Varnish

- a server-side http cache

Poul-Henning Kamp

phk@FreeBSD.org

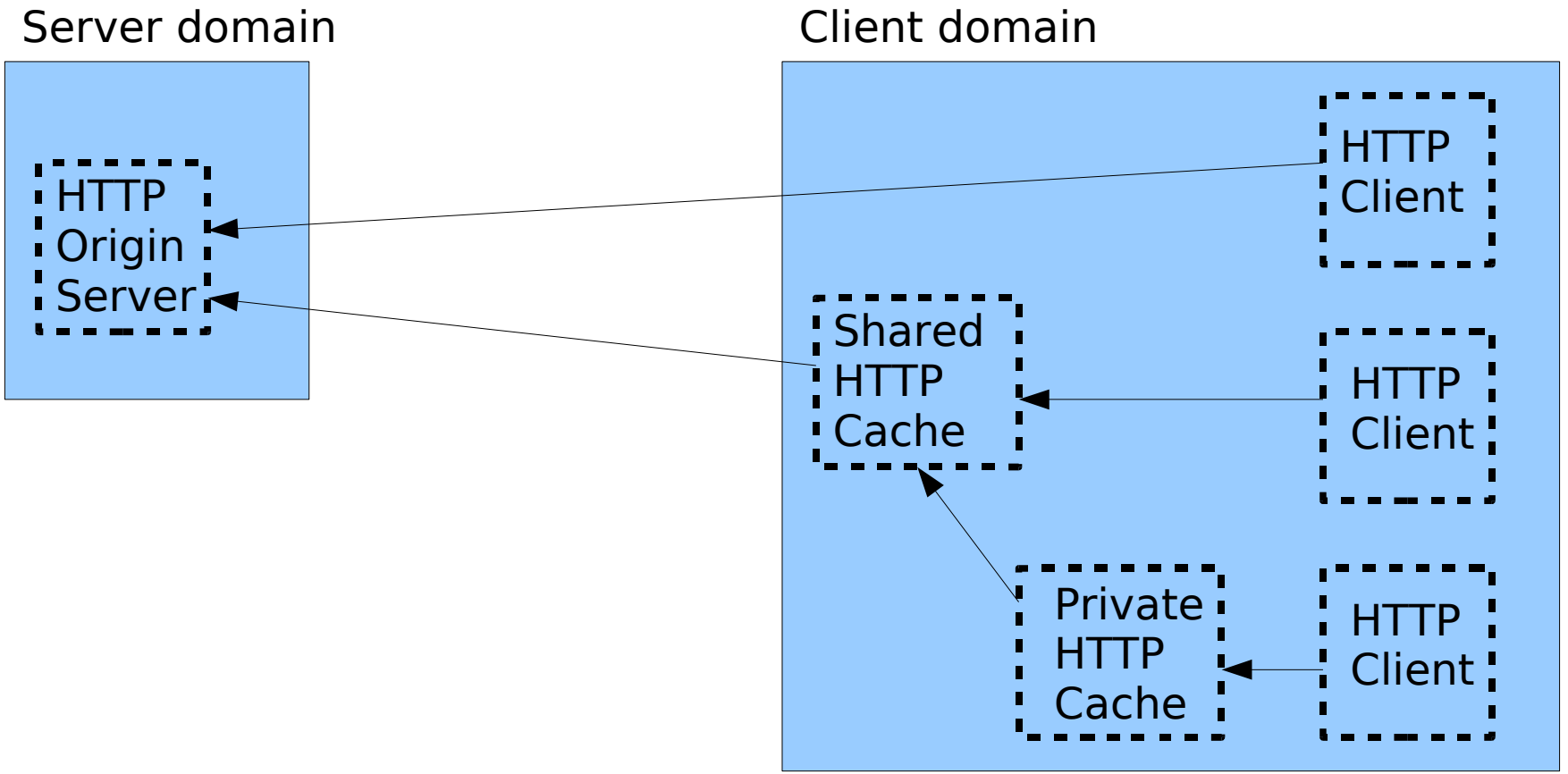
dictionary:Varnish

- tr. v. var·nished, var·nish·ing, var·nish·es
  - To cover with varnish.
  - To give a smooth and glossy finish to.
  - To give a deceptively attractive appearance to.

