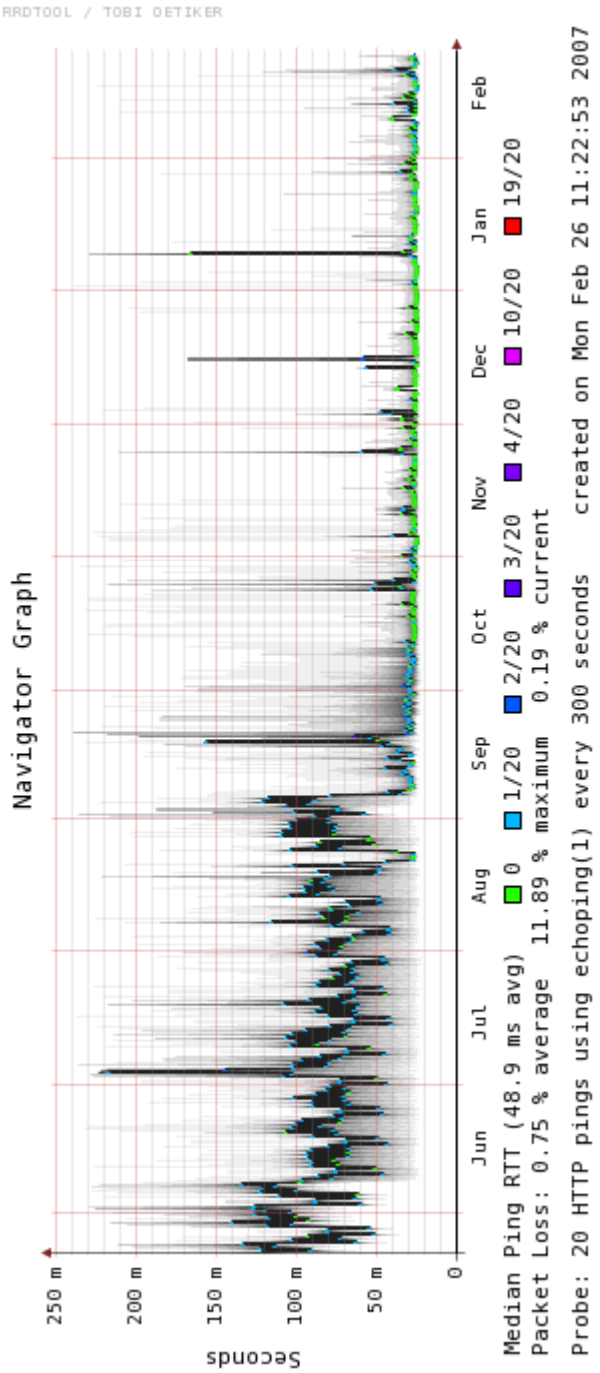
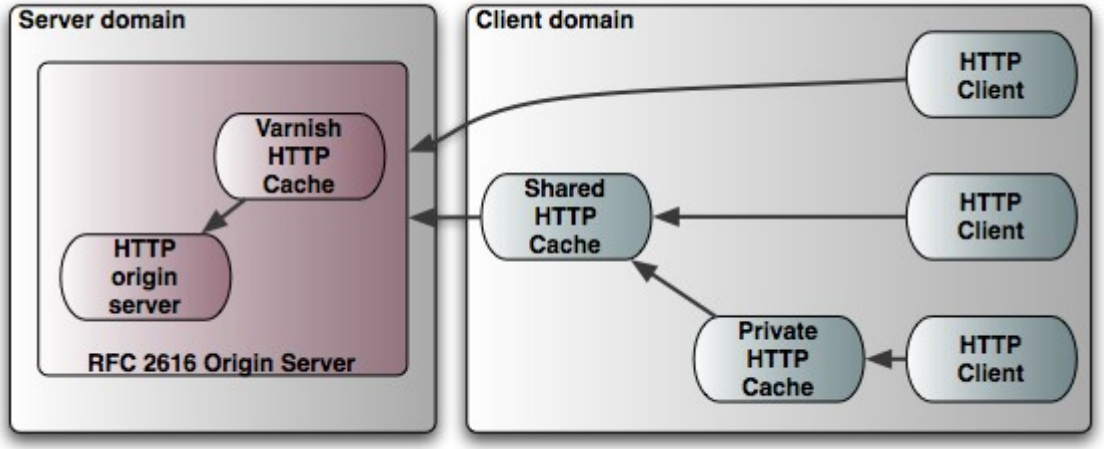
The Varnish Roadshow





Verdens Gang Presents

A Linpro Production

The Varnish Roadshow

starring

The Varnish HTTP accelerator

Designed and coded by: Poul-Henning Kamp

Project infrastructure: Dag-Erling Smørgrav

Based on an idea by: Anders Berg

Content management system

From Wikipedia, the free encyclopedia

Jump to: navigation, search

A content management system (CMS) is a computer software system used to assist its users in the process of content management. CMS facilitates the organization, control, and publication of a large body of documents and other content, such as images and multimedia resources. A CMS often facilitates the collaborative creation of documents. [...]

Suggested addition:

[...] CMS systems usually publish content via a terminally slow HTTP server, and therefore need HTTP acceleration.

RFC2616 cast list

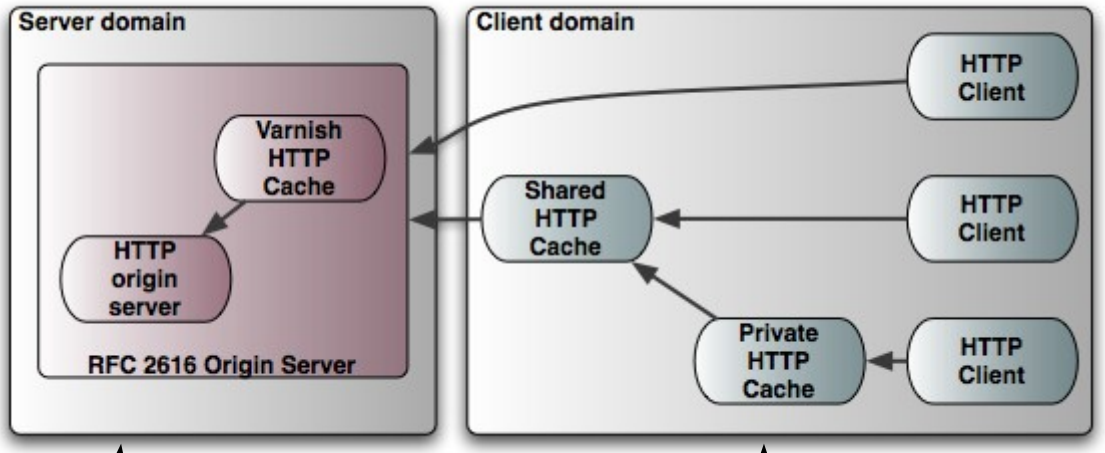
Web browsers

A HTTP Accelerator is not a HTTP cache.

Caching policy can be tailored to CMS system and site policies.

RFC2616 compliance as ”origin server”.

Can never cache per-user data



CMS system

Can sometimes cache per-user data

The website VG.no is one of Norways largest in terms of traffic.

