

## MP4 Report

chinhao2 | Chin-Hao Lo / hcma2 | Ho-Chih Ma

### Overview

In MP4, we designed a distributed file storage system that supports MapleJuice functionality. It is a one-leader to multiple-workers system in which the leader takes on the responsibility to partition files, create replicas as well as keep track of completion of Maple/Juice tasks. New commands added in this MP include Maple, Juice, Filter, and Join. When a server receives one of these commands, it will first notify the leader. Once the command is ready to execute, the leader will first obtain the file locally and hash/partition the file to improve load balancing. Then, the leader concurrently communicates to multiple work machines to execute the given command simultaneously.

For Maple and Juice, the leader will send the partitioned data, execution file, and complimentary arguments to the corresponding worker. Workers can work concurrently due to our multi-thread design on the leader side. The worker sends the result back to the leader once it's done with its job. Workers also send complete flag messages to the leader, triggering thread closing on the leader side. Leader will collect all the threads and examine which ones failed. If there is any failed thread, which could be the result of bad connection, worker failure, etc, the leader will select another worker to re-execute the failed task. This ensures machine failure tolerance.

For Filter, we implement a corresponding set of handle functions to process and execute filter commands. Users input: "filter SELECT ALL..." which invokes the filter functionality. Filter function includes a maple phase and a juice phase. In the maple phase, the leader partitions the designated dataset into several equivalent sizes of data, sending each of them to available workers. The execution file performs a regular expression check and output (1, data). For future improvement, we should output (x, data) where x is randomly selected from 1 to number of worker machines for better performance in the juice phase. When all maple tasks are complete, the leader creates a file that contains all the collected data and assigns one work to execute the juice phase. The leader will send the file it just created and an identity reduce execution file to a worker. The worker does the identity reduce and sends back the result.

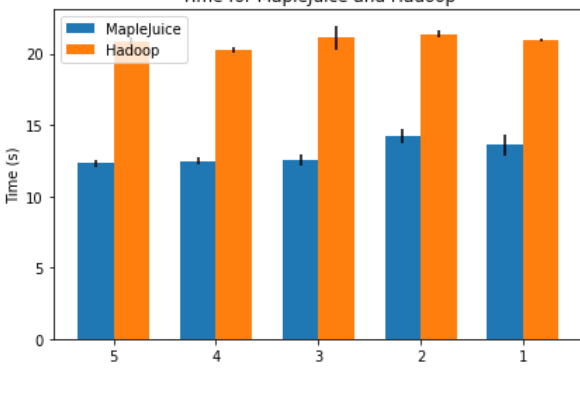
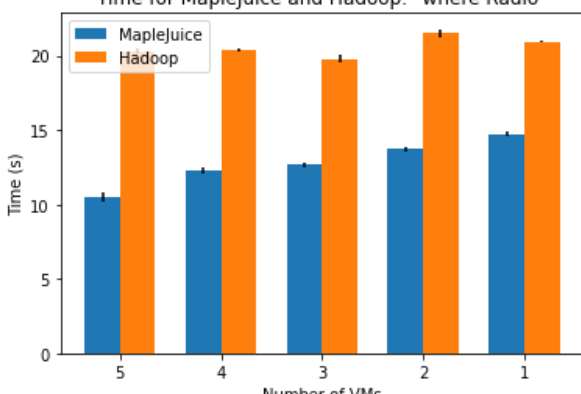
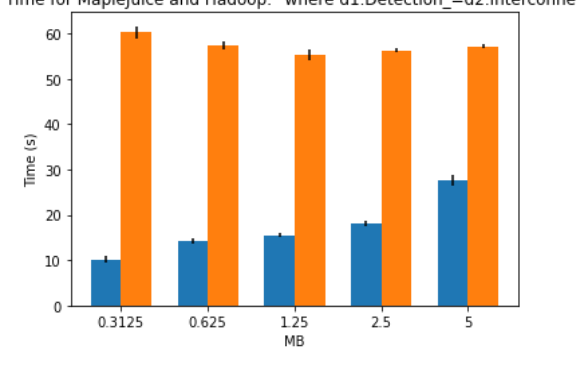
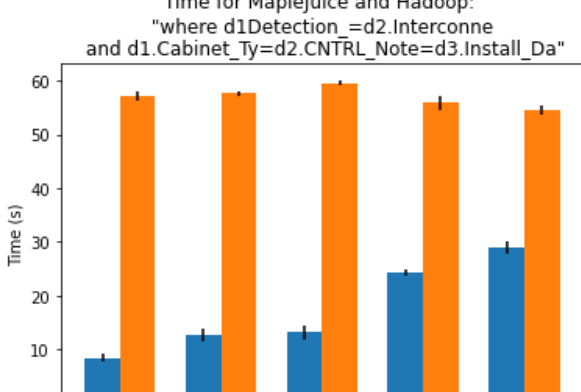
For Join, we first do a maple and an identity juice for both datasets first. The maple takes in the schema the dataset is associated with as input, and outputs (value of the schema:d1/d2, data). Then, we do an identity map and a juice that outputs all data if a file (grouped by key) has both d1 and d2 in it.

