High Availability Using MySQL Group Replication

Luís Soares (luis.soares@oracle.com)
Principal Software Engineer

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

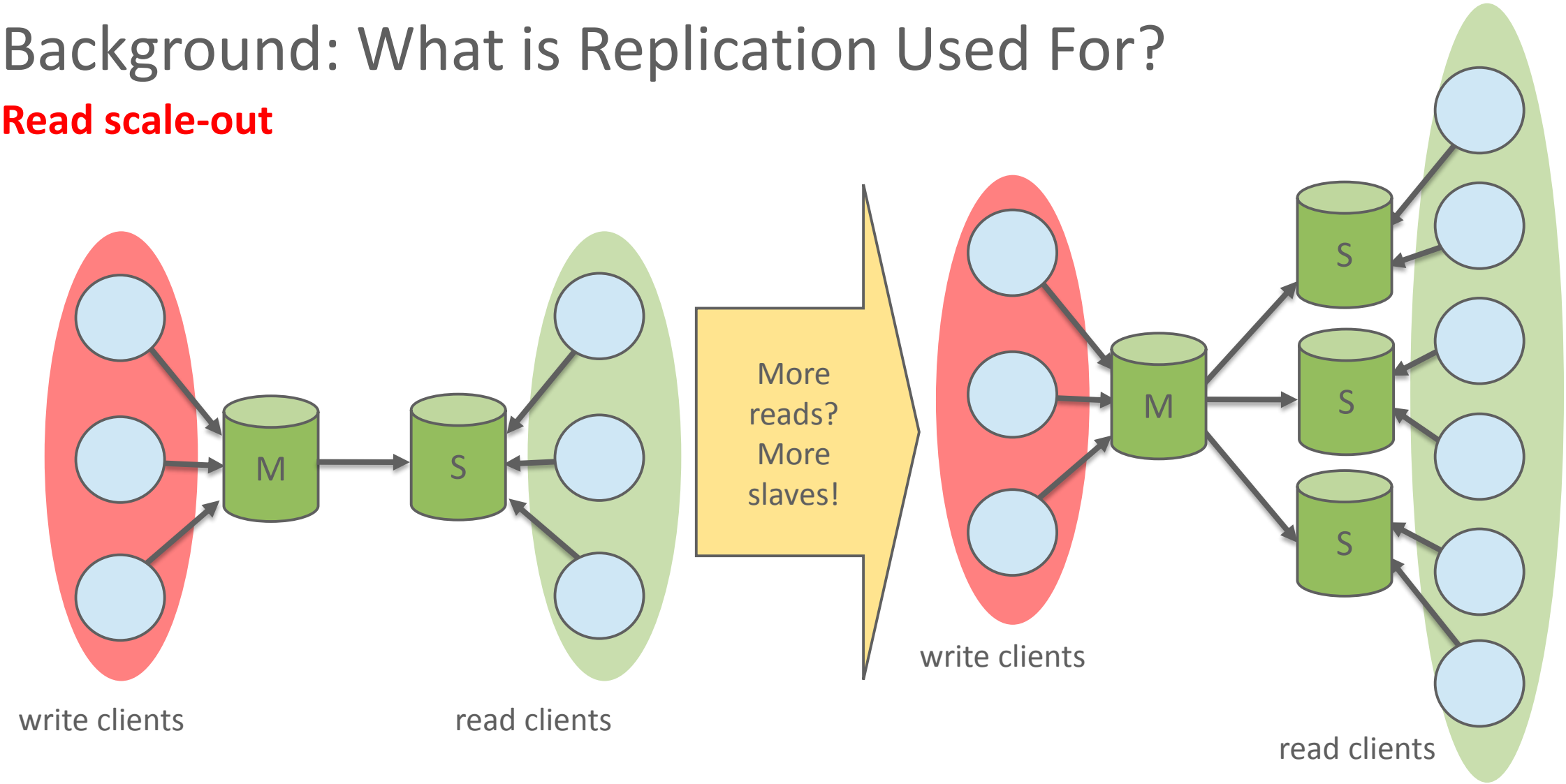
Program Agenda

1. Background
2. MySQL Group Replication
3. Architecture
4. Big Picture
5. Conclusion

1 Background

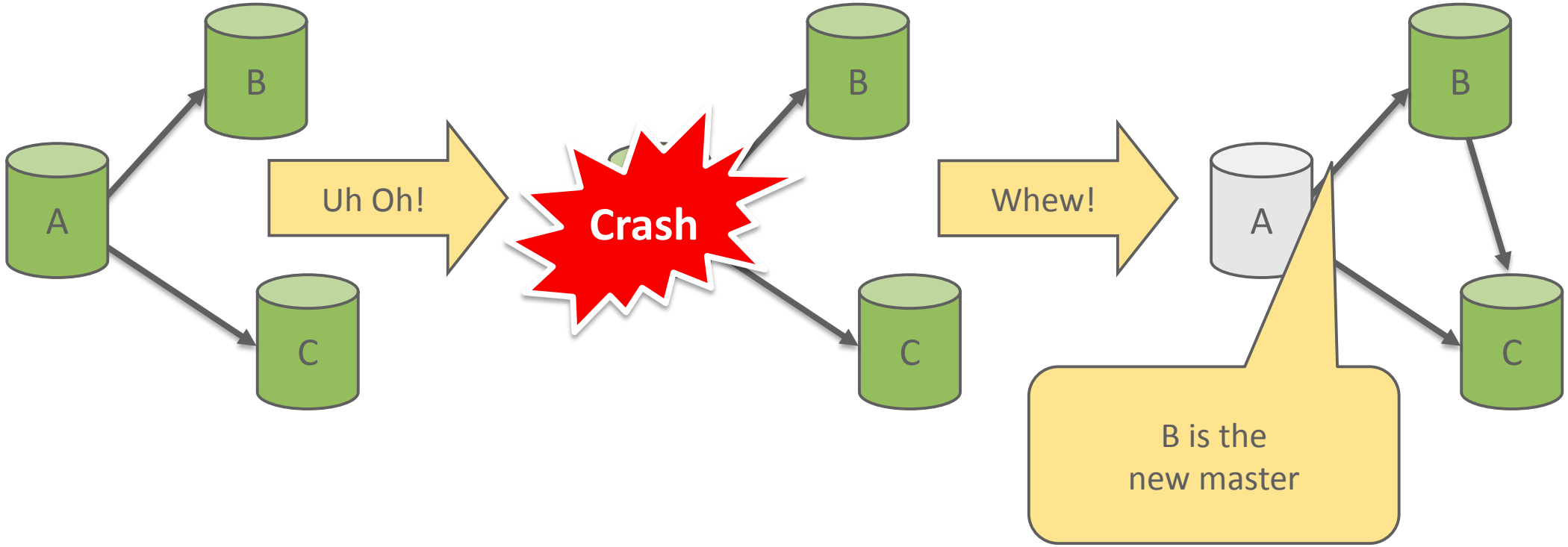
Background: What is Replication Used For?

Read scale-out



Background: What is Replication Used For?

Redundancy as a major building block for high availability: If master crashes, promote slave to master



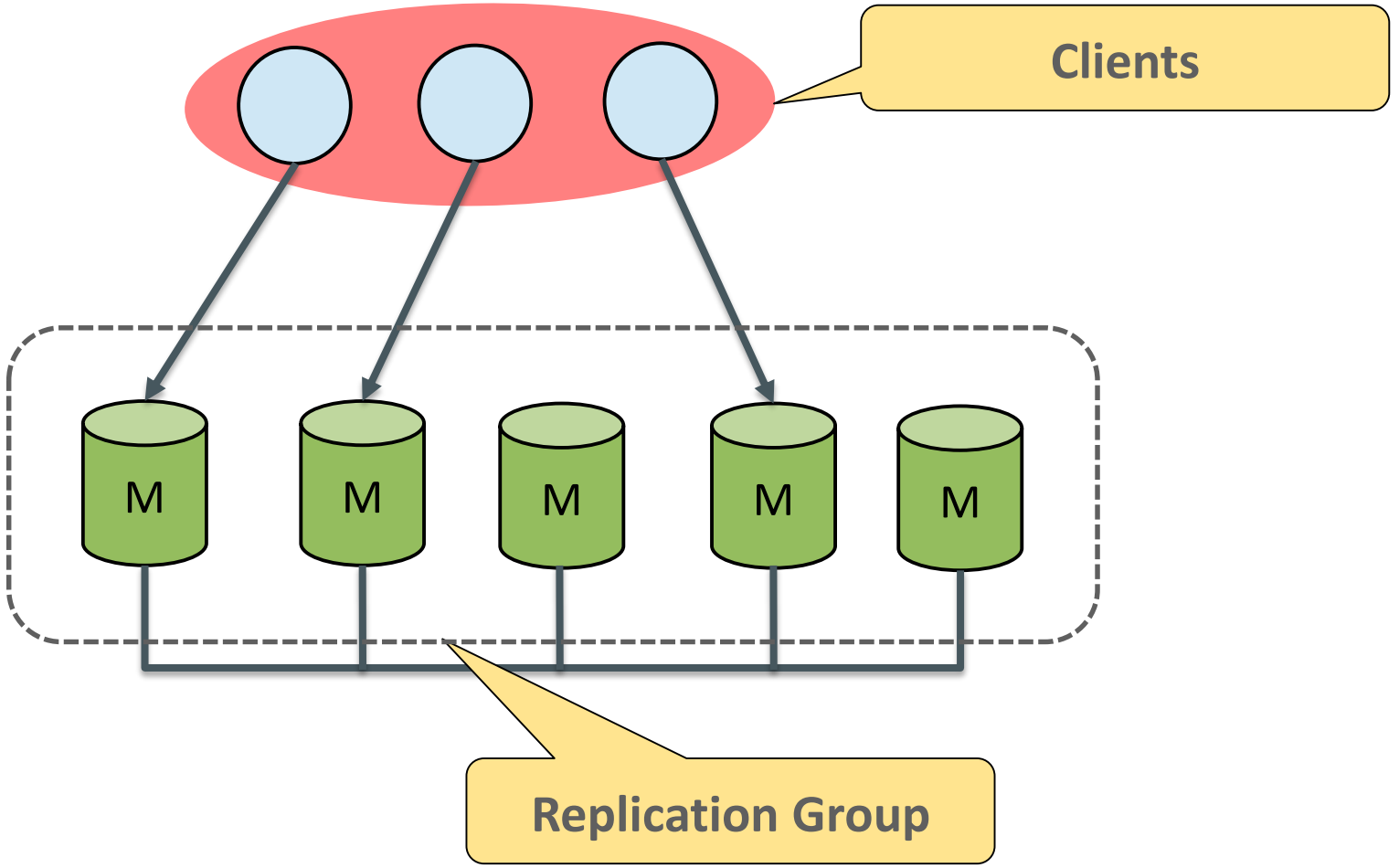
MySQL Group Replication

- **What is MySQL Group Replication?**

  "**Update everywhere** replication plugin for MySQL with built-in **automatic distributed recovery**, **conflict handling**, **group membership** and **distributed agreement**."

