Replication Enhancements in MySQL 5.7

2015/04/24

MySQL Global Business Unit
Shinya Sugiyama / 杉山真也
MySQL Principal Sales Consult, MySQL Global Business Unit

ORACLE

MySQL

SAFE HARBOR STATEMENT

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。
また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。
以下の事項は、マテリアルやコード、機能を提供することをコミットメントするものではない為、
購買決定を行う際の判断材料になさらないで下さい。

オラクル製品に関して記載されている機能の開発、リリースおよび時期については、
弊社の裁量により決定されます。

MySQL 5.7 Release Candidate Available!

RC

パフォーマンス & 拡張性

- MySQL 5.6比2倍の速度
- InnoDBの機能拡張:
  Online&Bulk load オペレーション高速化
- **レプリケーションの改善**
  (multi-source, multi-threaded slaves等)
- 新しいオプティマイザコストモデル:
  greater user control & better query performance

管理性

- Performance Schema改善
- MySQL SYS Schema改善
- セキュリティの向上:
  より安全な初期化, セットアップ&管理
- NEW! JSONのSupport (now in labs)

And many more new features and enhancements... http://mysqlserverteam.com/the-mysql-5-7-7-release-candidate-is-available/

ORACLE

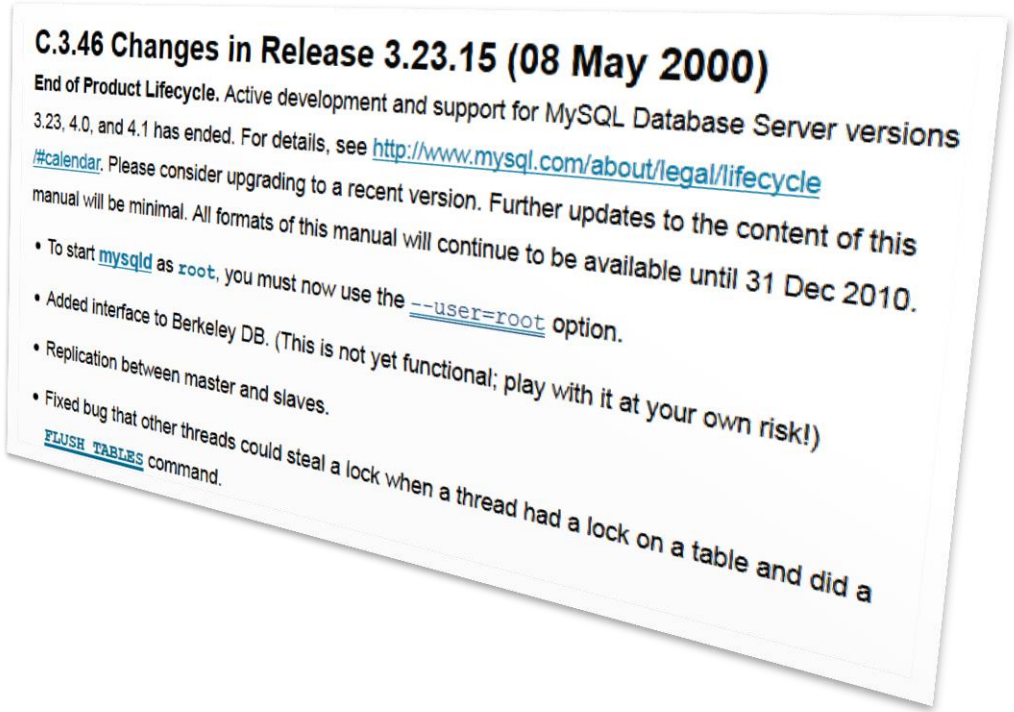
Program Agenda

1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. MySQL5.7におけるレプリケーション拡張
5. 参考情報

History of MySQL Replication

MySQLにReplication機能が実装されてから約15年が経ち、これまで多くの修正や機能改善が行われてきました。
昨年は、MySQLのReplicationの運用をサポートするMySQL UtilitiesやReplicationをベースとしたフレームワークで、更なる高可用性とシャーディングをサポートするMySQL Fabricも2014年5月に追加されました。
本日は、新たに次期メジャーバージョンのMySQL5.7に組み込まれる、レプリケーションにおいての機能拡張を更にご紹介させて頂きたいと思います。

MySQL Replication



C.3.46 Changes in Release 3.23.15 (08 May 2000)

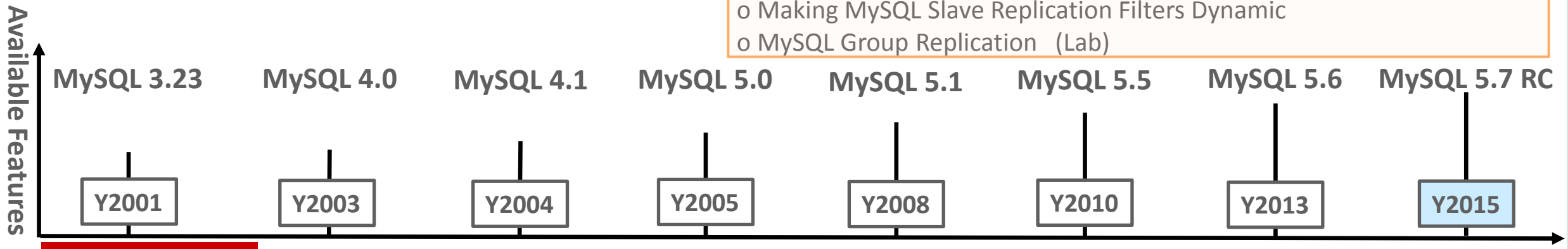
End of Product Lifecycle. Active development and support for MySQL Database Server versions 3.23, 4.0, and 4.1 has ended. For details, see http://www.mysql.com/about/legal/lifecycle #calendar. Please consider upgrading to a recent version. Further updates to the content of this manual will be minimal. All formats of this manual will continue to be available until 31 Dec 2010.

- To start mysqld as root, you must now use the --user=root option.
- Added interface to Berkeley DB. (This is not yet functional; play with it at your own risk!)
- Replication between master and slaves.
- Fixed bug that other threads could steal a lock when a thread had a lock on a table and did a FLUSH TABLES command.

抜粋：http://dev.mysql.com/doc/refman/4.1/en/news-3-23-15.html

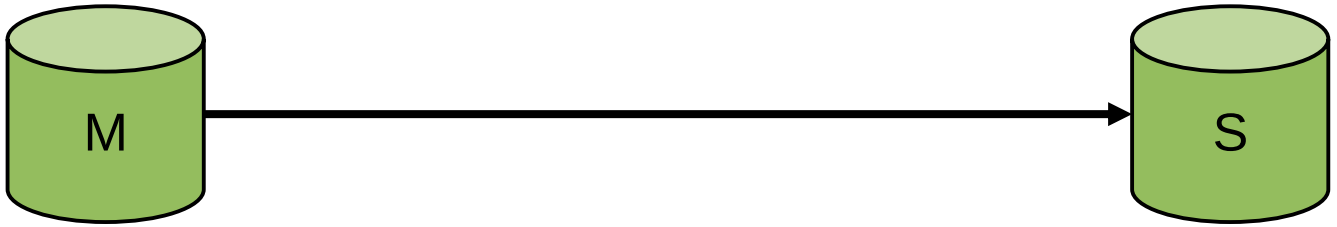
MySQL 3.23 - Generally Available, January 2001
o MySQL Replication came to be (3.23.15 – May 2000).
o Replication filters
MySQL 4.0 - Generally Available, March 2003
o Two Replication Threads instead of just one.
o Slave Relay logs.
MySQL 4.1 - Generally Available, October 2004
o Replication over SSL.
o Disk synchronization options for binary log.
MySQL 5.0 - Generally Available, October 2005
o Replication of Stored Routines and Triggers.
o Slave retries transactions on transient errors.
MySQL 5.1 - Generally Available, November 2008
o Row-based Replication (RBR).

MySQL 5.5 - Generally Available, December 2010
o Semi-sync replication.
o Replication Heartbeats.
o RBR type conversion.
MySQL 5.6 - Generally Available, February 2013
o Crash-safe Slaves.
o Global Transaction IDs.
o Replication Event Checksums.
o Binary Log Group Commit.
o Multi-threaded Slaves.
o RBR enhanced.
o MySQL Utilities 1.3, GA on August 2013
MySQL 5.7.7 RC, April 2015
o Multi-Threaded Inter-Transactional Replication
o Lossless Semi-Synchronous Replication
o Multi-Source Replication
o Enabling GTID Without Downtime
o Making MySQL Slave Replication Filters Dynamic
o MySQL Group Replication (Lab)



レプリケーション基本機能

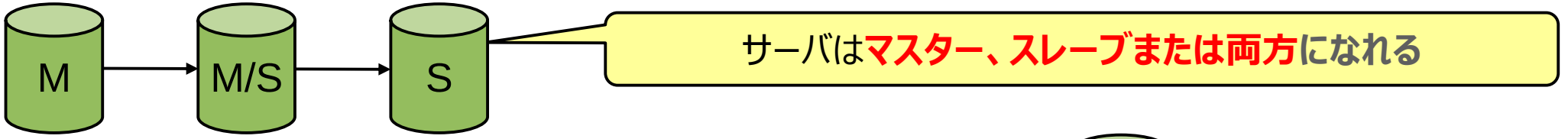
- データの複製(レプリカ)を別のサーバーに持てる機能
- MySQLの標準機能で、多数のWebサイト等で利用されている
  - シンプルな設定で利用可能
  - マスター→スレーブ 構成

