Técnica Map-Reduce

Agradecimientos

Este curso es una versión libre del curso on-line Introducción a la Ciencia de Datos (Bill Howe, Univ. de Washington) https://www.coursera.org/course/datasci

Escalabilidad de Datos

Operar con un volumen arbitrariamente grande de datos.

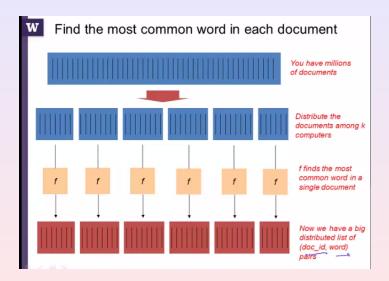
- ► En el pasado esta noción se restringía a operar con un volumen que superara el tamaño de la memoria (RAM)
 - La información se trae de disco y se procesa en porciones de tamaño más reducido
 - Las bases de datos convencionales resuelven este problema
- ► A partir del 2000 la alta disponibilidad de los servicios en Internet requiere otra solución
 - La latencia de los discos tiene tiempos de respuesta inaceptables
 - La información tiene que estar disponible en la memoria principal
 - Solución: clusters de muchas máquinas baratas trabajando cooperativamente en un mismo problema

Escalabilidad del Procesamiento de Datos

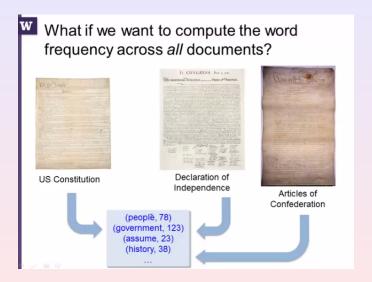
Otro aspecto importante es la complejidad de los algoritmos que procesan los datos

- ► En el pasado los algoritmos de complejidad polinomial *N*^m eran considerados aceptables
 - A medida que *m* crece, el impacto en un gran volumen de datos convierte al algoritmo en inaceptable
- ► En un cluster de k máquinas, la complejidad decrece N^m/k , esta solución es aceptable
- ► En la actualidad algunas aplicaciones requieren que su complejidad sea N * log(N)
 - ► Aplicaciones de procesamiento de streaming de datos
 - Es el tiempo mínimo para hacer una sóla pasada y guardar cada dato en un árbol

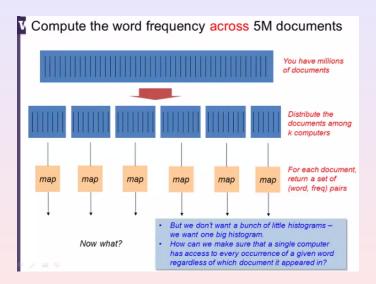
Procesamiento Distribuido en un Cluster



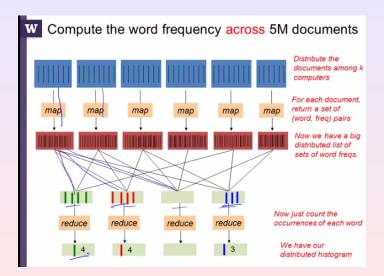
Histograma de Palabras

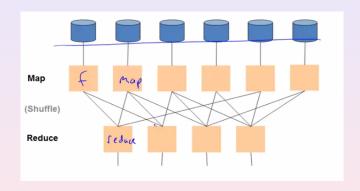


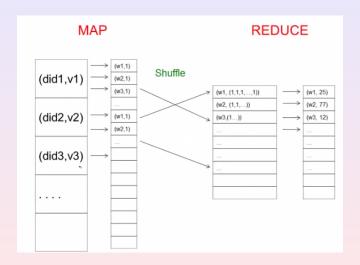
Fase de Map



Fase de Reduce







- Map-Reduce es un modelo de programación de alto nivel para procesamiento paralelo de grandes volúmenes de datos
- Hay que programar exclusivamente las fases de map() y de reduce()
- ► El modelo se ocupa de la distribución, la tolerancia a fallos, etc.
- Artículo de Google de 2004
- Implementación más popular es Hadoop (Apache)

Map-Reduce está compuesto por estas fases:

- 1. En cada nodo la función *mapper* se encarga de transformar los items en un conjunto de pares clave-valor
- 2. Se agrupan los pares clave-valor que tienen la misma clave
- 3. La función *reducer* procesa los valores asociados a cada clave y genera un output

```
map(String input key, String input value):
 // input_key: document name
 // input value: document contents
                                                ("history", 1)
 for each word w in input_value:
    EmitIntermediate(w, 1);
                        history (1,1,1,...)
reduce(String intermediate_key, Iterator intermediate_values):
 // intermediate key: word
 // intermediate values: ????
 int result = 0:
 for each v in intermediate_values:
    result += v;
 EmitFinal(intermediate key, result);
                   (history, 25)
                                                    slide source: Google, Inc.
```

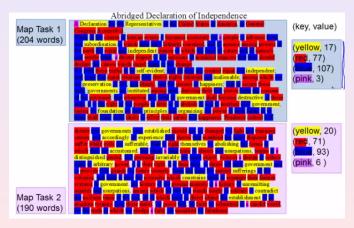
Documentos Grandes

Si eventualmente un documento fuese lo suficientemente grande como para no poder ser procesado por una sóla computadora en el cluster, el filesystem subyacente lo parte para que sea procesado por varias computadoras.

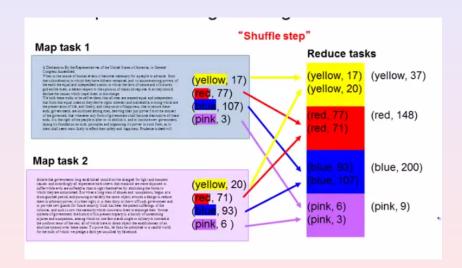


Ejemplo Palabras por Tamaño (Map)

En este ejemplo el objetivo es contar la cantidad de palabras cuya longitud es chica, mediana o grande.



Ejemplo Palabras por Tamaño (Reduce)



Cluster

- Muchas computadoras comunes en red procesando cooperativamente
- Paralelismo masivo: cientos o miles de computadoras
- Es muy alta la probabilidad de que haya fallas de hardware durante el procesamiento

File System Distribuido

- Concebido para procesamiento de grandes archivos (TBs, PBs)
- Cada archivo está particionado en bloques de 64MB
- ► Cada bloque está replicado por lo menos 3 veces en distintos racks (tolerancia a fallos)
- Implementaciones
 - ► Google GFS
 - Hadoop HDFS

- ► Es un framework liviano
- ► Garantiza el procesamiento paralelo
- ▶ Tolerante a fallos