Big Data Engineering With MapReduce and Hive

Data Science Dojo

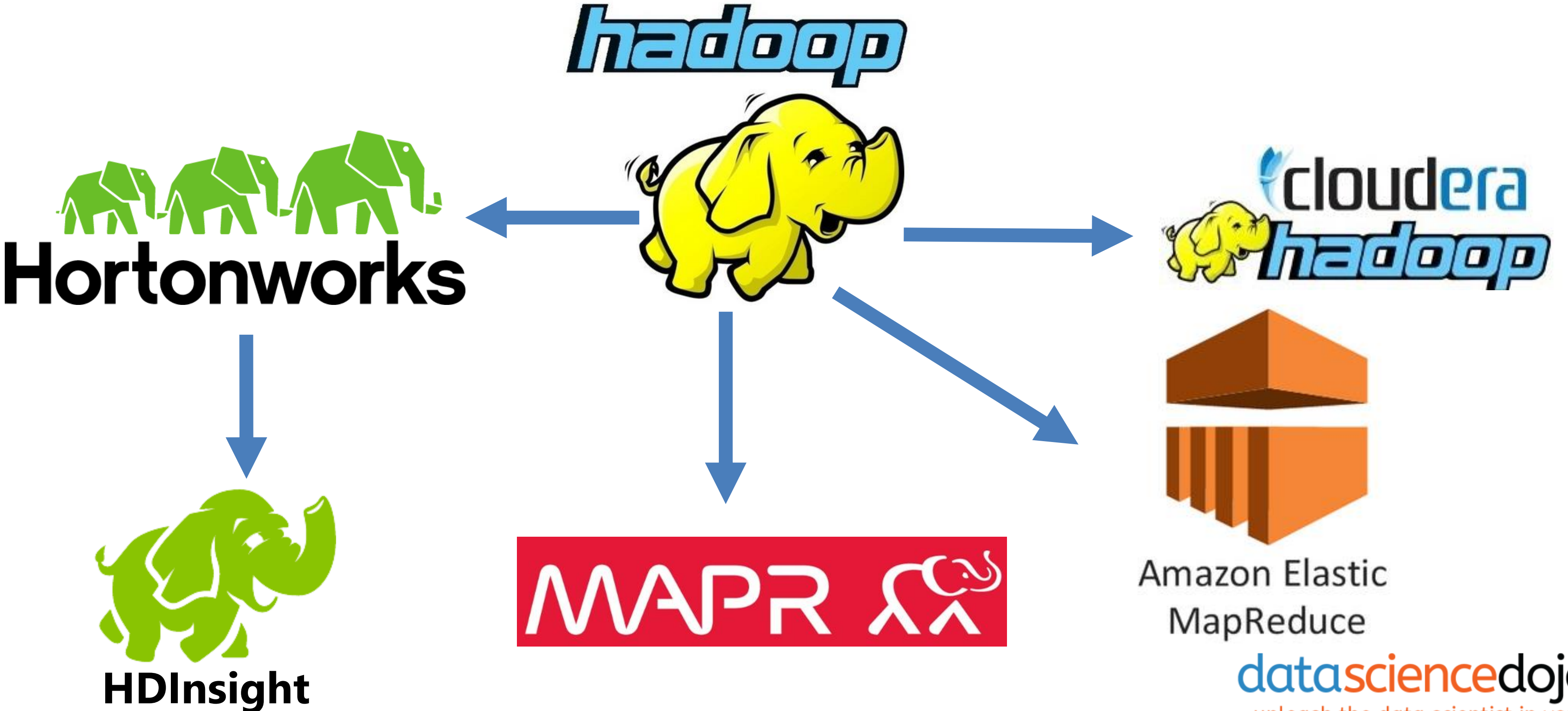
datasciencedojo
unleash the data scientist in you

