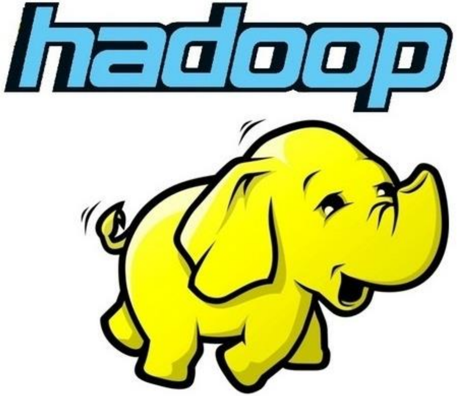
Big Data Engineering With MapReduce and Hive

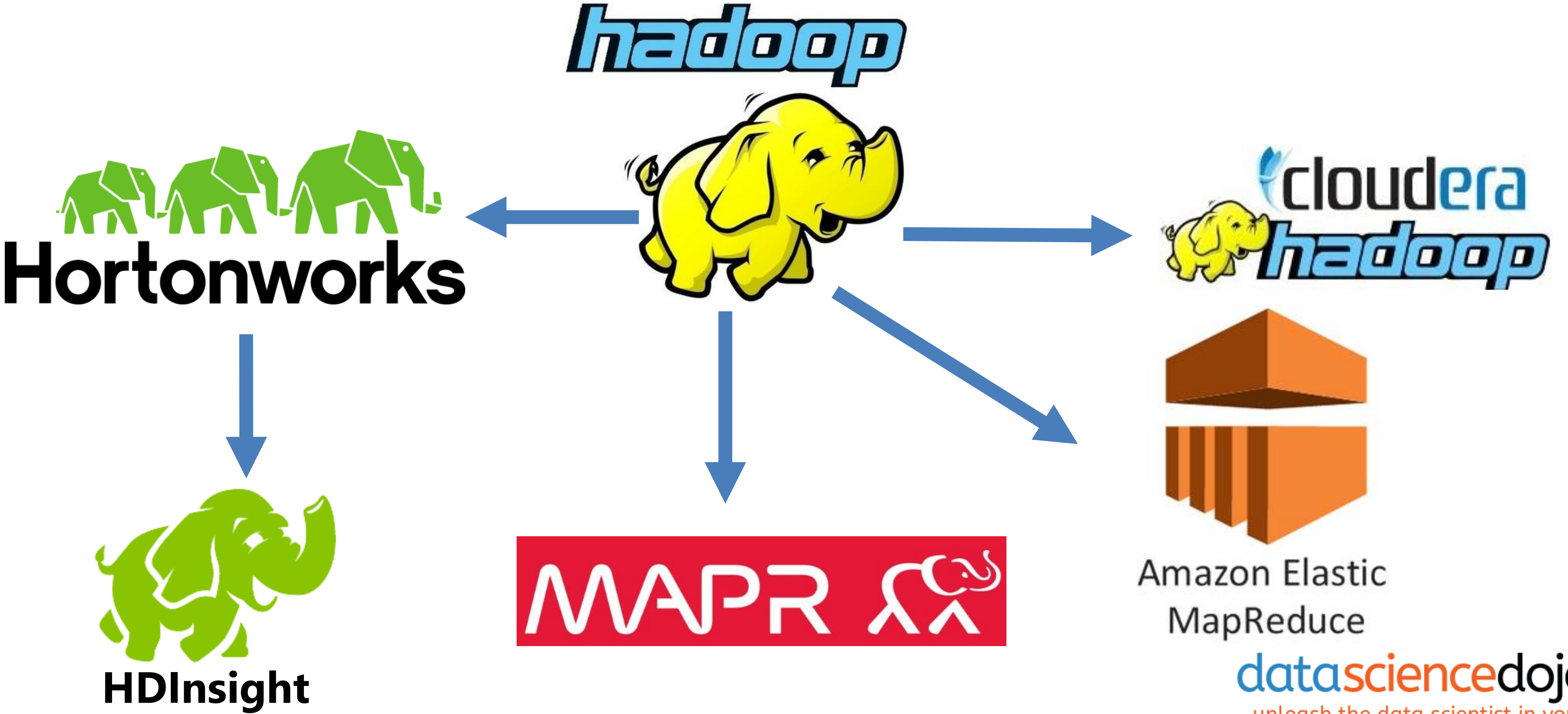
Data Science Dojo

datasciencedojo
unleash the data scientist in you

Agenda



Hadoop Implementations



hadoop

Hortonworks

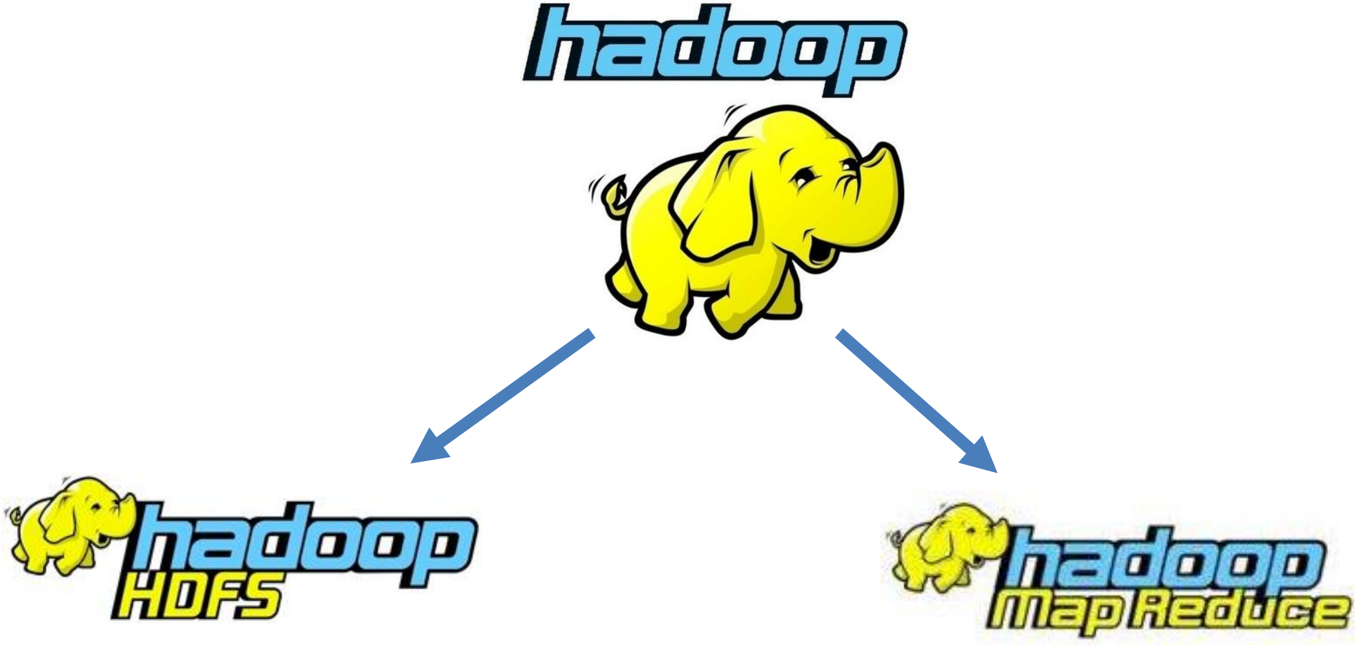
HDInsight

MAPR

cloudera hadoop

Amazon Elastic MapReduce

