МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

##### ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

**ЛАБОРАТОРНАЯ РАБОТА №4**

**по дисциплине**

**«Распределённые системы хранения данных»**

Вариант №35298

##### ***Выполнил:***

##### Студенты группы P3334

#### Баянов Равиль

#### Кузнецов Даниил

#### Преподаватель:

##### Николаев Владимир

##### Вячеславович

Санкт-Петербург

2025

**Содержание**

[**Задание** 3](#_Toc198505745)

[**Описание этапов выполнения** 4](#_Toc198505746)

[Этап 1. Конфигурация 4](#_Toc198505747)

[Этап 2. Симуляция и обработка сбоя 5](#_Toc198505748)

[Восстановление 6](#_Toc198505749)

[**Сложности** 7](#_Toc198505750)

[**Вывод** 8](#_Toc198505751)

# **Задание**

Цель работы - ознакомиться с методами и средствами построения отказоустойчивых решений на базе СУБД Postgres; получить практические навыки восстановления работы системы после отказа.

Работа рассчитана на двух человек и выполняется в три этапа: настройка, симуляция и обработка сбоя, восстановление.

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.

# **Описание этапов выполнения**

## Этап 1. Конфигурация

**Задание:**

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

**Выполнение:**

Для начала установим KVM и libvirt:

|  |
| --- |
| sudo apt update  sudo apt install qemu-kvm libvirt-daemon-system libvirt-clients virtinst bridge-utils  sudo usermod -aG libvirt $(whoami)  newgrp libvirt |

Настройка DHCP для виртуальных машин

Сохраняем в файле internal-net.xml

|  |
| --- |
| <**network**>  <**name**>internal-net</**name**>  <**bridge** name='virbr10' stp='on' delay='0'/>  <**ip** address='192.168.100.1' netmask='255.255.255.0'>  <**dhcp**>  <**range** start='192.168.100.100' end='192.168.100.200'/>  </**dhcp**>  </**ip**>  </**network**> |

Создадим диски

|  |
| --- |
| sudo qemu-img create -f qcow2 /var/lib/libvirt/images/postgres-A.qcow2 10G  sudo qemu-img create -f qcow2 /var/lib/libvirt/images/postgres-B.qcow2 10G  sudo qemu-img create -f qcow2 /var/lib/libvirt/images/postgres-C.qcow2 10G  sudo qemu-img create -f qcow2 /var/lib/libvirt/images/postgres-client.qcow2 10G |

Установим виртуальные машины

|  |
| --- |
| virt-install \  --name postgres-A \  --ram 2048 \  --vcpus 2 \  --disk path=/var/lib/libvirt/images/postgres-A.qcow2,format=qcow2 \  --os-variant ubuntu24.04 \  --cdrom /home/kuznecov/ITMO/DDB/Lab4/ubuntu-24.04.2-live-server-amd64.iso \  --network network=internal-net \  --graphics vnc  virt-install \  --name postgres-B \  --ram 2048 \  --vcpus 2 \  --disk path=/var/lib/libvirt/images/postgres-B.qcow2,format=qcow2 \  --os-variant ubuntu24.04 \  --cdrom /home/kuznecov/ITMO/DDB/Lab4/ubuntu-24.04.2-live-server-amd64.iso \  --network network=internal-net \  --graphics vnc  virt-install \  --name postgres-C \  --ram 2048 \  --vcpus 2 \  --disk path=/var/lib/libvirt/images/postgres-C.qcow2,format=qcow2 \  --os-variant ubuntu24.04 \  --cdrom /home/kuznecov/ITMO/DDB/Lab4/ubuntu-24.04.2-live-server-amd64.iso \  --network network=internal-net \  --graphics vnc  virt-install \  --name postgres-client \  --ram 2048 \  --vcpus 2 \  --disk path=/var/lib/libvirt/images/postgres-client.qcow2,format=qcow2 \  --os-variant ubuntu24.04 \  --cdrom /home/kuznecov/ITMO/DDB/Lab4/ubuntu-24.04.2-live-server-amd64.iso \  --network network=internal-net \  --graphics vnc |

Настройка кластеров

**Сервер A (192.168.122.253)**

Оставим сервер сначала: *sudo systemctl stop postgresql*

*postgresql.conf*

|  |
| --- |
| listen\_addresses = '\*'  wal\_level = replica  max\_wal\_senders = 10  synchronous\_standby\_names = 'postgres2'  wal\_keep\_size = 64  archive\_mode = on  archive\_command = 'cp %p /var/lib/postgresql/16/main/archive/%f' |

