# Table of Contents

Installing a Frontier squid cache server.................................................................1

Support......................................................................................................................1
Why use frontier-squid instead of regular squid?......................................................1
Hardware....................................................................................................................2
Software....................................................................................................................3

Puppet.......................................................................................................................3
Upgrading..................................................................................................................3
Preparation...............................................................................................................4
Installation...............................................................................................................4

Configuration..........................................................................................................5

- Moving disk cache and logs to a non-standard location........................................5
- Changing the size of log files retained...................................................................6
- Enabling monitoring.............................................................................................7
- Enabling discovery through WLCG Web Proxy Auto Discovery..............................7

Testing the installation................................................................................................7

Log file contents........................................................................................................8

Common issues.........................................................................................................9

- Inability to reach full network throughput............................................................9
- Log compression interfering with squid operation................................................9
- Running out of file descriptors.............................................................................9
- Conntrack failures..............................................................................................10

Alternate configurations........................................................................................10

- Restricting the destination..................................................................................10
- Running multiple squid workers.........................................................................11
- Rock cache...........................................................................................................11
- Running multiple services..................................................................................12
- Running multiple servers..................................................................................13
- Having squid listen on a privileged port...............................................................13
- Sending logs through syslog...............................................................................14
- Personal squid on a desktop/laptop.......................................................................15
  - Laptop disconnected network operation.............................................................15
- Manually managing squid.conf............................................................................15
- SELinux...............................................................................................................16
Installing a Frontier squid cache server

NOTE: these are the instructions for the frontier-squid package based on squid-4. For instructions for the frontier-squid2 package (with a "2" suffix on all the paths) based on squid-2 see InstallSquid2. For the instructions that used to be for the old frontier-squid package based on squid-2 see OldInstallSquid. The most significant feature that squid-3 and squid-4 have which squid-2 doesn’t is IPv6 support, but squid-3 and squid-4 are not completely upward compatible with squid-2; see the Upgrading section below for details of the differences.

The frontier-squid software package is a patched version of the standard squid http proxy cache software, pre-configured for use by the Frontier distributed database caching system. This installation is recommended for use by Frontier in the LHC CMS & ATLAS projects, and also works well with the CernVM FileSystem. Many people also use it for other applications as well; if you have any questions or comments about general use of this package contact frontier-talk@cern.ch.

Note to Open Science Grid users: this same package is also available from the Open Science Grid so it will probably be more convenient to you to follow the OSG frontier-squid installation instructions.

Note to users of EGI's UMD repository: the same package is also available in UMD so it might be easier for you to get it from there.

After completing a squid installation and configuration, CMS users should follow these further instructions for CMS squids. All WLCG users should register their squids with the WLCG.

Here is what is on this page:

Support

If you have any problems with the software or installation, or would like to suggest an improvement to the documentation, please submit a support request to the Frontier Application Development JIRA.

For rapid response to configuration questions, send e-mail to wlcg-squid-ops@cern.ch. Most questions can also be answered on the user's mailing list frontier-talk@cern.ch.

Why use frontier-squid instead of regular squid?

The most important reason is that with frontier-squid you get the benefit of many years of collective operational experience on the WLCG. The frontier-squid package contains configuration defaults and bug fixes that are known to work well with the applications used on the grid, plus some extra features in the packaging (see below).

The most important feature of frontier-squid is that it correctly supports the HTTP standard headers Last-Modified and If-Modified-Since better than other distributions of squid. The Frontier distributed database caching system, which is used by the LHC projects ATLAS and CMS, depends on proper working of this feature, so that is a big reason why the WLCG maintains this squid distribution. All currently released versions of squid-3 and squid-4 do not correctly support this feature, as documented in the infamous squid bug #7. Although the frontier-squid package expressly supports If-Modified-Since, it also works well with applications that do not require If-Modified-Since including CVMFS. The collapsed_forwarding feature is also missing from all versions of squid-3 prior to squid-3.5 (including the one in Red Hat EL6), and it is important for all grid applications that use squid and is enabled in the frontier-squid package by default. Also collapsed_forwarding with If-Modified-Since was not fixed until 3.5.27, which is newer than the default squid in Red Hat EL7. More details are in the note at the top of the MyOwnSquid twiki page.
In addition, the package has several additional features including these:

1. A configuration file generator, so configuration customizations can be preserved across package upgrades even when the complicated standard configuration file changes.
2. Automatic cleanup of the old cache files in the background when starting squid, to avoid problems with cache corruption.
3. Default access control lists to permit remote performance monitoring from shared WLCG squid monitoring servers at CERN.
4. The default log format is more human readable and includes contents of client-identifying headers.
5. Access logs are rotated throughout the day if they reach a configured size, to avoid filling up disks of heavily used squids. The logs are also compressed by default.
6. Multiple independent squid 'services' using the same configuration can be easily started on the same machine.
7. An add-on 'frontier-awstats' package that feeds into the WLCG Squid Monitoring awstats monitor that is useful primarily for identifying the source and type of requests. This package is most used on publicly-accessible squids.

**Hardware**

The first step is to decide what hardware you want to run the squid cache server on. These are some FAQs.

