Async replication

https://github.com/popkir/highload-social-network/commit/f071c6c8ba88a87300afb193d0174f1e204b4191

Как создать две асинхронные реплики

1. В docker-compose.yml замапил папки с данными в контейнер с базой данных:

   ```
   services:
     db_0:
       volumes:
       - ./postgres_data/db_0:/var/lib/postgresql/data
   ```
2. Запустил бэкенд и базу, накатил миграции (включая создание роли для репликации):

   ```
   docker-compose exec -it backend alembic revision head
   ```
   В папке `./postgres_data/db_0` появилось много интересных файлов.

3. Скопировал полученный на предыдущем шаге шаблон конфиг файла `./postgres_data/db_0/postgres.conf` в `./postgres_config/postgresql.db0.conf`.

   Добавил следующие строки:

   ```
   ssl = off
   wal_level = replica
   max_wal_senders = 4 # expected slave number
   ```
4. Посмотрел маску подсети через `docker network inspect social-network-default | grep Subnet`: 172.18.0.0/16.

   Скопировал полученный на предыдущем шаге шаблон конфиг файла `./postgres_data/db_0/pg_hba.conf` в `./postgres_config/pg_hba.conf`.

   Добавил следующую строку: `host    replication     replicator      172.18.0.0/16           scram-sha-256`

5. Добавил мэппинг конфигурационных файлов в контейнер:

   ```
   services:
     db_0:
       volumes:
       - ./postgres_data/db_0:/var/lib/postgresql/data
       -
   ./postgres_config/postgresql.db0.conf:/var/lib/postgresql/data/postgresql.co
       - ./postgres_config/pg_hba.conf:/var/lib/postgresql/data/pg_hba.conf
   ```
6. Запустил сервисы. Сделал бэкап для реплик:

   ```
   docker exec -it social-network-db_0 bash
   mkdir /pgslave
   pg_basebackup -h social-network-db_0 -D /pgslave -U replicator -v -P --
   wal-method=stream
   exit
   ```
7. Создал две пустые папки: `./postgres_data/db_1` и `./postgres_data/db_2`.

Скопировал в них бэкап первой базы данных:

```
docker cp social-network-db_0:/pgslave postgres_data/db_1
docker cp social-network-db_0:/pgslave postgres_data/db_2
```

8. Создал два контейнера в `docker-compose.yml`:

```
db_1:
 image: postgres:15
 container_name: social-network-db_1
 volumes:
   - ./postgres_data/db_1:/var/lib/postgresql/data
 networks:
   - social-network-default
 ports:
   - "15432:5432"
db_2:
  image: postgres:15
  container_name: social-network-db_2
  volumes:
    - ./postgres_data/db_2:/var/lib/postgresql/data
  networks:
    - social-network-default
  ports:
    - "25432:5432"
```

9. Запустил сервисы и проверил, что все работает.

10. Создал версию конфиг файла для реплик, сделав `cp postgres_config/postgresql.db0.conf postgres_config/postgresql.db1.conf` и добавив туда строку `primary_conninfo = 'host=social-network-db_0 port=5432 user=replicator password=replicator application_name=pgslave_db_1'`.

    Аналогично создал файл `postgres_config/postgresql.db2.conf` с параметром `application_name=pgslave_db_2`.

11. Создал пустой файл `postgres_config/standby.signal`.

12. Удалил файлы `postgres.conf` и `pg_hba.conf` из папок `postgres_data/db_0`, `postgres_data/db_1`, `postgres_data/db_2`.

13. Добавил мэппинги конфигурационных файлов в `docker-compose.yml`:

```
db_1:
  volumes:
    - ./postgres_data/db_1:/var/lib/postgresql/data
    -
./postgres_config/postgresql.db1.conf:/var/lib/postgresql/data/postgresql.co
    -
./postgres_config/pg_hba.conf:/var/lib/postgresql/data/pg_hba.conf
    -
./postgres_config/standby.signal:/var/lib/postgresql/data/standby.signal

db_2:
  volumes:
    - ./postgres_data/db_2:/var/lib/postgresql/data
    -
./postgres_config/postgresql.db2.conf:/var/lib/postgresql/data/postgresql.co
    -
./postgres_config/pg_hba.conf:/var/lib/postgresql/data/pg_hba.conf
```

```
    -
./postgres_config/standby.signal:/var/lib/postgresql/data/standby.signal
```

14. Перезапустил сервисы. Проверил по логам, что все успешно настроено.
15. Проверил еще раз, создав сессию к базе `db_0` (мастер) и выполнив запрос `select application_name, sync_state from pg_stat_replication;`, убедившись, что две реплики подключились в асинхронном режиме

Как перевести запросы на реплики

Я поменял файлы окружения и импорт из них в проекте, так, чтобы теперь все 3 ссылки были доступны.

