

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 Р33111**

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: А - основной, В и С - резервные. Для управления использовать pgpool-II. Репликация с А на В синхронная. Репликация с А на С асинхронная. Продемонстрировать, что новые данные реплицируются на В в синхронном режиме, а на С 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 `pskill -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.

```
[blps@2466588-mr34553:~]$ psql -U postgres -p 8888 -h localhost
psql (16.3 (Ubuntu 16.3-1.pgdg20.04+1), server 16.2 (Debian 16.2-1.pgdg120+2))
Type "help" for help.

[postgres=# sleect * from test;
ERROR:  syntax error at or near "sleect"
LINE 1: sleect * from test;
        ^

[postgres=# select * from test;
 id |      name      | flag
----+-----+-----
  1 | artem loh      | t
  1 | artem loh      | t
  1 | matthew cool   | t
(3 rows)

[postgres=# insert into test values (1, 'a', true);
INSERT 0 1
[postgres=# \q
```

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):

```
2024-05-21 23:22:08.577: main pid 1: LOG: reaper handler
2024-05-21 23:22:08.578: main pid 1: LOG: reaper handler: exiting normally
2024-05-21 23:22:21.804: psql pid 147: WARNING: failed to connect to PostgreSQL server, getaddrinfo() failed with error "Temporary failure in name resolution"
2024-05-21 23:22:21.805: psql pid 147: LOG: received degenerate backend request for node_id: 0 from pid [147]
2024-05-21 23:22:21.805: psql pid 147: LOG: signal_user1_to_parent_with_reason(0)
2024-05-21 23:22:21.805: psql pid 147: FATAL: failed to create a backend connection
2024-05-21 23:22:21.805: psql pid 147: DETAIL: executing failover on backend
2024-05-21 23:22:21.805: main pid 1: LOG: Pgpool-II parent process received SIGUSR1
2024-05-21 23:22:21.805: main pid 1: LOG: Pgpool-II parent process has received failover request
2024-05-21 23:22:21.806: main pid 1: LOG: == Starting degeneration. shutdown host postgres1(5432) ==
2024-05-21 23:22:21.808: main pid 1: LOG: Restart all children
2024-05-21 23:22:21.811: main pid 1: LOG: execute command: echo ">>> Failover - that will initialize new primary node search!"
>>> Failover - that will initialize new primary node search!
2024-05-21 23:22:21.813: main pid 1: LOG: find_primary_node_repeatedly: waiting for finding a primary node
2024-05-21 23:22:21.821: main pid 1: LOG: find_primary_node: standby node is 1
2024-05-21 23:22:21.821: main pid 1: LOG: find_primary_node: standby node is 2
```

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:

```
2024-05-22 09:00:30.660: main pid 1: LOG: find_primary_node: standby node is 1
2024-05-22 09:00:30.660: main pid 1: LOG: find_primary_node: standby node is 2
2024-05-22 09:00:31.672: main pid 1: LOG: find_primary_node: primary node is 1
2024-05-22 09:00:31.672: main pid 1: LOG: find_primary_node: standby node is 2
2024-05-22 09:00:31.673: main pid 1: LOG: create socket files[0]: /opt/bitnami/pgpool/tmp/.s.PGSQL.9898
2024-05-22 09:00:31.673: main pid 1: LOG: listen address[0]: localhost
2024-05-22 09:00:31.673: main pid 1: LOG: Setting up socket for ::1:9898
2024-05-22 09:00:31.674: main pid 1: LOG: Setting up socket for 127.0.0.1:9898
2024-05-22 09:00:31.675: pcp_main pid 70887: LOG: PCP process: 70887 started
2024-05-22 09:00:31.675: health_check pid 70889: LOG: process started
2024-05-22 09:00:31.676: health_check pid 70890: LOG: process started
2024-05-22 09:00:31.677: sr_check_worker pid 70888: LOG: process started
2024-05-22 09:00:31.683: main pid 1: LOG: pgpool-II successfully started. version 4.5.2 (hotooriboshi)
2024-05-22 09:00:31.684: main pid 1: LOG: node status[0]: 0
2024-05-22 09:00:31.684: main pid 1: LOG: node status[1]: 1
2024-05-22 09:00:31.684: main pid 1: LOG: node status[2]: 2
```

