COL334 Assignment 2

September 8, 2023

Aastha Rajani (2021CS10093), Disha Nitin Shiraskar (2021CS10578),
Namrata Thakur (2021CS10587), Srushti Junghare (2021CS50613)

1 Approach

- In this collaborative file download assignment, the primary objective is to efficiently retrieve a large file by organizing a network of multiple clients. We have implemented a collaborative file download mechanism using Master-Slave network.
- Each client writes a client program for connecting to the 'vayu server,' to fetch distinct parts of the target file. Among the four clients, one is designated as the 'master client,' while the rest act as 'slave clients.' All clients, including the master and slave clients, simultaneously connect to the 'vayu server' and request lines from the file through 'SENDLINE' requests.
- The master client assumes the responsibility of coordinating the file download process. To achieve this, the master client initiates its own server, allowing it to receive connections from slave clients. Meanwhile, each slave client also starts its server to facilitate communication with the master client. The primary role of the slave clients is to forward the lines they receive from the 'vayu server' to the master client. What sets the master client apart is that it receives lines from both the 'vayu server' and the other three slave clients.
- To effectively manage the received data, the master client employs an array. The array is used to index and store the content of lines received from all the sources, ensuring that the lines remain distinct. The master client continues to collect lines until it accumulates a total of 1000 unique lines, carefully indexing them in its data structure.
- Once this 1000-line threshold is reached, the master client takes on a more active role. It starts its server and begins the process of distributing these 1000 lines to the slave clients via their respective servers. The slave clients, in turn, update their local arrays to incorporate the lines received from the master client. This collaborative exchange ensures that the group collectively possesses all 1000 distinct lines.
- Finally, when all four clients have successfully received and assembled the entire file comprising 1000 lines, they start making their individual submissions to the vayu server. In this way, this approach helps in the faster download and assembly of the file, marking the completion of the collaborative file download process.

2 Error Handling

2.1 Slave disconnects while sending lines

- In both the client handler function of the master code and the connect to server function of the slave code, there's a while loop that continues until count reaches 1000. This loop is used to keep the peers working.
- When working with sockets, if the connection is interrupted or there's any other issue, these lines(with "socket name".recv and "socket name".connect) will raise a socket-related exception. When such an exception is raised inside the loop, the normal execution of the loop is interrupted, and Python looks for an exception handler to transfer control to.
- When a slave client disconnects and tries to reconnect, it enters try block of the connect to server function. The try block attempts to establish a connection with the server. If the connection is successful, the client continues fetching lines. Else, it enters the exception block at the end and continues to the next iteration of the outer while loop. It will keep trying to establish a connection until it succeeds, and it continues to fetch lines once the connection is reestablished.
- The client handler function in master code also functions similarly and tries to send connect requests to slave server until the connection is reestablished.

2.2 Slave disconnects while receiving lines

- This error is handled by the start server function in the master code and client handler function in the slave code.
- Client handler function has try block to connect to the master's server. If the slave disconnects it will be directed to the exception block at the end and then continues to attempt sending connection requests to the master's server. When it reconnects back, the connection is reestablished and normal functioning is resumed.
- Start server function deals with this error by accepting the slave's request again (after reconnection) and creating a completely new thread to again start sending lines to this slave. We have decided to use this approach observing the fact that, broadcasting the received lines to the slaves takes fractions of seconds and therefore we can afford to start sending all over again instead of keeping track of last line sent before disconnection.

2.3 Master/Slave disconnecting from vayu

- In both master and slave code there is a while loop inside the connect to server function which runs till the respective peer has received thousand distinct lines. This ensures that if a peer disconnects from vayu , it continues sending connection requests until the connection is reestablished due to the presence of try and except block.

2.4 Lines Received in Groups Instead of One at a Time

- The code utilizes a buffer variable to accumulate received data until it contains at least two newline characters . This indicates that at least one complete line has been received. The code then splits the buffer to extract one line at a time.
- This prevents the loss of lines received in response (if there are more than one) as they continue to be in the buffer. It also handles the cases of receiving only line number / half part of line in one iteration and the other half in the next.
- This logic effectively handles scenarios where lines arrive in groups or chunks. The code will correctly split, parse and process the lines, ensuring that each line is processed separately.

