Unidad 2: Modelo de mapeo y reducción

Algoritmos

Blanca Vázquez

Febrero 2022

Contexto

Imagen tomada de https://www.abc.es/

Contexto (2)

- Sincronización de tareas
  - Las operaciones de mapeo y reducción se ejecutan de manera simultánea
  - Los nodos de reducción reciben las llaves de manera ordenadas
- Creación de estructuras algebraicas que contribuyan a optimizar los procesos

Algoritmos del modelo MapReduce

El modelo de programación implementa diversos algoritmos matemáticos para dividir una tarea en pequeñas partes y para asignarlas a múltiples nodos.

- Ordenamiento
- Búsqueda
- Índice invertido

Ordenamiento

- Tarea: ordenar un conjunto de archivos, un valor por línea
- Algoritmo
  - Toma ventaja de las propiedades del reductor pares (llave,valor) las cuales son procesados en orden por llave. Los reductores se ordenan ellos mismos.

- Función de mapeo
  - La llave será el nombre de archivo y número de línea, el valor será el contenido de línea
  - Regresa el valor como llave $(k, v) \implies (v,)$

- Función de reducción
  - Función de identidad

Ordenamiento (2)

Aprovecha el ordenamiento de llaves por sistema de manejo de tareas, se define una función de partición tal que $k_1 < k_2 \implies hash(k_1) < hash(k_2)$

- Es usado por prueba de velocidad de Hadoop
- Es un carrera de resistencia .[en]tradas-salidas"

Búsqueda

- Tarea: encontrar documentos que contiene un patrón dado

- Función de mapeo
  - La llave será el nombre de archivo y número de línea, el valor será el contenido de línea
  - Regresa nombre de archivo como llave si se encuentre el patrón en el contenido

- Función de reducción
  - Identidad

Búsqueda (2)

- Una vez que se identificó al documento con el patrón, es necesario marcar ese documento (una sola vez)
- Se usa la función *Combiner* para convertir pares redundantes en un solo archivo (*filename*,)
- Reduce la entradas / salidas de la red

