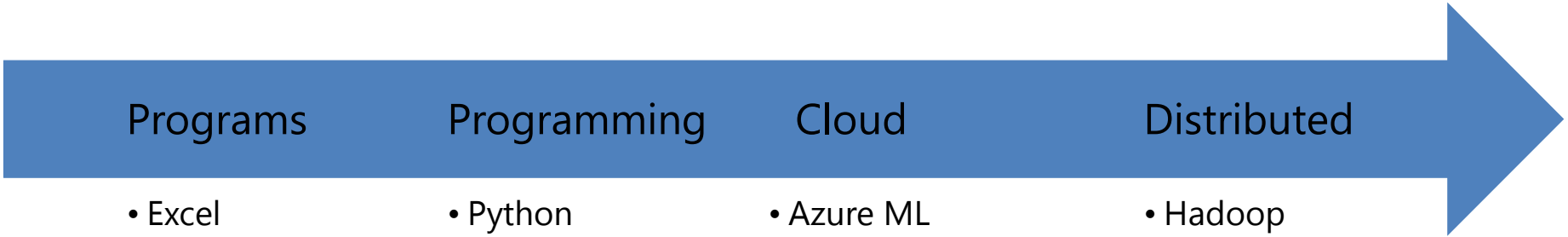
Big Data Engineering With MapReduce and Hive

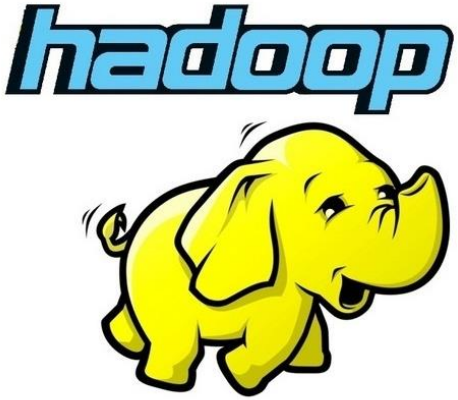
Data Science Dojo

Machine Learning Scaling



- Excel
- Python
- R
- SAS
- Azure ML
- AWS ML
- Watson Analytics
- Big ML
- R Shiny
- Cloud Virtual Machines
- Hadoop
- Spark
- H20
- Revolution R

Agenda



From a Data Scientist's Perspective

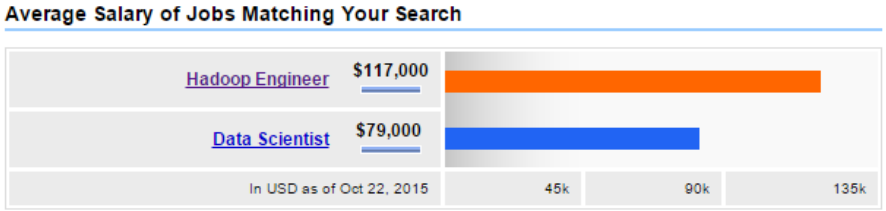
Goals:

- Teach you how to leverage an existing Hadoop cluster, self-service data query

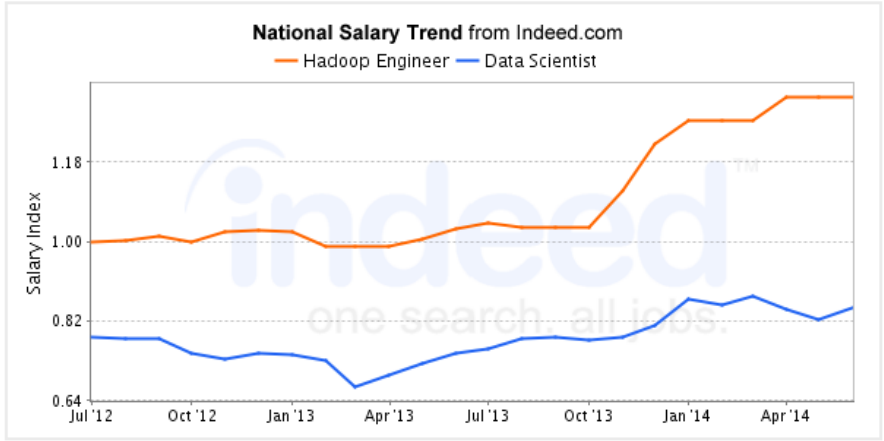
Not goals:

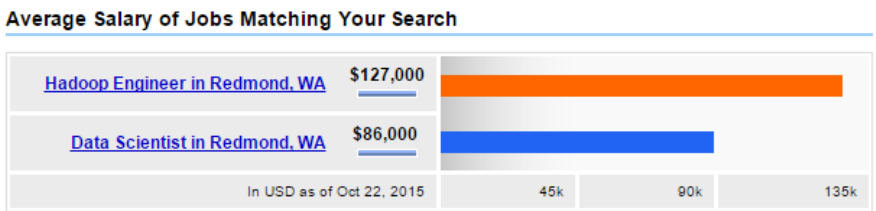
- Managing or administering a Hadoop cluster

Hadoop Engineers

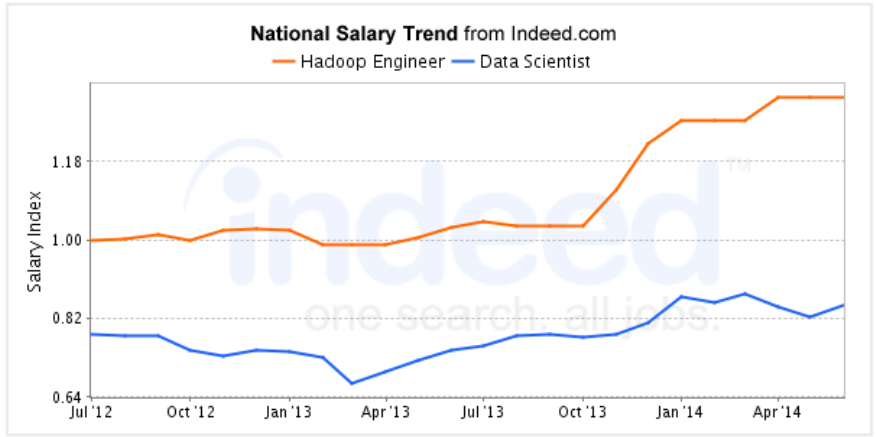


Average Hadoop Engineer salaries for job postings nationwide are 47% higher than average Data Scientist salaries for job postings nationwide.





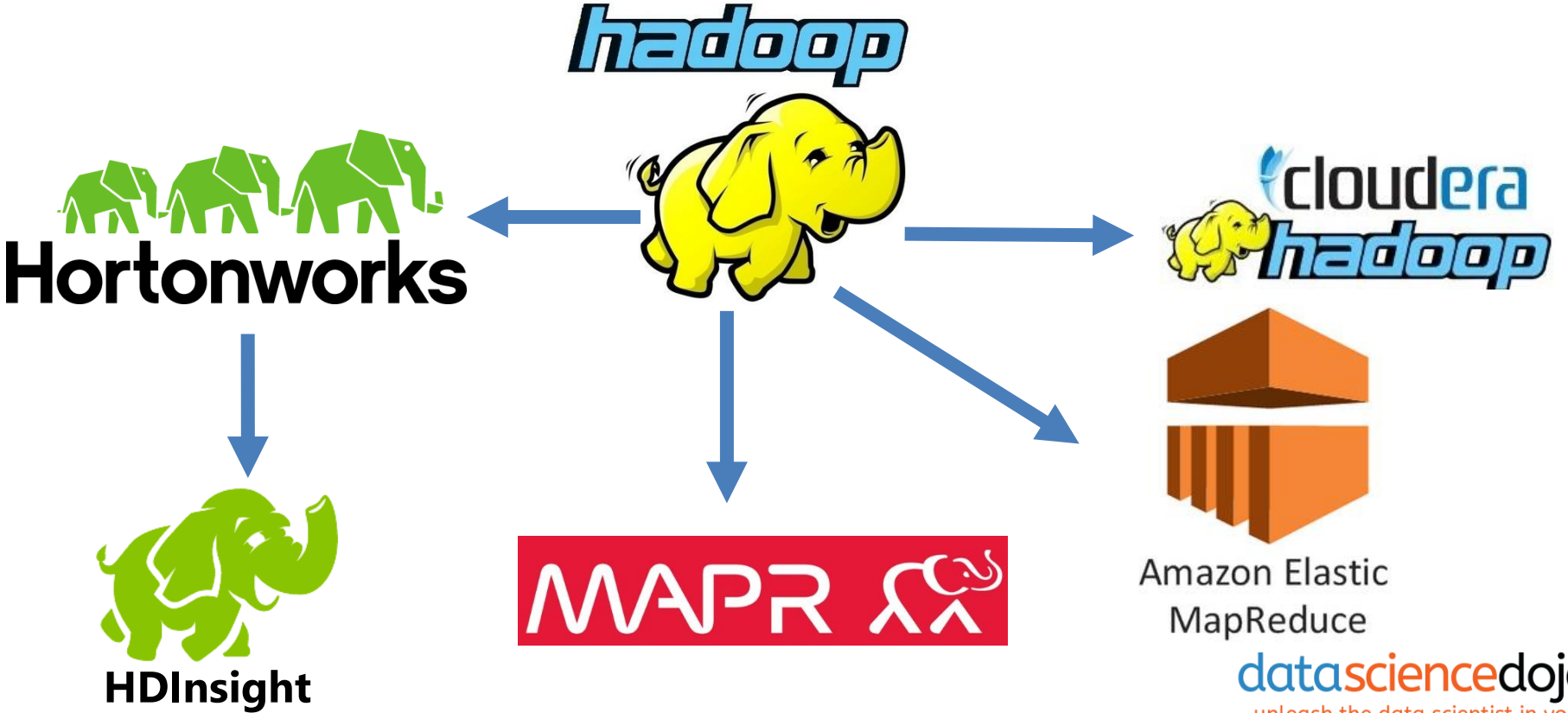
Average Hadoop Engineer salaries for job postings in Redmond, WA are 47% higher than average Data Scientist salaries for job postings in Redmond, WA.



