Varnish 2.1.2 安装与配置

网海过客
Blog:http://www.chinasa.net
QQ 群: 78335077
2010-08-10

目录

Varnish 2.1.2 安装与配置 1
Varnish 下载 2
Varnish 安装 2
Varnish 配置实例 2
Varnish 启动与停止，动态加载配置文件 6
Varnish 日志启动与日志切割 6
Varnish 缓存清除 7
Varnish 配置介绍 8
定义后端服务器 IP 和端口 8
定义访问控制列表,充许那些 IP 清除 varnish 缓存 8
判断 host 请求针对那个后端服务器 8
不充许非访问控制列表的 IP 进行 varnish 缓存清除 8
清除 url 中有 jpg|png|gif 等文件的 cookie 9
取消服务器上 images 目录下所有文件的 cookie 9
WEB 服务器指明不缓存的内容，varnish 服务器不缓存 10
指定 fonts 目录不缓存 10
指定要缓存的静态文类型 10
使用正则表达式指定缓存的内容 10
添加在页面 head 头信息中查看缓存命中情况 11
根据访问 url 地址或者目录，转发到不同的后端服务器 11
定义组,负载均衡+后端 web 服务器健康检查 11
防止爬虫，网络蜘蛛访问 13

防盗连接 13
禁止某个目录或者某个 url 地址访问 14
Rewrite urls 配置 14
Vcl 优化 15
Varnish 参考资料 16

环境:Centos 5.4 Varnish 2.1.2

Varnish 下载

下载地址:http://sourceforge.net/projects/varnish/files/

Varnish 安装

```
#  tar zxvf varnish-2.1.2.tar.gz
#  cd varnish-2.1.2
#  ./configure –prefix=/opt/varnish
#  make
#  make install
```

Varnish 配置实例

通过前端 varnish 缓存服务器反向代理后端 www.bbs.com 和 www.bbs1.com 网站，要求对静态文件 js|css|jpg|gif 等文件进行缓存 7 天,对网页中加有 no-cache 头信息页面不缓存。配置文件如下:

vi /opt/varnish/etc/varnish/bbs.vcl

```
backend bbs {
   .host = "192.168.0.144";
   .port = "80";
}
backend bbs1 {
   .host = "192.168.0.155";
   .port = "80";
}

acl local {
        "localhost";
```

```
            "127.0.0.1";
}

sub vcl_recv {
        if (req.http.host ~ "^(www.)?bbs.com$") {
            set req.backend = bbs;
        }
        elsif (req.http.host ~ "^(www.)?bbs1.com$") {
            set req.backend = bbs1;
        }
         else {
              error 404 "Unknown HostName!";
        }
        if (req.request == "PURGE") {
            if (!client.ip ~ local) {
                error 405 "Not Allowed.";
                return (lookup);
                }
        }
        if (req.request == "GET" && req.url ~ "\.(jpg|png|gif|swf|jpeg|ico)$") {
            unset req.http.cookie;
        }
        if (req.http.x-forwarded-for) {
           set req.http.X-Forwarded-For =
           req.http.X-Forwarded-For ", " client.ip;
        } else {
             set req.http.X-Forwarded-For = client.ip;
        }
        if (req.request != "GET" &&
          req.request != "HEAD" &&
          req.request != "PUT" &&
          req.request != "POST" &&
          req.request != "TRACE" &&
          req.request != "OPTIONS" &&
          req.request != "DELETE") {
            return (pipe);
        }
        if (req.request != "GET" && req.request != "HEAD") {
            return (pass);
        }
        if (req.http.Authorization || req.http.Cookie) {
            return (pass);
        }
        if (req.request == "GET" && req.url ~ "\.(php)($|\?)") {
```

```
                return (pass);
               }
          return (lookup);
          }

sub vcl_pipe {
        return (pipe);
}

sub vcl_pass {
        return (pass);
}

sub vcl_hash {
        set req.hash += req.url;
        if (req.http.host) {
                set req.hash += req.http.host;
        } else {
                set req.hash += server.ip;
        }
        return (hash);
}

sub vcl_hit {
        if (!obj.cacheable) {
                return (pass);
        }
        return (deliver);
}

sub vcl_miss {
        return (fetch);
}

sub vcl_fetch {
        if (!beresp.cacheable) {
                return (pass);
        }
        if (beresp.http.Set-Cookie) {
                return (pass);
        }
        if (beresp.http.Pragma ~ "no-cache" ||
    beresp.http.Cache-Control ~ "no-cache" ||
    beresp.http.Cache-Control ~ "private") {
```

```
        return (pass);
    }

        if (req.request == "GET" && req.url ~ "\.(js|css|mp3|jpg|png|gif|swf|jpeg|ico)$")
        {
                set beresp.ttl = 7d;
        }

        return (deliver);
}

sub vcl_deliver {
                set resp.http.x-hits = obj.hits ;
                if (obj.hits > 0) {
                        set resp.http.X-Cache = "HIT cqtel-bbs";
                } else {
                        set resp.http.X-Cache = "MISS cqtel-bbs";
                }
}

sub vcl_error {
        set obj.http.Content-Type = "text/html; charset=utf-8";
        synthetic {"
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
   <head>
      <title>"} obj.status " " obj.response {"</title>
   </head>
   <body>
      <h1>Error "} obj.status " " obj.response {"</h1>
      <p>"} obj.response {"</p>
      <h3>Guru Meditation:</h3>
      <p>XID: "} req.xid {"</p>
      <hr>
      <address>
          <a href="http://www.bbs.com/">bbs cache server</a>
      </address>
   </body>
</html>
"};
        return (deliver);
}
```

