**Отчет по ДЗ №9 Балансировка и отказоустойчивость.**

Для повышения общей отказоустойчивости системы проведены следующие мероприятия:

1. в стэк Docker добавлен контейнер с образом haproxy:2.4 для балансировки между слэйвами БД PostgreSQL. Содержимое файла конфигурации haproxy.cfg ниже:

global

maxconn 100

defaults

log global

mode tcp

retries 2

timeout client 30m

timeout connect 4s

timeout server 30m

timeout check 5s

listen stats

mode http

bind \*:8404

stats enable

stats uri /

listen pgReadWrite

bind \*:5000

option pgsql-check user primaryuser

default-server inter 3s fall 3

server pg\_slave pg\_slave:5432 check port 5432

listen pgReadOnly

bind \*:5001

option pgsql-check user standbyuser

default-server inter 3s fall 3

server pg\_master pg\_master:5432 check port 5432

server pg\_asyncslave pg\_asyncslave:5432 check port 5432

2. в файле нетрудно видеть что в стэке есть один мастер с именем pg\_slave:5432, к которому мапится порт 5000, и два слэйв сервера — pg\_master:5432 и pg\_asyncslave:5432, к которому мапится 5001 порт (Прошу необращать внимание на путаницу с именами серверов — результат ДЗ по репликации.).

3. В соответствии с заданием в стэке запущены два контейнера с основным приложением (сервисы авторизации и новостей). У них аналогичные конфиги с отличием в наименовании приложений. Пример конфига приложения для app-1 ниже:

env: "local"

storage\_path: "host=**haproxy-lb** port=**5000** user=postgres password=example dbname=otus\_homework sslmode=disable"

storage\_path\_for\_read: "host=**haproxy-lb** port=**5001** user=postgres password=example dbname=otus\_homework sslmode=disable"

cache\_path: "redis://cache\_master:6379/1"

rqueue\_path: "amqp://guest:guest@rabbit:5672/"

grpc:

port: 9001

web\_port: 8087

ws\_port: 8088

timeout: 10h

app:

name: "app-1"

id: 1

, нетрудно видеть что обращение к БД на чтение и запись происходит через балансировщик haproxy-lb по портам 5001 и 5000 соответственно.

4. Для балансировки нагрузки между контейнерами с приложением используется существующий контейнер с nginx. Содержимое конфигурационного файла nginx.conf представлено ниже:

events { worker\_connections 1024; }

http {

map $http\_x\_request\_id $requestid {

default $request\_id;

}

upstream dialogs {

server ms\_dialogs:9087;

}

upstream app { # Create an upstream for the web servers

**server app-1:8087 fail\_timeout=30s max\_fails=1;**

**server app-2:8087 fail\_timeout=30s max\_fails=1;**

}

server {

listen 80;

proxy\_set\_header X-Request-ID $requestid;

add\_header X-Request-ID $requestid;

location /dialog {

proxy\_pass http://dialogs; # load balance the traffic

}

location /nginx\_status {

stub\_status;

allow 127.0.0.1;

