# 新世代MySQL高可用方案

杜修文
Principal Sales Consultant
Oracle MySQL 全球事业部

# 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.

# 资料库如何高可用?

- 不只一篮鸡蛋

# MySQL 高可用方案是什么?

独

自

转

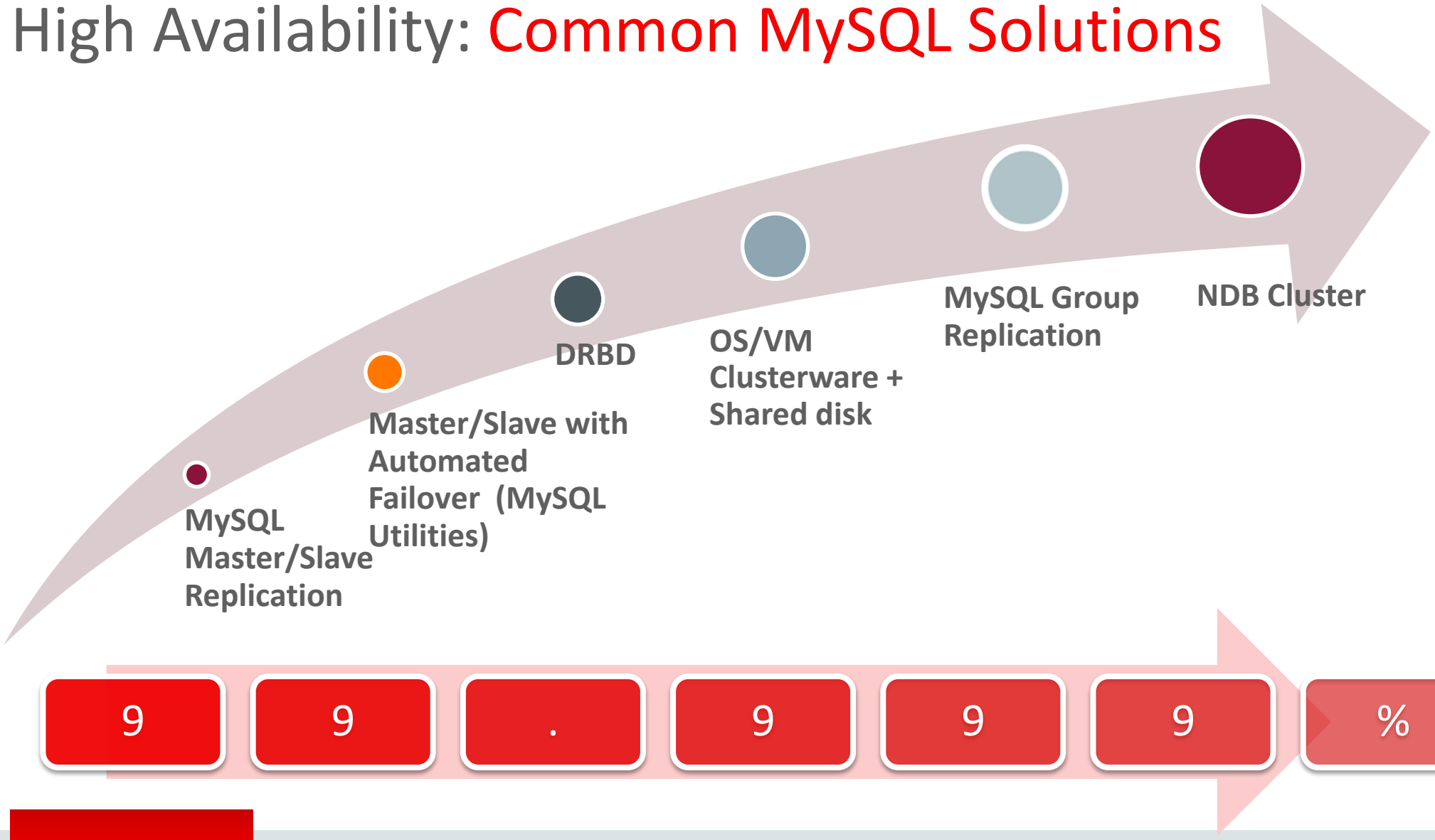
# MySQL高可用方案是什么?

