HTTP cache

or

”Ink by the barrel.”

or

Enough about Johannes Gutenberg, let's talk about Karl Georg Ferdinand Gilke.

Poul-Henning Kamp

<phk@FreeBSD.org>

???

CMS

Typography:
Multiple Fonts,
Hyphenation,
Ligatures &c
(sepll checking
an optional extra.)

Multimedia:
Scaling, Cropping,
Gamma correction (?)

Production:
Replication
Delivery

Content Creation:

Needs diverse input methods:

Text Editors, Image scaling/cropping, File import filters, Feeds...

Flexible Layout/Typography tools

WYSIWYG, Semantic Markup, CSS

Content cross referencing = expensive database lookups:

”Other articles about Paris Hilton”

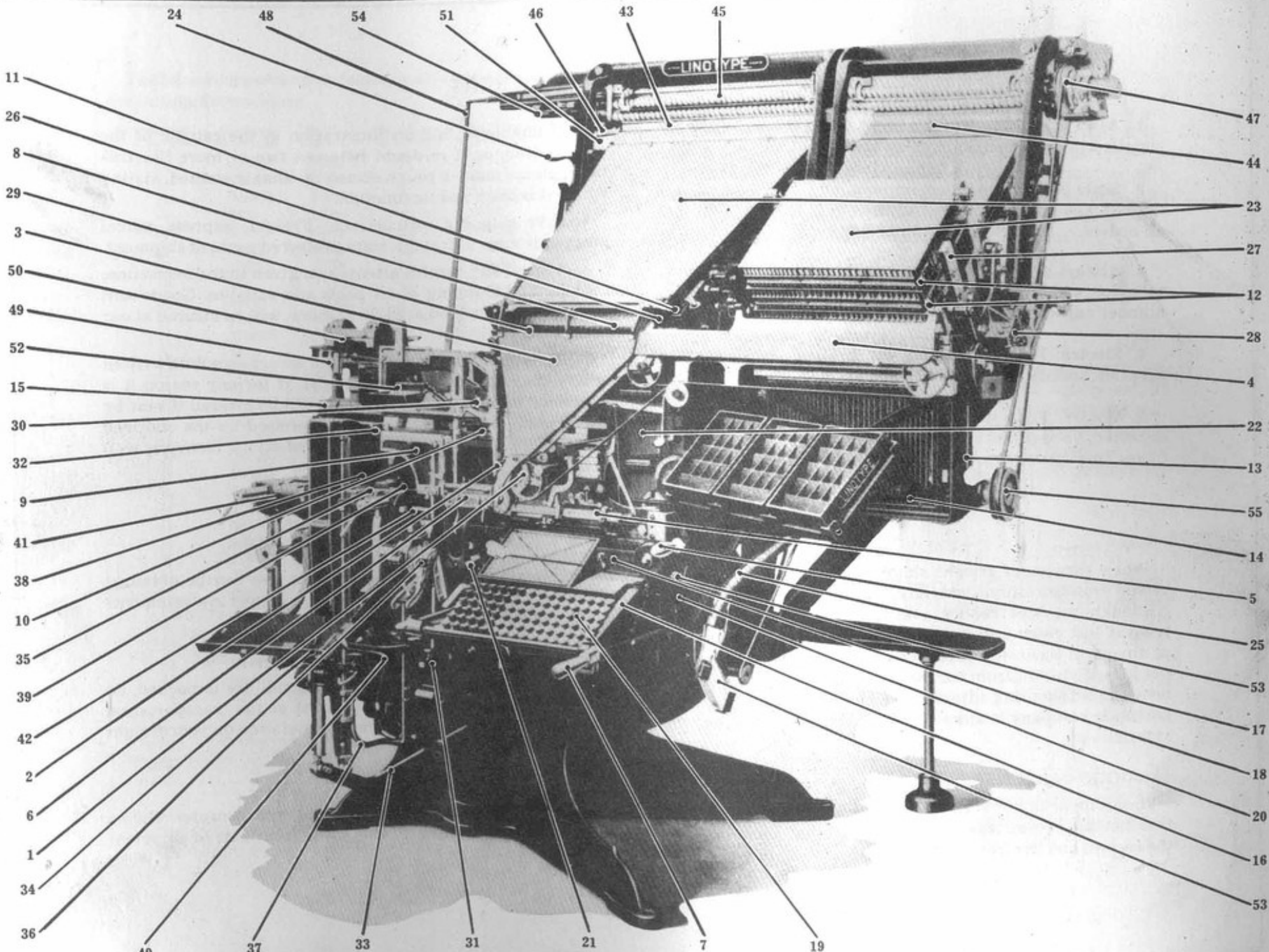
Composition rules can be complex:

”No airline ads, if «crash» present in headline”

User Generated Content

Discussions, galleries, personal views, chats etc.

KEY TO MECHANISMS



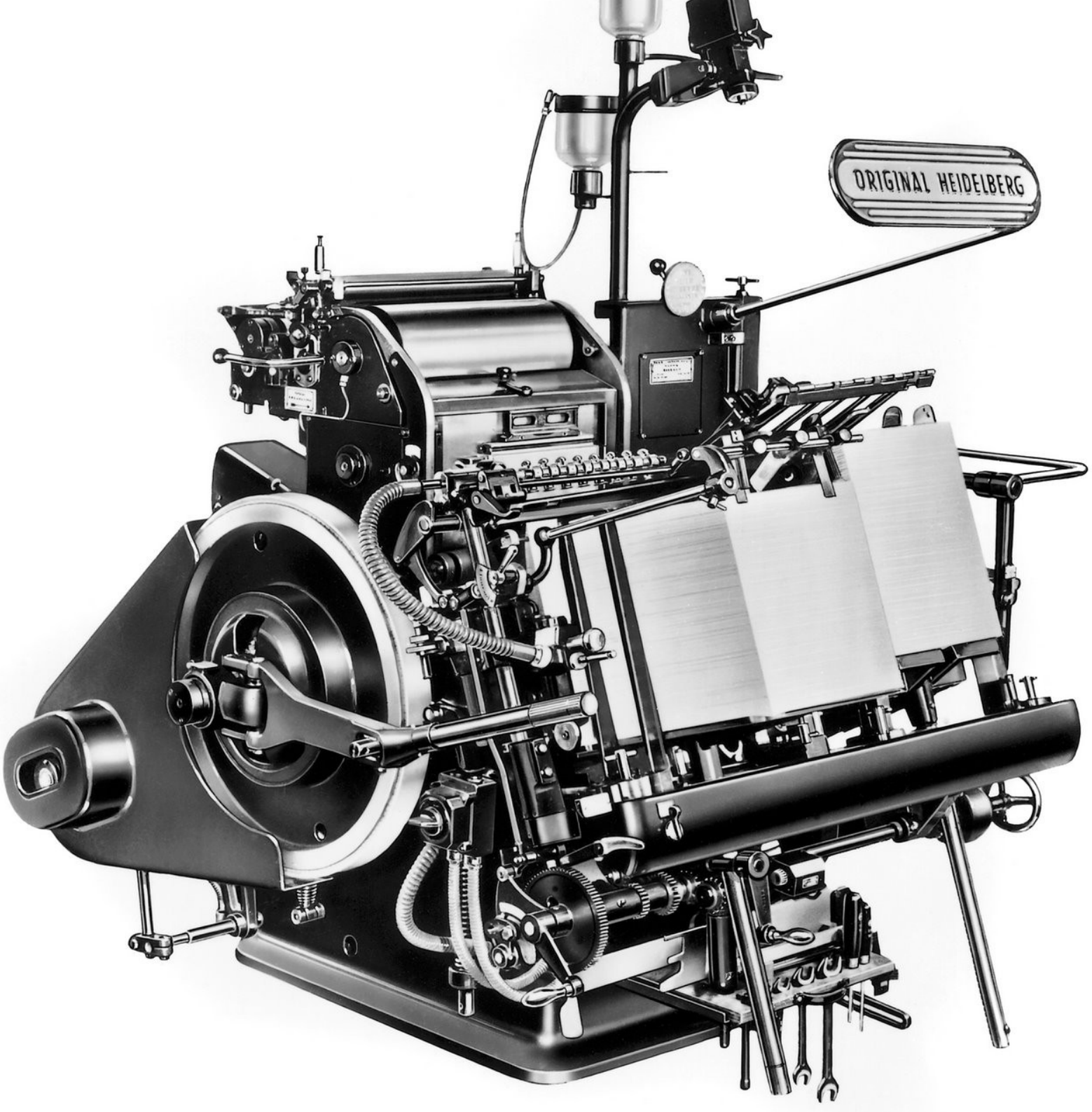
Content Production:

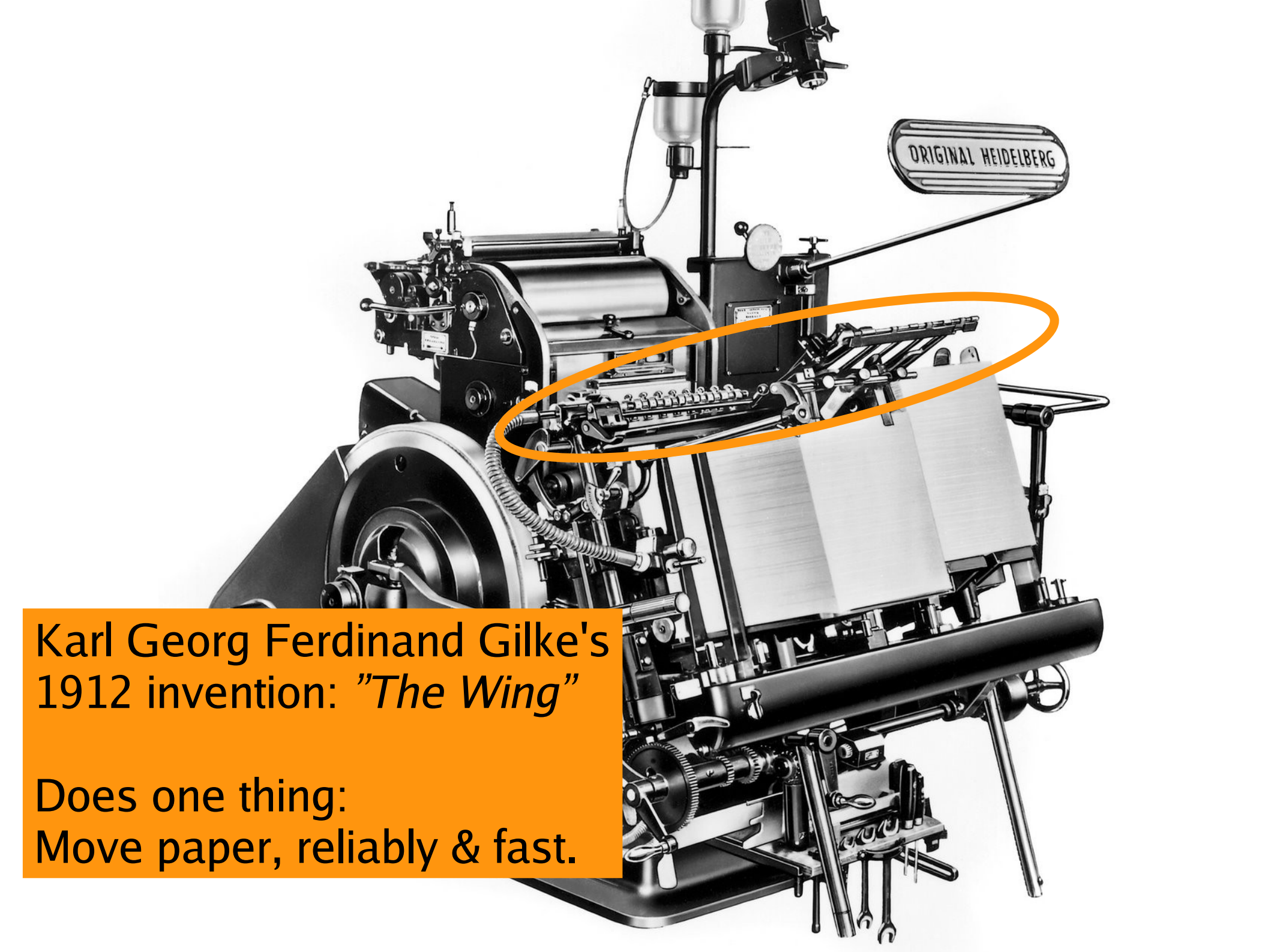
Repeated reproduction of master copy

It was printed on a monster - *a Heidelberg*.

You can smell the weight: Two tons, without the ink.

(”Catch me if you can”)


ORIGINAL HEIDELBERG



Karl Georg Ferdinand Gilke's
1912 invention: *"The Wing"*

Does one thing:
Move paper, reliably & fast.


http://www.youtube.com/watch?v=bLGU1Pba8Es

The Varnish Elevator Pitch:

Varnish delivers content fast & reliably

... reduces the load on your CMS database

... cheap hardware does 100+ kreq/s

... can assist in content composition

... can fix stupid mistakes, fast

... is Free & Open Source Software

... has commercial support

Photo: Michael Feistel

What you get...

...What you pay for

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.

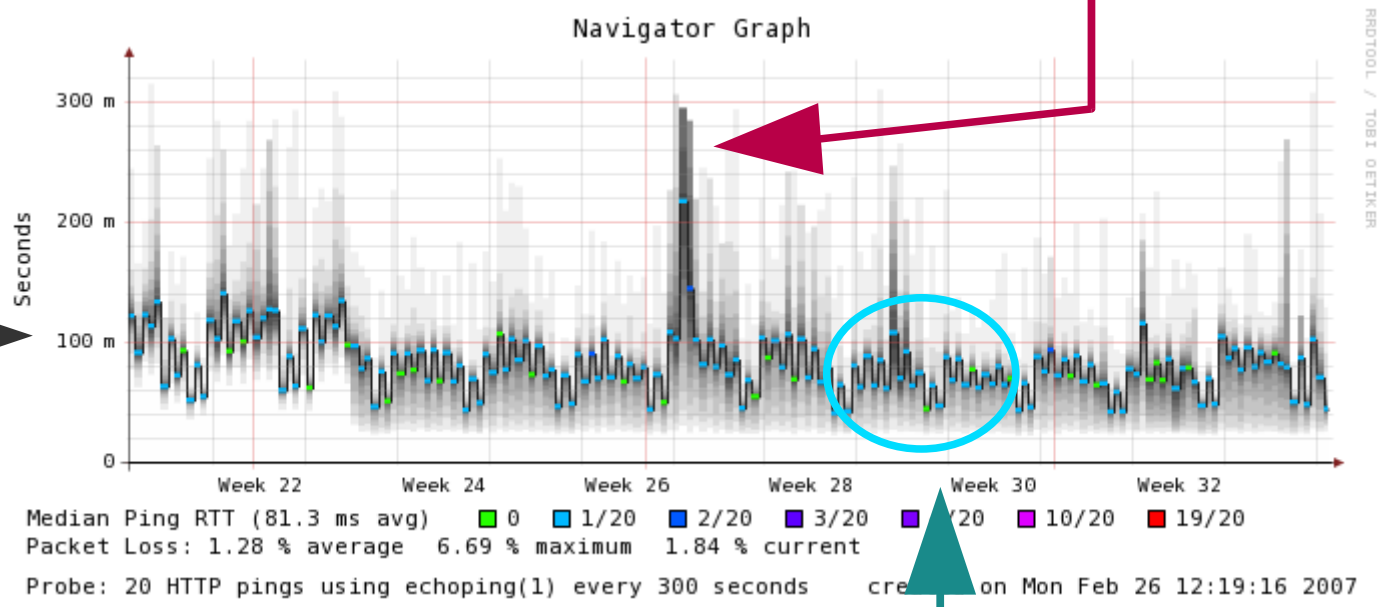
Unhappy with performance and stability.

VG NETT

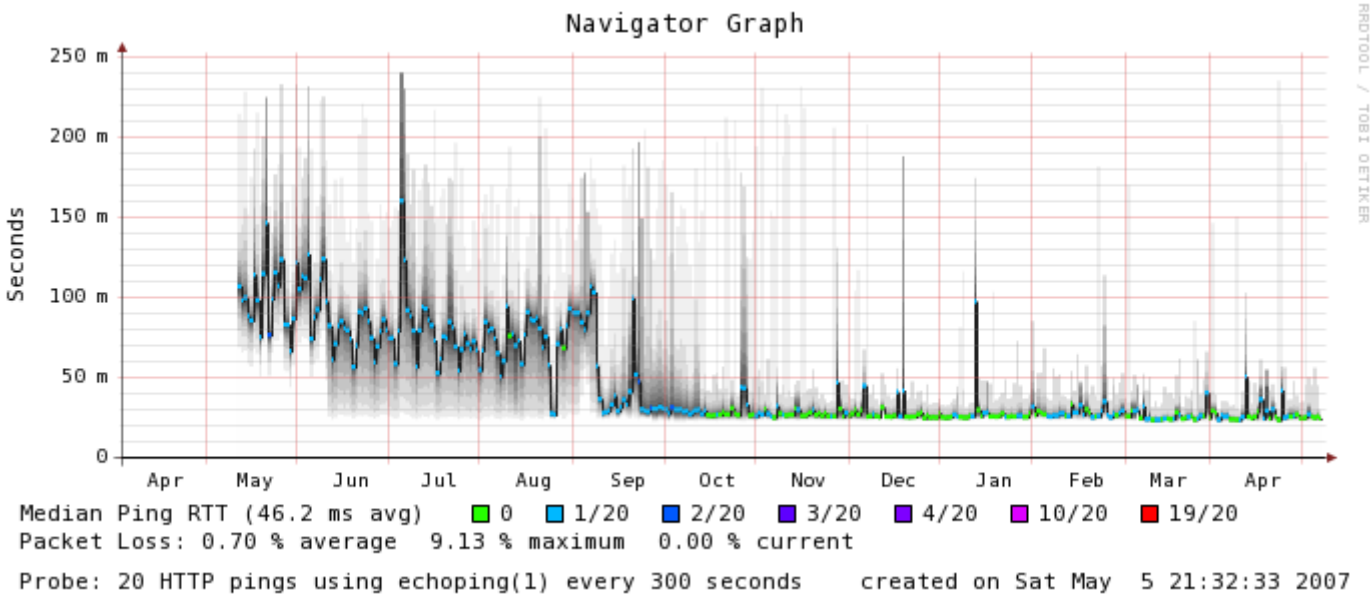
Multimedia arm of Norvegian newspaper ”Verdens Gang”

Slow response

Terrible peak-handling



Significant loss



Squid
12 servers

Varnish
3 servers

Starting from scratch, setting goals:

Varnish is only a HTTP accelerator.

