Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет программной инженерии и компьютерной техники

Отчёт
Лабораторная №2
по дисциплине «Распределенные системы хранения данных»
Вариант 33581

Выполнила: Нестеренко К. М., группа P3316

Преподаватель: Николаев В.В.

Санкт-Петербург
2025

Оглавление

Текст задания

Цель работы - на выделенном узле создать и сконфигурировать новый кластер БД Postgres, саму БД, табличные пространства и новую роль, а также произвести наполнение базы в соответствии с заданием. Отчёт по работе должен содержать все команды по настройке, скрипты, а также измененные строки конфигурационных файлов.
Способ подключения к узлу из сети Интернет через helios:
ssh -J sXXXXXX@helios.cs.ifmo.ru:2222 postgresY@pgZZZ

ssh -J s368575@helios.cs.ifmo.ru:2222 postgres0@ pg103
Способ подключения к узлу из сети факультета:
ssh postgresY@pgZZZ
Номер выделенного узла pgZZZ, а также логин и пароль для подключения Вам выдаст преподаватель.

Этап 1. Инициализация кластера БД

- Директория кластера: $HOME/jtp68
- Кодировка: KOI8-R
- Локаль: русская
- Параметры инициализации задать через переменные окружения

Этап 2. Конфигурация и запуск сервера БД

- Способы подключения: 1) Unix-domain сокет в режиме peer; 2) сокет TCP/IP, принимать подключения к любому IP-адресу узла
- Номер порта: 9581
- Способ аутентификации TCP/IP клиентов: по имени пользователя (изменено: паролю MD5)
- Остальные способы подключений запретить.
- 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 со сценарием OLTP:
  500 транзакций в секунду размером 16КБ; обеспечить высокую доступность (High Availability) данных.

- Директория WAL файлов: $HOME/tua27
- Формат лог-файлов: .csv
- Уровень сообщений лога: ERROR
- Дополнительно логировать: завершение сессий и продолжительность выполнения команд

Этап 3. Дополнительные табличные пространства и наполнение базы

- Создать новые табличные пространства для временных объектов: $HOME/zjq75, $HOME/cou57
- На основе template0 создать новую базу: darkbrowncity
- Создать новую роль, предоставить необходимые права, разрешить подключение к базе.
- От имени новой роли (не администратора) произвести наполнение ВСЕХ созданных баз тестовыми наборами данных. ВСЕ табличные пространства должны использоваться по назначению.
- Вывести список всех табличных пространств кластера и содержащиеся в них объекты.

Данные второй лабораторной (виртуальная машина): pg103, postgres0, 96mxgLYY

Выполнение

Этап 1. Инициализация кластера БД

Устанавливаем переменные окружения:

```
export PGDATA=$HOME/jtp68
export PGLOCALE=ru_RU.KOI8-R
export PGENCODE=KOI8-R
export PGUSERNAME=postgres0
export PGHOST=pg103
```

```
[postgres0@pg103 ~]$ env | grep ^PG
PGENCODE=KOI8-R
PGUSERNAME=postgres0
PGHOST=pg103
PGDATA=/var/db/postgres0/jtp68
PGLOCALE=ru_RU.KOI8-R
```

Создаём директорию кластера, инициализируем его и запускаем сервер:

```
mkdir -p $PGDATA
initdb --locale=$PGLOCALE --encoding=$PGENCODE --username=$PGUSERNAME --
waldir=$HOME/tua27 -D $PGDATA
pg_ctl -D $PGDATA -l logfile start
```

```
[postgres0@pg103 ~]$ pg_ctl -D $PGDATA -l logfile start
ожидание запуска сервера.... готово
сервер запущен
```

Теперь можем перейти к настройке файлов конфигурации

Этап 2. Конфигурация и запуск сервера БД

Способы подключения: 1) Unix-domain сокет в режиме peer; 2) сокет TCP/IP, принимать подключения к любому IP-адресу узла

