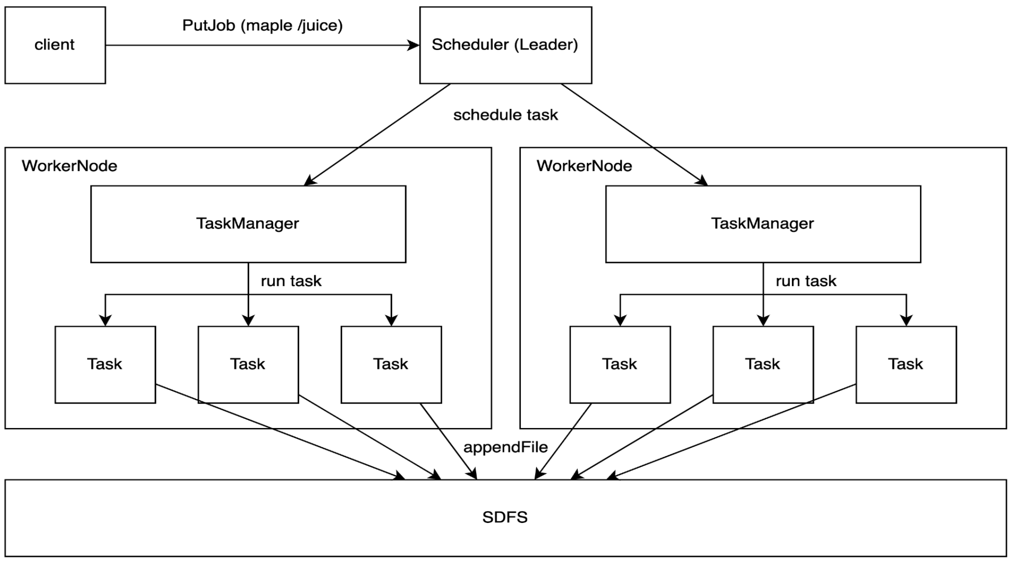
Machine Programming 4 – MapleJuice+SQL

Che-Kuang Chu(ckchu2), Jhih-Wei Lin(jhihwei2)

Design

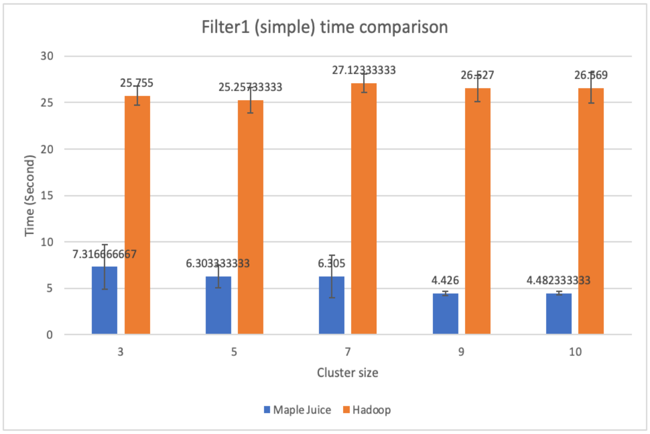


We implement two components for maple/juice: Scheduler, TaskManager

- Scheduler
  - receive Maple/Juice Job from client
  - decide how many tasks should be created based on num_maples and num_juices
  - assign input files to tasks
  - schedule tasks based on hash or range partition
  - detect task failure and re-schedule (fault-tolerant)
- Task Manager
  - receive tasks from the Scheduler
  - run the task on the local machine
  - output result to SDFS

Filter1

Dataset: 33MB / Regex: "zz" / Output Size: 4.1 MB



We observe that our MapleJuice runs faster as the cluster size increases. This outcome is anticipated because, in the test, we divided the job into smaller tasks and distributed them across each node. With an increase in the number of nodes, the load on each node is reduced, leading to faster task completion.

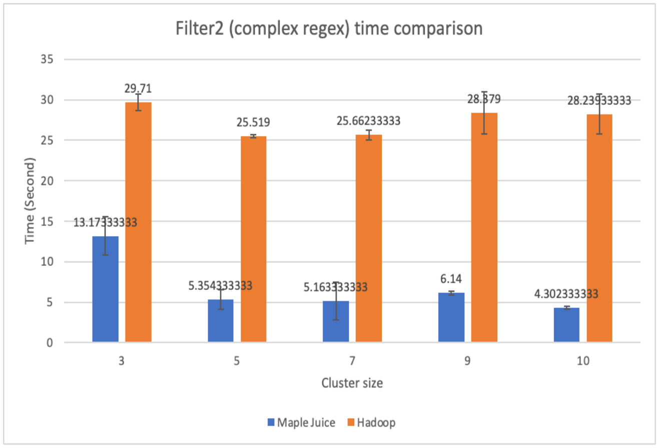
Machine Programming 4 – MapleJuice+SQL

Che-Kuang Chu(ckchu2), Jhih-Wei Lin(jhihwei2)

The performance of Hadoop MapReduce appears consistent across various cluster sizes, which is unexpected, as one would anticipate improved speed with larger clusters. Our hypothesis is that this outcome is attributed to the primary performance bottleneck in MapReduce not being directly correlated to cluster size, but rather to the longer spawning time of each task.

