MySQL最新情報
～MySQL Cloud Service、MySQL 8.0、MySQL InnoDB Clusterなどのご紹介～

2016/11/12

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

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。

Program Agenda

1. MySQL 8.0 DMR
2. MySQL Cluster 7.5 GA
3. MySQL Group Replication RCと拡張機能
4. Oracle MySQL Cloud Service
5. まとめ

DMR

MySQL 8.0

ORACLE

MySQL 5.7 – 全面的な改良

全部で200以上の改善！！

- Replication
- InnoDB
- Optimizer
- Security
- Performance Schema
- GIS

- Triggers
- Partitioning
- New! SYS Schema
- New! JSON
- Performance

The MySQL 5.7 Story *(cont.)*

Mobile

Schemaless Database

Improved MySQL

MySQL 8.0 DMR（開発途上版）

- 2016年9月12日リリース！
- フィードバック募集中
- バグ報告や機能追加要望はこちらから
  - MySQL Bugs
    https://bugs.mysql.com/

UTF-8

MySQL 8.0

- New! 最新のUnicode 9.0をサポート
- 国ごとの照合順序に対応中
  - アクセント、濁音、破裂音の区別
  - 大文字小文字の区別
- UCA DUCETをベース
  （デフォルトUnicode照合基本テーブル）👍
- **デフォルトキャラクターセットとしてのUTF8MB4**
  - MySQL 5.7の時からプロジェクト開始
  - 性能影響を減らすための数々の改善

JSON対応の拡張

MySQL 8.0

- MySQL Document Store
- MySQL Shellを使ってMySQLを管理
  - ワンストップDevOpsツール
  - 好みの言語を選択可能：SQL、JavaScript、Python、・・・

New! 不可視索引（Invisible Indexes）

- オプティマイザーから見えない索引
  - 索引の無効化とは異なる
  - データ更新時に不可視索引は更新される
- 2つのユースケース:
  - 仮削除（ゴミ箱）
  - 索引採用のテスト

ユースケース1:索引の仮削除(ゴミ箱)

使用例

- 索引の仮削除
```
ALTER TABLE Country ALTER INDEX c INVISIBLE;
```
- 索引の復旧
```
ALTER TABLE Country ALTER INDEX c VISIBLE;
```
- 索引の削除
```
ALTER TABLE Country DROP INDEX c;
```

ユースケース2：索引採用のテスト

- 新しい索引作成は、既存の実行計画を変化させる可能性があるため リスクを伴う
- 不可視索引は索引採用のテスト容易にする
  - 準備段階で不可視索引を作成し、必要に応じて一時的に有効化しながら 索引の有効性を確認する
  - 上手く動くことが確認出来てから索引を永続的に有効化する
  - 現時点では、クエリー単位で不可視索引を有効化する機能は無いが、 ユーザーからの要望が多く集まれば機能追加を検討している（※次ページ参照）

```
ALTER TABLE Country ADD INDEX c (Continent) INVISIBLE;
# after some time
ALTER TABLE Country ALTER INDEX c VISIBLE;
```

クエリー単位で不可視索引を有効化する機能について

- 機能追加要望をBug#83066で登録済みです
- Bug #83066
  - Feature Request : Use Invisible Indexes Specific Query
  - http://bugs.mysql.com/bug.php?id=83066
- 機能追加を希望する方は、“Affects me”をクリックして下さい！

不可視索引の確認方法

```
SELECT * FROM information_schema.statistics WHERE is_visible='NO';
*************************** 1. row ***************************
TABLE_CATALOG: def
 TABLE_SCHEMA: world
   TABLE_NAME: Country
   NON_UNIQUE: 1
 INDEX_SCHEMA: world
   INDEX_NAME: c
 SEQ_IN_INDEX: 1
  COLUMN_NAME: Continent
    COLLATION: A
  CARDINALITY: 7
     SUB_PART: NULL
       PACKED: NULL
     NULLABLE:
   INDEX_TYPE: BTREE
      COMMENT: disabled
INDEX_COMMENT:
   IS_VISIBLE: NO
```

仮削除の候補選定に役立つsys.schema_unused_indexes

- sys.schema_unused_indexesから、使用していないインデックスを確認可能
  - DB起動後にアクセスがあったテーブルに付けられているインデックスが対象
  - 再起動後の稼働期間が十分でない場合は、この情報を確認後、不可視索引を使って安全にインデックスを削除可能

仮削除の候補選定に役立つsys.schema_unused_indexes

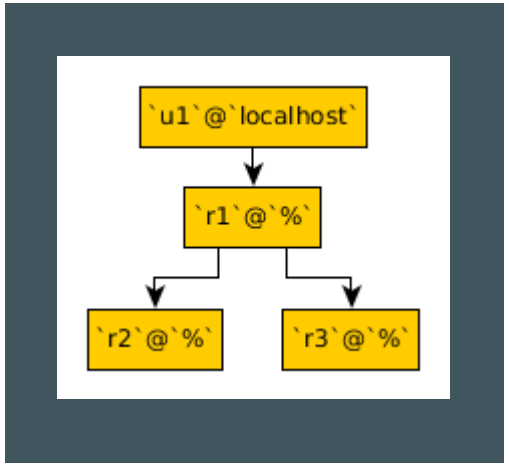
```
mysql> SELECT * FROM sys.schema_unused_indexes;
Empty set (0.00 sec)

mysql> use world;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SELECT * FROM sys.schema_unused_indexes;
+---------------+--------------------+-------------+
| object_schema | object_name        | index_name  |
+---------------+--------------------+-------------+
| world         | City               | CountryCode |
| world         | CountryLanguage    | CountryCode |
+---------------+--------------------+-------------+
2 rows in set (0.00 sec)
```

仮削除の候補選定に役立つsys.schema_unused_indexes

```
mysql> SELECT count(*) FROM world.City WHERE CountryCode='JPN';
+----------+
| count(*) |
+----------+
|      248 |
+----------+
1 row in set (0.00 sec)

mysql> SELECT * FROM sys.schema_unused_indexes;
+---------------+-------------------+--------------+
| object_schema | object_name       | index_name   |
+---------------+-------------------+--------------+
| world         | CountryLanguage | CountryCode |
+---------------+-------------------+--------------+
1 row in set (0.00 sec)
```

New! ロール（権限をまとめて付与/剥奪）

- ロール作成/削除、ロールへの権限付与
- ユーザー/ロールに対してロールの付与
- デフォルトロールを定義、
  特定のホストのみロールを使用可能
- ROLES_GRAPHML()関数でロールを可視化

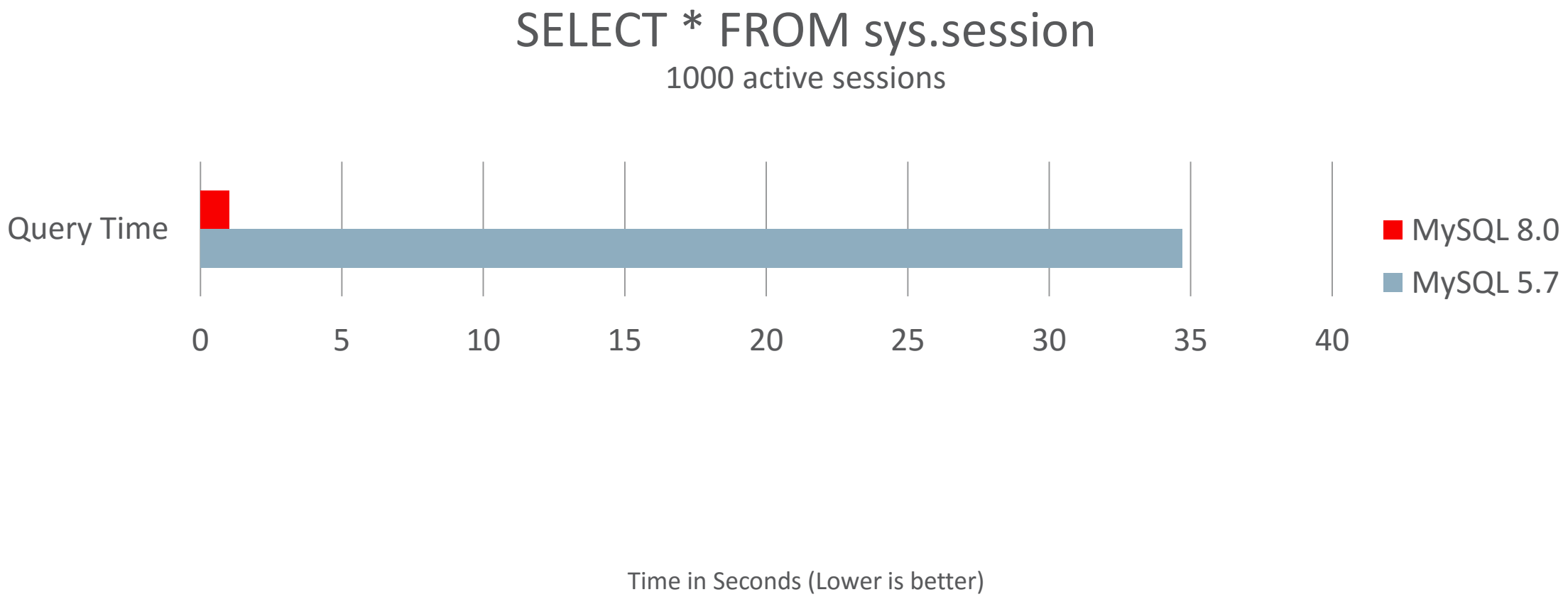


New! パフォーマンス・スキーマ・インデックス

- パフォーマンス・スキーマのテーブルへより効率的なアクセスが可能に
- **93個のテーブル**に対して**全部で115個のインデックスを作成（ハッシュ・インデックス）**
- 追加のオーバーヘッドはほとんど無い
  - 物理的なインデックスのメンテナンスは必要ない
  - オプティマイザーにより良い実行計画を選択させる

パフォーマンスの比較

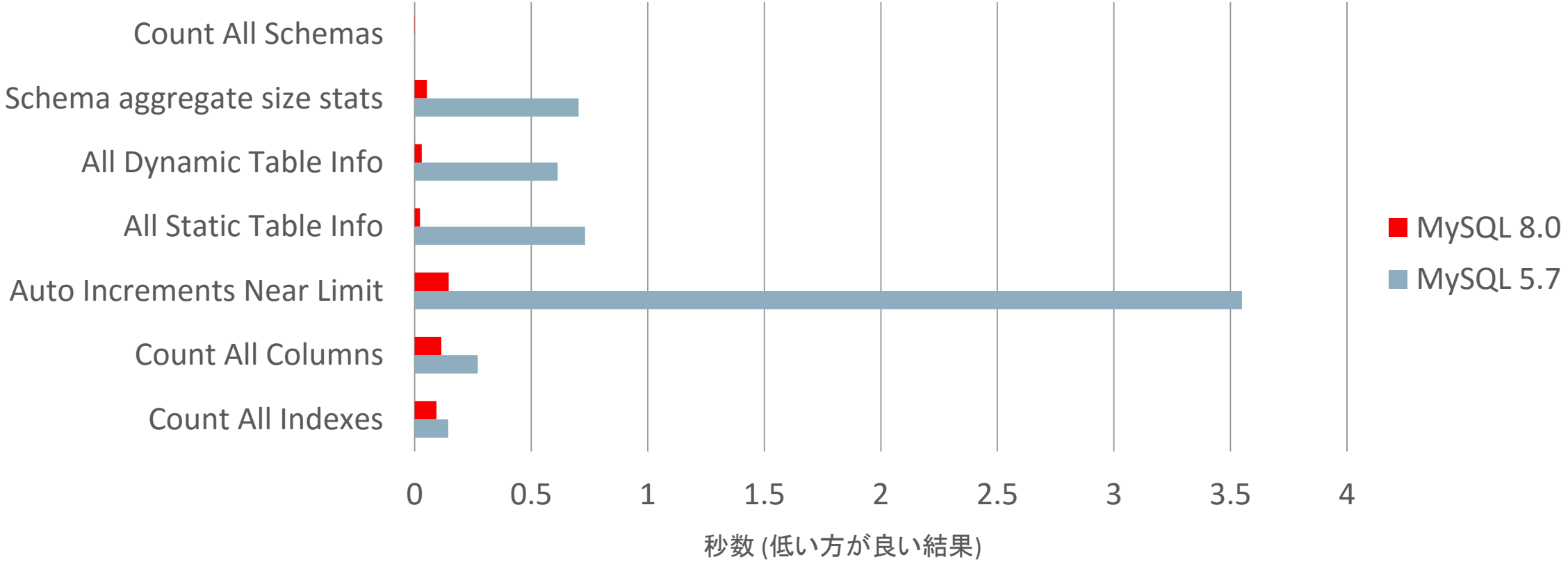
30倍以上高速！



インフォメーション・スキーマのパフォーマンス改善

100 schemas times 50 tables (5000 tables)

既に7/10のクエリーが高速化できている

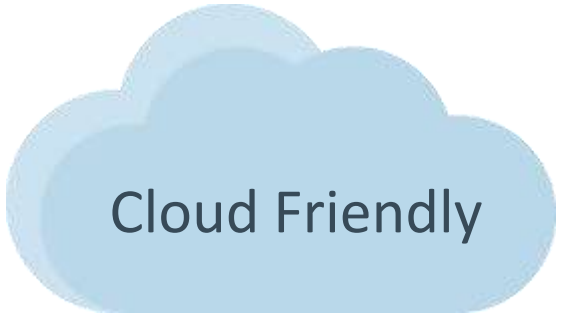


New! エラーを確認できるテーブル追加（パフォーマンス・スキーマ）

Aggregation	Table Name
By Account	events_errors_summary_by_account_by_error
By Host	events_errors_summary_by_host_by_error
By Thread	events_errors_summary_by_thread_by_error
By User	events_errors_summary_by_user_by_error
Global	events_errors_summary_global_by_error

```
SELECT * FROM test.no_table;
ERROR 1146 (42S02): Table 'test.no_table' doesn't exist

SELECT * FROM performance_schema.events_errors_summary_global_by_error
WHERE sum_error_handled > 0 OR SUM_ERROR_RAISED > 0¥G
*************************** 1. row ***************************
     ERROR_NUMBER: 1146
       ERROR_NAME: ER_NO_SUCH_TABLE
        SQL_STATE: 42S02
 SUM_ERROR_RAISED: 1
SUM_ERROR_HANDLED: 0
       FIRST_SEEN: 2016-09-11 20:52:42
        LAST_SEEN: 2016-09-11 20:52:42
1 row in set (0.00 sec)
```

New! 設定変更の永続化



- 以下の構文でシステム変数の変更を永続化可能
  - `SET PERSIST sql_mode='STRICT_TRANS_TABLES,NO_ENGINE_SUBSTITUTION';`
- システム変数変更のためにファイルシステムへのアクセス不要
- read_only=ON⇒OFFへの移行時など、再起動がしばらくできない場合などに便利
  - my.cnfの修正を忘れるリスクの回避
  - my.cnfの書き間違いにより再起動に失敗するリスクの回避
- システム変数がどこで設定されたかを確認出来る情報も追加（performance_schema.variables_info）

New! 設定変更の永続化

```
mysql> SELECT * FROM performance_schema.variables_info WHERE variable_source != 'COMPILED';
+--------------------+-----------------+---------------+-----------+-------------+
| VARIABLE_NAME      | VARIABLE_SOURCE | VARIABLE_PATH | MIN_VALUE | MAX_VALUE   |
+--------------------+-----------------+---------------+-----------+-------------+
| basedir            | GLOBAL          | /etc/my.cnf   | 0         | 0           |
| datadir            | GLOBAL          | /etc/my.cnf   | 0         | 0           |
| foreign_key_checks | DYNAMIC         |               | 0         | 0           |
| log_error          | GLOBAL          | /etc/my.cnf   | 0         | 0           |
| server_id          | GLOBAL          | /etc/my.cnf   | 0         | 4294967295  |
+--------------------+-----------------+---------------+-----------+-------------+
5 rows in set (0.00 sec)

mysql> SHOW GLOBAL VARIABLES LIKE 'sort_buffer_size';
+------------------+--------+
| Variable_name    | Value  |
+------------------+--------+
| sort_buffer_size | 262144 |
+------------------+--------+
1 row in set (0.00 sec)

mysql> SET PERSIST sort_buffer_size=1*1024*1024;
Query OK, 0 rows affected (0.00 sec)
```

New! 設定変更の永続化

```
mysql> SHOW GLOBAL VARIABLES LIKE 'sort_buffer_size';
+------------------+---------+
| Variable_name    | Value   |
+------------------+---------+
| sort_buffer_size | 1048576 |
+------------------+---------+
1 row in set (0.00 sec)

mysql> SELECT * FROM performance_schema.variables_info WHERE variable_source != 'COMPILED';
+--------------------+-----------------+---------------+-----------+----------------------+
| VARIABLE_NAME      | VARIABLE_SOURCE | VARIABLE_PATH | MIN_VALUE | MAX_VALUE            |
+--------------------+-----------------+---------------+-----------+----------------------+
| basedir            | GLOBAL          | /etc/my.cnf   | 0         | 0                    |
| datadir            | GLOBAL          | /etc/my.cnf   | 0         | 0                    |
| foreign_key_checks | DYNAMIC         |               | 0         | 0                    |
| log_error          | GLOBAL          | /etc/my.cnf   | 0         | 0                    |
| server_id          | GLOBAL          | /etc/my.cnf   | 0         | 4294967295           |
| sort_buffer_size   | DYNAMIC         |               | 32768     | 18446744073709551615 |
+--------------------+-----------------+---------------+-----------+----------------------+
6 rows in set (0.00 sec)
```

