Bases de datos distribuidas II

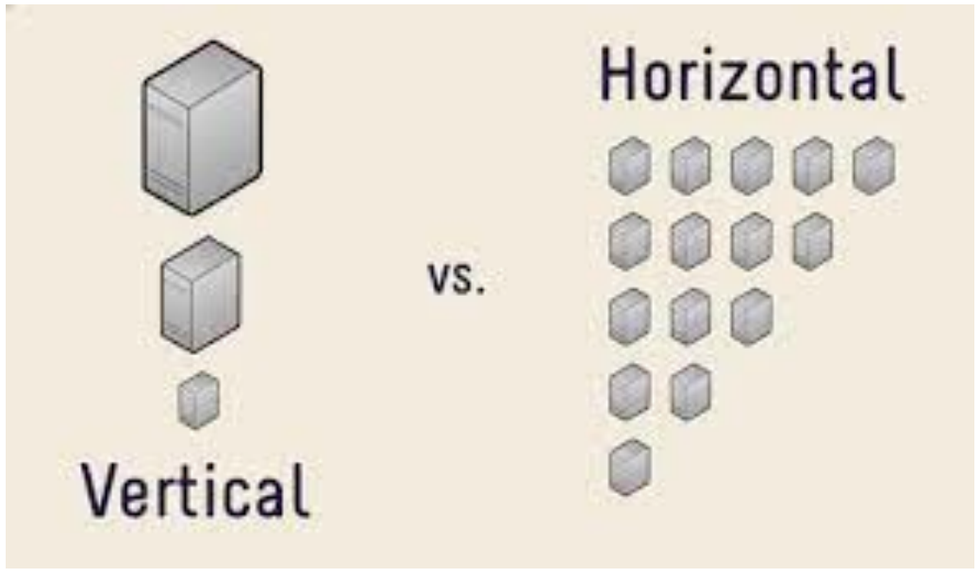
Alex Di Genova

21/11/2022

Bases de datos Distribuidas

Sistemas centralizados

- Escalamiento vertical
    - Cada vez que había requisitos de mayor volumen de datos o escrituras más frecuentes, la reacción obvia era equipar el servidor de base de datos con mayor capacidad en términos de velocidad de procesador, tamaño de memoria o espacio en disco.
- Escalamiento horizontal
    - Conectar varios servidores más en una red permite mejorar el rendimiento y la latencia de un sistema de base de datos a costa de la **coordinación y sincronización** de los servidores.



Una base de datos distribuida es una colección de datos que se distribuyen físicamente en varios servidores interconectados por una red mientras que lógicamente estan relacionados.

Bases de datos Distribuidas

fragmentación de datos

- En un sistema de base de datos distribuida, dos preguntas importantes son (1) cómo se puede dividir el conjunto completo de datos en la base de datos en subconjuntos (**fragmentación de datos**) y (2) cómo se pueden distribuir los subconjuntos entre los servidores de base de datos en la red (**asignación de datos**).

- Una vez que se ha establecido una buena fragmentación y asignación, una base de datos distribuida puede aprovechar:

  - **Localidad de datos** (idealmente, los registros de datos en el mismo fragmento a menudo se acceden juntos),

  - **Minimización de los costos de comunicación** (idealmente, no es necesario mover registros de datos a otro servidor para responder una consulta),

  - **Mejor eficiencia de la gestión de datos** (idealmente, las consultas en fragmentos más pequeños se pueden ejecutar más rápido que en el gran conjunto de datos y las estructuras de índice pueden ser más pequeñas y, por lo tanto, los registros de datos se pueden encontrar más rápido),

  - **Balanceo de carga** (idealmente, a todos los servidores se les asigna la cantidad óptima de datos y solo tienen que procesar las consultas de los usuarios de acuerdo con las capacidades de cada servidor y sin el peligro de los puntos de acceso, es decir, sin sobrecargar algunos servidores con la mayoría de los usuarios). consultas mientras que los otros servidores están en su mayoría inactivos)

