Team05 - Yuankai (yuankai4) and Boyou (boyouh2)

MP3 Report: Simple Distributed File System Team05

Design: We store 3 replicas for each SDFS file and also for the metadata that is stored in the master. Therefore, we have 4 copies in the system and it will have at least one copy left when there are 3 failures at the same time. We let the first VM join the system to be the master. If the master fails, we will run a ring election to elect the new master. We use the highest initiator id to make sure only one election will be finished. The new master will get the metadata information from one of the remaining master backups. Also, the master will run periodically to make sure the replica and metadata are correct.

When putting the file, the client will ask the master node where to store the file and the master will let the client know where to send it. Next, the client sends the file to a node and sends a notification message when sending is done. After the node gets the notification, it will tell the master that the file is stored in its location and send the file and the upload success notification to its successor to store the replica if needed. Then, the file metadata in the master and its backup can be updated. Therefore, we can tolerate the failure of the client after one copy of the file is stored in the system. We set the needed **W** value to 4. The master will send a message to the original client that the put is finished when it gets 4 replicas. Since only the master will receive the put request and assign the versions, we have a total ordering that order will be the sequence that the master receives the message.

When getting the file, we set **R** value to 1. Therefore, W and R will be guaranteed to have an overlap. The client will ask the master about the storing locations and try to get the file from any of the storing locations. The master will also send the version information to let the client request the latest or multiple versions.

Similarly, the delete command will also go to the master and the master will let all the current storing locations remove the file and the master will also remove it from its metadata. LS command will also need to go to the master to get the result. The “store” command can be handled locally because each node will keep a record of the files it is storing.

When the master receives a message about a node leaving, it will check the metadata and make extra replicas for the files that were stored in the leaving node. We didn’t do balancing when a new node joins because we want to keep the system simple and moving the files will use lots of resources.

Past MP Use: The file system is based on MP2’s membership protocol. We use the topology that we built to decide where to store the files and use the failure detector to help us know when we should elect a new master. We keep the overall structure about how to communicate among the VMs, but also make small changes to separate the ports. We did an update to allow multiple introducers so that we can tolerate the failure of the only introducer in MP2. MP1 also helps us to debug our system when it becomes more complicated. We figured out the error in the failure detector after we make some modifications based on the logs.

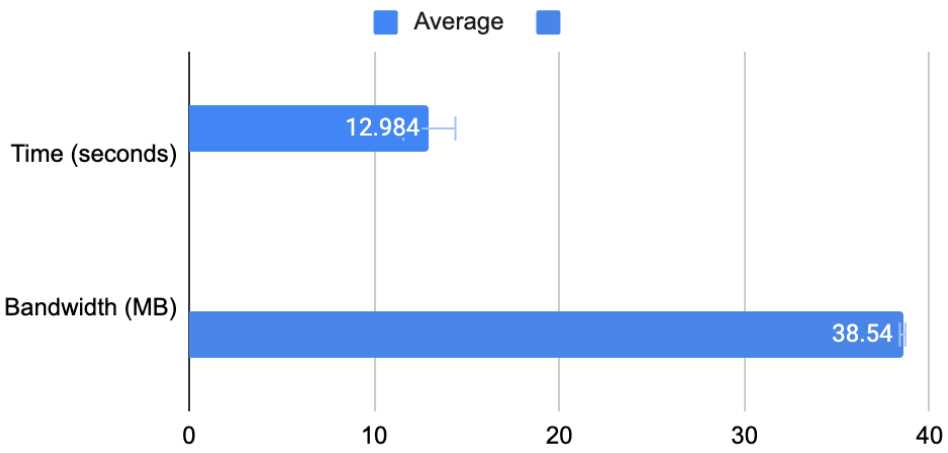
Measurements: (i) re-replication time and bandwidth upon a failure

The file we use is 38MB (tried to make it close to 40MB). It is expected that the bandwidth is a little more than the size of the file since we are doing a replication of the file and some other message transfers, including file metadata sharing. The standard deviation is small since the operations are about the same each time.

	Datapoints	Average	Std

Team05 - Yuankai (yuankai4) and Boyou (boyouh2)

