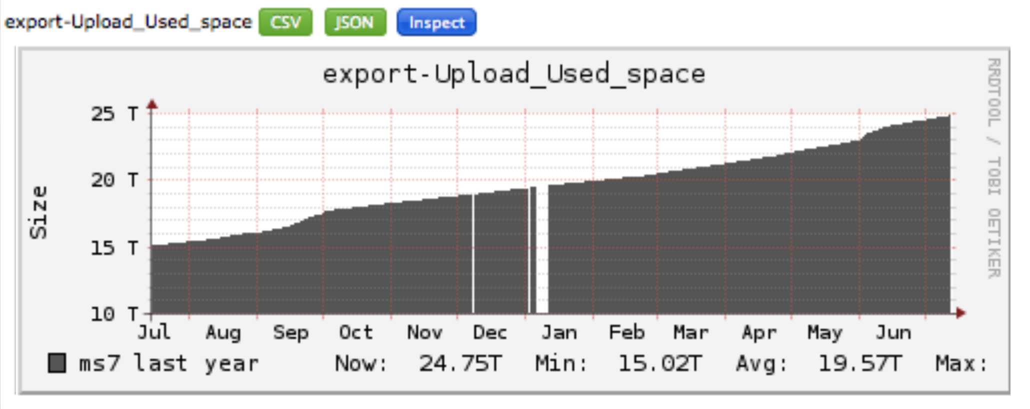
WIKIMEDIA
FOUNDATION

Swift and Media Storage at Wikimedia

Ben Hartshorne
Operations Engineer
<bhartshorne@wikimedia.org>

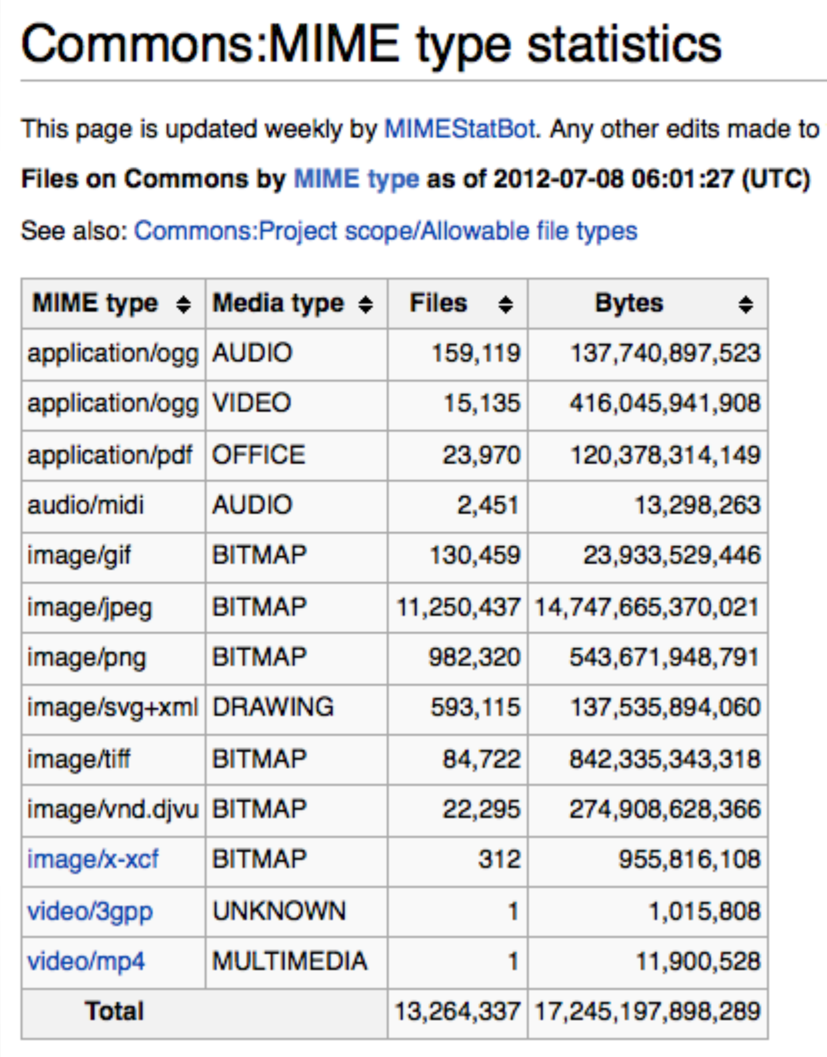
What is Media Storage?

- All images, sounds, and videos on all wikis
- All scaled versions of all those images
- It just keeps growing..



http://ganglia.wikimedia.org/latest/?r=year&cs=&ce=&m=&c=Miscellaneous+pmtpa&h=ms7.pmtpa.wmnet&tab=m&vn=&mc=2&z=medium&metric_group=ALLGROUPS

http://commons.wikimedia.org/wiki/Commons:



Commons:MIME type statistics

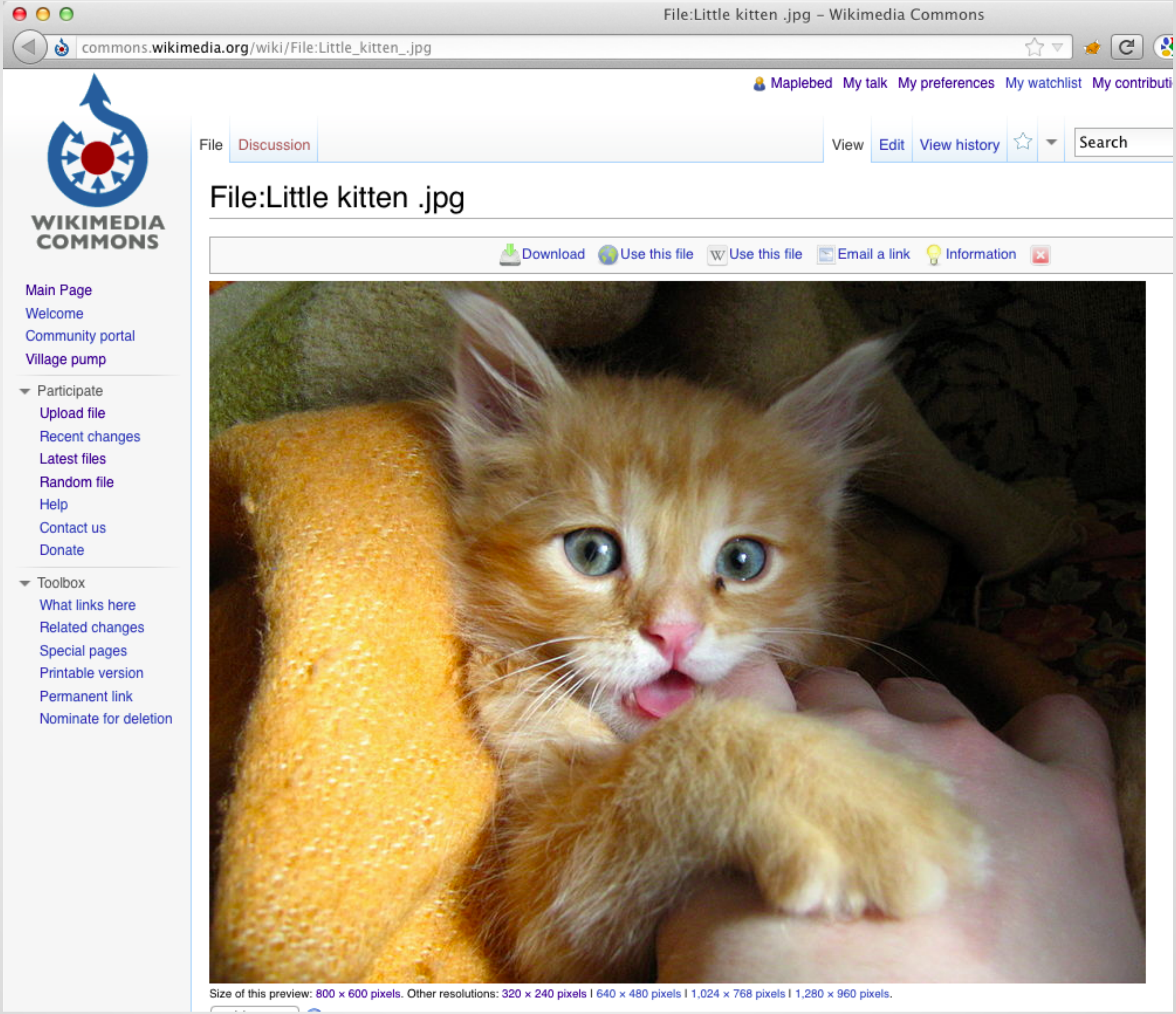
This page is updated weekly by MIMEStatBot. Any other edits made to

Files on Commons by MIME type as of 2012-07-08 06:01:27 (UTC)

See also: Commons:Project scope/Allowable file types

MIME type	Media type	Files	Bytes
application/ogg	AUDIO	159,119	137,740,897,523
application/ogg	VIDEO	15,135	416,045,941,908
application/pdf	OFFICE	23,970	120,378,314,149
audio/midi	AUDIO	2,451	13,298,263
image/gif	BITMAP	130,459	23,933,529,446
image/jpeg	BITMAP	11,250,437	14,747,665,370,021
image/png	BITMAP	982,320	543,671,948,791
image/svg+xml	DRAWING	593,115	137,535,894,060
image/tiff	BITMAP	84,722	842,335,343,318
image/vnd.djvu	BITMAP	22,295	274,908,628,366
image/x-xcf	BITMAP	312	955,816,108
video/3gpp	UNKNOWN	1	1,015,808
video/mp4	MULTIMEDIA	1	11,900,528
Total		13,264,337	17,245,197,898,289

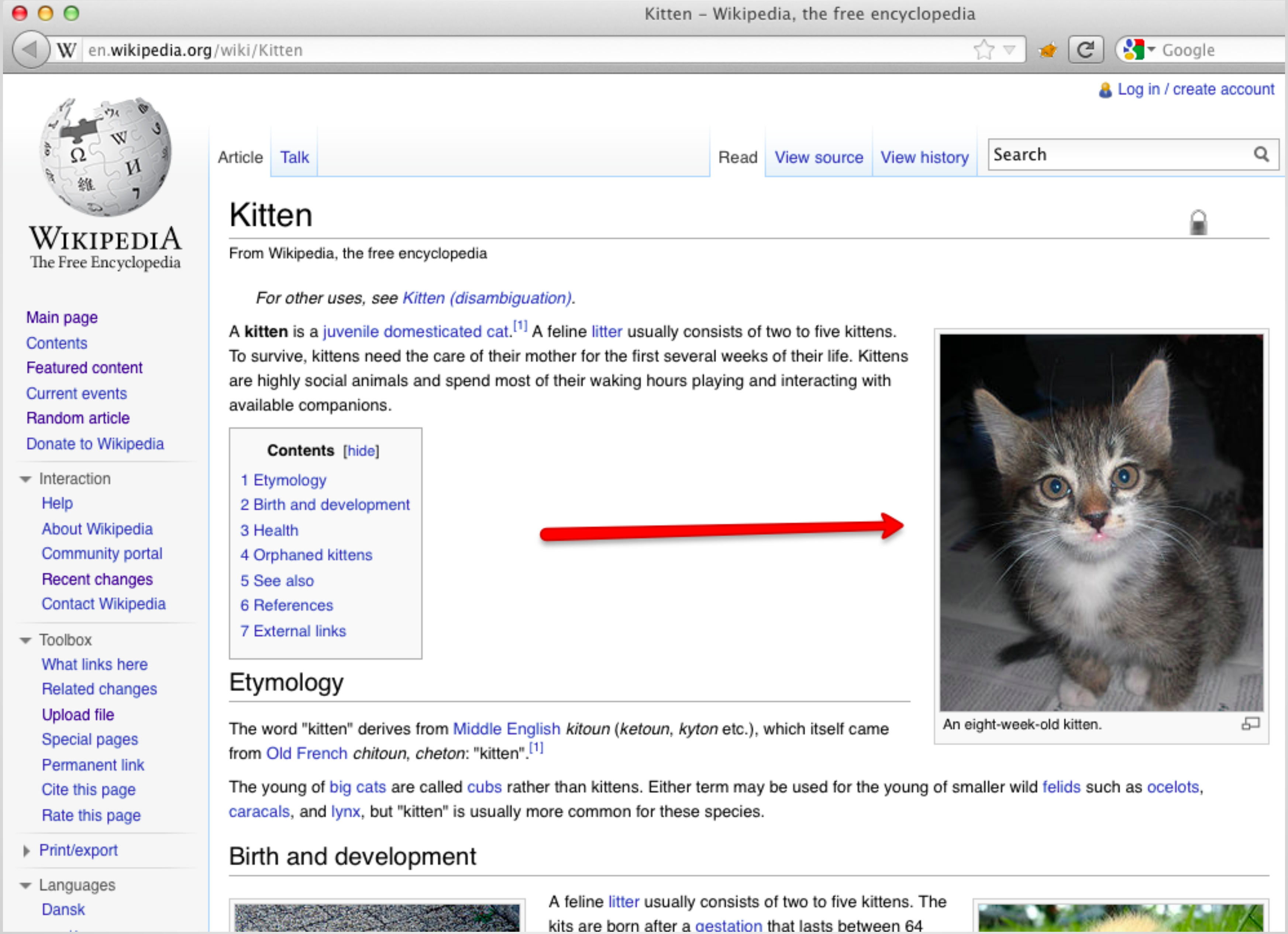
Original Media



File:Little kitten .jpg

Size of this preview: 800 × 600 pixels. Other resolutions: 320 × 240 pixels | 640 × 480 pixels | 1,024 × 768 pixels | 1,280 × 960 pixels.

