Services

Quarterly review Q1 2014/15

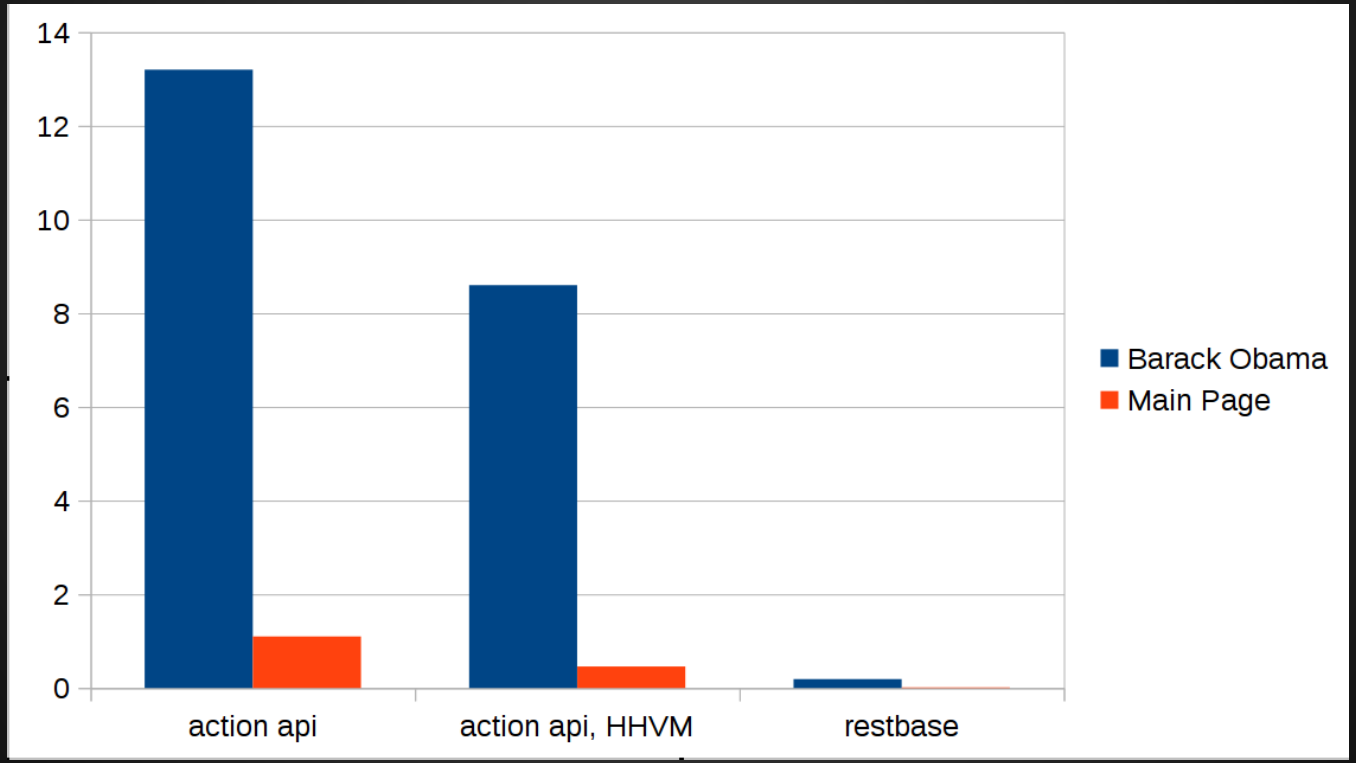
High-level goals

Very high level: Move towards services (see RFCs)
- for technical & organizational reasons

Abstract out and scale storage layer
- high-level interface
- handle complexities of distributed storage systems (consistency, indexing)
- address existing DB scaling challenges (external store, revisions, link tables)
- cross-DC replication

Provide a fast and consistent content / storage API
- cacheable, low per-request overhead
- backed by storage and backend services
- provide generic monitoring and orchestration for backend services

High-level goals



High-level goals

Eliminate slow-down for authenticated page views
- 95+ % of traffic already served from Varnishes
- most app servers busy generating custom content HTML for logged-in views
- at most only 50 edits/s, typical 10/s across all projects
- planned to expand this to logged-in editors since 2004
- requires client-side user preference implementation
- also improves VE load time

Support other teams with storage, API & service needs
- especially Parsoid, VE, Mobile, Flow

Security
- least privilege, signed capabilities, efficient API requests

Resourcing

Matt decided to leave July 1st (to build flying cars)

Hardik Juneja officially joined in August (part-time, student in Delhi)
- working on RESTBase secondary indexing

Re-did JDs, added senior position; seem to finally get some traction
- two volunteers in last two days
- several promising applications
- looking for 1-2 *senior* candidates out of three open reqs
  - ideally with distributed storage & security skills