Overall, this MP concludes all the MPs we've done in CS425. We are extremely satisfied with the system we've built along and the knowledge we've obtained this semester.

### Experiment Data

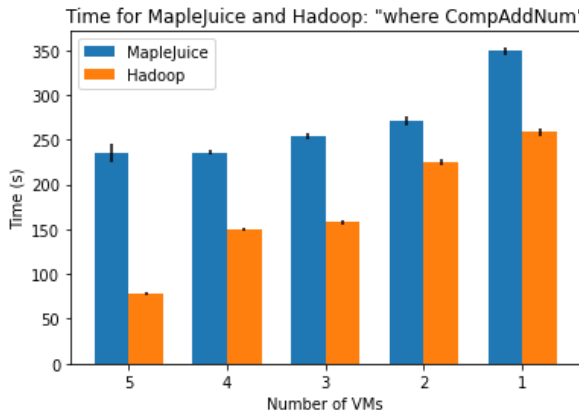
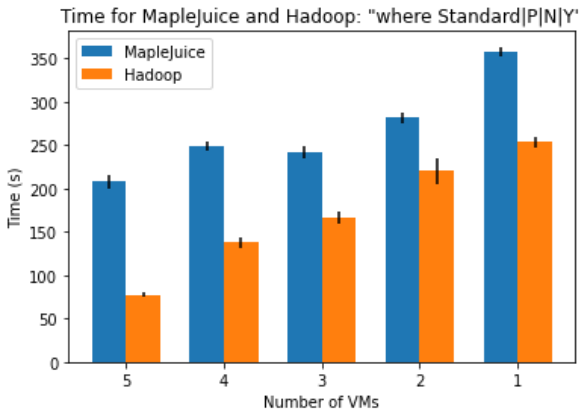
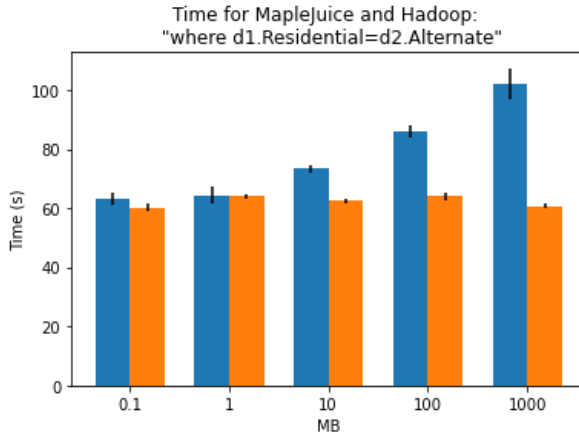
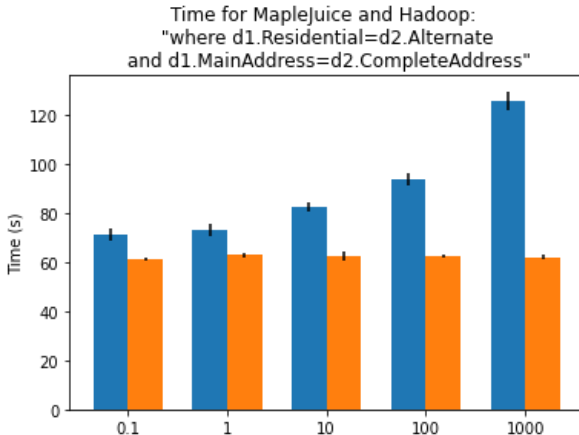
We experimented with a total of eight different scenarios and compared the performance of our MapleJuice verses Hadoop. We tested on both filter and join, varying numbers of VMs, sizes of input files and regular expressions to obtain more robust results.

