# VARNISH

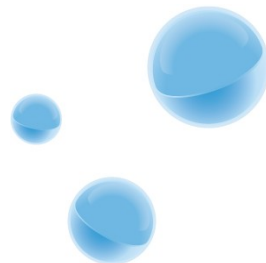
Makes Websites Fly

# Varnish Cache and you

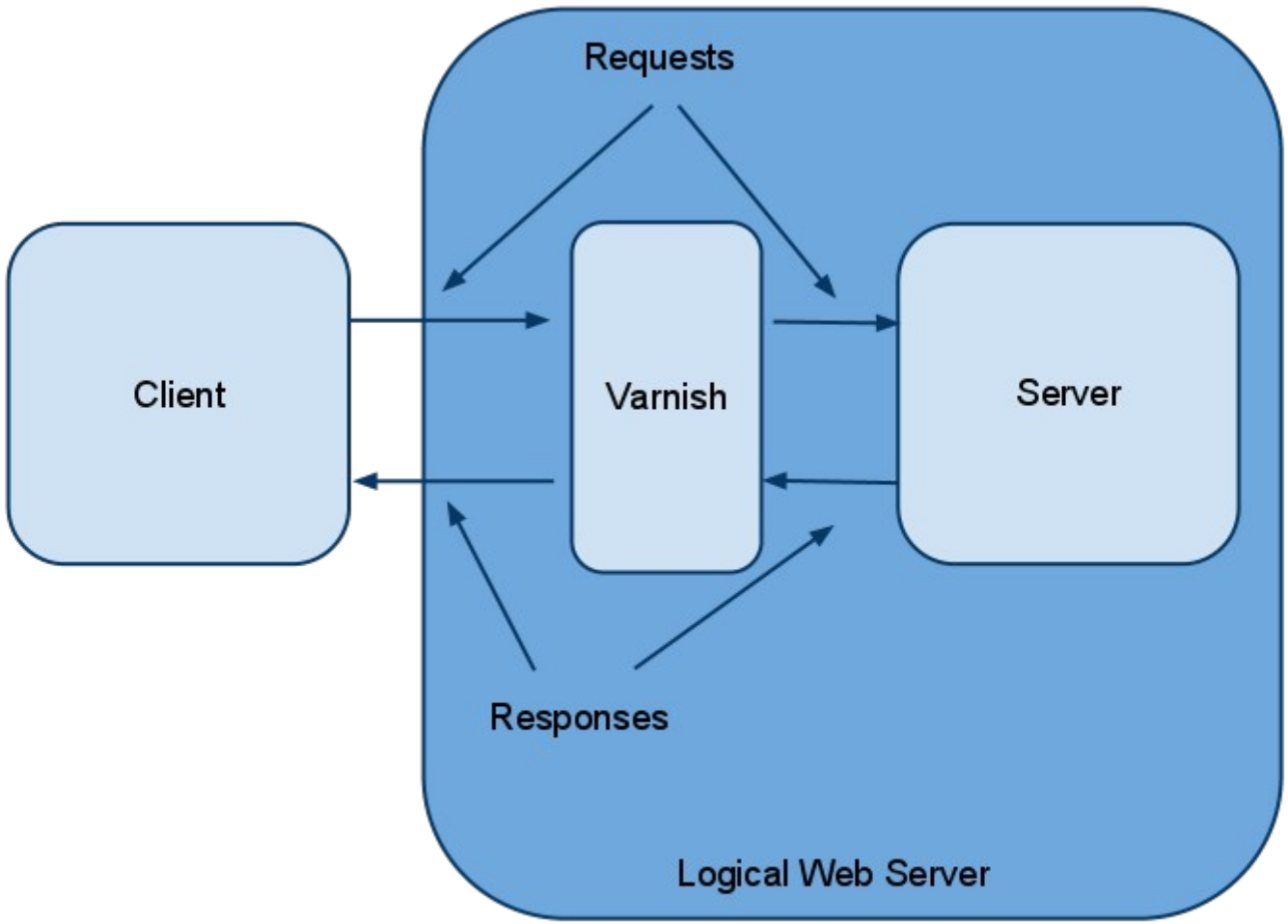
Kristian Lyngstøl
Product Specialist
Varnish Software AS

