Implementación de Hot-Standby y Stream Replication en PostgreSql

Juan Catalán Olmos
jcatalanolmos[at]gmail[dot]com

5 de agosto de 2013

1. Definiciones y Parámetros

El presente informe es a modo de introducción para la implementación de las características de *"Hot Standby (HS)"* y *"Stream Replication (SR)"* las cuales son dos características separadas pero que pueden funcionar de forma complementaria para una configuración también conocida como "Binary Replication". Ambas características totalmente disponibles desde la entrega de la versión de PostgreSql 9.0 en adelante.

En el presente informe se utilizarán dos máquinas con :

- Postgresql 9.1
- Ubuntu 12.04 LTS 32 bits

Si bien las pruebas finales se realizarán con distinto *hardware*, la implementación final se realizará con una cantidad mayor de similitudes en todo nivel (Hardware y Software) entre servidores, y sus resultados serán presentados como anexo al presente.

Ya que existen múltiples fuentes que hacen referencia a los procesos de instalación de estos dos componentes, se obviarán estos procesos.

1.1. Beneficios

Dentro de las características que posee esta configuración están:

- Poseer una simple, pero completa capacidad de replicar una base de datos en producción, concendiendo la posibilidad de una rápida reacción ante bajasdel servicio teniendo solo unos pocos segundos de perdida de datos, incluso bajo circunstancias catastróficas.
- Capacidad de balanceo de carga de lectura/escritura entre el servidor maestro y sus distintos esclavos [1].
- Ejecutar reportes u otro tipo de consultas que implequen un largo tiempo de ejecución o carga sobre un servidor esclavo, liberando de esa cargar y trabajo al servidor principal.
- Replicar todos los DDL, incluyendo tablas y cambios en los índices, e incluso la creación de nuevas bases de datos.
- Replicación de una base de datos con arquitectura *multi-tenant*, evitando la especificación de requerimientos para llaves primarias o cambios en la base de datos.

1.2. Restricciones

- Replicar una tabla, esquema o base de datos específica. La replicación binaria hace referencia a la instancia de PostgreSQL (o también llamado "Cluster")
- Replicación con arquitectura Multi-Master. Una replicación binaria del tipo Multi-Master según la documentación es tecnológicamente imposible.
- Replicar distintas versiones de PostgreSQL o entre distintas plataformas.
- Aún PostgreSQL no puede realizar replicación sin los permisos de administración requeridos en el servidor.
- Replicación de datos de forma sincrónica, garantizando llegar a evitar en su totalidad la perdida de datos, aunque esta característica ya está disponible.

[1] Un error común es el tratar de encontrar el valor presente en una secuencia en un servidor esclavo, esto de acuerdo a los documentos estudiados no es posible.

Debido a las razones anteriormente mencionadas, es que se pueden utilizar otras herramientas como Slony-I, Londiste, Bucardo, pgPool2 u otros[2] para poder satisfacer otras necesidades.

1.3. Manejo de Logs y ficheros WAL [7]

Cada uno de los servidores de PostgreSQL[3], posee un sistema para el registro de todas las transacciones que se realizan (en inglés el *"Transaction Log"*, el cuál está ubicado en directorio de datos dentro de la carpeta *PGDATA/pg_xlog* . Este registro consiste en *snapshots* binarios de la información escrita hacia registros sincronizados ante cada modificación de la información contenida en cada una de las bases de datos, en caso de que algún problema inesperado interrumpa el servicio, asegurando que la información no sea corrompida y las transacciones no completadas se pierdan.

También desde la versión de PostgreSQL 8.0, se pueden realizar copias de una base de datos y replicar los cambios realizados a esta base de datos maestra, conocido como *"Point in Time Recovery"* también conocido como *"Log Shipping"* o *Traspaso de Registros*.

Estos registros consisten en segmentos de datos de 16MB generados desde segmentos de páginas de 8K con nuevos datos[4].Debido a que son datos binarios, no es posible saber las modificaciones realizadas ni el estado anterior o posterior del servidor a través del análisis de estos registros.

Estos registros además son tratados por el sistema como buffers, siendo eliminados una vez que el sistema logró un estado seguro, siendo ya innecesarios para una recuperación post caída. Además, cada registro sólo puede ser aplicado a una base de datos *binary identical* la cuál generó el registro.

[2] Revisar links [5 | 6]

[3] Servidores hace referencia a la instancia de PostgreSQL, servicio o también llamado Cluster, el cual puede contener múltiples blases de datos

[4] No son los procedimientos SQL o *SQL Statements*

1.4. Tipos de Replicación

Para una configuración del tipo Maestro/Esclavo, se entenderá que los registros de transacción son generados por el Maestro y que el servidor secundario o esclavo es quién recibe estos registros para ser aplicados de acuerdo a la configuración descrita.