We use the TrafficSignalIntersections dataset to experiment and plot the following four graphs. The dataset has 121 rows (entries).

 <table><caption>Time for MapleJuice and Hadoop</caption><tr><th>Number of VMs</th><th>MapleJuice (s)</th><th>Hadoop (s)</th></tr><tr><td>5</td><td>~12.5</td><td>~19.5</td></tr><tr><td>4</td><td>~12.5</td><td>~20.5</td></tr><tr><td>3</td><td>~12.5</td><td>~21.5</td></tr><tr><td>2</td><td>~14.5</td><td>~21.5</td></tr><tr><td>1</td><td>~14.0</td><td>~21.0</td></tr></table>	Number of VMs	MapleJuice (s)	Hadoop (s)	5	~12.5	~19.5	4	~12.5	~20.5	3	~12.5	~21.5	2	~14.5	~21.5	1	~14.0	~21.0	 <table><caption>Time for MapleJuice and Hadoop: "where Radio"</caption><tr><th>Number of VMs</th><th>MapleJuice (s)</th><th>Hadoop (s)</th></tr><tr><td>5</td><td>~10.5</td><td>~19.5</td></tr><tr><td>4</td><td>~12.5</td><td>~20.5</td></tr><tr><td>3</td><td>~12.5</td><td>~19.5</td></tr><tr><td>2</td><td>~14.0</td><td>~22.0</td></tr><tr><td>1</td><td>~15.0</td><td>~21.0</td></tr></table>	Number of VMs	MapleJuice (s)	Hadoop (s)	5	~10.5	~19.5	4	~12.5	~20.5	3	~12.5	~19.5	2	~14.0	~22.0	1	~15.0	~21.0
Number of VMs	MapleJuice (s)	Hadoop (s)																																			
5	~12.5	~19.5																																			
4	~12.5	~20.5																																			
3	~12.5	~21.5																																			
2	~14.5	~21.5																																			
1	~14.0	~21.0																																			
Number of VMs	MapleJuice (s)	Hadoop (s)																																			
5	~10.5	~19.5																																			
4	~12.5	~20.5																																			
3	~12.5	~19.5																																			
2	~14.0	~22.0																																			
1	~15.0	~21.0																																			
<p>Regex: “where Video Radio Fiber null None”. In this case, we can see that MapleJuice and SDFS outperform Hadoop. There is not much trend as the number of VMs decreases, possibly because the file size is too small to result in noticeable trends.</p>	<p>Regex: “where Radio” Here, MapleJuice still outperforms Hadoop, but we can see a slight trend in the increase of process time as the number of VMs decreases. This is somewhat an expected behavior as now each worker is responsible for more jobs.</p>																																				