- **What does the MySQL Group Replication plugin do for the user?**
  - Removes the need for handling server fail-over.
  - Provides fault tolerance.
  - Enables update everywhere setups.
  - Automates group reconfiguration (handling of crashes, failures, re-connects).
  - Provides a highly available replicated database.
  - Automatic distributed coordination (protects against split-brain and message loss).
  - Less admin overhead, means more fun time!

MySQL Group Replication



Some Theory Behind It...

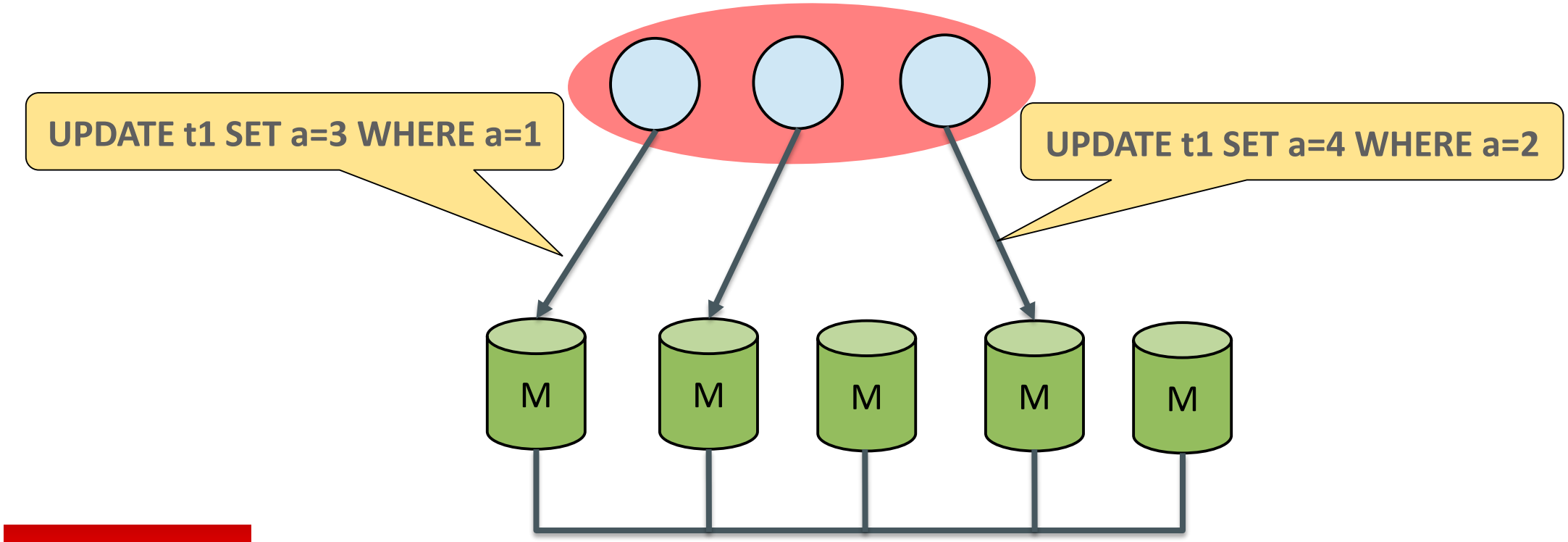
- Implementation based in Replicated Database State Machines
  - Group Communication Primitives resemble properties of Databases.
- Deferred update replication: **propagate atomically**, **check conflicts**, **eventually apply**
  - Distributed state machine requires agreed delivery – implies total order;
  - Deterministic certification requires total order delivery.
- Membership Service
  - Which servers are participating in the replication group at a given moment in time? (associated with a logical timestamp [- view identifier]).

2 MySQL Group Replication

2.1 Multi-Master

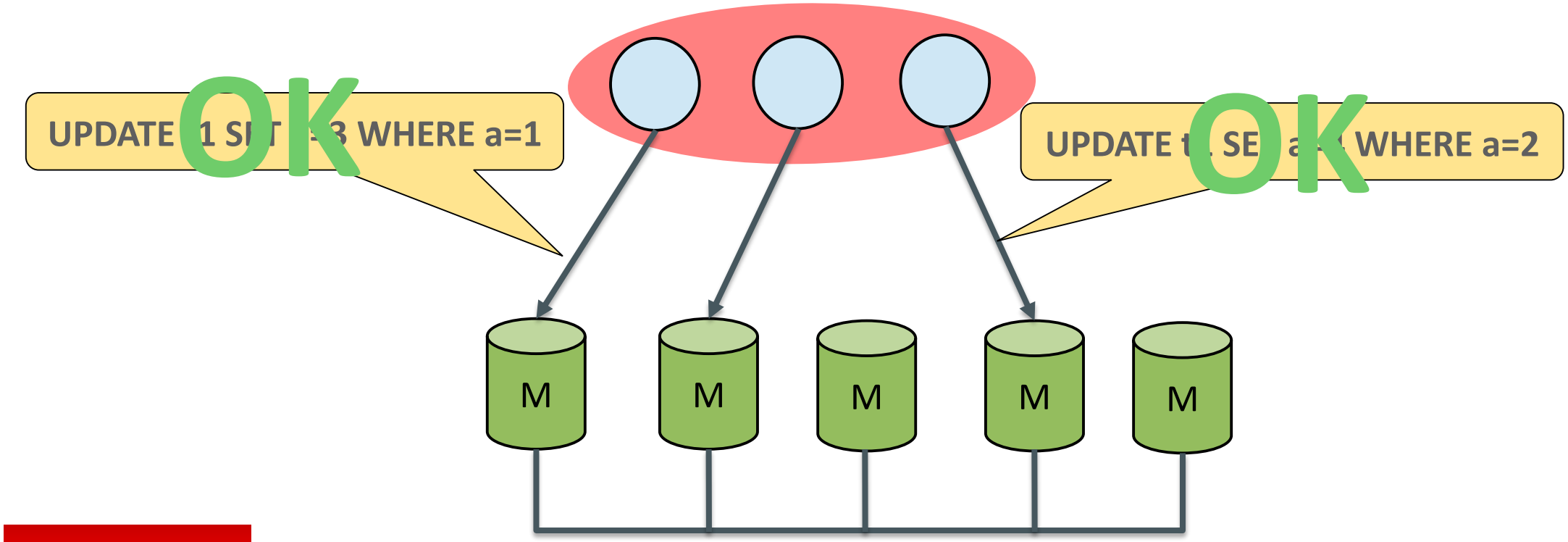
Multi-Master update everywhere!

- Any two transactions on different servers can write to the same tuple.
- Conflicts will be detected and dealt with.
  - First committer wins rule.



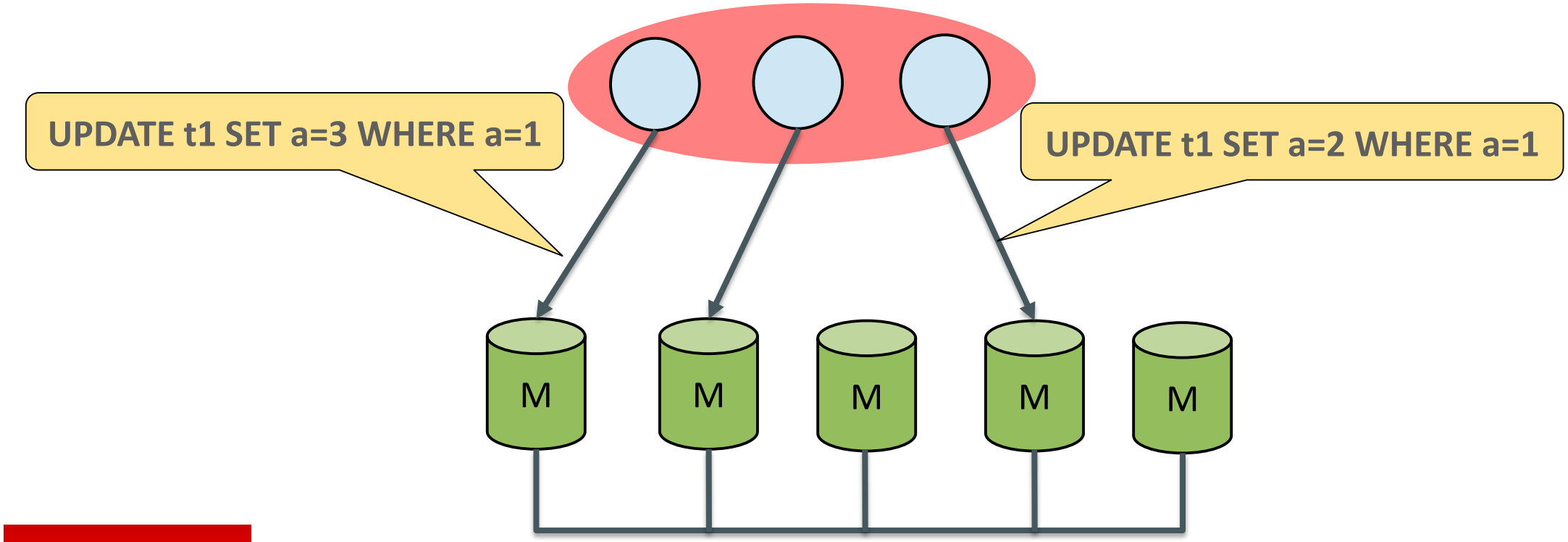
Multi-Master update everywhere!

- Any two transactions on different servers can write to the same tuple.
- Conflicts will be detected and dealt with.
  - First committer wins rule.



