DATOS MASIVOS I

UNIDAD II MODELO DE MAPEO Y REDUCCIÓN

ALGORITMOS DEL MODELO MAP – REDUCE

Raúl Galindo Hernández

16 de Febrero de 2023

Sincronización de Tareas

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

Sincronización de Tareas

- 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 Map – Reduce

- 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) $\Longrightarrow$ (v,)

- Función de reducción.
  - Función de identidad.

Ordenamiento

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 *entradas – salidas*.

Búsqueda

Tarea. Encontrar documentos que contienen 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

- 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.

Combiner Function

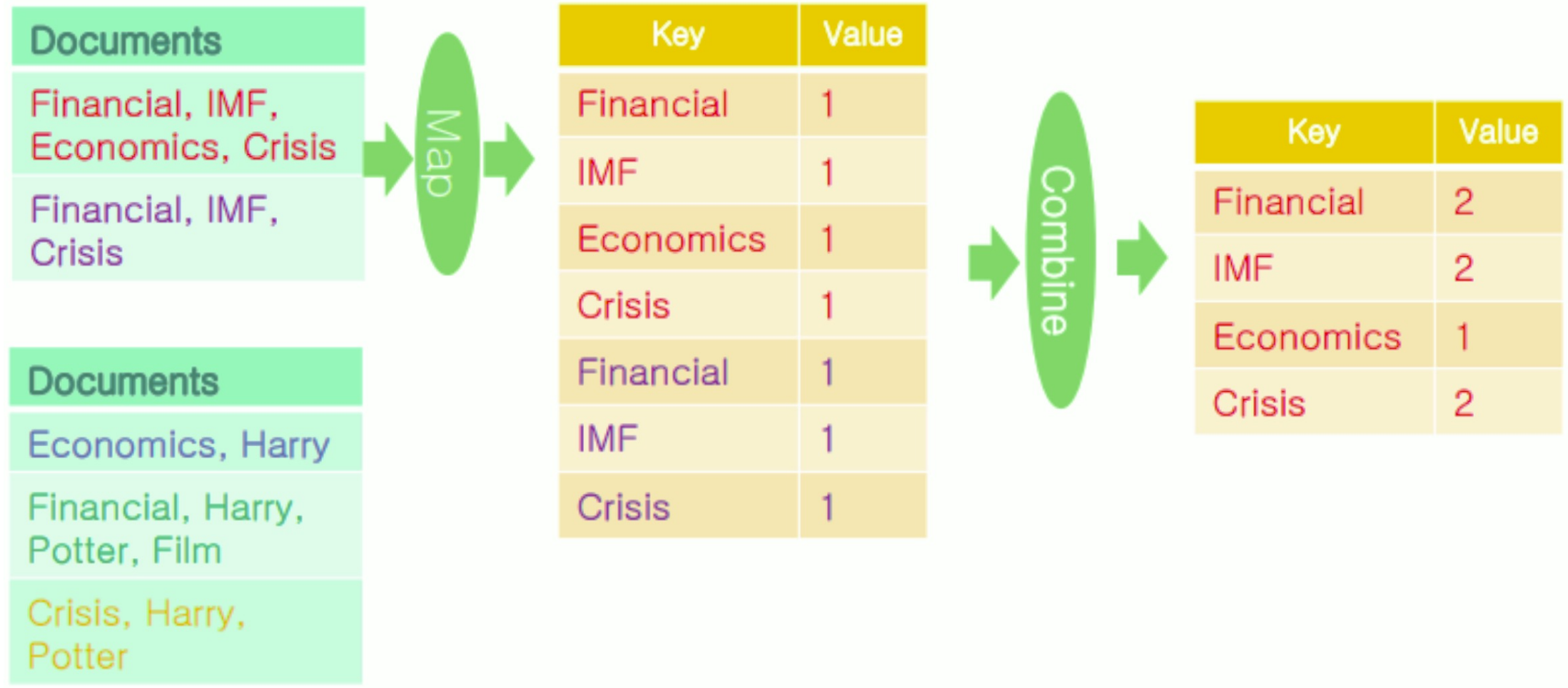


Imagen tomada de Kyuseok Shim, 2013.

