Making MySQL-8.0 XA transaction processing crash safe

zettadb.com

David Zhao
Zhao Wei (David Zhao) twitter/linkedin/wechat: david.zhao.cn@gmail.com

Database kernel developer in Oracle 2007-2015
- Berkeley DB
- MySQL

Database kernel developer in Tencent 2015-2019
- TDSQL --- most popular distributed RDBMS inside Tencent and Tencent Public/Private Cloud
  - Lead the effort to evolve TDSQL from a table-sharding solution to a distributed RDBMS

Left Tencent and started Kunlun project in Aug 2019
- To build an epochal distributed RDBMS based on widely adopted technologies(MySQL & PostgreSQL).
- For more info about kunlun project visit www.zettadb.com
- Kunlun project is open source at https://github.com/zettadb/kunlun
About Kunlun Distributed RDBMS

- **Goals & objectives**
  - A distributed RDBMS for PostgreSQL and MySQL users
    - with knowledge & lessons learned from work on TDSQL
    - compatible with PostgreSQL in DML grammar and most DDL grammar
    - compatible with MySQL in client protocol and DML grammar
    - to manage multi-terabyte data for both OLTP and inplace OLAP workloads
  - Full-fledged NewSQL capabilities
    - highly available & fully consistent, crash safe & fault tolerant
    - highly scalable: table sharding & elastic horizontal scale-out
  - Cloud native & DBaaS

- **Stands on the shoulders of giants**
  - Kunlun computing node derived from PostgreSQL-11.5
  - Kunlun storage nodes derived from Percona-MySQL-8.0
  - keeps up with upper streams
  - Finished over 70% kernel development, released 0.7 in Sep. 2020
Kunlun Architecture

Global Metadata Replication (GMR)
Global Transaction Processing (GTP)
Elastic Scale Out (ESO)
Intelligent Adaptive Partitioning (IAP)

Kunlun Distributed DBMS Cluster

Blockchain Nodes (CN1, CN2, CN3, CN4)
Storage Shards (SS1, SS2, SS3, SS4)
Primary
Replica
Replica

Kunlun Metadata Cluster

SCBR

Kunlun Cluster_mgr

Globally Consistent Backup&Restore (GCBR)
Globally Parallel Query Processing (GPQP)
JIT Query Compiling (JQC)
Intelligent Query Optimization (IQO)
Kunlun Advantages & highlights

- Share nothing & adaptive sharding (IAP)
  - distribute tables/tablets intelligently to multiple storage shards
  - multiple nodes for read & write queries, premium scalability
- Distributed transaction mgmt & distributed query processing
  - works transparently, no burden for users, no client code changes needed
  - appears like a standalone DB server for apps and app developers
- Crash safety & fault tolerance, high availability & strong consistency (GTP)
  - any node/network fault at any time won’t break transaction ACID guarantees
- Elastic horizontal scale-out (ESO) (*)
  - Automatic & elastic & continuous & on demand
  - unnoticible by application or end users
- Full-fledged distributed query processing (IQO) (on-going)
  - cross shard join, subqueries, prepared statement, jit
  - OLAP analysis, stored procedures and more
- Global parallel query processing (GPQP)
  - Inside computing nodes (CN)
  - CN send queries to target storage nodes (SN) asyncly so SS work in parallel
  - Inside a SN (*)
Kunlun Basics

- Computing nodes
  - accept & validate user connections
  - accept & process user queries
    - parse -> optimize -> execute(send SQL -> receive & assemble)
    - executes DDLs and DMLs
  - doesn't store user data, only store metadata locally
    - takes trivial storage/memory space
    - stores user data in storage shards
    - can be built from metadata in metadata shard, equivalent to stateless proxies
      - needs no HA measures, no burden for DBAs
  - add/remove nodes on demand, nodes independent from each other

- Support most PostgreSQL query processing features
  - Most DDLs and DMLs grammars
  - indexes, views, materialized views, sequences; prepared statement, jit
  - cross shard join, subqueries, OLAP queries, stored procedures (on-going)
  - All basic data types (numeric, string/text, date/time/timestamp, enum), json&spatial in future
  - Will support mysql client protocol and common MySQL private DML grammar(pending)
Kunlun Basics

- **Storage shards**
  - Stores application(user) data in standalone tables
    - PG single tables
    - PG table partitions
  - execute mostly single table queries
    - in a global transaction's local transaction branch
    - no table partitioning
  - Currently uses MySQL group replication(MGR) single primary mode for shard HA
    - primary election
    - robust consistency guarantees
  - Will support more types of HA (*) solutions
    - strongly consistent row based binlog replication
    - shared storage
  - Uses innodb only so far
    - never untransactional engines(myisam, etc)
    - maybe myrocks in future
  - Require kunlun-percona-MySQL-8.0.18-9
    - developed based on percona-MySQL-8.0.18-9
    - contains critical bug fixes & supporting features & performance enhancements
    - will keep up with upper stream
Kunlun Distributed Transaction Processing

