Introduction to Data Analytics with Spark

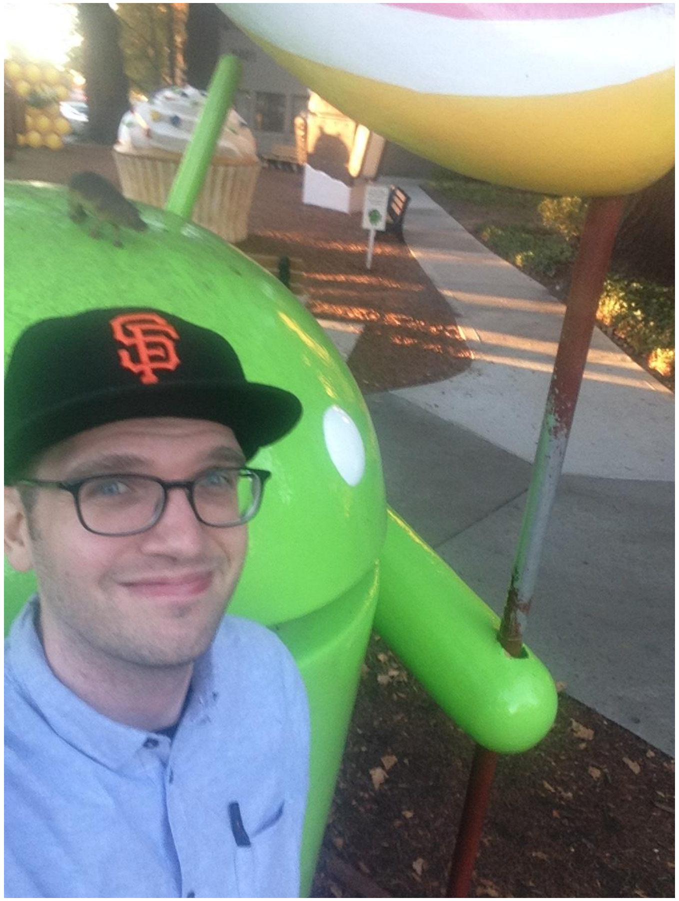
Tackling big data problems
with a cluster computer

compute canada | calcul canada

Calcul Québec

Who am I?

- Félix-Antoine Fortin
- Computer eng. (M. Sc., ~PhD)
- Main interests
  - Advanced Computing
  - Data Analytics
  - Python
- Projects at Compute Canada
  - Digital Humanities
  - Interactive Computing
  - 3D Rendering

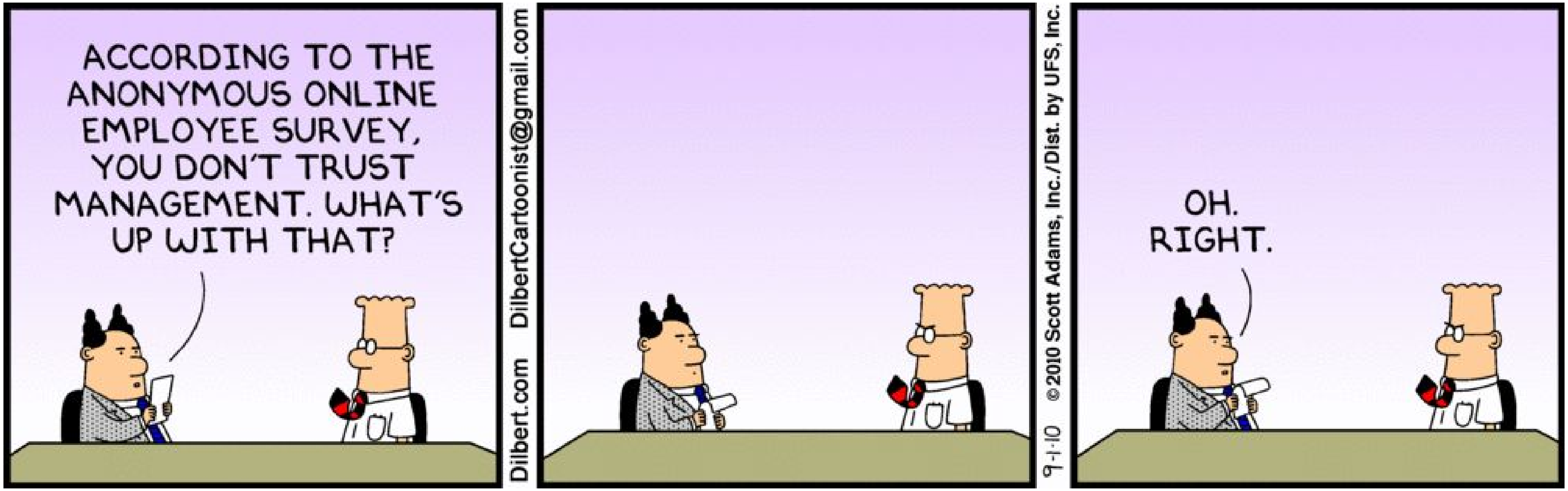


@CmdNtrf

Survey

By the end of the day, share your opinion at

https://goo.gl/ooTmjS



It is anonymous... I swear!

Outline

1. Big Data
2. Introduction to Apache Spark
   2.1. Hands-on
   2.2. Manipulating arbitrary objects
3. Key-Value Pairs in Spark
   3.1. Hands-on
4. Spark SQL
5. Machine Learning

https://github.com/calculquebec/cq-formation-spark

Setup check

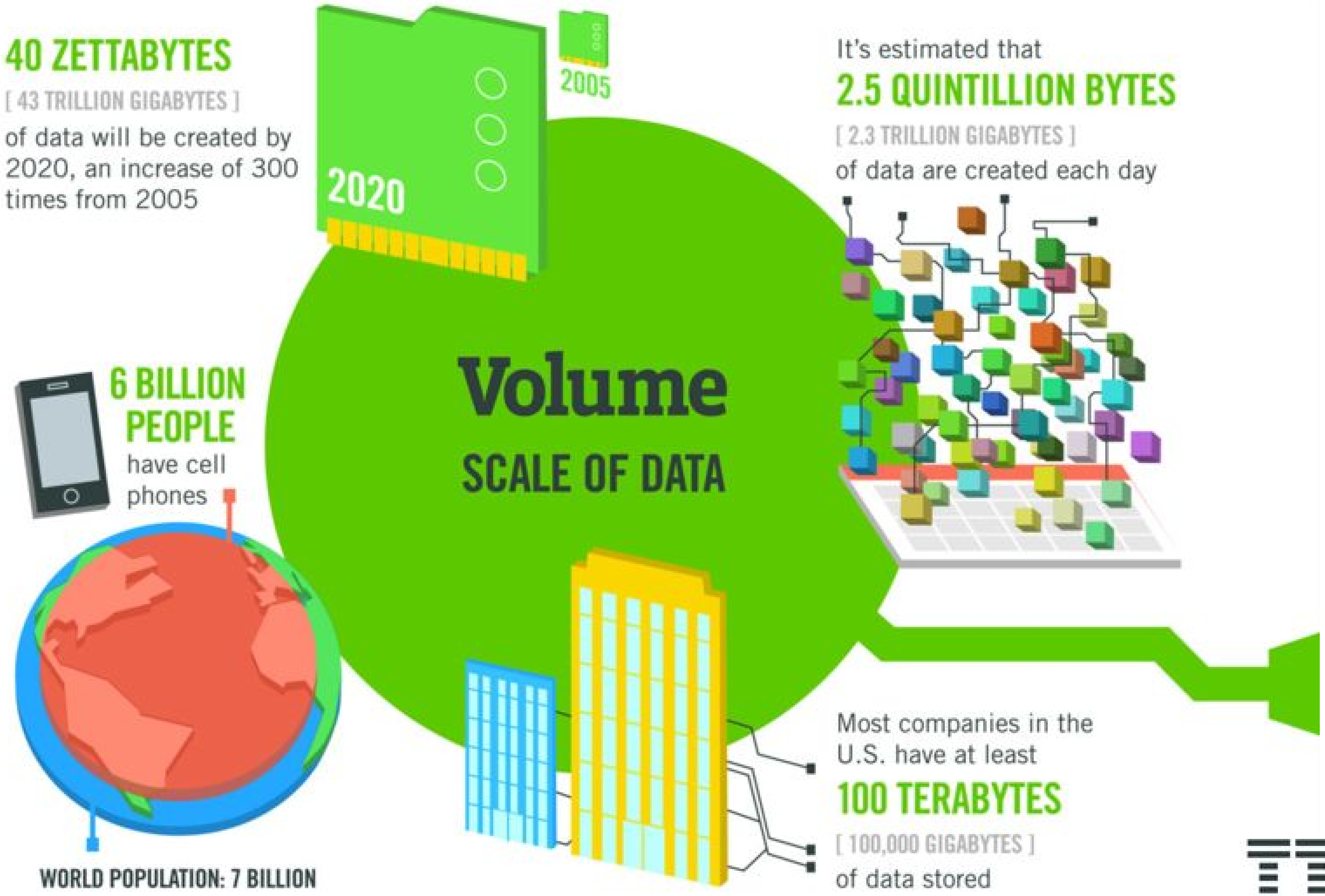
compute canada | calcul canada

Calcul Québec

What Qualifies Big Data?