Combiner Function

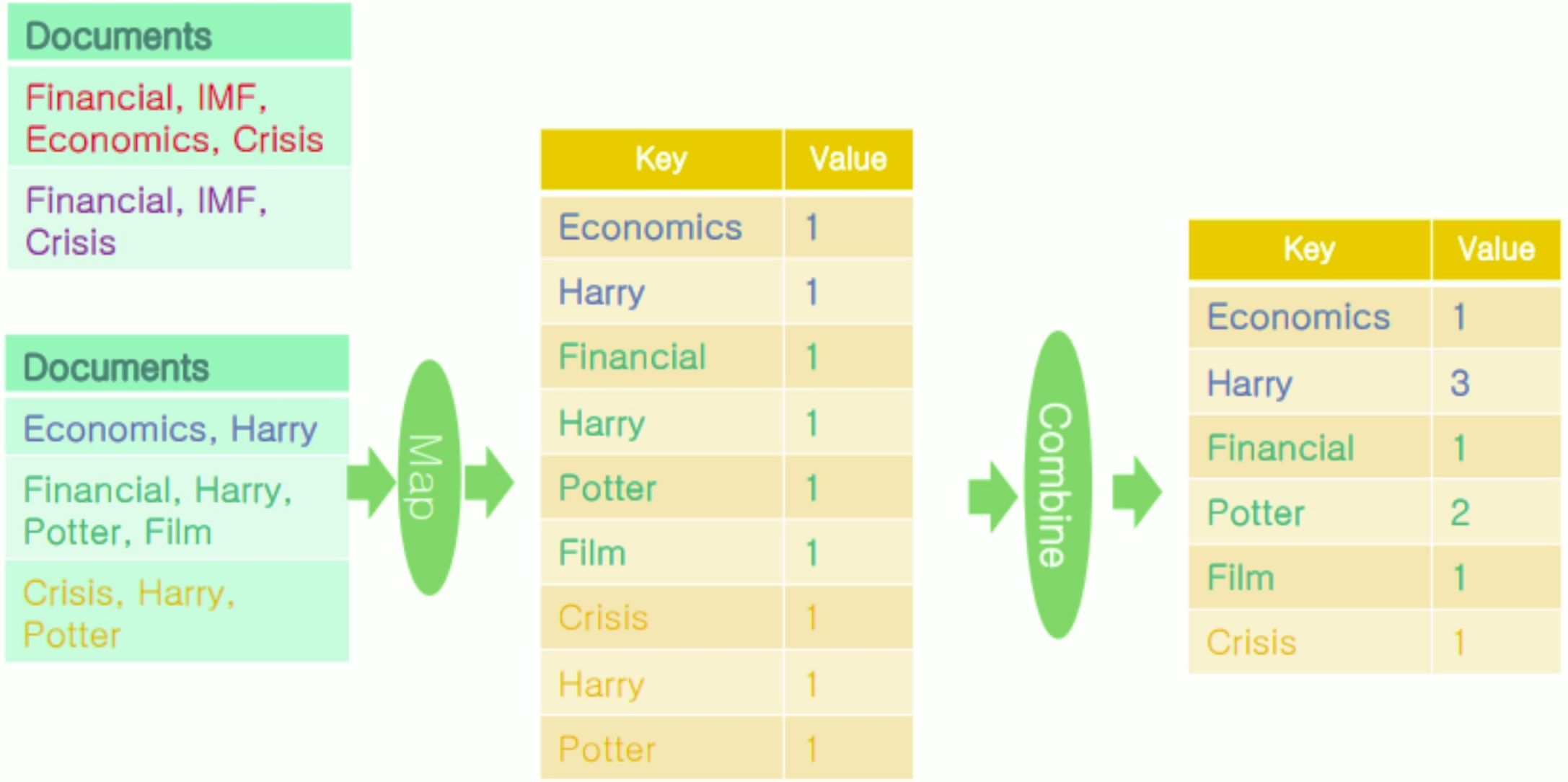


Imagen tomada de Kyuseok Shim, 2013.

