DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Caso Práctico: "Big Data"
Algoritmos paralelos con MapReduce

Caso práctico: "Big Data"

- Ciencia de datos
- Big Data
- MapReduce
- Hadoop
- Spark

Ciencia de datos

- **Análisis de datos [data analysis]**
  1962 John Tukey (Princeton & Bell Labs)
- **Ciencia de datos [data science]**
  1985 C.F. Jeff Wu (Georgia Institute of Technology)
- **KDD [Knowledge Discovery in Databases]**
  1989 Gregory Piatetsky-Shapiro (KDnuggets)
- **Minería de datos [data mining]**
  1990's @ Bases de datos

Ciencia de datos

Más denominaciones afines…

- **Inteligencia de negocio [Business Intelligence]**
  1958 Hans Peter Luhn (IBM)
  → 1989 Howard Dresner (Gartner)
- **Aprendizaje automático [ML: Machine Learning]**
  1959 Arthur Samuel (IBM)
  @ IA
- **Reconocimiento de formas [pattern recognition]**
  1967 k-NN (término prestado de psicología)
  @ Visión artificial [CV: Computer Vision]

Big Data

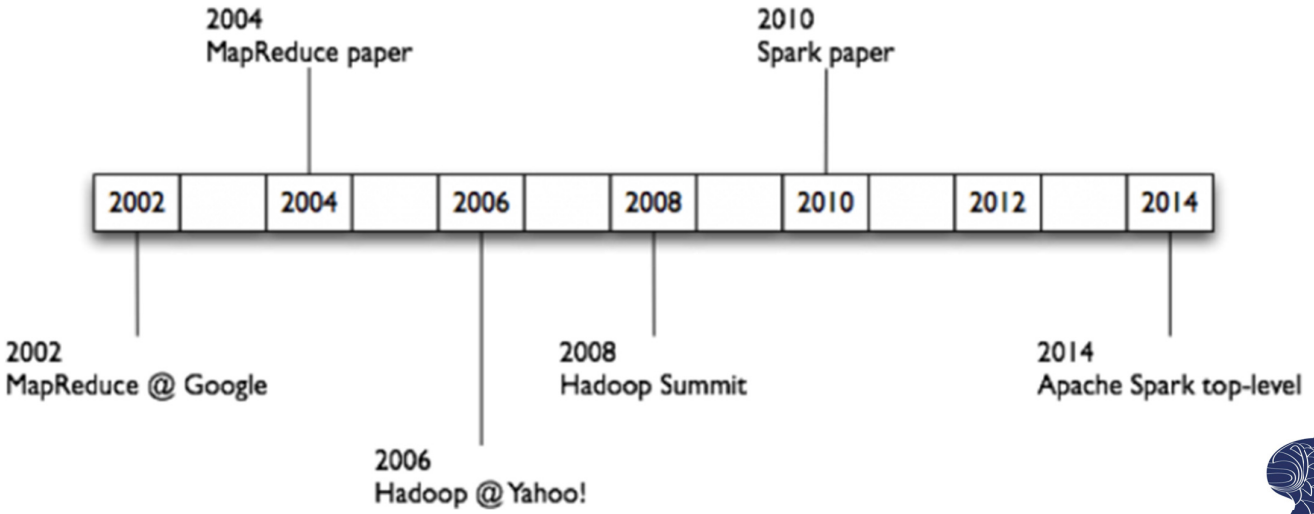
La importancia de la eficiencia de un algoritmo…

n	10	100	1000	10000	100000	1000000
O(n)	10ms	0.1s	1s	10s	100s	17 min
O($n \cdot \log_2 n$)	33ms	0.7s	10s	2 min	28 min	6 horas
O(n^2)	100ms	10s	17 min	28 horas	116 días	32 años
O(n^3)	1s	17min	12 días	31 años	32k años	32M años

Big Data

¿Cuándo pasa a ser "big data"?

