VUG7 Keynote:

Dude! Where's my Varnish 4.0 ?

Distractions
&
Challenges

Poul-Henning Kamp

phk@Varnish.org

@bsdphk

Things my great grandmother said:

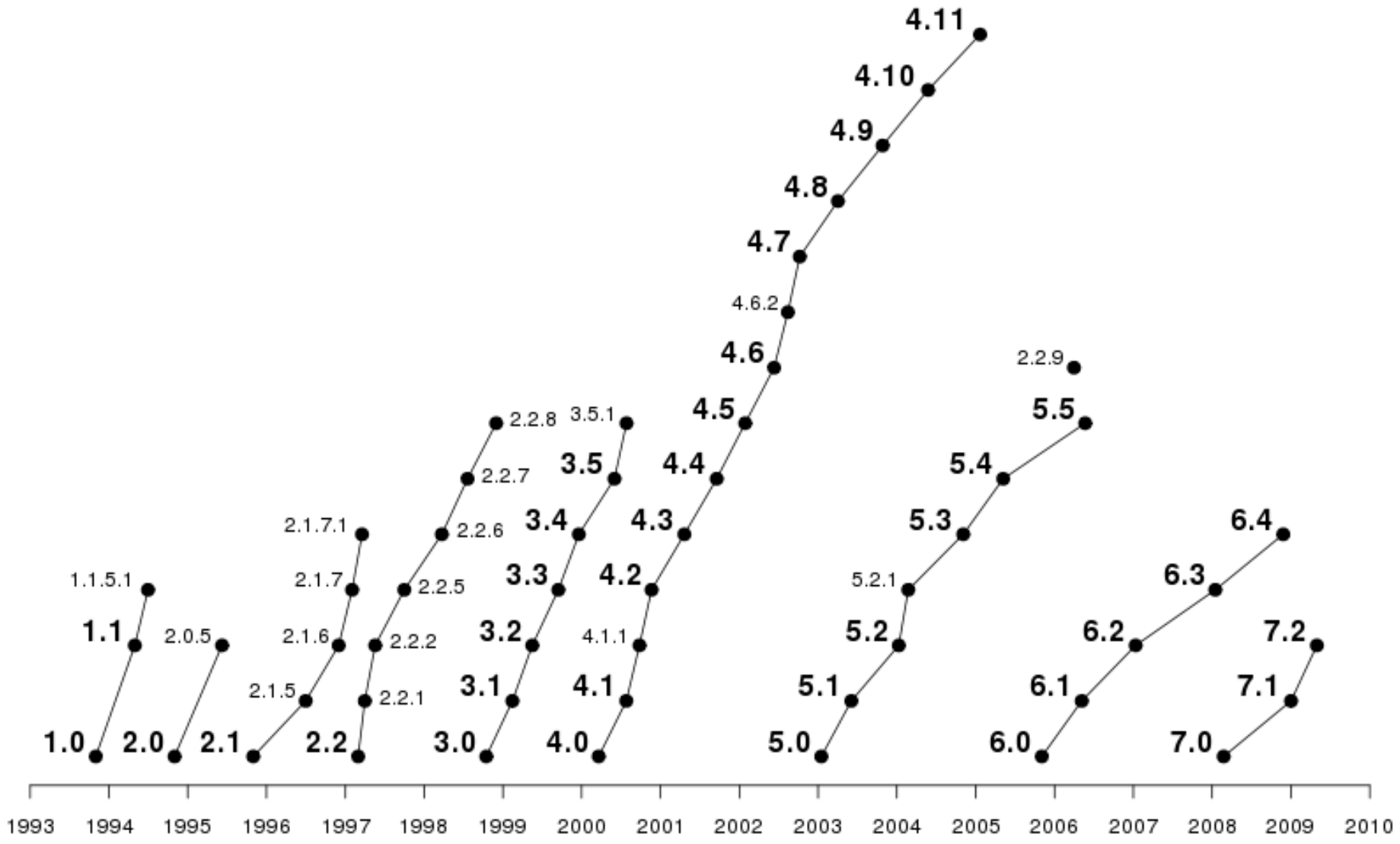
”Kaneføre og gæstebud ta’r man som det kommer”

Roughly translated:

Sleigh-weather and parties happen when they do


NTE SENER
PAPER ONLY


TIL SALG
2 store grundstykker
2.776 og 6.569 m²
Til boligformål
home ERHVERV
Tlf. 58 58 08 40


4.11
4.10
4.9
4.8
4.7
4.6.2
4.6
2.2.9
4.5
5.5
2.2.8
3.5.1
4.4
5.4
2.2.7
3.5
4.3
5.3
6.4
2.1.7.1
2.2.6
3.4
4.2
5.2.1
6.3
1.1.5.1
2.1.7
2.2.5
3.3
4.1.1
5.2
6.2
7.2
1.1
2.0.5
2.1.6
2.2.2
3.2
4.1
5.1
6.1
7.1
2.1.5
2.2.1
3.1
1.0
2.0
2.1
2.2
3.0
4.0
5.0
6.0
7.0
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010

Flag days vs. regular days

Things we can do in major releases:

- Change how Varnish works
- Change VCL syntax
- Change VMOD APIs
- Change important defaults

Things we can do in minor releases:

- Fix bugs
- Add VMODs
- Speed things up

Varnish 4: Changing assumptions

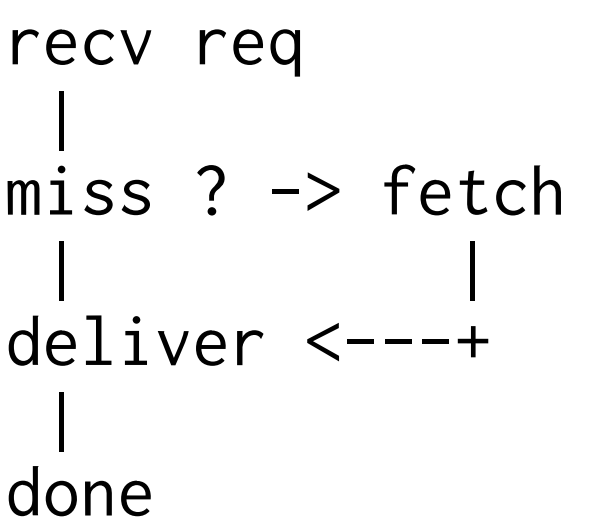
HTTP/1.0: Serialized communication

req — resp — req — resp - res — resp

HTTP/2.0: Multiplexed communication

req1+req2+req3 — resp2 — req4 — resp1,resp3,push1

Varnish model for HTTP/1.0:



Varnish model for HTTP/2.0:

?

HTTP/2.0 doesn't exist yet

And may not exist for some years

And may not be any good once it exists

Varnish model for HTTP/2.0:

”session thread” vs. ”state machine”
   - depends on protocol design.

Per request thread
   - Runs the VCL-obstacle course
   - Triggers fetches

Per fetch thread
   - Background / parallel / streaming

```
Req thread                          Fetch thread

Miss:

Allocate busyobj+objcore
vcl_miss{}
                  ------------>     Build bereq
                  <------------     release req
Deliver grace/                      vcl_backend_fetch{}
     or                             send request
     .                              rx response (hdrs)
     .                              vcl_backend_response{}
Wait for valid obj <------          mark ”valid”
Streaming delivery <------          rx body
                                    *done*
```

```
Req thread                          Fetch thread

Pass:

Allocate busyobj+objcore
vcl_pass{}
                  ------------>     Build bereq
                                    vcl_backend_fetch{}
                                    send request
                                    send req.body
      .           <------------     release req
      .                             rx response (hdrs)
      .                             vcl_backend_response{}
Wait for valid obj <------          mark ”valid”
Streaming delivery <------          rx body
                                    *done*
```

```
sub vcl_backend_fetch {
    return (fetch);
}
```

```
sub vcl_backend_response {
    if (beresp.ttl <= 0s ||
        beresp.http.Set-Cookie ||
        beresp.http.Surrogate-control ~ "no-store" |
        (!beresp.http.Surrogate-Control &&
          beresp.http.Cache-Control ~
            "no-cache|no-store|private") ||
        beresp.http.Vary == "*") {
                /*
                 * Mark as "Hit-For-Pass"
                 * for the next 2 minutes
                 */
                set beresp.ttl = 120 s;
                set beresp.do_pass = true;
    }
    return (deliver);
}
```

```
sub vcl_lookup {
    if (obj.uncacheable) {
        return (pass);
    }
    if (obj.ttl >= 0s) {
        return (deliver);
    }
    if (obj.ttl + obj.grace > 0s) {
        return (deliver_stale);
    }
    return (deliver);
}
```

