three parallel threads -**

1. reading from **vayu**
2. reading from the master client
3. sending to the master client.

Each slave client maintains its own dictionary to store the lines. Whenever a new line is added to the dictionary, that line is sent to the master client. Parallely, it also receives from the master client and adds the received line to its dictionary if it's a new line.

We have implemented a two-way communication in the receiving and sending thread for master-slave. The slave sends an acknowledgment on the same thread of receiving so that the master client is alerted and it sends a new line and by this method, we get to know that the connection is intact. In case of sending to the master thread, the client receives a response from the master and it then sends a new line. The **termination condition** is when the client has received all 1000 lines and the queue that was maintained for the master client for popping is empty i.e. it has sent everything new it had. A new line detection is always occurring at the receiver's end.

1.2 Master Client :

The Master client runs two parallel threads for sending and receiving from each slave i.e. **6 parallel threads and one separate thread** for reading from "vayu".

We maintained three separate queues for sending to each of the slave clients. Whenever a new line received from a certain slave is added in the dictionary, the same line is added to the queues of the remaining two slave clients and popped accordingly. The **termination condition** is when the master client has received all 1000 lines and all the queues of slave clients are empty i.e it has sent everything new it had. A new line detection is always occurring at the receiver's end.

In this way, all the information is being shared among all the peers with a central authority to manage distribution.

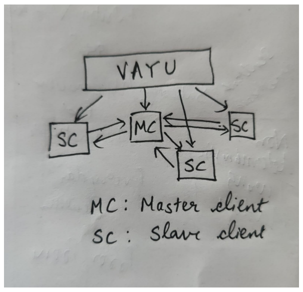

Figure 1: P2P Network Exchange Overview

1.3 Error Checking :

There can be cases when the connection with "vayu" or a client gets broken during the execution of the program and we must restore the connection since it's highly crucial.

If the master client fails, the entire network will slow down. In case of such exceptions, we have directed the code to reconnect to the server or the client using the try-except block in Python. Any exception thrown redirects the code to the except part which forces the client to reconnect. We have also used timeouts to ensure that a client waits for at max 5 seconds to receive an acknowledgement response in case of slow transfer. Finally, all the lines are assembled properly in ascending order of line number and submitted back to vayu. All sockets are finally closed and all threads are terminated.

2 Graph Analysis

2.1 With 1 peer

We only ran the master client program to download the file. As shown in plot, the time taken to get unique lines grows exponentially especially around 800 unique lines. The time taken to download 600 unique lines is around 20 secs (approx), but the time to download the entire file is more than **a minute**.

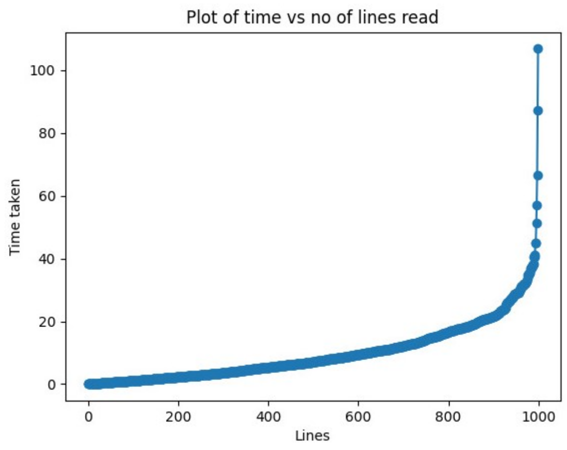

Figure 2: Downloading the file using one client

2.2 With 2 peers

We ran the master client program on one device and client program on one other. The total time to download the file was observed to be around **40 secs**. Even the first 600 unique lines were received under 10 secs.

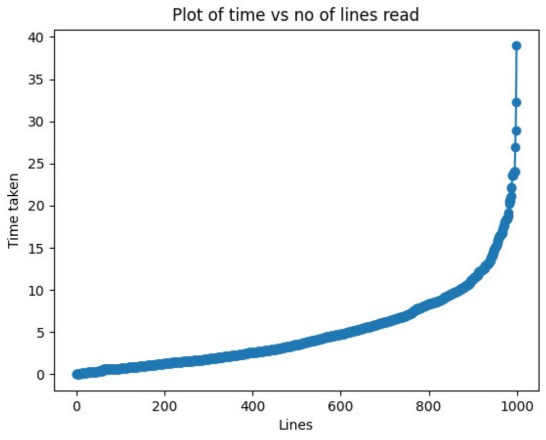

Figure 3: Downloading the file using 2 clients

2.3 With 3 peers

We ran the master client program on one device and client program on 2 other devices. The total time to download the file was observed to be around **30 - 35 secs**. We see improvement in time taken to download the file as compared to previous cases.

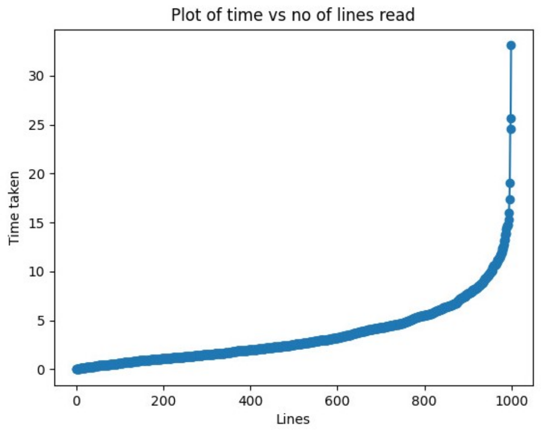

Figure 4: Downloading the file using 3 clients

2.4 With 4 peers

We ran the master client program on one device and client program on 3 other devices. The total time to download the file was observed to be around **20 secs**. We see improvement in time taken to download the file as compared to previous cases. Hence, a single client downloading a file vs 4 clients working in cooperation to download the same file, the latter takes significantly less time as compared to former.

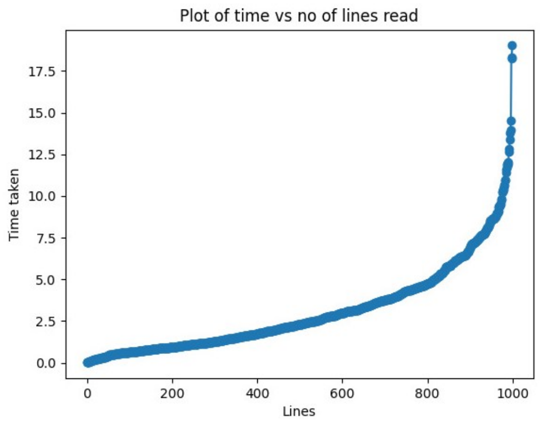

Figure 5: Downloading the file using 4 clients

We can observe that for reading 1000 lines, the decrement in time goes like 100 ->45 ->35 ->20 in seconds and it is non linear in nature. There is a sudden drop while reading from one peer to two peers and eventually it is more or less the same drop. We can claim that it is exponentially decreasing in nature.