UPLEX
understanding complex systems

# IMS
## Backend Conditional Requests
## aka Cache Refresh/Refresh

VUG5
March 23, 2012

Geoffrey Simmons
Nils Goroll

VARNISH
Makes Websites Fly

UPLEX
understanding complex systems

## Agenda

- brief history
- experimental-ims
  - How it works
  - Refresh and storage
- Configuration
  - keep, stale_obj
  - vcl_stale(), vcl_refresh()?
- Too complex?

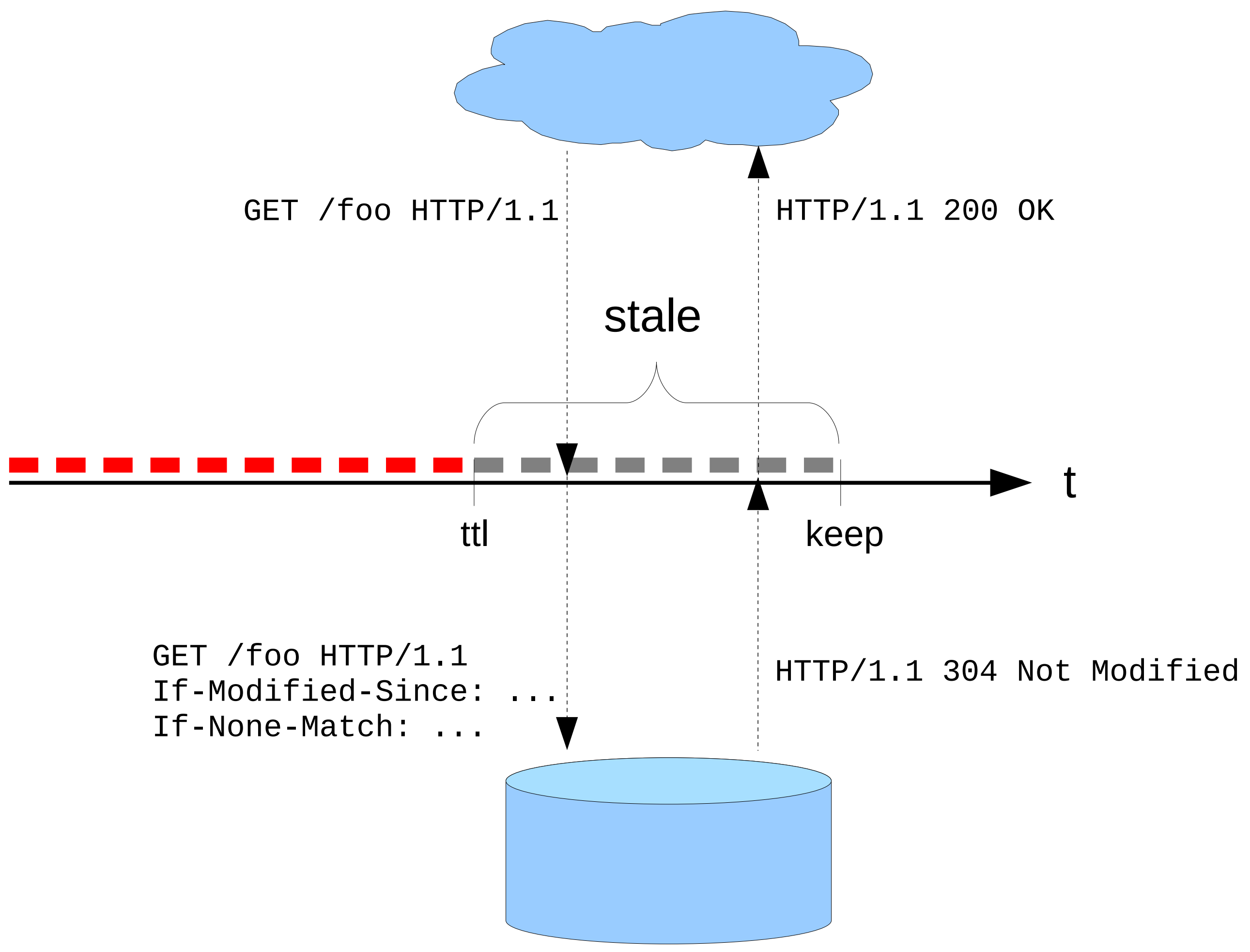
## Brief History

UPLEX
understanding complex systems

- 2010-09-10
  - Received first Rackspace Implementation
- 2010-09-23
  - Proposed design on varnish-dev
- 2011-03-01
  - First patch posted
- Since 2011-06-02
  - Maintaining git branch experimental-ims

