MP4 Report from Haorong Sun (Haorong4) and Haoyu Zhang (haoyuz3)

The MP4 is built on the distributed file system of MP3.
The heartbeat will also carry synchronization information.
The system is built as a virtual ring structure.

The MapleJuice system contains a master and several workers.
The master will distribute the tasks evenly to every worker, and gather the result returned by each worker process. All the intermedia files are generated inside the DFS.

Fault-tolerance:

the master will record the task and the target files for every worker. If a worker finish processing a file, it will append the content to the DFS and sent master a signal, the Master will remove the corresponding file in the task list of that worker.
If there is some failure happen, the master will pick another live worker and send the remaining task of the failing machine.

Maple:

When the DFS receive the maple command, it will forward the information to master first.
After the master get the command, it will divide the files according to the number of workers specified by the user. It will send the name of executable file, the prefix of intermedia file, and the name list of files for that machine.

After receiving the request from the Master, the worker will start to process the data files, it will gather the information output by the executable file, store in a buffer, once the information is reaching the limit amount, it will append the information and clear the buffer. This is in order to reduce the routing traffic as much as possible. The maple will generate several files according to the number of keys. Each file will contain the data for one key.

Juice:

The Juice is working similarly as the Maple.
When the DFS receive the juice command, it will forward the information to master first.
After the master get the command, it will divide the files according to the number of workers specified by the user. It will send the name of executable file, the prefix of intermedia file, and the name list of files for that machine.

After receiving the request from the Master, the worker will start to process the data files, it will gather the information output by the executable file, store in a buffer, once the information is reaching the limit amount, it will append the information and clear the buffer. The result of the Juice process is one destinated file with all the gathered information.

Append:

In this MP, we implemented the “append” function. In append, we have two steps:

Step 1: We send the data to the receivers. The receivers will buffer the data, and start the timeout clock

Step 2: When one file has been processed, the master will send a confirmation message to each receiver, and the member will write the buffered message to distributed file system. If a worker crashes when it is processing the file, then all receivers will timeout and clear the buffer data. The master will assign a new worker to the remaining files.

Distributed File System:

There will be a master for the entire system. The master will decide where the replicas of file is located. And every time a file’s status is updated, the master will share the file information though the ring structure. Every member who receives the update request will compare the new information’s timestamps to the local ones, and merge incoming list with the local list, choosing the data with the newest timestamp. Then pushing it to all its neighbors. If all the incoming info is older or equal to local info, block the message.

If the current master fails, the system will start a new election. It will check the membership list and choose the member whose IP address has the highest hash value as the new master.

The master process will also check if there are needs for re-replication, we set the master to check through its failure list for every 5 seconds. If there are new failures, the master will go through the file information list, and choose one remaining replica to send file copy to new positions.

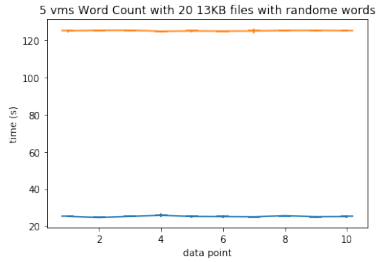
We use TCP connection to transfer files. We transfer files in blocks of 1024KB. Each node has two threads listening for incoming connections. One thread listens for connections that query master, since every node may be a master. The other thread listens for connections, which comes from other nodes, that put or get files.

While debugging the system, MP1 is very helpful, since it could help us to gather all the file information from 10 vms to one terminal, saving lots of time and effert.

The following measurements are all measured using Bmon.

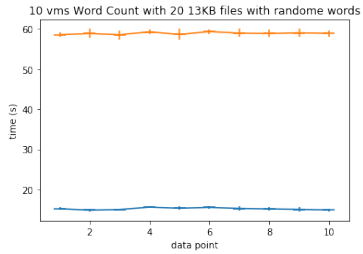
We have 20 text files with size about 13 KB. We measure ten data point for the running of both MapleJuice and Hadoop by using the same executable files. And here is the graph:

Measurement 1:



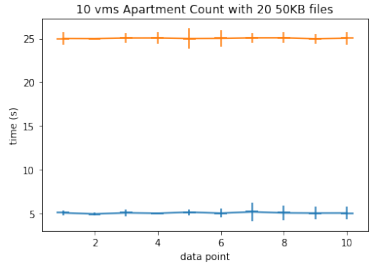
This is the a error bar graph of measurement with 5 vms, our program can finish the task with a pretty stable time around 25 seconds, while Hadoop may usually take around 2 minutes

Measurement 2:



This is the error bar graph measurement with 10 vms, our program can finish the task with a pretty stable time around 15 seconds, while Hadoop may usually take around 58 seconds.

Measurement 3:



This is another application about counting the number of apartments with more than 10 units, our program can finish the task with a pretty stable time around 5 seconds, while Hadoop may usually take around 25 seconds.