

Red Hat Content Accelerator 2.2

Reference Manual

 Red Hat, Inc.

Red Hat Content Accelerator 2.2: Reference Manual

Copyright © 2001, 2002 by Red Hat, Inc.



Red Hat, Inc.

1801 Varsity Drive
Raleigh NC 27606-2072
USA
Phone: +1 919 754 3700
Phone: 888 733 4281
Fax: +1 919 754 3701
PO Box 13588
Research Triangle Park NC
27709 USA

TUX(EN)-2.2-RHI (2002-05-17T17:56-0400)

Red Hat is a registered trademark and the Red Hat Shadow Man logo, RPM, the RPM logo, and Glint are trademarks of Red Hat, Inc.

Linux is a registered trademark of Linus Torvalds.

All other trademarks and copyrights referred to are the property of their respective owners.

Copyright © 2001, 2002 by Red Hat, Inc. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation. A copy of the license is available at <http://www.gnu.org/copyleft/fdl.html>.

Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.

Distribution of the work or derivative of the work in any standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

Table of Contents

| | |
|--|-----------|
| 1. What is Red Hat Content Accelerator? | 5 |
| 1.1. New Red Hat Content Accelerator 2.2 Features | 6 |
| 1.2. Summary of System Requirements | 6 |
| 1.2.1. Current Limitations | 6 |
| 2. Installation | 7 |
| 2.1. Installation Instructions | 7 |
| 2.1.1. Log Files | 7 |
| 3. Configuration | 9 |
| 3.1. Modes of Operation | 9 |
| 3.2. Compressed Gzip Data Stream | 9 |
| 3.3. Parameters | 10 |
| 3.3.1. /proc/sys/net/tux Parameters | 10 |
| 3.3.2. Init Script Parameters | 14 |
| 3.3.3. /proc/net/tux Parameters | 15 |
| 3.3.4. Required Parameters | 15 |
| 3.4. Starting Red Hat Content Accelerator | 16 |
| 3.4.1. IRQ Affinity | 16 |
| 3.5. Stopping Red Hat Content Accelerator | 17 |
| 3.6. Debugging Red Hat Content Accelerator | 17 |
| 3.7. MIME Types | 17 |
| 3.8. HTTP Cache Control | 17 |
| 3.9. Mass Virtual Hosting | 18 |
| 3.9.1. virtual_server | 18 |
| 3.9.2. mass_hosting_hash | 19 |
| 3.9.3. string_host_tail | 19 |
| 3.10. Red Hat Content Accelerator as an FTP Server | 20 |
| 3.10.1. Security Features | 20 |
| 4. Security | 23 |
| 5. User-space Loadable Modules | 25 |
| Index | 27 |

What is Red Hat Content Accelerator?

Red Hat Content Accelerator is a kernel-based Web server licensed under the GNU General Public License (GPL).

It is currently limited to serving static webpages and coordinating with kernel-space modules, user-space modules, and regular user-space Web server daemons to provide dynamic content. Regular user-space Web servers do not need to be altered in any way for Red Hat Content Accelerator to coordinate with them. However, user-space code has to use a new interface based on the `tux(2)` system call.

Although dynamic content is becoming increasingly popular, there is still a need to serve static content. For example, nearly all images are static. Red Hat Content Accelerator can serve static content very efficiently from within the Linux kernel. A similar operation is already performed by the Network File System (NFS) daemon that runs in the kernel.

Red Hat Content Accelerator also has the ability to cache dynamic content. Red Hat Content Accelerator modules (which can be build in kernel space or in user space; user space is recommended) can create "objects" which are stored using the page cache. To respond to a request for dynamic data, a Red Hat Content Accelerator module can send a mix of dynamically-generated data and cached pre-generated objects, taking maximal advantage of Red Hat Content Accelerator's zero-copy architecture.

This new architecture for serving dynamic content requires a new API. The current API's for CGI can not be sufficiently mapped to Red Hat Content Accelerator's API. Thus, existing CGI applications must be converted before Red Hat Content Accelerator will process them. If the CGI application does not require the increased speed of Red Hat Content Accelerator, Red Hat Content Accelerator can process it by running the CGI application normally. This is done through Red Hat Content Accelerator's CGI module. Red Hat Content Accelerator can also handle a complex request (CGI or otherwise) by redirecting it to another Web server daemon such as Apache. In other words, static content, Red Hat Content Accelerator modules, old-style CGI applications, and programs specifically written for other webservers can be run on the same system with Red Hat Content Accelerator as the main web server.

In summary, the differences between Red Hat Content Accelerator and other webservers as well as the benefits of using Red Hat Content Accelerator include:

- Red Hat Content Accelerator runs partly within a custom version of kernel 2.4.x or higher and partly as a user-space daemon.
- With a capable network card, Red Hat Content Accelerator enables direct scatter-gather DMA from the page cache directly to the network, thus avoiding data copies.
- Whenever Red Hat Content Accelerator is unsure how to process a request or receives a request it is unable to handle, it always redirects the request to the user-space web server daemon to handle it in an RFC-compliant manner. An example of this user-space web server daemon is Apache.



