* **Apache Pig :→**

Apache pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data ~~[illegible]~~ representing them as data flows. Pig is generally used with Hadoop. We can perform all the data manipulation operations in Hadoop using Apache pig.
To write data analysis Programs, Pig provides a high-level language known as Pig Latin. This language provides various ~~operati~~ operators using which programmers can develop their own functions for reading, writing and processing data.

* **features of Pig :→**

1. **Rich set of Operators :→**
It provides many operators to perform operations like Join, Sort, filer etc

2. **Easy of Programming :→**
Pig Latin is similar to SQL and it is easy to write a pig script if we are good at SQL.

3. **optimization opportunities :→**
The tasks in Apache pig optimize their execution automatically, so the programmers need to focus only on semantics of the language.

4. **Extensibility :→**
Using the existing operators, users can develop their own functions to read, Process and write data.

5. Handles all kinds of data :→
Apache pig analyzes all kinds of data, both structured as well as ~~[illegible]~~ unstructured. It stores the results in HDFS.

* Admiring the pig Architecture :→
Pig is made up of two (count 'em, two) Components :

1. The language itself :→
As proof that programmers have a sense of humor the programming language for pig is known as pig Latin, a high-level language that allows you to write data processing and analysis programs.

2. The pig Latin Compiler :→
The pig Latin Compiler Converts the pig Latin Code into executable Code. The executable Code is either in the form of MapReduce jobs or it can spawn a process where a virtual Hadoop instance is created to run the pig Code on a single node.

The sequence of MapReduce program enable pig programs to do data processing and analysis in parallel, leveraging Hadoop MapReduce and HDFS. Running the pig job in virtual Hadoop instance is a useful strategy for testing ~~out our~~ pig Scripts.