Función combinar

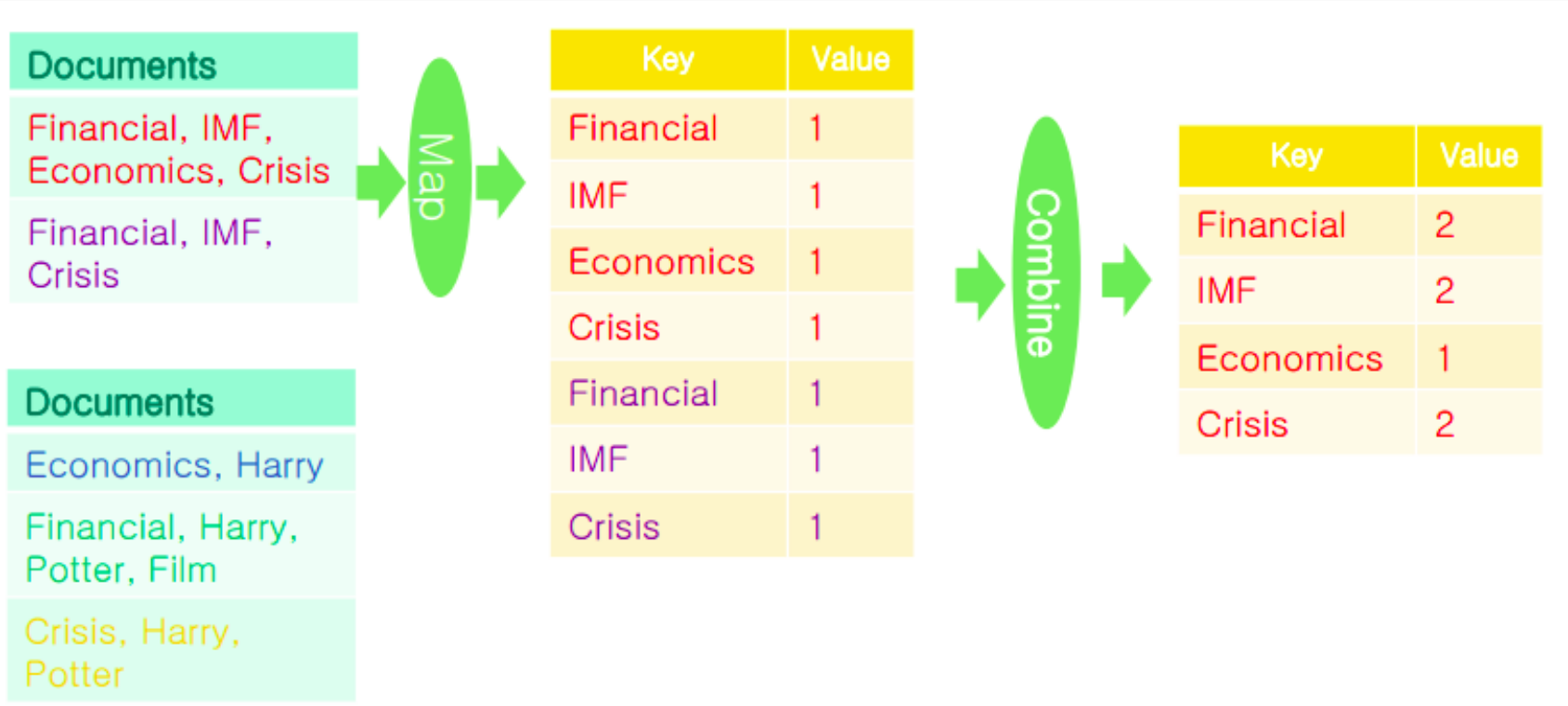

Imagen tomada de Kyuseok Shim, 2013.

Función combinar

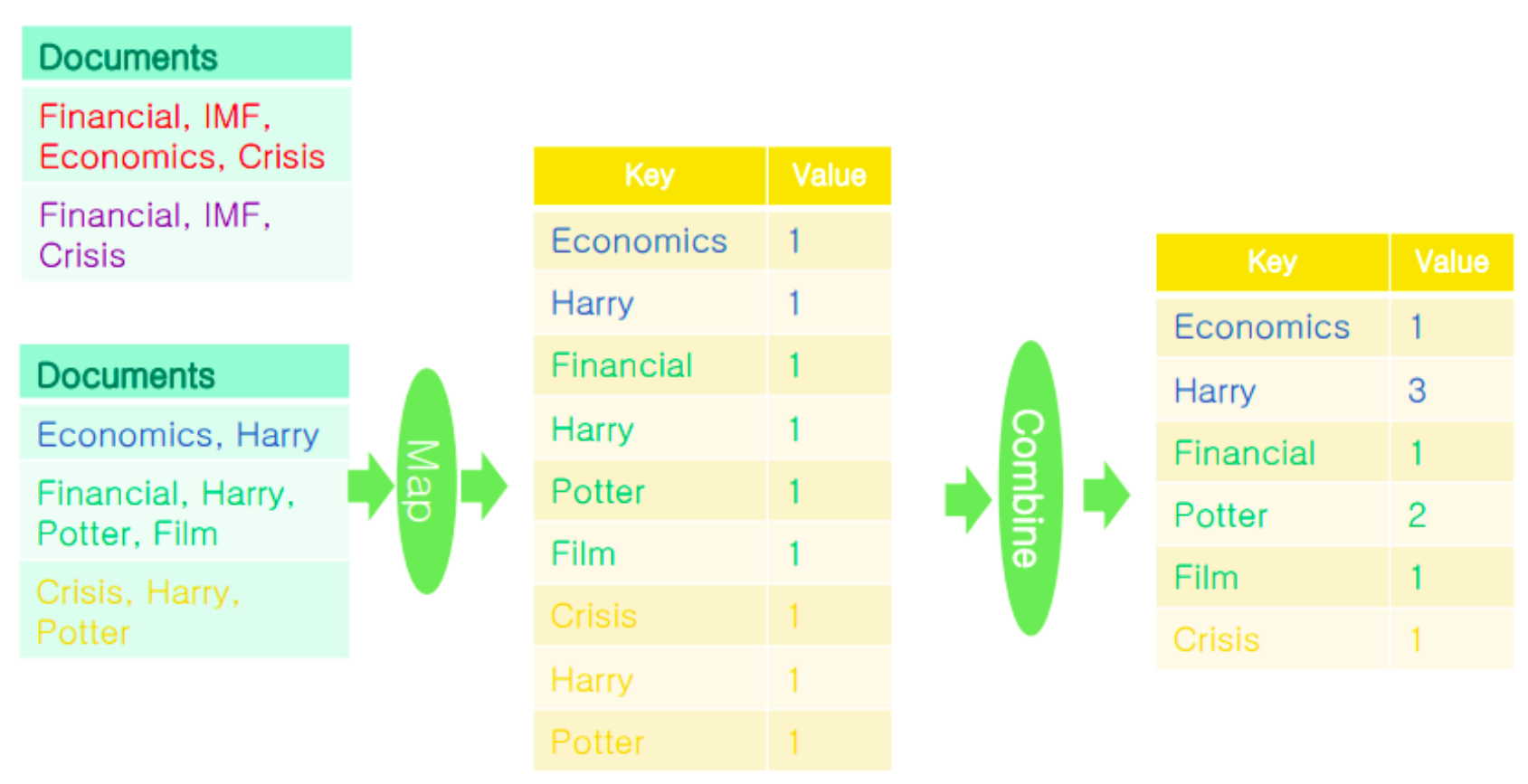

Imagen tomada de Kyuseok Shim, 2013.

