Hadoop

What is Hadoop?

Hadoop is an open-source framework developed by the Apache Software Foundation that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from a single server to thousands of machines, each offering local computation and storage. Hadoop is highly fault-tolerant and is designed to detect and handle failures at the application layer, delivering a highly available service on top of a cluster of computers.

What Can We Do with Hadoop?

Store Large Data Sets:

HDFS (Hadoop Distributed File System) allows you to store massive amounts of data across multiple machines. It is designed to handle large files (gigabytes to petabytes) and supports high-throughput access to data.

Process Large Data Sets:

MapReduce is Hadoop`s core processing model that allows you to process large data sets in parallel across a distributed cluster.

It breaks down tasks into smaller sub-tasks and processes them concurrently.

Data Analysis:

Hadoop supports various data analysis tools like Apache Hive for SQL-like querying and Apache Pig for scripting,

making it easier to analyze large volumes of data.

Data Warehousing:

Hadoop can be used as a data warehouse with tools like Hive and HBase,

which support structured and semi-structured data storage and querying.

Data Integration:

Tools like Apache Sqoop and Flume can be used to import data from relational databases

or streaming sources into Hadoop for further processing and analysis.

Machine Learning and Data Mining:

Hadoop can integrate with tools like Apache Mahout or Spark MLlib

to perform machine learning tasks on large data sets.

Summary

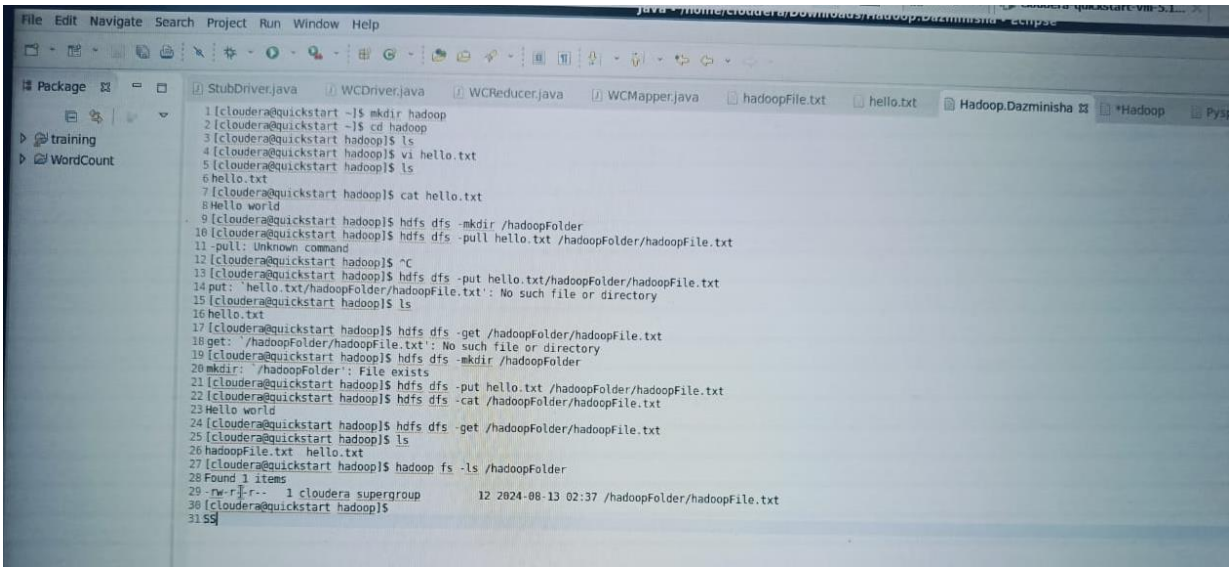
Hadoop is a powerful framework for distributed storage and processing of big data.

With Hadoop, you can store large volumes of data across multiple machines,

process it using parallel computation, analyze it using tools like Hive and Pig,

and integrate data from various sources. Hadoop commands allow you to interact with HDFS,

submit and monitor jobs, query data, and manage resources within a Hadoop cluster.



