Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Распределённые системы хранения данных

Лабораторная работа 2

Вариант 85426, pg121, пользователь postgres2

Выполнил:
Митичев Иван Дмитриевич
Группа:
P3316

Преподаватель:
Николаев Владимир Вячеславович

2025 г.

Санкт-Петербург

Задание

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`
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/ykk83`
- Кодировка: ANSI1251
- Локаль: русская
- Параметры инициализации задать через аргументы команды

Этап 2. Конфигурация и запуск сервера БД

- Способы подключения: 1) Unix-domain сокет в режиме peer; 2) сокет TCP/IP, только localhost
- Номер порта: `9426`
- Способ аутентификации TCP/IP клиентов: по имени пользователя
- 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:
  250 одновременных пользователей, 3 сессий на каждого; каждая сессия инициирует до 10 транзакций на запись размером 24КБ; обеспечить максимальную производительность.

- Директория WAL файлов: `$PGDATA/qfg95`
- Формат лог-файлов: `.csv`

- Уровень сообщений лога: `ERROR`
- Дополнительно логировать: контрольные точки и попытки подключения

Этап 3. Дополнительные табличные пространства и наполнение базы

- На основе шаблона template0 пересоздать базу postgres в новом табличном пространстве: `$HOME/twv39`
- На основе template1 создать новую базу: `lazywhitewood`
- 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.

Ссылка на гитхаб со всеми скриптами/конфигами

Karabas890/RSHDLab2

Подключение

Способ подключения к узлу из сети Интернет через helios:
ssh -J s368527@helios.cs.ifmo.ru:2222 postgres2@pg121
Способ подключения к узлу из сети факультета:

ssh -p 2222 s368527@se.ifmo.ru
ssh postgres2@pg121

Инициализация кластера БД

- Директория кластера: $HOME/ykk83
- Кодировка: ANSI1251
- Локаль: русская
- Параметры инициализации задать через аргументы команды

```
ssh -J s368527@helios.cs.ifmo.ru:2222 postgres2@pg121
PGDATA=$HOME/ykk83
export PGDATA
# Создание каталога под кластер
mkdir -p $HOME/ykk83
#Создаём новую директорию под WAL
mkdir -p $HOME/qfg95
# Инициализация кластера (аргументы команды initdb)
initdb -D "$HOME/ykk83" -X "$HOME/qfg95" --locale=ru_RU.CP1251 --
encoding=WIN1251 --username=postgres2
```

```
#Создаём новую директорию под WAL
mkdir -p $HOME/qfg95
# Инициализация кластера (аргументы команды initdb)
initdb -D "$HOME/ykk83" -X "$HOME/qfg95" --locale=ru_RU.CP1251 --encoding=WIN1251 --username=postgres2
Файлы, относящиеся к этой СУБД, будут принадлежать пользователю "postgres2".
От его имени также будет запускаться процесс сервера.

Кластер баз данных будет инициализирован с локалью "ru_RU.CP1251".
Выбрана конфигурация текстового поиска по умолчанию "russian".

Контроль целостности страниц данных отключён.

исправление прав для существующего каталога /var/db/postgres2/ykk83... ок
исправление прав для существующего каталога /var/db/postgres2/qfg95... ок
создание подкаталогов... ок
выбирается реализация динамической разделяемой памяти... posix
выбирается значение max_connections по умолчанию... 100
выбирается значение shared_buffers по умолчанию... 128MB
выбирается часовой пояс по умолчанию... Europe/Moscow
создание конфигурационных файлов... ок
выполняется подготовительный скрипт... ок
выполняется заключительная инициализация... ок
сохранение данных на диске... ок

initdb: предупреждение: включение метода аутентификации "trust" для локальных подключений
initdb: подсказка: Другой метод можно выбрать, отредактировав pg_hba.conf или ещё раз запустив initdb с ключом -A, --auth-local или --auth-host.

Готово. Теперь вы можете запустить сервер баз данных:

    pg_ctl -D /var/db/postgres2/ykk83 -l файл_журнала start
```

```
pg_ctl -D "$HOME/ykk83" -l "$HOME/ykk83/logfile" start
```

```
[postgres2@pg121 ~]$ pg_ctl -D "$HOME/ykk83" -l "$HOME/ykk83/logfile" start
ожидание запуска сервера.... готово
сервер запущен
[postgres2@pg121 ~]$
```

