Создание тайлового сервера на основе данных OpenStreetMap и mod tile

Обсудить в форуме Комментариев — 9

Эта страница опубликована в основном списке статей сайта
по адресу http://gis-lab.info/qa/mod_tile.html

Рассматривается процесс создания собственного тайлового сервера на основе данных OpenStreetMap на примере операционной системы CentOS 6.

Содержание

Введение

Сегодня во многих картографических веб-приложениях в качестве одной из подложек используется слой на базе тайлов OpenStreetMap. Популярность данной подложки обусловлена во-первых политикой предоставления тайлов, позволяющей свободно использовать тайлы OpenStreetMap в своих приложениях, а во-вторых - простотой их подключения в современных веб-клиентах, таких как Leaflet и OpenLayers.

Однако иногда возникает необходимость, чтобы данная подложка функционировала в том числе и при отсутствии Интернета (Интранет-приложения) или, будучи основанной на тех же исходных данных, имела совсем другое представление. Первым решением данной задачи, которое приходит в голову - это скачать необходимые тайлы с сайта OpenStreetMap. Однако это не очень удачное решение, так как политика использования тайлов запрещает их массовое скачивание и, кроме того, данный вариант не позволит настроить собственную символику подложки, отличную от стандартной. Правильным решением данной задачи является получение исходных данных OpenStreetMap и построение на их основе собственного набора тайлов. Решению данной задачи и посвящена данная статья.

Набор тайлов может быть подготовлен различными способами, но наиболее универсальный вариант предоставления доступа к тайлам - это использование тайлового сервера - программного обеспечения, работающего по следующим принципами:

- при запросе тайла - сервер проверяет его наличие в своём кэше, если тайл найден - он возвращается клиенту;
- если тайл не найден, то осуществляется его рендеринг, после чего он сохраняется в кэше и только

после этого возвращается клиенту.

Стек программных продуктов тайлового сервера openstreetmap.org

Существуют различные инструменты создания тайловых серверов со своими плюсами и минусами, мы же рассмотрим данный процесс в разрезе стека технологий, применяемых для создания тайлов на сервере openstreetmap.org. Данный стек состоит из 5 компонентов: mod_tile, renderd, Mapnik, osm2pgsql и PostgreSQL/PostGIS.

- **mod_tile** - это модуль веб-сервера Apache, который отдаёт кэшированные тайлы и определяет нуждаются ли те или иные тайлы в отрисовке (в зависимости от того есть ли они в кэше и не истек ли срок их актуальности).
- **renderd** представляет собой систему управления очередью запросов на рендеринг, предназначенную для оптимизации нагрузки такими запросами.
- **Mapnik** - рендерер, превращающий векторные данные в растровые тайлы и используемый renderd.
- **osm2pgsql** - инструмент загрузки исходных данных OpenStreetMap в базу данных.
- **PostgreSQL/PostGIS** - собственно СУБД.

Данный стек технологий работает только в UNIX-подобных операционных системах и не работает в Windows, так как использует для связи mod_tile и renderd доменные сокеты Unix.

Если в качестве операционной системы вы планируете использовать Debian или Ubuntu, то вам лучше обратиться к инструкции по развертыванию тайлового сервера для данных операционных систем из пакетов. На том же ресурсе имеется инструкция как развернуть тайловый сервер на Ubuntu (Debian) из исходных кодов. В случае же если ваша операционная система отличается от вышеназванных, то полностью следовать описанным в них шагам не получится, так как некоторые из них довольно Debian-специфичны. Вот тут, возможно, и пригодится данная статья.

Установка программного обеспечения

В качестве базового дистрибутива мы будем рассматривать операционную систему CentOS 6 x86_64, но вся последовательность шагов может быть легко адаптирована и под любой другой Linux-дистрибутив. Большинство из необходимого ПО (mod_tile, renderd, Mapnik, osm2pgsql) для данной операционной системы в репозиториях по-умолчанию отсутствуют (либо устарели), а ставить из исходных кодов не очень удобно.

Поэтому для удобной установки необходимых пакетов лучше использовать специальный репозиторий. Как его подключить и работать с ним будет рассмотрено далее. Руководство по самостоятельной сборке RPM-пакетов можно почитать здесь.

Подключение необходимых репозиториев

Подключаем репозиторий EPEL 6:

```
yum localinstall http://fedora-mirror01.rbc.ru/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
```

Подключаем репозиторий с PostgreSQL:

```
yum localinstall http://yum.postgresql.org/9.3/redhat/rhel-6-x86_64/pgdg-centos93-9.3-1.noarch.rpm
```

В файле конфигурации стандартного репозитория, расположенного по адресу */etc/yum.repos.d/CentOS-Base.repo* в секции *base* и *updates* добавьте строку:

```
exclude=postgresql*
```

Подключаем репозиторий enetres, содержащий свежую версию библиотеку Boost C++ 1.55, используемую при сборке Mapnik. Для подключения данного репозитория скопируйте файл в директорию */etc/yum.repos.d/*.

