NoSQL

Agradecimientos

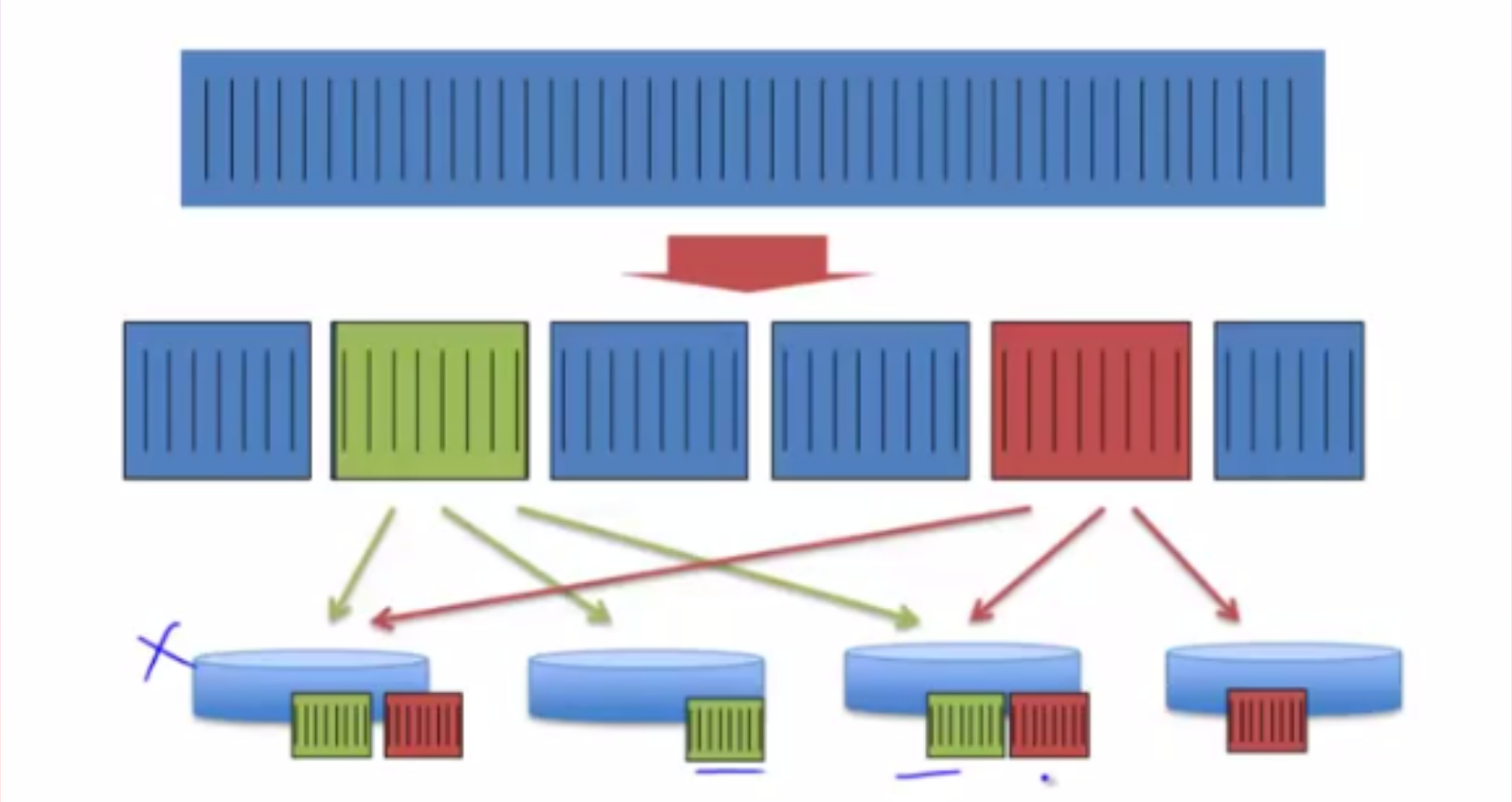
- Este curso es una versión libre del curso on-line Introducción a la Ciencia de Datos (Bill Howe, Univ. de Wasshington) https://www.coursera.org/course/datasci

