

SOMACHINE



INSTITUTO DE
ASTROFÍSICA DE
ANDALUCÍA



CSIC



Andalusian
Research Institute in
Data Science and
Computational Intelligence



UNIVERSIDAD
DE GRANADA



Big Data: Foundations and Frameworks

A. Fernández. Instituto Andaluz Interuniversitario en Data Science and Computational Intelligence. **Universidad de Granada.**



Astronomy questions require astronomical data

How the first stars and galaxies were formed? What is the nature of the dark matter?

Is there life in other planets?

Simulations to reproduce the observable universe are complex, and generate Big Data

Outline



1

- Big Data. Big Data Science

2

- Why Big Data? Google and the MapReduce programming model

3

- Big Data technologies: Hadoop / Spark ecosystem

4

- Big Data Analytics: Libraries for Data Analytics in Big Data. Case studies

5

- Final Comments

Outline



1

- **Big Data. Big Data Science**

2

- Why Big Data? Google and the MapReduce programming model

3

- Big Data technologies: Hadoop / Spark ecosystem

4

- Big Data Analytics: Libraries for Data Analytics in Big Data. Case studies

5

- Final Comments



What is Big Data

Big Data and Data Science



SCALE OF DATA
VOLUME



FORMS OF DATA
VARIETY

BIG DATA

VELOCITY

ANALYSIS OF DATA-FLOW



VERACITY

UNCERTAINTY OF DATA



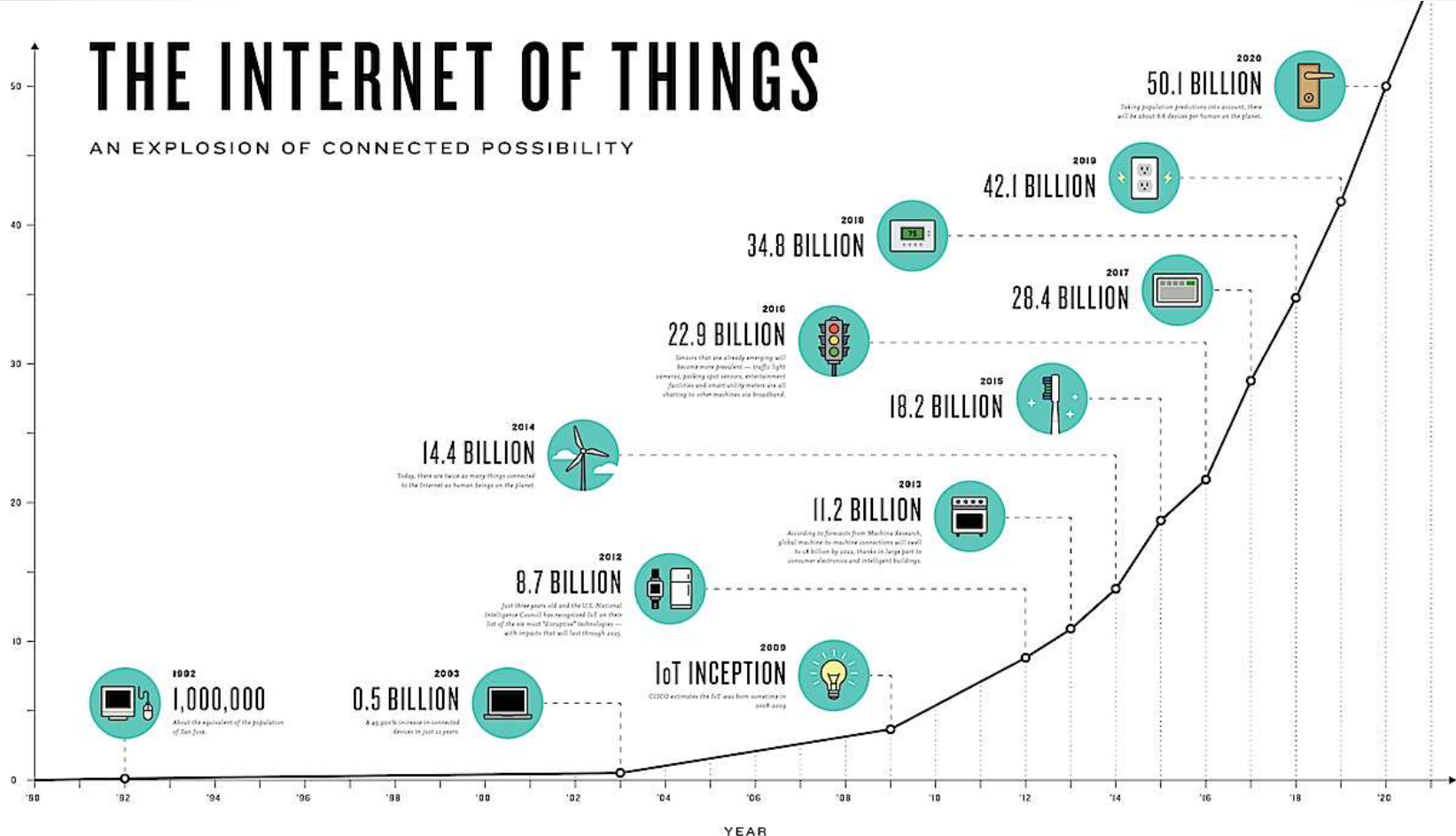
What is Big Data?

The “4” Vs determining the sides of the story

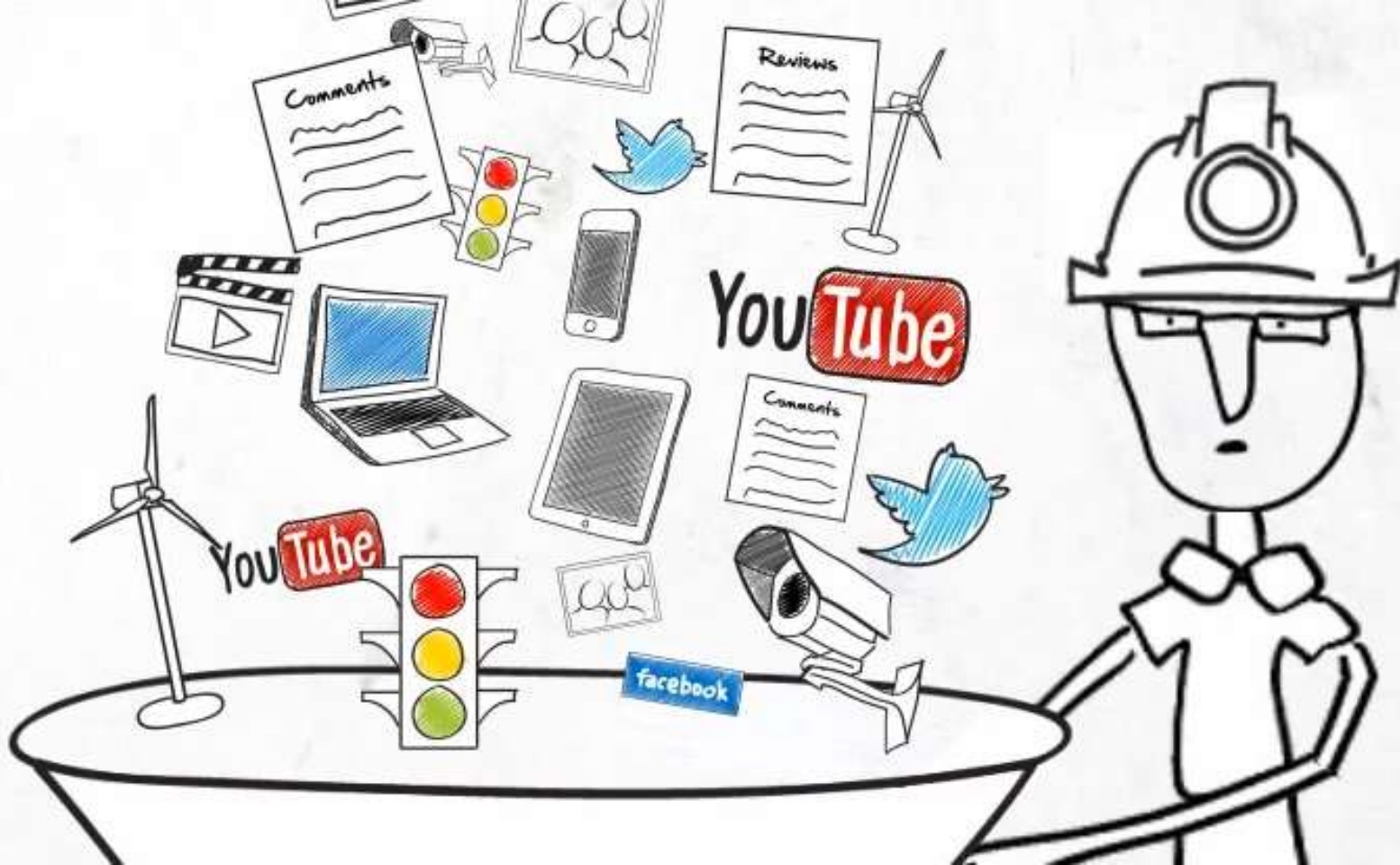
THE INTERNET OF THINGS

AN EXPLOSION OF CONNECTED POSSIBILITY

BILLIONS OF DEVICES

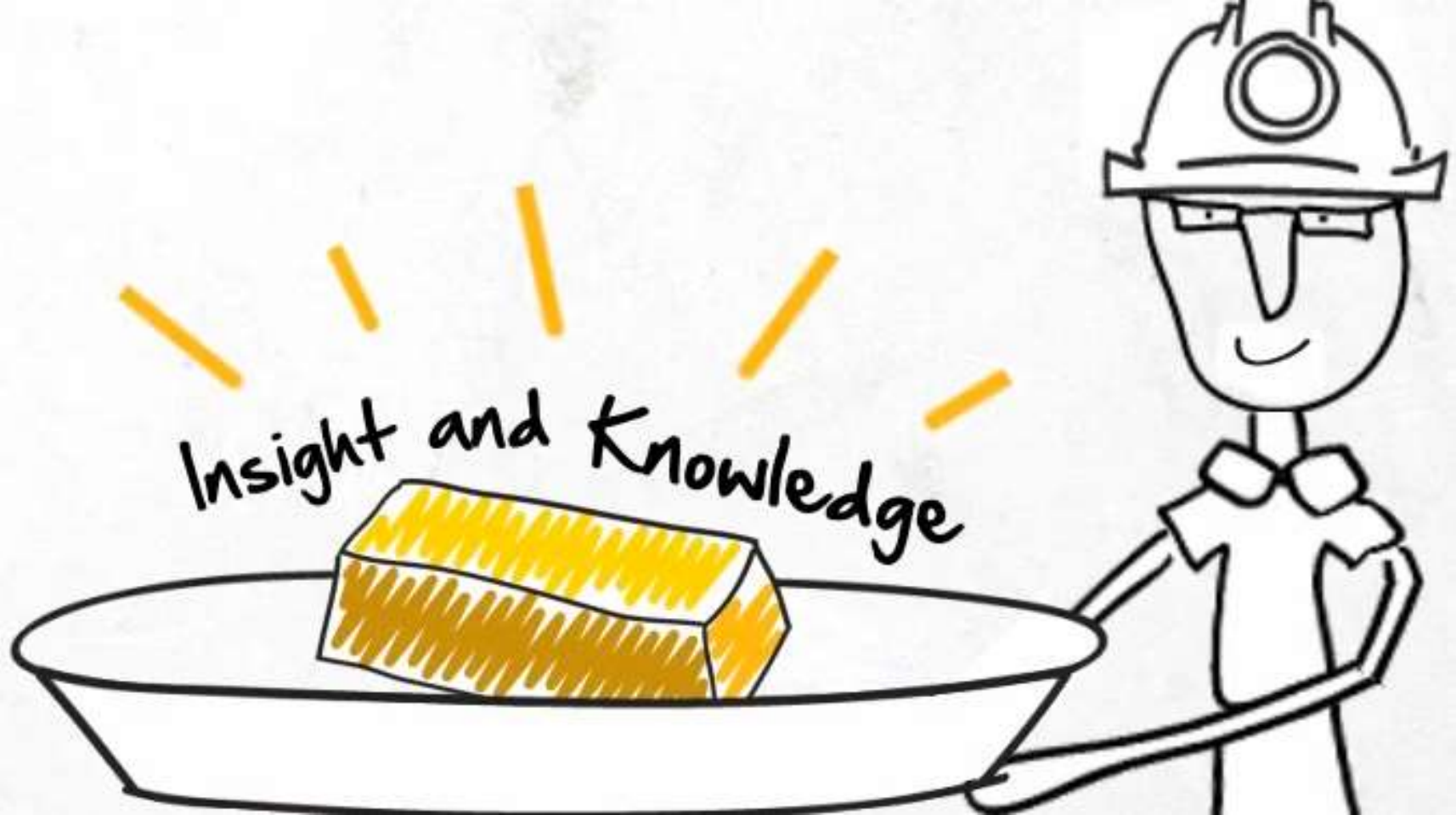


The growth in data generation and their possibilities



The “Internet of Things”

Different sources generate a large amount of data



What is Big Data?

Methods to transform raw data to actionable knowledge

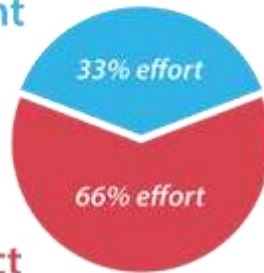
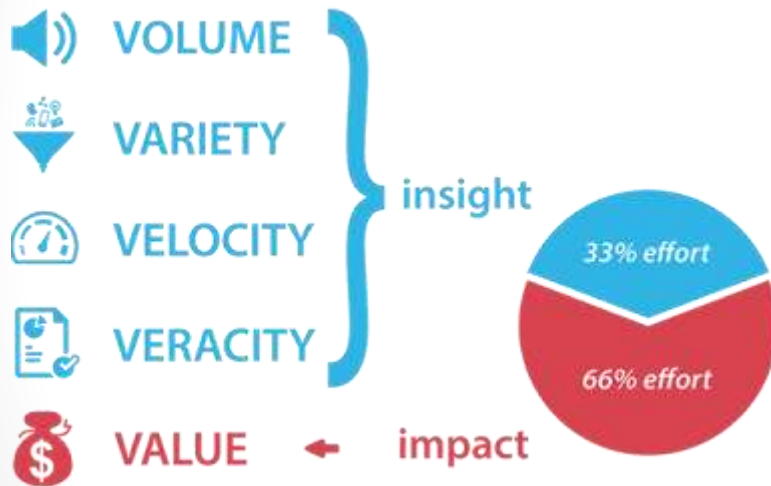
What is Big Data? Change the way we design solutions and solve problems

There is not a standard definition!

“*Big Data*” involves data whose **Volume**, **Variety** (diversity), **Velocity**, & complexity requires new techniques, algorithms and analyses to extract **valuable knowledge** (hidden).