Конфигурация и запуск сервера БД

- Способы подключения: 1) Unix-domain сокет в режиме peer; 2) сокет TCP/IP, только localhost
- Номер порта: 9426
- Способ аутентификации TCP/IP клиентов: по имени пользователя
- Остальные способы подключений запретить.

postgresql.conf:

```
# Слушать не только localhost, порт 9426
listen_addresses = '*'
port = 9426
# Параметры для OLTP нагрузки
# Одновременных пользователей: 250 * 3 сессии = 750 (макс. соединений)
max_connections = 750
```

pg_hba.conf:

```
# Peer-аутентификация для локального Unix-сокета
local   all             all                                     peer

# 2. TCP/IP подключения только с localhost — md5
host    all             all             127.0.0.1/32            md5
host    all             all             ::1/128                 md5
```

Параметры должны быть подобраны в соответствии со сценарием OLTP:
250 одновременных пользователей, 3 сессий на каждого; каждая сессия инициирует до 10 транзакций на запись размером 24КБ; обеспечить максимальную производительность.

max_connections:

Поскольку всего планируется поддерживать 250 пользователей * 3 сессии = 750 сессий одновременно, я выбрал значение max_connections равным 750.

shared_buffers:

Для shared_buffers рекомендуется использовать 25% от основной памяти сервера (в случаях, когда память сервера больше 1Гб). Возьмем 2Гб.

temp_buffers:

Поскольку в сценарии не планируется использование транзакций, работающих с большими объемами данных 128 MB — хорошо для OLTP с временными операциями.

work_mem:

Точно не известно, на сколько часто будут использоваться запросы с сортировками и хэшированием в транзакциях, однако, лучше взять с небольшим запасом, то есть следующее значение после значения по умолчанию – 16Мб.

checkpoint_timeout:

Так как в сценарии требуется обеспечить максимальную производительность, а увеличение checkpoint_timeout приведет к нагрузке на систему из-за более частых сохранений, оставим значение по умолчанию – 5min.

effective_cache_size:

Для указанной в сценарии высокой производительности выберем effective_cache_size = 10Гб, чтобы больше запросов использовали индексы и выполнялись быстрее.

fsync:

Так как сценарий требует обеспечить максимальную производительность, отключаем fsync. Это ускорит обработку запросов, ведь теперь они не будут принудительно писаться на диск после коммита, то есть увеличится производительность, поскольку запись на диск – небыстрая операция. Однако, в случае сбоя системы, данные окажутся утеряны. Это осознанный риск, ведь про сохранность данных в сценарии ничего не сказано и требование заключается именно в максимальной производительности любыми способами.

commit_delay:

Этот параметр устанавливает задержку перед сохранением в WAL. Таким образом, можно за одно сохранение зафиксировать сразу множество транзакций, увеличив таким образом производительность системы, что и нужно по сценарию. Поставим commit_delay = 10мс.

Директория WAL файлов: `$PGDATA/qfg95`

Задали при инициализации

Формат лог-файлов: `.csv`

Уровень сообщений лога: `ERROR`

Дополнительно логировать: контрольные точки и попытки подключения

```
# Формат лог-файлов
log_destination = 'csvlog'        # логируем в формате CSV
logging_collector = on            # включает сбор логов во внешние фа

# Путь и формат файлов логов
log_directory = 'log'             # директория логов внутри $PGDATA
log_filename = 'postgresql-%a'    # будет создавать файл на каждый де

log_truncate_on_rotation = on     # перезаписывать лог при ротации
log_rotation_age = 1d             # ротация каждый день
log_rotation_size = 0             # ротация не зависит от размера

# Уровень логирования
log_min_messages = error          # логируем только ERROR и выше

# Дополнительное логирование
log_checkpoints = on              # логировать контрольные точки
log_connections = on              # логировать подключения
log_disconnections = on           # логировать отключения
```

Дополнительные табличные пространства и наполнение базы

На основе шаблона template0 пересоздать базу postgres в новом табличном пространстве: `$HOME/twv39`