Agenda

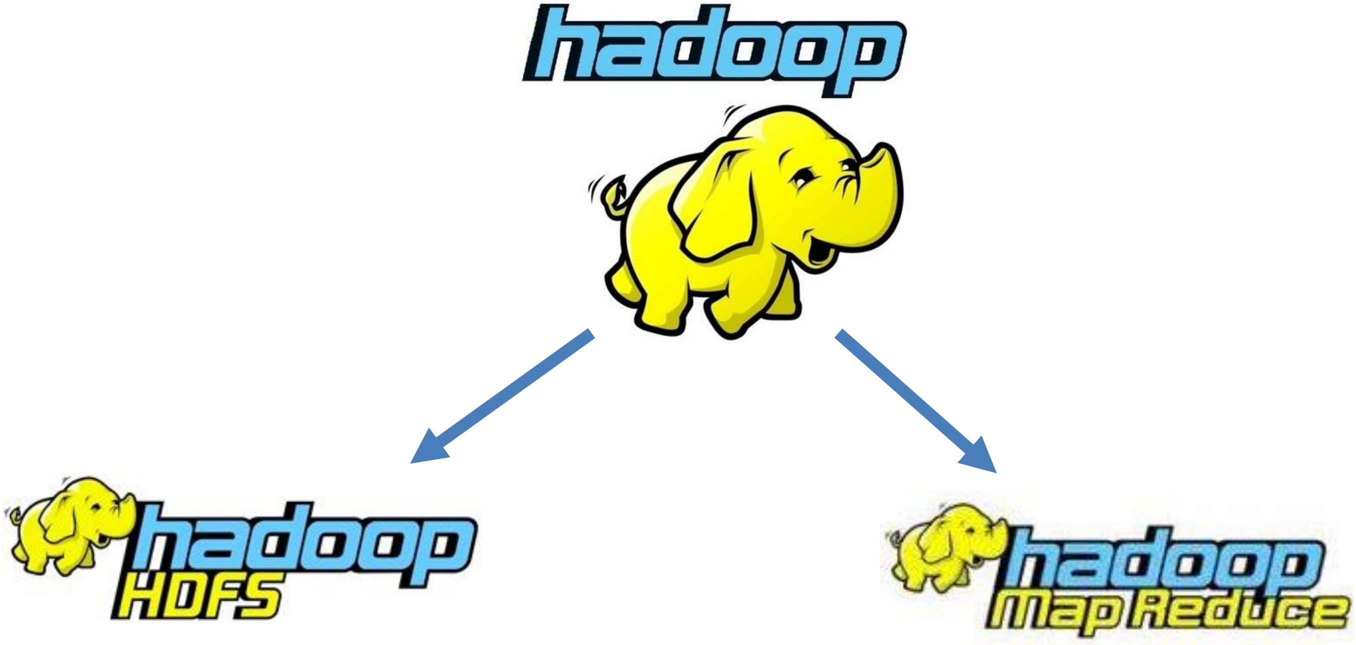
hadoop

HIVE

Hadoop Implementations

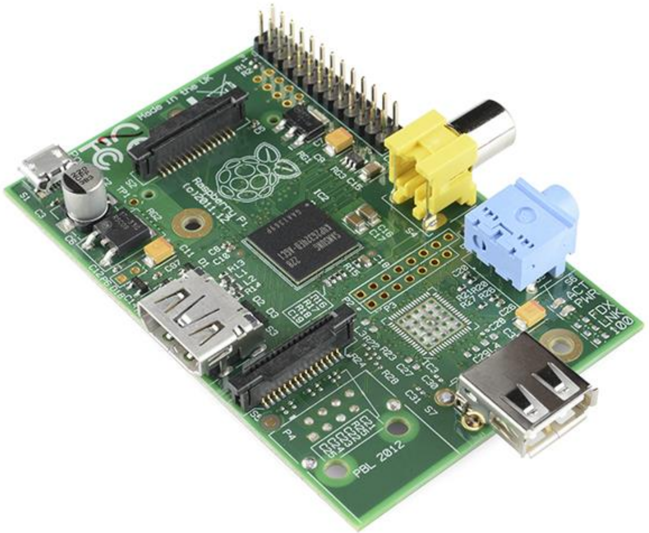


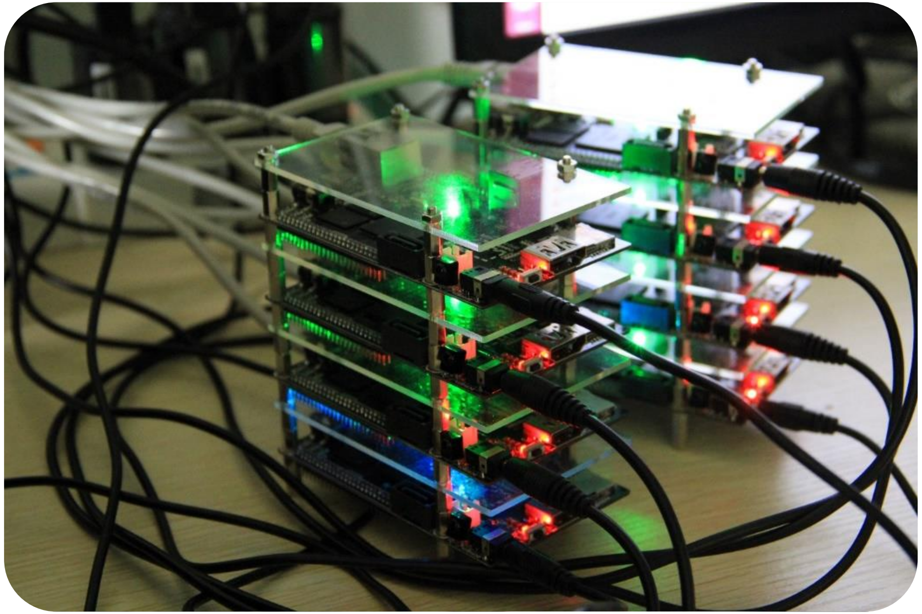
Hadoop



Turn Back The Clock, The Mainframe

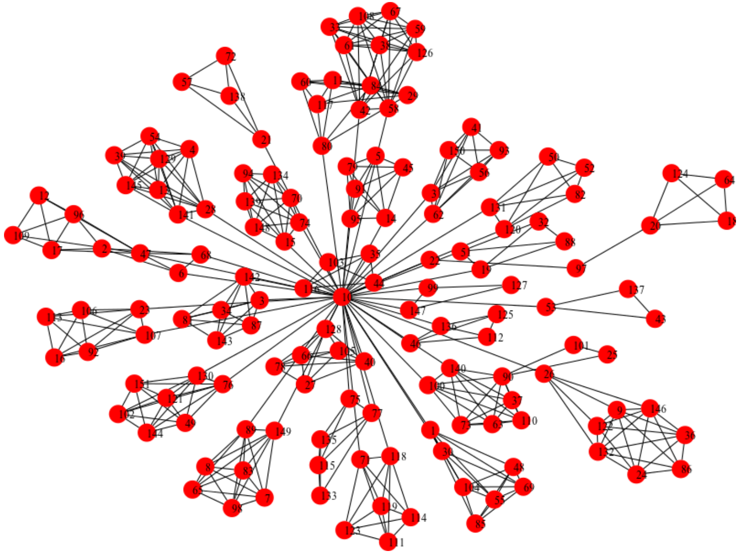
Distributed Computing





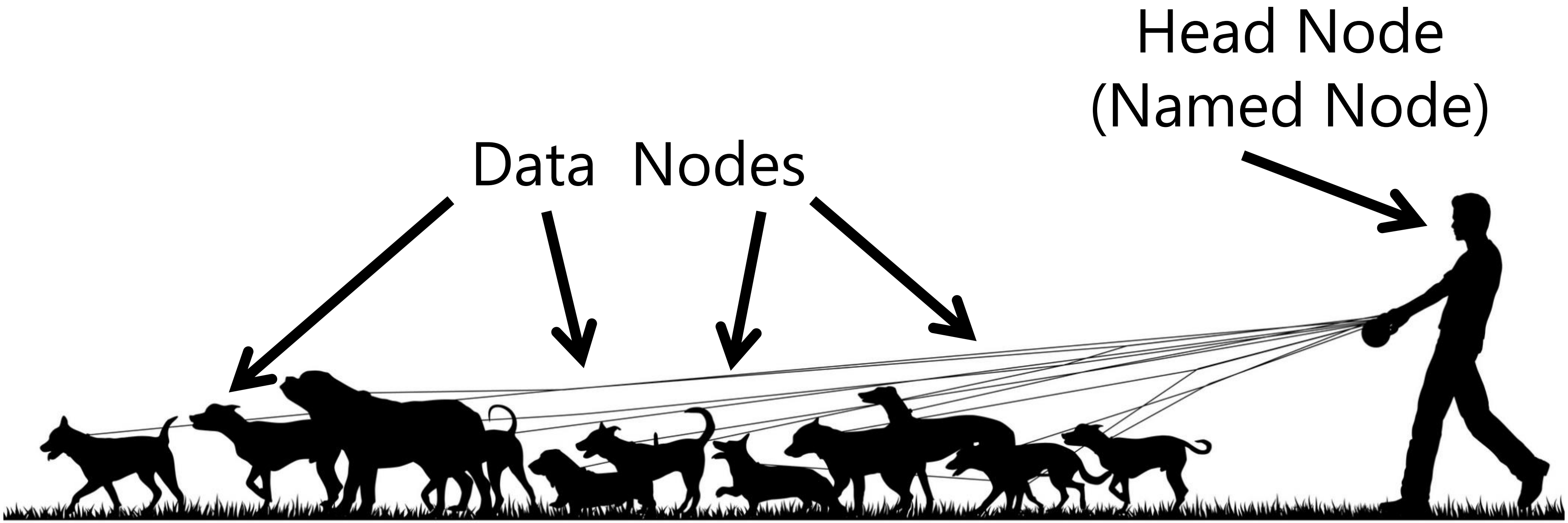
datasciencedojo
unleash the data scientist in you

Cloud Computing



datasciencedojo
unleash the data scientist in you

If dogs were servers...