What is Big Data? The one “V”





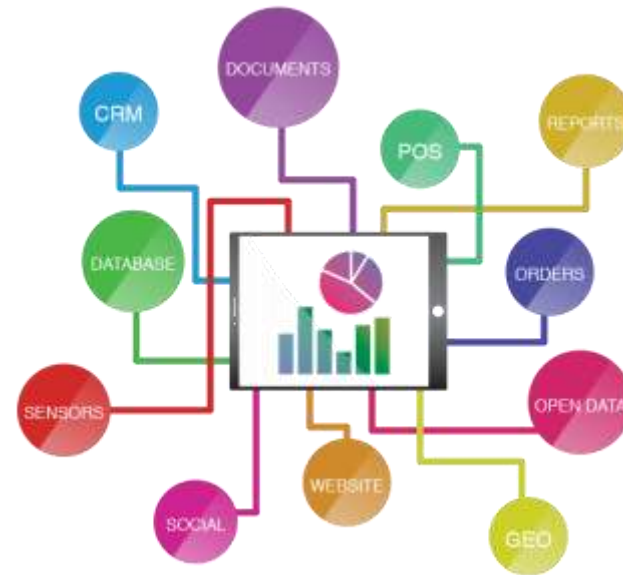
Data Science and Big Data Analytics

Word cloud comprising some important terms

Big Data Analytics → Smart Data



- While the volume, variety and velocity aspects refer to data generation process and how to capture and store the data,
- Veracity and value aspects deal with the quality and the usefulness of the data leading to the point.



Big Data Analytics → Smart Data

- Smart Data (veracity and value) aims to filter out the noise and hold the valuable data,
- It can be effectively used by enterprises and governments for planning, operation, monitoring, control, and intelligent decision making.



What makes data to be “Smart”

Accurate:

- Data must be what it says it is with enough precision to drive value. Data quality matters.

Actionable:

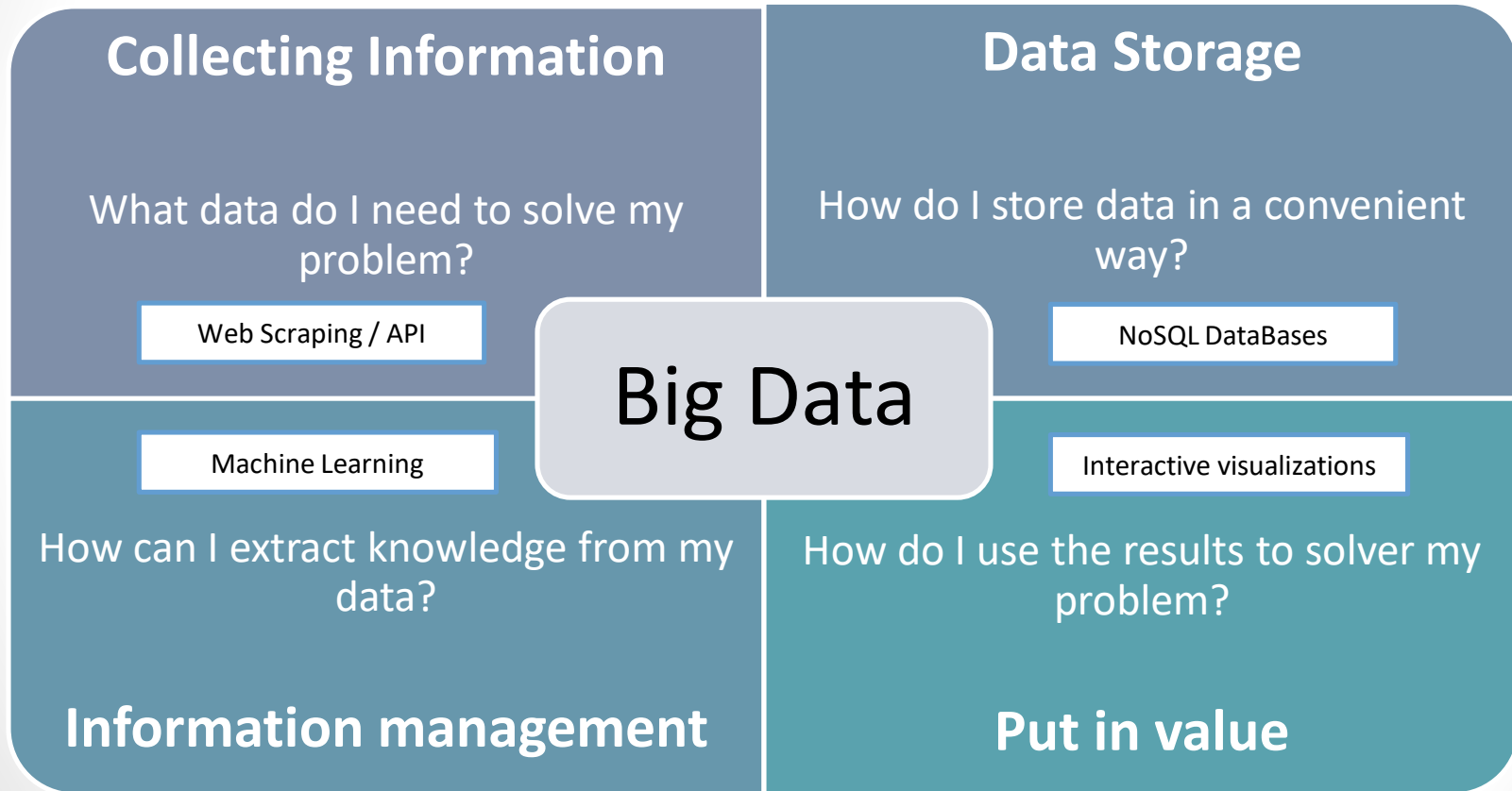
- Data must drive an immediate scalable action maximizing a business objective. Scalable action matters.

Agile:

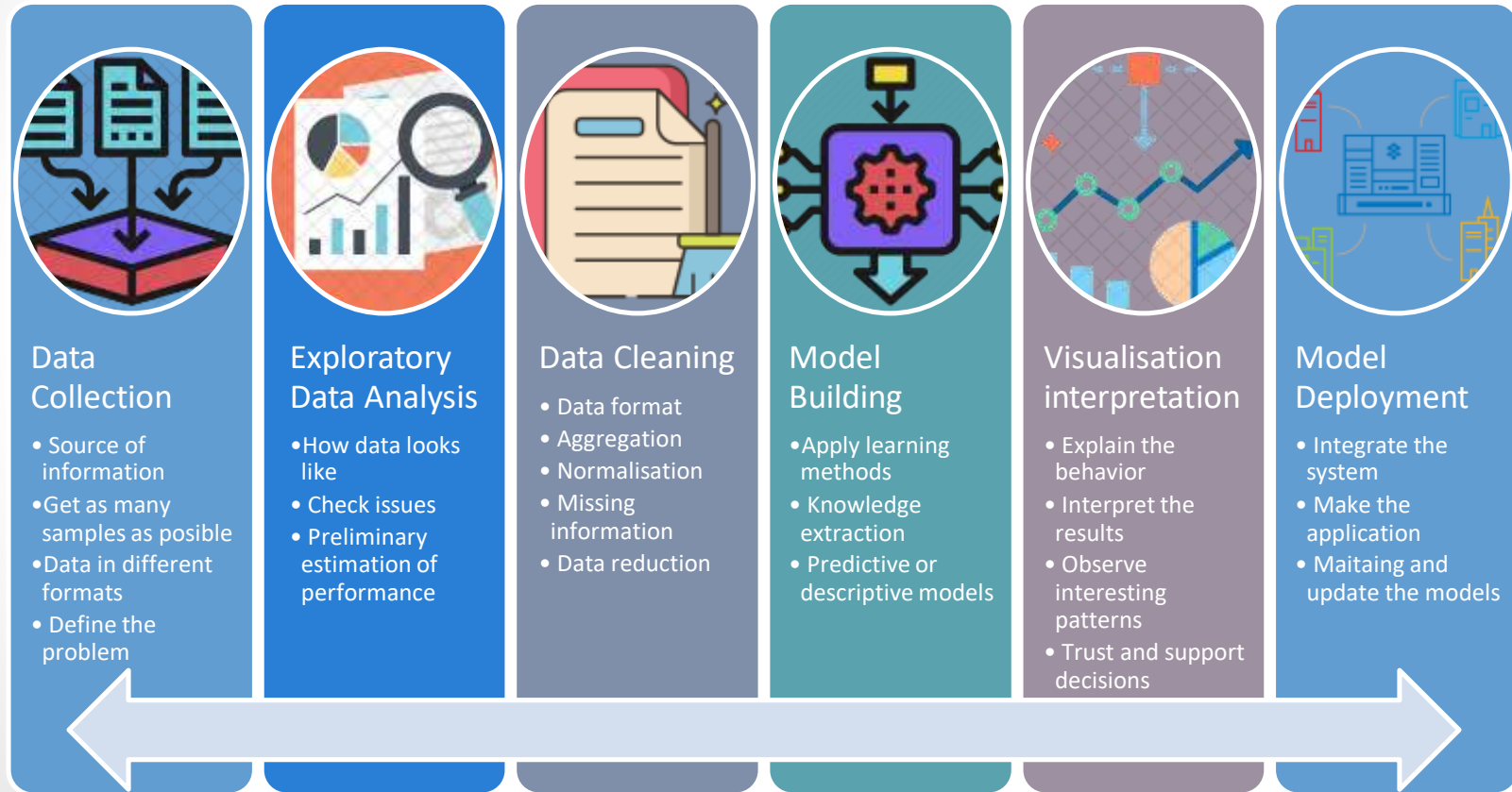
- Data must be available in real-time and ready to adapt to the changing business environment. Flexibility matters.

