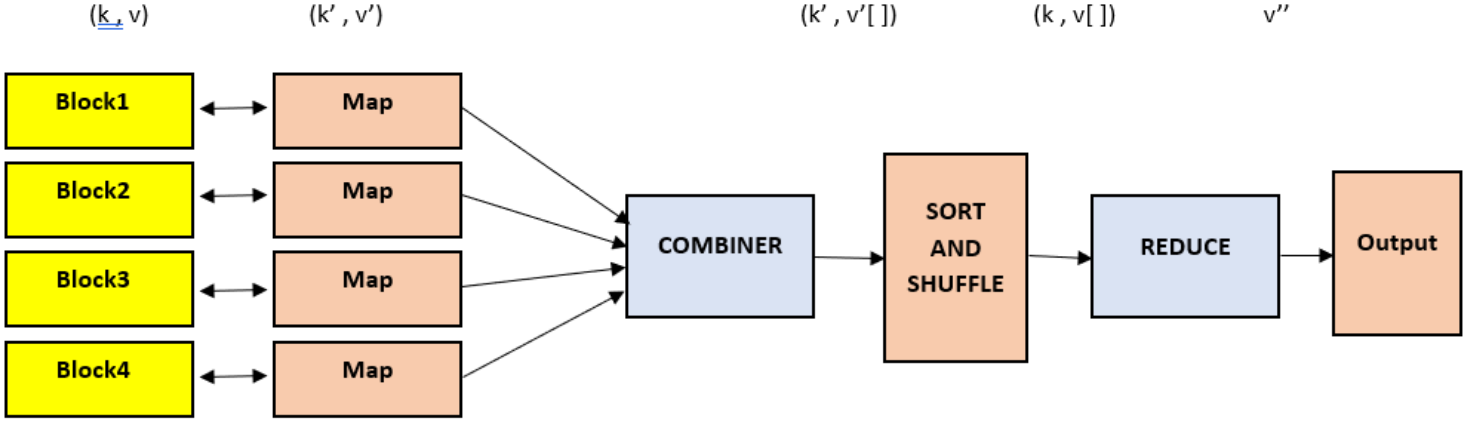
Tema 2

Arquitecturas Especializadas: MAPREDUCE. Fuciones generadoras.

J. M. Haut

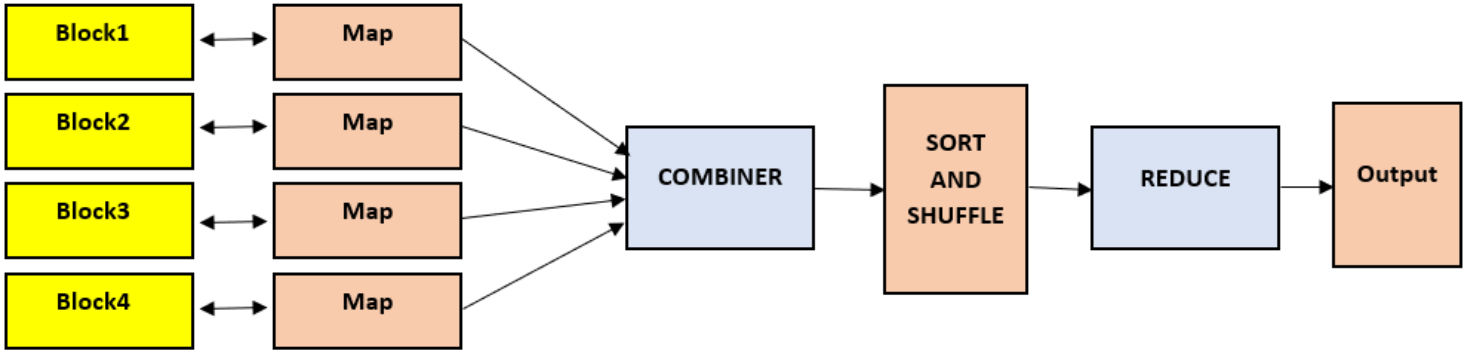
Programación Mapreduce

- La programación MapReduce es un modelo de procesamiento de datos diseñado para manejar conjuntos de datos grandes en un entorno distribuido y paralelo a través de un grupo de computadoras. Implica dos fases principales: "map" y "reduce".



