UNIVERSITÀ
DEGLI STUDI
DI PADOVA

J. Pazzini
PADOVA UNIVERSITY

10 - MAP-REDUCE & HADOOP

Management and Analysis of Physics Datasets - Module B

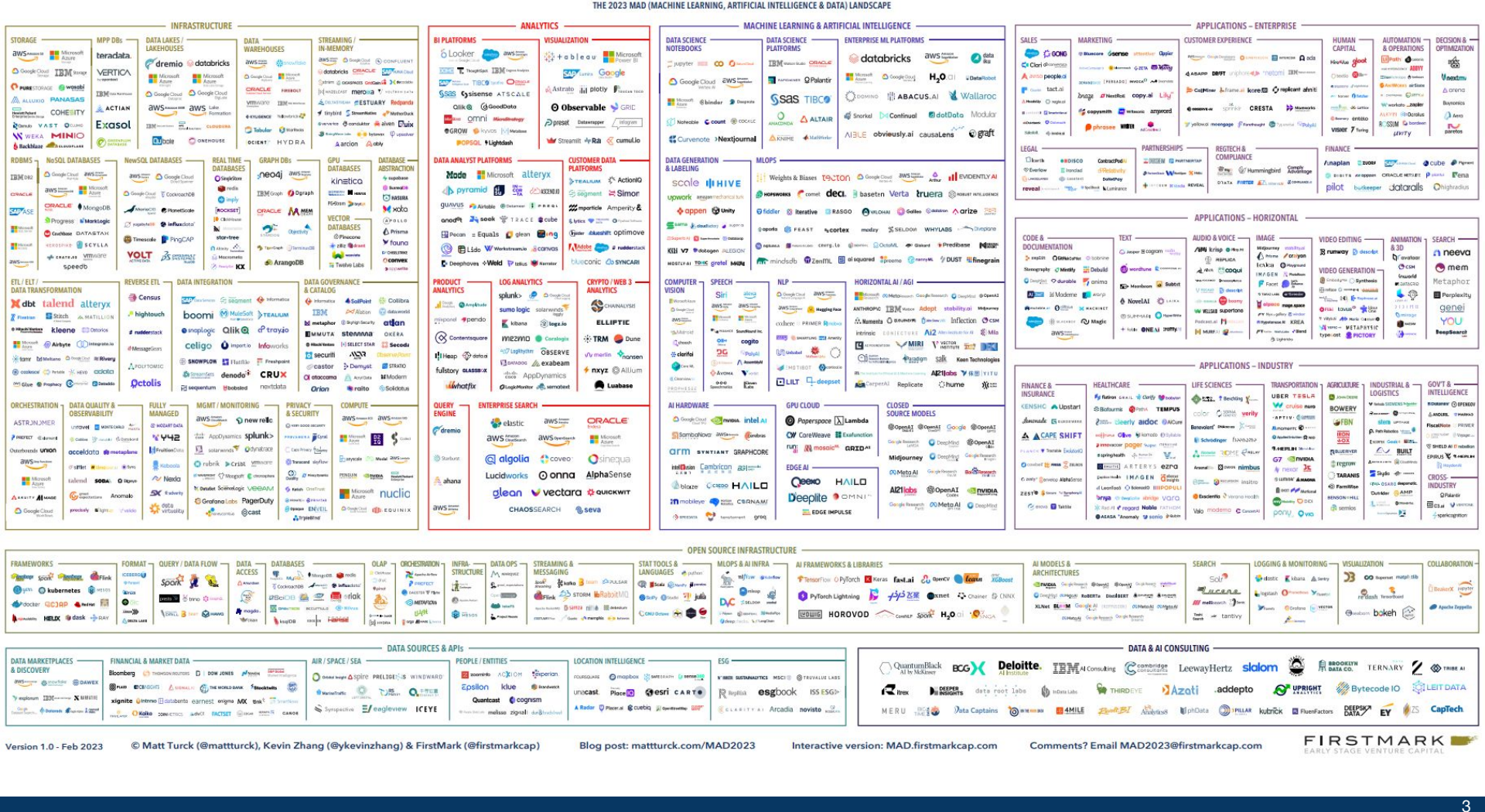
Physics of Data

A.A. 2023/2023

BIG DATA - 3Vs

Recently (from the ~2000s) several improvements over the processing of large datasets came from **BigData** technologies, born and maintained mostly from institutions and companies "outside" of the scientific community