Classical news site: rapidly changing contents in a slow CMS system.

12 Squid caches used as accelerators.

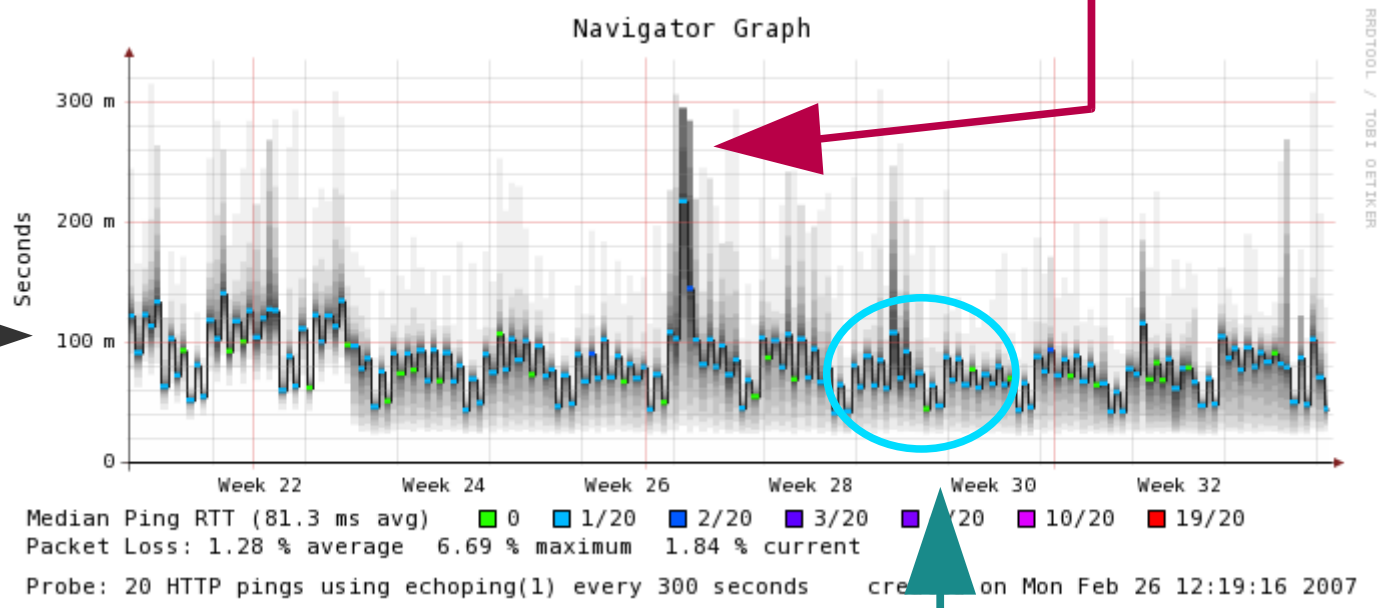
Unhappy with performance and stability.

VG NETT

Multimedia arm of Norvegian newspaper ”Verdens Gang”

Slow response

Terrible peak-handling

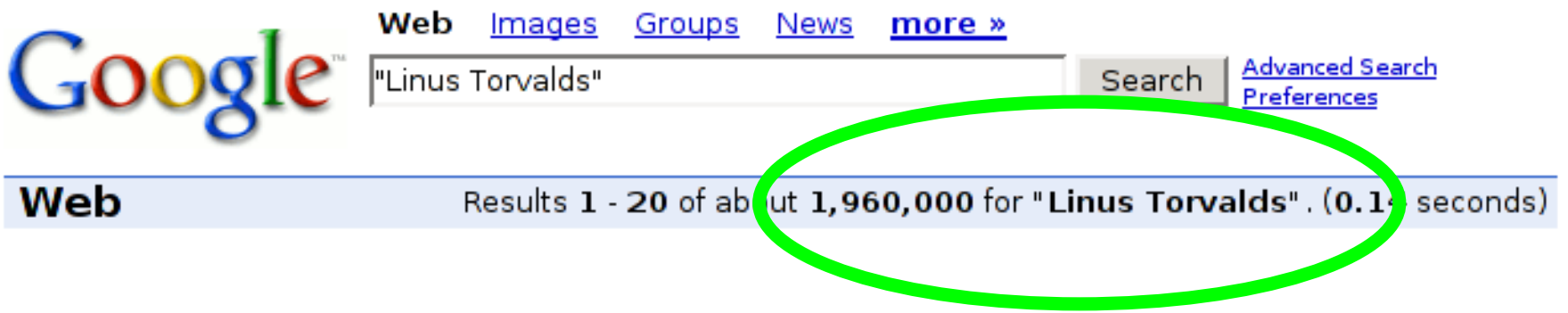


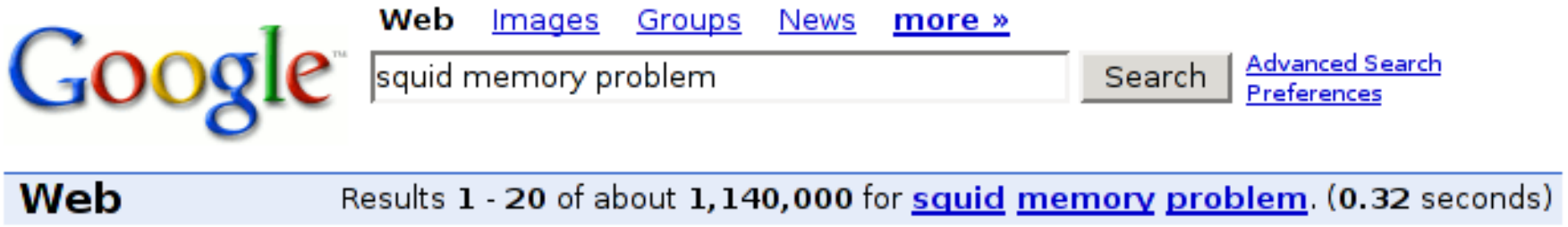
Significant loss

Obviously, using a client side cache for server acceleration is asking for trouble.