HDFS & MapReduce

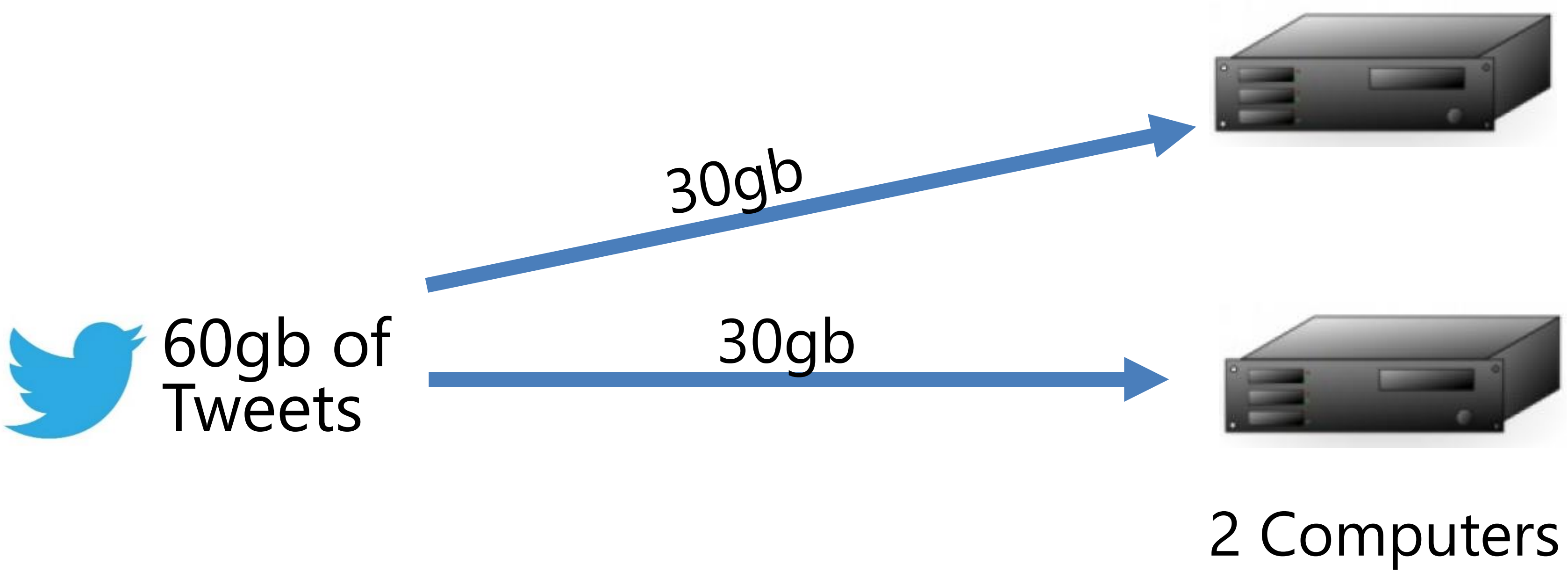
60gb of Tweets

60gb

1 Computer

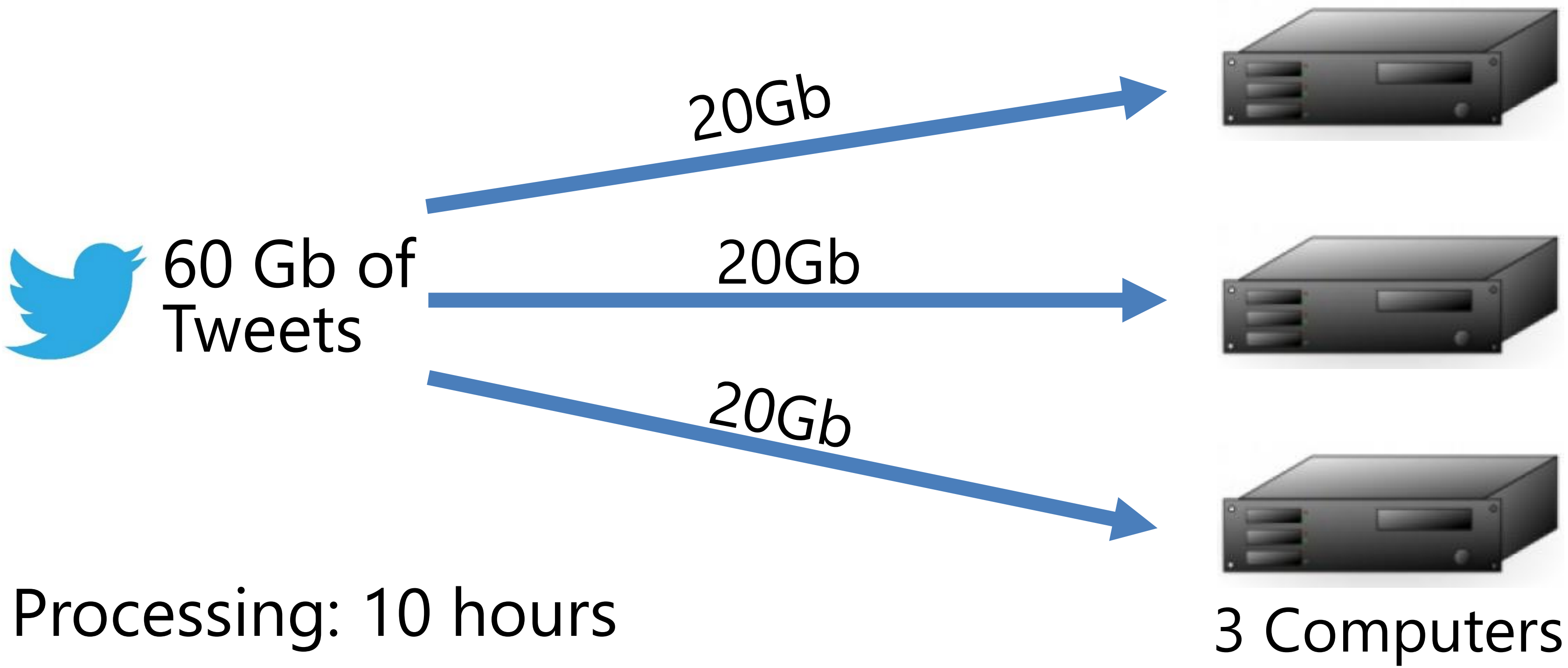
Processing: 30 hours

HDFS & MapReduce



Processing: 15 hours

HDFS & MapReduce



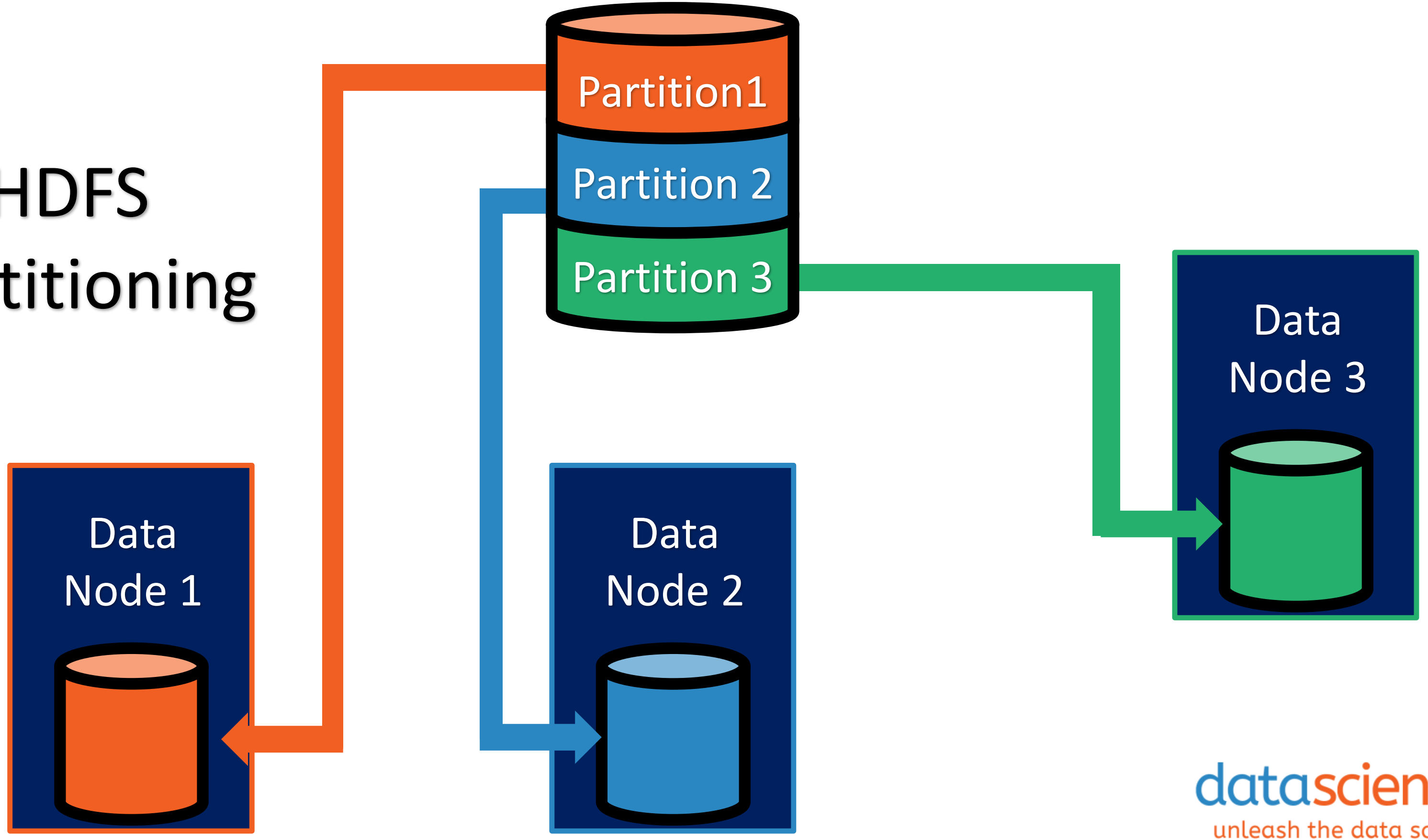
Processing: 10 hours

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

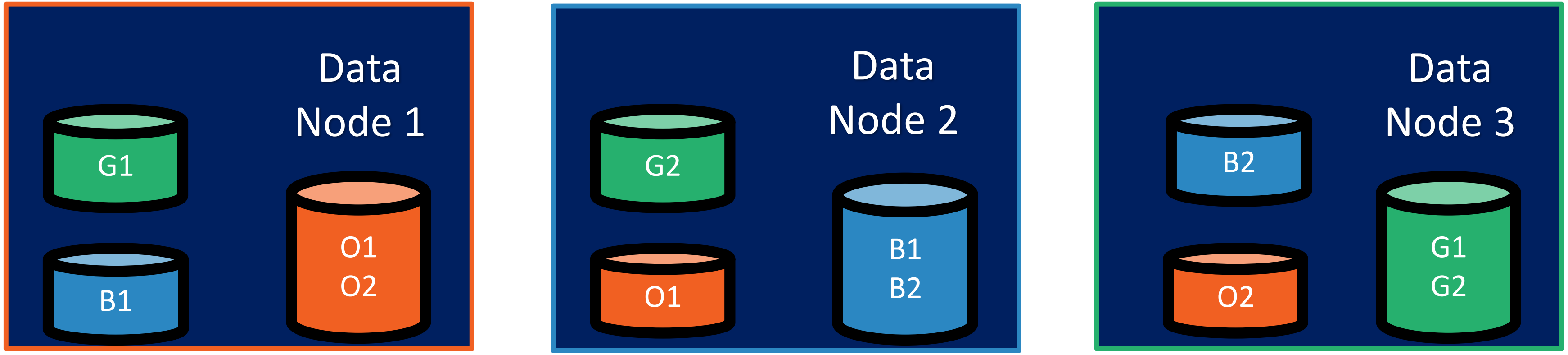
HDFS

HDFS Partitioning



datasciencedojo
unleash the data scientist in you

HDFS Redundancy



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

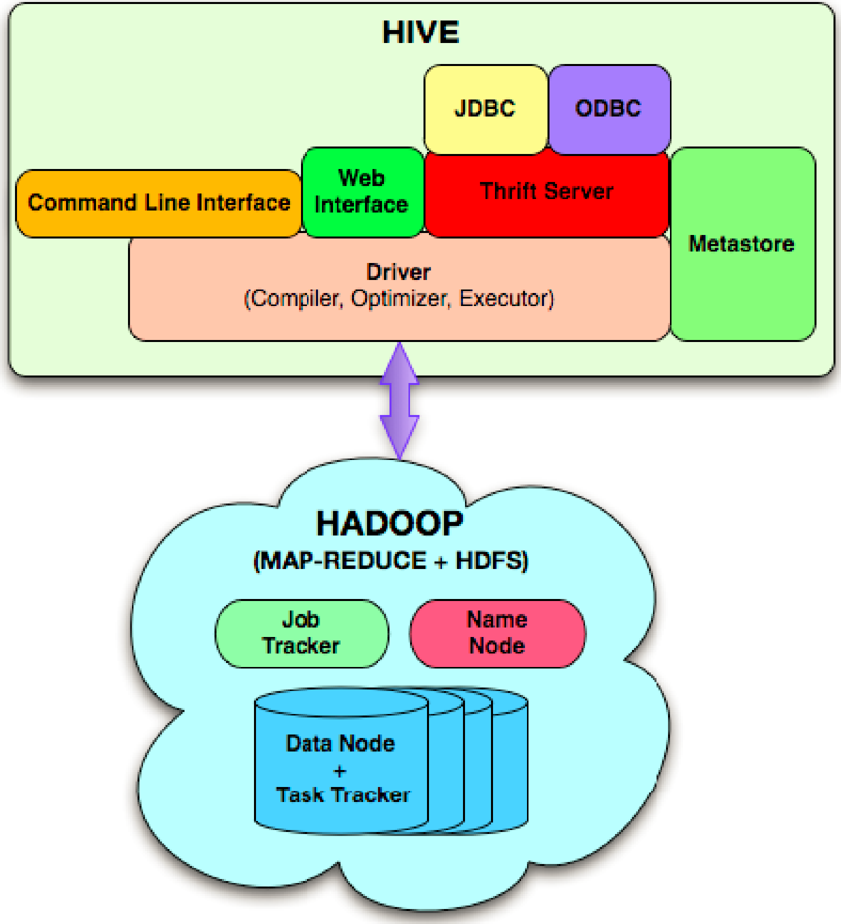
Oozie: Workflow management

Pig: Simpler scripting for MapReduce transformations

Sqoop: Data import and export

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

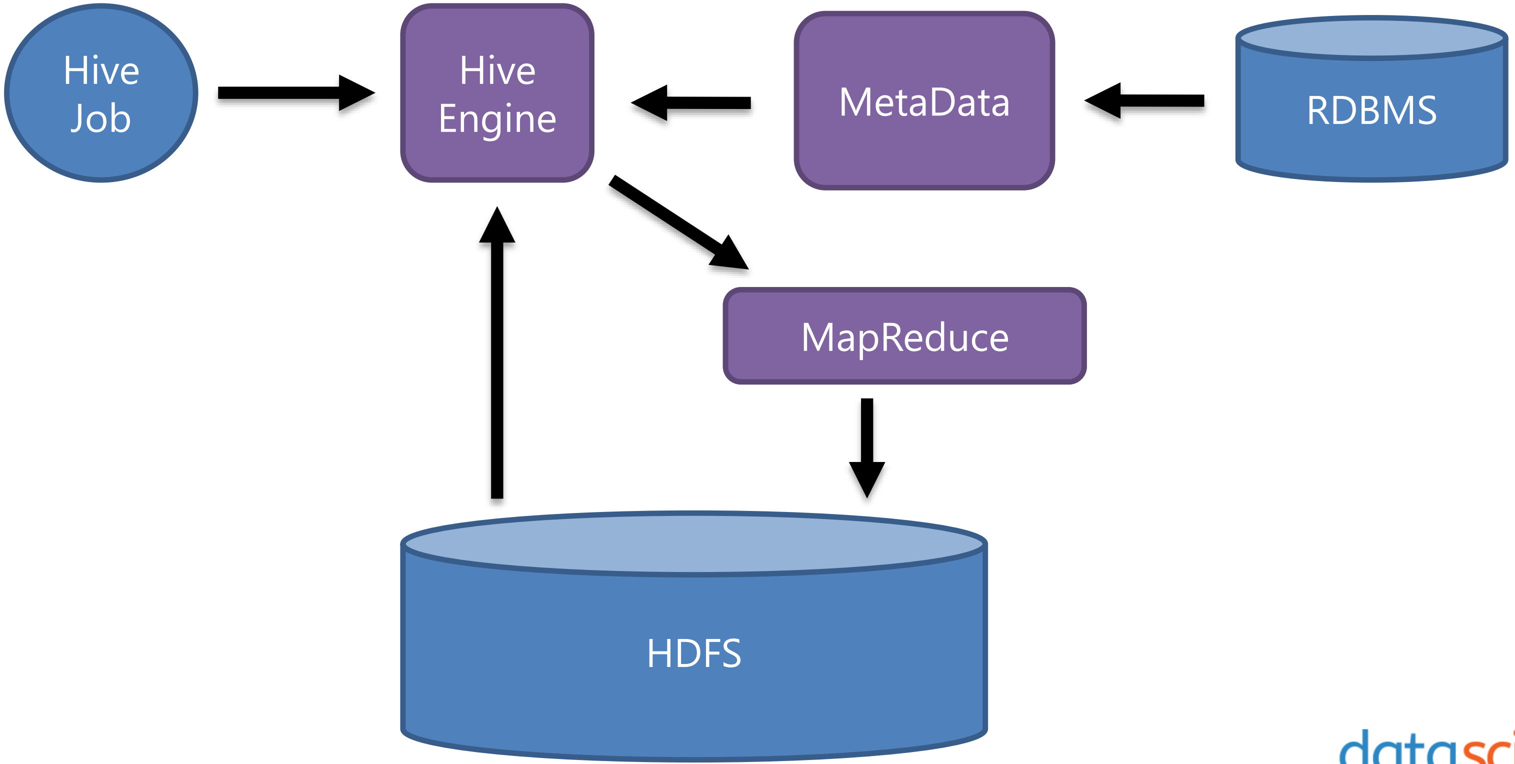
Zookeeper: Coordinates processes in distributed systems

