Written February 2009 by Jonny Bergkvist, email jonny.bergkvist at yahoo.no.

Varnish 2.0.x and eZ publish 4.0.x

Technical implementation guide for running Varnish and eZ publish on the same server to achieve high performance on your eZ publish web site.

Server-environment for this use-case is Debian Linux Etch with the deb-packages for apache2, php5, mysql-server 5 and related packages. Varnish is compiled from source because the deb-package of Varnish is too old.

Ref: http://varnish.projects.linpro.no/, and http://ez.no/ezpublish

«Varnish is a state-of-the-art, high-performance HTTP accelerator. It uses the advanced features in Linux 2.6, FreeBSD 6/7 and Solaris 10 to achieve its high performance.»

«eZ Publish is an Open Source Content Management System chosen by thousands of enterprises and organizations world wide. It helps you build corporate websites, intranets, webshops and media portals. eZ Publish is 100% Open Source, available either as a free download or as an enterprise solution "eZ Publish Premium" with support, guarantees and maintenance.»

My example doesn't utilize cache-purging on publish. I have configured Varnish to automatically purge cache for content after 5 minutes, and images, media, css and js are purges every 10 minutes.

Purge on publish should be possible to setup by integrating the eZ extension: http://projects.ez.no/all2evcc, but I have not done it yet.

Varnish installation

Grab Varnish 2.0.x from source and compile and install following the Varnish Installation documentation at http://varnish.projects.linpro.no/wiki/Installation

Create configuration file /etc/default/varnish and /etc/varnish/vcl.conf

Create startup-file /etc/init.d/varnish and symlink to startup-file in default runlevel to start varnish on server boot.

Varnish configuration

Example configuration file /etc/default/varnish:

```
# Configuration file for varnish
#
# /etc/init.d/varnish expects the variable $DAEMON_OPTS to be set from this
# shell script fragment.
#


# Maximum number of open files (for ulimit -n)
NFILES=131072
```

```
# Default varnish instance name is the local nodename.  Can be overridden with
# the -n switch, to have more instances on a single server.
INSTANCE=$(uname -n)

# This file contains 4 alternatives, please use only one.

## Alternative 1, Minimal configuration, no VCL
#
# Listen on port 6081, administration on localhost:6082, and forward to
# content server on localhost:8080.  Use a fixed-size cache file.
#
#DAEMON_OPTS="-a :80 \
#             -T localhost:6082 \
#           -b localhost:81 \
#           -u www-data -g www-data \
#           -s file,/var/lib/varnish/$INSTANCE/varnish_storage.bin,1G"


## Alternative 2, Configuration with VCL
#
# Listen on port 6081, administration on localhost:6082, and forward to
# one content server selected by the vcl file, based on the request.  Use a
# fixed-size cache file.
#
 DAEMON_OPTS="-a :80 \
             -T localhost:6082 \
             -f /etc/varnish/vcl.conf \
             -s file,/var/lib/varnish/$INSTANCE/varnish_storage.bin,1G"


## Alternative 3, Advanced configuration
#
# See varnishd(1) for more information.
#
# # Main configuration file. You probably want to change it :)
# VARNISH_VCL_CONF=/etc/varnish/default.vcl
#
# # Default address and port to bind to
# # Blank address means all IPv4 and IPv6 interfaces, otherwise specify
# # a host name, an IPv4 dotted quad, or an IPv6 address in brackets.
```

```
# VARNISH_LISTEN_ADDRESS=
# VARNISH_LISTEN_PORT=6081
#
# # Telnet admin interface listen address and port
# VARNISH_ADMIN_LISTEN_ADDRESS=127.0.0.1
# VARNISH_ADMIN_LISTEN_PORT=6082
#
# # The minimum number of worker threads to start
# VARNISH_MIN_THREADS=1
#
# # The Maximum number of worker threads to start
# VARNISH_MAX_THREADS=1000
#
# # Idle timeout for worker threads
# VARNISH_THREAD_TIMEOUT=120
#
# # Cache file location
# VARNISH_STORAGE_FILE=/var/lib/varnish/$INSTANCE/varnish_storage.bin
#
# # Cache file size: in bytes, optionally using k / M / G / T suffix,
# # or in percentage of available disk space using the % suffix.
# VARNISH_STORAGE_SIZE=1G
#
# # Backend storage specification
# VARNISH_STORAGE="file,${VARNISH_STORAGE_FILE},${VARNISH_STORAGE_SIZE}"
#
# # Default TTL used when the backend does not specify one
# VARNISH_TTL=120
#
# # DAEMON_OPTS is used by the init script.  If you add or remove options, make
# # sure you update this section, too.
# DAEMON_OPTS="-a ${VARNISH_LISTEN_ADDRESS}:${VARNISH_LISTEN_PORT} \
#              -f ${VARNISH_VCL_CONF} \
#              -T ${VARNISH_ADMIN_LISTEN_ADDRESS}:${VARNISH_ADMIN_LISTEN_PORT} \
#              -t ${VARNISH_TTL} \
#              -w ${VARNISH_MIN_THREADS},${VARNISH_MAX_THREADS},${VARNISH_THREAD_TIMEOUT}
\
#              -s ${VARNISH_STORAGE}"
#



## Alternative 4, Do It Yourself
```

```
#
# DAEMON_OPTS=""
```

