Mecanismo de actualización de réplicas: Esquema de propagación

En un Sistema de Base de Datos Distribuida, el mecanismo de control de replicación asegura la consistencia de datos entre copias existentes. Categorizan los mecanismos acorde a la forma en que las modificaciones se propagan y qué copia es actualizada primariamente.

La propagación de los cambios sobre la Base de Datos Distribuida puede hacerse dentro o fuera de los límites de una transacción. En el caso de **replicación *eager (ansiosa)*,** las actualizaciones ocurren como una unidad lógica de trabajo, mantiene todas las copias exactamente sincronizadas en todas las localidades. La utilización del esquema ***eager*** permite en todo momento que cualquier lectura sobre una copia del dato obtenga como resultado una información correcta.

La **propagación *lazy* (perezoso)** propaga las modificaciones de los datos fuera del alcance de la operación que los genera. El esquema ***lazy*** demora la propagación de los cambios hasta que la transacción haya finalizado, implementando la propagación de los cambios como un proceso que corre en segundo plano.

Esquema de propagación *Eager* o de actualización de replicación sincrónica

Un esquema *Eager* mantiene todas las copias exactamente sincronizadas en todas las localidades modificando todas las réplicas como parte de una transacción atómica. El esquema de replicación *eager* permite una ejecución que no presenta anomalías en la concurrencia.

La utilización del esquema *eager* permite en todo momento que cualquier lectura sobre una copia del dato obtenga como resultado una información correcta.

Ventajas y costos del esquema de propagación eager o de actualización de replicación sincrónica

- Debido a que todas las actualizaciones dentro del proceso de replicación ocurren como una unidad lógica de trabajo, las transacciones exhiben lo que comúnmente se llaman propiedades ACID, generando, de esta forma, una ejecución, sin inconsistencias ningún tipo.
- Consistencia fuerte. La actualización original se demora hasta que todas las copias se actualicen, originando una transacción global. Así, cuando una transacción comete, todos las copias tienen el mismo valor.

Desventajas surgidas con este tipo de esquemas

- Disminución de desempeño: debido a que se necesita sincronizar cada acceso a datos con las demás réplicas durante la ejecución de la transacción, lo convierten en un esquema muy caro.
- Mayores tiempos de respuesta: Esta sobrecarga de mensajes tiende a reducir el desempeño de actualización e incrementar los tiempos de respuestas transaccionales, este esquema no puede dar una respuesta al usuario hasta que la actualización haya sido cometida por completo.
- La mayor desventaja es que el número de operaciones por transacción aumenta con el grado de replicación de los datos (número de nodos).
- No son apropiados para ambientes móviles. Por un lado reducen el desempeño de las actualizaciones con sincronizaciones extras, y por el otro, en un ambiente móvil los nodos están normalmente desconectados. De aquí que una actualización sincronizada no sea válida para este tipo de ambientes.

Esquema de propagación Lazy o de actualización asíncrona

Un esquema de propagación lazy permite que, primeramente, un réplica sea actualizada por la transacción original. Las actualizaciones de las restantes se propagan asincrónicamente, generalmente con una transacción para cada sitio o localidad.

La actualización asincrónica de réplicas o esquema de propagación lazy proporciona las siguientes ventajas

- Mayor desempeño: el tamaño de las transacciones es reducido, solo se limitan a actualizar la copia primaria de datos y, por consiguiente, son menores los bloqueos necesarios. Se mejora, de esta forma, el desempeño global del sistema.
- Mayor facilidad de escalabilidad: el número de bloqueos necesarios es menor que los esquemas eager debido a que las transacciones son más cortas.
- Aptos para ambientes móviles: estos esquemas dan una respuesta inmediata al usuario, los cambios de la propagación son realizados luego; este tipo de respuesta es de gran importancia dada la proliferación de aplicaciones para usuarios móviles, donde una copia no está siempre conectada al resto del sistema y no tiene sentido esperar hasta que las actualizaciones se hayan cometido para permitir al usuario ver los cambios.

Desventajas surgidas con este tipo de esquemas:

- Consistencia débil: debido a que cada transacción ejecuta localmente e independientemente, proporciona una consistencia débil. Existirá siempre algún grado de retraso entre el tiempo de cometer la copia origen y el tiempo para disponerla en las demás réplicas, permitiendo que las copias puedan divergir y se produzcan posibles inconsistencias en los datos.
- Acceso a "datos viejos": surge como consecuencia de la consistencia débil. Existe la posibilidad que la propagación asincrónica pueda causar que una transacción de actualización obtenga valores viejos de algún dato, resultando una ejecución que genera un estado inconsistente de la base de datos distribuida.