Machine Programming 3- Simple Distributed File System

An Phan(anphan2), Chang Liu(changl25)

Design for SDFS (Simple Distributed File System):

1. Components of SDFS:

a. Local File: Each machine in the SDFS cluster stores its local files.

b. Leader: One machine in the cluster acts as the leader, responsible for managing the distributed file system.

c. File to Server Mapping: The leader maintains a mapping that associates each file in the SDFS with the four machines that hold its replicas. This mapping ensures that each file is replicated on four different machines for fault tolerance.

d. Server to File Mapping: This mapping consolidates the local files from every machine in the cluster, effectively creating a unified view of the entire file system.

2. Handling Failures:

- When a machine fails, the leader selects another machine to hold that missing replicas that the failed machine held.

- Leader Election: If the leader machine itself fails, the system employs a leader election mechanism. The node with the highest attribute value takes over as the new leader and informs the other nodes. The new leader also requests the local files from other machines to restore the Server to File mapping.

3. Query Handling:

- For every operation, such as "put localfilename sdfsfilename," "get sdfsfilename localfilename," or "delete sdfsfilename," the queries are sent to the leader. The leader then determines the appropriate action to take based on the operation.

4. Replication Level:

Each file in SDFS will have four replicas distributed across different machines. This redundancy ensures data availability and fault tolerance in the face of at most 3 machine failures.

5. Starvation Avoidance:

To avoid starvation for both read and write operations, the leader executes the command in FIFO order, and allows at most 2 concurrent read for the same file.

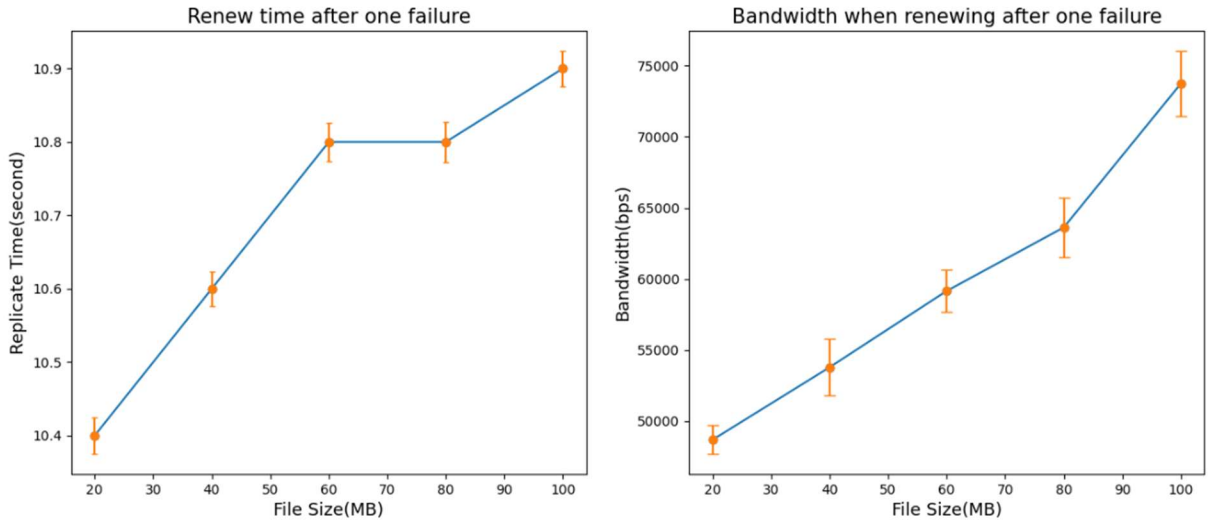
Past MP Use:

- The failure detection mechanism from MP2 is utilized to maintain the membership list for MP3 and to gossip information about the current leader's status.

- MP1 is leveraged to retrieve execution information from each machine, which aids in debugging and provides a comprehensive overview of what transpired after issuing a query on each machine.

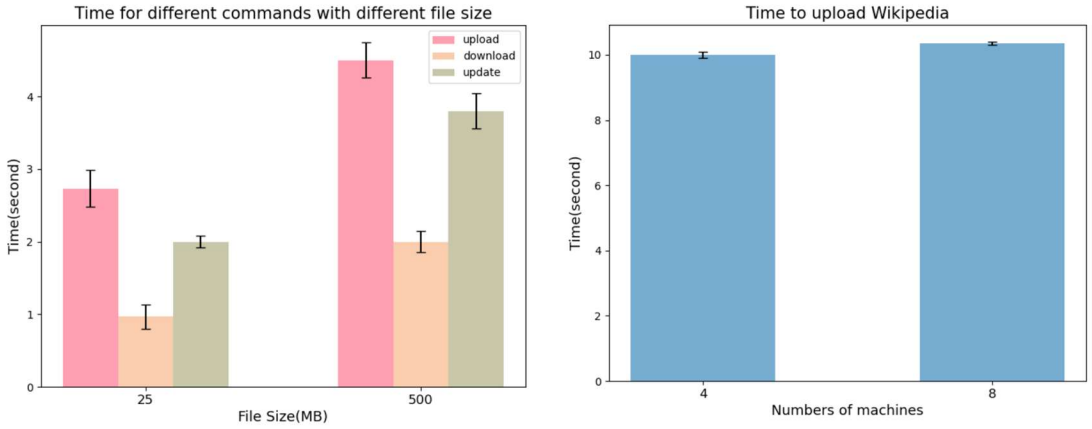
Measurements:

1. Re-replication time and bandwidth upon a failure:



The time needed to renew and the required bandwidth after a failure both tend to increase as the file size grows. This is a natural outcome because larger files take more time to replicate to another machine. However, it's important to note that the increase in required bandwidth is intended to expedite the replication process. As a result, the difference in replication time between various file sizes is relatively small. In most cases, most of the time spent in this process is dedicated to detecting the failure, rather than the replication itself.

