# INFO-H515: BIG DATA: DISTRIBUTED MANAGEMENT

## Lecture 1: Introduction, HDFS, Map/Reduce

Dimitris Sacharidis

# LECTURE OUTLINE

What is Big Data?

How is Big Data Used?

Big Datacenters

HDFS

Map/Reduce

# WHAT IS BIG DATA?

*“Big data is like teenage sex:*
*everyone talks about it,*
*nobody really knows how to do it,*
*everyone thinks everyone else is doing it,*
*so everyone claims they are doing it …”*

—Dan Ariely

# THE BIG THREE


BIG DATA

DATA ANALYTICS

DATA SCIENCE

# THE BIG THREE


BIG DATA

# THE BIG THREE


BIG DATA

Some History

# The data deluge

Source: The Economist, February 25, 2010

# The Data Deluge

"

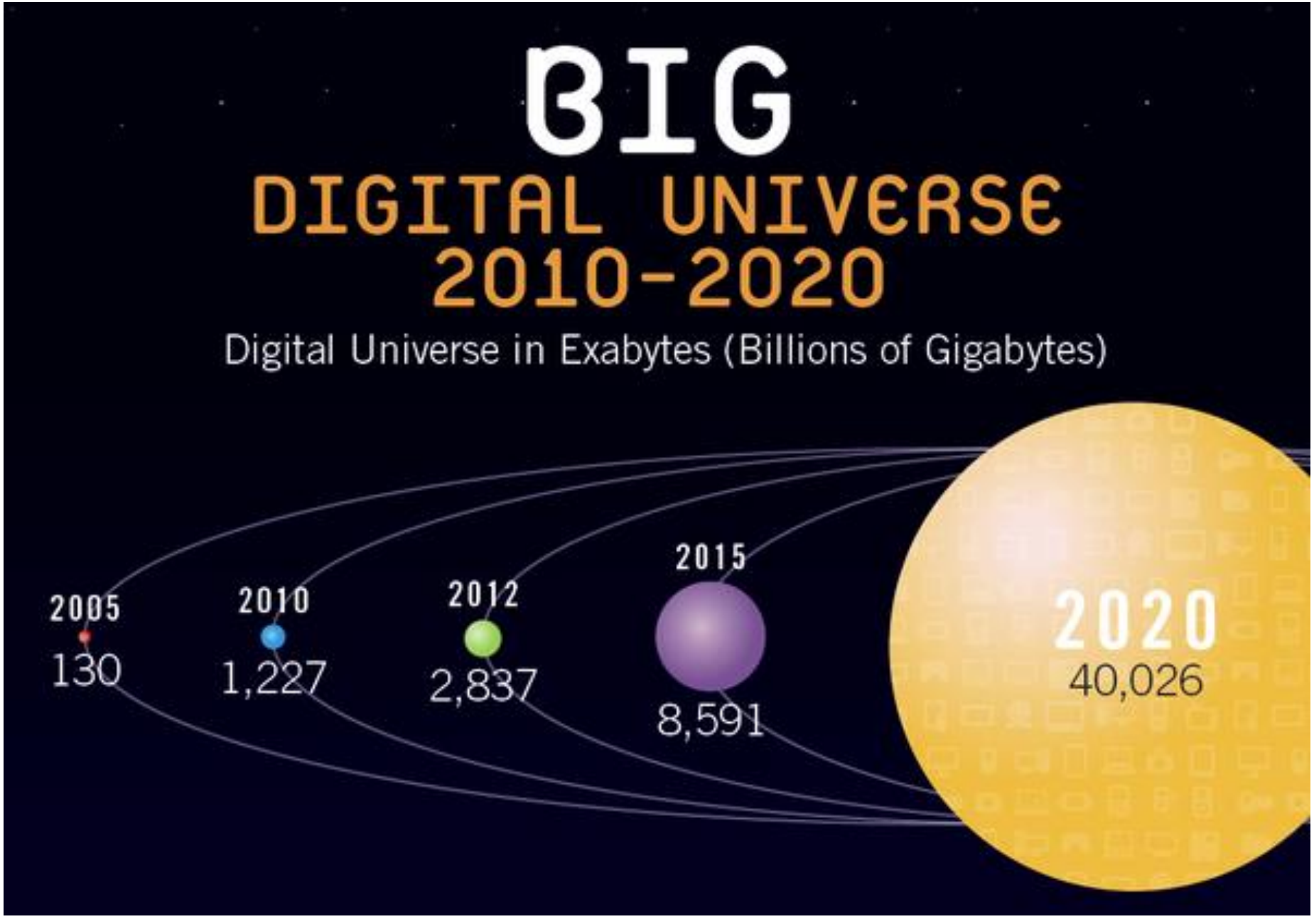
*EIGHTEEN months ago, Li & Fung, a firm that manages supply chains for retailers, saw 100 gigabytes of information flow through its network each day. Now the amount has increased tenfold. During 2009, American drone aircraft flying over Iraq and Afghanistan sent back around 24 years' worth of video footage. New models being deployed this year will produce ten times as many data streams as their predecessors, and those in 2011 will produce 30 times as many.*

Source: The Economist, February 25, 2010

# The Data Deluge

"*Everywhere you look, the quantity of information in the world is soaring. According to one estimate, mankind created 150 exabytes (billion gigabytes) of data in 2005. This year, it will create 1,200 exabytes. Merely keeping up with this flood, and storing the bits that might be useful, is difficult enough. Analysing it, to spot patterns and extract useful information, is harder still.*

Source: The Economist, February 25, 2010


BIG
DIGITAL UNIVERSE
2010-2020
Digital Universe in Exabytes (Billions of Gigabytes)
2005
130
2010
1,227
2012
2,837
2015
8,591
2020
40,026

# The Data Deluge

"

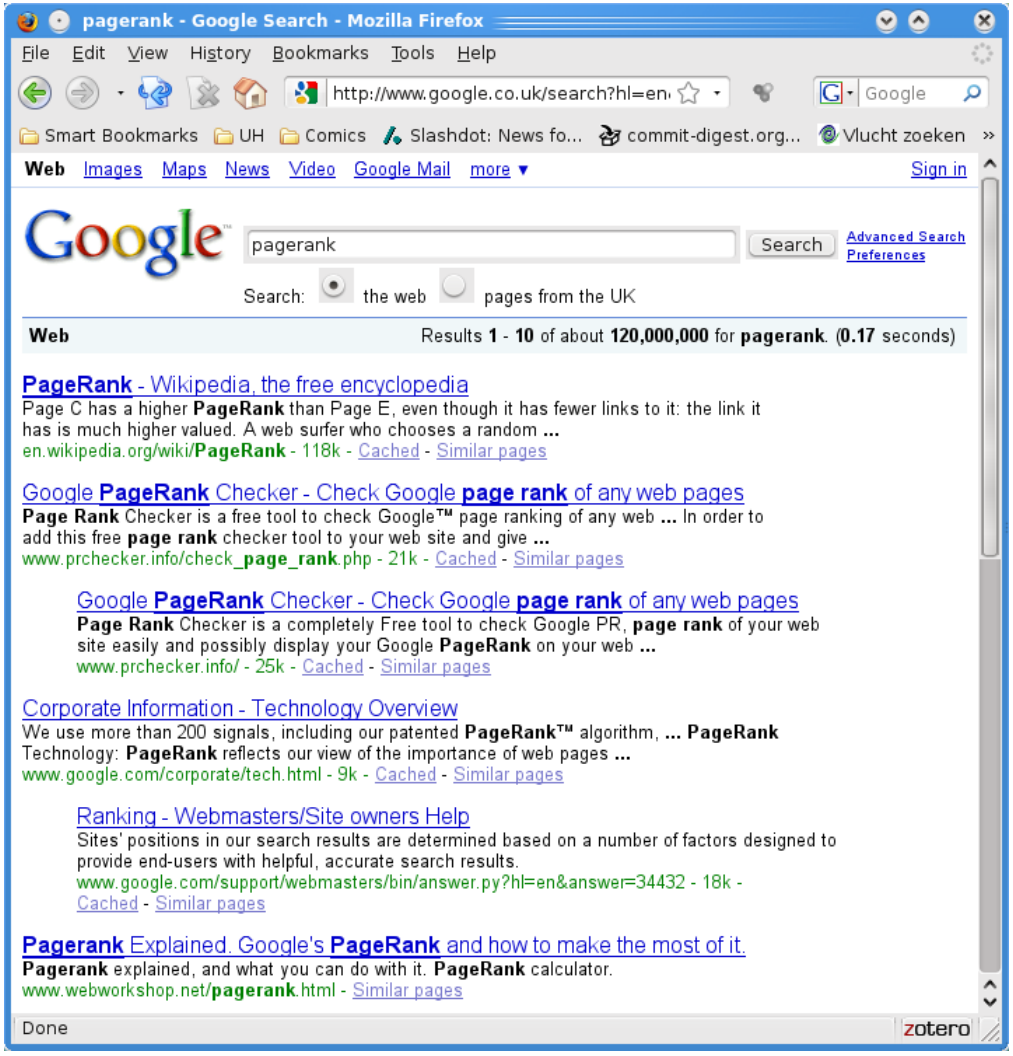
*By the 2030s, computer data storage may* ***surpass 1 yottabyte*** *($10^{24}$), the largest number with an official metric prefix.*

*The International Bureau of Weights and Measures (BIPM) in Paris* ***recommends new names****—ronna and quecca—as prefixes for $10^{27}$ and $10^{30}$.*

1 gigabyte = $10^9$ bytes
1 terabyte = $10^{12}$ bytes
1 petabyte = $10^{15}$ bytes
1 exabyte = $10^{18}$ bytes
1 zetabyte = $10^{21}$ bytes
1 yottabyte = $10^{24}$ bytes
**1 ronnabyte = $10^{27}$ bytes**
**1 queccabyte = $10^{30}$ bytes**

https://www.science.org/content/article/you-know-kilo-mega-and-giga-metric-system-ready-ronna-and-quecca

# Is this really problematic?




Assume you're google at the beginning of the 2000's and you want to index the web

# Is this really problematic?

"

*Although it is hard to accurately determine the size of the Web at any point in time, it is safe to say that it consists of hundreds of billions of individual documents and that it continues to grow. If we assume the Web to contain 100 billion documents, with an average document size of 4 kB (after compression), the Web is about 400 TB …*

*… The size of the resulting inverted (web search) index depends on the specific implementation, but it tends to be on the same order of magnitude as the original repository.*

Source: The Datacenter as a Computer
Luiz André Barroso (Google)
Jimmy Clidaras (Google)
Urs Hölzle (Google)

# BIG DATA: SOME DEFINITIONS (1/4)

> *Big Data is a term encompassing the use of techniques to capture, process, analyse and visualize potentially large datasets in a reasonable timeframe not accessible to standard IT technologies. By extension, the platform, tools and software used for this purpose are collectively called Big Data technologies.*

– Nessi[1]

[1] The European Technology Platform dedicated to Software, Services and Data

# BIG DATA: SOME DEFINITIONS (2/4)

> *Big data refers to datasets whose size is beyond the ability of typical database software tools to capture, store, manage, and analyze. This definition is intentionally subjective and incorporates a moving definition of how big a dataset needs to be in order to be considered big data ... We assume that, as technology advances over time, the size of datasets that qualify as big data will also increase.*
>
> – MCKinsey Global Institute [2]

[2] Big data: The next frontier for innovation, competition, and productivity, June 2011

# BIG DATA: SOME DEFINITIONS (3/4)

> *Big data are data sets that are so voluminous and complex that traditional data processing application software are inadequate to deal with them. Big data challenges include capturing data, data storage, data analysis, search, sharing, transfer, visualization, querying, updating and information privacy.*
>
> *Lately, the term big data tends to refer to the use of predictive analytics, user behavior analytics, or certain other advanced data analytics methods that extract value from data, and seldom to a particular size of data set.*

—Wikipedia

# BIG DATA: SOME DEFINITIONS (4/4)

> *Big data refers to extremely large datasets that cannot be easily managed, processed, or analyzed using traditional data processing methods. It is characterized by the three V's: volume, variety, and velocity, which describe the sheer amount of data, the diversity of data types, and the speed at which it is generated. Big data technologies enable businesses and researchers to gain insights and make data-driven decisions from these vast, dynamic datasets.*
>
> —ChatGPT (02/2025)

# THE BIG THREE


BIG DATA

DATA ANALYTICS

DATA SCIENCE

# THE BIG THREE

DATA ANALYTICS

# DATA ANALYTICS: A DEFINITION

> *Data analysis, also known as analysis of data or data analytics, is a process of inspecting, cleansing, transforming, and modeling data with the goal of discovering useful information, suggesting conclusions, and supporting decision-making.*

—Wikipedia


Large amounts of
RAW DATA
EXTRACT
TRANSFORM
LOAD
DATA WAREHOUSE
FORECASTING
$f(x) = \frac{1}{\sigma 2\pi} e^{-(x-\mu)^2/(2\sigma^2)}$
Predicting FUTURE customer behaviour
bell curve
BUSINESS INTELLIGENCE
Useful
Improve STRATEGY
DATA MINING
Searching for hidden patterns ...
012012
012012
012012
Decision-making support
This way
That way
Graphical ANALYSIS
QUERY & REPORTING
Spot
NEW
OPPORTUNITIES

# THE BIG THREE


BIG DATA

DATA ANALYTICS

DATA SCIENCE

# THE BIG THREE

DATA SCIENCE

# DATA SCIENCE: SOME DEFINITIONS (1/3)

> *Data science is the study of the generalizable extraction of knowledge from data.*

—Vasant Dhar[3]

From the article:

- The term science implies knowledge gained through systematic study.
- A data scientist requires an integrated skill set spanning mathematics, machine learning, artificial intelligence, statistics, databases, and optimization, along with a deep understanding of the craft of problem formulation to engineer effective solutions.