NoSQL

- Sistemas de gestión de datos que surgieron para dar soporte a aplicaciones web de gran tamaño
- Las bases de datos relacionales no pueden escalar a 1000s de máquinas
  - Limitaciones relacionadas con la consistencia y la transaccionalidad

Contexto operativo

- Escalabilidad (1000s de nodos)
- Alta disponibilidad
- Tolerancia a fallos
- Soportar updates de las réplicas (copias redundantes)



Consistencia

UNIVERSITY of WASHINGTON

Example

User: Joe
Friends: Sue, …
Status: "I'm sleepy"
Wall: "…", "…"

User: Sue
Friends: Joe, Kai, …
Status: "Headed to new Bond flick"
Wall: "…", "…"

User: Kai
Friends: Sue, …
Status: "Done for tonight"
Wall: "…", "…"

Write: Update Sue's status. Who sees the new status, and who sees the old one?

Databases: *"Everyone MUST see the same thing, either old or new, no matter how long it takes."*

NoSQL: *"For large applications, we can't afford to wait that long, and maybe it doesn't matter anyway"*

Consistencia eventual

El modelo de consistencia que predomina en las bases de datos NoSQL se denomina *consistencia eventual*:

- En caso de un update, las réplicas convergen a la misma información a lo largo de un cierto tiempo (o sea: no es inmediato como en el modelo transaccional puro del modelo relacional)
- Lo que las aplicaciones ven mientras tanto es difícil de predecir debido a esta naturaleza progresiva de propagación de los cambios

Características generales de Bases NoSQL

- Capacidad de escalar horizontalmente (muchos nodos) operaciones simples:
  - Búsquedas por clave, leer o escribir un conjunto reducido de registros
- Capacidad de replicar y particionar la información entre muchos nodos
- API simple (no SQL)
- Modelo de concurrencia simple (no transaccional)
- Uso eficiente de índices distribuidos y de la memoria para el almacenamiento de los datos
- Capacidad de agregar dinámicamente nuevos atributos a los registros (sin esquema predefinido)

Taxonomía de Bases NoSQL

Es posible clasificar las herramientas NoSQL de acuerdo a la caracterización de la información que procesan:

- **Documento**: estructuras anidadas (XML, JSON)
- **Registros extensibles**: conjuntos de atributos (esquema); se pueden agregar atributos dinámicamente
- **Objetos Key-Value**: conjuntos de pares clave-valor (sin anidamiento)

Bases NoSQL más influyentes

Los productos más importantes y sus respectivos aportes son:

- **Memcached** (Key-Value Store): índices en memoria escalables; distribuir y replicar objetos entre muchos nodos
- **Dynamo** (Amazon; Key-Value Store): consistencia eventual (mayor disponibilidad y escalabilidad si se relaja el requerimiento de consistencia pura)
- **BigTable** (Google): escalar a miles de nodos la persistencia de registros

Memcached

- Esencialmente es un servicio de cache en memoria principal
- Se puede instalar encima de cualquier base de datos para mejorar el tiempo de respuesta
- Introdujo el concepto de “consistent hashing” para relocalizar la información en caso de que cambie la cantidad de nodos del cluster

Técnica de Hashing (tradicional)