Varnish 启动与停止，动态加载配置文件

创建 www 用户

```
#    useradd www
```

启动:

```
# /opt/varnish/sbin/varnishd -u www -g www -f /opt/varnish/etc/varnish/bbs.vcl -a 192.168.0.125:80 -s file,/data/varnish_cache/varnish_cache.data,3G -w 1024,51200,10 -t 3600 -T 192.168.0.125:3500
```

参数:

-u 以什么用运行
-g 以什么组运行
-f varnish 配置文件
-a 绑定 IP 和端口
-s varnish 缓存文件位置与大小
-w 最小，最大线程和超时时间
-T varnish 管理端口，主要用来清除缓存

停止:

```
#    pkill varnishd          #结束 varnishd 进程
```

动态加载配置文件

```
#    /opt/varnish/bin/varnishadm -T 192.168.0.125:3500
     vcl.load vcl-name_vcl "配置文件路径"        # vcl-name  这里可以任意取名
     vcl.use vcl-name
     vcl.show vcl-name                                  #显示 vcl-name 配置文件内容
```

Varnish 日志启动与日志切割

启动日志,方便分析网站访问情况。

```
#    /opt/varnish/bin/varnishncsa -w /opt/varnish/logs/varnish.log &
```

参数:
-w 指定 varnish 访问日志要写入的目录与文件

为了分析具体每一天的日志，要做 varnish 日志切割。

```
#    cat /data/shell/cutvlog.sh
```

脚本如下:

```
#!/bin/sh
vlog= /opt/varnish/logs/varnish.log
logs_path=/var/log/varnish-log
date=$(date -d "yesterday" +"%Y-%m-%d")
pkill -9 varnishncsa
mkdir -p /var/log/varnish-log
mkdir -p ${logs_path}/$(date -d "yesterday" +"%Y")/$(date -d "yesterday" +"%m")/
mv    /opt/varnish/logs/varnish.log  ${logs_path}/$(date  -d  "yesterday"  +"%Y")/$(date  -d
      "yesterday" +"%m")/varnish-${date}.log
/opt/varnish/bin/varnishncsa -w /opt/varnish/logs/varnish.log &
```

使用计划任务，每天晚上凌晨 00 点运行日志切割脚本。

```
#    crontab –e
0 0 * * * /data/shell/cutvlogg.sh
```

Varnish 缓存清除

```
#     /opt/varnish/bin/varnishadm   -T   192.168.0.144:3500   purge   "req.http.host   ~
      www.bbs.com$ && req.url ~ /static/image/tt.jsp"
```

说明：

192.168.0.144:3500	为被清除缓存服务器地址
www.bbs.com	为被清除的域名
/static/image/tt.jsp	为被清除的 url 地址列表

清除所有缓存

```
#    /opt/varnish/bin/varnishadm -T 127.0.0.1:3500 url.purge *$
```

清除 image 目录下所有缓存

```
#    /opt/varnish/bin/varnishadm -T 127.0.0.1:3500 url.purge /image/
```

Varnish 配置介绍

定义后端服务器 IP 和端口

```
backend bbs {                          #定义后端服务器名
   .host = "192.168.0.144";            #定义后端服务器 IP
   .port = "80";                       #定义后端服务器端口
}
```

定义访问控制列表,允许那些 IP 清除 varnish 缓存

```
acl local {
        "localhost";
        "127.0.0.1";
}
```

