Wikipedia's CDN

Research, Engineering, Free Software

Emanuele Rocca
Wikimedia Foundation

March 26th 2018

How does Wikipedia end up on my screen?

Outline

- Wikimedia Foundation
- CDN Ingredients
- In Practice

Wikimedia Foundation

Wikimedia Foundation

Non-profit organization focusing on free, open-content, wiki-based Internet projects.

WMF: what it does NOT do

- Edit Wikipedia
- Use advertisement or VC money

WMF: what it does

- Owns the wikipedia.org domain
- Raises money through donations
- Controls the servers (19 Site Reliability Engineers)
- Develops and deploys software (66 SWE)

Alexa Top Websites

Company	Revenue	Employees	Server count
Google	$89.4 billion	73,992	2,000,000+
Facebook	$40.6 billion	25,105	180,000+
Baidu	$13.4 billion	46,391	100,000+
Wikimedia	$81.9 million	304	1,000+
Yahoo	$1.31 billion	8,500	100,000+

Traffic Volume

- Average: ∼100k/s, peaks: ∼140k/s
- Can handle more for huge-scale DDoS attacks

DDoS Example

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

The Wikimedia Family

WIKIPEDIA
The Free Encyclopedia
Wiktionary
The free dictionary
Wikiquote
WIKIBOOKS
WIKISOURCE
WIKIDATA
WIKINEWS
WIKIVERSITY
WIKISPECIES
free species directory
MediaWiki
WIKIMEDIA
COMMONS
wiki
voyage
WIKIMEDIA
FOUNDATION

Values

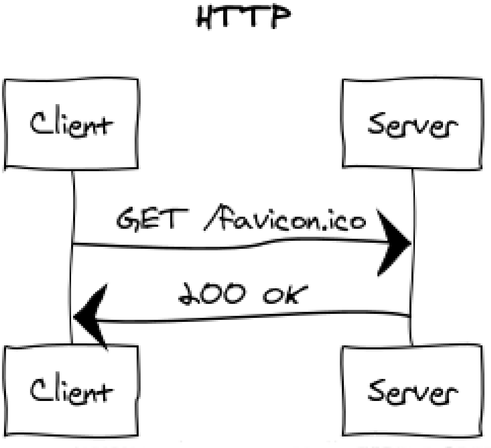
- Deeply rooted in the free culture and free software movements
- Infrastructure built exclusively with free and open-source components
- Design and build in the open, together with volunteers

Build In The Open

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

CDN Ingredients

How does Wikipedia end up on my screen?


HTTP
Client
Server
GET /favicon.ico
200 OK
Client
Server

Thank you! Any questions?