Better configuration.

Better management.

(Much) faster.

Content management focused feature set

RFC2616 cast list

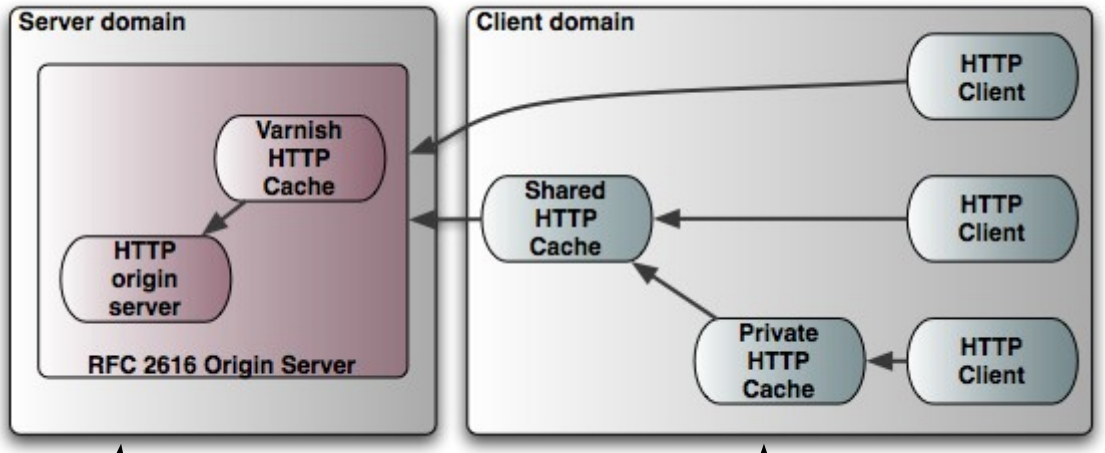
A HTTP Accelerator is not a HTTP cache.

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

RFC2616 compliance as ”origin server”.

