Wikimedia Architecture
Doing More With Less

Asher Feldman <asher@wikimedia.org>
Ryan Lane <ryan@wikimedia.org>
Wikimedia Foundation Inc.

Overview

- Intro
- Scale at WMF
- How We Work
- Architecture Dive

Top Five Worldwide Sites (Q411)

Company	Users	Revenue	Employees	Server count
Google	1+ billion	$37.9B	32,000+	1,000,000+
Microsoft®	905 million	$69B	93,000	50,000+
facebook.	714 million	$3.8B	3,000+	60,000+
YAHOO!®	689 million	$6B	1,200	50,000+
WIKIMEDIA	490 million	$30m	75	700

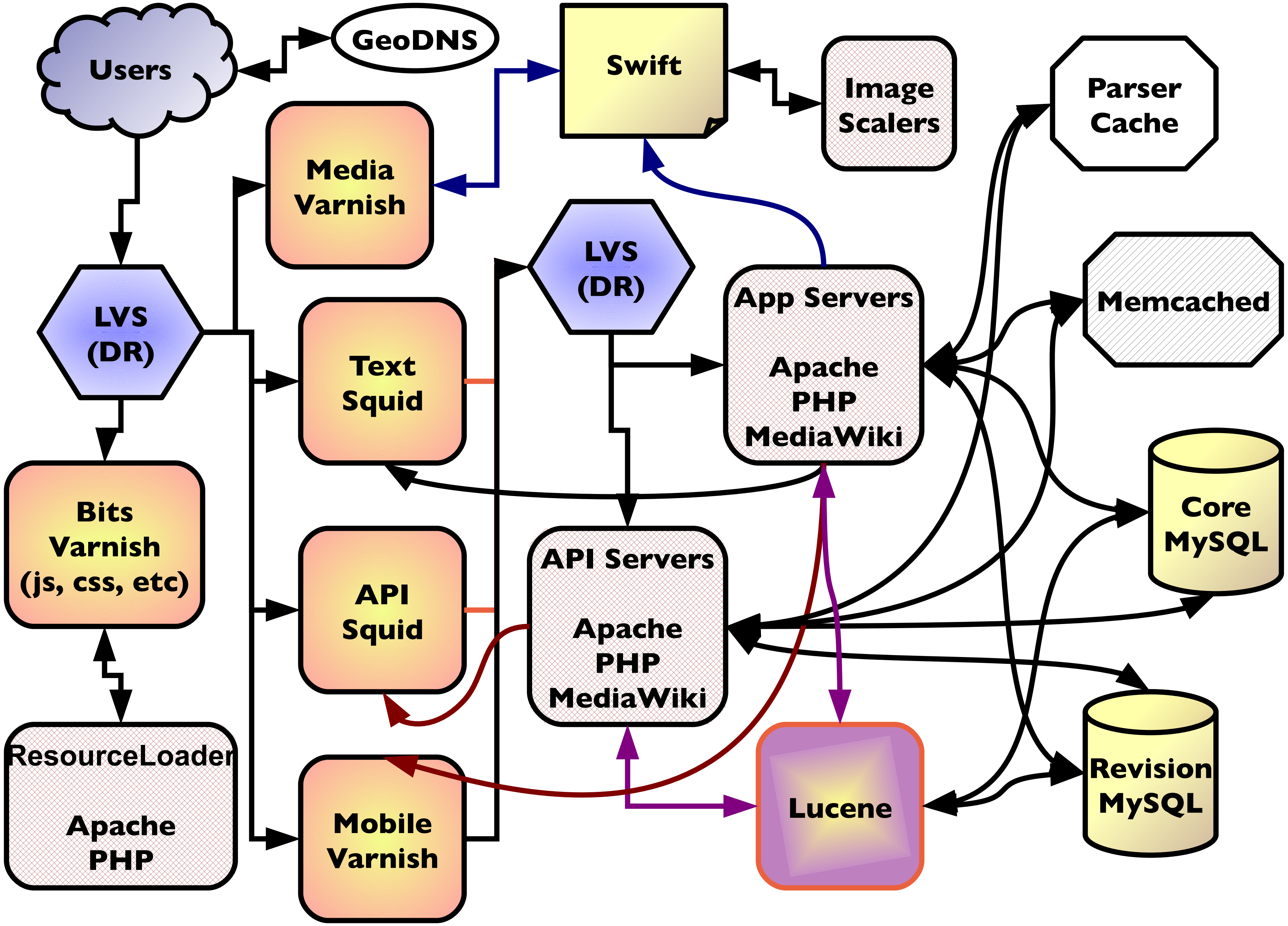
What We're Working With

- > 490 Million unique visitors per month
- 1.6 Billion Article Views per day
- 10 Billion HTTP Requests Served per day
- Mostly served by around 130 Apache/PHP/memcached, 50 MySQL, 100 Squid/Varnish, 24 Lucene, and 20 swift + image processing servers

