Data Base Administrator PostgreSQL

Ernesto Quiñones A.
ernesto@eqsoft.net

Sesión 9

Herramientas de Administración

Administración de Bloqueos

PgPool

Herramientas de Administración
TOP y HTOP

TOP es un comando convencional de Linux / Unix que nos permite ver los procesos que se ejecutan en el servidor.

Los procesos ejecutados por PostgreSQL son normalmente ejecutados por los usuarios “postgres”, “pgsql” ó “postmaster”.

```
ernesto@depeche:~/aaa$ ps aux | egrep postgres
postgres   961  0.0  0.1 101580  2228 ?        S    Oct08   0:06 /usr/lib/postgresql/8.4/bin/postgres
-D /var/lib/postgresql/8.4/main -c config_file=/etc/postgresql/8.4/main/postgresql.conf
postgres   989  0.0  0.3 101704  6208 ?        Ss   Oct08   0:11 postgres: writer process
postgres   990  0.0  0.0 101580   568 ?        Ss   Oct08   0:08 postgres: wal writer process
postgres   991  0.0  0.0 102380  1320 ?        Ss   Oct08   0:09 postgres: autovacuum launcher process
postgres   992  0.0  0.0  73704   848 ?        Ss   Oct08   0:17 postgres: stats collector process
```

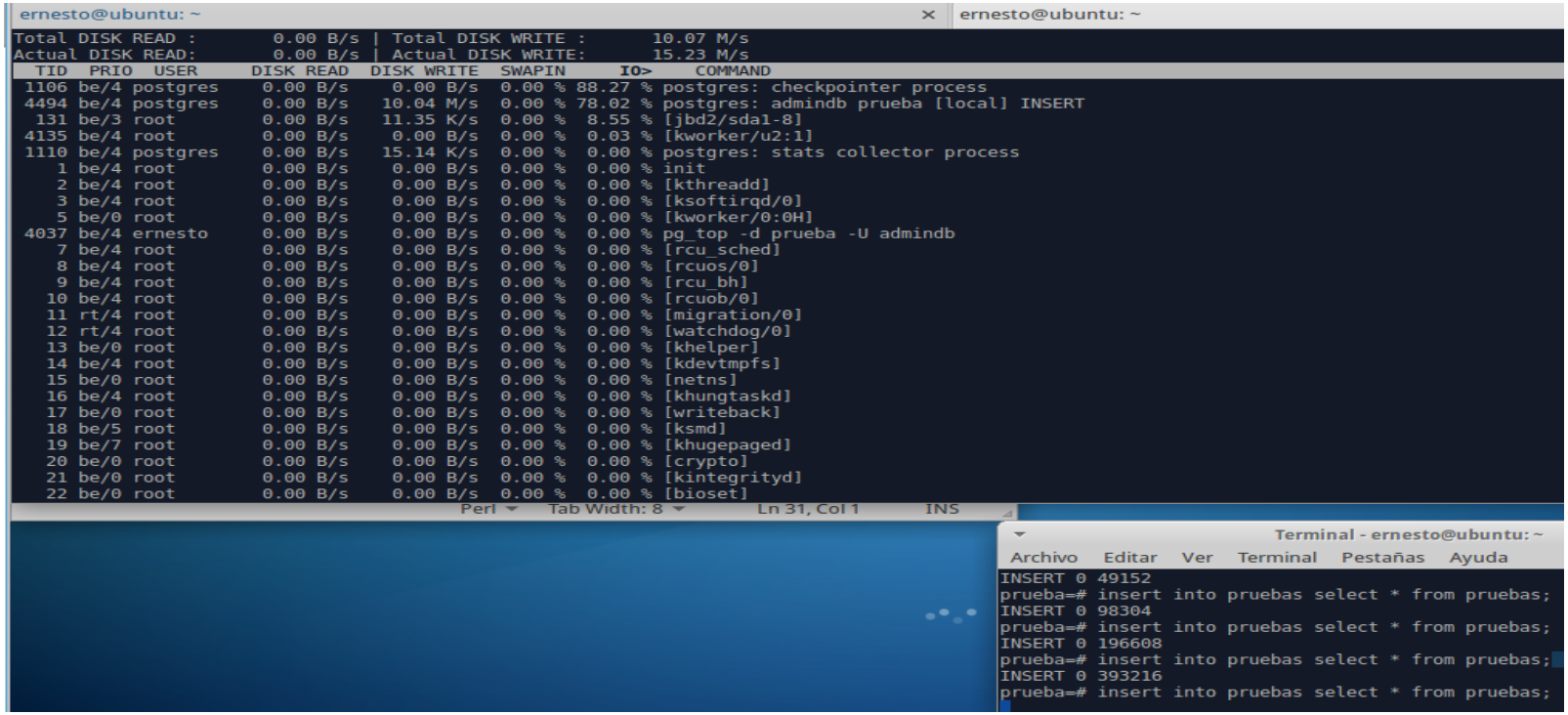
Recuerde que PostgreSQL maneja un procesador por conexión, si tiene varios procesadores un proceso lanzado no usará varios procesadores, solo uno, deja los otros disponibles para otras conexiones que lanzan sus propios procesos, la ventaja del uso de esta funcionalidad se complementa con la capacidad de administración de usuarios concurrentes que tenga el sistema operativo que se está usando (que soporte SMP).

