Q1-Why MapReduce program is needed in Pig Programming?

Answer:

1. Pig raises the level of abstraction for processing large datasets.
2. Pig is made up of two pieces:

-The language used to express data flows, called Pig Latin.

-The execution environment to run Pig Latin programs.

3. A Pig Latin program is made up of a series of operations, or transformations, that are applied to the input data to produce output.

4. Taken as a whole, the operations describe a data flow, which the Pig execution environment translates into an executable representation and then runs.

5. Pig turns the transformations into a series of MapReduce jobs, but as a programmer you are mostly unaware of this, which allows you to focus on the data rather than the nature of the execution.

Q2-What are advantages of pig over MapReduce?

Answer:

1. Hadoop MapReduce is a compiled language whereas Apache Pig is a scripting language.
2. Pig and Hive provide higher level of abstraction whereas Hadoop MapReduce provides low level of abstraction.
3. Pig shortens the length of code as compared to MapReduce.
4. When using Pig for executing jobs, Hadoop developers need not worry about any version mismatch.
5. There is very limited possibility for the developer to write java level bugs when coding in Pig.

Q3- What is pig engine and what is its importance?

Answer:

As shown in the figure, there are various components in the Apache Pig framework.



**Parser**

Initially the Pig Scripts are handled by the Parser. It checks the syntax of the script, does type checking, and other miscellaneous checks. The output of the parser will be a DAG (directed acyclic graph), which represents the Pig Latin statements and logical operators.

In the DAG, the logical operators of the script are represented as the nodes and the data flows are represented as edges.

**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.

**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.

**Importance:**

1. Pig engine acts as an interpreter between Pig Latin script and MapReduce jobs.
2. It translates Pig Latin script into a series of MapReduce jobs, but the programmer is mostly unaware of this, which enables the programmer to focus on the data rather than the nature of the execution.

Q4-What are the modes of Pig execution?

Answer:

1. **MapReduce/Hadoop Mode:**

Here Pig jobs run as a series of MapReduce jobs picking the input and output paths from HDFS.

2. **Local Mode:**

Here the entire Pig job runs as a single JVM picking the local Unix path for execution.

Q5-What is grunt shell in Pig?

Answer:

1. Grunt is an interactive shell for running Pig commands.

2. Grunt is started when no file is specified for Pig to run, and the -e option is not used.

3. It is also possible to run Pig scripts from within Grunt using run and exec.

Q6-What are the features of Pig Latin language?

Answer:

1. Pig can operate on data with or without metadata.

2. Pig is intended to be a language for parallel data processing.

3. Pig is not tied to any particular framework.

4. Pig processes data very fast-no need to go through a time-consuming data import process prior to running queries, as in conventional databases.

5. Pig allows integration of user code where ever possible, so it supports user defined:

• Field transformation functions

• Aggregates

• Conditionals

Q7- Is Pig Latin commands case sensitive?

Answer:

1. The names (aliases) of relations and fields are case sensitive.

2. The names of Pig Latin functions are case sensitive.

3. The names of parameters and all other Pig Latin keywords are case insensitive.

Q8. What is a data flow language?

Answer:

1. In a dataflow language, you have a stream of data which is passed from instruction to instruction to be processed.

2. Conditional execution, jumps and procedure calls route the data to different instructions.

3. This could be seen as data flowing through otherwise static instructions like how electrical signals flow through circuits or water flows through pipes.

4. A dataflow "if" statement would route the data to the correct branch.