mod_asn Documentation

Release 1.7

Peter Pöoeml

Feb 27, 2022

CONTENTS

Release 1.7

Date Feb 27, 2022

Contents:

CHAPTER

ONE

INTRODUCTION

This is the documentation for mod_asn. mod_asn is an Apache module doing lookups of the *autonomous system (AS)* number, and the *network prefix*, that an IP address is contained in.

It is written with scalability in mind. To do high-speed lookups, it uses the PostgreSQL *ip4r data type* that is indexable with a *Patricia Trie* algorithm to store network prefixes.

It comes with a script to create such a database, and update it with snapshots from a router's "view of the world".

The module sets the looked up data as *Apache env table* variables, for use by other Apache module to do things with it, or for logging – and it can add the data as response headers to the client.

Example HTTP response headers:

```
HTTP/1.1 200 OK
Date: Thu, 12 Feb 2009 23:24:33 GMT
Server: Apache/2.2.11 (Linux/SUSE)
X-Prefix: 83.133.0.0/16
X-AS: 13237
```

1.1 Performance

The database with all ~250.000 prefixes is about 20-30MB in size in the form of a PostgreSQL database. Without any tuning, it is able to do >3000 lookups per second on a MacBook Pro (tested with random IPs, a single connection, and client written in Python running on the same machine).

The Apache module is extremely lightweight.

1.2 Design notes

Performed with a *Patricia Trie* algorithm, the lookup is very efficient. The Patricia Trie is a special radix tree that works it way from bit to bit, starting at the most significant bit. At each bit, there are two alternative "paths". Or put another way, the space of prefixes is roughly divided in two halfs at each point. The ip4r datatype achieves this by implementing an index that works this way. Without the index, a full table scan would be required, plus bitmask prefix match, for each of the ~250.000 candidate rows.

"Conventional" storage in databases is possible with a workaround, e.g. with two long integers denoting each prefix in a MySQL database. But this would require an SQL "between" query. An additional column would be needed to store the prefix length, in order to find the closest match (the most narrow prefix). The built-in inet/cidr data type in PostgreSQL doens't help either because it can't be indexed. With conventional methods, only about 30 lookups per second can be achieved with a database.

Having the data in a real database makes it accessible for other means as well; for instance, it is easily possible to query it for the list of prefixes that an AS announces. In addition, the storage in the database offers the possibility to change and update the data (or even completely replace it) in a simple way, by doing this in transaction, which means that it won't block any running queries.

The implementation here makes the database accessible through a request processing handler inside Apache. For usage outside of Apache, a small libpq-based standalone daemon could be written that queries the database. Alternatively, a small handler could be written for mod_asn that does nothing than read an IP address from a request body (or URL) and return the result (effectively implementing such a specialized server within Apache).

One argument for the ip4r data type in PostgreSQL is that it is IPv6-ready. Some IPv6 autonomous systems already exist (about 800 as of the beginning of 2009).

Update June 2015: There are ~9747 IPv6 ASs (mapped to ~24000 prefixes). ip4r fully supports IPv6 since version 2.x, and is currently rolled out. mod_asn uses this capability from version 1.7 on.

1.3 Usage with MirrorBrain

mod_asn can support mod_mirrorbrain (see http://mirrorbrain.org). mod_mirrorbrain can use the data (set in the subprocess environment) for its mirror selection algorithm.

The `mb` tool that comes with MirrorBrain provide means to query the database:

```
 # mb iplookup mirror.susestudio.com
130.57.19.0/24 (AS3680)
 # mb iplookup mirror.susestudio.com --all-prefixes
130.57.19.0/24 (AS3680)
130.57.0.0/16, 130.57.0.0/20, 130.57.19.0/24, 130.57.32.0/21, 137.65.0.0/16,
147.2.0.0/17, 151.155.0.0/16, 164.99.0.0/16, 192.31.114.0/24, 192.94.118.0/24,
192.108.102.0/24, 192.149.26.0/24, 195.109.215.0/24, 212.153.69.0/24
```

CHAPTER

TWO

LICENSE

CHAPTER

THREE

INSTALLATION

3.1 Prerequirements

A recent enough version of the Apache HTTP server is required. 2.2.6 or later should be used. In addition, the apr-util library needs to be 1.3.0 or newer. This is because the DBD database pool functionality was developed mainly between 2006 and 2007, and reached production quality at the time.

3.2 Installing the ip4r data type into PostgreSQL

You need to install the contributed *ip4r* data type into PostgreSQL. This project is found at http://ip4r.projects.postgresql.org/. To install it, a shared library needs to be built to be loaded into PostgreSQL, and an SQL script needs to be run to make the data type known to PostgreSQL and install functions that use it.

It would be preferable to use a binary package if one exists for your operating system:

openSUSE/SLE rpm package: http://download.opensuse.org/repositories/server:/database:/postgresql/

The Debian or Ubuntu package is called postgresql-*VERSION*-ip4r. For Ubuntu 14.04 it's postgresql-9.3-ip4r, for Debian Jessie it's postgresql-9.4-ip4r. The install command would look like this:

```
apt-get install postgresql-9.4-ip4r libapache2-mod-asn
```

Gentoo portage overlay: http://github.com/ramereth/ramereth-overlay/tree

If a manual install is required, you need the PostgreSQL devel package of your operating system and compile a shared library, following the procedure described in the installation instructions provided with the software.

After installing the shared object by package or manual install, you will need to register that extension in your database(s). In old PostgreSQL-versions (up to 9.1) you run a SQL script provided with the ip4r sources, in later versions you instead register an extension:

```
su - postgres
psql -c "CREATE EXTENSION ip4r" template1                    # on Debian/Ubuntu >=␣
↪postgre 9.1
psql -f /usr/share/postgresql/8.4/contrib/ip4r.sql template1  # on Debian/Ubuntu <␣
↪postgre 9.1
psql -f /usr/share/postgresql-ip4r/ip4r.sql template1         # on openSUSE
```

`template1` means that all databases that are created later will have the datatype available. To install it onto an existing database, use your database name instead of "template1".

For instance, if you have an existing `mirrorbrain` database, you would install the data type on it like this:

```
su - postgres
psql -c "CREATE EXTENSION ip4r" mirrorbrain                    # on Debian/Ubuntu >=␣
↪postgre 9.1
psql -f /usr/share/postgresql/8.4/contrib/ip4r.sql mirrorbrain  # on Debian/Ubuntu <␣
↪postgre 9.1
psql -f /usr/share/postgresql-ip4r/ip4r.sql mirrorbrain         # on openSUSE
```

It is normal to see a a good screenful of output printed out by the above **psql** command.

3.3 Creating the database table

Assuming that a database exists already, execute the following command to install the pfx2asn table into it. The `asn.sql` file ships with mod_asn:

```
psql -U <dbuser> -f asn.sql <dbname>
```

Note: The command creates a table named *pfx2asn* in the database named <dbname>. Since the table name is used in some other places, so you should not change its name.

Example: assuming the database already exists (when installing MirrorBrain) and you are on Debian:

```
su - mirrorbrain
psql -f /usr/share/doc/libapache2-mod-asn/asn.sql
```

If you see some `NOTICE` printed out by the command, that's normal; it's due to the default logging setup of PostgreSQL which is verbose.

3.4 Config file for the import script

If you happen to have a MirrorBrain setup, you'll have a configuration file named `/etc/mirrorbrain.conf`, which is automatically used by the **asn_import** script. No further configuration is needed then. If you have several MirrorBrain instances, the instance into which to import the data can be selected with the `-b` commandline option.

Alternatively, you need to create config file with the database connection info, named `/etc/asn_import.conf`, looking like this:

```
[general]
user = database_user
password = database_password
host = database_server
dbname = name_of_database
```

3.5 Load the database with routing data

The data is downloaded and imported into the database with the following command:

```
asn_get_routeviews | asn_import
```

It is recommendable to run the command as unprivileged user, for safety reasons (as any network client).

It might take a few minutes to download and process the data - about 30MB are downloaded, and the data is about 1GB uncompressed (as of 2009) (2010: 13MB compressed, 0.5G uncompressed). The script has to process over 5 million entries, and it is optimized for that job.

In the postgresql database, the data set will be small again.

The command shown above can be used to update the database with fresh routeviews data, by just running it again. This is explained in the next section.

3.6 Keep the data up to date

The data changes almost constantly, but most of the changes will be microscopic and won't directly matter to you. However, you should regularly update from time to time. A weekly (or even monthly) schedule could be entirely sufficient, depending on what you use the data for.

Warning: You should be aware of the fact that routeviews.org kindly provides this data to the public, and you should use their bandwidth with consideration.

Therefore, the MirrorBrain project provides a daily mirror at http://mirrorbrain.org/routeviews/ containing the latest snapshot. This location is used by the provided scripts.

The same command as you ran initially can be used to update the database with fresh routeviews data, by just running it again. This works in production while the database is in active use; it is done in a way that doesn't block any ongoing connections.

Note: The tarball with the data snapshot will be downloaded only if it doesn't exist already in the current working directory. To redownload it, remove the file first.

A cron snippet for running the script daily to download and import the data could look as shown below:

```
35 2 * * *   mirrorbrain  sleep $(($RANDOM/16)); asn_get_routeviews | asn_import
```

If you have a MirrorBrain setup, and possibly several MirrorBrain instances, you could update each database like this:

```
# update ASN data in all MB instances
35 2 * * *   mirrorbrain  sleep $(($RANDOM/16)); \
                           for i in $(mb instances); do \
                             asn_get_routeviews | asn_import -b $i; done
```

The `sleep` command serves to randomize the job time a bit, and allows the example to be used verbatim. Also note that in the example the scripts are called without the `.py` extension.

The data is downloaded to the user's home directory in this case. Make sure the script runs in a directory where other users don't have write permissions.

3.7 Install the Apache module

There are binary packages of mod_asn at the following locations:

openSUSE/SLE: http://download.opensuse.org/repositories/Apache:/MirrorBrain/

Debian/Ubuntu: http://download.opensuse.org/repositories/Apache:/MirrorBrain/

Gentoo portage overlay: http://github.com/ramereth/ramereth-overlay/tree

To manually build mod_asn, all you need to do normally is to use **apxs2** with -c to compile and -i to install the module:

```
apxs2 -ci mod_asn.c
```

To enable the module to be loaded into Apache, you typically will have to run a command like the following - depending on your platform:

```
a2enmod asn
```

3.8 Configure Apache / mod_dbd

mod_dbd provides the database connection pool that is used by mod_asn. The module needs to be loaded into Apache:

```
a2enmod dbd
```

The DBD module needs a database adapter which connects to the database.

Put the following configuration into server-wide context:

```
# configure the dbd connection pool.
# for the prefork MPM, this configuration is inactive. Prefork simply uses 1
# connection per child.
<IfModule !prefork.c>
        DBDMin  0
        DBDMax  32
        DBDKeep 4
        DBDExptime 10
</IfModule>
```

As you might note, the cited configuration is relevant for threaded MPMs only. If you plan to use the prefork MPM, you don't need it. You should however consider using a threaded MPM if you intend to serve high volumes of requests, because it will scale better, which is partly due to the fact that the threads within one process can share a common database pool, which results in fewer connections that are better utilized, and persistance of connections.

The database driver needs to be configured as well, by putting the following configuration into *server-wide* **or** *vhost* context. Make the file *chmod 0640* and owned by *root:root*, because it will contain the database password:

```
DBDriver pgsql
DBDParams "host=localhost user=mb password=12345 dbname=mb connect_timeout=15"
```

3.9 Troubleshooting

If Apache doesn't start, or anything else seems wrong, make sure to check Apache's error_log. It usually points into the right direction.

A general note about Apache configuration which might be in order. With most config directives, it is important to pay attention where to put them - the order does not matter, but the context does. There is the concept of directory contexts and vhost contexts, which must not be overlooked. Things can be "global", or inside a <VirtualHost> container, or within a <Directory> container.

This matters because Apache applies the config recursively onto subdirectories, and for each request it does a "merge" of possibly overlapping directives. Settings in vhost context are merged only when the server forks, while settings in directory context are merged for each request. This is also the reason why some of mod_asn's config directives are programmed to be used in one or the other context, for performance reasons.

The install docs you are reading attempt to always point out in which context the directives belong.

3.10 Configure mod_asn

ASLookup

Simply set `ASLookup On` in the directory context where you want it to be active. The shipped config (`mod_asn.conf`) shows an example.

ASSetHeaders

Set `ASSetHeaders Off` if you don't want the data to be added to the HTTP response headers. In that case, the lookup result is only available through the env table for perusal of other Apache modules.

ASIPHeader

The client IP address looked up is the one that the requests originates from. If mod_asn is running behind a frontend server and can't see the original client IP address, the frontend may pass the IP via a header and mod_asn can look at the header instead. You can configure this like below:

```
ASIPHeader X-Forwarded-For
```

ASIPEnvvar

Alternatively, if you need to use mod_rewrite, you can also make mod_asn look at any variable in Apache's subprocess environment for the IP, for instance:

```
ASIPEnvvar CLIENT_IP
```

ASLookupDebug

`ASLookupDebug` can be set to `On` to switch on debug logging. This can be done per directory.

ASLookupQuery

You may use the `ASLookupQuery` directive (server-wide context) to define a custom SQL query. The compiled in default is:

```
SELECT pfx, asn FROM pfx2asn WHERE pfx >>= ip4r(%s) ORDER BY ip4r_size(pfx) LIMIT 1
```

3.11 Testing

Once mod_asn is configured, you should be able to verify that it works by doing some arbitrary request and looking at the response:

```
 % curl -sI 'http://download.opensuse.org/distribution/11.1/iso/openSUSE-11.1-Addon-Lang-
↪i586.iso'
HTTP/1.1 302 Found
Date: Fri, 26 Jun 2009 22:35:50 GMT
Server: Apache/2.2.11 (Linux/SUSE)
X-Prefix: 87.78.0.0/15
X-AS: 8422
X-MirrorBrain-Mirror: ftp.uni-kl.de
X-MirrorBrain-Realm: country
Location: http://ftp.uni-kl.de/pub/linux/opensuse/distribution/11.1/iso/openSUSE-11.1-
↪Addon-Lang-i586.iso
Content-Type: text/html; charset=iso-8859-1
```

(The *X-Prefix* and *X-AS* headers are not present in the response if mod_asn is configured with `ASSetHeaders Off`.

When testing with local IP addresses like 192.168.x.x, there's not much to look up. These addresses are reserved for local use (see **RFC 1918**). You could however play with sending X-Forwarded-For headers, provided that you configured "ASIPHeader X-Forwarded-For", and can lookup arbitrary IPs thereby. You can use **`curl`** with the following option, causing it to add an X-Forwarded-For header with arbitrary value to the request headers:

```
% curl -sv -H "X-Forwarded-For: 128.176.216.184" <url>
```

It can be helpful to set `ASLookupDebug On` for some directory - you'll see every step which the module does being logged to the error_log.

3.12 Logging

Since the data being looked up is stored in the subprocess environment, it is trivial to log it, by adding the following placeholder to the `LogFormat`:

```
ASN:%{ASN}e P:%{PFX}e
```

That's it!

Questions, bug reports, patches are welcome at mirrorbrain@mirrorbrain.org.

CHAPTER

FOUR

UPGRADING

4.1 Upgrading PostgreSQL (including ip4r data)

When upgrading PostgreSQL, it is important to look at the version number difference. If the third digit changes, no special procedure is needed (except when the release notes explicitely hint about it).

When the first or second digit change, then a dump-and-reload cycle is usually needed.

For instance, when upgrading from 8.3.5 to 8.3.7 nothing needs to be done. When upgrading from 8.3.7 to 8.4, you need to dump and reload.

You might want to follow the instructions that your vendor provides. If your vendor doesn't provide an upgrade procedure, be warned that the database needs to be dumped before upgrading PostgreSQL.

See `pg_dumpall(1)` for how to dump and reload the complete database.

Warning: If your vendor's upgrade procedure automatically saves the previous PostgreSQL binaries in case they are needed later, the procedure might not take into account that the `ip4r.so` shared object might need to be saved as well. Hence, you might be unable to start the old binaries, when the ip4r shared object has been upgraded already.

Hence, it is recommended that you do a complete dump of the databases before upgrading, and load that after upgrading.

Note: When upgrading to 8.4, `ident sameuser` is no longer a valid value in `pg_hba.conf`. Replace it with `ident`.

CHAPTER

FIVE

GLOSSARY

Apache env table Within the Apache HTTP server, a table that is maintained during request processing. It is used by handlers that run during request processing to pass along arbitrary data to be available in later phases. Hense the name, environment table.

Autonomous system (AS) A collection of connected Internet Protocol (IP) routing prefixes under the control of one or more network operators that presents a common source files for one Sphinx project. See http://en.wikipedia.org/wiki/Autonomous_system_number for more information.

ip4r data type An indexable data type for ranges of IPv4 addresses. PostgreSQL has builtin data types for IP addresses and ranges ('inet' and 'cidr'), but they cannot be indexed. See http://ip4r.projects.postgresql.org/

MirrorBrain A download redirector and Metalink generator, which can use mod_asn to refine the selection of content mirror servers. See http://mirrorbrain.org/

Network prefix A topological space of addresses of networked computers grouped together, which is significant for routing decisions. See http://en.wikipedia.org/wiki/Subnetwork for a detailed introduction.

Patricia trie A specialized data structure based on a so-called "trie", used to store a set of strings. In the case of IP addresses / network prefixes, it is known to perform best for efficient lookups. See http://en.wikipedia.org/wiki/Radix_tree for more info.

CHAPTER

SIX

RELEASE NOTES/CHANGE HISTORY

6.1 Release 1.7 (TBA)

mod_asn now supports AS lookups of IPv6 addresses. This requires a current version of the contrib ip4r PostgreSQL data type (which despite its name now supports IPv6), and a slightly changed database scheme - it is probably simplest if you drop the `pfx2asn` database table and recreate it.

6.2 Release 1.6 (r100, Jan 7, 2014)

This release adjusts for the API changes in Apache 2.4. Thanks Cristian Rodriguez for the help. (issue 128, issue 129)

This release also fixes a bug in the **`asn_get_routeviews`** script: It could fail when the BGP routing data snapshot contains bogus AS numbers. (issue 93) Patch courtesy of *agy*.

`asn_get_routeviews` now allows to only download routing data, but don't process it, by using the switch `--download-only`. In addition, `--no-download` can be used if the data is distributed by other means, e.g. with distro updates. Thanks Dagobert Michelsen for the suggestion! (issue 127)

This release also adds documentation.

6.3 Release 1.5 (r88, Sep 5, 2010)

This release fixes one important bug, and improves documentation.

- mod_asn now avoids lookups of IPv6 addresses. The database of AS (autonomous system) numbers is IPv4-only, and in addition, attempted lookups seem to cause problems within the PostgreSQL `ip4r` contrib data type. The symptom was a failure of the database after a while of running, and subsequent error messages from Apache. See issue 58.
- The used version of the APR/APR-Util library is now checked when Apache starts, and not when the module is compiled. This is useful to choose the correct way to access the database, which unfortunately changed between the 1.2 and 1.3 (APR-Util) release. This change makes the deployment more robust, because even if a user mixes packages from different distro versions on a system, mod_asn will still work correctly. This improves the existing fix for issue 7.
- The documentation has been updated with
  - updated examples of Debian package names and filenames
  - an improved example about installing onto an existing database

6.4 Release 1.4 (r79, Mar 27, 2010)

This release does not bring about significant user-visible changes, but under the hood, some optimizations were done.

- For more efficient database connection usage, mod_asn now closes the used connection when its handler quits. Before, a connection with lifetime of the request was acquired; if a long-running handler runs after mod_asn, this could mean that the connection is blocked for other threads until the end of the request. This could occur, for instance, when mod_mirrorbrain ran later, but exited early because a file was supposed to be delivered directly. This was tracked in issue 44.
- Database errors from the lower DBD layer are now resolved to strings, where available. In relation to this: if an IP address is not found it isn't necessarily an error, because it could be a private IP, for instance, which is never present in global routing tables. That case is now logged with NOTICE log level.
- When compiling mod_asn with the Apache Portable Runtime 1.2, different semantics are used to access database rows, couting from 0 instead of from 1. It seemed to work either way (maybe because only a single row is accessed), but hopefully now it is done more correctly and therefore safer in the future. See issue 29 and issue 7 for the context.
- In the documentation, the support scripts are now mentioned without their `.py` suffix in the example for data import, which might be less confusing.

6.5 Release 1.3 (r70, Jul 30, 2009)

- Bugs in the **asn_get_routeviews** and **asn_import** scripts were fixed:
  - The logic which decided whether to download the routing data snapshot file was fixed. If **asn_get_routeviews** is called and it finds a file which was downloaded less then 8 hours ago, the file is reused. If no file exists or the file is older than 8 hours, it is downloaded again.
  - Deletion of existing entries in the database is now prevented, if not at least one entry has been imported. This fixes a bug where the routing data would be deleted if the script was called with no input.

6.6 Release 1.2 (Jul 28, 2009)

- **asn_get_routeviews** script:
  - download data from the mirror provided by the MirrorBrain project, so routeviews.org doesn't get additional traffic by additional users downloading from them
- the documentation has been moved into a docs subdirectory, and rewritten in reStructured Text format, from which HTML is be generated via Sphinx (http://sphinx.pocoo.org/). When the documentation is changed in subversion, the changes automatically get online on http://mirrorbrain.org/mod_asn/docs/
- documentation updates
  - section *Keep the data up to date* added
  - add *Upgrading* notes about PostgreSQL (8.4)
  - install the new documentaion when building Debian or RPM packages
- "debian" subdirectory added, for Debian package builds
- the Subversion repository was moved to http://svn.mirrorbrain.org/svn/mod_asn/trunk/

6.7 Release 1.1 (Jul 4, 2009)

- mod_asn.c:
  - bump version (1.1)
  - update year in copyright header
- **asn_import** script:
  - be able to read config from /etc/asn_import.conf or /etc/mirrorbrain.conf; thus, the script doesn't need to be edited any longer with database configuration data and credentials.
    1. if a MirrorBrain config file is found, it is used (and the MirrorBrain instance can be selected with -b on the commandline, if needed)
    2. alternatively, the script looks for a config file named /etc/asn_import.conf.
- **asn_get_routeviews** script:
  - handle the slightly changed format of routeviews data
  - more sanity checks for parsing newer routing data
- INSTALL:
  - add links to binaries for Debian and ebuilds for Gentoo
  - add instructions for troubleshooting and testing
  - correct a wrong example of loading mod_asn instead of mod_dbd
  - added example for cron snippet for updating the routing database
  - documentation about the newly supported config file
- add debian subdirectory for building Debian packages

6.8 Release 1.0 (Mar 31, 2009)

- mod_asn.c:
  - fix bug that lead to ignorance of variables in the subprocess environment set by ASIPEnvvar, which falsely looked for the wrong variable name (one that was configured via ASIPHeader).
- document an example how to log the looked up data

6.9 Older changes

Please refer to the subversion changelog: http://svn.mirrorbrain.org/svn/mod_asn/trunk respectively http://svn.mirrorbrain.org/viewvc/mod_asn/trunk/

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- search
- *Glossary*

INDEX