Open Source, Open Culture

- Around 25 Full Time Software and Operations Engineers, but a global community of Open Source developers helping to develop and extend MediaWiki
- Even Operations follows the model – our puppet cluster configuration code is public, as are our monitoring services. Volunteers can model changes in our virtualized Labs cluster
- Everything we use is Open Source. We often modify apps (such as squid) to meet our needs and scale, but we always release the changes

Basic Architecture



Asher Feldman, asher@wikimedia.org, Wikimedia Foundation Inc.
Ryan Lane, ryan@wikimedia.org, Wikimedia Foundation Inc.

The Wiki software

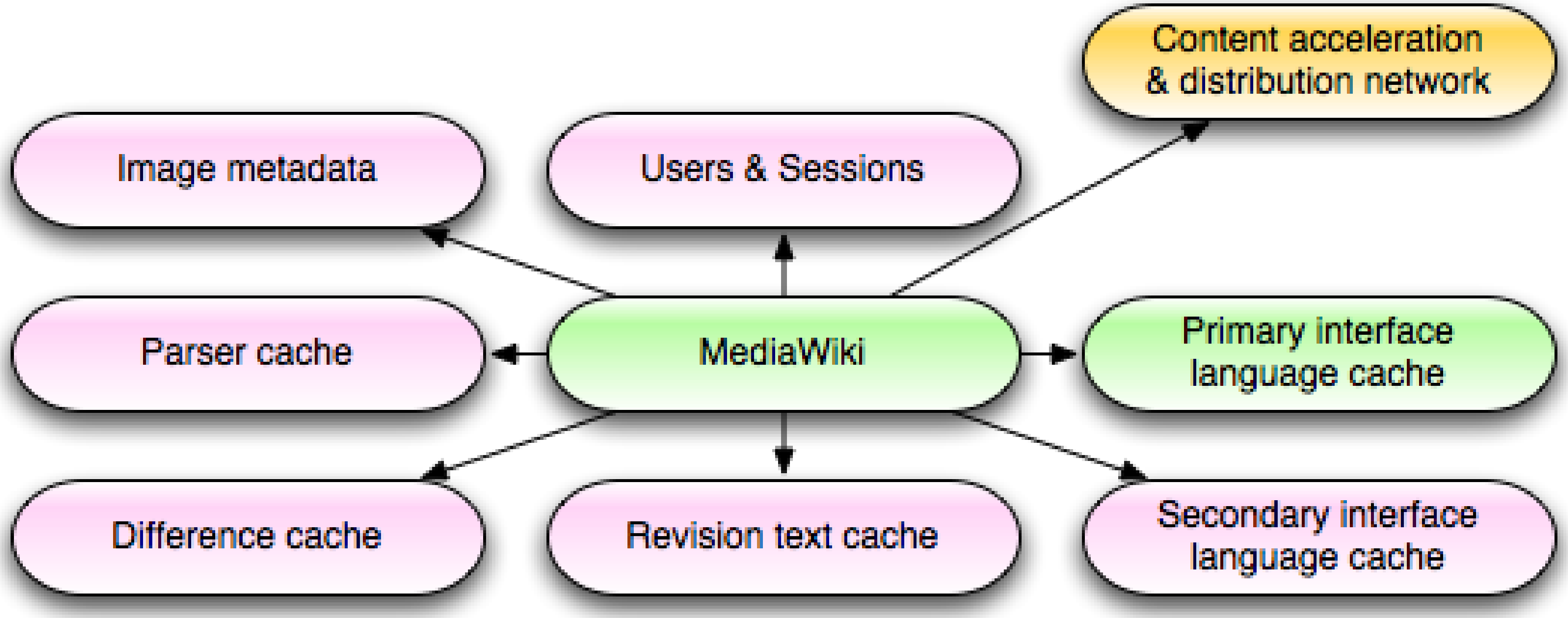
- All Wikimedia projects run on a MediaWiki platform
- Designed primarily for Wikimedia sites
- Very scalable (mostly), very good localization
- Open Source PHP software (GPL)

