Prof R. Madana Mohana

BIG DATA ANALYTICS

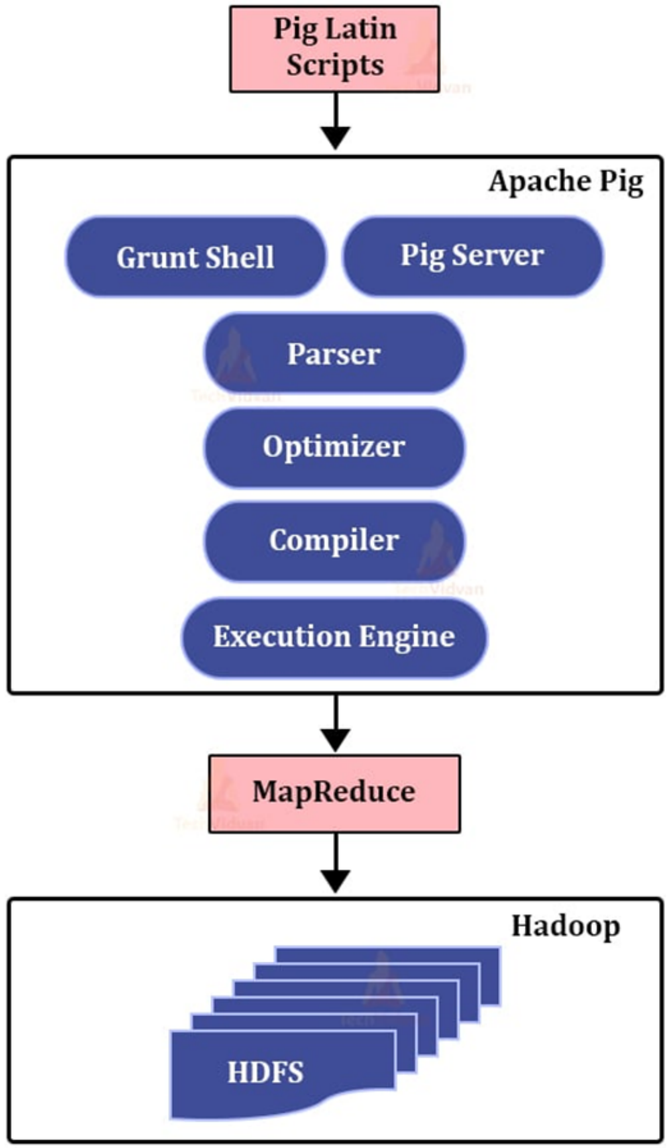
Apache Pig

Comparison with Databases

https://www.youtube.com/c/RASINENIMADANAMOHANA

Apache Pig: Architecture

Apache Pig: *Architecture*



Apache Pig: *Architecture*

Parser:

- As a **Pig Latin** program is executed, each statement is parsed in turn.
- If any syntactical errors are encountered, the interpreter halts and displays the error. Otherwise it builds a logical plan for each **Pig Latin** Statement (Operator)
- The logical plan for the statement is added to the logical plan for the program so far, and then the interpreter moves on to the next statement.

Apache Pig: *Architecture*

Parser:

- No data processing takes place while the logical plan of the program is being constructed.
- The output of the **parser** will be a **DAG** (directed acyclic graph), which represents the **Pig Latin** statements and logical operators.

Apache Pig: *Architecture*

Optimizer:

The logical plan (DAG) is passed to the logical optimizer, which carries out the logical optimizations such as projection and pushdown.

Compiler:

The compiler compiles the optimized logical plan into a series of MapReduce jobs.

Apache Pig: *Architecture*

Execution engine:

- Finally the MapReduce jobs are submitted to **Hadoop** in a sorted order.
- Finally, these MapReduce jobs are executed on **Hadoop** producing the desired results.

Apache Pig: Comparison with Databases

Apache Pig: *Differences between Pig and MapReduce*

Apache Pig	MapReduce
It is a scripting language.	It is a compiled programming language.
Pig is a data flow language.	MapReduce is a data processing paradigm.
Abstraction is at higher level.	Abstraction is at lower level.
It has less line of code as compared to MapReduce.	Lines of code is more.
Less effort is needed for Apache Pig.	More development efforts are required for MapReduce.
Code efficiency is less as compared to MapReduce.	As compared to Pig efficiency of code is higher.
Performing Join, filter, sorting or ordering operations are quite simple.	Relatively difficult to perform Join, filter, sorting or ordering operations between datasets.

