INTRODUCTION OF BIG DATA TECHNOLOGIES

UNIT I: Introduction

Introduction to Big Data Platform – Challenges of Conventional Systems - Intelligent data analysis – Nature of Data - Analytic Processes and Tools - Analysis vs Reporting – Modern Data Analytic Tools. Big Data Analytics Process, Big Data Analytics for Business. Identifying problem and solving problem in Big Data environment. Analyzing Unstructured vs. Structured Data, Databases.

A short definition of Big Data

The definition of big data is data that contains greater variety, arriving in increasing volumes and with more velocity.

Types of Big Data Technologies (+ Management Tools)

1. Data storage

- Apache Hadoop
- Apache Spark
- Apache Hive
- Apache Flume
- ElasticSearch
- MongoDB

2. Data mining

- Rapidminer
- Presto

3. Data analytics

- Apache Spark
- Splunk
- KNIME

4. Data visualization

- Tableau
- Power BI

Apache Hadoop

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

MongoDB

MongoDB is a source-available cross-platform document-oriented database program.

Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas

Presto

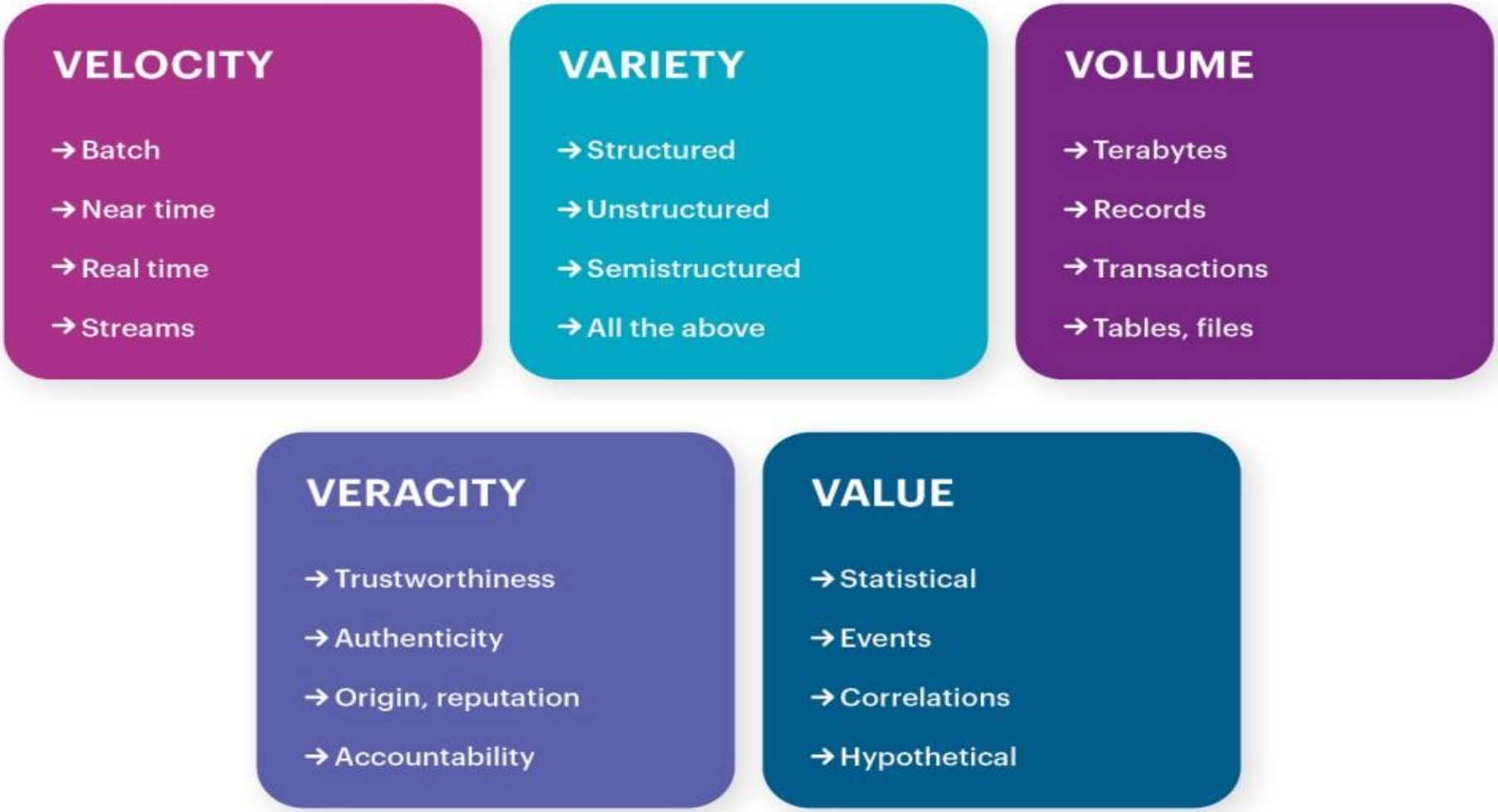
A single Presto query can process data from multiple sources like HDFS, MySQL, Cassandra, Hive and many more data sources. Presto is built in Java and easy to integrate with other data infrastructure components. Presto is powerful, and leading companies like Airbnb, DropBox, Groupon, Netflix are adopting it.

PySpark is a Python API for Apache Spark, while Spark is an open-source big data processing framework written in Scala. The main differences between PySpark and Spark are:

1. PySpark is written in Python, while Spark is written in Scala.

2. PySpark is easier to use as it has a more user-friendly interface, while Spark requires more expertise in programming.

The 5 Vs of Big Data



Source: Ishwarappa, J. Anuradha (2015). A Brief Introduction on Big Data 5Vs Characteristics and Hadoop Technology. Elsevier B.V. www.sciencedirect.com

The V's of Big Data

Volume: It refers to the size of the data. This data is generated in an incredible amount each second such as cell phones, social media, online transactions, etc.

Velocity: It is the speed at which data is generated, collected and analyzed.

Variety: This represents the different types of big data. It could be structured - having fixed format and size, semi-structured - has a structure but cannot be stored

in a database, and unstructured - Does not have any format and is hard to analyze.

Value: It means how much data is useful and meaningful. Value refers to the ability to turn your data useful for business.

Veracity: It means the trustworthiness of data in terms of quality and accuracy. Extracting loads of data is not useful if the data is messy and poor in quality (Twitter posts with abbreviations, spelling mistakes, etc.

Different processing paradigms

- *Batch processing* is when data are collected and submitted to the system in batches without human interaction. Processing is carried out at a later time depending on the availability of resources. Examples of batch processing are: monthly reporting, scientific simulations, model building.
- *Real-time processing* is when a response is guaranteed within a given time frame (seconds, milliseconds, ...). Real-time processing is required by interactive applications such as ATM transactions or computer vision.