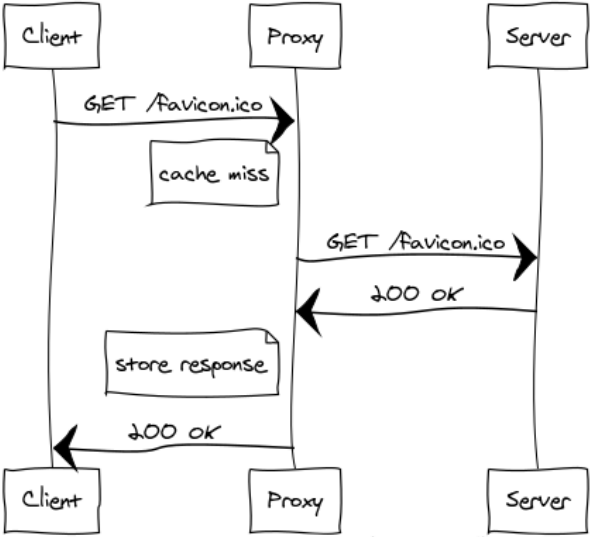
CDN Ingredients

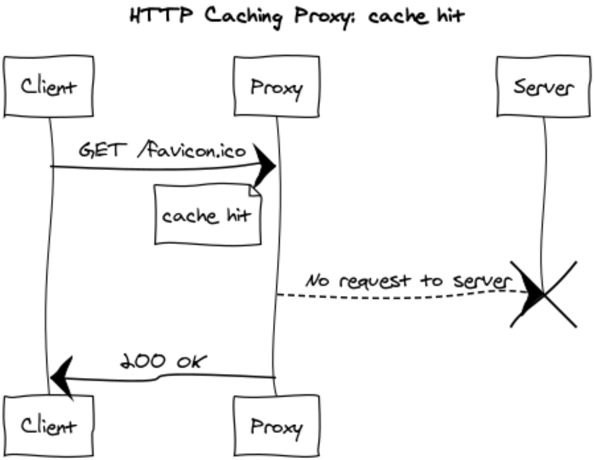
- HTTP Caching
- Load balancing

Caching proxies

Reduce application server load
by caching HTTP responses

HTTP Caching Proxy: cache miss

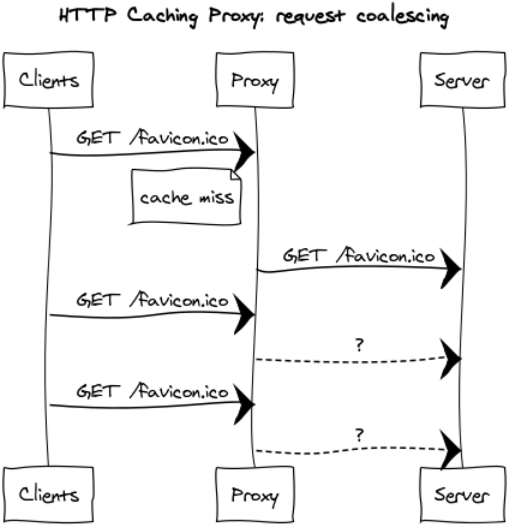



HTTP Caching Proxy: cache hit
Client
Proxy
Server
GET /favicon.ico
cache hit
No request to server
200 OK
Client
Proxy

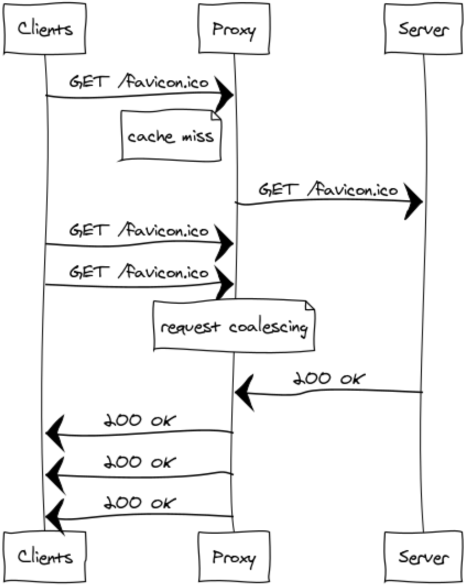
The devil is in the detail

The cache receives multiple requests for the same page before receiving a response from the server.

What should it do?


HTTP Caching Proxy: request coalescing
Clients
Proxy
Server
GET /favicon.ico
cache miss
GET /favicon.ico
GET /favicon.ico
?
GET /favicon.ico
?
Clients
Proxy
Server

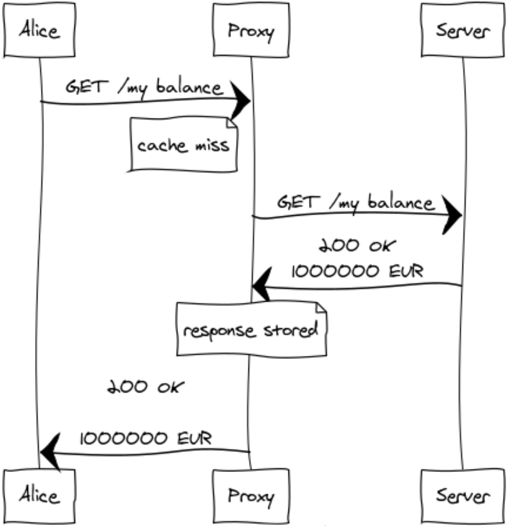
HTTP Caching Proxy: request coalescing



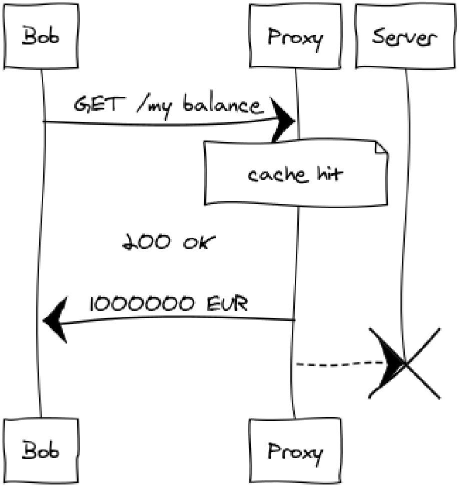
The devil is in the detail

How about your bank account!