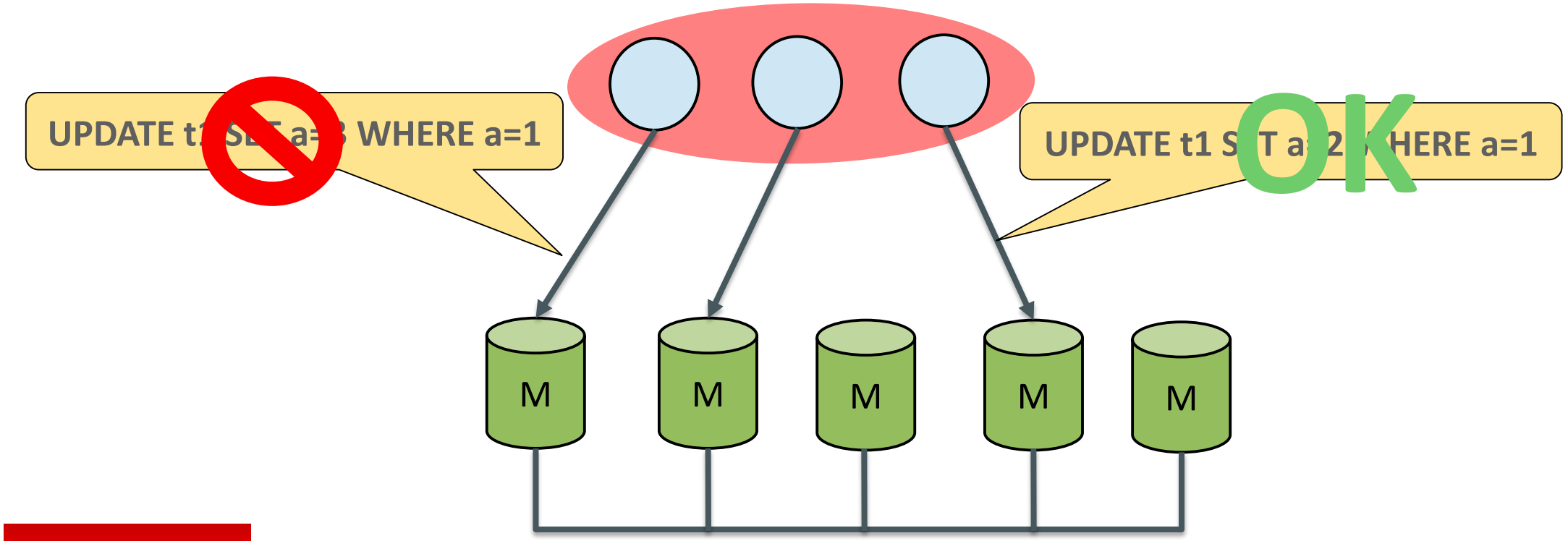
Multi-Master update everywhere!

- Any two transactions on different servers can write to the same tuple.
- Conflicts will be detected and dealt with.
  - First committer wins rule.



Multi-Master update everywhere!

- Any two transactions on different servers can write to the same tuple.
- Conflicts will be detected and dealt with.
  - First committer wins rule.



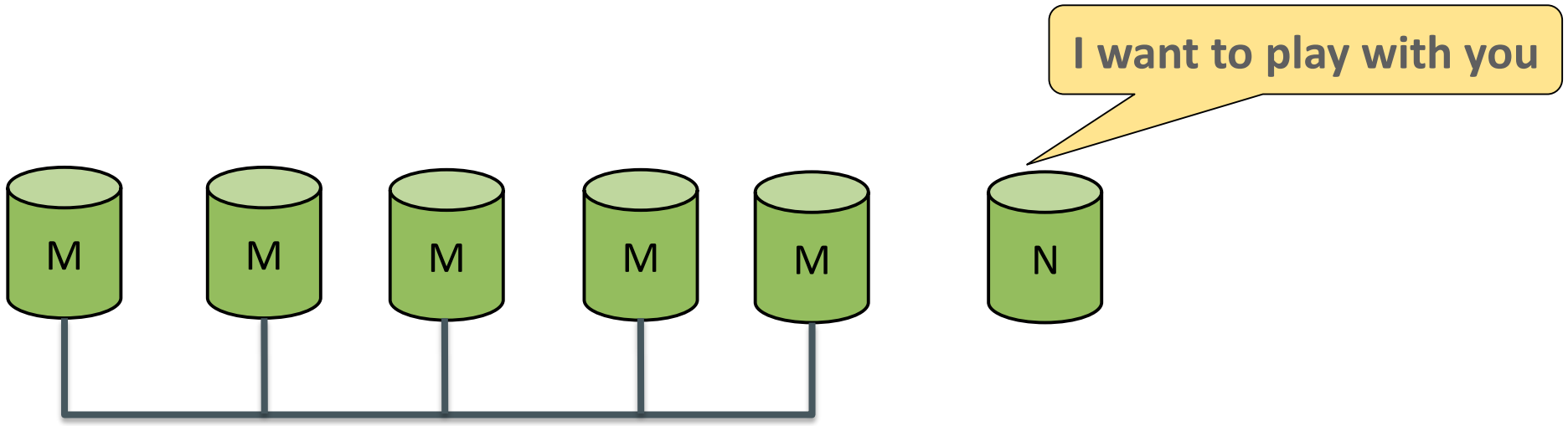
2 MySQL Group Replication

2.1 Multi-Master

2.2 Automatic distributed server recovery

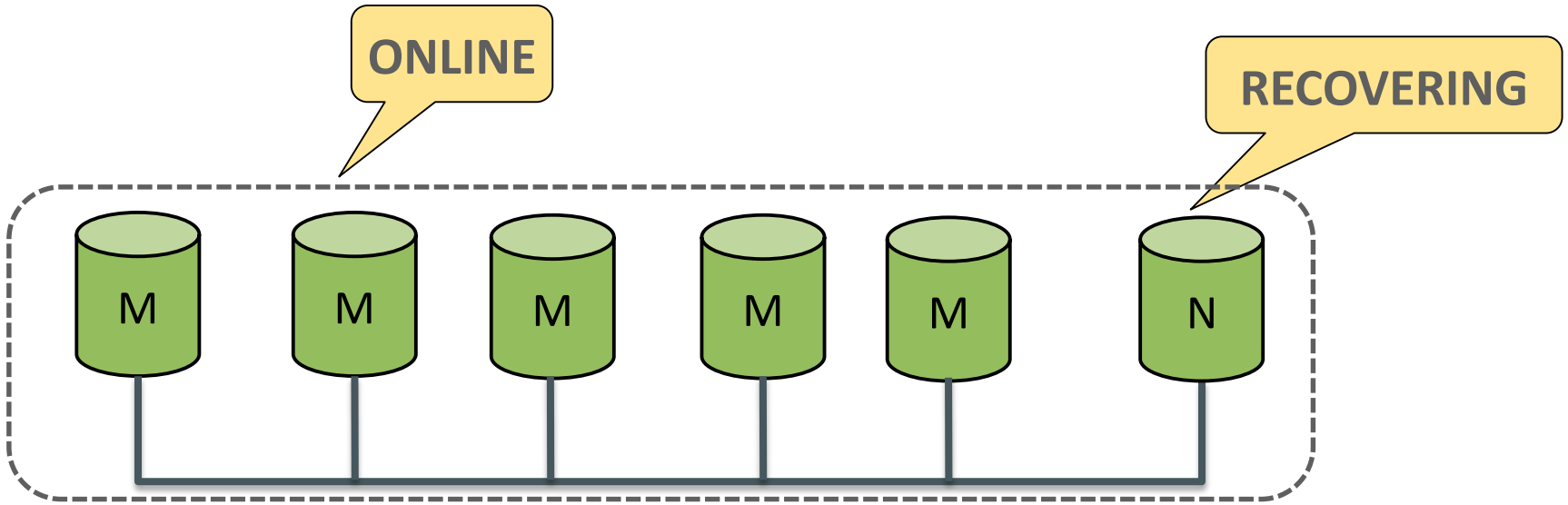
Automatic distributed server recovery!

- Server that joins the group will automatically synchronize with the others.



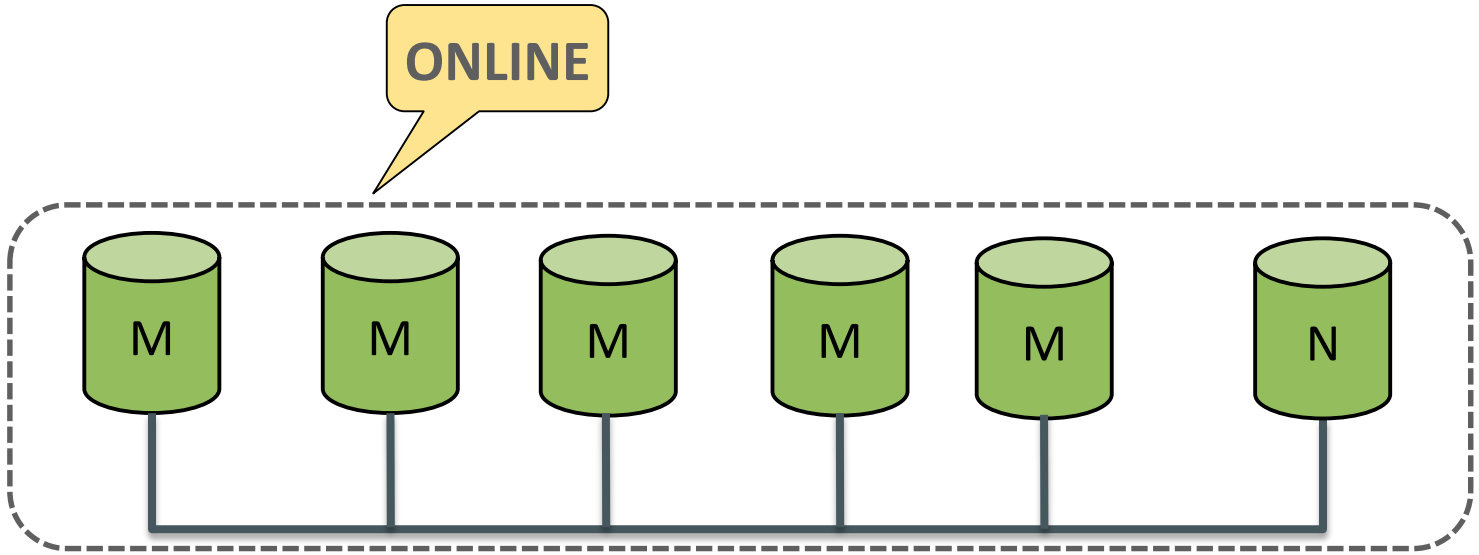
Automatic distributed server recovery!

- Server that joins the group will automatically synchronize with the others.