- Номер порта: 9581
- Способ аутентификации TCP/IP клиентов: (изменено) по паролю MD5
- Остальные способы подключений запретить.

```
# - Connection Settings -

listen_addresses = '*'          # what IP address(es) to listen on;
                                # comma-separated list of addresses;
                                # defaults to 'localhost'; use '*' for all
                                # (change requires restart)
port = 9581                     # (change requires restart)
```

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# 1) Unix-domain
local   all             all                                     peer

# 2) TCP/IP
host    all             all             0.0.0.0/0               md5
```

Настройка параметров сервера БД. Параметры должны быть подобраны в соответствии со сценарием OLTP: 500 транзакций в секунду размером 16КБ; обеспечить высокую доступность (High Availability) данных.

max_connections:

```
port = 9581
max_connections = 500
#reserved_connections = 0
```

Для обеспечения обработки 500 транзакций в секунду с размером 16 КБ, я выбрала значение max_connections = 500, полагая, что каждое соединение будет обрабатывать хотя бы одну транзакцию в секунду. Если установить меньшее значение, есть вероятность ситуации, когда сервер не сможет обрабатывать все запросы. С другой стороны, слишком большое количество соединений может привести к нехватке памяти, так как каждое соединение потребляет ресурсы (work_mem, temp_buffers). Поэтому, на мой взгляд, 500 является оптимальным значением.

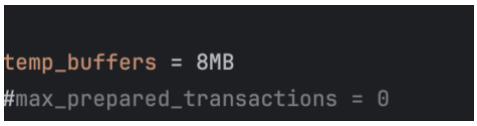
shared_buffers:

```
# - Memory -

shared_buffers = 2GB
```

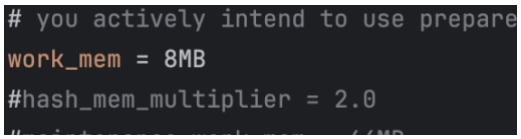
Документация PostgreSQL рекомендует устанавливать значение shared_buffers как 25% от объёма памяти. Увеличение shared_buffers часто требует увеличения некоторых параметров ядра системы, также значение более 2 ГБ поддерживаются только в 64-битных системах. Выберем значение 2GB.

temp_buffers:



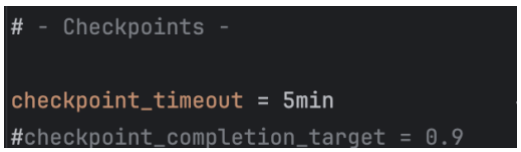
Как значение по умолчанию рекомендуется 8MB, если активно используются временные таблицы, его можно увеличить. В моем случае OLTP-сценарий, где транзакции обычно простые и короткие, не работают с большим объёмом данных, поэтому можно оставить значение по умолчанию.

work_mem:



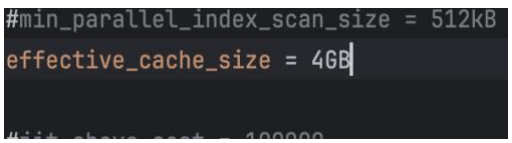
Если запросы не включают сложные сортировки или хеширования, можно оставить значение по умолчанию: 4MB. В OLTP-сценарии такие операции возможны, поэтому можно рассмотреть следующее после универсального большее значение. А так как work_mem выделяется степенями двойки, установим 8MB.

checkpoint_timeout:



По сценарию необходимо обеспечить высокую доступность данных, то есть в том числе минимизировать время восстановления после сбоя. С другой стороны, слишком маленькое значение checkpoint_timeout может ухудшить производительность из-за частых контрольных точек. В моем сценарии нагрузка достаточно высокая, поэтому я не считаю нужным уменьшать значение checkpoint_timeout. Оставим значение по умолчанию (5 минут).

effective_cache_size:



Это представление влияет на оценку стоимости использования индекса; чем выше это значение, тем больше вероятность, что будет применяться сканирование по индексу, чем ниже, тем более вероятно, что будет выбрано последовательное сканирование. Не может быть меньше shared_buffers, выберем значение 4GB.

fsync:

```
fsync = on
```

Рекомендуют выключать этот параметр только при работе с полностью одноразовыми данными. А так как сохранение данных критически важно в данном сценарии, я оставлю fsync включенным.

commit_delay:

```
commit_delay = 0
#commit_siblings = 5
```

Параметр commit_delay можно оставить на 0, чтобы не вводить дополнительную задержку и быстро подтверждать транзакции, что важно в моем сценарии.

Директория WAL файлов: `$HOME/tua27`

Формат лог-файлов: `.csv`

```
log_destination = 'csvlog'
#log_destination = 'stderr'
```

Уровень сообщений лога: `ERROR`

```
log_min_messages = error
```

Дополнительно логировать: завершение сессий и продолжительность выполнения команд

```
log_disconnections = on
log_duration = on
```

```
log_min_duration_statement = 0
```

И перезапустим сервер с новыми настройками с помощью

```
pg_ctl reload
```

Этап 3. Дополнительные табличные пространства и наполнение базы

Для создания табличных пространств и бд воспользуемся template1:

```
psql -p 9581 -d template1 -h pg103
```

Создаем новые табличные пространства для временных объектов: $HOME/zjq75, $HOME/cou57

```
CREATE TABLESPACE zjq75 LOCATION '/var/db/postgres0/zjq75';
CREATE TABLESPACE cou57 LOCATION '/var/db/postgres0/cou57';
```

```
                Список табличных пространств
    Имя      | Владелец  |       Расположение
-------------+-----------+--------------------------
 cou57       | postgres0 | /var/db/postgres0/cou57
 pg_default  | postgres0 |
 pg_global   | postgres0 |
 zjq75       | postgres0 | /var/db/postgres0/zjq75
(4 строки)
```

На основе template0 создадим новую базу: darkbrowncity:

```
CREATE DATABASE darkbrowncity WITH TEMPLATE template0 OWNER postgres0;
```

```
                                                Список баз данных
    Имя      | Владелец  | Кодировка | Провайдер локали |  LC_COLLATE   |   LC_CTYPE    | локаль ICU | Правила ICU |      Права доступа
-------------+-----------+-----------+------------------+---------------+---------------+------------+-------------+-------------------------
 darkbrowncity | postgres0 | KOI8R     | libc             | ru_RU.KOI8-R | ru_RU.KOI8-R |            |             |
 postgres    | postgres0 | KOI8R     | libc             | ru_RU.KOI8-R | ru_RU.KOI8-R |            |             |
 template0   | postgres0 | KOI8R     | libc             | ru_RU.KOI8-R | ru_RU.KOI8-R |            |             | =c/postgres0           +
```

Создадим новую роль, предоставив необходимые права, разрешим подключение к базе.

```
CREATE ROLE new_role LOGIN PASSWORD 'new';
GRANT CONNECT ON DATABASE darkbrowncity TO new_role;
```

Создадим тестовые таблицы и дадим на них нужные права:

```
CREATE TABLE test_table1 (
    id SERIAL PRIMARY KEY,
    NAME TEXT NOT NULL,
    VALUE INTEGER
) TABLESPACE zjq75;
```

```
CREATE TABLE test_table2 (
    id SERIAL PRIMARY KEY,
    NAME TEXT NOT NULL,
    VALUE INTEGER
) TABLESPACE cou57;
```

```
GRANT INSERT ON TABLE "public"." test_table1" TO new_role;
GRANT USAGE, SELECT ON SEQUENCE PUBLIC.TEST_TABLE1_ID_SEQ TO NEW_ROLE;
GRANT INSERT ON TABLE "public"."test_table2" TO new_role;
GRANT USAGE, SELECT ON SEQUENCE PUBLIC.TEST_TABLE2_ID_SEQ TO NEW_ROLE;
```

От имени новой роли (не администратора) произвести наполнение ВСЕХ созданных баз тестовыми наборами данных. ВСЕ табличные пространства должны использоваться по назначению.

