

# **Big Data, MapReduce y el ecosistema Hadoop**



**Bases de Datos No Relacionales**  
Instituto de Computación, FING, Udelar – 2021  
CC-BY Lorena Etcheverry [lorenae@fing.edu.uy](mailto:lorenae@fing.edu.uy)

# **Agenda**

- **Big Data: algunas definiciones**
- El paradigma de programación MapReduce
- Patrones de diseño sobre MapReduce
- Hadoop y *Hadoop Distributed Filesystem*



VOLUME

DATA SIZE

VELOCITY

SPEED OF CHANGE

VARIETY

DIFFERENT FORMS  
OF DATA SOURCES

VERACITY

UNCERTAINTY OF  
DATA

Fuente: <http://www.usine-digitale.fr/>

## SO WHAT IS A PETABYTE ANYWAY?

Source – [www.mozy.com](http://www.mozy.com)

# WHAT IS A PETABYTE?

TO UNDERSTAND A PETABYTE WE MUST FIRST UNDERSTAND A GIGABYTE.

1 GIGABYTE = 7 MINUTES OF HD-TV VIDEO

2 GIGABYTES = 20 YARDS OF BOOKS ON A SHELF

4.7 GIGABYTES = SIZE OF A STANDARD DVD-R

THERE ARE A MILLION GIGABYTES IN A PETABYTE

*"Let me repeat that: we create as much information in two days now as we did from the dawn of man through 2003."*

(That's something like 5 Exabytes of Data). - Eric Schmidt  
– Google 8/10

# A PETABYTE IS A LOT OF DATA

1 PETABYTE = 20 MILLION FOUR-DRAWER FILING CABINETS FILLED WITH TEXT

1 PETABYTE = 13.3 YEARS OF HD-TV VIDEO

1.5 PETABYTES = SIZE OF THE 10 BILLION PHOTOS ON FACEBOOK

15+ PETABYTES = INTERNET USER'S DATA BACKED UP ON MOZY.COM

20 PETABYTES = THE AMOUNT OF DATA PROCESSED BY GOOGLE PER DAY

20 PETABYTES = TOTAL HARD DRIVE SPACE MANUFACTURED IN 1995

50 PETABYTES = THE ENTIRE WRITTEN WORKS OF MANKIND, FROM THE BEGINNING OF RECORDED HISTORY, IN ALL LANGUAGES

Twitter:  
Over 7TB a Day in Tweets.

A ZETABYTE IS ONE MILLION PETABYTES!

Facebook:  
More than 750 Million Users.

Average user creates 90 Pieces of content each month.  
More than 30B pieces of content shared each month.



## Data Management

Aquisition & recording

Extraction, cleaning & annotation

Integration, aggregation & representation

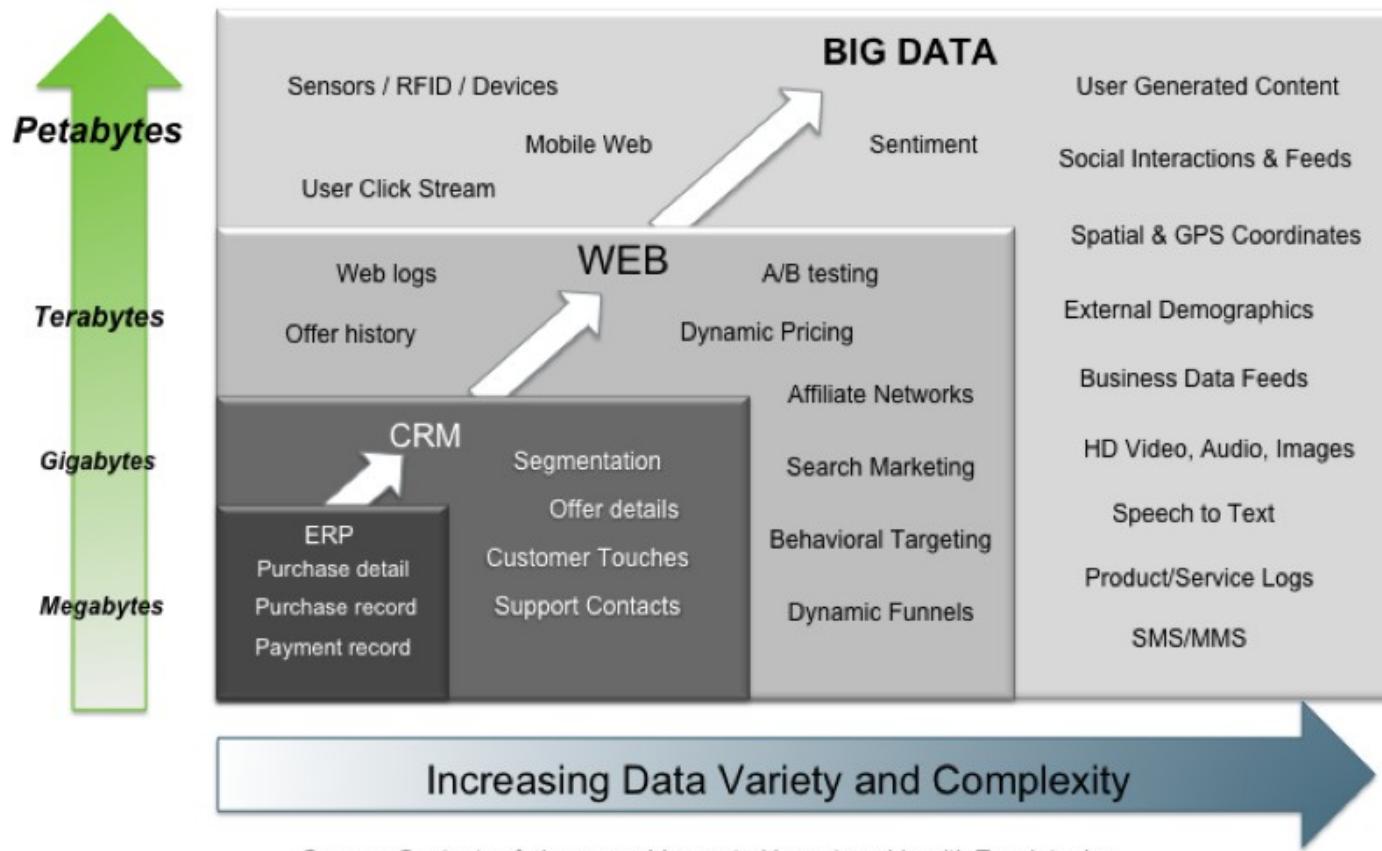
## Analytics

Modelling & analysis

Interpretation

# Sobre la naturaleza del Big Data

Big Data = Transactions + Interactions + Observations



Fuente: <http://hortonworks.com/blog/7-key-drivers-for-the-big-data-market/>

# Big Data y la frescura del dato

- El “mundo viejo” trabaja sobre transacciones, datos históricos
- Este nuevo mundo en algunos casos trabaja sobre datos casi en *real-time*
  - *Ej:*
    - *stream analytics*
    - *sentiment analysis*



# **Big Data para la academia**

Michael Stonebraker considera que *Big Data* quiere decir al menos tres cosas:

- gran **volumen**
  - análisis de datos simple (SQL)
  - análisis de datos complejo (no SQL)
- gran **velocidad**
  - “drink from a fire hose”
- gran **variedad**
  - integrar una gran cantidad de fuentes diversas

M. Stonebraker Big Data Means at Least Three Different Things....,

<http://www.nist.gov/itl/ssd/is/upload/NIST-stonebraker.pdf>

<http://www.bizjournals.com/boston/blog/startups/2013/03/michael-stonebraker-what-is-big-data.html>

# **Big Data representa una disrupción**

- Michael Stonebraker: "Big Data, Technological Disruption and the 800 Pound Gorilla in the Corner" (setiembre 2018)

[https://www.youtube.com/watch?  
v=KRcecxdGxvQ](https://www.youtube.com/watch?v=KRcecxdGxvQ)

# Big Data y la industria

- 85% of Fortune1000 companies have big data initiatives planned or in progress
- 83% say they'll consider making greater use of real-time data in 2013
- 63% say use of info (including big data) and **analytics** is creating competitive advantage. 73% say leveraging data has increased value.
- 84% say using data helps make better business decisions
- 65% say company leadership sees data management/analysis as source of value, not a drain on resource

Big data: What you need to know from the new Econsultancy report, 2013.

