PIG 4

IMPORTANT QUESTIONS

PART-A

Q.1 ***What are the different ways of executing a Pig script?***

Ans. The different ways of executing a Pig script are as follows:

(i) Grunt shell

(ii) Script file

(iii) Embedded script

Q.2 ***What is Pig scripts?***

Ans. Pig Scripts: They are written in Pig Latin using built-in operators and UDFs, and submitted to the execution environment.

Q.3 ***Explain parser.***

Ans. Parser: Completes type checking and checks the syntax of the script. The output of the parser is a Directed Acyclic Graph (DAG).

Q.4 ***What is the aim of optimizer.***

Ans. Optimizer: Performs optimization using merge, transform, split, etc. Optimizer aims to reduce the amount of data in the pipeline.

Q.5 ***Write the function of compiler and execution engine.***

Ans. Compiler: Converts the optimized code into MapReduce jobs automatically.

Execution Engine: MapReduce jobs are submitted to execution engines to generate the desired results.

Q.6 ***Explain tuple operator.***

Ans. Tuple : A tuple is an ordered set of fields that can contain different data types for each field. It is represented by braces ().

Example: (1,3)

Q.7 ***What do you mean by bag and map operator?***

Ans. Bag : A bag is a set of tuples represented by curly braces {}.

Example: {(1,4), (3,5), (4,6)}

Map : A map is a set of key-value pairs used to represent data elements. It is represented in square brackets [].

Example: [key#value, key1#value1,.....]

Q.8 ***What is the use of having filters in Apache Pig?***

Ans. FilterOperator is used to select the required tuples from a relation based on a condition. It also allows you to remove unwanted records from the data file.

Example: Filter the products with a whole quantity that is greater than 1000

A = LOAD '/user/Hadoop/phone_sales' USING PigStorage(',') AS (year:int, product:chararray, quantity:int);
B = FILTER A BY quantity > 1000

"phone_sales" data

year, product, quantity
2000, phone, 1000
2001, phone, 1500
2002, phone, 1700
2003, phone, 1200
2004, phone, 800
2005, phone, 900

Q.9 ***Suppose there's a file called "test.txt" having 150 records in HDFS. Write the PIG command to retrieve the first 10 records of the file.***

Ans. To do this, we need to use the limit operator to retrieve the first 10 records from a file.

Load the Data in Pig:

test_data = LOAD "/user/test.txt" USING PigStorage(',') as (field1, field2,.....);

Limit the Data to First 10 Records:

Limit_test_data = LIMIT test_data 10;

Q.10 ***What are the features of bag?***

Ans. Features of Bag :

(i) A bag can have duplicate tuples.

(ii) A bag can have tuples with differing numbers of fields. However, if Pig tries to access a field that does not exist, a null value is substituted.

(iii) A bag can have tuples with fields that have different data types. However, for Pig to effectively process bags, the schemas of the tuples within those bags should be the same.

Q.11 ***What is an outer bag?***

Ans. Outer Bag : An outer bag is nothing but a relation.

Q.12 ***What is an inner bag?***

Ans. Inner Bag : An inner bag is a relation inside any other bag.

Example: (4,{(4,2,1),(4,3,3)})

In the above example, the complete relation is an outer bag and {(4,2,1),(4,3,3)} is an inner bag.

PART-B

Q.13 ***What are the various diagnostic operators available in Apache Pig?***

Ans. Pig has Dump, Describe, Explain, and Illustrate as the various diagnostic operators.

Dump : The dump operator runs the Pig Latin scripts and displays the results on the screen. Load the data using the "load" operator into Pig.

Display the results using the "dump" operator.

Describe : Describe operator is used to view the schema of a relation.

Load the data using "load" operator into Pig.

View the schema of a relation using "describe" operator.

Explain : Explain operator displays the physical, logical and MapReduce execution plans.

Load the data using "load" operator into Pig.

Display the logical, physical and MapReduce execution plans using "explain" operator.

