2/6/2022

Takhti
Date
Page

Big Data Analytics.

(*) Big Data :-

* It is a collection of data that is huge in volume and growing exponentially with time.

* None of traditional data mangement tools can store it or process it efficiently.

(*) Big Data Analytis :-

It is the process of examining large dataset containg a variety of datatypes to uncover hidden pattern or unknown corellation for making trends, customer preference and other useful information.

(*) Challenges :-

Insufficient understanding and acceptance of big Data

* Confusion while big data tool's selection.

* Paying loads of money.

* Data integration.
* Data security.

* Some other challenges are :-
  - Capture
  - Storage
  - Analysis.

* Problems with Traditional large scale system :-

* Data has increased tremendously. So the traditional system find it challenging to handle such data.

* Majority of data comes in the form of semi-structured or unstructured data.

* Traditional large scale systems are designed to store only structured data.

* Big data is generated at a high velocity. Traditional systems lacks in high velocity because it is designed for steady data retention rather than rapid growth.

Takhti
Date
Page

* Data is so expensive to store in traditional system. Data is filtered and aggregated and large volume of data is are thrown out.

* Minimizing the data to be analysed reduces the accuracy of the results.

Sources of Big Data :-

- Social networking sites.
- E-commerce sites.
- Weather station
- Telecom company
- Share market.

3 V's of Big Data :-

- Velocity
- Variety
- Volume

* Velocity :- Refers to the speed at which data is being created in real time.

Takhti
Date
Page

* It determines the potential of data that how fast the data is generated and processed to meet the demands.

eg. → 3.5 billion searches per day are made on google.

↳ facebook user are increasing by 22 % ~~per~~ year by year.

* Variety :- It refers to the heterogenous sources and nature of data.

Structured, semi-structured and unstructured.

eg: Photos, Videos, Pdfs, emails, etc.

* Volumes :- It defines the huge amount of data.

eg: In year 2016, the estimated global traffic was 6.2 exabytes or 6.2 billion GB.

Takhti
Date
Page

Types of Big-Data :-

- Structured (Tabular data)
- Semi-structured (xml and log files).
- Unstructured. (Photos, videos, text, audio, etc)

Structured data :- Any data that can be stored, accessed and processed in form of fixed format is termed as structured data. (e.g. Tabular data).

Unstructured data :- This data refers to the data that lacks any specific form or structured.

* This makes it very difficult and time consuming to process and analyze unstructured data.

example :- text, video, audio and images.

Semi-structured data :- This data pertains to the data containg both the formats, that is structured and unstructured data.

Takhti
Date
Page

* To be precise, it refers to the data that although has not been classified under a particular repository (database), yet contains vital information or tags that segregates individual elements within the data.

* Require parser to access data.
eg. xml, log files.