## Big Data, Big Opportunity

Most "big data" research currently centers around the advantages of high-quality data and the resulting percentage of firms that see improved ROI after investing in better data-gathering techniques.

But no research has focused on the impact an investment in analytics technologies that improve the usability of and accessibility to a company's existing data has on performance – both in sales and productivity.

**10%** Can lead to large returns

For the median FORTUNE 1000 COMPANY, a 10% increase in usability of and accessibility to data means significant boosts in productivity and sales.

What does this mean?

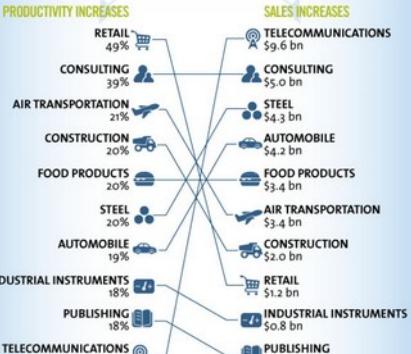
### PRODUCTIVITY INCREASE

A 10% increase in **USABILITY** of data translates to an increase of **\$2.01 billion** in total revenue per year.

### SALES INCREASE

A 10% increase in **ACCESSIBILITY** to data translates to an additional **\$65.67 million** in net income per year.

### Let's Look At Some Specific Industries



### Standout Industries In Sales Increases



To see what data mobility, usability and accessibility mean according to this infographic, check out the "Measuring the Business Impacts of Effective Data" study by The University of Texas at Austin commissioned by Sybase. Additional stats from US Department of Labor, State of the Ad Show, Mashable.com, Toyota and Rethink Vision.

SYBASE | Wikibon

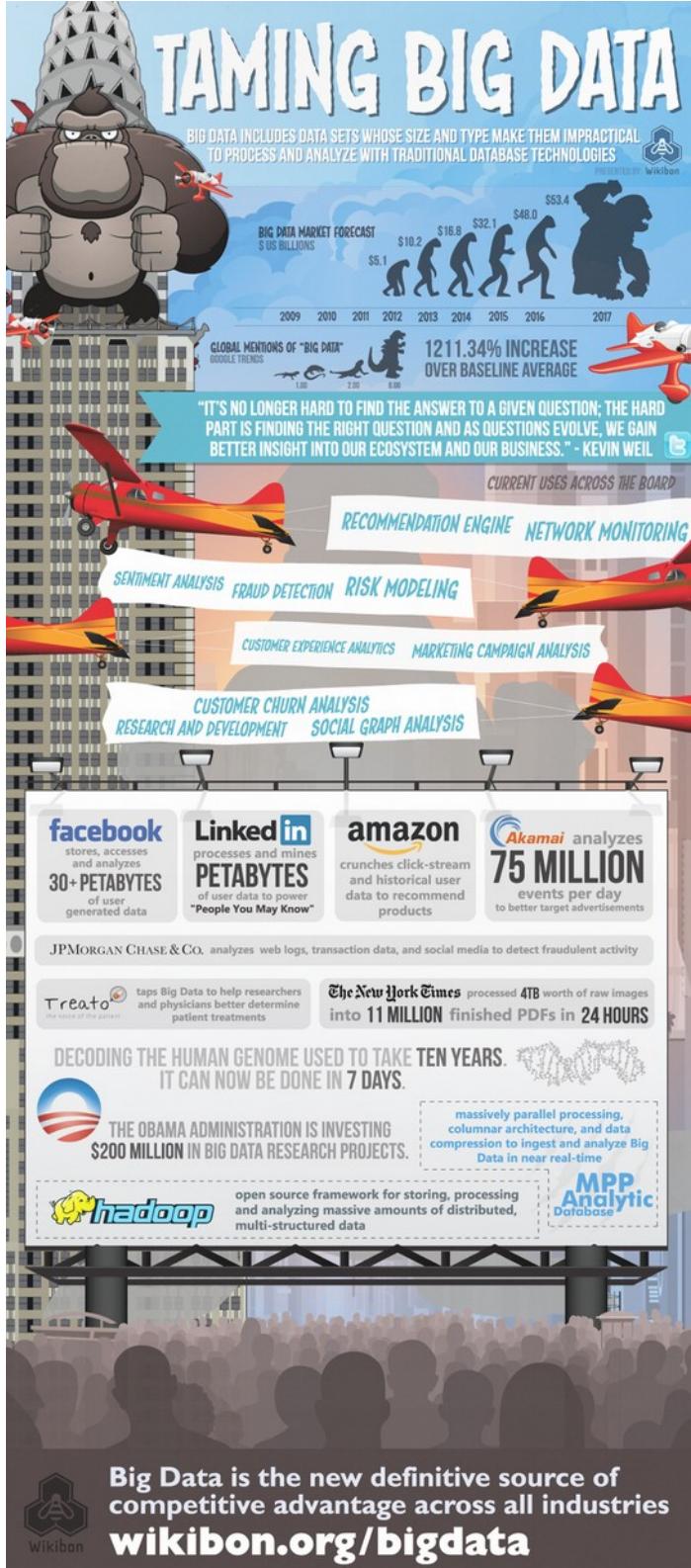
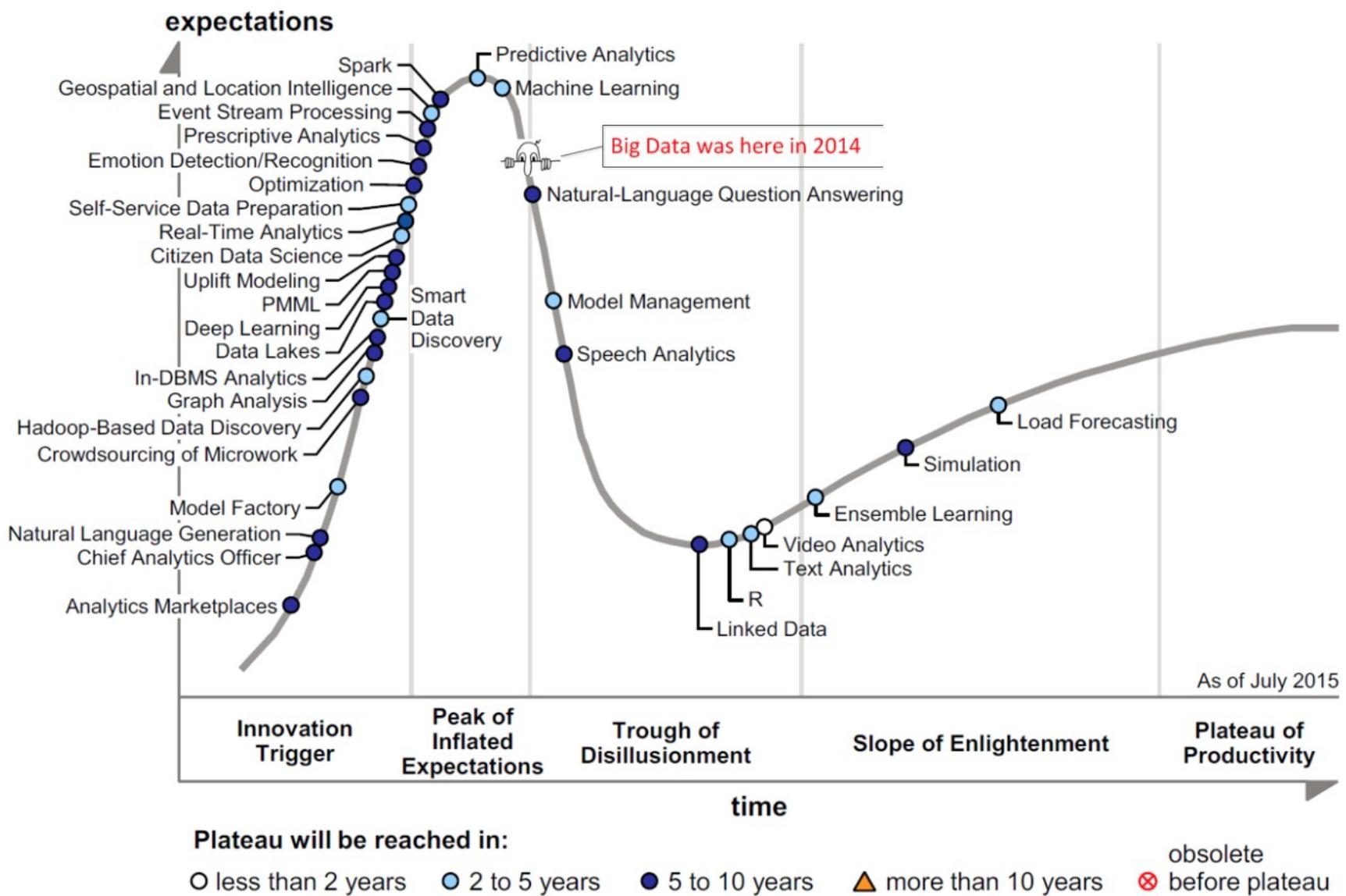