Thumbnails (Scaled Media)



Kitten – Wikipedia, the free encyclopedia

en.wikipedia.org/wiki/Kitten

Google

Log in / create account

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox
What links here
Related changes
Upload file
Special pages
Permanent link
Cite this page
Rate this page

Print/export

Languages
Dansk

Article | Talk | Read | View source | View history | Search

Kitten

From Wikipedia, the free encyclopedia

For other uses, see Kitten (disambiguation).

A **kitten** is a juvenile domesticated cat.[1] A feline litter usually consists of two to five kittens. To survive, kittens need the care of their mother for the first several weeks of their life. Kittens are highly social animals and spend most of their waking hours playing and interacting with available companions.

Contents [hide]

An eight-week-old kitten.

Etymology

The word "kitten" derives from Middle English *kitoun* (*ketoun*, *kyton* etc.), which itself came from Old French *chitoun*, *cheton*: "kitten".[1]

The young of big cats are called cubs rather than kittens. Either term may be used for the young of smaller wild felids such as ocelots, caracals, and lynx, but "kitten" is usually more common for these species.

Birth and development

A feline litter usually consists of two to five kittens. The kits are born after a gestation that lasts between 64

What do we need from a Media Store?

- Large Capacity
  - currently 25TB; minimum for growth: 50-100TB
- Fault Tolerence
  - any component must be able to fail without impact
- Medium Throughput
  - rate of image requests, additions is about 100/s
- Medium Latency
  - most end-user actions are served from cache

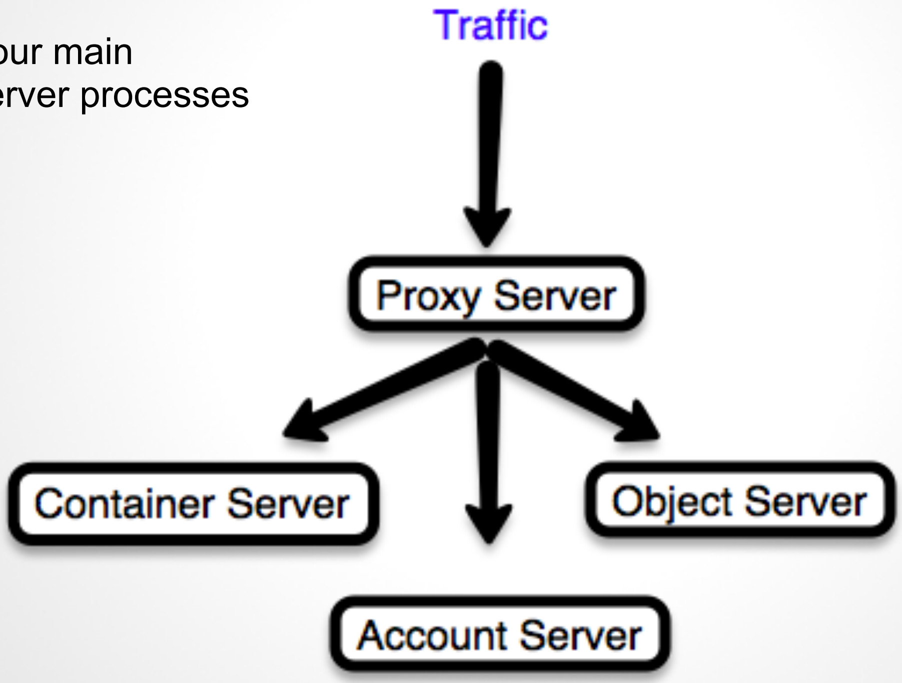
What is Swift?

OpenStack Storage (http://openstack.org/software/openstack-storage/)

<buzzword>scalable fault tolerant object store</buzzword>

- Scalable: Increase cluster size (and throughput) by adding additional hardware; latency shouldn't increase with cluster size
- Fault Tolerant: No single point of failure
- Object Store: Not a filesystem - stores whole objects