Oslo, February, 2012

Slides at http://kly.no/presentations/



# Web application accelerator




or a *caching HTTP proxy*

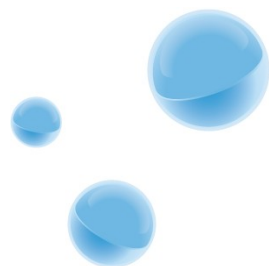
# Speed

- It's fast enough not to worry about it

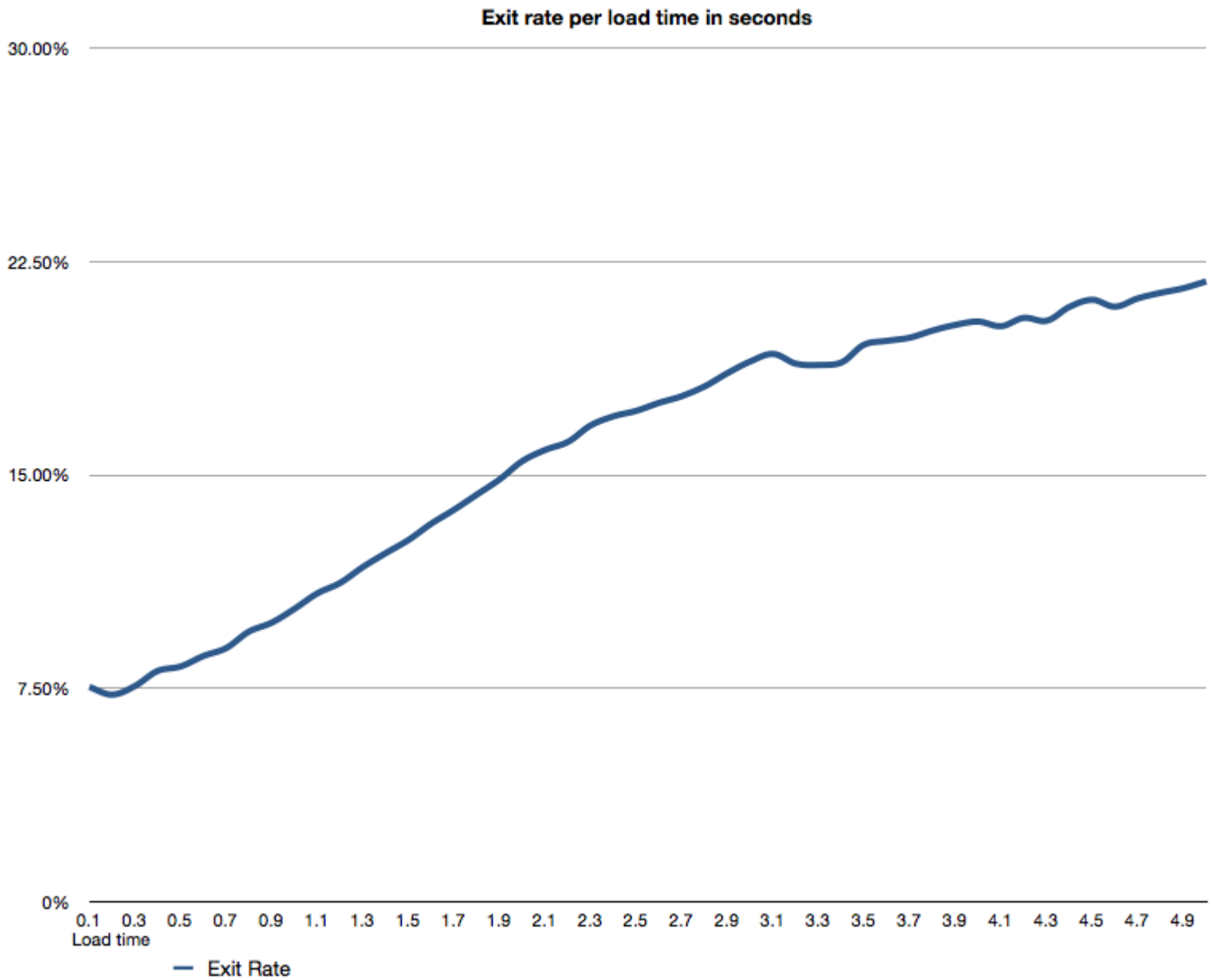
# It matters

- Latency, robustness and flexibility matters

- Tiny site
- Huge site
- Behind CDN
- Not in production
- Already in production
- Too complex
- I'll fix it later (along with world peace)

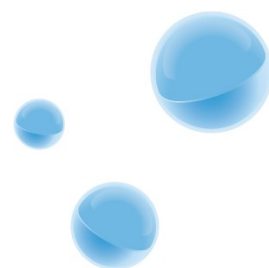


# It matters



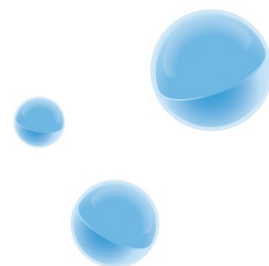

Courtesy of Artur Bergman, http://www.flickr.com/photos/crucially/3716344792/

Why wait until it becomes a problem?