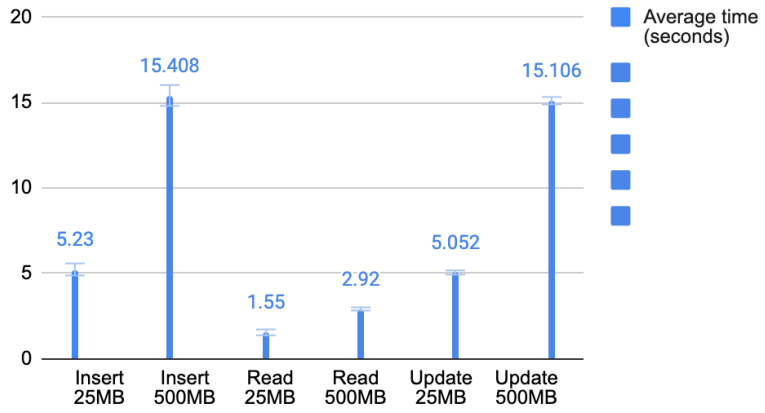
Time (seconds)	15.53, 11.21, 12.84, 12.77, 12.57	12.984	1.405
Bandwidth (MB)	38.4, 38.8, 38.4, 38.6, 38.5	38.54	0.15



(ii) Times to insert, read, and update, file of size 25 MB, 500 MB

It is expected that operations to the 25MB file are faster than the operations on the 500MB one. Insert and update are about the same because updating is just saving the file with a larger version number. It is also expected that the standard deviation is small for all the operations because the data flow of the operations is the same each time.

	Insert (seconds)	Read (seconds)	Update (seconds)
25MB	5.64, 5.52, 5.35, 4.92, 4.72	1.30, 1.56, 1.46, 1.83,1.60	5.09, 4.98, 4.85, 5.15, 5.19
500MB	16.09, 15.83, 15.69, 14.43, 15.00	3.01, 3.04, 2.84, 2.91, 2.80	15.05, 15.47, 14.99, 15.19, 14.83



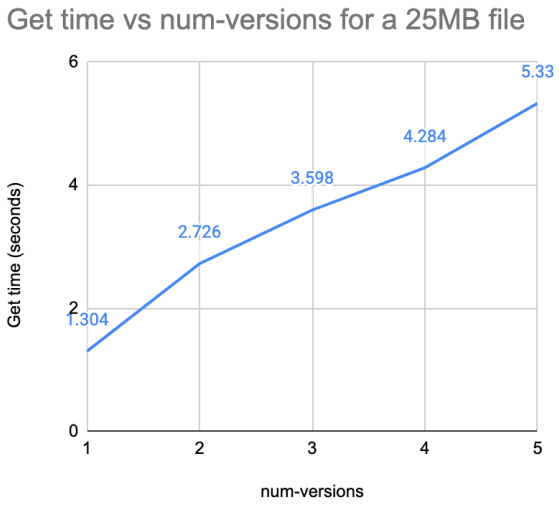
(iii) Plot the time to perform get-versions as a function of num-versions;

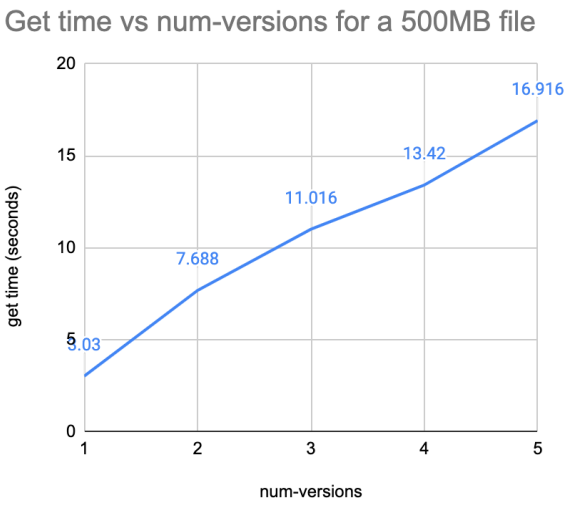
It is expected that the graph is about liner from the data on both 25MB and 500MB files. It runs a bit faster when the node gets the file from itself.

num-versions	1	2	3	4	5

Team05 - Yuankai (yuankai4) and Boyou (boyouh2)

Get time for 25MB (seconds)	1.23, 1.27, 1.36, 1.41, 1.25	2.89, 2.40, 3.01, 2.53, 2.80	3.27, 3.57, 4.04, 3.69, 3.42	4.42, 4.11, 4.01, 4.30, 4.58	5.32, 5.80, 5.05, 5.26, 5.22
Get time for 500MB (seconds)	3.00, 3.28, 3.09, 3.01, 2.77	5.62, 8.85, 7.77, 9.73, 6.47	8.47, 10.35, 12.17, 12.86, 11.23	10.22, 13.62, 13.19, 19.88, 10.19	16.29, 18.98, 19.39, 12.58, 17.34





(iv) **Store Wikipedia with 4 machines and 8 machines**

It is expected that it takes less time to store on 8 machines because the total network is the same and it separates the load of the storing so that more files can be stored in parallel.

	Storing time (seconds)	Average (seconds)	Std
4 Machines	362, 287, 263, 310, 284	301.2	33.855
8 Machines	237, 226, 207, 229, 252	230.2	14.689