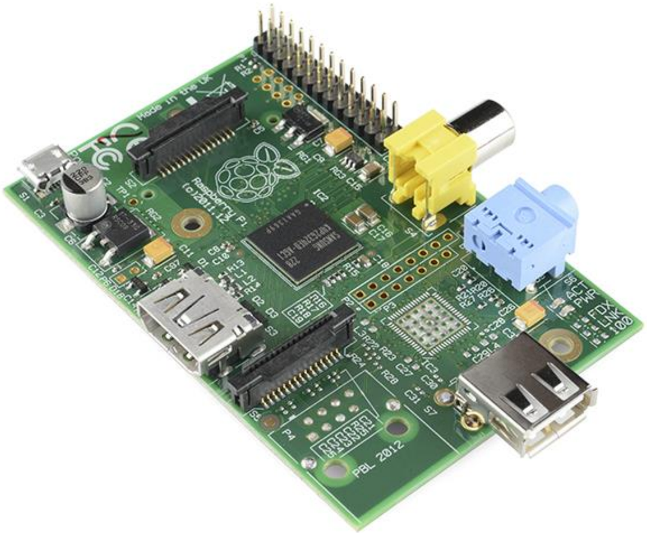
Hadoop

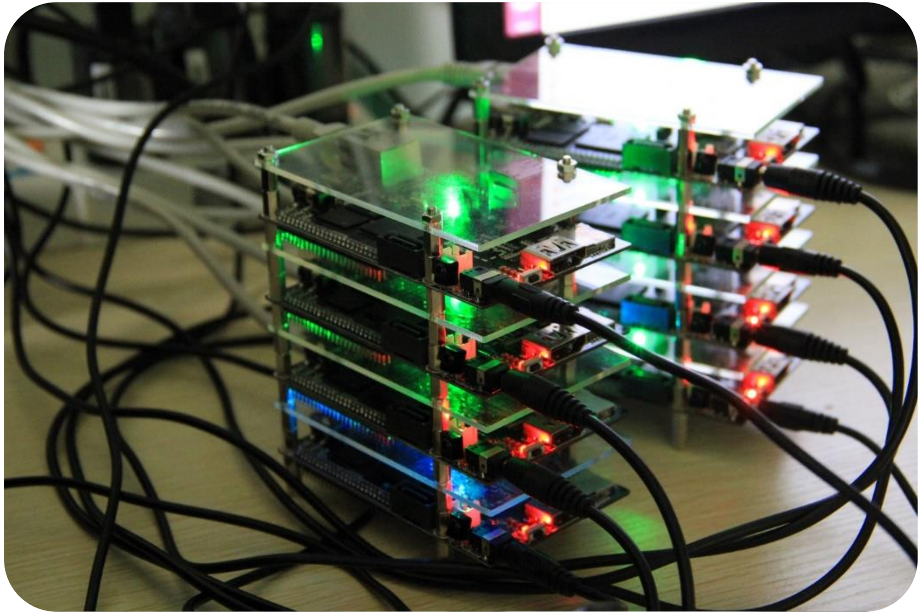


Turn Back The Clock, The Mainframe

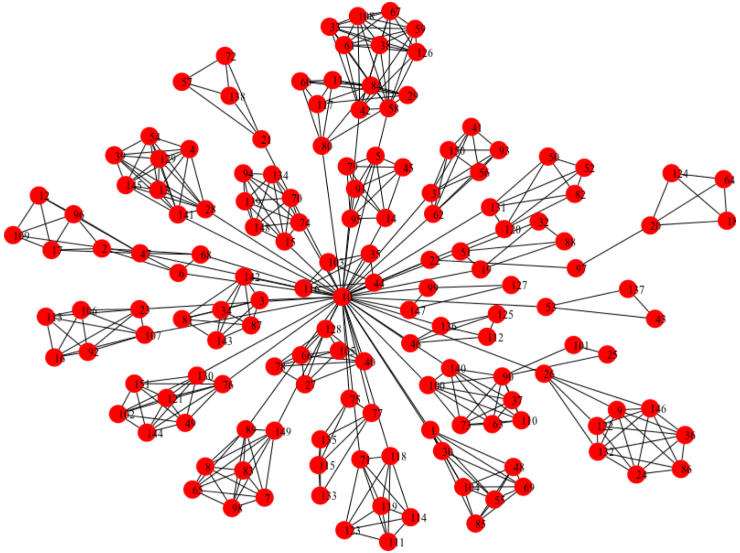
datasciencedojo
unleash the data scientist in you

Distributed Computing

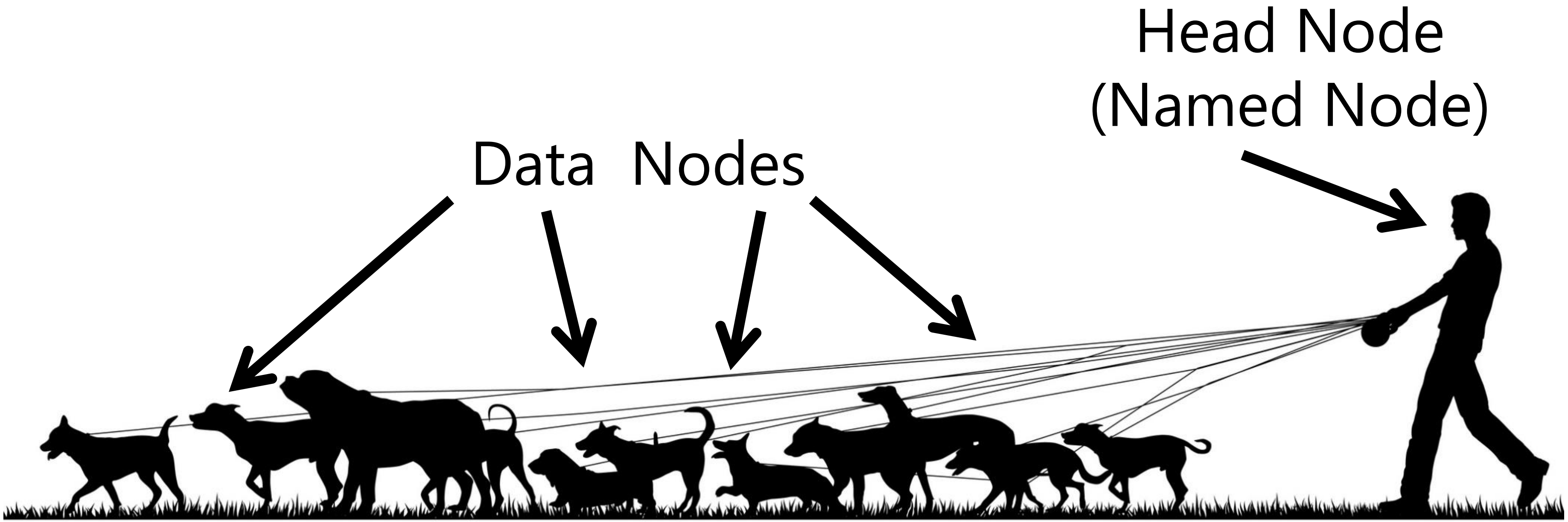




Cloud Computing



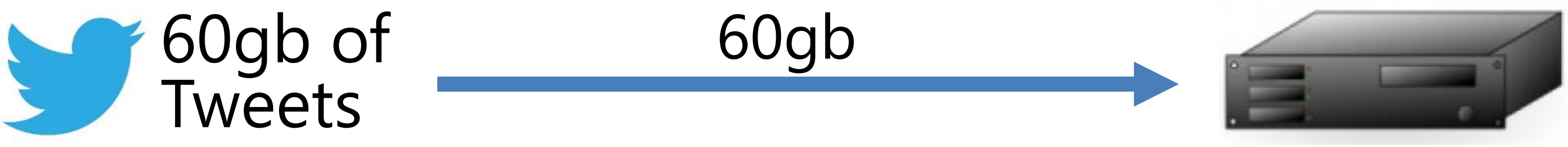
If dogs were servers…



Head Node
(Named Node)

