



Serving Wikipedia with ATS

ATS Summit Sunnyvale, California

Emanuele Rocca

Site Reliability Engineer @ WMF

October 8th 2019



Traffic Server is now being used to serve
Wikipedia

Outline

- ▶ Introduction to Wikimedia Foundation
- ▶ Old CDN Architecture with Varnish
- ▶ New CDN Architecture with ATS
- ▶ Work done
- ▶ Conclusions

Wikimedia Foundation

Wikimedia Foundation

- ▶ Non-profit organization focusing on free, open-content, wiki-based Internet projects
- ▶ No ads, no VC money
- ▶ Entirely funded by small donors
- ▶ 350 employees (of which 33 SRE and 80 SWE)

The Wikimedia Family



WIKIPEDIA
The Free Encyclopedia



Wiktionary
The free dictionary



WIKISOURCE



WIKINEWS



WIKIBOOKS



WIKISPECIES
free species directory



WIKIDATA



WIKIVERSITY



MediaWiki



**WIKIMEDIA
COMMONS**



**WIKIMEDIA
FOUNDATION**

Why our own CDN?

- ▶ Autonomy
- ▶ Privacy
- ▶ Risk of censorship

Traffic Volume

- ▶ Average: ~100k rps, peaks: ~140k rps
- ▶ Can handle more for large-scale DDoS attacks

DDoS Example



Source: jimieye from flickr.com (CC BY 2.0)

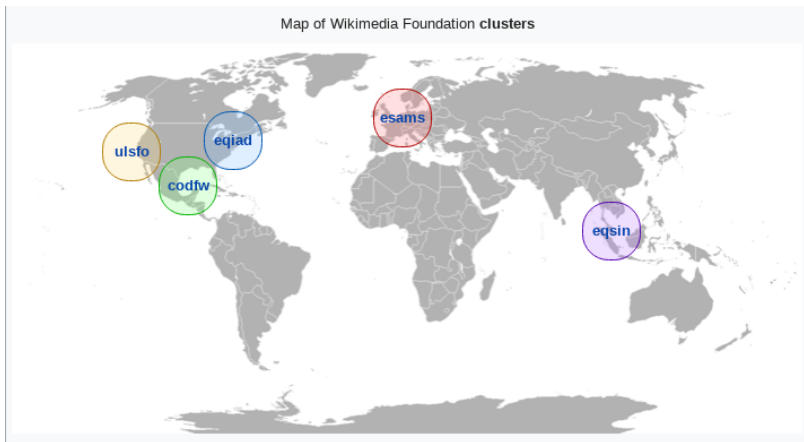
Values

- ▶ Deeply rooted in the free culture and free software movements
- ▶ Infrastructure built exclusively with free and open-source components
- ▶ Design and build in the open, together with volunteers

Build In The Open

- ▶ github.com/wikimedia
- ▶ gerrit.wikimedia.org
- ▶ phabricator.wikimedia.org
- ▶ wikitech.wikimedia.org
- ▶ grafana.wikimedia.org

