МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное образовательное учреждение высшего образования “Национальный исследовательский университет ИТМО”

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №4

по дисциплине

“Распределённые системы хранения данных”

вариант 64348

Выполнили:

Иванов Матвей Сергеевич

Шульга Артём Игоревич

группа P33111

Преподаватель:

Николаев Владимир Вячеславович

г. Санкт-Петербург, 2024

Задание

Требования к выполнению работы

- В качестве хостов использовать одинаковые виртуальные машины.
- В первую очередь необходимо обеспечить сетевую связность между ВМ.
- Для подключения к СУБД (например, через psql), использовать отдельную виртуальную или физическую машину.
- Демонстрировать наполнение базы и доступ на запись на примере не менее, чем двух таблиц, столбцов, строк, транзакций и клиентских сессий.

Этап 1. Конфигурация

Настроить репликацию postgres на трёх узлах: A - основной, B и C - резервные. Для управления использовать pgpool-II. Репликация с A на B синхронная. Репликация с A на C асинхронная. Продемонстрировать, что новые данные реплицируются на B в синхронном режиме, а на C с задержкой.

Этап 2. Симуляция и обработка сбоя

2.1 Подготовка:

- Установить несколько клиентских подключений к СУБД.
- Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

2.2 Сбой:

Симулировать программную ошибку на основном узле - выполнить команду pkill -9 postgres.

2.3 Обработка:

- Найти и продемонстрировать в логах релевантные сообщения об ошибках.
- Выполнить переключение (failover) на резервный сервер.
- Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

Этап 3. Восстановление

- Восстановить работу основного узла - откатить действие, выполненное с виртуальной машиной на этапе 2.2.
- Актуализировать состояние базы на основном узле - накатить все изменения данных, выполненные на этапе 2.3.
- Восстановить исправную работу узлов в исходной конфигурации (в соответствии с этапом 1).
- Продемонстрировать состояние данных и работу клиентов в режиме чтение/запись.

Ход выполнения

Этап 1. Конфигурация

Создаём пользователя на master узле для репликации:

```
CREATE USER PHYSREPL WITH REPLICATION
```

Для начала добавим конфигурации в postgresql.conf на Master узле:
Установим минимальный уровень WAL на реплику
```
wal_level = replica
```

Включим для СУБД режим архивирования WAL
```
archive_mode = on
```

Настроим копирование WAL (scp) на резервный узел
```
archive_command = 'cp %p /archive/%f'
```

Максимальное количество WAL отправителей
```
max_wal_senders=5
```

Максимальное количество WAL для хранения в Master
```
wal_keep_size=50
```

Названия application для синхронной репликации
```
synchronous_standby_names = 'postgres3_repl'
```

```
pg_hba.conf
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    replication     physrepl             all                trust
```

```
pg_basebackup -R -h localhost -U physrepl -D postgres1-backup -P -p 6000
```

Slave узлы восстанавливаем из нашего `postgres1-backup`

Slave узел для асинхронной репликации:

Добавляем `standby.signal`

В postgresql.conf добавляем строку:

```
primary_conninfo = 'host=postgres1 port=5432 dbname=postgres
user=physrepl application_name=postgres2_repl'
```

Slave узел для синхронной репликации:

Добавляем `standby.signal`

В postgresql.conf добавляем строку:

```
primary_conninfo = 'host=postgres1 port=5432 dbname=postgres
user=physrepl application_name=postgres3_repl'
```

