**HTTP** caches help reduce perceived lag, network utilization, and may improve performance of web applications.

Caches can also be used as a filtering proxy, restricting access to certain sites or resources.

Caches come in two flavors, forward and reverse.

A forward cache is used to speed up the **HTTP** access in a network. When multiple browsers request the same content from the same cache, the content may be returned from the cache instead of being requested from the original server. Forward caches include:

- **Squid.**
- **Tinyproxy.**
- **Apache.**

A reverse cache will speed up the perceived lag from an **HTTP** application server to any client. Reverse caches include:

- **Squid.**
- **Nginx.**
- **Tinyproxy.**

12.4. Proxy SSL

When a browser fetches an **`https://`** URI, one of the following two things happen:

- A **`CONNECT`** method request is made to the proxy server, and traffic is transparently forwarded to the destination. The proxy has no ability to filter on **URI**, path, query string, or other information.
- The browser directly connects to the **HTTPS** server, bypassing the proxy.

Because the **`CONNECT`** method passes any **TCP** traffic, you should enable this option with care.

It is possible to decrypt and monitor the **SSL** traffic using **Squid**, however there are legal, ethical, and security concerns in doing so.

12.5. Cache Hierarchy

Cache hierarchies are further extending the caching idea. Groups of cache servers working in concert can increase the caching efficiency, route the traffic to the best link, and support a higher number of clients.

There are two types of cache server hierarchy setups, which can be intermixed:

- **Peer to Peer:**
  Cache servers ask all or some of their peers if they have already cached content; if not, the cache server requests the content itself.
- **Parent/Child:**
  Cache servers ask a "parent" server if it has content; the parent will then fetch the content on the behalf of the "child".

The main **squid** configuration file, **`squid.conf`**, can be found in the following locations:

- On **CentOS**: **`/etc/squid/squid.conf`**.
- On **Ubuntu**: **`/etc/squid3/squid.conf`**.
- **On OpenSUSE:** **`/etc/squid/squid.conf`**

Commonly configured options include:

- **`http_port`**: Port to listen on for incoming proxy requests.
- **`http_access`**: Allow or deny access to certain **HTTP** requests.
- **`hierarchy_stoplist`**: Set of strings which disable the cache hierarchy settings.

**Squid** can also parse and check its syntax with a built-in syntax checker:

```
# squid -k parse
```

The **`-k`** switch takes the following options as well:

- **`reconfigure`**: Reload the configuration file.

**Squid** can also parse and check its syntax with a built-in syntax checker:

```
# squid -k parse
```

The `-k` switch takes the following options as well:

- **`reconfigure`**: Reload the configuration file.
- **`rotate`**: Rotate and re-open the log files.
- **`shutdown`**: Safe shutdown.
- **`interrupt`**: Unclean shutdown.
- **`kill`**: Hard unclean shutdown.
- **`debug`**: Enable debug in log files.

**A**ccess **C**ontrol **L**ists (**ACL**s) control most of the functions of the **Squid** daemon:

```
acl <ACLNAME> <ACLTYPE> arguments...
```

By quoting the argument, **ACL**s can also include separate files with one argument per line:

```
acl <ACLNAME> <ACLTYPE> "<FILENAME_TO_INCLUDE>"
```

To see possible **ACL** types, visit http://wiki.squid-cache.org/SquidFaq/SquidAcl#ACL_elements .

To enable a parent cache server, use the following configuration option:

```
cache_peer parent.example.com parent 3128 3130
```

To enable a sibling "peer" cache server, use the following configuration option:

```
cache_peer childcache.example.com sibling 3128 3130
```

Access to the the peer cache can be controlled with the following option:

```
cache_peer_access <PEER_NAME> <allow|deny> <ACLNAME>
```

For additional details, you can visit http://www.squid-cache.org/Doc/config/ .

Access control is the main reason to use a proxy. The **ACL** system of **Squid** has options to control almost every aspect of an **HTTP** request. Access control can restrict by time of day, by domain/URI, by user (logging into proxy), and by content.

To enable the **ACL** named **hourlyworkers** to only use the proxy during business hours, do:

```
acl workinghours time MTWHF 08:00-18:00
http_access allow hourlyworkers workinghours
http_access deny hourlyworkers
```

To restrict by a part of the URI, do:

```
acl banned_reddit url_regex ^http://.*reddit.com/.*$
http_access deny banned_reddit
```

To allow only authenticated users to use the following configuration, do:

```
acl valid_users proxy_auth REQUIRED
http_access allow valid_users
http_access deny all
```

When building **ACL**s or configuration files for **Squid**, remember that the first match wins. Therefore, start your **ACL**s with

the most specific options in the beginning.