*sudo mkdir -p /var/lib/postgresql/16/main/archive*

*sudo chown postgres:postgres /var/lib/postgresql/16/main/archive*

*pg\_hba.conf*

|  |
| --- |
| host replication replicator 192.168.122.198/32 md5 |

Создадим роль replicator для восстановления

|  |
| --- |
| **CREATE** ROLE replicator **WITH** REPLICATION LOGIN ENCRYPTED PASSWORD 'replicator\_pass'; |

Перезапустим сервер

*sudo systemctl restart postgresql*

**Сервер B (192.168.122.198)**

Остановим сервер сначала: *sudo systemctl stop postgresql*

Очистим также каталог кластера и создадим бэкап

|  |
| --- |
| sudo rm -rf /var/lib/postgresql/16/main  sudo mkdir /var/lib/postgresql/16/main  sudo chown -R postgres:postgres /var/lib/postgresql/16/main  sudo chmod -R 700 /var/lib/postgresql/16/main  sudo pg\_basebackup -h 192.168.122.253 -D /var/lib/postgresql/16/main -U replicator -Fp -Xs -P -R |

*postgresql.conf*

|  |
| --- |
| listen\_addresses = '\*'  primary\_conninfo = 'host=192.168.122.253 port=5432 user=replicator password=replicator\_pass application\_name=postgres2'  synchronous\_standby\_names = ''  hot\_standby = on |

*pg\_hba.conf*

|  |
| --- |
| host replication replicator 192.168.122.192/32 md5 |

Запустим сервер: *sudo systemctl start postgresql*

Сервер C (192.168.122.192)

Остановим сервер сначала: *sudo systemctl stop postgresql*

Очистим каталог кластера и создадим бэкап

|  |
| --- |
| sudo rm -rf /var/lib/postgresql/16/main  sudo mkdir /var/lib/postgresql/16/main  sudo chown -R postgres:postgres /var/lib/postgresql/16/main  sudo chmod -R 700 /var/lib/postgresql/16/main  sudo pg\_basebackup -h 192.168.122.198 -D /var/lib/postgresql/16/main -U replicator -Fp -Xs -P -R |

*postgresql.conf*

|  |
| --- |
| listen\_addresses = '\*'  primary\_conninfo = 'host=192.168.122.198 port=5432 user=replicator password=replicator\_pass application\_name=postgres3'  hot\_standby = on |

*pg\_hba.conf*

|  |
| --- |
| local all postgres md5  local all all md5 |

Проверим репликацию на postgres1

*SELECT \* FROM pg\_stat\_replication;*

Видим, что репликация с postgres2 не синхронная и имя сервера не совпадает, зададим параметры вручную.

|  |
| --- |
| **ALTER** **SYSTEM** **SET** primary\_conninfo = 'user=replicator password=replicator\_pass host=192.168.122.253 port=5432 application\_name=postgres2'; |

Настроим теперь сервер Client для pgpool-II (192.168.122.192)

*sudo apt install pgpool2 -y*

*pgpool.conf*

|  |
| --- |
| backend\_hostname0 = '192.168.122.253'  backend\_port0 = 5432  backend\_weight0 = 1  backend\_data\_directory0 = '/var/lib/postgresql/16/main'  backend\_flag0 = 'ALWAYS\_PRIMARY'  backend\_hostname1 = '192.168.122.198'  backend\_port1 = 5432  backend\_weight1 = 1  backend\_data\_directory1 = '/var/lib/postgresql/16/main'  backend\_flag1 = 'ALLOW\_TO\_FAILOVER'  backend\_hostname2 = '192.168.122.192'  backend\_port2 = 5432  backend\_weight2 = 1  backend\_data\_directory2 = '/var/lib/postgresql/16/main'  backend\_flag2 = 'DISALLOW\_TO\_FAILOVER'  enable\_pool\_hba = on  pool\_passwd = 'pool\_passwd'  load\_balance\_mode = off  master\_slave\_mode = on  master\_slave\_sub\_mode = 'stream' |

*postgresql.conf*

|  |
| --- |
| health\_check\_period = 10  health\_check\_timeout = 20  health\_check\_user = 'postgres'  health\_check\_password = 'postgres'  health\_check\_database = 'postgres'  logging\_collector = on  log\_directory = '/tmp/pgpool\_logs'  log\_filename = 'pgpool-%Y-%m-%d\_%H%M%S.log' |