Nunca mate un proceso que demora mucho

Herramientas de Administración
ioTOP

Esta herramienta nos permite monitorear el consumo de accesos de lectura/escritura a los dispositivos de almacenamiento, por defecto no viene instalado.

```
ernesto@ubuntu:~$ sudo apt-get install iotop
Leyendo lista de paquetes... Hecho
...
Configurando iotop ...
```



Herramientas de Administración
ioStat

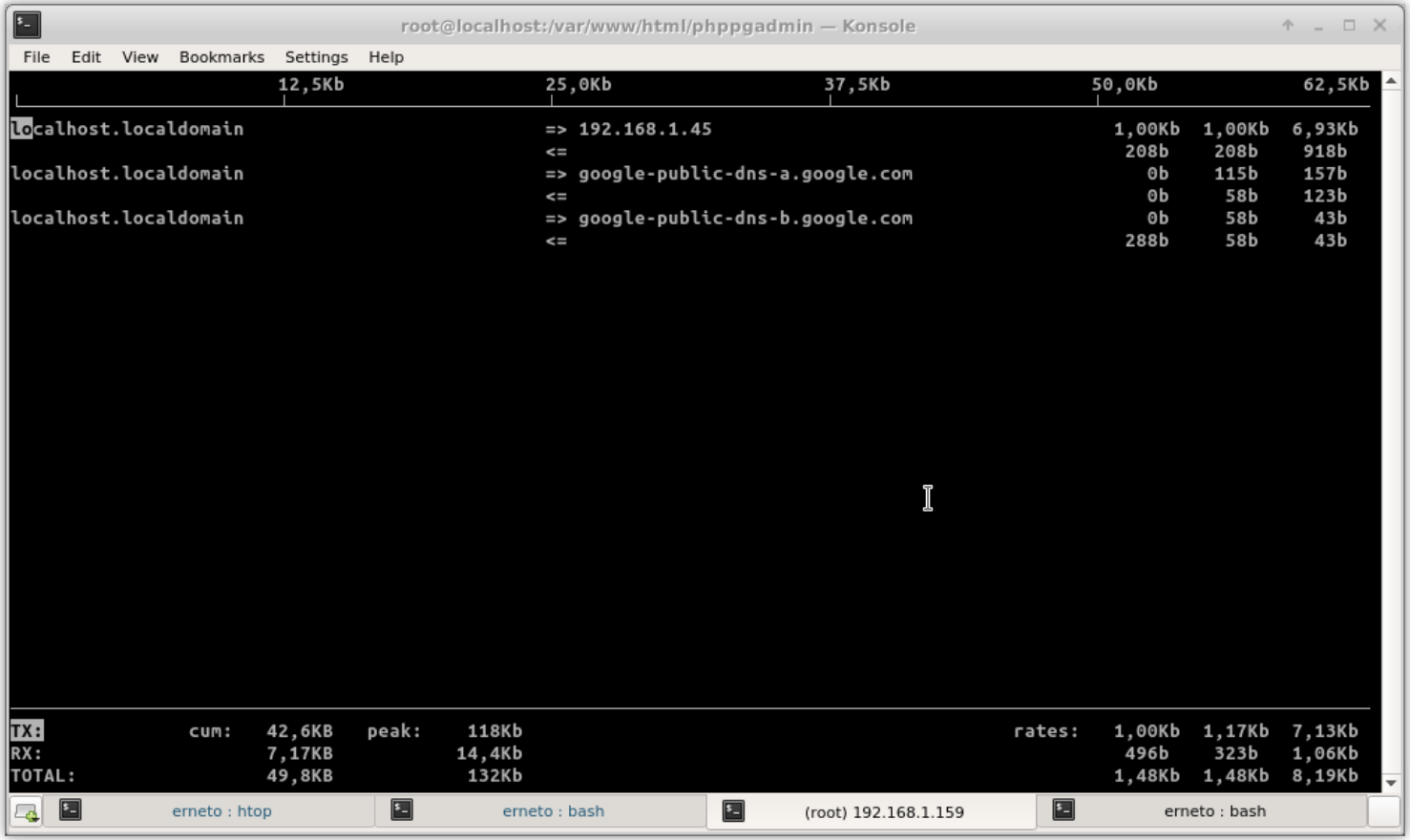
Permite monitorizar el uso de los HD por separado, importante para determinar balanceo de cargas, esta herramienta viene como parte de las utilidades del paquete “sysstat”.

```
[root@host]# iostat -k
Linux 2.6.32-71.el6.x86_64 (host.dominio) 05/21/2011   _x86_64_ (2 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           6.32    0.00    1.36    1.56    0.00   90.76

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda               5.58        71.51        40.52     496736     281500
```