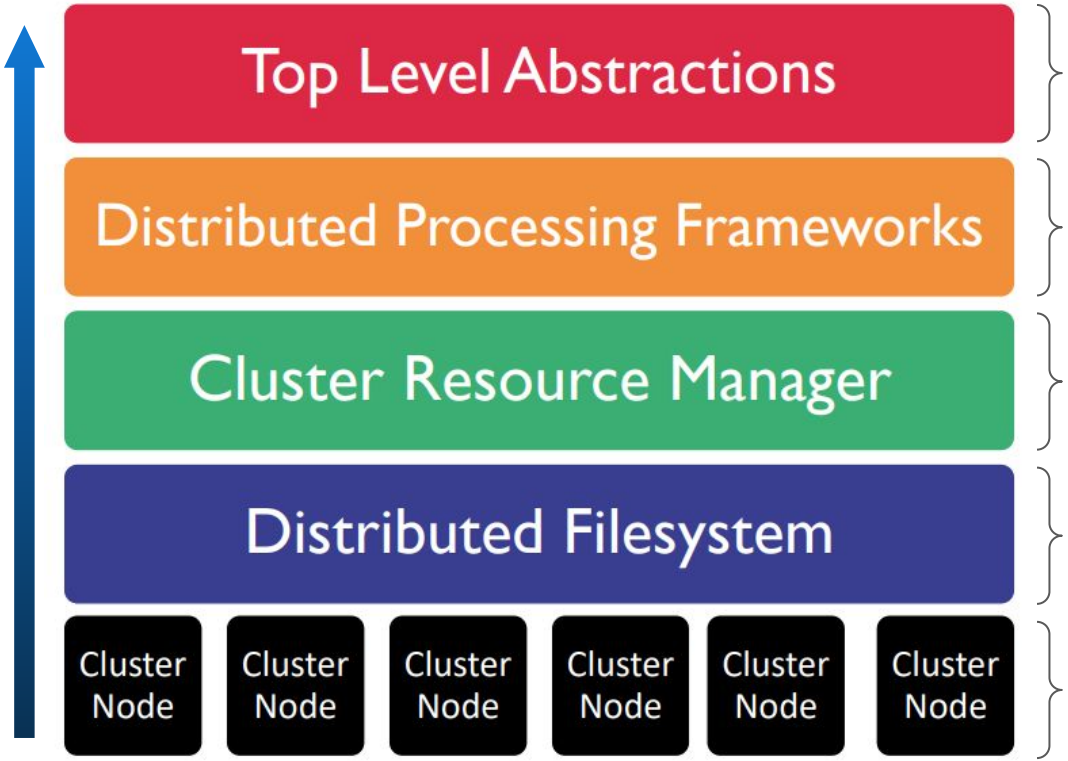
DISTRIBUTED PROCESSING - DATA SCIENCE & BIG DATA



Version 1.0 - Feb 2023 © Matt Turck (@mattturck), Kevin Zhang (@ykevinzhang) & FirstMark (@firstmarkcap) Blog post: mattturck.com/MAD2023 Interactive version: MAD.firstmarkcap.com Comments? Email MAD2023@firstmarkcap.com

FIRSTMARK EARLY STAGE VENTURE CAPITAL

Key ingredients to Big Data processing / analytics



APIs / “programming language”

SW framework allowing distributed processing

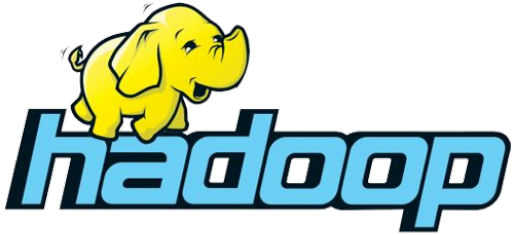
Cluster “orchestrator / scheduler”

DFS (or Distributed DBs)

Computing nodes (physical / cloud)

HADOOP

Born in 2006 within Yahoo! and based on a 2003 paper about the Google Distributed File System



Was (and still is, up to some extent) widely regarded as the main framework for storage and parallel distributed processing of BigData

Facebook
A 1100-machine cluster with 8800 cores and about 12 PB raw storage
A 300-machine cluster with 2400 cores and about 3 PB raw storage

Twitter
Multiple clusters storing over 500 PB divided in four groups (real time, processing, data warehouse and cold storage)
Largest cluster: >10000 nodes

Yahoo!
More than 100,000 CPUs in >40,000 computers running Hadoop
Largest cluster: 4500 nodes with (2x4cpu servers w/ 4x1TB disk + 16GB RAM)