Illustrate : Illustrate operator gives the step-by-step execution of a sequence of statements.

Load the data using "load" operator into Pig.

Show the step-by-step execution of a sequence of statements using "illustrate" operator.

Q.14 ***State the usage of the group, order by, and distinct keywords in Pig scripts.***

Ans. The group statement collects various records with the same key and groups the data in one or more relations.

Example: Group_data = GROUP Relation_name BY AGE

The order statement is used to display the contents of relation in sorted order based on one or more fields.

Example: Relation_2 = ORDER Relation_name1 BY (ASC|DSC)

Distinct statement removes duplicate records and is implemented only on entire records, and not on individual records.

Example: Relation_2 = DISTINCT Relation_name1

Q.15 ***What are the relational operators in Pig?***

Ans. The relational operators in Pig are as follows:

COGROUP : It joins two or more tables and then performs GROUP operation on the joined table result.

CROSS : This is used to compute the cross product (cartesian product) of two or more relations.

FOREACH : This will iterate through the tuples of a relation, generating a data transformation.

JOIN : This is used to join two or more tables in a relation.

LIMIT : This will limit the number of output tuples.

SPLIT : This will split the relation into two or more relations.

UNION : It will merge the contents of two relations.

ORDER : This is used to sort a relation based on one or more fields.

Q.16 ***Write a short note on admiring the Pig architecture.***

Ans. Admiring the Pig Architecture : Pig is made up of two components:

(i) The Language Itself : The programming language for Pig is known as Pig Latin, a high-level language that allows you to write data processing and analysis programs.

(ii) 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 programs enables Pig programs to do data processing and analysis in parallel, leveraging Hadoop MapReduce and HDFS. Running the Pig job in the virtual Hadoop instance is a useful strategy for testing your Pig scripts. Figure shows how Pig relates to the Hadoop ecosystem.

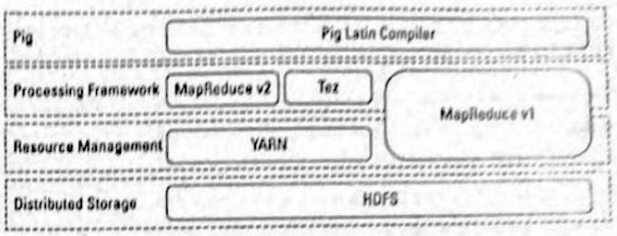

Fig. : Pig architecture

Pig programs 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. Apache Tez provides a more efficient execution framework than MapReduce. YARN enables application frameworks other than MapReduce (like Tez) to run on Hadoop. Hive can also run against the Tez framework.

Q.17 ***Explain grunt.***

Ans. Grunt : Grunt has line-editing facilities like those found in GNU Readline (used in the bash shell and many other command-line applications). For instance, the Ctrl-E key combination will move the cursor to the end of the line. Grunt remembers command history, too, and you can recall lines in the history buffer using Ctrl-P or Ctrl-N (for previous and next) or, equivalently, the up or down cursor keys.

Another handy feature is Grunt's completion mechanism, which will try to complete Pig Latin keywords and functions when you press the Tab key. For example, consider the following incomplete line:

```
grunt> a = foreach b ge
```

If you press the Tab key at this point, ge will expand to generate, a Pig Latin keyword:

```
grunt> a = foreach b generate
```

You can customize the completion tokens by creating a file named *auto complete* and placing it on Pig's classpath (such as in the conf directory in Pig's install directory), or in the directory you invoked Grunt from. The file should have one token per line, and tokens must not contain any whitespace. Matching is case-sensitive. It can be very handy to add commonly used file paths (especially because Pig does not perform file-name completion) or the names of any user-defined functions you have created.

You can get a list of commands using the help command. When you've finished your Grunt session, you can exit with the quit command.

Q.18 ***Why do we need Apache Pig?***

Ans. Programmers who are not so good at Java normally used to struggle working with Hadoop, especially while performing any MapReduce tasks. Apache Pig is a boon for all such programmers.

