**HAProxy** (High Availability Proxy) – это популярный открытый балансировщик нагрузки TCP/HTTP и прокси-сервер для систем Linux, Solaris и FreeBSD. Он позволяет улучшить производительность и ошибкоустойчивость серверного окружения путём распределения рабочей нагрузки между несколькими серверами (например, между веб-сервером, сервером баз данных и сервером приложений). HAProxy используют многие проекты с высокой нагрузкой: GitHub, Imgur, Instagram и Twitter.

Данная статья ознакомит вас с основными понятиями HAProxy и балансировки нагрузки. Также здесь вы найдёте примеры настроек и полезные советы, которые помогут вам повысить производительность и надёжность серверной среды.

**Терминология HAProxy**

В балансировке нагрузки и проксировании существует много терминов и понятий. Рассмотрим самые основные из них.

**Списки управления доступом (ACL)**

В балансировке нагрузки списки управления доступом используются для проверки определённых условий и выполнения действий (например, для выбора сервера, блокировки запроса и т.п.) согласно результату этой проверки. Списки контроля доступа обеспечивают гибкую переадресацию сетевого трафика на основе различных факторов (например, с учётом количества подключений к бэкэнду).

ACL выглядит так:

acl url\_blog path\_beg /blog

Такой ACL выполняется, если путь запроса пользователя начинается с /blog (например, http://yourdomain.com/blog/blog-entry-1).

**Бэкэнд**

Бэкэнд – это набор серверов, которые обрабатывают запросы. Настройки этих серверов находятся в разделе backend конфигурационного файла HAProxy. В основном бэкэнд определяется:

* алгоритмом балансировки нагрузки;
* и списком серверов и портов.

Бэкэнд может состоять из одного или нескольких серверов. В целом, чем больше на бэкэнде серверов, тем он надёжнее и тем выше потенциальная мощность нагрузки. За счет распределения нагрузки между несколькими серверами растёт производительность и отказоустойчивость приложения.

Так выглядит настройка двух бэкэндов (web-backend и blog-backend) с двумя серверами в каждом.

backend web-backend  
balance roundrobin  
server web1 web1.yourdomain.com:80 check  
server web2 web2.yourdomain.com:80 check  
backend blog-backend  
balance roundrobin  
mode http  
server blog1 blog1.yourdomain.com:80 check  
server blog1 blog1.yourdomain.com:80 check

* Строка balance roundrobin определяет алгоритм балансировки нагрузки.
* Строка mode http определяет уровень балансировки (в данном случае это уровень 7).
* Опция check в конце директив server настраивает проверку состояния серверов бэкэнда.

**Фронтэнд**

Фронтэнд определяет, каким образом запросы должны направляться на бэкэнд. Настройки фронтэнда определены в разделе frontend конфигурации HAProxy и состоят из следующих компонентов:

* Наборы IP-адресов и портов (например, 10.1.1.7:80, \*:443);
* Списки управления доступом.
* Правила use\_backend, которые определяют, какой бэкэнд использовать в зависимости от условий ACL.
* Правила default\_backend которые обрабатывают случаи, не отвечающие предыдущим правилам.

Фронтэнд можно настроить для поддержки различных типов сетевого трафика, о чём и пойдёт речь в следующем разделе.

**Типы балансировки нагрузки**

Теперь рассмотрим базовые типы балансировки нагрузки.