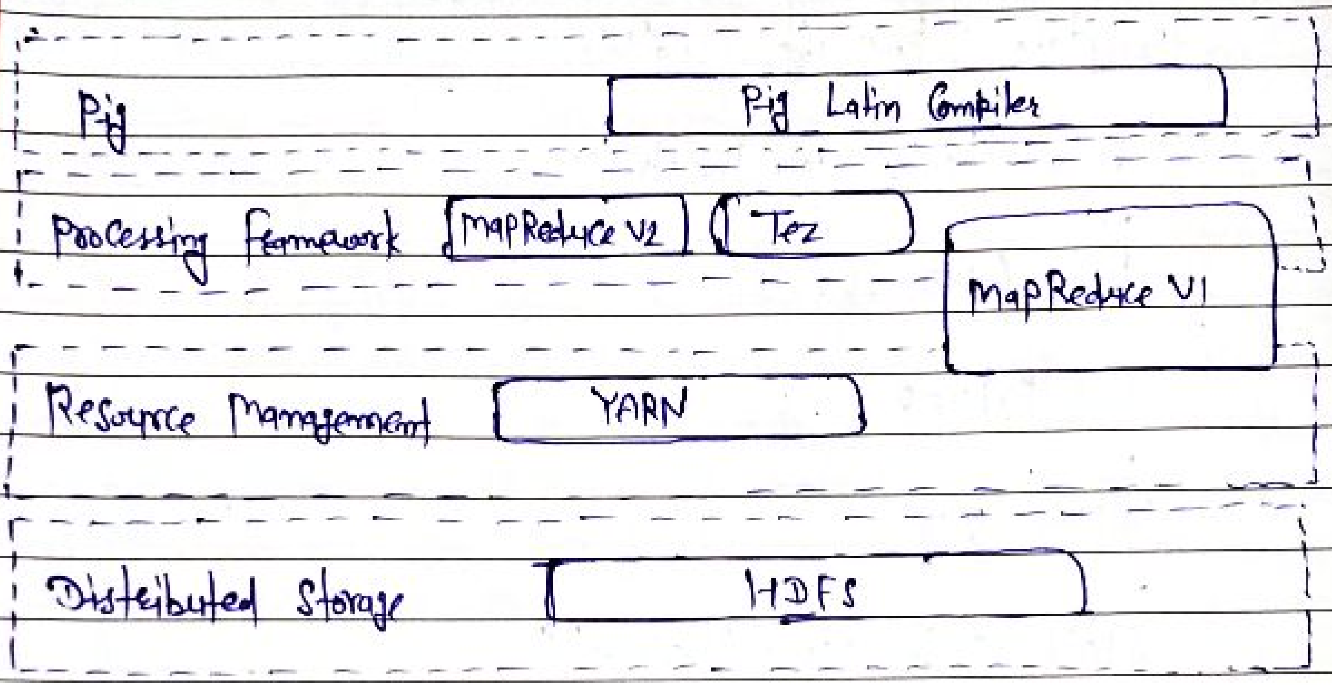

Figure 1: ~~Pig~~ Pig Architecture

Pig Program can run on MapReduce 1 or MapReduce 2 without any Code changes, regardless of what mode your Cluster is running. However, Pig Scripts can also run using the Tez API instead.

Going with the Pig Latin Application flow :→

At its Core, Pig Latin is a dataflow language, where ~~you~~ we define a data stream and a series of transformations that are applied to the data as it flows through our application. This is in Contrast to a Control flow language (like C or Java), where we write a series of instructions. In Control flow language, we use Constructs like loop and Conditional logic (like an if Statement) we won't find loops and if statements in Pig Latin.

If we need some convincing that working with pig is a significantly easier than having to write map and Reduce programs, start by taking a look at some real pig syntax.
The following Listing specifies sample pig code to illustrate the data processing dataflow.

```
A = LOAD 'data_file.txt';
...
B = GROUP ... ;
C = FILTER ... ;
...
DUMP B;
..
STORE C ~~bat~~ INTO 'Results';
```

* Working through the ABCs of Pig Latin :→
Pig Latin is the language for pig programs. Pig translates the pig Latin script into MapReduce jobs that can be executed within Hadoop cluster. when coming up with Pig Latin, the development team followed three key design principles :-

(1) keep it simple :→
Pig latin provides a streamlined method for interacting with Java MapReduce. It's an abstraction, in other words, that simplifiers the creation of parallel programs on the Hadoop cluster for data flows and analysis.
Complex tasks may require a series of interrelated data transformations - such series are encoded as data flow sequences.

writing data transformation and flows as pig Latin scripts instead of Java MapReduce programs make these programs easier to write, understand and maintain.

(2) Make it Smart :→
You may recall that the pig Latin Compiler does the work of transforming a pig Latin program into a Series of Java MapReduce Jobs. The trick is to make sure that the Compiler Can optimize the execution of these Java MapReduce Jobs automatically, allowing the user to focus on Semantics rather than on how to optimize and access the dataset.

(3) Don't limit development :→
Make ~~sup~~ pig extensible so that developers Can add functions to address their particular business problems.

* Evaluating Local and Distributed modes of Running Pig Scripts :→
Before you Can run your first pig script, you need to have a handle on how pig programs Can be packaged with the pig Server.
Pig has two modes for running Script, as shown in figure :-

