
ORACLE®

MySQLレプリケーション入門

Yoshiaki Yamasaki / 山崎 由章
MySQL Senior Sales Consultant, Asia Pacific and Japan

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

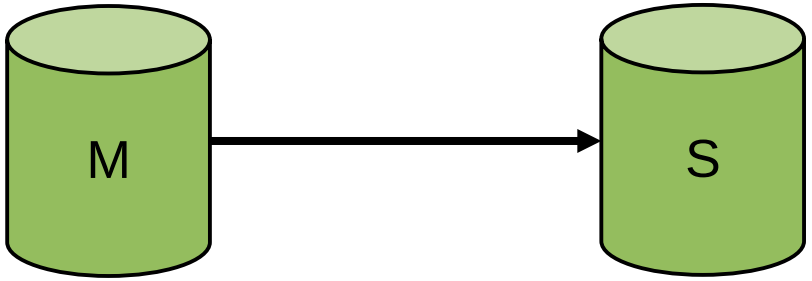
1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. レプリケーションの設定方法
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報

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. レプリケーションの設定方法
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報

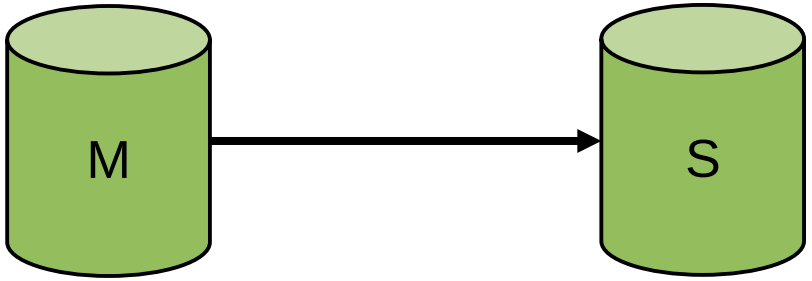
レプリケーションとは？

- 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

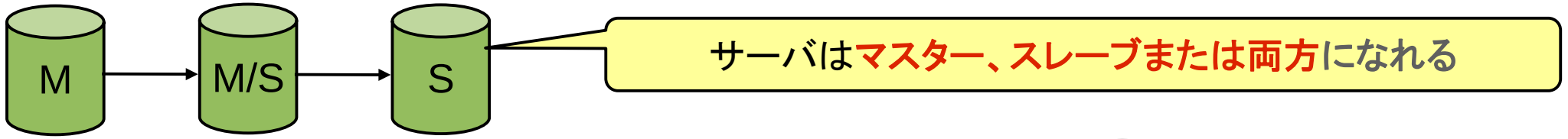


レプリケーションとは？

- 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