Apache Pig: *Differences between Pig and MapReduce*

Apache Pig	MapReduce
Programmer with a basic knowledge of SQL can work conveniently.	Complex Java implementations require exposure to Java language.
Use multi-query approach, thereby reducing the length of codes significantly.	Require almost 20 times more the number of lines to perform the same task.
No need for compilation for execution; operators convert internally into MapReduce jobs.	Long compilation process for jobs.
Provides nested data types like tuples, bags and maps.	No such data types.

Apache Pig: *Differences between Pig and SQL*

Apache Pig	SQL
Pig Latin is a procedural language.	SQL is a declarative language.
In Apache Pig, schema is optional. We can store data without assigning a schema.	Schema is mandatory in SQL.
The data model in Apache Pig is nested relational.	The data model used in SQL is flat relational.
Apache Pig provides limited opportunity for Query optimization.	There is more opportunity for query optimization in SQL.

Apache Pig: *Differences between Pig and SQL*

In addition to *above differences*, **Apache Pig Latin**:

- Allows splits in the pipeline.
- Allows developers to store data anywhere in the pipeline.
- Declares execution plans.
- Provides operators to perform ETL (Extract, Transform, and Load) functions.

Apache Pig: *Differences between Pig and Hive*

Apache Pig	Hive
Originally created at Yahoo.	Originally created at Facebook.
Exploits Pig Latin language.	Exploits HiveQL.
Pig Latin is dataflow language.	HiveQL is a query processing language.
Pig Latin is a procedural language and it fits in pipeline paradigm.	HiveQL is a declarative language.
Handles structured, unstructured and semi-structured data.	Mostly used for structured data.

Apache Pig: *Comparison with Databases*

- Having seen **Pig** in action, it might seem that **Pig Latin** is similar to **SQL**.
- The presence of such operators as **GROUP BY** and **DESCRIBE** reinforces this impression.
- However, there are several differences between the two languages, and between **Pig** and **relational database management systems (RDBMS**s) in general.

Apache Pig: *Comparison with Databases*

- The most significant difference is that **Pig Latin** is a *data flow programming language*, whereas **SQL** is a *declarative programming language*.
- In other words, a **Pig Latin** program is a step-by-step set of operations on an input relation, in which each step is a single transformation.

Apache Pig: *Comparison with Databases*

- By contrast, **SQL** statements are a set of constraints that, taken together, define the output.
- In many ways, programming in **Pig Latin** is like working at the level of an **RDBMS** query planner, which figures out how to turn a declarative statement into a system of steps.
- **RDBMSs** store data in tables, with tightly predefined schemas.

Apache Pig: *Comparison with Databases*

- **Pig** is more relaxed about the data that it processes: we can define a schema at runtime, but it's optional.
- Essentially, it will operate on any source of tuples (although the source should support being read in parallel, by being in multiple files, for example), where a **UDF** is used to read the tuples from their raw representation.

Apache Pig: *Comparison with Databases*

- The most common representation is a text file with tab-separated fields, and **Pig** provides a built-in **load** function for this format.
- Unlike with a traditional database, there is no data import process to load the data into the **RDBMS**.
- The data is loaded from the filesystem (usually **HDFS**) as the first step in the processing.

Apache Pig: *Comparison with Databases*

- **Pig**'s support for complex, nested data structures further differentiates it from **SQL**, which operates on flatter data structures.
- Also, **Pig'**s ability to use **UDFs** and streaming operators that are tightly integrated with the language and **Pig**'s nested data structures makes **Pig Latin** more customizable than most **SQL dialects**.

Apache Pig: *Comparison with Databases*

- **RDBMSs** have several features to support online, low-latency queries, such as transactions and indexes, that are absent in **Pig**.
- **Pig** does not support random reads or queries on the order of tens of milliseconds. Nor does it support random writes to update small portions of data; all writes are bulk streaming writes, just like with **MapReduce**.

Apache Pig: *Comparison with Databases*

- **Hive** (covered in upcoming lectures) sits between **Pig** and conventional **RDBMSs.**
- Like **Pig**, **Hive** is designed to use **HDFS** for storage, but otherwise there are some significant differences. Its query language, **HiveQL**, is based on **SQL**, and anyone who is familiar with **SQL** will have little trouble writing queries in **HiveQL**.

Apache Pig: *Comparison with Databases*

- Like **RDBMSs**, **Hive** mandates that all data be stored in tables, with a schema under its management; however, it can associate a schema with preexisting data in **HDFS**, so the **load** step is optional.
- **Pig** is able to work with **Hive tables** using **HCatalog.**