UPLEX
understanding complex systems

# Lesser known VMODs

(some of them)

VUGX
December 3rd 2015

Geoffrey Simmons
Nils Goroll

UPLEX
understanding complex systems

## Agenda

- Who we are
- vmod_re
- vmod_vslp
- vmod_dcs
- others

## Who we are

UPLEX
understanding complex systems

- 5+ tech people company
  - 3 C-Hackers
- Understand→ optimize, fix, secure
- Varnish: Support, Ops, Consulting, Development (vmods, core)
- Simple business model: €/h
- Most outdated website, no twitter, no blogs, no frills

UPLEX
understanding complex systems

## vmod_re

# VMOD re

**regex matching and backref capture**

understanding complex systems

```
# Standard backref capture with regsub()
# Write the value of the foo cookie to the Foo header

sub vcl_recv {
  if (req.http.Cookie ~ "\bfoo\s*=\s*\w+")) {
    set req.http.Foo
        = regsub(req.http.Cookie, ".*\bfoo\s*=\s*(\w+).*$", "\1");
    }
}
```

- **Two regex matches (condition and regsub)**
- **regsub() must match the entire string**
  - **`'.*'` at start may cause deep backtracking**
- **Cumbersome and verbose**

# VMOD re

## regex matching and backref capture

understanding complex systems

```
import re;

sub vcl_init {
  new foomatcher = re.regex("\bfoo\s*=\s*(\w+)");
}

sub vcl_recv {
  if (foomatcher.match(req.http.Cookie)) {
    set req.http.Foo = foomatcher.backref(1, "fallback");
  }
}
```

**`backref()`: nth subexpression from most recent call to `match()`**

# VMOD re

## dynamic (runtime) regex compilation

```
import re;

sub vcl_backend_response {
  if (bematcher.match_dyn(beresp.http.URL-Regex, bereq.url)) {
    set beresp.http.Foo = bematcher.backref(1, "fallback");
  }
}
```

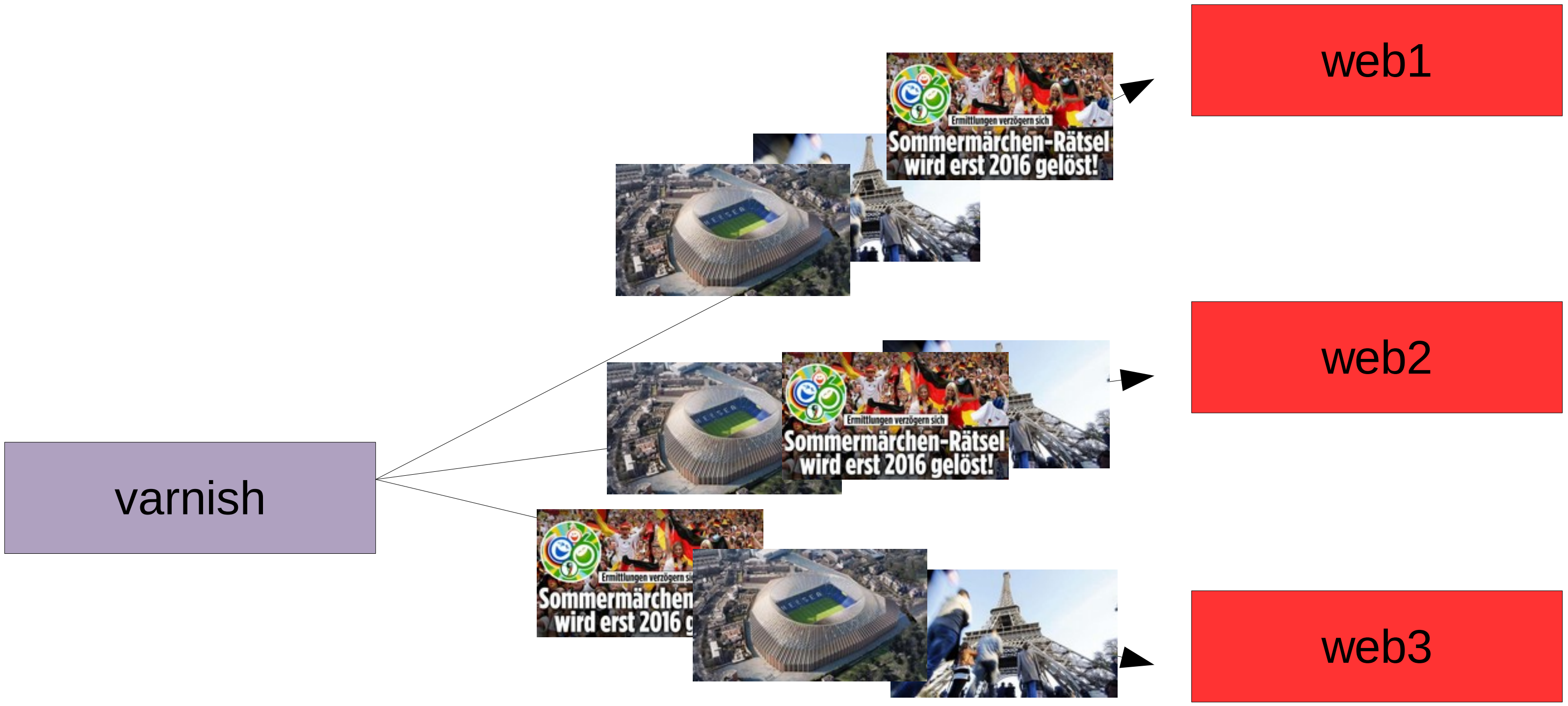
**`match_dyn()` compiles the regex in the first parameter on every call**

