MapReduce

18 November 2025 09:04

Elections

	Stage 1	Stage 2
MH	A - 30 B - 20 C - 10	A - 60 B - 35 C - 16
AP	A - 20 B - 10 C - 5	Party A is winner !!!
GOA	A - 10 B - 5 C - 1	

This is example of MapReduce

What is MapReduce

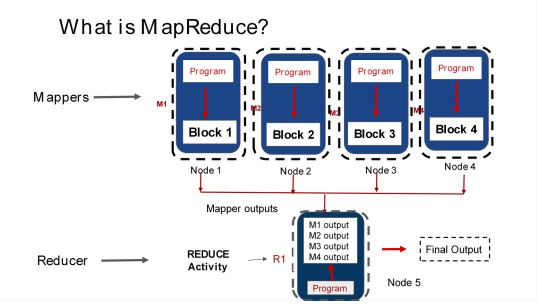
It's a programming paradigm which can process the data on multiple systems.
It works on divide and conquer principle.

2 stages of MapReduce

1. Map ---> (Compulsory)
2. Reduce -> (optional)

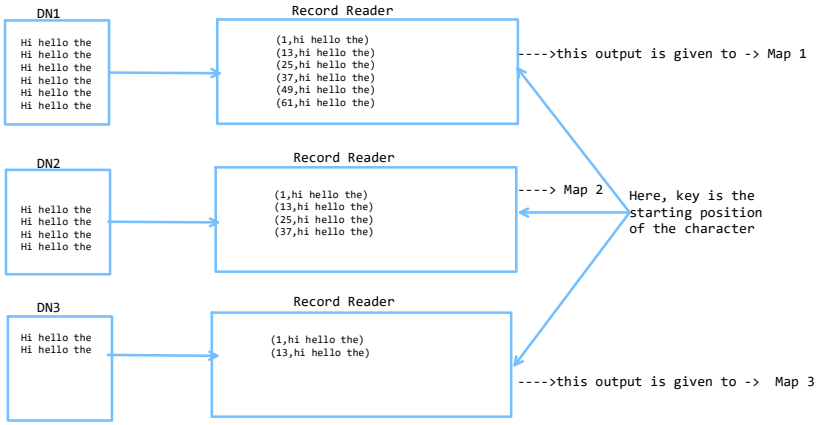
By default :
no. of mappers = no. of blocks (128 MB)
no. of reducer = 1
but we can add more reducers.

- Both Map and Reduce work on Key - Value pairs.

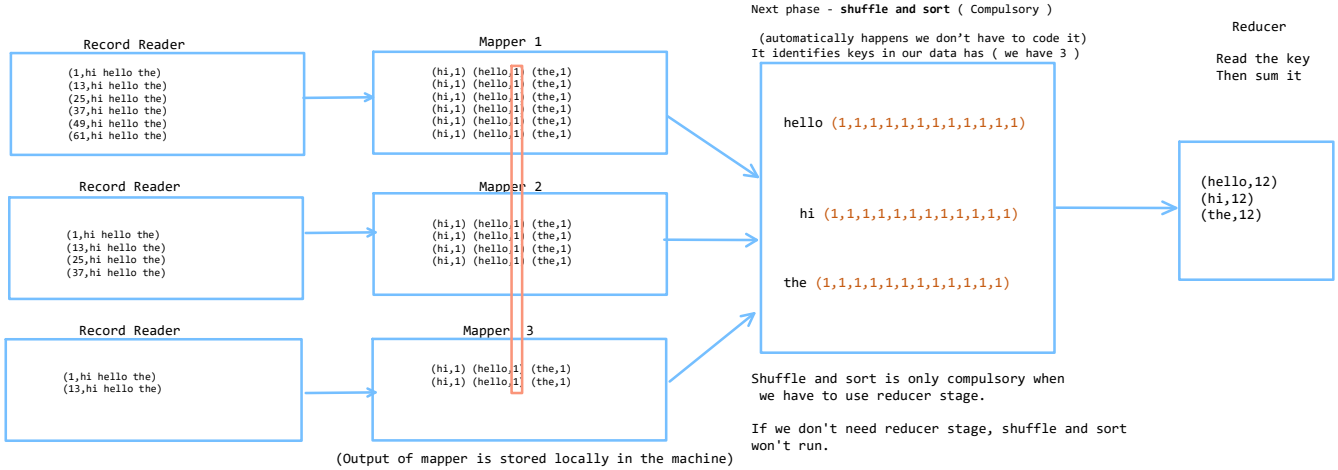


When we write a mapper program and we run it,
- it identifies the data.
- for each block (128MB) 1 mapper is assigned.
- the mapper will output the values (key-value pairs).
- in such way, all mappers output the processed data.
- reducer combines the outputs and gives the reduced output (key-value pairs)