Подключаем репозиторий nextgis, содержащий все остальные необходимые компоненты: скопируйте файл в директорию */etc/yum.repos.d/*.

Установка PostgreSQL/PostGIS

Устанавливаем PostgreSQL:

```
yum install postgresql93-server
service postgresql-9.3 initdb
service postgresql-9.3 start
chkconfig postgresql-9.3 on
```

Устанавливаем PostGIS:

```
yum install postgis2_93
```

Установка mod_tile, renderd, Mapnik, osm2pgsql

Устанавливаем основные инструменты:

```
yum install apache2-mod_tile renderd mapnik mapnik-python osm2pgsql
```

Создание базы данных и загрузка данных

Создаём суперпользователя от имени которого будет вестись работа с базой данных(назовём его *dr*, а базу данных *gis*):

```
su - postgres -c "createuser dr -s -P -e"
su - postgres -c "createdb -E UTF8 -O dr gis"
```

Загружаем функции PostGIS:

```
su - postgres -c "psql -d gis -c 'CREATE EXTENSION postgis;'"
su - postgres -c "psql -d gis -c 'ALTER TABLE geometry_columns OWNER TO dr;'"
su - postgres -c "psql -d gis -c 'ALTER TABLE spatial_ref_sys OWNER TO dr;'"
```

Загружаем данные в базу. Будем загружать данные на территорию СНГ. Существует несколько способов получения исходных данных OpenStreetMap, один из них - ежедневные выгрузки данных на территорию бывшего СССР в форматах XML и PBF, расположенные здесь.

```
mkdir ~/src
cd ~/src
wget http://data.gis-lab.info/osm_dump/dump/latest/local.osm.pbf
osm2pgsql -U dr -W --slim -C 1500 --number-processes 4 -d gis --drop local.osm.pbf
```

При использовании osm2pgsql мы указали объём выделяемой оперативной памяти 1.5 Гб (*-C* 1500), включили slim-режим (*--slim*, подробности о режимах загрузки данных можно найти в документации) и активировали загрузку в 4 процесса (*--number-processes* 4). Кроме того, так как мы не планируем diff-обновления наших данных, то используя ключ *--drop*, мы автоматически удаляем создаваемые при загрузке данных slim-таблицы, что значительно уменьшает размер нашей базы данных. Загрузка данных будет продолжаться несколько часов, сколько конкретно - зависит от аппаратной част и настроек программного обеспечения, более подробную информацию можно получить в данной презентации.

Настройка стилей Mapnik

В качестве стилей тайлов будем использовать стандартный стиль OpenStreetMap. Скачиваем файлы с описанием стиля:

```
cd ~/src
svn co http://svn.openstreetmap.org/applications/rendering/mapnik mapnik-style
```

Стандартный файл стилей Mapnik подразумевает, что для создания тайлов, содержащих береговые линии и территории, занимаемые океанами, на мелких масштабах используются отдельные файлы, а не данные из базы, поскольку так гораздо быстрее. Загружаем эти файлы (порядка 400 Мб):

```
cd ~/src/mapnik-style
./get-coastlines.sh /usr/local/share
```

При выполнении данной команды вы получите сообщение об ошибке следующего вида:

```
bunzip2 is not installed in /bin/bunzip2, it is needed by this script
```

Чтобы узнать, куда у вас в системе установлен *bunzip2*, выполните команду:

```
which bunzip2
```

После чего откройте файл *get-coastlines.sh* и отредактируйте следующую строку:

```
BUNZIP2=/bin/bunzip2
```

В нашем случае она выглядит так:

```
BUNZIP2=/usr/bin/bunzip2
```

После этого снова запускайте этот файл. Должен начаться процесс загрузки данных:

```
./get-coastlines.sh /usr/local/share
```

После того как все необходимые файлы были получены, необходимо осуществить небольшую правку конфигурационных файлов для адаптации их под свою систему:

```
cd ~/src/mapnik-style/inc
cp fontset-settings.xml.inc.template fontset-settings.xml.inc
cp datasource-settings.xml.inc.template datasource-settings.xml.inc
cp settings.xml.inc.template settings.xml.inc
```

