Apache Pig

COMP9313: Big Data Management

What is Apache Pig?

- It is a tool/platform which is used to analyze larger sets of data representing them as data flows
- Pig is an integral component of Hadoop eco system.
- Some History: in **2006**, Apache Pig was developed as a research project at Yahoo, especially to create and execute MapReduce jobs on every dataset. In **2007**, Apache Pig was open sourced via Apache incubator. In **2008**, the first release of Apache Pig came out. In **2010**, Apache Pig graduated as an Apache top-level project.

Where does the Name Pig come from?

- Pigs eat anything
- Pigs live anywhere
- Pigs are domestic animals
- Pigs fly

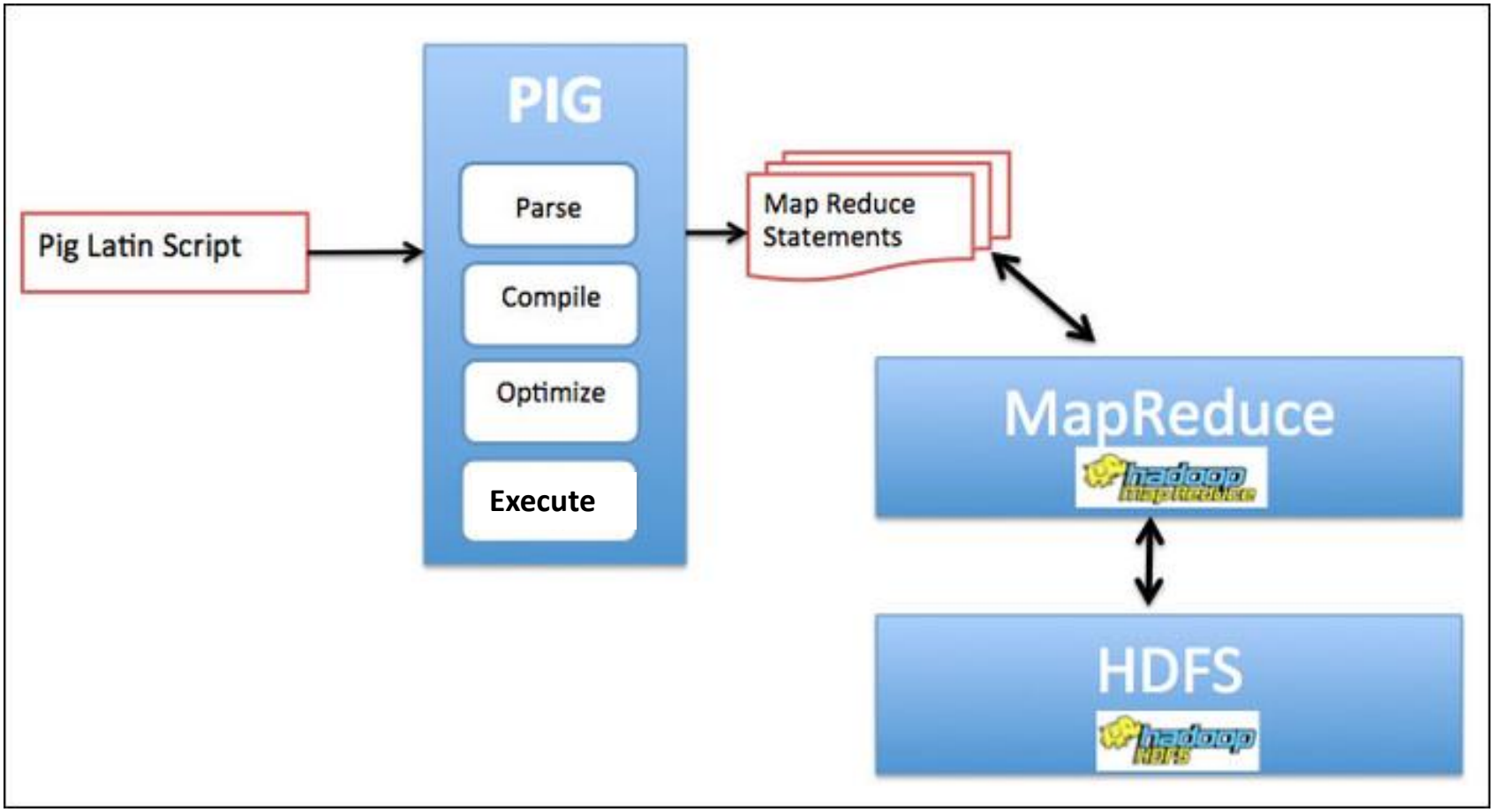
Why Use Pig?