Создал фабрики сессий ко всем трем серверам.

Вместо использования scoped_session напрямую, создал вспомогательный класс:

```
class SessionManager:
    master_session = scoped_session(master_session_factory)
    slave_sessions = [scoped_session(s) for s in
slave_session_factories]
    current = master_session
```

Перевел уже имеющиеся запросы на использование `SessionManager.current`.

Создал фабрику декораторов, подменяющих сессии в скоупе функции, завернутой в декоратор, на выбранный или случайный слейв, а затем заменяющих их обратно:

```
def with_slave(which_slave=None):
    def decorator(func):
        @wraps(func)
        async def wrapper(*args, **kwargs):
            if which_slave is not None:
                SessionManager.current =
SessionManager.slave_sessions[which_slave]
            else:
                SessionManager.current =
random.choice(SessionManager.slave_sessions)
            res = await func(*args, **kwargs)
            SessionManager.current = SessionManager.master_session
            return res
        return wrapper
    return decorator
```

Эндпоинты, которые хочу перевести на слейв, завернул в этот декоратор:

```
@router.get("/search", response_model=dict)
@with_slave()
@close_session
async def search_user(first_name: str = None, last_name: str = None):
    ...
```

Для проверки, что подмена сессий происходит корректно, провел нагрузочный тест, в котором одновременно подавалась нагрузка на эндпоинты, перведенные на реплики, и на эндпоинты, производящие запись в базу. Ошибок не было.

Нагрузочный тест - до и после подключения реплик

Протокол

Нагрузочный тест проведен с помощью фреймворка Locust.

Нагрузка была распределена поровну между `user/get/{id}` и `user/search`.

После подключения реплик, оба этих запроса были переведены на реплики для чтения. Был использован случайный выбор реплики при обработке каждого запроса.

В качестве профиля нагрузки я использовал следующие параметры:

- 3 минуты с 1 одновременным пользователем, скорость появления новых пользователей 1 пользователь/с
- 3 минуты с 10 одновременными пользователями, скорость появления новых пользователей 10 пользователей/с
- 3 минуты с 100 одновременными пользователями, скорость появления новых пользователей 100 пользователей/с
- 3 минуты с 1000 одновременными пользователями, скорость появления новых пользователей 1000 пользователей/с

Замерялись следующие характеристики:

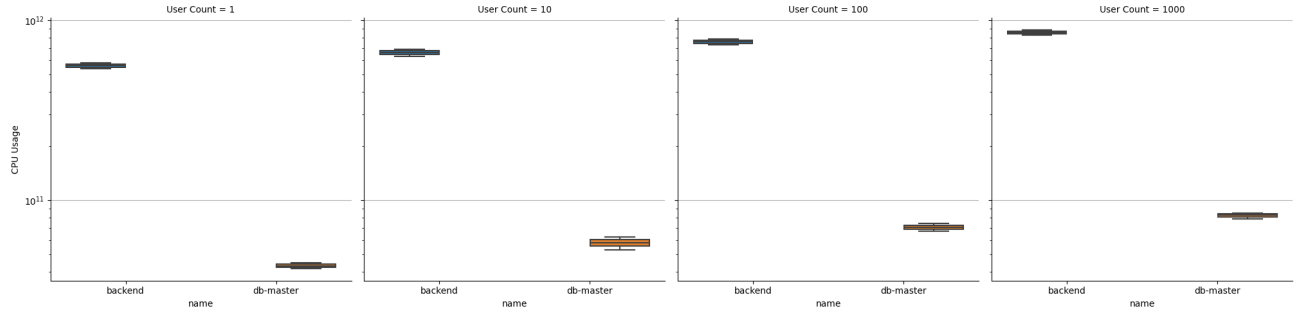
- использование процессора, памяти и диска (через docker stats)
- latency (50%, 95%) и throughput запросов с нагружающей стороны (через locust).

