pgbouncer工具安装使用

* **安装libevent**

libevent下载地址： <http://libevent.org/>

安装：

# wget https://github.com/libevent/libevent/releases/download/release-2.1.11-stable/libevent-2.1.11-stable.tar.gz

# tar -zxvf libevent-2.1.11-stable.tar.gz

# cd libevent-2.1.11-stable

# ./configure --prefix=/usr/libevent/libevent

# make

# make install

* **安装pgbouncer**

pgbouncer下载地址：

<https://pgbouncer.github.io/downloads/>

wget http://www.pgbouncer.org/downloads/files/1.11.0/pgbouncer-1.11.0.tar.gz

# tar -zxvf pgbouncer-1.11.0.tar.gz

# cd pgbouncer-1.11.0

# ./configure --prefix=/usr/pgbouncer/ --with-libevent=/usr/libevent/libevent

# make

# make install

* **配置pgbouncer**

配置 这两个文件：pgbouncer.ini 和users.txt文件

cd /usr/pgbouncer

mkdir etc log

cp /usr/pgbouncer/share/doc/pgbouncer/pgbouncer.ini /usr/pgbouncer/etc/

cp /usr/pgbouncer/share/doc/pgbouncer/userlist.txt /usr/pgbouncer/etc/

vi /usr/pgbouncer/etc/pgbouncer.ini

[databases]

pg\_benchmarksql=host=192.168.100.124 port=5432 user=benchmarksql dbname=benchmarksql pool\_size=100

[pgbouncer]

logfile = /usr/pgbouncer/log/pgbouncer.log

pidfile = /usr/pgbouncer/pgbouncer.pid

admin\_users = pgbadmin

listen\_addr = 0.0.0.0

listen\_port = 6432

auth\_type = trust

auth\_file = /usr/pgbouncer/etc/userlist.txt

ignore\_startup\_parameters = extra\_float\_digits

vi /usr/pgbouncer/share/doc/pgbouncer/userlist.txt "pgbadmin" "pgbadmin"

* **管理pgbouncer**
* **Pgbouncer开关服务**

安装目录赋权给postgres用户：chown -R postgres:postgres /usr/pgbouncer，后续需用postgres用户启动pgbouncer

**启动：** cd /usr/pgbouncer/bin

./pgbouncer -d /usr/pgbouncer/etc/pgbouncer.ini #实际使用时用-d参数，表明以后台程序的方式运行。

./pgbouncer -v /usr/pgbouncer/etc/pgbouncer.ini  # 初次使用以及检查问题时建议使用-v参数，该参数能使pgbouncer打出较详细的log，便于定位问题。

**重启：** cd /usr/pgbouncer/bin

./pgbouncer -R -d /usr/pgbouncer/etc/pgbouncer.ini

**停止：** cd /usr/local/pgsql/bin

./psql -p 6432 -U pgbadmin -h 192.168.100.122 pgbouncer

shutdown;

**加载配置：** cd /usr/local/pgsql/bin

./psql -p 6432 -U pgbadmin -h 192.168.100.122 pgbouncer

reload;

* **查看pgbouncer管理内容**

pgbouncer有一个虚拟的db存在，名称就是“pgbouncer”，执行如下命令进入到管理的终端：

cd /usr/local/pgsql/bin

./psql -U pgbadmin -p 6432 -h 192.168.100.122 pgbouncer

show config; 查看配置

show lists; 显示连接池的计数信息

show databases; 查看库

show pools;查看连接池信息

show clients; 用来查看客户端连接信息

**show help;**

查看管理命令的帮助信息,这些命令都必须以分号结尾

**show config;**

显示当前配置设置,一个配置一行,字段如下：   
\* key：配置变量名称   
\* value：配置值   
\* changeable：yes 或 no,显示这个变量是否可以在运行时修改如果为 no,那么这个变量只能在启动的时候修改