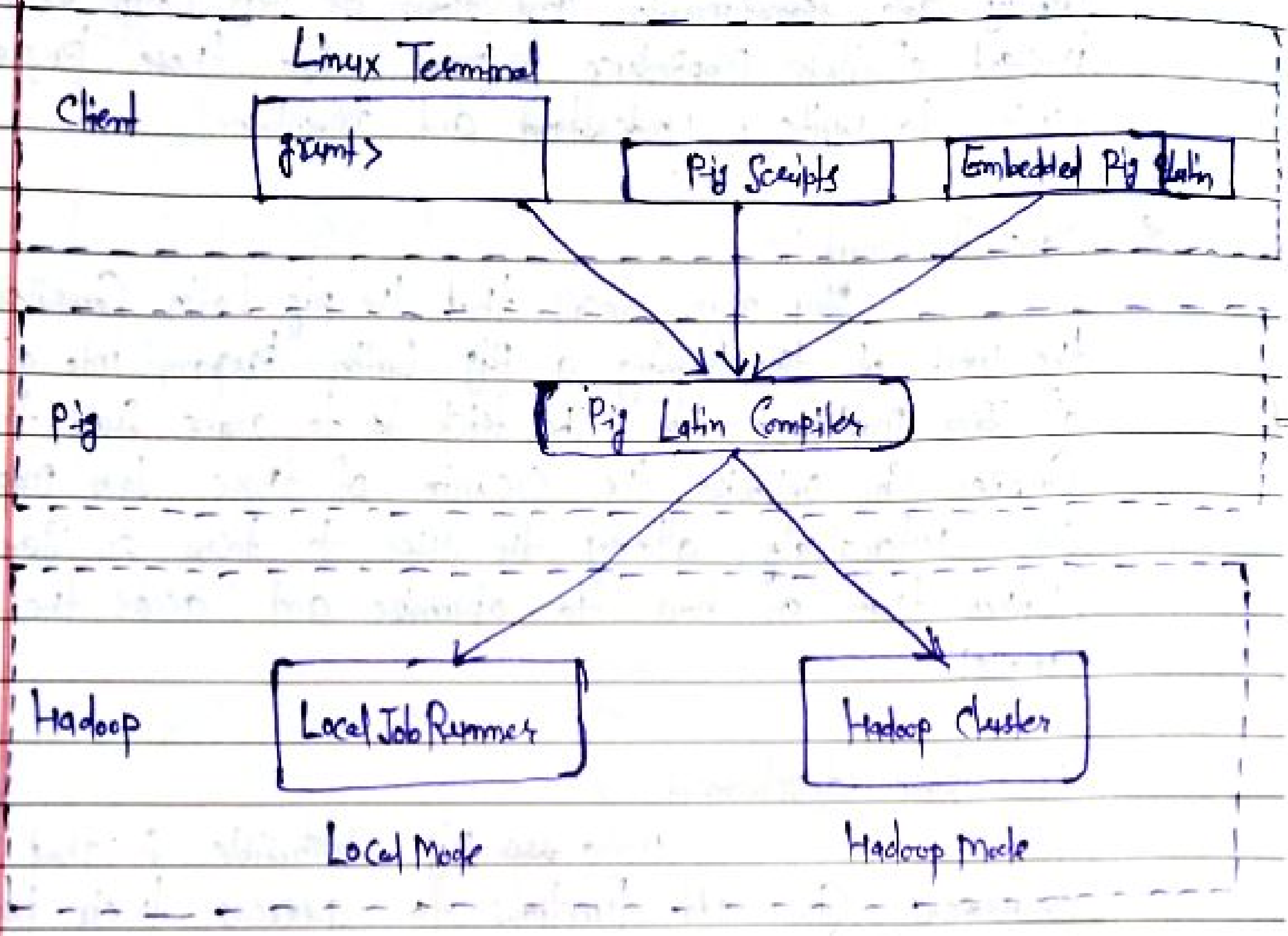

figure : Pig modes

Local mode:→

All scripts are run on a single machine without requiring Hadoop MapReduce and HDFS. This can be useful for developing and testing pig logic. If you're using a small set of data to develope or test your code, then local mode could be faster then going through the MapReduce infrastructure.

Local mode doesn't require Hadoop. When you run in Local mode, the pig program runs in the context of a local Java Virtual Machine, and data access is via the local file system of a single machine. Local mode is actually a local Simulation of MapReduce in Hadoop's LocalJobRunner class.

MapReduce Mode / Hadoop mode :→

MapReduce mode is also known as Hadoop Mode. Pig is executed on the Hadoop cluster. In this Case, the pig script gets Converted into a Series of MapReduce Jobs that are then run on the Hadoop Cluster. If you have a terabyte of data that you want to perform operations on and you want to interactively develop a program, you may Soon find things slowing down Considerably and you may Start growing your Storage.