1) Do I need to dedicate a node to squid and only squid?

This is up to you. It is a strongly recommended. It depends on how many jobs try to access the squid simultaneously and what else the machine is used for (see question 2). Large sites may need more than one squid (see question 4). The node needs to have network access to the internet, and be visible to the worker nodes. Virtual machines can help isolate other uses of a physical machine, but it doesn't isolate disk and especially network usage so they can be problematic.

2) What hardware specs (CPU, memory, disk cache)?

For most purposes 2 cores at 2GHZ, 4GB memory, and 100 GB for the disk cache should be adequate. This excludes the space needed for log files which is determined by how heavily the system is used and what the clean up schedule is. The default in the rpm always rotates the logs every day and removes the oldest log after 10 rotates, and four times an hour it will also rotate if the access log is bigger than 5GB. By default logs are compressed after rotate and typically are reduced to less than 15% of their original size, so allowing 12GB for logs should be sufficient. On heavily used systems the default will most likely keep logs for too short of a time, however, so it's better to change the default (instructions below) and allow at least 25GB for logs.

From what we have seen, the most critical resource is the memory. If the machine serves other purposes, make sure the other tasks don't use up all the memory. Squid runs as a single thread, so if that is the only use of the machine, having more than 2 cores is a waste (unless you are running multiple squid workers). You should also avoid network filesystems such as AFS and NFS for the disk cache.

Here is a description of normal squid memory usage: If you have a decent amount of spare memory, the kernel will use that as a disk cache, so it's a good chance that frequently-Requested items will, in fact, be served from RAM (via the disk cache) even if it's not squid's RAM. Let cache_mem handle your small objects and the kernel handle the larger ones. The default frontier-squid configuration prevents large objects from going in the memory cache. There is an exception to this rule when using the 'rock' cache_dir type; see the details in the Rock cache section.

3) What network specs?

**Why use frontier-squid instead of regular squid?**
The latencies will be lower to the worker nodes if you have a large bandwidth. The network is almost always the bottleneck for this system, so at least a gigabit for each squid machine is highly recommended. If you have many job slots, 2 bonded gigabit network connections or a 10-gigabit connection is even better. Squid on one core of a modern CPU can pretty much keep up with 2 gigabits. Each squid process is single-threaded so if you're able to supply more than 2 gigabits, multiple squid processes on the same machine need to be used to serve the full throughput. This is supported (instructions below) but each squid needs its own disk cache space (unless using rock cache).

4) How many squids do I need?

Sites with over 500 job slots should have at least 2 squids for reliability. We currently estimate that sites should have one gigabit on a squid per 1000 grid job slots. A lot depends on how quickly jobs start; an empty batch queue that suddenly fills up will need more squids. The number of job slots that can be safely handled per gigabit increases as the number of slots increase because the chances that they all start at once tends to go down.

5) How should squids be load-balanced?

There are many ways to configure multiple squids: round-robin DNS, load-balancing networking hardware, LVS, etc. The simplest thing to do is just set up two or more squid machines independently and let Frontier handle it by making a small addition to the frontier client configuration to have the client do the load balancing (described for CMS in the section on multiple squid servers). If there are many thousands of job slots, hardware-based load balancers can be easily overloaded, so DNS-based or client-based load balancing will probably be called for.

6) Can I put squid behind a NAT?

Possibly, but if so it should not be the same NAT shared by the worker nodes, otherwise if the squid fails it becomes very difficult to tell on the upstream servers whether it is a badly performing squid or direct connections from the worker nodes. It is much better for the squid to be on a machine with its own public IP address.

Software

The instructions below are for the frontier-squid rpm version >= 3.5.20-1.1 on Redhat Enterprise Linux (RHEL) version 5, 6, or 7 based system. The rpm is based on the frontier-squid source tarball, and there are also instructions for installing directly from the frontier-squid tarball available. Please see the Release Notes for details on what has changed in recent versions. If, for some reason, you prefer to use a non frontier-squid distribution of squid, see MyOwnSquid.

Puppet

A puppet module for configuring frontier-squid is available on puppet-forge which understands a lot of the following instructions. If you're using puppet, check there first.

Upgrading

When upgrading from another frontier-squid-3 or frontier-squid-4 release, a simple yum update frontier-squid should do. When upgrading from the frontier-squid-2 series, it is mostly similar but be aware of the following incompatibilities

1. frontier-squid-3 and frontier-squid-4 support IPv6, so if you have an IPv6 address assigned on the machine and might get incoming IPv6 addresses, you need to include IPv6 addresses in the
The handling of multiple squid processes for high performance is very different. Instead of just creating cache subdirectories, you need to explicitly include `{process_number}` in the cache_dir filename and set the `workers` option. Also the logs and monitoring ports are combined instead of separated. See details below.

3. The handling of cpu core affinity is not done automatically or enabled through a `SETSQUIDAFFINITY` environment variable, it is done through a `cpu_affinity_map` that you need to define. Details are below.

4. The handling of independent squid services is very different. Instead of setting `SQUID_MULTI_PEERING=false`, there are a number of options that you need to set. See details below.

