



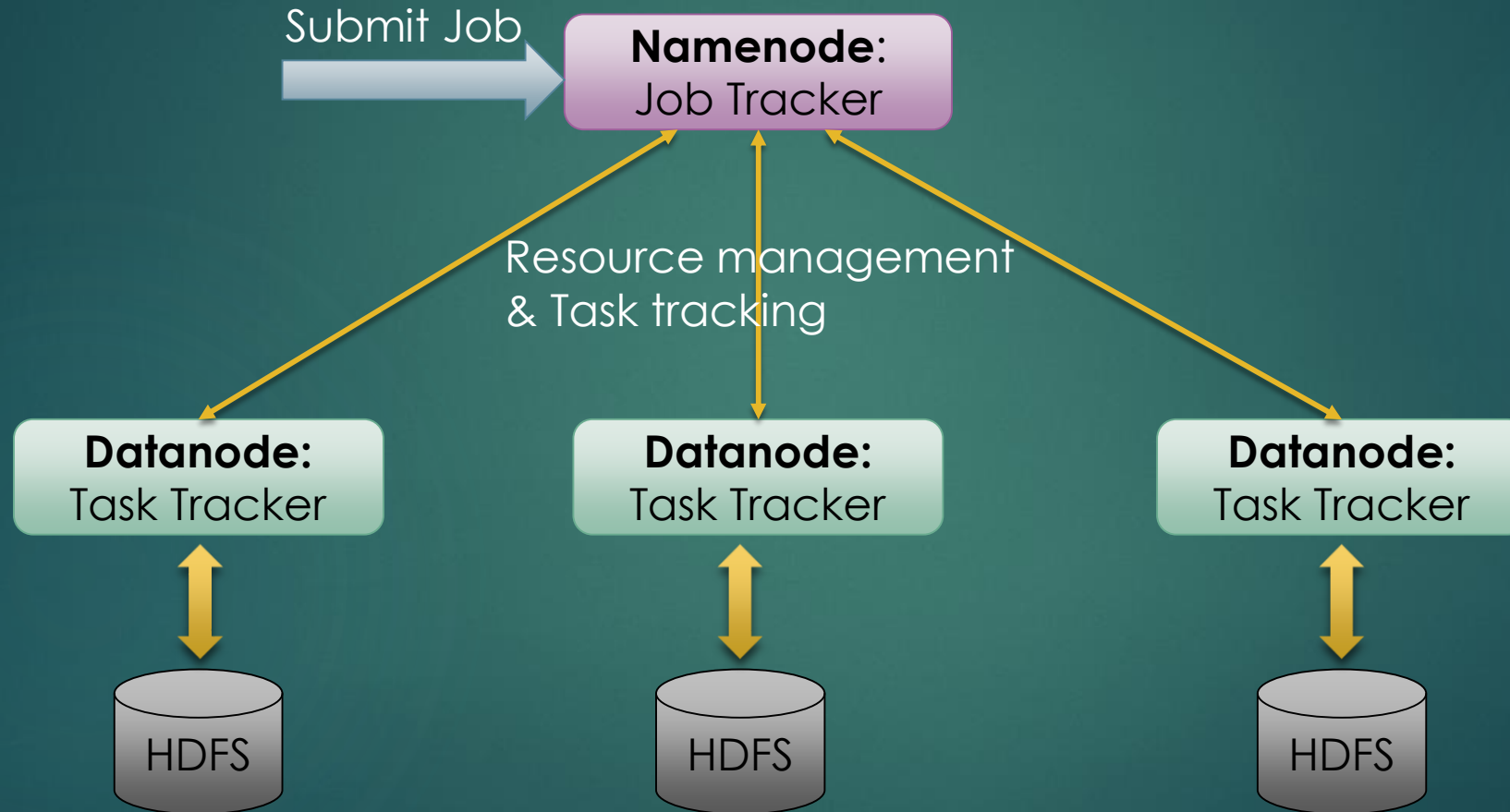
# Recap

2

- ▶ HDFS filesystem
  - Block
  - Read and Write operations
  - CLI commands
  - Network topology and rack awareness
- ▶ MapReduce
- ▶ Hadoop 1 vs Hadoop 2
- ▶ YARN

