Assignment 1:

Problem statement:

Configure client node for hadoop cluster

Solution:

Okay so we have our first cluster up and running! Now let us do two things here. First, we will see how to configure the Hadoop client and then we will run an interesting Map Reduce program

Configuring Hadoop Client

We will use another Linux machine as the Hadoop client. Follow the steps mentioned in the last assignment to make a clone of the base machine. Now, change the hostname of this machine as client.mycluster.com and assign it an IP address belonging to our cluster. Also edit the hosts file on this machine to include the IP addresses of all other machines in the cluster.

As you have already figured out, this machine already has Java and all the Hadoop files configured. For a Hadoop client, we don't need all the configuration settings.

Keep the core-site.xml file as it is. Hadoop client will read this file to understand where Namenode is running and will connect to it.

Keep the mapred-site.xml file also the same. This way the client will know where Job Tracker is running so that it can submit jobs.

Also keep a copy of yarn-site.xml

Edit the hdfs-site.xml file and keep only the replication factor and block size properties.

Empty the masters and slaves file

Now we have configured the client. You can run all the HDFS commands from the client and it will be executed on the cluster.

Running a Sample MapReduce Program (Hadoop 1)

Power on our 6 node cluster and start all the daemons. Run the following commands and make sure that everything is working fine.

[hadoop@namenode ~]$ hadoop dfsadmin -report

Configured Capacity: 55708704768 (51.88 GB)

Present Capacity: 35268349952 (32.85 GB)

DFS Remaining: 35268235264 (32.85 GB)

DFS Used: 114688 (112 KB)

DFS Used%: 0%

Under replicated blocks: 0

Blocks with corrupt replicas: 0

Missing blocks: 0

Datanodes available: 3 (3 total, 0 dead)

Name: 192.168.10.103:50010

Decommission Status : Normal

Configured Capacity: 18569568256 (17.29 GB)

DFS Used: 36864 (36 KB)

Non DFS Used: 6814085120 (6.35 GB)

DFS Remaining: 11755446272(10.95 GB)

DFS Used%: 0%

DFS Remaining%: 63.3%

Last contact: Fri Jul 24 01:57:32 EDT 2015

Name: 192.168.10.104:50010

Decommission Status : Normal

Configured Capacity: 18569568256 (17.29 GB)

DFS Used: 36864 (36 KB)

Non DFS Used: 6812561408 (6.34 GB)

DFS Remaining: 11756969984(10.95 GB)

DFS Used%: 0%

DFS Remaining%: 63.31%

Last contact: Fri Jul 24 01:57:33 EDT 2015

Name: 192.168.10.105:50010

Decommission Status : Normal

Configured Capacity: 18569568256 (17.29 GB)

DFS Used: 40960 (40 KB)

Non DFS Used: 6813708288 (6.35 GB)

DFS Remaining: 11755819008(10.95 GB)

DFS Used%: 0%

DFS Remaining%: 63.31%

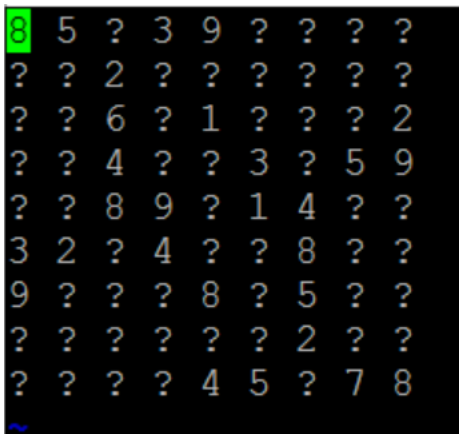
Last contact: Fri Jul 24 01:57:32 EDT 2015

We will first run a sample program which can solve a Sudoku puzzle. In order to do this we need to input the Sudoku puzzle to Hadoop. You can either create your own puzzle, save it as text and give it to Hadoop or use the one available as an example. If you are using your own puzzle, create it in such a way that there are rows and space separated columns and put a ? for blocks that has to be resolved.

Our sample puzzle is located in
/home/hadoop/hadoop/src/examples/org/apache/hadoop/examples/dancing

The file name is puzzle1.dta.

