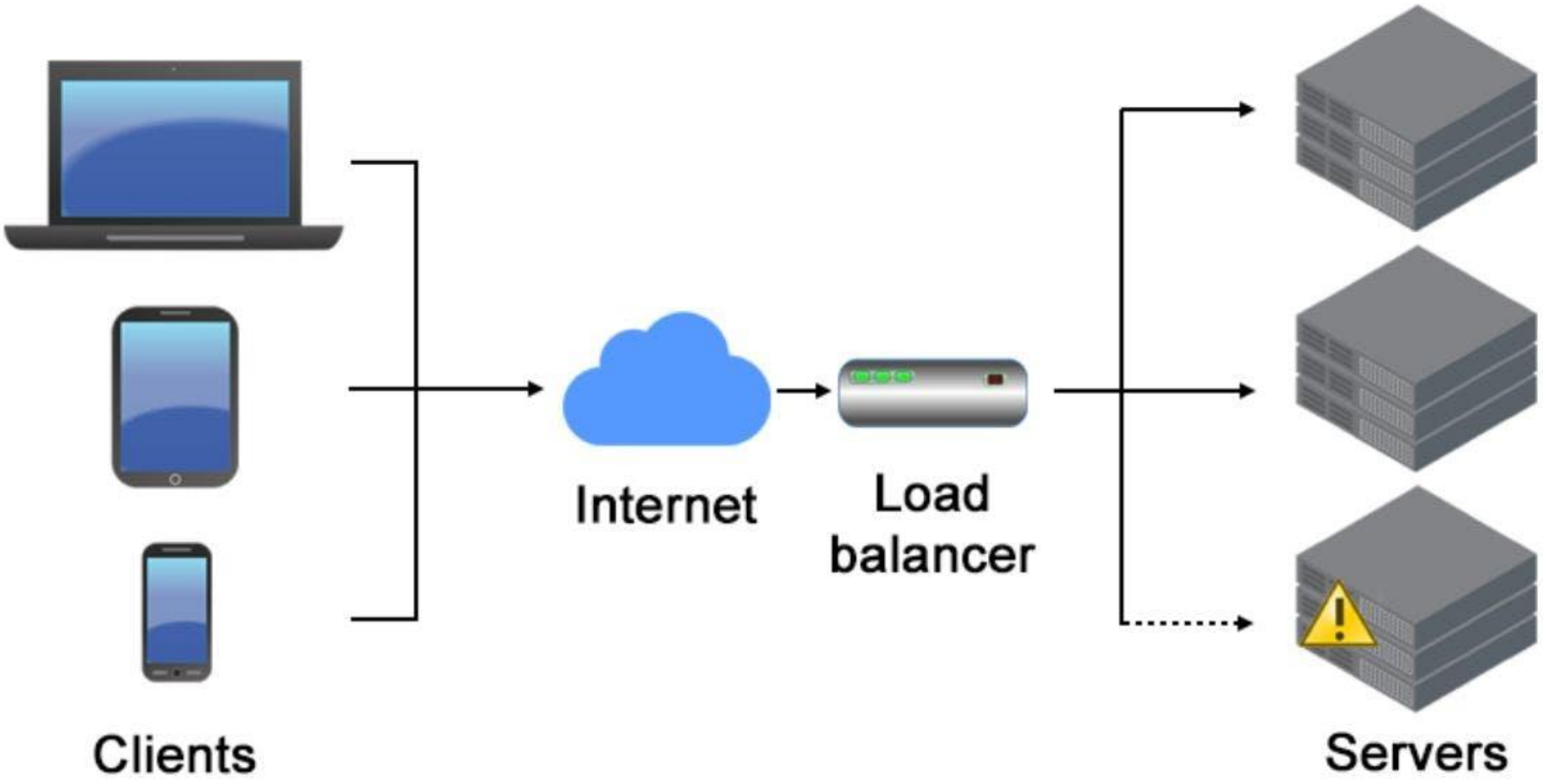
Тема:
Основные сервисы на Linux .
Балансировка нагрузки.