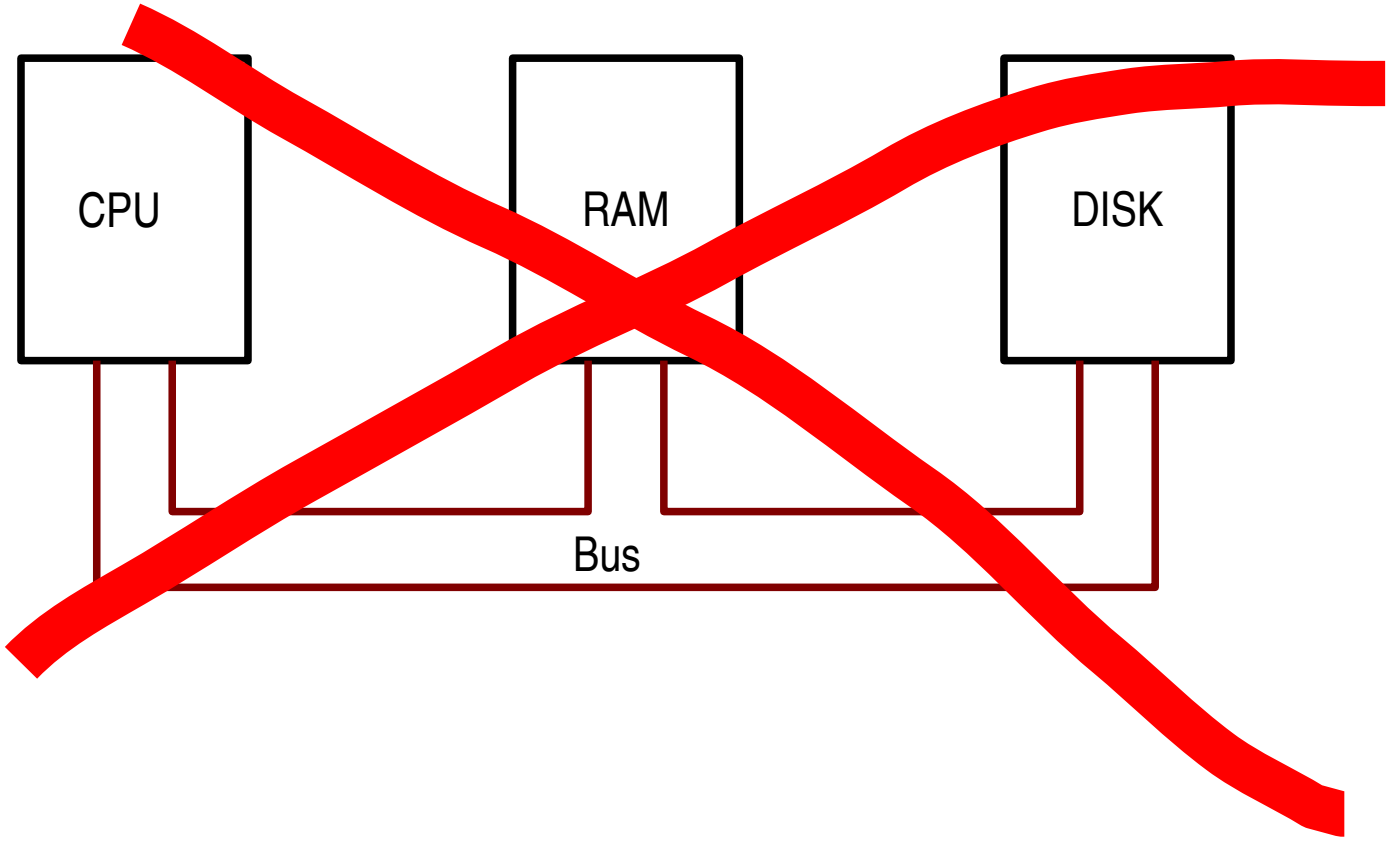
But there are no better OSS alternatives.

Besides, why **does** Squid suck so badly ?

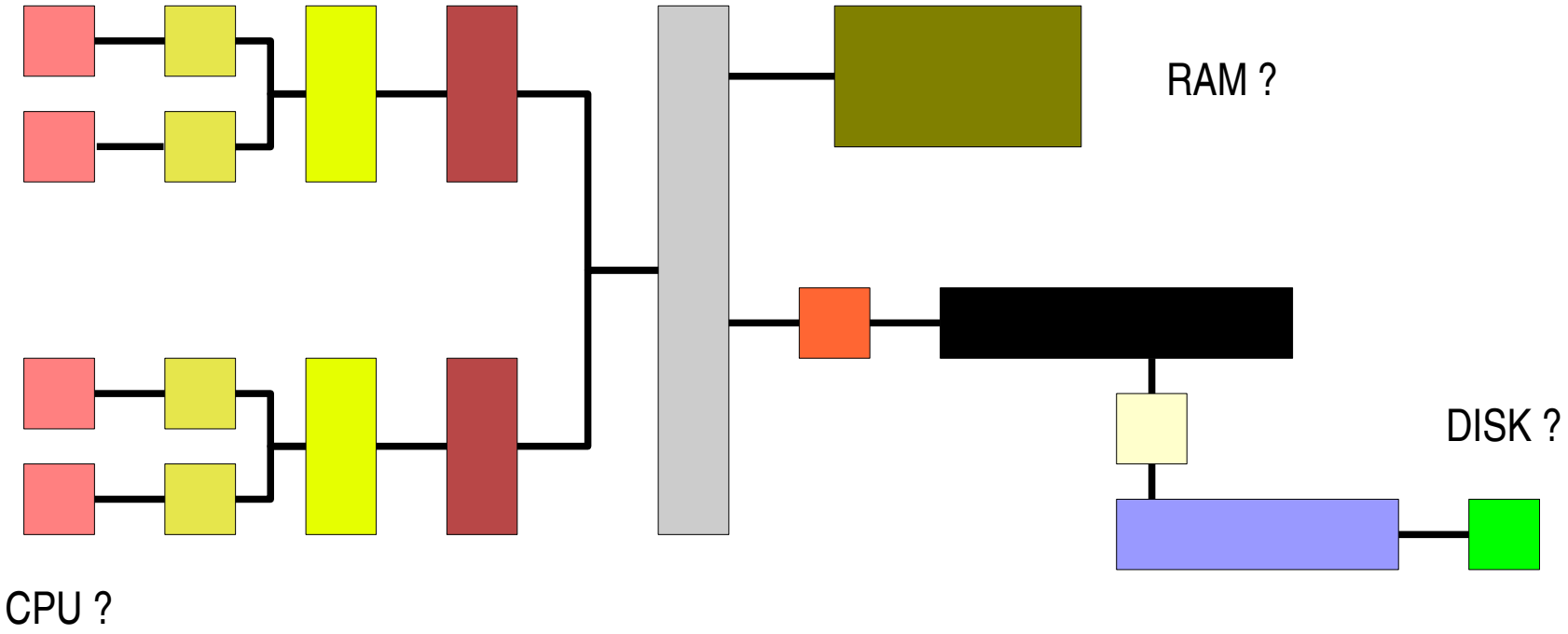

Google
Web
Images
Groups
News
more »
"Linus Torvalds"
Search
Advanced Search
Preferences
Web
Results 1 - 20 of about 1,960,000 for "Linus Torvalds". (0.1 seconds)



Squid is an antique software design



Not your dads computer anymore:



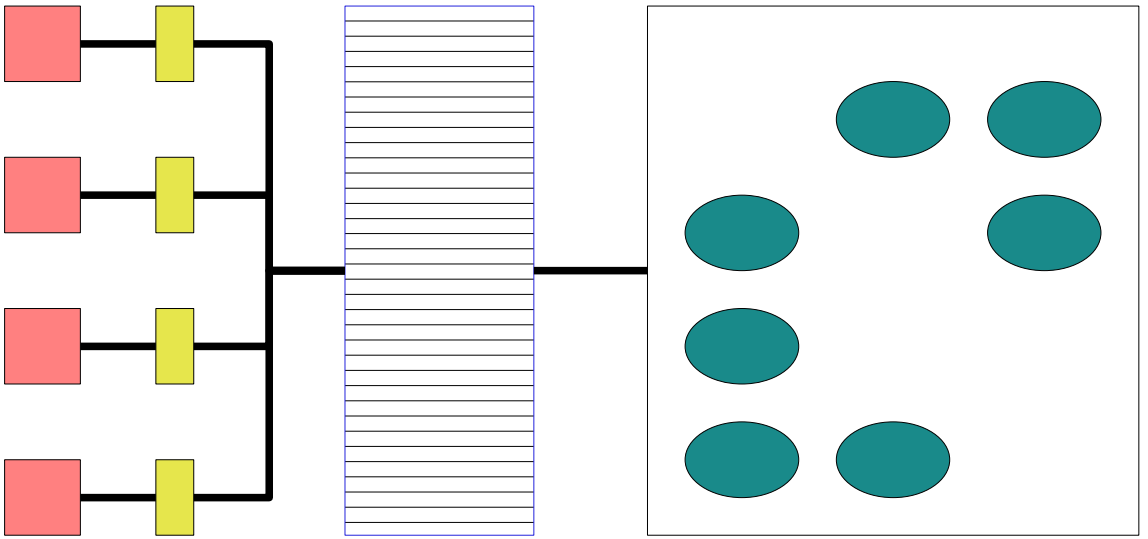
(and besides, we have this operating system which virtualizes all of it)

CPUs/Cores

Caches

The Virtual Page Cache

formerly known as "RAM"



