Replicación Postgresql 9.1 usando Slony-I 2.2

Objetivo

- Implementar mecanismo de replicación basado en Slony-I entre dos servidores PostgreSQL 9.1[1]

Requisitos

- Lectura y comprensión de los apuntes [2], [3]
- Comprender el concepto de log de transacciones, técnicas de backup y recuperación.
- Comprender tecnologías implementadas en PG tales como WAL y PITR.
- Que el alumno cuente con conocimientos básicos de sistemas operativos Unix / Linux: file system, network file system, shell scripts, etc.
- Que el alumno cuente con conocimientos básicos de ANSI C, herramientas make, procesos de compilación y linkedición de aplicaciones ANSI C.
- Contar con acceso administrativo a dos servidores PG 9.1 instalados sobre Linux Debian amd64 y funcionando, ambos servidores deben estar en el mismo entorno de red que les permita su comunicación. No obstante, este tutorial también puede aplicarse sobre el mismo servidor local.

Introducción

Slony-I se encuentra catalogada en [2] como una solución de replicación maestro-esclavo basada en triggers. Solo el servidor maestro podrá ejecutar queries de actualización, mientras que los servidores esclavos (standby) serán solo de consulta, podrán ejecutar queries de consulta y serán actualizados en forma asincrónica a través de “batches de actualización” que enviará el servidor maestro.

1. Instalación

-Asegurarse de tener instalados los siguientes paquetes de software:

```
$ aptitude install build-essential
$ aptitude install gcc
$ aptitude install gcc-doc
$ aptitude install perl
$ aptitude install make
$ aptitude install jade
$ aptitude install openjade
$ aptitude install flex
$ aptitude install libpq-dev  (instalar headers para libpq)
$ aptitude install postgresql-client
$ aptitude install postgresql-contrib-9.1
$ aptitude install postgresql-server-dev-9.1
```

-Descargar los fuentes y la documentación Slony-I de su pagina web:
http://slony.info/downloads/2.2/source/

en este caso se utilizó la versión 2.2 (docs y sources; también disponibles en http://www.grch.com.ar/docs/bdd/apuntes/unidad.iv/slony-I)

-Descomprimir los fuentes y la documentación bajada en un directorio, en dicho directorio, hacer con usuario root (y hasta lograr que compile y se instale sin errores) lo siguiente:

```
$ ./configure
$ make
$ make install
```

2. Algunos conceptos slony-I

cluster:conjunto de instancias de bases de datos postgresql a ser replicadas, cada script Slonik especifica su cluster con la instrucción:

```
cluster name = mireplica;
```

Slony-I creará en cada instancia de cada base de datos del cluster, el namespace/schema `_mireplica` (en este caso).

Nodo: base de datos PostgreSQL que esta participando en la replicación. Se define al comienzo del script Slonik, usando la directiva:

```
NODE 1 ADMIN CONNINFO = 'dbname=testdb host=server1
user=slony';
```

`CONNINFO` es la información de conexión que internamente utiliza la función `PQconnectdb()` de la libreria C libpq.
Cada nodo tiene su numero . Cada cluster se compone de un conjunto de nodos.

Replication set (conjunto de replicación): conjunto de tablas a ser replicadas entre los distintos nodos del cluster. Se pueden definir distintos conjuntos de replicación y el flujo de replicación puede ser distinto entre los distintos conjuntos.

Origen,proveedores,suscriptores: cada replicación tiene un nodo origen, en ese nodo origen es el único lugar en donde se permiten cambios sobre los datos, es el "master provider" (proveedor maestro), nodo maestro que provee los datos. Los otros nodos son suscriptores, se suscriben al conjunto de replicación, indicando que desean recibir datos (esclavos). El nodo origen nunca puede considerarse un suscriptor. Se puede hacer una replicación en cascada en Slony-I, donde un suscriptor puede luego ser proveedor de otro suscriptor.

Demonio slon: proceso escrito en C que administra la actividad de replicación en cada nodo.
Slon procesa eventos de dos tipos:

-eventos de configuración:

provocados por la ejecución del script slonik, el cual envía los cambios de configuración del cluster.

-eventos SYNC (de sincronización):

las actualizaciones de las tablas que son replicadas se agrupan en SYNC's; estos conjuntos de transacciones son aplicadas a los nodos suscriptores de la replicación.

Procesador de configuración slonik: procesa scripts escritos en "little language" que son usados para enviar eventos de actualización de configuración al cluster. Por ejemplo: agregar o quitar nodos ,cambiar paths de comunicación, agregar o quitar suscriptores, etc.

Paths de comunicación: Slony-I usa PostgreSQL DSNs en 3 contextos para establecer acceso a las bases de datos:

-SLONIK ADMIN CONNINFO permite controlar cómo un script slonik puede acceder a los nodos del cluster.