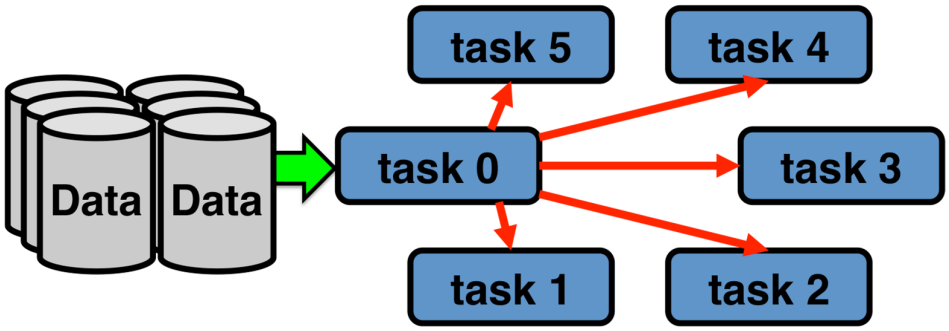
Cuando hacen falta técnicas paralelas para procesar los datos…



MapReduce

Aplicaciones

- "data-intensive": Grandes volúmenes de datos.
- "I/O-bound": más tiempo en acceder a los datos que en procesarlos (E/S domina sobre tiempo de CPU).

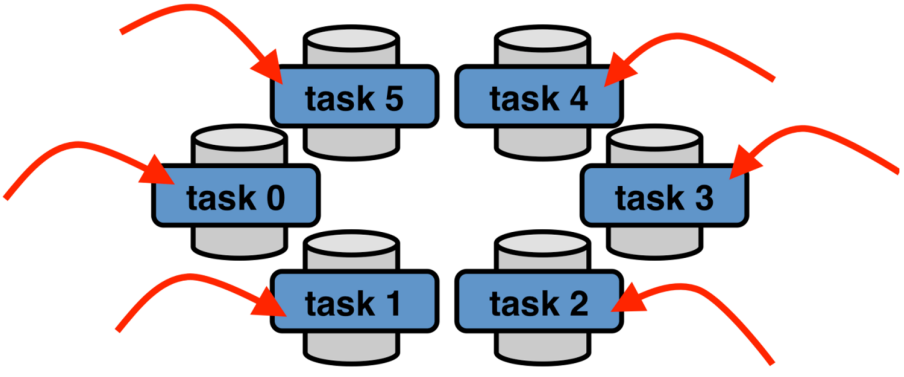


Solución tradicional (p.ej. MPI)
"bring the data to compute"

MapReduce

Aplicaciones

- "data-intensive": Grandes volúmenes de datos.
- "I/O-bound": más tiempo en acceder a los datos que en procesarlos (E/S domina sobre tiempo de CPU).



Solución MapReduce
"bring compute to the data"