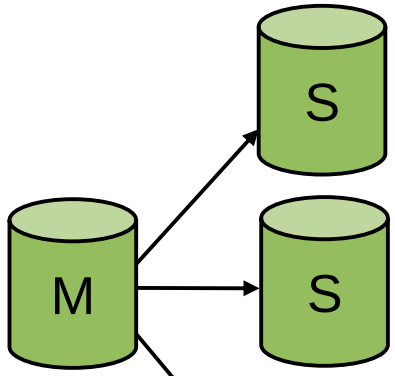


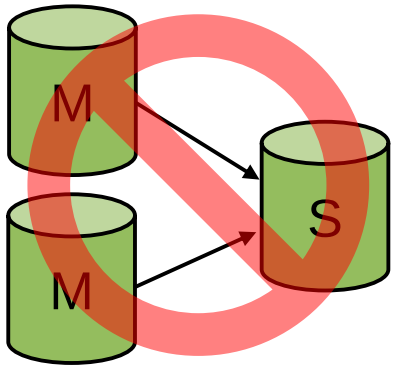
補足



サーバは**マスター、スレーブまたは両方**になれる

マスターは**複数のスレーブ**を持てる

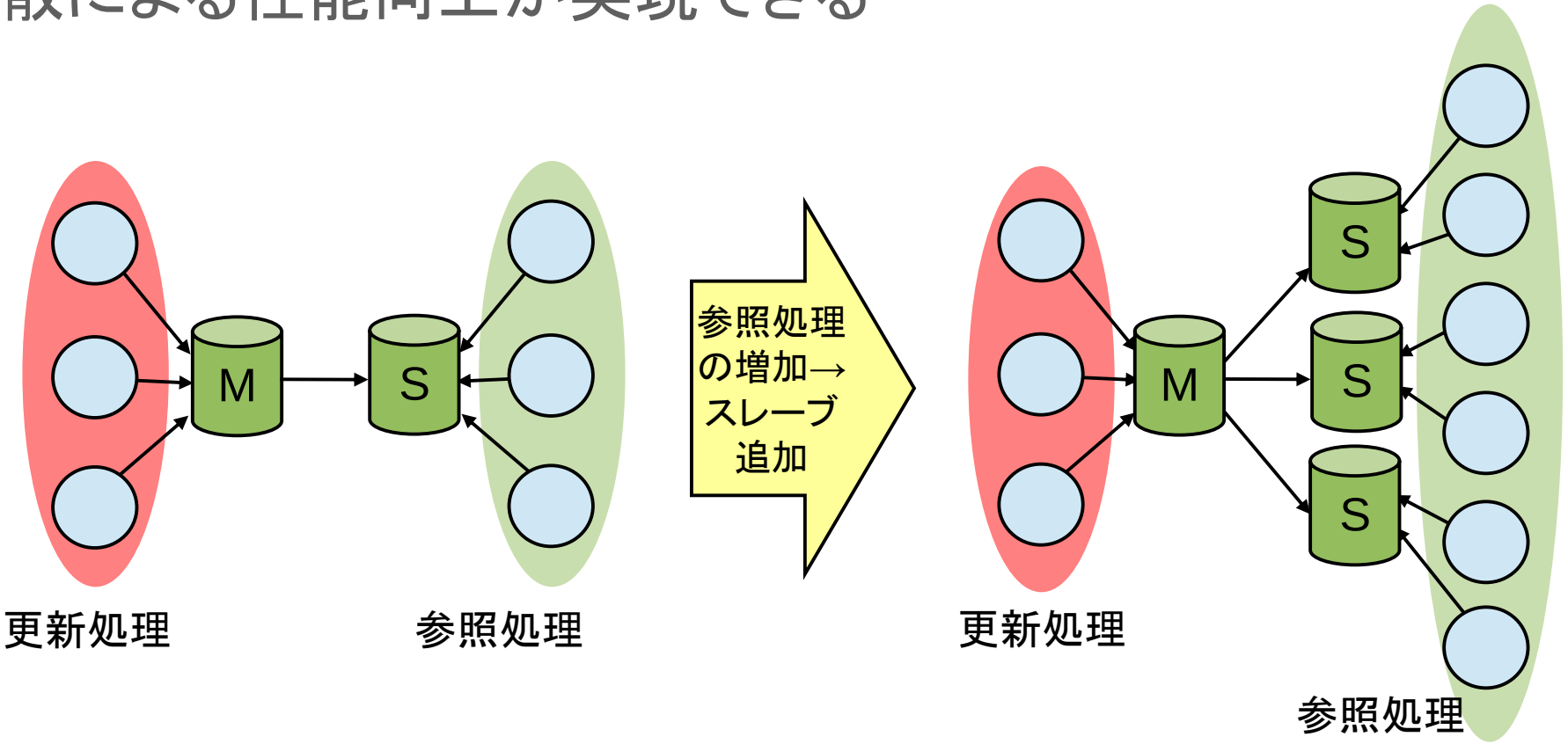




スレーブは**1つのマスターのみ**を持てる

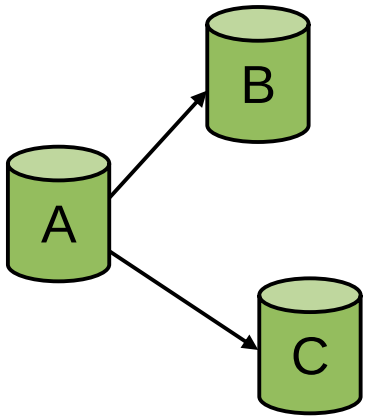
レプリケーションの利点：参照性能の向上

- 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



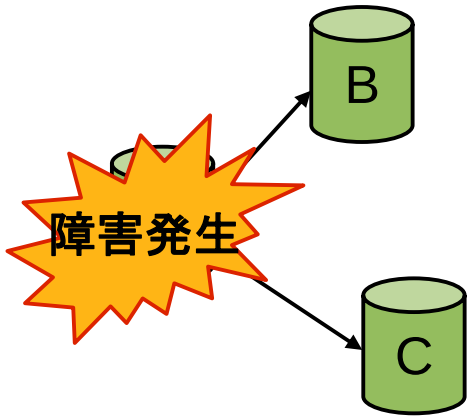
レプリケーションの利点：高可用性構成の実現

- 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



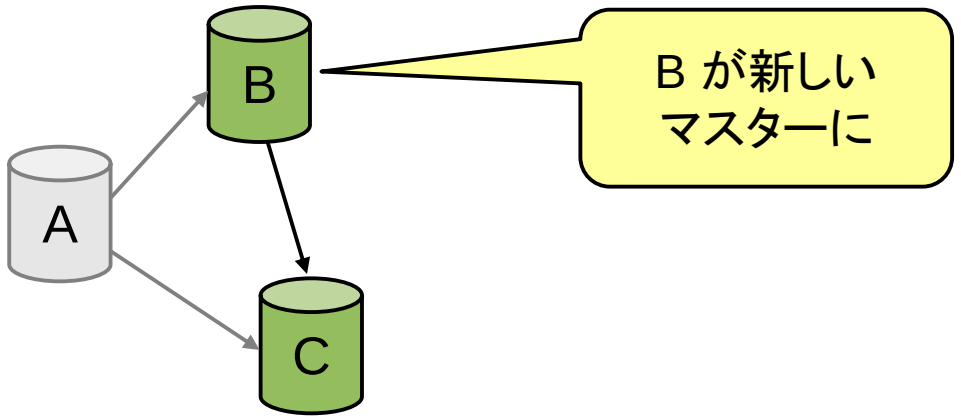
レプリケーションの利点：高可用性構成の実現

- 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

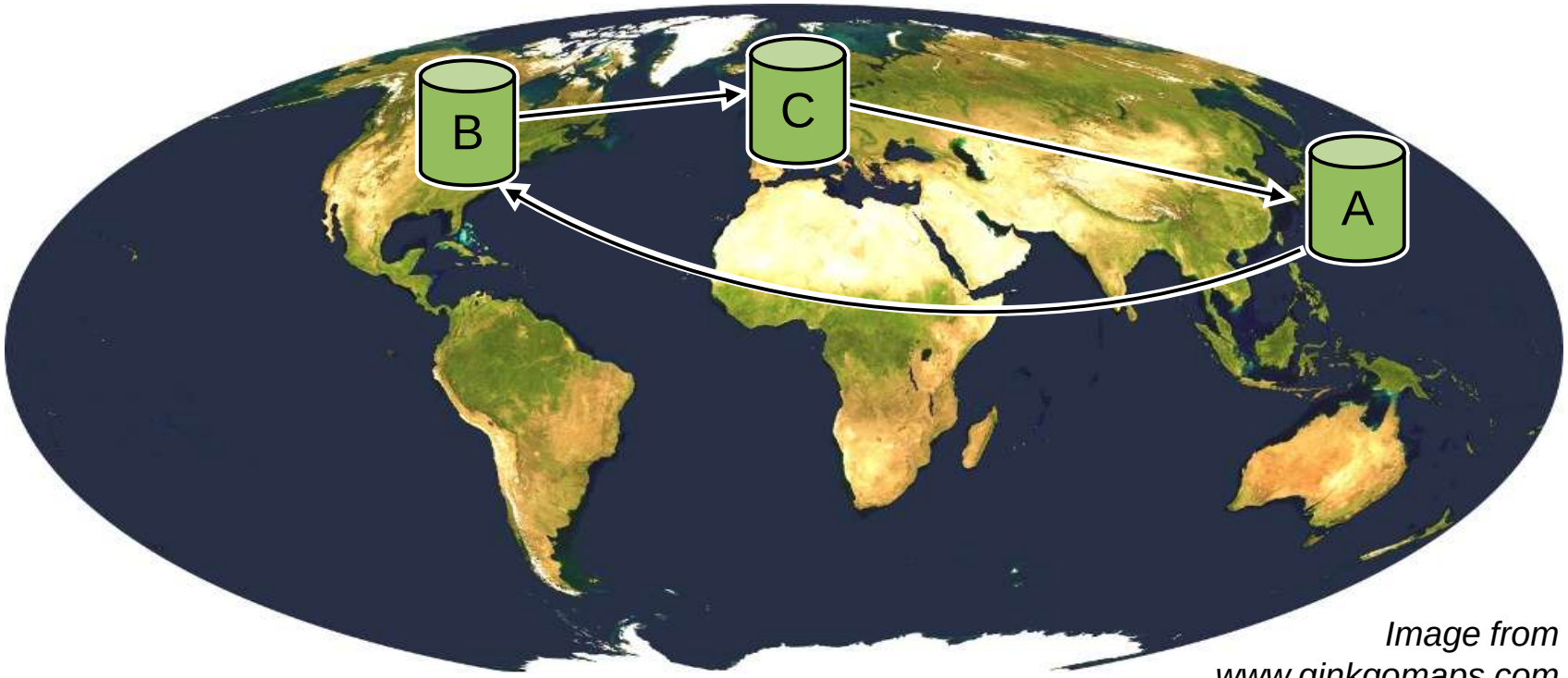
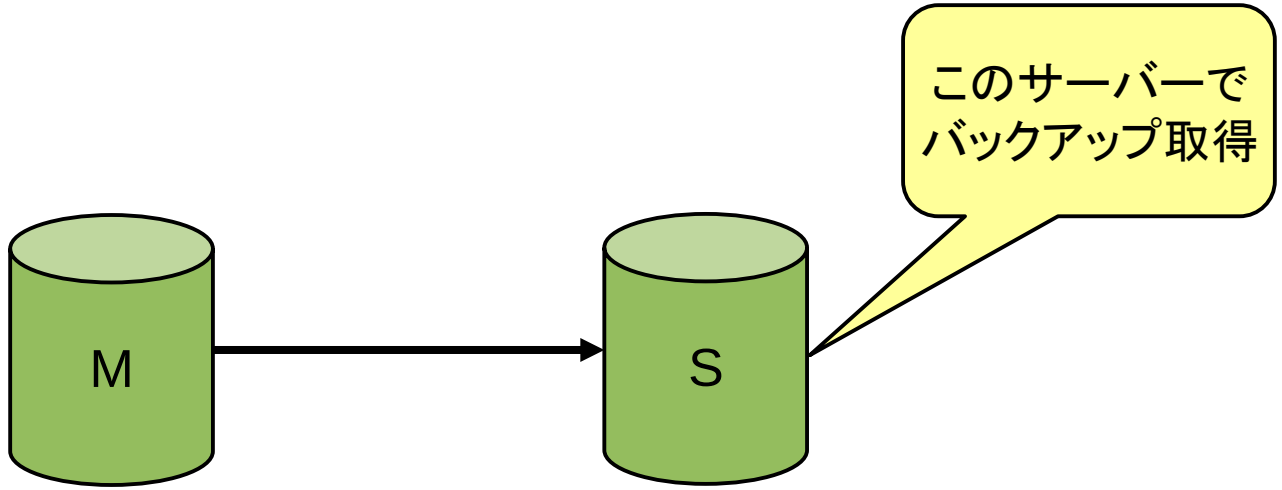