Пробуем подключиться и прочитать/добавить новые данные:

```
exit
[blps@2466588-mr34553:~$ psql -U postgres -p 8888 -h localhost
psql (16.3 (Ubuntu 16.3-1.pgdg20.04+1), server 16.2 (Debian 16.2-1.pgdg120+2))
Type "help" for help.

postgres=# \dt
      List of relations
 Schema | Name | Type  | Owner
-----+-----+-----+-----
 public | test | table | postgres
(1 row)

postgres=# insert into test values (123, 'replica', true);
INSERT 0 1
postgres=# \q
```



## Этап 3. Восстановление

Для восстановления работоспособности перематываем WALы:

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

```
blps@2466580-mr34553:~$ sudo su postgres  
postgres@2466580-mr34553:/home/blps$ /usr/lib/postgresql/16/bin/pg_rewind --target-pgdata postgres1/ --source-server="host=localhost port=7000 user=postgres" --progress  
pg_rewind: connected to server  
pg_rewind: executing "/usr/lib/postgresql/16/bin/postgres" for target server to complete crash recovery  
2024-05-22 10:58:10.991 GMT [4034766] LOG: skipping missing configuration file "/home/blps/postgres1/postgresql.auto.conf"  
2024-05-22 10:58:11.007 UTC [4034766] LOG: database system was interrupted; last known up at 2024-05-22 10:51:24 UTC  
2024-05-22 10:58:11.007 UTC [4034766] LOG: database system was not properly shut down; automatic recovery in progress  
2024-05-22 10:58:11.008 UTC [4034766] LOG: redo starts at 0/40000A0  
2024-05-22 10:58:11.008 UTC [4034766] LOG: invalid record length at 0/4004220: expected at least 24, got 0  
2024-05-22 10:58:11.009 UTC [4034766] LOG: redo done at 0/40041E8 system usage: CPU: user: 0.00 s, system: 0.00 s, elapsed: 0.00 s  
2024-05-22 10:58:11.010 UTC [4034766] LOG: checkpoint starting: end-of-recovery immediate wait  
2024-05-22 10:58:11.010 UTC [4034766] LOG: checkpoint complete: wrote 6 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.001 s, sync=0.001 s, total=0.001 s, longest=0.000 s, average=0.000 s; distance=16 kB, estimate=16 kB; lsn=0/4004220, redo lsn=0/4004220  
2024-05-22 10:58:11.015 UTC [4034766] WARNING: database "template1" has a collation version mismatch  
2024-05-22 10:58:11.015 UTC [4034766] DETAIL: The database was created using collation version 2.36, but the operating system provides version 2.31.  
2024-05-22 10:58:11.015 UTC [4034766] HINT: Rebuild all objects in this database that use the default collation and run ALTER DATABASE template1 REFRESH COLLATION VERSION, if you are using PostgreSQL with the right library version.  
PostgreSQL stand-alone backend 16.3 (Ubuntu 16.3-1.pgdg20.04+1)  
backend> 2024-05-22 10:58:11.029 UTC [4034766] LOG: checkpoint starting: shutdown immediate  
2024-05-22 10:58:11.044 UTC [4034766] LOG: checkpoint complete: wrote 0 buffers (0.0%); 0 WAL file(s) added, 0 removed, 0 recycled; write=0.001 s, sync=0.001 s, total=0.015 s, longest=0.000 s, average=0.000 s; distance=16367 kB, estimate=16367 kB; lsn=0/5000028, redo lsn=0/5000028  
pg_rewind: servers diverged at WAL location 0/4004220 on timeline 1  
pg_rewind: rewinding from last common checkpoint at 0/4000028 on timeline 1  
pg_rewind: reading source file list  
pg_rewind: reading target file list  
pg_rewind: reading WAL in target  
pg_rewind: need to copy 36 MB (total source directory size is 61 MB)  
37752/37752 kB (100%) copied  
pg_rewind: creating backup label and updating control file  
pg_rewind: syncing target data directory  
pg_rewind: Done!  
postgres@2466580-mr34553:/home/blps$
```

Теперь запустим postgres1.

