MCDI

Maestría en Ciencia de Datos e Información

INFOTEC POSGRADOS

Paradigma map-reduce

Dra. Magali Arellano Vázquez

INFOTEC

Map-Reduce

Introducción a Map-Reduce I

MapReduce es un *framework* (modelo de programación) utilizado por Google para dar soporte a la computación paralela sobre grandes colecciones de datos en grupos de computadoras y al *commodity computing*.
El nombre del *framework* está inspirado en los nombres de dos importantes métodos, macros o funciones en programación funcional: **Map y Reduce**.

Map I

Básicamente Map, procesa en una función cada uno de los elementos que están dentro de un arreglo y forma un nuevo arreglo, por supuesto los trata de forma diferente, no es un `for` camuflado.

La función *map*()

Se encarga del mapeo y es aplicada en paralelo para cada ítem en la entrada de datos. Produce una lista de pares ($k2$, $v2$) por cada llamada.

Map II

Después de eso, el *framework* de MapReduce junta todos los pares con la misma clave de todas las listas y los agrupa, creando un grupo por cada una de las diferentes claves generadas.
Desde el punto de vista arquitectural el *master node* toma el *input*, lo divide en pequeñas piezas o problemas de menor identidad, y los distribuye a los denominados *worker nodes*.
Un *worker node* puede volver a sub-dividir, dando lugar a una estructura arbórea. El *worker node* procesa el problema y pasa la respuesta al nodo maestro.

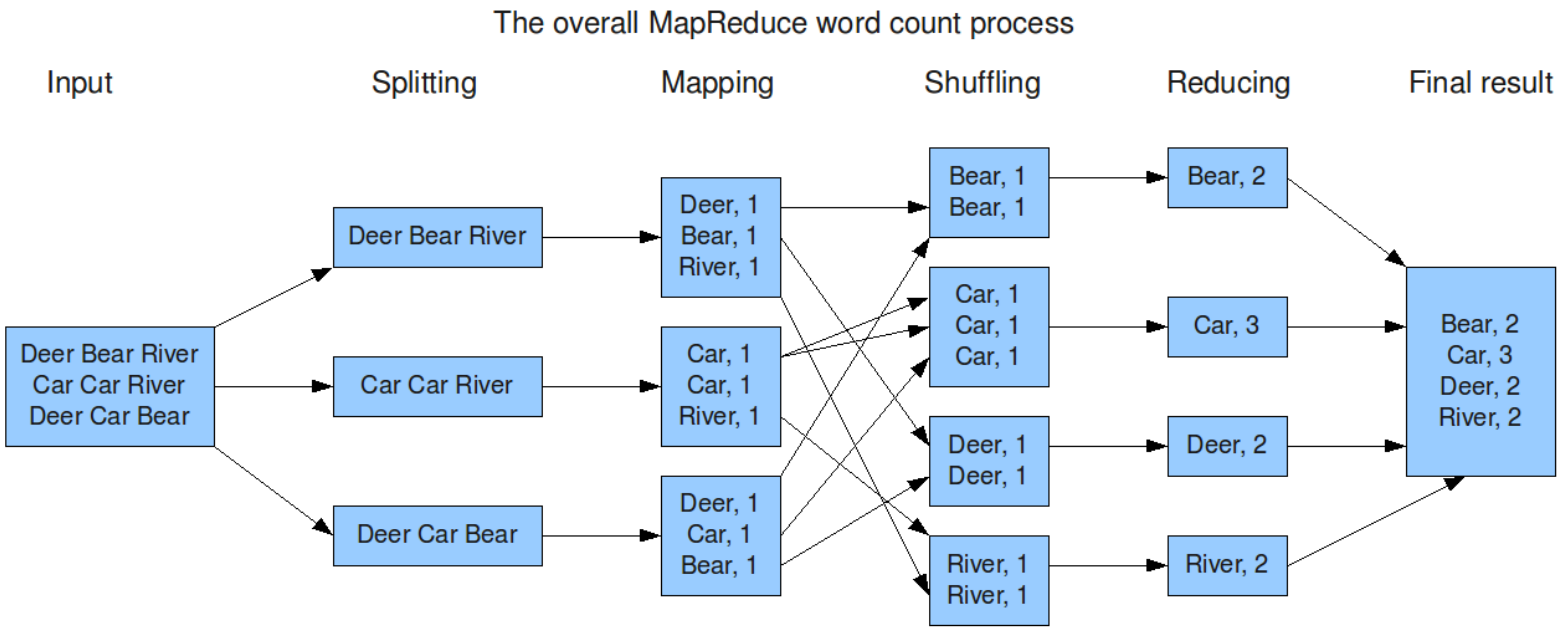
Reduce I

Reduce en contra parte de Map, lo que hace es recibir pares de datos, y los procesa dando como resultado un tercer dato, que es usado en la siguiente llamada.