-el parámetro DSN de slon, le indica al proceso slon el path a utilizar hacia la base de datos que administra.

-SLONIK STORE PATH que controla como los demonios slon se comunican con los nodos remotos. Estos paths son almacenados en sl_path.

Se debe contar con un path de comunicación entre cada nodo suscriptor y su proveedor.

SSH Tunneling (tunel ssh): Si no se puede establecer una conexión directa a un nodo debido a la existencia de un firewall intermedio, se puede establecer un tunel ssh y hacer que Slony-I opere sobre el mismo.

3. Limitaciones Slony-I

Slony-I no replica:

-BLOB'S

-DDL's

-usuarios y roles

debido a que Slony-I replica a través de triggers y lo anterior no puede ser capturado por eventos en triggers.

Los comandos DDL's pueden transmitirse a través del script slonik, utilizando el comando SLONIK EXECUTE SCRIPT, en forma manual, pasando a cada nodo, cada uno de los cambios realizados.

4. Replicando la primer Base de Datos

Ejemplo de replicación sobre el mismo servidor (localhost) con 2 bases de datos.

Archivo de configuración servidor: /etc/postgresql/9.1/main/postgresql.conf
Archivo de configuración cliente: /etc/postgresql/9.1/main/pg_hba.conf

-Hacer backup de los archivos de configuración servidor y cliente
-Crear las siguientes variables de entorno en bash, en sesión linux, con usuario postgres (administrador PG) hacer:

```
$ export CLUSTERNAME=slony_example
$ export MASTERDBNAME=pgbench
$ export SLAVEDBNAME=pgbenchslave
$ export MASTERHOST=localhost
$ export SLAVEHOST=localhost
$ export REPLICATIONUSER=pgsql
$ export PGBENCHUSER=pgbench
```

-Comprobar las variables:

```
$ echo $CLUSTERNAME $MASTERDBNAME $SLAVEDBNAME $SLAVEHOST $REPLICATIONUSER
$PGBENCHUSER
```

-Crear el usuario pgbench para usarlo en la replicación, siempre con usuario postgres hacer:

```
$ createuser -SRD $PGBENCHUSER
```

-Poner clave a usuario postgres pgbench, con usuario postgres hacer:

```
$ psql
postgres=# ALTER ROLE pgbench WITH ENCRYPTED PASSWORD 'pgsql';
postgres=# \q

$ createuser -SRD $REPLICATIONUSER
postgres=# ALTER ROLE pgsql WITH ENCRYPTED PASSWORD 'pgsql';
postgres=# \q
```

-Preparar las bases de datos, con usuario postgres hacer:

```
$ createdb -O $PGBENCHUSER -h $MASTERHOST $MASTERDBNAME
(usar clave de usuario pgbench pgsql)
```

```
$ createdb -O $PGBENCHUSER -h $SLAVEHOST $SLAVEDBNAME
(usar clave de usuario pgbench pgsql)
```

-Se requiere contar con un path de ejecución a /usr/lib/postgresql/9.1/bin/ , en donde hay herramientas de PG:

```
$ export PATH=$PATH:/usr/lib/postgresql/9.1/bin
```

(Otra opción es poner todas estas instrucciones en un script bash, grabarlo, copiarlo al home del usuario postgres, para luego poder ejecutarlo las veces que quiera, con usuario root, hacer:

```
$ cp ./UNLu/11078/slony-I.sh /var/lib/postgresql/slony-I.sh
$ chmod +x /var/lib/postgresql/slony-I.sh
```

asumiendo que el script se llama ./UNLu/11078/slony-I.sh
)

-Con usuario postgres, ejecutar el comando PG pgbench :

```
$ pgbench -i -s 1 -U $PGBENCHUSER -h $MASTERHOST $MASTERDBNAME
---------------------------------------
postgres@debian2:~$  pgbench  -i  -s  1  -U  $PGBENCHUSER  -h  $MASTERHOST
$MASTERDBNAME
Password:
NOTICE:  la tabla «pgbench_branches» no existe, ignorando
NOTICE:  la tabla «pgbench_tellers» no existe, ignorando
NOTICE:  la tabla «pgbench_accounts» no existe, ignorando
NOTICE:  la tabla «pgbench_history» no existe, ignorando
creating tables...
10000 tuples done.
20000 tuples done.
30000 tuples done.
40000 tuples done.
50000 tuples done.
60000 tuples done.
70000 tuples done.
80000 tuples done.
90000 tuples done.
100000 tuples done.
set primary key...
NOTICE:    ALTER  TABLE  /  ADD  PRIMARY  KEY  creará  el  índice  implícito
«pgbench_branches_pkey» para la tabla «pgbench_branches»
NOTICE:    ALTER  TABLE  /  ADD  PRIMARY  KEY  creará  el  índice  implícito
«pgbench_tellers_pkey» para la tabla «pgbench_tellers»
NOTICE:    ALTER  TABLE  /  ADD  PRIMARY  KEY  creará  el  índice  implícito
«pgbench_accounts_pkey» para la tabla «pgbench_accounts»
vacuum...done.
postgres@debian2:~$
---------------------------------------------
```