Data Nodes

HDFS & MapReduce



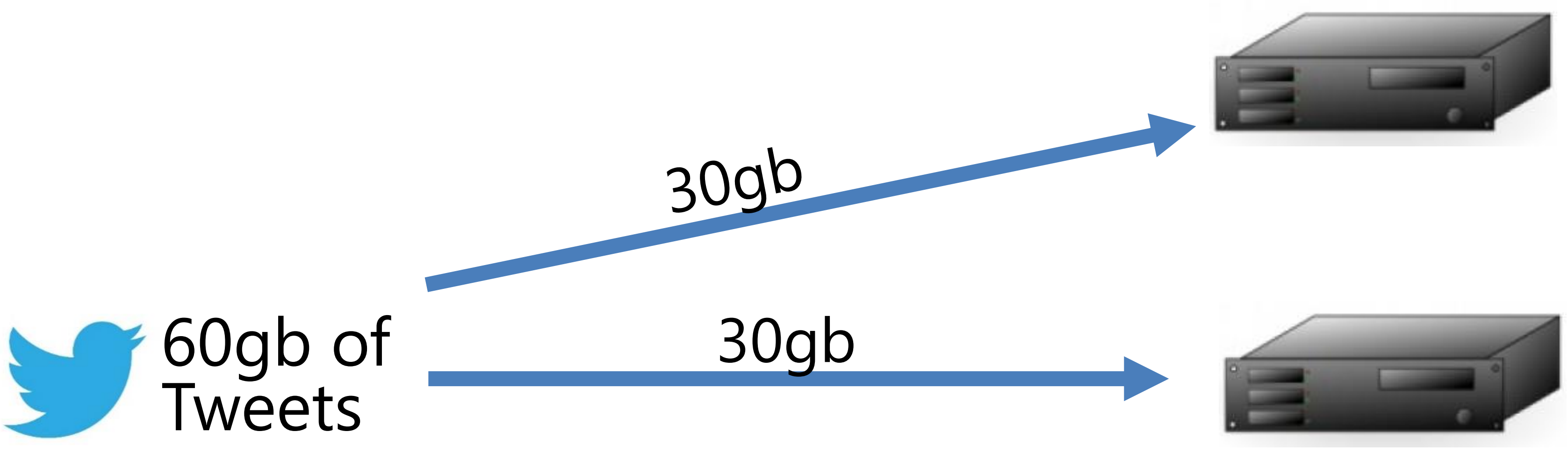
60gb of Tweets

60gb

1 Computer

Processing: 30 hours

HDFS & MapReduce



60gb of Tweets

30gb

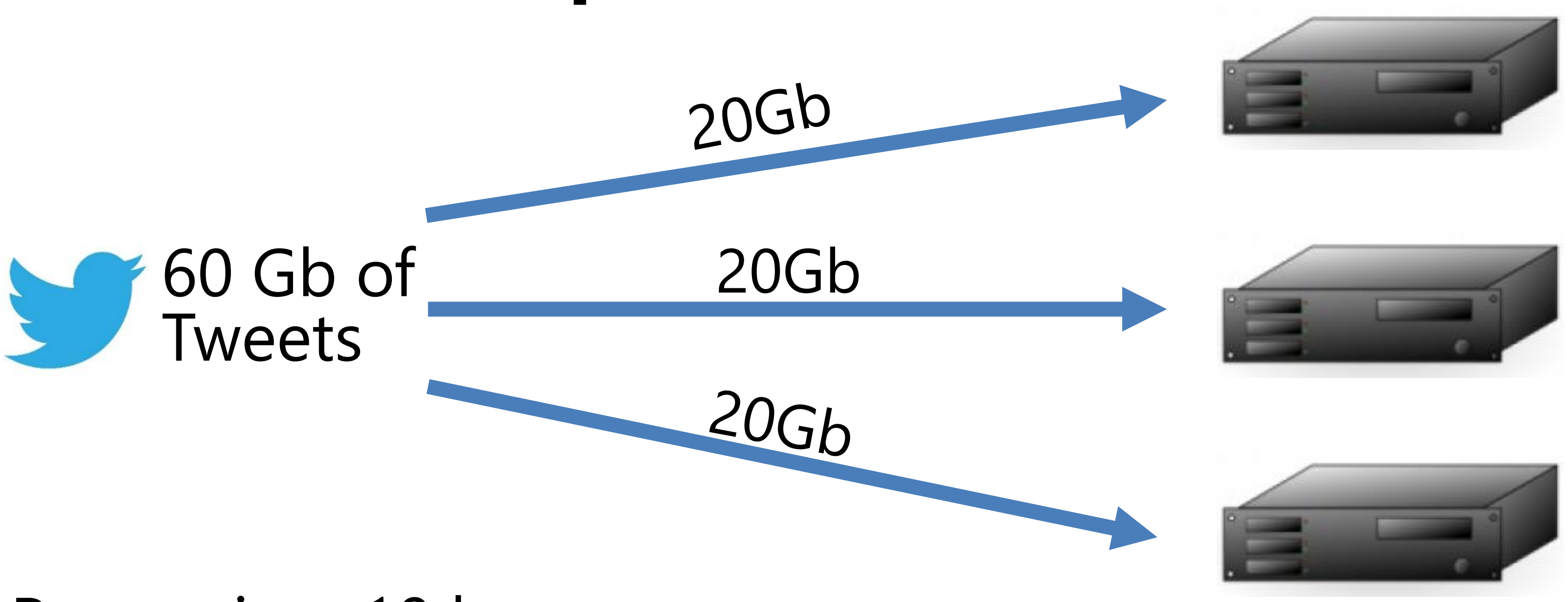
30gb

2 Computers

Processing: 15 hours

datasciencedojo
unleash the data scientist in you

HDFS & MapReduce



60 Gb of Tweets

20Gb

20Gb

20Gb

Processing: 10 hours

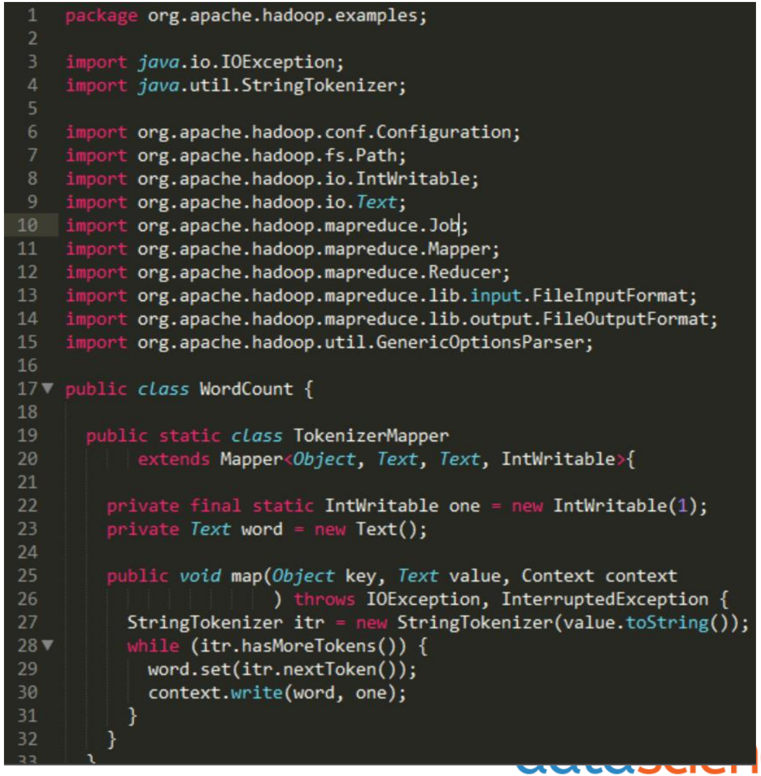
3 Computers

Most Cases, Linear Scaling Of Processing Power

Number of Computers	Processing Time (hours)
1	30
2	15
3	10
4	7.5
5	6
6	5
7	4.26
8	3.75
9	3.33

Limitations with MapReduce

- ~70 lines of code to do anything
- Slow
- Troubleshooting multiple computers
- Good devs are scarce
- Expensive certifications



```java
package org.apache.hadoop.examples;

import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;

public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
```

Ambari: Cluster provisioning, management, and monitoring

Avro (Microsoft .NET Library for Avro): Data serialization for the Microsoft .NET environment