- 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.

Pig's Features

- **Rich set of operators**: In order to perform several operations, Pig offers many operators. Such as join, sort, filer and many more.
- **Ease of programming**: Since you are good at SQL, it is easy to write a Pig script. Because of Pig Latin as same as SQL.
- **Optimization opportunities**: In Apache Pig, all the tasks optimize their execution automatically. As a result, the programmers need to focus only on the semantics of the language.
- **Extensibility**: Through Pig, it is easy to read, process, and write data. It is possible by using the existing operators. Also, users can develop their own functions.
- **UDFs**: By using Pig, we can create User-defined Functions in other programming languages like Java. Also, can invoke or embed them in Pig Scripts.
- **Handles all kinds of data**: Pig generally analyzes all kinds of data. Even both structured and unstructured. Moreover, it stores the results in **HDFS**.

The Architecture of Apache Pig

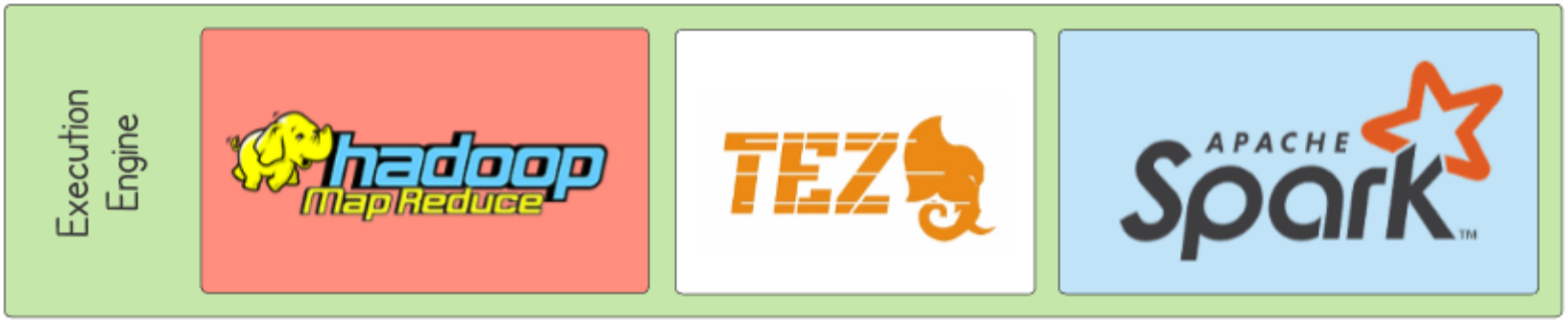


The Architecture of Apache Pig

- **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. These MapReduce jobs are executed on Hadoop producing the desired results.

Can Execution Engine be something other than MapReduce?

- Pig execution Modes:
  - Local Mode
  - MapReduce Mode
  - Tez Mode
  - Spark Mode