Source: Ineed.com

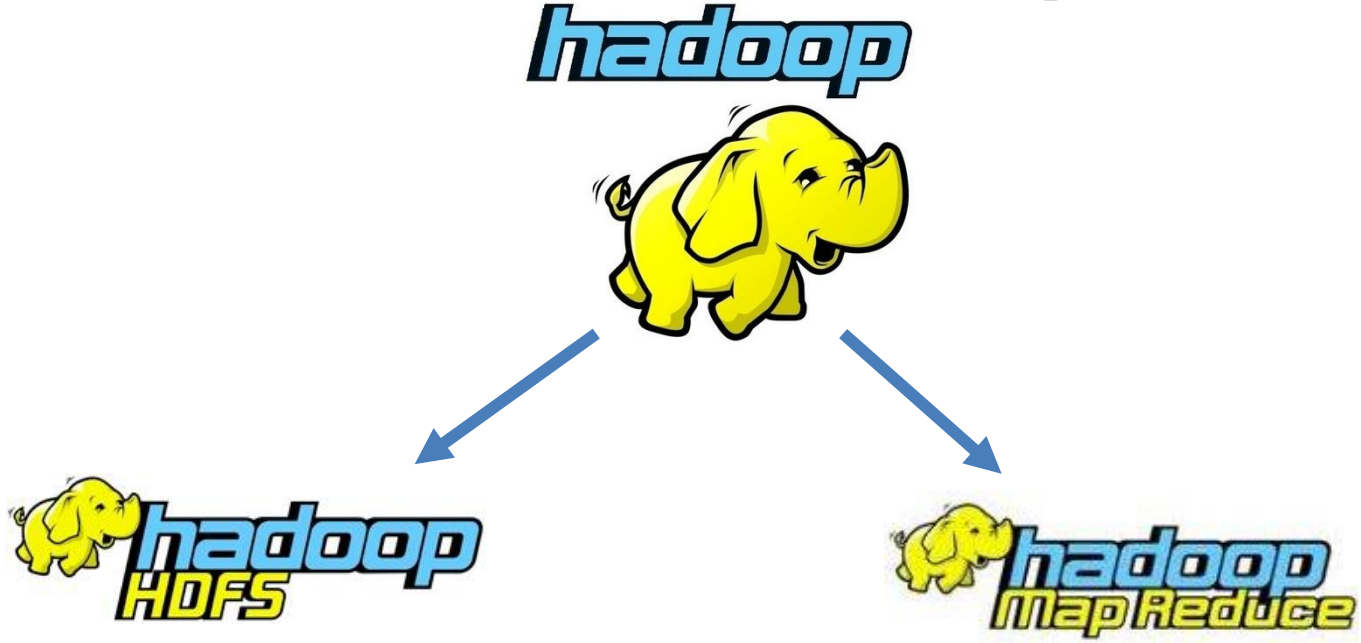
datasciencedojo
unleash the data scientist in you

Hadoop Implementations



datasciencedojo
unleash the data scientist in you

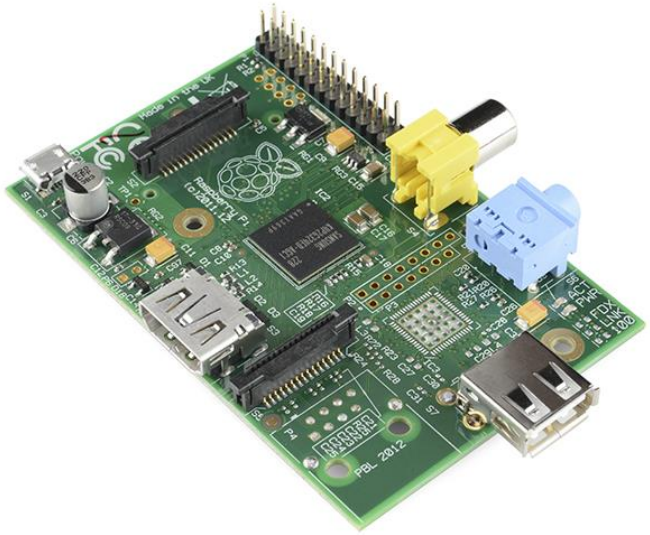
(Vanilla/Base) Hadoop

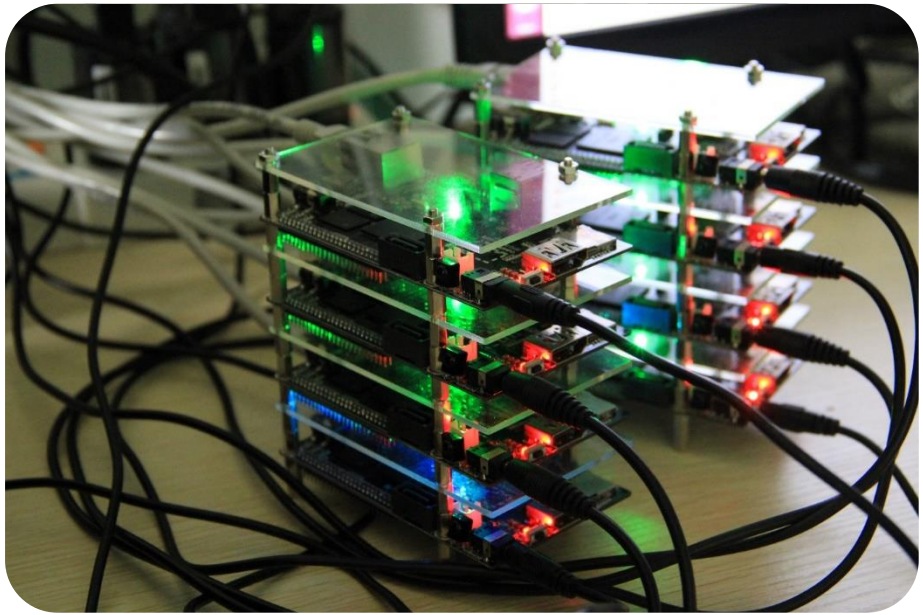


Processing engine for distributed batch processing.