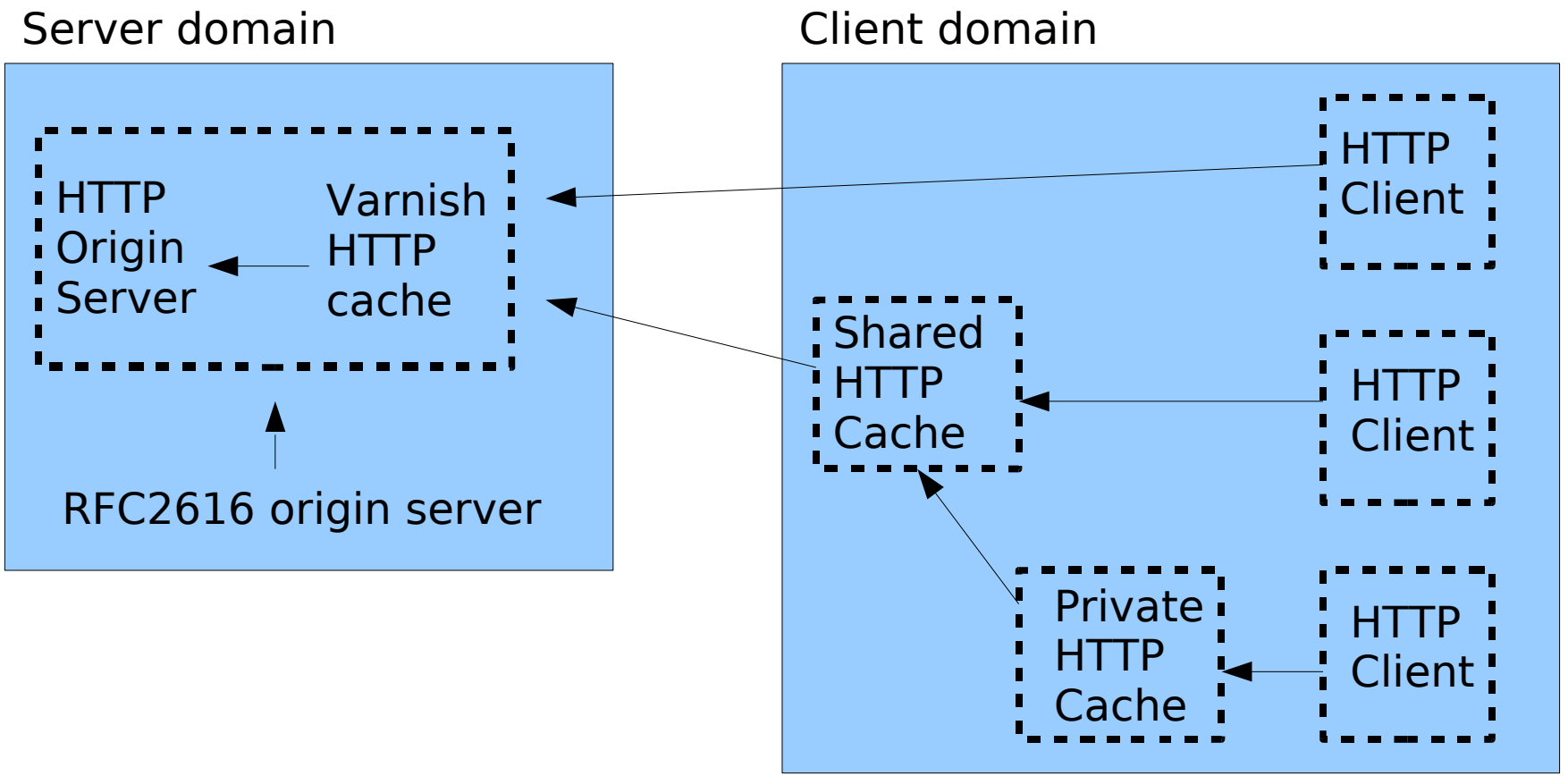
Varnish Cheat-Sheet

- Web-accelleration for slow CMS systems
- Narrow focus on server side speedup
  - No FTP etc.
  - Content provider features
- Modern Hardware/Software
  - 64bit, large RAM, sendfile, accept filters
  - Also runs on 32bit machines
  - SMP/Multicore friendly architecture

RFC2616 on HTTP caches



RFC2616 and Varnish



Client Cache Situation

- Origin servers are adversarial.
- Anything the origin server says is law.
- If in doubt: don't cache.
- Be transparent at any cost.
- If origin server does not reply: error.