The Big Data * Vs in Images

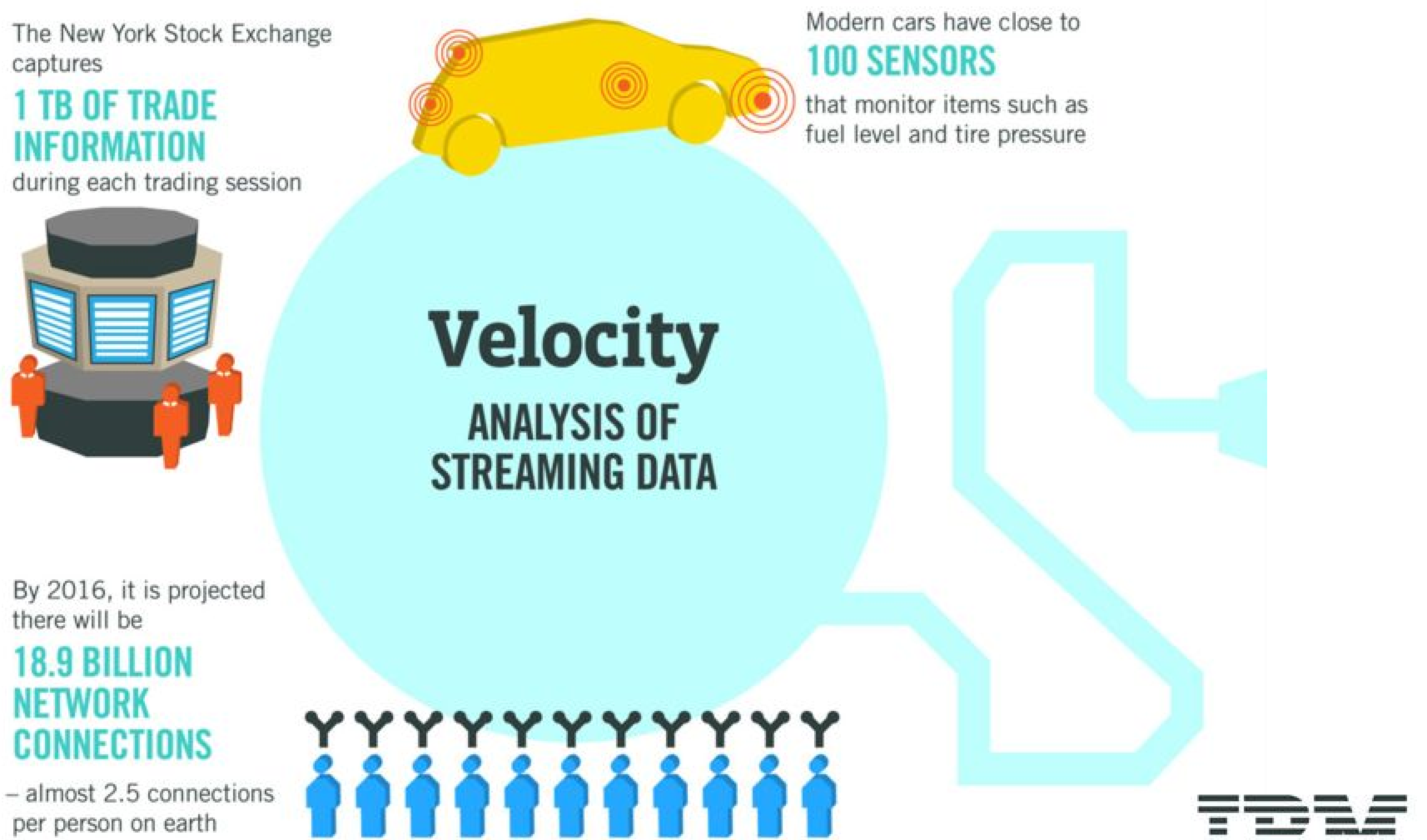
Big Data - Volume



Source : http://www.ibmbigdatahub.com/infographic/four-vs-big-data

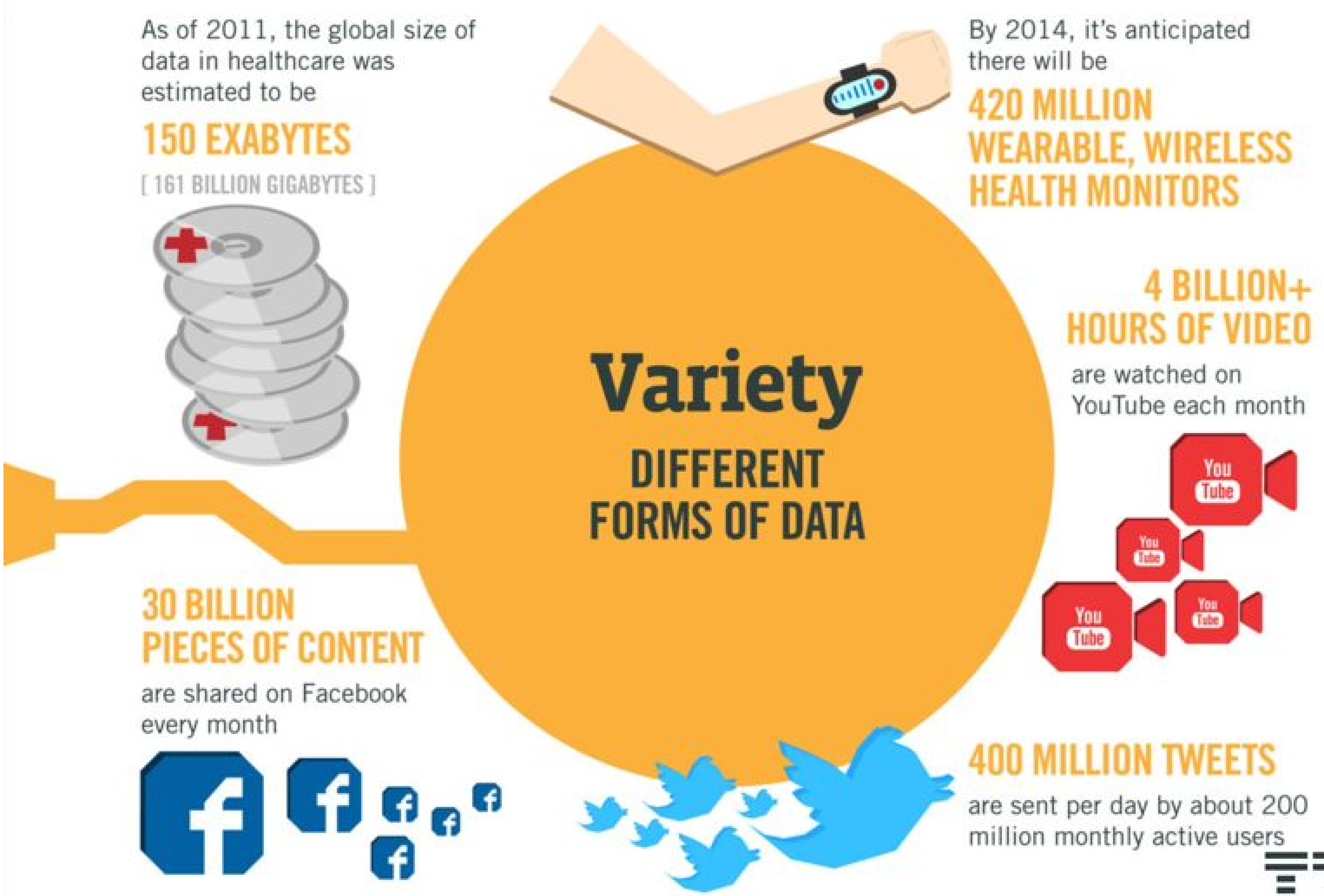
Big Data - Velocity

Calcul Québec



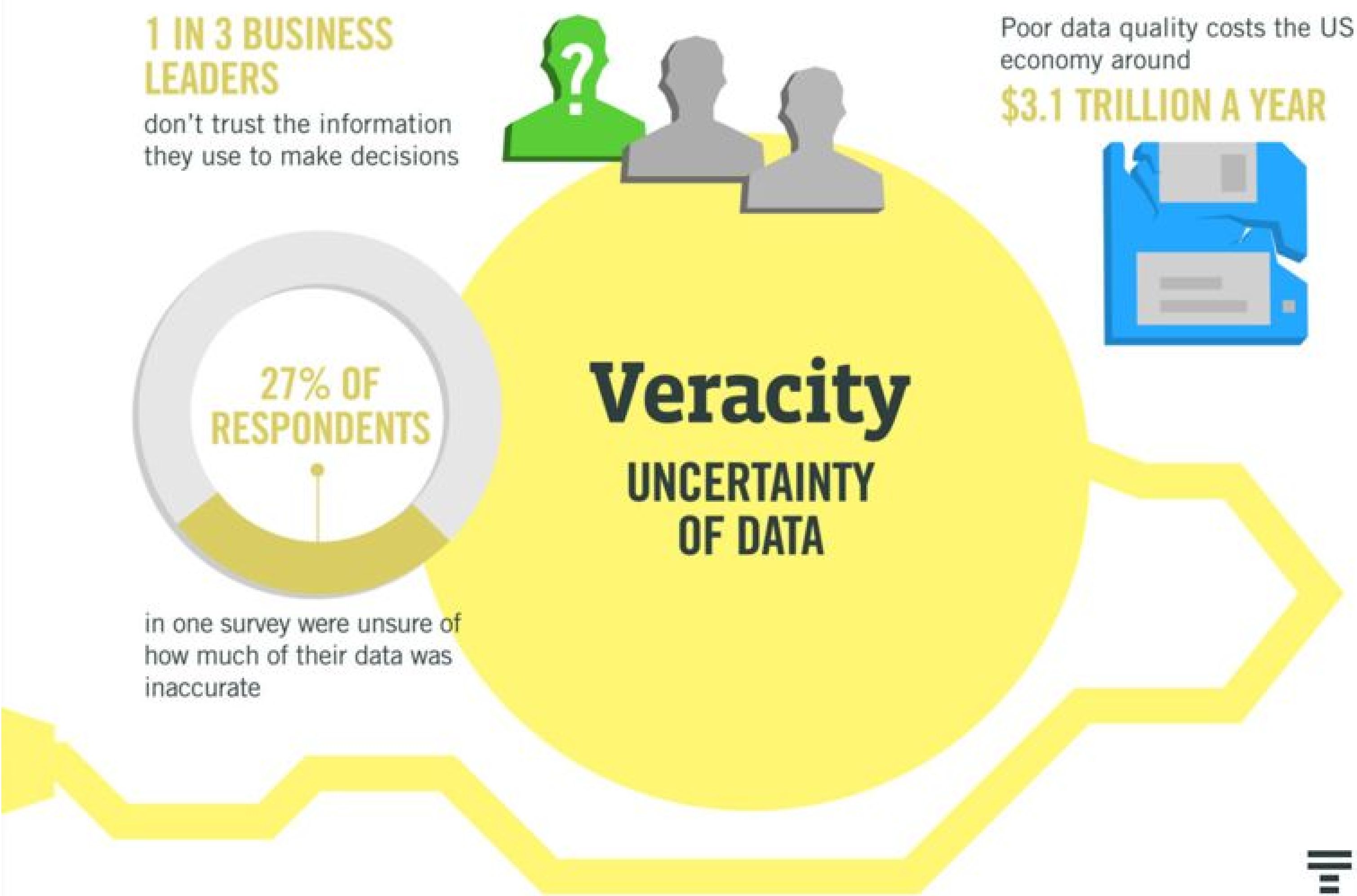
Source : http://www.ibmbigdatahub.com/infographic/four-vs-big-data

Big Data - Variety



Source : http://www.ibmbigdatahub.com/infographic/four-vs-big-data

Big Data - Veracity



Source : http://www.ibmbigdatahub.com/infographic/four-vs-big-data

Why Data Analytics?

Eric Schmidt : "One day we had a conversation where we figured we could just try and predict the stock market...and then we decided it was illegal. So we stopped doing that."

http://www.bloombergview.com/articles/2015-01-23/capital-one-fraud-researchers-may-also-have-done-some-fraud

Problems

We want to process this data but

- Too much data for a single computer
  - Does not fit in memory
  - Does not fit on disk

Solution:

- Use data parallelism

Type of parallelism

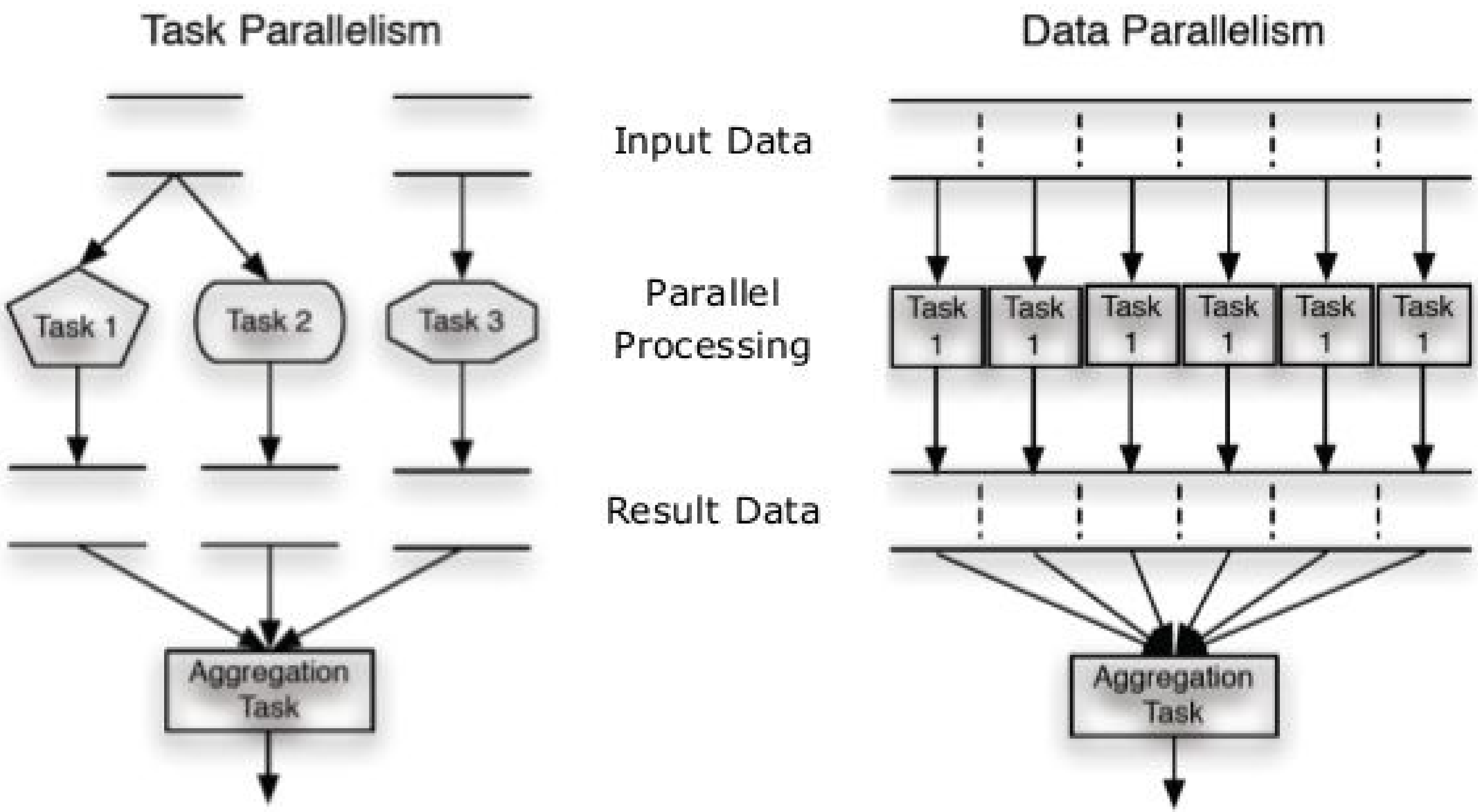
Task parallelism

- Each process has a specific set of tasks and it applies these tasks to input data.