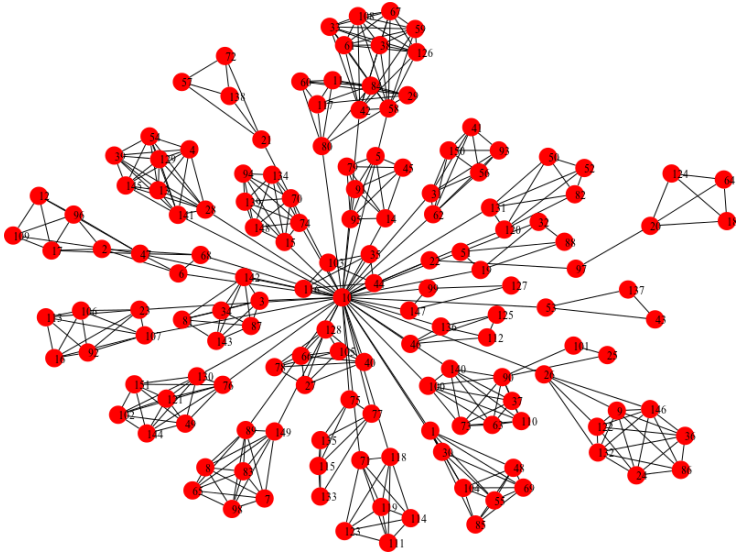
Turn Back The Clock, The Mainframe

Distributed Computing

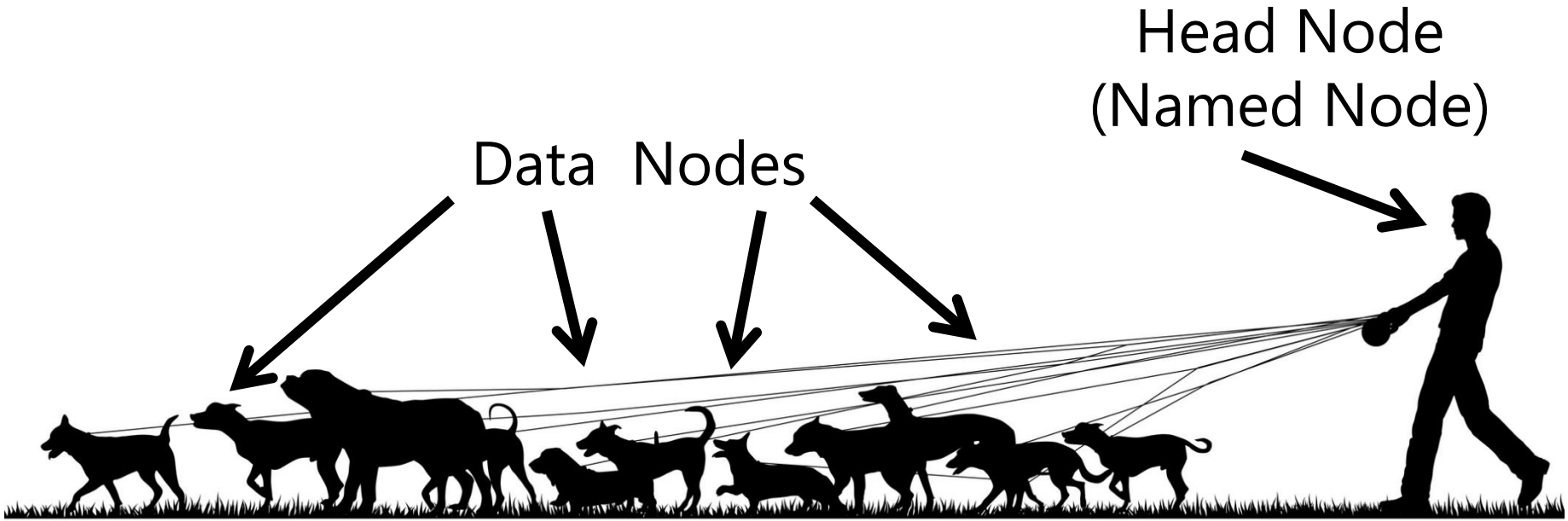




Cloud Computing



If dogs were servers…



HDFS & MapReduce

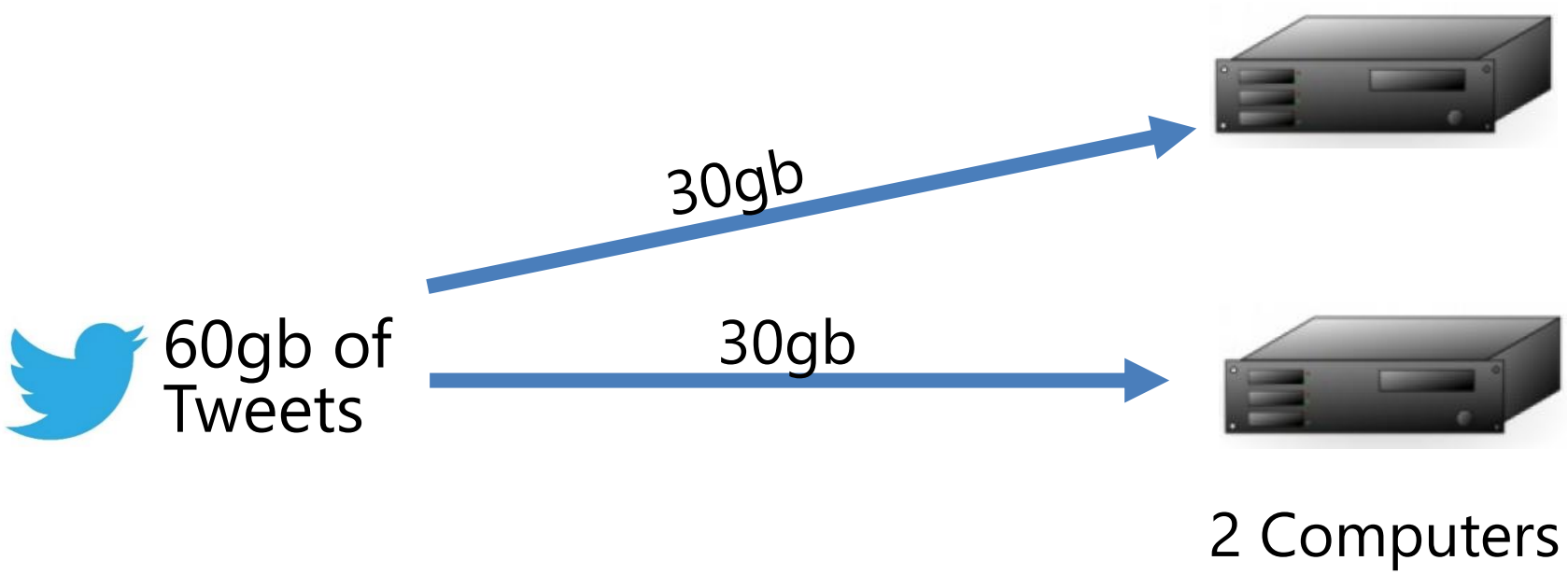
60gb of Tweets

60gb

1 Computer

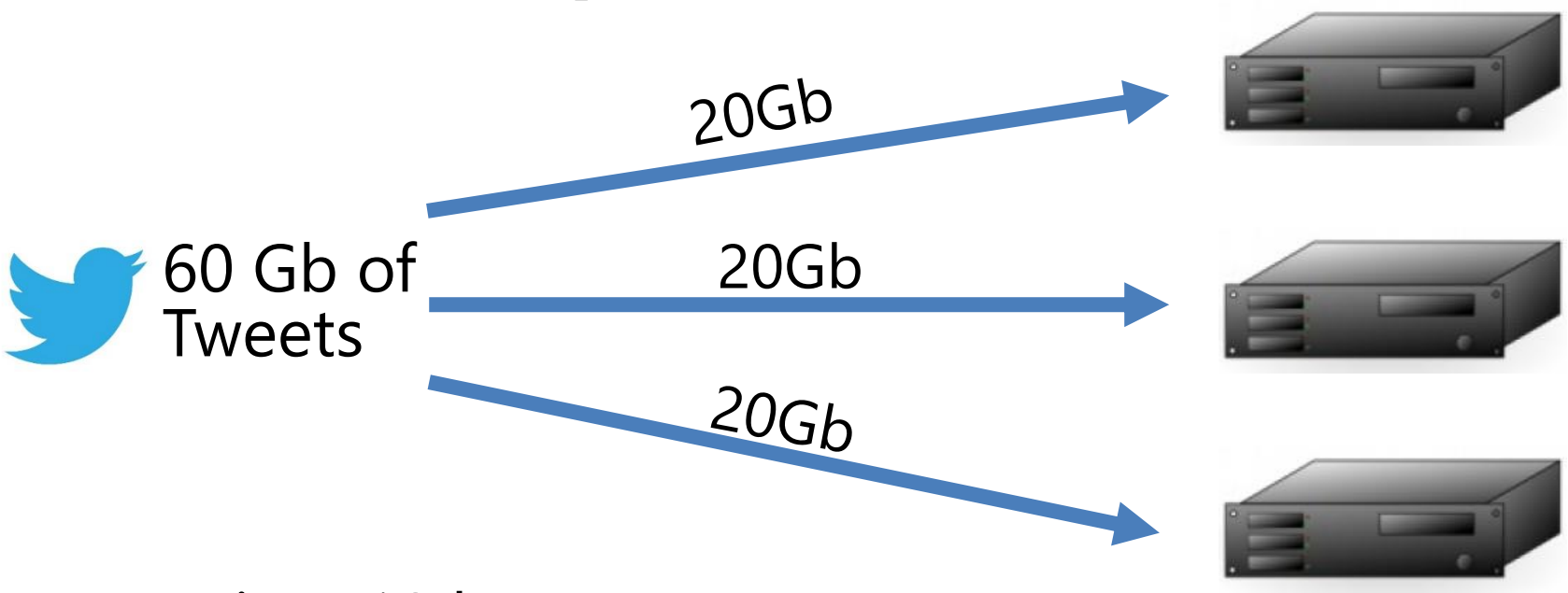
Processing: 30 hours

HDFS & MapReduce



Processing: 15 hours

HDFS & MapReduce



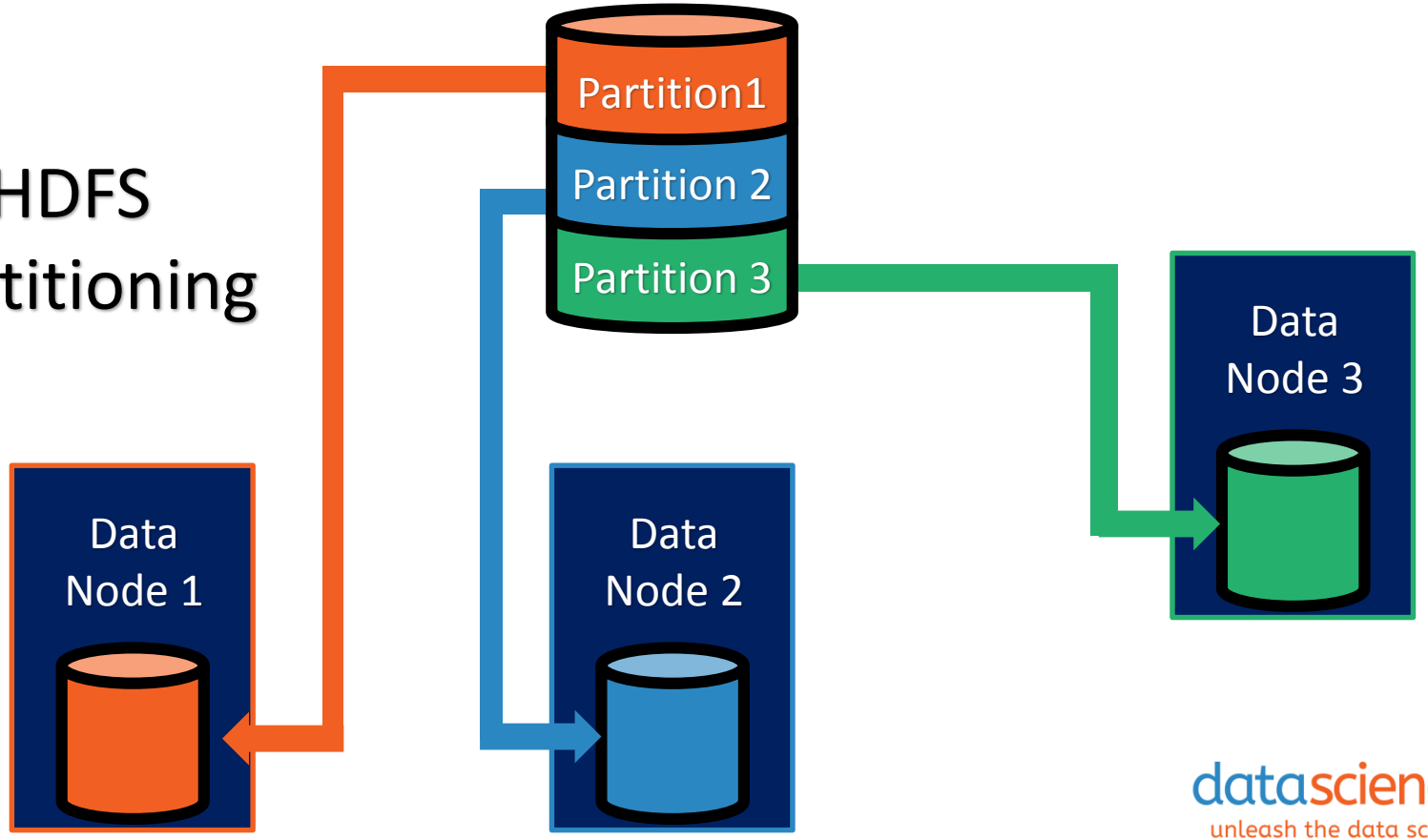
Processing: 10 hours

Most Cases, Linear Scaling Of Processing Power

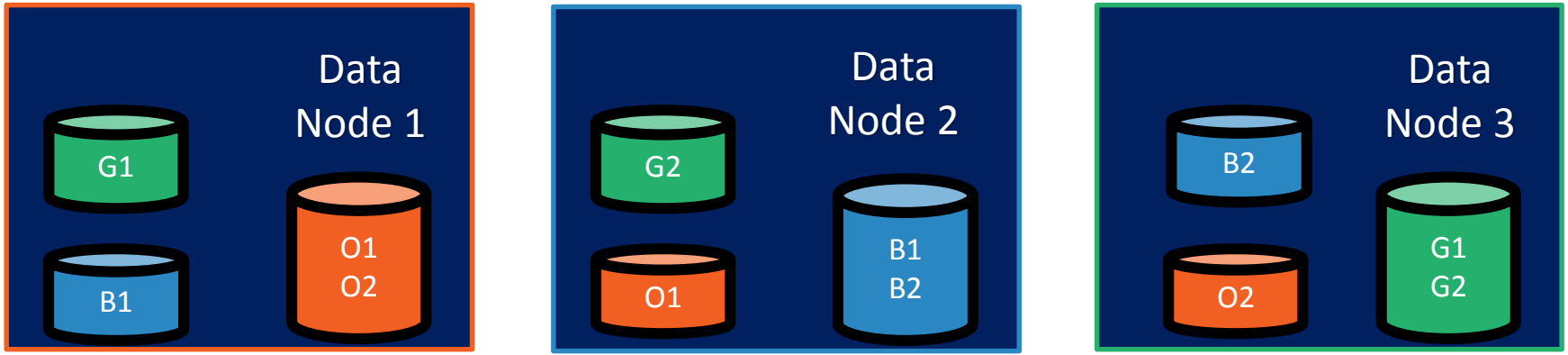
Number of Computers	Processing Time (hours)
1	30
2	15
3	10
4	7.5
5	6
6	5
7	4.26
8	3.75
9	3.33