`https://code.uplex.de/uplex-varnish/libvmod-re`

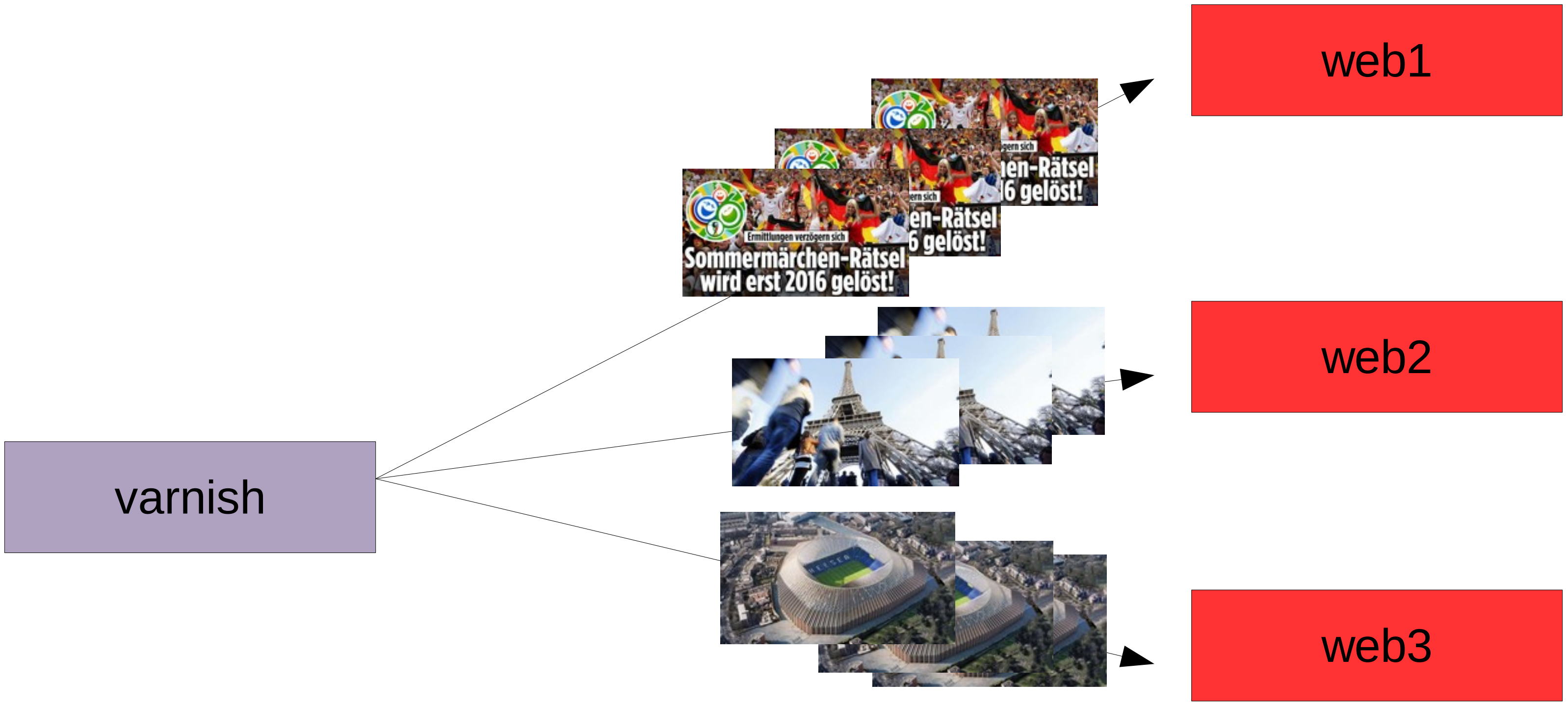
## VSLP

- Main motivation
  - Sharding
  - Clustering
- The Problem with modulo hashing
- Consistent Hashing
- More

# Round-robin


varnish
web1
web2
web3
Ermittlungen verzögern sich
Sommermärchen-Rätsel wird erst 2016 gelöst!