"Regular" Hashing

Assign M data keys to N servers

assign each key to server = k mod N

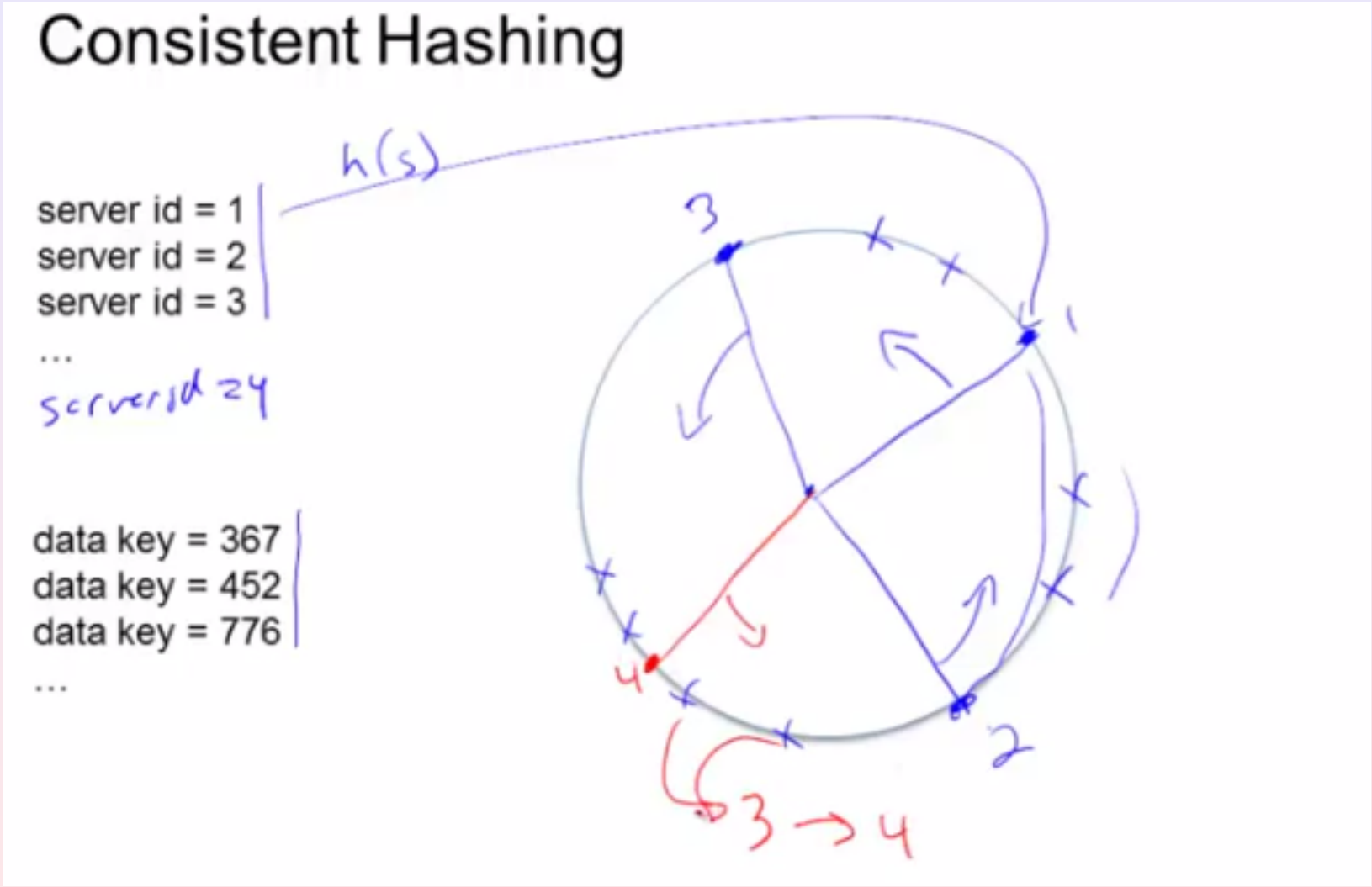
Example: N=3

key 0 -> server 0
key 1 -> server 1
key 2 -> server 2
key 3 -> server 0
key 4 -> server 1
...

What happens if I increase the number of servers from N to 2N?