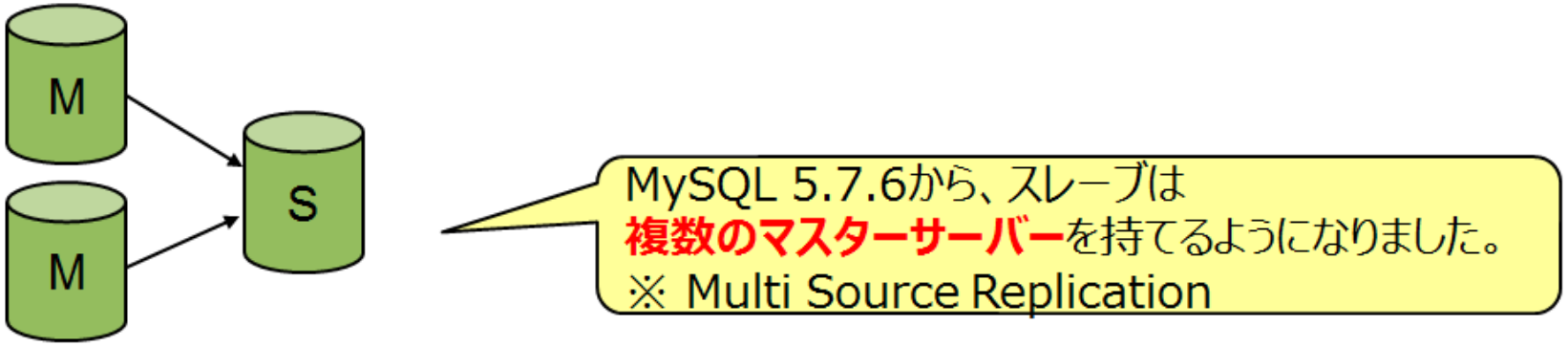


[マスターサーバー]
データを変更
変更内容をスレーブに転送

[スレーブサーバー]
マスターでの変更内容を受け取る
変更内容をデーターベースに反映

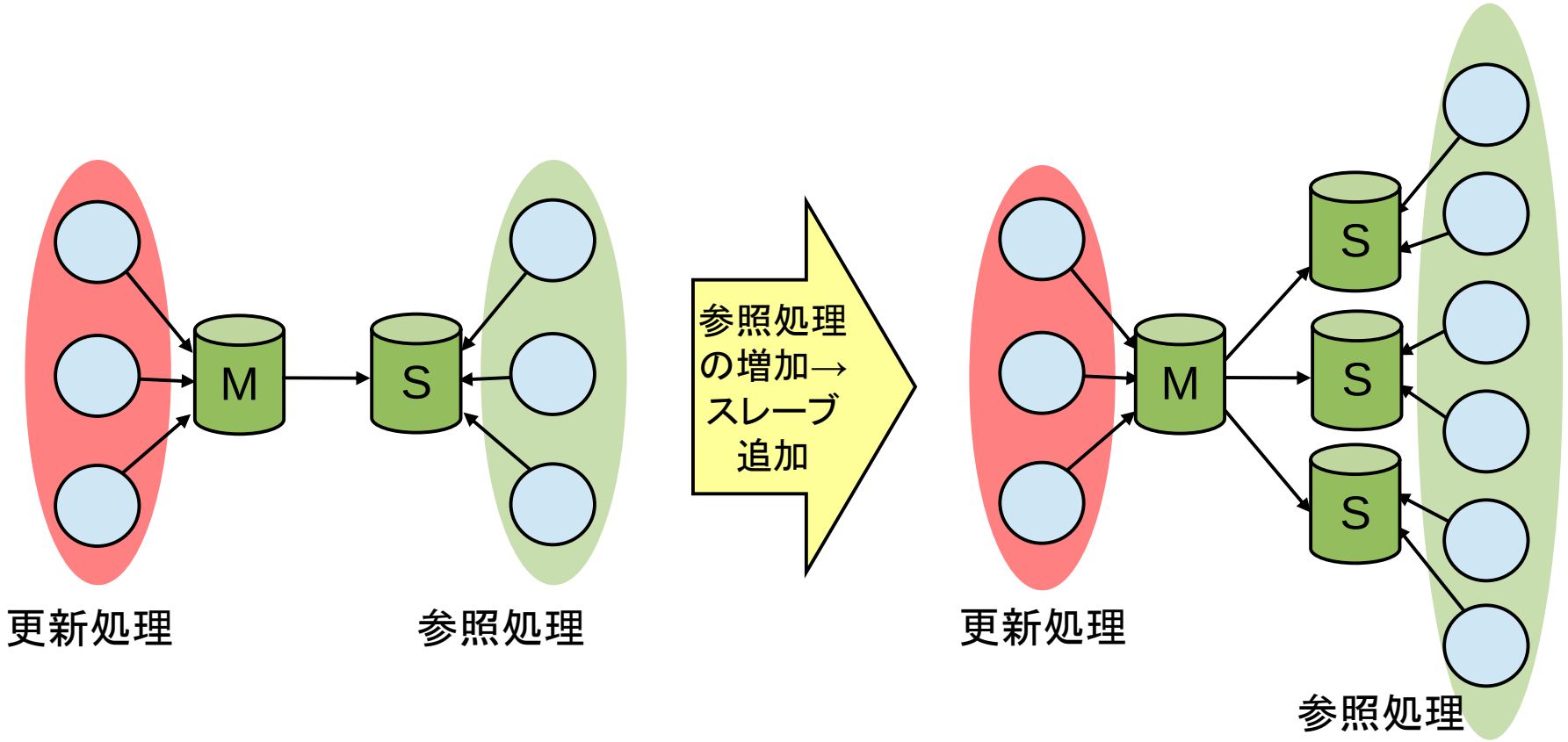
MySQLレプリケーショントポロジー





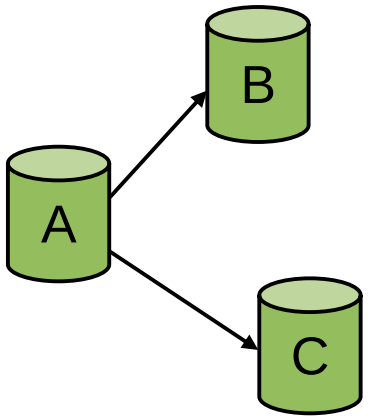
レプリケーションの利点：参照性能の向上

- 参照処理の負荷が高い場合は、スレーブサーバーを追加することで、負荷分散による性能向上が実現できる



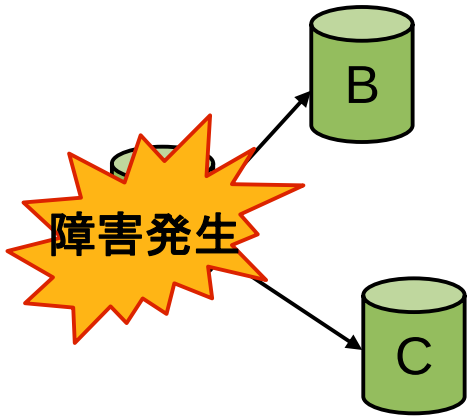
レプリケーションの利点：高可用性構成の実現

- マスターの障害時に、スレーブをマスターに昇格することで高可用性を実現可能



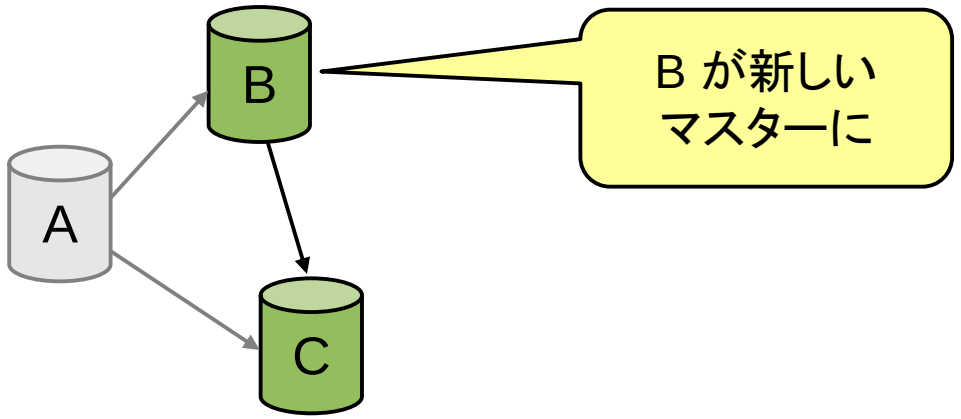
レプリケーションの利点：高可用性構成の実現

- マスターの障害時に、スレーブをマスターに昇格することで高可用性を実現可能



レプリケーションの利点：高可用性構成の実現

- マスターの障害時に、スレーブをマスターに昇格することで高可用性を実現可能



レプリケーションの利点：地理的冗長性の実現

- 地理的に離れた場所に、災害対策サイトを構築可能

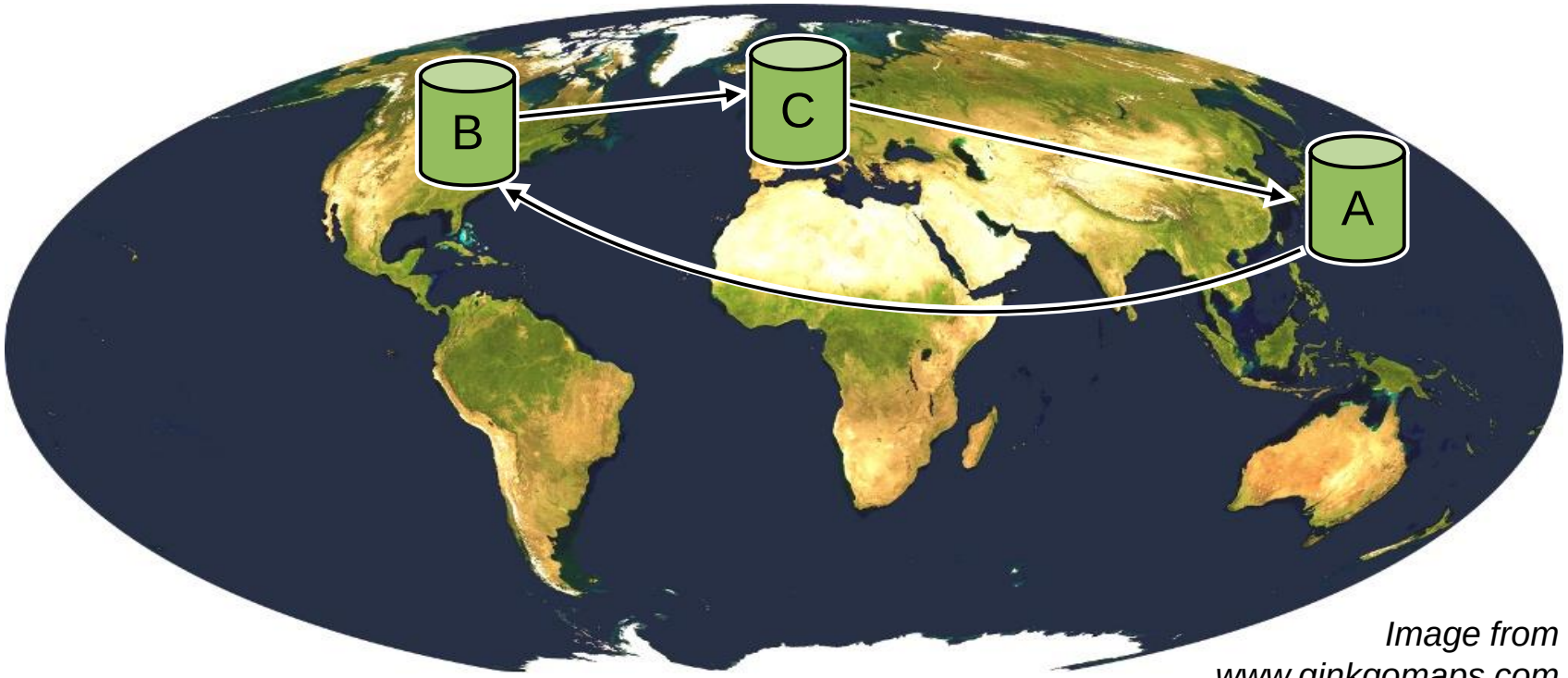
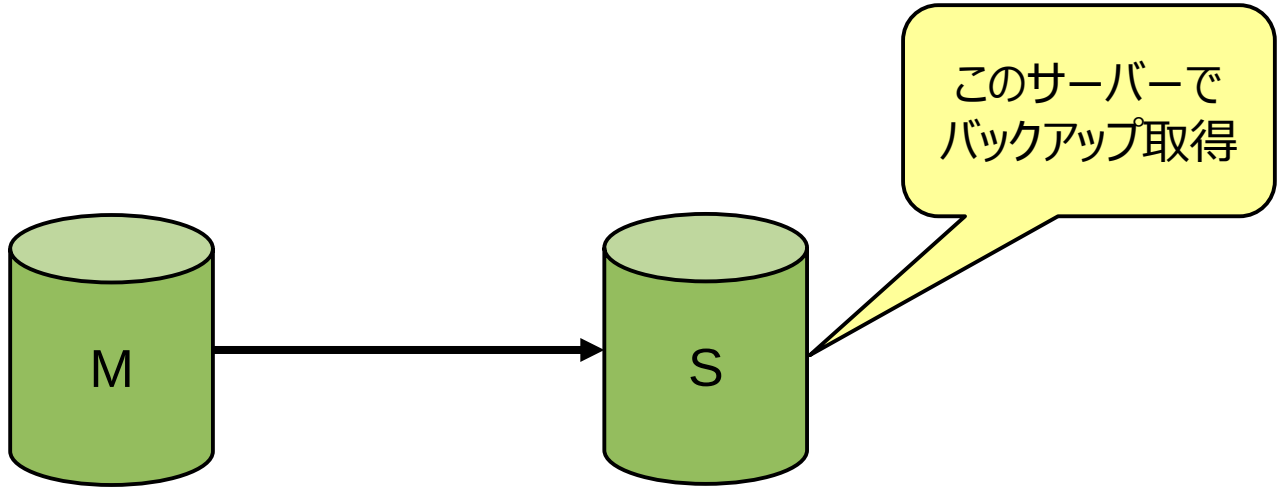


Image from www.ginkgomaps.com

レプリケーションの利点：バックアップサーバーとしての利用

- スレーブサーバーでバックアップを取得することで、
  マスターサーバーに影響を与えずにバックアップ取得可能
  - 例）マスターサーバーは常時稼働させつつ、スレーブサーバーでDBを停止して
    コールドバックアップを取得する事も可能。

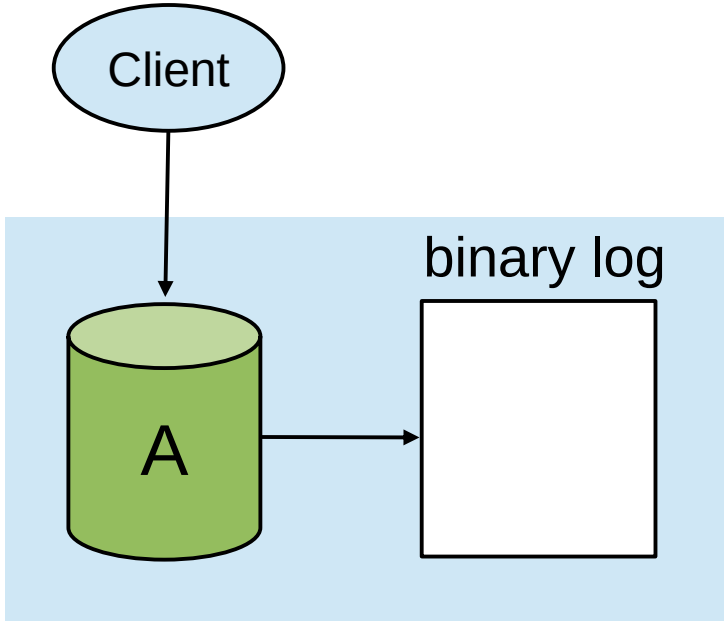


Program Agenda

1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. MySQL5.7におけるレプリケーション拡張
5. 参考情報