Swift Architecture

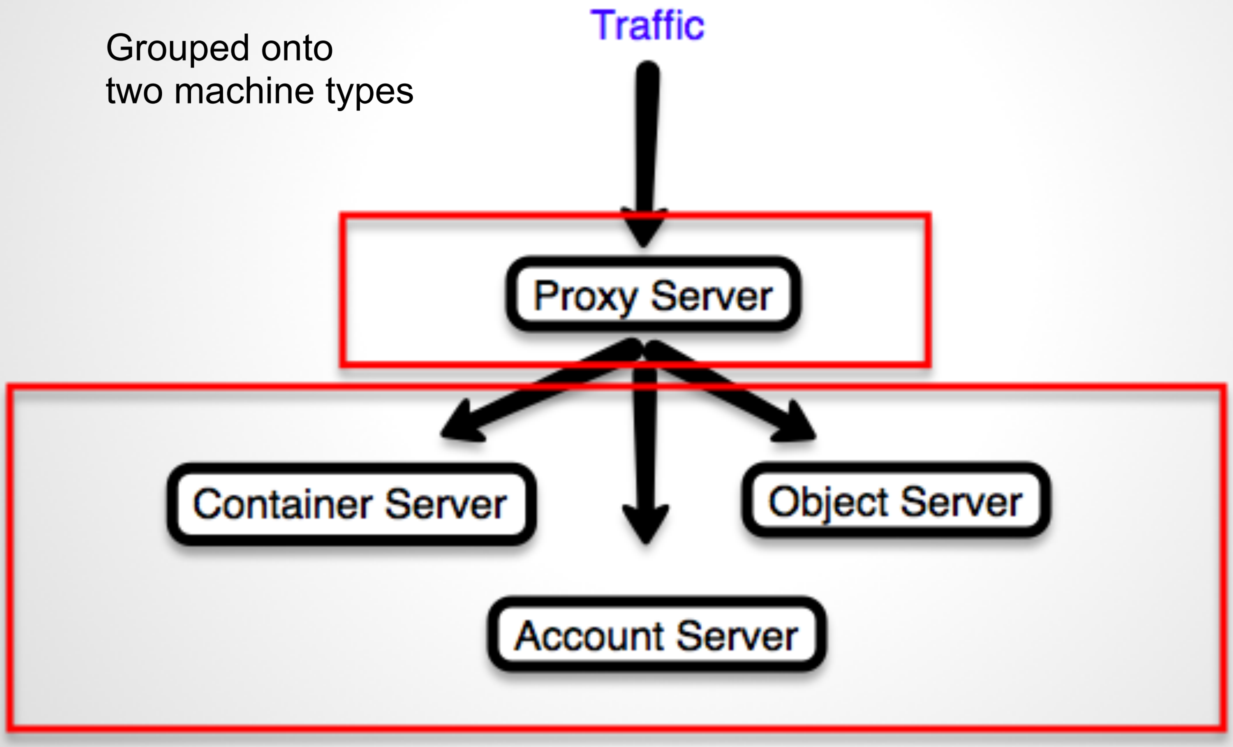


Four main
server processes

Many ancillary processes for background jobs: synchronization, auditing, replication, etc.

Swift Architecture

Grouped onto
two machine types

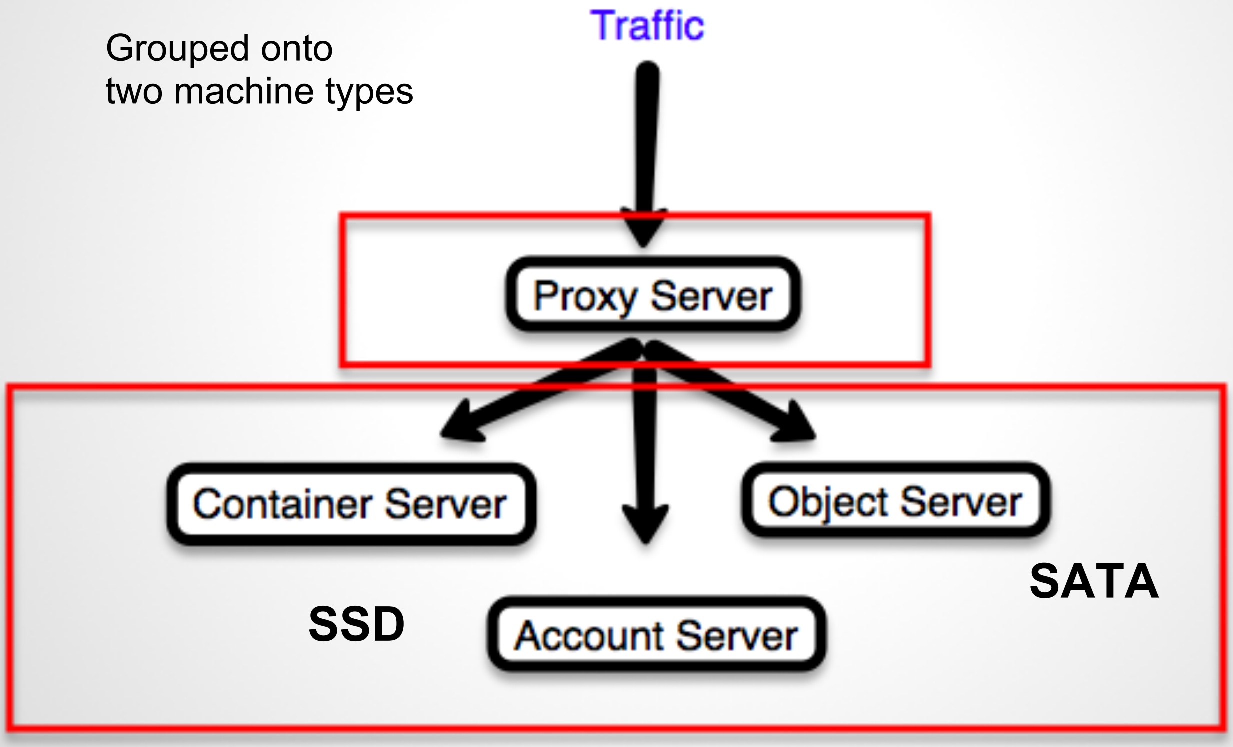


Machine Types

- Frontend Proxy Server
    - dual 6-core CPU
    - 16GB RAM
    - two 250GB SATA disks RAID1

- Backend Storage Server
    - dual 6-core CPU
    - 48GB RAM
    - two 160GB SSDs
    - twelve 2TB SATA disks (no RAID)