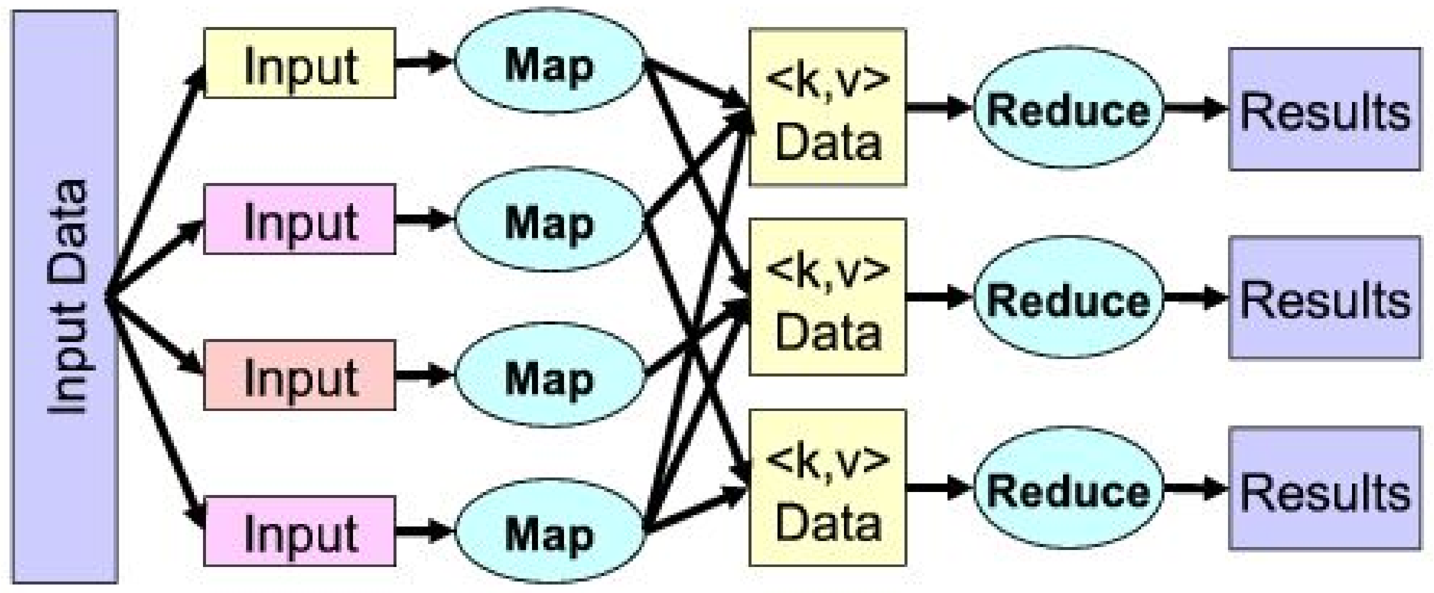
Data parallelism

- Each process has a specific set of data and it applies input task on these data.

Type of parallelism



Map-Reduce Paradigm



Paradigm popularized by Hadoop

Map-Reduce example

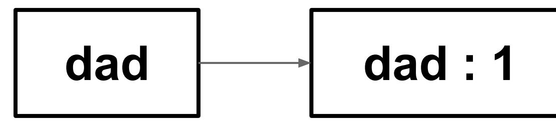
Calcul Québec

Counting words in a set of documents

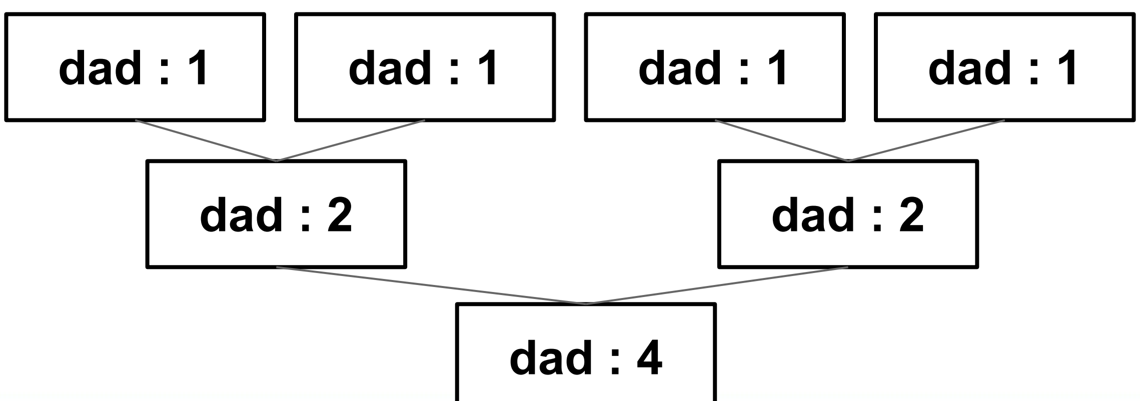
- ✓ multiples files
- ✓ a lot of words in each files

We want to compute the frequency of each unique word

- Map: associate a frequency to a word:



- Reduce: sum frequencies by word:



Calcul Québec

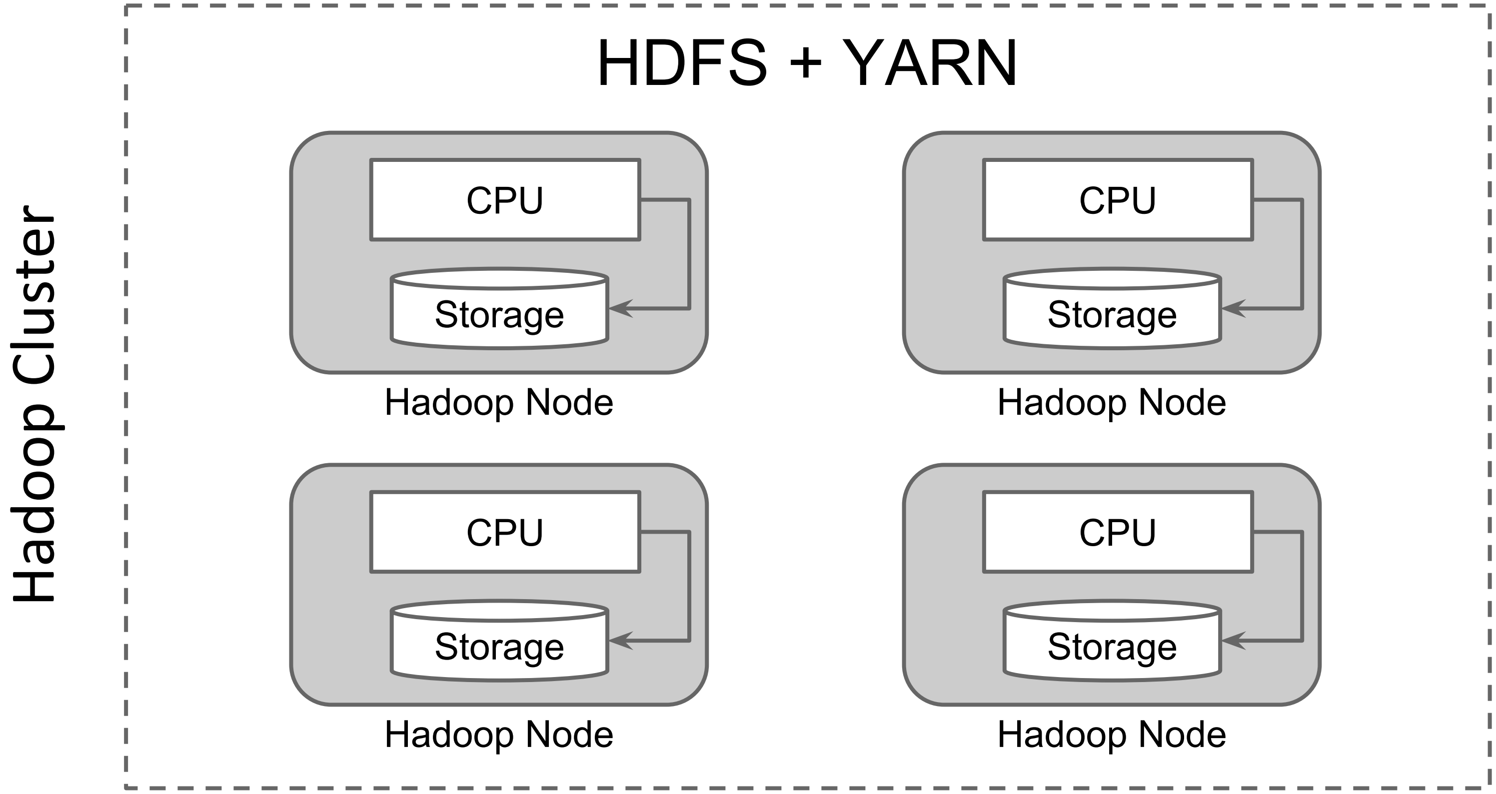
Hadoop

Framework allowing the distributed processing of a large quantity of information using a **simple programming model based on data.**

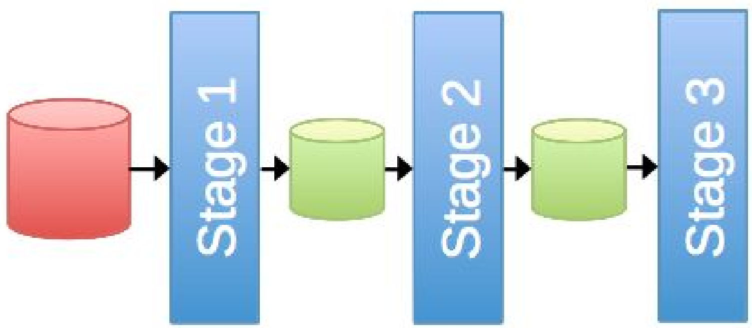
HDFS + Map-Reduce

Hadoop - architecture

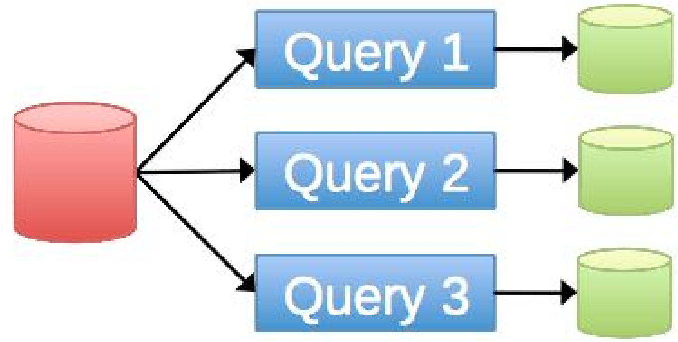
Hadoop regroups the processing and the storage of data on the same node (data locality)



