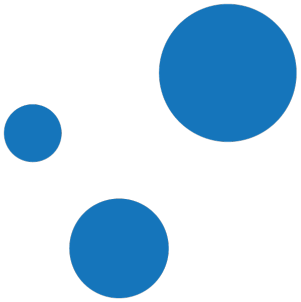

VARNISH SOFTWARE

Varnish Cache Plus Manual Documentation

Release version

Varnish Software

Jul 28, 2017

Contents

CHAPTER 1

Introduction

This is the administrator manual for Varnish Cache Plus.

It describes the extra features and functionalities that are in Varnish Cache Plus, or are possible with Varnish Cache Plus, that is not available in regular Varnish Cache.

The intended audience of this technical manual are systems administrators and managers looking into Varnish Plus for solving their caching needs.

It is assumed that the reader has a good understanding of existing Varnish Cache concepts, as they are explained here:

> https://book.varnish-software.com/4.0/

1.1 What is Varnish Cache Plus

Varnish Cache Plus is a special version of Varnish Cache made by Varnish Software for paying customers.

The main goal of Varnish Cache Plus is to give customers faster access to newly developed features that have yet to make it into the public Varnish Cache version, giving customers a competitive advantage over users of the free version.

Varnish Cache Plus should not be confused with Varnish Plus, a product offering by Varnish Software. Varnish Cache Plus is one of the software components available for Varnish Plus customers. For more information on this see our web page:

> https://www.varnish-software.com/products/varnish-plus

1.2 Versioning and release schedule

Varnish Cache Plus is versioned after the original Varnish Cache release it is based on, with an additional number indicating the patch level.

For example is the Varnish Cache Plus 4.1.2r1 release based on Varnish Cache 4.1.2, and it is the first release on top of that version.

The expected release cycle of a maintenance Varnish Cache Plus releases is 2-4 months after a VC release.

Immediate concerns like security updates are handled by our support function and will be handled outside the regular release schedule.

1.3 Varnish modules

Varnish Cache Plus contains a set of Varnish Modules (vmods) that extends the functionality of Varnish.

Examples of modules available:

- vmods from the varnish-modules package, including cookie handling, header modifications, saint-mode, tcp for connection throttling, variable support, request rate throttling, and advanced cache invalidation with softpurge and xkey. (previously called hashtwo)
- HTTP (curl) client interface, memcached client, IP geolocation, hash functions (digest) functionality.
- Runtime ACL creation, fast hash table support with timeouts, calendar and accounting functions for paywalls/metered access. (Varnish Plus only)

This list is not meant to be complete, but as a pointer to what is available. Other Varnish Plus components may package other vmods that are documented elsewhere.

Legacy modules available:

boltsort (3.0) Fast sorting of request arguments for applications where GET arguments are not ordered by default. Improves cache hit rate.

ipcast (3.0) Match a string against an ACL in VCL. Only needed in Varnish 3.0, builtin functionality in 4.0 and newer.

See the installation chapter on how these can be installed and used.

1.4 Package repository

All software related to Varnish Cache Plus and modules is available in Redhat and Ubuntu package repositories.

These repositories are available on https://repo.varnish-software.com/ using a customer specific username and password. For access credentials contact support.

CHAPTER 2

Installing

2.1 Installing Varnish Cache Plus

Varnish Cache Plus is distributed in prepackaged form for the most common Linux operating system distributions.

The list of supported distributions are:

- RedHat Enterprise Linux 6
- RedHat Enterprise Linux 7
- Ubuntu Linux 14.04 LTS (trusty)

Legacy versions have different sets of supported platforms.

Varnish Cache Plus is supported on 64bit systems.

Customers should contact support@varnish-software.com for access to the software.

2.2 Installing Varnish modules

In 4.1 many Varnish modules (vmods) are embedded in the `varnish-plus` package, and no extra installation is required.

Modules with third-party package dependencies (libcurl, libmemcached) are available in a separate package called `varnish-plus-vmods-extra`. This package is not installed by default.

In legacy versions VMODs are either available in repository packages called `vmod-NAME` or `libvmod-NAME` (VCP3.0), or more recently in the `varnish-plus-vmods` package. (VCP4.0)

CHAPTER 3

Upgrading

This section describes in general terms how to upgrade to newer versions of Varnish Cache Plus.

Many (most) of the changes needed will be identical to a Varnish Cache upgrade, so the open-source documentation is referenced below.

These instructions are not meant to be copied verbatim, but rather give an indication of the considerations and steps necessary when upgrading.

Support can help out with VCL configuration changes between versions.

3.1 Minor version upgrades

Upgrades between minor versions, for example from 4.0.3r4 to 4.0.3r5, are considered stable and will not require manual intervention.

3.2 Upgrade to Varnish Cache Plus 4.1

Upgrade from 4.0 to 4.1 are usually straight forward.

Before the upgrade the list of supported platforms should be reviewed.

3.2.1 Software update

On Enterprise Linux:

```
# mv /etc/yum.repos.d/varnish-4.0-plus.repo /etc/yum.repos.d/varnish-4.1-
↪plus.repo
# sed -i -e 's|4.0-plus|4.1-plus' /etc/yum.repos.d/varnish-4.1-plus.repo
# yum clean metadata
# yum update varnish-plus
```

On Ubuntu 14.04 LTS (trusty):

```
# mv /etc/apt/sources.list.d/varnish-4.0-plus.list /etc/apt/sources.list.d/
↪varnish-4.1-plus.list
# sed -i -e 's|4.0-plus|4.1-plus' /etc/apt/sources.list.d/varnish-4.1-plus.
↪list
```

```
# apt-get update
# apt-get upgrade varnish-plus
```

On all platforms the `varnish-plus-vmods` package has been retired. Modules are now embedded in the main `varnish-plus` package.

The `varnish-plus-vmods-extra` package is still available.

3.2.2 Configuration update

VCL code only requires minimal changes, see https://www.varnish-cache.org/docs/4.1/whats-new/upgrading.html .

3.3 Upgrade to Varnish Cache Plus 4.0

Between 3.0 and 4.0 there were significant changes to VCL, which made the transition more difficult than was expected.

Main upgrade documentation can be found online: https://www.varnish-cache.org/docs/4.0/whats-new/upgrading.html

The most common substitutions can be done automatically using the varnish3to4 script.