[3] Data Science and Predictions. Communications of the ACM 56(12), 2014.

# DATA SCIENCE: SOME DEFINITIONS (2/3)

> *Data science, also known as data-driven science, is an interdisciplinary field about scientific methods, processes, and systems to extract knowledge or insights from data in various forms, either structured or unstructured.*
>
> *Data science is a "concept to unify statistics, data analysis and their related methods" in order to "understand and analyze actual phenomena" with data. It employs techniques and theories drawn from many fields within the broad areas of mathematics, statistics, information science, and computer science.*

—Wikipedia

# DATA SCIENCE: SOME DEFINITIONS (3/3)

> *Data science is the field that combines statistical analysis, machine learning, and data processing techniques to extract valuable insights from large and complex datasets. It involves collecting, cleaning, analysing, and interpreting data to inform decision-making and solve problems across various industries. Data scientists use programming, mathematics, and domain expertise to uncover patterns and trends in data.*

—ChatGPT (02/2025)

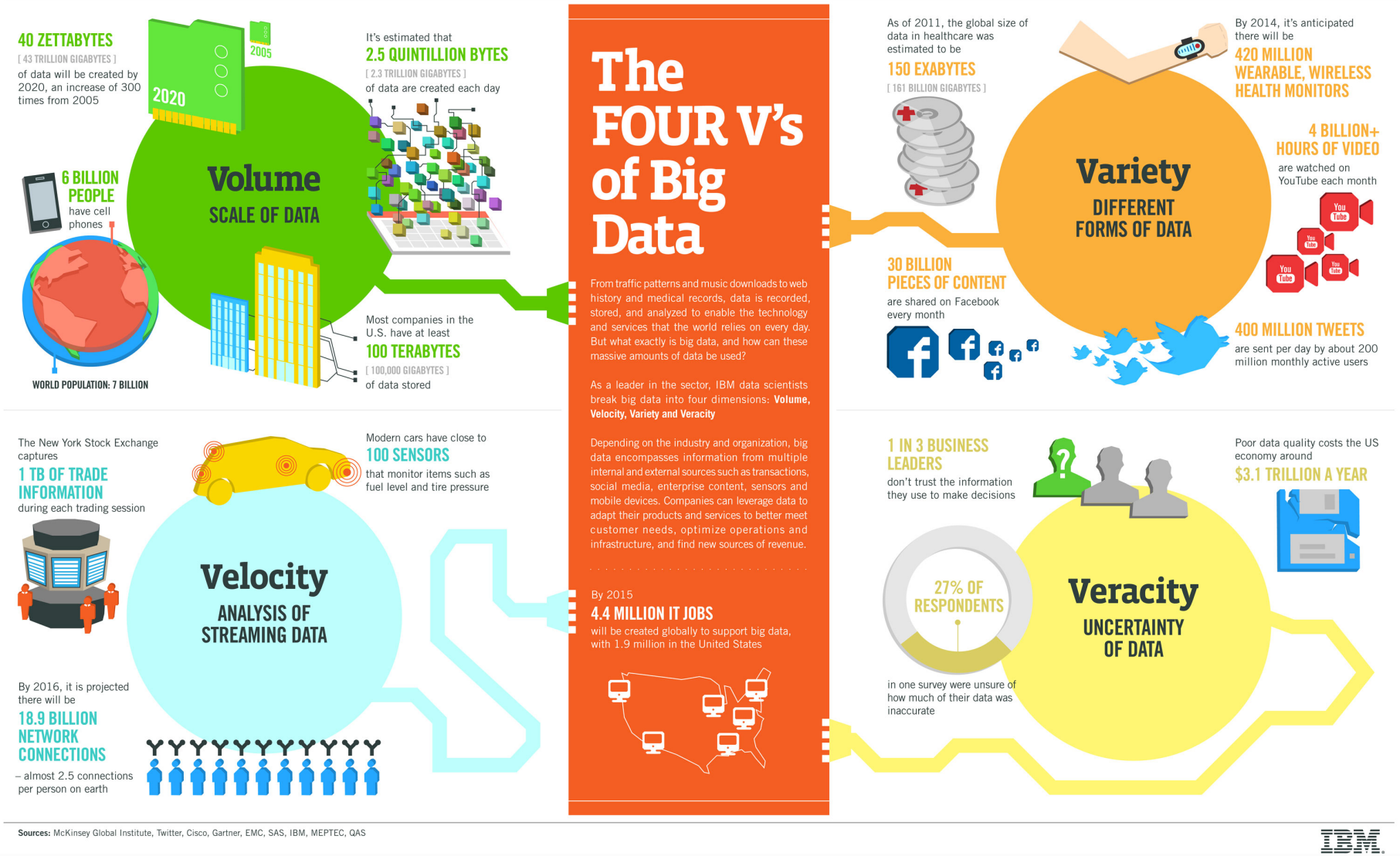
# CONCLUSION


Recall the course objective:

Introduce the fundamental notions, principles, and research results concerning modern, scalable, and fault-tolerant ways for managing and analyzing massive amounts of data using parallel and distributed systems.

The course is hence concerned with Big Data and Big Data Analytics, and thus relevant for Data Science.

# THE CHALLENGES OF BIG DATA




Source: `http://www.ibmbigdatahub.com/sites/default/files/infographic_file/`

# HOW IS BIG DATA USED?

# HOW IS BIG DATA USED ?


Slides taken from `https://www.slideshare.net/Dell/big-data-use-cases-3601989`

# BIG DATA USE CASES:

1. OPTIMIZE FUNNEL CONVERSION

2. BEHAVIORAL ANALYTICS

3. CUSTOMER SEGMENTATION

4. PREDICTIVE SUPPORT

5. MARKET BASKET ANALYSIS AND PRICING OPTIMIZATION

6. PREDICT SECURITY THREATS

7. FRAUD DETECTION

8. INDUSTRY SPECIFIC

# 1. OPTIMIZE FUNNEL CONVERSION

Big data analytics allows companies to track leads through the entire sales conversion process, from a click on an adword ad to the final transaction, in order to uncover insights on how the conversion process can be improved.


## COMPANY
T-Mobile

## INDUSTRY
Communication

## EMPLOYEES
38,000

## TYPE
Optimize Funnel Conversion

## PURPOSE:

T-mobile uses multiple indicators, such as billing and sentiment analysis, in order to identify customers that can be upgraded to higher quality products, as well as to identify those with a high lifetime customer-value, so its team can focus on retaining those customers.


## COMPANY
Celcom Axiata Berhad

## INDUSTRY
Communication

## EMPLOYEES
Enterprise

## TYPE
Optimize Funnel Conversion

## PURPOSE:

Celcom Axiata Berhad adopted a big data solution in order to improve customer retention and boost its market share by improving the marketing campaign process. The company used real-time data to create a personalized campaign for each customer based on which products or offers the customer would most want or need.

# 2. BEHAVIORAL ANALYTICS

With access to data on consumer behavior, companies can learn what prompts a customer to stick around longer, as well as learn more about their customer's characteristics and purchasing habits in order to improve marketing efforts and boost profits.


## COMPANY
Mastercard

## INDUSTRY
Finance

## EMPLOYEES
67,000

## TYPE
Behavioral Analytics

## PURPOSE:

With 1.8 billion customers, MasterCard is in the unique position of being able to analyze the behavior of customers in not only their own stores, but also thousands of other retailers. The company teamed up with Mu Sigma to collect and analyze data on shoppers' behavior, and provide the insights it finds to other retailers in benchmarking reports.


## COMPANY
Time Warner Cable

## INDUSTRY
Entertainment

## EMPLOYEES
34,000

## TYPE
Behavioral Analytics & Customer Segmentation

## PURPOSE:

With services like Hulu and Netflix competing for viewers' attention, Time Warner collects data on how frequently customers tune in, the effect of bandwidth on consumer behavior, customer engagement and peak usage times in order to improve their service and increase profits. The company also segments its customers for advertisers by correlating viewing habits with public data—such as voter registration information—in order to launch highly targeted campaigns to specific locations or demographics.

tweet this


## COMPANY
Nestlé

## INDUSTRY
Food & Beverage

## EMPLOYEES
>330,000

## TYPE
Behavioral Analytics

## PURPOSE:

Customer complaints and PR crises have become more difficult to handle thanks to social media. To better keep track of customer sentiment and what is being said about the company online, Nestle created a 24/7 monitoring centre to listen to all of the conversations about the company and its products on social media. The company will actively engage with those that post about them online in order to mitigate damage and build customer loyalty.

tweet this

# 3. CUSTOMER SEGMENTATION

By accessing data about the consumer from multiple sources, such as social media data and transaction history, companies can better segment and target their customers and start to make personalized offers to those customers.


## COMPANY
Heineken

## INDUSTRY
Food & Beverage

## EMPLOYEES
64,252

## TYPE
Customer Segmentation

## PURPOSE:

Thanks to its partnerships with Google and Facebook, Heineken has access to vasts amounts of data about its customers that it uses to create real-time, personalized marketing messages. One project provides real-time content to fans who happen to be watching a sponsored event.

tweet this


## COMPANY
Spotify

## INDUSTRY
Entertainment

## EMPLOYEES
5,000

## TYPE
Customer Segmentation & Behavioral Analytics

## PURPOSE:

Spotify uses data from user profiles and users' playlists, and historical data on music played to provide recommendations for each user. By combining data from millions of users, Spotify is able to make recommendations even if a particular user doesn't have an extensive history with the site.

# 4. PREDICTIVE SUPPORT

Through sensors and other machine-generated data, companies can identify when a malfunction is likely to occur. The company can then preemptively order parts and make repairs in order to avoid downtime and lost profits.


## COMPANY
Southwest Airlines

## INDUSTRY
Travel

## EMPLOYEES
>45,000

## TYPE
Predictive Support

## PURPOSE:

Southwest analyses sensor data on their planes in order to identify patterns that indicate a potential malfunction or safety issue. This allows the airline to address potential problems and make necessary repairs without interrupting flights or putting passengers in danger.

tweet this


## COMPANY
Engine Yard

## INDUSTRY
Cloud Storage

## EMPLOYEES
130

## TYPE
Predictive Support

## PURPOSE:

Engine yard provides big data analytics to its users, so they can monitor the performance of applications in real time, pinpoint problems with the infrastructure and optimize the platform to correct performance issues.

tweet this

# 5. MARKET BASKET ANALYSIS & PRICING OPTIMIZATION

By quickly pulling data together from multiple sources, retailers can better optimize their product selection and pricing, as well as decide where to target ads.


## COMPANY
Coca-Cola Co.

## INDUSTRY
Food

## EMPLOYEES
146,200

## TYPE
Market Basket Analysis

## PURPOSE:
Coca-Cola uses an algorithm to ensure that its orange juice has a consistent taste throughout the year. The algorithm incorporates satellite imagery, crop yields, consumer preferences and details about the flavours that make up a particular fruit in order to determine how the juice should be blended.

# 6. PREDICT SECURITY THREATS

Big data analytics can track trends in security breaches and allow companies to proactively go after threats before they strike.


## COMPANY
Rabobank

## INDUSTRY
Finance

## EMPLOYEES
27,000

## TYPE
Predict Security Threats

## PURPOSE:

Rabobank analysed criminal activities at ATMs to determine factors that increased the risk of becoming victimized. It discovered that proximity to highways, weather conditions and the season all affect the risk of a security threat.

tweet this

amazon.com

## COMPANY
Amazon

## INDUSTRY
Online Retail

## EMPLOYEES
110,000

## TYPE
Predict Security Threats

## PURPOSE:
With more than 1.5 billion items in its catalog, Amazon has a lot of product to keep track of and protect. It uses its cloud system, S3, to predict which items are most likely to be stolen, so it can better secure its warehouses.

tweet this

# 7. FRAUD DETECTION

Financial firms use big data to help them identify sophisticated fraud schemes by combining multiple points of data.

ZIONS BANK®

**COMPANY**
Zion's Bank

**INDUSTRY**
Finance

**EMPLOYEES**
2,700

**TYPE**
Fraud Detection

## PURPOSE:

Zions Bank uses data analytics to detect anomalies across channels that indicate potential fraud. The fraud team receives data from 140 sources—some in real-time—to monitor activity, such as if a customer makes a mobile banking transaction at the same time as a branch transaction.

Discovery Health

## COMPANY
Discovery Health

## INDUSTRY
Insurance

## EMPLOYEES
5,000

## TYPE
Fraud Detection

## PURPOSE:
Discovery Health uses big data analytics to identify fraudulent claims and possible fraudulent prescriptions. For example, it can identify if a healthcare provider is charging for a more expensive procedure than was actually performed.

tweet this

# 8. INDUSTRY SPECIFIC

Virtually every industry has invested in big data to help solve specific challenges those industries face. Healthcare, for example, uses big data to improve patient outcomes, and agriculture uses data to boost crop yields.

KAYAK

## COMPANY
Kayak

## INDUSTRY
Travel

## EMPLOYEES
101

## TYPE
Industry Specific

## PURPOSE:

Kayak uses big data analytics to create a predictive model that tells users if the price for a particular flight will go up or down within the next week. The system uses one billion search queries to find the cheapest flights, as well as popular destinations and the busiest airports. The algorithm is constantly improved by tracking the flights to see if its predictions are correct.

