

# Создаем распределенную сеть доставки контента

**Виталий Банковский**

**Прошли те времена, когда ваш сайт был единственным поставщиком каких-либо услуг и пользователи были согласны ждать несколько секунд, пока загрузится содержимое сайта. В условиях жесткой конкуренции преимущества в виде качественного контента и богатой функциональности сходят на нет, если конкурент даже с меньшей функциональностью доставляет контент быстрее, чем ваш сайт.**

**К**омпания, в которой я работаю, является одним из ведущих поставщиков услуг социальных сетей во многих странах мира. И было решено улучшить качество наших служб, а именно ускорить доставку контента потребителям. Для этого предполагалось построить локальные датацентры на каждом континенте.

Были рассмотрены два варианта: локальные датацентры (datacenter, DC) с полным набором серверов (сервера базы данных и приложений) и датацентры с серверами, которые кэшировали бы данные с центрального источника.

Первый вариант был сразу отменен из-за высокой стоимости установки и обслуживания. О принципах построения второго варианта я и расскажу в статье.

## Немного теории

Распределенная сеть доставки контента, Content Delivery Network (CDN), состоит из следующих компонентов:

- система определения месторасположения посетителя;
- механизм выбора близлежащего регионального датацентра;
- кэширующие серверы;
- центральные серверы;
- система обнаружения отказов региональных датацентров.

Немного поговорим о первых двух подсистемах. Для определения месторасположения посетителя и перенаправления его на близлежащий датацентр существует три метода:

- **Перенаправление (Redirect).** В этом случае посетитель заходит на центральный сервер и за-

тем перенаправляется на локальный датацентр средствами протокола HTTP, где и продолжает работать. К минусам такого решения стоит отнести необходимость дублирования всего функционала центрального датацентра в региональных, медленная скорость обслуживания при первом посещении, невозможность перенаправить клиента на другой датацентр в случае отказа регионального датацентра.

- **Выбор оптимального пути с использованием основ Border Gateway Protocol (BGP).** В этом случае в каждом региональном датацентре устанавливается маршрутизатор, анонсирующий сеть поставщика услуг, например 10.40.40.0/24. Когда провайдер клиента получает несколько вариантов маршру-

та к этой сети, он выбирает наиболее короткий. А наиболее короткий путь и означает путь к близлежащему датацентру. К плюсам такого решения можно отнести надежность механизма выбора датацентра и высокую скорость (порядка нескольких секунд) переключения на запасные датацентры. К минусам стоит отнести сложность и высокую стоимость, т.к. каждый датацентр должен иметь свой маршрутизатор с поддержкой BGP, также необходимость получения сети как минимум класса C от регионального координатора.

■ **GeoDNS.** При использовании этого метода сервер имен определяет месторасположение компьютера посетителя и выдает адреса ближайшего к клиенту датацентра.

В этой статье я расскажу принципы построения CDN с использованием GeoDNS.

## Компоненты и структурная схема

Как уже упоминалось ранее, наша система будет состоять из следующих компонентов:

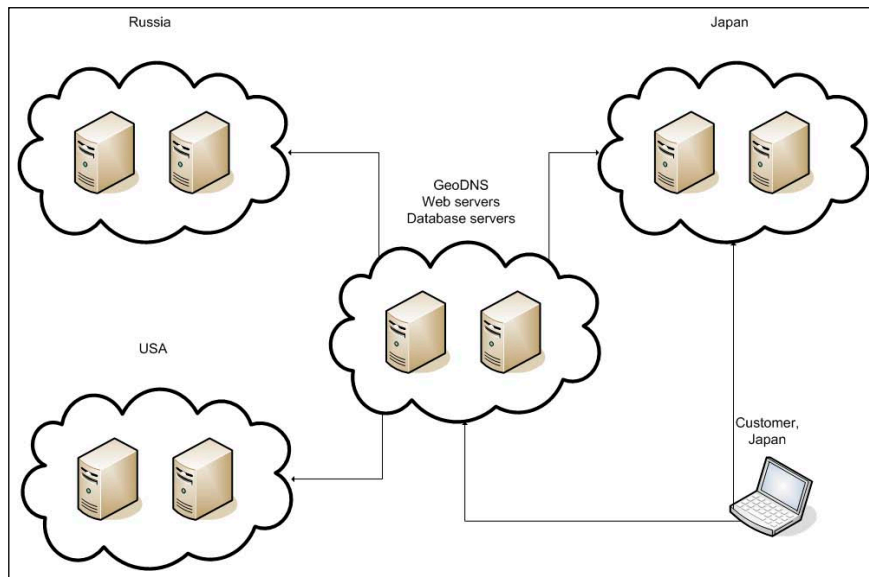
- центральный датацентр;
- датацентры с кэширующими серверами;
- GeoDNS;
- системы обнаружения отказов.