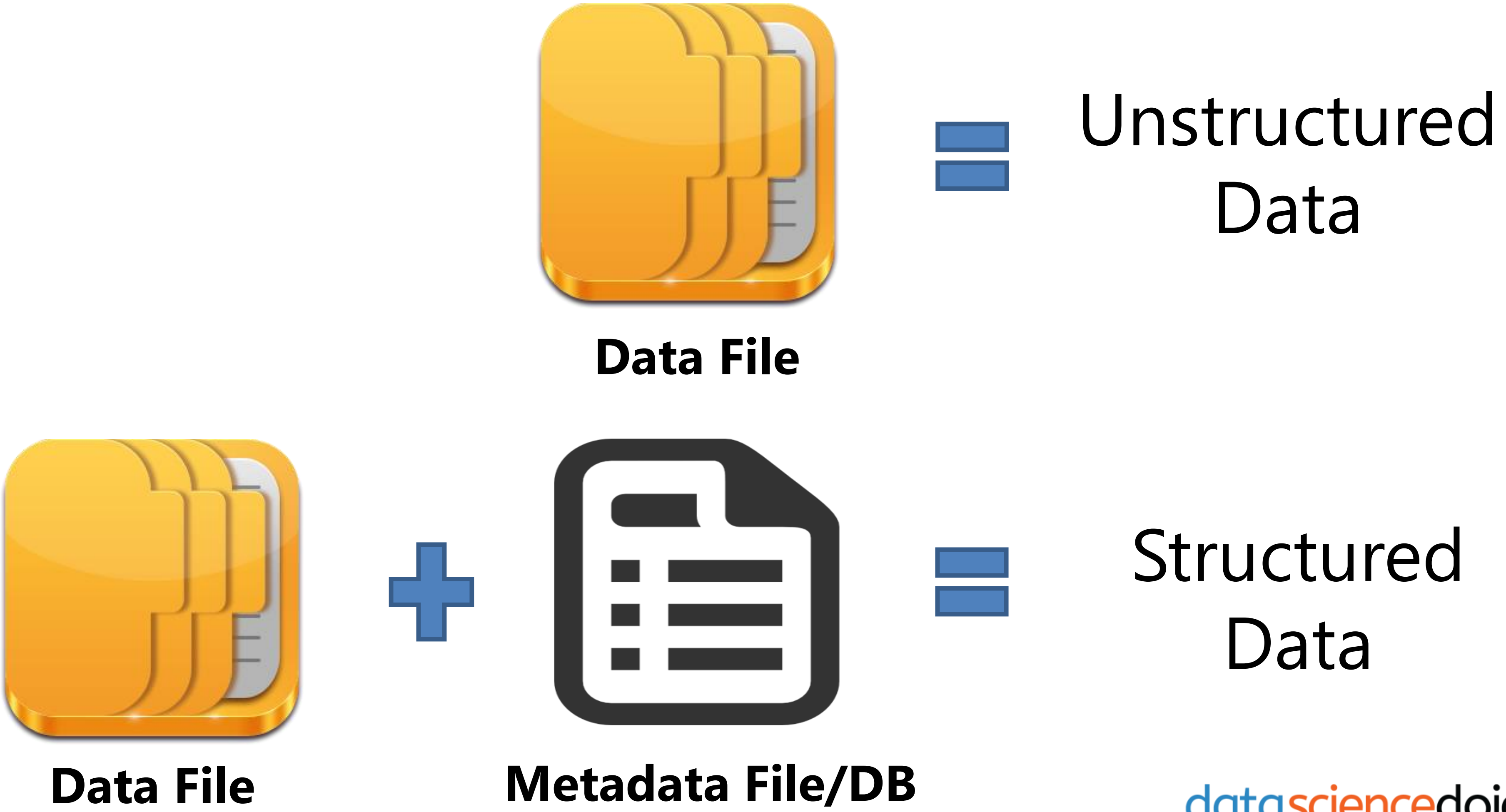

HIVE
JDBC
ODBC
Command Line Interface
Web Interface
Thrift Server
Metastore
Driver
(Compiler, Optimizer, Executor)
HADOOP
(MAP-REDUCE + HDFS)
Job Tracker
Name Node
Data Node
+
Task Tracker

