WIKIPEDIA
The Free Encyclopedia

WIKINEWS

WIKIBOOKS

Wikimedia architecture

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

WIKIVERSITY

Topics

- Intro
- Our technical operations
- Global architecture
- Application servers
- Storage
- Caching
- Load balancing
- Content Delivery Network (CDN)
- The site architecture you can edit
- Community involvement

Top five worldwide sites

Company	Users	Revenue	Employees	Server count
Google	920 million	$23 billion	20,600	1,000,000+
Microsoft	740 million	$58 billion	93,000	50,000+
YAHOO!	600 million	$6 billion	13,900	50,000+
facebook	500 million	$300 million	1,200	30,000+
WIKIMEDIA	400 million	$20 million	50	350

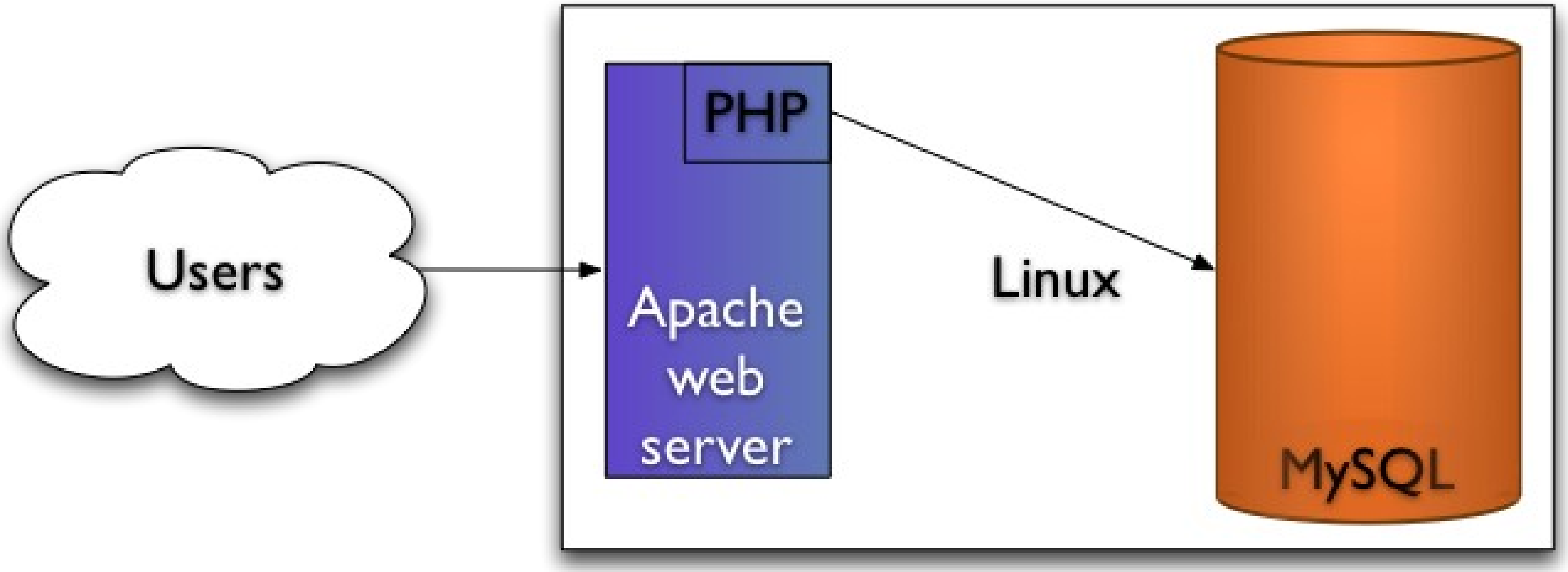
Our operations

- Currently managed by ~6 ops engineers
- Historically ad-hoc, “fire fighting mode”
- Technical staff spread out globally
- Always someone awake...but no on-call
- Working to engage community operations contributors

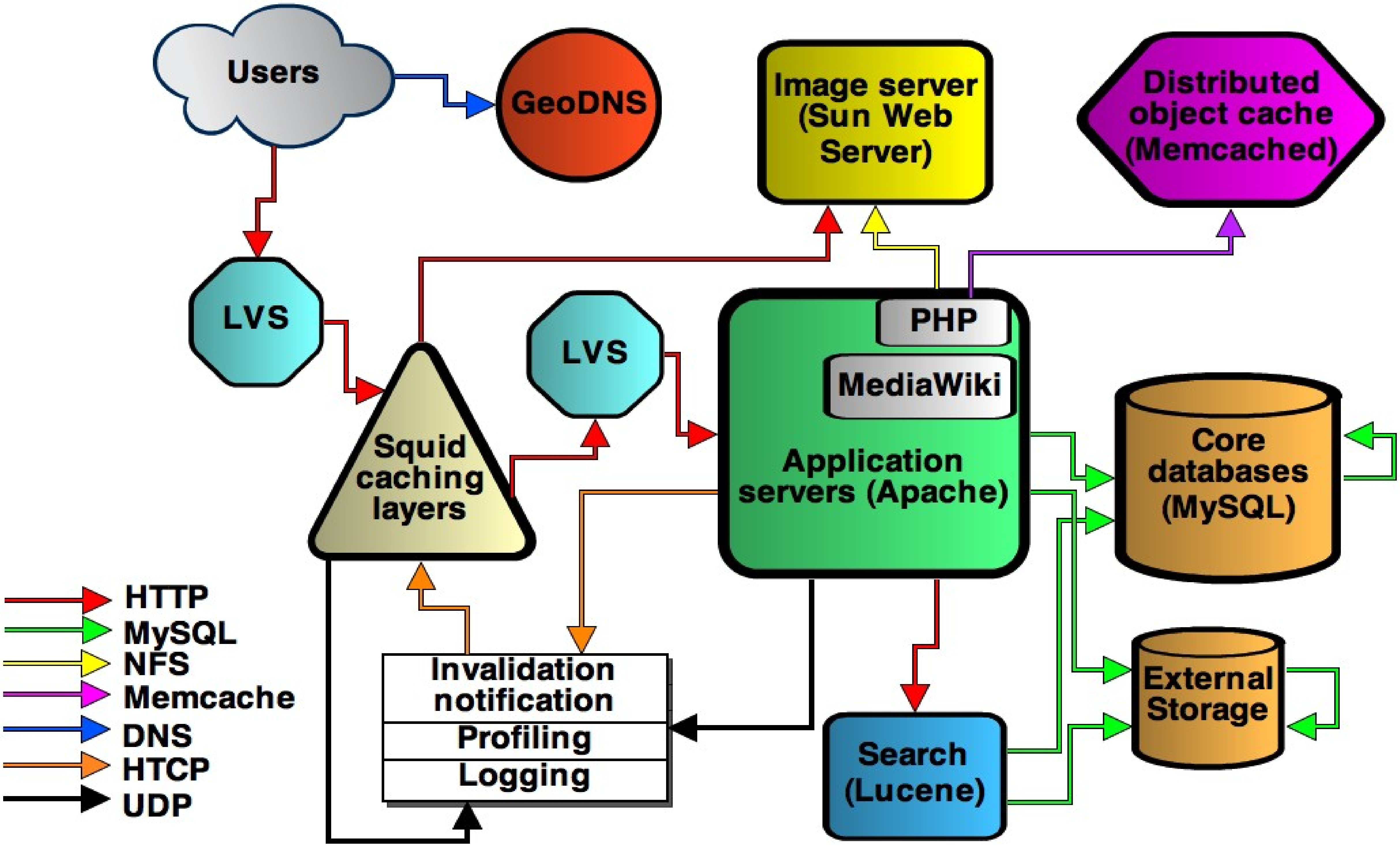
Operations communication

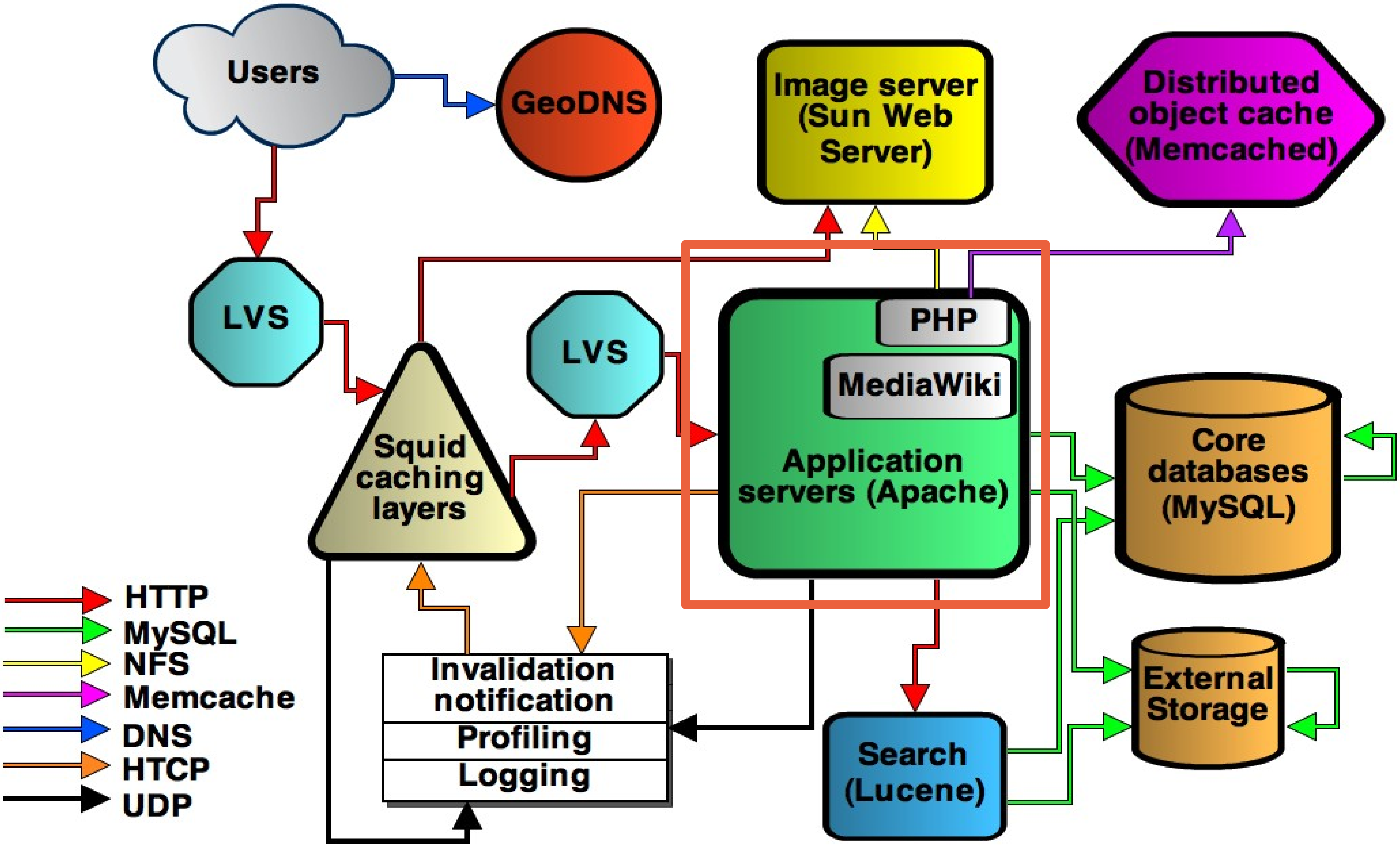
- Most communication public on IRC
- Documentation in a wiki (http://wikitech.wikimedia.org)
- Sensitive communication via private lists and resource trackers