Server Cache Situation

- Backend (origin server) is our friend.
- We can be responsible for modifying the origin servers instructions.
  - Change TTL, rewrite URLs etc.
- Whatever happens: protect the backend.
- If backend does not reply: do something!

CMS systems

- **`GET / HTTP/1.1`**
  - Hang on, I need to look up a few hundred things in my database and then do a lot of editing with some badly written software.
- **`HEAD / HTTP/1.1`**
  - Hang on, I need to look up a few hundred things in my database and then do a lot of editing with some badly written software, and then I will throw the result away.

CMS systems are SLOW!

- Complex content generation process
- Single database prevents clustering
- Expensive software ditto.

- => Need for server side caching.
  - Apache
  - Squid
  - $CALL

Apache as cache

- Not what Apache is built for.
- Not what Apache is good at.
- Square peg, round hole.

Squid is a cache...

- ...built for client side caching.
- Lots of unwanted functionality.
  - Authentication.
  - FTP
- 1980'ies software design.
- Fragile and buggy.
- Wrong configuration features.
- Wrong policy decisions.

$CALL

- Akamai will happily take your money.
- But you loose control.
- You're stuck with their service.

Enter Varnish...

- Built for server-side caching.
- 2006 software design:
  - Multi-Processor / Multi-Core / Multi-Thread.
  - Virtual Memory, shared memory.
  - Sendfile, Accept filter.
- Content provider features:
  - Instant URL invalidation.
  - Full Policy Control.

Content Manager Features

- URL invalidation with no delay
  - Regexp matching
- TTL control (via VCL)
- Load/Situation mitigation (via VCL)
- Shared Memory Log
  - Fast special purpose log-tailers