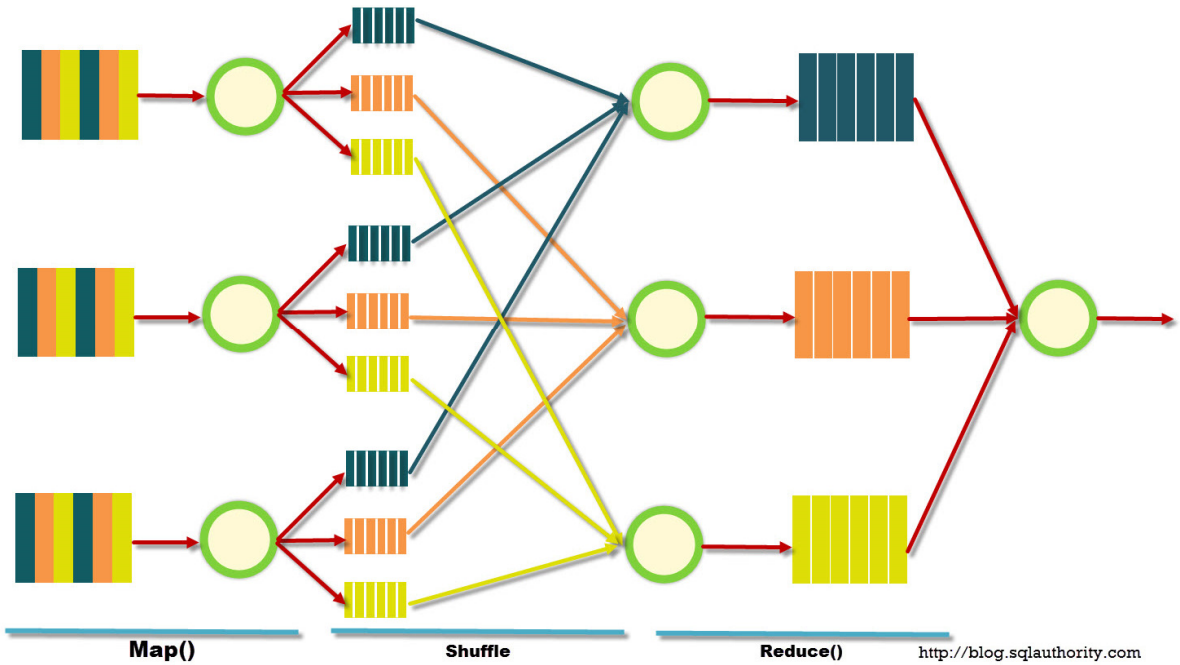
MapReduce

Modelo de programación cuya implementación permite procesar grandes cantidades de datos en un clúster utilizando algoritmos paralelos distribuidos.

Origen: Programación funcional
Funciones map y reduce de LISP (años 60): Vectorización

- map() permite realizar operaciones en paralelo sobre distintos fragmentos del conjunto de datos global.
- reduce() permite agregar/resumir los resultados obtenidos en paralelo.

MapReduce



MapReduce ≈ SELECT ... GROUP BY ...

MapReduce

Ejemplo

Contar la frecuencia de cada palabra en cada documento:

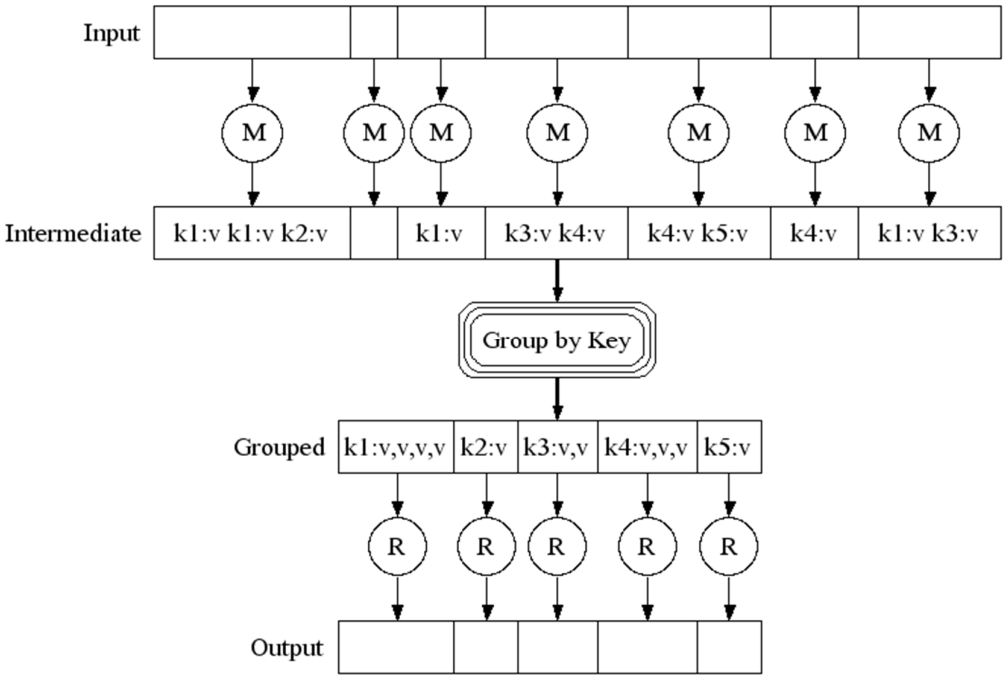
```
map(String input_key, String input_value):
    // input_key: document name
    // input_value: document contents
    for each word w in input_value:
        EmitIntermediate(w, "1");

reduce(String output_key, Iterator intermediate_values):
    // output_key: a word
    // output_values: a list of counts
    int result = 0;
    for each v in intermediate_values:
        result += ParseInt(v);
    Emit(AsString(result));
```