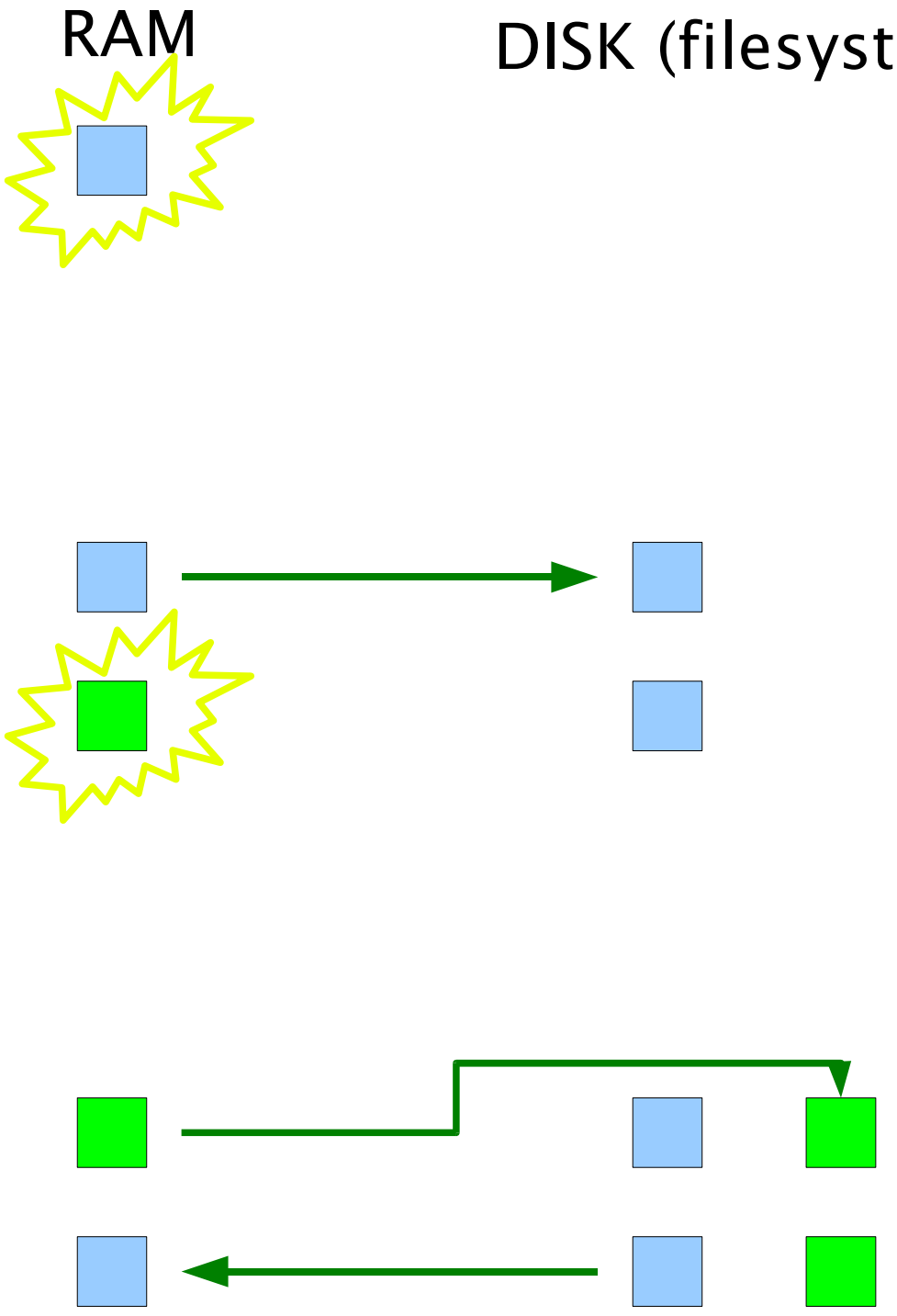
Objects,
possibly
on storage

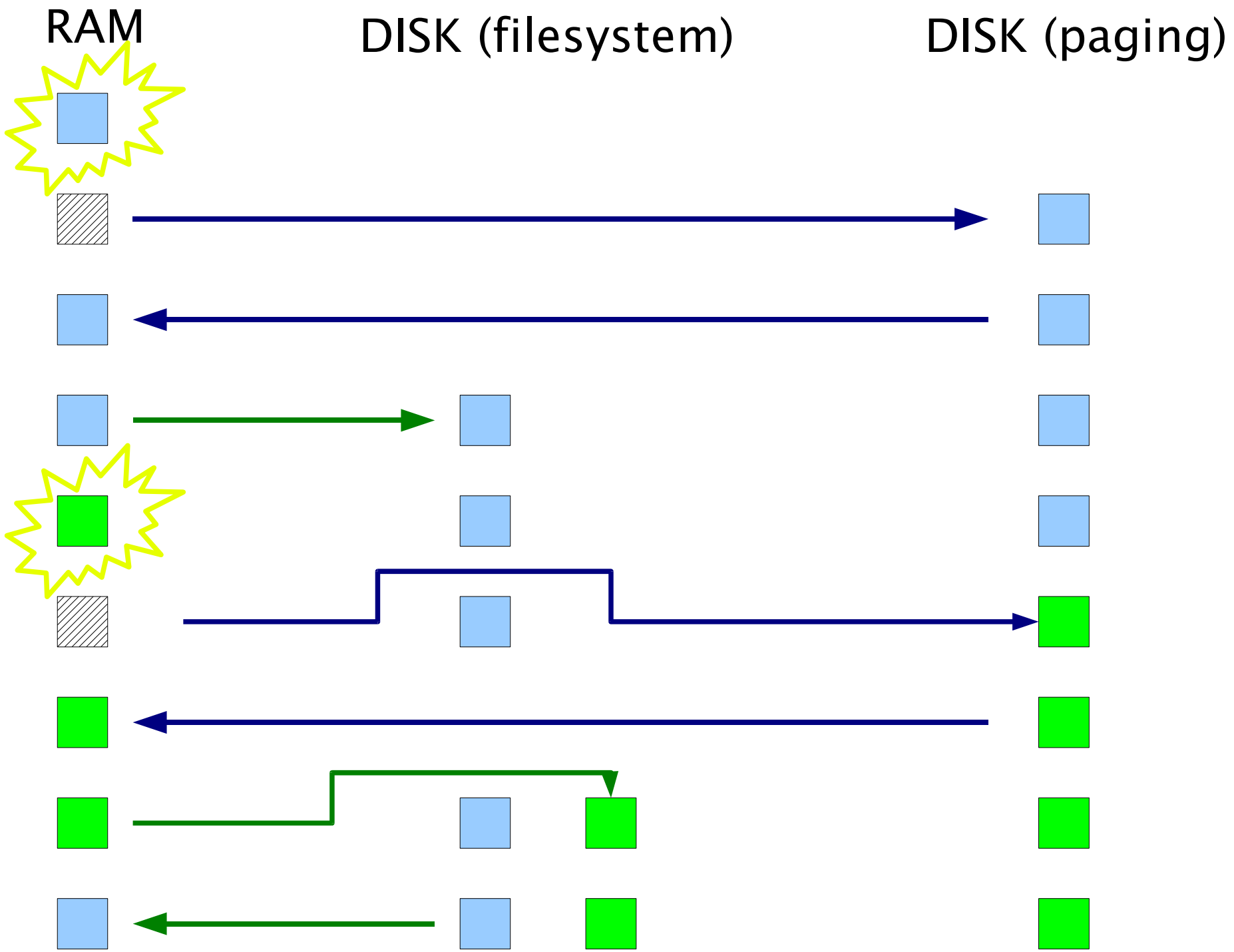
Squid moves data between RAM and disk.

Squid does not use virtual memory at all.

Kernel definitively schedules data between RAM and disk.

The fight multiplies disk-I/O by 2-5 times


RAM
DISK (filesystem)


RAM
DISK (filesystem)
DISK (paging)

Fighting the kernel from userland is plain stupid...

I know what you're thinking, proc:

'Did he send SIGHUP or SIGTERM ?'

Well, to tell you the truth, I've forgotten myself in all this excitement.

But being as this is SIGKILL, the most powerful signal in UNIX, and would blow your memory clean, you've got to ask yourself a question:

'Do I feel lucky?'

Well, do ya, proc?

Starting from scratch:

- Varnish is a HTTP accelerator only.
- Better configuration.
- Better management.
- (Much) faster.
- Content Management Features

What is it about configuration files ?

```
$ cat /etc/foobar.conf
# foobard configuration file
# copied from example.conf
#         /svend 19870104
# updated to new version
#         /knud 19941231
# various changes
#         /valdemar 19960523
# DON'T MESS WITH THIS!!!
```

```
HDXHSVVaCS=12

allocation_modulus=3.17⊕ℵ+∭f(21.4^μ•βe^-ij)

# overflow queue [default=7]
overflow_queue=7
```

```
process_backwards=if_acl_does_not

acl_set= { 1 2 3 4 6 7 21 }

invert_acls=odd

acl_reset = { 3 !8 }
```

VCL - Varnish Configuration Language

```
sub vcl_recv {
  if (req.request != "GET" && req.request != "HEAD") {
    pass;
  }
  if (req.http.Expect) {
    pipe;
  }
  if (req.http.Authenticate || req.http.Cookie) {
    pass;
  }
  lookup;
}
```

Why Yet Another Language ?

Why not simply use {PERL,Tcl,Python,Ruby...} ?

1. The programmers will not be programmers.
2. Speed.
3. Did you miss ”domain specific” ?

But you need to write a compiler ??

Without the hazzle of architecture fitting, (which registers, which instructions &c) a compiler is really just a text-processing app.

We compile VCL to C and use cc(1).

...in 3000 lines of code.

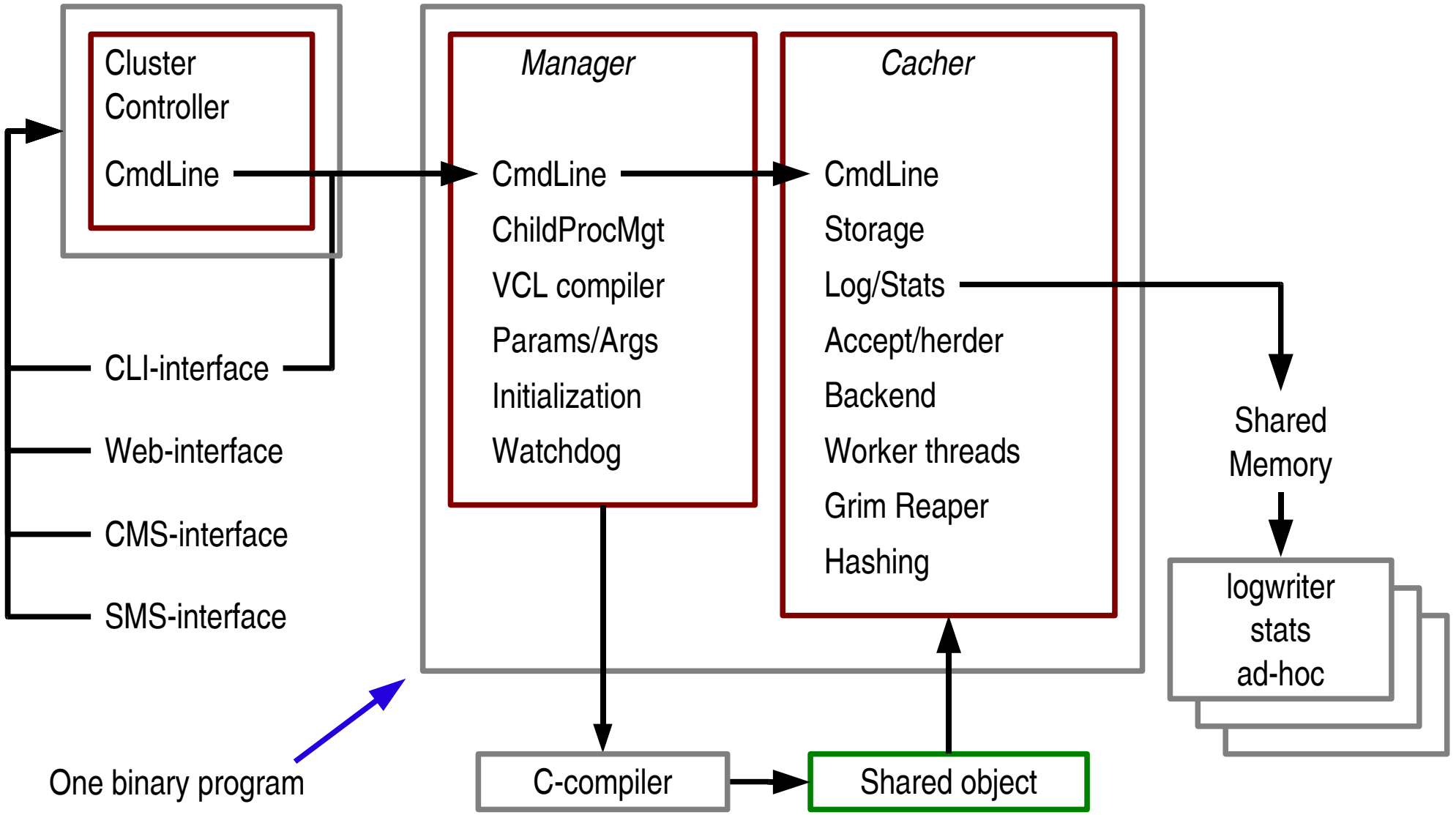
Managing Varnish

Command Line Interface for real-time control

Management/Worker process split:

- Manager can restart worker
- Allows priviledge separation
- Contains threading to worker process

Varnish architecture



The 02:45am CheatSheet:

```
# scp fromhost:varnishd .

# ./varnishd -b mywebserver -a :80

# ssh dnshost ndc reload
```

Performance and speed

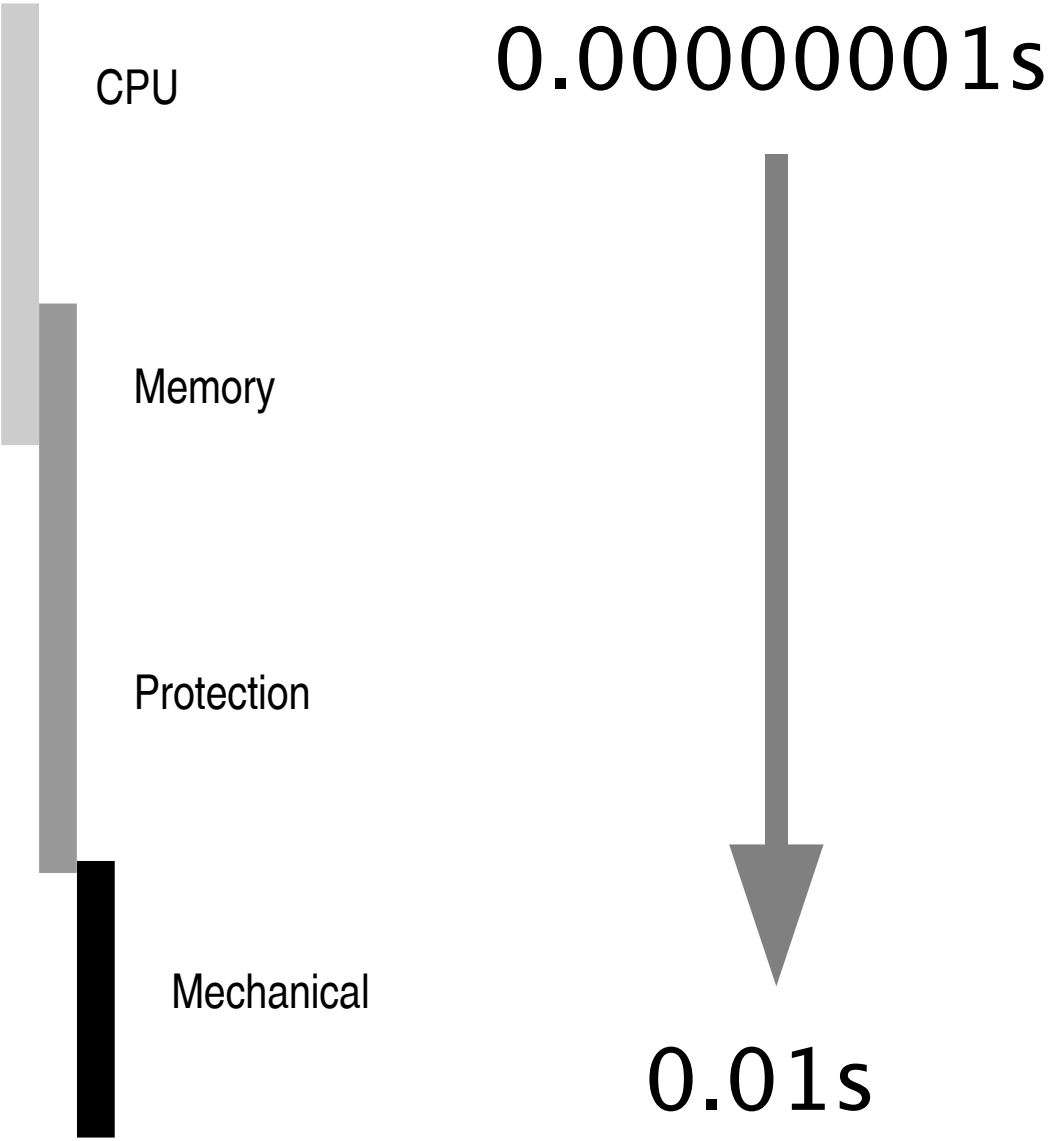
Program for performance

Use modern features:

- Virtual Memory
- sendfile(2), accept_filters(2), kqueue(2)
- (and every other trick in the book)

Performance Pricelist

- char *p += 5;
- strlen(p);
- memcpy(p, q, l);
- Locking
- System Call
- Context Switch
- Disk Access
- Filesystem operation



Classical logging is horribly expensive:

Filesystem operation, called once.

```
FILE *flog;

flog = fopen("/var/log/mylog", "a");

[...]

fprintf(flog, "%s Something went wrong with %s\n",
    timestamp(), foo2str(object));
fflush(flog);
```

Disk I/O, called 1 mio times

1 · 0.010 + 1,000,000 * .001 = 16 minutes

Logging to shared memory is almost free:

```
char *logp, *loge;

fd = open(...);
logp = mmap(..., size);
loge = logp + size;

[...]
logp[1] = LOG_ERROR;
logp[2] = sprintf(logp + 3,
    "Something went bad with %s", foo2str(obj));
logp[3 + logp[2]] = LOG_END;
logp[0] = LOG_ENTRY;
logp += 3 + logp[2];
```

Filesystem ops, called once.

