Sort on Hadoop/Spark

Author:
Aditya Kumar

Advisor:
Ioan Raicu

Abstract

The aim of this project is to conduct Sort benchmarks on Hadoop and Spark. The benchmarks have been written from scratch and have been tested on Amazon AWS Cloud EC2 on-spot c3.large instance.

Table Of Contents

1. Introduction

The goal of this programming assignment is to enable us to gain experience programming with:

1. Amazon Web Services, specifically the EC2 cloud (http://aws.amazon.com/ec2/)
2. The Hadoop framework (http://hadoop.apache.org/)
3. The Spark framework (http://spark.apache.org/)

1.1. Amazon Web Services

Amazon Web Services (AWS), is a collection of cloud computing services that make up the on-demand computing platform offered by Amazon.com. These services operate from 12 geographical regions across the world. The most central and best-known of these services arguably include Amazon Elastic Compute Cloud, also known as "EC2", and Amazon Simple Storage Service, also known as "S3".

1.2. Apache Hadoop

Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware. Hadoop is an Apache top-level project being built and used by a global community of contributors and users. It is licensed under the Apache License 2.0.

1.3. Apache Spark

Apache Spark is a fast and general processing engine compatible with Hadoop data. It can run in Hadoop clusters through YARN or Spark's standalone mode, and it can process data in HDFS, HBase, Cassandra, Hive, and any Hadoop InputFormat. It is designed to perform both batch processing (similar to MapReduce) and new workloads like streaming, interactive queries, and machine learning.

2. Runtime Environment Settings

- AMI- Amazon Linux AMI 2016.03.0 (HVM), SSD Volume Type
- Instance- c3.large
- vCPUs- 2
- Memory- 3.7 GB
- Storage- 500GB
- Java Version- Java-1.7.0
- Hadoop Version- Hadoop-2.7.2
- Spark Version- Spark- 1.6.0

3. Configuration

3.1. Set up a one node Virtual Cluster

I set up a virtual cluster of 1 node using the Amazon EC2 Dashboard in few minutes by making a spot request of a c3.large instance using the AMI ami-08111162. Once the instance started running, i transferred all the files to the instance using scp connection and unzipped hadoop and spark setup files. I then connected using ssh and modified the configuration files:

1. core-site.xml
2. hadoop-env.sh
3. hdfs-site.xml
4. mapred-site.xml
5. yarn-site.xml
6. slaves

3.2. Set up a 16 node Virtual Cluster

I set up a virtual cluster of 1 node using the Amazon EC2 Dashboard in few minutes by making a spot request of a c3.large instance using the AMI ami-08111162. After setting up the following config files i took an AMI and then started the other 16 nodes i.e. the slaves.

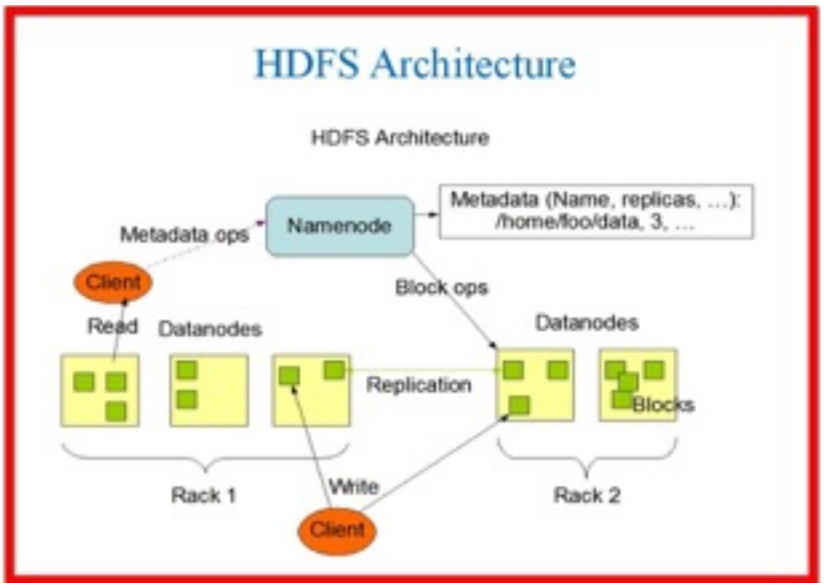
1. core-site.xml
2. hadoop-env.sh
3. hdfs-site.xml
4. mapred-site.xml
5. yarn-site.xml
6. slaves
7. master

Once the instance started running, i transferred all the files to the master using scp connection and unzipped hadoop and spark setup files.