HDFS

HDFS Partitioning



HDFS Redundancy



Limitations with MapReduce

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

```
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

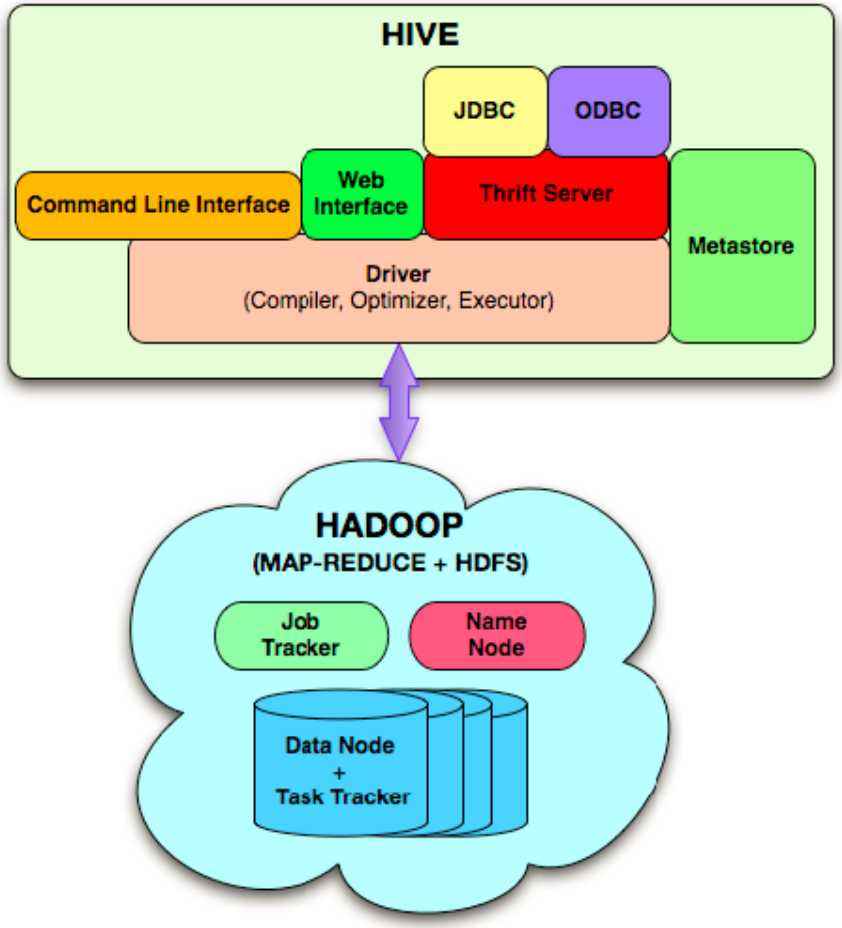
Oozie: Workflow management

Pig: Simpler scripting for MapReduce transformations

Sqoop: Data import and export

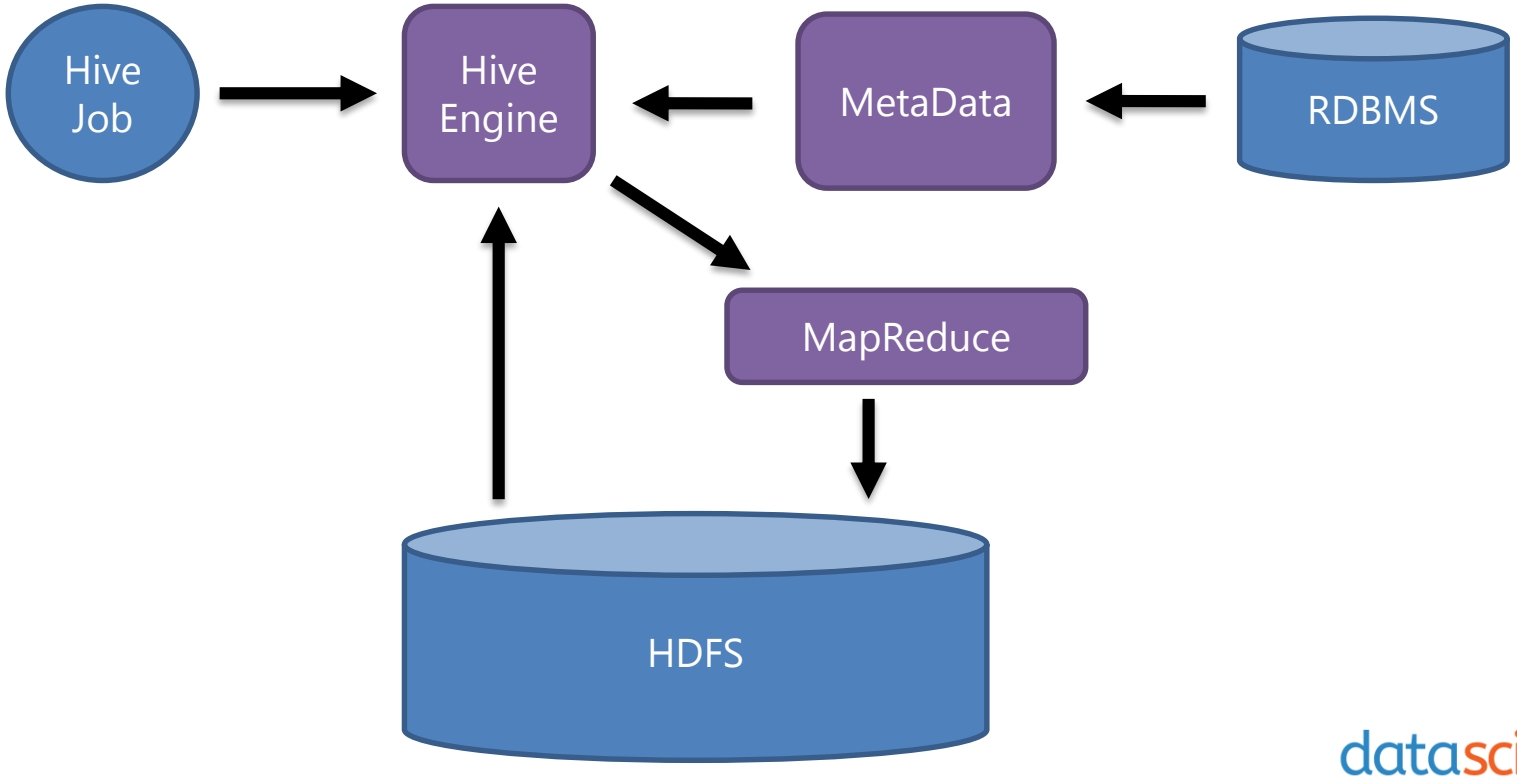
Storm: Real-time processing of fast, large data streams

Zookeeper: Coordinates processes in distributed systems



Hive Jobs

Hive Architecture



Data File

=

Unstructured Data

Data File

+

Metadata File/DB

=

Structured Data

Semi Structured Data

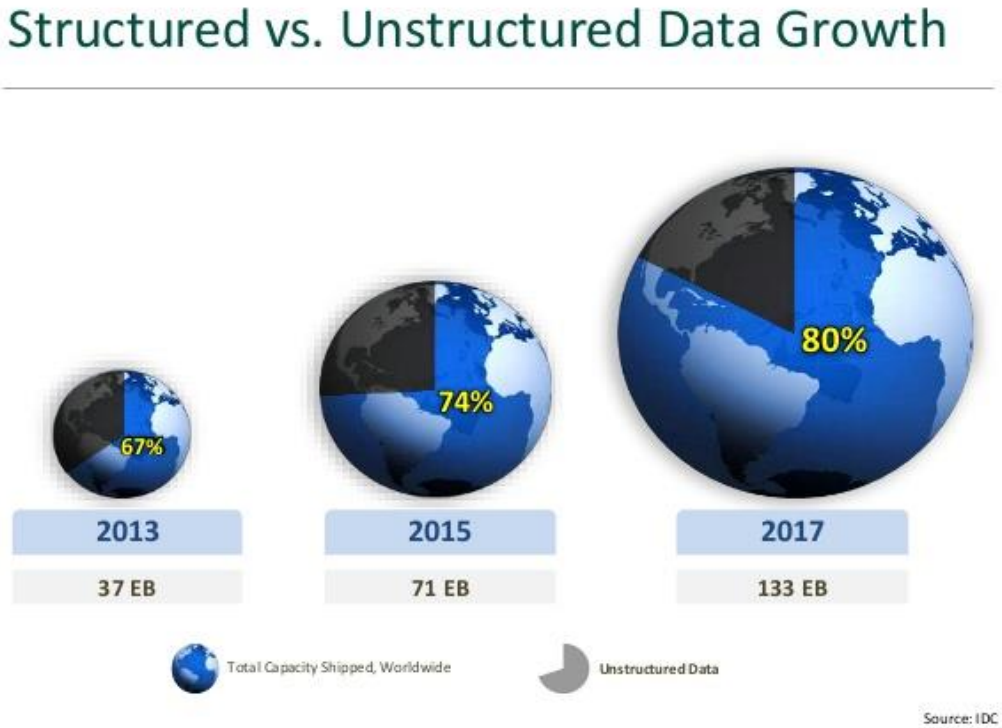
Self Describing Flat Files

- XML
- JSON
- CSV
- TSV

Why Hive?

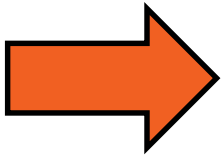