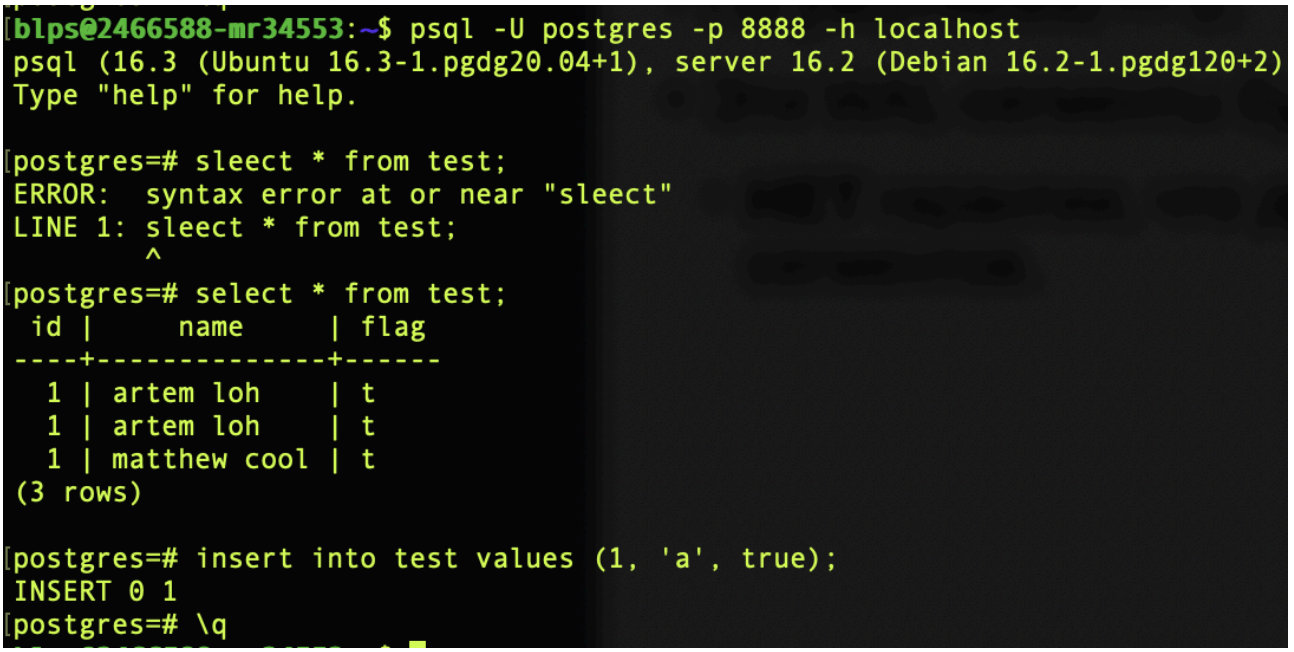
Docker Compose файл с запуском pgpool2 и Postgres кластеров.

```
version: "3"
services:
  postgres1:
    image: postgres
    ports:
      - "6000:5432"
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: mysecretpassword
      POSTGRES_DB: rshd
    volumes:
      - ./postgres1:/var/lib/postgresql/data
  postgres2:
    image: postgres
    ports:
      - "7000:5432"
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: mysecretpassword
      POSTGRES_DB: rshd
    volumes:
      - ./postgres2:/var/lib/postgresql/data
  postgres3:
    image: postgres
    ports:
      - "8000:5432"
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: mysecretpassword
      POSTGRES_DB: rshd
    volumes:
      - ./postgres3:/var/lib/postgresql/data
```

```
pgpool:
  image: docker.io/bitnami/pgpool:4
  ports:
    - "8888:5432"
  environment:
    PGPOOL_BACKEND_NODES: 0:postgres1,1:postgres2,2:postgres3
    PGPOOL_SR_CHECK_USER: postgres
    PGPOOL_SR_CHECK_PASSWORD: mysecretpassword
    PGPOOL_POSTGRES_USERNAME: postgres
    PGPOOL_POSTGRES_PASSWORD: mysecretpassword
    PGPOOL_ADMIN_USERNAME: admin
    PGPOOL_ADMIN_PASSWORD: mysecretpassword
    PGPOOL_POSTGRES_CUSTOM_USERS: customuser
    PGPOOL_POSTGRES_CUSTOM_PASSWORDS: custompassword
    PGPOOL_ENABLE_LOAD_BALANCING: 'yes'
    PGPOOL_FAILOVER_ON_BACKEND_ERROR: 'on'
  healthcheck:
    test: ["CMD", "/opt/bitnami/scripts/pgpool/healthcheck.sh"]
    interval: 10s
    timeout: 5s
    retries: 5
```