Hadoop's MapReduce is a typical batch processing tool.

Structured vs. unstructured data

- by *structured* data one refers to highly organized data that are usually stored in relational databases or data warehouses. Structured data are easy to search but unflexible in terms of the three "V"s.

- *Unstructured* data come in mixed formats, usually require pre-processing, and are difficult to search. Unstructured data are usually stored in noSQL databases or in *data lakes* (these are scalable storage spaces for raw data of mixed formats).

- With *semi-structured* data one usually refers to structured data containing unstructured elements (such as free text).

Examples of structured/unstructured data

Property	Structured Data	Unstructured Data
Predefined Data Models	Yes	No
Data type	The structured data can have data types such as Text, Numbers, Strings,binary	The unstructured data has raw data formats such as Image, Sound, Video, documents
Technology	Relational database tables	Character and binary data
Storage	Relational databases	No-SQL database
Flexibility	Schema dependent and less flexible	More flexible due to no schema
Scalability	Difficult to scale	more scalable
Common Used	OLTP, data warehouses	Storing logs, events, multimedia files, Data lakes
Analysis	The machine learning algorithms can easily access the structured data due to fixed data structure	It is a challenging task to perform data analysis on unstructured data
Analysis	Customer records, Sales transactions, products information	Multimedia, event logs, streaming, scientific records

The challenges of Big Data

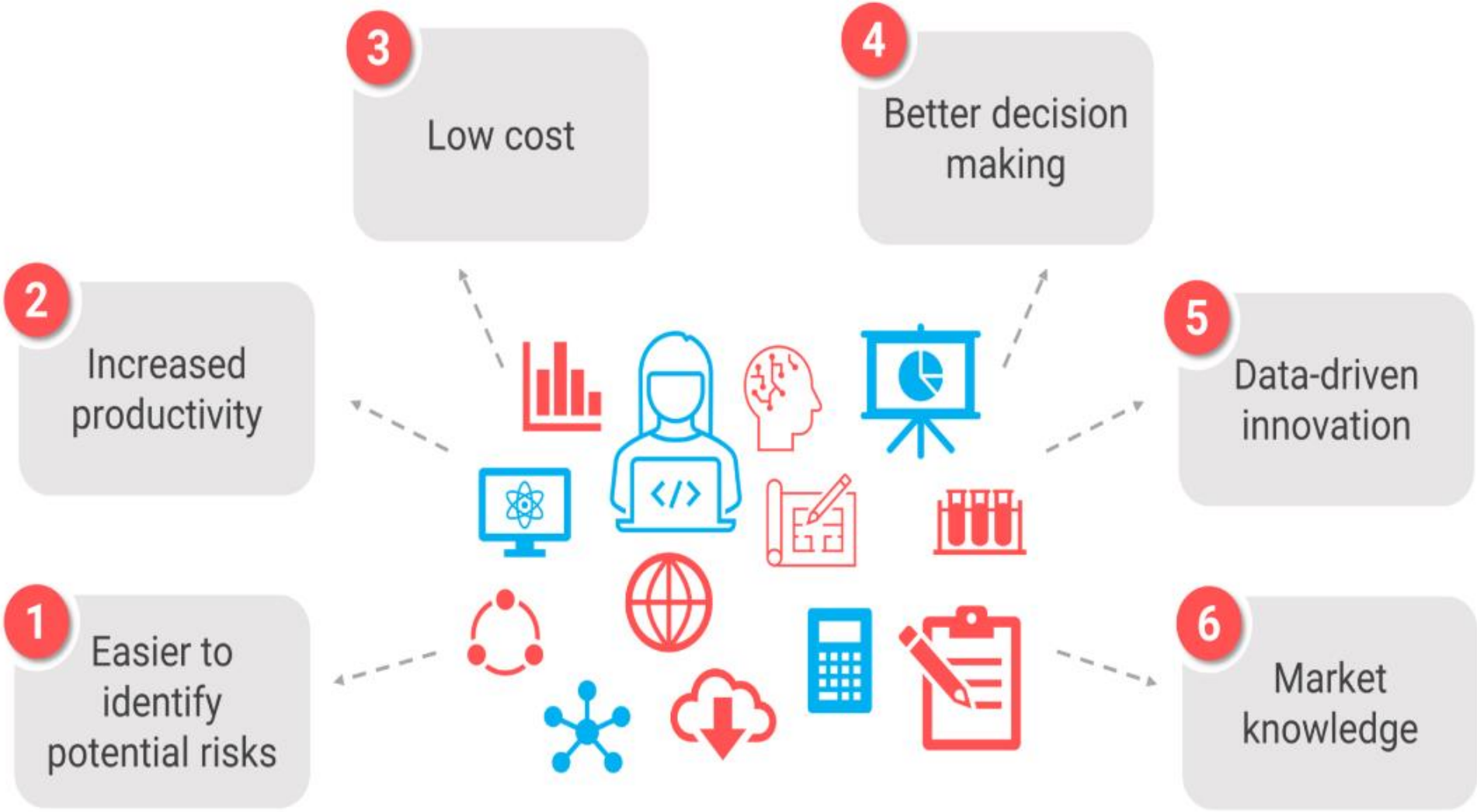
When working with large amounts of data you will sooner or later face one or more of these challenges:

- disk and memory space
- processing speed
- hardware faults
- network capacity and speed
- need to optimize resources
-
-

Use Big Data software tools address these challenges.


BIG DATA CHALLENGES
POOR DATA QUALITY
ISSUES IN SCALING
INCORRECT INTEGRATION
REAL-TIME DATA
LACK OF EXPERTISE
SECURITY AND PRIVACY
ORGANIZATIONAL RESISTANCE
DATA VERIFICATION
HEAVY EXPENSES
WIDE TECHNOLOGIES' VARIETY

6 Advantages of Big Data



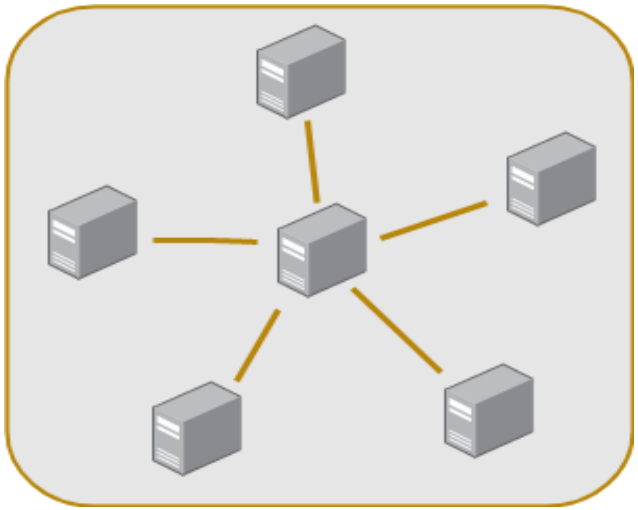
Distributed computing for Big Data

Source: VSC-4 ©Matthias Heisler

Traditional data processing tools are inadequate for large amounts of data.