(entre guiones se encuentra la salida del comando pgbench)

-La tabla pgbench_history no tiene clave primaria y Slony-I requiere que tenga clave primaria, para crear su clave primaria, hacer:

```
$ psql -U $PGBENCHUSER -h $MASTERHOSTHOST -d $MASTERDBNAME
(poner clave a usuario pgbench pgsql)
```

```
pgbench=>begin;
pgbench=>alter table pgbench_history add column id serial;
pgbench=>update pgbench_history set id = nextval('pgbench_history_id_seq');
pgbench=>alter table pgbench_history add primary key(id);
pgbench=>commit;
pgbench=>\q
```

-Slony-I requiere que este instalado el lenguaje pl/pgSql (puede que ya este instalado), para ello, ejecutar lo siguiente con usuario postgres:

```
$ createlang -h $MASTERHOST plpgsql $MASTERDBNAME
```

-Slony-I no copia automáticamente el contenido de toda la tabla cuando un esclavo se suscribe, sino que ello debe hacerse la primera vez en forma manual, utilizando la herramienta PG pg_dump, con usuario postgres hacer:

```
$ pg_dump -s -U $PGBENCHUSER -h $MASTERHOST $MASTERDBNAME > dump.sql
(usar clave de usuario pgbench pgsql)
$ psql -U $PGBENCHUSER -h $SLAVEHOST  -f dump.sql $SLAVEDBNAME
(usar clave de usuario pgbench pgsql)
$ rm dump.sql
```

-Arrancamos la replicación en forma manual, en una terminal:
para 5 conexiones concurrentes, cada una procesando 1000 transacciones contra la base de datos pgbench en el servidor localhost con usuario pgbench , con usuario postgres hacer:

```
$ pgbench -s 1 -c 5 -t 1000 -U $PGBENCHUSER -h $MASTERHOST $MASTERDBNAME
---------------------------------------
Scale option ignored, using pgbench_branches table count = 1
starting vacuum...end.
transaction type: TPC-B (sort of)
scaling factor: 1
query mode: simple
number of clients: 5
number of threads: 1
number of transactions per client: 1000
number of transactions actually processed: 5000/5000
tps = 67.385096 (including connections establishing)
tps = 67.511762 (excluding connections establishing)
---------------------------------------
```

(entre guiones se encuentra la salida del comando pgbench)

-Configurando la base de datos para replicación, usando script slonik:
crear el siguiente archivo de texto y guardarlo como `slonik-conf.txt` (el contenido del mismo se encuentra entre guiones, no incluir los guiones):

```
----------------------------------------
cluster name = cluster1;
```

```
node 1 admin conninfo = 'dbname=pgbench host=localhost user=postgres
password=pgsql';
node 2 admin conninfo = 'dbname=pgbenchslave host=localhost user=postgres
password=pgsql';

init cluster ( id=1, comment = 'Master Node');

create set (id=1, origin=1, comment='All pgbench tables');

set add table (set id=1, origin=1, id=1, fully qualified name =
'public.pgbench_accounts', comment='accounts table');
set add table (set id=1, origin=1, id=2, fully qualified name =
'public.pgbench_branches', comment='branches table');
set add table (set id=1, origin=1, id=3, fully qualified name =
'public.pgbench_tellers', comment='tellers table');
set add table (set id=1, origin=1, id=4, fully qualified name =
'public.pgbench_history', comment='history table');

store node (id=2, comment = 'Slave node', event node=1);
store path (server = 1, client = 2, conninfo='dbname=pgbench host=localhost
user=postgres password=pgsql');
store path (server = 2, client = 1, conninfo='dbname=pgbenchslave
host=localhost user=postgres password=pgsql');
-----------------------------------------
```

-Ejecutar el script slonik, con usuario postgres, hacer:

```
$ slonik slonik-conf.txt
```

-Hasta aquí ya tenemos las dos bases de datos con el cluster armado, la base de datos pgbench es el master y la base de datos pgbenchslave es la base de datos esclava; ambas dentro del servidor PG localhost.

