CS425 MP3 - SDFS (javeria2, ssong38)

Design*:*

We used the failure detector (MP2) along with an added layer for the SDFS. Our new algorithm is an extension of MP2, All features are listed as under -

1. **Master node:**
   We have a primary master node and two secondary master nodes. When the primary node fails, one of the secondary masters are elected and promoted as the primary master. The primary node handles the file metadata by updating its filemap, it then heartbeats the updated filemap to all other nodes. We store metadata on all nodes, like Gnutella system.
2. **Putting a new file into the system:**
   When a user wants to put a file into the SDFS, the system makes a copy of the file into the local SDFS folder, which only stores the file which are in the SDFS system. The primary node updates its file map. The node will also send the file to next two successors to make a replication on the successor node.
3. **Deleting a file from the system:**
   When the user wants to delete a file from the SDFS, the system will send the delete message to the primary node. The primary node will send out the message to the locations which stores the file and ask them to delete the file locally. Meanwhile, it will remove the entry from the file map.
4. **Node Leave and Crash:**
   We achieve consistency by looking at the membership list. If the status of the node is not alive, we will update the file map by deleting all of the records related to that specific node.
5. **ls a file:**
   If any node wants to query the location of a certain file, it will be done locally since every node stores the metadata containing file information.
6. **Store:**
   If any node wants to get the file list of the local machine, we will check the local folder for the SDFS file storage and return all filenames to the user.
7. **Getting a file:**
   If any node wants to get a file from the system, the node will send a query to the primary master node, get one of the file locations, send a message to the storage location and the node which has the file will return a copy to the user.

Testing*:*

1. Re-replication time and bandwidth -

Round	1	2	3	4	5

Re-replication Time (Unit: s)	0.22	0.24	0.21	0.22	0.25

The re-replication time is calculated from the time after one of nodes failed to the time that another node get the file which was on a node that crashed.

Bandwidth consumption -

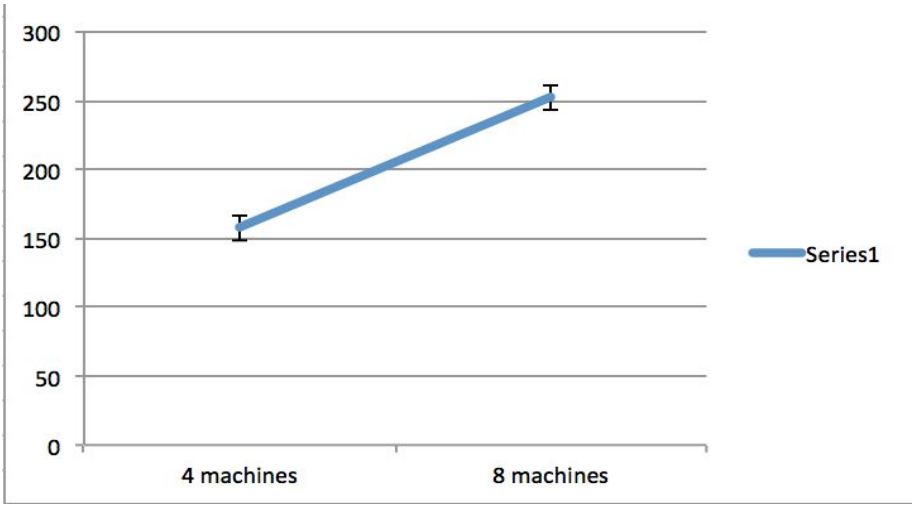
2. Times to insert, read and update , file of size 25MB, 500MB under no failure (Unit:s) -

Action	Insert	Read	Update
25MB (Average)	0.23	0.06	0.26
500MB (Average)	2.6	1.1	4.1

3. Time to detect write-write conflicts for two consecutive writes within 1 minutes into the same file -

The average time for detecting the write-write conflicts is 0.08s. Standard deviation is 0.011s. This is as expected because we store the file-timestamp map on each node. The node can detect if two consecutive writes happen within 1 minutes really fast.

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



After the test, with 4 machines, storing takes an average of 157.6s and the STDEV is 9.52s; with 8 machines, storing takes an average of 252.4s and the STDEV is 11.61s. The time works as we expected. The reason is that for storing the 4 replications of the file, we are actually make 6 copies; for storing the 8 replications of the file, we only make 9 copies which is one more put operation in our system. Thus, the time difference is not large.