Администрирование Linux
Лекция 11
Обзор Apache httpd 2.x

Иртегов Д.В.

Новосибирский гос. Университет

2014

Немного про протокол HTTP

- HyperText Transmission Protocol
- Первоначально разработан для передачи гипертекстовых документов HTML
- Используется для
  - Передачи статического и динамического контента веб-страниц,
  - Взаимодействия пользователей с веб-приложениями
  - Взаимодействия клиентских JavaScript программ с серверами
  - Взаимодействий сервер-сервер (гл. обр. XML/JSON RPC)
  - WebDAV, SVN over http, да тыщи их…

Логика работы протокола

- Логика, в основе своей, простая
  - Запрос-ответ
  - Данные и метаданные («заголовок»)
  - Основные типы запросов
    - GET uri ver – получить содержимое страницы
    - HEAD uri ver – получить метаданные страницы
    - POST uri ver – передать данные, например, заполненную форму
  - Основные типы ответов
    - 200 OK (дальше идет заголовок и тело страницы)
    - 2?? – ответы с дополнительным сообщением, например
      - 204 No content
      - 206 Partial content
    - 3?? – «предупреждение»: запрос валидный, но что-то не так, например
      - 301 – Moved permanently (HTTP redirect)
    - 4?? – ошибочный запрос,
      - 404 – страница не найдена
    - 5?? – ошибка сервера
      - 500 – internal server error

Версии протокола

- HTTP 0.9 – до сих пор где-то ездят,
  - например, lynx(1)/links(1) рапортуют 0.9
    Конкретно links можно мозги вправить
- HTTP 1.0 – считается устаревшей, но поддержка обязательна
  - На каждую пару запрос-ответ надо устанавливать соединение TCP
- HTTP 1.1
  - Connection pooling – много запросов в одном соединении
  - Набор обязательных полей в заголовке
  - Докачка (Range:)
- SPDY (HTTP 2.0 draft)
  - Request queueing
    можно слать новые запросы, не дожидаясь окончания ответа
  - Фреймы: к какому запросу какая часть ответа относится
  - Теперь это, фактически, бинарный протокол
    из телнета валидный диалог не проведешь
  - Header compression

Ключевые понятия – URL и URI

- Uniform Resource Locator

  `<схема>://<логин>:<пароль>@<хост>:<порт>/<путь>?<парам>#<якорь>`
- Uniform Resource Identifier

  `/<путь>?<парам>#<якорь>`
  - Идентификатор ресурса в пределах сервера
  - Выглядит как путь в файловой системе: каталоги, разделенные прямыми слэшами
  - НЕ ОБЯЗАТЕЛЬНО является путем в файловой системе
  - Может содержать параметры (обычно у динамических ресурсов) и якорь

Ключевые понятия - заголовок

- HTTP header (метаданные)
- Основные поля запроса
  - host – должен совпадать с <хост> из URL, обязательно в 1.1
  - UserAgent – тип браузера
  - Referer – откуда получена ссылка
  - Желательная кодировка и язык документов
  - Authorization
  - Cookie
- Основные поля ответа
  - MIME тип данных
    - text/html, image/jpeg, application/msword
  - MIME кодировка
    - KOI8, UTF-8, application/gzip
  - Дата модификации,
  - TTL, pragma: no cache – управление кэшированием
  - Длина (не обязательно)
  - Cookie

Ключевые понятия - прокси

- Proxy – по русски, посредник
- Прямой HTTP proxy
  - вместо URI получает полный URL
  - Перенаправляет его целевому хосту
- Используются для
  - Кэширования
  - Снижения нагрузки на файрволлы и NAT
  - Фильтрации трафика

Обратный прокси

- Стоит перед сервером или серверами
  - Клиент обычно не знает о его существовании
  - Получает URI и заголовок host
  - Перенаправляет его на определенный сервер или сервера
- Используются для
  - Кэширования
  - Балансировки загрузки
  - Доступа к серверам интранета
  - Интеграции веб-приложений, размещенных на разных серверах, в единый сайт

Популярные веб-серверы

- Apache httpd
  - Why Linux is like a wigwam?
  - All patches and feathers, apache inside
  - LAMP stack (Linux, Apache, MySQL, PHP)
- NGINX
- MS IIS
- Да тыщи их, на самом деле

Что умеет apache

- Раздавать статические ресурсы
  - Uri = имя файла относительно DocumentRoot
- Раздавать динамические ресурсы
  - CGI – Common Gateway Interface
    - Запуск на сервере практически произвольной программы с параметрами
  - Модули – расширения apache в виде специальных .so файлов
    - Например, интерпретаторы php, perl
  - FastCGI – протокол для общения со специальным сервером для запуска скриптов
    - fastcgiwrapper – довольно тупая запускалка
    - PHP FHM – довольно умная запускалка для скриптов PHP
- Делать всякие фокусы
  - Виртуальные директории
  - Виртуальные хосты
  - Mod_rewrite
  - Proxy redirect