Google File System (GFS) :-

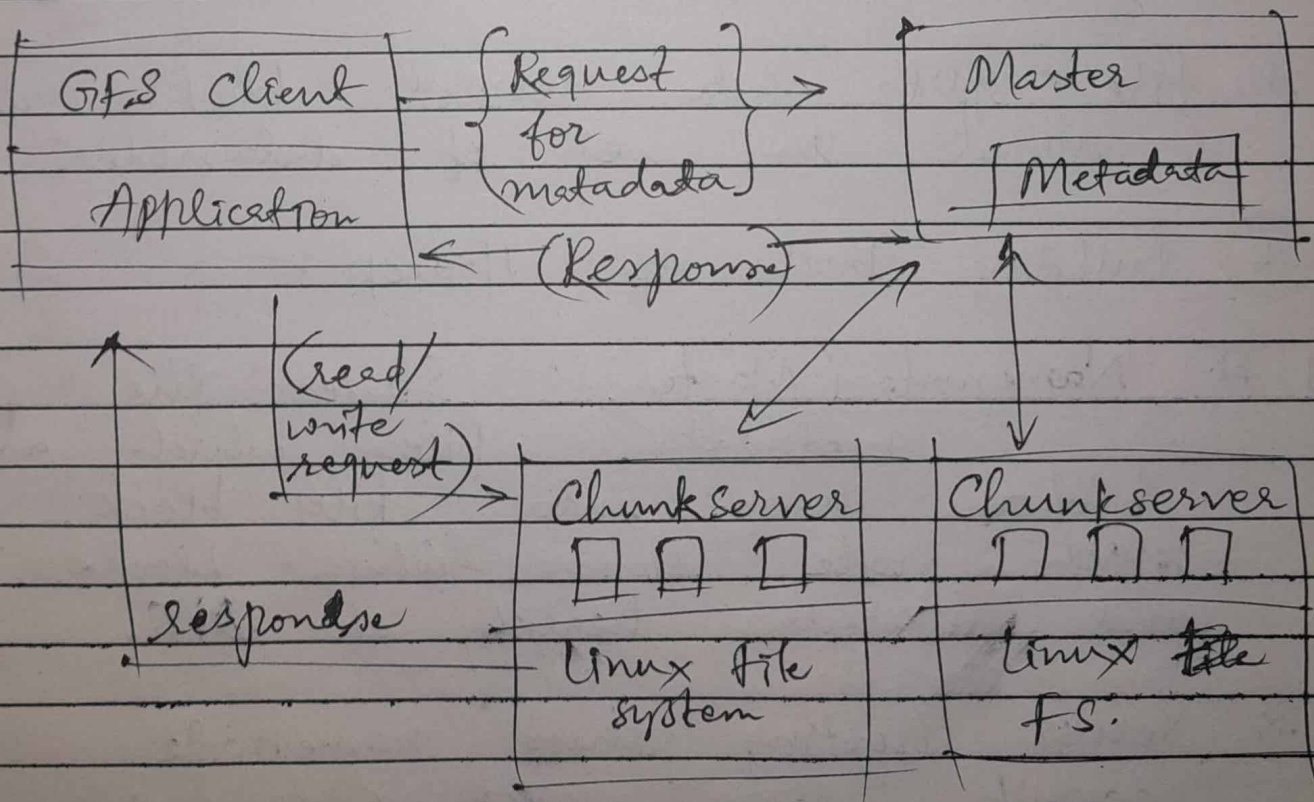
*→ GFS is a scalable distributed file system, and used for large distributed data (big data), and intensive applications.

* GFS provides fault tolerance.
* GFS runs on inexpensive hardware.
* Delivers high performance to a large number of clients.
* files are organised heirarchically in directories and identified by path names.

Takhti
Date
Page

* Supports usual operations such as create, delete, open, close, read, write on files.
* Snapshot operation (creates a copy of a file or a directory at low cost).
* Record operation :- allows multiple clients to append data to the same file concurrently.

(*) GFS Architecture :-



Takhti
Date
Page

(*) Hadoop Distributed file system (HDFS) :-

* based on GFS and provides a distributed file system.
* designed to run on large clusters (thousand of computers).
* master - slave architecture.
  - NameNode (contains metadata)
  - DataNodes. (contains actual data)
* file splits into several blocks and stored in set of datanodes.

(*) Building blocks of Hadoop :-

1. (*) Namenode (Master) :- Stores the metadata, files which are broken down into file block, which node stores which blocks, and overall health of DFS.
   * Server hosting these namenode doesn't store user data or performs any computations.

Takhti
Date
Page

* Negative impact of Namenode is that if it fails then the entire Hadoop clusters will fail.

(*) DataNode (Slave) :- Stores the (2) actual data.

* Client can communicates directly with the Datanode.

* It performs reading and writing HDFS blocks, to actual files from the local file system.

* Can communicate directly with other datanode ~~with~~ to replicate its data blocks for redundancy.

(*) Secondary NameNode (SNN) :- (3)

* Assistant that monitors the state of cluster.
* Each cluster has one SNN.
* Communicates with namenode to take snapshots of HDFS metadata at intervals defined by the cluster configuration.