Figure 1. Hype Cycle for Advanced Analytics and Data Science, 2015



Source: Gartner (July 2015)

<https://www.datasciencecentral.com/profiles/blogs/big-data-falls-off-the-hype-cycle>

# **Agenda**

- Big Data: algunas definiciones
- El paradigma de programación MapReduce
- Patrones de diseño sobre MapReduce
- Hadoop y Hadoop Distributed Filesystem

# MapReduce

- Paradigma de programación paralela
- Entorno de ejecución distribuido.
- El modelo básico tiene dos fases:
  - Fase **map**: generar parejas (clave,valor) a partir de la entrada
  - Fase **reduce**: agrupar las parejas a partir del valor de la clave y producir una salida

Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters." OSDI (2004)

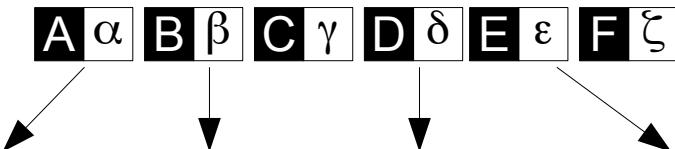
# MapReduce (ii)

- En el modelo básico se deben programar dos funciones:
  - map:  $(k_1, v_1) \rightarrow [(k_2, v_2)]$
  - reduce:  $(k_2, [v_2]) \rightarrow [(k_3, v_3)]$
- Vamos a aplicarlo a un problema simple: conteo de palabras

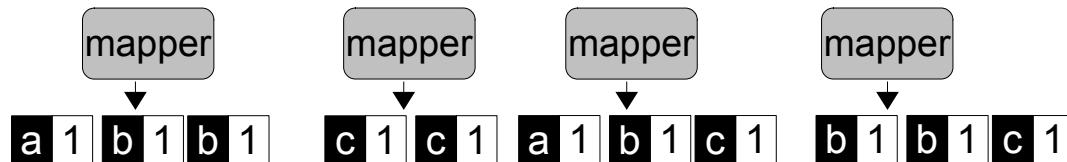
# Ejemplo: conteo de palabras

- Dado un texto contar la cantidad de ocurrencias de cada palabra

Split



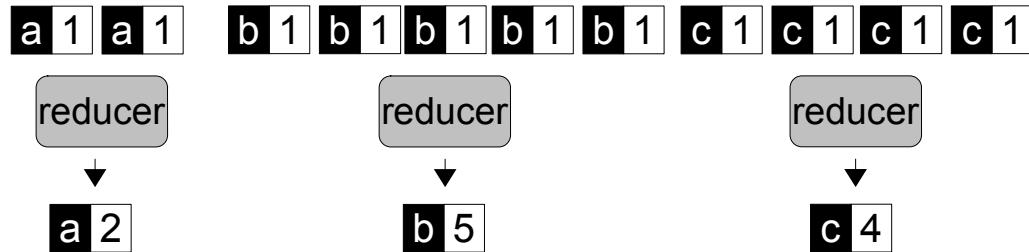
Map



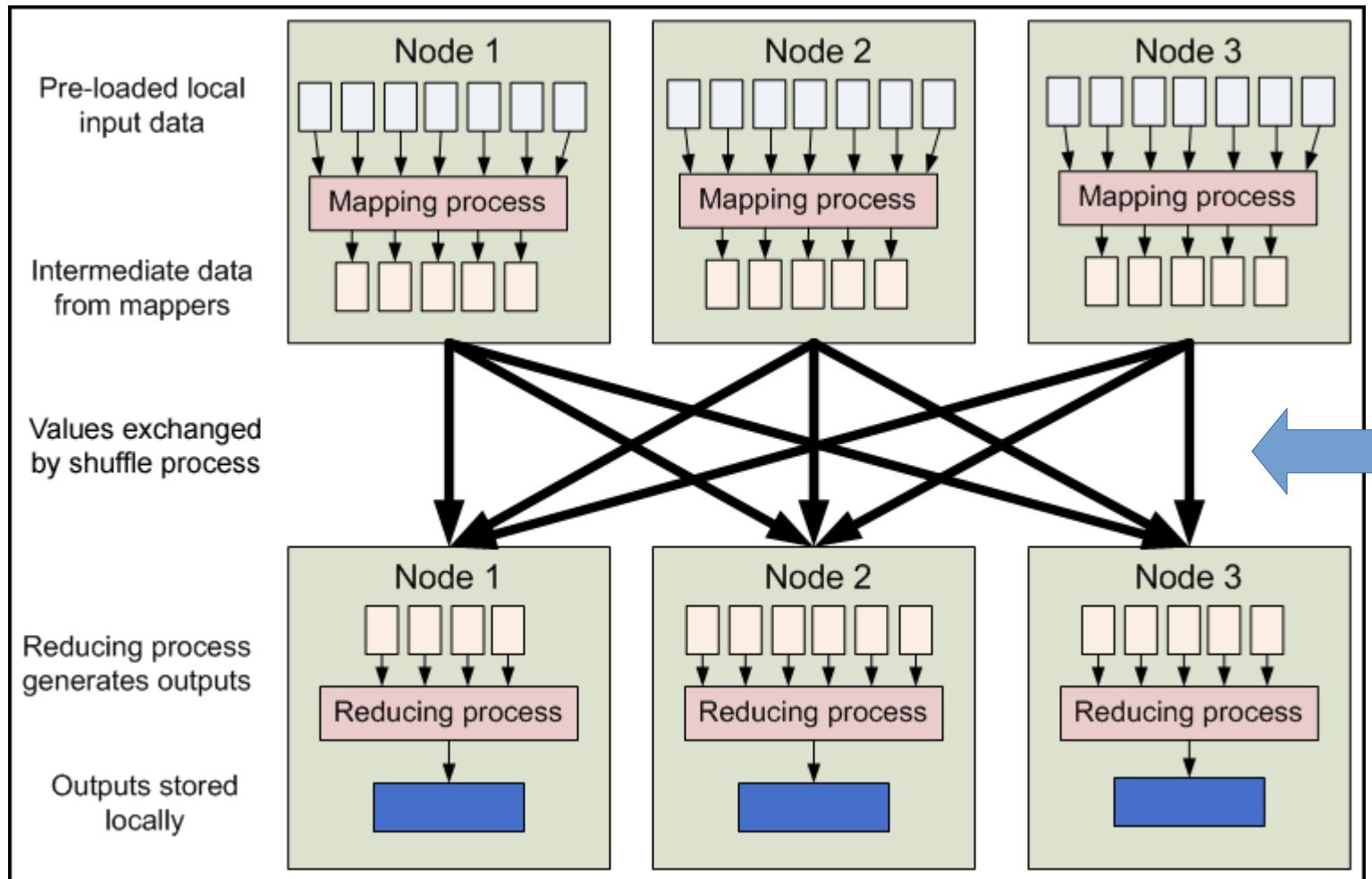
```
1: class MAPPER
2:   method MAP(docid a, doc d)
3:     for all term t ∈ doc d do
4:       EMIT(term t, count 1)
```

Shuffle & sort

Reduce



```
1: class REDUCER
2:   method REDUCE(term t, counts [c1, c2, ...])
3:     sum ← 0
4:     for all count c ∈ counts [c1, c2, ...] do
5:       sum ← sum + c
6:     EMIT(term t, count sum)
```