https://cwiki.apache.org/confluence/display/HADOOP2/PoweredBy

HADOOP

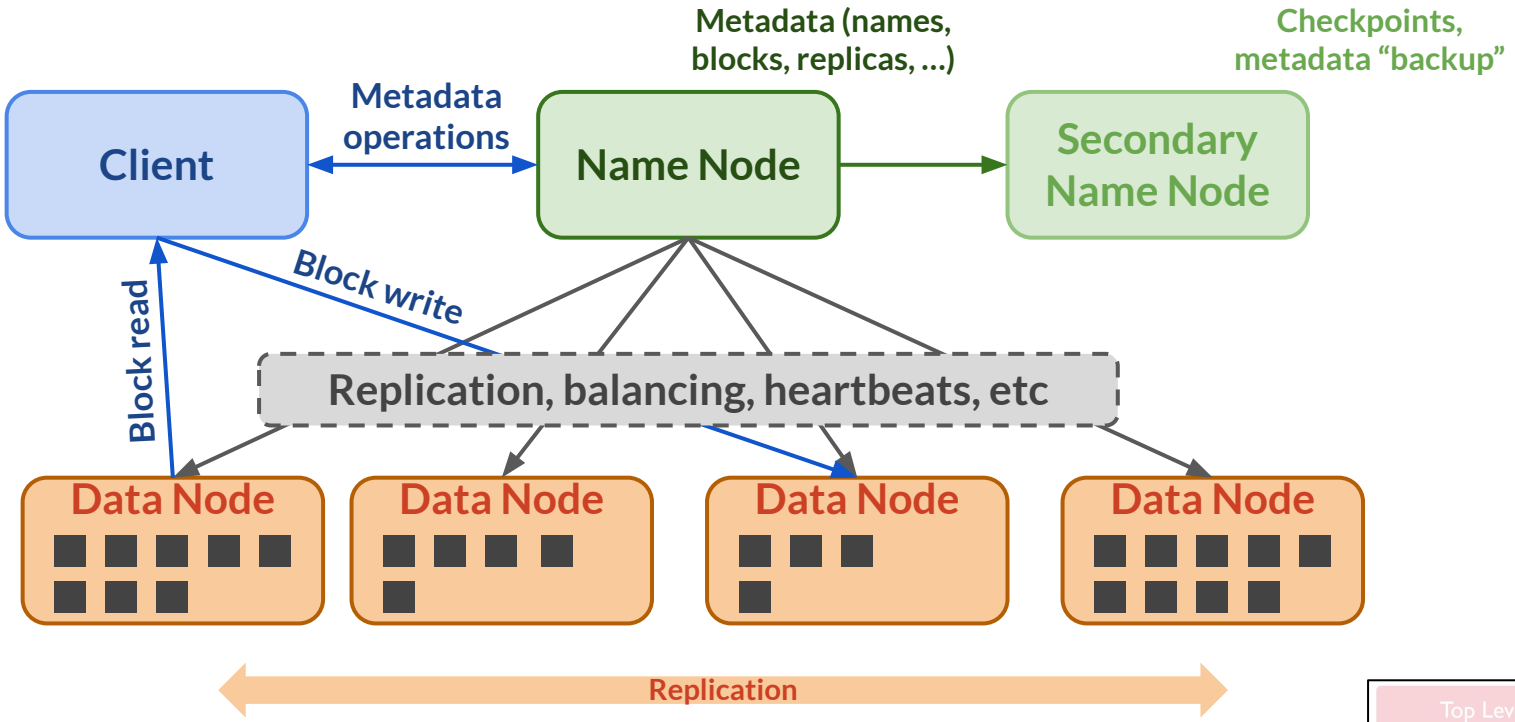
- Open source (managed by the Apache foundation)
- Developed and written in Java
- Designed to efficiently scale with the number of computing resources (horiz.)

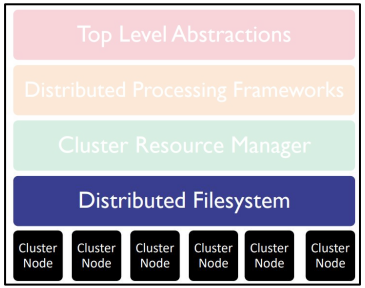
Main components:

- Hadoop Common → *Java-based libraries and utilities*
- Hadoop DFS (HDFS) → *storage*
- Hadoop YARN → cluster manager + "*scheduler*"
- Hadoop MapReduce → *distributed programming model*

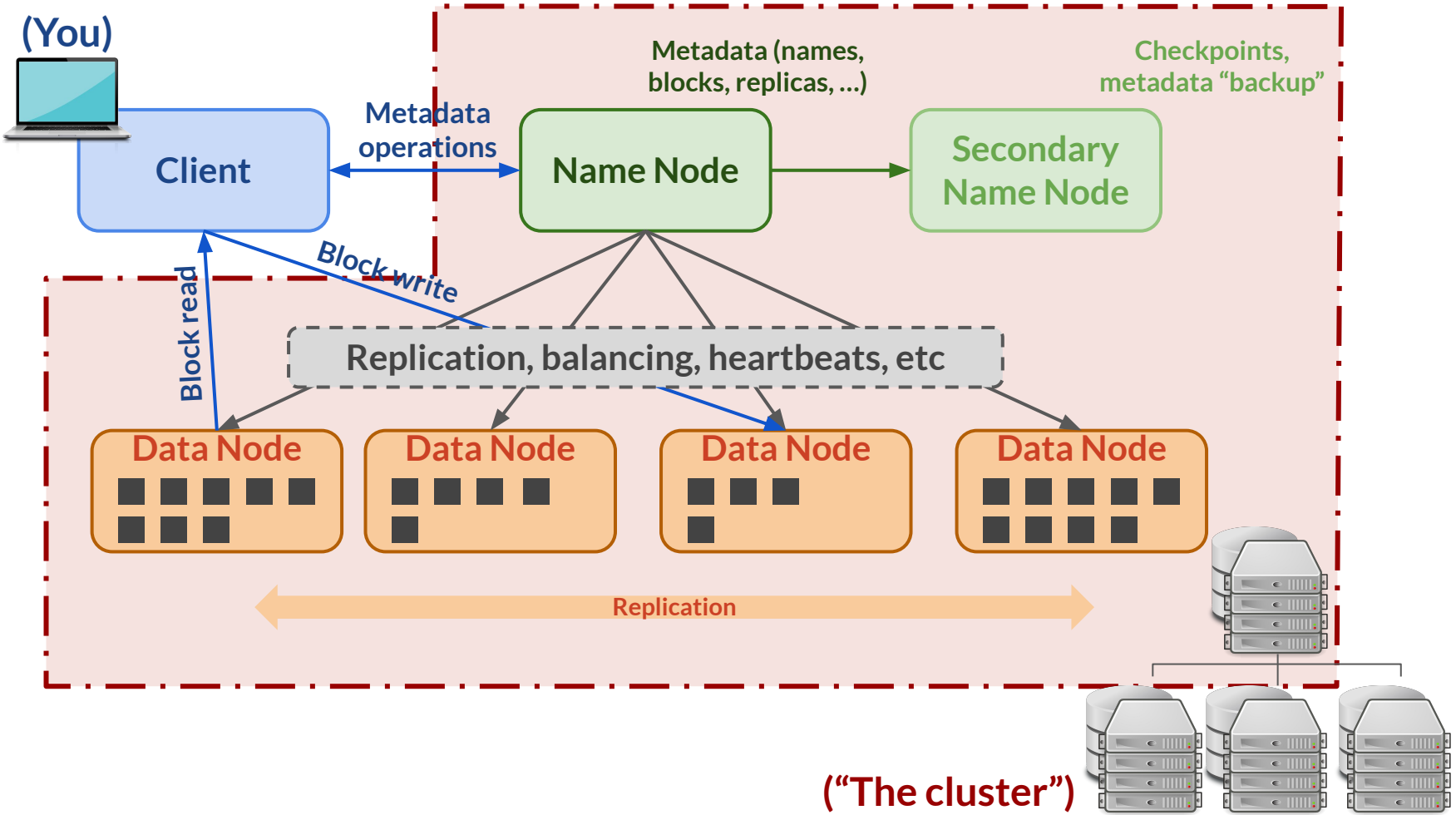
(+ other components as HDBase, …)