Every *existing key needs to be remapped, and we're screwed.*

Consistent Hashing

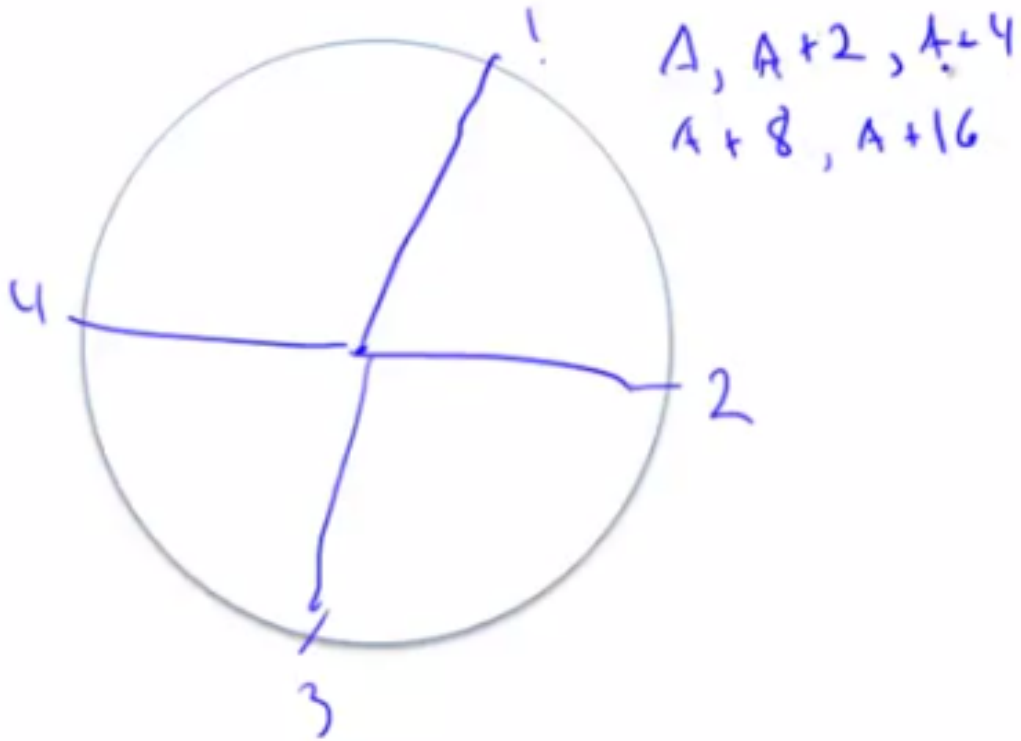


Ruteo de consultas en Consistent Hashing

- Tiempo de respuesta en orden logarítmico

Consistent Hashing: Routing

server id = 1
server id = 2
server id = 3
...