Image from www.ginkgomaps.com

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

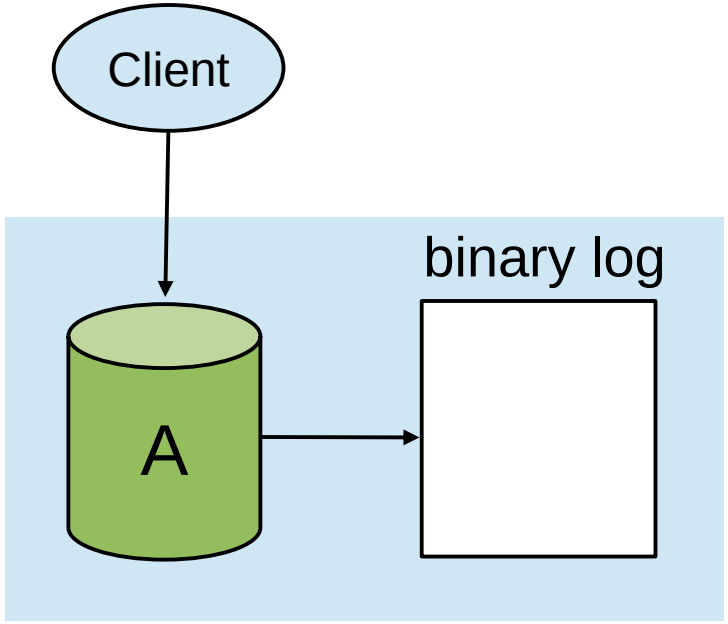


Program Agenda

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. レプリケーションの設定方法
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報