HBase: Non-relational database for very large tables

HDFS: Hadoop Distributed File System

Hive: SQL-like querying

Mahout: Machine learning

MapReduce and YARN: Distributed processing and resource management

Oozie: Workflow management

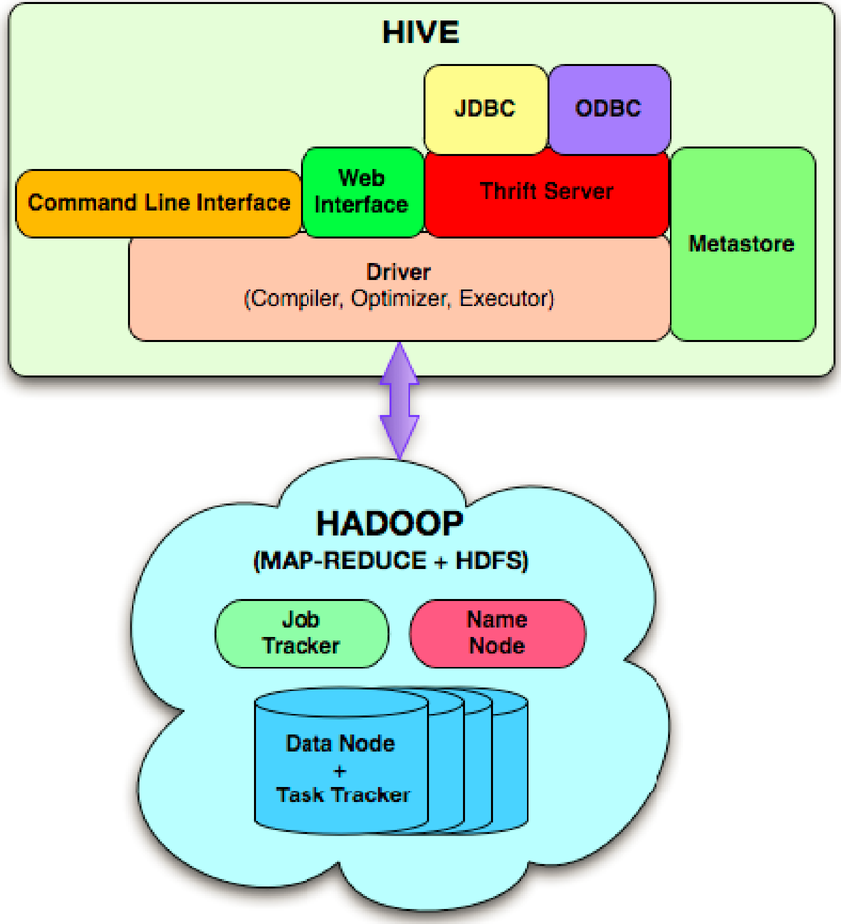
Pig: Simpler scripting for MapReduce transformations

Sqoop: Data import and export

Storm: Real-time processing of fast, large data streams

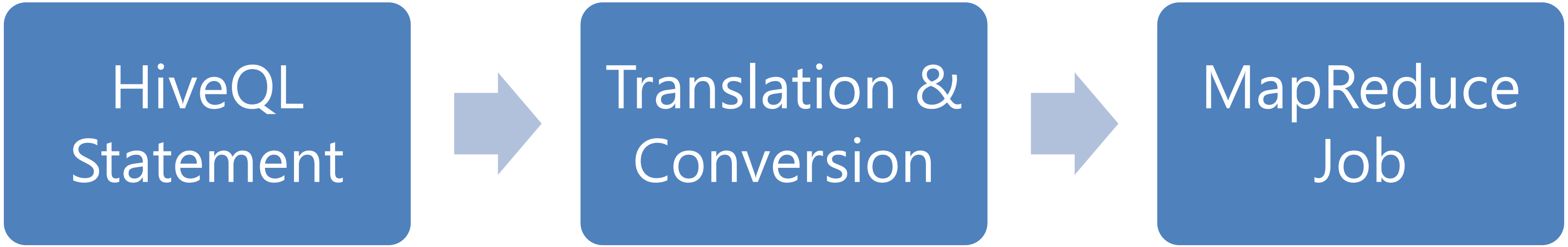
Zookeeper: Coordinates processes in distributed systems

HIVE



HIVE
JDBC
ODBC
Command Line Interface
Web Interface
Thrift Server
Metastore
Driver
(Compiler, Optimizer, Executor)
HADOOP
(MAP-REDUCE + HDFS)
Job Tracker
Name Node
Data Node
+
Task Tracker