Architecture: LAMP...



...on steroids.


Users
GeoDNS
Image server (Sun Web Server)
Distributed object cache (Memcached)
LVS
LVS
PHP
MediaWiki
Application servers (Apache)
Squid caching layers
Core databases (MySQL)
Invalidation notification
Profiling
Logging
External Storage
Search (Lucene)
HTTP
MySQL
NFS
Memcache
DNS
HTCP
UDP


Users
GeoDNS
Image server (Sun Web Server)
Distributed object cache (Memcached)
LVS
LVS
PHP
MediaWiki
Application servers (Apache)
Squid caching layers
Core databases (MySQL)
External Storage
Search (Lucene)
Invalidation notification
Profiling
Logging
HTTP
MySQL
NFS
Memcache
DNS
HTCP
UDP

The Wiki software

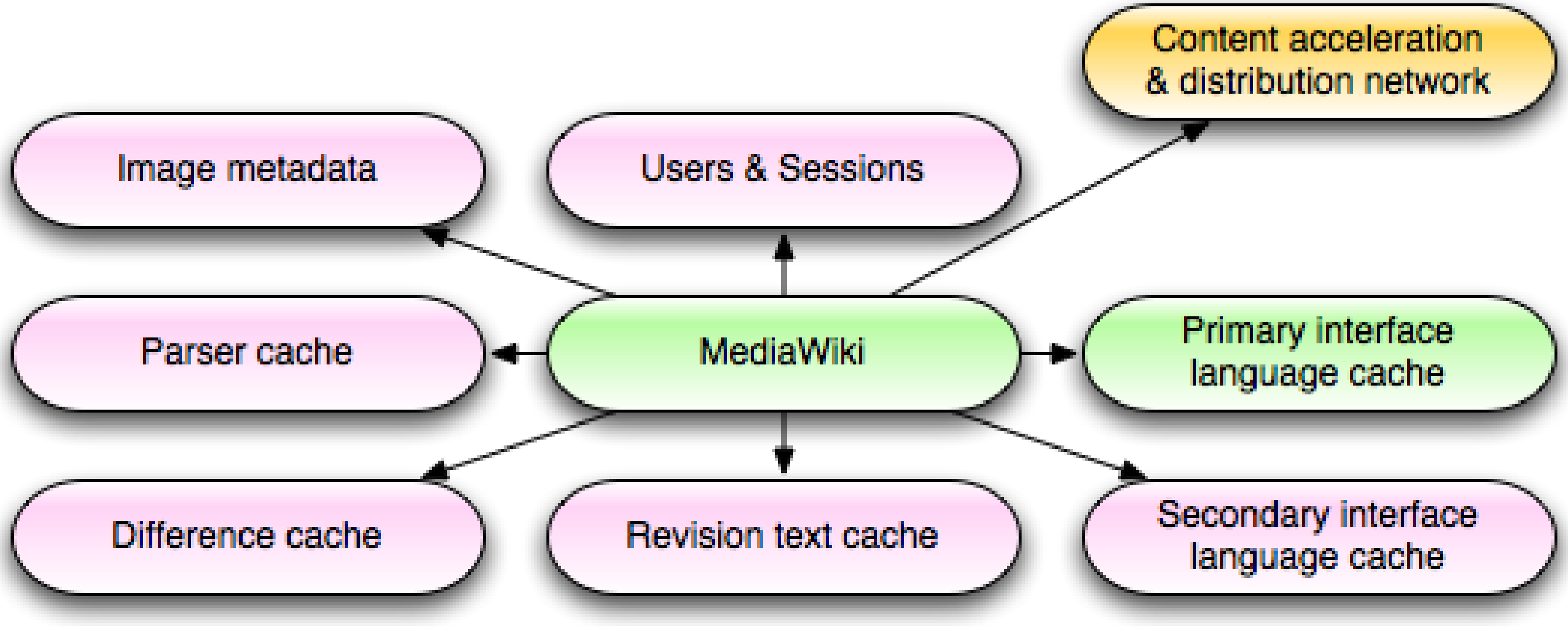
- All Wikimedia projects run on a MediaWiki platform
- Designed primarily for Wikimedia sites
- Open Source PHP software (GPL)
- Very scalable, very good localization

MediaWiki optimization

- We optimize by...
  - caching expensive operations
  - focusing on the hot spots in the code (profiling!)
- If a MediaWiki feature is too expensive, it doesn't get enabled

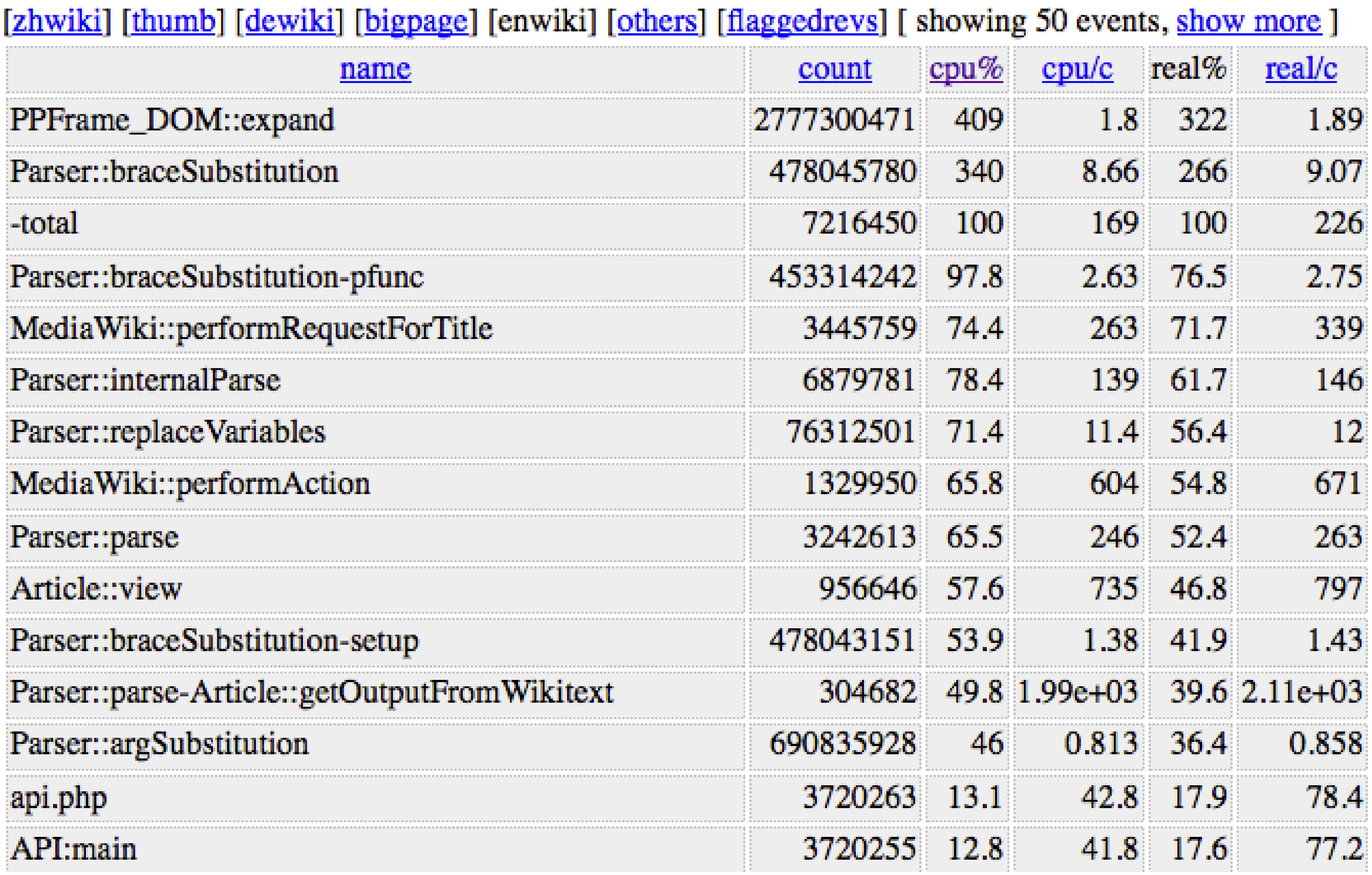
MediaWiki caching

- Caches everywhere
- Most of this data is cached in Memcached, a distributed object cache


