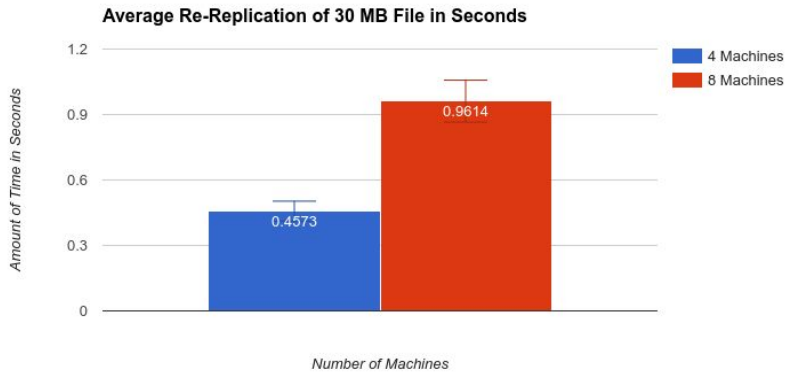
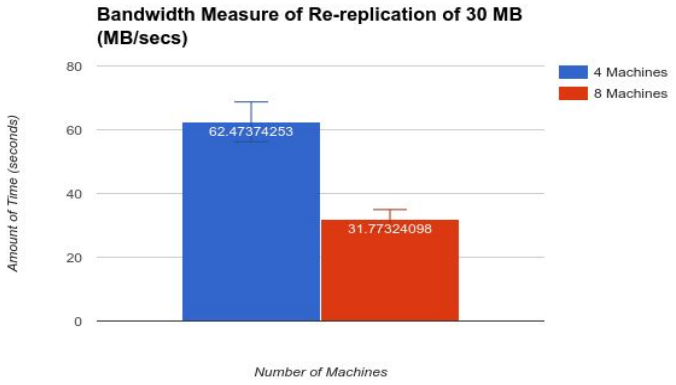
CS 425 MP3: Simple Distributed File System (SDFS)
Team Members: Imani Palmer, Ramya Nayanaswamy
Programming Language: python2.7

We implemented two different designs for our SDFS. The first design was masterless. All nodes have a basic knowledge of which nodes are alive and the files are distributed to all alive nodes. Without a master there is no single point of failure, however, the initial put function is very I/O intensive. In the second design, we rely on a master node that is elected using the Bully algorithm. Once a leader node is elected, the system allows for SDFS commands. Once we were able to determine a failure of a node, we replicated the necessary information to other nodes. We also use blocks in order to aid in active replication of files. Both of our designs rely on our MP2 failure detection scheme of SWIM. MP1 allowed us to debug our logs in to make sure we were both not only detecting failures but as well as when a leader was elected. The following graphs are based on our second master-based design.

Re-Replication Time and Bandwidth Upon a Failure with a 30 MB file

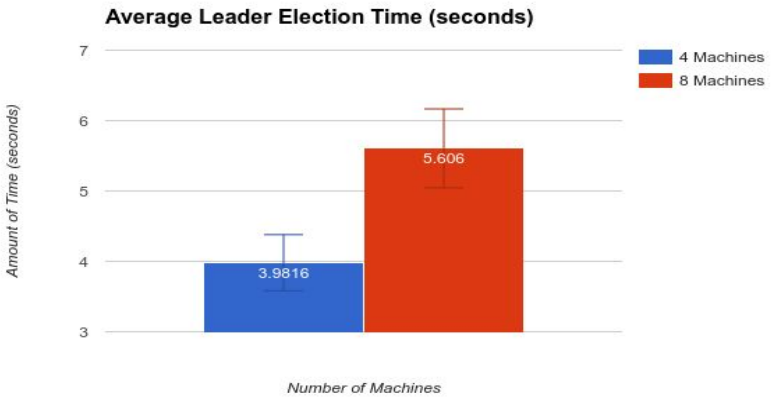
The average re-replication time of the 30 MB file across four machines is 0.4573 seconds. The average re-replication time of the 30 MB file across eight machines is 0.9814 seconds. The average re-replication time increased because the time it took to detect the failure increased. This is because the number of nodes has increased and the SWIM failure detection has to consider acknowledgements from more nodes. Once the failure is detected the replication is minimal.