HTTP caching no bueno when Bob comes



HTTP caching no bueno when Bob comes

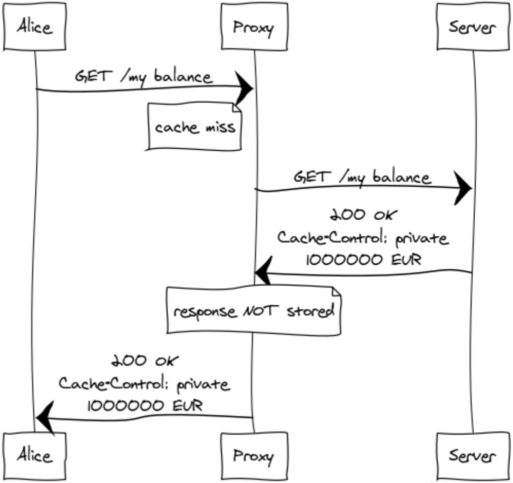


Response headers

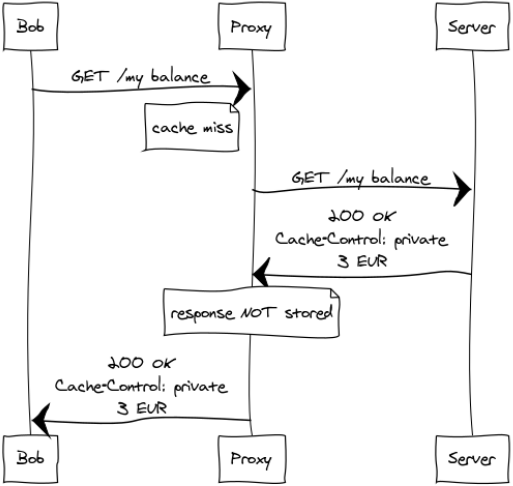
Cache-Control: private

- The response is intended for a single user
- Shared caches must not store it

HTTP caching all bueno when Bob comes



HTTP caching all bueno when Bob comes



Paper: Hypertext Transfer Protocol (HTTP/1.1): Caching

Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed.,

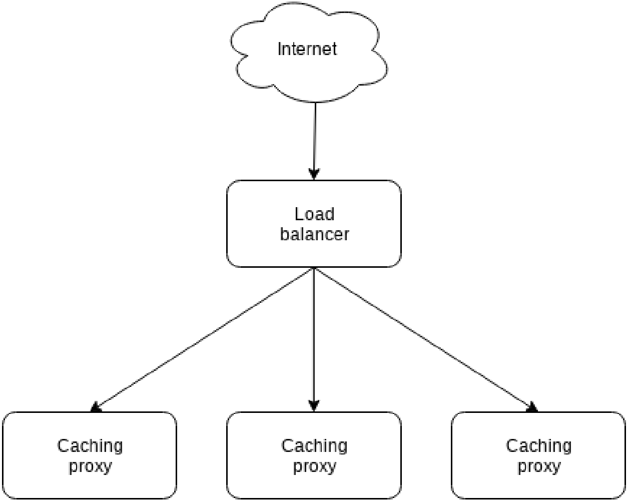
Hypertext Transfer Protocol (HTTP/1.1): Caching

RFC 7234, June 2014.

Load balancing

- One caching proxy is of course not enough
  - Scalability
  - High Availability
- We need to deploy multiple cache servers
- Traffic should be distributed among them somehow evenly

Load balancing



Load balancing

- Load balancers can work at different layers of the networking stack
- L4: backend selection based on layer 3/4 information
- L7: backend selection based on (guess what) layer 7 information

Load balancing: backend selection

L7 HTTP load balancer
We want all requests for the document /foobar to end up on a given cache proxy

Load balancing: backend selection

- Hash the request url!
- In traditional hash tables, mapping is defined by a modular operation
- Changing the number of slots causes nearly all keys to be remapped
- What happens if servers come and go?

Paper: Consistent Hashing

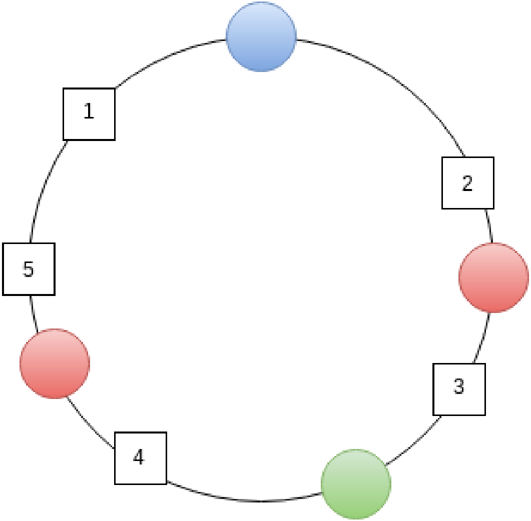
Karger, D., Lehman, E., Leighton, F., Levine, M., Lewin, D., and Panigrahy, R.

Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web.

In Proceedings of the 29th Annual ACM Symposium on Theory of Computing (El Paso, TX, May 1997)

Consistent Hashing

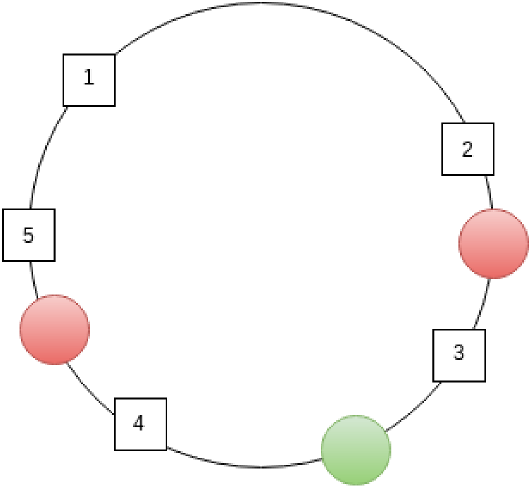
- Map each object to a point on a circle
- Map each bucket to many pseudo-random points on the circle
- To find an object's bucket, find the object on the circle, and walk clockwise till you find the bucket



Blue: 1, 5
Red: 2, 4
Green: 3

Consistent Hashing

- If we remove a bucket, the items that mapped to it must be redistributed among the remaining ones
- Values mapping to other buckets will still do so and do not need to be moved