CGI, FastCGI, модули

- Как выбрать способ запуска в конкретном случае?
  - CGI
    - Исполняются в отдельном процессе на том же сервере
    - + Можно исполнять что угодно
      например, скриптовые языки, для которых нет интерпретатора-модуля
    - + Если процесс помрет, серверу ничего не будет
    - + Можно использовать suexec
      запуск от имени владельца файла со скриптом
      используется в разделяемом хостинге, или если разрешают скрипты в домашних каталогах
    - - Запуск и уничтожение процесса – дорогая операция
    - - Инициализация среды исполнения происходит при каждом запуске
    - - Вместо падения, процесс может зациклиться или сожрать всю память
      - В юниксе есть квоты процессорного и астрономического времени, памяти и пр., но все это надо настраивать

CGI, FastCGI, модули

- Как выбрать способ запуска в конкретном случае?
  - Модули
    - Исполняются в контексте процесса Apache
    - + Инициализация среды исполнения происходит однократно
    - + Нет накладных расходов на запуск процесса
    - + Некоторые модули кэшируют компилированный код между запусками
    - - Код работает от имени пользователя apache
    - - Падение модуля убивает соответствующий процесс Apache
      умирает не весь Apache, а только один процесс, но
      - остальной апач это замечает не сразу,
      - перезапуск процесса – дорого,
      - поэтому частые падения снижают наблюдаемую производительность
    - - Утечки памяти в модуле жрут память процесса Apache
    - - Не все полезные программы доступны в виде модулей
    - - Не все модули совместимы с mpm_worker
    - - Некоторые модули имеют ограничения или другие отличия от автономной реализации

CGI, FastCGI, модули

- Как выбрать способ запуска в конкретном случае?
  - FastCGI
    - Исполняются в контексте отдельного постоянно запущенного процесса (или пула процессов), возможно на другом сервере
    - + Инициализация может происходить однократно
    - + Нет накладных расходов на запуск процесса
    - + Некоторые FastCGI серверы кэшируют компилированный код между запусками
    - + Некоторые серверы позволяют использовать аналог suexec
    - + Возможно распределение и даже динамическая балансировка загрузки
    - - Не все полезные программы доступны в виде умных FastCGI есть универсальный fcgiwrapper, но выгоды от него по сравнению с простыми CGI сомнительны

Конфигурация apache

- Состоит из
  - основного файла
    в RHeL - /etc/httpd/conf/httpd.conf
  - Include-файлов
    /etc/httpd/conf.d/*.conf
  - Файлов .htaccess в каталогах данных
- XML-like блочная структура
  - Сервер (верхний уровень)
  - VirtualHost
  - Directory[Match], Location[Match], Files[Match]
  - Условные директивы – IfDefine, IfModule

Демонстрация

- Файл /etc/httpd/conf/httpd.conf с настройками по умолчанию

Основные модули

- Core – собственно сервер
- Управление процессами и потоками
  - Prefork – apache 1.x совместимый
    много однопоточных процессов
  - Worker – многопоточные процессы
    Выгоден по производительности,
    но не все модули совместимы
    (из популярных – mod_perl)

Настройки core

- ServerRoot – файлы с настройками
- DocumentRoot – файлы данных
- ScriptAlias – исполняемые файлы
- Listen – TCP порты
- User, Group
- LoadModule
- LoadFile – подгрузить .so, который сам не является модулем, но необходим для работы модуля
- TypesConfig, AddType – MIME типы содержимого
- AddHandler .ext – обработчик по расширению
- SetHandler – обработчик для всех файлов в каталоге, независимо от расширения
- LogLevel, LogFormat, CustomLog

Упражнение

- Поднять httpd с настройками по умолчанию
  - yum install httpd
  - service httpd start
  - iptables -I INPUT 6 -p tcp --dport 80 –j \ ACCEPT
- Открыть корневой документ в браузере
- Найти DocumentRoot в настройках сервера
- Подложить туда .html файл и открыть его в браузере

Настройки MPM

- MaxClient – количество одновременно обрабатываемых соединений TCP
  - Каждое соединение требует один
    - процесс (prefork)
    - нить (worker)
- ListenBacklog – очередь непринятых запросов TCP (если не указано, используется системное значение)
- StartServers, ServerLimit
- ThreadsPerChild (worker)
- MinSpareServers, MaxSpareServers – количество незанятых серверов
- MaxRequestsPerChild – после обработки такого количества запросов, процесс сервера перезапускается (защита от утечек памяти, в т.ч. в модулях)