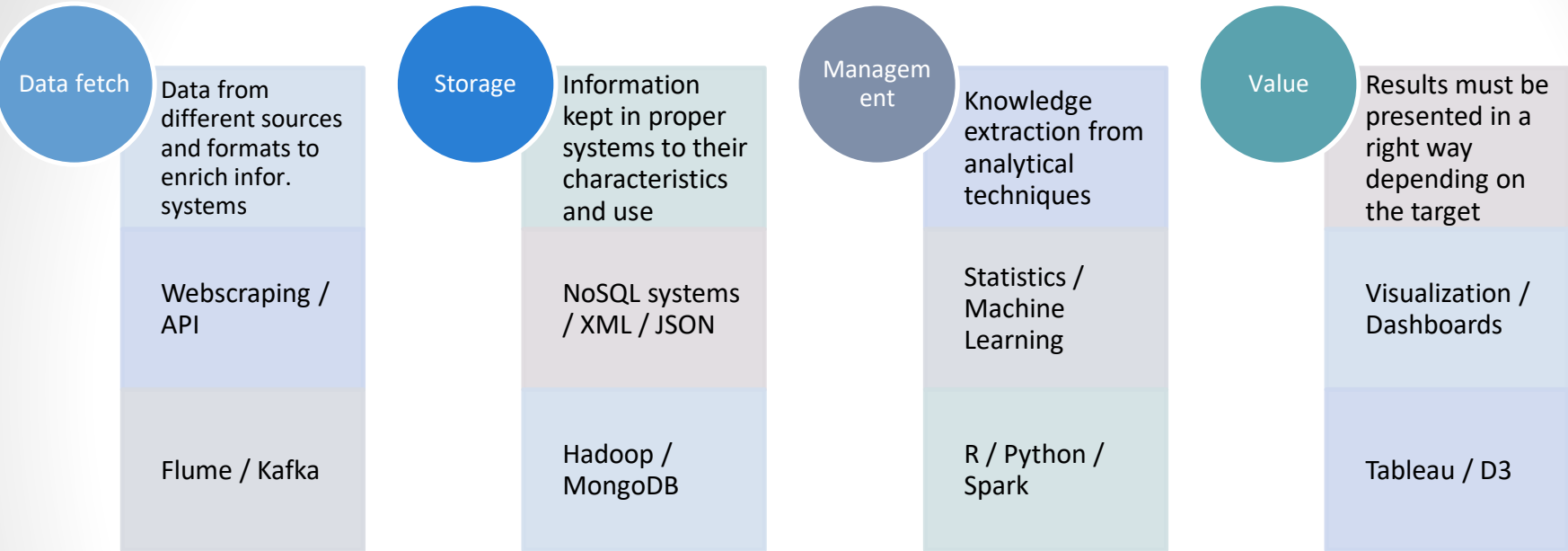


Big Data simplifies the translation from data to actionable knowledge



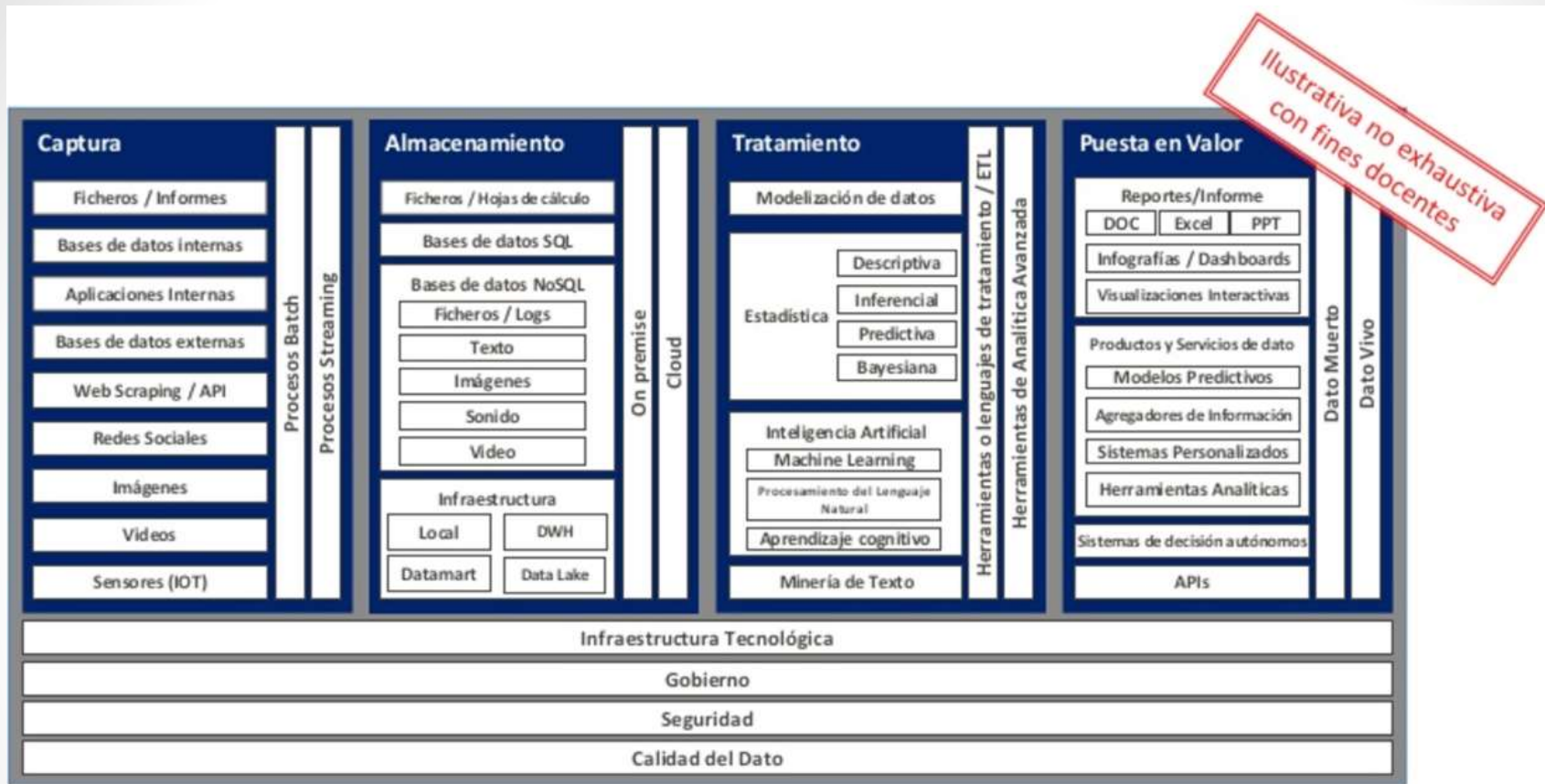
Data Science LifeCycle





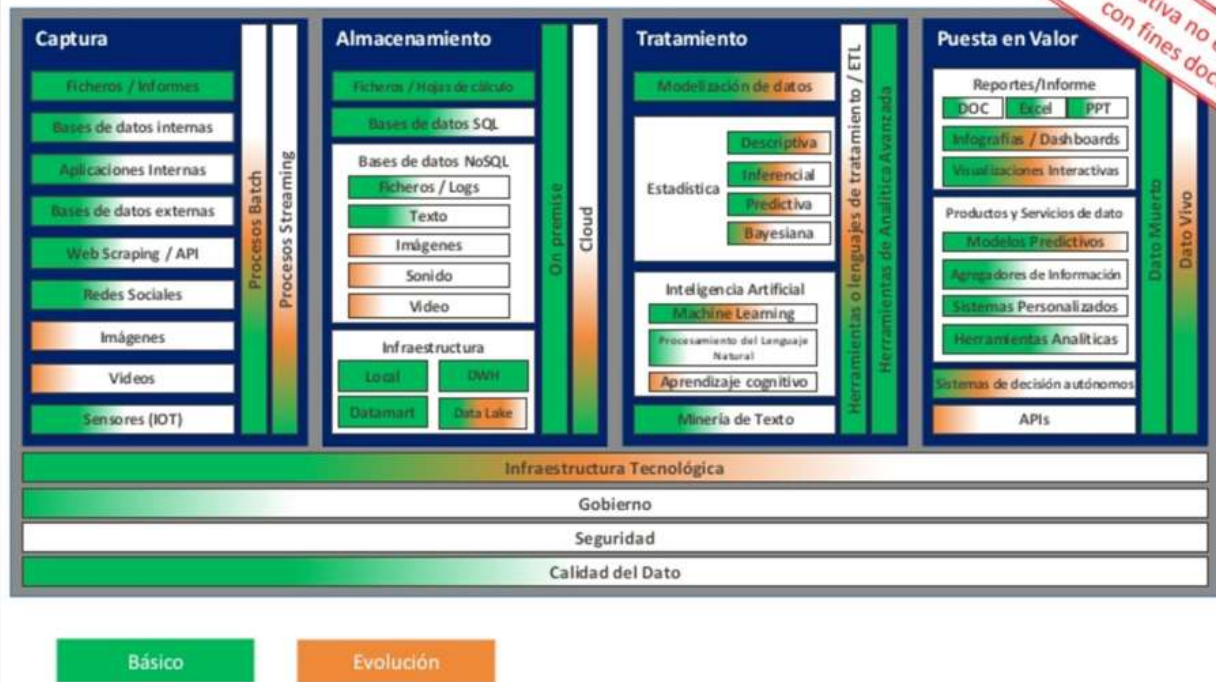
All management information procedures must consider transversal aspects such as information quality, traceability, security, privacy, among others

Examples of Tools and Technologies – A multidisciplinary approach



Information Systems: Capabilities.

Source: Antonio Pita Lozano (UNIR)



Ilustrativa no exhaustiva
con fines docentes

Capacidades

- Tratamiento de información
- Análisis de la información
- Modelos predictivos
- Soluciones basadas en datos

Tecnologías

- Lenguajes de programación
- Paquetes estadísticos
- Lenguajes de bases de datos
- Herramientas de visualización

Principales Retos

- Tener una visión global del proceso de gestión de la información desde la captura hasta la puesta en valor

Big Data: Data Scientist

Source: Antonio Pita Lozano (UNIR)

Issues when working in Big Data

Cheap, abundant storage.

Faster processors.

Affordable open source, distributed big data platforms (Hadoop).

Parallel processing, MPP, virtualization, grid environments, high throughputs.

Cloud computing and other flexible resource allocation arrangements.



INFRASTRUCTURE



ANALYTICS & MACHINE INTELLIGENCE



APPLICATIONS - ENTERPRISE



APPLICATIONS - INDUSTRY



OPEN SOURCE



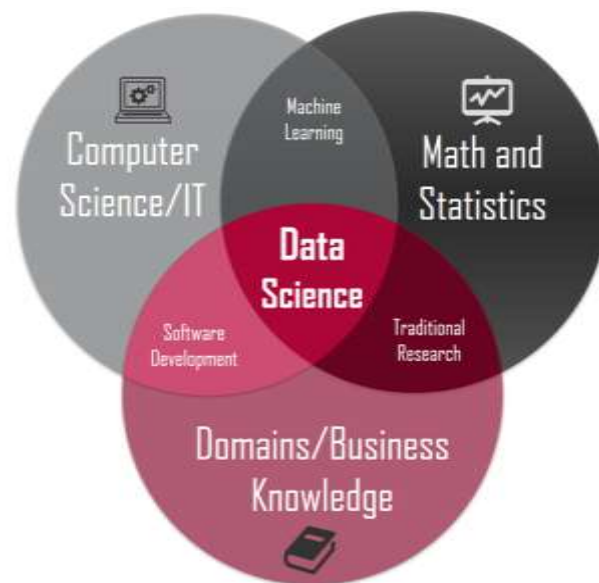
DATA SOURCES & APIs



(Big) Data Science



- Synergy of the traditional scientific method with the ability to explore, learn and obtain a deep knowledge in (Big) data
- It is not just pattern mining on data...
 - ... but mainly to explain those patterns



Outline



1

- Big Data. Big Data Science

2

- **Why Big Data? Google and the MapReduce programming model**

3

- Big Data technologies: Hadoop / Spark ecosystem

4

- Big Data Analytics: Libraries for Data Analytics in Big Data. Case studies

5

- Final Comments



Why Big Data?

The MapReduce programming model

Pulsars detection from radiofrequency by Imaging stacking

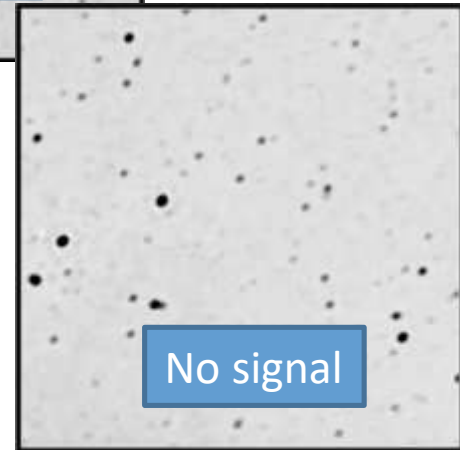
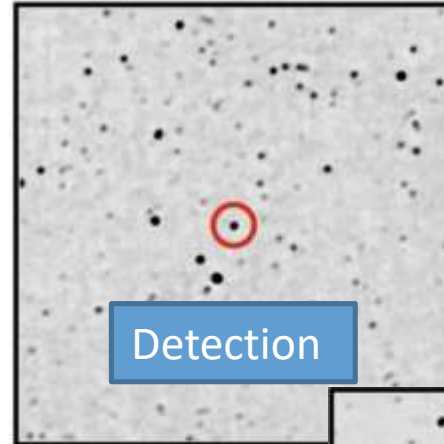
Use flux density as a measure of spectral power



Positive detection when is 5 times noise in the local radio



Aggregate regions of an image to amplify signal to noise ratio.



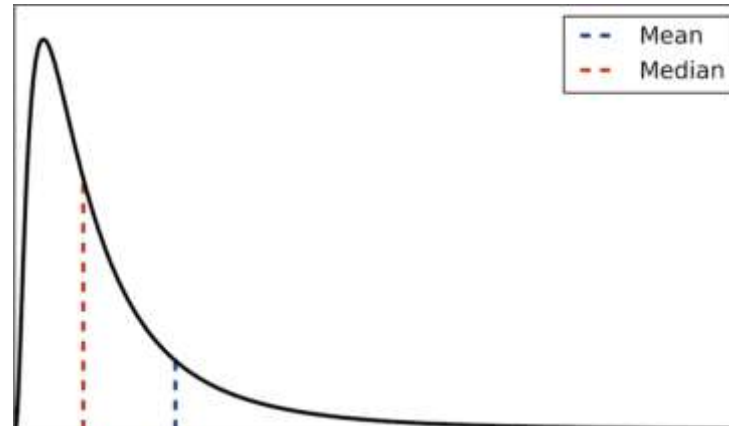
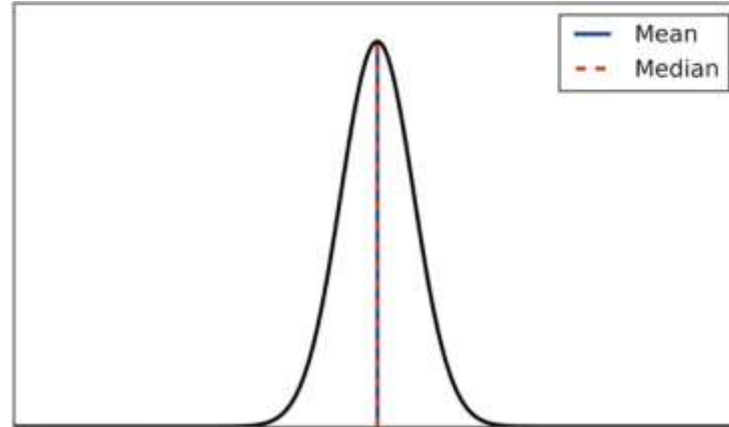
Procedure for pulsar detection: How to aggregate values?

Find images containing positions of known pulsars

Crop and shift each pulsar to the centre of each image

Aggregate brightness at each pixel

Create new image



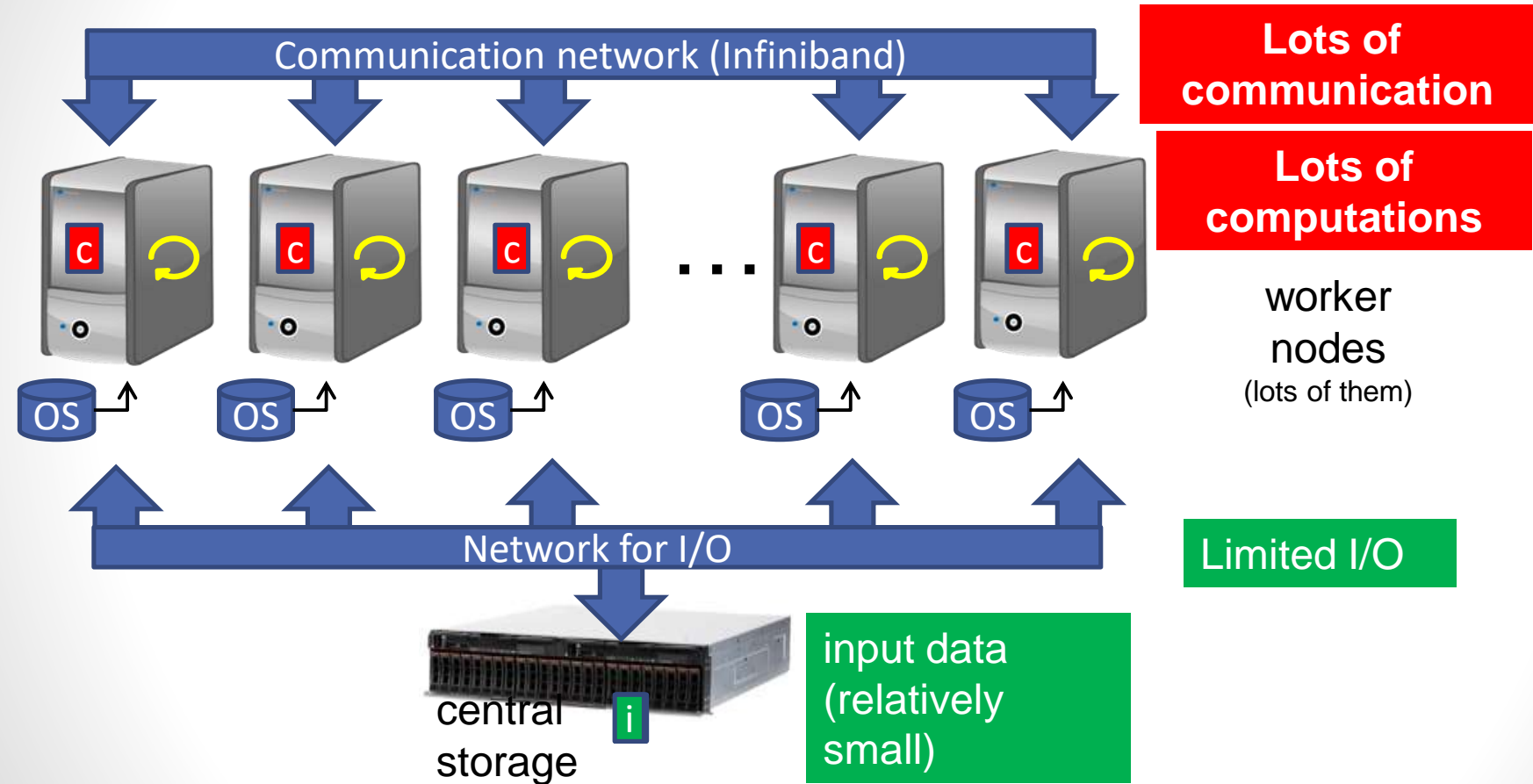
Challenge: A example on scalability.