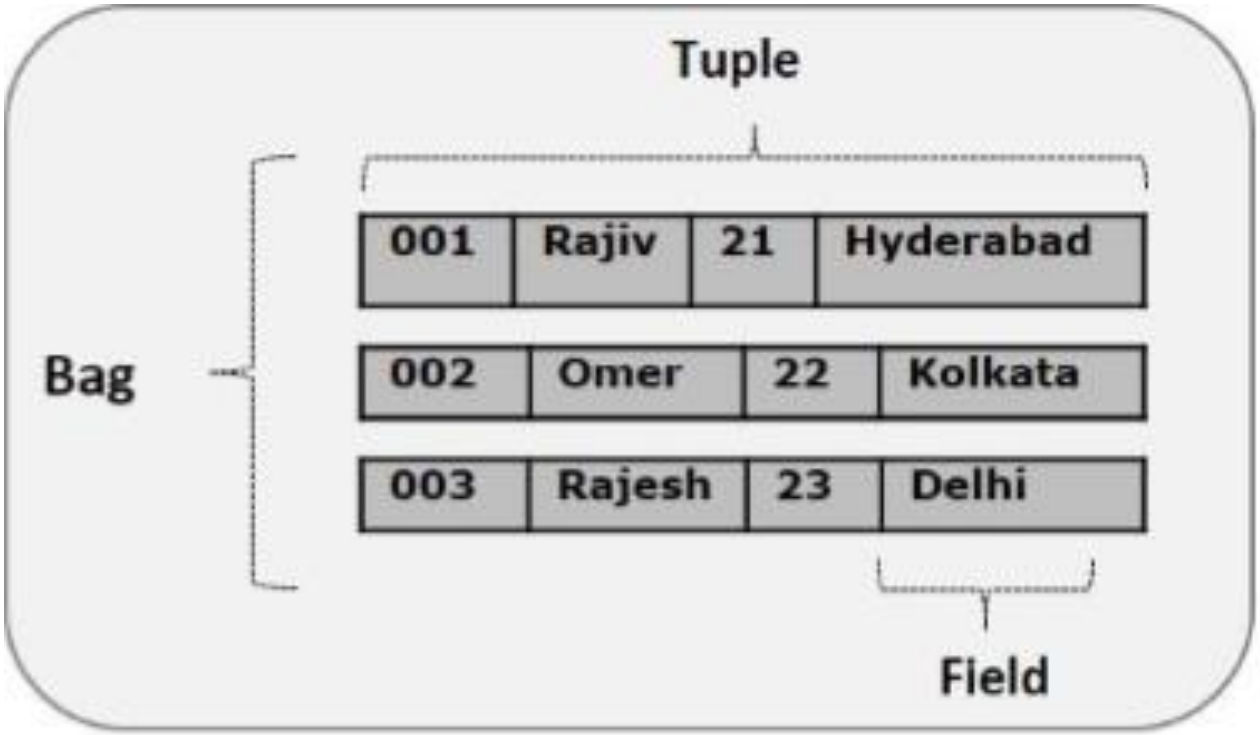


Pig on Spark and Tez

- Pig will convert the logic expressed in a Pig Latin code into a data structure known as Directed Acyclic Graphs (DAGs)
- When creating a DAG, Pig will ensure that the steps are ordered in a way that optimizes computation.
- For example, let's say we have a script that contains a JOIN operation followed by a FILTER step. Pig will investigate whether the FILTER step can be done before the JOIN step. If yes, Pig will perform the FILTER step prior to the JOIN step, thus reducing the number of data points to be joined (JOIN is a computationally expensive operation!) Because of the power of DAGs, Pig running on Tez or Spark always leads to better performance compared to MapReduce