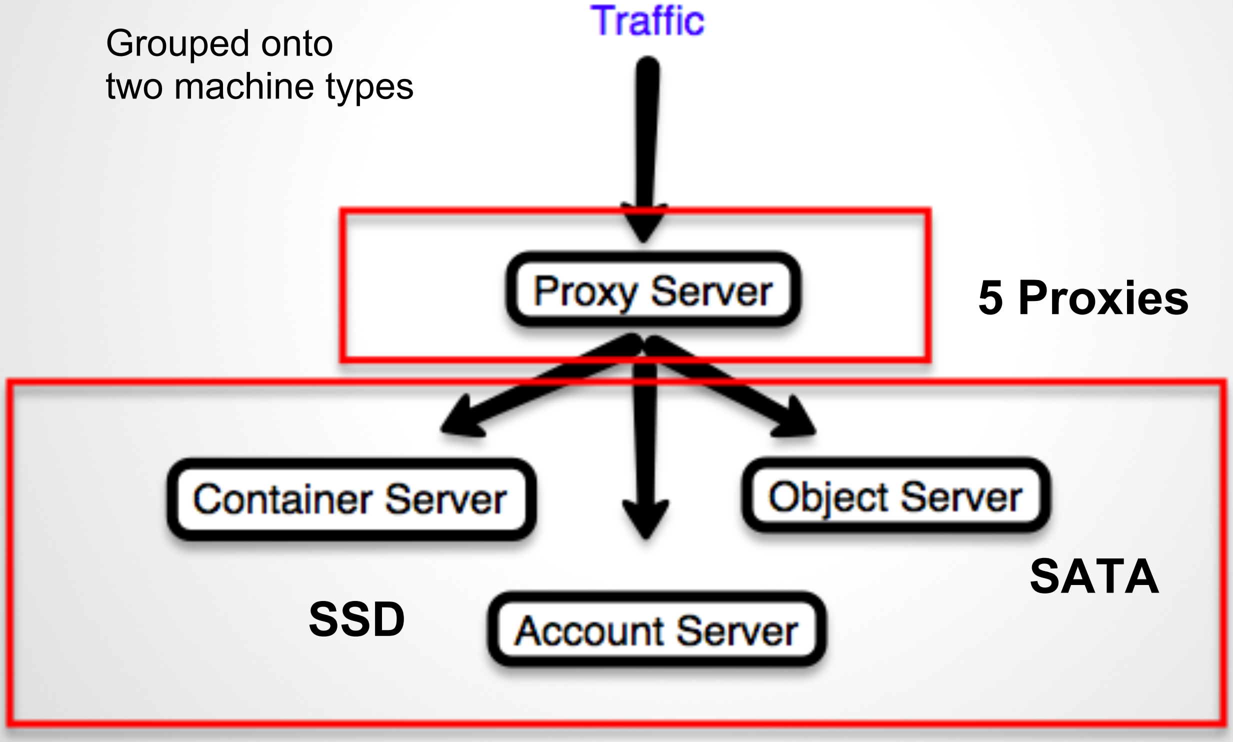
Swift Architecture



Grouped onto
two machine types

Swift Architecture

Grouped onto
two machine types



5 Proxies

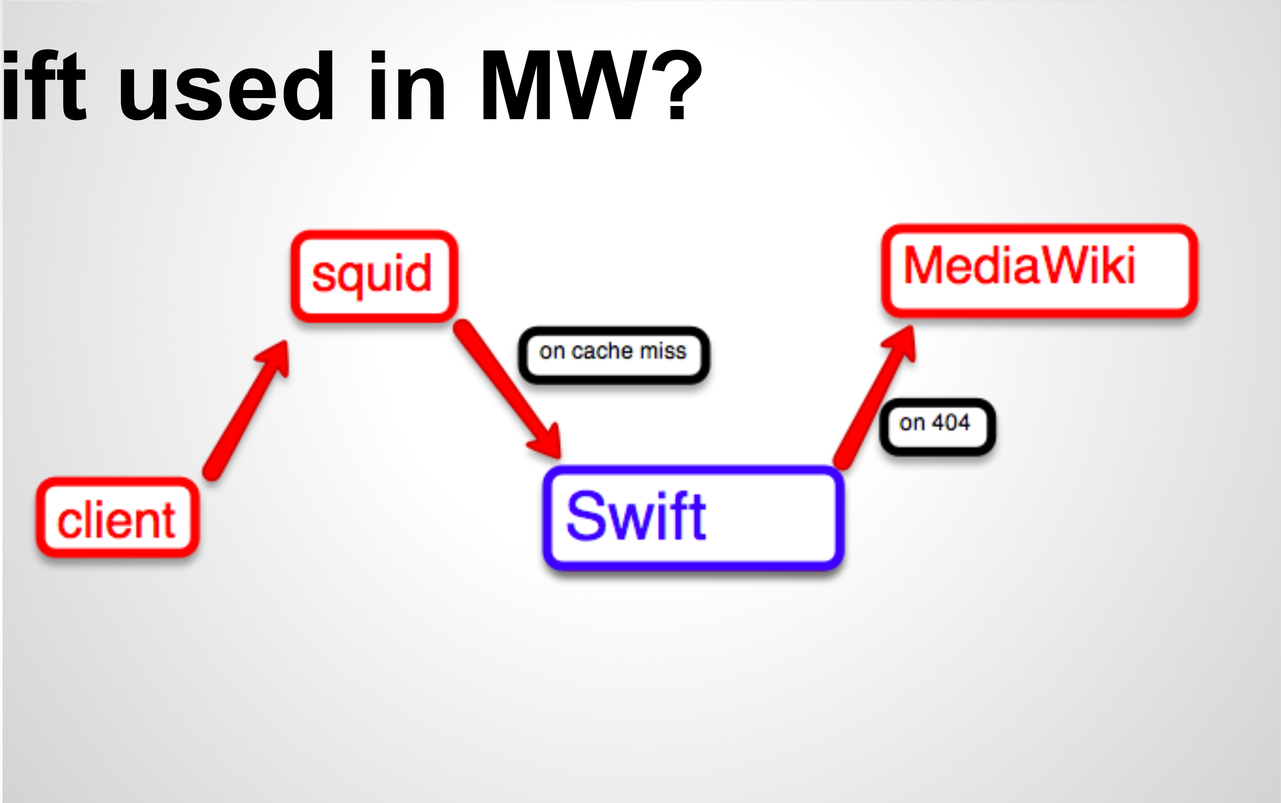
12 Storage

SSD

SATA

How is Swift used in MW?

- Thumbnails



- Originals
  - Mediawiki FileBackend class has multiple modules; calls Swift using CloudFiles

WIKIMEDIA
FOUNDATION

Thanks!

