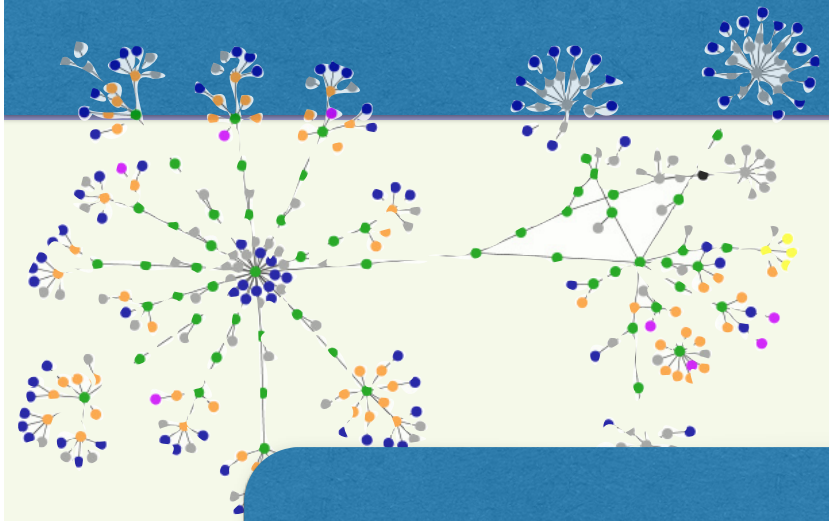
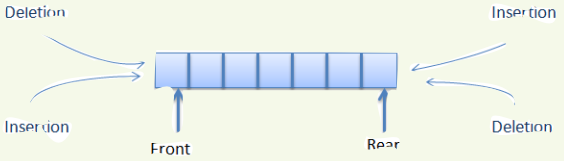
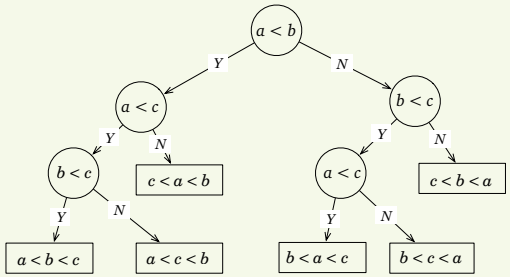




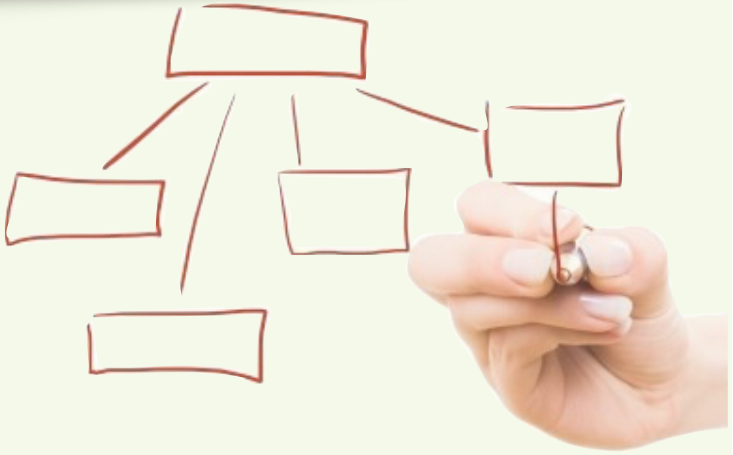


COURSE
"Técnicas Matemáticas para Big Data"
University of Aveiro

Algorithm 2.1: INSERTION-SORT(A)

```
1  for j ← 2 to A.size do
2      key ← A[j]
       // Insert A[j] into the sorted sequence A[1..j − 1]
3      i ← j − 1
4      while i > 0 and A[i] > key do
5          A[i + 1] ← A[i]
6          i−−
7      A[i + 1] ← key
```

