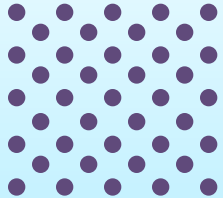
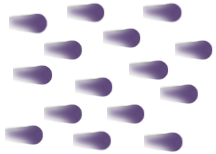
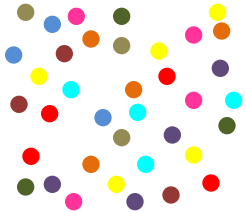
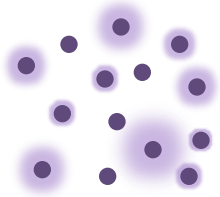
Big Data Engineering with Distributed Systems

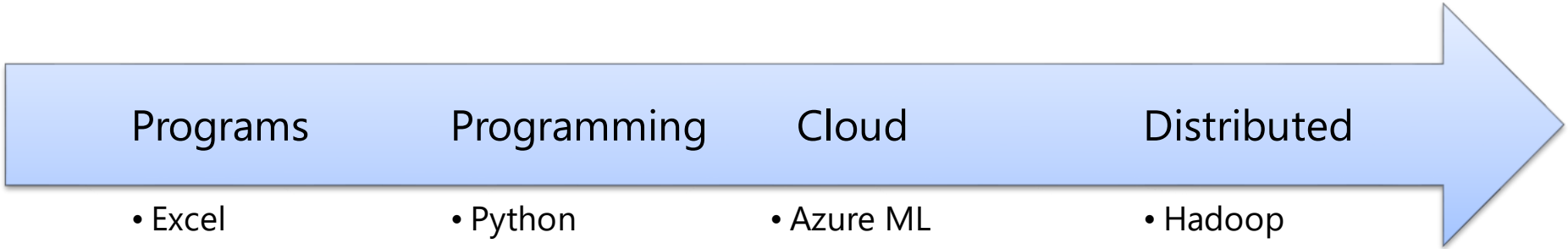
Data Science Dojo

Batch Processing

Volume	Velocity	Variety	Veracity	Value
				
Data at rest Terabytes to exabytes of existing data to process	**Data in motion** Streaming data, milliseconds to seconds to respond	**Data in many forms** Structured, unstructured, text, and multimedia	**Data in doubt** Uncertainty due to data inconsistency and incompleteness, ambiguities, latency, deception, and model approximations	**Data can have different value** Not all bytes are created equal

- Save up all your raw data and process all at once
- Addresses the 'volume' problem of big data

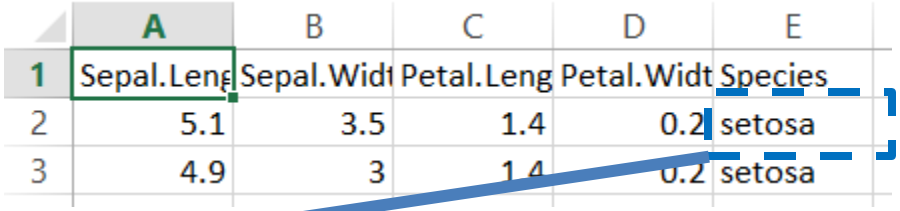
Machine Learning Scaling



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

datasciencedojo
data science for everyone

Excel: Cell Meta Data



	A	B	C	D	E
1	Sepal.Leng	Sepal.Widt	Petal.Leng	Petal.Widt	Species
2	5.1	3.5	1.4	0.2	setosa
3	4.9	3	1.4	0.2	setosa

E2 Cell = Application, Address, AllowEdit, Areas, Borders, BottomPadding, Comment, Column, ColumnIndex, Creator, **Font,** FitText, Height, HeightRule, ID, Interior, LeftPadding, NestingLevel, RightPadding, Row, RowIndex, Shading, Tables, TopPadding, VerticalAlignment, **Value,** Width, WordWrap

"Value": "Setosa"


```
"Font":{
    "Application": "Microsoft Excel",
    "Background": None,
    "Bold": True,
    "Color": 0,
    "ColorIndex": 5,
    "Creator": "XCEL",
    "FontStyle": "Bold Italic",
    "Italic": True,
    "Name": "Comic Sans MS",
    "OutlineFont": True,
    "Parent": None,
    "Shadow": False,
    "Size": 12,
    "Strikethrough": False,
    "Subscript": False,
    "ThemeColor": 12,
    "ThemeFont:": 2,
    "TintAndShade": 1,
    "Superscript": False,
    "Underline": False,
}
```

R Limits

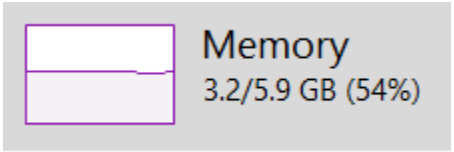
- Single core
- Single threaded
- All in memory (RAM)
- Vectors & Matrices capped at 4,294,967,295 elements (rows) if 32-bit version; 2^32 - 1

R Limits: RAM

- All in memory (RAM)

$$Max\ Data\ Limit = (\ Total\ RAM\ Access\ x\ 80\%) - Normal\ RAM\ Usage$$

Laptop Example:



$$Max\ Data\ Limit = (\ 5.9\ GB\ x\ 80\%)\ - 3.2\ \text{GB}$$
$$Max\ Data\ Limit = \sim 1.52 GB$$

*R data frames actually bloats data files by 3x

R's Data Limit = $1.52 GB\ \div 3 = 518.8\ MB$

R Limits: RAM

INSTANCE	CORES	RAM	DISK SIZES	PRICE
G1	2	28 GB	384 GB	$0.67/hr (~$498/mo)
G2	4	56 GB	768 GB	$1.34/hr (~$997/mo)
G3	8	112 GB	1,536 GB	$2.68/hr (~$1,994/mo)
G4	16	224 GB	3,072 GB	$5.36/hr (~$3,988/mo)
G5	32	448 GB	6,144 GB	$9.65/hr (~$7,180/mo)

Azure's Biggest Virtual Machine

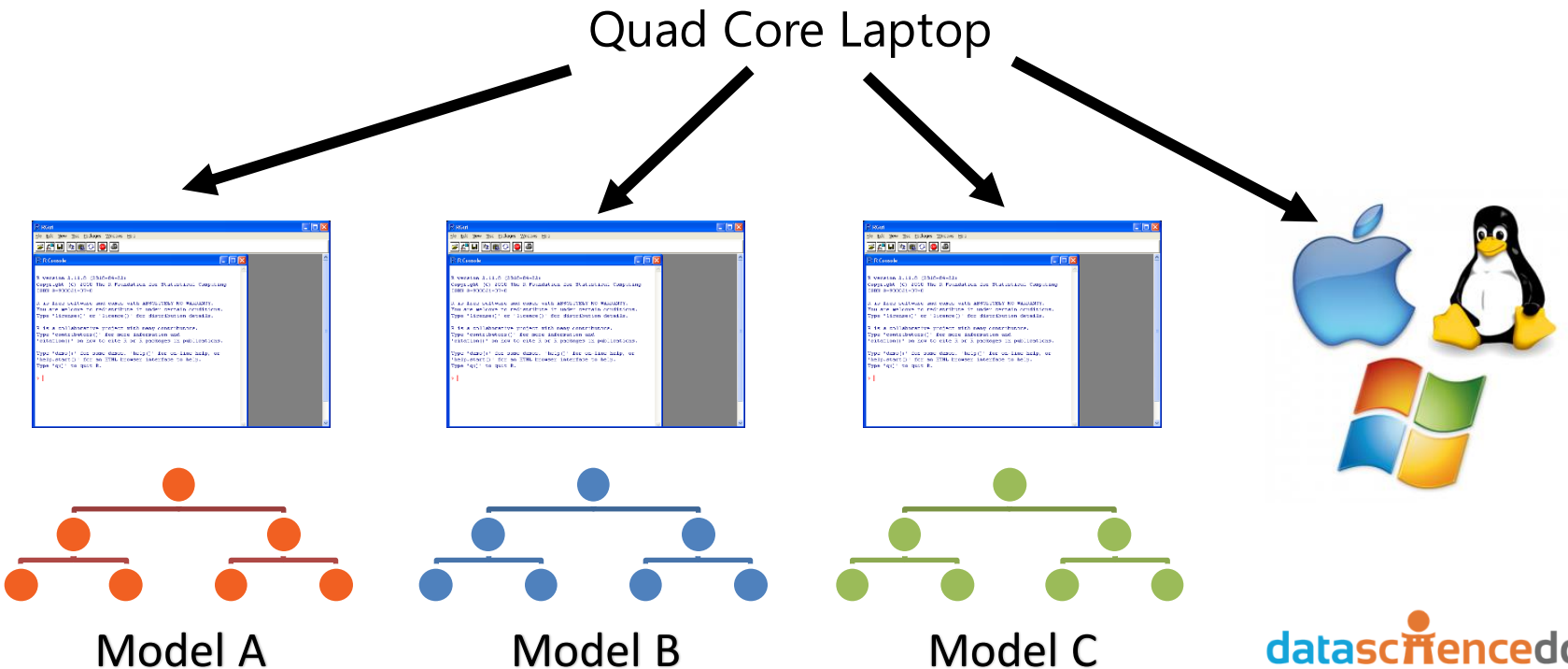
$$Max\ Data\ Limit = (\ 448gb\ x\ 80\%\ ) - 1GB$$

$$Max\ Data\ Limit = \sim \mathbf{357.4gb}$$

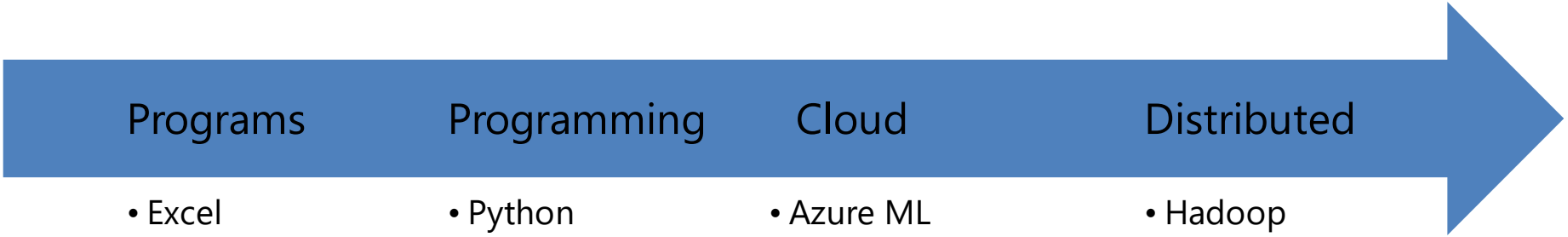
datasciencedojo
data science for everyone