* %iowait es el parámetro que principalmente deberíamos controlar, un número alto significa que tenemos una saturación en el uso de los dispositivos de almacenamiento y que debemos empezar a buscar alternativas de almacenamiento.

Herramientas de Administración
iftop

Permite monitorizar el tráfico de red.



Herramientas de Administración
PG_TOP

PgTop es una herramienta que nos facilita el seguimiento y control de los procesos y bloqueos que se están ejecutando en una determinada base de datos.

http://ptop.projects.postgresql.org/

a)Consultamos el nombre del paquete:

```
ernesto@ubuntu:~$ sudo apt-cache search pgtop
pgtop - PostgreSQL performance monitoring tool akin to top
ptop - Paquete vacío de transición
```

b)Instalamos el PgTop:

```
ernesto@ubuntu:~$ yum install pg_top10
Install  1 Package

Total download size: 52 k
Installed size: 114 k
Is this ok [y/d/N]: y
Downloading packages:
pg_top10-3.7.0-5.rhel7.x86_64.rpm                                  |  52 kB  00:00:01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : pg_top10-3.7.0-5.rhel7.x86_64                          1/1
  Verifying  : pg_top10-3.7.0-5.rhel7.x86_64                          1/1

Installed:
  pg_top10.x86_64 0:3.7.0-5.rhel7

Complete!
```

Herramientas de Administración
PG_TOP

c)Ejecutamos el PgTop con un usuario con privilegios de super usuario, esta herramienta requiere que el password del usuario a utilizar para conectarnos este en una variable de entorno (esto podría ser un gran problema de seguridad).

```
[user@host]# export  PGPASSWORD=dbadmin  <-- aquí debe ir el password del usuario que se
                                             conectará a la base de datos, generalmente es
                                             un superusuario para que tenga la suficiente
                                             cantidad de permisos a las estructuras internas
                                             del motor.
```