Aurora Health Care

## COMPANY
Aurora Health Care

## INDUSTRY
Health Care

## EMPLOYEES
30,000

## TYPE
Industry Specific

## PURPOSE:

Aurora collects internal as well as national data in order to create a benchmark for healthcare quality. It also analyzes data on groups of patients with similar medical conditions, to reveal trends in the diseases and to identify the right candidates for medical research. Finally, the real-time data analysis allows Aurora to predict and improve patient outcomes ,and so far has reduced readmissions by 10 percent.

tweet this


## COMPANY
Shell

## INDUSTRY
Oil

## EMPLOYEES
87,000

## TYPE
Industry Specific

## PURPOSE:

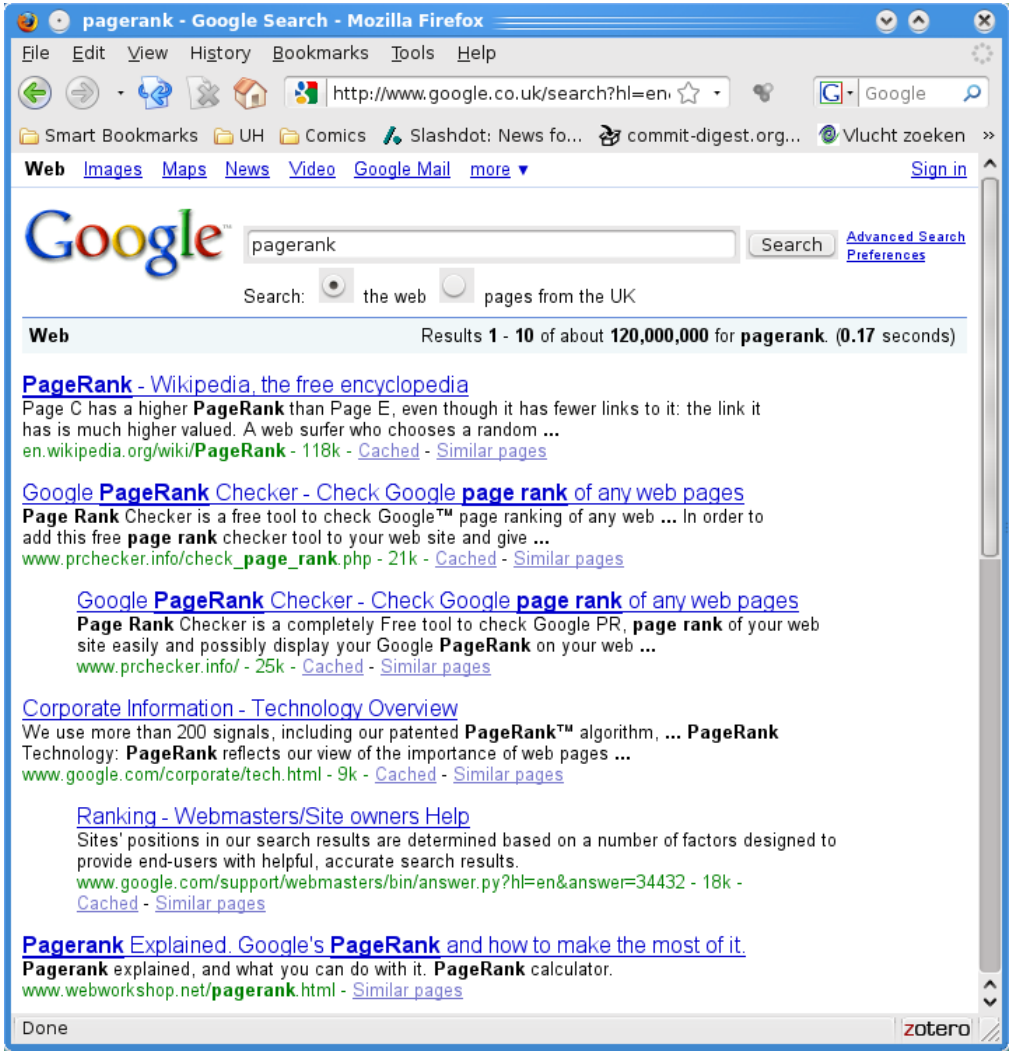
Shell uses sensor data to map its oil and gas wells in order to increase output and boost the efficiency of its operations. The data received from the sensors is analyzed by artificial intelligence and rendered in 3D and 4D maps.

# TAKE-AWAY

In all of these examples, data is stored, processed, and analyzed, to turn it into value.

# BIG DATACENTERS

# How do you process Big Data ?

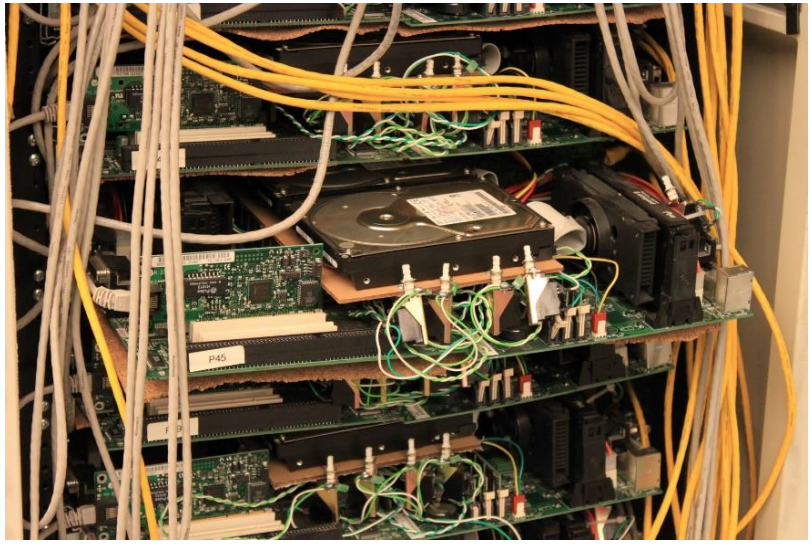


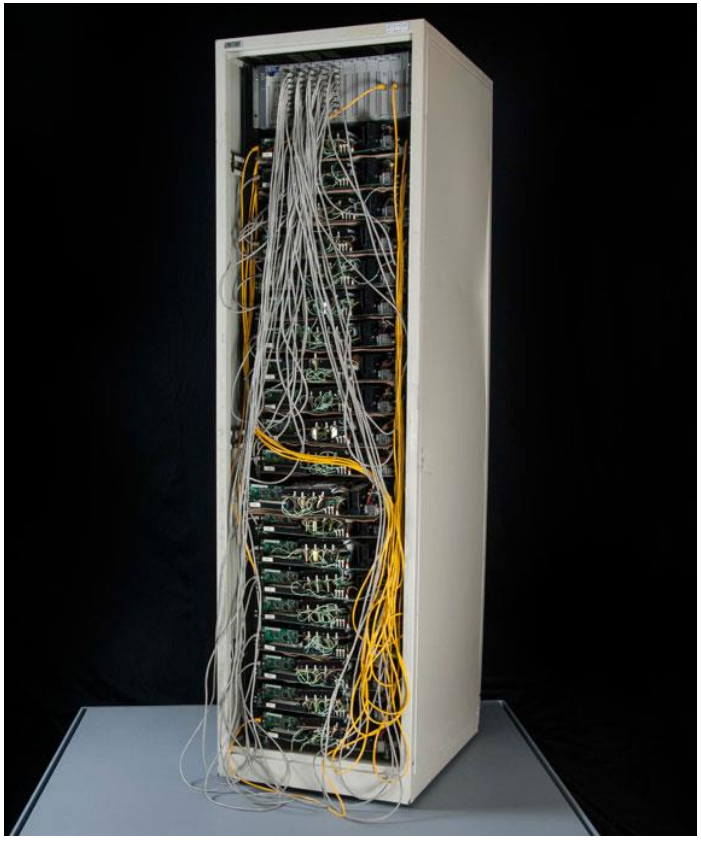

Assume you're google at the beginning of the 2000's and you want to index the web

# Execution environment - 1997


- Just a bunch of computers.
- The web index fit on a single computer
- Redundancy to ensure availability

# Execution environment - 1999

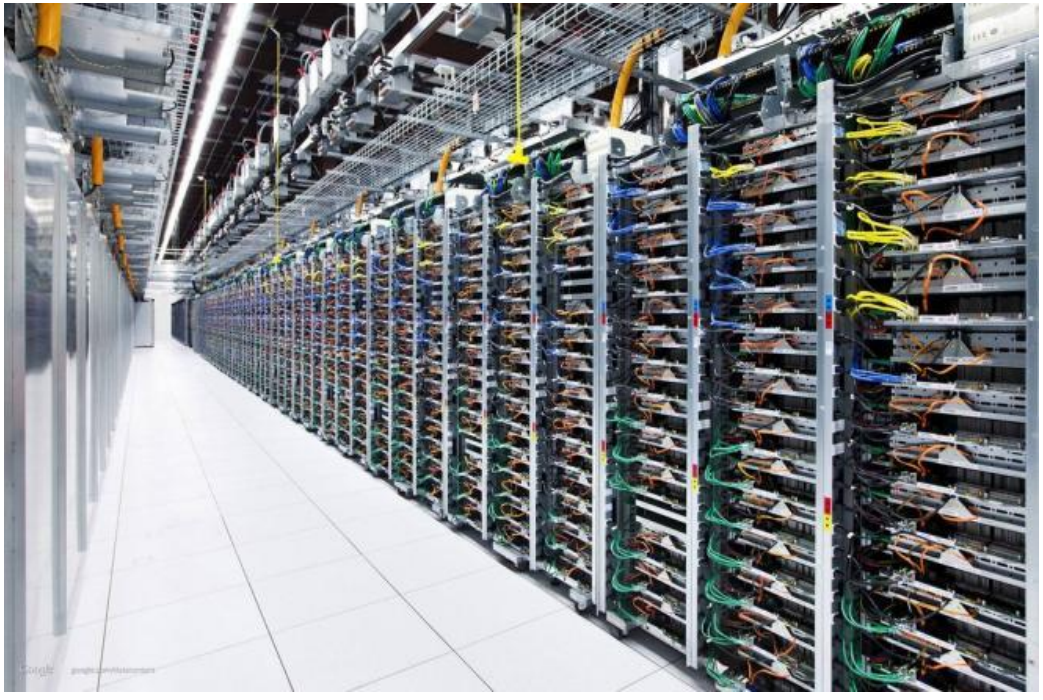




The Google “Corckboard” rack

- Compute servers consist of multiple CPUs (possibly with multiple cores per CPU), and attached hard disks,
- Servers are collected in racks. High-speed Ethernet connections (at least 1Gbps) connect servers in a rack together

# Execution environment - currently



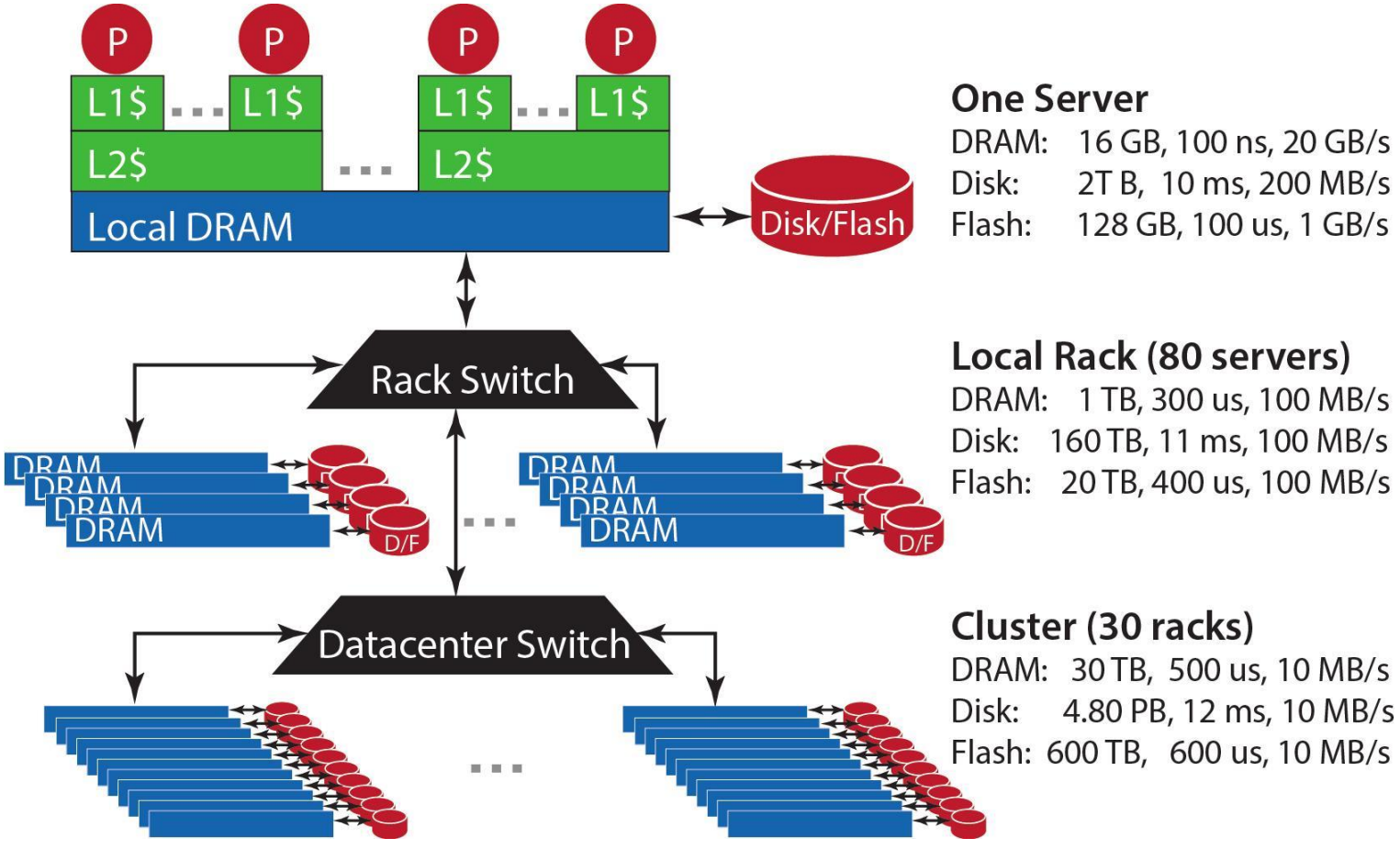
An image of the google datacenter in Mons (Belgium)

- Racks are connected together to central network switches using multi-Gbps redundant links.
- Access of data from other racks is slower than data on same computer/same rack
- Many racks together form a data center

# Discussion

- Each compute node is kept simple by design:
  - Mid-range computers are much cheaper than powerful high-range computers …
  - … and consume less energy
  - … but google has lots of them!