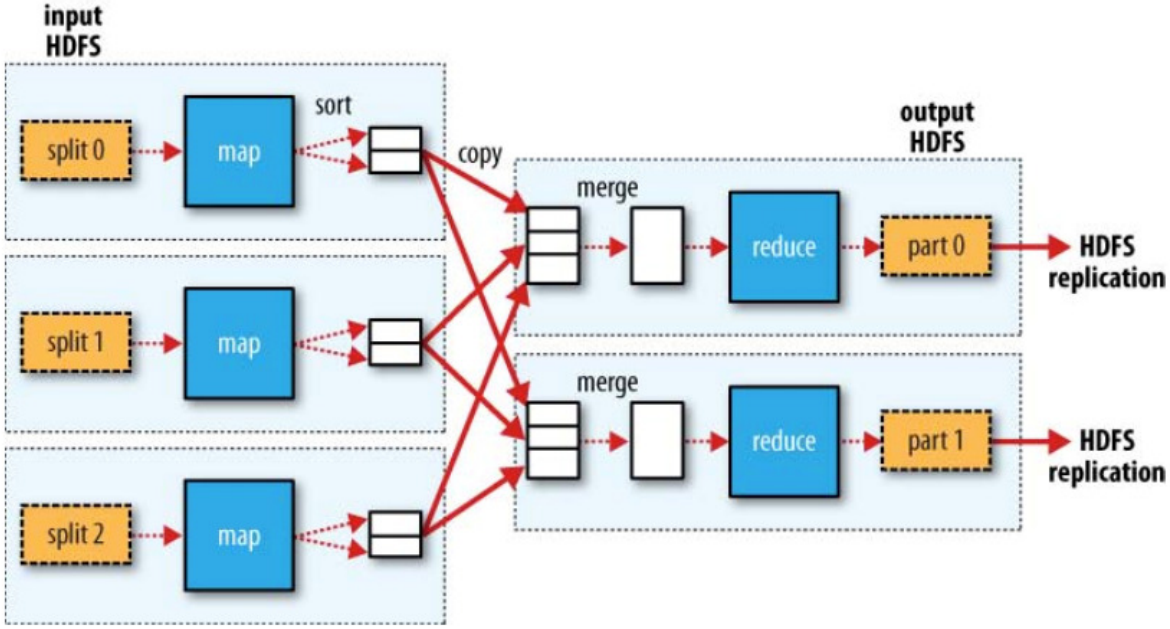

MapReduce

Ejemplo

Ejecución

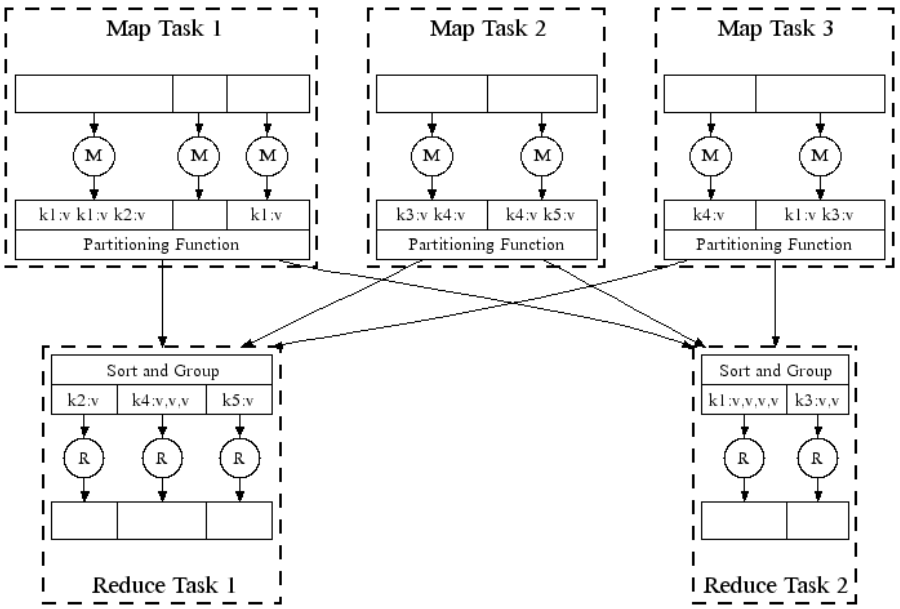


MapReduce

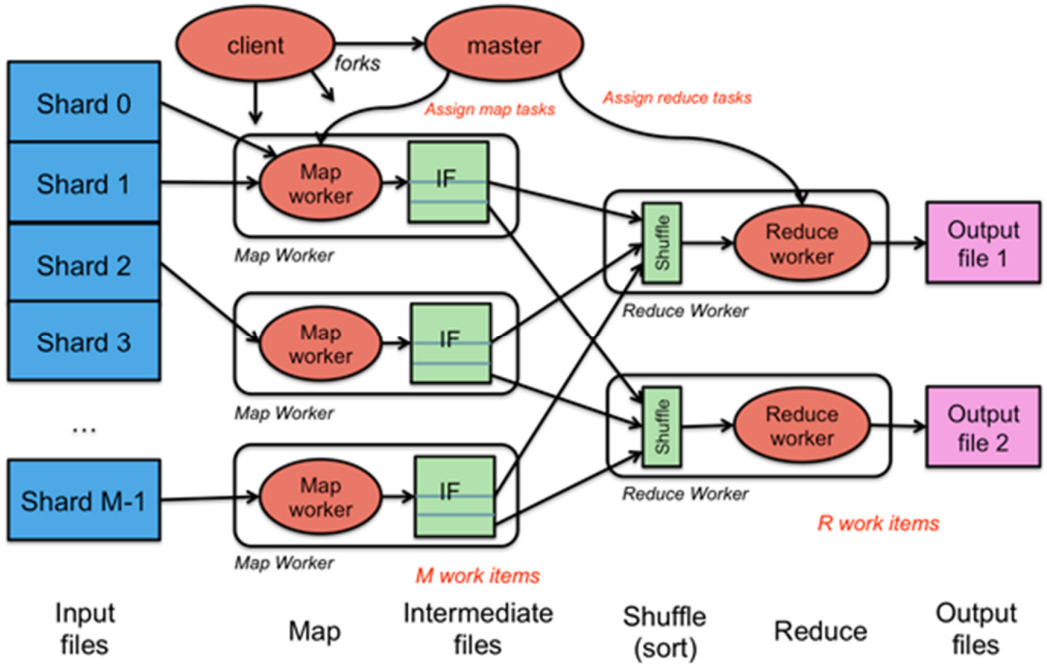


Ejemplo

Ejecución



MapReduce



MapReduce

Implementación

- Sin modelo de datos, sólo ficheros.
- Ofrece escalabilidad y tolerancia a fallos