VMOD objects

```
sub vcl_init {
    new foo = directors.random();
    foo.add_backend(s1, 1);
    foo.add_backend(s2, 1);
}

sub vcl_recv {
    set req.backend = foo.backend();
}
```

```
Object random() {
    Method VOID .add_backend(BACKEND, REAL)
    Method BACKEND .backend()
}
```

VCL context structure

We used to use ”struct req*” as the root pointer

(We'll always have a request, right ?)

With vcl_backend_(fetch|response){} we have a struct ”busyobj*” as root pointer.

So now we need to pass wrk*,req*,busyobj* and ...

Solve this with a new ”struct vrt_ctx” container

```
struct vrt_ctx {
        unsigned                magic;
#define VRT_CTX_MAGIC           0x6bb8f0db
        unsigned                method;
        unsigned                *handling;
        struct vsl_log          *vsl;
        struct VCL_conf         *vcl;
        struct ws               *ws;
        struct req              *req;
        struct http             *http_req;
        struct http             *http_obj;
        struct http             *http_resp;
        struct busyobj          *bo;
        struct http             *http_bereq;
        struct http             *http_beresp;
};
```

Varnish log filtering

```
param.show vsl_mask
200 373
vsl_mask    -VCL_trace,-WorkThread,-Hash
            Default is default
            Mask individual VSL messages from being
            logged.  default Set default value
            Use +/- prefixe in front of VSL tag name,
            to mask/unmask individual VSL messages.
param.set vsl_mask +vcl_trace,-expkill
200 0

param.show vsl_mask
200 371
vsl_mask    -ExpKill,-WorkThread,-Hash
[...]
```

Workspace changes

Used to be:

Requests have private workspace
Allocate all dynamic space on thread-stack

Including all space for backend-fetch

=> Even 99.99% hit rate will cause all threads
to drag around space for backend fetches.

Workspace changes

Changed to:

- Requests have private workspace
- Backend requests (”busyobj”) ditto
- Thread stack is used minimally

Why:

- N_threads > N_req >> N_busyobj
- = More threads use less memory

Purge been properly adopted

```
sub vcl_recv {
    if (req.method == ”PURGE”) {
        return (purge);
    }
}

sub vcl_purge {
    /* XXX: Could decide to fetch here */
    return (error(200, ”Purged”));
}
```

The new varnishlog/ncsa stuff

Overall idea: Tcpdump like filteringer language

Also: Track the req/fetch changes in varnishd

Ask Martin, not me :-)

Varnish penetration ?

varnish site:netcraft.com

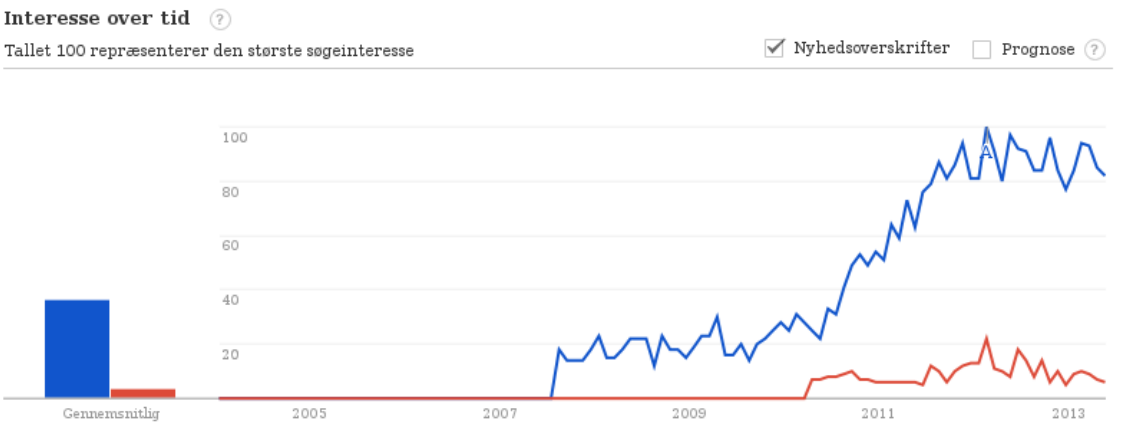
Internet Billeder Maps Indkø

Ca. 43.000 resultater (0,08 sekunder)