R Limits: Single Core

- Single core
- Single threaded



Machine Learning Scaling



- Excel

- Python
- R
- SAS

- Azure ML
- AWS ML
- Watson Analytics
- Big ML
- Cloud Virtual Machines

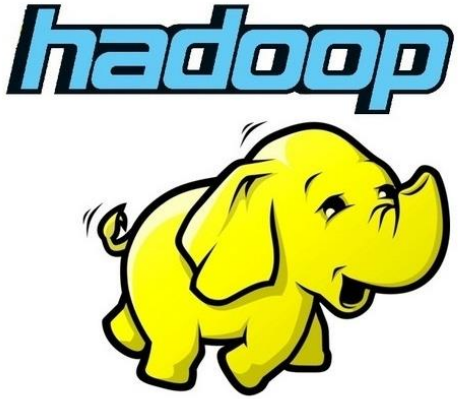
- Hadoop
- Spark
- H20
- Revolution R

Distributed R Solutions:

https://cran.r-project.org/web/views/HighPerformanceComputing.html

datasciencedojo
data science for everyone

Agenda



From a Data Scientist's Perspective

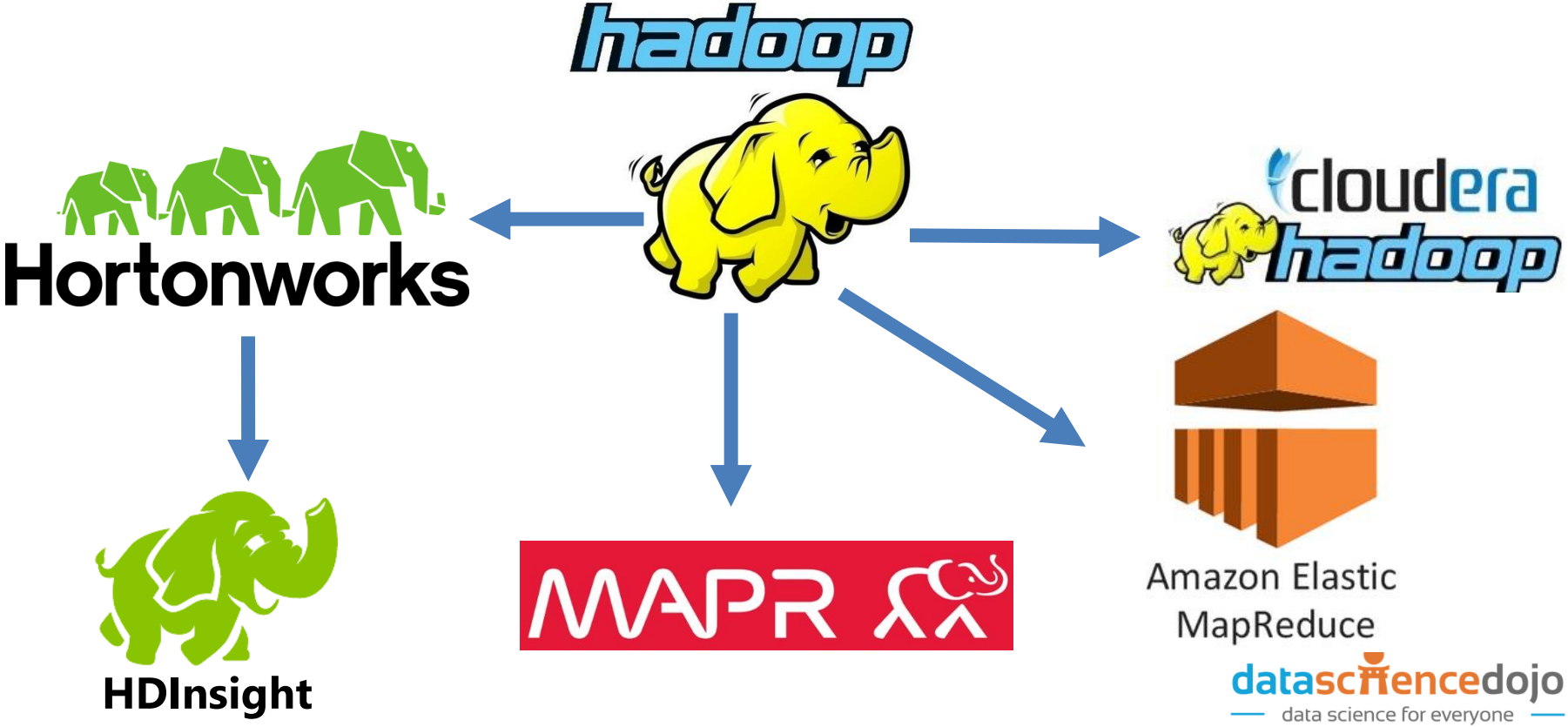
Goals:

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

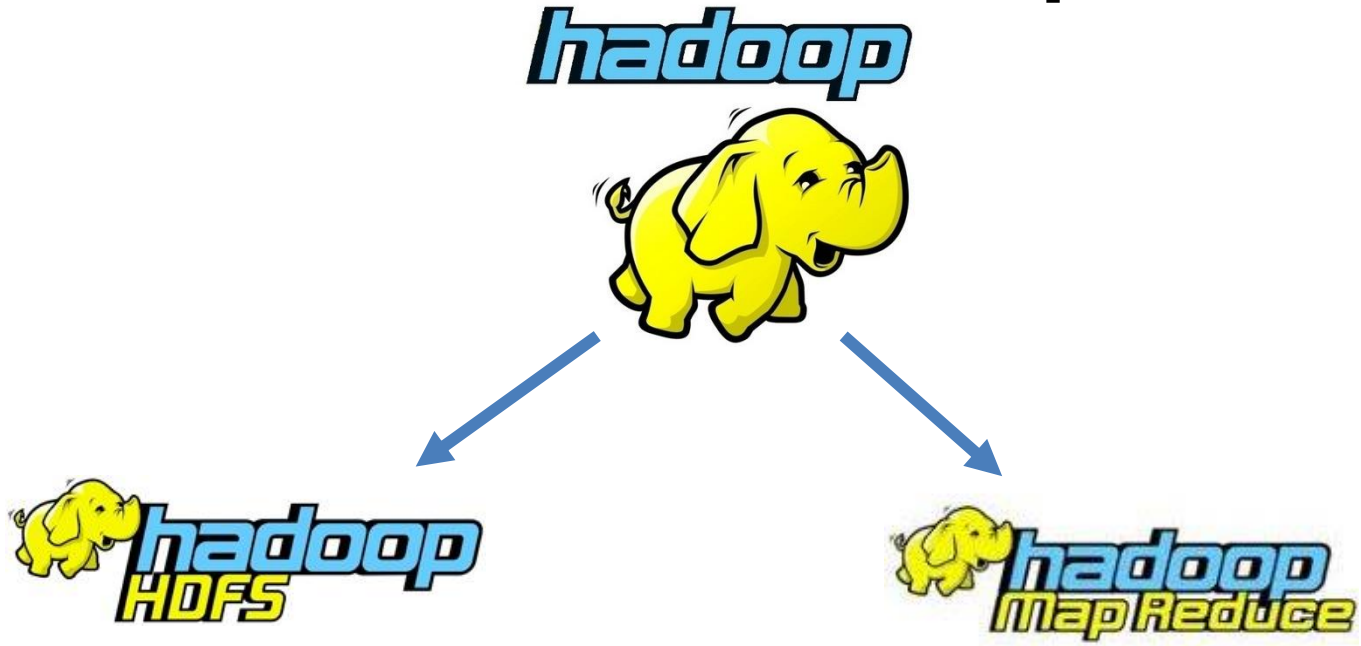
Not goals:

- Managing or administering a Hadoop cluster

Hadoop Implementations



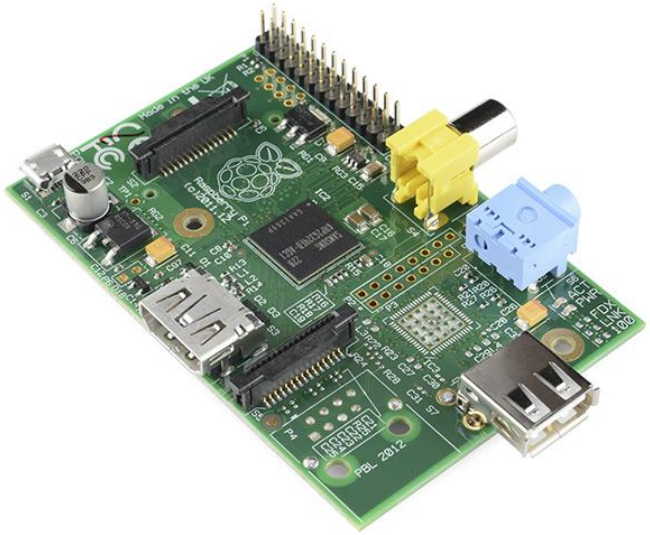
(Vanilla/Base) Hadoop

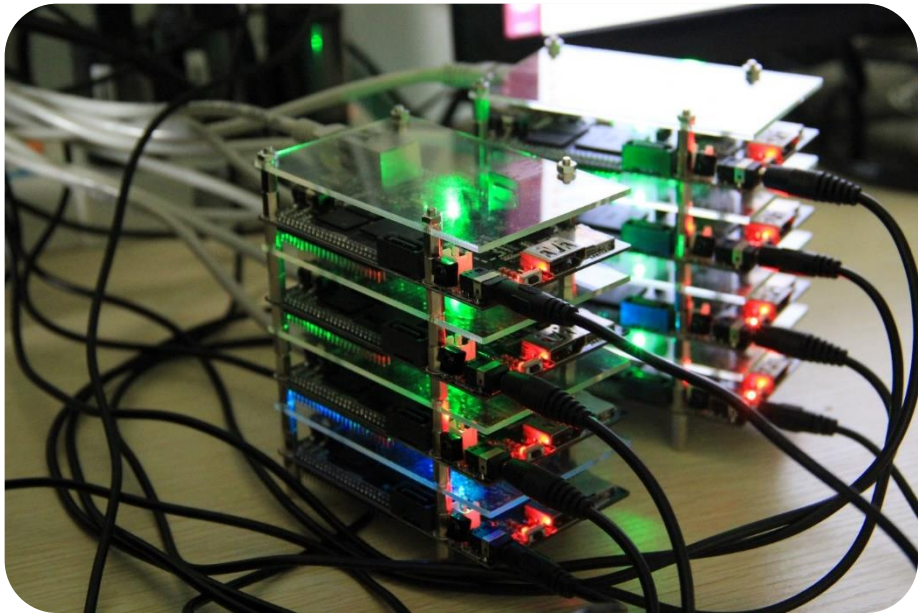


Processing engine for distributed batch processing.