CHAPTER 4

Varnish Cache Plus features

4.1 Client SSL/TLS termination

4.1.1 Description

The SSL/TLS addon in Varnish Plus is a complete setup for doing SSL/TLS termination in front of Varnish Cache Plus.

Varnish Plus SSL/TLS addon consists of a supported helper process (called "hitch") that does SSL/TLS termination, and PROXY protocol support between the helper process and Varnish Cache Plus.

4.1.2 VCL example

VCL code:

```
import std;
sub vcl_recv {
    # on PROXY connections the server.ip is the IP the client connected to.
    # (typically the DNS-visible SSL/TLS virtual IP)
    std.log("Client connected to " + server.ip);
    if (std.port(server.ip) == 443) {
        # client.ip for PROXY requests is set to the real client IP, not␣
↪the
        # SSL/TLS terminating proxy.
        std.log("Real client connecting over SSL/TLS from " + client.ip);
    }
}
```

4.1.3 Installation

The SSL/TLS addon can be installed with:

```
$ yum install varnish-plus-addon-ssl        # redhat systems
or
$ apt-get install varnish-plus-addon-ssl    # ubuntu based systems
```

This will download and install the addon from the Varnish Plus repositories.

4.1.4 Configuration

Setting up the SSL/TLS addon is simple:

- Add `-a 127.0.0.1:6086,PROXY -a :80` to DAEMON_OPTS in */etc/varnish/varnish.params* (systemd) or */etc/(sysconfig|default)/varnish* (sysv)

  This will enable a separate listening port that accepts PROXY protocol connections. The normal HTTP listen port must therefore be added as well, in this case ":80" or (by default) ":6081". Restart Varnish to enable the new listening socket.

  Note: In VCP4.0 another format is used for listening sockets. Only a single `-a` is allowed and format is: `-a proxy@127.0.0.1:6086,:80`.

- Create a test key and self-signed certificate for testing (this may already have been done during package installation).

```
/etc/pki/tls/certs/make-dummy-cert /etc/hitch/testcert.pem
```

- Start and persist the SSL/TLS helper process.

  `systemctl enable hitch` (systemd), `update-rc.d hitch defaults` (trusty) or `chkconfig hitch on` (EL6). `service hitch start`

SSL/TLS support is now available on port 8443. This can be changed to 443 in `/etc/hitch/hitch.cfg`.

4.1.5 Availability

SSL/TLS addon was added to Varnish Plus starting from Varnish Cache Plus 4.0.3r2.

4.2 Backend SSL/TLS

4.2.1 Description

Varnish Cache Plus 4.0 and later has support for using SSL/TLS on backend connections.

This means that any miss, pass or piped requests handled by Varnish Plus will be encrypted when sent over the network to a backend/origin server.

Typical usage is a setup with caches distributed around the world to be closer to the end user, and it is important that internal session data is not available to any third parties.

4.2.2 VCL example

VCL code:

```
backend default {
    .host = "backend.example.com";
    .port = "https";          # This defaults to https when SSL
    .ssl = 1;                 # Turn on SSL support
    .ssl_sni = 1;             # Use SNI extension
    .ssl_verify_peer = 1;     # Verify the peer's certificate chain
}
```

4.2.3 Installation

Support for backend SSL/TLS is built into in supported versions of Varnish Cache Plus, and does not require any extra installation steps.

Backend SSL/TLS introduces a requirement for OpenSSL which is maintained and updated through the operating system. When using this functionality it is important to follow security best practices and keep the systems update to avoid loss of confidentiality.

4.2.4 Availability

Backend SSL/TLS support was added to Varnish Plus starting from Varnish Cache Plus 4.0.3r3, and is also available in Varnish Cache Plus 4.1 series.

4.3 Dynamic backend generation

4.3.1 Description

Dynamic backends in Varnish allows backends to be defined on request time instead of being predefined. Any request header can be used as the source for creating a backend definition.

A clear advantage of this is the possibility of DNS name based backends that move to scale up and down as load shifts. This is a great feature that fits well with autoscaling cloud deployments.

There are two different modules for this in Varnish Plus: vmod-goto and vmod-named. To keep this brief, the example is for vmod-goto and further information on vmod-named can be found in its man page.

4.3.2 VCL example

To send API calls to a cluster of machines answering to the "api.example.com" sub-domain, on port 8080:

```
import goto;

sub vcl_backend_fetch {
    if (bereq.url ~ "^/api/") {
        set bereq.backend = goto.backend("api.example.com", "8080");
    }
}
```

This will create a backend lasting for 10 seconds based on a DNS lookup for the address given.

4.3.3 Availability

Dynamic backend generation is available starting from Varnish Cache Plus 4.1.4r2.

4.3.4 Installation

The goto and named VMODs are prebuilt for supported versions and included inside the varnish-plus package.

The package contains further usage instructions, accessible using `man vmod_named` and `man vmod_goto`.

Contact support@varnish-software.com if you need assistance.

4.4 Massive Storage Engine v2.0 (4.1)

4.4.1 Description

Varnish Massive Storage Engine v2.0 (MSE2) is an improved storage backend for Varnish, replacing the traditional malloc and file storages.

MSE2 adds data persistence across Varnish and server restarts, while continuing to offer the performance improvements seen in MSE1 on Varnish Cache Plus 4.0.

MSE1 and MSE2 are designed and tested with storage sizes up to 10 TB, but expected to work on far larger workloads.

4.4.2 Availability

MSE2 is available in Varnish Cache Plus 4.1.2r1.

4.4.3 Installation

MSE2 is built into supported Varnish Cache Plus versions, and does not need additional installation steps.

4.4.4 Configuration

MSE2 introduces a two stage configuration, where the storage files on disk are created by *mkfs.mse* (embedded tool) before being used by Varnish Cache Plus.

Steps:

```
# mkdir -p /var/lib/mse/book /var/lib/mse/store
# mkfs.mse -b /var/lib/mse/book/,1G -s /var/lib/mse/store,100G
```

This will create a small 100 GB storage segment that Varnish Cache Plus can use, with a 1 GB bookkeeping journal for persistence.

Bookkeeping file should be on a file system backed by storage with fast random access speeds.