Combiner Function

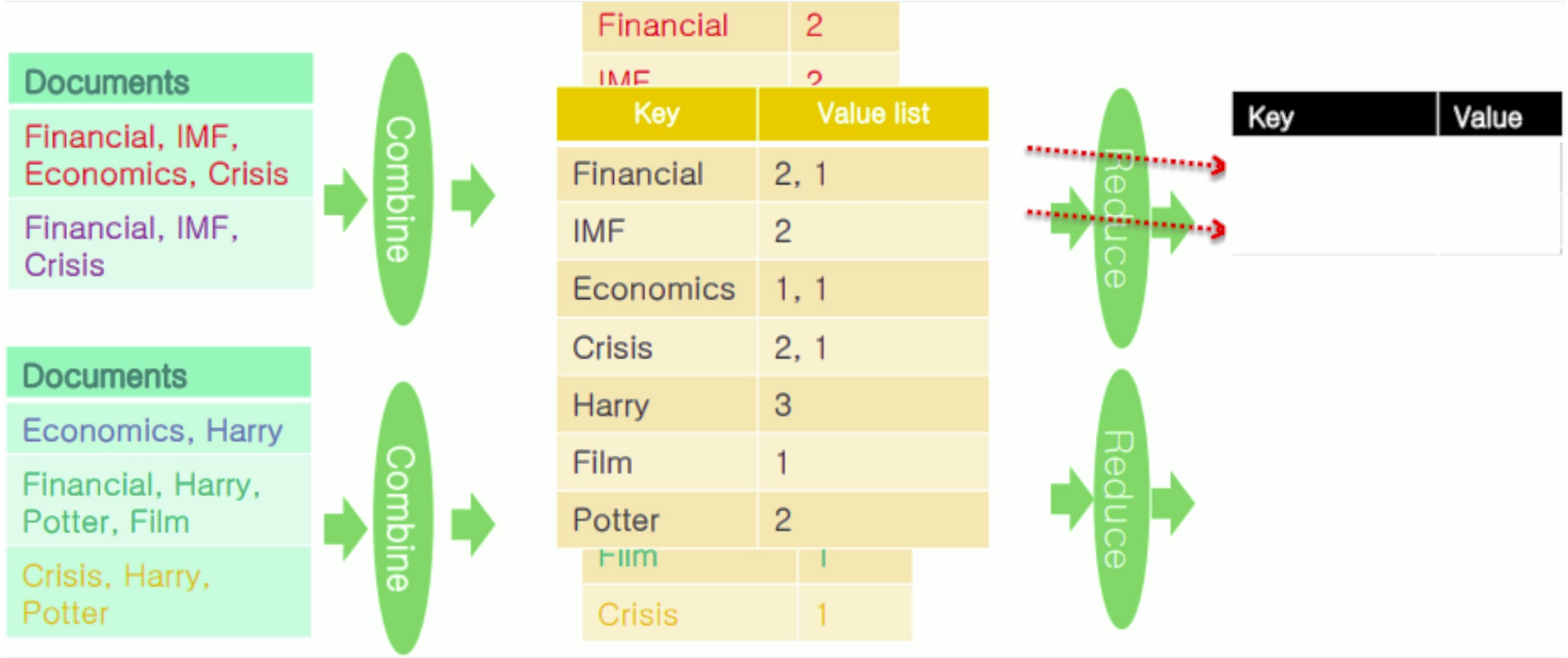


Imagen tomada de Kyuseok Shim, 2013.

Índice Invertido

- Este algoritmo es el más utilizado en los sistemas de recuperación de información (por ejemplo, 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