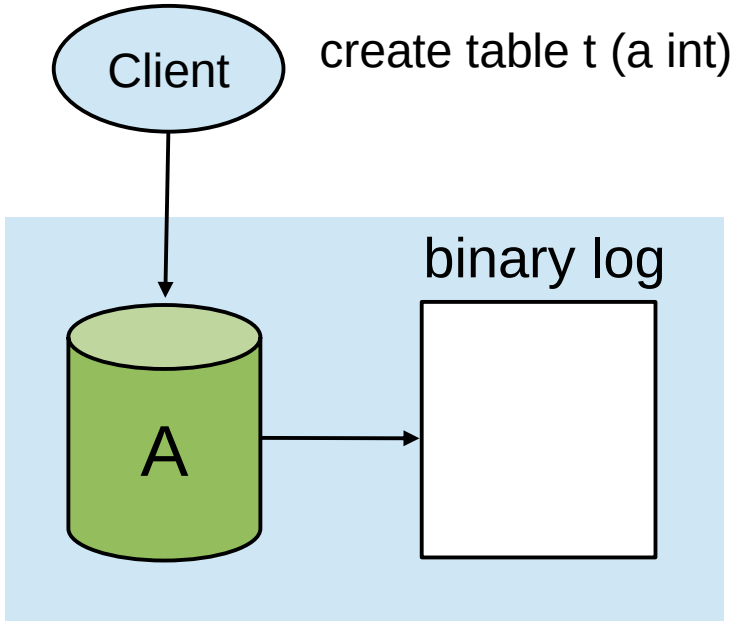
レプリケーションの仕組み

- 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



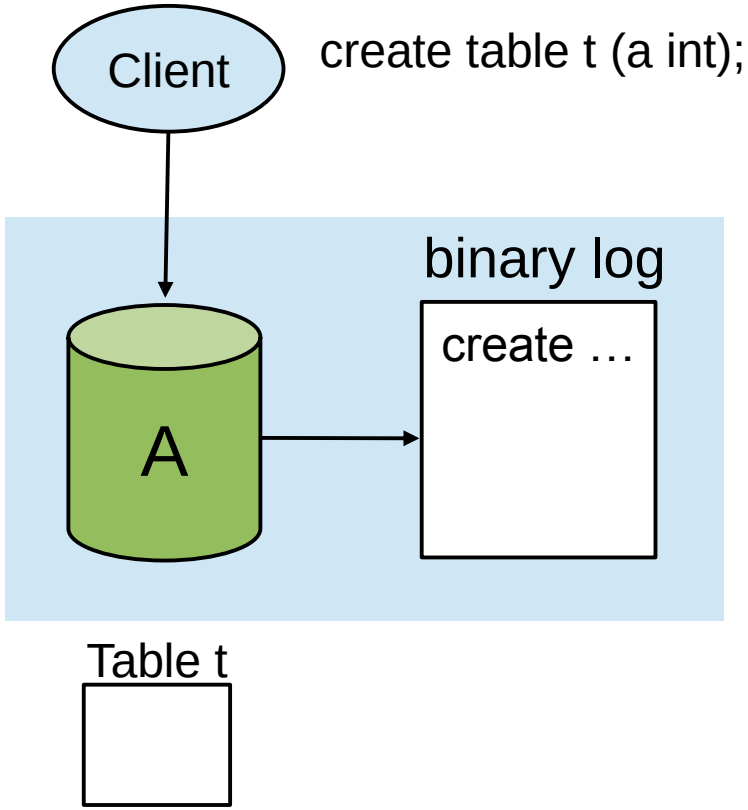
レプリケーションの仕組み

- 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



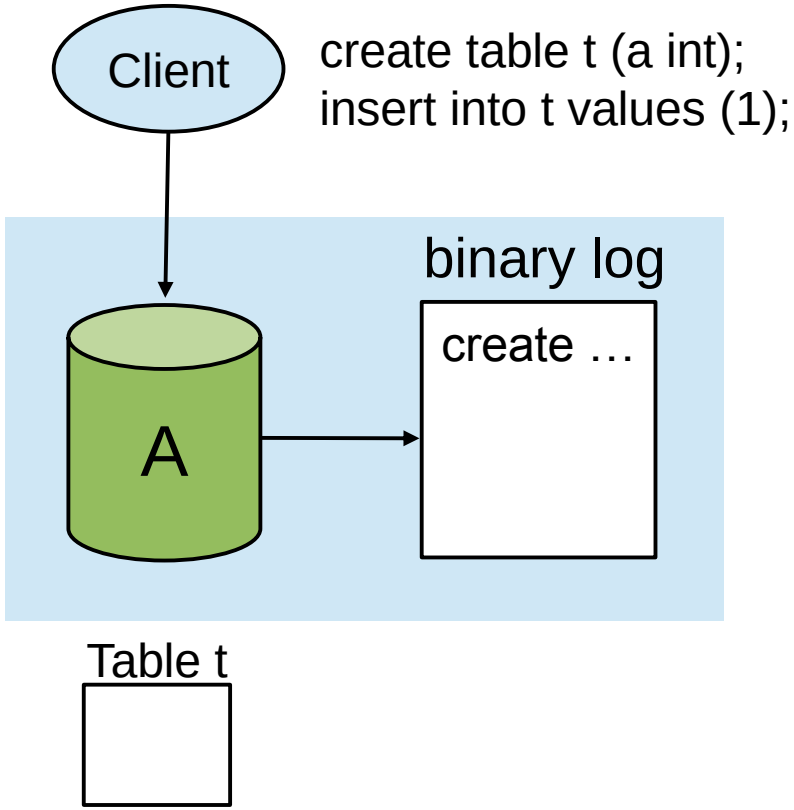
レプリケーションの仕組み

- 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



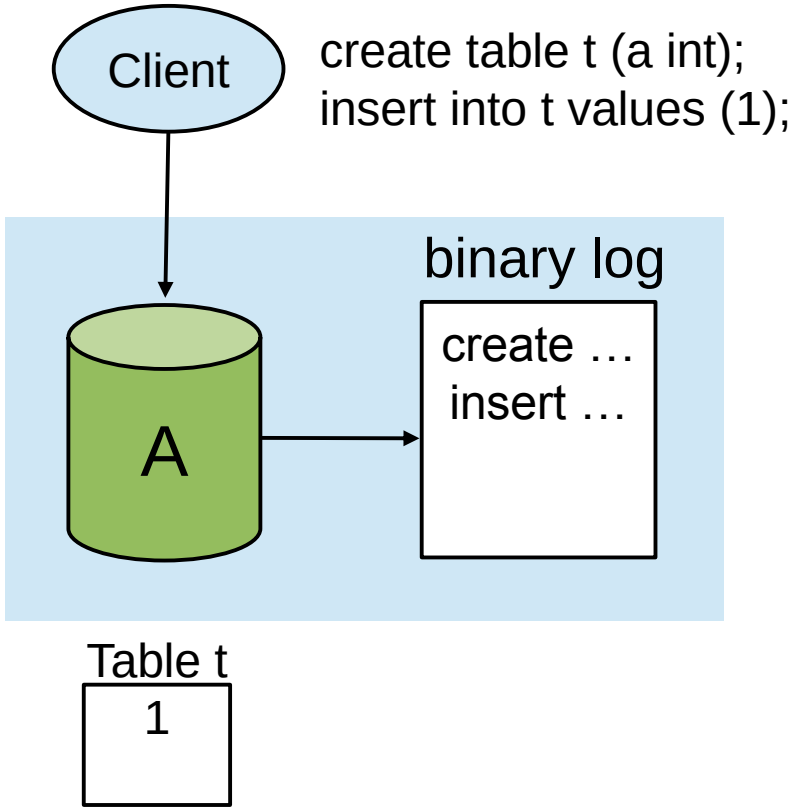
レプリケーションの仕組み

- 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