 <table><caption>Time for MapleJuice and Hadoop: "where d1.Detection_ =d2.Interconne"</caption><tr><th>File Size (MB)</th><th>MapleJuice (s)</th><th>Hadoop (s)</th></tr><tr><td>0.3125</td><td>~10.5</td><td>~60.5</td></tr><tr><td>0.625</td><td>~14.5</td><td>~57.5</td></tr><tr><td>1.25</td><td>~15.5</td><td>~55.5</td></tr><tr><td>2.5</td><td>~18.5</td><td>~56.5</td></tr><tr><td>5</td><td>~28.5</td><td>~57.5</td></tr></table>	File Size (MB)	MapleJuice (s)	Hadoop (s)	0.3125	~10.5	~60.5	0.625	~14.5	~57.5	1.25	~15.5	~55.5	2.5	~18.5	~56.5	5	~28.5	~57.5	 <table><caption>Time for MapleJuice and Hadoop: "where d1Detection_ =d2.Interconne and d1.Cabinet_Ty=d2.CNTRL_Note=d3.Install_Da"</caption><tr><th>File Size (MB)</th><th>MapleJuice (s)</th><th>Hadoop (s)</th></tr><tr><td>0.3125</td><td>~8.5</td><td>~57.5</td></tr><tr><td>0.625</td><td>~12.5</td><td>~57.5</td></tr><tr><td>1.25</td><td>~13.5</td><td>~59.5</td></tr><tr><td>2.5</td><td>~24.5</td><td>~56.5</td></tr><tr><td>5</td><td>~29.5</td><td>~54.5</td></tr></table>	File Size (MB)	MapleJuice (s)	Hadoop (s)	0.3125	~8.5	~57.5	0.625	~12.5	~57.5	1.25	~13.5	~59.5	2.5	~24.5	~56.5	5	~29.5	~54.5
File Size (MB)	MapleJuice (s)	Hadoop (s)																																			
0.3125	~10.5	~60.5																																			
0.625	~14.5	~57.5																																			
1.25	~15.5	~55.5																																			
2.5	~18.5	~56.5																																			
5	~28.5	~57.5																																			
File Size (MB)	MapleJuice (s)	Hadoop (s)																																			
0.3125	~8.5	~57.5																																			
0.625	~12.5	~57.5																																			
1.25	~13.5	~59.5																																			
2.5	~24.5	~56.5																																			
5	~29.5	~54.5																																			
<p>Regex: “where d1.Detection_ =d2.interconne” In the example of Join, Hadoop’s runtime doesn’t really increase as the size of the file increases, even though it still runs slower than MapleJuice. This indicates Hadoop might outperform our system when input size goes up.</p>	<p>Regex: “where d1Detection_ =d2.Interconne andd1.Cabinet_Ty=d2.CNTRL_Note=d3.Install_Da” We see a similar pattern as the graph on the left, that MapleJuice still outperforms Hadoop for 0.3 to 5 megabytes files. However, if the file size keeps going up, Hadoop will eventually outperform our system.</p>																																				