(Мы попробовали мерять latency запросов в базе через postgres-exporter + prometheus + grafana, но не достигли понятно интерпретируемого результата)

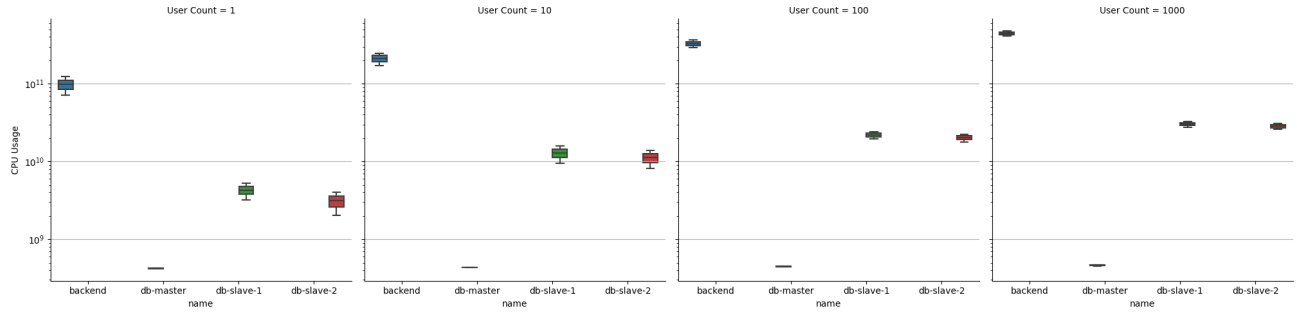
Первые 60 секунд каждой из фаз нагрузки были исключены из анализа.

CPU

Без реплик



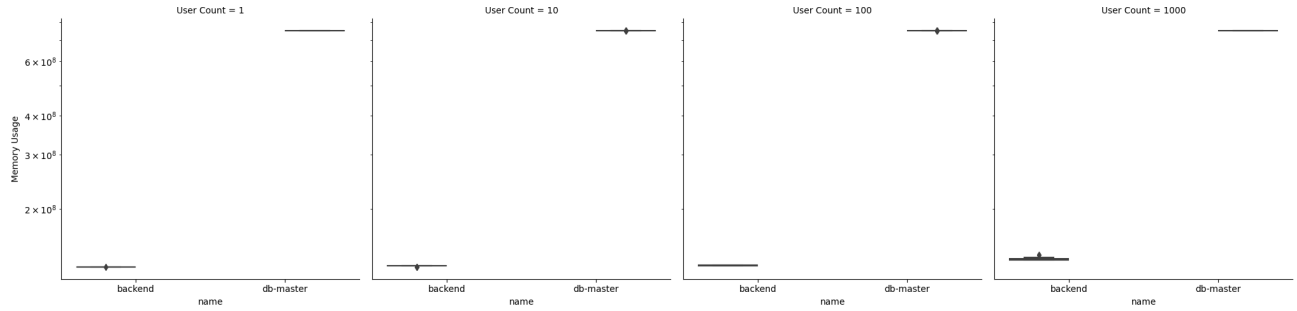
С использованием реплик



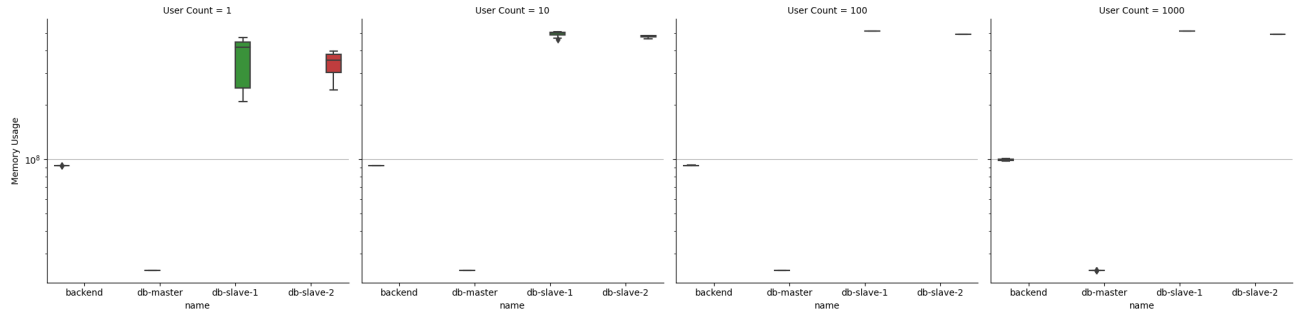
Очевидно, при переводе запросов на реплики, мастер потребляет меньше ресурсов процессора.