- 各组件独立,没有共用
- 运作自动化,不需人工介入
- 故障移转对应用程式透明的

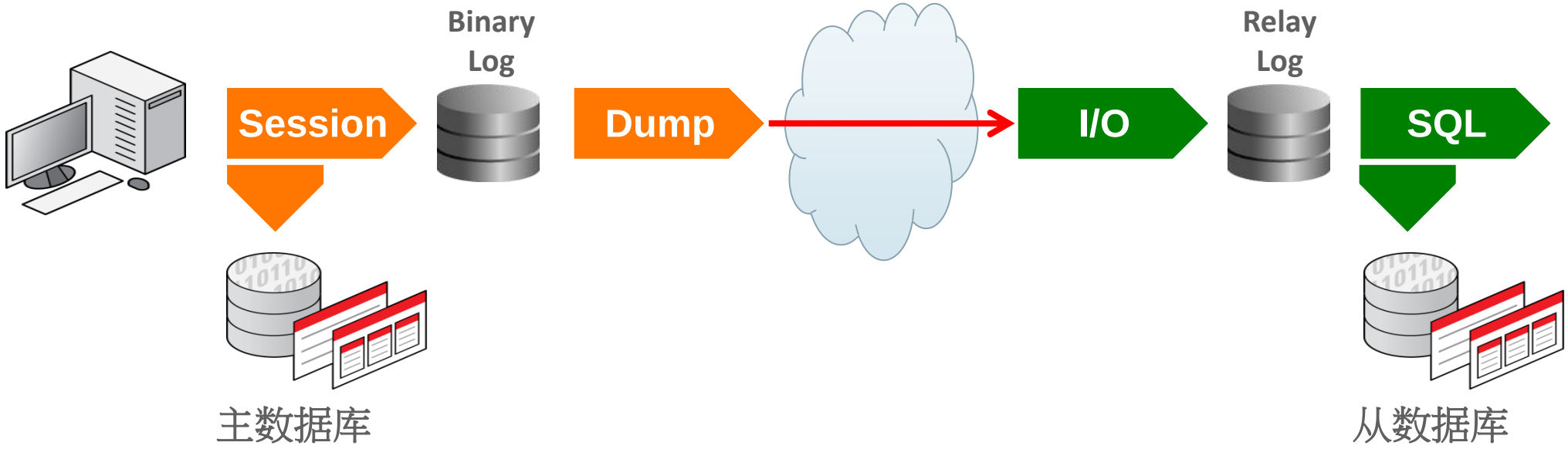
# MySQL高可用方案是什么?

- 各组件独立,没有共用
- 运作自动化,不需人工介入
- 故障移转对应用程式透明的

# High Availability: Common MySQL Solutions

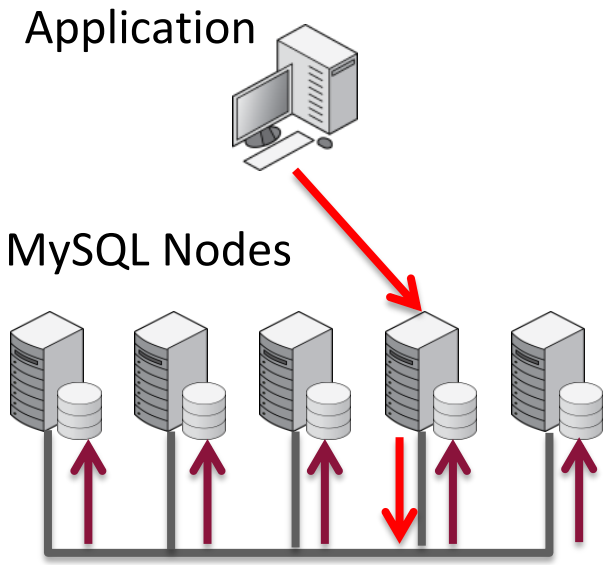


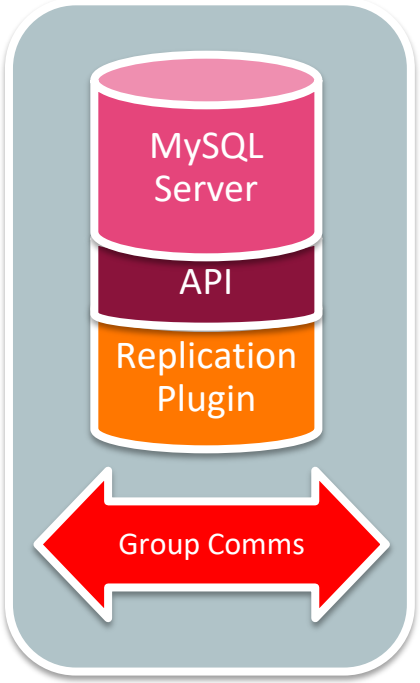
# MySQL复制的流程-非同步复制




- Session thread处理来自应用的更新 – 将资料写到主数据库和相关事件到二进位日志
- Dump thread:自二进位日志读事件且将之送到从库
- I/O thread收复制事件且将之存到从库的缓存日志
- SQL thread:自从库的缓存日志读复制事件且将之写到从库中

