**Hadoop Tutorial**

Good education video: <https://www.youtube.com/watch?v=mafw2-CVYnA>

**Apache Hadoop** – Big Data Framework – Parallel /Distributed data storage

Due to the increasing volume of data needed for analytics / models and customer needs, traditional data storage/access cannot keep up. Hadoop solution provides a distributed (Hadoop cluster - data locality) and parallel solution (map) into smaller tasks all controlled by process called MapReduce. No more bottlenecks to a single data source or single data access point, Hadoop is scalable.

**HDFS** – Hadoop Distributed File System

NameNode – Master node – manages all datanodes via heartbeats

DataNode – Slave node –where actual data is stored in data blocks

* All data stored in 128MB Datablocks (default size)
* So, if a 380MB file is to be stored, HDFS breaks the file into 3 datablocks and distributed across all DataNodes in the Hadoop cluster

Secondary NameNode – metadata

3 advantages of using distributed data solution:

* Can add additional DataNodes as needed to meet the data needs of the cluster.
* Distributed data allows for parallel processing time, so leverages the processing power of each node. So if it takes 4 sec to retrieve 1 TB of data, and that data is spread across 3 DataNodes, you can retrieve it 3 times faster.
* Replicated data (each datablock is replicated (3 times by default) across the DataNodes

**Write Mechanism** – Pipeline Setup

Let’s say you have the CUI Model results you want to store on Hadoop:

* You would use an HDFS Client (Client node – data access like service) and perform a Write Request to the NameNode where you want to store your data.
* The NameNode will return the IP Address (location) of the DataNodes you will need (i.e. 3 DataNodes – DN1, DN4, DN6).
* The ClientNode then “asks” DN1 if it is ready to store the datablocks. DN1 then asks DN4, who asks DN6. When they are all ready the “Pipeline” is established.
* ClientNode then sends datablock to DN1. DN1 then sends a copy to DN4, and DN4 sends copy to DN6. This replicates the data to the 3 datanodes identified by the NameNode.
* Acknowledge process takes place starting with DN6 to DN4 indicating success. DN4 then acknowledges to DN1, which returns to ClientNode by DN1 and the NameNode is notified.
* The writing of multiple blocks are occurring at the same time.

**Read Mechanism**

Let’s say you want to read Blocks A & B from Hadoop:

* Use an HDFS ClientNode to perform a Read Request to a NameNode your data exists on.
* The NameNode will return the IP Address of the DataNodes where the data exists.
* ClientNode will fetch all the DataBlocks simultaneously from each DataNode and returned to the ClientNode.

**MapReduce**

Consists of two main tasks, Map and Reduce.

Map returns key/value pairs where the MapReduce with Reduce all of those parallel processed Maps into the aggregated output requested.

*Example MapReduce process*

Input Splitting Mapping Shuffling Reducing Result

2015, (1, 1)

Ford, (1, 1)

F150, (1, 1)

VehModel, F150 AMM Data

2015 Ford F150

John Doe

CUI Score

Company Code

2009, (1)

Chevy, (1)

Malibu, (1)

Data

2015 Ford F150

John Doe

CUI Score

Company Code

2015, (1)

Ford, (1)

F150, (1)

VehModel, F150 AMM Data

2015 Ford F150

John Doe

CUI Score

Company Code

2009, (1)

Chevy, (1)

Malibu, (1)

Data

2015 Ford F150

John Doe

CUI Score

Company Code

2015 Ford F150

2009 Chevy Malibu

2015 Ford F150

John Doe

CUI Score

Company Code

2015, 1

Ford, 1

F150, 1

Data

2015 Ford F150

John Doe

CUI Score

Company Code

*cTOM or AMM Data*

2015 Ford F150

2009 Chevy Malibu

2015 Ford F150

AMM Data

2015 Ford F150

John Doe

CUI Score

Company Code

ModelYear, 2015

VehMake, Ford

VehModel, F150 AMM Data

2015 Ford F150

John Doe

CUI Score

Company Code

ModelYear, 2009

VehMake, Chevy

VehModel, Malibu AMM Data

2015 Ford F150

John Doe

CUI Score

Company Code

2015, 1

Ford, 1

F150, 1

VehModel, F150 AMM Data

2015 Ford F150

John Doe

CUI Score

Company Code

2009, 1

Chevy, 1

Malibu, 1

Data

2015 Ford F150

John Doe

CUI Score

Company Code

ModelYear, 2015

VehMake, Ford

VehModel, F150 AMM Data

2015 Ford F150

John Doe

CUI Score

Company Code

Developer Steps:

1. Write the Mapper Code. This is the logic indicating how the Map task will process the data to produce the key-value pair to be aggregated.

Java Code (example):

Public static class Map extends Mapper<LongWritable,Text,Text,IntWriteable> {

public void map(LongWritable key, Text value, Context context) throws IO Exception {

String line = value.toString();

StringTokenizer tokenizer = new StringTokenizer(line);

While (tokenizer.hasMoreTokens()) {

Value.set(tokenizer.nextToken());

context.write(value, new IntWritable(1));

}

1. Next write the Reducer Code. The logic combines the intermediate key-value pairs generated by Mapper to give a final aggregated output.

Java Code (example):

public static class Reduce extends Reducer<Text,IntWritable,Text,IntWriteable> {

public void reduce(Text key, Iterable<IntWritable> values,Context context) throws IO Exception {

int sum=0;

for(IntWritable x: values)

{

sum+=x.get();

}

context.write(key), new IntWritable(sum)));

}

1. Write the Driver Code. This is the configurations details of map reduce job.

Example:

Configuration conf = new Configuration();

Job job = new Job(conf, “Vehicle retrieval”);

job.setJarByClass(VehicleRetrieve.class);

job.setMapperClass(Map.class);

job.setReducerClass(Reduce.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

job.setInputFormatClass(TextInputFormat.class);

job.setOutputFormatClass(TextOutputFormat.class);

Path outputPath = new Path(args[1]));

Example Command Prompt (execute Hadoop MapReduce process):

Create input and output directories

Pull the file into input directory

Execute your Hadoop program (MapReduce)

$hadoop fs –put test.text /VehicleRetrieve/input

$hadoop jar vehicleRetrieval.jar in.SF.VehRet/input dir/output dir

$hadoop fs –ls /VehRet/output

$hadoop fs –cat/VehRet/output/\*0

**MapReduce Job Workflow – YARN Architecture**

ResourceManager

6.1 Start container

5. Allocate

Resources

4.1 Start container

2. Submit job

MR Task

YARN child

6.2 launch

task JVM

7. run

NodeManager

AppMaster JVM

NodeManager

JVM

1.Run job

JVM

RM Node

4.2 Launch

AppMaster

Client Node

3. Get app ID

YARN – Yet another resource negotiator. Performs the MapReduce work.

Hadoop Architecture: HDFS & YARN combined

Hadoop Cluster – Bunch of Slave Nodes (servers) controlled by a switch which are in turn controlled by a Core Switch.

Hadoop Ecosystem (Tooling)

Flume – Used to ingest unstructured and semi-structured data into Hadoop

Sqoop – Used to ingest structured data into Hadoop (most likely what we will be used with AMM)

Hive – Facebood SQL

PIG – Scripting – its own language (PIG latin). 10 lines of PIG code = 100 lines of MapReduce code.

\*MapReduce – can use Java

\*SPARK – Real time processing

Oozie – scheduling Hadoop jobs

Storm – managing streams of data in realtime

\*KAFKA – realtime data streaming tool

Solr Lucene – searching and indexing tool

Zookeeper – used to manage the Hadoop clusters