MySQL 5.7 レプリケーション新機能

日本オラクル株式会社
MySQL Global Business Unit

Safe Harbor Statement

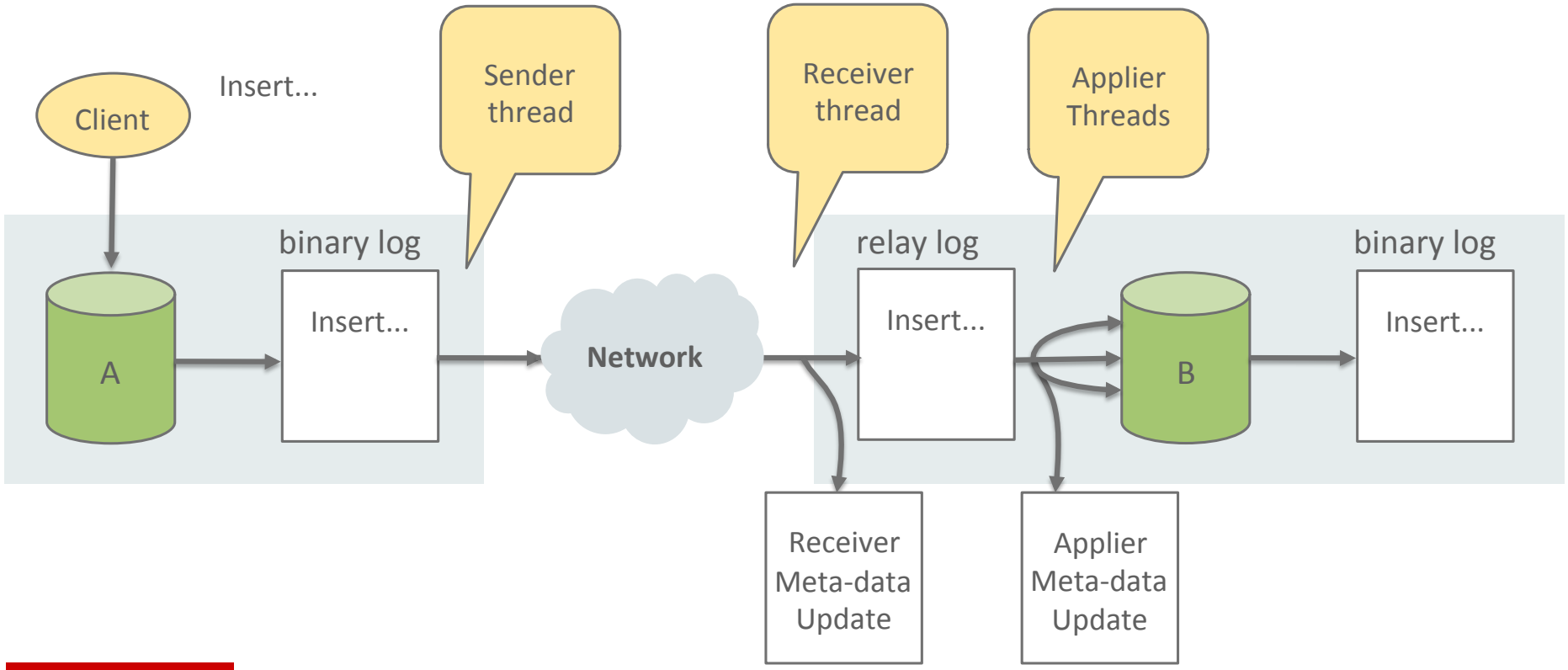
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development,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

Program Agenda

1. MySQL レプリケーション概要
2. MySQL 5.7 レプリケーション新機能
3. MySQL Fabric
4. MySQL Router GA
5. Labsで開発中の機能
6. ロードマップ
7. まとめ

MySQL レプリケーション概要

概要: レプリケーションアーキテクチャ



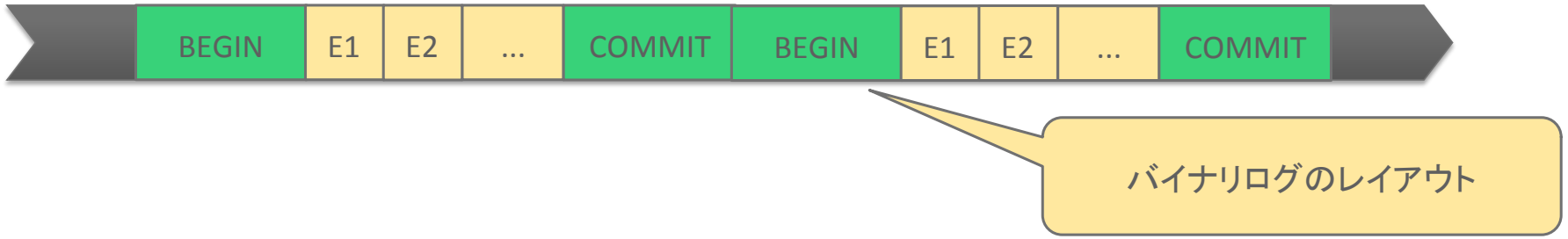
概要: レプリケーションアーキテクチャ

- 各スレッドの役割

スレッド名	本資料での名称	役割
Binlogダンプスレッド	Senderスレッド	バイナリログの内容をスレーブに送信
ACK Receiverスレッド	ACK Receiverスレッド	スレーブからのACKを受信
スレーブI/Oスレッド	Receiverスレッド	バイナリログの内容を受信しリレーログに記録
スレーブSQLスレッド	Applierスレッド	リレーログの内容をデータに反映

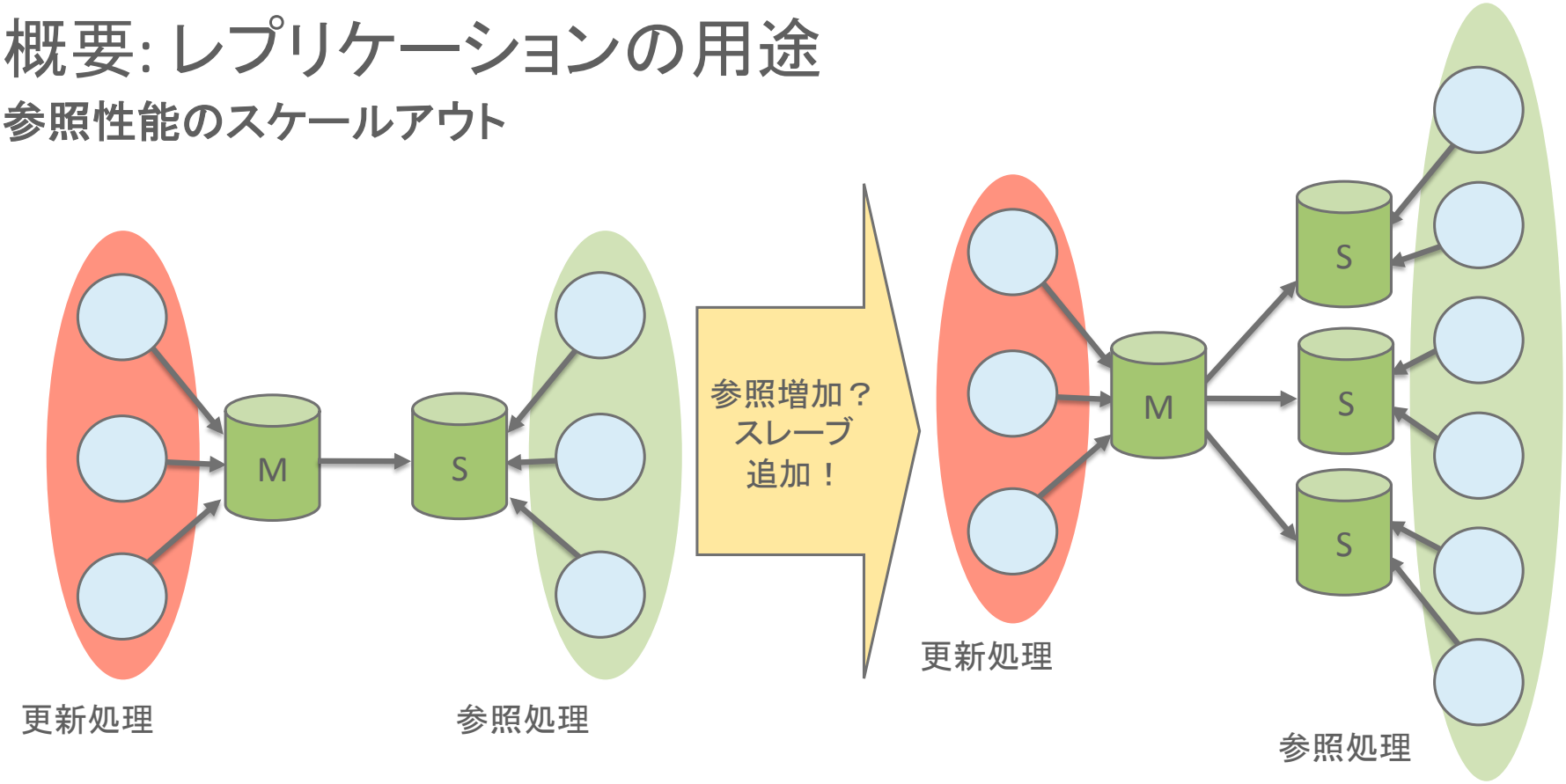
概要: レプリケーションアーキテクチャ

- **バイナリログ**
  - マスターでの変更点を記録するログファイル
  - 文(STATEMENT)ベースまたは行(ROW)ベース、またはMIXED
  - 各トランザクションはイベントのグループに分割される
  - 主な操作: ローテーション、内容の表示、GITD, ...

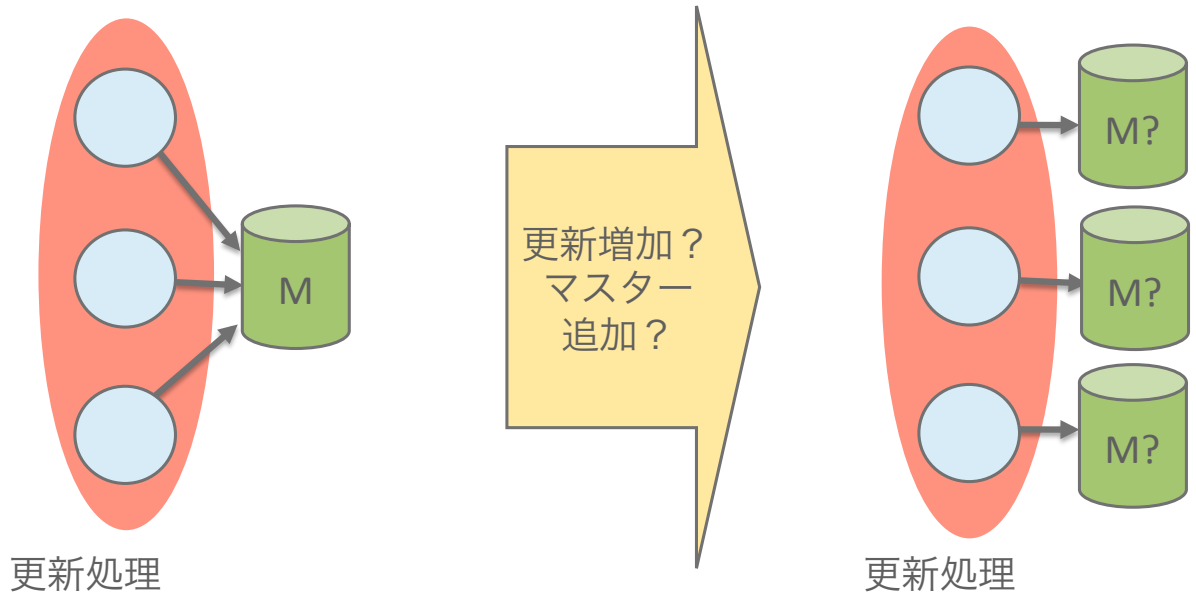


概要: レプリケーションの用途

参照性能のスケールアウト



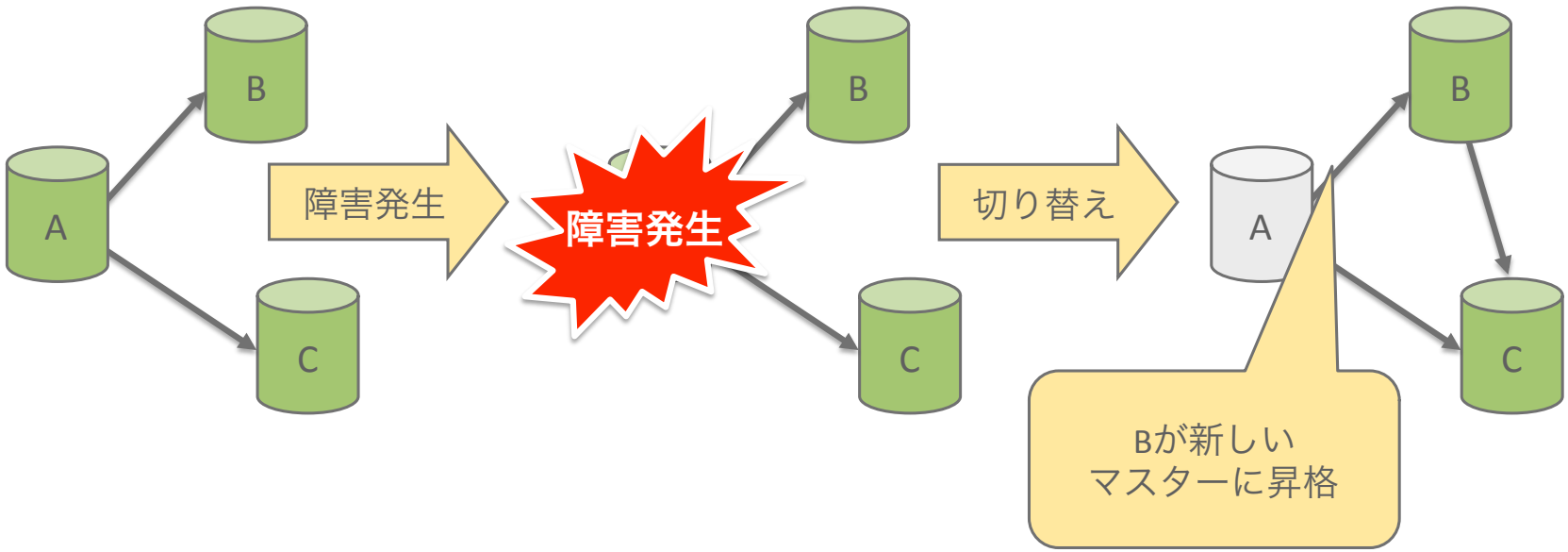
更新処理のスケールアウトは？



MySQL Fabric によるシャーディング構成...