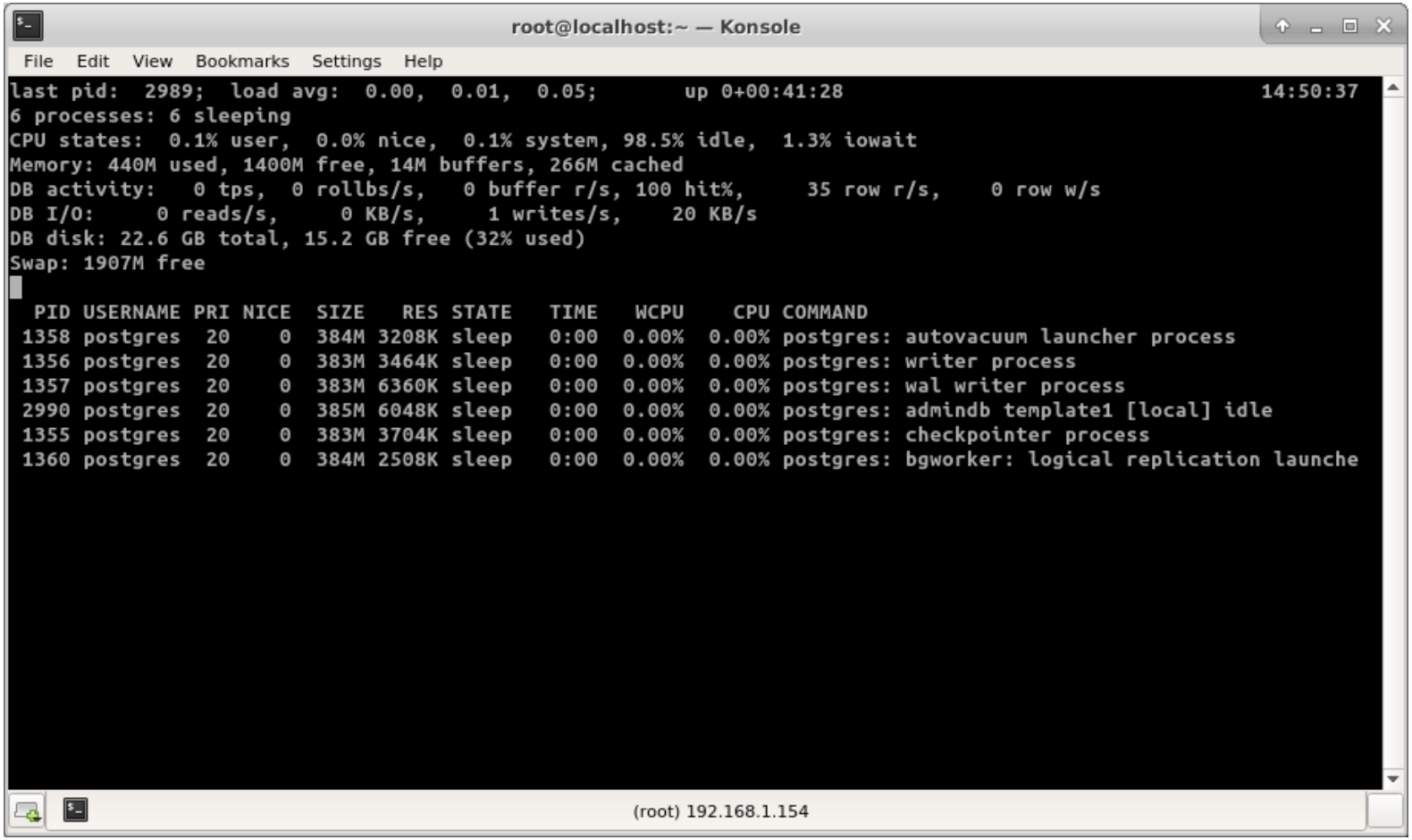
```
ernesto@ubuntu:~$ pg_top -d prueba -U admindb
```

ACTUALIZACION PG_TOP10: ya es posible ingresar el password

```
[root@localhost ~]# pg_top -d template1  -U admindb  -W
Password:
```

En la pantalla principal podemos ver información del CPU, memoria, transferencia de datos, disponibilidad de buffer, swap disponible, etc. Además de las diferentes conexiones activas a la base de datos.

Herramientas de Administración
PG_TOP



Herramientas de Administración
PG_TOP

```
Top version 3.7.0, Copyright (c) 1984 through 2007, William LeFebvre
A top users display for Unix
These single-character commands are available:

^L      - redraw screen
<sp>    - update screen
A       - permite ver el plan de ejecución de un query, especifique el PID del proceso
C       - toggle the use of color
E       - show execution plan (UPDATE/DELETE safe) (ver versión actualizada del plan de ejecución)
I       - show I/O statistics per process (Linux only)
L       - show locks held by a process
M       - sort by memory usage
N       - sort by pid
P       - sort by CPU usage
R       - show user table statistics
Q       - show current query of a process
T       - sort by time
X       - show user index statistics
c       - permite ver la linea de comandos completa
d       - change number of displays to show
e       - list errors generated by last "kill" or "renice" command
h or ?  - help; show this text
i       - toggle the displaying of idle processes
k       - kill processes; send a signal to a list of processes
          not avilable when connected to a remote database
n or #  - change number of processes to display
o       - specify sort order (cpu, size, res, time, command)
          index stats (idx_scan, idx_tup_fetch, idx_tup_read)
          table stats (seq_scan, seq_tup_read, idx_scan, idx_tup_fetch,
                       n_tup_ins, n_tup_upd, n_tup_del)
          i/o stats (pid, rchar, wchar, syscr, syscw, reads, writes, cwrites, command)
q       - quit
r       - renice a process
          not avilable when connected to a remote database
s       - change number of seconds to delay between updates
t       - Toggle between cumulative or differential statistics when viewing
          user table or user index statistics.
u       - display processes for only one user (+ selects all users)

Not all commands are available on all systems.
```