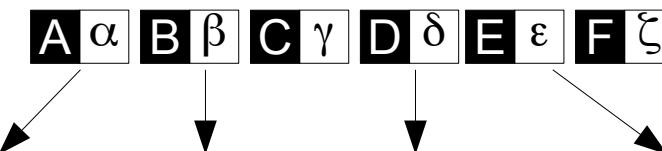


Reducir todo lo posible la cantidad de datos que pasan a la fase de reduce

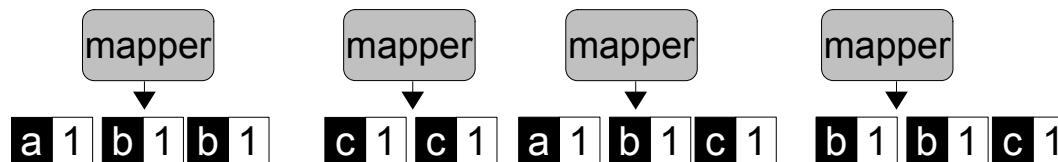
# Conteo de palabras : *combiners*

- Permiten computar resultados parciales a la salida de cada *mapper* (mini-reducers)

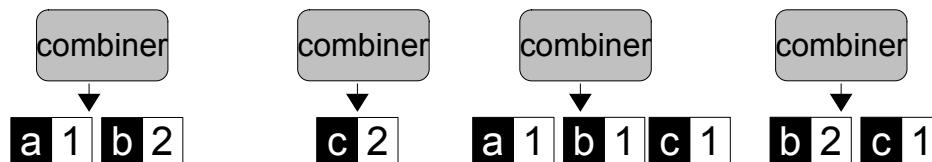
**Split**



**Map**

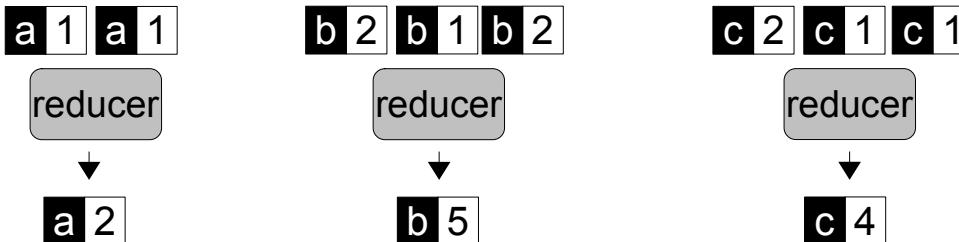


**Combine**



Shuffle & sort

**Reduce**



```

1: class MAPPER
2:   method MAP(docid a, doc d)
3:     for all term t ∈ doc d do
4:       EMIT(term t, count 1)

```

```

1: class REDUCER
2:   method REDUCE(term t, counts [c1, c2, ...])
3:     sum ← 0
4:     for all count c ∈ counts [c1, c2, ...] do
5:       sum ← sum + c
6:     EMIT(term t, count sum)

```

```

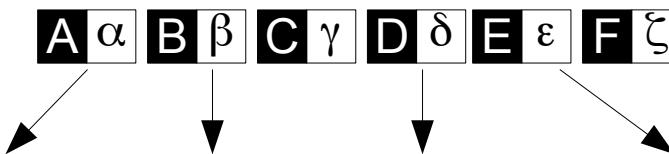
1: class REDUCER
2:   method REDUCE(term t, counts [c1, c2, ...])
3:     sum ← 0
4:     for all count c ∈ counts [c1, c2, ...] do
5:       sum ← sum + c
6:     EMIT(term t, count sum)

```

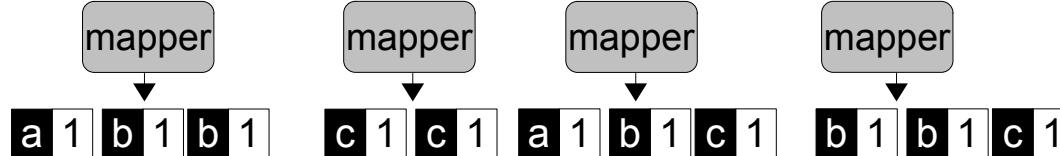
# Conteo de palabras : *partitioners*

- Permiten determinar cómo se distribuyen las parejas en los diferentes *reducers*

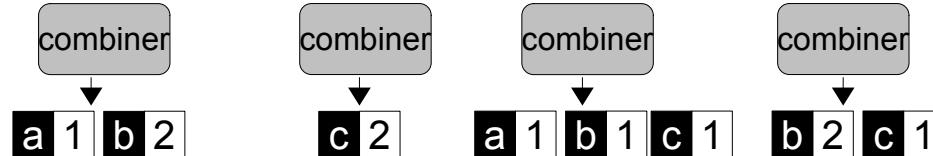
Split



Map



Combine

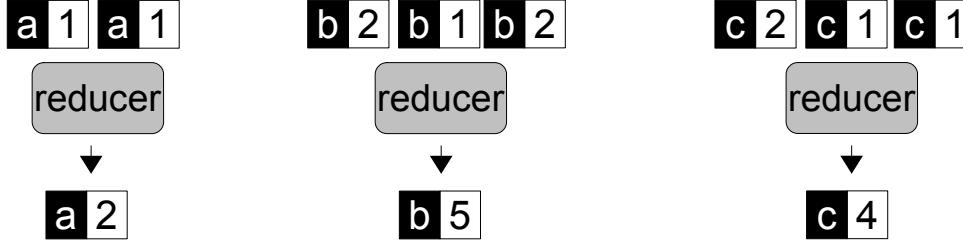


Partition



Shuffle & sort

Reduce



Un método de particionado sencillo consiste en computar  $\text{hash}(k) \bmod \#reducers$ .

# **Agenda**

- Big Data: algunas definiciones
- El paradigma de programación MapReduce
- **Patrones de diseño sobre MapReduce**
- Hadoop y Hadoop Distributed Filesystem

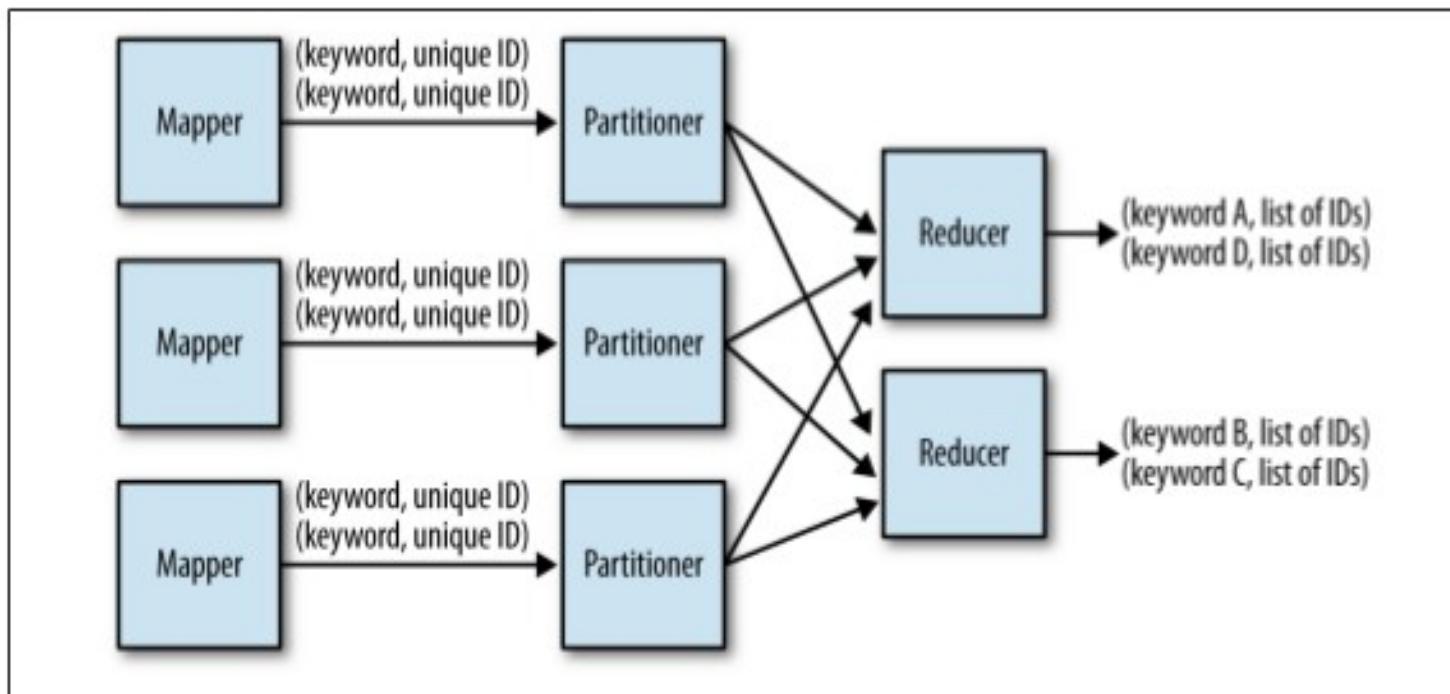
# Patrones en MapReduce

- No puedo resolver cualquier problema con esta técnica
- Para algunos problemas hay patrones de diseño definidos:
  - Sumarización e indexado
  - Filtrado
  - Organización de datos (particionado, transformaciones)
  - Join