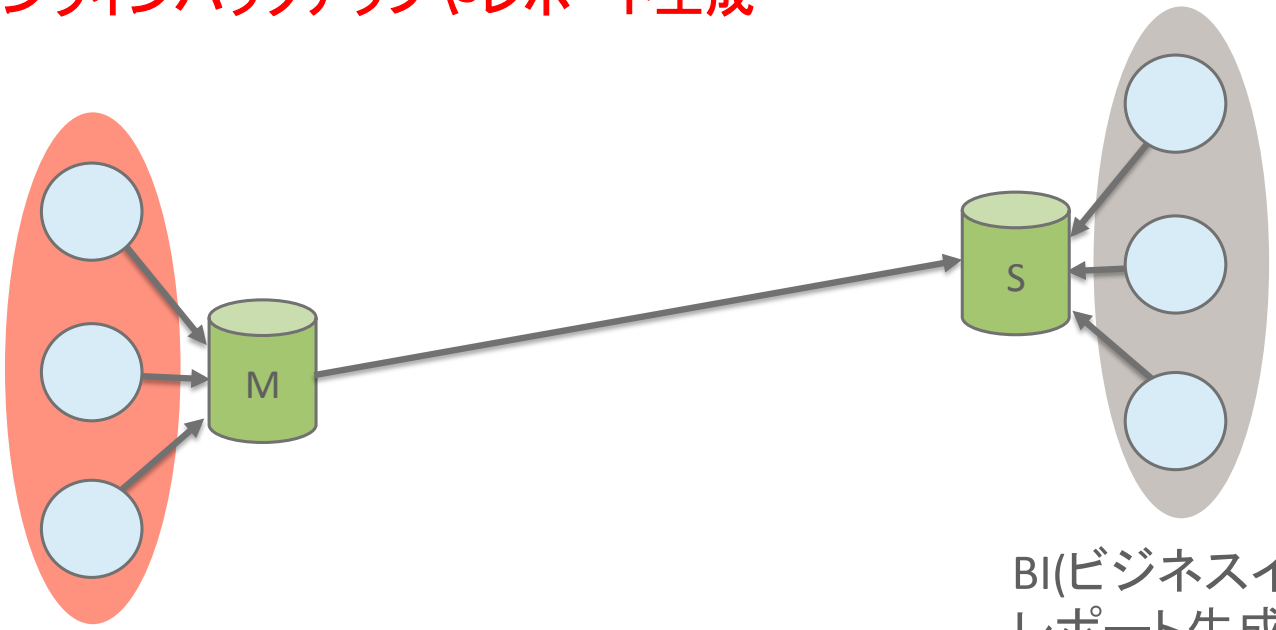
概要: レプリケーションの用途

耐障害性: マスターの障害時に、スレーブをマスターに昇格



概要: レプリケーションの用途

オンラインバックアップやレポート生成



更新処理

BI(ビジネスインテリジェンス)や
レポート生成アプリ、
データ分析アプリなど

概要:レプリケーションの用途

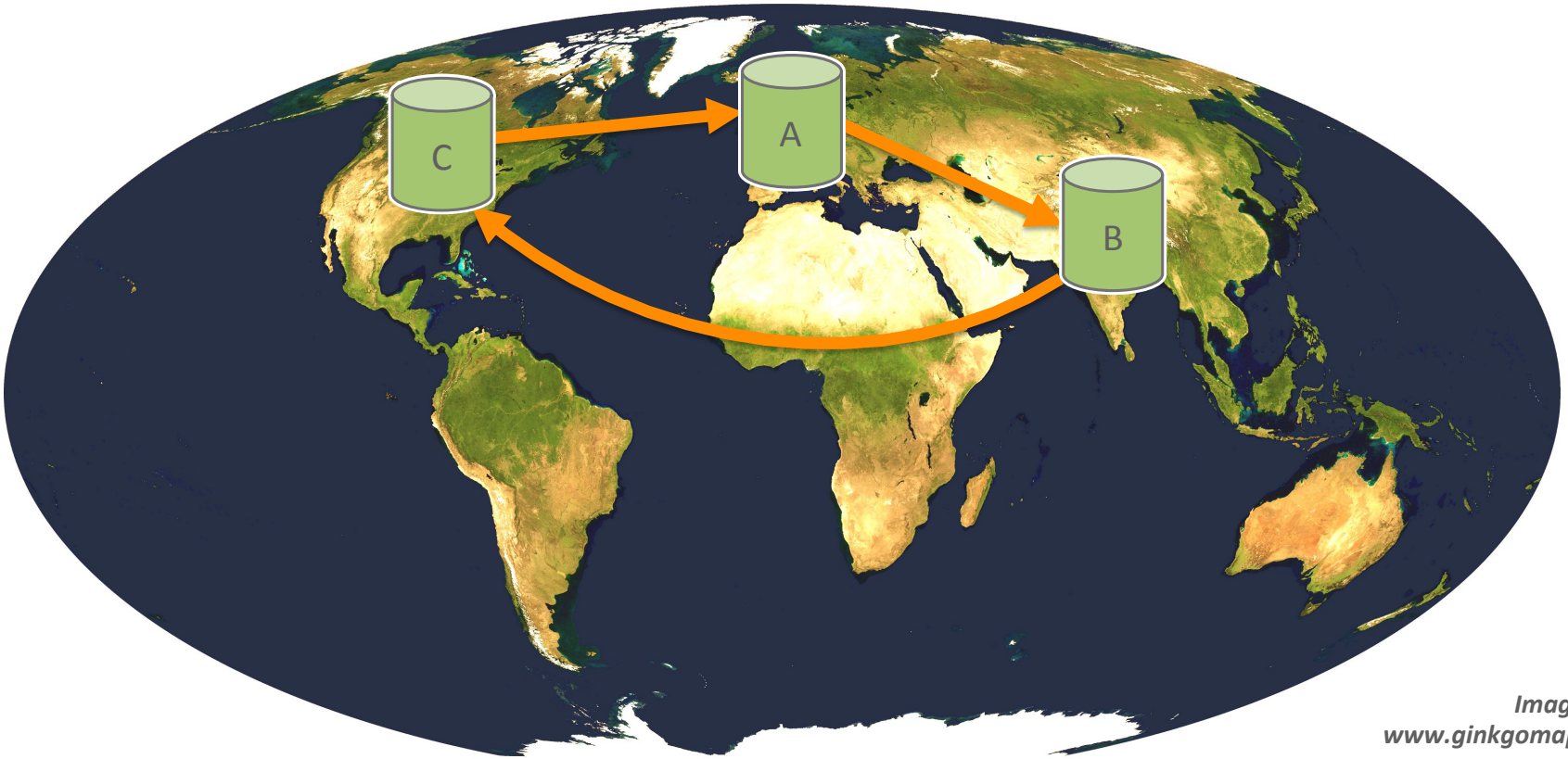


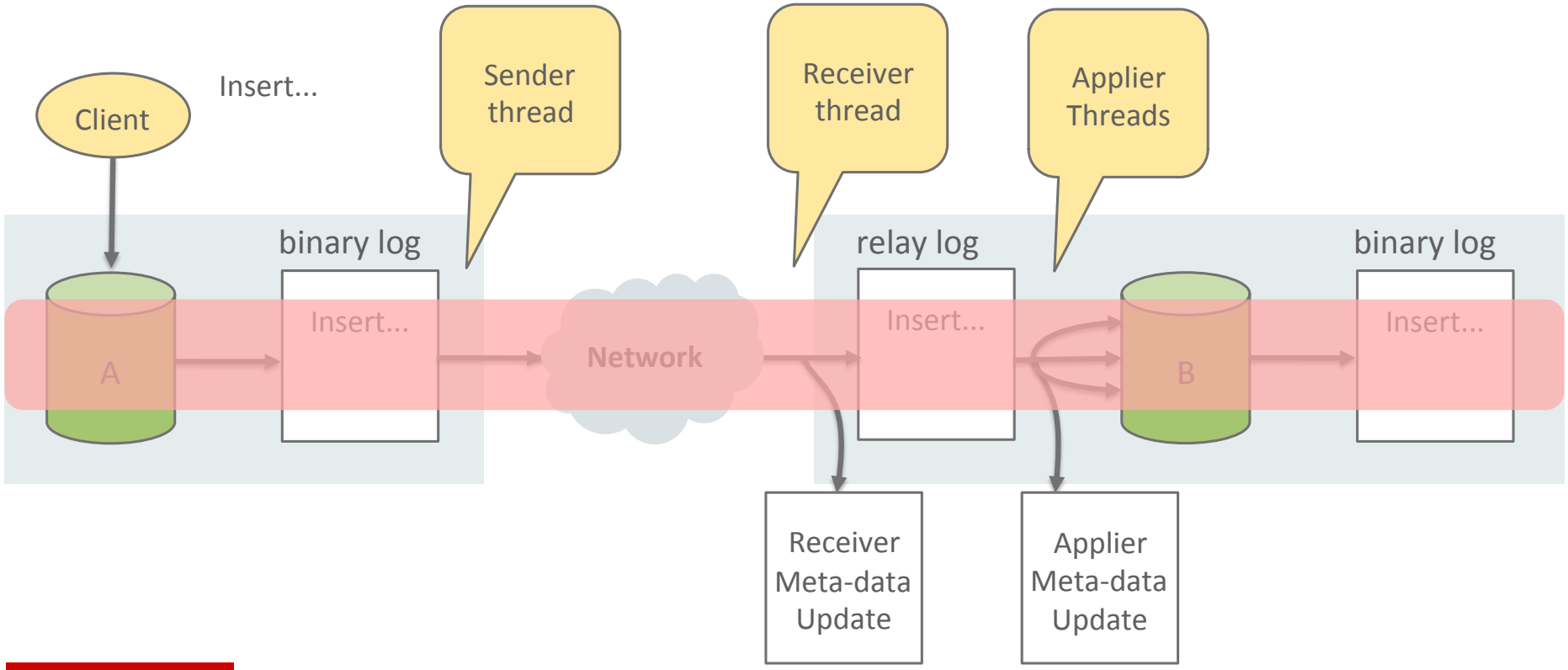
Image from
www.ginkgomaps.com

MySQL 5.7 レプリケーション新機能

MySQL 5.7 レプリケーション新機能

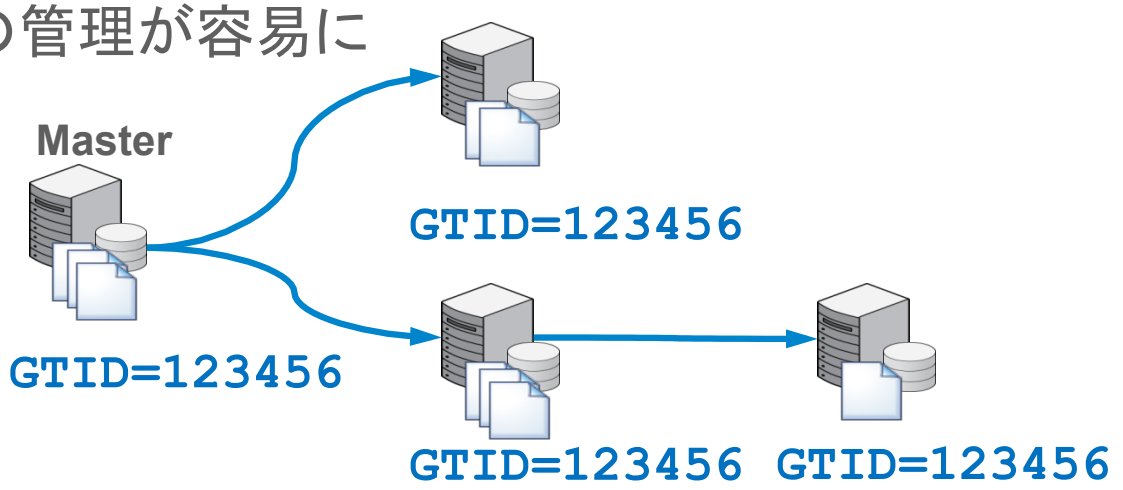
運用効率 / オンライン

GTID(Global Transaction Identifiers)のオンラインでの設定



グローバルトランザクションID (GTID)

- レプリケーション環境でのトランザクションの追跡/比較が容易に可能
  - トランザクションを一意に識別できる識別子をバイナリログに記録
  - GTIDの書式 *サーバUUID:トランザクションID [-トランザクションID]*
    例) 3E11FA47-71CA-11E1-9E33-C80AA9429562:1-5
- フェイルオーバーのために最新のスレーブを自動認識
- 多段構成のレプリケーションの管理が容易に



GTID(Global Transaction Identifiers)のオンラインでの設定

- GTIDの設定変更がオンラインに
  - GTIDの設定変更(onまたはoff)の変更中も参照更新可能
- サーバ間でのデータ同期不要
- サーバ再起動不要
- レプリケーション構成の変更不要
  - 任意のレプリケーション構成で利用可能
- 運用中にGITDの利用開始または停止が可能

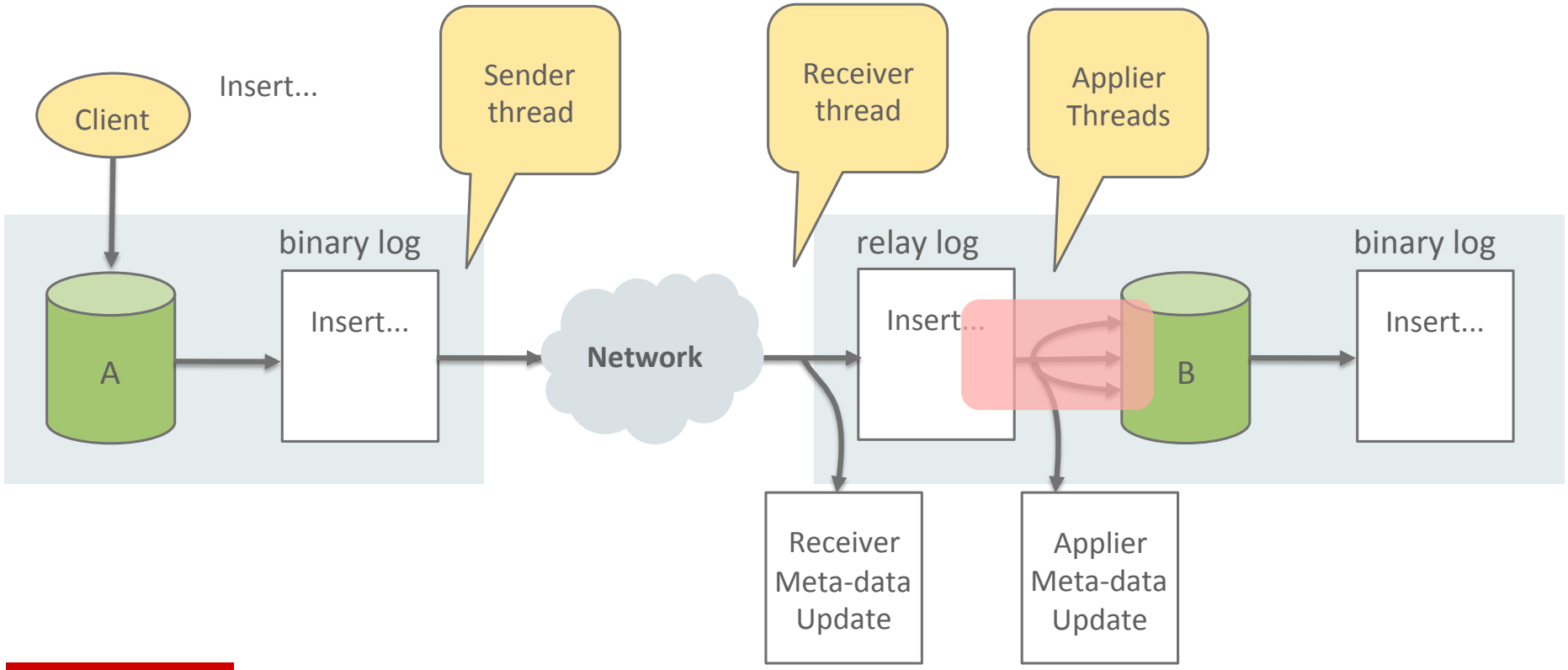
GTID(Global Transaction Identifiers)のオンラインでの設定