Hive Jobs

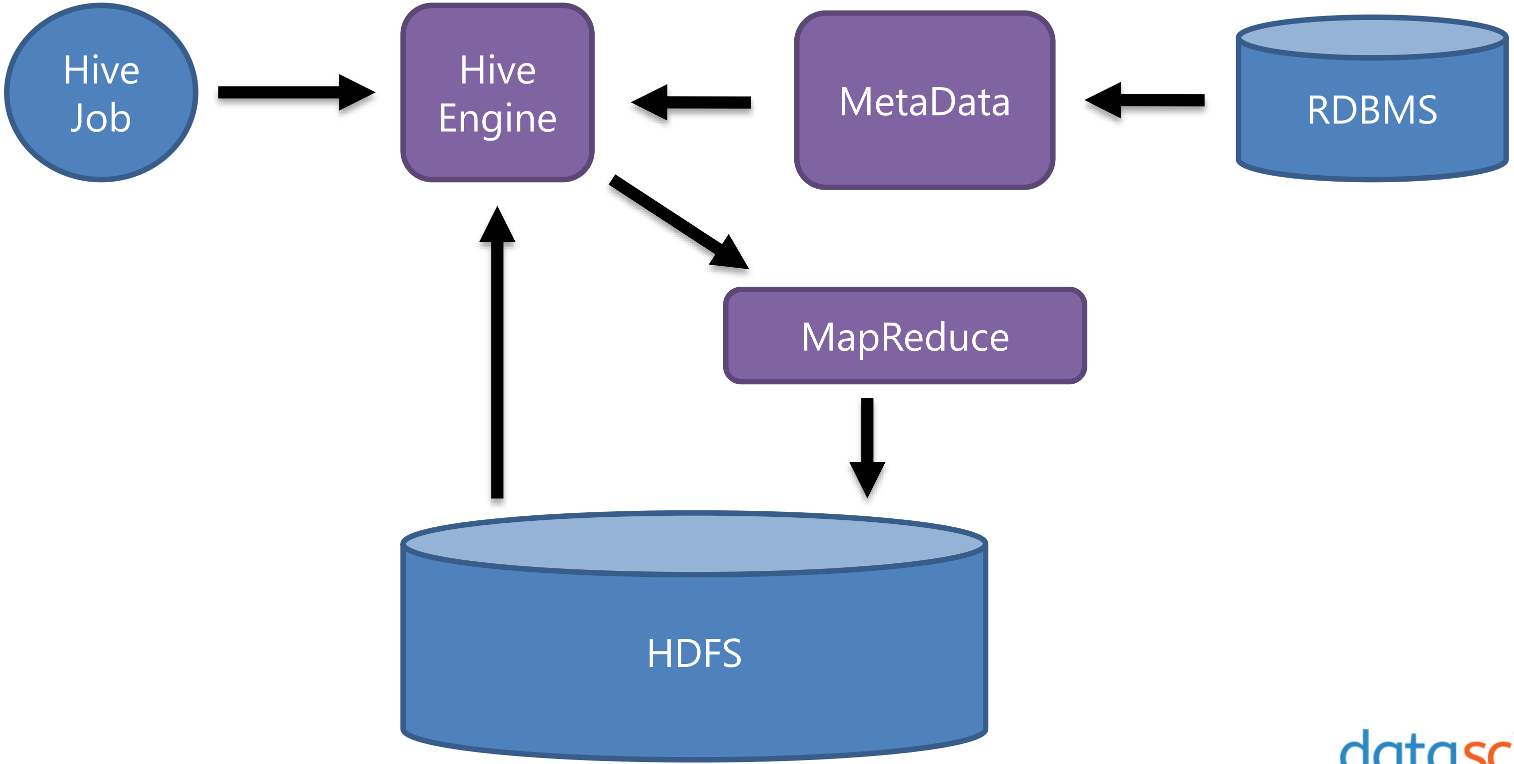


HiveQL Statement

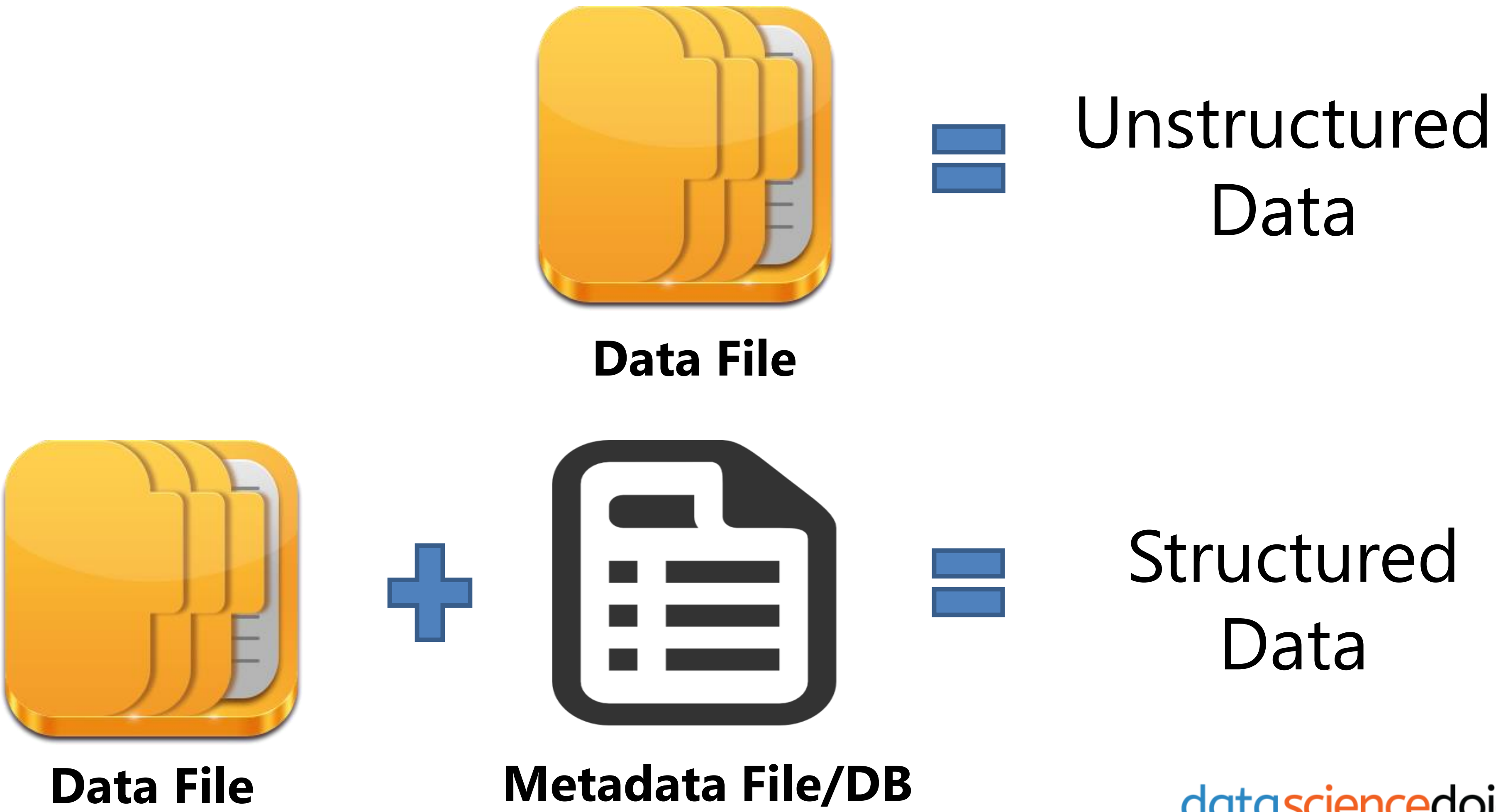
Translation & Conversion

MapReduce Job

Hive Architecture



datasciencedojo
unleash the data scientist in you



Data File = Unstructured Data

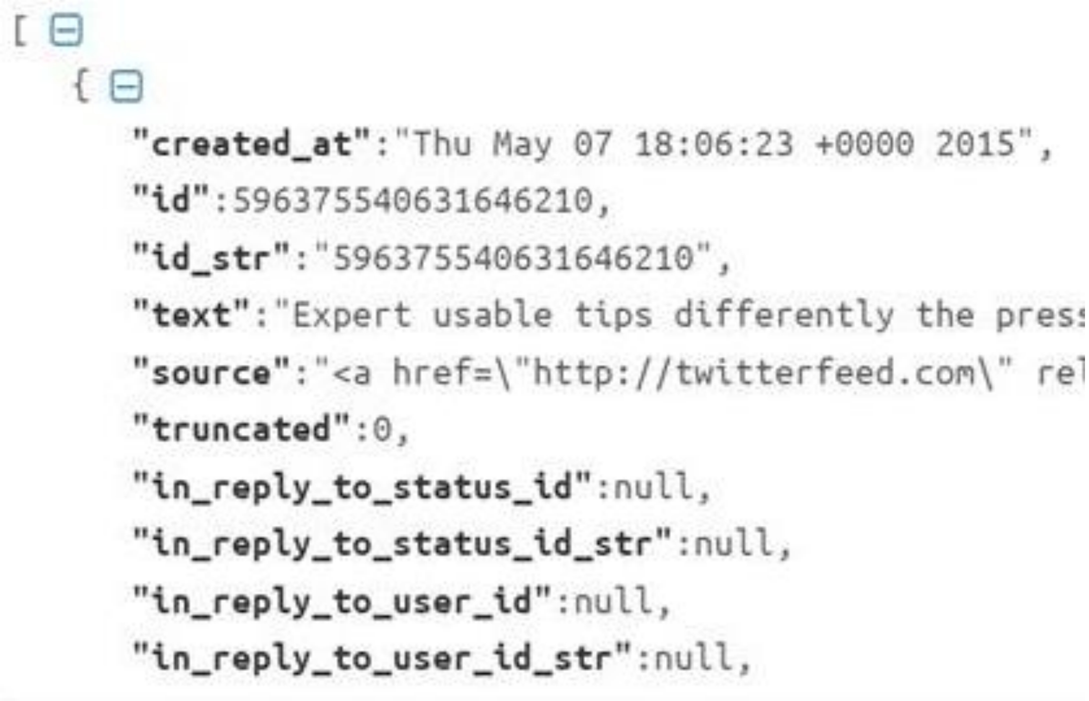
Data File + Metadata File/DB = Structured Data

Semi Structured Data

Self Describing

Flat Files

- XML
- JSON
- CSV
- TSV



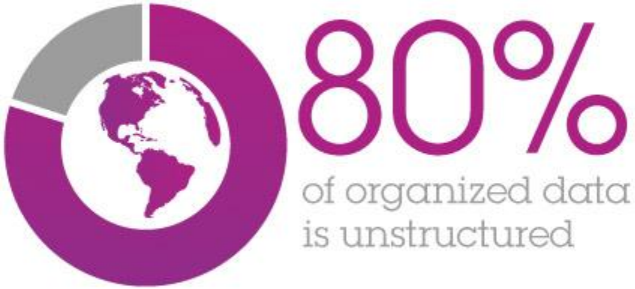
```
[ ⊟
  { ⊟
    "created_at":"Thu May 07 18:06:23 +0000 2015",
    "id":596375540631646210,
    "id_str":"596375540631646210",
    "text":"Expert usable tips differently the pres
    "source":"<a href=\"http://twitterfeed.com\" re
    "truncated":0,
    "in_reply_to_status_id":null,
    "in_reply_to_status_id_str":null,
    "in_reply_to_user_id":null,
    "in_reply_to_user_id_str":null,
```