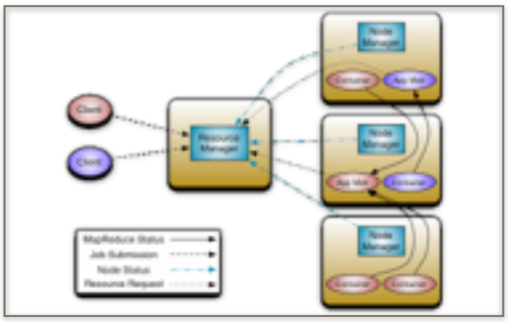
4. Hadoop Implementation

As mentioned earlier, Apache Hadoop is an open source framework for cloud computing, and more precisely for storage and large scale processing on commodity clusters. Hadoop is an alternative to the Google Mapreduce framework. Hadoop allows to distribute Map reduce jobs on several workers nodes which are handled by one or more master node. Hadoop is made of the following components:

1. HDFS is the Hadoop distributed file system that is responsible for the storage of data. It is composed by a NameNode on the master which holds all the metadata regarding the stored files and manages the file system namespace and regulates access to files by clients and DataNodes on the workers which contain actual data and manage storage attached to the nodes that they run on.



2. YARN is Apache Hadoop NextGen MapReduce that is responsible for resource management and job scheduling/monitoring. It is composed by a ResourceManager on the master which is responsible for allocating resources to the various running applications and accepting job-submissions. On the other side , the NodeManager is the per-machine framework agent who is responsible for containers and monitoring resource usage.



Installation:

The assignment requires to setup a cluster of 17 nodes with hadoop installed on it. And it required to separates setups for the master node and the workers. The setup can be done for one worker and then replicated using an AMI.

First we install the Master. We need to install a Java virtual machine on all the node, which is used to run hadoop. We chose the Java-7-openjdk version. Then we chose to install Hadoop v2.7.2 which is fairly recent and using YARN rather than former MapReduce on legacy versions of Hadoop. Then we need to configure the environment variables for hadoop.

We also need to setup ssh connexion between the master and the workers so that they can connect automatically without password. To do so we use a RSA key authentication in which the master provides its public key to all the workers. This will allow hadoop scripts to launch automatically the HDFS and YARN daemons on all the nodes.

Then we need to setup the HDFS configuration. There are two configuration files which are called hdfs-site.xml and core-site.xml. The first one specifies the path to the local filesystem for both the NameNode and the DataNodes. The second one holds the domain name of the NameNode so that the hadoop modules know where the NameNode is.

Second we configure YARN using the yarn-site.xml configuration file. In this file we setup the resource allocation of hardware according to our machines. As we will run our HadoopSort on c3 large instances, we chose a maximum memory for each scheduler to 2048 MB, 1 virtual cores per container, 2 virtual cores per scheduler and 4096MB of memory for the NodeManager.

Finally we need to format the namenode to the hadoop file system. Once it's done we can start the HDFS and YARN daemons on the master. Assuming the Workers are already configured, we can add their domain name to the slave's file of the master. It will allow hadoop to run all the daemons on all the node at the same time.

We need now to configure the workers. As for the master we install Java and download the same version of hadoop. We also setup the environment variables as previously. Then we configure the hdfs-site.xml file by setting the domain name of the machine running the NameNode daemon ie. the master. To configure YARN we modify the yarn-site.xml by adding the domain name of the ResourceManager ie. the master.

Mapper:

The mapper class extracts the line from the input data set and splits the line into the bytes which are to be sorted and the bytes which are not to be sorted. That is, the first 10 bytes are the "sortBytes" and the rest of the bytes are the "restbytes".

Reducer:

The reducer class just prints the sorted data in an output file.

Partitoner:

It manages the partitioning of the keys that are part of the intermediate map output. Here the number of reducers comes into existence. The intermediate keys are sent to one of the reducers and to which reducer it goes is decided in the partitioner.

Sort Comparator:

It is implemented to decided the sort order of the key to be sorted.

Grouping Comparator:

It is implemented to group the map output keys into a single key.

Configuration Files

Once i implemented the hadoop sort, we need to compile it with the hadoop-commons library and then create a jar file. But the cluster has to be configured a different way according the number of slave on which we intent to run the job. Thus the conf/master is supposed to specify the master's address. The conf/slaves specifies the slaves addresses so that the master can automatically connect with the workers. the conf/core-site file specifies the NameNode address. The conf/hdfs-site specifies the path on the local filesystem of a DataNode where it should store its blocks, and Path on the local filesystem where the NameNode stores the metadata. Finally the conf/mapred-site specifies parameters such as the number of core required for each task, the memory allocated for map and reduce tasks or the heap size for jvm childs.
All of the configuration files must include port numbers when it refers to the master domain name URI because these ports are used by Hadoop’s daemons to communicate amongst themselves (to schedule jobs, replicate blocks, etc.). So if those ports are not fixed two daemons may use the same port leading to the fact that one daemon will be blocked. We

can change the numbers of mapers and reducers from the configuration files by adding cores and memory to containers.