MediaWiki optimization

- We try to optimize by...
  - not doing anything stupid (this is hard!)
  - caching expensive operations
  - focusing on the hot spots in the code (profiling!)
- If a MediaWiki feature is too expensive, it doesn't get enabled on Wikipedia

MediaWiki caching

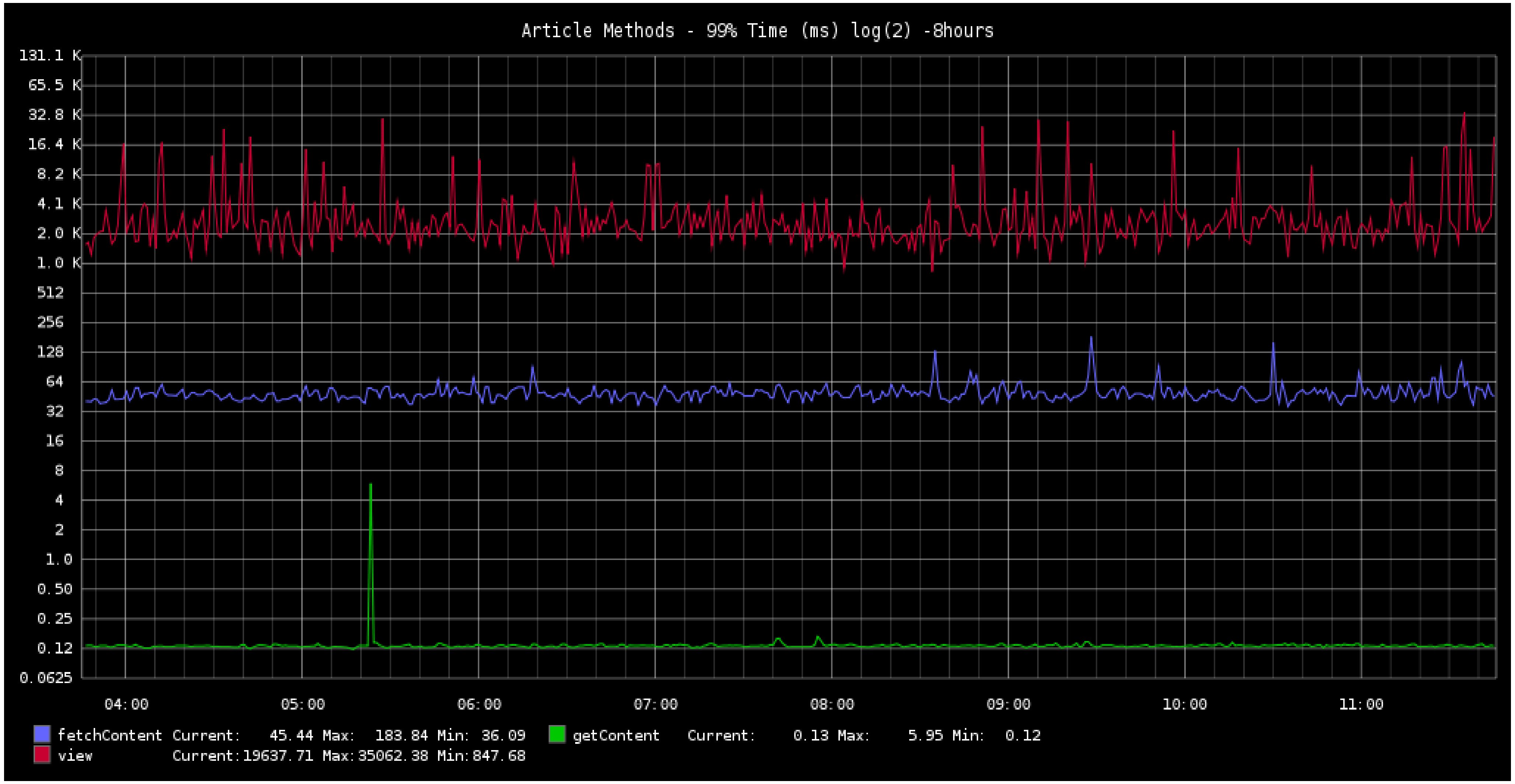
- Objects + queries cached in memcached
- Article html (parsed wikitext) stored in Parser Cache – combined memcached + MySQL blob storage, to scale far beyond available ram