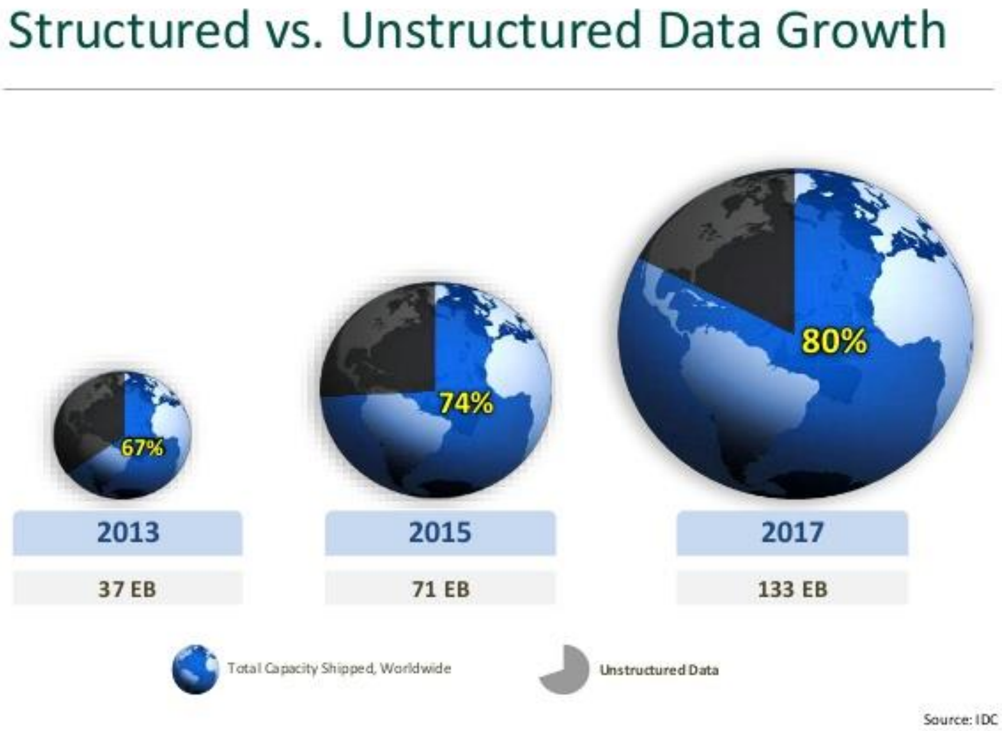
datasciencedojo
unleash the data scientist in you

Why Hive?

- SQL spoken here (HiveQL)
- ODBC driver
- BI Integration
- Supports only Structured Data

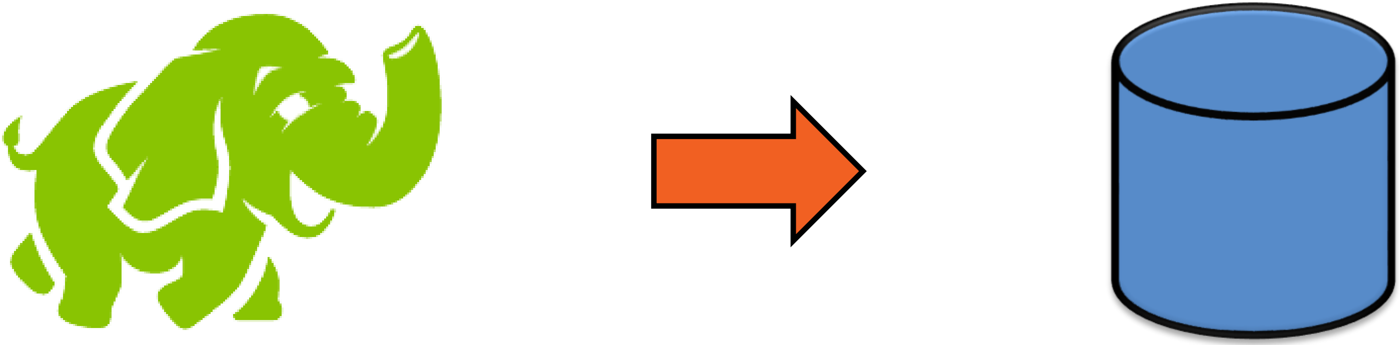
Limitations




Structured vs. Unstructured Data Growth

Source: IDC

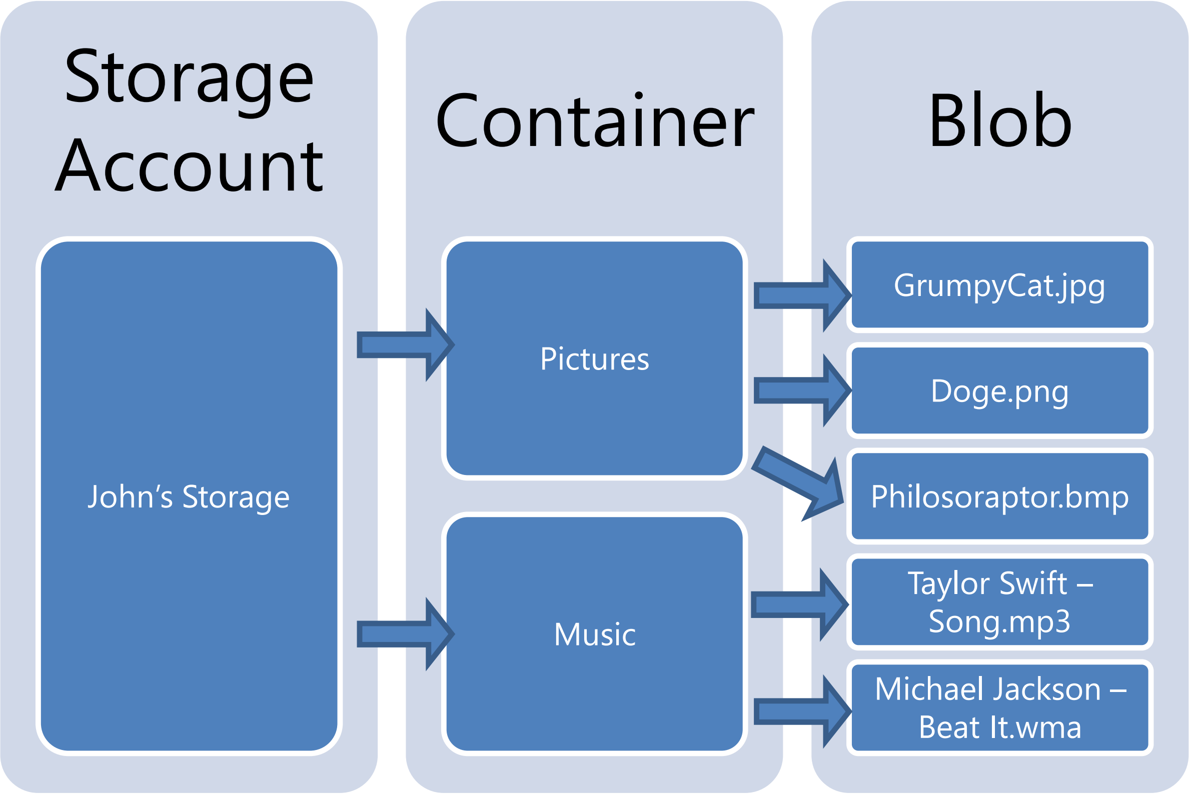
Azure Blob Storage



HDInsight

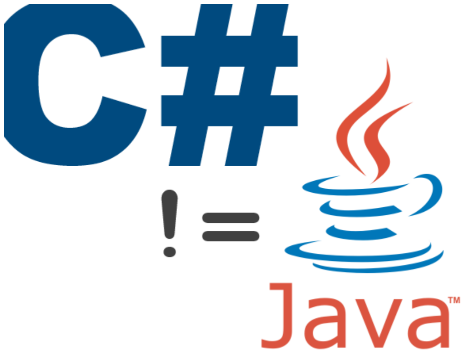
Blob Storage

Azure Blob Storage



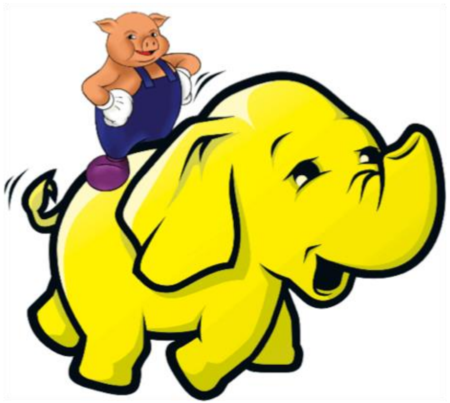
datasciencedojo
unleash the data scientist in you