Web browsers

Can never cache per-user data



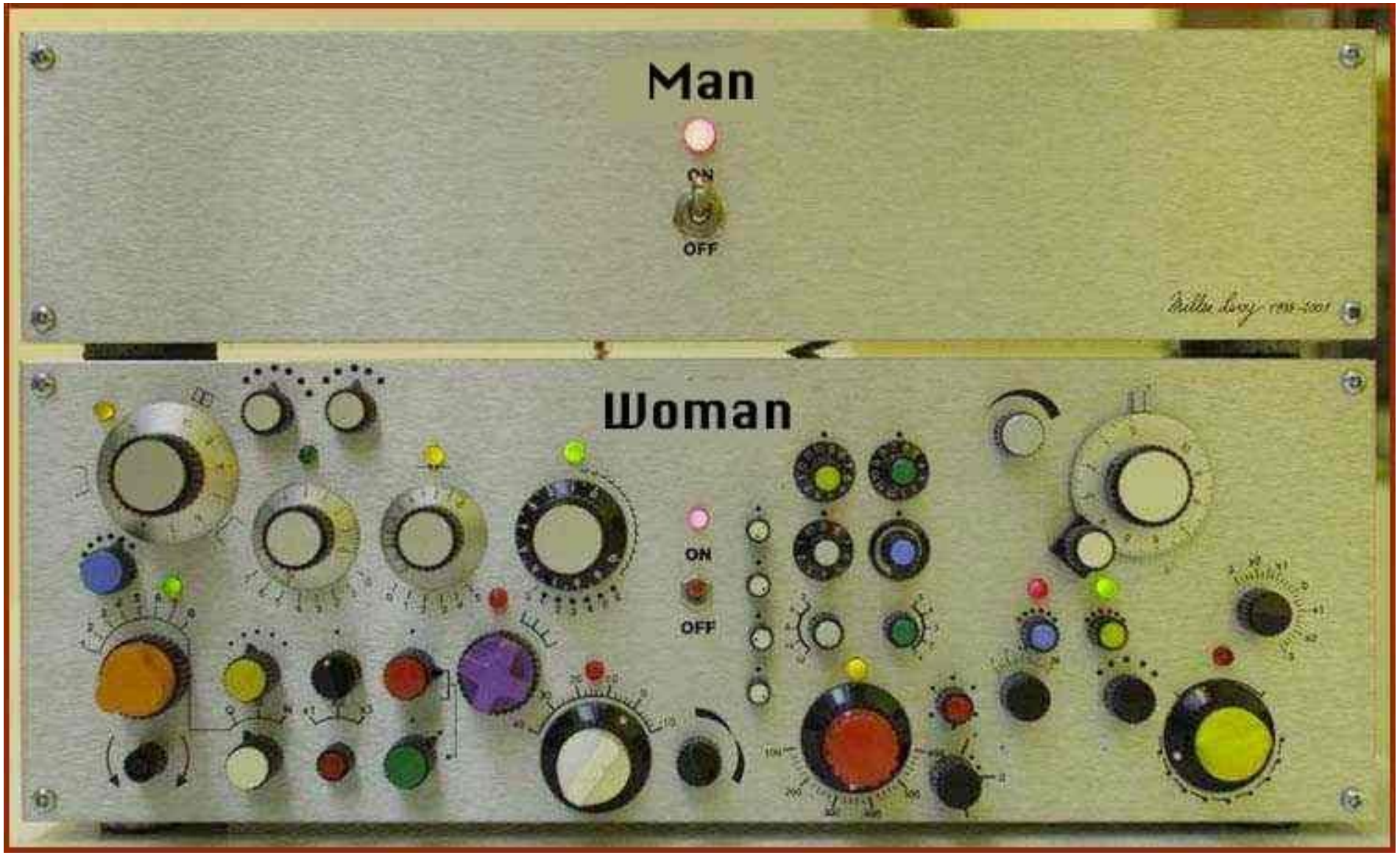
CMS system

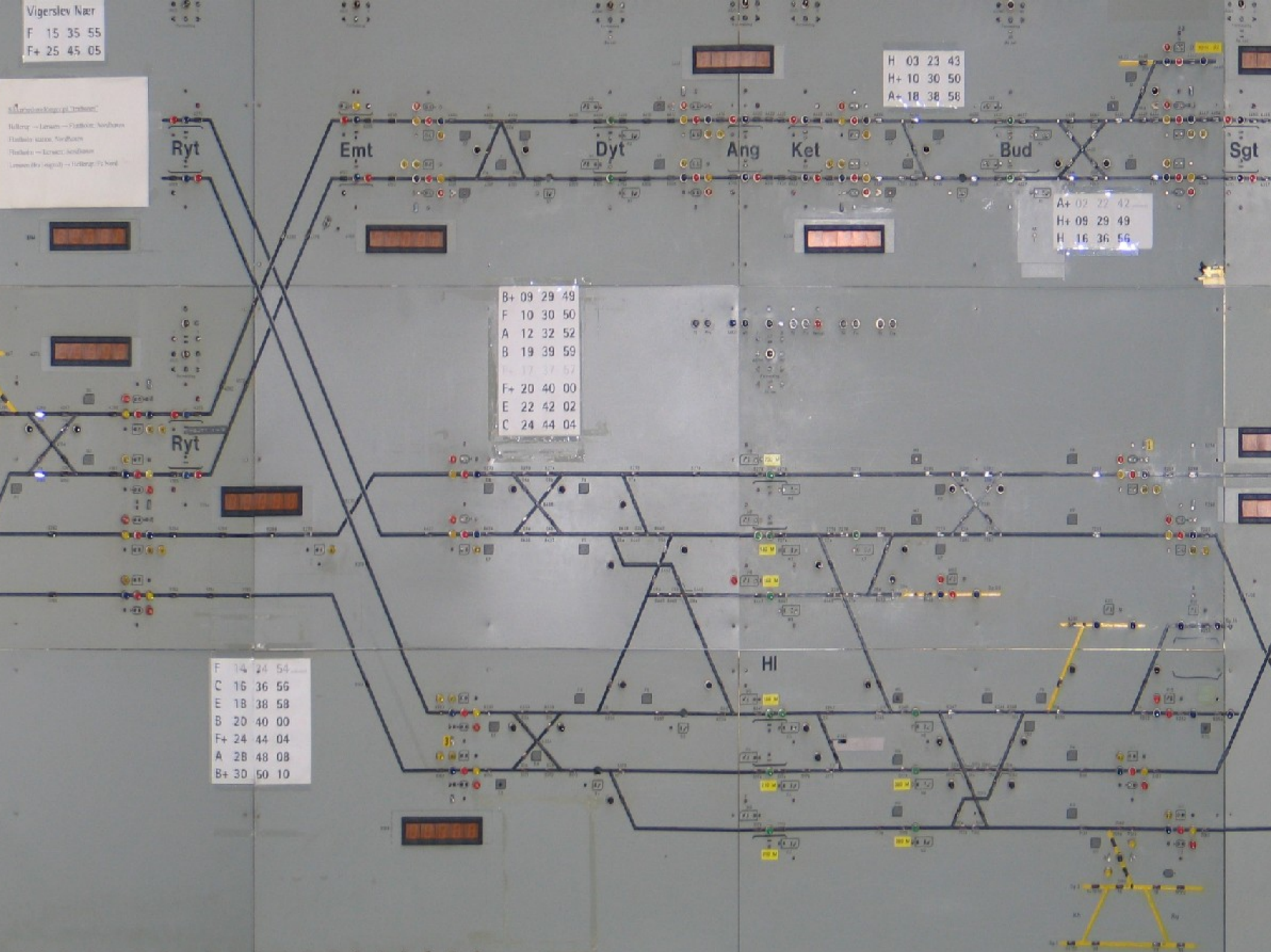
Can sometimes cache per-user data

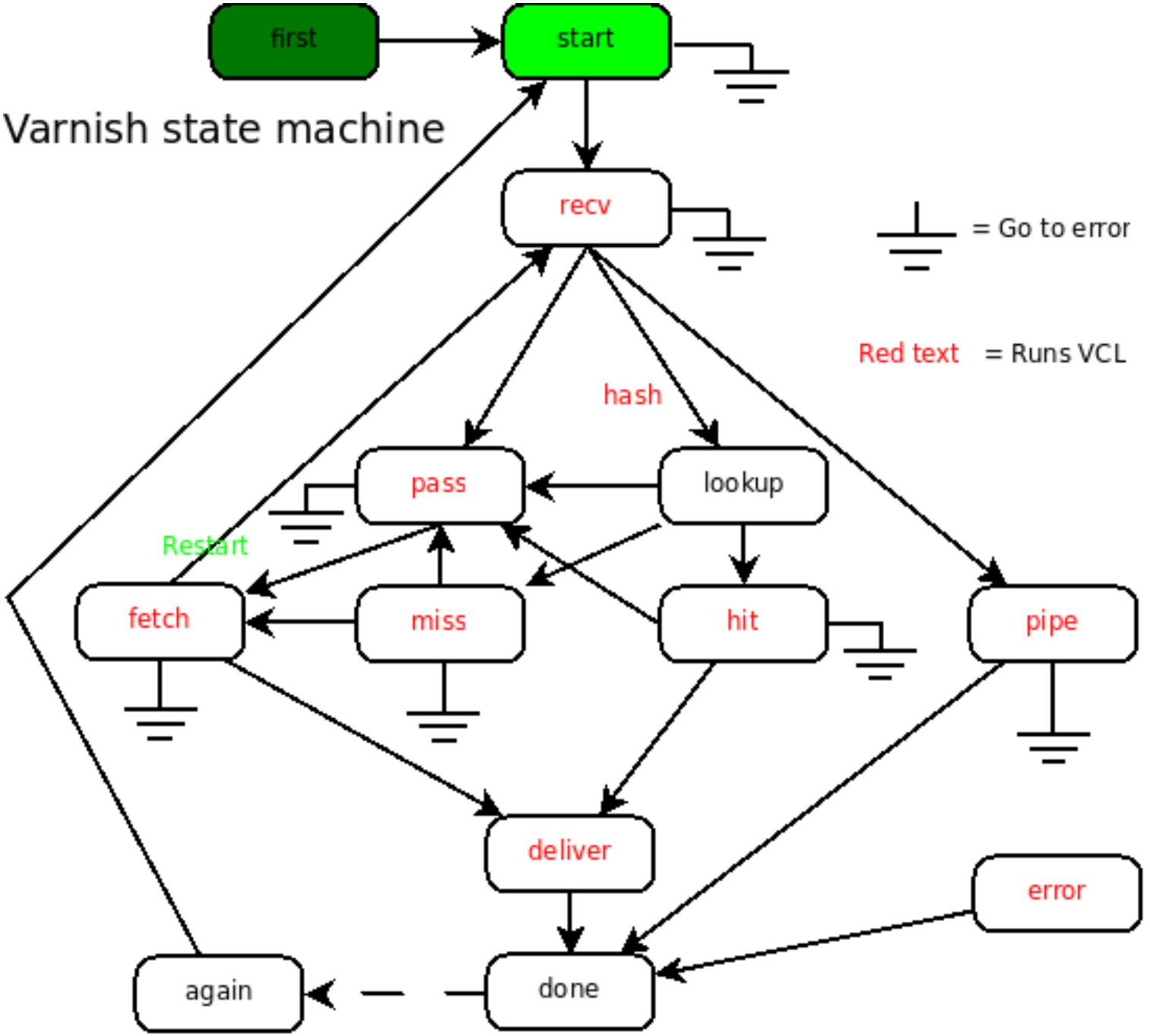
```
$ cat /etc/foobar.conf
# copied from example.conf
#        /svend 19870104
# updated to new version
#        /knud 19941231
# various changes
#        /valdemar 19960523
# DON'T MESS WITH THIS!!!
```

```
# The manual has the details.
HDXHSVVaCS=0

# Dimensionality of chosen of space
allocation_base=3.1418⊕ℵ+∭f(21.4^μ•βe^-ij)

# overflow method (0-7) [default=7]
overflow_method=8
```

```
acl_set= { 1 2 3 4 6 7 21 }

invert_acls=server

acl_reset = { 3 !8 }
```


Man
ON
OFF
Woman
ON
OFF


Vigerslev Nær
F 15 35 55
F+ 25 45 05
Ryt
Emt
Dyt
Ang
Ket
Bud
Sgt
H 03 23 43
H+ 10 30 50
A+ 18 38 58
A+ 02 22 42
H+ 09 29 49
H 16 36 56
B+ 09 29 49
F 10 30 50
A 12 32 52
B 19 39 59
F+ 20 40 00
E 22 42 02
C 24 44 04
Ryt
HI
F 14 34 54
C 16 36 56
E 18 38 58
B 20 40 00
F+ 24 44 04
A 28 48 08
B+ 30 50 10


first
start
Varnish state machine
recv
= Go to error
Red text = Runs VCL
hash
pass
lookup
Restart
fetch
miss
hit
pipe
deliver
error
again
done

VCL - Varnish Configuration Language

```
sub vcl_recv {
   if (req.url ~ ”(\.\.|\.exe)”) {
      error(999, ”Bugger off.”);
   }
   if (client.ip ~ editor_ips) {
      set req.http.x-cms = ”no-stats”;
      return(pass);
   }
   if (req.url ~ ”\.(jpg|png|gif|css)$”) {
      unset req.http.cookie;
      unset req.http.authenticate;
      set req.backend = static_backend;
   }
   if (req.url == ”hotstory.hmtl”) {
      set req.url = ”hotstory.html”);
   }
}
```

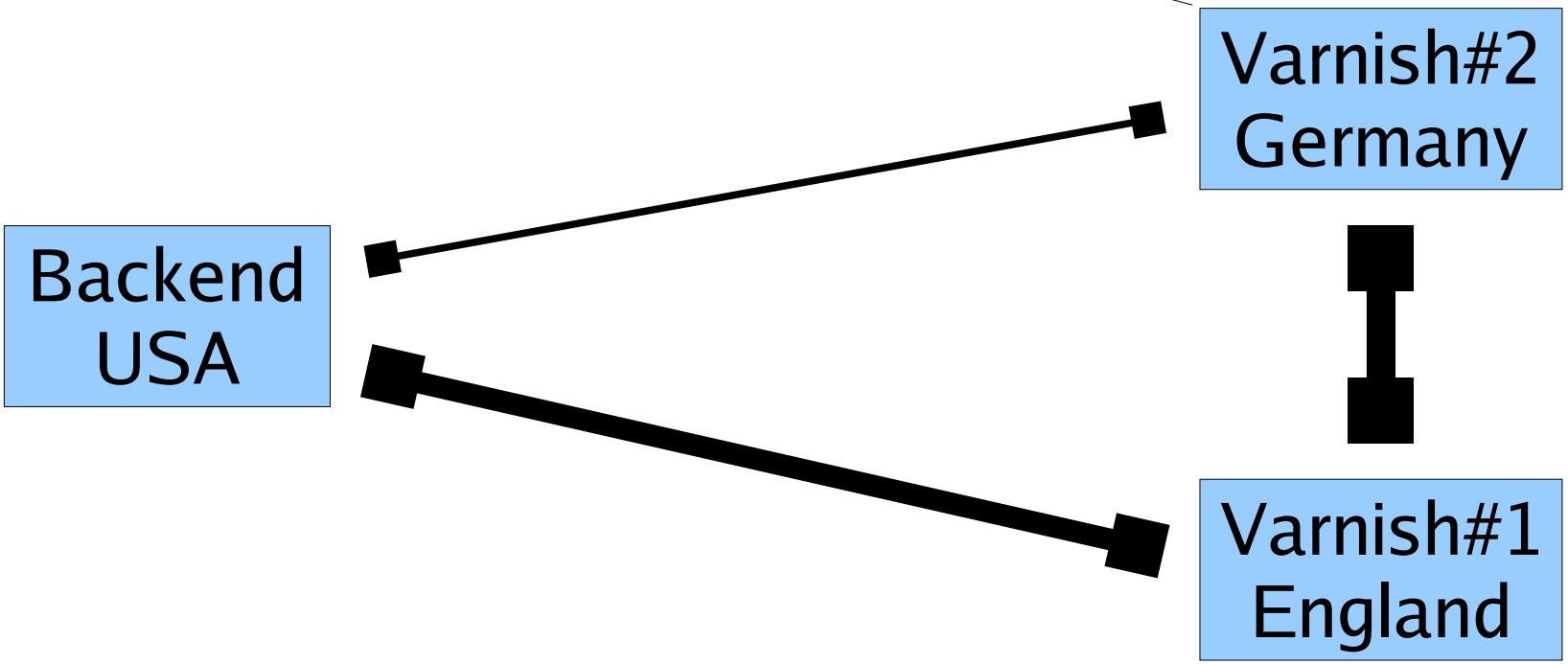
Why VCL rocks:

Compiled to binary/shlib via C-code
- ⇒Runs full speed

You can have multiple VCL's loaded at the same time
- ⇒Switch between them without restart
- ⇒Instantaneous

Allows you to do anything you might fancy
- ⇒Inline-C code, 'nuff said.
- ⇒Modules/shlib will make it easier (3.0 feature)

```
sub vcl_recv {
  if (client.ip == "varnish1") {
    set req.backend = usa;
  } else {
    set req.backend = england;
  }
}
```



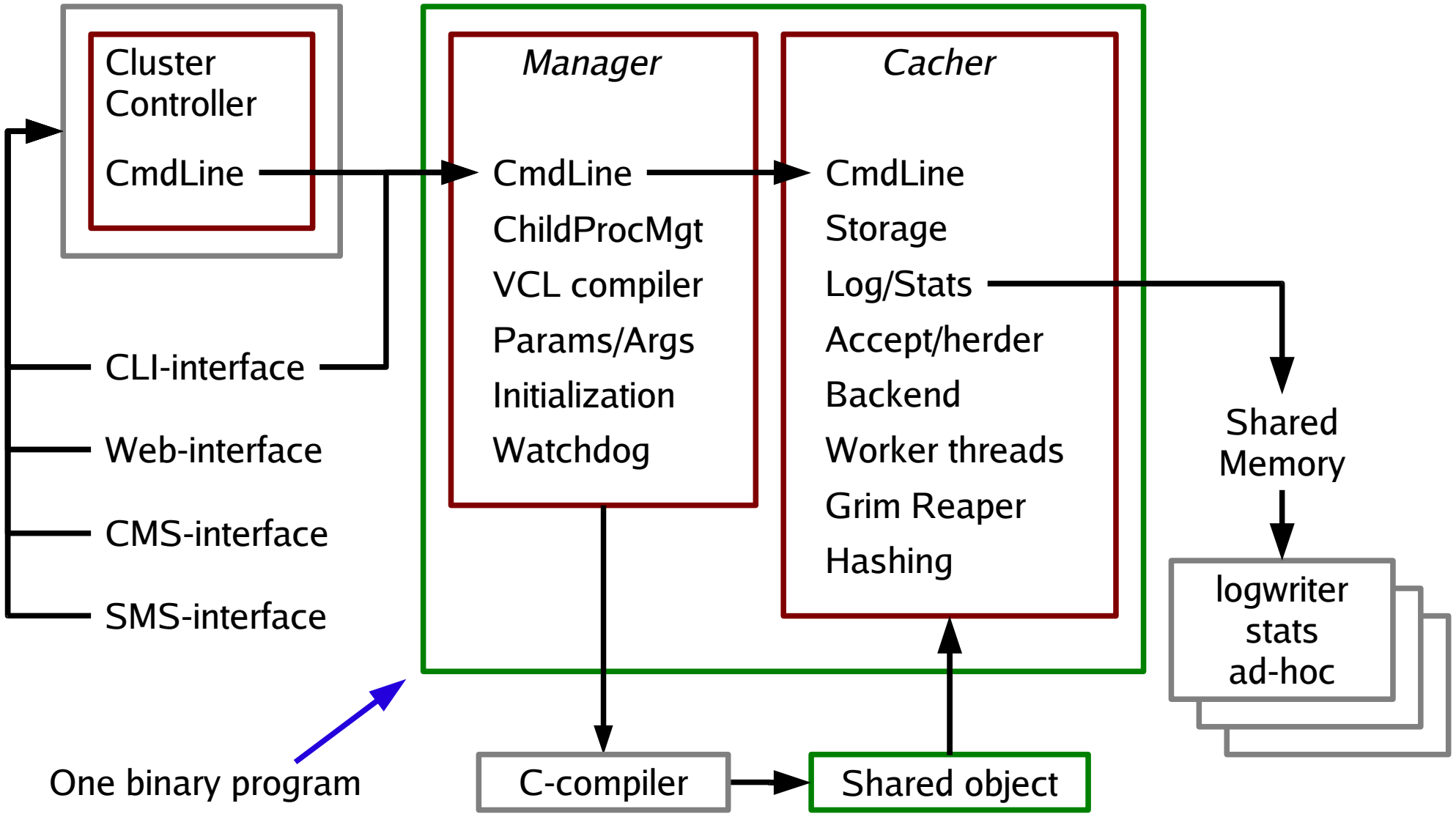
Managing Varnish

Command Line Interface for real-time control

Management/Worker process split:

- Manager (re)starts worker
- Allows privilege separation
- Contains multithreading to worker process