О настройках MPM

- MaxClients – это количество одновременно активных соединений TCP. Оно определяется
  - Загрузкой сервера (hits per second)
  - Временем обработки одного запроса, которое определяется
    - Объемом передаваемых данных
    - Временем работы скриптов на сервере
    - Протоколом (HTTP 1.0, 1.1, SPDY)
    - Средней скоростью связи до клиента
      - Max – локальная сеть, интранет
      - Min – клиенты на GPRS/3G или на другом континенте
      - Если проблема в скорости до клиента, имеет смысл поставить NGINX
- Много клиентов – это много процессов/нитей, а каждый процесс и каждая нить потребляют память
  - По умолчанию, Linux/x64 – 8Мб стека на одну нить
- Также нити потребляют процессор
  если загрузка процессора ~100%, добавлять процессы бессмысленно
- Увеличение количества нитей может упираться в какие-то другие лимиты, например, квоту на количество соединений с СУБД
- Мораль: универсальных рекомендаций нет, нужно мерить по ситуации

Поддержка SPDY

- В базовой поставке RHeL6 нету
- Нужно скачать модуль (.rpm) с сайта http://code.google.com/p/mod-spdy/
- Это не репозиторий yum, поэтому автоматического обновления не будет
- Работает только поверх https

Настройки директории

- Виртуальные каталоги
  - Alias URI /directory/on/filesystem
  - ScriptAlias URI /directory/filesystem
- Блоки
  - <Directory *shell-style wildcard*> - по файловой системе
  - <DirectoryMatch *regexp*>
  - <Location *wildcard*>, <LocationMatch *regexp*> - по URI
  - <File wildcard>, <FileMatch regexp>
- Директивы
  - AllowOverride – какие параметры можно менять в .htaccess
  - Order allow,deny
  - Allow/Deny all/IP[netmask]/domain name/env=value
  - DirectoryIndex (index.html)
  - Options FollowSymLinks, Indexes

Демонстрация

- Настройки директорий, виртуальные директории, локации и .htaccess в дефолтных настройках RHEL и на рабочих серверах

Redirect и ProxyPass

- По синтаксису похожи на Alias, но вместо пути указывается URL
- Redirect *URI URL* – http redirect
- ProxyPass *URI URL* – обратный прокси требует загруженного mod_proxy

Демонстрация

- Одинаковые директивы Redirect и ProxyPass в конфиге сервера, и разница в поведении браузера при открытии соответствующих URL

Виртуальные хосты

- <VirtualHost>: Port-based, IP-based
- <NameVirtualHost>: Name-based (HTTP 1.1)
- Виртуальный хост может иметь свои настройки
  - Свой DocumentRoot
  - Свои виртуальные каталоги
  - Почти все параметры можно поменять
  - Нельзя поменять список загруженных модулей
- Иногда вообще всю настройку сервера делают в виртуальном хосте (проще потом мигрировать с сервера на сервер)
- Настройка https делается только в виртуальном сервере
- Модуль vhost_alias обеспечивает генерацию виртуальных хостов по шаблонам

Демонстрация

- Описания виртуальных хостов на рабочих серверах

Обзор модулей

- Модули добавляют
  - Функциональность
    - Могут обрабатывать определенные файлы интерпретаторы языков, imagemap
    - Могут обрабатывать весь контент перекодировка, добавление полей в заголовок, кэширование
    - Могут сами генерировать контент DirectoryIndex, ServerConfig
  - Директивы конфигурации
    - Например, модули auth* добавляют директивы управления авторизацией

mod_rewrite

- распознает URL и поля запроса по шаблонам
- делает контекстные замены
- Может выполнять широкий набор действий по результатам распознавания
- Примеры
  - Прозрачная переделка статического сайта в динамический
    ```
    RewriteRule ^(.*).html$ $1.php
    ```
  - Умная прозрачная переделка
    ```
    RewriteCond $1.php -f
    RewriteCond $1.html !-f
    RewriteRule ^(.*).html$ $1.php
    ```
  - Защита от хотлинкинга картинок
    ```
    RewriteCond %{HTTP_REFERER} !^$
    RewriteCond %{HTTP_REFERER} !www.example.com [NC]
    RewriteRule \.(gif|jpg|png)$  /images/go-away.png [R,NC]
    ```

Синтаксис

- RewriteRule *URIregexp URIreplace [action/options]*
- *action:*
  - PT – passthrough, обработать результат замены как обычный URI
  - P,R – proxypass, redirect
  - F – forbidden (HTTP 403)
  - G – gone (HTTP 410)
  - CO – добавить к ответу cookie
- options
  - NC – nocase (шаблон не чувствительный к регистру)
  - B – escape backreferences
  - C – chain rule (следующие правила обрабатываются, только если текущее правило сработало. Аналог if)
  - N – next (после замены, правило выполняется еще раз. Аналог while. Как и настоящий while, может зациклиться)