Структура с тремя кэширующими датацентрами и одним центральным показана на **рисунке**. При обращении к домену `example.com`, который обслуживается данной CDN, клиент обращается к локальному DNS, который, в свою очередь, обращается к GeoDNS, расположенному в центральном DC. GeoDNS определяет месторасположение клиента и выдает IP-адрес датацентра в Японии.

Итак, приступим к самой процедуре установки и настройки основных компонентов. В своей работе я использую CentOS 4.x семейства RedHat, поэтому все настройки и пути будут описаны для этого семейства дистрибутивов.

## Установка и настройка GeoDNS

Существует несколько программных и программно-аппаратных реше-



Структурная схема Content Delivery Network

ний GeoDNS. Я остановился на бесплатном BIND версии 9, для которого существует патч, который позволяет работать с базой данных регионов. Эта база данных содержит сети и их принадлежность к странам и регионам. Более подробно об этой базе можно почитать на сайте поставщика <http://www.maxmind.com>. Этот же поставщик предоставляет API для языков программирования C, Java, Pascal, Perl, PHP, Python, Ruby, VB.NET, C#. Этот API также можно использовать в других приложениях, например, для определения страны посетителя сайта, как это сделано в рассматриваемой дальнейшей службе.

## Установка и настройка базы данных регионов

Как уже говорилось, база данных стран и регионов содержит сети и их принадлежность к регионам или странам. Поставщик базы предоставляет несколько типов лицензий, самые интересные из которых приведены в **таблице**.

Этот поставщик также бесплатно предоставляет базу данных по странам, обновляемую один раз в месяц. Получаем необходимую базу с сайта

<http://www.maxmind.com> и устанавливаем в каталог `/var/geodns/`:

```
gzip -d GeoIP.dat.gz
mkdir /var/geodns
mv GeoIP.dat /var/geodns/
```

С этого сайта получаем библиотеку, предоставляющую API GeoDNS и устанавливаем ее:

```
tar -xzf GeoIP.tar.gz
./configure --with-dbdir=/var/geodns
make
make install
ldconfig
```

## Установка сервера имен

Для сервера имен я использовал BIND 9, для которого существует соответствующий патч, позволяющий серверу имен работать с базой данных регионов через вышеуказанный API.

Патч доступен по адресу <http://www.caraytech.com/geodns>. Получаем патч и исходники BIND 9 с сайта <http://www.isc.org/sw/bind> и устанавливаем оба компонента:

```
tar -xzf bind-9.4.1-geodns-patch.tar.gz
cp bind-9.4.1-geodns-patch/ patch.diff bind-9.4.2
```

Сравнение лицензий

Тип лицензии	Стоимость для каждого сервера, USD	Стоимость обновлений, в месяц
База данных по странам	50	12
База данных по регионам*	150	36
База данных по городам*	360	90
База данных типов подключения (DSL, кабель, T1)	370	90

\* Только США/Канада

```
cd bind-9.4.1
patch -p 1 < patch.diff
CFLAGS="-I/usr/local/geoip/include"
LDLFLAGS="-L/usr/local/geoip/lib -lGeoIP"
./configure --prefix=/usr/local/geobind
make
make install
```

## Настройка сервера имен

В BIND 9 существует возможность создавать «виды». В зависимости от параметров запроса (в нашем случае – адреса посетителя, который транслируется в страну с помощью базы данных регионов), BIND использует различные определения зон. После установки BIND наша конфигурация может выглядеть следующим образом:

```
; стандартные опции конфигурации
options {
    directory "/usr/local/geobind/etc";
    notify yes;
    pid-file "/var/run/geo-named.pid";
    statistics-file "/var/log/geo-named.stats";
};

; представление для региона USA и Canada
view "us" {
    match-clients { country_US; country_CA; };
    recursion no;
    zone "example.com" {
        type master;
        file "us.db";
    };
};

; представление для региона JP
view "jp" {
    match-clients { country_jp; };
    recursion no;
    zone "example.com" {
        type master;
        file "jp.db";
    };
};

; представление для региона RU
view "ru" {
    match-clients { country_ru; };
    recursion no;
    zone "example.com" {
        type master;
        file "ru.db";
    };
};

; представление для всех остальных регионов
view "central" {
    match-clients { any; };
    recursion no;
    zone "example.com" {
        type master;
        file "central.db";
    };
};
```