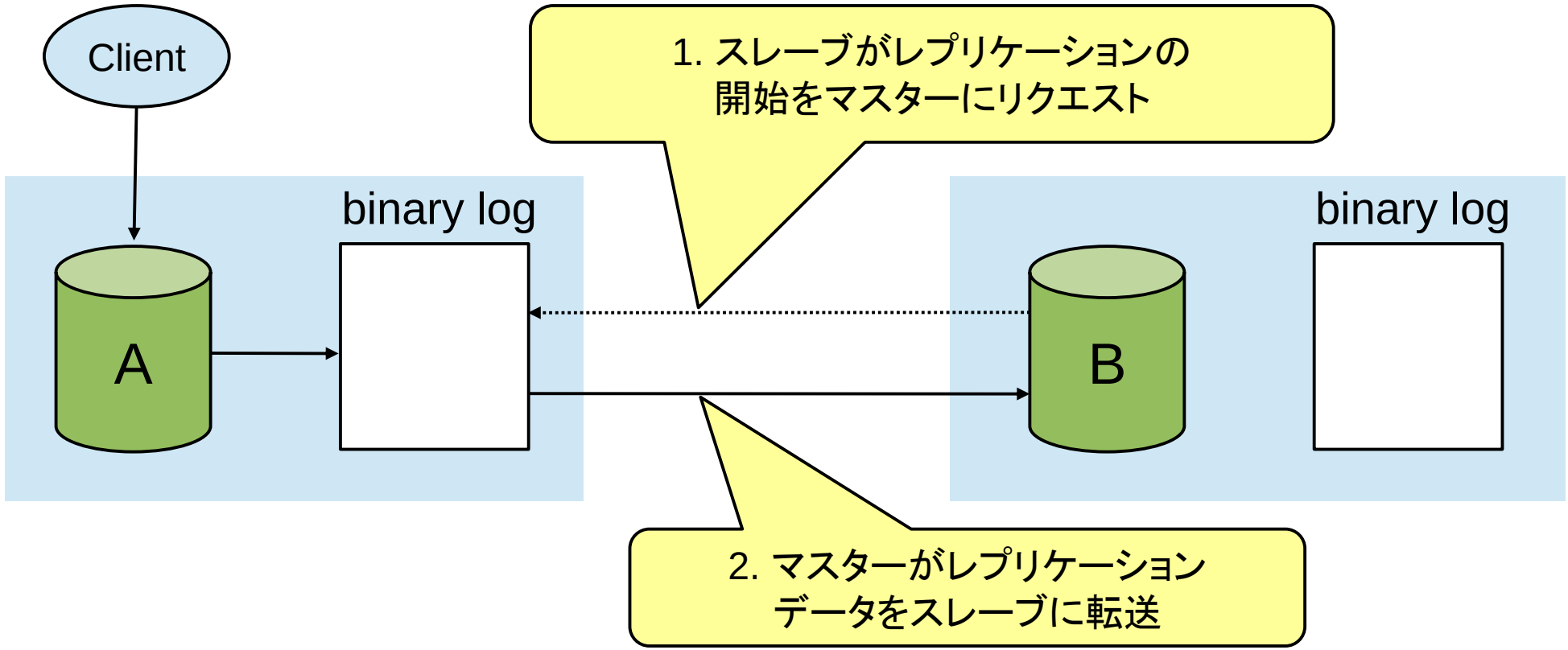
レプリケーションの仕組み

- 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



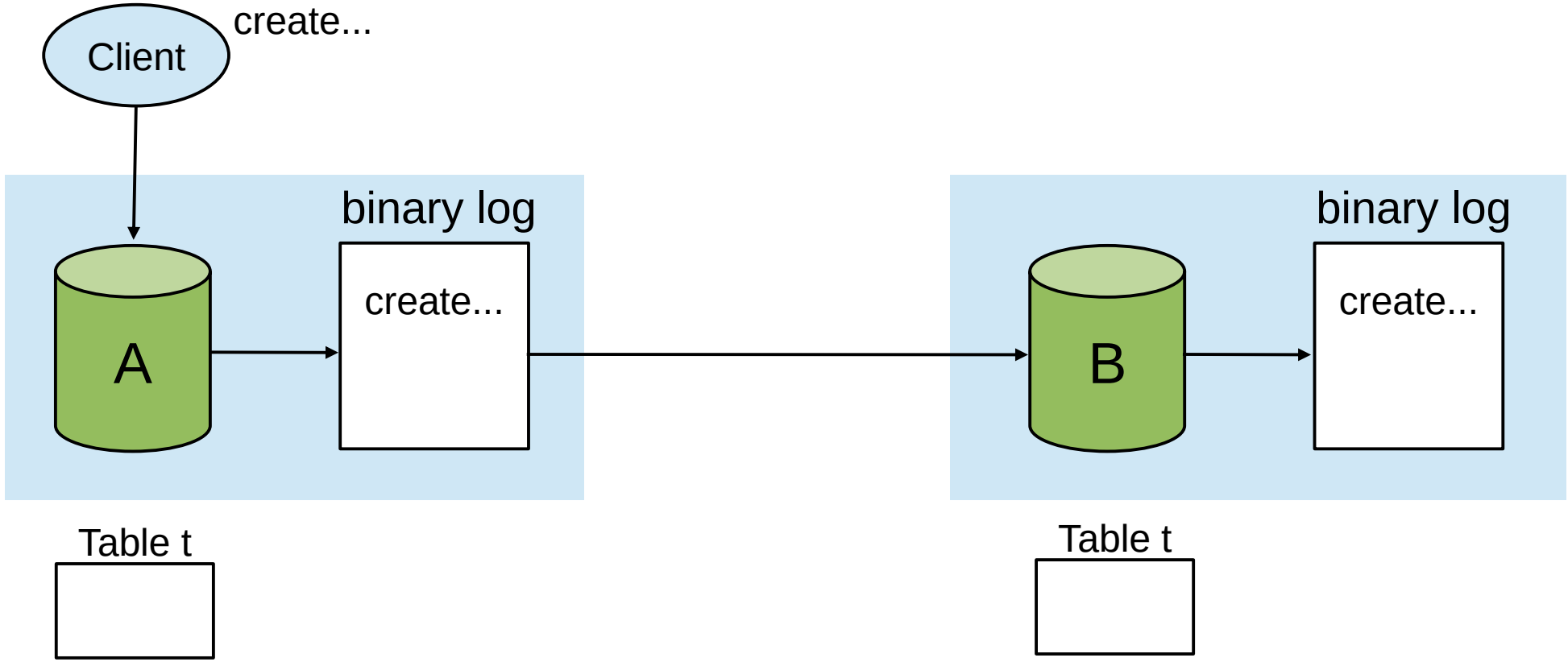
レプリケーションの仕組み

- スレーブからレプリケーション開始



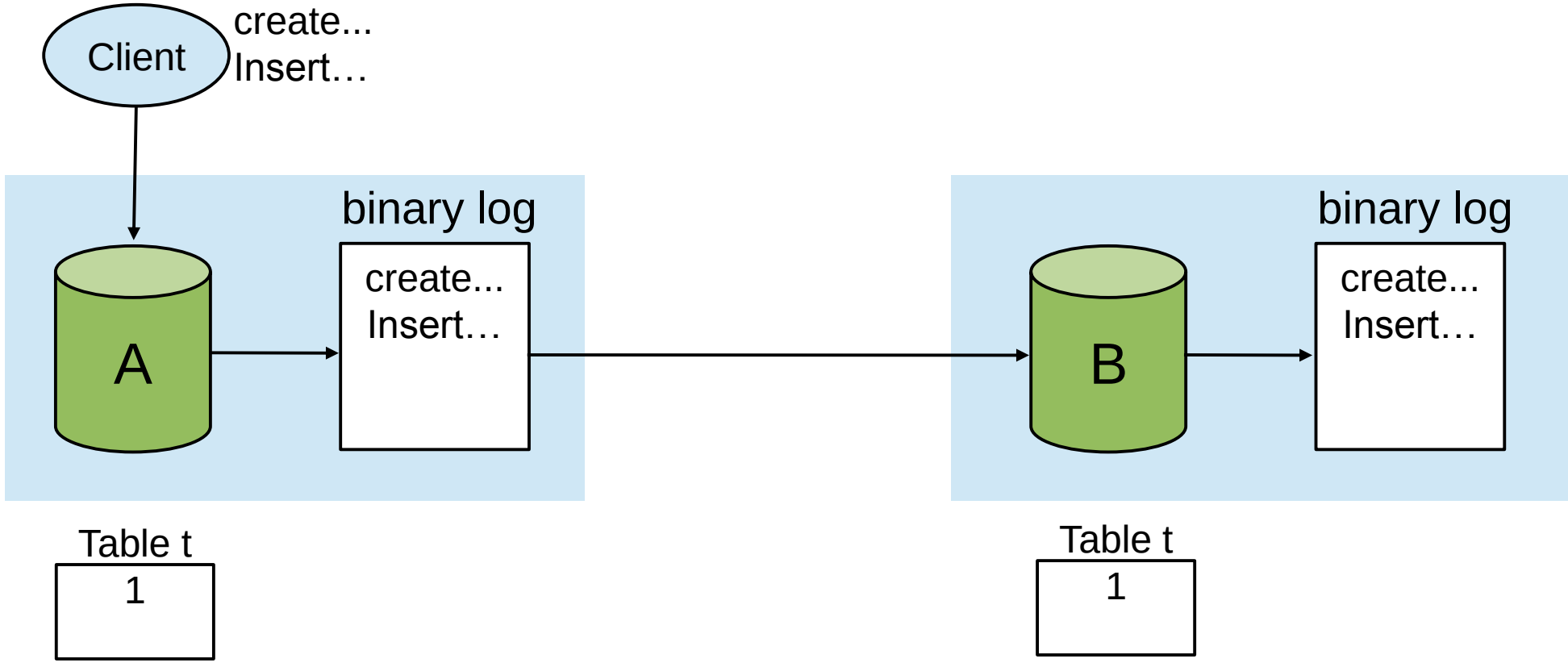
レプリケーションの仕組み

- 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
- 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. レプリケーションの設定方法
5. バイナリログの管理方法
6. その他の考慮事項
7. 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