Ex. Return the word count

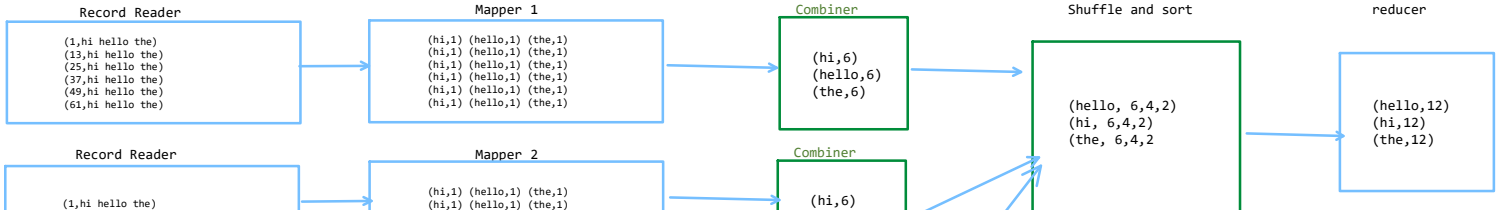


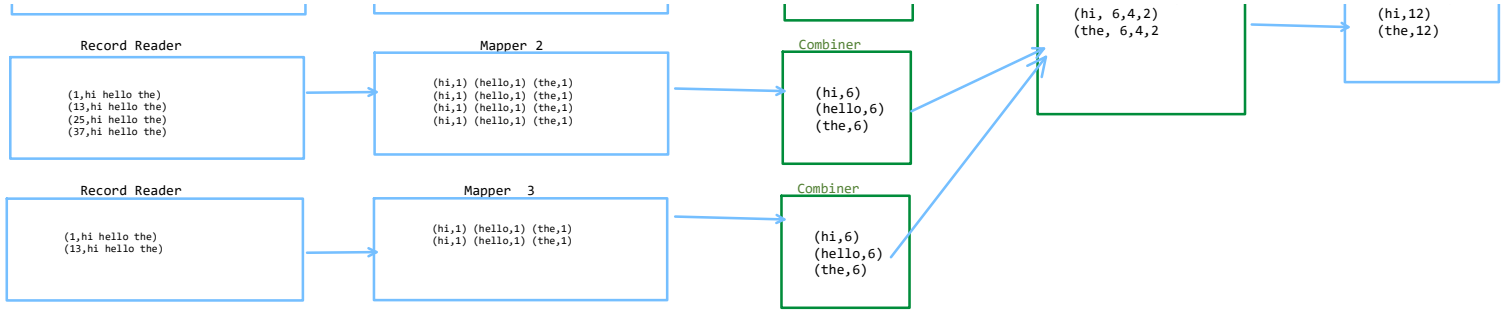
In MapReduce a library called RecordReader (built-in) will inspect data
- check if it is key-value pair, if not it will convert data into key-value pair



OR

COMBINER is a reducer which runs locally
It's an optional stage
Combiner makes execution faster





On HDFS,
   Home directory -> /user/hadoop

Create EMR, connect with mobaXterm

Run commands from file FsShellCommands_Updated.txt

All hadoop commands start with hdfs dfs

1. hdfs dfs
2. hdfs dfs -ls /user/hadoop   --> prints directory
3. hdfs dfs -ls -R /user/hadoop -> Recursively prints all the directories and sub directories
4. hdfs dfs mkdir /user/hadoop/StocksDir -> create a directory
5. hdfs dfs -ls -> -> shows what is in home directory

1. Cd /usr/share/dict
2. Ls
3. Nano words
4. Cd

1. hdfs dfs -put /usr/share/dict/words /user/hadoop/StocksDir
2. hdfs dfs -tail /user/hadoop/StocksDir/words

1. cd -> Home directory

Go to emr cluster : applications : Hue UI URL : username hadoop, password Demo@123
Click on files from vertical menu
   We can see StocksDir, inside it our words file