# Organize

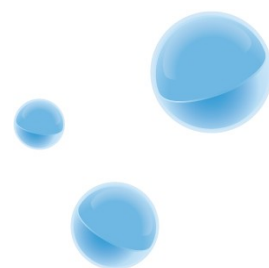
- Static content
  - Images, js, css, etc
- Semi-dynamic content
  - Front page, content pages, articles, etc
- Dynamic content
  - Administration UI, survey forms, interactive chat sessions, etc
- Most web-content is not dynamic



# The Architecture of Varnish

- 64 bit - support 32 bit
- Heavily multi-threaded
- Work with the OS
- Avoid synchronization operations if possible

**Let the sysadmin figure out the policy**

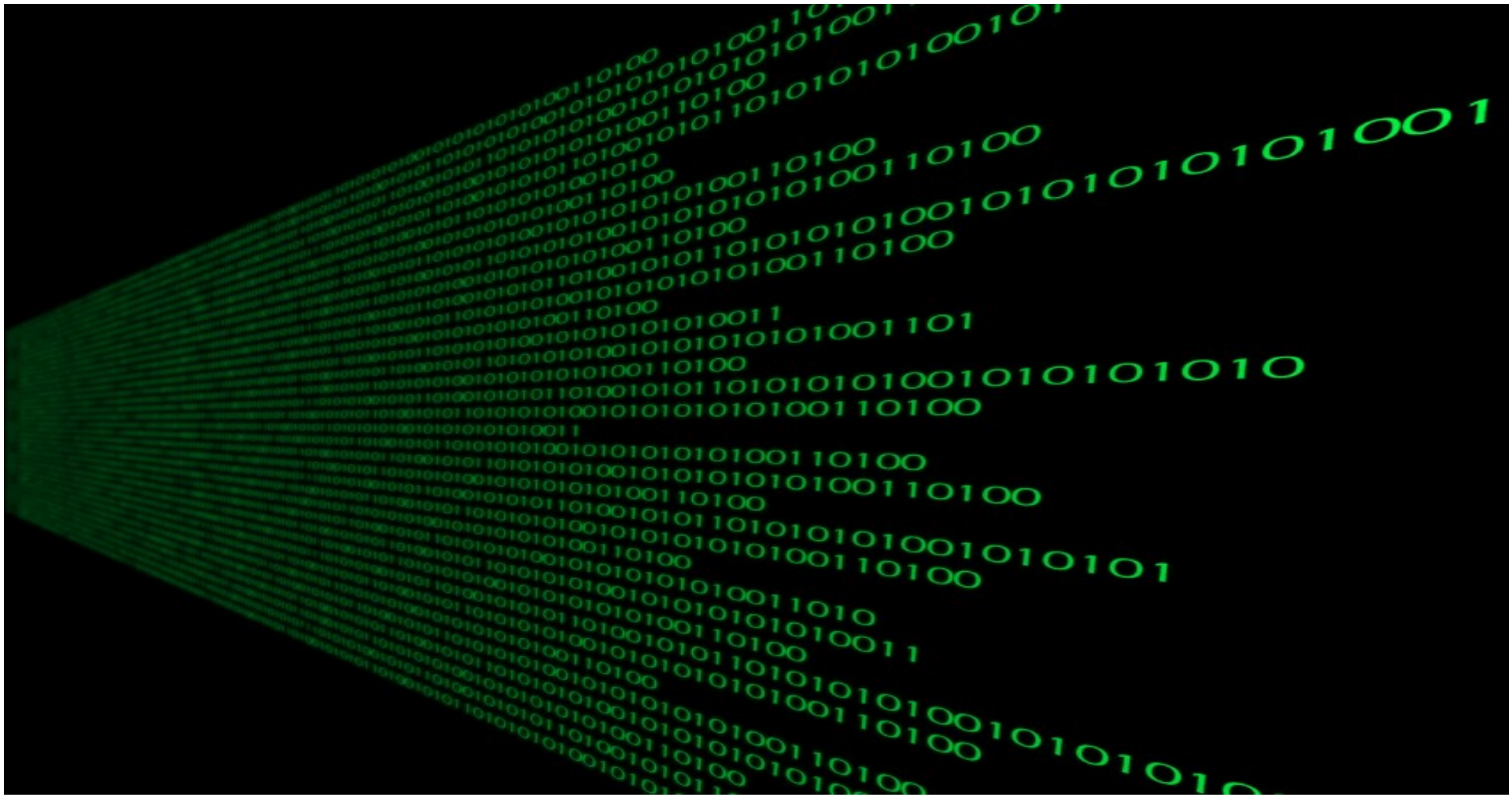


# Welcome to the future

- Less than X GB of active data? Use memory.
- Otherwise: Use SSD storage
- Don't “tune” legacy systems – upgrade
- You can cache everything
- Using 32bit is silly, a waste of everyone's time, and kills kittens
  - Why do you hate the kittens!?

Photo by: Sasan Geranmehr (cc-by-sa)

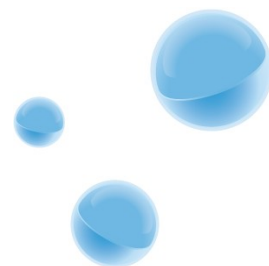
# Config to Code



Policy based configuration
VCL

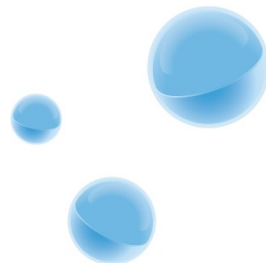
# Varnish Configuration Language

- VCL provides mechanisms
- Administrators provide policy



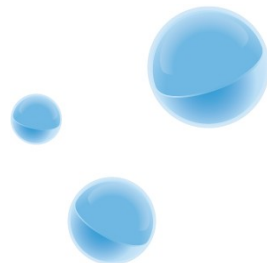
# Varnish Configuration Language

- An interface to the inner workings of Varnish
- Transformed to C, compiled and linked in
- Fast run-time configuration switching



# Varnish Configuration Language

- (Did I mention it was fast?)

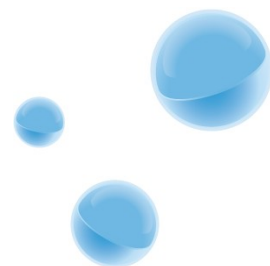