Bases de datos Distribuidas

Typos de fragmentación

- **Manual**: una fragmentación manual requiere que el administrador de base de datos identifique fragmentos en un conjunto de datos y configure el sistema utilizando estos fragmentos.
- **Random**: cada registro de datos tiene la misma probabilidad de ser asignado a un fragmento.
- **Basada en la estructura**: Analiza la definición del esquema de datos (o el modelo de datos en general) e identifica las subestructuras que constituyen los fragmentos.
- **Basada en valores**: Analiza los valores contenidos en los elementos de datos para definir los fragmentos.
- **Basada en rangos:** como una forma especial de fragmentación basada en valores, la fragmentación basada en rangos divide la clave principal (u otro atributo que se puede ordenar) en intervalos inconexos pero consecutivos. Cada intervalo define un fragmento.
- **Basada en hashing**: si cada registro de datos tiene una clave el hash de esta clave puede determinar el servidor de base de datos al que se asigna el registro.
- **Basada en costos**: Se basa en una función de costos para encontrar una buena fragmentación. El objetivo es encontrar una fragmentación con un costo total mínimo.
- **Basada en afinidad:** Se basa en una especificación de qué tan afines son ciertos registros de datos; con qué frecuencia se accede a ellos juntos en una solicitud de lectura.
- **Clustering**: se pueden usar algoritmos especializados para encontrar subconjuntos coherentes en los datos (los llamados clústeres).

Bases de datos Distribuidas

fragmentación de modelos relacionales

- **Vertical**: Subconjuntos de columnas forman los fragmentos (**basada en estructura**) e identificables por una tupla clave (recombinacion mediante **joins**). En el ejemplo suponemos que A,B y C,D son más afines que los otros subset de columnas.

- **Horizontal**: Subconjuto de filas forman los fragmentos (**basada en valores**). La recombinación se logra tomando la **unión** de las tuplas en los diferentes fragmentos.

- **Derivada**: Una fragmentación horizontal en una tabla primaria induce una fragmentación horizontal en una tabla secundaria relacionada. Los fragmentos primarios y derivados con valores coincidentes para los atributos de unión se pueden almacenar en el mismo servidor

- **Hibrida**: Combinación arbitraria de fragmentacion horizontal y vertical.

- Fragmentaciones para bases de datos basadas en archivos, clave-valor y grafos siguen las mismas ideas.

Original data	A	B	C	D
	a_1	b_1	c_1	d_1
	a_2	b_2	c_2	d_2
	a_3	b_3	c_3	d_3

Fragment 1	ID	A	B
	1	a_1	b_1
	2	a_2	b_2
	3	a_3	b_3

Fragment 2	ID	C	D
	1	c_1	d_1
	2	c_2	d_2
	3	c_3	d_3

Original data	A	B	C	D
	a_1	b_1	c_1	d_1
	a_2	b_2	c_2	d_2
	a_3	b_3	c_3	d_3

Fragment 1	A	B	C	D
	a_1	b_1	c_1	d_1
	a_2	b_2	c_2	d_2

Fragment 2	A	B	C	D
	a_3	b_3	c_3	d_3

Bases de Datos Distribuidas

Asignación de datos

- Los fragmentos idealmente deben distribuirse uniformemente en todos los servidores (balanceo de carga).
- Tipos de asignación:
  - **Basada en rangos**
  - **Basada en costos**: Problema de optimización que intentamos programación lineal.
    - Similar al problema de ensamble de contenedores (bin pa
      - K servidores corresponden a K contenedores
      - Los contenedores tienen una capacidad maxima W
      - n framentos corresponden a n objetos.
      - Cada objeto tiene u peso (capacidad) wi < W
      - Los objetos deben ser ubicados en un numero minim sin exeder la capacidad maxima.
    - En una formulación simple podemos considerar la capacidad de almacenamiento (W) de los servidores y el consumo de cada fragmento (wi).

