Sistemas de Alta Disponibilidad en PostgreSQL

Los servidores de bases de datos pueden trabajar juntos para permitir que uno de ellos se haga cargo rápidamente del servicio, si el servidor principal falla, en lo que se conoce como **Alta Disponibilidad** (High Availability en inglés), o para permitir que varios servidores sirvan los mismos datos, consiguiendo así un **Equilibrio de Carga** (Load Balancing).

Pero no existe una solución única pues la sincronización no es sencilla debido a las escrituras. Una escritura debe propagarse a todos los servidores para que las futuras solicitudes de lectura a esos servidores devuelvan resultados consistentes.

Cada solución aborda el problema de una manera diferente atendiendo a una estrategia concreta. **Algunas estrategias son nativas de PostgreSQL** y forman parte de su núcleo y **otras son soluciones externas**, incluso disponemos de soluciones hardware.

La estrategia más común es permitir que sólo un servidor atienda las escrituras. Los servidores que pueden modificar datos se denominan maestros (master) o primarios (primary). Los servidores que rastrean los cambios en el maestro para mantenerse actualizados se denominan servidores esclavos (slave) o en espera (standby). Un servidor standby que no permite conexiones y cuya única función es la de servir de respaldo al primario cuando éste caiga se le conoce como warm standby, y uno que permite conexiones y atiende consultas de solo lectura se llama hot standby.

Algunas soluciones son síncronas, lo que significa que una transacción de modificación de datos no se considera comprometida hasta que todos los servidores hayan confirmado la transacción. Esto garantiza que un failover no perderá ningún dato y que todos los servidores en equilibrio de carga devolverán resultados consistentes sin importar qué servidor se consulte.

Por el contrario, las soluciones asíncronas permiten cierto retraso entre el momento del commit y su propagación a los otros servidores, lo que deja abierta la posibilidad de que algunas transacciones se pierdan en el camino y que los servidores con equilibrio de carga arrojen resultados ligeramente obsoletos.

Como siempre, la elección entre una estrategia síncrona o asíncrona dependerá del sistema del que se trate.

Si la comunicación síncrona, por las características particulares del sistema, resulta demasiado lenta habría que optar por la asíncrona.

Si se quiere priorizar la Alta Disponibilidad tendríamos que decantarnos por utilizar una estrategia asíncrona porque si hay problemas con el servidor en standby que le impidan replicar las transacciones, el servidor primario quedará bloqueado a la espera de que la standby confirme que se ha actualizado.

Por el contrario, **si quiere priorizar la disponibilidad de los datos, optaremos por una estrategia síncrona**. Imagine la base de datos de un banco. No sería válido que se pierda, por ejemplo, una transacción sobre una transferencia de dinero de la cuenta de un titular en la de otro. En este caso, en lugar de disponer del 100% de Disponibilidad, el requisito es 0% pérdida de datos por lo que, si la réplica no puede guardar la transferencia de dinero, ésta no se guardará tampoco en la principal, quedándose el sistema bloqueado hasta que en la réplica se aplique. El cliente entenderá más fácilmente que tenga que esperar un poco para hacer la transferencia, en cambio no entenderá que su transferencia se haya perdido ni valorará que a cambio tuvo un servicio impecable y sin interrupciones.

Las distintas soluciones también se pueden clasificar por su granularidad. Algunas soluciones se ocupan de replicar un servidor de bases de datos completo mientras que otras soluciones permitirán replicar a nivel de base de datos, de tabla, etc.

Una vez entendidos estos conceptos vamos a ver las diferentes soluciones:

Discos compartidos (Shared Disk Failover)

En este caso se utiliza una única matriz de discos que es compartida por varios servidores.

Como sólo tenemos una copia de la base de datos, no existe la sobrecarga producida para garantizar la sincronización. Si el servidor principal falla, un servidor en espera puede montar e iniciar la base de datos.

La ventaja principal es que este sistema permite una rápida conmutación por error (failover) sin pérdida de datos y, para el cliente, el precio de la conmutación es similar a la de un bloqueo en base de datos.

El inconveniente de este método es que si la matriz de discos compartidos falla, o se daña, ni el servidor principal ni los servidores en espera funcionarán.