Cluster Map



eqiad: Ashburn, Virginia - cp10xx

codfw: Dallas, Texas - cp20xx

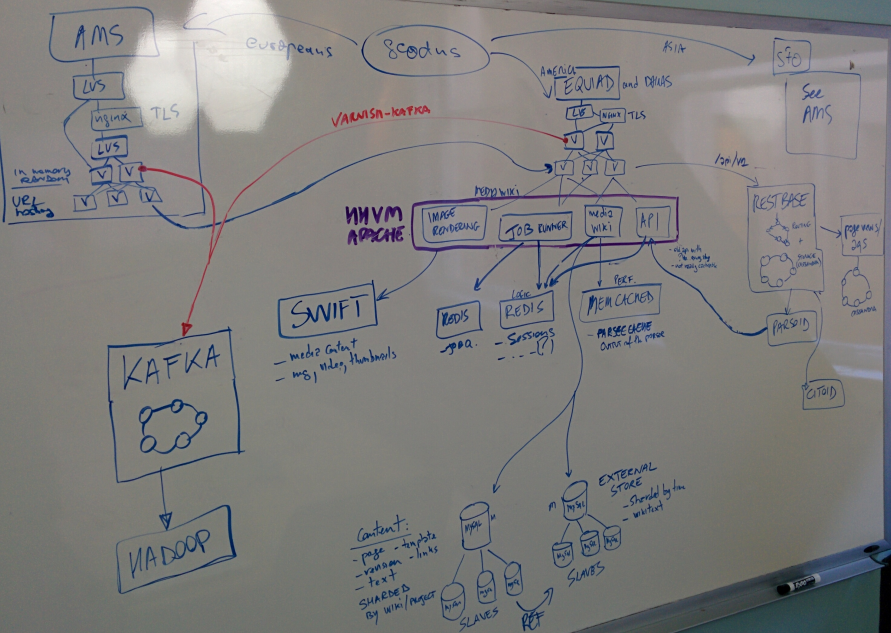
esams: Amsterdam, Netherlands - cp30xx

ulsfo: San Francisco, California - cp40xx

eqsin: Singapore - cp50xx

A day in the life of an HTTP request

- ▶ ~~Geographic DNS Routing~~
- ▶ L4 Load Balancing
- ▶ ~~TCP connection establishment~~
- ▶ ~~TLS Termination~~
- ▶ HTTP Caching
- ▶ L7 Load Balancing



AMS

Europeans

scodus

ASIA

SFO

See AMS

AMERICA EQUAD and DALLAS

VARNISH-KAFKA

In memory redundancy

URL hosting

KAFKA

HADOOP

NHVM APACHE

IMAGE RENDERING
JOB RUNNER
MEDIA WIKI
API

SWIFT

- media content
- ms, video, thumbnails

REDIS

LOKIC REDIS

- sessions
- ...

MEM CACHED

- PARSE CACHE
- errors of the page

REST BASE

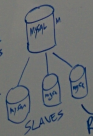
Page views 245

CITOID

EXTERNAL STORE

SLAVES

- Content:
- page - templates
 - version - links
 - text
- SHARDED BY WIKI/PROJECT



REF

Old CDN (Varnish)

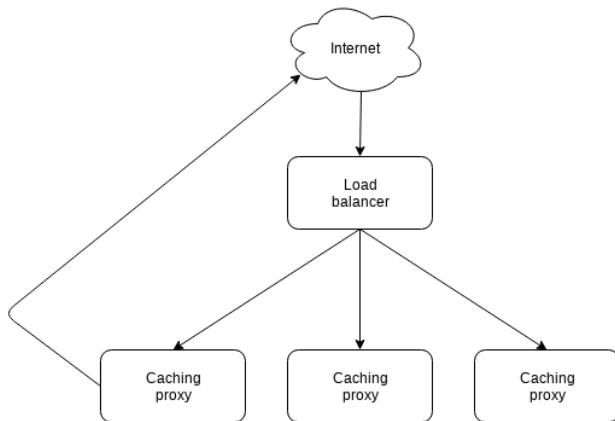
Load balancers and cache servers

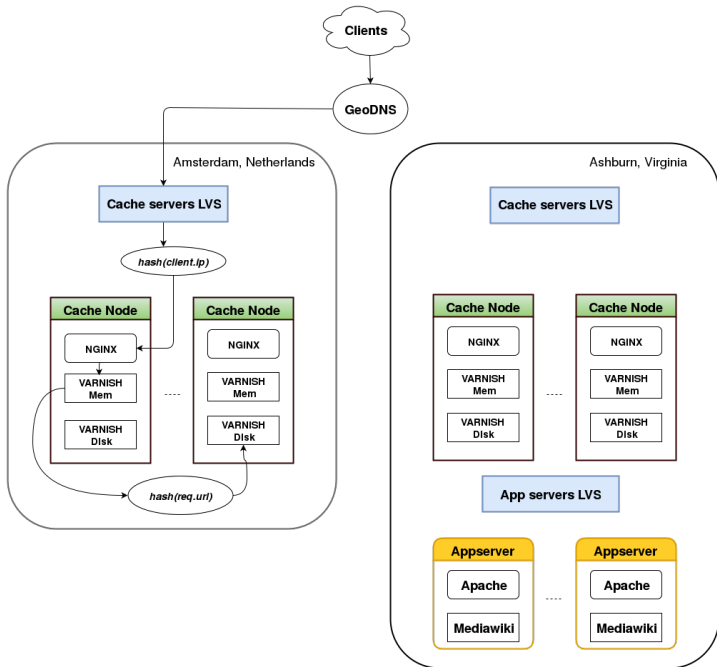
- ▶ Load balancers running Linux Virtual Server
- ▶ HTTP cache proxies running two Varnish instances per server
 - ▶ In-memory: faster, smaller. Effective cache size:
~avg(total mem size)
 - ▶ On-disk: slower, much larger. Effective cache size:
~sum(total disk size)