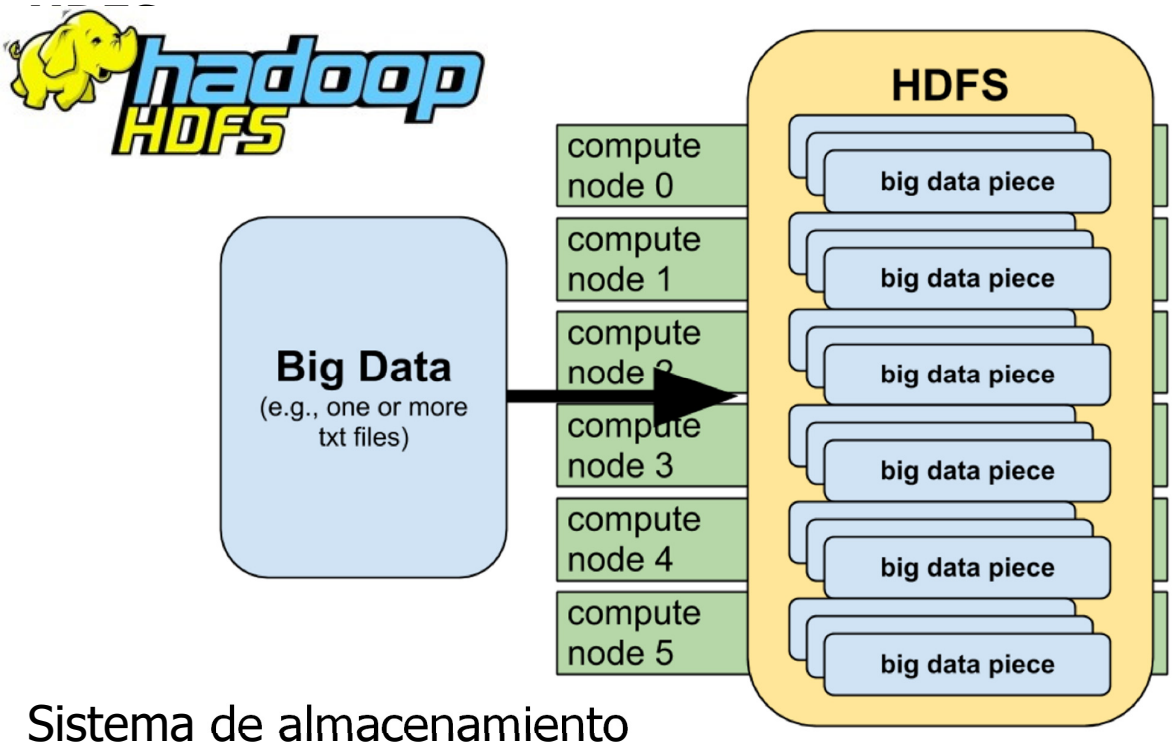
Implementación original: Google

- GFS [Google File System]

Implementación open-source: Hadoop (Yahoo!)

- HDFS [Hadoop Distributed File System]

MapReduce

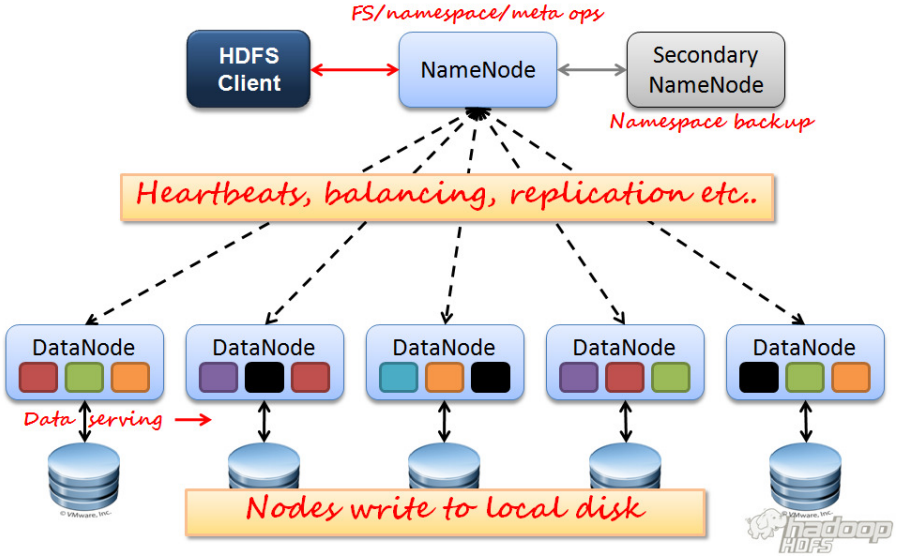


Sistema de almacenamiento distribuido, escalable y tolerante a fallos.

