Yi Zhu (yizhu10), Mingxi Sun (mingxis2), Yuzhong Deng (yuzhong5)

CS425 MP2 Report

System Design:

Our SDFS system mimics the architecture design of Hadoop Distributed File System ("HDFS"). On each VM, we would run two servers for *NameNode* and *DataNode*, together with a gossip-style *MembershipList* server for failure detection. To tolerate up to three machine failures at a time, our SDFS system would make four replicas for each file in the system.

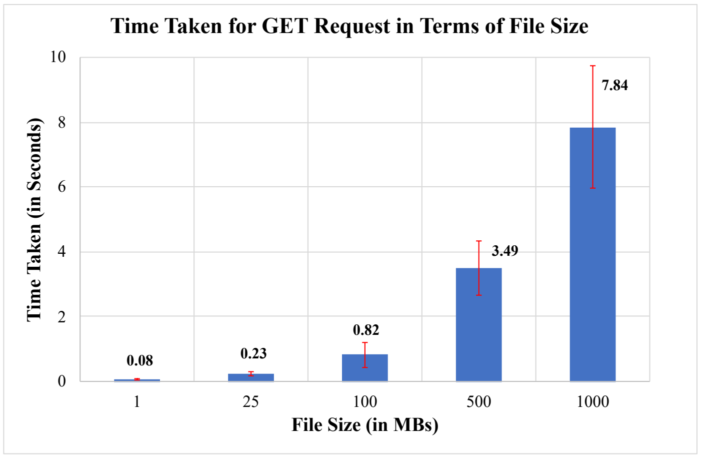
- **NameNode:** Although each VM hosts a *NameNode* server, there is only one primary NameNode existing in our SDFS system. This primary NameNode handles all incoming client requests (e.g. put, get) from different VMs. The singularity of primary NameNode ensures that at most one machine could write to SDFS at any time, while also avoiding any conflict operations such as write-write or write-read.
- **DataNode:** Concurrently with a *NameNode* server, each VM also runs a *DataNode* server that handles instructions coming from the Primary NameNode. The possible instructions include (1) saving a local file to the SDFS file directory, (2) deleting a file from the SDFS file directory, and (3) copying a file from another VM's SDFS file directory.
- **MembershipList:** The membership list is used to detect newly joined nodes or failed nodes in the group. The Primary NameNode used information in the membership list to assign replicas for a new SDFS file. When any node failures are detected, the Primary NameNode would now also be alerted and promptly re-replicate the SDFS files whose replicas are lost due to the node failure.

Whenever a GET client request is received on a VM, it would first pass this GET request to the *NameNode* server, which would direct the request to the Primary NameNode via a UDP message. Once the Primary NameNode received this request, it would check whether this request would cause any violation. For example, the Primary NameNode cannot process the GET request if the previous write is still in progress or there is currently fewer than 4 VMs in the group. If no violation occurs, the Primary NameNode would instruct the *DataNode* server on the VM that received the GET request to copy the requested file from a chosen VM that has a replica of that file. The PUT and DELETE client requests would be handled in a similar fashion.

Since the Primary NameNode is a single point of failure in our SDFS VM cluster, we include three additional backup NameNodes. All client requests, as well as the acknowledgements from the *DataNode* servers, would also be sent to the three backup NameNodes as well, such that every backup NameNode would have up-to-date information for all SDFS files. When the Primary NameNode has failed, one of the backup NameNodes will be elected as the new Primary NameNode.

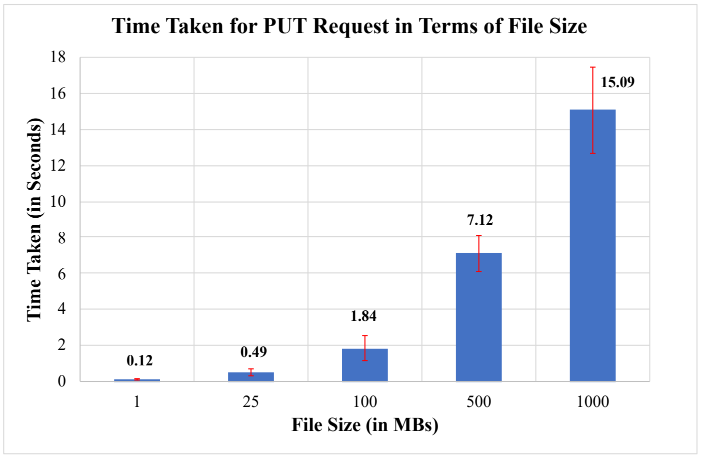
Yi Zhu (yizhu10), Mingxi Sun (mingxis2), Yuzhong Deng (yuzhong5)

Get Speed In terms of File Size (on 6 VMs):



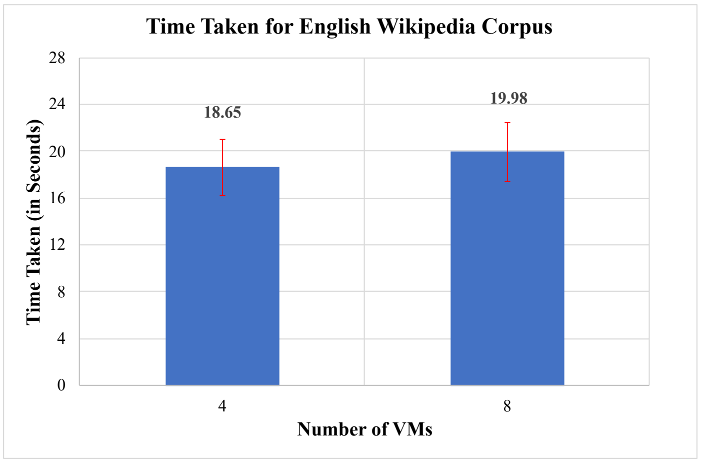
We observed that the read time is roughly linear relative to the file size. Also notice that the read time has a larger standard error than the put speed. We think this is because when the client calls the PUT request on the VM that happens to have a replica of the requested file, the read time is significant lesser compared to reading a file from a different VM.

Put Speed In terms of File Size (on 6 VMs):



Same as read time, we observed that the write time is roughly linear relative to the file size. However, the time to write a file in SDFS is twice as long as the time taken to read a file. Such time difference, we reasoned, is due to the fact our system needs to make four replicas of each incoming file.

Put Speed for English Wikipedia Corpus:



Notice here that the time taken to read the entire English Wikipedia corpus is roughly the same when we used 4 nodes compared to 8 nodes. This is not surprising, given that each SDFS file would only be replicated four times, and therefore the write speed would not be impacted by the number of nodes in the system.