# Characteristics



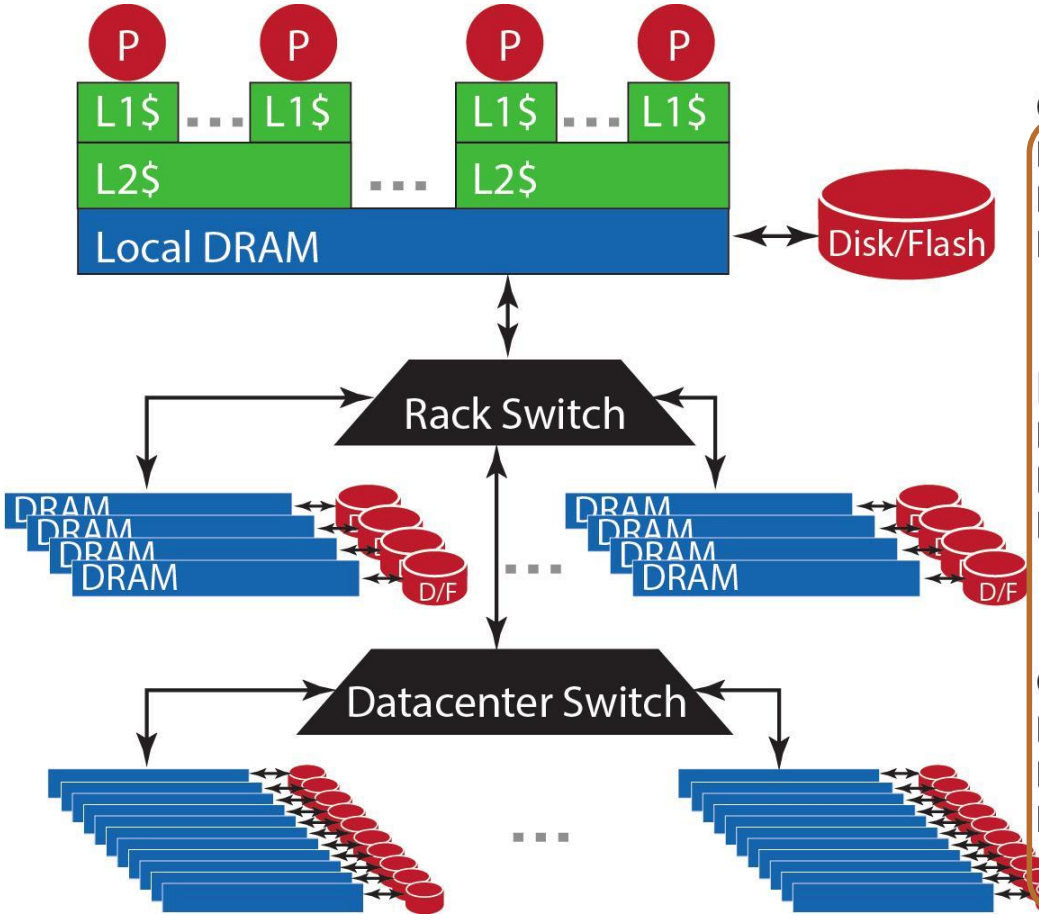

**One Server**
DRAM: 16 GB, 100 ns, 20 GB/s
Disk: 2T B, 10 ms, 200 MB/s
Flash: 128 GB, 100 us, 1 GB/s

**Local Rack (80 servers)**
DRAM: 1 TB, 300 us, 100 MB/s
Disk: 160 TB, 11 ms, 100 MB/s
Flash: 20 TB, 400 us, 100 MB/s

**Cluster (30 racks)**
DRAM: 30 TB, 500 us, 10 MB/s
Disk: 4.80 PB, 12 ms, 10 MB/s
Flash: 600 TB, 600 us, 10 MB/s

Figure Source: The Datacenter as a Computer

# Characteristics




**One Server**
DRAM: 16 GB, 100 ns, 20 GB/s
Disk: 2T B, 10 ms, 200 MB/s
Flash: 128 GB, 100 us, 1 GB/s

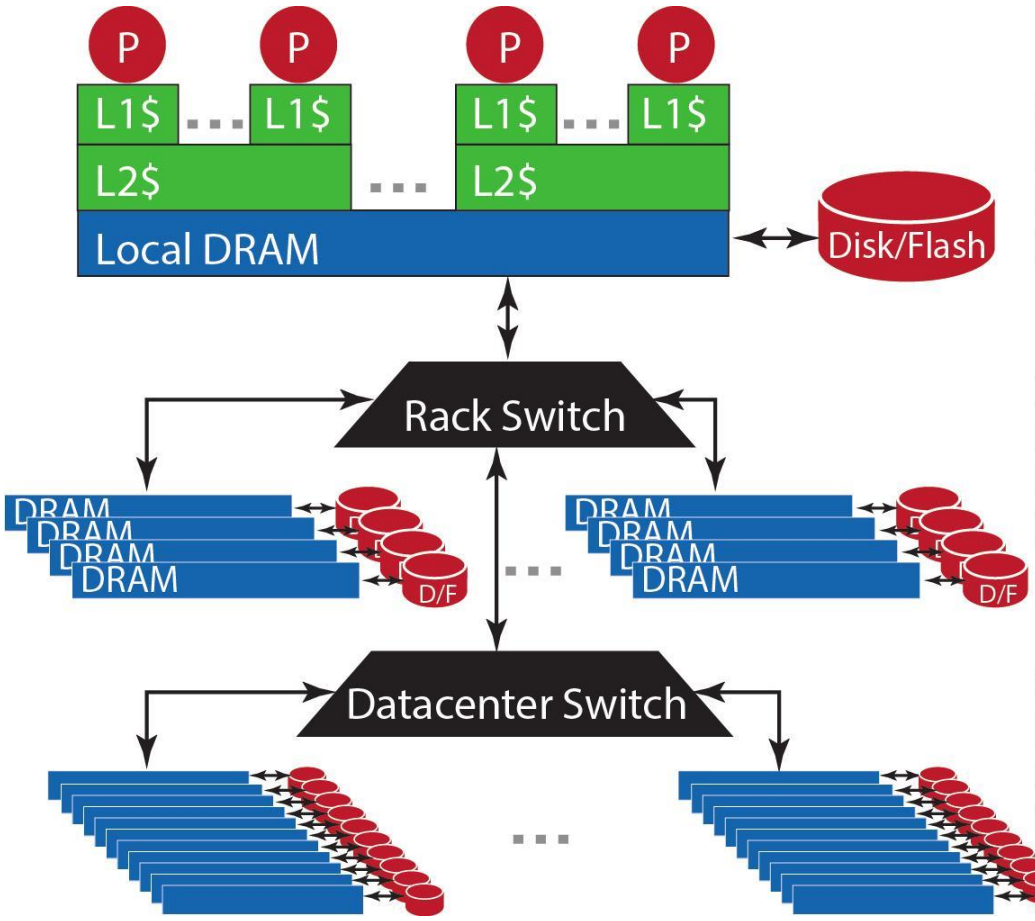
**Local Rack (80 servers)**
DRAM: 1 TB, 300 us, 100 MB/s
Disk: 160 TB, 11 ms, 100 MB/s
Flash: 20 TB, 400 us, 100 MB/s

**Cluster (30 racks)**
DRAM: 30 TB, 500 us, 10 MB/s
Disk: 4.80 PB, 12 ms, 10 MB/s
Flash: 600 TB, 600 us, 10 MB/s

Figure Source: The Datacenter as a Computer

Capacity: The amount of data we can store per server/rack/datacenter

# Characteristics




**One Server**
DRAM: 16 GB, 100 ns, 20 GB/s
Disk: 2T B, 10 ms, 200 MB/s
Flash: 128 GB, 100 us, 1 GB/s

**Local Rack (80 servers)**
DRAM: 1 TB, 300 us, 100 MB/s
Disk: 160 TB, 11 ms, 100 MB/s
Flash: 20 TB, 400 us, 100 MB/s

**Cluster (30 racks)**
DRAM: 30 TB, 500 us, 10 MB/s
Disk: 4.80 PB, 12 ms, 10 MB/s
Flash: 600 TB, 600 us, 10 MB/s

Figure Source: The Datacenter as a Computer

Latency: The time it takes to fetch a data item, when asked on local machine/another server on the same rack/another server on a different rack

# Characteristics

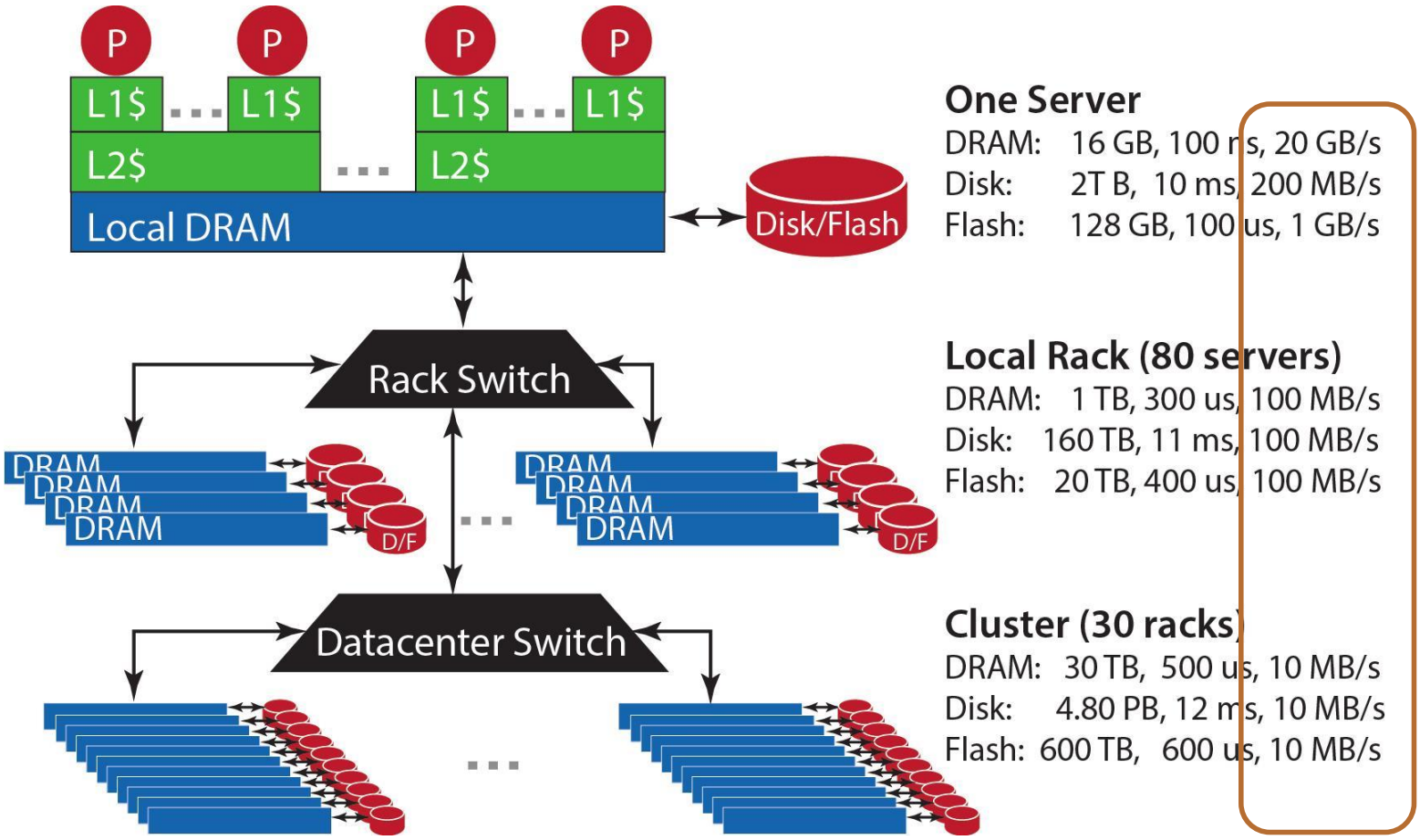



Figure Source: The Datacenter as a Computer

Bandwidth: the speed at which data can be transferred to the same machine/another server on the same rack/another server on a different rack

# Characteristics

**Conclusion:**

- Huge storage capacity
- Latency between racks
  = 1/10 latency on rack level
  ≈ 1/10 latency on server level
- Bandwidth between racks
  = 1/10 bandwidth on rack level
  = ½ to 1/10 bandwidth on server level

**One Server**
DRAM: 16 GB, 100 ns, 20 GB/s
Disk: 2T B, 10 ms, 200 MB/s
Flash: 128 GB, 100 us, 1 GB/s

**Local Rack (80 servers)**
DRAM: 1 TB, 300 us, 100 MB/s
Disk: 160 TB, 11 ms, 100 MB/s
Flash: 20 TB, 400 us, 100 MB/s

**Cluster (30 racks)**
DRAM: 30 TB, 500 us, 10 MB/s
Disk: 4.80 PB, 12 ms, 10 MB/s
Flash: 600 TB, 600 us, 10 MB/s

Figure Source: The Datacenter as a Computer

# What do we gain? Parallelism!

- Let us consider the maximal aggregate bandwidth: the speed by which we can analyze data in parallel assuming ideal data distribution over servers & disks

- "Embarrassingly parallel" example: count the number of times the word "Belgium" appears in documents on the Web.