フォーマット	説明	バイナリログのサイズ	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

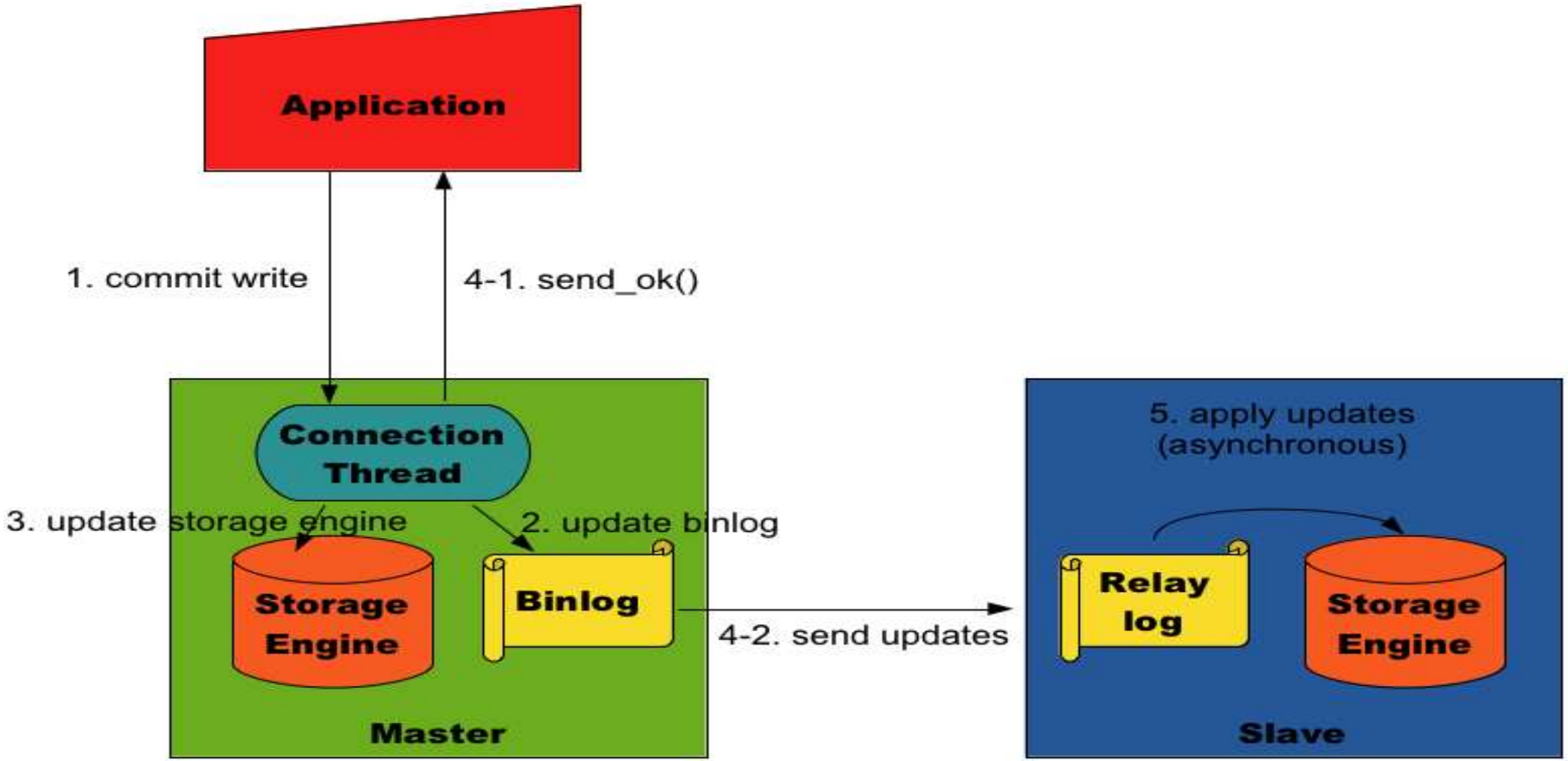
Non-deterministicとは？

- 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

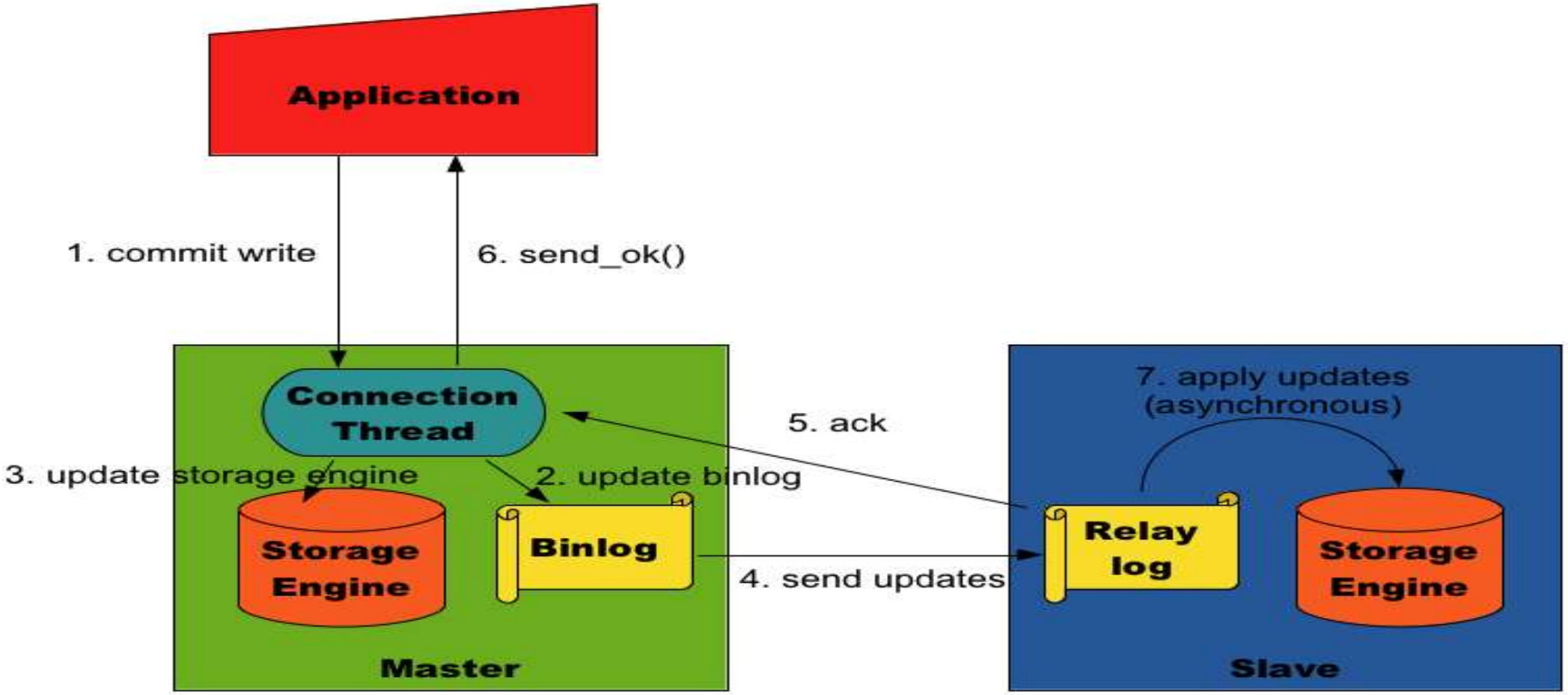
同期方式による違い

- 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

非同期レプリケーション

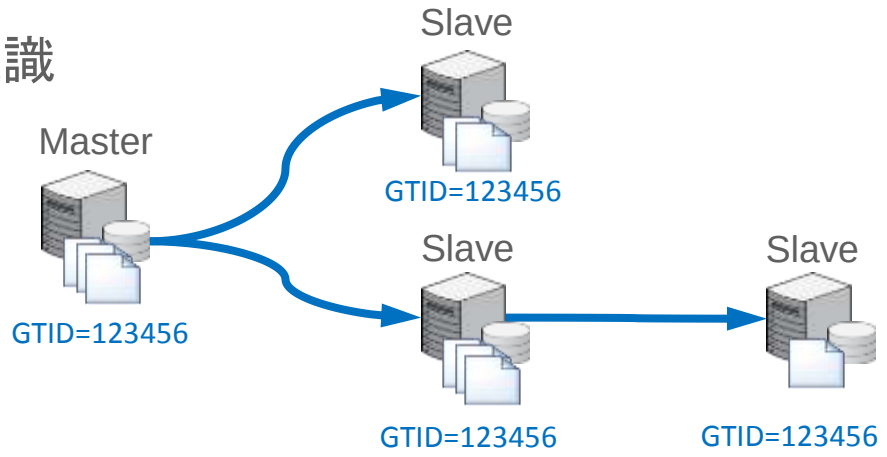


準同期レプリケーション



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



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. **レプリケーションの設定方法**
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報

レプリケーションの設定方法(GTIDを使わない場合)

1. レプリケーション用のパラメータを設定
2. マスターサーバーのバックアップを取得して、
   スレーブサーバーにリストア
   - バックアップ取得時のバイナリログファイルのファイル名とポジションを記録しておく
3. マスターサーバーにレプリケーション用ユーザーを作成
4. スレーブサーバーで CHANGE MASTER TO コマンドを実行
5. スレーブサーバーで START SLAVE コマンドを実行

1. レプリケーション用のパラメータ設定

- マスター : 下記オプションを設定して起動
  - server-id
  - log-bin
  - datadir *
- スレーブ : 下記オプションを設定して起動
  - server-id
  - datadir *
  - port *
  - socket *　(Lunix系OSの場合)
  - read_only　(必須ではないが、設定を推奨)

* は、テスト目的で1台のサーバー内でマスター、スレーブを作成する場合に必要な設定

2. バックアップを取得してスレーブサーバーへリストア

- コールドバックアップを取得してリストアする
- mysqldumpでバックアップを取得してリストアする
  - **バックアップ取得例**

```
$ mysqldump --user=root --password=root --master-data=2 ¥
    --socket=/usr/local/mysql/data/mysql.sock ¥
    --hex-blob --default-character-set=utf8 --all-databases ¥
    --single-transaction > mysql_bkup_dump.sql
```

