Национальный Исследовательский Университет
Информационных Технологий, Механики и Оптики

Факультет программной инженерии и компьютерной техники

Лабораторная работа 2

по дисциплине
«Распределенные системы хранения данных»

Вариант - 11

Выполнил:
Студент группы P33101,
Патутин В.М.

Преподаватель:
Николаев В.В.

Санкт-Петербург
2021г.

Задание ЛР:

На выделенном узле создать и сконфигурировать новый кластер БД, саму БД, табличные пространства и новую роль в соответствии с заданием. Произвести наполнение базы.

Персональный пароль для работы с узлом выдается преподавателем.
Обратите внимание, что домашняя директория пользователя /var/postgres/$LOGNAME

Этапы выполнения работы:
Инициализация кластера БД
- Имя узла — pg103.
- Имя пользователя — postgres2.
- Директория кластера БД — $HOME/u08/gtt22.
- Кодировка, локаль — UTF8, английская
- Перечисленные параметры задать через аргументы команды.

Конфигурация и запуск сервера БД
- Способ подключения к БД — TCP/IP socket, номер порта 9011.
- Остальные способы подключений запретить.
- Способ аутентификации клиентов — по имени пользователя.
- Настроить следующие параметры сервера БД: max_connections, shared_buffers, temp_buffers, work_mem, checkpoint_timeout, effective_cache_size, fsync, commit_delay. Параметры должны быть подобраны в соответствии с аппаратной конфигурацией: оперативная память 16 ГБ, хранение на жёстком диске (HDD);
- Директория WAL файлов — поддиректория в PGDATA.
- Формат лог-файлов — log.
- Уровень сообщений лога — NOTICE.
- Дополнительно логировать — завершение сессий и продолжительность выполнения команд.

Дополнительные табличные пространства и наполнение
- На основе шаблона template1 пересоздать базу postgres в новом табличном пространстве:
  - $HOME/u07/gtr77.
- На основе template1 создать новую базу — whitecat.
- От имени новой роли (не администратора) произвести наполнение существующих баз тестовыми наборами данных. Предоставить права по необходимости. Табличные пространства должны использоваться по назначению.
- Вывести список всех табличных пространств кластера и содержащиеся в них объекты.

Пароль для подключения к узлу: ldFwVgdm
Пожалуйста, не выкладывайте пароль в общий доступ.
Не забывайте останавливать свой экземляр БД, когда прекращаете с ним работать.

Выполнение:

1) Инициализация кластера БД:

```
$ initdb --locale=en_US.UTF-8 -D $HOME/u08/gtt22
The files belonging to this database system will be owned by user "postgres2".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

creating directory /var/postgres/postgres2/u08/gtt22 ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Europe/Moscow
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the option -A, or
--auth-local and --auth-host, the next time you run initdb.

Success. You can now start the database server using:

    pg_ctl -D /var/postgres/postgres2/u08/gtt22 -l logfile start
```

2) Конфигурация и запуск сервера БД:

- Изменим pg_hba.conf:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            ident
# IPv6 local connections:
host    all             all             ::1/128                 ident
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     ident
host    replication     all             127.0.0.1/32            ident
host    replication     all             ::1/128                 ident
```

- Изменим postgresql.conf:
  port - устанавливаем порт сервера:

```
                                        # defaults to 'localhost'; use '*' for all
                                        # (change requires restart)
port = 9011                             # (change requires restart)
```

max_connections - устанавливаем максимальное число одновременных подключений к серверу БД. В целом изначальное значение в 100 меня устраивало и я не нашел никаких точных рекомендаций, поэтому увеличил до 128, чтобы в будущем было удобно делить:

```
max_connections = 128                   # (change requires restart)
#superuser_reserved_connections = 3     # (change requires restart)
#unix_socket_directories = '/tmp'       # comma-separated list of directories
```

shared_buffers - Задаёт объём памяти, который будет использовать сервер баз данных для буферов в разделяемой памяти.

Рекомендация - если вы используете выделенный сервер с объёмом ОЗУ 1 ГБ и более, разумным начальным значением shared_buffers будет 25% от объёма памяти. Выделять для shared_buffers более 40% ОЗУ вряд ли будет полезно.
Формула - ОЗУ*25% = 0.25 * 16384МБ = 4096МБ

```
shared_buffers = 4096MB                 # min 128kB
                                        # (change requires restart)
#huge_pages = try                       # on, off, or try
```

При установлении такого значений была получена следующая ошибка:

```
2022-04-03 23:53:15.463 MSK [2414] FATAL:  could not map anonymous shared memory: Resource temporarily unavailable
2022-04-03 23:53:15.463 MSK [2414] LOG:  database system is shut down
```

Великий гугл сказал, что «Это может быть связано с тем, что shared_buffersзадано слишком большое значение», поэтому значение было изменено на 3072МБ. Проблема была решена.

temp_buffers - задаёт максимальное число временных буферов для каждого сеанса. Сложно решить какое значение необходимо, потому что нет информации для чего будет использоваться кластер.