Varnish architecture



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

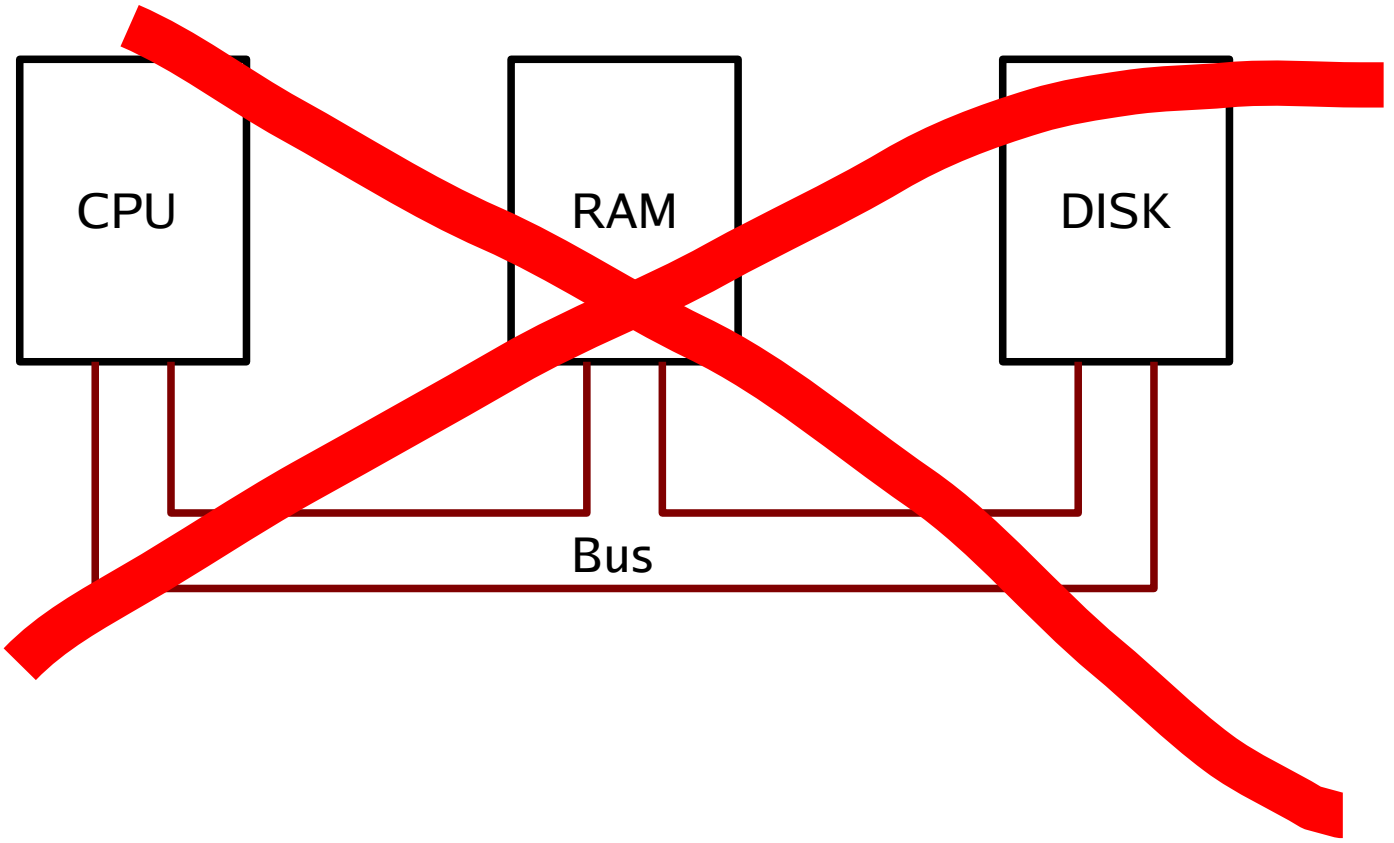
```
param.show overflow_max
200 330
overflow_max     100 [%]
                 Default is 100
                 Limit on overflow queue length in
                 percent of thread_pool_max parameter.

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

Performance and speed

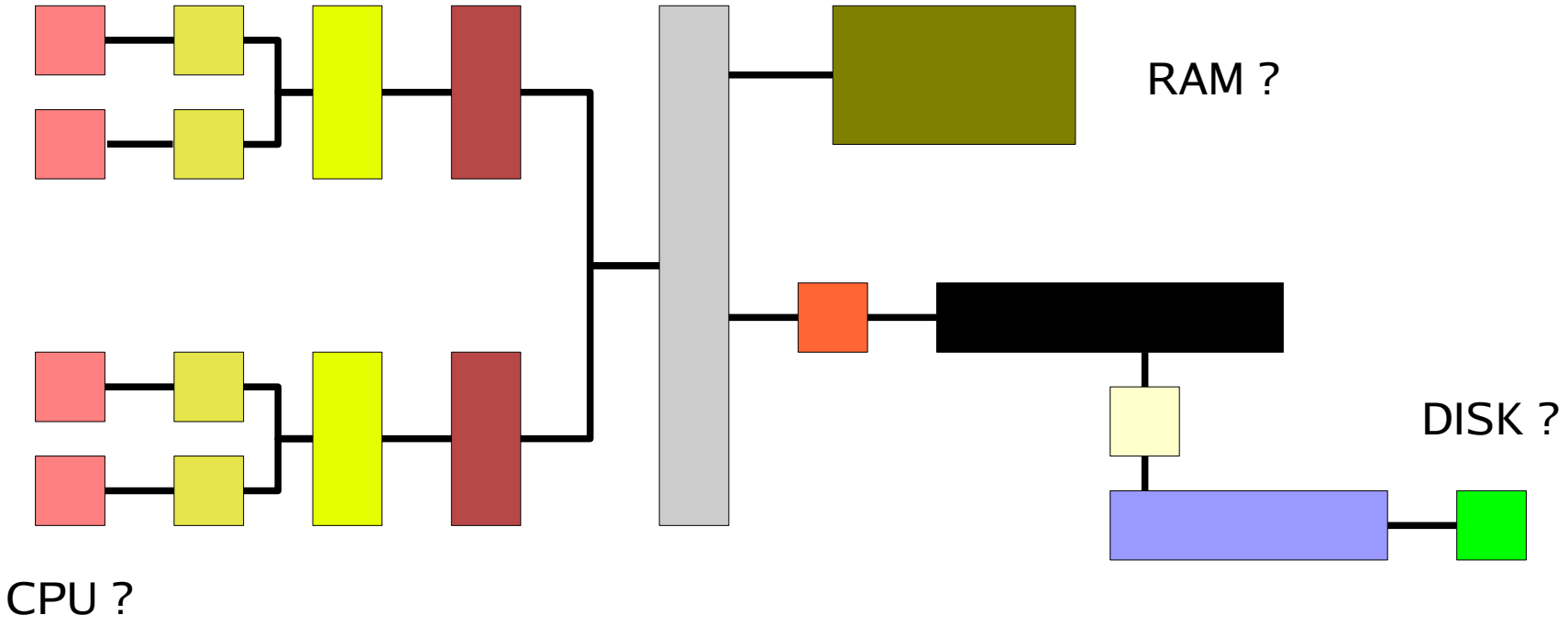
Program for performance from day one

Use modern features:

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


CPU
RAM
DISK
Bus

Not your dads computer anymore:



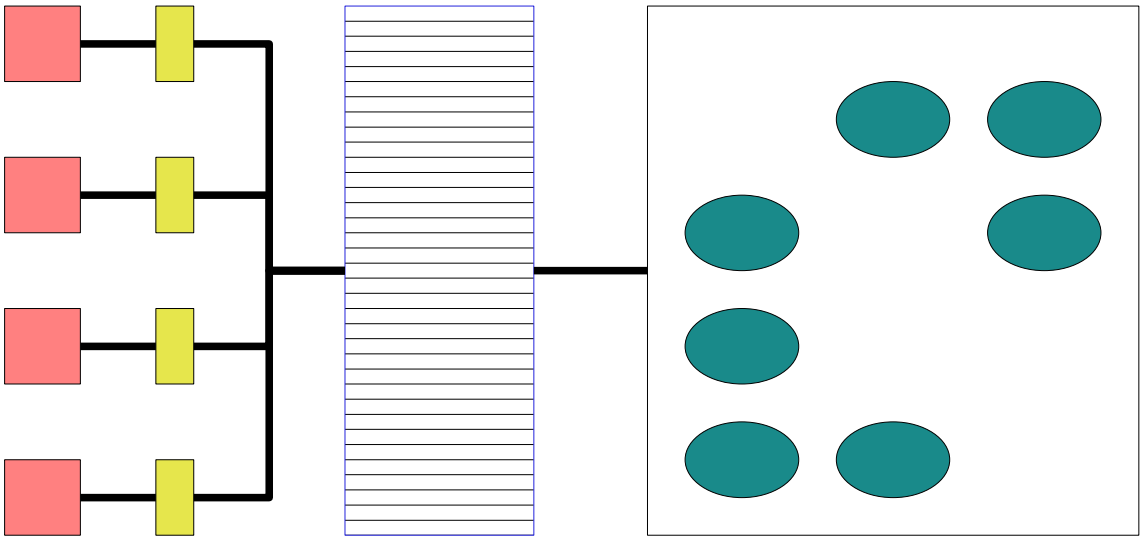
...and besides, the operating system virtualizes all of this

CPUs/Cores

Caches

The Virtual Page Cache

formerly known as "RAM"



Object storage

high RTT

Performance Pricelist

- char *p += 5;
- strlen(p);
- memcpy(p, q, l);
- Locking
- System Call
- Context Switch
- I/O Disk Access
- File operation
- Directory operation

CPU

Memory

Protection

Mechanical

100,000,000/s

50/s

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 · 0.010s + 1,000,000 * 0.001s = 16 minutes

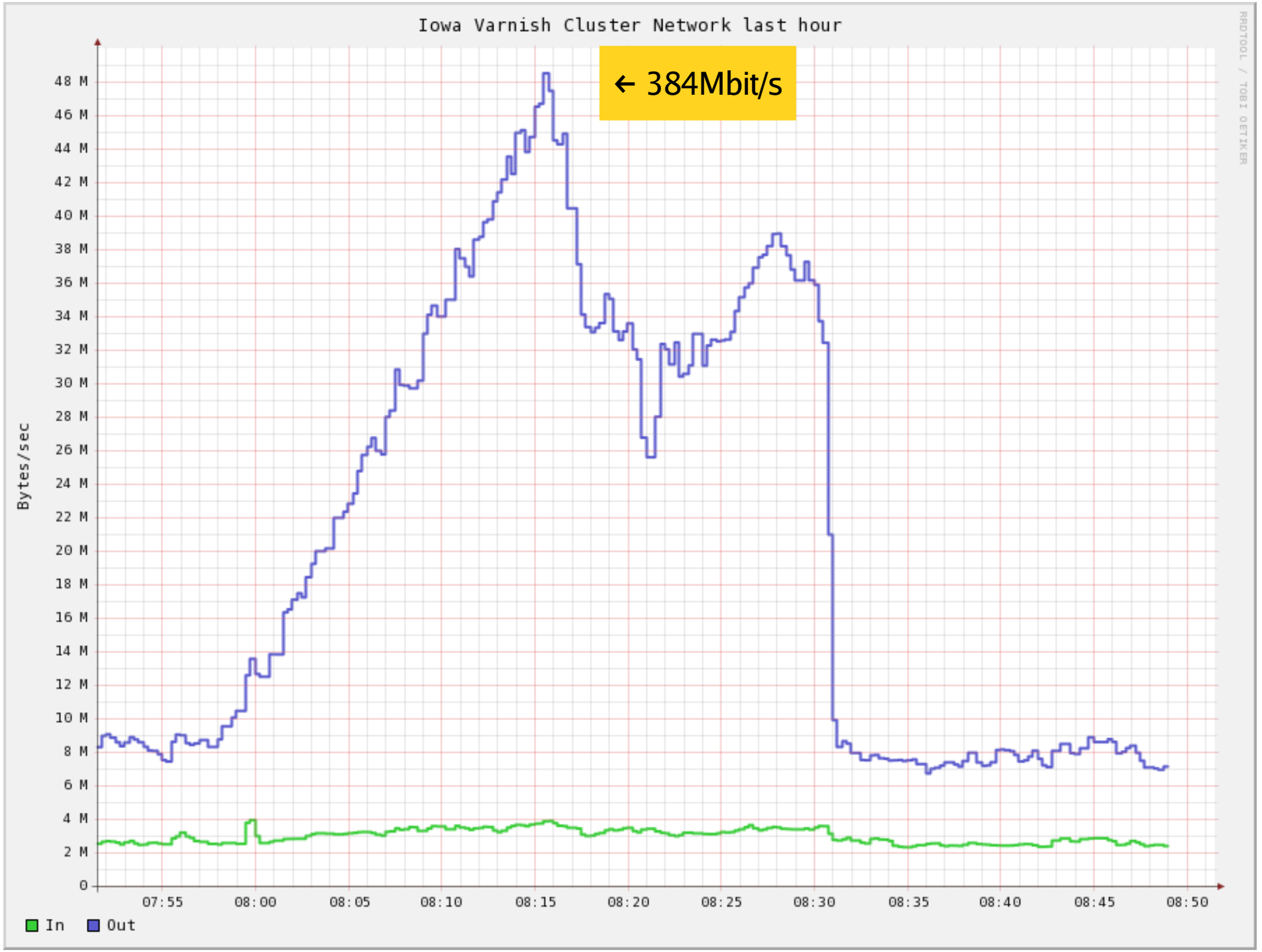
Logging to shared memory is almost free:

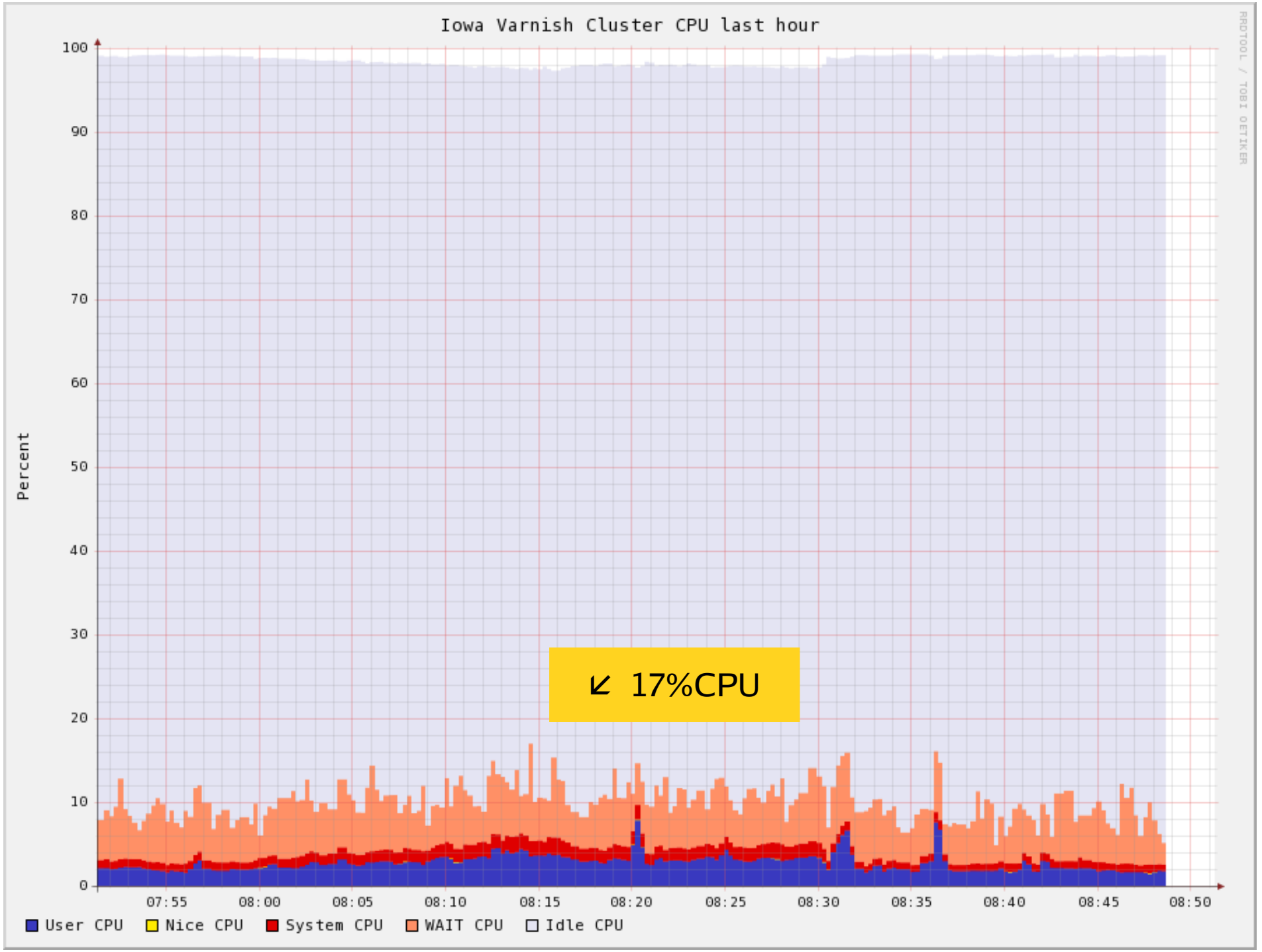
```
char *logp, *loge;