Content acceleration & distribution network
Image metadata
Users & Sessions
Parser cache
MediaWiki
Primary interface language cache
Difference cache
Revision text cache
Secondary interface language cache

MediaWiki profiling

http://gdash.wikimedia.org/

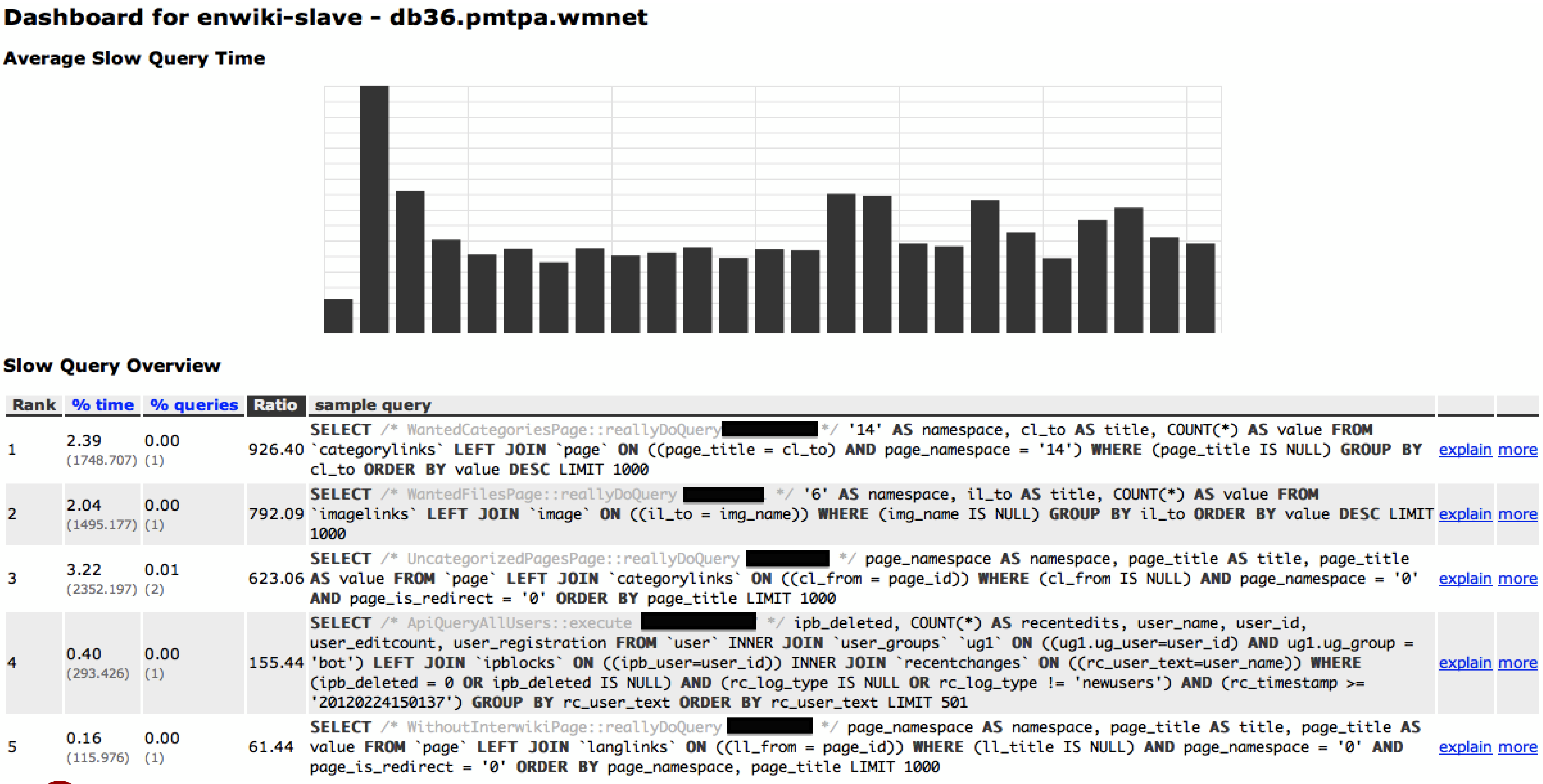
(and others)



Core databases

- Mysql 5.1 with the Facebook patch set
- One master per shard, many replicated slaves
- Reads are load balanced amongst unlagged slaves, writes to master
- Separate big, popular wikis from smaller wikis (shard by wiki, but we need to do better to keep scaling)
- Profile with mk-query-digest and the percona-toolkit