- Using Pig Latin, programmers can perform MapReduce tasks easily without having to type complex codes in Java.
- Apache Pig uses multi-query approach, thereby reducing the length of codes. For example, an operation that would require you to type 200 lines of code (LoC) in Java can be easily done by typing as less as just 10 LoC in Apache Pig. Ultimately Apache Pig reduces the development time by almost 16 times.
- Pig Latin is SQL-like language and it is easy to learn Apache Pig when you are familiar with SQL.
- Apache Pig provides many built-in operators to support data operations like joins, filters, ordering, etc. In addition, it also provides nested data types like tuples, bags, and maps that are missing from MapReduce.

Q.19 ***Write the features of Pig.***

Ans. Features of Pig : Apache Pig comes with the following features :

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 you 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. **UDF's :** Pig provides the facility to create User-defined Functions in other programming languages such as Java and invoke or embed them in Pig Scripts.
6. **Handles all kinds of data :** Apache Pig analyzes all kinds of data, both structured as well as unstructured. It stores the results in HDFS.

Q.20 ***Write the differences between Apache Pig and MapReduce.***

Ans. Apache Pig Vs MapReduce : Listed below are the major differences between Apache Pig and MapReduce.

Apache Pig	MapReduce
Apache Pig is a data flow language.	MapReduce is a data processing paradigm.
It is a high level language.	MapReduce is low level and rigid
Performing a Join operation in Apache Pig is pretty simple.	It is quite difficult in MapReduce to perform a Join operation between datasets.
Any novice programmer with a basic knowledge of SQL can work conveniently with Apache Pig.	Exposure to Java is must to work with MapReduce.
Apache Pig uses multi-query approach, thereby reducing the length of the codes to a great extent.	MapReduce will require almost 20 times more the number of lines to perform the same task.
There is no need for compilation. On execution, every Apache Pig operator is converted internally into a MapReduce job.	MapReduce jobs have a long compilation process.

Q.21 ***Explain Pig Latin application flow.***

Ans. At its core, Pig Latin is a dataflow language, where you define a data stream and a series of transformations that are applied to the data as it flows through your application. This is in contrast to a control flow language (like C or Java), where you write a series of instructions.

In control flow languages, you use constructs like loops and conditional logic (like an if statement). You won't find loops and if statements in Pig Latin.

If you need some convincing that working with Pig is a significantly easier row to hoe than having to write Map

and Reduce programs, start by taking a look at some real Pig syntax:

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

Some of the text in this example actually looks like English, right? Not too scary, at least at this point. Looking at each line in turn, you can see the basic flow of a Pig program. (Note that this code can either be part of a script or issued on the interactive shell called Grunt.)

1. Load: You first load (LOAD) the data you want to manipulate. As in a typical MapReduce job, that data is stored in HDFS. For a Pig program to access the data, you first tell Pig what file or files to use. For that task, you use the LOAD 'data_file' command.

Here, 'data_file' can specify either an HDFS file or a directory. If a directory is specified, all files in that directory are loaded into the program.

If the data is stored in a file format that isn't natively accessible to Pig, you can optionally add the USING function to the LOAD statement to specify a user-defined function that can read in (and interpret) the data.

2. Transform: You run the data through a set of transformations that, way under the hood and far removed from anything you have to concern yourself with, are translated into a set of Map and Reduce tasks.

The transformation logic is where all the data manipulation happens. Here, you can FILTER out rows that aren't of interest, JOIN two sets of data files, GROUP data to build aggregations, ORDER results, and do much, much more.

3. Dump: Finally, you dump (DUMP) the results to the screen or Store (STORE) the results in a file somewhere.

You would typically use the DUMP command to send the output to the screen when you debug your programs. When your program goes into production, you simply change the DUMP call to a STORE call so that any results from running your programs are stored in a file for further processing or analysis.

Q.22 ***Explain the various pig latin operations.***

Ans. Pig Latin Operations : In a Hadoop context, accessing data means allowing developers to load, store, and stream data, whereas transforming data means taking advantage of Pig's ability to group, join, combine, split, filter, and sort data.