Distributed computation makes it possible to work with Big Data optimizing time and available resources.

What is distributed computing?



A distributed computer system consists of several interconnected *nodes*. Nodes can be physical as well as virtual machines or containers.

When a group of nodes provides services and applications to the client as if it were a single machine, then it is also called a *cluster*.

Benefits of distributed computing

- **Performance**: supports intensive workloads by spreading tasks
- across nodes
- **Scalability**: new nodes can be added to increase capacity
- **Fault tolerance**: resilience in case of hardware failures

The Hadoop distributed computing architecture

Hadoop for distributed data processing

Hadoop is a framework for running jobs on clusters of computers that provides a good abstraction of the underlying hardware and software.

"Stripped to its core, the tools that Hadoop provides for building distributed systems – for data storage, data analysis, and coordination – are simple. If there's a common theme, it is about raising the level of abstraction – to create building blocks for programmers who just happen to have lots of data to store, or lots of data to analyze, or lots of machines to coordinate, and who don't have the time, the skill, or the inclination to become distributed systems experts to build the infrastructure to handle it.[2]"

[2] White T. *Hadoop: The Definitive Guide*. O'Reilly, 2015.

Hadoop: some facts

Hadoop[3] is an open-source project of the Apache Software Foundation. The project was created to facilitate computations involving massive amounts of data.

- its core components are implemented in Java
- initially released in 2006. Last stable version is 3.3.1 from June 2021
- originally inspired by Google's MapReduce[4] and the proprietary GFS (Google File System)

[3] Apache Software Foundation. *Hadoop*. url: https://hadoop.apache.org.

[4] J. Dean and S. Ghemawat. "MapReduce: Simplified data processing on large clusters." In: *Proceedings of Operating Systems Design and Implementation (OSDI)*. 2004.

Hadoop's features

Hadoop's features addressing the challenges of Big Data:

- scalability
- fault tolerance
- high availability
- distributed cache/data locality
- cost-effectiveness as it does not need high-end hardware
- provides a good abstraction of the underlying hardware
- easy to learn
- data can be queried trough SQL-like endpoints (Hive, Cassandra)

Hadoop's distinguishing features

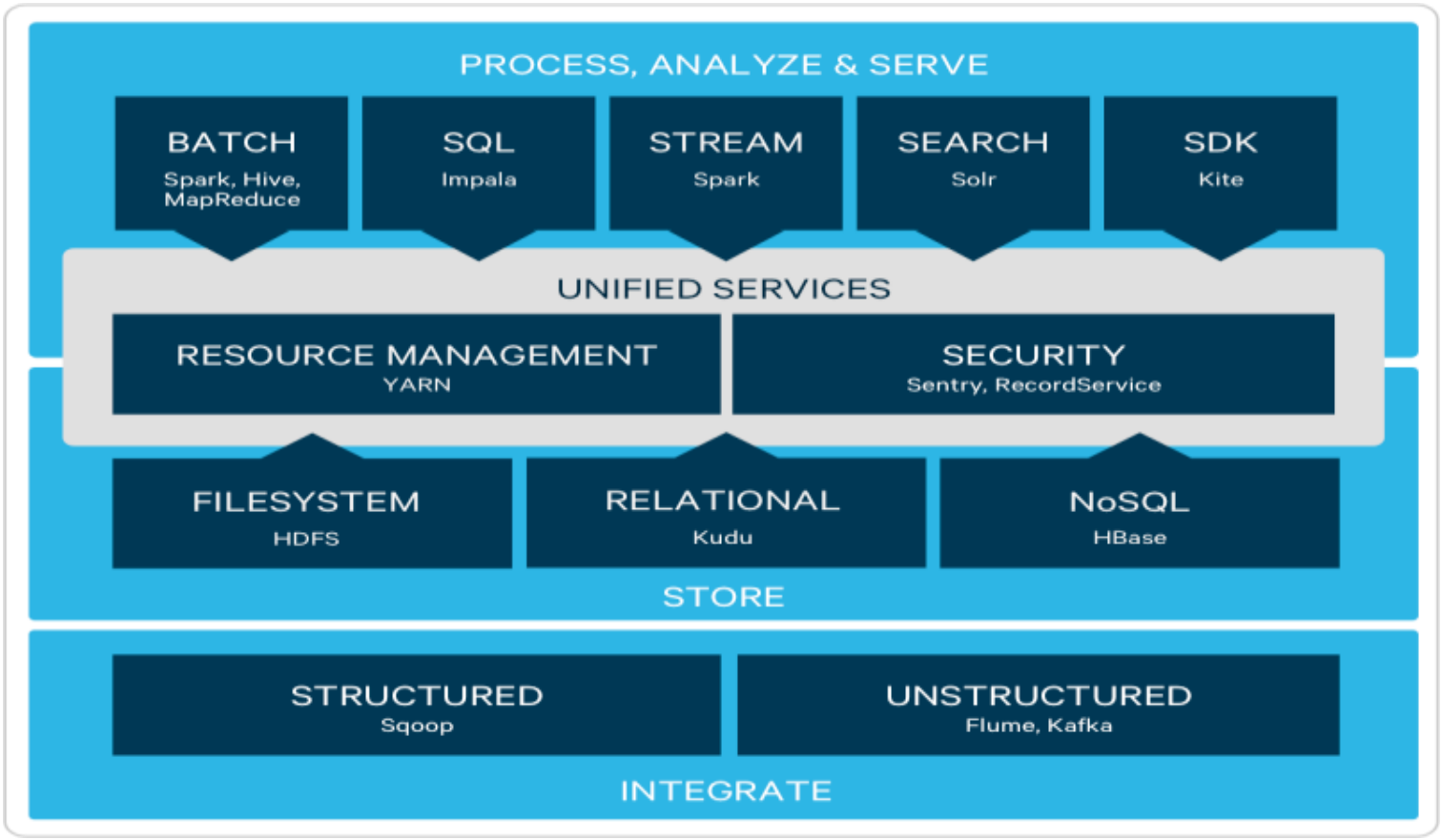
- ***fault tolerance***: the ability to withstand hardware or network failures (also: *resilience*)
- ***high availability*** : this refers to the system minimizing downtimes by eliminating single points of failure
- ***data locality*** : task are run on the node where data are located, in order to reduce time-consuming transfer of data

The Hadoop core

The core of Hadoop consists of:

- Hadoop common, the core libraries
- HDFS, the Hadoop Distributed File System
- MapReduce
- the YARN (Yet Another Resource Negotiator) resource manager

The Hadoop ecosystem



Source: Cloudera

There's a whole constellation of open source components for collecting, storing, and processing big data that integrate with Hadoop.