HDFS, A REMINDER…





HDFS, A REMINDER…



MAP-REDUCE

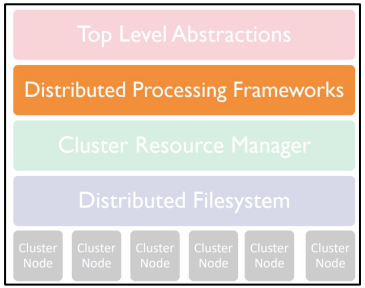
Paradigm providing a programming model for distributed processing in Hadoop

→ not a programming language per se, but rather a paradigm for developing distributed algorithms for potentially very large datasets

Based on 2 main phases:

1. **Map**

2. **Reduce**



MAP-REDUCE

Paradigm providing a programming model for distributed processing in Hadoop

→ not a programming language per se, but rather a paradigm for developing distributed algorithms for potentially very large datasets

Based on 2 main phases:

1. **Map**

   Split the input data into a series of <key, value> pairs.

   Process each <key, value> pair and return a list of <key, value> pairs

$$(k_{input}, v_{input}) \xrightarrow{map} \text{list}\,(k_{intermediate}, v_{intermediate}) = [(k_{int}^{1}, v_{int}^{1}), (k_{int}^{2}, v_{int}^{2}), (k_{int}^{3}, v_{int}^{3}), \dots]$$

2. **Reduce**

MAP-REDUCE

Paradigm providing a programming model for distributed processing in Hadoop

→ not a programming language per se, but rather a paradigm for developing distributed algorithms for potentially very large datasets

Based on 2 main phases:

1. **Map**

   Split the input data into a series of <key, value> pairs.

   Process each <key, value> pair and return a list of <key, value> pairs

$$(k_{input}, v_{input}) \xrightarrow{map} \text{list}\,(k_{intermediate}, v_{intermediate}) = [(k_{int}^{1}, v_{int}^{1}), (k_{int}^{2}, v_{int}^{2}), (k_{int}^{3}, v_{int}^{3}), \dots]$$

2. **Reduce**

   Take as input the list of intermediate <key, value> pairs.

   Run aggregation of the values from all pairs with same key to produce a final list of <key, value> pairs, or a single value

$$\text{list}\,(k_{int}, v_{int}) \rightarrow \text{list}\,\big(k_{int}, \text{list}(v_{int})\big) \xrightarrow{reduce} \text{list}\,(k_{output}, v_{output}) \quad \text{or} \quad v_{output}$$

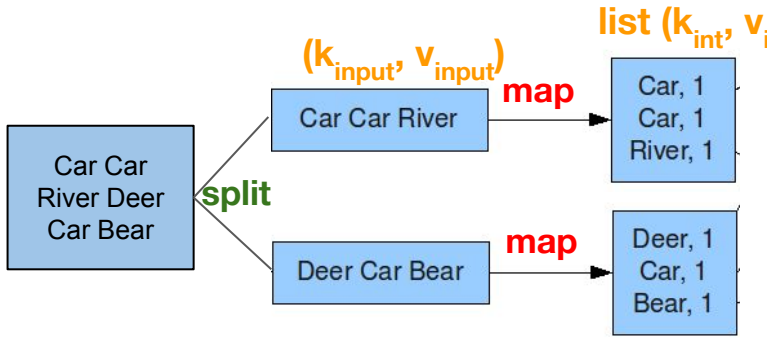
MAP-REDUCE

Paradigm providing a programming model for distributed processing in Hadoop

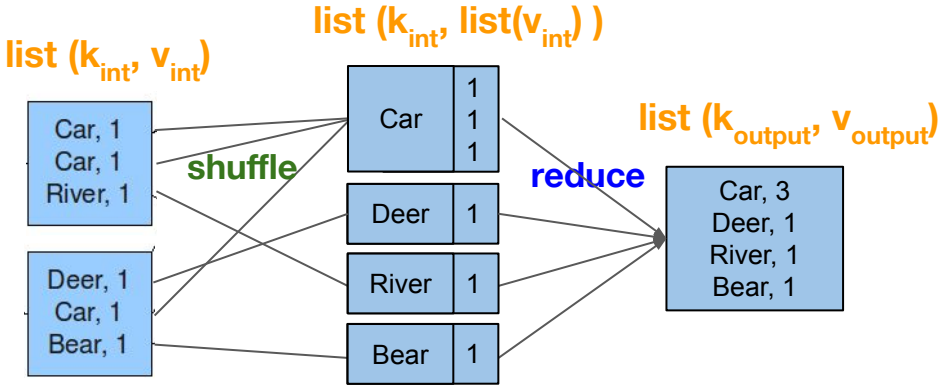
→ not a programming language per se, but rather a paradigm for developing distributed algorithms for potentially very large datasets