# deny all;

}

location / {

**proxy\_read\_timeout 1s;**

**proxy\_pass http://app**; # load balance the traffic

}

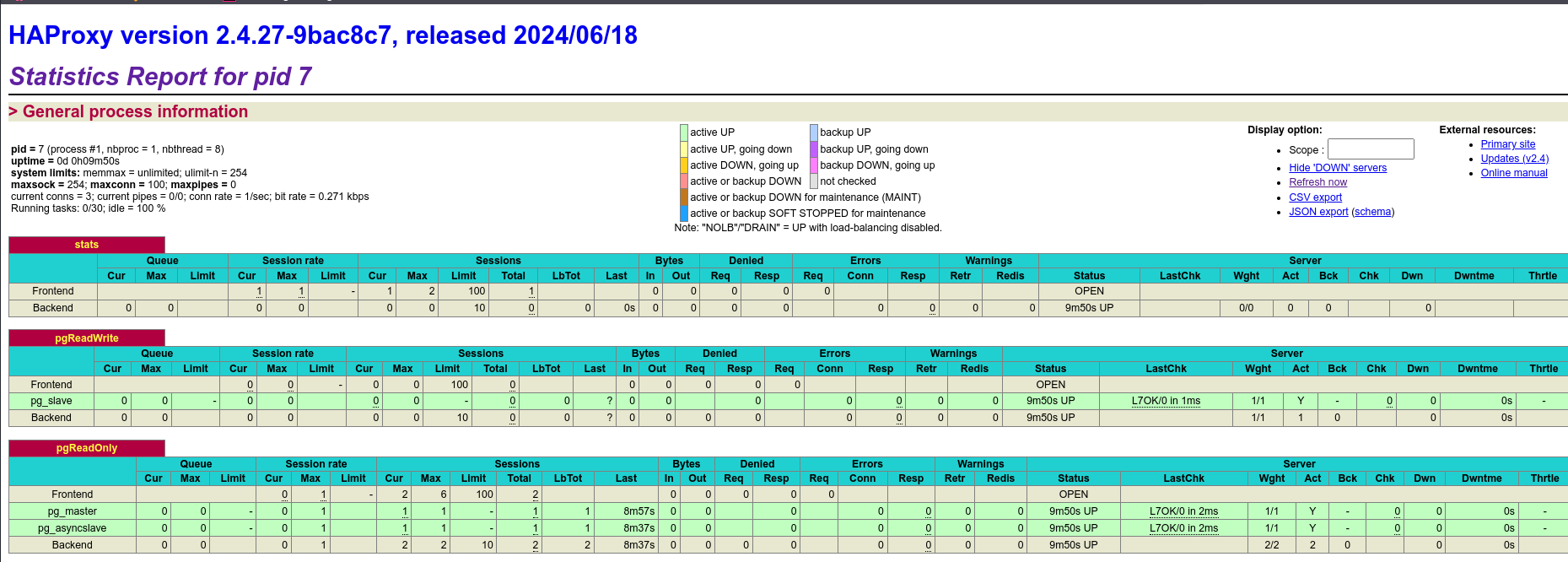
}

}

, в файле жирным шрифтом выделены настройки, позволяющие осуществлять балансировку трафика между контейнерами приложения.

5. Нагрузку генерировала, используя Postman, путем формирования GET запроса по адресу  *localhost:8080/post/get/ebbefbc4-c8c3-4c91-b067-2890a303e1b1.*

В лог файле systemSN.log нетрудно видеть, что при работе всех контейнеров запросы чередуются между app-1 и app-2 и соответственно каждый запрос от app-1/-2 балансируется между экземплярами реплик БД. Не нашла в логе где это посмотреть, поэтому приведу скрин с системы мониторинга haproxy ниже:

Рисунок 1 — Дашборд HAPROXY до отключения реплик.

Также привожу выдержку из лог файла по работе nginx и app-1/-2:

app-1 | time=2024-08-05T20:56:30.513Z level=INFO msg="request received" protocol=grpc grpc.component=server grpc.service=proto.PostService grpc.method=PostGet grpc.method\_type=unary peer.address=127.0.0.1:52728 grpc.start\_time=2024-08-05T20:56:30Z grpc.recv.duration=14.414µs grpc.request.content="post\_id:\"ebbefbc4-c8c3-4c91-b067-2890a303e1b1\""

app-1 | time=2024-08-05T20:56:30.513Z level=INFO msg="attempting to SEARCH post by post\_id" op=Post.Get post\_id=ebbefbc4-c8c3-4c91-b067-2890a303e1b1

nginx | 172.28.5.254 - - [05/Aug/2024:20:56:30 +0000] "GET /post/get/ebbefbc4-c8c3-4c91-b067-2890a303e1b1 HTTP/1.1" 200 241 "-" "PostmanRuntime/7.40.0"

app-2 | time=2024-08-05T20:56:45.326Z level=INFO msg="request received" protocol=grpc grpc.component=server grpc.service=proto.PostService grpc.method=PostGet grpc.method\_type=unary peer.address=127.0.0.1:48860 grpc.start\_time=2024-08-05T20:56:45Z grpc.recv.duration=13.278µs grpc.request.content="post\_id:\"ebbefbc4-c8c3-4c91-b067-2890a303e1b1\""

app-2 | time=2024-08-05T20:56:45.326Z level=INFO msg="attempting to SEARCH post by post\_id" op=Post.Get post\_id=ebbefbc4-c8c3-4c91-b067-2890a303e1b1

nginx | 172.28.5.254 - - [05/Aug/2024:20:56:45 +0000] "GET /post/get/ebbefbc4-c8c3-4c91-b067-2890a303e1b1 HTTP/1.1" 200 241 "-" "PostmanRuntime/7.40.0"

app-1 | time=2024-08-05T20:56:47.135Z level=INFO msg="request received" protocol=grpc grpc.component=server grpc.service=proto.PostService grpc.method=PostGet grpc.method\_type=unary peer.address=127.0.0.1:52728 grpc.start\_time=2024-08-05T20:56:47Z grpc.recv.duration=15.958µs grpc.request.content="post\_id:\"ebbefbc4-c8c3-4c91-b067-2890a303e1b1\""

app-1 | time=2024-08-05T20:56:47.135Z level=INFO msg="attempting to SEARCH post by post\_id" op=Post.Get post\_id=ebbefbc4-c8c3-4c91-b067-2890a303e1b1