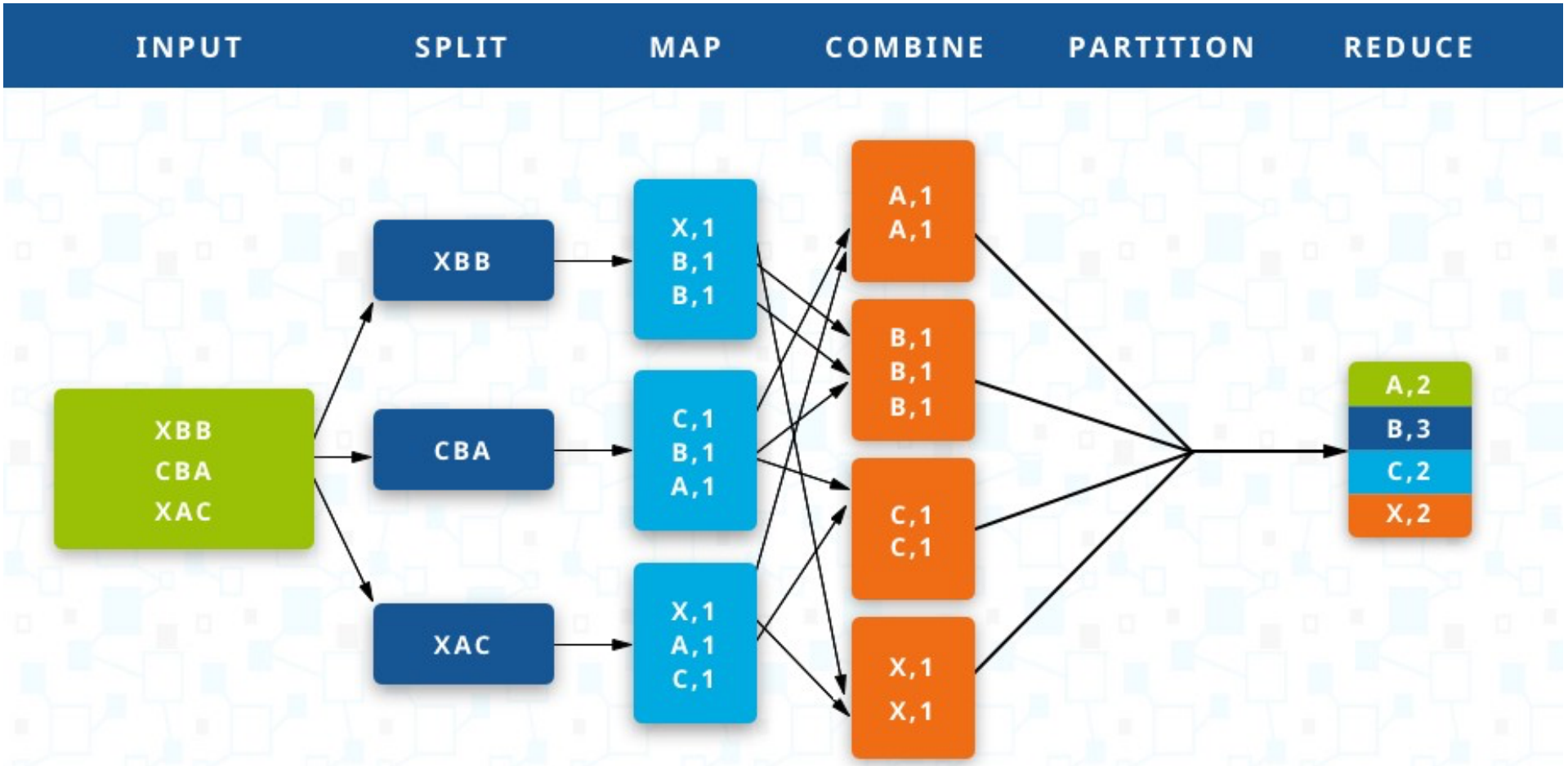
Master in Mathematics and Applications



MapReduce Scheme

A year after Google published a white paper describing the MapReduce framework (2004), Doug Cutting and Mike Cafarella created Apache Hadoop. Hadoop has moved far beyond its beginnings in web indexing and is now used in many industries for a huge variety of tasks that all share the common theme of variety, volume and velocity of structured and unstructured data. Hadoop is increasingly becoming the go-to framework for large-scale, data-intensive deployments.