- Compute the mean is straightforward, but does not provide accurate results
- Compute the median leads to a significant increase of computational resources. Why?
 - Let's say we start from 600,000 images
 - 1 image has a 200x200 pixel resolution --> 40,000 pixels
 - 1 pixel needs 8 bytes in memory
 - The amount of memory needed: 192 GB!!
- All data cannot be in memory at the same time. Available solutions:
 - Invest money on better equipment
 - Reformulation of the problem
 - Smart design of an scalable solution -> **Go for linear and parallel solutions**

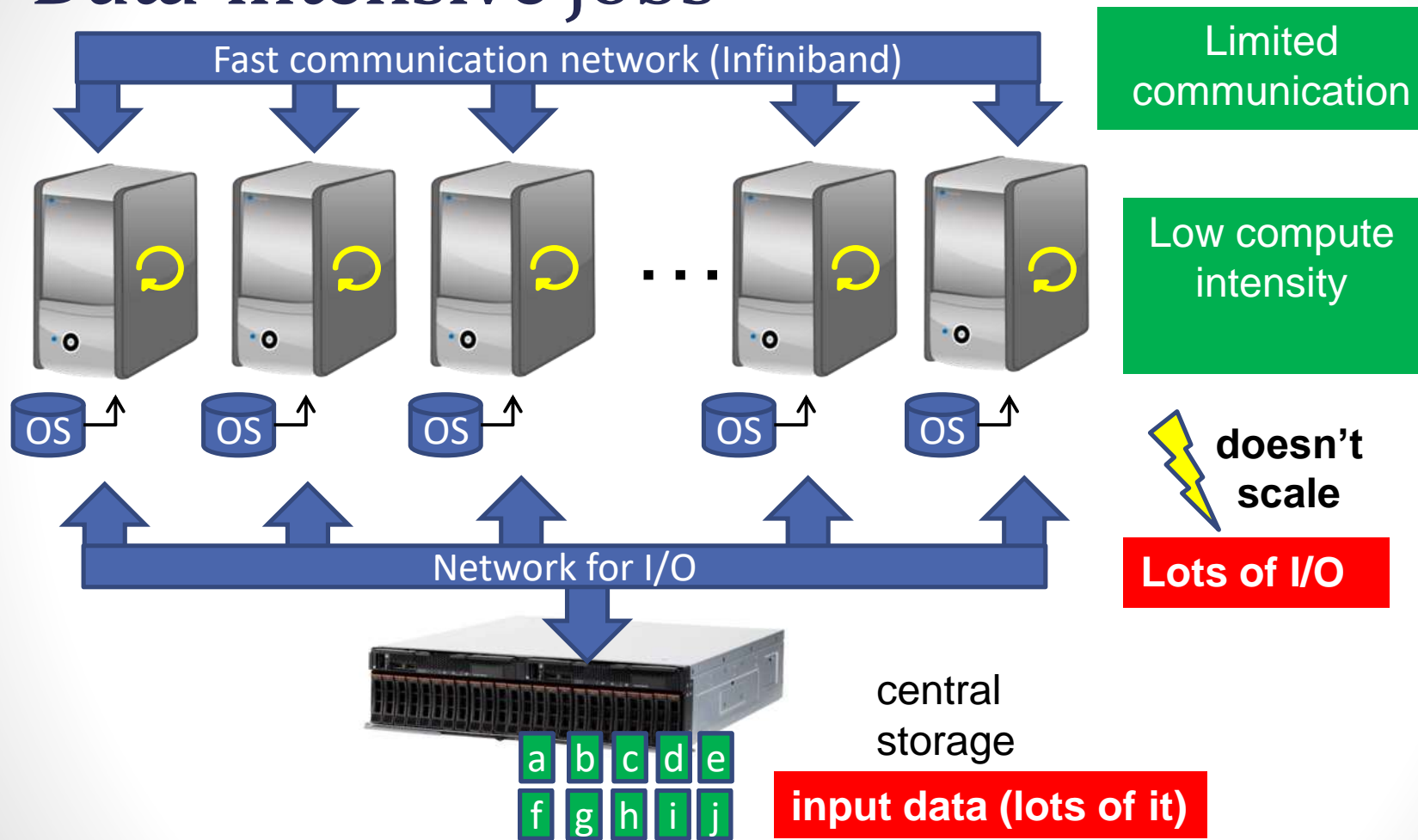
Dealing data intensive applications: Scale-up vs. Scale-out



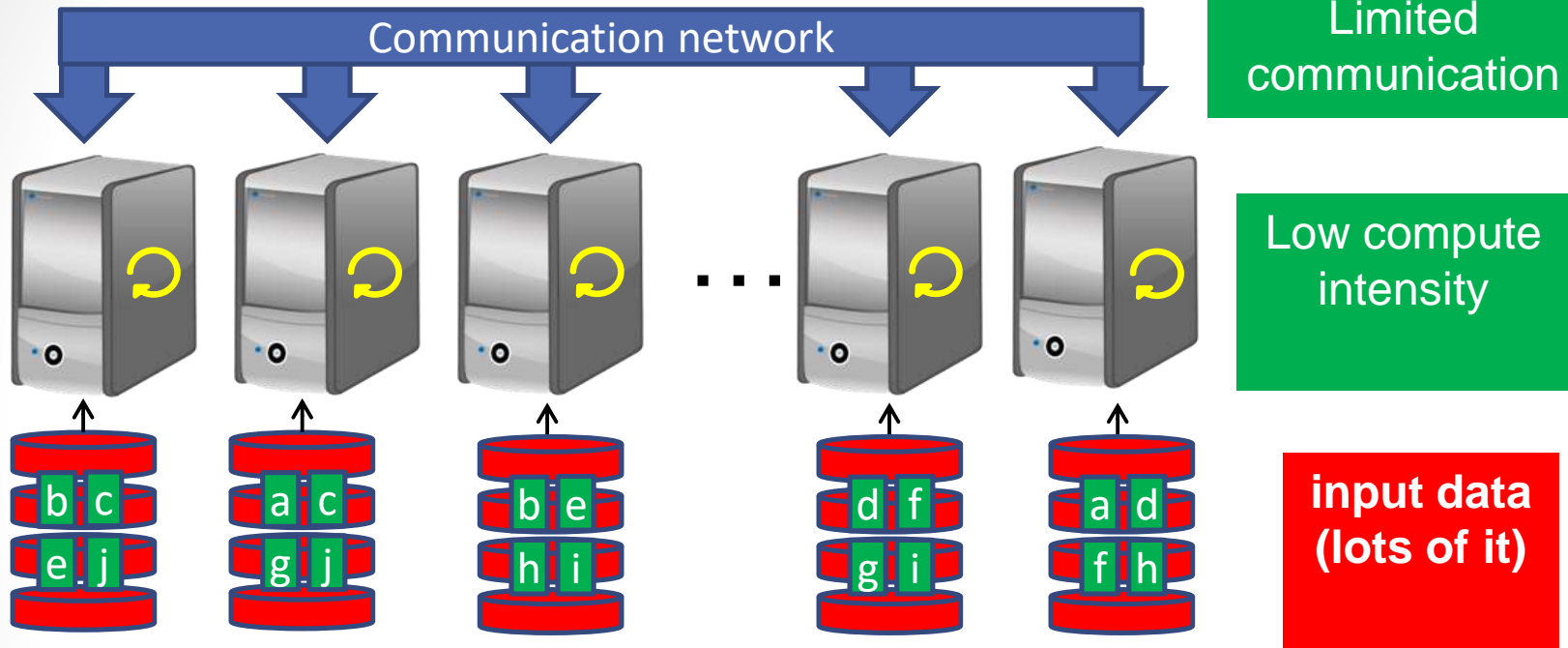
Traditional HPC way of doing things



Data-intensive jobs

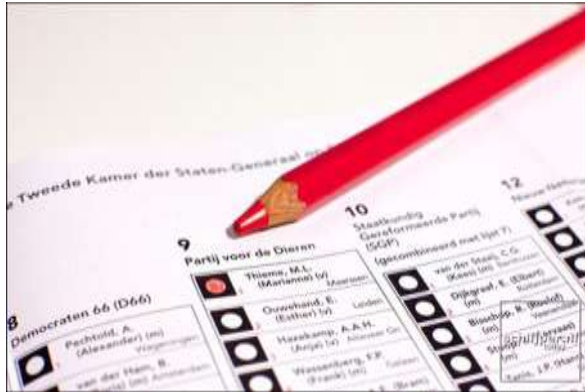
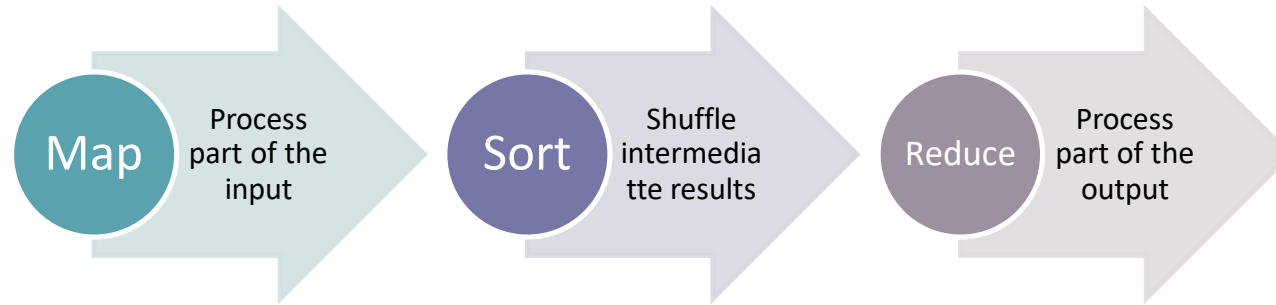


Data-intensive jobs



Solution: store data on local disks of the nodes that perform computations on that data (“**data locality**”)

We need new parallel programming paradigm



Example: vote counting



With fault tolerance

Distributed systems in Big Data.

Aim: to apply an operation to all data

- One machine cannot process or store all data
 - Data is distributed in a cluster of computing nodes
 - It does not matter which machine executes the operation
 - It does not matter if it is run twice in different nodes (due to failures or straggler nodes)
 - We look for an abstraction of the complexity behind distributed systems
- **DATA LOCALITY is crucial**
 - Avoid data transfers between machines as much as possible

Distributed systems in Big Data

New programming model: MapReduce

- *“Moving computation is cheaper than moving computation and data at the same time”*
- **Idea**
 - Data is distributed among nodes (distributed file system)
 - Functions/operations to process data are distributed to all the computing nodes
 - Each computing node works with the data stored in it
 - Only the necessary data is moved across the network

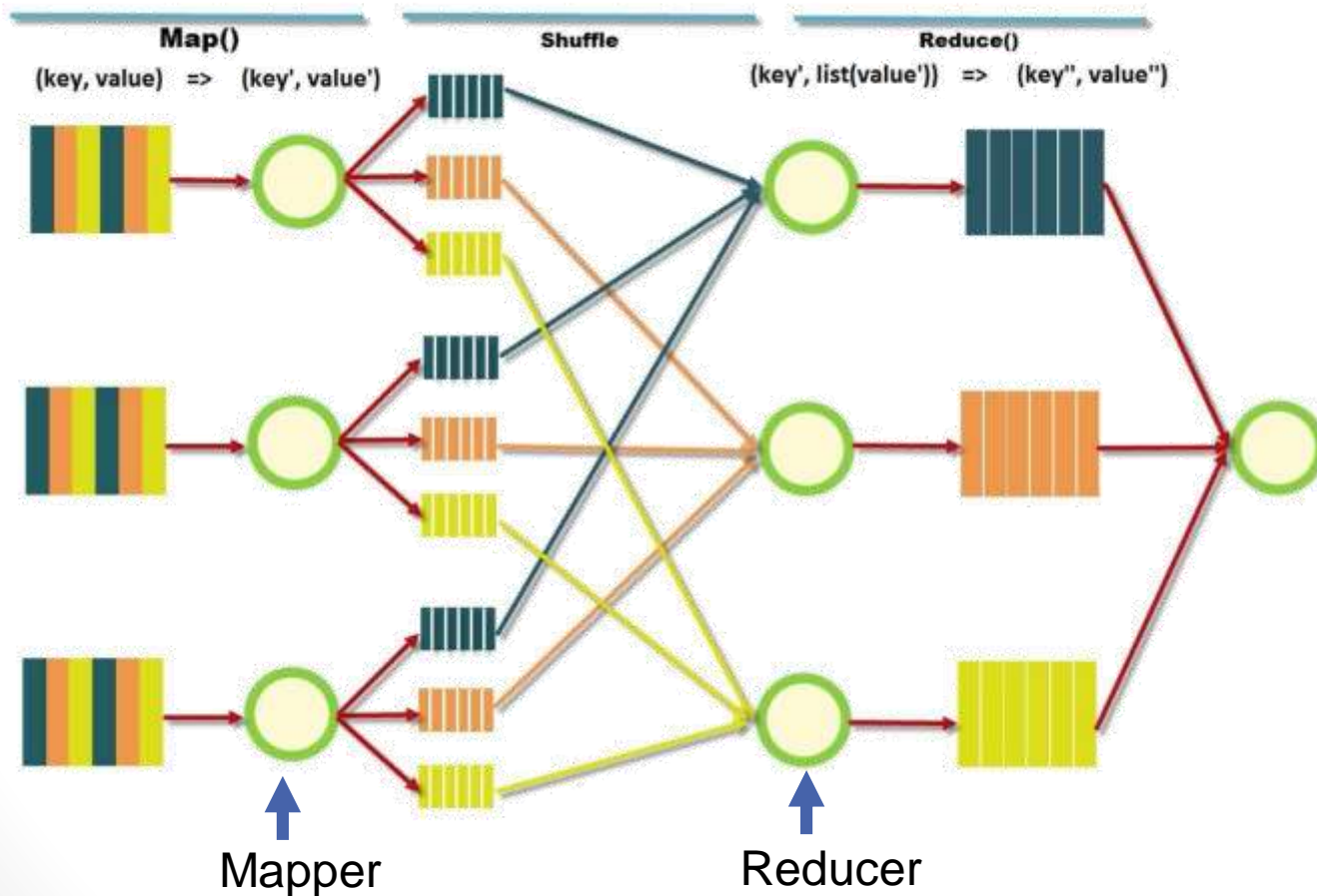
Hadoop Distributed File System: HDFS

- Distributed File System written in Java
- Scales to clusters with **thousands of computing nodes**
 - Each node stores part of the data in the system
- **Fault tolerant** due to data replication
- Designed for big files and low-cost hardware
 - GBs, TBs, PBs..
- **Efficient for read and append operations** (random updates are rare). **Overhead** when writing into disk.
- High throughput (for bulk data) more important than low latency