The output from *mkfs.mse* contains the `-s mse...` configuration string that should be passed to Varnish as a start up argument.

If a persistence is not required, *mkfs.mse* can be run without the `-b` argument to only create the store files.

MSE2 has a small set of tunable parameters that must be set when the storage files are made. It is recommended to keep most of these at the default values. For optimal performance `-p big_alloc=SIZE` (default 1 MB) can be tuned to approximately the size of the larger cached object.

See mkfs.mse(1) man page for more information.

4.4.5 Storage sizing

Sizing the storage should be done based on the following recommendations.

On setups with gigabyte range storages, bookkeeping file should be around 1% of the storage size.

Setups in the terabyte range should have a bookeeping file size around 0.5% of storage size.

Incorrect bookkeeping file sizing can be seen in `g_sparenode` in *varnishstat*. This should never run out, if this happens objects will be evicted to make room for any new ones. Having spare nodes will waste a comparatively small amount of disk space in the bookkeeping file and does not do any harm.

If running on a system with standalone disks (no raid controller), use separate `-smse` instances for each disk. If the disks are SSDs, the bookeeping file can be kept on the same disk.

File system should be EXT4 and the total usage of a single disk/filesystem should not go above 95% capacity.

MSE2 is a userspace filesystem and fully allocates the storage on creation. TRIM for SSD storages is not necessary.

4.5 Massive Storage Engine (4.0)

4.5.1 Description

Varnish Massive Storage Engine v1.0 (MSE or MSE1) is an improved storage backend for Varnish.

It is a replacement for the original malloc and file storage backends.

Main improvements are decreased disk IO load and lower storage fragmentation.

MSE is designed and tested with storage sizes up to 10 TB, but expected to work on far larger workloads.

4.5.2 Availability

MSE1 is available in Varnish Cache Plus 4.0 from 4.0.2r1 and newer. In Varnish Cache Plus 4.1 MSE1 has been replaced by MSE2.

4.5.3 Installation

MSE is built into supported Varnish Cache Plus versions, and does not need additional installation steps.

4.5.4 Configuration

MSE is configured in the same way that the file storage backend is:

```
# varnishd -s mse,<file>,<size>[,<segments>]
```

It expects a data file path on disk and what size it should be. Number of storage segments can be left out.

MSE introduces a set of new parameters:

```
mse_bigalloc                1M [bytes] (default)
mse_delay_writes            on [bool] (default)
mse_membuf_size             4 [pages] (default)
mse_minextfree              4k [bytes] (default)
mse_nuke_limit              10 (default)
mse_pad_writes              on [bool] (default)
mse_prune_factor            2 (default)
mse_prune_loop              10 (default)
mse_sendfile_min            0b [bytes] (default)
```

It is recommended to keep these at the default values, with the exception of mse_bigalloc. This parameter should be as large as the bigger objects cached, for example 5MB for HTTP Live Streaming setups.

4.5.5 Advanced configuration

4.5.6 Using multiple stores

Using multiple storage segments is possible. For example it can be beneficial to use malloc storage (standard) for normal objects, and MSE for special objects.

This can be done through setting explicit storage in `vcl_backend_response`, and giving each storage defined on the command line a name:

```
# varnishd -s memory=malloc,1G -s msestorage=mse,/var/lib/varnish/mse,100G
```

Example VCL:

```
# Note: advanced usage
sub vcl_backend_response {
    set beresp.storage_hint = "memory";
    if (bereq.url ~ ".mp4$") {
        set beresp.storage_hint = "msestorage";
    }
}
```

4.5.7 Number of MSE segments

Number of storage segments within MSE can be set on the command line.

Increasing it reduces lock contention when doing memory allocations, especially when forcefully expunging content is necessary.

See the *varnishd* manual pages for more information on tuning this parameter for big cache sizes.

4.6 Package Relocation (Enterprise Linux)

4.6.1 Description

Varnish Cache Plus supports relocated installation. This means that the software can be installed to a separate non-system-wide location, for example */opt/varnish/*.

Normal use of this is to comply with local (security) policies. This functionality should be used sparingly, as it brings with extra complications when it comes to installation and setup of the software.

4.6.2 Installation

Installing relocated must be done using the RPM tool directly, as the normal package manager *yum* does not support it.

A script for doing semi-automatic modifications of changed paths in installed scripts/config files is added. Run this as described after the package installation is done.

Installation steps:

```
$ yum install yum-utils
$ yumdownloader varnish-plus varnish-plus-libs varnish-plus-libs-devel
(Install packages that Varnish Plus depends on, see below)
$ rpm -Uvh --relocate /=/opt/varnish varnish-plus*.rpm
$ /usr/share/doc/varnish-plus-*/relocate /opt/varnish
```

To install the packages Varnish depends on it is easiest to install it non-relocated using *yum* first, and then uninstall it. This allows the automatic dependency solver in *yum* to do the work for you.

All files should now be installed to */opt/varnish/*. The *relocate* script will try to create system-wide symlinks for some necessary paths. PATH, LDPATH and MANPATH defaults will be appended to.

After this it should be possible to start Varnish Cache Plus using the normal procedure: `/etc/init.d/varnish start`

4.6.3 Availability

Relocatable install is supported on Enterprise Linux 6 starting from Varnish Cache Plus 4.0.3r5. Enterprise Linux 7 has the necessary changes, but will require some additional manual steps on installation.

Availability of other Varnish Plus components is limited when running on a relocated installation.

4.7 DeviceAtlas for Varnish

4.7.1 Description

DeviceAtlas is a commercial device detection database offered by Afilias subsidiary dotMobi. It has comprehensive knowledge about HTTP client's specifications, abilities and physical dimensions. This can be used for tailoring HTML responses to the specific type and size of the unit.

By moving this determination into Varnish, it is possible to do this with the normal low latency and high scalability that Varnish provides.

This is implemented as a Varnish Module (VMOD) that interfaces between Varnish and the dataset supplied by dotMobi.