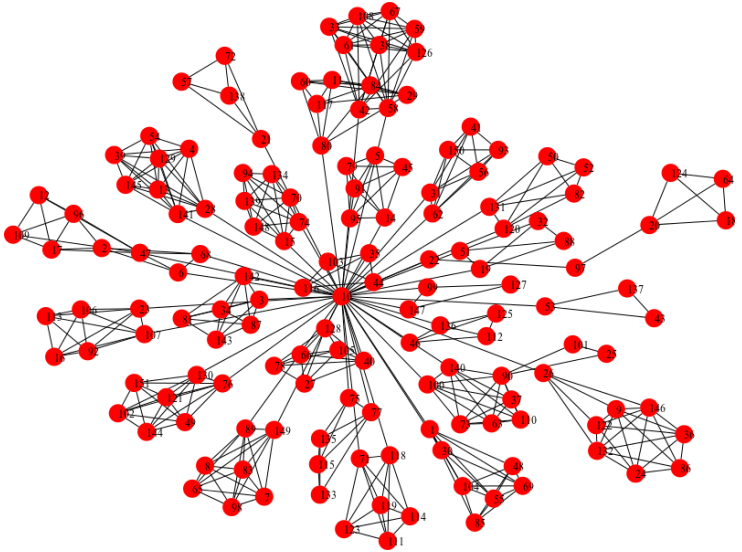
Turn Back The Clock, The Mainframe

Distributed Computing

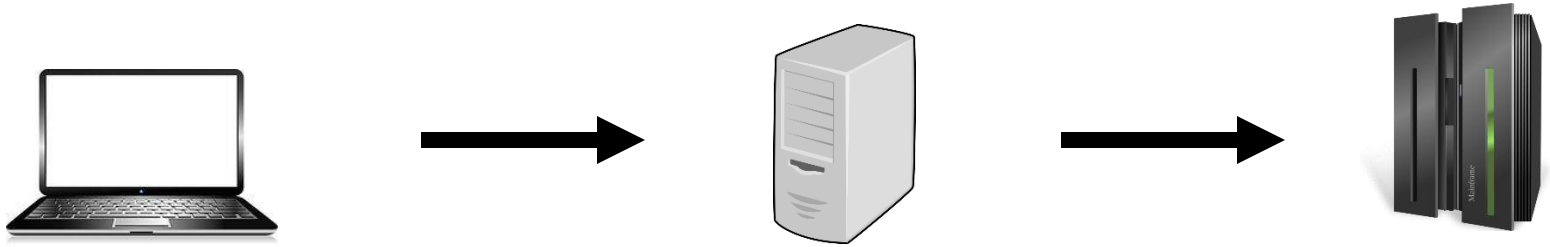




Cloud Computing



Scaling Computational Power



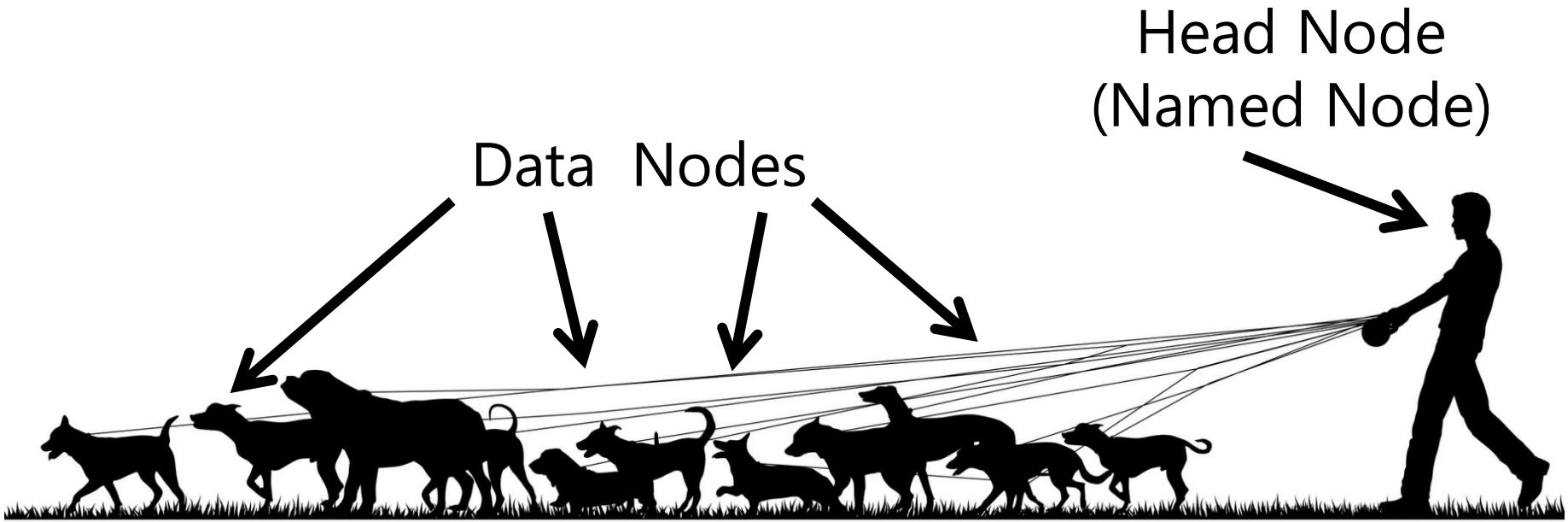
Old Scaling:

- Vertical Scaling, Scaling UP
- High performance computers

New Scaling:

- Horizontal Scaling, Scaling OUT
- Commodity hardware, distributed

If dogs were servers...



