Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной Инженерии и Компьютерной Техники

Дисциплина: Распределенные системы хранения данных

Лабораторная работа 2

Вариант: 312512

Выполнил:

Гурьянов Кирилл Алексеевич

Группа: P33302

Преподаватель:

Шешуков Дмитрий Михайлович

Санкт-Петербург

2024

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 s312471@helios.cs.ifmo.ru:2222 postgres3@pg198

Способ подключения к узлу из сети факультета:
ssh postgres3@pg198

Номер выделенного узла pg198, а также логин и пароль для подключения Вам выдаст преподаватель.

Обратите внимание, что домашняя директория пользователя /var/postgres/$LOGNAME

Этап 1. Инициализация кластера БД
Директория кластера: $HOME/jcf50
Кодировка: UTF8
Локаль: английская
Параметры инициализации задать через переменные окружения

Этап 2. Конфигурация и запуск сервера БД
Способ подключения: сокет TCP/IP, принимать подключения к любому IP-адресу узла
Номер порта: 9512
Остальные способы подключений запретить.
Способ аутентификации клиентов: по паролю MD5
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 20ГБ, хранение на твердотельном накопителе (SSD).

Директория WAL файлов: $PGDATA/pg_wal
Формат лог-файлов: .csv
Уровень сообщений лога: NOTICE
Дополнительно логировать: завершение сессий и продолжительность выполнения команд

Этап 3. Дополнительные табличные пространства и наполнение базы

Создать новое табличное пространство для индексов: $HOME/zgf65

На основе template1 создать новую базу: lazybluearmy

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.

Цель

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.

Этап 1. Инициализация кластера

```
PGDATA=$HOME/jcf50
PGLOCALE=en_US.UTF-8
PGENCODE=UTF8
```

```
PGUSERNAME=postgres3
export PGDATA PGLOCALE PGENCODE PGUSERNAME
env | sort
...
PGDATA=/var/db/postgres3/jcf50
PGENCODE=UTF8
PGLOCALE=en_US.UTF-8
...
mkdir -p $PGDATA

initdb --pgdata=$PGDATA --locale=$PGLOCALE --encoding=$PGENCODE
--username=$PGUSERNAME
```

Файлы, относящиеся к этой СУБД, будут принадлежать пользователю "postgres3".
От его имени также будет запускаться процесс сервера.

Кластер баз данных будет инициализирован с локалью "en_US.UTF-8".
Выбрана конфигурация текстового поиска по умолчанию "english".

Контроль целостности страниц данных отключён.

исправление прав для существующего каталога /var/db/postgres3/jcf50... 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... W-SU
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
Другой метод можно выбрать, отредактировав pg_hba.conf или используя ключи -A,
--auth-local или --auth-host при следующем выполнении initdb.

Готово. Теперь вы можете запустить сервер баз данных:

pg_ctl -D /var/db/postgres3/jcf50 -l файл_журнала start

Этап 2. Конфигурация и запуск сервера БД

```
pg_ctl -D /var/db/postgres3/jcf50 -l logfile start
```

```
ожидание запуска сервера.... готово
сервер запущен
```

pg_hba.conf

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     reject
# IPv4 local connections:
host    all             all             all                     md5
# IPv6 local connections:

host    all             all             ::1/128                 reject
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     reject
host    replication     all             127.0.0.1/32            reject
host    replication     all             ::1/128                 reject
```

postgresql.conf

Устанавливаем TCP-порт открываемый сервером. Этот порт используется
для всех IP-адресов, через которые сервер принимает подключения.

port = 9512

Определяет максимальное число одновременных подключений к серверу БД. По умолчанию обычно это 100 подключений, но это число может быть меньше, если ядро накладывает свои ограничения.

max_connections = 100

Устанавливаем алгоритм шифрования паролей. Когда в CREATE ROLE или ALTER ROLE задается пароль, этот параметр определяет, каким алгоритмом его шифровать. Возможные значения данного параметра — scram-sha-256 (пароль будет шифроваться алгоритмом SCRAM-SHA-256) и md5 (пароль сохраняется в виде хеша MD5).
password_encryption = md5

Необходимо установить значений shared_buffers, который задаёт объём памяти, который будет использовать сервер баз данных для буферов в разделяемой памяти. Значение shared_buffers советуется устанавливать в 25% от ОЗУ. Однако установка shared_buffers в 5 ГБ приводит в следующей ошибку при попытке подключиться через psql:

```
[postgres3@pg198 ~/jcf50]$ psql -p 9512 -d postgres -h localhost
psql: ошибка: подключиться к серверу "localhost" (::1), порту 9512 не удалось: could not fork
 new process for connection: Resource temporarily unavailable
```

Таким образом, если рассмотреть гипотетическую систему, в которой у нас реально имеется 20 ГБ ОЗУ, то shared_buffers стоит установить в 5 ГБ, однако в наших реалиях, установить его более чем в 1.5 ГБ не удалось.

shared_buffers = 1535MB

Temp_buffers задаёт максимальный объём памяти, выделяемой для временных буферов в каждом сеансе. Эти существующие только в рамках сеанса буферы используются исключительно для работы с временными таблицами. Данное значение не стоит делать слишком большим для избежания неэффективного использования памяти. Но и вычислить значение довольно сложно. Так как от этого параметра зависит сколько памяти будет потреблять каждое подключение, то этот параметр лучше всего эмпирически регулировать в ходе работы в зависимости от свободной памяти.

temp_buffers = 32MB

Work_mem задаёт базовый максимальный объём памяти, который будет использоваться во внутренних операциях при обработке запросов (например, для сортировки или хеш-таблиц), прежде чем будут задействованы временные файлы на диске. Таким образом, общий объем памяти может многократно превосходить значение work_mem. Это следует учитывать, выбирая подходящее значение. При превышении этого параметра будет использовано временное пространство, расположенное на диске – запросы будут выполняться медленнее.

work_mem = 128MB

Fsync. Если этот параметр установлен, сервер старается добиться, чтобы изменения были записаны на диск физически, выполняя системные вызовы fsync(). Это даёт гарантию, что кластер баз данных сможет вернуться в согласованное состояние после сбоя оборудования или операционной системы.

Хотя отключение fsync часто даёт выигрыш в скорости, это может привести к неисправимой порче данных в случае отключения питания или сбоя системы. Поэтому отключать fsync рекомендуется, только если вы легко сможете восстановить всю базу из внешнего источника.

Думаю, что риск потери данных превышает выигрыш от от прироста производительности. Таким образом устанавливаем данный параметр в on.

fsync = on

Параметр commit_delay добавляет паузу перед собственно выполнением сохранения WAL. Эта задержка может увеличить быстродействие при фиксировании множества транзакций, позволяя зафиксировать большее число транзакций за одну операцию сохранения WAL, если система загружена достаточно сильно и за заданное время успевают зафиксироваться другие транзакции. Однако этот параметр также увеличивает задержку максимум до commit_delay при каждом сохранении WAL. Эта задержка окажется бесполезной, если никакие другие транзакции не будут зафиксированы за это время.

Изменять данное значение имеет смысл только после тестирования нашей системы на реальных пользователях, чтобы понять, насколько велика нагрузка транзакциями на нашу систему. Для начала устанавливаем в 0.

commit_delay = 0

checkpoint_timeout – это максимальное время между автоматическими контрольными точками в WAL. Значение по умолчанию — пять минут (5min). Увеличение этого параметра может привести к увеличению времени, которое потребуется для восстановления после сбоя. Значение по умолчанию должно отлично подойти.

checkpoint_timeout = 5min

effective_cache_size определяет представление планировщика об эффективном размере дискового кеша, доступном для одного запроса. 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. Это всего лишь ориентир, а не точный объем выделенной памяти или кеша. Он не выделяет фактическую память, но сообщает оптимизатору объем кеша, доступный в ядре. Данное значение советуется устанавливать не меньше половины доступной ОЗУ. Т.е. для моего варианта данное значение подойдет 10 GB, однако в связи с тем, что реальной памяти у меня меньше, то устанавливаем его в значение:

effective_cache_size = 1GB

Когда параметр archive_mode включён, полные сегменты WAL передаются в хранилище архива командой archive_command. Если значение данного параметра off, то archive_command не выполняется.

archive_mode = on

Команда локальной оболочки, которая будет выполняться для архивации завершенного сегмента WAL. Любое вхождение %p в этой строке заменяется путем архивируемого файла.

archive_command = 'cp %p $PGDATA/pg_wal'

Postgres поддерживает несколько методов протоколирования сообщений сервера: stderr, csvlog, jsonlog и syslog. Если в log_destination включено значение csvlog, то протоколирование ведется в формате CSV (разделенные запятыми значения). Это удобно для программной обработки журнала.

log_destination = 'csvlog'

Параметр включает сборщик сообщений (logging collector). Это фоновый процесс, который собирает отправленные в stderr сообщения и перенаправляет их в журнальные файлы.

logging_collector = on

Управляет минимальным уровнем сообщений, записываемых в журнал сервера.

log_min_messages = notice

Включает протоколирование завершения сеанса. В журнал выводится примерно та же информация, что и с log_connections, плюс длительность сеанса.

log_disconnections = on

Записывает продолжительность каждой завершенной команды.

log_duration = on

Этап 3. Дополнительные табличные пространства и наполнение базы

Создание табличного пространства для индексов

```
mkdir $HOME/zgf65
CREATE TABLESPACE indexspace LOCATION '/var/db/postgres3/zgf65';
CREATE TABLESPACE

\db
          Список табличных пространств
   Имя      | Владелец  |       Расположение
------------+-----------+------------------------
indexspace | postgres3 | /var/db/postgres3/zgf65
pg_default | postgres3 |
```

```
pg_global   | postgres3 |
(3 строки)
```

Создание новой роли и предоставление прав