Этап 2. Симуляция и обработка сбоя

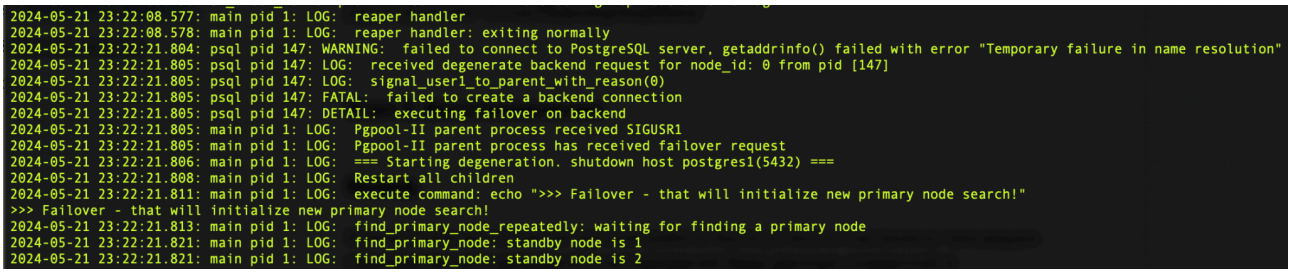
Создаём таблицы в базе данных, добавляем тестовые данные, проверяем, что данные синхронизируются.



Можно проверить, когда была последняя репликация и в каком режиме они работают (sync/async):

select * from pg_stat_replication;

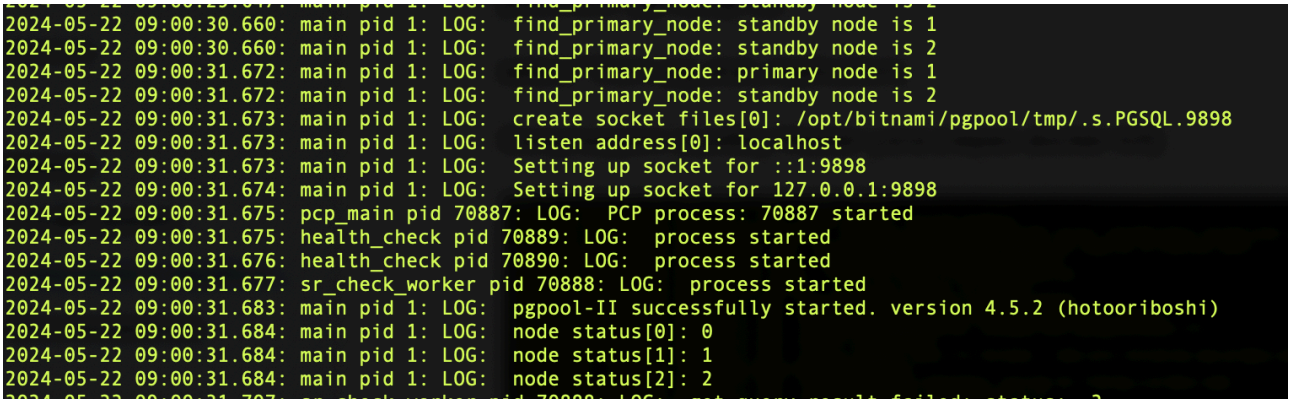
Симулируем ошибку Master-узла (через docker kill):