Further information can be found on:

- https://www.varnish-software.com/partner/device-atlas

4.7.2 VCL example

VCL code:

```
import deviceatlas;
sub vcl_recv {
    if (deviceatlas.lookup_int(req.http.User-Agent, "displayWidth") <=␣
↪300) {
        set req.http.X-devicetype = "tiny";
    }
}
```

DeviceAtlas supports all of the recommended VCL configurations described in the Varnish Cache documentation:

https://www.varnish-cache.org/docs/3.0/tutorial/devicedetection.html

4.7.3 Availability

DeviceAtlas is available in packaged form for Varnish Cache and Varnish Cache Plus.

4.7.4 Installation

DeviceAtlas support for Varnish is a product offering on top of Varnish Plus. It requires a separate license from dotMobi.

Contact sales@varnish-software.com for more information.

4.8 Parallel ESI

4.8.1 Description

Varnish Cache Plus features an improved ESI delivery implementation.

Traditionally, all ESI includes in Varnish were fetched in sequential order, one after the other as they are required for delivery.

The improved ESI delivery implementation will seek out all the include fragments and issue backend fetches for all of them concurrently, which in turn significantly reduces the load times for ESI content.

4.8.2 Availability

Parallel ESI is available in Varnish Cache Plus 4.1.4r5 and later.

4.8.3 Installation and configuration

Parallel ESI is built into supported Varnish Cache Plus versions, and does not require any additional installation steps.

The parallel fetch behavior replaces the previous sequential fetch implementation. No parameter settings are required in order to enable it.

4.9 Transfer rate limiting

4.9.1 Description

Socket pacing is a Linux method for rate limiting TCP connections in a network friendly way.

When running on systems with the necessary kernel support, Varnish can now artificially slow down sending of response bodies to requesting clients.

4.9.2 VCL example

VCL code:

```
import std;
import tcp;

sub vcl_recv {
    # Limit all clients to 1000 KB/s.
    tcp.set_socket_pace(1000);
    # In 4.0:
    # std.set_socket_pace(1000);
}
```

4.9.3 Availability

Rate limiting is available starting from Varnish Cache Plus 4.0.3r1.

Kernel support is needed for this feature to work. Ubuntu 14.04 (trusty) is known to work.

4.9.4 Installation

Rate limiting is built into Varnish Cache Plus and does not need specific installation.

Servers utilizing socket pacing must change their network scheduler. This can be done with:

```
$ tc qdisc add dev eth0 root handle 1: fq
```

This change should be persisted across reboots, for example via */etc/rc.local*.

4.10 hashtwo (4.0)

4.10.1 Description

hashtwo is an implementation of surrogate cache keys for Varnish Cache Plus 4.0.

It operates through having the backend webserver tag resources with one or more identifiers, which is then organised for very speedy invalidation inside Varnish.

This allows for invalidating content from the cache using the fast purge method, at request rates of tens of thousands per second.

Further information can be found on:

- https://www.varnish-software.com/blog/advanced-cache-invalidation-strategies
- https://www.varnish-cache.org/utility/secondary-hash-hash-ninja

In 4.1 this module has been open sourced and renamed to *xkey*.

4.10.2 VCL example

hashtwo is implemented as a set of Varnish core extensions which is used by a proprietary Varnish module (VMOD).

VCL code:

```
import hashtwo;
sub vcl_recv {
     if (req.http.x-hashtwo-purge) {
          if (hashtwo.purge(req.http.x-hashtwo-purge) != 0) {
               error 200 "Purged";
          } else {
               error 404 "Key not found";
          }
     }
}

sub vcl_fetch {
     # Use the header vmod to add multiple headers for multiple keys
     set beresp.http.X-HashTwo = "secondary_hash_key";
}
```

4.10.3 Availability

hashtwo is available starting from versions:

- Varnish Cache Plus 3.0.5r1
- Varnish Cache Plus 4.0.0r1

In Varnish Cache Plus 4.1 the surrogate key support is available under the name *xkey*.

4.10.4 Installation

The hashtwo VMOD is prebuilt for supported versions and can be installed using regular package managers from Varnish Software repositories.

The package contains further installation and usage instructions.

Contact support@varnish-software.com if you need assistance.

4.11 Traffic accounting (3.0)

4.11.1 Description

Traffic accounting helps companies running shared Varnish infrastructure to bill their customers for data transfers.

These counters can either be extracted with the *varnishncsa* logging utility, or bespoke logging applications that read *varnishlog* data directly.

4.11.2 Example output

The output from *varnishlog* contains the additional records:

```
12 SessionOpen  c 77.40.251.100 59726 :6081
12 ReqStart     c 77.40.251.100 59726 32454431
12 RxRequest    c GET
12 RxURL        c /
12 RxProtocol   c HTTP/1.1
12 RxHeader     c TE: deflate,gzip;q=0.3
12 RxHeader     c Connection: TE, close
12 RxHeader     c Host: www.example.com
12 RxHeader     c User-Agent: lwp-request/6.03 libwww-perl/6.05
12 VCL_call     c recv lookup
12 VCL_call     c hash
12 Hash         c /
12 Hash         c www.example.com
12 VCL_return   c hash
12 VCL_call     c miss fetch
12 Backend      c 14 default default
12 VCL_Log      c bereq_hdr_sz:164
12 VCL_Log      c bereq_sz:164
12 VCL_Log      c beresp_hdr_sz:321
12 TTL          c 32454431 RFC 604800 -1 -1 1400591008 0 1400591008␣
↪1401195808 604800
12 VCL_call     c fetch deliver
12 ObjProtocol  c HTTP/1.1
12 ObjResponse  c OK
[.. cut for brevity ..]
12 TxHeader     c Via: 1.1 varnish
12 TxHeader     c Connection: close
12 VCL_Log      c req_hdr_sz:135
12 VCL_Log      c req_body_sz:0
12 VCL_Log      c req_sz:135
12 VCL_Log      c resp_hdr_sz:387
12 VCL_Log      c resp_body_sz:1270
12 VCL_Log      c resp_sz:1657
12 Length       c 1270
12 VCL_Log      c beresp_body_sz:1270
12 VCL_Log      c beresp_sz:1591
12 ReqEnd       c 32454431 1400591007.697987318 1400591007.913085699
                  0.000149488 0.214946985 0.000151396
12 SessionClose c Connection: close
12 StatSess     c 77.40.251.100 59726 0 1 1 0 0 1 387 1270
```

4.11.3 Exporting accounting data with varnishncsa

varnishncsa has a configurable output format. By taking the default format and extending it with the necessary `%{VCL_Log:KEYNAME}` statements, it is possible to output the accounting records into a regular Apache/NCSA logfile.