Administración de Bloqueos

Dado que PostgreSQL es una base de datos que soporta concurrencia entonces es posible que un dato al momento de ser consultado alguien lo este borrando ó modificando en otra transacción, para determinar en una consulta si los datos que estamos usando están siendo afectados por otro usuario tenemos los valores de xmin y xmax, cuando xmax está en blanco quiere decir que el dato no está siendo afectado por otra transacción, cuando xmin y xmax tienen el mismo valor entonces el dato o fue modificado o fue borrado por otra transacción.

```
prueba=# select *, xmin,xmax from tablita;
 id | xmin | xmax
----+------+------
  1 |  661 |    0
  3 |  661 |    0
  5 |  661 |    0
  7 |  661 |    0
```

En otra sesión ejecutamos un lock a la tabla del ejemplo y actualizamos algunos campos, el XMAX de los registros cambiará.

```
prueba=# select *, xmin,xmax from tablita;
 id | xmin | xmax
----+------+------
  1 |  661 |    0
  3 |  661 |  661                          <--- el dato fue borrado o modificado
  5 |  661 |    0
  7 |  661 |    0
```

Administración de Bloqueos

Una manera sencilla de determinar cuantas conexiones tenemos vivas, y que están haciendo es consultar la tabla pg_stat_activity

```
prueba=# select datname, usename, current_query, query_start from pg_stat_activity;

 datname | usename |                                current_query                |  query_start

---------+---------+- ----------------------------------------------------+ -----------
 prueba  | dbadmin | select datname, usename, current_query, query_start from
                                                    pg_stat_activity; | 2011-10-01
                                                                        12:25:09.482213-05
```

Administración de Bloqueos

Para ver el tiempo de ejecución de un query solo necesitas restar el current_timestamp – el menor tiempo entre query_start y xact_start.

```
prueba=# select datname, usename, substr(current_query,1,15),
         query_start, xact_start,
         current_timestamp - least(query_start, xact_start) as runtime
         from pg_stat_activity;

 datname | usename |     substr      |          query_start          |
xact_start           | runtime
---------+---------+-----------------+-------------------------------
+-------------------------------+----------
 prueba  | dbadmin | select datname, | 2011-10-01 12:29:14.000863-05 | 2011-10-01
12:29:14.000863-05 | 00:00:00
```

Administración de Bloqueos

Consultando la vista PG_LOCKS podemos verificar los bloqueos activos al momento de ejecutar la consulta, recuerden que los bloqueos siempre suceden, estos no son malos, sino se rompería la integridad de los datos, el problema es tener bloqueos que duren demasiado tiempo:

```
prueba2=# select * from pg_locks;

  locktype    | database | relation | page | tuple | virtualxid | transactionid | classid |
objid | objsubid | virtualtransaction | pid  |        mode         | granted | fastpath
--------------+----------+----------+------+-------+------------+---------------+---------
+-------+----------+--------------------+------+---------------------+---------+----------
virtualxid    |          |          |      |       | 4/139      |               |         |
      |          | 4/139              | 8266 | ExclusiveLock       | t       | t
virtualxid    |          |          |      |       | 3/28       |               |         |
      |          | 3/28               | 8154 | ExclusiveLock       | t       | t
relation      |    16385 |    11187 |      |       |            |               |         |
      |          | 2/1191             | 8256 | AccessShareLock     | t       | t
virtualxid    |          |          |      |       | 2/1191     |               |         |
      |          | 2/1191             | 8256 | ExclusiveLock       | t       | t
relation      |    16385 |    16442 |      |       |            |               |         |
      |          | 3/28               | 8154 | AccessExclusiveLock | t       | f
transactionid |          |          |      |       |            |           849 |         |
      |          | 3/28               | 8154 | ExclusiveLock       | t       | f
relation      |    16385 |    16442 |      |       |            |               |         |
      |          | 4/139              | 8266 | AccessShareLock     | f       | f
(7 rows)
```