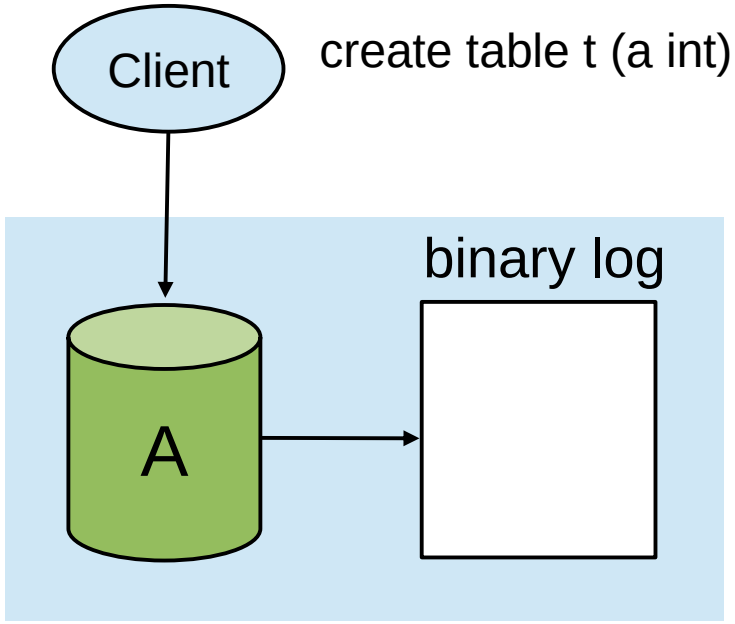
レプリケーションの仕組み

- マスターサーバーでの全ての変更点をバイナリログに記録する



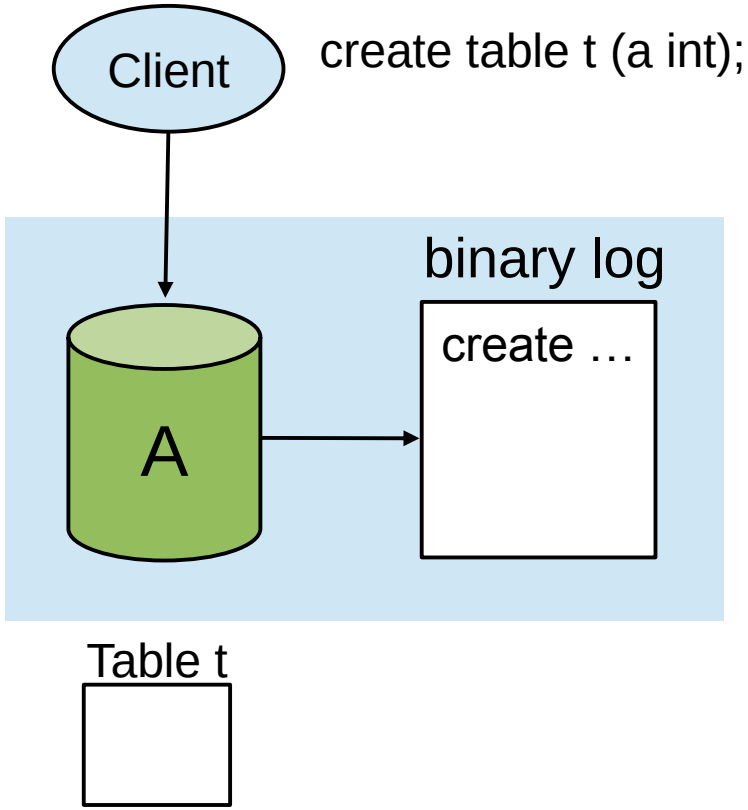
レプリケーションの仕組み

- マスターサーバーでの全ての変更点をバイナリログに記録する



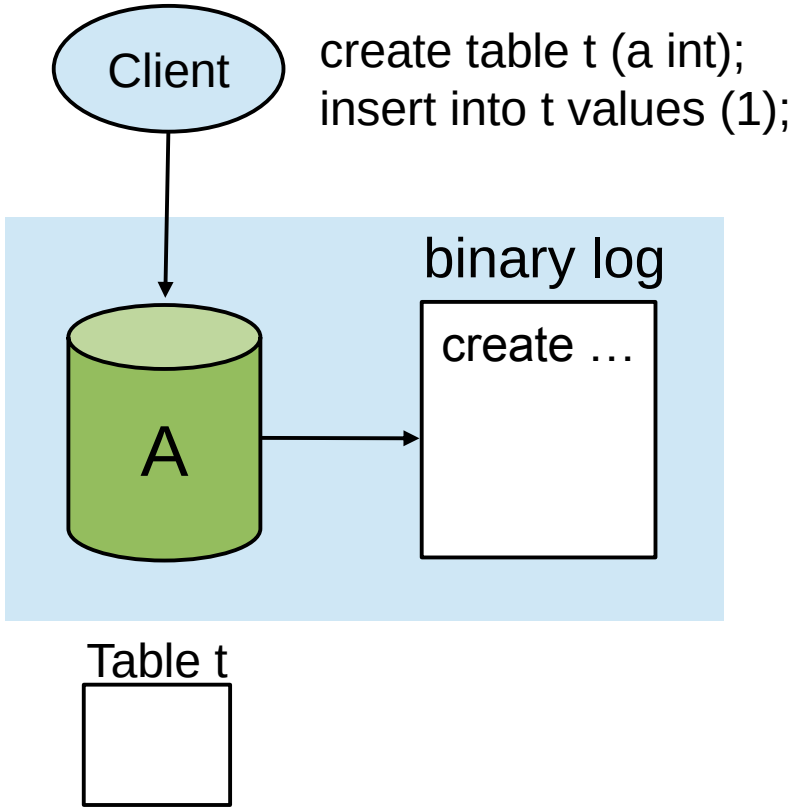
レプリケーションの仕組み

- マスターサーバーでの全ての変更点をバイナリログに記録する



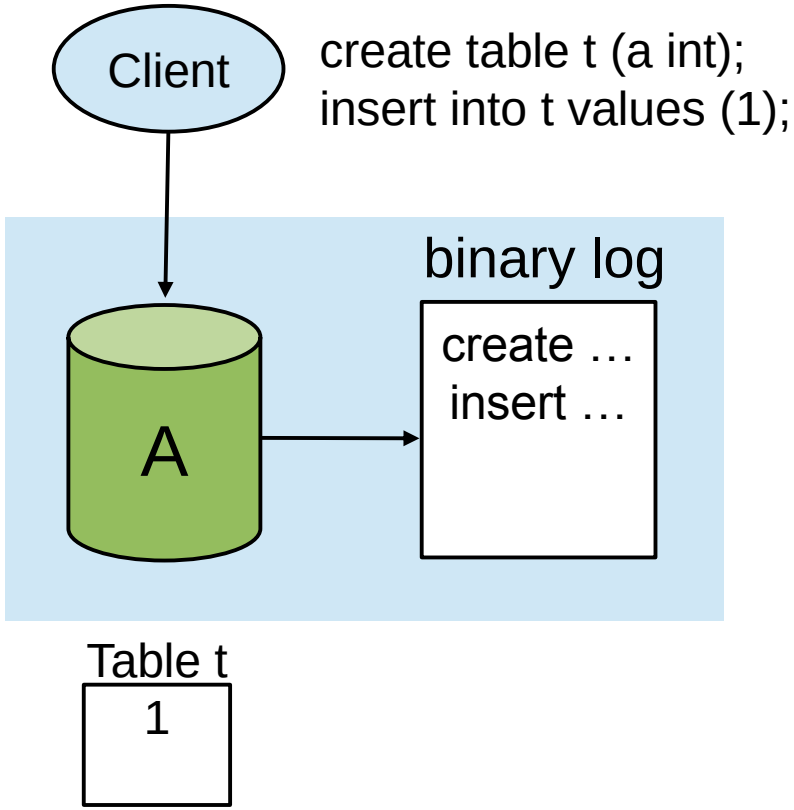
レプリケーションの仕組み

- マスターサーバーでの全ての変更点をバイナリログに記録する



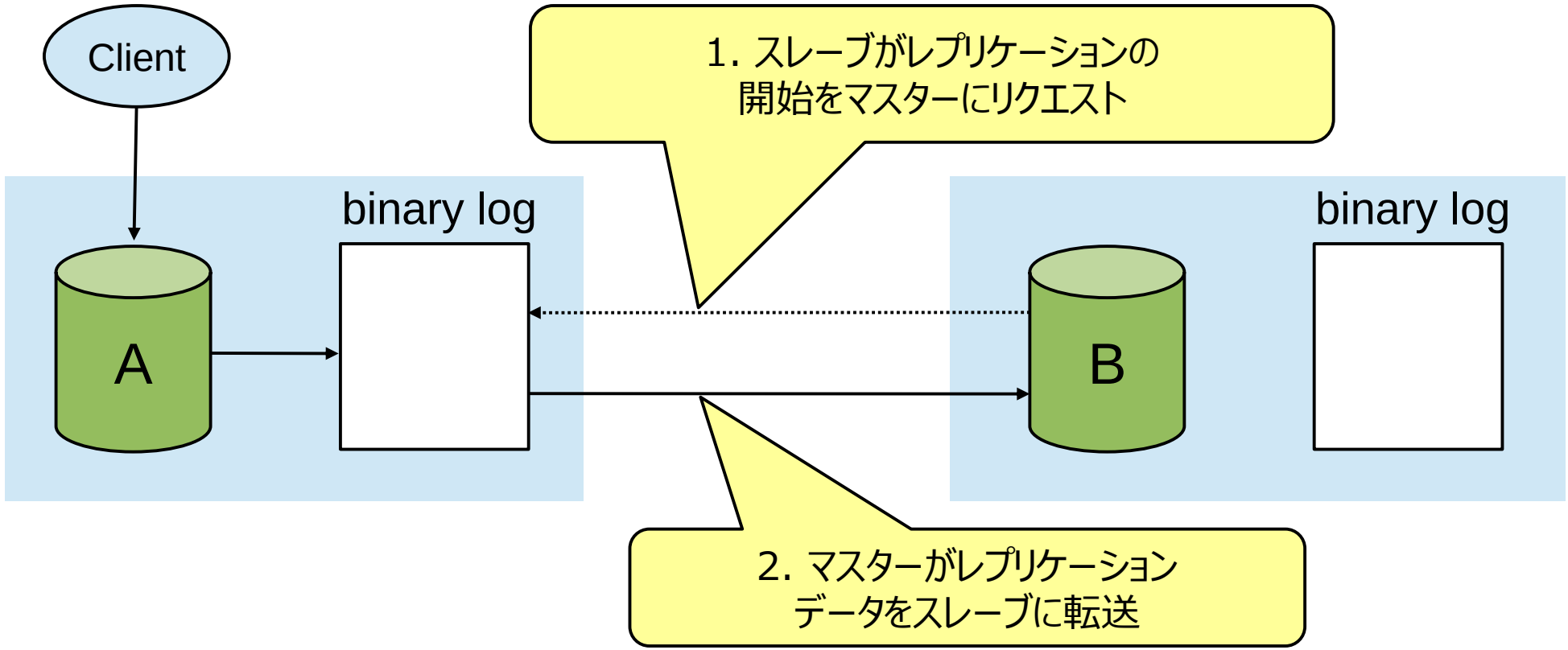
レプリケーションの仕組み

- マスターサーバーでの全ての変更点をバイナリログに記録する



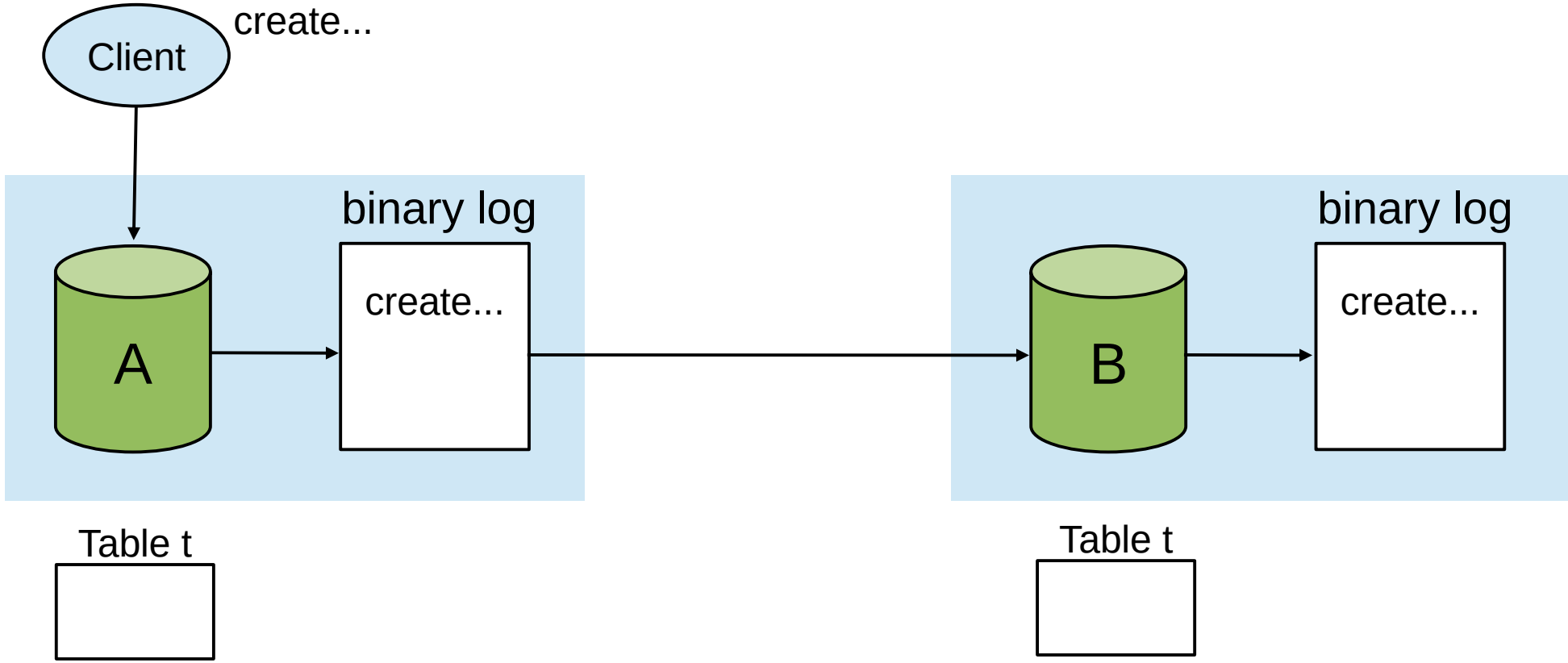
レプリケーションの仕組み

- スレーブからレプリケーション開始



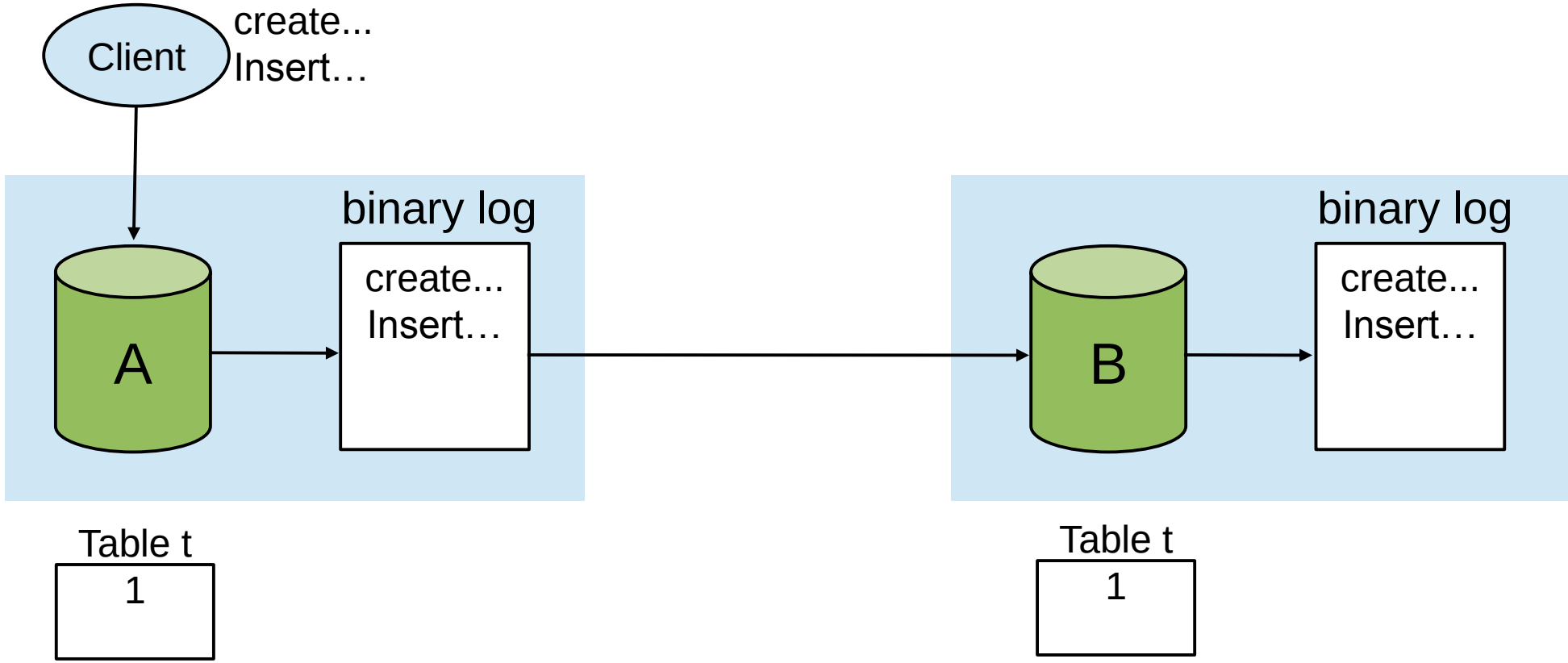
レプリケーションの仕組み

- バイナリログの内容をスレーブに転送し、実行



レプリケーションの仕組み

- バイナリログの内容をスレーブに転送し、実行



スレーブ上に存在するファイル、スレッド

- ファイル
  - リレーログファイル：マスターから受け取った変更点を記録したファイル
  - master.info：マスターへの接続に必要な情報や、読み取りを開始する
    バイナリログの位置情報(バイナリログファイル名とポジション)が記録されている
  - relay-log.info：リレーログをどこまで適用したかを記録している