Ben Hartshorne
Operations Engineer
<bhartshorne@wikimedia.org>

http://wikitech.wikimedia.org/view/Swift

end of presentation

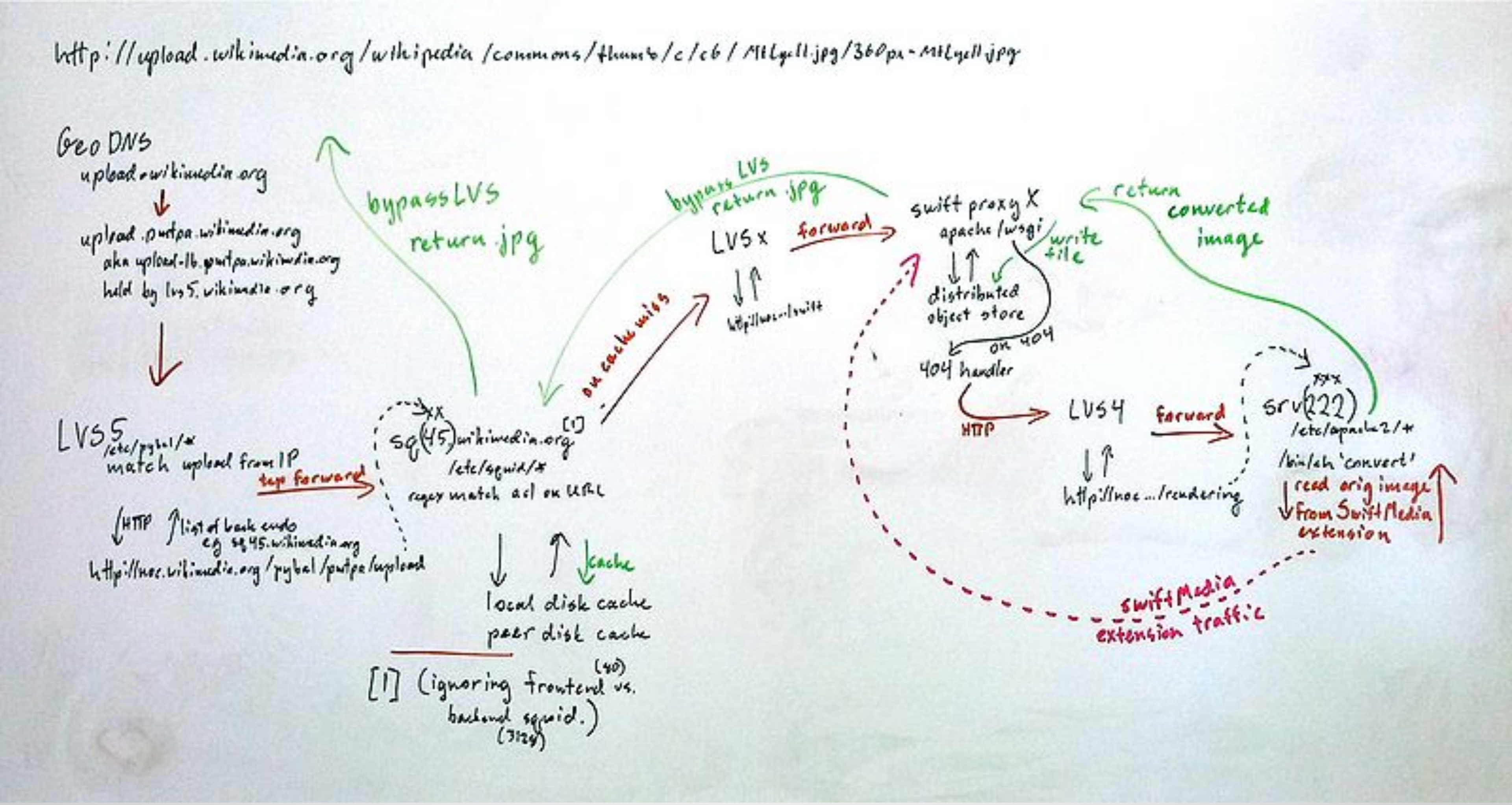
some optional slides follow - they might be used if people ask specific questions

Rewrite middleware

- New thumbnails are scaled on demand
- 404 handler tries to scale images
  that don't exist
- swift-proxy is built for this
  - in /etc/swift/proxy-server.conf:

    ```
    [pipeline:main]
    pipeline = rewrite healthcheck cache swauth proxy-server
    ```
- rewrite does two things
  - call back to get the scaled version of the image
  - write that scaled version into swift

Query flow: client to scaled image



What about that 404 handler?

Perfect for middleware in the proxy pipeline

```
[pipeline:main]
pipeline = rewrite healthcheck cache swauth proxy-server
```

Rewrite does two things:

- Handle 404s
  - if the object doesn't exist in swift
  - call back to mediawiki to generate the image
  - optionally write the generated image into swift

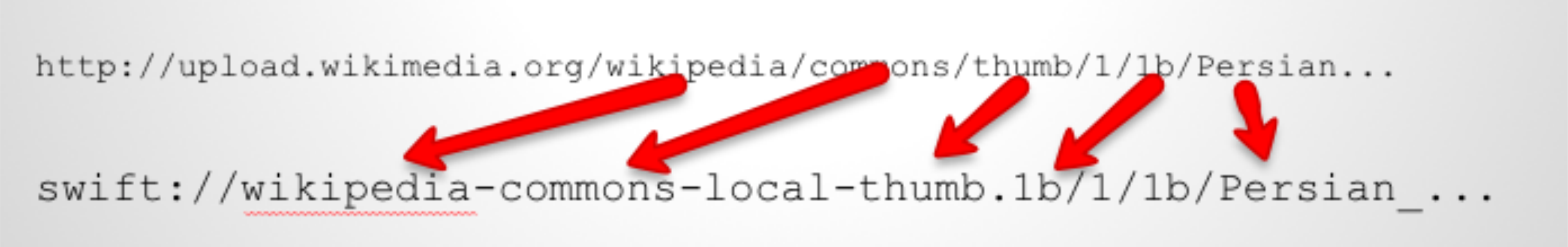
What about that 404 handler?

Perfect for middleware in the proxy pipeline

```
[pipeline:main]
pipeline = rewrite healthcheck cache swauth proxy-server
```

Rewrite does two things:

- Change URL into Container / Object



```
http://upload.wikimedia.org/wikipedia/commons/thumb/1/1b/Persian...

swift://wikipedia-commons-local-thumb.1b/1/1b/Persian_...
```

