Caching Rails with Varnish

Hank Beaver
www.rubyslacker.com

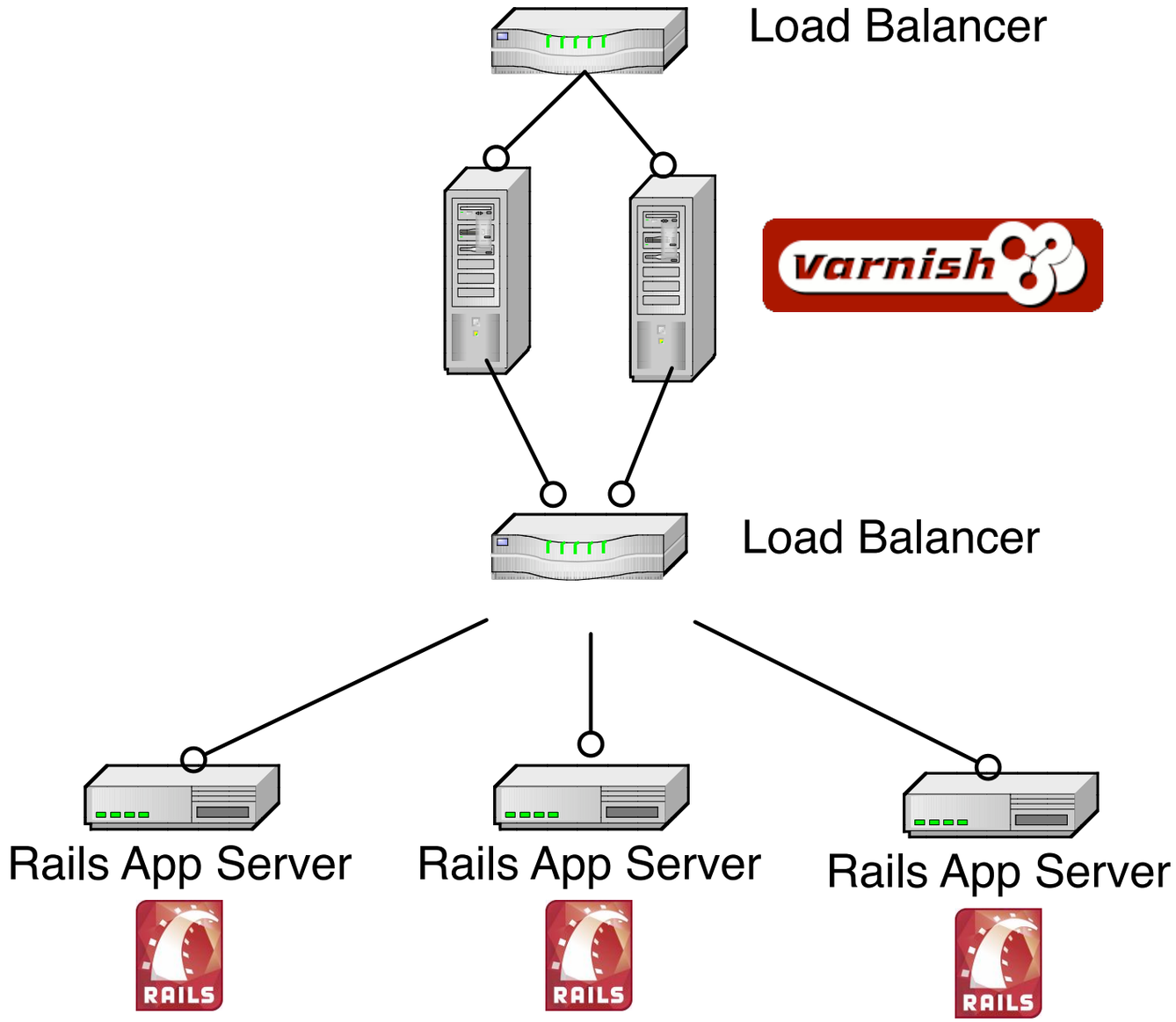
What is Varnish?