Some useful tools that integrate with Hadoop

Just to mention a few:

Spark in-memory computation engine superseding MapReduce

Kafka a distributed streaming system that allows to integrate multiple streams of data for real-time processing

Zookeeper synchronization tool for distributed systems

Hbase a noSQL database (key-value store) that runs on the Hadoop distributed filesystem

Hive a distributed datawarehouse system

Presto a distributed SQL query engine

Oozie a workflow scheduler

All these tools are open source.

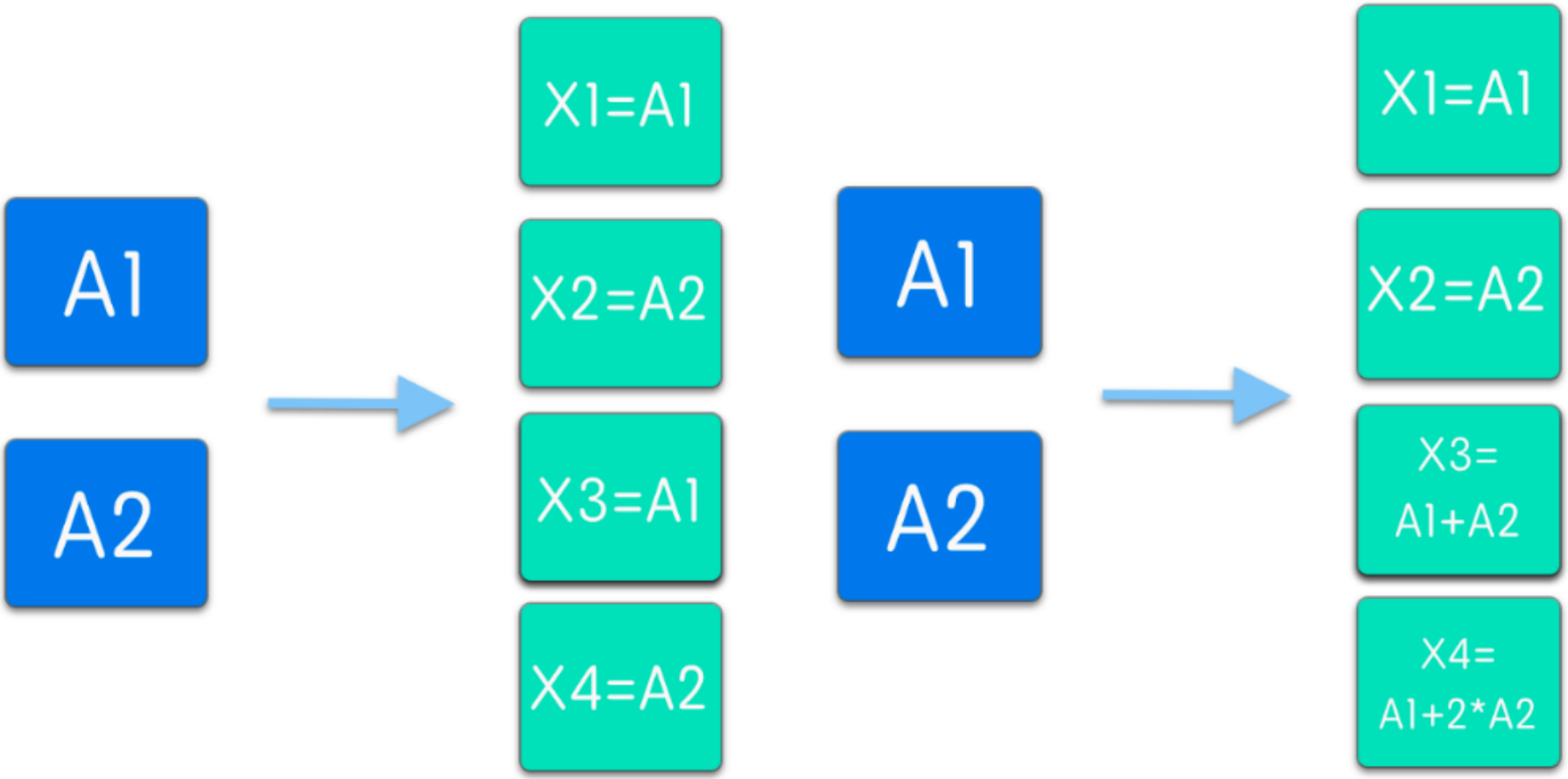
The Hadoop Distributed File System (HDFS)

HDFS stands for Hadoop Distributed File System and it takes care of partitioning data across a cluster.

In order to prevent data loss and/or task termination due to hardware failures HDFS uses either

- replication (creating multiple copies —usually 3— of the data)
- erasure coding

Data redundancy (obtained through replication or erasure coding) is the basis of Hadoop's fault tolerance.


A1
A2
X1=A1
X2=A2
X3=A1
X4=A2
A1
A2
X1=A1
X2=A2
X3=
A1+A2
X4=
A1+2*A2

Replication vs. Erasure Coding

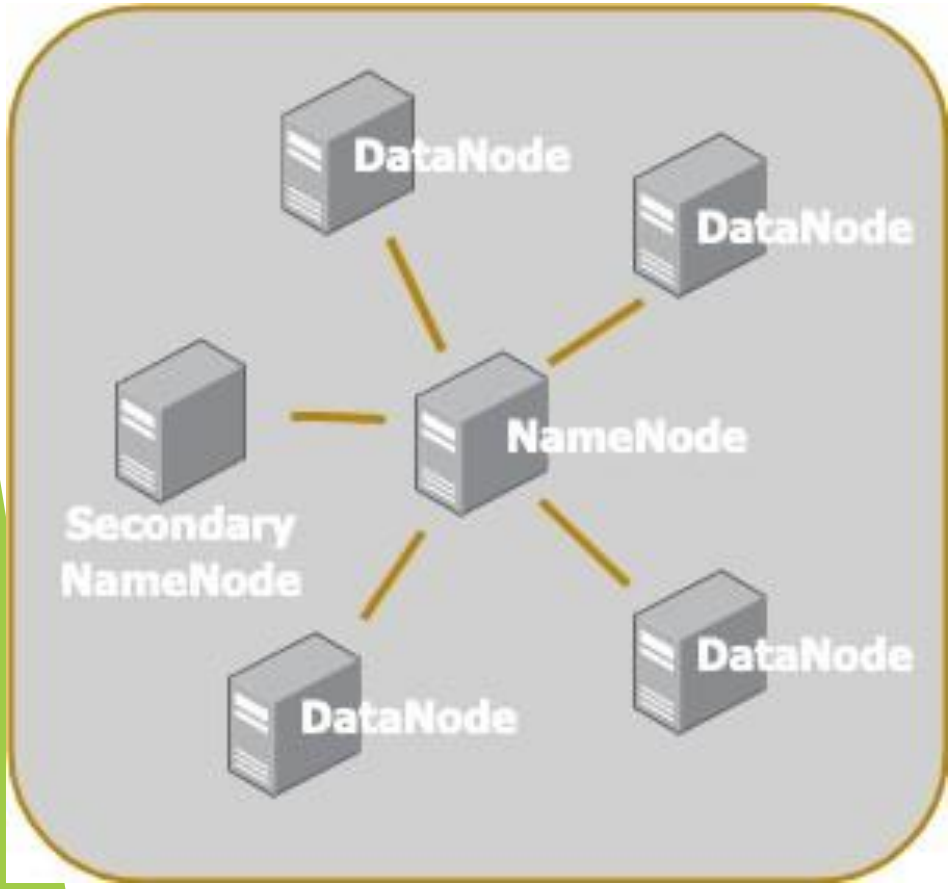
In order to provide protection against failures one introduces:

- data redundancy
- a method to recover the lost data using the redundant data

Replication is the simplest method for coding data by making n copies of the data. n-fold replication guarantees the availability of data for at most n − 1 failures and it has a storage overhead of 200% (this is equivalent to a storage efficiency of 33%).

Erasure coding provides a better storage efficiency (up to to 71%) but it can be more costly than replication in terms of performance.

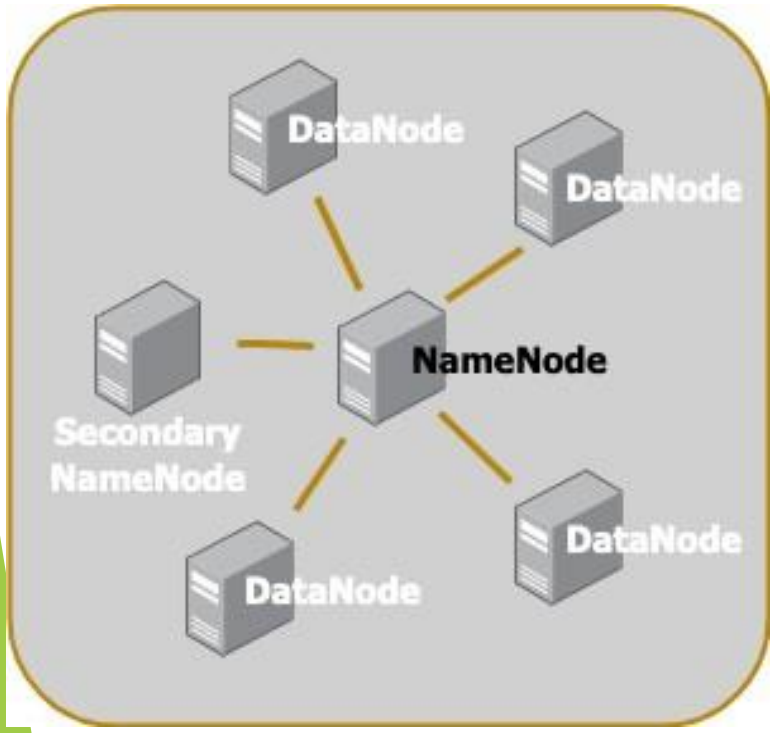
HDFS architecture



A typical Hadoop cluster installation consists of:

- a NameNode
- a secondary NameNode
- multiple DataNodes

HDFS architecture: NameNode

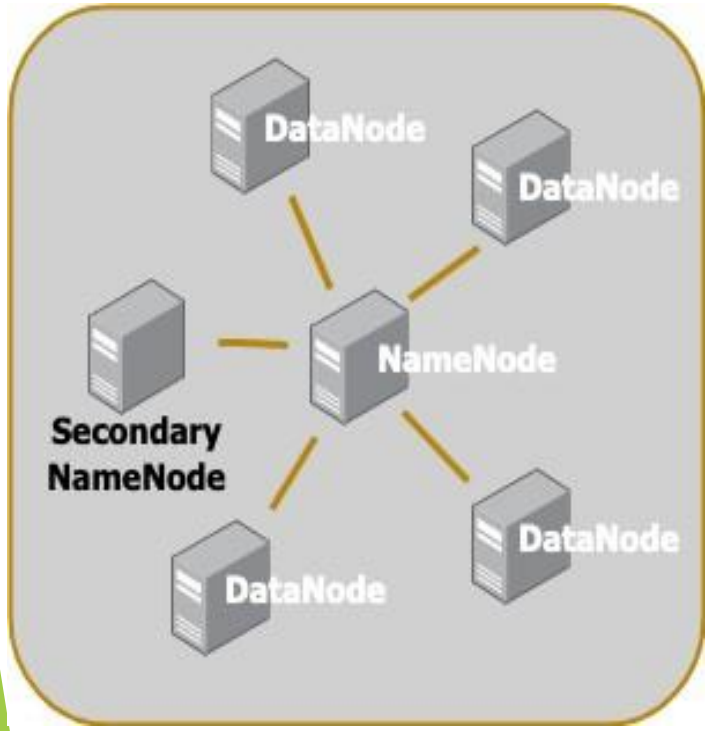


NameNode

The NameNode is the main point of access of a Hadoop cluster. It is

responsible for the bookkeeping of the data partitioned across the DataNodes, manages the whole filesystem metadata, and performs load balancing