- SQL spoken here (HiveQL)
- ODBC driver
- BI Integration
- Supports only Structured Data

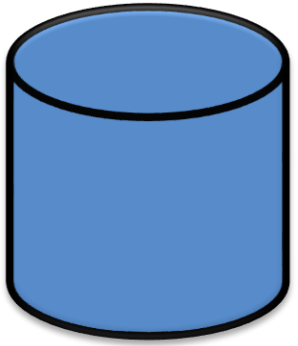
Limitations



Azure Blob Storage

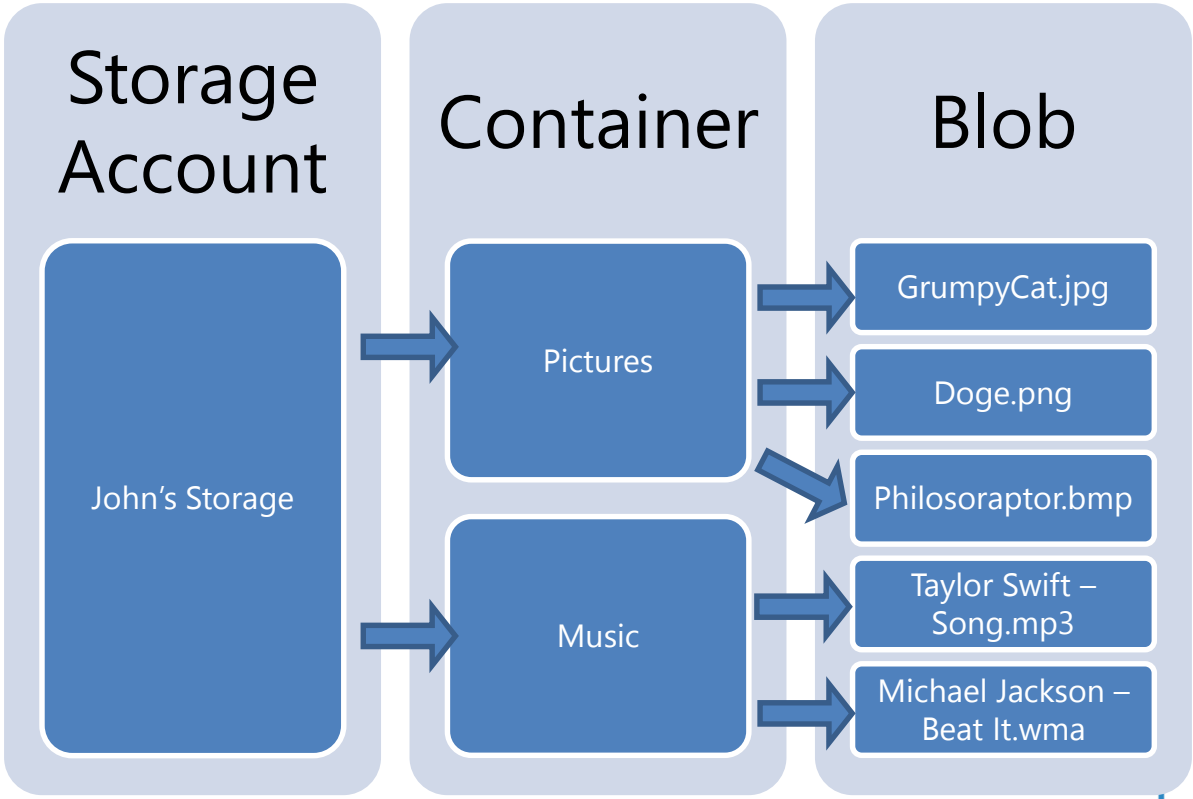
HDInsight



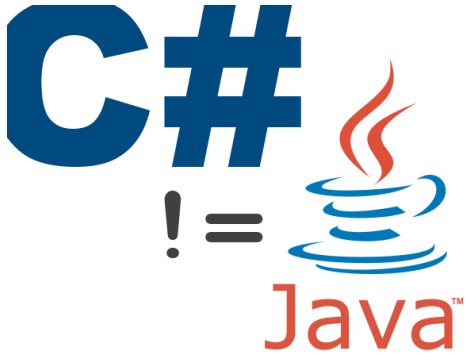


Blob Storage

Azure Blob Storage

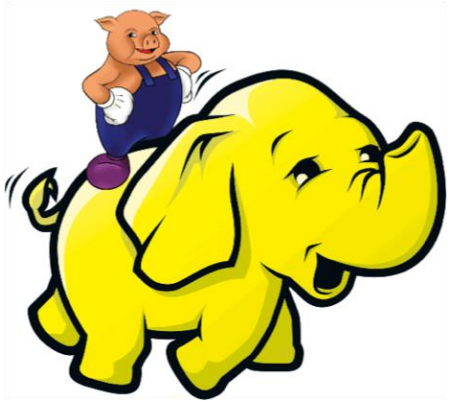


When to Use Each



vs

vs



C#
Java
MapReduce

Hive

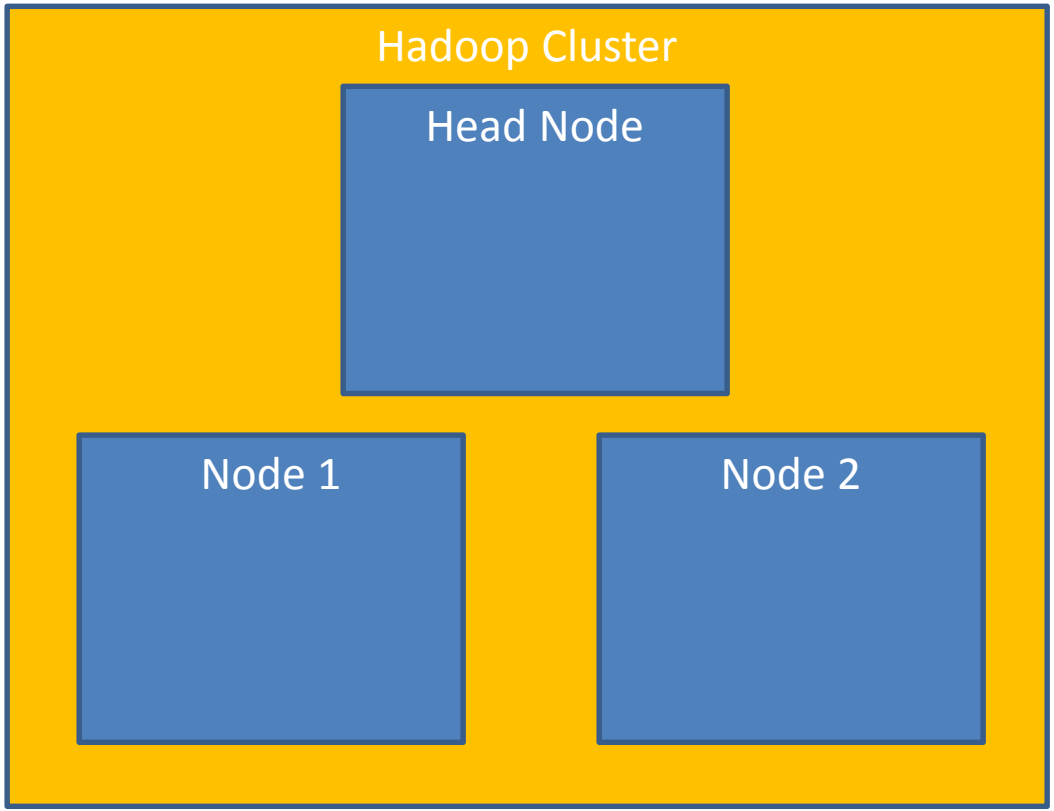
Pig

MapReduce, via Playing Cards

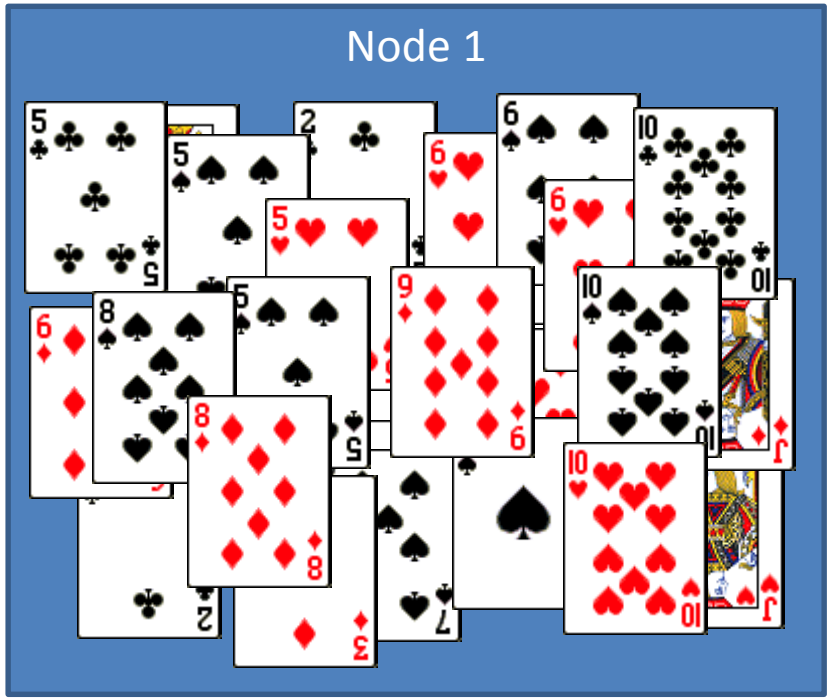
Let's count the number of spades, clubs, hearts, and diamonds in a stack of cards, the way map reduce would.

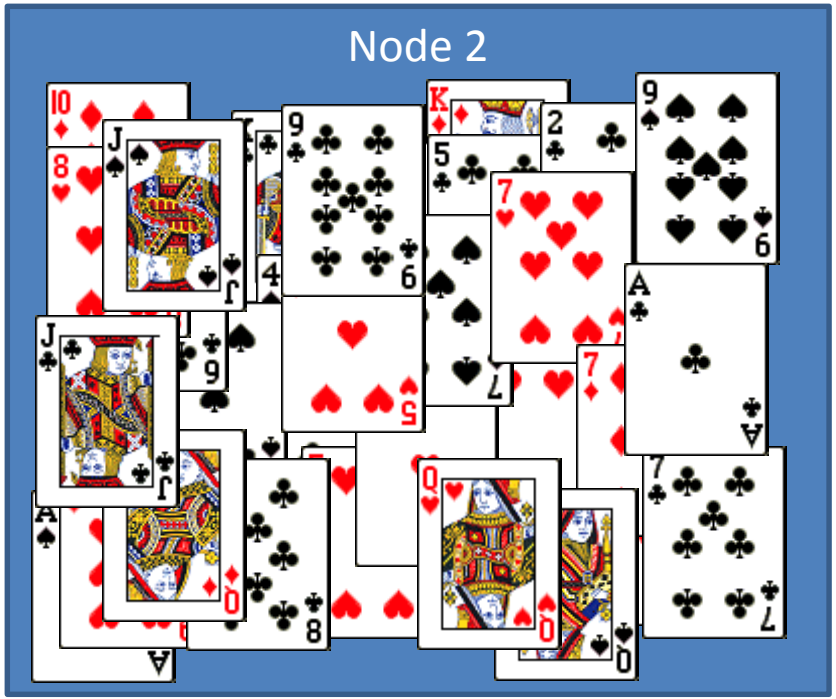
- Each card represents a row of data
- Each suite represents an attribute of the data

Using a 2 Data Node Cluster

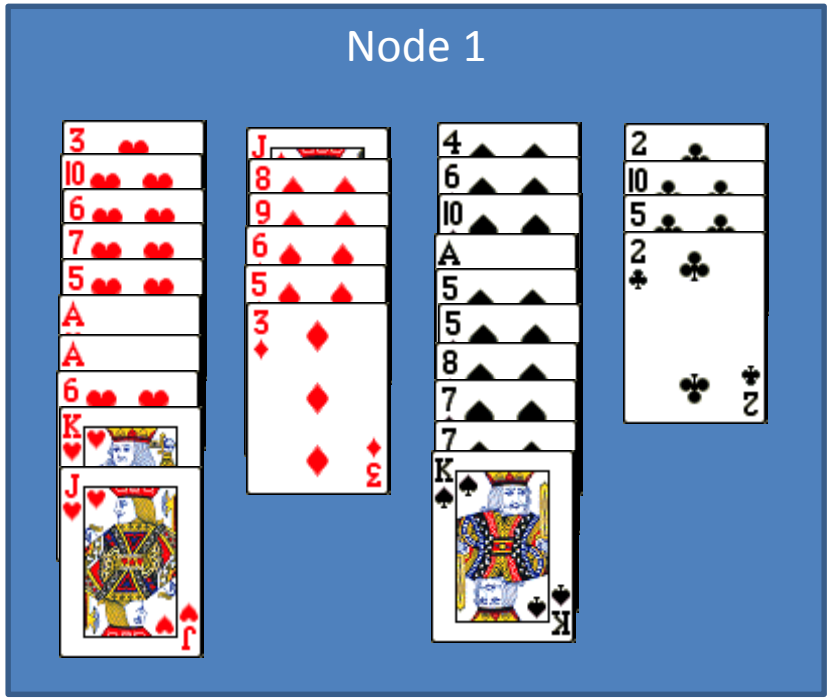