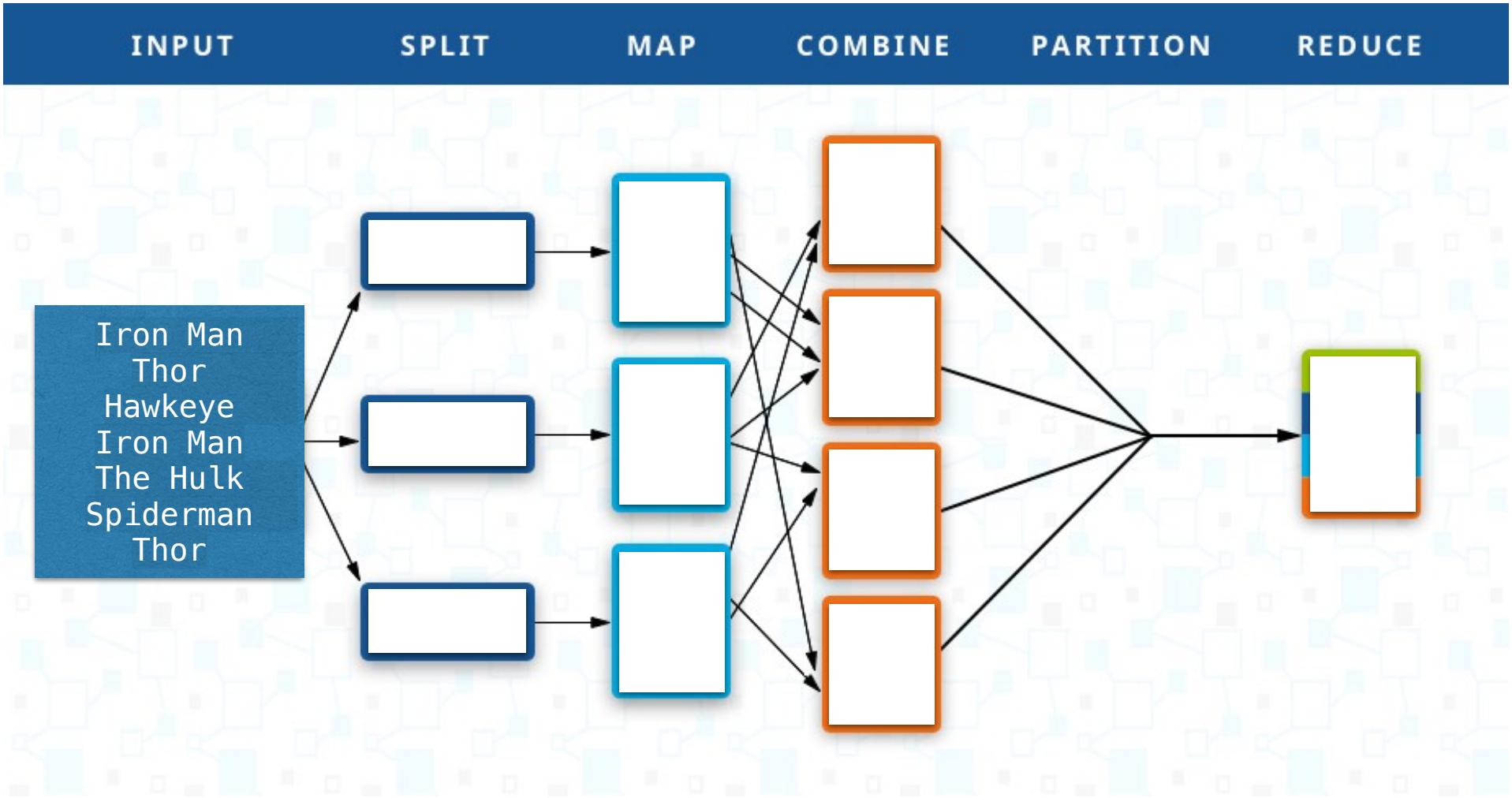
Problem: count letters



(key,value)

MapReduce is a big data processing technique, and a model for how to programmatically implement that technique. Its goal is to sort and filter massive amounts of data into smaller subsets, then distribute those subsets to computing nodes, which process the filtered data in parallel.

