Репозиторий: https://github.com/artstesh/otus-highload/tree/haproxy

Ветка: haproxy

Запуск:

1. Переходим в \deployment\Replication
2. Выполняем start-replica.bat
3. Для проверки статуса репликации запускаем check-status.bat (опционально)

\*Средства мониторинга поднимаются start.bat’ом в \deployment\Monitoring

Для генерации рандомных записей необходимо выполнить get-запрос на контейнер SSO(по умолчанию 5001 порт) /User/generate?count=<количество записей>

Нагрузка проводилась с помощью k6, скрипт лежит в \replica-stress\loadtest.js

**Схема развертывания**

Клиенты → Nginx (балансировщик) → [App1, App2, App3] → HAProxy → [PostgreSQL Master, PostgreSQL Slave1, PostgreSQL Slave2]

**Конфигурация HAProxy для PostgreSQL**

**Файл конфигурации: haproxy.cfg**

global  
 daemon  
 maxconn 256  
 log 127.0.0.1 local0  
  
defaults  
 log global  
 mode tcp  
 timeout connect 5000ms  
 timeout client 50000ms  
 timeout server 50000ms  
 option tcplog  
 option tcp-check  
  
*# Бэкенд только для записи (мастер)*backend postgres\_write  
 mode tcp  
 balance first  
 option pgsql-check user postgres  
 server postgres postgres:5432 check port 5432 inter 2s fall 3 rise 2  
  
 option pgsql-check user postgres  
 tcp-check send PgSQL  
 tcp-check expect string "PostgreSQL"  
  
*# Бэкенд только для чтения (слейвы)*backend postgres\_read  
 mode tcp  
 balance roundrobin  
 option pgsql-check user postgres  
 server postgres-slave1 postgres-slave1:5432 check port 5432 inter 2s fall 3 rise 2  
 server postgres-slave2 postgres-slave2:5432 check port 5432 inter 2s fall 3 rise 2  
  
 option pgsql-check user postgres  
 tcp-check send PgSQL  
 tcp-check expect string "PostgreSQL"  
  
*# Фронтенд для операций записи (только мастер)*frontend postgres\_write\_frontend  
 bind \*:5443  
 mode tcp  
 default\_backend postgres\_write  
  
 *# Логирование для отладки* option tcplog  
 log global  
  
*# Фронтенд для операций чтения (только слейвы)*frontend postgres\_read\_frontend  
 bind \*:5442  
 mode tcp  
 default\_backend postgres\_read

**Описание конфигурации HAProxy**

1. **Разделение чтения/записи**:
   * **Порт 5443** - только для операций записи, направляется на мастер
   * **Порт 5442** - только для операций чтения, балансируется между слейвами
2. **Алгоритмы балансировки**:
   * Для записи: balance first - всегда использует первый доступный сервер (мастер)
   * Для чтения: balance roundrobin - циклическое распределение между слейвами
3. **Проверка работоспособности**:
   * проверка с помощью pgsql-check
   * Интервал проверки: 2 секунды
   * Пороги: 2 успешные проверки для восстановления, 3 неудачные для отключения

**Конфигурация Nginx для балансировки приложений**

**Файл конфигурации: nginx.conf**

events {  
 worker\_connections 1024*;*}  
  
http {  
 ssl\_protocols TLSv1.2 TLSv1.3*;* ssl\_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384*;* ssl\_prefer\_server\_ciphers off*;* ssl\_session\_cache shared:SSL:10m*;* ssl\_session\_timeout 10m*;* upstream api\_backend {  
 least\_conn*;* server sso1:80 max\_fails=3 fail\_timeout=5s*;* server sso2:80 max\_fails=3 fail\_timeout=5s*;* server sso3:80 max\_fails=3 fail\_timeout=5s*;* }  
  
 server {  
 listen 443 ssl*;  
  
 # SSL сертификаты* ssl\_certificate /etc/nginx/ssl/nginx-selfsigned.crt*;* ssl\_certificate\_key /etc/nginx/ssl/nginx-selfsigned.key*;* add\_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always*;* add\_header X-Content-Type-Options nosniff*;* add\_header X-Frame-Options DENY*;* add\_header X-XSS-Protection "1; mode=block"*;* location / {  
 proxy\_pass http://api\_backend*;* proxy\_http\_version 1.1*;* proxy\_set\_header Upgrade $http\_upgrade*;* proxy\_set\_header Connection keep-alive*;* proxy\_set\_header Host $host*;* proxy\_set\_header X-Real-IP $remote\_addr*;* proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for*;* proxy\_set\_header X-Forwarded-Proto $scheme*;* proxy\_cache\_bypass $http\_upgrade*;  
  