Here, we use Champaign Map Databases (Address Points) to plot the below four graphs.+

This dataset has 52,037 rows (entries)

 <p>Time for Maplejuice and Hadoop: "where CompAddNum"</p> <table><tr><th>Number of VMs</th><th>MapleJuice (s)</th><th>Hadoop (s)</th></tr><tr><td>5</td><td>~235</td><td>~80</td></tr><tr><td>4</td><td>~235</td><td>~150</td></tr><tr><td>3</td><td>~255</td><td>~160</td></tr><tr><td>2</td><td>~270</td><td>~225</td></tr><tr><td>1</td><td>~350</td><td>~260</td></tr></table>	Number of VMs	MapleJuice (s)	Hadoop (s)	5	~235	~80	4	~235	~150	3	~255	~160	2	~270	~225	1	~350	~260	 <p>Time for Maplejuice and Hadoop: "where Standard P N Y"</p> <table><tr><th>Number of VMs</th><th>MapleJuice (s)</th><th>Hadoop (s)</th></tr><tr><td>5</td><td>~210</td><td>~75</td></tr><tr><td>4</td><td>~250</td><td>~140</td></tr><tr><td>3</td><td>~245</td><td>~170</td></tr><tr><td>2</td><td>~280</td><td>~220</td></tr><tr><td>1</td><td>~360</td><td>~255</td></tr></table>	Number of VMs	MapleJuice (s)	Hadoop (s)	5	~210	~75	4	~250	~140	3	~245	~170	2	~280	~220	1	~360	~255
Number of VMs	MapleJuice (s)	Hadoop (s)																																			
5	~235	~80																																			
4	~235	~150																																			
3	~255	~160																																			
2	~270	~225																																			
1	~350	~260																																			
Number of VMs	MapleJuice (s)	Hadoop (s)																																			
5	~210	~75																																			
4	~250	~140																																			
3	~245	~170																																			
2	~280	~220																																			
1	~360	~255																																			
<p>Regex: "where CompAddNum"</p> <p>Same as our prediction earlier, as the size of the dataset increases, Hadoop eventually outperforms MapleJuice in this graph. Due to the increase of the size of the dataset, we can also observe a more obvious trend of runtime increase as the number of VMs goes down in this graph.</p>	<p>Regex: "where Standard P N Y"</p> <p>Similar observations can be seen in this graph. As we discussed in the overview section, to further improve the performance of our system, we should randomly select a number from 1 to the number of VMs as the output key for the Maple phase to improve load balancing and partitioning in the Juice phase.</p>																																				
 <p>Time for Maplejuice and Hadoop: "where d1.Residential=d2.Alternate"</p> <table><tr><th>MB</th><th>MapleJuice (s)</th><th>Hadoop (s)</th></tr><tr><td>0.1</td><td>~65</td><td>~60</td></tr><tr><td>1</td><td>~65</td><td>~65</td></tr><tr><td>10</td><td>~75</td><td>~65</td></tr><tr><td>100</td><td>~85</td><td>~65</td></tr><tr><td>1000</td><td>~105</td><td>~60</td></tr></table>	MB	MapleJuice (s)	Hadoop (s)	0.1	~65	~60	1	~65	~65	10	~75	~65	100	~85	~65	1000	~105	~60	 <p>Time for Maplejuice and Hadoop: "where d1.Residential=d2.Alternate and d1.MainAddress=d2.CompleteAddress"</p> <table><tr><th>MB</th><th>MapleJuice (s)</th><th>Hadoop (s)</th></tr><tr><td>0.1</td><td>~70</td><td>~60</td></tr><tr><td>1</td><td>~75</td><td>~65</td></tr><tr><td>10</td><td>~85</td><td>~65</td></tr><tr><td>100</td><td>~95</td><td>~65</td></tr><tr><td>1000</td><td>~125</td><td>~65</td></tr></table>	MB	MapleJuice (s)	Hadoop (s)	0.1	~70	~60	1	~75	~65	10	~85	~65	100	~95	~65	1000	~125	~65
MB	MapleJuice (s)	Hadoop (s)																																			
0.1	~65	~60																																			
1	~65	~65																																			
10	~75	~65																																			
100	~85	~65																																			
1000	~105	~60																																			
MB	MapleJuice (s)	Hadoop (s)																																			
0.1	~70	~60																																			
1	~75	~65																																			
10	~85	~65																																			
100	~95	~65																																			
1000	~125	~65																																			
<p>Regex: "where d1.residential=d2.Alternate"</p> <p>The graph further verifies our deduction on the relationship between MapleJuice's runtime and Hadoop's. When the input file dataset size exceeds 1MB, we can see that Hadoop starts to outperform MapleJuice. When the file size reaches 1GB, the runtime of MapleJuice is almost twice of Hadoop's.</p>	<p>Regex: "where d1.Residential=d2.Alternate and d1.MainAddress=d2.CompleteAddress"</p> <p>Again, similar trend can be seen here except Hadoop now outperforms MapleJuice in all sizes, which could be a result that the regex is more complex, requiring higher runtime during Maple phase and resulting in smaller output.</p>																																				