5. For the `accel` mode to the `http_port` option, used for reverse proxies (such as Frontier launchpads), the default of the `vhost` option has changed to be enabled so you need to add no-vhost. That option is always coupled with a `cache_peer` option, which had typically been inserted with `insertline("^http_port", "cache_peer ...")`, but squid-3 and squid-4 have no setting of `http_port` by default so instead use `setoption("cache_peer", "...")`.

6. The `ignore_ims_on_miss` option (which had been commonly used on Frontier launchpads) is not supported.

7. Remove setting `quick_abort_min` to -1, because with it we sometimes see squid-3 and squid-4 crash.

8. There are other squid2 options that are not available, but mostly they are not used on the WLCG. If you are using obscure options, look for documentation of them in `/etc/squid/squid.conf`.

**Preparation**

By default the frontier-squid rpm installs files with a "squid" user id and group. If they do not exist, the rpm will create them. If your system has its own means of creating logins you should create the login and group before installing the rpm. If you want the squid process to use a different user id (historically it has been "dbfrontier"), then for example before installing the rpm create the file `/etc/squid/squidconf` with the following contents:

```bash
export FRONTIER_USER=dbfrontier
export FRONTIER_GROUP=dbfrontier
```

where you can fill in whichever user and group id you choose.

**Installation**

First, if you have not installed any frontier rpm before, execute the following command as the root user:

```bash
# rpm -Uvh http://frontier.cern.ch/dist/rpms/RPMS/noarch/frontier-release-1.1-1.noarch.rpm
```

If it warns about creating `/etc/yum.repos.d/cern-frontier.repo.rpmnew`, then move that file into place:

```bash
# mv /etc/yum.repos.d/cern-frontier.repo.rpmnew /etc/yum.repos.d/cern-frontier.repo
```

Next, install the package with the following command:

```bash
# yum install frontier-squid
```

Set it up to start at boot time with this command:

```bash
# chkconfig frontier-squid on
```
Configuration

Custom configuration is done in /etc/squid/customize.sh. That script invokes functions that edit a supplied default squid.conf source file to generate the final squid.conf that squid sees when it runs. Comments in the default installation of customize.sh give more details on what can be done with it. Whenever /etc/init.d/frontier-squid runs it generates a new squid.conf if customize.sh has been modified. The purpose for this unique technique is in order to allow new rpm versions to update the default configuration, while still allowing the system administrator complete freedom for changing the configuration, including places where the order of options is important (such as the http_access option). customize.sh is guaranteed to be completely under the control of the system administrator; any update to it from the rpm can be safely ignored. There is an option to disable the use of customize.sh in order to manually manage squid.conf; see the details below.

It is very important for security that squid not be allowed to proxy requests from everywhere to everywhere. The default customize.sh allows incoming connections only from standard private network addresses and allows outgoing connections to anywhere. If the machines that will be using squid are not on a private network, change customize.sh to include the network/maskbits for your network. For example:

```
setoption("acl NET_LOCAL src", "188.185.0.0/16 2001:1459:201::/48")
```

The script allows specifying many subnets - just separate them by a blank. Include IPv6 as well as IPv4 addresses if your site has them. If you would like to limit the outgoing connections please see the section below on restricting the destination.

If you want to, you can change the cache_mem option to set the size squid reserves for caching small objects in memory, but don't make it more than 1/8th of your hardware memory. The default 256 MB should be fine, leaving a lot of memory for disk caching by the OS, because squid generally performs best with large objects in disk cache buffers.

Change the size of the cache_dir (the third parameter) to your desired size in MB. The default is only 10 GB which is rather stingy. For example, for 100 GB set it to this:

```
setoptionparameter("cache_dir", 3, "100000")
```

Make sure that there will always be sufficient space in the cache_dir directory, plus at least 10% extra since the cleanup algorithm takes time and might allow the space used to go a little over the specified amount.

Now that the configuration is set up, start squid with this command:

```
# service frontier-squid start
```

To have a change to customize.sh take affect while squid is running, run the following command:

```
# service frontier-squid reload
```

Moving disk cache and logs to a non-standard location

Often the filesystems containing the default locations for the disk cache (/var/cache/squid) and logs (/var/log/squid) isn't large enough and there's more space available in another filesystem. To move them to a new location, simply change the directories into symbolic links to the new locations while the service is stopped. Make sure the new directories are created and writable by the user id that squid is running under. For example if /data is a separate filesystem:

```
# service frontier-squid stop
# mv /var/log/squid /data/squid_logs
# ln -s /data/squid_logs /var/log/squid
```
Alternatively, instead of creating symbolic links you can set the cache_log and coredump_dir options, the second parameter of the cache_dir option, and the first parameter of the access_log option in /etc/squid/customize.sh. For example:

```bash
setoption("cache_log", "/data/squid_logs/cache.log")
setoption("coredump_dir", "/data/squid_cache")
setoptionparameter("cache_dir", 2, "/data/squid_cache")
setoptionparameter("access_log", 1, "daemon:/data/squid_logs/access.log")
```

It's recommended to use the "daemon:" prefix on the access_log path because that causes squid to use a separate process for writing to logs, so the main process doesn't have to wait for the disk. It is on by default for those who don't set the access_log path. Alternatively, the messages can be sent through syslog; see the instructions below.