- GTIDを有効にする作業をよりシンプルに

1) SET @@GLOBAL.GTID_MODE = OFF_PERMISSIVE; (全サーバ上)
2) SET @@GLOBAL.GTID_MODE = ON_PERMISSIVE; (全サーバ上)
3) レプリケーションの遅延以上の時間待機する
4) SET @@GLOBAL.GTID_MODE = ON; (全サーバ上)

http://dev.mysql.com/doc/refman/5.7/en/replication-mode-change-online-enable-gtids.html

レプリケーションのフィルタをオンラインで変更

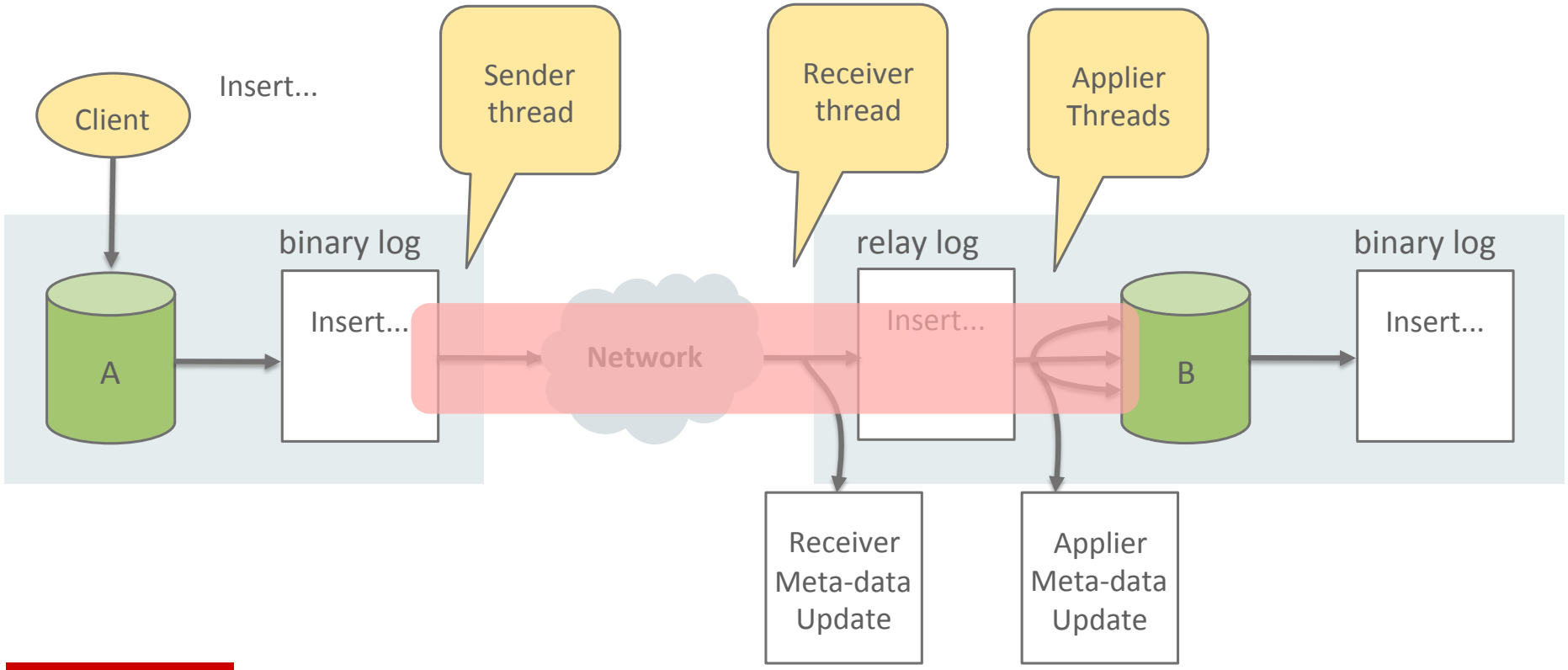


レプリケーションのフィルタをオンラインで変更

- スレーブのレプリケーションフィルタを動的に変更
  - スレーブサーバの再起動不要
  - 全てのスレーブでのフィルタをサポート
  - 各種の文字コードによる値の設定が可能

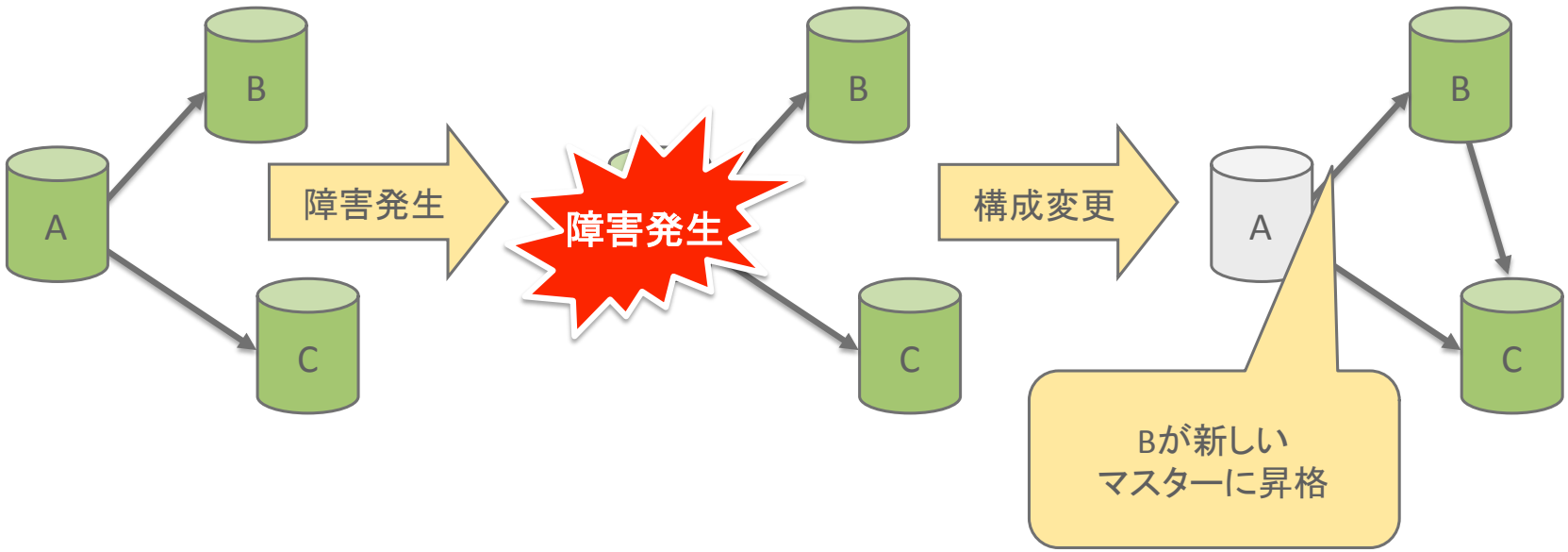
```
mysql> CHANGE REPLICATION FILTER REPLICATE_DO_DB= (db1, db2)
```

新マスターへの切り替え/フェールオーバーをオンラインで



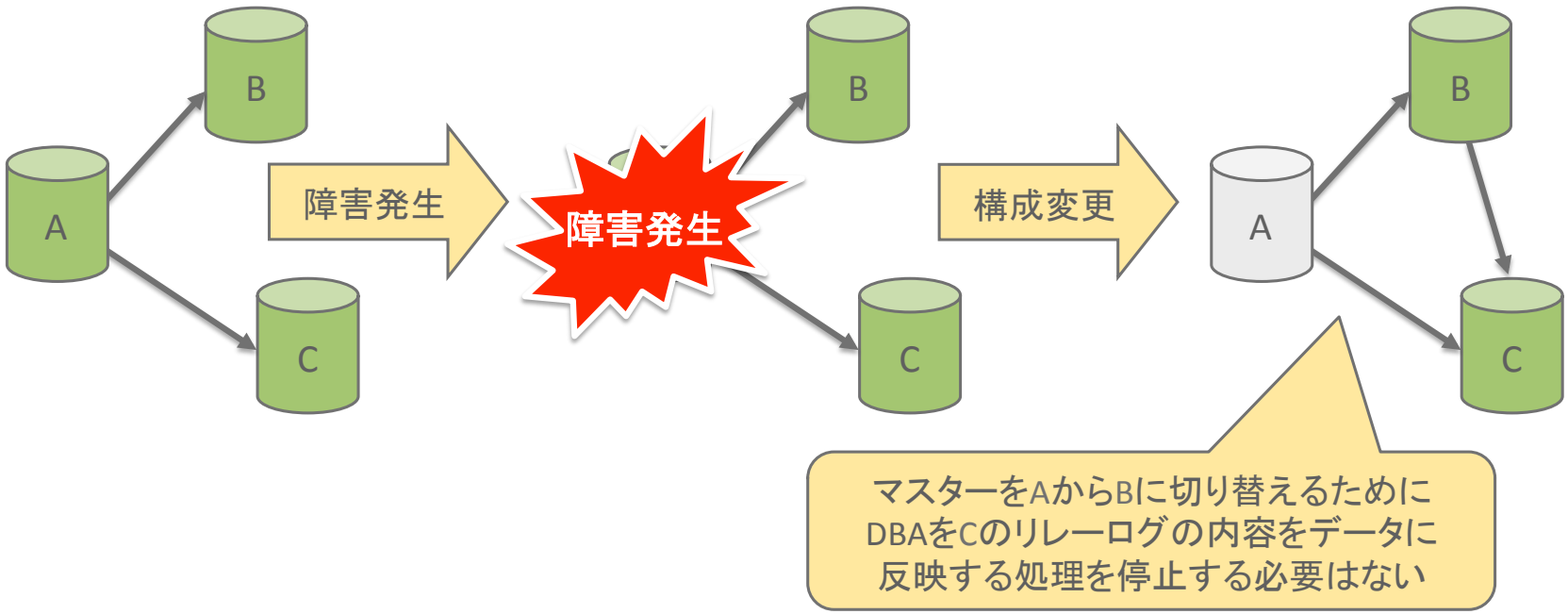
新マスターへの切り替え/フェールオーバーをオンラインで

マスターをAからBへの切り替える際にApplierスレッドの停止不要



新マスターへの切り替え/フェールオーバーをオンラインで

マスターをAからBへの切り替える際にApplierスレッドの停止不要



新マスターへの切り替え/フェールオーバーをオンラインで

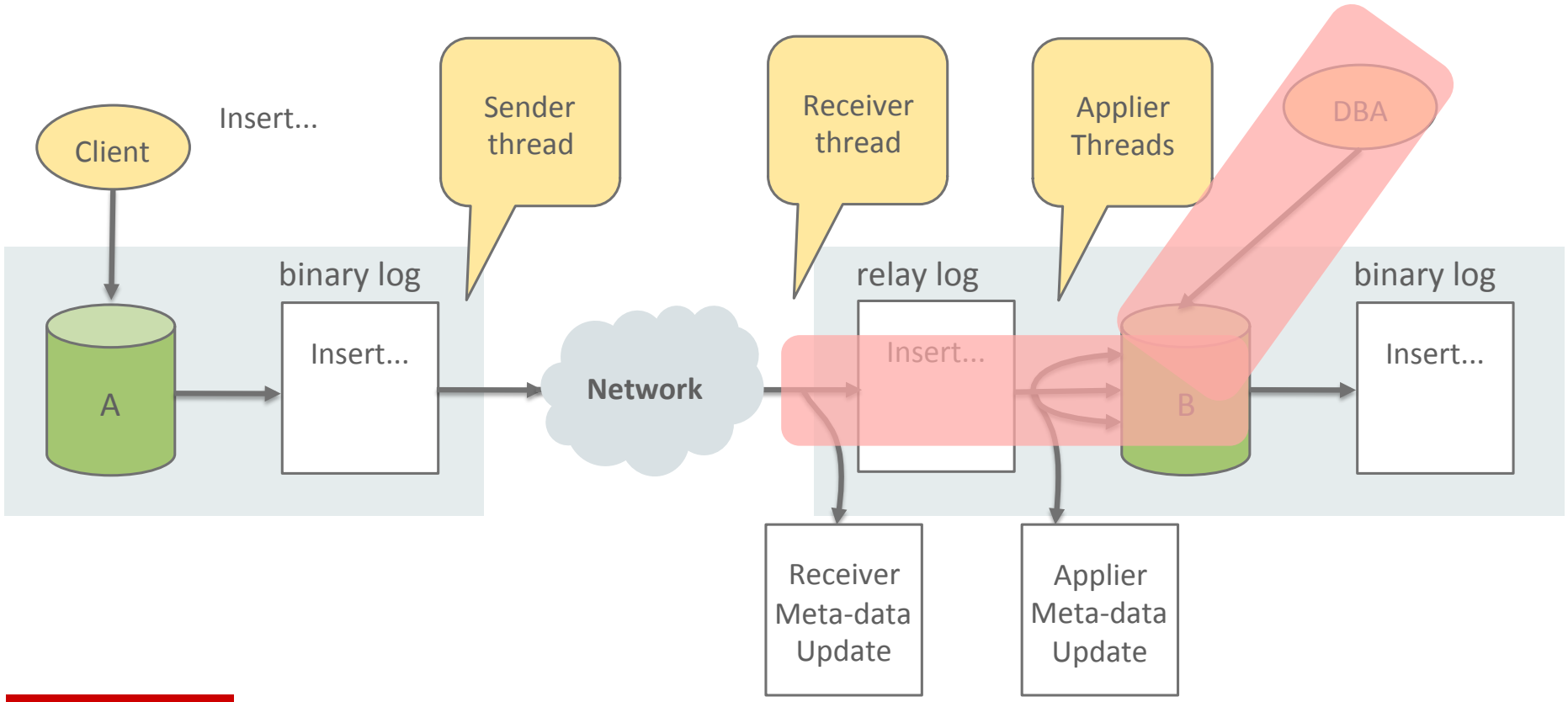
- フェールオーバー中もオンラインで処理を継続:

```
mysql> STOP SLAVE IO_THREAD;
mysql> CHANGE MASTER TO MASTER_HOST='master2', …;
mysql> START SLAVE IO_THREAD;
```

- Receiverスレッドの停止、マスター変更、再開の間もApplierスレッドは処理を継続
- Applierスレッドの設定変更中もReceiverスレッドは処理を継続

```
mysql> STOP SLAVE SQL_THREAD;
mysql> CHANGE MASTER TO MASTER_DELAY=3600, …;
mysql> START SLAVE SQL_THREAD;
```

