UTE EC

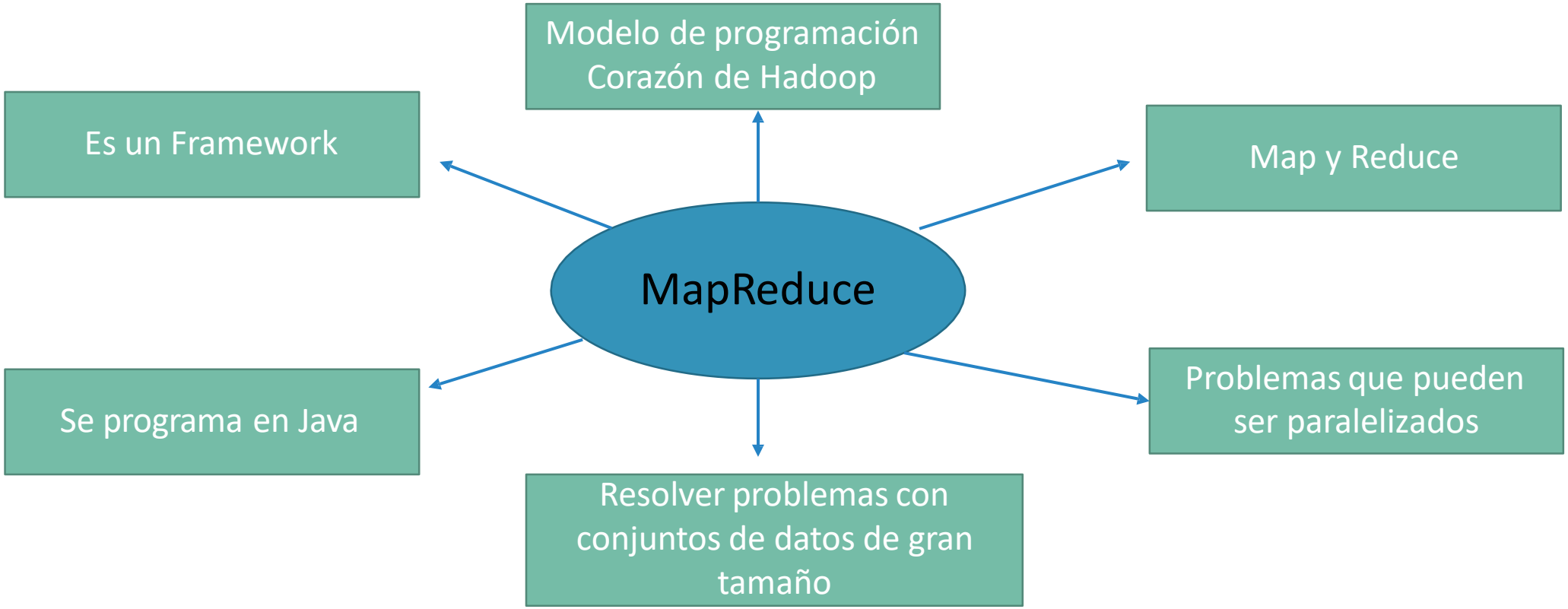
EDUCACIÓN EN LÍNEA

Computación Paralela y Distribuida

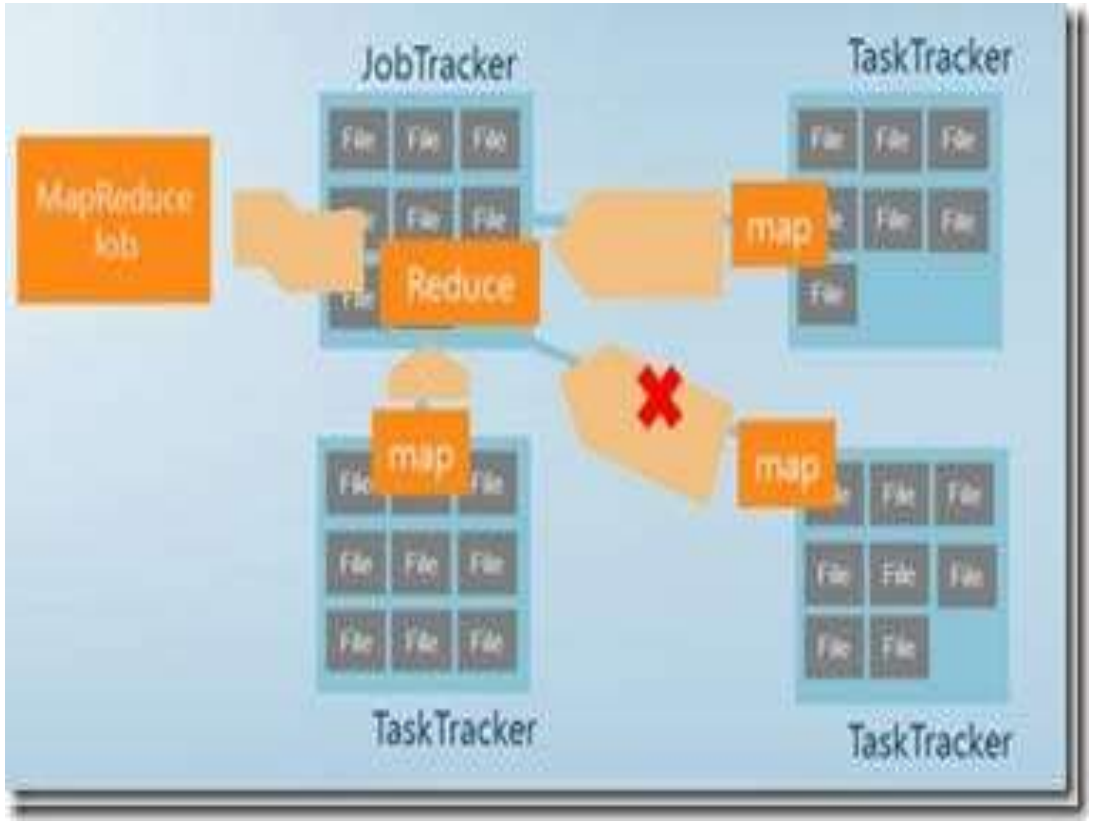
MapReduce

Integrantes: Dayana Quinde
Estefania Ugsha

¿Qué es MapReduce?



En la Figura vemos los componentes principales del Framework, donde se aprecia que las funciones de map son ejecutadas por los TaskTrackers mientras que las funciones de Reduce es ejecutada por el JobTracker.



- El Framework MapReduce tiene una arquitectura maestro / esclavo.
- Cuenta con un servidor maestro o JobTracker y varios servidores esclavos o TaskTrackers, uno por cada nodo del clúster.
- El JobTracker es el punto de interacción entre los usuarios y el framework MapReduce.
- Los usuarios envían trabajos MapReduce al JobTracker, que los pone en una cola de trabajos pendientes y los ejecuta en el orden de llegada.
- El JobTracker gestiona la asignación de tareas y delega las tareas a los TaskTrackers.
- Los TaskTrackers ejecutan tareas bajo la orden del JobTracker y también manejan el movimiento de datos entre la fase Map y Reduce.

Para ver las diferencias entre JobTracker y TaskTracker vamos a ver las características de cada uno