Beyond Map-Reduce



Iterative algorithm



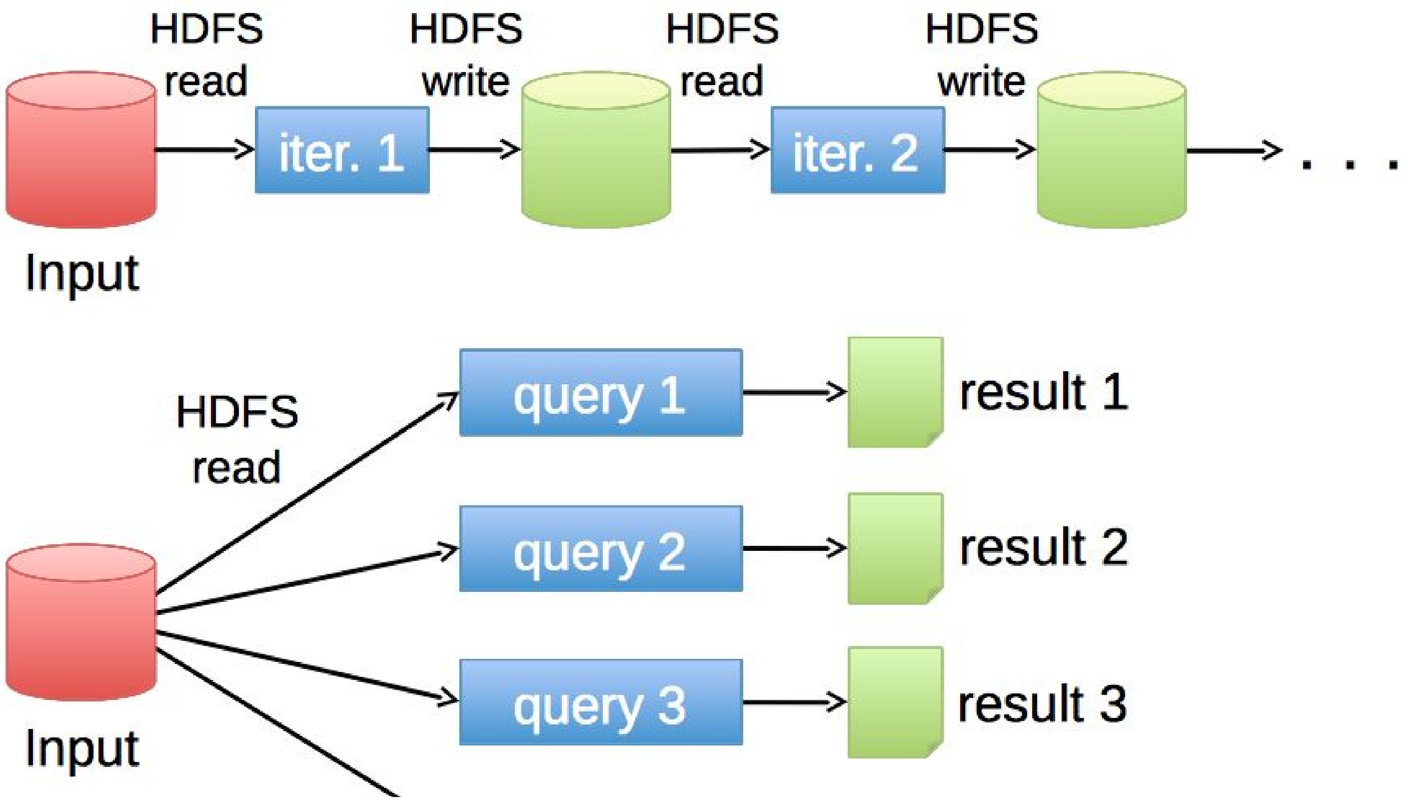
Interactive analysis

Complex and interactive tasks require something Map-Reduce cannot offer:

An efficient primitive for sharing data

Beyond Map-Reduce

Hadoop's Primitive for data sharing: storage!

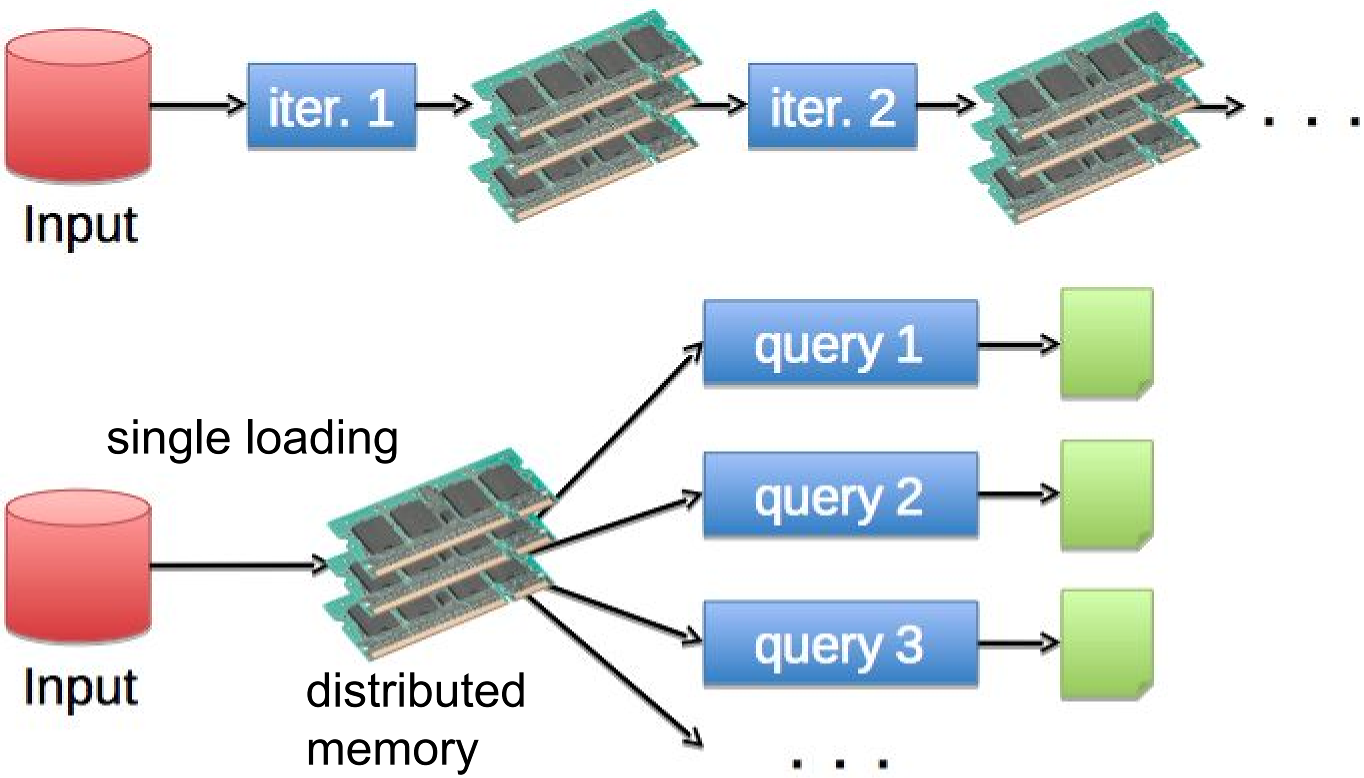


Serialisation and I/O = up to 90% of job time!

Introduction to Spark

- Open Source Project since 2010
- Over 700 developers contributing in 2015
- Principles:
  - Ease data scientist's task
  - Provide a rich function library
  - Access diverse data sources
  - **Use cache to avoid data moving**

