MP3-Simple Distributed File System

Shaowen Wang (shaowen2)

Songwei Feng (songwei3)

System Design

The Simple Distributed File System can be divide into three section : The file system section, the failure detector section, and the election section. And we used MP1 to query the log files which generated by log4j during execution, and it largely facilitates the process of debugging.

For the file system, all servers are able to put/get/delete files in the distributed system. One of the server is the master server, which stores a list of all the files in system. It is responsible for deciding the location that every file should stored and where are the replications of files. Other servers query master for locations to update, get or delete the file. If the master is down, the new master will be re-elected quickly. To make writes and reads fast, we used a **quorum** to handle writes and reads, except the first put. In addition, we implemented the functions of totally ordering all updates of a file and requesting confirmation from the second user when there are two updates to one file within 1 minute. During the put, the client server request 3 servers from the master to store the file' replicas and we defined the **replication strategy** as storing the replicas in the server and its successors. Here we are doing **active replication**. During the get and update, the master will return two of three servers' address which stores this file to client server, and the client server will get latest file from the two servers. For delete, the client server sends requests to master, the master will relay the request itself to individual servers containing the file. When the number of one file's replica is less than 3, the master will search the membership list and asks the successor of failed server's successor to replicate the file if it doesn't store the replica, and so on if there are at least three servers. And when there are less than 3 servers, make sure every one has the replica, which ensures data stores in the system can tolerate to two machine failures at a time.

The failure detector section based on the Distribute Group Membership that we implemented in MP2. We implement the failure detection with Bidirectional Ring-style heart beating, and each machine in the system always has a full membership list , two successors and two predecessors.

The leader election section based on the pre-fixed leader group with different priority: Servers in the system will check the IP address of the current leader before the operation, and get the leader with the highest priority. Thus, every server in the system is consensus to the current leader. In addition, when the leader is in work, it will send copies of the file list to the other member in the leader group regularly, in this way make sure the steadiness of the file list.

System Measurements

Re-replication time and bandwidth

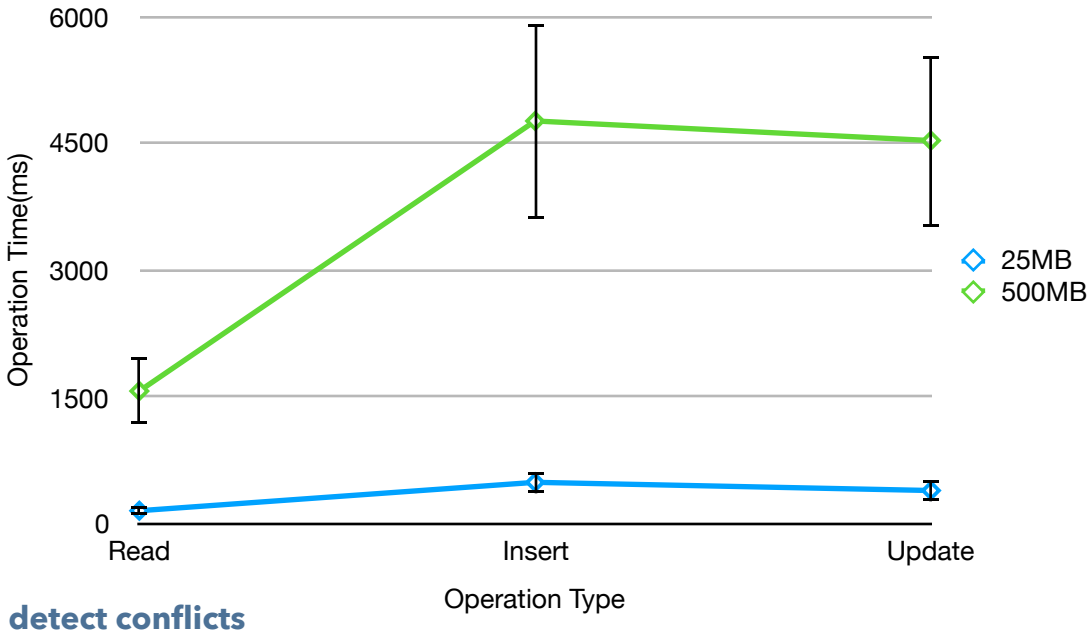
The **re-replication time** of the Simple Distributed File System **average** over 5 readings is **183 ms**, which about 150ms for transferring the file and about 33ms for average time elapsed between a failure and the master detects the loss of replication. The re-replication time is measured as the time between a server storing one file fails and replication has been done in a new server. The **standard deviation** of the re-replication time is **10.3ms**.

We ran our experiment for just one 40MB file per server. the **bandwidth** is **148 megabytes/sec** for the file transfer with a standard deviation of **17 megabytes/sec.**

Time to read ,insert and update

As show in the below plot, we plot the **average** operation time with **standard deviation bar** at 6 data points the system after we divided the operation into read, insert and update separately with 25MB file and 500 file. For each data point, we run 6 trials. When comparing these plots, we can see the Insert time is almost 3 times the read time, this is expected because we write to 3 servers and only read from latest one. And update time is less than the insert mainly because the quorum is used. In addition, with the file size increases, the operation time grows.

	Read Mean	Read Std	Insert Mean	Insert Std	Update Mean	Update Std
25MB	153ms	36ms	489ms	130ms	392ms	126ms
500MB	1569ms	403ms	4765ms	1161ms	4533ms	1008ms



Time to detect conflicts

The time to detect write-write conflicts for two consecutive writes within 1 minute to the same file in our system is **average** to **501 ms** with a **standard deviation** of **37ms**.

Time to store Wikipedia corpus into SDFS

The **average time** to store the entire English Wikipedia corpus(1.35GB) into SDFS with 4 machines is **9.7 s** with a **standard deviation** of **2.3s** and **11.2s** for 8 machines with a **standard deviation** of **2.4s**, which is under our expect since with more machines the query time will increase.