Dongming Lei (dlei5)
Qixin Zhu (qzhu3)

Design

In our SDFS, we have 5 major components: udp-based group membership and failure detection algorithms, tcp-based remote procedure call for distributed grep and reliable message transfer, SDFS manager to handle distributed file system logic and finally a CLI component to interact with user.

Failure Detection and Group Membership

Same as completed in MP2.

Leader Election

Our leader election algorithm is similar to Bully algorithm choosing lowest machine number based on the group membership information. Once a current leader's failure is detected, re-election will be initiated. All remaining processes will vote for the lowest machine number they know. Once a node receive votes, it will use quorum to decide whether it is elected as the new lead. If true, it will send messages to all nodes asking them to update their master. Thus the next lowest machine number remaining in the membership list will be elected as new master/leader.

Distributed FileSystem Logic

We support the following commands: PUT, GET, DELETE, LS, and STORE.
PUT request is first routed to master. Master checks that if the target sdfs file was recently updated within 60 seconds, if so, the master will ask the client for confirmation. Otherwise or the confirmation was received within 30 seconds, master will select 3 different target nodes to store this file and then return this information to the client and store the metadata. Once client received this information, it will send the file data to those 3 target nodes. When target node received the data, it will save the data to its own storage and record its metadata. At client side, it will use QUORUM to decide when the PUT is finished, that is if we put to 3 replicas, we return success once 2 are done.

GET request first ask master to get who has the file and contact them to get file data. QUORUM is also used here, that is if we read from 3 replicas, we return the one with highest version once 2 copies are received. During this process, if some replicas version is less than the latest version, we'll also initiate a stale version repair for that node.

DELETE request first ask master to get a list of who has the file. When master get this request, the master will remove this file's entry from the metadata. The client will then ask the replicas in this list to delete the file and remove from its local metadata.

Replication Strategy

Per requirement, our SDFS is tolerant up to two machine failures at at a time, therefore, we want to maintain 3 replica per file. After failure(s) get detected or stale version detected, we initialize a repair process to copy data from the remaining node to new nodes that are still alive. Therefore, at all time we try to keep 3 replica per file.

Usefulness of MP1

MP1 is useful for us. It provides us a great convenience to get local log messages without having to log into the machine and type in multiple commands to retrieve and grep the log.

CS 425 MP3

Dongming Lei (dlei5)
Qixin Zhu (qzhu3)

Measurement (Unit in seconds)

Rounds	1	2	3	4	5	Avg	STD
re-replication time on failure	1.18	1.27	1.82	1.17	1.35	1.358	0.2401
bandwidth on failure MBps	40	35.9	22.8	38.2	33.8	34.14	6.0437
times to insert 25MB	1.84	1.91	2.99	1.82	1.83	2.078	0.4571
times to insert 500MB	16.94	22.27	26.4	31.55	26.84	24.8	4.9085
time to read 25MB	1.85	1.68	1.67	1.72	1.71	1.726	0.0647
time to read 500MB	7.13	7.96	7.05	7.57	7.11	7.364	0.3509
time to update 25MB	1.81	1.84	1.83	1.83	1.83	1.828	0.0098
time to update 500MB	25.76	29.4	21.85	34.6	37.62	29.846	5.7248
time to detect write-write conflict	1.03E-05	1.09E-05	9.77E-06	1.02E-05	9.25E-06	1.01E-05	5.51E-07
time to store Wikipedia with 4 machines	43.04	44.77	44.78	39.71	30.65	40.59	5.3030
time to store Wikipedia with 8 machines	37.86	45.97	28.85	31.55	40.57	36.96	6.1654

Discussion

(1) Re-replication time and bandwidth upon a failure: Bandwidth is slightly less than 40MBps and the time cost is slightly more than 1s, which is expected since the total amount of traffic is 40MB.

(2) Time to insert, read and update: Time to insert and update is close, which is as expected since the underlying mechanism is the same. Time to read is typically less than time to insert, this may be the result that we only need to actually transfer 1 copy of file to the client instead of 2 (when inserting).

(3) Time to detect write conflict: the conflict detection is trivial. This is because the master can check the record quickly, without reading the file on disk.

(4) Time to store the Wikipedia corpus: there is no significant difference between a 4-machine system and an 8-machine system. The main time cost comes from file transfer through network and there is always 3 replica needed to be transfer regardless the number of machines.