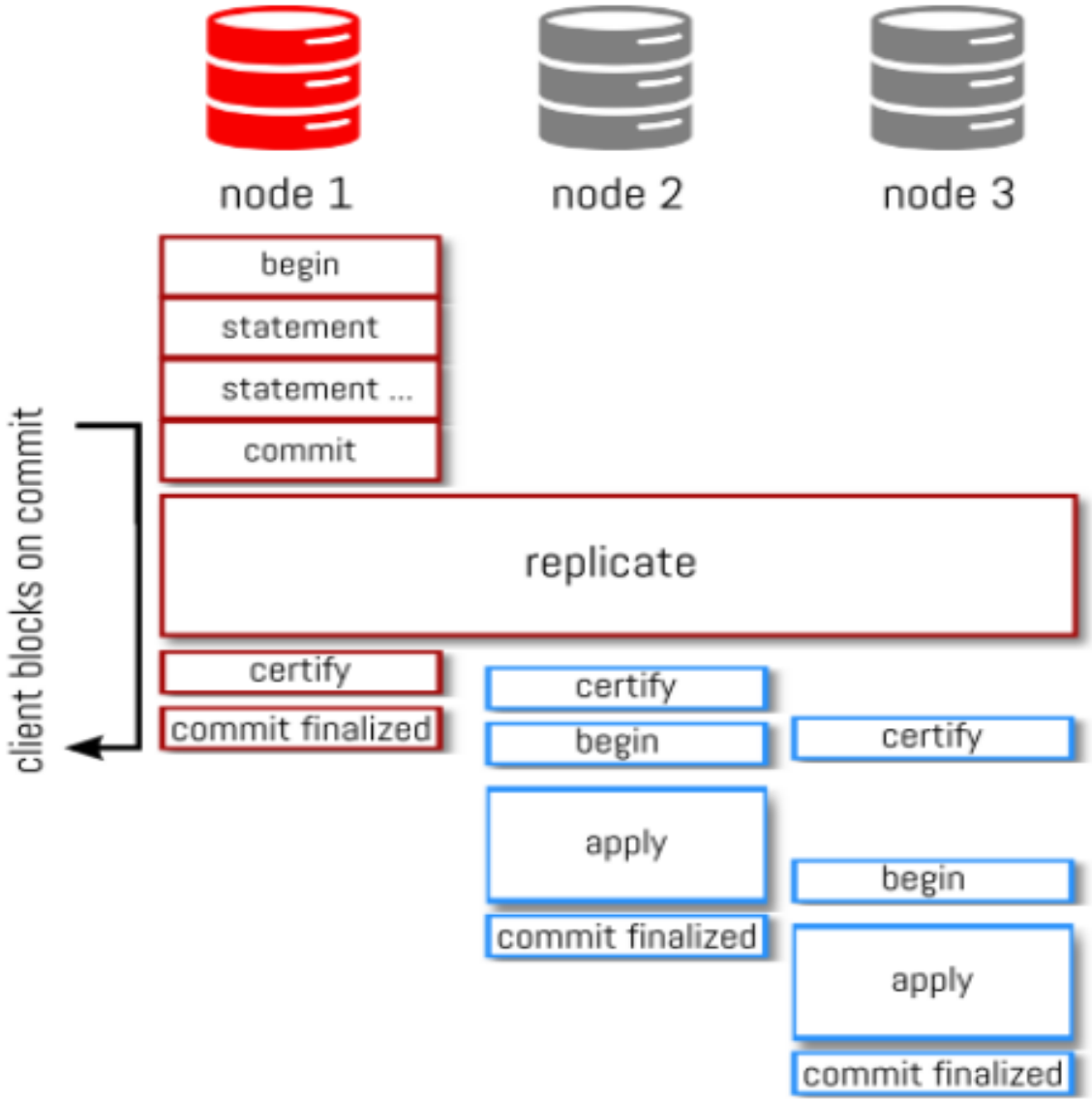
# MySQL群组复制




- 无共用实时同步数据库系统
  - 任何一个服务器都可以更新的多主
  - 可侦测冲突且解决之(事务转回)
  - Optimistic State Machine Replication
- 自动组成员管理和故障侦测
  - 不需服务器层的故障移转
  - 弹性横向扩张和收缩
  - 无单点故障
  - 自动重构
- 完整整合
  - InnoDB
  - GTID-based replication
  - PERFORMANCE_SCHEMA