Узнаем IP на клиентской машине

|  |
| --- |
| ip a | grep inet  inet 127.0.0.1/8 scope host lo  inet6 ::1/128 scope host noprefixroute  inet 192.168.122.34/24 metric 100 brd 192.168.122.255 scope global dynamic enp1s0  inet6 fe80::5054:ff:fede:969b/64 scope link |

Настроим пароли для pgpool

|  |
| --- |
| sudo chown postgres:postgres /etc/pgpool2/pool\_passwd  sudo chmod 640 /etc/pgpool2/pool\_passwd  sudo chmod 644 /etc/pgpool2/pgpool.conf  sudo pg\_md5 --md5auth --username=postgres postgres  sudo cat /etc/pgpool2/pool\_passwd  postgres:md53175bce1d3201d16594cebf9d7eb3f9d |

Для всех машин разрешим подключение с клиента

Поменяем аутентификацию на всех нодах на md5, а также добавим строку для подключения с сервера pgpool

*postgresql.conf*

|  |
| --- |
| password\_encryption = md5 |

*pg\_hba.conf*

|  |
| --- |
| host all all 192.168.122.34/32 md5 |

Проверим статус pgpool

*psql -h localhost -p 9999 -U postgres -c "show pool\_nodes;"*

Проверим репликацию

Присоединимся к БД через pgpool и создадим таблицы с данными

*psql -p 9999 -U postgres*

|  |
| --- |
| **CREATE** **TABLE** products (  id SERIAL **PRIMARY** KEY,  name TEXT **NOT** **NULL**,  price NUMERIC  );  **CREATE** **TABLE**  postgres=# **CREATE** **TABLE** orders (  id SERIAL **PRIMARY** KEY,  product\_id INT **REFERENCES** products(id),  quantity INT,  created\_at TIMESTAMP **DEFAULT** now()  );  **CREATE** **TABLE**  postgres=# **BEGIN**;  **BEGIN**  postgres=\*# **INSERT** **INTO** products (name, price) **VALUES**  ('Laptop', 1200.00),  ('Phone', 600.00);  **INSERT** 0 2  postgres=\*# **INSERT** **INTO** orders (product\_id, quantity) **VALUES**  (1, 2),  (2, 1);  **INSERT** 0 2  postgres=\*# **COMMIT**;  **COMMIT** |

Убедимся, что на репликах всё продублировано

Убеждаемся в задержке репликации

Проверка синхронности реплик

|  |
| --- |
| **SELECT**  application\_name,  state,  sync\_state  **FROM** pg\_stat\_replication;  application\_name | state | sync\_state  ------------------+-----------+------------  postgres3 | streaming | async  (1 row) |

Узел A:

|  |
| --- |
| **SELECT**  application\_name,  client\_addr,  state,  sync\_state,  sent\_lsn,  write\_lsn,  flush\_lsn,  replay\_lsn,  pg\_wal\_lsn\_diff(sent\_lsn, replay\_lsn) **AS** replication\_lag\_bytes,  replay\_lag  **FROM** pg\_stat\_replication;  application\_name | client\_addr | state | sync\_state | sent\_lsn | write\_lsn | flush\_lsn | replay\_lsn | replication\_lag\_bytes | replay\_lag  ------------------+-----------------+-----------+------------+------------+------------+------------+------------+-----------------------+------------  postgres2 | 192.168.122.198 | streaming | sync | 0/29390EE0 | 0/29390EE0 | 0/29390EE0 | 0/29390EE0 | 0 |  (1 row) |

Узел B

|  |
| --- |
| **SELECT**  application\_name,  client\_addr,  state,  sync\_state,  sent\_lsn,  write\_lsn,  flush\_lsn,  replay\_lsn,  pg\_wal\_lsn\_diff(sent\_lsn, replay\_lsn) **AS** replication\_lag\_bytes,  replay\_lag  **FROM** pg\_stat\_replication;  application\_name | client\_addr | state | sync\_state | sent\_lsn | write\_lsn | flush\_lsn | replay\_lsn | replication\_lag\_bytes | replay\_lag  ------------------+-----------------+-----------+------------+------------+------------+------------+------------+-----------------------+-----------------  postgres3 | 192.168.122.192 | streaming | async | 0/29390EA8 | 0/29390EA8 | 0/29390EA8 | 0/29390EA8 | 0 | 00:00:00.041515  (1 row) |

## Этап 2. Симуляция и обработка сбоя

**Задание:**

2.1 Подготовка:

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

2.2 Сбой:

Симулировать переполнение дискового пространства на основном узле - заполнить всё свободное пространство раздела с PGDATA “мусорными” файлами.