HDFS architecture: Secondary NameNode

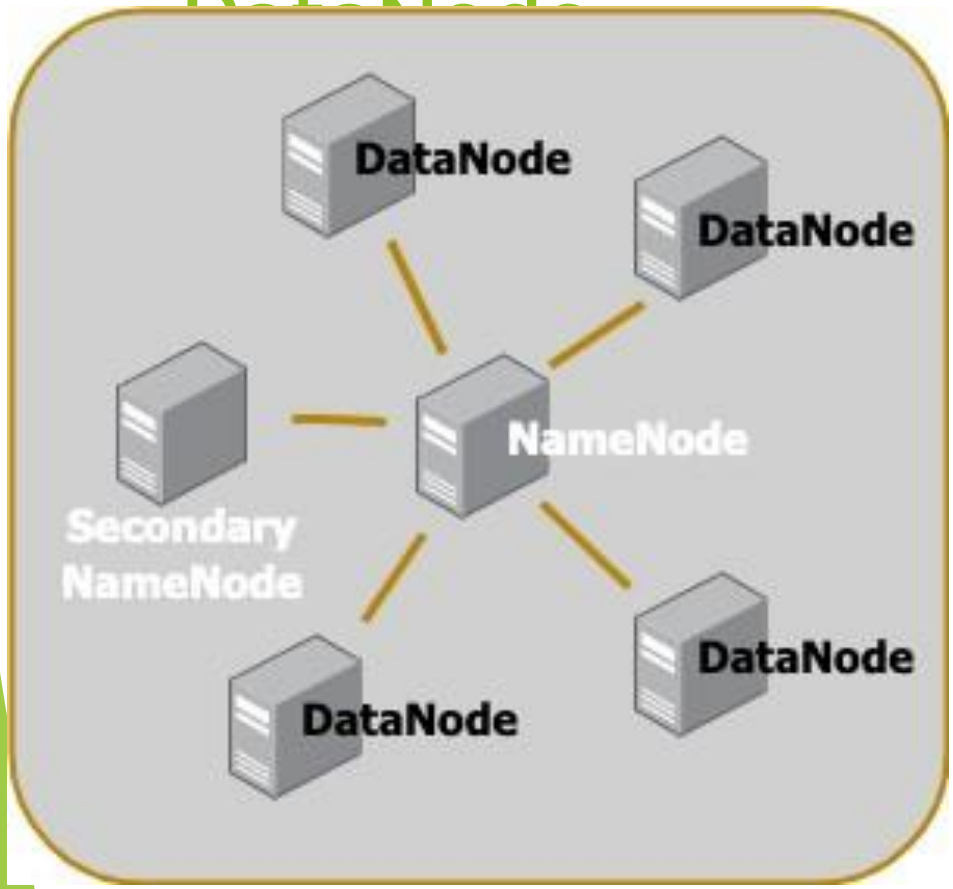


Secondary NameNode

Keeps track of changes in the NameNode performing regular snapshots, thus allowing quick startup.

An additional *standby node* is needed to guarantee high availability (since the NameNode is a single point of failure).

HDFS architecture: DataNode



DataNode

Here is where the data is saved and the computations take place (data nodes should actually be called “data and compute nodes”).

HDFS architecture: internal data representation

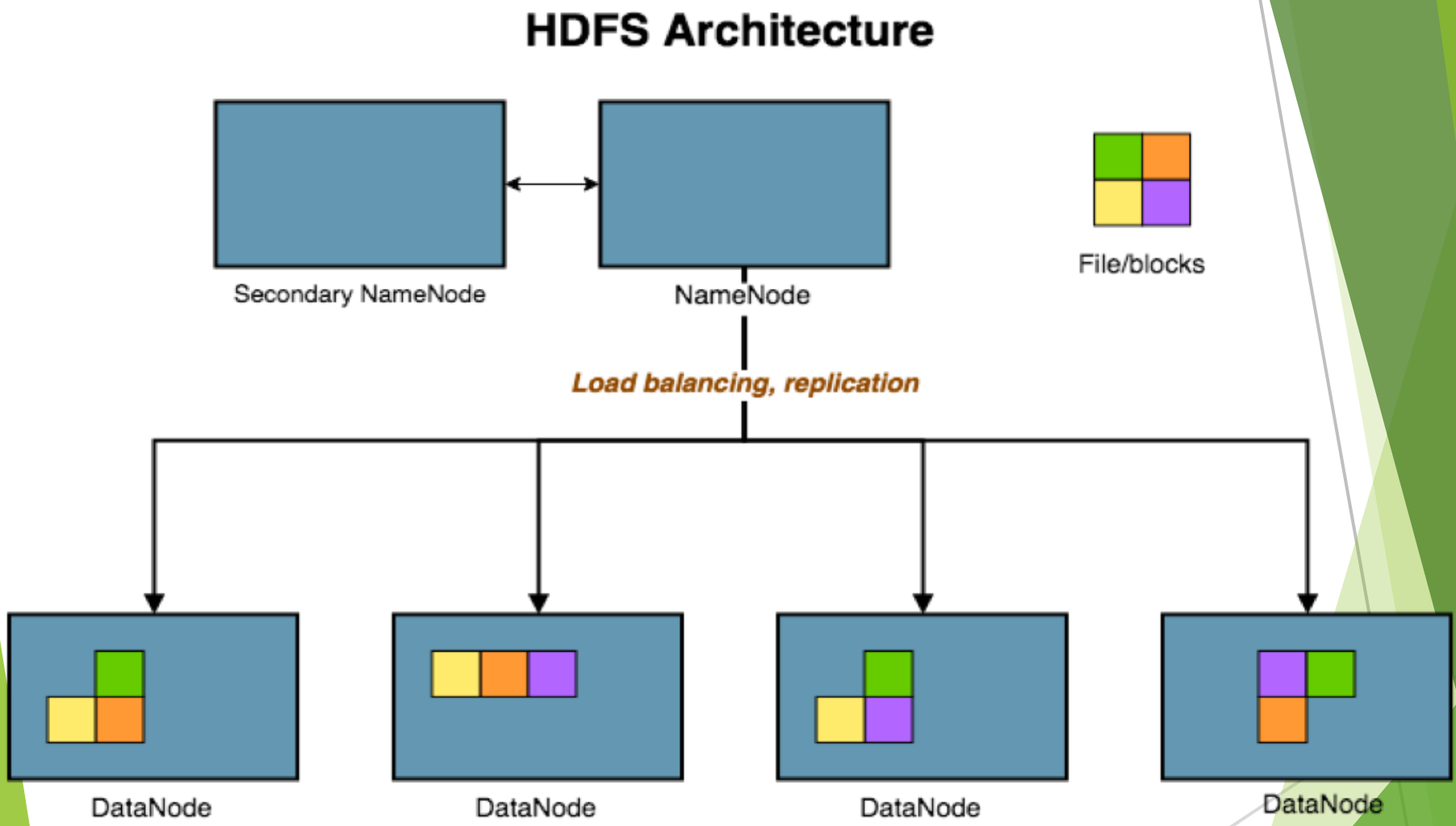
HDFS supports working with very large files.

Internally, data are split into $blocks$. One of the reason for splitting data into blocks is that in this way block objects all have the same size.

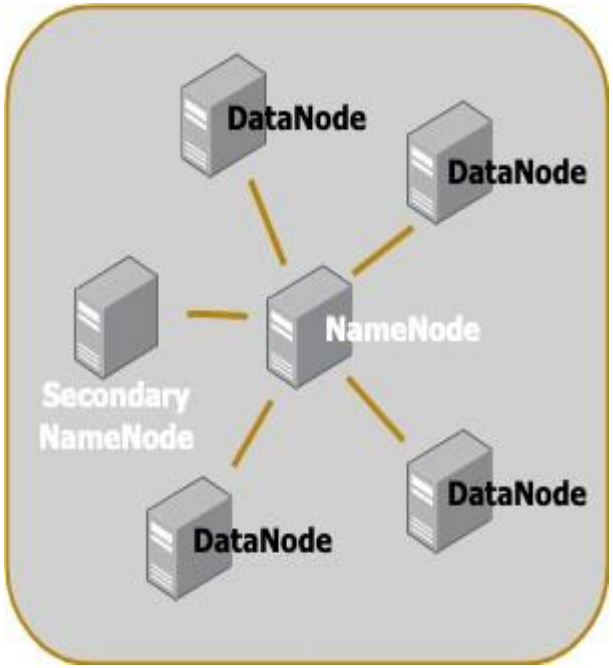
The block size in HDFS can be configured at installation time and it is by default **128MiB** (approximately **134MB**).