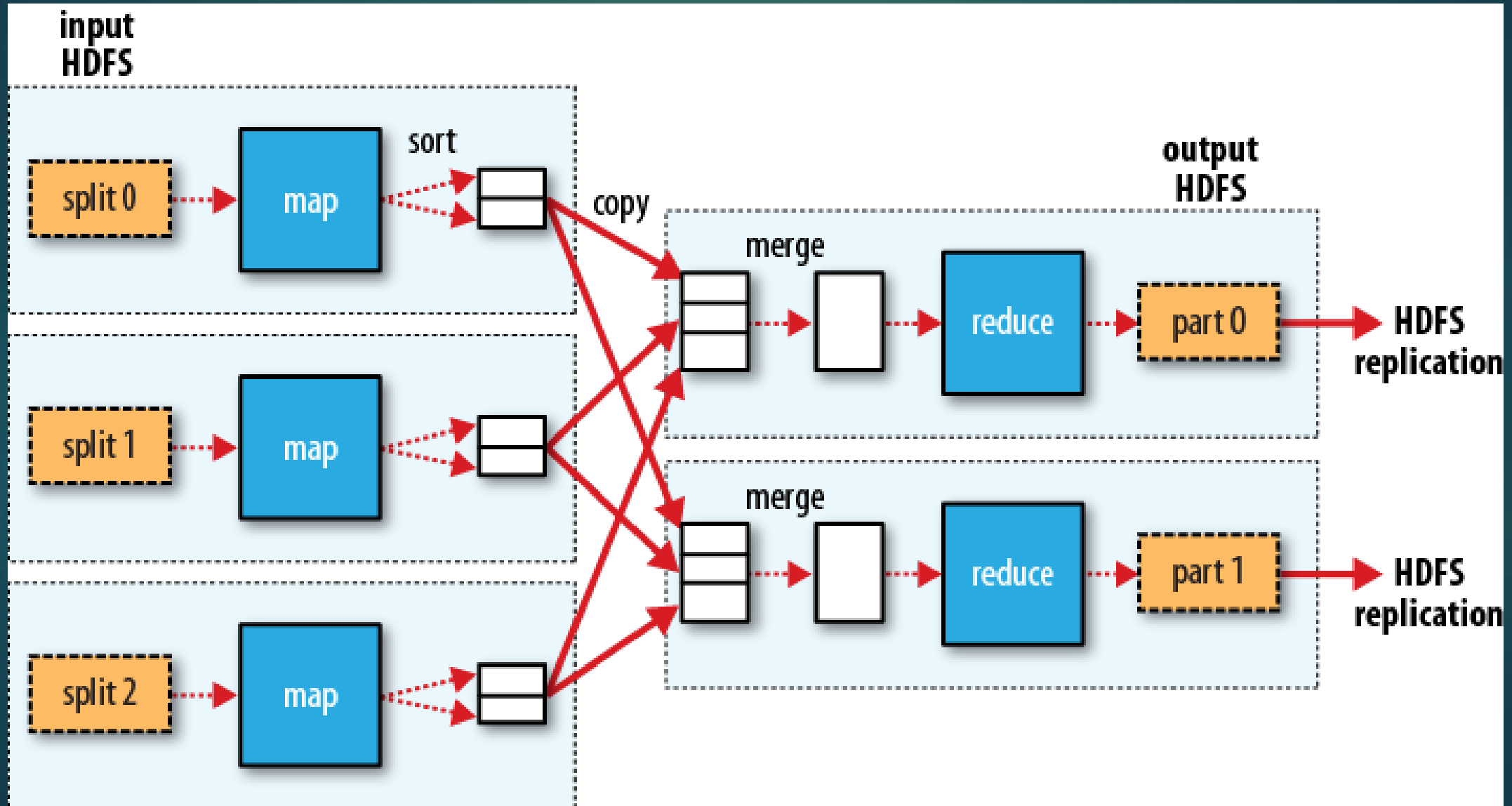
# Job management

3



# MapReduce Stages

4





# Mapreduce: working example

5

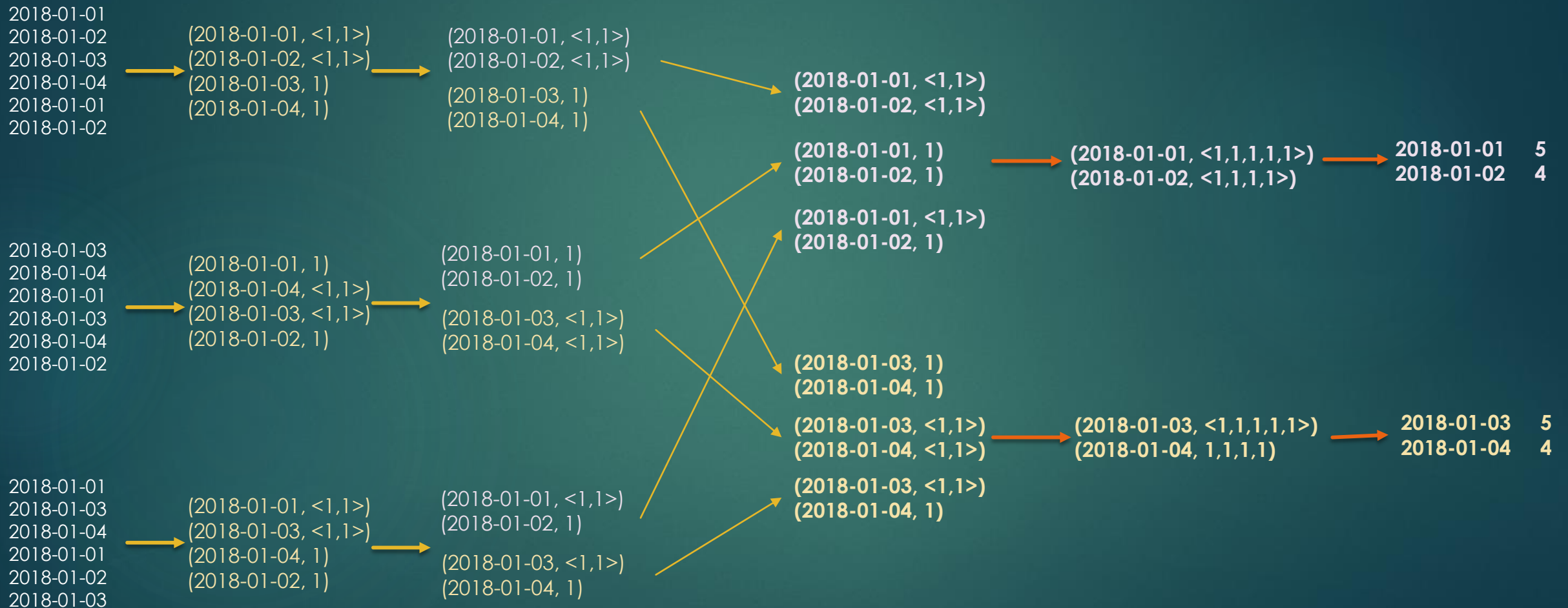
Map

Partition & Sort

Shuffle

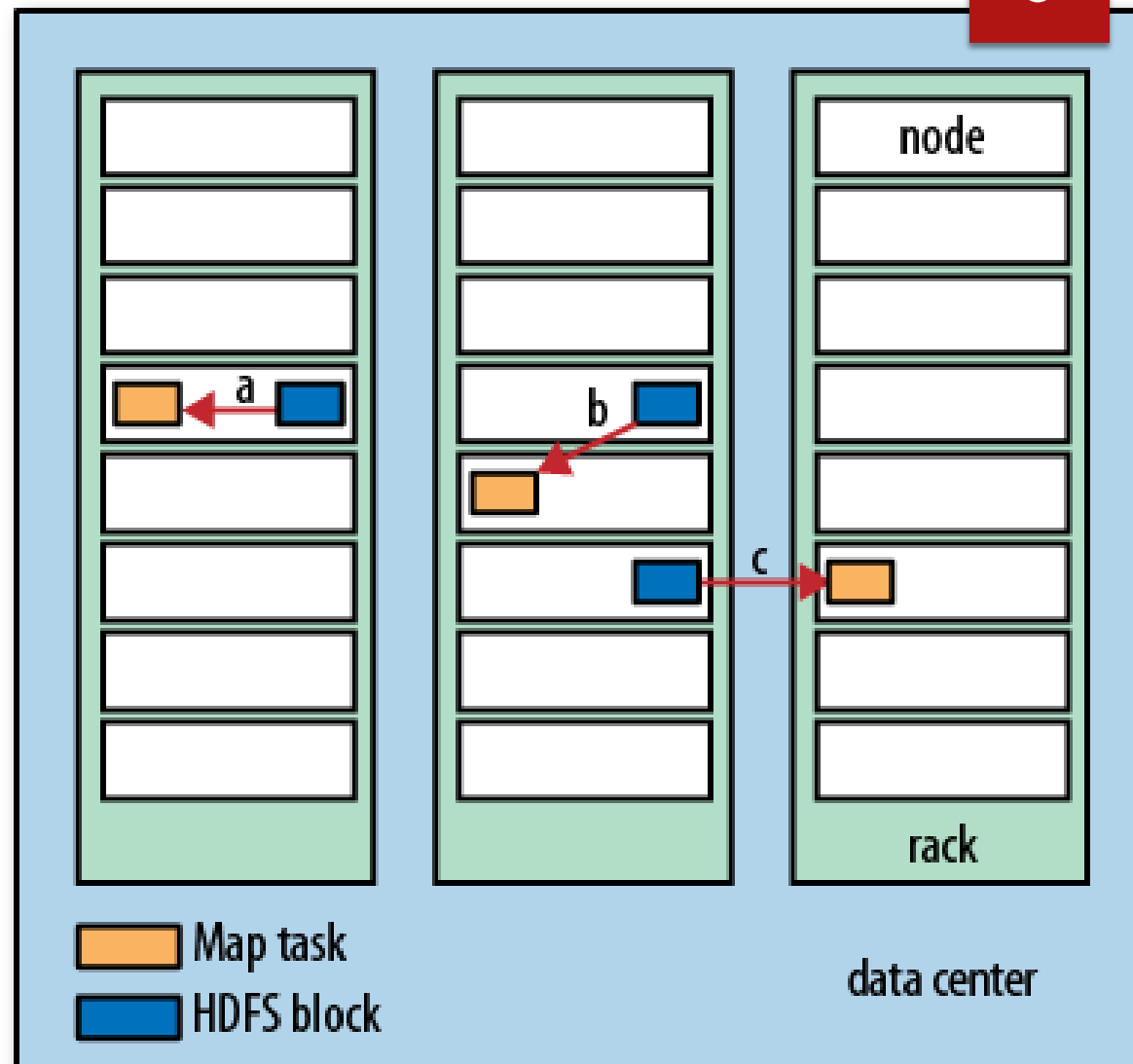
Merge

Reduce



# Task to node mapping

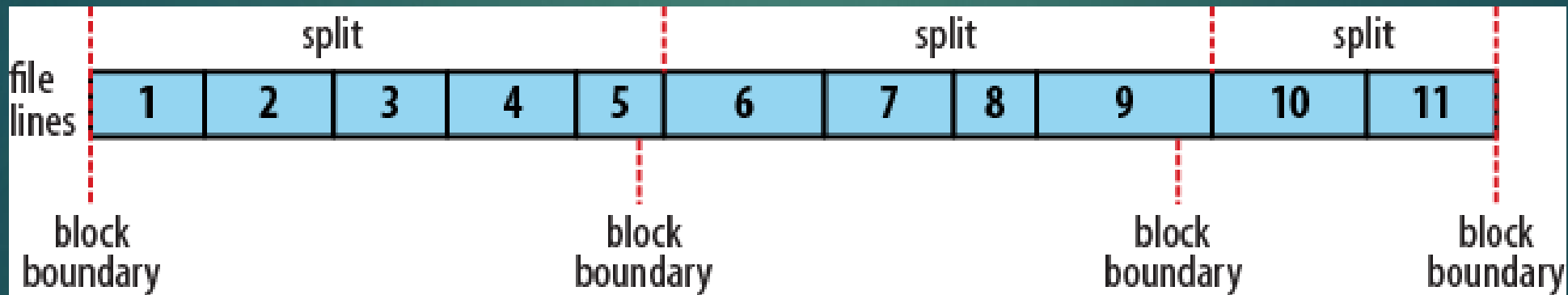
- Notion of data locality



# Input Splits

7

- ▶ Blocks are of fixed size
- ▶ Good chances of records being split between two block



# MapReduce: Mapper code

```
public class WebHitCounterMapper extends
Mapper<Input Key, Input Value, Output Key, Output Value>
{

    public void map(Input Key, Input Value, Context context)
throws IOException, InterruptedException {

        <MAP Logic goes here>

        context.write(Output Key, Output Value)
    }
}
```



# MapReduce: Reducer code

```
public class WebHitCounterReducer extends