New! 設定変更の永続化

```
mysql> shutdown;
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
[1]+  終了                    mysqld
[root@yyamasaki mysql]# mysqld &
[1] 2500
[root@yyamasaki mysql]# mysql -u root -p
Enter password:
<<中略>>
mysql> SELECT * FROM performance_schema.variables_info WHERE variable_source != 'COMPILED';
+--------------------+-----------------+-------------------------------------+ <<後略>>
| VARIABLE_NAME      | VARIABLE_SOURCE | VARIABLE_PATH                       | <<後略>>
+--------------------+-----------------+-------------------------------------+ <<後略>>
| basedir            | GLOBAL          | /etc/my.cnf                         | <<後略>>
| datadir            | GLOBAL          | /etc/my.cnf                         | <<後略>>
| foreign_key_checks | DYNAMIC         |                                     | <<後略>>
| log_error          | GLOBAL          | /etc/my.cnf                         | <<後略>>
| server_id          | GLOBAL          | /etc/my.cnf                         | <<後略>>
| sort_buffer_size   | PERSISTED       | /usr/local/mysql/data/mysqld-auto.cnf | <<後略>>
+--------------------+-----------------+-------------------------------------+ <<後略>>
6 rows in set (0.00 sec)
```

New! 設定変更の永続化

```
mysql> exit
Bye
[root@yyamasaki mysql]# more /usr/local/mysql/data/mysqld-auto.cnf
{ "mysql_server": {"sort_buffer_size": "1048576" } }
[root@yyamasaki mysql]#
```

New! トランザクショナルなデータディレクトリ

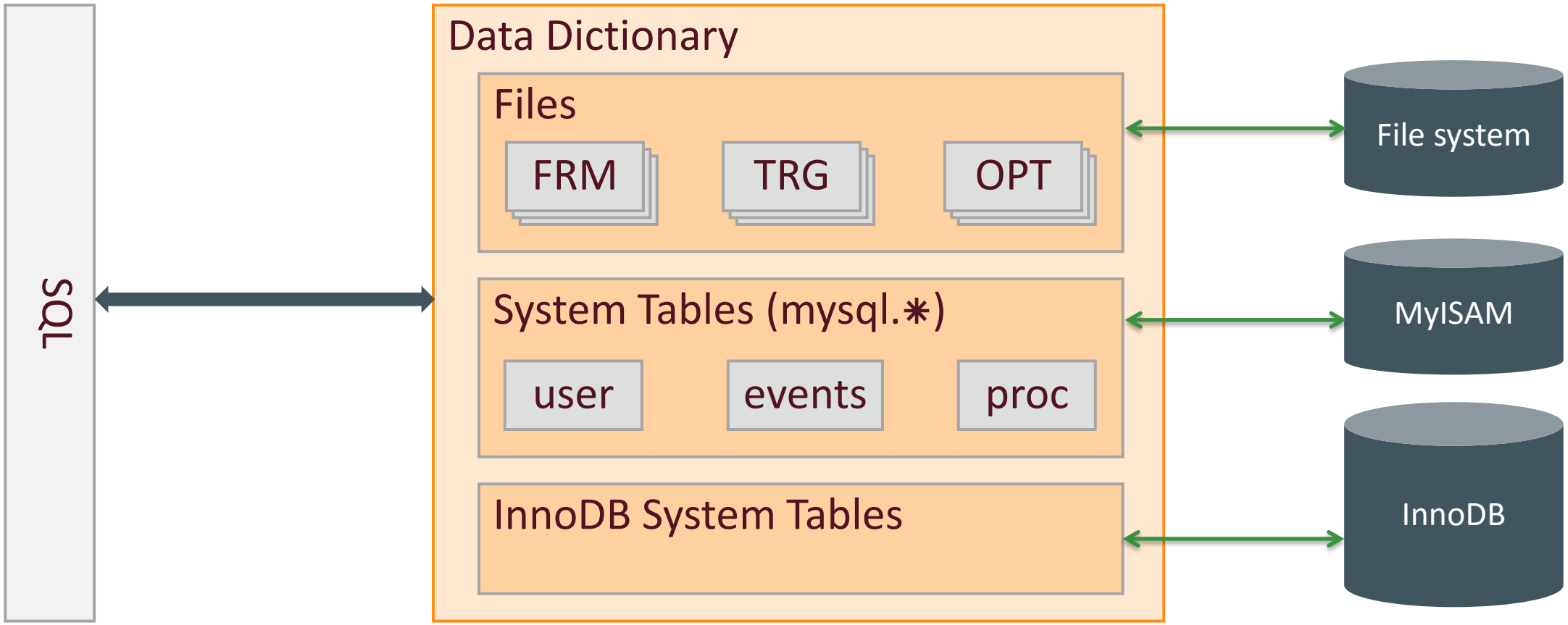
- 信頼性の向上
- InnoDBを使ってデータディレクトリを実現
  - No FRM files
  - No DB.OPT files
  - No TRG files
  - No TRN files
  - No PAR files
- MySQL 8.0のデフォルトインストールでは、MyISAMテーブルを含まない

トランザクショナルなデータディレクトリ

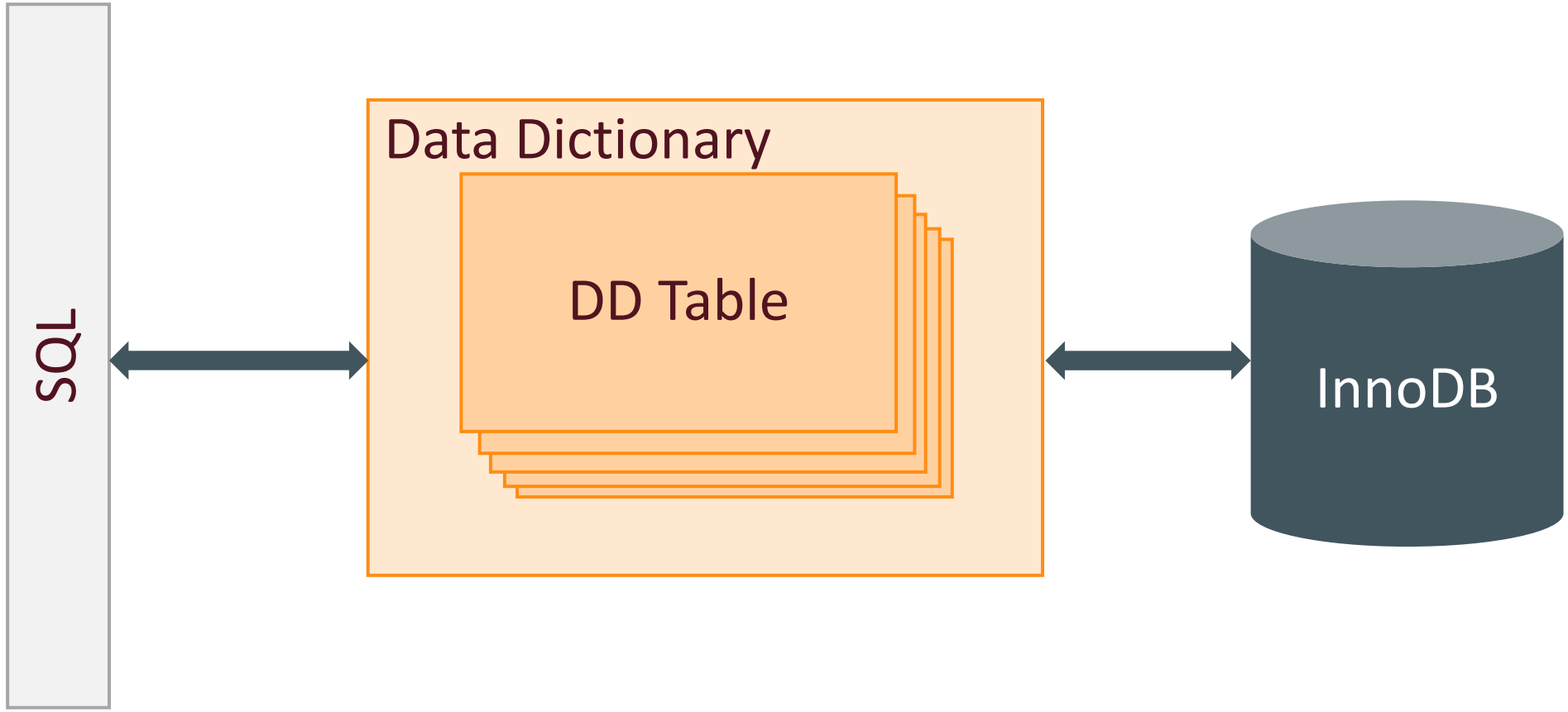
Additional Benefits

- クロスプラットフォームでの相互運用性を向上
  - ファイルシステムに依存しない
- アトミックなDDL
  - レプリケーションの向上
  - クラッシュセーフでない特殊なケースを排除
- MDL for Foreign Keys
- 柔軟なメタデータAPI
  - 新機能の追加を容易に

MySQL Data Dictionary before MySQL 8.0



Transactional Data Dictionary in MySQL 8.0



Labs Common Table Expressions（WITH句）

- WITH句
- 再帰的なクエリの記述
- 複雑なSQLの簡素化

```
WITH t1 AS (SELECT * FROM tblA WHERE a='b')
SELECT * FROM t1;
```

再帰的CTE（Recursive CTE）

```
WITH RECURSIVE cte AS
( SELECT ... FROM table_name /* "seed" SELECT */
   UNION ALL
   SELECT ... FROM cte, table_name) /* "recursive"  SELECT */
SELECT ... FROM cte;/
```

- 再帰的CTEはサブクエリーでCTE自身を参照する
- “seed”のSELECTが1回実行され、初期のデータセットが作成される。その後、完全な結果セットが得られるまで”recursive”のSELECTが繰り返し実行される
- 階層構造の参照に便利（親、子、部分、下位部品）
- Oracle DatabaseのCONNECT BY句に近い機能

再帰的CTEのシンプルな例

```
Print 1 to 10 :

WITH RECURSIVE qn AS
  ( SELECT 1 AS a
    UNION ALL
    SELECT 1+a FROM qn WHERE a<10
  )
SELECT * FROM qn;
```

```
1
2
3
4
5
6
7
8
9
10
```

Labs 降順索引（Descending Indexes）

For B+tree indexes

```
CREATE TABLE t1 (
  a INT,
  b INT,
  INDEX a_b (a DESC, b ASC)
  );
```

- In 5.7: 索引は昇順で作成される。降順でSELECTした時は索引を後ろ向きにスキャンする
- In 8.0: 索引を降順でも作成可能。降順でSELECTした時は索引を前向きにスキャンする

利点:

- 高速（索引は後ろ向きよりも前向きにスキャンしたほうが高速）
- ファイルソートを避けるために索引を使用できるケースの拡大

他にも多くの点を改良

- New! Doxygenを使ったソースコードのドキュメント化
- New! プラグイン・インターフェース
- GIS Supportの拡張
- Query Hints Supportの拡張
- Scan Query Performanceの向上
- BLOB Storageの改善
- Memcached Interfaceの改善
- オプティマイザの改善
- コストモデルの改善
- スケーラビリティの向上
- パーサーのリファクタリング
- 一時テーブルの改善
- C++11とツールチェーンの向上
- レプリケーション・アプライヤー・スレッドの進捗レポート
- gtid_executedに値が入っていても、GTID_PURGEDを設定可能

GA

MySQL Cluster 7.5

ORACLE

MySQL Clusterとは？

- MySQLとは開発ツリーの異なる別製品
- 共有ディスクを使わずに、アクティブ-アクティブのクラスタ構成が組めるインメモリデータベース (一部のデータはディスクに格納することも可能)
- 元々はSQLを使わないデータベースだったが、MySQLと統合されSQLも使えるようになった(NoSQL(KVS)とSQLの両方が使えるデータベース)
  - MySQL Clusterの基礎となる技術は、通信機器ベンダのエリクソンで携帯通信網の加入者データベース向けに開発されたEricsson Network DataBase（NDB）と呼ばれていた技術

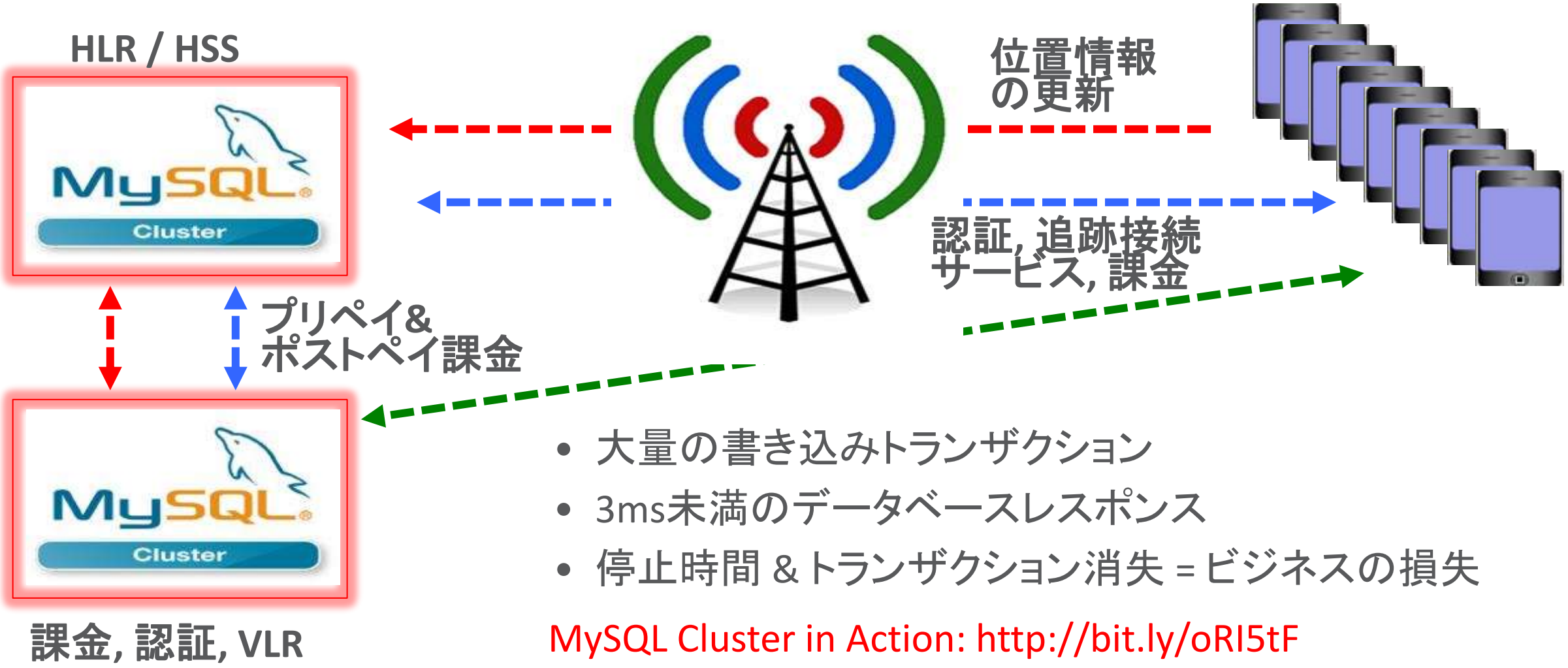
MySQL Clusterが向いているシステム

- 高可用性が求められるシステム
  - 携帯電話の通信インフラを支えるために開発された技術がベース
  - 単一障害点が無い構成
  - 障害発生時に、アプリケーションは処理をリトライすれば存続したノードで処理を継続
- 同時多発的に大量のトランザクションが発生するシステム
  - サーバー台数を増加することで負荷分散可能
- 参照処理だけでなく、更新処理に対しても拡張性が求められるシステム
  - 自動的にデータをシャーディングし、更新処理についても負荷分散可能

MySQL Clusterの特徴

リアルタイム	• 永続性を持つインメモリテーブル ＋ ディスクテーブル • 低レイテンシ
参照更新性能の高い拡張性	• 自動シャーディング、マルチマスタ • ACIDトランザクション、OLTPとリアルタイム分析
99.999% の可用性	• シェアードナッシング、単一障害点無し • 自動復旧、オンラインメンテナンス
SQL + NoSQL	• キー・バリュー型＋複雑なリレーショナルな処理 • SQL + Memcached + Node.js + JavaScript + Java + HTTP/REST & C++
低コスト	• オープンソース＋商用版運用支援ツール • 特殊なハードウェア不要、管理監視ツール群、サポート

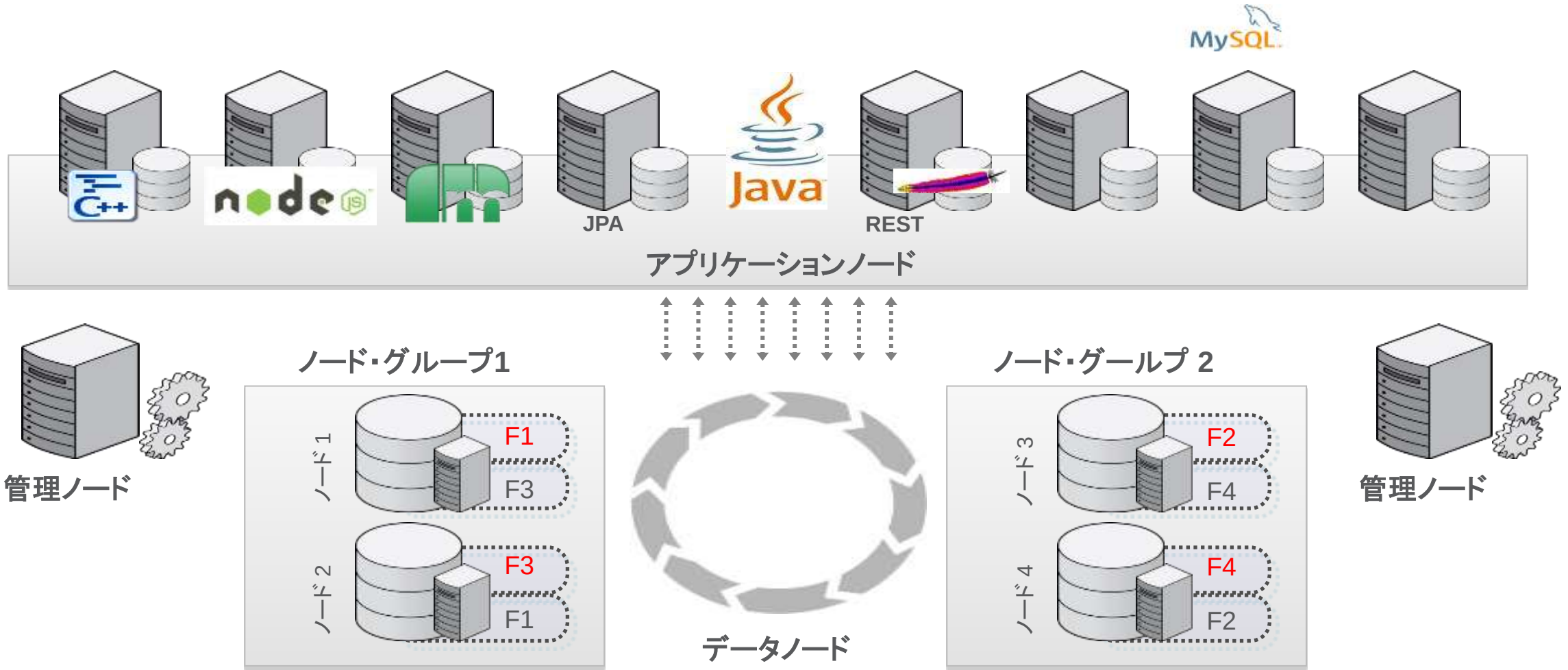
導入事例: 携帯電話ネットワーク



- 大量の書き込みトランザクション
- 3ms未満のデータベースレスポンス
- 停止時間 & トランザクション消失 = ビジネスの損失

MySQL Cluster in Action: http://bit.ly/oRI5tF

MySQL Clusterアーキテクチャ



MySQL Cluster 7.5

GA

MySQL Cluster NDB 7.5.4 (2016-10-18)