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


Iowa Varnish Cluster Network last hour
← 384Mbit/s
Bytes/sec
48 M
46 M
44 M
42 M
40 M
38 M
36 M
34 M
32 M
30 M
28 M
26 M
24 M
22 M
20 M
18 M
16 M
14 M
12 M
10 M
8 M
6 M
4 M
2 M
0
07:55
08:00
08:05
08:10
08:15
08:20
08:25
08:30
08:35
08:40
08:45
08:50
In
Out
RRDTOOL / TOBI OETIKER


Iowa Varnish Cluster CPU last hour
RRDTOOL / TOBI OETIKER
Percent
100
90
80
70
60
50
40
30
20
10
0
07:55
08:00
08:05
08:10
08:15
08:20
08:25
08:30
08:35
08:40
08:45
08:50
↙ 17%CPU
User CPU
Nice CPU
System CPU
WAIT CPU
Idle CPU

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
```

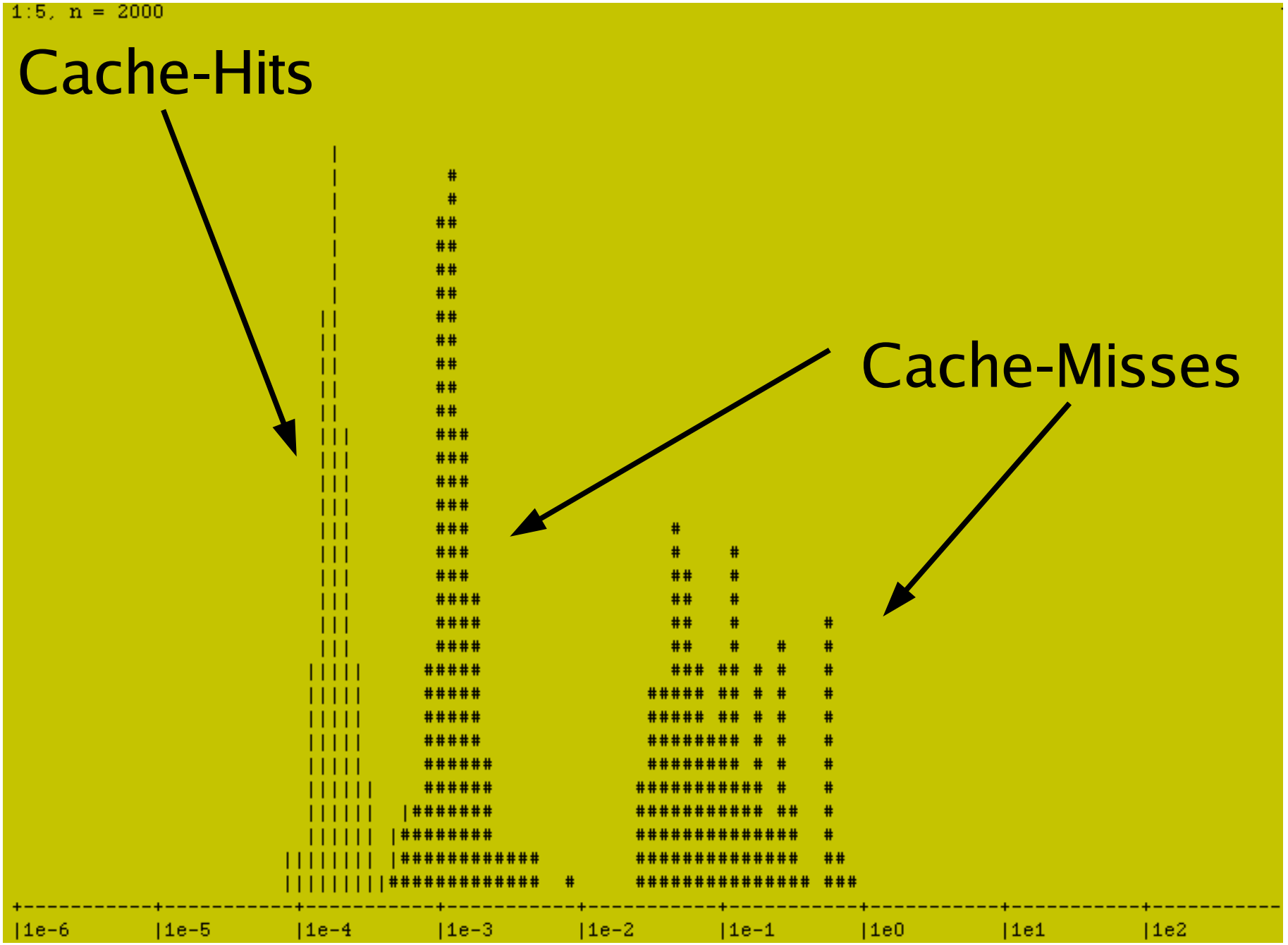
Varnishtop(1) – logfile ”top” program

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 /
```

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
```

Content Management Features:

Instant action purges (regexp or exact match)

TTL/Caching policy control in VCL

Load/Situation mitigation in VCL

Header Washing

Vary

Edge-Side-Includes (ESI)

Purges:

NB:
Varnish 3.0
Terminology

Cache eviction based on exact criteria

Only through http transaction (= cache hit)

Can take all ”Vary:” versions of object.

Bans:

Cache-hit prevention based on loose criteria