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

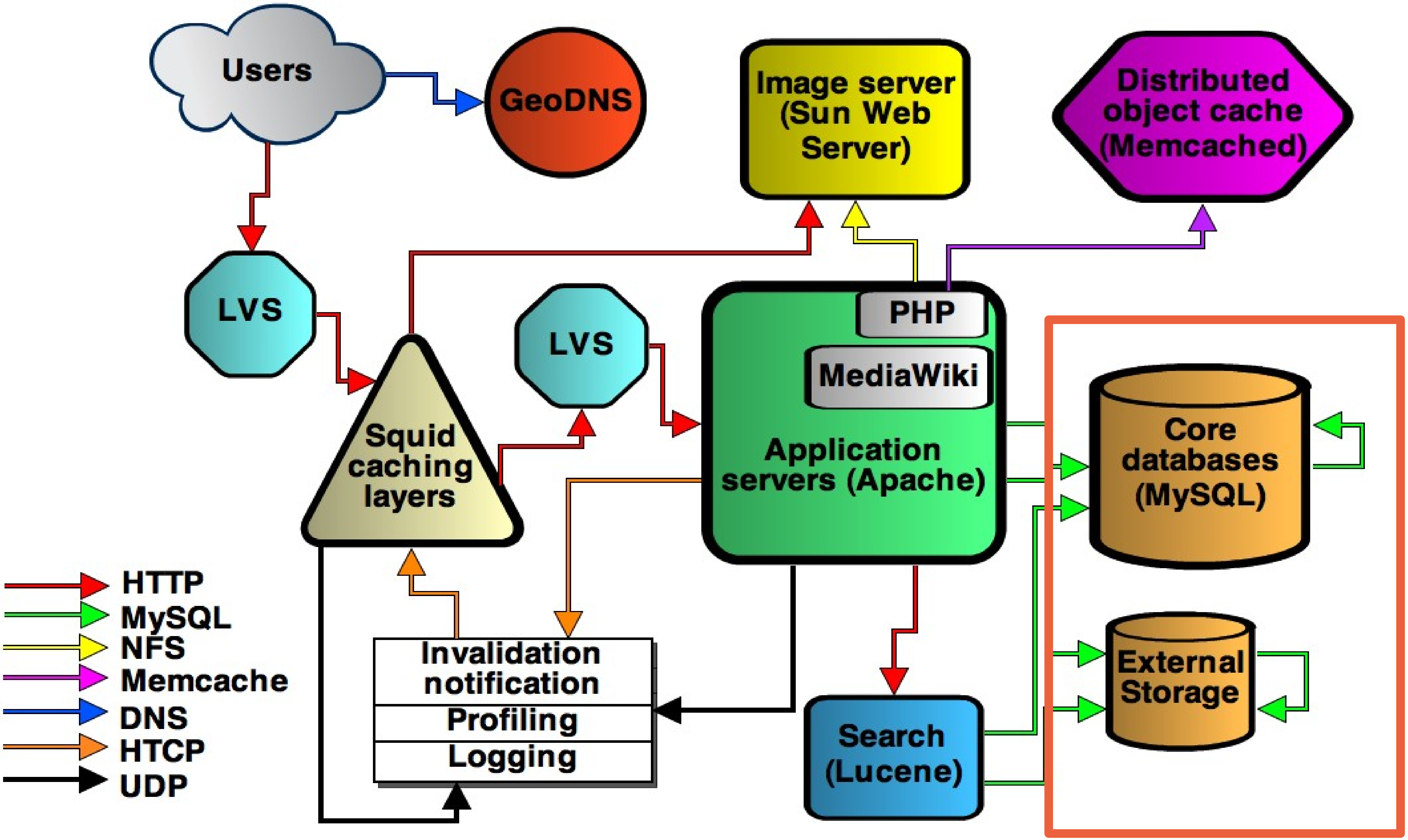
MediaWiki profiling

http://noc.wikimedia.org/cgi-bin/report.py



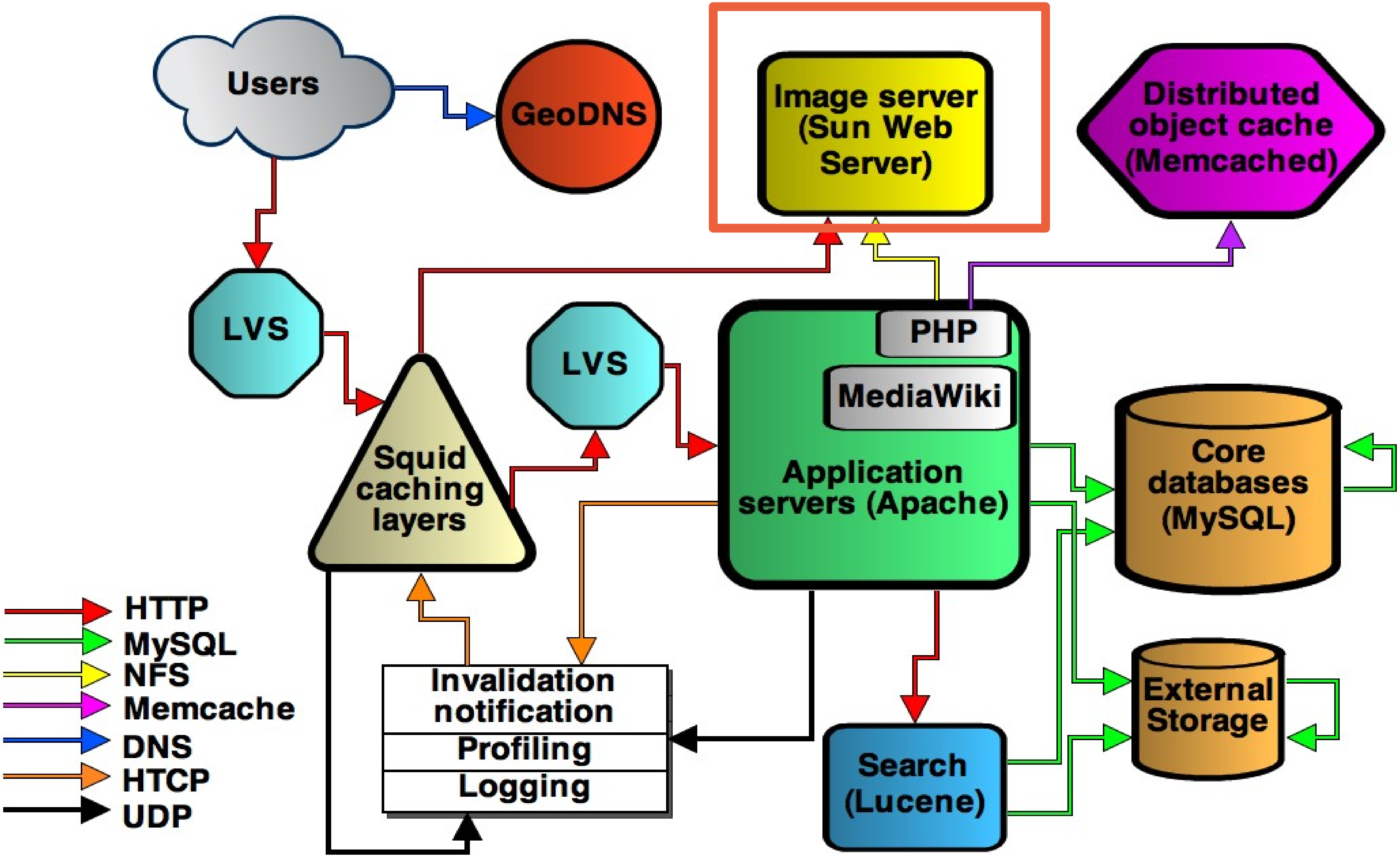
[zhwiki] [thumb] [dewiki] [bigpage] [enwiki] [others] [flaggedrevs] [showing 50 events, show more]

name	count	cpu%	cpu/c	real%	real/c
PPFrame_DOM::expand	2777300471	409	1.8	322	1.89
Parser::braceSubstitution	478045780	340	8.66	266	9.07
-total	7216450	100	169	100	226
Parser::braceSubstitution-pfunc	453314242	97.8	2.63	76.5	2.75
MediaWiki::performRequestForTitle	3445759	74.4	263	71.7	339
Parser::internalParse	6879781	78.4	139	61.7	146
Parser::replaceVariables	76312501	71.4	11.4	56.4	12
MediaWiki::performAction	1329950	65.8	604	54.8	671
Parser::parse	3242613	65.5	246	52.4	263
Article::view	956646	57.6	735	46.8	797
Parser::braceSubstitution-setup	478043151	53.9	1.38	41.9	1.43
Parser::parse-Article::getOutputFromWikitext	304682	49.8	1.99e+03	39.6	2.11e+03
Parser::argSubstitution	690835928	46	0.813	36.4	0.858
api.php	3720263	13.1	42.8	17.9	78.4
API:main	3720255	12.8	41.8	17.6	77.2


Users
GeoDNS
Image server (Sun Web Server)
Distributed object cache (Memcached)
LVS
Squid caching layers
LVS
PHP
MediaWiki
Application servers (Apache)
Core databases (MySQL)
External Storage
Invalidation notification
Profiling
Logging
Search (Lucene)
HTTP
MySQL
NFS
Memcache
DNS
HTCP
UDP

Core databases

- One master, many replicated slaves
- Load balanced reads to slaves, write operations to master
- Separate database per wiki
- Separate big, popular wikis from smaller wikis (sharding)