- Node Sizes of 128TB
- Read Optimized Tables
- Global Tables

パフォーマンス・キャパシティ

- MySQL 5.7
- JSON Data Type
- Generated Columns
- Records-Per-Key Optimization

SQLの改善

- Improved Reporting
- Improved Logging
- Improved Visibility
- Improved Debugging
- Improved Restore

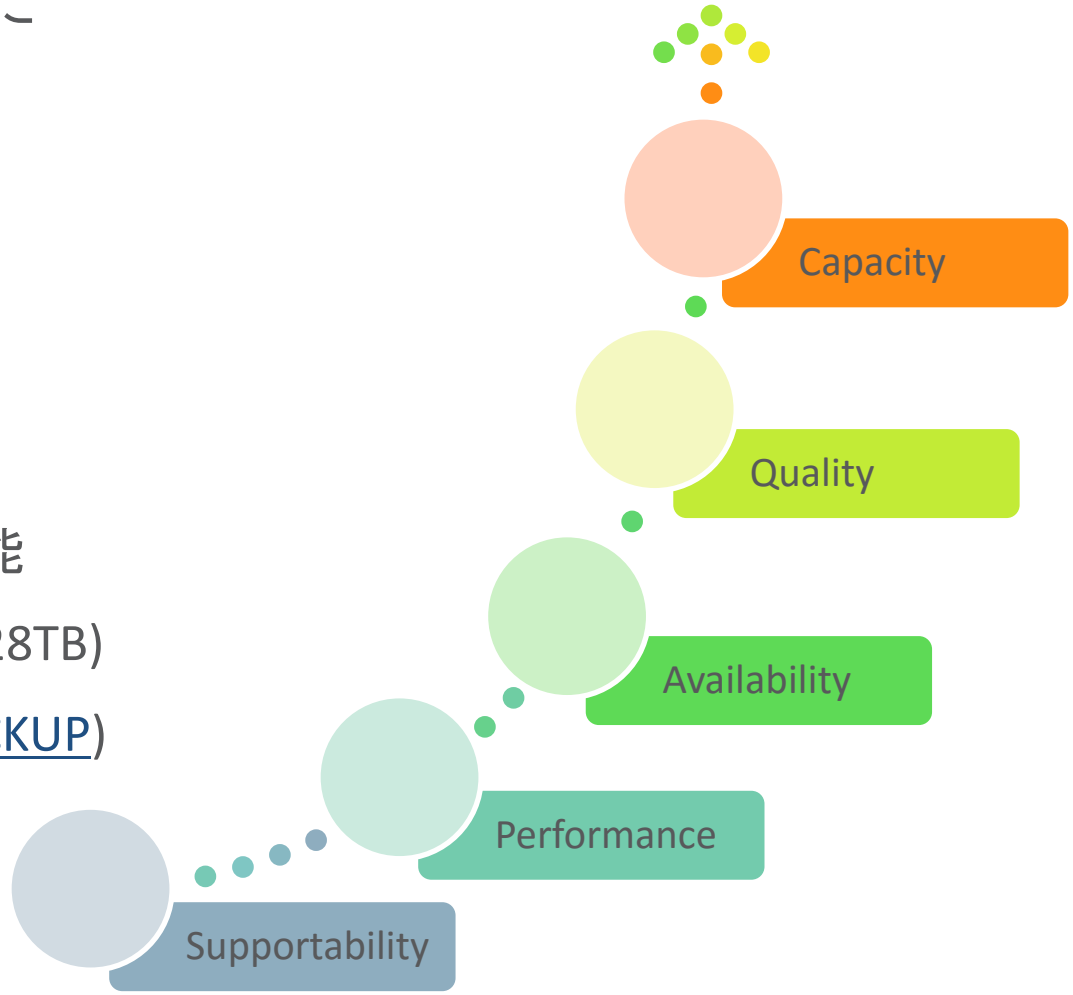
管理の強化

https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster.html
https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster-what-is-new.html

MySQL Cluster 7.5: 機能改善点

GA

- ndbinfoの拡張により、細かい設定をより詳しく確認可能に
- オプティマイザの改善
- エラーメッセージをより具体的に分かり易く改善
- ロック・レポートの改善
- バックアップ＆リストアの高速化
- ndb_binlog_indexをACID対応ストレージエンジンに変換
- スレッドの優先順位づけによる詳細なチューニングも可能
- 固定列フラグメントサイズが16GB以上をサポート(最大128TB)
- テーブル参照性能の向上 (ndb_read_backup, READ_BACKUP)
- MySQL5.7ベースとなり, JSONデータ型を利用可能に
- その他、多数の改善

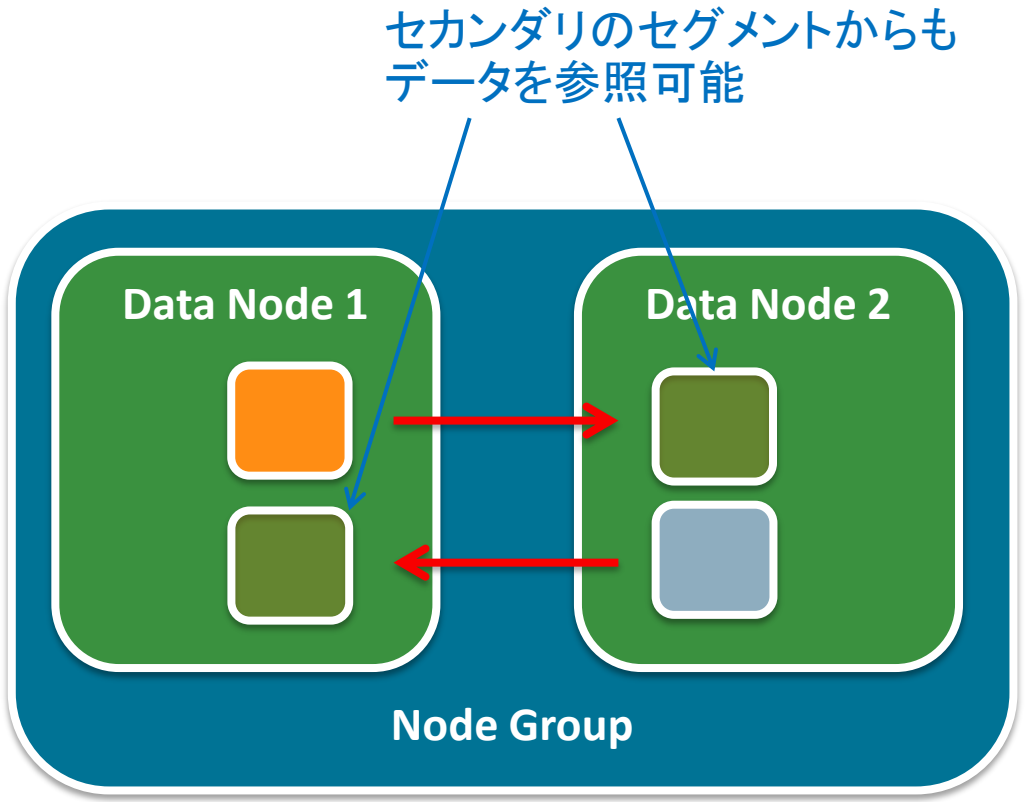


MySQL5.7で実装された機能が利用可能

JSONデータ型やGenerated Columnに対応

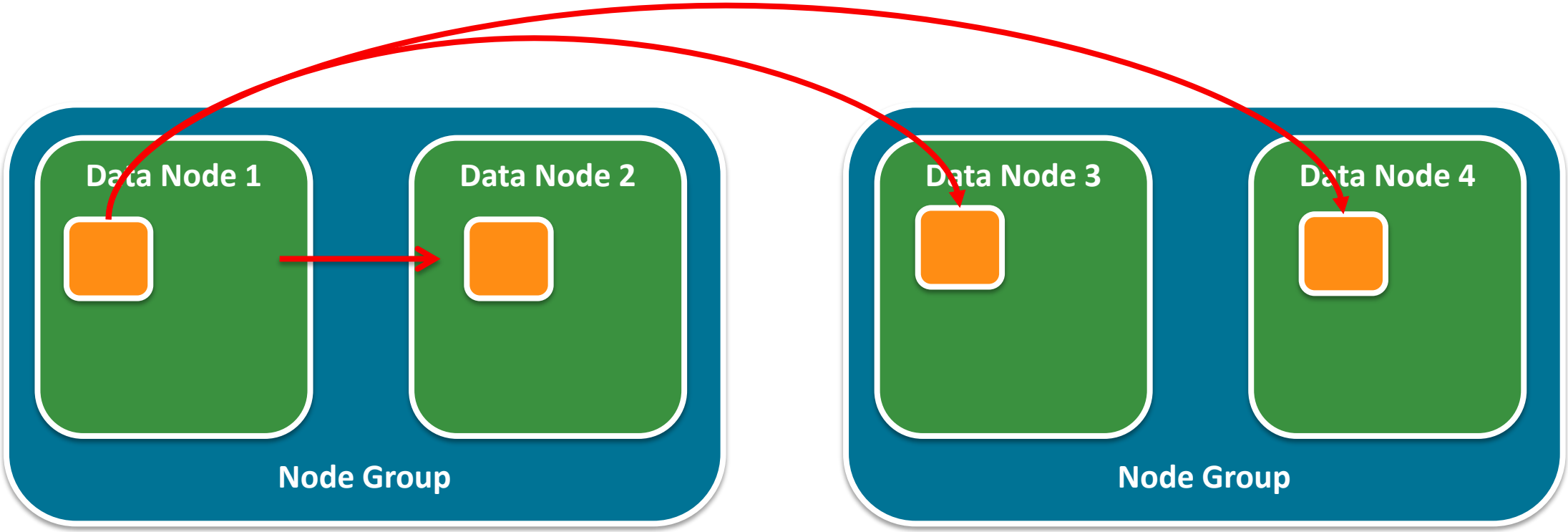
```
mysql> select @@version;
+---------------------------------+
| @@version                       |
+---------------------------------+
| 5.7.13-ndb-7.5.3-cluster-gpl-log |
+---------------------------------+
1 row in set (0.00 sec)

mysql> show create table JSON75¥G
*************************** 1. row ***************************
       Table: JSON75
Create Table: CREATE TABLE JSON75 (
user_id varchar(32) GENERATED ALWAYS AS (json_unquote(json_extract(`doc`,'$.user_id'))) STORED NOT NULL,
doc json DEFAULT NULL,
last_update timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (`user_id`)
) ENGINE=ndbcluster DEFAULT CHARSET=utf8mb4

mysql> select * from JSON75 where user_id = 1;
+---------+-----------------------------------------+---------------------+
| user_id | doc                                     | last_update         |
+---------+-----------------------------------------+---------------------+
| 1       | {“name”: “ユーザーA”, “user_id”: 1}      | 2016-09-30 21:45:47 |
+---------+-----------------------------------------+---------------------+
```

Reading from backup　(ndb_read_backup, READ_BACKUP)



- バックアップからの読み取りは、どのコピーからもデータの読み取りを可能にします。
- 7.4までは、全ての参照処理は、プライマリーフラグメントのみ参照。

Fully replicated (FULLY_REPLICATED)



- Fully replicatedは、対象テーブルに対して、どのデータノードにおいても、ローカルでデータの読み書きが出来るようになります。
- 静的データに最適で、より高速なJOIN処理を実現可能。

参考) COMMETによるテーブルのオプション設定と確認

```
mysql> show create table T_NDB01¥G
*************************** 1. row ***************************
       Table: T_NDB01
Create Table: CREATE TABLE `T_NDB01` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `comment` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=ndbcluster AUTO_INCREMENT=11
  DEFAULT CHARSET=utf8mb4 COMMENT='NDB_TABLE=READ_BACKUP=1
  PARTITION_BALANCE=FOR_RA_BY_LDM'
```

```
mysql> show create table T_NDB02¥G
*************************** 1. row ***************************
       Table: T_NDB02
Create Table: CREATE TABLE `T_NDB02` (
  `id` int(11) unsigned NOT NULL AUTO_INCREMENT,
  `comment` varchar(100) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=ndbcluster AUTO_INCREMENT=11
  DEFAULT CHARSET=utf8mb4 COMMENT='FULLY_REPLICATED=1
  NDB_TABLE=READ_BACKUP=1 PARTITION_BALANCE=FOR_RP_BY_NODE'
```

```
mysql> SELECT TABLE_NAME, TABLE_SCHEMA, TABLE_COMMENT
    -> FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_NAME like "T_NDB0_";
+------------+--------------+------------------------------------------------------------------------------+
| TABLE_NAME | TABLE_SCHEMA | TABLE_COMMENT                                                                |
+------------+--------------+------------------------------------------------------------------------------+
| T_NDB00    | TEST_DB      |                                                                              |
| T_NDB01    | TEST_DB      | NDB_TABLE=READ_BACKUP=1 PARTITION_BALANCE=FOR_RA_BY_LDM                      |
| T_NDB02    | TEST_DB      | FULLY_REPLICATED=1 NDB_TABLE=READ_BACKUP=1 PARTITION_BALANCE=FOR_RP_BY_NODE  |
+------------+--------------+------------------------------------------------------------------------------+
```

参照: https://dev.mysql.com/doc/refman/5.7/en/create-table-ndb-table-comment-options.html

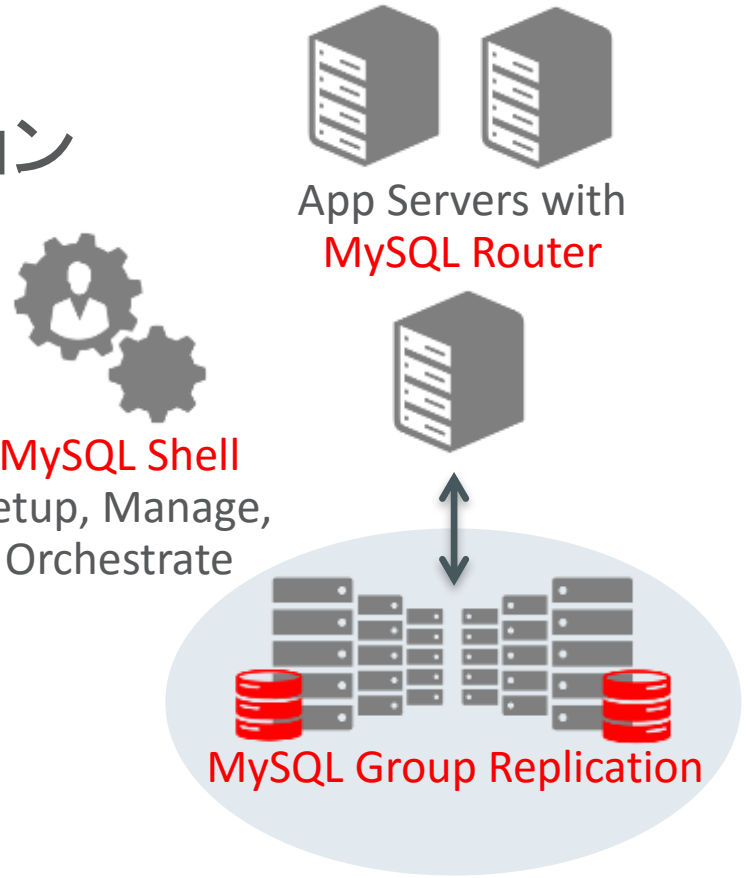
RC

MySQL Group Replication

ORACLE

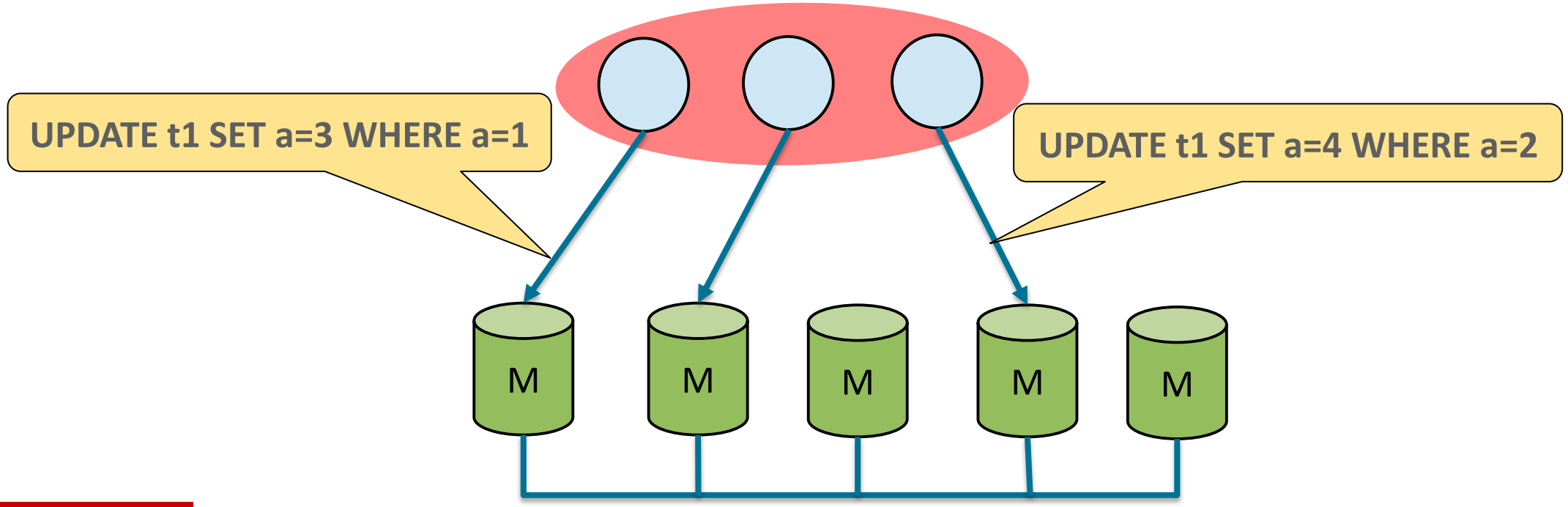
MySQL Group Replication

- マルチマスターレプリケーション用プラグイン
- MySQL5.7以降で利用可能な仮想同期レプリケーション
- Active/Active構成でどのノードからでも更新可能
- 分散型フォールトトレランス
- 自動的にサーバーのフェイルオーバーを行う
- 自動的にレプリケーションを再構成
- 自動的に競合検出＆解消
- MySQLがサポートする全てのプラットフォームに対応
  - Linux, Windows, Solaris, OSX, FreeBSD