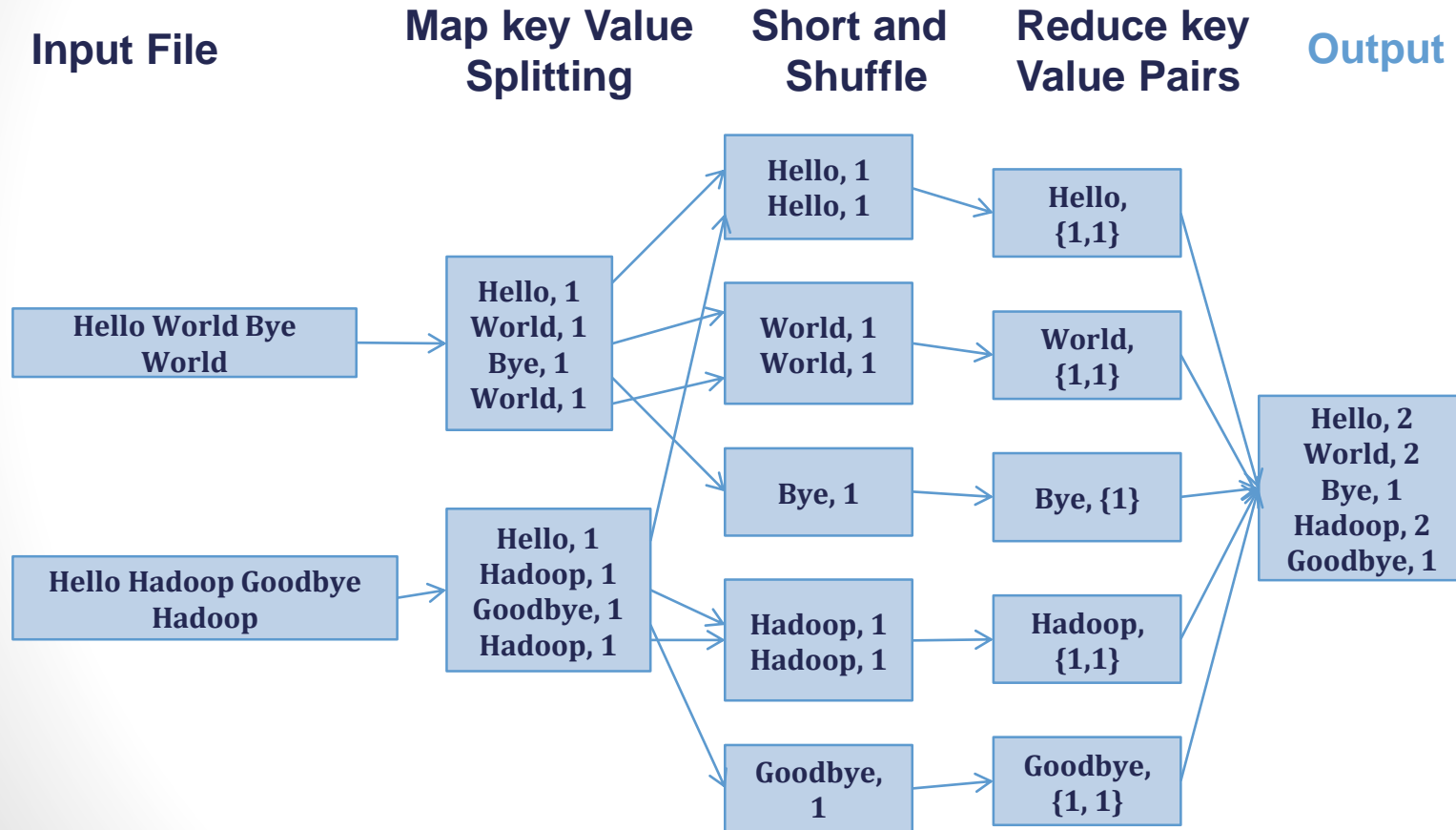
MapReduce

- Parallel Programming model
- **Divide & conquer strategy**
 - **divide**: partition dataset into smaller, independent chunks to be processed in parallel (map)
 - **conquer**: combine, merge or otherwise aggregate the results from the previous step (reduce)
- Based on **simplicity** and **transparency** to the programmers, and assumes **data locality**
- Becomes popular thanks to the open-source project Hadoop! (Used by [Google](#), [Facebook](#), [Amazon](#), ...)

MapReduce: How it works



MapReduce: WordCount Example





MapReduce: Features

	Automatic parallelization:	<ul style="list-style-type: none">• Size of the INPUT DATA → multiple MAP tasks• Number of <key, value> intermediate partitions → several REDUCE tasks
	Scalability:	<ul style="list-style-type: none">• It works over any cluster of nodes /processor• Can work from 2 to 10,000 machines
	Programming transparency	<ul style="list-style-type: none">• Management of the failures of the machine• Management of the communication among machines

MapReduce

Main Features

- Scalable architectures
- Optimized planning
- Elasticity and availability
- Flexibility
- Security and authentication

Limitations

- Machine Learning: Iterative computation
- Graph processing
- Real time processing (streams)
- Functionality of process communication
- Difficultiy in the MR-like implementation

Outline



1

- Big Data. Big Data Science

2

- Why Big Data? Google and the MapReduce programming model

3

- **Big Data technologies: Hadoop / Spark ecosystem**

4

- Big Data Analytics: Libraries for Data Analytics in Big Data. Case studies

5

- Final Comments



Big Data Technologies

Hadoop Ecosystem

Hadoop Ecosystem:

Apache Hadoop Modules



<http://hadoop.apache.org/>

Hadoop Common:

Utilities
supporting
other
Hadoop
modules.

Hadoop Distributed File System (HDFS):

The file
system that
grant
access

Hadoop YARN:

Framework for management the programming
resources.

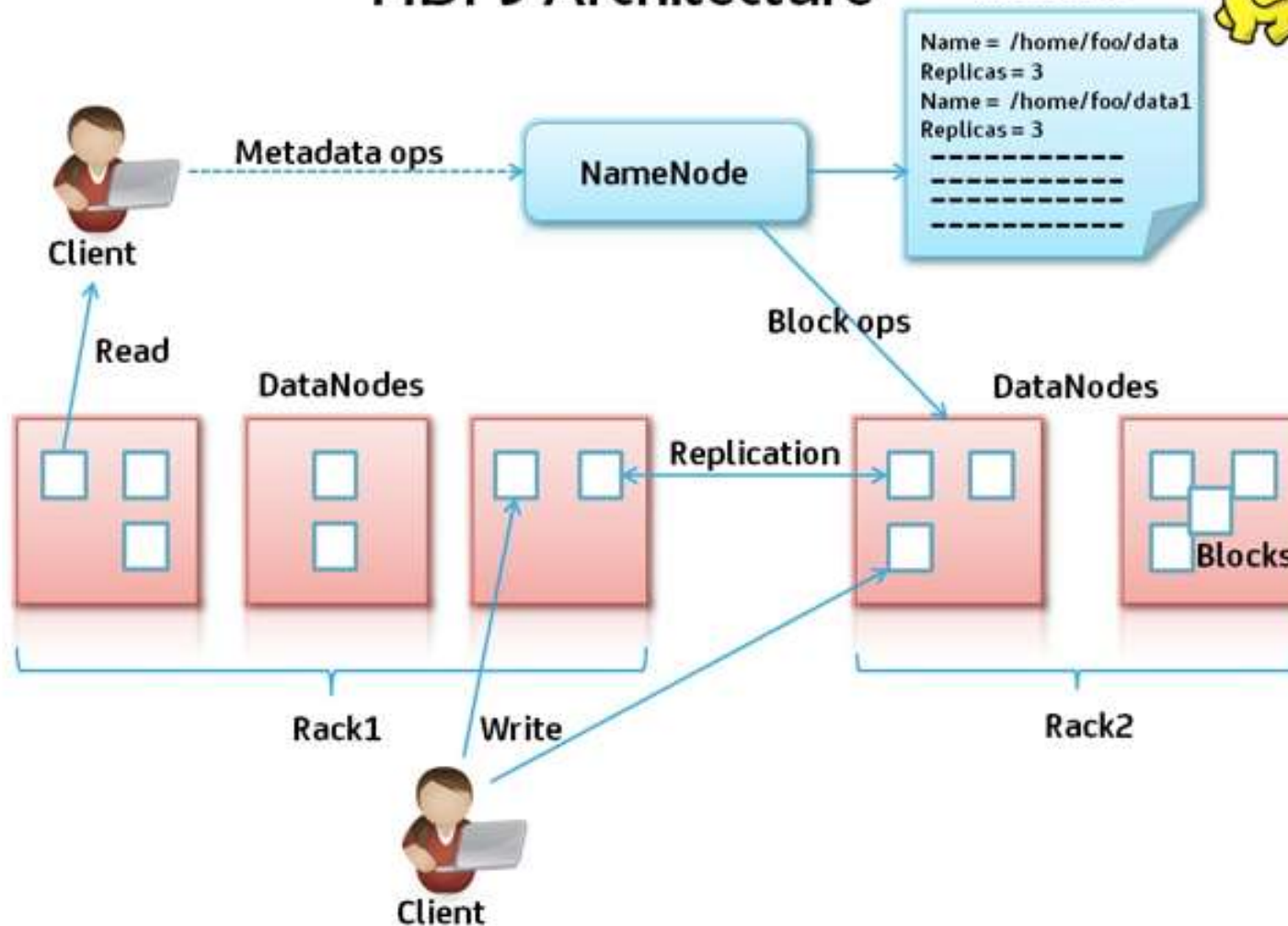


Hadoop Ecosystem

Complete technologies to address Big Data Analytics

HDFS Architecture

Metadata



File Management with HDFS

- Different from the traditional local system.
- It is mounted on a special system.
- Directory structure in HDFS is as follows:
 - /tmp temporal storage
 - /user user storage
 - /usr storage for applications, etc.
 - /var logs for applications, etc.

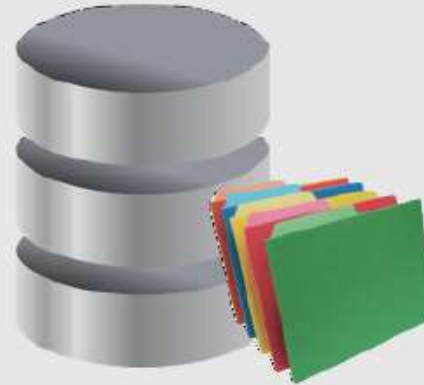
File management with HDFS

Careful!! The storage space in HDFS is different to that of the user.

user space
`/user/mbdx506000265/`

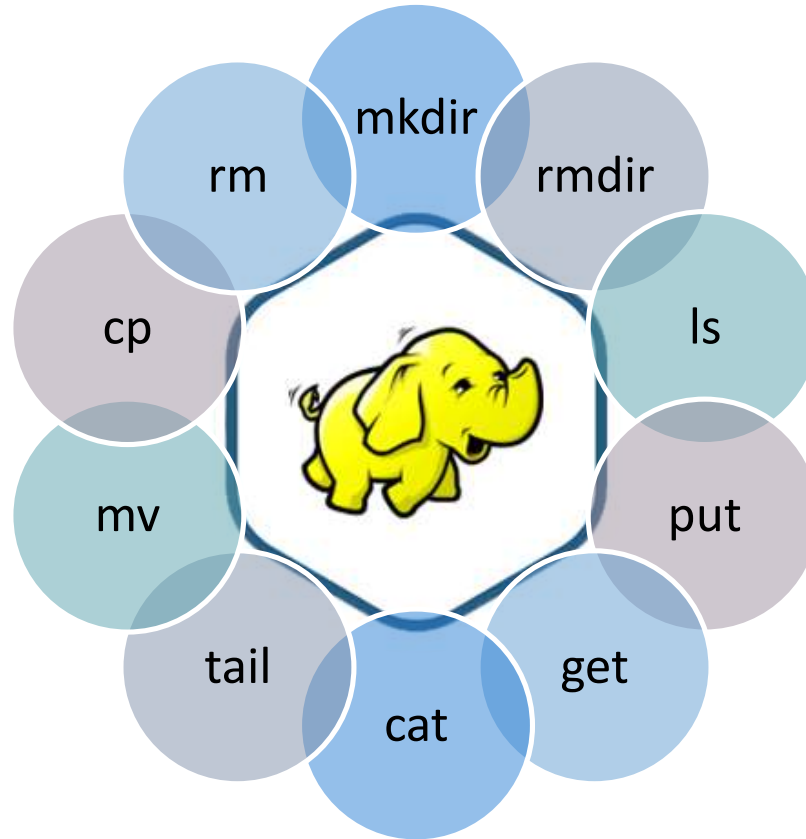


`/home/mbdx50600265/`

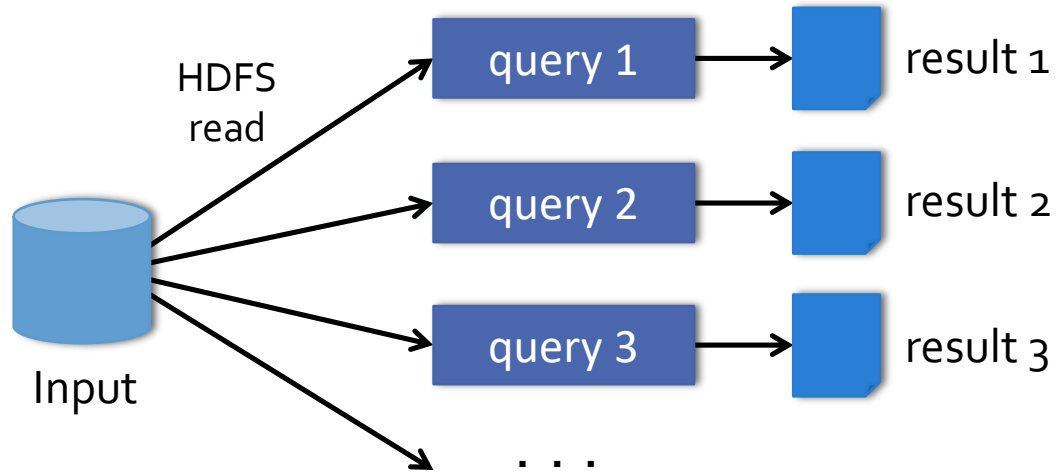
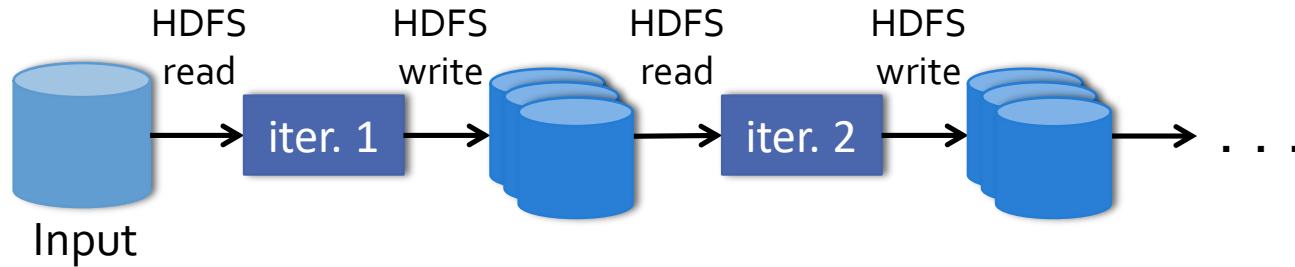


File management with HDFS.