**Среда без балансировки нагрузки**

Среда простого приложения без балансировки нагрузки выглядит так:

Пользователь → Интернет → Веб-сервер → Сервер баз данных

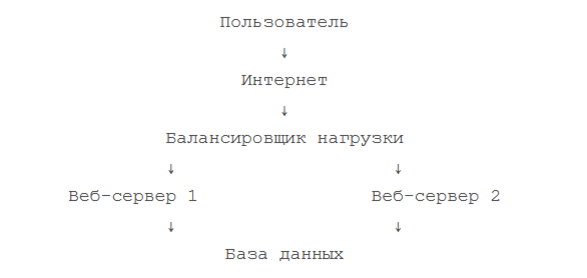
То есть пользователь подключается непосредственно к веб-серверу (к вашему домену, yourdomain.com). И если этот единственный веб-сервер прекращает работу, пользователь не сможет попасть на сайт. Кроме того, если множество пользователей одновременно попытается открыть сайт, сервер может попросту не справиться с нагрузкой: сайт будет загружаться очень медленно или же пользователь вовсе не сможет открыть его.

**Балансировка нагрузки на уровне 4**

Самый простой способ балансировки нагрузки сетевого трафика между несколькими серверами – это балансировка на уровне 4 (также он называется транспортным уровнем).

Такая балансировка будет распределять трафик на основе диапазонов IP-адресов и портов. К примеру, если запрос поступает от http://yourdomain.com/anything, трафик будет отправлен на бэкэнд, который отвечает за поддержку yourdomain.com на порте 80.

Вот общая схема балансировки нагрузки на транспортном уровне:

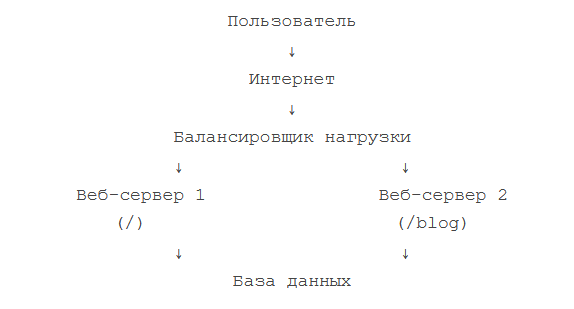


В таком случае каждый запрос сначала попадает к балансировщику, который направляет его к одному из серверов бэкэнда; этот сервер, в свою очередь, отвечает на запрос пользователя и передаёт ему результат. Как правило, все серверы бэкэнда обслуживают одинаковый контент: в противном случае пользователь может получить непоследовательное или нецелостное отображение. Обратите внимание, что оба веб-серверы подключены к одному серверу баз данных.

**Балансировка нагрузки на уровне 7**

Другой, более сложный метод настройки балансировки нагрузки – это балансировка на уровне 7 (уровень приложения). В таком случае балансировщик распределяет запросы между серверами бэкэнда на основе содержания запроса пользователя. Этот режим балансировки нагрузки позволяет запускать несколько серверов приложений на одном домене или порте.

Общая схема балансировки нагрузки на уровне приложения имеет такой вид:



В данном случае если пользователь запрашивает yourdomain.com/blog, он будет направлен на веб-сервер 2. Остальные запросы передаются на бэкэнд, который может обслуживать другое приложение. В данном случае оба бэкэнда используют один и тот же сервер базы данных.

Настройка фронтэнда может выглядеть так:

frontend http  
bind \*:80  
mode http  
acl url\_blog path\_beg /blog  
use\_backend blog-backend if url\_blog  
default\_backend web-backend

Это настроит фронтэнд http, который поддерживает входящий трафик на порте 80.

* acl url\_blog path\_beg /blog соответствует запросам, которые начинаются с /blog.
* use\_backend blog-backend if url\_blog использует ACL для проксирования трафика на серверы blog-backend.
* default\_backend web-backend обслуживает остальной трафик, который будет отправлен на web-backend.

**Алгоритмы балансировки нагрузки**

Далее балансировщик выбирает из пула один из серверов согласно одному из алгоритмов. HAProxy поддерживает несколько алгоритмов балансировки.

Помимо алгоритмов балансировки нагрузки, серверы отбираются с помощью параметра weight, который определяет, как часто может использоваться тот или иной сервер по сравнению с другими серверами.