Data Access

Operator	Explanation
LOAD/STORE	Read and write data to file system
DUMP	Write output to standard output (stdout)
STREAM	Send all records through external binary
FOREACH	Apply expression to each record and output one or more records
FILTER	Apply predicate and remove records that don't meet condition
GROUP/COGROUP	Aggregate records with the same key from one or more inputs
JOIN	Join two or more records based on a condition

Transformations

Operator	Explanation
CROSS	Cartesian product of two or more inputs
ORDER	Sort records based on key
DISTINCT	Remove duplicate records
UNION	Merge two data sets
SPLIT	Divide data into two or more bags based on predicate
LIMIT	Subset the number of records

Operators for Debugging and Troubleshooting

Operator	Explanation
DESCRIBE	Return the schema of a relation
DUMP	Dump the contents of a relation to the screen
EXPLAIN	Display the MapReduce execution plans

Q.23 ***What are the advantages of using Pig over Mapreduce?***

Ans. In Mapreduce :

(i) Development cycle is very long. Writing mappers and reducers, compiling.

(ii) Packaging the code, submitting jobs, and retrieving the results is a time.

(iii) Consuming process.

(iv) Performing Data set joins is very difficult.

(v) Low level and rigid, and leads to a great deal of custom user code that is hard to maintain and reuse is complex.

In pig :

(i) No need of compiling or packaging of code. Pig operators will be converted into map or reduce tasks internally.

(ii) Pig Latin provides all of the standard data-processing operations, such as join, filter, group by, order by, union, etc.

(iii) High level of abstraction for processing large data sets.