Reducer<Input Key, Input Value, Output Key, Output Value>
{
    public void reduce(Input Key, Iterable<Value Data type>
values, Context context) throws IOException,
InterruptedException {

    <REDUCE logic goes here>

    context.write(Output Key, Output Value);
}
}
```

# MapReduce: Driver Code

10

```
public class WebHitCounterMain {  
    public static void main(String[] args) throws Exception {  
  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf, "Daily Web Hit Counter");  
  
    }  
}
```

# MapReduce: Driver Code

11

```
public class WebHitCounterMain {  
    public static void main(String[] args) throws Exception {  
  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf, "Daily Web Hit Counter");  
  
        job.setJarByClass(main.WebHitCounterMain.class);  
        job.setMapperClass mapper.WebHitCounterMapper.class);  
        job.setReducerClass(reducer.WebHitCounterReducer.class);  
  
    }  
}
```

# MapReduce: Driver Code

12

```
public class WebHitCounterMain {  
    public static void main(String[] args) throws Exception {  
  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf, "Daily Web Hit Counter");  
  
        job.setJarByClass(main.WebHitCounterMain.class);  
        job.setMapperClass(mapper.WebHitCounterMapper.class);  
        job.setReducerClass(reducer.WebHitCounterReducer.class);  
  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(IntWritable.class);  
  
    }  
}
```

# MapReduce: Driver Code

13

```
public class WebHitCounterMain {  
    public static void main(String[] args) throws Exception {  
  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf, "Daily Web Hit Counter");  
  
        job.setJarByClass(main.WebHitCounterMain.class);  
        job.setMapperClass mapper.WebHitCounterMapper.class);  
        job.setReducerClass(reducer.WebHitCounterReducer.class);  
  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(IntWritable.class);  
  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
  
    }  
}
```

# MapReduce: Driver Code

14

```
public class WebHitCounterMain {  
    public static void main(String[] args) throws Exception {  
  
        Configuration conf = new Configuration();  
        Job job = Job.getInstance(conf, "Daily Web Hit Counter");  
  
        job.setJarByClass(main.WebHitCounterMain.class);  
        job.setMapperClass(mapper.WebHitCounterMapper.class);  
        job.setReducerClass(reducer.WebHitCounterReducer.class);  
  
        job.setOutputKeyClass(Text.class);  
        job.setOutputValueClass(IntWritable.class);  
  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
  
        System.exit(job.waitForCompletion(true) ? 0 : 1);  
    }  
}
```