## Backend sharding


varnish
Ermittlungen verzögern sich
Sommermärchen-Rätsel wird erst 2016 gelöst!
web1
web2
web3

- **Hash by URL / ID etc.**
- **Can be done with hash director**

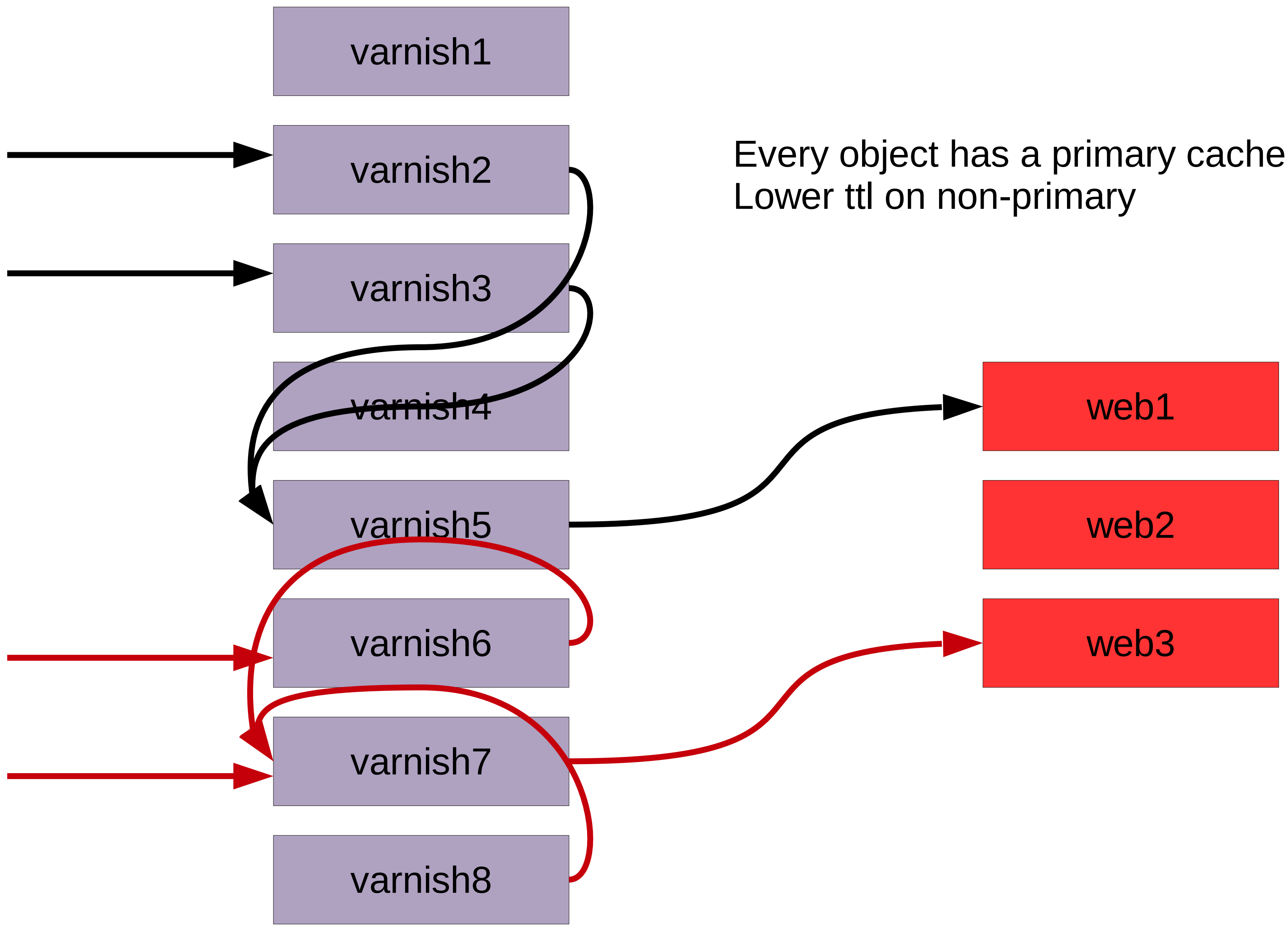
## Consistent hashing

- Standard hash director:

  ```
  backend = backends[ hash % n ]
  ```

  - If number/health state of backends changes, most/all hashes get a new backend
- Consistent hashing:
  - Minimal disturbance of sharding to backends