Package

training

WordCount

StubDriver.java | WCDriver.java | WCReducer.java | WCMapper.java | hadoopFile.txt | hello.txt | Hadoop.Dazminisha | *Hadoop 2

```
[cloudera@quickstart ~]$ ls
avro                      enterprise-deployment.json  Pictures
cloudera-manager          express-deployment.json     Public
cm_api.py                 external_jars               rakeshdata
codegen_categories.java   external-unified            rakeshdata1
data1                     hadoop                      sparkjars_exec
Dazminisha                input.txt                   Templates
Desktop                   kerberos                    Videos
devices.json              lib                         WordCount.jar
Documents                 Music                       workspace
Downloads                 parcels                     zeyo_tab.java
eclipse                   parquet_write
emp.java                  part_dir
[cloudera@quickstart ~]$ cat > /home/cloudera/WCFile.txt
TVS team
TVS team
Hadoop
Hadoop
Cloudera
Hadoop
Cloudera
Hadoop
Cloudera^Z
[1]+  Stopped                 cat > /home/cloudera/WCFile.txt
[cloudera@quickstart ~]$ hdfs dfs -mkdir / inputWC
mkdir: `/': File exists
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 7 items
drwxrwxrwx   - hdfs     supergroup          0 2017-10-23 10:29 /benchmarks
drwxr-xr-x   - cloudera supergroup          0 2024-08-13 02:37 /hadoopFolder
drwxr-xr-x   - hbase    supergroup          0 2024-08-13 01:11 /hbase
drwxr-xr-x   - solr     solr                0 2017-10-23 10:32 /solr
drwxrwxrwt   - hdfs     supergroup          0 2020-05-22 02:53 /tmp
drwxr-xr-x   - hdfs     supergroup          0 2020-05-22 02:53 /user
drwxr-xr-x   - hdfs     supergroup          0 2017-10-23 10:31 /var
[cloudera@quickstart ~]$ hdfs dfs -mkdir /inputWC
[cloudera@quickstart ~]$ hdfs dfs -ls /
Found 8 items
drwxrwxrwx   - hdfs     supergroup          0 2017-10-23 10:29 /benchmarks
drwxr-xr-x   - cloudera supergroup          0 2024-08-13 02:37 /hadoopFolder
drwxr-xr-x   - hbase    supergroup          0 2024-08-13 01:11 /hbase
drwxr-xr-x   - cloudera supergroup          0 2024-08-13 03:15 /inputWC
drwxr-xr-x   - solr     solr                0 2017-10-23 10:32 /solr
drwxrwxrwt   - hdfs     supergroup          0 2020-05-22 02:53 /tmp
drwxr-xr-x   - hdfs     supergroup          0 2020-05-22 02:53 /user
drwxr-xr-x   - hdfs     supergroup          0 2017-10-23 10:31 /var
[cloudera@quickstart ~]$ hdfs dfs -put /home/cloudera/WCFile.txt /inputWC/
[cloudera@quickstart ~]$ hdfs dfs -cat /inputWC/WCFile.txt
TVS team
TVS team
Hadoop
Hadoop
Cloudera
Hadoop
Cloudera
Hadoop
```

Cloudera Live : Welco...

File Edit Navigate Search Project Run Window Help

Package | StubDriver.java | WCDriver.java | WCReducer.java | WCMapper.java | hadoopFile.txt | hello.txt | Hadoop.Dazminisha | *Hadoop | Pyspark

training
WordCount

```
Cloudera
Hadoop
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/WordCount.jar WCDriver /inputWC/WCFile.txt /outWC
24/08/13 03:23:25 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
24/08/13 03:23:26 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
24/08/13 03:23:26 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this
24/08/13 03:23:26 INFO mapred.FileInputFormat: Total input paths to process : 1
24/08/13 03:23:26 WARN hdfs.DFSClient: Caught exception
java.lang.InterruptedException
    at java.lang.Object.wait(Native Method)
    at java.lang.Thread.join(Thread.java:1252)
    at java.lang.Thread.join(Thread.java:1326)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.closeResponder(DFSOutputStream.java:967)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.endBlock(DFSOutputStream.java:705)
    at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.java:894)
24/08/13 03:23:26 INFO mapreduce.JobSubmitter: number of splits:2
24/08/13 03:23:26 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1723536650149_0001
24/08/13 03:23:27 INFO impl.YarnClientImpl: Submitted application application_1723536650149_0001
24/08/13 03:23:27 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1723536650149_0001/
24/08/13 03:23:27 INFO mapreduce.Job: Running job: job_1723536650149_0001
24/08/13 03:23:36 INFO mapreduce.Job: Job job_1723536650149_0001 running in uber mode : false
24/08/13 03:23:36 INFO mapreduce.Job:  map 0% reduce 0%
24/08/13 03:23:45 INFO mapreduce.Job:  map 50% reduce 0%
24/08/13 03:23:46 INFO mapreduce.Job:  map 100% reduce 0%
24/08/13 03:23:50 INFO mapreduce.Job:  map 100% reduce 100%
24/08/13 03:23:51 INFO mapreduce.Job: Job job_1723536650149_0001 completed successfully
24/08/13 03:23:51 INFO mapreduce.Job: Counters: 49
    File System Counters
        FILE: Number of bytes read=130
        FILE: Number of bytes written=430807
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=300
        HDFS: Number of bytes written=33
        HDFS: Number of read operations=9
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Launched map tasks=2
        Launched reduce tasks=1
        Data-local map tasks=2
        Total time spent by all maps in occupied slots (ms)=15310
        Total time spent by all reduces in occupied slots (ms)=3055
        Total time spent by all map tasks (ms)=15310
        Total time spent by all reduce tasks (ms)=3055
        Total vcore-milliseconds taken by all map tasks=15310
        Total vcore-milliseconds taken by all reduce tasks=3055
        Total megabyte-milliseconds taken by all map tasks=15677440
        Total megabyte-milliseconds taken by all reduce tasks=3128320
    Map-Reduce Framework
        Map input records=8
        Map output records=10
        Map output bytes=104
        Map output materialized bytes=136
        Input split bytes=204
```

Writable | Insert | 59 : 42

Cloudera Live : Welco... | java - /home/cloudera/...

