**CS425 MP4 Report**

**Group 4: Yi-Hsin Chen(yihsinc2), Che-Lin Huang (clhuang2)**

**Design**

**Master Failure**

To deal with master failure, we have two assumptions: **client doesn’t fail** and **workers don’t fail**. Under our design, our system **could finish the graph computing task** even the original master fails. The key idea is using client/worker to help backup master restore the progress of the task.

On the client side: once a client sends out a request for graph computing, it will first send the graph to the master. After the maser received the graph raw data and save it into the SDFS, it will synchronize task-related information with backup master. After the synchronization is done, the master will reply a “OK” message to client. This is the check point at the client side, if the master fails before the checkpoint, the client will simply resend task request/graph to the backup master. If the master fails after the checkpoint, since backup master has already gotten the task information (sent by the original master), the client will only **re-trigger** the task (w/o sending graph again).

On the worker side: in each iteration, we could divide the worker status into two phases: **GRAPH RECEIVING / GRAPH COMPUTING**. Once the backup master is up and re-triggered by the client, it will ask workers for their status to decide its action.

1. GRAPH RECEIVING: if master fails before each worker receives the EOF message from the master, workers will send a message to the backup master to ask for re-distribution of the graph.
2. GRAPH COMPUTING: if each worker successfully receives its sub-graph, after computation of the current iteration is completed, each worker will send the current iteration number to the backup master, and the backup master only needs to update its iteration number. And the task could go on.

**Worker Rejoins/Fails**

Before the start of each iteration, master will check the worker pool. If at least one worker fails, it simply re-partition the graph and re-distribute it among workers. If at least one new worker joins the cluster, the master will collect the computation results from existing workers first and then re-distribute the graph among the new pool of workers.

**../圖片%201.pdf ../圖片%202.pdf**

**Quorum Read/Write**

To write a file sdfsfilename, the client will send request to the corresponding coordinator, and the coordinator will replicate the file to its first 2 successors, once 2 of the 3 (quorum condition) writing processes are done, the coordinator will inform the client that the writing process is finished. At the same time, the coordinator will still try to write the file to the last replica.

To read a file sdfsfilename, based on our design, the coordinator will always keep the newest version of sdfsfilename by caching, then the coordinator could directly return the file back to the client, which is fast.

**Write-Write Conflict Detection**

For detecting write-write conflict regarding to sdfsfilename, the client will first send a message to the coordinator to ask it check the timestamp of sdfsfilename. If the last update time of the file is within 1 minute, the coordinator will ask the client to confirm the write.

**Usage of MP1**

We use MP1 to grep the log files from different VMs, which tremendously enhance the debugging efficiency since we don’t need to go through all the VMs one by one to check logs.

**Experiment**

1. The re-replication time is measured as the time from when a node’s failure was detected to when the re-replication was done. The bandwidth is the replica size divided by the re-replication time. Both the re-replication time and the bandwidth used are fairly good, which meets our expectation.

../Desktop/1.pdf../Desktop/2.pdf

1. The results are shown below. As expected, the larger file takes longer time to insert, update and read. Read is faster since the coordinator always has the newest version of sdfsfilename due to caching, the file is read directly from the coordinator. The insert time and update time are approximately the same.



1. Time to detect write-write conflicts for two consecutive writes to the same file within 1 minute is measured as the time between when the client sent duplicated put request and when the confirmation prompt showed up. The time is very fast since it doesn’t involve file I/O and the coordinator only need to compare the timestamp in its SDFS and the put request time.
2. The time to store the entire English Wikipedia corpus into SDFS is measured as the time between when the client launched the put request and when the client received the message indicating quorum write is done. The more machine makes more traffic in the network, thus it takes more time to store the file of same size.