Red: 2, 4 -> Red: 2, 4, 1, 5

Green: 3 -> Green: 3

A day in the life of an HTTP request

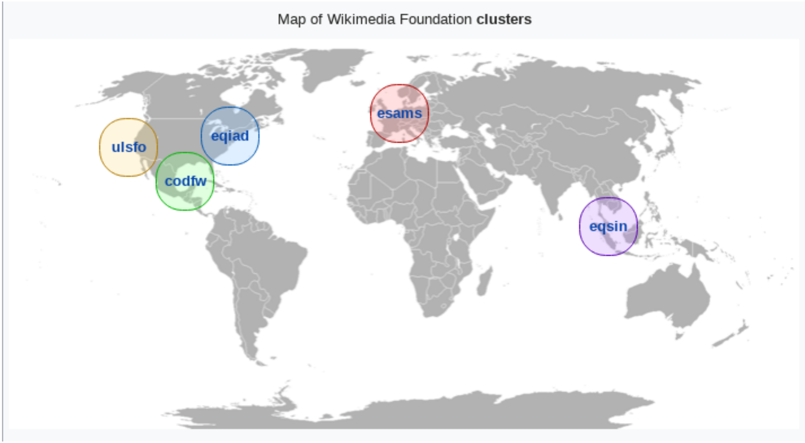
A day in the life of an HTTP request

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

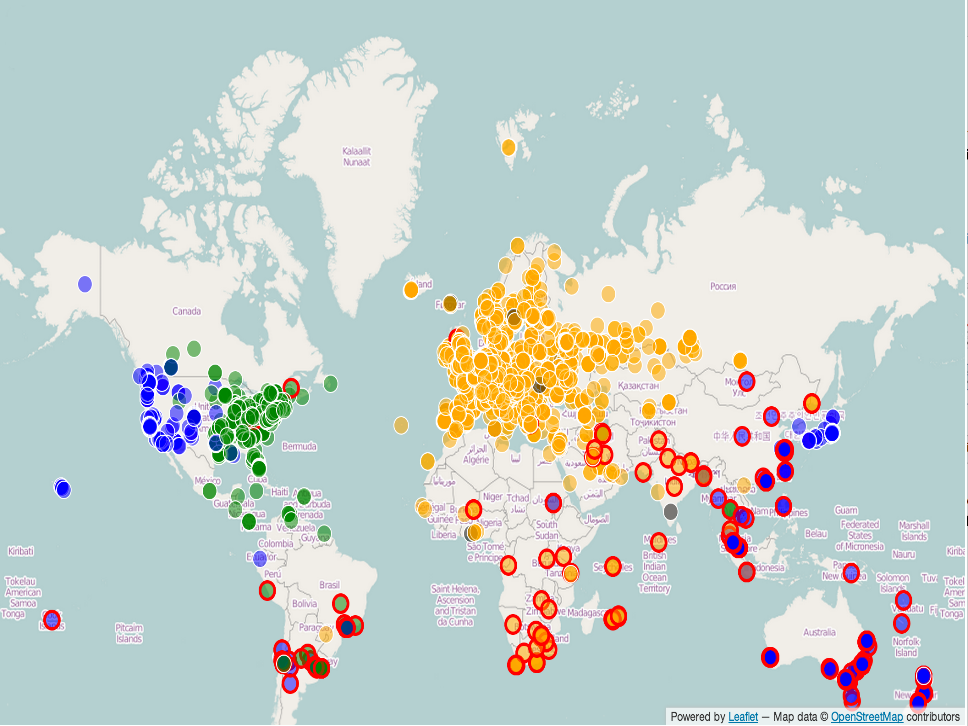
Geographic DNS routing

We get sent to the closest data centre

Cluster Map

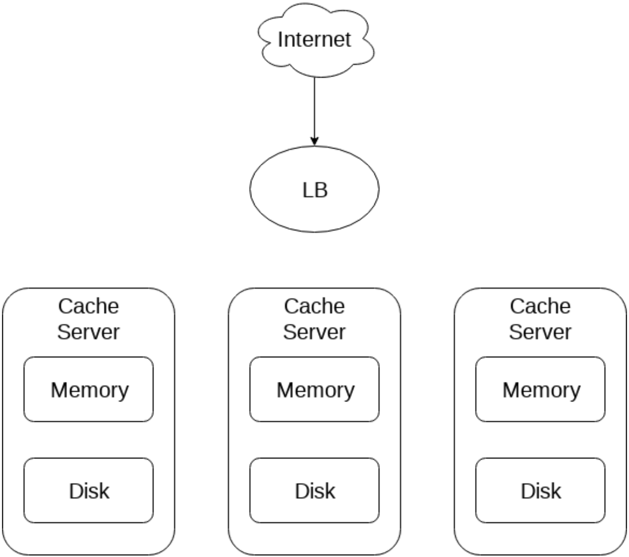


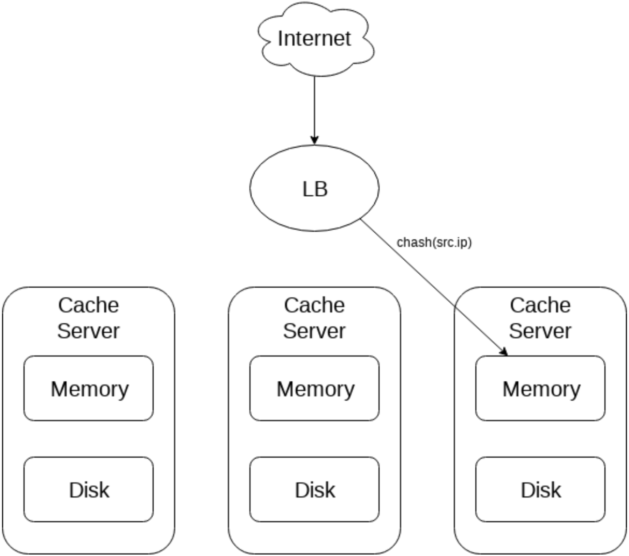
eqiad: Ashburn, Virginia - cp10xx
codfw: Dallas, Texas - cp20xx
esams: Amsterdam, Netherlands - cp30xx
ulsfo: San Francisco, California - cp40xx
eqsin: Singapore - cp50xx


Kalaallit Nunaat
Canada
Россия
Қазақстан
Тоҷикистон
México
Cuba
Haiti
Bermuda
Guatemala
Panama
Venezuela
Guyana
Colombia
Ecuador
Perú
Brasil
Bolivia
Paraguay
Algérie
الجزائر
Niger
Tchad
Nigeria
Liberia
South Sudan
São Tomé e Príncipe
Madagascar
Saint Helena, Ascension and Tristan da Cunha
British Indian Ocean Territory
Maldives
Seychelles
中华人民共和国
Philippines
Indonesia
Guam
Belau
Federated States of Micronesia
Marshall Islands
Nauru
Solomon Islands
Vanuatu
Australia
Norfolk Island
Kiribati
Tokelau
American Samoa
Tonga
Cook Islands
Pitcairn Islands
Powered by Leaflet — Map data © OpenStreetMap contributors