レプリケーション監視の改善

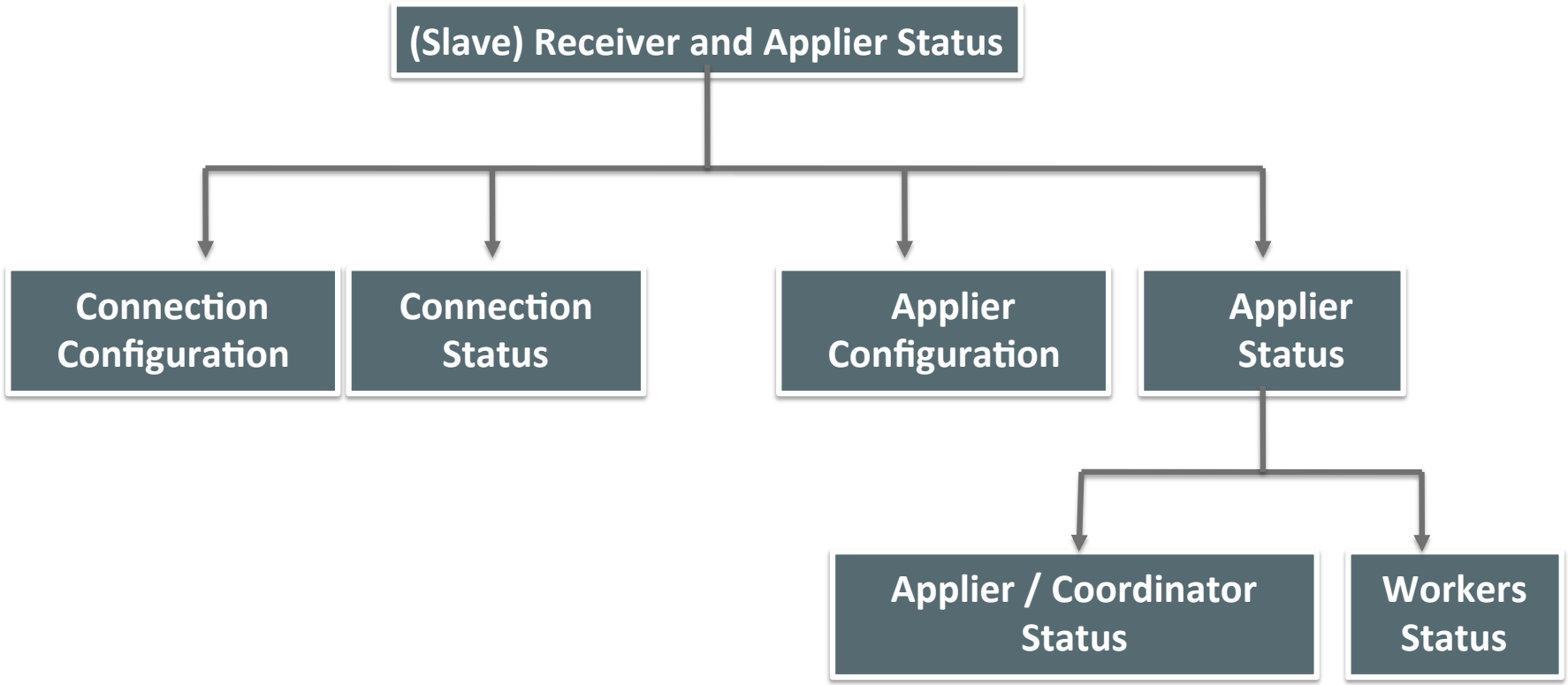


レプリケーション監視の改善

Performance SchemaのReplication関連テーブル

- SQL文にて監視
- 意味の異なる情報は別々の場所に格納
- 拡張可能, 新しい機能との連携
- より正確で一貫性のある識別子の名称
- マスター/スレーブ、マルチソース、グループレプリケーション対応

レプリケーション監視の改善



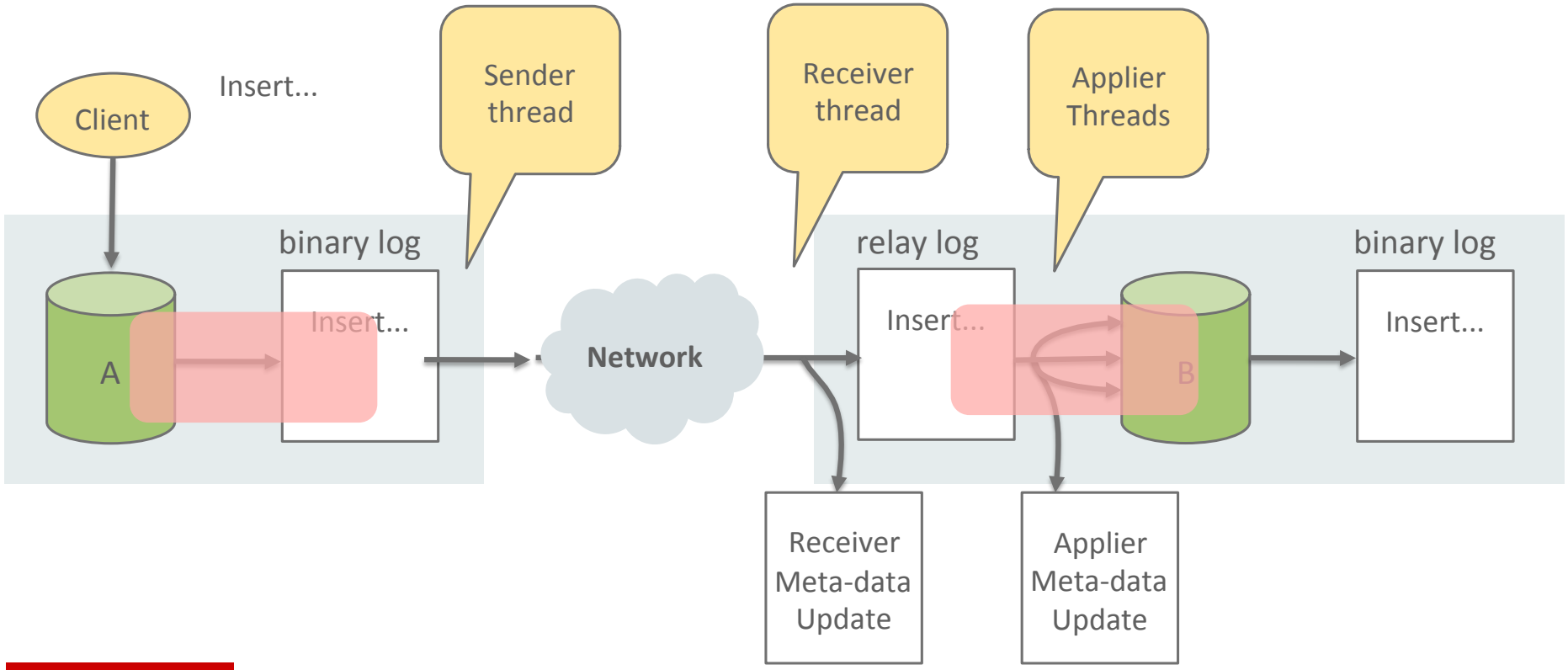
レプリケーション監視の改善

```
mysql> select * from performance_schema.replication_applier_status_by_worker\G
*************************** 1. row ***************************
         CHANNEL_NAME:
            WORKER_ID: 1
            THREAD_ID: 35
        SERVICE_STATE: ON
LAST_SEEN_TRANSACTION: 4ba0eb86-63c3-11e4-92ba-28b2bd168d07:2368
    LAST_ERROR_NUMBER: 0
   LAST_ERROR_MESSAGE:
 LAST_ERROR_TIMESTAMP: 0000-00-00 00:00:00
*************************** 2. row ***************************
         CHANNEL_NAME:
            WORKER_ID: 2
            THREAD_ID: 36
        SERVICE_STATE: ON
LAST_SEEN_TRANSACTION: 4ba0eb86-63c3-11e4-92ba-28b2bd168d07:2367
    LAST_ERROR_NUMBER: 0
   LAST_ERROR_MESSAGE:
 LAST_ERROR_TIMESTAMP: 0000-00-00 00:00:00
2 rows in set (0,00 sec)
```

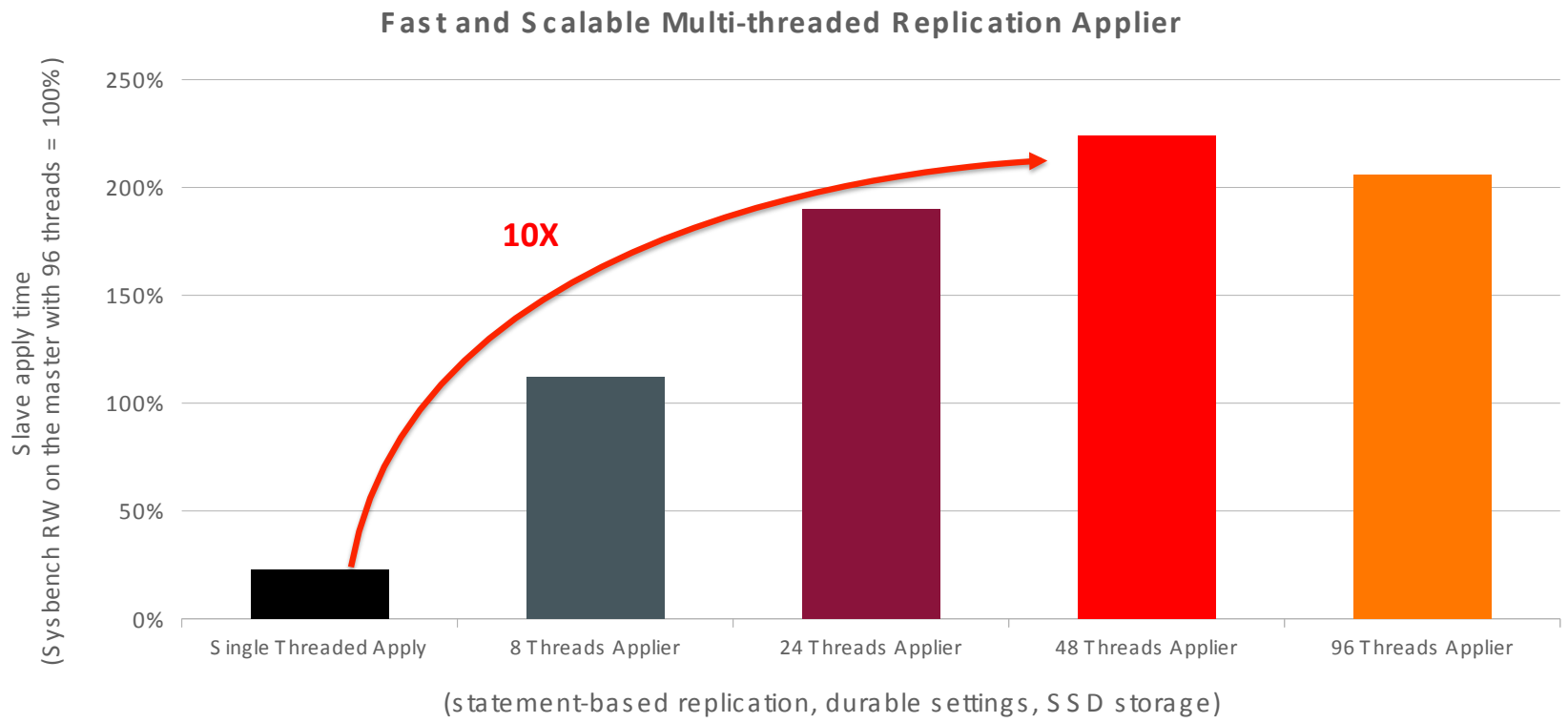
MySQL 5.7 レプリケーション新機能

パフォーマンス

Applierスレッドの性能向上 – ロックベースの並列処理



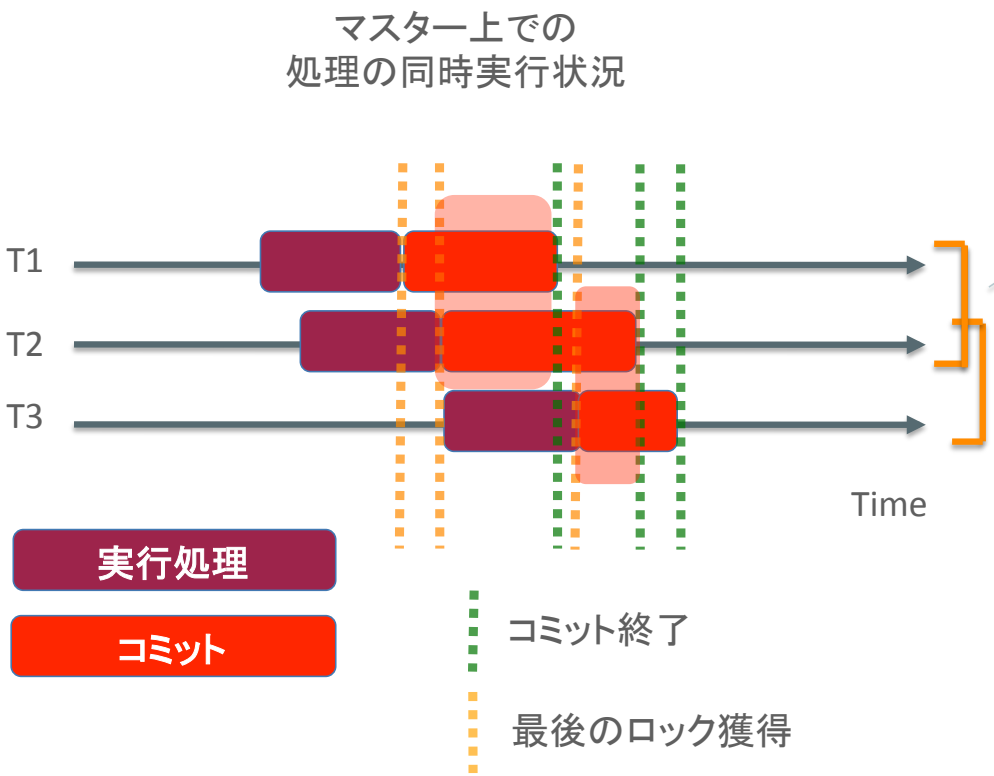
Applierスレッドの性能向上 – ロックベースの並列処理



Applierスレッドの性能向上 – ロックベースの並列処理

- マスターでの実行状況の情報をもとに並列処理:
  - マスター上で相互にブロックしない同時実行トランザクションは、「競合しないトランザクション」としてバイナリログに記録
- スレーブ上では:
  - 「競合しないトランザクション」は並列実行される;
  - 同時実行トランザクションは個別にコミットされ、相互に待つことはない

Applierスレッドの性能向上 – ロックベースの並列処理



T1とT2はスレーブ上で
並列実行される

T2とT3はスレーブ上で
並列実行される

- T2はT1と並列実行されるようにスケジュールされる
- T3はT1の終了後に実行されるようにスケジュールされる

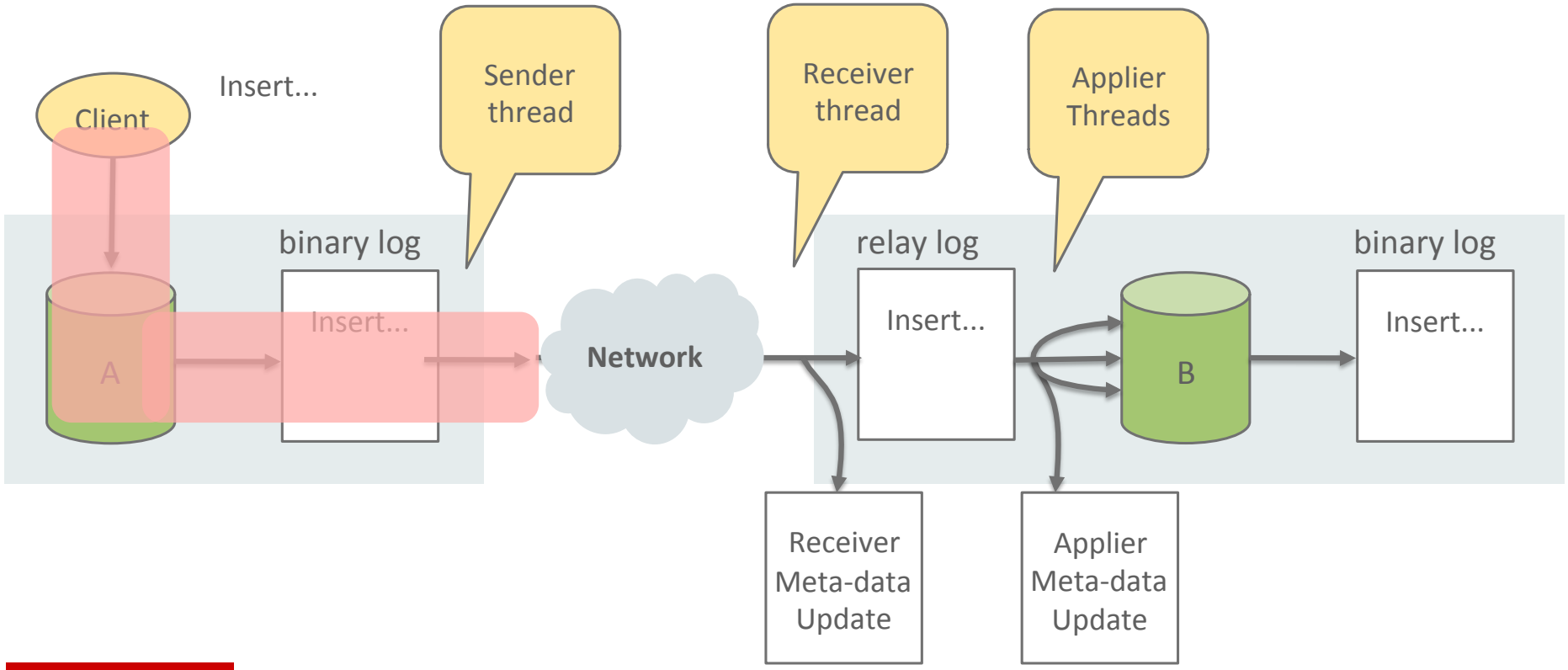
Applierスレッドの性能向上 – ロックベースの並列処理

- 文(STATEMENT)ベースおよび行(ROW)ベースの両フォーマット対応
- スケジュールのポリシーは下記コマンドで制御:

```
mysql> SET slave_parallel_type= [logical_clock|database]
```