# Varnish *can* save the request

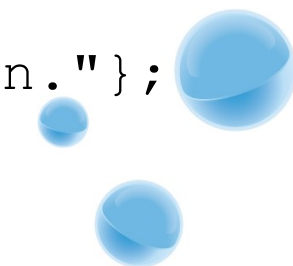
Retry, rewrite, reroute failed TX

# How we can save a request

1) Try to fetch the object from app-server #1
2) Rewrite the request, try app-server #2
3) Try to find a stale object in cache and serve it
4) Serve an error



```
sub vcl_recv {
    set req.backend = appserver1;
    if (req.restarts == 1) {
        set req.backend = appserver2;
    }
    set req.grace = 30s;
    if (!req.backend.healthy) {
        set req.grace = 1d;
    }
}
sub vcl_error {
    if (req.restarts == 0) {
        return (restart);
    }
    set obj.http.Content-Type = "text/plain";
    synthetic{"Re-fill your coffee cup and try again."};
}
```

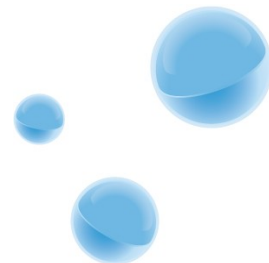


# Fully programmable

Can do awesome tricks!

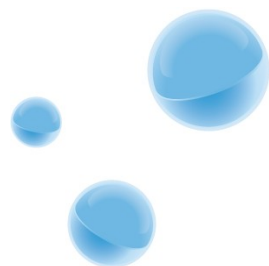
# VCL trick #731

```
sub vcl_recv {
 if (req.http.host ==   “example.com” &&
     req.url ~ “^/fun/” &&
     ( req.http.referer !~
       “^http://example.com/”) ) {
         error 403 “No hotlinking please”;
}
```



# Varnish Modules

- Extends VCL and largely replaces in-line C
- Present in 3.0, used in production
- Much improved in Varnish 3.0+1
- Modules include
  - Redis, curl, digest, crashhandler, memcached, URL code, URL sort, variable support, DeviceAtlas, header manipulation, authentication, and much more.
  - https://www.varnish-cache.org/vmods