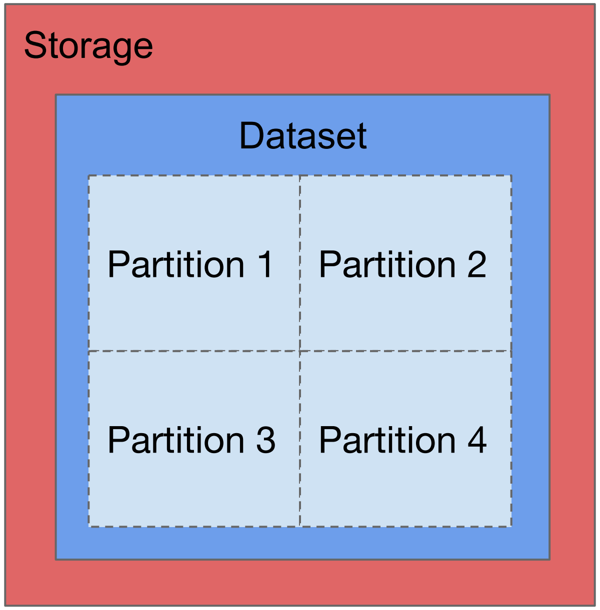
Principles behind Spark



Source: http://fr.slideshare.net/tsailiming/spark-meetup1-intro-to-spark

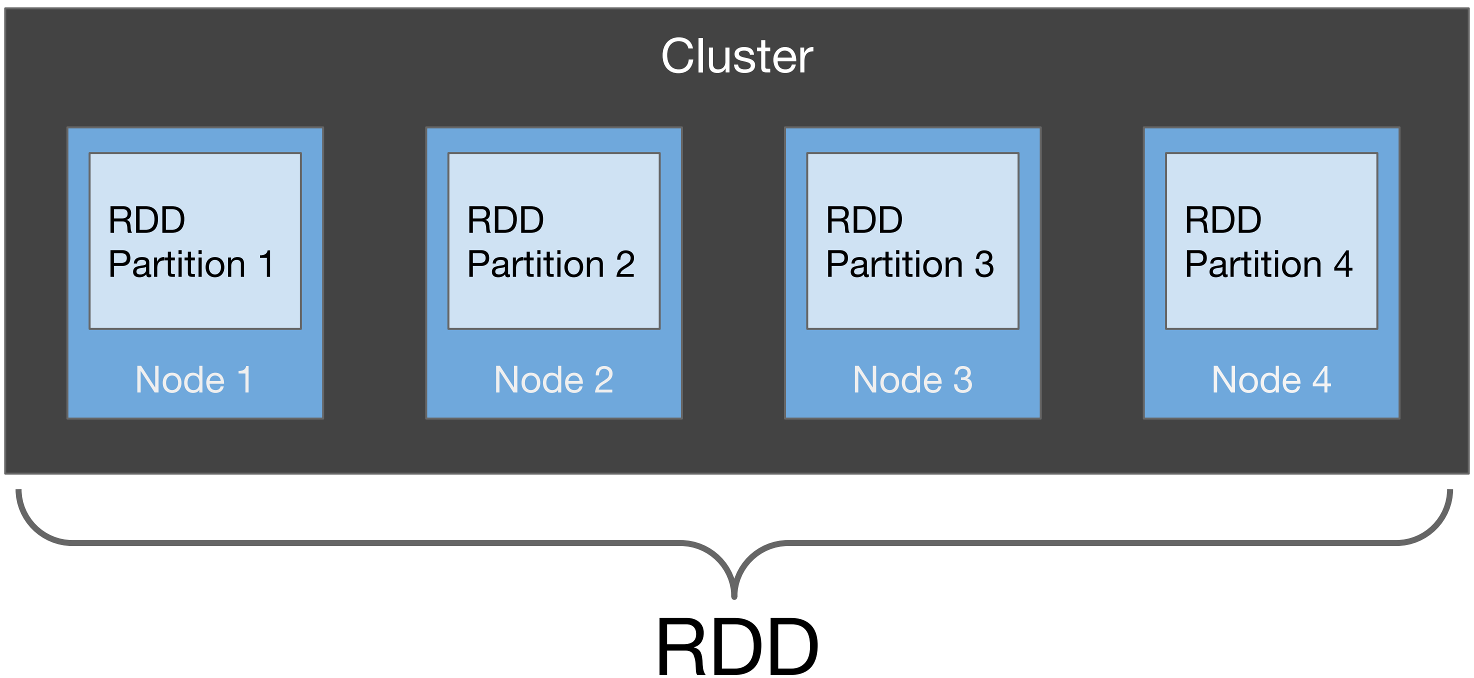
Resilient Distributed Dataset

- Logical data collection
- Data cannot fit in the memory of a single node



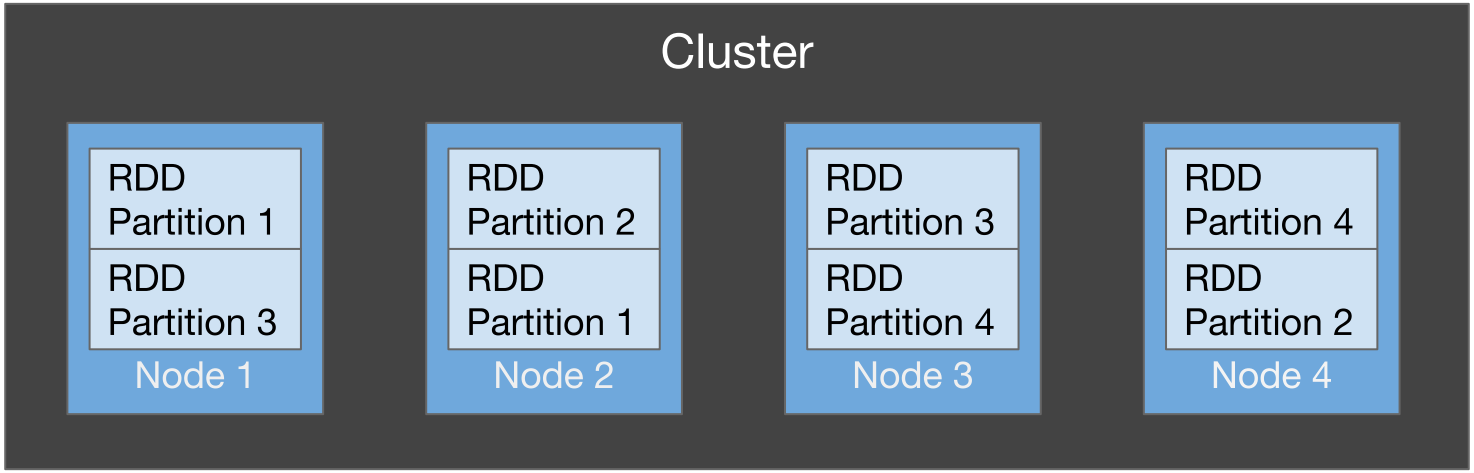
Resilient Distributed Dataset

- Partition data between nodes
- Store the data in memory*

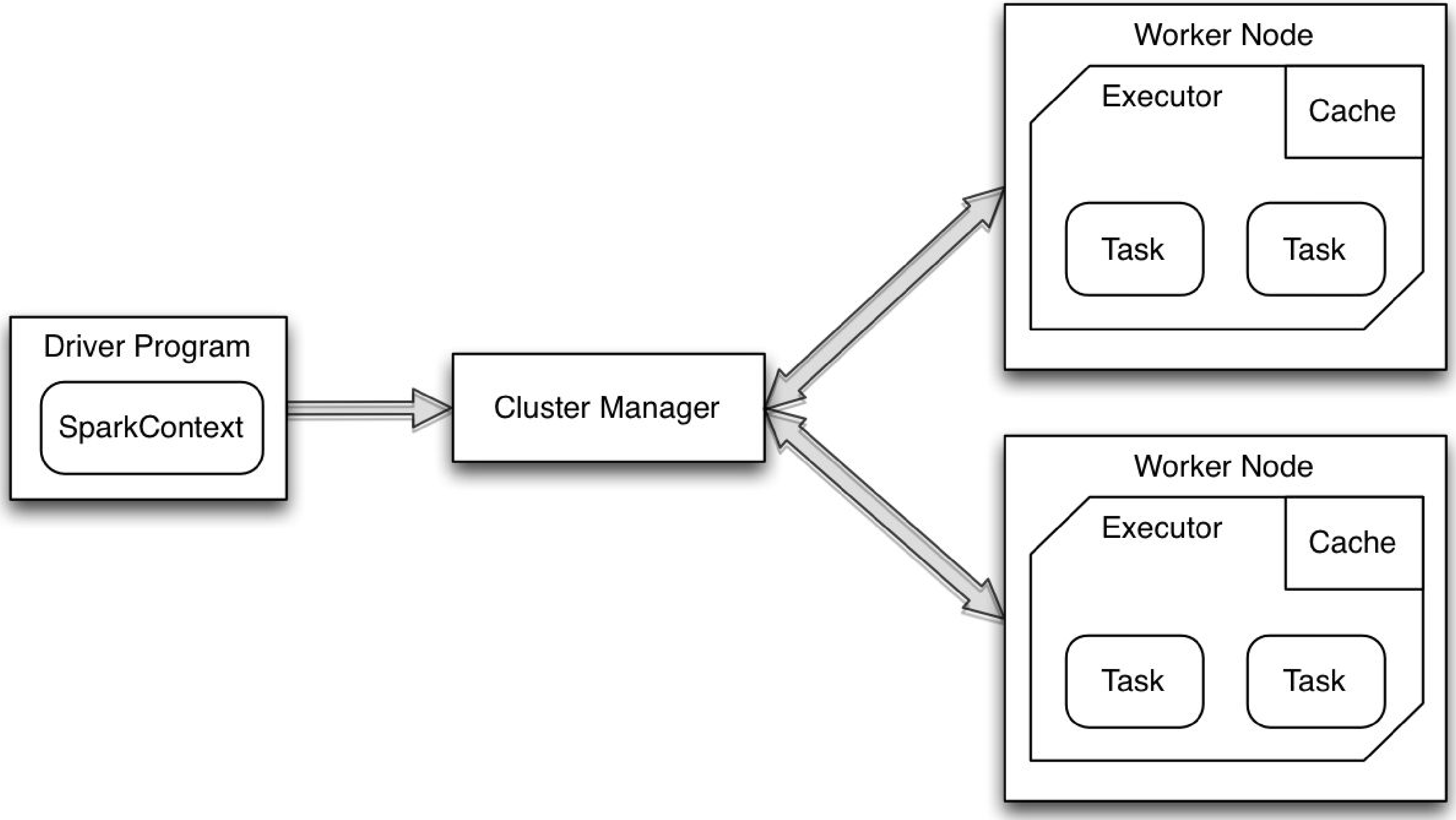


Resilient Distributed Dataset

- Fault tolerance:
  - Partition replication on multiple nodes
  - Memorization of RDD transformations



Spark Architecture



Spark: performance

Terasort contest

	Hadoop Record	Spark 100TB	Spark 1PB
Size	100TB	100TB	1000 TB
Duration	72 minutes	**23 minutes**	234 minutes
# Nodes	2100	206	190
# Cores	50,400	6592	6080
Instance type	Dedicated	EC2 (i2.x8large)	EC2 (i2.x8large)

- Up to 100x faster than Hadoop MapReduce in memory
- Up to 10x faster on disk

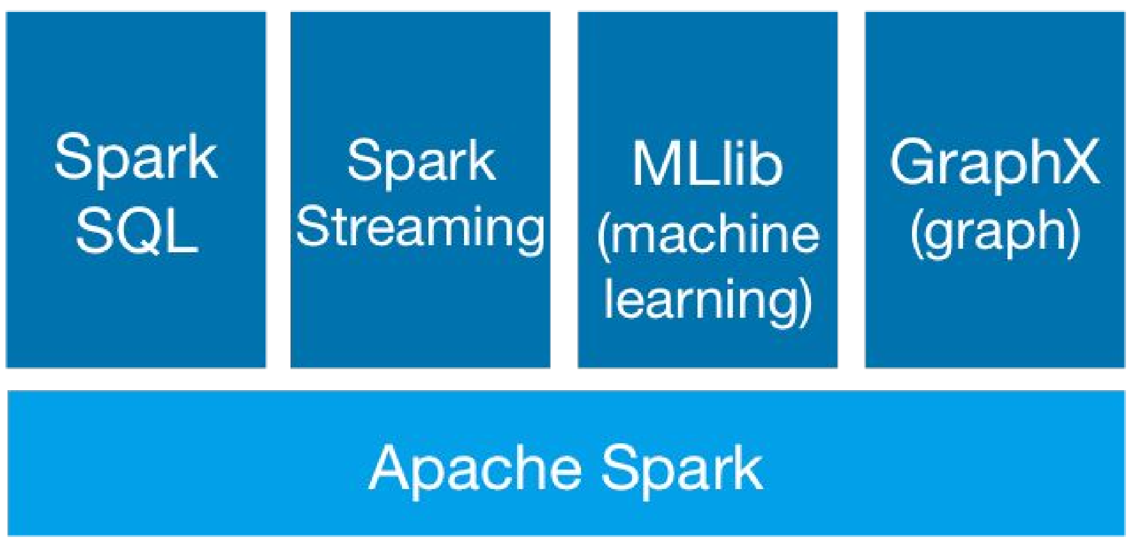
Reference : http://sortbenchmark.org/

Spark: ecosystem

API

- Java
- Scala
- Python
- R

Librairies



Spark RDD: creation

RDD can be created from different external sources. Files can be multiples and compressed

`textFile`

It is also possible to parallelize arbitrary objects*.

`parallelize`

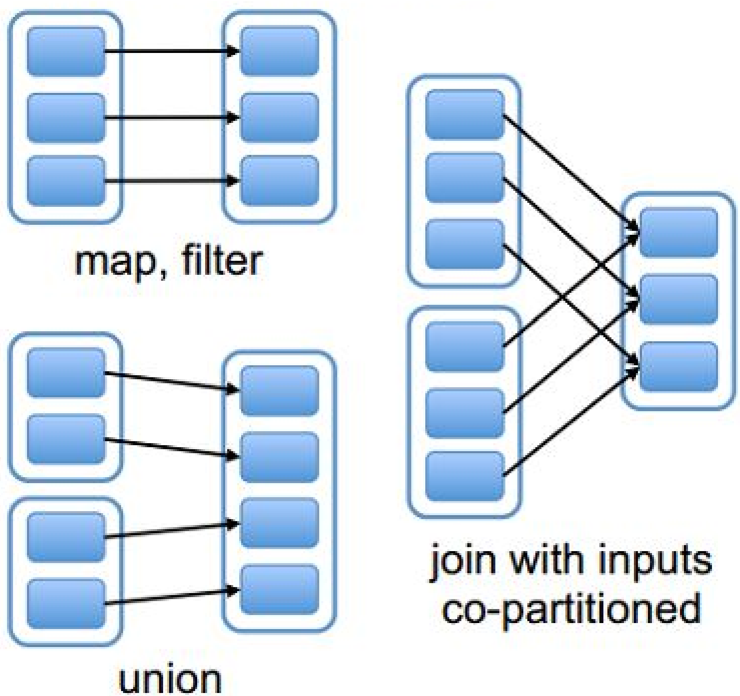
Spark RDD: transformations

- RDDs are immutable.
- These functions create a new RDD lazily (lazy evaluation).

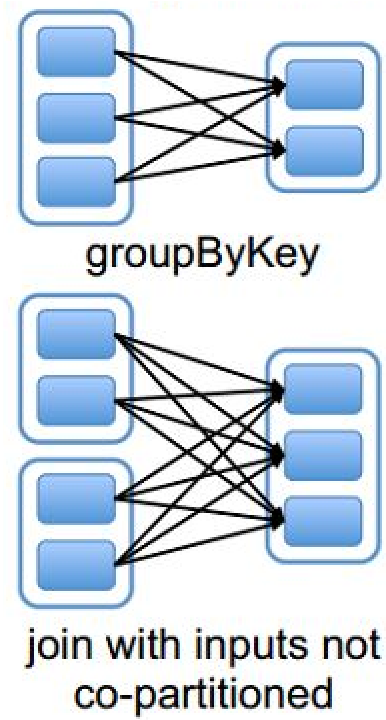
`map`	`distinct`	`mapPartitions`
`filter`	`groupByKey`	`union`
`flatMap`	`reduceByKey`	`intersection`
`sample`	`sortByKey`	`cartesian`
	`join`	`cogroup`