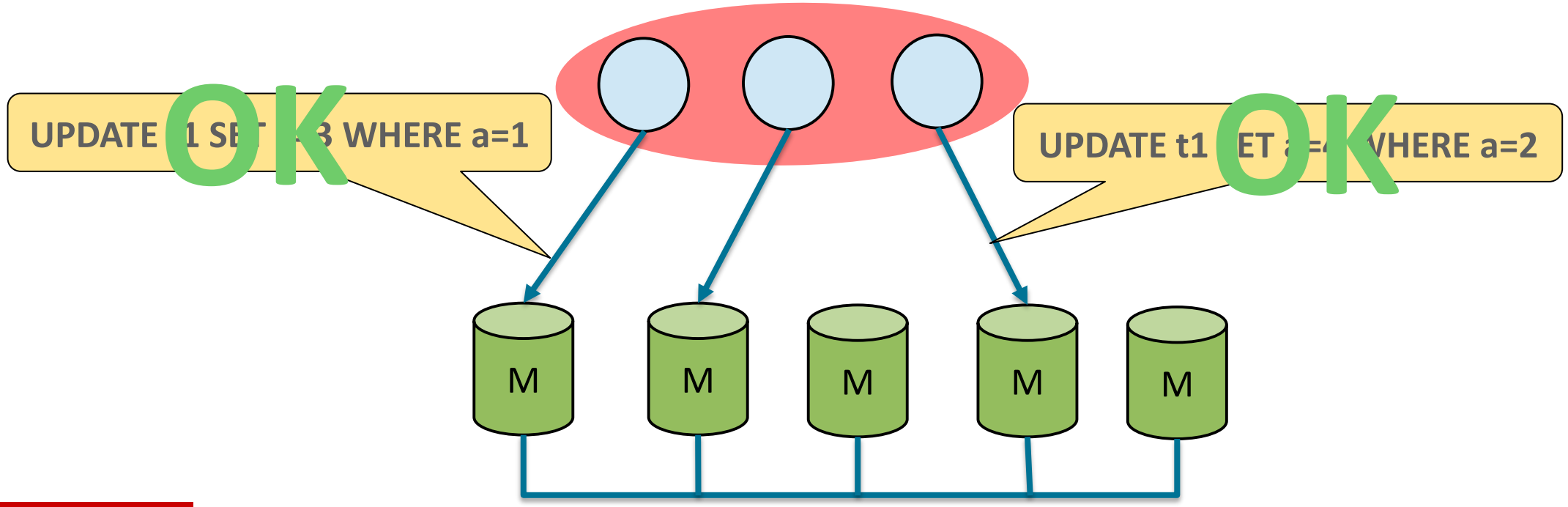
マルチマスター、どこでも更新可能！

- 異なるサーバー上での2つのトランザクションは、同じデータを更新可能
- 競合が検出された場合、自動的に対処される
  - 先にコミットしたトランザクションが優先される



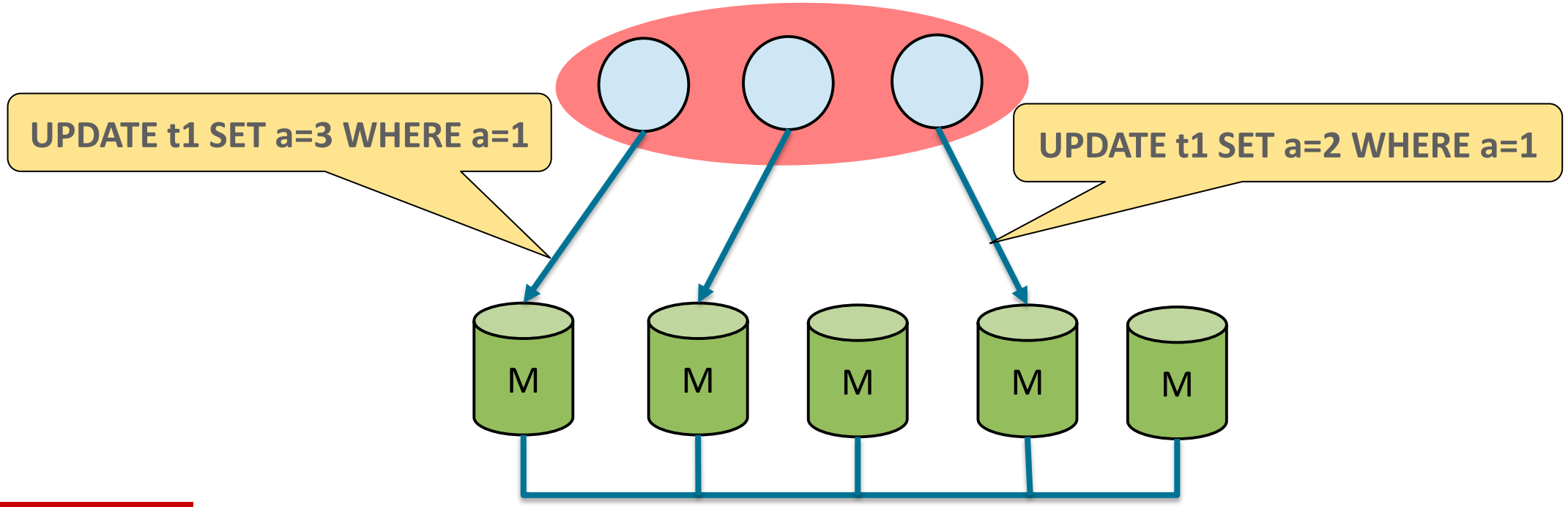
マルチマスター、どこでも更新可能！

- 異なるサーバー上での2つのトランザクションは、同じデータを更新可能
- 競合が検出された場合、自動的に対処される
  - 先にコミットしたトランザクションが優先される



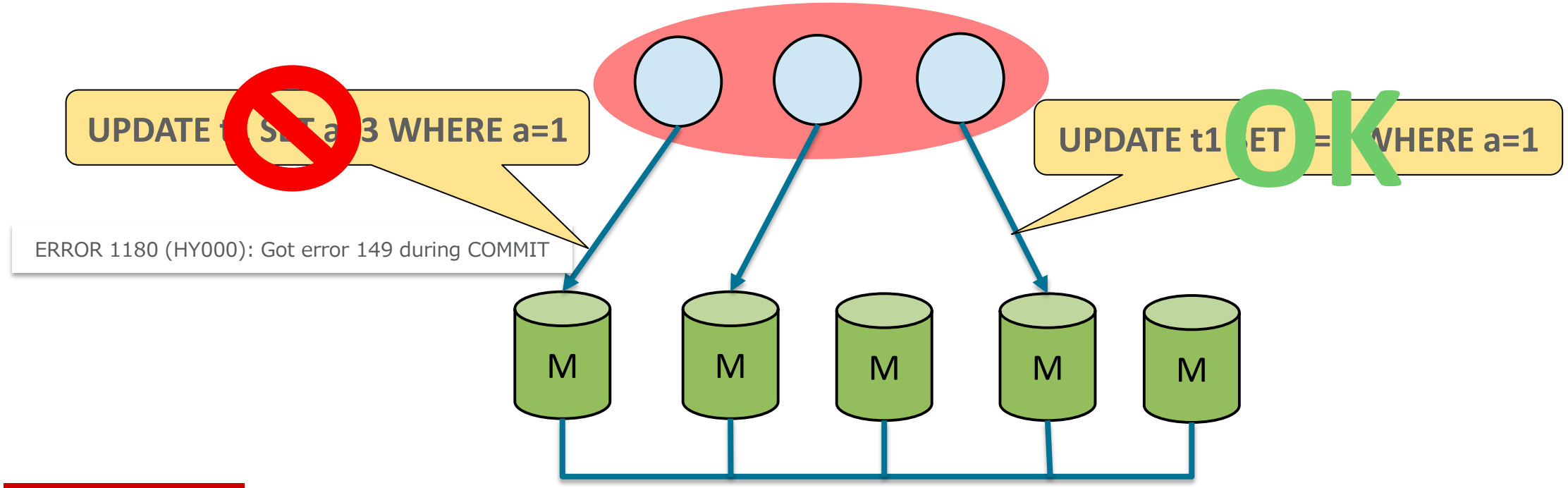
マルチマスター、どこでも更新可能！

- 異なるサーバー上での2つのトランザクションは、同じデータを更新可能
- 競合が検出された場合、自動的に対処される
  - 先にコミットしたトランザクションが優先される



マルチマスター、どこでも更新可能！

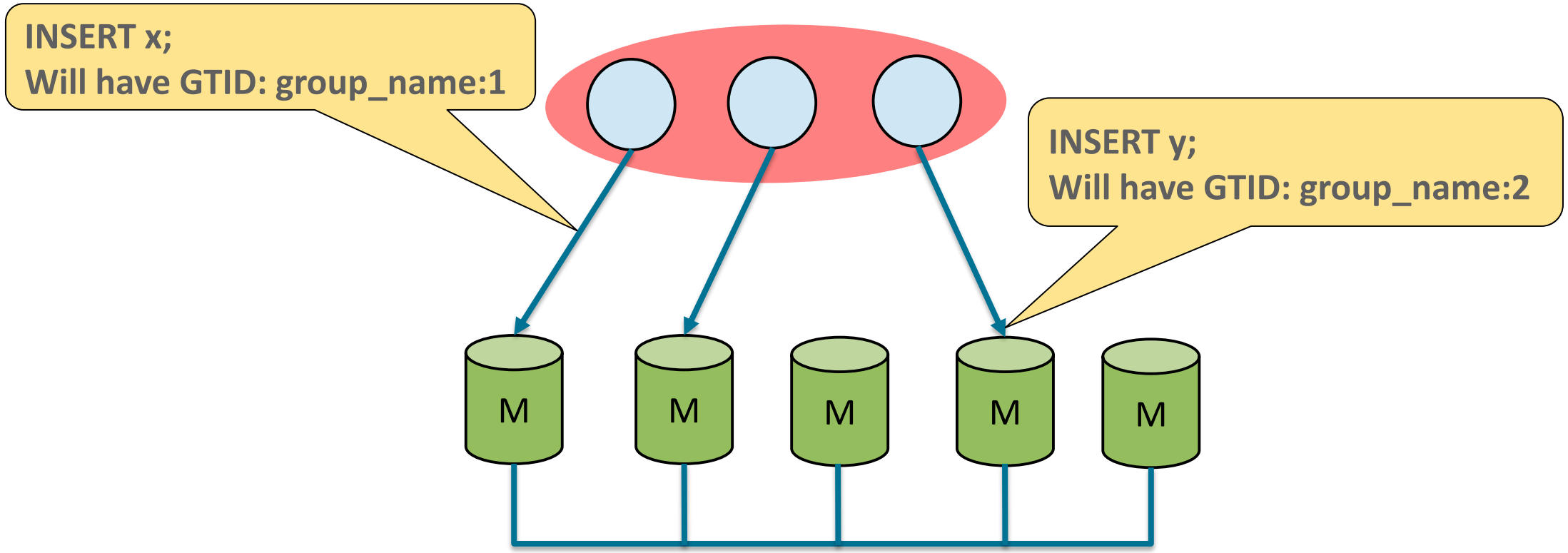
- 異なるサーバー上での2つのトランザクションは、同じデータを更新可能
- 競合が検出された場合、自動的に対処される
  - 先にコミットしたトランザクションが優先される



Full GTIDサポート

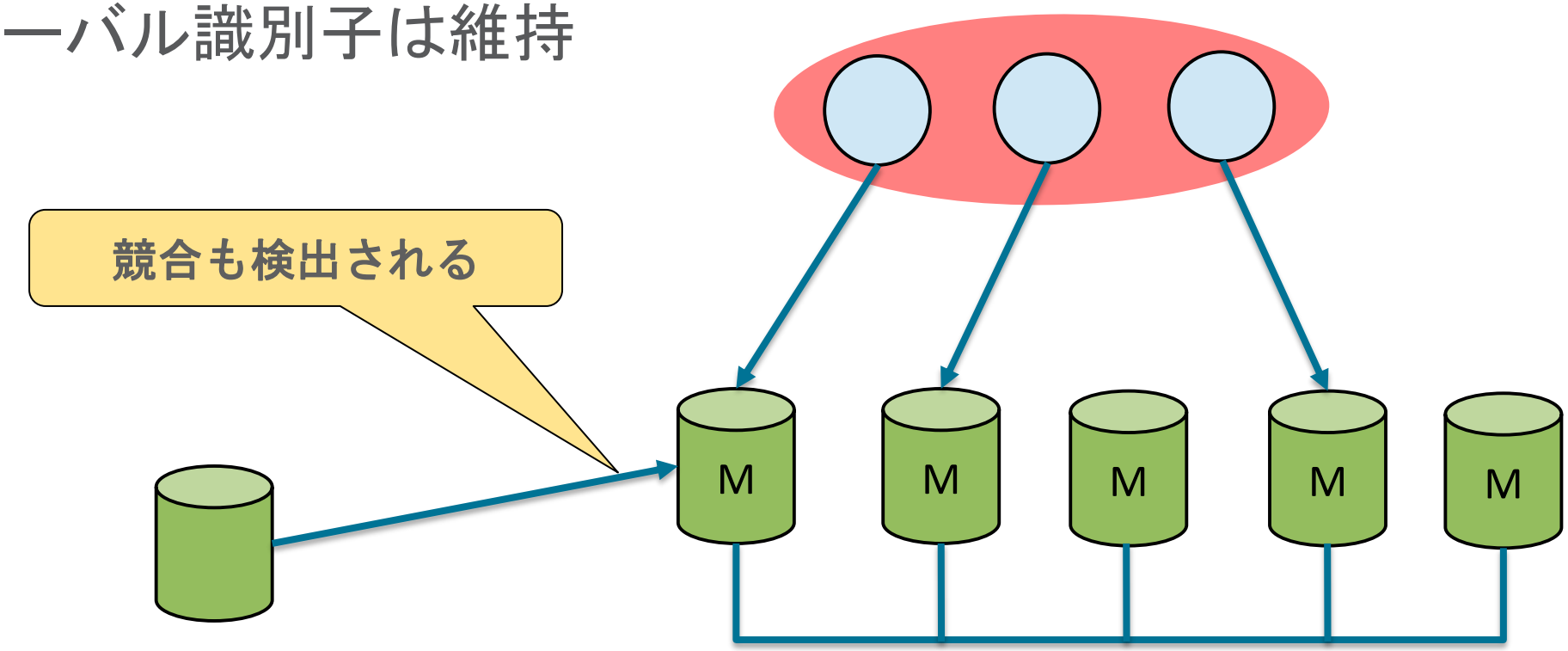
- 全グループメンバーは同じUUID、グループ名を持つ

```
mysql> SET GLOBAL group_replication_group_name= "00000000-1111-2222-3333-123456789ABC";
```



