

Кэшируем динамический контент

Виталий Банковский

Продолжая цикл статей, я расскажу, как можно создавать кэширующие сервера для динамического контента. (Первую статью цикла «Создаем распределенную сеть доставки контента» см. в №2 за 2008 г.)

Немного о кэшировании

Сейчас на рынке работают несколько компаний, предоставляющих услуги по кэшированию и доставке контента. Но все они, насколько мне известно, имеют один существенный недостаток – неумение достоверно кэшировать динамический контент. Часть из них вообще не может кэшировать динамический контент, а часть обновляют локальный кэш страниц только периодически, что непригодно для страниц социальных сетей. И, как я подозреваю, причина одна – для того чтобы построить акселератор, который бы хранил достоверную копию динамической страницы, необходима плотная интеграция с оригинальным сайтом.

Технология кэширования динамических страниц

Здесь я не изобрету ничего нового, лишь применю стандартные средства протокола HTTP для построения акселератора. В целом вся система состоит из сервера-акселератора и оригинального сайта и работает по следующей схеме:

- Пользователь обращается к акселератору, который является проксирующим сервером по отношению к сайту.
- Если акселератор не имеет локальной копии страницы, то он обращается на сайт за ней, отдает страницу пользователю и сохраняет в локальном кэше.
- Если акселератор уже имеет эту страницу, то он отправляет запрос на оригинальную страницу с заголовком If-Modified-Since xxx, где xxx – дата последней модификации страницы, сохраненной в кэше.
- Когда оригинальный сайт получает такой запрос, он сравнивает дату модификации своей страницы с этой датой.

- Если страница уже была модифицирована, то отдается полная версия страницы.
- Если дата последней модификации такая же, как в кэше, то сайт должен отдать пустую страницу со статусом HTTP 304.
- Когда кэш получает ответ со статусом HTTP 304, то он отдает страницу из локального кэша.

Компоненты системы

Для построения такой системы я использовал модуль mod_accel Игоря Сысоева [2]. На данный момент идет активная разработка такого же модуля для nginx, но на момент написания этой статьи он еще не был готов.

Установка и настройка

В своей работе я использую CentOS семейства Redhat, поэтому описание процедуры устоновки и настройки будет ориентировано на этот дистрибутив. Первым делом необходимо получить библиотеку EAPI, которая нужна для компиляции модуля mod_accel. Ее можно скачать с сайта [3]. Библиотеку устанавливать не нужно.

```
tar -xzf mod_ssl-2.8.4-1.3.20.tar.gz
cd mod_ssl-2.8.4-1.3.20
./configure --with-apache=../apache-1.3.20
make
```

Далее получаем версию Apache 1.3.20 с сайта [4] и распаковываем архив:

```
tar -xzf apache_1.3.20.tar.gz
```

Также нужна библиотека MM, которая доступна на сайте [5].

```
tar xzf mm-1.2.1.tar.gz
cd mm-1.2.1
./configure
make
make install
```

Получаем модуль mod_accel и запускаем процесс обновления исходных кодов Apache:

```
tar -xzf mod_accel-1.0.34.tar.gz
cd ./configure \
--with-apache=../apache_1.3.20 \
--with-eapi=../mode_ssl-2.8.4-1.3.20/pkg.eapi
make # накладывает патчи
```

Далее собираем Apache и устанавливаем:

```
cd apache_1.3.20
EAPI MM=../mm-1.2.1 ./configure --enable-rule=EAPI \
--activate-module=src/modules/accel/libaccel.a
make
make install
```

Настройка Apache

Затем настраиваем Apache для кэширования нашего сайта. Сосредоточимся на основных моментах конфигурационного файла сервера Apache:

```
# Путь, где будут храниться закэшированные файлы.
# Уровень иерархии каталогов - 1
AccelCacheRoot /home/dcache 1

# Включить кэширование
AccelNoCache off

# Начальный URL и адрес оригинального сайта
AccelPass / http://10.10.10.11:80/

# Отключить удаление файлов из кэша
AccelUnlinkNoCached off

# Передавать в переменной X-Host переменную среды Host,
# пришедшей на акселератор.
AccelSetXHost on

# Включить передачу IP-адреса посетителя через
# переменную X-Real-IP нашему сайту
AccelSetXRealIP on

# Включить передачу оригинального URL нашему сайту через
# переменную X-URL
AccelSetXURL on
```

Перед запуском сервера Apache необходимо создать каталог для хранения страниц кэша и поменять владельца и группу, чтобы процесс apache мог записывать кэш в этот каталог:

```
mkdir /home/dcache
chown -R apache.apache /home/dcache
```

Если процесс apache работает под другим именем пользователя и группы, то нужно соответственно поменять параметры команды chown.

Включение запуска сервера Apache в процедуру начальной загрузки сервера состоит в создании файла /etc/init.d/apache со следующим содержанием:

```
#!/bin/sh
# chkconfig: 2345 55 25
```

```
case "$1" in
start)
echo -n "Starting: apache"
/usr/local/apache/bin/apachectl start
echo "."
;;

stop)
echo -n "Stopping service: apache"
killall httpd
echo "."
;;

restart)
$0 stop
sleep 2
$0 start
;;

*)
echo "Usage: /etc/init.d/apache {start|stop|restart}" >&2
exit 1
;;
esac
exit 0
```

Запускаем наш акселератор:

```
/etc/init.d/apache start
```

Интеграция динамической части сайта с акселератором

Статические элементы сайта будут успешно кэшироваться и без специальных настроек. Для того чтобы динами-



VPS хостинг

Удобный. Надежный. Мощный.
Для веб-сайтов, баз данных и электронной почты.