Example configuration file /etc/varnish/vcl.conf:

```
# Default backend definition.  Set this to point to your content
# server.


backend b1 {
    .host = "127.0.0.1";
    .port = "6081";
}


acl purge {
      "localhost";
}


## Called when a client request is received
#
#pipe = just a pipe through varnish. No checks whatsoever
#pass = Go through all checks -> everything, just dont lookup in the cache

sub vcl_recv {

# Add a unique header containing the client address
unset req.http.X-Forwarded-For;
  set    req.http.X-Forwarded-For = client.ip;


##### always cache these items:

        ## images
        if (req.request == "GET" && req.url ~ "\.(gif|jpg|jpeg|bmp|png|tiff|tif|ico|img|
tga|wmf)$") {
                set req.backend = b1;
                lookup;
        }
```

```
        ## various other content pages
        if (req.request == "GET" && req.url ~ "\.(css|js)$") {
               set req.backend = b1;
               lookup;
        }

        ## multimedia
        if (req.request == "GET" && req.url ~ "\.(svg|swf|mov|avi|wmv)$") {
               set req.backend = b1;
               lookup;
        }

#### do not cache these files

#        if (req.request == "GET" && req.url ~ "\.(cfm)$") {
#                pass;
#        }

#### do not cache these rules:

        # pass mode can't handle POST (yet)
        if (req.request == "POST") {
                pipe;
        }

        if (req.request != "GET" && req.request != "HEAD") {
                pipe;
        }
        if (req.http.Expect) {
                pipe;
        }
        if (req.http.Authenticate || req.http.Authorization) {
                pass;
        }


#### if there is a purge make sure its coming from $localhost

       if (req.request == "PURGE") {
                if(!client.ip ~ purge) {
                        error 405 "Not Allowed";
```

```
                }
                purge_url(req.http.X-Purge-Url);
                error 200 "Purged";
        }

####


#### unknown function to "normalize the Accept-Encoding headers"

        if (req.http.Accept-Encoding) {
                if (req.http.Accept-Encoding ~ "gzip") {
                        set req.http.Accept-Encoding = "gzip";
                } elsif (req.http.Accept-Encoding ~ "deflate") {
                        set req.http.Accept-Encoding = "deflate";
                } else {
                        # unkown algorithm
                        unset req.http.Accept-Encoding;
                }
        }

####  don't cache authenticated sessions
        if (req.http.Cookie && req.http.Cookie ~ "is_logged_in=") {
                pipe;
        }
        //  Varnish doesn't do INM requests so pass it through if no If-Modified-Since
was sent
        if (req.http.If-None-Match && !req.http.If-Modified-Since) {
                pass;
        }

#### if it passes all these tests, do a lookup anyway;
        lookup;

}

#
## Called when entering pipe mode
#
sub vcl_pipe {
        pipe;
}
```

```
#
## Called when entering pass mode
#
sub vcl_pass {
        pass;
}

#
## Called when entering an object into the cache
#
sub vcl_hash {
        set req.hash += req.url;
        set req.hash += req.http.host;
        hash;
}

#
## Called when the requested object was found in the cache
#
sub vcl_hit {

#  if (obj.http.X-Cache == "MISS") {
#                set obj.http.X-Cache = "HIT";
#        }
  if (!obj.cacheable) {
                pass;
        }
        deliver;
}

#
## Called when the requested object was not found in the cache
#
sub vcl_miss {
    fetch;
}

#
## Called when the requested object has been retrieved from the
## backend, or the request to the backend has failed
```

```
#
sub vcl_fetch {
    # default time to live for cache objects
    set obj.ttl = 300s;

    if (!obj.cacheable) {
        pass;
    }

    if (obj.http.Set-Cookie ~ "is_logged_in=deleted(.*)") {
        deliver;
    }

    if (obj.http.Set-Cookie) {
        pass;
    }

if (req.request == "GET" && req.url ~ "\.(gif|jpg|jpeg|bmp|png|tiff|tif|ico|img|tga|wmf)
$") {
 set obj.ttl = 600s;
        deliver;
        }

        ## various other content pages
        if (req.request == "GET" && req.url ~ "\.(css|js|html)$") {
set obj.ttl = 600s;
        deliver;
        }

        ## multimedia
        if (req.request == "GET" && req.url ~ "\.(svg|swf|ico|mp3|mp4|m4a|ogg|mov|avi|
wmv)$") {
set obj.ttl = 600s;
        deliver;
}

    # for varnish 2.0:
    # set obj.prefetch =  -30s;

    deliver;
}
```

```
#
#
## Called before a cached object is delivered to the client
#
sub vcl_deliver {
    deliver;
}

#
## Called when an object nears its expiry time
# not supportet yet, see http://varnish.projects.linpro.no/ticket/367
#sub vcl_timeout {
#    fetch;
#}

#
## Called when an object is about to be discarded
#
sub vcl_discard {
    discard;
}
```