Spark RDD: transformations*

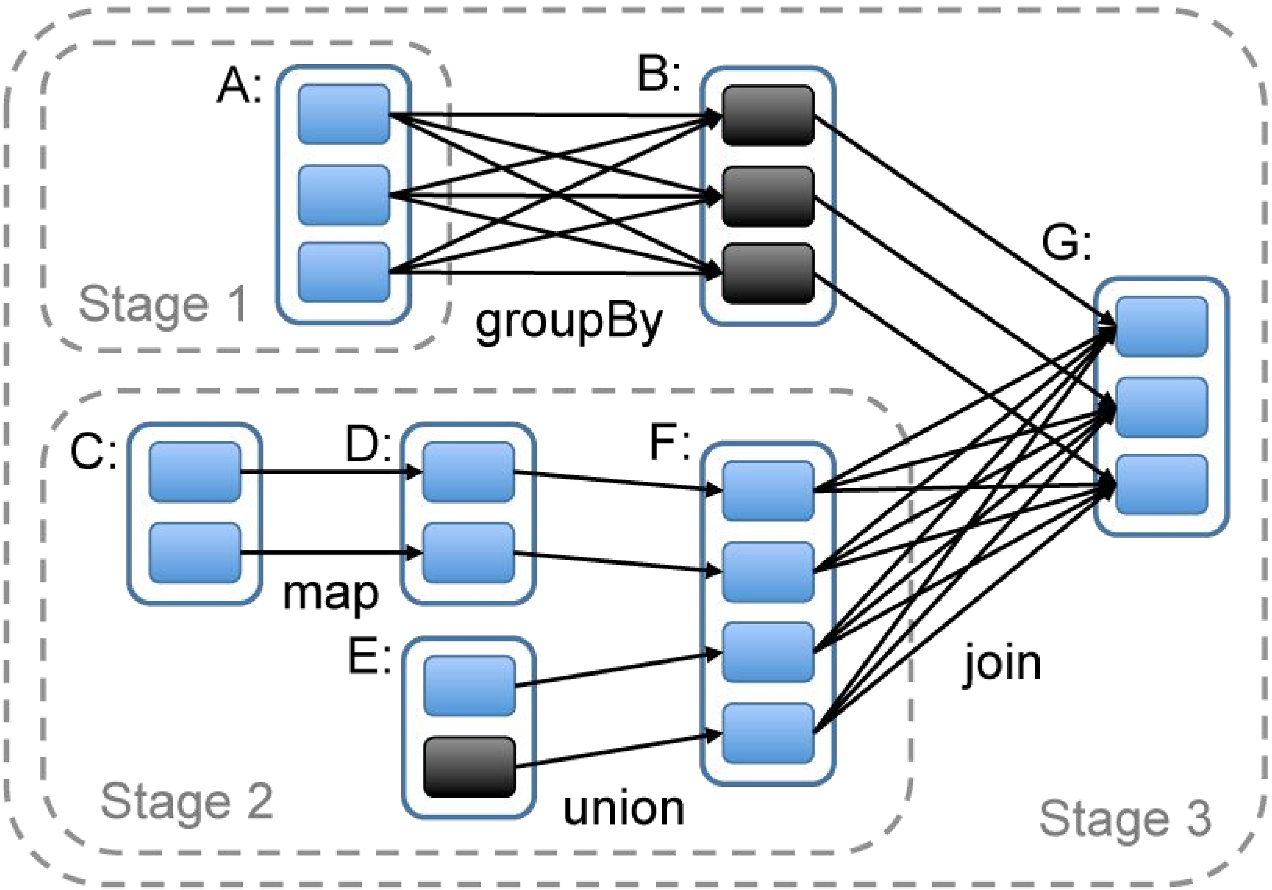
Narrow dependencies



Wide dependencies



Spark RDD: transformations*



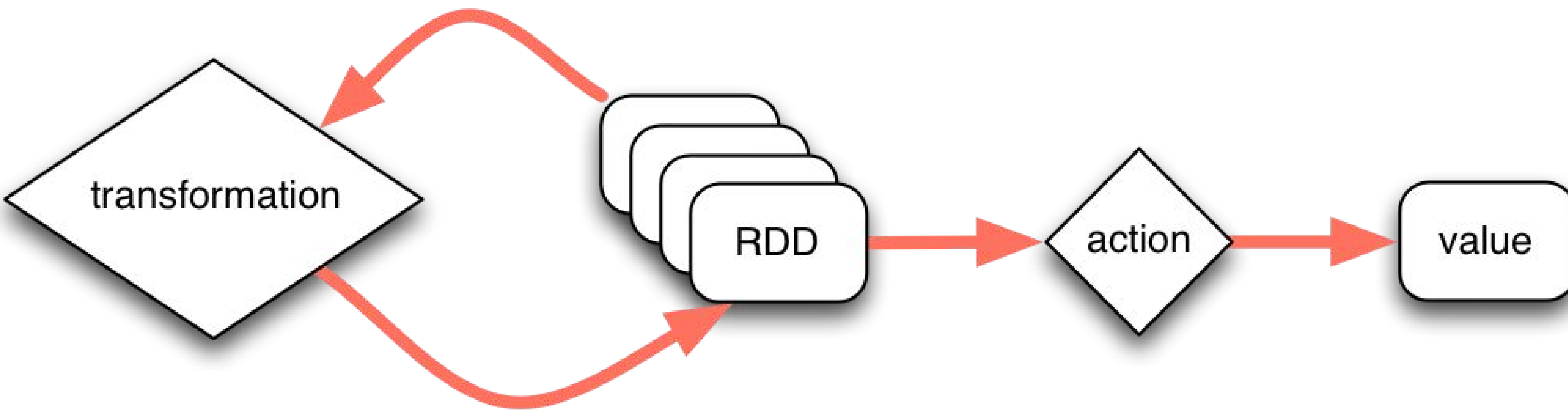
Wide dependencies imply partition shuffling between nodes (costly operation)

Spark RDD: actions

Actions produce an immediate result that needs to fit in the memory of the driver or on disk.

```
reduce      take
collect     takeSample
count       saveAsTextFile
first       foreach
```

Spark RDD: Lifecycle



Spark RDD: persistence

Dataset are not necessarily kept in memory.

`cache`

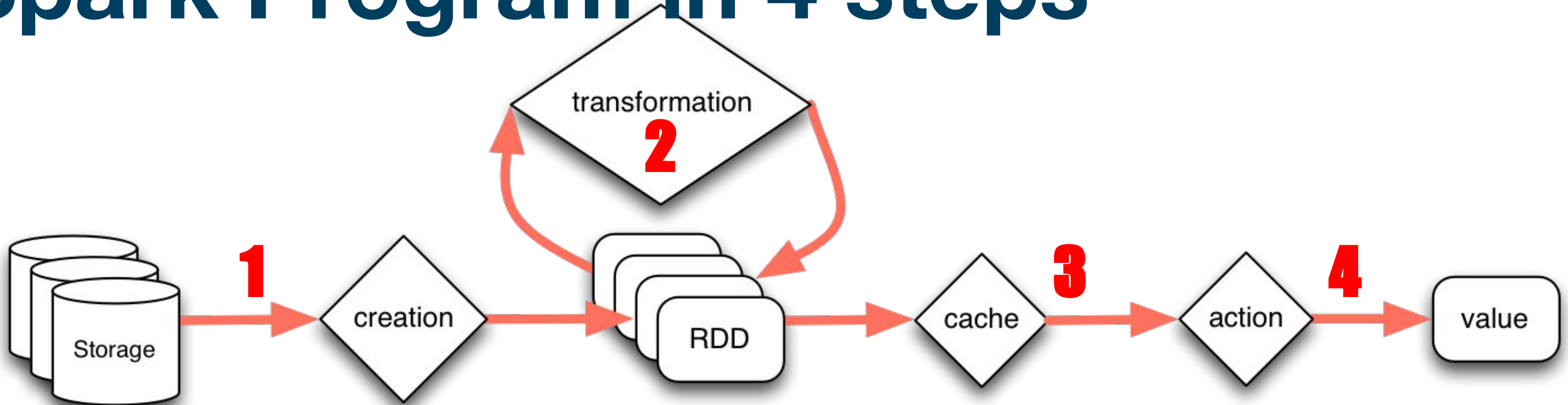
If there is not enough memory to store the whole dataset, different schemes of persistence can be used.

`persist`

To evict the RDD from persistent memory.

`unpersist`

Spark Program in 4 steps



1. Create input RDDs from external data or parallelize data from in the driver program.
2. Transform them lazily to define new RDDs using transformations like `map()`, `filter()`, `etc.`
3. Ask Spark to `cache()` any intermediate RDDs that will need to be reused
4. Launch action such as `count()` or `collect()` to start parallel computation optimized and executed by Spark.

Spark: counting words

```
sc = pyspark.SparkContext()

file = sc.textFile("...")

counts = file.flatMap(lambda line: line.split(" "))\
             .map(lambda word: (word, 1))\
             .reduceByKey(lambda a, b: a + b)

counts.saveAsTextFile("...")
```

Spark: estimating π with MC

```
def sample(p):
    x, y = random(), random()
    return 1 if x*x + y*y < 1 else 0

sc = pyspark.SparkContext()
count = sc.parallelize(range(0, NUM_SAMPLES)) \
          .map(sample) \
          .reduce(lambda a, b: a + b)

print("Pi est approximativement = %f" % \
      (4.0 * count / NUM_SAMPLES))
```

Basics Hands-on!

Getting Ready

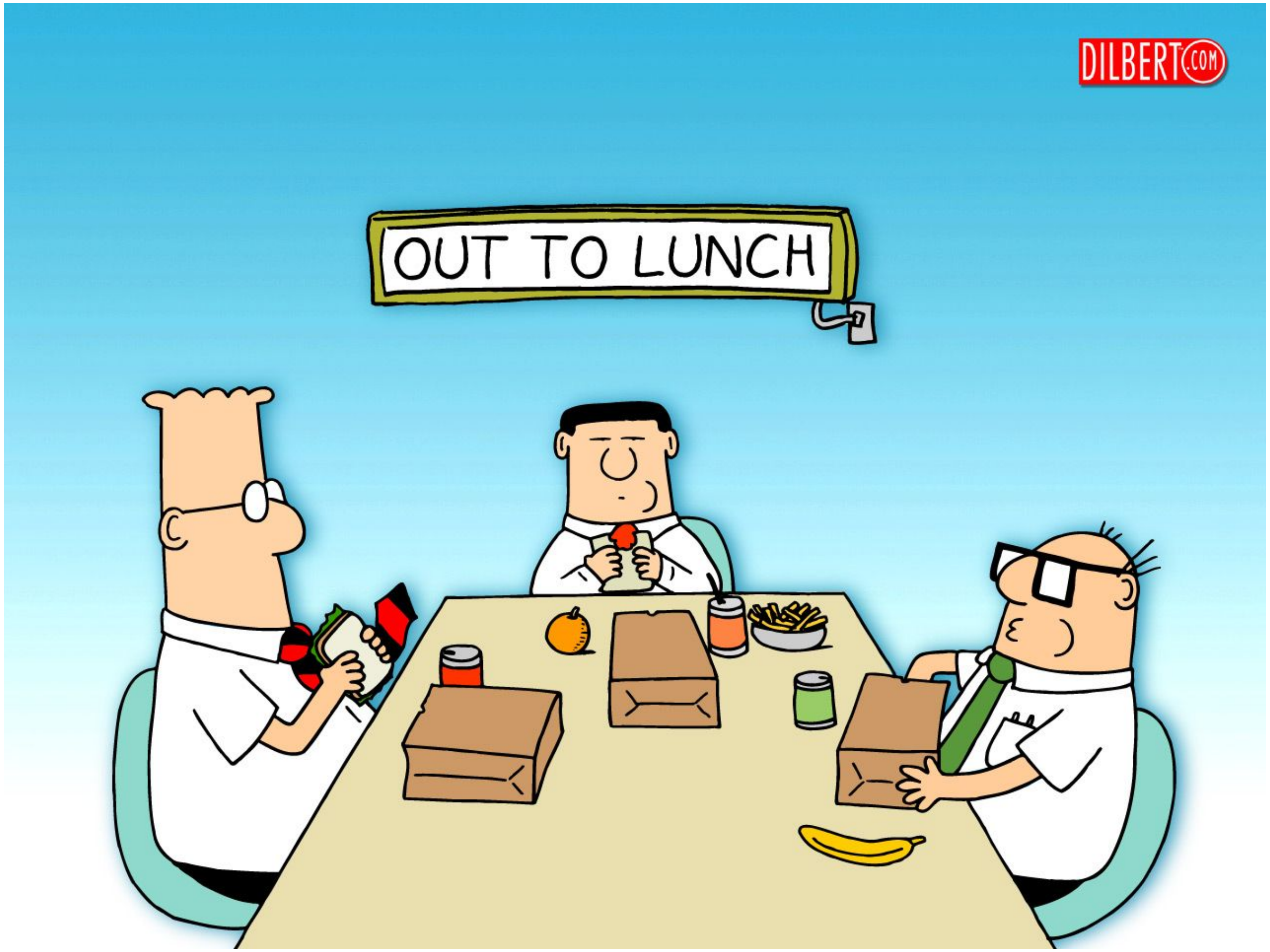
1. Clone course repo
   `git clone https://github.com/calculquebec/cq-formation-spark`
2. Start the VM
   `cd cq-formation-spark; vagrant up`
3. Pull recent changes
   `git checkout -f .; git pull`
4. Go to your personal Jupyter instance
   http://localhost:8001/

0-Configuration.ipynb

1-Tools_Intro.ipynb

2-Spark_Intro.ipynb

3-Manipulating_Images.ipynb



Time for a break!