RAM

Без реплик



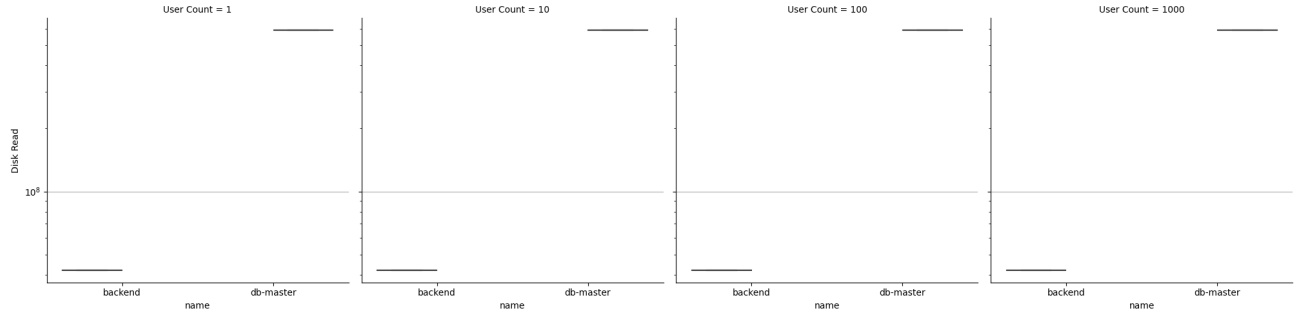
С использованием реплик



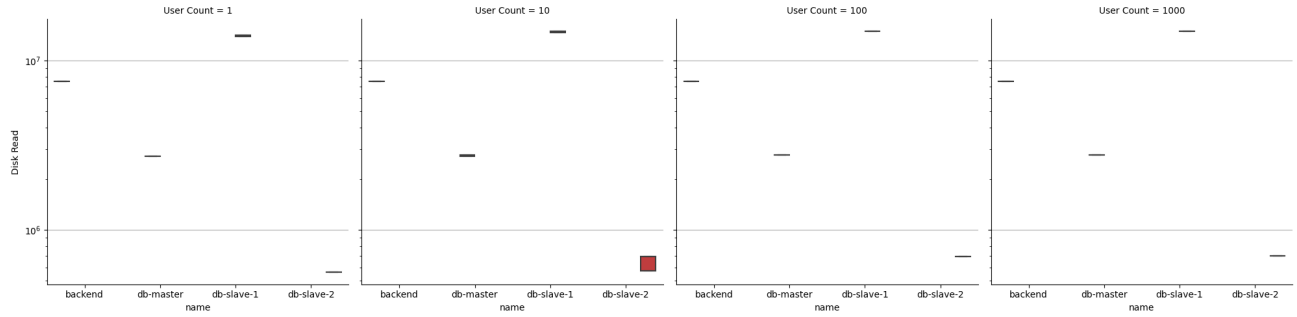
При переводе запросов на реплики, мастер потребляет меньше ресурсов памяти.

Disk Reads

Без реплик



С использованием реплик

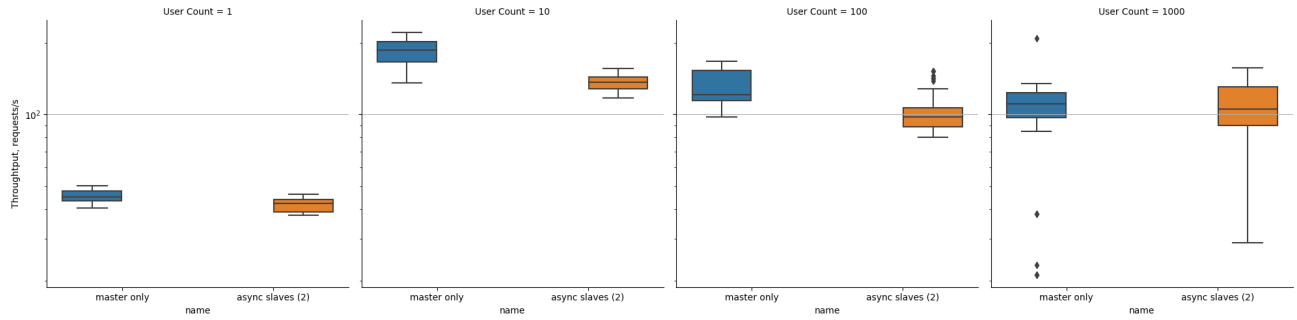


При переводе запросов на реплики, мастер потребляет меньше ресурсов диска.