# Performance tuning

15

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

# MapReduce Job chaining

16

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

# MapReduce Job chaining

17

## ▶ Two separate jobs

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

# MapReduce Job chaining

18

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

# Challenges with Hadoop 1

19

- ▶ Applications were limited to MapReduce implementations only
- ▶ Namenode machine crash or maintenance activity
- ▶ Namespace scaling
- ▶ Backup and Recovery
- ▶ Batch oriented architecture
- ▶ Support for various file formats
- ▶ Dual responsibilities of Job tracker

# Hadoop 2

20

- ▶ Support for other data processing engines
- ▶ High Availability
- ▶ HDFS Federation
- ▶ HDFS Snapshot
- ▶ Introduced Streaming and Interactive analysis
- ▶ Support for various file formats
- ▶ Yarn



# YARN

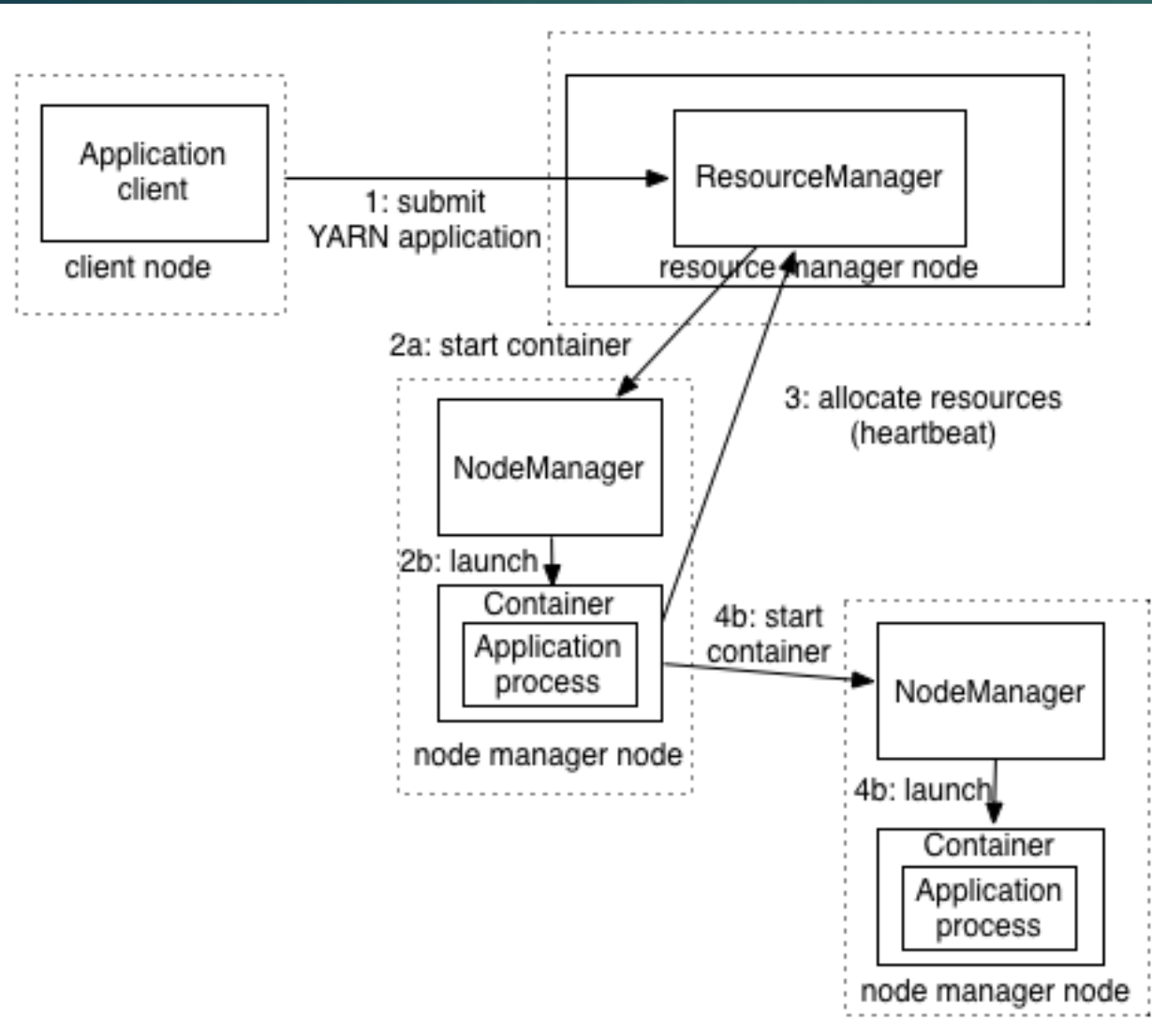
21

## ► Yet Another Resource Negotiator

MapReduce 1	YARN
Job Tracker	Resource Manager, Application Master and Timeline server
Task Tracker	Node Manager
Slot	Containers

# YARN model

22



# Agenda for today

23

- ▶ Pig: Mapreduce scripting
- ▶ Installing Hadoop on single node
- ▶ Routine SQL operations