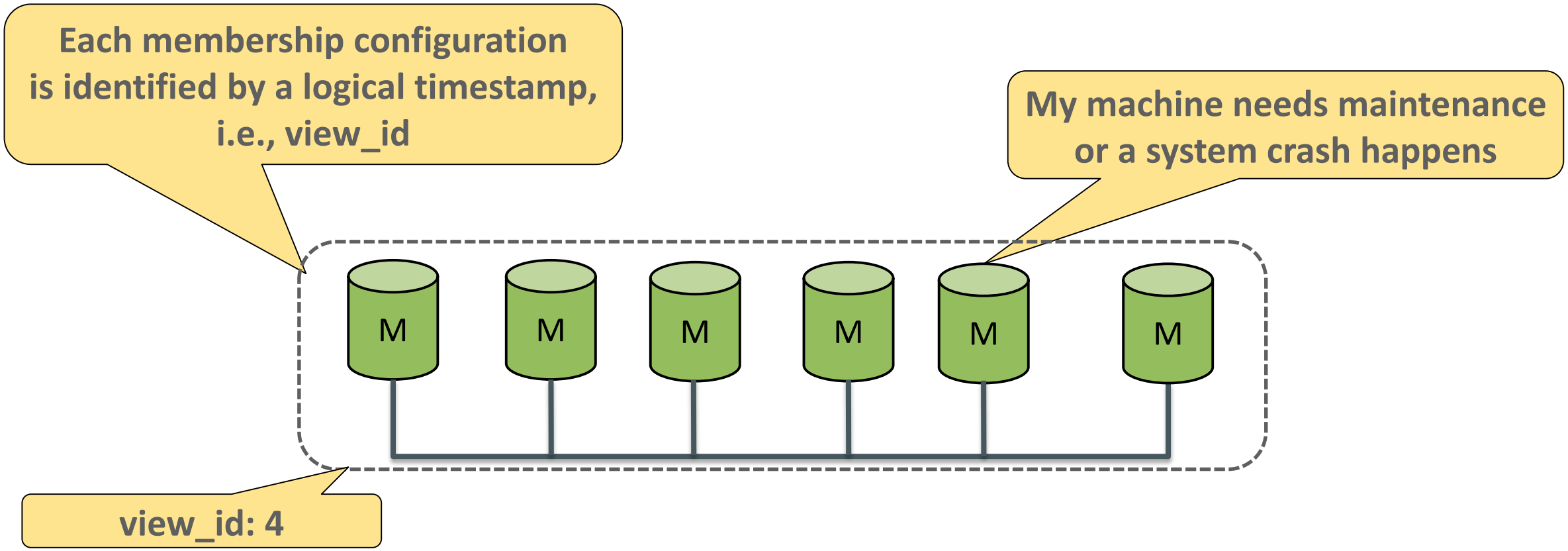
Automatic distributed server recovery!

- Server that joins the group will automatically synchronize with the others.



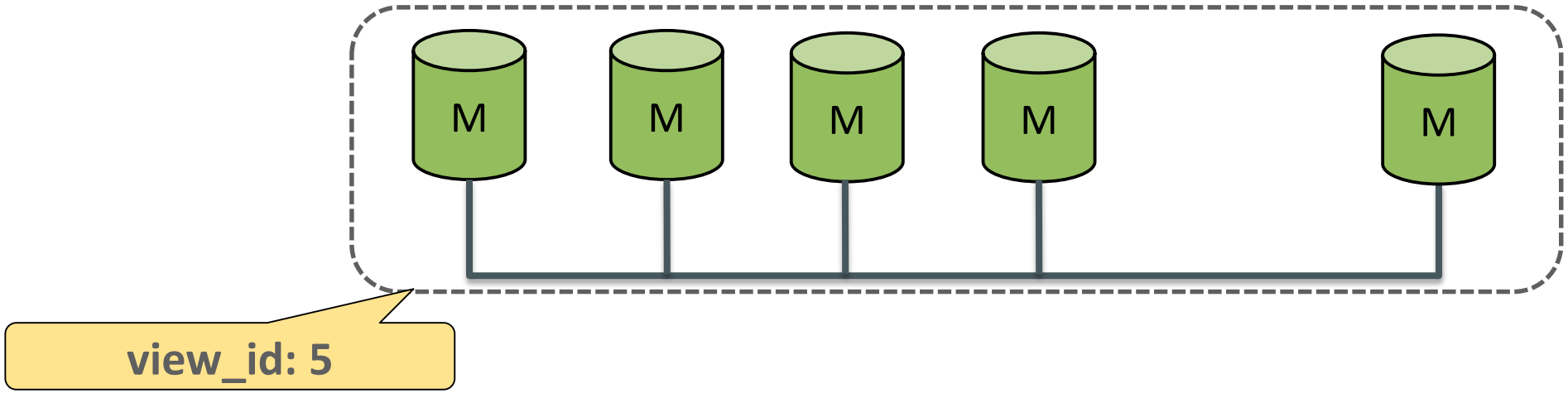
Automatic distributed server recovery!

- If a server leaves the group, the others will automatically be informed.



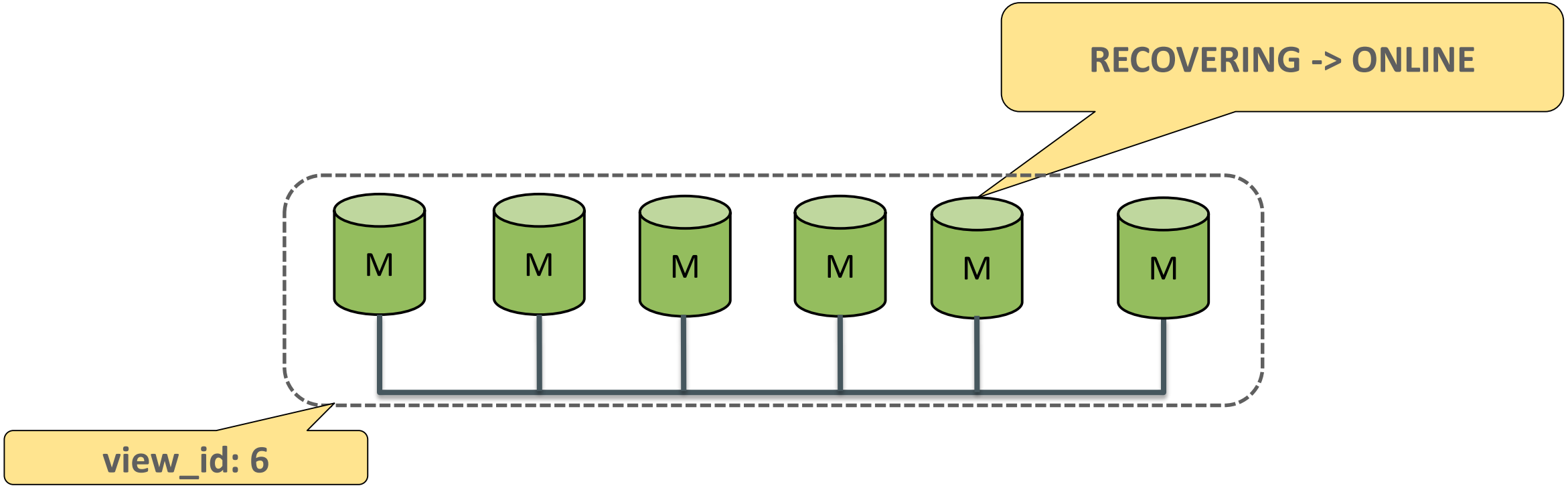
Automatic distributed server recovery!

- If a server leaves the group, the others will automatically be informed.



Automatic distributed server recovery!

- Server that (re)joins the group will automatically synchronize with the others.



2 MySQL Group Replication

2.1 Multi-Master

2.2 Automatic distributed server recovery

2.3 MySQL/InnoDB look & feel

MySQL/InnoDB look & feel!

- Load the plugin and start replicating.
- Monitor group replication stats though Performance Schema tables.

```
mysql> SET GLOBAL group_replication_group_name= "9eb07c6d-5e24-11e5-854b-34028662c0cd";
mysql> START GROUP_REPLICATION;

mysql> SELECT * FROM performance_schema.replication_group_members\G
*************************** 1. row ***************************
 CHANNEL_NAME: group_replication_applier
 MEMBER_ID: 597dbb72-3e2c-11e4-9d9d-ecf4bb227f3b
 MEMBER_HOST: nightfury
 MEMBER_PORT: 13000
 MEMBER_STATE: ONLINE
*************************** 2. row ***************************
 ...
```

MySQL/InnoDB look & feel!

- Load the plugin and start replicating.
- Monitor group replication stats though Performance Schema tables.

```
mysql> SELECT * FROM performance_schema.replication_group_member_stats\G
*************************** 1. row ***************************
 CHANNEL_NAME: group_replication_applier
 VIEW_ID: 1428497631:3
 MEMBER_ID: e38fdea8-dded-11e4-b211-e8b1fc3848de
 COUNT_TRANSACTIONS_IN_QUEUE: 0
 COUNT_TRANSACTIONS_CHECKED: 12
 COUNT_CONFLICTS_DETECTED: 5
 COUNT_TRANSACTIONS_VALIDATING: 6
 TRANSACTIONS_COMMITTED_ALL_MEMBERS: 8a84f397-aaa4-18df-89ab-c70aa9823561:1-7
 LAST_CONFLICT_FREE_TRANSACTION: 8a84f397-aaa4-18df-89ab-c70aa9823561:7
```

MySQL/InnoDB look & feel!

- Load the plugin and start replicating.
- Monitor group replication stats though Performance Schema tables.

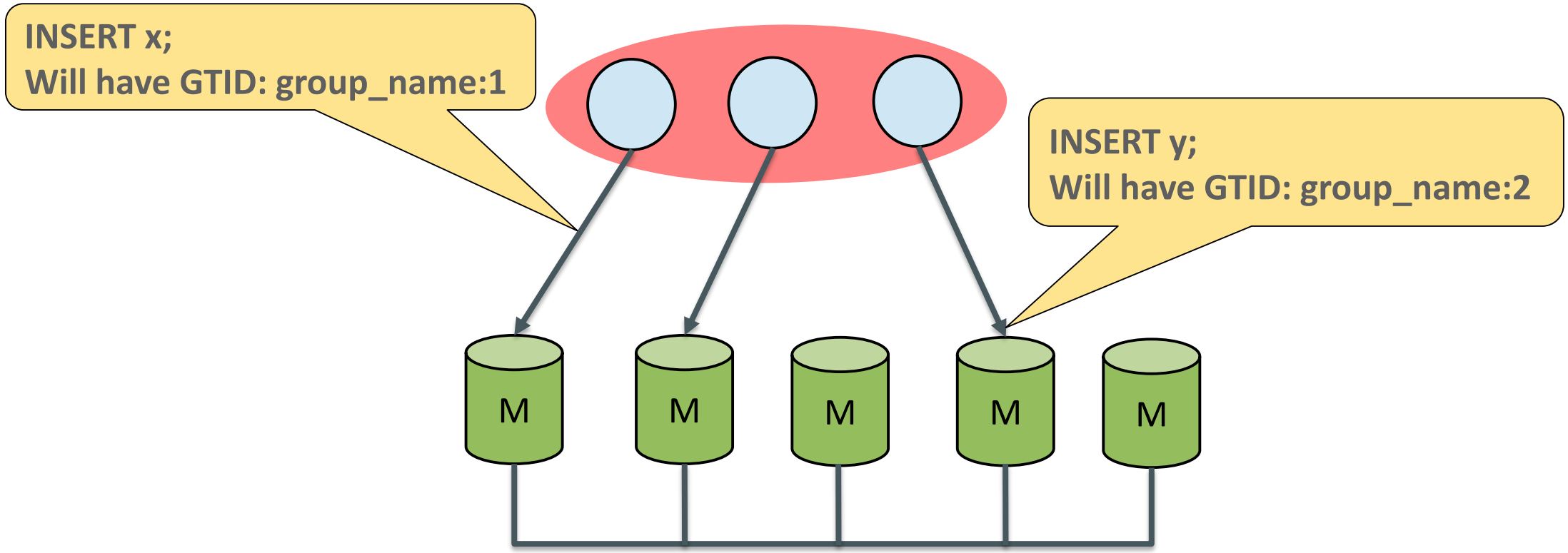
```
mysql> SELECT * FROM performance_schema.replication_connection_status\G
*************************** 1. row ***************************
 CHANNEL_NAME: group_replication_applier
 GROUP_NAME: 8a94f357-aab4-11df-86ab-c80aa9429563
 SOURCE_UUID: 8a94f357-aab4-11df-86ab-c80aa9429563
 THREAD_ID: NULL
 SERVICE_STATE: ON
 ...
```