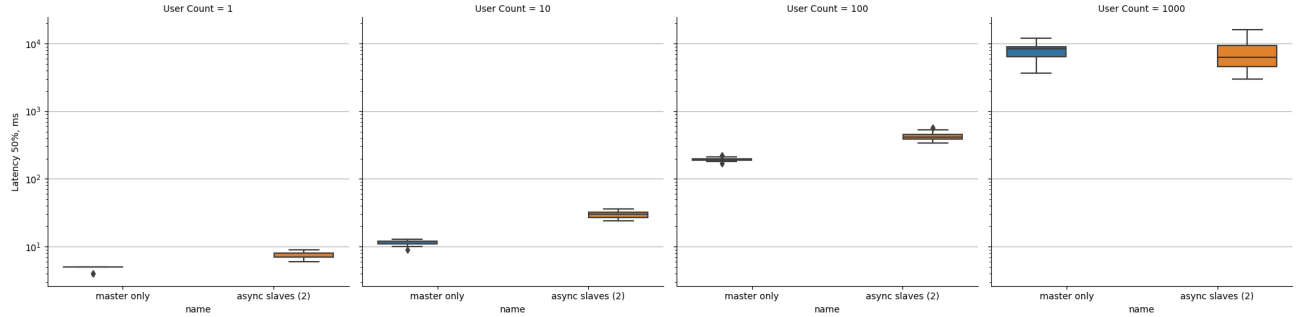
Почему одна из реплик якобы потребляет еще меньше? Не знаю, не хочу повторять эксперимент. Нагрузка между репликами точно была расределена поровну, возможно, баг сбора телеметрии.

Метрики со стороны клиента

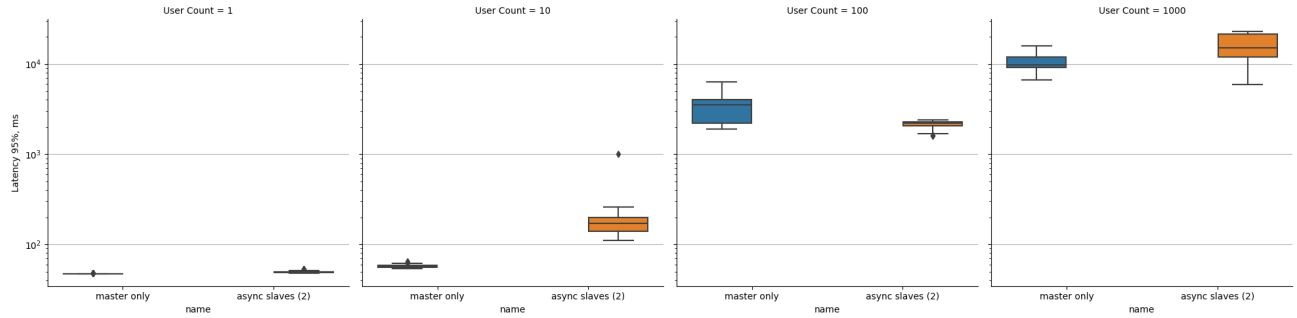
Throughput



Latency 50%



Latency 95%



Можно заметить, что подключение реплик:

- незначительно уменьшает throughput
- чуть более значительно увеличивает latency

Quorum sync replication

https://github.com/popkir/highload-social-network/commit/badcb4067e476951fe1715d99fcd33e83112d74c

Как было сделано

1. В файл `postgres_config/postgresql.db0.conf` добавлены следующие строки:

   ```
   synchronous_commit = on
   synchronous_standby_names = 'ANY 1 (pgslave_db_1, pgslave_db_2)'
   ```
2. Сервисы перезапущены, проверил в логах и в `pg_stat_replication`, что настройка применена успешно.