➔ PHP5, MySQL5, MS SQL, .NET, FTP,
Mail-сервер, root-доступ.

24 ЧАСА
ТЕХПОДДЕРЖКА

ДОМЕН
ЗА НАШ
СЧЕТ!

(495) 799-00-18
<http://www.rusonyx.ru>

ческие страницы нашего сайта могли быть закешированы, необходимо ввести проверку даты последней модификации страниц в кэше с датами последней модификации страниц. Для этого я модифицировал наш сайт таким образом, что каждая страница имеет дату последней модификации, хранимой в базе данных. Когда на сайт передается запрос из акселератора на какую-то страницу, сайт сравнивает дату из заголовка if-modified-since с датой последнего обновления страницы, и в зависимости от результатов проверки скрипт на сайте возвращает или полную версию страницы с кодом HTTP, равным 200, или пустую страницу с кодом 304.

Также обязательно нужно обратить внимание на страницы, которые никогда не должны быть кэшированы. Например, страницы авторизации, обновления учетных записей пользователей и так далее.

К сожалению, такая плотная интеграция зависит от программного обеспечения на сайте, поэтому в качестве примера целесообразно привести упрощенный пример скрипта сайта, проинтегрированного с акселератором:

Листинг 1. Пример программы сайта

```
<?php

$cache=1;

# Проверяем, пришел ли к нам запрос из акселератора
# с заголовком If-modified-since. Если такой заголовок
# существует, это означает, что акселератор имеет
# локальную копию этой страницы

if ($_SERVER['HTTP_IF_MODIFIED_SINCE'])
{
    $txt = "cache";
    $len = strlen($txt);

    # print 304 code
    header("Status: 304", false, 304);
    header("Content-Type: text/html", false, 304);

    # Печатаем длину контента, иначе акселератор решит
    # что контент – динамический и не кэширует страницу
    header("Content-Length: $len", false, 304);

    # Печатаем дату последней модификации страницы
    header("Last-Modified: Tue, 16 Oct 2007 12:45:26 GMT", ,
        false, 304);
    header("Pragma: cache", false, 304);
    header("Cache-Control: cache", false, 304);
    header("Content-Length: $len", false, 304);

    # Дата, когда документ считается устаревшим, должна
    # быть будущей, иначе акселератор не сохранит страницу

    $dt=time();
    $dt+=1;
    header("Expires: " . "
        gmdate("D, d M Y H:i:s", $dt) . " GMT");
    echo $txt;
    exit;
}
else
{
    # В этой секции мы будем генерировать некешируемый
    # контент или контент, который еще не кэширован
    # в акселераторе

    $content = get_some_content();

    # Вычисляем длину содержимого, иначе акселератор
    # решит, что это – динамический контент

    $len = strlen($content);
    header("Status: 200");
    header("Content-Type: text/html");
    header("Content-Length: $len");
}
```

```
# Печатаем дату последней модификации страницы
header("Last-Modified: Tue, 16 Oct 2007 12:45:26 GMT");

# Также у нас есть глобальная переменная.
# При генерации страниц для некешируемых страниц
# нужно выставить эту переменную равной нулю

if ($cache>0)
{
    # Указываем акселератору, что нужно
    # кэшировать контент
    header("Pragma: cache");
    header("Cache-Control: cache");
}
else{
    # Указываем акселератору что не нужно
    # кэшировать контент
    header("Pragma: no-cache");
    header("Cache-Control: no-cache");
}

# Как и раньше, устанавливаем дату устаревания документа
$dt=time();
$dt+=1;
header("Expires: " . "
    gmdate("D, d M Y H:i:s", $dt) . " GMT");


# Печатаем контент
echo $content;
exit;}

?>
```

Схема работы приложения может быть построена следующим образом:

- В начале программы переменная \$cache устанавливается в 1.
- Проверяется наличие заголовка if-modified-since и сравнивается с датой последнего обновления запрошенной страницы. В зависимости от результата проверки переменная \$cached устанавливается в 0 или 1.
- Во всех процедурах динамической генерации контента проверяется значение переменной \$cached. Если переменная равна 1, то все эти процедуры завершаются, и программа переходит в финальную фазу – генерация контента как показано выше. Если переменная равна 0, то программа создает контент с нуля.
- Во всех процедурах, где создается некешируемый контент, переменная \$cache устанавливается равной 0.
- Вызывается процедура печати контента (как показано выше в примере).

Заключение

В этой статье я рассмотрел основные принципы построения акселератора для кэширования динамического контента. За рамками статьи остался способ, позволяющий дополнительно ускорить систему кэширования через хранение дат последней модификации страниц в одной из наиболее быстрых систем хранения данных – memcached. 

1. Банковский В. Создаем распределенную сеть доставки контента. //Системный администратор, №2, 2008 г. – С. 64-68. – <http://www.samag.ru/cgi-bin/go.pl?q=articles;n=02.2008;a=01>.
2. http://sysoev.ru/mod_accel.
3. <http://www.modssl.org>.
4. <http://apache.org>.
5. <http://www.engelschall.com/sw/mm>.
5. <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>.
6. <http://www.danga.com/memcached>.