※ MySQL 5.6以降では、ファイルの代わりにテーブルでレプリケーションの管理を行う事が可能です。
5.7以降のマルチソースレプリケーションを利用する場合は、テーブルでの管理を選択する必要があります。

```
-rw-rw----. 1 mysql mysql 139  4月 24 04:27 master.info
-rw-rw----. 1 mysql mysql  71  4月 24 04:27 relay-log.info
```

```
+--------------------------+
| Tables_in_mysql (slave%) |
+--------------------------+
| slave_master_info        |
| slave_relay_log_info     |
| slave_worker_info        |
+--------------------------+
```

- スレッド
  - I/Oスレッド：マスターから受診したバイナリログをリレーログファイルとして保存
  - SQLスレッド：リレーログファイル内の更新内容をDBへ反映する

Program Agenda

1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. MySQL5.7におけるレプリケーション拡張
5. 参考情報

レプリケーションの種類

- バイナリログの記録方式による違い
  - STATEMENT：文(SQL文)
  - ROW：行
  - MIXED：文と行が混在
- 同期方式による違い
  - 非同期：変更点を非同期で転送
  - 準同期：変更点を同期で転送し、非同期でDBに反映
- GTIDを使用する/しない による違い
  - GTIDを使用しない：従来からあるレプリケーションモード
  - GTIDを使用する：MySQL 5.6で追加されたレプリケーションモード

バイナリログの記録方式のよる違い

MySQL 5.7.7以前のバージョンでは、**statement-base**フォーマットがディフォルトです。
MySQL 5.7.7以降は, **row-based** フォーマットがディフォルト値になりました。

フォーマット	説明	バイナリログサイズ	Non-deterministic	スレーブでトリガーが動作するか？
SBR (Statement Based Replication)	SQL文がそのまま バイナリログに記録される	小	×	動作する
RBR (Row Based Replication)	更新されたデータそのものが記録される	大	○	動作しない (トリガーにより変更された行の変更は伝搬される)
MBR (Mixed Based Replication)	SBRとRBRを状況に応じて切り換える	小	○	SBRの場合動作する RBRの場合動作しない

Non-deterministicとは？

- 非決定性なSQL文＝実行するたびに結果が変わる可能性があるSQL文
  - UUID()、UUID_SHORT()
  - USER()
  - FOUND_ROWS()
  - LOAD_FILE()
  - SYSDATE()
  - GET_LOCK()、RELEASE_LOCK()
  - IS_FREE_LOCK()、IS_USED_LOCK()
  - MASTER_POS_WAIT()
  - SLEEP()
  - VERSION()
  - ソートなしのLIMIT句
  - UDF、非決定性のストアドプロシージャ/ファンクション
  - INFORMATION_SCHEMAの参照
  - READ-COMMITTED/READ-UNCOMMITTED

```
root@localhost [REPLI]> insert into T_REPLI01(n_time,s_time) values(NOW(3),SYSDATE(6));
Query OK, 1 row affected (0.00 sec)

root@localhost [REPLI]> select * from T_REPLI01;
+----+-------------------------+----------------------------+
| id | n_time                  | s_time                     |
+----+-------------------------+----------------------------+
|  1 | 2015-04-23 05:31:00.427 | 2015-04-23 05:31:00.427164 |
|  2 | 2015-04-23 05:31:39.105 | 2015-04-23 05:31:39.105423 |
|  3 | 2015-04-23 05:32:06.337 | 2015-04-23 05:32:06.337917 |
|  4 | 2015-04-23 05:32:19.974 | 2015-04-23 05:32:19.975044 |
|  5 | 2015-04-23 05:32:23.685 | 2015-04-23 05:32:23.685741 |
+----+-------------------------+----------------------------+
```

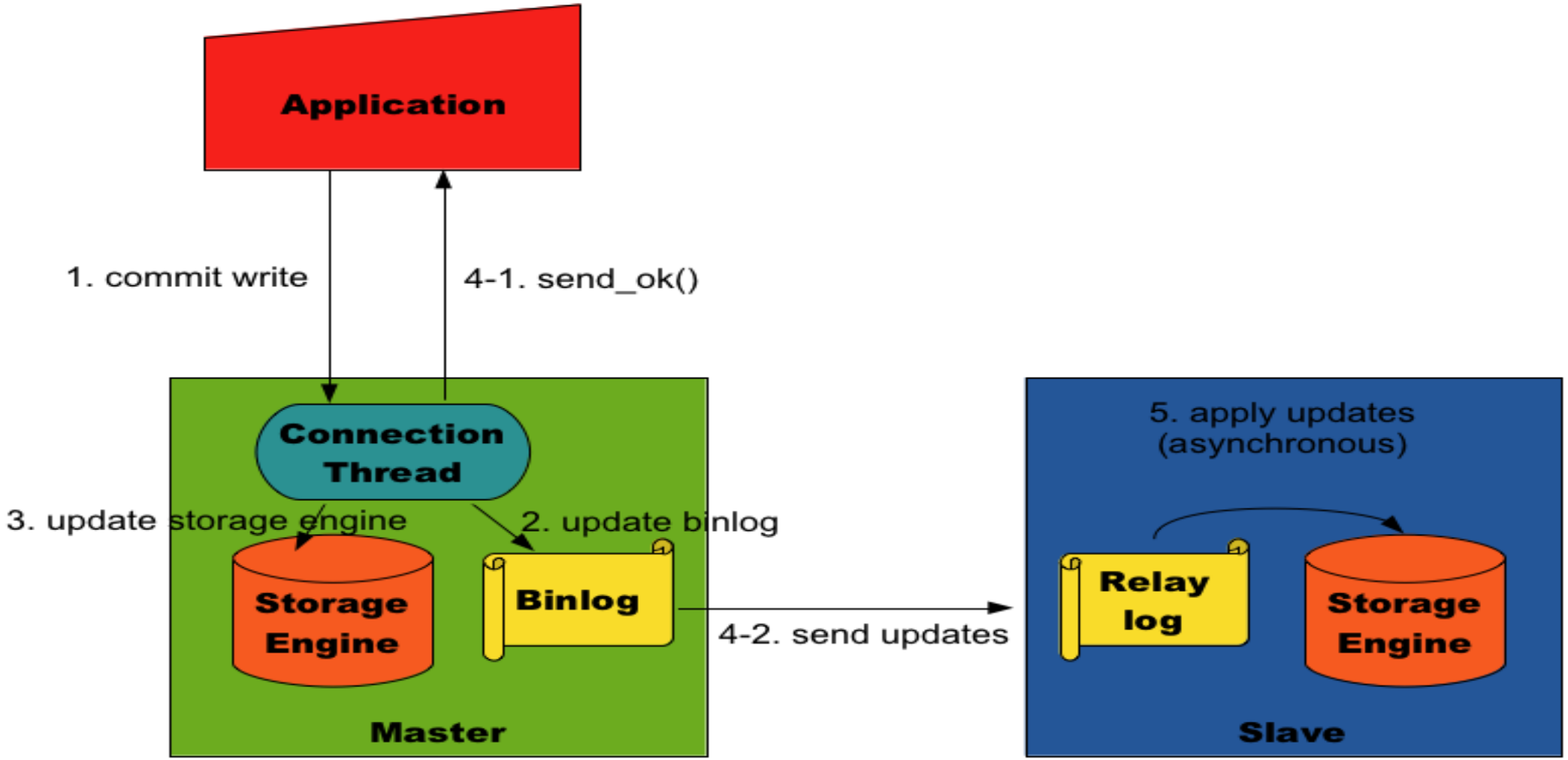
同期方式による違い

- 非同期(デフォルト)
  - 変更点を非同期で転送
  - メリット：準同期よりもマスターサーバーの更新処理のパフォーマンスがいい
  - デメリット：マスターサーバーに障害が発生した場合、障害発生直前の更新内容がスレーブに伝搬されていない可能性がある
- 準同期(MySQL 5.5から追加された機能)
  - 変更点を同期で転送し、非同期でDBに反映
  - メリット：マスターサーバーに障害が発生した場合、障害発生直前の更新内容もスレーブに伝搬されている
  - デメリット：非同期よりもマスターサーバーの更新処理のパフォーマンスが悪い

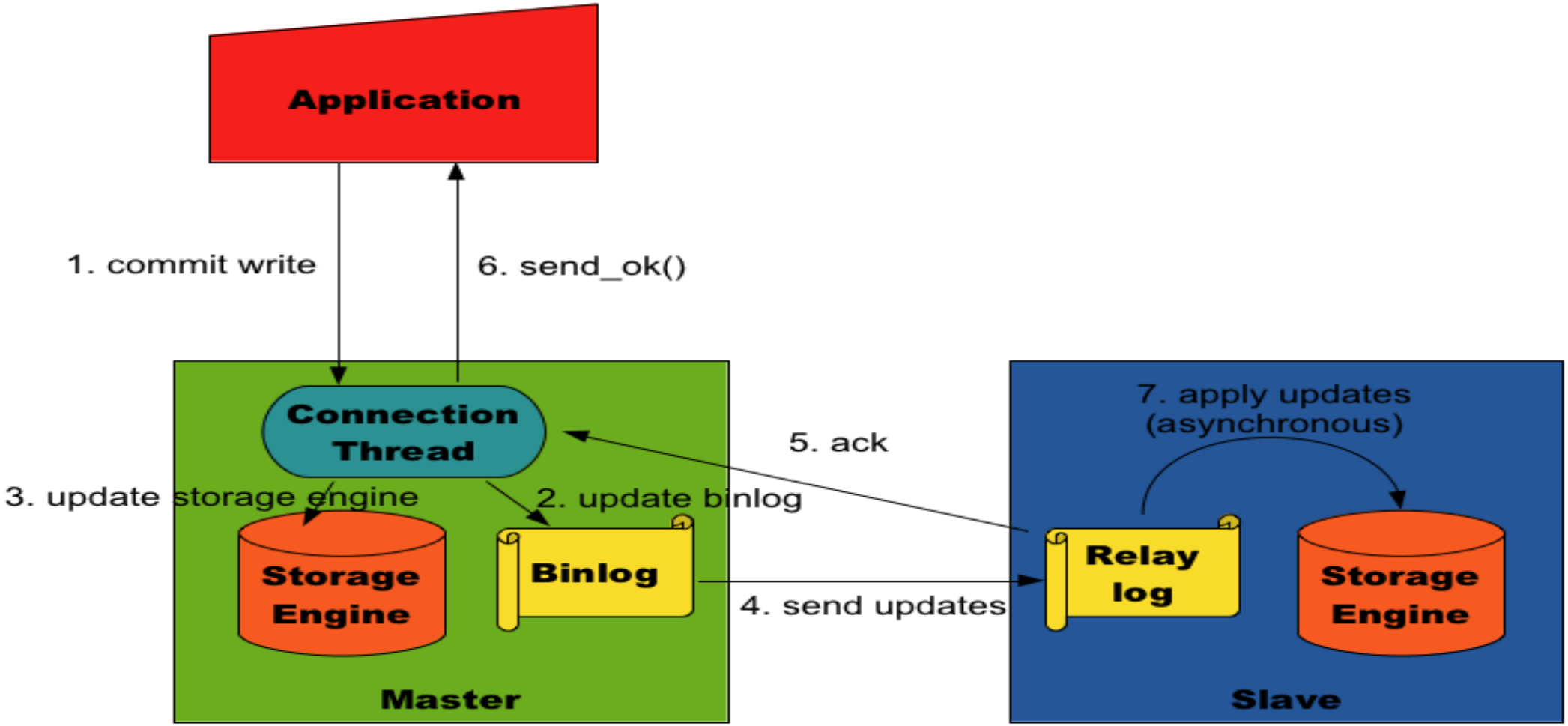
```
mysql> SET rpl_semi_sync_master_wait_point= [AFTER_SYNC|AFTER_COMMIT]
mysql> SET GLOBAL rpl_semi_sync_master_wait_for_slave_count= N [Default = 1]
```

※ MySQL5.7.2で準同期が改良され、データ同期時にマスター障害が発生してもデータ損失が発生しない、Loss-less 準同期レプリケーションに改良されました。 [AFTER_SYNC = Loss Less]