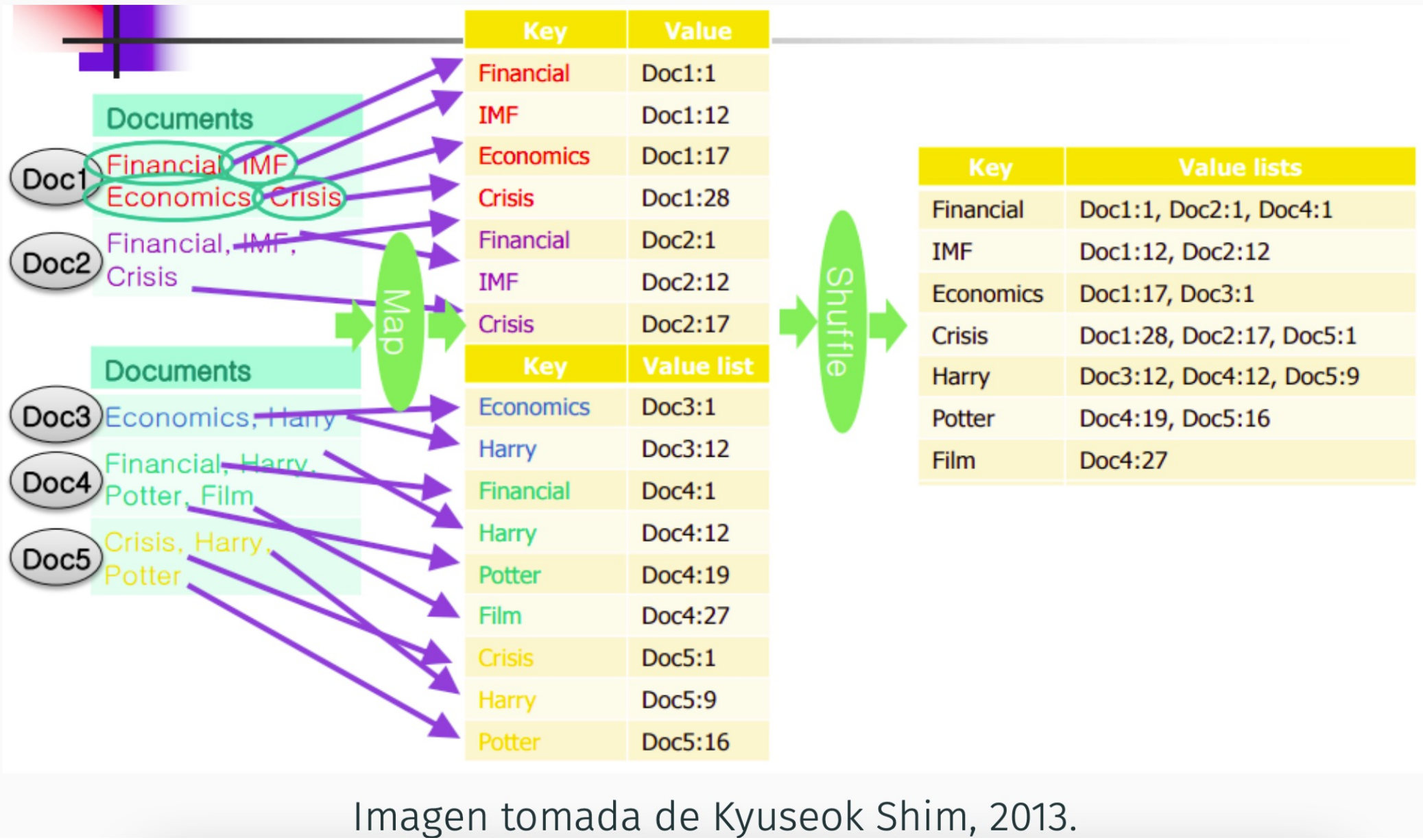


Imagen tomada de Kyuseok Shim, 2013.

Grafos, ¿Qué son?

$G = (V, E)$

- 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 dirígidos.
- 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 Map – Reduce

El procesamiento con grafos:

- ✓ 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 Map – Reduce?
- ¿Cómo recorrer el grafo usando Map – Reduce?