План занятия:

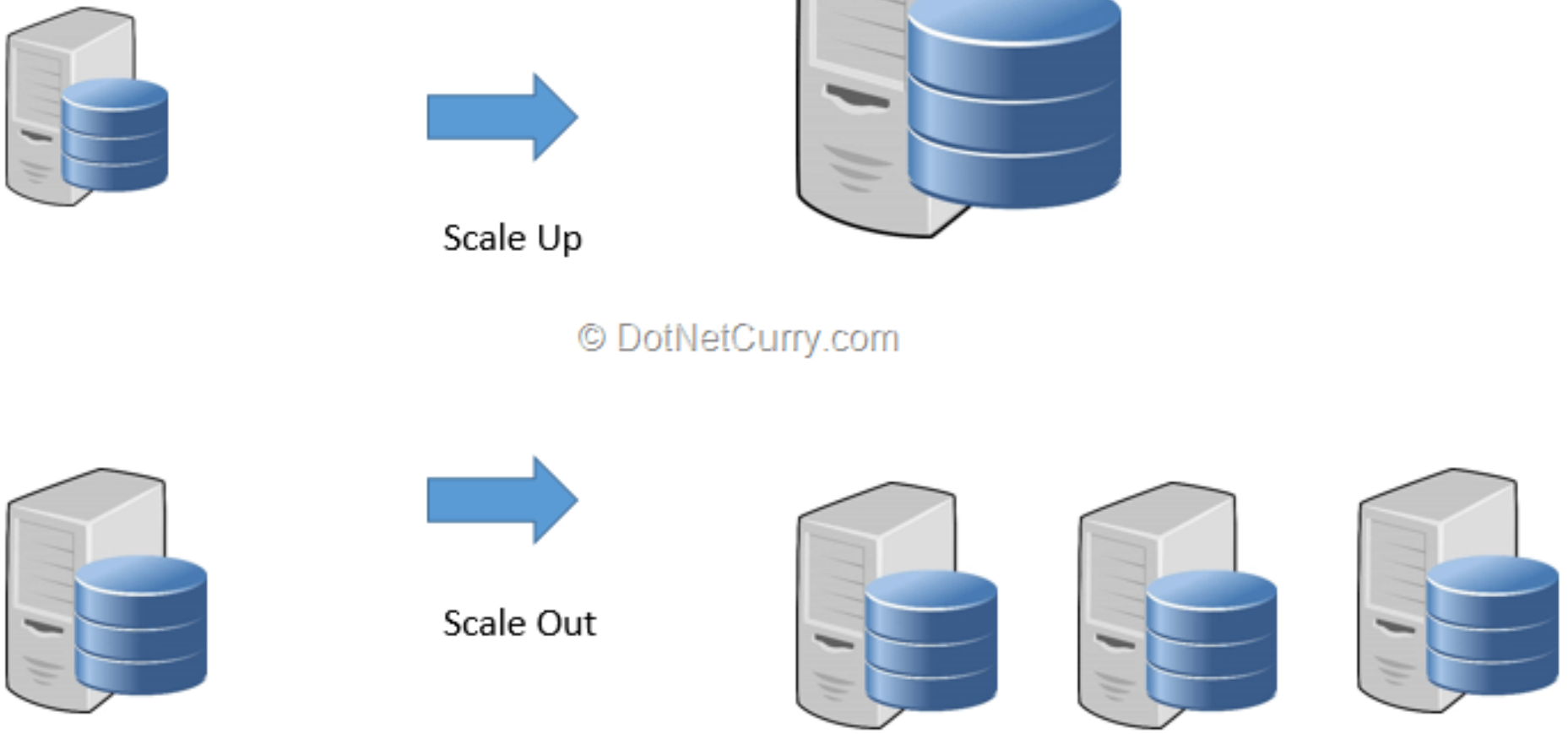
1. Масштабирование. Scale UP, Scale OUT. Stateless, Statefull.
2. Балансировка нагрузки.
3. Утилита ipvsadm.

1. Масштабирование. Scale UP, Scale OUT. Stateless, Statefull.

Масштабирование — это процесс адаптации системы под изменяющуюся нагрузку путём увеличения её вычислительных ресурсов.

