MP3 Report Team 63

Team Member: Vincent Chen(vfchen2), Jen-Chieh Yang(jy75)

1. Design:

The primary process initiates threads for MP2's failure detection as the first step. Subsequently, the system triggers a leader thread if the server is identified as a leader (determined by selecting the server with the minimum server ID from the membership list). When the original leader fails, the next minimum ID server would identify itself as leader and continue the job scheduling process. The server also runs a receiver process to accept requests from the leader. These requests are initially forwarded to the leader, which then decides which servers should receive the forwarded request.

For replication purposes, the system randomly selects a server that is both alive (according to the membership list) and does not already contain the specified file in the file table. Given that the leader forwards requests to four different servers for each "put" operation, replication on each node failure ensures that there are at least four replicas maintained throughout the entire system. Since we would need 5 seconds to detect failures, the membership list may not be updated in time, and there would be false puts to send the re-replicated files to a server that is just dead. Hence, when failures occur simultaneously, we would check that the replicated file can be sent to the designated server to ensure this doesn't happen.

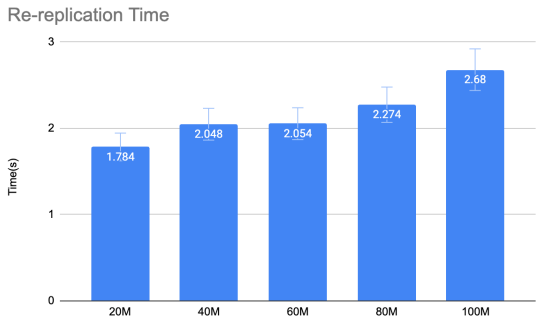
The system also maintains a list of jobs and a counter for each file, where the counter can serve as a lock. The counters would be modified when initiating a new task or getting "ACK" from the server. Jobs are added to the list sequentially, and the system iterates through this list at each timestamp. This approach helps prevent starvation, as tasks related to the same file are executed on a first-come, first-served basis. The system would also maintain a flag and counters for times of continuous same operations for each file. This would be able to prevent too much "read" before "write" and vice versa.

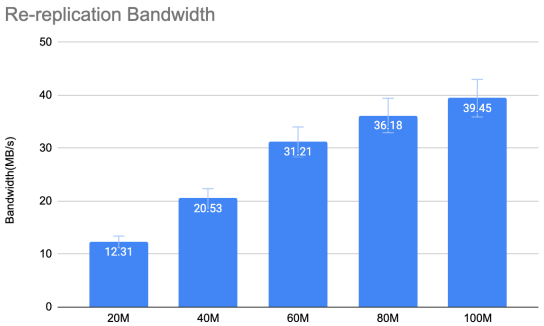
2. Previous Use of MP Components:

Additionally, we utilize changes in the membership list (MP2) to determine when replication should be initiated as failure is detected. Furthermore, MP1 is employed to retrieve specific logs on the MP3 server for debugging purposes.

