POSTGRESQL的SLONY-I同步复制实践

by [root](http://www.cnphp6.com/archives/author/root) ⋅ [Leave a Comment](http://www.cnphp6.com/archives/111087#comments)

官方文档：

slony-I

系统要求：[http://slony.info/documentation/requirements.html](http://slony.info/documentation/requirements.html" \t "_self)

配置与安装：[http://slony.info/documentation/administration.html#INSTALLATION](http://slony.info/documentation/administration.html" \l "INSTALLATION" \t "_self)

使用配置：[http://slony.info/documentation/tutorial.html#FIRSTDB](http://slony.info/documentation/tutorial.html" \l "FIRSTDB" \t "_self)

pgbench使用：[http://www.postgresql.org/docs/9.2/static/pgbench.html](http://www.postgresql.org/docs/9.2/static/pgbench.html" \t "_self)

环境：

主库：centos linux 32bit虚拟机，ip为192.168.13.128

            PostgreSQL9.2.13

            Slony-I 2.2

备库： centos linux 32bit虚拟机， ip为192.168.100.236

           PostgreSQL 9.4.

本文主要以主库配置过程为主，备库假设已安装完毕并可远程连接，仅在涉及备库的时候做相应的后续操作即可。

1.源码编译安装PostgreSQL

主库源码编译安装PostgbreSQL9.2。

注意：PG版本要必须是Slony支持的版本，详见 <http://slony.info/documentation/requirements.html> 。

安装目录：/opt/pgsql44

2. PostgreSQL源码安装pgbench

在postgresql源码安装包下，执行配置安装：

cd /contrib

make

make install

 即可将pgbench安装到/opt/pgsql944/bin下。

3.源码编译安装Slony-I

slony下载地址：[http://slony.info/downloads/2.2/source/](http://slony.info/downloads/2.2/source/" \t "_self)

下载并解压slony-2.2，然后进行编译配置安装，要确保每一步安装正确后再进行下一步。

cd ../slony-2.2

export PGMAIN=/opt/pgsql944

./configure --with-pgconfigdir=/opt/pgsql944/bin --with-perltools

gmake all

gamke install

4.设置slony复制的主、备库的环境变量

  postgres用户下执行：

export CLUSTERNAME=slony\_example

export MASTERDBNAME=masterdb

export SLAVEDBNAME=slavedb

export MASTERHOST=192.168.13.128

export SLAVEHOST=192.168.100.234

export REPLICATIONUSER=postgres

export PGBENCHUSER=pgbench

修改MASTERHOST和SLAVEHOST时，尽量不要使用localhost，因为可能会导致错误：ERROR remoteListenThread\_1: db\_getLocalNodeId() returned 2 – wrong database?。

5.创建 第4步设置的环境变量 参数

5.1根据环境变量创建相应对象

根据环境变量来创建相应对象

--在主服务器中

cd /opt/pgsql2/bin

./createuser -SRD $PGBENCHUSER  --若已创建则无需再创建

./createdb -O $PGBENCHUSER -h $MASTERHOST $MASTERDBNAME

--在备用服务器中

cd /opt/pgsql2/bin

./createuser -SRD $PGBENCHUSER  --若已创建则无需再创建

./createdb -O $PGBENCHUSER -h $SLAVEHOST $SLAVEDBNAME

--在主服务器

./pgbench -i -s 1 -U $PGBENCHUSER -h $MASTERHOST $MASTERDBNAME

--在主服务器查看，已产生如下数据表

[postgres@localhost bin]$ ./psql -U $PGBENCHUSER -h $MASTERHOST -d $MASTERDBNAME

psql (9.2.13)

Type "help" for help..

masterdb=# \d

                   List of relations

 Schema |          Name          |   Type   |  Owner

--------+------------------------+----------+----------

 public | pgbench\_accounts       | table    | postgres

 public | pgbench\_branches       | table    | postgres

 public | pgbench\_history        | table    | postgres

 public | pgbench\_history\_id\_seq | sequence | postgres

 public | pgbench\_tellers        | table    | postgres

(5 rows)

    以上产生的表pgbench\_history没有主键， Slony-I需要有一个合适的主键，所以接下来为pgbench\_history创建一个恰当的主键:

cd bin

./psql -U $PGBENCHUSER -h $MASTERHOST -d $MASTERDBNAME -c "begin; alter table

pgbench\_history add column id serial; update pgbench\_history set id =

nextval('pgbench\_history\_id\_seq'); alter table pgbench\_history add primary key(id);

commit;"

