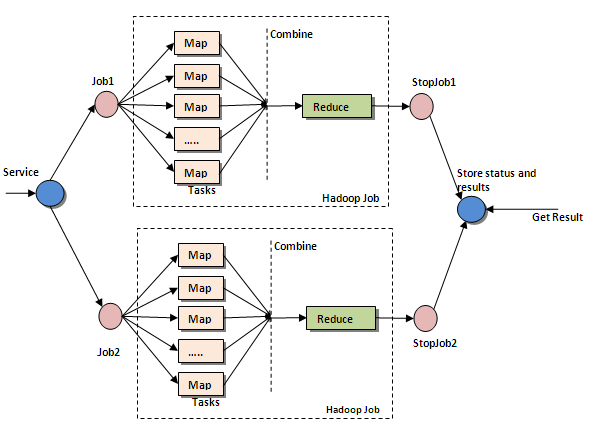
Hadoop Map-Reduce

Nisha Halyal

UTD\_ID: 2021165156

**Introduction:**

The term MapReduce actually refers to distinct tasks that Hadoop programs perform. The first is the map job, which takes a set of data and converts it into another set of data, where individual elements are broken down into key/value pairs. The reduce job takes the output from a map as input and combines those data key/value pairs into a smaller set of tuples.



The jobs submitted by client are fed to Name node as jar file which will be replicated across slave/data nodes as blocks. The mapper function does mapping over data whose result will be shuffled and sorted to get the output of mapper task. The output from mapper task is the input for reducer task as key/value pair where the reducer will reduce the pair to get final output.

**About project:**

In this project we write a MapReduce program to compute the total crime incidents of each crime type in each region and analyze the result in terms of CPU usage, memory usage, number of mapper and reducer tasks and their performance for i) many small input flies ii) one large input file or consolidated data, under different settings.

The project includes an input file with huge data related to crime database. It contains many CSV files with street-level crime and outcome information. There are total 1341 files. We have a fixed block size say 64 MB. If each file is less than or equal to block size then it is considered to one block and if any file size exceeds block size then it is split into smaller chunks which are less than or equal to block size.

**Scenarios:**

1. Providing large number of smaller files for region definition as first digit of the coordinates:

Number of Maps =1341

Number of Reducers =1

Total committed heap usage (bytes) =300578439168

CPU time spent (ms) =1851940

Physical memory (bytes) snapshot =276171337728

Virtual memory (bytes) snapshot =1407369281536

1. Providing a single file with a very large data set for region definition as first digit of the coordinates:

Number of Maps =33

Number of Reducers =1

Total committed heap usage (bytes) = 11756699648

CPU time spent (ms) = 174390

Physical memory (bytes) snapshot = 11530608640

Virtual memory (bytes) snapshot = 35662807040

1. Providing large number of smaller files for region definition as first three digits of the coordinates:

Number of Maps =1341

Number of Reducers =1

Total committed heap usage (bytes) = 300948914176

CPU time spent (ms) = 1933830

Physical memory (bytes) snapshot = 275685158912

Virtual memory (bytes) snapshot = 1407359000576

1. Providing a single file with a very large data set for region definition as first three digits of the coordinates:

Number of Maps =33

Number of Reducers =1

Total committed heap usage (bytes) = 11667832832

CPU time spent (ms) = 207830

Physical memory (bytes) snapshot = 11580551168

Virtual memory (bytes) snapshot = 35663065088

**Analysis:**

* By the above shown result we can observe that heap usage in bytes is less for consolidated record i.e. a single file with a very large data set compared to number of smaller files.
* We can also observe that the CPU usage / CPU time spent is also less for a single file with a very large data set compared to number of smaller files, i.e. CPU for consolidated for region definition first digit =2.9 min