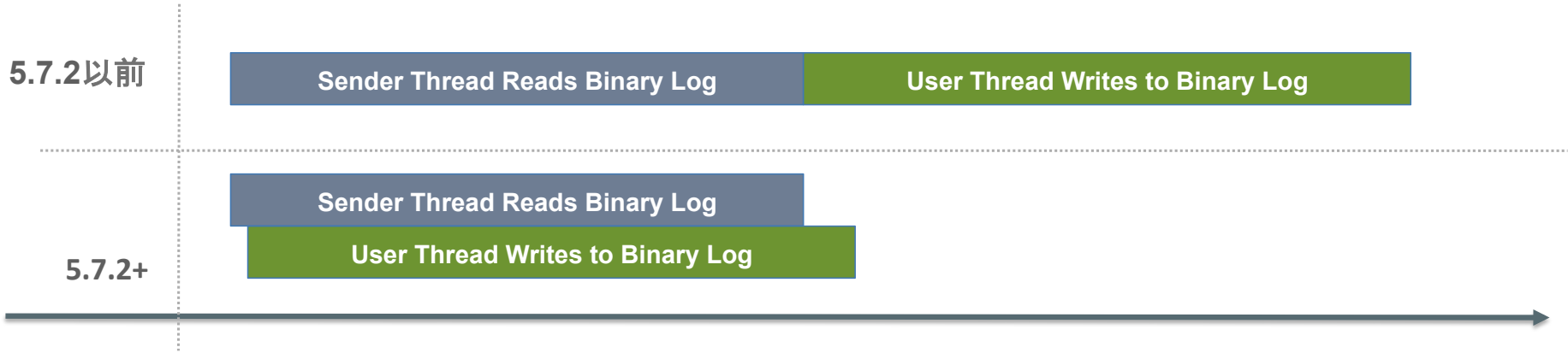
  - `logical_clock` – ロックのタイムスタンプを通じてスケジューリング
  - `database` – 5.6と同様のスケジューリング (データベースが異なれば並列実行)
- スレーブの性能拡張性は引き続き改良！

ユーザスレッドとSenderスレッドの同期の改善

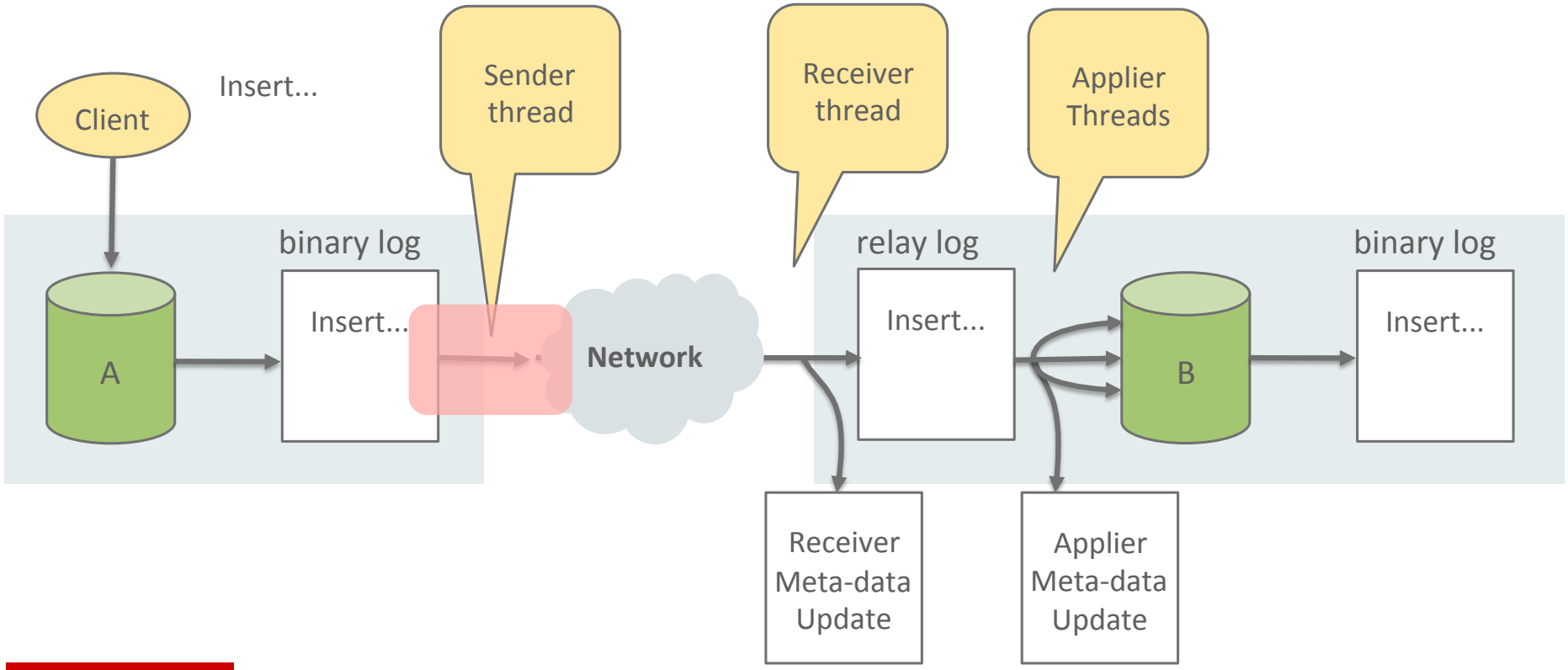


ユーザスレッドとSenderスレッドの同期の改善

- バイナリログに対するユーザスレッドでの書き込みと Senderスレッドでの読み込みを並列化
  - Senderスレッドによるユーザセッション処理のブロックを最小化
  - ユーザセッションとSenderスレッドのスループットを向上



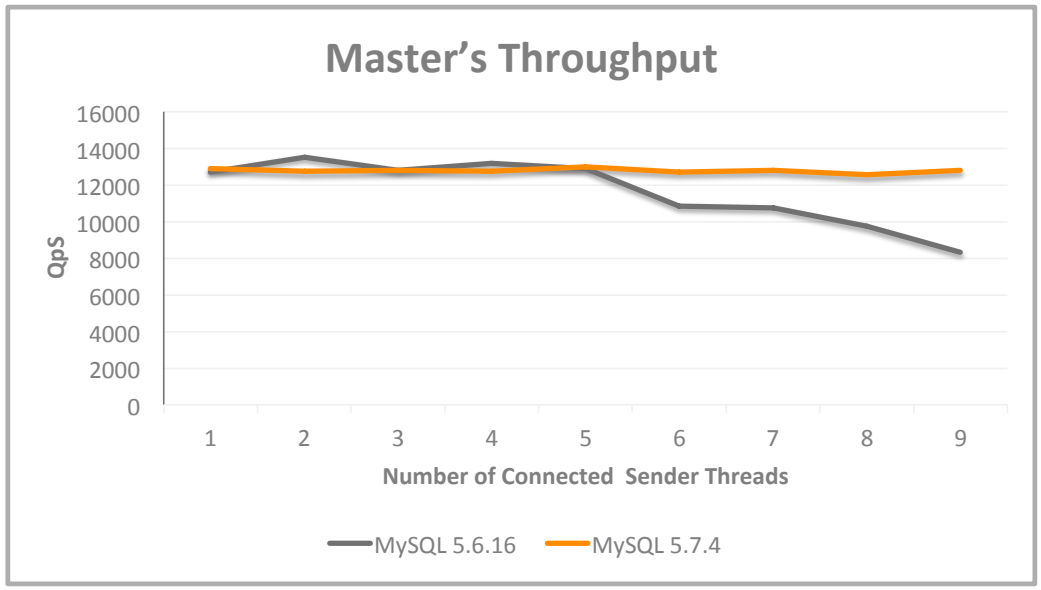
Senderスレッドの高速化



Senderスレッドの高速化

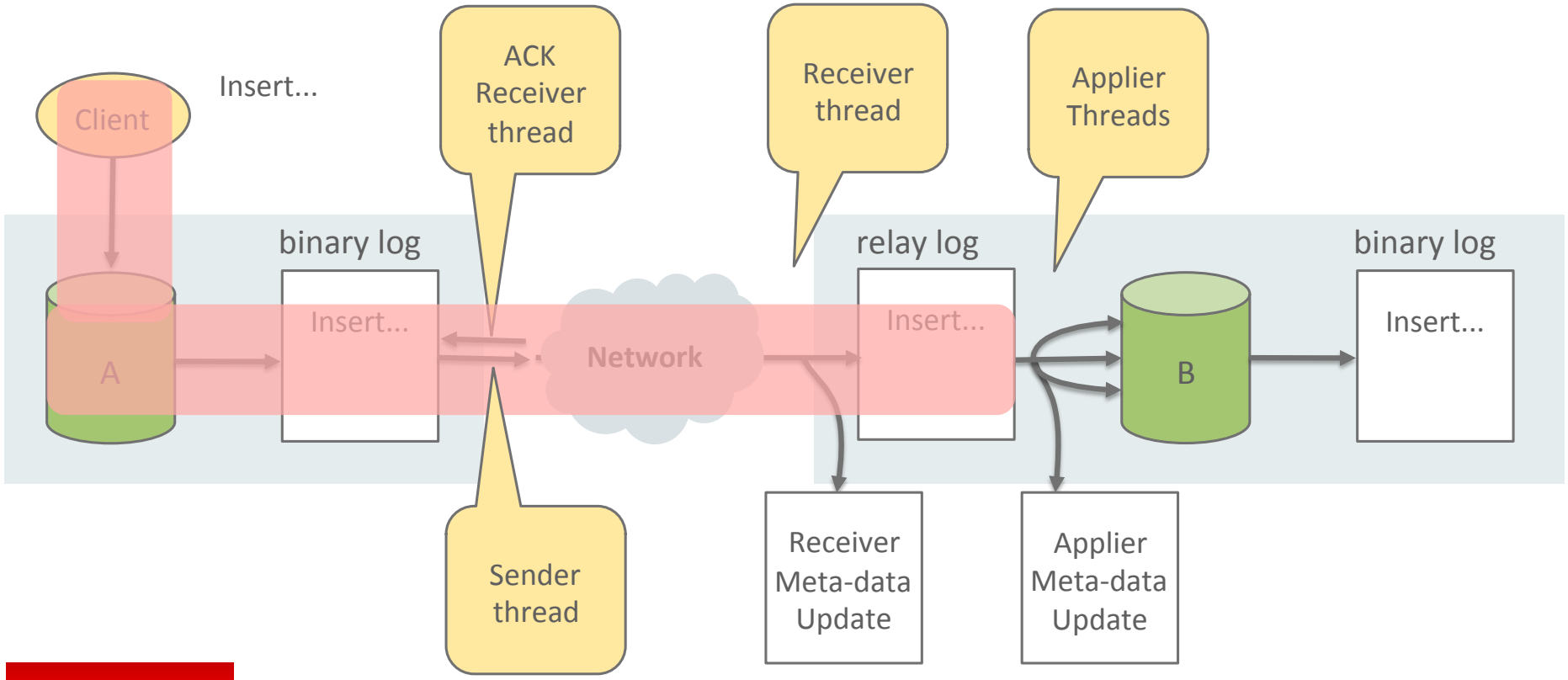
- アロケートされた送信バッファを毎回解放しないように変更
- より大きな送信バッファが必要な場合は、その際に拡張
- 拡張された送信バッファが使われなくなると動的に縮小
- MySQL 5.7.2でのSenderスレッドの改良:
  - マスターの性能拡張性の向上;
  - リソース消費の削減 (CPU);
  - 負荷のピーク時のマスター特にDumpスレッドの処理負荷の軽減

Senderスレッドの高速化



- mysqlslapを使ったマイクロベンチマーク
- 100万クエリ, concurrency=200, commit=1
- スレーブ台数を増加させる(mysqlbinlogをリモートから接続してスレーブのように振る舞わせる)
- 48 cores HT / 512 GB RAM / HDD

準同期レプリケーションの高速化 – ACK Receiverスレッド



準同期レプリケーションの高速化 – ACK Receiverスレッド

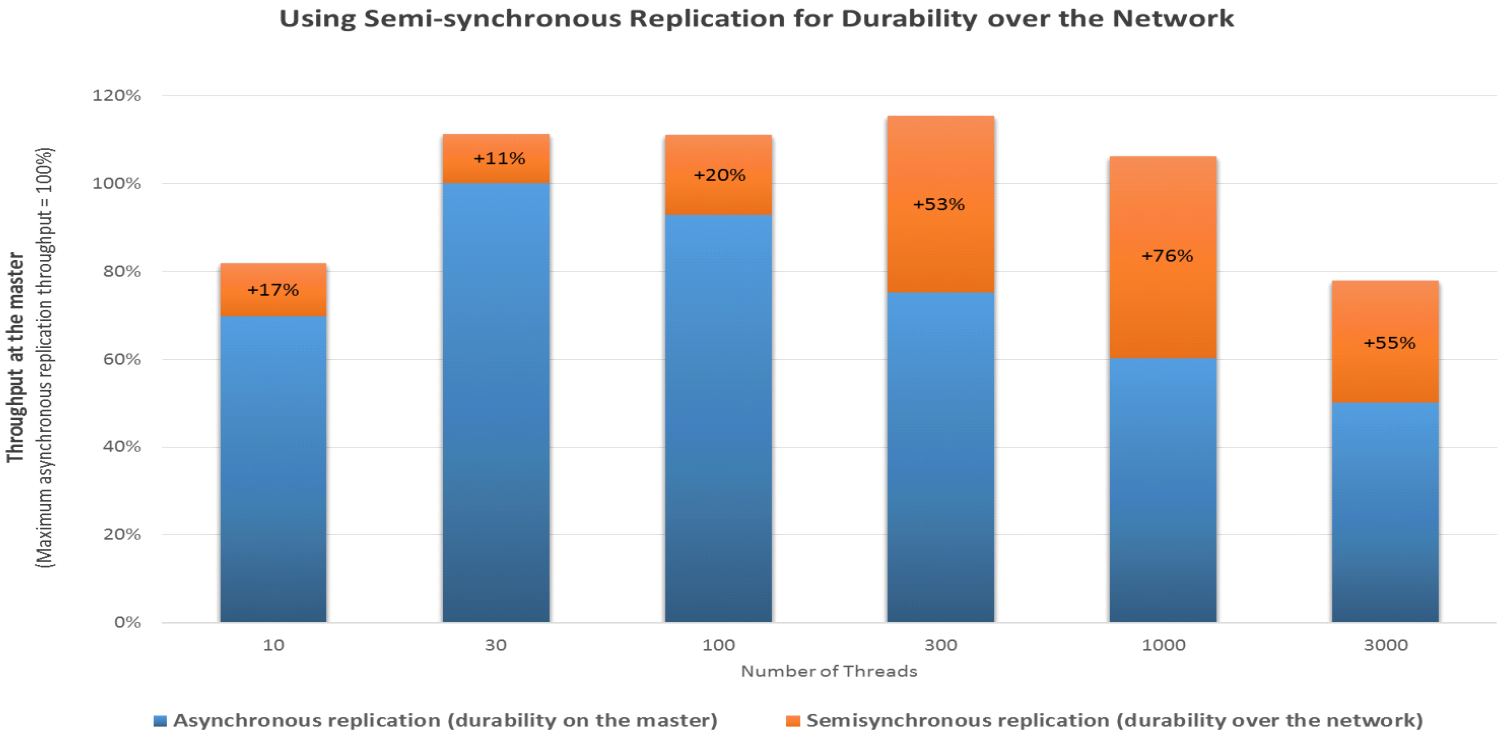
- 連続したトランザクションがスレーブからのACKをお互いに待たない
  - トランザクションt1とt2を同時にSenderスレッドがスレーブに送る
  - それぞれのACKを別のスレッドが受け取る
  - t2の準同期のレイテンシにはt1の処理によるレイテンシは含まれない
- 準同期レプリケーション有効時にスレッドが開始される

```
mysql> SET GLOBAL rpl_semi_master_enabled= ON
```