Example command line:

```
$ varnishncsa -F '%h %l %u %t "%r" %s %b "%{Referer}i" "%{User-agent}i"
        %{Varnish:time_firstbyte}x %{VCL_Log:BeResp}x %s %{VCL_Log:resp_sz}
↪x
        %{VCL_Log:req_sz}x %{VCL_Log:beresp_sz}x %{VCL_Log:bereq_sz}x'
```

Which will produce log lines (line feeds added for formatting) like this:

```
77.40.251.100 - - [20/May/2014:15:11:18 +0200] "GET http://example.com/␣
↪HTTP/1.1"
        200 1270 "-" "lwp-request" 0.198806763 - 200 1659 135 1591 166
77.40.251.100 - - [20/May/2014:15:11:20 +0200] "GET http://example.com/␣
↪HTTP/1.1"
        200 1270 "-" "lwp-request" 0.000338793 - 200 1670 135 - -
77.40.251.100 - - [20/May/2014:15:11:20 +0200] "GET http://example.com/␣
↪HTTP/1.1"
        200 1270 "-" "lwp-request" 0.000045300 - 200 1670 135 - -
```

The initial request is a miss leading to a backend fetch, and have data for `beresp_sz` and `bereq_sz`. The remaining requests are cache hits and only have client-side counter values.

4.11.4 Availability

Traffic accounting is available in Varnish Cache Plus 3.0.5r1 and newer.

This functionality has been incorporated into Varnish Cache 4.0 and later using a slightly different format.

4.11.5 Installation

Traffic accounting is built into Varnish Cache Plus 3.0 and is always enabled.

Contact support@varnish-software.com if you need assistance in extracting the data in a format suitable for your accounting system.

4.12 leastconn

4.12.1 Description

This VMOD implements a least connections director for Varnish. The director will direct traffic to the backend with the least number of active connections.

Each backend has an associated weight. A backend with e.g. weight 10 will balance at twice the number of connection as a backend with weight 5.

An optional linear ramp up time can be specified for the director. If set, the weight of each backend is linearly increased in the time frame until it reaches the set weight. This allows for backends that have e.g. recently become healthy again to be eased into the workload. Without a ramp up time, the newly inserted backend would get all the load until it has its share, which can have detremental effects on some applications.

Only actual backends can be added, ie. it is not stackable. If another director is added as a backend, it will be ignored.

Example:

```
vcl 4.0;
import leastconn;

backend backend1 {
        .host = "192.168.0.10";
        .port = "80";
}

backend backend2 {
        .host = "192.168.0.11";
        .port = "80";
}

sub vcl_init {
        new dir = leastconn.leastconn();
        dir.rampup(1m);
        dir.add_backend(backend1, 3);
        dir.add_backend(backend2, 8);
}

sub vcl_backend_fetch {
        set bereq.backend = dir.backend();
}
```

CHAPTER 5

Troubleshooting

Varnish Cache Plus is a high-performance web accelerator and may be difficult to get a handle on initially.

5.1 Online help resources

There are a lot of forums and pages offering Varnish help.

Here are some we find useful:

- https://stackoverflow.com/questions/tagged/varnish
- varnish-misc email list: https://www.varnish-cache.org/lists/mailman/listinfo/varnish-misc

5.2 Panics

Varnish Cache Plus is normally quite stable.

There should not be any panics in normal usage.

As backend behaviour and client loads are not easily simulated and tested, sometimes Varnish ends up in a state where it recognizes that it should not be. In those cases Varnish will perform a controlled panic, syslog what went wrong, and restart. This output contains useful information on what happened.

In cases where this is not written to syslog on a crash, support can offer guidance on how to extract it from core dumps.

5.3 Getting help

Contact Varnish Software support on support@varnish-software.com.

Often support will ask you for a varnishgather data collection. If you run this script and send that on the initial request, support will be able to help you sooner.

CHAPTER 6

Appendix A: Changelogs

6.1 Changelog for Varnish Cache Plus 4.1

6.1.1 Varnish Cache Plus 4.1.7r3 (2017-07-27)

- Per MSE segment counters added to the manual.

6.1.2 Varnish Cache Plus 4.1.7r2 (2017-07-07)

- Fit full backend names in 128 chars (VCP issue #151)
- The default value for vcl_reclen is now 4048 bytes (up from 255)
- Fix goto.dns_backend()

6.1.3 Varnish Cache Plus 4.1.7r1 (2017-06-28)

- Makes goto's backend resolutions always non-blocking.
- Add dns_director() and dns_backend() to goto and deprecate the other methods.
- Add vmod_vha to ease VHA integration.
- Close a race in probe's release of connection pool handles.

6.1.4 Varnish Cache Plus 4.1.7r1-beta1 (2017-06-23)

- Work around a synchronization issue with regard to late overwrites of object attributes in the fetch cycle. (VCP issue #143)

6.1.5 Varnish Cache Plus 4.1.6r2 (2017-05-30)

- Edgestash 1.0.6 via vmod_edgestash.
- Embedded VMODs from varnish-modules have been updated to version 0.12.0.
- Add libvmod_rewrite. This is a utility vmod for giving lists of rewrite rules to apply.
- Make debug.jemalloc_stats take a default argument of 'a'. This reduces the amout of output significantly (VCP issue #142).

- Increase default cli_limit to 128k. This makes it possible to transfer larger buffers in the varnish-gather data (VCP issue #142).

6.1.6 Varnish Cache Plus 4.1.6r1 (2017-05-09)

- Add a debug.jemalloc_stats CLI command to print statistics from the jemalloc library.
- Fix a potential backend leak in vmod-goto.
- Add optional `host_header` argument to goto functions.
- Add a serial number in goto backend names to avoid collisions.
- Make startup_timeout only take effect if it is larger than cli_timeout. This fixes issues experienced on upgrade to the latest VCP when the change to cli_timeout didn't take effect. (VCP issue 141).