Q.24 ***Explain Pig script interfaces in Hadoop.***

Ans. The Pig programming language is designed to handle any kind of data tossed its way – structured, semi structured, unstructured data, you name it. 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, for example). 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 acts as a command interpreter where you 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 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
```

[**Note :** By default, when you specify the Pig command without any parameters, it starts the Grunt shell in Hadoop mode. If you 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]

Q.25 ***Explain nested model and also explain interactive modes of Pig.***

Ans. Pig Latin has a fully-nestable data model with Atomic values, Tuples, Bags or lists, and Maps. This implies one data type can be nested within another. Pig Latin Nested Data Model is shown in the following Fig. 1.

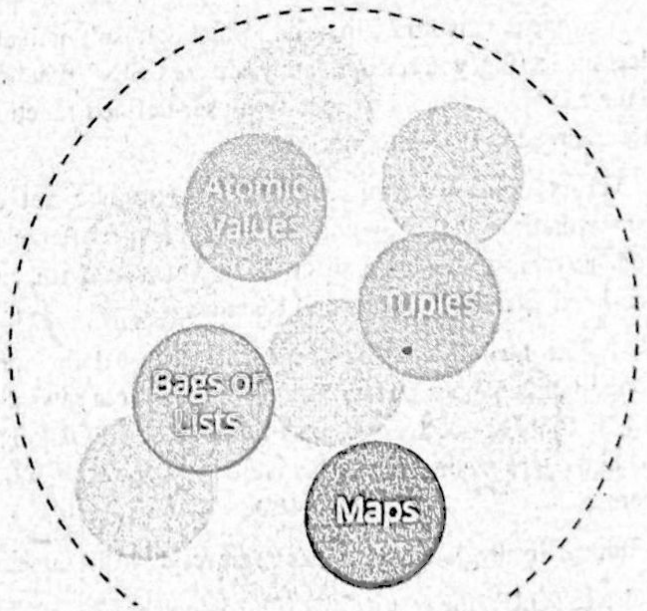

Fig. 1

The advantage is that this is more natural to programmers than flat Tuples. Also, it avoids expensive joins.

Now we will look into different execution modes pig works in.

Interactive Mode : Interactive mode means coding and executing the script, line by line, as shown in the image given below.

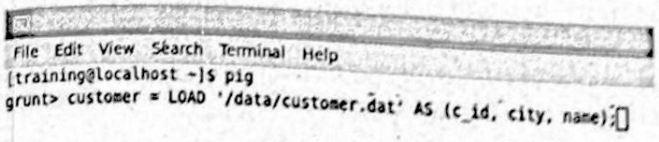

Fig. 2

Batch Mode : In Batch mode, all scripts are coded in a file with the extension .pig and the file is directly executed as shown in the diagram given below :

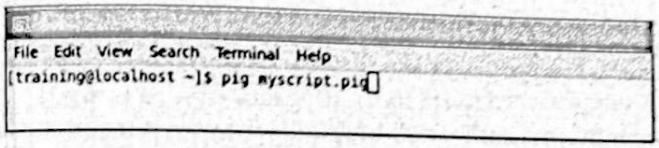

Fig. 3

PART-C

Q.26 ***Write detailed note on Pig.***

Ans. Pig raises the level of abstraction for processing large datasets. MapReduce allows you the programmer to specify a map function followed by a reduce function, but working out how to fit your data processing into this pattern, which often requires multiple MapReduce stages, can be a challenge. With Pig, the data structures are much richer, typically being multivalued and nested; and the set of transformations you can apply to the data are much more powerful–they include joins, for example, which are not for the faint of heart in MapReduce.

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. There are currently two environments: Local execution in a single JVM and distributed execution on a Hadoop cluster.

A Pig Latin program is made up of a series of operations, or transformations, that are applied to the input data to produce output. Taken as a whole, the operations describe a data flow, which the Pig execution environment translates into an executable representation and then runs. Under the covers, 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.

Pig is a scripting language for exploring large datasets. One criticism of MapReduce is that the development cycle is very long. Writing the mappers and reducers, compiling and packaging the code, submitting the job(s), and retrieving the results is a time-consuming business, and even with screaming, which removes the compile and package step, the experience is still involved. Pig's sweet spot is its ability to process terabytes of data simply by issuing a half-dozen lines of Pig Latin from the console. Indeed, it was created at Yahoo! to make it easier for researchers and engineers to mine the huge datasets there. Pig is very supportive of a programmer writing a query, since it provides several commands for introspecting the data structures in your program, as it is written. Even more useful, it can perform a sample run on a representative subset of your input data, so you can see whether there are errors in the processing before unleashing it on the full dataset.

Pig was designed to be extensible. Virtually all parts of the processing path are customizable: Loading, storing, filtering, grouping, and joining can all be altered by user-defined functions (UDFs). These functions operate on Pig's nested data model, so they can integrate very deeply with Pig's operators. As another benefit, UDFs tend to be more reusable than the libraries developed for writing MapReduce programs.

Pig isn't suitable for all data processing tasks, however. Like MapReduce, it is designed for batch processing of data. If you want to perform a query that touches only a small amount of data in a large dataset, then Pig will not perform well, since it is set up to scan the whole dataset, or at least large portions of it.

