



MySQL at Wikipedia

How we do relational data at the Wikimedia Foundation

Jaime Crespo
Percona Live Europe 2015
-Amsterdam, 23 Sep 2015-

Jaime Crespo



- Sr. Database Administrator at Wikimedia Foundation
- Used to work as a trainer for Oracle (MySQL), as a Consultant (Percona) and as a Freelance administrator (DBAHire.com)

Agenda

1. The Wikimedia Foundation	4. Reliability
2. MySQL details	5. Challenges
3. Performance & Architecture	6. Q&A

MySQL at Wikipedia

THE WIKIMEDIA FOUNDATION

Wikimedia Foundation



Some stats...

- 530-430 Million UVPM (not counting mobile devices)
- 17-20 Billion page views per month
- 14-18K new editors per month
- 35 Million Wikipedia Articles
- 8K new Wikipedia articles per day
- 27 Million open/free media files



More stats: reportcard.wmflabs.org

What makes us different

- The Wikimedia Foundation is a non profit
- Funded exclusively by donations
- These are our principles
 - Freedom and open source
 - Serving every human being
 - Transparency
 - Accountability
 - Stewardship
 - Shared power
 - Internationalism
 - Free Speech
 - Independence

https://wikimediafoundation.org/wiki/Resolution:Wikimedia_Foundation_Guiding_Principles

Openness

- Most companies are based around a proprietary technologies
- All the source code we create and use on our infrastructure is free software
 - <http://git.wikimedia.org/>
- All the configuration and provisioning infrastructure is also freely licensed
 - <http://git.wikimedia.org/tree/operations%2Fpuppet.git>



Transparency & Accountability

- All software and infrastructure changes are publicly posted*:
 - <https://gerrit.wikimedia.org/r/#/q/status:merged+project:operations/puppet,n,z>
 - https://wikitech.wikimedia.org/wiki/Server_Admin_Log
- Issue tracker is publicly accessible
 - <https://phabricator.wikimedia.org/>
- Most monitoring is publicly accessible

*except security issues (until corrected) and private information



Privacy

- Obligated to respect our users' privacy
- SSL is enforced throughout all services
- We host all our code, data and services (up to our possibilities) and do not share it with 3rd parties
 - No usage of CDNs, public clouds



No dependency

- Even companies using open source try to bind you to their service
- We provide you not only the software, but also the data dumps and the documentation to create your own fork of our projects
 - <https://dumps.wikipedia.org/>
 - <https://wikitech.wikimedia.org>
 - Except user's private data

Community Resources

- Many contributors that are not employees with production server access
- We also provide a Virtual machine (Labs) and a shared hosting platform (tools) with access to database replicas open to contributors
 - <https://wikitech.wikimedia.org/wiki/Help:Contents>
 - https://wikitech.wikimedia.org/wiki/Help:Tool_Labs



Team

- 11 people in “Technical Operations”, including 1 DBA
 - There is also Labs Ops, Datacenter Ops, Fundraising Ops, Analytics Ops, Release Engineering, Services, Devs, Performance & many volunteers supporting us
- We may not be the busiest site, but “there is literally nowhere else serving as many page views per engineer”

MySQL at Wikipedia

MYSQL DETAILS

What do we use MySQL for?

- Core relational data (users, text & file metadata, ...)
 - Regular browser requests
 - Editing API
- Reliable Key-value store:
 - Content of each page (revision)
- Disk-based caching:
 - Secondary caching level for parsed wikitext, formulas, etc.
- Analytics and events (with difficulty)
- Most internal services with database needs



What do we not use MySQL for?

(I)

- Restful API
 - Cassandra
- Crunched analytics
 - Hadoop
- Memory caching
 - Memcache
- Queueing
 - Redis



What do we not use MySQL for?

(II)

- Search and logs
 - Elasticsearch and logstash
- Compression
 - Pages use application-side compression
- File storage
 - We use Swift



<http://blog.wikimedia.org/2012/02/09/scaling-media-storage-at-wikimedia-with-swift/>

MySQL versions

- Past: Facebook 5.1 fork
- Currently finishing upgrading MySQL 5.5 to custom MariaDB 10 package
<http://blog.wikimedia.org/2013/04/22/wikipedia-adopts-mariadb/>
- Relaying on several 3rd party utilities: Percona Xtrabackup and Toolkit, mydumper, etc.

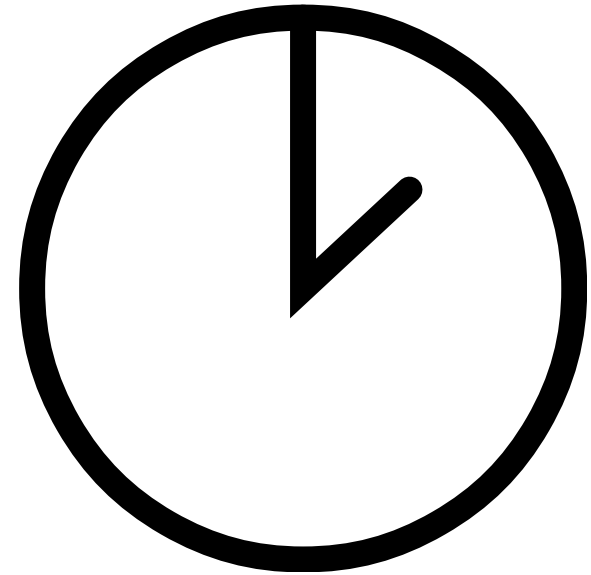


Why MariaDB?

- WMF, “corporate” contributor of the MariaDB Foundation
- In general, avoiding “lock-in” for production, but certain features are great:
 - Multi-source replication
 - TokuDB
 - Index statistics as static tables/histograms
 - Open source pool of connections
- Things we patch/would require from upstream/3rd party:
 - Query rewriting plugin
 - Delayed slave
 - Max query running time
 - Extended PRIMARY KEY issues
 - Replication state in transactional tables

Some MySQL stats

- ~22 Billion queries a day
 - Top recorded throughput for enwiki is 145K QPS
- >800 wikis in 280 languages
- 99.99% availability for enwiki in the last 6 months
- ~20TB of non-duplicate live data
- 2.5 Billion article revisions
- 95 percentile of query execution time is 332us
 - (API) queries running longer than 300s are killed



my.cnf

- <https://git.wikimedia.org/blob/operations%2FPuppet/10169911757ada824c11ee4e3dcd214bd229f247/templates%2Fmariadb%2Fproduction.my.cnf.erb>
- Particularities
 - MariaDB Pool-of-threads (max_connections = 5000)
 - charset = BINARY
 - rpl_semi_sync*
 - userstat=1
 - innodb_buffer_pool_dump_at_startup



MySQL at Wikipedia

PERFORMANCE & ARCHITECTURE

Hardware and operating systems

- Standard x86_64 servers (several providers)
- 64-192GB of RAM
- Mostly on HDs
 - Hardware RAID controller (RAID 10)
 - Currently integrating SSDs for vertical scalability
- GNU/Linux
 - Ubuntu Trusty; some machines still on Precise
 - Currently Migrating to Debian Jessie

Servers

- 1300 hosts
 - ~120 varnish caches
 - ~320 main applications servers, scalers, job runners
 - 140 active MySQL servers (including support and labs services)
 - 31 Elasticsearch servers
 - 20 LVS
 - 48 media storage frontends and backends



<http://ganglia.wikimedia.org>

Mediawiki software

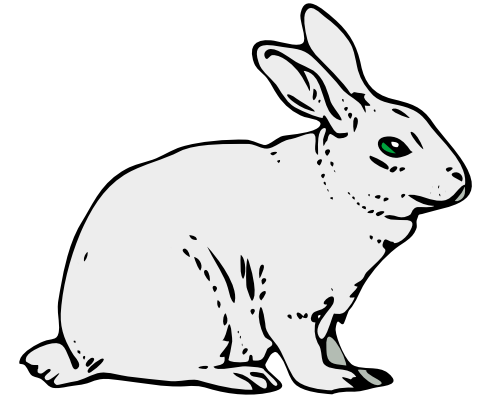
- Running on Apache with PHP-HHVM
- Mediawiki implements its own ORM that allows database independency
 - MySQL and sqlite are the main maintained engines
- Read-write is split at application side
 - Writes and important reads go to the master
 - Most reads go to the slaves
 - Chronology is checked at application side



<https://www.mediawiki.org/wiki/MediaWiki>

Caching

- Caching reads and queuing writes
 - HTTP varnish caching eliminates 9/10th of the traffic
 - Table level caching (templatelinks, externallinks) makes special pages trivial
 - Those are calculated asynchronously by redis jobs on slaves
 - HTML and unrendered wikitext is also cached and stored on memcached/parsercache db servers

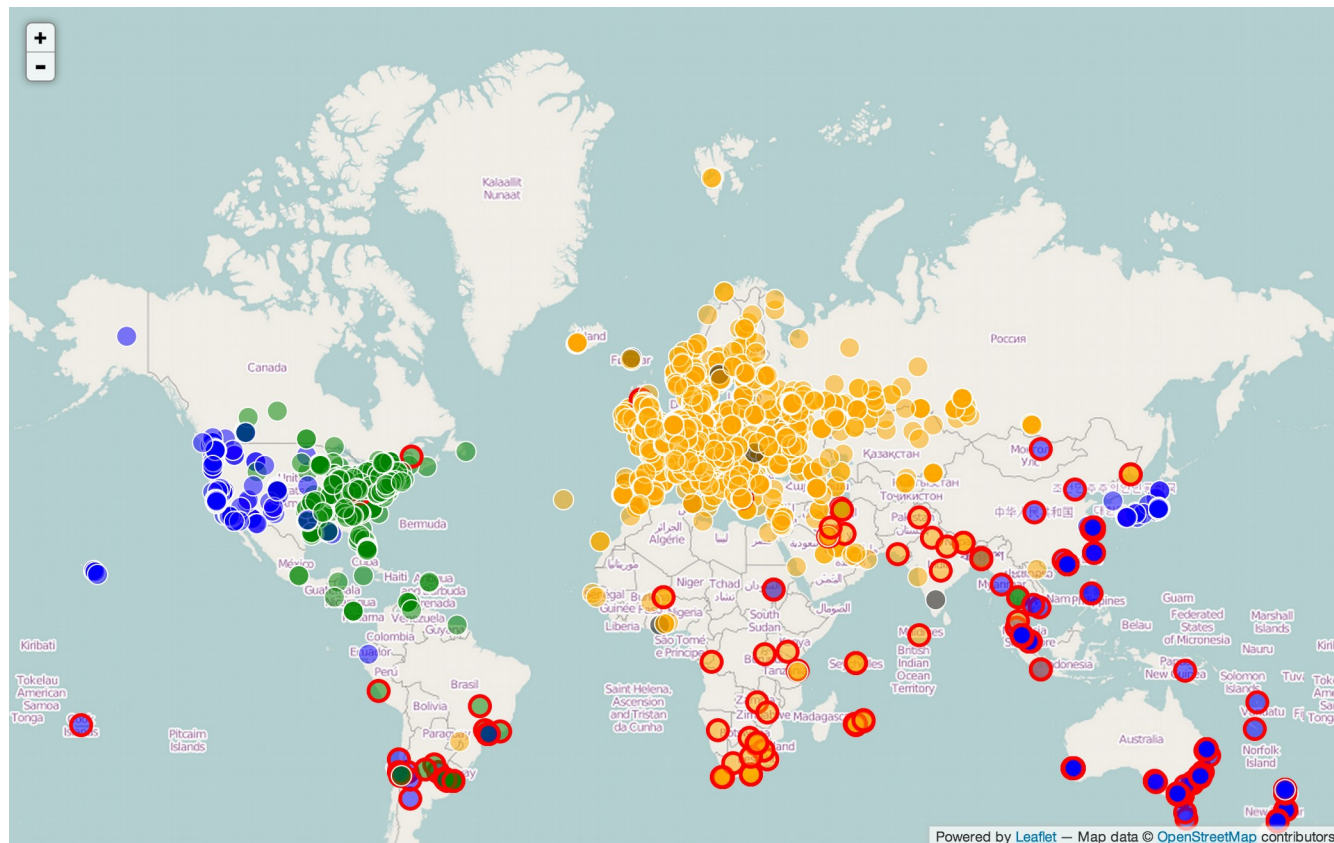


Datacenters

- Servers are distributed among 4 datacenters:
 - Ashburn, Virginia (eqiad)
 - Austin, Texas (codfw)
 - Amsterdam (esams)
 - San Francisco, California (ulsfo)
- Only active for caching (passive for application servers, for now)

<http://blog.wikimedia.org/2013/01/19/wikimedia-sites-move-to-primary-data-center-in-ashburn-virginia/>

DNS-based CDN



<http://blog.wikimedia.org/2014/07/11/making-wikimedia-sites-faster/>
<http://blog.wikimedia.org/2014/07/09/how-ripe-atlas-helped-wikipedia-users/>

MySQL Functional groups

- “Core” Production Servers
- External Storage
- External Clusters
- Miscellaneous internal services
- Parsercache
- Analytics
- Labs

MySQL Shards: Core servers

- Most relational data: users, metadata, etc.
 - s1: English Wikipedia
 - s2: Large wikis
 - s3: Most small wikis (~800)
 - s4: Commons
 - s5: Wikidata and German Wikipedia
 - s6: Large wikis
 - s7: Centralauth, metawiki and some large wikipedias

More details: <https://noc.wikimedia.org/db.php>

MySQL Shards: External Storage and External cluster

- Key-value storage where the actual revision text is
 - es1: Read-only Clusters
 - es2-es3: Read/write cluster
- x1: Very dynamic data / global data (mostly writes)
 - Notifications
 - Extension data with very different query patterns

MySQL Shards: Misc

- m1-m5: Internal services databases (puppet, phabricator, openstack, wordpress, ...)
- Parsercache (pc): secondary cache level for rendered content
- Analytics and research: MySQL replicas and event logging for data analysis and statistics
 - Make heavy use of multi-source replication for cross-shard joins

MySQL Shards: LabsDB

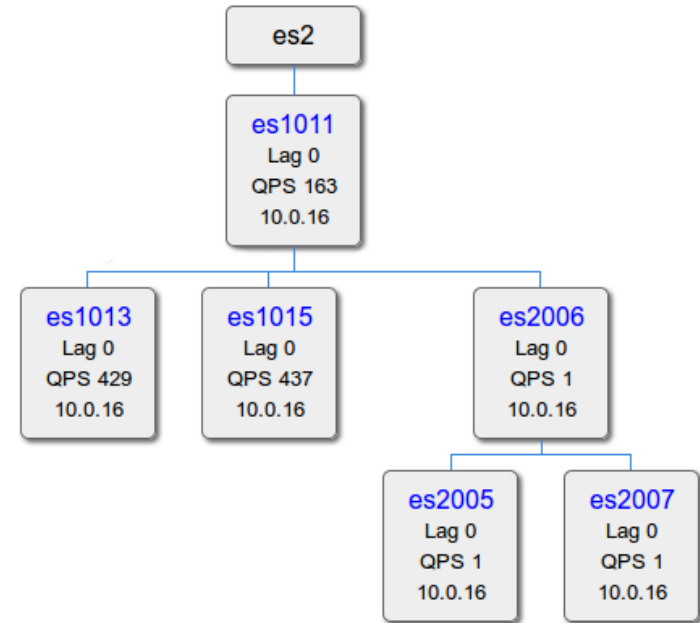
- Replicas for Virtual Machines (labs) and community contributors (tools)
- Shared mysqls (and postgresql) for tool users
- Requires sanitizing
- Challenging to administrate due to the large difference between number of users and resources available

MySQL at Wikipedia

RELIABILITY

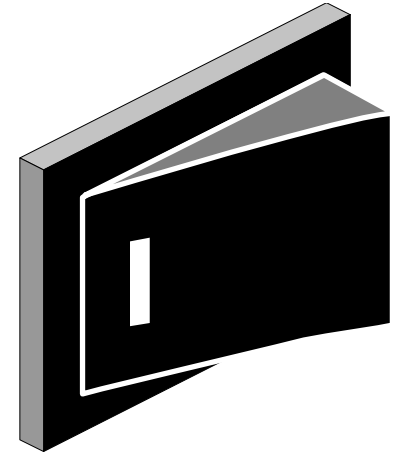
Shard components

- 1 Master
- 2-14 slaves with traditional replication
 - Geographically distributed over 2 datacenters
- Semi-sync replication to avoid data loss



Master Failover

- No automatic failover on the core servers for masters
 - Wikis will go to read-only mode if the master fails
 - An operator will perform the failover (hopefully) in less than 15 minutes
- HAProxy
 - Only used for full automatic failover for misc. services



Slave Automatic Failover

- Mediawiki-controlled
- A slave is not used if:
 - it is unresponsive
 - Its lag is larger than the configured limit (and there are other available slaves)
- Other errors (or for maintenance) require human intervention for depooling

Load-Balancing

- Also mediawiki-controlled
- Each slave as a weight (0-N)
- It can also have a role (API, slow, dump, watchlist, recentpages, contributions, logpager)
 - It helps avoiding disrupting all nodes and with buffer pool for certain query patterns
- Datacenters are active-active only for caches, applications and mysql are still active-passive

Data Recovery

- Weekly logical backups from a spare slave (6 month retention)
 - Mostly unused except for issue investigation
 - 30-day retention on binary logs
- ~Biweekly public XML dumps
- On node failure, recovery is handled by cloning from another slave (rsync or xtrabackup)
- 24-hour delayed slave with all shards (multi-source, TokuDB)



Maintenance

- No maintenance windows
 - code deployments 24/7
- No integrated system-depending on the change:
 - pt-online-schema-change/
online schema change
 - Always enough redundancy for switchover
 - Batched update