1. hdfs dfs -mkdir /user/hadoop/AnotherStocksDir
2. hdfs dfs -ls -R /user/hadoop/

1. hdfs dfs -mkdir /user/hadoop/ThirdStocksDir
2. hdfs dfs -ls -R /user/hadoop/

1. hdfs dfs -cp /user/hadoop/StocksDir/words /user/hadoop/AnotherStocksDir/newfile1.txt
2. hdfs dfs -cp /user/hadoop/StocksDir/words /user/hadoop/AnotherStocksDir/newfile2.txt
3. hdfs dfs -cp /user/hadoop/StocksDir/words /user/hadoop/AnotherStocksDir/newfile3.txt
4. hdfs dfs -ls -R /user/hadoop/

```
[hadoop@ip-172-31-68-208 dict]$ hdfs dfs -mkdir /user/hadoop/AnotherStocksDir
[hadoop@ip-172-31-68-208 dict]$ hdfs dfs -ls -R /user/hadoop/
drwxr-xr-x   - hadoop   hadoop          0 2025-11-18 07:22 /user/hadoop/AnotherStocksDir
drwxr-xr-x   - ec2-user hadoop          0 2025-11-18 07:11 /user/hadoop/StocksDir
-rw-r--r--   1 hadoop   hadoop    4953680 2025-11-18 07:11 /user/hadoop/StocksDir/words
[hadoop@ip-172-31-68-208 dict]$ hdfs dfs -mkdir /user/hadoop/ThirdStocksDir
[hadoop@ip-172-31-68-208 dict]$ hdfs dfs -ls -R /user/hadoop/
drwxr-xr-x   - hadoop   hadoop          0 2025-11-18 07:22 /user/hadoop/AnotherStocksDir
drwxr-xr-x   - ec2-user hadoop          0 2025-11-18 07:11 /user/hadoop/StocksDir
-rw-r--r--   1 hadoop   hadoop    4953680 2025-11-18 07:11 /user/hadoop/StocksDir/words
drwxr-xr-x   - hadoop   hadoop          0 2025-11-18 07:24 /user/hadoop/ThirdStocksDir
[hadoop@ip-172-31-68-208 dict]$ hdfs dfs -cp /user/hadoop/StocksDir/words /user/hadoop/AnotherStocksDir/newfile1.txt
[hadoop@ip-172-31-68-208 dict]$ hdfs dfs -cp /user/hadoop/StocksDir/words  /user/hadoop/AnotherStocksDir/newfile2.txt
[hadoop@ip-172-31-68-208 dict]$
[hadoop@ip-172-31-68-208 dict]$ hdfs dfs -cp /user/hadoop/StocksDir/words  /user/hadoop/AnotherStocksDir/newfile3.txt
[hadoop@ip-172-31-68-208 dict]$ hdfs dfs -ls -R /user/hadoop/
drwxr-xr-x   - hadoop   hadoop          0 2025-11-18 07:26 /user/hadoop/AnotherStocksDir
-rw-r--r--   1 hadoop   hadoop    4953680 2025-11-18 07:26 /user/hadoop/AnotherStocksDir/newfile1.txt
-rw-r--r--   1 hadoop   hadoop    4953680 2025-11-18 07:26 /user/hadoop/AnotherStocksDir/newfile2.txt
-rw-r--r--   1 hadoop   hadoop    4953680 2025-11-18 07:26 /user/hadoop/AnotherStocksDir/newfile3.txt
drwxr-xr-x   - ec2-user hadoop          0 2025-11-18 07:11 /user/hadoop/StocksDir
-rw-r--r--   1 hadoop   hadoop    4953680 2025-11-18 07:11 /user/hadoop/StocksDir/words
drwxr-xr-x   - hadoop   hadoop          0 2025-11-18 07:25 /user/hadoop/ThirdStockDir
drwxr-xr-x   - hadoop   hadoop          0 2025-11-18 07:24 /user/hadoop/ThirdStocksDir
[hadoop@ip-172-31-68-208 dict]$ ^C
[hadoop@ip-172-31-68-208 dict]$
```