HDFS & MapReduce

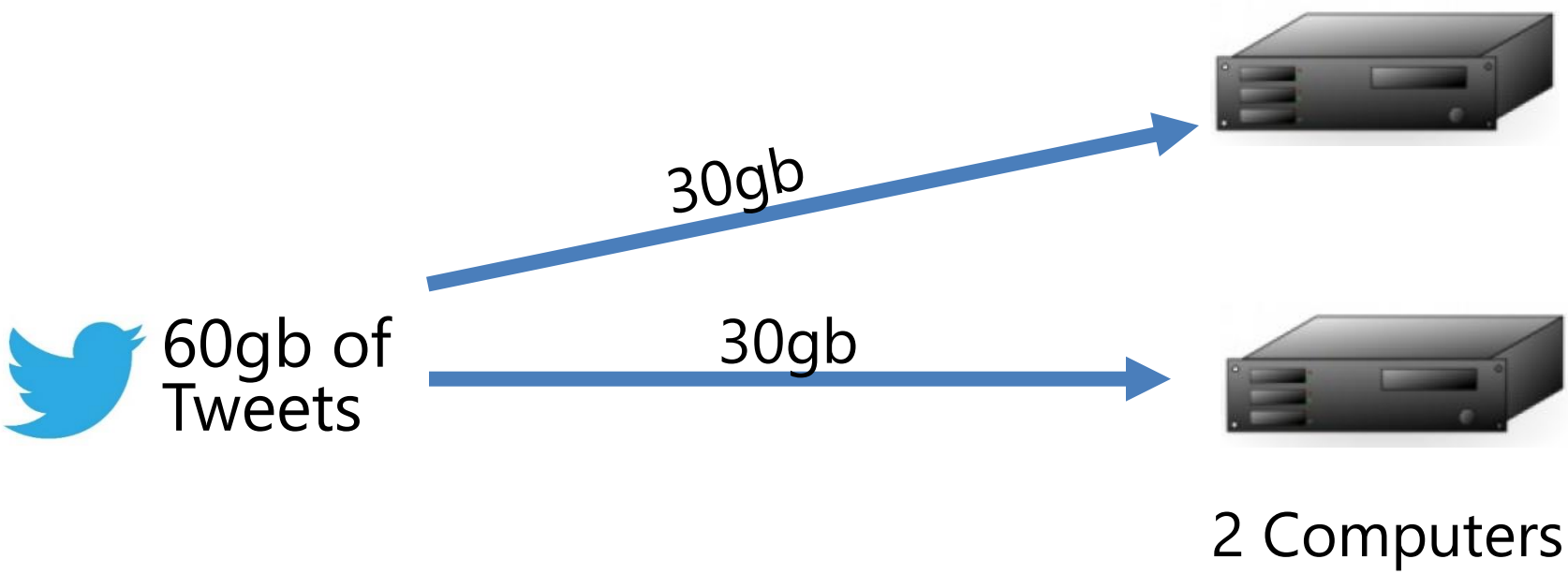
60gb of Tweets

60gb

1 Computer

Processing: 30 hours

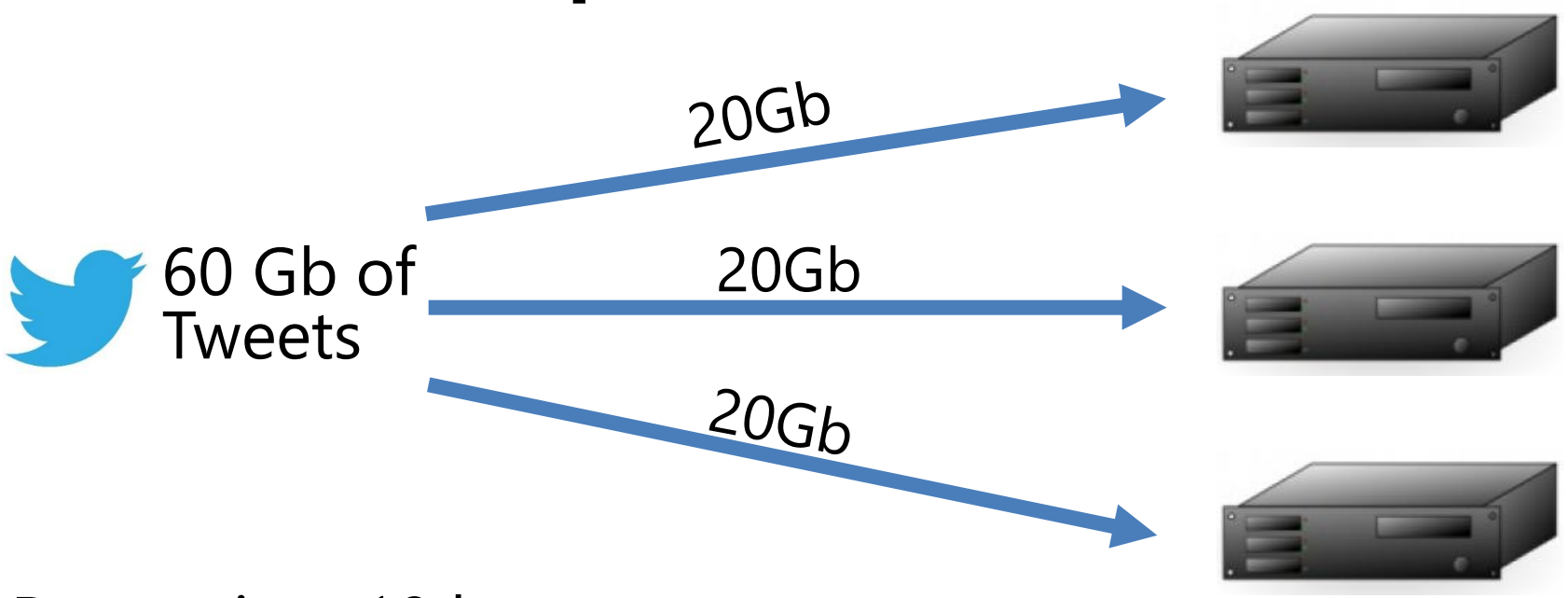
HDFS & MapReduce



Processing: 15 hours

datasciencedojo
data science for everyone

HDFS & MapReduce



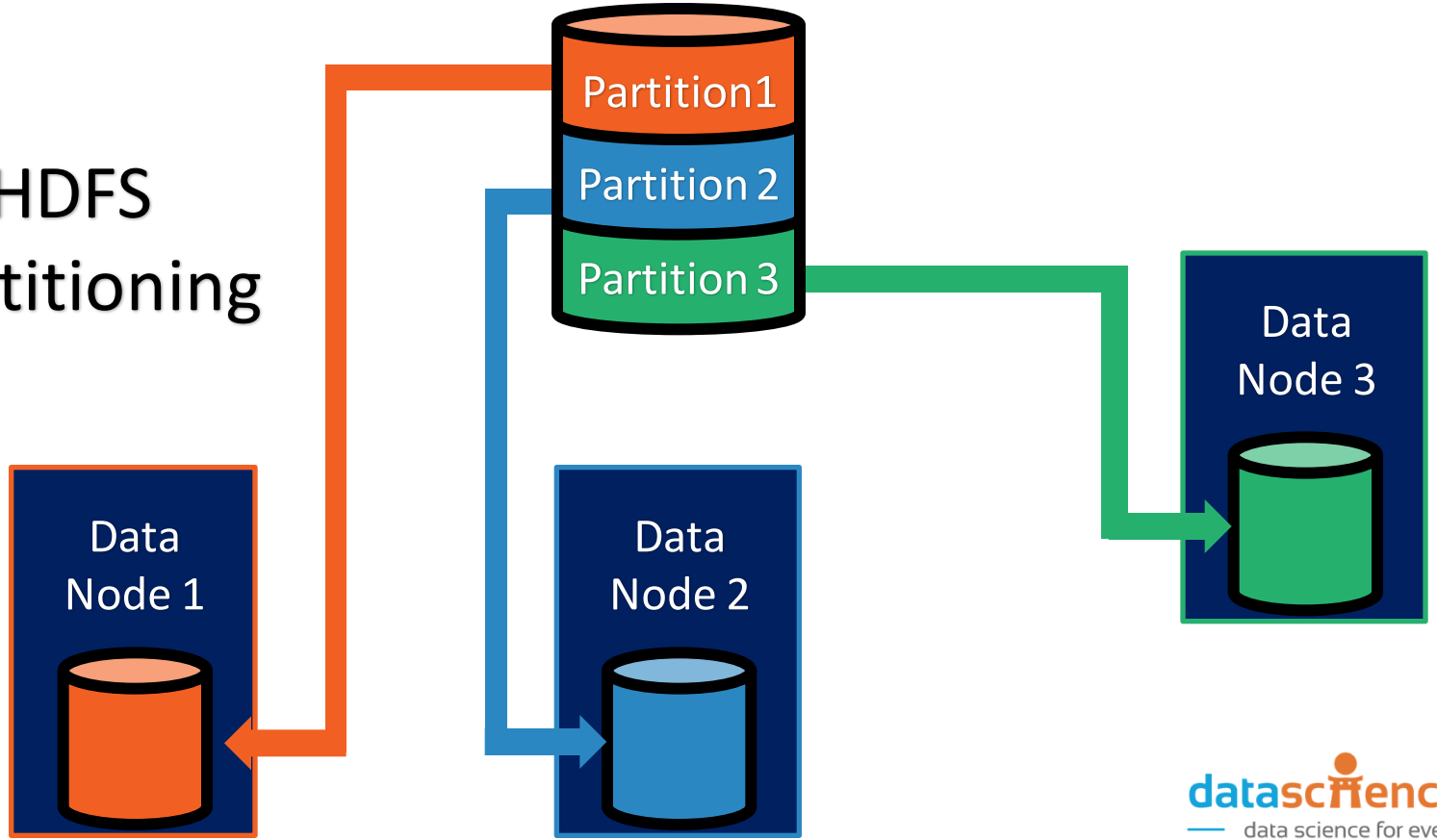
Processing: 10 hours

Most Cases, Linear Scaling Of Processing Power

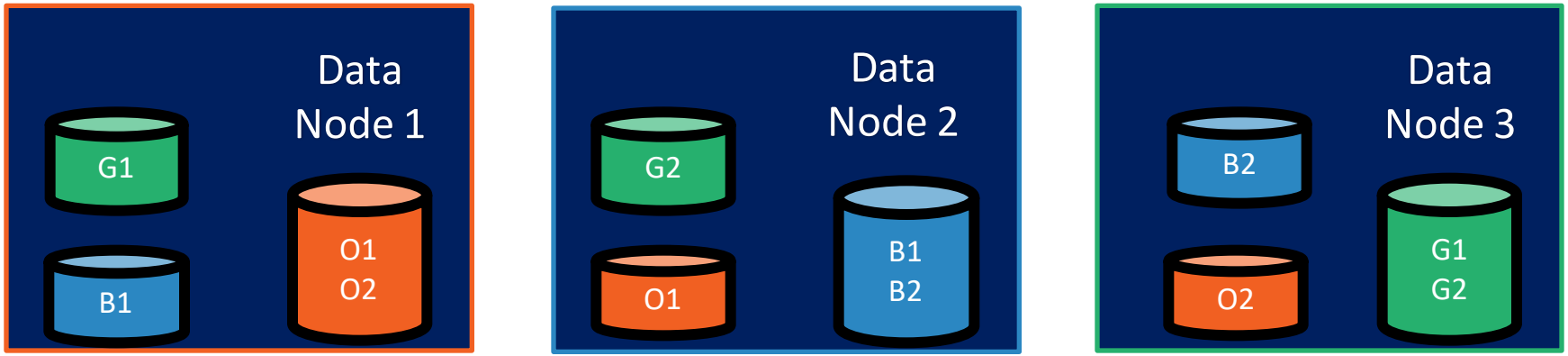
Number of Computers	Processing Time (hours)
1	30
2	15
3	10
4	7.5
5	6
6	5
7	4.26
8	3.75
9	3.33