На основе template1 создать новую базу: `lazywhitewood`

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.

```
# 1. Создаём новое табличное пространство
mkdir -p $HOME/twv39

# Подключаемся к psql как суперпользователь
psql -p 9426 -U postgres2 -d postgres

-- Создаём tablespace в /var/db/postgres2/twv39 (путь должен быть абсолютным
и доступен PostgreSQL серверу)
CREATE TABLESPACE twv39 LOCATION '/var/db/postgres2/twv39';
#Подключаемся к другой базе (например, template1), т.к. нельзя удалить базу,
к которой мы подключёны
psql -p 9426 -U postgres2 -d template1
-- Отключаем подключения к базе postgres (чтобы можно было её пересоздать)
SELECT pid, usename, application_name, client_addr
FROM pg_stat_activity
WHERE datname = 'postgres';
-- сюда вставляем свой pid
SELECT pg_terminate_backend(<pid>);
-- Удаляем старую базу postgres
DROP DATABASE postgres;
-- Создаём новую базу postgres на основе template0 в tablespace twv39
CREATE DATABASE postgres
    TEMPLATE template0
    TABLESPACE twv39;

#Проверка
psql -p 9426 -U postgres2 -d postgres
--Проверяем, что она использует нужное табличное пространство:
SELECT datname, pg_tablespace.spcname AS tablespace
FROM pg_database
         JOIN pg_tablespace ON pg_database.dattablespace = pg_tablespace.oid
WHERE datname = 'postgres';




# Подключаемся к psql как суперпользователь
psql -p 9426 -U postgres2 -d postgres

-- Создаём новую базу lazywhitewood на основе template1
CREATE DATABASE lazywhitewood
    WITH TEMPLATE = template1
    OWNER = postgres2;

-- Создаём новую роль new_role с паролем и правами подключения
CREATE ROLE new_role LOGIN PASSWORD 'new';

-- Даем роли права подключаться к обеим базам
GRANT CONNECT ON DATABASE postgres TO new_role;
```

```
GRANT CONNECT ON DATABASE lazywhitewood TO new_role;

-- Разрешаем роли создавать объекты в tablespace twv39 (если нужно)
GRANT CREATE ON TABLESPACE twv39 TO new_role;
-- Создать таблицу в таблицпространстве postgres
CREATE TABLE test_table_postgres (
                                    id SERIAL PRIMARY KEY,
                                    name TEXT NOT NULL,
                                    value INTEGER
) TABLESPACE twv39;

-- Дать права на вставку и использование sequence роли new_role
GRANT INSERT, SELECT ON test_table_postgres TO new_role;
GRANT USAGE, SELECT ON SEQUENCE test_table_postgres_id_seq TO new_role;

-- Подключиться к базе postgres
psql -h localhost  -p 9426 -U new_role -d postgres
INSERT INTO test_table_postgres (name, value) VALUES
                                                ('A', 1),
                                                ('B', 2),
                                                ('C', 3);
--проверка
SELECT * FROM test_table_lw;

-- Подключиться к базе lazywhitewood (под postgres2)
psql -p 9426 -U postgres2 -d lazywhitewood

-- Создать таблицу в таблицпространстве twv39
CREATE TABLE test_table_lw (
                               id SERIAL PRIMARY KEY,
                               name TEXT NOT NULL,
                               value INTEGER
) TABLESPACE twv39;

-- Дать права на вставку и использование sequence роли new_role
GRANT INSERT, SELECT ON test_table_lw TO new_role;
GRANT USAGE, SELECT ON SEQUENCE test_table_lw_id_seq TO new_role;

-- Вставляем
psql -h localhost -p 9426 -U new_role -d lazywhitewood
INSERT INTO test_table_lw (name, value) VALUES
                                          ('X', 10),
                                          ('Y', 20),
                                          ('Z', 30);
--проверка
SELECT * FROM test_table_lw;



--Выведем 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;
```

Вывод

В ходе выполнения лабораторной работы был создан и сконфигурирован кластер БД на выделенном узле, я познакомился с различными вариантами конфигурации. Также была создана БД, новая роль, табличные пространства и заполнение тестовыми данными.