Существуют два основных подхода к масштабированию: **Scale Up** (вертикальное масштабирование) и **Scale Out** (горизонтальное масштабирование).

Оба метода имеют свои преимущества и недостатки и могут применяться в зависимости от специфики задач и архитектуры системы.


Scale Up
© DotNetCurry.com
Scale Out

Scale UP (вертикальное масштабирование):

Описание: При вертикальном масштабировании система увеличивает свою производительность путем добавления ресурсов на одном узле или сервере.

Применение: Scale UP подходит в случаях, когда требуется улучшить производительность одного сервера или узла, добавляя больше процессоров, памяти, дискового пространства или других ресурсов. Это может быть полезно, когда приложение имеет ограничения в однопоточной производительности или когда требуется обработка больших объемов данных на одном сервере.

Преимущества:

Простота управления: Масштабирование происходит на одном сервере, что упрощает управление и администрирование системы.

Меньшие затраты на инфраструктуру: Вертикальное масштабирование не требует дополнительных серверов или сетевой инфраструктуры.

Ограничения:

Физические ограничения: Возможно, достигнуто предельное количество ресурсов, которое можно добавить на одном сервере.

Одна точка отказа: Если сервер выходит из строя, вся система может быть недоступна.

Scale OUT (горизонтальное масштабирование):

Описание: При горизонтальном масштабировании система увеличивает свою производительность путем добавления дополнительных узлов или серверов.

Применение: Горизонтальное масштабирование используется, когда требуется повысить общую пропускную способность и отказоустойчивость системы. Путем добавления новых серверов и распределения нагрузки между ними можно достичь более высокой масштабируемости и поддерживать работоспособность системы даже при отказе отдельных узлов.

Преимущества:

Высокая масштабируемость: Добавление новых серверов позволяет системе линейно увеличивать пропускную способность и обработку запросов.

Высокая отказоустойчивость: При отказе одного сервера остальные серверы могут продолжать работу, обеспечивая доступность системы.

Ограничения:

Усложненное управление: Требуется управление и согласование между несколькими серверами.

Дополнительные затраты: Добавление новых серверов требует дополнительных затрат на инфраструктуру, сетевое оборудование и управление.

Доступность (Availability) относится к свойству информационной системы или сервиса быть доступным и функционирующим в течение определенного времени. В более простых терминах, доступность отражает готовность системы отвечать на запросы пользователей и обеспечивать доступ к своим функциональным возможностям.

Доступность обычно измеряется в процентах и выражает время, в течение которого система доступна, по отношению к полному времени наблюдения.

Для достижения высокой доступности системы применяются различные подходы и методы, включая:

Дублирование и репликацию компонентов или серверов для обеспечения резервирования и отказоустойчивости.

Использование механизмов балансировки нагрузки для распределения запросов между несколькими серверами или ресурсами.

Резервное копирование данных и системное восстановление для снижения риска потери данных и быстрого восстановления после отказа.

Мониторинг и предупреждение о состоянии системы для быстрого обнаружения проблем и реагирования на них.

Способы управления состоянием **Stateful и Stateless.**

Stateless приложения

Stateless приложения не сохраняют информацию о состоянии пользователя или сессии между запросами. Каждый запрос от клиента к такому приложению является самодостаточным и не зависит от предыдущих запросов или сессий. Это означает, что любой сервер может обработать любой запрос, так как для выполнения запроса не требуется знание о предыдущих взаимодействиях.

Преимущества:

•Проще в разработке и развертывании.

•Легко масштабируются, поскольку новые экземпляры могут быть добавлены без необходимости синхронизации состояний.

•Лучше подходят для облачных вычислений и безсерверных архитектур.

Примеры:

•Веб-сервисы, предоставляющие REST API.

•Статические веб-сайты.

Stateful приложения

Stateful приложения сохраняют информацию о состоянии пользователя или сессии между запросами. Это означает, что для корректной обработки запроса сервер должен иметь доступ к данным о предыдущих взаимодействиях с клиентом. Такие приложения требуют специального механизма для хранения и доступа к состоянию, что может включать базы данных, сессии в памяти, файлы cookie и т. д.

