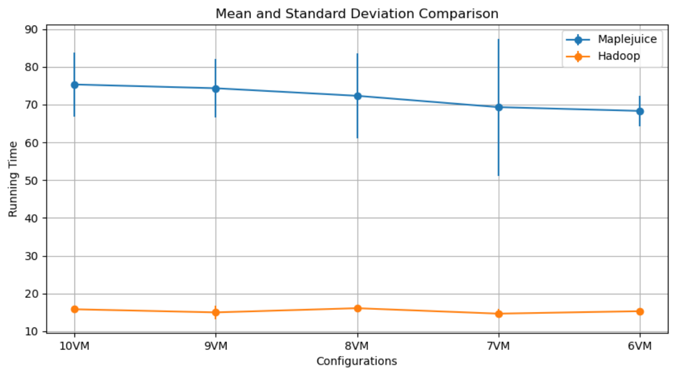
MP-4 Report by Colin Zhou (colinz2) and Keyang Xuan (keyangx3)

Design: For this MP, we used an architecture where the user would be able to type in a maple or juice command in the command line. The request would be sent to the leader, who would queue the request, and then alert the querier machine to send the input files over to the leader. The leader is responsible for partitioning the files for the maple stage (we used hash partitioning), and also doing hash or range partitioning for the juice stage depending on the user input. The worker nodes are responsible for completing the maple and juice stages (generating intermediate files with the prefix specified in the command) and sending the results back to the leader. After the leader receives the results of the juice stage, it will combine all the results from each worker into one file (filename specified in juice command) and then send that back to the querier machine. We are using a progress tracker on the leader (a python dictionary) to keep track of the progress of each worker machine, and detect when a failure of a worker necessitates a reallocation of tasks. For the maple stage, each worker is emitting key value pairs where the key is the first value in the column of the structured data, and the value is everything in the entire row. The key value pairs are written into files as tuples encoded as strings (these strings will then be decoded in the juice stage and the values extracted). We used a failure detector and SDFS from MP3 to detect node failures and to store files.

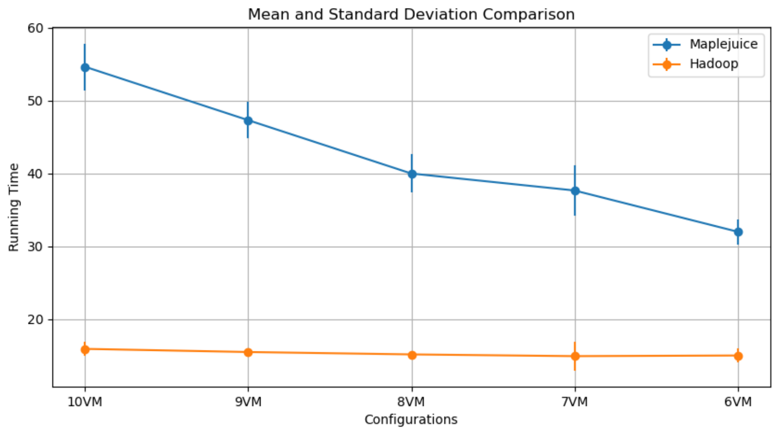
Experiments: *All datasets used in the experiment were synthetically generated.

Simple regular expression (Filter 1): We used the employees file (schema is employee_id, employee_name, dept_number) with about 4 million rows (100mb roughly). We tried to find all the rows with “Meghan” in the row.

We found that Hadoop was very consistent, maintaining performance despite the cluster numbers changing. This might be due to how Hadoop’s internal mechanisms handle the number of tasks to each worker machine. Our MapleJuice was relatively stable, although the times did go down, probably due to the leader not having to open connections to so many machines, thus saving on latency and file writing time.