Copy this file and put it in your home directory. If you open it, you can see the puzzle as follows



So that is our puzzle. Once the file has been copied, run the following command to solve it using MapReduce. This uses the example program which is there in /home/Hadoop/Hadoop

[*hadoop@namenode hadoop]$ hadoop jar hadoop-examples-1.2.1.jar sudoku /home/hadoop/puzzle1.dta*

Solving /home/hadoop/puzzle1.dta

8 5 1 3 9 2 6 4 7

4 3 2 6 7 8 1 9 5

7 9 6 5 1 4 3 8 2

6 1 4 8 2 3 7 5 9

5 7 8 9 6 1 4 2 3

3 2 9 4 5 7 8 1 6

9 4 7 2 8 6 5 3 1

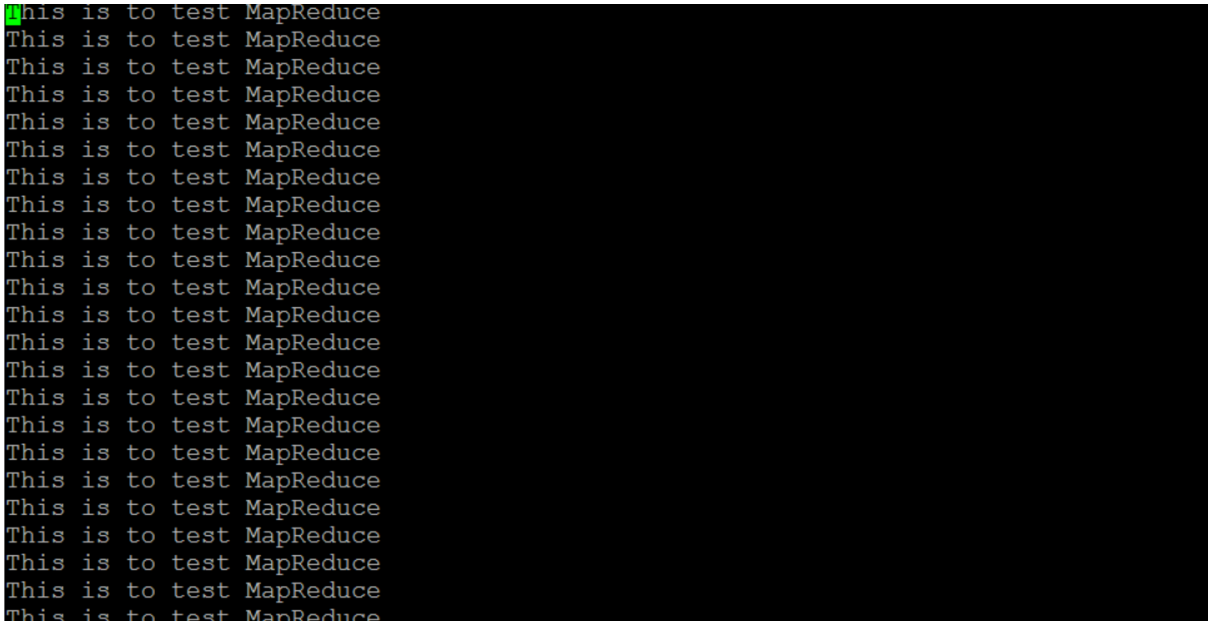
1 8 5 7 3 9 2 6 4

2 6 3 1 4 5 9 7 8

Found 1 solutions.

Well, this was so quick and we could not make out what has happened! So let us try another example. This time we will use the wordcount program to count the number of unique entries in a sample text file.

First create a sample file in your home directory. I have created a file called mydata.txt and it has the following line repeating many times. Close to 250 times I think.



Now create a folder named input in HDFS and copy this file there

[hadoop@namenode ~]$ hadoop fs -mkdir /input

[hadoop@namenode ~]$ hadoop fs -put mydata.txt /input

Now go to the Hadoop folder and run the program as follows.

[hadoop@namenode hadoop]$ hadoop jar hadoop-examples-1.2.1.jar wordcount /input/mydata.txt /output

Here, we are giving the HDFS input path, and mentioning the output to be saved in a folder named /output. HDFS will automatically create this folder to save the output. You will see something similar to this.

15/07/24 02:16:14 INFO util.NativeCodeLoader: Loaded the native-hadoop library

15/07/24 02:16:14 INFO input.FileInputFormat: Total input paths to process : 1

15/07/24 02:16:14 WARN snappy.LoadSnappy: Snappy native library not loaded

15/07/24 02:16:15 INFO mapred.JobClient: Running job: job_local1126220904_0001

15/07/24 02:16:15 INFO mapred.LocalJobRunner: Waiting for map tasks

15/07/24 02:16:15 INFO mapred.LocalJobRunner: Starting task: attempt_local1126220904_0001_m_000000_0

15/07/24 02:16:15 INFO util.ProcessTree: setsid exited with exit code 0

15/07/24 02:16:15 INFO mapred.Task: Using ResourceCalculatorPlugin : org.apache.hadoop.util.LinuxResourceCalculatorPlugin@66587927

15/07/24 02:16:15 INFO mapred.MapTask: Processing split: hdfs://namenode.mycluster.com:8020/input/mydata.txt:0+8424

15/07/24 02:16:15 INFO mapred.MapTask: io.sort.mb = 100

15/07/24 02:16:15 INFO mapred.MapTask: data buffer = 79691776/99614720

15/07/24 02:16:15 INFO mapred.MapTask: record buffer = 262144/327680

15/07/24 02:16:15 INFO mapred.MapTask: Starting flush of map output

15/07/24 02:16:15 INFO mapred.MapTask: Finished spill 0

15/07/24 02:16:15 INFO mapred.Task: Task:attempt_local1126220904_0001_m_000000_0 is done. And is in the process of commiting

15/07/24 02:16:15 INFO mapred.LocalJobRunner:

15/07/24 02:16:15 INFO mapred.Task: Task 'attempt_local1126220904_0001_m_000000_0' done.