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.

**Выполнение:**

|  |
| --- |
| sudo -u postgres -s  cd /var/lib/postgresql/16/main  dd if=/dev/zero **of**=diskfill bs=1M status=progress |

Видим, что основной узел отказал

Видим сообщения об ошибках в /var/lib/postgresql/16/main/log

Failover:

Выдвигаем сервер B:

sudo -u postgres /usr/lib/postgresql/16/bin/pg\_ctl -D /var/lib/postgresql/16/main promote

pcp\_detach\_node -h localhost -p 9898 -U postgres -n 0

pcp\_promote\_node -h localhost -p 9898 -U postgres -n 1

sudo systemctsl restart pgpool2

Попробуем создать таблицу на клиенте – всё успешно.

## Восстановление

**Задание:**

* 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.

**Выполнение:**

Удалим мусорные файлы

sudo rm /var/lib/postgresql/16/main/diskfill

tekassh1@postgres1:~$ sudo -u postgres rm -rf /var/lib/postgresql/16/main

tekassh1@postgres1:~$ sudo -u postgres mkdir /var/lib/postgresql/16/main

Делаем бекап  
sudo -u postgres pg\_basebackup -h 192.168.122.198 -D /var/lib/postgresql/16/main -U replicator -P -R

Запускаем сервер : sudo systemctl start postgresql

сейчас в режиме реплики

/var/lib/postgresql/16/main/standby.signal

SELECT pg\_is\_in\_recovery();

Делаем основной сервер primary:

**sudo -u postgres /usr/lib/postgresql/16/bin/pg\_ctl -D /var/lib/postgresql/16/main promote**

Старый primary убираем (сервер B):

```

tekassh1@postgres2:~$ sudo -u postgres touch /var/lib/postgresql/16/main/standby.signal

tekassh1@postgres2:~$ sudo -u postgres ls /var/lib/postgresql/16/main/standby.signal

/var/lib/postgresql/16/main/standby.signal

tekassh1@postgres2:~$ sudo systemctl restart postgresql

Обновляем pgpool (сервер client):

```

pcp\_attach\_node -h localhost -p 9898 -U postgres -n 0

```

Восстановление timeline (сервер B)

|  |
| --- |
| sudo systemctl stop postgresql  sudo systemctl stop postgresql  sudo rm -rf /var/lib/postgresql/16/main/\*  sudo rm -rf /var/lib/postgresql/16/main  sudo -u postgres mkdir /var/lib/postgresql/16/main  sudo chmod 700 /var/lib/postgresql/16/main  sudo chown -R postgres:postgres /var/lib/postgresql/16/main  sudo -u postgres pg\_basebackup -h 192.168.122.253 -D /var/lib/postgresql/16/main -U replicator -P -R  Password:  203725/203725 kB (100%), 1/1 tablespace  ALTER SYSTEM SET primary\_conninfo = 'user=replicator password=replicator\_pass host=192.168.122.253 port=5432 application\_name=postgres2';  sudo systemctl restart postgresql |

Восстановление timeline (сервер C)

|  |
| --- |
| sudo systemctl stop postgresql  sudo rm -rf /var/lib/postgresql/16/main/\*  sudo rm -rf /var/lib/postgresql/16/main  sudo -u postgres mkdir /var/lib/postgresql/16/main  sudo chmod 700 /var/lib/postgresql/16/main  sudo chown -R postgres:postgres /var/lib/postgresql/16/main  sudo -u postgres pg\_basebackup -h 192.168.122.198 -D /var/lib/postgresql/16/main -U replicator -P -R  Password:  203725/203725 kB (100%), 1/1 tablespac  ALTER SYSTEM SET primary\_conninfo = 'user=replicator password=replicator\_pass host=192.168.122.198 port=5432 application\_name=postgres3';  sudo systemctl restart postgresql |

После этого проверка, что всё работает.

# **Сложности**

Основная сложность была в правильной настройке pgpool так, чтобы распределение запросов действовало по нашей конфигурации. Также в определённый момент возникла проблема с выведением кластера из строя, так как не получалось правильно забить диск мусорными файлами.

# **Вывод**

В данной лабораторной работе мы реализовали Active-Standby кластеров PostgreSQL. У нас получилось настроить pgpool по заданию и выполнить восстановление primary сервера после его сбоя.