It is replication factors (no. of copies) -> By default 1

hdfs dfs -setrep 3 /user/hadoop/StocksDir/words -> sets the replication factor to 3

```
[hadoop@ip-172-31-68-208 dict]$ hdfs dfs -setrep 3 /user/hadoop/StocksDir/words
Replication 3 set: /user/hadoop/StocksDir/words
[hadoop@ip-172-31-68-208 dict]$ ^C
[hadoop@ip-172-31-68-208 dict]$ hdfs dfs -ls -R /user/hadoop/
drwxr-xr-x   - hadoop   hadoop          0 2025-11-18 07:26 /user/hadoop/AnotherStocksDir
-rw-r--r--   1 hadoop   hadoop    4953680 2025-11-18 07:26 /user/hadoop/AnotherStocksDir/newfile1.txt
-rw-r--r--   1 hadoop   hadoop    4953680 2025-11-18 07:26 /user/hadoop/AnotherStocksDir/newfile2.txt
-rw-r--r--   1 hadoop   hadoop    4953680 2025-11-18 07:26 /user/hadoop/AnotherStocksDir/newfile3.txt
drwxr-xr-x   - ec2-user hadoop          0 2025-11-18 07:11 /user/hadoop/StocksDir
-rw-r--r--   3 hadoop   hadoop    4953680 2025-11-18 07:11 /user/hadoop/StocksDir/words
drwxr-xr-x   - hadoop   hadoop          0 2025-11-18 07:25 /user/hadoop/ThirdStockDir
drwxr-xr-x   - hadoop   hadoop          0 2025-11-18 07:24 /user/hadoop/ThirdStocksDir
[hadoop@ip-172-31-68-208 dict]$
```

To delete a directory, it must be empty unless we do force delete.

```
hdfs dfs -rm /user/hadoop/StocksDir/word   -> delete the file
hdfs dfs -rmdir /user/hadoop/StocksDir/    -> remove the directory
hdfs dfs -rm -R  /user/hadoop/StocksDir/   -> force delete
```

Kerberos is default security framework in hadoop

- Rack Awareness (administration side of hadoop) IMP for CCEE at least 1 question is asked

Rack is a cupboard

Hadoop will distribute our blocks in multiple racks..

Ex. We have 3 replicas of B1 block,
Hadoop will place 1st replica in Rack 1, even if hadoop placed replica 2 in same Rack by any chance, Strictly hadoop will keep 3rd replica in Another rack

This results into Fault Tolerence
And faster speed
Users get access to the blocks that are nearest to them.
No load on single block.

Very IMP

core-site.xml
hdfs-site.xml
yarn-site.xml

Will be auto created when we install a hadoop cluster,
our hadoop cluster's configuration is in these files

We can change functionalities in these files,
Ex. If we want to change the default replication factor, we can change that in hdfs_site.xml

Cd /etc/hadoop/conf
Ls

↑ Here are the files.

Exercise

Q1 : Output max closing price for each stock.

First 2 rows are :

NYSE,AEA,2010-02-08,4.42,4.42,4.21,4.24,205500,4.24
NYSE,AEA,2010-02-05,4.42,4.54,4.22,4.41,194300,4.41

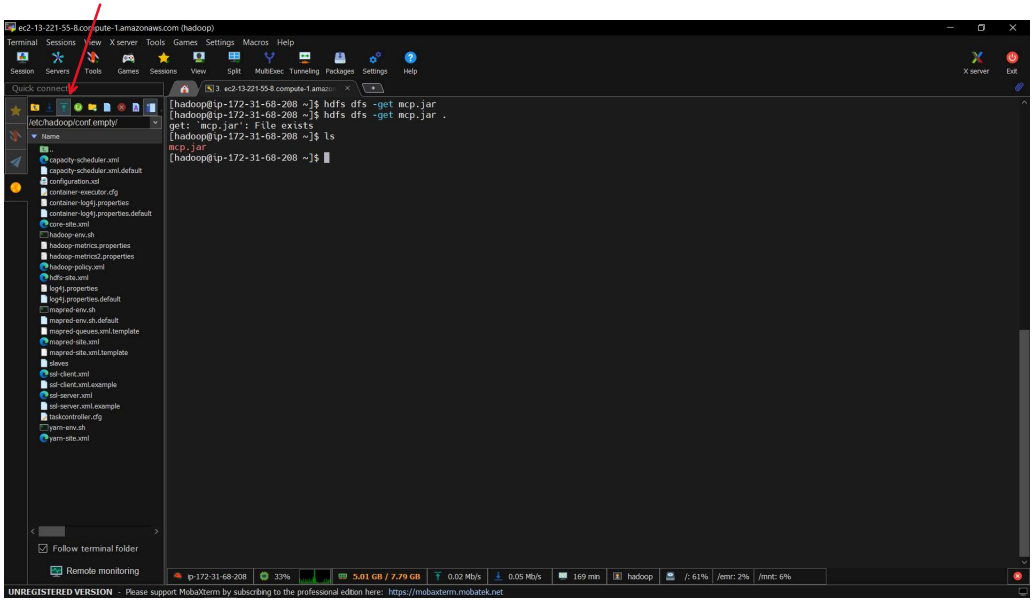
Columns are :
Stock exchange, stock, date, opening price, high, low, closing price, something, closing price