Добавим обратно ноду в pgpool:

```
pcr_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.

```
2024-05-22 11:12:07.030: main pid 1: LOG: reaper handler  
2024-05-22 11:12:07.837: main pid 1: LOG: reaper handler: exiting normally  
2024-05-22 11:12:16.602: sr_check_worker pid 4929: LOG: get_query_result failed: status: -2  
2024-05-22 11:12:16.602: sr_check_worker pid 4929: CONTEXT: while checking replication time lag  
2024-05-22 11:12:46.625: sr_check_worker pid 4929: LOG: get_query_result failed: status: -2  
2024-05-22 11:12:46.625: sr_check_worker pid 4929: CONTEXT: while checking replication time lag  
2024-05-22 11:12:58.602: main pid 1: LOG: reaper handler  
2024-05-22 11:12:58.602: main pid 1: LOG: reaper handler: exiting normally  
2024-05-22 11:13:07.274: pcp_main pid 4928: LOG: forked new pcp worker, pid=5883 socket=7  
2024-05-22 11:13:07.278: pcp_child pid 5883: LOG: received failback request for node_id: 0 from pid [5883]  
2024-05-22 11:13:07.278: pcp_child pid 5883: LOG: signal_user1_to_parent_with_reason(0)  
2024-05-22 11:13:07.278: main pid 1: LOG: Pgpool-II parent process received SIGUSR1  
2024-05-22 11:13:07.278: main pid 1: LOG: Pgpool-II parent process has received failover request  
2024-05-22 11:13:07.278: main pid 1: LOG: invalid failback request, status: [2] of node id : 0 is invalid for failback  
2024-05-22 11:13:07.281: pcp_main pid 4928: LOG: PCP process with pid: 5883 exit with SUCCESS.  
2024-05-22 11:13:07.281: pcp_main pid 4928: LOG: PCP process with pid: 5883 exits with status 0  
2024-05-22 11:13:16.645: sr_check_worker pid 4929: LOG: get_query_result failed: status: -2  
2024-05-22 11:13:16.645: sr_check_worker pid 4929: CONTEXT: while checking replication time lag
```

```
eplication mode and not all backends were down  
2024-05-22 11:18:57.184: main pid 1: LOG: find_primary_node_repeatedly: waiting for finding a primary node  
2024-05-22 11:18:57.196: main pid 1: LOG: find_primary_node: primary node is 0  
2024-05-22 11:18:57.197: main pid 1: LOG: find_primary_node: standby node is 1  
2024-05-22 11:18:57.197: main pid 1: LOG: find_primary_node: standby node is 2  
2024-05-22 11:18:57.200: main pid 1: LOG: failover: set new primary node: 0  
2024-05-22 11:18:57.200: main pid 1: LOG: failover: set new main node: 0  
2024-05-22 11:18:57.201: main pid 1: LOG: === Failback done. reconnect host postgres2(5432) ===  
2024-05-22 11:18:57.202: sr_check_worker pid 6616: LOG: worker process received restart request  
2024-05-22 11:18:58.201: pcp_main pid 6615: LOG: restart request received in pcp child process  
2024-05-22 11:18:58.203: main pid 1: LOG: PCP child 6615 exits with status 0 in failover()  
2024-05-22 11:18:58.203: main pid 1: LOG: fork a new PCP child pid 6648 in failover()  
2024-05-22 11:18:58.203: main pid 1: LOG: reaper handler  
2024-05-22 11:18:58.204: main pid 1: LOG: reaper handler: exiting normally
```

## **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.