2. Times to insert, read, and update and time to store the entire English Wikipedia corpus:



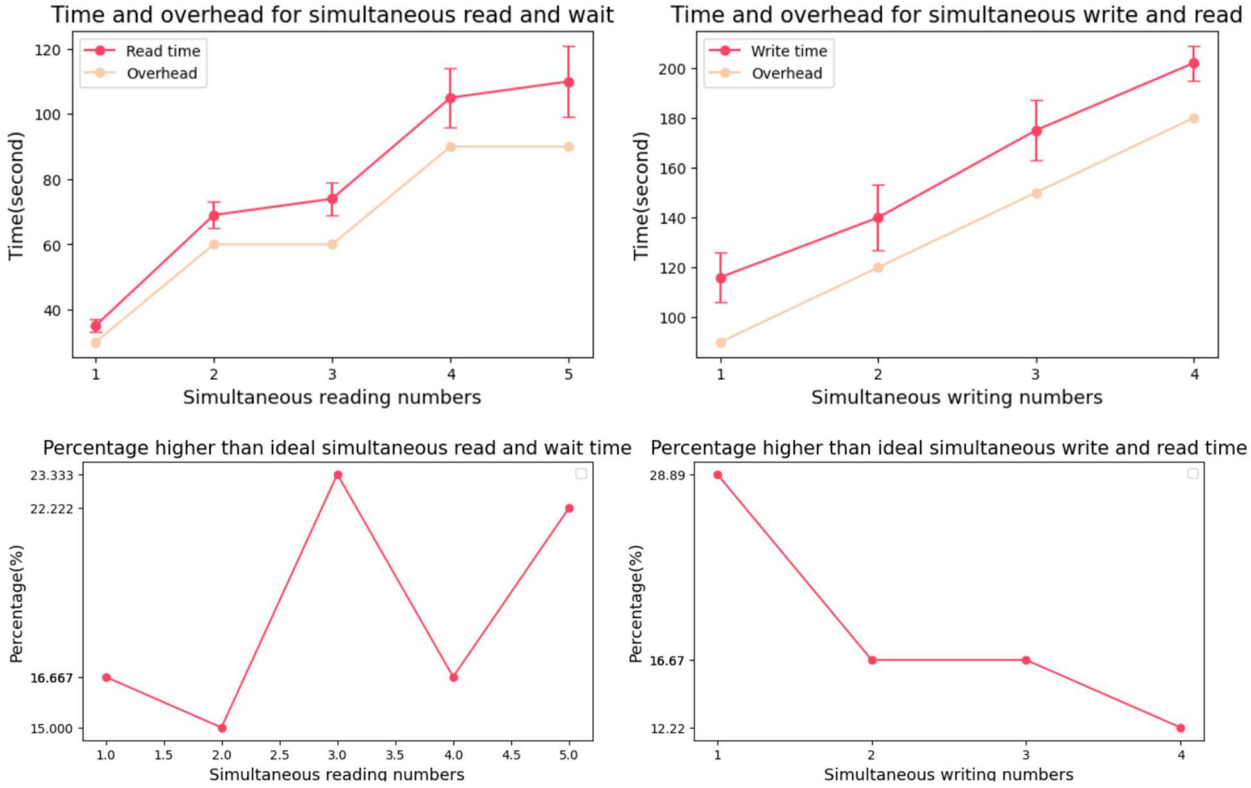
In terms of completion time, the upload operation takes the longest, followed by the update, and then the download. This order is directly proportional to the complexity of each command, with the most complex operation taking the most time, and the simplest taking the least. It's worth noting that the completion time for all these commands increases as the file size grows.

An interesting observation is that there is no discernible difference in upload time when the SDFS system utilizes either 4 or 8 machines. Regardless of the number of machines involved, the upload operation consistently takes the same amount of time.

3. Read-Wait and Write-Read time:

To calculate the overhead time for Read-Wait, we apply the formula: ceil((m+1)/2) * 30 because we permit at most two readers to access the data simultaneously, each takes30 second to complete.

For the overhead time of Write-Read, we use the formula (m+2) * 30. This formula encompasses both Write-Read and Write-Write operations, which must be executed sequentially, each takes 30 second.



The Read-Wait time and Write-Read time follow a similar trend to the overhead time. Read-Wait time tends to be approximately 20% longer than the overhead time, while Write-Read time is approximately 15% higher than the overhead.

The reason for these differences is that, in addition to the overhead time, the SDFS requires confirmation from various machines when executing these commands. This additional communication overhead adds to the total time. In an ideal scenario, where machine connection times are negligible, the commands would behave as if they were executed on a single, unified machine. However, real-world communication complexities introduce these delays.