1.4.1. PITR y Warm Standby

Bajo una configuración PITR, los registros transaccionales son copiados y guardados hasta que estos son requeridos, que puede ser cuando un servidor que estaba en modo secundario se "activa o levanta" y se comienza el proceso de restauración a través de la utilización de estos. PITR es principalmente utilizado para realizar recuperaciones de bases de datos o análisis Forence sobre estas. Tambiés es utilizado para realizar respaldos incrementales sobre bases de datos masivas, teniendo ventajas ante pg_dump.

En Warm Standby, los registros son copiados desde el maestro al esclavo y aplicados inmediatamente estos son recibidos. Si bien el esclavo posee casi la misma información, ya que puede existir un pequenõ desfase de la información debido al traspaso y ejecución del registro, no es posible ejecutar consultas sobre este servidor, pero en caso de ser necesario, puede ser rápidamente llevado a un estado completamente operacional. Esta configuración requiere la caracteristica de Traspaso de Registros, y es comunmente utilizada como respaldo ante fallas.

1.4.2. Hot Standby [3]

Esta configuración es básicamente identica a Warm Standby, pero además se le agrega la ventaja que el esclavo además puede recibir carga a través de consultas de lectura, permitiendo distribuir carga a nivel de consultas de lectura.

"HS" solamente requiere cierto espacio en el maestro y posee, teóricamente, escalabilidad infinita. Un WAL[5] puede ser distribuido entre varios servidores a través de distintos métodos ya sea a través de NAS o por SFTP. Aún así, debido a que la replicacioón es realizada a través del traspaso de

[5]WAL: Write Ahead Log

registros de 16MB, el esclavo puede encontrarse a minutos de diferencia del principal, causando conflictos ya sea ante una recuperación o para el balance de cargas.

1.4.3. Stream Replication [4]

"SR" mejora todas las otras configuraciónes con la utilización de una conexión por red entre las bases de datos del Esclavo y el Maestro, en vez de mover los registros de 16MB. Esto permite que los cambios se pasen a través de la red de forma casi inmediata luego de haber sido completados en el Maestro, pero a diferencia de "HS", coloca cierto nivel carga sobre el Maestro y se debe poseer la capacidad de crear estas conexiones.

Bajo esta configuración, tanto en el Maestro como el Esclavo se inician procesos especiales llamados *"walsender"* y *"walreceiver"* los cuales son los encargados de transmitir las páginas de datos por la red, requiriendo una conexión con bastante movimiento por cada esclavo que se anexe al Maestro, generando una carga incremental de acorde aumenten los esclavos. Aún así la carga sigue siendo baja permitiendo anexar y soportar varios esclavos fácilmente.

Aún cuando esta configuración no requiere el traspaso de registros para una operación normal, puede ser necesario la realización de estos para iniciar la replicación o en caso de comenzar a retrasarse a nivel de datos el Esclavo con respecto al Maestro.

2. Implementación

Para esta configuración, es necesario cumplir por lo menos con los siguientes prerequisitos:

- 2 o más servidores con similares sistemas operativos, como Ubuntu 12.04 LTS 32 bits.
- La misma versión de PostgreSql 9.X en todos los servidores.
- Acceso a shell de PostgreSql como superusuario en PostgreSQL en todos los servidores.

- Conocimientos de como iniciar, detener y recargar las configuraciones en Postgresql.
- A lo menos PostgreSQL 9.0 corriendo en el servidor Principal.
- Una base de datos en el Servidor Primario.
- Acceso como usuario postgres o root en los servidores con acceso a la red.

Además es altamente recomendable manejar llaves ssh entre los usuarios postgres para poder acceder sin problemas entre ambos servidores.

2.1. Quick Start

Para la implementación de par de servidores de postgresql para realizar *"HR"* y *"SR"* es necesario realizar los siguientes pasos:

1. Instalar los requerimientos necesarios para la base de datos.
2. Inicializar la base de datos y realizar una copia binaria de la misma.
3. Configurar el servidor primario. Modificar postgresql.conf y pg_hba.conf.
4. Configurar el(los) Servidor(es) secundario(s). Modificar postgresql.conf y crear recovery.conf
5. Iniciar los servidores

2.2. Implementación Detallada

Se anexará una vez que se impelemente en su totalidad la configuración.

3. Bibliografía

Los siguientes son links que se recomienda revisar y desear conocer:

1. http://www.postgresql.org/docs/current/static/warm-standby.html
2. http://www.postgresql.org/docs/current/static/hot-standby.html
3. http://wiki.postgresql.org/wiki/Hot_Standby
4. http://wiki.postgresql.org/wiki/Streaming_Replication
5. http://stackoverflow.com/questions/12764981/which-built-in-postgres-replication-fits-my-django-based-use-case-best
6. http://wiki.postgresql.org/wiki/Replication,_Clustering,_and_Connection_Pooling
7. http://wiki.postgresql.org/wiki/Binary_Replication_Tutorial
8. http://www.postgresql.org/docs/current/static/warm-standby.html#SYNCHRONOUS-REPLICATION
9. http://www.postgresql.org/docs/9.2/static/runtime-config-replication.html