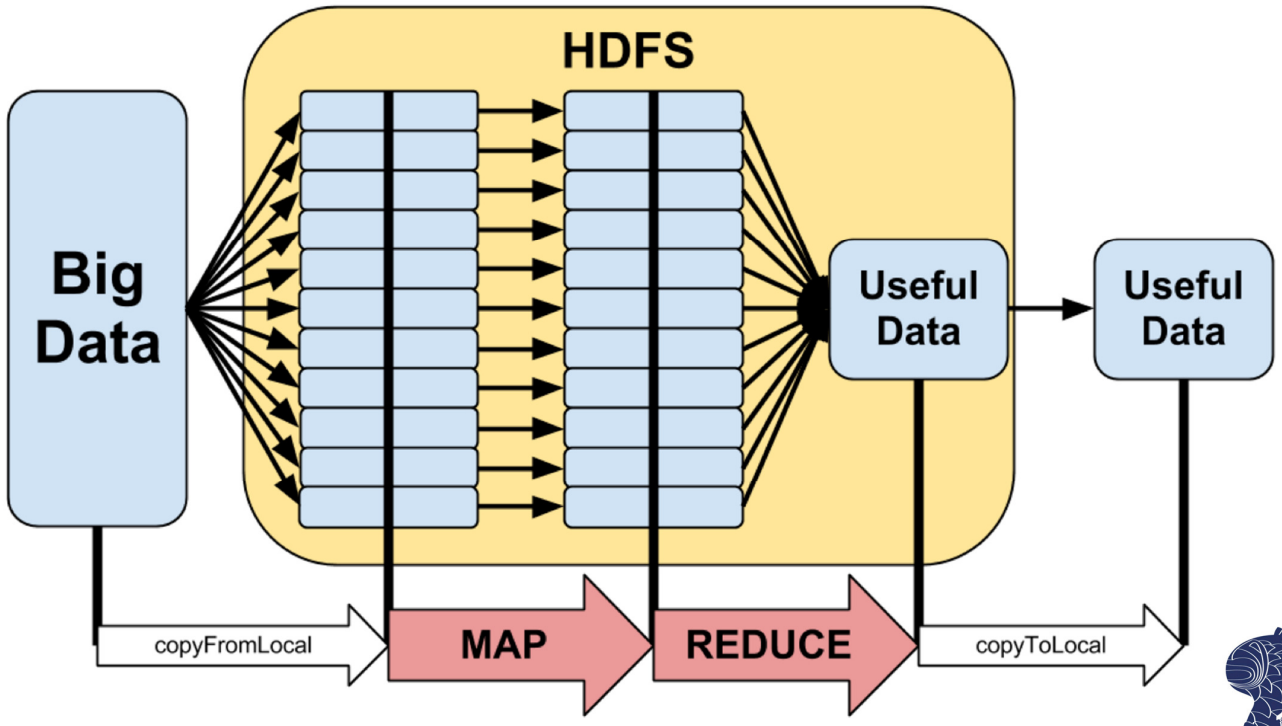
MapReduce

Tolerancia a fallos

- "heartbeat": ping periódico a cada tarea
- Replicación de datos:
  p.ej. 3 copias de cada fragmento de 64MB (Hadoop)