Note

Apache is used throughout this document as the user-space web server daemon for readability.

For questions or comments about Red Hat Content Accelerator or this documentation, join the `<tux-list@redhat.com>` mailing list. For instructions on joining the mailing list, see <http://www.redhat.com/mailling-lists/>.

Also visit the Red Hat Content Accelerator Support page available at:

<http://www.redhat.com/services/techsupport/tux/>.

1.1. New Red Hat Content Accelerator 2.2 Features

The Red Hat Content Accelerator 2.2 release is an incremental upgrade to Red Hat Content Accelerator 1.0 and keeps source-code level compatibility with user-space modules.

The incremental enhancements include

- True zero-copy disk reads — Whereas Red Hat Content Accelerator 1.0 copied files into a temporary buffer, Red Hat Content Accelerator 2.2 is integrated with the page cache and thus uses zero-copy block IO.
- Generic zero-copy network writes — Red Hat Content Accelerator 2.2 uses the generic zero-copy TCP framework.
- Zero-copy parsing — Where possible, Red Hat Content Accelerator parses input packets directly. Even in RAM-limited situations, Red Hat Content Accelerator now does full, back-to-back zero-copy I/O.

Other changes include

- Enhanced user-space utilities and module support.
- Mass virtual hosting support — The host-based virtual server patch has been added to Red Hat Content Accelerator. There is no limit on the number of virtual hosts supported, only RAM and disk space.
- CGIs can be bound to particular CPUs or can be left unbound.
- A number of bugs were fixed which caused performance problems — Red Hat Content Accelerator 2.2 is now significantly faster than Red Hat Content Accelerator 1.0!

1.2. Summary of System Requirements

- Red Hat Content Accelerator Customized 2.4.x-based version of the kernel or higher
- x86, Alpha, IA64 or PowerPC/64 platform (should work on PowerPC/32, untested on Sparc)
- Alternate Web server such as Apache running on the same server to process unknown requests

1.2.1. Current Limitations

- Red Hat Content Accelerator can only call the other Web server such as Apache on the same server. In future revisions, it will allow the rollover of unsupported content to an alternate server.

Chapter 2.

Installation

This chapter describes how to install Red Hat Content Accelerator.

2.1. Installation Instructions

1. For optimal performance, create a separate RAID partition as the document root for Red Hat Content Accelerator.
2. Configure and install the kernel with Red Hat Content Accelerator support built-in, if it has not already been provided with Red Hat Content Accelerator configured.
3. Install the Red Hat Content Accelerator package with the command `rpm -Uvh tux-2.1.0-2.i386.rpm` (modify as necessary for new versions...)
4. Create an `index.html` file in `/var/www/html`, the default document root directory.
5. Start Red Hat Content Accelerator with the command `service tux start` (or `./tux.init start` on Linux systems not running Red Hat Linux), and test the URL `http://localhost/` with `lynx` or any Web browser.

The latest Red Hat Content Accelerator releases can be downloaded from <http://people.redhat.com/~mingo/TUX-patches/>. To install a TUX patch use the following instructions:

1. Download the latest Red Hat Content Accelerator patch and userspace utilities from <http://people.redhat.com/~mingo/TUX-patches/>.
2. Apply the Red Hat Content Accelerator patch to a vanilla 2.4.2 kernel tree with the command `patch -p0 < tux2-full-2.4.2-X6` (where `tux2-full-2.4.2-X6` is the Red Hat Content Accelerator kernel patch).
3. Use `make oldconfig` to enable Red Hat Content Accelerator in the kernel config, compile it, and boot into the Red Hat Content Accelerator kernel.
4. Compile and install the userspace utilities, where `tux-2.0.25` is the version of Red Hat Content Accelerator you want to install:

```
tar xzvf tux-2.0.25.tar.gz
cd tux-2.0.25
make
make install
```

5. Create an `index.html` file in `/var/www/html`, the default document root directory.
6. Start Red Hat Content Accelerator with the command `service tux start` (or `./tux.init start` on Linux systems not running Red Hat Linux), and test the URL `http://localhost/` with `lynx` or any Web browser.

2.1.1. Log Files

For each request, Red Hat Content Accelerator logs the address of the requestor, a date and time stamp accurate to at least one second, specification of the file requested, size of the file transferred, and the final status of the request.

The log files for Red Hat Content Accelerator are stored in `/var/log/tux` in binary format. In this binary format, the log files are approximately 50% smaller than standard ASCII text log files.

To view log files use the command `/usr/sbin/tux2w3c /var/log/tux`. The `tux2w3c` program converts the binary log files into into standard W3C-conforming HTTPD log files. If you want to save the ASCII output, you can redirect the output to a file: `/usr/sbin/tux2w3c /var/log/tux > tux.log`, where `tux.log` is the name of the output file.