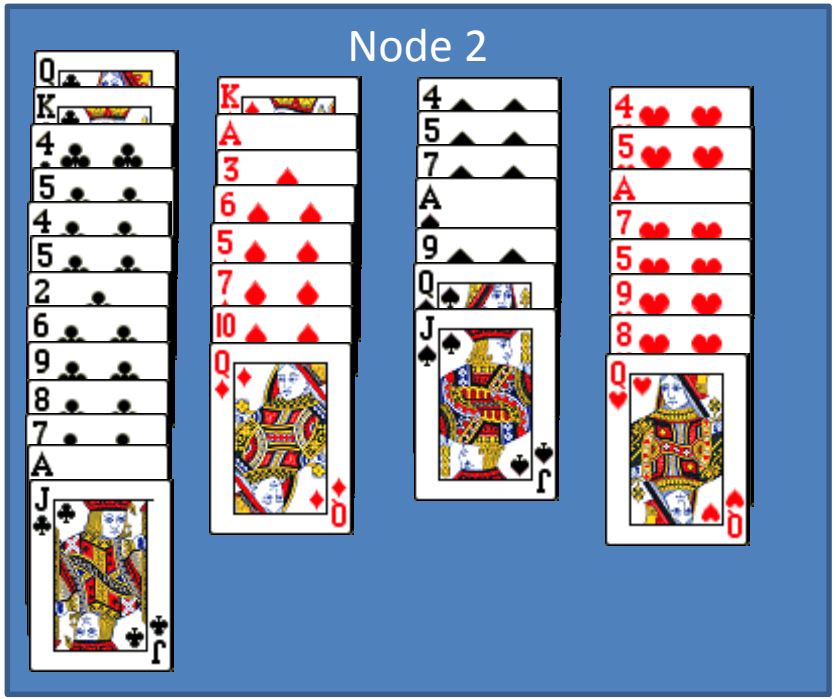
Mapping: Each Node's HDFS



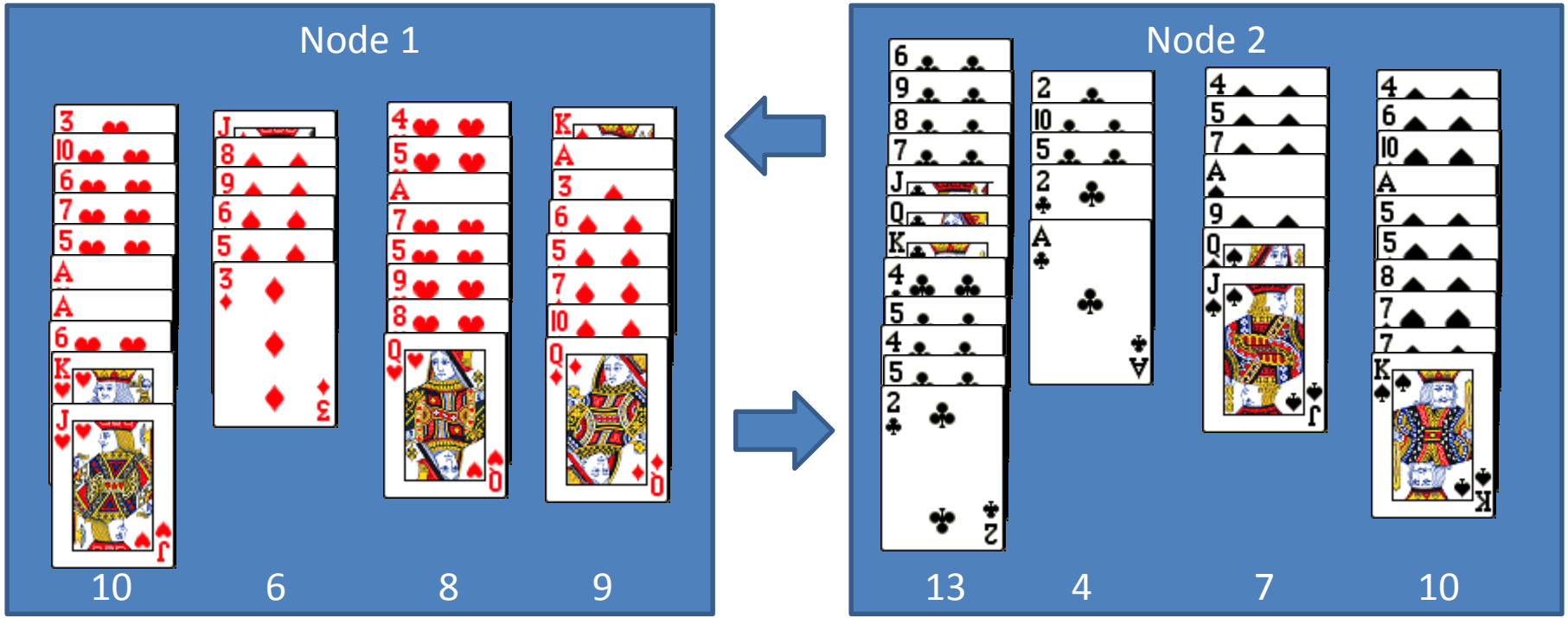


Mapping: Node Sorting

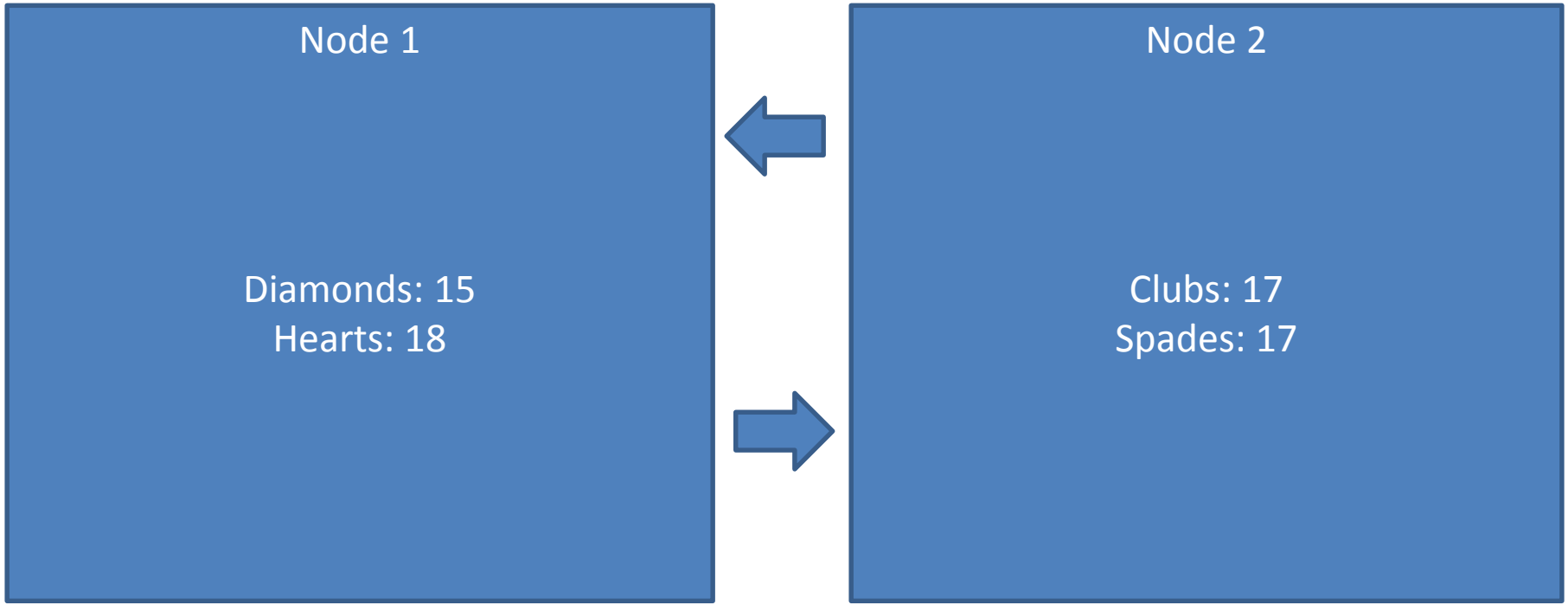




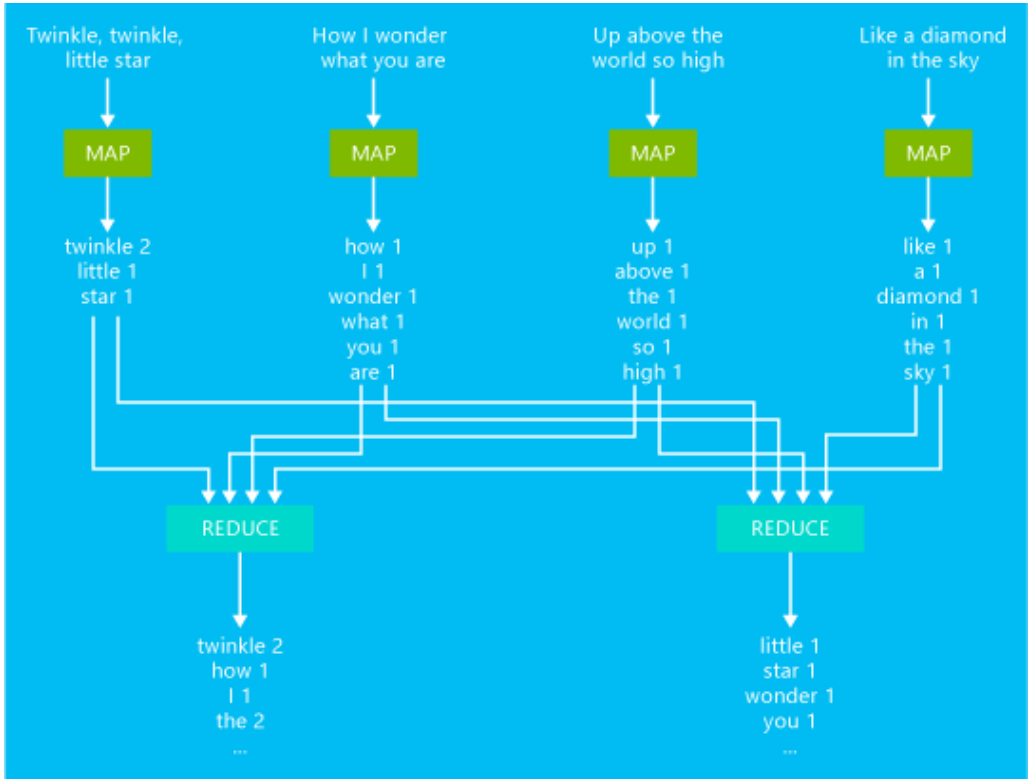
Mapping: Node Shuffle, Data Transfer



Mapping: Node Shuffle, Data Transfer



Word Count, via MapReduce()



Source: https://azure.microsoft.com/en-us/documentation/articles/hdinsight-use-mapreduce/

Databases

movie	year	rating	director
Aliens	1986	8.2	James (I) Cameron
Animal House	1978	7.5	John (I) Landis
Apollo 13	1995	7.5	Ron Howard
Batman Begins	2005	NULL	Christopher Nolan
Braveheart	1995	8.3	Mel (I) Gibson
Fargo	1996	8.2	Ethan Coen
Fargo	1996	8.2	Joel Coen
Few Good Men, A	1992	7.5	Rob Reiner
Fight Club	1999	8.5	David Fincher

Normalization, joining