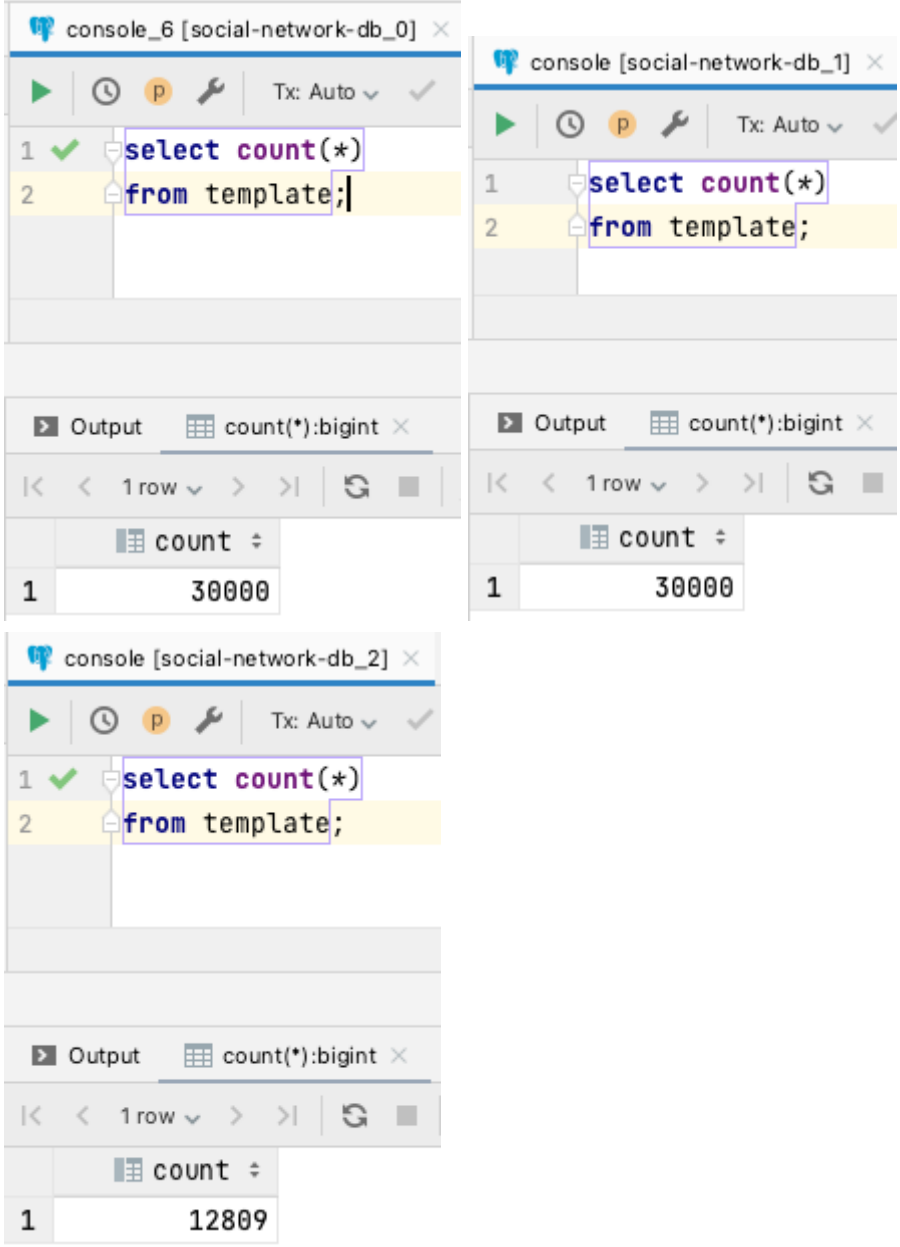
3. Подаем нагрузку на запись в таблицу `template` через эндпоинт `template/generate`. Записываем 30000 записей.

4. После ~12000 записей, останавливаем сервер `db2`: `docker stop social-network-db_2`