Problem: find duplicates



Implementations

The following software and data systems implement MapReduce:

- **Hadoop** — Developed by the Apache Software Foundation. Written in Java, with a language-agnostic API.
- **Spark** — Developed by AMPLab at UC Berkeley, with APIs for Python, Java, and Scala.
- **Disco** — A MapReduce implementation originally developed by Nokia, written in Python and Erlang.
- **MapReduce-MCI** — Developed at Sandia National Laboratories, with bindings for C, C++, and Python.
- **Phoenix** — A threaded MapReduce implementation developed at Stanford University, written in C++.

Limitations

Due to its constant shuffling of data, MapReduce is poorly suited for iterative algorithms, such as those used in ML (machine learning).

Alternatives

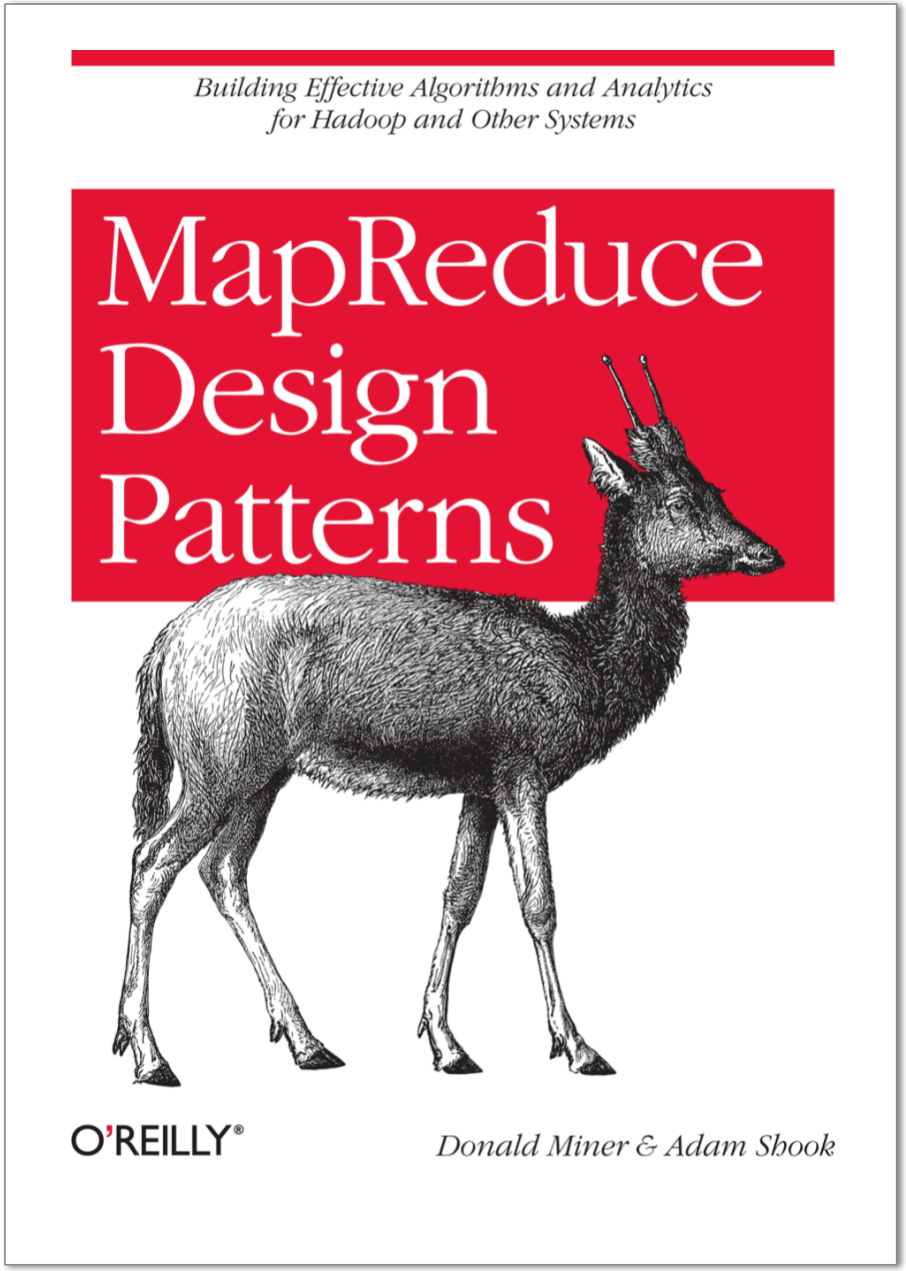
Alternatives to traditional MapReduce, that aim to alleviate these bottlenecks, include:

- **Amazon EMR** — High-level management for running distributed frameworks on the AWS (Amazon Web Services) platform, such as Hadoop, Spark, Presto, and Flink.
- **Flink** — An open-source framework for stream processing of incomplete or in-transit data. Optimized for large datasets, developed by Apache.
- **Google Cloud Dataflow** — Distributed data processing for GCP (Google Cloud Platform).
- **Presto** — An open-source distributed SQL query engine for big data

- **Spark** — RDD and DGG approach (current).

Book Recommendation:

Try some exercises

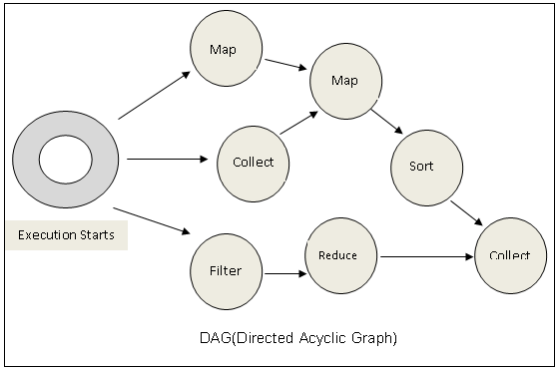


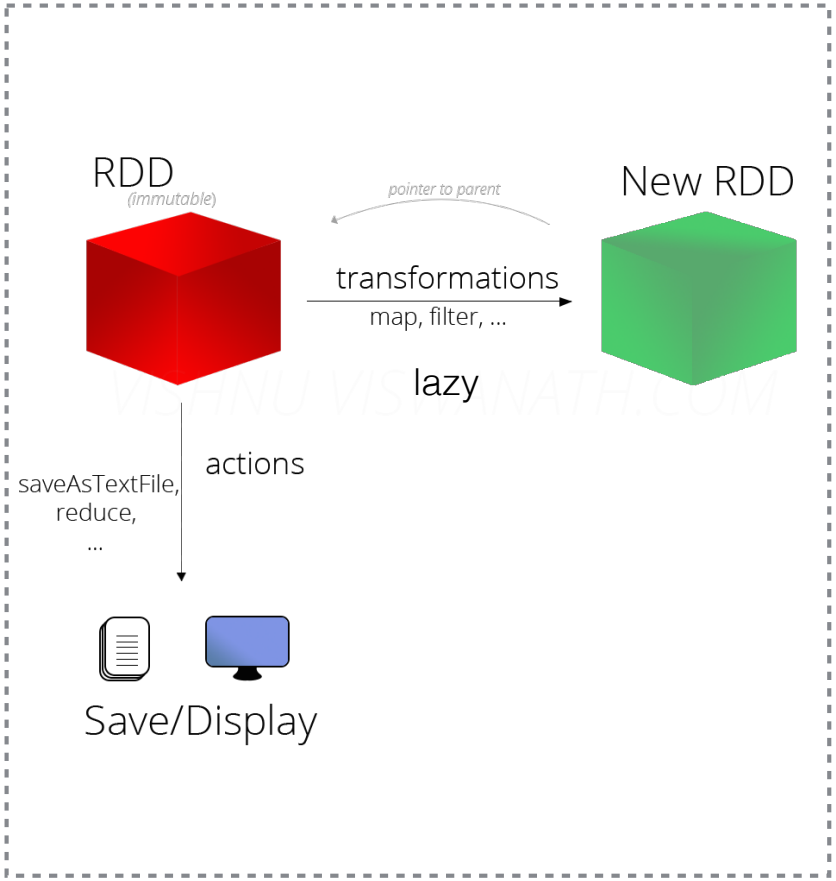
2013 with 251 pg

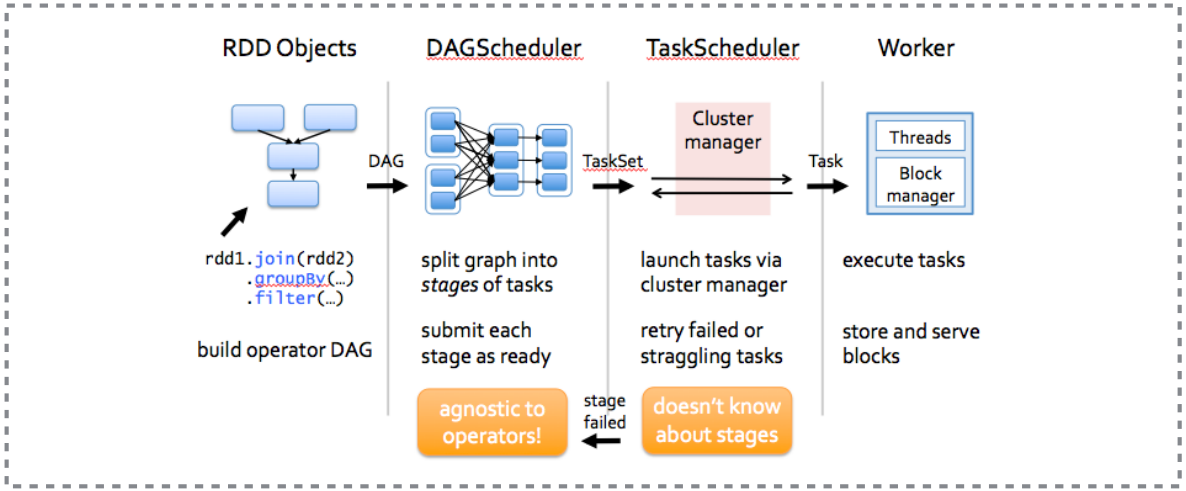
1. Resilient Distributed Datasets (RDD)

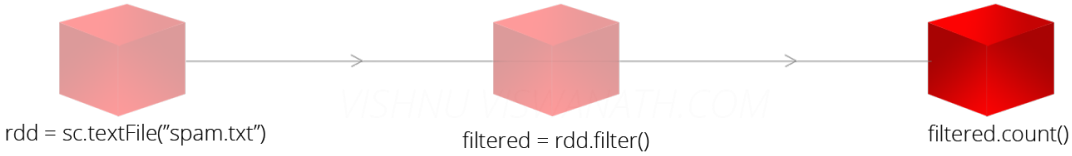
- RDDs are divided into smaller chunks called **Partitions;**
- Executing some **action**, a task is launched per partition;
- More partitions then more **parallelism** (Spark automatically decides the number of partitions);
- Partitions of an RDD are **distributed** across all the nodes in the network.

2. Directed Acyclic Graph (DAG)










```
val rdd = sc.textFile("spam.txt")
val filtered = rdd.filter(line => line.contains("money"))
filtered.count()
```

https://spark.apache.org