What is Big Data?

The definition of big data is data that contains greater variety, arriving in increasing volumes and with more velocity.

5 Vs of Big Data

Volume:

Huge quantity of data is generated

Velocity:

The data is generated at a high speed

Variety:

Different sources and types of data. Structures, unstructured and semi-structured.

Veracity:

Truthfulness or reliability of data

Value:

Worth of information

What is Big Data Analytics?

It is the process of collecting, processing, cleaning, and analyzing big data to identify problems, opportunities, and increase bottom lines.

Types of Big Data Analytics:

Diagnostic Analysis

Understanding why something is happening

Descriptive Analysis

Describe the current state of the business or industry

Predictive Analysis

Make predictions about the future based on past data

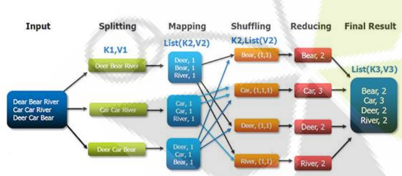
Prescriptive Analysis

Suggest solutions based on the result of descriptive or predictive analysis

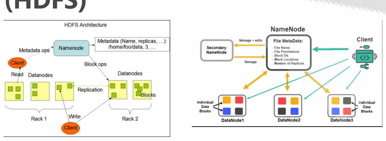
Features of Hadoop

Open Source
Network of Computers
Distributed Storage
Parallel Processing
Works on Commodity Hardware
Assumes Hardware will Fail

MapReduce



Hadoop Distributed File System (HDFS)



Stages of Big Data Analytics

Problem Definition:

Defining the goal, budget, and timeline of the project.

Data Definition:

Define the data requirements and internal and external sources for getting the data.

Data Acquisition:

Collect the data from these sources.

Data Extraction:

Extract the most useful features of the data collected.

Data Munging:

Converting the data into the best format.

Data Analysis:

Analyse the data to extract meaningful pattern from the data

Data Visualization:

Communicate the result of the analysis using visualizations

Utilization of the Result:

Utilize the result of the analysis to drive business decisions.

Components of Hadoop

Hadoop Commons

Hadoop Common refers to the collection of common utilities and libraries that support other Hadoop modules.

Hadoop Distributed File System (HDFS)

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware.

MapReduce

MapReduce is a programming paradigm that enables massive scalability across hundreds or thousands of servers in a Hadoop cluster.

Yet Another Resource Negotiator (YARN)

The fundamental idea of YARN is to split up the functionalities of resource management and job scheduling/monitoring into separate daemons.

Hadoop Ozone

Ozone is a scalable, redundant, and distributed object store for Hadoop and Cloud-native environments.

Hadoop Commands

version

Returns the version of Hadoop

Syntax: hadoop version

mkdir

To create a new directory on HDFS

Syntax: hdfs dfs -mkdir <path of directory>

ls

To list the contents of a directory on HDFS

Syntax: hdfs dfs -ls <path of directory>

put

To move content from the local file system to HDFS

Syntax: hdfs dfs -put <path of source> <path of destination>

copyFromLocal

To copy data from the local file system to HDFS

Syntax: hdfs dfs -copyFromLocal <path of source> <path of destination>

get

To move data from HDFS to the local file system

Syntax: hdfs dfs -get <path of source> <path of destination>

copyToLocal

To copy data from HDFS to the local file system

Syntax: hdfs dfs -copyToLocal <path of source> <path of destination>

cat

To view the contents of a file on HDFS

Syntax: hdfs dfs -cat <path to file>

mv

To move data from one location on HDFS to another.

Syntax: hdfs dfs -mv <path of source> <path of destination>

cp

To copy data from one location on HDFS to another

Syntax: hdfs dfs -cp <path of source> <path of destination>

rm

To delete data on HDFS

Syntax: hdfs dfs -rm -r <path to data>

jar

To execute MapReduce code written in a jar file

Syntax: hadoop jar <path of jar file> <name of class> <path of input file> <path of output file>

Apache Pig

Provides abstraction over MapReduce

Uses a scripting language called Pig Latin

Created by Yahoo Research in 2006

Moved to Apache in 2007

Components

Parser

Check the syntax of script and returns a Directed Acyclic Graph where the operators are represented as nodes and data flows are represented as edges

Optimizer

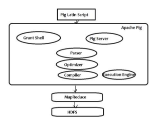
Carries out logical optimizations

Compiler

Compiles the optimized logical plan into a series of MapReduce jobs

Execution Engine

Executes the MapReduce jobs and stores the result in HDFS



Data Model

Scalar Types

Integers: 4-byte signed integers

Long: 8-byte signed integers

Float: Stores floating point numbers

Double: Double precision floating point numbers

Chararray: String or character array

Bytearray: Blob or array of bytes

Complex Types

Maps: Key value pairs [<key1>#<value1>, <key2>#<value2>,...]

Tuple: Collection of fields. (<field1>, <field2>, ...)