Command “hdfs dfs -<option>”



Data Sharing in MapReduce



Slow due to replication, serialization, and disk IO

Apache Spark (Birth 2009-2010)



Fast and Expressive Cluster Computing
Engine Compatible with Apache Hadoop

Up to **10x** faster on disk,
100x in memory

Efficient

- General execution graphs
- In-memory storage

2-5x less code

Usable

- Rich APIs in Java, **Scala**, Python
- Interactive shell

What is Spark?



Data processing engine (only)

Without a distributed file system

- Uses other existing DFS
 - HDFS, NoSQL...
 - Hadoop is not a prerequisite

Works with different cluster management tools

- Hadoop (YARN)
- Mesos
- Standalone mode (included in Spark)

What is Spark?



Spark SQL
structured data

Spark Streaming
real-time

MLib
machine
learning

GraphX
graph
processing

Spark Core

Standalone Scheduler

YARN

Mesos

Spark Goal

Provide distributed memory abstractions for clusters to support apps with working sets

Retain the attractive properties of MapReduce:

- Fault tolerance (for crashes & stragglers)
- Data locality
- Scalability

Initial Solution: augment data flow model with “resilient distributed datasets” (RDDs)

Apache Spark

- KEY Concept: RDD (Resilient Distributed Datasets)
 - Fault-tolerant collection of elements that can be operated on in parallel.
- There are two ways to create RDDs:
 - Parallelizing an existing collection in your driver program
 - Referencing a dataset in an external storage system, such as a shared filesystem, HDFS, Hbase.
- Objects spread across a cluster, stored in RAM or on Disk
- *Can be cached for future reuse*

Apache Spark: Operations

Transformations

Create a new RDD / dataset from an existing one

Lazy in nature: executed only when some action is performed

Example

- Map(func)
- Filter(func)
- Distinct()

Actions

Returns a value or exports data after performing a computation

Example

- Count()
- Reduce(func)
- Collect()

Persistence

Caching RDD / dataset in-memory for future operations

Store on disk or RAM or mixed

Example

- Persist()
- Cache

RDDs operations. Word Count example

```
>>> lines = sc.textFile("README.md") # Creates an RDD
>>> lines.count() # Counts the number of elements in the
RDD
```

```
127
```

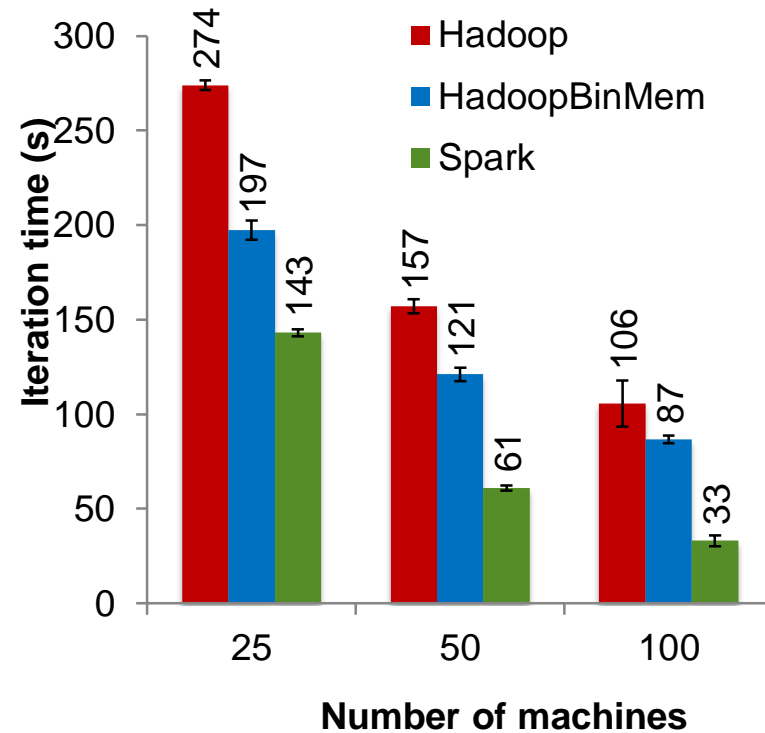
```
>>> lines.first() # First element of the RDD -> 1st line
of README.md
```