--查看此时的pgbench\_history的表结构

./psql -U $PGBENCHUSER -h $MASTERHOST -d $MASTERDBNAME

psql (9.2.13)

Type "help" for help.

masterdb=# \d pgbench\_history

                                   Table "public.pgbench\_history"

 Column |            Type             |                          Modifiers

--------+-----------------------------+-----------------------------------------

---------------------

 tid    | integer                     |

 bid    | integer                     |

 aid    | integer                     |

 delta  | integer                     |

 mtime  | timestamp without time zone |

 filler | character(22)               |

 id     | integer                     | not null default nextval('pgbench\_histor

y\_id\_seq'::regclass)

Indexes:

    "pgbench\_history\_pkey" PRIMARY KEY, btree (id)

5.2创建pl/pgsql过程语言

     Slony-I 需要数据库有 pl/pgSQL 过程语言，如果模板数据 template1已经安装了pl/pgSQL，那么新建的$MASTERDBNAME也就也有了pl/pgSQL，如果已经存在了，则不需要执行以下语句：

--在bin目录下

./createlang -h $MASTERHOST plpgsql $MASTERDBNAME

5.3 从主库将表定义倒入备库

   当备库Slony-I 订阅主库之后，Slony-I不会自动从主库拷贝表定义到备库，所以我们需要把表定义从主库导入备库，我们通过 pg\_dump来实现表定义的主、备同步：

--把在pgbench数据库建好的表同步到pgbenchslave数据库中：

--在主服务器中执行dump，然后将执行结果拷贝到备用服务器中执行，即可实现同步

--在bin目录下

./pg\_dump -s -U $REPLICATIONUSER -h $MASTERHOST $MASTERDBNAME

**5.4  运行pgbench并生成测试数据**

  为了演示Slony-I允许动态复制订阅,我们需要启动pgbench。如果你运行pgbench应用程序在一个单独的终端窗口的前台,你可以用不同的参数来停止并重新启动它。You’ll need to re-export the variables again so they are available in this session as well.

 运行pgbench的典型命令是像这样的:

--在bin目录下

./pgbench -s 1 -c 5 -t 1000 -U $PGBENCHUSER -h $MASTERHOST $MASTERDBNAME

 以上将会以PGBENCHUSER的用户身份，对MASTERHOST主机上的MASTERDBNAME这个数据库，运行5个数据库的并发客户端，每个执行1000个事务。

--查看pgbench\_account的数据行数

./psql -U $PGBENCHUSER -h $MASTERHOST -d $MASTERDBNAME

psql (9.2.13)

Type "help" for help.

masterdb=# select count(\*) from pgbench\_accounts;

 count

--------

 100000

(1 row)

6. 配置slony

**6.1官方文档理论介绍：**

**Configuring the Database For Replication.**

  Creating the configuration tables, stored procedures, triggers and configuration is all done through the [slonik](http://slony.info/documentation/slonik.html) tool. It is a specialized scripting aid that mostly calls stored procedures in the master/slave (node) databases.

  The example that follows uses [slonik](http://slony.info/documentation/slonik.html) directly (or embedded directly into scripts). This is not necessarily the most pleasant way to get started; there exist tools for building [slonik](http://slony.info/documentation/slonik.html) scripts under the tools directory, including:

* [Section 6.1.1](http://slony.info/documentation/additionalutils.html#ALTPERL) – a set of Perl scripts that build [slonik](http://slony.info/documentation/slonik.html) scripts based on a single slon\_tools.conf file.
* [Section 6.1.2](http://slony.info/documentation/additionalutils.html#MKSLONCONF) – a shell script (*e.g.* – works with Bash) which, based either on self-contained configuration or on shell environment variables, generates a set of [slonik](http://slony.info/documentation/slonik.html) scripts to configure a whole cluster.