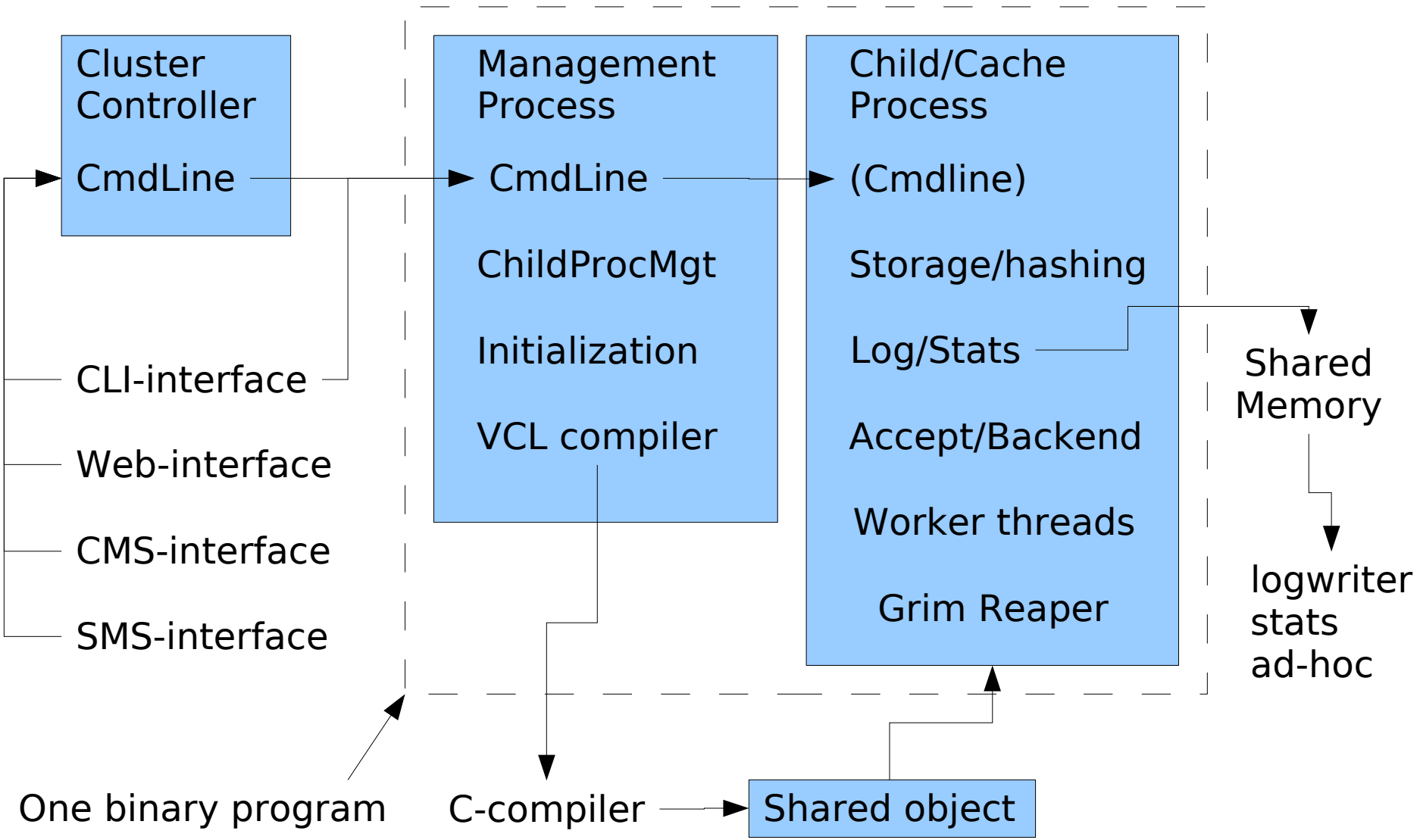
OaM

- Cluster controller
  - Invalidate URL on all caches instantly
  - Change VCL on all caches simultaneously.
  - Aggregate statistics
- Command Line Interface
  - Remote access via ssh/mobile phone
  - Interrogation/configuation/investigation

Performance Design

- Don't copy data if we can avoid it
  - Avoid text-processing headers
- Convert Chunked encoding to Direct
- Also cache ”cannot be cached” info
- Maximize session usage
  - Pass-through mode understands chunked encoding etc.
  - Pipe mode for weird stuff (selected by VCL)

Varnish Architecture



Varnish Config Language

- Simple domain specific language
  - Compiled via C language to binary
    - Transparantly!
  - Dynamically loaded
  - Multiple configs loaded concurrently
- Instant switch from one VCL to another.
  - Can be done from VCL(!)