6.1.7 Varnish Cache Plus 4.1.5r2 (2017-04-21)

- Added a new startup_timeout for CLI commands that take a long time when the child process is starting. This is especially useful for very large MSE setups.
- Added SSL support to vmod-named.
- Fix a workspace and proxy protocol related issue (VCP issue 134).

6.1.8 Varnish Cache Plus 4.1.5r2-beta2 (2017-04-07)

- Fix a VCL temperature issue when the child fails on warming the VCL, which would cause a missmatch between master and child's VCL temperature state leading to asserts later. (VC pull 2273)
- Fix a couple of issues with regard to libvmod_goto's cleanup code that would cause problems when discarding a VCL and the VCL was never set to warm, and when transitioning from warm to cold and then back to warm again. This could lead to asserts and/or leaking of threads. (VCP issue 127).
- Add more verbose error logging on master-child CLI communication.
- Fix a problem related to dynamic backend cleanup that would cause the child process' CLI thread to become stuck. This would again cause the master process to kill the child because of CLI timeout. (VC issue 2295)

6.1.9 Varnish Cache Plus 4.1.5r2-beta1 (2017-02-23)

- Add protocol byte counters to the ReqAcct and BereqAcct log records. These counters show the number of bytes that were successfully read or written to/from the OS socket buffers, including protocol overhead. These counters are more accurate, as they will not show bytes that was never sent e.g. on a client hangup. Varnishncsa has been updated to make use of these new byte counts. (Issue #116).
- Fix request body support on cache miss (VC issue 1927)

6.1.10 Varnish Cache Plus 4.1.5r1 (2017-02-13)

No changes since 4.1.5r1-beta1

6.1.11 Varnish Cache Plus 4.1.5r1-beta1 (2017-02-10)

- Add parameters *connect_timeout*, *first_byte_timeout*, *between_bytes_timeout* and *max_connections* to vmod-goto.
- Fix a resource leak in vmod-goto leading to unrealeased backends.
- Backend SSL code updated to support OpenSSL 1.1.0
- Embedded VMODs from varnish-modules have been updated to v0.11.0.
- Add parameters *ssl_sni*, *ssl_verify_peer* and *ssl_verify_host* to vmod-goto. These function like their counterparts used in *backend* definitions.
- Fix an issue with MSE2 where we did not journal correctly that an object had been deleted. This could lead to a situation where we attempted to repopulate a half deleted object on restart, leading to assert. (Issue #112).

6.1.12 Varnish Cache Plus 4.1.4r5 (2016-12-13)

- Fix a leak in parsing string in vmod-goto.
- Introducing parallel ESI: Varnish will now do fetches of ESI fragments in parallel.
- Add an esi_maxdepth counter. This indicates the number of times parameter max_esi_depth was hit.
- vmod-kvstore added get/set for BACKEND types and ini file support
- Fix vmod-goto related crash when discarding VCLs.
- vmod-goto now logs failures to retrieve at least one IP.

6.1.13 Varnish Cache Plus 4.1.4r4 (2016-12-02)

- Send stream events from the MSE object iterator when reaching the end of available data. This allows receiving streaming client connections to keep closer to the end of available data. (Issue #99)
- Fix and clarify argument priority of goto.backend() and goto.director().
- Various fixes in vmod-named

6.1.14 Varnish Cache Plus 4.1.4r4-beta1 (2016-11-24)

- vmod-session has been added. It lets you set the session idle timeout on a per session basis. Some changes in the core was necessary.

6.1.15 Varnish Cache Plus 4.1.4r3 (2016-11-03)

- Fix timeout issue on SSL backend probes. (Issue #90)
- vmod-rtstatus 1.2.1 (28f1ffc) has been added to the distribution.
- New runtime parameter *clock_step* specifying how much observed clock step we allow before panicking. (varnish-cache.org issue 1874)
- Least-connection backend director has been added to the distribution.
- Avoid losing varnishadm CLI synchronization. (varnish-cache.org issues 2026 and 2010)
- Clean up vsm files on startup failure. (varnish-cache.org issue 2115)
- Fix bug that added superfluous "duplicate link" in varnishlog. (varnish-cache.org issue 1830)

6.1.16 Varnish Cache Plus 4.1.4r2 (2016-10-10)

- [vmod-goto] Added support for acting as a director, allowing for stacking them behind other directors.
- Close a race between the ban lurker and nuked objects allowing the lurker to evaluate objects being dismantled. (VS issue #91 and #92)
- Remove a feature designed to reduce the size of coredumps by eliminating unneeded object payload data. This caused an excessive amount of process map entries to be created, causing kernel resource exhaustion and memory allocation failures. (VS issue #93)

6.1.17 Varnish Cache Plus 4.1.4r1 (2016-09-22)

- vmod-goto has been added to the distribution, allowing for using backends defined at request-time.

6.1.18 Varnish Cache Plus 4.1.4r1-beta1 (2016-09-14)

- Changes added between Varnish Cache 4.1.3 and Varnish Cache 4.1.4-beta1 except commits ca3fde2, fe1c483, 2b279cd and 1774719 has been added. See doc/changes.rst for details.
- Add .ssl_verify_host attribute to *backend* . If enabled, the connection will fail if the peer's certificate hostname does not match the hostname for this connection.

6.1.19 Varnish Cache Plus 4.1.3r2-beta1 (unreleased)

- Parameter *workspace_client* default increased from 64k to 96k.
- Parameter *workspace_backend* default increased from 64k to 128k.
- Fix an MSE persistent issue when having a combination of ESI, gzip and failed fetch which would lead to an assert. (VS issue #76)
- Fix a problem where MSE would attempt to persist failing objects (e.g. due to fetch failure), causing persisted problems. (VS issue #77)
- Fix a locking issue in MSE object freeing code path allowing the hole expansion to attempt to free an object being dismantled in another thread causing assertion. (VS issue #69)

- Fix a race on setting maximum stream limits for the simple stevedores. This affected all of the stevedores except MSE. (VS issue #71).

6.1.20 Varnish Cache Plus 4.1.3r1 (2016-07-08)

- Changes added between Varnish Cache 4.1.2 and Varnish Cache 4.1.3 has been added. See doc/changes.rst for details.
- Embedded VMODs from varnish-modules have been updated to v0.9.1.

