Полусинхронная репликация

1. Поднимаем кластер в докере из репы https://github.com/skalentev/OTUS-SoNet.git, pg1 – мастер, pg2 – асинхронная реплика:

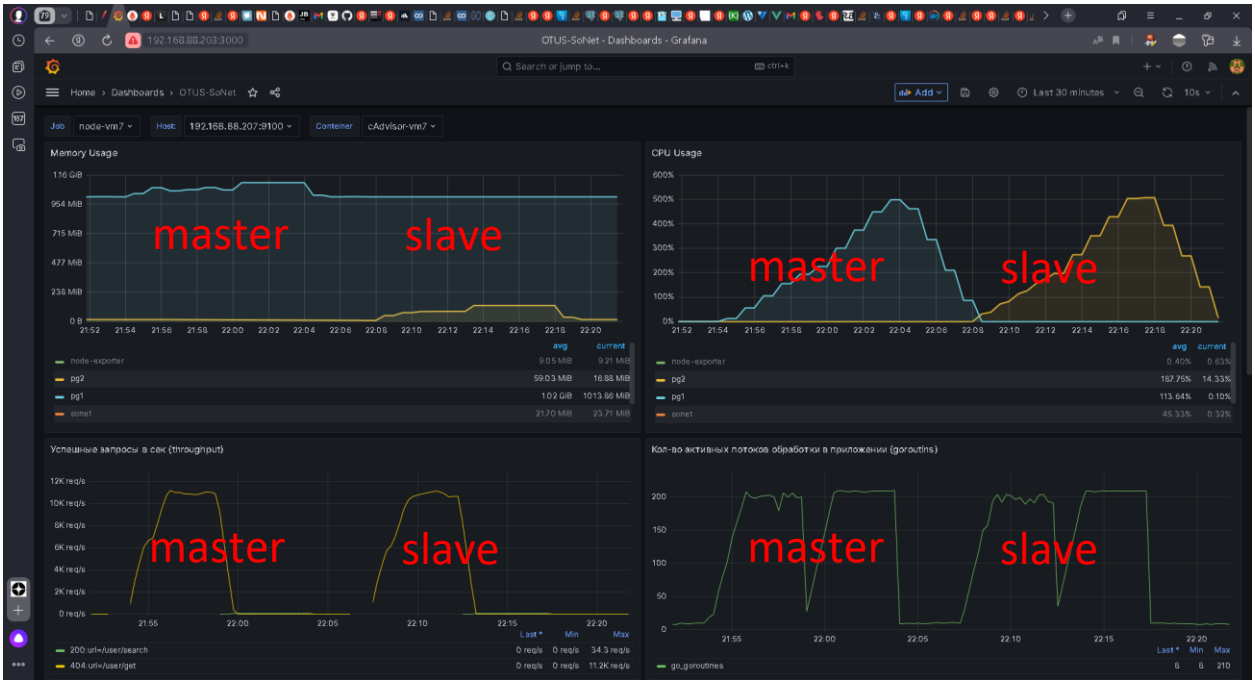
на мастере в конфиге: *wal_level = replica*
на слейве: *primary_conninfo = 'host=pg1 port=5432 user=replicator password=pass application_name=pg2'*

```
git clone https://github.com/skalentev/OTUS-SoNet.git
cd OTUS-SoNet

sudo docker compose -f ./Cluster/docker-compose.yml up -d pg1
sudo docker cp Cluster/Postgresql1.conf pg1:/var/lib/postgresql/data/postgresql.conf
sudo docker cp Cluster/pg hba.conf pg1:/var/lib/postgresql/data/pg hba.conf
sudo docker compose -f ./Cluster/docker-compose.yml restart pg1

sudo docker exec pg1 mkdir -p /pgslave;
sudo docker exec -e PGPASSWORD='pass' pg1 pg basebackup -h pg1 -D /pgslave -U replicator
-v -P --wal-method=stream
sudo docker cp pg1:/pgslave /tmp/pgslave
sudo cp -r /tmp/pgslave/ /tmp/data pg2/
sudo cp Cluster/Postgresql2.conf /tmp/data_pg2/postgresql.conf
sudo cp Cluster/pg hba.conf /tmp/data pg2/pg hba.conf
sudo cp Cluster/standby.signal /tmp/data pg2/standby.signal
sudo docker compose -f ./Cluster/docker-compose.yml up -d pg2
```

2. Делаем отдельный datasource для user\get и user\search

   (см. https://github.com/skalentev/OTUS-SoNet)

3. Замеряем нагрузку при настройке запросов на мастер pg1 (время на графике 21:54-22:04) и на реплику pg2 (22:08 – 22:18)



Чтение с реплики мастер не нагружает.

Пропускная способность при настройке на мастер:

Label	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Throughput	# Samples	Error %	Received KB/sec
userGetId100	8	6	16	23	54	0	291	9746.3/sec	2924341	100.00%	1275.40
userSearch100	2452	1926	4272	5787	12455	65	44915	34.0/sec	10269	0.00%	3096.37
TOTAL	17	6	16	23	64	0	44915	4872.6/sec	2934610	99.65%	2189.16

Пропускная способность при настройке на реплику:

Label	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Throughput	# Samples	Error %	Received KB/sec
userGetId100	8	6	16	23	54	0	239	9541.3/sec	2862815	100.00%	1248.57
userSearch100	2513	1962	4478	6299	13196	68	26573	33.2/sec	10025	0.00%	3021.50
TOTAL	17	6	17	24	64	0	26573	4769.0/sec	2872840	99.65%	2138.41

4. Запускаем вторую реплику pg3

```
sudo cp -r /tmp/pgslave/ /tmp/data pg3/
sudo cp Cluster/Postgresql3.conf /tmp/data pg3/postgresql.conf
sudo cp Cluster/pg hba.conf /tmp/data pg3/pg hba.conf
sudo cp Cluster/standby.signal /tmp/data_pg3/standby.signal
sudo docker compose -f ./Cluster/docker-compose.yml up -d pg2 pg3
```

5. –
6. –
7. Добавляем в конфиг мастера pg1 postgresql.conf настройки:

```
synchronous_commit = on
synchronous_standby_names = 'ANY 1 (pg2, pg3)'
```

Проверяем:

```
cluster=# select application_name, sync_state from pg_stat_replication;
 application_name | sync_state
------------------+------------
 pg2              | quorum
 pg3              | quorum
```

8. Выполняем 100 запросов /user/login, которые создают запись в таблицу session
9. На 26 итерации выполняем sudo docker stop pg2
10. В постмане дожидаемся окончания записи:

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	100	1m 49s	200	1061 ms

На реплике pg3 проверяем количество записей:

```
cluster=# select count(1) from public.session;
 count
-------
   100
```

11. Промоутим pg3:

В конфиг добавляем

```
synchronous_commit = on
synchronous_standby_names = 'ANY 1 (pg1, pg2, pg3)'
```

выполняем:

```
cluster=# select pg_promote();
 pg_promote
------------
 t
(1 row)
```

На pg2 в конфиге меняем адрес мастера на pg3

```
primary_conninfo = 'host=pg3 port=5432 user=replicator password=pass application_name=pg2'
```

Останавливаем pg1 (бывший мастер) и запускаем перенастроенный слейв pg2

```
docker stop pg1
docker start pg2
```

12. Проверяем число записей на pg2:

```
sudo docker exec -ti pg2 psql -d cluster -U user
cluster=# select count(1) from public.session;
 count
-------
   100
```

Потерь нет.