# MySQL Group Replication
## Full Transaction life cycle



# 必要修件 (设计上的规划)

- 只支持InnoDB.
- 每张表都要有用户定的主键.
- 要开启全局事务辨识码(GTID).
- 乐观执行事务: 事务有可能在提交时,因为和组内其他成员的事务冲突而放弃.
- 同一组内最多可有9个服务器.
- 二进位日志活动(ROW)
- applier服务器要开启二进位日志且用复制的applier (SQL)线程写入.

# MySQL Group Replication: 布署模式

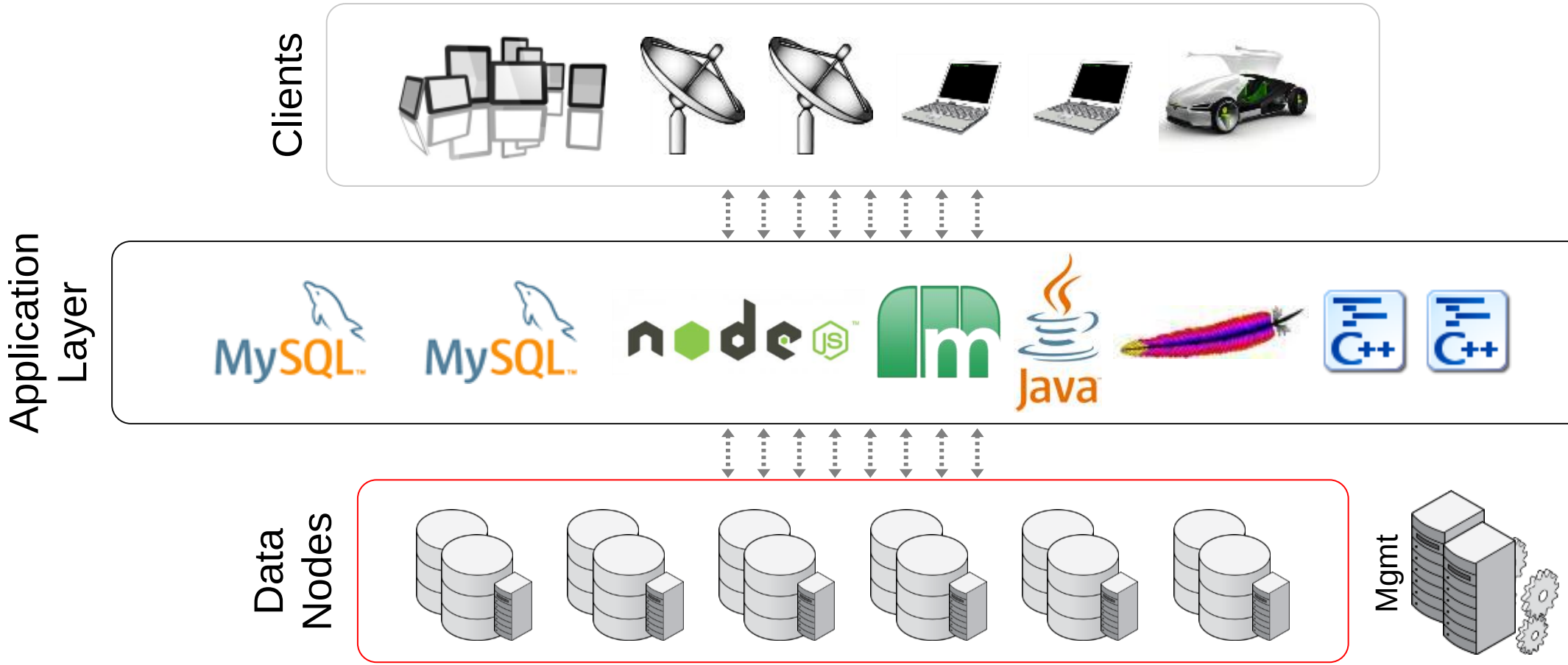
- Single-Primary Mode
  - 组内只有一个单一主服务器,用于读-写.其他的服务器都为只读.
- Multi-Primary Mode
  - 组内所有服务器任何时间都可以执行所有动作,包括更改状态(RW transactions)