u'# Apache Spark'

```
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" "))
                    .map(lambda word: (word, 1))
                    .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```


Spark vs. Hadoop

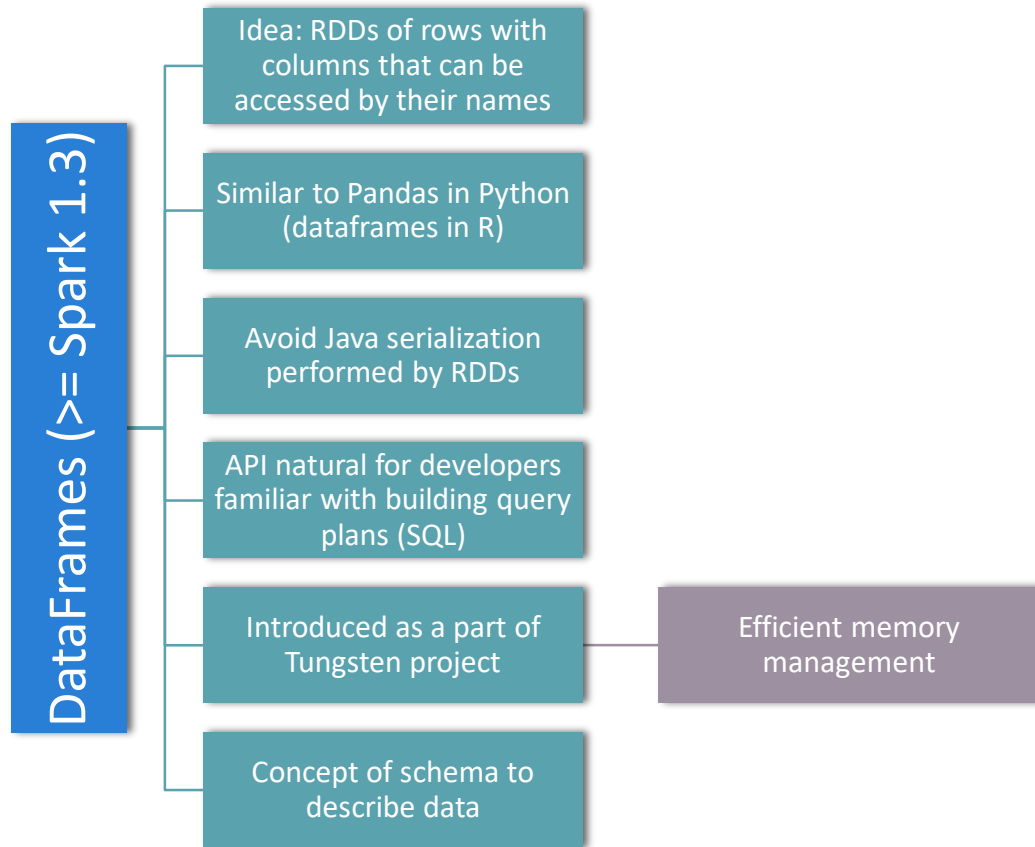
- **Lines of code for K-Means**
 - Spark ~ 90 lines –
 - Hadoop ~ 4 files, > 300 lines



M. Zaharia et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. NSDI 2012.

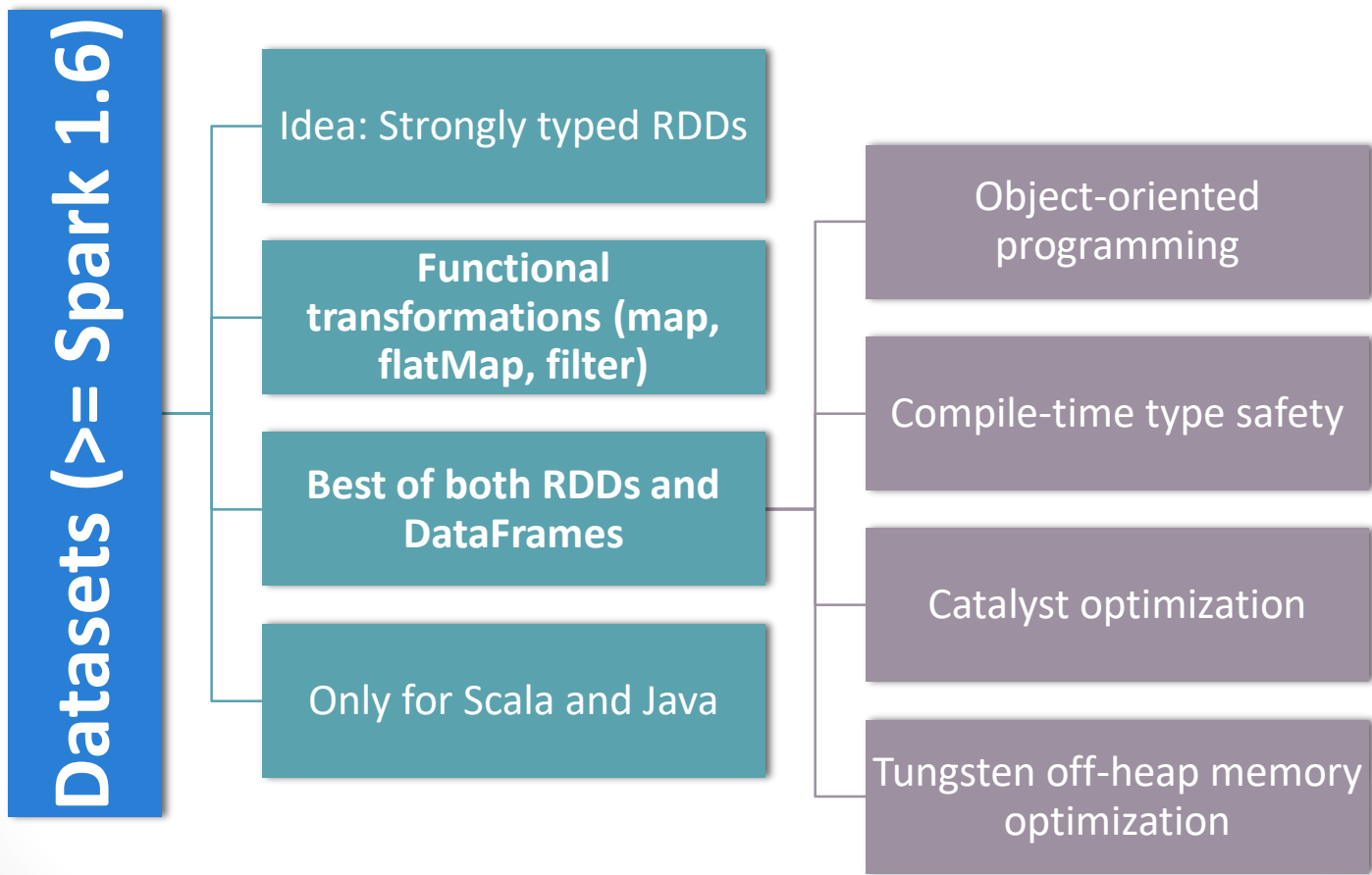
SparkSQL: Datasets & DataFrames

Structured APIs



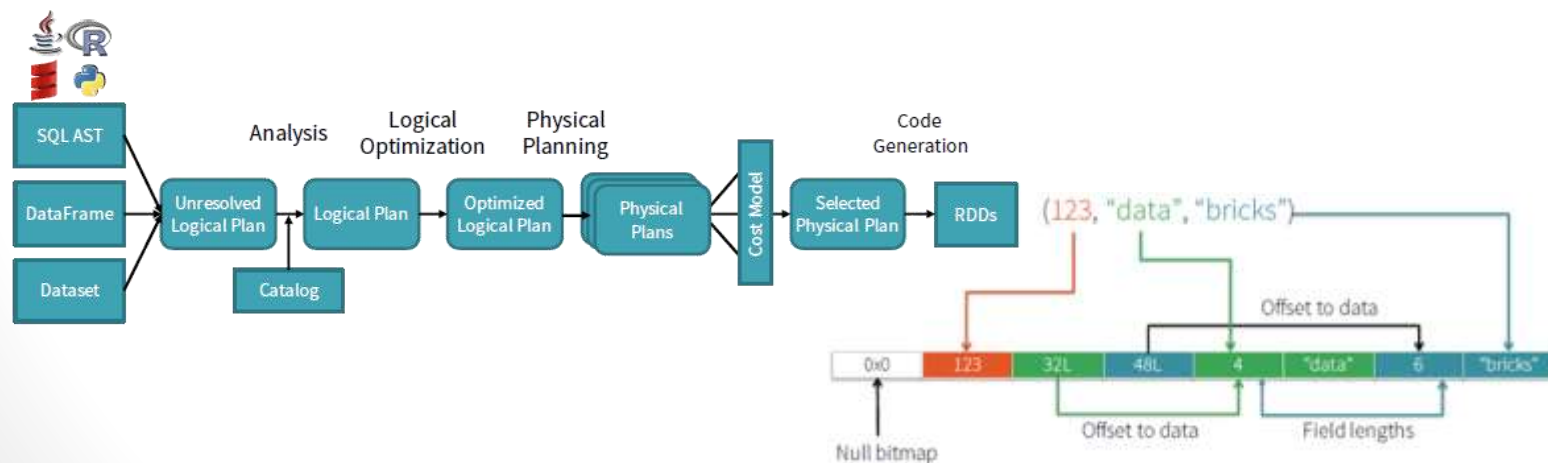
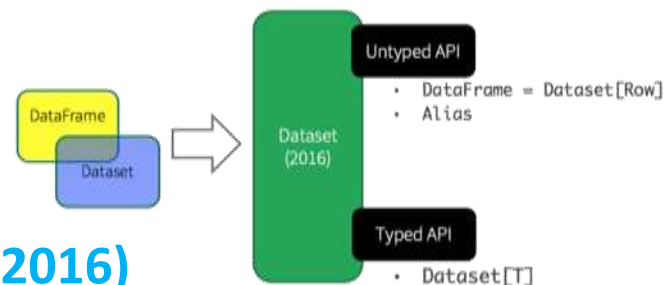
SparkSQL: Datasets & DataFrames

Structured APIs



SparkSQL: Datasets & DataFrames

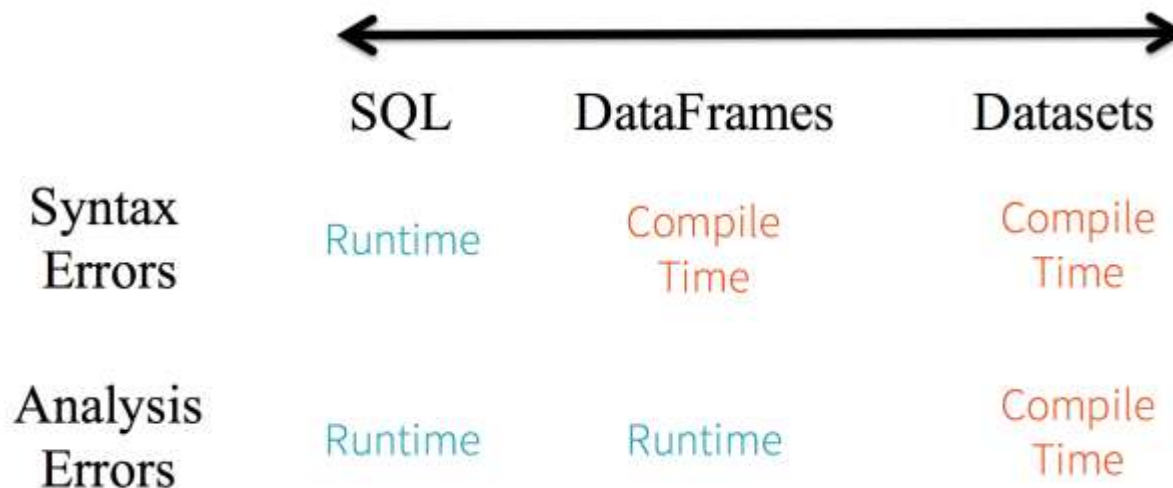
- **Structured APIs**
 - **DataFrames and Datasets**
 - **Fused in Spark 2.0 (November 2016)**
 - A DataFrame is just a Dataset of Rows: Dataset[Row]
 - Both make use of Catalyst and Tungsten projects



SparkSQL: Datasets & DataFrames

- **Structured APIs in Spark**

- Analysis of the **reported errors** before a job is executed



Current Spark Environment

Spark has released version 3.0

- June 2020: 10 year anniversary!
- Spark 3.0 is said to be 2 times faster than Spark 2.4
- SparkSQL is the main component for the update: APIs for DataFrames.
- PySpark has become the widest used language:
 - Improvement of functionality and usability
 - Pandas UDF redesign and types
 - Error handling

Feature highlights

- Adaptive query execution;
- Dynamic partition pruning;
- ANSI SQL compliance;
- Significant improvements in pandas APIs;
- New UI for structured streaming;
- Up to 40x speedups for calling R user-defined functions;
- Accelerator-aware scheduler;
- SQL reference documentation.

<https://spark.apache.org/releases/spark-release-3-0-0.html>

Which Language Should I Use?

- Standalone programs can be written in any, but console is only Python & Scala
- Python developers: can stay with Python for both
- Java developers: consider using Scala for console (to learn the API)
- **Performance:** Java / Scala will be faster (statically typed), but Python can do well for numerical work with NumPy
- **Effectiveness:** Scala and python code is less lengthy than java. Scala has faster performance.
- **Comfort:** Moving from one language to another?
- **Future Prospects:** Many supported languages, Scala keeps being the main one.



How do I get access to a Big Data platform?

Cloud platforms with
complete IaaS / PaaS



Google Cloud Platform

Pseudo-distributed installation

Pre-compiled Hadoop and
Spark stand-alone

Installation in a cluster

cloudera



kubernetes



OpenNebula

MESOS

Outline



1

- Big Data. Big Data Science

2

- Why Big Data? Google and the MapReduce programming model

3

- Big Data technologies: Hadoop / Spark ecosystem

4

- **Big Data Analytics: Libraries for Data Analytics in Big Data. Case studies**

5

- Final Comments



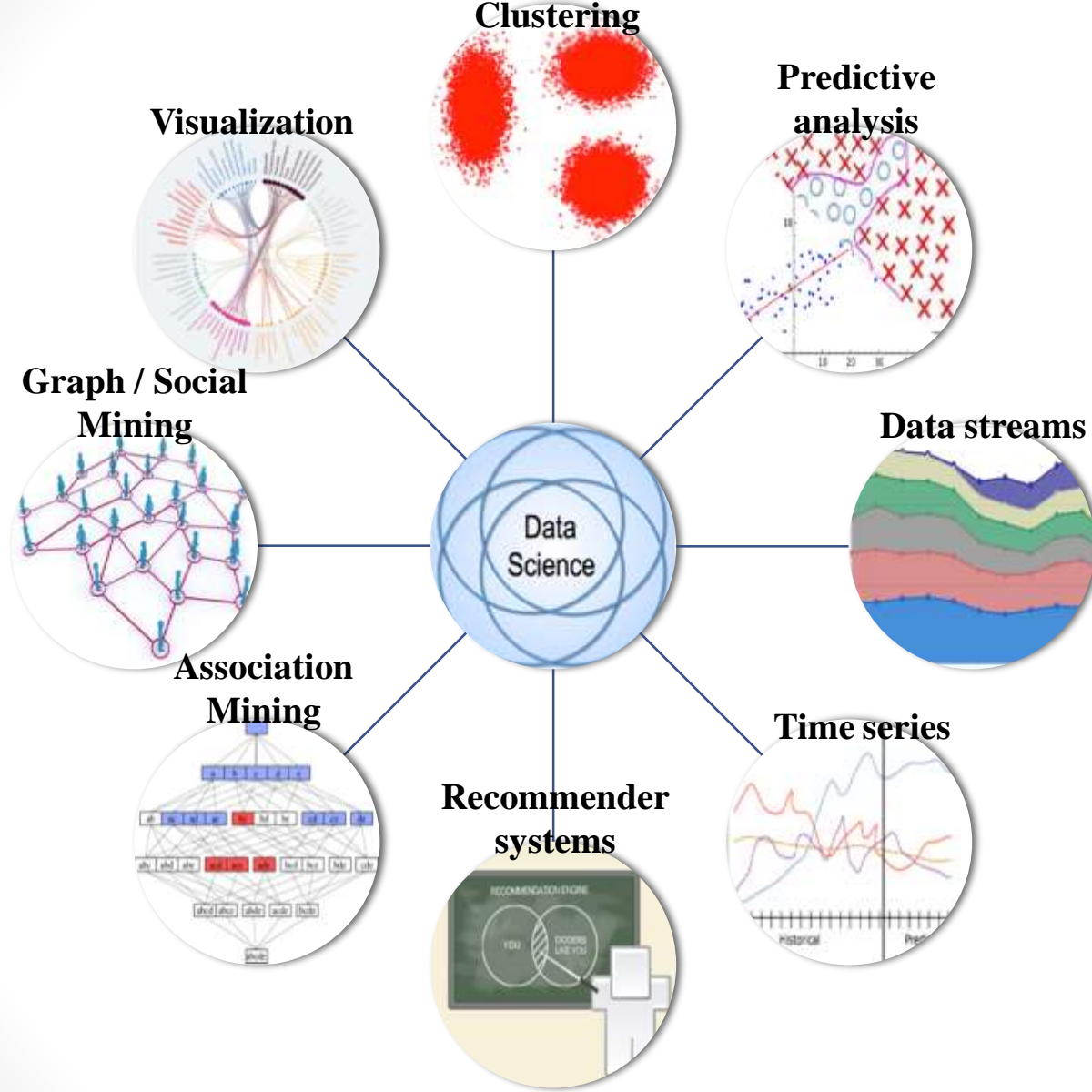
Big Data Analytics

Machine Learning tools and case studies

Evolution of the popularity of Big Data and related concepts



<https://trends.google.com/>



Machine Learning for Big Data

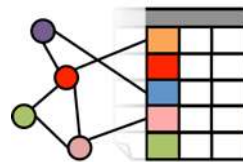
- Data mining techniques have demonstrated to be very useful tools to extract new valuable knowledge from data.
- Knowledge extraction process from big data has become a very difficult task for most of the data mining tools.
- The main challenges are to deal with:
 - The increasing scale of data
 - at the level of instances
 - at the level of features
 - The complexity of the problem.
 - ...and many other points (unstructured data, redundancy, ...)

Big Data Analytics Tools: Spark MLlib

- Goal: make practical Machine Learning scalable and easy.
- At a high level, it provides tools such as:
 - **ML Algorithms**: common learning algorithms such as classification, regression, clustering, and collaborative filtering
 - **Featurization**: feature extraction, transformation, dimensionality reduction, and selection
 - **Pipelines**: tools for constructing, evaluating, and tuning ML Pipelines
 - **Persistence**: saving and load algorithms, models, and Pipelines
 - **Utilities**: linear algebra, statistics, data handling, etc.



Spark Libraries



GraphX



Spark
MLlib

Spark SQL



Spark

MLlib: Main Guide

- Basic statistics
- Data sources
- Pipelines
- Extracting, transforming and loading data
- Classification and Regression
- Clustering
- Collaborative filtering
- Frequent Pattern Mining
- Model selection and tuning
- Advanced topics

MLlib: RDD-based API

- Data types
- Basic statistics
- Classification and regression
- Collaborative filtering
- Clustering
- Dimensionality reduction
- Feature extraction and transformation
- Frequent pattern mining
- Evaluation metrics
- PMML model export
- Optimization (developer)

Machine Learning Library (MLlib) Guide

MLlib is Spark's machine learning (ML) library. Its goal is to make practical machine learning scalable and easy. At a high level, it provides tools such as:

- ML Algorithms: common learning algorithms such as classification, regression, clustering, and collaborative filtering
- Featurization: feature extraction, transformation, dimensionality reduction, and selection
- Pipelines: tools for constructing, evaluating, and tuning ML Pipelines
- Persistence: saving and load algorithms, models, and Pipelines
- Utilities: linear algebra, statistics, data handling, etc.

Announcement: DataFrame-based API is primary API

The MLlib RDD-based API is now in maintenance mode.

As of Spark 2.0, the [RDD](#)-based APIs in the `spark.mllib` package have entered maintenance mode. The primary Machine Learning API for Spark is now the [DataFrame](#)-based API in the `spark.ml` package.

What are the implications?

- MLlib will still support the RDD-based API in `spark.mllib` with bug fixes.
- MLlib will not add new features to the RDD-based API.
- In the Spark 2.x releases, MLlib will add features to the DataFrames-based API to reach feature parity with the RDD-based API.

Why is MLlib switching to the DataFrame-based API?

- DataFrames provide a more user-friendly API than RDDs. The many benefits of DataFrames include Spark Datasources, SQL/DataFrame queries, Tungsten and Catalyst optimizations, and uniform APIs across languages.