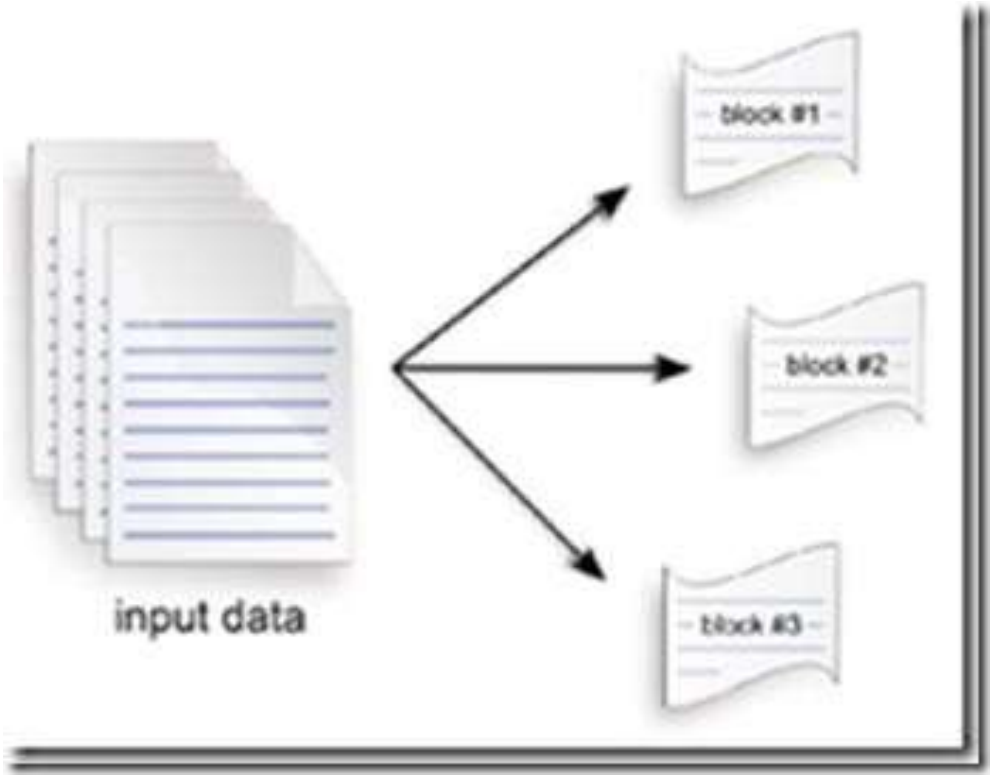
- **JobTracker**: Servidor maestro de MapReduce
- Capacidad para manejar metadatos de trabajos
  - ❑ Estado de la petición del trabajo
  - ❑ Estado de las tareas que se ejecutan en TaskTracker
- Decide sobre la programación
- Hay exactamente un JobTracker por cluster.
- Recibe peticiones de tareas enviadas por el cliente.
- Programa y monitoriza los trabajos MapReduce con TaskTrackers.

- **TaskTracker**: Servidor esclavo de MapReduce
- Ejecuta las solicitudes de trabajo de JobTrackers
- Obtiene el código que se ejecutará
- Aplica la configuración especifica del trabajo
- Comunicación con el JobTracker:
  - ❑ Envíos de la salida, finalizar tareas, actualización de tareas, etc.

Map

- La función Map recibe como parámetros un par de (clave, valor) y devuelve una lista de pares.
- Esta función se encarga del mapeo y se aplica a cada elemento de la entrada de datos, por lo que se obtendrá una lista de pares por cada llamada a la función Map
- Después se agrupan todos los pares con la misma clave de todas las listas, creando un grupo por cada una de las diferentes claves generadas.
- No hay requisito de que el tipo de datos para la entrada coincida con la salida y no es necesario que las claves de salida sean únicas.

Map ($clave_1$, $valor_1$) –> lista ($clave_2$, $valor_2$)



