Design

In our system, there is one master node with a metadata list, which stores bookkeeping information regarding files on the SDFS and their host machines. We use All-to-All heartbeating protocol to ensure tolerance upto three node failures. Nodes periodically heartbeat their list of files stored in SDFS to the master in order to keep track of the filesystem metadata.

When a client initiates an action on SDFS, it will send a message to master most of the time, then wait for the master's response. Based on the response, the client will try to send or receive a file with sdfs_fiilename from the target nodes provided by the master. For each entry of the metadata list, there is a status field signaling if the file is currently being written, read, or idle. When a client initiates a PUT or GET for sdfs_filename that's currently being written, the master will tell the client to do the action later, since sdfs_filename is not idle.

Master also maintains a thread that checks the metadata list regularly, and sends REPLICATE messages to available nodes to ensure at least 4 replicas at any given time. Upon receiving REPLICATE, client directly connects to target to transfer file.

Basic filesystem access functionalities for the system have been implemented, with synchronization between threads achieved using mutually exclusive locks. This is to prevent concurrent writes and read-write conflicts while accessing the simplified filesystem, as well as the bookkeeping global structures (the membership list and the metadata dict).

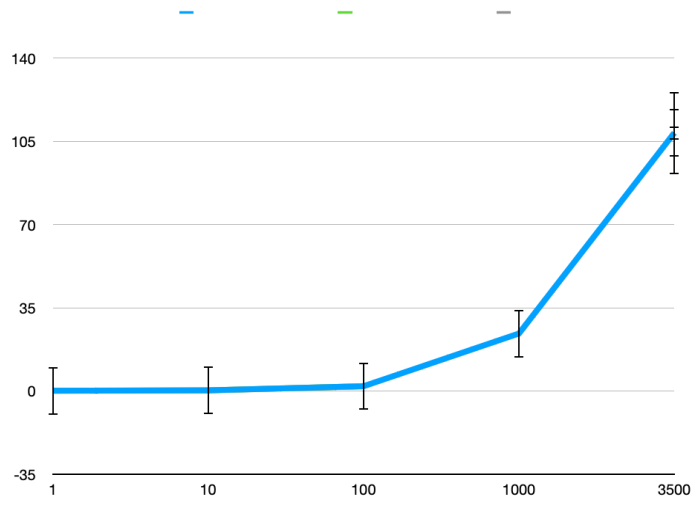

Fig 1 - Blue = 4 machines
PUT time for file size from 1 MB to 3500 MB (wikipedia corpus). We can see that the put time increases roughly linearly with file size, but standard deviation of put time increases drastically at filesize = 1000 and 3500 MB.
STDDEV = (0.002, 0.003, 0.0016, 2.664, 5.953)
The error bars indicate the standard deviation for both sets of measurements.

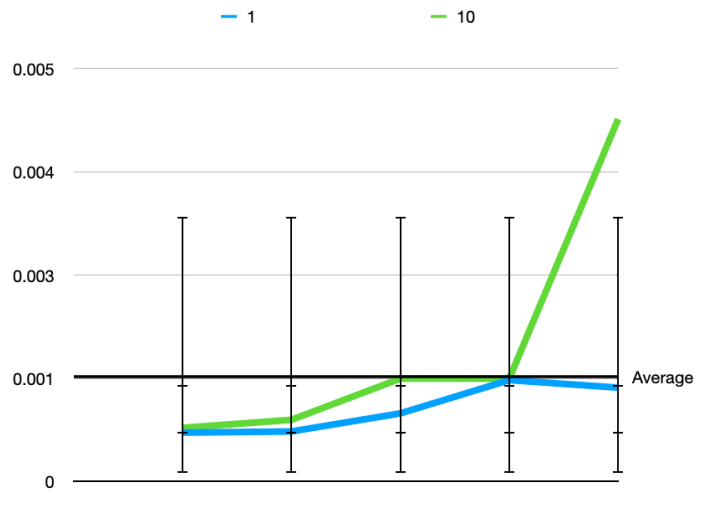

Fig 2 - Green = 8 machines, Blue = 4 machines
We can see that the timings for 4 machines and 8 machines are quite similar. We believe this is because we did not shard our files. We just send the whole wikipedia file at a time. A replica stores the whole file, not a chunk. If we were to shard our data, we might expect 8 machines to be faster. There's also a good amount of variance because it's too quick for consistent measurement.