Big Data Engineering with Distributed Systems

Data Science Dojo

datasciencedojo

data science for everyone

Agenda

- Introduction:
  - Data engineering for data scientists
  - The “5 Vs” of Big Data
- A key problem – machine learning at scale
- Distributed computing with Apache Hadoop & Hive
- Hadoop in the Azure cloud
- Machine learning at scale with Apache Mahout
- Distributed computing v2.0 – Apache Spark

Data Engineering for Data Scientists

Driving a car

vs

Servicing a car

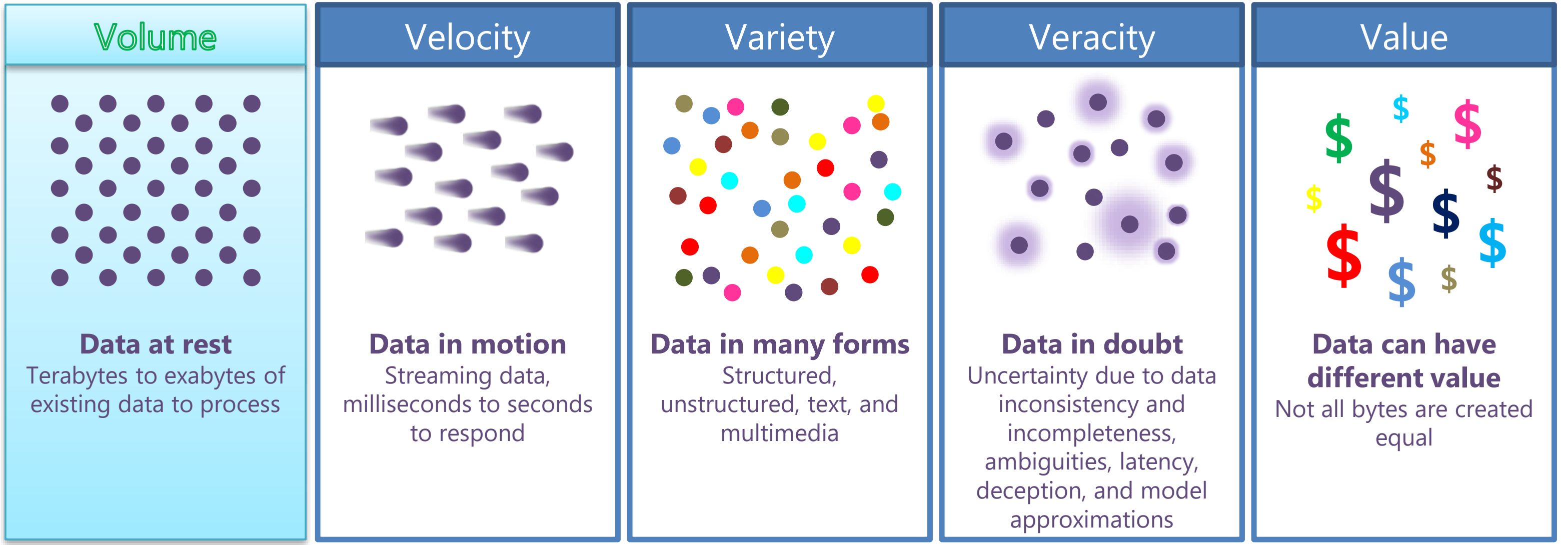
Goals:

- Teach you about data engineering topics/concepts

Non goals:

- Managing or administering a Hadoop cluster

5 Vs of Big Data



Volume

Data at rest
Terabytes to exabytes of existing data to process

Velocity

Data in motion
Streaming data, milliseconds to seconds to respond

Variety

Data in many forms
Structured, unstructured, text, and multimedia

Veracity

Data in doubt
Uncertainty due to data inconsistency and incompleteness, ambiguities, latency, deception, and model approximations

Value

Data can have different value
Not all bytes are created equal

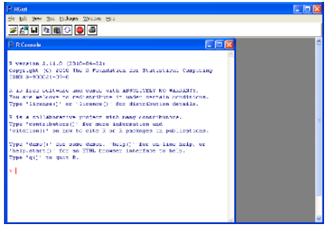
- **Goal:** As data scientists we want cost-effective access to the raw materials for our data products!

MACHINE LEARNING AT SCALE

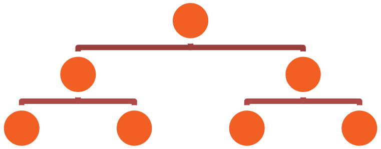
OSS R Limits

- Single core
- Single threaded

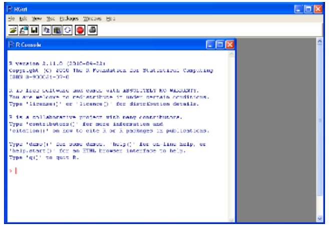
Quad Core Laptop



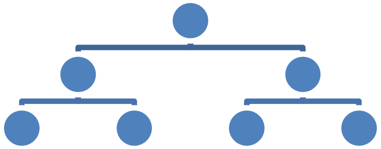
```
R version 2.11.0 (2010-04-22)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```



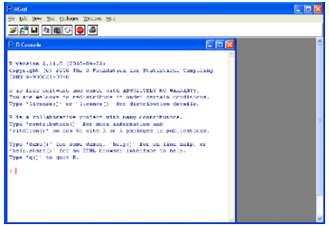
Model A



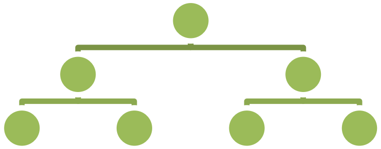
```
R version 2.11.0 (2010-04-22)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```



