ix Replication Lag with Parallel Replica Threads (Doc ID 2491599.1) To Bottom

In this Document

Applies to:

MySQL Server - Version 5.7 and later
HeatWave - Version N/A to N/A
Information in this document applies to any platform.
Disclaimer: Starting from v8.0, some terms have been deprecated. Older releases will only use the deprecated terminology, and new releases will only use new terminology. Please see https://dev.mysql.com/doc/mysqld-version-reference/en/ for a complete list of those changes, and in which minor versions it happened.

Symptoms

When having a replication topology, replication lag could be present depending on many factors.

Changes

MySQL 5.5 has single threaded replica binlog applier and this often resulted in a bottleneck and caused replication lag to increase.

MySQL 5.6 introduced the DATABASE parallel replication scheme, which considers that any transactions that work on different databases can be executed in parallel on the replica applier.

MySQL 5.7 introduced the LOGICAL_CLOCK parallel replication scheme, extending parallelism to transactions within the same database. Initially it considered that only transactions that committed together on the same binary log group commit could be executed together on the replica, but then that was improved upon to work on intervals of dependencies, reducing the impact of serialization points that were seen at group commit boundaries.

MySQL 5.7.14 introduces transaction_write_set_extraction with XXHASH64 along with;
MySQL 5.7.22 introduced binlog_transaction_dependency_tracking=WRITESET dependency tracking for further parallelization.

MySQL 8.0 (and MySQL Database Service) defaults to transaction_write_set_extraction=XXHASH64 and binlog_transaction_dependency_tracking=WRITESET

Cause

High concurrency on the source server and storage latency can contribute to replication lag, even when using one of the newer features such as LOGICAL_CLOCK.

Newer architecture have multiple CPU's that are not being utilized efficiently enough on the replica to allow better paralellism.

Factors such as configuration, shape/size of compute resources and application workload all need to be checked and considered.

Solution

Check Source and Relay Configuration

Check the source, and any relay replication servers (eg in a A->B->C setup). Collect from each involved MySQL instance:

```
SELECT NOW(), @@hostname;
SELECT * from performance_schema.global_variables WHERE VARIABLE_NAME IN ('binlog_transaction_dependency_tracking','binlog_transaction_dependency_history_size','transaction_write_set_extraction','binlog_row_image','binlog_row_value_options','binlog_transaction_compression','replica_parallel_workers');
```

The source binary log applier can be configured to direct the replica in how to best make the replica parallel threads apply the data. In an A->B->C scenario, you should add configuration to both A and B to ensure all replicas take advantage.

The interface for choosing the source of parallelization is implemented as a new server system variable *binlog_transaction_dependency_tracking* which can take any one of the values COMMIT_ORDER, WRITESET, or WRITESET_SESSION. COMMIT_ORDER (the default in 5.7) causes parallelization information to be logged using commit timestamps; WRITESET (default in 8.0) causes this information to be logged using write sets in such a way that any transactions not updating the same row can be parallelized; and WRITESET_SESSION acts in the same fashion as WRITESET, except that updates originating with the same session cannot be reordered. The size of the row hash history that is kept in memory for tracking transaction dependencies can be set using *binlog_transaction_dependency_history_size* option.

Don't Set binlog_format=MIXED And binlog_transaction_dependency_tracking=WRITESET(_SESSION) (Doc ID 2982715.1)

Adjusting configuration at runtime

If you adjust a replicas configuration at runtime, and wish to ensure new settings are taking effect (eg for other replicas in a A->B->C scenario), you will need to issue a STOP/START of the replication threads.

https://dev.mysql.com/doc/refman/5.7/en/stop-slave.html
https://dev.mysql.com /doc/refman/5.7/en/start-slave.html

For MySQL 5.7 or 8.0 source servers:

```
SET GLOBAL transaction_write_set_extraction = XXHASH64;
SET GLOBAL binlog_transaction_dependency_tracking = WRITESET;
```

For MySQL 5.7 replica or relay servers:

```
STOP SLAVE;
SET GLOBAL slave_parallel_type = LOGICAL_CLOCK;
SET GLOBAL slave_parallel_workers = 4;
SET GLOBAL transaction_write_set_extraction = XXHASH64;
SET GLOBAL binlog_transaction_dependency_tracking = WRITESET;
START SLAVE;
```

For MySQL 8.0 replica or relay servers:

```
STOP REPLICA;
SET GLOBAL replica_parallel_type = LOGICAL_CLOCK;
SET GLOBAL replica_parallel_workers = 4;
SET GLOBAL transaction_write_set_extraction = XXHASH64;
SET GLOBAL binlog_transaction_dependency_tracking = WRITESET;
START REPLICA;
```

You will also need to add these values in the configuration file to make these changes permanent.

Note that enabling the transaction dependency tracking will not take effect until any existing binary logs or relay logs are processed.

Check Resource Consumption of the replica

CPU

Check that CPU is not a factor in the slowdown of replication processing.

DiskIO

Check that DiskIO operations/volume performance utilization is not a factor in any slowdown of the replica.

Check replica hardware shape/size

Replicas most often need to at least match the size of the source, and sometimes exceeding the shape/size is required to keep pace.

Check replication worker distribution history

Historical worker distribution can be indicative of a workload that cannot be parallelized. Please check the source/relay configurations in *'Check Source and Relay Configuration'* first to ensure best practice for parallelizing workloads.

Check for primary keys

```
SELECT t.table_schema, t.table_name,t.engine
FROM information_schema.tables t
LEFT JOIN (select table_schema, table_name, constraint_name from information_schema.key_column_usage UNION ALL select 1, 1, 1) k
ON t.table_schema = k.table_schema
AND t.table_name = k.table_name
AND k.constraint_name = 'PRIMARY'
WHERE
t.table_schema NOT IN ('mysql', 'information_schema', 'performance_schema', 'SYS')
AND t.table_type = 'BASE TABLE'
AND k.constraint_name IS NULL;
```

References

https://mysqlhighavailability.com/improving-the-parallel-applier-with-writeset-based-dependency-tracking/
https://dev.mysql.com/doc/refman/5.7/en/replication-options-binary-log.html#sysvar_binlog_transaction_dependency_tracking
NOTE:2982715.1 - Don't Set binlog_format=MIXED And binlog_transaction_dependency_tracking=WRITESET(_SESSION)