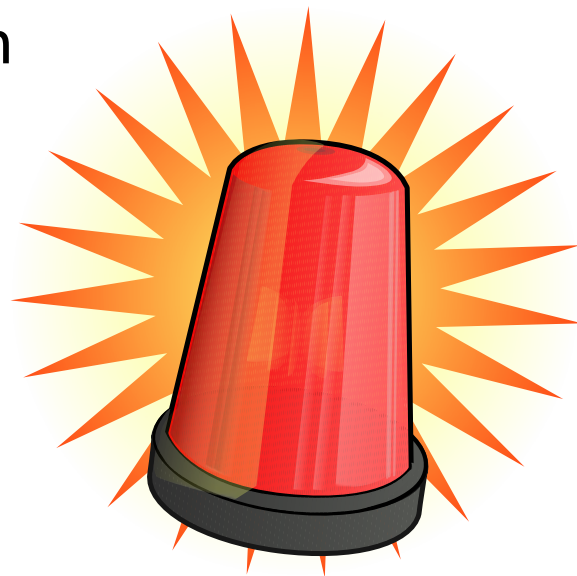
<https://wikitech.wikimedia.org/wiki/Deployments>

Lessons learned about recovery

- Avoid flopping services: STONITH
- Chaos/monkey testing (we call it deployment schedule)
- Backups are useless: have a faster recovery plan
 - Data recovery <> service recovery
- Avoid active-passive setups:
 - Avoid failover -you won't be ready when needed
 - Have redundancy and a 30% resource utilization
- Automatize and log everything (even if run manually)

Monitorization

- “Ecosystem” problem: too many of them
 - Ganglia: basic parameters
 - Icinga: alerts
 - Graphite & Graphana: custom graphs
 - Logstash: centralization of logs
 - Application db errors and slow queries
 - Custom DB monitoring system: “Tendrill”
 - Graphs, slow queries and reports
 - pt-query-digest
 - Ishmael web interface (deprecated)



MySQL at Wikipedia

CHALLENGES

Infrastructure and code

- Writes are not an issue for us -reads are
 - Logged users and POST requests are not cached
- 15 year old PHP application means technical debt
 - Dependency on statement-based replication
 - No real utf-8 support at the time
 - No sql_mode set (WIP)

Best things about MySQL

- InnoDB is reliable
- Easy to use
- Fast
- Not trying to be smart
- Wide 3rd party support (utilities)

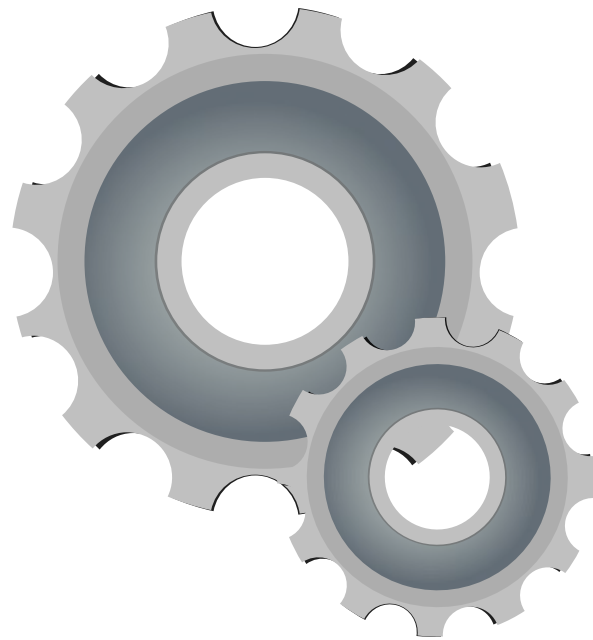


Worst things about MySQL

- Many manual operations (provisioning, replication, HA, partitioning)
 - They have to be automated by us
 - Some of them are slowly being implemented
- Lack of proper compression (both reliable and performant)

Future (I)

- SSDs and vertical scaling
- Compression (InnoDB, RocksDB, TokuDB?)
- OLAP/Column based solution for analytics
- Fully Active-Active over several datacenters
 - Multimaster?
- Better maintenance and recovery automation



Future (II)

- Integrated query analysis and debugging (P_S?)
- Better monitorization
 - Smoke tests for data integrity, strange states, etc.
- 10.1? 5.7? WebscaleSQL? Galera?
- Better sanitization process (binlog processor)
- Rearchitecture connection handling



You can help us!

- Apply for the DBA full time position:
<http://grnh.se/0y4pxm>
- Clone our puppet repo and start sending us patches
 - Or create your own wiki-based tool on Tool-Labs
- Join us at [#wikimedia-operations](#) and [#wikimedia-databases](#) at Freenode

Q&A