非同期レプリケーション

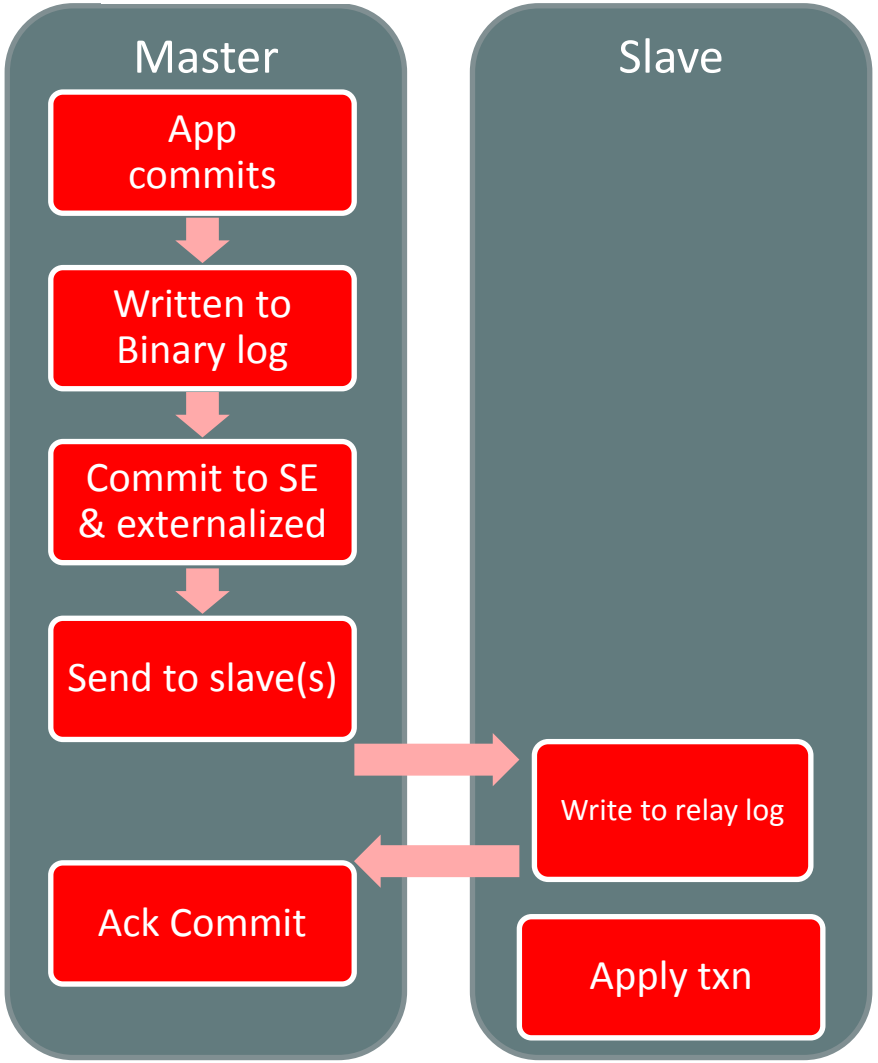


準同期レプリケーション



準同期レプリケーション

MySQL 5.6 [AFTER_COMMIT]

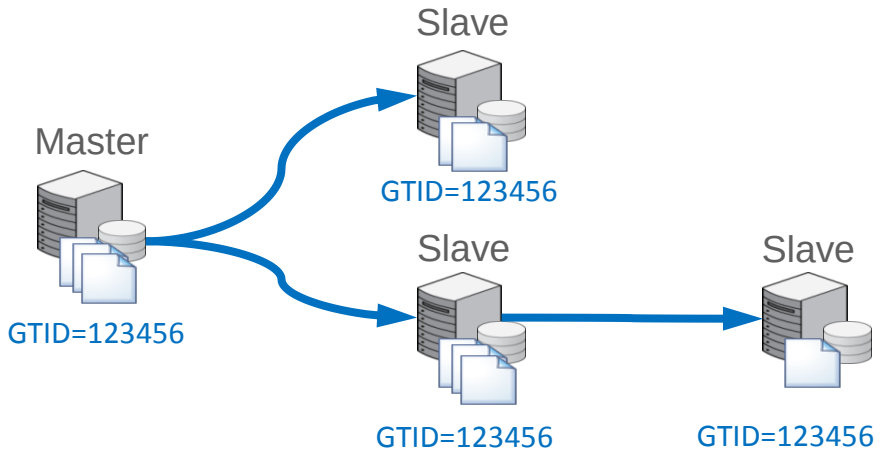


MySQL 5.7.2 “Lossless” [AFTER_SYNC]

参照 : Loss-less Semi-Synchronous Replication on MySQL 5.7.2

GTIDを使用する/しないによる違い

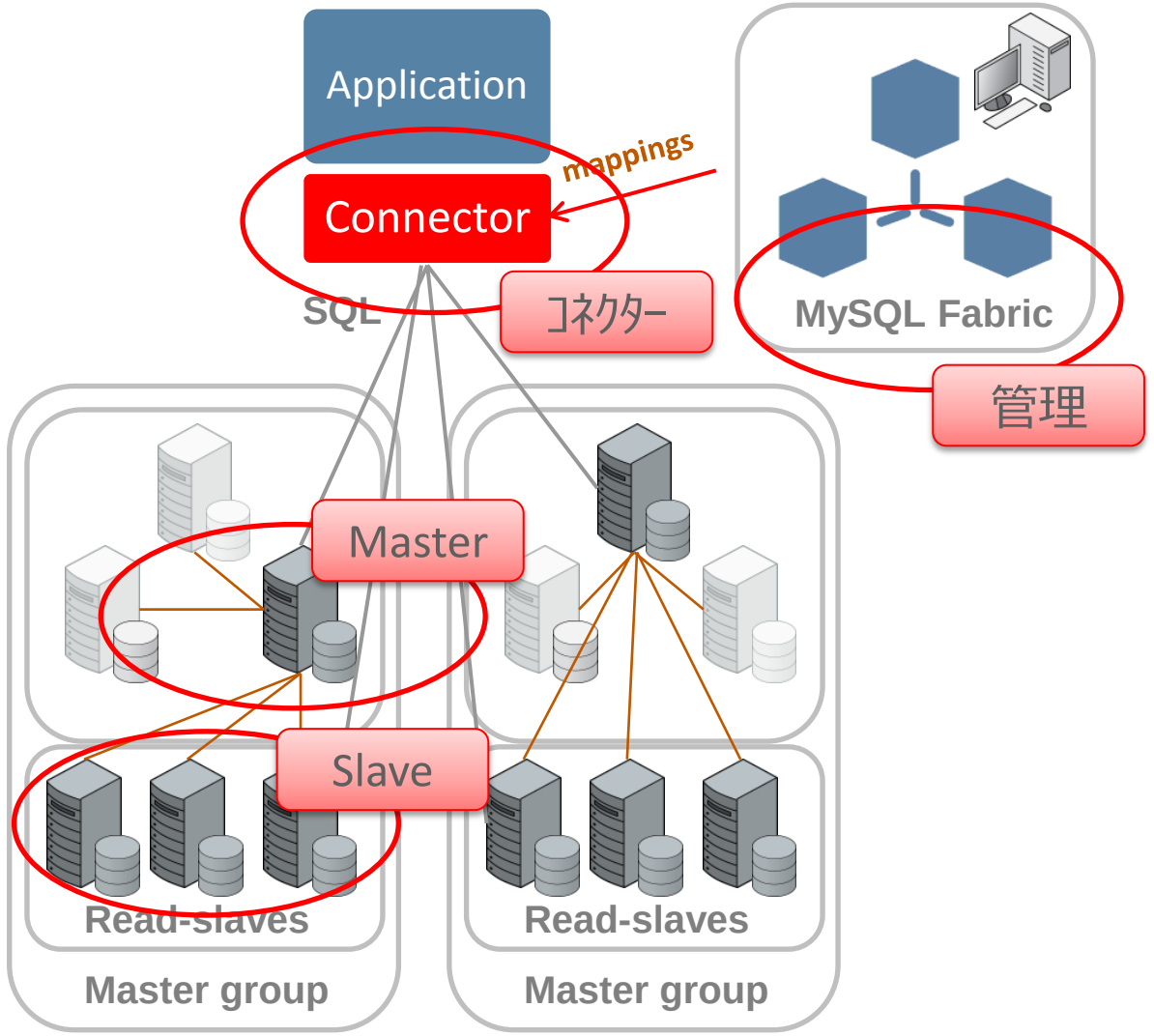
- GTID(グローバルトランザクションID)
  - MySQL 5.6で追加された機能
  - 複数台のレプリケーション環境でも容易にトランザクションの追跡/比較が可能
    - トランザクションをグローバルで(レプリケーションを構成するMySQLサーバー全てにおいて)一意に識別できる識別子をバイナリログに記録
  - 使用する場合は、レプリケーションの運用方法が従来の方式とは変わる
    - レプリケーション開始時にポジションを自動認識
    - フェイルオーバーの為に、最も最新のスレーブを自動認識
  - 多段構成のレプリケーションが容易に
  - MySQL Utilitiesでのレプリケーション管理,
    MySQL Fabricフレームワークは GTIDでの運用が必要



レプリケーション統合型フレームワーク (MySQL Fabric)

- データ・シャーディングによる拡張性を実現する事が可能
- レプリケーション自動フェールオーバーで、高可用性を実現
- OpenStack Novaとの連携
- MySQL Utilitiesの一部として提供 (2014-05-27　ver.1.4.3～)
  - GTIDモードによるレプリケーション機能を活用している (MySQL 5.6.5以降で使用可能)

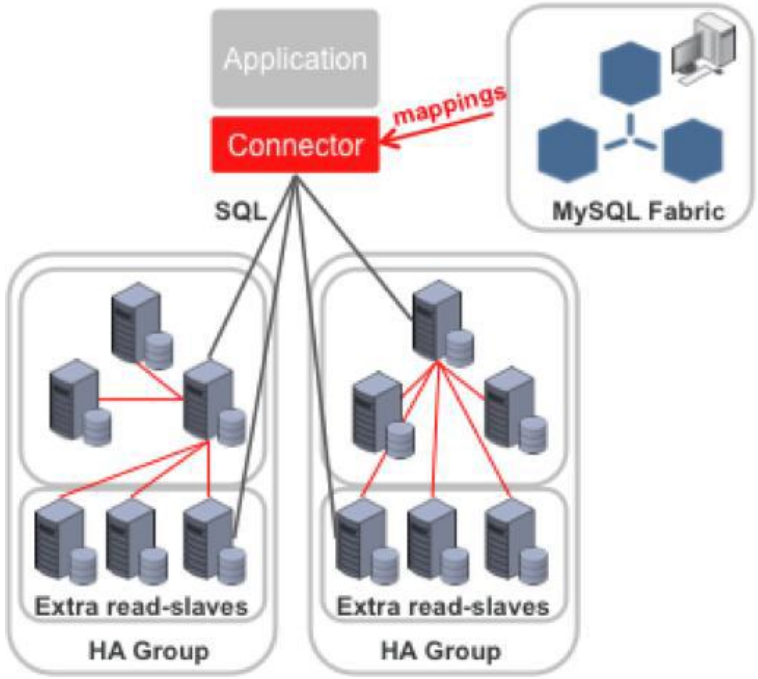
参照： MySQL Fabric の特徴と利点
http://www-jp.mysql.com/products/enterprise/fabric/features.html



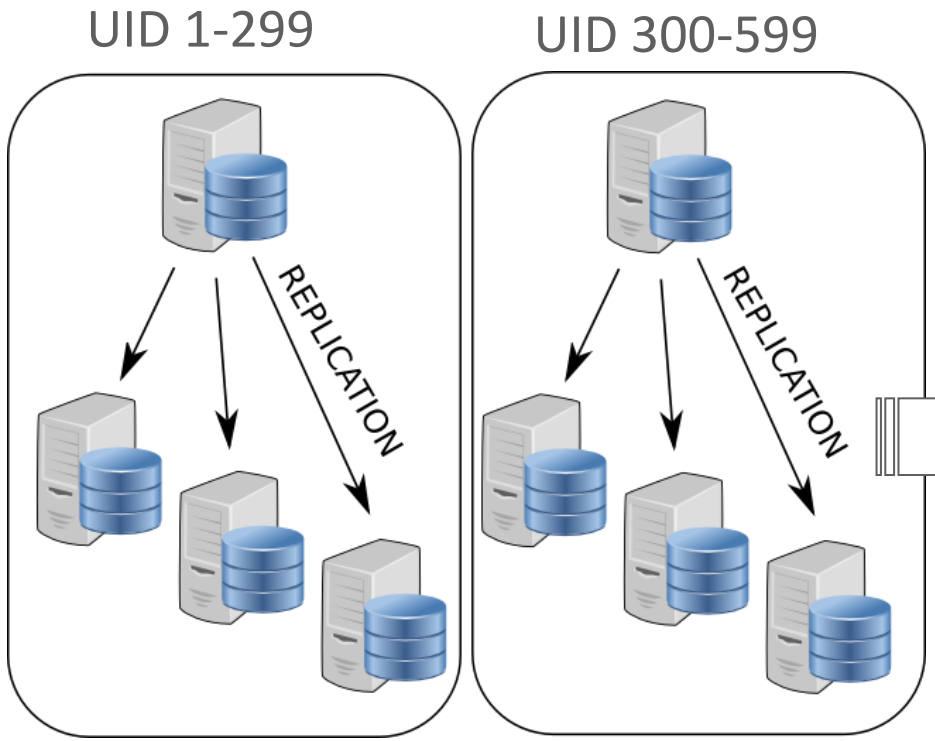
レプリケーションの書き込み拡張 (MySQL Fabric)

書き込みスケーラビリティ
大規模なデータセット
性能改善

より多くの書き込みを処理することが可能
大き過ぎるデータベース/単一サーバーに収まらないデータ
小さなインデックスサイズ/ワーキングセットに分割



Single Master in each HA Group.




```
mysql> select * from shard_ranges;
+-----------------+-------------+----------+
| shard_mapping_id | lower_bound | shard_id |
+-----------------+-------------+----------+
|                1 | 1           |        1 |
|                1 | 300         |        2 |
+-----------------+-------------+----------+
2 rows in set (0.00 sec)

mysql> select * from shard_tables;
+-----------------+---------------+-------------+
| shard_mapping_id | table_name     | column_name |
+-----------------+---------------+-------------+
|                1 | test.employees | emp_no      |
+-----------------+---------------+-------------+
1 row in set (0.00 sec)
```