3. Measurements

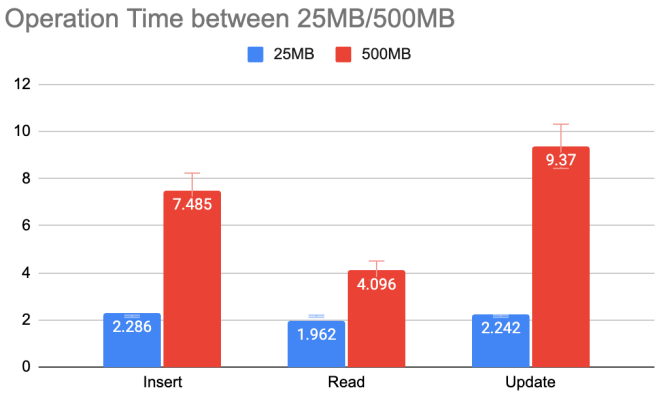
Test1





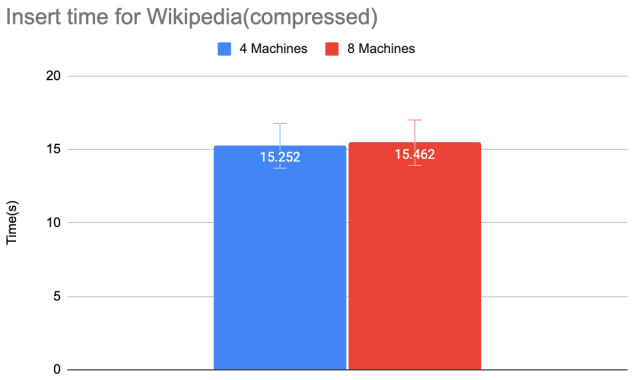
For the overhead, we didn't include the time for failure detection, which would take up to 5 seconds. Both the time and bandwidth gradually increase when the transmitted file size increases. The reason is that the file transfer speed is far greater than the overhead of our scheduling process. Hence, with larger files, the bandwidth could be portioned more towards the file transfer part. Therefore, the bandwidth will increase as the file size increases until the file transfer speed reaches maximum.

Test2



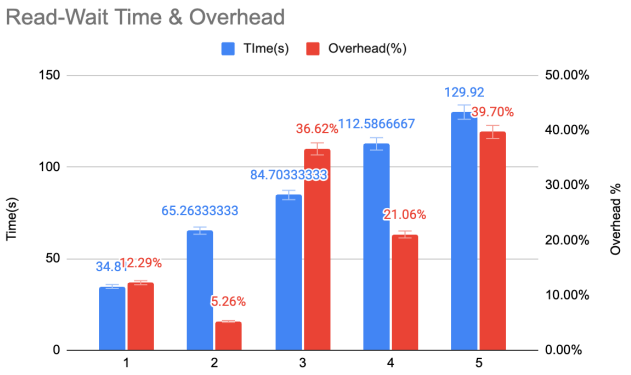
For Insert and update, both methods refer to "put." Therefore, we got a similar performance. Since the "put" operation requires sending multiple replicas to different machines and needs to send updated file table information. Insert and Update would take a bit longer than Read, which only needs to fetch data once and doesn't update the file table.

Test3



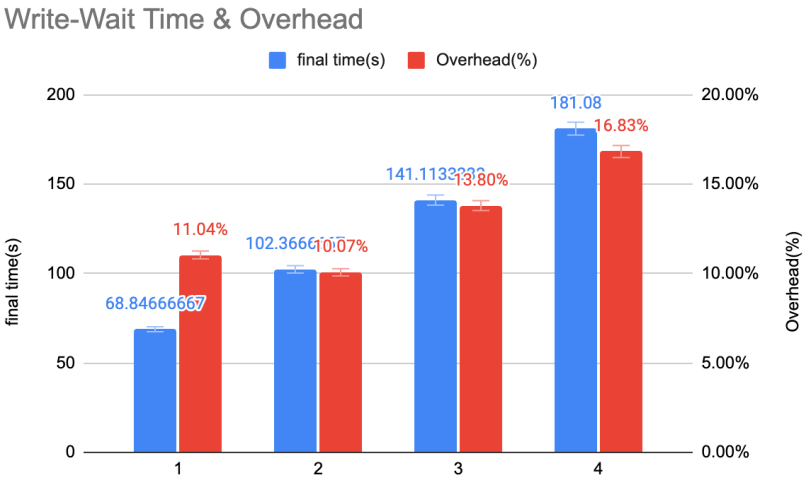
Since we try to keep each file with 4 replications on the file system, we would be only sending out 4 replications to 4 of 8 servers. In this design, the time for 4 machines should be close to 8 machines, which has a little time difference for sending the updated information from the file table is less with only 4 servers.

Test4



Ideally, we should observe the result of M=2 equals M=3 and M=4 equals M=5 since the scheduler allows 2 “read” operations at the same time. However, since there might be a variance in VMs’ bandwidth (see last section), the time difference (latency) might be propagated to later tasks. Overall, we observed that the overhead of M=2 and M=4 (first read task of continuous 2 read tasks) were lower than M=3 and M=5.

Test 5



Since the “write” operation is only allowed once at a time, we could observe that the overheads of different Ms are closed. The time to finish different M increases closely to the time to a single “write” operation. The latency might be due to the same reason of different bandwidths on different machines, as mentioned in test4.

* Observation

While we tried to move test files to different VMs, we observed that different VMs might have different bandwidths and might vary on a large scale. This would directly affect the result of our result.