* Checking out the Pig Script Interfaces :→

The pig Programming language is designed to handle any kind of data tossed its ways - Structured, Semi-Structured, unstructured data, ~~or~~

Pig programs Can be packaged in three different ways :-

(1) Script :→

This method is nothing more than a file Containing Pig Latin Commands, identified by the pig suffix (FlightData.pig). Ending your Pig program with the .pig extension is a Convention but not required. The Commands are interpreted by the Pig Latin Compiler and executed in the Order determined by the pig optimizer.

(2) Grunt :→

~~Grunt~~ Grunt acts as a Command interpreter where ~~you~~ we can interactively enter pig Latin at the Grunt Command line and immediately see the response. This method is helpful for prototyping during initial development and with what-if scenarios.

(3) Embedded :→

Pig Latin statements can be executed within Java, python or Javascript programs.

Pig scripts, Grunt shell pig commands, and embedded pig programs can run in either Local mode or MapReduce mode. The Grunt Shell provides an interactive shell to submit pig commands or run pig scripts.
To start the Grunt ~~Shell~~ Shell ~~is~~ ~~executed~~ ~~is~~ in Interactive mode, Just submit the command pig at your shell. To specify whether a script or Grunt shell is executed locally or in Hadoop mode just specify it in the – x flag to the pig Command.
The following is an example of how you'd specify running your pig script in local mode.

– Pig -x local milesPerCarrier.pig

Here's how you'd run the pig script in Hadoop mode, which is the default if you don't specify the flag.

– Pig -x mapreduce milesPerCarrier.Pig

By default, when we specify the pig Command ~~withoouan~~ without any parameters, it starts the Grunt shell in Hadoop mode. If we want to start the Grunt shell in local mode just add the -x local flag to the Command.
Here is an example :-
pig -x local

* Scripting with Pig Latin :->
Hadoop is a rich and quickly evolving ecosystem with a growing set of new applications. Rather than try to keep up with all the requirements for new Capabilities, pig is designed to be extensible via user-defined functions, also known as ~~UDF~~ UDFs. UDFs Can be written in a number of programming languages, including Java, Python and Javascript. Developers are also posting and sharing a growing Collection of UDFs online.

(Look for ~~Piggy~~ Piggy Bank and DataFu, to name just two examples of such online Collections)
Some of the Pig UDFs that are part of these repositories are LOAD/STORE function (XML, for example), date time, functions, text, math, and stats functions.

Pig Can also be Embedded in host languages such as Java, Python and Javascript, which allows you to integrate Pig with your existing applications.

* HDFS Architecture :→

HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored ~~as~~ across multiple machines. These files are stored in redundant fashion to rescure the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

HDFS Architecture.