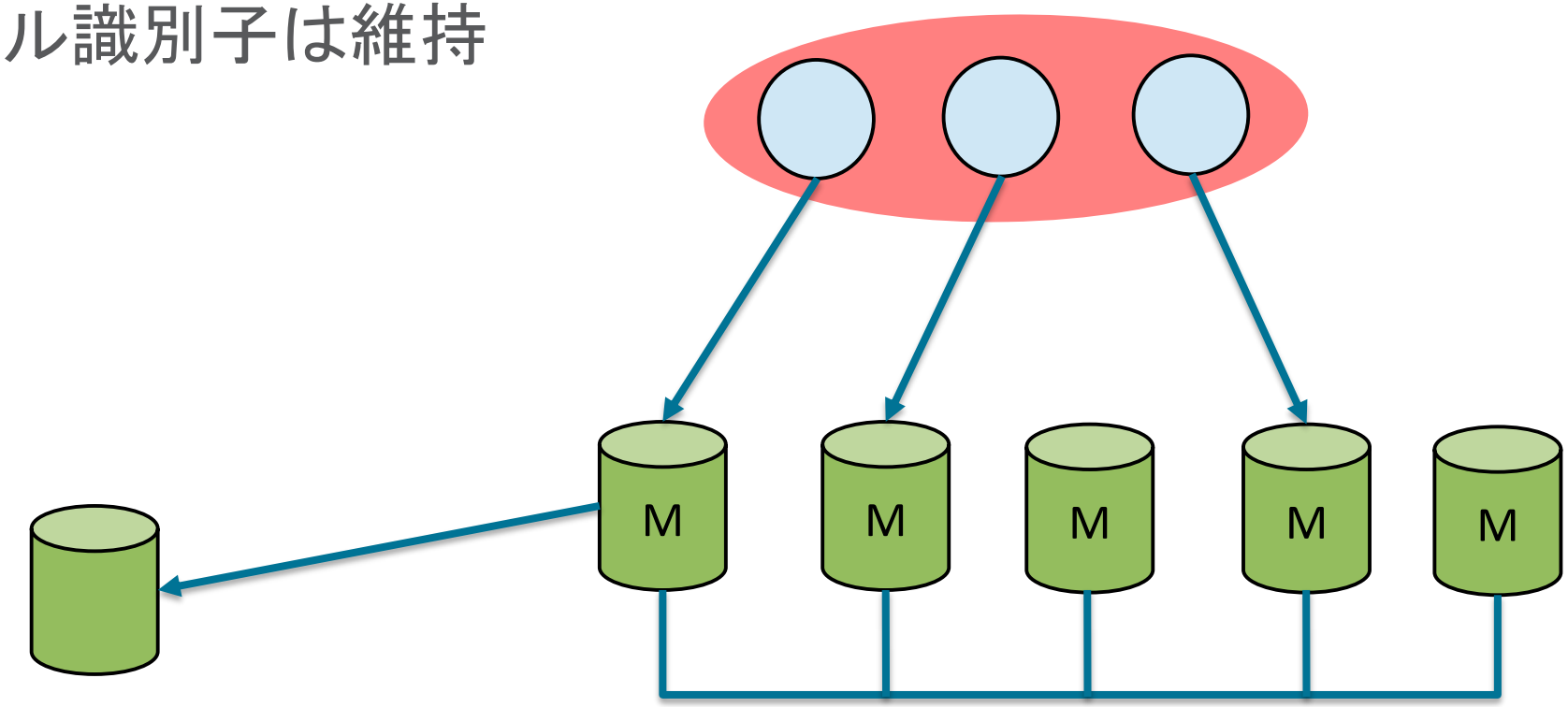
Full GTIDサポート

- グループ外のサーバーからグループにレプリケート可能
- グローバル識別子は維持



Full GTIDサポート

- グループからグループ外のサーバーにレプリケート可能
- グローバル識別子は維持



Single Primary Mode

- シングルメンバーが書き込み可能なマスター(PRIMARY)として動作して、他のグループメンバーはホットスタンバイ(SECONDARY)として動作します。
  - リーダー選択メカニズムにて、どのメンバーがPRIMARYとして動作すべきか、判断して自動的にグループ内でコーディネートします。
- この新しいコンフィギュレーションモード導入の背後にある主な動機は、グループレプリケーションユーザーに対して、ユーザーが使い慣れている非同期レプリケーションに近い、シングルマスターレプリケーションの環境を選択出来るようにする為

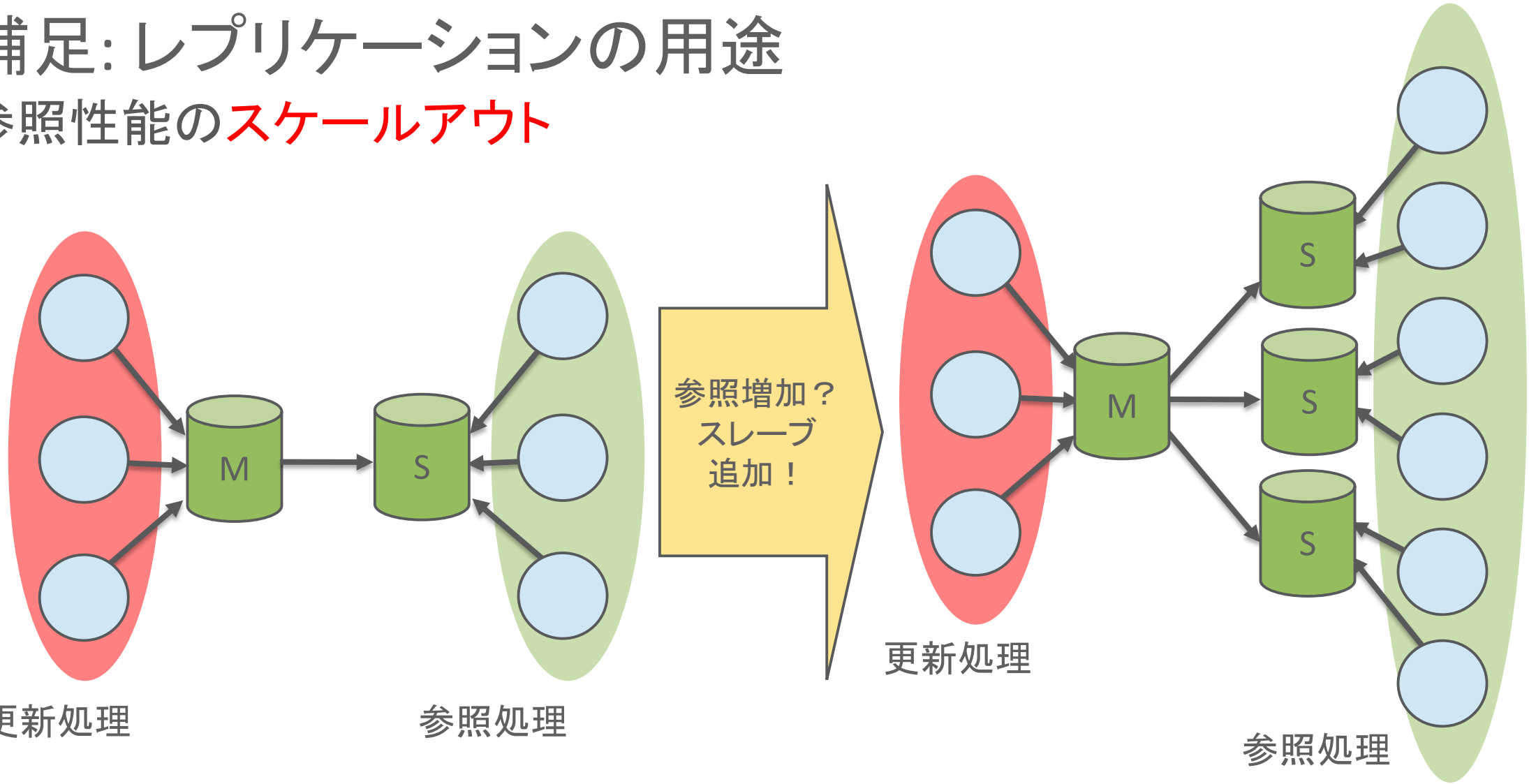
```
+----------------------------------------+-------+
| Variable_name                          | Value |
+----------------------------------------+-------+
| group_replication_single_primary_mode | ON    |
+----------------------------------------+-------+
```

```
+-----------------------------------------------------+-------+
| Variable_name                                       | Value |
+-----------------------------------------------------+-------+
| group_replication_enforce_update_everywhere_checks | OFF   |
+-----------------------------------------------------+-------+
```

[1]: http://mysqlhighavailability.com/mysql-group-replication-for-mysql-5-7-15/

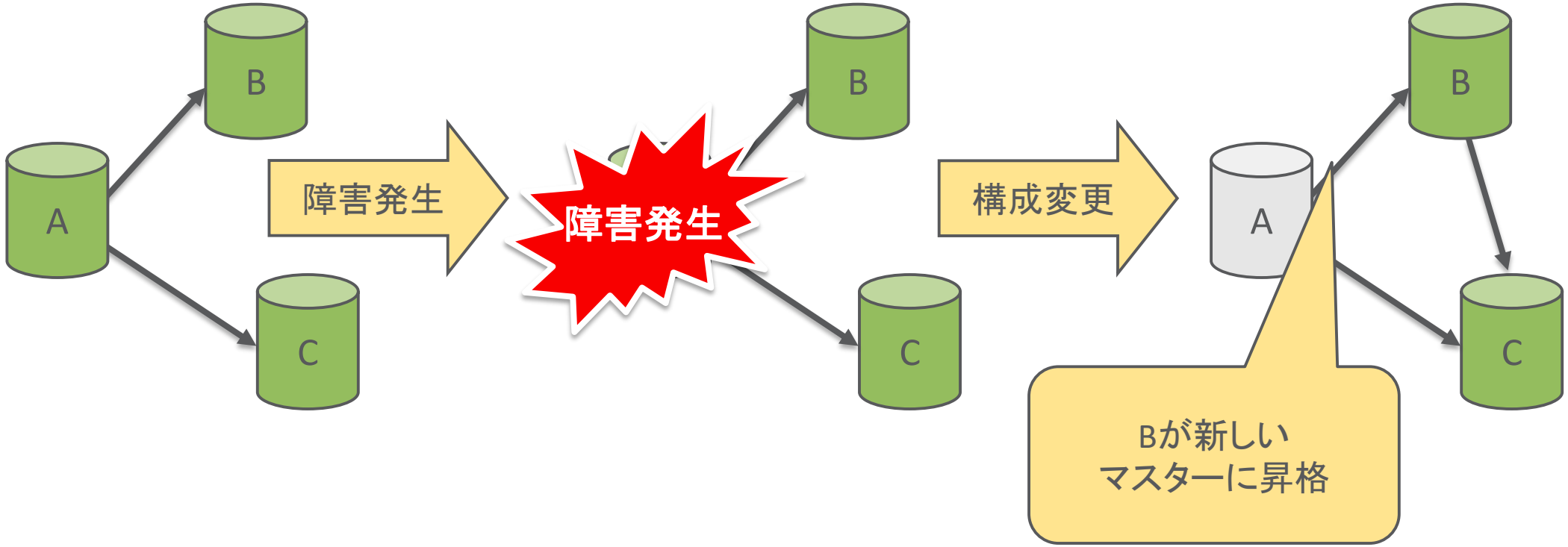
補足: レプリケーションの用途

参照性能のスケールアウト



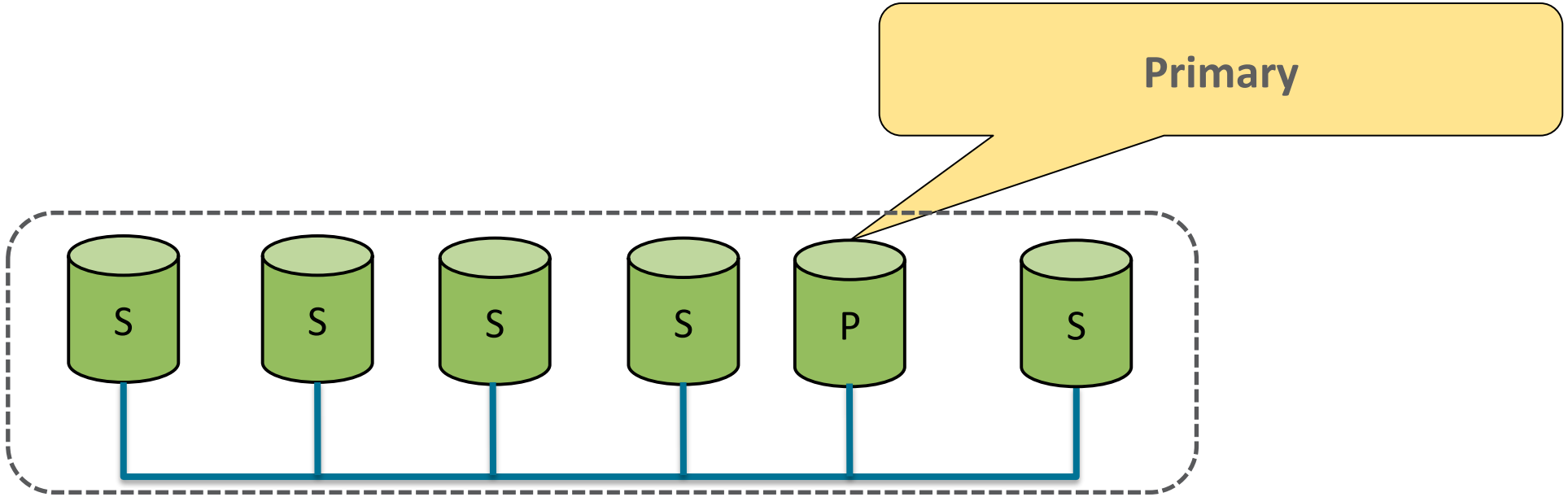
補足: レプリケーションの用途

冗長性: マスターがクラッシュした場合, スレーブをマスターに昇格



Single Primary Mode

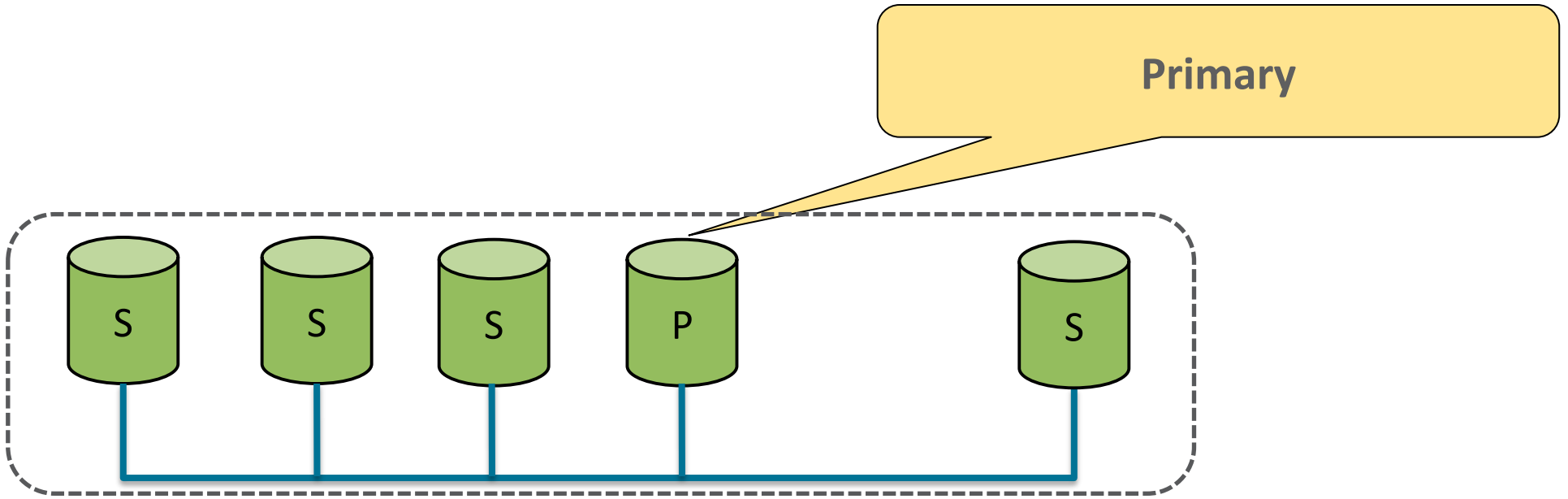
- 自動的なリーダー選択メカニズム
  - Secondaryノードは参照のみ可能 (書き込んだ場合: ERROR 1290)




```
ERROR 1290 (HY000): The MySQL server is running with the --super-read-only option so it cannot execute this statement
```

Single Primary Mode

- 自動的なリーダー選択メカニズム



[1]: http://mysqlhighavailability.com/mysql-group-replication-for-mysql-5-7-15/

MySQL InnoDB Cluster

MySQL InnoDB Clusterとは？

- 以下のコンポーネントを組み合わせから構成される MySQLの高可用性フレームワーク
  - MySQL Group Replication
  - MySQL Router
  - MySQL Shell

Ease-of-Use

Built-in HA

MySQL
InnoDB
cluster

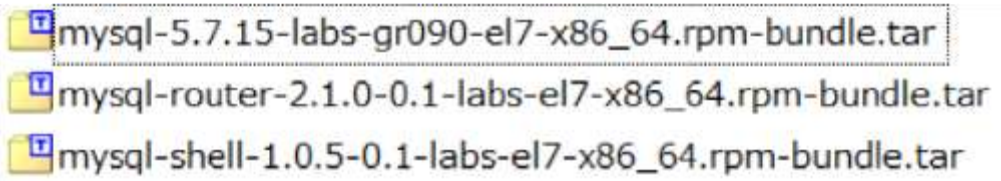
Out-of-Box Solution

Everything Integrated

Scale-Out

High Performance

MySQL Labs :: MySQL InnoDB Cluster 5.7.15 Preview
http://labs.mysql.com/



Ease-of-Use

- 15分でインストール, HA ,スケールアウト設定が可能
- MySQLユーザーの為のシングルインターフェイス
- 簡単にセットアップ, スケールアウト, 管理 & モニタリング
- 優れた品質