- "An accelerator for incoming traffic. We actually claim that Varnish is ten to twenty times faster than the popular Squid cache on the same kind of hardware!"
- It replaces page caching in front of Rails at network level.

Benefits of Varnish

- Regarding Rails caching: "Caching is ugly. You'll be polluting parts of your application code with code that has nothing to do with the business problem..."[1]
- Decouples your caching/acceleration from app.
- Can work concurrently with multiple apps.
- Supports ESI[5], so response can be served later by a content delivery network(CDN).

Example Topology



Installation & Setup

- Obtain source or binaries
- For OSX:
  1. Startup up with -F or launchd for newer versions.
  2. Remove IPv6 hosts entries. (next)

OSX IPv6 error

```
Only one address is allowed.
Please specify which exact address you want to
use, we found these:
    ::1
    fe80::1%lo0
    127.0.0.1
```

- Comment out in /etc/hosts:

```
::1                 localhost
fe80::1%lo0         localhost
```

HTTP Caching Headers, Cache-Control

- Used to specify directives that MUST be obeyed by all caching mechanisms along the request/response chain.[4]
- Example: "Cache-Control: no-cache"
- We use Cache-Control to control Varnish behavior.

Am I cached?

- "For a cache hit, X-Varnish will contain both the ID of the current request and the ID of the request that populated the cache. It makes debugging Varnish a lot easier."[3]
- `X-Varnish: 1825423056 1825423050`

Sample Rails app to accelerate

- Blog-like app
- Dynamic twitter search list
- Cookies set for current viewed post
- Dynamic datetime at header
- Backend CRUD

Baseline performance (no caching)

- ab -n 200 -c 20
- Rails: Requests per second: 1.30

VCL

- The Varnish language for caching rules and configuration
- "VCL programs are transparently compiled to executable machine code and therefore run at full CPU speed when executed"[2]
- Inline C is supported

VCL Routines

- vcl_recv()
  Called after a request is received from the browser, but before it is processed.
- vcl_fetch()
  Called when the request has been sent to the backend and a response has been received from the backend.
- vcl_error()

App VCL Rules

- No explicit URL patterns
- Uses Rails/app to determine cache-ability via Cache-Control, Varnish modifies.
- “max-age” determines object time to live.
- Only GET requests do lookup, not POST.
- Static assets have their cache buster removed in Rails response*

```
*ActionView::Helpers::AssetTagHelper.cache_asset_timestamps = false
```

Demonstration Caching Requirements

- Home Page caches for 120 seconds.
- Time header caches for 10 seconds.
- Personal Info caches 30 seconds. (not very private site)
- Last viewed post cookie never caches.
- CRUD pages are not cached.

How is it implemented in Rails?

- Let's look at some code.

cache-control method

```
class ApplicationController < ActionController::Base
...
  def cache_control(options = {})
      options[:type] ||= 'public'
      options[:ttl]  ||= 900   #low for testing
      headers['Cache-Control'] =
        "#{options[:type]},max-age=#{options[:ttl]}"
  end
```

```
class ContentController < ApplicationController
...
  before_filter do |controller|
       controller.cache_control({:ttl => 120})
  end
```

Setting cookies with AJAX

```
<!-- In content/home rendering ‘posts/view’ partial-->
<script>
  $.ajax({url:"/fragments/viewed_posts/<%=post.id
%>" ,cache:false});
</script>
```

```
class FragmentsController < ApplicationController
...
  def viewed_posts
    if params[:id]
            cookies[:viewed_posts] =
            {:value => "#{params[:id]}", :expires =>
1.week.from_now}
    end
    render :text => "You just viewed post: #{params[:id]} at
#{Time.now} "
  end
```

A cookie is set in response on viewed_post action!