Answers

1) What is a Master node? What is a Slaves node?

Answer:

Master node:

The master node in hadoop cluster controls and hosts the management services and the storage services such as the Name Node, Job tracker, Resource Manager, Checkpoint node, and secondary name node. It manages the whole process that is to be followed.

Slave node:

Slave nodes actually perform the work that is to be done. These are controlled by the master node. The data is saved in the slave nodes and the process also takes place in the slave node.

2) Why do we need to set unique available ports to those configuration files on a shared environment? What errors or side-effects will show if we use same port number for each user?

Answer: We need to set unique ports for the configuration files in a shared environment because each port has some specific significance. For example in the core-site.xml file the port number is set for the hadoop instance.

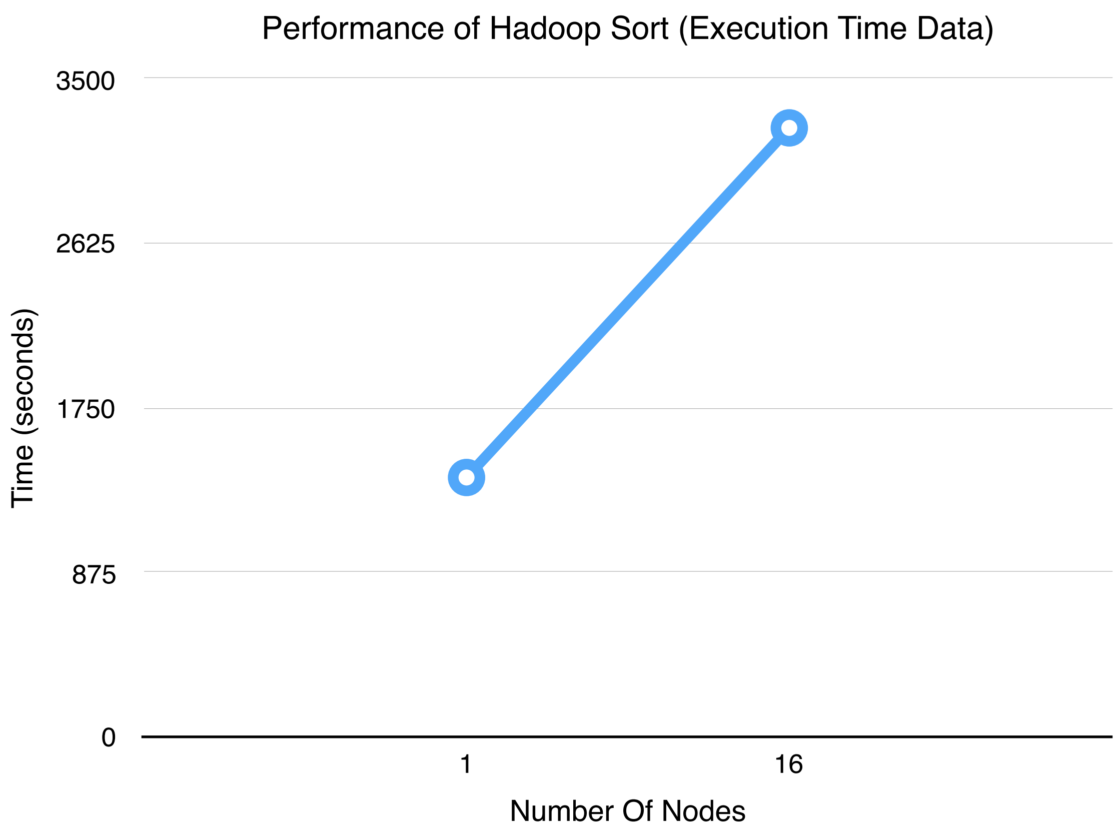
If a same port is assigned to different users then there will be a confusion between the function the ports perform. Here, a task can be assigned to a port which was not supposed to perform it.