### Changing the size of log files retained

Rather than using standard logrotate configurations for log rotation, frontier-squid controls log rotation itself. The primary reason for this is that with the Frontier application and many worker node clients, log files tend to grow too quickly and overrun available space.

The access.log is rotated each night, and also if it is over a given size (default 5 GB) when it checks 4 times each hour. You can change that maximum value by exporting the environment variable SQUID_MAX_ACCESS_LOG in /etc/sysconfig/frontier-squid to a different number of bytes. You can also append M for megabytes or G for gigabytes. For example for 20 gigabytes each you can use:

```bash
export SQUID_MAX_ACCESS_LOG=20G
```

By default, frontier-squid compresses log files when they are rotated, and saves up to 9 access.log.N.gz files where N goes from 1 to 9. In order to estimate disk usage, note that the rotated files are typically compressed to a bit under 15% of their original size, and that the uncompressed size can go a bit above $SQUID_MAX_ACCESS_LOG because the cron job only checks four times per hour. For example, for SQUID_MAX_ACCESS_LOG=20G the maximum size will be a bit above 20GB plus 9 times 3GB, so allow 50GB to be safe.

If frontier-awstats is installed (typically only on central servers), an additional uncompressed copy is also saved in access.log.0.

An alternative to setting the maximum size of each log file, you can leave each log file at the default size and change the number of log files retained, for example for 50 files (about 6GB total space) set the following in /etc/squid/customize.sh:

```bash
setoption("logfile_rotate", "50")
```

It is highly recommended to keep at least 3 days worth of logs, so that problems that happen on a weekend can be investigated during working hours. If you really do not have enough disk space for logs, the log can be disabled with the following in /etc/squid/customize.sh:

```bash
setoption("access_log", "none")
```

Then after doing service frontier-squid reload (or service frontier-squid start if squid was stopped) remember to remove all the old access.log* files.
On the other hand, the compression of large rotated logs can take a considerably long time to process, so if you have plenty of disk space and don't want to have the additional disk I/O and cpu resources taken during rotation, you can disable rotate compression by putting the following in /etc/sysconfig/frontier-squid:

```bash
export SQUID_COMPRESS_LOGS=false
```

That uses the old method of telling squid to do the rotation, which keeps access.log.N where N goes from 0 to 9, for a total of 11 files including access.log. When compression is turned off, the default SQUID_MAX_ACCESS_LOG is reduced from 5GB to 1GB, so override that to set your desired size. When converting between compressed and uncompressed format, all the files of the old format are automatically deleted the first time the logs are rotated.

See also the section Log compression interfering with squid operation below.

### Enabling monitoring

The functionality and performance of your squid should be monitored from CERN using SNMP. The monitoring site is [http://wlcg-squid-monitor.cern.ch/](http://wlcg-squid-monitor.cern.ch/).

To enable this, your site should open incoming firewall(s) to allow UDP requests to port 3401 from the CERN IP address ranges, 128.142.0.0/16, 188.184.128.0/17, 188.185.48.0/20 and 188.185.128.0/17. If you have IPv6 enabled or want to be ready for it, also allow the same requests from 2001:1458:300::/46 and 2001:1459:300::/46. When that is ready, register the squid with WLCG to start the monitoring.

Note: some sites are tempted to not allow requests from the whole range of IP addresses listed above, but we do not recommend that because the monitoring IP addresses can and do change without warning. Opening the whole CERN range of addresses has been cleared by security experts on the OSG and CMS security teams, because the information that can be collected is not sensitive information. If your site security experts still won't allow it, the next best thing you can do is to allow the aliases wlcgsquidmon1.cern.ch and wlcgsquidmon2.cern.ch. Most firewalls do not automatically refresh DNS entries, so you will also have to be willing to do that manually whenever the values of the aliases change.

### Enabling discovery through WLCG Web Proxy Auto Discovery

The WLCG maintains a service for Web Proxy Auto Discovery. Most of the squids that can be discovered come from the ATLAS and CMS frontier squid per-site configurations, and that is the preferred way to do it for long term installations. However for short-term squids such as those run in clouds, squids can be added to the service by putting the following in /etc/sysconfig/frontier-squid:

```bash
export SQUID_AUTO_DISCOVER=true
```

After that is done then for organizations (as defined by the Maxmind GeoIP organizations database) that do not have a squid registered through ATLAS or CMS, the squids should show up in http://wlcg-wpad.cern.ch/wpad.dat and http://wlcg-wpad.fnal.gov/wpad.dat within 10 minutes after frontier-squid is started.

This feature makes use of a software package and service called shoal.

