CS425 MP3 Report

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

Design

Coordinator (Master Server) Election & File Storage

We use **per-key coordinator** to make sure **totally order** and make our system **scalable**. For totally order, we further implement a queue on the coordinator to receive the put file request from clients. The coordinator will ask clients to send files according to queue which will satisfy totally order. We adopt a **Cassandra-like algorithm**. First, to decide the position of a node in the virtual ring, we pass the ID of each node into a consistent hash function, then we mod the returned hash value by 2^8. Second, each node has the full membership list (use MP2 to maintain the full membership list). To decide which coordinator we need to contact when we want to do file operations regarding to sdfsfilename, we do the same thing to sdfsfilename to decide its value in the ring, then the first node (we call it targetNode) whose value is equal or larger to the value of this file is the coordinator. To store a file in the SDFS, the procedure is the similar.

Replication Strategy

Since the SDFS is required to be tolerant up to **2** machine failures, we store **3 replicas** for each file to prevent file loss, and we simply replicate the file to the **first 2 successors** of targetNode.

To deal with re-replication, first we maintain a neighbor list containing the **first 2 predecessors and first 2 successors** for each node. By doing this, files on a node might need to be moved or delete **only when the neighbor list of this node is changed**. We summarize our strategy as follows:

1. **Node Added:**
   (1) The newly-added node will ask its **first successor** to check its SDFS files: for each file, once the new node is the targetNode corresponding to the file, the first successor will send it to the new node.
   (2) For all existing nodes whose predecessors are changed, they will check the SDFS files stored on them: for each file stored on a node, if the corresponding targetNode is no longer the first two predecessors of the node, we **delete** the file on the node.
2. **Node Removed:**
   If the successors of a node are changed, we check all the SDFS files stored on this node: for each file sdfsfilename, if the node is the new targetNode corresponding to the file, we send the file to its **new first two successors.**

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

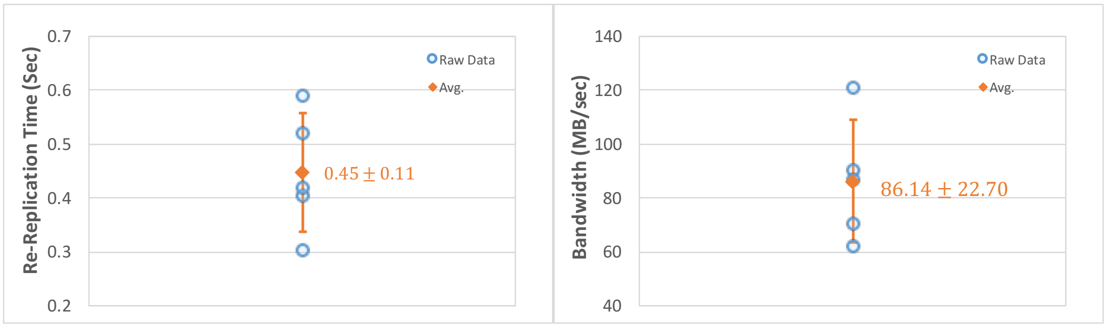
For detecting write-write conflict regarding to sdfsfilename, the client will first send a message to the coordinator to ask it check the timestamp of sdfsfilename. If the last update time of the file is within 1 minute, the coordinator will ask the client to confirm the write.

Usage of MP1

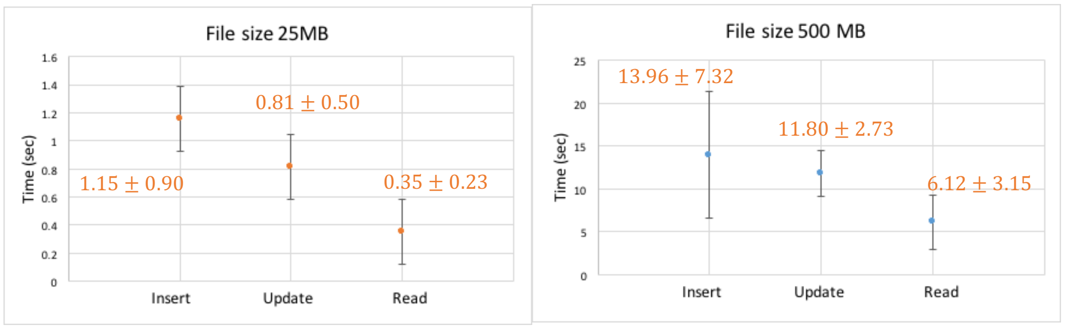
We use MP1 to grep the log files from different VMs, which tremendously enhance the debugging efficiency since we don't need to go through all the VMs one by one to check logs.

Experiment

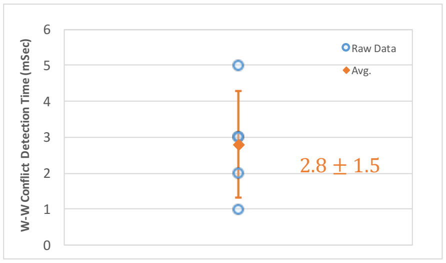
i. The re-replication time is measured as the time from when a node's failure was detected to when the re-replication was done. The bandwidth is the replica size divided by the re-replication time. Both the re-replication time and the bandwidth used are fairly good, which meets our expectation.



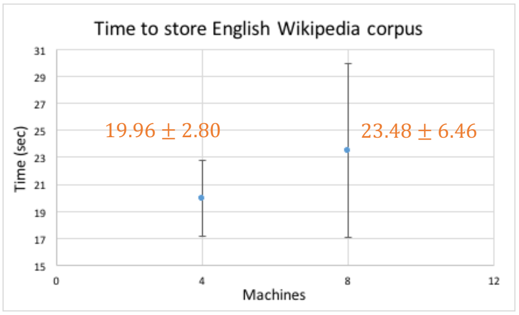
ii. The results are shown below. As expected, the larger file takes longer time to insert, update and read. Read is faster since the coordinator always has the newest version of sdfsfilename due to caching, the file is read directly from the coordinator. The insert time and update time are approximately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



iii. 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.



iv. 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.


Time to store English Wikipedia corpus
Time (sec)
Machines
19.96 ± 2.80
23.48 ± 6.46