2.5 Handling Exceptions

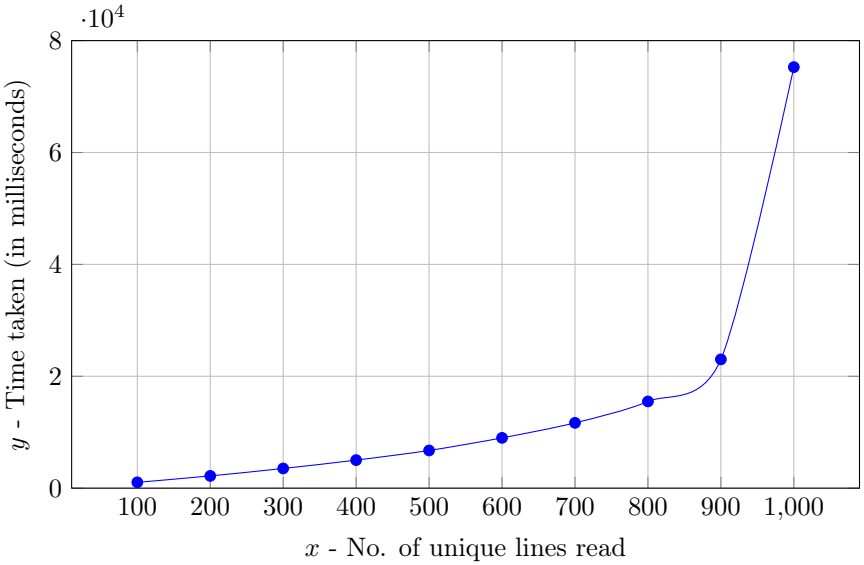
- Throughout both the master and slave client codes, there are try blocks to encapsulate potentially error-prone operations such as socket connections, sending data, and receiving data.
- In the event of an exception, the code gracefully handles it by closing the socket if it was open and setting appropriate flags or variables (e.g., connected and flag) to control the flow of the program.
- Additionally, the code prints informative error messages, allowing for easier debugging and troubleshooting. This is especially important in a distributed system where network issues can be common

3 Plots

- PLOT 1- One client downloading the file by itself from VAYU:-

x	100	200	300	400	500	600	700	800	900	1000
y	1040	2191	3520	5014	6741	8990	11676	15500	23021	75265

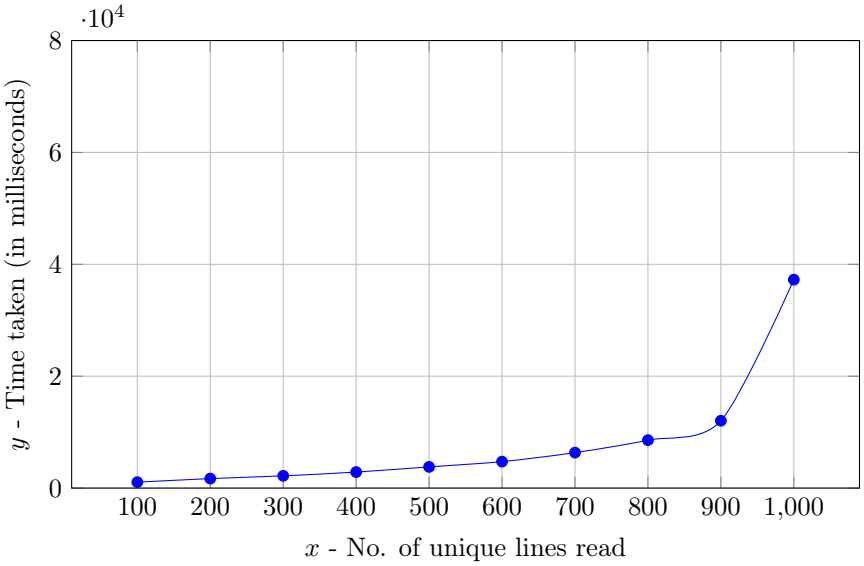
Table 1: One client downloading the file by itself from VAYU



- PLOT 2- Two peers working in cooperation :-

x	100	200	300	400	500	600	700	800	900	1000
y	1055	1699	2189	2867	3787	4732	6341	8570	12041	37257