2 MySQL Group Replication

- 2.1 Multi-Master
- 2.2 Automatic distributed server recovery
- 2.3 MySQL/InnoDB look & feel
- 2.4 Full GTID support

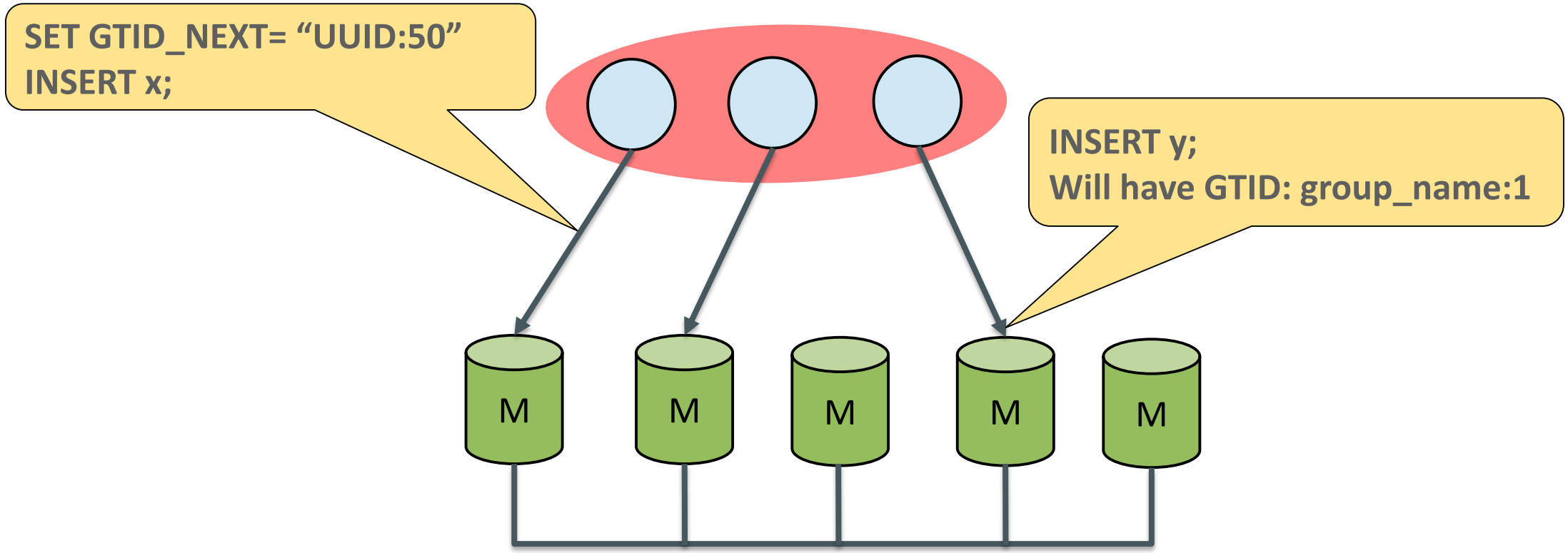
Full GTID support!

- All group members share the same UUID, the group name.



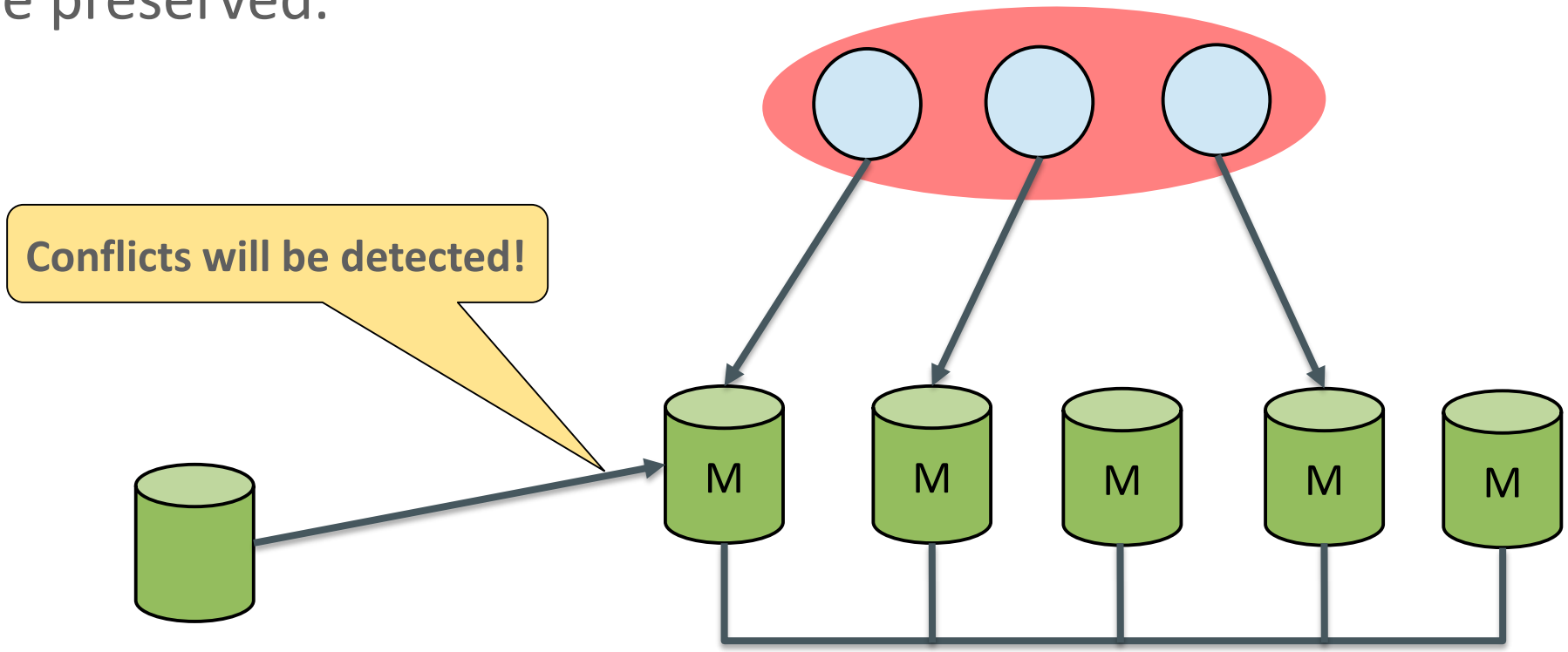
Full GTID support!

- Users can specify the identifier for the transaction.



Full GTID support!

- You can even replicate from a outside server to a group, global identifiers will be preserved.

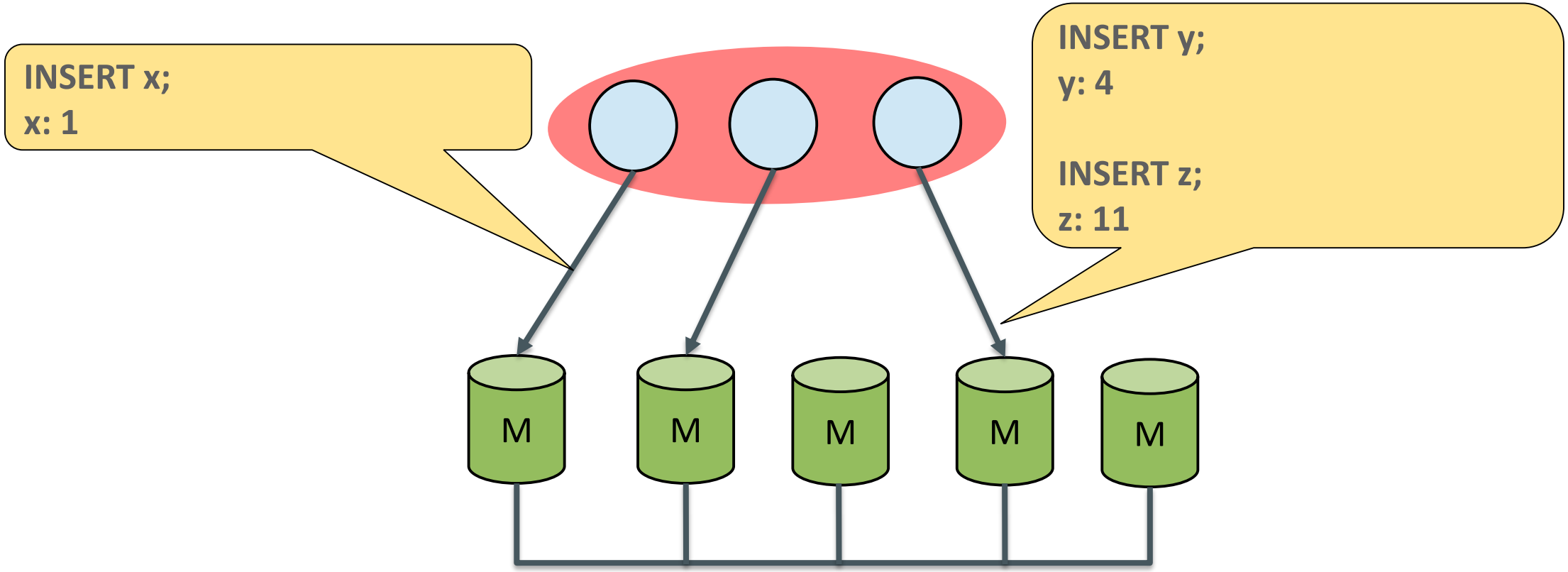


2 MySQL Group Replication

- 2.1 Multi-Master
- 2.2 Automatic distributed server recovery
- 2.3 MySQL/InnoDB look & feel
- 2.4 Full GTID support
- 2.5 Auto-increment configuration/handling

