**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**. To be more specific, 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 28. Second, each node has the full membership list (we 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 by put localfilename sdfsfilename, the procedure is the same as that we select the coordinator.

**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 the case that **node added/removed**, 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:**
2. The newly-added node will ask its **first successor** to check the SDFS files stored on it: for each file, once the new node is the targetNode corresponding to the file, the first successor will send it to the new node.
3. 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.
4. **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 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**

1. The measured extra bandwidth (excludes heartbeat signals) of a node leaves/joins/fails a group with 4 members is: 13/28/39, which matches the calculated results. Let  
2. The results are shown below. As expected, the higher the message loss rate, the higher the false positive rate.
3. Dd



1. dsdsd