Hive Jobs

HiveQL Statement → Translation & Conversion → MapReduce Job

Hive Architecture




Data File
Unstructured Data
Data File
Metadata File/DB
Structured Data

Semi Structured Data

Self Describing

Flat Files

- XML
- JSON
- CSV
- TSV

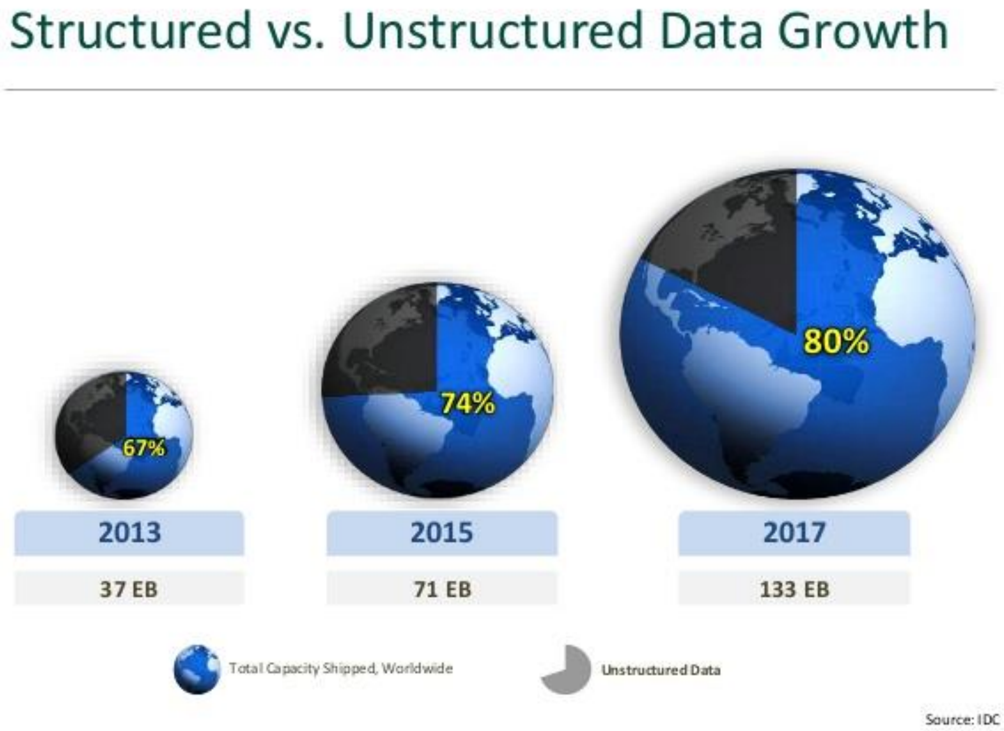

```
[
  {
    "created_at":"Thu May 07 18:06:23 +0000 2015",
    "id":596375540631646210,
    "id_str":"596375540631646210",
    "text":"Expert usable tips differently the pres
    "source":"<a href=\"http://twitterfeed.com\" re
    "truncated":0,
    "in_reply_to_status_id":null,
    "in_reply_to_status_id_str":null,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


datasciencedojo
unleash the data scientist in you

Why Hive?

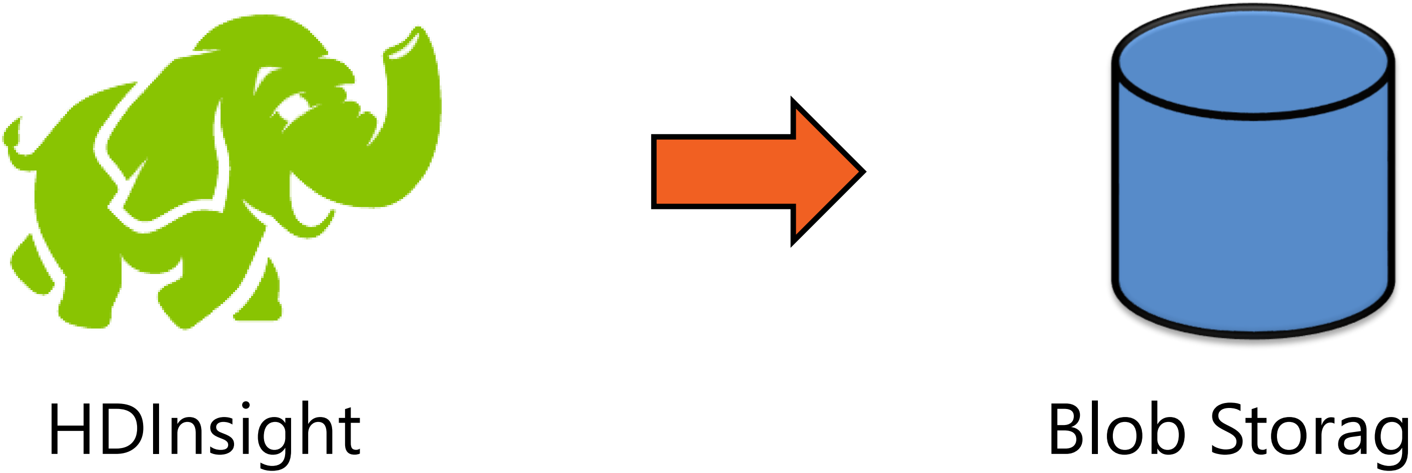
- SQL spoken here (HiveQL)
- ODBC driver
- BI Integration
- Supports only Structured Data

Limitations

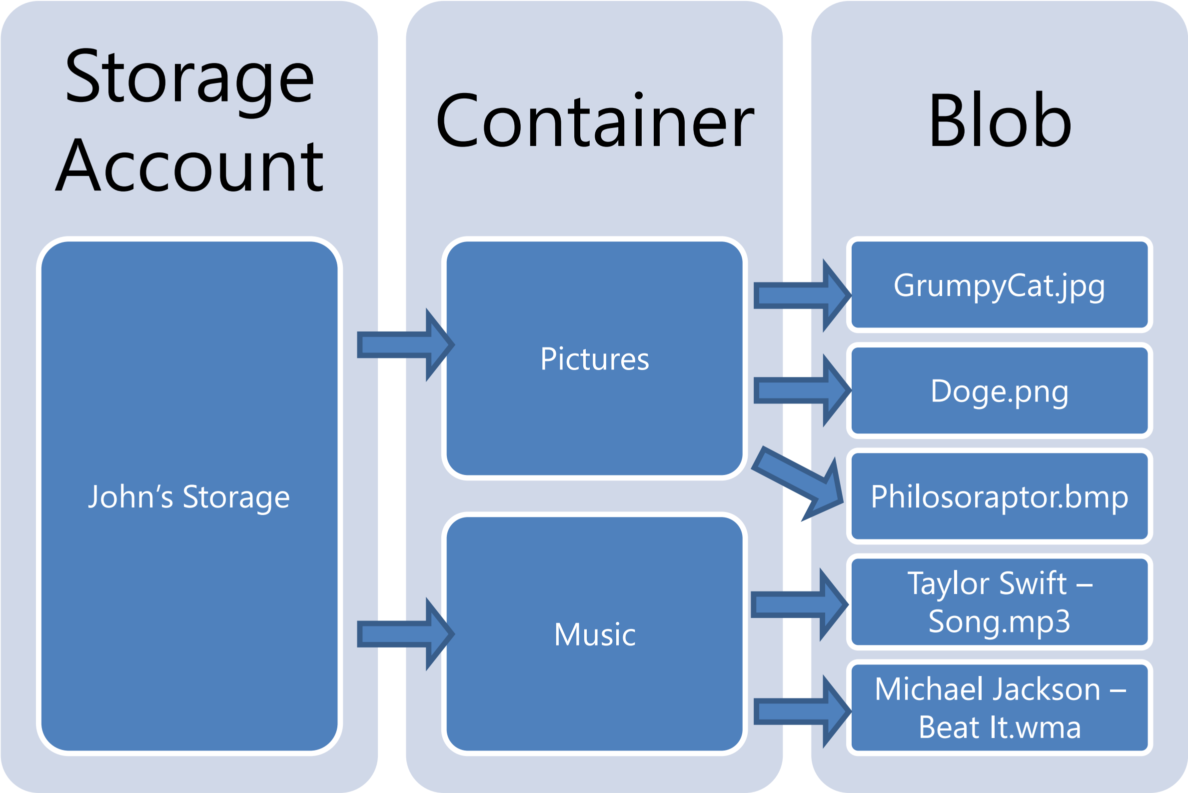


datasciencedojo
unleash the data scientist in you

Azure Blob Storage



Azure Blob Storage



When to Use Each

VS

VS

C#
Java
MapReduce

Hive

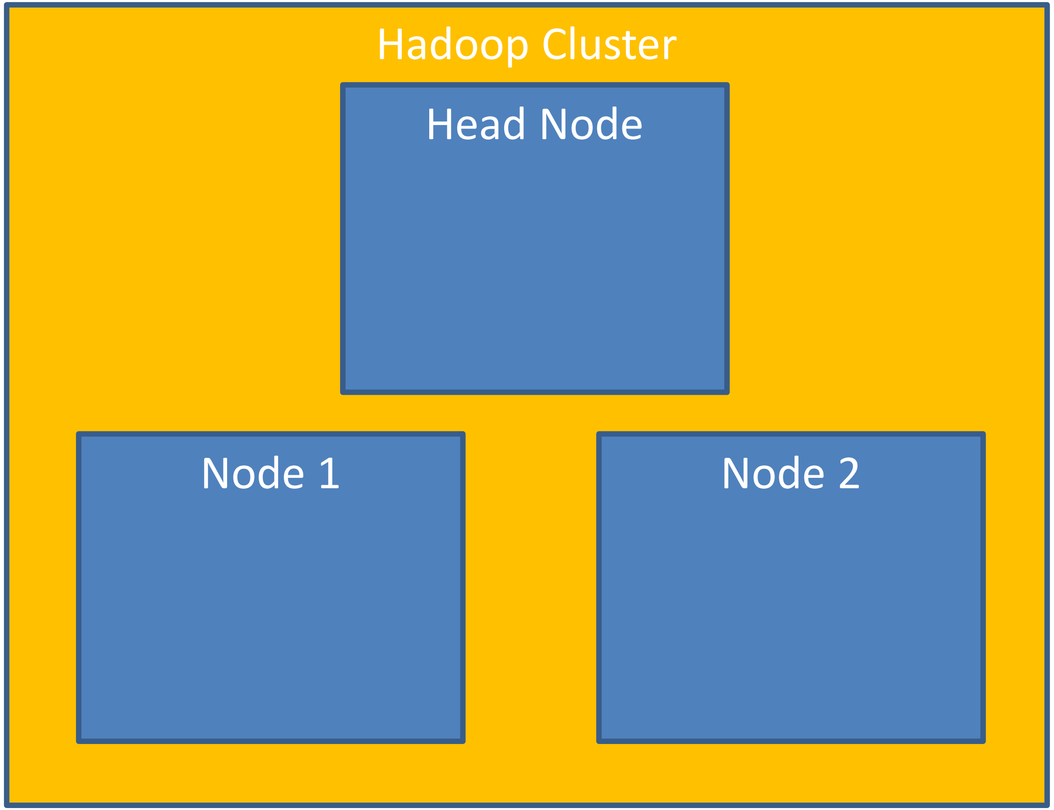
Pig

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