※バックアップ取得時のバイナリログファイルのファイル名とポジションを記録しておく

補足:mysqldumpのオプション

- **--master-data=2**
  - バックアップ取得のバイナリファイル名とバイナリファイル内の位置(**Position**)を
    コメントとしてバックアップファイルに記録
- **--hex-blog**
  - バイナリ型(**BINARY**、**VARBINARY**、**BLOG**) と**BIT**型のデータを16進数表記で出力
- **--default-character-set**
  - **mysqldump**がデフォルトで利用するキャラクタセットを指定。
    通常はMySQLサーバのシステム変数**default-character-set**と同じものを指定すれば良い
- **--all-databases**
  - 全てのデータベースをバックアップ
- **--lock-all-tables**
  - 全てのテーブルをロックしてバックアップを取得する
- **--single-transaction**
  - InnoDBがサポートしているトランザクションの仕組みを利用して、
    InnoDBテーブルに限り一貫性のとれたバックアップを取得する

注意事項：mysqldumpによるバックアップ

- データの整合性を保つために、バックアップ取得中は、テーブルに関するDDL文（※）を実行しないこと

  ※ALTER TABLE, CREATE TABLE, DROP TABLE,RENAME TABLE, TRUNCATE TABLE

- マニュアルの“--single-transaction”オプションの説明部分より引用
  - 「While a --single-transaction dump is in process, to ensure a valid dump file (correct table contents and binary log coordinates), no other connection should use the following statements: ALTER TABLE, CREATE TABLE, DROP TABLE, RENAME TABLE, TRUNCATE TABLE.」

3. マスターサーバーにレプリケーション用ユーザーを作成

- "REPLICATION SLAVE"権限を付与してユーザーを作成
  - 例

```
CREATE USER ‘repl'@'localhost' IDENTIFIED BY ‘repl';
GRANT REPLICATION SLAVE ON *.* TO ‘repl'@'localhost';
```

4. スレーブサーバーで CHANGE MASTER TO コマンドを実行
5. スレーブサーバーで START SLAVE コマンドを実行

- CHANGE MASTER TO コマンドを実行
- START SLAVE コマンドを実行
  - 例

```
CHANGE MASTER TO MASTER_HOST='localhost',
 -> MASTER_USER='repl',
 -> MASTER_PASSWORD='repl',
 -> MASTER_LOG_FILE='bin.000001',
 -> MASTER_LOG_POS=1790;
START SLAVE;
```

※青字部分は、バックアップ取得時に記録したファイル名とポジションを指定

※MySQL 5.6の場合、セキュリティ向上のためにCHANGE MASTER TO時にMASTER_USER、MASTER_PASSWORDを指定せずに、START SLAVE時に指定することも可能。(master.info内にユーザ名/パスワードが保存されることを防ぐ)

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. レプリケーションの設定方法
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報

バイナリログの管理

- **SHOW MASTER STATUS** コマンドで現在使用中のバイナリログファイル名とポジションを確認
- **SHOW MASTER LOGS** コマンドで全てのバイナリログファイル名を列挙
- **FLUSH [BINARY] LOGS** コマンドまたはMySQLサーバの再起動でログファイルのローテーション
- **PURGE MASTER** コマンドで特定の時点までのバイナリログを削除
- **RESET MASTER** コマンドで全てのバイナリログを削除

※バイナリログは溜まり続けるファイルであるため、運用の中で定期的に削除が必要

バイナリログの管理

```
mysql> SHOW MASTER STATUS;
+--------------+----------+--------------+------------------+
| File         | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------+----------+--------------+------------------+
| MySQL.000007 |      107 |              |                  |
+--------------+----------+--------------+------------------+
1 row in set (0.00 sec)

mysql>
mysql> SHOW MASTER LOGS;
+--------------+-----------+
| Log_name     | File_size |
+--------------+-----------+
| MySQL.000001 |      1110 |
| MySQL.000002 |      2797 |
...
| MySQL.000007 |       107 |
+--------------+-----------+
7 rows in set (0.00 sec)
```

バイナリログの管理

```
mysql> FLUSH BINARY LOGS;
Query OK, 0 rows affected (0.42 sec)

mysql> SHOW MASTER STATUS;
+---------------+----------+--------------+------------------+
| File          | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+---------------+----------+--------------+------------------+
| MySQL.000008 |      107 |              |                  |
+---------------+----------+--------------+------------------+
1 row in set (0.00 sec)

mysql> SHOW MASTER LOGS;
+---------------+-----------+
| Log_name      | File_size |
+---------------+-----------+
| MySQL.000001 |      1110 |
...
| MySQL.000007 |       146 |
| MySQL.000008 |       107 |
+---------------+-----------+
8 rows in set (0.00 sec)
```

バイナリログの管理

```
mysql> PURGE MASTER LOGS TO 'MySQL.000003';
Query OK, 0 rows affected (0.06 sec)

mysql> SHOW MASTER LOGS;
+---------------+-----------+
| Log_name      | File_size |
+---------------+-----------+
| MySQL.000003 |      2315 |
| MySQL.000004 |       628 |
| MySQL.000005 |      1090 |
| MySQL.000006 |       126 |
| MySQL.000007 |       146 |
| MySQL.000008 |       107 |
+---------------+-----------+
6 rows in set (0.00 sec)
```

バイナリログの管理

```
mysql> RESET MASTER;
Query OK, 0 rows affected (0.06 sec)

mysql> SHOW MASTER LOGS;
+---------------+-----------+
| Log_name      | File_size |
+---------------+-----------+
| MySQL.000001 |       107 |
+---------------+-----------+
1 row in set (0.00 sec)
```