Мы определили четыре «представления»: US, RU, JP и универсальную зону, которая используется, если IP-адрес посетителя не входит ни в один из трех регионов. Для каждого представления мы создали файл зоны, в котором содержатся IP-адреса локального для каждой страны датацентра. Я приведу два примера определения зон US и RU.

Зона US (файл us.db):

```
@ IN SOA inet.example.com. root.example.com. (
    2007110206 ; Serial yyyymmddnn
    1800       ; Refresh 30 min
    1200       ; Retry 20 min
    2592000    ; Expire 30 days
    300 )      ; Default TTL

;
IN      NS      ns1
IN      NS      ns2
```

```
; серверы имен в центральном датацентре

ns1 IN      A      40.40.40.41
ns2 IN      A      40.40.40.42

; 66.66.66.66 – адрес кэширующего сервера в USA
@ IN      A      66.66.66.66
www IN     A      66.66.66.66

jp IN      A      22.22.22.22
us IN      A      66.66.66.66
ru IN      A      77.77.77.77
```

Зона RU (файл ru.db)

```
@ IN SOA inet.example.com. root.example.com. (
    2007110206 ; Serial yyyymmddnn
    1800       ; Refresh 30 min
    1200       ; Retry 20 min
    2592000    ; Expire 30 days
    300 )      ; Default TTL

;
IN      NS      ns1
IN      NS      ns2

; серверы имен в центральном датацентре

ns1 IN      A      40.40.40.41
ns2 IN      A      40.40.40.42

; 77.77.77.77 – адрес кэширующего сервера в Russia
@ IN      A      77.77.77.77
www IN     A      77.77.77.77

jp IN      A      22.22.22.22
us IN      A      66.66.66.66
ru IN      A      77.77.77.77
```

Хочу обратить внимание, что я выставил короткий TTL зоны (время, через которое серверы имен на стороне посетителя должны получать обновленные зоны с главного DNS) равным 5 минутам. Это используется для сокращения времени переключения с отказавших датацентров на запасные.

После запуска GeoBIND можно приступить к тестированию из разных стран, для чего воспользуемся программой dig, которая входит в состав большинства UNIX-систем.

Тестирование из России:

```
> dig example.com ns1.example.com A +nostats
```

```
;; ANSWER SECTION:
example.com.      44303 IN      A      77.77.77.77

;; AUTHORITY SECTION:
example.com.      34103 IN      NS      ns1.example.com
example.com.      34103 IN      NS      ns2.example.com
```

Тестирование из США:

```
> dig example.com ns1.example.com A +nostats
```

```
;; ANSWER SECTION:
example.com.      44303 IN      A      66.66.66.66

;; AUTHORITY SECTION:
example.com.      34103 IN      NS      ns1.example.com
example.com.      34103 IN      NS      ns2.example.com
```

Итак, мы построили DNS, который будет направлять посетителей в региональные датацентры.

## Построение кэширующих серверов

Теперь переходим к установке систем кэширования в региональных датацентрах. Эти системы, при обращении к ним

пользователей, будут обращаться к центральному DC, сохранять ответы в кэше и отдавать контент уже непосредственно из кэша. Существует несколько программных пакетов – Apache, Nginx, Squid, Lighttpd, которые умеют работать в таком режиме. При выборе пакета я сразу откинул Apache и Squid из-за их «прожорливости» и выбрал Lighttpd, с которым у меня уже был опыт работы, хотя Nginx предоставляет аналогичные возможности. К тому же lighttpd на момент внедрения этого решения был намного лучше документирован. Проект сервера находится по адресу <http://lighttpd.net>.