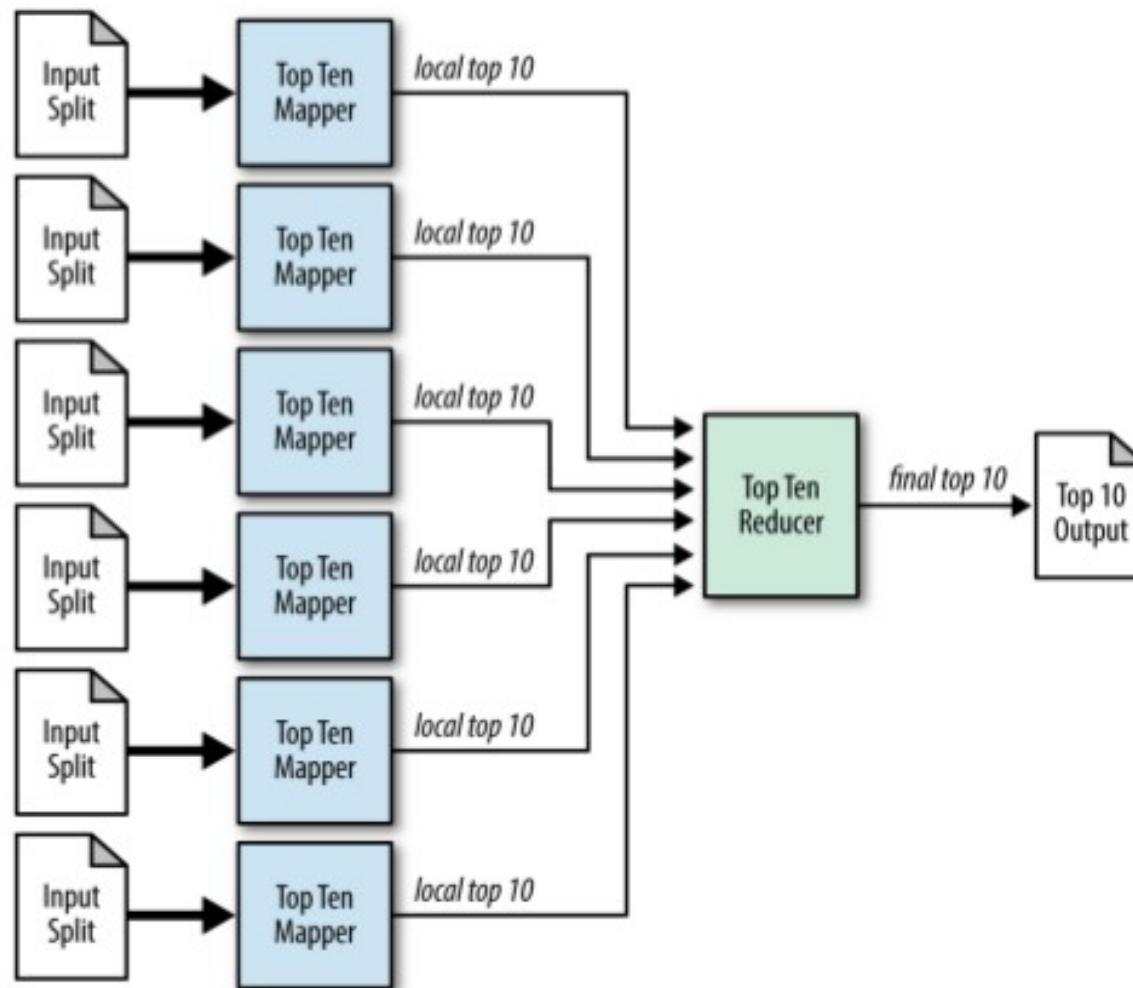
Donald Miner and Adam Shook. MapReduce Design Patterns Building Effective Algorithms and Analytics for Hadoop and Other Systems. O'Reilly Media, Inc., 2012.

# Indices inversos

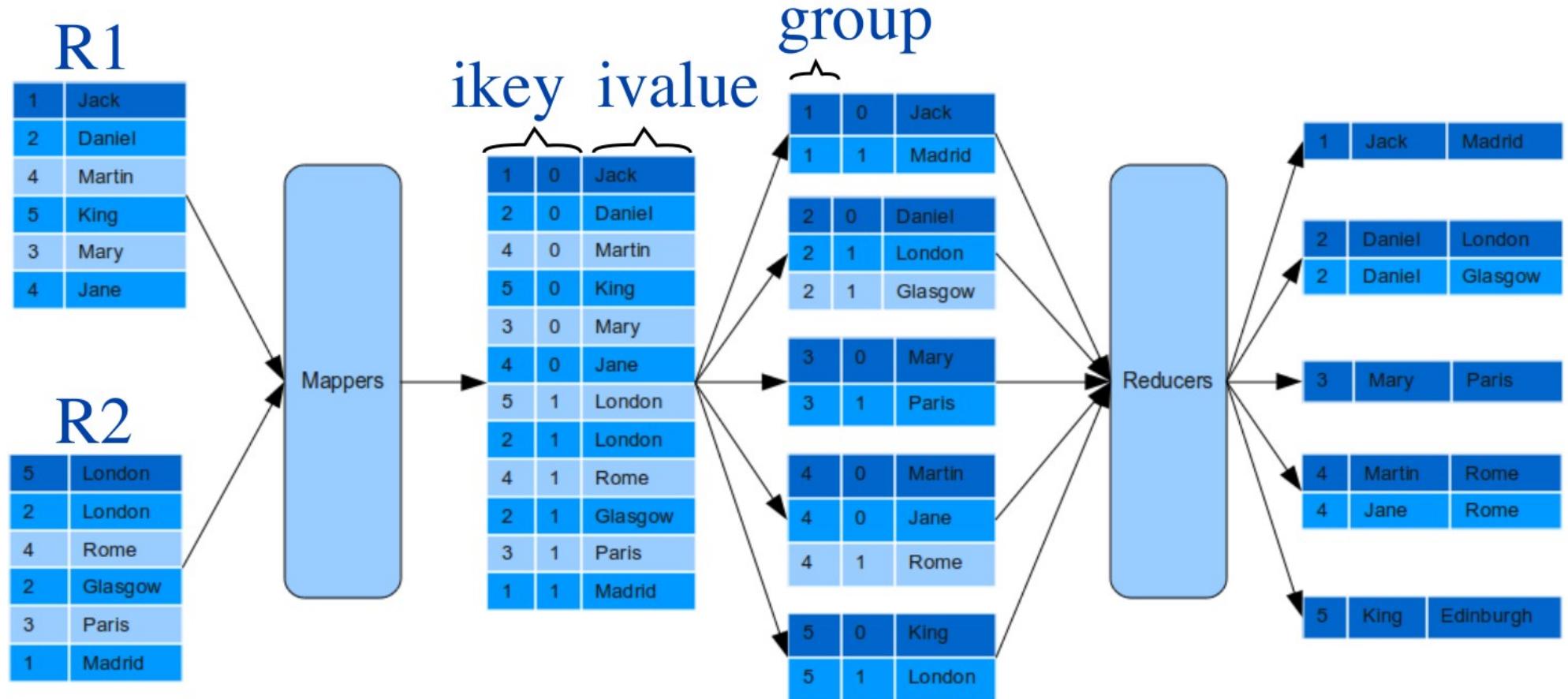
- Fue el caso de uso que dio origen a MapReduce en Google
- Construir una lista de URLs que contienen cierta palabra