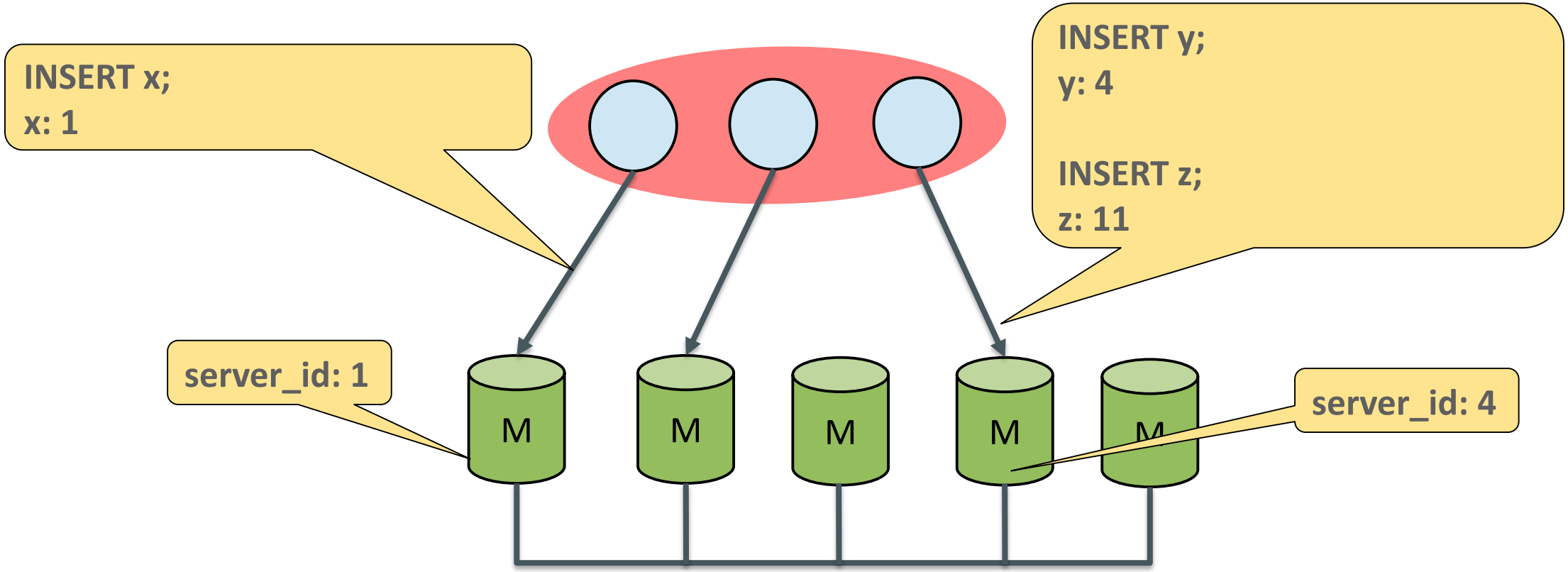
Auto-increment configuration/handling

- Group is configured not to generate the same auto-increment value on all members.



Auto-increment configuration/handling

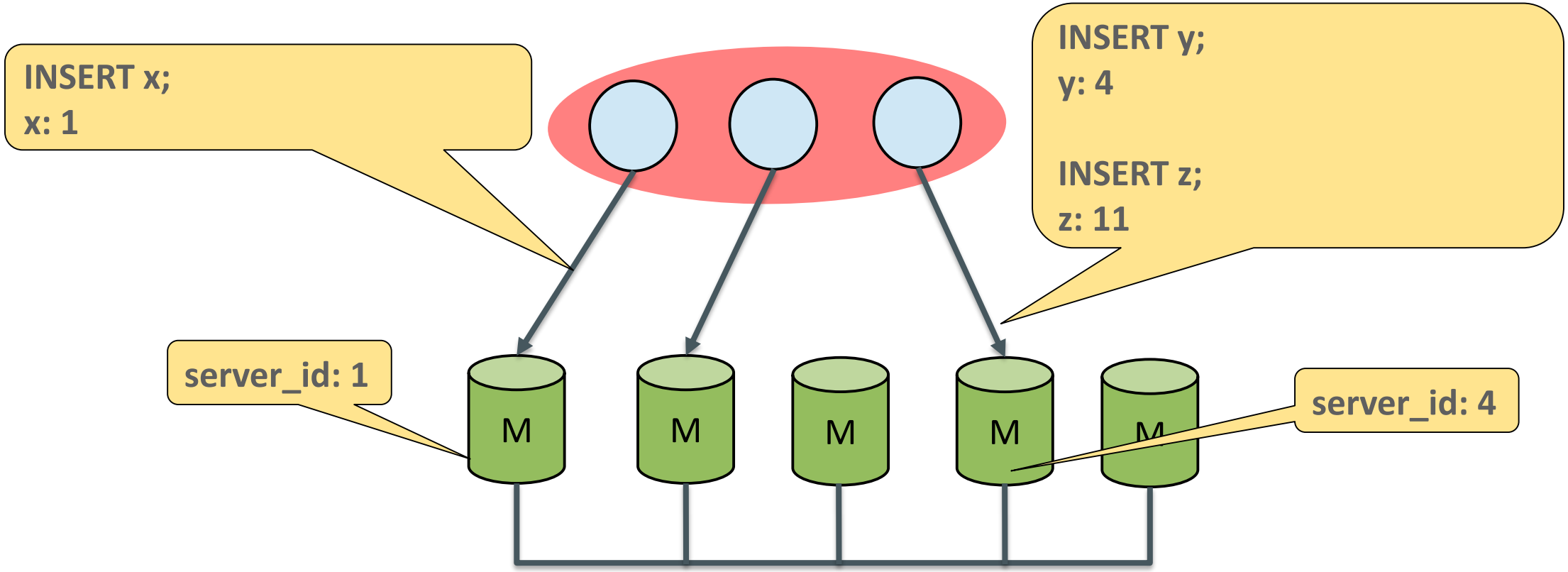
- By default, the offset is provided by server_id and increment is 7 [1].



[1]: http://mysqlhighavailability.com/mysql-group-replication-auto-increment-configuration-handling/

Auto-increment configuration/handling

- Users can change the increment size to their needs using GROUP_REPLICATION_AUTO_INCREMENT_INCREMENT option.



2 MySQL Group Replication

- 2.1 Multi-Master
- 2.2 Automatic distributed server recovery
- 2.3 MySQL/InnoDB look & feel
- 2.4 Full GTID support
- 2.5 Auto-increment configuration/handling
- 2.6 New distributed agreement and communication engine

New Distributed Agreement and Communications Engine

- Multiple OS support.
  - Linux, but also Windows, OSX, Solaris, FreeBSD…
- No third-party software required.
- No network multicast support required.
  - MySQL Group Replication can now operate on cloud based installations on which multicast is disallowed.
- No message size limit.
- No separate process.
  - MySQL Group Replication is now self-contained on the same software stack.

2 MySQL Group Replication

- 2.3 MySQL/InnoDB look & feel
- 2.4 Full GTID support
- 2.5 Auto-increment configuration/handling
- 2.6 New distributed agreement and communication engine
- 2.7 Requirements
- 2.8 Limitations

Requirements (by design)

- Support for InnoDB only.
- Primary key is required on every table.
- Requires global transaction identifiers turned on.
- Optimistic execution: transactions may abort on COMMIT due to conflicts with concurrent transactions on other members.

2 MySQL Group Replication

- 2.3 MySQL/InnoDB look & feel
- 2.4 Full GTID support
- 2.5 Auto-increment configuration/handling
- 2.6 New distributed agreement and communication engine
- 2.7 Requirements
- 2.8 Limitations

Limitations

- Concurrent schema changes are not supported.

3 Architecture

3.1 Introduction

MySQL Group Replication is

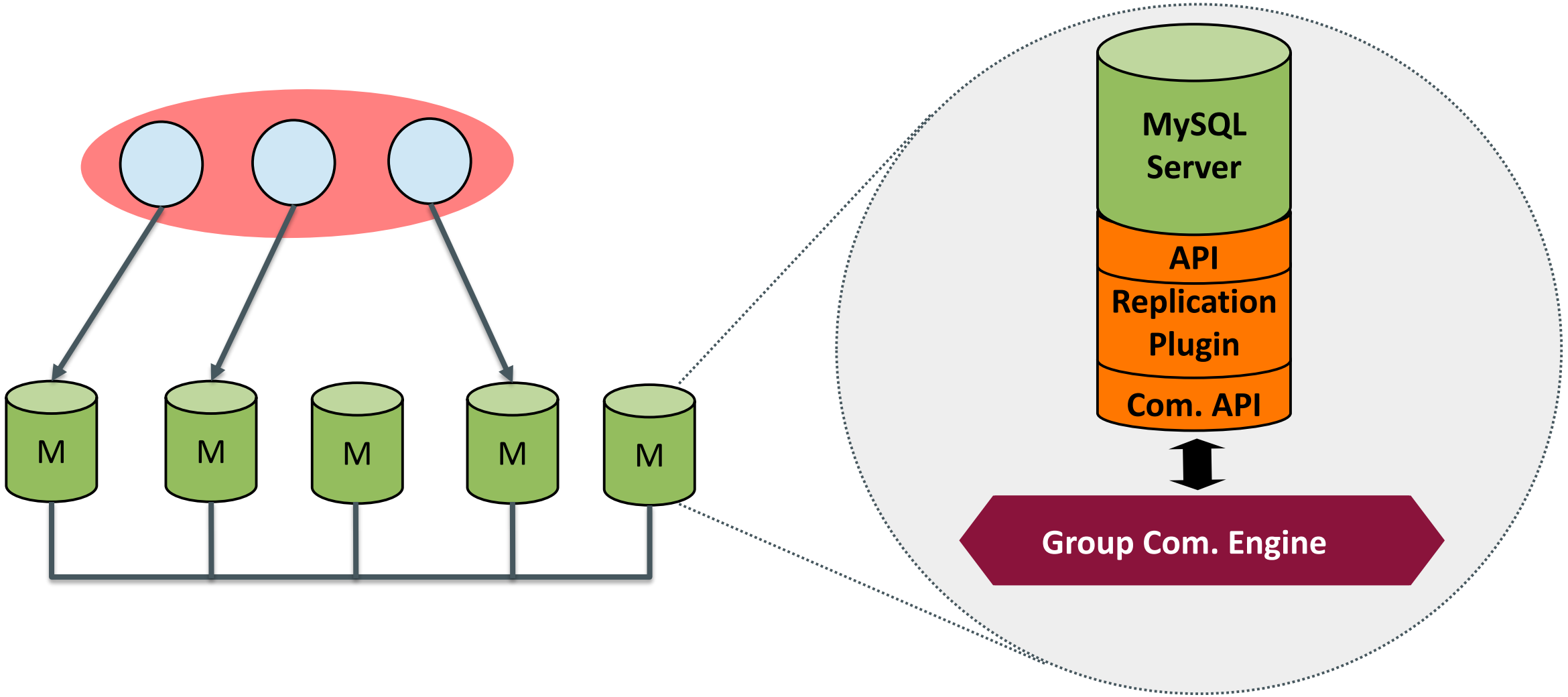
- **Built on top of proven technology!**
  - Shares much of MySQL Replication infrastructure – thence does not feel alien!
  - Multi-Master approach to replication.
- **Built on reusable components!**
  - Layered implementation approach.
  - Interface driven development.
  - Decoupled from the server core.
  - The plugin registers as listener to server events.
  - Reuses the capture procedure from regular replication.
  - Provides further decoupling from the communication infrasctructure.