Cache cluster

- Load balancers running Linux Virtual Server
- HTTP cache proxies running Varnish in memory (faster, smaller)
- HTTP cache proxies running Varnish on disk (slower, much larger)


Internet
LB
Cache Server
Memory
Disk
Cache Server
Memory
Disk
Cache Server
Memory
Disk



- L4 load balancing, backend selection based on IP
- Effective cache size: ~avg(mem size)

TCP Connection Establishment

- SYN
- SYN/ACK
- ACK

Paper: TCP Fast Open

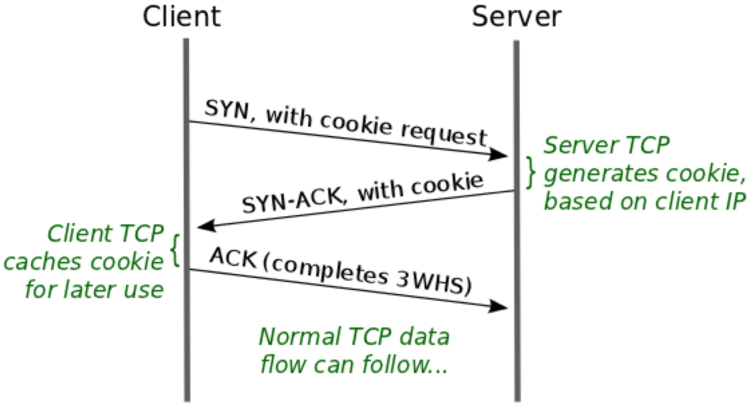
S. Radhakrishnan, Y. Cheng, J. Chu, A. Jain, and B. Raghavan.

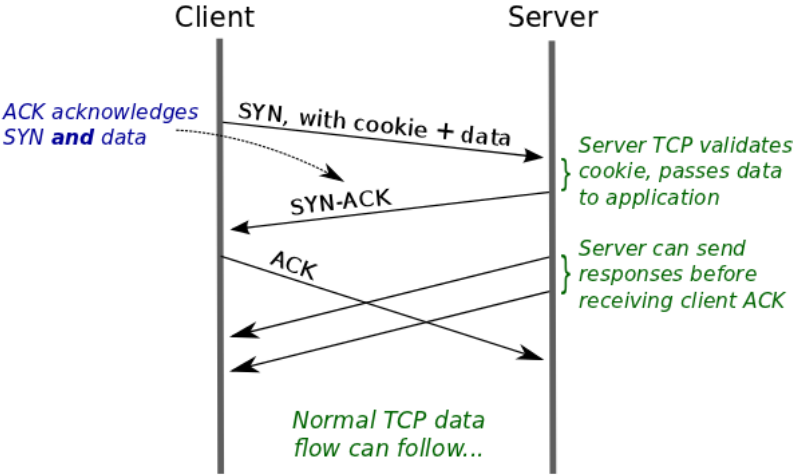
TCP Fast Open.

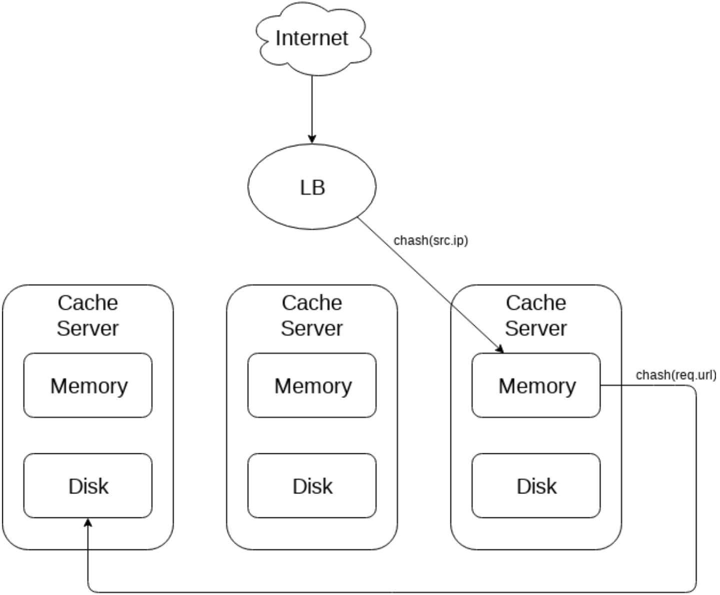
In Proc. of the International Conference on emerging Networking EXperiments and Technologies (CoNEXT), 2011.