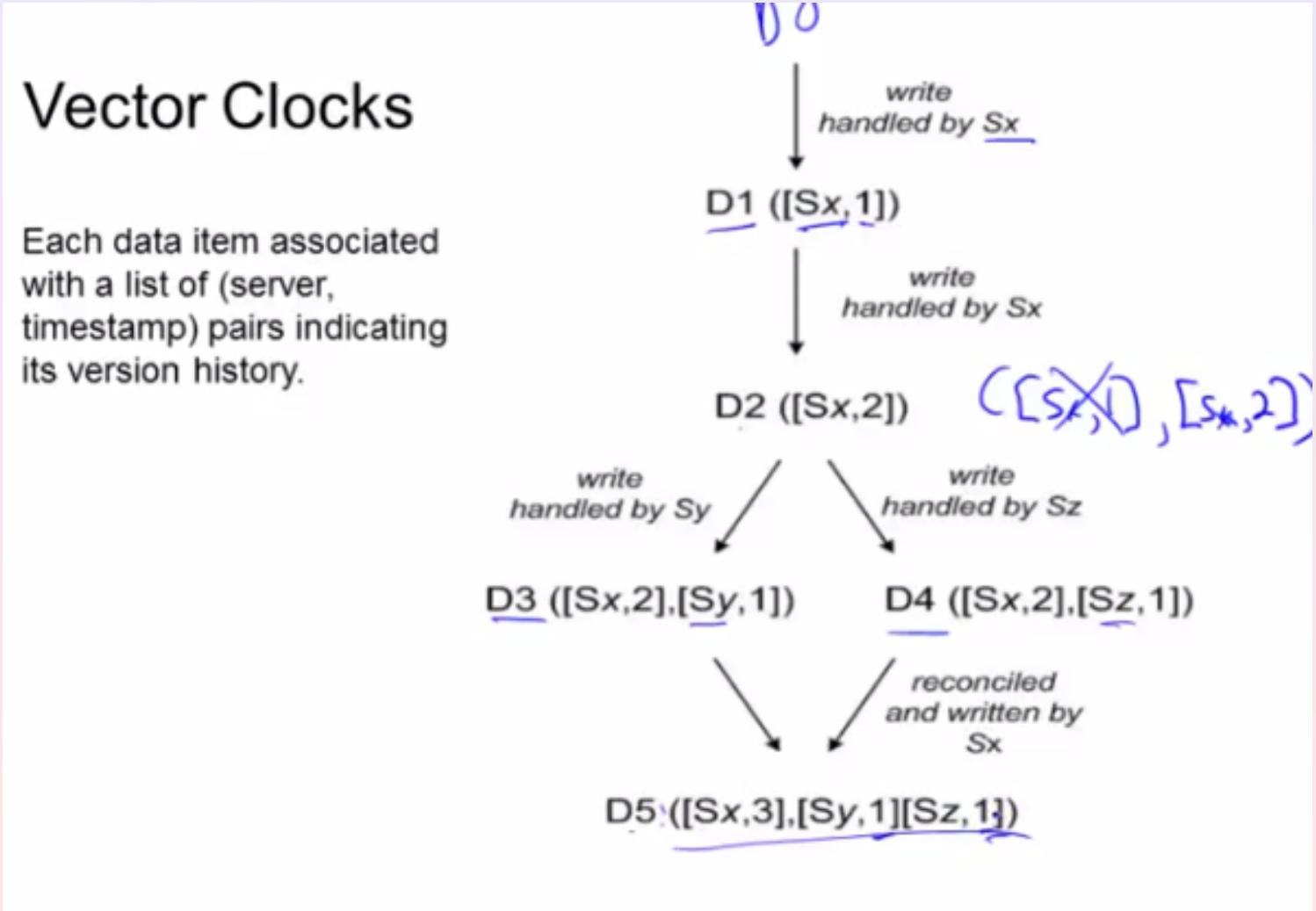


DynamoDB (Amazon)

- Implementación: DHT (Dynamic Hash Table)
- Replicación: la misma información está replicada en varios nodos (tolerancia a fallos, alta disponibilidad)
- La consistencia eventual se implementa mediante la técnica de "vector clocks"
- La consolidación de la información se hace en el momento de la lectura:
  - Las escrituras no fallan nunca (es decir: no se reportan errores de concurrencia en las escrituras porque es costoso controlarlo)
  - Resolución de conflictos: la "última escritura gana"

Técnica de Vector Clocks

Los *vector clocks* son vectores de pares (Server, Timestamp) asociados a los valores que se van modificando



Técnica de Vector Clocks

Vector Clocks Example

- A client writes D1 at server SX:
  D1 ([SX,1])
- Another client reads D1, writes back D2; also handled by SX:
  D2 ([SX,2]) (D1 garbage collected)
- Another client reads D2, writes back D3; handled by server SY:
  D3 ([SX,2], [SY,1])
- Another client reads D2, writes back D4; handled by server SZ:
  D4 ([SX,2], [SZ,1])
- Another client reads D3, D4: CONFLICT !

Técnica de Vector Clocks

Data 1	Data 2	Conflict?
([SX,3],[SY,6])	([SX,3],[SZ,2])	
([SX,3])	([SX,5])	
([SX,3],[SY,6])	([SX,3],[SY,6],[SZ,2])	
([SX,3],[SY,10])	([SX,3],[SY,6],[SZ,2])	
([SX,3],[SY,10])	([SX,3],[SY,20],[SZ,2])	

CouchDB

- Procesamiento de documentos
  - Los documentos son conjuntos de pares clave-valor (JSON, XML)
- Soporta búsquedas por clave secundaria (o sea: por otros valores además de la clave primaria)
- Garantiza cierto nivel de transaccionalidad a nivel de documento
- Implementa vistas
- Implementa Map-Reduce

Joins

- Algo importante que las bases de datos NoSQL relegan (en general) es la posibilidad de hacer joins
- Hay variantes que colocalizan los datos de modo de que ciertas entidades queden anidadas (ej.: en un blog las entidades post y comentario pueden quedar intercaladas)

Algunas críticas a NoSQL

- No soportan ACID (como las bases de datos transaccionales)
  - Atomicity: las transacciones se ejecutan de modo todo (commit) o nada (rollback)
  - Consistency: las transacciones pasan la base de un estado válido a otro (en relación a la reglas del modelo de datos)
  - Isolation: las transacciones no se interfieren; el resultado de la ejecución es como si las transacciones se hubieran ejecutado en serie (en algún orden)
  - Durability: los cambios son persistentes (tolerancia a fallos); log
- No tienen un lenguaje de alto consulta de alto nivel (SQL)
- No están estandarizadas