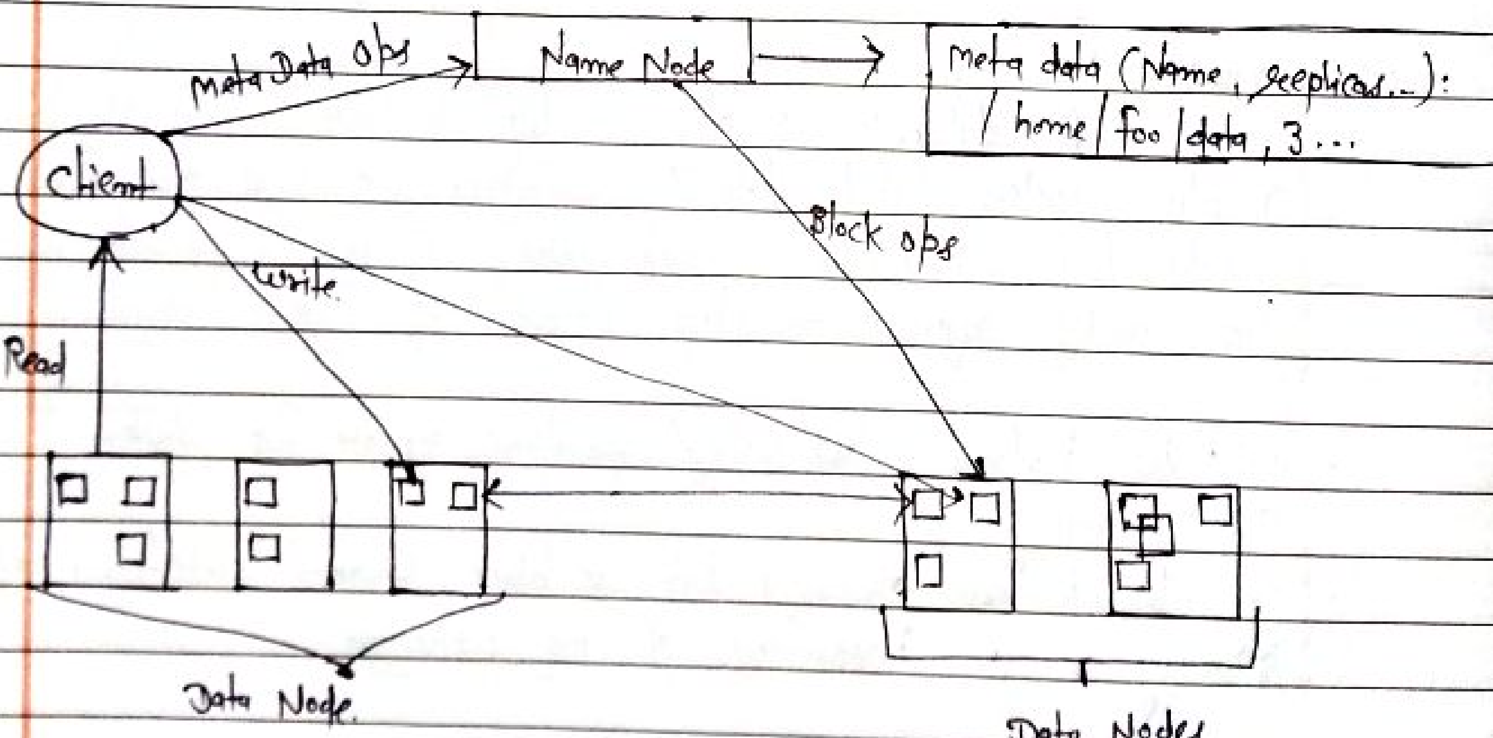

figure :- HDFS Architecture.

→ NameNode :→

The NameNode is the Commodity hardware that Contains the GNU/LINUX operating System and the NameNode Software. It is a Software that Can be run on Commodity hardware. The system having the NameNode acts as the master Server and it does the following tasks —

- Manages the files
- Store Meta Data
- It also executes file System operations such as renaming, closing, and opening files and directories.

→ DataNode :→

The datanode is a Commodity hardware having the GNU/Linux operating System and dataNode Software. for every Node (Commodity hardware/System) in a Cluster, there will be a datanode. These nodes manage the data Storage of their System.

- DataNodes perform read-write operations on the file System, as per Client request.
- They also perform operations such as block Creation, deletion, and replication according to the instructions of the NameNode.

→ NameNode :→

The NameNode is the Commodity hardware that Contains the GNU/LINUX operating System and the NameNode Software. It is a Software that Can be run on Commodity hardware. The System having the NameNode acts as the master Server and it does the following tasks –

- Manages the files
- Store MetaData
- It also executes file System operations such as renaming, closing, and opening files and directories.

→ DataNode :→

The datanode is a Commodity hardware having the GNU/Linux operating System and dataNode Software. for every Node (Commodity hardware/System) in a Cluster, there will be a datanode. These nodes manage the data Storage of their System.

- DataNodes perform read-write operations on the file System, as per Client request.
- They also perform operations such as block Creation, deletion, and replication according to the Instructions of the NameNode.