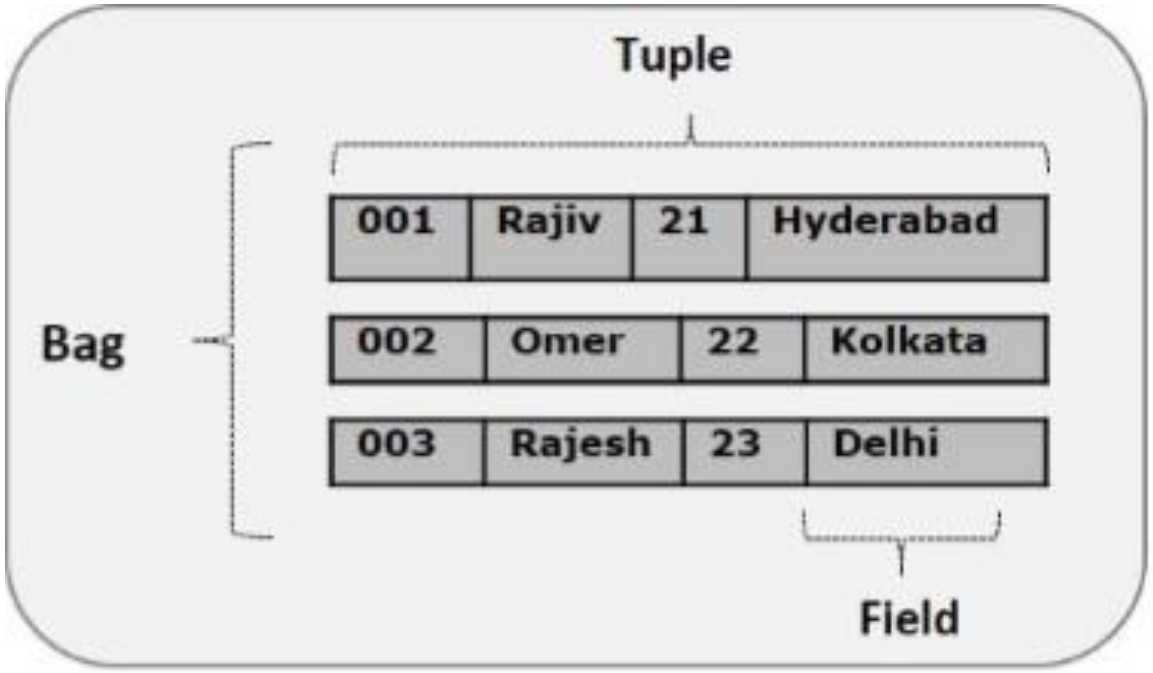
Pig Data Model

- Atom
- Tuple
- Bag
- Map



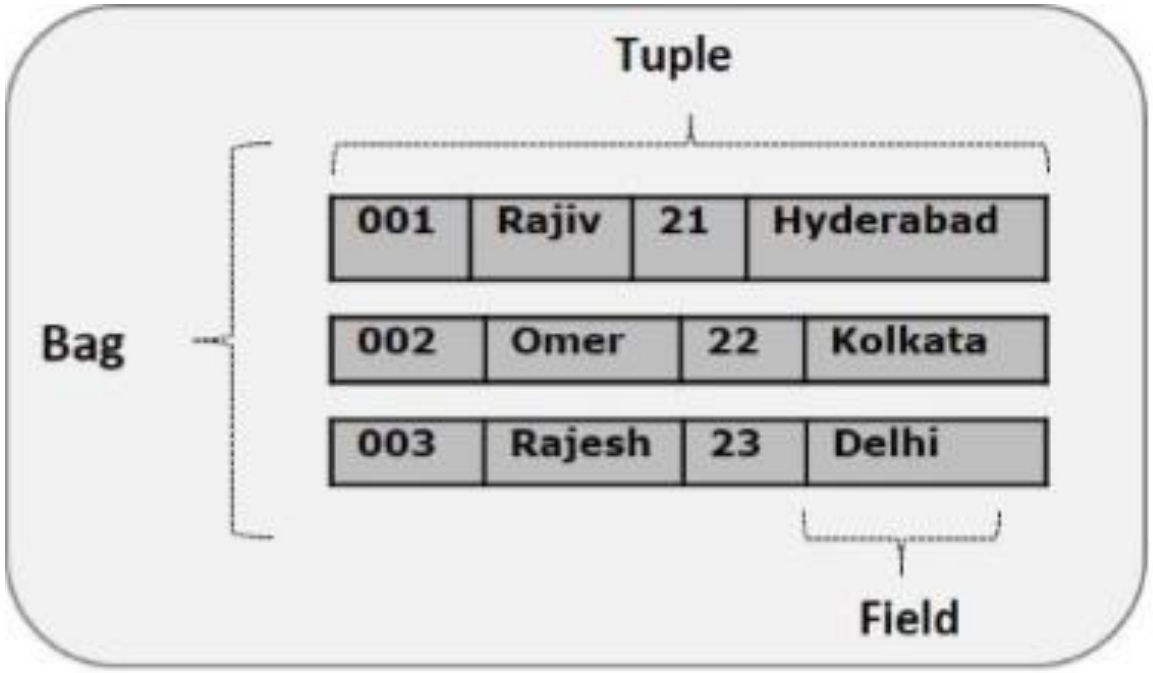
Atom

- Any single value in Pig Latin, irrespective of their data, type is known as an **Atom**. It is stored as string and can be used as string and number. int, long, float, double, chararray, and bytearray are the atomic values of Pig. A piece of data or a simple atomic value is known as a **field**.
- **Example** – 'raja' or '30'