Example /etc/init.d/varnish start/stop file:

```
#!/bin/sh
#
# varnish       Control the varnish HTTP accelerator

### BEGIN INIT INFO
# Provides:          varnish
# Required-Start:    $local_fs $network
# Required-Stop:     $local_fs $network
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start HTTP accelerator
# Description:       This script provides a server-side cache
#                    to be run in front of a httpd and should
#                    listen on port 80 on a properly configured
#                    system
### END INIT INFO
```

```
# Source function library
. /lib/lsb/init-functions

NAME=varnishd
DESC="HTTP accelerator"
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin
DAEMON=/usr/local/sbin/varnishd
PIDFILE=/var/run/$NAME.pid

test -x $DAEMON || exit 0

# Include varnish defaults if available
if [ -f /etc/default/varnish ] ; then
        . /etc/default/varnish
fi

# Open files (usually 1024, which is way too small for varnish)
ulimit -n ${NFILES:-131072}

# If $DAEMON_OPTS is not set at all in /etc/default/varnish, use minimal useful
# defaults (Backend at localhost:8080, a common place to put a locally
# installed application server.)
DAEMON_OPTS=${DAEMON_OPTS:--b localhost}

case "$1" in
       start)
              output=$(/bin/tempfile -s.varnish)
              log_daemon_msg "Starting $DESC"
              log_progress_msg $NAME
                 if start-stop-daemon \
                 --start --quiet --pidfile ${PIDFILE} --exec ${DAEMON} -- \
                    -P ${PIDFILE} ${DAEMON_OPTS} > ${output} 2>&1; then
                     log_end_msg 0
              else
                     log_end_msg 1
                     cat $output
                     exit 1
              fi
              rm $output
              ;;
       stop)
```

```
            log_daemon_msg "Stopping $DESC"
            log_progress_msg $NAME
            if start-stop-daemon \
               --stop --quiet --pidfile $PIDFILE --retry 10 \
               --exec $DAEMON; then
                  log_end_msg 0
            else
                  log_end_msg 1
            fi
            ;;
      restart|force-reload)
            $0 stop
            $0 start
            ;;
      *)
            log_success_msg "Usage: $0 {start|stop|restart|force-reload}"
            exit 1
        ;;
esac

exit 0
```

Apache configuration

To be able to log the actual users in apachelog a special logformat needs to be created in apache. Apache is set up to listen on port 6081, which Varnish makes the backend requests to.

Example config file /etc/apache2/conf.d/apache2.append.conf

```
LogFormat "%{X-Forwarded-For}i %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" varnishcombined
Listen 6081
NameVirtualHost 127.0.0.1:6081
```

Example virtualhost config file /etc/apache/sites-available/example.com

```
<VirtualHost 127.0.0.1:6081>
    KeepAlive Off
    CustomLog /var/log/apache2/www.hit.no.access.log varnishcombined
    ServerName example.com
    DocumentRoot /example.com
# ...more vhost config directives ...
</Virtualhost>
```

eZ publish configuration

Some configuration of eZ publish must be done to make it behave the right way together with Varnish. Example.com uses frontend editing, and a cookie «is_logged_in» is created by eZ publish and checked by Varnish to identify if a user/editor is logged in. Logged-in sessions are not cached by Varnish (see /etc/varnish/vcl.conf).

Patched ezpublish/index.php with pubsvn.ez.no nextgen rev. 21377

(this will be included standard in ez publish 4.1)

```
        if ( isset( $moduleResult["external_css"] ) )
            $use_external_css = $moduleResult["external_css"];
    }
}
// added rev. 21377 from nextgen: trunk/index.php
 $currentUser = eZUser::currentUser();
       $ini = eZINI::instance();


       $wwwDir = eZSys::wwwDir();
       // On host based site accesses this can be empty, causing the cookie to be set
for the current dir,
       // but we want it to be set for the whole eZ publish site
       $cookiePath = $wwwDir != '' ? $wwwDir : '/';


       if ( $currentUser->isLoggedIn() )
       {
           setcookie( 'is_logged_in', 'true', 0, $cookiePath );
           header( 'Etag: ' . $currentUser->attribute( 'contentobject_id' ) );
       }
       else
       {
           setcookie( 'is_logged_in', false, 0, $cookiePath );
       }
// END rev. 21377.



if ( $module->exitStatus() == eZModule::STATUS_REDIRECT )
```

Example settings/override/site.ini.append.php:

```
[HTTPHeaderSettings]
CustomHeader=enabled
# Cache-Control values are set directly
Cache-Control[]
```

```
Cache-Control[/]=
# Expires specifies time offset compared to current time
# Default expired 2 hours ago ( no caching )
Expires[]
Expires[/]=300
# Pragma values are set directly
Pragma[]
Pragma[/]=
```

Example pagelayout.tpl:

```
<html><head>
<meta http-equiv="Pragma" content="no-cache" />
<meta http-equiv="Cache-control" content="no-cache" />
...
</head><body>
...
</body></html>
```