$$\text{minimize} \sum_{k=1}^{K} y_k \qquad \text{(minimize amount of servers)}$$

$$\text{subject to} \sum_{k=1}^{K} x_{ik} = 1 \qquad \text{(each fragment } i \text{ assigned to one server)}$$

$$\sum_{i=1}^{n} w_i \cdot x_{ik} \le W \cdot y_k \qquad \text{(capacity of each server } k \text{ not exceeded)}$$

$$y_k \in \{0, 1\} \qquad k = 1, \ldots, K$$

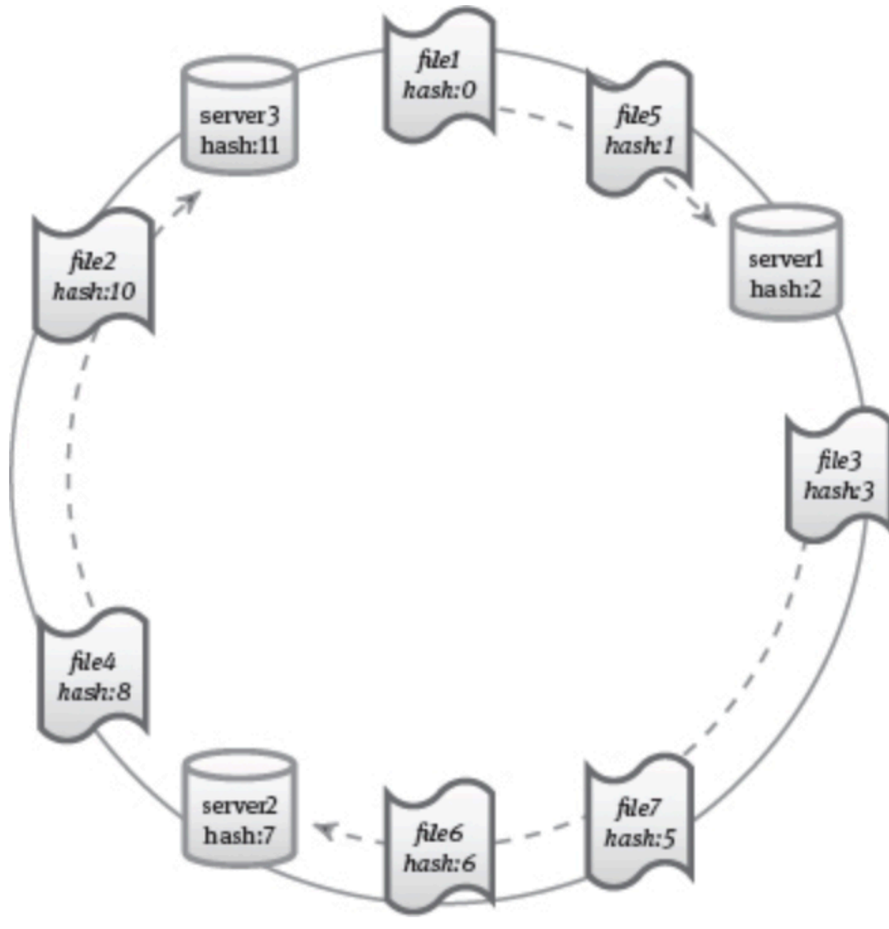
$$x_{ik} \in \{0, 1\} \qquad k = 1, \ldots, K, \;\; i = 1, \ldots, n$$

- **Basada en hashing**: utiliza una función hash (como MD5 o MurmurHash) sobre los fragmentos para determinar el servidor al que se asigna cada fragmento.

Bases de Datos Distribuidas

Asignación de datos hashing

- Es la estrategia de asignacion más usada.
- Los valores hash se ven como un anillo: cuando alcanzamos el valor hash más alto, comenzamos nuevamente desde 0.
- El valor hash se calcula no solo para cada fragmento sino también para cada servidor de base de datos.
- Al calcular un valor hash para un servidor de base de datos, cada servidor tiene una posición fija en el anillo; la ventaja de estos valores hash es que presumiblemente distribuyen los servidores uniformemente en el anillo. De manera similar, para cada elemento de datos se calcula un valor hash. De esta manera, los datos también se distribuyen bien en el anillo.



Bases de Datos Distribuidas

Replicas y sincronización

- La replicación se refiere al concepto de almacenar varias copias de un registro de datos en diferentes servidores de bases de datos (numero de copias predeterminado -> factor).