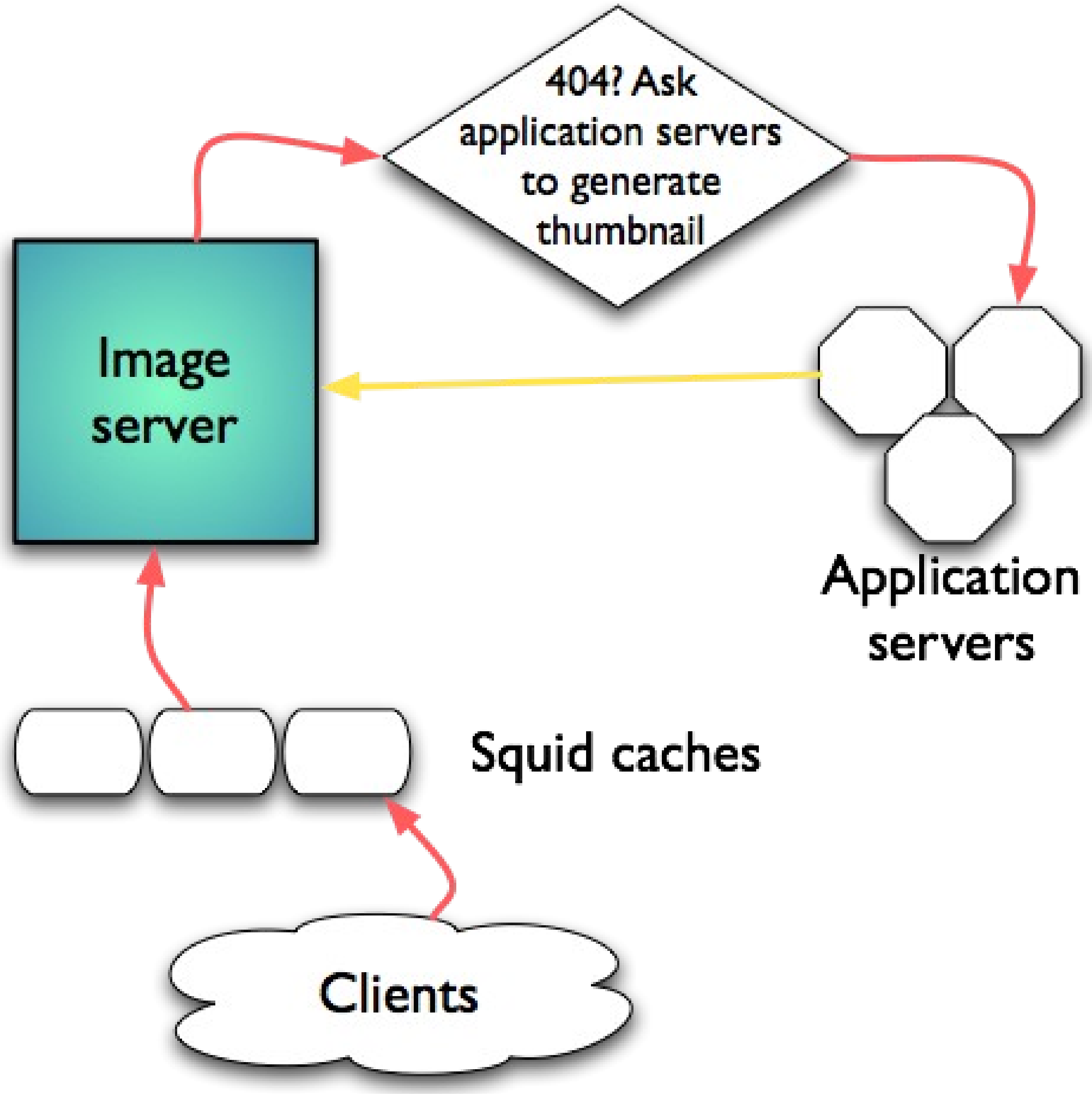

Users
GeoDNS
Image server (Sun Web Server)
Distributed object cache (Memcached)
LVS
LVS
PHP
MediaWiki
Application servers (Apache)
Core databases (MySQL)
Squid caching layers
Invalidation notification
Profiling
Logging
Search (Lucene)
External Storage
HTTP
MySQL
NFS
Memcache
DNS
HTCP
UDP

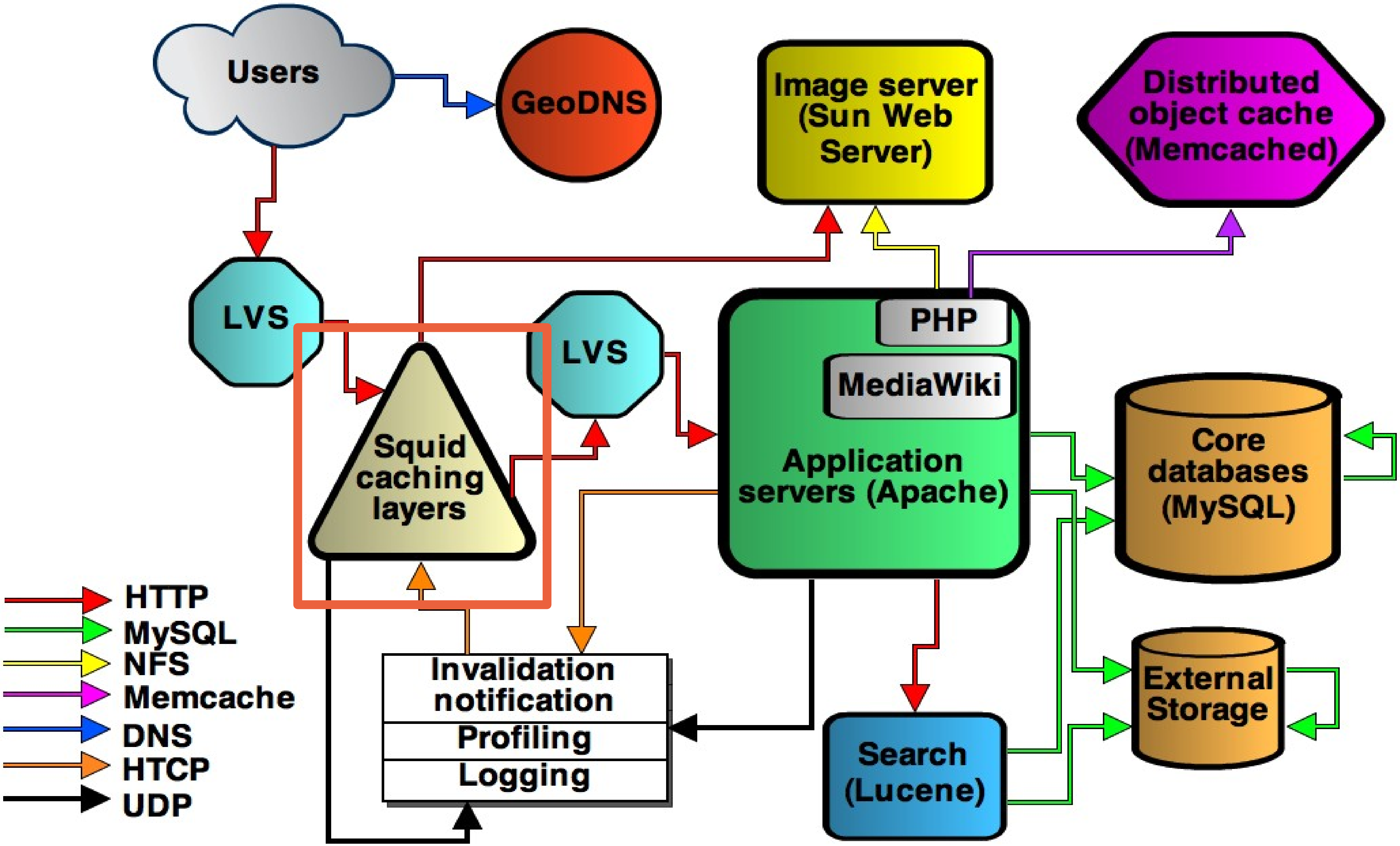
Media storage

- Current solution is not scalable
- Replacing with an open source distributed file system
  - Likely OpenStack Swift

Thumbnail generation

- `stat()` on each request is too expensive, so assume every file exists
- If a thumbnail doesn't exist, ask the application servers to render it