- 準同期レプリケーション無効時にスレッドが停止される

```
mysql> SET GLOBAL rpl_semi_master_enabled= OFF
```

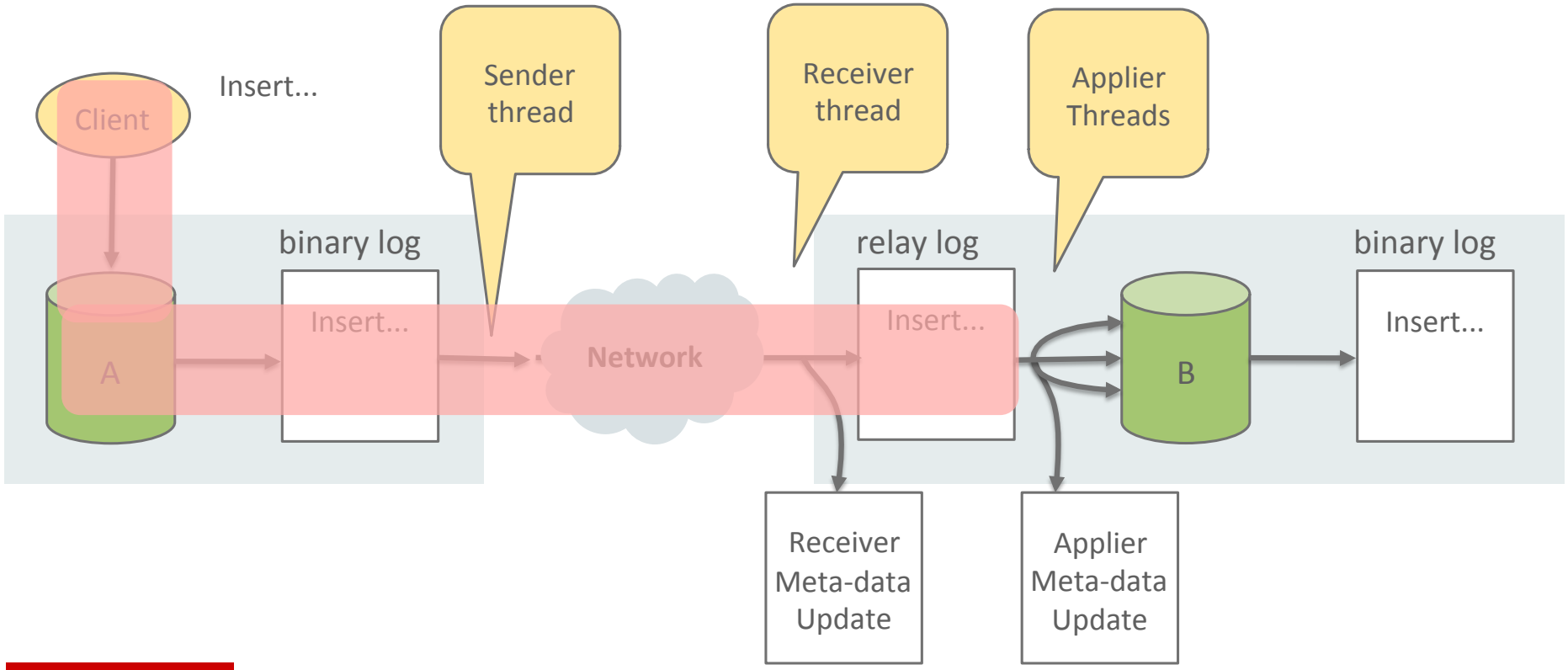
準同期レプリケーションの高速化: 性能拡張性向上



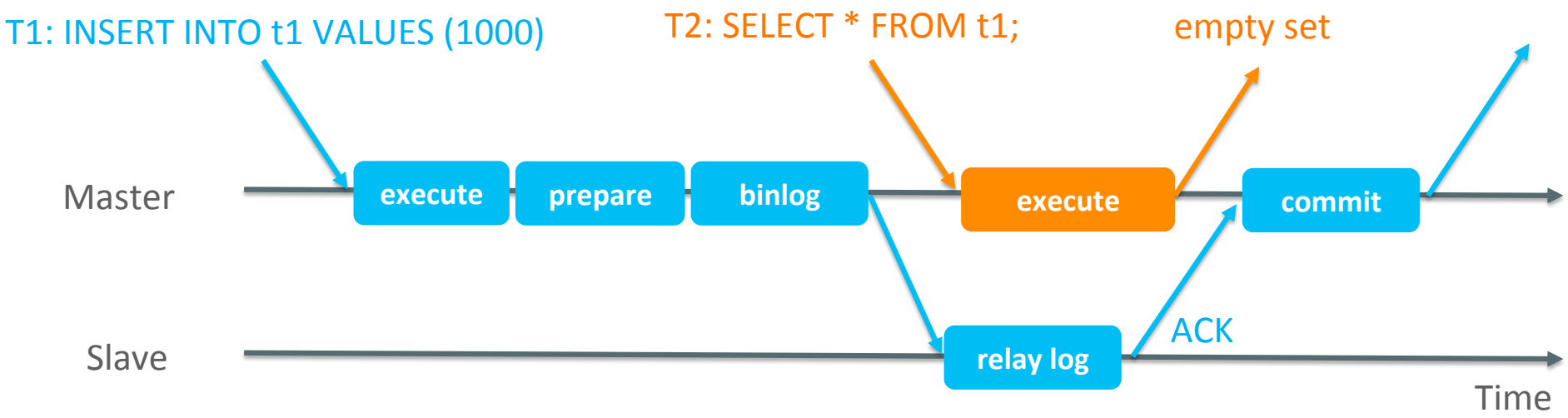
MySQL 5.7 レプリケーション新機能

信頼性

Loss-less準同期レプリケーション



Loss-less準同期レプリケーション

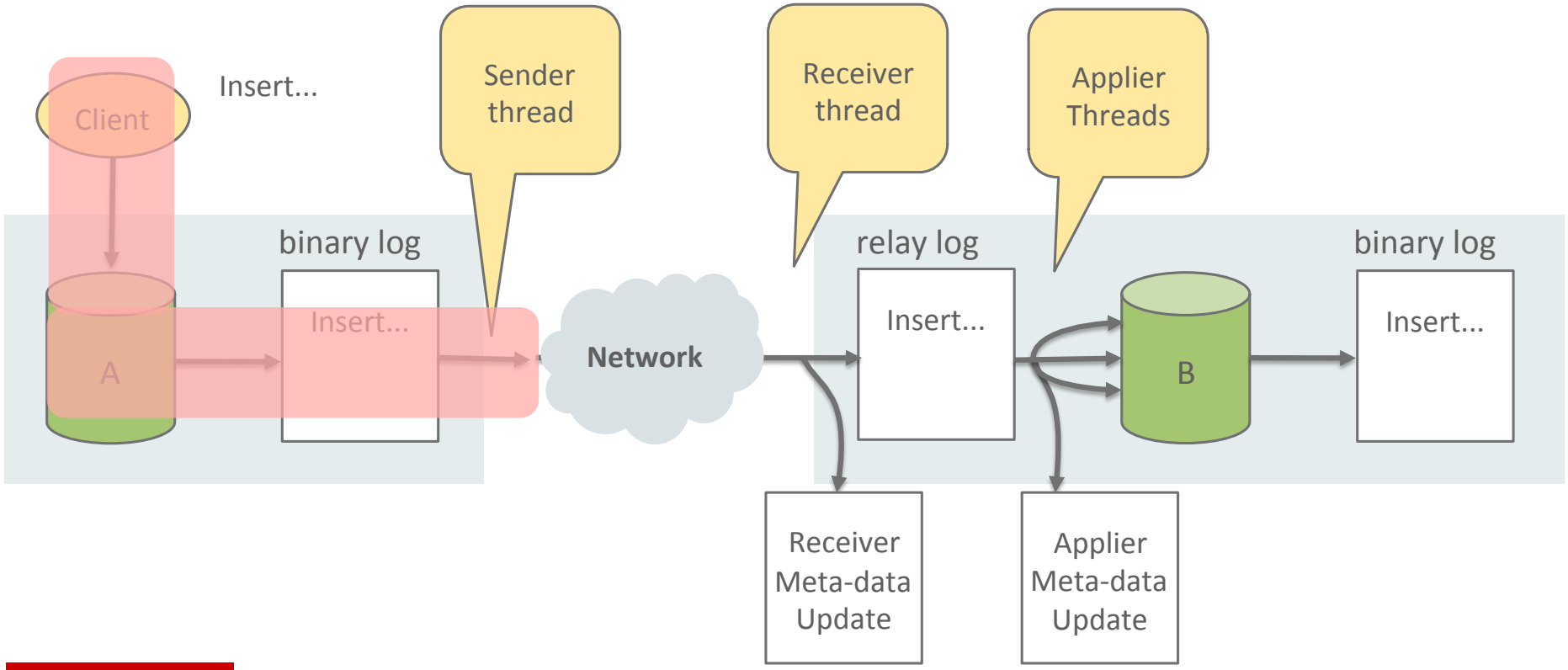


Loss-less準同期レプリケーション

- マスターはスレーブからのACKを受け取ってからコミット
  (5.6まではコミット後にスレーブに処理を転送)
  - 他のトランザクションはACK待ちの間は該当トランザクションによる変更は見えない
- マスターに障害が発生した際でも、スレーブに転送されたトランザクションのみが他のトランザクションから見える状態
- MySQL 5.6までの挙動か新しいLoss-lessを選択可能

```
mysql> SET rpl_semi_sync_master_wait_point= [AFTER_SYNC|AFTER_COMMIT]
```

準同期レプリケーション – 複数ACKを待つ



準同期レプリケーション – 複数ACKを待つ

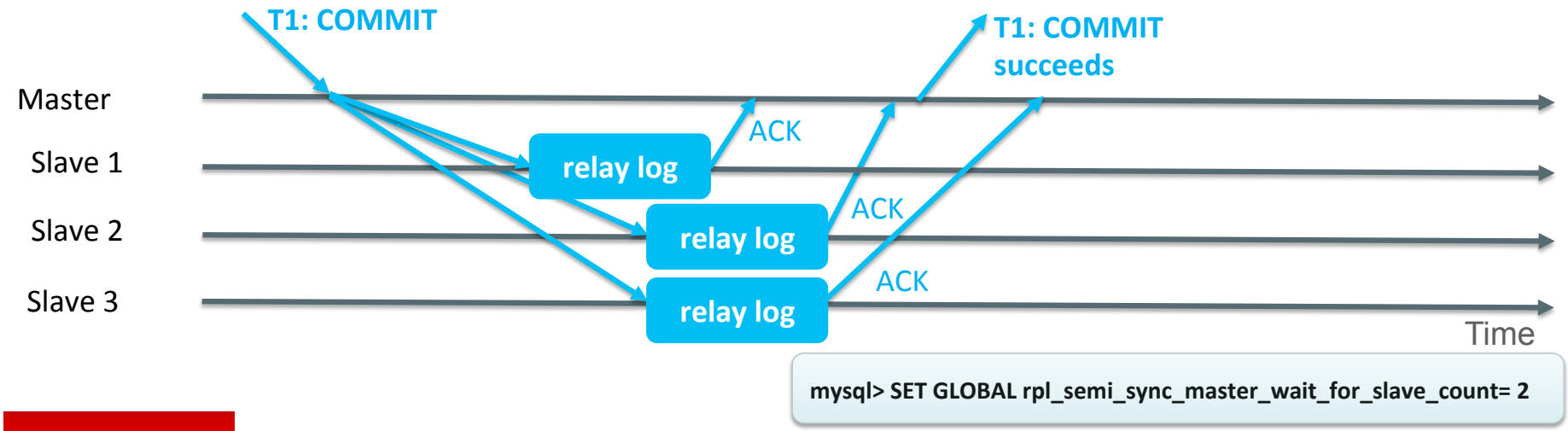
- 指定したN台のスレーブからACKを受信するまでコミットを行わない
- 動的に設定可能:

```
mysql> SET GLOBAL rpl_semi_sync_master_wait_for_slave_count= N
```

準同期レプリケーション – 複数ACKを待つ

- 指定したN台のスレーブからACKを受信するまでコミットを行わない
- 動的に設定可能:

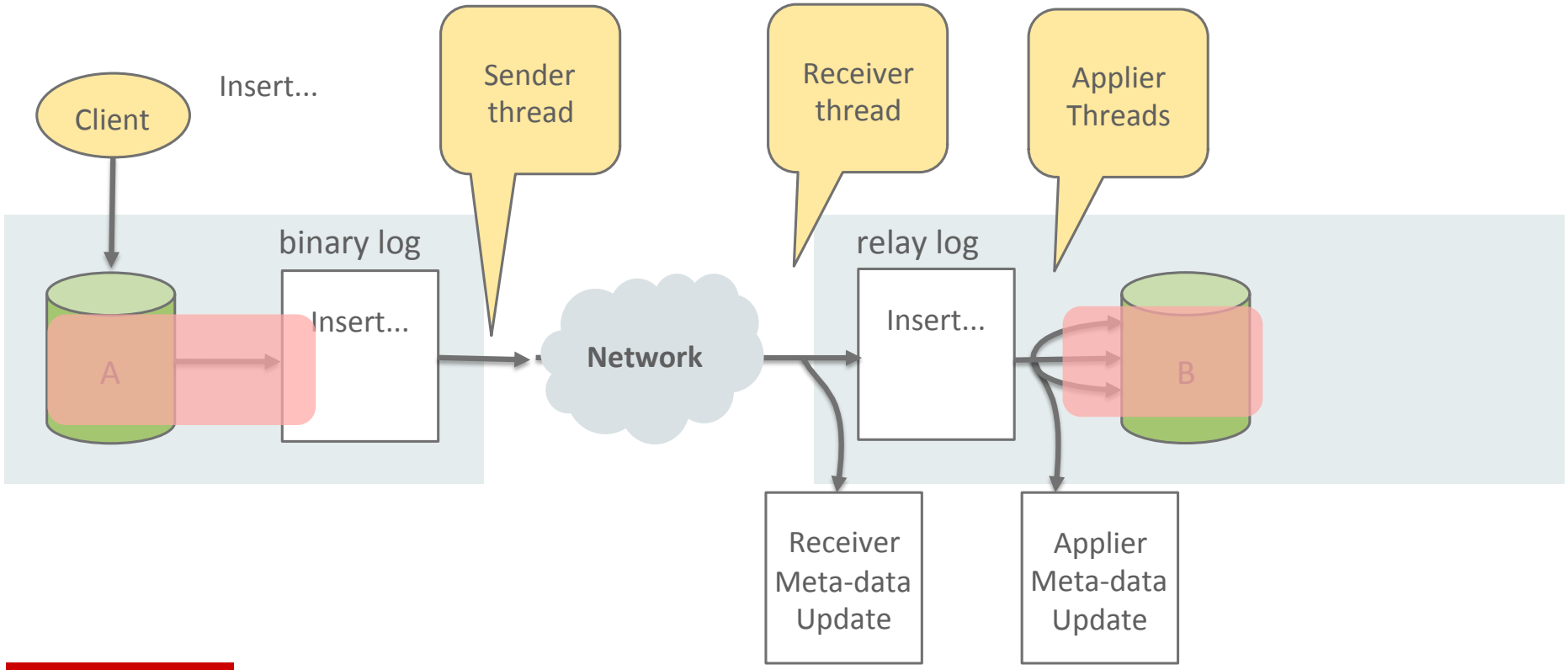
```
mysql> SET GLOBAL rpl_semi_sync_master_wait_for_slave_count= N
```



MySQL 5.7 レプリケーション新機能

柔軟性

GTIDをテーブルに格納



GTIDをテーブルに格納

利用例

- バイナリログを有効にしていないスレーブでもGTIDを利用可能
  - マスターに昇格する予定のないスレーブでもGTIDを利用した auto positioning(自動的にトランザクションの進捗を見つける仕組み)を利用可能

GTIDをテーブルに格納
実装方式

- バイナリログ有効時
  - GTIDをバイナリログのみに格納 (トランザクション処理の一部として)
  - バイナリログのローテーション時にそれまでに実行済みのGTIDを
    - 新しいバイナリログに記録
    - gtid_executedテーブルに記録
- バイナリログ無効時
  - GTIDをgtid_executedテーブルに格納 (トランザクション処理の一部として)
  - gtid_executed_compression_periodで設定したトランザクション数毎に「圧縮」
- GTIDは常に一連のトランザクション処理の一部として格納される

GTIDをテーブルに格納

実装方式

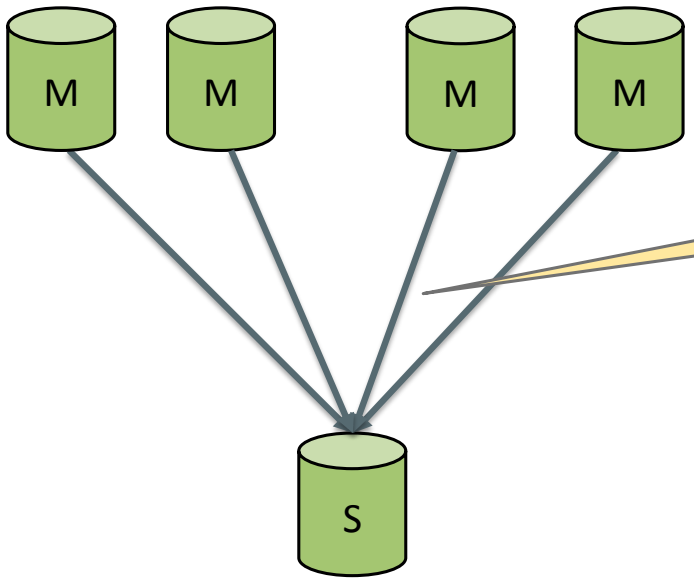
- GTIDをMySQLのシステムテーブルに格納 (値の範囲を格納)

```
CREATE TABLE gtid_executed(
   source_uuid CHAR(36) NOT NULL,
   interval_start BIGINT NOT NULL,
   interval_end BIGINT NOT NULL,
   PRIMARY KEY(source_uuid, interval_start)
 );
```

- トランザクションのコミット時に各GTIDがテーブルにも格納される
- GTID個別の値から一定間隔毎に値の範囲に「圧縮」する
  - 新しいサーバ変数で間隔を制御可能: gtid_executed_compression_period

```
mysql> SET GLOBAL gtid_executed_compression_period= N;(N – トランザクション数)
```

マルチソースレプリケーション



1台のスレーブが複数のマスターを持つ構成

1台のサーバにデータを集約する構成:

- バックアップを1台で実行;
- 分析用途の複雑なクエリの実行;
- クラスタ間レプリケーションのデータハブ

マルチソースレプリケーション

- 1台のサーバ(スレーブ)が複数のソース(マスター)からレプリケーション
- 複数のチャネル(チャネル: 接続スレッド、リレーログ、Applierスレッド)は個別に稼働/停止可能
- マルチスレッドスレーブとの統合
  - 各チャネルがそれぞれのマルチスレッドApplierスレッドの管理
- パフォーマンススキーマの新しいテーブルにて稼働監視可能
  - replication_applier_status_by_coordinator テーブルにて
    チャネル毎のApplierスレッドが複数あることが確認可能
  - replication_connection_statusテーブルにて
    ソース毎のマスターへの接続スレッドが複数あることが確認可能

マルチソースレプリケーション

- CHANGE MASTER TO実行時にFOR CHANNELでチャネルを指定

```
CHANGE MASTER TO
      MASTER_HOST='master1', MASTER_USER='rpl',
      MASTER_PORT=3451, MASTER_PASSWORD='',
      MASTER_AUTO_POSITION = 1 FOR CHANNEL 'master-1';
```

- 指定したチャネルだけレプリケーションを開始/停止することも可能

```
START SLAVE thread_types FOR CHANNEL 'master-1';
START SLAVE thread_types FOR CHANNEL 'master-1';
```

- 指定したチャネルだけレプリケーション設定をリセットすることも可能

```
RESET SLAVE thread_types FOR CHANNEL 'master-1';
```

マルチソースレプリケーション

- GTIDとの統合
- クラッシュセーフスレーブとの統合
  - 障害復旧時にレプリケーションの進捗関連のメタデータをテーブルからリカバリ
  - **`--master-info-repository=TABLE`**
  - **`--relay-log-info-repository=TABLE`**
- ソースの数には事実上上限はない
  (標準バイナリでは256, ソースからビルドすれば変更可能)
- 各ソースを個別に設定可能

MySQL 5.7 レプリケーション新機能

その他の改良点

“細かな改良”, でも実践で役立つ数多くの改良点!

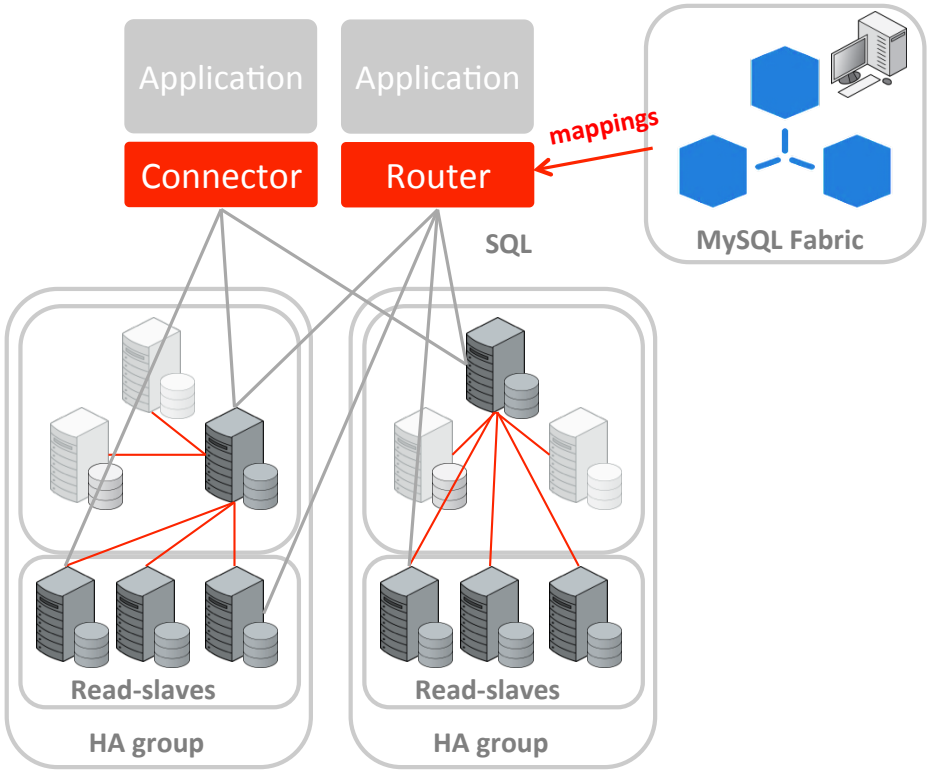
- マルチスレッドのApplierスレッドで、失敗した**トランザクションの再試行**が可能
- マルチスレッドのApplierスレッドの**コミット順を維持**するオプション
- **mysqlbinlog**ツールに**SSL**オプション
- **mysqlbinlog**に**データベース名のリライトルール**適用可能
- 指定されたGTIDになるまでApplierスレッドの処理を一時的に停止する関数
  - `WAIT_UNTIL_SQL_THREAD_AFTER_GTIDS(gtid_set[, timeout][,channel])`
  - `WAIT_FOR_EXECUTED_GTID_SET(gtid_set[, timeout])`
- MySQLのクライアントサーバ間**プロトコルにGITD**の情報を返す
- バイナリログ利用時にも**XAトランザクション**をサポート
- デフォルト値の変更 例) binlog_format=**ROW**, sync_binlog=**1**

MySQL Fabric

シャーディングと高可用性

MySQL Fabric 1.5

高可用性構成とシャードによる拡張性

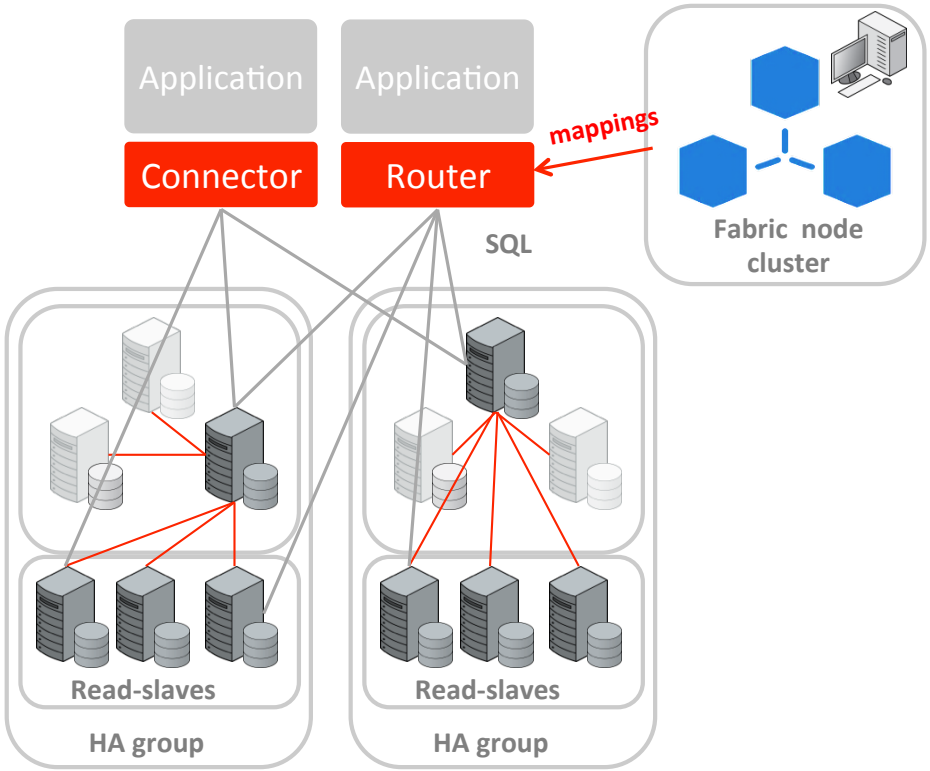


- 高可用性構成
  - サーバ群を監視し、マスタへの自動昇格
  - アプリケーションに影響を最小限に抑えるフェールオーバー
- シャードによるスケールアウトも可能
  - アプリケーションはシャードキーを提供
  - Range または Hash
  - シャード管理ツール
  - グローバルアップデート & テーブル
- 接続オプション
  - Fabric対応Connectors
  - MySQL Router
- OpenStackのサーバプロビジョニング
  - Nova および Neutron APIをサポート

MySQL Fabric 1.6

High Availability + Sharding-Based Scale-out

BETA

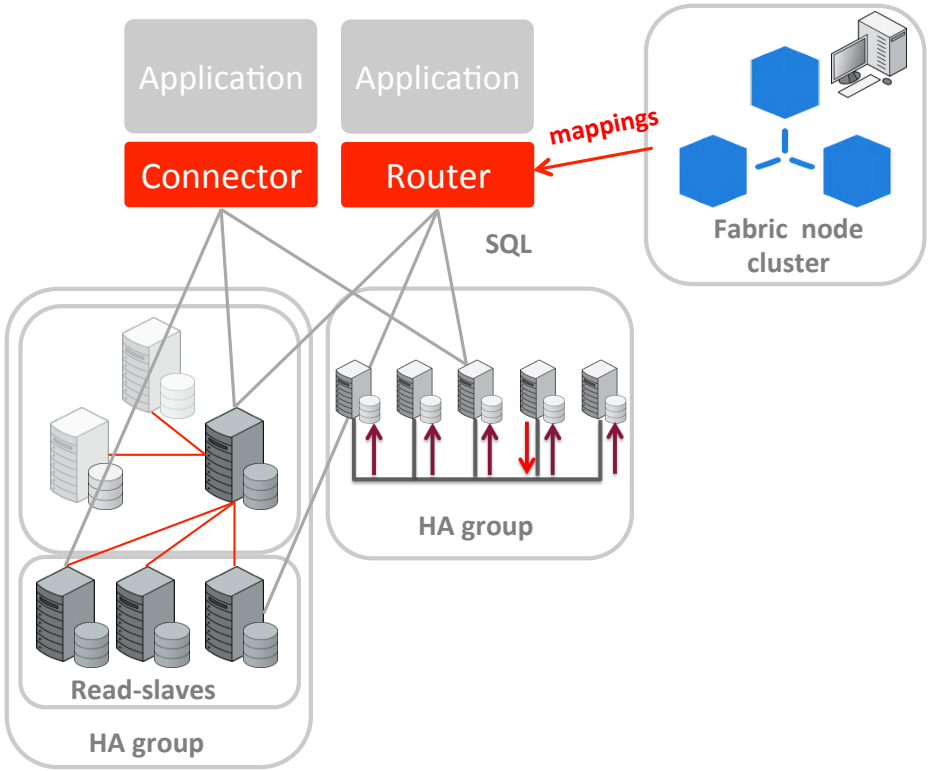


- 高可用性構成
  - サーバ群を監視し、マスタへの自動昇格
  - アプリケーションに影響を最小限に抑えるフェールオーバー
  - 単一障害点無し (SPOF)
- シャードによるスケールアウトも可能
  - アプリケーションはシャードキーを提供
  - Range または Hash
  - シャード管理ツール
  - グローバルアップデート & テーブル
- 接続オプション
  - Fabric対応Connectors
  - MySQL Router
- OpenStackのサーバプロビジョニング
  - Nova および Neutron APIをサポート