Database profiling



Dashboard for enwiki-slave - db36.pmtpa.wmnet

Average Slow Query Time

Slow Query Overview

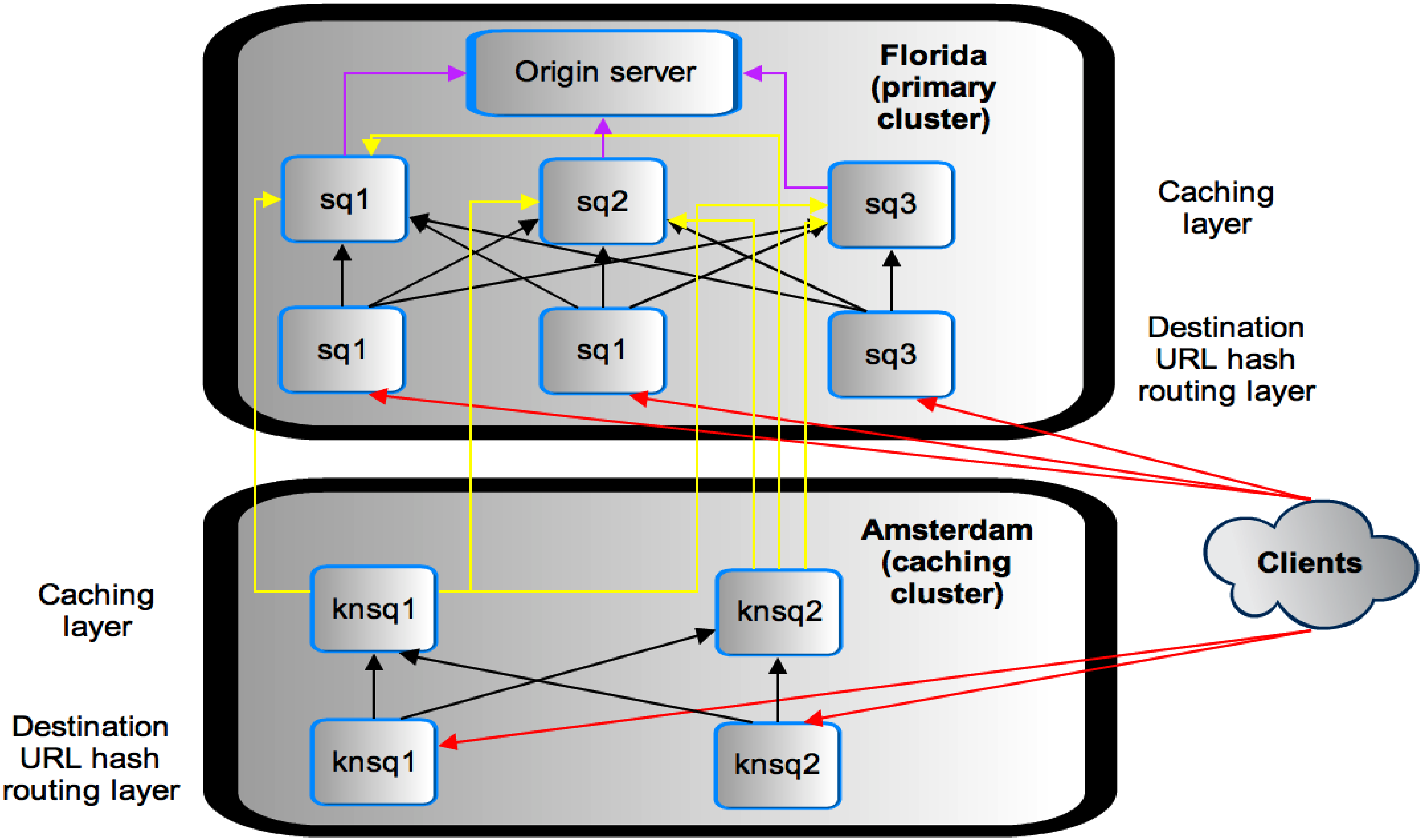
Rank	% time	% queries	Ratio	sample query		
1	2.39 (1748.707)	0.00 (1)	926.40	SELECT /* WantedCategoriesPage::reallyDoQuery [redacted] */ '14' AS namespace, cl_to AS title, COUNT(*) AS value FROM `categorylinks` LEFT JOIN `page` ON ((page_title = cl_to) AND page_namespace = '14') WHERE (page_title IS NULL) GROUP BY cl_to ORDER BY value DESC LIMIT 1000	explain	more
2	2.04 (1495.177)	0.00 (1)	792.09	SELECT /* WantedFilesPage::reallyDoQuery [redacted] */ '6' AS namespace, il_to AS title, COUNT(*) AS value FROM `imagelinks` LEFT JOIN `image` ON ((il_to = img_name)) WHERE (img_name IS NULL) GROUP BY il_to ORDER BY value DESC LIMIT 1000	explain	more
3	3.22 (2352.197)	0.01 (2)	623.06	SELECT /* UncategorizedPagesPage::reallyDoQuery [redacted] */ page_namespace AS namespace, page_title AS title, page_title AS value FROM `page` LEFT JOIN `categorylinks` ON ((cl_from = page_id)) WHERE (cl_from IS NULL) AND page_namespace = '0' AND page_is_redirect = '0' ORDER BY page_title LIMIT 1000	explain	more
4	0.40 (293.426)	0.00 (1)	155.44	SELECT /* ApiQueryAllUsers::execute [redacted] */ ipb_deleted, COUNT(*) AS recentedits, user_name, user_id, user_editcount, user_registration FROM `user` INNER JOIN `user_groups` `ug1` ON ((ug1.ug_user=user_id) AND ug1.ug_group = 'bot') LEFT JOIN `ipblocks` ON ((ipb_user=user_id)) INNER JOIN `recentchanges` ON ((rc_user_text=user_name)) WHERE (ipb_deleted = 0 OR ipb_deleted IS NULL) AND (rc_log_type IS NULL OR rc_log_type != 'newusers') AND (rc_timestamp >= '20120224150137') GROUP BY rc_user_text ORDER BY rc_user_text LIMIT 501	explain	more
5	0.16 (115.976)	0.00 (1)	61.44	SELECT /* WithoutInterwikiPage::reallyDoQuery [redacted] */ page_namespace AS namespace, page_title AS title, page_title AS value FROM `page` LEFT JOIN `langlinks` ON ((ll_from = page_id)) WHERE (ll_title IS NULL) AND page_namespace = '0' AND page_is_redirect = '0' ORDER BY page_namespace, page_title LIMIT 1000	explain	more