Sample log file output:

```
195.4.12.3 - - Fri Nov 9 01:05:56 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov 9 01:06:10 2001 "GET / HTTP/1.1" - 2890 200
255.255.255.255 - - Fri Nov 9 01:06:10 2001 "GET /icons/apache_pb.gif HTTP/1.1" -
0 404
195.4.12.3 - - Fri Nov 9 01:06:10 2001 "GET /poweredby.png HTTP/1.1" - 1154 200
195.4.12.3 - - Fri Nov 9 01:06:04 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov 9 01:06:22 2001 "GET /manual/index.html HTTP/1.1" - 5557 200
195.4.12.3 - - Fri Nov 9 01:06:04 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov 9 01:06:22 2001 "GET /manual/images/apache_header.gif HTTP/1.1" -
4084 200
195.4.12.3 - - Fri Nov 9 01:06:04 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov 9 01:06:22 2001 "GET /manual/images/pixel.gif HTTP/1.1" -
61 200
195.4.12.3 - - Fri Nov 9 01:06:04 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov 9 01:06:26 2001 "GET /manual/invoking.html HTTP/1.1" -
1 200
195.4.12.3 - - Fri Nov 9 01:06:04 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov 9 01:06:35 2001 "GET /manual/stopping.html HTTP/1.1" -
1 200
195.4.12.3 - - Fri Nov 9 01:06:04 2001 "GET /test.html HTTP/1.1" - 53 200
195.4.12.3 - - Fri Nov 9 01:06:37 2001 "GET /manual/howto/ssi.html HTTP/1.1" -
18523 200
195.4.12.3 - - Fri Nov 9 01:06:41 2001 "GET /manual/new_features_1_3.html HTTP/1.1" -
34531 200
```


Chapter 3.

Configuration

This chapter describes how to configure the Red Hat Content Accelerator.

3.1. Modes of Operation

The recommended mode of operation is to have Red Hat Content Accelerator running as the main Web server and Apache run as the assistant.

- Client Port: 8080 (or other)
- Web Server Port: 80

For the recommend mode where Red Hat Content Accelerator is the main web server, the configuration for the user-space daemon must be changed to use port 8080. For Apache configuration, the changes are made in the configuration file `/etc/httpd/conf/httpd.conf` by changing the line

```
Port 80
```

to

```
Port 8080
```

For security reasons, the line

```
BindAddress *
```

should be changed to

```
BindAddress 127.0.0.1
```

This will prevent outside users from accessing Apache directly. You must restart Apache for the changes to take effect with the command `/etc/rc.d/init.d/httpd restart`.

The alternate mode of operation is to have the user-space daemon such as Apache as the main web server and Red Hat Content Accelerator as the assistant.

- Client Port: 80
- Web Server Port: 8080 (or other)

3.2. Compressed Gzip Data Stream

Red Hat Content Accelerator is now able to send compressed (gzip) data. This has the potential to decrease the amount of data the Web server sends to the client browser and decrease the browser's load time.

By default, this data compression is disabled. To enable it, add the following line to `/etc/sysctl.tux`:

```
net.tux.compression=1
```

The Gzip file with the extension `.gz` must be in the same directory as the uncompressed versions of the pages you wish to serve. All of the following conditions must be true for Red Hat Content Accelerator to send the `.gz` file. Otherwise, the original file(s) are sent.

- The Red Hat Content Accelerator compression feature is on in `/etc/sysctl.tux`.
- The client has explicitly stated to support gzip encoding.
- The original file exists, is a regular file, and has the proper permissions.
- The `.gz` file exists, is a regular file, and has the proper permissions.
- The `.gz` file is newer than or has the same-date as the original file.
- The size of the `.gz` file is smaller than original file.

**Note**

A cron job can be created to generate a new gzip file from the latest uncompressed data in each directory.

3.3. Parameters

This section describes how to configure Red Hat Content Accelerator via the available TUX parameters.

**Warning**

Most parameters can only be set when Red Hat Content Accelerator is not active.

**Note**

`CGI_UID` and `CGI_GID` are no longer available. All Red Hat Content Accelerator threads are executed with `DAEMON_UID` and `DAEMON_GID`.

3.3.1. `/proc/sys/net/tux` Parameters

The following parameters are set through `/proc/sys/net/tux`. Note this has changed from the original location of `/proc/sys/net/http` and `/proc/net/http`.