Filter2

Dataset: 33MB (same as Filter1) / Regex: "^H.*zz" / Output Size: 1.3 MB



We observe improved speed as the cluster size increases, which aligns with the same reasoning as observed in Filter1. Additionally, our runtime is faster than that of Filter1. This aligns with expectations since a more complex regex in our mapper function results in fewer intermediate files, leading to a reduction in the number of files processed by the reducer and, consequently, a decrease in overall running time.

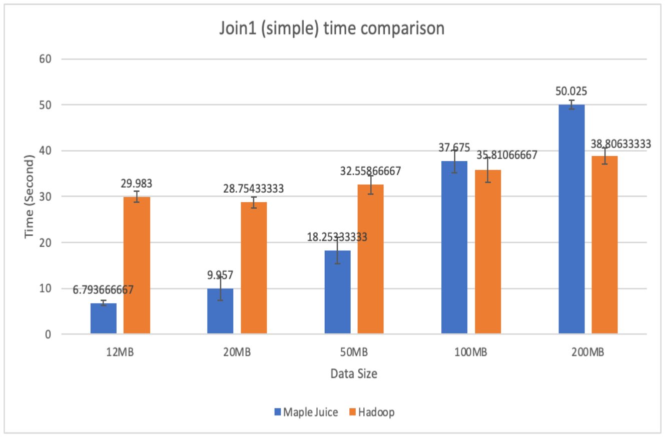
Moreover, our standard deviation is notably high, primarily attributed to our retry mechanism. In case of unexpected errors (e.g., timeouts), we introduce a one-second delay and reattempt the actions. Consequently, certain trials may experience prolonged durations due to these retries.

In contrast, Hadoop's performance appears to resemble that of Filter1. We attribute this similarity to the bottleneck in performance lying with the longer spawning time of tasks. As we have similar sizes of tasks between the two cases, there is no significant deviation between the performance of Filter2 and Filter1 in the Hadoop context.

Join1

Number of VMs: 10 / Output Size: 8 MB

Query: SELECT ALL FROM d0.csv, d1.csv WHERE d1.col1 = d2.col1



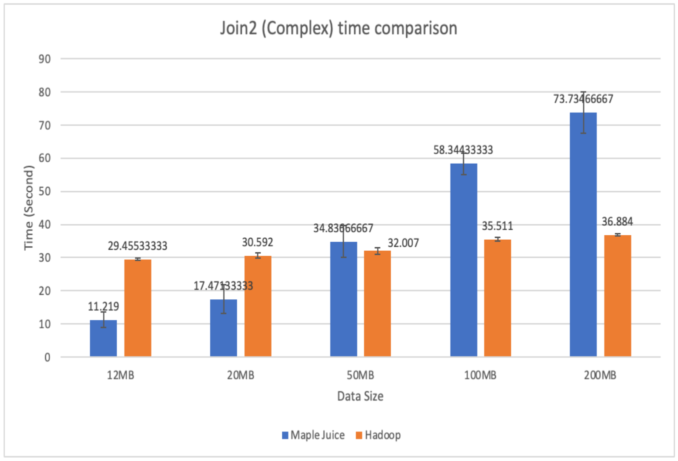
We noted an increase in running time as the file size expanded, which was expected. This prolonged duration can be attributed to the additional time required for three main factors: (1) task splitting based on file lines,

(2) the execution of each task, and (3) the get/put/append operations with SDFS.

In contrast to our MapleJuice implementation, Hadoop's runtime is less significantly affected by changes in file size. This suggests that Hadoop exhibits superior performance in processing individual tasks. We hypothesize the majority of Hadoop's runtime is dedicated to scheduling tasks and managing resources. As a result, there is minimal variance in performance time across different file sizes.

Join2

Number of VMs: 10 / Output Size: 8 MB
Query: SELECT ALL FROM d0.csv, d1.csv WHERE d1.col1 = d2.col1



Why more complex:

We have introduced additional types of common elements in both datasets, leading to the spawning of more Reduce tasks. In the case of Join1, where only one type of common element exists, it results in the generation of a single intermediate file.

We observed a rise in running time as the file size expanded, as anticipated for the same reasons outlined in Join1. Furthermore, the running time exceeded that of Join1. This outcome was anticipated as we introduced complexity to the dataset, leading to an increased number of intermediate files between the mapping and reducing phases. Consequently, more operations on SDFS are required, extending the overall runtime.

We observe that the processing times in Hadoop are similar to those in Join1. As previously mentioned, our hypothesis is that Hadoop spends a significant amount of time on task scheduling and spawning instead of file processing. Therefore, the observed result aligns with our expectations.