When to Use Each



VS

VS



C#
Java
MapReduce

Hive

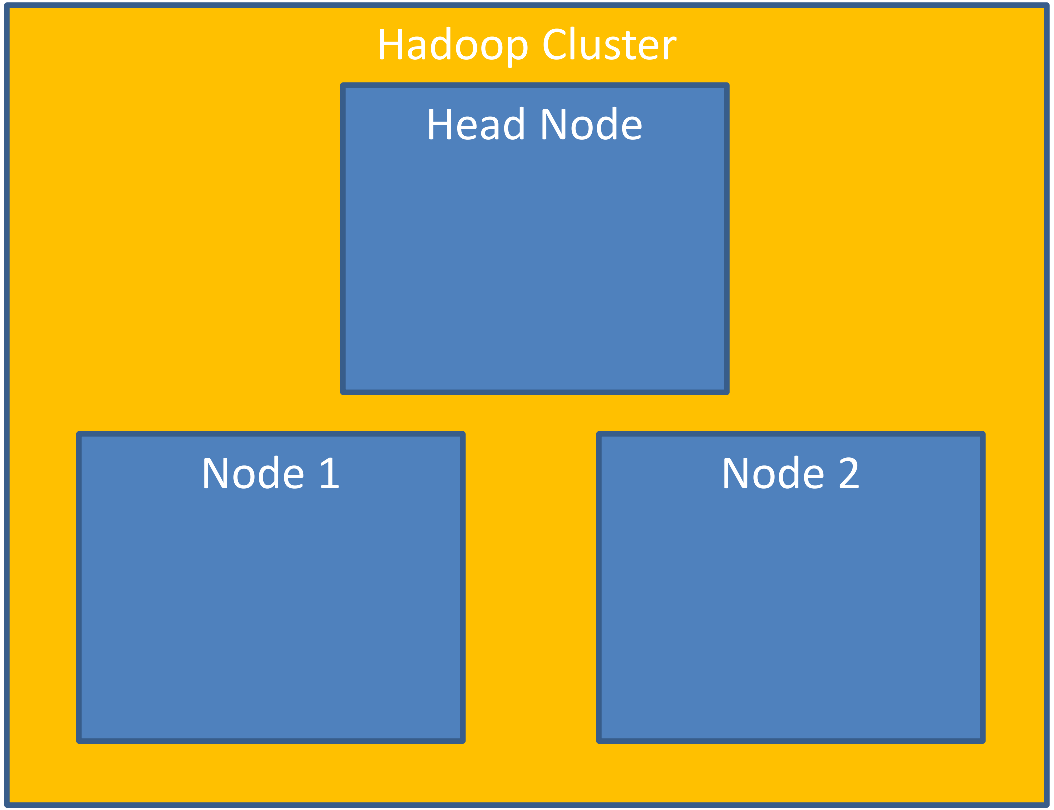
Pig

MapReduce, via Playing Cards

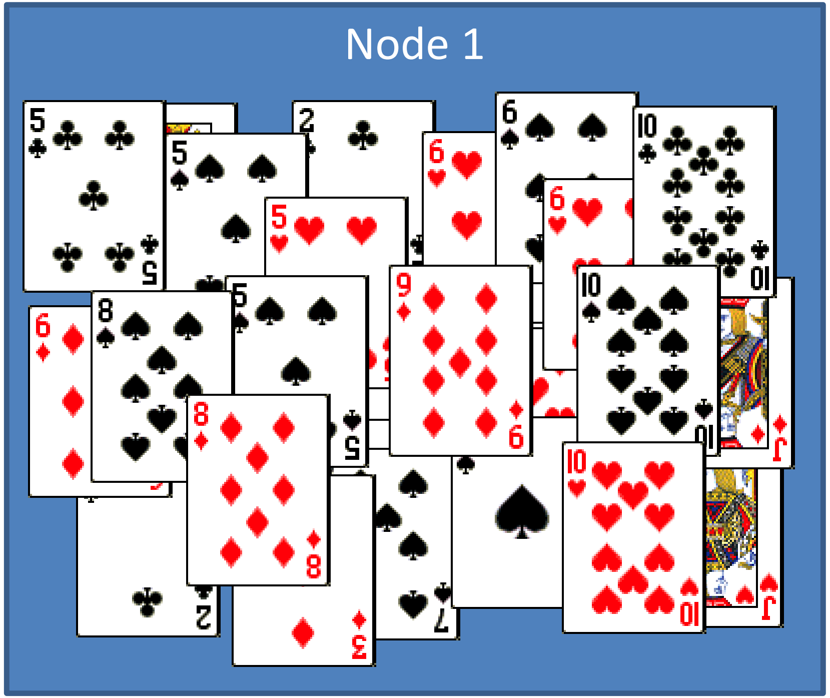
Let’s count the number of spades, clubs, hearts, and diamonds in a stack of cards, the way map reduce would.

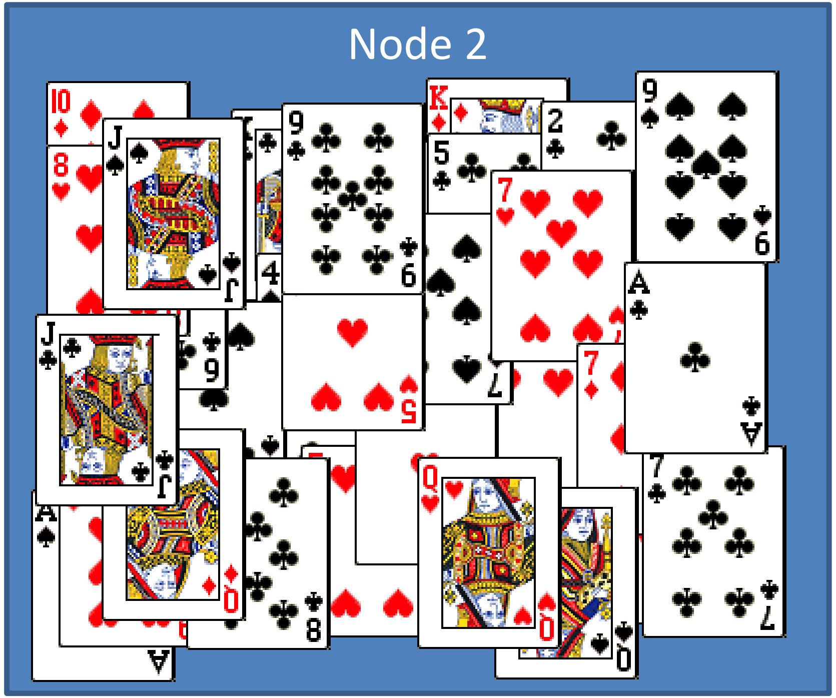
- Each card represents a row of data
- Each suite represents an attribute of the data