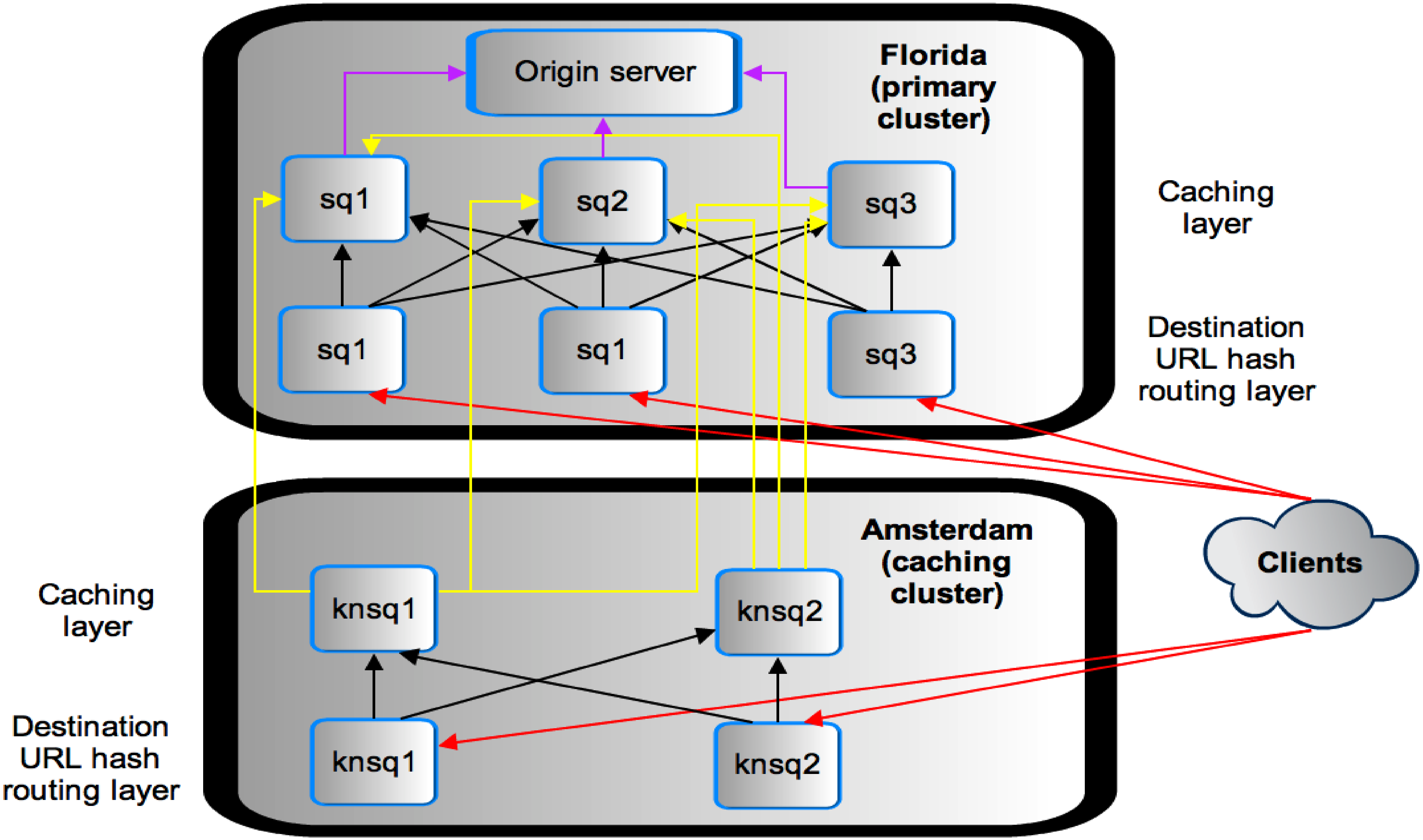



Users
GeoDNS
Image server (Sun Web Server)
Distributed object cache (Memcached)
LVS
Squid caching layers
LVS
PHP
MediaWiki
Application servers (Apache)
Core databases (MySQL)
Invalidation notification
Profiling
Logging
Search (Lucene)
External Storage
HTTP
MySQL
NFS
Memcache
DNS
HTCP
UDP

Squid caching

- Caching reverse HTTP proxy
- Serves most of our traffic
- Split into two groups
- Hit rates: 95% for Text, 98% for Media, since the use of CARP

CARP

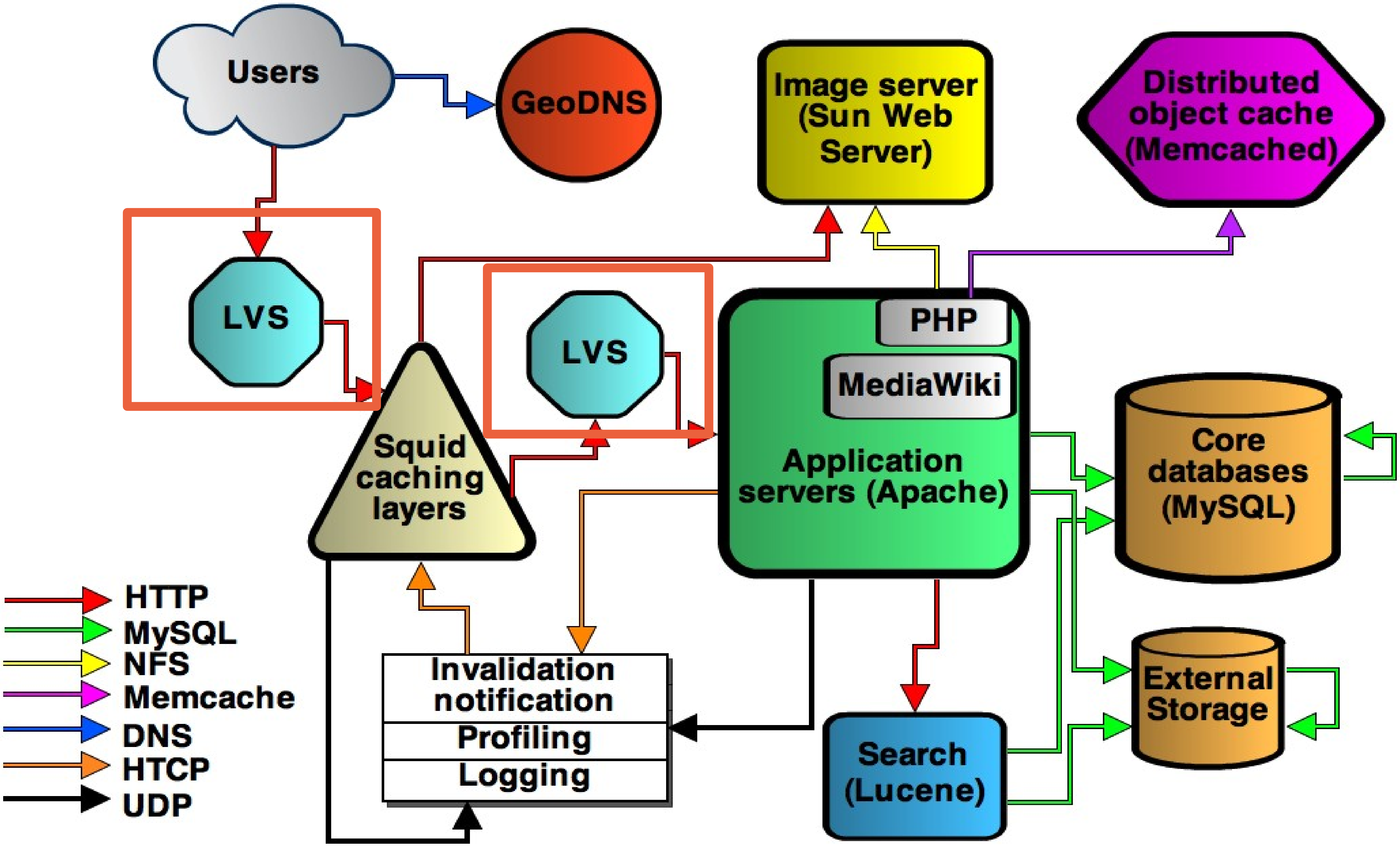

Origin server
Florida
(primary
cluster)
sq1
sq2
sq3
Caching
layer
sq1
sq1
sq3
Destination
URL hash
routing layer
Amsterdam
(caching
cluster)
Clients
Caching
layer
knsq1
knsq2
Destination
URL hash
routing layer
knsq1
knsq2

Squid 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

Varnish Caching

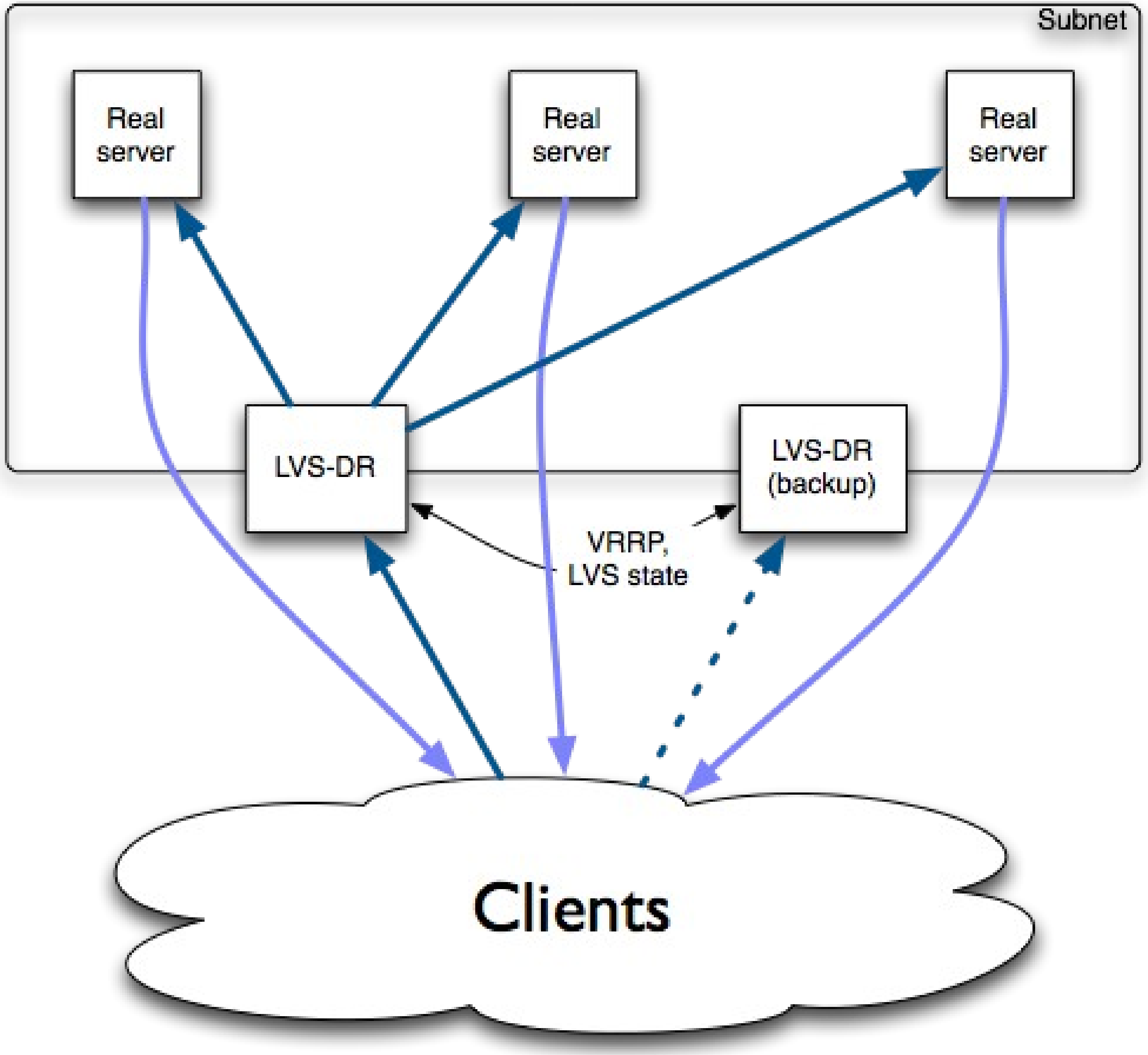
- 2-3 times more efficient than Squid
- Will eventually replace the Squid architecture
- Currently used for delivering static content such as javascript and css files (bits)