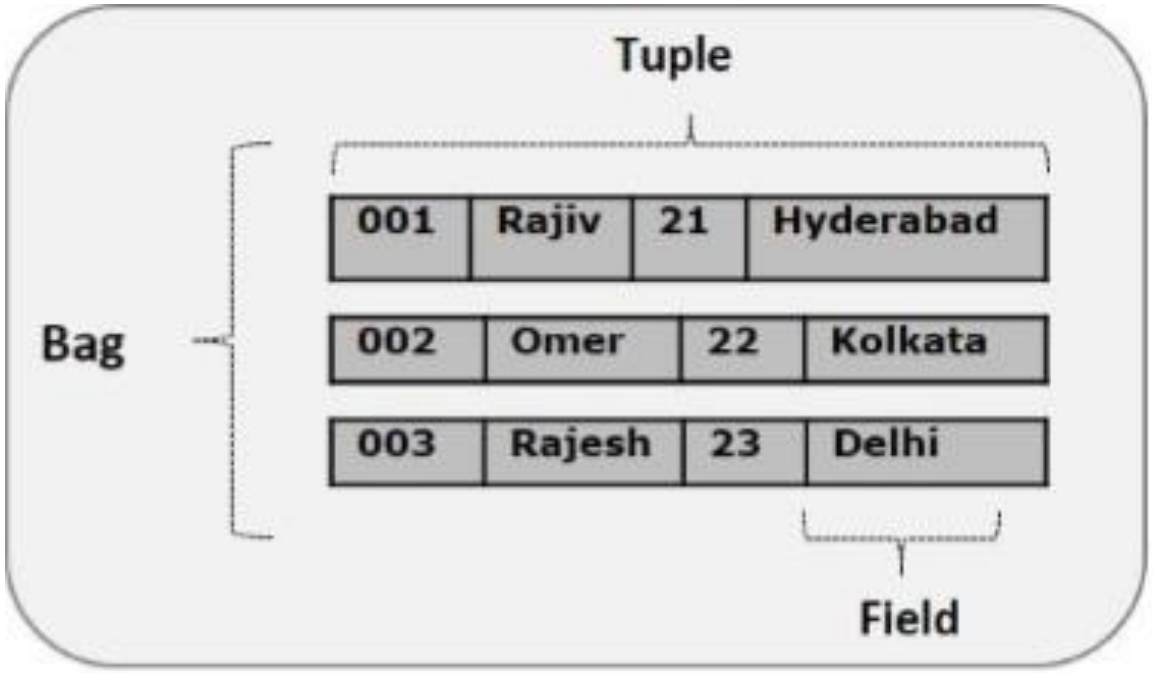
Tuple

- **Tuple:** A record that is formed by an ordered set of fields is known as a tuple, the fields can be of any type. A tuple is similar to a row in a table of RDBMS.
- **Example** – (Raja, 30)