```
psql -h pg103 -p 9581 -d darkbrowncity -U new_role
```

```
INSERT INTO test_table1 (NAME, VALUE) VALUES
('A', 1),
('B', 2),
('C', 3);

INSERT INTO test_table2 (NAME, VALUE) VALUES
('A', 1),
('B', 2),
('C', 3);
```

Вывести список всех табличных пространств кластера и содержащиеся в них объекты

```
SELECT
  CASE WHEN ROW_NUMBER() OVER (PARTITION BY COALESCE(t.spcname,
'pg_default') ORDER BY c.relname) = 1
       THEN COALESCE(t.spcname, 'pg_default')
       ELSE NULL
  END AS spcname,
  c.relname
FROM pg_tablespace t
FULL JOIN pg_class c ON c.reltablespace = t.oid
ORDER BY COALESCE(t.spcname, 'pg_default'), c.relname;
```

```
 spcname |              relname
------------+-----------------------------------------------
 cou57      | pg_toast_16409
            | pg_toast_16409_index
            | test_table2
            | test_table2_pkey
 pg_default | _pg_foreign_data_wrappers
            | _pg_foreign_servers
            | _pg_foreign_table_columns
            | _pg_foreign_tables
            | _pg_user_mappings
            | administrable_role_authorizations
            | applicable_roles
            | attributes
            | character_sets
            | check_constraint_routine_usage
            | check_constraints
            | collation_character_set_applicability
            | collations
            | column_column_usage
            | column_domain_usage
```

```
        | column_options
        | column_privileges
        | column_udt_usage
        | columns
        | constraint_column_usage
        | constraint_table_usage
        | data_type_privileges
        | domain_constraints
        | domain_udt_usage
        | domains
        | element_types
        | enabled_roles
        | foreign_data_wrapper_options
        | foreign_data_wrappers
        | foreign_server_options
        | foreign_servers
        | foreign_table_options
        | foreign_tables
        | information_schema_catalog_name
        | key_column_usage
        | parameters
        | pg_aggregate
        | pg_aggregate_fnoid_index
...skipping...
        | pg_db_role_setting_databaseid_rol_index
        | pg_parameter_acl
        | pg_parameter_acl_oid_index
        | pg_parameter_acl_parname_index
        | pg_replication_origin
        | pg_replication_origin_roiident_index
        | pg_replication_origin_roname_index
        | pg_shdepend
        | pg_shdepend_depender_index
        | pg_shdepend_reference_index
        | pg_shdescription
        | pg_shdescription_o_c_index
        | pg_shseclabel
        | pg_shseclabel_object_index
        | pg_subscription
        | pg_subscription_oid_index
        | pg_subscription_subname_index
        | pg_tablespace
        | pg_tablespace_oid_index
        | pg_tablespace_spcname_index
        | pg_toast_1213
        | pg_toast_1213_index
        | pg_toast_1260
```

```
          | pg_toast_1260_index
          | pg_toast_1262
          | pg_toast_1262_index
          | pg_toast_2396
          | pg_toast_2396_index
          | pg_toast_2964
          | pg_toast_2964_index
          | pg_toast_3592
          | pg_toast_3592_index
          | pg_toast_6000
          | pg_toast_6000_index
          | pg_toast_6100
          | pg_toast_6100_index
          | pg_toast_6243
          | pg_toast_6243_index
 zjq75    | pg_toast_16400
          | pg_toast_16400_index
          | test_table1
          | test_table1_pkey
(424 строки)
```

Код

https://github.com/KseniyaNesterenko/rshd_labs/tree/main/lab2

Выводы

В ходе выполнения работы я создала кластер и настроила его по заданию. Также ыбла создана новая бд и произведено её наполнение с помощью новой роли.