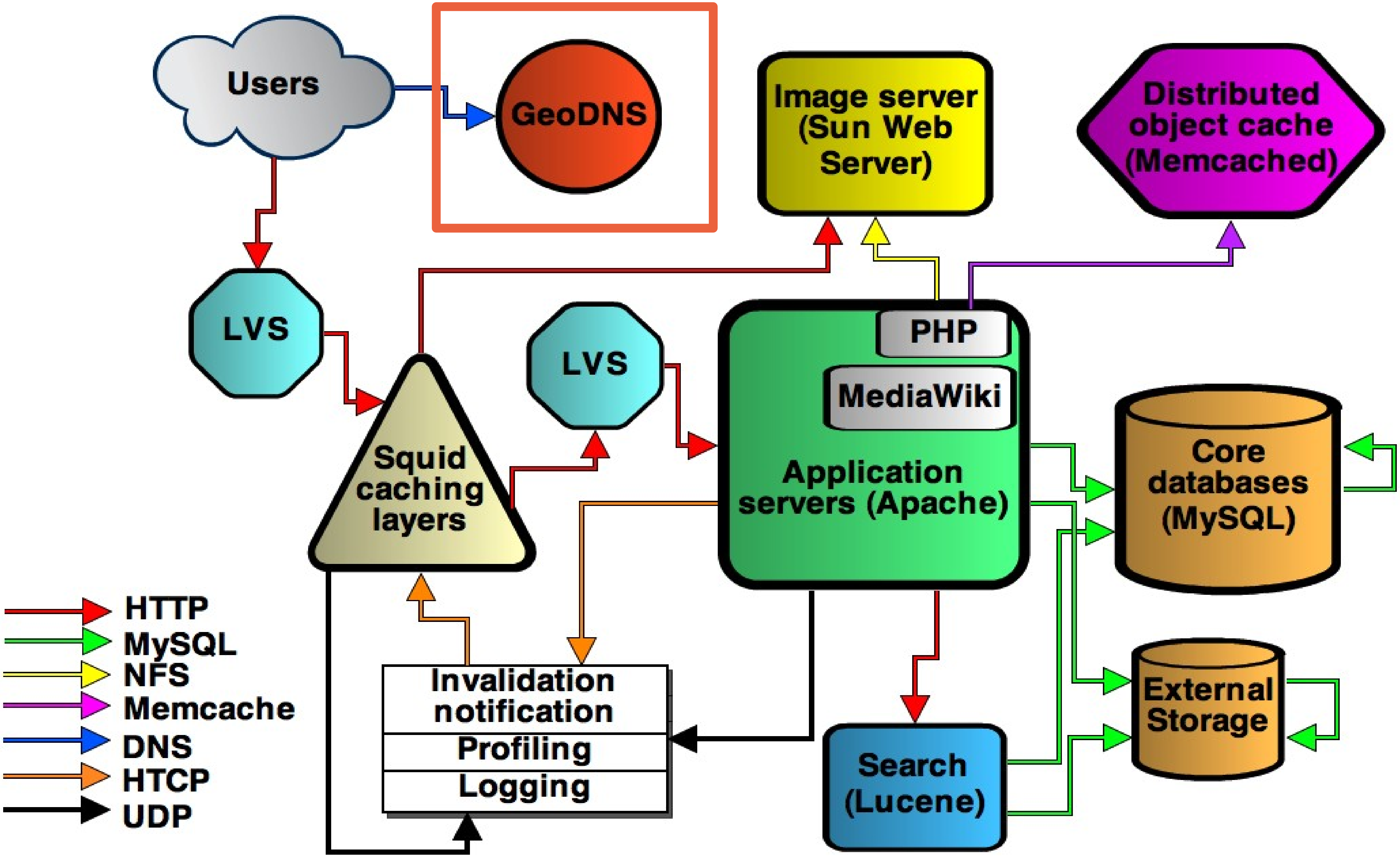


Users
GeoDNS
Image server (Sun Web Server)
Distributed object cache (Memcached)
LVS
LVS
PHP
MediaWiki
Application servers (Apache)
Squid caching layers
Core databases (MySQL)
External Storage
Search (Lucene)
Invalidation notification
Profiling
Logging
HTTP
MySQL
NFS
Memcache
DNS
HTCP
UDP

Load Balancing: LVS-DR

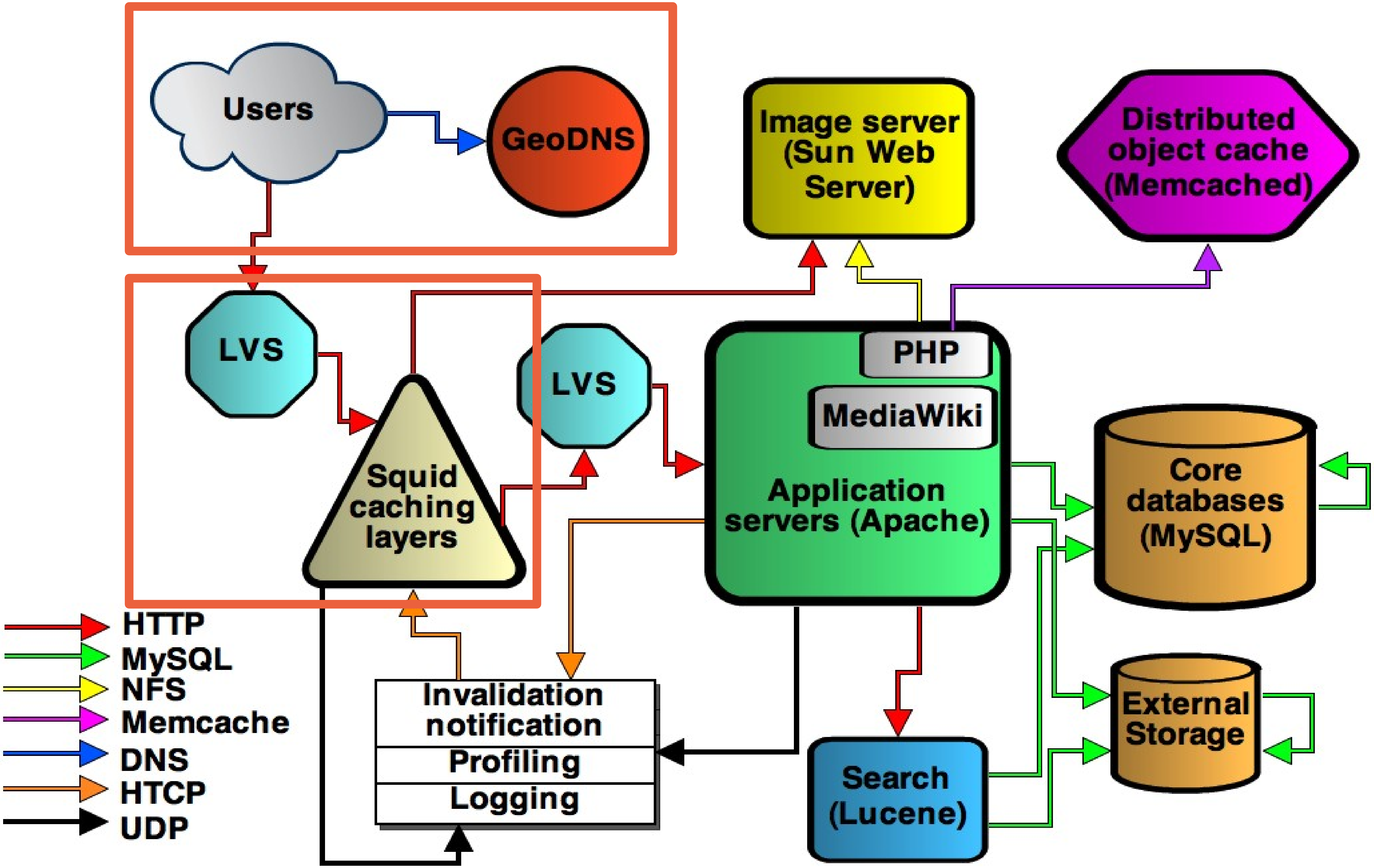
- Linux Virtual Server
- Direct Routing mode
- All real servers share the same IP address
- The load balancer divides incoming traffic over the real servers
- Return traffic goes directly!