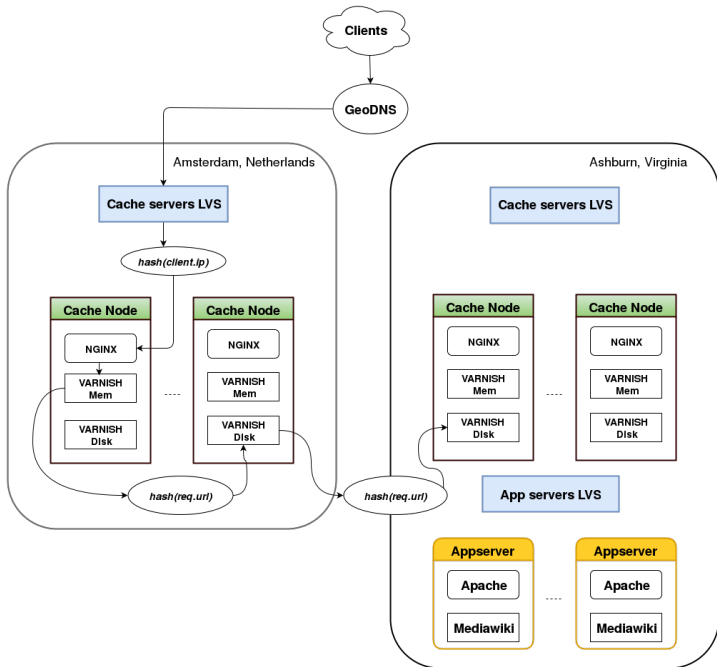
Load balancing

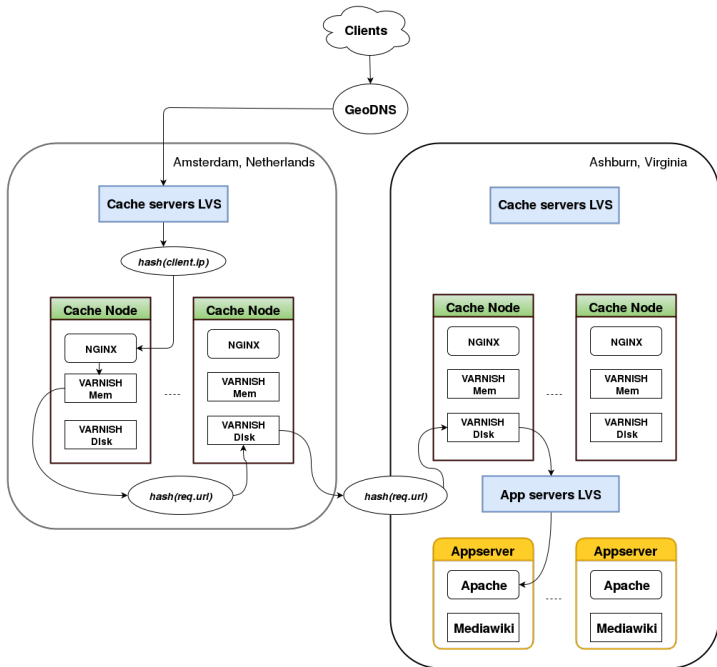
- ▶ All requests go through the load balancer
- ▶ Responses go straight to the client

Load balancing: direct routing









Inter-DC Traffic

- ▶ Encryption between data centers necessary
- ▶ IPsec between cache servers
- ▶ Minimal hitrate on "remote DCs"
- ▶ Architectural constraints due to Varnish not supporting outgoing TLS