nginx | 172.28.5.254 - - [05/Aug/2024:20:56:47 +0000] "GET /post/get/ebbefbc4-c8c3-4c91-b067-2890a303e1b1 HTTP/1.1" 200 241 "-" "PostmanRuntime/7.40.0"

app-2 | time=2024-08-05T20:56:47.979Z level=INFO msg="request received" protocol=grpc grpc.component=server grpc.service=proto.PostService grpc.method=PostGet grpc.method\_type=unary peer.address=127.0.0.1:48860 grpc.start\_time=2024-08-05T20:56:47Z grpc.recv.duration=19.43µs grpc.request.content="post\_id:\"ebbefbc4-c8c3-4c91-b067-2890a303e1b1\""

app-2 | time=2024-08-05T20:56:47.979Z level=INFO msg="attempting to SEARCH post by post\_id" op=Post.Get post\_id=ebbefbc4-c8c3-4c91-b067-2890a303e1b1

nginx | 172.28.5.254 - - [05/Aug/2024:20:56:47 +0000] "GET /post/get/ebbefbc4-c8c3-4c91-b067-2890a303e1b1 HTTP/1.1" 200 241 "-" "PostmanRuntime/7.40.0"

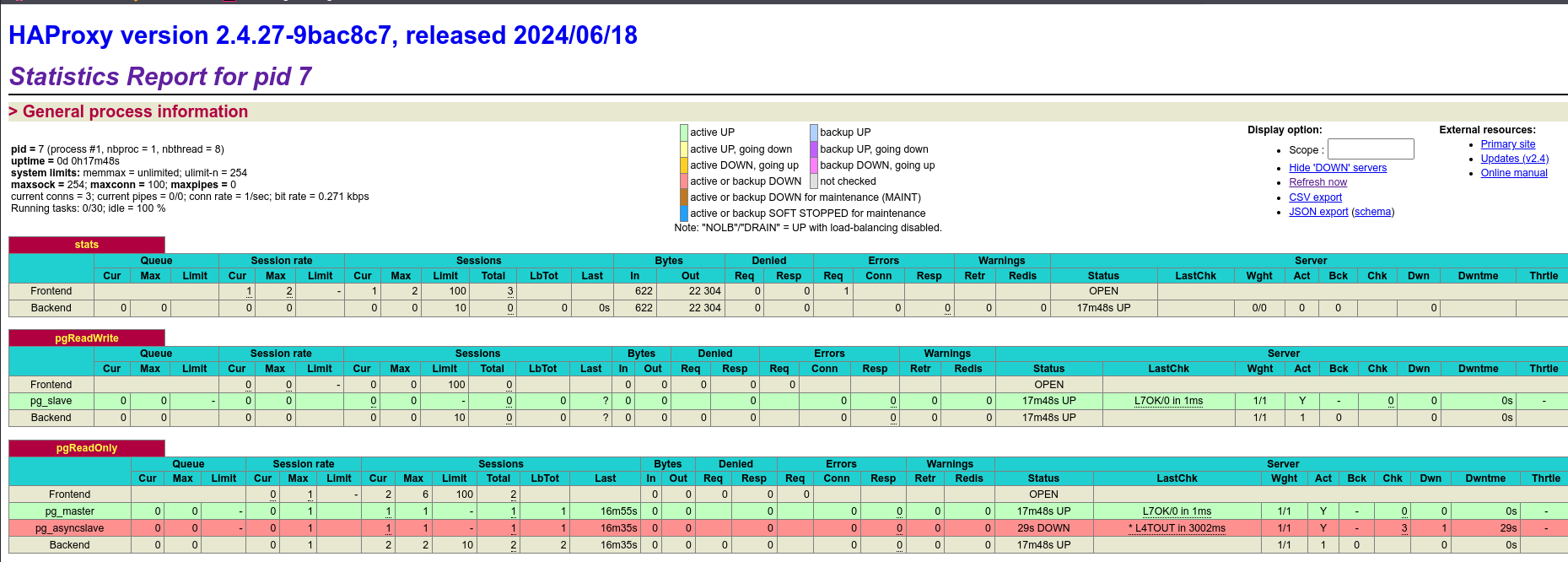
Также скрин части лога для наглядности:

Рисунок 2 — лог работы системы в штатном режиме.