Note: Hadoop sees data as a bunch of records and it processes multiple files the same way it does with a single file. So, if the input is a directory instead of a single file, it will process all files in that directory.

HDFS architecture



Management of DataNode failures



Each DataNode sends a *heartbeat* message to the NameNode periodically.

Whenever a DataNode becomes unavailable (due to network or hardware failure), the NameNode stops sending requests to that node and creates new replicas of the blocks stored on that node.

Blocks versus partitions

File partitions are logical divisions of the data and should not be confused with blocks, that are physical chunks of data (i.e. each block has a physical location on the hardware).

The WORM principle of HDFS

The Hadoop Distributed File System relies on a simple design principle for data known as Write Once Read Many (WORM).

"A file once created, written, and closed need not be changed except for appends and truncates. Appending the content to the end of the files is supported but cannot be updated at arbitrary point. This assumption simplifies data coherency issues and enables high throughput data access.[5]*"*

The data immutability paradigm is also discussed in Chapter 2 of "Big Data".[6]

---

[5] Apache Software Foundation. *Hadoop*.

[6] Warren J. and Marz N. *Big Data*. Manning publications, 2015.

Data biases

Whenever one works with data, one should keep in mind that data is inherently biased.

For instance, in data harvested from the web some categories of people or themes could be underrepresented due to social, cultural, economic conditions.

And if that's not enough, alone choosing what kind of data to focus on introduces bias.

A good starting point for thinking about biases is.[8]

[8] Ricardo Baeza-Yates. "Bias on the web." In: *Communications of the ACM* 61.6 (2018), pp. 54–61.

MapReduce

MapReduce: Idea

The MapReduce paradigm is inspired by the computing model commonly used in functional programming.

Applying the same function independently to items in a dataset either to transform (*map*) or collate (*reduce*) them into new values, works well in a distributed environment.

MapReduce: Idea

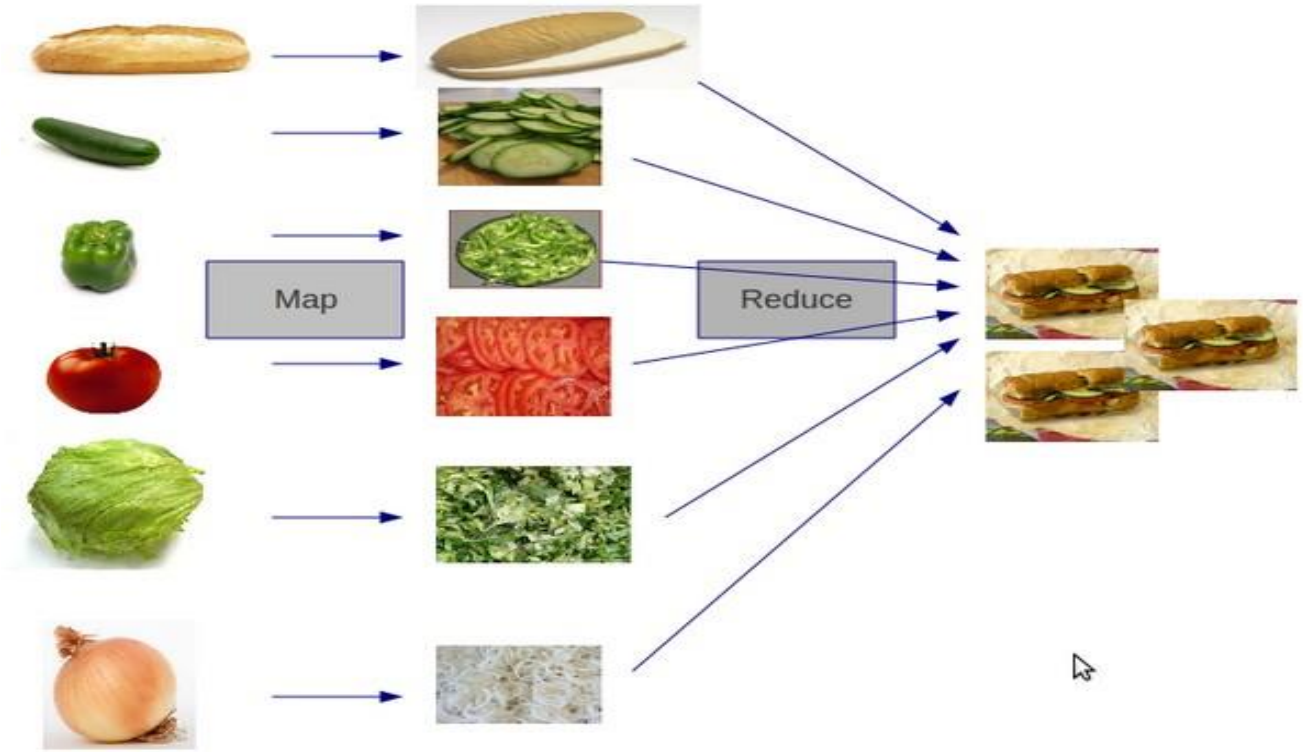


Image source: Stack Overflow

The origins of MapReduce

The 2004 paper "*MapReduce: Simplified Data Processing on Large Clusters*" by two members of Google's R&D team, Jeffrey Dean and Sanjay Ghemawat, is the seminal article on MapReduce.

The article describes the methods used to split, process, and aggregate the large amount of data for the Google search engine.

The open-source version of MapReduce was later released within the Apache Hadoop project.

The phases of MapReduce

The phases of a MapReduce job:

- **split**: data is partitioned across several computer nodes
- **map**: apply a map function to each chunk of data
- **sort & shuffle**: the output of the mappers is sorted and distributed to the reducers
- **reduce**: finally, a reduce function is applied to the data and an output is produced