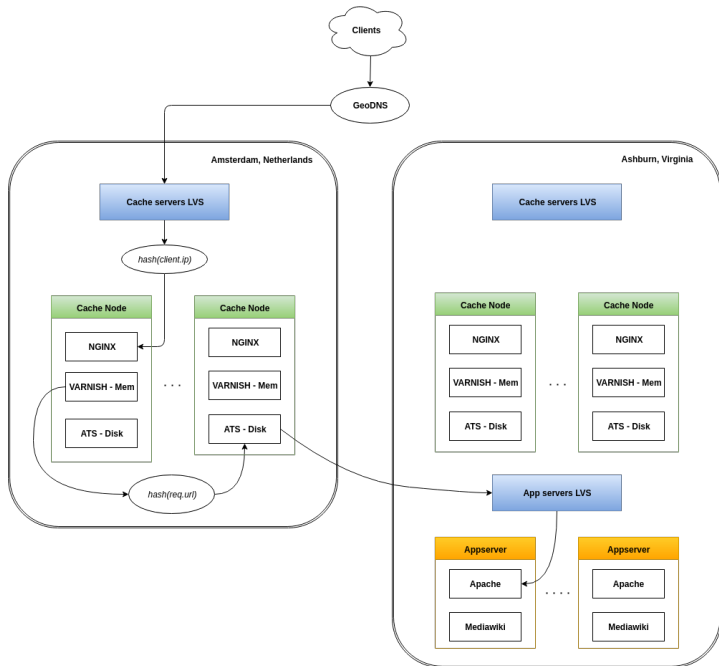
Problems with Varnish

- ▶ Bad scalability issues with the "file" storage backend
- ▶ No TLS support whatsoever, neither incoming nor outgoing
- ▶ Open-core business model, crucial features made proprietary

New CDN (ATS)

ATS Sandwich

- ▶ In the process of replacing Nginx with ATS for TLS termination
- ▶ Work done by my colleague Valentín Gutierrez
- ▶ This presentation focuses on large on-disk caches instead



Simpler architecture

- ▶ IPsec removed entirely thanks to outbound TLS support in ATS
- ▶ No need for caches to be aware of those in other DCs
- ▶ Saving primary DC caches lots of requests
- ▶ No need to change inter-DC routing when depooling a site

Upload cache cluster

- ▶ Multimedia files, OpenStack Swift
- ▶ 42 servers
- ▶ 45k rps
- ▶ Fully converted to ATS

Issues found during transition

- ▶ Segmentation fault in verify_config #4466
- ▶ RAM cache usage growth #5179
- ▶ Segmentation fault due to compress plugin #5787
- ▶ FIFO logfile removed on configuration reload #4635

Detailed transition info:

<https://phabricator.wikimedia.org/T213263>

Text cache cluster

- ▶ Primary wiki traffic
- ▶ 36 servers
- ▶ 100k rps
- ▶ ATS on one of those for production traffic testing
- ▶ Converting the remaining 35 this quarter!

Work done

Debian Packaging

- ▶ 8.x packages backported to Debian Stretch. GCC in Stretch does not support C++17, using Clang instead
- ▶ Upgrading CDN nodes to Buster soon, switching back to GCC
- ▶ Now co-maintaining official Debian packages with Jean Baptiste Favre and Aron Xu

Puppet

- ▶ Remap rules
- ▶ Caching rules
- ▶ Storage
- ▶ Logging
- ▶ Multi-instance support with traffic_layout

<https://github.com/wikimedia/puppet>

Lua

~700 lines of custom Lua code, of which ~350 are tests

- ▶ Per-remap scripts calling `ts.hook()`
 - ▶ MediaWiki request mangling
 - ▶ Path normalization, RFC 3986 section 6
- ▶ Default code calling `do_global__`
 - ▶ X-Cache response header
 - ▶ Force caching
 - ▶ Avoid caching

```
profile::trafficserver::backend::mapping_rules:
```

- type: map
 - target: http://upload.wikimedia.org
 - replacement: https://swift.discovery.wmnet
 - params:
 - '@plugin=/usr/lib/trafficserver/modules/tslua.so'
 - '@pparam=/etc/trafficserver/lua/normalize-path.lua'
 - # decode /
 - '@pparam="2F"'
 - # encode ! \$ & ' () * + , : ; = @ []
 - '@pparam="21 24 26 27 28 29 2A 2B 2C 3A 3B 3D 40 5B 5D"'
 - '@plugin=/usr/lib/trafficserver/modules/tslua.so'
 - '@pparam=/etc/trafficserver/lua/x-mediawiki-original.lua'
- type: regex_map
 - target: 'http://(.*)/w/api.php'
 - replacement: https://api-rw.discovery.wmnet/w/api.php
 - params:
 - '@plugin=/usr/lib/trafficserver/modules/tslua.so'
 - █ '@pparam=/etc/trafficserver/lua/rb-mw-mangling.lua'

Cacheability

- ▶ Initially used heuristics and Negative Response Caching
- ▶ Finally decided to require explicit Cache-Control instead
- ▶ Unset Cache-Control in `do_global_read_response` for what we consider uncacheable, set it for negative responses we want to cache
- ▶ Restore the original CC value in `TS_LUA_HOOK_SEND_RESPONSE_HDR`

Does server permit storing?