キー範囲分割
UID 600-899
UID 900-1199
UID 1200-1499
……

Program Agenda

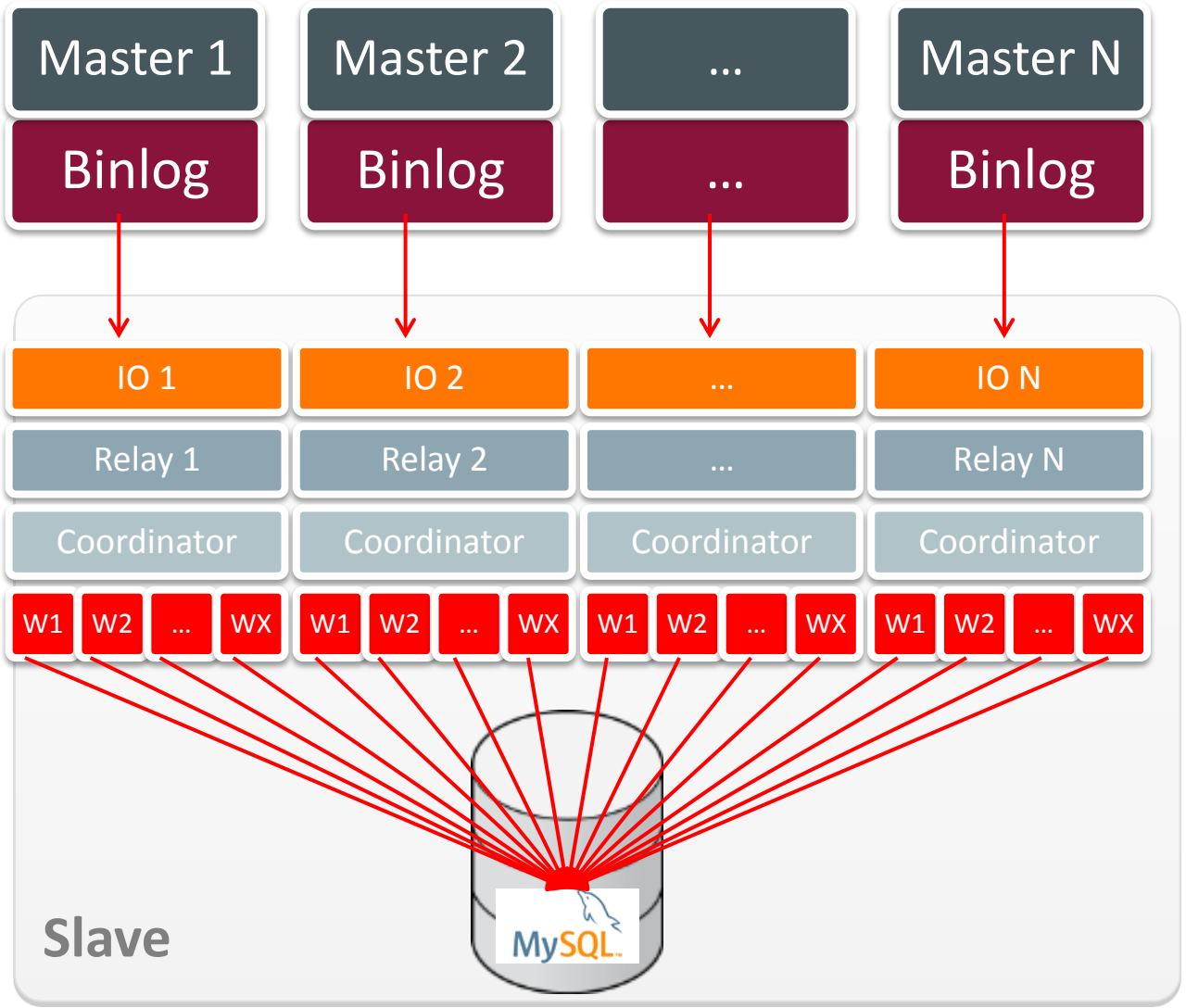
1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. MySQL5.7におけるレプリケーション拡張
5. 参考情報

MySQL 5.7におけるレプリケーションの機能拡張

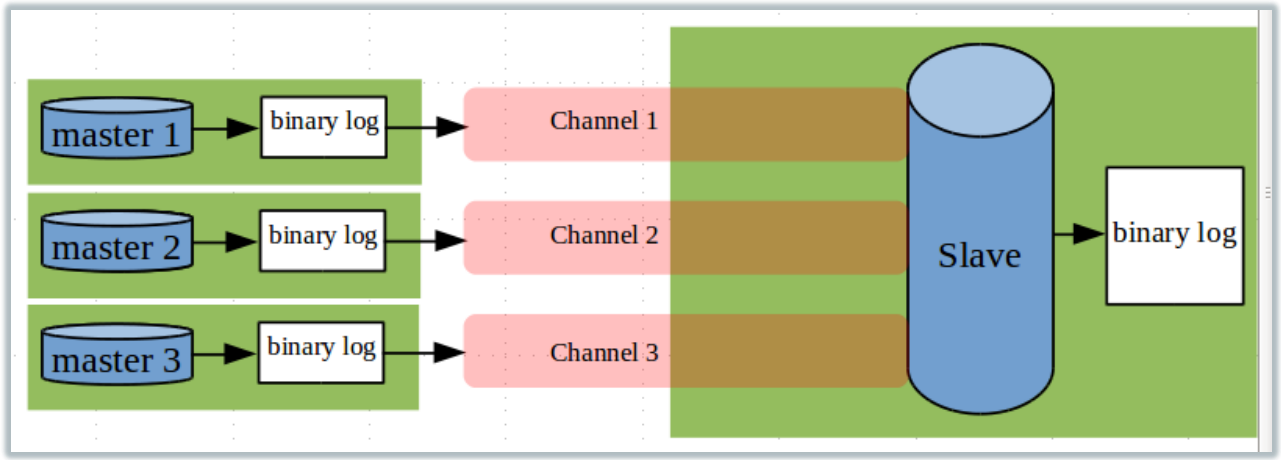
- Multi-Source Replication
- Performance enhancement of Multi-Thread Slave
- gtid_mode is now dynamic
- Making MySQL Slave Replication Filters Dynamic
- Preparing implementation of Group Replication [Labs]

MySQL 5.7: Multi-Source Replication

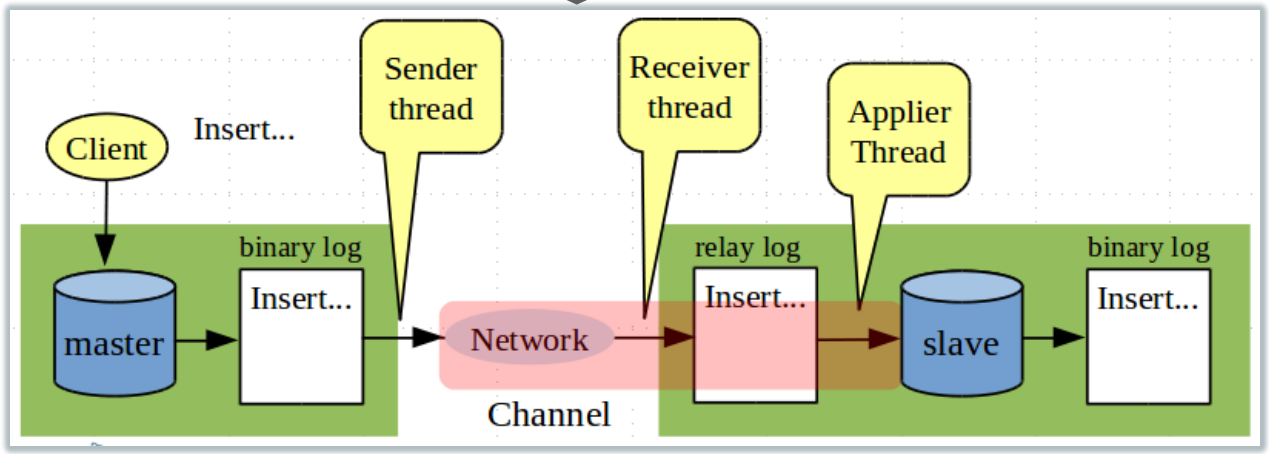
- 複数のマスターでの変更点を1台のスレーブに集約
  - 全てのシャードのデータを集約
  - より柔軟なレプリケーション構成
  - バックアップ処理を集約
- 準同期レプリケーション&改良版マルチスレッドスレーブ対応
- スレーブ側でのフィルタリングが可能



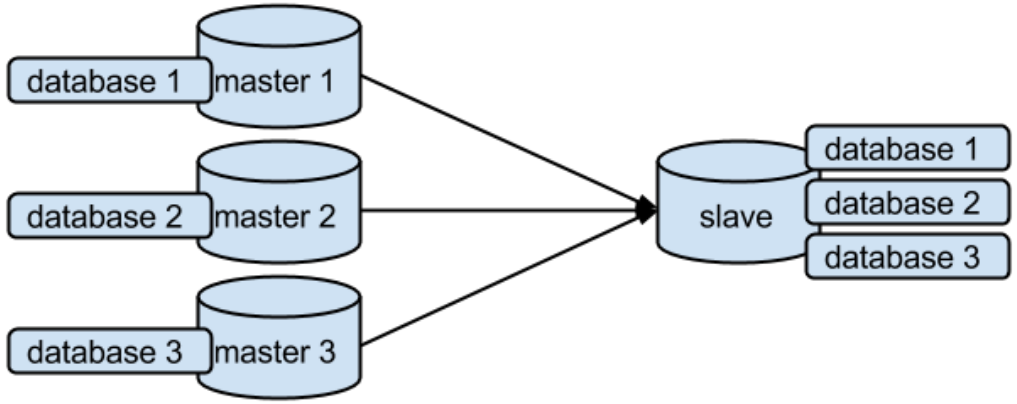
MySQL 5.7: Multi-Source Replication



各チャネルごとにレプリケーションを構成



マルチソースレプリケーションには、競合検知や解消する仕組みは組み込まれていないので、それぞれの送信元のデータが競合しないようにアプリケーションの設計を行う必要があります。
例）
- データベース分割
- テーブルのシャーディング



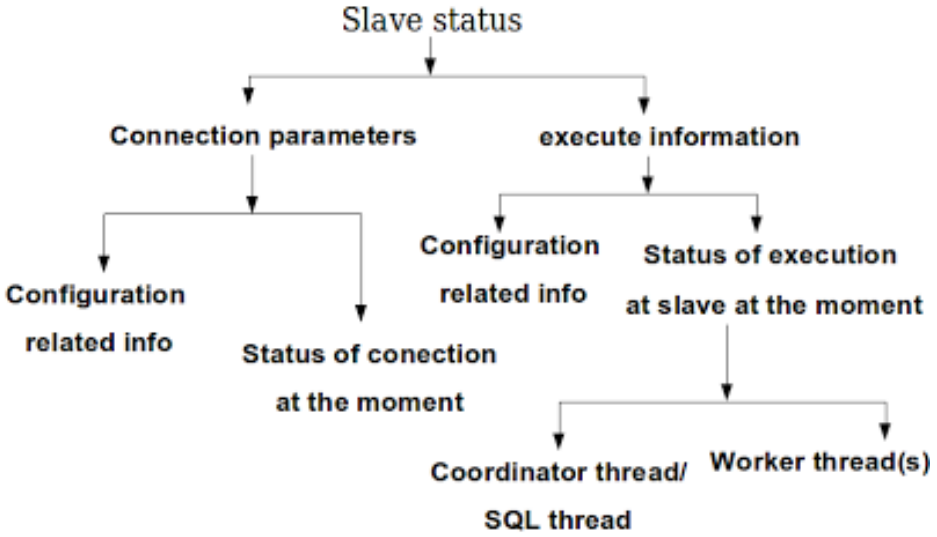
参照: 17.1.4.1 MySQL Multi-Source Replication Overview

MySQL 5.7: Multi-Source Replication

レプリケーションの状態をPerformance_Schemaにて確認可能

```
mysql> select * from  performance_schema.replication_connection_status\G
*************************** 1. row ***************************
CHANNEL_NAME                : CHANNEL1
SOURCE_UUID                 : 7cff7406-23ca-11e3-ac3e-5c260a83b12b
THREAD_ID                   : 13
SERVICE_STATE               : ON
RECEIVED_TRANSACTION_SET    : 7cff7406-23ca-11e3-ac3e-5c260a83b12b:1-4
LAST_ERROR_NUMBER           : 0
LAST_ERROR_MESSAGE          :
LAST_ERROR_TIMESTAMP        : 0000-00-00 00:00:00
LAST_HEARTBEAT_TIMESTAMP    :
COUNT_RECEIVED_HEARTBEATS   :
*************************** 2. row ***************************
CHANNEL_NAME                : CHANNEL2
...
```

One row per CHANNEL



replication_connection_status

```
root@localhost [REPLI]> desc performance_schema.replication_connection_status;
+----------------------------+---------------------------------+------+-----+---------------------+-------------------------------+
| Field                      | Type                            | Null | Key | Default             | Extra                         |
+----------------------------+---------------------------------+------+-----+---------------------+-------------------------------+
| CHANNEL_NAME               | char(64)                        | NO   |     | NULL                |                               |
| GROUP_NAME                 | char(36)                        | NO   |     | NULL                |                               |
| SOURCE_UUID                | char(36)                        | NO   |     | NULL                |                               |
| THREAD_ID                  | bigint(20) unsigned             | YES  |     | NULL                |                               |
| SERVICE_STATE              | enum('ON','OFF','CONNECTING')   | NO   |     | NULL                |                               |
| COUNT_RECEIVED_HEARTBEATS  | bigint(20) unsigned             | NO   |     | 0                   |                               |
| LAST_HEARTBEAT_TIMESTAMP   | timestamp                       | NO   |     | CURRENT_TIMESTAMP   | on update CURRENT_TIMESTAMP   |
| RECEIVED_TRANSACTION_SET   | text                            | NO   |     | NULL                |                               |
| LAST_ERROR_NUMBER          | int(11)                         | NO   |     | NULL                |                               |
| LAST_ERROR_MESSAGE         | varchar(1024)                   | NO   |     | NULL                |                               |
| LAST_ERROR_TIMESTAMP       | timestamp                       | NO   |     | 0000-00-00 00:00:00 |                               |
+----------------------------+---------------------------------+------+-----+---------------------+-------------------------------+
```

threads

```
root@localhost [performance_schema]> desc performance_schema.threads;
+--------------------+---------------------+------+-----+---------+-------+
| Field              | Type                | Null | Key | Default | Extra |
+--------------------+---------------------+------+-----+---------+-------+
| THREAD_ID          | bigint(20) unsigned | NO   |     | NULL    |       |
| NAME               | varchar(128)        | NO   |     | NULL    |       |
| TYPE               | varchar(10)         | NO   |     | NULL    |       |
| PROCESSLIST_ID     | bigint(20) unsigned | YES  |     | NULL    |       |
| PROCESSLIST_USER   | varchar(16)         | YES  |     | NULL    |       |
| PROCESSLIST_HOST   | varchar(60)         | YES  |     | NULL    |       |
| PROCESSLIST_DB     | varchar(64)         | YES  |     | NULL    |       |
| PROCESSLIST_COMMAND| varchar(16)         | YES  |     | NULL    |       |
| PROCESSLIST_TIME   | bigint(20)          | YES  |     | NULL    |       |
| PROCESSLIST_STATE  | varchar(64)         | YES  |     | NULL    |       |
| PROCESSLIST_INFO   | longtext            | YES  |     | NULL    |       |
| PARENT_THREAD_ID   | bigint(20) unsigned | YES  |     | NULL    |       |
| ROLE               | varchar(64)         | YES  |     | NULL    |       |
| INSTRUMENTED       | enum('YES','NO')    | NO   |     | NULL    |       |
+--------------------+---------------------+------+-----+---------+-------+
```

MySQL 5.7: Multi-Source Replication

Multi-Sourceレプリケーション特有のオプションや制限がありますが、基本的にはスタンダードなMySQLレプリケーションと同じです。
データを集約して集計するなどのニーズが想定される場合は、是非GA版に向けて試してみて頂ければと思います。

Masters in a multi-source replication topology can be configured to use either global transaction identifier (GTID) based replication, or binary log position-based replication. Slaves in a multi-source replication topology require TABLE based repositories. Multi-source replication is not compatible with FILE based repositories.

