CS 425 - Distributed Systems MP3

Group 1 – Anirudh Jayakumar(ajayaku2) and Banu Muthukumar(muthkmr2)

Introduction: We have built a simple distributed file system for MP3 with Master-Worker architecture. The Leader/Master election is done through the bully algorithm.

Design choices:

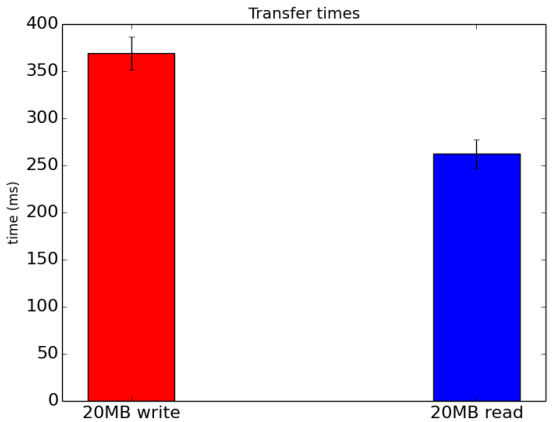
- **Membership List and Failure detection:** As in MP2, each node maintains a membership list that maintains information about all the nodes that are alive in the network. There is periodic heartbeating to *k* randomly chosen nodes to send the membership lists at regular intervals.
- **Introducer:** The introducer is the point of contact for any new incoming member to join the existing SDFS group. The introducer assigns a unique serial number for each node. The introducer IP is fixed. If the introducer fails, no new nodes can join till it is back up. There is periodic checkpointing of data to aid the introducer to re-enter the group.
- **Leader:** The leader is the primary administrator for the SDFS. The leader keeps track of all the existing files in the system, their locations and also routes get, put and delete requests appropriately. Additionally, the leader oversees the existence of the specified number of replicas for all the files. On startup, the introducer is deemed as the leader. On failure of the introducer, the new leader is elected. The leader is elected using the Bully election algorithm.
- **File Reports:** The file list for every node is periodically sent to the leader so that the leader can keep track of the locations of all the files and their replicas.
- **Adding files**: To add any new file to the system, the node containing the file (client) sends an add request to the leader. The leader responds with a randomly chosen IP for placing the file. The client then copies the file over to the given IP (master doesn't deal with any of the data except it's own). Once this is done, the node containing the new file copy (SDFS) will replicate the file in two other randomly chosen nodes.
- **Getting files:** To get a file from the system, the client requests the master/leader for the file. The leader responds with the list of IPs where the file is present. The client establishes a connection with one of these nodes and requests for the file.
- **Deleting files**: To delete a file, the client sends the filename to the leader. The leader looks up the locations of the file and deletes all the copies of this file.
- **Replication**: Since we assume a maximum of two failures, each file is replicated 3 times to ensure that no data is lost. Even if a particular node fails, there will be two other replicas of all the files that the failed node held. The leader periodically checks the replication count of all the files. This way, any replicas lost due to a failed node are eventually reinstated in other nodes (randomly chosen). If, due to any reason, there are more replicas than the required amount, the extras are deleted.
- **Communication**: The introducer VM runs apache thrift RPC service. All the participant nodes have a proxy to the thrift service running on the introducer. UDP messaging is used for membership gossiping and file report. File operations like get, delete and add are handled via the thrift RPC service.

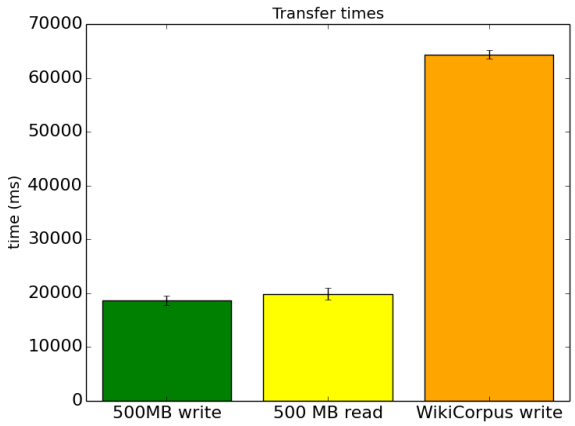
- **SDFSClient:** Client is implemented as a separate entity, which could reside inside or outside the cluster. The client connects to the introducer to get the current leader information. All file operation and monitoring commands are directly made at the master through the thrift service.

MP1: The various failure detection events, leader election progress, file operations etc were logged and saved in a file. On searching for the expected keywords, we were able to retrieve the logs along with timestamps quite easily which helped us in identifying unexpected behavior.

Experiments:

1. Re-replication time - Once the replication manager decides on re-replicating, the experiments shows a **mean latency of 655.4 ms** with a **standard deviation of 23.58 ms** when one replica needs to be created and a **mean latency of 1061 ms with a standard deviation of 57.16 ms** when two replicas are created. For each replica transfer a total two synchronous thrift calls are made. Internally, thrift may have multiple TCP calls for each RPC invocation.
2. Transfer times: The transfer times were studied for read and writes on 20 MB and 500 MB files. The latencies are shown in the plots below. The write and read latencies for **20 MB file is 369 ms** and **262 ms** respectively. The lower latency of the read could be due to the quicker lookup as compared to adding new entry to the table. It could also be due to the CPU caching of the hash table used to store file information. The write and read latencies for the **500 MB file is 18665 ms** and **19893 ms** respectively. The **Wikipedia corpus**, which is around 1.3 GB took around **64357 ms** to write to the SDFS. All the statistics are tabulated in the table below.
3. The **mean time to elect the new leader** once the failure is detected is **45.8 ms** with a **standard deviation of 3.34 ms.** The absolute time after the failure will depend on the failure detector interval. We set the interval for our experiments to 6 secs. Therefore an upper bound for the leader election after failure will be (detection interval + detection time + mean_time to elect leader).





Operations	20MB write(ms)	20MB read(ms)	500MB write(ms)	500MB read(ms)	1-replica (ms)	2-replica (ms)	Wikicorpus write (ms)
Mean	369	262.4	18665	19893	655	1061	64357
SD	17.47	15.33	944	1134.38	23.58	57.16	818