Administración de Bloqueos

Para cruzar que usuarios estan generando que bloqueos podemos cruzar las vistas PG_LOCKS y PG_STAT_ACTIVITY.

```
prueba2=# select datname, usename, pid,query from pg_stat_activity;

datname | usename  | pid  |                              query
--------+----------+------+-----------------------------------------------------------
prueba2 | postgres | 8256 | select datname, usename, pid,query from pg_stat_activity;
prueba2 | usr2     | 8154 | alter table tablita add column valores3 int;
prueba2 | usr3     | 8266 | select * from tablita;
prueba2 | usr2     | 8275 | select * from pg_stat_activity;

prueba2=# select database,relation,pid, mode from pg_locks;

database | relation | pid  |       mode
---------+----------+------+-------------------
         |          | 8266 | ExclusiveLock
         |          | 8154 | ExclusiveLock
   16385 |    11187 | 8256 | AccessShareLock
         |          | 8256 | ExclusiveLock
   16385 |    16442 | 8154 | AccessExclusiveLock
         |          | 8154 | ExclusiveLock
   16385 |    16442 | 8266 | AccessShareLock
```

Administración de Bloqueos

Tipos de Locks a nivel de tabla:

- **ACCESS SHARE** : se genera cuando se hace un SELECT, solamente lecturas.
- **ROW SHARE** : se genera cuando se hace un SELECT FOR UPDATE, SELECT FOR SHARE
- **ROW EXCLUSIVE** : se genera cuando se hace un INSERT, UPDATe o DELETE
- **SHARE UPDATE EXCLUSIVE** : se genera en tareas de administración y mantenimiento de tablas, entre ellas están VACUUM (mientras no sea FULL), ANALYZE, CREATE INDEX CONCURRENTLY, ALTER TABLE VALIDATE y ALTER TABLE.
- **SHARE** : se genera al crear ejecutar un CREATE INDEX (mientras no sea CONCURRENTLY).
- **SHARE ROW EXCLUSIVE** : Se lanza de forma manual y para procesos donde queremos conscientemente proteger nuestra data de cambios.
- **EXCLUSIVE** : se genera cuando se actualiza una Vista Materializada, permite sin embargo lecturas en las tablas en paralelo.
- **ACCESS EXCLUSIVE** : se genera al ejecutar un DROP TABLE, TRUNCATE, REINDEX, CLUSTER, y VACUUM FULL. Alternativamente algunas variaciones de ALTER TABLE tambien generan este bloqueo. Este es el bloqueo por defecto cuando se hace un Lock a un tabla.

Administración de Bloqueos

La siguiente tabla muestra los conflictos entre los diferentes tipos de bloqueo, como se puede ver el Access Exclusive es el más complicado de manejar de todos.

Requested Lock Mode	Current Lock Mode							
	ACCESS SHARE	ROW SHARE	ROW EXCLUSIVE	SHARE UPDATE EXCLUSIVE	SHARE	SHARE ROW EXCLUSIVE	EXCLUSIVE	ACCESS EXCLUSIVE
ACCESS SHARE								X
ROW SHARE							X	X
ROW EXCLUSIVE					X	X	X	X
SHARE UPDATE EXCLUSIVE				X	X	X	X	X
SHARE			X	X	X	X	X	X
SHARE ROW EXCLUSIVE			X	X	X	X	X	X
EXCLUSIVE		X	X	X	X	X	X	X
ACCESS EXCLUSIVE	X	X	X	X	X	X	X	X

PgPool es un producto que nos permite administrar un “pool” de conexiones activas a la base de datos, la idea es tener un conjunto de conexiones persistentes separadas en la base de datos que atenderán una o más aplicaciones especificas, PgPool es un intermediario entre la aplicación y la base de datos que mantiene conexiones persistentes siempre abiertas y disponibles para su uso.

¿Porque tener un pool de conexiones?

1) **PostgreSQL no está altamente diseñado para responder rápidamente en conectarse a la base de datos, aunque esto sucede rápidamente es necesario aperturar procesos por cada conexión lo cual demora y tiene un costo alto.**
2) **Generalmente son las aplicaciones web las que tienen la mayor demanda de abrir y cerrar conexiones.**
3) **Es necesario limitar la cantidad de concurrencia a la base de datos para balancear el uso de recursos en el servicio.**

