Netid: Joey Stack(jkstack2) Hanliang Jiang(hj33)

a) Design:

We use our previous MP3 as the basic file system. As described in the former report, we use leader list to maintain a map where each node can track where a file is physically stored, which has the same function as “Hadoop fs -ls” command. With the help of SDFS, we first implement basic maple and juice methods, where maple phase takes as input a few files with a given prefix as a parameter and output a whole bunch of intermediate files per key, and juice phase takes all the temp files in SDFS as input and generate files where result is stored. The process of maple juice functions just like Hadoop does; however, for better user experience, we further implement methods specially for SQL queries. Instead of giving the whole SQL code to the maple and juice executables, we parse the SQL in advance, which better support SQL queries. In order to test the functionalities of our program, we first write a word counter to make sure all the code functions well. After ensuring the system works robustly on a simple word counter, we further developed functionality and stability for the program.

In case of accidental node crash, we use the same way as we did in the former machine problem. After testing our code across multiple number of machines, the system can tolerate up to three arbitrary crashes. This result is actually quite impressive compared to Hadoop cluster, as Hadoop can behave unstably once inconsistency occurs between configuration files and actually active nodes or when the system shut down without running stop shell scripts.

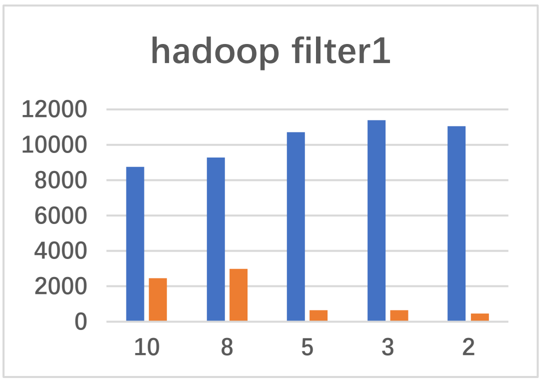
During the process of executing multiple maple and juice jobs, the maple server first get a list of active virtual machines, then shard the files and assign the files fairly to these machines. Upon assigning the tasks, the leader maintains background threads, which waits for the tasks to finish, so does juice. As soon as the active node get the task, they fetch the files from SDFS, and call the maple or juice executable to process corresponding lines and output the intermediate files or result files to the local file storage, then return a list of newly generated files. The node then upload these files to SDFS, which completes the maple or juice jobs.

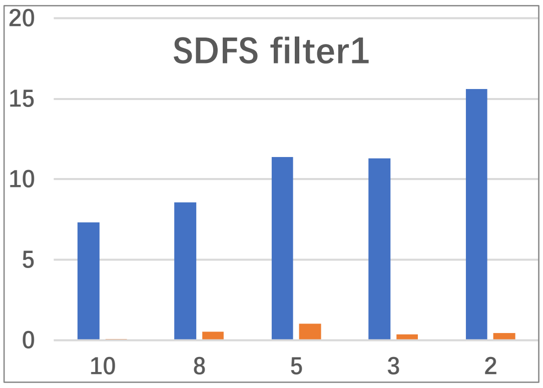
b) Experiment design:

* run 5 trials for each with 3 duplicated tests for each trial and plot trial # by average and deviation of corresponding running time (8 plots total)

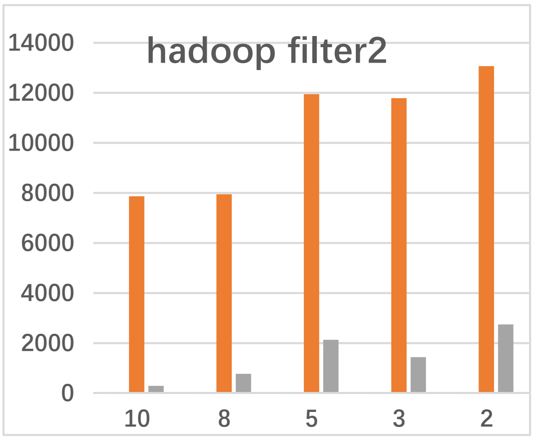
a) For the filter function, we compare the performance of our code and Hadoop mapreduce jobs. For simplicity, we write different scripts for Hadoop and SDFS. We choose a simple regular expression(.*TT.*) and a complex one([A-Za-z]*L/[A-Za-z]*) and run them on Test.csv, a traffic dataset with 35 raws.