# experimental-ims

UPLEX understanding complex systems

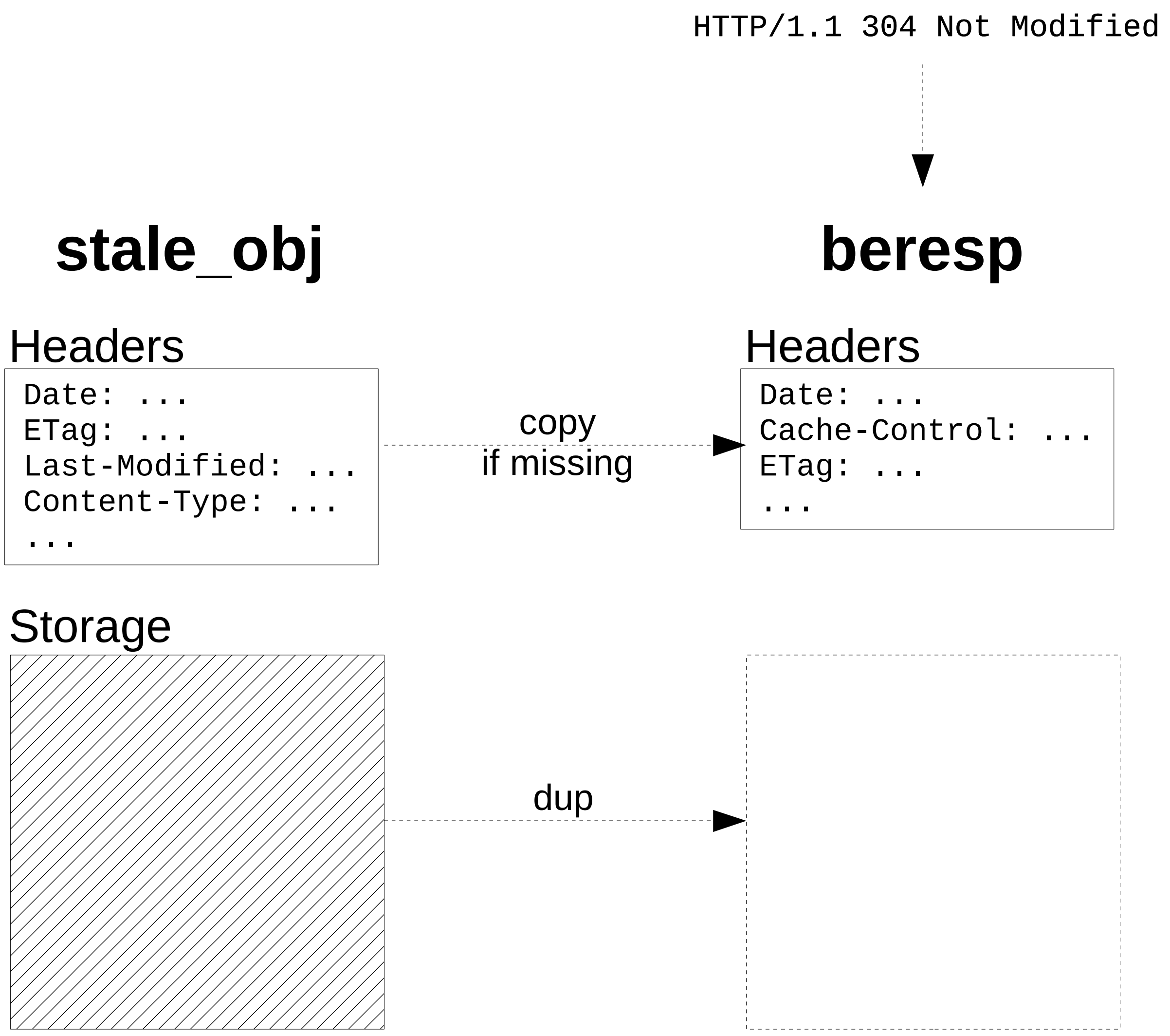


```
GET /foo HTTP/1.1
If-Modified-Since: ...
If-None-Match: ...
```

```
HTTP/1.1 304 Not Modified
```