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

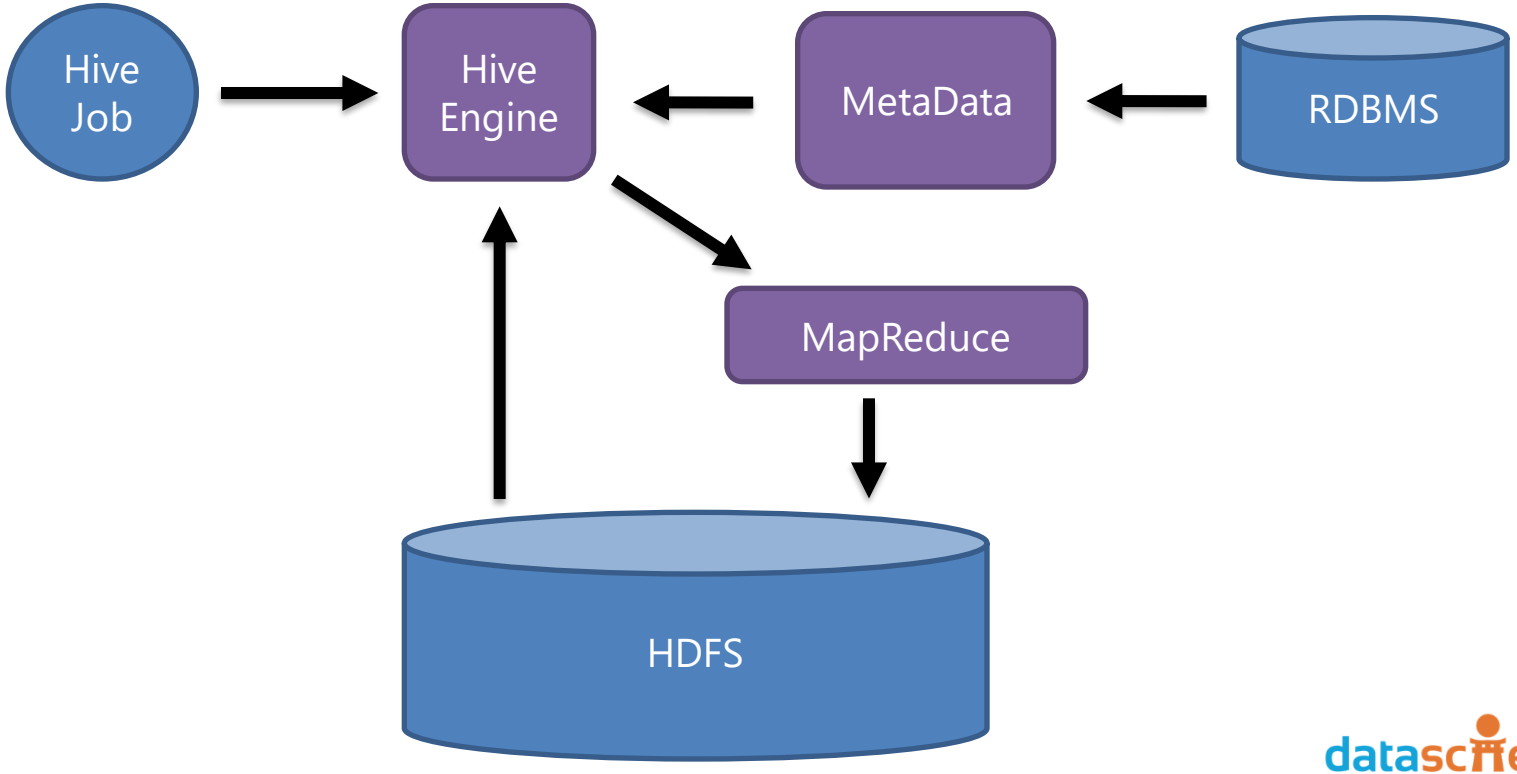
Sqoop: Data import and export

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

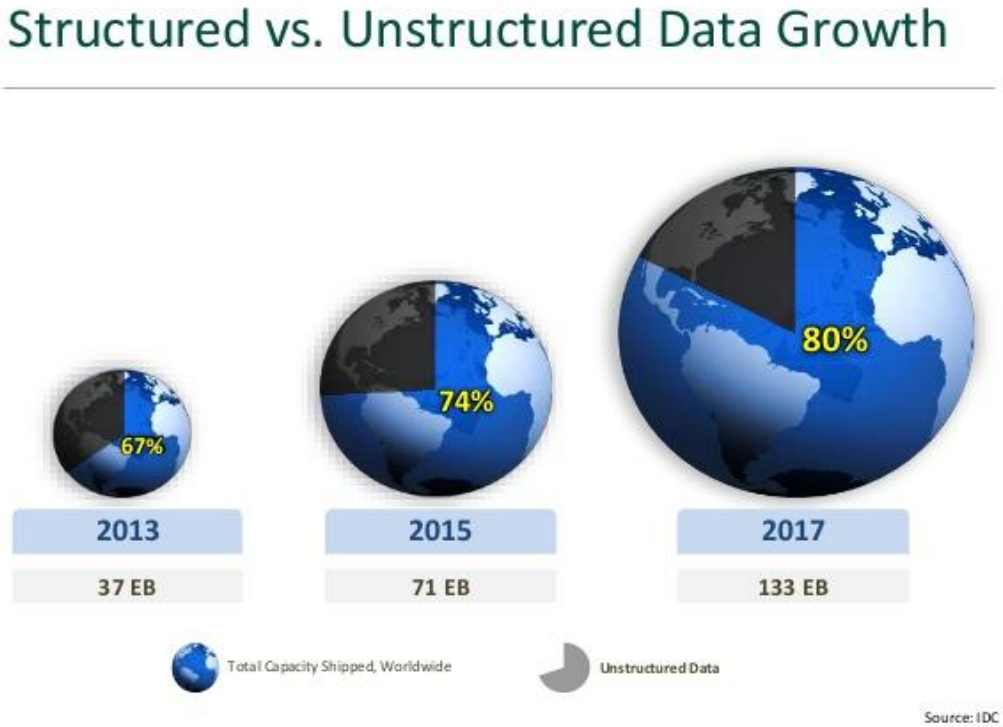
Self Describing Flat Files

- XML
- JSON
- CSV
- TSV

Why Hive?

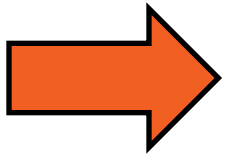
- SQL spoken here (HiveQL)
- ODBC driver
- BI Integration
- Supports only Structured Data

Limitations



datasciencedojo
data science for everyone

Azure Blob Storage



HDInsight

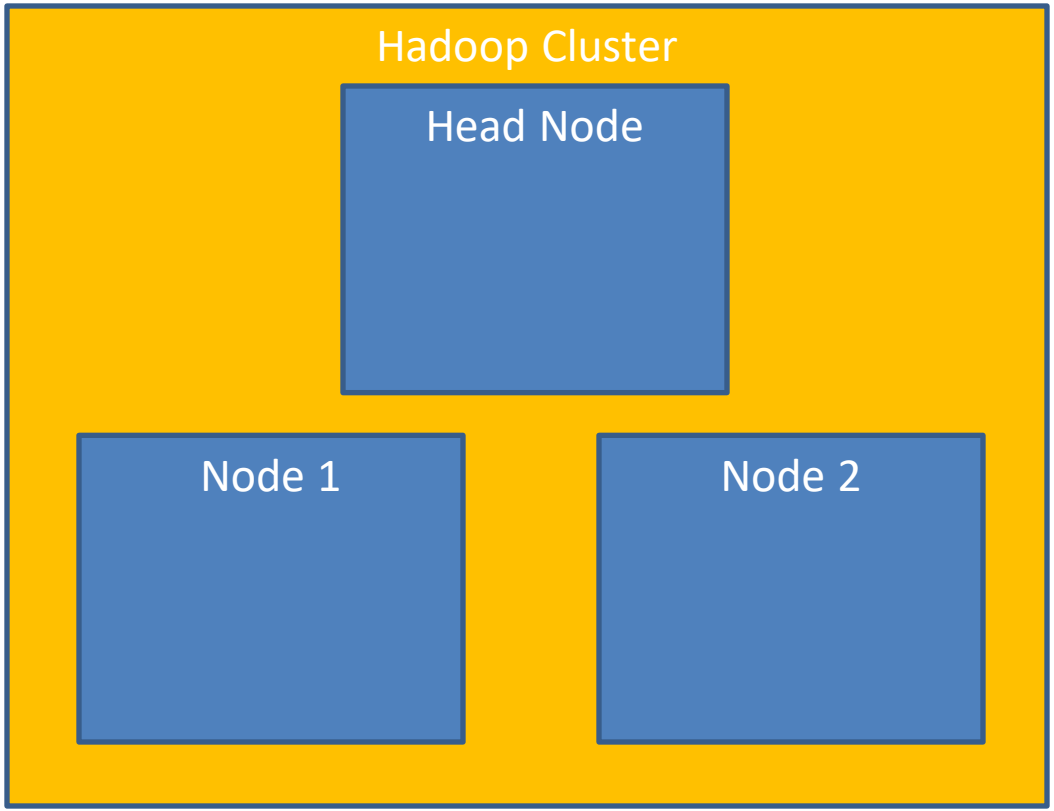
Blob Storage

MapReduce, via Playing Cards

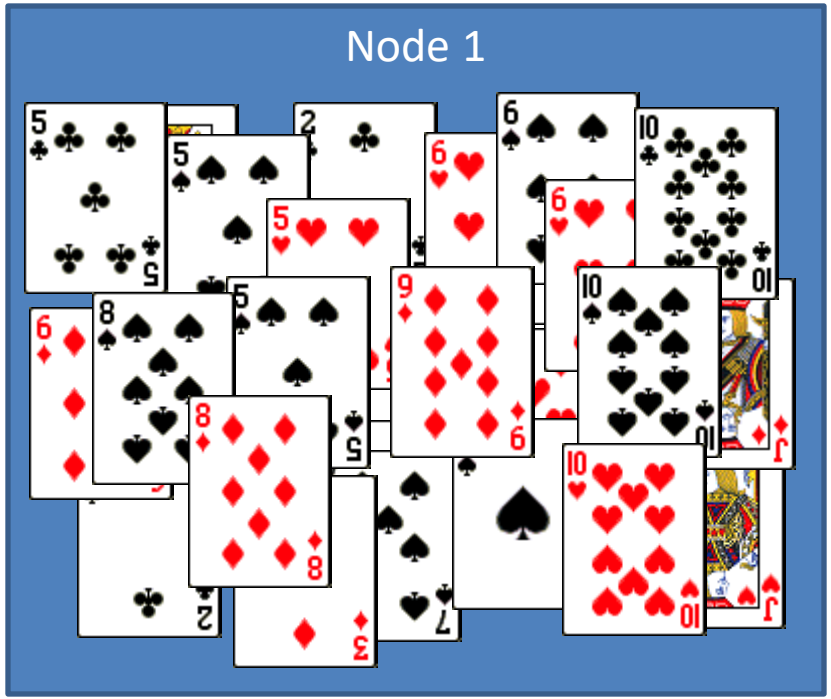
Let's count the number of spades, clubs, hearts, and diamonds in a stack of cards, the way map reduce would.

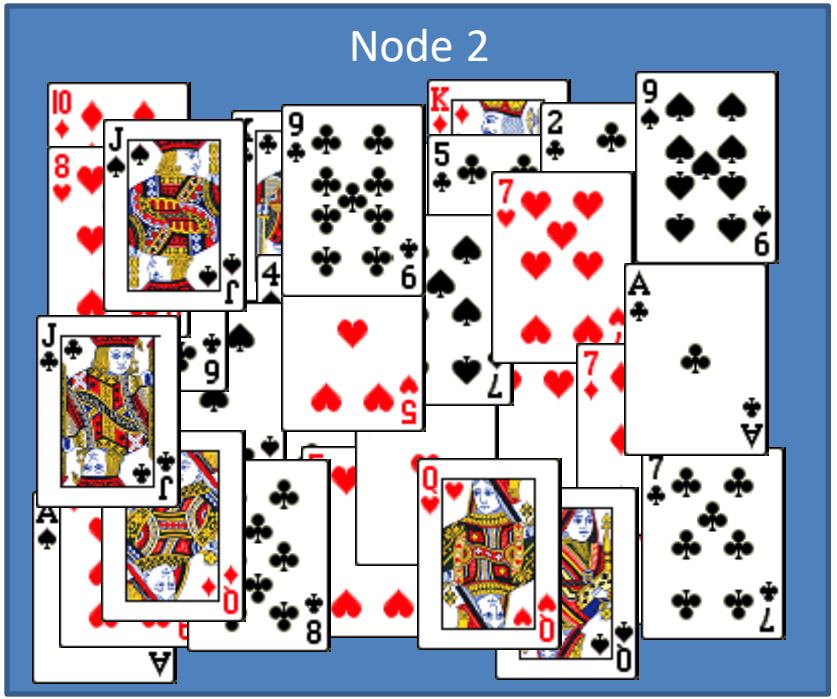
- Each card represents a row of data
- Each suit & number represents an attribute of the data