  - Mejora la confiabilidad del sistema de base de datos distribuida al ofrecer una mayor disponibilidad de datos: si una de las réplicas no puede manejar una consulta de usuario, otra réplica puede tomar el relevo.

  - Ofrece una latencia más baja que un sistema no replicado al permitir el balanceo de carga, la localidad de datos y la paralelización: cualquier réplica puede responder a las consultas de un usuario.

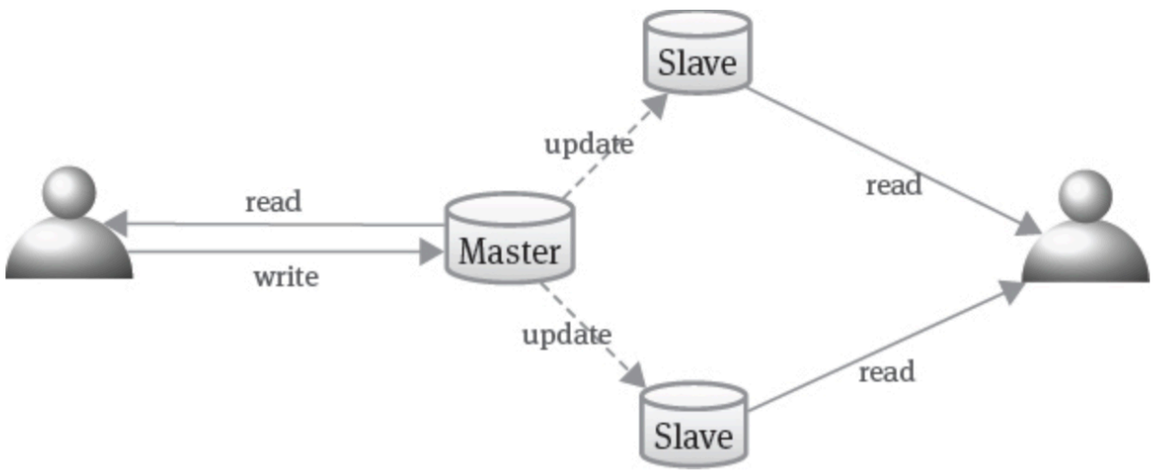
  - Problema de concurrencia: dos o más usuarios pueden actualizar simultáneamente el mismo registro de datos en diferentes réplicas y el sistema de base de datos debe ofrecer un mecanismo para resolver este conflicto.