Открываем файл *settings.xml.inc* и приводим его к следующему виду:

```
<!--
Settings for symbols, the spatial reference of your postgis tables, coastline
shapefiles directory, and their prefix names.
-->

<!-- use 'symbols' unless you have moved the symbols directory -->
<!ENTITY symbols "symbols">

<!-- use the '&srs900913;' entity if you have called osm2pgsql without special flags
(or with -m); use '&srs4326;' if you have used -l -->
<!ENTITY osm2pgsql_projection "&srs900913;">

<!-- used for 'node in way' ST_DWithin spatial operations -->
<!-- Use 0.1 (meters) when your database is in 900913     -->
<!-- Use 0.000001 (degrees) when your database is in 4326 -->
<!ENTITY dwithin_900913 "0.1">
<!ENTITY dwithin_4326 "0.00001">
<!ENTITY dwithin_node_way "&dwithin_900913;">

<!-- use 'world_boundaries', which is the usual naming for the local folder the
coastline shapefiles are unzipped into -->
```

```
<!ENTITY world_boundaries "/usr/local/share/world_boundaries">

<!-- use 'planet_osm' unless you have customized your database table prefix using the
osm2pgsql 'prefix' flag -->
<!ENTITY prefix "planet_osm">
```

Открываем файл *datasource-settings.xml.inc* и приводим его к следующему виду (указав собственный пароль и охват, соответствующий загруженным в базу данным):

```
<!--
Settings for your postgres setup.

Note: feel free to leave password, host, port, or use blank
-->

<Parameter name="type">postgis</Parameter>
<Parameter name="password">Ijdg83wk</Parameter>
<!-- <Parameter name="host">%(host)s</Parameter> -->
<!-- <Parameter name="port">%(port)s</Parameter> -->
<Parameter name="user">dr</Parameter>
<Parameter name="dbname">gis</Parameter>
<!-- this should be 'false' if you are manually providing the 'extent' -->
<Parameter name="estimate_extent">false</Parameter>
<!-- manually provided extent in epsg 900913 for whole globe -->
<!-- providing this speeds up Mapnik database queries -->
<Parameter name="extent">-20037508.33,4183149.83,20037508.34,17014106.6</Parameter>
```

Замечание: начиная с августа 2013 года, тайлы на сайте openstreetmap.org отрисовываются не с помощью XML-стиля, а с помощью стиля в формате CartoCSS. Это полностью переписанный на CartoCSS старый XML-стиль, что значительно упрощает его модификацию. Стиль сделан как можно более похожим на старый.

Настройка renderd

Откройте файл, содержащий настройки renderd. Данный файл расположен по адресу */etc/renderd.conf*. Приведите его к следующему виду (так как в нашем примере файл стилей Mapnik находится в домашнем каталоге пользователя, то отредактируйте значение параметра *XML* в соответствии со своим пользователем (в нашем примере имя пользователя в системе и имя пользователя в базе данных совпадают, у вас это могут быть совершенно разные пользователи)):

```
[renderd]
socketname=/var/run/renderd/renderd.sock
num_threads=4
tile_dir=/var/lib/mod_tile
stats_file=/var/run/renderd/renderd.stats

[mapnik]
plugins_dir=/usr/lib64/mapnik/input
font_dir=/usr/share/fonts/dejavu
font_dir_recurse=1

[default]
URI=/osm_tiles/
TILEDIR=/var/lib/mod_tile
XML=/home/dr/src/mapnik-style/osm.xml
HOST=localhost
CORS=*
TILESIZE=256
```

Список всех настроек, которые могут быть добавлены в этот файл можно найти здесь.

Создадим директорию в которую будет помещён Unix-сокет для взаимодействия mod_tile и renderd и директорию, в которой будут храниться кэшированные тайлы (соответствующие пути мы прописали в файле

renderd.conf):

```
mkdir /var/run/renderd
mkdir /var/lib/mod_tile
```

Сделаем владельцами данных каталогов пользователя, от имени которого будет запускаться renderd. Пусть в нашем случае это будет системный пользователь dr:

```
chown dr /var/run/renderd
chown dr /var/lib/mod_tile
```

Настройка mod_tile

Если Apache ещё не установлен, то установите его:

```
yum install httpd
```

Перейдите в директорию с настройками Apache */etc/httpd/conf.d* и создайте там файл *mod_tile.conf* следующего содержания:

```
LoadModule tile_module /usr/lib64/httpd/modules/mod_tile.so

<VirtualHost *:80>
    ServerName localhost
    ServerAdmin webmaster@localhost

    LoadTileConfigFile /etc/renderd.conf
    ModTileRenderdSocketName /var/run/renderd/renderd.sock
    # Timeout before giving up for a tile to be rendered
    ModTileRequestTimeout 0
    # Timeout before giving up for a tile to be rendered that is otherwise missing
    ModTileMissingRequestTimeout 30

    ErrorLog /var/log/httpd/error_log

    # Possible values include: debug, info, notice, warn, error, crit,
    # alert, emerg.
    LogLevel warn

    CustomLog /var/log/httpd/access_log combined

</VirtualHost>
```

Описание всех инструкций, которые могут быть добавлены в этот файл можно найти здесь.