Asher Feldman, asher@wikimedia.org, Wikimedia Foundation Inc.
Ryan Lane, ryan@wikimedia.org, Wikimedia Foundation Inc.

Squid caching

- Caching reverse HTTP proxy
- Serves our article and api traffic, other clusters have moved to Varnish
- Uses CARP to consistently route requests to backend caches
- Hit rate is > 85%

CARP


Origin server
Florida
(primary
cluster)
sq1
sq2
sq3
sq1
sq1
sq3
Caching
layer
Destination
URL hash
routing layer
Amsterdam
(caching
cluster)
knsq1
knsq2
knsq1
knsq2
Clients
Caching
layer
Destination
URL hash
routing layer

Varnish Caching

- Used for delivering static content such as js and css files (bits), media (uploads), and our mobile site
- 2-3 times more efficient than Squid
- We've served >40k requests/second from a single server
- URI hashing between tiered caches replaces CARP (we are adding consistent hashing to Varnish)
- Will eventually replace squid entirely in our infrastructure, but some development is still needed

Cache invalidation

- Wiki pages are edited at an unpredictable rate
- Users should always see current revision
- Invalidation through expiry times not acceptable
- Purge implemented using multicast UDP based HTCP protocol

Media storage

- Images inline in articles (thumbnails) are stored in Swift, recently replacing an unscalable NFS based solution
- Part of OpenStack, Swift is like an Open Source implementation of S3 (even supports the S3 API)
- Soon originals and video will be stored in and served from swift as well

Content Distribution Network (CDN)