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

```
START SLAVE thread_types FOR CHANNEL channel;
```

```
STOP SLAVE thread_types FOR CHANNEL channel;
```

```
RESET SLAVE FOR CHANNEL channel;
```

Multi-Source Replicationチュートリアル

こちらのサイトに手順が紹介されています

17.1.4.2 Multi-Source Replication Tutorials
http://dev.mysql.com/doc/refman/5.7/en/replication-multi-source-tutorials.html

MySQL 5.7: スキーマ内マルチスレッドスレーブ

- シングルスレッドのスレーブと比較して 5倍 のスループット
  - アプリケーション側での変更不要
  - バイナリログのグループコミットでの遅延を伴う操作不要
- GTID & クラッシュセーフスレーブ利用
- Sysbench OLTP test
  - 1,000万行
  - SSD / 48 core HT / 512 GB RAM

--slave-parallel-type

1. DATABASE : (Default) Use the db partitioned MTS (1 worker per database)
2. LOGICAL_CLOCK: Use logical clock based parallelization mode.

マルチスレッドスレーブセットアップ手順

ON MASTER:

1. start master with
--binlog-max-flush-queue-time=0

```
+-----------------------------+-------+
| Variable_name               | Value |
+-----------------------------+-------+
| binlog_max_flush_queue_time | 0     |
+-----------------------------+-------+
```

※Default 値は0

ON SLAVE:

1.a. Start slave server with
--slave-parallel-type=LOGICAL_CLOCK --slave-parallel-workers=N

```
+---------------------+----------+
| Variable_name       | Value    |
+---------------------+----------+
| slave_parallel_type | DATABASE |
+---------------------+----------+
```

※Default 値はDATABASE

又は

1.b Start the slave server normally.
Change the MTS options dynamically using the following
mysql: STOP SLAVE: --if the slave is running
mysql: SET GLOBAL SLAVE_PARALLEL_TYPE='LOGICAL_CLOCK';
mysql: SET GLOBAL SLAVE_PARALLEL_WORKER=N;
mysql: START SLAVE:

```
+-----------------------+-------+
| Variable_name         | Value |
+-----------------------+-------+
| slave_parallel_workers | 3     |
+-----------------------+-------+
```

※Default 値は0

スキーマ内マルチスレッドスレーブ – Sample View

```
mysql> set global slave_parallel_workers=5;
Query OK, 0 rows affected (0.00 sec)

mysql> start slave;
Query OK, 0 rows affected, 1 warning (0.23 sec)

mysql> select WORKER_ID, SERVICE_STATE FROM performance_schema.replication_execute_status_by_worker;
+-----------+---------------+
| WORKER_ID | SERVICE_STATE |
+-----------+---------------+
|         0 | ON            |
|         1 | ON            |
|         2 | ON            |
|         3 | ON            |
|         4 | ON            |
+-----------+---------------+
5 rows in set (0.00 sec)

mysql> show processlist;
+----+-------------+-----------------+------+---------+------+----------------------------------------+------------------------------+
| Id | User        | Host            | db   | Command | Time | State                                  | Info                         |
+----+-------------+-----------------+------+---------+------+----------------------------------------+------------------------------+
| 11 | root        | localhost:58522 | NULL | Query   |    0 | init                                   | show processlist             |
| 19 | system user |                 | NULL | Connect |   75 | Queueing master event to the relay log | NULL                         |
| 20 | system user |                 | NULL | Connect |    0 | Reading event from the relay log       | NULL                         |
| 21 | system user |                 |      | Connect |  190 | Executing event                        | insert into test.t5 values(1) |
| 22 | system user |                 |      | Connect |  190 | Executing event                        | insert into test.t2 values(5) |
| 23 | system user |                 |      | Connect |  190 | Executing event                        | insert into test.t3 values(9) |
| 24 | system user |                 |      | Connect |  190 | Executing event                        | insert into test.t1 values(7) |
| 25 | system user |                 |      | Connect |  190 | Executing event                        | insert into test.t4 values(3) |
+----+-------------+-----------------+------+---------+------+----------------------------------------+------------------------------+
8 rows in set (0.00 sec)
```

Created 5 tables in a single test database on master and used 5 clients to do inserts on them, in parallel.

参照 : MySQL 5.7 Enhanced MTS

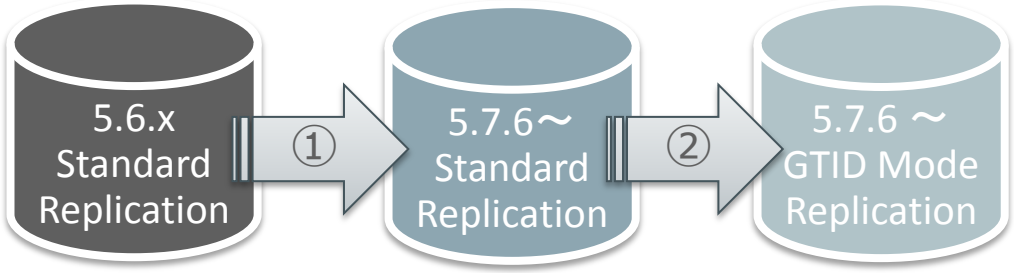
GTIDモード(gtid_mode)レプリケーションへのオンライン移行

MySQL Replicationの設定や運用を簡素化する為に、GTIDモードのレプリケーションを利用する方法がありますが、
既にGTIDモード以前のレプリケーションで運用されている場合は、稼働中の全てのサーバーを停止してからGTIDモードのレプリケーションへ
移行する必要がありました。しかし、稼働中のサービスを停止して切り替える事は難しいのが現実です。
MySQL 5.7.6 以降のMySQLにアップグレードする事で、オンラインのままMySQLのレプリケーションを切り替える事が出来るようになりました。

Offline procedure

It is still possible to use the old, offline procedure.
The procedure is as follows:

1. Disable all write operations.
2. Wait for all transactions to propagate
   from the master(s) to all slaves.
3. Stop all servers.
4. On each server, set gtid-mode=ON in the configuration file.
5. Start all servers.
6. Enable write operations.



Online procedure

各サーバーで以下のコマンドを実行

詳細: 17.1.5.2 Enabling GTID Transactions Online
17.1.5.4 Verifying Replication of Anonymous Transactions

1	SET @@GLOBAL.ENFORCE_GTID_CONSISTENCY = WARN;
2	SET @@GLOBAL.ENFORCE_GTID_CONSISTENCY = ON;
3	SET @@GLOBAL.GTID_MODE = OFF_PERMISSIVE;
4	SET @@GLOBAL.GTID_MODE = ON_PERMISSIVE;
5	SHOW STATUS LIKE 'ONGOING_ANONYMOUS_TRANSACTION_COUNT';
6	SET @@GLOBAL.GTID_MODE = ON;
7	On each server, add gtid-mode=ON to my.cnf.
8	STOP SLAVE ; CHANGE MASTER TO MASTER_AUTO_POSITION = 1; START SLAVE;

参照 : **Enabling Global Transaction Identifiers Without Downtime in MySQL 5.7.6**

MySQL Slave Replication Filters Dynamic

MySQL-5.7.3以前は、ユーザーはOptionファイル(my.cnf)か、コマンドラインパラメータを使用してフィルタリングルールを設定することが出来ますが、いずれの場合もMySQLサーバはフィルタリングルールの変更を反映する為に再起動する必要があります。

MySQL-5.7.3の新しく導入された ***"CHANGE REPLICATION FILTER"*** コマンドを利用する事で、
ユーザーは*slave* 側のレプリケーションフィルターを動的に、サーバーの再起動せずに適用する事が可能です。

The following slave replication filters can be changed dynamically using this command.

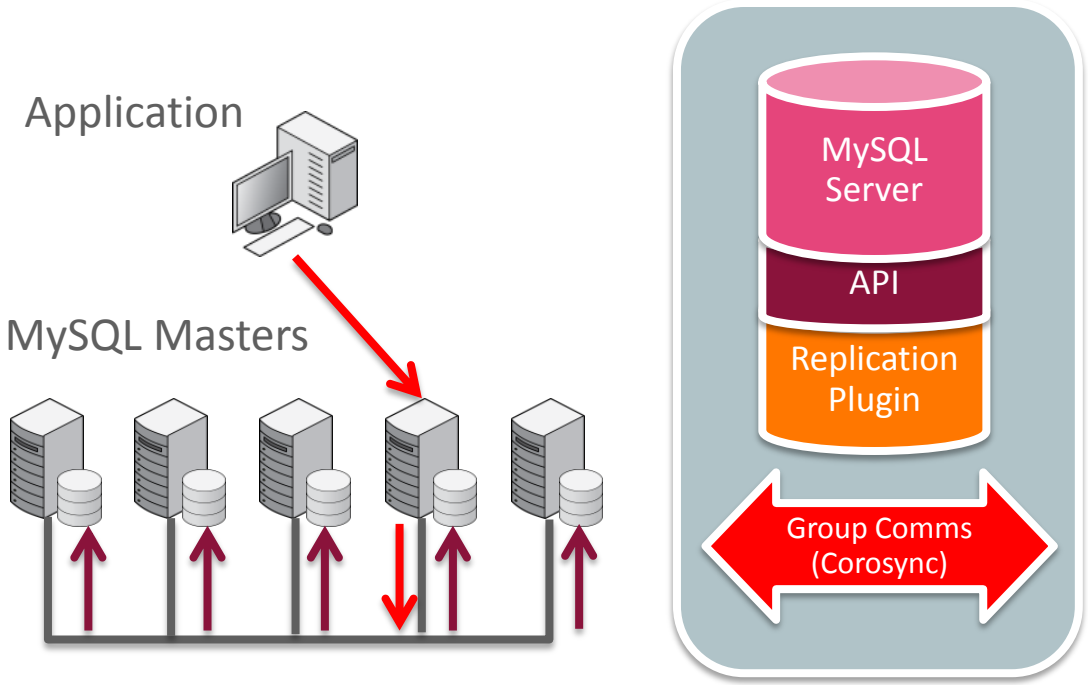
- REPLICATE_DO_DB
- REPLICATE_IGNORE_DB
- REPLICATE_DO_TABLE
- REPLICATE_IGNORE_TABLE
- REPLICATE_WILD_DO_TABLE
- REPLICATE_WILD_IGNORE_TABLE
- REPLICATE_REWRITE_DB

例) フィルター設定手順

```
mysql> STOP SLAVE SQL_THREAD;
Query OK, 0 rows affected (0.05 sec)
mysql> CHANGE REPLICATION FILTER
REPLICATE_DO_DB=(db1);
Query OK, 0 rows affected (0.00 sec)
```

参照: MySQL-5.7.3- Making MySQL Slave Replication Filters Dynamic

MySQL 5.7: グループレプリケーション



- シェアード・ナッシング型”疑似”同期レプリケーション
- 更新はマルチ・マスタ型でどこでも可能
  - 矛盾の検知と解決(トランザクションのロールバック
  - “Optimistic State Machine” レプリケーション
- グループメンバーの管理と障害検知を自動化
  - サーバのフェールオーバー不要
  - 構成の拡張/縮小の柔軟性
  - 単一障害点無し
  - 自動再構成
- 既存構成との統合
  - InnoDB
  - GTIDベースのレプリケーション
  - PERFORMANCE_SCHEMA

MySQL HA & Scaling Solutions

	MySQL Replication	MySQL Fabric	Oracle VM Template	Oracle Clusterware	Solaris Cluster	Windows Cluster	DRBD	MySQL Cluster
App Auto-Failover	✖	✔	✔	✔	✔	✔	✔	✔
Data Layer Auto-Failover	✖	✔	✔	✔	✔	✔	✔	✔
Zero Data Loss	MySQL 5.7	MySQL 5.7	✔	✔	✔	✔	✔	✔
Platform Support	All	All	Linux	Linux	Solaris	Windows	Linux	All
Clustering Mode	Master + Slaves	Master + Slaves	Active/Passive	Active/Passive	Active/Passive	Active/Passive	Active/Passive	Multi-Master
Failover Time	N/A	Secs	Secs +	Secs +	Secs +	Secs +	Secs +	< 1 Sec
Scale-out	Reads	✔	✖	✖	✖	✖	✖	✔
Cross-shard operations	N/A	✖	N/A	N/A	N/A	N/A	N/A	✔
Transparent routing	✖	For HA	✔	✔	✔	✔	✔	✔
Shared Nothing	✔	✔	✖	✖	✖	✖	✔	✔
Storage Engine	InnoDB+	InnoDB+	InnoDB+	InnoDB+	InnoDB+	InnoDB+	InnoDB+	NDB
Single Vendor Support	✔	✔	✔	✔	✔	✖	✔	✔