Key-Value Pairs

Distributed data can be organized into key-value pairs

Why? Act on each key in parallel or regroup data across the network

What? Extract fields from an RDD can be used as keys

How? Run a `map()` function that returns key/value pairs

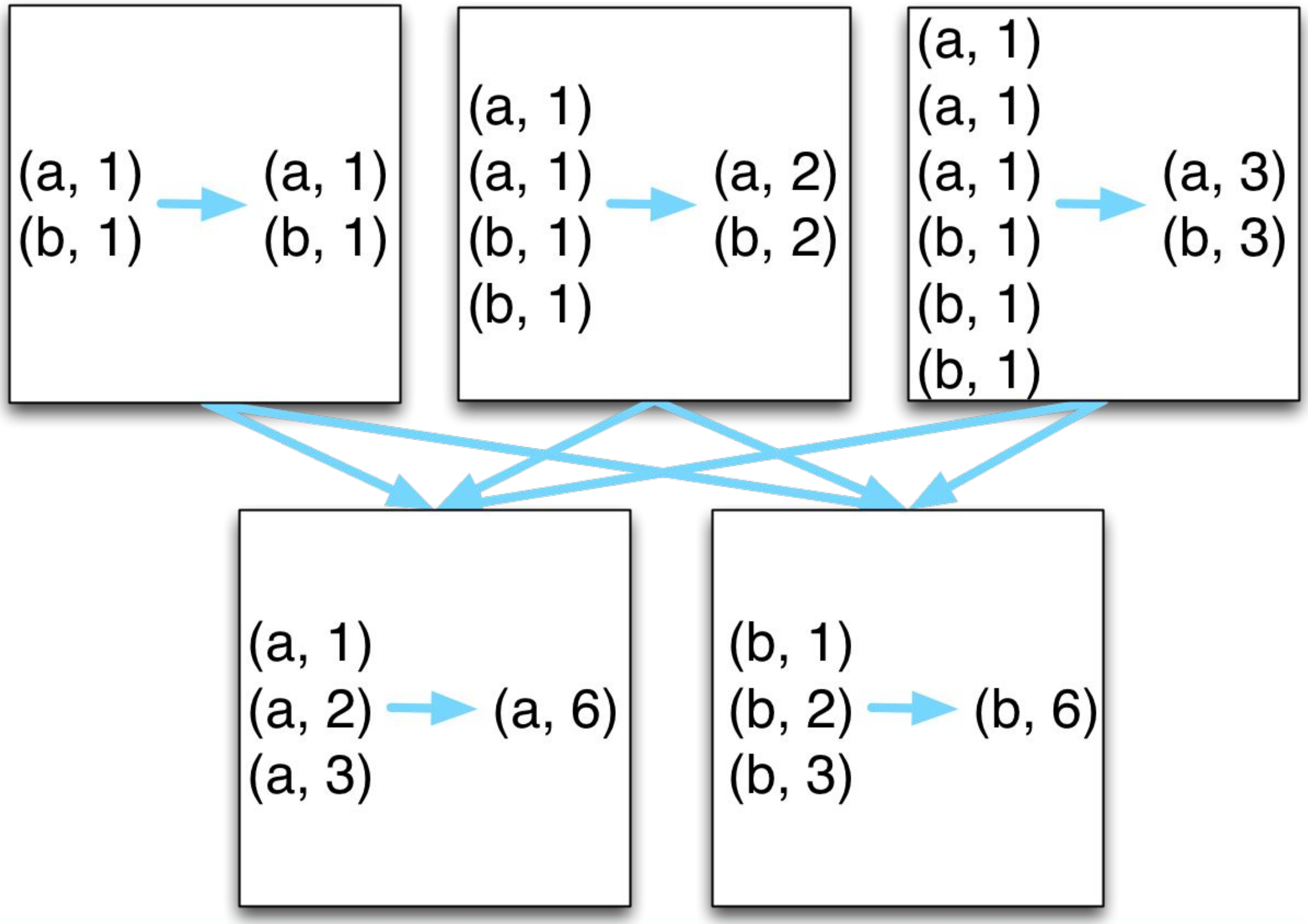
value
821, xbox, white, 450
830, ps4, black, 500
900, wii, white, 200

`map()`

key	value
821	(xbox, white, 450)
830	(ps4, black, 500)
900	(wii, white, 200)

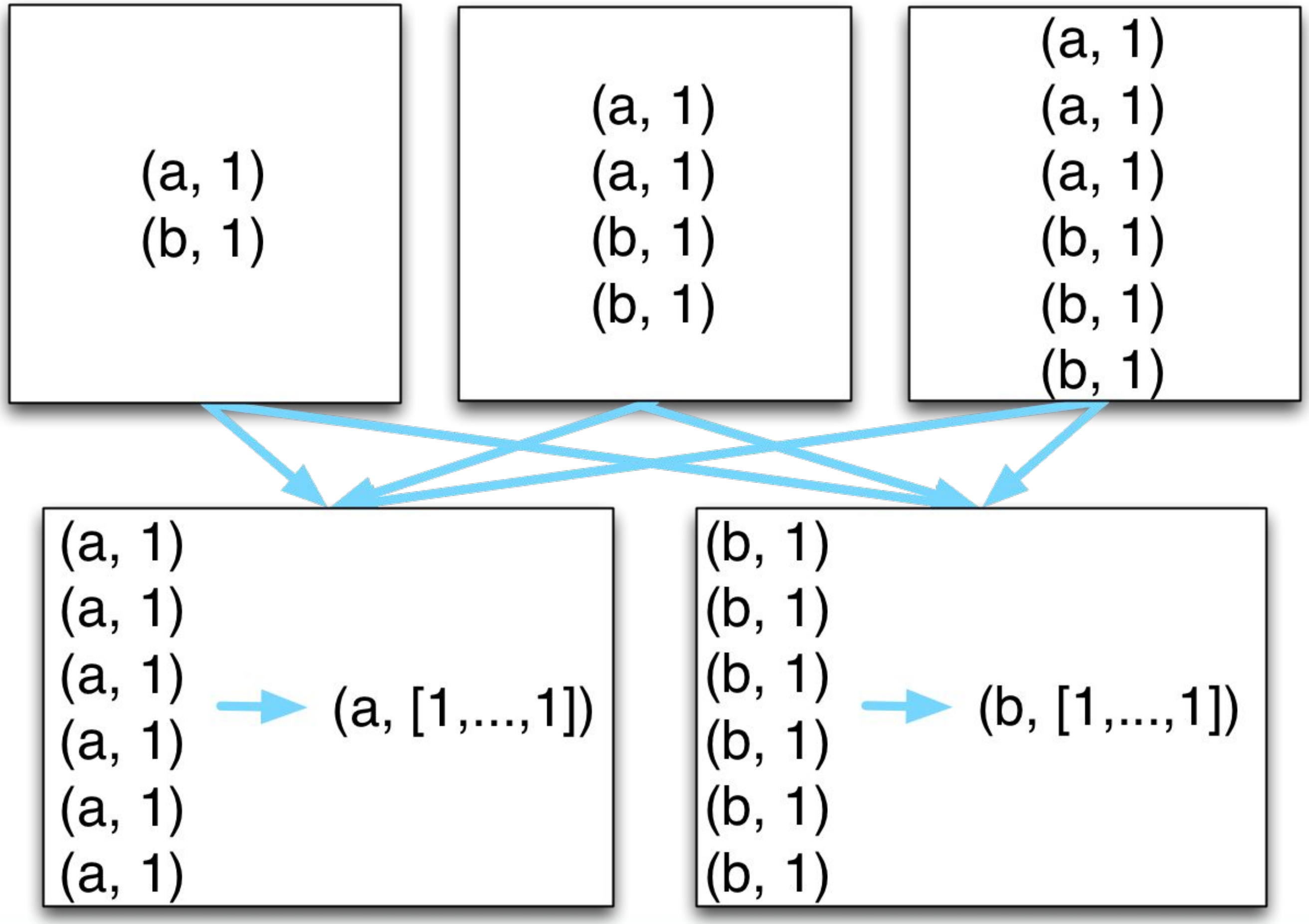
Transformations

`RDD.reduceByKey(func)`



Transformations

`RDD.groupByKey()`



Transformations

Calcul Québec

`RDD.join(RDD)`

key	value
821	xbox
830	ps4
900	wii

key	value
821	450
830	500
901	50

`join()`

key	value
821	(xbox, 450)
830	(ps4, 500)

Only 2 entries?

Transformations

`RDD.[left,right]OuterJoin(RDD)`

key	value
821	xbox
830	ps4
900	wii

key	value
821	450
830	500
901	50

`leftOuterJoin()`

key	value
821	(xbox, 450)
830	(ps4, 500)
900	(wii, None)

`rightOuterJoin()`

key	value
821	(xbox, 450)
830	(ps4, 500)
901	(None, 50)

Transformations

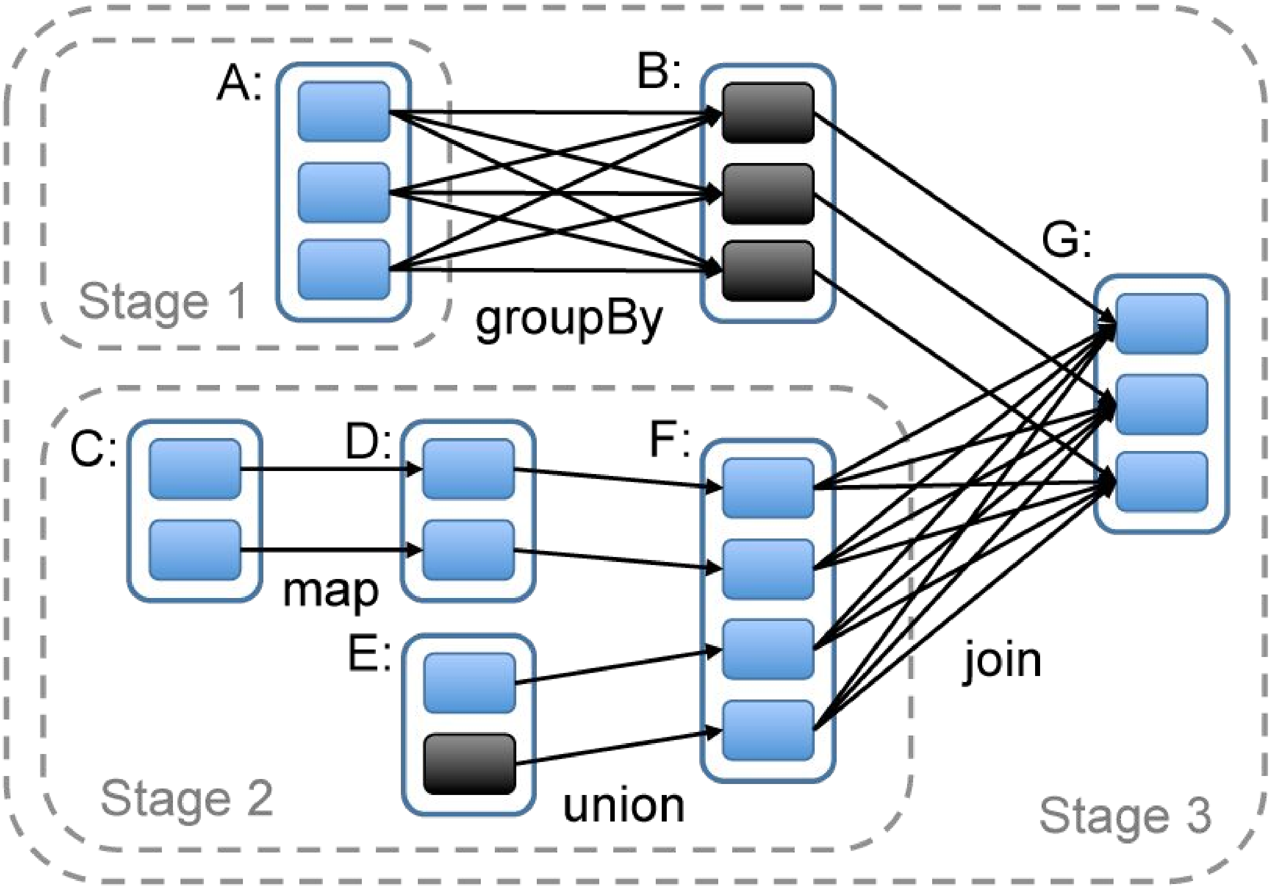
Function name	Purpose
`RDD.zip(RDD)`	Return key-value pairs with elem. from each RDD
`RDD.mapValues(func)`	Apply a function to each value without the key
`RDD.keys()`	Return an RDD of just keys.
`RDD.values()`	Return an RDD of just values.
`RDD.sortByKey()`	Return an RDD sorted by the keys.

Actions

Function name	Purpose
`RDD.countByKey()`	Count the number of elements for each key
`RDD.collectAsMap()`	Collect the result as a dictionary for easy lookup.
`RDD.lookup(key)`	Return all values associated with the provided key.

Shuffling

Remember this?



Shuffling

- Certain operations within Spark trigger an event known as the shuffle.
- It is a mechanism for re-distributing data so that it is grouped differently across partitions.
- Involves copying data across executors and machines, making the shuffle a complex and costly operation.
- To optimize shuffling, we need to customize how Spark split the RDD in partition. Only possible with pair RDDs.

Key-Value Pairs Hands-on!

4-Key-Value_Pairs.ipynb

5-SparkSQL.ipynb

6-Dataframes.ipynb

Machine Learning

Calcul Québec

ML solves problems that cannot be solved by numerical means alone.

Which of these people are good friends with each other?

Will this person like this movie?

Is this cancer?