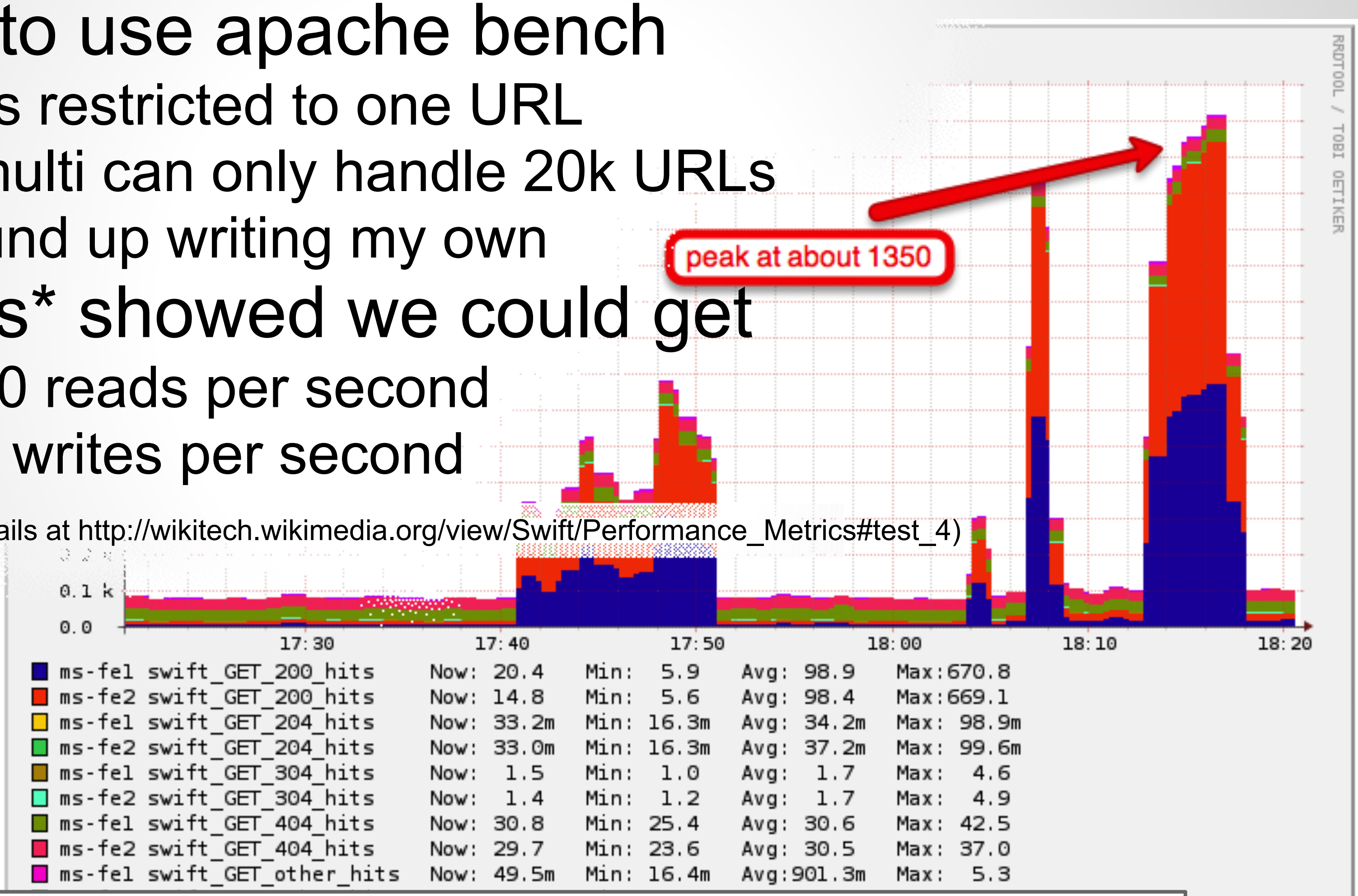
Integration with Mediawiki

- MW storage mechanisms abstracted to a FileBackend class with multiple subclasses
  - local filesystem, swift, azure, S3, etc.
- All interactions with the FileBackend implemented as appropriate for each backend storage module
- Swift storage implemented using CloudFiles
  - https://github.com/rackspace/php-cloudfiles
- More detail on this part: Aaron Schulz

Throughput and Latency Performance

Initial tests

- Tried to use apache bench
  - ab is restricted to one URL
  - abmulti can only handle 20k URLs
  - wound up writing my own
- geturls* showed we could get
  - 1300 reads per second
  - 120 writes per second
  - (full details at http://wikitech.wikimedia.org/view/Swift/Performance_Metrics#test_4)



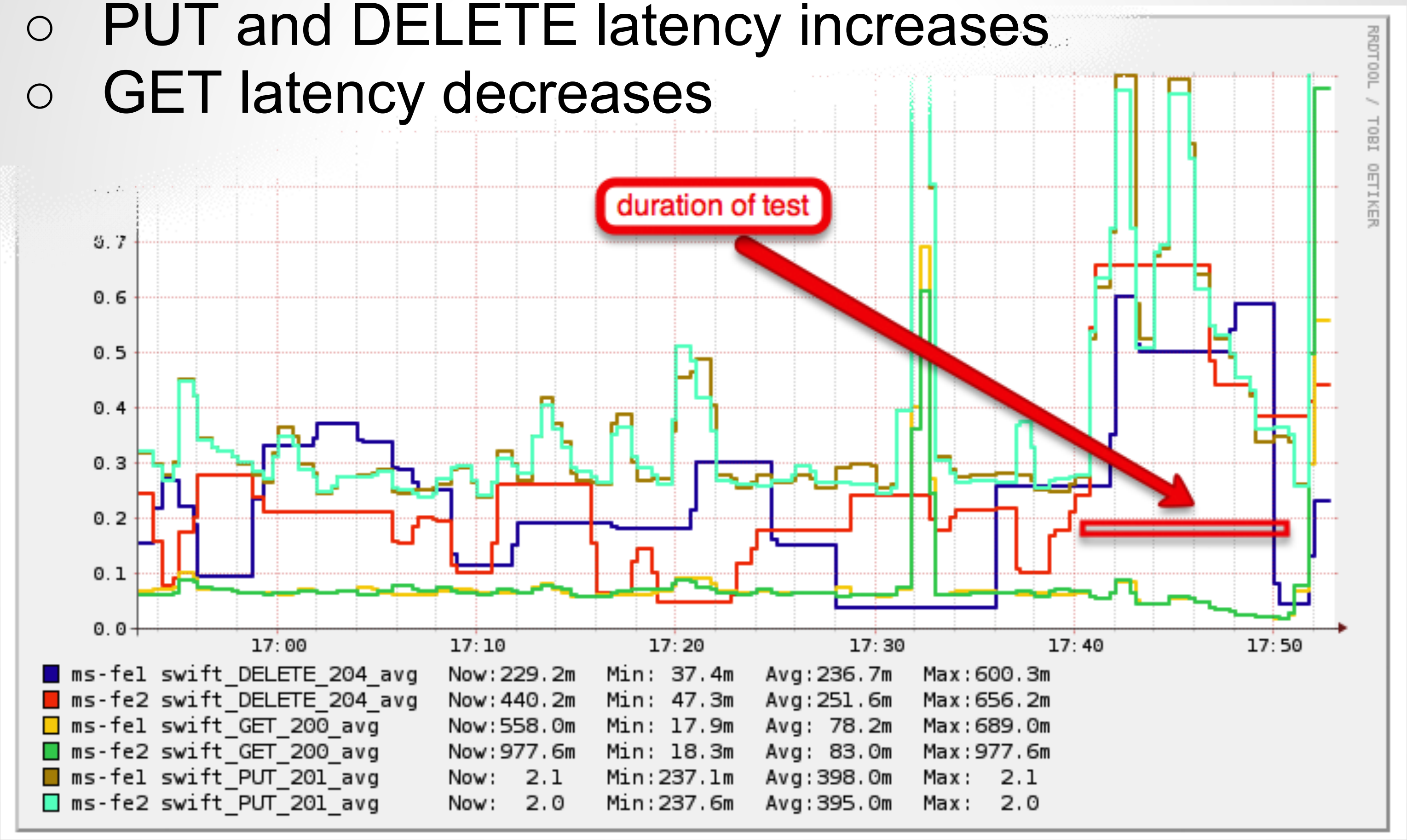
peak at about 1350

```
🟦 ms-fe1 swift_GET_200_hits    Now: 20.4   Min:  5.9   Avg: 98.9   Max:670.8
🟥 ms-fe2 swift_GET_200_hits    Now: 14.8   Min:  5.6   Avg: 98.4   Max:669.1
🟨 ms-fe1 swift_GET_204_hits    Now: 33.2m  Min: 16.3m  Avg: 34.2m  Max: 98.9m
🟩 ms-fe2 swift_GET_204_hits    Now: 33.0m  Min: 16.3m  Avg: 37.2m  Max: 99.6m
🟫 ms-fe1 swift_GET_304_hits    Now:  1.5   Min:  1.0   Avg:  1.7   Max:  4.6
🟩 ms-fe2 swift_GET_304_hits    Now:  1.4   Min:  1.2   Avg:  1.7   Max:  4.9
🟩 ms-fe1 swift_GET_404_hits    Now: 30.8   Min: 25.4   Avg: 30.6   Max: 42.5
🟥 ms-fe2 swift_GET_404_hits    Now: 29.7   Min: 23.6   Avg: 30.5   Max: 37.0
🟪 ms-fe1 swift_GET_other_hits  Now: 49.5m  Min: 16.4m  Avg:901.3m  Max:  5.3
```

