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

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

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

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



master

slave

slave

master

master

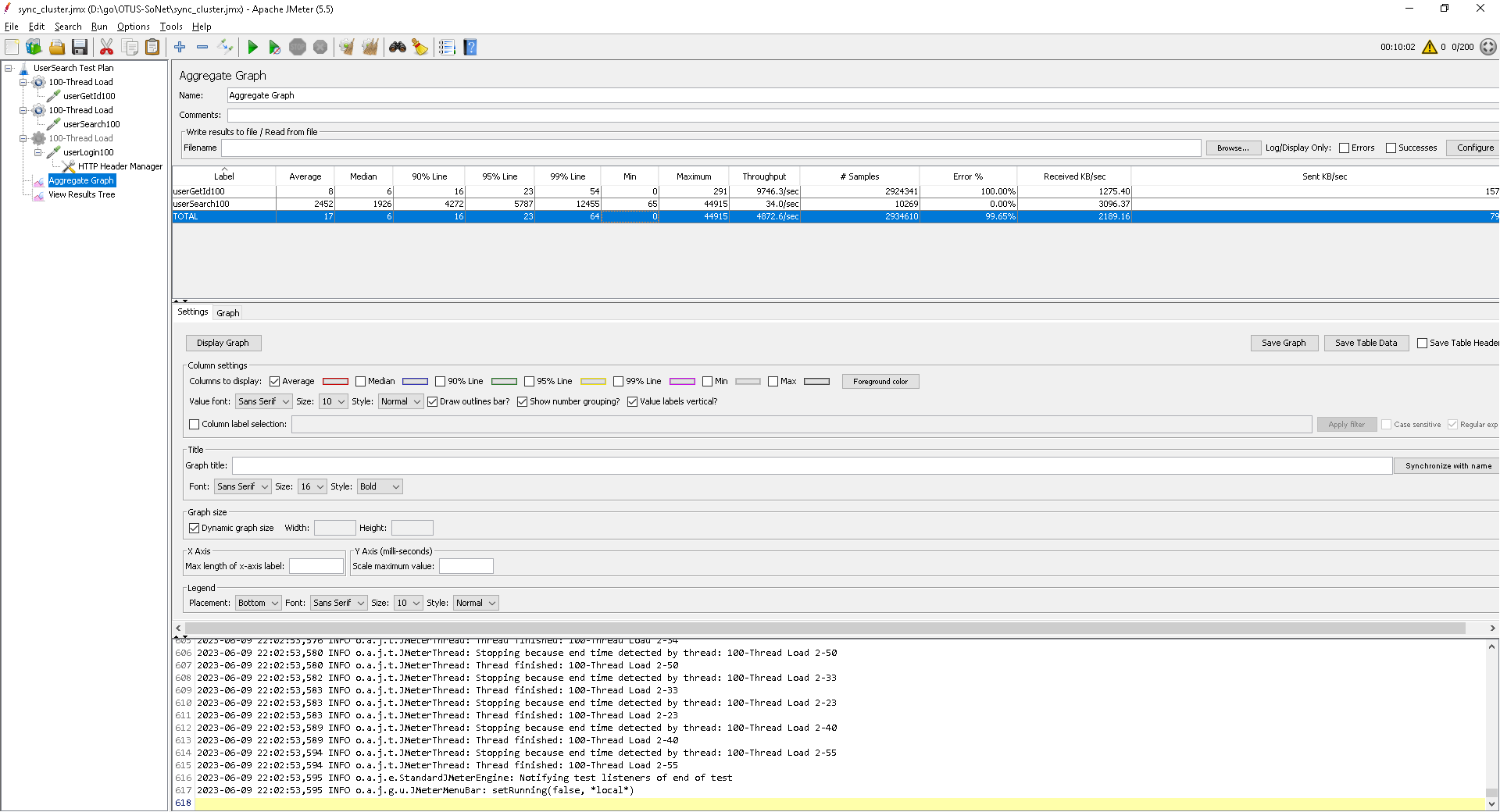
slave

slave

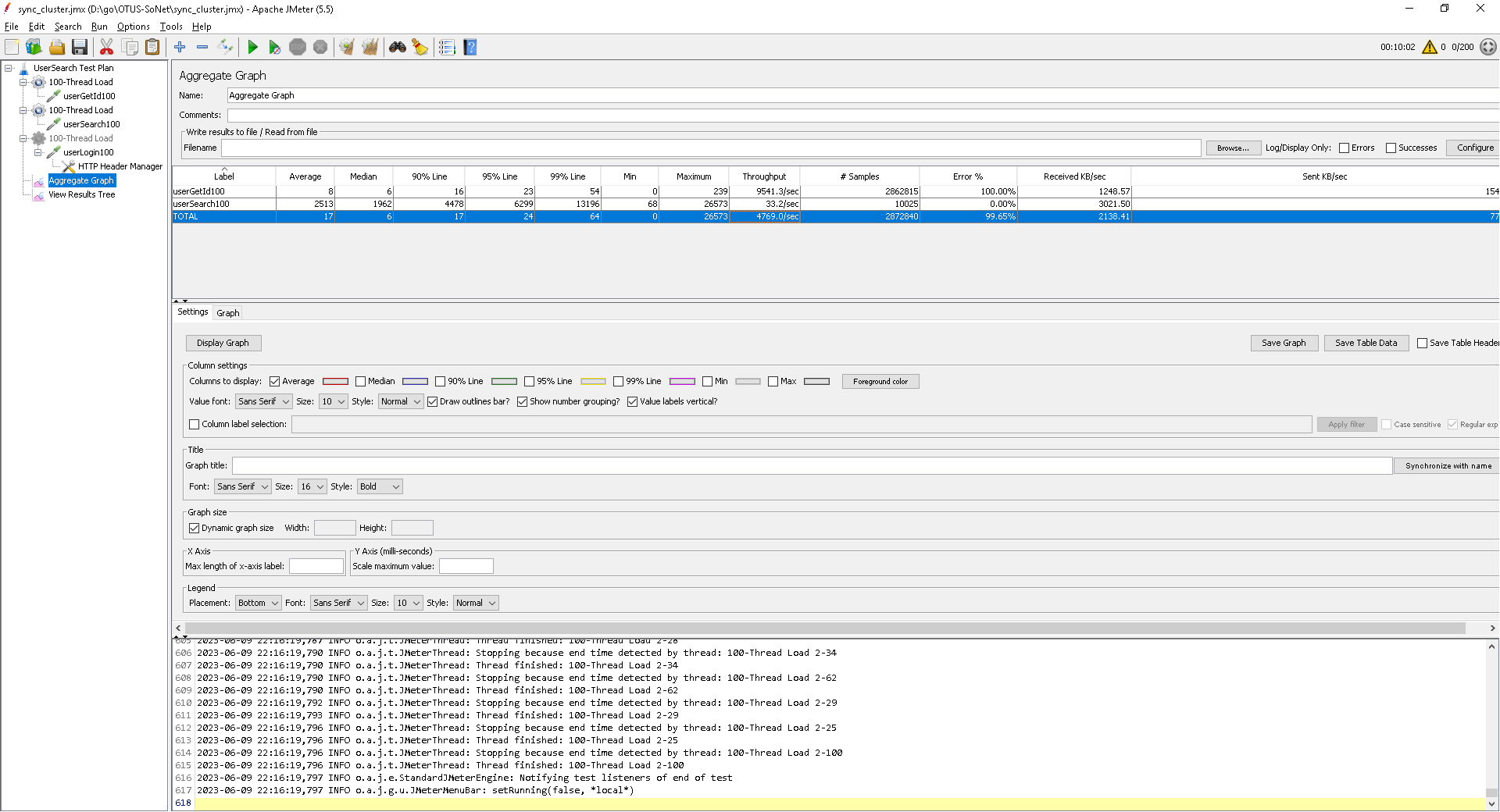
master

Чтение с реплики мастер не нагружает.