# Filtrado: top - k



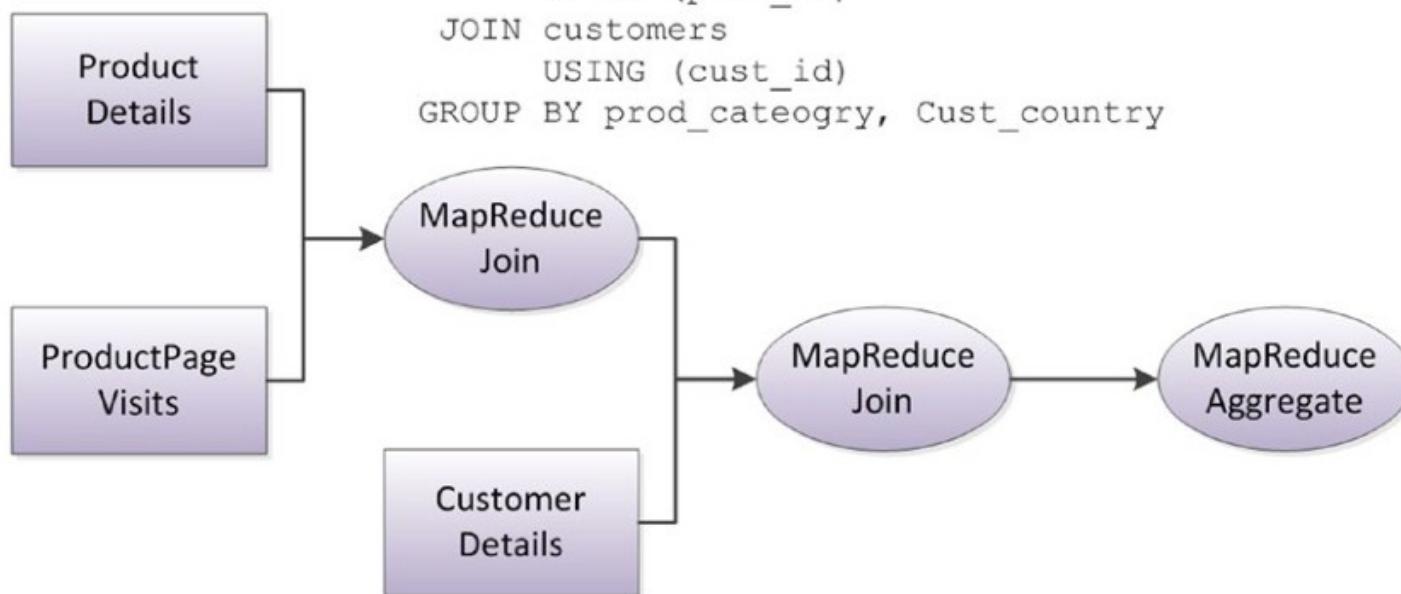
# Patrones de Join (reduce side join)



# Componiendo tareas

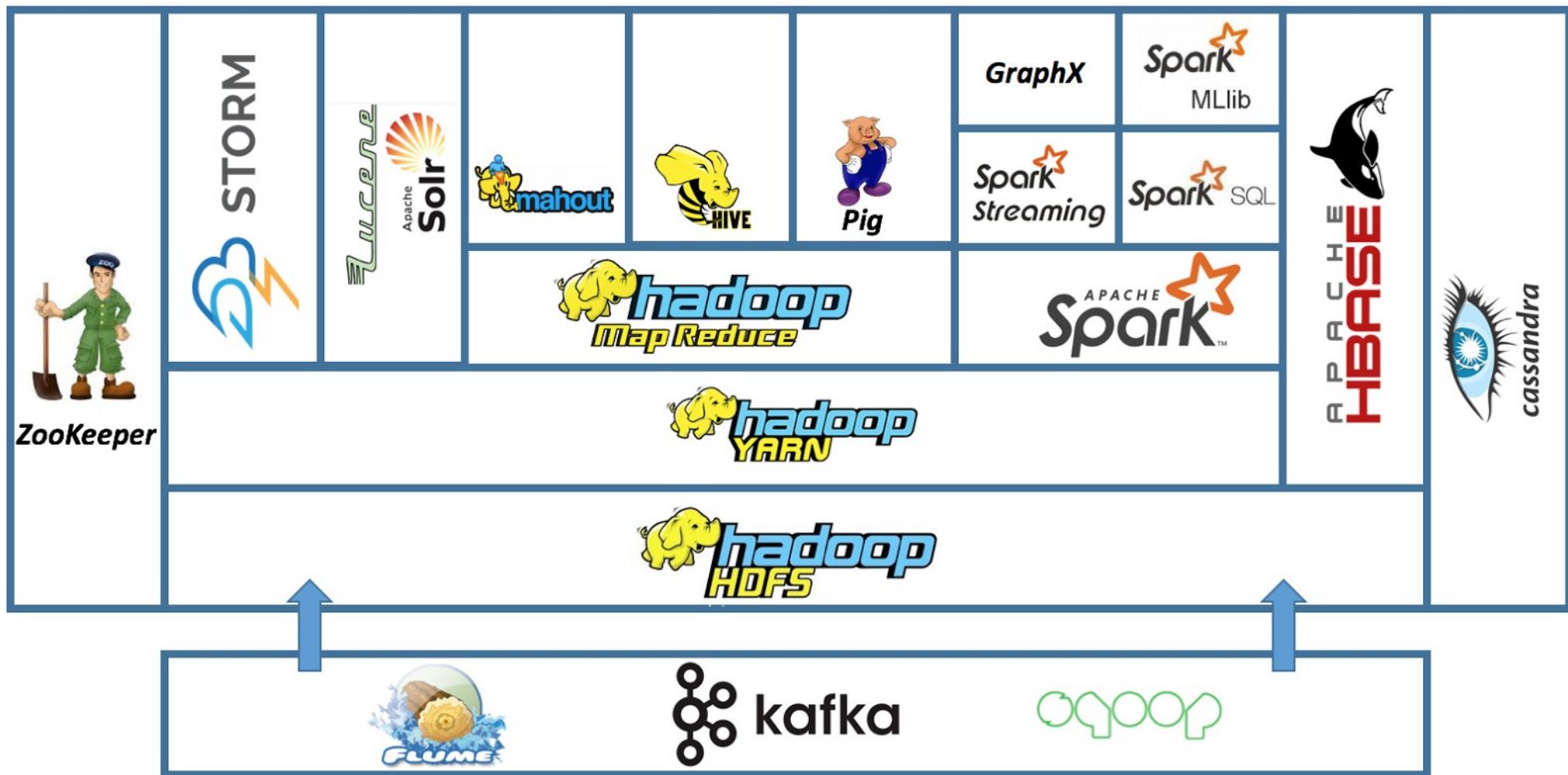
Equivalent of:

```
SELECT prod_category, Cust_country,  
       SUM(visits)  
  FROM products  
 JOIN product_page_visits  
    USING (prod_id)  
 JOIN customers  
    USING (cust_id)  
 GROUP BY prod_cateogry, Cust_country
```



# **Agenda**

- Big Data: algunas definiciones
- El paradigma de programación MapReduce
- Patrones de diseño sobre MapReduce
- **Hadoop y Hadoop Distributed Filesystem**



