CS425 MP4 Report
Yuechuan Zhang (yz134)
Xinshuo Lei (xinshuo3)

Design
Our MapleJuice framework is built on top of SDFS and contains three major components:

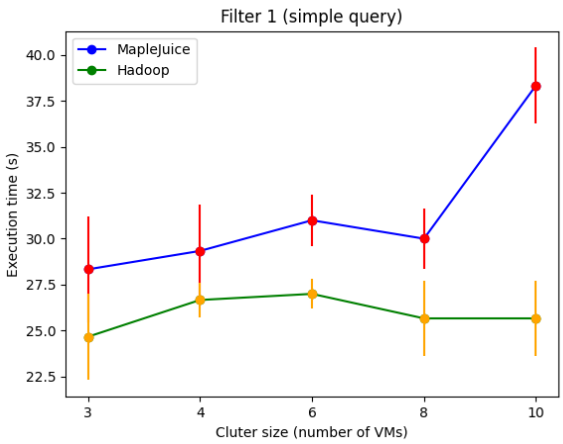
MapleJuice Job Manager: This service runs on the leader node, accepts Maple/Juice job submissions from clients, and overwatches the job execution. For a Maple job, the job manager evenly partitions the input file fetched from SDFS and correspondingly assigns Maple tasks to different MapleJuice worker nodes. For a Juice job, the manager searches the key set related to the input prefix from SDFS and distributes keys among Juice tasks. In both cases, the job manager is responsible for starting Maple/Juice tasks on worker nodes and rescheduling for failed ones.

MapleJuice Node Manager: This is the worker node daemon that executes the Maple/Juice tasks based on instructions from MapleJuice Job Manager. For each task, the worker thread fetches executables and input files from SDFS as instructed. It then runs the executable against the input file and writes back to SDFS.

SQL Layer: This is a client application that parses user queries from command line and submits jobs to the Maple Juice Job Manager correspondingly. Upon invocation, it dynamically generates executables from local templates based on query arguments and uploads them, together with the input file to SDFS. The SQL Layer client then calls the MapleJuice API and eventually fetches the query output from SDFS to the local folder.

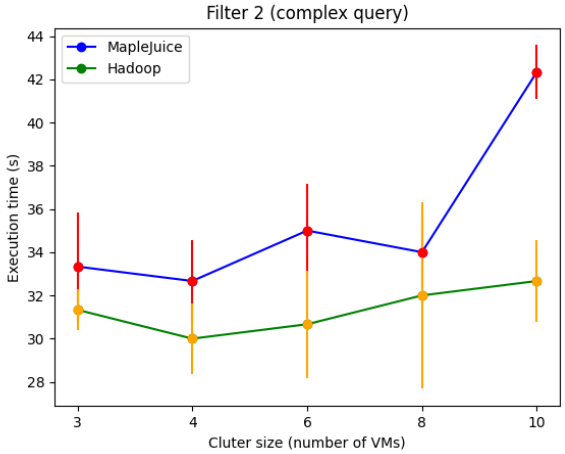
Comparison with Hadoop
1. Filter queries
In order for the leader election algorithm to run smoothly, the minimum cluster size we chose is 3 VMs. The filter query is performed on a csv data file that is approximately 100 MB.

For the simple query, we input a regex that looks for the sequence “Al” in each line. For Hadoop, varying cluster size doesn’t seem to have a large impact on performance. We believe this might be due to the fact that Hadoop dynamically decides the number of map tasks and number of



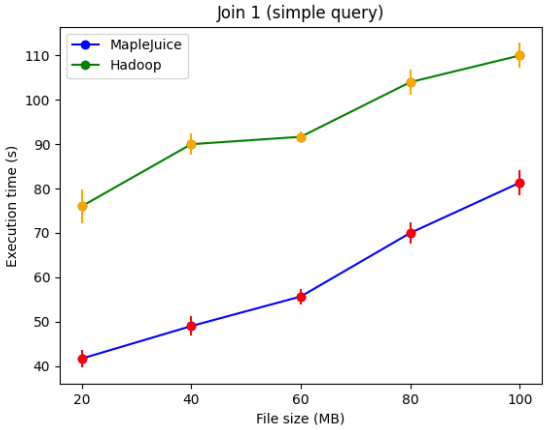
maple tasks based on input file size, and therefore, having more VMs in the cluster didn't affect the number of tasks that were executed. For our MapleJuice engine, the execution time is pretty stable for 3-8 VMs. When there are 10 VMs in the cluster, we observe a large increase in execution time. This is most likely due to the fact that we take the number of VMs in the cluster into consideration when statically setting the number of Maple and Juice tasks invoked for a query. When there are 10 VMs in the cluster, more tasks are executed. However, since the input file is relatively small, having more worker threads executing in parallel fails to improve performance, but rather adds extra overhead due to increased network communication including RPC and file transmissions.

For the filter queries, our MapleJuice engine is outperformed by Hadoop by 2 - 3 seconds. This marginal difference, as explained above, might be due to our static configuration of the MapleJuice task number required for each query.



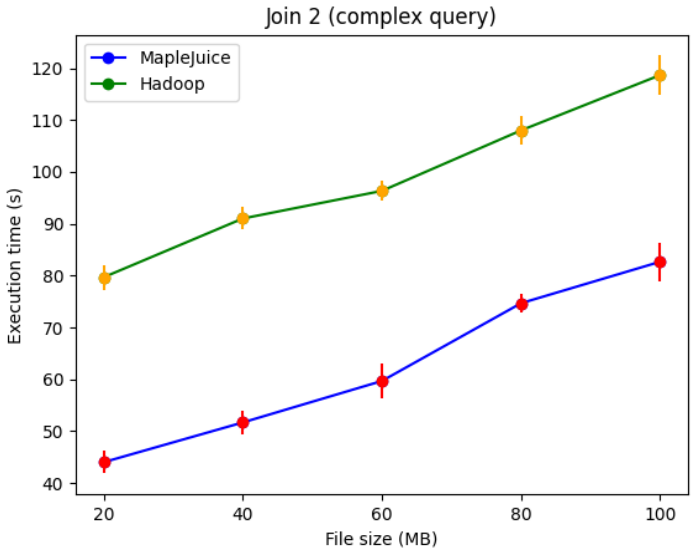
For the complex query, we match the regular expression on one column instead of the whole line. We also used a more complex regular expression ("Al | B+"). The performance trend is approximately the same as the simple query, with 4 - 6s increase in execution time.

2. Join queries



For the simple join query, we joined two datasets (customers and products) on one field using an exact match. As file size increases, the execution time of both Hadoop and our MapleJuice engine increases. This matches our expectation since larger input files usually lead to more time spent on partitioning input, transferring files, and running the executables. The execution time of a join query on two datasets that are each 100MB is around twice the execution time of a filter query. This is expected since for join, our implementation is to perform two Maples (one on each dataset), then one Juice.

For the join queries, our MapleJuice engine is able to outperform Hadoop. This is mainly due to the fact that Hadoop streaming is not very flexible with chaining multiple Map tasks with one Reduce task. Therefore, we have to perform 3 separate MapleReduce phases (1 for each dataset, then 1 for the combined result).



For the complex join query, we joined two datasets (customers and product) on more than one field, one of which is numerical. The performance trend is basically the same as the simple query, with a slight increase in execution time.