CS425, Distributed Systems: Fall 2017
Machine Programming 3 – Simple Distributed File System
Team Member: Haowen Jiang, Tiancheng Wu
NetID: haowenj2, twu54

Design:

The purpose of this project is to build simple distributed file system based on Mp2 – distributed group membership. The whole project can be separate into five functions: put file, get file, delete file, list file saving directory and list file name saved in local machine.

- Put-file: Since the system should tolerant 2 machines illegal exit, at least three replicas exist. Meanwhile, quorum methods should be applied in this project which means the whole system should have at least 3 up to date replicas. Put-file function should write at least 3 files every time. The idea of writing file to the other machines are based on TCP communication.
- Get-file: Reading file from the system and load it into the local machine. This function is also based on TCP communication since the TCP is much more stable than UDP. During reading progress, this function will compare several replicas information through analyzing the timestamp. The most recent one will deliver the file to the query your query machine.
- Delete-file: Sine every machine has other machines' file information, the current machine will send delete message to others when receiving delete command from the users. In other words, the related file will be delete from the local machine when each machine receives the delete message. At the same time, the metadata in global_file_info and local_file_info should be removed from the dictionary.
- List file saving directory: For every put-file function, the replicas file information should be sent to all other machines, which means every machine define a dictionary to save replicas information.
- List file name in local machine: Whenever receiving a writing request and deciding to agree it, the machine will save the file name into a local dictionary.

Except these upwards function, SDFS should deal with machines illegal exit and write-write problems. Whenever a machine which has certain replicas illegal quits from the system, the other machines which has the same replicas will elect a leader to send the replicas to a random machine which exist on the ring. For the write-write condition, whenever the write-file request happens on a same machine and the time gap smaller than 1 min, the current machine will ask the users to decide whether to rewrite the file.

Data:

	re-replication time (s)	bandwidth(byte)
1	0.299413	40000773
2	0.394633	40000776
3	0.364436	40000782
4	0.311234	40000782
5	0.290417	40000773
avg	0.3320266	40000777.2
standard dev	0.04526938	4.54972527

25M	insert(s)(25M)	get(s)(25M)	update(s)(25M)
1	0.439534	0.261217	0.38945
2	0.451007	0.191983	0.493709
3	0.476643	0.224819	0.38703
4	0.398383	0.216334	0.352517
5	0.361668	0.178836	0.431371
avg	0.425447	0.2146378	0.4108154
standard dev	0.04547804	0.03190284	0.05411749

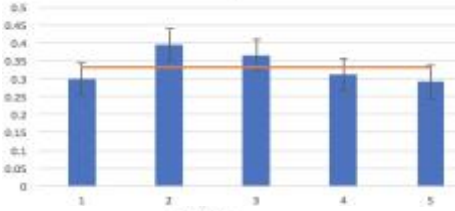
500M	insert(s)(500M)	get(s)(500M)	update(s)(500M)
1	8.418887	4.583297	10.498908
2	7.856483	3.957222	10.884127
3	7.965008	3.529003	9.982311
4	7.765859	4.481789	10.564054
5	7.919513	3.987895	8.566947
avg	7.98515	4.1078412	10.0992694
standard dev	0.25371512	0.42954105	0.91561842

	write-write conflict time(s)
1	0.00009
2	0.000088
3	0.000113
4	0.000085
5	0.000085
avg	0.0000922
standard dev	1.18195 * 10^-5

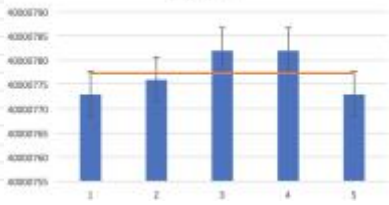
	EW time(s)(4 machines)
1	22.065538
2	20.025582
3	20.689485
4	19.895866
5	25.347917
avg	21.6048776
standard dev	2.26487061

	EW time(s)(8 machines)
1	23.429806
2	22.487869
3	21.638541
4	19.767447
5	19.534986
avg	21.3717298
standard dev	1.69558129

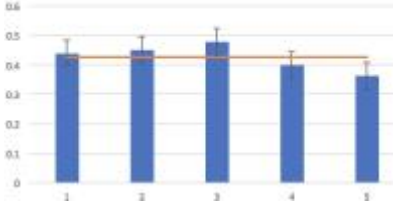
re-replication



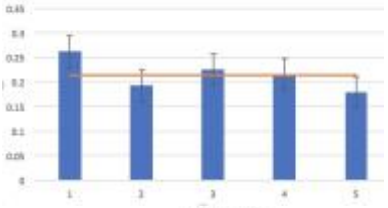
bandwidth



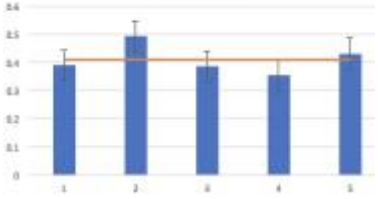
25M Insert



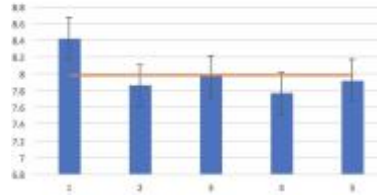
25M get



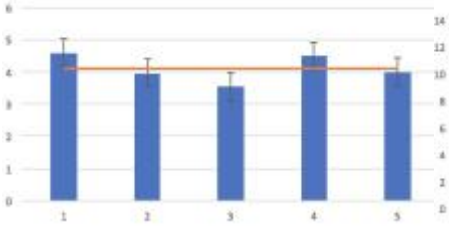
25M update



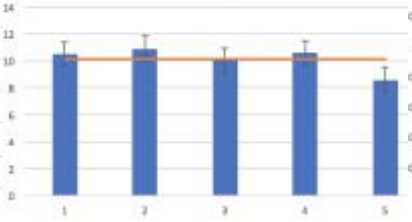
500M insert



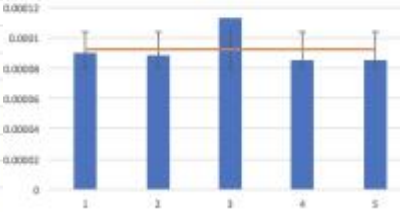
500M get



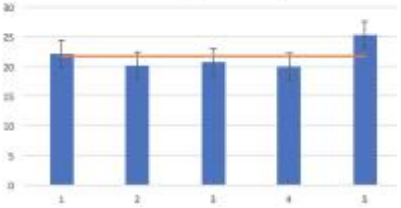
500M update



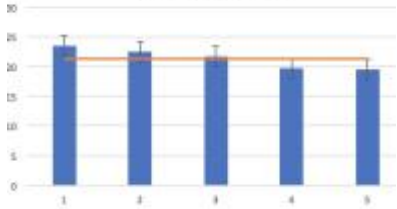
write-write conflict time (s)



EW time(s)(4 machines)



EW time(8 machines)



Above is what we expected: get/set time will increase if file size increases, but will not change significantly with more machines given large files. Write-write conflict detection should be fast, and message overhead should be small.