Ans :

1. Upload file NYSE_Data.txt to hadoop
   - Stocks/NYSE_Data.txt

2. MapReduce Program has to be uploaded into local file system of master machine. (name node) (means MobaXterm)
   - File is mcp.jar
   (one way is, upload the mcp.jar to hadoop using GUI and then use
   "hfds dfs -get mcp.jar ." <-- this command. After this type "ls" to see the file.

   OR we can just upload it using following button.



3. Run the jar file
   - yarn jar mcp.jar MaxClosePrice /user/hadoop/Stocks/NYSE_Data.txt /user/hadoop/output

4. We'll get output like this :

```
[hadoop@ip-172-31-68-208 conf]$ cd
[hadoop@ip-172-31-68-208 ~]$ clear
[hadoop@ip-172-31-68-208 ~]$ hdfs dfs -get mcp.jar
[hadoop@ip-172-31-68-208 ~]$ hdfs dfs -get mcp.jar .
get: `mcp.jar': File exists
[hadoop@ip-172-31-68-208 ~]$ ls
mcp.jar
[hadoop@ip-172-31-68-208 ~]$ yarn jar mcp.jar MaxClosePrice /user/hadoop/Stocks/NYSE_Data.txt /user/hadoop/output
25/11/18 09:31:23 INFO client.RMProxy: Connecting to ResourceManager at
ip-172-31-68-208.ec2.internal/172.31.68.208:8032
25/11/18 09:31:23 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the
Tool interface and execute your application with ToolRunner to remedy this.
```

```
25/11/18 09:31:24 INFO input.FileInputFormat: Total input files to process : 1
25/11/18 09:31:24 INFO lzo.GPLNativeCodeLoader: Loaded native gpl library
25/11/18 09:31:24 INFO lzo.LzoCodec: Successfully loaded & initialized native-lzo library [hadoop-lzo rev
ff8f5709577defb6b78cdc1f98cfe129c4b6fe46]
25/11/18 09:31:24 INFO mapreduce.JobSubmitter: number of splits:1
25/11/18 09:31:24 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1763448497480_0001
25/11/18 09:31:25 INFO impl.YarnClientImpl: Submitted application application_1763448497480_0001
25/11/18 09:31:25 INFO mapreduce.Job: The url to track the job:
http://ip-172-31-68-208.ec2.internal:20888/proxy/application_1763448497480_0001/
25/11/18 09:31:25 INFO mapreduce.Job: Running job: job_1763448497480_0001
25/11/18 09:31:36 INFO mapreduce.Job: Job job_1763448497480_0001 running in uber mode : false
25/11/18 09:31:36 INFO mapreduce.Job:  map 0% reduce 0%
25/11/18 09:31:45 INFO mapreduce.Job:  map 100% reduce 0%
25/11/18 09:31:52 INFO mapreduce.Job:  map 100% reduce 20%
25/11/18 09:31:56 INFO mapreduce.Job:  map 100% reduce 40%
25/11/18 09:31:57 INFO mapreduce.Job:  map 100% reduce 100%
25/11/18 09:31:57 INFO mapreduce.Job: Job job_1763448497480_0001 completed successfully
25/11/18 09:31:57 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=2210341
                FILE: Number of bytes written=5439283
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=40991000
                HDFS: Number of bytes written=2009
                HDFS: Number of read operations=18
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=10
        Job Counters
                Killed reduce tasks=1
                Launched map tasks=1
                Launched reduce tasks=5
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=10616832
                Total time spent by all reduces in occupied slots (ms)=119620608
                Total time spent by all map tasks (ms)=6912
                Total time spent by all reduce tasks (ms)=38939
                Total vcore-milliseconds taken by all map tasks=6912
                Total vcore-milliseconds taken by all reduce tasks=38939
                Total megabyte-milliseconds taken by all map tasks=10616832
                Total megabyte-milliseconds taken by all reduce tasks=119620608
        Map-Reduce Framework
                Map input records=735026
                Map output records=735026
                Map output bytes=5841483
                Map output materialized bytes=2210321
                Input split bytes=139
                Combine input records=0
                Combine output records=0
                Reduce input groups=203
                Reduce shuffle bytes=2210321
                Reduce input records=735026
                Reduce output records=203
                Spilled Records=1470052
                Shuffled Maps =5
                Failed Shuffles=0
                Merged Map outputs=5
                GC time elapsed (ms)=831
                CPU time spent (ms)=11180
                Physical memory (bytes) snapshot=1743540224
                Virtual memory (bytes) snapshot=26407182336
                Total committed heap usage (bytes)=1494745088
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=40990861
        File Output Format Counters
                Bytes Written=2009
[hadoop@ip-172-31-68-208 ~]$
```