Grafos y Map – Reduce

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

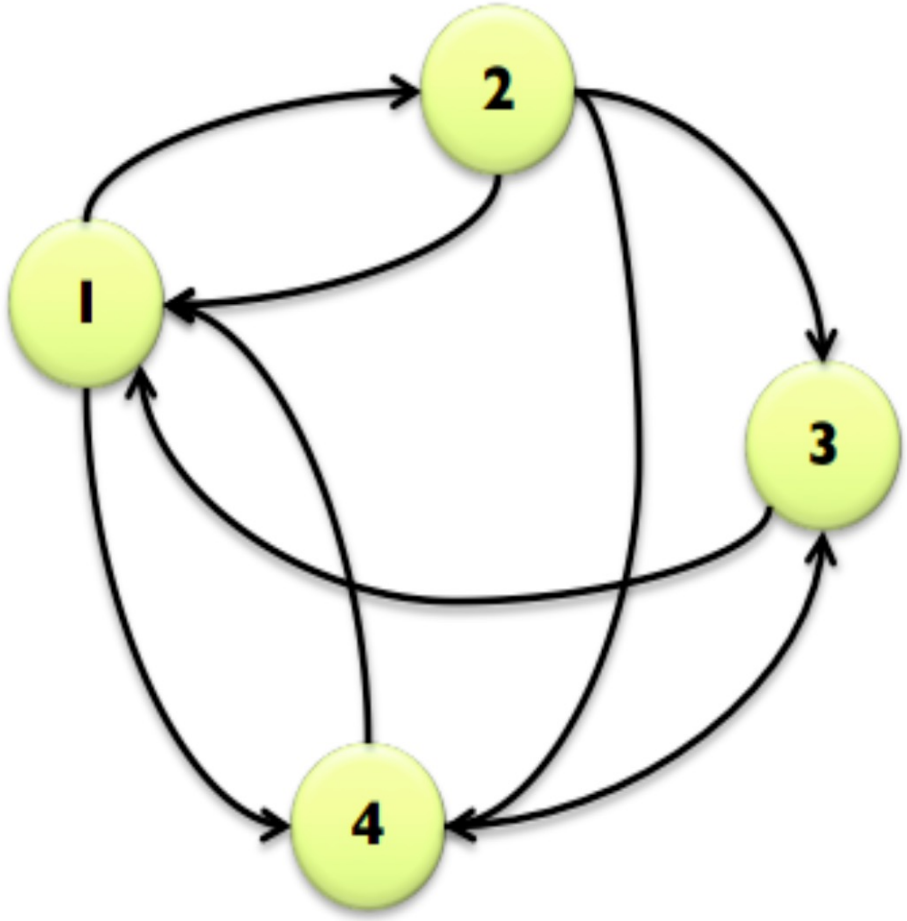


Imagen tomada de Jimmy Lin, 2013.

Matrices de Adyacencia: Pros and Cons

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

Listas de Adyacencia

	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 (Shortest Path: BFS)

- Realizar cálculos en una estructura de datos de grafos requiere procesamiento en cada nodo.
- Cada nodo contiene datos específicos del nodo, así como enlaces (edges) a otros nodos.
- El cálculo debe atravesar el gafo y realizar el paso de cálculo.

Encontrar el Camino Más Corto (Shortest Path: BFS)

- Realizar cálculos en una estructura de datos de grafos requiere procesamiento en cada nodo.

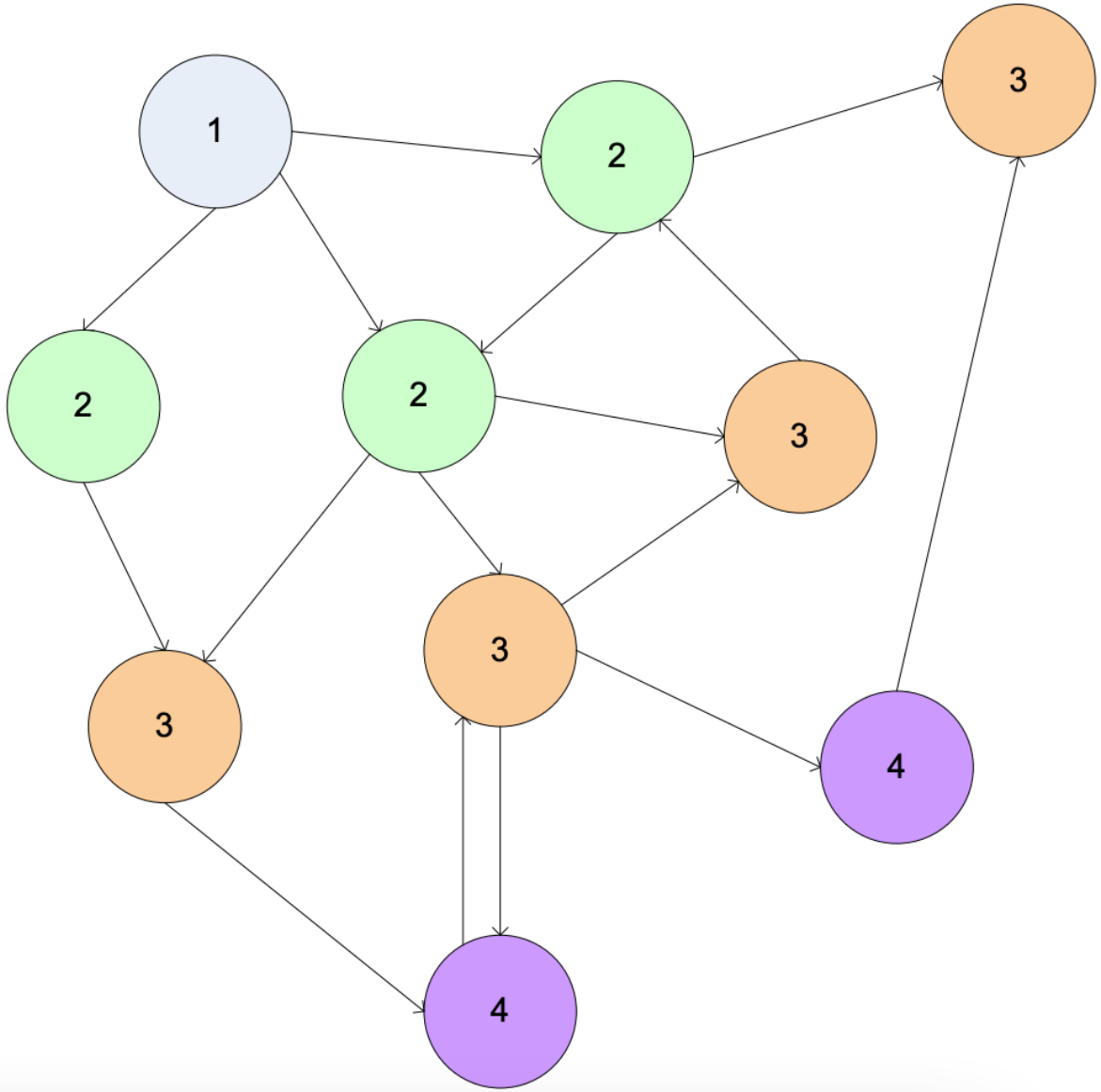
- Cada nodo contiene datos específicos del nodo, así como enlaces (edges) a otros nodos.