The phases of MapReduce

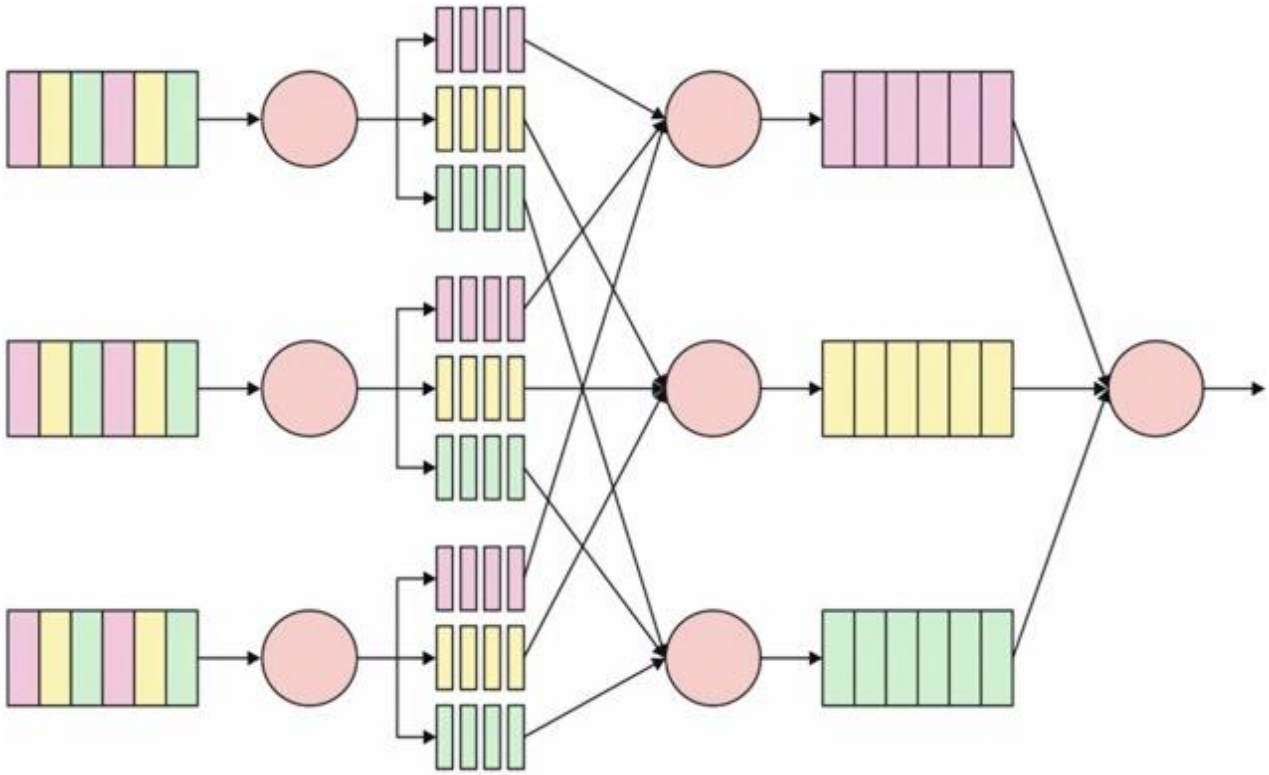


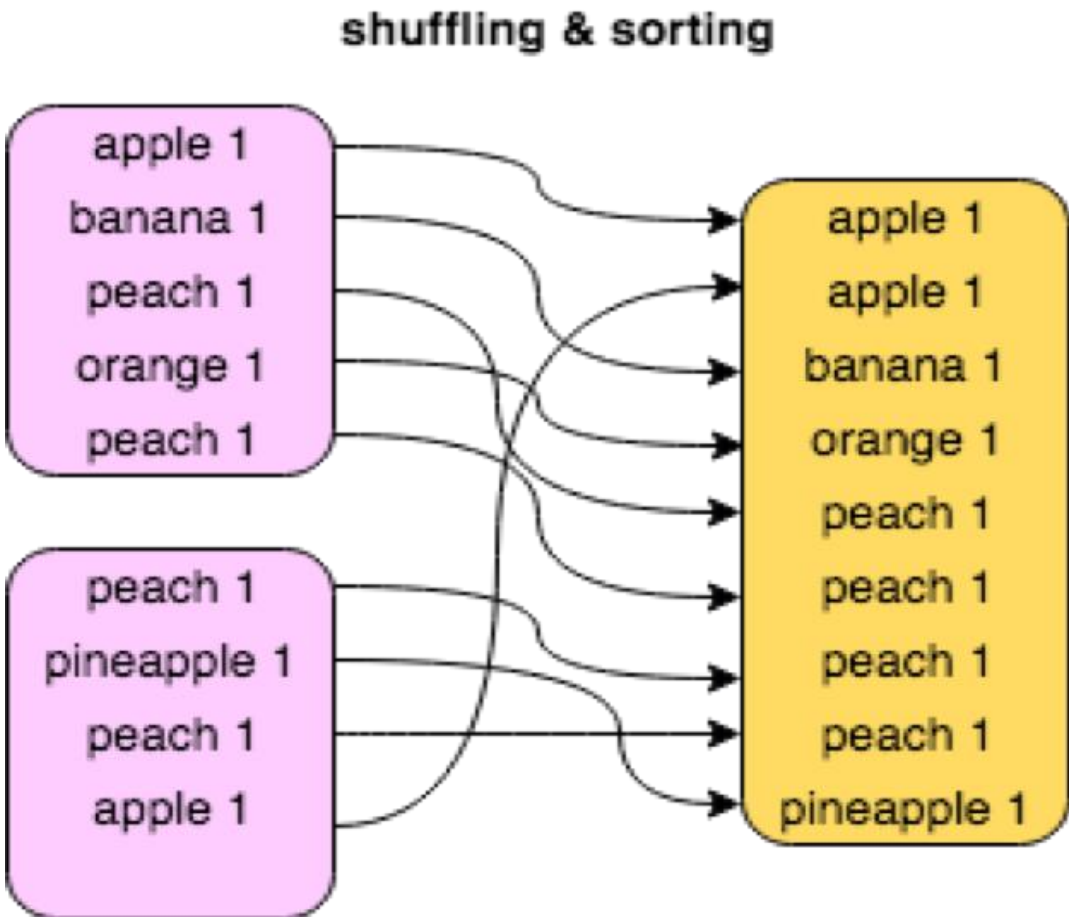
Image source: Nature

MapReduce: shuffling and sorting

The shuffling and sorting phase is often the the most costly in a MapReduce job as the mapping outputs has to be merged and sorted in order to transfer them to the reducer(s).

The purpose of sorting is to provide data that is already grouped by key to the reducer. This way reducers can iterate over all values from each group.

MapReduce: shuffling and sorting



MapReduce: shuffling and sorting

It is also possible for the user to interact with the splitting, sorting and shuffling phases and change their default behavior, for instance by managing the amount of splitting or defining the sorting comparator.

While splitting, sorting and shuffling are done by the framework, the map and reduce functions are defined by the user.

MapReduce: Additional Notes

- Usually a single mapper and reducer do not suffice for a task -> Chaining MapReduce Jobs
- Output key-value pair can contain custom input format or custom data types in case e.g more or special objects have to be passed.

MapReduce: Key Takeaways

- the same map (and reduce) function is applied to all the chunks in the data
- the mapping and reduce functions have to be defined, custom splitting or sorting are optional as they are given by most MapReduce libraries.
- the map computations can be carried out in parallel because they're completely independent from one another.