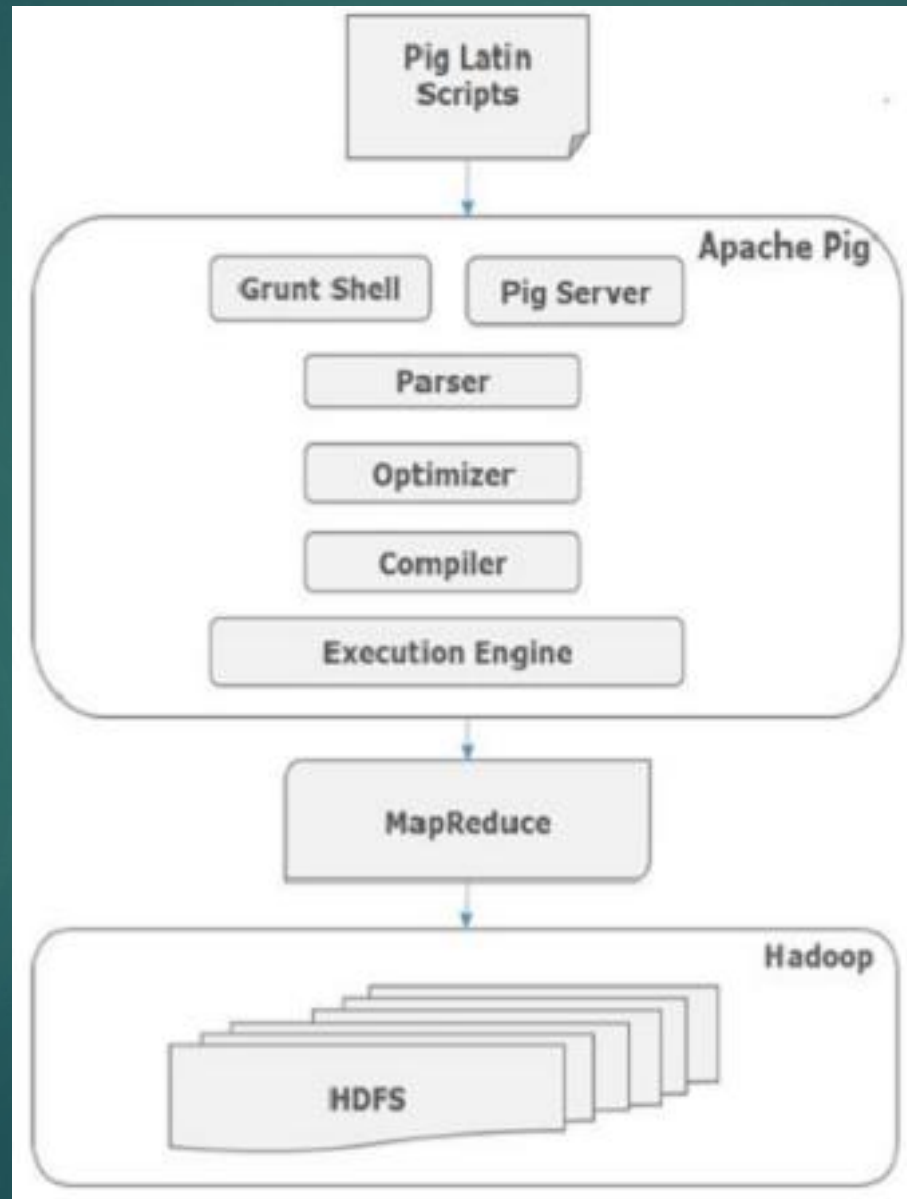
# Introduction

25

- ▶ 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

26





# Execution

27

## ▶ Accessing approaches:

1. Batch mode: Run a script file
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

28

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

# Various loaders

29

► Supports various loader formats

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

# Operator: LOAD

30

► To load data from storage system  
`lines=LOAD 'myfile' AS (line: chararray);`

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

# Operator: DUMP

31

- ▶ Print the data on console

DUMP RelationName;

DUMP sample\_books;

# Operator: LOAD cont...

32

- ▶ 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

33

- ▶ Take sample records

New\_Rel = LIMIT RelationName <Sample Count>;

Sample\_books = LIMIT books 5;

# Operator: FOREACH

34

- ▶ Select specific columns

New\_Rel = FOREACH RelationName GENERATE  
col1, col2, col3....;

book\_no\_author = FOREACH books GENERATE id, name, year;

# Operator: JOIN

35

- ▶ Joins two relations/datasets

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

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

# Operator: SORT

36

- ▶ Sort a relation based on key

New\_rel = ORDER RelationName BY ColumnName  
asc;

books\_sorted\_by\_year = ORDER books BY year asc;

# Operator: FILTER

37

- ▶ Filter the dataset

New\_rel = FILTER RelationName BY (Condition);

books\_before\_2000 = FILTER books BY (year < 2000)

# Operator: DISTINCT

38

- ▶ Remove duplicates

New\_rel = DISTINCT RelationName;

bedupe = DISTINCT books\_before\_2000;

# Aggregate

39

- ▶ Aggregate based on a key

GroupRel = GROUP RelName BY columnName;

group\_review = group book\_review by books::id;

