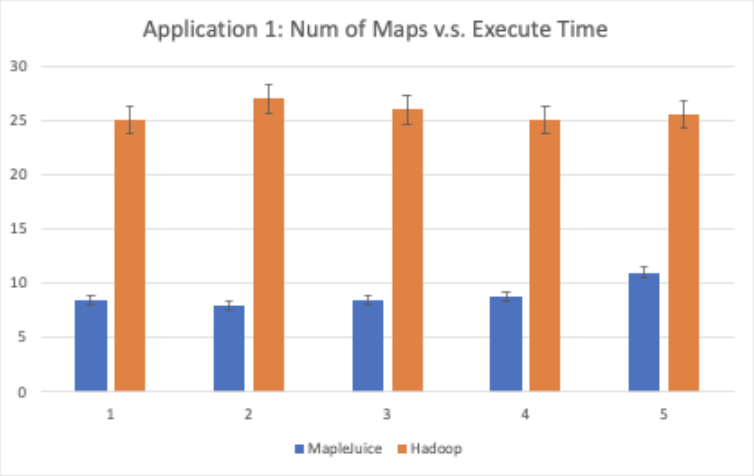
CS425 MP3 Report
Yifan Cao (yifanc7) and Enyi Jiang (enyij2)

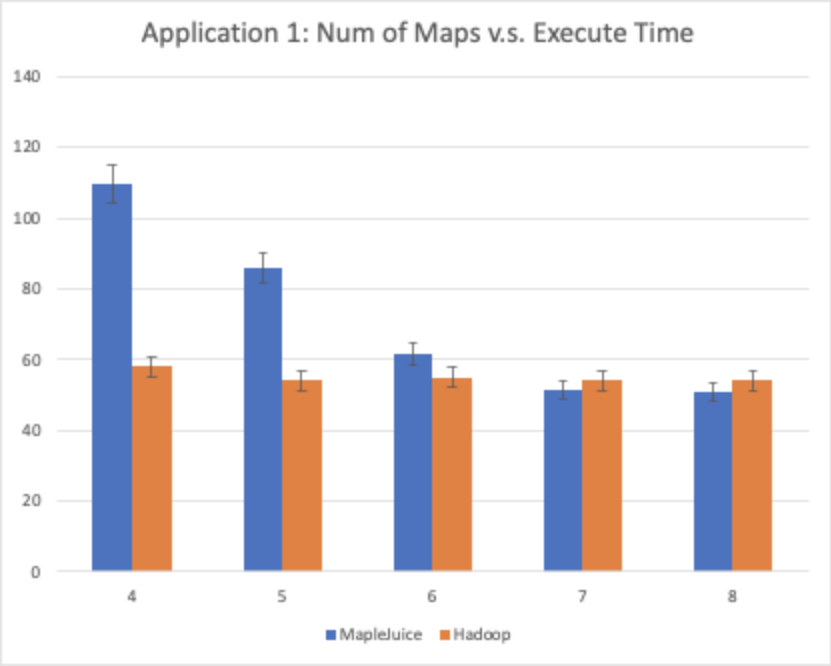
Design: Master is chosen as the first VM. MP3 uses scp cmd to send and receive files. MP3 consists of MasterReceiver class, Receiver class, Sender class, MapleJuice class, and MapleJuiceFactory class, and FailureDetector class.
Receiver: responsible for receiving files assigned by the master and uses maple juice to process the files received. Every common server will have a receiver.
MasterReceiver: a subclass of the receiver. Besides completing tasks of receivers, the master receiver is also responsible for receiving maple juice commands entered by the user, assigning blocks of datasets to different servers, and collecting the result of maple juice from other servers.
Sender: responsible for sending maple juice requests to the master receiver. Every server has a sender. When the user enters a maple/juice command in a server, its sender will send a corresponding request to the master receiver. The master receiver will then begin the maple/juice stage.
MapleJuice: an abstract class with two abstract methods: ***maple*** and ***juice***. All maple juice applications inherit from the maple juice class.
MapleJuiceFactory: a class responsible for creating maple juice applications based on the input key passed into the function.
FailureDetector: responsible for detecting failures and handling false positives. When failure occurs, the master will reassign the files originally assigned to the failed server to a living server, and remove the failed server from its membership list. When a false positive occurs, the master will add the false-positive server to its membership so that it can assign files to this server in the future.

Plots:
Application 1 Map of New York City to find the buildings met with standards



Application 1 is an easy, small task with ~80M data and O(1) per line. We can see when we use a different number of maps/maples, our MapleJuice all outperforms Hadoop. I think Hadoop as a mature distributed system, it will have a more complicated procedure to start up / end up. Thus, when our application takes very little time to train, Hadoop will still have relatively large overhead costs, which results in the differences in this application.

Application 2 Condorcet Voting with three candidates



For application 2, it is more complicated than application 1 with a bigger size of data and O(k^2) per line (k = # of candidates). In this case, Hadoop outperforms our MapleJuice for most of the time. Hadoop uses HDFS, which is more advanced than our implementation of SDFS. Thus, it will have a smaller delay for the file operations. In addition, our system just has parallel processing for a number of maples. However, Hadoop can achieve more parallelism during different stages of MapReduce. Thus, this is the reason why it will have a smaller time delay when the number of maples/maps is small. We can observe that when the number becomes 7 or 8, the differences between these two performances are not so significant because our MapleJuice also achieves reasonably good parallelism then.