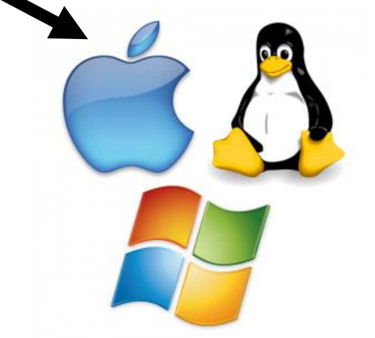
Model B



```
R version 2.11.0 (2010-04-22)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```



Model C



OSS R Limits

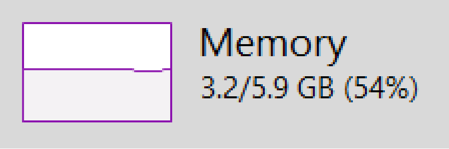
- Single core
- Single threaded
- All in memory (RAM)
- Vectors & Matrices capped at 4,294,967,295 elements (rows) if 32-bit version; 2^32 - 1

OSS R Limits: RAM

- All in memory (RAM)

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

Laptop Example:



Memory

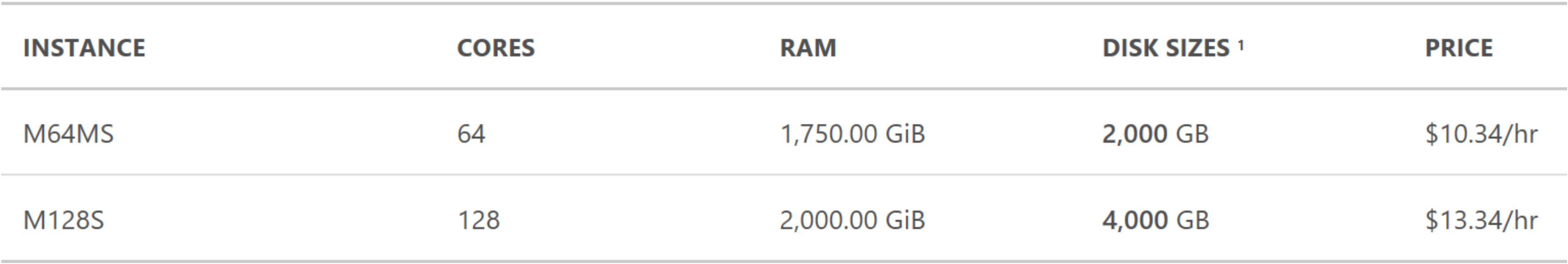
3.2/5.9 GB (54%)

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

*R data frames actually bloats data files by 3x

$$R\ Data\ Limit = {\sim}1.52gb\ \div 3 = {\sim}506.7mb$$

OSS R Limits: RAM



INSTANCE	CORES	RAM	DISK SIZES [1]	PRICE
M64MS	64	1,750.00 GiB	**2,000** GB	$10.34/hr
M128S	128	2,000.00 GiB	**4,000** GB	$13.34/hr

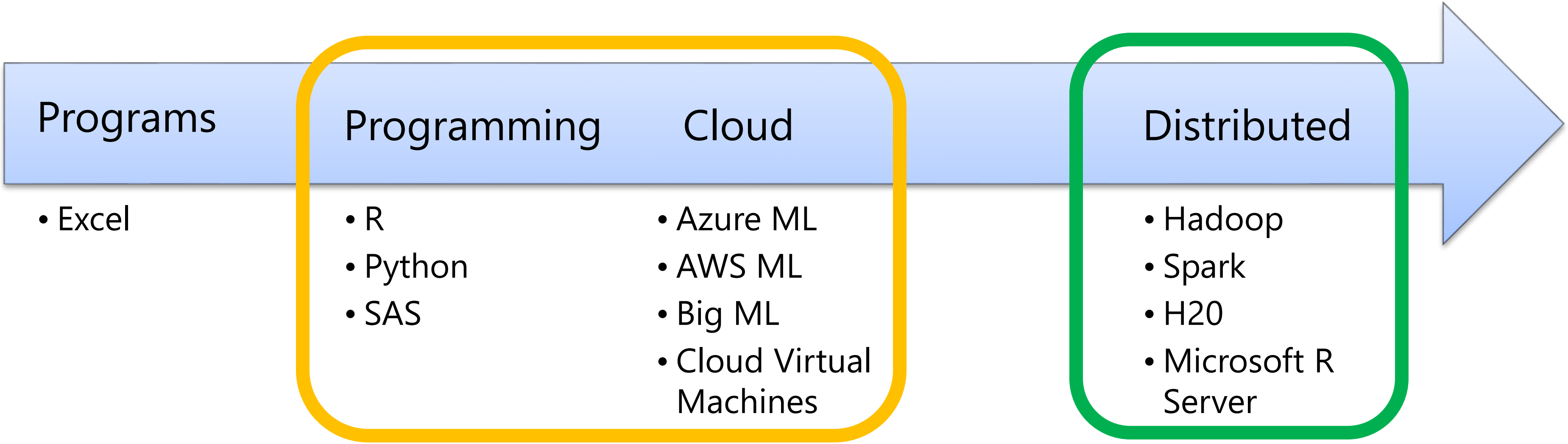
Azure's VM with largest RAM*:

$$Max\ Data\ Limit = (\ 2000gb\ x\ 80\%\ )\ - 1gb$$
$$Max\ Data\ Limit = \sim 1600gb$$
$$R\ Data\ Limit = \sim 1600gb\ \div 3 = \sim 533.33\ gb$$

24x7x52 Annual Cost: $116,938.44!

*Data collected 06/07/2017

Machine Learning Scaling



Programs

- Excel

Programming

- R
- Python
- SAS

Cloud

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

Distributed

- Hadoop
- Spark
- H20
- Microsoft R Server

This only gets us so far!

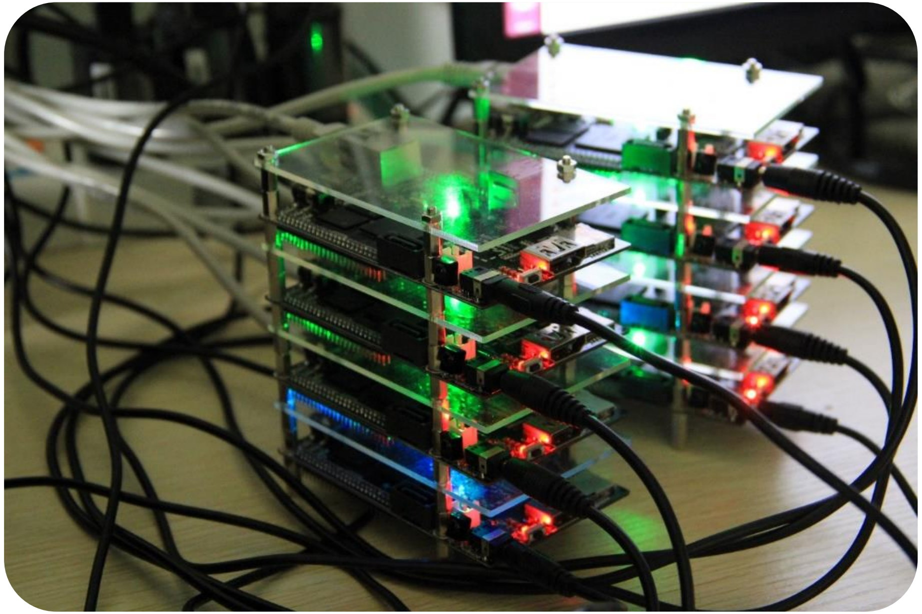
Big data scale!

DISTRIBUTED COMPUTING WITH APACHE HADOOP

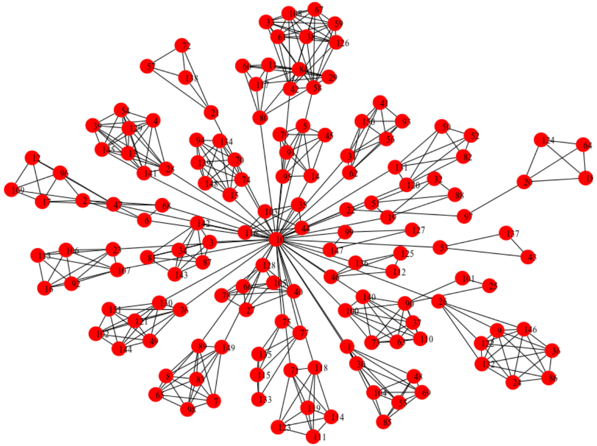
Turn Back The Clock, The Mainframe

- "Big Iron"
- Backbone of computing for decades.
- Still widely used.
- "Scale-up" model of shared computing.
- Core platform is cost effective, ecosystem is not (e.g., software licensing).
- The original VM host!

Distributed Computing

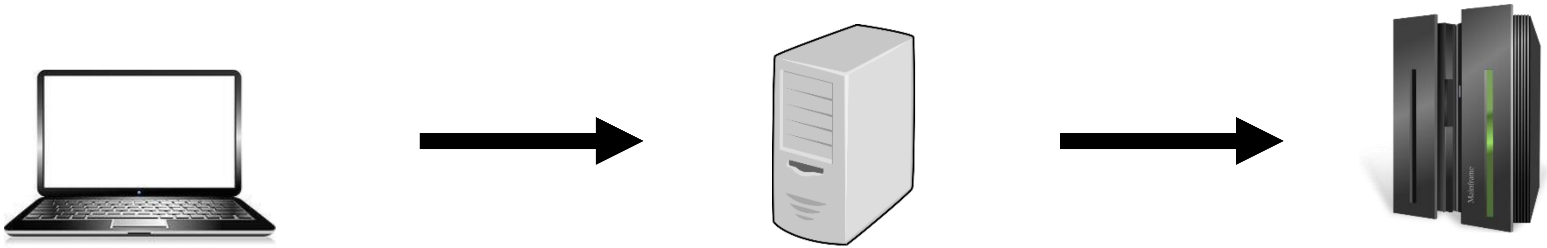


Cloud Computing



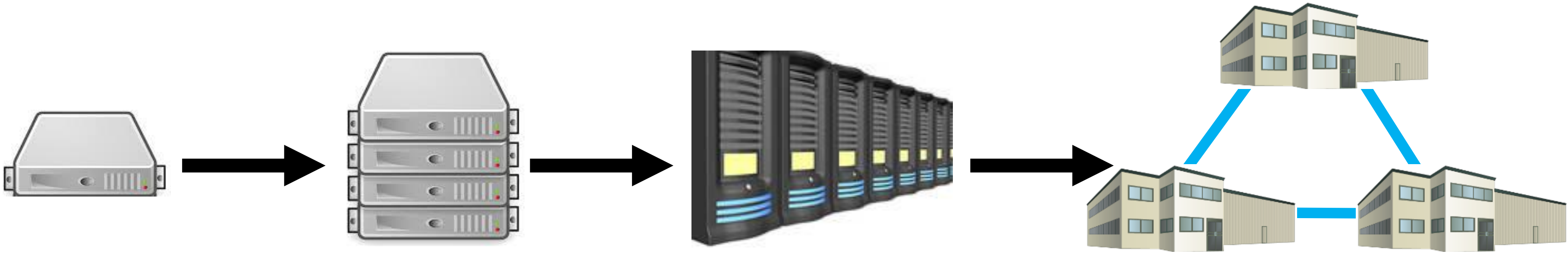
- Conceptually – a combination of mainframe and distributed computing.
- VM hosts are now the “Big Iron”.
- Many VMs work together to distribute workloads.
- Some workloads on dedicated HW (e.g., SAP HANA).

Scaling Computational Power



Old Scaling:

- Vertical Scaling, Scaling UP
- High performance computers



New Scaling:

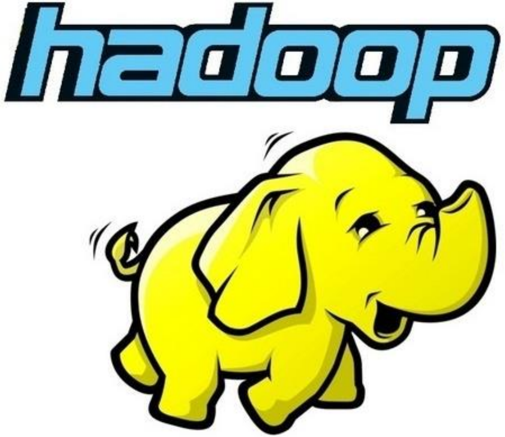
- Horizontal Scaling, Scaling OUT
- Commodity hardware, distributed

datasciencedojo

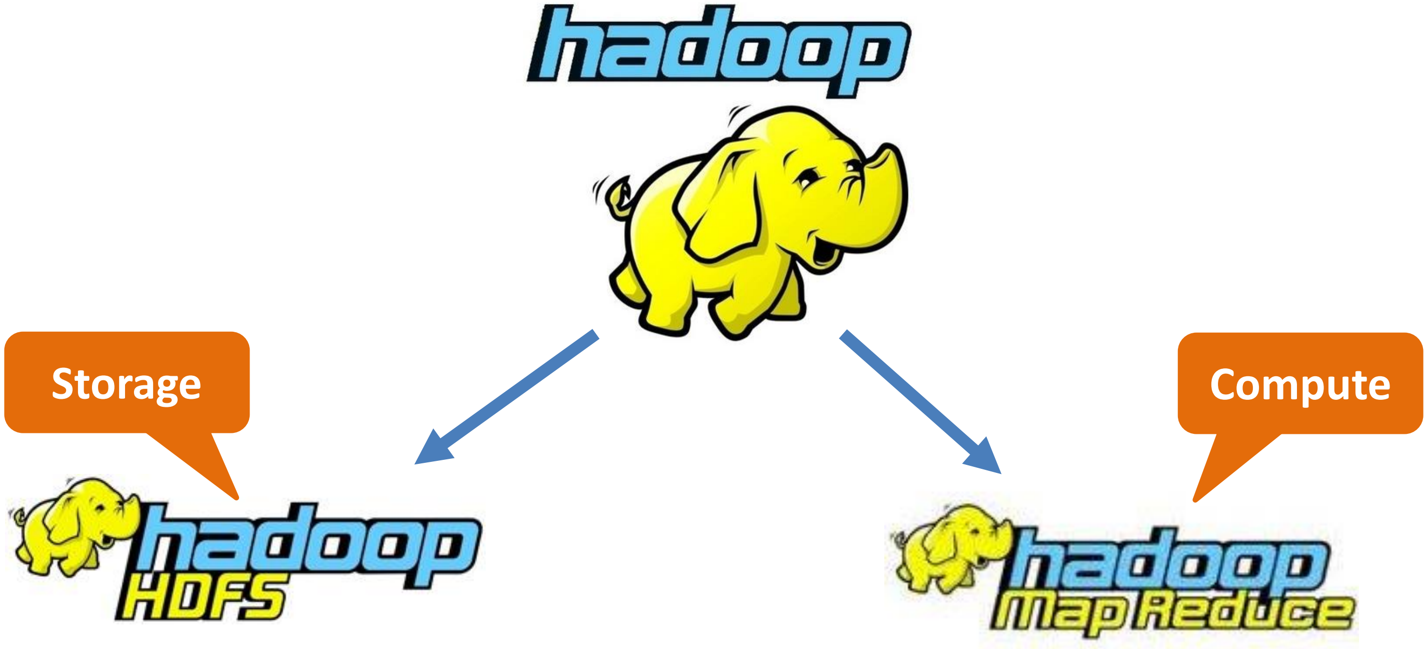
data science for everyone

What is Hadoop?

- OSS Platform for distributed computing over Internet-scale data.
- Originally built at Yahoo!
- Implementation of ideas (e.g., MapReduce) published by Google.
- The de facto standard big data platform.
- Named after a stuffed animal belonging to Doug Cutting's son.



Hadoop at Base



Storage

Compute

Distributed batch processing engine for big data.

HDFS & MapReduce

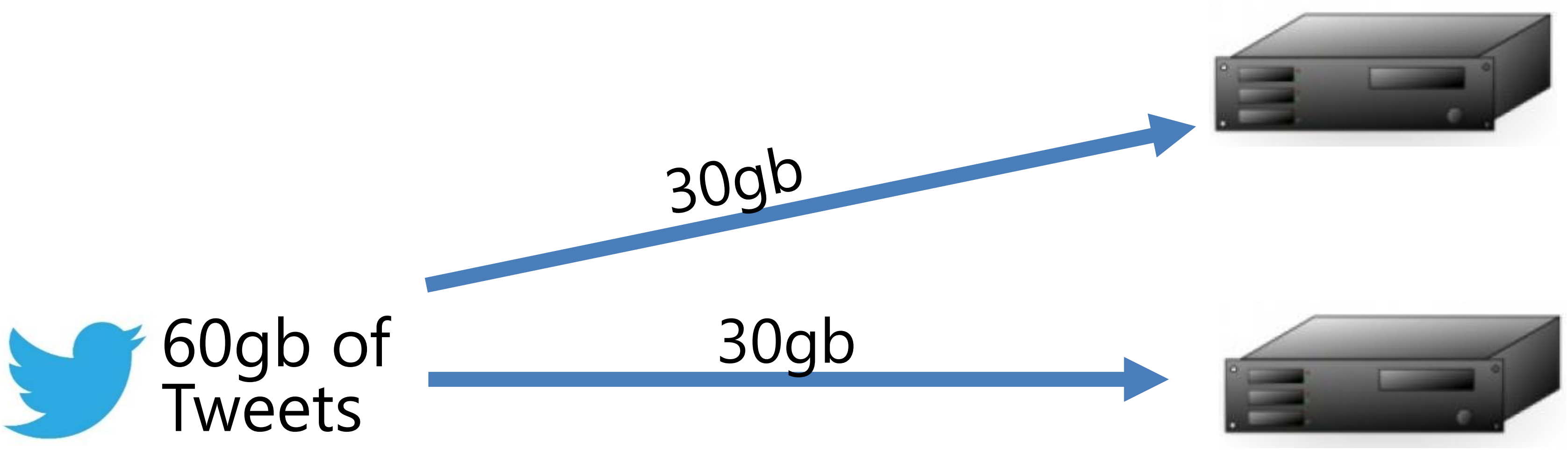
60gb of Tweets

60gb

1 Computer

Processing: 30 hours

HDFS & MapReduce



60gb of Tweets

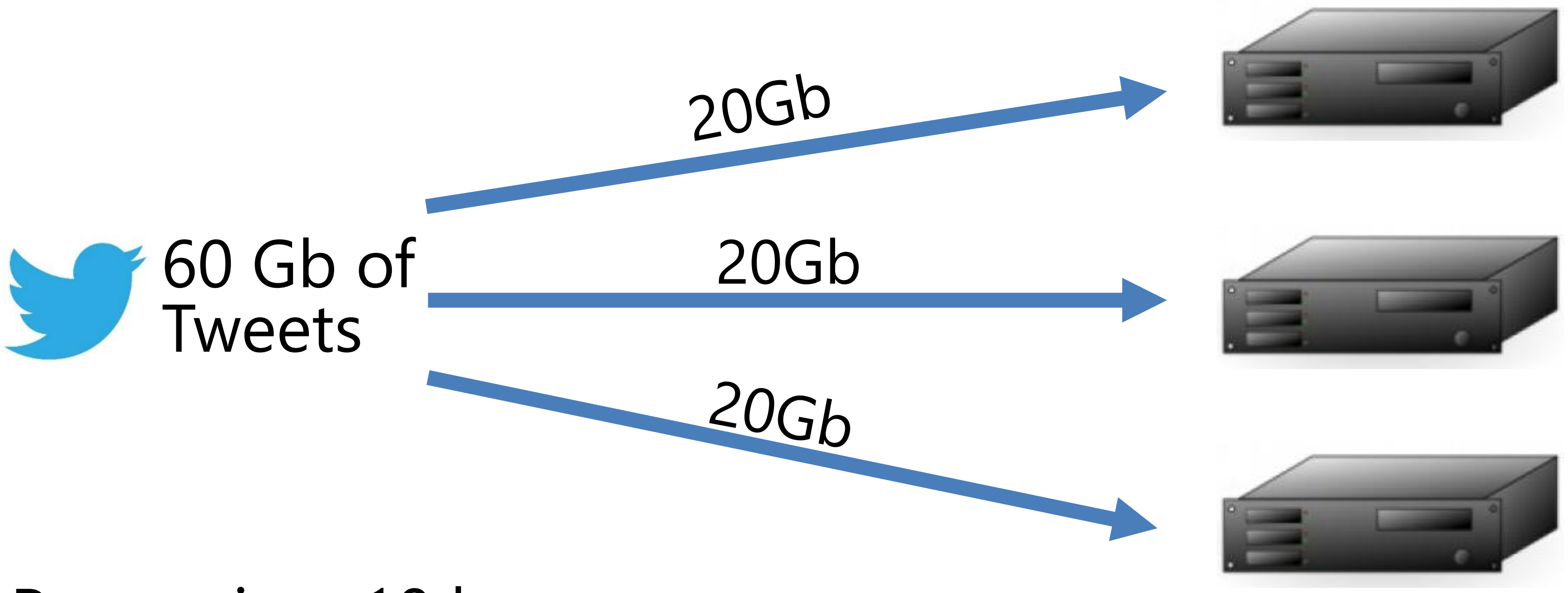
30gb

30gb

2 Computers

Processing: 15 hours

HDFS & MapReduce



60 Gb of Tweets

20Gb

20Gb

20Gb

Processing: 10 hours

3 Computers

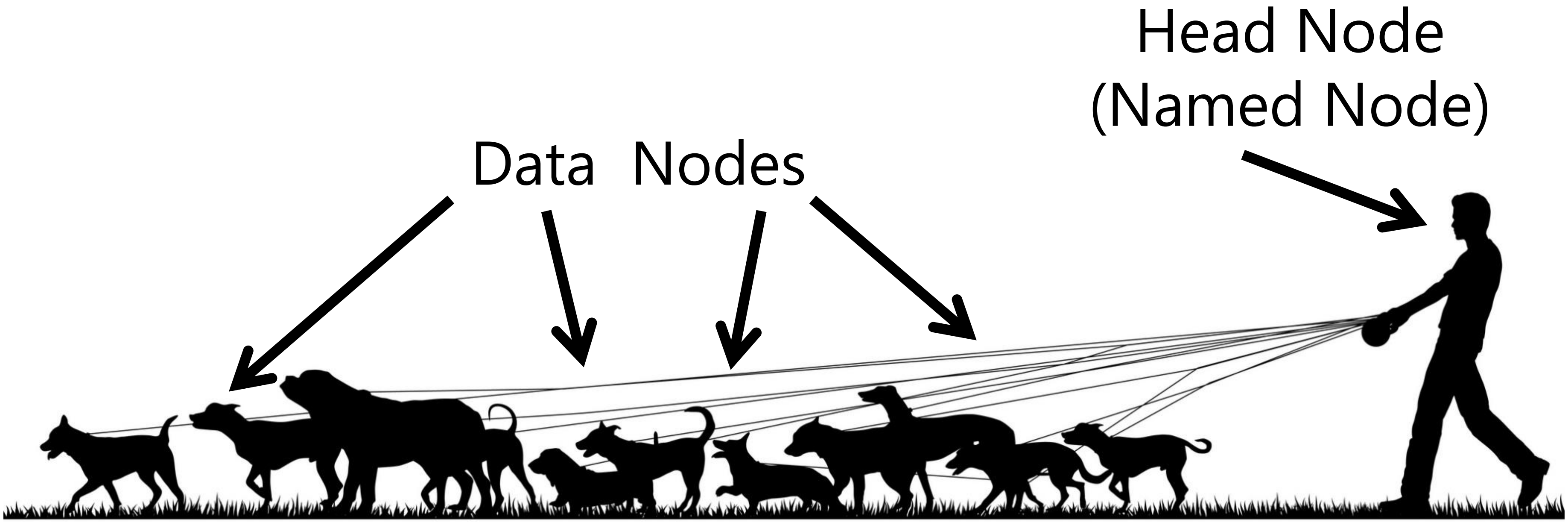
Most Cases, Linear Scaling Of Processing Power

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

datasciencedojo

data science for everyone

If dogs were servers...

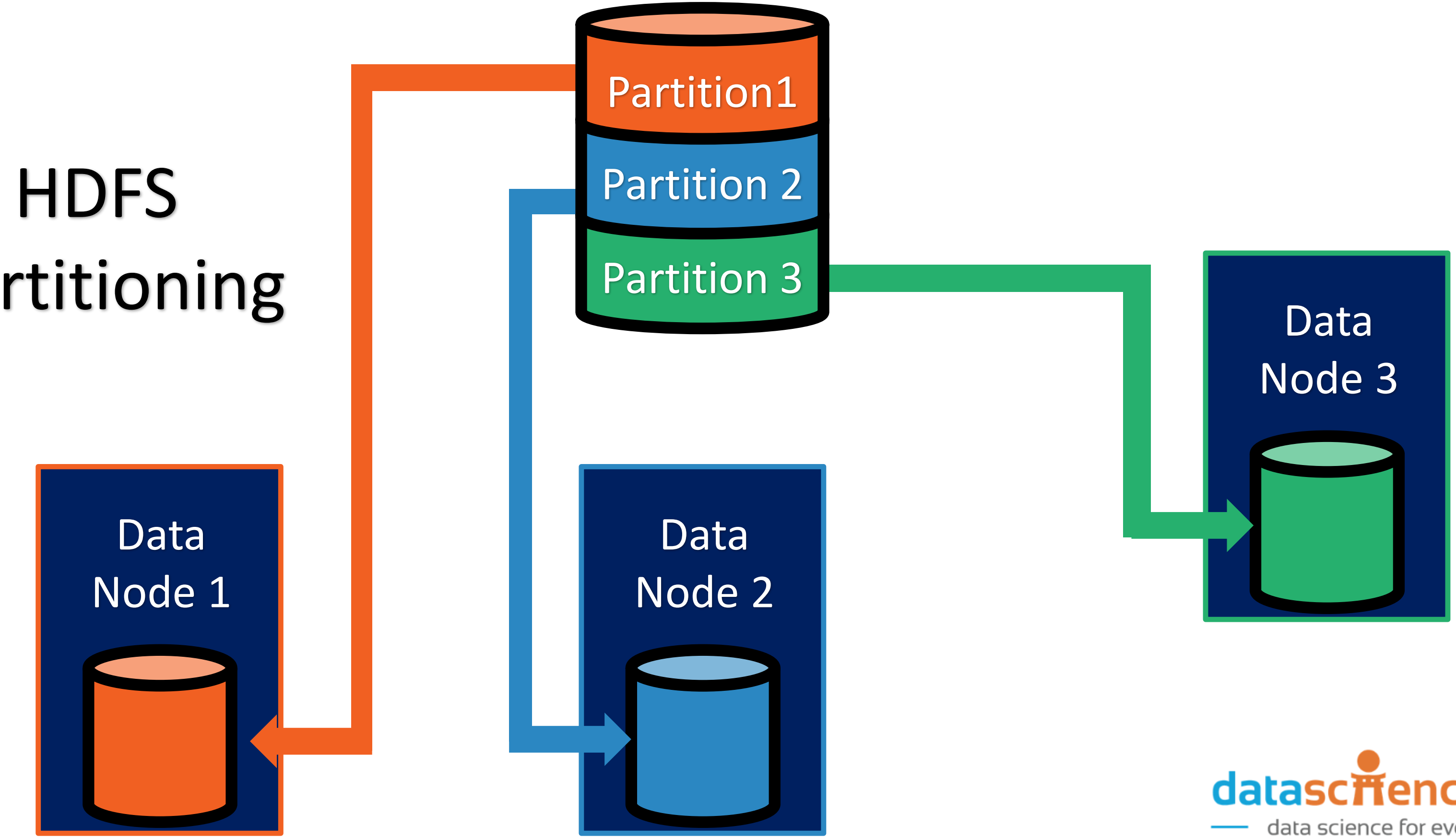


Head Node
(Named Node)

Data Nodes

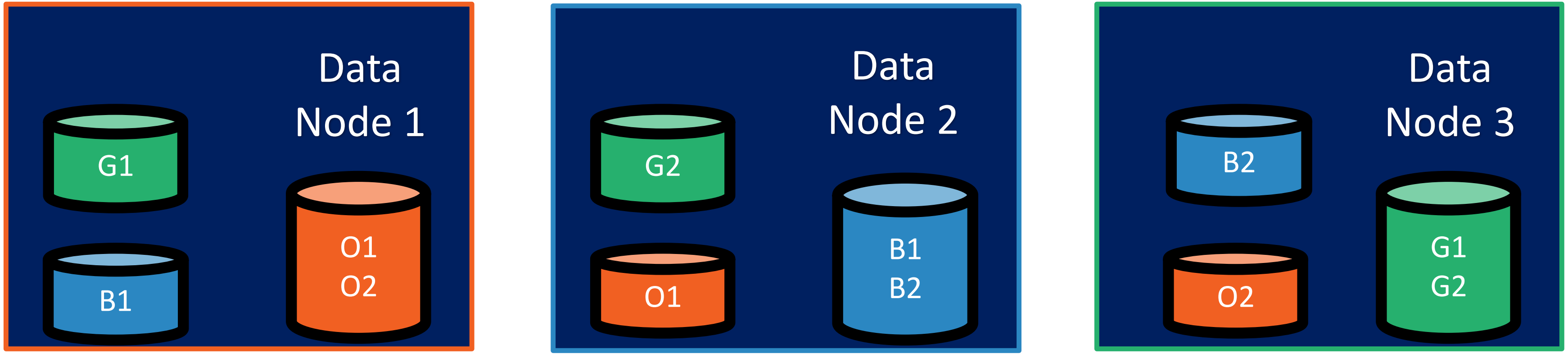
HDFS

HDFS Partitioning

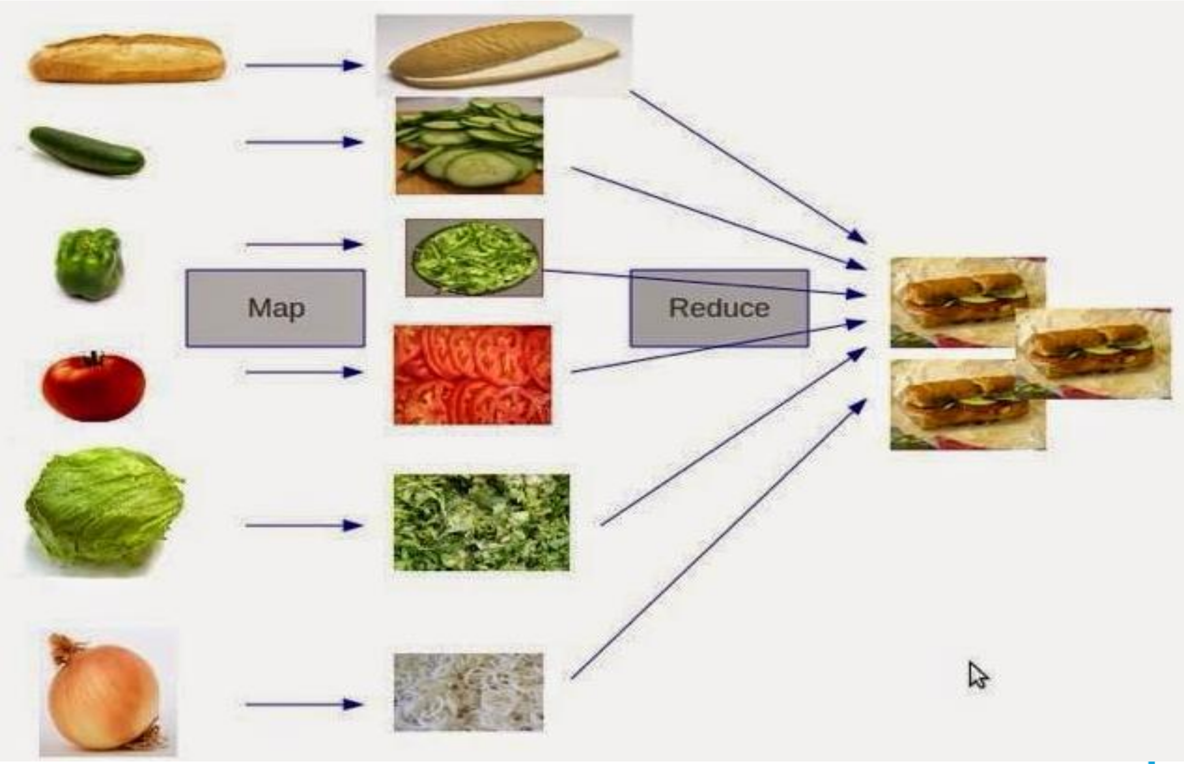


datasciencedojo
data science for everyone

HDFS Redundancy

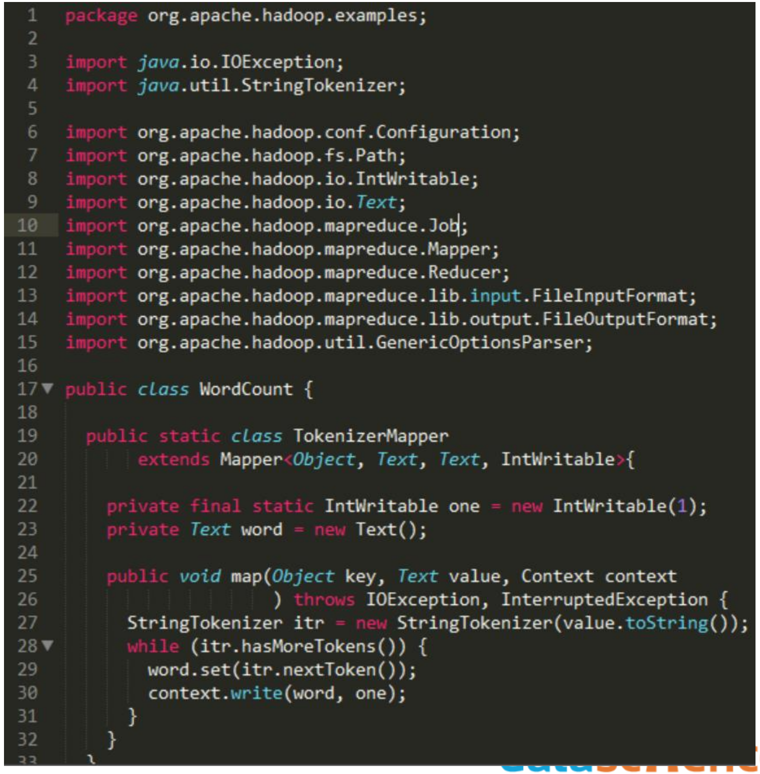


MapReduce – Sandwich Analogy



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);
      }
    }
  }
```

DISTRIBUTED COMPUTING WITH APACHE HIVE

What is Hive?

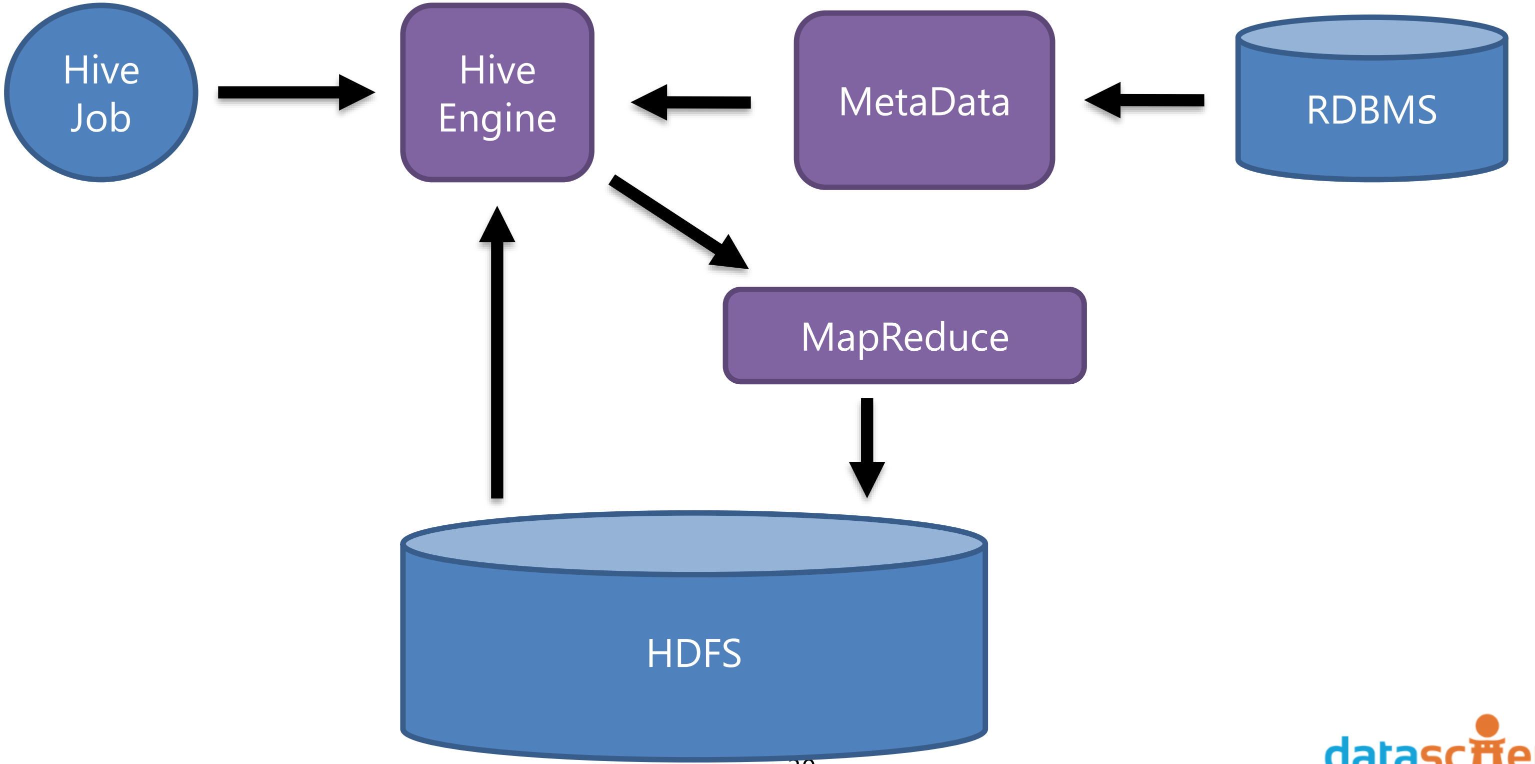
- Abstraction built on top of MapReduce & HDFS.
- Makes Hadoop look like an RDBMS (e.g., coding in SQL).
- Developed by Facebook to democratize Hadoop.
- Applies structure to data at runtime (“schema on read”).

Schema on Read

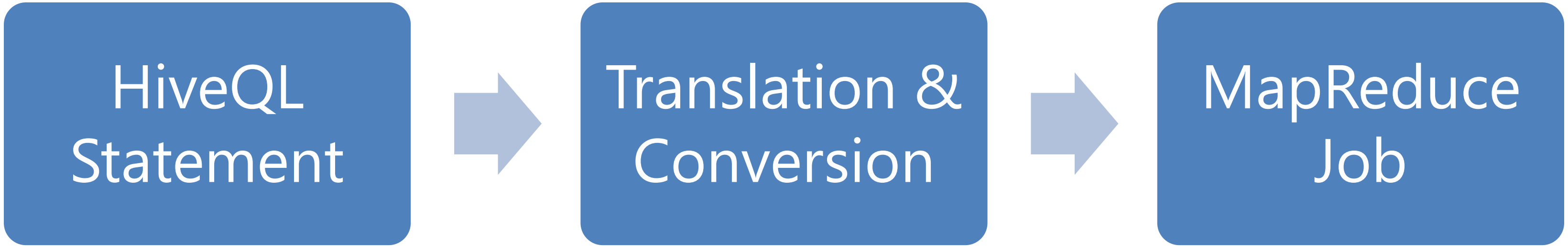
Data File = Unstructured Data

Data File + Metadata File/DB = Structured Data

Hive Architecture



Hive Jobs

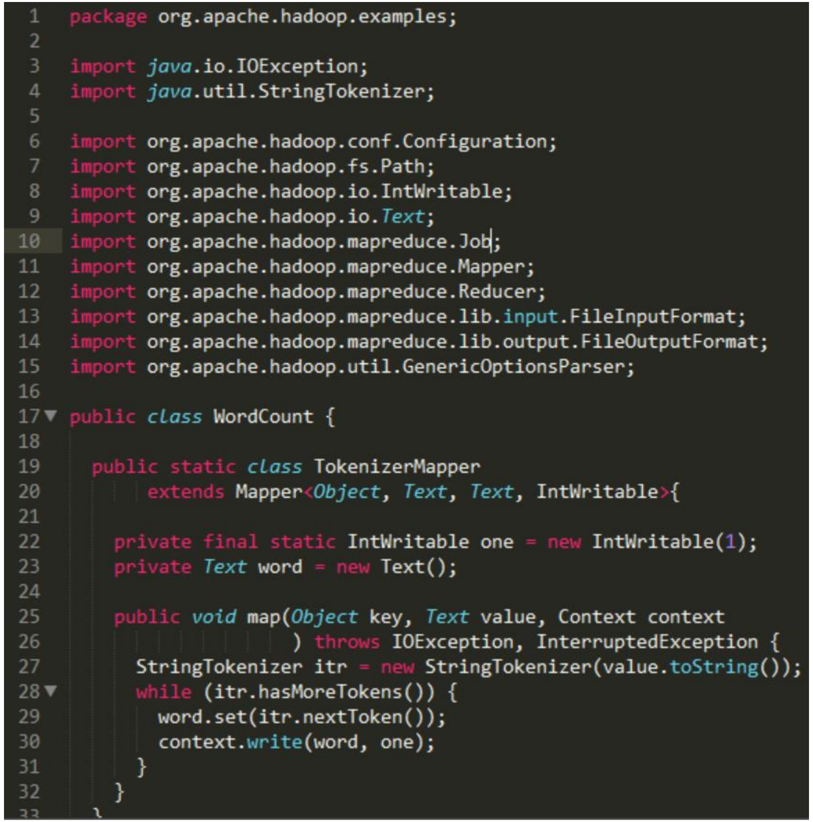


HiveQL Statement

Translation & Conversion

MapReduce Job

Word Count Revisited



```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);
      }
    }
  }
```

vs.

```sql
SELECT word,
       COUNT(*) AS word_count
FROM words
GROUP BY word
```

SQL Don’ts in HIVE

SELECT * FROM ANYTHING: This brings back everything. Everything doesn’t fit on a single computer.

JOIN: Join will take hours or days to perform and eat up all cluster bandwidth for everyone else trying to use it in the queue.

ORDER BY: Sorting is very computationally expensive.

Sub Queries: A sub query essentially creates a secondary table, which will be huge in HIVE.

Interactivity: SQL in DBMS is interactive because its almost instantaneous.

Execution Engine: Tez

The Stinger Initiative

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

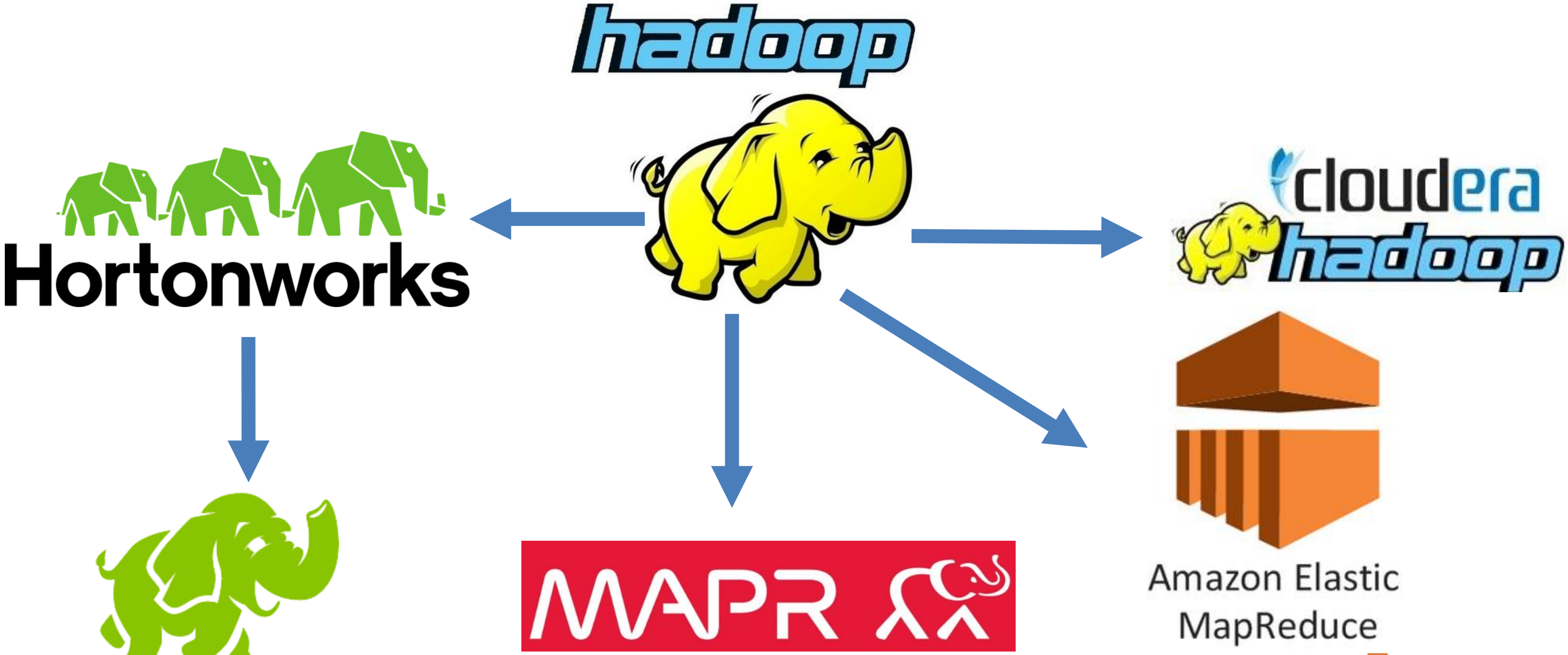
- 44 companies
- 145 developers
- 392k lines of Java code

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

HADOOP IN THE AZURE CLOUD

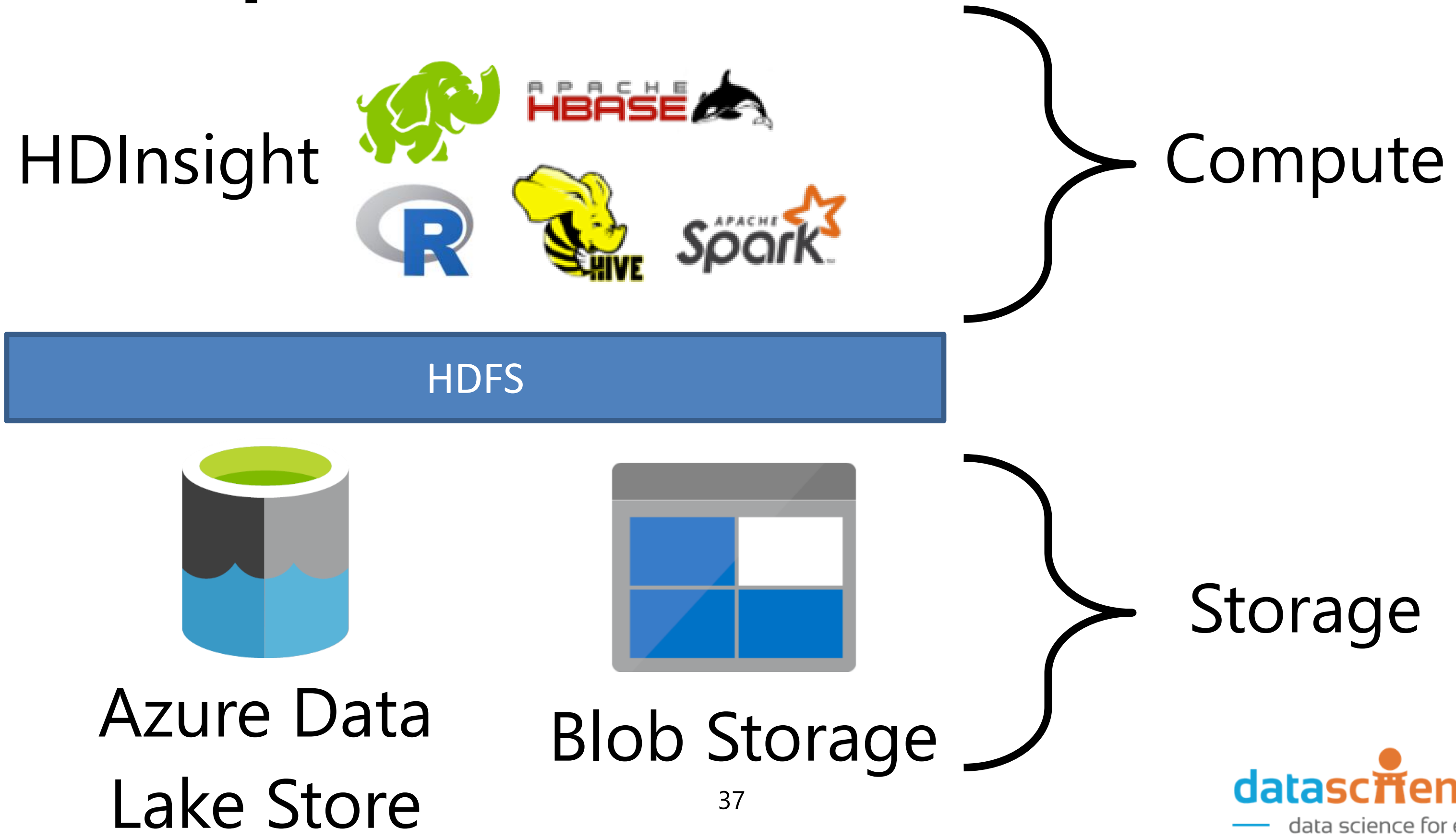
Hadoop Implementations



Amazon Elastic MapReduce

HDInsight

Hadoop in Azure



HDInsight

Compute

HDFS

Azure Data Lake Store

Blob Storage

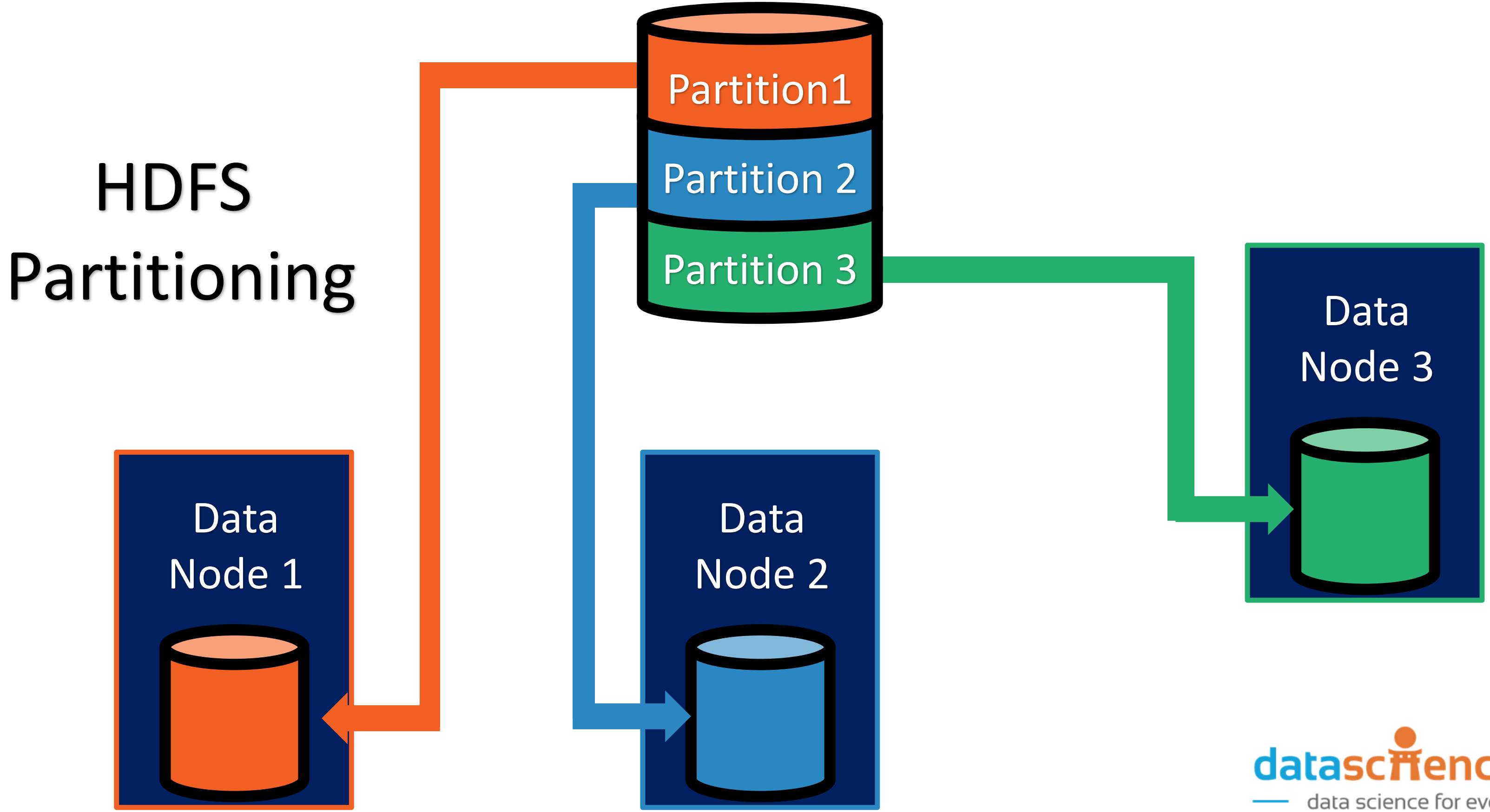
Storage

MACHINE LEARNING AT SCALE - REVISITED

What is Mahout?

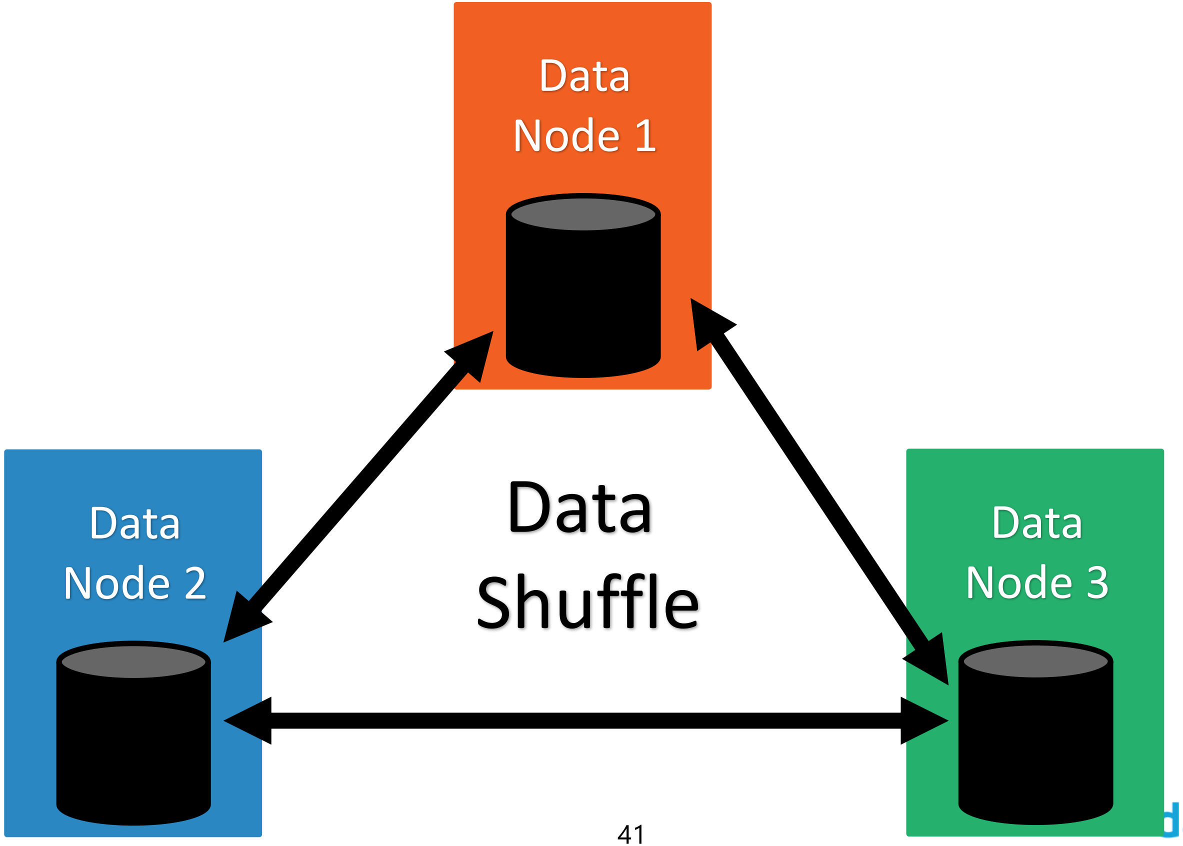
- Distributed Machine Learning platform.
- Built on top of MapReduce and HDFS.
- Script-based and command line interfaces.
- R-like language implementation.

Distributed Random Forest

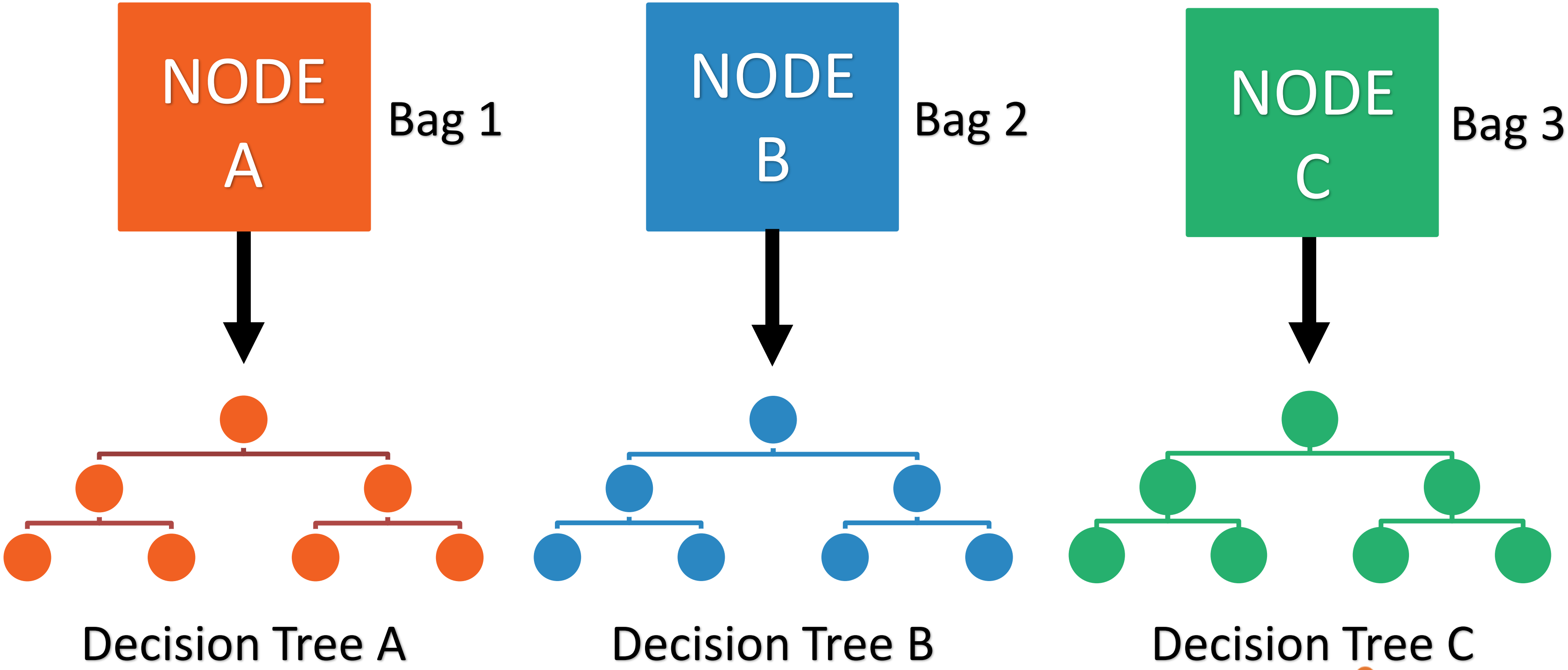


datasciencedojo
data science for everyone

Distributed Random Forest

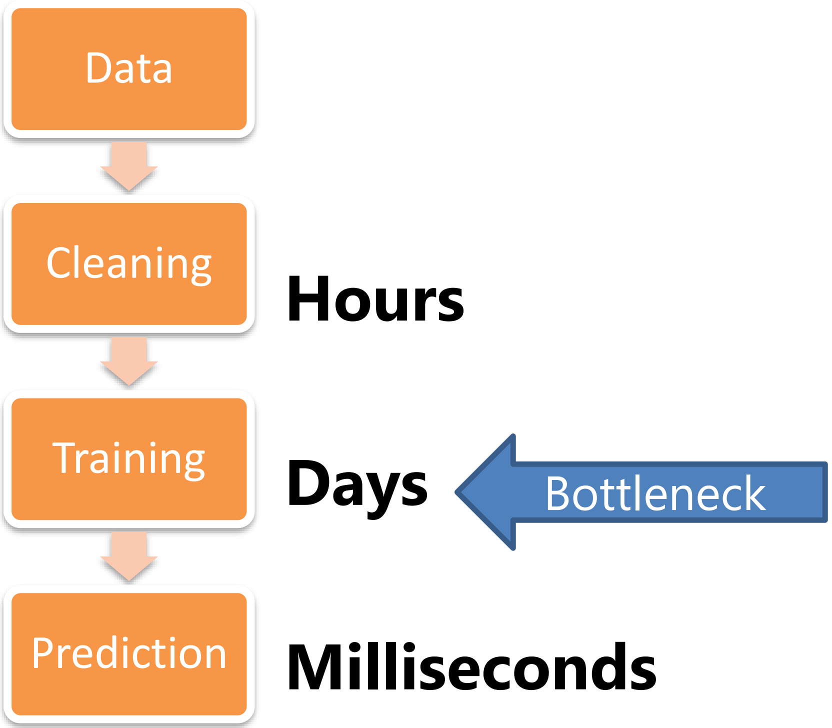


Distributed Random Forest



datasciencedojo
data science for everyone

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

DISTRIBUTED COMPUTING V2.0 – APACHE SPARK

What is Spark?

- "A fast and general engine for large-scale data processing."

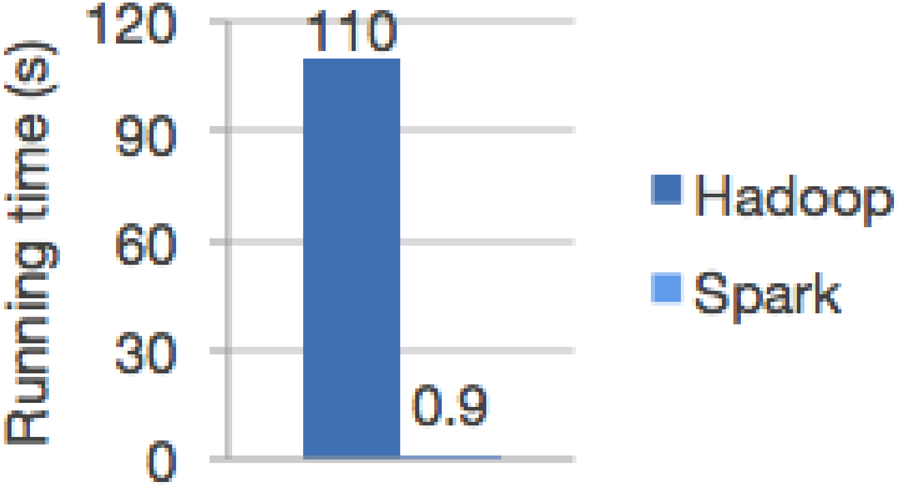
- Designed to incorporate the goodness of Hadoop and address Hadoop's shortcomings.

- Can complement Hadoop via integration with both HDFS and Hive.

Why Spark? Improved Perf!

- Up to 10x faster than Hadoop working with data from disk.

- Up to 100x faster working with data stored in memory!



Big Data, Faster!

3x faster on 10x fewer machines!

Daytona GraySort Contest: Sort 100 TB of data!

<u>Previous World Record:</u>

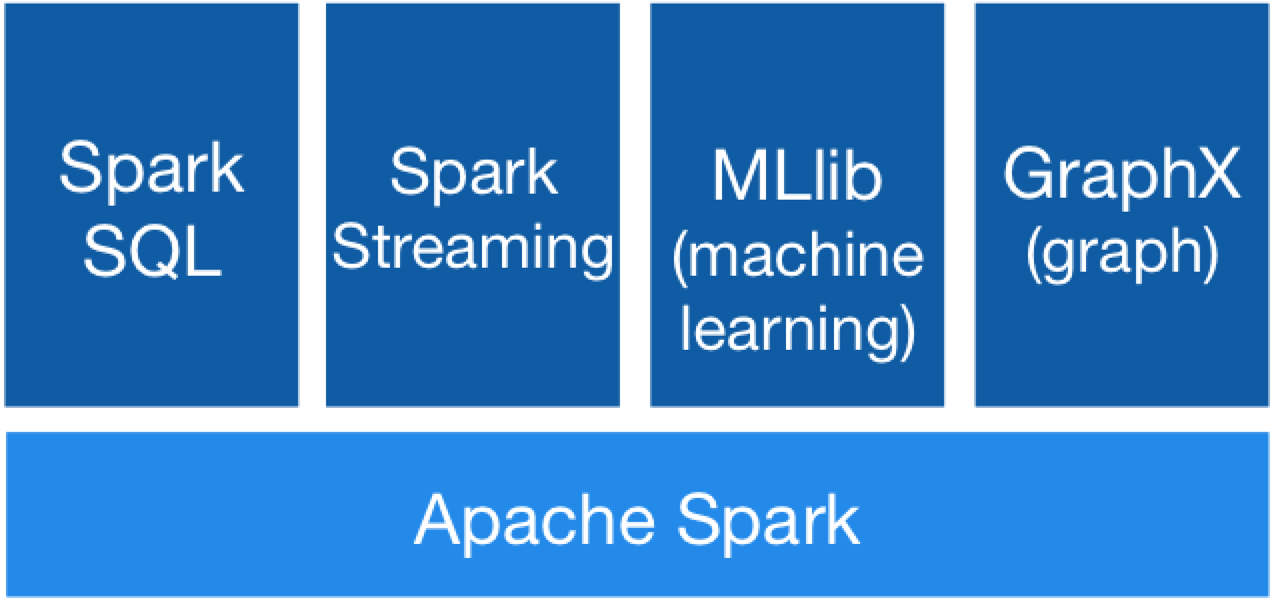
- Method: Hadoop
- Yahoo!
- 72 Minutes
- 2100 Nodes

<u>2014:</u>

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

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

Conceptual Architecture



Spark SQL

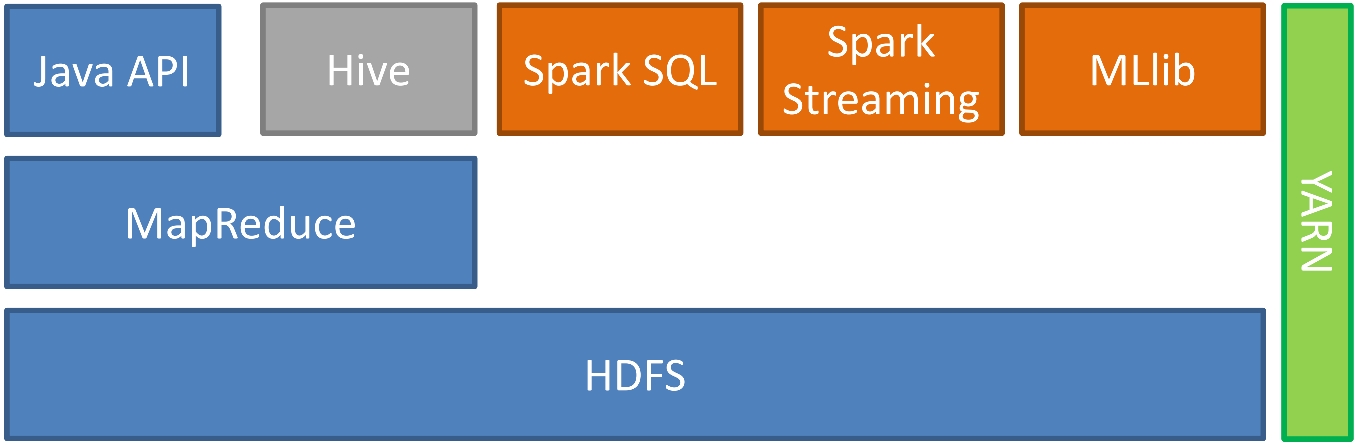
Spark Streaming

MLlib (machine learning)

GraphX (graph)

Apache Spark

Spark and Hadoop



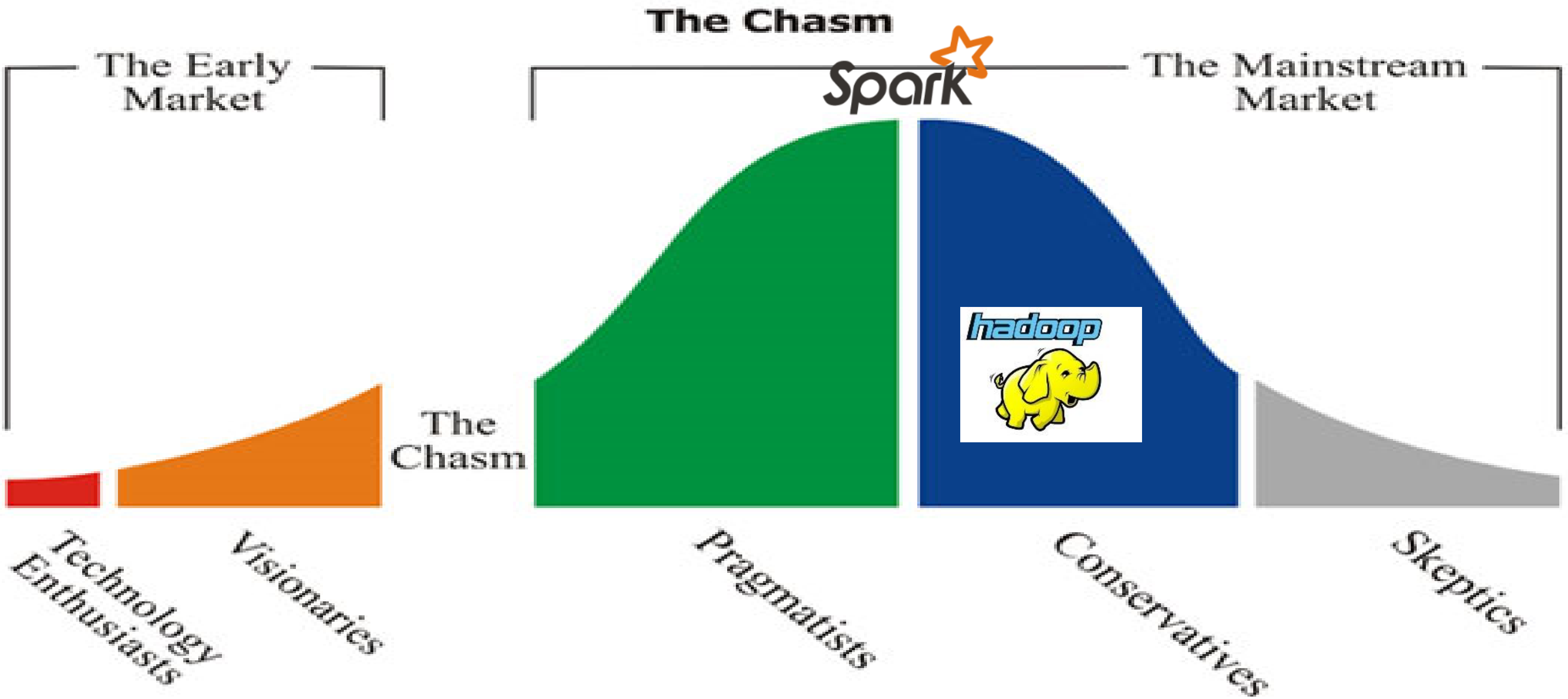
- Spark can be deployed on a Hadoop cluster and share cluster resources via YARN.
- Spark, however, does not require Hadoop!

Why is Spark Faster?

- First, Spark processing implements *lazy execution*:
  - Data operations are either *transformations* or *actions*.
  - Transformations are not executed immediately, but are stored.
  - When an action is issued, Spark evaluates all stored transformations and optimizes processing before executing.

- Second, Spark performs most processing in-memory:
  - RAM is far faster than using disk storage – even SSD drives.
  - More RAM in the cluster allows Spark to processes data faster.

Technology adoption life cycle



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

QUESTIONS

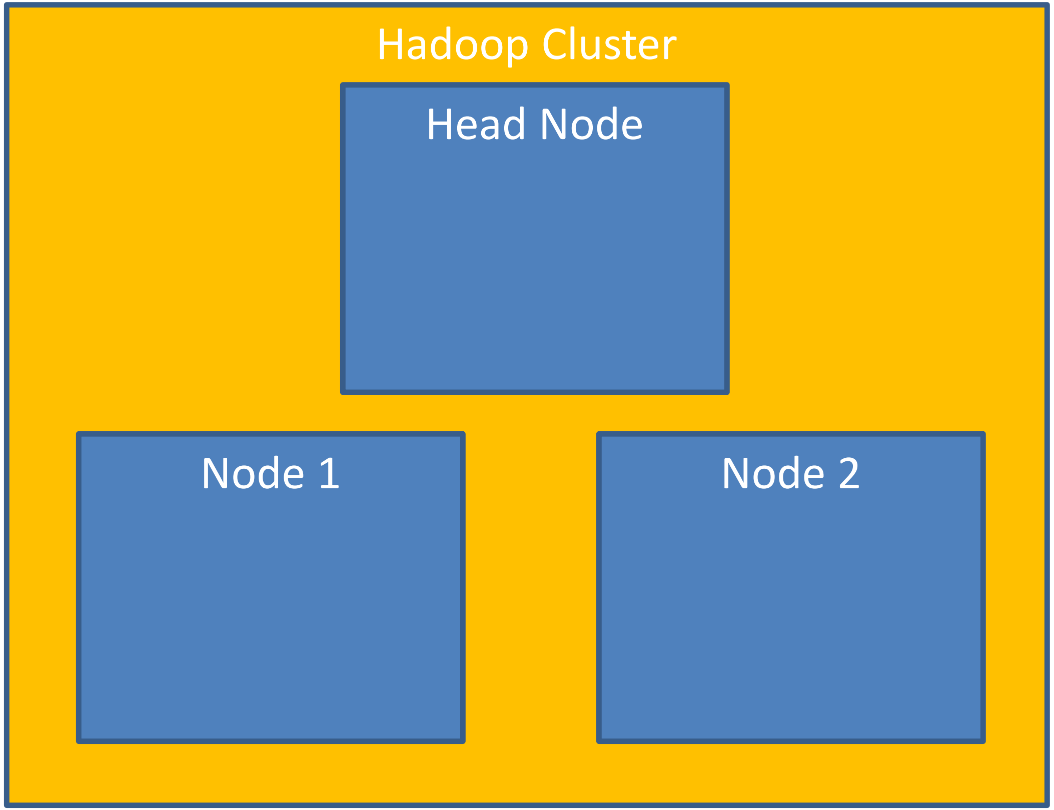
APPENDIX

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.

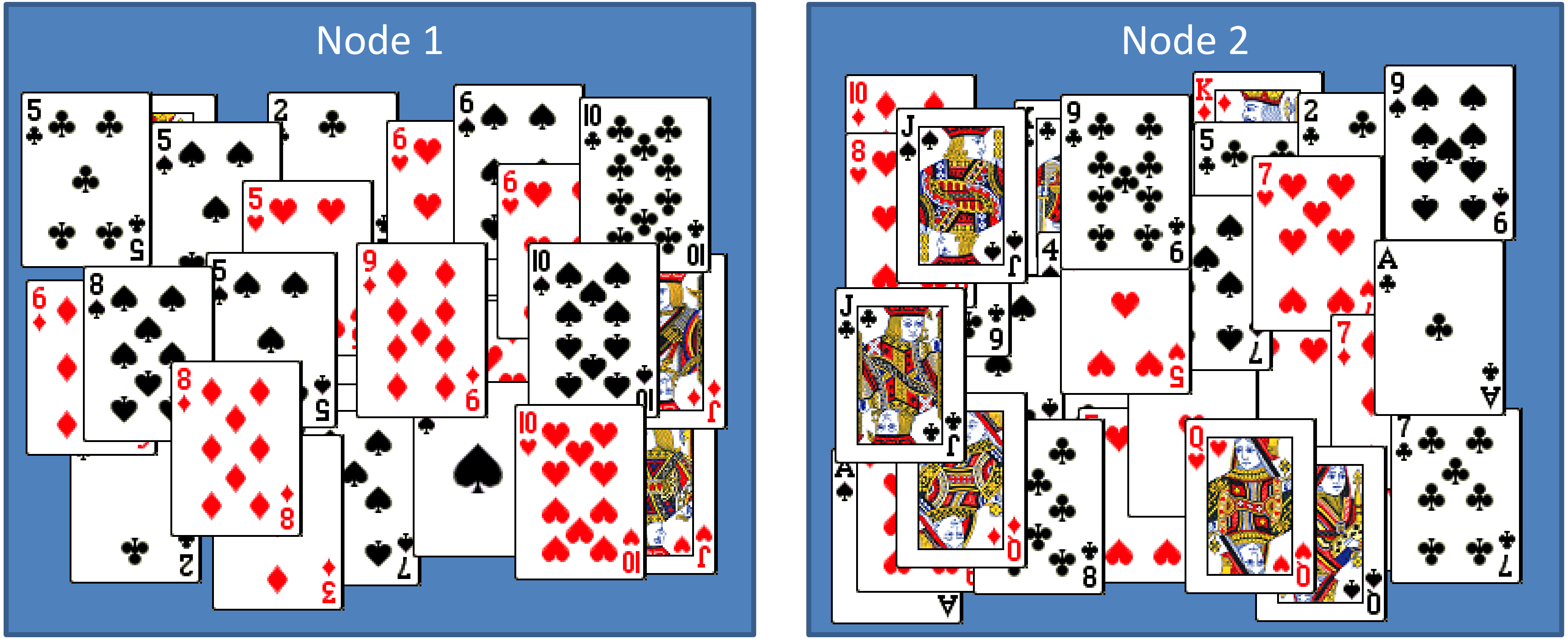
- Each card represents a row of data
- Each suit & number represents an attribute of the data

Using a 2 Data Node Cluster

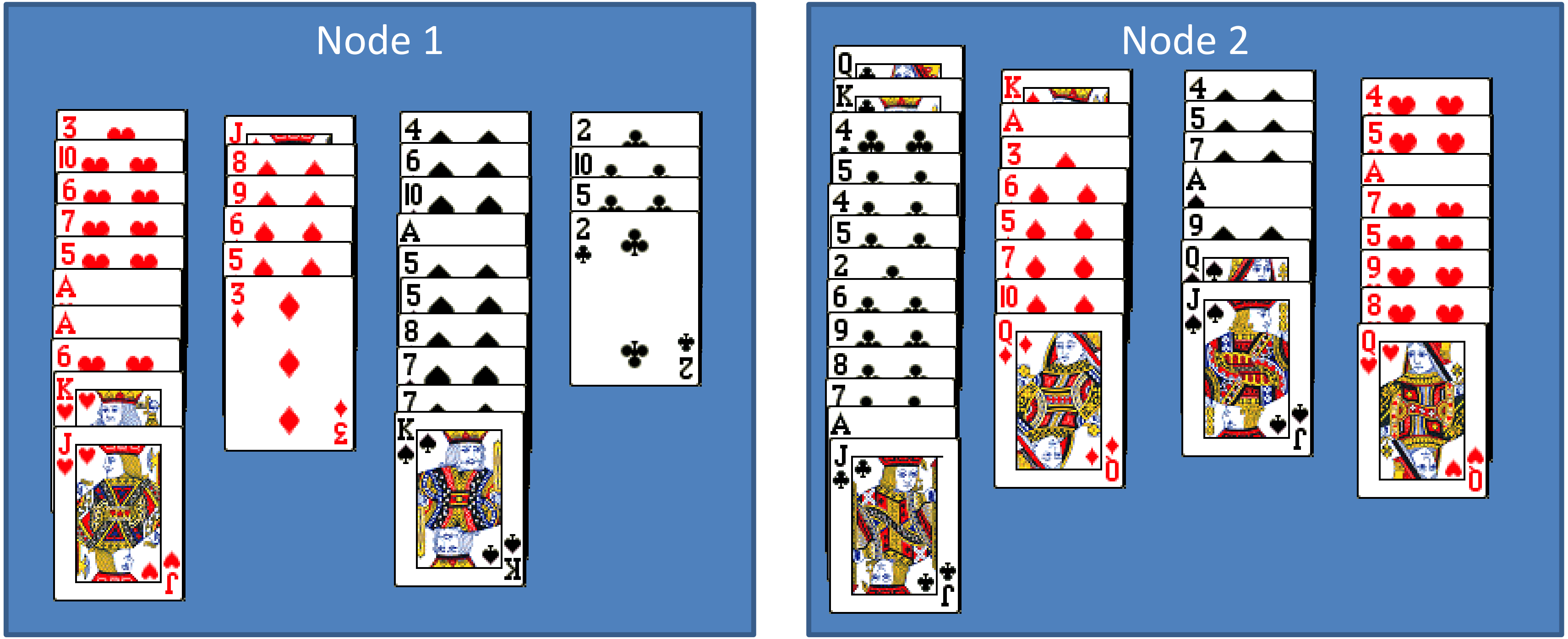


datasciencedojo
data science for everyone

Mapping: Each Node's HDFS



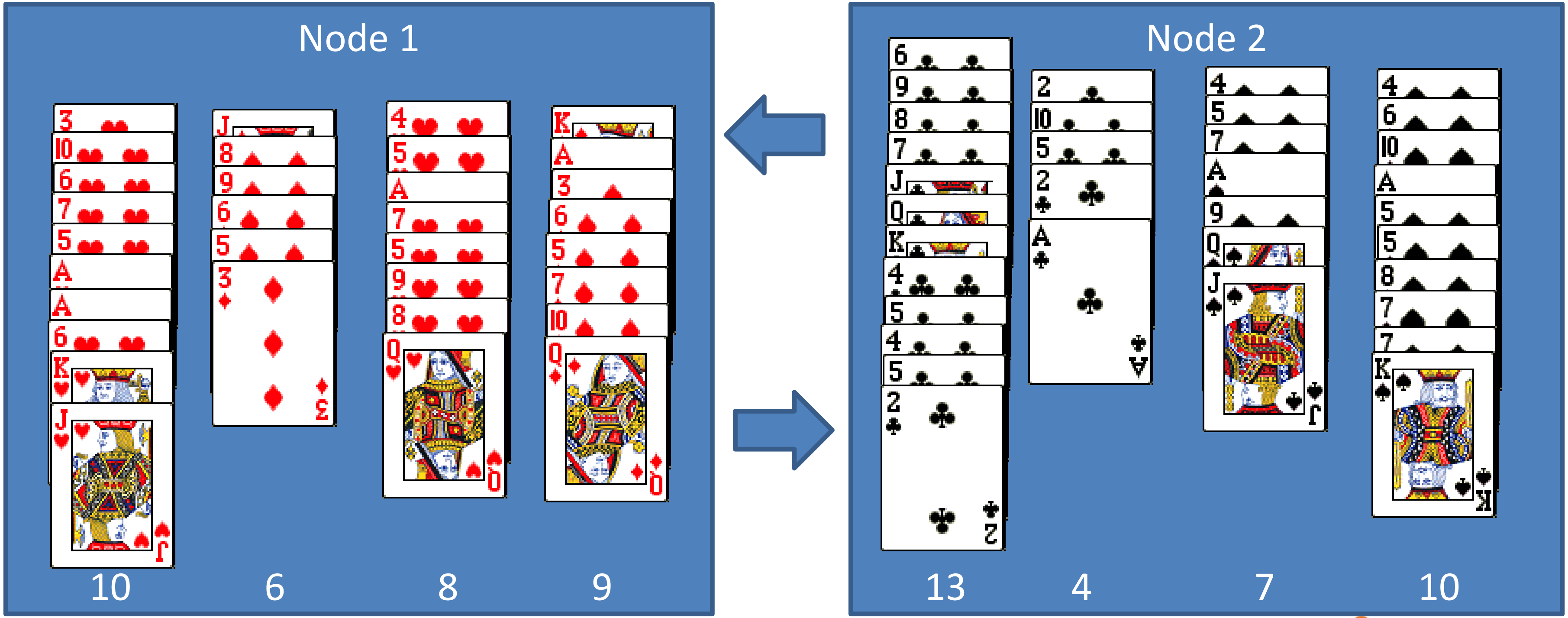
Mapping: Node Sorting



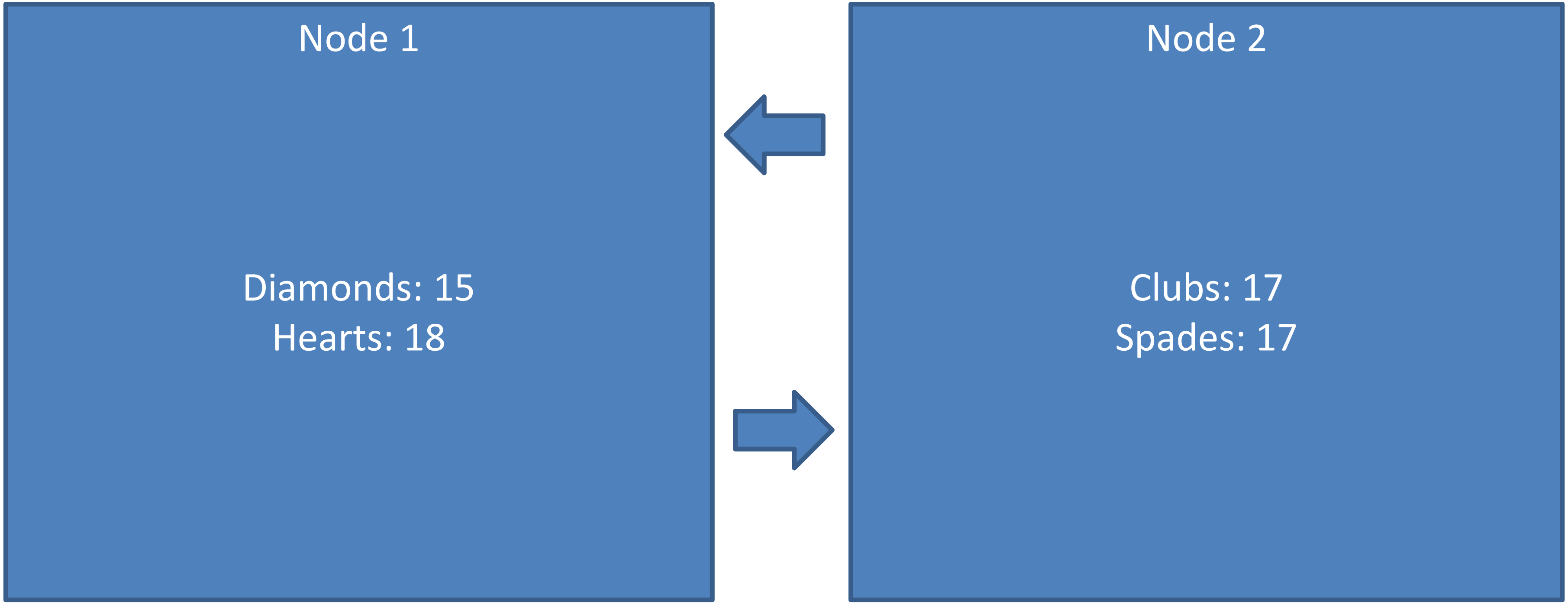
Node 1

Node 2

Mapping: Node Shuffle, Data Transfer



Mapping: Node Shuffle, Data Transfer



Node 1

Diamonds: 15
Hearts: 18

Node 2

Clubs: 17
Spades: 17