Programación MapReduce

- **Fase de Map**: Los datos se dividen en fragmentos más pequeños, y se aplica una función "map" de manera independiente a cada fragmento para transformar los datos en pares clave-valor.
- **Fase de Reduce**: La salida de la fase de "map" se agrupa por clave, y se aplica una función "reduce" a cada grupo de datos con la misma clave para combinar y resumir datos relacionados, generando un resultado final.



Programación MapReduce

- MapReduce se usa ampliamente en sistemas de procesamiento de datos distribuidos y fue popularizado por Google. Los programadores definen las funciones "map" y "reduce" de acuerdo con las necesidades específicas de sus aplicaciones. Este enfoque permite el procesamiento eficiente en paralelo de grandes conjuntos de datos distribuidos.

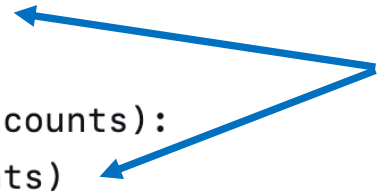
- Hadoop, un marco de código abierto, implementa MapReduce como un componente central para el procesamiento distribuido de datos. Aunque existen tecnologías más recientes como Apache Spark que ofrecen alternativas más rápidas y versátiles, MapReduce sigue siendo fundamental en el mundo de la computación distribuida.

Programación MapReduce

```
#!/usr/bin/env python
from mrjob.job import MRJob
class MRWordCount(MRJob):
    def mapper(self, _, line):
        for word in line.split():
            yield(word, 1)

    def reducer(self, word, counts):
        yield word, sum(counts)

if __name__ == '__main__':
    MRWordCount.run()
```



Funciones generadoras

Funciones Generadoras

- Una función generadora es un concepto de programación que se utiliza en lenguajes como **Python (NO C++ NO Java)**. Permite a una función pausar su ejecución y luego continuar desde donde se detuvo. Esto es útil cuando necesitas generar secuencias potencialmente largas de datos sin cargar todo en la memoria al mismo tiempo. En lugar de devolver una lista o un conjunto completo de datos, una función generadora produce un valor a la vez, lo que ahorra memoria y puede ser más eficiente.
- En Python, las funciones generadoras se crean utilizando la palabra clave yield. Cuando una función con **yield** se llama, no se ejecuta de inmediato, sino que crea un objeto generador que se puede usar para obtener valores uno por uno. Cada vez que se llama a **yield**, la función se pausa y "retiene" su estado hasta la próxima llamada, lo que permite continuar desde donde se quedó.

Funciones Generadoras

```python
def generador_ejemplo():
    yield 1
    yield 2
    yield 3

gen = generador_ejemplo()

print(next(gen))  # Imprimirá 1
print(next(gen))  # Imprimirá 2
print(next(gen))  # Imprimirá 3
```

Python

Funciones Generadoras

```cpp
#include <iostream>
#include <vector>

std::vector<int> generador_ejemplo() {
    std::vector<int> datos;
    datos.push_back(1);
    datos.push_back(2);
    datos.push_back(3);
    return datos;
}

int main() {
    std::vector<int> datos = generador_ejemplo();

    for (int valor : datos) {
        std::cout << valor << std::endl;
    }

    return 0;
}
```

C++

Funciones Generadoras

```java
import java.util.Iterator;

public class GeneradorEjemplo implements Iterable<Integer> {

    private Integer[] datos = {1, 2, 3};

    @Override
    public Iterator<Integer> iterator() {
        return new GeneradorIterator();
    }

    private class GeneradorIterator implements Iterator<Integer> {
        private int indice = 0;

        @Override
        public boolean hasNext() {
            return indice < datos.length;
        }

        @Override
        public Integer next() {
            return datos[indice++];
        }
    }
```

```java
    public static void main(String[] args) {
        GeneradorEjemplo generador = new GeneradorEjemplo();
        for (int valor : generador) {
            System.out.println(valor);
        }
    }
}
```

Java

Funciones Generadoras

- Ten en cuenta que, a diferencia de las funciones generadoras en Python, esta implementación en C++/Java no tiene la capacidad de pausar y reanudar la ejecución, ya que C++/Java no admite la noción de "pausar" una función para continuar más tarde. En lugar de eso,
  - En C++, simplemente se usa un contenedor para almacenar y devolver los datos generados.
  - En Java, se utiliza un enfoque basado en clases y objetos para generar datos de manera eficiente.

Tema 2

Arquitecturas Especializadas: MAPREDUCE. Fuciones generadoras.

J. M. Haut