6.1.21 Varnish Cache Plus 4.1.2r2 (2016-06-16)

- Add a memory dump on CHECK_OBJ assertions. This will dump some memory from the area of the offending object to help with debugging.
- Add MSE object structure debug output to the panic log. This is to help with debugging.
- Add a shared maps section to the panic output.
- Add backend mode to varnishncsa.
- Fix failure to call fetch processor clean up (typically gzip/ungzip or ESI buffer leak) when handling fetch failures due to workspace exhaustion.
- Fix failure to call object finalization when inserting synthetic objects in the cache. This would cause MSE objects to not store their attributes correctly, causing assertions on access. (VCP #62)
- Fix ESI byte code allocation size handling and resulting buffer overflow. Tracked in varnish-cache.org ticket 1941.
- Fix a ESI+gzip corner case which had escaped notice until now. Tracked in varnish-cache.org ticket 1878.
- Avoid small memory leak on malformed ESI directives. (VC issue 1912)
- Release memory instead of crashing on malformed ESI. (VC issue 1904)
- Revive the backend_conn counter. Tracked in varnish-cache.org ticket 1725.
- When the log is overrun and reacquired in the logging tools, the -d option is maintained.

6.1.22 Varnish Cache Plus 4.1.2r1 (2016-03-30)

No significant changes since 4.1.2r1-beta3.

Bugs fixed:

- Avoid assertion on errors reported during fetch processor initialization (typically running out of backend workspace). Tracked in varnish-cache.org ticket 1871.

6.1.23 Varnish Cache Plus 4.1.2r1-beta3 (2016-03-29)

This is Varnish Cache Plus 4.1.2r1-beta3, based on Varnish Cache 4.1.2.

Changes since 4.1.2r1-beta2:

- Man pages for new vmods have been added.

- Correct handling of duplicate headers on IMS header merge. This ensures all instances of a header on the source (cached object) is copied to the new IMS-verified object. Previously only the first instance of a given header was copied. Tracked in varnish-cache.org ticket 1879.
- Parameter *mse_sendfile_min* retired. No sendfile in MSE2.
- Align exported bans to avoid losing one on restart. (Issue #55)
- Remove cosmetic *varnishadm* tab completion warning. (Issue #34)
- vmod-cookie updated to remove debug output. (Issue #50)
- vmod-acl is now included.
- *mkfs.mse* has been moved to /usr/sbin/.

6.1.24 Varnish Cache Plus 4.1.2r1-beta2 (2016-03-18)

This is Varnish Cache Plus 4.1.2r1-beta2, based on Varnish Cache 4.1.2.

Changes since 4.1.2r1-beta1:

- Rework how the persistance MSE book is read during startup, to avoid random IO leading to long startup time.
- vmod-kvstore (hash map datastructure in VCL) is now included.
- *mkfs.mse* man page added. varnishd(1) discrepancies on mse syntax updated.
- Workaround for VC issue #1806 added, fixing problems seen when a POST request piped over a reused backend connection.

6.1.25 Varnish Cache Plus 4.1.2r1-beta1 (2016-03-11)

This is Varnish Cache Plus 4.1.2r1-beta1, based on Varnish Cache 4.1.2 with the following additions:

- MSE2 storage module. (See `mkfs.mse -h`)
- Backend SSL support. See *README-SSLBACKEND.rst* and *vcl(7)*.
- Embedded Varnish modules: - cookie - header - var - vsthrottle - softpurge - saintmode - tcp - paywall - xkey

6.1.26 Varnish Cache Plus 4.1.2r0-tp1 (2016-02-19)

This is a technology preview (TP) release of VCP4.1.

It is based on Varnish Cache 4.1.2-beta1 with MSE2 (incl. persistence support) added. No other additions.

6.2 Changelog for Varnish Cache Plus 4.0

6.2.1 Varnish Cache Plus 4.0.4r2 (2017-07-27)

- Backport a fix for a bias issue with the hash director. Tracked in varnish-cache.org ticket 1658.

- Preserve the vdir API.
- Support OpenSSL 1.1. Varnish Plus 4.0 now compiles on distributions with OpenSSL 1.1.

6.2.2 Varnish Cache Plus 4.0.4r1 (2016-10-28)

No functional changes.

6.2.3 Varnish Cache Plus 4.0.4r1-beta1 (2016-10-24)

- *sigsegv_handler* is now enabled by default. (Issue #32)
- Add a memory dump on CHECK_OBJ assertions. This will dump some memory from the area of the offending object to help with debugging.
- Add a shared maps section to the panic output.
- Wrap VXID at 30 bits. The top two bits of the 32-bit VXIDs are used as backend/client markers. Log records emitted using these bits would confuse the logging API causing excessive memory usage in VSL client applications (e.g. varnishlog). Tracked in varnish-cache.org ticket 1762.
- Ensure that probe timeouts are honored when using probes with backend SSL. Earlier, the probes could hang if e.g. there was packet loss during the SSL handshaking. (Issue #90)

This release is built on Varnish Cache 4.0.4-beta1.

6.2.4 Varnish Cache Plus 4.0.3r6 (2016-04-11)

- Properly encode HTTP headers with weird characters to C identifiers. This fixes VCL compilation issues when mixing '-' and '_' in header names. Tracked in varnish-cache.org ticket 1768.
- Close piped sessions in the absence of a backend. This could lead to an assert if the director used failed to provide a healthy backend on pipe. Tracked in varnish-cache.org ticket 1815.
- Trigger HTTP protocol parse failure on whitespace in the URL. Tracked in varnish-cache.org ticket 1862.
- Only use fallocate() on filesystems with sane behaviour on this system call. Specifically xfs is known to require the size available as free space to succeed when calling fallocate() even on a fully allocated file on RHEL systems. This causes problems specifically for MSE and it's large storage file. Only ext4 is whitelisted for fallocate(). Backported from varnish-cache.org ticket 1792. Ref varnish-cache-plus ticket #61.
- Issue a warning if you point MSE to use a directory for storage. Using a directory is not recommended because of a race with the kernels unlinking of the previous storage file on daemon restart, causing no space errors.
- Close a race in MSE with regard to access and freeing of the temporary object header storage. This could lead to assertions in late delivery. Tracked in varnish-cache-plus ticket #33.