Table 3-1. Red Hat Content Accelerator Configuration Parameters

| Name | Default | Description |
|------------|---------|---|
| serverport | 80 | No longer available. To change the Red Hat Content Accelerator HTTP server port, use the command <code>echo 'http://0.0.0.0:80' > /proc/net/tux/0/listen/0</code> , where 80 is the port number. |
| clientport | 8080 | The port listened to by the userspace http-daemon |

| Name | Default | Description |
|-------------------|-------------------------|--|
| documentroot | /var/www/html | The directory where the web pages are stored. If using the init script <code>/etc/rc.d/init.d/tux</code> , <code>documentroot</code> should be set in <code>/etc/sysconfig/tux</code> as <code>DOCROOT</code> . |
| http_subdocroot | No value set by default | The directory, relative to the <code>documentroot</code> , where the web pages are stored. Red Hat Content Accelerator defaults to using <code>documentroot</code> if <code>http_subdocroot</code> has no value. |
| ftp_subdocroot | No value set by default | The directory, relative to the <code>documentroot</code> , where the files to be served by the FTP server are stored. Red Hat Content Accelerator defaults to using the <code>documentroot</code> defined for the HTTP server if <code>ftp_subdocroot</code> has no value. |
| ftp_log_retr_only | 0 | If set to 0, Red Hat Content Accelerator will log every other command as well. If set to 1, Red Hat Content Accelerator will only log RETR FTP commands to cut down the log size. |
| ftp_wait_close | 1 | If set to 1, Red Hat Content Accelerator will wait for data socket to close before sending completion message to command socket. Certain clients (for example, lynx) get confused by Red Hat Content Accelerator's high level of asynchronicity. This setting slows down FTP RETR downloads and directory listings and increases packet count, but it works around broken FTP clients. If set to 0, Red Hat Content Accelerator will not wait for the FTP client to notice the closed data socket. |
| ftp_login_message | 0 | If set to 1, the number of users that are logged in and the bandwidth currently served by Red Hat Content Accelerator is shown after logging in to the FTP server. If set to 0, this information is not shown. |
| 404_page | 404.html | If Red Hat Content Accelerator does not manage to look up a requested page then it first tries to look up the document specified in <code>404_page</code> . If the 404 page can not be found, the canned 404 message is sent. The file is relative to the <code>documentroot</code> . |

| Name | Default | Description |
|-----------------|--|--|
| threads | The number or server-threads, set at most to 1 per CPU | The number of kernel threads (and associated daemon threads) to be used. Can not be greater than the number of CPUs on the system. If using the init script <code>/etc/rc.d/init.d/tux</code> , threads should be set in <code>/etc/sysconfig/tux</code> as <code>TUXTHREADS</code> . |
| mode_allowed | S_IROTH | Required permissions for files Red Hat Content Accelerator will process. See "man 2 stat" for all values. |
| mode_forbidden | dir+sticky+execute | Files with this permission-mask are "forbidden" and will not be processed by Red Hat Content Accelerator. See "man 2 stat" for all values. |
| nonagle | 2 | If set to 0, standard Nagle output packet merging. If set to 1, no Nagle merging of output packets. If set to 2, TCP_CORK-style output packet merging. |
| push_all | 0 | If set to 0, may merge subsequent packets. If set to 1, force a packet boundary right after the end of the Red Hat Content Accelerator request. |
| compression | 0 | If set to 0, it is disabled. If set to 1, sending gzip compressed data is turned on. See Section 3.2 for details. |
| cgiroot | <code>/var/www/tux/cgiroot/</code> | The directory in which Red Hat Content Accelerator runs CGI programs. Set by default to <code>\$DOCROOT</code> in the tux init script. |
| cgi_cpu_mask | 0xffffffff | The default value allows CGI scripts to execute on all CPUs. This value can be set to bind newly started CGI scripts to a single CPU or a set of CPUs. The CPUs are represented in a 32-bit bitmask, where bit 1 is CPU#0, bit 2 is CPU#1, etc. This value has not effect on single-processor systems. |
| cgi_inherit_cpu | 0 | If set to 1, all newly started CGI scripts inherit the CPU-binding of the CGI-starting Red Hat Content Accelerator thread — all processes started by the CGI script will be bound to the same CPU as the parent CGI. |
| max_connect | 1000 | Maximum number of concurrent connections. |
| max_header_len | 3000 | Maximum header size in bytes. |

| Name | Default | Description |
|----------------------|--------------|--|
| max_output_bandwidth | 0 | Maximum output bandwidth (per connection) used up by keepalive requests in bytes/sec. The default value of 0 means off or unlimited bandwidth. Can be as low as 1 byte/sec. This parameter replaces max_keepalive_bw. |
| max_keepalive | 1000 | Maximum number of open keepalive connections. After having reached max_keepalives connections, Red Hat Content Accelerator zaps old connections based on LRU. |
| keepalive_timeout | 0 | Unfinished and should not be used. A per-client-connection timer that will time out if a request does not arrive within a pre-specified time. Timeout value is set in seconds. |
| max_object_size | 100MB | Maximum file size Red Hat Content Accelerator is willing to serve specified in bytes. |
| Dprintk | 0 | If TUX_DEBUG is turned on, then print out very verbose messages to syslog. Should only be used for debugging purposes. |
| ack_pingpong | 1 | Delay TCP ACK for incoming frames in the hopes of a subsequent output frame. Separate ACK will happen nevertheless, if no output frame is generated within a timeout. |
| all_userspace | 0 | If set to 1, every complete and valid HTTP request will be bounced to the first user-space module. The user-space module "takes control" over the entire URL space. Then, the user-space module can make a decision to 1) serve a static reply, 2) serve a cached dynamic reply, or 3) create a dynamic reply. If set to 0, all_userspace is disabled. |
| application_protocol | 0 | If set to 1, it enables the Red Hat Content Accelerator FTP server. If set to 0, this feature is disabled. Refer to Section 3.10 for details. |
| logentry_align_order | N/A | Currently unused. |
| logfile | /var/log/tux | The filename of the Red Hat Content Accelerator binary logfile. Refer to Section 2.1.1 for more information. |
| logging | 0 | If set to 1, logging is enabled. If set to 0, logging is disabled. |