- Each server has multiple CPUs and can read from multiple disks in parallel. As such, each server can analyze many documents in parallel.
- At the end, sum the per-server counters (which can be done very fast)

# What do we gain? Parallelism!

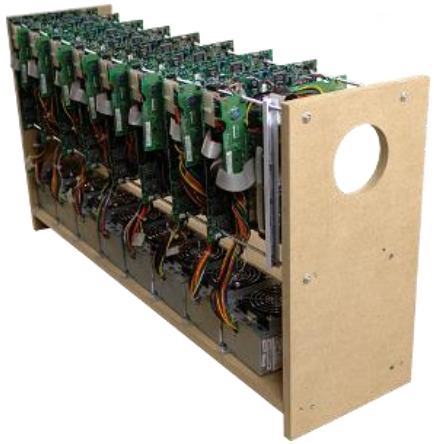
- Let us consider the maximal aggregate bandwidth: the speed by which we can analyze data in parallel assuming ideal data distribution over servers & disks

| Component | | Max Aggr Bandwidth |
|---|---|---|
| 1 Hard Disk | | 100 MB/sec (≈ 1 Gbps) |
| Server | = 12 Hard Disks | 1.2 GB/sec (≈ 12 Gbps) |
| Rack | = 80 servers | 96 GB/sec (≈ 768 Gbps) |
| Cluster/datacenter | = 30 racks | 2.88 TB/sec (≈ 23 Tbps) |

- Scanning 400TB hence takes 138 secs ≈ 2,3 minutes
- Scanning 400TB *sequentially* at 100 MB/sec takes ≈ 46,29 days

# The challenges (1/2)

- Scalable software development: allow growth without requiring re-architecting algorithm/applications

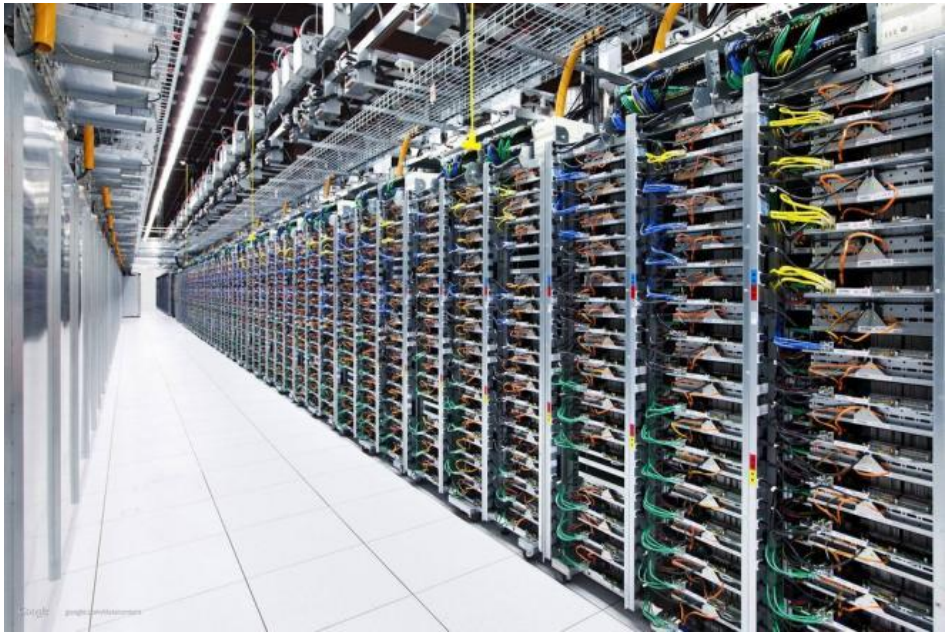


Auto scale

# The challenges (2/2)

- Fault-tolerance: if you have 1000's of machines, failures happen every day.

- Fault-tolerance is typically addressed through redundancy and/or re-execution



# FAULT-TOLERANCE: MOTIVATIONAL EXAMPLE

- Mean Time To Failure (MTTF) of typical hard-disk:

$$1,000,000 \text{ hours} = 114 \text{ years}$$

# FAULT-TOLERANCE: MOTIVATIONAL EXAMPLE

- Mean Time To Failure (MTTF) of typical hard-disk:

$$1,000,000 \text{ hours} = 114 \text{ years}$$

- Annual Failure Rate (AFR) for a given MTTF

$$\text{AFR} = \frac{\#\text{hours in 1 year}}{\text{MTTF}} = \frac{365 \times 24}{10^6} = 0.876\%$$

# FAULT-TOLERANCE: MOTIVATIONAL EXAMPLE

- Mean Time To Failure (MTTF) of typical hard-disk:

$$1,000,000 \text{ hours} = 114 \text{ years}$$

- Annual Failure Rate (AFR) for a given MTTF

$$\text{AFR} = \frac{\#\text{hours in 1 year}}{\text{MTTF}} = \frac{365 \times 24}{10^6} = 0.876\%$$

- Expected number of disk failures per year in a datacenter with 100,000 disks:

$$\#\text{disks} \times \text{AFR} = 10^5 \times 0.876\% = 876 \frac{\text{disks}}{\text{year}} = 2 \frac{\text{disks}}{\text{day}}$$

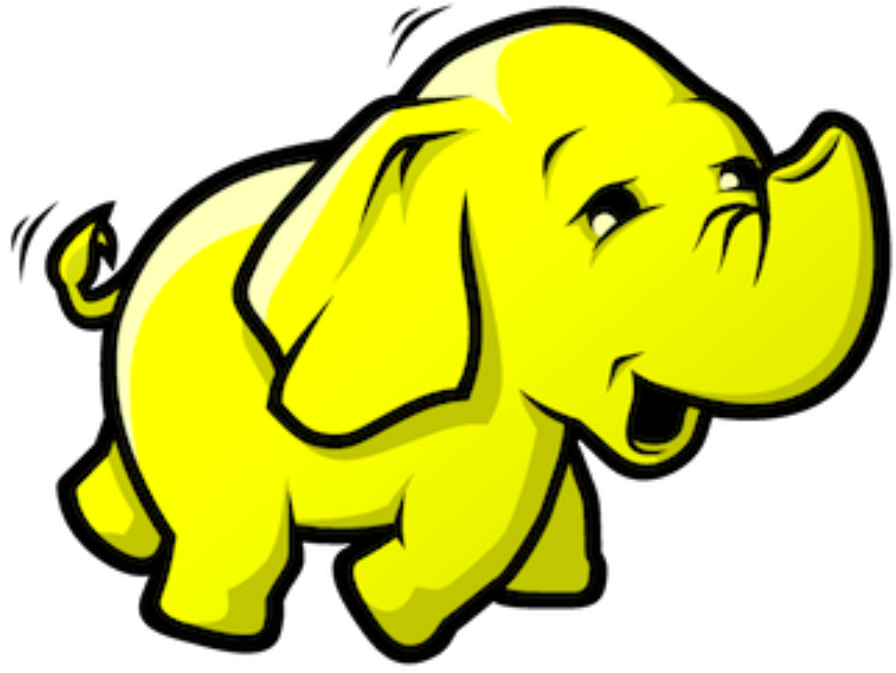
# Google's solutions

- New programming models and frameworks for distributed and scalable data analysis

| Name | Purpose | Open Source Impl |
|---|---|---|
| Google File System | A distributed file system for scalable storage and high-throughput retrieval |  Apache Hadoop |
| Map Reduce | A programming model + execution environment for general-purpose distributed batch processing | • HDFS<br>• M/R |
| BigTable | A NoSQL Database | Apache HBase |
| Dremel, F1 | A query language for interactive SQL-like analysis of structured datasets | Apache Spark/Drill |
| … | *Lots of research ongoing!* | …. |

High-level design described in a series of papers; no implementation available

# HDFS



# HDFS:
# THE HADOOP DISTRIBUTED FILE SYSTEM

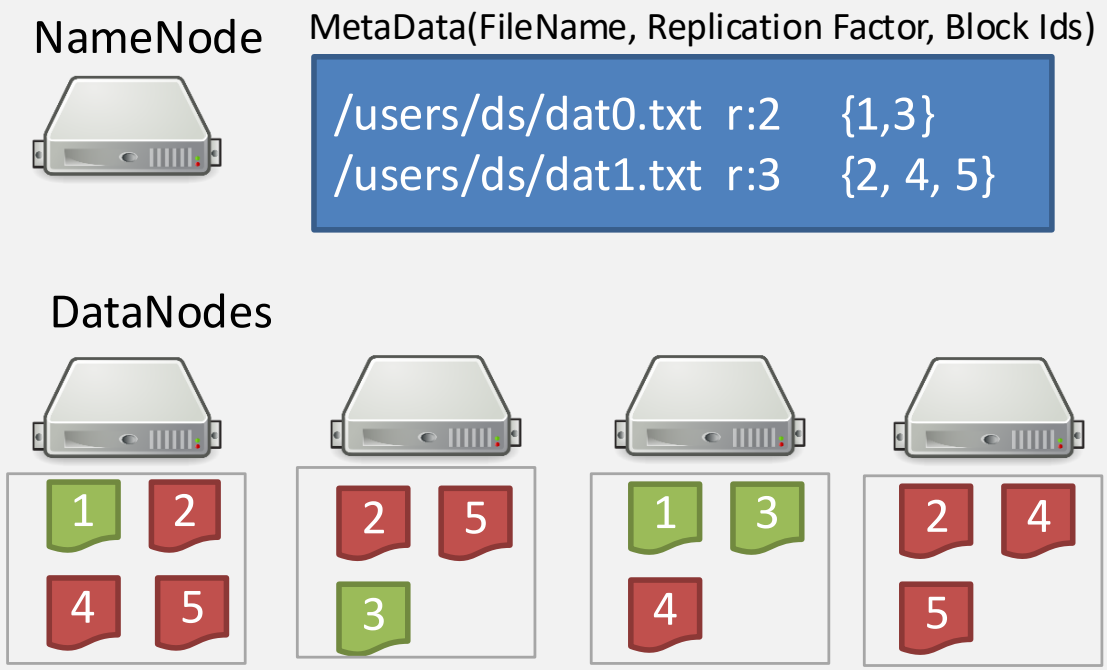
# HDFS Architecture

- HDFS has a master/slave architecture
- Master = NameNode (NN) manages the file system and regulates access to files by clients.
- Slaves = DataNodes (DN), usually one per server in the cluster, manage storage attached to the server that they run on.

- Files are transparently broken down into blocks (default 128 MB).
- Blocks are replicated across datanodes. Replication is rack-aware.

dat0.txt 1 3

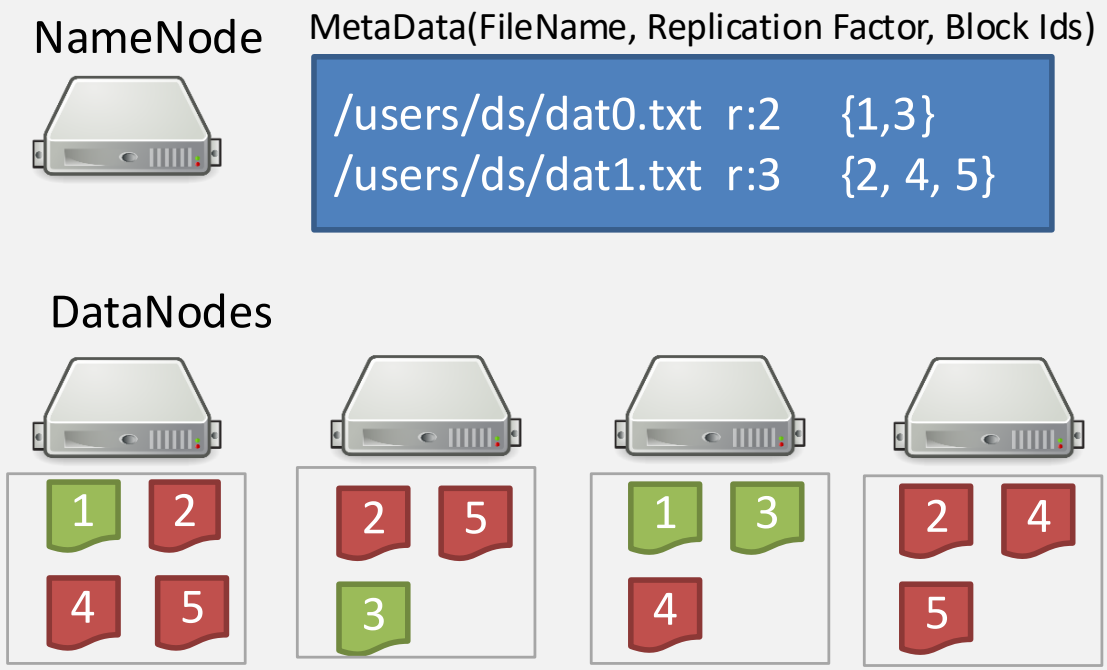
dat1.txt 2 4 5



# HDFS Architecture

- HDFS has a master/slave architecture
- Master = NameNode (NN) manages the file system and regulates access to files by clients.
- Slaves = DataNodes (DN), usually one per server in the cluster, manage storage attached to the server that they run on.