CPU for consolidated for region definition first three digits =3.46 min

CPU for many smaller files for region definition first digit =30.86 min

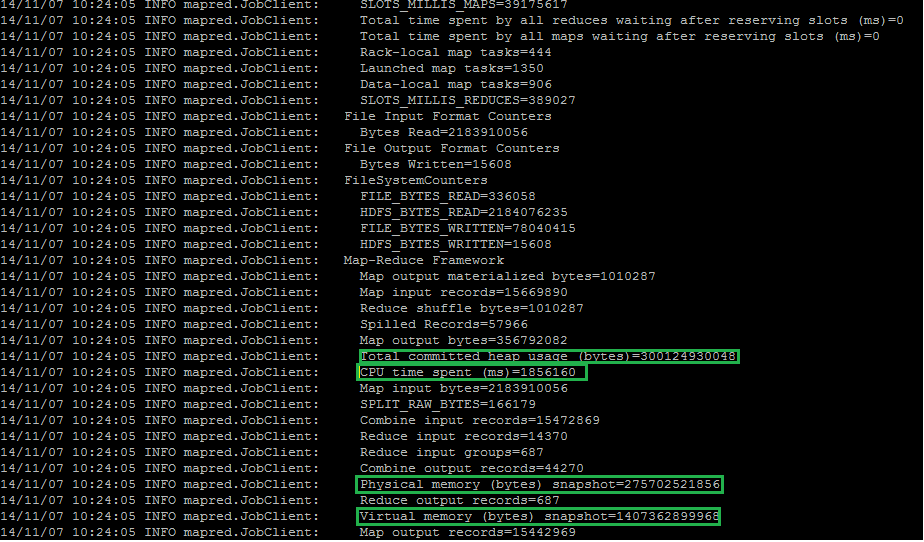
CPU for many smaller files for region definition first three digits =32.23 min

The above analysis is with respect to CPU and Memory usage for many small files and single large file for region with different settings.

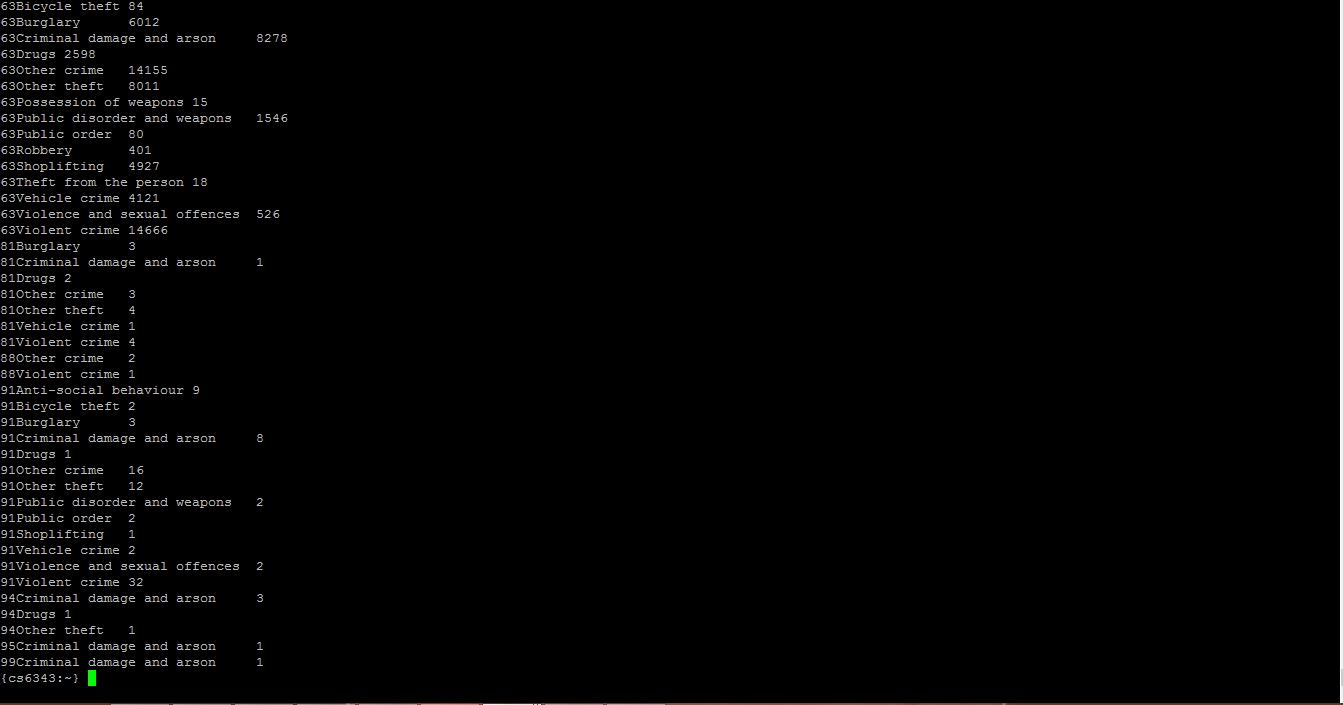
My observation for the given input file is that number of files is 1341; hence I observed for large number of small files, number of blocks distributed over data nodes is 1341, i.e. number of blocks =1341 and each block has an individual mapper task hence number of map tasks =1341 (as shown above) and as per my observation reducer task is set to 1 as default until we manually set it to some other integer using *conf.setNumReduceTasks (int)* command; whereas for single large file number of files is only 1 but as it exceeds the block size it is split and number of blocks obtained is 33 (as explained above in introduction) and as observed by me number of map tasks =number of blocks, hence 33 and as observed reducer task is set to 1 as default.