Out-of-Box Solution

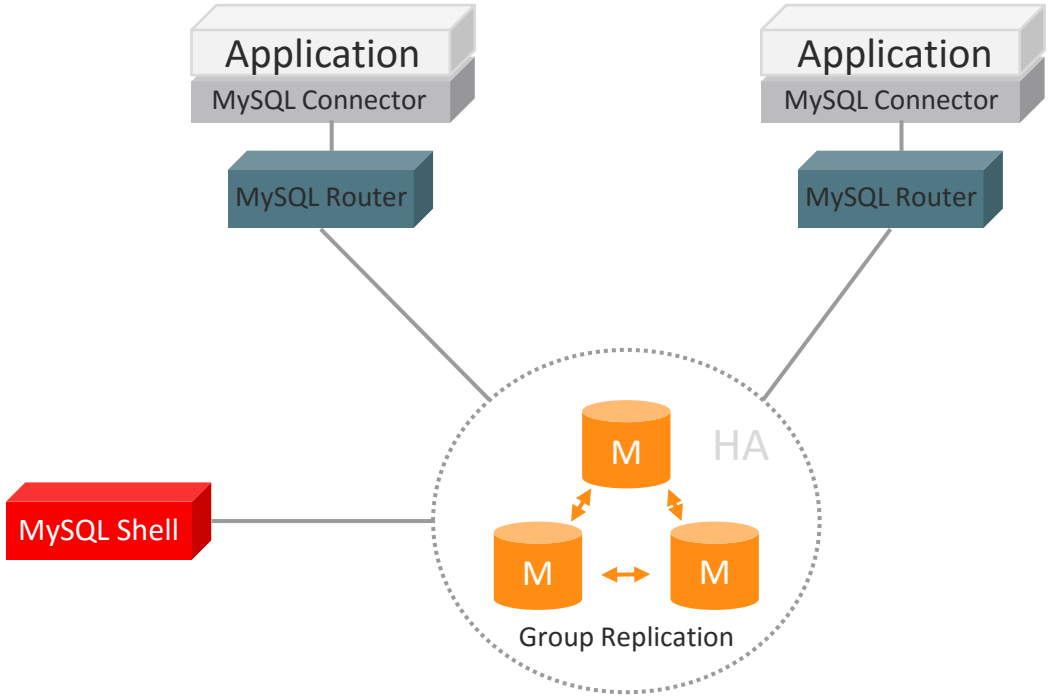
- 統合ソリューション vs. 個別のコンポーネント
- 設計& 開発済み環境を同時に提供
- 検証済み環境を同時に提供
- 管理及び監視環境を同時に提供

Scale-Out

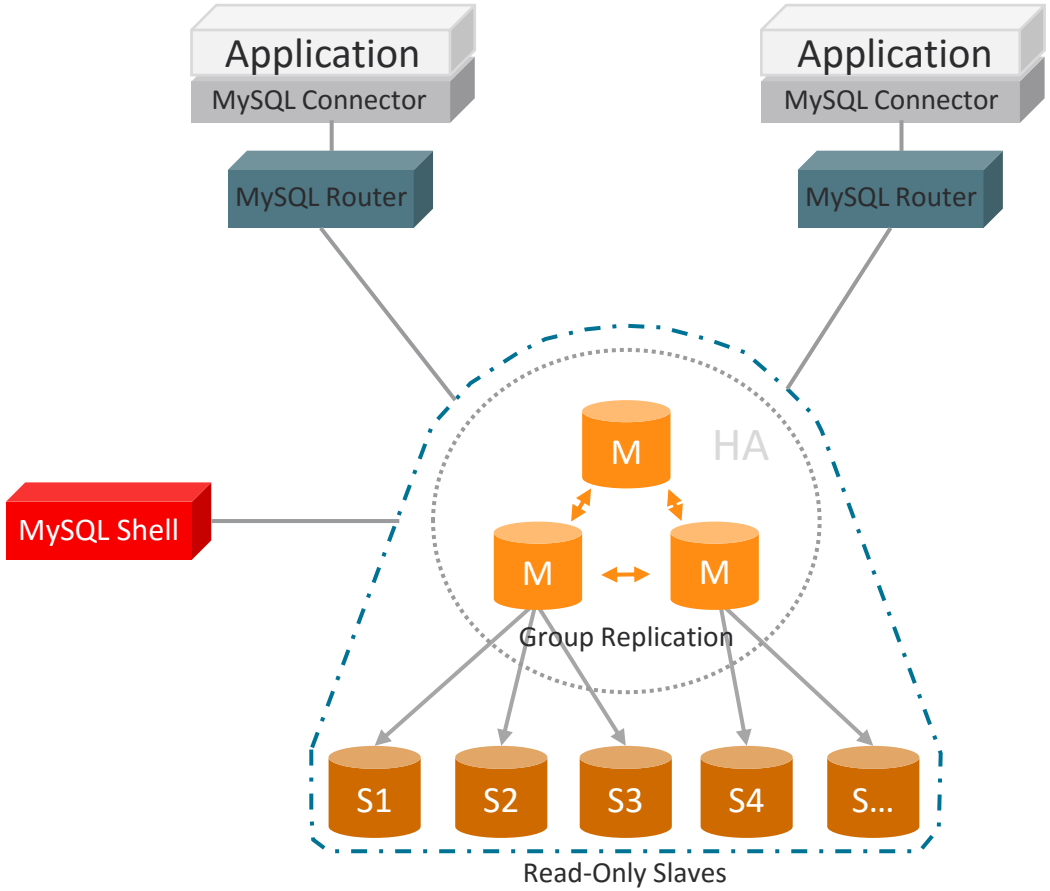
- ワールドクラスの性能を維持
- 自動フェイルオーバー含め信頼性のあるHAをサーバー側で提供
- 参照処理の拡張:レプリケーション
- 書き込み処理の拡張:シャーディング

ORACLE

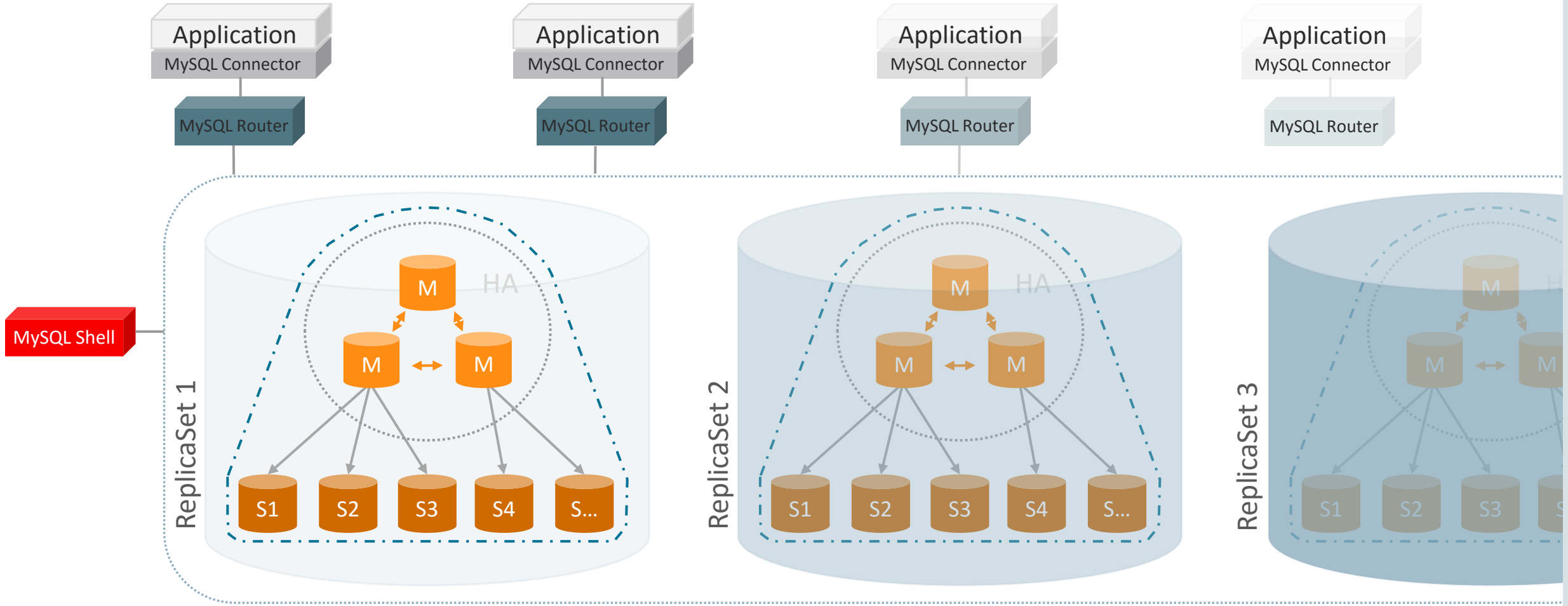
MySQL InnoDB Cluster: Architecture – Step 1



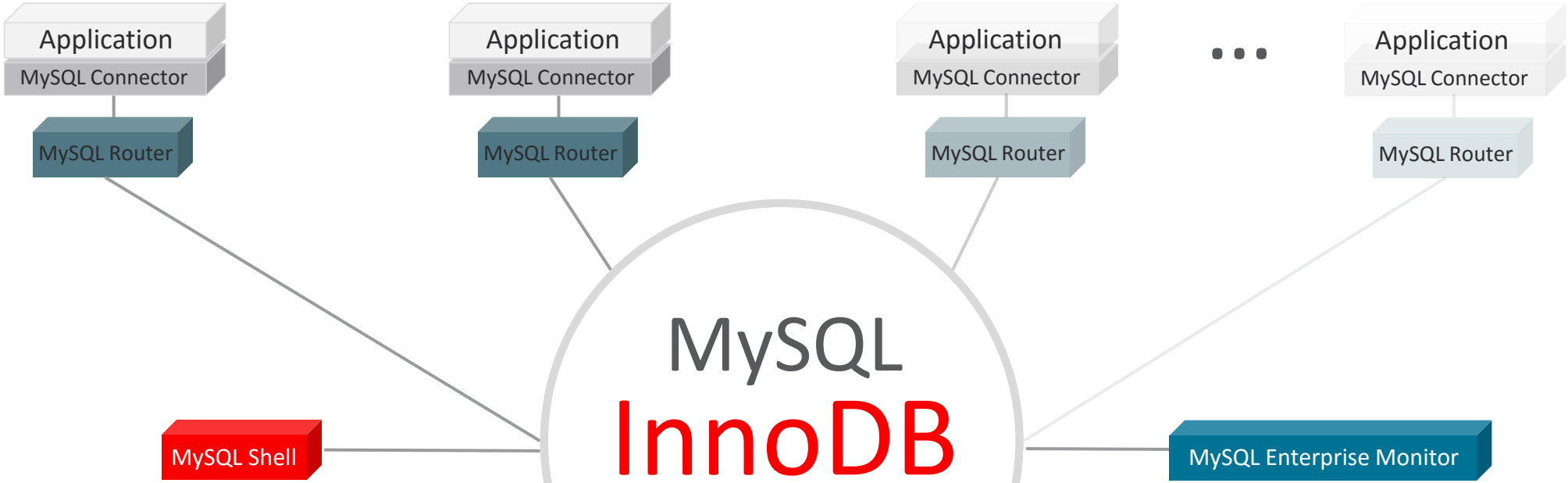
MySQL InnoDB Cluster: Architecture – Step 2



MySQL InnoDB Cluster: Architecture – Step 3

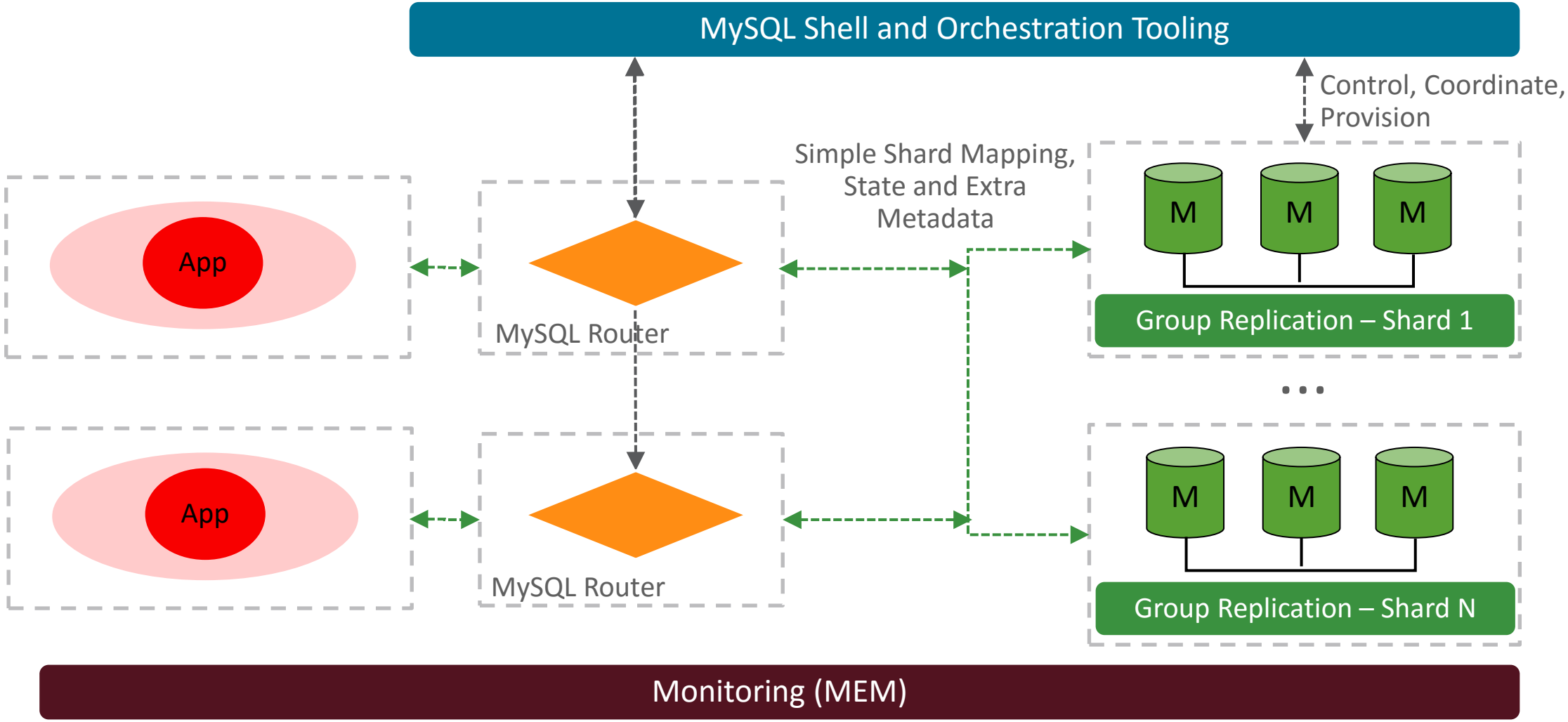


MySQL InnoDB Cluster: High Level Architecture




```
mysql-js> cluster.status()
 {
   "clusterName": “NewAppCluster",
   "defaultReplicaSet": {
      "status": "Cluster tolerant to up to ONE failure.",
      "topology": {
          “hanode1:3306": {
              "address": “hanode1:3306",
              "status": "ONLINE",
              "role": "HA",
              "mode": "R/W",
```

参考) MySQL InnoDB Cluster: The End Goal



MySQL InnoDB Cluster: ゴール

単一製品: MySQL

- 全てのコンポーネントを同時に開発
- 全てのコンポーネントを同時に検証
- 一つのパッケージとして提供

近代的な柔軟性

- C++ 11 (ISO標準 ISO/IEC 14882:2011)
- Protocol Buffers
- 開発フレンドリー

容易な利用

- シングルクライアント: MySQL Shell
- 容易なパッケージング
- 同種のサーバー群

スケールアウト

- シャード・クラスター
- Nレプリカセットのフェデレーテッド構成
- 各レプリカセットによるシャードの管理

デモ: https://www.youtube.com/watch?v=JWy7ZLXxtZ4

DMR

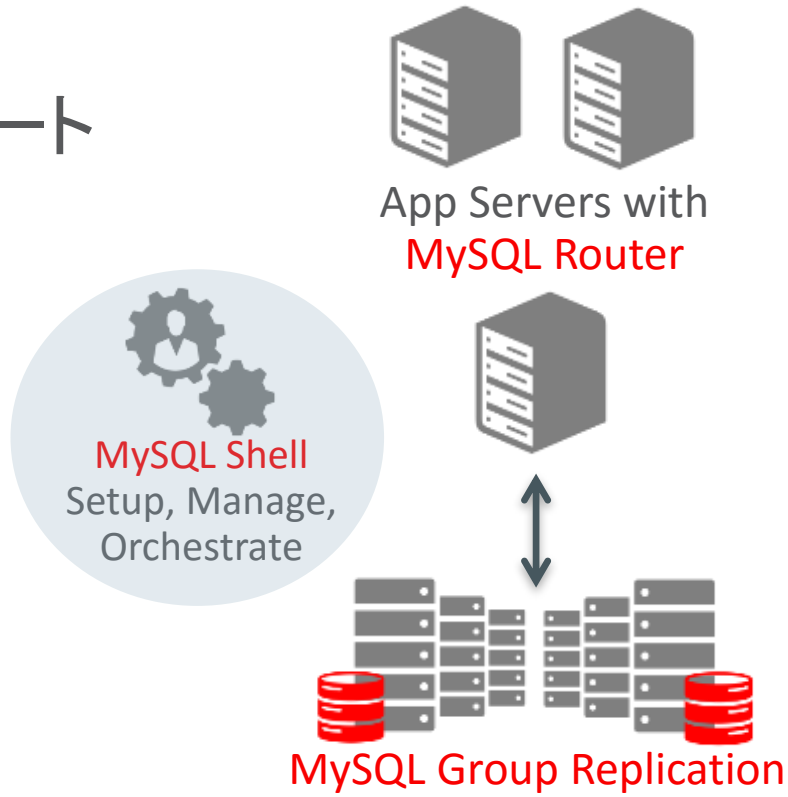
MySQL Shell

すべての運用管理タスクのための、統一された単一クライアント

MySQL Shell DMR (2016-04)

- 多言語対応: JavaScript, Python, and SQL
- ドキュメントとリレーショナルモデルの両方をサポート
- 開発と管理用に完全なAPIを提供

”MySQL Shell provides the developer and DBA with a single intuitive, flexible, and powerfull interface for all MySQL related tasks!”




```
[root@misc01 admin]# mysqlsh --help | egrep -i "Start in"
  --sql                 Start in SQL mode using a node session.
  --sqlc                Start in SQL mode using a classic session.
  --js                  Start in JavaScript mode.
  --py                  Start in Python mode.
[root@misc01 admin]#
```