Original plan for Q1

- Implement first iteration of REST API front-end (restface)
  - Alpha deploy to api.wikimedia.org
  - Simple proof of concept implementation of page metadata end point for Parsoid HTML views (redlinks etc)
- First iteration on Rashomon storage service (with versioned blob and queue buckets & lame auth)
  - Deploy & start using it with Parsoid
  - Wrap HTML load/save in restface
- PDF render service deployment
- Citation service
- Mathoid deploy
- HTML templating: documentation, in particular KnockOff compiler & PHP implementation
  - Iterate once we have feedback from users & think more about use for content & messages

RESTBase

Merger of restface & rashomon (now RESTBase -- Cassandra backend package)

- REST content API backed by high-level storage
- calls out and orchestrates backend services; central point for monitoring, logging etc
- tables similar to DynamoDB: automatic secondary indexes, ideas for 2PC transactions
  - originally planned for Q3, but made sense to do earlier
- layered abstractions on top of tables: buckets, meta buckets
  - much simplified by table abstraction layer

Deployed in labs, prod pending puppetization (next weeks) - slightly behind schedule

- Parsoid to pull in data from RESTBase on HTML save

Misc backend services

Citoid (url -> citation): just deployed

- Marielle Volz (OPW), Matt helped a lot early on
- all the credit for deployment work to Roan (thanks!)
- now waiting for VE integration

Mathoid (latex -> SVG & MathML): just deployed, but not used yet from MediaWiki

- Moritz Schubotz and MathJax folks with some help from Gabriel
- again, all deployment work credit to Roan
- unclear ownership of MediaWiki integration

Misc backend services

PDF rendering / OCG: just deployed

- mostly Matt & Scott's work with help from Max, Brad, Arlo & Gabriel
- next step: simplify, focus on HTML + CSS, use phantomjs 2
- Scott drives this; more details later

HTML templating

Goals:

- best-of-class performance
- handle XSS issues automatically (style, href, src); automate security review
- ideally reuse existing reactive framework rather than rolling our own
- choose solution that supports isomorphic rendering well

Demonstrated that secure & efficient can be combined with KnockOff / TAssembly

- documented these
- added one feature requested by mobile after eval

Then nothing happened for months..

Recently had a productive meeting with front-end standardization group

- there seems to be a better shared understanding of issues now

Lets call this a cautious success.

Q2

Fill open positions. Q2 goals assume some success with that.

Main area:

- RESTBase
- Edit & page view performance
- Security
- Tooling / infrastructure

Q2: RESTBase

finish secondary index implementation

auto-generate API documentation (Swagger 2)

systematic performance & error monitoring for both RESTBase & backend services

request flow tracing & logging

entry points:

- misc services (citations, math, parsoid, PDF?)
- work with mobile apps & other teams on their needs
- HTML & wikitext saves
- refine PHP API wrapping: identify cacheable endpoints

reliable event queuing

- for reliable updates / purges, async HTTP jobs
- Kafka backend (already have ops experience from analytics)

Q2: Performance

RESTBase storage

- < 15ms 95th for small (<10k) resources, <250ms 95th for 2mb resources

Editing

- eliminate Varnish cache misses with RESTBase deploy
- start using Parsoid HTML for page views
  - no need to load any HTML on edit (only small amount of JSON metadata)
  - finish pushing all user preferences to the client, same HTML for everyone
  - develop gadget into beta feature
- help VE implement fast saves (don't wait for re-parse, just display HTML)
- investigate HTML-based abuse filter / non-wikitext save pipeline

Logged-in page views

- static HTML also *eliminates* content-related slow page loads for editors (this is big!)

Q2: Security

Goals

- least privilege
  - don't trust (most) code, trust time-limited signatures
  - aim for limiting access to user data to single auth service
- cheap signature verification for common ops

converged on rough design in discussion with Chris, see SOA auth RFC

- https://www.mediawiki.org/wiki/Requests_for_comment/SOA_Authentication
- OAuth2 based, signed JWTs

work with Chris on phase 1 implementation, verify in RESTBase using JWT lib

look into existing auth service solutions like passport

- spend a day or so on a simple prototype auth service
- figure out details of longer-term plan with Chris & Tyler, incl. who is going to own this

Q2: Tooling

Deployment & packaging

- Work with ops & release engineering on a better way to deploy & package services
- Pain points:
  - lack of integration between config management and deploy system
  - takes several days to puppetize new service & work around salt & trebuchet bugs
  - duplicate effort between packages and puppetizations (prod, labs, vagrant)
- Some progress during Q1:
  - OCG and Parsoid are slowly converging (shared deploy mechanism, Jenkins jobs)

https://www.mediawiki.org/wiki/Services

#wikimedia-services

service@wikimedia.org (team)

services@lists.wikimedia.org (public)