Yifei Teng (teng9)
Yu Wang (yuwang17)

CS425 MP3 Report

The algorithm we designed for determining replicas is to render a special hashcode in which given a key, it would generate an order of nodes which would go through all nodes and it guarantees that different key would has different order.

The replication strategy is that whenever a node is started, the node will search all the other nodes' files and check if there's one need to be replicated. If yes, it will then download that file. If any node drop, then all the other nodes would check if they also store the files that the dropped node has. If yes, meaning there are only two backup instead of three when the node drops, then the node which supposed to backup this file will be notified and start to backup this file.

The downloading algorithm is to traverse all backups of the file along a probe sequence given by the hash function, and find a downloadable one to download. The uploading algorithm is simply to concurrently upload the file to its three replicas.

To handle large files that may not fit in memory, our program streams them to disk as they are being downloaded/uploaded.

MP1's grep is useful here for debugging. We used MP1's distributed grep to search debug error text on the log file generated by our program and analyze for debugging.

Re-replication time and Bandwidth of a failure (with 10 machines)

	AVG	STDEV
Re-replication time	0.1718 s	0.048380781 s
Bandwidth after a failure	181.4740015 mb/s	16.42923536 mb/s

Rereplication happens pretty quickly after the node is down.

Time between master failure and new master being reinstated

Our implementation does not use any leader election mechanism, since the hashing function we used is sufficient to be served as leader as explained in the first paragraph.

Yifei Teng (teng9)
Yu Wang (yuwang17)

Time to *write/read* a file of size 20MB, 500MB (with 10 machines)

(write)	AVG	STDEV
20 MB File	0.1094 s	0.025481366 s
500 MB File	3.9576 s	0.461960357 s

(read)	AVG	STDEV
20 MB File	0.3212 s	0.319264311 s
500 MB File	1.0826 s	0.40149004 s

The average time taken to **read** a file, regardless of file size, is a lot faster than the average time taken to **write** a file. And the larger the file, it takes more time to read and download that file. But the time taken to write the larger file does not increase proportionally with file size, most likely due to ramp up time of the TCP connection.

Time to store the entire English Wikipedia corpus into SDFS with 4 machines and 8 machines

	AVG	STDEV
4 machines	5.1922 s	0.551834327 s
8 machines	5.1831 s	0.512564287 s

The time taken for 4 and 8 machines is quite similar, since our design only replicates to three nodes.