# 限制

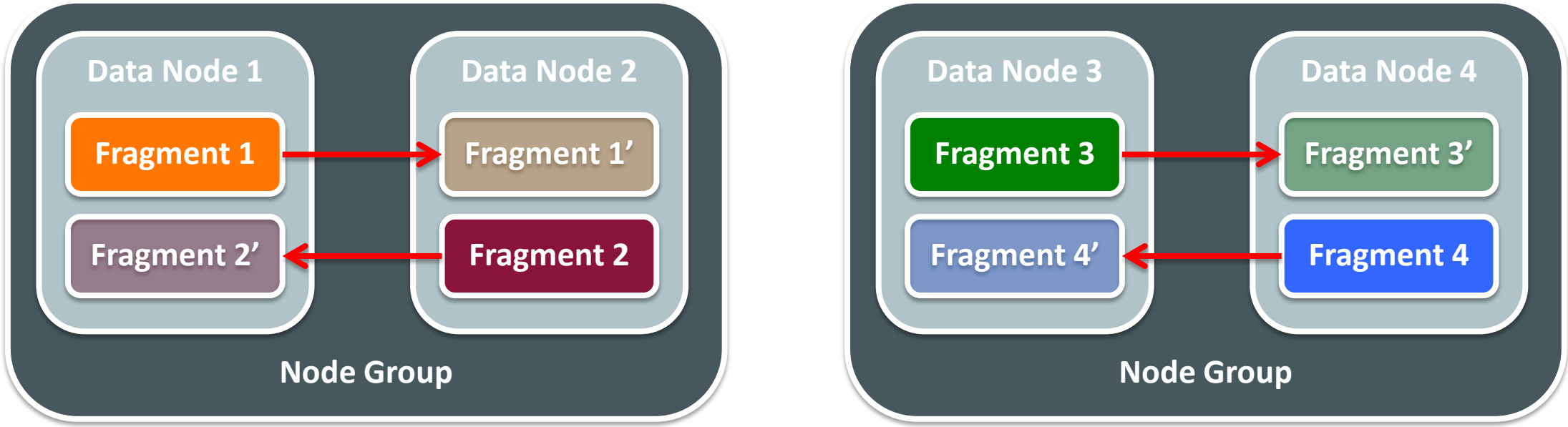
- 不支持SERIALIZABLE 隔离级别
- 不支持Savepoints
- 只有IPV4
- 很大的事务,抄到组内其他服务器需要5秒以上时,可能造成群组沟通的失败
- GR不会处理人工加的表锁和命名的锁
- 在多主模式
  - 不支持在不同服务器同一个对象所做的并发的DDL和DDL+DML
  - 不支持外键的cascade约制
  - 不支持有多层外键相依的表,特别是定有CASCADING的外键

