Report

CS425-MP3-Group 9

System Design

There are 4 different services in each SDFS server : the failure detector, the election layer, the file server layer and the master layer. The entire system consists of many identical SDFS servers and one SDFS proxy server. The SDFS proxy server is assumed to be the endpoint for this SDFS service. So all clients communicate with this SDFS proxy server to get the current master among the SDFS servers. The SDFS proxy server acts as the introducer in group membership protocol and does not store any file as part of its service. If the SDFS proxy fails we assume that no new node can join the SDFS service cluster and no client requests can be handled until the SDFS proxy is back again. The failure detector uses SWIM algorithm that we implemented in MP2. The election layer uses a simplified bully algorithm. The algorithm is as follows : If a node detects failure of master using the failure detector layer, it calls for election and checks if it is next in line to be master. If not, it waits for coordinator message (and times out if not received). If it is the next in line to be master, it multicasts coordinator messages to all members in the group which send back okay messages if they agree. The check if it is potential master is possible by using group membership list. Finally the SDFS proxy server is let known who the master is.
The master stores a list of which files are replicated by which nodes and periodically checks the replication of all files. if replication drops to less than 3 (due to some node failure), it asks a randomly chosen alive server to replicate the file. The master also handles requests from clients. During put, the client requests 3 servers from the master to put data in and then the client writes to as many of these 3 as it can. So we are doing active replication. This ensures that despite 2 failures at a time, we still have one copy of the file still left. For get request, the client gets list of servers with file from master and then gets the file from server. For delete, the client sends the request to the master, the master then relays the request itself to the individual servers containing the file.

Experiments

1. Re-replication time and bandwidth

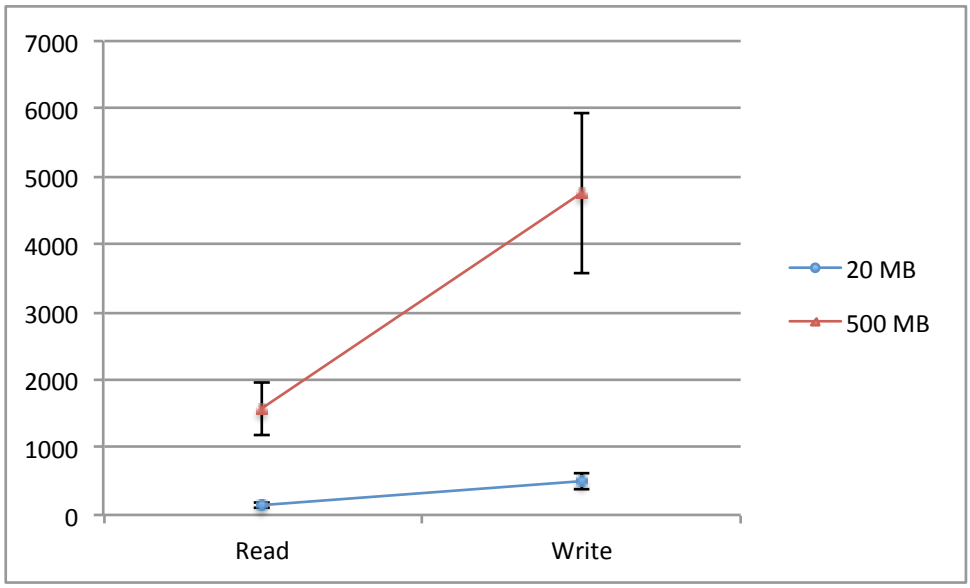
The re-replication time measured as the time between a node storing one file fails and master receiving the notification that replication has been done in a new node. The average re-replication time (average over 5 readings) was found to be **191 ms with a standard deviation of 11.2 ms**. In this 191 ms, about 145 ms is the time to transfer the file between servers (average time to get a file as we will see later in 3). The rest is the average time elapsed between failure and master detecting replication loss.
The bandwidth measurements are broken down into the different services in the system. The failure detector service shows the same bandwidth characteristics as before : bandwidth for first detector goes up to **206 bytes/s**, for others it goes up to **92 bytes/sec**. In the file server service layer, the bandwidth increase is only in the new replication server of the file and one of the remaining 2 not failed servers storing the file. The bandwidth for the file transfer has been measured to be **156 megabytes/sec**. The election layer has 0 bandwidth usage in case of the failed node not being a master. If the failed node is master, the bandwidth usage depends on number of files replicated per server. We ran our experiment for just one file per server, the bandwidth increases to **476 bytes/sec** for the newly elected master. The bandwidth in other servers increases to **39 bytes/sec** on average.

2. Time between master failure and new master elected

The time between master failure and new master being reinstated in measured to be **86 ms** on average with a standard deviation of **13.1 ms**. This time includes the failure being detected by the failure detector layer, the election algorithm duration (master being elected, master multicasting coordinator messages and master getting the list of files replicated in them from the other servers).

3. Times to read and write 1 file

	Read Mean	Read std Deviation	Write Mean	Write Std Deviation
20 MB	145 ms	37 ms	497 ms	127 ms
500 MB	1574 ms	397 ms	4744 ms	1194 ms



In the above chart the Y-axis has time in milli secs.
We see that the time to write is approximately 3 times the time to read. This is expected because the client attempts to write in 3 servers during write, but reads from the 1st available server during read. The standard deviations increase proportionately. We also see that the time for 500 MB is not 25 times the time required for 20 MB file. This might be because of the effect of the times taken by header messages and end messages which are fixed in both and do not increase on increasing file size.

4. Time to store wikipedia corpus in SDFS

We found wikipedia compressed corpus of size 12 gb. The time required to write the file in SDFS was measured to be 110 seconds and the time to retrieve back from the SDFS was measured to be 28 seconds.