Based on 2 main phases:

1. **Map**

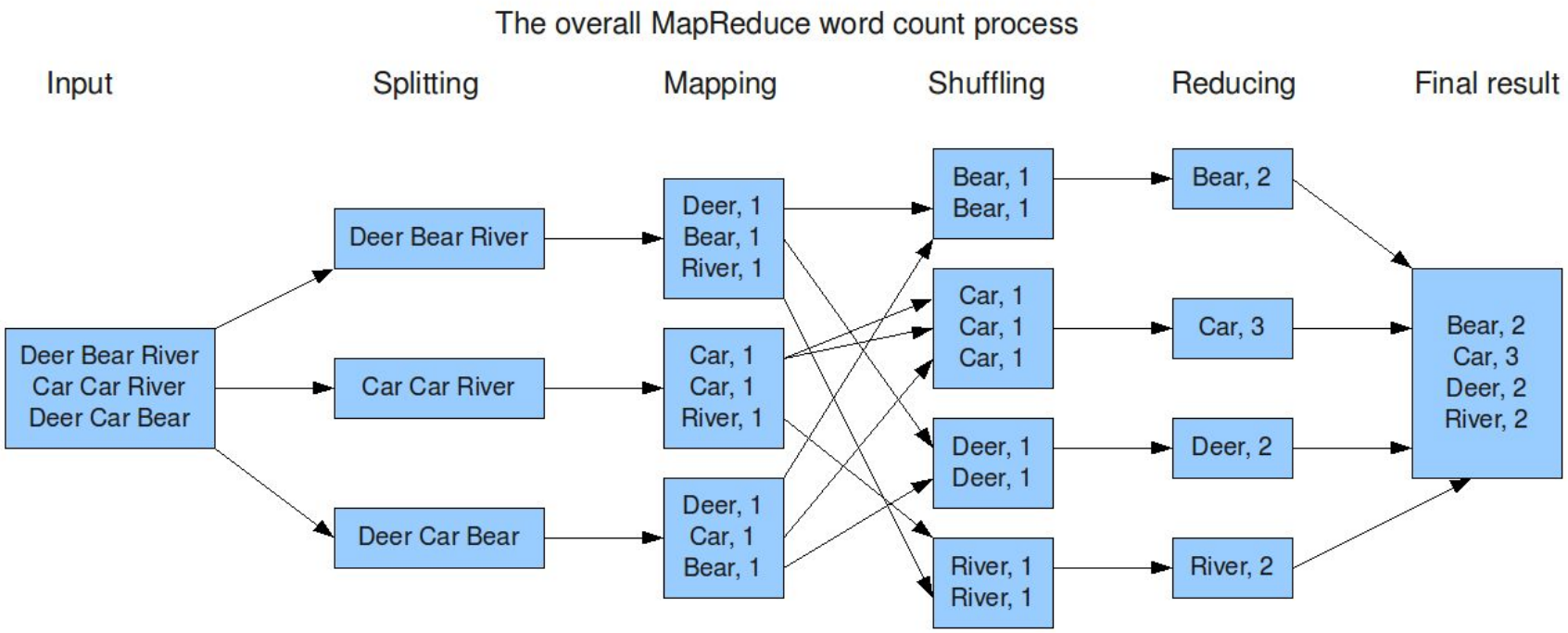


2. **Reduce**



Word count "mandatory" example

MAP-REDUCE

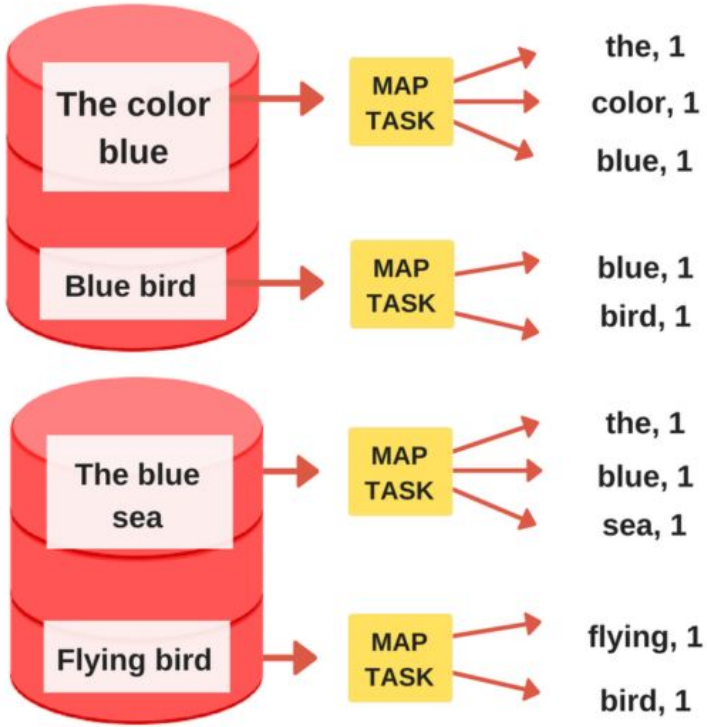

The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

MAP

Data is read from the distributed file system, partitioned among a set of computing nodes in the cluster, and sent to them as a set of key-value pairs