Преимущества:

•Позволяют предоставлять более персонализированный пользовательский опыт.

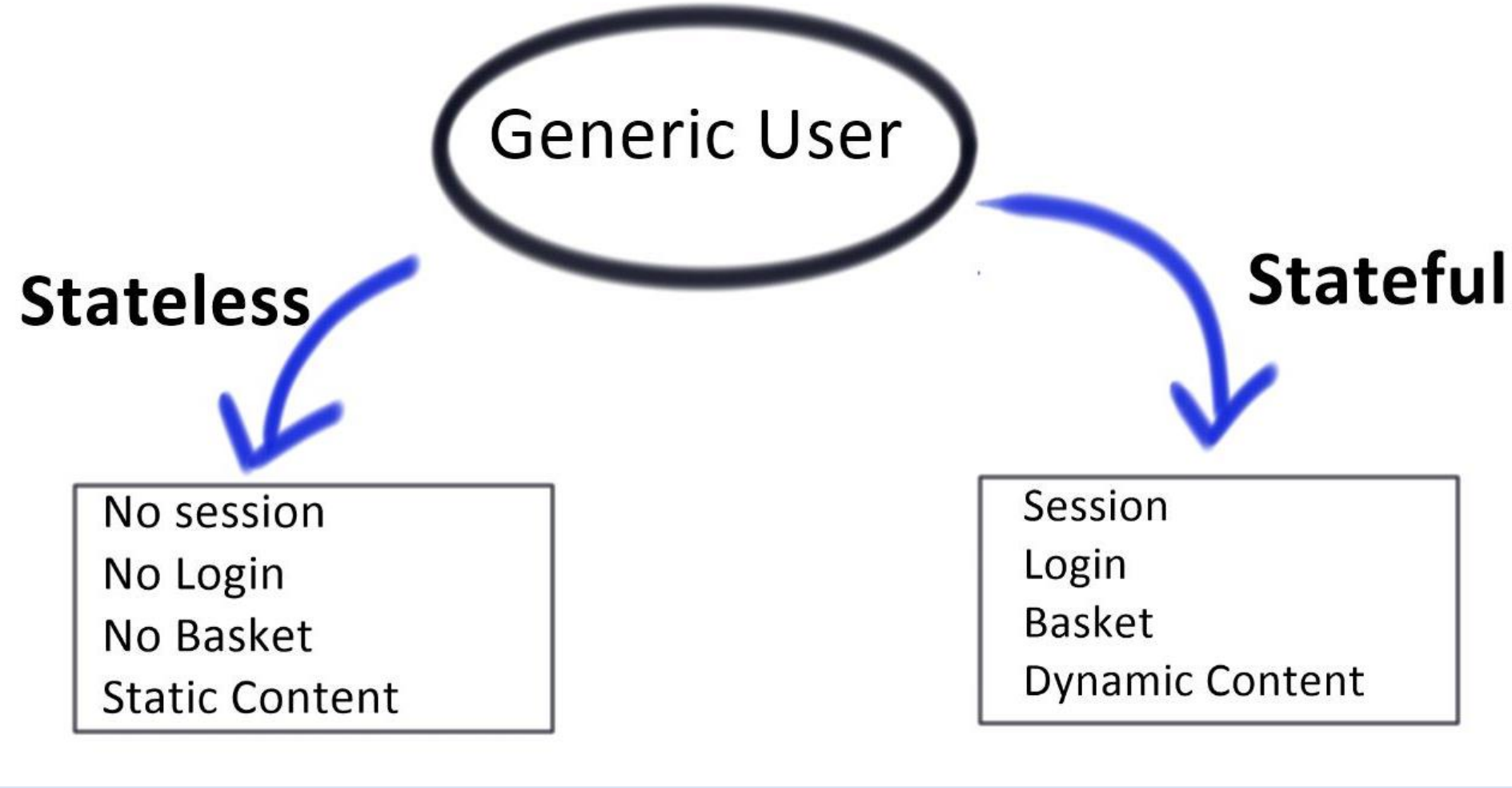
•Необходимы для определенных типов приложений, где требуется постоянство информации о состоянии (например, онлайн-игры, интернет-банкинг).