```
ban req.url ~ ".*royal.*naked.*"
```

CLI or http transaction

All the other stuff you can do in VCL

TTL control
cache/pass/pipe decision
URL rewrites
header washing
IP based access control
DoS prevention
Spider-dieting
mod_security-like screening
&c &c.

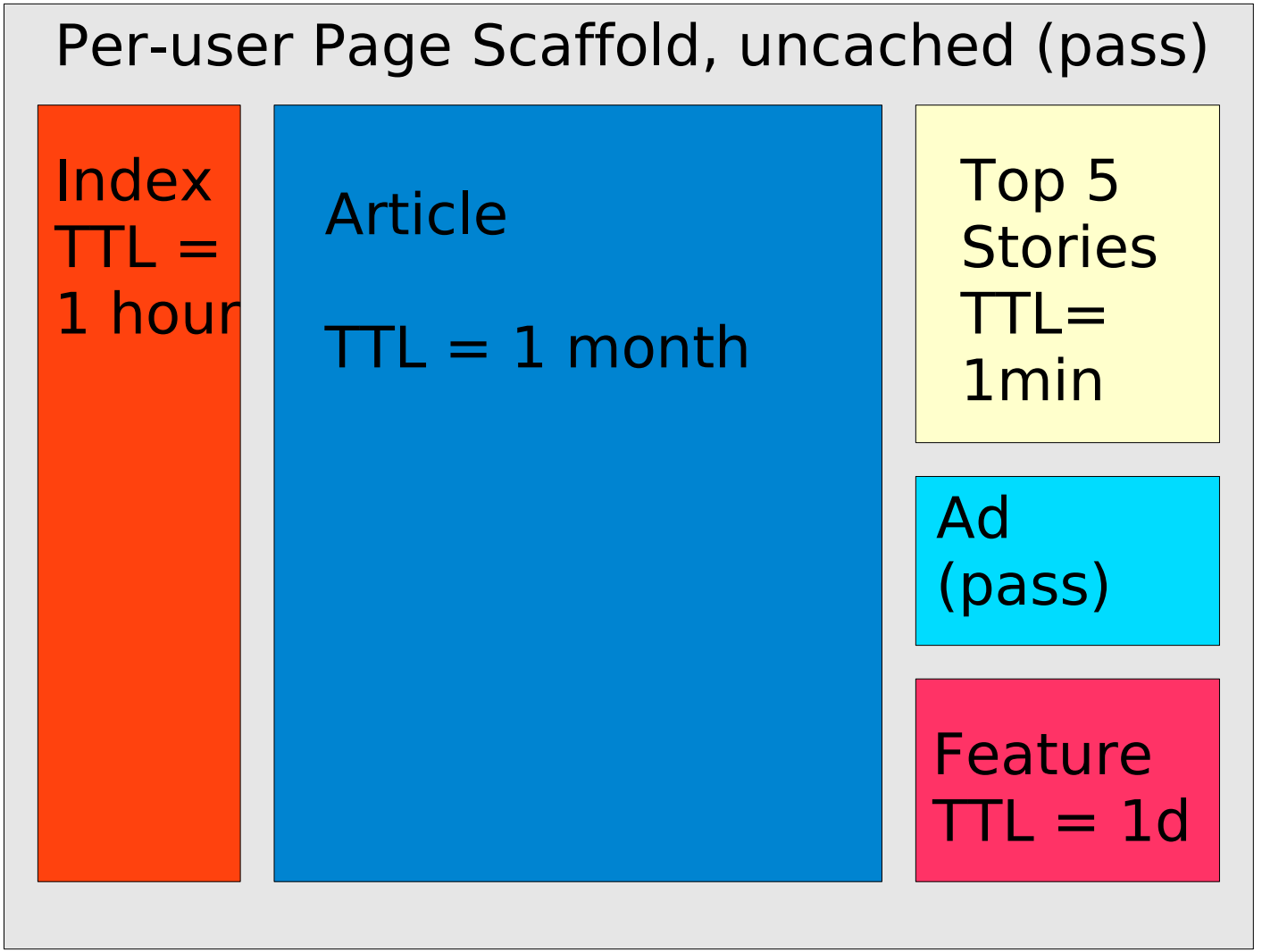
And if VCL can not do it ?

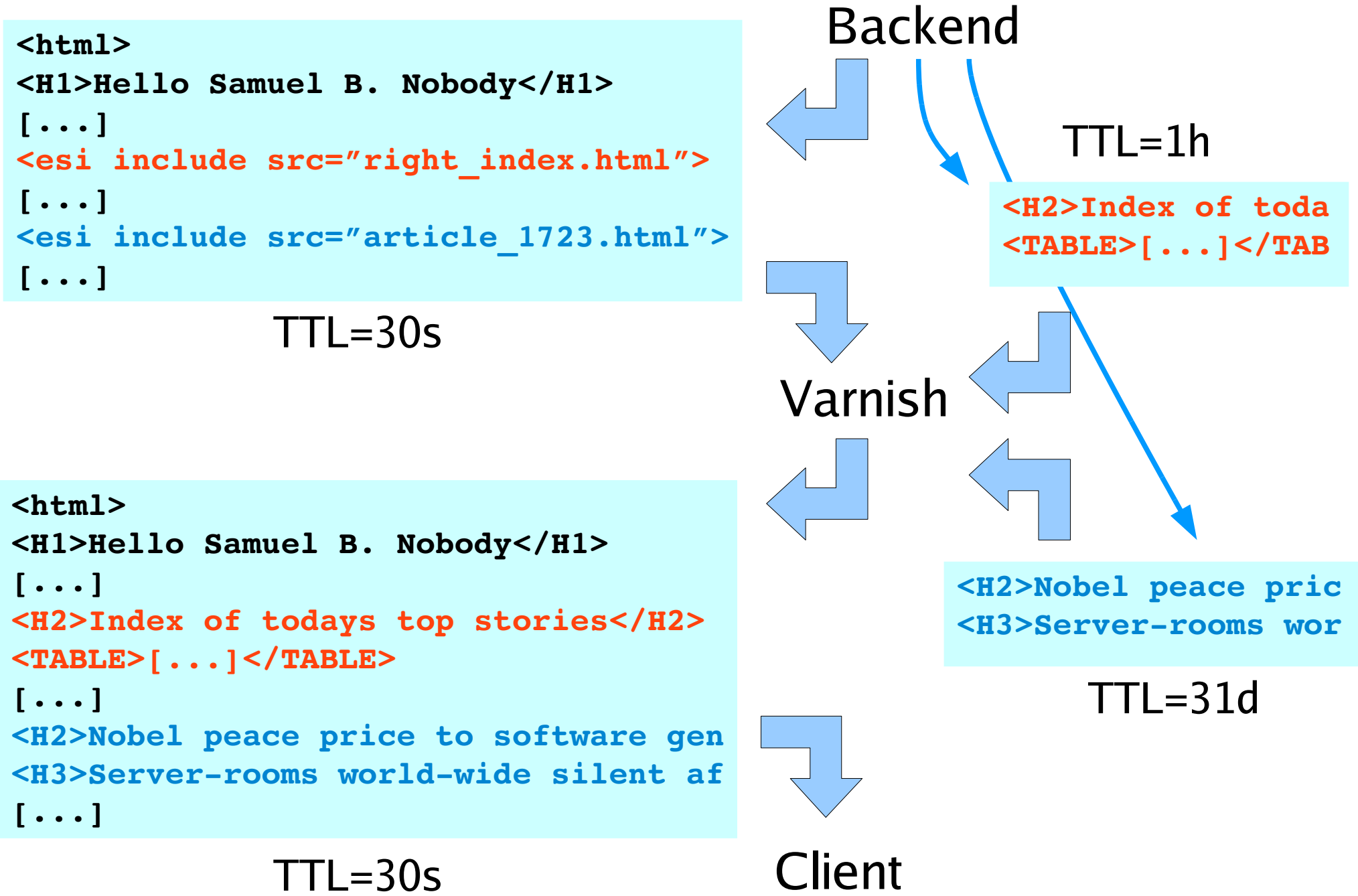
Inline C code:

```
sub vcl_recv {
  C{
    syslog(LOG_INFO, "Trouble: %u”, foo);
  }C
}
```

In 3.0 also: ”modules” in the form of shared libraries.

Edge-Side-Includes (”ESI”)




<html>
<H1>Hello Samuel B. Nobody</H1>
[...]
<esi include src="right_index.html">
[...]
<esi include src="article_1723.html">
[...]
TTL=30s
Backend
TTL=1h
<H2>Index of toda
<TABLE>[...]</TAB
Varnish
<html>
<H1>Hello Samuel B. Nobody</H1>
[...]
<H2>Index of todays top stories</H2>
<TABLE>[...]</TABLE>
[...]
<H2>Nobel peace price to software gen
<H3>Server-rooms world-wide silent af
[...]
<H2>Nobel peace pric
<H3>Server-rooms wor
TTL=31d
TTL=30s
Client

And if I had more time I would also mention:

Round-Robin, Random, Client & Hash backend directors
Backend Health-polling, grace mode, saint mode
Modular storage-, hash-, waiter-code APIs
Libvarnishapi.a library for stats/log/cli access
Ipv6 support on all network connections
ACL's are compiled to C-code too = very fast
PSK security on CLI connection
Privilege separation/drop (manager/worker process)
Written entirely in C, only 64kloc (incl. JEmalloc)
Cache hit = 7 system calls (typ: 15-30µs)
Almost 10% of source lines are asserts
82.3% of approx 50kloc covered by 189 test-cases
Builtin useful panic/backtrace formatter
Why marines.com sent email saying ”You saved our ass”

var·nish (värʹnĭsh) 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.

Commercial support:
Varnish-Software.com

Reference OS:
FreeBSD, Linux
Packages available:
Yes!
Portable to:
Any reasonable POSIX