Using a 2 Data Node Cluster

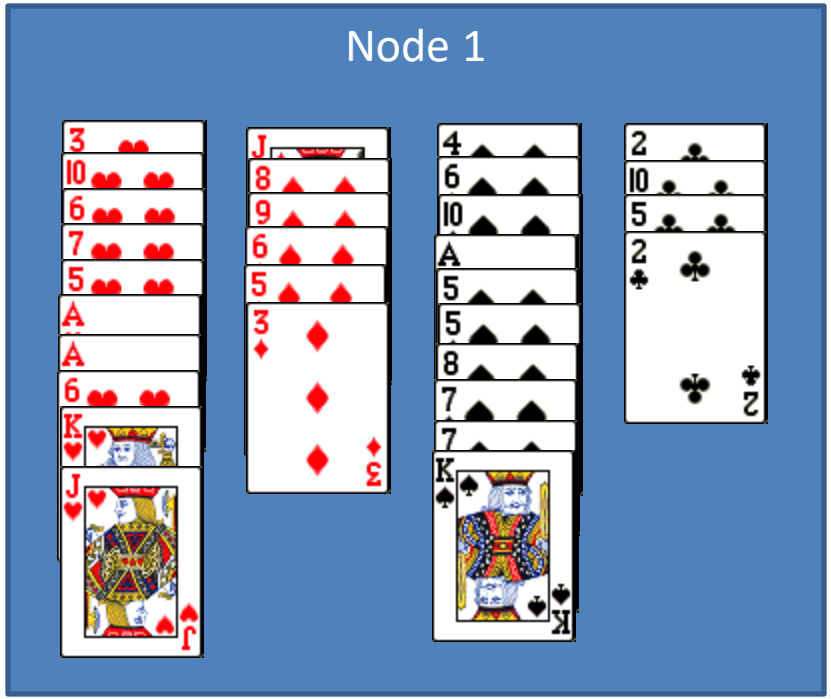


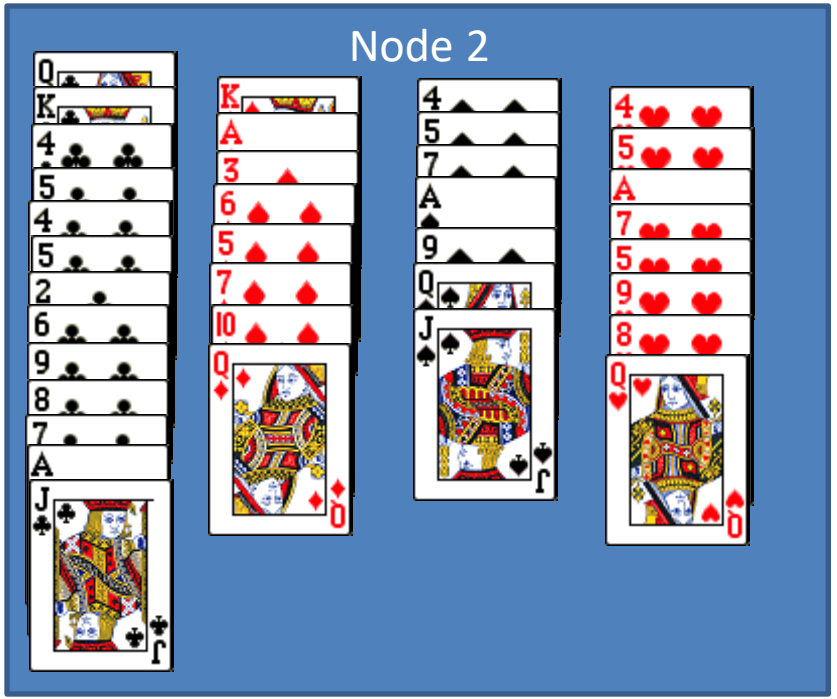
Mapping: Each Node's HDFS



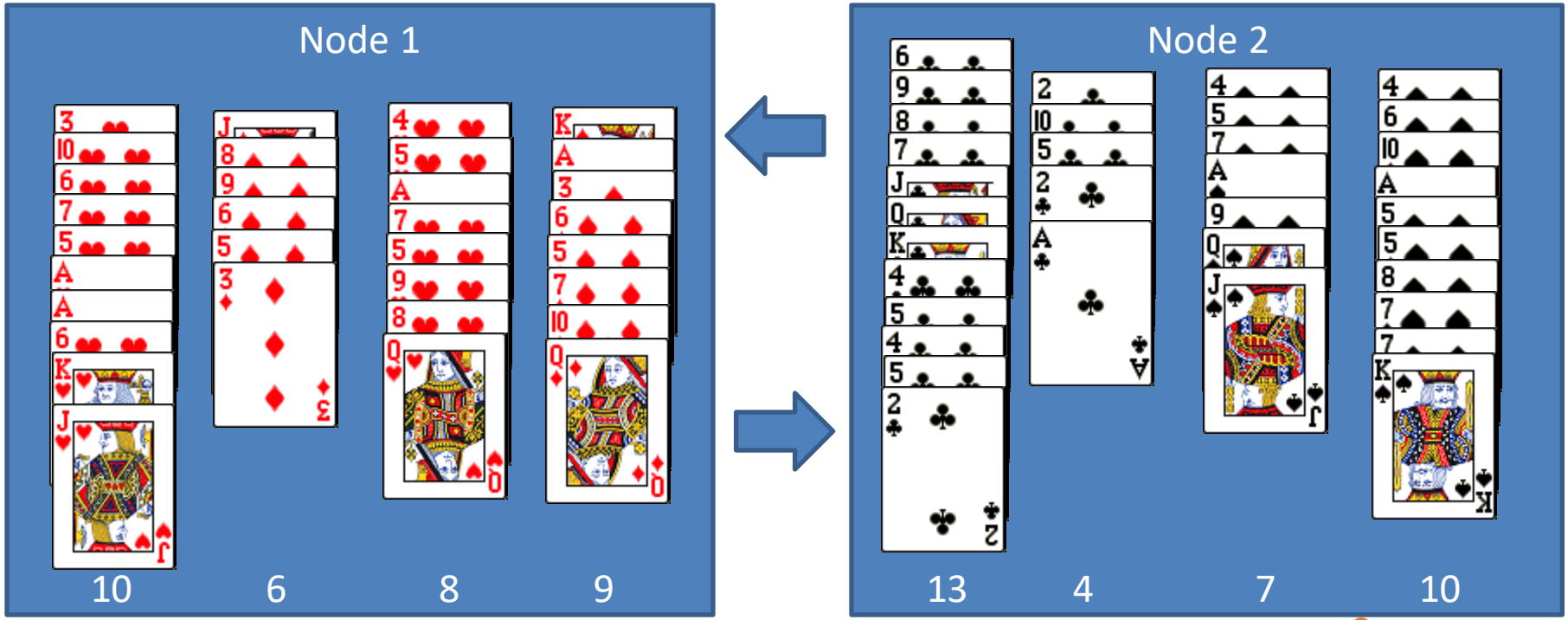


Mapping: Node Sorting



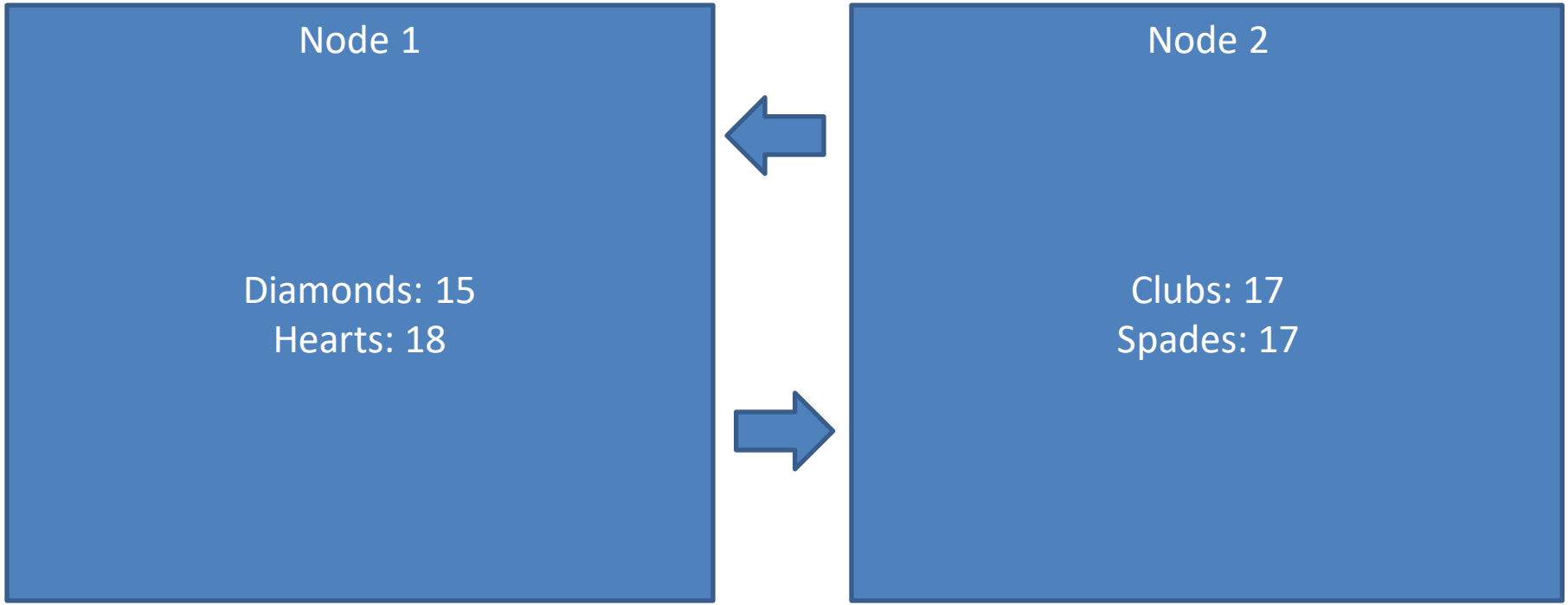


Mapping: Node Shuffle, Data Transfer



datasciencedojo
data science for everyone

Mapping: Node Shuffle, Data Transfer



Execution Engine: Tez

The Stinger Initiative

2011, the world got together and declared MapReduce to be terrible.

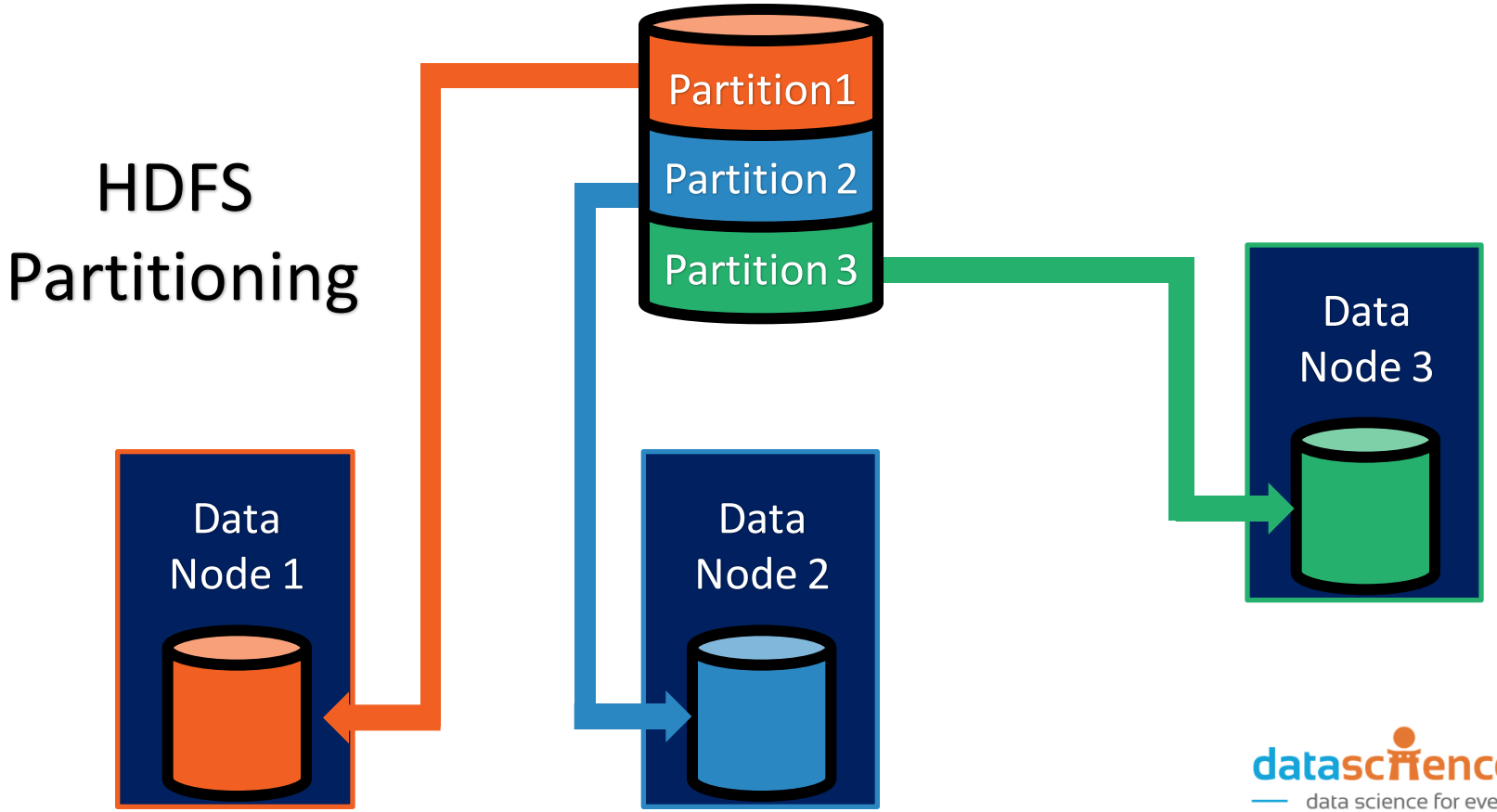
- 44 companies
- 145 developers
- 392k lines of Java code

Hadoop 2.0 with Yarn & Tez

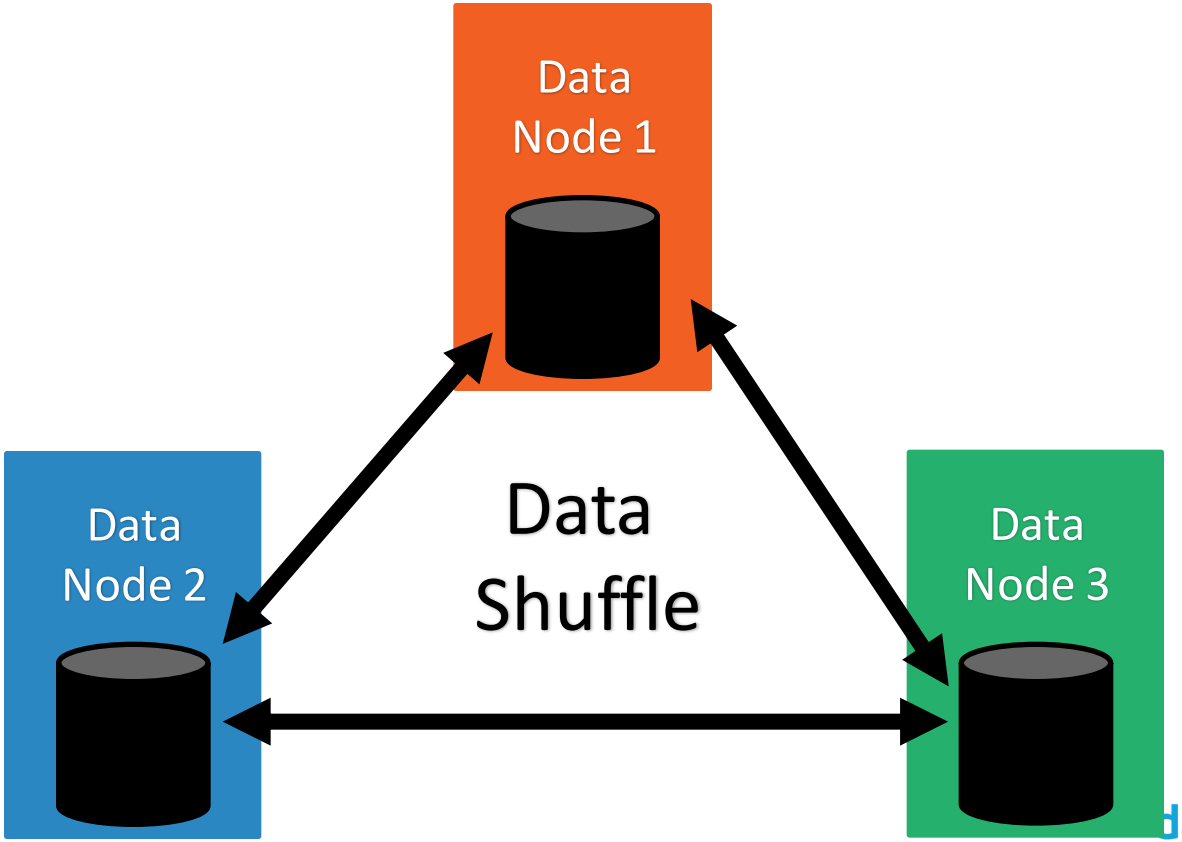
- Tez dropped hive query times by **90%, 100x performance**
- Utilizes Apache Yarn
  - Yarn: resource manager for multi-cluster computing
- Introduced partial in-memory, local head nodes
- Rewrote HiveQL as an actual language, instead of translation

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

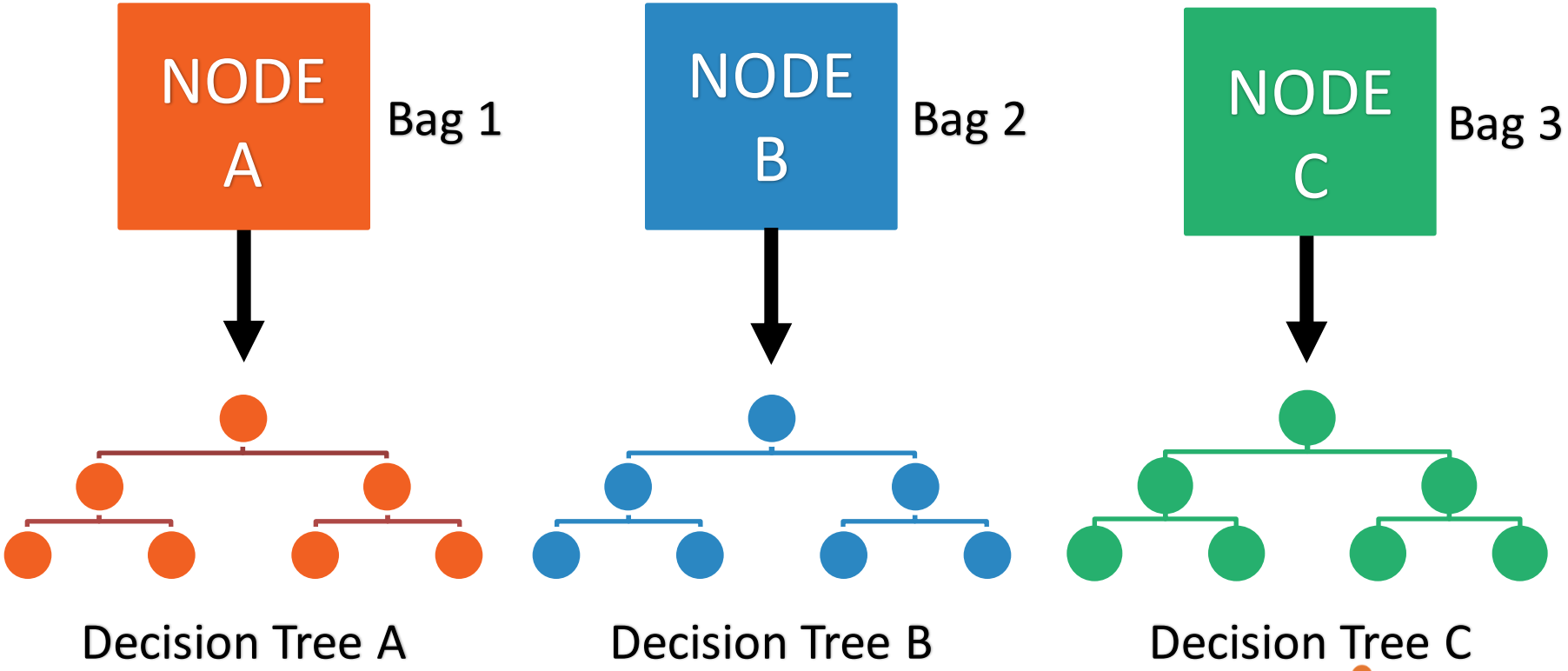
Distributed Random Forest



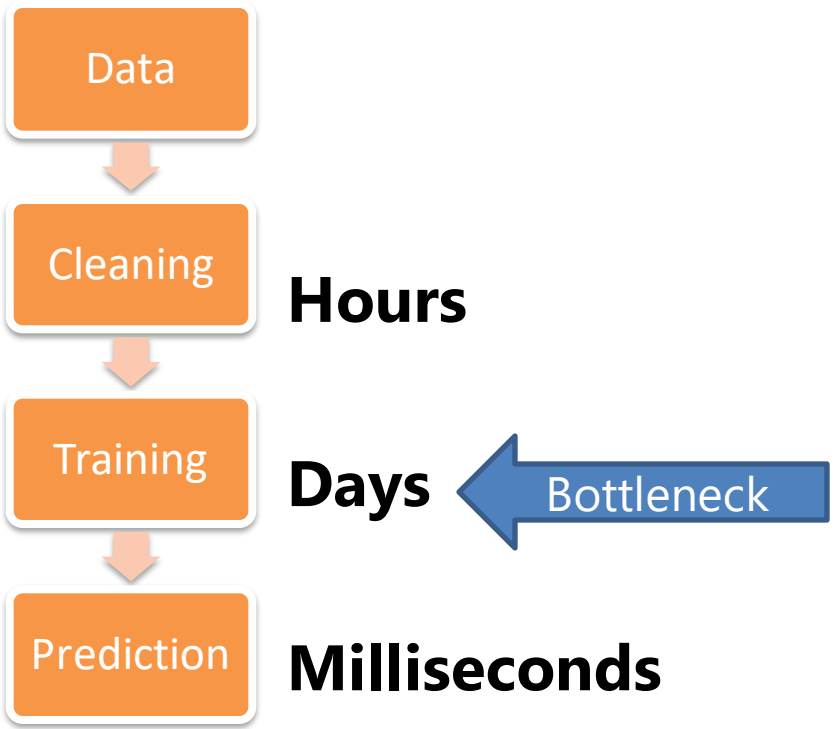
Distributed Random Forest



Distributed Random Forest

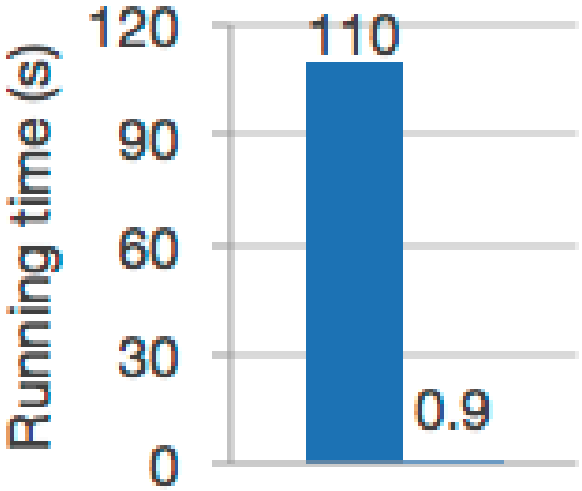


Processing Times - Machine Learning



- Large scale systems are only needed for training
- Phones can use models outputted by mahout to predict new data
- After a model is trained, save the model to any IO file type and reload it where you want

datasciencedojo
data science for everyone



In-Memory: 100x times faster than Hadoop

3x faster on 10x few machines

Datona GraySort Benchmark: Sort 100 TB of data

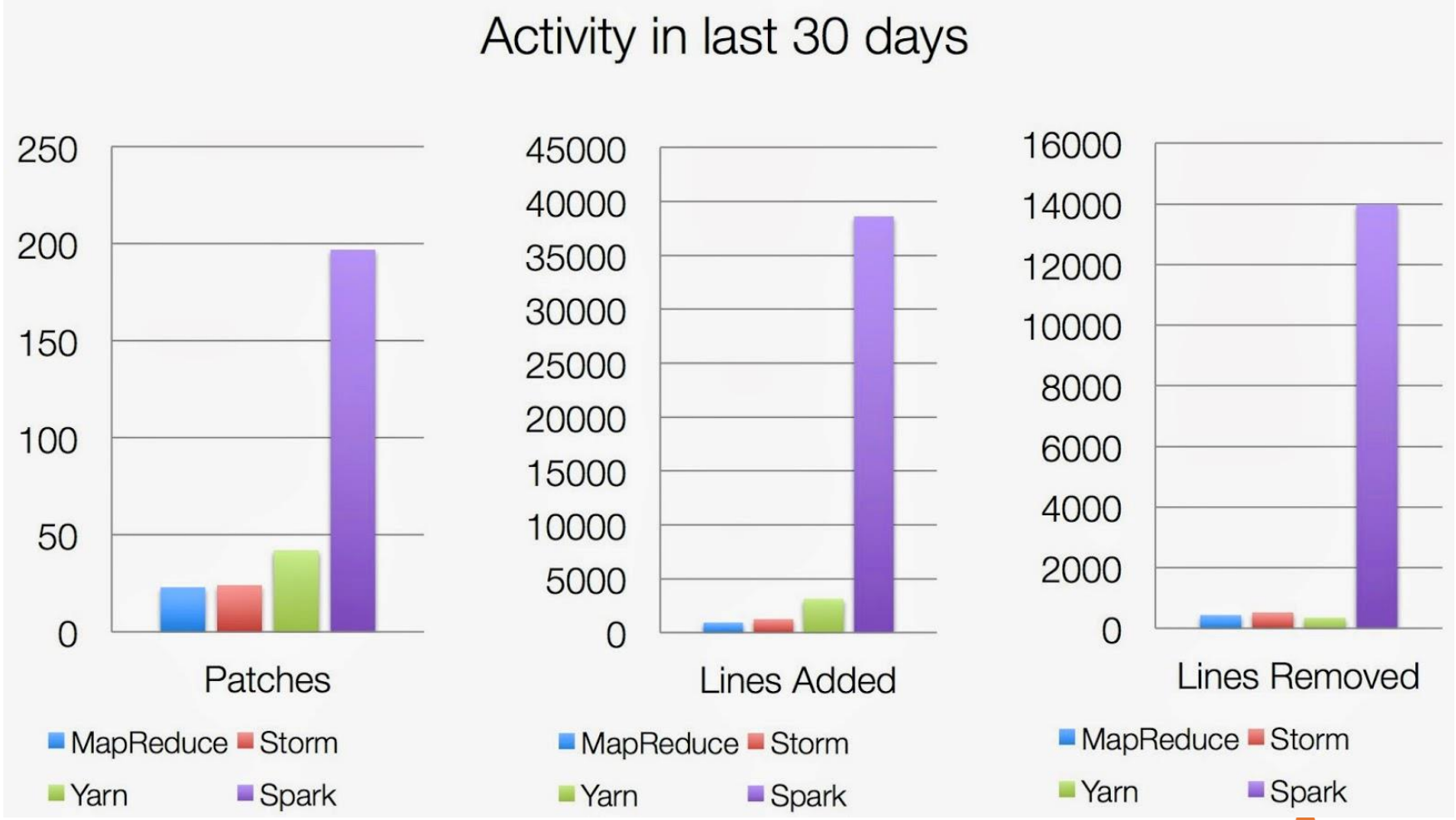
Previous World Record:

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

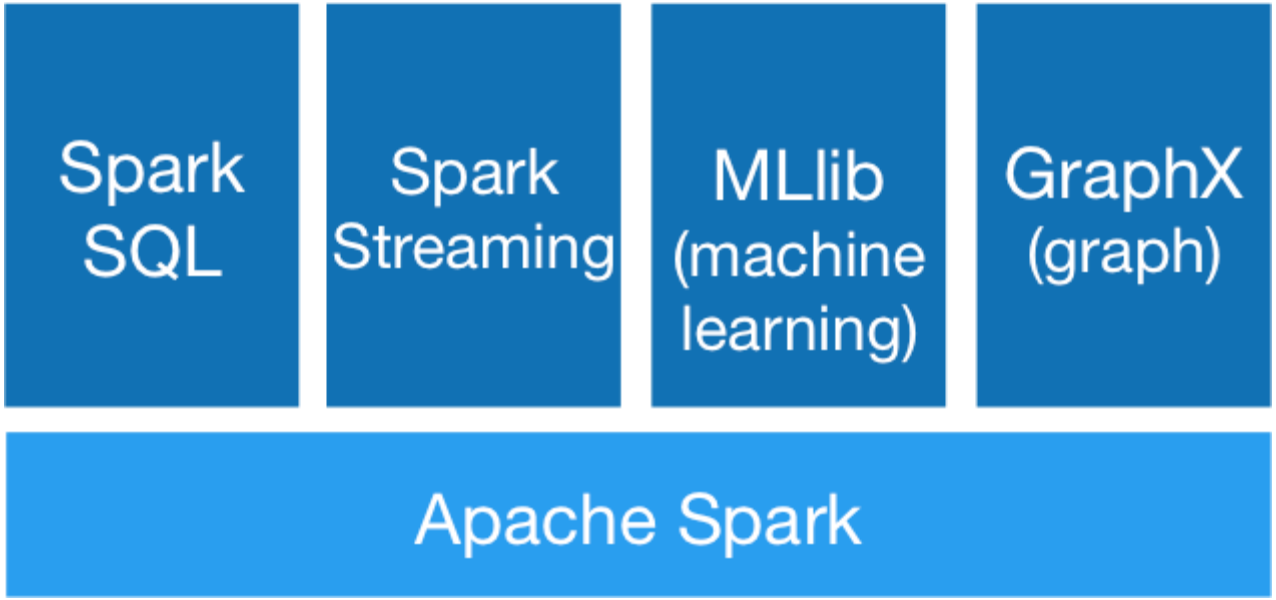
2014:

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

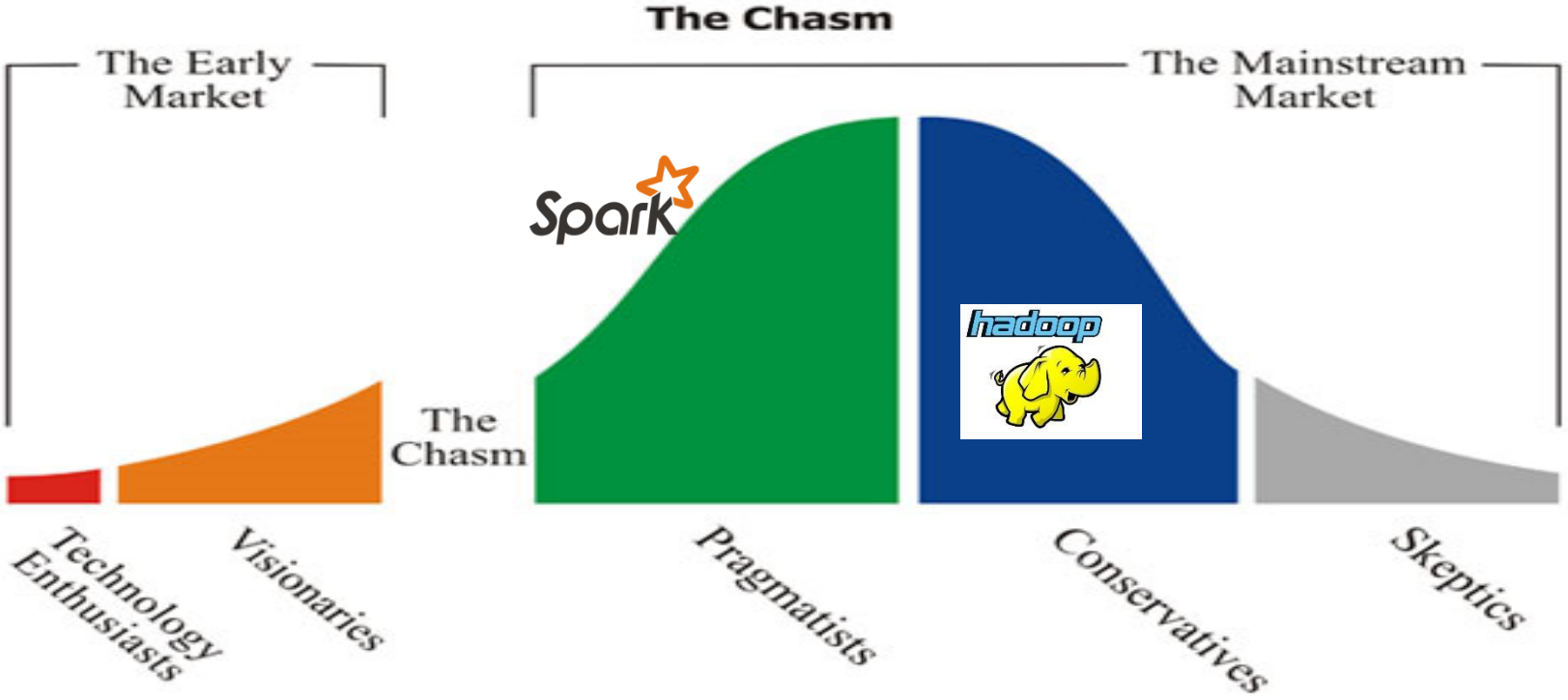
Source: https://databricks.com/blog/2014/10/10/spark-petabyte-sort.html



Source: Xiangrui Meng, Data Bricks



Technology adoption life cycle



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

QUESTIONS

Enjoying the bootcamp?

We'd love it if you could write a short review of Data Science Dojo!

Switch Up (https://www.switchup.org/bootcamps/data-science-dojo)
Course Report (https://www.coursereport.com/schools/data-science-dojo)

Your reviews help other people find and attend our bootcamp.

datasciencedojo
data science for everyone