# MySQL集群的架构

## 无单点故障,能在线扩容



# 数据分片

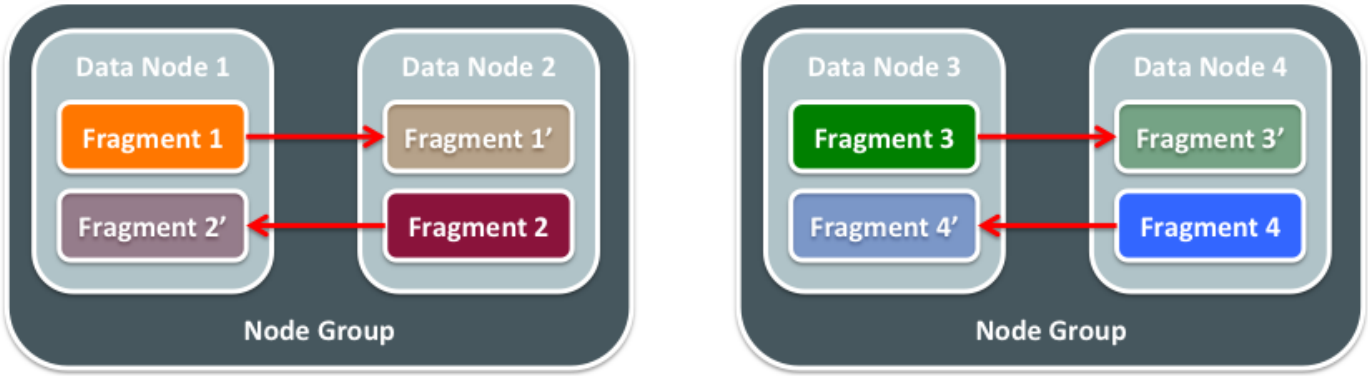



- 数据在节点组间的分布是对应用透明的
- 各Fragment在一个数据节点的更动以同步的方式复制到同一数据节点组的其他节点

# 数据分片

Shard Key

| User-id (PK) | Service (PK) | Data |
|---|---|---|
| 1773467253 | chat | xxx |
| 6257346892 | chat | xxx |
| 1773467253 | photos | xxx |
| 7234782739 | photos | xxx |
| 8235602099 | reminders | xxx |
| 8437829249 | location | xxx |




- DBA可选择主键的一部份做分片键
- 对分片键做哈希运算决定各行要分到那一个Fragment

# 主-主跨域复制

- 在MySQL Cluster实例间做非同步复制
  - 主-主
  - 任何地方都可以更新
  - 冲突侦测
    - 通过例通知应用
    - 可以选择自动解决冲突
  - 自动冲突解决
  - 冲到的交易和依附于其上的交易会回滚
- 不需更改应用的数据架构

# MySQL 是什么?

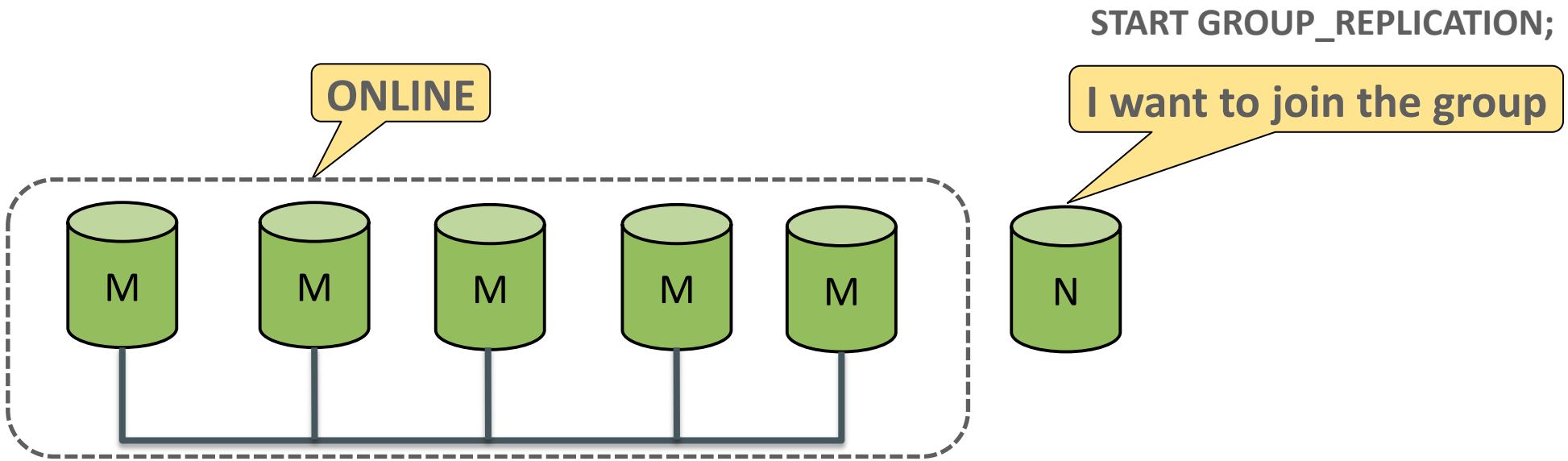
- 各组件独立,没有共用
- 运作自动化,不需人工介入
- 故障移转对应用程式透通的