- Optimized for:
  - Large files
  - Read throughput
  - Appending writes (files are append-only)
- Replication ensures:
  - Durability
  - Availability
  - Throughput



# HDFS Common Pitfalls

- HDFS = optimized for large files.
  - A single HDFS block is 128 MB  per default.
  - It does not make sense to store files smaller than 128 MB in HDFS!

- Files are immutable, optimized for sequential access.
  - New files that are created can only be appended to (no random-access)
  - Once a file is created, it is immutable. Changing a file requires re-creating it.

# HDFS Implementation

> *The NameNode and DataNode are pieces of software designed to run on commodity machines. These machines typically run a GNU/Linux operating system (OS). HDFS is built using the Java language; any machine that supports Java can run the NameNode or the DataNode software. Usage of the highly portable Java language means that HDFS can be deployed on a wide range of machines.*

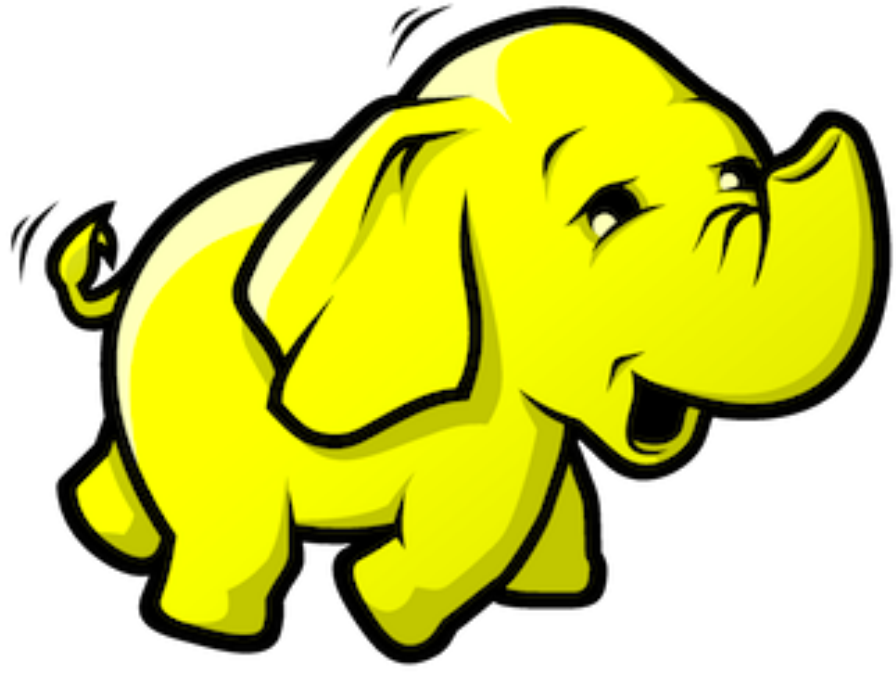
Source: Hadoop documentation

- This implies that clients only need to install a JAR file to access the HDFS

# Typical HDFS commands

```
bin/hadoop fs -ls

bin/hadoop fs -mkdir

bin/hadoop fs -copyFromLocal

bin/hadoop fs -copyToLocal

bin/hadoop fs -moveToLocal

bin/hadoop fs -rm
```

# MAP/REDUCE



# MAP/REDUCE: SIMPLIFIED DATA PROCESSING ON LARGE CLUSTERS

# MapReduce: Simplified Data Processing on Large Clusters

by Jeffrey Dean and Sanjay Ghemawat

## Abstract

MapReduce is a programming model and an associated implementation for processing and generating large datasets that is amenable to a broad variety of real-world tasks. Users specify the computation in terms of a *map* and a *reduce* function, and the underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, handles machine failures, and schedules inter-machine communication to make efficient use of the network and disks. Programmers find the system easy to use: more than ten thousand distinct MapReduce programs have been implemented internally at Google over the past four years, and an average of one hundred thousand MapReduce jobs are executed on Google's clusters every day, processing a total of more than twenty petabytes of data per day.

## 1 Introduction

Prior to our development of MapReduce, the authors and many others at Google implemented hundreds of special-purpose computations that process large amounts of raw data, such as crawled documents, Web request logs, etc., to compute various kinds of derived data, such as inverted indices, various representations of the graph structure of Web documents, summaries of the number of pages crawled per host, and the set of most frequent queries in a given day. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault tolerance, data distribution and load balancing in a library. Our abstraction is inspired by the *map* and *reduce* primitives present in Lisp and many other functional languages. We realized that most of our computations involved applying a map operation to each logical record in our input in order to compute a set of intermediate key/value pairs, and then applying a reduce operation to all the values that shared the same key in order to combine the derived data appropriately. Our use of a functional model with user-specified map and reduce operations allows us to parallelize large computations easily and to use reexecution as the primary mechanism for fault tolerance.

The major contributions of this work are a simple and powerful interface that enables automatic parallelization and distribution of large-scale computations, combined with an implementation of this interface that achieves high performance on large clusters of commodity PCs. The programming model can also be used to parallelize computations across multiple cores of the same machine.

Section 2 describes the basic programming model and gives several examples. In Section 3, we describe an implementation of the MapReduce interface tailored towards our cluster-based computing environment. Section 4 describes several refinements of the programming model that we have found useful. Section 5 has performance measurements of our implementation for a variety of tasks. In Section 6, we explore the use of MapReduce within Google including our experiences in using it as the basis for a rewrite of our production indexing system. Section 7 discusses related and future work.

## 2 Programming Model

The computation takes a set of *input* key/value pairs, and produces a set of *output* key/value pairs. The user of the MapReduce library expresses the computation as two functions: map and reduce.

Map, written by the user, takes an input pair and produces a set of *intermediate* key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key *I* and passes them to the reduce function.

The reduce function, also written by the user, accepts an intermediate key *I* and a set of values for that key. It merges these values

# MapReduce: Simplified Data Processing on Large Clusters

by Jeffrey Dean and Sanjay Ghemawat

## Abstract

MapReduce is a programming model and an associated implementation for processing and generating large datasets that is amenable to a broad variety of real-world tasks. Users specify the computation in terms of a *map* and a *reduce* function, and the underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, handles machine failures, and schedules inter-m
cient use of the network and disks. Programmers find the
thousand distinct MapReduce programs have been implen
past four years, and an average of one hundred thousand
Google's clusters every day, processing a total of more than

## 1 Introduction

Prior to our development of MapReduce, the authors and many others at Google implemented hundreds of special-purpose computations that process large amounts of raw data, such as crawled documents, Web request logs, etc., to compute various kinds of derived data, such as inverted indices, various representations of the graph structure of Web documents, summaries of the number of pages crawled per host, and the set of most frequent queries in a given day. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

The major c
interface that e
large-scale com
interface that a
modity PCs. Th
computations a

Section 2 de
examples. In Sec
interface tailore
Section 4 descri
we have found u
implementation
MapReduce with
sis for a rewrite o
lated and future

## 2 Program

The computatio
set of *output* k
expresses the co

Map, written
*intermediate* key
all intermediate
and passes them

The reduce function, also written by the user, accepts an intermediate key $I$ and a set of values for that key. It merges these values

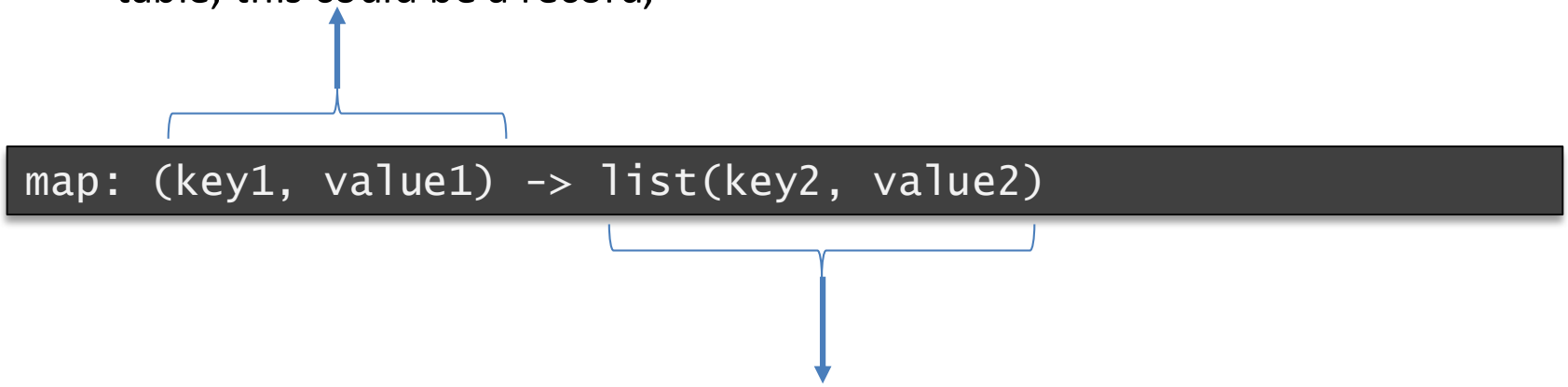
As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault tolerance, data distribution and load balancing in a library. Our abstraction is inspired by the *map* and *reduce* primitives present in Lisp and many other functional languages. We realized that most of our computations involved applying a map operation to each logical record' in our input in order to compute a set of intermediate key/value pairs, and then applying a reduce operation to all the values that shared the same key in order to combine the derived data appropriately. Our use of a functional model with user-specified map and reduce operations allows us to parallelize large computations easily and to use reexecution as the primary mechanism for fault tolerance.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault tolerance, data distribution and load balancing in a library. Our abstraction is inspired by the map and reduce primitives present in Lisp and many other functional languages. We realized that most of our computations involved applying a map operation to each logical record' in our input in order to compute a set of intermediate key/value pairs, and then applying a reduce operation to all the values that shared the same key in order to combine the derived data appropriately. Our use of a functional model with user-specified map and reduce operations allows us to parallelize large computations easily and to use reexecution as the primary mechanism for fault tolerance.

# M/R Computational Model

- A M/R job is specified by two functions: map and reduce

The input key/value pair represents a logical record in the input data source. In the case of a file this could be a line, or if the input source is a database table, this could be a record,




A single input key/value pair may result in zero or more output key/value pairs.

# M/R Computational Model

- A M/R job is specified by two functions: map and reduce

The reduce function is called once per unique map output key, and receives a list of all values emitted for that key


Like the map function, reduce can output zero to many key/value pairs.

# M/R Computational Model

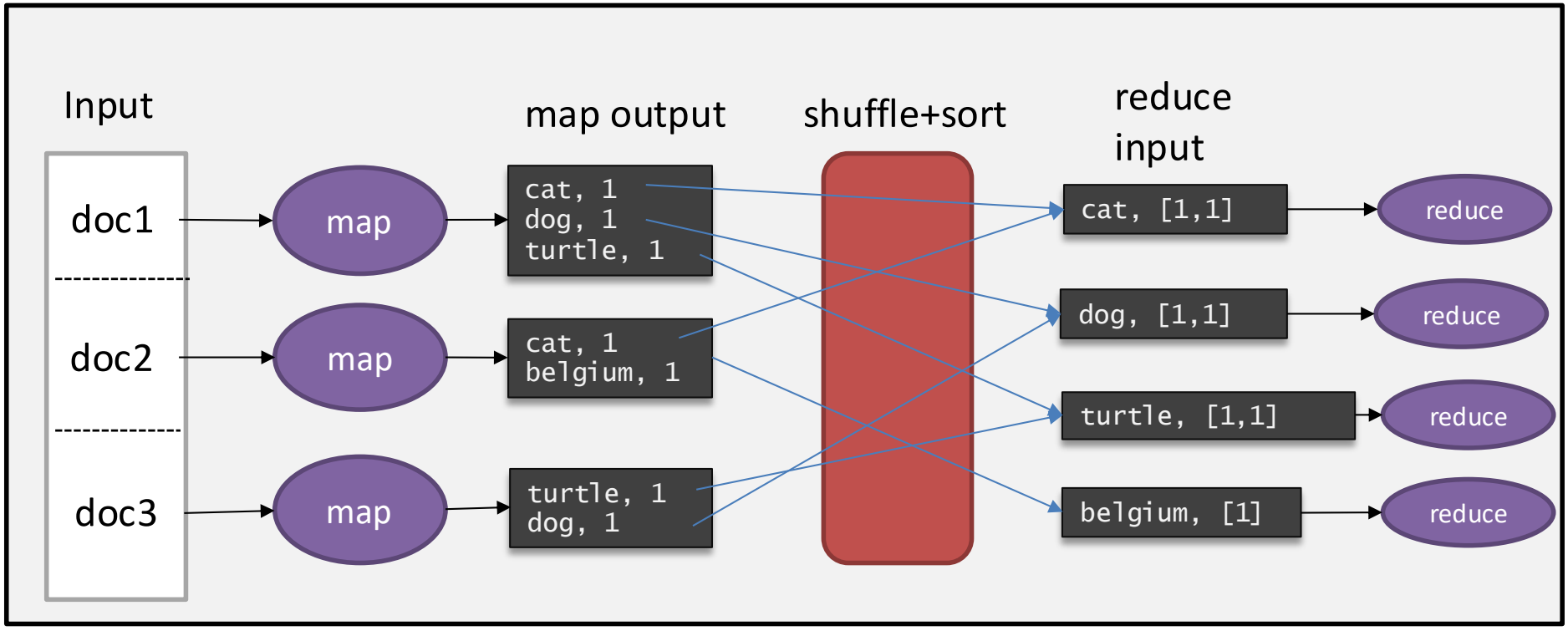
- For each word occurring in a document on the Web, count the number of occurrences across all documents.

```
def map(docid, line):
        for each word in line:
           yield (word, 1)
```

```
def reduce(word, list-of-occ-numbers):
           yield (word, sum(list-of-occ-numbers))
```