### Testing the installation

Download the following python script fnget.py (Do a right-click on the link and save the file as fnget.py)

Test access to a Frontier server at CERN with the following commands:

```bash
$ chmod +x fnget.py #(only first time)
```

Changing the size of log files retained
The response should be similar to this:

Using Frontier URL: http://cmsfrontier.cern.ch:8000/FrontierProd/Frontier
Query: select 1 from dual
Decode results: True
Refresh cache: False

Query started: 10/30/12 20:04:09 CET
Query ended: 10/30/12 20:04:09 CET
Query time: 0.0179278850555 [seconds]

Query result:
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE frontier SYSTEM "http://frontier.fnal.gov/frontier.dtd">
<frontier version="3.29" xmlversion="1.0">
<transaction payloads="1">
<payload type="frontier_request" version="1" encoding="BLOBzip">
<data>eJxjY2BgYDRkA5JsfqG+Tq5B7GxgEXYAGs0CVA==</data>
<quality error="0" md5="5544fd3e96013e694f13b44ee3c" records="1" full_size="25"/>
</payload>
</transaction>
</frontier>

Fields:
1 NUMBER

Records:
1

This will return whatever you type in the select statement, for example change 1 to 'hello'. The "dual" table is a special debugging feature of Oracle that just returns what you send it.

Now to test your squid, replace yoursquid.your.domain in the following command with the name of your squid machine

$ export http_proxy=http://yoursquid.your.domain:3128

and perform the fnget.py test twice again. It should pass through your squid, and cache the response. To confirm that it worked, look at the squid access log (in /var/log/squid/access.log if you haven't moved it). The following is an excerpt:

128.220.233.179 - - [22/Jan/2013:08:33:17 +0000] "GET http://cmsfrontier.cern.ch:8000/FrontierProd/Frontier?type=frontier_request:1:DEFAULT&encoding=BLOBzip HTTP/1.1" 200 810 "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1500.95 Safari/537.36"
128.220.233.179 - - [22/Jan/2013:08:33:19 +0000] "GET http://cmsfrontier.cern.ch:8000/FrontierProd/Frontier?type=frontier_request:1:DEFAULT&encoding=BLOBzip HTTP/1.1" 200 809 "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1500.95 Safari/537.36"

Notice the second entry has a "TCP_MEM_HIT", that means the object was cached in the memory. Any subsequent requests for this object will come from the squid cache until the cached item expires.

Log file contents

Error messages are written to cache.log (in /var/log/squid/access.log if you haven't moved it) and are generally either self-explanatory or an explanation can be found with google.

Logs of every access are written to access.log (also in /var/log/squid/access.log if you haven't moved it) and the default frontier-squid format contains these fields:
1. Source IP address
2. User name from ident if any (usually just a dash)
3. User name from SSL if any (usually just a dash)
4. Date/timestamp query finished in local time, and +0000, surrounded by square brackets
5. The request method, URL, and protocol version, all surrounded by double quotes
6. The http status (result) code
7. Reply size including http headers
8. Squid request status (e.g. TCP_MISS) and heirarchy status (e.g. DEFAULT_PARENT) separated by a colon
9. Response time in milliseconds
10. The contents of the X-Frontier-Id header or a dash if none, then a space, then the contents of the cvmfs-info header, or a dash if none, all surrounded by double quotes (no client sends both so entries will always either start with "- " or end with " -")
11. The contents of the Referer header or a dash if none, surrounded by double quotes
12. The contents of the User-Agent header or a dash if none, surrounded by double quotes

**Common issues**

**Inability to reach full network throughput**

If you have a CPU that can't quite keep up with full network throughput, we have found that up to an extra 15% throughput can be achieved by binding the single-threaded squid process to a single core, to maximize use of the per-core on-chip caches. This is not enabled by default, but you can enable it by putting the following in `/etc/squid/customize.sh`:

```bash
setoption("cpu_affinity_map", "process_numbers=1 cores=2")
```

If that little boost isn't enough, try running multiple squid workers.

**Log compression interfering with squid operation**

Log compression has been observed on at least one machine to interfere with squid operation. That was an old 10-gbit machine with slow disks, high traffic, and 3 squid processes. These are some possible mitigations. Details of how to do many of these things are in the section Changing the size of log files retained section above.

1. Make sure there's a "daemon:" prefix on the access_log if you have changed its value.
2. Reduce the max log size before compression and increase the number of log files retained, to decrease the length of time of each log compression.
3. Disable compression if you have the space.
4. As root run `ionice -c1 -p PID` with the process id of each running squid process, or of each logfile-daemon process if you're using the "daemon:" prefix. This raises their I/O priority above ordinary filesystem operations.
5. As a last resort, disable the access log completely.

**Running out of file descriptors**

By default, frontier-squid makes sure that there are at least 4096 file descriptors available for squid, which is usually enough. However, under some situations where there are very many clients it might not be enough. When this happens, a message like this shows up in cache.log:

```
WARNING! Your cache is running out of filedescriptors
```
There are two ways to increase the limit:

1. Add a line such as `ulimit -n 16384` in `/etc/sysconfig/frontier-squid`
2. Set the `nofile` parameter in `/etc/security/limits.conf` or a file in `/etc/security/limits.d`. For example use a line like this to apply to all accounts:

   `* - nofile 16384`

   or replace the `*` with the squid user name if you prefer.