```
SELECT
   m.name AS movie,
   m.year AS year,
   m.rank AS rating,
   CONCAT(d.first_name, " ", d.last_name)
      AS director
FROM movies AS m
JOIN movies_directors AS md
   ON m.id = md.movie_id
JOIN directors AS d
   ON md.director_id = d.id
;
```

Movie Information

movie	year	rating	director
Aliens	1986	8.2	James (I) Cameron
Animal House	1978	7.5	John (I) Landis
Apollo 13	1995	7.5	Ron Howard
Batman Begins	2005	NULL	Christopher Nolan
Braveheart	1995	8.3	Mel (I) Gibson
Fargo	1996	8.2	Ethan Coen
Fargo	1996	8.2	Joel Coen
Few Good Men, A	1992	7.5	Rob Reiner
Fight Club	1999	8.5	David Fincher

Database = Normalization

director

id	first_name	last_name
24758	David	Fincher
66965	Jay	Roach
72723	William	Shatner

movie_directors

director_id	movie_id
24758	112290
66965	209658
72723	313398

movies

id	name	year	rank
112290	Fight Club	1999	8.5
209658	Meet the Parents	2000	7
210511	Memento	2000	8.7

Data Warehouse = Denormalization

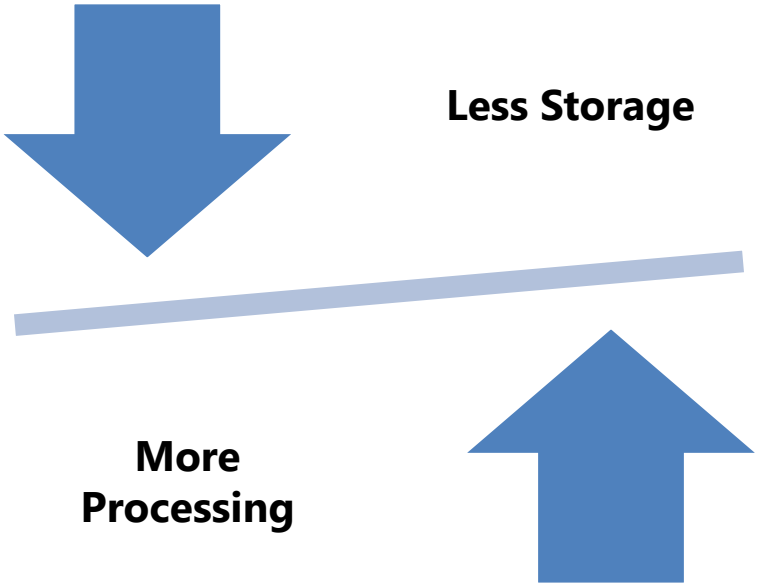
student	course	grade
Bart	Computer Science 142	B-
Milhouse	Computer Science 142	B+
Bart	Computer Science 143	C
Lisa	Computer Science 143	A+
Milhouse	Computer Science 143	D-
Ralph	Computer Science 143	B
Lisa	Computer Science 154	A+
Nelson	Computer Science 154	D+
Ralph	Informatics 100	D+

Tables:

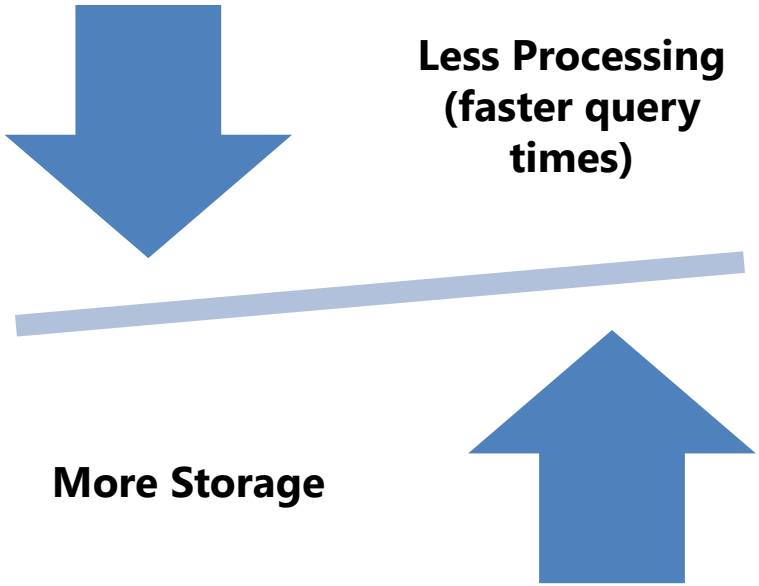
- Students Table
- Courses Table
- Roster Table

Trade-Offs

Normalization



Denormalization



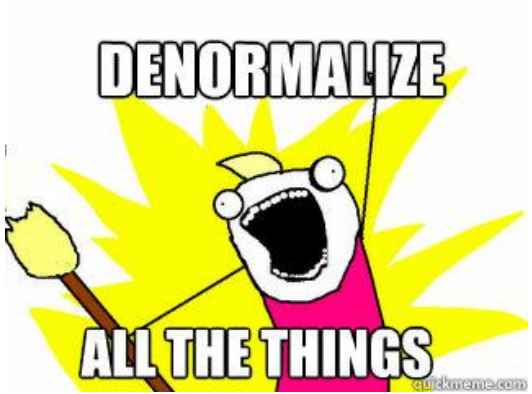
Costs, Storage vs Processing

Processing

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

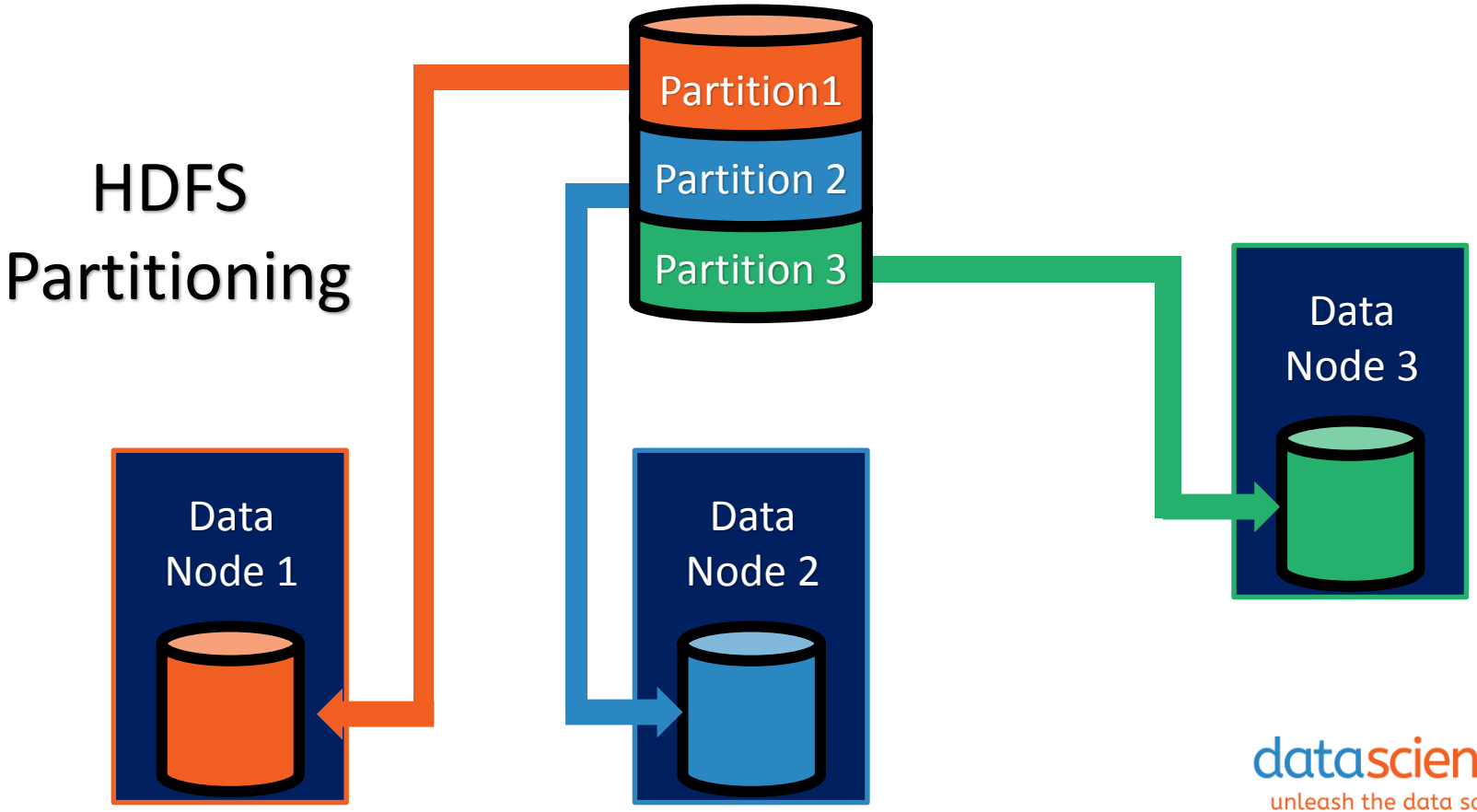
Storage

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

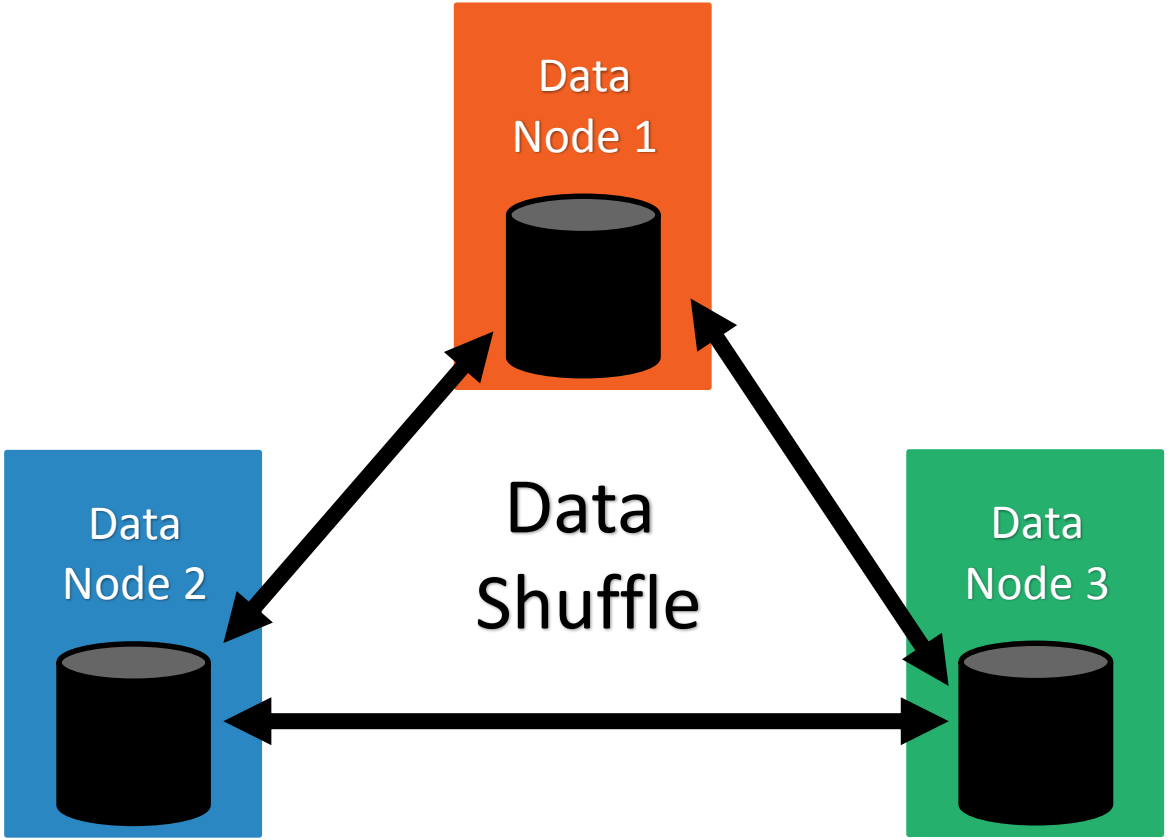


- Distributed Machine Learning
- Installed into Hadoop & Spark
- R-like language Implementation

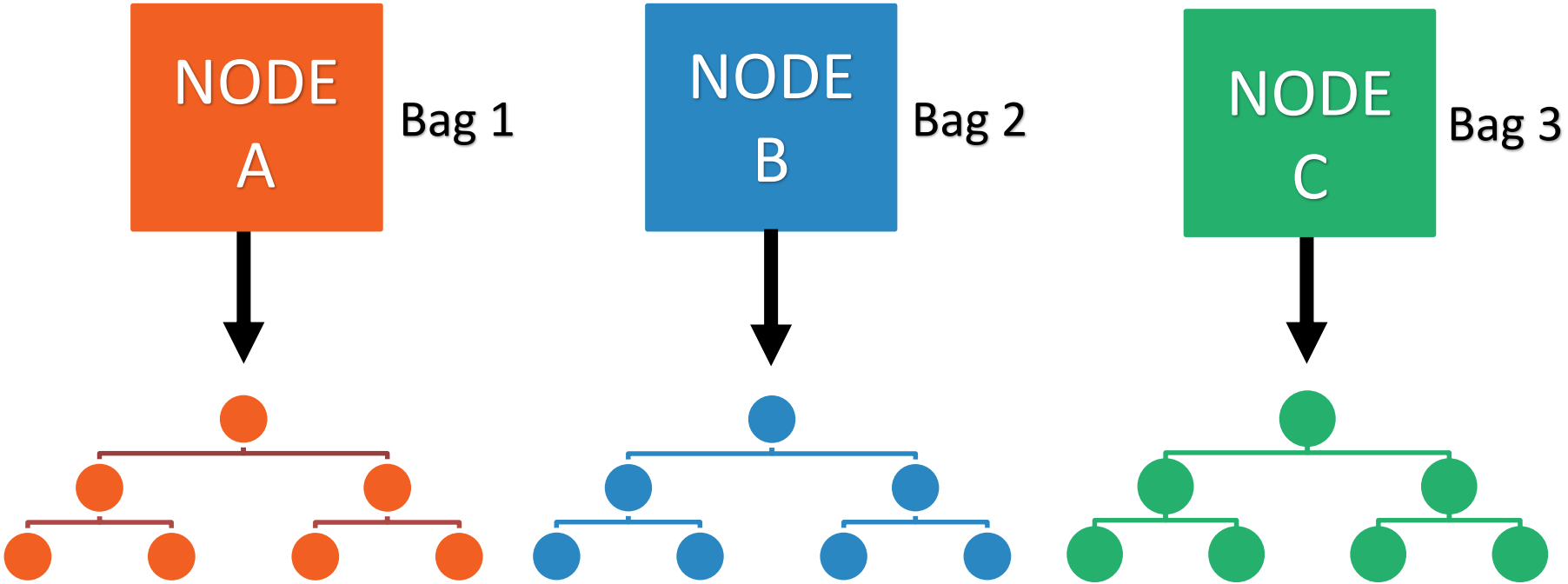
Distributed Random Forest

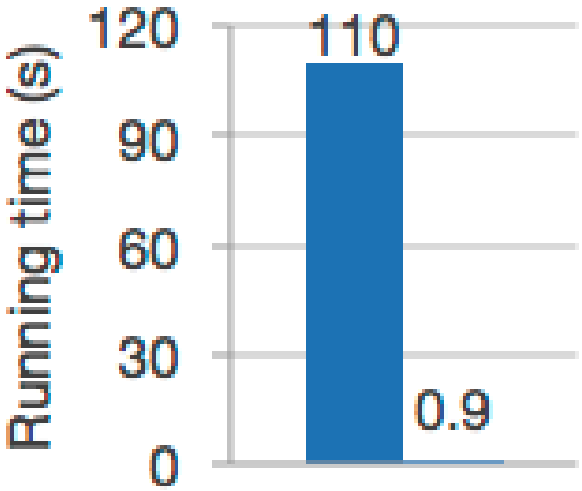


Distributed Random Forest



Distributed Random Forest





In-Memory: 100x times faster than Hadoop

3x faster on 10x few machines

Datona GraySort Benchmark: Sort 100 TB of data

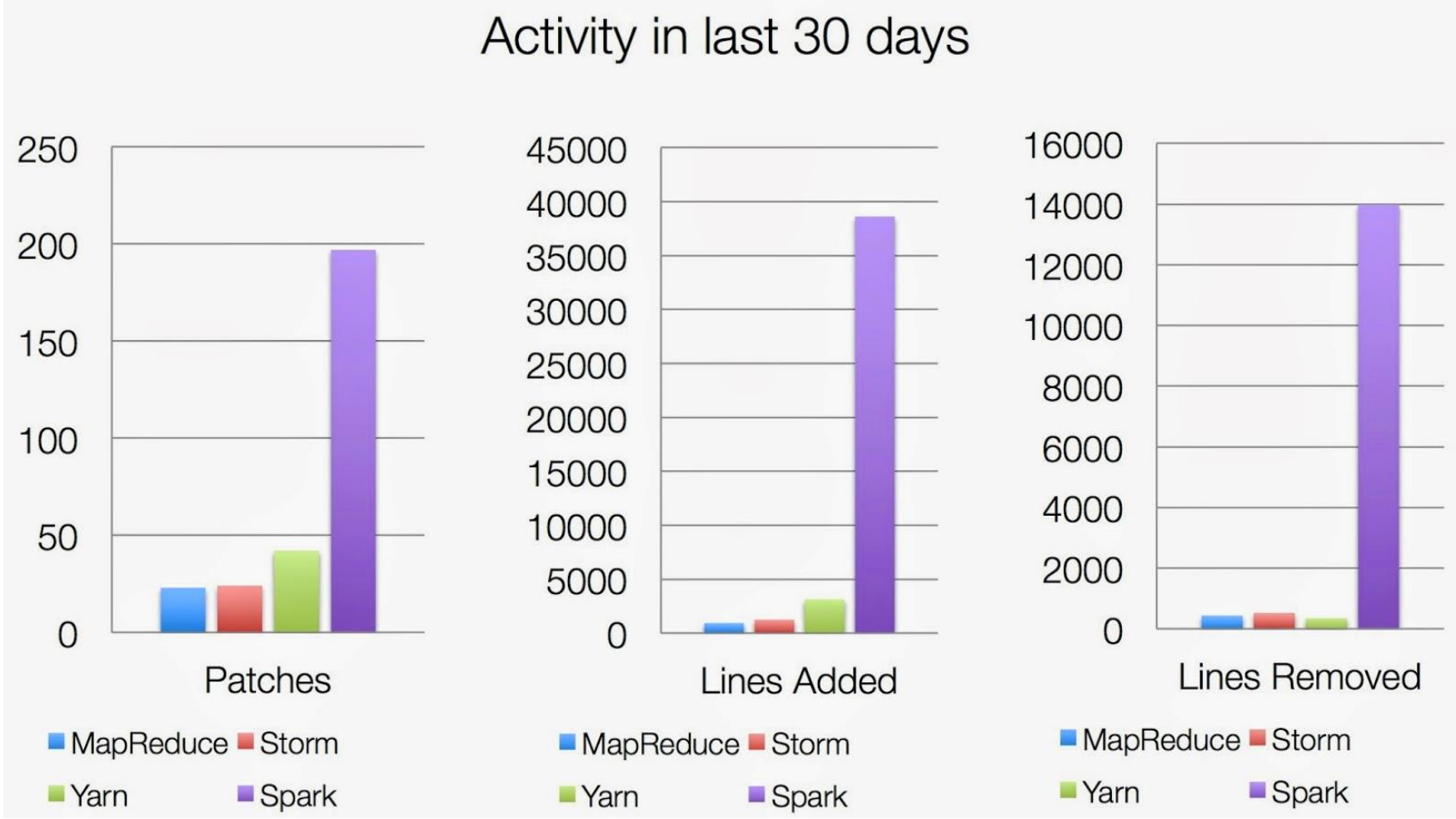
Previous World Record:

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

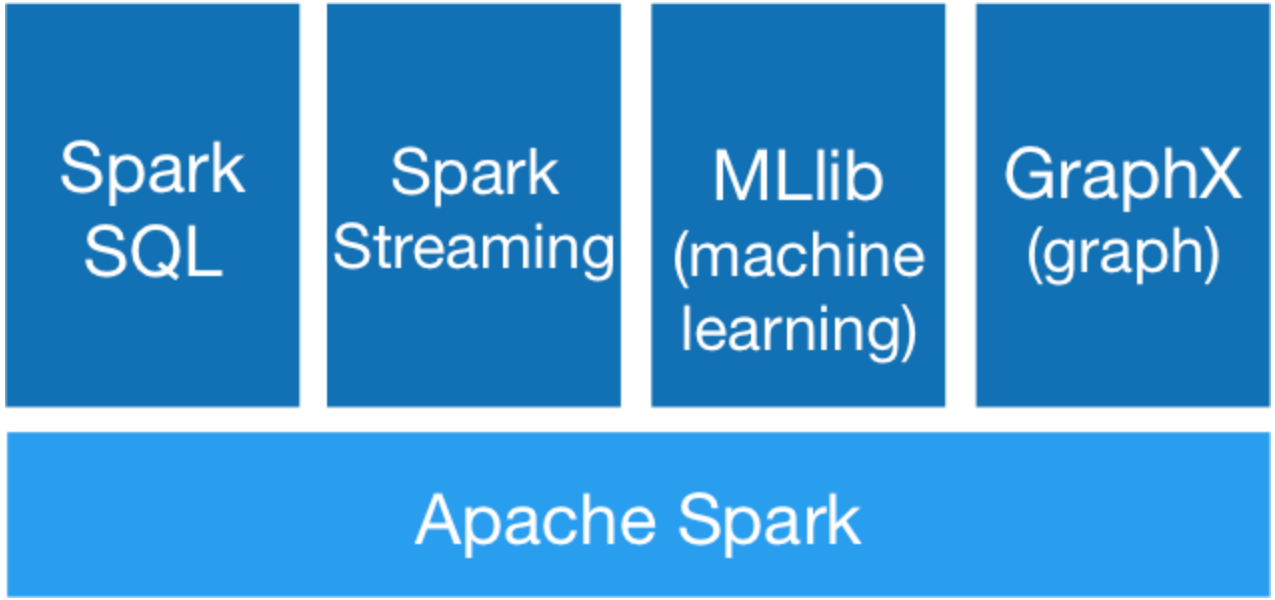
2014:

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

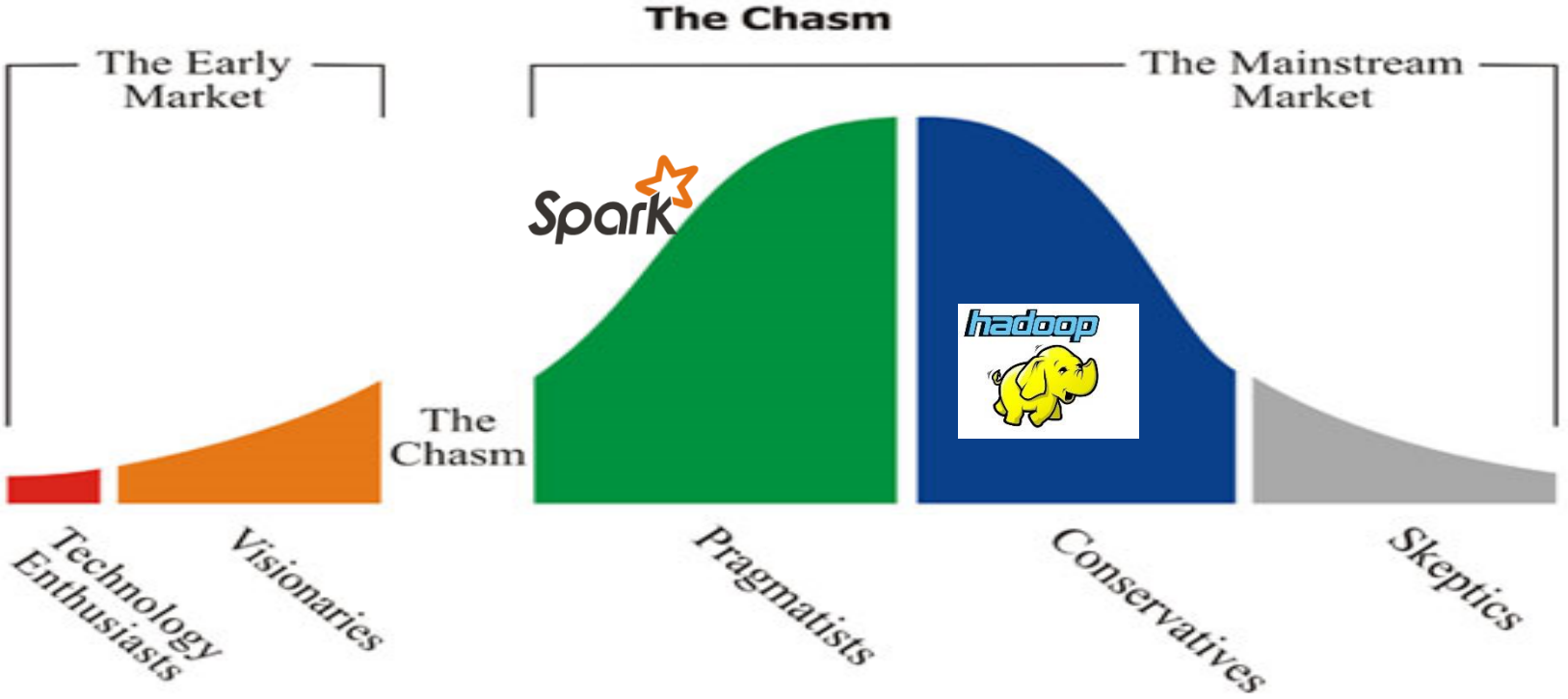
Source: https://databricks.com/blog/2014/10/10/spark-petabyte-sort.html



Source: Xiangrui Meng, Data Bricks



Technology adoption life cycle



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

QUESTIONS