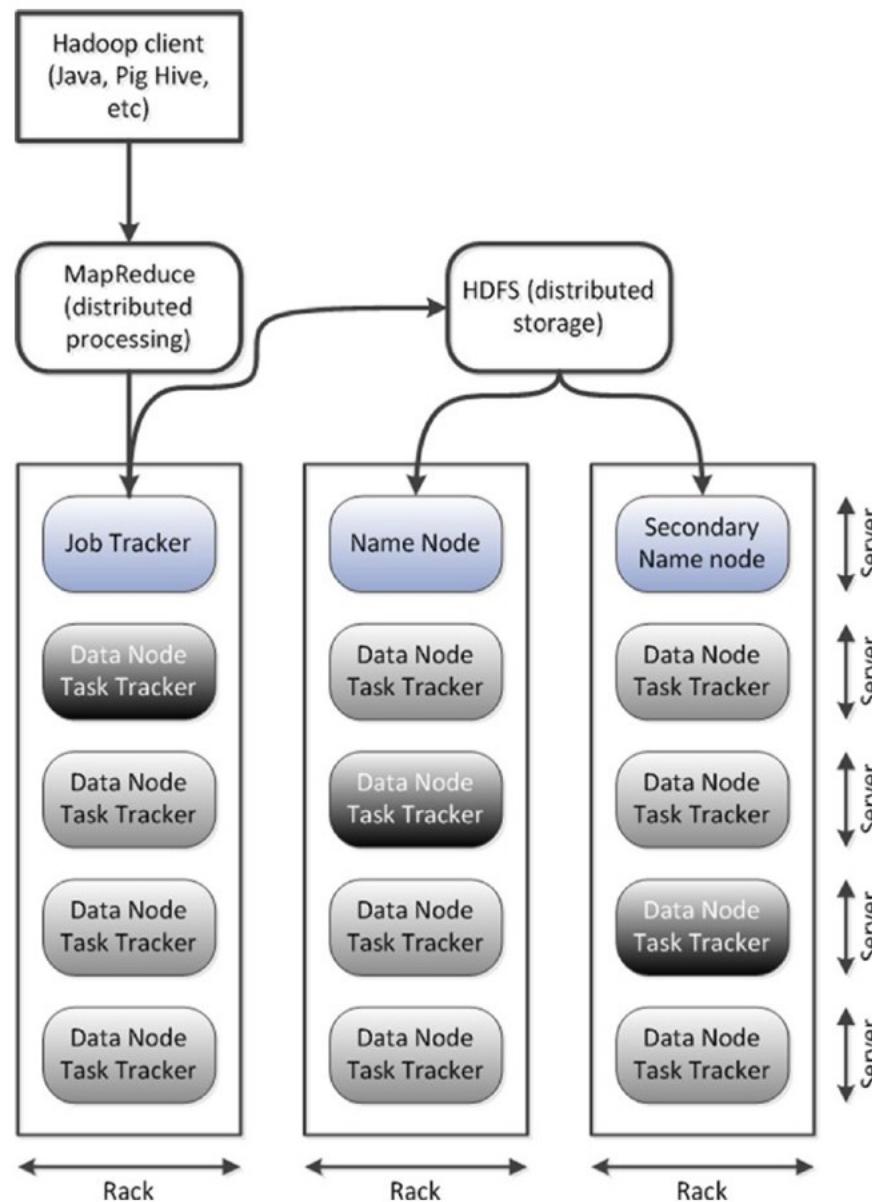
# Hadoop: algunas características

- Un entorno de ejecución distribuído.
- Pensado para ejecutar en *commodity hardware*.
- Altamente escalable.
- Redundancia de datos.
- *Schema on read* en lugar de *Schema on write*.

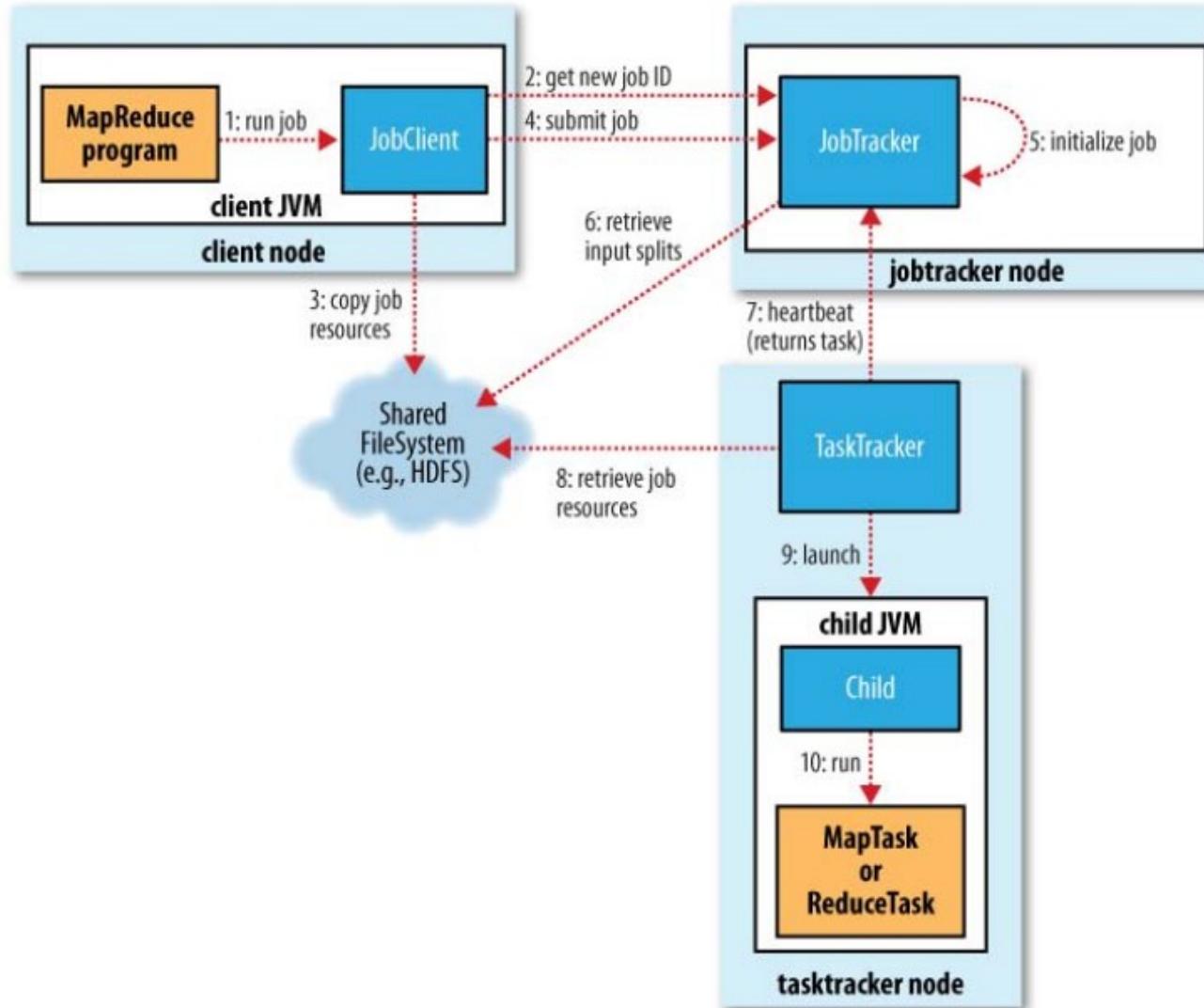
# **Hadoop como entorno resuelve:**

- *Scheduling* de tareas
- “Mueve” el código a los datos (comenzar la tarea en el nodo que tiene los datos necesarios)
- Sincronización entre procesos
- Errores y manejo de fallas