判断 host 请求针对那个后端服务器

```
sub vcl_recv {
      if (req.http.host ~ "^(www.)?bbs.com$") {
          set req.backend = bbs;
      }
      else {
          error 404 "Unknown HostName!";   #如果都不匹配，返回 404 错误
       }
}
```

不充许非访问控制列表的 IP 进行 varnish 缓存清除

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

清除 url 中有 jpg|png|gif 等文件的 cookie

```
if (req.request == "GET" && req.url ~ "\.(jpg|png|gif|swf|jpeg|ico)$") {
    unset req.http.cookie;
  }
```

取消服务器上 images 目录下所有文件的 cookie

```
sub vcl_recv {
if (req.url ~ "^/images") {
unset req.http.cookie;
    }
}
```

判断 req.http.X-Forwarded-For ，如果前端有多重反向代理，这样可以获取客户端 IP 地址。

```
if (req.http.x-forwarded-for) {
  set req.http.X-Forwarded-For =
  req.http.X-Forwarded-For ", " client.ip;
} else {
    set req.http.X-Forwarded-For = client.ip;
}
```

针对请求和 url 地址判断，是否在 varnish 缓存里查找

```
if (req.request != "GET" && req.request != "HEAD") {
    return (pass);
}   #   对非 GET|HEAD 请求的直接转发给后端服务器
if (req.request == "GET" && req.url ~ "\.(php)($|\?)") {
     return (pass);
   } #   对 GET 请求，且 url 里以.php 和.php?结尾的，直接转发给后端服务器
return (lookup); #   除了以上的访问以外，都在 varnish 缓存里查找
}
```

WEB 服务器指明不缓存的内容，varnish 服务器不缓存

```
if (beresp.http.Pragma ~ "no-cache" ||
   beresp.http.Cache-Control ~ "no-cache" ||
   beresp.http.Cache-Control ~ "private") {
      return (pass);
   }
```

指定 fonts 目录不缓存

```
if (req.url ~ "^/fonts/") {
      return (pass);
   }
```

指定要缓存的静态文类型

让 varnish 服务器缓存

```
if (req.request == "GET" && req.url ~ "\.(js|css|mp3|jpg|png|gif|swf|jpeg|ico)$") {
      set beresp.ttl = 7d;    #    缓存内容时间  d 表示天  s  表示秒
   }
```

使用正则表达式指定缓存的内容

```
if (req.request == "GET" && req.url ~ "\/[0-9]\.htm$") {
    set beresp.ttl = 300s;
}
```

添加在页面 head 头信息中查看缓存命中情况

```
sub vcl_deliver {
                    set resp.http.x-hits = obj.hits ;
                    if (obj.hits > 0) {
                            set resp.http.X-Cache = "HIT cqtel-bbs";
                    } else {
                            set resp.http.X-Cache = "MISS cqtel-bbs";
                    }
        }
```

根据访问 url 地址或者目录，转发到不同的后端服务器

可减少后端 web 服务器压力。如，把 url 地址中有 imgcache 目录的连接，转发到 192.168.0.155 服务器。

```
backend bbs {
    .host = "192.168.0.144";
    .port = "80";
}
backend imgcache {
            .host = "192.168.0.155";
            .port = "80";
}
sub vcl_recv {
         if (req.http.host ~ "^(www.)?bbs.com$" && req.url ~ "^/imgcache/") {
            set req.backend = imgcache;
         }
        elsif (req.http.host ~ "^(www.)?bbs.com") {
            set req.backend = bbs;
        }
 }
```

定义组,负载均衡+后端 web 服务器健康检查

```
director lb_bbs random {
            .retries = 6;           #   尝试后端 6 次检测,判断是否健康
                    {
            .backend = bbs;
            .weight = 2;            # 后端服务器权重
```

```
                        }
                {
                 .backend = bbs1;
                 .weight = 2;
                 }
        }
        sub vcl_recv {
              if (req.http.host ~ "^www.test.com") {
                    set req.backend = lb_bbs;
                 }
               else {
                    error 404 "Unknown HostName!";
               }
```