# M/R: opportunity for parallelism

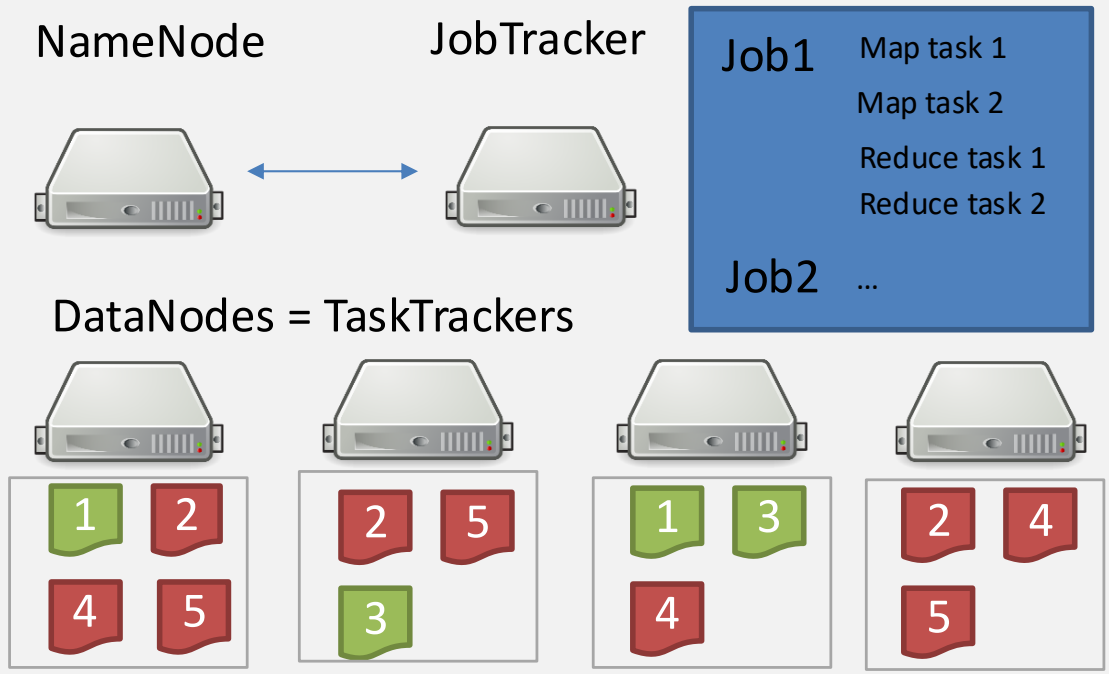
- We can spawn multiple copies of the map function (called map tasks) in parallel (at most one for each key/value pair).
- Likewise, we can spawn multiple copies of the reduce function (called reduce tasks) in parallel (at most one for each unique key output by the map).



# M/R Execution: Hadoop V1 Architecture

- Master = JobTracker – accepts jobs, decomposes them into map and reduce tasks, and schedules them for remote execution on child nodes.
- Slave = TaskTracker – accepts tasks from Jobtracker and spawns child processes to do the actual work.
- Idea: ship computation to data

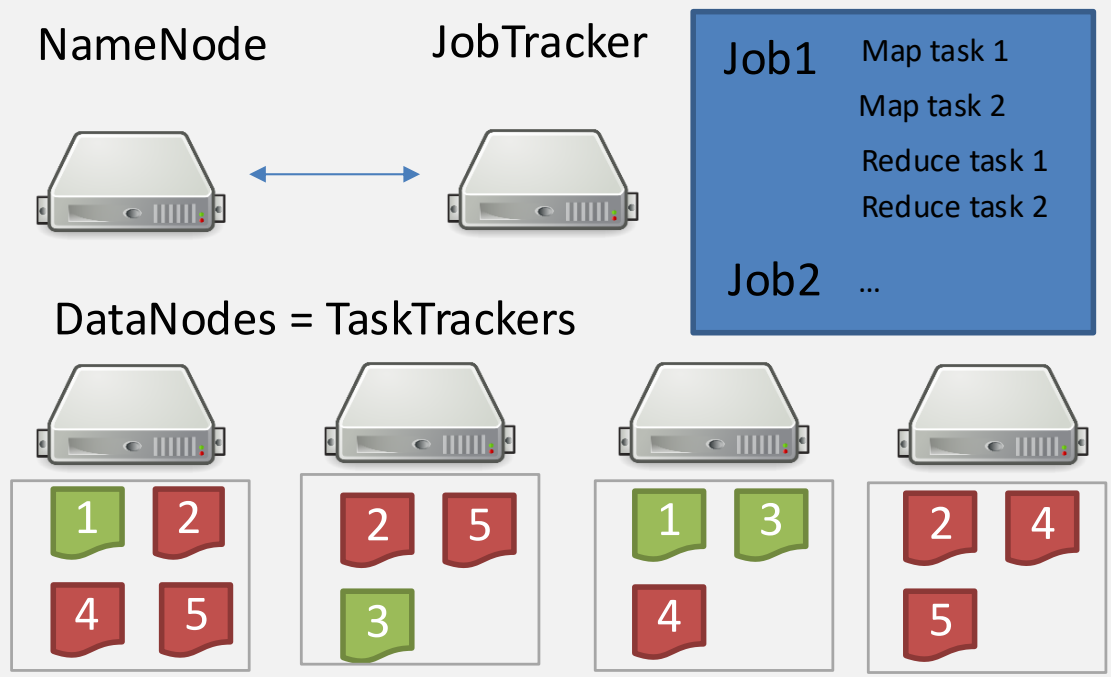
- The JobTracker accepts M/R jobs.
- If the input is a HDFS file, a map task is created for each block and sent to the node holding that block for execution.
- Map output is written to local disk.



# M/R Execution: Hadoop V1 Architecture

- Master = JobTracker – accepts jobs, decomposes them into map and reduce tasks, and schedules them for remote execution on child nodes.
- Slave = TaskTracker – accepts tasks from Jobtracker and spawns child processes to do the actual work.
- Idea: ship computation to data

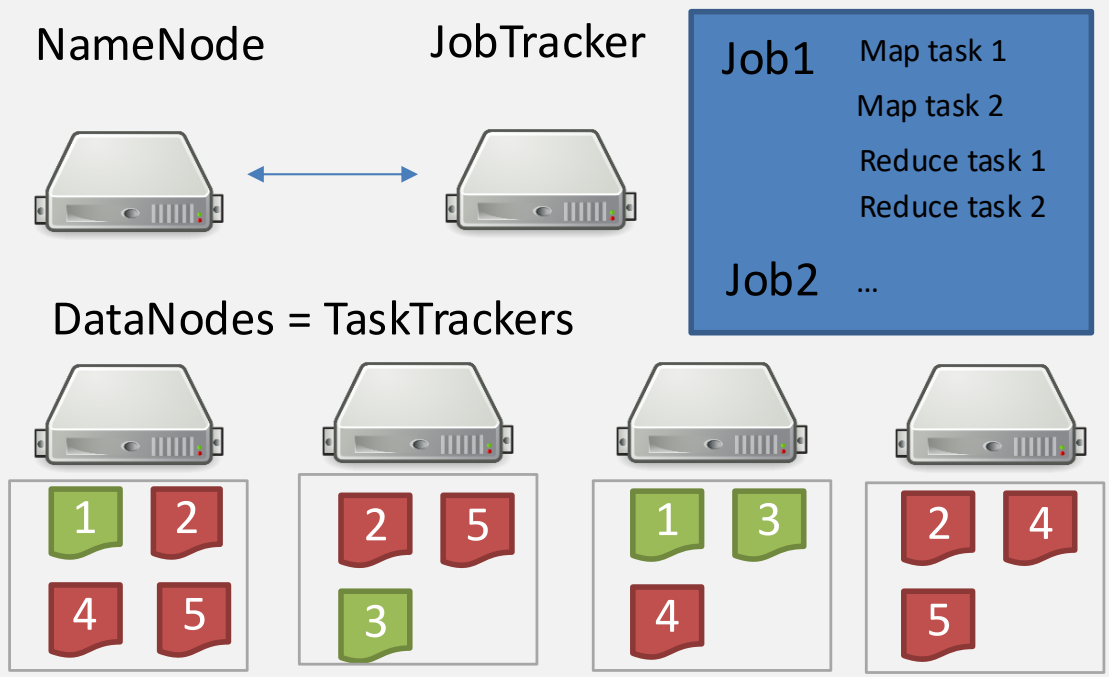
- The JobTracker creates reduce tasks intelligently
- Reduce tasks read the map outputs over the network and write their output back to HDFS.



# M/R Execution: Hadoop V1 Architecture

- Master = JobTracker – accepts jobs, decomposes them into map and reduce tasks, and schedules them for remote execution on child nodes.
- Slave = TaskTracker – accepts tasks from Jobtracker and spawns child processes to do the actual work.
- Idea: ship computation to data

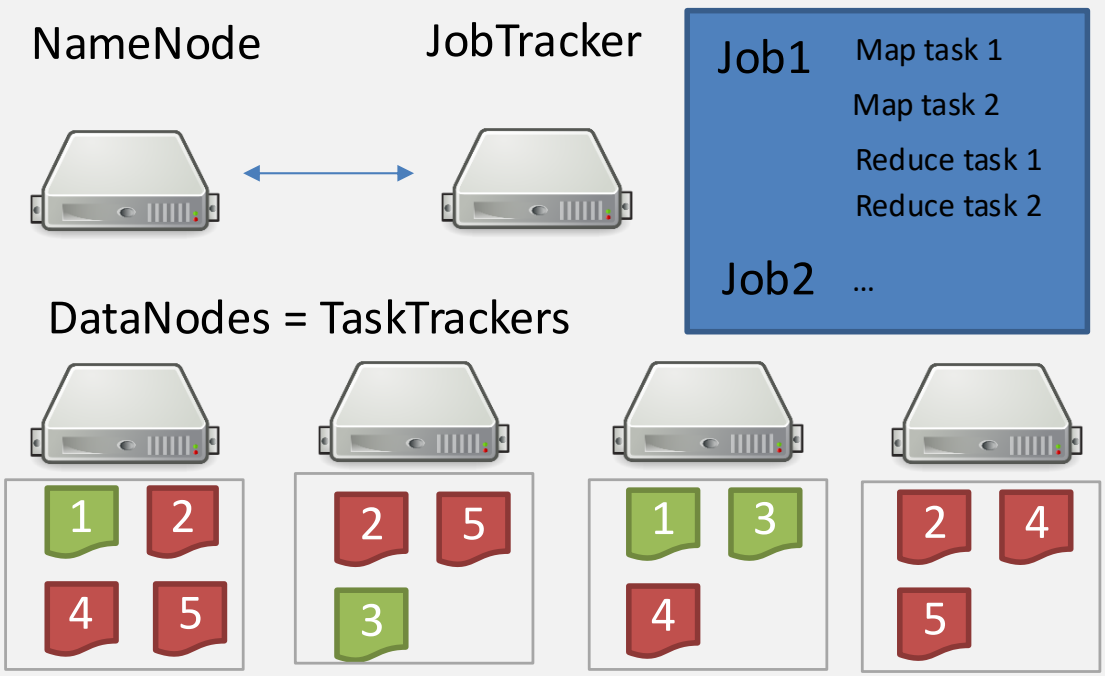
- Load balancing: The JobTracker monitors for stragglers and may spawn additional map on datanodes that hold a block replica. Whichever node completes first is allowed to proceed. The other(s) is/are killed.