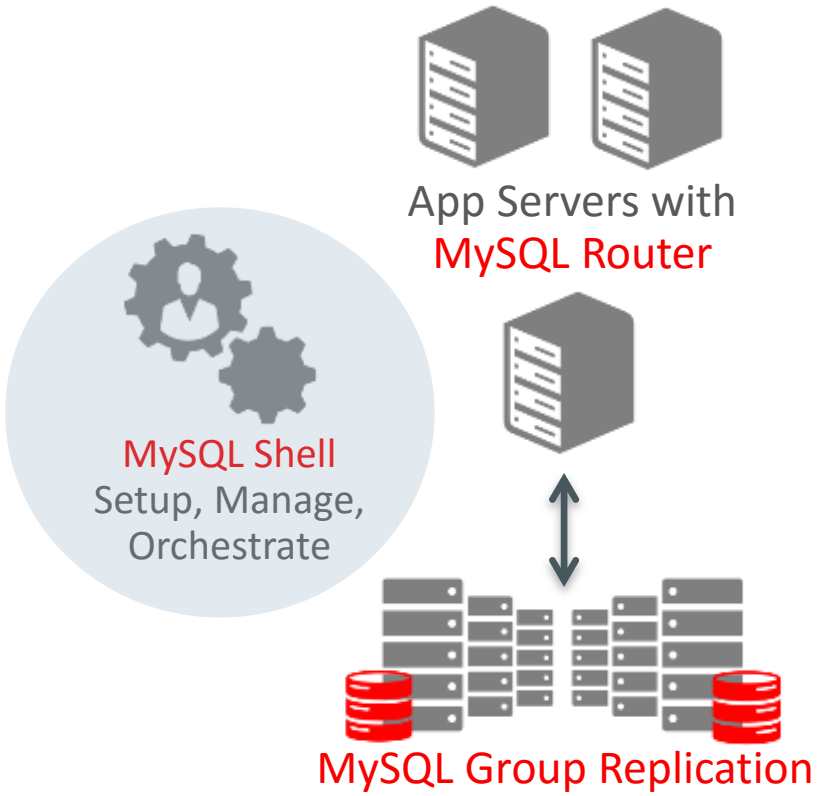
MySQL Shell: 管理用API

データベース管理者向けインターフェース

- `mysql-js> dba.help()`
- グローバル変数 'dba'がMySQLの管理用APIにアクセスする為に使用可能
- DBA管理オペレーション
  - Manage MySQL InnoDB clusters
    - クラスター作成
    - MySQLインスタンスの構築
    - クラスターの状況を確認可能
    - MySQLインスタンスの開始・停止
    - MySQLインスタンスの検証 …

DMR +labs

MySQL Shell DMR (2016-09)



```
mysql-js> dba.help()
The global variable 'dba' is used to access the MySQL AdminAPI functionality
and perform DBA operations.It is used for managing MySQL InnoDB clusters.

The following properties are currently supported.
 - verbose Enables verbose mode on the Dba operations.
The following functions are currently supported.

 - createCluster              Creates a MySQL InnoDB cluster.
 - deleteLocalInstance        Deletes an existing MySQL Server instance on localhost.
 - deployLocalInstance        Creates a new MySQL Server instance on localhost.
 - dropMetadataSchema         Drops the Metadata Schema.
 - getCluster                 Retrieves a cluster from the Metadata Store.
 - help                       Provides help about this class and it's members
 - killLocalInstance          Kills a running MySQL Server instance on localhost.
 - resetSession               Sets the session object to be used on the Dba operations.
 - startLocalInstance         Starts an existing MySQL Server instance on localhost.
 - stopLocalInstance          Stops a running MySQL Server instance on localhost.
 - validateInstance           Validates an instance for usage in Group Replication.

For more help on a specific function use dba.help('<functionName>')
e.g. dba.help('deployLocalInstance')
```

MySQL Shellの機能拡張

MySQL Shell
Setup, Manage, Orchestrate

DMR
+labs

ノードの追加, Group Replication設定, Router連携機能を実装

MySQL Shell – Deploy MySQL Instances

```
shell> mysqlsh
mysql-js> dba.deployLocalInstance(3306)
mysql-js> dba.deployRemoteInstance('192.168.1.2:3306')
mysql-js> dba.deployRemoteInstance('192.168.1.3:3306')
```

MySQL Shell – Add MySQL Router

```
shell> mysqlrouter --bootstrap localhost:3306
shell> mysqlrouter &
shell> mysqlsh --uri root@localhost:6442
```

MySQL Shell – Create InnoDB Cluster

```
shell> mysqlsh --uri root@localhost:3306
mysql-js> cluster = dba.createCluster('NewAppCluster')
mysql-js> cluster.addInstance('root@192.168.1.2:3306')
mysql-js> cluster.addInstance('root@192.168.1.3:3306')
```

MySQL Shell – Check Status

```
shell> mysqlsh --uri root@localhost:3306
mysql-js> cluster = dba.getCluster()
mysql-js> cluster.status()
```

https://www.youtube.com/watch?v=JWy7ZLXxtZ4

ORACLE

Oracle MySQLクラウドサービスによる TCOの最適化

MySQL Powers the Cloud

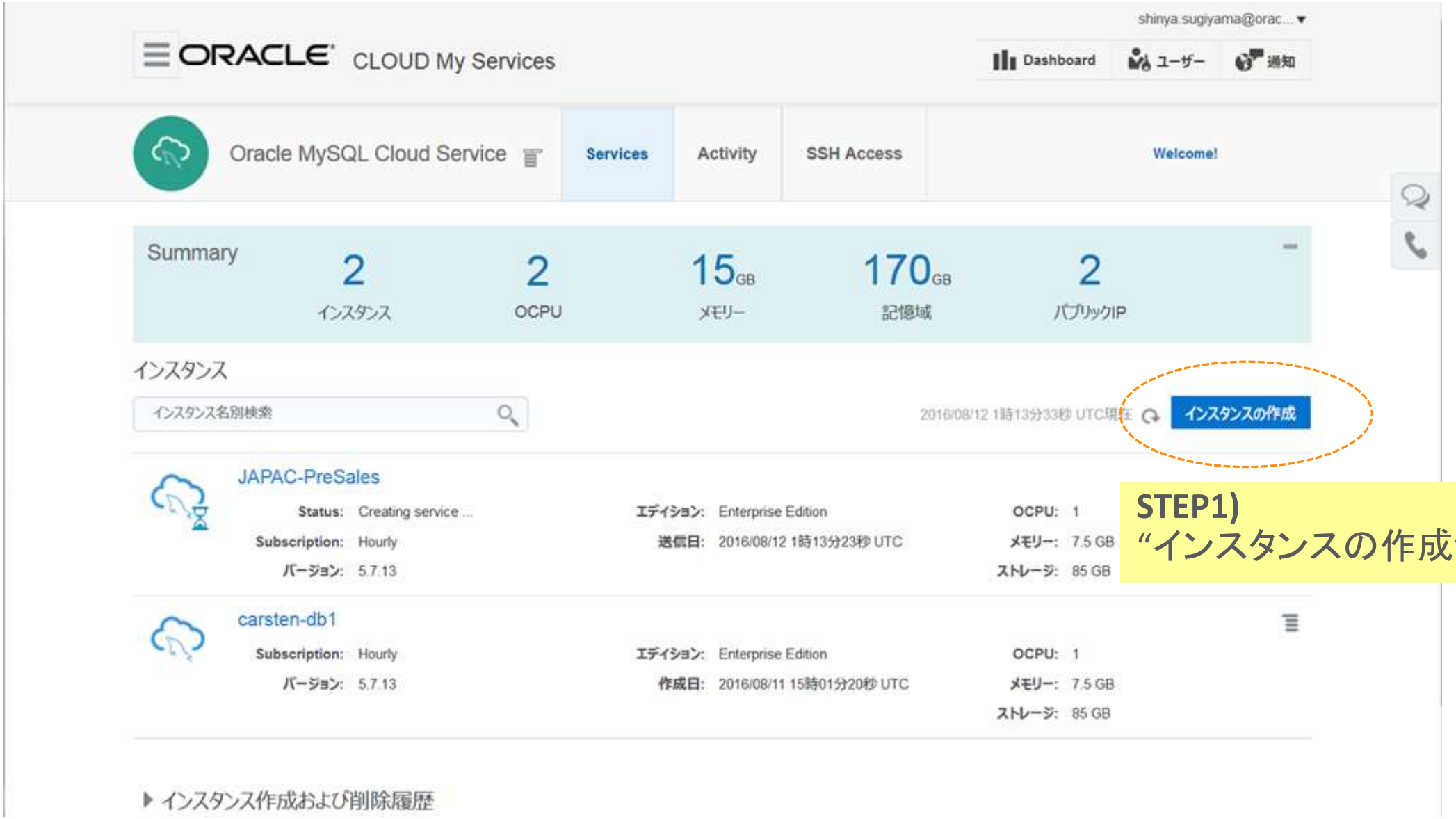
Oracle MySQLクラウドサービス

The World's Most Popular Open Source Database in the Oracle Cloud

Oracle MySQL Cloud Service は、
迅速,安全且つコスト効率良く
MySQLをデプロイする事が可能。

MySQL Cloud Service: 価値提案

- シンプル
  - わずか数回のクリックで,素早くMySQLデータベース・インスタンスが利用可能。
- 自動化
  - データベース管理を自動化するツールで簡単にMySQLを管理する事が可能。
- 統合
  - 迅速な開発と展開の為に、Oracleクラウドサービスとの統合
- エンタープライズ対応
  - パフォーマンス、セキュリティ&アップタイム用のOracleの実証済みのMySQLエンタープライズ・エディションを標準提供。



STEP1)
“インスタンスの作成”をクリック

Provide details for this Oracle MySQL Cloud Service [Enterprise Edition] instance.

Selection Summary

Service Configuration

* Service Name: JAPACSC01
Service Description: New Release Evaluation
* SSH Public Key: ssh-rsa AAAAB3NzaC1yc2EAA 編集
* Compute Shape: OC3 - 1.0 OCPU, 7.5GB RAM

Backup and Recovery Configuration

Backup Destination: Both Cloud and Disk Storage
* Cloud Storage Container: Storage-mysqlsc/JAPACSC01
* Cloud Storage User Name: shinya.sugiyama@oracle.com
* Cloud Storage Password: ••••••••••
Create Cloud Storage Container

Configuration

* Usable Database Storage (GB): 25
* Administration User: root
* Administration Password: ••••••
* Confirm Administration Password: ••••••
* Database Schema Name: JAPAC
Configure Enterprise Monitor: Yes
* Manager User: mem_admin
* Manager Password: •••••••••
* Confirm Manager Password: •••••••••
* Agent User: mem_agent
* Agent Password: •••••••••
* Confirm Agent Password: •••••••••

STEP2)
ホスト名を入力しカタログからサーバータイプを選択し作成。必要に応じてObject Storage, MySQL Enterprise Monitorの設定を入力し完了。

作成時間: 約10分

Restart Service Completed
Service Name: JAPAC-PreSales
Operation: Restart Service
Status: Succeeded
Start Time: 2016/08/12 7時46分39秒 UTC
End Time: 2016/08/12 7時52分37秒 UTC

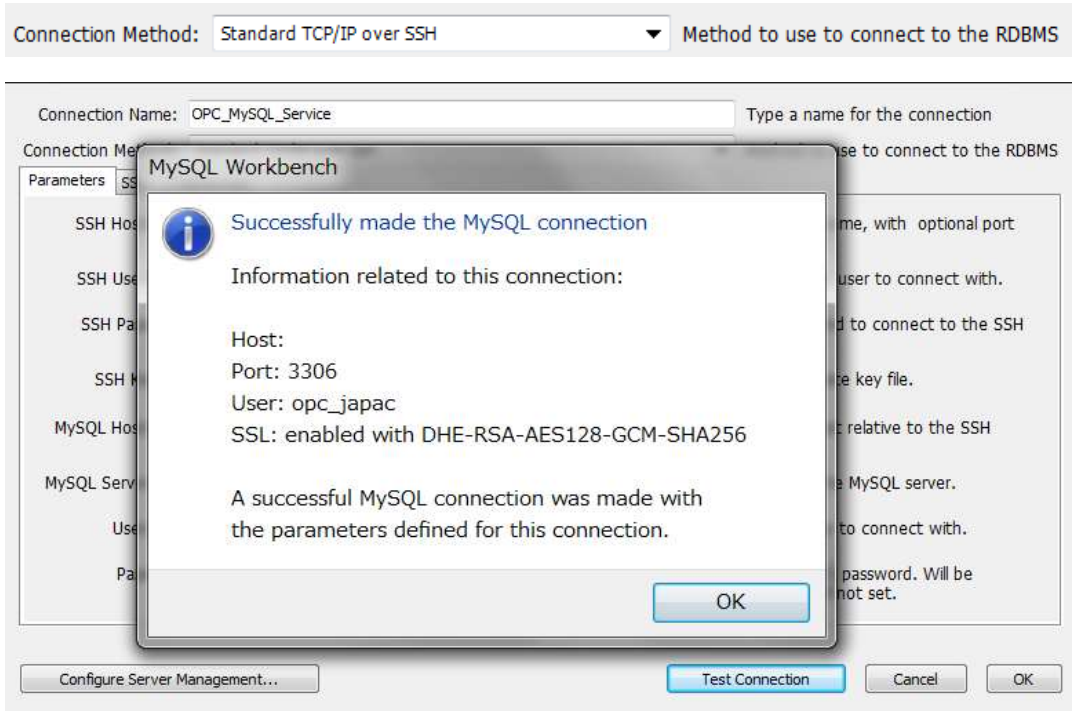
Create Service Completed
Service Name: JAPAC-PreSales
Operation: Create Service
Status: Succeeded
Start Time: 2016/08/12 1時13分23秒 UTC
End Time: 2016/08/12 1時21分01秒 UTC

STEP3)
Public IPが設定されているので、アサインされたIPに対して鍵認証でログインする事が可能。

SSHを利用した接続

```
**********************************************************************************************
*                                    Welcome to                                              *
*                                  MySQL Cloud Service                                       *
*                                       by                                                   *
*                                     Oracle                                                 *
*       If you are an unauthorised user please disconnect IMMEDIATELY                        *
**********************************************************************************************
********************************** MySQL Information **********************************
* Status:  RUNNING                                                                           *
* Version:   5.7.13                                                                          *
**********************************************************************************************
************************** Storage Volume Information **************************
* Volume      Used             Use%          Available   Size     Mounted on    *
* MySQLlog    6.3G -------- 34%                  13G    20G   /u01/translog *
* bin         2.6G ------- 28%                 6.7G   9.8G   /u01/bin      *
* data        151M -- 1%                       24G    25G   /u01/data     *
**********************************************************************************************
[opc@japac-presales-mysql-1 ~]$
```

Workbench経由でのSSH接続



```
mysql> select PLUGIN_NAME,PLUGIN_STATUS,PLUGIN_TYPE,LOAD_OPTION from PLUGINS
    -> where PLUGIN_TYPE <> 'INFORMATION SCHEMA';
+-----------------------+---------------+--------------------+----------------------+
| PLUGIN_NAME           | PLUGIN_STATUS | PLUGIN_TYPE        | LOAD_OPTION          |
+-----------------------+---------------+--------------------+----------------------+
| binlog                | ACTIVE        | STORAGE ENGINE     | FORCE                |
| mysql_native_password | ACTIVE        | AUTHENTICATION     | FORCE                |
| sha256_password       | ACTIVE        | AUTHENTICATION     | F                    |
| InnoDB                | ACTIVE        | STORAGE ENGINE     | F                    |
| PERFORMANCE_SCHEMA    | ACTIVE        | STORAGE ENGINE     | F                    |
| MRG_MYISAM            | ACTIVE        | STORAGE ENGINE     | F                    |
| MyISAM                | ACTIVE        | STORAGE ENGINE     | FORCE                |
| MEMORY                | ACTIVE        | STORAGE ENGINE     | FORCE                |
| CSV                   | ACTIVE        | STORAGE ENGINE     | FORCE                |
| BLACKHOLE             | DISABLED      | STORAGE ENGINE     | OFF                  |
| partition             | ACTIVE        | STORAGE ENGINE     | ON                   |
| FEDERATED             | DISABLED      | STORAGE ENGINE     | OFF                  |
| ARCHIVE               | DISABLED      | STORAGE ENGINE     | OFF                  |
| ngram                 | ACTIVE        | FTPARSER           | ON                   |
| audit_log             | ACTIVE        | AUDIT              | FORCE_PLUS_PERMANENT |
| thread_pool           | ACTIVE        | DAEMON             | ON                   |
| authentication_pam    | ACTIVE        | AUTHENTICATION     | ON                   |
| auth_socket           | ACTIVE        | AUTHENTICATION     | ON                   |
| validate_password     | ACTIVE        | VALIDATE PASSWORD  | ON                   |
+-----------------------+---------------+--------------------+----------------------+
```

MySQL Enterprise版のバイナリーがインストール済みの為、Enterprise版の機能が利用する事が可能。

Backup | Patching

2016/08/30 6時57分34秒 UTC現在

Summary

0 MB
Storage Cloud Volume Used

0 MB
Backup Volume Used

0 %
Backup Volume Percent Used

Daily at 16時15分00秒 UTC
Incremental Backups

Every Tuesday at 16時15分00秒 UTC
Full Backups

Not Available
Last Successful Backup

MySQL Enterprise Backupも実装されていて、
DashboardからバックアップジョブⅠ設定、
Point In Timeリカバリー含めて管理可能。