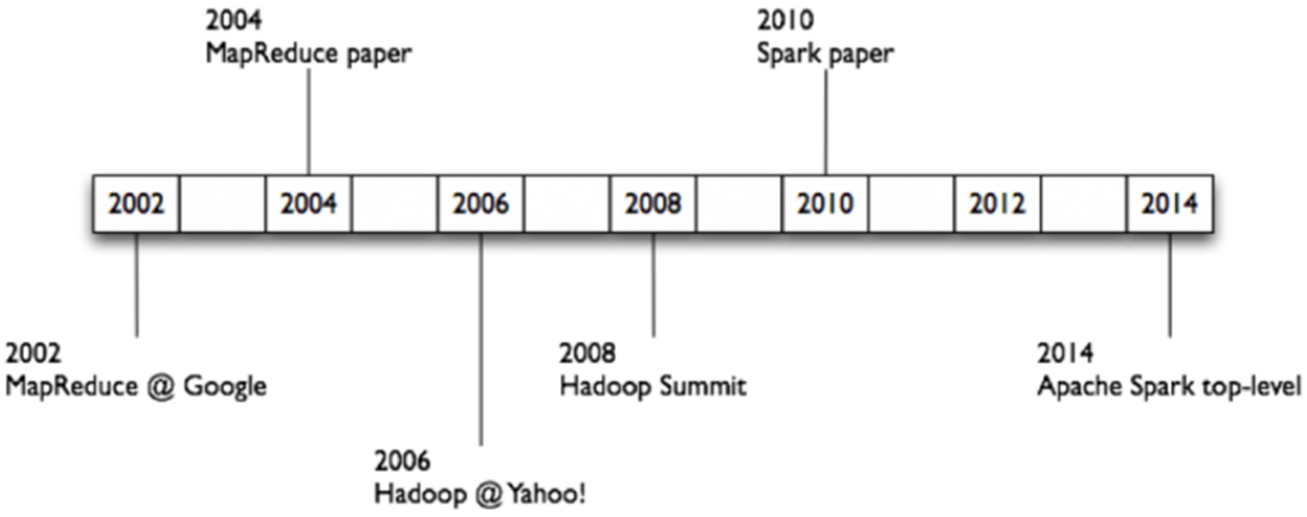


MapReduce



MapReduce

Evolución de las soluciones MapReduce



2004

MapReduce: Simplified Data Processing on Large Clusters

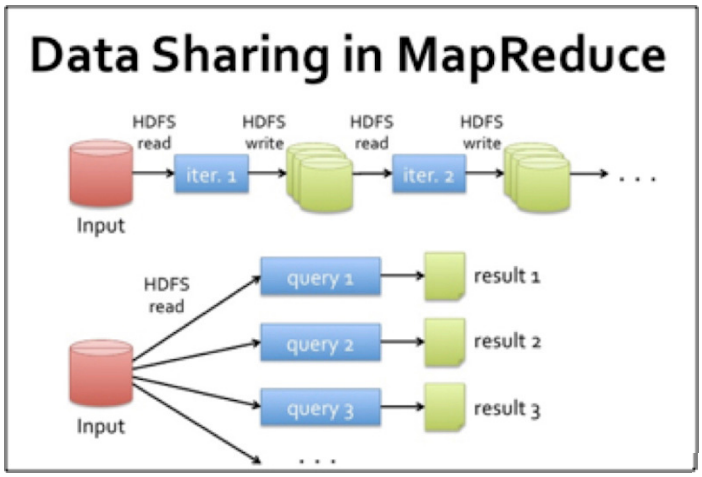
Jeffrey Dean and Sanjay Ghemawat

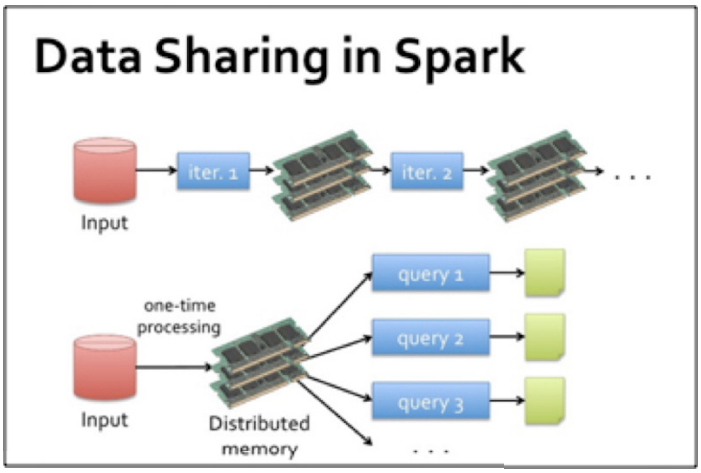
jeff@google.com, sanjay@google.com

Google, Inc.

MapReduce

Hadoop vs. Spark





hadoop MapReduce

Spark
Lightning-Fast Cluster Computing

MapReduce

Benchmark

Ordenar 100TB en disco

2013 Record: Hadoop
2100 machines
72 minutes

2014 Record: Spark
207 machines
23 minutes

Daytona Graysort Benchmark
http://sortbenchmark.org/
Récord actual: Tencent Sort, 134 segundos, 44.8 TB/min (2016)