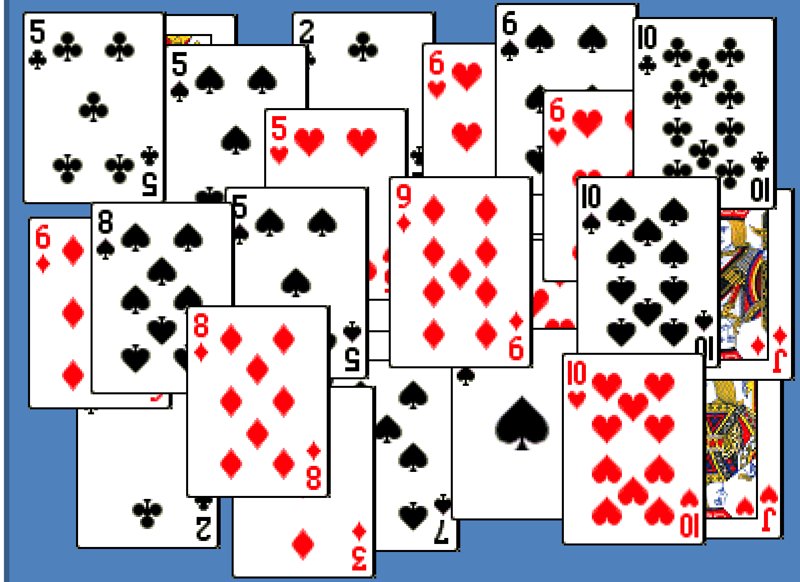
Using a 2 Data Node Cluster



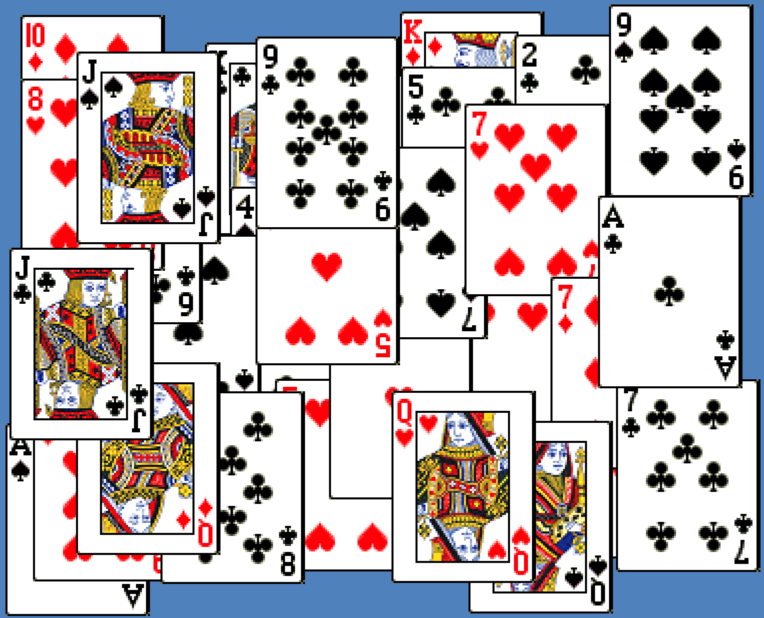
datasciencedojo
unleash the data scientist in you

Mapping: Each Node's HDFS

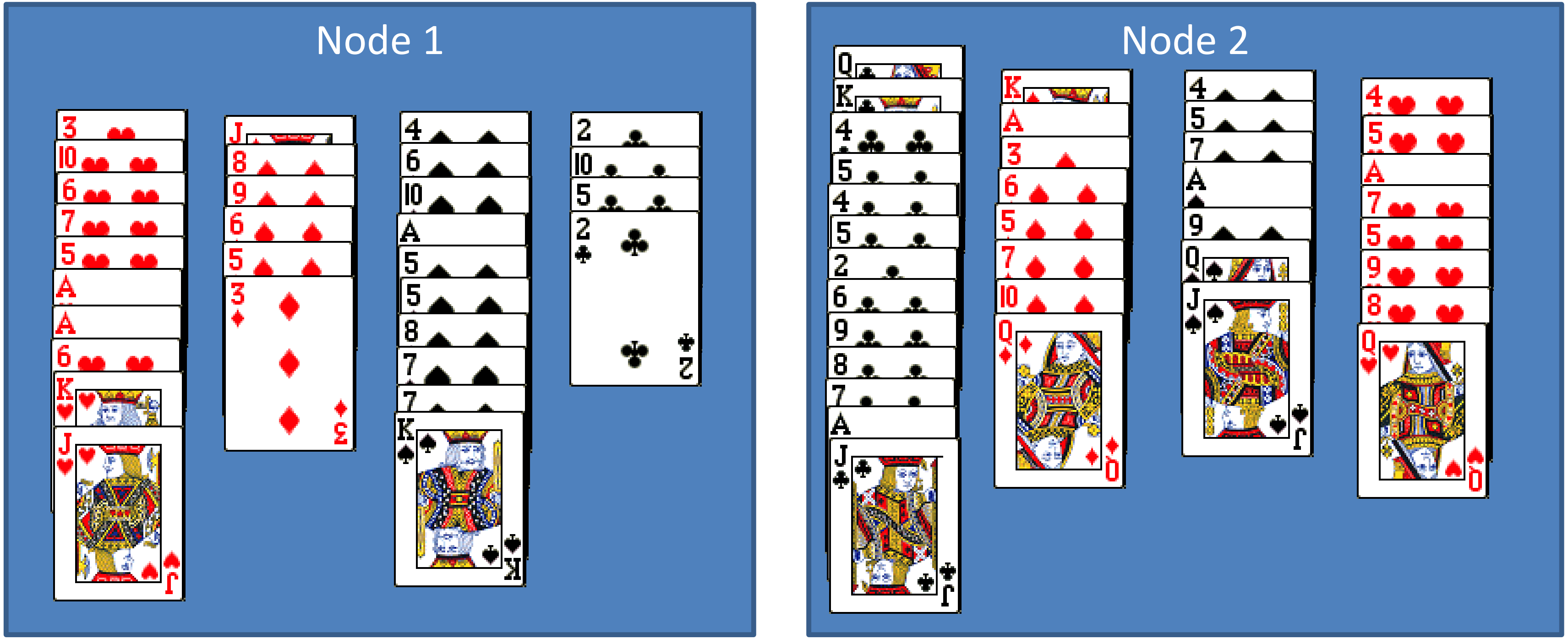
Node 1



Node 2

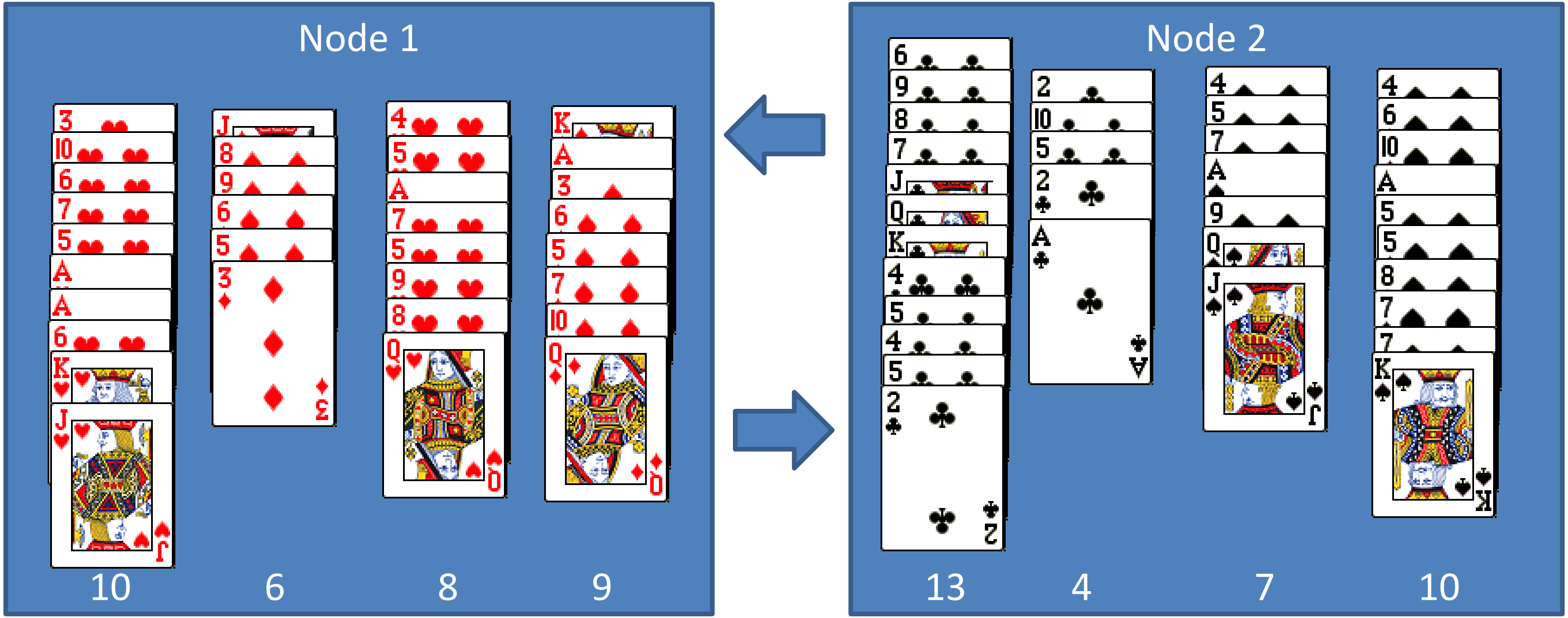


Mapping: Node Sorting



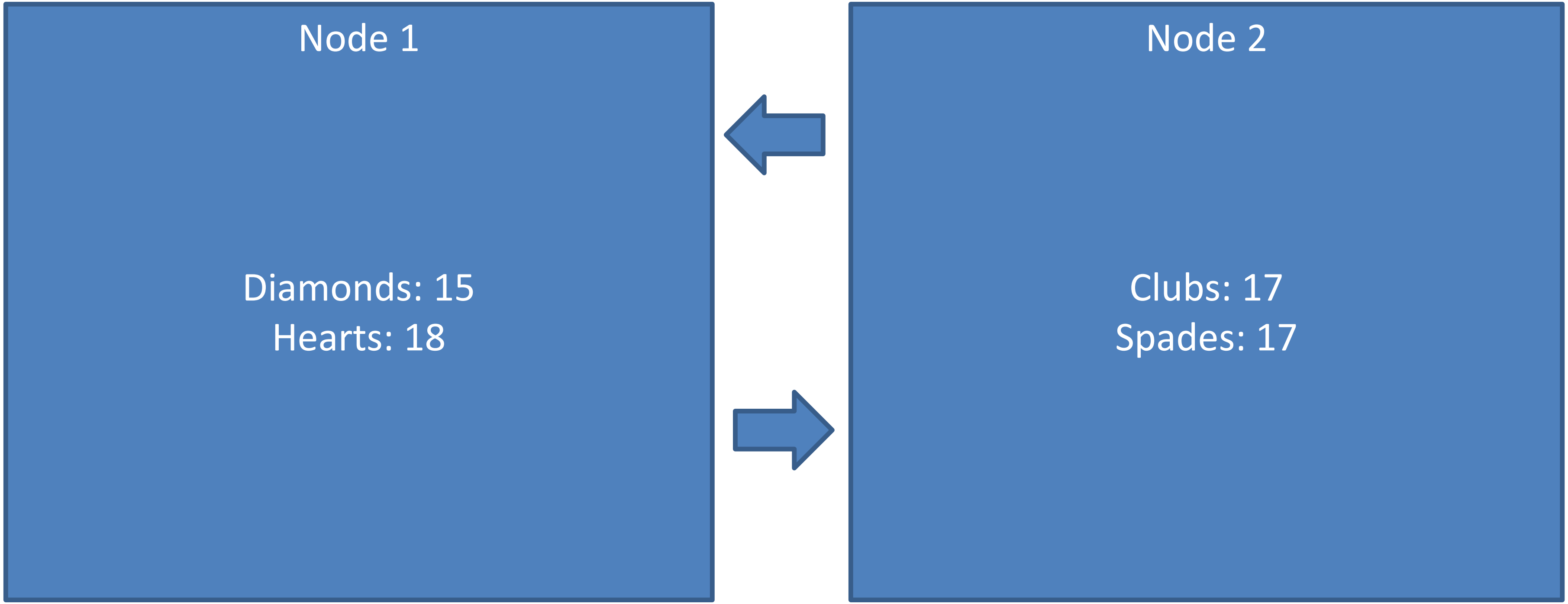
datasciencedojo
unleash the data scientist in you

Mapping: Node Shuffle, Data Transfer



datasciencedojo
unleash the data scientist in you

Mapping: Node Shuffle, Data Transfer



Database = Normalization

select * from **courses**;

id	name	teacher_id
10001	Computer Science 142	1234
10002	Computer Science 143	5678
10003	Computer Science 154	9012
10004	Informatics 100	1234

select * from **students**;

id	name	email	password
123	Bart	bart@fox.com	bartman
404	Ralph	ralph@fox.com	catfood
456	Milhouse	milhouse@fox.com	fallout
789	Nelson	muntz@fox.com	haha!
888	Lisa	lisa@gmail.com	vegan

select * from **grades**;

student_ids	course_id	grade
123	10001	B-
123	10002	C
456	10001	B+
888	10002	A+
888	10003	A+
404	10004	D+
456	10002	D-
404	10002	B
789	10003	D+

Normalization, joining