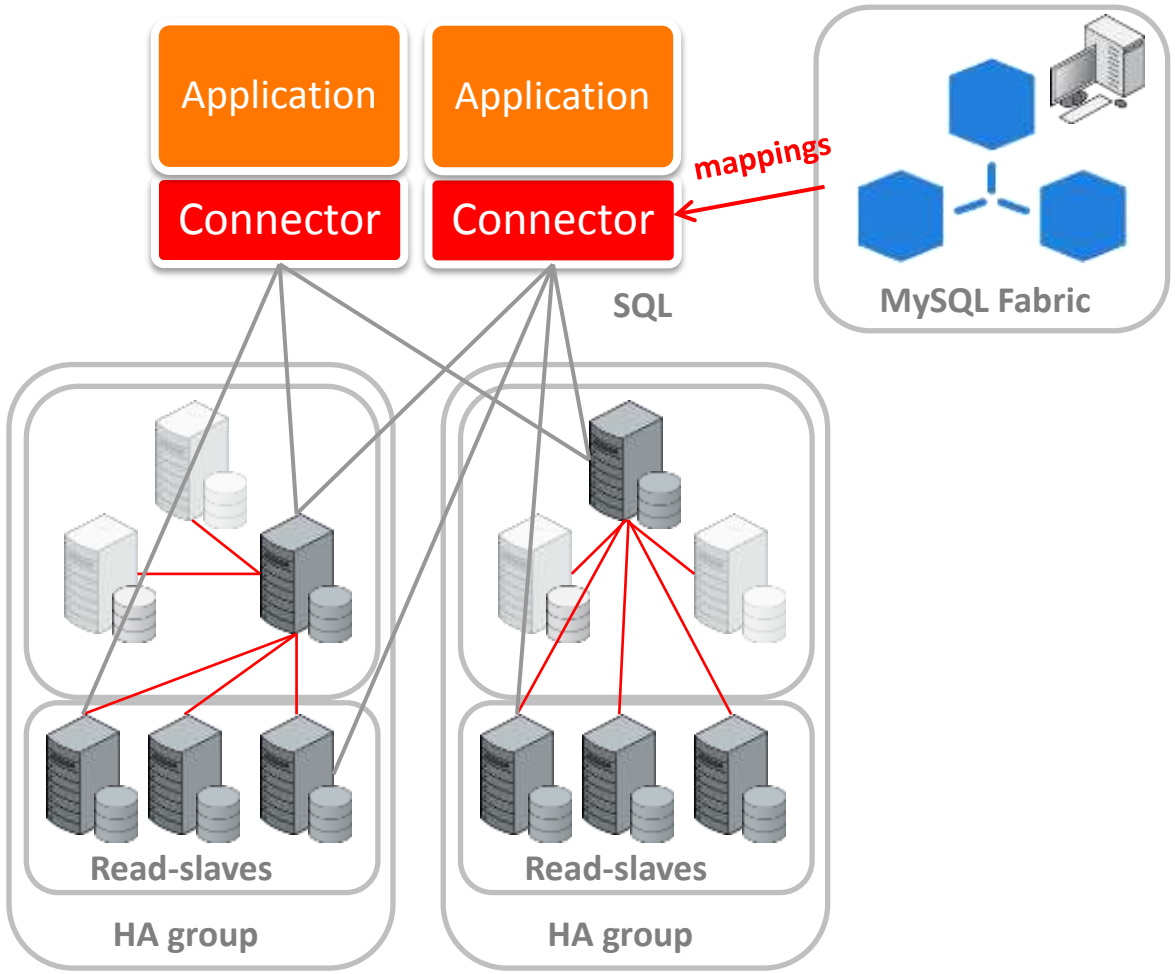
Program Agenda

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. レプリケーションの設定方法
5. バイナリログの管理方法
6. **その他の考慮事項**
7. 参考情報

その他の考慮事項

- MySQLレプリケーション単独では用意されていない機能
  - 高可用性構成としての利用時にフェールオーバーさせる仕組み
    =>MySQL 5.6にて、自動フェールオーバーできるスクリプトを提供(MySQL Utilities)
  - 更新と参照の処理を振り分ける仕組み
    =>MySQL Fabricで制御可能
  - スレーブ間でのロードバランスの仕組み
    =>MySQL Fabricで制御可能

MySQL Fabric 1.5: 高可用性 & シャーディング



- OpenStack との統合
- 高可用性
  - サーバの監視; スレーブの自動昇格と透過的なレプリケーション切り替え
- シャーディングによる拡張性
  - アプリケーションがシャードのキーを提供
    - 整数型、日付型、文字列型
  - レンジまたはハッシュ
  - シャード再構成可能
- Fabric対応コネクタ利用: Python, Java, PHP, .NET, C (labs)
  - プロキシを使わないので低レイテンシ、ボトルネック無し

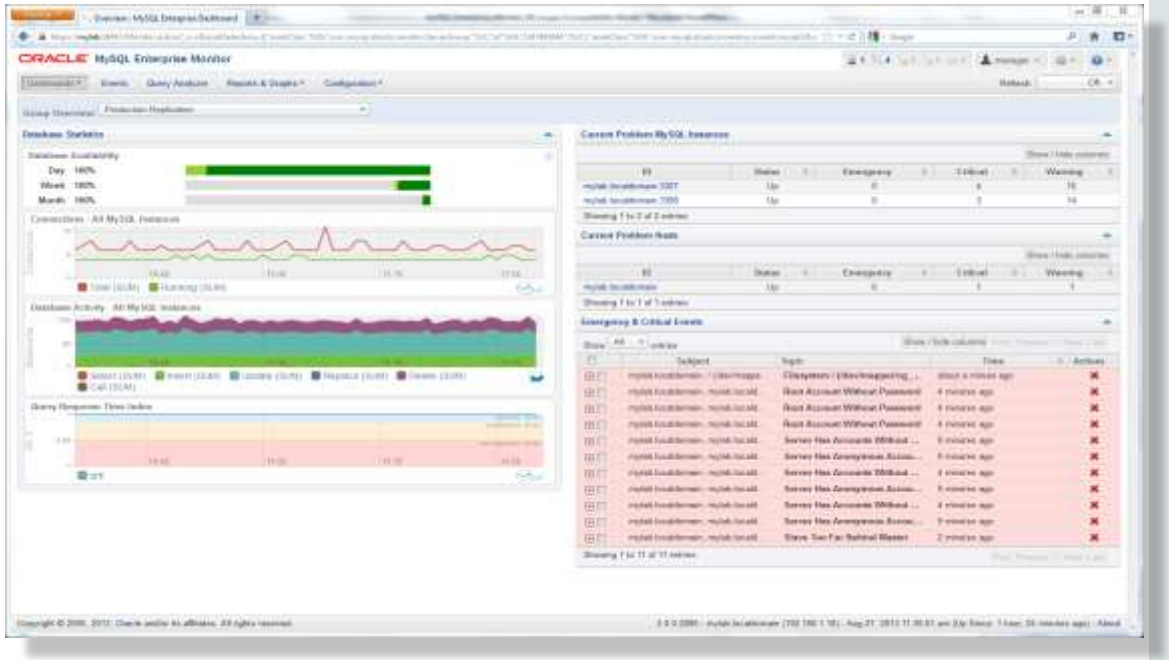
その他の考慮事項

- 一度に大量の更新処理を実行しない
  - スレーブの遅延を防ぐための工夫
- レプリケーションが正しく運用できているか監視する
  ⇒ MySQL Enterprise Monitorで監視可能

MySQL Enterprise Monitor

- 複数のMySQLサーバを一括監視可能なダッシュボード
- システム中のMySQLサーバやレプリケーション構成を自動的に検出し監視対象に追加
- ルールに基づく監視と警告
- **問題が発生する前に通知**
- 問題のあるSQL文の検出、統計情報の分析が可能なQuery Analyzer

参照： MySQL Enterprise Monitor



“バーチャルなMySQL DBA”アシスタント

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

"The MySQL Enterprise Monitor is an absolute must for any DBA who takes his work seriously."

- Adrian Baumann, System Specialist
Federal Office of Information Technology & Telecommunications

Replication Monitoring

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

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. レプリケーションの設定方法
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報

ホワイトペーパー

- MySQLレプリケーション - MySQL 5.5によるスケーラビリティと可用性の強化
  http://www.mysql.com/why-mysql/white-papers/wp-mysql-5-5-replication-ja/

- MySQL 5.6 レプリケーション : 概要
  http://www.mysql.com/why-mysql/white-papers/mysql-replication-ja/

- MySQL 5.6 レプリケーション : チュートリアル - スケーラビリティと可用性の強化
  http://www.mysql.com/why-mysql/white-papers/mysql-replication-tutorial-ja/

MySQL5.6～ 主なレプリケーションパラメータ オプション

レプリケーション関連設定	概要
binlog_row_image=minimal	行イメージを全て保持するのではなく、最低限のカラムの情報だけ保持するように、バイナリログのフォーマットを変更可能です。full, minimal, noblob
slave_parallel_workers=n	スレーブ側での処理をマルチスレッド化できるため、スレーブの遅延を改善出来る可能性があります。（スキーマ単位）
relay_log_info_repository=TABLE	ポジションの情報をInnoDB上のテーブルに記録する為、クラッシュセーフになりました。(“relay_log_recovery=ON”も合わせて設定)
relay_log_recovery=ON	リレーログに問題があった時に、マスターから自動的に読み直します。
binlog_checksum=NONE	バイナリーログチェックサム CRC32, NONE (デフォルト値：CRC32)
binlog_rows_query_log_events=ON	SQL文に関する情報をバイナリログに追加できます。レプリケーションの追跡や問題発生時のデバッグに役に立ちます。(mysqlbinlog –vv ）
MASTER_DELAY=N	CHANGE MASTER実行時に指定。遅延させたい時間を秒単位で指定
MASTER_BIND	CHANGE MASTER実行時に指定。スレーブサーバーが複数のNICを持っている場合、マスターとの接続に使用するNICを明示的に指定できるようになりました。

Hardware and Software
Engineered to Work Together


ORACLE®