3 Architecture

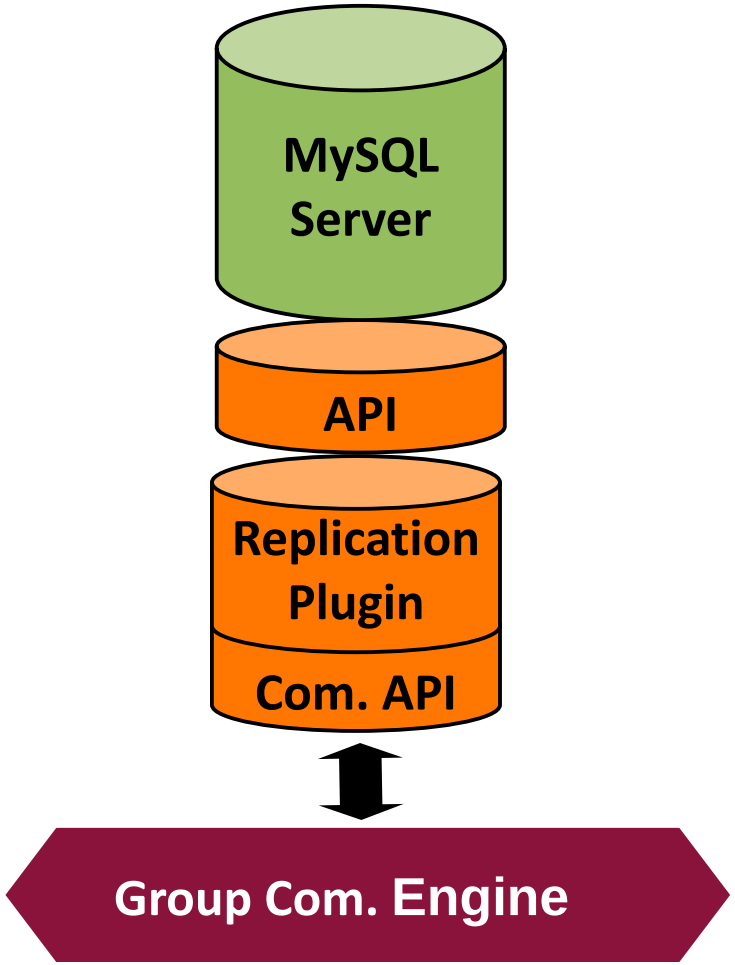
3.1 Introduction

3.2 Major Building Blocks

Major Building Blocks (1)

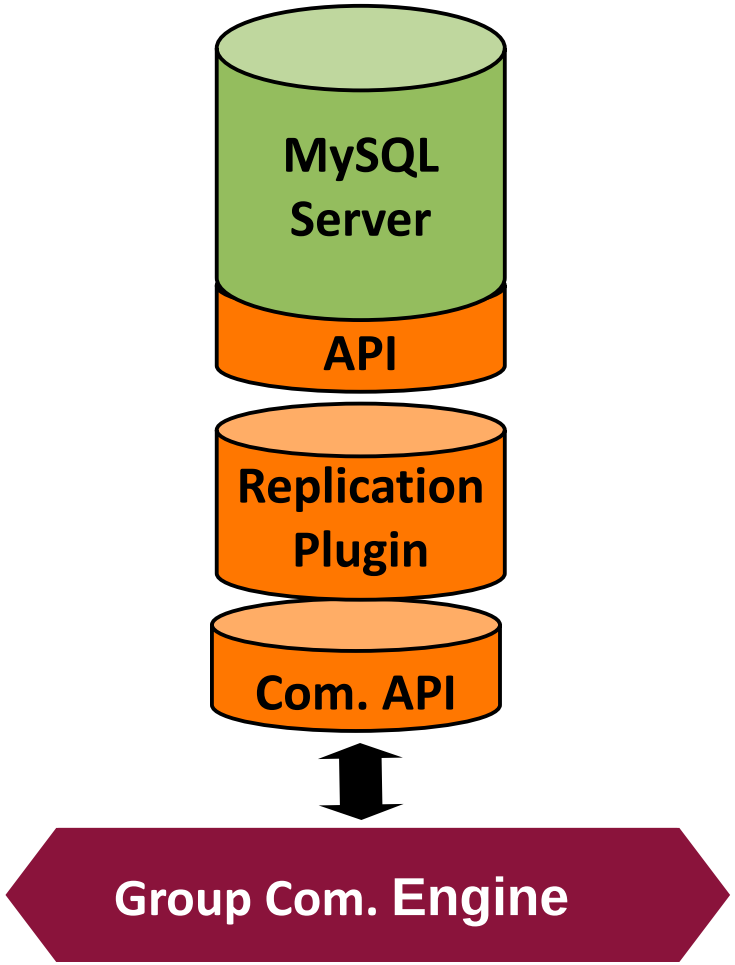


Major Building Blocks (2)



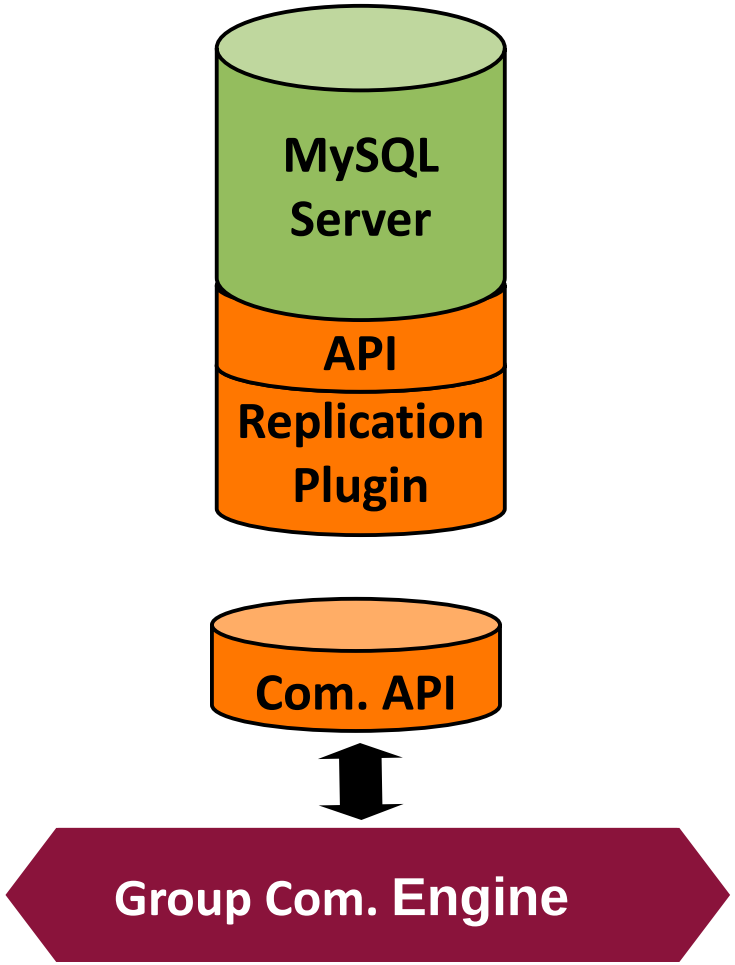
- Server calls into the plugin through a generic interface
  - (Most of server) internals are hidden from the plugin.
  - Some of the semi-sync interfaces were reused. Others were deployed.
- Plugin interacts with the server through a generic interface
  - Replication plugin determines the fate of the commit operation through a well defined server interface.
  - The plugin makes use of the relay log infrastructure to inject changes in the receiving server.

Major Building Blocks (3)



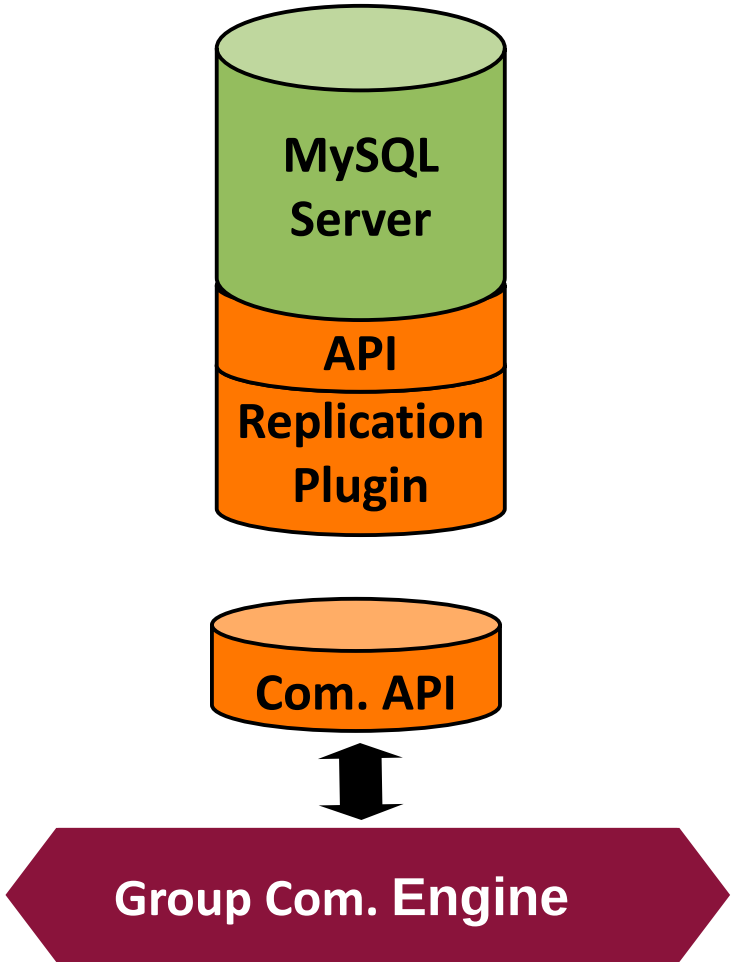
- The plugin is responsible for
  - Maintaining distributed execution context.
  - Detecting conflicts.
  - Handling distributed recovery:
    - Detect membership changes;
    - Donate state if needed;
    - Collect state if needed.
  - Receiving and handling transactions from other members.
  - Deciding the fate of on-going transactions.

Major Building Blocks (4)



- The communication API (and bindings) is responsible for:
  - Abstracting the underlaying communication engine from the plugin itself.
  - Maping the interface to a specific communication engine.

