Ayushi Singh (ayushis3), Rishu Bagga (rbagga2), Sam Pal (sspal2)
Group 17

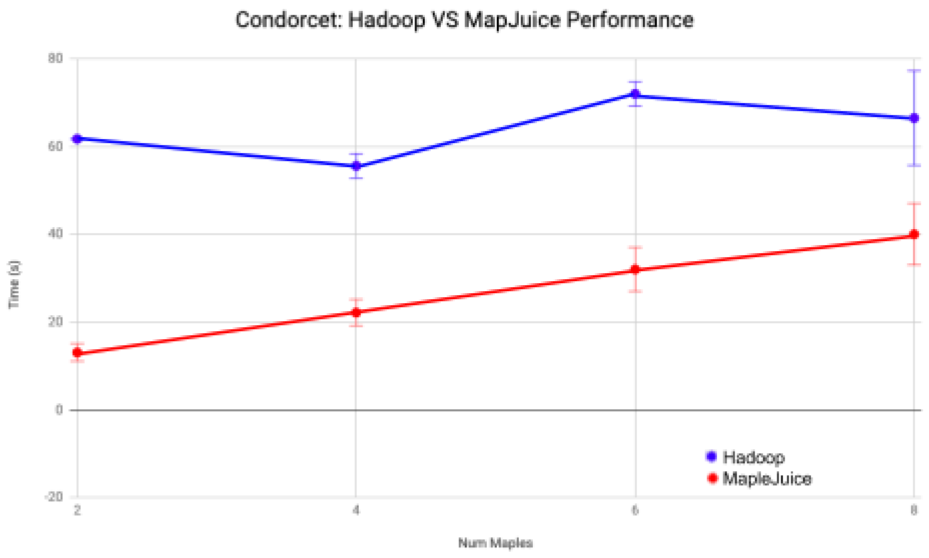
CS 425 MP 3

Design
For this MP, our master/introducer takes in an input file as well as the maple and juice executables. Then, the introducer splits up the input file by using the modulus operator to uniformly distribute the input across all worker machines somewhat evenly. Each worker machine then runs maple on its given input files, saves the output as a file in the format "sdfs_intermediate_filename_prefix", and PUTs the output onto the SDFS filesystem. Then, the introducer GETs all the maple outputs from every machine and creates new files that aggregate the outputs by key. We ensure that all machine outputs are accounted for through the use of a global counter. Once that has been modified, those new files are PUT onto the SDFS filesystem and the introducer partitions those files to the worker machines once again, but now for the juice phase. Each worker machine GETs a certain key file, and this assignment is determined by shuffling, which involves utilizing both hash and range partitioning. We computed a hash partitioning using the modulus operator on a hashed key and did range partitioning by sorting the input alphabetically and slicing it somewhat evenly by the number of partitions we defined in the input. Finally, each worker machine runs the juice task and appends the result to one unified "sdfs_dest_filename" file. In the case where a machine fails, we will have the first machine to detect its failure finish the process on its behalf.

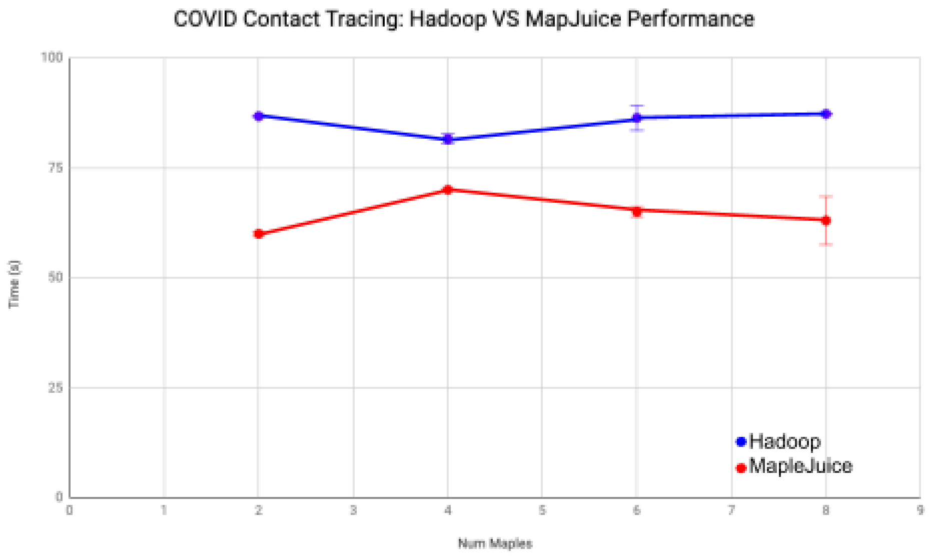
Applications
Our two applications are Condorcet Winner and Contact Tracing. We implemented these in MapleJuice and Hadoop using the homework instructions. Condorcet Winner uses two chained Map-Reduce jobs, and Contact Tracing uses four chained Map-Reduce Jobs. Applications written in both MapleJuice and Hadoop have the same functionality to facilitate accurate experimentation as seen below.

Expectations
As num maples increased, we expected to see a decrease in the amount of time it takes to execute because the work is more distributed (input is partitioned across a larger number of maple tasks), so each machine handles a lower volume of input. When comparing Hadoop and MapleJuice, we expected to see MapleJuice perform better because we had a couple of optimizations in place - we utilized sets in order to facilitate O(1) access for data, we process batches of lines of input instead of one line individually (which is how Map technically works), and we utilize SDFS storage to our advantage by avoiding replication of unnecessary GETs and PUTs made, so we don't GET a file we already have.

Application 1 (Condorcet) Plot



Application 2 (COVID Contact Tracing) Plot



Actuality

In actuality, there was an increase in the amount of time because time as the number of maples increased because there was time taken in the distribution of the input data, in the aggregation of outputs after the maple and reduce phase, as well as variable network latencies during the time of execution. The performance of MapleJuice was more efficient than Hadoop because of the optimizations we mentioned earlier.