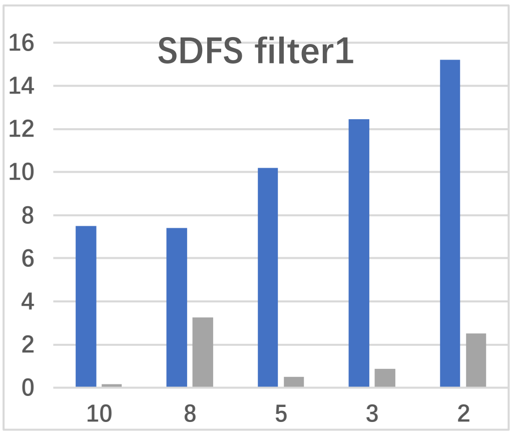
i) for .*TT.*:





ii) for [A-Za-z]*L/[A-Za-z]*:

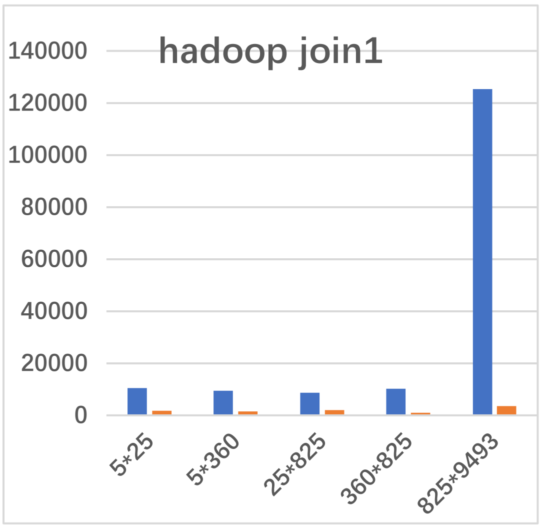


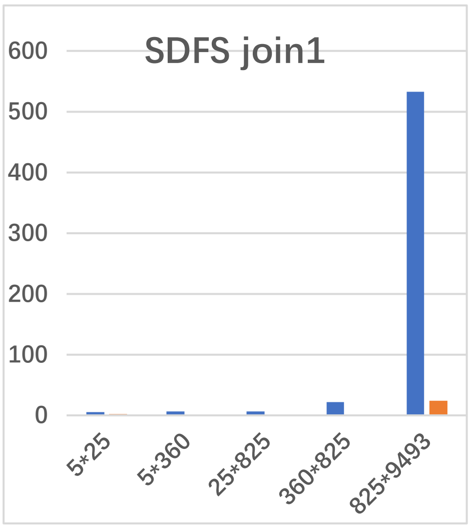


As we can see from the histogram, when more machines are simultaneously executing mapreduce jobs, it takes relatively shorter amount of time, and if the cluster size is smaller than 5 machines, the standard deviations becomes very big, which means running time actually become very unstable. That makes much sense, as more clusters can handle the task more evenly and results in a higher stability.

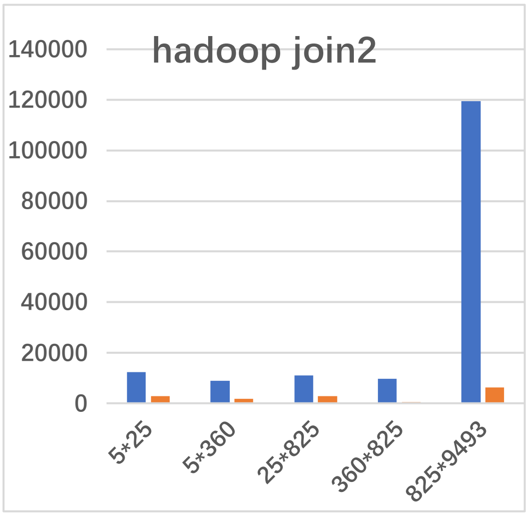
b) For the join function, we compare the performance of our code and Hadoop mapreduce jobs across multiple datasets. For simplicity, we also write different scripts for Hadoop and SDFS. For each pair of datasets, we choose a simple query(e.g. D1.OBJECTID=D2.OBJECTID) and a complex one(D4.units=D5.CompAddNum) and run the tests on D1 through D5, a bunch of municipal statistic datasets with scales ranging from 5 raws to 9493 rows.

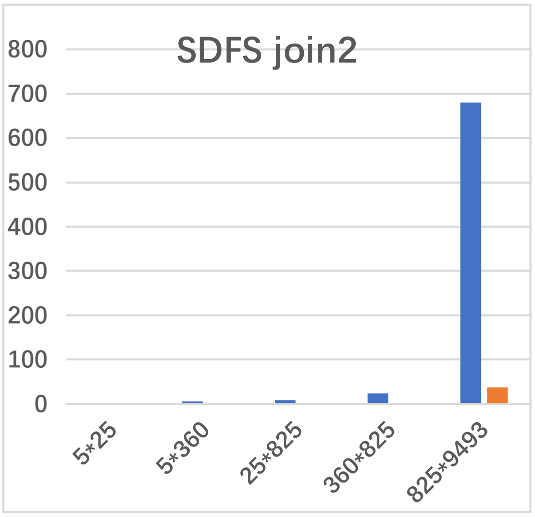
i) Simple query:





ii) Complex query:





From the histogram, we can figure out that when processing on relatively small datasets, the average running time does not vary much; however, when processing relatively big datasets, running time can change a lot. The reason for that is the system scheduling overhead is big compared to a small dataset, so only when the dataset is really big can this factor affect the result.