Takhti
Date
Page

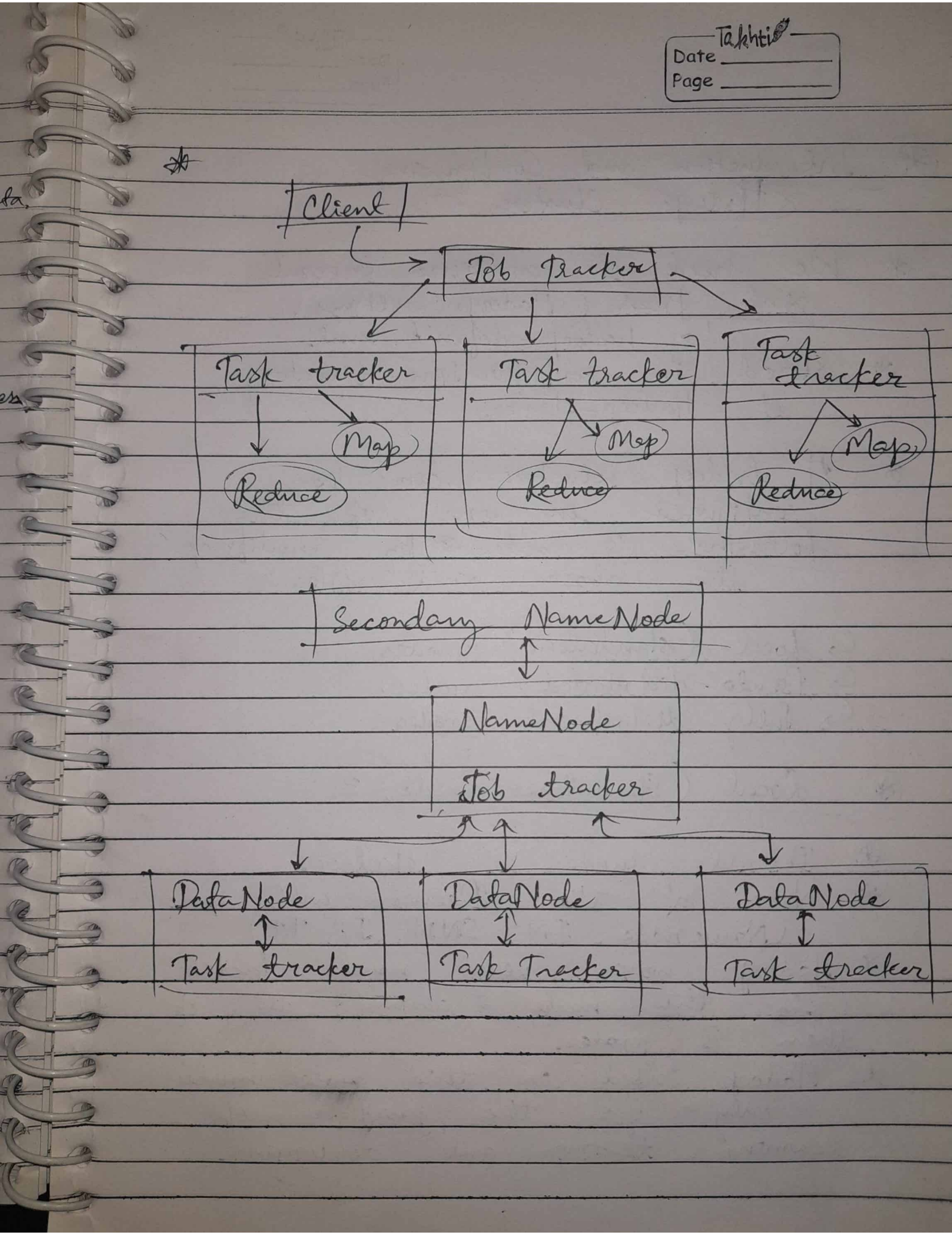
* Snapshots helps in minimizing the downtime and loss of data.

(*) (4) Job-Tracker = Link between your app and Hadoop.

