Varnish Tips & Tricks, 2015 edition

ConFoo 2015
Montreal, Canada

Magnus Hagander
magnus@hagander.net

Magnus Hagander

- Redpill Linpro
  - Infrastructure services
  - PostgreSQL / Databases
- Varnish
  - Medium/large scale deployments
  - Reviews
  - 24/7 support services

Varnish

- High performance web cache
  - (yes, really)
- Most websites are slow
- Most websites are (partially) cacheable
  - If your solution is flexible enough

Varnish

- Cache static assets
  - You're slow at this
- Cache semi-dynamic assets
  - You're even slower at this
- Cache API calls
  - Thank you REST!

Varnish

- BSD License Open Source
  - Lead architect is a FreeBSD hacker...
- Proprietary enterprise version
  - Varnish Software AS
  - Adds some extra features
  - Comes with SLA support

Pick your version

- Varnish 4.0
  - "current stable"
  - Released April 2014
- Varnish 3.0
  - "old stable"
  - Released June 2011
- Don't use anything older!

Examples

- I'll use Varnish 3.0
  - typo warnings!

Varnish 4.0

- Fairly large re-architecture
- Native streaming
- Background (re)-fetch
  - De-couple frontend from backend
  - Grace on first miss
- Much better log interface

Installation source

- repo.varnish-cache.org
  - RHEL 5-7
  - Debian (squeeze, wheezy, jessie)
  - Ubuntu (lucid, precise, trusty)
- Beware of distro built-in
  - (versions)

How much memory?

- More!
  - Always more :)
- Website hotspot
  - Often surprisingly small
  - Iterative process
- Working memory

Storage type

- Fits in RAM: malloc
- Does not fit in RAM: file
  - Or buy more RAM!
- Never: persistent

Configuring varnish

- VCL
  - Configuration is code
- Replace without restart
  - No cache-loss
  - No connection-loss

VCL

- "Unlimited" flexibility
- vmod's
- Inline-C
  - last resort

VCL

- Can make all decisions in VCL
  - Cache: yes/no
  - Cache-time
  - etc
- Can override everything from backend
  - "Yes, I will cache this"
  - Deal with uncooperative backends

VCL best practices

- Avoid overrides
- Let backend dictate rules
  - If possible!
- Easier maintenance
- Knowledge in the right place

Cache time from backend

- Include explicit cache info

```
HTTP/1.0 200 OK
Cache-Control: maxage=60
Content-Type: text/html; charset=utf-8
```

Cache time from backend

- Separate client and server cache time
  - Since we can forcibly expire from server

```
HTTP/1.0 200 OK
Cache-Control: s-maxage=600, maxage=30
Content-Type: text/html; charset=utf-8
```

Cache prevention from backend

- Have backend tag uncacheable pages
  - So we can cache by default
- Watch out for default CMS values!

```
HTTP/1.0 200 OK
Cache-Control: no-cache
Content-Type: text/html; charset=utf-8
```

Beware of Vary

- Instructs browser/cache to store separate copies
- "Acceptable" use:
  - Accept-Encoding, Accept-Language
- "Bad" use:
  - User-Agent, Cookie, *

```
HTTP/1.0 200 OK
Vary: User-Agent
Content-Type: text/html; charset=utf-8
```

Override in VCL

- Always better to set in backend
- Sometimes not possible
  - CMS
  - Frameworks
  - etc..
- Only then, override in vcl

Overrides in vcl

```
sub vcl_fetch {
   unset beresp.http.vary;
   unset beresp.http.cache-control;
   set beresp.ttl = 1h;
}
```

VCL best practices

- Let default code run
  - In most cases
  - Only explicitly return when necessary
- Modify incoming request/response instead

Overrides in vcl

```
sub vcl_fetch {
   if (beresp.http.cache-control ~ "no-cache") {
     unset beresp.http.cache-control;
     set beresp.ttl = 1h;
   }
   if (req.url ~ "^/static/") {
     unset beresp.http.cache-control;
     set beresp.ttl = 4h;
   }
}
```

Cookies

- Cookies...

Cookies

- The cache-killer
- http protocol makes it harder
- Varnish can help clean up

Cookies vs http

- Cookies included on all requests
  - once set
- Even for static assets
  - Never cookie-dependent!

Cookies vs http

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

Cookies vs http

- Cookies still included over wire
  - Just not varnish -> backend
- Consider separate subdomain
  - Also increases browser parallelism
- Can point to same Varnish instance
  - Instead, normalize hostname!