Проверка корректности произведённых настроек

Запускаем renderd вручную:

```
renderd -f -c /etc/renderd.conf
```

Если при запуске вы столкнётесь с подобной проблемой, то поставьте в систему пакет *proj-epsg*:

```
yum install proj-epsg
```

Включаем возможность автоматической загрузки Apache после перезагрузки системы:

```
chkconfig httpd on
```

При запущенном renderd в отдельной консоли перезапускаем Apache:

```
service httpd restart
```

Должно появиться сообщение следующего содержания:

```
Starting httpd: Wed Jul 09 19:45:29 2014 notice Loading tile config default at
/osm_tiles/ for zooms 0 - 20 from tile directory /var/lib/mod_tile with extension .png
and mime type image/png
```

Пробуем открыть какой-нибудь тайл, например:

```
http://localhost/osm_tiles/0/0/0.png
```

Если вы получили тайл с первого раза - значит вам повезло. Если нет, то смотрите интерактивные логи запущенного renderd и логи Apache:

```
tail /var/log/httpd/error_log
```

Наиболее распространённая ошибка в логах Apache:

```
socket connect failed for: /var/run/renderd/renderd.sock with reason: Permission denied
```

Проверьте права доступа к сокету для пользователя, от имени которого запущен Apache:

```
sudo -u apache ls -lh /var/run/renderd/renderd.sock
```

В документации по mod_tile сказано, что SELinux блокирует соединение mod_tile и renderd, поэтому должен быть отключен (требуется перезагрузка). Также в случае возникновения подобных проблем попробуйте отключить сервис iptables, если он запущен.

Как только вам удастся получить готовый тайл - останавливайте renderd (*Ctrl+C*), запущенный в foreground-режиме.

Автоматический запуск renderd

Очевидно, что каждый раз вручную запускать renderd после перезагрузки сервера - это не дело, поэтому нам необходима возможность автоматического запуска renderd. В репозитории mod_tile (renderd входит в его состав) присутствует init-скрипт, выполняющий эту задачу. Однако он Debian-специфичен и нам не подходит. Напишем свой.

Создаём файл */etc/init.d/renderd* следующего содержания:

```
#!/bin/bash
#
# renderd    Mapnik rendering daemon
#
# chkconfig: 345 70 30
# description: Mapnik rendering daemon
# processname: renderd

# Source function library.
. /etc/init.d/functions

PATH=/sbin:/usr/sbin:/bin:/usr/bin
DESC="Mapnik rendering daemon"
NAME=renderd
DAEMON=/usr/bin/$NAME
DAEMON_ARGS="-c /etc/renderd.conf"
PIDSOCKDIR=/var/run/$NAME
PIDFILE=$PIDSOCKDIR/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME
```

```
RUNASUSER=dr

start() {
        if [ -e $PIDFILE ];
        then
            echo -n "$NAME already started"
            RETVAL=1
        else
            echo -n "Starting $NAME: "
            [ -d "$PIDSOCKDIR" ] ||  mkdir -p $PIDSOCKDIR && chown $RUNASUSER
$PIDSOCKDIR
            daemon --user $RUNASUSER $DAEMON $DAEMON_ARGS 2> /dev/null
            RETVAL=$?
            [ $RETVAL -eq 0 ] && touch $PIDFILE
        fi
        echo
        return $RETVAL
}

stop() {
        echo -n "Shutting down $NAME: "
        killproc $DAEMON && success || failure
        RETVAL=$?
        [ $RETVAL -eq 0 ] && rm -f $PIDFILE
        echo
        return $RETVAL
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        ;;
    *)
        echo "Usage: $SCRIPTNAME {start|stop|restart}"
        exit 1
        ;;
esac
exit $RETVAL
```

Делаем его исполняемым:

```
chmod u+x /etc/init.d/renderd
```

Запускаем сервис и добавляем его в автозагрузку:

```
service renderd start
chkconfig renderd on
```

Предварительный рендеринг тайлов

В текущей конфигурации тайлы будут отрисовываться по запросу. То есть если mod_tile не найдёт запрашиваемый тайл в кэше, то он обратится за ним к renderd. Однако процесс отрисовки тайлов требует определенного времени и чтобы не заставлять ждать клиента, запрашивающего тайл, пока этот тайл отрисуется, можно подготовить необходимые тайлы заранее. В состав пакета renderd помимо самого демона входит еще несколько вспомогательных утилит, среди которых *render_list*. С помощью данной утилиты можно

указать масштабный уровень и диапазоны x и y координат тайлов и осуществить предварительный рендеринг. Пример рендеринга тайлов 7 уровня на территорию России:

```
render_list --socket=/var/run/renderd/renderd.sock --all --max-zoom=7 --min-zoom=7 --min-x=73 --max-x=127 --min-y=18 --max-y=47 --num-threads=4
```

Пример Python-скрипта расчета координат тайлов по охвату интересующей области:

```
# Russia bbox
XMIN = 2989951.00
XMAX = 20037508.34
YMIN = 5039930.86
YMAX = 14222184.29

# Sphere radius
R = 20037508.342789244

for z in range(19):
  r = 2*R/(256*2**z)

  tile_w = tile_h = 256*r

  x_min = (R+XMIN)/tile_w
  x_max = (R+XMAX)/tile_w
  y_min = (R-YMAX)/tile_h
  y_max = (R-YMIN)/tile_h

  print "render_list --socket=/var/run/renderd/renderd.sock --all --max-zoom=%d --min-zoom=%d --min-x=%d --max-x=%d --min-y=%d --max-y=%d --num-threads=4" % (z, z, int(x_min), int(x_max), int(y_min), int(y_max))
```

Для получения списка тайлов в прямоугольнике с координатами WGS-84, либо по произвольному полигону обрезки, воспользуйтесь polytiles.py. Пример команды:

```
./polytiles.py --poly russia.poly --zooms 6 10 --export russia_tiles.lst -q
render_list -s /run/renderd/renderd.sock < russia_tiles.lst
```

Актуальность тайлов

Чтобы поддерживать тайлы в актуальном состоянии, mod_tile должен знать дату на которую были загружены исходные данные в базу PostGIS. В случае, если дата загрузки данных в базу свежее даты создания тайла, то при следующем обращении к этому тайлу он будет отрисован заново. Дата загрузки данных в базу хранится как timestamp файла *planet-import-complete*, который должен находиться в директории с тайлами:

```
touch /var/lib/mod_tile/planet-import-complete
```

Ежеминутное обновление

Чтобы обновлять данные в базе, не обязательно каждый раз обрабатывать всю выгрузку (а то и планету) целиком: достаточно настроить ежеминутное обновление данных базы. На сервер OSM выкладываются файлы изменений за эти периоды, и с помощью osmosis и osm2pgsql их можно накатить на рабочую базу. Если вы планируете подобного рода обновления, то данные должны быть загружены в базу с помощью osm2pgsql без использования ключа *--drop*.

Если вы собирали mod_tile из исходников, найдите там файл openstreetmap-tiles-update-expire, иначе скачайте его с github. Поправьте пути в начале файла. Запускать скрипт нужно от имени пользователя, который запускает renderd и прочее (по инструкции выше — dr). Сначала нужно инициализировать рабочие каталоги: запустите

```
./openstreetmap-tiles-update-expire YYYY-MM-DD
```

где дата — день, когда был подготовлен загруженный в базу данных дамп. Если будет на день-два раньше — ничего страшного, дублирования не произойдёт. Дальше всё просто: запустите crontab -e от пользователя dr и добавьте строку:

```
* * * * * /путь/к/openstreetmap-tiles-update-expire
```

Проверить, что обновление идёт без ошибок, можно по логу в /var/log/tiles/run.log. При полном обновлении базы данных не забывайте отключать вызов скрипта в crontab. Пока он остановлен, можно скачать правильный state.NNN.txt в /var/lib/mod_tile/.osmosis/state.txt, чтобы начать обновление с более ранней даты. Максимальный интервал, на который обновляется база за один вызов скрипта, указывается в файле /var/lib/mod_tile/.osmosis/configuration.txt в ключе maxInterval, по умолчанию — один час.

Полезные ссылки

- mod_tile
- osm2pgsql
- Mapnik
- switch2osm
- Подготовка карт для генератора тайлов Mapnik
- Настройка базы данных для генератора тайлов Mapnik
- Установка генератора тайлов Mapnik
- Установка renderd и mod_tile - системы отрисовки тайлов по запросу
- Тайловый сервер - что с ним делать?

Обсудить в форуме Комментариев — 9

Последнее обновление: 2014-07-18 19:46

Дата создания: 09.07.2014
Автор(ы): Denis Rykov