| Name | Default | Description |
|------------------------|--|---|
| redirect_logging | 1 | Set to 0 to suppress redirected connections. Can be changed at runtime and takes effect immediately. |
| referer_logging | 0 | If set to 1, referer logging is enabled and will be automatically printed by <code>tux2w3c</code> if the referer entry is present. If set to 0, referer logging is disabled. |
| max_backlog | 2048 | Maximum size of SYN backlog of the Red Hat Content Accelerator listening socket. |
| virtual_server | 0 (off) | Turns on mass virtual hosting. Hosts are headers from the browser that are directly turned into <code>\$DOCROOT/<Host> 'virtual docroots.'</code> This way any number of hosts can be served by a single Red Hat Content Accelerator server without any performance penalty at all. Refer to Section 3.9 for details. |
| mass_hosting_hash | 0 (off) | If <code>virtual_server</code> is enabled, this parameter modifies the hostname mapping to be more effective for a large number of hosts. Refer to Section 3.9 for details. |
| strip_host_tail | 0 (off) | If <code>virtual_server</code> is enabled, this parameter strips off hostname components. Refer to Section 3.9 for details. |
| zerocopy_parse | 1 | Use the input packet buffer as a temporary buffer and avoids copying input data. |
| defer_accept | 0 (disabled if <code>keepalive_timeout</code> or <code>max_keepalives</code> is set) | If set to 1, then Red Hat Content Accelerator processes will not be woken up on the initial SYN-ACK event of a new TCP connection, but only after the first real data packet has arrived. If set to 0, this feature is disabled. |
| http_dir_indexing | 0 (disabled) | If set to 1, Red Hat Content Accelerator will list files in readable directories if an index file does not exist. |
| generate_cache_control | 1 (enabled) | If set to 1, generate HTTP-expiration headers according to <code>/etc/tux.mime.types</code> . Refer to Section 3.8 for details. |
| generate_etags | 1 (enabled) | If set to 1, generate the Etag HTTP header. Refer to Section 3.8 for details. |
| generate_last_mod | 1 (enabled) | If set to 1, generate the Last-Modified HTTP header. Refer to Section 3.8 for details. |

3.3.2. Init Script Parameters

If the Red Hat Content Accelerator init script `/etc/rc.d/init.d/tux` is used, the following parameters can be set in the file `/etc/sysconfig/tux` (see Table 3-2). They should *not* be set in `/etc/sysctl.conf` because the init script will override parameters set in `/etc/sysctl.conf`. Using the init script is the preferred method for starting Red Hat Content Accelerator. For parameters beyond those supported by `/etc/sysconfig/tux`, you can use `/etc/sysctl.tux`, which is read after the Red Hat Content Accelerator module is loaded; `/etc/sysctl.conf` is read before the Red Hat Content Accelerator module is loaded and so Red Hat Content Accelerator-specific settings in `/etc/sysctl.conf` do not take effect.

Table 3-2. `/etc/sysconfig/tux` parameters

| Parameter | Default | Description |
|-----------------------|--|--|
| TUXTHREADS | The number of server-threads, set at most to 1 per CPU | The number of kernel threads (and associated daemon threads) to be used, cannot be greater than the number of CPUs on the system |
| DOCROOT | <code>/var/www/html</code> | The document root, the directory where the web pages are stored. |
| DAEMON_UID | nobody | UID (user) as which the daemon runs. |
| DAEMON_GID | nobody | GID (group) as which the daemon runs. |
| CGIROOT | <code>/var/www/html</code> | The directory where the CGI programs are stored. CGI programs can be started in the chroot environment by default. Set <code>CGIROOT=/</code> if you want CGI programs to have access to the whole system. |
| MAX_KEEPALIVE_TIMEOUT | 30 | Timeout value for each HTTP connection. Use this to prevent connection hangs. |
| TUXMODULES | demo.tux demo2.tux demo3.tux demo4.tux | list of user-space loadable Red Hat Content Accelerator modules, see <code>man 2 tux</code> for more information |
| MODULEPATH | / | Path to the user-space loadable Red Hat Content Accelerator modules |

3.3.3. `/proc/net/tux` Parameters

After starting Red Hat Content Accelerator, the `/proc/net/tux` directory contains the file `stat`. This file contains statistics on every allocated request structure. As this works even if `TUX_DEBUG` is turned off, this should help debugging things a bit more. It can also be used to calculate file download status. For example, `TUX/FTP - the 100*f_pos/filelen` gives the current progress of download.