**show pools;**

列出连接池   
\* database：数据库名   
\* user：用户名   
\* cl\_active：当前 active (活跃)的客户端连接的个数   
\* cl\_waiting：当前 waiting (等待)的客户端连接个数   
\* sv\_active：当前 active (活跃)的服务器连接个数   
\* sv\_idle：当前 idle (空闲) 的服务器连接个数   
\* sv\_used：当前 used (在使用)的服务器连接个数   
\* sv\_tested：当前 tested (测试过)的服务器连接个数   
\* sv\_login：当前 login (登录)到 PostgreSQL 服务器的个数   
\* maxwait：队列中第一个(最老的那个)客户端等待的时间长度,单位是秒.如果这个数值开始上升,那么就意味着当前的连接池中的服务器处理请求的速度不够快.原因可能是服务器过载,也可能只是 pool\_size 太小

**show stats;**

\* database：统计是根据每个数据库分比例的

\* total\_requests：连接池处理的SQL请求的总数

\* total\_received：接收到的网络流量的总字节数

\* total\_sent：发出的网络流量的总字节数

\* total\_query\_time：活跃在与数据库上面的时间开销总数,单位是微秒

\* avg\_req：在最后一次统计过程中的每秒平均请求数

\* avg\_recv：每秒(从客户端)接收到的平均数据量

\* avg\_sent：每秒发送(给客户端)的平均数据量

\* avg\_query：平均的查询时间,单位是微秒

**show servers;**

列出数据库与pgbouncer之间连接   
\* type：S,表示服务器   
\* user：gbouncer用于连接服务器的用户名   
\* database：服务器端的数据库名   
\* state：pgbouncer 服务器连接的状态 active、used、idle   
\* addr：PostgreSQL服务器的IP地址   
\* port：PostgreSQL服务器的端口   
\* local\_addr：本地机器上的发起连接地址   
\* local\_port：本地机器上的发起连接端口   
\* connect\_time：连接建立的时间   
\* request\_time：请求发出的时间   
\* ptr：这个连接的内部对象地址,用做唯一 ID   
\* link：这个服务器对应的客户端地址

**show clients;**

列出客户端及客户端连接状态   
\* type：C,表示客户端   
\* user：客户端连接的用户   
\* database：数据库名   
\* state：客户端连接的状态 active、used、waiting或者idle之一   
\* addr：客户端的 IP 地址   
\* port：客户端连接去的端口   
\* local\_addr：本地机器上连接到的对端地址   
\* local\_port：本地机器上的连接到的对端端口   
\* connect\_time：最后的客户端连接的时间戳   
\* request\_time：最后的客户端请求的时间戳   
\* ptr：这个连接的内部对象的地址,用做唯一 ID   
\* link：这个客户端连接对应的服务器的地址

**show lists;**

显示连接池的计数信息   
\* databases：数据库的个数   
\* users：用户的个数   
\* pools：连接池的个数   
\* free\_clients：空闲客户端的个数   
\* used\_clients：已用的客户端的个数   
\* login\_clients：处于已登录状态的客户端个数   
\* free\_servers：空闲服务器个数   
\* used\_servers：已用服务器个数

**show databases;**

列出pgbouncer数据库别名及相关数据库   
\* name：已配置的数据库名字记录   
\* host：pgbouncer 连接到的主机名   
\* port：pgbouncer 连接到的端口号   
\* database：pgbouncer 实际连接的数据库名   
\* force\_user：当用户是连接字串的一部分的时候,在 pgbouncer 和 PostgreSQL 之间的连接会强制成给出的用户,不管 client user 是什么   
\* pool\_size：最大的服务器端连接数目

**show fds;**

显示正在使用的 fd 列表如果连接的用户的用户名是 “pgbouncer”,那么通过 unix socket 连接,并且和运行的进程有同样的 UID,实际的 fd 列表是通过这个连接传递的这个机制用于做在线重启   
- fd：文件描述符的数字值   
- task：pooler,client 或 server 之一   
- user：使用该 FD 的连接用户   
- database：使用该 FD 的连接的数据库   
- addr：使用该 FD 的连接的 IP 地址,如果使用的是 unix socket,就是 unix   
- port：使用该 FD 的连接的端口号   
- cancel：这个连接的取消键字   
- link：对应的服务器/客户端的 fd如果为 idle (空闲)则为 NULL