Примеры:

•Веб-приложения, требующие аутентификации и сессии пользователя.

•Приложения для интернет-магазинов с корзиной покупок.

•Базы данных, приложения электронной почты.


Generic User
Stateless
Stateful
No session
No Login
No Basket
Static Content
Session
Login
Basket
Dynamic Content

1. Балансировка нагрузки.

Балансировка нагрузки между серверами — это процесс распределения входящих сетевых или прикладных запросов по нескольким серверам.

Цель балансировки — увеличение доступности и надёжности компонентов системы, уменьшение времени отклика и улучшение общей производительности за счёт параллельной обработки запросов.

Для реализации балансировки используются специальные устройства или программное обеспечение.

Типы балансировщиков нагрузки

- **Аппаратные балансировщики**: Специализированные устройства, оптимизированные для балансировки нагрузки. Они могут обеспечивать высокую производительность, но их стоимость значительно выше программных аналогов.
- **Программные балансировщики**: ПО, устанавливаемое на обычные серверы или виртуальные машины.

Уровни балансировки.

Процедура балансировки осуществляется при помощи целого комплекса алгоритмов и методов, соответствующим следующим уровням модели OSI:

сетевому;

транспортному;

прикладному.

Балансировка на сетевом уровне предполагает решение следующей задачи: нужно сделать так, чтобы за один конкретный IP-адрес сервера отвечали разные физические машины. Такая балансировка может осуществляться с помощью множества разнообразных способов.

DNS-балансировка.

Балансировка по IP с использованием дополнительного маршрутизатора.

Балансировка по территориальному признаку.

Балансировка на транспортном уровне. Клиент обращается к балансировщику, тот перенаправляет запрос одному из серверов, который и будет его обрабатывать. Выбор сервера, на котором будет обрабатываться запрос, может осуществляться в соответствии с самыми разными алгоритмами (об этом ещё пойдёт речь ниже): путём простого кругового перебора, путём выбора наименее загруженного сервера из пула и т.п.

Балансировка на прикладном уровне. При балансировке на прикладном уровне балансировщик работает в режиме «умного прокси». Он анализирует клиентские запросы и перенаправляет их на разные серверы в зависимости от характера запрашиваемого контента.

Алгоритмы и методы балансировки

Равномерное распределение нагрузки (**Round Robin**):

Описание: Запросы равномерно распределяются между серверами по круговому принципу.

Как работает: Каждый новый запрос направляется на следующий сервер в списке, и после достижения последнего сервера обход начинается сначала.

Преимущества: Простота реализации, равномерное распределение нагрузки между серверами.

Взвешенное распределение нагрузки (**Weighted Round Robin**):

Описание: Запросы распределяются между серверами с учетом их весов или пропорций.

Как работает: Каждый сервер имеет ассоциированный вес, который определяет долю нагрузки, которую он должен обрабатывать. Запросы направляются на серверы в соответствии с их весами.

Преимущества: Возможность управления распределением нагрузки с учетом производительности и возможностей каждого сервера.

Алгоритм наименьшей загрузки (**Least Connection**):

Описание: Запросы направляются на сервер с наименьшей активной загрузкой или наименьшим количеством активных соединений.

Как работает: Балансировщик нагрузки отслеживает количество активных соединений на каждом сервере и направляет новые запросы на сервер с наименьшей нагрузкой.

Преимущества: Равномерное распределение нагрузки на основе текущей загрузки серверов.

Алгоритм наименьшей задержки (**Least Response Time**):

Описание: Запросы направляются на сервер с наименьшей задержкой ответа или наиболее быстрым временем обработки.

Как работает: Балансировщик нагрузки измеряет время отклика каждого сервера и направляет запросы на сервер с наименьшей задержкой ответа.

Преимущества: Маршрутизация запросов к наиболее отзывчивым серверам, что может улучшить время отклика для конечных пользователей.

Алгоритм хэширования (**Hash-based**):

Описание: Запросы маршрутизуются на основе хэш-функции, использующей определенные атрибуты запроса, такие как IP-адрес клиента или URL.