Using a 2 Data Node Cluster



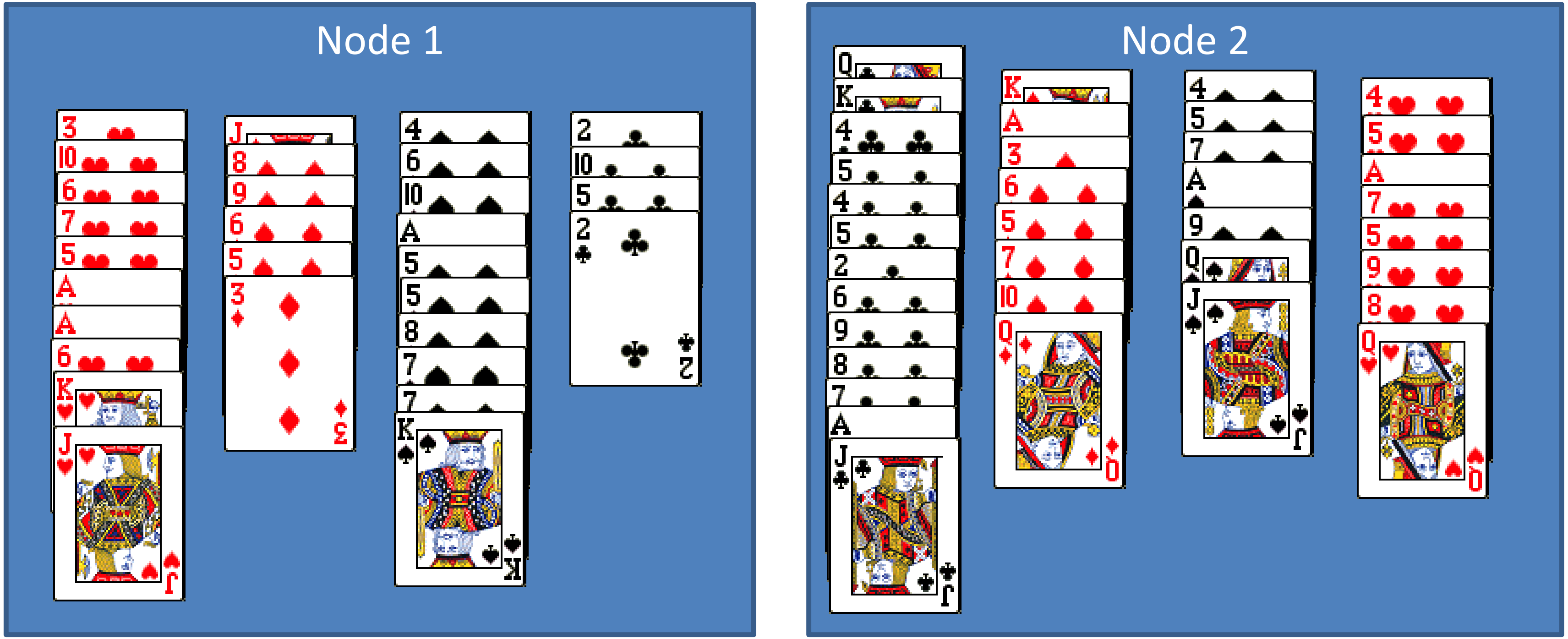
Mapping: Each Node’s HDFS



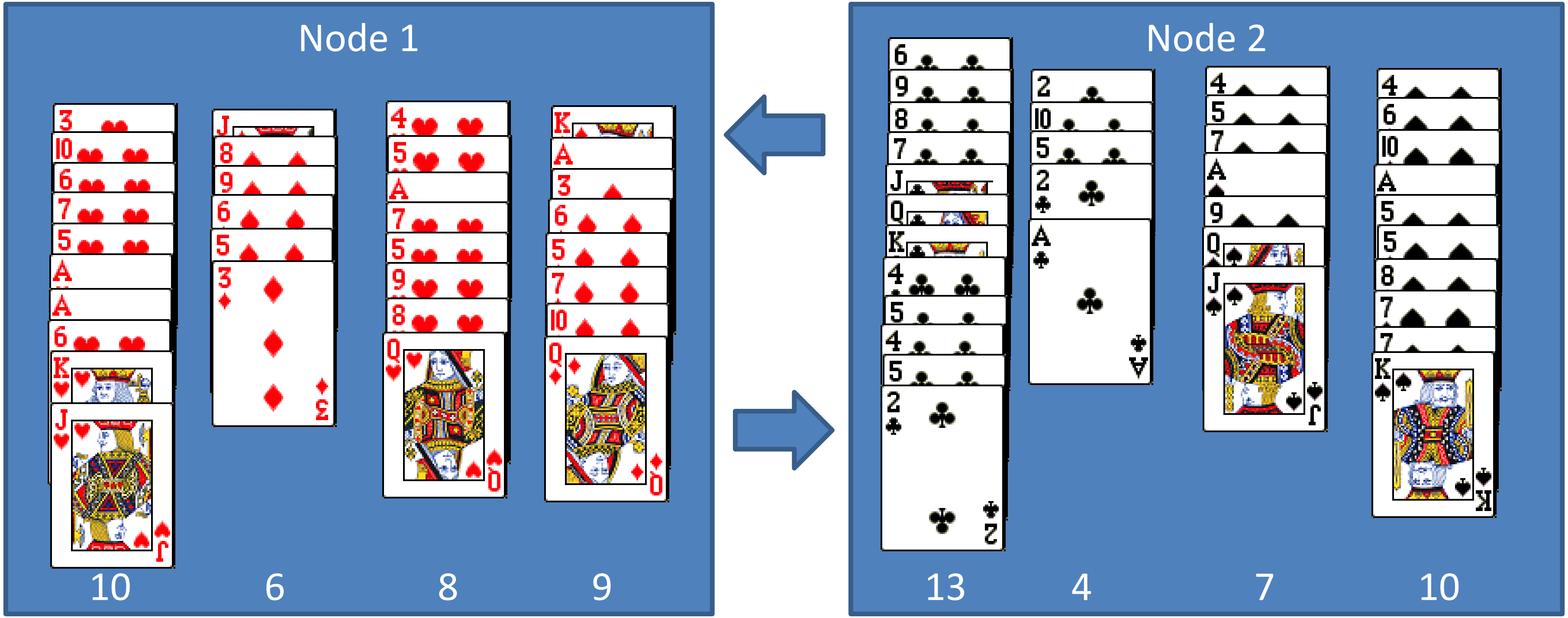


datasciencedojo
unleash the data scientist in you

Mapping: Node Sorting

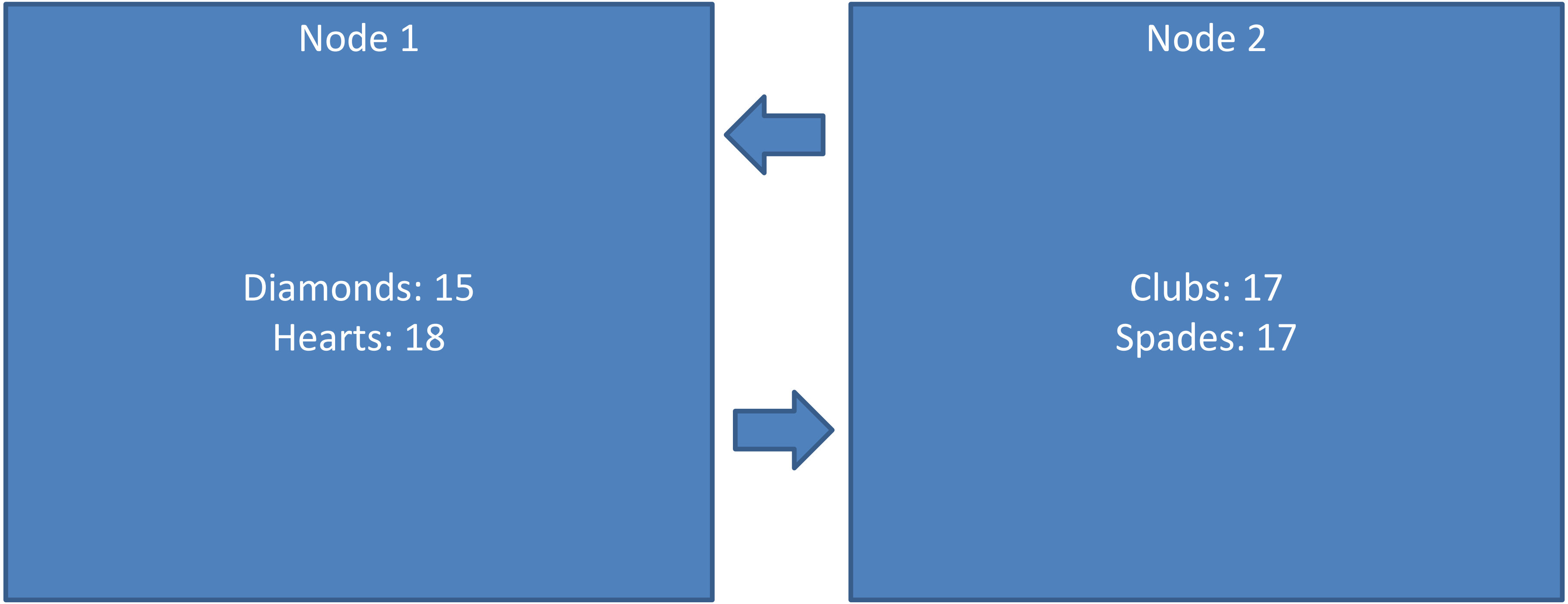


Mapping: Node Shuffle, Data Transfer



datasciencedojo
unleash the data scientist in you

Mapping: Node Shuffle, Data Transfer



Node 1

Diamonds: 15
Hearts: 18

Node 2

Clubs: 17
Spades: 17

QUESTIONS