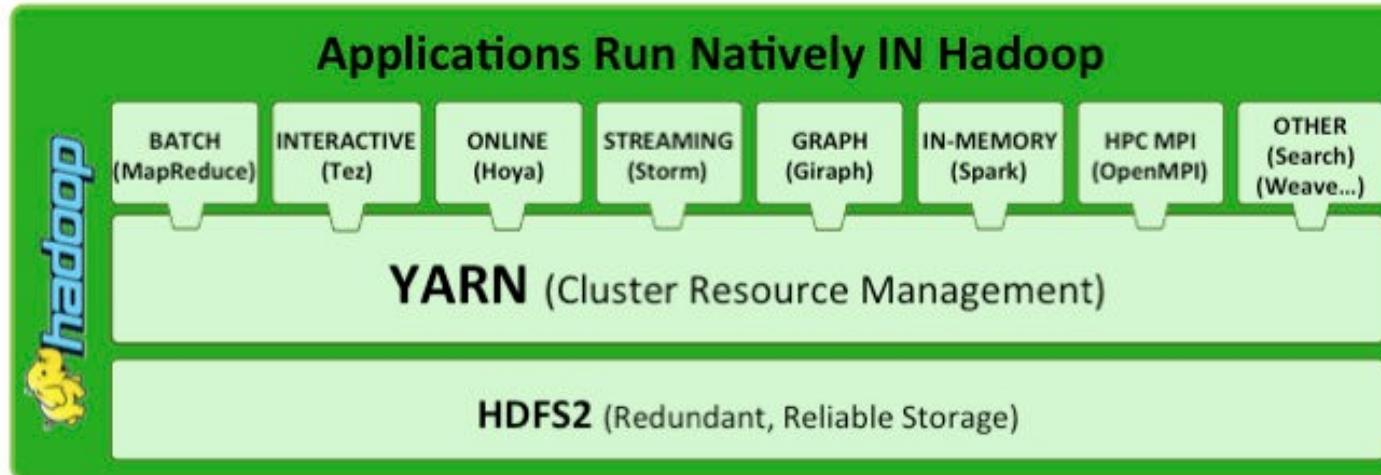
# Arquitectura: Hadoop 1.0



# Ejecutando un programa MapReduce en Hadoop 1.0

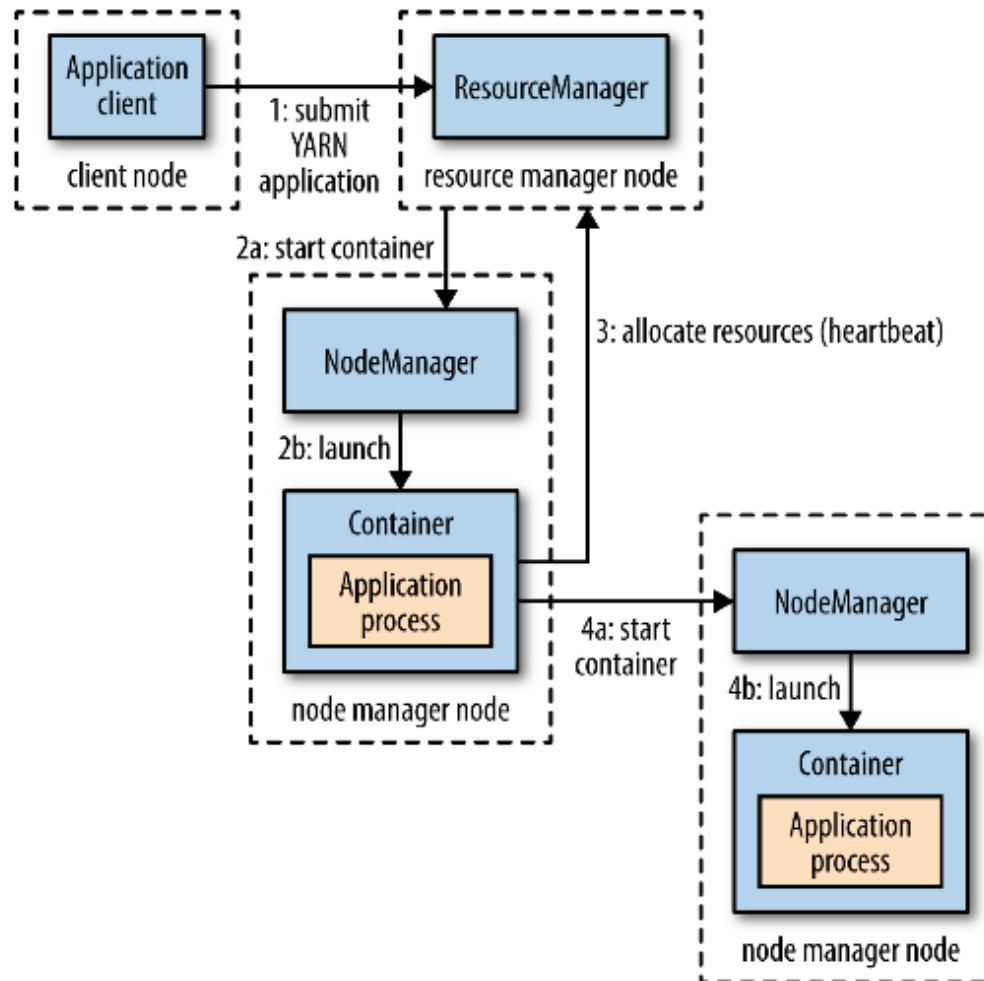


# Arquitectura: Hadoop 2.0



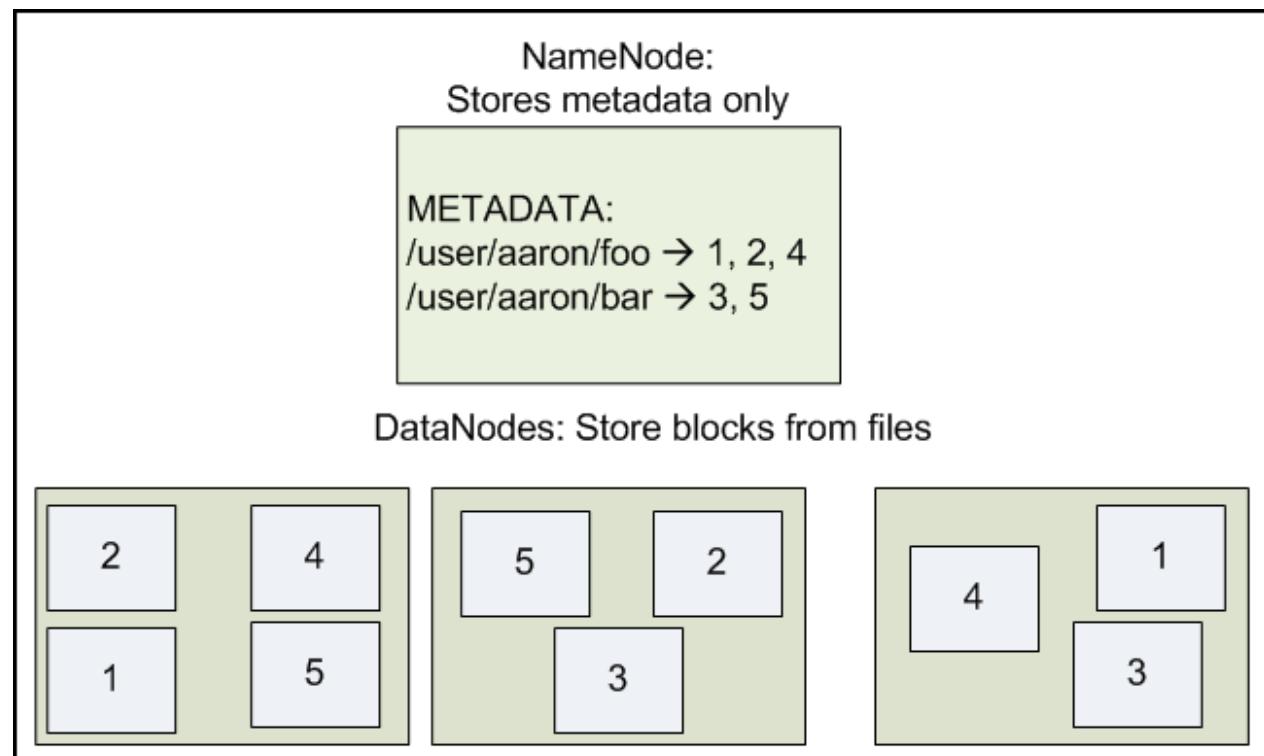
- Aparece YARN (Yet Another Resource Negotiator)
- El JobTracker y TaskTracker son reemplazados por 3 componentes:
  - **ResourceManager**: controla el acceso a todos los recursos
  - **NodeManager**: corre en cada nodo y maneja sus recursos. Responde al RM
  - **ApplicationManager**: controla la ejecución de la tarea.

# Ejecutando un programa en Hadoop 2.0



# Hadoop Distributed Filesystem (HDFS)

- Sistema de archivos distribuído
- Organiza los datos en *bloques* (64 MB)
- Dos tipos de nodos:
  - namenode
  - datanodes



# Pero programar sobre Hadoop no es sencillo :(

- Aparecen abstracciones
- Pig es un lenguaje de alto nivel que permite realizar consultas
  - B = ORDER A BY col4 DESC;
  - C = LIMIT B 10;
- Hive provee una abstracción sobre Hadoop.
  - Modelo de datos: tablas, vistas, etc
  - HiveQL como lenguaje de consultas