CS425 MP3 Report

Taipeng Liu(taipeng2) & Xiang Li(xiangl14)

1. Design

Our TCP-based SDFS is built on top of MP2’s reliable failure detection protocol. We employ Remote Procedure Call (RPC) to implement the client-server model in this system. Each machine can both act as a client to request, and also runs “datanode” server to store file replicas. Also, as a master-slave model, we will also have exactly one machine running “namenode” server to take care of replica assignment. Our election principle is always pick smallest nodeID among full membershiplist. When executing put(write), get(read) or delete functions, client will dial specific VM’s address:port 8885 to establish a connection with each “datanode” server. Then it may call remotely to perform all these functions as is locally. It’s proved to be efficient by using Go’s RPC package.

**Replication Strategy:** Master’s main role is keep listening on port 8884 and figure out the list of nodeID who should store the replicas for the file when being called remotely from client. We apply passive replication with 4 replicas for each file to deal with at most 3 simultaneous failure. Also, Specifically, master will maintain “FileMap” to store metadata for each sdfsfile, and “NodeMap” organized with nodeID to check each correct VM’s sdfsfiles. Each time client inserts, updates, deletes or failure happens, we updates this two crucial containers.

**Features:** We use a quorum with W = 4 and R = 1 which achieves fast read and assures consistency in that at least one replica will be alive in the system under massive failure. Besides, we use mutex to count on the number of responded replicas. For put & delete request, we will wait until all W = 4 replicas finish their write operations and return. However, we will return as long as the first replica nodes finish for get request.

**Usage of MP2&MP1:** Our MP3’s implementation highly relies on MP2’s membershiplist to check failed nodeID then ensures fast and accurate re-replication. Besides, if MP2 fails to provide completeness on failure detection, we will be under risk of losing files. MP1 is useful for querying log files remotely from one client. As before, we create “MP3\_[# of VM].log” file on each VM for reference. We print log with keyword of corresponding function name or which will helps us pinpoint quickly, i.e. “grep -E putfile” to get related logs of function putfile.

2. Measurement

(1)8 VMs 40MB File:

Re-replicate time (Response time or count on transfer time?): 12.05ms 8.45ms 208.24ms?

Bandwidth: including 40MB file + Msg

(2) 8 VMs

25MB File:

Insert 616.866565ms ; 382.08644ms; 399.369239ms; 204.361979ms; 351.76362ms

Update 184.425278ms; 180.84445ms; 171.301222ms; 196.461731ms; 215.656728ms

Read: 182.313103ms; 152.2091ms; 100.249336ms; 153.98633ms; 168.475936ms

500MB File:

Insert 4.73169899s; 5.214846242s; 5.331799916s; 4.517704529s; 4.108312937s

Update: 4.375422235s; 4.386429906s; 3.782075332s; 5.303016344s; 4.275420965s

Read: 3.154962569s; 2.439896312s; 4.611378909s; 3.495656439s; 3.517878664s