In some cases, Pig doesn't perform as well as programs written in MapReduce. However, the gap is narrowing with each release, as the Pig team implements sophisticated algorithms for implementing Pig's relational operators. It's fair to say that unless you are willing to invest a lot of effort optimizing Java MapReduce code, writing queries in Pig Latin will save you time.

Q.27 *Explain how to install and run a Pig.*

Ans. Installing and Running Pig : Pig runs as a client-side application. Even if you want to run Pig on a Hadoop cluster, there is nothing extra to install on the cluster: Pig launches jobs and interacts with HDFS (or other Hadoop filesystems) from your workstation.

Installation is straightforward. Java 6is a prerequisite (and on Windows, you will need Cygwin). Download a stable release from http://pig.apache.org/releases.html, and un-pack the tarball in a suitable place on your workstation:

% tar xzf pig-x.y.z.tar.gz

It's convenient to add Pig's binary directory to your command-line path. For example:

% export PIG_INSTALL=/home/ton/pig-x.y.z

% export PATH=$PATH:$PIG_INSTALL/bin

You also need to set the JAVA_HOME environment variable to point to a suitable Java installation.

Try typing pig -help to get usage instructions.

Execution Types : Pig has two execution types or modes: Local mode and MapReduce mode.

(i) Local Mode : In local mode, Pig runs in a single JVM and accesses the local filesystem. This mode is suitable only for small datasets and when crying out Pig.

The execution type is set using the -x or -exectype option. To run in local mode, set the option to local:

% pig -x local

grunt>

This starts Grunt, the Pig interactive shell.

(ii) MapReduce Mode : In MapReduce mode, Pig translates queries into MapReduce jobs and runs them on a Hadoop cluster. The cluster may be a pseudo- or fully distributed cluster. MapReduce mode (with a fully distributed cluster) is what you use when you want to run Pig on large datasets.

To use MapReduce mode, you first need to check that the version of Pig you down loaded is compatible with the version of Hadoop you are using. Pig releases will only work against particular versions of Hadoop; this is documented in the release notes.

Pig honors the HADOOP_HOME environment variable for finding which Hadoop client to run. However if it is not set, Pig will use a bundled copy of the Hadoop libraries. Note that these may not match the version of Hadoop running on your cluster, so it is best to explicitly set HADOOP_HOME.

Next, you need to point Pig at the cluster's namenode and jobtracker. If the installation of Hadoop at HADOOP_HOME is already configured for this, then there is nothing more to do. Otherwise, you can set HADOOP_CONF_DIR to a directory containing the Hadoop site file (or files) that define fs.default.name and mapred.Job.tracker.

Alternatively, you can set these two properties in the pig.properties file in Pig's conf directory (or the directory specified by PIG_CONF_DIR}. Here's an example for a pseudo- distributed setup:

fs.default.name=hdfs: //localhost/

mapred. job. tracker=localhost:8021

Once you have configured Pig to connect to a Hadoop cluster, you can launch Pig, setting the -x option to mapreduce, or omitting it entirely, as MapReduce mode is the default:

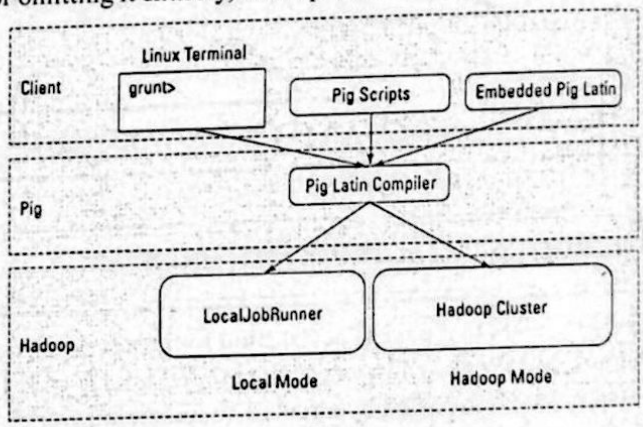

Fig.

Running Pig Programs : There are three ways of executing Pig programs, all of which work in both local and MapReduce mode:

Script : Pig can run a script file that contains Pig commands. For example, pig script.pig runs the commands in the local file script.pig. Alternatively, for very short scripts, you can use the -e option to run a script specified as a string on the command line.

Grunt : Grunt is an interactive shell for running Pig commands. Grunt is started when no file is specified for Pig to run, and the -e option is not used. It is also possible to run Pig scripts from within Grunt using run and exec.

Embedded : You can run Pig programs from Java using the PigServer class, much like you can use JDBC to run SQL programs from Java. For programmatic access to Grunt, use PigRunner.

Q.28 *Explain the structure of Pig latin.*

Ans. Structure : A Pig Latin program consists of a collection of statements. A statement can be thought of as an operation, or a command. For example, a GROUP operation is a type of statement:

```
grouped_records = GROUP records BY year;
```

The command to list the files in a Hadoop filesystem is another example of a statement:

```
ls /
```

Statements are usually terminated with a semicolon, as in the example of the GROUP statement. In fact, this is an example of a statement that must be terminated with a semicolon: It is a syntax error to omit it. The ls command, on the other hand, does not have to be terminated with a semicolon. As a general guideline, statements or commands for interactive use in Grunt do not need the terminating semicolon. This group includes the interactive Hadoop commands, as well as the diagnostic operators like DESCRIBE. It's never an error to add a terminating semicolon, so if in doubt, it's simplest to add one.

Statements that have to be terminated with a semicolon can be split across multiple lines for readability:

```
records = LOAD 'input /ncdc/micro-tab/sample.txt'
AS (year:chararray, temperature:int, quality:int);
```

Pig Latin has two forms of comments. Double hyphens are single-line comments. Everything from the first hyphen to the end of the line is ignored by the Pig Latin interpreter:

```
- - My program
DUMP A; -- What's in A?
```

C-style comments are more flexible since they delimit the beginning and end of the comment block with /*and */ markers. They can span lines or be embedded in a single line:

```
/*
* Description of my program spanning
* multiple lines.
*/
A = LOAD 'input/pig/join/A';
B = LOAD 'input/pig/join/B';
C = JOIN A BY $0, /* ignored */ B BY $1;
DUMP C;
```

Pig Latin has a list of keywords that have a special meaning in the language and cannot be used as identifiers. These include the operators (LOAD, ILLUSTRATE), commands (cat, ls), expressions (matches, FLATTEN), and functions (DIFF, MAX)–all of which are covered in the following sections.

Pig Latin has mixed rules on case sensitivity. Operators and commands are not case- sensitive (to make interactive use more forgiving); however, aliases and function names are case-sensitive.

The language used to analyze data in Hadoop using Pig is known as Pig Latin. It is a high level data processing language which provides a rich set of data types and operators to perform various operations on the data.

To perform a particular task programmers using Pig, programmers need to write a Pig script using the Pig Latin language, and execute them using any of the execution mechanisms (Grunt Shell, UDFs, Embedded). After execution, these scripts will go through a series of transformations applied by the Pig Framework, to produce the desired output.

Internally, Apache Pig converts these scripts into a series of MapReduce jobs, and thus, it makes the programmer's job easy. The architecture of Apache Pig is shown below.

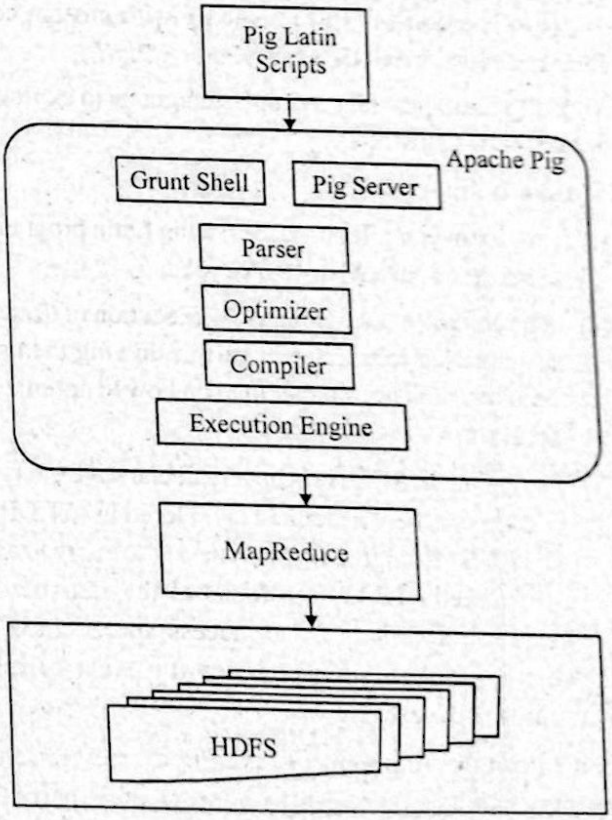

Fig.

Q.29 ***Explain working through the ABC of Pig Latin.***

Ans. Working through the ABC of Pig Latin :

- Pig Latin is the language for Pig progr: ns.
- Pig translates the Pig Latin script into MapReduce jobs that can be executed within Hadoop cluster.
- Pig Latin development team followed three key design principles:

1. Keep it simple

(i) Pig Latin is an abstraction for MapReduce that simplifies the creation of parallel programs on the Hadoop cluster for data flows and analysis.

(ii) Complex tasks may require a series of interrelated data transformations – such series are encoded as data flow sequences.

(iii) Writing Pig Latin scripts instead of Java MapReduce programs makes these programs easier to write, understand, and maintain because :

(a) you don't have to write the job in Java.

(b) you don't have to think in terms of MapReduce.

(c) you don't need to come up with custom code to support rich data types.

(iv) Pig Latin provides a simpler language to exploit your Hadoop cluster.

2. Make it smart

(i) Pig Latin compiler transform a Pig Latin program into a series of Java MapReduce jobs.