**DISABLE db;**

拒绝指定数据库上所有新客户端连接

### ENALBLE db;

准许之前DISABLE命令之后的新客户端连接

### PAUSE [db];

尝试从所有服务器断开连接(等待query完成),在所有query完成之前,此命令不会返回,在数据库重新启动时使用.如果给出了数据库名字则只对该数据库有用

### KILL db;

立即删除给定数据库上所有客户端以及数据库连接

### SUPEND;

刷新所有socket缓存,并且停止监听,在缓存flush之前此命令不会有任何返回.使用场景:pgbouncer在线重新启动时使用

### RESUME [db]

从之前PAUSE或者SUPEND命令恢复之前状态

### SHUTDOWN;

pgbouncer进程退出

### RELOAD;

重新加载其配置文件并更新可更改的配置

* **通过pgbouncer连接数据库**

cd /usr/local/pgsql/bin

./psql -h 192.168.100.122 -p 6432 -U pgbadmin -d benchmarksql

**benchmarksql通过pgbouncer链接数据库**

conn=jdbc:postgresql://192.168.100.122:6432/benchmarksql

* **pgbouncer配置文件说明**
* **databases配置项**

databases配置说明：每行由key=value对组成,其中key为对外数据库名称,value由多个以空格隔开的key=value对的连接串及相关参数对组成；

实例:

pg\_benchmarksql= host=192.168.100.124 dbname=benchmarksql port=5432 user=benchmarksql password=benchmarksql pool\_size=500 client\_encoding=UNICODE datestyle=ISO connect\_query=’SELECT 1’

上面的配置说明了该pgbouncer创建了针对192.168.100.124上的benchmarksql库的一个连接池，该连接池对调用方的呈现的数据库名称是pg\_benchmarksql ,它映射到了benchmarksql上。所有访问pbbouncer上的pg\_benchmarksql的请求都会转到benchmarksql上完成。

连接串的各个参数说明:

databases里面的数据库可以配置多个

dbname: 后端数据库名称

host: 后端数据库的主机名或者IP地址

port: 后端数据库监听端口

user: 连接后端数据库的用户名

password: 连接后端数据库的密码

pool\_size: 配置连接池的大小,若没有配置此项将使用[pgbouncer]部分中default\_pool\_size配置的值

connect\_query: 在连接使用之前执行一个SQL语句,用于探测此连接是否正常.如果执行该语句出错,则选择另外一个连接

max\_db\_connections: 配置数据库范围的最大值（即数据库中的所有池都不会有这么多的服务器连接

client\_encoding: 制定客户端字符集编码

datestyle: 指定日志类型参数

timezone: 指定时区

注意:如果在连接串中没有指定user和password,那么pgbouncer将使用给客户端连接pgbouncer时的用户名和密码来连接后端数据库,并为每个不同的用户建立一个连接池;如果连接中指定了user和password,pgbouncer将使用这里设置的用户名和密码来连接后端数据库,这样对使用这项配置的数据库来说,就只有一个连接池了。

* **[pgbouncer]部分的配置**

主要分为下面几类：

* 通用配置项
* 日志配置项
* 控制界面访问控制配置项
* 连接健康检查和超时配置项
* 危险的超时配置项
* 底层网络配置项

下面来依次说这些配置：

* 通用配置

logfile 指定日志文件，默认值是/var/log/pgbouncer/pgbouncer.log

pidfile 指定pid文件位置，默认值是/var/run/pgbouncer/pgbouncer.pid

listen\_addr 监听的地址，默认值是127.0.0.1，可以使用\*号表示监听所有IP地址。

listen\_port 监听的端口，默认值是6432

unix\_socket\_dir 指定unix socket文件的目录，默认为/tmp目录

unix\_socket\_mode 指定unix socket文件的权限，默认值为0777

unix\_socket\_group 指定unix socket文件的组，默认无

user 指定启动PgBouncer的用户名，windows系统不支持此设置

auth\_type 认证的类型，默认是trust，其他值包括md5，crypt，plain，any。用的较多的是md5

auth\_file 认证文件的位置，默认值是/etc/pgbouncer/userlist.txt

pool\_mode 指定池的模式，默认是session模式，还可以是transaction和statement

max\_client\_conn 允许的最大连接数

default\_pool\_size 默认的池大小，默认值20

min\_pool\_size 最小的池大小，每个连接池至少会向后端数据库保持多少个连接

reserve\_pool\_size 连接池的保留连接数

reserve\_pool\_timeout 保留连接的超时时间

server\_round\_robin 负载均衡的方式是否设置为“round robin”，默认为关闭，即后进先出

ignore\_startup\_parameters 默认情况PgBouncer只会跟踪一些默认参数，并且能检测这些参数的变化，保持这些参数和客户端的一致。在这个配置后面跟的配置后被PgBouncer忽略，不会被检查。

