Dylan Huang - dphuang2
Alex Zhang - azhang38
ECE 428
Group 77

MP3 Report

Implementation

Our algorithm for leader election is similar to the bully algorithm but does not use election or victory messages and instead always assumes the first IP in its sorted membership list is the leader.

To maintain ordering, we use a master node that coordinates the location of replicas through consistent filename hashing and replicating on the key replica node's 3 successors. When a PUT operation is executed, the master node sends a signal to the 4 replica nodes to accept the incoming file. The rest of the nodes are signalled to decline the incoming file and drop the connection to notify the client not to send the file. GET operations are handled by simply first asking a node to send confirmation that the file exists on its sdfs. Once the first confirmation is sent, it then receives the file and rejects all other send requests from the nodes. To maintain consistency, we use quorum values of W=ALL and R=1.

Re-replication is accomplished by having every node periodically iterates over its file list and sends a request to re-replicate to everyone else in the replica group. If a node already has the replica, it simply notifies the requester and no file is transmitted. If a node is in the replica group for a file and receives a request for re-replication, then it accepts the file.

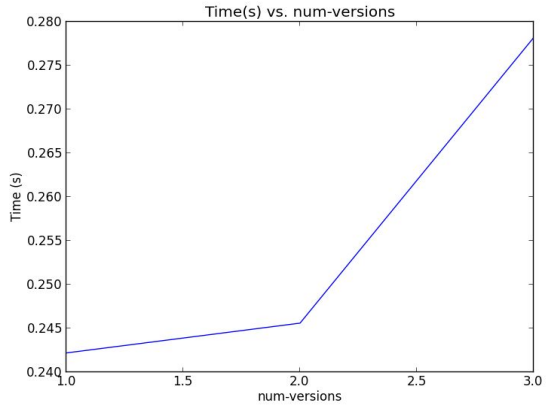
Since our simple leadership election and periodic re-replication process leads to over-replication in the case of inconsistent membership lists, we have another "clean up" thread on every node that periodically deletes replicas that are not supposed to exist on its file list. This way, we do not store too many replicas and replicas are always stored at their proper location.

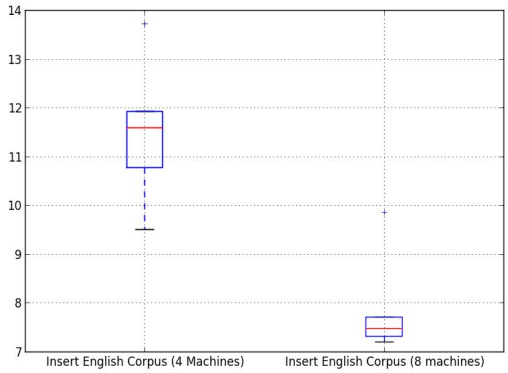
Usefulness of MP1

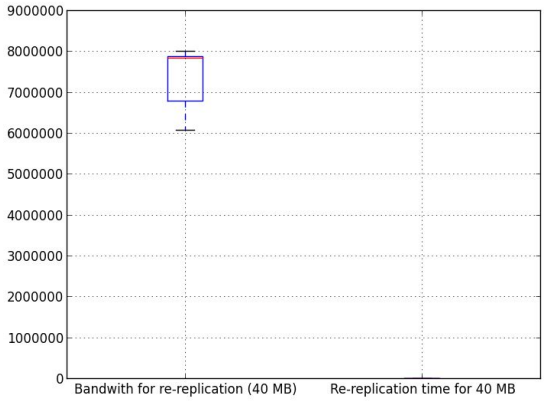
MP1 was essential to debugging our system as our logs could be easily grepped from a single system. It was also helpful by having an existing structure that we implemented on top of.

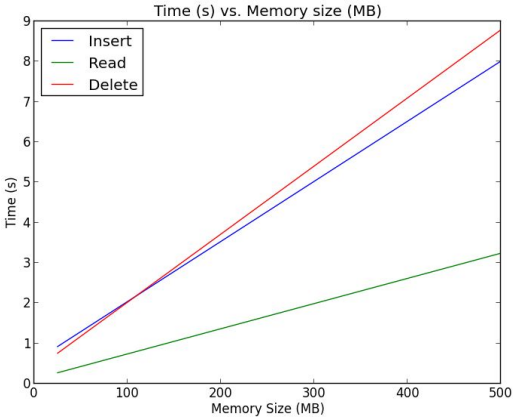
Dylan Huang - dphuang2
Alex Zhang - azhang38
ECE 428
Group 77

Data Analysis









From looking at the plots, some quick conclusions we can make is that the time does not increase a lot when num-versions is 2 compared to when num-versions is 3, separating the english corpus appears to be better when 8 machines which is probably due to more machines available for splitting the work. In terms of bandwidth usage, there appears to be a high bandwidth usage with the average on the higher end of the boxplot but the amount of time it takes to re-replicate files is consistent. Finally, we see that insert and delete appear to have very close running times as the memory size that the files occupy increases, but Read on the other hand, requires less time overall. This is consistent with what was discussed in class.