5. Ждем завершения обработки запроса. Видим логи об успешной записи 30000 строк:

```
social-network-backend  | INFO:     172.18.0.1:39216 - "GET /template/generate?number=30000 HTTP/1.1" 200 OK
social-network-backend  | [2023-07-17,06:23:47.877 : MainThread] [INFO] [request 62e9cafc] Inserted 1000 templates out of 999
social-network-backend  | [2023-07-17,06:23:54.141 : MainThread] [INFO] [request 62e9cafc] Inserted 2000 templates out of 1999
social-network-backend  | [2023-07-17,06:24:02.132 : MainThread] [INFO] [request 62e9cafc] Inserted 3000 templates out of 2999
social-network-backend  | [2023-07-17,06:24:08.310 : MainThread] [INFO] [request 62e9cafc] Inserted 4000 templates out of 3999
social-network-backend  | [2023-07-17,06:24:14.287 : MainThread] [INFO] [request 62e9cafc] Inserted 5000 templates out of 4999
social-network-backend  | [2023-07-17,06:24:20.424 : MainThread] [INFO] [request 62e9cafc] Inserted 6000 templates out of 5999
social-network-backend  | [2023-07-17,06:24:26.167 : MainThread] [INFO] [request 62e9cafc] Inserted 7000 templates out of 6999
social-network-backend  | [2023-07-17,06:24:32.330 : MainThread] [INFO] [request 62e9cafc] Inserted 8000 templates out of 7999
social-network-backend  | [2023-07-17,06:24:38.348 : MainThread] [INFO] [request 62e9cafc] Inserted 9000 templates out of 8999
social-network-backend  | [2023-07-17,06:24:44.192 : MainThread] [INFO] [request 62e9cafc] Inserted 10000 templates out of 9999
social-network-backend  | [2023-07-17,06:24:49.936 : MainThread] [INFO] [request 62e9cafc] Inserted 11000 templates out of 10999
social-network-backend  | [2023-07-17,06:24:55.748 : MainThread] [INFO] [request 62e9cafc] Inserted 12000 templates out of 11999
social-network-db_2     | 2023-07-17 06:25:01.053 UTC [1] LOG:  received fast shutdown request
social-network-db_2     | 2023-07-17 06:25:01.058 UTC [1] LOG:  aborting any active transactions
social-network-db_2     | 2023-07-17 06:25:01.058 UTC [31] FATAL:  terminating walreceiver process due to administrator command
social-network-db_2     | 2023-07-17 06:25:01.088 UTC [26] LOG:  shutting down
social-network-db_2     | 2023-07-17 06:25:01.088 UTC [26] LOG:  restartpoint starting: shutdown immediate
social-network-db_2     | 2023-07-17 06:25:01.408 UTC [26] LOG:  restartpoint complete: wrote 279 buffers (1.7%); 0 WAL file(s) add
ed, 0 removed, 0 recycled; write=0.305 s, sync=0.003 s, total=0.321 s; sync files=9, longest=0.002 s, average=0.001 s; distance=388
1 kB, estimate=3881 kB
social-network-db_2     | 2023-07-17 06:25:01.408 UTC [26] LOG:  recovery restart point at 0/1DAC76D8
social-network-db_2     | 2023-07-17 06:25:01.408 UTC [26] DETAIL:  Last completed transaction was at log time 2023-07-17 06:25:01.
036241+00.
social-network-db_2     | 2023-07-17 06:25:01.421 UTC [1] LOG:  database system is shut down
social-network-db_2 exited with code 0
social-network-db_2 exited with code 0
social-network-db_1     | 2023-07-17 06:25:02.017 UTC [27] LOG:  restartpoint starting: time
social-network-backend  | [2023-07-17,06:25:02.278 : MainThread] [INFO] [request 62e9cafc] Inserted 13000 templates out of 12999
social-network-db_0     | 2023-07-17 06:25:05.530 UTC [26] LOG:  checkpoint starting: time
social-network-backend  | [2023-07-17,06:25:08.059 : MainThread] [INFO] [request 62e9cafc] Inserted 14000 templates out of 13999
social-network-backend  | [2023-07-17,06:25:13.555 : MainThread] [INFO] [request 62e9cafc] Inserted 15000 templates out of 14999
social-network-backend  | [2023-07-17,06:25:19.038 : MainThread] [INFO] [request 62e9cafc] Inserted 16000 templates out of 15999
social-network-backend  | [2023-07-17,06:25:24.444 : MainThread] [INFO] [request 62e9cafc] Inserted 17000 templates out of 16999
social-network-db_1     | 2023-07-17 06:25:29.186 UTC [27] LOG:  restartpoint complete: wrote 281 buffers (1.7%); 0 WAL file(s) add
ed, 0 removed, 0 recycled; write=27.153 s, sync=0.001 s, total=27.170 s; sync files=9, longest=0.001 s, average=0.001 s; distance=3
881 kB, estimate=3881 kB
social-network-db_1     | 2023-07-17 06:25:29.186 UTC [27] LOG:  recovery restart point at 0/1DAC76D8
social-network-db_1     | 2023-07-17 06:25:29.186 UTC [27] DETAIL:  Last completed transaction was at log time 2023-07-17 06:25:29.
181372+00.
social-network-backend  | [2023-07-17,06:25:29.864 : MainThread] [INFO] [request 62e9cafc] Inserted 18000 templates out of 17999
social-network-db_0     | 2023-07-17 06:25:33.650 UTC [26] LOG:  checkpoint complete: wrote 284 buffers (1.7%); 0 WAL file(s) added
, 1 removed, 0 recycled; write=28.095 s, sync=0.001 s, total=28.120 s; sync files=10, longest=0.001 s, average=0.001 s; distance=60
27 kB, estimate=6027 kB
social-network-backend  | [2023-07-17,06:25:35.119 : MainThread] [INFO] [request 62e9cafc] Inserted 19000 templates out of 18999
social-network-backend  | [2023-07-17,06:25:40.334 : MainThread] [INFO] [request 62e9cafc] Inserted 20000 templates out of 19999
social-network-backend  | [2023-07-17,06:25:45.433 : MainThread] [INFO] [request 62e9cafc] Inserted 21000 templates out of 20999
social-network-backend  | [2023-07-17,06:25:50.749 : MainThread] [INFO] [request 62e9cafc] Inserted 22000 templates out of 21999
social-network-backend  | [2023-07-17,06:25:56.047 : MainThread] [INFO] [request 62e9cafc] Inserted 23000 templates out of 22999
social-network-backend  | [2023-07-17,06:26:01.401 : MainThread] [INFO] [request 62e9cafc] Inserted 24000 templates out of 23999
social-network-backend  | [2023-07-17,06:26:06.540 : MainThread] [INFO] [request 62e9cafc] Inserted 25000 templates out of 24999
social-network-backend  | [2023-07-17,06:26:11.866 : MainThread] [INFO] [request 62e9cafc] Inserted 26000 templates out of 25999
social-network-backend  | [2023-07-17,06:26:17.185 : MainThread] [INFO] [request 62e9cafc] Inserted 27000 templates out of 26999
social-network-backend  | [2023-07-17,06:26:22.445 : MainThread] [INFO] [request 62e9cafc] Inserted 28000 templates out of 27999
social-network-backend  | [2023-07-17,06:26:27.822 : MainThread] [INFO] [request 62e9cafc] Inserted 29000 templates out of 28999
social-network-backend  | [2023-07-17,06:26:33.133 : MainThread] [INFO] [request 62e9cafc] Inserted 30000 templates out of 29999
social-network-backend  | [2023-07-17,06:26:33.134 : MainThread] [INFO] [request 62e9cafc] Successfully inserted 30000 templates
```