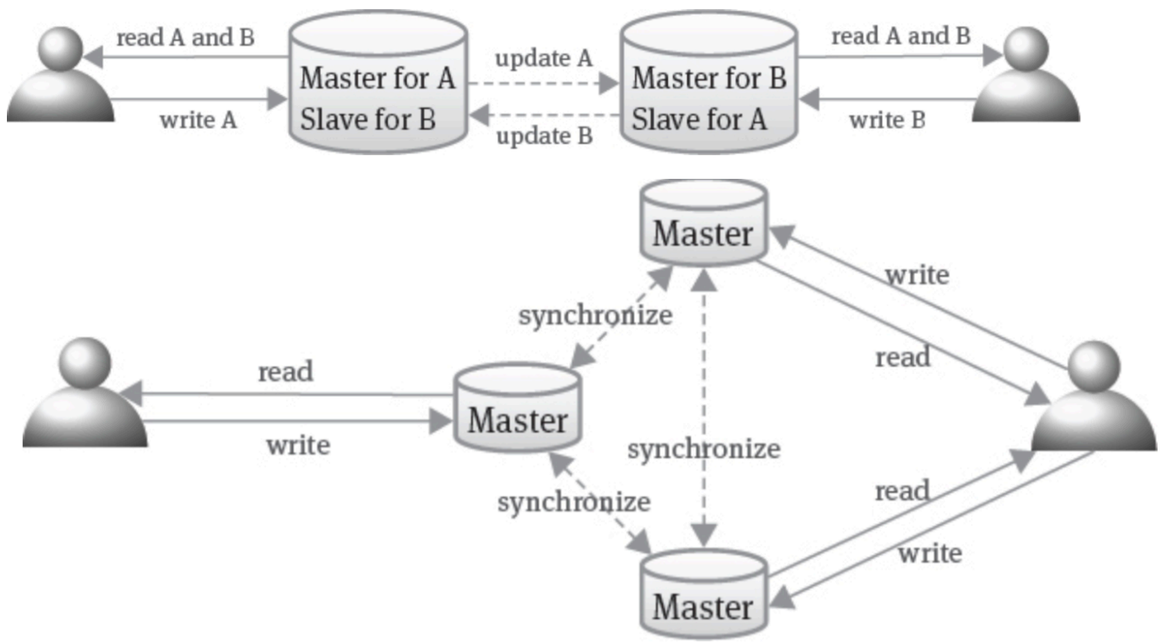
Bases de Datos Distribuidas

Modelos de replicas

- **Modelo maestro-esclavo**: El nodo maestro controla las operaciones de escritura.
    - Tener un único servidor maestro para todas las solicitudes de escritura en el sistema de la base de datos es un cuello de botella. Una solución pragmática es dividir el conjunto de todos los registros en particiones separados y asignar a cada partición un servidor maestro.

- **Modelo de multiples maestros**: Todos los servidores aceptan solicitudes de escritura y lectura para un elemento pero los servidores tienen que sincronizar regularmente su estado entre ellos.





Bases de Datos Distribuidas

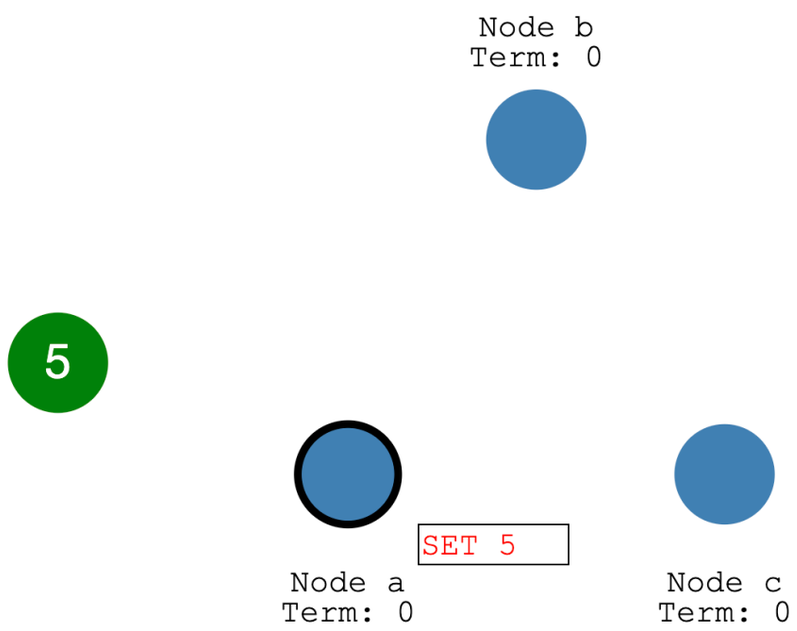
Control de concurrencia distribuido

- El control de concurrencia distribuido asegura la correcta ejecución de las operaciones (más generalmente, las transacciones) que afectan a los datos que se almacenan de forma distribuida en diferentes servidores de bases de datos.
    - En el caso de la replicación de datos, una aplicación típica de un protocolo de control de concurrencia es sincronizar todas las réplicas de un registro cuando se emite una escritura a uno de los servidores de bases de datos.

- Existen distintos protocolos, pero los mas usados son los protocolos de **consenso de quórum** que requieren una cierta mayoría de agentes para acordar un valor propuesto. La definición exacta de mayoría depende de el comportamiento de lectura y escritura y los tipos de fallas contra los que el protocolo debe ser resistente.

Bases datos distribuidas

El algoritmo RAFT



- http://thesecretlivesofdata.com/raft/

Consultas?

Consultas o comentarios?

Muchas gracias