Наиболее распространёнными являются эти алгоритмы:

1. Согласно алгоритму Round Robin (или алгоритм кругового обслуживания) серверы получают трафик последовательно. Балансировщик выбирает первый сервер в списке и передаёт ему первый запрос, второй сервер получит второй запрос и т.д.
2. Балансировщик выбирает для обслуживания трафика наименее загруженный сервер (то есть сервер с наименьшим количеством подключений на текущий момент). Такой алгоритм особенно полезен, если для обслуживания трафика нужны длинные сессии.
3. Балансировщик нагрузки выбирает сервер на основе хэша исходного IP запроса (например, на основе IP-адреса посетителя). При этом все запросы конкретного пользователя будут обслуживаться одним и тем же сервером бэкэнда.

**Липкие сессии**

Некоторым приложениям необходимо, чтобы в течение работы пользователь оставался подключенным к одному и тому же серверу бэкэнда. Алгоритм, использующий IP-адрес посетителя, может обеспечить это условие. Также привязать сессию к серверу можно с помощью так называемых липких сессий (sticky sessions): при этом балансировщик устанавливает куки, и после этого все запросы этой сессии будут направлены на один и тот же физический сервер.

**Проверка состояния сервера**

HAProxy должен отправлять трафик только на те активные серверы, которые в состоянии его обслужить. для этого он выполняет проверку состояния – health check, что позволяет избежать настройки бэкэнда вручную. По умолчанию проверка состояния проводится путём попытки установить TCP-соединение с сервером и, таким образом, убедиться, что он прослушивает IP-адрес и порт.

Если сервер не проходит проверку балансировщика, он автоматически удаляется из пула серверов и не будет получать трафик, пока не пройдёт проверку. Если все серверы бэкэнда не прошли проверку, сервис становится недоступным до тех пор, пока не восстановится хотя бы один сервер бэкэнда.

**Другие средства балансировки**

Если HAProxy – слишком сложное решение на данном этапе, вы можете выбрать более простое средство:

* Linux Virtual Servers (LVS) – простой и быстрый балансировщик нагрузки на транспортном уровне, включенный во многие дистрибутивы Linux.
* Nginx – быстрый и надёжный веб-сервер, который можно использовать в качестве обратного прокси-сервера и балансировщика нагрузки. Nginx часто используется в связке с HAProxy.

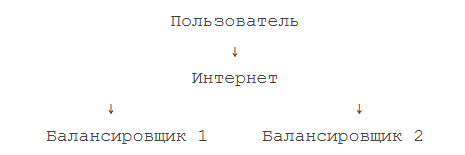
**Высокая доступность**

При балансировке на уровне 4 и 7 трафик направляется на один из нескольких серверов бэкэнда. Но в такой ситуации балансировщик нагрузки становится единой точкой отказа: если он выйдет из строя, приложение будет недоступно.

Инфраструктуры, в которых нет единой точки отказа, обладают высокой доступностью. Это значит, что в случае отказа любого компонента инфраструктуры приложение будет работать.

Получается, что в вышеописанных примерах балансировщик нагрузки устраняет единую точку отказа с бэкэнда, но при этом сам становится этой точкой.

Чтобы «обезвредить» балансировщик как единую точку отказа, вы можете добавить в инфраструктуру второй балансировщик и создать кластер, в котором они будут проверять состояние друг друга. Каждый из них в равной степени способен обнаруживать и восстанавливать сбои.



Если первый балансировщик перестанет работать, DNS передаст пользователей второму балансировщику. Изменения DNS иногда требуют много времени. Чтобы ускорить и автоматизировать переключение между балансировщиками, многие администраторы используют системы, которые поддерживают плавающие IP-адреса (или другие технологии преобразования IP). Эти технологии позволяют устранить некоторые проблемы, возникающие во время изменений DNS, предоставляя статичные IP-адреса, которые в дальнейшем при необходимости можно преобразовать. Доменное имя может быть связано с тем же IP, в то время как сам IP-адрес перемещается между серверами бэкэнда.