The Map tasks process the input records **independently of each other** and produce intermediate results as key-value pairs

The intermediate results are **stored on the local disk of the node** running the Map task

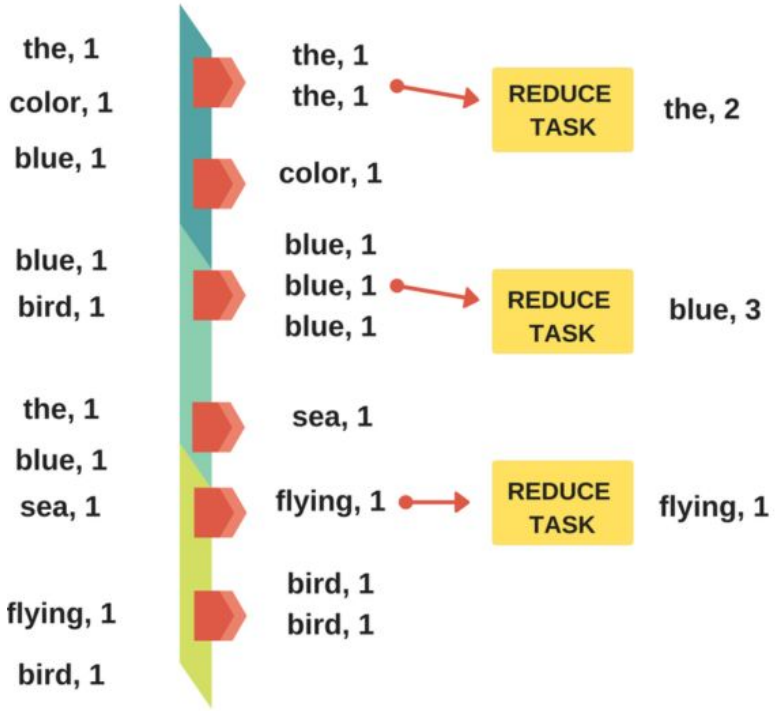


DATA LOCALITY

REDUCE

When all Map tasks are completed, the Reduce phase begins with the **shuffle and sort** step in which the intermediate data is sorted by the key and the key-value pairs are grouped and shuffled to the reduce tasks

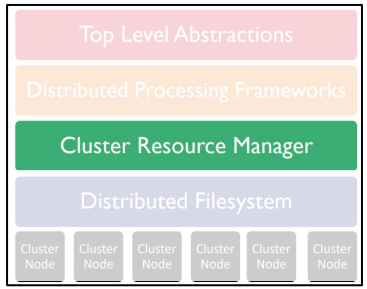
The reduce tasks then take the key-value pairs **grouped by the key** and run the reduce function for each group of key-value pairs



YARN

Yet **A**nother **R**esource **N**egotiator

→ takes on the role of the "OS" for Hadoop, managing the execution and scheduling of jobs