TCP Fast Open

- Speed of light cannot be changed
- The number of roundtrips can
- Allow SYN packets to carry data
- Cookie used to authenticate client

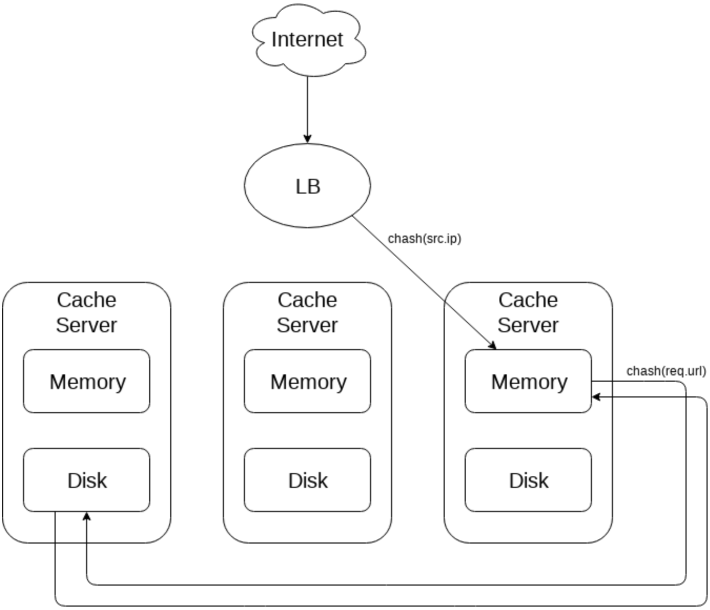

Client
Server
SYN, with cookie request
Server TCP generates cookie, based on client IP
SYN-ACK, with cookie
Client TCP caches cookie for later use
ACK (completes 3WHS)
Normal TCP data flow can follow...


Client
Server
ACK acknowledges
SYN **and** data
SYN, with cookie + data
Server TCP validates
cookie, passes data
to application
SYN-ACK
ACK
Server can send
responses before
receiving client ACK
Normal TCP data
flow can follow...

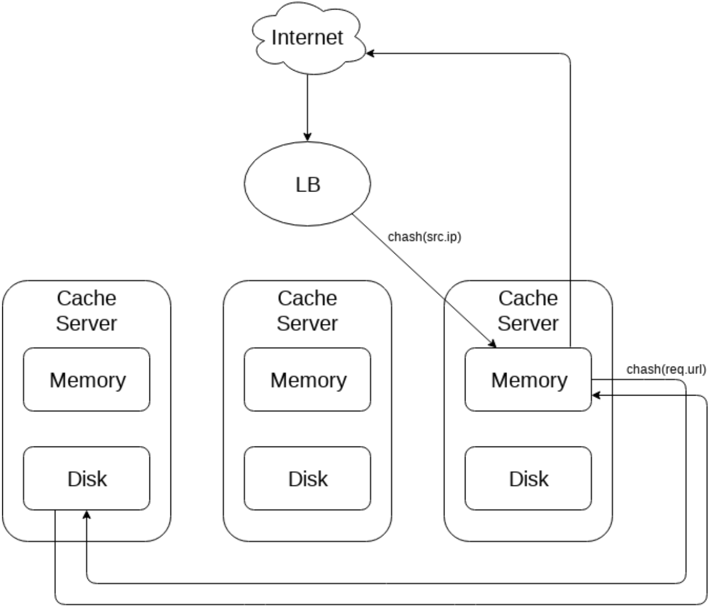


Cache miss

- L7 load balancing, backend selection based on request URL
- Effective cache size: ~sum(disk size)


Internet
LB
chash(src.ip)
Cache Server
Memory
Disk
Cache Server
Memory
Disk
Cache Server
Memory
Disk
chash(req.url)

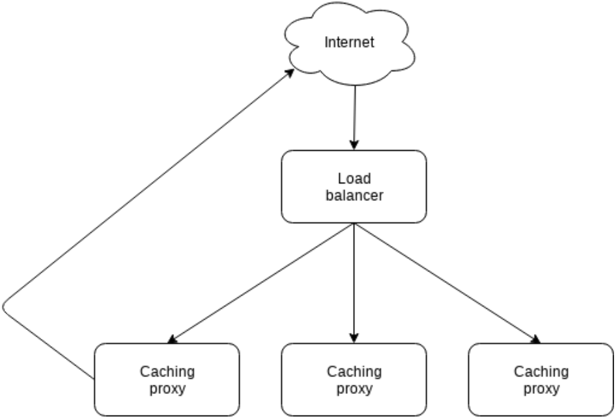
Cache hit


Internet
LB
chash(src.ip)
Cache Server
Memory
Disk
Cache Server
Memory
Disk
Cache Server
Memory
Disk
chash(req.url)

Load balancing: direct routing

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

Load balancing: direct routing



That's a particularly smart idea for HTTP traffic.