- Global Transaction Coordinator
  - component of a computing node
  - Tracks txn branch states
  - 2PC for multi-shard writers
  - 1PC for single shard writers
  - 1PC for read-onlytxn branches
  - before 2nd phase, group log commit decisions in commit-log
  - Abort txn on error during txn commit
Potential Failures

- Node failures
  - Computing nodes (CNs)
  - Primary nodes of storage shards
  - Replica nodes of storage shards
  - Primary nodes of metadata shards
- Network failures between
  - CNs & Meta shard primary
  - CNs & Storage shard primary
  - A primary MySQL node and its replicas
- Fail while many global txns are in 2PC commit phases
  - partial commits
- Fail during binlog group commit
  - many regular txns doing internal 2PC (prepare->commit)
  - many XA txns doing prepare
  - many prepared XA txns doing commit
Requirement for crash-safety & consistency in MySQL

- Keep consistency between
  - a primary and its replicas
    - replica local sync
    - same binlog event groups
  - innodb and binlog
    - for any txn T, T in innodb <=> T in binlog
    - key: no extra event groups in binlog
    - challenge: binlog is passive and inflexible
  - gtid set in innodb undo log, binlog and mysql.gtid_executed (MGE)
    - New requirement for MySQL-8.0 for clone
    - txn.gtid -> undo log -> mysql.gtid_executed (MGE)
MySQL Binlog Group Commit (BGC)
MySQL-8.0 XA Transaction Processing Issues

- XA Prepare handling in community mysql
  - step 1: binlog prepare GT
    - binlog group commit flush&sync phases
    - wait for its reception by replicas before flush(MGR) or after sync (semisync)
  - step 2: engine(innodb, myrocks, etc) prepare GT

- Issue 1: primary may crash between steps #1 and #2
  - Some replicas may have replicated binlogs of GT and replayed it, so GT is prepared in replica(s)
  - Crashed primary node has inconsistent binlogs&innodb
  - after primary switch, old primary(now replica) can’t do ‘XA COMMIT GT‘, replication blocks

- Solution: do binlog prepare after all engine prepare, so the new procedure is:
  - step 1: engine(innodb, myrocks, etc) prepare GT
  - step 2: binlog prepare GT
  - the solution causes issue 2
- XA Prepare Issue 2: dilemma caused by MySQL-8.0 clone
  - no gtid at step #1, only has gtid at beginning of step #2
- Solution: interleave operations
  - engine prepare -> generate gtid -> write gtid to undo log -> flush binlog
  - innodb recovery: abort prepared txns without gtid in its undo log
  - gtid -> gtid_persistor -> MGE after binlog sync
Binlog Recovery
- engine recovery first
- scans last binlog file to form internal XA txn ids $S$
  - why only last binlog?
- commit/abort each recovered prepared innodb txn if it is/isn't in $S$

Issue 3: external XA txns are ignored in binlog recovery
Solution: for each prepared external XA txn $x_i$ in every storage engine
- commit $x_i$ if 'XA COMMIT $x_i$' is found in last binlog file
- otherwise abort $x_i$ if 'XA ROLLBACK $x_i$' is found in last binlog file
- otherwise leave $x_i$ prepared if 'XA PREAPRE $x_i$' is found in $S$
- otherwise abort $x_i$ if 'XA PREPARE $x_i$' isn't found in $S$
- special handling for 'XA COMMIT ONE PHASE'
MySQL-8.0 XA Transaction Processing Issues

- Where are prepared txns?
  - prepared XA txns can live across binlog file rotations
  - **Issue 4**: not sufficient to scan last binlog file for prepared external XA txns
    - XA PREPARE & XA COMMIT seperated apart
- Solution: record all current prepared external XA txns at creation of each new binlog file
  - kunlun-percona: pack into Previous_gtids event
  - loss: ecosystem compatibility
  - gains: distributed txn crash safety
mysql> xa start 'xa2';
Query OK, 0 rows affected (0.00 sec)

mysql> insert into t1 values(7,8);
ERROR 1044 (3D005): No database selected
mysql> use test
Database changed
mysql> insert into t1 values(7,8);
Query OK, 1 row affected (0.01 sec)

mysql> xa end 'xa2';
Query OK, 0 rows affected (0.01 sec)

mysql> xa prepare 'xa2';
Query OK, 0 rows affected (0.01 sec)

mysql> xa start 'xa3';
Query OK, 0 rows affected (0.01 sec)

mysql> insert into t1 values(9,10);
Query OK, 1 row affected (0.00 sec)

mysql> xa end 'xa3';
Query OK, 0 rows affected (0.00 sec)

mysql> xa prepare 'xa3';
Query OK, 0 rows affected (0.00 sec)
Thank you!