## Refresh & storage

UPLEX understanding complex systems

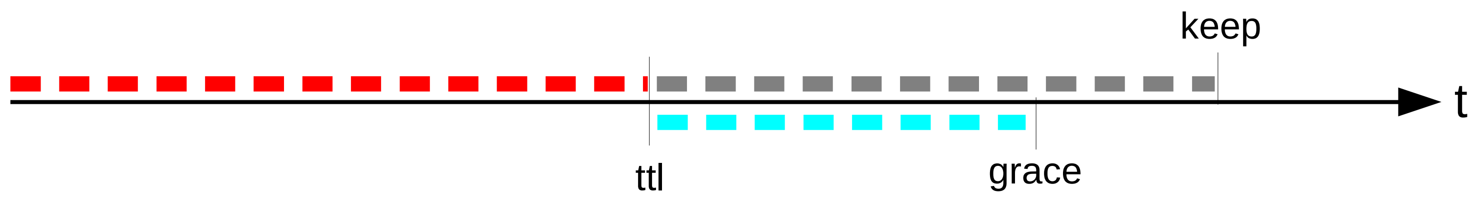


## Storage dup

UPLEX understanding complex systems

- dup == copy
  is unavoidable for `-spersistent`
- dup for other stevedores
  - Should be:
    - New object points to stale_obj's storage
    - Refcounts for storage
  - Currently in experimental-ims:
    - Copy for <u>all</u> stevedores

## keep & grace



**In experimental-ims:**

- **Both intervals begin after ttl**
  - **keep == grace, keep < grace, keep > grace all possible**
- **Expiration after ttl + max(keep, grace)**
- **grace has priority in overlapping intervals**

## stale_obj

UPLEX understanding complex systems

**Currently in experimental-ims:**

```
sub vcl_miss {
    if (stale_obj) {
        set bereq.http.X-Foo = stale_obj.http.X-Bar;
    }
}
```

- **All fields read-only**
  - **Otherwise just like obj**
- **In vcl_miss(), vcl_fetch(), vcl_error()**
  - **May or may not exist!**

## stale_obj

UPLEX
understanding complex systems

```
# DON'T DO THIS!
sub vcl_miss {
    set bereq.http.X-Foo = stale_obj.http.X-Bar;
}

# Always check stale_obj for existence
sub vcl_miss {
    if (stale_obj) {
        set bereq.http.X-Foo = stale_obj.http.X-Bar;
    }
}
```

- No way to prevent this error
  - SHM log records a VCL_error message
- Does stale_obj need to be exposed?

## vcl_stale(), vcl_refresh()

understanding complex systems

**vcl_stale() ↔ vcl_miss()**
**vcl_refresh() ↔ vcl_fetch()**

- **stale_obj always exists**
  - **stale_obj could just be obj**
- **Two new vcl_subs()?**

## Generalized vcl_stale()

UPLEX understanding complex systems

- Either the grace or stale/keep scenario
  - Alternative to vcl_hit() & vcl_miss()
  - Just one timer after ttl?
- Idea: Pull C-logic into vcl
  - pseudocode for HSH_Lookup
    (after searching for the "best" object)

```
if (oc && (now < ttl))
    next step is HIT;
else if (oc && (now < ttl + keep))
    next step is STALE;
else
    next step is MISS;
```

## vcl_stale() in default.vcl

UPLEX
understanding complex systems

```
sub vcl_stale {

    /* existing grace logic */
    if (req.busy || !req.backend.healthy)
        return (deliver);
    return (fetch)
}
```

- **deliver: deliver the stale object**
- **fetch: attempt a (possibly conditional) fetch**
- **pass, error, restart: as usual**

# vcl_stale()

understanding complex systems

```
sub vcl_stale {
    /* only part of the current grace logic */
    if (!req.backend.healthy)
        return(deliver);

    /* override conditional requests */
    if (req.url ~ "^/always-fetch") {
        unset bereq.http.If-Modified-Since;
        unset bereq.http.If-None-Match;
        return(fetch);
    }

    /* deliver for restarts */
    if (req.url ~ "^/app" && req.restarts > 0)
        return (deliver);
}
```

## vcl_refresh() ?

UPLEX understanding complex systems

**Expose the stale object after fetch?**

**If so ...**

```
sub vcl_fetch {

    if (stale_obj && stale_obj.hits > 10000)
        set beresp.ttl = 30m;
}

/* ... or ... ? */

sub vcl_refresh {

    if (obj.hits > 1000)
        set beresp.ttl = 30m;
}
```

UPLEX
understanding complex systems

## Thanks!

Questions?

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

VARNISH
Makes Websites Fly