- El cálculo debe atravesar el gafo y realizar el paso de cálculo.

¿Cómo recorremos un grafo en MapReduce?
¿Cómo representamos el grafo para esto?

Encontrar el Camino Más Corto (Shortest Path: BFS)

- Breadth-First Search es un algoritmo iterativo sobre grafos.
- La frontera avanza desde el origen un nivel con cada paso.



Encontrar el Camino Más Corto (Shortest Path: BFS)

- Problema: Esto no “encaja” en Map – Reduce.

Encontrar el Camino Más Corto (Shortest Path: BFS)

- Solución: Pasos iterados a través del grafo.

- Map – Reduce: Mapea algunos nodos, el resultado incluye nodos adicionales que se introducen en pases sucesivos de Map – Reduce.

Recordando: Listas de Adyacencia

	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.

Intuición

- Podemos definir la solución a este problema de forma inductiva:

$$\textbf{Distancia(nodoDeInicio) = 0}$$

- Para todos los nodos *n* accesibles directamente desde nodoDeInicio,

$$\textbf{DistanceA(n) = 1}$$

- Para todos los nodos *n* accesibles desde algún otro conjunto de nodos S,

$$\textbf{DistanciaA(n) = 1 + min(DistanciaA(m), m} \in \textbf{S)}$$

De la Intuición al Algoritmo

- Una tarea de Mapeo recibe un nodo *n* como clave (llave) y (D, points-to) como su valor.
  - D es la distancia al nodo desde el inicio
  - points-to es una lista de nodos accesibles desde *n*
  - ∀ p ∈ points-to, emite (p, D+1)

- La tarea Reduce recolecta las distancias posibles a una *p* dada y selecciona la mínima.

Encontrar el Camino Más Corto (Shortest Path: BFS)

```
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

De la Intuición al Algoritmo

- Esta tarea de Map – Reduce puede avanzar a la frontera conocida en un salto.
- Para realizar todo el BFS, un componente que no es de Map – Reduce después alimenta la salida de este paso hacia atrás, a la tarea de Map – Reduce para otra iteración.
- Problema: ¿A dónde fue la lista de points-to pendientes?

De la Intuición al Algoritmo

Problema: ¿A dónde fue la lista de points-to pendientes?

Solución: Mapper también emite (*n*, points-to)

Criterio de Terminación (BFS)

- El algoritmo parte de un nodo inicial.
- Las iteraciones subsecuentes incluyen muchos más nodos del grafo a medida que la frontera avanza

¿Esto termina alguna vez?

Criterio de Terminación (BFS)

- ¡Sí! Eventualmente, se dejarán de descubrir rutas entre nodos y no se encontrarán mejores distancias (más cortas). Cuando la distancia es la misma, nos detenemos.

- El Mapper debe emitir (*n*, D) para garantizar que la "distancia actual" es llevada a Reduce.

Criterio de Terminación (BFS)

- El algoritmo de Dijkstra es más eficiente porque en cualquier paso del algoritmo, solo persigue bordes (edges) desde el camino de costo mínimo dentro de la frontera.

- La versión de Map – Reduce explora todas las rutas en paralelo; no es tan eficiente en general, pero la arquitectura es más escalable.

- Equivalente a Dijkstra para peso = 1

Recordando

Las tareas de mapeo y reducción:

- Son hilos independientes en cada nodo.

Combine Functions (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.

Recordando

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.