```
select
    g.student_id,
    c.name as course,
    g.grade as grade
from grades as g
join courses c
    on g.course_id = c.id
join students s
    on g.student_id = s.id
;
```

student	course	grade
Bart	Computer Science 142	B-
Milhouse	Computer Science 142	B+
Bart	Computer Science 143	C
Lisa	Computer Science 143	A+
Milhouse	Computer Science 143	D-
Ralph	Computer Science 143	B
Lisa	Computer Science 154	A+
Nelson	Computer Science 154	D+
Ralph	Informatics 100	D+

Data Warehouse = Denormalization

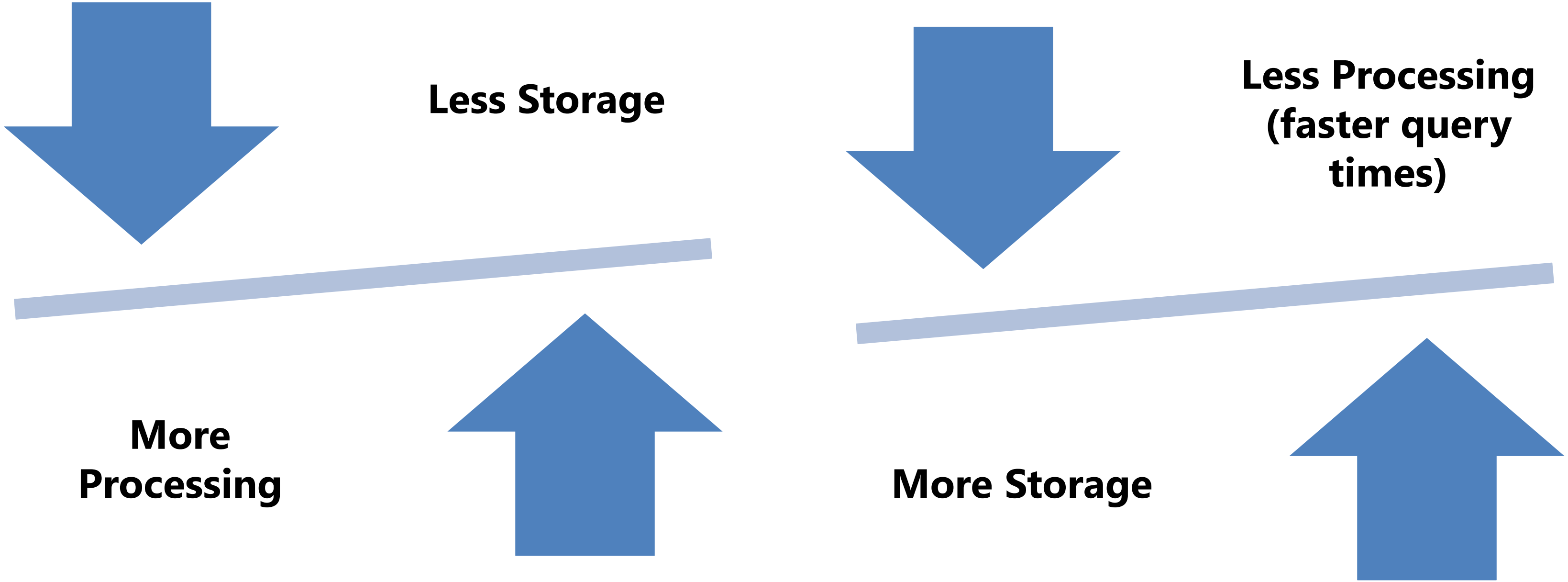
student	course	grade
Bart	Computer Science 142	B-
Milhouse	Computer Science 142	B+
Bart	Computer Science 143	C
Lisa	Computer Science 143	A+
Milhouse	Computer Science 143	D-
Ralph	Computer Science 143	B
Lisa	Computer Science 154	A+
Nelson	Computer Science 154	D+
Ralph	Informatics 100	D+

Tables:
- Students Table
- Courses Table
- Roster Table

Trade-Offs

Normalization

Denormalization



Costs, Storage vs Processing

Storage

US – N. Virginia	US – N. California	EU – Ireland

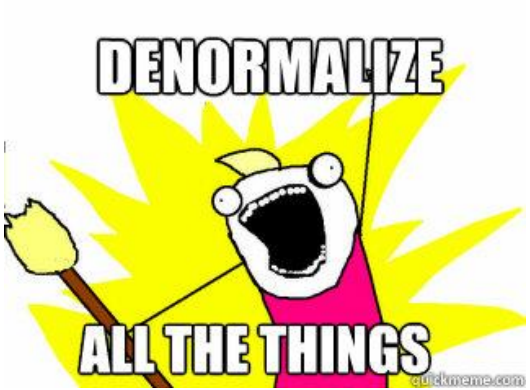
Standard On-Demand Instances	Linux/UNIX Usage	Windows Usage
Small (Default)	$0.085 per hour	$0.12 per hour
Large	$0.34 per hour	$0.48 per hour
Extra Large	$0.68 per hour	$0.96 per hour

Processing

US – Standard | US –

Storage

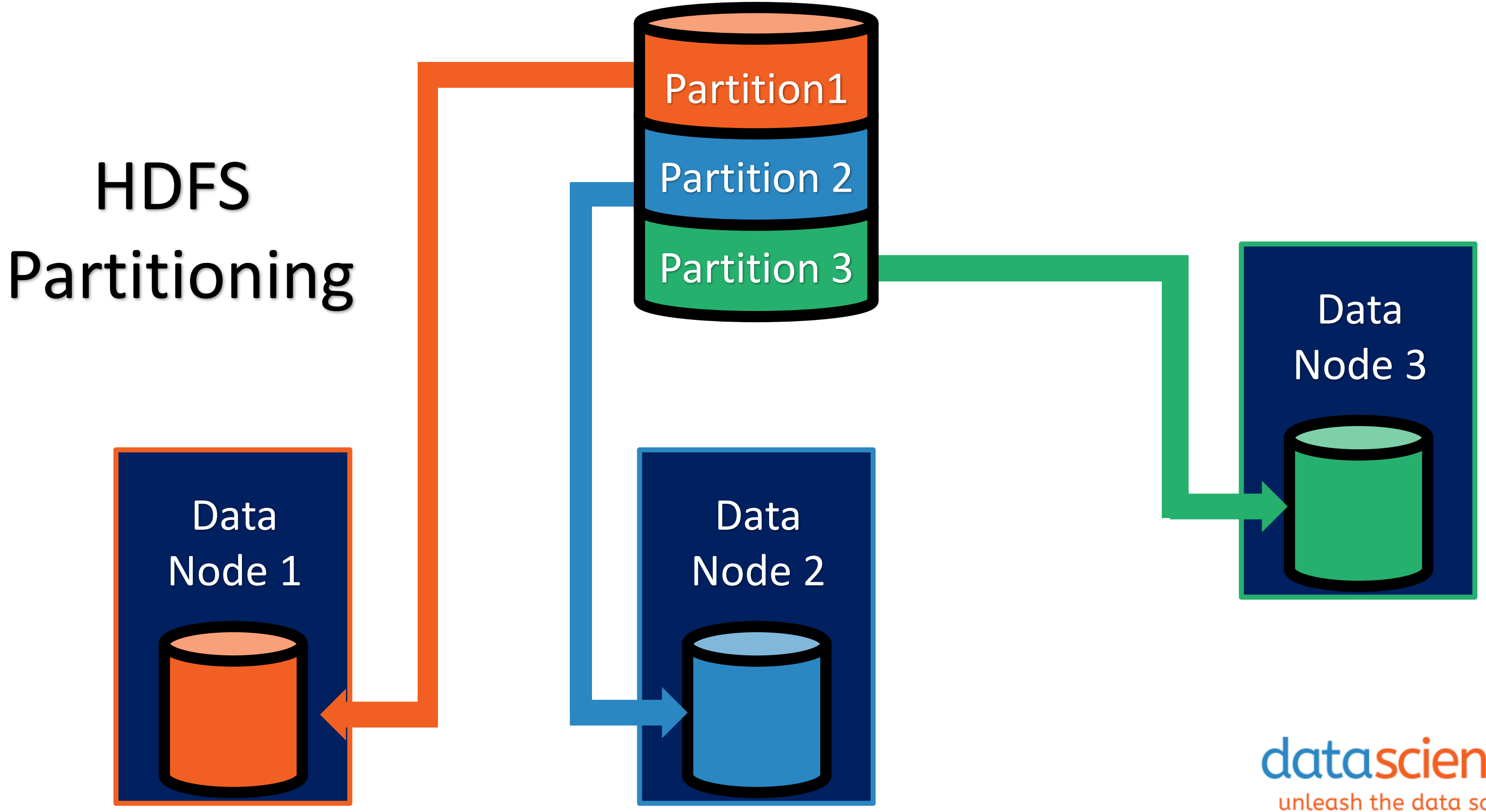
Tier	Pricing
First 50 TB / Month of Storage Used	$0.150 per GB
Next 50 TB / Month of Storage Used	$0.140 per GB
Next 400 TB /	$0.130 per GB



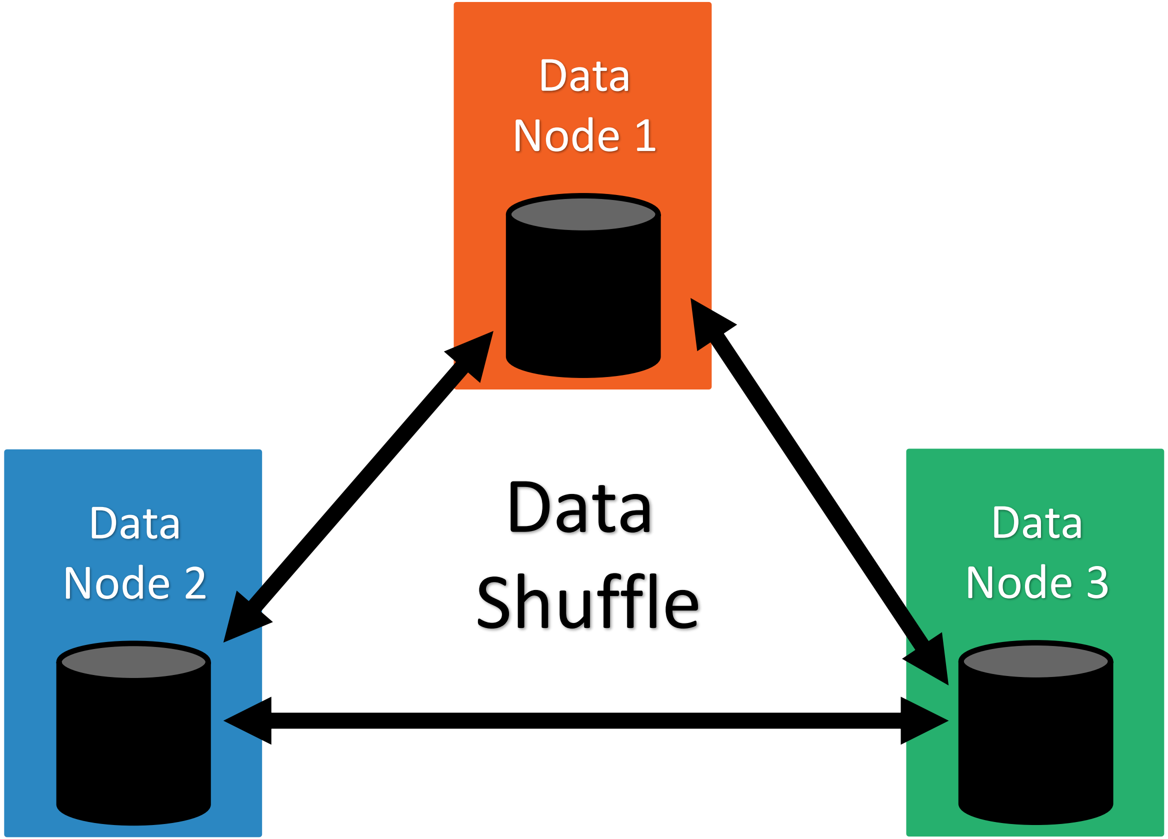
mahout

- Distributed Machine Learning
- Installed into Hadoop & Spark
- R-like language Implementation

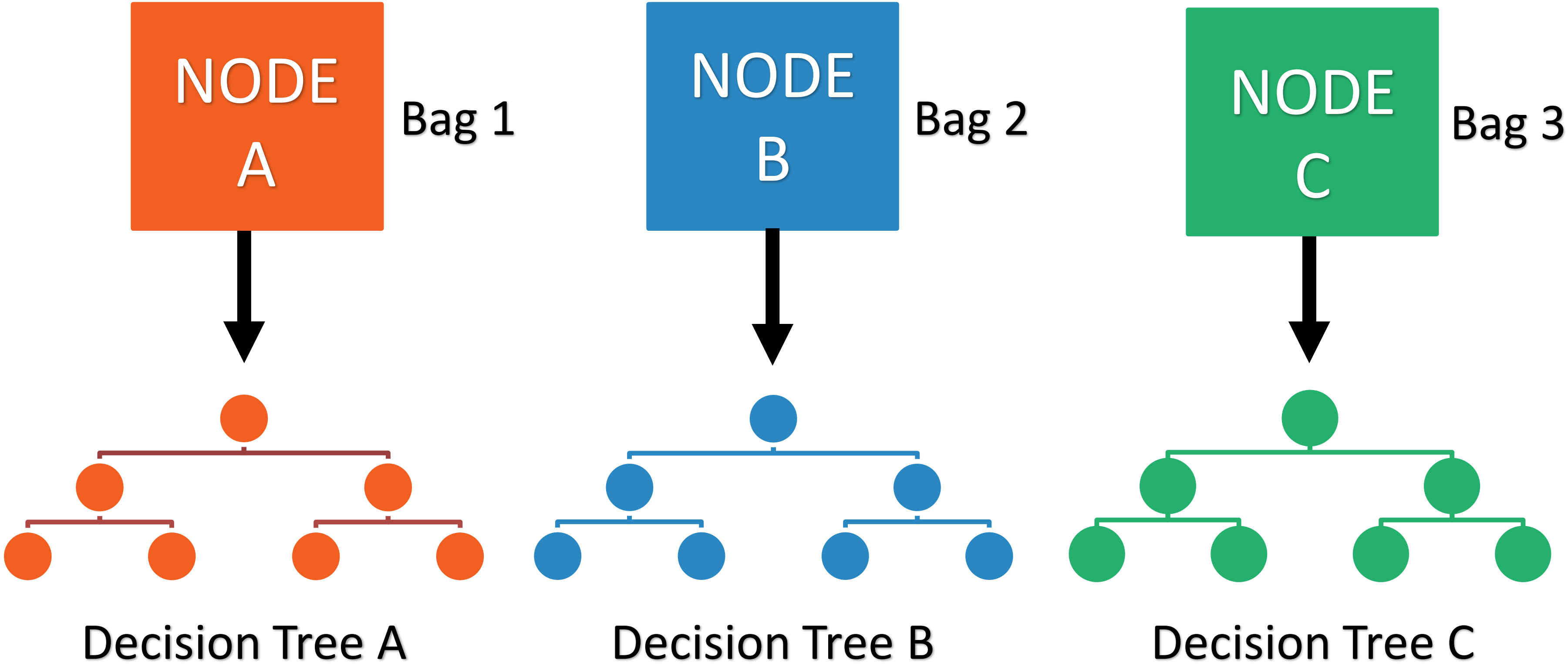
Distributed Random Forest



Distributed Random Forest