### Connntrack failures

Heavily loaded systems can sometimes experience system log messages "nf_conntrack: table full, dropping packet" and failed connections from clients. This can be fixed by disabling connection tracking on the squid port with these iptables commands:

```
# iptables -t raw -A PREROUTING -p tcp --dport 3128 -j NOTRACK
# iptables -t raw -A OUTPUT -p tcp --sport 3128 -j NOTRACK
```

Do the same with `ip6tables`, and make sure that the settings get saved for restoring after reboots.

Disabling conntrack has also been reported to improve network stack performance by 20%, although we have not verified it.

Note: the above PREROUTING rule has been observed to make a port forwarding rule from the same port have no effect (it works to use it on the destination of the port forward), and the OUTPUT rule has been observed to block the response from a port forwarding rule. One user observed that the PREROUTING rule alone fixed the conntrack failures.

### Alternate configurations

#### Restricting the destination

The default behavior is to allow the squid to be used for any destination, and that’s the recommendation for most sites. If you want to restrict the destinations anyway, frontier-squid provides some pre-defined access controls for the most common destinations on the WLCG. They are

1. CMS_FRONTIER - CMS Frontier conditions data servers
2. ATLAS_FRONTIER - ATLAS Frontier conditions data servers
3. MAJOR_CVMFS - the major WLCG CVMFS stratum 1 servers

In addition, there are two commented out lines using a general `RESTRICT_DEST` access control which you can use to set a regular expression that restricts connections to any set of hosts of your choice.

To use one of the pre-defined access controls, use two lines like this (for example with CMS_FRONTIER):

```
uncomment("acl CMS_FRONTIER")
insertline("^# http_access deny !RESTRICT_DEST", "http_access deny !CMS_FRONTIER")
```

To use a combination of two of the pre-defined acls, use "http_access allow" followed by "http_access deny !", for example:

```
uncomment("acl CMS_FRONTIER")
uncomment("acl MAJOR_CVMFS")
insertline("^# http_access deny !RESTRICT_DEST", "http_access allow CMS_FRONTIER")
```
If for some reason you want to have a different destination or destinations you can instead use a regular expression with the RESTRICT_DEST lines, for example:

```bash
setoptionparameter("acl RESTRICT_DEST", 3, "^\{(cms|atlas).*frontier.*)\.cern\.ch|frontier.*\.racf\.bnl\.gov\}$")
``` 

Once you have restricted the destination, it isn't so important anymore to restrict the source. If you want to leave it unrestricted you can change the NET_LOCAL acl to 0.0.0.0/0 (unless you want to restrict both):

```bash
setoption("acl NET_LOCAL src", "0.0.0.0/0")
```

## Running multiple squid workers

If you have either a particularly slow machine or a high amount of bandwidth available, you probably will not be able to get full network throughput out of a single squid process. For example, our measurements with a 10 gigabit interface on a 2010-era machine with 8 cores at 2.27Ghz showed that 3 squids were required for full throughput. The amount of bandwidth that each core can handle has not seemed to change much since then, so generally 3 or 4 squid processes are recommended for 10gbit/s network interfaces.

Multiple squids can be enabled very simply by doing these steps:

- Stop frontier-squid and remove the old cache (/var/cache/squid/* if you didn't change it)
- Add the following options in /etc/squid/customize.sh to add (for example) 3 worker processes. If there are more than 3 squid workers, increase the workers option and both lists of numbers in the `cpu_affinity_map`.

```bash
setoption("workers", 3)
setoptionparameter("cache_dir", 2, "/var/cache/squid/squid${process_number}")
setoption("cpu_affinity_map", "process_numbers=1,2,3 cores=2,3,4")
```

- Optionally, if you want to be able to debug problems that might be related to different worker processes behaving differently, add these options to put the worker name into access.log and its number in http headers:

```bash
setoptionparameter("logformat awstats", 3, "kid${process_number}")
setoption("visible_hostname", `'uname -n'/${process_number}`)
```

- Start frontier-squid again.

This will share everything but the disk cache. Be aware that each worker can use up to the total amount of space set in the cache_dir parameter 3. Divide the total amount of space you want to allow by the number of workers. For example with 3 workers and a cache_dir 3rd parameter of 100000, up to 300GB will be used. The subdirectories for the caches will be automatically created if their parent directory is writable by the user id that squid is run under.

If you want to revert to a single squid, reverse the above process including cleaning up the corresponding cache directories.

## Rock cache

The rock cache has a great potential because it supports sharing a single cache between multiple workers. It's the last piece that makes the sharing between workers complete. All the cached objects are stored in a single sparse file as a kind of database. Unfortunately, it has a bad bug that makes it unusable for any application based on If-Modified-Since, including Frontier, when used with collapsed_forwarding: it does not re-validate...
expired objects so they behave as if they’re not cached at all. (squid-3 had a related bug with or without collapsed_forwarding). If you can guarantee that it will only be used by applications such as CVMFS that do not use If-Modified-Since, here is how to configure it.

Since rock cache is implemented as a separate process, it becomes a performance bottleneck unless you use squid’s own shared memory cache to cache all objects. The default configuration of frontier-squid only stores small objects in the memory cache, so it requires changing that limit and making the memory cache to be a large size. On the other hand when not using rock cache it works best to maximize use of the kernel file system buffers instead of squid’s memory cache for large objects (although that experience is more from squid-2, we do not have a lot of performance comparisons with it in squid-3 or squid-4).