```
                                        # (change requires restart)
temp_buffers = 64MB                     # min 800kB
#max_prepared_transactions = 0          # zero disables the feature
```

work_mem - задаёт объём памяти, который будет использоваться для внутренних операций сортировки и хеш-таблиц, прежде чем будут задействованы временные файлы на диске.
Рекомендация - предел для work_mem можно вычислить, разделив объём доступной памяти (физическая память минус объём под совместно используемые страницы shared_buffers) на максимальное число одновременно используемых активных соединений.
Формула - (ОЗУ- shared_buffers)/ max_connections = 96МБ

```
# you actively intend to use prepared transactions.
work_mem = 96MB                         # min 64kB
#hash_mem_multiplier = 1.0              # 1-1000.0 multiplier on hash table work_mem
```

checkpoint_timeout - Максимальное время между автоматическими контрольными точками в WAL (в секундах). Дефолтное значение в 5 минут идеально сбалансировано.

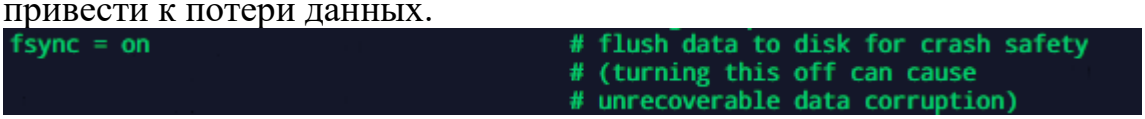
```
checkpoint_timeout = 5min               # range 30s-1d
#checkpoint_completion_target = 0.9     # checkpoint target duration, 0.0 - 1.0
```

effective_cache_size - Определяет представление планировщика об эффективном размере дискового кеша, доступном для одного запроса.
Рекомендация - В качестве начального значения можете использовать 25-50% доступной памяти. Точно должен быть >= shared_buffers.
Формула – 0.5 * 16Гб = 8 ГБ

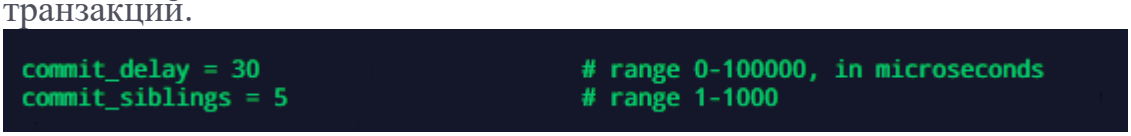
```
#min_parallel_index_scan_size = 512kB
effective_cache_size = 8GB
```

fsync - Если этот параметр установлен, сервер &project; старается добиться, чтобы изменения были записаны на диск физически, выполняя системные

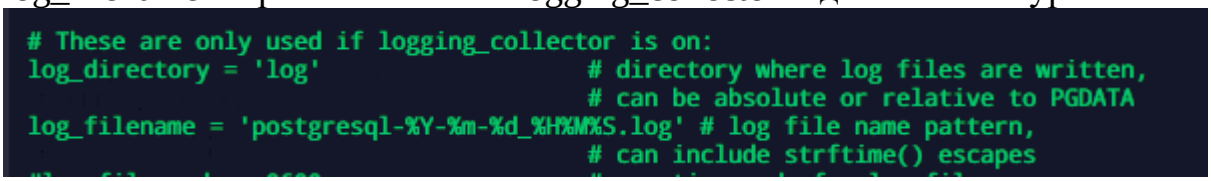
вызовы fsync() или другими подобными методами. Это даёт гарантию, что кластер баз данных сможет вернуться в согласованное состояние после сбоя оборудования или операционной системы. Отключение дает выигрыш в скорости, однако может привести к потери данных.



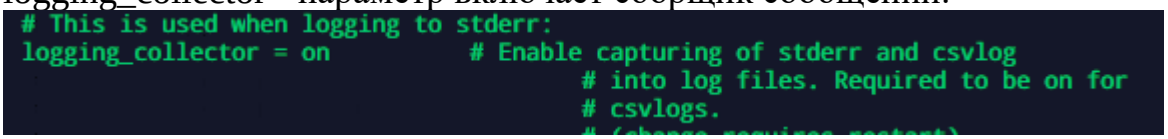
commit_delay - добавляет паузу (в микросекундах) перед собственно выполнением сохранения WAL. Эта задержка может увеличить быстродействие при фиксировании множества транзакций, позволяя зафиксировать большее число транзакций за одну операции сохранения WAL, если система нагружена достаточно сильно и за заданное время успевают зафиксироваться другие транзакции. Эта задержка окажется бесполезной, если никакие другие транзакции не будут зафиксированы за это время, поэтому она добавляется, только в если момент запроса сохранения WAL активны как минимум commit_siblings других транзакций.



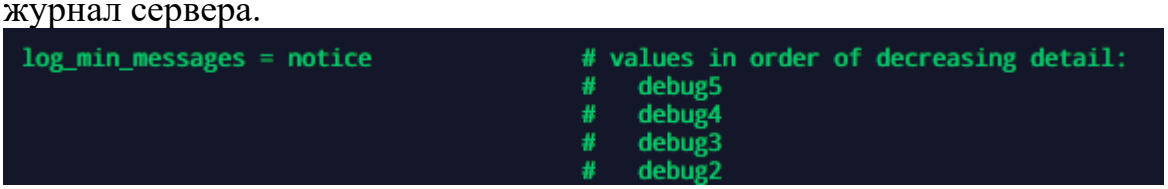
log_directory - при включённом logging_collector, определяет каталог, в котором создаются журнальные файлы.
log_filename - При включённом logging_collector задаёт имена журнальных файлов.



logging_collector - параметр включает сборщик сообщений.



log_min_messages - управляет минимальным уровнем сообщений, записываемых в журнал сервера.



log_disconnections – Включает протоколирование завершения сеанса.
log_duration - Записывает продолжительность каждой завершённой команды.

Запускаем сервер сервер:

```
bash-3.2$ pg_ctl start -D $HOME/u08/gtt22
waiting for server to start....2022-04-04 00:32:27.826 MSK [4811] LOG:  redirecting log output to logging collector process
2022-04-04 00:32:27.826 MSK [4811] HINT:  Future log output will appear in directory "log".
 done
server started
bash-3.2$
```

Подключаемся:

```
bash-3.2$ psql -p 9011 -d postgres
psql (14.2)
Type "help" for help.

postgres=#
```

3)Создаем новое табличное пространство:

```
postgres=# CREATE TABLESPACE laba LOCATION '/var/postgres/postgres2/u07/gtr77';
CREATE TABLESPACE
```

```
postgres=# \db
                List of tablespaces
    Name    |   Owner   |             Location
------------+-----------+----------------------------------
 laba       | postgres2 | /var/postgres/postgres2/u07/gtr77
 pg_default | postgres2 |
 pg_global  | postgres2 |
(3 rows)
```

Пересоздаем базу postgres на основе шаблона template1 в новом табличном пространстве:

```
postgres=# CREATE DATABASE newPostgres WITH TEMPLATE = template1 TABLESPACE = laba;
CREATE DATABASE
```

```
 newpostgres | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres    | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0   | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres2          +
             |           |          |             |             | postgres2=CTc/postgres2
 template1   | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres2          +
             |           |          |             |             | postgres2=CTc/postgres2
```

Создаем новую базу с именем whitecat:

```
newpostgres=# \c newpostgres
You are now connected to database "newpostgres" as user "postgres2".
newpostgres=# CREATE DATABASE whitecat WITH TEMPLATE = template1;
CREATE DATABASE
newpostgres=# \l
 newpostgres | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres    | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0   | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres2          +
             |           |          |             |             | postgres2=CTc/postgres2
 template1   | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres2          +
             |           |          |             |             | postgres2=CTc/postgres2
 whitecat    | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
```

Создаем новую роль:

```
newpostgres=# CREATE ROLE s282606 LOGIN;
CREATE ROLE
newpostgres=# grant create on tablespace laba to s282606;
GRANT
newpostgres=# create user vovan;
CREATE ROLE
newpostgres-# \du
 postgres2 | Superuser, Create role, Create DB, Replication, Bypass RLS | {}
 s282606   |                                                            | {}
 vovan     |                                                            | {}
```

```
newpostgres=# grant s282606 to vovan;
GRANT ROLE
```

Наполнение базы данных:

```
newpostgres=> \conninfo
You are connected to database "newpostgres" as user "vovan" via socket in "/tmp" at port "9011".
newpostgres=> create table first(id serial primary key, column1 int);
CREATE TABLE
newpostgres=> create table second(id serial primary key, column1 int);
CREATE TABLE
```

4) Вывод таблиц:

```
postgres=# \db+
                                                List of tablespaces
    Name    |   Owner   |            Location             |   Access privileges   | Options |  Size   | Description
------------+-----------+---------------------------------+-----------------------+---------+---------+-------------
 laba       | postgres2 | /var/postgres/postgres2/u07/gtr77 | postgres2=C/postgres2+|         | 8321 kB |
            |           |                                 | s282606=C/postgres2   |         |         |
 pg_default | postgres2 |                                 |                       |         | 32 MB   |
 pg_global  | postgres2 |                                 |                       |         | 560 kB  |
(3 rows)
```

```
postgres=# \l
                                  List of databases
    Name     |   Owner   | Encoding |   Collate   |    Ctype    |    Access privileges
-------------+-----------+----------+-------------+-------------+-------------------------
 newpostgres | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 postgres    | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
 template0   | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres2           +
             |           |          |             |             | postgres2=CTc/postgres2
 template1   | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =c/postgres2           +
             |           |          |             |             | postgres2=CTc/postgres2
 whitecat    | postgres2 | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
(5 rows)
```

Выводы:

Таким образом мною была проделана следующая работа: На выделенном узле создан и сконфигурирован новый кластер БД, сама БД, табличные пространства и новая роль в соответствии с заданием. Произведено наполнение базы.