Instituto Tecnológico de Costa Rica

Jesús Andrés Cortés Álvarez

Carné: 2021579439

Lectura 8/9

Explique la diferencia entre modelos de consistencia data-centric y client-centric

- **Data-centric:** Una característica importante es la coherencia analizada desde el punto de vista de las réplicas, en la perspectiva centrada en datos hay dos dimensiones: Modelos para especificar la consistencia que describen los modelos que permiten medir y especificar los niveles de consistencia que son tolerables para la aplicación. El otro modelo seria el modelo de ordenamiento de consistente de operaciones, que describen los modelos de consistencia que especifican que orden de operaciones se garantiza en las réplicas.
- **Client-centric:** Es la coherencia centrada en el cliente es la coherencia analizada desde el punto de vista de los clientes. Este modelo también presenta dos dimensiones, como lo son **eventual consistency** y **client consistency guarantees**, la dimensión de eventual consistency, establece que todas las réplicas se volverán consistentes gradualmente si no se produce ninguna operación de actualización y la dimensión de client consistency guarantees define que cada proceso del cliente debe garantizar cierto nivel de coherencia al acceder al valor de los datos en diferentes replicas.

Comente similitudes y diferencias entre los modelos de consistencia de Redis y Cassandra

- **Redis:** Es un almacén de estructuras de datos en memoria de código abierto, que se utiliza como base de datos, caché e intermediaria de mensajes, Admite estructuras de datos como cadenas, hashes, listas, conjuntos, mapas de bits, entre muchos otros. Redis optimiza los datos de la memoria priorizando el alto rendimiento, la complejidad computacional, la alta eficiencia del espacio de memoria y e bajo tráfico de red de aplicaciones. Redis garantiza alta disponibilidad ampliando su arquitectura. En una configuración de clúster, Redis eventualmente es consistente cuando el cliente lee nodos de réplica.

  Redis Cluster implementa etiquetas de hash, las etiquetas garantizan que se asignen dos claves en la misma ranura, parte de la clave tiene que ser una subcadena común entre dos claves y dentro de corchetes.

  La arquitectura de Redis implementa el modelo **master-slave** sin proxies, eso significa que la aplicación es redirigido al nodo que tiene los datos solicitados, cada nodo maestro tiene una ranura hash, así mismo esta ranura tiene de 1 a N replicas (Incluido el maestro y sus réplicas), cuando el nodo maestro recibe una solicitud por medio de una aplicación, la maneja y la propaga a de forma asincrónica cualquier cambio a sus réplicas.

  En la configuración de replicación asincrónica, si el nodo maestro muere antes de replicar y después de reconocer al cliente, los datos se pierden permanentemente, por lo tanto, Redis Cluster no puede garantizar la persistencia de escritura en todo momento.

- **Cassandra:** Es una base de datos NoSQL de columnas, Cassandra prioriza la alta disponibilidad cuando está sujeta a la partición de red, esta base de datos también en capaz de ajustarse para ser de consistencia fuerte cuando está sujeta a particiones de red.

  Cassandra describe los datos mediante columnas, un espacio de claves es el contenedor más externo de todo el conjunto de datos, correspondiente a todo la base de datos, una columna contiene un par de nombre-valor y una marca de tiempo. Esta marca de tiempo es necesaria al resolver conflictos de coherencia.

  Cassandra crece distribuyendo datos en un conjunto de nodos, designados como cluster, cada nodo capaz de responder a las solicitudes de los clientes, cada nodo trabaja en la solicitud de un cliente, se convierte en el coordinador de esa solicitud. La replicación es la estrategia que utiliza Cassandra para lograr un sistema de alta disponibilidad. También es importante mencionar que Cassandra se diseñó inicialmente para ser eventualmente consistente, de alta disponibilidad y baja latencia, su consistencia se puede ajustar para que coincida con los requisitos del cliente.

Anteriormente se logra observar las principales características de estos modelos de consistencia en las que se diferencian, además presentan algunas similitudes entre los dos modelos, como por ejemplo que ambas priorizan la alta disponibilidad, todo esto mediante sus métodos de replicación, que como se mencionó anteriormente utilizan diferentes métodos, ambos buscan la alta disponibilidad.

Comente como afecta el rendimiento y funcionamiento de una base de datos los siguientes modelos de consistencia:

- **Strong Consistency:** Una característica que podría afectar es el hecho de al realizar una operación de lectura de un objeto debe esperar hasta que se confirme la escritura antes de poder leer la nueva versión, a pesar de tener alta consistencia, compromete el escalamiento al disminuir disponibilidad y partición de red.

- **Weak Consistency:** Este modelo no garantiza un orden especifico de eventos, una operacion de lectura no garantiza la devolución del ultimo valor escrito, otra característica es que este modelo conduce a un sistema altamente escalable ya que no necesita involucrar a más de una réplica o un nodo de solicitud de cliente.

- **Eventual Consistency:** Las réplicas en este modelo convergen al mismo estado de los datos, la ventana de inconsistencia dependerá de los retrasos de la comunicación entre replicas y sus fuentes, la carga del sistema y la cantidad de réplicas involucradas.

- **Causal Consistency:** Si algún proceso de este modelo actualiza un objeto determinado , todos los procesos que reconocen la actualización de este objeto consideran el valor actualizado, fortalecer el modelo de consistencia eventual para que sea consistencia eventual para que sea consistencia casual disminuye la disponibilidad y propiedades de partición de red del sistema.

- **Read-your-writes Consistency:** Este modelo permite garantizar que una réplica esté al menos lo suficientemente actualizada para que una transacción especifica realice los cambios. La coherencia de escritura y lectura se logra cuando el sistema garantiza que una vez actualizado el registro, cualquier intento de leer el registro devolverá el valor actualizado,

- **Session Consistency:** Cuando un proceso realiza una solicitud al sistema de almacenamiento en el contexto de una sesión, seguirá un modelo de coherencia de lectura y escritura mientras exista esa

sesión, al utilizar coherencia todas las lecturas están actualizadas con las escrituras de esa sesión, pero las escrituras se pueden retrasar. Un dato importante es que los datos de otras sesiones van en orden correcto, pero garantiza que estén actualizadas.

- **Monotic Reads Consistency:** Las lecturas monótonas no se aplican a operaciones realizadas por diferentes procesos, solo lecturas del mismo proceso, las lecturas pueden estar totalmente disponibles, incluso durante una partición de red, donde todos los nodos pueden progresar.

- **Monotic Writes Consistency:** Una operación de escritura invocada por un proceso en un objeto determinado debe completarse antes de cualquier operación de escritura posterior en el mismo objeto por parte del mismo proceso, las escrituras monótonas no aplican a operaciones realizadas por diferentes procesos, solo lecturas del mismo proceso. Las lecturas monótonas pueden estar totalmente disponibles, incluso durante una partición de red.

Use sus propias palabras y lo discutido en clase acerca de arquitectura de bases de datos distribuidas.

- A lo largo del curso fue muy interesante observar cómo mejorar el rendimiento de las bases de datos, como por ejemplo trabajar con diferentes servidores, donde se aplican réplicas de datos según se haya definido, los temas como las votaciones para definir en qué lugar se va ir replicando, el tema de tener ubicaciones físicas de las bases de datos y como esto favorece el rendimiento, en el caso de que uno de los servidores caiga, se cuenta con la opción de tener otro disponible.