6.2.5 Varnish Cache Plus 4.0.3r6-beta1 (2016-03-16)

- Prevent a segmentation fault on adding a backend that already exists but have lost one of its IPv4 or IPv6 components.

- Prevent an erraneous assertion from triggering when exceeding the VSM shared counter space and dropping backends.
- Fix handling of duplicate headers on IMS header merge. Tracked in varnish-cache.org ticket 1879.

6.2.6 Varnish Cache Plus 4.0.3r5 (2016-02-18)

No changes since 4.0.3r5-beta1.

6.2.7 Varnish Cache Plus 4.0.3r5-beta1 (2016-01-22)

- Varnish Cache trac ticket 1823 backported. Rush/wake requests on waiting list so VCL references can be freed more easily.
- Fix race condition when an object is both banned and attempted nuked at the same time.
- Remember to lock backend reference count on VCL load.

6.2.8 Varnish Cache Plus 4.0.3r4 (2015-11-19)

- Improve HTTP Range handling, especially on streaming objects received as chunked from the backend server. Adds req.do_range in VCL to disable range handling per request.
- Fix issue with backend TLS health probes failing when connection is closed without a close_notify. (issue #20)
- Backend TLS/SSL documentation has been improved.

6.2.9 Varnish Cache Plus 4.0.3r3 (2015-06-22)

- Patches applied to VC 4.0 since 4.0.3 applied. See changes.rst for details.
- No functional changes to SSL/TLS code.

6.2.10 Varnish Cache Plus 4.0.3r3-beta1 (2015-05-21)

- Add experimental support for SSL/TLS on backend connections. OpenSSL is now a required library.

6.2.11 Varnish Cache Plus 4.0.3r2 (2015-04-27)

- No code changes since 4.0.3r2-beta2.

6.2.12 Varnish Cache Plus 4.0.3r2-beta2 (2015-04-15)

- Added new parameter *timeout_reqbody*. This can be used to enable an experimental request timeout to mitigate the RUDY attack. Disabled by default.

6.2.13 Varnish Cache Plus 4.0.3r2-beta1 (2015-03-31)

- Added support for PROXY protocol. (client-side)

http://www.haproxy.org/download/1.5/doc/proxy-protocol.txt

6.2.14 Varnish Cache Plus 4.0.3r1 (2015-03-20)

- No changes since 4.0.3r1-rc1.

6.2.15 Varnish Cache Plus 4.0.3r1-rc1 (2015-03-03)

- Changes and bugfixes introduced in Varnish Cache 4.0.3 are added.
- Added limitation on maximum single segment size in MSE.

This only concerns internal size of storage blocks (<= UINT_MAX -1) and does not affect maximum object size. Added after debugging a object corruption bug on RHEL6 with kernel 2.6.32-279.14.1. Fixed in (or before) 2.6.32-431.17.1.

- Added support for socket pacing / rate limiting.

Added function std.set_socket_pace() that allows for delivering a response body TCP rate limited using Linux socket pacing.

6.2.16 Varnish Cache Plus 4.0.2r1 (2014-11-27)

This is the first release of Varnish Cache Plus 4.0, built on top of Varnish Cache 4.0.2-mse2

Changes on top of Varnish Cache 4.0.2-mse2:

- Add libvmod-hashtwo expiry callback
- Added modifications required to run libvmod-hashtwo in 4.0.

6.2.17 Changes from 4.0.1-mse1 to 4.0.2-mse2 (2014-11-17)

- Added support for ESI.
- Backend response Content-Length header is kept in more instances, allowing for less use of chunked encoding and lower memory overhead per cached object. (Bug #1506)
- Bug #1596
- Bug #1612

6.3 Changelog for Varnish Cache Plus 3.0

6.3.1 Changes from 3.0.6r3 to 3.0.7r1 (unreleased)

This release is based on Varnish Cache 3.0.7.

Bugs fixed:

- VC issue 1531 / VCP issue #10: libedit and varnishadm.

6.3.2 Changes from 3.0.6r2 to 3.0.6r3 (2014-09-08)

- std.ip() was backported from 4.0.
- Fix bug on traffic accounting where bytes transfer was not reset when using HTTP Keepalive.
- Fix bug on traffic accounting's beresp_body_sz when streaming.

6.3.3 Changes from 3.0.6r1 to 3.0.6r2 (2014-07-17)

- **Maintenance release.** This release updates the pkg-config setup to simplify building of VMOD packages. No functional changes to Varnish Cache Plus itself.

6.3.4 Changes from 3.0.5r1 to 3.0.6r1 (2014-06-25)

- **Maintenance release.** This release pulls in changes made between in Varnish Cache 3.0.5 and Varnish Cache 3.0.6 (unreleased) into Varnish Cache Plus.

6.3.5 Changes from 3.0.2s to 3.0.5r1. (2014-05-19)

- **(T11) Rebased from Varnish Cache 3.0.5.** This release is built on top of the Varnish Cache 3.0.5 release.
- **(T9) System jemalloc is now used (and required) instead of our bundled copy.** The bundled version of jemalloc in 3.0 is fairly old. This was added to the Varnish source tree at a time where jemalloc wasn't available prepackaged. It is now, and we should use this newer (and better) version over our old bundled copy.
- **(T21) Added byte counters for accounting.** Accounting records for bytes transferred are available on varnishlog. This functionality is an early version of the accounting record work that is in 4.0. It has seen extensive production traffic.

6.3.6 3.0.2s (2012/2013)

Features and improvements in 3.0.2s versus Varnish Cache 3.0:

- **Concurrent streaming fetches are not blocking.** In regular 3.0, streaming support meant that the first client requesting a resource got it streaming through from the backend, while the remaining clients were given a graced copy or put on the waiting list.

  In 3.0.2s this restriction is lifted. The 2..n'th client get bytes as soon as they are retrieved from the backend.
- **Support for client-side Range requests while streaming.** The Range request header is ignored for streaming requests in Varnish Cache 3.0. This restriction is lifted in 3.0.2s.
- **Added hashtwo callbacks.** Added the necessary callbacks to support the hashtwo surrogate keys VMOD.

- **Improved persistence.** In stock 3.0 varnish crashes if the allocated storage fills up. Improved the persistent storage backend so it supports forced expiry using the FIFO algorithm.