# MySQL 是什么?

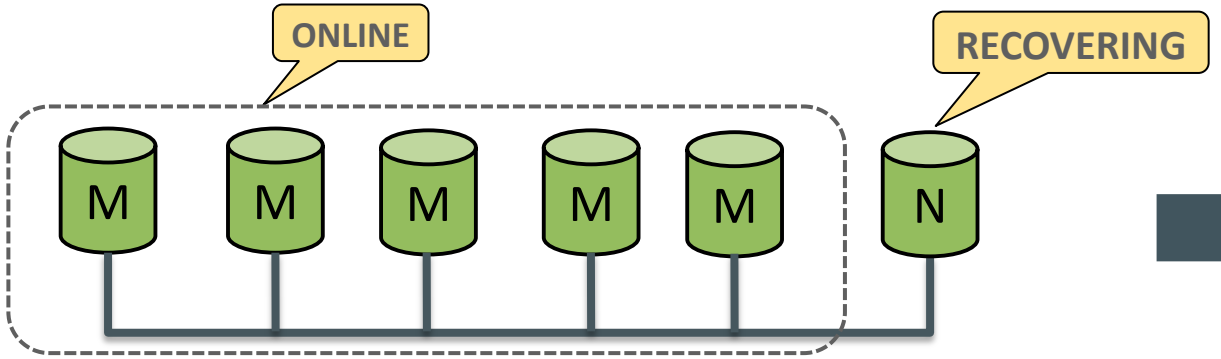
- 各组件独立,没有共用
- 运作自动化,不需人工介入
- 故障移转对应用程式透通的

# 为群组加上节点

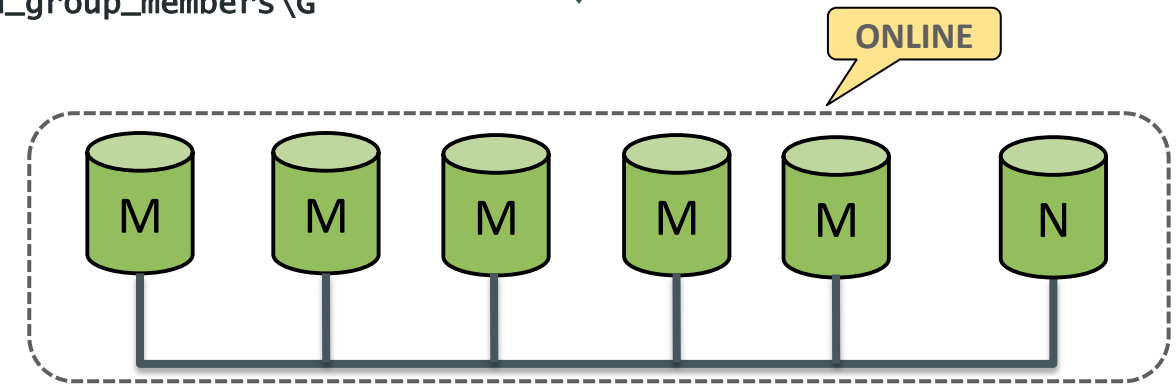
- 加入群组的节点自动找一个现有的(捐献者)节点同步数据
  - 会查该节点和其他节点的差异
  - 提示: 在加入新节点前先(例如以备份回复)抄一份基本数据到新节点



# 为群组加上节点

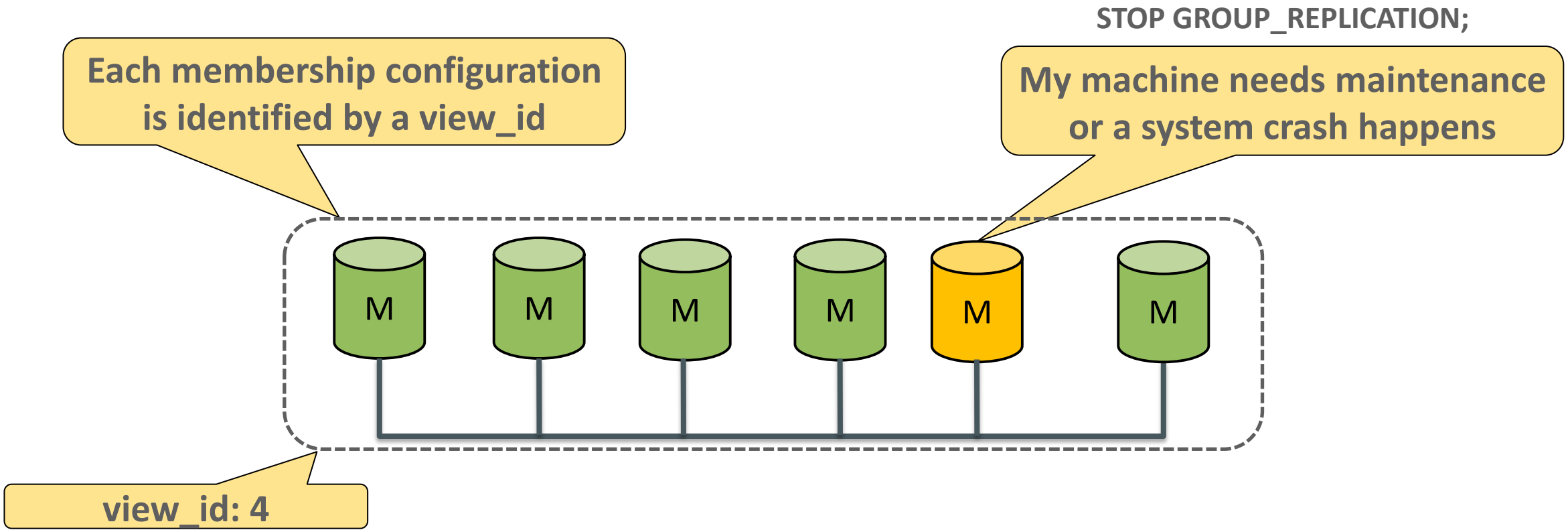



```
SELECT * FROM performance_schema.replication_group_members\G
```