* It determines the execution plans by determining which files to process.
* Assign nodes to different tasks
* and monitors all tasks as they are running.
* If a task fail, it will automatically relaunch the task.
* Every cluster has only one Job-tracker.

(*) (5) Task Tracker =

* Responsibility is to constantly (heartbeat) communicate with the Job tracker.
* If Jobtracker fails to receive a beat heartbeat from a task trackers within a specified amount of time, then it will assume the task tracker has crashed and it will resubmit the task to other node in the cluster.


Takhti
Date
Page
Client
Job tracker
Task tracker
Map
Reduce
Task tracker
Map
Reduce
Task tracker
Map
Reduce
Secondary NameNode
NameNode
Job tracker
DataNode
Task tracker
DataNode
Task Tracker
DataNode
Task tracker

Takhti
Date
Page

⊛ Introduction and Configuring Hadoop Cluster :-

* We need to configure several xml files (Hadoop settings).
  ↳ hadoop-default.xml.
* Stored in configuration directory of Hadoop_Home.

* A hadoop cluster can be configured in one of the following 3 modes by modifying the above xml files :-

  ↳ Local (standalone) modes.
  ↳ Pseudo-distributed modes.
  ↳ Fully distributed modes.

⊛ Local (Standalone mode) :-

* Default mode of Hadoop.
* None of the Daemon will run. (Namenode, DN, SNN, JT, TT)
* Hadoop works very much faster in this mode among all of them 3 modes.
* Hadoop used in this ~~node~~ mode only for the purpose of learning, testing and debugging.

Takhti
Date
Page

Pseudo - distributed mode (Single node cluster) :-

* Master and slave processes are handled by single system.
* All the processes inside cluster will run independently to each other.
* All daemon will run seperately on various JVM (Java Virtual machine).

* Used for ↳ development and debugging purpose.
  ↳ Managing the Input/output purpose.

~~(PDA). Introduction and configuring Hadoop cluster :-~~

Fully Distributed mode (Multinode cluster)

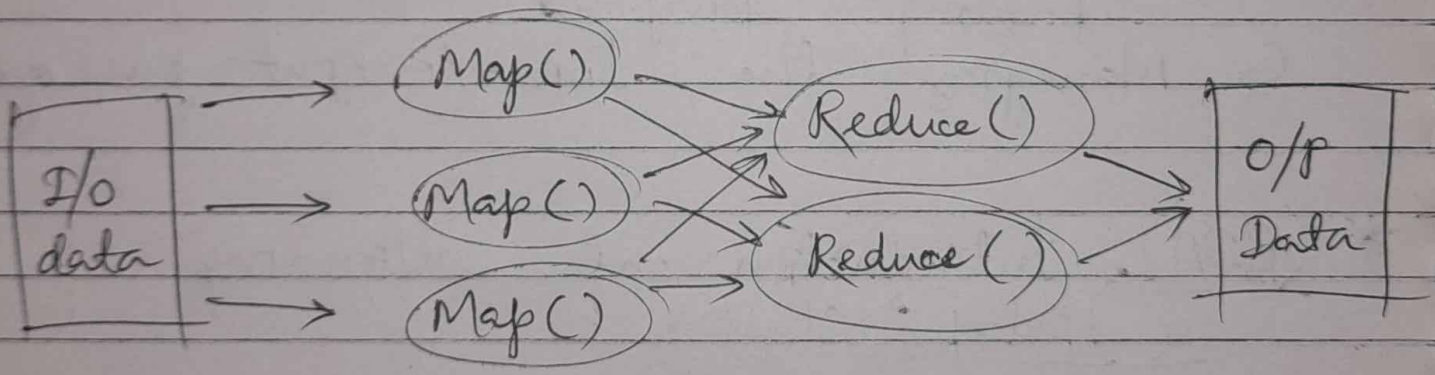
* An actual hadoop cluster runs in this mode. ~~Has multiple mode.~~
* It has multiple nodes. few of them run Master Daemon (NN) and the rest run slave daemon (DN).