It is possible to bind the logger thread to any particular CPU (or group of CPUs), so you can localize IO, via `/proc/net/tux/log_cpu_mask` The default is to run on any CPU.

3.3.4. Required Parameters

Before starting Red Hat Content Accelerator, the following parameters must be set:

- `serverport`

- clientport
- DOCROOT



The DOCROOT for Red Hat Content Accelerator must be the same document root directory as Apache or other user-space daemon running as the assistant web server for Red Hat Content Accelerator to properly redirect requests.

3.4. Starting Red Hat Content Accelerator

Red Hat Content Accelerator can be started by issuing the command:

```
/etc/rc.d/init.d/tux start
```

This script is written to start Red Hat Content Accelerator on a single-processor as well as a multi-processor server.

If you choose to write your own script to start Red Hat Content Accelerator or start it from the `/usr/sbin/tux` binary, you can use the following options:

Table 3-3. /usr/sbin/tux options

| Option | Description |
|-------------------------------------|--|
| -t, --threads= <i>N</i> | number of tux threads |
| -d, --docroot= <i>path</i> | directory path for document root |
| -m, --modpath= <i>path</i> | directory path for user-space loadable Red Hat Content Accelerator modules |
| -d, --daemon | run in the background as a daemon |
| -D, --date-interval= <i>seconds</i> | how often (in seconds) to update the date string, the default is 1 second |
| -.?, --help | show help message |
| --usage | display brief usage message |



IRQ affinity is a small performance boost. If you are not experiencing any performance difficulties, it is not recommended you try the following.

3.4.1. IRQ Affinity

Binding IRQ's to a group of CPU's is a new feature of the 2.4 kernel. While it was originally developed as part of Red Hat Content Accelerator, it is now a generic and independent kernel feature. Every IRQ source in Linux has an entry in `/proc/irq` directory. For example, the settings for IRQ 40 is stored in `/proc/irq/40`. IRQ affinity, or IRQ bindings, is configured though the `smp_affinity` setting in that directory. For example, the `smp_affinity` for IRQ 40 is in `/proc/irq/40/smp_affinity`. The value of the `smp_affinity` setting is a bitmask of all CPU's that are permitted as a resource for the given IRQ. The default value for `smp_affinity` is the HEX value `0xffffffff`. This means the processes for the IRQ are sent to all CPU's. You are not allowed to turn off all CPU's for an IRQ. If the IRQ controller

does not support IRQ affinity, the value can not be changed from the default. If multiple CPU's are defined, then the IRQ source uses the least busy CPU. This is called 'lowest priority APIC routing.' IRQ affinity is achieved by binding an IRQ to a specific CPU or group of CPU's by echoing a HEX value to `smp_affinity` for the IRQ.

Thus, Red Hat Content Accelerator thread N is bound to CPU N. If a single Red Hat Content Accelerator thread is used (which is recommended) and there is only one network interface card, then the network interface card's IRQ should be bound to CPU0.

3.5. Stopping Red Hat Content Accelerator

If Red Hat Content Accelerator was started with the `/etc/rc.d/init.d/tux start` script, stop Red Hat Content Accelerator by executing the `/etc/rc.d/init.d/tux stop` script. This will unload all user-space Red Hat Content Accelerator modules automatically.

If you did not use the scripts provided, stop Red Hat Content Accelerator with the command `/usr/sbin/tux -s` or `/usr/sbin/tux --stop`.

3.6. Debugging Red Hat Content Accelerator

To print out the state and various other information about Red Hat Content Accelerator, execute the `gettuxconfig` script. You must be root to run this script.

The `checkbindings` shell script checks an existing Red Hat Content Accelerator SMP configuration, whether all IRQ, interface, and listening socket bindings and affinities are set up correctly. It assumes that the interfaces `eth0`, `eth1`, `eth2`, and so on are used linearly and mapped linearly. The script warns if it finds any inefficiency.

3.7. MIME Types

Red Hat Content Accelerator supports three types of MIME types starting with version 2.0.13 and kernel patch 2.4.2-P3. They are defined in `/etc/tux.mime.types`.

Table 3-4. MIME Types

| MIME Type | File Extension | Description |
|--------------|---------------------|---|
| TUX/redirect | <code>pl php</code> | All extensions listed after TUX/redirect will be redirected to the secondary server. |
| TUX/CGI | <code>cgi pl</code> | All extensions listed after TUX/CGI will be handled by the Red Hat Content Accelerator CGI engine directly. |
| TUX/module | <code>tux x</code> | All extensions listed after TUX/module will be handled by Red Hat Content Accelerator userspace modules. |

The TUX/redirect MIME type will redirect all requests to files ending in `.pl` or `.php` to Apache, without having to check for file permissions.

The TUX/CGI MIME type specify scripts that should be located in the `$DOCTYPE/cgi-bin` directory, or the directory specified by the `cgiroot` parameter. Refer to Section 3.3 for details.

Refer to Chapter 5 for details about the TUX/module MIME type.