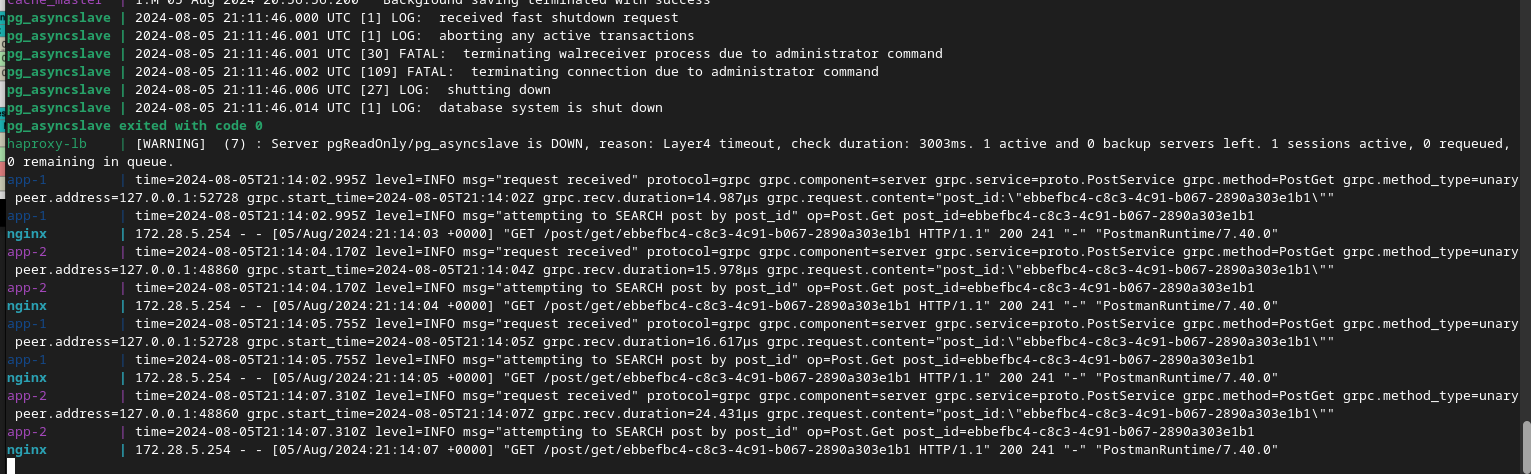
Заметила особенность что лог от приложений пишется раньше лога от nginx.

6. Далее отключаю асинхронную реплику (pg\_asynclsave) и снова делаю несколько запросов.

Принт-скрин haproxy:

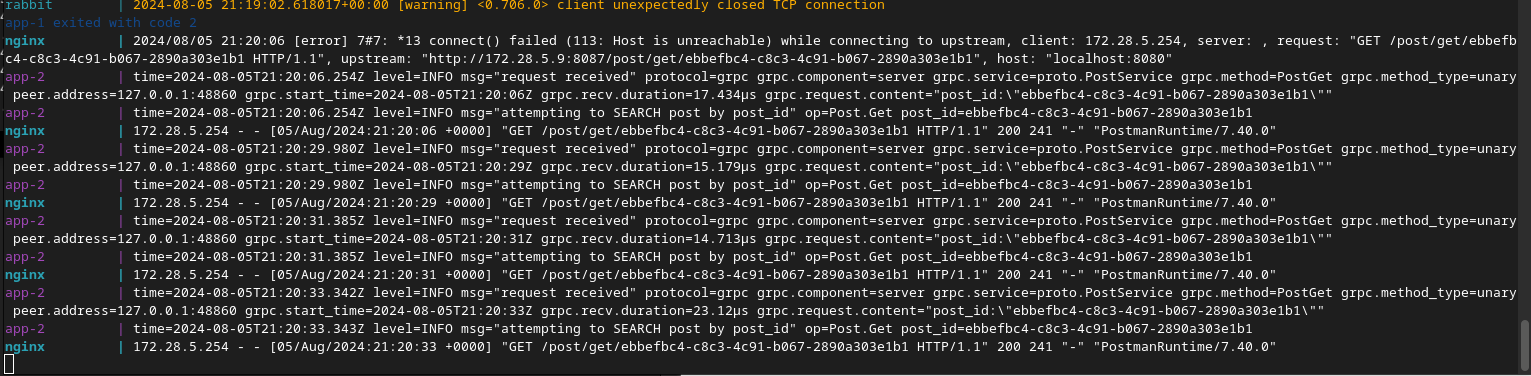
Рисунок 3 — Дашборд HAPROXY после отключения реплики.

Принт-скрин части лога:

Рисунок 4 — Лог работы после отключения реплики.

Нетрудно видеть, что запросы по прежнему проходят через оба экземпляра приложения без проблем.

7. Отключаю контейнер с приложением app-1 и делаю несколько запросов. Принт-скрин лога системы ниже:

Рисунок 5 — Лог работы системы после отключения app-1.

Также нетрудно видеть что работа происходит только через второй экземпляр приложения.

**Выводы**:

В результате выполнения ДЗ мы перепроектировали систему с целью добавления функций балансировки нагрузки и улучшения надежности путем добавления дополнительных контейнеров приложения и балансировщиков нагрузки haproxy и nginx. Проведенные мероприятия позволят нашей системе не падать, или хотя бы делать это красиво.