ESI Include

- How to get a fragment of the page to be dynamic with ESI?
- `<esi:include src="/fragments/time" />`
- render_esi helper method[6]

```
def render_esi(path)
  if Rails.env == 'development'
    ...
    #Does an AJAX include here for dev mode!
  else
    '<esi:include src="' + path + '" />'
  end
end
```

Using ESI to assemble page

```
<!-- In main layout for content/home -->
...
<p class="pretty_time"><%=render_esi('/fragments/time')%></p>
...
```

```
<li><strong>The last post viewed:</strong> <span
id="my_cookie">Empty.</span></li>
...

<%= render_esi('/fragments/personal_info')%>
...

<script>
$(document).ready(function() {
        // set the cookie info in DOM using cookie object
           $('#my_cookie').html($.cookie('viewed_posts'));
});
</script>
```

App VCL Rules

- Let's examine VCL to make the aforementioned happen.
- Take a breath. More code.

Backend

```
backend default {
 .host = "127.0.0.1";
 .port = "3000";
}
```

vcl_recv

```
if (req.request != "GET") {
  pipe;
}
if (req.request == "GET") {
  remove req.http.cookie;
  remove req.http.authenticate;
  remove req.http.Etag;
  remove req.http.If-None-Match;
  lookup;
}
  pass;
```

vcl_fetch

```
if (obj.status >= 300) {
  pass;
}
if (obj.http.Cache-Control ~ "no-cache" ||
   obj.http.Cache-Control ~ "private") {
  pass;
}
if (obj.http.Cache-Control ~ "public") {
  unset obj.http.Set-Cookie;
  unset obj.http.Cache-Control;
  unset obj.http.Etag;
  set obj.http.Cache-Control = "no-cache"
  esi;
  deliver;
}
  pass;
```

Custom Cache-Hit Header

- Makes debugging even more clear
- Will add header “X-Cache” with value of “HIT” or “MISS”

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

Rule Testing

- Look at headers directly to Varnish vs. Rails
- Load balancers, CDN may strip non-standard HTTP headers.
- Remember to url.purge if in doubt
- Use “error” method in VCL to debug flow
- http://varnish.projects.linpro.no/attachment/wiki/VCLExamples/varnish_vcl.pdf?

The Guru!

- vcl_error
- Can customize this message.

Administration/Tools

- Telnet to Managment port (-T localhost: 6082)
- varnishadm
- varnishstat
- varnishlog

Re-Loading Rules

- Just restart Varnish
- vcl.use command*
- vcl.use will NOT complain about syntax errors. It quietly fails and DOES NOT reload changes. This is good.

```
vcl.load main_config \
  /Users/hbeaver/code/varnish_demo/config/app.vcl
vcl.use main_config
```

Purging Cache

- url.purge .*
- url.purge /content/home/1
- HTTP PURGE supported in VLC
- Cap task or Rake task

Varnish Performance

- ab -n 200 -c 20
- Varnish: Requests per second: **1822.76**
- **1400x** Faster for baseline app!

Gotchas

- POSIX Regex
- Single Quotes
- POSTs and 500s

Geeky Notes

- Supports rewrites
- Very basic load balancer(round robin and random)
- Architecture Notes. Read this: http://varnish.projects.linpro.no/wiki/ArchitectNotes