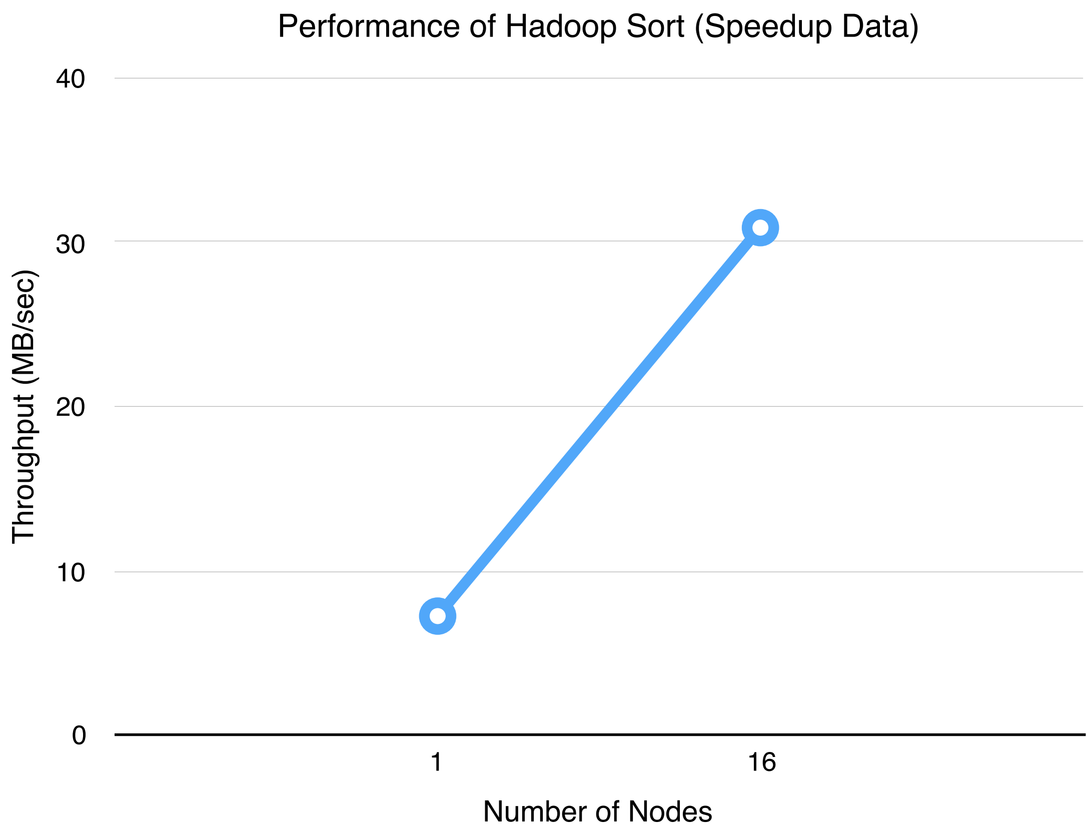
3) How can we change the number of mappers and reducers from the configuration file?

Answer: The number of mappers and reducers can be changed in the configuration file "mapred-site_tempalate.xml". It can be changed in the following way:

```
<name>mapred.tasktracker.map.tasks.maximum</name>
    <value>3</value>
<name>mapred.tasktracker.reduce.tasks.maximum</name>
    <value>4</value>
```

Performance:





5. Spark Implementation

Installation:

The assignment requires to setup a cluster of 17 nodes with spark installed on it.

Installing Spark on a one node cluster is relatively easy as an installation script can start it:

```
./spark-ec2 -k spark -i spark.pem launch my-spark-cluster -s 1 -t
c3.large -r us-east-1 --spot-price=0.18
```

To start the spark instance we do the following:

- Create KeyPair "spark.pem"
- Go to Security Credentials
- Create new access key
- Download access key ID and secret ID
- export AWS_ACCESS_KEY_ID = ""
- export AWS_SECRET_ACESS_KEY= ""

To terminate:

```
./spark-ec2 —region=us-east-1 destroy my-spark-cluster
```

To run:

```
./bin/spark-submit \
  --class SparkSort \
  --master spark://52.90.212.206:7077 \
  --executor-memory 20G \
  --total-executor-cores 100 \
  /path/to/sparksort.jar \
  1000
```

Problems:

I wasn't able to get the spark instance in the beginning, after that the connection took a lot of time and the deadline was crossed so, submitted without the output. Also the spark framework took time in understanding.

6. References:

[1] https://hadoop.apache.org/docs/r2.7.1/api/org/apache/hadoop/examples/Sort/package-summary.html

[2] http://ant.apache.org/bindownload.cgi

[3] http://www.oracle.com/technetwork/java/javase/downloads/index.html

[4] http://hadoop.apache.org/docs/current1/mapred_tutorial.html

[5] http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html

[6] http://spark.apache.org/downloads.html

[7] http://spark.apache.org/docs/latest/cluster-overview.html

[8] http://www.ordinal.com/gensort.html

[9] http://sortbenchmark.org

[10] http://sortbenchmark.org/2014_06_CloudSort_v_0_4.pdf