3.8. HTTP Cache Control

Starting with Red Hat Content Accelerator 2.2.1 and kernel patch 2.4.17-A1, Red Hat Content Accelerator supports the *Cache-Control: max-age=x* HTTP extension in addition to the basic cache-control properties such as the `ETag` and `Last-Modified` HTTP headers. While the basic properties allow the client browser to know whether the content in its cache is up-to-date while communicating with the server, this new feature allows browsers and proxies to cache the server's response for the specified time without the need to reconfirm with the server every time the content in its cache is accessed, reducing content load time for users, bandwidth and server resources usage.

Expiration time is set on a per file extension basis, in `/etc/tux.mime.types` discussed in previous paragraph. While a standard entry may look like below:

```
text/html    htm html
```

The following format should be used to assign expiration time to a file extension:

```
text/html    htm|1800 html|1800
```

This will set the expiration time for `.htm` and `.html` files to 1800 seconds (half an hour). If no expiration time is provided for a file extension, no `Cache-Control` directive will be issued while serving the file.

You may set a different expiration time for every extension, even if they belong to the same MIME type:

```
text/html    htm|1800 html|3600
```

or

```
text/html    htm|1800 html
```



Note

To disable this feature on a server-wide basis, set the `generate_cache_control` parameter in `/proc/sys/net/tux` to 0.

3.9. Mass Virtual Hosting

Red Hat Content Accelerator supports mass virtual hosting, which enables a very high number of virtual hosts.

There are three tunables that deal with virtual hosting:

- `virtual_server` — Valid values are 0, 1, 2, or 3.
- `mass_hosting_hash` — Valid values are 0, 1, 2, or 3.
- `strip_host_tail` — Value must be an integer.

These three tunables depend on each other, and the `strip_host_tail` tunable is only used if host based virtual serving is enabled. Otherwise, it is ignored.

3.9.1. virtual_server

If the value is set to 0, virtual hosting is disabled:

```
http://www.example.com/a.html => $DOCROOT/a.html
```

If the value is set to 1, host-based virtual hosting is enabled:

```
http://www.some.site.com/a.html => $DOCROOT/some.site.com/a.html
```



Note

Red Hat Content Accelerator strips off the `www.` prefix variants and transforms the hostname to lowercase.

If the value is set to 2, IP-based virtual hosting is enabled:

```
http://www.some.site.com/a.html => $DOCROOT/1.2.3.4/a.html
```

If the value is set to 3, a mixture of host-based and IP-based virtual hosting is enabled:

```
http://www.some.site.com/a.html => $DOCROOT/1.2.3.4/some.site.com/a.html
```

3.9.2. mass_hosting_hash

The `mass_hosting_hash` tunable modifies the hostname mapping to be more effective for a large number of hosts.

If the value is set to 0, `mass_hosting_hash` is disabled.

If the value is set to 1:

```
http://www.some.site.com/a.html => $DOCROOT/s/some.site.com/a.html
```

If the value is set to 2:

```
http://www.some.site.com/a.html => $DOCROOT/s/so/some.site.com/a.html
```

If the value is set to 3:

```
http://www.some.site.com/a.html => $DOCROOT/s/so/som/some.site.com/a.html
```

3.9.3. string_host_tail

The `strip_host_tail` tunable strips off hostname components, starting at the end of the hostname.

If the value is set to 0, this tunable is disabled.

If the value is set to 1:

```
http://www.some.site.com/a.html => $DOCROOT/some.site/a.html
```

If the value is set to 2:

```
http://www.some.site.com/a.html => $DOCROOT/site/a.html
```

and so on...

3.10. Red Hat Content Accelerator as an FTP Server

Starting with version 2.0.21 of Red Hat Content Accelerator and version 2.4.2-U7 of the Red Hat Content Accelerator patched kernel, Red Hat Content Accelerator can be configured to run as an anonymous FTP server.

To use Red Hat Content Accelerator as an HTTP and FTP server at the same time, use the following commands:

```
echo "http://0.0.0.0:80" > /proc/net/tux/0/listen/0
echo "ftp://0.0.0.0:21" > /proc/net/tux/0/listen/1
```

By default, the document root for the FTP server is the document root for the HTTP server set as `DOCROOT` in `/etc/sysconfig/tux` or the value of `/proc/sys/net/tux/documentroot`.

To configure different document roots for the HTTP and FTP server, set the `DOCROOT` in `/etc/sysconfig/tux` and execute the following commands:



Note

The `http_subdocroot` and `ftp_subdocroot` are relative to `DOCROOT`.

```
echo '/www/' > /proc/sys/net/tux/http_subdocroot
echo '/ftproot/' > /proc/sys/net/tux/ftp_subdocroot
```

Restart Red Hat Content Accelerator to apply the changes:

```
service tux restart
```

After executing these commands, the Red Hat Content Accelerator FTP server will be running on port 21.

To have it display directory listings, run the `generatetuxlist` script from the FTP docroot. This script creates the files `.TUX-LIST` and `.TUX-NLIST` files that cache the directory listing. Everytime the FTP docroot directory changes, the script must be re-run to generate an updated directory listing.