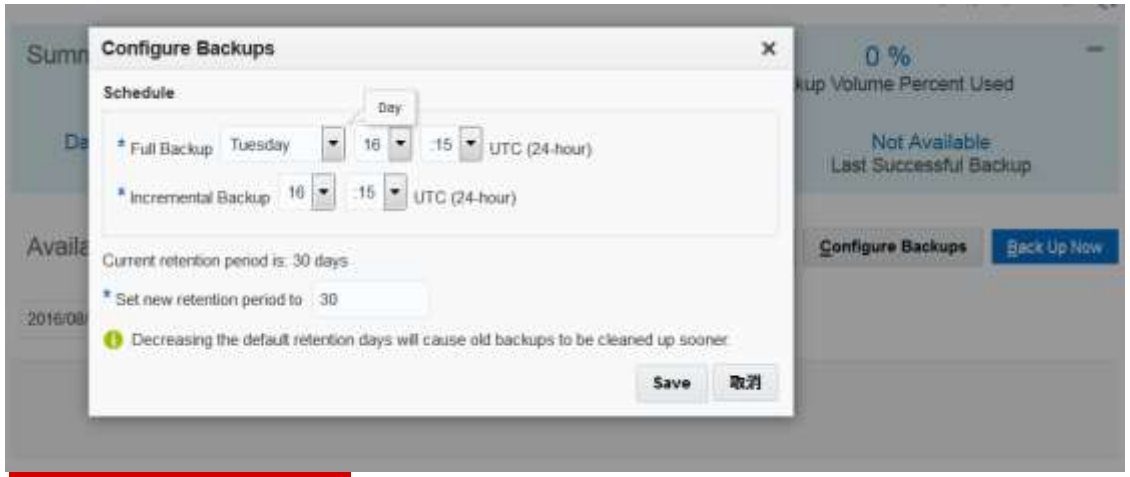
Available Backups

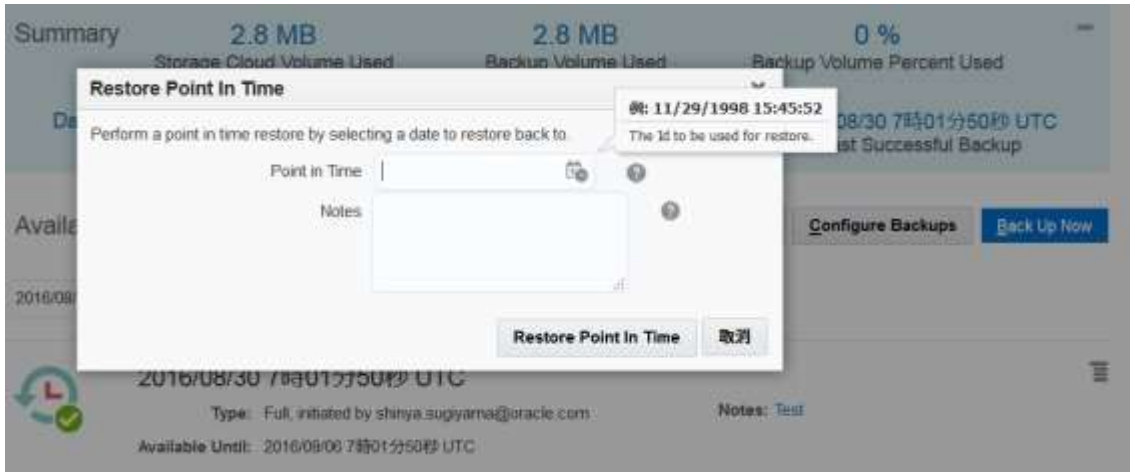
Restore Point In Time | Configure Backups | Back Up Now

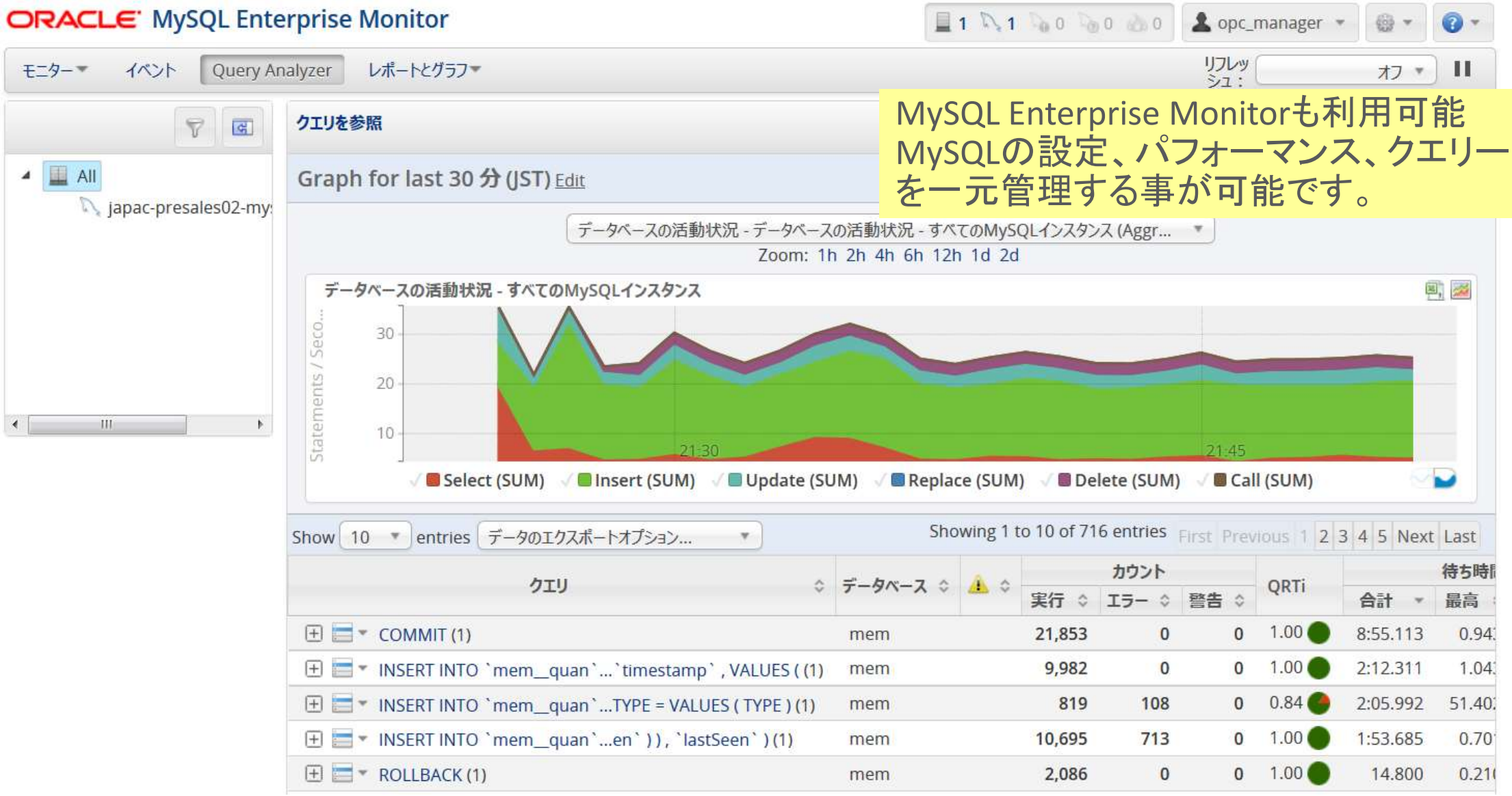
2016/08/23 to Enter end date

No backups available.

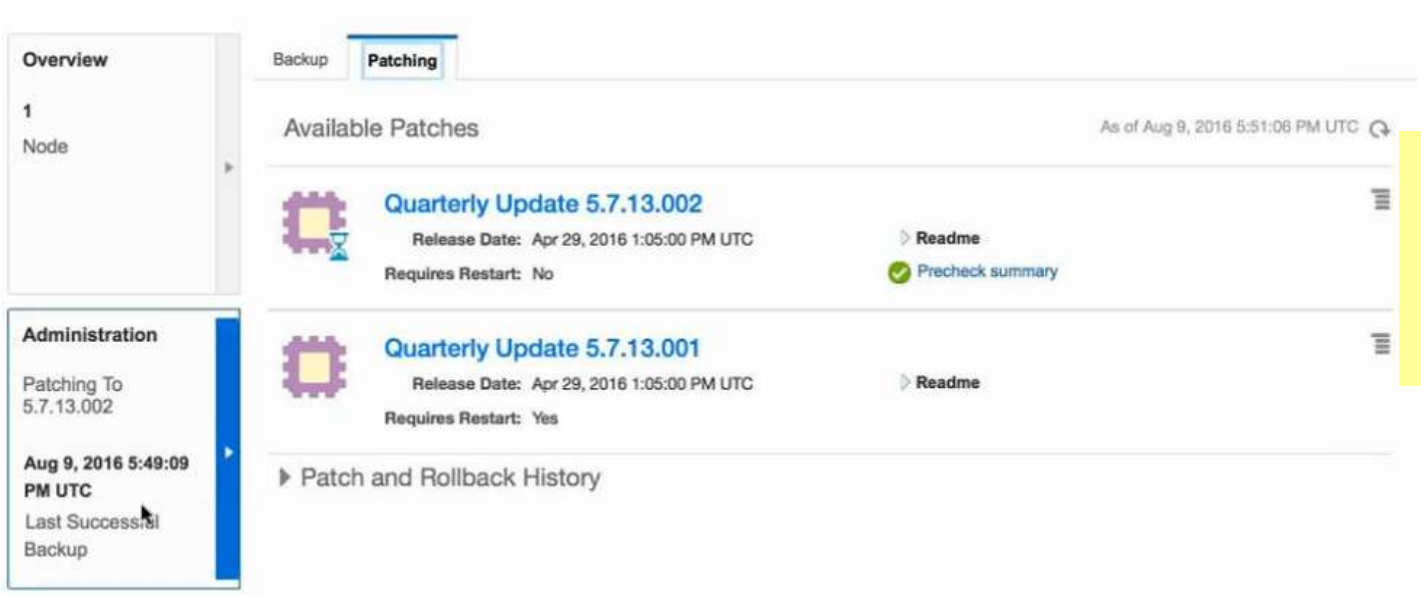
Restore History

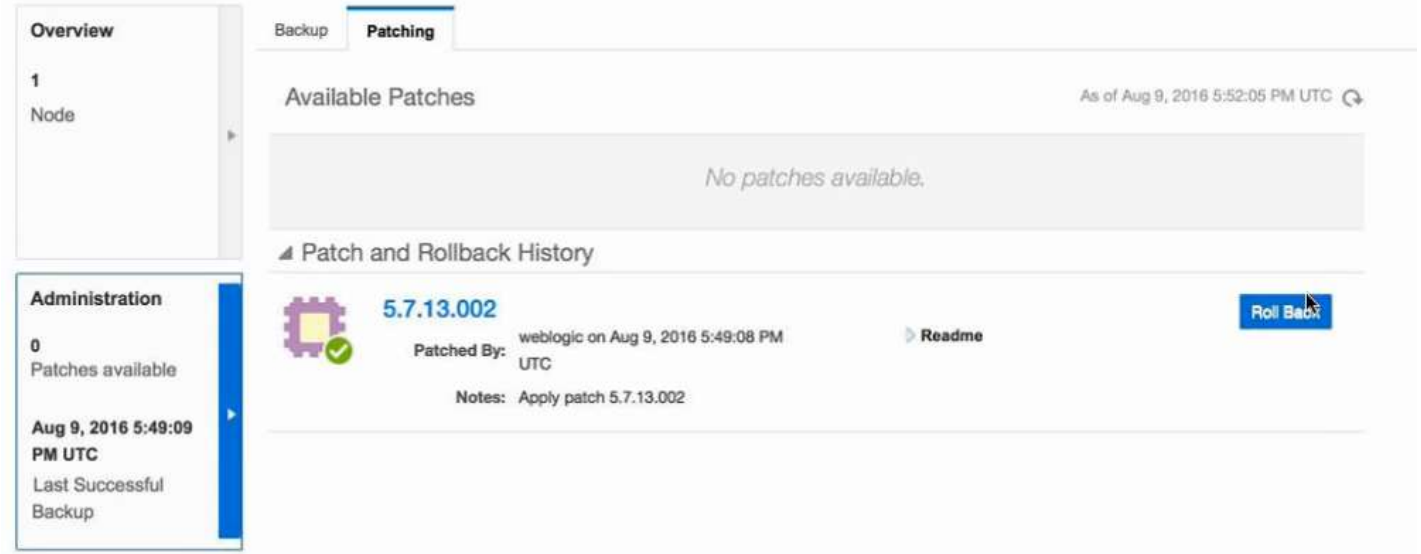






MySQL Enterprise Monitorも利用可能
MySQLの設定、パフォーマンス、クエリー等を一元管理する事が可能です。





Dashboardから、MySQLのパッチ適用、適用前の事前検証と適用後のロールバックを実施する事が可能

※適用前にMEBでバックアップが自動取得されます。

最高レベルのセキュリティ

スケーラビリティと可用性

MySQLエキスパート
テクニカルサポート

Oracleクラウド環境へ統合

ハイブリッドにデプロイ可能
クラウド& オンプレミス

TCOの削減

MySQL Cloud Service: ビジネス上のメリット

- **ビジネスの俊敏性を向上**:
  イノベーションにリソースを集中し、迅速に最新のアプリケーションを提供。
- **確実なセキュリティ,パフォーマンス,稼働時間**:
  ソースレベルから、最も包括的なMySQL Cloud プラットホームを利用する事が可能。
- **TCO（総所有コスト）を削減:**
  稼働時間を向上させながら、インフラストラクチャ及びデータベース管理操作コストを節約可能。

REST APIを利用して、自動化する事も可能です。
http://docs.oracle.com/cloud/latest/stcomputecs/STCSA/QuickStart.html
※ MySQL ServiceのRESTマニュアルは現在、準備中です。

MySQL Cloud Service Pricing

課金方法は2つオプション

- Metered　(実際に利用した分のみのコスト負担)
- No-Metered (月単位での固定課金)

Japanese Yen

Buy Now

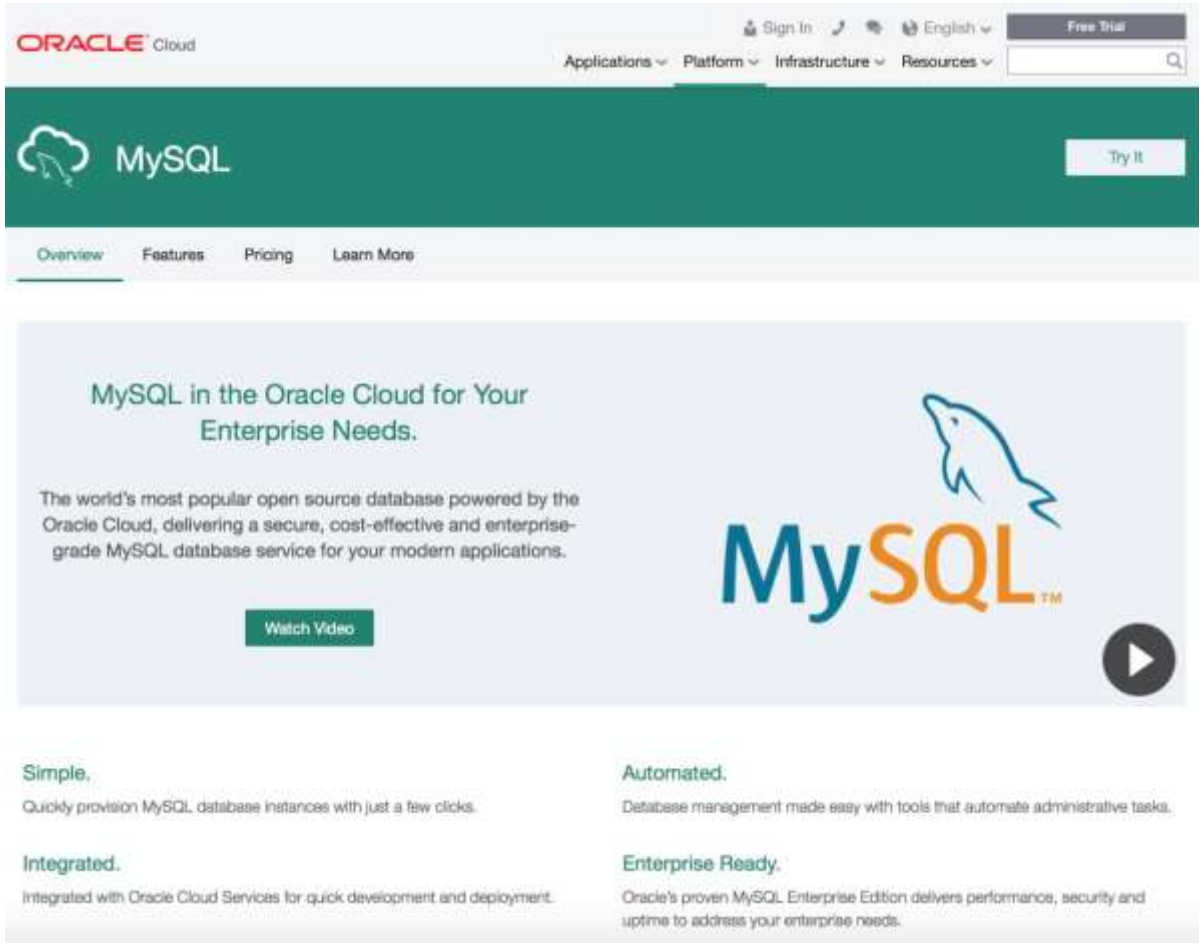
Product	Price	Metric
MySQL Cloud Service - Metered	￥28	OCPU / Hour
MySQL Cloud Service - Non-metered	￥15,600	OCPU / Month

1 OCPU = 2 vCPU

```
[root@japacsc01-mysql-1 opc]# cat /proc/cpuinfo | grep processor
processor       : 0
processor       : 1
[root@japacsc01-mysql-1 opc]#
```

詳細: https://cloud.oracle.com/en_US/mysql/pricing

詳細情報 @ cloud.oracle.com/mysql



Sign up today for a free trial @

https://cloud.oracle.com/mysql


ORACLE®