推荐：定义组,负载均衡+后端 web 服务器健康检查

定义单个后端负载均衡+后端 web 服务器健康检查

```
backend bbs {
   .host = "192.168.0.144";
   .probe = {
              .url = "/";
              .interval = 5s;
              .timeout = 1 s;
              .window = 5;
              .threshold = 3;
     }
  }
backend imgcache {
    .host = "192.168.0.123";
    .probe = {
              .url = "/";
              .interval = 5s;
              .timeout = 1 s;
              .window = 5;
              .threshold = 3;
    }
 }

director lb_bbs random {
              {
              .backend = bbs;
              .weight = 2;
```

```
                        }
                {
                 .backend = imgcache;
                 .weight = 2;
                 }
}
        sub vcl_recv {
              if (req.http.host ~ "^www.test.com") {
                    set req.backend = lb_bbs;
               }
             else {
                    error 404 "Unknown HostName!";
             }
```

url \\哪个 url 需要 varnish 请求。
Interval \\检查的间隔时间
Timeout \\等待多长时间探针超时
Window \\varnish 将维持 5 个 sliding window 的结果
Threshold \\至少有 3 次.windows 检查是成功的，就宣告 backends 健康

注：本方法经测试，前端 varnish 只匹配后端 nginx 配置文件里的第一个虚拟主机。配置上不太灵活，不推荐使用。

防止爬虫，网络蜘蛛访问

可根据 user-agent 来判断是否访问。
例：禁止 baidu、google 网络蜘蛛访问

```
if (req.http.User-Agent ~ "(Baiduspider|google)") {
 error 405 "Not allowed.";
}
```

防盗连接

禁止 bbs.com 和 bbs1.com 域名连接

```
if (req.http.Referer ~    "(bbs\.com|bbs1\.com)") {
            error 405 "Not allowed.";
      }
```

例:禁止非 bbs.com|bbs1.com 域名进行连接,请求从写为 www.aa.com//static/image/logo.png 地址。如果请求匹配 bbs.com|bbs1.com 域名，刚按正常请求处理。

```
if (req.http.referer ~ "http://.*") {
        if ( !(req.http.referer ~ "(bbs\.com|bbs1\.com)")) {
                set req.http.host = "www.aa.com";
                set req.url = "/static/image/logo.png";
                }
                return (lookup);
        }

        else {
                return (pass);
            }
```

禁止某个目录或者某个 url 地址访问

例:禁止 www.bbs.com/images 目录访问
www.bbs.com/forum.php url 地址访问

```
if (req.http.host ~ "^(www.)?bbs.com$" && req.url ~ "^/images") {
        error 403;
 }
if (req.http.host ~ "^(www.)?bbs.com$" && req.url ~ "^/forum.php") {
        error 403;
 }
```

Rewrite urls 配置

例:将 www.bbs.com/abc.html 转向到 www.bbs.com/abc.php

```
if (req.http.host ~ "^(www.)?bbs.com$" && req.url ~ "^/abc.html") {
        set req.http.host = "www.bbs.com";
        set req.url = "/abc.php";
    }
```

Vcl 优化

Grace mode

如果您的服务每秒有数千万的点击率，那么这个队列是庞大的，没有用户喜欢等待服务器响应。为了使用过期的 cache 给用户提供服务，我们需要增加他们的 TTL，保存所有 cache 中的内容在 TTL 过期以后 30 分钟不删除，使用以下 VCL：

```
sub vcl_fetch {
  set beresp.grace = 30m;
  }
```

Varnish 还不会使用过期的目标给用户提供服务，所以我们需要配置以下代码，在 cache 过期后的 15 秒内，使用旧的内容提供服务：

```
sub vcl_recv {
    set req.grace = 15s;
}
```

注:此项功能在测试时没生效

Saint mode

有时候，服务器很古怪，他们发出随机错误，您需要通知 varnish 使用更加优雅的方式处理它，这种方式叫神圣模式（saint mode）。Saint mode 允许您抛弃一个后端服务器或者另一个尝试的后端服务器或者 cache 中服务陈旧的内容。让我们看看 VCL 中如何开启这个功能的：

```
sub vcl_fetch {
 if (beresp.status == 500) {
    set beresp.saintmode = 10s;
    restart;
 }
 set beresp.grace = 5m;
}
```

当我们 beresp.saintmode 10 秒，这个 url 的请求 varnish 将在 10 秒内不访问服务器。一个黑名单，或多或少。一个 restart 就会执行，如果您有其他的后端有能力提供这些服务器内容，varnish 会尝试他们。当您没有可用的后端服务器，varnish 将使用它过期的 cache 提供服务内容。

注:此项功能在测试时没生效

Varnish 参考资料

官方文档:
http://varnish-cache.org/docs/
参考文档:
http://blog.liuts.com/category/22/
http://blog.izhoufeng.com/posts/70.html
http://linuxguest.blog.51cto.com/195664/354889