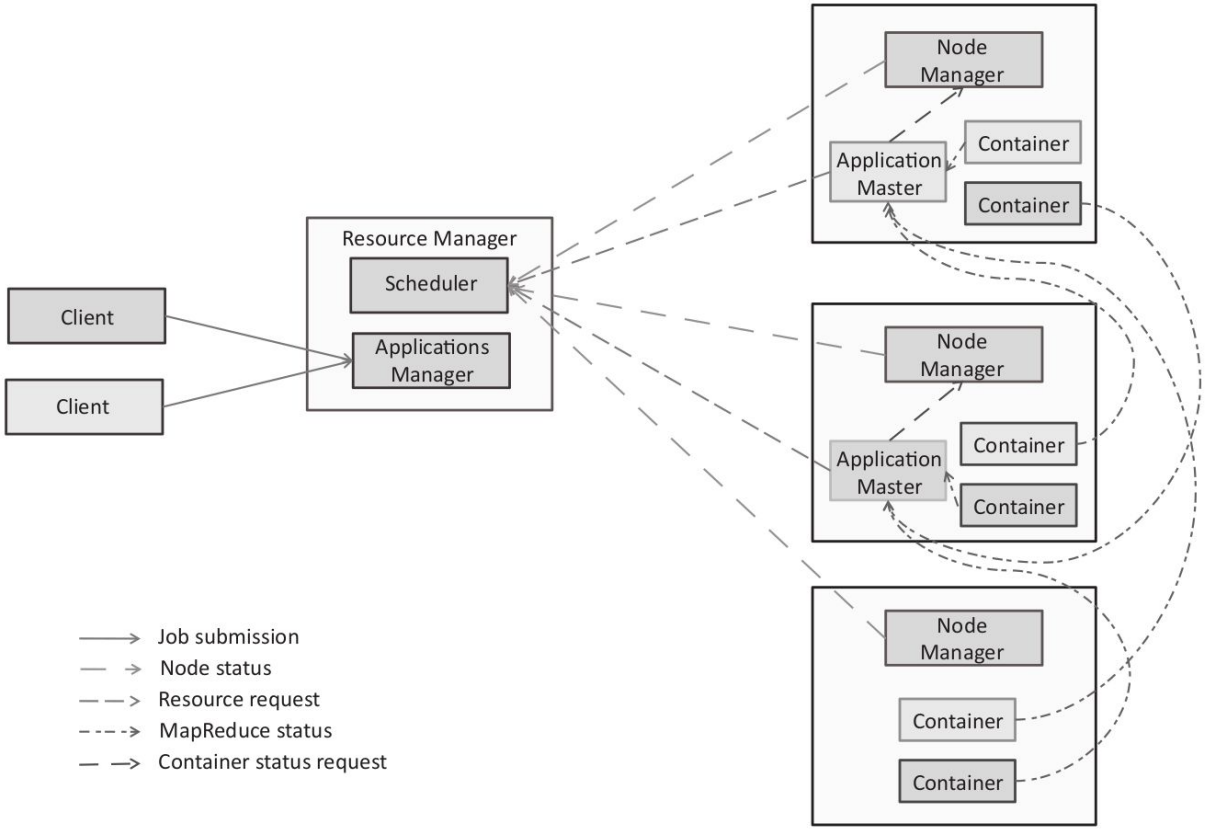


The key components of YARN:

- **Resource Manager (RM)** → manages the assignment of computing resources to applications
  - *Scheduler*
  - Applications Manager
- **Application Master (AM)** → negotiates resources from the RM and works with the NMs to execute and monitor the tasks
- **Node Manager (NM)** → manages the user processes on a machine
- **Container** → set of resources allocated by the RM (memory, CPU) to run a specific task

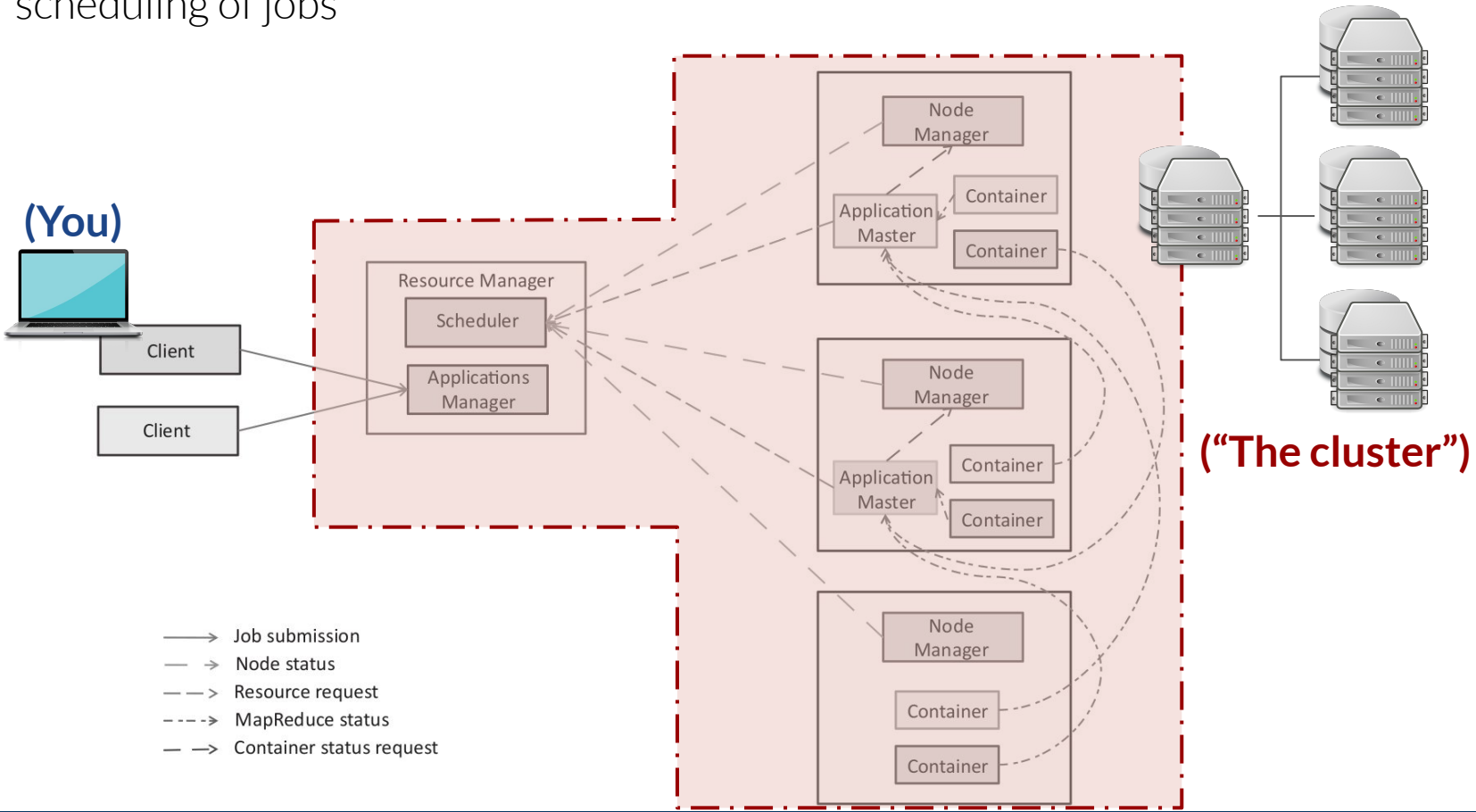
Yet **A**nother **R**esource **N**egotiator