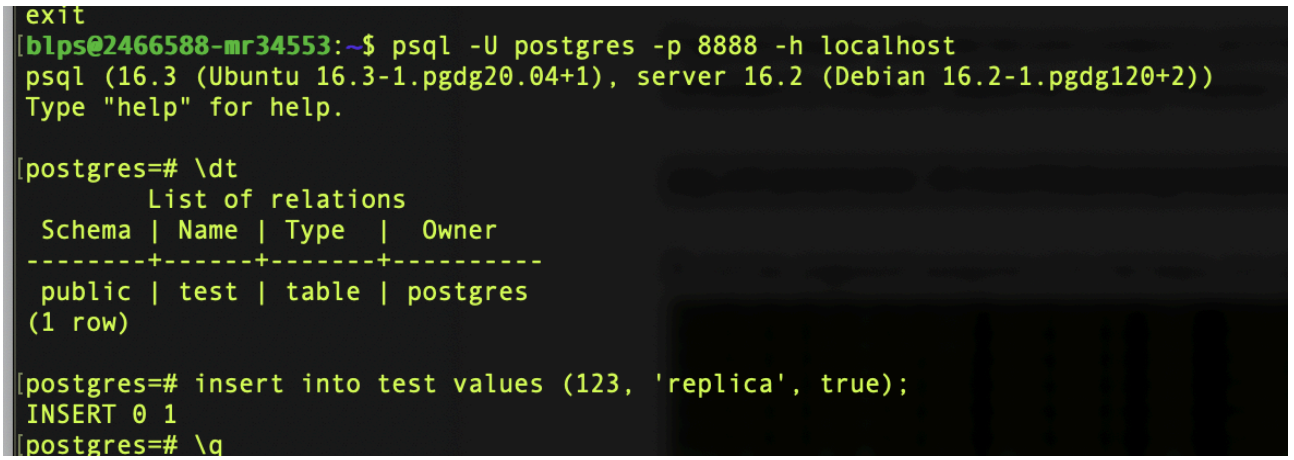
Pgpool ждёт объявления нового primary узла (пока только 2 standby). Чтобы объявить новый (временный) Master на ноде postgres2 (или postgres3) вызываем:

pg_ctl promote -D /var/lib/postgresql/data/

В логах pgpool видим что теперь он primary:



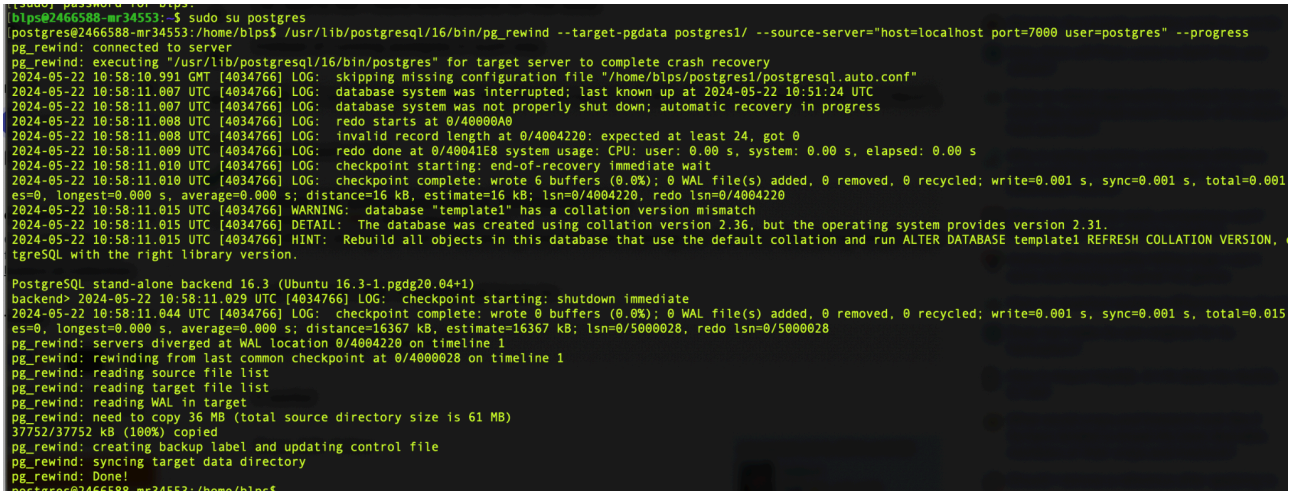
Пробуем подключиться и прочитать/добавить новые данные:



Этап 3. Восстановление

Для восстановления работоспособности перемотаем WALы:

/usr/lib/postgresql/16/bin/pg_rewind --target-pgdata postgres1/
--source-server="host=localhost port=7000 user=postgres" --progress

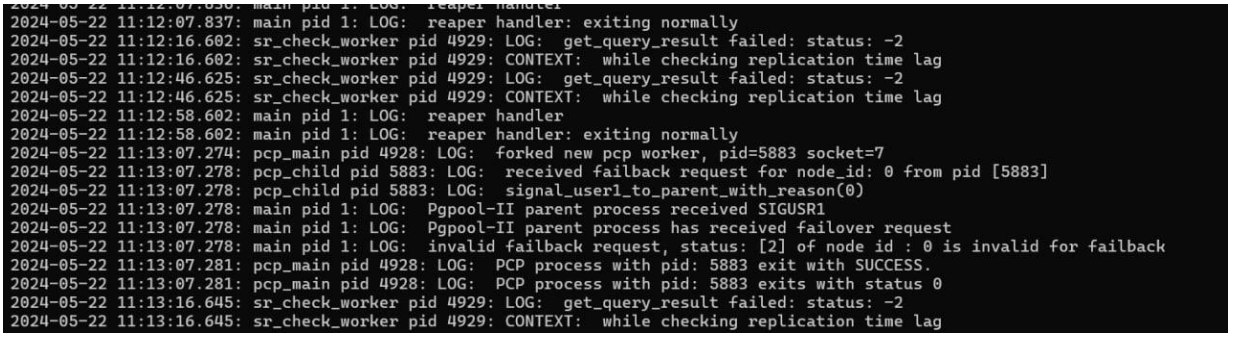


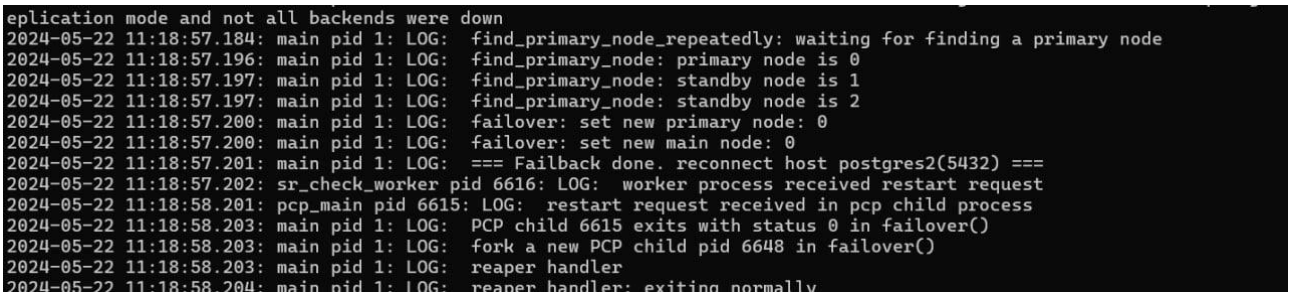
Теперь запустим postgres1.

Добавим обратно ноду в pgpool:

pcp_attach_node -h localhost -p 9898 -U admin -W -n 0

У теперь 2 primary ноды, нам нужно вернуть postgres2 в standby. Для этого добавляем standby.signal и перезапускаем postgres2.





Вывод

В результате выполнения данной работы была успешно настроена процедура синхронной и асинхронной репликации с Master узла на Slave узлы, а также в качестве управления узлам настроен pgpool II. Это позволило обеспечить сохранность данных и снизить риски потери информации, а также гарантировать высокую доступность базы данных и возможность быстрого восстановления при потере одного из узлов.