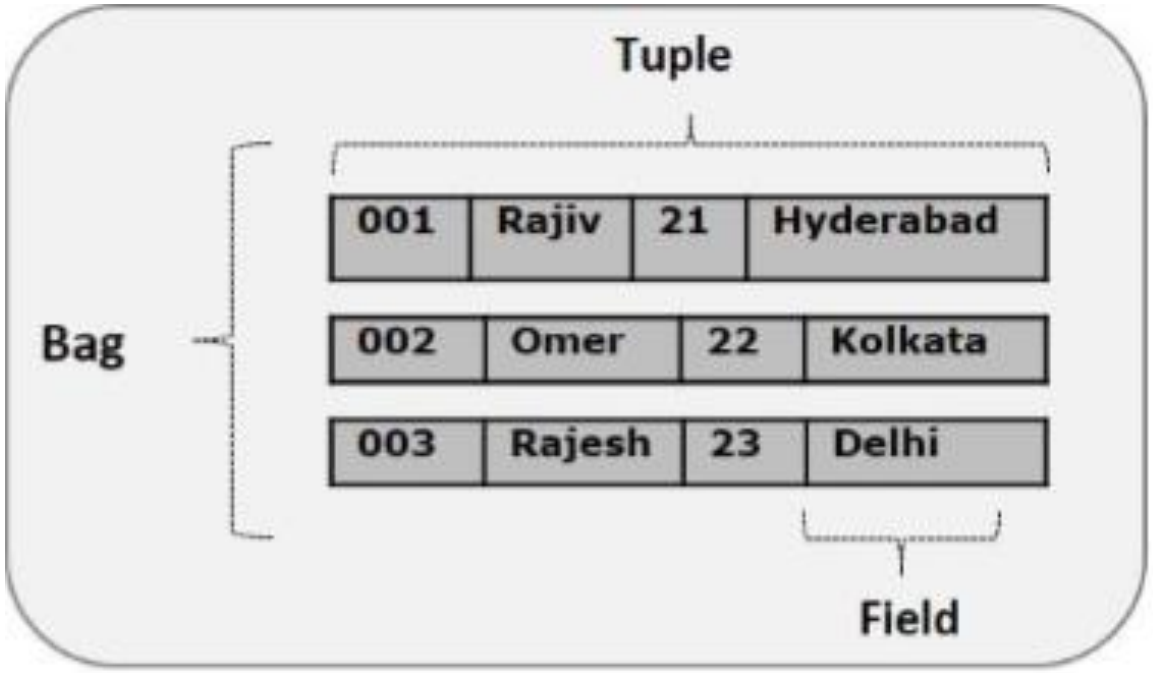
Bag

- **Bag:** A bag is an unordered set of tuples. In other words, a collection of tuples (non-unique) is known as a bag. Each tuple can have any number of fields (flexible schema). A bag is represented by '{}'. It is similar to a table in RDBMS, but unlike a table in RDBMS, it is not necessary that every tuple contain the same number of fields or that the fields in the same position (column) have the same type.
- **Example** – {(Raja, 30), (Mohammad, 45)}



Map

- **Map:** A map (or data map) is a set of key-value pairs. The **key** needs to be of type chararray and should be unique. The **value** might be of any type. It is represented by '[]'
- **Example** – [name#Raja, age#30]



Relation

- Relation: Relation represents the complete database. A relation is outer bag. We can call a relation as a bag of tuples.
- A Pig relation is similar to a table in a relational database, where the tuples in the bag correspond to the rows in a table.
- Unlike a relational table, however, Pig relations don't require that every tuple contain the same number of fields or that the fields in the same position (column) have the same type.

Grunt

- Grunt is Pig command shell used to write Pig Latin scripts.
- The Grunt shell provides a set of utility commands. These include utility commands such as **clear, help, history, quit,** and **set**; and commands such as **exec, kill,** and **run** to control Pig from the Grunt shell

Pig Latin

- Pig Latin is a Dataflow programming language (i.e. it is limited on control flow commands)
- A Pig Latin program consists of a series of operations or transformations which are applied to the input data to produce output. These operations describe a data flow which is translated into an executable representation, by Pig execution environment.
- Underneath, results of these transformations are series of MapReduce jobs which a programmer is unaware of.
- Pig allows the programmer to focus on data rather than the nature of execution.

Pig Latin Statements

- Pig Latin statements are the basic constructs you use to process data using Pig.
- A Pig Latin statement is an operator that takes a relation as input and produces another relation as output. (This definition applies to all Pig Latin operators except LOAD and STORE which read data from and write data to the file system.)
- Pig Latin statements may include expressions and schemas. Pig Latin statements can span multiple lines and must end with a semi-colon (;).

Pig WordCount example

```
lines = LOAD '/user/hadoop/HDFS_File.txt' AS (line:chararray);

words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word;

grouped = GROUP words BY word;

wordcount = FOREACH grouped GENERATE group, COUNT(words);

DUMP wordcount;
```

Questions?