Memory and arithmetic, 1 mio calls

2 · 0.010 + 1,000,000 * .00001 = 10 seconds

Varnishtop(1) – logfile ”top” program

What is my most popular URL ?

```
$ varnishtop -i rxurl
```

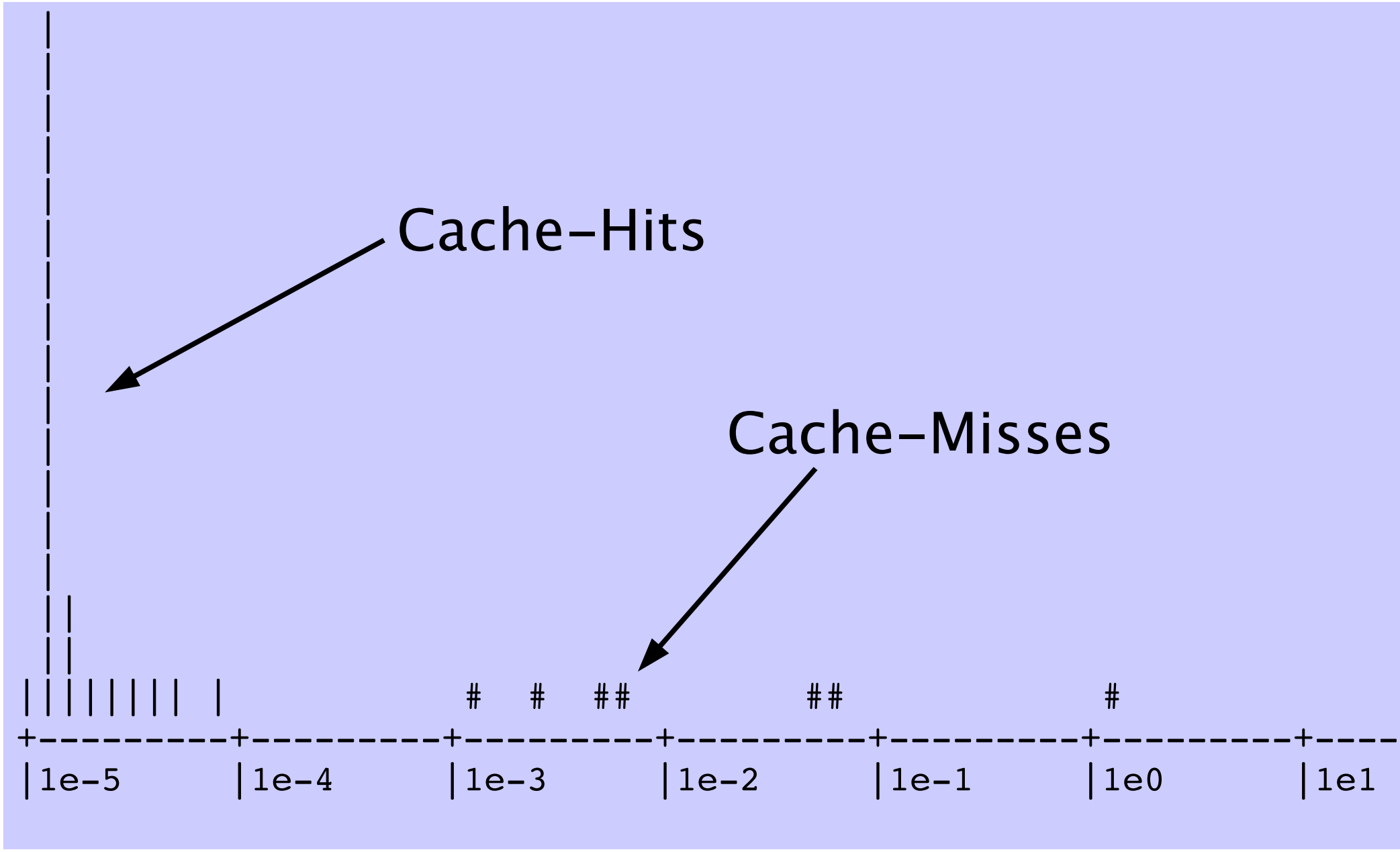
```
   1304.86 /tmv11.js
    989.08 /sistenytt.html
    495.05 /include/global/art.js
    491.01 /css/hoved.css
    490.05 /gfk/ann/n.gif
    480.08 /gfk/ann/ng.gif
    468.12 /gfk/front/tipsvg.png
    352.66 /css/ufront.css
    317.75 /t.gif
    306.79 /gfk/plu2.gif
    298.84 /css/front.css
    292.84 /gfk/min2.gif
    280.94 /css/blog.css
    279.84 /
```

Where does my traffic come from ?

```
$ varnishtop -i rxheader -I Referer
```

```
33913.74 Referer: http://www.vg.no/
 4730.72 Referer: http://vg.no/
  925.62 Referer: http://www.vg.no
  510.10 Referer: http://www.vg.no/pub/vgart.hbs?art
  434.37 Referer: http://www.vg.no/export/Transact/m
  349.55 Referer: http://www.vg.no/pub/vgart.hbs?art
  344.66 Referer: http://www.vg.no/pub/vgart.hbs?art
  324.06 Referer: http://www.vg.no/export/Transact/t
  297.25 Referer: http://www.nettby.no/user/
  263.82 Referer: http://www.vg.no/sport/fotball/
  242.55 Referer: http://www.vg.no/pub/vgart.hbs?art
```

Varnishhist(1) - Response-time histogram



Real-time statistics via shared memory

```
16:23:13
Hitrate ratio:        9        9        9
Hitrate avg:     0.9986   0.9986   0.9986

    17772105       435.55       301.26 Client connections accepted
   130213161      3623.22      2207.26 Client requests received
   129898315      3617.23      2201.93 Cache hits
       85043         0.00         1.44 Cache hits for pass
      227180         4.99         3.85 Cache misses
      313630         4.99         5.32 Backend connections initiated
         439         0.00         0.01 Backend connections recyles
          54         0.00         0.00 Backend connections unused
        6196         1.00         0.11 N struct srcaddr
        1656       -24.97         0.03 N active struct srcaddr
        3222         0.00         0.05 N struct sess_mem
        2258       -51.95         0.04 N struct sess
       65685         5.99         1.11 N struct object
       65686         5.99         1.11 N struct objecthead
```

CLI management

```
$ telnet localhost 81
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
param.show
200 675
default_ttl             120 [seconds]
thread_pools            5 [pools]
thread_pool_max         1500 [threads]
thread_pool_min         1 [threads]
thread_pool_timeout     120 [seconds]
overflow_max            100 [%]
http_workspace          8192 [bytes]
sess_timeout            5 [seconds]
pipe_timeout            60 [seconds]
send_timeout            600 [seconds]
auto_restart            on [bool]
[...]
```

CLI management

```
param.show overflow_max
200 330
overflow_max     100 [%]
                 Default is 100
                 Limit on overflow queue length in
                 percent of thread_pool_max parameter.

                 NB: We don't know yet if it is a good
                 idea to change this parameter.
                 Caution advised.
```

CLI management

```
help
200 281
Available commands:
ping [timestamp]
start
stop
stats
vcl.load <configname> <filename>
vcl.inline <configname> <quoted_VCLstring>
vcl.use <configname>
vcl.discard <configname>
vcl.list
param.show [-l] [<param>]
param.set <param> <value>
help [command]
url.purge <regexp>
dump.pool

[...]
```

”Minus-d-d debugging”

Daemons have traditionally taken '-d'
to mean ”run in foreground with debugging”.

Varnish allows you to send the process into
”daemon mode” when started with -d.

No need to stop & start again, once you're
happy with how it is running.