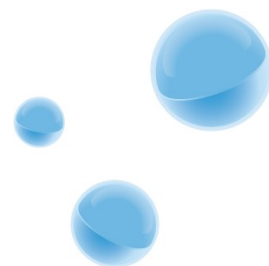


# Has an excellent load balancer

Health probes, sticky sessions++

# Bans

- Invalidate objects
- Instantly added – no blocking
- Ban on (almost) anything!
- Ban lurker
- Ban from VCL
- Ban from CLI

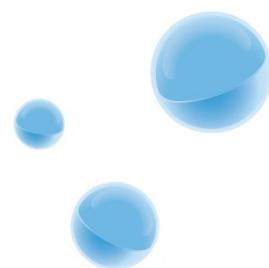


# Purge examples

```
if (req.request == “PURGE”) {
        purge(“obj.http.Cache-Channel ~” req.url);
        error 200 “Purge added.”
}

if (req.request == “PURGE” && client.ip ~ purgers) {
     purge(“obj.http.Cache-Channel ~” req.url);
}

$ varnishadm -T localhost:6082 purge req.url == /
```

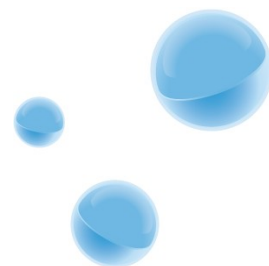


# Ban examples

```
if (req.request == “BAN”) {
        ban(“obj.http.Cache-Channel ~” req.url);
        error 200 “Ban added.”
}

if (req.request == “BAN” && client.ip ~ banners) {
     ban(“obj.http.Cache-Channel ~” req.url);
}

$ varnishadm -T localhost:6082 ban req.url == /
```

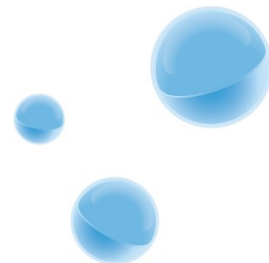


# Fully manageable


powerful CLI - simple protocol

# Logging Varnish

- Most logs are never read
- Logs to shared memory
- Extensive information – cheap
- Separate tools to analyse the data
- Flexible



# Logging tools

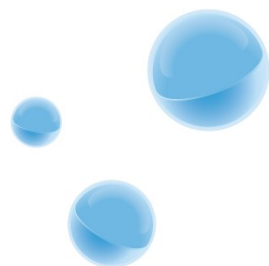
- Varnishlog
- Varnishstat
- Varnishhist
- Varnishtop
- Varnishncsa
- Varnishsizes

```
37+13:04:10
Hitrate ratio:       10      100      125
Hitrate avg:     0.9875   0.9858   0.9858

   436566336       247.99       134.58 Client connections accepted
  3738238196      2149.92      1152.41 Client requests received
  3647901706      2113.92      1124.56 Cache hits
     2995698         2.00         0.92 Cache hits for pass
    71691301        25.00        22.10 Cache misses
    83240848        36.00        25.66 Backend connections success
       35457         0.00         0.01 Backend connections failures
    70714372        31.00        21.80 Backend connections reuses
    82628227        35.00        25.47 Backend connections recycles
        2475            .            .   N struct srcaddr
        1097            .            .   N active struct srcaddr
```

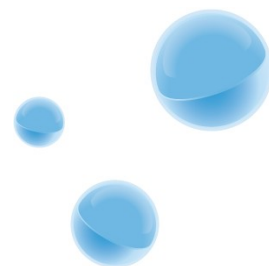
# Further features

- ESI, Gzip, grace, saint mode
- Purging, custom log entries, in-line C
- Streaming
- Multi-tier cache logic, customized and synthesized messages, access control lists
- Native IPv6 (and IPv4) support
- Etc etc etc.



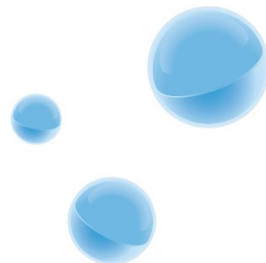
# Future development

- Better VMOD packaging
- Speed improvements (always speed)
- More streaming delivery
- Reduced memory footprint
- Extended ESI support
- Better support for conditional GET requests
- ...



# Recapitulation

- Caching HTTP proxy
- Configurable state machine with a DSL – no *one trick pony*
- Deployed both in front and in the back. Everywhere there is HTTP
- http://www.varnish-cache.org
- http://www.varnish-software.com



# Contact information

Kristian Lyngstøl

Product Specialist

kristian@varnish-software.com

+47 99014497

http://www.varnish-software.com