Replicación del sistema de archivos (File System Replication)

En este caso, todos los cambios en un sistema de ficheros se reflejan en un sistema de ficheros que reside en otra máquina.

Aquí la única restricción es que la replicación debe realizarse de una manera que garantice que el servidor en standby realice las escrituras en el mismo orden que se hicieron en el maestro.

DRBD es una solución popular de replicación de sistemas de archivos para Linux.

Envío de WAL (WAL Shipping)

Los WAL son los registros de escritura anticipada (Write Ahead Log).

Los servidores standby se pueden mantener actualizados leyendo el flujo de WAL.

Si el servidor principal falla, el standby contiene caso todos los datos del servidor principal y se puede convertir rápidamente en el nuevo servidor maestro.

Puede ser síncrono o asíncrono y sólo se puede hacer para todo el clúster de base de datos.

El inconveniente de esta opción es que el fichero WAL no se replica en el servidor en standby hasta que no se llena y esto puede significar una pérdida de datos de hasta 16MB (el tamaño por defecto de los ficheros WAL).

Streaming Replication

Este sistema es como el anterior pero mejora el inconveniente que hemos comentado sobre la pérdida de datos, pues no espera a que el fichero WAL se llene para ser replicado en el servidor standby sino que lo hace transacción a transacción conforme estas llegan.

De la misma forma, puede ser síncrono o asíncrono y sólo se puede hacer para todo el servidor de bases de datos.

Logical Replication

La replicación lógica permite que un servidor de base de datos envíe un flujo de modificaciones de datos a otro servidor de manera que se permite replicar los cambios de tablas individuales.

La replicación lógica no requiere que un servidor en particular sea designado como maestro o esclavo, permite que los datos fluyan en múltiples direcciones.

Replicación maestro/esclavo basada en disparadores (Trigger-Based Master-Standby Replication)

Slony-I es un ejemplo de este tipo de replicación, con granularidad por tabla y soporte para múltiples servidores en standby que, de forma asíncrona, aplican los cambios que el servidor maestro envía.

Middleware de replicación basado en SQL (SQL-Based Replication Middleware)

Middleware es un software que se sitúa entre un S.O. y las aplicaciones que se ejecutan en él. Funciona, básicamente, como una capa de abstracción para permitir la comunicación y la administración de datos en aplicaciones distribuidas.

Con el middleware de replicación basado en SQL, un programa intercepta todas las consultas SQL y las envía a uno o todos los servidores. Cada servidor funciona de forma independiente.

Las consultas de lectura y escritura deben enviarse a todos los servidores de modo que cada servidor reciba los cambios. Pero las consultas de solo lectura se pueden enviar a un solo servidor, lo que permite que la carga de trabajo de lectura se distribuya entre ellos.

Pgpool-II es un buen ejemplo de este tipo de replicación.

Replicación Multimaestro (Multimaster Replication)

La replicación multimaestro puede ser síncrono o asíncrono.

En el modo asíncrono cada servidor funciona de forma independiente y se comunica periódicamente con los demás servidores para identificar las transacciones en conflicto. Los conflictos pueden ser resueltos por usuarios o por reglas de resolución de conflictos.

Bucardo es un ejemplo de este tipo de replicación.

En el modo síncrono cada servidor puede aceptar solicitudes de escritura y los datos modificados se transmiten desde el servidor original a todos los demás servidores antes de que se confirme cada transacción.

La actividad de escritura intensa puede provocar un bloqueo excesivo y retrasos en la confirmación, lo que conduce a un rendimiento deficiente.

Algunas implementaciones usan disco compartido para reducir la sobrecarga de comunicación.

Este tipo de replicación (síncrona con varios maestros) es más adecuada para cargas de trabajo que sean principalmente de lectura, aunque su gran ventaja es que cualquier servidor puede aceptar solicitudes de escritura.

Existen otras técnicas para balancear la carga que no podrían considerarse un sistema de réplica como tal, como son el particionado de datos y la ejecución de consultas en paralelo en múltiples servidores.

Además de todo lo que hemos visto hay que tener en cuenta que, como PostgreSQL es de código abierto y se puede ampliar fácilmente, varias empresas han adoptado PostgreSQL y han creado soluciones comerciales con capacidades conjuntas de conmutación por error, replicación y equilibro de carga.