方法一：**Using slonik Command Directly**

   此处省略详细介绍，详见文档[http://slony.info/documentation/tutorial.html#FIRSTDB](http://slony.info/documentation/tutorial.html" \l "FIRSTDB" \t "_self)。

**方法二：Using the altperl Scripts**

**Using the**[Section 6.1.1](http://slony.info/documentation/additionalutils.html#ALTPERL)**scripts is an alternative way to get started; it allows you to avoid writing slonik scripts, at least for some of the simple ways of configuring Slony-I. The**slonik\_build\_env**script will generate output providing details you need to build a**slon\_tools.conf**, which is required by these scripts. An example**slon\_tools.conf**is provided in the distribution to get you started. The altperl scripts all reference this central configuration file centralize cluster configuration information. Once slon\_tools.conf has been created, you can proceed as follows。**

# Initialize cluster:

$ slonik\_init\_cluster  | slonik

# Start slon  (here 1 and 2 are node numbers)

$ slon\_start 1

$ slon\_start 2

# Create Sets (here 1 is a set number)

$ slonik\_create\_set 1 | slonik

# subscribe set to second node (1= set ID, 2= node ID)

$ slonik\_subscribe\_set 1 2 | slonik

**You have now replicated your first database.**

**6.2具体配置步骤（根据以上的方法二）：**

 默认情况下，slony的配置文件样本已安装/usr/local/etc/slon\_tools.conf-sample。

cd /usr/local/etc

--复制一个slon\_tools.conf-sample，命名为slon\_tools.conf

cp slon\_tools.conf-sample /usr/local/etc/slon\_tools.conf

--开始编辑slon\_tools.conf

vi slon\_tools.conf

   配置详情：

    #修改CLUSTER\_NAME为前面我们设置的

    # The name of the replication cluster.  This will be used to

    # create a schema named \_$CLUSTER\_NAME in the database which will

    # contain Slony-related data.

    $CLUSTER\_NAME = 'slony\_example';

    #修改节点信息为我们的主、备节点的信息，样本中提供的多出的节点可以注释或者删除掉。

    # Include add\_node lines for each node in the cluster.  Be sure to

    # use host names that will resolve properly on all nodes

    # (i.e. only use 'localhost' if all nodes are on the same host).

    # Also, note that the user must be a superuser account.

    add\_node(node     => 1,

             host     => '162.168.13.128',

             dbname   => 'masterdb',

             port     => 5433,

             user     => 'postgres',

             password => 'postgres');

    add\_node(node     => 2,

             host     => '192.168.100.236',

             dbname   => 'slavedb',

             port     => 5433,

             user     => 'postgres',

             password => 'postgres');

     #修改要同步的表或者序列

     # This array contains a list of tables that already have primary keys.

        "pkeyedtables" => [

                           'pgbench\_accounts',

                           'pgbench\_branches',

                           'pgbench\_history',

                           'pgbench\_tellers',

                           ],

    #将序列和 无主键表同步的配置示例，注释掉。

    # For tables that have unique not null keys, but no primary

    # key, enter their names and indexes here.

#lyy comment this

#       "keyedtables" => {

#           'table3' => 'index\_on\_table3',

#           'table4' => 'index\_on\_table4',

#       },

    # Sequences that need to be replicated should be entered here.

# lyy comment this

#       "sequences" => ['sequence1',

#                       'sequence2',

#                       ],

#    },

7.启动并运行slony复制

7.1启动slon并复制：

#以下操作在任意目录下执行均可，用户是root

# Initialize cluster:

slonik\_init\_cluster

# 启动slon  (此处 1 和 2 是 node 节点号)

slon\_start 1

slon\_start 2

# 创建集合 (此处 1 是一个 set 集合号)

slonik\_create\_set 1

# 订阅 集合1 到 节点2 (1= set ID, 2= node ID)

slonik\_subscribe\_set 1 2

    目前，第一个数据库已经复制完毕。

7.2到备库验证复制情况

    现在，pgbench数据库中的数据就同步到pgbenchslave数据库中去了，这时去pgbenchslave中去检查pgbench\_accounts表中的数据数。

[postgres@localhost bin]$ ./psql  -p 5433 -h $SLAVEHOST -d $SLAVEDBNAME -U $PGBENCHUSER

psql (9.4.0)

Type "help" for help.

slavedb=#  select count(\*) from pgbench\_accounts;

 100000

   此后主库对pgbench\_accounts表中的数据进行增删后，备库都将作出相应的增删，这说明主备库已经实现了数据的同步复制。

8.Slony-I的其他操作

    slony的switchover操作（也就是把主节点改成从节点，从节点升级为主节点）：

slonik\_move\_set set1 node1 node2 | slonik

     slony的failver操作:

slonik\_failover node1 node2 | slonik

参考资料：<http://blog.chinaunix.net/uid-15145533-id-2775796.html>

Posted in: [数据库](http://www.cnphp6.com/archives/category/%e6%95%b0%e6%8d%ae%e5%ba%93)