If we run it again, it'll give exception :

```
[hadoop@ip-172-31-68-208 ~]$ yarn jar mcp.jar MaxClosePrice /user/hadoop/Stocks/NYSE_Data.txt /user/hadoop/output
25/11/18 09:33:57 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-68-208.ec2.internal/172.31.68.208:8032
Exception in thread "main" org.apache.hadoop.mapred.FileAlreadyExistsException: Output directory
hdfs://ip-172-31-68-208.ec2.internal:8020/user/hadoop/output already exists
        at org.apache.hadoop.mapreduce.lib.output.FileOutputFormat.checkOutputSpecs(FileOutputFormat.java:146)
```

```
        at org.apache.hadoop.mapreduce.JobSubmitter.checkSpecs(JobSubmitter.java:268)
        at org.apache.hadoop.mapreduce.JobSubmitter.submitJobInternal(JobSubmitter.java:141)
        at org.apache.hadoop.mapreduce.Job$11.run(Job.java:1341)
        at org.apache.hadoop.mapreduce.Job$11.run(Job.java:1338)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:422)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupInformation.java:1844)
        at org.apache.hadoop.mapreduce.Job.submit(Job.java:1338)
        at org.apache.hadoop.mapreduce.Job.waitForCompletion(Job.java:1359)
        at MaxClosePrice.main(MaxClosePrice.java:45)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:239)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:153)
[hadoop@ip-172-31-68-208 ~]$
```

The output is stored into Output folder in Home directory.

- There must be 5 files. Because by default aws uses 5 reducers.

YARN Architecture

YARN (Yet Another Resource Negotiator)

Master Node (ResourceManager)
- Global resource scheduler
- Knows free vcores/RAM across cluster (via NodeManagers)
- Allocates containers

Slave Nodes (NodeManager)
- Manages resources per node
- Reports status to RM
- Launches containers (JVMs)

Job Execution Flow in YARN

1. User submits a program (e.g. map-reduce job)
2. **ResourceManager** receives it
3. RM selects a NodeManager
4. NM launches **ApplicationMaster** (AM)

ApplicationMaster (AM)
- Runs inside a container
- One AM per application/job
- Responsibilities:
  - Check job details & resources
  - Request more containers from RM
  - Launch map/reduce tasks
  - Monitor tasks
  - Handle failures/retries
  - Report progress to user

Containers in YARN
- A container = CPU + Memory bundle
- Typically runs a JVM (but language can vary)
- Default (configurable):
  - **1 GB RAM**
  - **1 vCore**
- Containers run:
  - Mappers
  - Reducers
  - App Master

Data Locality

- ResourceManager **tries to run mappers where HDFS blocks exist**
- Reduces network IO
- Speeds processing

MapReduce Execution Summary

- Mappers process data in **parallel**
- Intermediate data shuffled to reducers
- Reducers complete computation

Where Output Goes

Reducers **write directly into HDFS**

Lifecycle

- App Master created per job
- Containers allocated
- Tasks executed
- After completion:
  - AM shuts down
  - Containers freed

For each job:

- RM → launches **ApplicationMaster**
- AM requests containers
- Mappers run on data nodes
- Output → Reducers → HDFS
- AM ends after completion