

|—) (—| |— (—| | | (—| |

—) | — (—) (—) | — (—) | — | — | — | — | — ) (—)

$$(\underline{\quad}| \quad | \underline{\quad}) \quad (\underline{\quad}|$$

| \vee | (\exists| \exists| \Big| \neg) (\exists| \exists| \exists| \exists| \exists| \exists| \exists| \exists|

## Contenido

- 1. Introducción.
  - 2. Qué es MapReduce?
  - 3. Procesando un Documento.
  - 4. Procesamiento de Varios Documentos.
  - 5. Análisis de Complejidad.
  - 6. Resumen.
  - 7. Ejercicios.

## — 1. Introducción.

Para obtener de forma adecuada un mecanismo de procesamiento paralelo, no es suficiente unir un gran número de máquinas o de procesadores. Se debe establecer el mecanismo para dividir el programa que se desea hacer entre las máquinas.

En los años 90s, muchas de las compañías tenían estos problemas. Tenían "data centers", tenían una gran cantidad de máquinas, pero no contaban con una manera adecuada para dividir los programas entre las máquinas.

Por supuesto, tenían problemas interesantes por resolver, búsquedas en web, construcción de índices, búsquedas textuales, etc.

Uno de los primeros modelos que se construyeron para distribuir programas en una gran cantidad de máquinas fue el algoritmo de MapReduce.

## — 2. Qué es MapReduce?

MapReduce permite el procesamiento distribuido. Los Maps se pueden realizar de forma paralela, si cada una de estas operaciones es independiente de las otras, en la práctica, las operaciones de Map pueden utilizar una cantidad infinita de procesadores. De igual manera los reductores, Reduce, toman las salidas de los Maps para producir los resultados deseados.

Para lograr el proceso completo, se utilizan dos etapas:

- 1) Map.  
Se procesa cada elemento de una lista.
  - 2) Reduce.  
Una lista de números se reduce, usualmente a un único número.

### **3. Procesando un Documento.**

Supongan que tenemos el siguiente documento representado como una lista:

doc01:  
[yo,quiero,un,gato,gato,no,perro,gato];

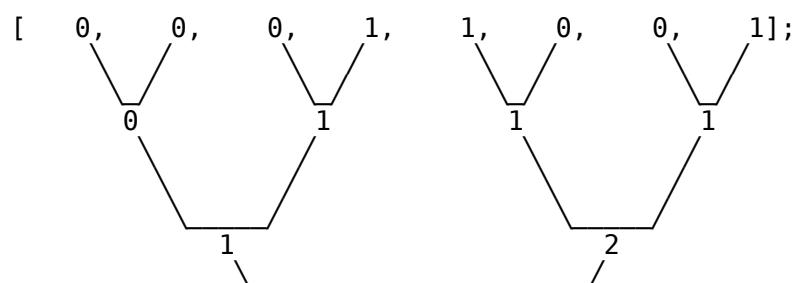
- ### ● El proceso de Map:

De forma paralela, se puede realizar el conteo de la palabra gato:

```
[ yo,quiero, un, gato, gato, no, perro, gato];
   |       |     |       |       |       |
[ 0,      0,    0,     1,     1,     0,     0,     1];
```

Ahora aplicamos el proceso de Reduce.

- ### ●● El proceso de Reduce.



En total se tienen 3 gatos.

En realidad se tiene 3 veces la palabra gato. :)

#### 4. Procesamiento de Varios Documentos.

Supongamos que tenemos 4 documentos. En cada uno de estos documentos deseamos contar el número de veces que aparece las palabras "gato" y "perro".

Documento1:

[hoy,compré,un,perro,y,un,gato,...]

Supongamos que las palabras aparecen:

gato: 5 veces.

perro: 3 veces.

Documento2:

Supongamos que las palabras aparecen:

gato: 5 veces.

perro: 0 veces.

Documento3:

Supongamos que las palabras aparecen:

gato: 14 veces.

perro: 3 veces.

Documento4:

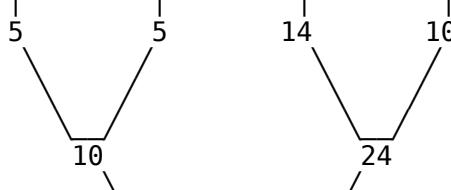
Supongamos que las palabras aparecen:

gato: 10 veces.

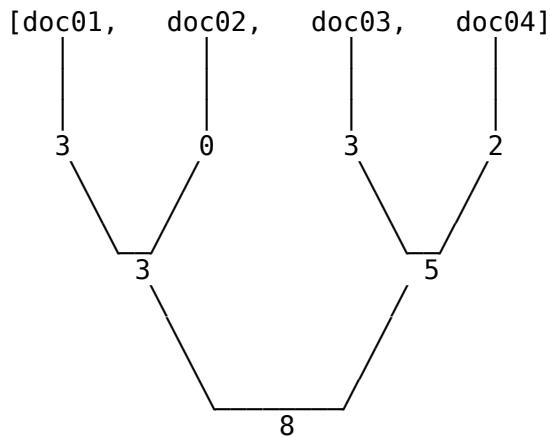
perro: 2 veces.

Se puede realizar el siguiente proceso en paralelo, para contar la palabra gato.  
La primera parte es el "map" y las sumas sucesivas son el "reduce".