```
CREATE DATABASE lazybluearmy WITH TEMPLATE = template1;
CREATE DATABASE
postgres=# \l
                                    Список баз данных
    Имя       | Владелец  | Кодировка | LC_COLLATE  |  LC_CTYPE   |
Права доступа
--------------+-----------+-----------+-------------+-------------+-----
--------------------
lazybluearmy | postgres3 | UTF8      | en_US.UTF-8 | en_US.UTF-8 |
postgres     | postgres3 | UTF8      | en_US.UTF-8 | en_US.UTF-8 |
template0    | postgres3 | UTF8      | en_US.UTF-8 | en_US.UTF-8 |
=c/postgres3           +
             |           |           |             |             |
postgres3=CTc/postgres3
template1    | postgres3 | UTF8      | en_US.UTF-8 | en_US.UTF-8 |
=c/postgres3           +
             |           |           |             |             |
postgres3=CTc/postgres3
(4 строки)
```

```
\c lazybluearmy
```
Вы подключены к базе данных "lazybluearmy" как пользователь "postgres3".
```
CREATE TABLE army (id SERIAL PRIMARY KEY, country VARCHAR(50), size
int);
CREATE TABLE
CREATE INDEX ON army(size) TABLESPACE indexspace;
CREATE INDEX
CREATE ROLE korako LOGIN PASSWORD 'qwerty';
CREATE ROLE
GRANT INSERT ON army TO korako;
GRANT
```

Наполнение баз тестовыми данными

```
psql -p 9512 -d lazybluearmy -h localhost -U korako
insert into army (country, size) values ('Новая Зеландия', 10000);
```

```
ERROR:  permission denied for sequence army_id_seq
psql -p 9512 -d lazybluearmy -h localhost
GRANT USAGE ON SEQUENCE army_id_seq TO korako;
GRANT
psql -p 9512 -d lazybluearmy -h localhost -U korako

\i insert.sql
INSERT 0 1
INSERT 0 1
...
INSERT 0 1
SELECT * FROM army;
ERROR:  permission denied for table army

psql -p 9512 -d lazybluearmy -h localhost
SELECT * FROM army;
id  |                   country                    |  size
-----+----------------------------------------------+---------
  1 | Китай                                        | 2035000
  2 | Индия                                        | 1460350
  3 | Соединённые Штаты Америки                    | 1395350
  4 | Корейская Народно-Демократическая Республика | 1280000
  5 | Россия                                       |  900000
...
```

Вывод табличных пространств кластеры и содержащихся в них объектов

```
SELECT * FROM pg_tablespace;
 oid  |  spcname   | spcowner | spcacl | spcoptions
-------+------------+----------+--------+------------
 1663 | pg_default |       10 |        |
 1664 | pg_global  |       10 |        |
16384 | indexspace |       10 |        |
(3 строки)

SELECT c.relname, t.spcname FROM pg_class c JOIN pg_tablespace t ON
c.reltablespace = t.oid;
                relname                 |  spcname
----------------------------------------+------------
army_size_idx                           | indexspace
pg_toast_1262                           | pg_global
pg_toast_1262_index                     | pg_global
```

```
pg_toast_2964                           | pg_global
pg_toast_2964_index                     | pg_global
pg_toast_1213                           | pg_global
pg_toast_1213_index                     | pg_global
pg_toast_1260                           | pg_global
pg_toast_1260_index                     | pg_global
pg_toast_2396                           | pg_global
pg_toast_2396_index                     | pg_global
pg_toast_6000                           | pg_global
pg_toast_6000_index                     | pg_global
pg_toast_3592                           | pg_global
pg_toast_3592_index                     | pg_global
pg_toast_6100                           | pg_global
pg_toast_6100_index                     | pg_global
pg_database_datname_index               | pg_global
pg_database_oid_index                   | pg_global
pg_db_role_setting_databaseid_rol_index | pg_global
pg_tablespace_oid_index                 | pg_global
pg_tablespace_spcname_index             | pg_global
pg_authid_rolname_index                 | pg_global
pg_authid_oid_index                     | pg_global
pg_auth_members_role_member_index       | pg_global
pg_auth_members_member_role_index       | pg_global
pg_shdepend_depender_index              | pg_global
pg_shdepend_reference_index             | pg_global
pg_shdescription_o_c_index              | pg_global
pg_replication_origin_roiident_index    | pg_global
pg_replication_origin_roname_index      | pg_global
pg_shseclabel_object_index              | pg_global
pg_subscription_oid_index               | pg_global
pg_subscription_subname_index           | pg_global
pg_authid                               | pg_global
pg_subscription                         | pg_global
pg_database                             | pg_global
pg_db_role_setting                      | pg_global
pg_tablespace                           | pg_global
pg_auth_members                         | pg_global
pg_shdepend                             | pg_global
pg_shdescription                        | pg_global
pg_replication_origin                   | pg_global
pg_shseclabel                           | pg_global
(44 строки)
```

Вывод

В ходе выполнения лабораторной работы я познакомился с системными каталогами Postgresql, научился составлять запросы к системным каталогам с целью получения информации о таблицах, создал процедуру, которая по заданному названию таблицы и пользователю получит информацию обо всех столбцах таблицы.