15/07/24 02:16:15 INFO mapred.LocalJobRunner: Finishing task: attempt_local1126220904_0001_m_000000_0

15/07/24 02:16:15 INFO mapred.LocalJobRunner: Map task executor complete.

15/07/24 02:16:16 INFO mapred.Task: Using ResourceCalculatorPlugin : org.apache.hadoop.util.LinuxResourceCalculatorPlugin@5e713aa4

15/07/24 02:16:16 INFO mapred.LocalJobRunner:

15/07/24 02:16:16 INFO mapred.Merger: Merging 1 sorted segments

15/07/24 02:16:16 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 58 bytes

15/07/24 02:16:16 INFO mapred.LocalJobRunner:

15/07/24 02:16:16 INFO mapred.Task: Task:attempt_local1126220904_0001_r_000000_0 is done. And is in the process of commiting

15/07/24 02:16:16 INFO mapred.LocalJobRunner:

15/07/24 02:16:16 INFO mapred.Task: Task attempt_local1126220904_0001_r_000000_0 is allowed to commit now

15/07/24 02:16:16 INFO mapred.JobClient: map 100% reduce 0%

15/07/24 02:16:16 INFO output.FileOutputCommitter: Saved output of task 'attempt_local1126220904_0001_r_000000_0' to /output

15/07/24 02:16:16 INFO mapred.LocalJobRunner: reduce > reduce

15/07/24 02:16:16 INFO mapred.Task: Task 'attempt_local1126220904_0001_r_000000_0' done.

15/07/24 02:16:17 INFO mapred.JobClient: map 100% reduce 100%

15/07/24 02:16:17 INFO mapred.JobClient: Job complete: job_local1126220904_0001

15/07/24 02:16:17 INFO mapred.JobClient: Counters: 22

15/07/24 02:16:17 INFO mapred.JobClient: Map-Reduce Framework

15/07/24 02:16:17 INFO mapred.JobClient: Spilled Records=10

15/07/24 02:16:17 INFO mapred.JobClient: Map output materialized bytes=62

15/07/24 02:16:17 INFO mapred.JobClient: Reduce input records=5

15/07/24 02:16:17 INFO mapred.JobClient: Virtual memory (bytes) snapshot=0

15/07/24 02:16:17 INFO mapred.JobClient: Map input records=312

15/07/24 02:16:17 INFO mapred.JobClient: SPLIT_RAW_BYTES=116

15/07/24 02:16:17 INFO mapred.JobClient: Map output bytes=14352

15/07/24 02:16:17 INFO mapred.JobClient: Reduce shuffle bytes=0

15/07/24 02:16:17 INFO mapred.JobClient: Physical memory (bytes) snapshot=0

15/07/24 02:16:17 INFO mapred.JobClient: Reduce input groups=5

15/07/24 02:16:17 INFO mapred.JobClient: Combine output records=5

15/07/24 02:16:17 INFO mapred.JobClient: Reduce output records=5

15/07/24 02:16:17 INFO mapred.JobClient: Map output records=1560

15/07/24 02:16:17 INFO mapred.JobClient: Combine input records=1560

15/07/24 02:16:17 INFO mapred.JobClient: CPU time spent (ms)=0

15/07/24 02:16:17 INFO mapred.JobClient: Total committed heap usage (bytes)=321527808

15/07/24 02:16:17 INFO mapred.JobClient: File Input Format Counters

15/07/24 02:16:17 INFO mapred.JobClient: Bytes Read=8424

15/07/24 02:16:17 INFO mapred.JobClient: FileSystemCounters

15/07/24 02:16:17 INFO mapred.JobClient: HDFS_BYTES_READ=16848

15/07/24 02:16:17 INFO mapred.JobClient: FILE_BYTES_WRITTEN=424978

15/07/24 02:16:17 INFO mapred.JobClient: FILE_BYTES_READ=285946

15/07/24 02:16:17 INFO mapred.JobClient: HDFS_BYTES_WRITTEN=46

15/07/24 02:16:17 INFO mapred.JobClient: File Output Format Counters

15/07/24 02:16:17 INFO mapred.JobClient: Bytes Written=46

Note that it is generating a unique job ID for each job. You can also see the status of Map and Reduce jobs.

You may try to copy multiple text files of bigger size to the input folder and try the same job. Give the input path as /input/* . This will take some time to complete and you can check the web UI of job tracker to see the status as well.