Recap

- Impala ad-hoc query processing tool
- Hive – Data warehousing solution on Hadoop
- MapReduce optimization
- MapReduce chaining

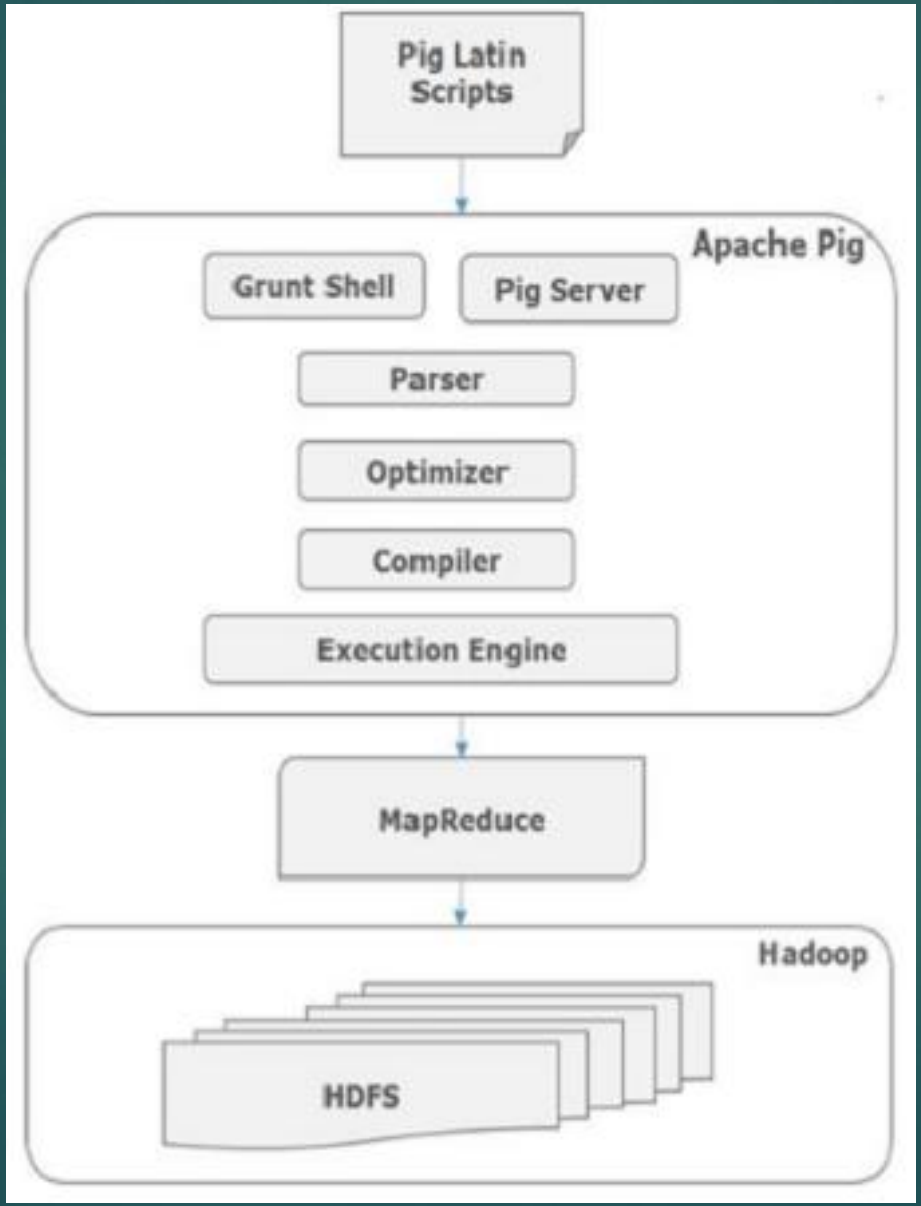
Agenda for today

- Pig
- Hive
- Impala

Introduction

- High Level Scripting Language developed by Yahoo originally
- Transforms SQL like language called Pig Latin into Java code
- Follows lazy evaluation
- Supports UDF written in multiple languages

Architecture



Execution

- Accessing approaches:

1. Batch mode: submit a script directly
2. Interactive mode: Grunt, the pig shell
3. PigServer for Java program

- Execution mode:

1. Local mode: pig –x local
2. Mapreduce mode (default): pig –x mapreduce

Applications

- Process web logs
- Build user behavior models
- Process images
- Build maps of the web
- Do research on large data sets

Data types

- Scalar Types: Int, long, float, double, boolean, null, chararray, bytearray
- Complex Types: fields, tuples, bags, relations

Operator: LOAD

- To load data from storage system

```
lines=LOAD ‘myfile’ AS (line: chararray);
```

- Supports various loader formats

1. PigStorage
2. TextLoader
3. BinStorage

Operator: LOAD cont…

- Load data without schema

```
relXYZ = LOAD ‘yourfile.csv' USING PigStorage(',');
```

- Load data with schema

```
relXYZ = LOAD ‘yourfile.csv' USING PigStorage(',') as
(col1:datatype, col2:datatype,…);
```

Operator: LIMIT

- To take sample records

```
New_Rel = LIMIT  RelationName <Sample Count>;
```

Operator: DUMP

- Print the data on console

```
DUMP RelationName;
```

Operator: FOREACH

- Select specific columns

```
New_Rel = FOREACH RelationName GENERATE
driverId, eventTime, eventType;
```

Operator: JOIN

- Joins two relations/datasets

```
join_data = JOIN  relation1 BY (column1), relation2
BY (column1);
```

Operator: SORT

- Sort a relation based on key

```
New_rel = ORDER RelationName BY ColumnName
asc;
```

Operator: FILTER

- Filter the dataset

```
New_rel = FILTER RelationName BY (Condition);
```

Operator: DISTINCT

- Remove duplicates

```
New_rel = DISTINCT RelationName;
```

Operator: STORE

- Store the output

```
STORE relationName INTO 'output_directory' USING PigStorage(',');
```

PigServer API

```
import java.io.IOException;
import org.apache.pig.PigServer;
public class idlocal{
public static void main(String[] args) {
    try {
      PigServer pigServer = new PigServer("local");
      runIdQuery(pigServer, "passwd");
      }
      catch(Exception e) {}
      }
    public static void runIdQuery(PigServer pigServer, String inputFile) throws IOException {
      pigServer.registerQuery("A = load '" + inputFile + "' using PigStorage(':');");
      pigServer.registerQuery("B = foreach A generate $0 as id;");
      pigServer.store("B", "id.out");
}}
```

UDF

- Prepare a Jar file
- Register the Jar
- Define alias
- Use it

https://www.tutorialspoint.com/apache_pig/apache_pig_user_defined_functions.htm