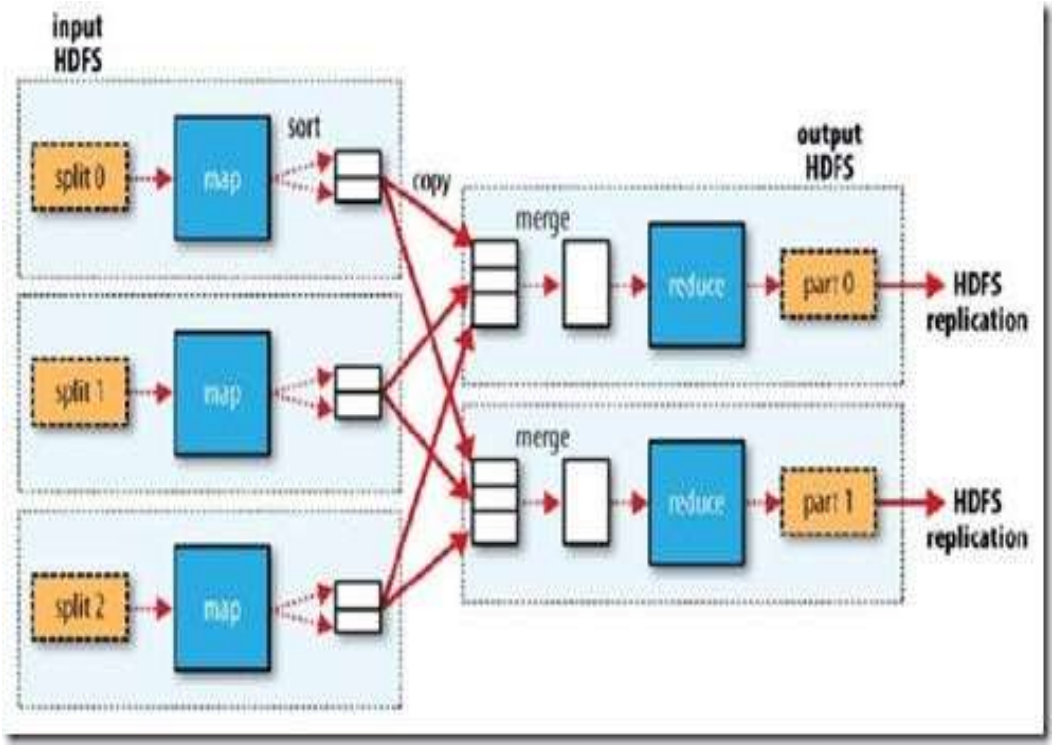
La operación de Map se paraleliza, el conjunto de archivos de entrada se divide en varias tareas llamado FileSplit, ver como se muestra la figura el tamaño típico de bloque es de 128MB. Las tareas se distribuyen a los nodos TaskTrackers, y estos se pueden realizar la misma tarea si hiciera falta.

Reduce

- La función Reduce se aplica en paralelo para cada grupo creado por la función Map().
- La función Reduce se llama una vez para cada clave única de la salida de la función Map. Junto con esta clave, se pasa una lista de todos los valores asociados con la clave para que pueda realizar alguna fusión para producir un conjunto más pequeño de los valores.

  Reduce ($clave_2$, lista($valor_2$)) –> lista($valor_2$)

- Cuando se inicia la tarea reduce, la entrada se encuentra dispersa en varios archivos a través de los nodos en las tareas de Map.
- Los datos obtenidos de la fase Map se ordenan para que los pares clave-valor sean contiguos ,esto hace que la operación Reduce se simplifique ya que el archivo se lee secuencialmente.
- Si se ejecuta el modo distribuido estos necesitan ser primero copiados al filesystem local en la fase de copia.
- Al final, la salida consistirá en un archivo de salida por tarea reduce ejecutada.



Por lo tanto, N archivos de entrada generará M mapas de tareas para ser ejecutados y cada mapa de tareas generará tantos archivos de salida como tareas Reduce hayan configuradas en el sistema.

Ejemplo:

Este ejemplo de MapReduce es un proceso para contar las apariciones de cada palabra en un conjunto de documentos:

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

La función map() en este caso divide un documento en palabras (es decir lo tokeniza) mediante el empleo de un simple 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 "La casa de la pradera" la función map retornaría: ("la", "1"), ("casa", "1"), ("de", "1"), ("la", "1"), ("pradera", "1").

UTE EC

¡GRACIAS!