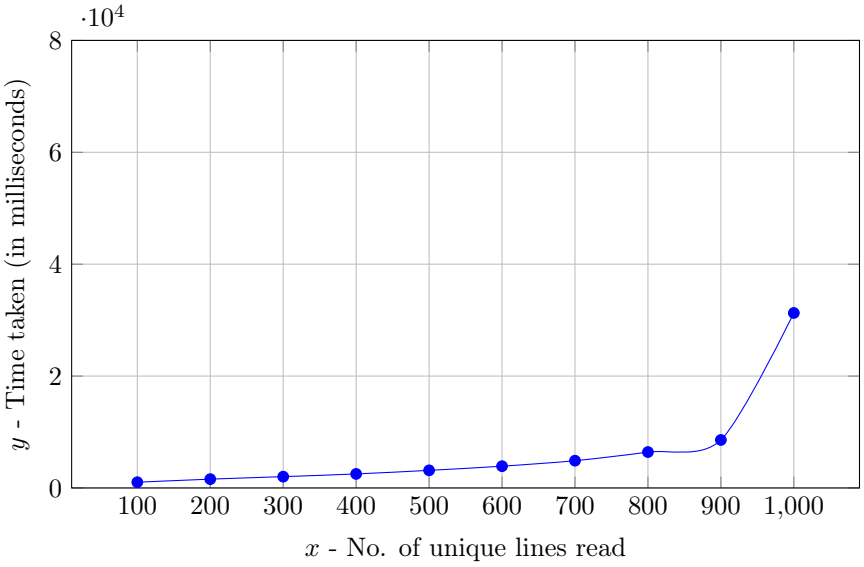
Table 2: Two peers working in cooperation



- PLOT 3 - Three peers working in cooperation :-

x	100	200	300	400	500	600	700	800	900	1000
y	1024	1575	2021	2507	3147	3891	4873	6418	8579	31279

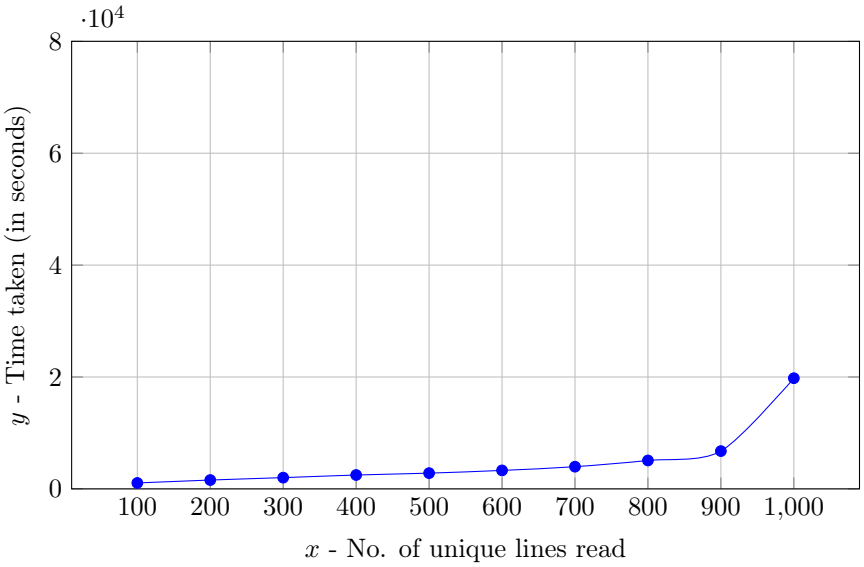
Table 3: Three peers working in cooperation



- PLOT 4- Four peers working in cooperation:-

x	100	200	300	400	500	600	700	800	900	1000
y	1055	1579	2010	2477	2813	3300	3965	5077	6753	19782

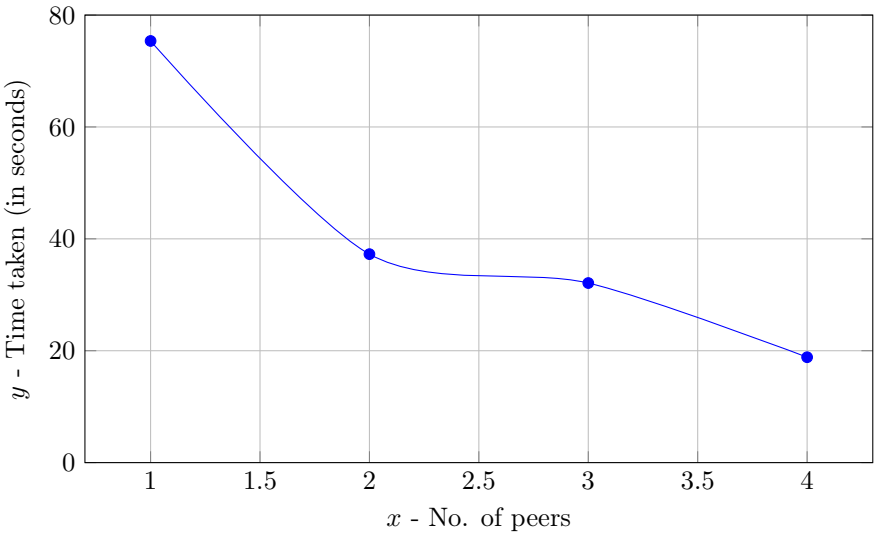
Table 4: Four clients working in cooperation