[doc01, doc02, doc03, doc04]



Si se desean contar los perros:



## 5. Análisis de Complejidad.

### •• MapReduce de Un Documento.

Si se utilizan múltiples procesadores, la función de map toma tiempo  $O(1)$ .

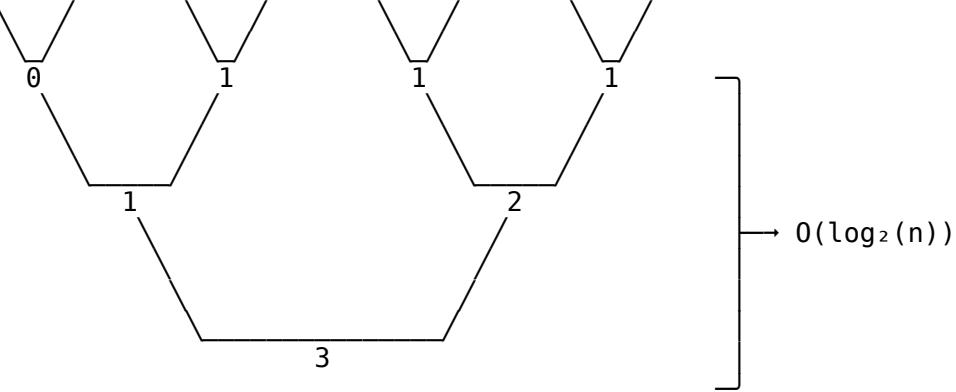
```

[ yo,quiero,      un,   gato,   gato,   no,   perro,   gato];
[ 0,      0,      0,      1,      1,      0,      0,      1];
  0(1)  0(1)  0(1)  0(1)  0(1)  0(1)  0(1)  0(1)
  
```

Después usamos el proceso de Reduce.

```

  0(1)  0(1)  0(1)  0(1)  0(1)  0(1)  0(1)  0(1)
  [ 0,      0,      0,      1,      1,      0,      0,      1];
  —————— 0(1)
  
```

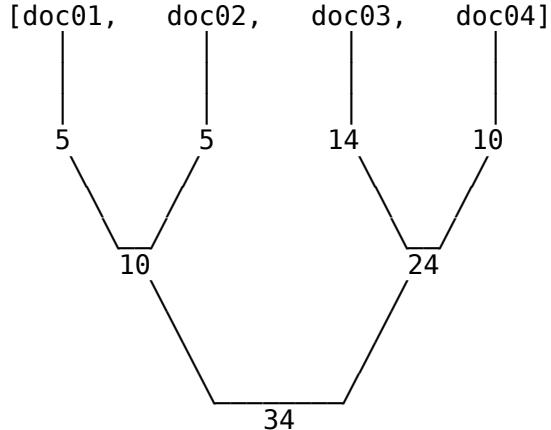


Por lo tanto el tiempo total está dado por:

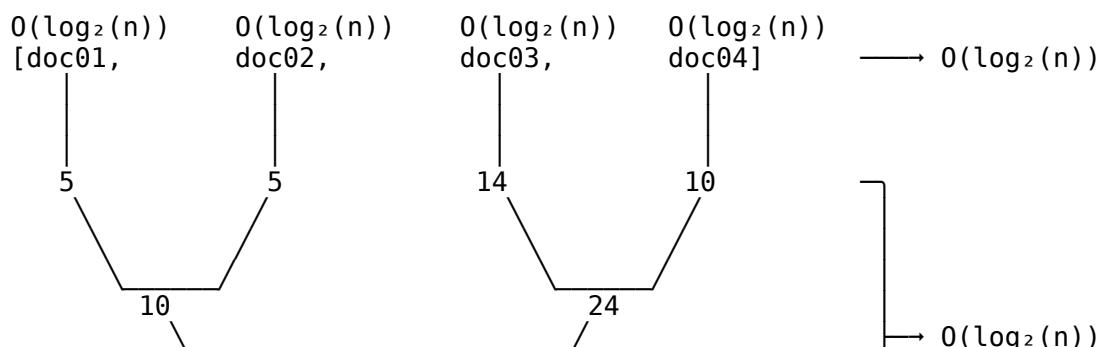
$$O(1) + O(\log_2(n)) = O(\log_2(n))$$

### ●● MapReduce de Varios Documentos.

Tenemos el siguiente proceso:



Cuyo tiempo de complejidad es el siguiente:



Por lo tanto el tiempo total estará dado por:

$$O(\log_2(n)) + O(\log_2(n)) = 2*O(\log_2(n))$$

Además del incremento de velocidad, el valor de "n" que se utiliza en cada paso es muy pequeño.

De igual manera, se puede utilizar un procesamiento paralelo para contar cualquier otra palabra.

## ----- 6. Resumen.

- Los algoritmos paralelos pueden realizar múltiples operaciones al mismo tiempo.
- Uno de los modelos más conocidos es el proceso de mapReduce.

## ----- 7. Ejercicios.

### ●● Ejercicio 1.

Construya el código para hacer este proceso de mapReduce.

En Wxmaxima tendrá que usar las operaciones de "maplist" y "apply".  
Puede utilizar el "help" para encontrar como trabajan estas funciones.