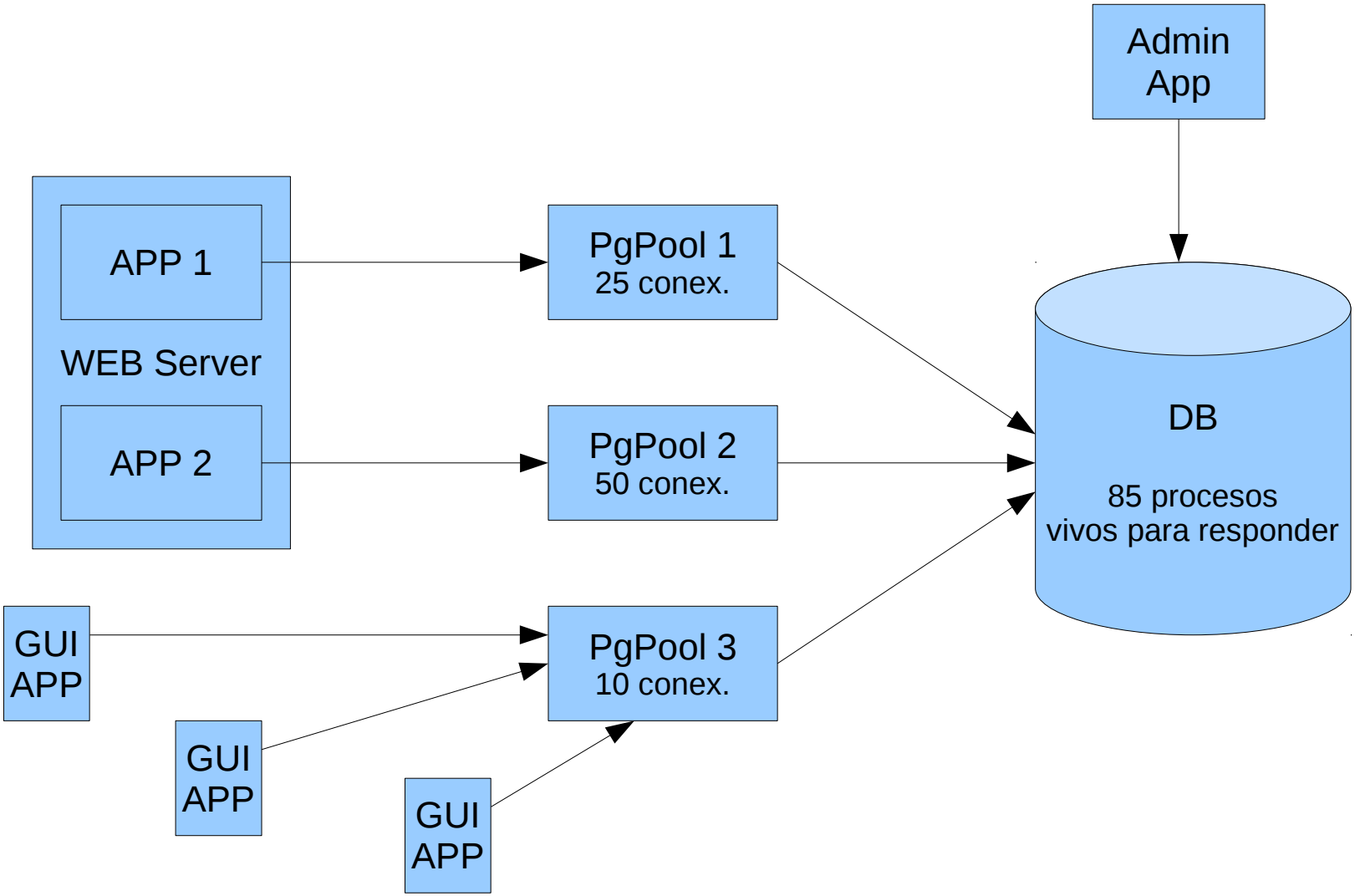
¿Cuantas conexiones debo configurar para que esten permanentemente disponibles?

Eso es una pregunta complicada de responder, la formula no oficial dice que hasta 3 veces la cantidad de “cores” que tengan los procesadores del servidor en el que se encuentra instalada la base de datos, debe recordarse que PostgreSQL usa un core por cada conexión establecida a la vez.

¿Es buena idea configurar de golpe más de 100 conexiones?

No, podría saturarse el servicio con conexiones que no son usadas nunca, se puede empezar en 100 e ir disminuyendo el número hasta encontrar la cantidad real que es necesaria.

PgPool inicia sus propios procesos para trabajar con la base de datos, al suceder esto la cantidad de RAM consumida puede ser significante.