Here are few screen shots for my observation:

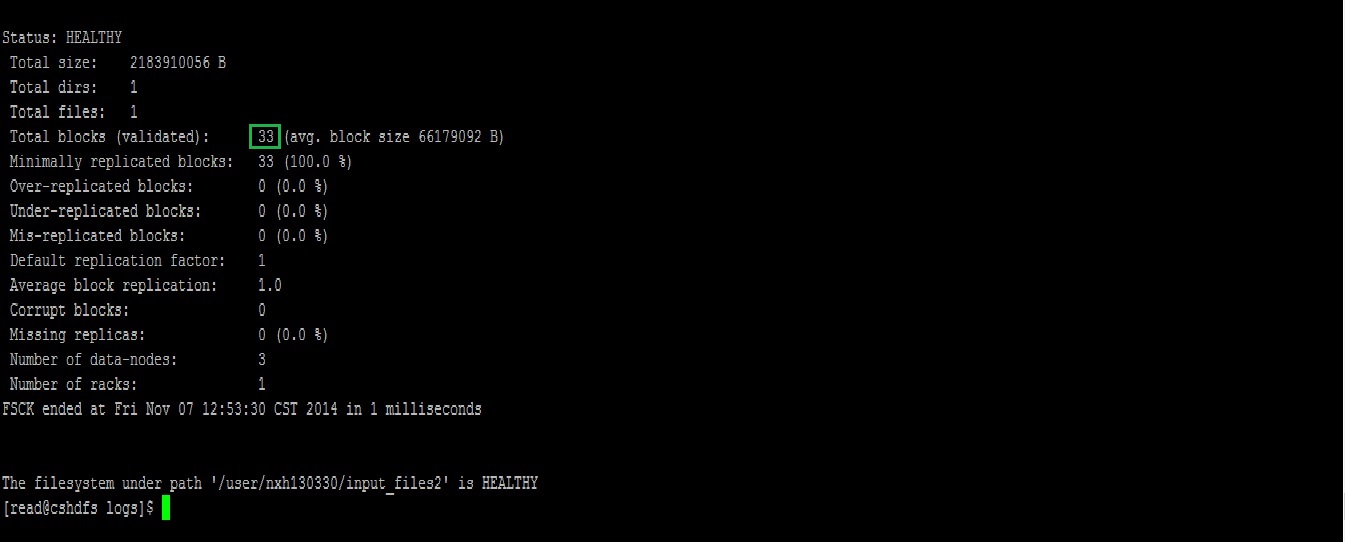
For large number of small files:



This is the output file for region definition with first digit:



Block size for consolidated file:



**Analysis according to different region definition:**

1. As per my observation Performance of CPU is always less for region definition with first digit whereas it is more for region definition with first three digits for both cases of many small files and a large consolidated file.