disable\_pqexec 是否禁止简单查询协议，默认值为0。简单查询协议允许一个请求发送多个SQL，容易导致SQL注入攻击。

* 日志配置项：

syslog 是否打开syslog，windows下打开eventlog，默认值为0，表示不打开。

syslog\_ident 默认为PgBouncer

syslog\_facility

log\_connections 是否记录连接成功的日志，默认值为1，表示记录

log\_disconnections 是否记录断开连接的日志，默认值为1，表示记录

log\_pooler\_errors 连接池法网客户端的错误是否记录在日志中，默认值为1，表示记录

stats\_period 把汇总的统计信息写入日志的时间周期，默认是60s

* 控制界面访问控制配置项：

admin\_users 管理用户名，默认值是postgres

stats\_users 允许连接到控制界面，查看连接池只读信息的用户列表。可以执行除“SHOW FDS”以外的其他“SHOW”命令

server\_reset\_query

server\_check\_delay 空闲的连接多长时间进行一次健康检测，判断连接是否可用。如果设置为0，则立即检测，默认值为30s

server\_check\_query，进行健康检查的SQL语句，如果为0，表示不检测，默认值为“select 1;”

server\_lifetime 连接的存活时间，连接超过这个时间就会被关闭，默认为3600，设置为0表示只使用一次。

server\_idle\_timeout 连接的idle时间，超过此时间，连接会被关闭。默认为600

server\_connect\_timeout 后端数据库的login时间超过这个值就会被关闭。默认为15s

server\_login\_retry 传教到后端数据库的连接失败后，等多长时间后重试，默认为15s

client\_login\_timeout 客户端与PgBouncer建立连接后，如果无法在这段时间内完成登录，那么连接会断开，默认为60s

* 危险超时配置项

指的是为防止一些未知错误或者原因导致系统卡住的针对性配置。

query\_timeout 允许超过该时间值的SQL会被断开，应该比SQL实际的执行时间稍长，也需要比数据库的statement\_timeout参数值更大。为了应付一些未知的网络问题。默认为0.0，禁止使用

query\_wait\_timeout 请求在队列中等待被执行的最长时间，如果超过该时间还没有分配到连接，就会断开。默认为0，禁止使用。

client\_idle\_timeout 客户端连接空闲超过该时间，则断开连接。默认值为0，禁止使用

idle\_transaction\_timeout 客户端启动事务后，超过这个时间没有结束事务，则关闭这个客户端连接。默认值为0，禁止使用

* 底层网络连接配置

pkt\_buf 用于网络包的内部缓冲区大小，会影响发出的TCP包的大小即内存的使用，默认值为2048，一般保持默认值

max\_packet\_size 通过PgBouncer最大的包大小，包可以是一个SQL，也可以是返回的结果，默认值是2147483647

listen\_backlog TCP监听函数listen的Backlog参数，默认值为128

sbuf\_loopcnt 处理过程中，每个连接处理多少数据就切换到下一个连接。默认为5，如果设置为0，表示不限制。不限制时，一个连接发送大量数据，另外的连接可能就会空闲，导致被结束掉。

tcp\_defer\_accept linux下，默认为45，其他平台为0。详细解释用man 7 tcp来查看

tcp\_socket\_buffer 默认没有设置

tcp\_keepalive 是否以操作系统的默认值打开基本的keepalive 设置，在linux下，操作系统的keepalive里，默认值时tcp\_keepidle=7200, tcp\_keepintvl-75,tcp\_keepcnt=9，其他操作系统类型，默认值为1

tcp\_keepcnt 默认未设置

tcp\_keepidle 默认未设置

tcp\_keepintvl 默认未设置