Takhti
Date
Page

* Data that is used is distributed across different nodes.
* This actually the production mode of Hadoop.

Writing MapReduce Programs :-

* Software framework.
* Used for writing app. which process big amounts of data in parallel on large cluster.



* Single master (Job-Tracker) and one slave (Task-Tracker) per cluster.

Weather Dataset :-

* Data formal - from NCDC (National climatic data centre).

Takshti
Date
Page

* line oriented ~~ASCII~~ ASCII format.
* Each line is record.
* Data files are organized by Data and weather stations.

Analyzing the Data with Hadoop:

↳ Weather Dataset → Map Reduce → Hadoop

↳ I/P - Raw data from NCDC.

↳ Pull out year and air temperature in map function.

Hadoop MapReducer: (Record Reader):

↳ Loads data and convert into key value pair.

↳ Its instance is defined by the Input format.

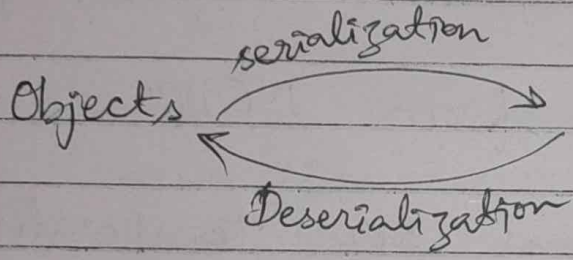
* Combiner: It is used to summarise the map output records with the same key. (semi-reducer).

Takhti
Date
Page

* Partitioner :- Controls the partitioning of the keys of the intermediate mapoutputs.
  ↳ by using hash function.

* The Writable Interface :-

*



* Hadoop uses its own serialization format writables.

* It defines two methods :-
  ↳ DataOutput (write)
  ↳ DataInput (readfields)

* Program :-

```
package org.apache.hadoop.io;
import java.io.DataOutput;
import java.io.DataInput;
public interface Writable
{
    void write(DataOutput out);
    void readfields(DataInput in);
}
```

Takhti
Date
Page

Note :- If writable is not present in Hadoop, then it uses the serialization of Java which increases the data over head in n network.

⊛ Writable Comparable and Comparator :-

* Sub Interface of Writables.
* Implementation of writable comparable is similar to writable but with an additional compare to method.

* Program :-

```
public interface WritableComparable
extends Writable, Comparable
{
  void readFields (DataInput in);
  void write (DataOutput out);
  int compareTo (WritableComparable O);
}
```

(returns -1, 0, 1)

$(K_1, V_1) \rightarrow$ Map $\rightarrow (K_2, V_2)$

$(K_2, \text{List}[V_2]) \rightarrow$ Reduce $\rightarrow (K_3, V_3)$

↑ (use here).

Takhti
Date
Page

* Program :-

```
package org.apache.hadoop.io;
import java.util.Comparator;
public interface RawComparator extends
  Comparator
{ public int compare (byte[]b, int s1,
  int l1, byte[]b2, int s2, int l2);
  }
```

* Writable comparator is a general purpose implementation of Raw Comparator for Writable Comparable classes.

Writable Classes :-

* Hadoop provides classes that wrap the Java primitives types and implements the writable comparable and writable interfaces.

* Provided in org.apache.hadoop.io package
  - set() → store a wrapped value.
  - get() → retrieve wrapped value.

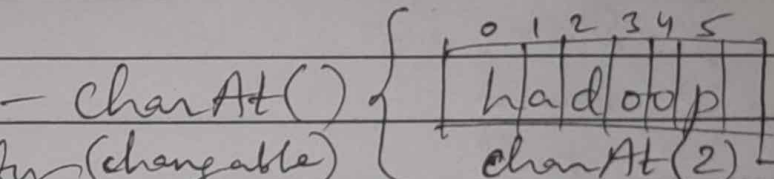
Takhti
Date
Page

Text :-

* Writable for UTF-8 sequences.
* Can be thought ~~as~~ of as writable equivalent of Java.long.String
* Uses an int to store no. of bytes in the string encoding. So, maximum value is 2GB.