* Region definition with first digit :

1. For large number of small files: CPU time =30.86 min
2. For a huge consolidated file: CPU time =2.9 min

* Region definition with first three digit :

1. For large number of small files: CPU time =32.23 min
2. For a huge consolidated file: CPU time =3.46 min

2. For region definition with first three digits has large number of output records as compared to region with definition of first digit.

**The result after introducing error in couple of CSV files:**

After introducing error in few files the system throws error as shown below (to obtain this error message program with name “error.java should be run).

*java.lang.ArrayIndexOutOfBoundsException: 7*

*at RegionCrimeCount$Map.map(RegionCrimeCount.java:5)*

*at RegionCrimeCount $Map.map(RegionCrimeCount.java:2)*

*at org.apache.hadoop.mapred.MapRunner.run(MapRunner.java:50)*

*at org.apache.hadoop.mapred.MapTask.runOldMapper(MapTask.java:430)*

*at org.apache.hadoop.mapred.MapTask.run(MapTask.java:366)*

*at org.apache.hadoop.mapred.Child$4.run(Child.java:255)*

*at java.security.AccessController.doPrivileged(Native Method)*

*at javax.security.auth.Subject.doAs(Subject.java:396)*

*at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1190)*

*at org.apache.hadoop.mapred.Child.main(Child.java:249)*

**Analysis for different number of reducers’ tasks:**

To set different number of reducer tasks I used the following command:

*JobConf.setNumReduceTasks(int)*

where int is the number of reducer tasks set (this line is commented in all the .java files provided by me).

As the number of reducer tasks increase CPU usage also increases.

CPU time with 1 reducer = 1851940

CPU time with 7 reducers = 2072330

**Here is the output of jobtracker for program with set reducer tasks:**

JobTracker Summary:

2014-11-07 19:28:51,176 INFO org.apache.hadoop.mapred.JobTracker: jobToken generated and stored with users keys in /home/hadoop/mapred/job\_201411071727\_0041/jobToken

2014-11-07 19:28:51,278 INFO org.apache.hadoop.mapred.JobInProgress: job\_201411071727\_0041:

***nMaps=1341 nReduces=7*** max=-1

2014-11-07 19:28:51,278 INFO org.apache.hadoop.mapred.JobTracker: Job job\_201411071727\_0041 added successfully for user 'nxh130330' to queue 'default'

2014-11-07 19:28:51,278 INFO org.apache.hadoop.mapred.AuditLogger: USER=nxh130330 IP=10.176.92.129 OPERATION=SUBMIT\_JOB TARGET=job\_201411071727\_0041 RESULT=SUCCESS.

**Analysis for performance on different mapper tasks:**

The performance of map and reduce increase when input is a large data file because the number of blocks produced is less and as map depends on block size, number of mapper tasks is less.

**Analysis on how system handles error:**

The system assigns the replicas of block over all slave nodes. If there is an error for any input then the worker sends the problem to the master node. If master sees failures more than attempt number set for the same record then the master gives up the task and stops all worker replicas for the same record.

**Analysis** **on jobtracker and namenode summary:**

(The jobtracker and namenode summary for each scenario is present in folders in data file folder.)

JobTracker gives information on number of splits for input data file, number of map task and reduce task and result of job.

**Conclusion:**

The conclusion drawn from above analysis for various runs is that region definition with first digit is efficient than region definition with first three digits. And providing single large consolidated file is efficient compared to providing many small files. Lesser the reducer tasks better is the CPU utilization.