Пропускная способность при настройке на мастер:



Пропускная способность при настройке на реплику:



1. 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

1. –
2. –
3. 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;

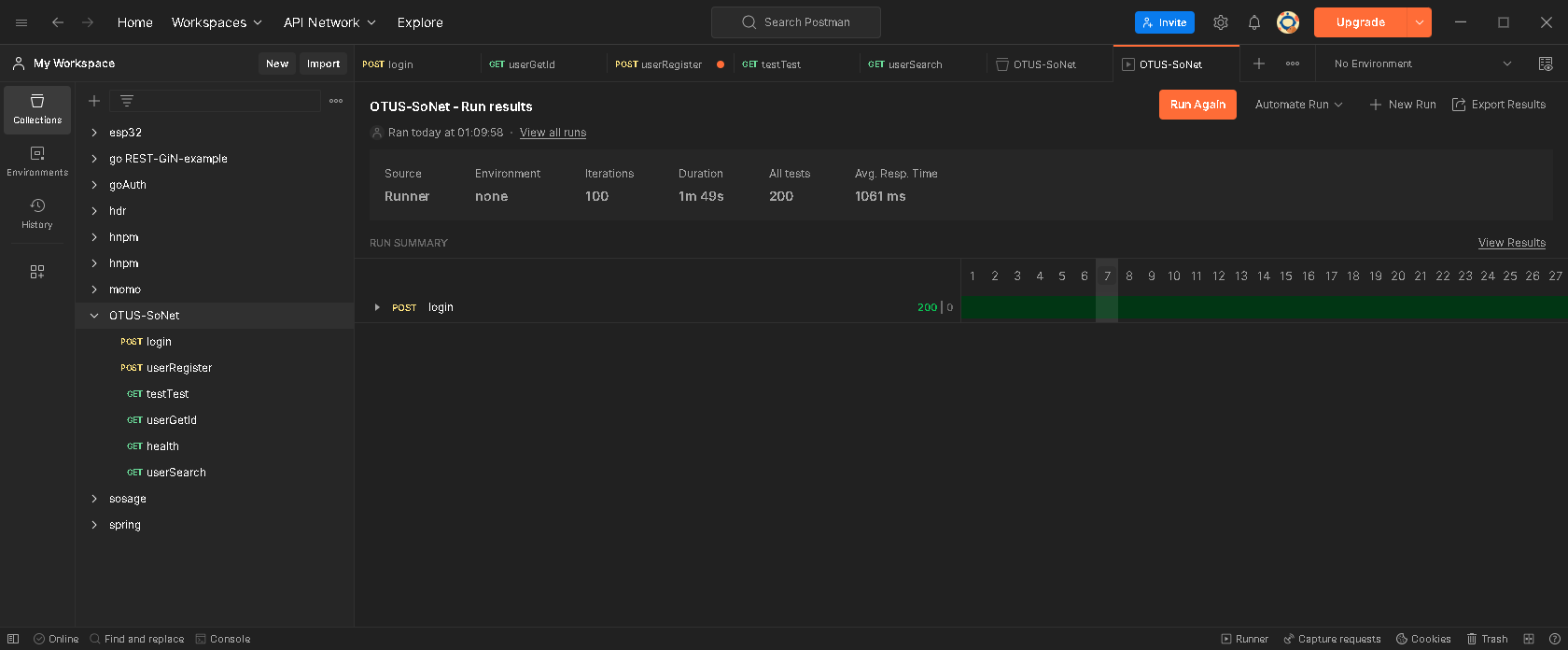
application\_name | sync\_state

------------------+------------

pg2 | quorum

pg3 | quorum

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



На реплике pg3 проверяем количество записей:

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

count

-------

100

1. 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

1. 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.