So use these options in /etc/squid/customize.sh to use rock cache:

```bash
# with rock cache store every object in cache_mem, use roughly 75% of available memory
setoption("cache_mem", "24 GB")
setoption("maximum_object_size_in_memory", "1 GB")
setoption("cache_dir", "rock /var/cache/squid 100000")
setoption("memory_cache_shared", "on")
```

Even when using multiple workers, do not include ${process_number} in the cache_dir path, because one cache is shared for all the workers. The memory_cache_shared option above should be on for multiple workers with rock cache (it is irrelevant when there is only one worker).

**Running multiple services**

To run multiple independent squid services on the same machine add a setting like this to

```
/etc/sysconfig/frontier-squid:
```

```bash
export SQUID_NUM_SERVICES=3
```

or however many services you want. A squid.conf configuration macro ${service_number} varies in value from 0 to $$SQUID_NUM_SERVICES-1. When using this feature, you must include ${service_number} in the cache_dir, access_log, cache_log, and pid_filename options to separate the services from each other. Also there is a macro `setserviceoption` that enables you to change the values of numerical options for each service name. First set environment variables in the bash part of `/etc/squid/customize.sh`:

```bash
WORKERS=1
SERVICES=${SQUID_NUM_SERVICES:-1}
HOSTNAME=`hostname`
```

Then in the awk portion use these options:

```bash
setoptionparameter("cache_dir", 2, "/var/cache/squid/squid${service_name}")
setoptionparameter("access_log", 1, "daemon:/var/log/squid/squid${service_name}/access.log")
setoption("cache_log", "/var/log/squid/squid${service_name}.cache.log")
setoption("pid_filename", "/var/run/squid/squid${service_name}.pid")
setoption("visible_hostname", "'$HOSTNAME'/${service_name}")
setserviceoption("http_port", ",", 3128, '${SQUID_NUM_SERVICES}', -1)
setserviceoption("snmp_port", ",", 3401, '${SQUID_NUM_SERVICES}', 1)
# the number of cores in the lists should be at least as much as $WORKERS
setserviceoption("cpu_affinity_map", "process_numbers=1,2 cores=", "2,3", '${SQUID_NUM_SERVICES}', 'SQUID_NUM_SERVICES')
```

This sets the http_port to 3128 in service 0, 3127 in service 1, etc, and sets snmp_port to 3401 in service 0, 3402 in service 1, etc. For details on the parameters to the `setserviceoption` macro see the comments in `/etc/squid/customhelps.awk`.

Note that extra disk space and memory will be used for every service; the only thing shared between the
services is the configuration. If using more than one worker for each service without rock cache, increase the value of $WORKERS and include both ${service_name} and ${process_number} in the cache_dir path.

Different options can be set for different services by enclosing them in squid.conf macros like

```
if ${service_name} = 0
  <options for service 0>
else
  <options for services other than 0>
endif
```

Note, however, that squid does not support nesting if statements with different macros, so you can not set different configurations for different worker process numbers within particular service numbers. The if statements are also very simple, and there's no other form that you can use than the above simple form (the right hand side even has to be an integer, so that's why frontier-squid sets the service "name" to integers). The else portion is optional. The setserviceoption macro simply generates if statements. If you want custom options for different services it is probably easiest to do it outside of the awk section of customize.sh, probably at the end to append options to the end of squid.conf, otherwise it takes a lot of ugly insertline statements.

**Running multiple servers**

As mentioned above, any site with over 500 job slots should run at least two squid servers for reliability. Squid has a feature called "cache_peer sibling" along with "icp_port" that enables squids to first check their peer squid(s) for a cached item that they need before going to the origin server. Unfortunately, this option is incompatible with the "collapsed_forwarding" feature in that if both are enabled, it leads to deadlocks. Collapsed_forwarding is much more important than cache_peer sibling, because tens (typically) of jobs or even more can request the same object at the same time when new data needs to be loaded into a squid, where cache_peer sibling only reduces the queries sent upstream in half (if there are two squid servers). With collapsed_forwarding enabled, when many jobs that read the same data start close together in a batch, even if there's a delay between starting each job, they tend to synchronize themselves while waiting for initial loading of the cache because those that start later immediately read cached items until they catch up with the rest that are waiting for the initial loading.

So our recommendation with multiple squid servers is to not enable cache_peer configurations. It is possible to use a cache_peer parent configuration, where one squid always reads from another, but the most straightforward usage of that eliminates the reliability advantage of multiple servers, adding a new single point of failure. The CMS online squid configuration with a squid on every node does use this feature, listing multiple cache_peer parents in combination with monitorurl/monitortimeout options. The complication is not worth it, however, unless there are many squids involved.

With multiple workers in squid-3 and squid-4, each of the workers also independently query the upstream server. This is still a great improvement over having all jobs send their queries upstream, by a couple of orders of magnitude, so the difference between one worker and multiple workers sending their queries to the origin server is not very significant. When we are able to upgrade to squid-4 this problem should be solved by the use of rock cache.