6. Останавливаем все базы, запускаем их по одной и проверяем число записей в таблице `template`:



1. Удаляем строки из пункта 1 из `postgres_config/postgresql.db0.conf`. Добавляем `primary_conninfo = 'host=social-network-db_1 port=5432 user=replicator password=replicator application_name=pgslave_db_0'`

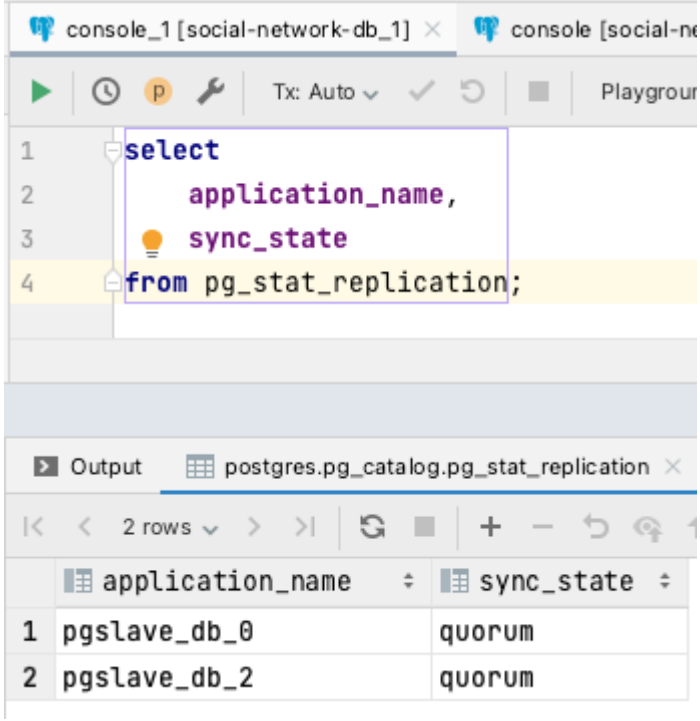
2. Вносим в `postgres_config/postgresql.db1.conf`:

```
synchronous_commit = on
synchronous_standby_names = 'ANY 1 (pgslave_db_0, pgslave_db_2)'
```

3. Редактируем `docker-compose.yml` - удаляем мэппинг `standby.signal` из `db_1` и добавляем в `db_0`.

4. Проверяем, что в папках `postgres_data/db_{NUMBER}` нет файлов `postgresql.conf`, `pg_hba.conf`, `standby.signal`. Если есть, удаляем.

5. Перезапускаем все сервисы. Проверяем, что `db_1` стал мастером:



6. Проверяем, что `db_2` получил все потерянные записи: