LabsDBs for WMF tools and contributors: get more data, faster

Jaime Crespo

Wikimedia Developers Summit 2017

-San Francisco, 9 Jan 2017-

Agenda

- LabsDBs – what is it and how to use it?
- What's new (new servers!)
- Q&A & discussion:
  - Future of the service
  - Needs
  - Ideas
  - Problems

What are LabsDBs?

- It is a service funded by the Wikimedia Foundation and maintained (part time) by staff from Operations (DBAs) and Labs Teams + the generosity of incredible helpful volunteers
- It provides relational database services for Wikimedia Cloud and Tools users
  - The most popular service is a sanitized, real-time copy of the production MySQL Servers “replicas”
  - There are other standalone MySQL and Postgres servers

Who has access to it?

- Staff, researchers and volunteers that use Wikimedia Cloud (OpenStack on WMF)
  - Handled manually
- Users with a project on Tools
  - Access is granted automatically
- Anyone with a Wikimedia wiki account!
  - Using Quarry: http://quarry.wmflabs.org/

What is it used for?

- Some developers use it to gather info about wiki content and test its code
- Researchers use it to do its work, in addition to dumps
- Many tools that cannot do everything with the API are allowed to perform its own complex tasks. E.g. catscan, report generation, etc.
- Some bots analyze edits and perform maintenance using it (Anti-spam, broken links)

What kind of data you will find on replicas?

- All public revision and page meta information (title, timestamp, comment, user name, page properties, …)
- Public user information
- Template usage, internal and external links, image usage, interwikis (Wikidata)
- Image metadata
- For details of the schema:
  - https://www.mediawiki.org/wiki/Manual:Database_layout
  - https://phabricator.wikimedia.org/diffusion/MW/browse/master/maintenance/tables.sql

What kind of data you will NOT find on replicas?

- Private data (user IPs, passwords, emails, personal info, deleted revisions, private wikis, private user messages and notifications)
  - It is filtered out before leaving production realm
- Actual wikitext content
  - It is on a separate set of servers, too large and difficult to securely allow arbitrary access- use the API to get those
- Flow discussions
  - It is on a separate set of servers
- Things not in MySQL (jobs, cache, restbase, ...)

How to use the replica service

- When you create a new tool (remember, access is provided to tools, not users), a replica.cnf file will be generated automatically
- Use the data on it (user, password) to connect to the servers for a quick query, using an interactive client or for your application
- More on: https://wikitech.wikimedia.org/wiki/Help:Tool_Labs/Database

When NOT to use the replica service

- To perform some action/analysis for every edit:
  - Use the IRC/streaming service (it is much more efficient!)
    https://wikitech.wikimedia.org/wiki/Irc.wikimedia.org
    https://wikitech.wikimedia.org/wiki/RCStream
- To analyze large sets of subsets of data that do not require real-time:
  - Use the dumps instead- it will be faster and will not generate database overhead
- To perform queries that can be done as efficiently using the API
  - The production API is served by dozens of servers, providing a more reliable service and dataset

Don’ts (I)

- Persistent connections/pools of connections
  - Replicas are a shared environment- if everybody persisted 10 connections a few users would block everyone else from connecting.
  - Connect to the servers only when a query is needed, disconnect afterwards.
  - Disable persistent connections/pools on your programs and drivers
  - Idle connections are killed, as they take resources

Don’ts (II)

- Not reconnecting if connection fails:
  - Because connections can be killed at any moment if idle or server overloaded
  - Because the server can go down, automatically failing over to a different service
  - Try reconnecting a couple of times, then abort

Don’ts (III)

- Sending poorly optimized queries that can be done efficiently if rewritten
  - Make sure your query uses indexes
  - Make sure you do not read thousands of millions of rows in a single query if you are only going to return a few
  - If optimizations are not possible, you can break down your queries in smaller chunks

EXPLAIN on Labs

- Because of filtering methods, EXPLAIN does not work, as there is not direct access to the underlying tables
- SHOW EXPLAIN works instead:
  - https://tools.wmflabs.org/tools-info/optimizer.py
  - There are plans to make it easier to use T146483

Don’ts (IV)

- Queries running wild
  - For tools with a web interface, very rarely a user will wait more than ~5 minutes for a result to load
  - There are several ways to auto-impose execution limits:
    - `SELECT … LIMIT … ROWS EXAMINED rows_limit;`
    - `sql_select_limit`
    - `max_statement_time` (only on 10.1)

LabsDB problems

- Limited capacity: 2 5-year old servers, unreliable storage
- Data drift from production due to unsafe statements + filtering
- Different needs between webrequests and Analytics-like workload
- Hard to use, little documentation and help
  - Partially due to filtering
- Sometimes certain applications overload the server
- No automatic failover solution

New servers

- 3 new LabsDB servers:
  - Intel(R) Xeon(R) CPU E5-2637 v3 @ 3.50GHz
  - 512 GB of RAM
  - 12 TB of SSD disks
- 2 servers to filter data on production

New software

- MariaDB 10.1
- Automatic load balancing and failover with HAProxy
- Using row-based replication to guarantee data consistency
- Gotten rid of buggy TokuDB engine
  - Now using InnoDB engine (compressed), with a compressed ratio of 50%

Data reloaded & checked

- S1 and S3 fully reloaded and sanitized from production
  - Rest of the shards will follow soon
- Automatic script to detect private data
- Much, much easier to reload a shard if data gets lost or corrupted

Future

- Pre-generated and trigger-based summary tables (like watchlist_count)?
- More/different indexes/query plans than production
- How to setup resource limits?
  - API to schedule & serialize queries?
- User databases?