CS425 MP3 Simple Distributed File System Report

Yihan Zhang(yzhng249)

Hang Yuan(hyuan13)

We split VMs into workers and a single master. Master will retain a list of alive VMs, a map from file name to its list of all VMs(replicas), and a map from file name to its latest version. In each worker, all local files are stored in local directory and all sdfs files are stored in sdfs directory which will be created when worker joins the group.

In each put operation, one of the worker sends message "put localfilename sdfsfilename" to master and master will choose 4 VMs currently in its list and responds to the worker with message "version-num vm_a vm_b vm_c vm_d" where "version-num" is the latest version of the file that is being uploaded(sdfsfilename). Then the worker will write the file to those 4 VMs. Regardless of it's a insert or update on file, we always write to all 4 VMs. The pointer is incremented by 4 every time the put operation happens so that files will not be stored in the same 4 VMs every time.

In each get operation, one of the worker sends message "get sdfsfilename localfilename " to master and master will respond with "version-num vm_a" where version-num is the latest version of the file requested and vm_a is the first replica of the list of replicas that has the file "sdfsfilename " in it.

In each delete operation, one of the worker sends message "delete sdfsfilename" to master and master will responds with message "vm_a vm_b vm_c vm_d" and the worker will sends command "del_file fileName" to those 4 VMs and they will delete the file "fileName" locally.

In each list operation, one of the worker sends message "ls sdfsfilename" to master and master will respond with message "vm_a vm_b vm_c vm_d" so that the worker will find the ip addresses of them and print in the terminal.

In each get-versions operation, one of the worker sends message "get-versions sdfsfilename numversions localfilename" to master and master will respond with message "version_1 version_2 … version_n vm_a" where version_1 to version_n is the latest "numversions" number of versions of the file "sdfsfilename" and vm_a is the first replica of the list of replicas that has the file "sdfsfilename" in it.

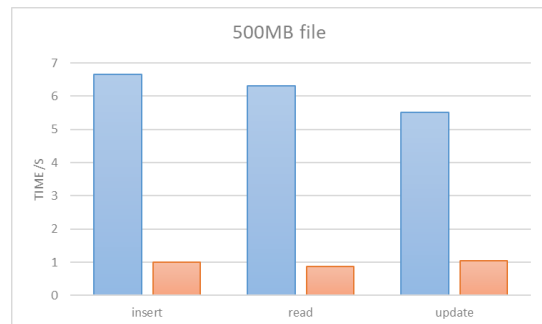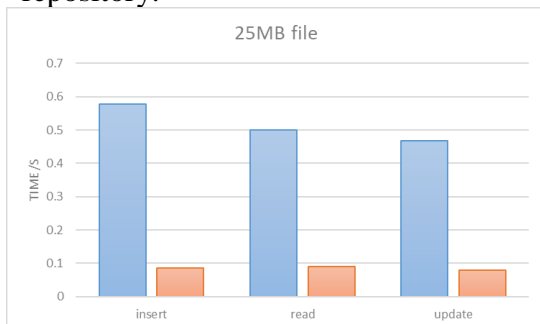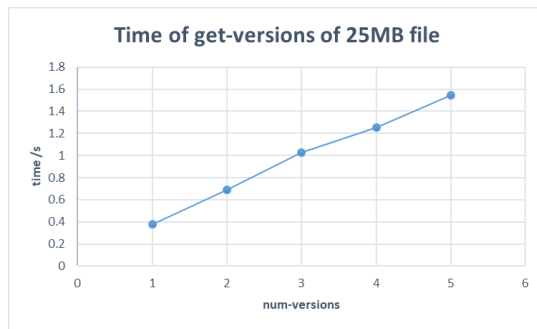In each store operation, each worker will print all files currently in its local repository.

Figure 1

Figure 2

Time of get-versions of 25MB file

time/s

num-versions

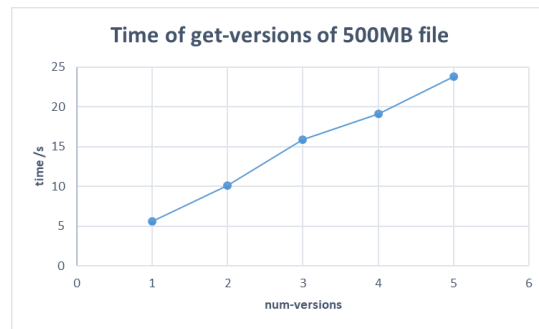Time of get-versions of 500MB file

time/s

num-versions

Figure 3

Figure 4

(i) In our sdfs system, the re-replication time upon a failure for a 40MB file is about 2 seconds. It takes 1.2 seconds to detect a failure and 0.8 seconds to replicate the 40MB file into a new replica host. The bandwith usage upon a failure for a 40MB file is 20000065 bps, which is calculated by the sum of 40MB TCP package plus 130B UDP package divided by 2 seconds.

(ii) Figure 1 and 2 show the average and standard deviation of insert time, read time and update time of 25MB file and 500MB file separately. The average insert, read and update time of 500MB file are almost ten times longer than those of 25MB file. As we can see from these two figures, both insert time are longer than update time. This may be caused by virtual machine cache. The read time is a little bit shorter than insert time. This may because in our design, when a worker would like to get a file, it will only contact to one replica host.

(iii) Figure 3 and 4 show the average get-versions time of 25MB file and 500MB file at the number of version from 1 to 5. The average get-versions time increase as the number of version goes up. This satisfy our expectation because larger num-versions means more files need to be transferred in the network. The file transmission time of larger num-versions is much longer than that of smaller num-versions.

(iiii) The time to store the entire English Wikipedia corpus into SDFS with 4 machines is 62s. The time to store the entire English Wikipedia corpus into SDFS with 8 machines is 68s.

How useful MP1 is for debugging MP3

When we were debugging for the problem that two replicas failed, we print all files that was being held in that replica to the log file and print list of replicas of each file and then re-assign replicas to each file. By looking at logs created by each replica and view them only on one replica, we don't have to search log files on each replica and this greatly saved us time when debugging.