- PLOT 5 - No. of Peers VS Time Taken to download all lines :-

x	1	2	3	4
y	75.4	37.3	32.1	18.8

Table 5: No. of peers VS Time taken to download all lines



4 Interpretation from Graphs

4.1 Total Time Variations

- When the number of clients doubled (from 1 to 2), the time required nearly halved (from 75.265228s to 37.257127s). This seems almost linear. Adding a third client doesn't cut the time in thirds compared to the single client scenario, but there is a reduction (from 75.265228s to 32.06012s). With four clients, there's a significant reduction in time again (from 75.265228s to 19.78222s).This is, however, still more than a quarter of the time taken by one client.

4.2 Why is the relationship not linear?

- The expectation for a linear relationship would be that when we double the clients, we halve the time, and when we quadruple the clients, the time should be quartered, and so on. However, the given data does not align with this linear assumption.
- This can be attributed to the fact that as the number of clients increases, the overall system's efficiency in exchanging and managing the parts of the file might decrease. There are also chances that two or more clients receive the same line, causing redundancy.
- As more clients are added, the likelihood of two or more clients requesting and receiving the same line from the server goes up. This redundancy doesn't contribute to overall progress, hence reducing the expected efficiency gains. The complexity of ensuring efficient coordination among clients increases non-linearly with the number of clients. This complexity can lead to delays, especially when ensuring that no lines are missed or redundantly downloaded.

4.3 Decrease in rate of receiving distinct lines over time

- In the beginning, the likelihood of a client receiving a new line (not received previously) is high. This explains the faster rate of obtaining unique lines in the initial stages. As more lines are received, the chance of getting a repetitive line (that was sent earlier) increases, which causes a decrease in the rate of obtaining unique lines. This is evident in the single client scenario where the time difference between 900 to 1000 lines is significantly large (from 23.021004s to 75.265228s).
- When multiple clients are working, they can collectively cover a larger portion of the file faster, especially in the initial phase. They can also exchange the parts of the file they have, reducing redundancy. The addition of more clients, therefore, undeniably speeds up the download process, though not linearly

4.4 Pros of downloading file collaboratively

- **Reduced Waiting Time:** In the collaborative method, once a peer receives a unique part of the file, it can share that part with other peers. This reduces the chance of peer waiting for the vayu server to send a particular line because there are high chances of receiving this line from the other clients, thereby reducing inefficiency and allowing faster downloads.
- **Optimized Bandwidth Usage:** By splitting the file into parts and having each peer download different sections concurrently, the combined bandwidth of all peers is utilized, making the download process faster.
- **Overcoming Rate Limiting:** As the vayu server has a rate limit per connection, using multiple peers can bypass this limitation. If there was single client, the maximum number of requests it would send the server would be bounded by the rate limit of the server, increases the chances of receiving empty line , thereby adding up to the overall time. On the other hand, each peer gets its share of the rate-limited bandwidth, so in total, they can download faster than a single peer limited by the same rate.

- **Resilience and Fault Tolerance:** If one peer faces issues (e.g., connection loss or slowdown), others can continue downloading and sharing the file parts. This collaborative approach offers better reliability than having a single point of failure.
- **Dynamic Load Sharing:** As peers complete downloading parts of the file, they can assist others, ensuring that the 'workload' is continually redistributed based on peer performance and availability.
- **Effective for Large Files:** For particularly large files, the probability of a single peer getting the same parts repetitively increases over time. Collaborative downloading minimizes this problem as peers can quickly exchange unique parts, reducing the overall download time.
- **Resource Optimization:** For servers, it's more efficient to send out different parts of a file to different peers than to send the entire file multiple times to different peers. This way, server resources, especially bandwidth, are optimized.

4.5 Increase in Efficiency

- The advantage of having a collaborative multiple-client network for downloading a large file is clearly visible in the last graph (Time Vs No. of Peers) . Minimum time required we recorded to download the file using this approach is close to 16s-17s. On the other hand, if there were 4 separate clients downloading the file individually , the total time taken would have been around 75.265 seconds. The overall increase in efficiency can be calculated as : According to the graph : ((75.265-19.782)/75.265) * 100 = (55.483/75.265) * 100 = 73.717%