**Having squid listen on a privileged port**

This package runs squid strictly as an unprivileged user, so it is unable to open a privileged TCP port less than 1024. The recommended way to handle that is to have squid listen on an unprivileged port and use iptables to forward a privileged port to the unprivileged port. For example, to forward port 80 to port 8000, use this:

```
# iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8000
```

**Running multiple services**
You can change the port that squid listens on with this in /etc/squid/customize.sh:

```
setoption("http_port","8000")
```

## Sending logs through syslog

The access logs can be configured to go through the syslog service. This is not recommended for very high volume squids that serve worker nodes, because it causes extra overhead, but it can work OK for relatively low volume squids and may be desired for sending the information to an additional destination.

To configure it for a single service, choose a "local" syslog destination that isn't currently in use, for example "local4":

```
setoptionparameter("access_log", 1, "syslog:local4.info")
```

This should be paired with the following lines in /etc/rsyslog.conf, placed before the /var/log/messages line:

```
local4.* /var/log/squid/access.log
local4.* ~
```

The directory of access.log must be the same as that of the cache_log parameter. The second line discards the messages from other processing, keeping the messages out of /var/log/messages.

You'll need to increase the log rate limits. Here are suggested settings, but you may need to increase them further if log messages are dropped. If you increase the number, increase all *RateLimitBurst settings together. In /etc/rsyslog.conf:

```
$SystemLogRateLimitBurst 10000
$SystemLogRateLimitInterval 5
```

In addition, on systemd-based systems add the following:

```
$imjournalRatelimitBurst 10000
$imjournalRatelimitInterval 5
```

and also the following in /etc/systemd/journald.conf:

```
RateLimitBurst=10000
RateLimitInterval=5s
```

Restart the rsyslog service after changing rsyslog.conf, and restart systemd-journald after changing journald.conf.

After the new file is created (once there is some activity logged), make it owned by the squid user (which is "squid" or $FRONTIER_USER if the default was changed) and world readable, for example

```
chown squid:squid /var/log/squid/access.log
chmod 644 /var/log/squid/access.log
```

This is important because log rotation is done as the squid user.

For multiple services, each service should use a separate syslog destination beginning with local0, and a separate subdirectory:

```
setoptionparameter("access_log", 1, "syslog:local${service_name}.info")
```

### Having squid listen on a privileged port

14
Then add corresponding lines in `/etc/rsyslog.conf` for each of the services:

```sh
local0.* /var/log/squid/squid0/access.log
local0.* ~
local1.* /var/log/squid/squid1/access.log
local1.* ~
local2.* /var/log/squid/squid2/access.log
local2.* ~
```

Also remember to restart rsyslog and make the files owned by the squid user and world readable.

### Personal squid on a desktop/laptop

If you want to install a Frontier squid on your personal desktop or laptop, just follow the same instructions as under Software above, except:

- For the `NET_LOCAL` acl, use "127.0.0.1/32"
- For the `cache_dir` size you can leave it at the default 10000 or even perhaps cut it down to 5000 if you want to.

#### Laptop disconnected network operation

If you want to be able to run a laptop disconnected from the network, add the following to `customize.sh`:

```sh
setoption("cachemgr_passwd", "none offline_toggle")
```

Then, load up the cache by running your user job once while the network is attached, and run the following command once:

```sh
squidclient mgr:offline_toggle
```

It should report "offline_mode is now ON" which will prevent cached items from expiring. Then as long as everything was preloaded and the laptop doesn't reboot (because starting squid normally clears the cache) you should be able to re-use the cached data. You can switch back to normal mode with the same command or by stopping and starting squid.

To prevent clearing the cache on start, put the following in `/etc/sysconfig/frontier-squid`:

```sh
export SQUID_CLEAN_CACHE_ON_START=false
```

If you do that before the first time you start squid (or if you ever want to clear the cache by hand), run this to initialize the cache:

```sh
# service frontier-squid cleancache
```

### Manually managing squid.conf

In order to disable the use of `customize.sh` and manually manage `squid.conf`, you can set the following in `/etc/sysconfig/frontier-squid`:

```sh
export SQUID_CUSTOMIZE=false
```

Note that this has the disadvantages of not automatically getting updates to `squid.conf` from new versions of the rpm and of not having access to the macros supported in `customize.sh`. When using this option, it is recommended to automate applying edits to `/etc/squid/squid.conf.frontierdefault` and generating `/etc/squid/squid.conf` from that so you will get any updated defaults from new versions of the rpm as
customize.sh does. If that is not possible, then start from a copy of /etc/squid/squid.conf.default as a minimum configuration since it will change less often than squid.conf.documented or squid.conf.frontierdefault because it does not include comments, and when using it with Frontier be sure to follow the recommendations in MyOwnSquid. Leave /etc/squid/squid.conf writable by the owner because that will prevent /etc/init.d/frontier-squid from overwriting it if the SQUID_CUSTOMIZE variable is ever accidentally left unset.

**SELinux**

As far as we are aware, the current version of frontier-squid works with SELinux enabled. Please let us know if you find that to not be the case.

Responsible: DaveDykstra