MySQL Fabric 1.6

High Availability + Sharding-Based Scale-out

BETA



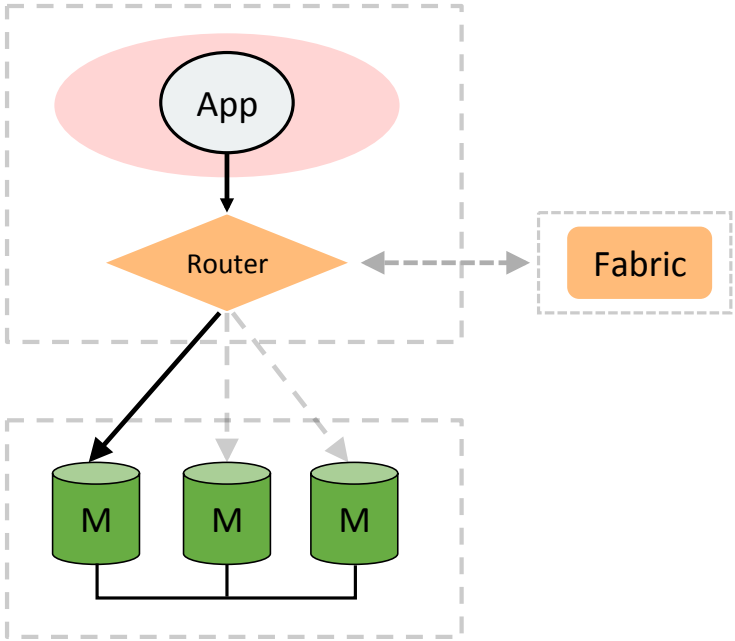
- 高可用性構成
  - サーバ群を監視し、マスタへの自動昇格
  - アプリケーションに影響を最小限に抑えるフェールオーバー
  - 単一障害点無し (SPOF)
- シャードによるスケールアウトも可能
  - アプリケーションはシャードキーを提供
  - Range または Hash
  - シャード管理ツール
  - グローバルアップデート & テーブル
- 接続オプション
  - Fabric対応Connectors
  - MySQL Router
- OpenStackのサーバプロビジョニング
  - Nova および Neutron APIをサポート

MySQL Router GA

接続とトランザクションのルーティング

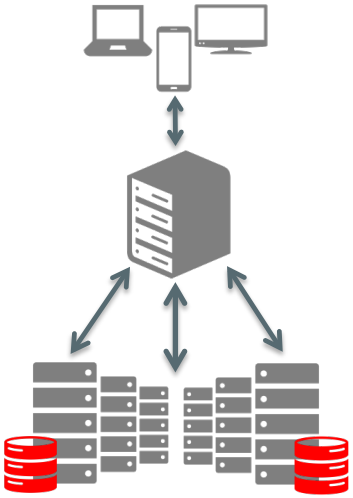
MySQL Router

- 開発の背景
  - MySQL Fabricを透過的に利用したい
    - Connectorの変更不要
    - Fabric対応Connectorがない言語からの利用 (e.g., PHP, Ruby, Perl, C).
  - 参照更新および参照のみの処理を配信
    - どのサーバがマスターかを事前に知る必要がない
    - 新しいマスターへの透過的なフェールオーバー
- 必要となるソフトウェア
  - 多機能かつ純正ツール: **MySQL Router**



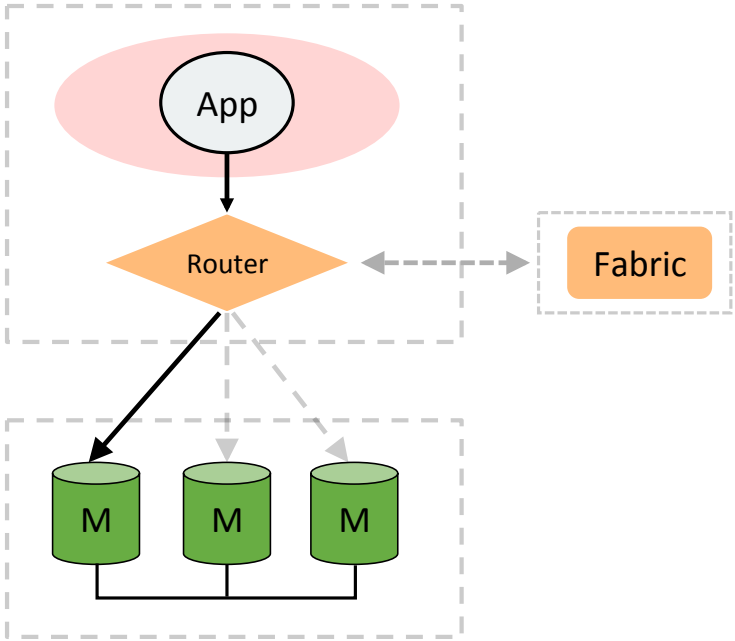
New! MySQL Router

- 接続とトランザクションのルーティング
- MySQLアプリケーションからのアクセスをシンプルに
  - MySQL Fabricサポートを簡単に
    - 高可用性構成
    - シャーディング
  - MySQL グループレプリケーション
  - 各種クラスタリング構成や高可用性構成
- プラグインAPIによる拡張性
- さらなるプラグインの追加 – データ集約、バイナリログ、ロードバランス …
  - ご要望お待ちしております



MySQL Router

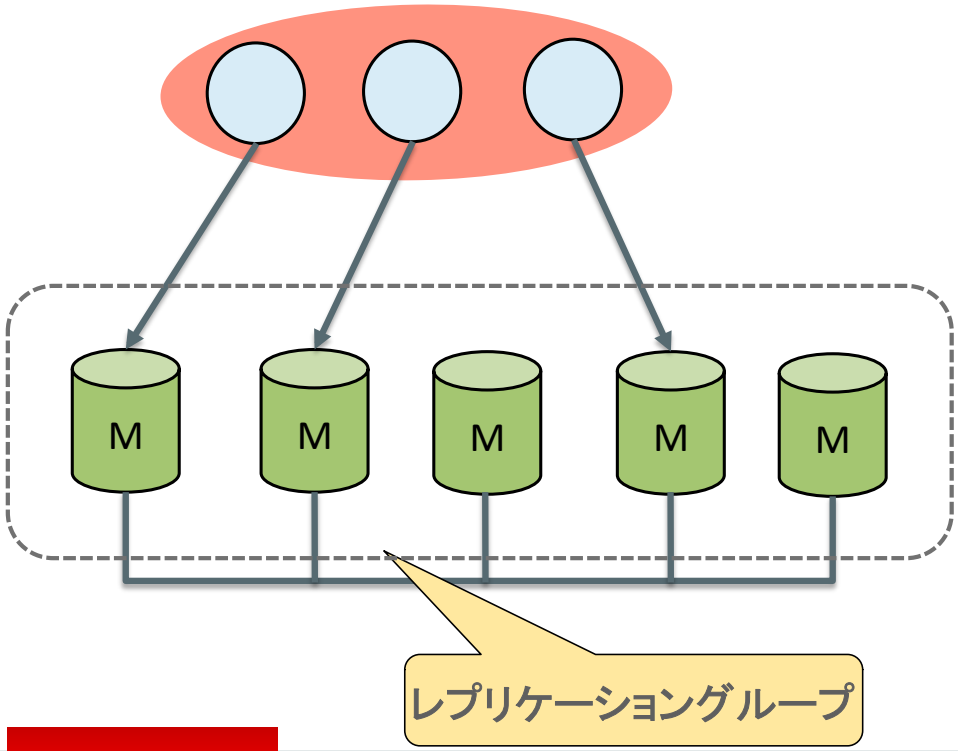
- 特徴
  - 高性能
  - プラグインアーキテクチャ
  - 簡単なセットアップ、設定、実装
- 機能
  - 接続の転送とシンプルなロードバランス
  - FabricのHAグループのシームレスな フェールオーバー
  - Fabric無しでのフェールオーバー (サードパーティ製のツール利用).
  - グループレプリケーション利用時の競合削減



Labsで開発中の機能

MySQL Group Replication Plugin

MySQLグループレプリケーションプラグイン



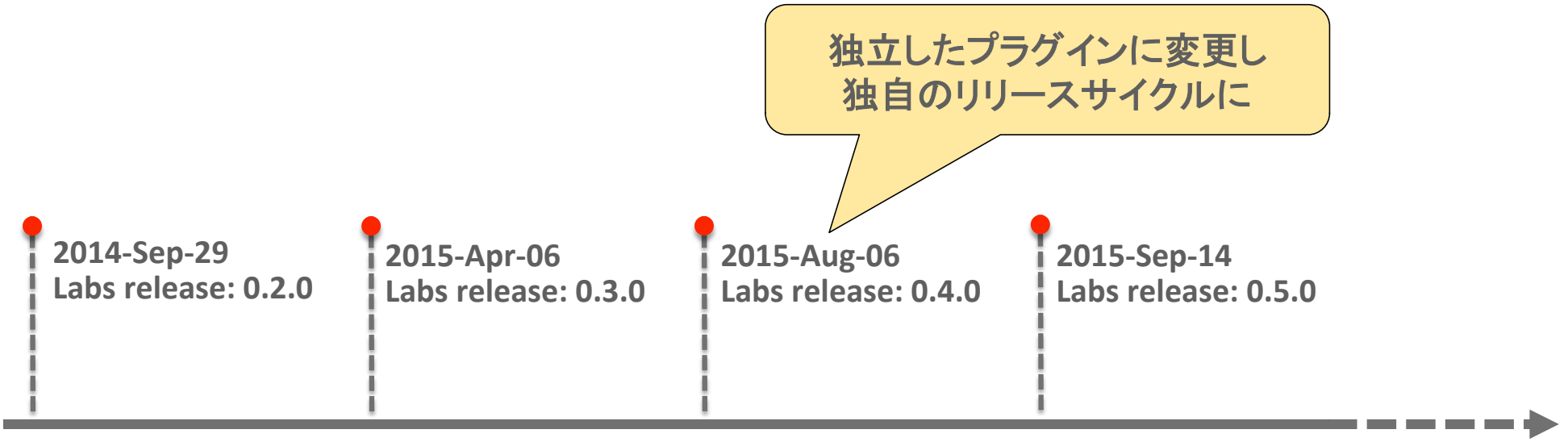
- マルチマスター型構成
- グループメンバーの自動管理と障害検知
- サーバのフェールオーバー不要
- 自動再構成
- 単一障害点無し
- シェアードナッシング型
- 「ステートマシン」レプリケーション
- InnoDB互換、特殊なハードウェア不要

MySQLグループレプリケーションプラグイン

- クラウドベースの環境にて、「Elastic」な要件に適合
  - API経由でMySQLサーバのコア機能と統合
  - **GTIDおよび行ベースレプリケーションとの統合**
  - **performance schemaのテーブルにて稼働監視**
  - 「Elastic」かつ自動復旧型: サーバの追加や削除を可能な限り自動化

MySQLグループレプリケーション

- コミュニティからのフィードバックをより取り入れるために labs.mysql.com でより高い頻度でリリース

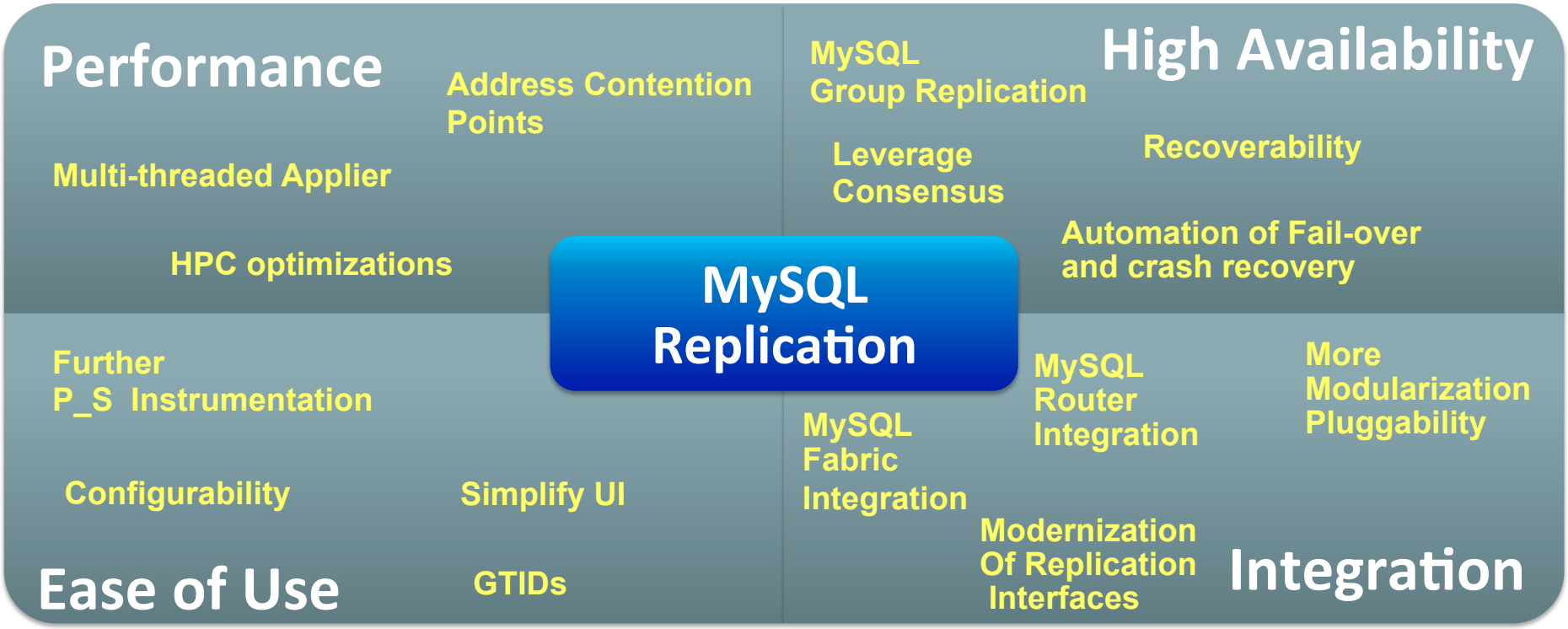


ロードマップ

What is Next?

- MySQL **Group Replication**
  - より高頻度でリリース
  - パフォーマンスの向上、安定性と使いやすさ
- MySQLレプリケーションの**使いやすさ**
  - より詳細な稼働統計情報とレプリケーション関連パフォーマンススキーマの拡張
  - よりシンプルな管理コマンド
- MySQLレプリケーションの**性能**
  - マルチスレッドスレーブの性能改善
  - 準同期レプリケーションの改善
- MySQL **Fabric**とMySQL **Router**の互換性向上

Focus



まとめ

まとめ

- MySQL 5.7の主なレプリケーション関連の改善項目:
  - 準同期レプリケーションの柔軟性向上かつ性能改善
  - バイナリログへのアクセス競合の削減
  - スキーマ内のマルチスレッドスレーブによる性能改善
  - オンラインでの設定変更: GTID, レプリケーションフィルタ, 各種設定項目
  - 監視項目の増加とパフォーマンススキーマ
  - マルチソースレプリケーションによるより柔軟な構成
  - 全体的な改良: トランザクションのリトライ、コミット順序の維持、XAサポートなど
- Labリリースにてさらなる新機能の開発: グループレプリケーションプラグインなど


ORACLE®