Note

The Red Hat Content Accelerator FTP server has been through numerous stresstests and FTP-client compatibility tests. However, it is still early software. It has no known bugs or security holes at the moment. It has not been tested with a wide number of FTP clients yet (only the most obvious ones).

3.10.1. Security Features

The following are security features of the Red Hat Content Accelerator FTP Server:

- Because Red Hat Content Accelerator does not start per-client processes, the memory allocation overhead for each FTP client logged in is less than 10 KB. This allows thousands of parallel connections.

- Paranoid parser and paranoid command-evaluation.
- Chroots to docroot.
- Never starts any external userspace process. All FTP functionality is done in a approximately 900 lines C module, in the kernel.
- Even in kernel mode the Red Hat Content Accelerator FTP Server drops all privileges and switches to uid and group `nobody`.
- Only the most trivial globbing (`mget *`) supported, and no recursion support.

Chapter 4.

Security

Red Hat Content Accelerator is designed to have very strict security. This is possible because the assistant user-space daemons is used to handle the complex exceptions.

Red Hat Content Accelerator only serves a file if

1. The URL does not contain ?.
2. The URL does not start with /.
3. The URL points to a file that exists.
4. The file is world-readable.¹
5. The file is not a directory.¹
6. The file is not executable.¹
7. The file does not have the sticky-bit set.¹
8. The URL does not contain any forbidden substrings such as . . .¹

1. Configurable through the sysctl parameters in `/proc/sys/net/tux`
1. Configurable through the sysctl parameters in `/proc/sys/net/tux`
1. Configurable through the sysctl parameters in `/proc/sys/net/tux`
1. Configurable through the sysctl parameters in `/proc/sys/net/tux`
1. Configurable through the sysctl parameters in `/proc/sys/net/tux`

Chapter 5.

User-space Loadable Modules

In addition to parts of Red Hat Content Accelerator running in kernel-space, user-loadable modules can also be written for Red Hat Content Accelerator.



Note

The API for the user-loadable modules is currently under development. This section of the documentation will be updated as the API becomes available.

User-space loadable modules are usually a single `.c` file and are compiled as shared libraries as a `.so` file. There can be an unlimited number of user-space HTTP modules, and they can be compiled in a language of choice. They have full address space protection, can not crash the kernel, and are unprivileged.

A list of user-space loadable Red Hat Content Accelerator modules and their location must be specified with the `TUXMODULES` parameter in `/etc/sysconfig/tux`. Refer to Section 3.3 for details.

Starting with Red Hat Content Accelerator version 2.0.13 and kernel patch 2.4.2-P3, user-space loadable modules do not require special permissions to be activated. Instead, the module is specified using a common MIME type definition file. The user-space modules must end with the file extension `.tux` or `.x` and specified with the `TUXMODULES` parameter in `/etc/sysconfig/tux`. The file must be owned by `root` with `root` as the group and must be world-readable. It does not have to be executable. For example, to use the `demo.tux` user-space loadable module, you might have the following file:

```
[root@m /]# ls -l /var/www/html/demo.tux
-rw-rw-r-- 1 root root 0 Sep 3 04:42 /var/www/html/demo.tux
```

If Red Hat Content Accelerator finds a URL object that has this MIME type, it searches the internal list of modules defined as `TUXMODULES` in `/etc/sysconfig/tux`. If there is a match, Red Hat Content Accelerator sends the request to the user-space loadable module.

For further information about writing a Red Hat Content Accelerator user-space loadable module, see the file `/usr/share/doc/tux-<version>/TUXAPI-user.txt`.

Index

Symbols

/etc/rc.d/init.d/tux start, 16
/etc/rc.d/init.d/tux stop, 17
/etc/sysconfig/tux, 14
/etc/sysctl.tux, 9
/proc/sys/net/tux, 10
/var/log/tux, 7

B

Benefits, 5

C

Client Port, 9
Compressed Data, 9
compression, 9
Configuration, 9
Current Limitations, 6

D

Debugging Red Hat Content Accelerator, 17
Dynamic Content, 5

F

FTP Server, 20
 Directory Listings, 20
 Document Root, 20

G

gettuxconfig, 17
Gzip, 9

I

init script parameters, 14
Installation, 7
Installation Instructions, 7
IRQ Affinity, 16

L

Limitations, 6
Log Files, 7

M

MIME Types, 17
Modes, 9
Modules
 User-loadable Modules, 25

P

Parameters, 10
 List of Parameters, 10
 Required Parameters, 15
Permissions, 23
Port, 9

R

Red Hat Content Accelerator 2.2
 enhancements, 6

S

Scripts, 16
Security, 23
Server Port, 9
Starting Red Hat Content Accelerator, 16
Static Content, 5
Stopping Red Hat Content Accelerator, 17
sysctl, 23
System Requirements, 6

T

tux.mime.types, 17
tux2w3c, 7

V

virtual hosting, 18

W

What is Red Hat Content Accelerator?, 5