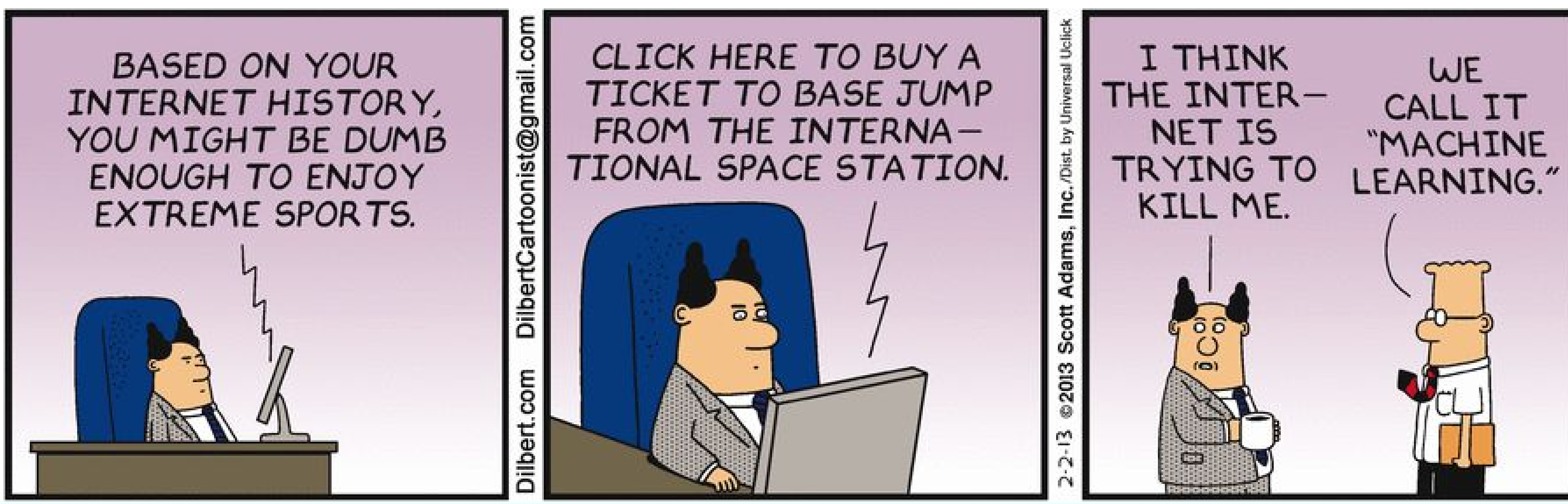
How do you fly this thing?

What is the market value of this house?

Who is this?

What did you say?

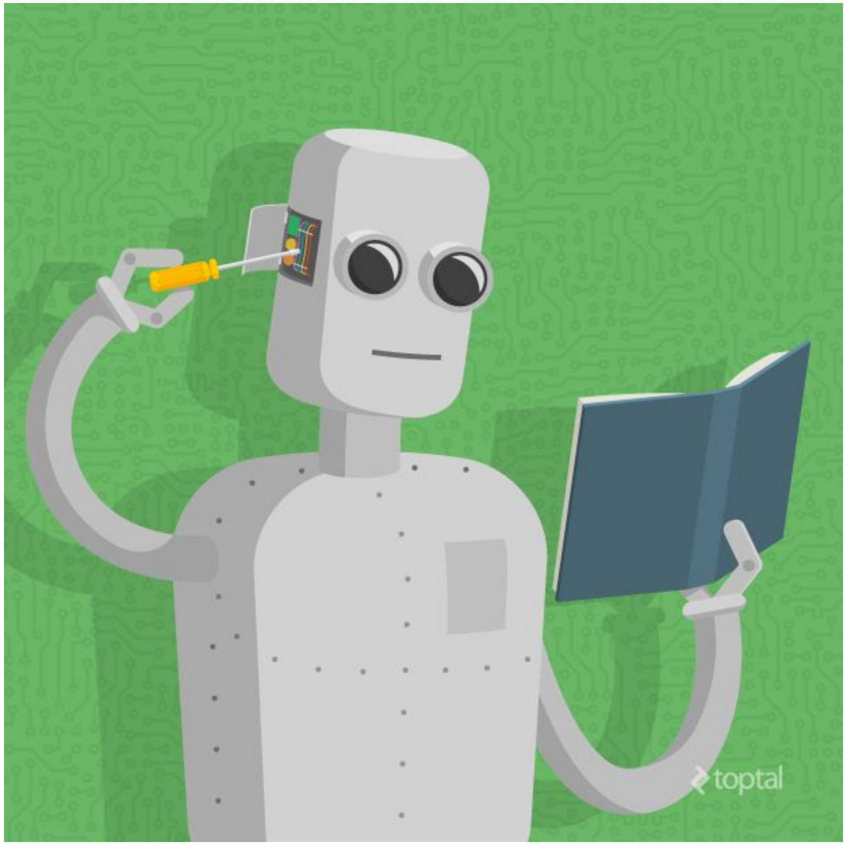
Will this rocket engine explode on take off?



Machine Learning with Spark

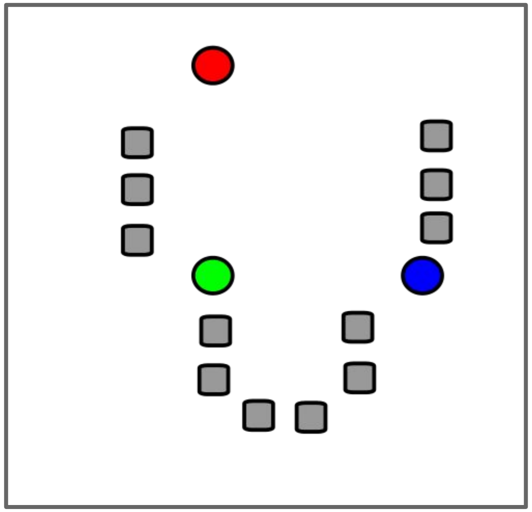
Calcul Québec

MLlib

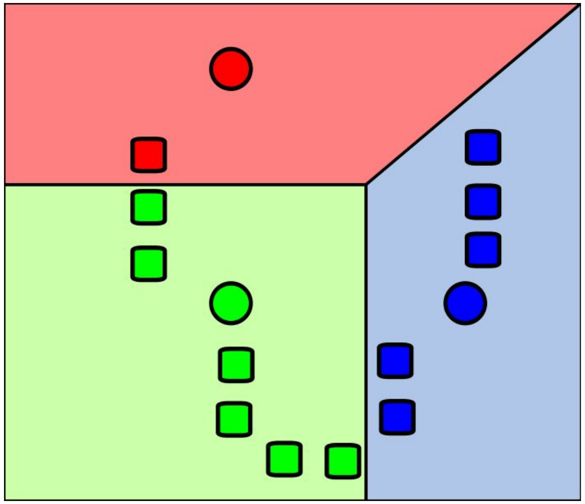


- ❑ Classification and Regression
- ❑ Collaborative Filtering
- ❑ Clustering
- ❑ Dimensionality Reduction
- ❑ Frequent Pattern Mining

Clustering: k-means

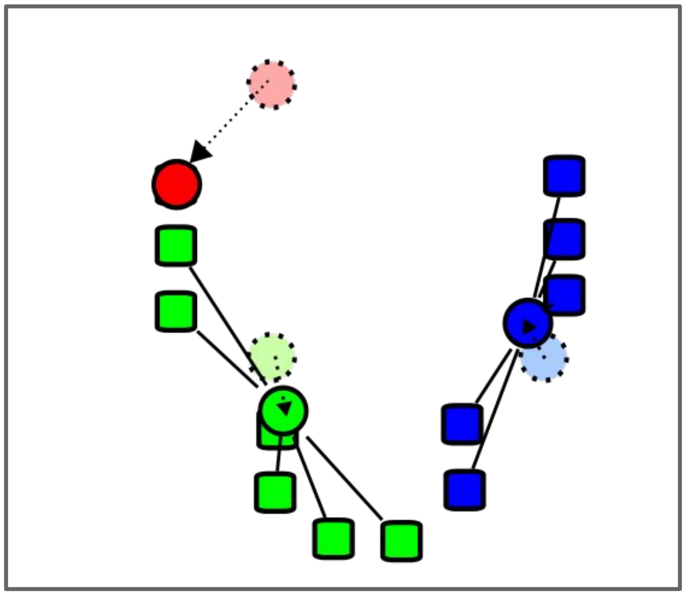


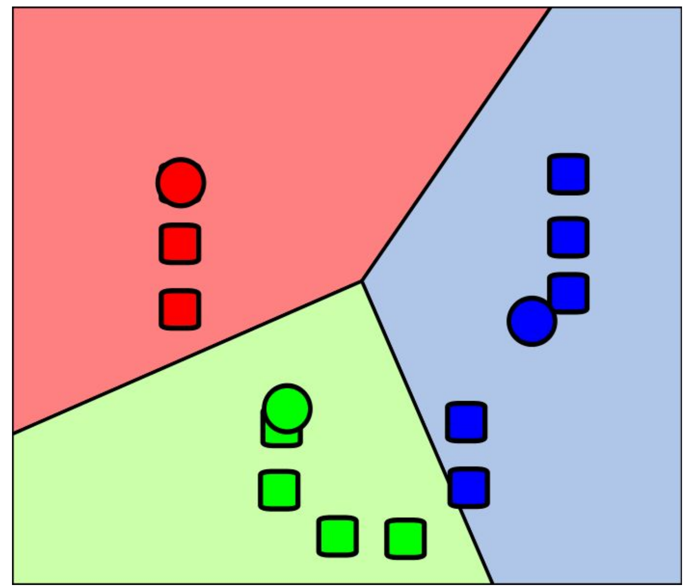
Step 1



Step 2

Clustering: k-means


Step 3


Step 4

```
KMeans.train(rdd, 3, maxIterations=10, runs=1)
```

7-Data_Algorithms.ipynb

Workshop Summary

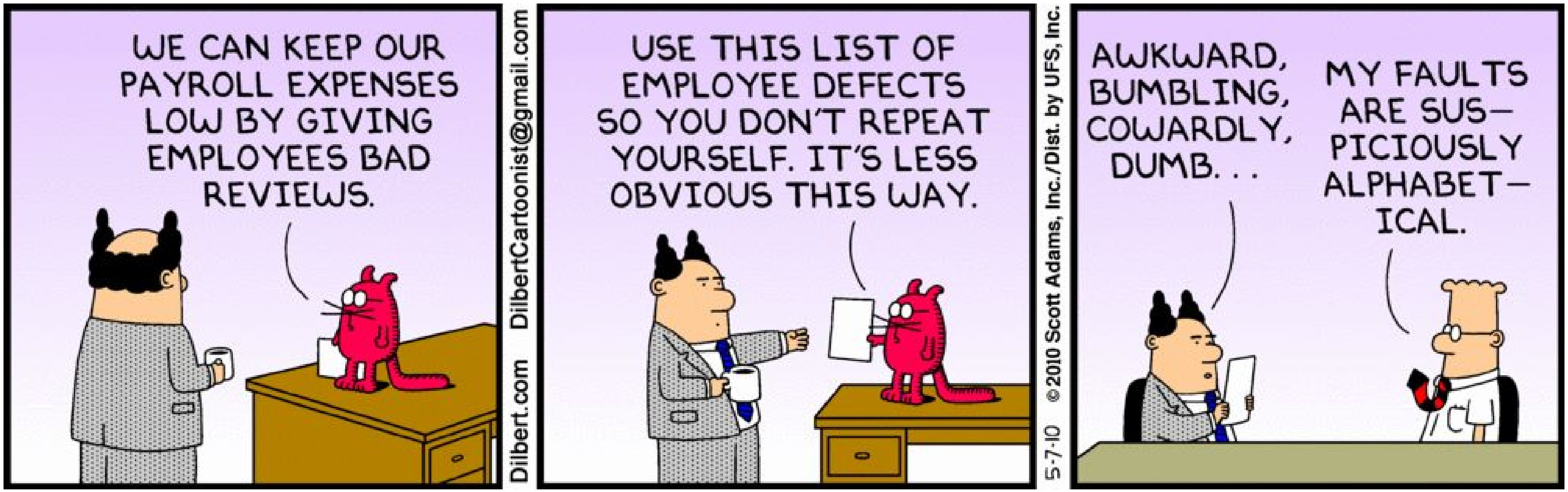
Calcul Québec

1. Introduction to Big Data
    a. Map-Reduce Paradigm
2. Introduction to Apache Spark
    a. Resilient Distributed Dataset (RDD)
    b. API
3. Apache Spark SQL
    a. Working with structured data
    b. DataFrames
4. Machine Learning
    a. Integrating Spark in data algorithms
    b. Using Spark MLlib

Survey

Thanks for letting us know your impressions on the day

https://goo.gl/ooTmjS



Pointeurs intéressants

Calcul Québec

- Prédiction des finalistes de la coupe du monde: https://github.com/GoogleCloudPlatform/ipython-soccer-predictions
- Blog sur le filtrage collaboratif: http://bugra.github.io/work/notes/2014-04-19/alternating-least-squares-method-for-collaborative-filtering/
- Article sur les chercheurs de Capital One: http://www.bloombergview.com/articles/2015-01-23/capital-one-fraud-researchers-may-also-have-done-some-fraud