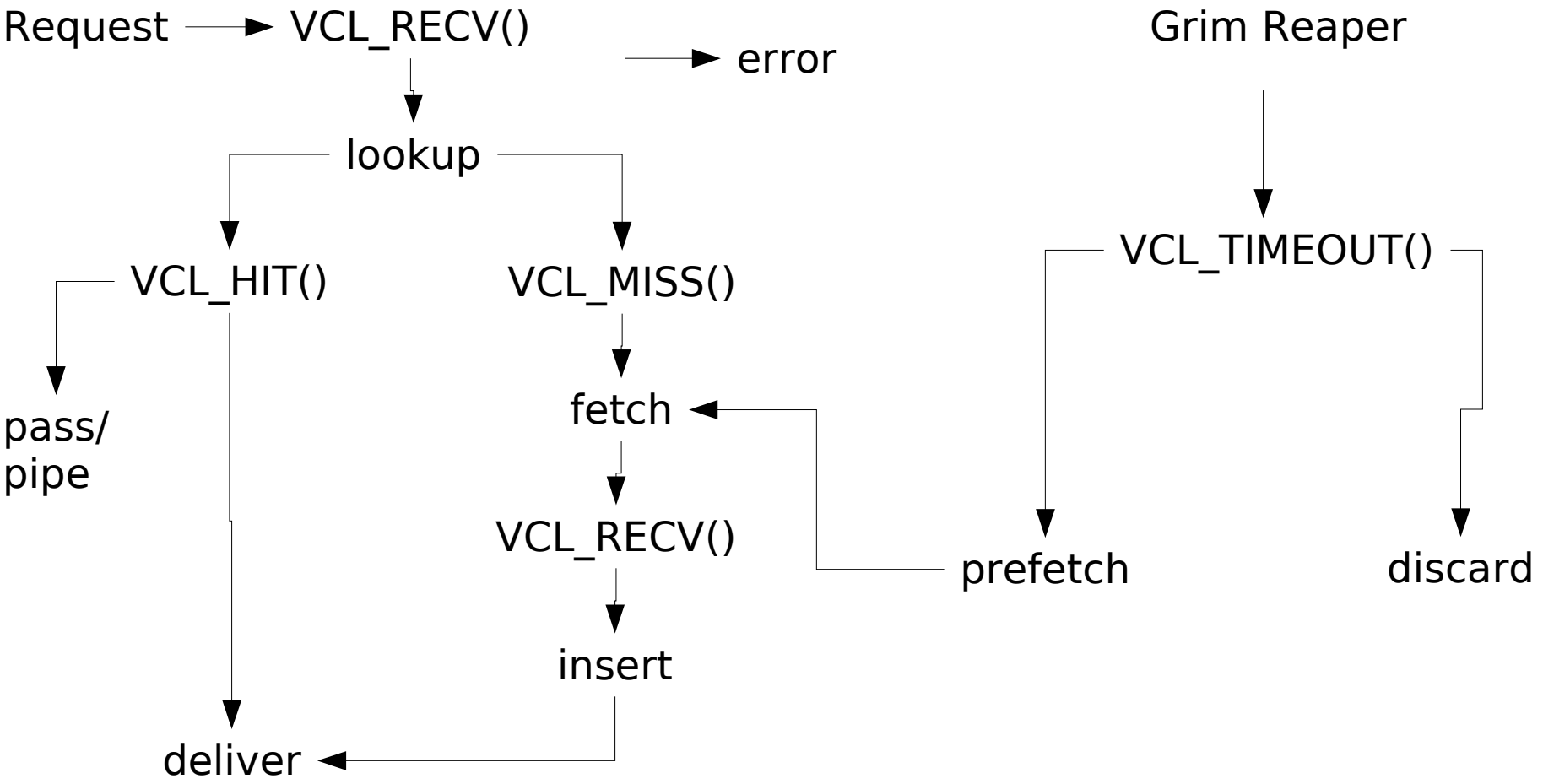
VCL example

```
if (client.ip in 10.0.0.0/8) {
        pass;
}

if (req.url.host ~ "cnn.no$") {
        rewrite req.url.host ”cnn.no” ”vg.no”
}

if (!backend.up) {
        if (obj.exist) {
                set obj.ttl += 10m;
                deliver;
        }
        switch_config ”ohhshit”;
}
```

VCL structure



Storage

- Storage methods are plugable
  - Malloc
    - Simple malloc(3)/free(3)
  - File
    - Mmap(2)'s one file
  - Tar
    - Loads tar(1) file with static objects

storage_file

- Uses a single file
- Most recently accessed/First fit allocation
- Mmap(2)'ed into process
- Use sendfile(2) if available
- Avoids filesystem namespace-lookup
- Not Persistent
  - Avoids integrity check.

Logging

- Logging to shared memory
  - No slowdown for real workload
- Daemons tailing shm generate ”real” output:
  - Apache format
  - Custom format
  - Realtime views

Statistics

- Stored in shared memory
- Programs can monitor & present data

Status

- All the ”technical” code is written
  - Live 'smoke test' today with www.vg.no
- ”Political” code needs to be filled in
  - Mostly VCL runtime parts.
- Profiling and performance tuning

Performance

- Lab test:
  - Dual Opteron Dual Core, 4G, 1 disk
  - FreeBSD 6-Stable (KSE threads)
  - 100.000 Objects @ 8K
  - 4 pipelined clients picking random objects
  - 97% hitrate (due to object expiry)
  - 200+ Mbit/s, 3000 req/sec.
  - CPU load ~2.0

Performance

- Live test:
  - Dual Opteron 2.4 GHz, 4GB RAM
  - FreeBSD 6-Stable, accept filters
  - www.vg.no Traffic, 3000-4000 objects
  - 95-97 % object hit rate
  - 1300-1400 req/sec
  - 100 Mbit/s (Filled the pipe)

Performance

- Live test:
  - Dual Opteron 2.4 GHz, 4GB RAM
  - FreeBSD 6-Stable, accept filters
  - www.vg.no Traffic, 3000-4000 objects
  - 95-97 % object hit rate
  - 1300-1400 req/sec
  - 100 Mbit/s (Filled the pipe)
  - **87% Idle CPU**

Varnish

- BSD license
- http://varnish.projects.linpro.no
- Sponsored by Verdens Gang
- Volunteers and testers needed ~juli'06