Thank You

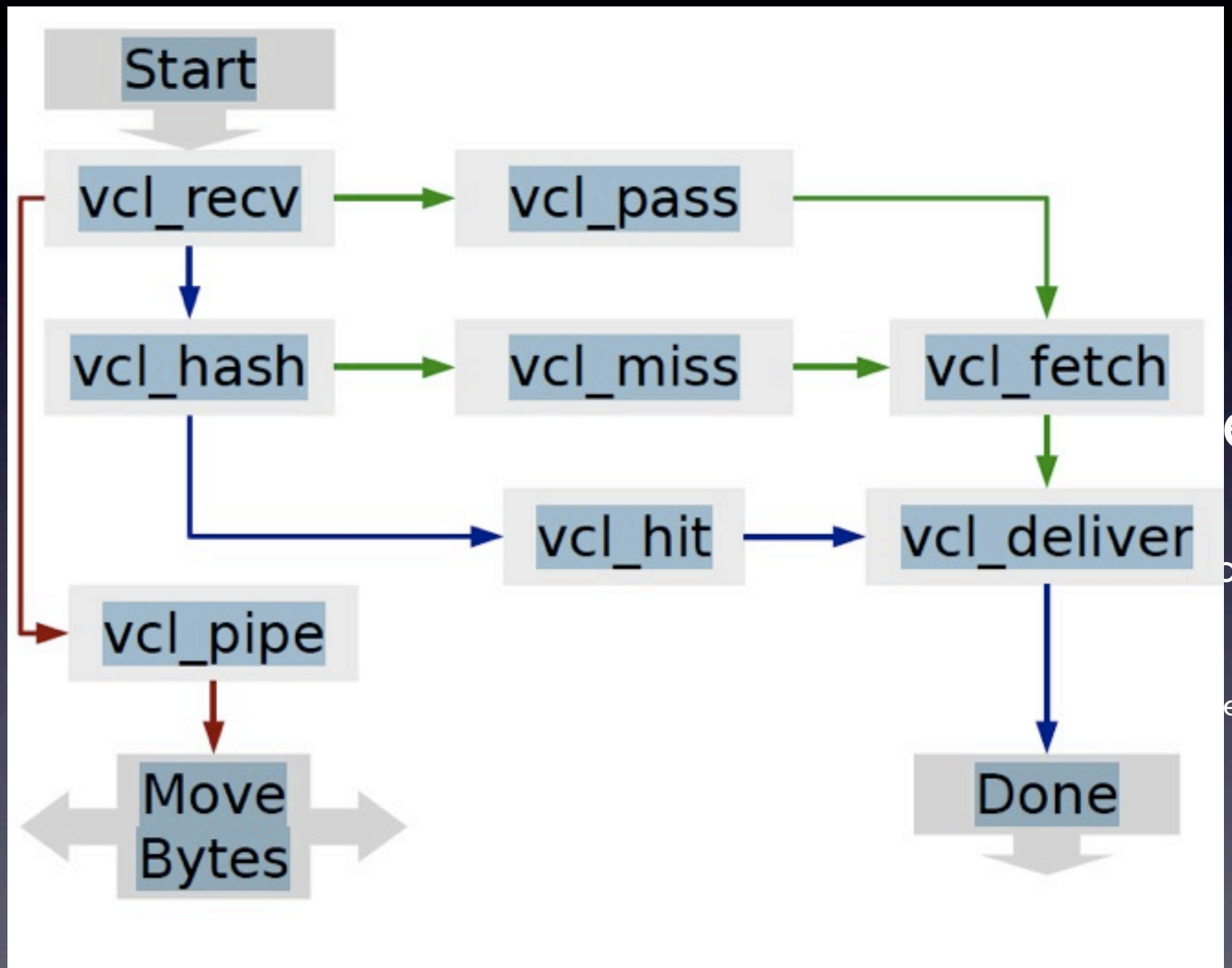
www.rubyslacker.com

http://github.com/blasterpal/varnish_rails_demo

References

- [1] Zygmuntowicz,E. & Tate, B. & and Begin C. (2008) Deploying Rails Applications
- [2] http://varnish.projects.linpro.no/browser/trunk/varnish-doc/en/handout/handout.pdf
- [3] http://varnish.projects.linpro.no/wiki/FAQ
- [4] http://en.wikipedia.org/wiki/List_of_HTTP_headers
- [5] http://en.wikipedia.org/wiki/Edge_Side_Includes
- [6] http://russ.github.com/2009/01/15/rails-varnish-and-esi.html

_VCL Flow



Source: http://varnish.projects.linpro.no/attachment/wiki/VCLExamples/varnish_vcl.pdf?format=raw