Distributed Random Forest



datasciencedojo
unleash the data scientist in you

Classification

	PC	Mahout	Spark
Logistic Regression - trained via SGD		<10m	
Naive Bayes			
Random Forest			
Hidden Markov Models			
Multilayer Perceptron			

Source: https://mahout.apache.org/users/basics/algorithms.html

Recommendation Engines

	PC	Mahout	Spark
User-Based Collaborative Filtering			
Item-Based Collaborative Filtering			
Matrix Factorization with ALS			
Matrix Factorization with ALS on Implicit Feedback			
Weighted Matrix Factorization, SVD++			

Source: https://mahout.apache.org/users/basics/algorithms.html

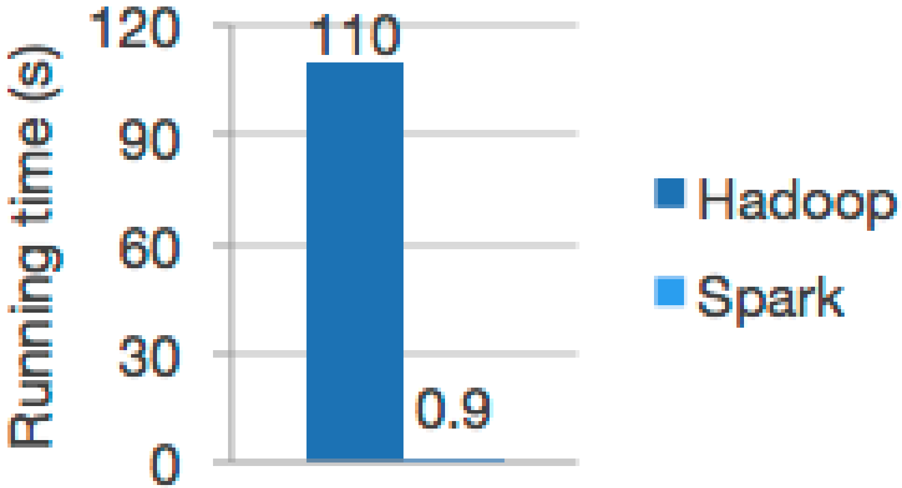
datasciencedojo
unleash the data scientist in you

Clustering

	PC	Mahout	Spark
k-Means Clustering			
Fuzzy k-Means			
Streaming k-Means			
Spectral Clustering			

Source: https://mahout.apache.org/users/basics/algorithms.html

Spark



In-Memory: 100x times faster than Hadoop

3x faster on 10x few machines

Datona GraySort Benchmark: Sort 100 TB of data

Previous World Record:

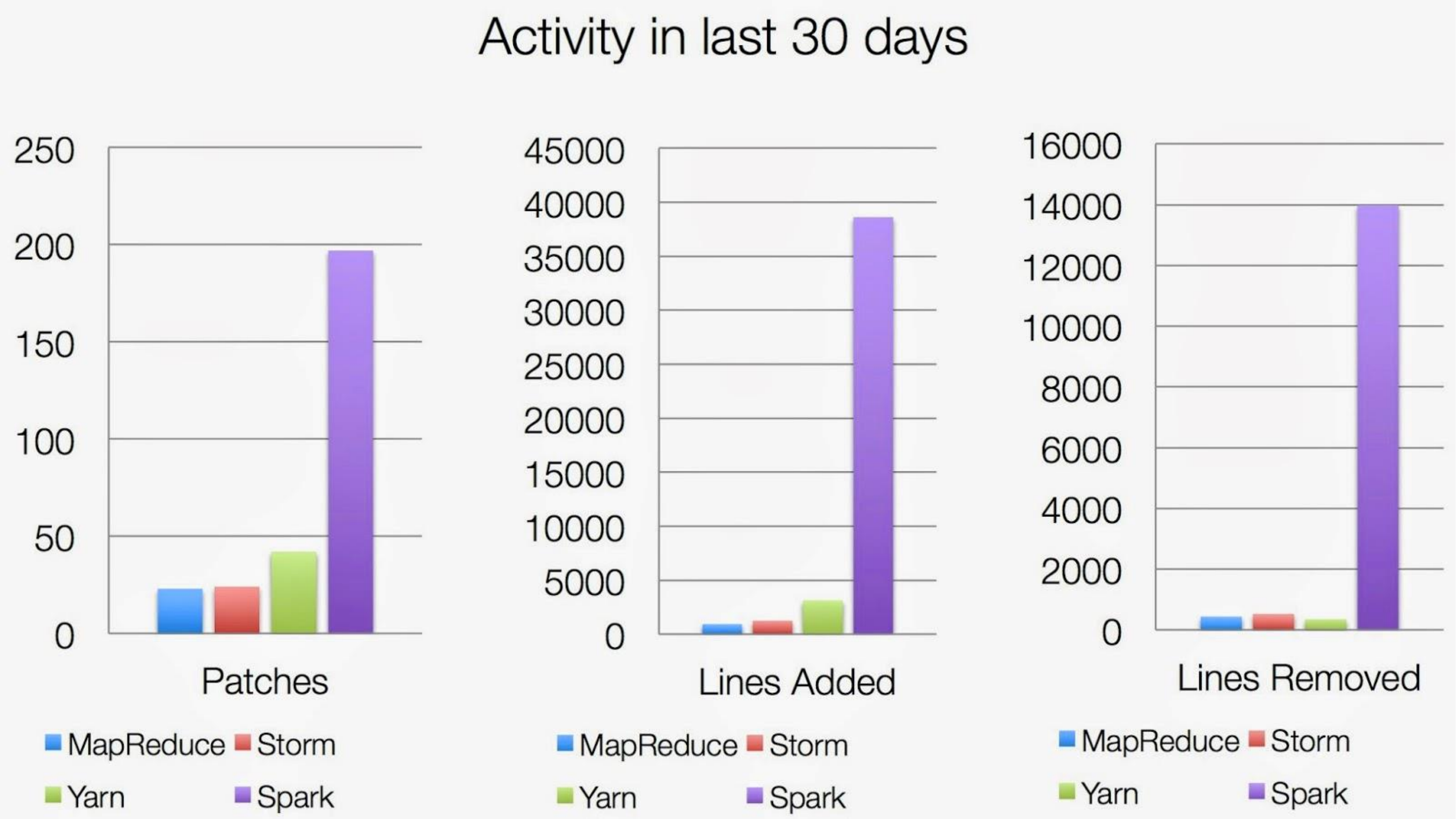
- Method: Hadoop
- Yahoo!
- 72 Minutes
- 2100 Nodes

2014:

- Method: Spark
- Databricks
- 23 Minutes
- 206 Nodes

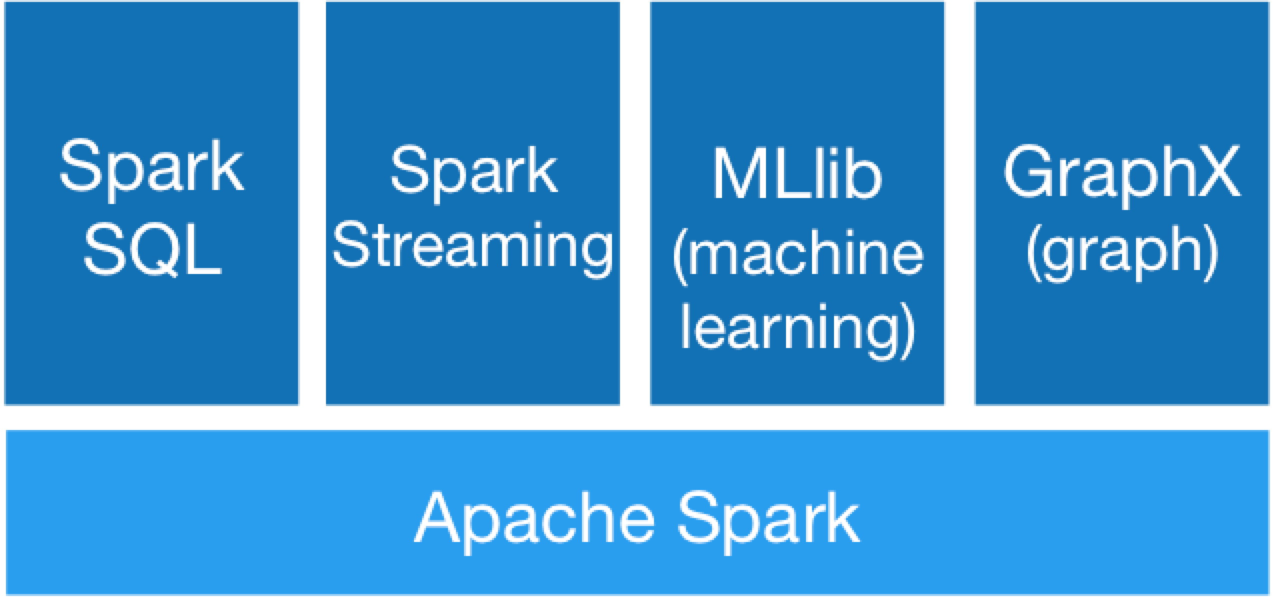
Source: https://databricks.com/blog/2014/10/10/spark-petabyte-sort.html

datasciencedojo
unleash the data scientist in you

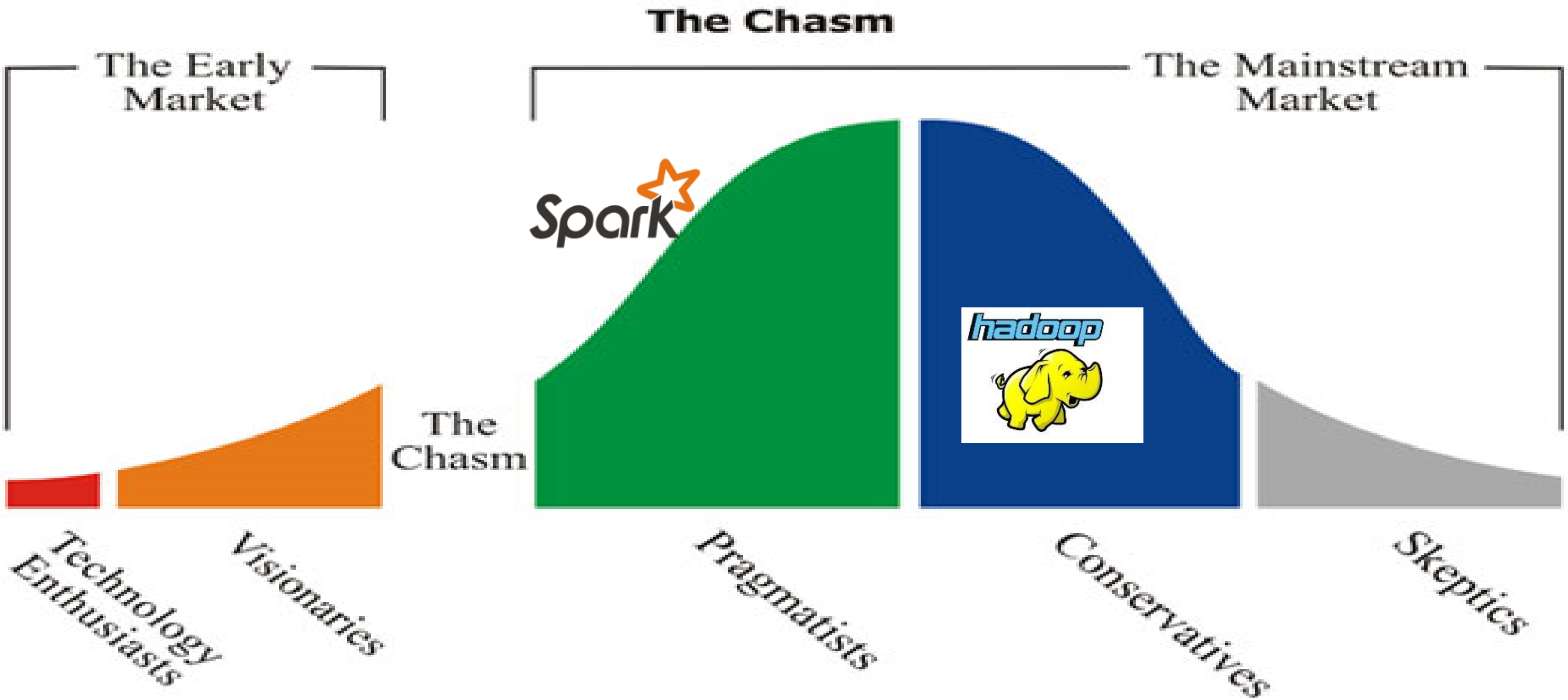


Source: Xiangrui Meng, Data Bricks

Spark


Spark SQL
Spark Streaming
MLlib (machine learning)
GraphX (graph)
Apache Spark

Technology adoption life cycle



Source: http://carlosmartinezt.com/2010/06/technology-adoption-life-cycle/

datasciencedojo
unleash the data scientist in you

QUESTIONS