## Recap

- Hadoop installation
- HDFS Java API

## Agenda for today

- MapReduce counters
- Performance tuning in MapReduce jobs
- MapReduce job chaining
- Pig
- Java programming & SQL for beginners

## Performance tuning

- Cluster configuration
- Use compression technique
- Tuning # mappers and reducers
- Use combiner
- Appropriate data type
- Reuse objects
- Profiling

https://blog.cloudera.com/blog/2009/12/7-tips-for-improving-mapreduce-performance/

## MapReduce Job chaining

- Two separate jobs
- Multiple mappers/reducers within same job

## MapReduce Job chaining

- Two separate jobs

1. Configure first job object and run it.
2. Configure second job object and run it

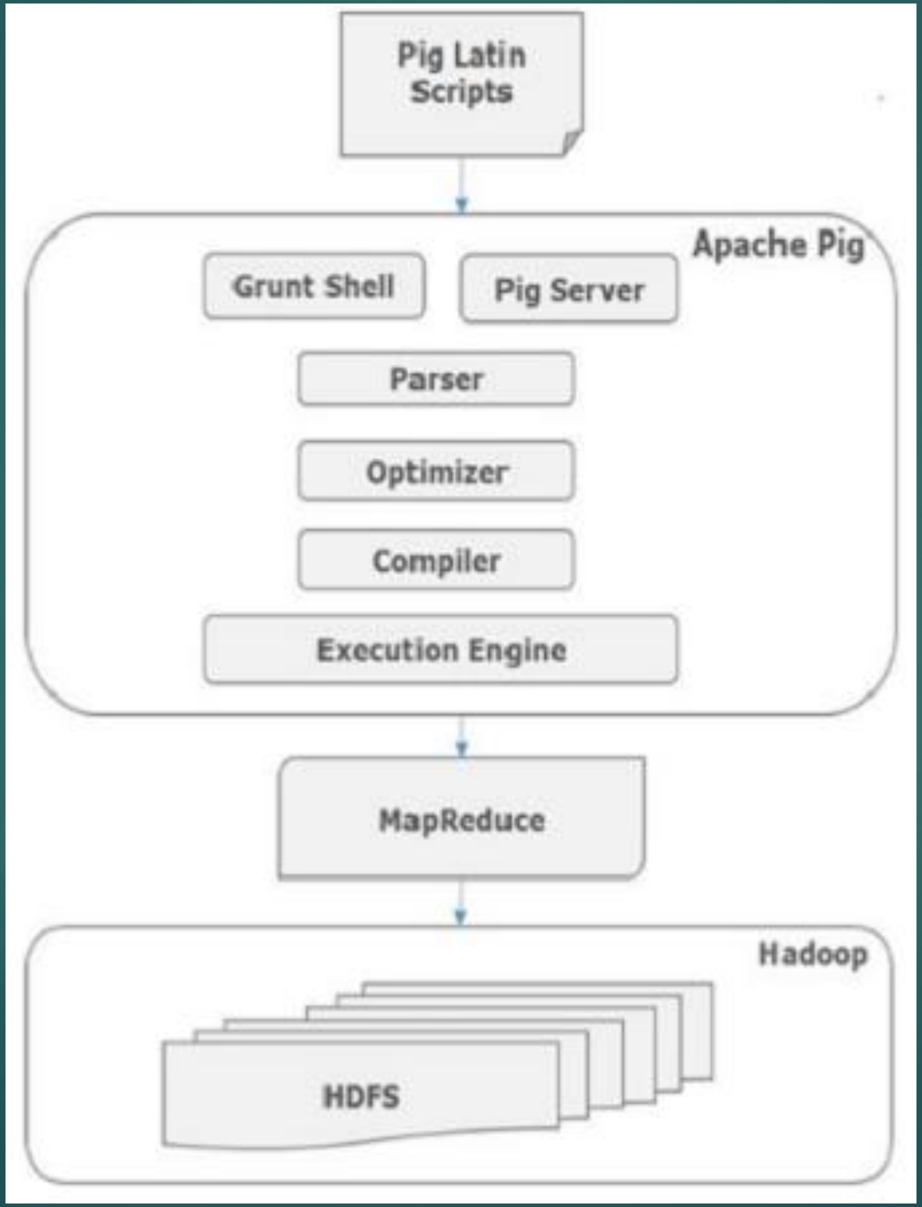
# MapReduce Job chaining

- Multiple mappers/reducers within same job
  - ChainMapper API: to add multiple mappers
  - ChainReducer API: to add multiple reducers

## 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

## 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);

```
books = LOAD ‘/data/pig/books.csv’ as (line: chararray)
```

# Operator: DUMP

- Print the data on console

```
DUMP RelationName;
```

```
DUMP sample_books;
```

## Various loaders

- Supports various loader formats

1. TextLoader
2. PigStorage
3. JsonLoader & JsonStorage
4. BinStorage
5. HBaseStorage
6. OrcStorage
7. MongoStorage

# Operator: LOAD cont…

- Load data without schema

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

books = LOAD '/data/pig/books.csv' USING PigStorage(',');

- Load data with schema

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

books = LOAD '/data/pig/books.csv' USING PigStorage(',') as (id:int, author:chararray, name:chararray, year:int);

# Operator: LIMIT

- Take sample records

New_Rel = LIMIT  RelationName <Sample Count>;

Sample_books = LIMIT books 5;

# Operator: FOREACH

- Select specific columns

```
New_Rel = FOREACH RelationName GENERATE col1, col2, col3….;
```

```
book_no_author = FOREACH books GENERATE id, name, year;
```

## Operator: JOIN

- Joins two relations/datasets

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

```
book_review = JOIN books BY (id), reviews BY (id)
```

# Operator: SORT

- Sort a relation based on key

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

```
books_sorted_by_year = ORDER books BY year asc;
```

## Operator: FILTER

- Filter the dataset

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

```
books_before_2000 = FILTER books BY (year < 2000)
```

## Operator: DISTINCT

- Remove duplicates

```
New_rel = DISTINCT RelationName;
```

```
bedupe = DISTINCT books_before_2000;
```

## Aggregate

- Aggregate based on a key

```
GroupRel = GROUP RelName BY columnName;

AggRel = FOREACH GroupRel GENERATE group ,
AVG(columnName)
```

```
group_review = group book_review by books::id;
avg_rating = foreach group_review generate group as id,
AVG($1.reviews::rating)
```

# Operator: STORE

- Store the output

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

```
STORE dedupe INTO ‘/data/pig/dedupe' 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

# Further reading

- Map Reduce job chaining:

https://mapr.com/blog/how-to-launching-mapreduce-jobs/

- Pig inbuilt functions

https://pig.apache.org/docs/latest/func.html