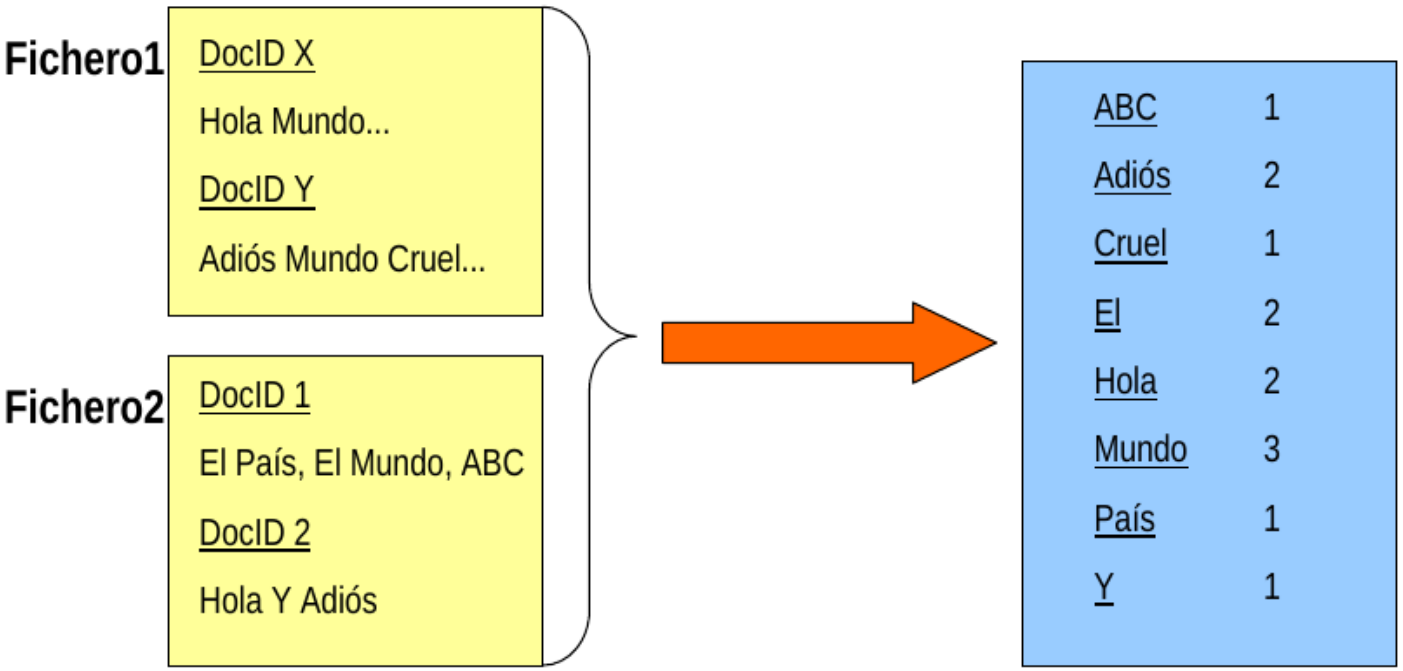
La función reduce()

Cada llamada a Reduce típicamente produce un valor v_3 o una llamada vacía, aunque una llamada puede retornar más de un valor. El retorno de todas esas llamadas se recoge como la lista de resultado deseado.

Esquema


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

Esquema


Fichero1
DocID X
Hola Mundo...
DocID Y
Adiós Mundo Cruel...
Fichero2
DocID 1
El País, El Mundo, ABC
DocID 2
Hola Y Adiós
ABC 1
Adiós 2
Cruel 1
El 2
Hola 2
Mundo 3
País 1
Y 1

Ejemplos

Ejemplo Map

```
1  map(String name, String document):
2      //  clave: nombre del documento
3      //  valor: contenido del documento
4      for each word w in document:
5        EmitIntermediate(w, 1);
```

La función *map*() divide un documento en palabra mediante el empleo de un analizador léxico, y emite una serie de tuplas de la forma (*clave*, *valor*) donde la clave es la palabra y el valor es “1”. Es decir, por ejemplo, del documento “El perro en el pasto” la función *map* retornaría: (“El”, “1”), (“perro”, “1”), (“en”, “1”), (“el”, “1”), (“pasto”,“1”).