Major Building Blocks (5)



- The communication engine:
  - Variant of Paxos developed at MySQL.
  - Building block to provide distributed agreement between servers.

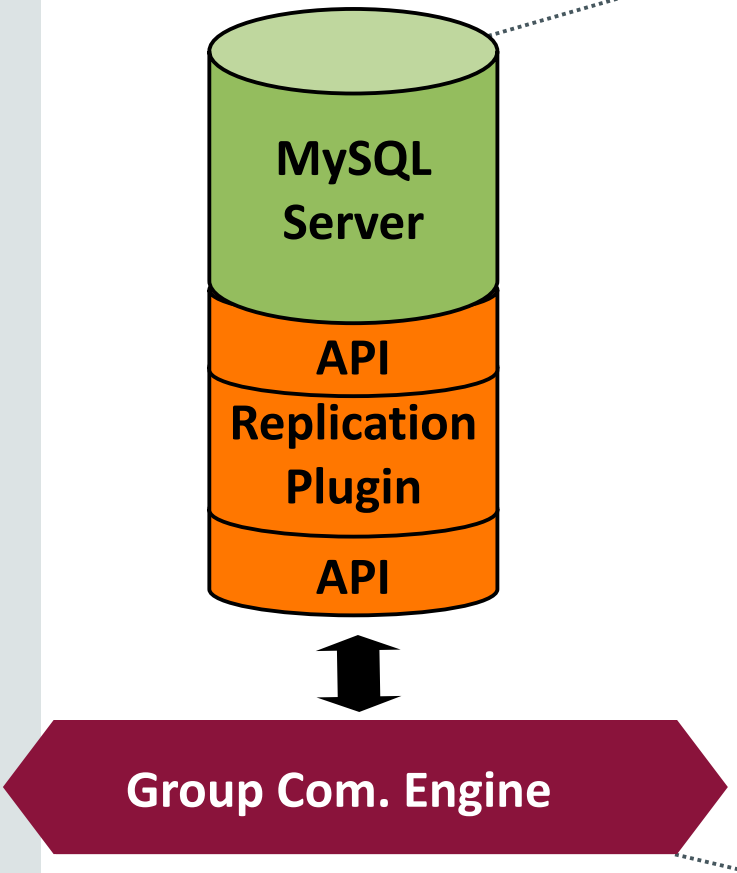
3 Architecture

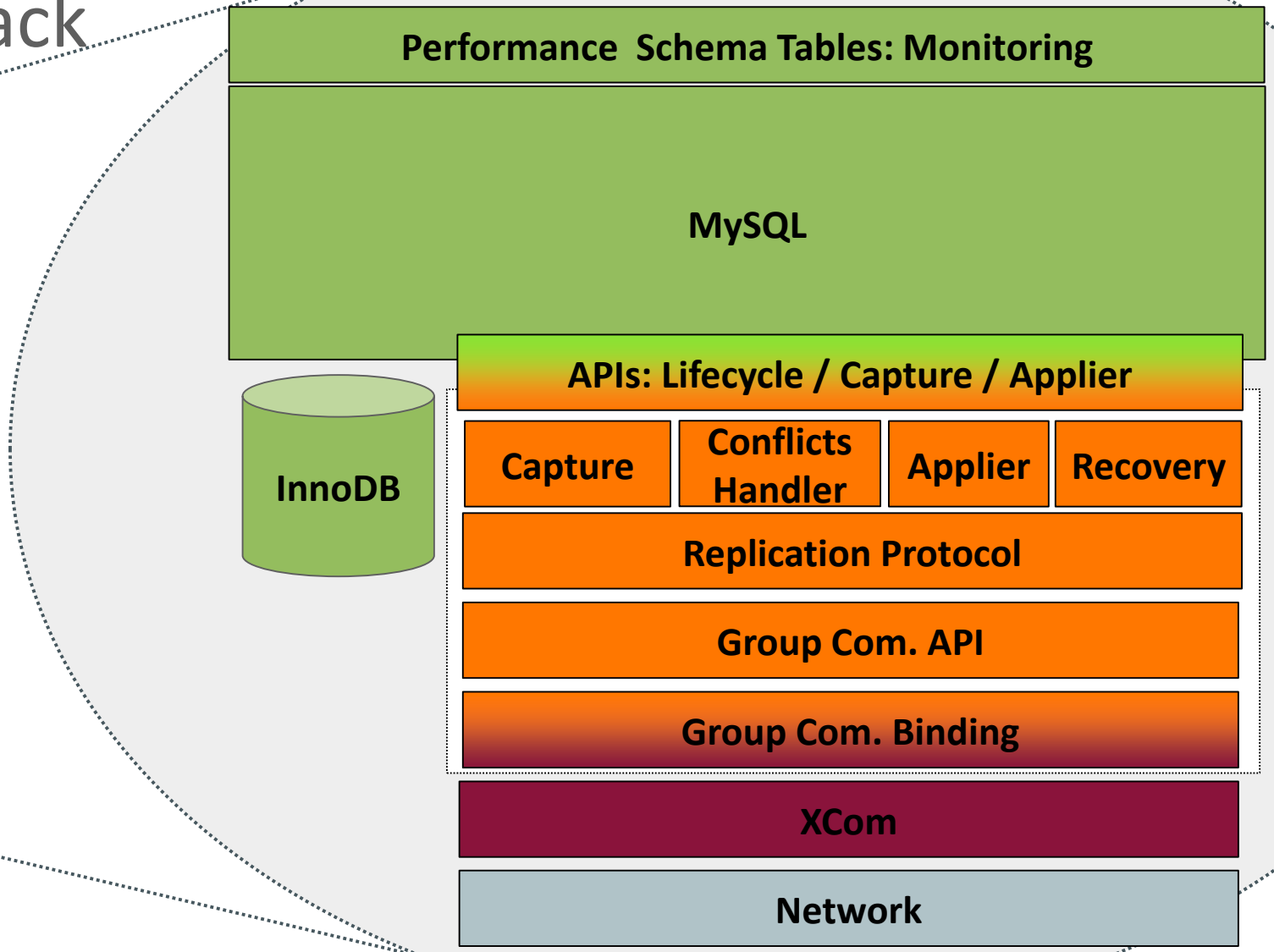
3.1 Introduction

3.2 Major Building Blocks

3.3 The Complete Stack

The Complete Stack





4 Use cases

4.1 Use cases

Use Cases

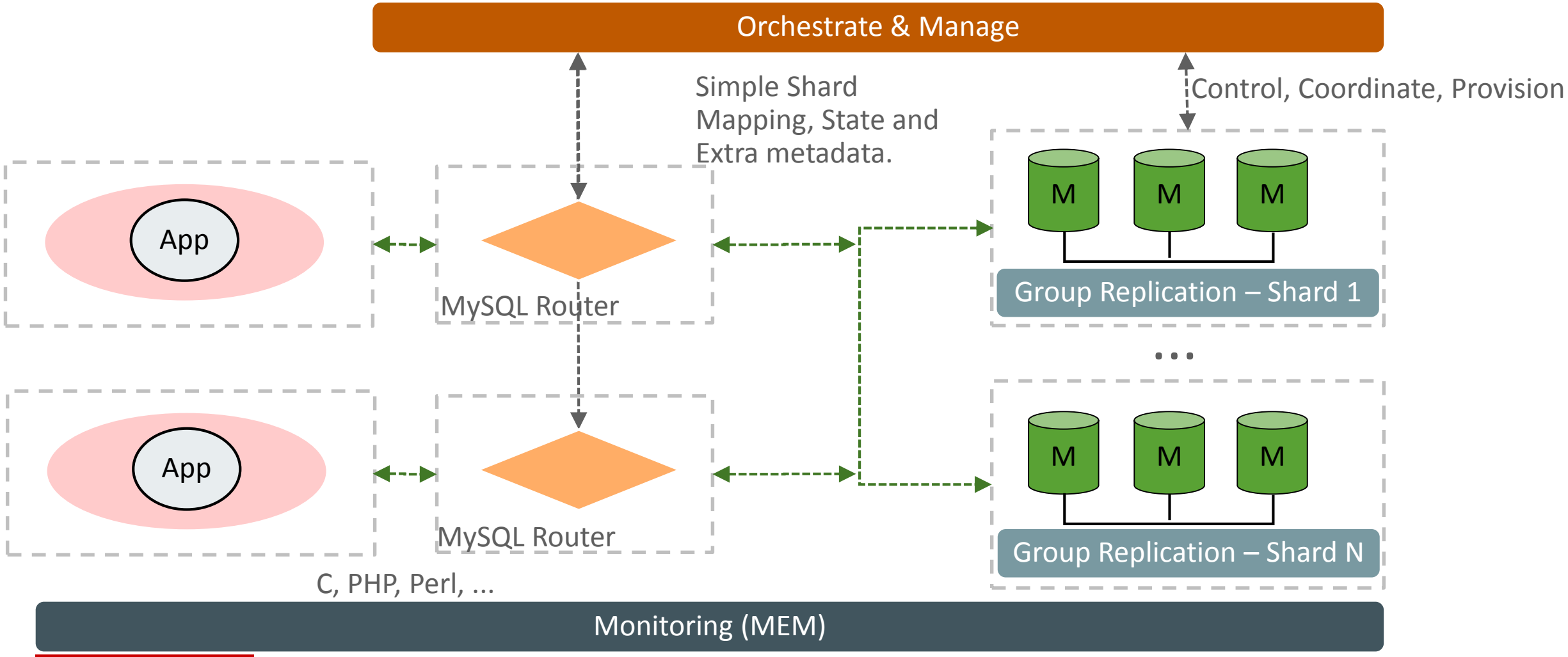
- **Elastic Replication**
  - Environments that require a very fluid replication infrastructure, where the number of servers has to grow or shrink dynamically and with as little pain as possible.
- **Highly Available Shards**
  - Sharding is a popular approach to achieve write scale-out. Users can use MySQL Group Replication to implement highly available shards. Each shard can map into a Replication Group.
- **Alternative to Master-Slave replication**
  - It may be that a single master server makes it a single point of contention. Writing to an entire group may prove more scalable under certain circumstances.

4 Use cases

4.1 Use cases

4.2 Big Picture

Dependable and Scalable MySQL Setups



5 Conclusion

Summary

- **Cloud Friendly**
  - Great techonology for deployments where elasticity is a requirement, such as cloud based infrastructures.
- **Integrated**
  - With server core through a well defined API.
  - With GTIDs, row based replication, performance schema tables.
- **Autonomic and Operations Friendly**
  - It is self-healing: no admin overhead for handling server fail-overs.
  - Provides fault-tolerance, enables multi-master update everywhere and a dependable MySQL service.

Where to go from here?

- Packages
  - http://labs.mysql.com

- Blogs from the Engineers (news, technical information, and much more)
  - http://mysqlhighavailability.com


ORACLE®