Content Management Features:

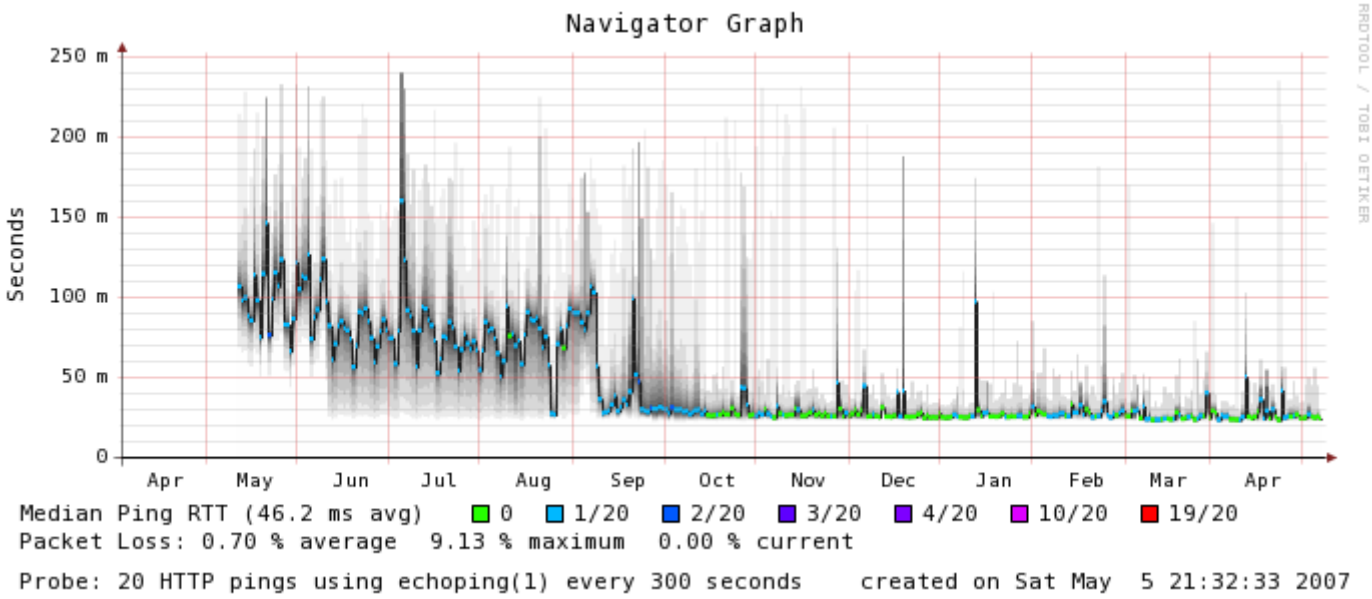
Instant purges (URL or regexp)

TTL/Caching policy control in VCL

Load/Situation mitigation in VCL

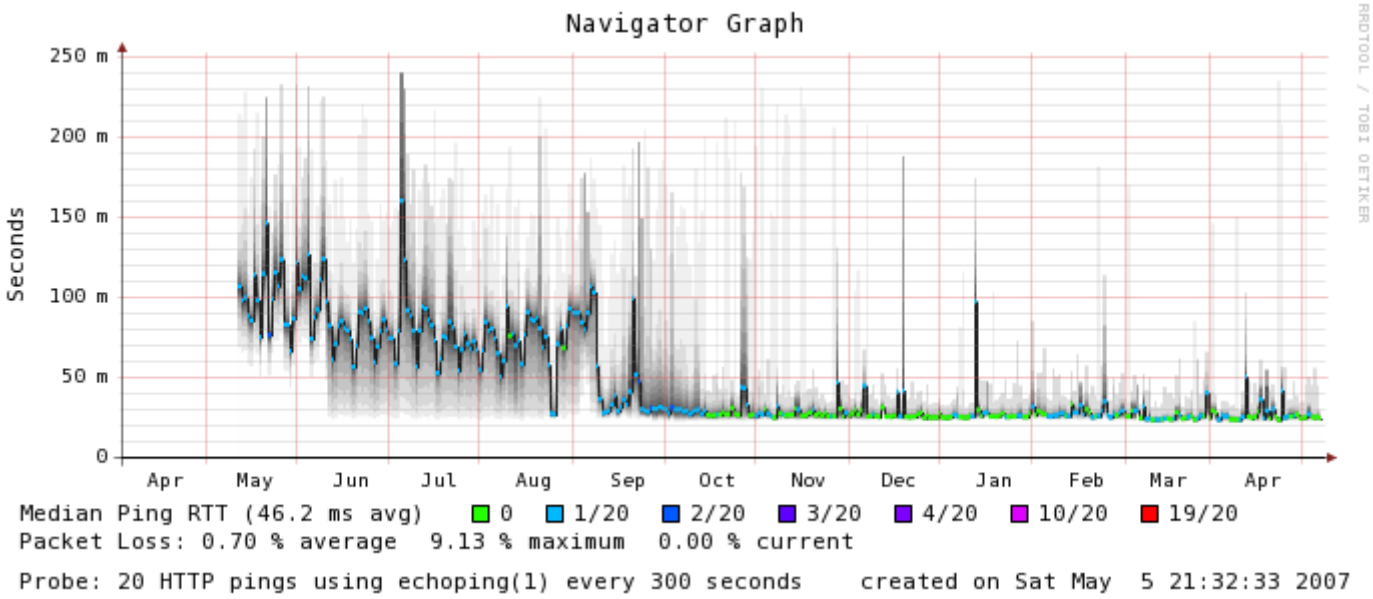
Version 2 features:

GZIP, Edge-side includes, Vary: etc.



Squid

Varnish



Squid
12 servers

Varnish
3 servers

9 servers @ 150W = 12 MWh/y

12 MWh * 500 kg = 6t CO_2 equiv.

Further savings:

Network equipment
Rack-space
Cooling
System Admin Hours

```
$ netstat -I bge1 8
            input          (bge1)           output
   packets  errs       bytes    packets  errs       bytes colls
     87497      0    16940942     105138      0  110891949     0
     86994      2    16507251     105683      0  113001355     0
^C
$ uptime
10:35PM  up 149 days,  7:32, 1 user, load averages: 0.18, 0.46, 0.
```

1000 obj/s @ 100Mbit/s

Sample	Min	Max	Median	Average	Stddev
Full	12,9	3001	16,2	71,1	338,5
90% fractile	12,9	26	15,9	16,3	1,7

(all times are in microseconds)

We...

...don't really know how fast.

...have seen 700 Mbit/s

...have heard 18k obj/s

...have not hit the limit yet.

Varnish version 1 status

Approx 22k lines of code
    (squid 2.5 sources: 122k)

A very solid foundation design/code wise.

Pretty darn good quality for a version 1
    still too many ”`assert(foo) /* XXX fix this */`”

Picking up more and more users
    (happy users, as far as I can tell)

Version 2 plans

Bugfixes
More VCL features
GZIP compression
Prefetching
Vary: processing
Edge-Side-Includes ”ESI”

Sponsors contact: per.buer@linpro.no

var·nish (vär′nĭsh)

n.

1. **a.** A paint containing [...]

tr.v. **var·nished**, **var·nish·ing**, **var·nish·es**

1. To cover with varnish.

2. To give a smooth and glossy finish to.

3. To give a deceptively attractive appearance to; gloss over.

http://varnish-cache.org

Commercial support:
Linpro.no
phk@FreeBSD.org

Reference OS:
FreeBSD, Ubuntu
Packages available:
Yes!
Portable to:
any reasonable POSIX