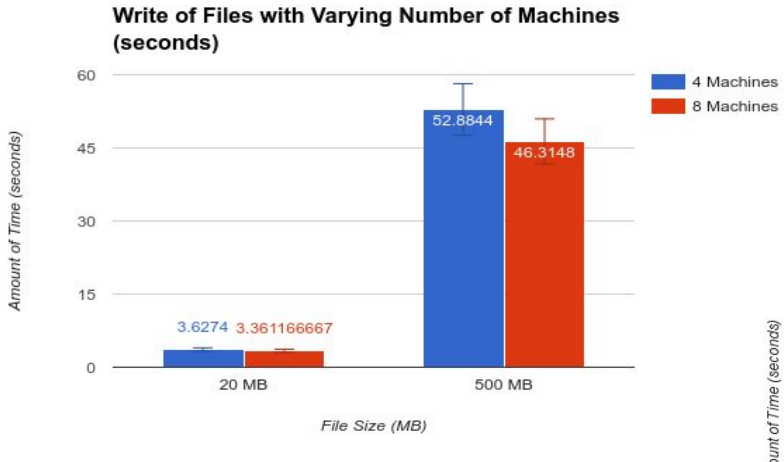
The bandwidth measurement is the size of the file (a.k.a 30 MB) divided by the number of seconds it took to perform the re-replication. The bandwidth for four machines is 62.47 MB/sec. The bandwidth for eight machines is 31.77 MB/sec. The bandwidth for eight machines is less than that of four machines because they are relying on the same network.

Time Between Master Failure and New Master being Reinstated

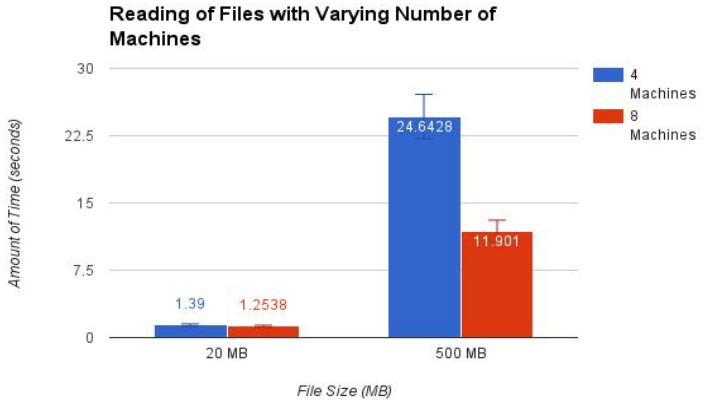


The average leader election time across four nodes is 3.98 seconds. The average leader election time across eight nodes is 5.608 seconds. The increase between the number nodes is because of the increase in time to detect a failure. As more nodes join the system it takes more time to detect a failure as well as more time to run the leader election protocol, as there is a need for more messages to be sent within the system to elect a leader.

Times to Read and Write Files of 20 MB and 500 MB



The average time to write 20 MB file with four machines is 3.83s. The average time to write 20 MB files with eight machines is 3.36s. The average time to write 500 MB file with four machines is 52.88s. The average time to write 500 MB file with eight machines is 46.31s. The average time to read 20 MB files with four machine is 1.39s. The average time to read 20 MB file with eight machines is 1.25s. The average time to read 500 MB file with four machines is 24.84s. The average time to read 500 MB file with eight machines 11.3s. The reading and writing of larger files requires more time than that of smaller files. This difference could be lessen with the usage of blocks. We did you blocks however, our block size is greater than 500 MB, making this useless in this example. The reading of files is also smaller across both file sizes. This is due to the fact that you only need to determine one location of a file and read that location. However, when you write a file it is necessary to write it to three different locations.



Store English Wikipedia corpus 1.3 GB

The average time to store the wikipedia corpus of size 1.3 GB is 68.80s across 4 machines. The average time to store the wikipedia corpus of size 1.3 GB is 62.74s. The storing of the english wikipedia corpus requires less time when going from 4 to 8 machines. One of the main causes of this is our block size. The corpus is broken down into two blocks, allowing for easier replication of the file across three different machines. This is enhanced when there are more nodes, in which to place the blocks. Blocks are more useful when there are a greater number of machines.