# Clustering



Every object has a primary cache server
Lower ttl on non-primary

## VSLP

- Consistent hashing
- Hash with CRC32, SHA256, RS
- Rampup
- pre-warm aka altsrv_p
- Use nth-backend from circle : restarts
- `https://code.uplex.de/uplex-varnish/libvmod-vslp`

UPLEX
understanding complex systems

## vmod_dcs

## VMOD dcs

### device classifier

understanding complex systems

```
import dcs;

sub vcl_recv {
  # Get a type name derived from User-Agent
  set req.http.X-UA-Type = dcs.type_name(dcs.classify());

  # Get a meta class for the derived type
  set req.http.X-UA-Class = dcs.type_class(dcs.classify());
}
```

- **maps User-Agent patterns to types and meta-classes**
- **C code is generated from the pattern DB to form a custom matcher.**
  - **scans the string once from left to right**
- **Benchmarks: > 200,000 matches/s (5 µs/match)**

## VMOD dcs

### pattern DB

```
# FORMAT: <pattern> <tab> <type>
#
# substr  type
# substr*substr*substr type
# !substr*substr type

mozilla*!android*tablet;*firefox/       Tablet
mozilla*android 2.1*movistar link       Mobile Phone
```

- **Use cases in production:**
  - **Device detection**
  - **Browser version compatibility check**
- **Supports Varnish versions 2, 3 and 4**

https://code.uplex.de/uplex-varnish/dcs_classifier

UPLEX
understanding complex systems

## Thanks!

UPLEX understanding complex systems

Questions?

- nils.goroll@uplex.de, +49-170-2723133
- geoff.simmons@uplex.de, +49-176-63690917

## extra slides

UPLEX
understanding complex systems

## Clustering benefits

- For n servers
  - Effective cache : x times larger
  - 1/n backend requests
  - ...

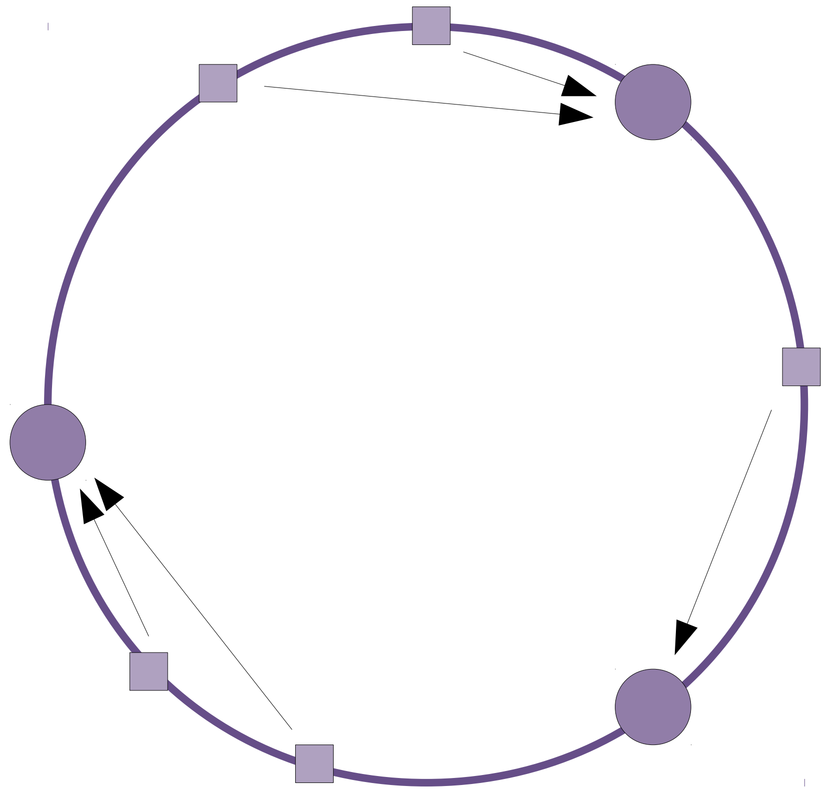
## How to cluster

UPLEX
understanding complex systems

- Can be done with the hash director
- Cluster:
  - Define director of varnish servers
  - Select backend
  - If self → go to real backend
  - Otherwise, recurse
  - Lower ttl for recursive requests

# Consistent Hashing

- **Minimizes re-mapping when backends change health / are added/removed**



**Actual implementation uses many more nodes**

## VSLP more

- **More applications:**
  - **Hash by IP:**
    - **Simple IP persistence**
  - **Hash by constant value**
    - **Priority Server**
  - **Hash by cache server name (in a cluster)**
    - **Each cache server uses a different backend**
  - **… how many more to come?**