503 guru meditation xid

Internet Billeder Maps Ind

Ca. 62.100 resultater (0,12 sekunder)



http://trends.builtwith.com/:

Websites using Varnish

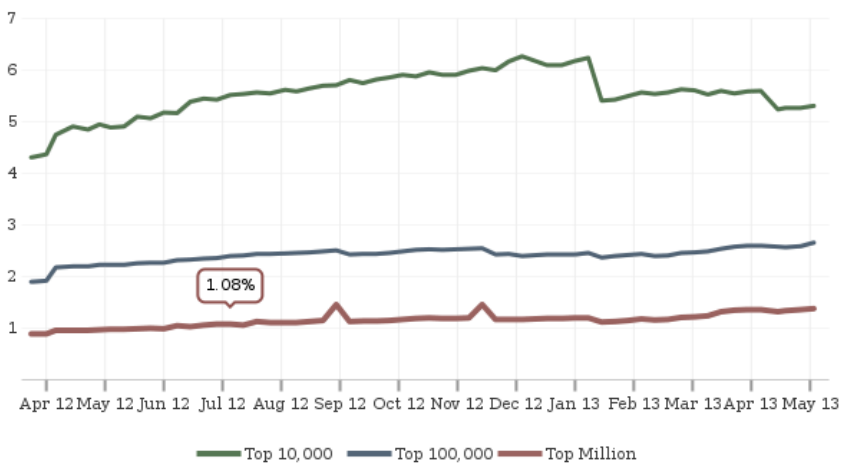
Get a list of websites using Varnish. We know of **471,877 websites using Varnish**.**99,030 websites** within the most visited sites on the internet and an additional **372,847** websites on the rest of the web.

Top sites using Varnish

Domain	SIRank
pinterest.com Full Profile	
answers.com Full Profile	
weather.com Full Profile	
go.com Full Profile	
examiner.com Full Profile	
nydailynews.com Full Profile	
merriam-webster.com Full Profile	
grindtv.com Full Profile	
people.com Full Profile	
usmagazine.com Full Profile	

Varnish Usage Trends

Varnish is a web accelerator / reverse proxy caching server.



VML: A four step programme

VML = Varnish Moral License

1 Happy users ask for VML

2 I send them an invoice

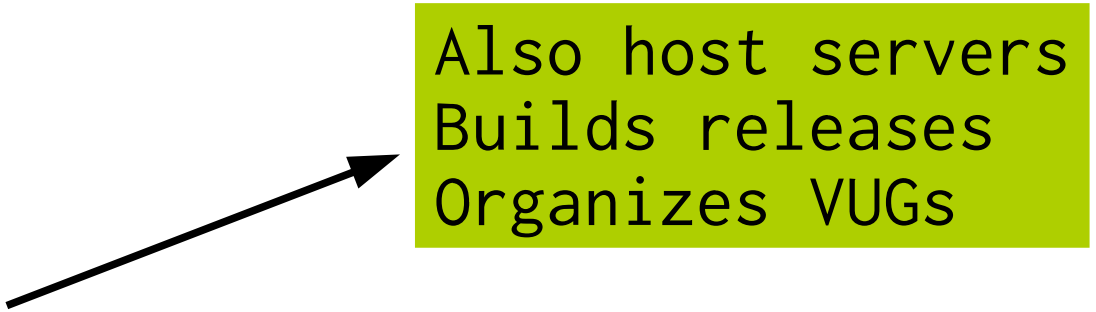
3 They pay the invoice

4 I develop more Varnish

VML Status so far:

2010: 900 hours
2011: 924 hours
2012: 586 hours
2013: 333 hours (jan-may)

Thank you!



* Varnish software
* GLOBO Communications
* UPLEX
* (You company could be here!)

VML going forward

Real-estate solves tax-limit (<33%)

Looking to ramp up Varnish share of work

Looking for more licenses:

    room for 6000-8000 EUR|USD pr. month

One-time/recurring: your choice

http://phk.freebsd.dk/VML/