(ii) 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 data.

(iii) For example, SQL is set up as a declarative query that you use to access structured data stored in an RDBMS. The RDBMS engine first translates the query to a data access method and then looks at the statistics and generates a series of data access approaches. The cost-based optimizer chooses the most efficient approach for execution.

3. Don't limit development : Make Pig extensible so that developers can add functions to address their particular business problems.

(i) Traditional RDBMS data warehouses make use of the ETL data processing pattern, where you extract data from outside sources, transform it to fit your operational needs, and then load it into the end target, whether it's an operational data store, a data warehouse, or another variant of database.

(ii) With big data, the language for Pig data flows goes with ELT instead: Extract the data from your various sources, load it into HDFS, and then transform it as necessary to prepare the data for further analysis.

Q.30 ***Write a detailed note on scripting with Pig Latin.***

Ans. 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 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 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 functions (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. It also helps overcome limitations in the Pig language. One of the most commonly referenced limitations is that Pig doesn't support control flow statements: if/else, while loop, for loop, and condition statements. Pig natively supports data flow, but needs to be embedded within another language to provide control flow. There are tradeoffs, however of embedding Pig in a control-flow language. For example if a Pig statement is embedded in a loop, every time the loop iterates and runs the Pig statement, this causes a separate MapReduce job to run.

Working with Scripts : Writing Pig Latin scripts is largely about packaging together the Pig Latin statements that you've successfully tested in Grunt. Pig scripting does have a few unique topics though. They're comments, parameter substitution, and multiquery execution.

1. Comments : As you'll reuse your Pig Latin script, it's obviously a good idea to leave comments for other people (or yourself) to understand it in the future. Pig Latin supports two forms of comments, single-line and multiline. You start the single-line comment by a double hyphen and the comment ends at the end of the line. You enclose the multiline comment by the /* and */ markers, similar to multiline comments in Java. For example, a Pig Latin script with comments can look like

When you write a reusable script, it's generally parameterized such that you can vary its operation for each run. For example, the script may take the file paths of its input and output from the user each time. Pig supports parameter substitution to allow the user to specify such information at runtime. it denotes such parameters by the $ prefix within the script. For example, the following script displays a user-specified number of tuples from a user-specified log file:

```
1 log = LOAD '$input' AS (user, time, query); lmt = LIMIT log $size;DUMP lmt;
```

copy

2. Multiquery Execution : In the Grunt shell, a DUMP or STORE operation processes all previous statements needed for the result. On the other hand, Pig optimizes and processes an entire Pig script as a whole. This difference would have no effect at all if your script has only one DUMP or STORE command at the end. If your script has multiple DUMP/ STORE, Pig script's multiquery execution improves efficiency by avoiding redundant evaluations. For example, let's say you have a script that stores intermediate data:

```
1 a = LOAD...b = some transformation of aSTORE b...c = some further transformation
```

copy

❑❑❑