Normalize hostname

- Transparent to browser
  - Not a redirect!

```
sub vcl_recv {
  if (req.http.host ~ "\.example\.com$") {
    set req.http.host = "example.com";
  }
```

Back to cookies

- Client-side cookies
  - Use local storage instead!
- Google Analytics
  - (or similar)
- Never used on backend
- Prevents caching by default

Client side cookies

- Edit away known ones
- See what's left...

```
sub vcl_recv {
  set req.http.Cookie = regsuball(req.http.Cookie, "(^|;\s*)(_[_a-z]+|has_js)=[^;]*", "");
  set req.http.Cookie = regsub(req.http.Cookie, "^;\s*", "");
  if (req.http.Cookie == "") {
     unset req.http.Cookie;
  }
}
```

Cookies from backend

- Any Set-Cookie will disable caching
  - Don't send them on all requests
  - Use cache-control to avoid caching!
- Set-cookie generate hit-for-pass

Cookies from backend

```
sub vcl_fetch {
  if (req.url ~ "^/static/") {
     unset beresp.http.set-cookie;
  }
}
```

Session cookies

- Don't generate session until you need it
  - Many CMSs generate on first visit
- Actively delete when user logs out!
  - Back to cached data!
- Disable caching or cache per user

Cache per user

- Keep one copy / user
- Significantly lower cache ratio
- Cache bloat!
- Limit to expensive pages!
- Set shorter cache-time!

Cache per user

```
sub vcl_recv {
   if (req.url ~ "^/expensive/" && req.http.cookie ~ "session=\d+") {
     set req.http.sessionid = regsub(req.http.cookie, "session=(\d+)", "\1");
   }
}
sub vcl_hash {
   if (req.http.sessionid) {
      hash_data(req.http.sessionid);
   }
}
```

Grace mode

Grace mode

- You should be using grace mode...

Grace mode

- Serve expired content
- Even when backend is down
  - Or just very slow
- Survive load spikes
- Hide downtime

Grace mode

- Each objects gets two timeouts
  - One how long to serve (beresp.ttl)
  - One how long to keep in cache (beresp.grace)

Grace mode - backend slow

- New request arrives
- Existing request to backend already in progress
- Intentionally serve stale content
  - Instead of waiting

Grace mode - backend down

- Request arrives when backend is down
- No point in asking for object from backend
- Intentionally serve stale content
  - Instead of "503 internal error"

Grace mode

- Requires backend health probes
  - Which you probably want anyway
- Poll backend at regular intervals
- Checks http status code
- Also used for load balancer

Grace mode

```
sub vcl_recv {
   if (req.backend.healthy) {
      set req.grace = 1m;
   } else {
      set req.grace = 24h;
   }
}
sub vcl_fetch {
   ...
   set beresp.grace = 12h;
}
```

API caching

- REST API's trivial to cache
  - Follows http standard
  - Don't use cookies
- Website JS apis included
  - Remember http cookie behavior
- Add required headers to hash

API caching

- SOAP
  - Don't even try...

API routing

- Varnish is an efficient http router
- Even without caching
- Often used as API router
- Match on any http header
  - Including sticky load balancing etc

Intelligent cache expiry

- Expired/remove on demand
- More can be cached
- Cache times can be longer

Simple cache expiry

- URL is known
- Send PURGE or similar request
- Simple and efficient

Intelligent cache expiry

- Expire based on any regexp
- Against any header element
  - E.g. URL
  - Or content type
  - Or custom header

Intelligent cache expiry

- Custom headers:

```
HTTP/1.1 200 OK
Cache-Control: s-maxage=14400
Content-Type: text/html; charset=utf-8
Date: Fri, 20 Feb 2015 16:14:54 GMT
Last-Modified: Wed, 18 Feb 2015 11:13:16 GMT
X-pgthread: :308480:
```

Intelligent cache expiry

```
sub vcl_deliver {
  remove resp.http.x-pgthread;
}

sub vcl_recv {
  if (req.url ~ "/varnish-purge" && client.ip ~ purge) {
    if (req.http.x-purge-thread) {
      ban("obj.http.x-pgthread ~ " + req.http.x-purge-thread)
    }
  }
}
```

Summary

Summary

- VCL is infinitely flexible
- Hopefully you won't need it!
  - KISS definitely applies!

Summary

- Varnish is Swiss army knife of http
- Not just caching!

Thank you!

Magnus Hagander
magnus@hagander.net
@magnushagander
http://www.hagander.net/talks/