- The DataFrame-based API for MLlib provides a uniform API across ML algorithms and across multiple languages.
- DataFrames facilitate practical ML Pipelines, particularly feature transformations. See the [Pipelines guide](#) for details.

What is "Spark ML"?

- "Spark ML" is not an official name but occasionally used to refer to the MLlib DataFrame-based API. This is majorly due to the `org.apache.spark.ml` Scala package name used by the DataFrame-based API, and the "Spark ML Pipelines" term we used initially to emphasize the pipeline concept.

Spark ML (Mllib): Basic Statistics

- Calculation of descriptive statistics: mean, variance, max, min, etc.
- Computation of the degree of correlation among variables.
- Stratified sampling, through two methods `sampleByKey` and `sampleByKeyExact`.
- Hypothesis contrast, for example through the Chi-square test.
- Generation of random data following a certain distribution, Normal or Poisson, for example.

Spark ML (Mllib): Classification and Regression

Classification models

- Simple, binomial, and multinomial logistic regression
- Random forest classifier
- Gradient-boosted tree classifier
- Multilayer perceptron classifier
- Linear Support Vector Machine
- One-vs-Rest classifier (a.k.a. One-vs-All)
- Naive Bayes
- Factorization machines classifier

Regression models

- Simple and Generalised Linear regression
- Decision tree regression
- Random forest regression
- Gradient-boosted tree regression
- Survival regression
- Isotonic regression
- Factorization machines regressor

Spark ML (Mllib): Unsupervised

Clustering algorithms

- K-means
- Latent Dirichlet allocation (LDA)
- Bisecting k-means
- Gaussian Mixture Model (GMM)
- Power Iteration Clustering (PIC)

Frequent pattern mining

- FP-Growth
- PrefixSpan

spark-packages.org

- External, community-managed list of third-party libraries, add-ons, and applications that work with Apache Spark.

Infrastructure Projects

- Zeppelin - Multi-purpose notebook supports 20+ language backends, including Apache Spark
- MLflow - Open source platform to manage the ML lifecycle, including deploying models from diverse machine learning libraries on Apache Spark.

Applications Using Spark

- Apache Mahout - Previously on Hadoop MapReduce, now switched using Spark as backend
- Natural Language Processing for Apache Spark - A library to provide simple, performant, and accurate NLP annotations for machine learning pipelines

Performance, Monitoring, and Debugging Tools for Spark

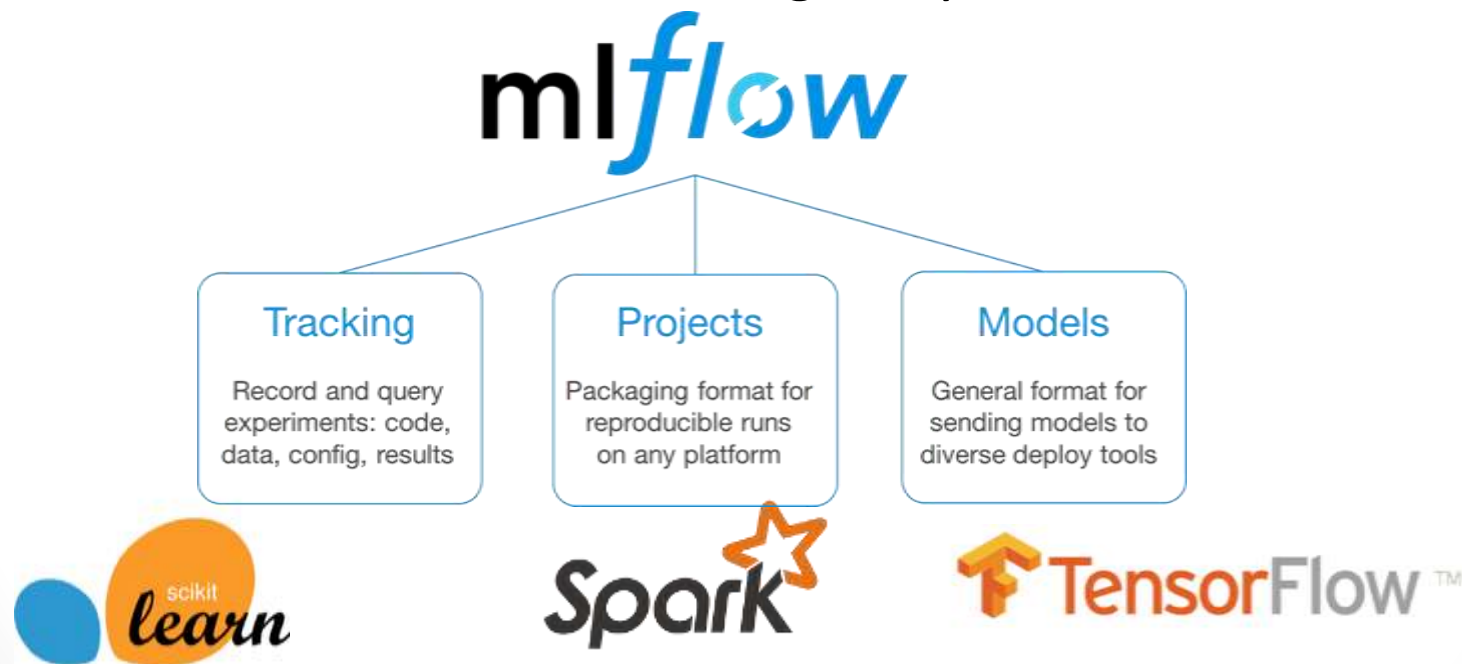
- Performance and debugging library - A library to analyze Spark and PySpark applications for improving performance and finding the cause of failures
- Data Mechanics Delight - Delight is a free, hosted, cross-platform Spark UI alternative backed by an open-source Spark agent. It features new metrics and visualizations to simplify Spark monitoring and performance tuning.

Additional Language Bindings

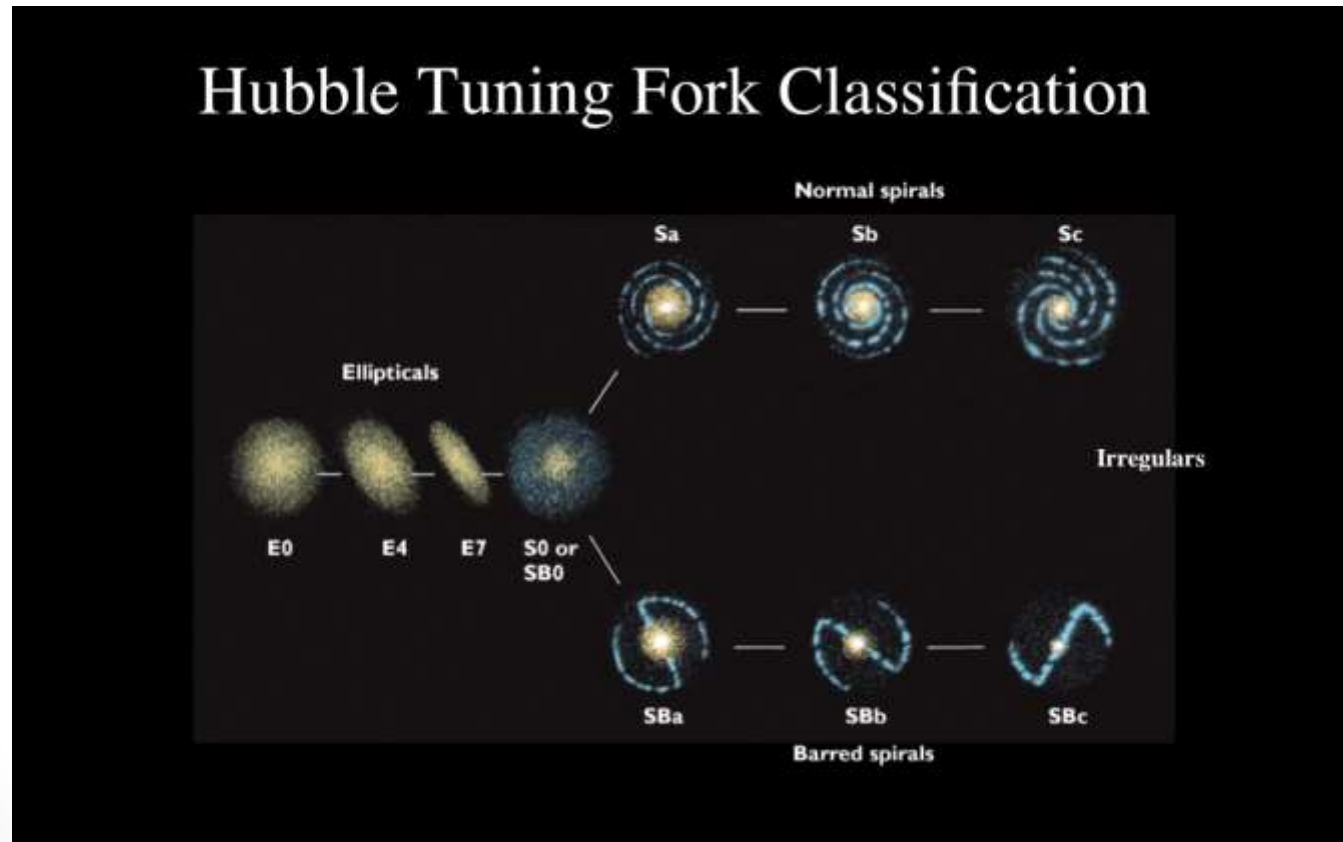
- C# / .NET: Mobius: C# and F# language binding and extensions to Apache Spark
- Julia: Spark.jl
- Kotlin: Kotlin for Apache Spark

A 4th generation of ML libraries?

A cross-cloud open source framework for the complete Machine Learning lifecycle.



Case Study: Classification of Galaxies



Collecting Data: Sloan Digital Sky Survey (SDSS) + Galaxy zoo project

- Manual labelling (high confidence galaxies)
- Three major classes: elliptical, spiral and merger
- Descriptive features:
 - Sloan filter colors
 - Color of the Galaxy: Age
 - Ellipticity of the Galaxy: Shape by minor / major axis
 - Luminosity profile



"The degree of success attained by the method depends largely upon the significance of the features selected as the basis of classification"

Edwin Hubble – The Realm of the Nebulae

Avoid raw pixels: high dimension (only for DL)

Classification Results for Galaxy dataset (10fcv, default parameters)

Spark Decision Tree

- Overall accuracy: 79.1% (4/5 of the galaxies well-classified)
- Possible drawbacks:
 - Overfitting risk
 - High bias on noisy data

Spark Random Forest

- Overall accuracy: 83.1% (better approach)
- Benefits of RF:
 - More stable approach
 - Feature importance interpretation

Source name	u-g colour	g-r colour	r-i colour	r-z colour	Ellipticity 1	Ellipticity 2	Luminosity (u)	Luminosity (r)	Luminosity (z)	Class
Galaxy 1	1.86	0.67	0.42	0.31	0.59	0.57	0.60	0.46	0.33	merger
Galaxy 2	2.11	0.97	0.57	0.34	0.63	0.63	0.34	0.43	0.31	merger
Galaxy 3	1.99	1.01	0.43	0.35	0.84	0.83	0.46	0.31	0.31	elliptical
Galaxy 4	1.98	0.93	0.45	0.30	0.86	0.87	0.33	0.30	0.30	elliptical
Galaxy 5	1.44	0.66	0.35	0.27	0.71	0.72	0.57	0.44	0.43	spiral
Galaxy 6	1.70	0.74	0.40	0.29	0.57	0.56	0.47	0.45	0.46	spiral
...

Classification features

Outline



1

- Big Data. Big Data Science

2

- Why Big Data? Google and the MapReduce programming model

3

- Big Data technologies: Hadoop / Spark ecosystem

4

- Big Data Analytics: Libraries for Data Analytics in Big Data. Case studies

5


- **Final Comments**



Final Comments

...and now what!?

Concluding remarks



Big data is a new computational paradigm	<ul style="list-style-type: none">• Big data is not HPC• More than just large datasets
We need a new strategies to perform ML in those datasets	<ul style="list-style-type: none">• Choosing the right technology is like choosing the right data structure in a program.
Frequently confused terms:	<ul style="list-style-type: none">• Hadoop and Spark are frameworks, MapReduce and RDDs are programming paradigms.

Summary: Big Data implies big opportunities

Useful knowledge in many fields of applications

Beware of bias during learning

- Acquire computational skills
- Problem solving
- Programming

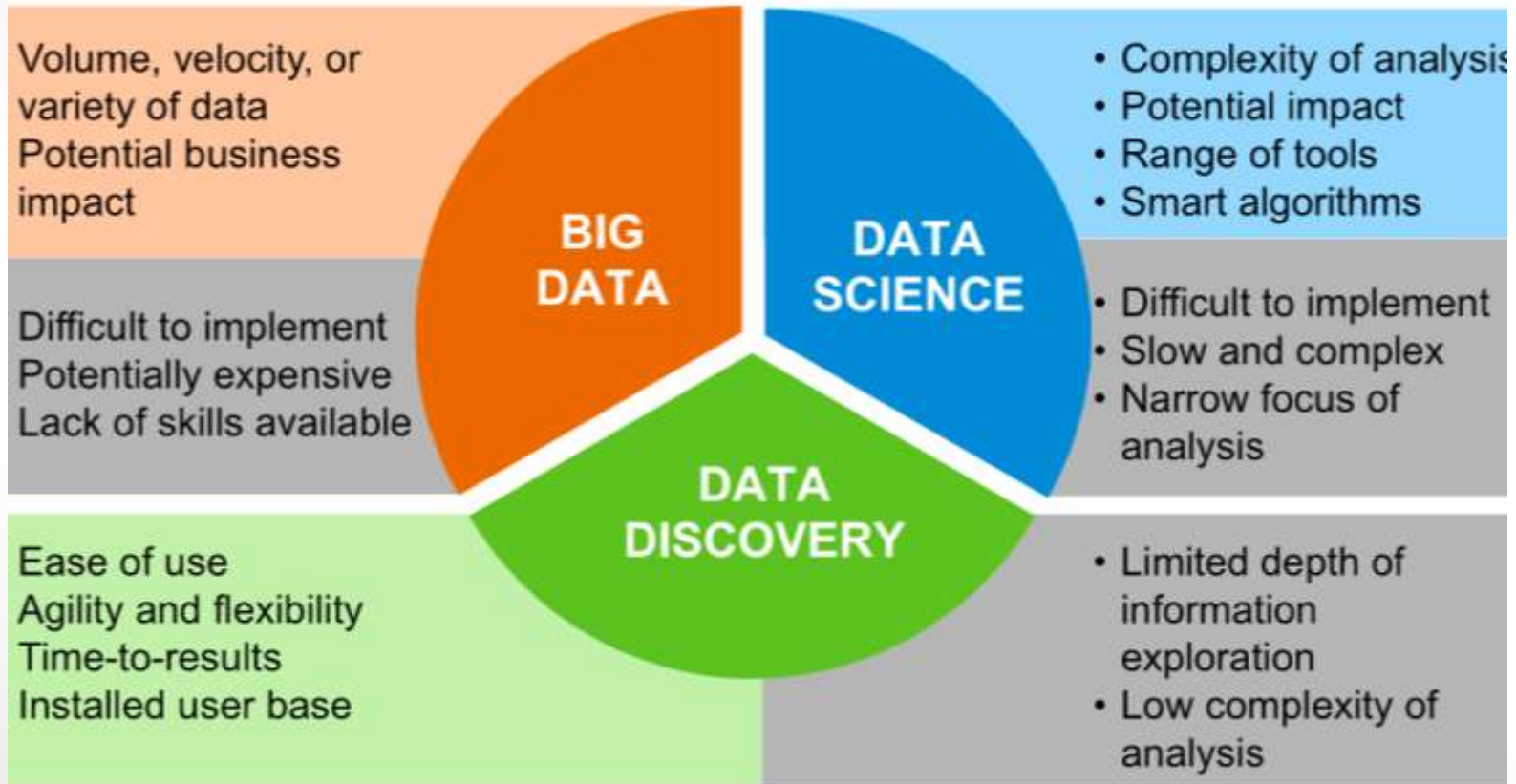
Significance of domain expertise

- Quality data leads to quality solutions
- Quality solutions must be explainable

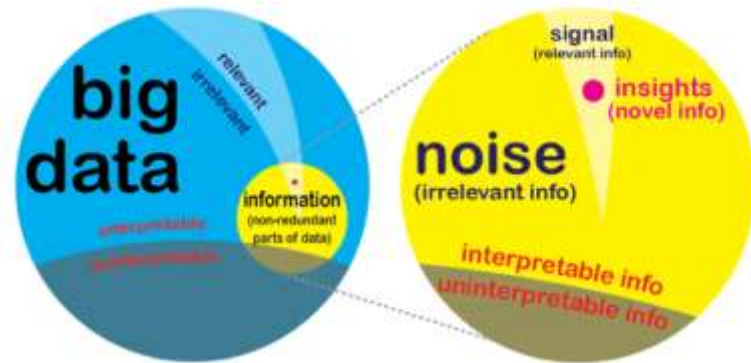
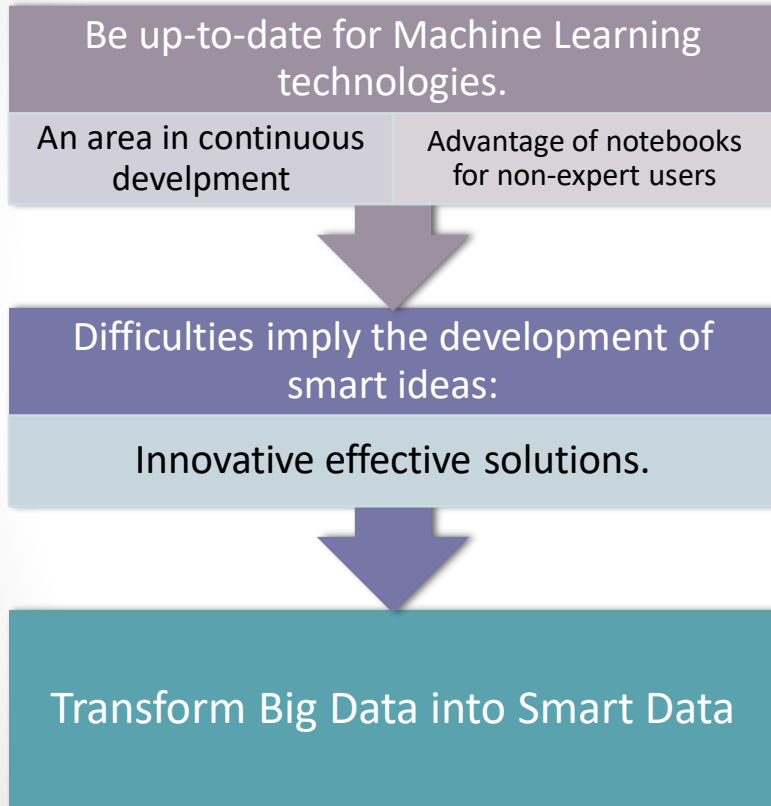
Need to develop / use scalable and accurate algorithms



Pros and Cons



What should be done from here?



THANKS!

QUESTIONS?



SOMACHINE

BIG DATA



UNIVERSIDAD
DE GRANADA



Instituto Andaluz Interuniversitario en
Data Science and Computational Intelligence

Big Data: Foundations and Frameworks

A. Fernández. Instituto Andaluz Interuniversitario en Data Science and Computational Intelligence. **Universidad de Granada.**