## Установка lighttpd

В первичной комплектации сервер lighttpd не содержит функций кэширования на диск (только в память). Поэтому кроме самого сервера необходимо скачать патч, включающий эту функциональность. Патч можно получить по адресу <http://trac.lighttpd.net/trac/wiki/Docs:ModCache>. После того как мы получили исходные коды самого сервера и патча, производим установку:

```
tar -xzf lighttpd-1.5.0-r1605.tar.gz
cd lighttpd-1.5.0
patch -p 0 < ../lighttpd-1.5.0.r1605.modcache.v.1.3.2.patch
./configure --with-pcre
make
make install
```

Перед установкой надо убедиться, что установлен пакет PCRE, версия для разработчика. В семействах RedHat и Debian он называется `pcre-devel`. Если нет возможности установить «родной» пакет, то его можно получить с сайта проекта по адресу <http://www.pcre.org>.

## Настройка lighttpd

В состав пакета lighttpd кроме исходных кодов входит типичный конфигурационный файл, которым мы и воспользуемся, скопировав его в каталог `/usr/local/etc/`:

```
cp doc/lighttpd.conf /usr/local/etc/
```

Далее идет описание наиболее важных для нашего использования параметров конфигурационного файла:

```
# Подключаем необходимые модули
server.modules = (
    "mod_cache",
    "mod_proxy_core",
    "mod_proxy_backend_http",
    "mod_proxy_backend_fastcgi",
    "mod_accesslog" )

# Каталог, в котором будут храниться закешированные данные
server.document-root = "/home/lighttpd/cache"

# Имя файла, в котором будут сохраняться ошибки
server.errorlog = "/www/logs/lighttpd.error.log"

# Имя файла, в котором будет храниться история обращений
accesslog.filename = "/var/log/lighttpd-access.log"

# Каталог, в котором должны сохраняться закешированные
# ответы с центрального DC. Значение этого параметра
# должно совпадать с значением переменной server.document-root
cache.bases=("/home/lighttpd/cache")

# Включаем кэширование
cache.enable = "enable"
```

```
# Задаем параметры кэширования. Включаем кэширование всего
# содержимого центрального сайта с временем обновления
# каждые 30 минут
cache.refresh-pattern = (
    "." => "20 ignore-reload override-expire"
)

#
# Далее прописаны параметры, описывающие наш центральный DC
#
# Адрес центрального DC
proxy-core.backends = ( "40.40.40.42:80" )

# Протокол, по которому нужно обращаться к центральному DC
proxy-core.protocol = "http"

# Включение режима «сотрудничества» кэширующей
# и проксирующей подсистем
proxy-core.worked-with-modcache = "enable"
```

Включение запуска сервера lighttpd в процедуру начальной загрузки сервера состоит в создании файла `/etc/init.d/lighttpd` со следующим содержимым:

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

case "$1" in
    start)
        echo -n "Starting: lighttpd"
        /usr/local/sbin/lighttpd -f /usr/local/etc/lighttpd.conf
        echo "."
        ;;
    stop)
        echo -n "Stopping service: lighttpd"
        killall lighttpd
        echo "."
        ;;
    restart)
        $0 stop
        sleep 2
        $0 start
        ;;
    *)
        echo "Usage: /etc/init.d/lighttpd {start|stop|restart}" >&2
        exit 1
        ;;
esac

exit 0
```

Для включения запуска этого скрипта в процедуру начальной загрузки сервера для дистрибутивов семейства RedHat нужно выполнить следующую команду:

```
chkconfig lighttpd --add
chkconfig lighttpd on
```

Для других дистрибутивов и операционных систем смотрите справочные руководства.

Запускаем сервер lighttpd:

```
/etc/init.d/lighttpd start
```

Если все нормально, то его можно увидеть в списке процессов. Если нет, то ошибки конфигурирования можно увидеть в файле `/var/log/lighttpd.error.log`.

## Тестирование системы кэширования

Для тестирования можно применить программу `wget`, которую нужно запустить на сервере в том же регионе, что и



датацентр, в котором мы только что установили кэширующую систему:

```
wget http://example.com
```

Если все нормально, то первое обращение будет зафиксировано в лог-файлах веб-сервера в центральном датацентре и лог-файле сервера `lighttpd` в региональном датацентре. Второе и последующие обращения будут зафиксированы только в лог-файле сервера `lighttpd`.

Таким же образом настраиваются все остальные региональные датацентры.

## Обнаружение отказов датацентров и их изоляция

Как уже говорилось ранее, наша схема с центральным датацентром и системой региональных кэширующих серверов позволяет переключать посетителей с отказавшего датацентра на центральный или запасной кэширующий датацентр. Я остановлюсь на примере, когда переключение происходит на центральную хостинговую площадку.

Процедура обнаружения и переключения работает по следующему алгоритму:

- Каждые 5 минут скрипт на языке Perl проверяет доступность каждого датацентра.
- Если обнаружен отказ в обслуживании, например, датацентра в Японии, то файл `jb.db` в конфигурации сервера имен замещается файлом `central.db`, увеличивается серийный номер записи и сервер имен перечитывает файлы зон.
- При восстановлении отказавшего датацентра происходит обратная замена файла с записями зоны.

Для тестирования я написал небольшой скрипт на языке Perl и воспользовался пакетом `nanos-plugins`, в который входит множество утилит по проверке доступности служб. Сайт пакета находится по адресу <http://nanosplugins.org>. Получаем последнюю версию и устанавливаем в каталог `/usr/local/nanos-plugins`:

```
tar -xzf nanos-plugins-1.4.11.tar.gz
cd nanos-plugins-1.4.11
./configure --prefix=/usr/local/nanos-plugins
make
make install
```

Пример скрипта:

```
#!/usr/bin/perl

# Путь к файлам зон сервера имен
my $bind_cfg='/usr/local/geobind/etc';

# Команда для перечитывания файлов зон сервером BIND9
my $bind_restart = '/usr/local/geobind/sbin/rndc reload';

# Описание наших кэширующих датацентров:
my $datacenters = [
    {id=>'us', name=>'us.example.com'},
    {id=>'ru', name=>'ru.example.com'},
    {id=>'jp', name=>'jp.example.com'},
];

# Примечание: имена типа jp.example.com были раньше
# прописаны в DNS и служат для обнаружения факта
# восстановления отказавших региональных датацентров
```


```
foreach my $dc (@$datacenters)
{
    # Вычисление контрольных сумм зон для дальнейшего
    # сравнения
    my $md1= (split(/\s+/, `md5sum $bind_cfg/$dc->{id}.db`))[0];
    my $md2= (split(/\s+/, `md5sum $bind_cfg/central.db`))[0];

    # Если контрольные суммы совпадают, то это означает,
    # что уже было переключение с отказавшего датацентра
    # на центральный
    if($md1 eq $md2)
    {
        # Проверяем состояние датацентра и, если был факт
        # восстановления, то переключаем пользователей
        # обратно на региональный датацентр
        if(!system("/usr/local/nanos-plugins/libexec/ `
            check_http -H $dc->{name}`"))
        {
            print "$dc->{id} has been recovered\n";
            system("cp $bind_cfg/$dc->{id}-original.db `
                $bind_cfg/$dc->{id}.db");
            restart_dns("$bind_cfg/$dc->{id}.db", 2);
        }
    }
    # Если переключения не было (разные контрольные суммы),
    # то тестируем доступность датацентра и переключаем
    # пользователей на центральный в случае проблем
    else
    {
        if(system("/usr/local/nanos-plugins/libexec/ `
            check_http -H $dc->{name}`"))
        {
            print "$dc->{id} has failed\n";
            system("cp $bind_cfg/$dc->{id}.db `
                $bind_cfg/$dc->{id}-original.db");
            system("cp $bind_cfg/central.db `
                $bind_cfg/$dc->{id}.db");
            restart_dns("$bind_cfg/$bind_cfg/`
                $dc->{id}.db", 1);
        }
    }
}

# Процедура увеличения серийного номера зоны
# и перезапуск сервера имен
sub restart_dns
{
    my ($file,$step)=@_;
    open(F,"<$file");
    my @lines=<F>;
    close(F);

    open(F, ">$file");
    foreach my $i(@lines)
    {
        if($i =~ /Serial/ && $i =~ /\d+/igs)
        {
            my $serial=$1;
            $serial+=$step;
            $i =~ s/$1/$serial/igs;
        }
        print F $i;
    }
    close(F);
    system($bind_restart);
}
```

## Заключение

В этой статье вы познакомились с основными принципами построения сети Content Delivery Network, которая будет эффективно работать в случае доставки статического контента. Систему кэширования динамического контента и систему обновления кэша в случае изменения данных я опишу в одной из моих следующих статей. 

1. <http://www.caraytech.com/geodns>.
2. <http://www.isc.org/sw/bind>.
3. <http://www.maxmind.com>.