Users
GeoDNS
Image server (Sun Web Server)
Distributed object cache (Memcached)
LVS
LVS
PHP
MediaWiki
Application servers (Apache)
Squid caching layers
Core databases (MySQL)
External Storage
Search (Lucene)
Invalidation notification
Profiling
Logging
HTTP
MySQL
NFS
Memcache
DNS
HTCP
UDP

Content Distribution Network (CDN)

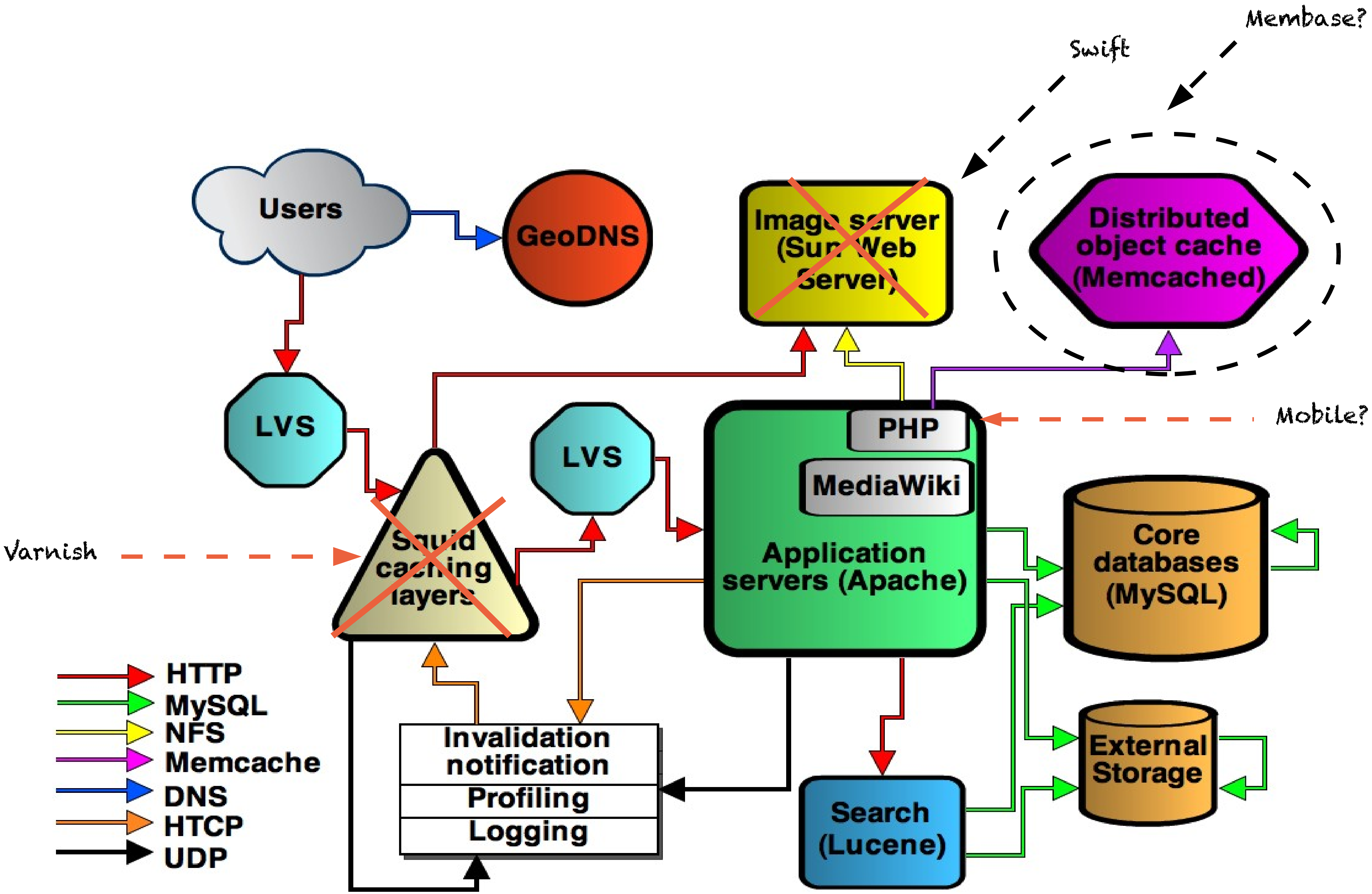
- 2 clusters on 2 different continents:
  - Primary cluster in Tampa, Florida
  - Secondary caching-only cluster in Amsterdam
- Adding a new primary datacenter in Virginia


Users
GeoDNS
Image server (Sun Web Server)
Distributed object cache (Memcached)
LVS
LVS
PHP
MediaWiki
Squid caching layers
Application servers (Apache)
Core databases (MySQL)
External Storage
Invalidation notification
Profiling
Logging
Search (Lucene)
HTTP
MySQL
NFS
Memcache
DNS
HTCP
UDP

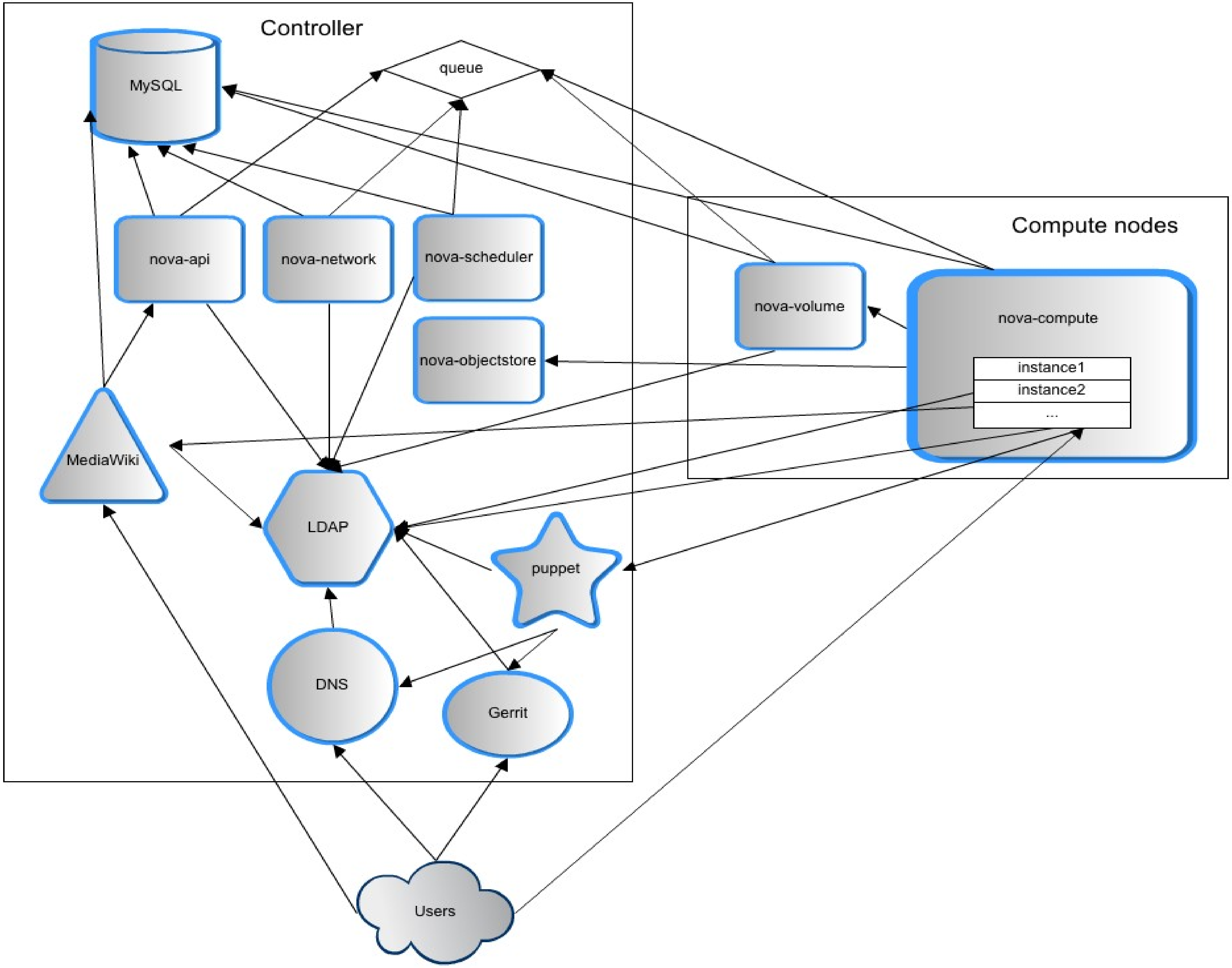
Geographic Load Balancing

- 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

The Site Architecture You Can Edit



Test/Dev Architecture



Basic use case

- Ops makes initial default project
  - Clone of production cluster
  - Used for most test/dev
- New projects mirror community or foundation initiatives
  - Devs build architecture in new project
  - Devs request merge for puppet changes via gerrit
  - Project instances moved to default project and tested
  - Project moved to production cluster

How to engage the community

- Discuss
- Commit
- Participate

How to engage the community

- Document
- Communicate changes

Our philosophy

- Engage early
- Release early, release often
- Scratch your own itch

Coding for WMF: Security

- Security is important. ***Really.***
- People rely on developers to write secure code, so:
  - An insecure extension in SVN...
  - An insecure extension on Wikipedia...