Features :-

* Indexing - charAt()



* Mutability (changable)
* Restoring to string.

Byte Writable :-

~~* Writ~~
~~Writable~~

* Wrapper for an array of binary data.
* Serialized format is an ~~interface~~ integer field (4 bytes) that specifies the number of bytes to follow, followed by bytes themselves).

ex. | 3 | 5 | (0000 0002 0305)

Null Writable :- Special type (has zero length serialization).

↳ No bytes are written to or read from the stream.
↳ used as placeholder
↳ stores empty value.

Takhti
Date
Page

Object and Generic Writable :-

* Object :-

↳ It is a general purpose wrappers for the following. [Datatypes].
- ↳ Java primitives
- ↳ String
- ↳ enum
- ↳ null
- ↳ arrays of any of these types.

↳ Useful when a field can be more than one types.

* Generic :- (O.W)

↳ Compared with Object Writable, this class is much more effective.

↳ Because O.W. will ~~appear~~ append the class declaration as a string into the output file ~~as~~ in every key value pair.

↳ G.W. implements configure interface, so that it will be configured by the framework.

Datatypes

Takhti
Date
Page

(*) Writable Collection = (W.C)

(*) Array W.C. → Array writable (1D)
↳ Two dimentional A.W. (2D).

* Array primitive writable is a wrapper for arrays of Java primitives

* Used for sequence files, keys or values.

(*) Map W.C = Hadoop provides 3 Map writable datatypes which implements java.util.Map interface.

* Abstract M.W.
* Map writable
* Sorted M.W.

(*) Pig = Tool/platform used to analyze big-data.
↳ Represents Big data as data flows.

* Why do we need Apache Pig?
↳ It is suitable for those programmers who are not good at Java.

↳ Uses query approach (results in reducing the length of code).

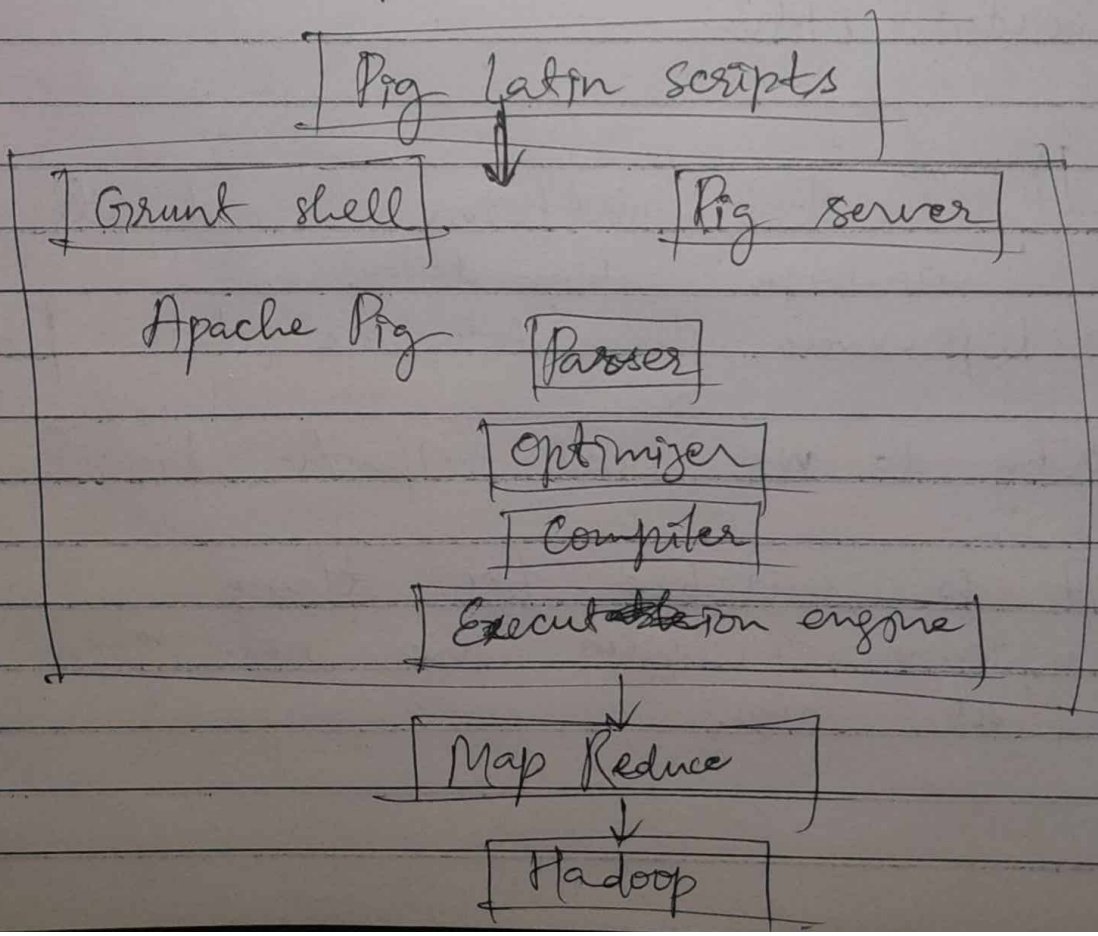
↳ Pig latin is SQL like language (easy to learn)