Ejemplo Reduce I

```
1  reduce(String word, Iterator partialCounts):
2      // word: una palabra
3      // partialCounts: un [[ Iterador (patron de diseno)| lista parcial ]]
           para realizar cuentas agregadas
4      int result = 0;
5      for each v in partialCounts:
6          result += ParseInt(v);
7      Emit(result);
```

Cada documento es dividido en palabras, y cada palabra se cuenta con valor inicial “1” por la función Map, utilizando la palabra como el resultado clave.
El *framework* reúne todos los pares con la misma clave y se alimenta a la misma llamada Reduce, por lo tanto, esta función sólo necesita la suma de todos los valores de su entrada para encontrar el total de

Ejemplo Reduce II

las apariciones de esa palabra. En el ejemplo anterior (“el”, “1”) aparece dos veces debido a que la clave “el” tiene dos ocurrencias, el resto de claves sólo aparece una vez.

Ejemplo

Ejemplo: Amigos en Facebook

Amigos en común

Cada usuario de Facebook tiene una lista de amigos (tenga en cuenta que los amigos son una relación bidireccional en Facebook. Si soy tu amiga, tu eres mi amigo).
Facebook tiene un montón de espacio en el disco y atienden a cientos de millones de peticiones diarias. Ellos han decidido pre-calcular cálculos que pueden reducir el tiempo de tramitación de las solicitudes. Una petición común es la petición es “Tú y Paco tienen 35 amigos en común”.
Cuando alguien visita tu perfil, podrá ver una lista de los amigos que tienen en común. Esta lista no cambia con frecuencia por lo que sería un desperdicio recalcular cada vez que el usuario visita el perfil.
Vamos a usar *map-reduce* para que podamos calcular todos los amigos en común una vez al día y almacenar los resultados. Más tarde, es sólo una búsqueda rápida. Tenemos un montón de discos, somos Facebook.

Estrategia

Asume que los amigos se almacenan como *Persona* $\rightarrow$ [*ListadeAmigos*], nuestra lista de amigos, entonces es:

$A \rightarrow BCD$
$B \rightarrow ACDE$
$C \rightarrow ABDE$
$D \rightarrow ABCE$
$E \rightarrow BCD$

Map

Cada línea será un argumento para un mapper. Por cada amigo en la lista de amigos, la salida será de un par clave-valor. La clave será la de un amigo junto con la persona. El valor será el de la lista de amigos.

Map

Para map $(A \rightarrow BCD)$:

$$(AB) \rightarrow BCD$$
$$(AC) \rightarrow BCD$$
$$(AD) \rightarrow BCD$$

Para map $(B \rightarrow ACDE)$:

$$(AB) \rightarrow ACDE$$
$$(BC) \rightarrow ACDE$$
$$(BD) \rightarrow ACDE$$
$$(BE) \rightarrow ACDE$$

Para map $(C \rightarrow ABDE)$:

$$(AC) \rightarrow ABDE$$
$$(BC) \rightarrow ABDE$$
$$(CD) \rightarrow ABDE$$
$$(CE) \rightarrow ABDE$$

Map

Para map $(D \rightarrow ABCE)$:

$$(AD) \rightarrow ABCE$$
$$(BD) \rightarrow ABCE$$
$$(CD) \rightarrow ABCE$$
$$(DE) \rightarrow ABCE$$

Para map $(E \rightarrow BCD)$:

$$(BE) \rightarrow BCD$$
$$(CE) \rightarrow BCD$$
$$(DE) \rightarrow BCD$$

Preprocesamiento

Antes de enviar los pares a los reducer, se agrupan por llaves y se obtiene:

$$(AB) \rightarrow (ACDE)(BCD)$$
$$(AC) \rightarrow (ABDE)(BCD)$$
$$(AD) \rightarrow (ABCE)(BCD)$$
$$(BC) \rightarrow (ABDE)(ACDE)$$
$$(BD) \rightarrow (ABCE)(ACDE)$$
$$(BE) \rightarrow (ACDE)(BCD)$$
$$(CD) \rightarrow (ABCE)(ABDE)$$
$$(CE) \rightarrow (ABDE)(BCD)$$
$$(DE) \rightarrow (ABCE)(BCD)$$

Reducer

Cada linea se pasará como argumento al reducer. La función reducer intersectará las listas de los valores y la salida de la misma llave con el resultado de la intersección.

$$(AB) \rightarrow (CD)$$
$$(AC) \rightarrow (BD)$$
$$(AD) \rightarrow (BC)$$
$$(BC) \rightarrow (ADE)$$
$$(BD) \rightarrow (ACE)$$
$$(BE) \rightarrow (CD)$$
$$(CD) \rightarrow (ABE)$$
$$(CE) \rightarrow (BD)$$
$$(DE) \rightarrow (BC)$$

Cuando D visita el perfil de B, rápidamente buscamos (BD) y se tienen 3 amigos en común (ACE).