まとめ

#	Category	Summary
1	**パフォーマンス**	MySQL5.7から同一スキーマ内でも複数スレッドが 処理可能になり、レプリケーションの処理が大幅に向上。 ※ 5.6ではインスタンスでは複数だが、スキーマ毎には一つのスレッドのみでした。
2	**管理性**	Multi-Sourceレプリケーションにより、複数インスタンスのデータを一つのサーバーに集約する事が可能なので、レポート集計や作成などの対応が、容易に出来るようになりました。また、バイナリーログポジションベースのレプリケーションから、GTIDベースのレプリケーションへオンラインでの移行が可能になりました。レプリケーション対象オブジェクトをオンラインでフィルターする事も可能になりました。
3	**可用性**	Loss-Lessレプリケーションにより、マスターとスレーブ間でのデータの完全性を確実にする事が出来、マスター障害発生時もマスターとスレーブ間でのデータ差異が発生しないようになりました。また、オンラインでレプリケーション設定変更可能。レプリケーションの設定変更もオンラインで対応可能な範囲が拡張され、メンテナンスによるシステムダウンタイムを減らす事が可能になりました。

Program Agenda

1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. MySQL5.7におけるレプリケーション拡張
5. 参考情報

	MySQL Editions		
	Standard Edition	Enterprise Edition	Cluster CGE
機能概要			
MySQL Database	✔	✔	✔
MySQL Connectors	✔	✔	✔
MySQL Replication	✔	✔	✔
MySQL Fablic		✔	✔
MySQL Partitioning		✔	✔
MySQL Utilities		✔	✔
Storage Engine: MyISAM, InnoDB	✔	✔	✔
Storage Engine: NDB (ndbcluster)			✔
MySQL Workbench SE/EE*	✔	✔	✔
MySQL Enterprise Monitor*		✔	✔
MySQL Enterprise Backup*		✔	✔
MySQL Enterprise Authentication (外部認証サポート) *		✔	✔
MySQL Enterprise Audit (ポリシーベース監査機能) *		✔	✔
MySQL Enterprise Encryption（非対称暗号化）*		✔	✔
MySQL Enterprise Firewall（SQLインジェクション対策）*			
MySQL Enterprise Scalability (スレッドプール) *		✔	✔
MySQL Enterprise High Availability (HAサポート) *		✔	✔
Oracle Enterprise Manager for MySQL*		✔	✔
MySQL Cluster Manager (MySQL Cluster管理) *			✔
MySQL Cluster Geo-Replication			✔

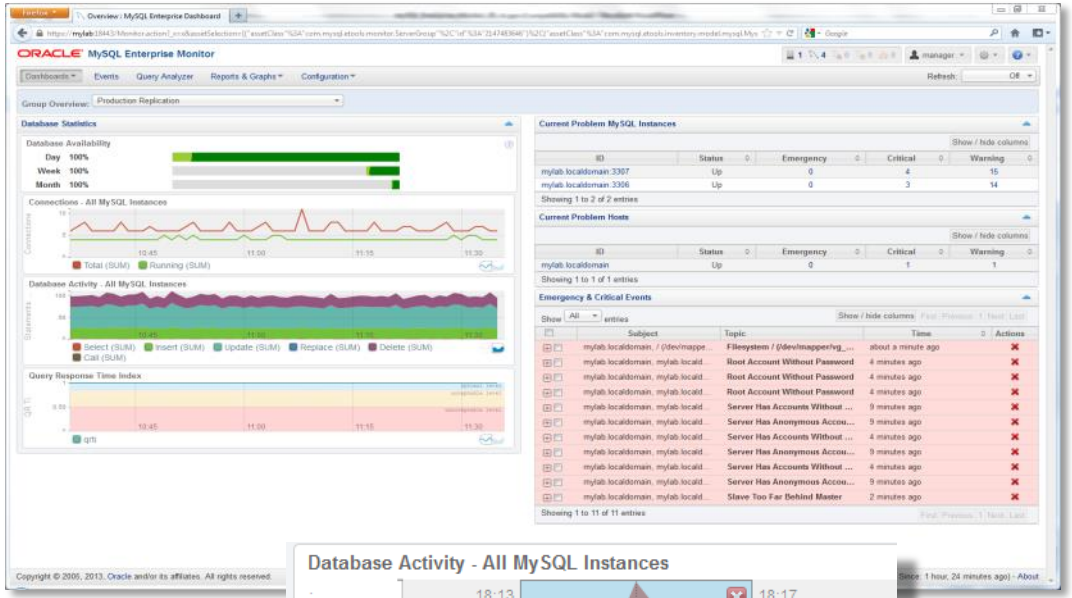
*商用版のみで利用可能な追加機能

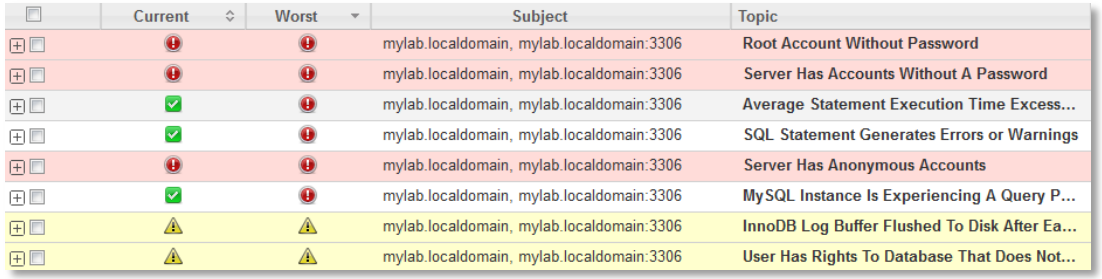
	MySQL Editions		
	Standard SE	Enterprise EE	Cluster CGE
Oracle Premium Support			
24時間365日サポート	✔	✔	✔
インシデント数無制限	✔	✔	✔
ナレッジベース	✔	✔	✔
バグ修正&パッチ提供	✔	✔	✔
コンサルティングサポート	✔	✔	✔
オラクル製品との動作保証			
Oracle Linux	✔	✔	✔
Oracle VM	✔	✔	✔
Oracle Solaris	✔	✔	✔
Oracle Enterprise Manager		✔	✔
Oracle GoldenGate		✔	✔
Oracle Data Integrator		✔	✔
Oracle Fusion Middleware		✔	✔
Oracle Secure Backup		✔	✔
Oracle Audit Vault and Database Firewall		✔	✔

※最新の対比表は、MySQL Editionsのサイトを参照下さい。

MySQL Enterprise Monitor

- パフォーマンスと可用性の監視
- 問題のあるSQL文の検知
- ディスク監視と容量プランニング
- クラウド対応アーキテクチャ
  - ポリシーベースの設定
  - エージェント導入不要
- MySQL監視を10分以内で開始可能





参照: [MySQL Enterprise Monitor]

Enterprise Replication Monitor

- レプリケーショントポロジーの自動検知
- マスター/スレーブのパフォーマンス監視
- レプリケーションアドバイザーによるサポート
- レプリケーションのベストプラクティスを提示

Replicationの遅延を検知して通知

Replication Monitoring

Servers	Type	Threads		Time Behind	Binary Logs				Log Space	
		IO	SQL		Current File	Position		Position	Binary Logs	Relay Logs
Replication 1 (4)	MIXED	✓	✓							
mylab.localdomain:3306	master/slave	✓	✓	00:00:00	mylab-bin.000001	791	...bin.000001	791	791 B	1.1 KB
mylab.localdomain:3307	master/slave	✓	✓	00:00:00	mylab-bin.000001	791	...ylab-bin.000001	791	791 B	1.1 KB
mylab.localdomain:3308	master/slave	✓	✓	00:00:00	mylab-bin.000001	986	mylab-bin.000001	791	0.96 KB	1.1 KB
MLORD-PC:3306	slave	✓	✓	00:00:00			mylab-bin.000001	986		1.29 KB

MySQL Enterprise Support

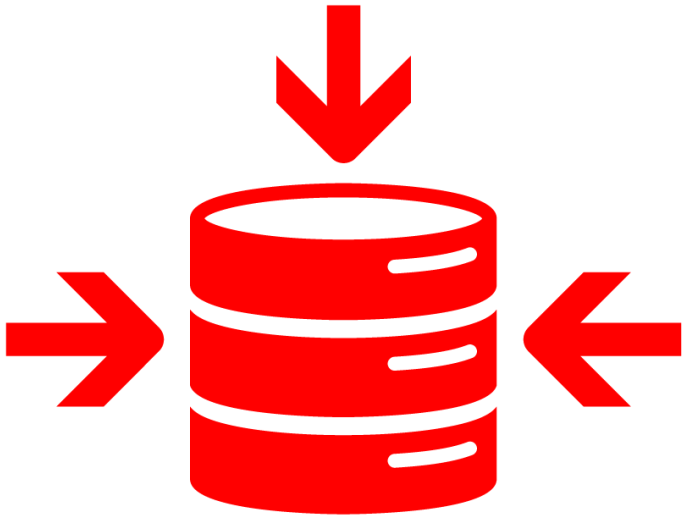
- 最大のMySQLのエンジニアリングおよびサポート組織
- MySQL開発チームによるサポート
- 29言語で世界クラスのサポートを提供
- メンテナンス・リリース、バグ修正、パッチ、アップデートの提供
- 24時間x365日サポート
- 無制限サポート・インシデント
- MySQL コンサルティング・サポート

Get immediate help for any MySQL issue, plus expert advice

MySQL Consultative Support

Make the Most of your Deployments

- リモート・トラブル・シューティング
- レプリケーション・レビュー
- パーティショニング・レビュー
- スキーマ・レビュー
- クエリー・レビュー
- パフォーマンス・チューニング
- ...and more



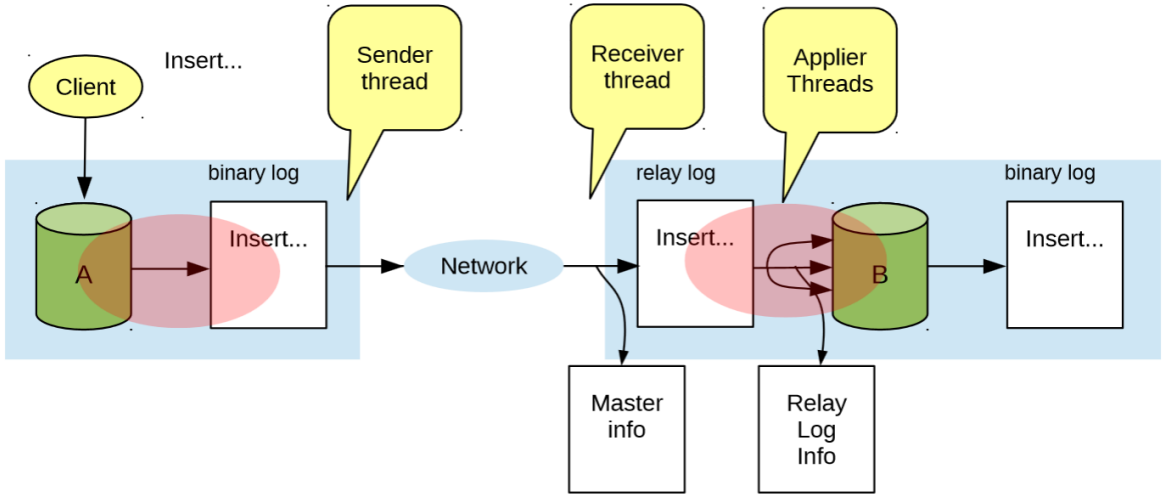
参照：MySQL コンサルティング・サポート

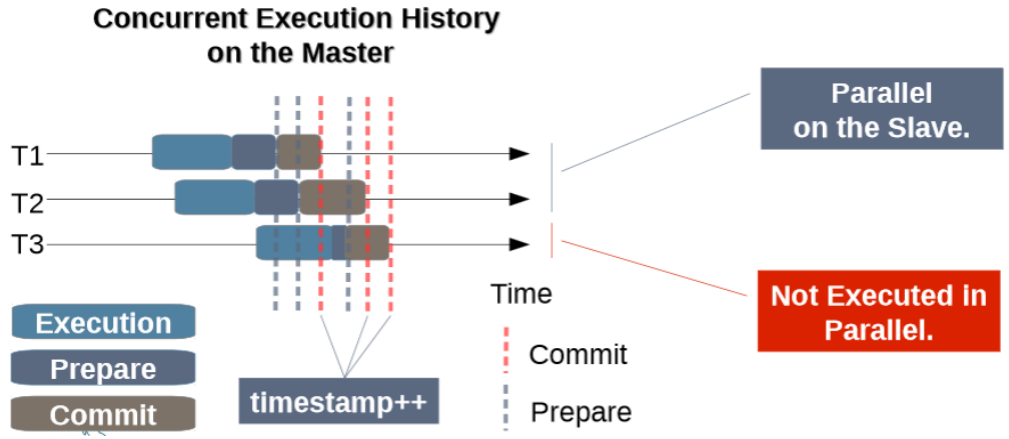
有難うございました

Q&A

Q:準同期レプリケーションを東京 – 大阪などの遠隔地間で同期をした場合、全体的に処理が遅くなるのではないでしょうか？
A: はい。準同期の場合、以下のコマンドのどちらを選択しても、アプリケーションにACKを返すのはスレーブログにも書き終わった後になるので、レスポンス重視の場合は遠隔地へのレプリケーションにはA-SYNCで同期させるのが宜しいかと思います。
SET rpl_semi_sync_master_wait_point= [AFTER_SYNC|AFTER_COMMIT]

Q: マルチスレッドスレーブの場合にどのスレッドが並行して実行されるのでしょうか？
SET slave_parallel_type= [**logical_clock**|database]
slave-parallel-type=LOGICAL_CLOCK --slave-parallel-workers=N
A: 以下の図のように、Applierスレッドが複数 立ち上がり並行して実行されます。





- 同じタイムスタンプのトランザクションは並行して実行されます。
- 並行して実行されるトランザクションはそれぞれ独立してコミットされる為、待機イベントは発生しません。