Edit Navigate Search Project Run Window Help

StubDriver.java | WCDriver.java | WCReducer.java | WCMapper.java | hadoopFile.txt | hello.txt | Hadoop.Dazminisha | *Hadoop | Pyspark

```
        HDFS: Number of bytes read=300
        HDFS: Number of bytes written=33
        HDFS: Number of read operations=9
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=2
    Job Counters
        Launched map tasks=2
        Launched reduce tasks=1
        Data-local map tasks=2
        Total time spent by all maps in occupied slots (ms)=15310
        Total time spent by all reduces in occupied slots (ms)=3055
        Total time spent by all map tasks (ms)=15310
        Total time spent by all reduce tasks (ms)=3055
        Total vcore-milliseconds taken by all map tasks=15310
        Total vcore-milliseconds taken by all reduce tasks=3055
        Total megabyte-milliseconds taken by all map tasks=15677440
        Total megabyte-milliseconds taken by all reduce tasks=3128320
    Map-Reduce Framework
        Map input records=8
        Map output records=10
        Map output bytes=104
        Map output materialized bytes=136
        Input split bytes=204
        Combine input records=0
        Combine output records=0
        Reduce input groups=4
        Reduce shuffle bytes=136
        Reduce input records=10
        Reduce output records=4
        Spilled Records=20
        Shuffled Maps =2
        Failed Shuffles=0
        Merged Map outputs=2
        GC time elapsed (ms)=312
        CPU time spent (ms)=2010
        Physical memory (bytes) snapshot=647938048
        Virtual memory (bytes) snapshot=8249905152
        Total committed heap usage (bytes)=587014144
    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=96
    File Output Format Counters
        Bytes Written=33
0
[cloudera@quickstart ~]$ hdfs dfs -ls /outWC
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2024-08-13 03:23 /outWC/_SUCCESS
-rw-r--r--   1 cloudera supergroup         33 2024-08-13 03:23 /outWC/part-00000
[cloudera@quickstart ~]$
```

Writable

Cloudera Live : Welco... | java - /home/cloudera/...

Hadoop commands

Ls command to list the files/folders

ls

mkdir Hadoop: to create a folder

mkdir

cd : to change the directory

cd Hadoop/

Now create a file inside this hadoop folder as below:

Vi command to create a new file.

Press `i` to go on edit mode in this file and type any content as per your wish.

Once text is typed, press `escape` and then :wq

esc and ":wq"

To see the content of the file do as below: use cat command

cat hi.txt

The above are all simple commands.

Now let`s create a folder on our HDFS system (Hadoop file system) and learn the new commands:

hdfs dfs is Hadoop specific commands. (Hadoop Distributed File system)

hdfs dfs -mkdir /hadoopFolder

hdfs dfs -mkdir /newFolder

Lets copy the local file hello.txt onto the HDFS system as below:

hdfs dfs -put hello.txt /hadoopFolder/hadoopFile.txt

here we are copying the data of hello.txt onto a new file on HDFS system called hadoopFile.txt

put: command to copy a file from local machine to HDFS system.

hdfs dfs -put hello.txt/hadoopFolder/hadoopFile.txt

This way we were able to move a local file of our computer onto the HDFS system.

open hdfs file

hdfs dfs -cat /hadoopFolder/hadoopFile.txt

get command: to move the file from HDFS to local system.

hdfs dfs -get /hadoopFolder/hadoopFile.txt

The below commands are similar to that of put command which copies the local file into hdfs system.

hadoop fs -copyFromLocal hello.txt /hadoopFolder

hadoop fs -ls /hadoopFolder

HADOOP MAPREDUCE example using a WordCount program:

We need to now create a simple text file for Word counting purpose:

Create a WCFile.txt using cat command as below.

Once done with entering the text , press ctrl-z to exit.

---

cat > /home/cloudera/WCFile.txt

hdfs dfs -mkdir /inputWC

hdfs dfs -ls /

Now lets create a folder in the hdfs system to store this WCFile.txt

Named the folder as inputWC

---

hdfs dfs -put /home/cloudera/WCFile.txt /inputWC/

Lets put this WCFile.txt onto the HDFS system:

Use the put command as below and also cross check the data using cat command

---

hdfs dfs -cat /inputWC/WCFile.txt

Now lets execute the jar file for wordCounting

---

hadoop jar /home/cloudera/WordCount.jar WCDriver /inputWC/WCFile.txt /outWC

The command to execute the jar file

hadoop jar /home/cloudera/WordCount.jar WCDriver /inputWC/WCFile.txt /outWC

Explaining the above command:

Hadoop jar -> this is a command to execute the jar file

/home/cloudera/WordCount.jar -> this is the location where our jar is located. WordCount.jar is the one which had exported from eclipse.

WCDriver -> This is a main java class which will be executed. [Refer the below code in eclipse]

/inputWC/WCFile.txt -> this is the file on hdfs which has the data to be counted.

/outWC -> this folder gets created once the below command is processed and stores the output.

To check the output execute as below

---

hdfs dfs -ls /outWC