Función combinar

Imagen tomada de Kyuseok Shim, 2013.

Índice invertido

Este algoritmo es el más utilizado en los sistemas de recuperación de información (ej., motores de búsqueda).

¿Cómo trabaja este algoritmo?

- Para cada término **t**, guardamos todos los documentos que contienen **t**
- Identificamos cada documento por un **docID**, el cuál es un número incremental

Índice invertido

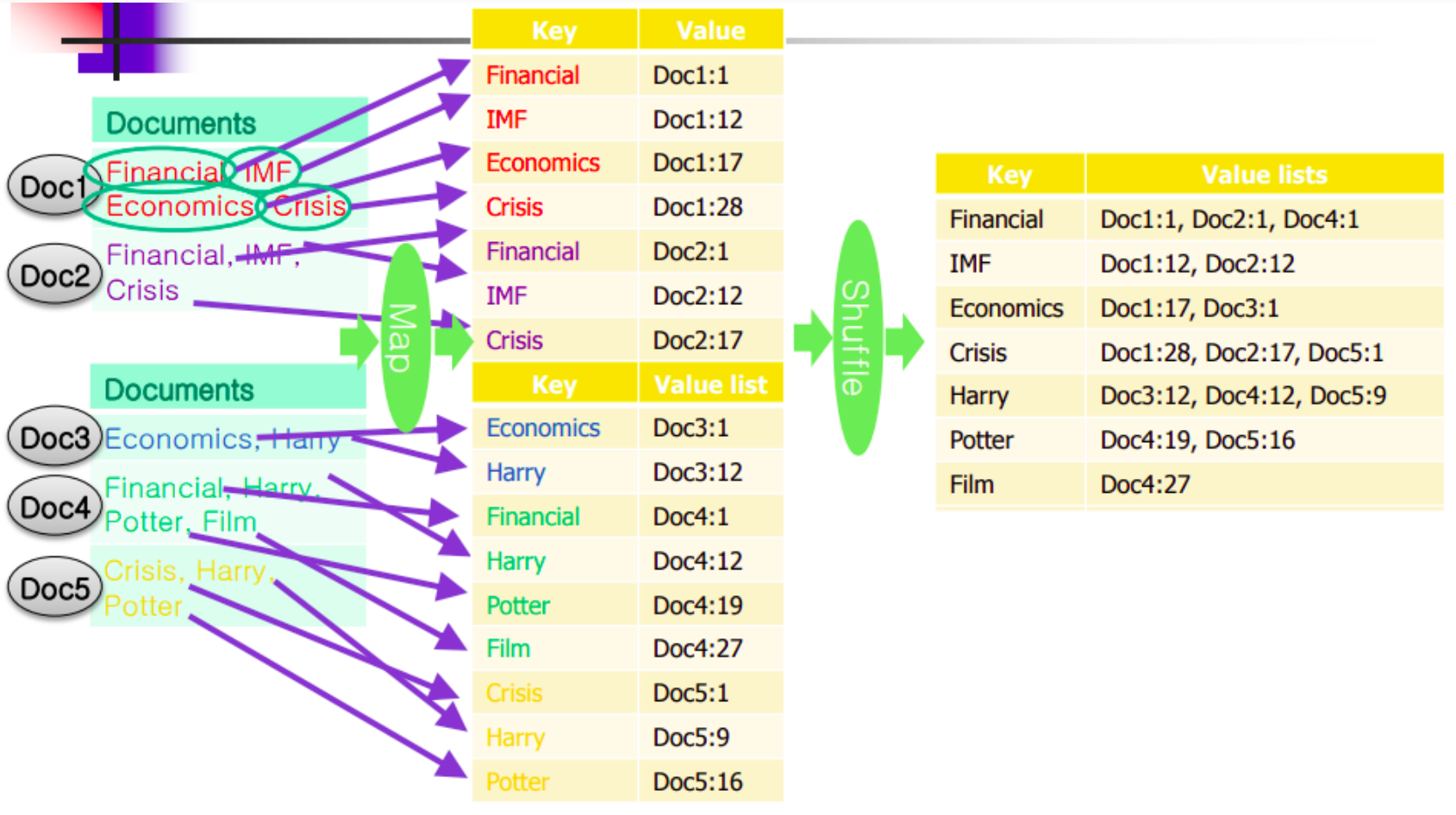

Imagen tomada de Kyuseok Shim, 2013.

¿Qué es un grafo?