↳ Provides many built in operators (Join, filters, etc).

↳ Provides nested data types (tuples, bags, maps).

* Admiring the Pig Architecture :-

Pig → Language (Pig latin), (Grunt shell)
Pig → Compiler



Takhti
Date
Page

* Going with the Pig Latin Application flow:

```
A = LOAD "data-file.txt"
B = GROUP ...;
C = FILTER ...;
DUMP B;
STORE C INTO "Results";
```

(*) Working through the ABC's of (Pig Latin):

↳ Keep it Simple. (an abstraction that ~~Make it~~ simplifies the creation of parallel programs).

↳ Keep it Smart (Compiler can optimize the Java MapReduce Jobs automatically).

↳ Don't limit development :- (Make Pig extensible so that developer can add functions to address their particular business problems).

(*) Looking at Pig datatypes and Syntax :-

↳ Scalar - contain single value.
↳ Complex - contain other types.

Takhti
Date
Page

* Atom :- for example, 'Ayshi', 68, ...

Pig's atomic value are scalar type, int, float, long, double, char-array etc.

* Tuple :- Sequence of field

* Bag :- Collection of non-unique tuples.

* Map :- Collection of key-value pairs.
  ↳ Key → char-array type
  ↳ Value → any datatype.

(*) Evaluating local and Distributed Modes of Running Pig Scripts :-

* Local Mode :- ↳ Useful when we have small set of dataset.
  ↳ Useful for developing and testing Pig logic.
  ↳ Doesn't require Hadoop.
  ↳ Pig programs runs in the context of local JVM.
  ↳ Data access via the local file system.

* Distributed / MapReduce / Hadoop Mode :-

Pig scripts → Map Reduce Jobs → Hadoop Cluster.

Takhti
Date
Page

* Checking out the pig script Interfaces :-

↳ **Script :-**
- .pig - ex. FlightData.pig.

Commands are interpreted by PL compiler and executed in the order determined by Pig optimizer.

↳ **Grunt :-** Acts as a command interpreter. Helpful for prototyping during initial development.

↳ **Embedded :-** Pig latin statements can be executed with Java, Python, or Javascript. programs.

* Scripting with Pig latin :-

↳ Pig is designed to be extensible via UDFs (user defined functions).

↳ Some of the Pig UDFs → LOAD/STORE function, data-time, math, stats, etc.

↳ Pig can be embedded in host languages - Python, Java, Javascript.

↳ Pig doesn't support control flow statements.
↳ Needs to embedded in other language for using control flow.