-Ahora debemos correr los demonios slon (se ejecuta uno por cada nodo en la replicación, en este caso, se trata de 2 nodos en un mismo servidor local), con usuario postgres, hacer :

```
$ slon cluster1 "dbname=pgbench user=postgres host=localhost
password=pgsql"
```

-Abrimos otra terminal con usuario postgres y ejecutamos el comando para la replicación de la base de datos esclava (*setear* path en la nueva terminal), con usuario postgres, hacer:

```
$ slon cluster1 "dbname=pgbenchslave user=postgres host=localhost
password=pgsql"
```

-Aun no se están replicando datos, se han sincronizado las configuraciones. Para sincronizar datos, abrir otra terminar con usuario postgres, *setear* PATH, crear el siguiente archivo de texto para slonik (no incluir guiones), siempre con usuario postgres, hacer:

```
-----------------------------------------
cluster name = cluster1;
node 1 admin conninfo = 'dbname=pgbench host=localhost user=postgres
password=pgsql';
node 2 admin conninfo = 'dbname=pgbenchslave host=localhost user=postgres
password=pgsql';
subscribe set ( id = 1, provider = 1, receiver = 2, forward = no);
-----------------------------------------
```

guardar archivo como `slonik.txt` y ejecutar slonik, siempre con usuario postgres:

```
$ slonik slonik.txt
```

-Luego de esto, el demonio slon en servidor esclavo copiará el contenido de las tablas de pgbench a pgbenchslave.
-Ya están replicadas las bases de datos, la base de datos esclava debería tener la misma información que la base de datos maestro.
-El siguiente bash script compara el contenido de ambas bases de datos, para verificar si el contenido de las mismas es idéntico (no incluir los guiones):

```
-------------------------------------------------
#!/bin/sh
echo -n "**** comparing sample1 ... "
psql -U postgres -h localhost pgbench >dump.tmp.1.$$ <<_EOF_
   select 'accounts:'::text, aid, bid, abalance, filler
   from pgbench_accounts order by aid;
   select 'branches:'::text, bid, bbalance, filler
   from pgbench_branches order by bid;
   select 'tellers:'::text, tid, bid, tbalance, filler
   from pgbench_tellers order by tid;
   select 'history:'::text, tid, bid, aid, delta, mtime, filler, id
   from pgbench_history order by id;
_EOF_
psql -U postgres -h localhost pgbenchslave >dump.tmp.2.$$ <<_EOF_
   select 'accounts:'::text, aid, bid, abalance, filler
   from pgbench_accounts order by aid;
   select 'branches:'::text, bid, bbalance, filler
   from pgbench_branches order by bid;
   select 'tellers:'::text, tid, bid, tbalance, filler
   from pgbench_tellers order by tid;
   select 'history:'::text, tid, bid, aid, delta, mtime, filler, id
   from pgbench_history order by id;
_EOF_
if diff dump.tmp.1.$$ dump.tmp.2.$$ >cluster1.diff ; then
   echo "Ok ambas bases de datos son iguales!"
   rm dump.tmp.?.$$
   rm cluster1.diff
else
   echo "ERROR - ver cluster1.diff para ver las diferencias"
fi
-------------------------------------------------
```

-Grabe y ejecute el script anterior para comprobar que ambas bases de datos son iguales.
-Ejecute un INSERT INTO... sobre cualquier tabla y compruebe que el cambio se replica en la base de datos esclava.
-Copie el comando INSERT INTO … anterior que ejecutó sobre la base de datos maestra y cópielo en la esclava y ejecútelo, obtendrá un mensaje de error como este:

```
ERROR:  Slony-I: Table pgbench_accounts is replicated and cannot be
modified on a subscriber node - role=0


********** Error **********

ERROR: Slony-I: Table pgbench_accounts is replicated and cannot be modified
on a subscriber node - role=0
SQL state: XX000
```

ya que solo la base de datos maestra puede ser actualizada.

5. Observaciones

-¿Cómo vuelvo a poner en marcha la replicación? se inician los servidores y se ejecutan los demonios slon en cada nodo y listo!. No es necesario volver a ejecutar script slonik.

-Se hicieron cambios y adaptaciones a lo indicado en [1] en este tutorial para lograr un correcto funcionamiento y fue probado y testeado en Linux Debian amd64.

Referencias

[1] Manual Documentación Slony 2.2.3, disponible en http://slony.info
[2] Cherencio, G., “Soluciones de Replicación en PostgreSQL 9.1”, apunte de asignatura 11078 Base de Datos II, UNLu, disponible en http://www.grch.com.ar/docs/bdd/apuntes/unidad.iii/11078-Soluciones%20de%20replicación.pdf
[3] Cherencio, G. “Procedimientos habituales y tips en PostgreSQL 9.1”, apunte de asignatura 11078 Base de Datos II, UNLu, disponible en http://www.grch.com.ar/docs/bdd/apuntes/unidad.iii/11078-Procedimientos%20y%20tips.pdf

Atte. Guillermo Cherencio
11078 Base de Datos II
11077 Base de Datos I
División Sistemas
Departamento de Ciencias Básicas
UNLu