Как работает: Хэш-функция применяется к атрибутам запроса, и результат хэша определяет, на какой сервер будет направлен запрос.

Преимущества: Постоянное направление запросов с одинаковыми атрибутами **на один и тот же сервер**, что полезно для сохранения состояния или кэширования.

3. Утилита ipvsadm

ipvsadm (IP Virtual Server Administration) - это утилита командной строки для управления и настройки балансировщика нагрузки IP Virtual Server (IPVS) в ядре Linux. IPVS представляет собой модуль ядра Linux, предоставляющий функциональность балансировки нагрузки на уровне транспортного (Transport) или прикладного (Application) уровня модели OSI.

Утилита ipvsadm поддерживает различные алгоритмы балансировки нагрузки для настройки IP Virtual Server (IPVS) в ядре Linux:

Round Robin (rr). Алгоритм rr равномерно распределяет запросы между реальными серверами в пуле.

Weighted Round Robin (wrr). Алгоритм wrr распределяет запросы между реальными серверами в пуле с учетом их весов.

Least Connection (lc). Алгоритм lc направляет запросы на реальный сервер с наименьшим количеством активных соединений.

Weighted Least Connection (wlc). Алгоритм wlc балансирует нагрузку, учитывая как веса реальных серверов, так и количество их активных соединений.

Destination Hashing (dh). Алгоритм dh маршрутизирует запросы на реальные серверы на основе хэш-функции, применяемой к IP-адресу или порту назначения.

Source Hashing (sh).Алгоритм sh маршрутизирует запросы на реальные серверы на основе хэш-функции, применяемой к IP-адресу или порту отправителя.

Основные этапы настройки ipvsadm в Linux:

Установка ipvsadm:

Убедитесь, что у вас установлен пакет ipvsadm на вашей системе. Вы можете использовать менеджер пакетов своего дистрибутива Linux для установки пакета. Например, для систем на основе Debian или Ubuntu можно выполнить следующую команду:

sudo apt-get install ipvsadm

```
sudo apt-get install ipvsadm
```

Загрузка модулей ядра:

Убедитесь, что модули ядра, необходимые для IPVS, загружены. (/proc/modules)

В большинстве дистрибутивов Linux модули ядра загружаются автоматически при использовании ipvsadm.

Если модули не загружены, вы можете загрузить их вручную с помощью команды:

```
sudo modprobe ip_vs
sudo modprobe ip_vs_rr
sudo modprobe ip_vs_wrr
sudo modprobe ip_vs_lc
```

Создание скрипта или файла конфигурации:

Создайте скрипт или файл конфигурации, в котором будут определены правила IPVS. Этот файл будет содержать команды ipvsadm для настройки балансировки нагрузки. Например, вы можете создать файл /etc/ipvsadm.conf и добавить в него следующие строки:

```
# Пример правил IPVS для балансировки нагрузки методом Round Robin
# Виртуальный сервер
ipvsadm -A -t <виртуальный IP>:<порт> -s rr
# Добавление реальных серверов в пул
ipvsadm -a -t <виртуальный IP>:<порт> -r <IP реального сервера1>:<порт> -g
ipvsadm -a -t <виртуальный IP>:<порт> -r <IP реального сервера2>:<порт> -g
```

Замените <виртуальный IP>, <порт>, <IP реального сервера1>, <порт> и <IP реального сервера2> на соответствующие значения вашей конфигурации.

Применение конфигурации IPVS:

Загрузите настройки IPVS из файла конфигурации с помощью команды:

```
sudo ipvsadm -R -n < /etc/ipvsadm.conf
```

Проверка конфигурации IPVS:

Проверьте текущую конфигурацию IPVS с помощью команды:

```
sudo ipvsadm -L
```

Эта команда отобразит текущие правила IPVS и информацию о виртуальных и реальных серверах.

Если вы хотите, чтобы настройки IPVS автоматически загружались при запуске системы, настройте соответствующий механизм автозагрузки, такой как systemd, init.d или другой.

Домашнее задание:

1. Изучить дополнительные материалы.