 # Таймауты* proxy\_connect\_timeout 5s*;* proxy\_send\_timeout 5s*;* proxy\_read\_timeout 5s*;* }

}

**Описание конфигурации Nginx**

1. **Алгоритм балансировки**: Least Connections
2. **HTTPS поддержка**: Самоподписанные сертификаты с HTTP to HTTPS редиректом
3. **Настройки проверки работоспособности**:
   * Максимум 3 неудачи перед временным исключением
   * Время исключения: 5 секунд
4. **Таймауты**: 5 секунд

**Условия эксперимента**

**Тестовое окружение**

* **Количество инстансов приложения**: 3
* **Количество узлов БД**: 1 мастер + 2 слейва
* **Инструмент нагрузки**: k6
* **Метод отказа**: kill -9 для процессов PostgreSQL и приложений

**Сценарий тестирования**

1. **Начальное состояние**: Все компоненты системы работают
2. **Фаза нагрузки**: Постоянные запросы к API (/User/search)
3. **Отказ слейва БД**: Принудительное завершение PostgreSQL slave1
4. **Отказ инстанса приложения**: Принудительное завершение одного из контейнеров приложения
5. **Мониторинг**: Наблюдение за автоматическим восстановлением

**Логи работы системы и анализ**

**Логи HAProxy при отказе слейва**

[WARNING] (8) : Server postgres\_read/postgres-slave1 is UP, reason: Layer7 check passed, code: 0, info: "PostgreSQL server is ok", check duration: 0ms. 1 active and 0 backup servers online. 0 sessions requeued, 0 total in queue.

[WARNING] (8) : Server postgres\_read/postgres-slave2 is UP, reason: Layer7 check passed, code: 0, info: "PostgreSQL server is ok", check duration: 0ms. 2 active and 0 backup servers online. 0 sessions requeued, 0 total in queue.

[WARNING] (8) : Server postgres\_read/postgres-slave1 is DOWN, reason: Layer7 timeout, check duration: 2001ms. 1 active and 0 backup servers left. 1 sessions active, 0 requeued, 0 remaining in queue.

**Анализ логов HAProxy**:

1. **Начальное состояние**: Оба слейва (slave1 и slave2) в состоянии UP
2. **Обнаружение отказа**: HAProxy обнаружил timeout при проверке slave1
3. **Время обнаружения**: 2001ms (чуть больше 2 секунд - интервал проверки)
4. **Автоматическое восстановление**: Оставшийся slave2 продолжает обслуживать запросы

**Логи Nginx при отказе инстанса приложения**

2025/10/10 12:27:31 [error] 29#29: \*74 upstream timed out (110: Operation timed out) while reading response header from upstream, client: 172.31.0.1, server: , request: "GET /User/search?firstName=Jad&lastname=Her HTTP/1.1", upstream: "http://172.31.0.6:80/User/search?firstName=Jad&lastname=Her", host: "localhost:5001"

**Анализ логов Nginx**:

1. **Тип ошибки**: Timeout при чтении ответа от upstream
2. **Запрос**: GET /User/search (операция чтения)
3. **Upstream**: 172.31.0.6:80 (один из инстансов приложения)
4. **Причина**: Приложение не ответило в течение 5 секунд (proxy\_read\_timeout)
5. **Автоматическое восстановление**: Nginx пометил инстанс как нерабочий и перенаправил запросы на другие инстансы

**Результаты тестирования**

**Показатели отказоустойчивости**

| Компонент | Время обнаружения | Время восстановления | Влияние на систему |
| --- | --- | --- | --- |
| PostgreSQL Slave | 2-3 секунды | Мгновенно | Минимальное, чтение продолжается на оставшемся слейве |
| Инстанс приложения | До 5 секунд | Мгновенно | Временные таймауты, затем автоматическое переключение |

**Выводы**

**1. Эффективность HAProxy**

* **Обнаружение отказов**: 2-3 секунды
* **Разделение чтения/записи**: Успешно работает через разные порты
* **Балансировка чтения**: Round-robin между слейвами корректно распределяет нагрузку

**2. Эффективность Nginx**

* **Балансировка нагрузки**: Least Connections алгоритм эффективно распределяет запросы
* **Обнаружение отказов**: До 5 секунд (зависит от таймаутов)
* **HTTPS терминация**: Успешно обрабатывает SSL/TLS соединения

**3. Отказоустойчивость системы**

* **При отказе слейва БД**: Система продолжает работу с минимальной деградацией
* **При отказе инстанса приложения**: Автоматическое переключение на здоровые инстансы
* **Общая доступность**: 100% в условиях тестирования с отказами