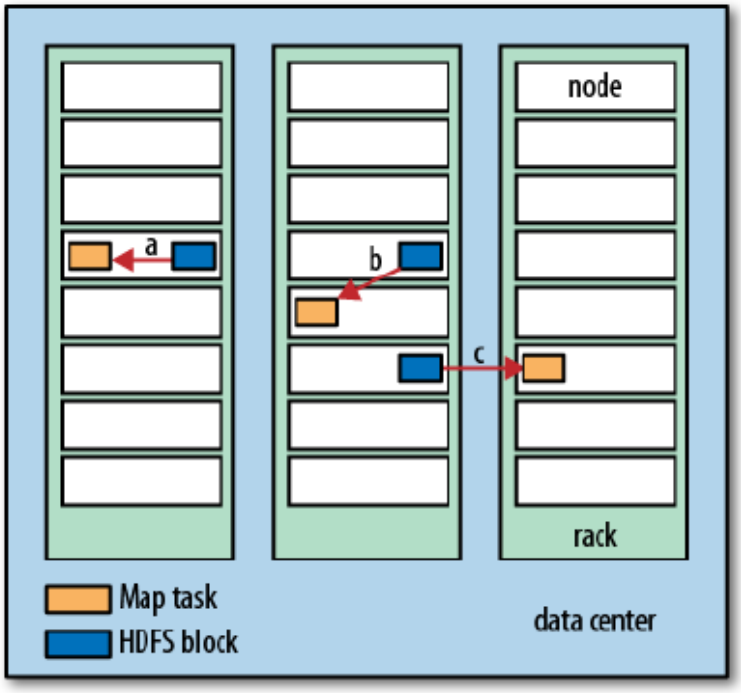
# M/R Execution: Hadoop V1 Architecture

- Master = JobTracker – accepts jobs, decomposes them into map and reduce tasks, and schedules them for remote execution on child nodes.
- Slave = TaskTracker – accepts tasks from Jobtracker and spawns child processes to do the actual work.
- Idea: ship computation to data

- Failures: In clusters with 1000s of nodes, hardware failures occur frequently. The same mechanism as for load balancing allows to cope with such failures

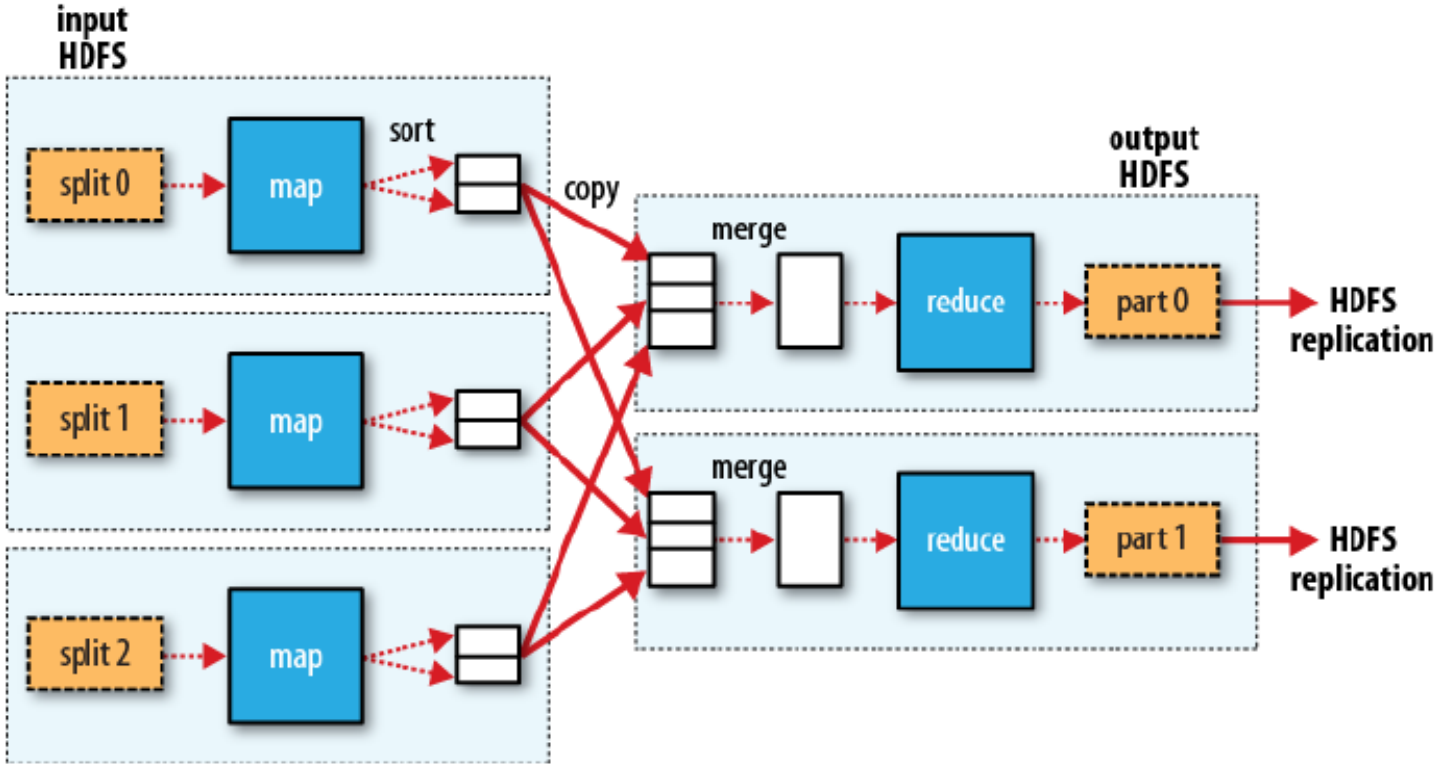


# M/R Scheduling: caveats




- Hadoop does its best to run the map task on a node where the input data resides in HDFS, hence exploiting data locality.
- Sometimes, however, all nodes hosting the HDFS block replicas for a map task's input split are busy.
- In that case, the job scheduler will look for a free map slot on a node in the same rack as one of the blocks. Very occasionally even this is not possible, so an off-rack node is used, which results in an inter-rack network transfer.

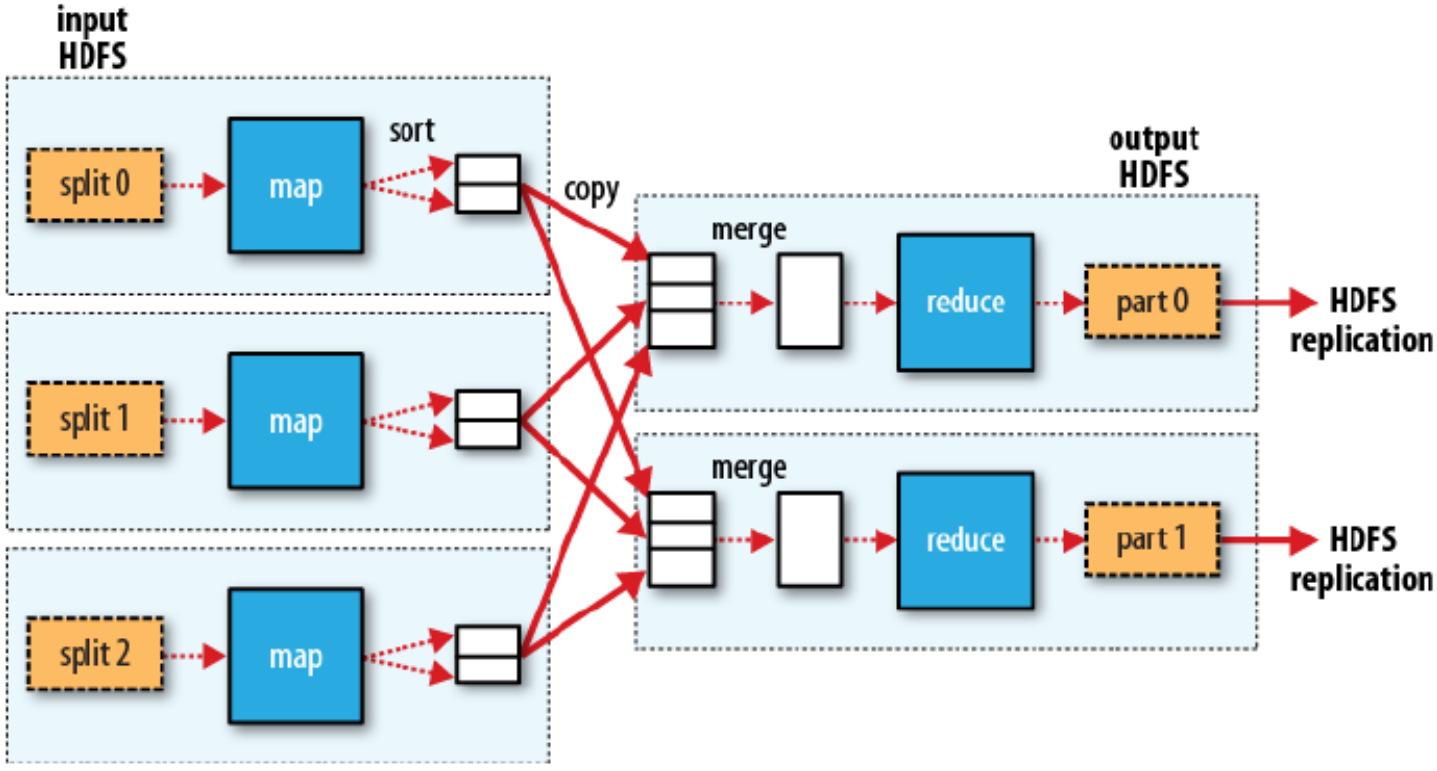
# M/R Operation: some more detail




- The number of map tasks (Mapper) is actually equal to the number of splits.
- Split is a logical division of the input data while block is a physical division of data.
- HDFS default block size is default split size.

Figure Source: Hadoop – The definitive Guide

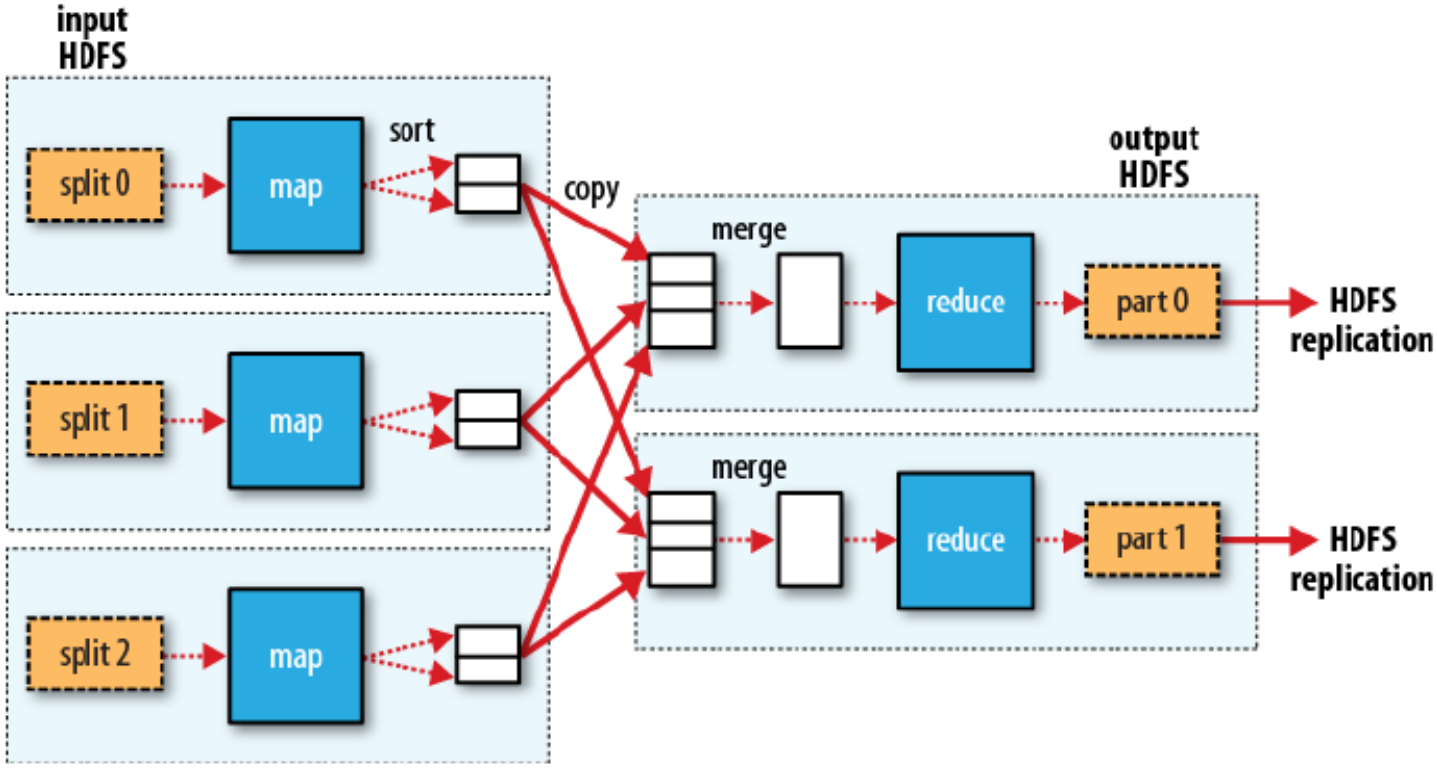
# M/R Operation: some more detail




- Each map task partitions its output; the number of partitions equals the number of reducers that will run later. Each partition is sorted already on the map-side. (One partition can still have many distinct keys!)
- The reducers copy their respective partition from the mappers, and merge them into one big sorted file.
- The net result is a distributed merge-sort between the map and reduce phases.

Figure Source: Hadoop – The definitive Guide

# M/R Operation: some more detail




- Optionally, one can specify a combine function that already performs a partial map-side reduction. The aim is then to decrease the size of data to be transferred.

```
map: (key1, value1) -> list(key2, value2)
Combine: (key2, list(value2)) -> list(key2, value2)
reduce: (key2, list(value2)) -> list(key3, value3)
```

Figure Source: Hadoop – The definitive Guide

# Combiner example

- For each word occurring in a document on the Web, count the number of occurrences across all documents.

```
def map(docid, line):
        for each word in line:
           yield (word, 1)
```

```
def combine(word, list-of-ones):
        yield (word, sum(list-of-ones))
```

```
def reduce(word, list-of-occ-numbers):
           yield (word, sum(list-of-occ-numbers))
```

# WRITING M/R PROGRAMS

- Hadoop M/R is written in Java. As such, the default way to write a M/R job is through its Java API.
- Because this can be cumbersome, Hadoop M/R also exposes the *streaming interface* which allows executables written in an arbitrary language (shell script, python, C, ...) to implement the map and reduce logic.
- In that case, the executable reads its input from stdin, and writes its output to stdout.

# REFERENCES


- S. Ghemawate, H. Gobioff, and H.-T. Leung. *The Google File System* `http://research.google.com/archive/gfs-sosp2003.pdf`
- *HDFS architecture*. `http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html`
- J. Dean and S. Ghemawat. *MapReduce: Simplified Data Processing on Large Clusters*. Communications of the ACM 2008.
- L. A. Barroso, J. Clidaras, U. Hölzle. *The datacenter as a computer*. `http://www.morganclaypool.com/doi/abs/10.2200/S00516ED2V01Y201306CAC024`
- T. White. *Hadoop: The Definitive Guide*. O'Reilly Media, 2010.

# Questions?

# ACKNOWLEDGEMENTS

Based on content created by Jan Hidders and Stijn Vansummeren.