ORACLE

Modernizing MySQL Replication for Higher Performance

Directions for moving replication performance forward

Karolina Szczepankiewicz
Senior Software Developer
MySQL Core - Replication
May 26, 2026

Safe harbor statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Replication bottlenecks

Concrete directions should be validated against production bottlenecks

Replication performance problems show up in different parts of the pipeline:

- Source commit coordination
- Network transfer and relay log I/O
- Replica scheduling and apply throughput
- Recovery, catch-up, and failover time
- CDC, filtering, and downstream consumers

Which bottlenecks dominate in your production workloads?

Groundwork already in motion

Modernizing internals reduces the cost and risk of future performance work.

These are not the full replication roadmap, but they are examples of completed work that makes the system easier to evolve.

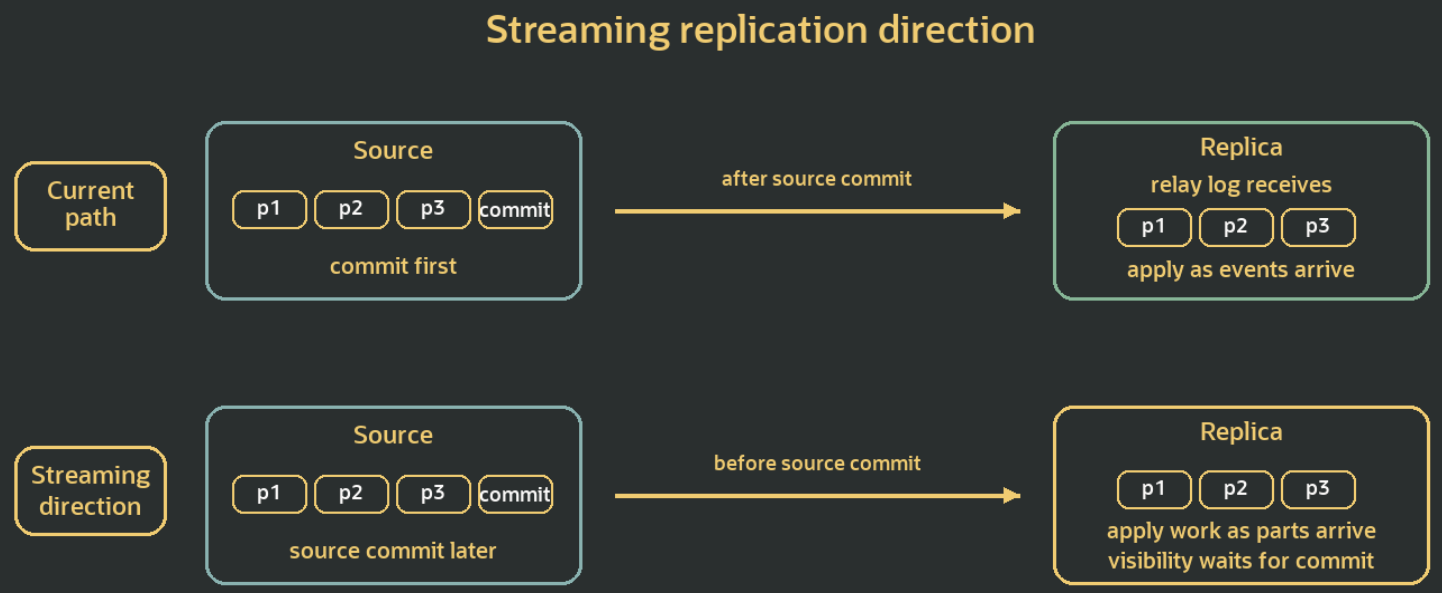
- Serialization/deserialization automation for faster replication wire changes
- Deprecation and removal work to keep only needed features
- GTID infrastructure redesign
- Replication libraries to increase component cohesion
- New Applier : Change Stream Applier available for evaluation in MYSQL Labs

New replication foundations expand the design space, but workload shape decides the performance outcome.

Streaming replication

Move changes through the pipeline earlier to reduce end-to-end latency.

Instead of waiting for a full transaction to commit before transferring work, streaming replication can send transaction parts earlier so the replica can start apply work sooner while preserving commit visibility semantics.



Which workloads would benefit most from earlier transaction delivery?

Transaction shape and replica apply

Small transactions and large transactions stress the applier in different ways.

Many small transactions

coordination overhead, metadata updates, commit cost

Very large transactions

lag, reduced parallelism, long apply units

Batching or reshaping work may help, but only if useful parallelism is not collapsed.

Binlog Group Commit enhancements

Align the binary log group with transaction coordination, engine commit, and durability work.

The source already forms meaningful commit groups in the binary log pipeline. Better alignment across the commit path can reduce repeated coordination cost and improve throughput under high concurrency.

- Use the binary log group as the natural unit of coordination
- Reduce repeated work across commit layers
- Better align transaction coordinator, storage engine commit, and durability
- Improve source-side throughput without changing transaction semantics
- Consider primitives that may also help replica-side execution

The replication-level goal is to reduce repeated coordination cost without changing transaction semantics.

Opt-in relay log behavior

Allow received events to move toward the applier without always waiting for RL durability

Some deployments could reduce latency and disk I/O by letting received events flow directly toward the applier, with relay log durability behavior made explicit and opt-in.

- Traditional path uses the relay log as a durable handoff
- Durability is important, but it adds disk I/O and latency
- Opt-in behavior can move events from receiver to applier earlier
- Existing conservative behavior should remain available

This must be explicit and opt-in because relay log behavior affects operational guarantees.

Structured change streams

Make replication changes easier to consume without duplicating raw binlog parsing logic.

A structured change-stream representation can make replication events easier to evolve, filter, and consume by replicas, CDC tools, migration systems, audit pipelines, and other downstream consumers.

- Provide a stable event representation above raw binlog internals
- Support schema-based formats such as protobuf or similar encodings
- Use generated serialization/deserialization for safer evolution
- Add metadata that helps filtering, auditing, and troubleshooting
- Avoid overhead for workloads that do not need extra structure

The binary log must remain efficient, and workloads that do not need extra structure should not pay for it.

Share the problem shape and be open to contributing the feature, not only the feature request.