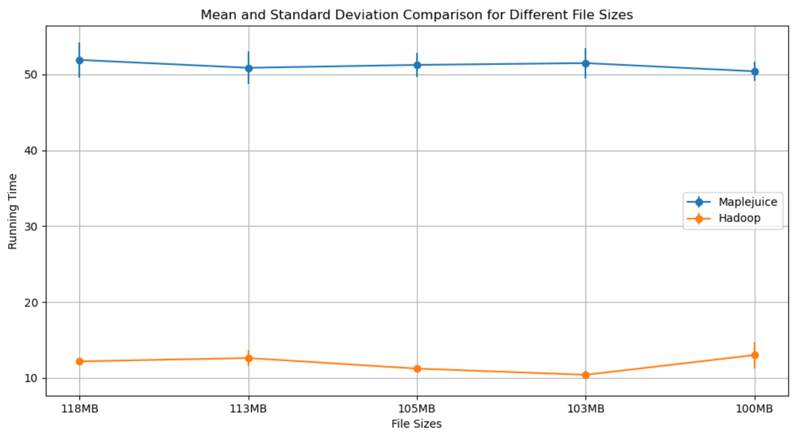


Complex regular expression (Filter 2): We used the same employees file again and tried to find all the matches with regex “es, 12” (matching numbers, letters, spaces). Again, our MapleJuice framework had better run time as the number of machines went down while Hadoop remained stable. It could be because as the number of VM’s goes down, our MapleJuice framework doesn’t have to open up extra connections or send files back and forth to extra machines, thus lowering latency and overall job completion time.



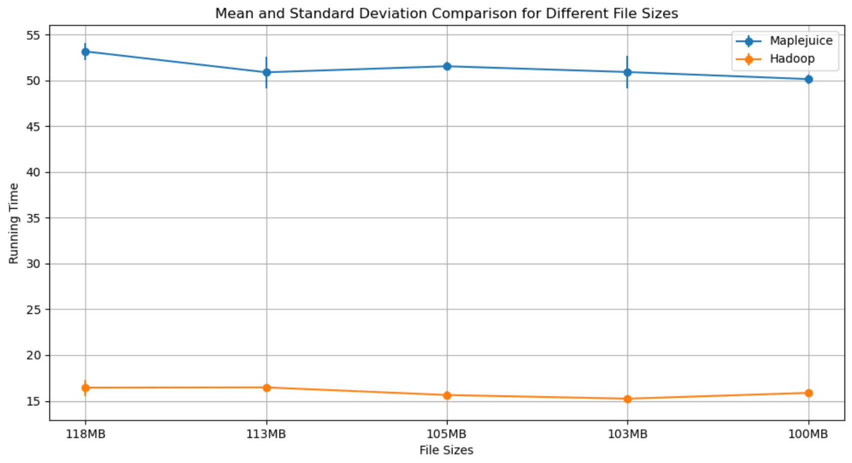
Simple Join (Join 1): For the join, we generated a departments file with schema (dept_id, dept_name, time_zone) where every single dept_id was unique. The simple join would find all the rows in the employee table with dept_id = 48 and join it with the single row in the departments_file with dept_id = 48.

We found that both Hadoop and our MapleJuice maintained a relatively consistent performance via different file sizes. This is expected since the five files we tried don’t vary too much in size (even though the files are over 100mb).



Complex Join (Join 2): For the complex join, we had an employees table, departments table, and time zones table with schema (time_zone, phone_number) where every time_zone was unique. We find all the rows where the employee dept id is = 48 and the time_zone was equal to “Asia/China”.

Similarly, the Hadoop and our MapleJuice maintained a relatively consistent performance via different file sizes. The overall theme here is that Hadoop is extremely consistent when the file sizes are varied. The biggest bottleneck for our implementation might be due to copying and sending the files between worker machines. Also, the partitioning by the leader could also take some time as only the leader does the partitioning.


Mean and Standard Deviation Comparison for Different File Sizes
Running Time
File Sizes
118MB
113MB
105MB
103MB
100MB
Maplejuice
Hadoop