AggRel = FOREACH GroupRel GENERATE group ,  
AVG(columnName)

avg\_rating = foreach group\_review generate group as id,  
AVG(\$1.reviews::rating)

# Operator: STORE

40

- ▶ Store the output

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

```
STORE dedupe INTO '/data/pig/dedupe' USING PigStorage(',');
```



# PigServer API

41

```
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

42

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

# Hadoop Installation

43

Standalone

Everything in one JVM. No HDFS installation

Pseudo-distributed

Mimic a distributed cluster on single physical machine

Distributed

Fully distributed cluster with multiple physical machines

# Reference

44

- ▶ Hadoop standalone vs pseudo-distributed

<https://stackoverflow.com/questions/23435333/what-is-the-difference-between-single-node-pseudo-distributed-mode-in-hadoop>

- ▶ Hadoop installation differences

<https://medium.com/@nidhinmahesh/getting-started-hadoop-mapreduce-hdfs-and-yarn-configuration-and-sample-program-febb1415f945>

- ▶ Download Ubuntu

<https://www.ubuntu.com/download/desktop>

- ▶ Hadoop installation step by step guide

[http://www.bogotobogo.com/Hadoop/BigData\\_hadoop\\_Install\\_on\\_ubuntu\\_16\\_04\\_single\\_node\\_cluster.php](http://www.bogotobogo.com/Hadoop/BigData_hadoop_Install_on_ubuntu_16_04_single_node_cluster.php)

- ▶ Map Reduce job chaining:

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

- ▶ Pig inbuilt functions

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

- ▶ Mapreduce job optimization

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

- ▶ Pig UDF

[https://www.tutorialspoint.com/apache\\_pig/apache\\_pig\\_user\\_defined\\_functions.htm](https://www.tutorialspoint.com/apache_pig/apache_pig_user_defined_functions.htm)