* geturls code available at https://gerrit.wikimedia.org/r/gitweb?p=operations/software.git;a=tree;f=geturls;hb=HEAD

Effect of load on performance

- Under heavy read load
  - PUT and DELETE latency increases
  - GET latency decreases

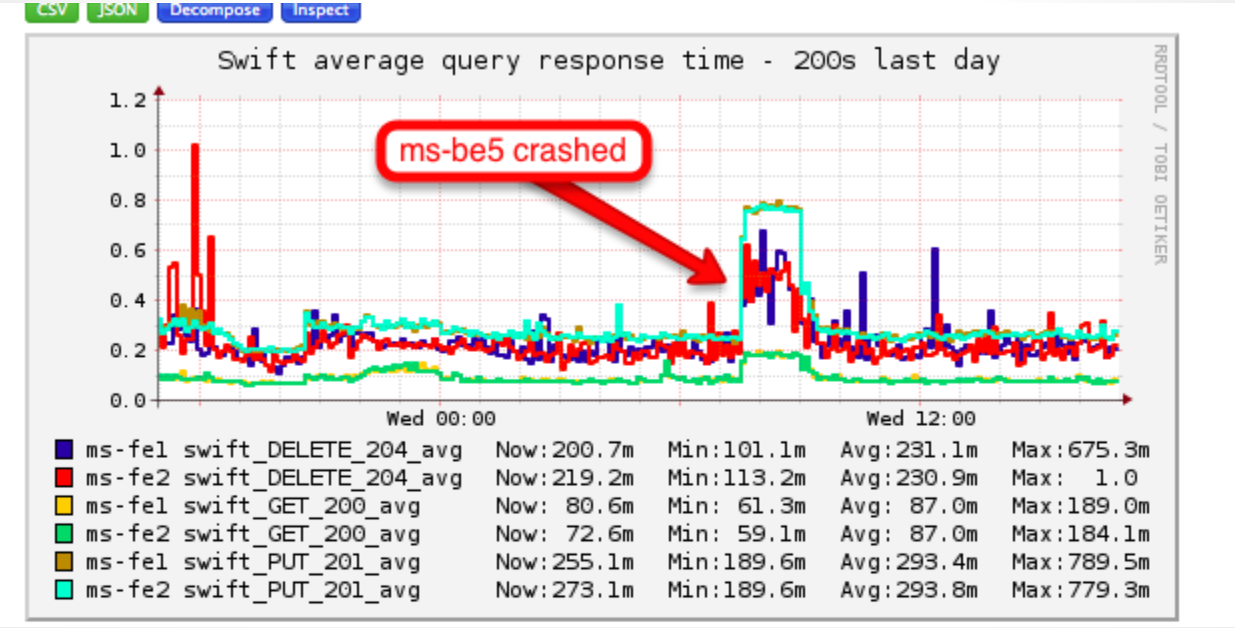


duration of test

```
🟦 ms-fe1 swift_DELETE_204_avg  Now:229.2m  Min: 37.4m  Avg:236.7m  Max:600.3m
🟥 ms-fe2 swift_DELETE_204_avg  Now:440.2m  Min: 47.3m  Avg:251.6m  Max:656.2m
🟨 ms-fe1 swift_GET_200_avg     Now:558.0m  Min: 17.9m  Avg: 78.2m  Max:689.0m
🟩 ms-fe2 swift_GET_200_avg     Now:977.6m  Min: 18.3m  Avg: 83.0m  Max:977.6m
🟫 ms-fe1 swift_PUT_201_avg     Now:   2.1  Min:237.1m  Avg:398.0m  Max:   2.1
🟩 ms-fe2 swift_PUT_201_avg     Now:   2.0  Min:237.6m  Avg:395.0m  Max:   2.0
```

Effect of node failure

- One (out of 5) storage nodes crashing
  - 0.5s timer on connection failures - adjustable
  - 2x read latency (from 100ms to 200ms)
  - 3x write latency (250ms to 750ms)
  - 2.5x delete latency (200ms to 500ms)
- No data (yet) on proxy nodes crashing



ms-be5 crashed

```
■ ms-fe1 swift_DELETE_204_avg   Now:200.7m  Min:101.1m  Avg:231.1m  Max:675.3m
■ ms-fe2 swift_DELETE_204_avg   Now:219.2m  Min:113.2m  Avg:230.9m  Max:  1.0
■ ms-fe1 swift_GET_200_avg      Now: 80.6m  Min: 61.3m  Avg: 87.0m  Max:189.0m
■ ms-fe2 swift_GET_200_avg      Now: 72.6m  Min: 59.1m  Avg: 87.0m  Max:184.1m
■ ms-fe1 swift_PUT_201_avg      Now:255.1m  Min:189.6m  Avg:293.4m  Max:789.5m
■ ms-fe2 swift_PUT_201_avg      Now:273.1m  Min:189.6m  Avg:293.8m  Max:779.3m
```

Some problems encountered along the way

- Effect of one storage node crashing on performance is too large
  - solved by reducing the connection timeout from 0.5s to 0.1s
- Container listing latency is high
  - solved by moving container and account servers to SSD leaving objects on spinning media
- Consistency issues with rewrite middleware
  - ETags help
  - Still have issues sometimes (cleaner script)
  - solved by having mediawiki write to swift instead
- It's difficult diagnosing problems with rewrite
  - natural effect of asynchronous code (eventlet)
  - eg. stack trace in proxy logs