- ▶ Wrote a SystemTap probe to inspect cache/no-cache decisions
- ▶ Instrument `is_response_cacheable`
- ▶ Print request details unless `does_server_permit_storing`

Origin server connection establishment

```
probe process("/usr/bin/traffic_server").statement(  
    "state_http_server_open@./proxy/http/HttpSM.cc:1718")  
{  
    server_name = user_string_n(  
        $this->t_state->current->server->name, 128)  
  
    t = &$this->t_state->txn_conf  
  
    printf("%s %d\n",  
        server_name, t->origin_max_connections)  
}
```

appservers-rw.discovery.wmnet 0

swift.discovery.wmnet 0

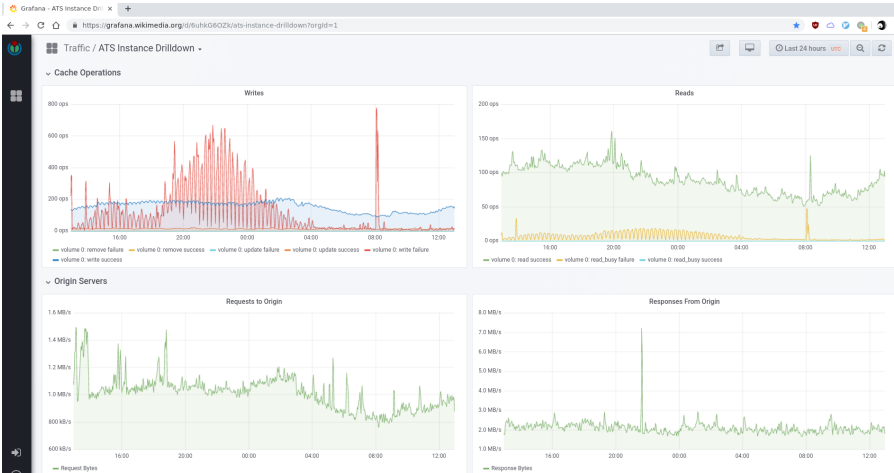
Logging

- ▶ Logging to named pipe
- ▶ Golang program called fifo-log-demux reading from the pipe
- ▶ Multiple clients connecting via Unix domain socket
- ▶ Client program called atslog to inspect logs at runtime
- ▶ Issues:
 - ▶ FIFO logfile removed on configuration reload #4635
 - ▶ Error messages logged if there is no reader

Prometheus integration

- ▶ prometheus-trafficserver-exporter for all stock trafficserver metrics
- ▶ Valentin and I maintain the package in Debian
- ▶ atsbackend.mtail exposing ttfb on a per-origin basis
- ▶ atsmtail.service boiling down to atslog | mtail

Grafana



Thundering herd avoidance

- ▶ Tried collapsed forwarding, decided to go for read while writer instead
- ▶ Conservatively returned 502 upon coalesce timeout expiration (failure to obtain cache open write lock) for a few months. Failing open now
- ▶ Optimal value for `max_open_write_retries` identified instrumenting `state_cache_open_write` with SystemTap
- ▶ How to avoid stalling on uncacheable responses?

Much more!

- ▶ Read-Only /etc #2505
- ▶ Systemd unit hardening
<https://phabricator.wikimedia.org/T200178>
- ▶ Icinga checks
<https://phabricator.wikimedia.org/T204209>

Conclusions

Positives

- ▶ Persistent storage
- ▶ TLS
- ▶ Lua
- ▶ Community! We would like to collaborate even more

Conclusions

- ▶ Wikipedia moving from Varnish and nginx to ATS
- ▶ Currently converting all on-disk caches and TLS terminators
- ▶ Big plans for the future!