Database Meeting 2015-07-15

Date

15 Jul 2015

Attendees

- Brian Van Klaveren, Vaikunth Thukral, Unknown User (npease), Andy Salnikov, John Gates, Fabrice Jammes, Serge Monkewitz, Andy Hanushevsky, Jacek Becla

Discussion items

DM-3124

- unsure how to pass none value to lua via twisted, possibly a string with word 'None'. Fabrice will investigate

Monitoring performance

- Consider using ELK for understanding performance of integration tests

"interactive", "non-interactive" queries

- need better name. Ideas tossed around:
    - interactive / batch
    - streaming / batch
    - synchronous / asynchronous
    - synchronous / batch
    - short / long
    - short / background
    - immediate / background
    - immediate / batch
    - prompt / deferred
- observations raised:
    - "interactive" implies that user is typing something. Often not the case
    - Batch implies a group of jobs
    - sync/async too cryptic for non computer scientist
    - long/short - users might think that it is only related to timing, but it is not
    - "immediate" raises expectations too much
- will discuss at the next DataAccess meeting. (Unknown User (xiuqin), you might want to have a peek to get a preview)

Notifications

- want to give user an option to decide how she wants to be notified about status change of non-interactive query. Options:
    - email
    - host/port that accepts http post requests
- user might also want periodic updates, eg with % complete info
- this is not part of query metadata work, we will handle it later

DM-2805

- common user case: user submits query, gets queryId, and periodically queries for results, without ever checking status. This advocates for having everything in results.
- So, we must store complete error information stored as part of results
- Thus in metadata, keep just query status (integer/enum - running/completed/failed/aborted)
- look into long running queries soon (eg, next cycle)

releasing DR data into production

- keep each DR release separate, don't mix
- LHC uses a simple table where they define which machines are production at any given time
- If we adopted that model, we would just need to make the update in such table and the cluster that was used to produce DR data would become a production cluster
- In our spreadsheet baseline, we did not enforce different releases on different sets of machines. This might have some (not very big probably) implications on cost. This needs discussions with Kian-Tat Lim and Donald Petravick.

scalability bug

- understood and solved, need to release the official patch. It was related to timing in xrootd, under heavy load data was written to a wrong buffer (and we were receiving a valid, but empty buffer on qserv side for some chunks). There was no memory corruption involved which made its so hard to debug
- next: see if there are any more problems

CSS v2

- planning to switch away from zookeeper, and use mysql instead. Reasons:
    - headaches with C++ bindings to zookeeper, forced us to use python in hacky way when loading data from zookeeper
    - we are less thrilled with watchers than we originally were (watcher can miss sending notifications, essentially making it much less useful for us)
    - we have some (not really proven) reservations about write/update performance, though some official zookeeper benchmarks claim 20K updates per sec
    - we are using zookeeper less than we were originally planning (query metadata info is in mysql, data distribution info will most likely be in distributed hash table), and we have mysql system for this sort of things anyway (for query metadata), thus simplifying the system and removing some moving parts like zookeeper is good
- Nate will be working on it
    - adding support for updates to CSS
    - implementing locking
    - unifying c++/python APIs (kv interface) - we want the C++ to be the master API, and expose it to python layer

SqlAlchemy

- we almost had a quorum, but we need Kian-Tat Lim for this discussion

Ingest

- Serge working on the ingest. Will have it ready early next week
- not doing anything fancy with mysql. For now, use direct mysql bindings, will switch to db module (or sqlalchemy depending what we decide) later