Consideraciones generales:

- **Las aplicaciones típicamente no requieren tantas conexiones como pensamos un número bajo en el pool será más que suficiente.**
- **Se estima que en entornos de mas de un par de cientos de usuarios es imperativo usar un pool de conexiones.**
- **El monitoreo básicamente es manual.**
- **PgPool puede ser mucho más que solo pool de conexiones, puede implementarse en entornos más complejos con balanceo de carga y Replicación.**

Documentación en:

http://www.pgpool.net/mediawiki/index.php/Main_Page
https://wiki.postgresql.org/wiki/Pgpool-II

INSTALACION

1. PgPool-II viene integrado en Ubuntu/CentOS:

```
[user@host]$yum install pgpool-II-10
```

EN LA ULTIMA VERSION DE PgPOOL-II ya no son necesarios los pases 2,3 y 4

2. Luego necesitaremos crear un superusuario para conectarse a la base de datos:

```
[user@host]$ su - postgres
[postgres@host]$ createuser --superuser pgpool2  <-- el nombre "pgpool2"
                                                     Puede variar
```

3. Ahora editamos el archivo "pg_hba.conf" para dar accesos a el usuario que hemos creado añadiendo esta linea:

```
host   all   pgpool2   IP/32   trust  <-- debemos reemplazar "IP" por la
                                          Dirección IP desde la que se
                                          conectará PgPool.
                                          Advierta el tema de colocar "trust"
                                          como método de autenticación,
                                          podría ser un problema de seguridad
                                          a futuro.
```

INSTALACION

4. Reiniciamos el servicio de Postgresql

```
[postgres@host]$ service postgresql restart  ← ubuntu
[user@host]$ systemctl restart postgresql-10 ← centos
```

5. Generamos una copia del archivo de configuración y configuramos el archivo “pgpool.conf”

```
[root@localhost ~]# cp /etc/pgpool-II-10/pgpool.conf.sample /etc/pgpool-II-10/pgpool.conf
```

Editamos el archivo de configuración.

```
allow_inet_domain_socket = 1    <-- activamos a PgPool para escuchar
                                    peticiones de conexión (no existe en CentOS)

port = 5433                     <-- Para conectarnos al pool no usaremos el
                                    puerto 5432 de PostgreSQL, usaremos el
                                    puerto 9999 u otro que especifiquemos.

socket_dir = '/tmp'             <-- Directorio donde se creará el PID del
                                    servicio.

backend_host_name ='centos6dvd.eqsoft.net'
backend_host_name ='10.0.2.15' <-- Es la dirección del servidor PostgreSQL,
                                    podemos configurarla como dirección IP o
                                    como nombre de HOST, en el segundo caso es
                                    recomendable añadir al archio /etc/hosts
                                    el nombre de este y la dirección IP.
```

INSTALACION

5. Configuramos el archivo “/etc/pgpool.conf”

```
backend_host_name ='centos6dvd.eqsoft.net'
backend_host_name ='10.0.2.15' <-- Es necesario que “postgresql.conf” tenga
                                   configurado lo siguiente:
                                      listen_addresses = '*'
                                      port = 5432

backend_port = 5432            <-- siempre y cuando “postgresql.conf” tenga
                                   en el parámetro “port” el valor 5432.

num_init_children = 32         <-- cantidad de procesos activos máximos que
                                   podremos levantar, cada uno de ellos es
                                   una conexión a la db.

max_pool = 4                   <-- Cantidad conexiones cacheadas por proceso
                                   activo, esto se usa en casos en que los
                                   procesos activos se agotan para los
                                   Usuarios que no son super usuarios.

child_life_time = 300          <-- tiempo máximo en segundos que vivirá una
                                   conexión sin usarse.
```

INSTALACION

6. Iniciamos el proceso

```
[postgres@host] pgpool          <-- los siguientes parámetros pueden ser usados
                                    -d     información de debug
                                    -n     no inicia como demónio
                                    stop   para el servicio del pool
```

7. Probamos el servicio

```
[root@host] psql template1 -h IP -p 9999 -U pgpool2
```

RECOMENDACIONES

* Nunca inicie el servicio como usuario “root”, puede ser víctima de un ataque de buffer overflow.

Problemas comunes

P: El PID del servicio se quedó pegado por un reinicio brusco.
S: Borrar /tmp/pgpool.pid y reiniciar el servicio.

P: No se aceptan nuevas conexiones
S: Aumentar el parámetro num_init_children
S: Verificar que proceso no esta liberando la base de datos

Para generar un archivo de LOG del estado de PgPool podemos hacer lo siguiente:

```
[postgres@host]$ pgpool -n -d > /tmp/pgpool.log  2>&1 &
```