Common vulnerabilities to avoid

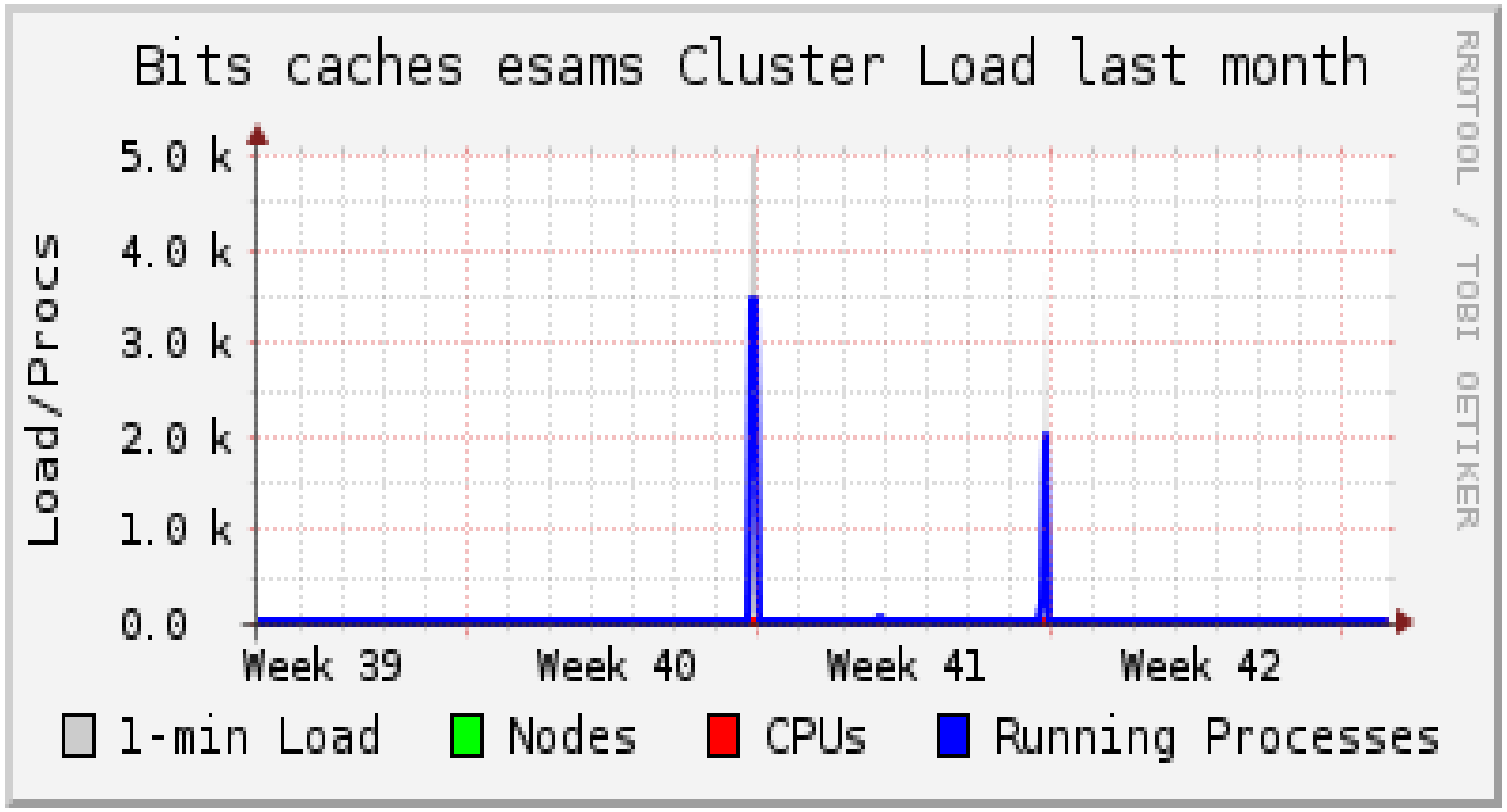
- SQL injection
- Cross site scripting (XSS)
- Cross site request forgery (CSRF)
- Register Globals

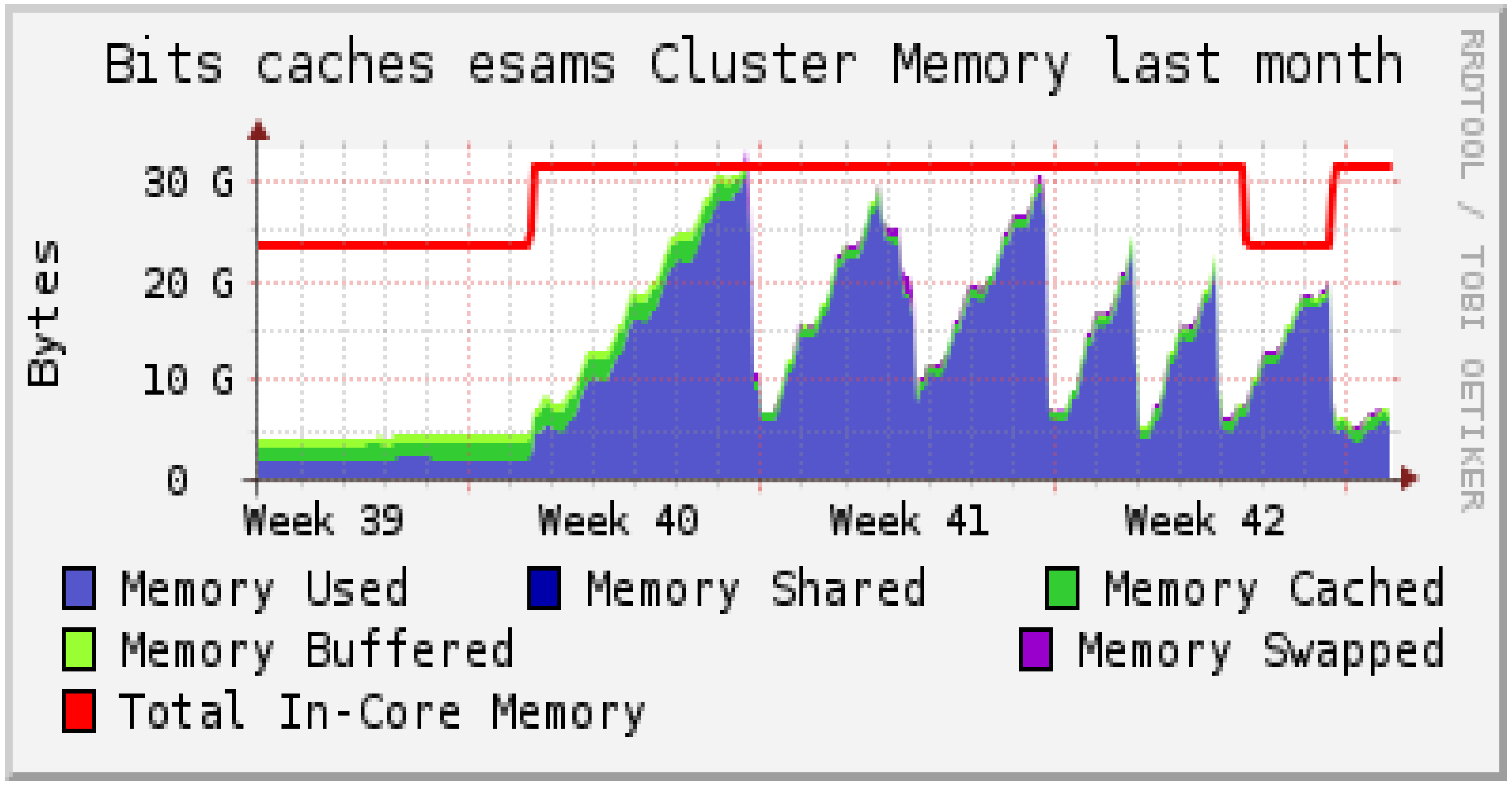
General notes on security

- Don't trust *anyone*
- Sanitize all input
- Write code that is *demonstrably secure*
- Best of all: try to break and hack your own code

Coding for WMF: Scalability and performance

- Wikimedia sites are huge
  - 5th most visited web presence
- Code must be:
  - Performant
  - Scalable


Bits caches esams Cluster Load last month
Load/Procs
5.0 k
4.0 k
3.0 k
2.0 k
1.0 k
0.0
Week 39
Week 40
Week 41
Week 42
1-min Load
Nodes
CPUs
Running Processes
RRDTOOL / TOBI OETIKER


Bits caches esams Cluster Memory last month
Bytes
30 G
20 G
10 G
0
Week 39
Week 40
Week 41
Week 42
Memory Used
Memory Shared
Memory Cached
Memory Buffered
Memory Swapped
Total In-Core Memory
RRDTOOL / TOBI OETIKER

Coding for WMF: Scalability and performance

- Cache
- Profile
- Optimize
- Ask for advice!

Coding for WMF: Concurrency

- Assume a clustered architecture, *always*
- Your code will run concurrently
  - It can result in strange bugs

Closing notes

- We rely heavily on open source
- Always looking for efficiencies
- Looking for more efficient management tools
- Looking for more contributors

Questions, comments?

- E-mail: Ryan Lane <ryan@wikimedia.org>
- IRC: Freenode, Nick: Ryan_Lane, Channels: #wikimedia-tech, #wikimedia-operations, #mediawiki, #openstack

Communication resources

- Mailing lists
  - http://www.mediawiki.org/wiki/Mailing_lists
  - Important lists:
    - mediawiki-l: A MediaWiki support list
    - wikitech-l: A MediaWiki developer's list
    - mediawiki-api: A MediaWIki developer's list for the API
- IRC channels (on freenode)
  - #mediawiki: A MediaWiki support channel

Developer resources

- http://www.mediawiki.org/wiki/Developer_hub/ja - developer hub
  - Developer hub: lists resources, guidelines, and code documentation
- http://www.mediawiki.org/wiki/How_to_become_a_MediaWiki_hacker/ja
  - How to become a MediaWiki hacker: introduction into how to do MediaWiki development
- http://www.mediawiki.org/wiki/Security_for_developers
  - Security for developers: essential security documentation

Developer resources

- http://www.mediawiki.org/wiki/Manual:Coding_conventions/ja
  - Coding conventions: conventions required for all Wikimedia run software
- http://www.mediawiki.org/wiki/Localisation/ja
  - Localisation: resources to write code that can be easily localised
- http://www.mediawiki.org/wiki/Code_review_guide
  - Code review guide: how your code will be reviewed before inclusion