→ takes on the role of the "OS" for Hadoop, managing the execution and scheduling of jobs



Yet **A**nother **R**esource **N**egotiator

→ takes on the role of the "OS" for Hadoop, managing the execution and scheduling of jobs



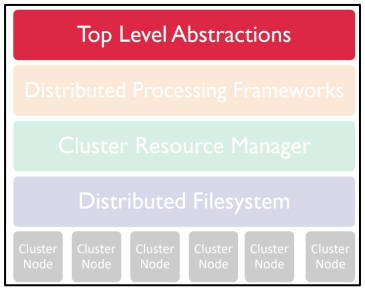
HADOOP HIGH-LEVEL ABSTRACTIONS

Pig (latest release in 2017)
high-level language to write data analysis scripts, translated into MapReduce programs
[developed at Yahoo! to write and run MapReduce jobs on large datasets]

Hive
SQL-like query language (Hive Query Language), for querying data residing in HDFS, further organized into tables as in Relational DBs
[initially developed by Facebook, later used and developed by Netflix and others]

Oozie
workflow scheduler system that allows managing Hadoop jobs with more than one MapReduce job to be chained together

...



THE HADOOP ECOSYSTEM

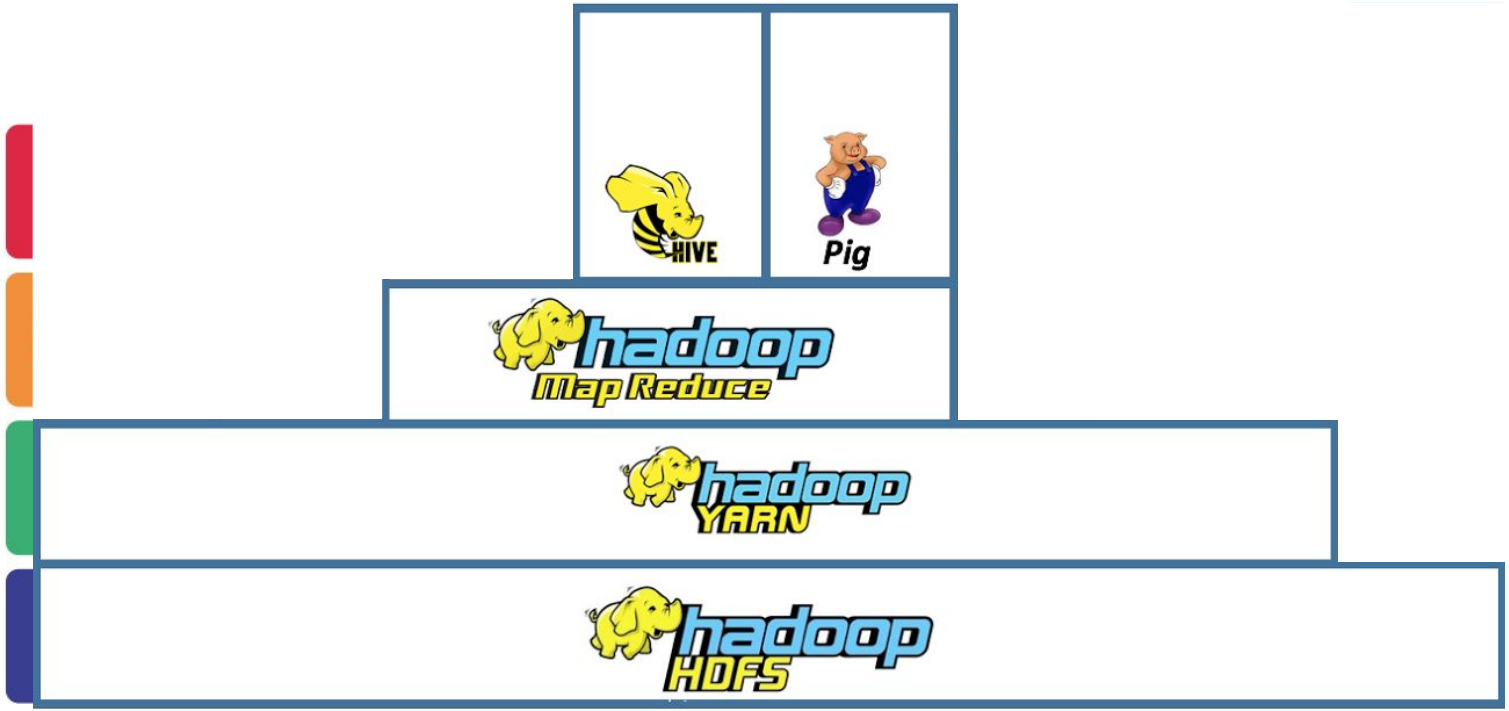
APIs / “programming language”

SW framework allowing distr. processing

Cluster “orchestrator / scheduler”

DFS

THE HADOOP ECOSYSTEM



http://hadoopilluminated.com/

A BIGDATA ECOSYSTEM