- $G = (V, E)$
  - V representa un conjunto de vértices (nodos)
  - E representa un conjunto de aristas (enlaces)
  - Tanto vértices como aristas pueden contener información adicional
- Tipos de grafos
  - Dirigidos o no dirigidos
  - Con y sin ciclos
- Los G están en todas partes:
  - Redes sociales
  - Estructura de hipervínculos en la web
  - Estructuras físicas de las computadoras

Algunos problemas con los grafos

- Encontrar la ruta más corta
  - Tráfico en internet
- Búsqueda de árboles mínimos
  - Tendido de fibra óptica
- Encontrar el flujo máximo
  - Programación de aerolíneas
- Identificar comunicaciones 'especiales'
  - Terrorismo / conspiraciones
  - Enfermedades

Grafos y MapReduce

- El procesamiento con grafos involucra:
  - Ejecutar cálculos en cada uno de los nodos: basados en características de los nodos y de las aristas.
  - Propagar los cálculos 'atravesando' el grafo.
- Preguntas claves
  - ¿Cómo representar los datos de un grafo en MapReduce?
  - ¿Cómo recorrer el grafo usando MapReduce?

Representación de los grafos

- Los grafos comúnmente son representados como una matriz de adyacencias o una lista de adjacencias.

	1	2	3	4
1	0	1	0	1
2	1	0	1	1
3	1	0	0	0
4	1	0	1	0

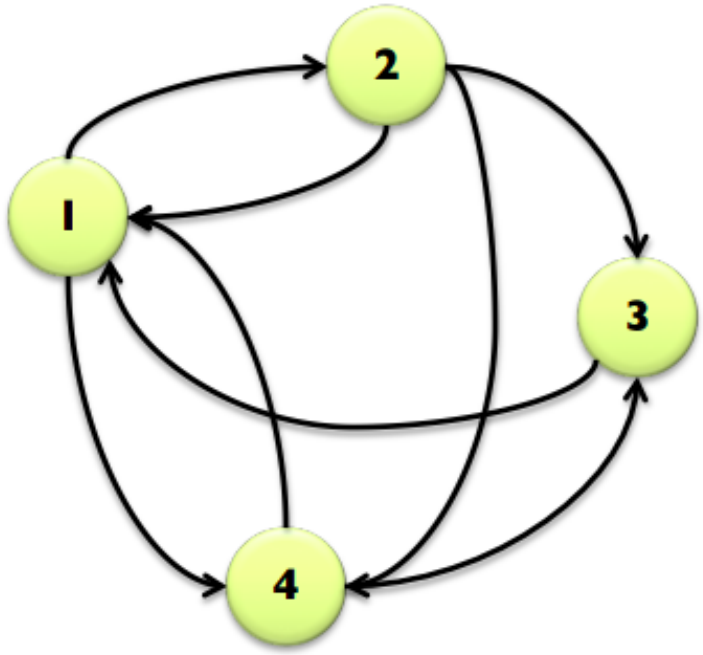

Imagen tomada de Jimmy Lin, 2013.

Ventajas de las matrices de adyacencias

- Ventajas
  - Manipulación accesible de los datos
  - Iteración sobre filas y columnas, correspondientes con los enlaces salientes y entrantes
- Desventajas
  - Muchos zeros
  - Ocupan gran cantidad de espacio

Listas de adyacencias

	1	2	3	4
1	0	1	0	1
2	1	0	1	1
3	1	0	0	0
4	1	0	1	0

1: 2, 4
2: 1, 3, 4
3: 1
4: 1, 3

Imagen tomada de Jimmy Lin, 2013.

Encontrar el camino más corto

```
1: class Mapper
2:     method Map(nid n, node N)
3:         d ← N.Distance
4:         Emit(nid n, N)                          ▷ Pass along graph structure
5:         for all nodeid m ∈ N.AdjacencyList do
6:             Emit(nid m, d + 1)                  ▷ Emit distances to reachable nodes

1: class Reducer
2:     method Reduce(nid m, [d1, d2, ...])
3:         d_min ← ∞
4:         M ← ∅
5:         for all d ∈ counts [d1, d2, ...] do
6:             if IsNode(d) then
7:                 M ← d                           ▷ Recover graph structure
8:             else if d < d_min then              ▷ Look for shorter distance
9:                 d_min ← d
10:        M.Distance ← d_min                      ▷ Update shortest distance
11:        Emit(nid m, node M)
```

Algoritmo BFS

Recordemos..

- Las tareas de mapeo y reducción
  - Son hilos independientes en cada nodo
- Funciones de combinar
  - Reduce el tamaño de las funciones de mapeo
  - Ejecuta mini-funciones de reducción en cada nodo
  - Disminuye el costo para el ordenamiento
- Cada tarea de mapeo y reducción puede opcionalmente usar dos funciones: *init*() y *close*()
  - *init*() llamado al inicio de cada tarea
  - *close*() llamado al final de cada tarea