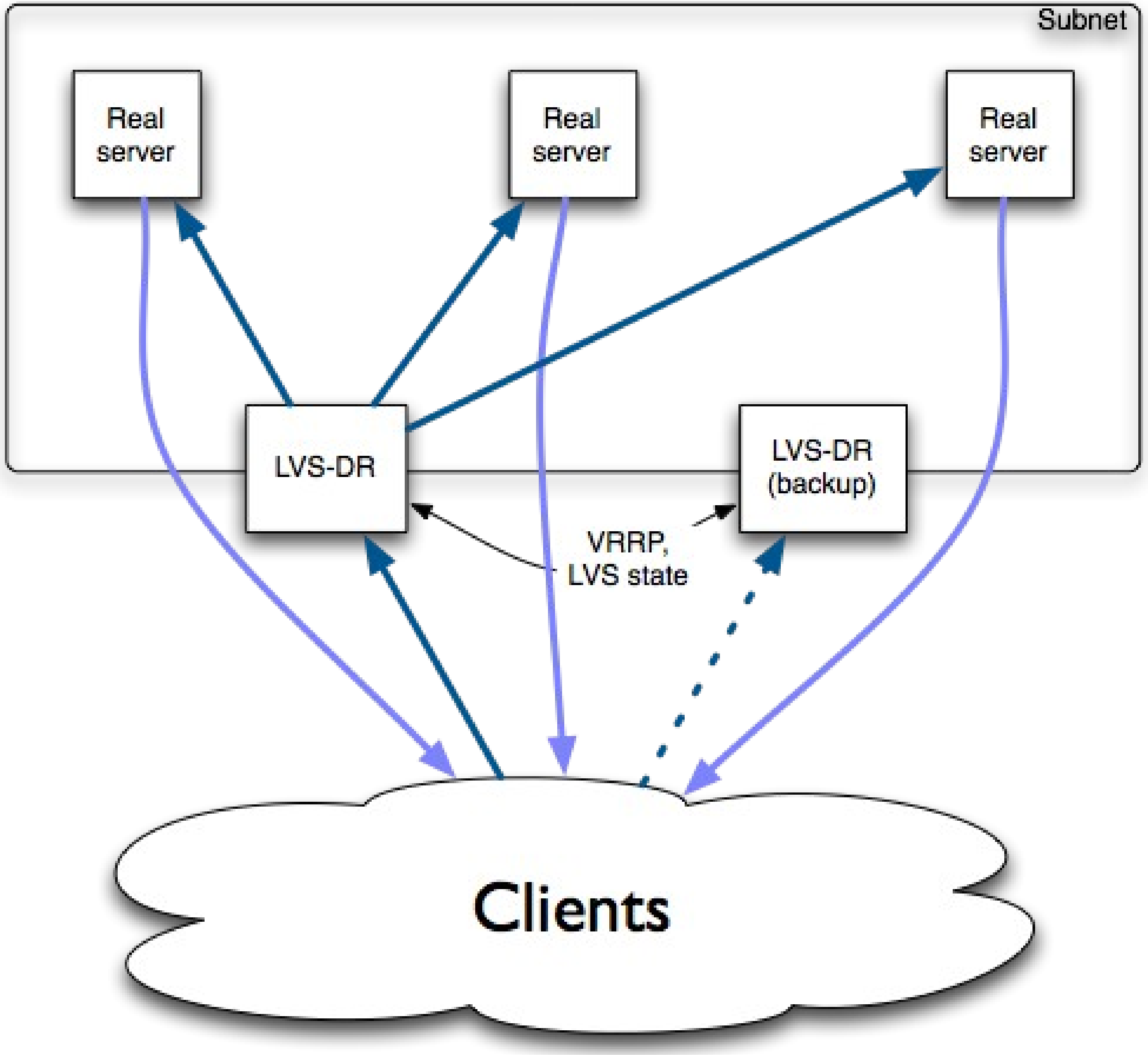
- 3 clusters on 2 different continents:
  - Redundant application and cache clusters in Tampa, Florida and Ashburn, Virginia. Currently Tampa is our primary application cluster but cached traffic is served from Ashburn
  - Europe served by a caching-only cluster in Amsterdam (uncached content is proxied to Tampa)
  - Soon building a new caching cluster in California which will lower latency to Asia

Geographic Load Balancing

- Most users use DNS resolver close to them
- Map IP address of resolver to a country code
- Deliver CNAME of close datacenter entry based on country
- Using PowerDNS with a Geobackend

Load Balancing: LVS-DR

- Linux Virtual Server
- Direct Routing mode
- All real servers share the same IP address
- The load balancer divides incoming traffic over the real servers
- Return traffic goes directly!



Closing notes

- We rely heavily on open source
- Still some big scaling issues to tackle around parsing and editing
- Always looking for efficiencies
- Looking for more efficient management tools
- Looking for more contributors

Questions, comments?

- E-mail: Asher Feldman <afeldman@wikimedia.org>