Bag: Collection of tuples. {<tuple1>, <tuple2>,...}

Note

Apache pig allows null values

If schema is available, Pig will use it for error checking and optimizations

If schema is not available, Pig will make the best guess based on how the script treats the data

Type casting syntax: (<datatype>)<variable name>

Commands

Fs

List all the files in HDFS

Syntax: fs -ls

Clear

Clear the interactive grunt shell

Syntax: clear

History

Syntax: history

Reading Data

<relation name> = LOAD <path> USING <function> AS <schema>;

Storing Data

STORE <relation name> INTO <path> USING <function>;

Dump

Display results on the screen for debugging

Syntax: dump <relation name>;

Describe

Helps the programmer to view the schema of the relation

Syntax: describe <relation name>;

Explain

Helps to review the logical, physical, and mapreduce execution plans

Syntax: explain <relation name>;

Illustrate

Gives step-by-step explanation of statements in Pig Execution

Syntax: illustrate <relation name>;

Group

Groups data based on the same key

Syntax: <grouped relation> = GROUP <relation name> BY <key>;

Join

Combine two or more relations

Syntax: <joined relation> = JOIN <relation1> BY <key1>, <relation2> by <key2>;

Cross

Calculates the cross product of two or more relations

Syntax: <result relation> = CROSS <relation1>, <relation2>;

Union

Merges two relations

Syntax: UNION <relation1>, <relation2>;

Filter

Helps in filtering out the tuples out of relation, based on certain conditions

<filtered relation> = FILTER <relation name> BY <condition>;

Distinct

Helps in removal of redundant tuples from the relation

Syntax: <distinct relation> = DISTINCT <relation name>;

Foreach

Helps in generating data transformations based on column data

Syntax: <result relation> = FOREACH <relation name> GENERATE <keys>;

Order by

Displays the result in a sorted order based on one or more fields.

Syntax: <result relation> = ORDER <relation> BY <key> [DESC];

Limit

Gets limited number of tuples from the relation

Syntax: <result relation> = LIMIT <relation> <value>;

Apache Hive

Hive is a Big Data Analytics tool that provides an SQL-like abstraction to MapReduce

Features of Hive

Works with structured data

Scalable and fast

Schema is stored in a database in Hive. Processed data is stored in HDFS.

SQL-inspired language - HiveQL

Uses a directory structure

Partitions and buckets enable fast retrieval

Components of Hive

Hive Server

Accepts requests from Thrift, JDBC, or ODBC client

Beeline

Java console-based utility for connecting with relational databases and executing SQL queries

Hive Driver

Receives HQL statements submitted by user, creates a session, and sends the query to the compiler

Hive Compiler

Performs semantic analysis and type checking and then generates a directed acyclic graph

Hive Optimizer

Performs transformations to improve the efficiency and scalability

Execution Engine

Execute the plan using Hadoop

Metastore

Central repository which stores the metadata in the structure of tables and partitions

HCatalog

Table and storage management

WebHCat

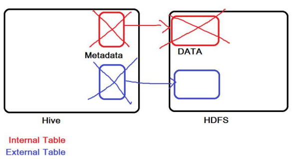
Rest API HCatalog

Data Model

Tables

Data is stored in HDFS

Metadata is stored in Hive



Partitions

Hive organizes tables into partitions for grouping the same type of data based on a column or partition key.

The queries are faster n slices of data

Buckets

Tables or partitions can be subdivided into buckets using functions of a column to give more structure to the data

It results in more efficient queries

Each bucket is a file in the table directory or partition directory.

Commands

Create

It is used to create database or table

Syntax: create database <database name>;

create table [external] <table name>(<schema>) row format delimited fields terminated by <delimiter> stored as textfile;

Show

Syntax: show databases;

show tables;

Alter

It is used to make changes to existing table

Describe

It describes the table columns

Syntax: describe <tablename>;

Truncate

Used to permanently truncate and delete the rows of the table

Syntax: truncate <tablename>;

Drop

Deletes the table data

Syntax: drop <tablename>;

Use

To select a database

Syntax: use <database name>;

Load:

Used to move data into corresponding Hive table.

Syntax: load data [local] inpath <file path> into table <table name>;

Insert:

Used to insert the data into Hive table

Insert Overwrite is used to overwrite the existing data in the table or partition

Insert Into is used to append the data into existing data in the table

Syntax: from <table name> insert [overwrite/into] table <table name> <query>;

Select:

Syntax: select <columns> from <table name>;

Grouping:

Group command is used to group the result set by one or more columns

Syntax: select <columns> from <table name> group by <column name>;

Join

Combines fields from two tables by using values common to each

Syntax: select <columns> from <table1> <alias1> join <columns> from <table2> <alias2> on <condition>;

The result of a left outer join (or simply left join) for tables A and B always contains all records of the "left" table (A), even if the join-condition does not find any matching record in the "right" table (B).

A right outer join (or right join) closely resembles a left outer join, except with the treatment of the tables reversed. Every row from the "right" table (B) will appear in the joined table at least once.

The joined table will contain all records from both tables, and fill in NULLs for missing matches on either side.

Quit

To exit the interactive hive shell

Syntax: quit;