Paper: Linux Virtual Server

W. Zhang.

Linux Virtual Server for Scalable Network Services.

In Proceedings of the Linux Symposium, July 2000.

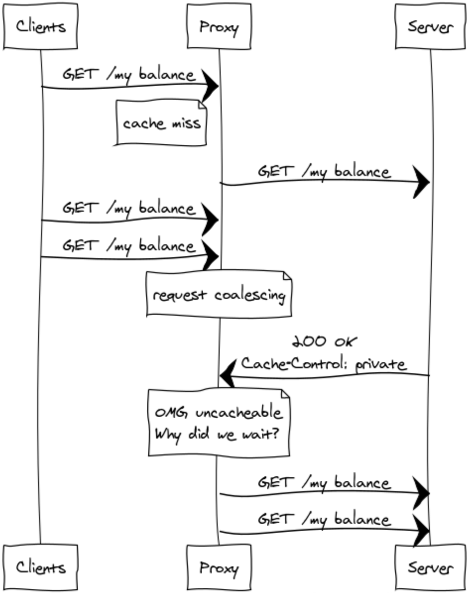
Conclusions: you know more things

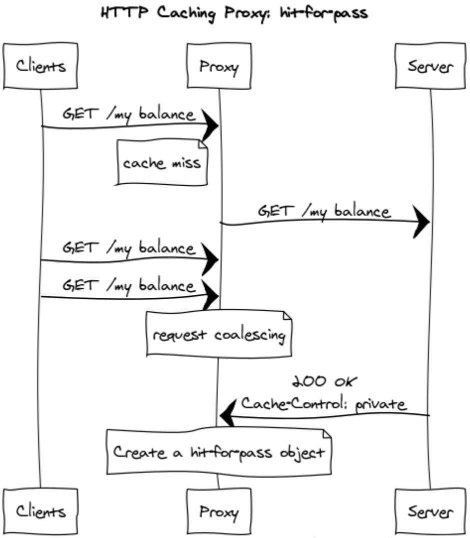
- Wikipedia is one of the largest websites in the world
- It is run by a non-profit called Wikimedia Foundation
- HTTP Caching
- L4/L7 Load Balancing
- Consistent Hashing
- Geographic DNS Routing
- TCP Fast Open
- LVS Direct Routing

The devil is in the detail

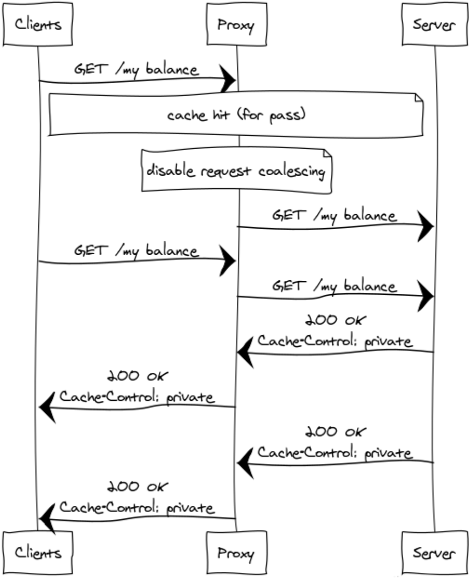
Request coalescing with uncacheable responses

HTTP Caching Proxy: request coalescing can go wrong




HTTP Caching Proxy: hit-for-pass
Clients
Proxy
Server
GET /my balance
cache miss
GET /my balance
GET /my balance
GET /my balance
request coalescing
200 OK
Cache-Control: private
Create a hit-for-pass object
Clients
Proxy
Server

HTTP Caching Proxy: hit-for-pass


Clients
Proxy
Server
GET /my balance
cache hit (for pass)
disable request coalescing
GET /my balance
GET /my balance
GET /my balance
200 OK
Cache-Control: private
200 OK
Cache-Control: private
200 OK
Cache-Control: private
200 OK
Cache-Control: private
Clients
Proxy
Server