CS425 MP3 Report Wenhao Su (wenhaos3) Yichen Yang (yy18) Group 02

Simple Distributed File System Design:

In this project, since we want to tolerant for 3 failures, we need at least 4 replicas. To minimize the bandwidth usage, we choose to use 4 replicas for each SDFS file. We use quorum with W = 4 and R = 1. For every write, we want to make sure all the SDFS files are updated. Therefore, for each read, we just need to fetch 1 replica.

In order to make sure that every file is stored in the correct masters and slaves according to their hashed values, we may need to rearrange the sdfs files each time a node leaves or joins.

When a leave occurs, each machine first checks all the sdfs files it has. If this machine is a file master and the leave action refreshed its slaves list, then it sends a put to its new slave. If this machine should become a new file master, then it sends a put to its slave that do not have the file. For each file, there should be at most one put request for each leave.

When a join occurs, each machine first checks all the sdfs files it has. If this machine is a file master and the join action refreshed its slaves list, then it sends a put to its new slave and send a delete to its old slave that is no longer in its slave list. If this machine is no longer a file master, then it sends a put to the new file master and send a delete to its slave that should no longer hold the file. Delete does not happen when the total number of machines are less than or equal to four. For each file, there should be at most one put request and one delete request for each join.

For “put”: server will first check whether the localfile exist. Then it will check the ip addresses of master and slaves for this file according to hash value and virtual ring, then use TCP connection to send the file to the targets. A put is considered success when all servers storing this file. The target server will check the previous modify to the file and ask for confirmation if two updates is within 1 minute.

For “get”: sever will first check the ip addresses of master for this file according to hash value and virtual ring. Then it will ask the targets whether the file exists. Finally, it will use TCP connection to require the file from the targets.

For “delete”: server will first check the ip addresses of master for this file according to hash value and virtual ring. Then it will ask the targets whether the file exists. Finally, it will use TCP connection to send the delete request to all the targets and wait for delete confirmation.

For “ls”: server will first check the ip addresses of master for this file according to hash value and virtual ring. Then it will ask the targets whether the file exist. If the file exists, it will send back the ip address to the client.

For “store”: The server will return the SDFS File Information stored on the server to the client.

The use of MP1:

We need to use log files to debug the whole system since we cannot use IDE or other debuggers to debug it. What we can do is to write down different logs to see which part goes wrong. Therefore, it is very useful to use MP1 to get the log files and read it in one machine to see where goes wrong.

Measurements：

i. re-replication time for 40MB file upon failure:

	Average	Standard Deviation
re-replication time (s)	0.6332	0.08067032
bandwidth (MBps)	32.036	0.28360183

Here we use nload to monitor the average background bandwidth during re-replication. The average bandwidth may be less than the max bandwidth so that it may take less time than directly calculating the time based on bandwidth and file size.

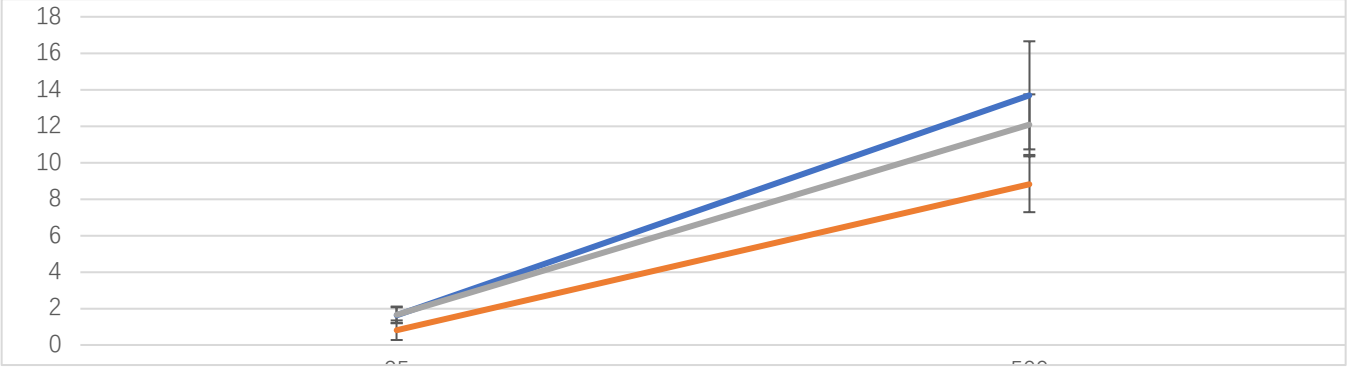
ii. times to insert, read and update:

1) file with size of 25 MB:

	Average	Standard Deviation
insert (put) (s)	1.629375	0.43826702
read (get) (s)	0.81	0.45003079
update (put) (s)	1.658	0.5369581

2) file with size of 500 MB:

	Average	Standard Deviation
insert (put) (s)	13.693	2.96409028
read (get) (s)	8.81466667	1.66153082
update (put) (s)	12.085	1.53077366



iii. time to store the entire English Wikipedia corpus to SDFS with 4/8 machines:

Here the English Wikipedia is the raw.en.tgz file with size = 1.35 GB

Put time	Average	Standard Deviation
4 machines (s)	31.7658	7.68386359
8 machines (s)	28.6888	4.02517754

Discussion for Measurement:

From the previous data and plot, we can see that the average replica time is very low, and the bandwidth is very high. This shows that our system is very fast.

For the insert, read and update time for 25MB and 500MB file, it will take longer time for insert, read and put 500 MB file on the SDFS than 25MB which follows our expectation since we need more time to transmit the larger file. The insert and update time is similar since they both need to copy the local file to all the replicas. For the read, it will only copy one file from one replica so it will take less time than insert and update. The ratio between the same operation between two size is similar, shows that the time is mainly depend on the bandwidth.

For the store entire English Wikipedia on SDFS for 4 and 8 machines. Since for this SDFS, we use 4 replicas for each file, it should take similar time for put the entire file to the SDFS for these two conditions and the result meets our expectation.