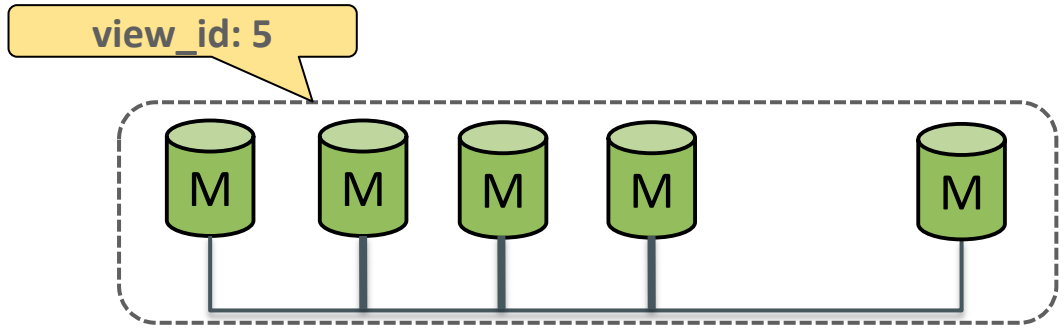
# 自动化群组管理

- 如果有一节点离开了群组,其他的节点会自动发现

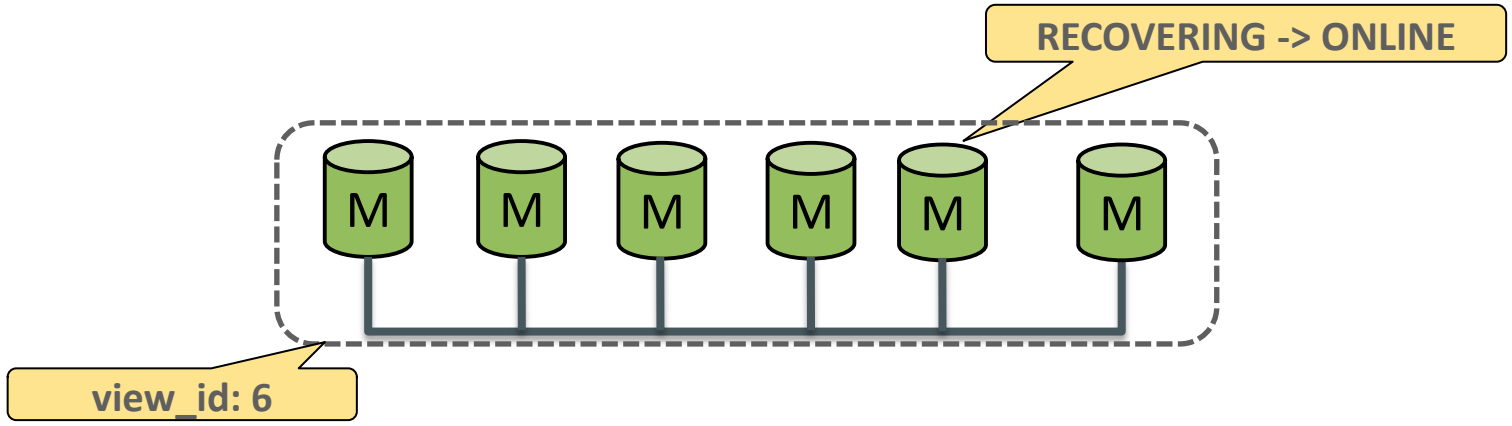


# 为群组加上节点

- 如果有一节点离开了群组,其他的节点会自动发

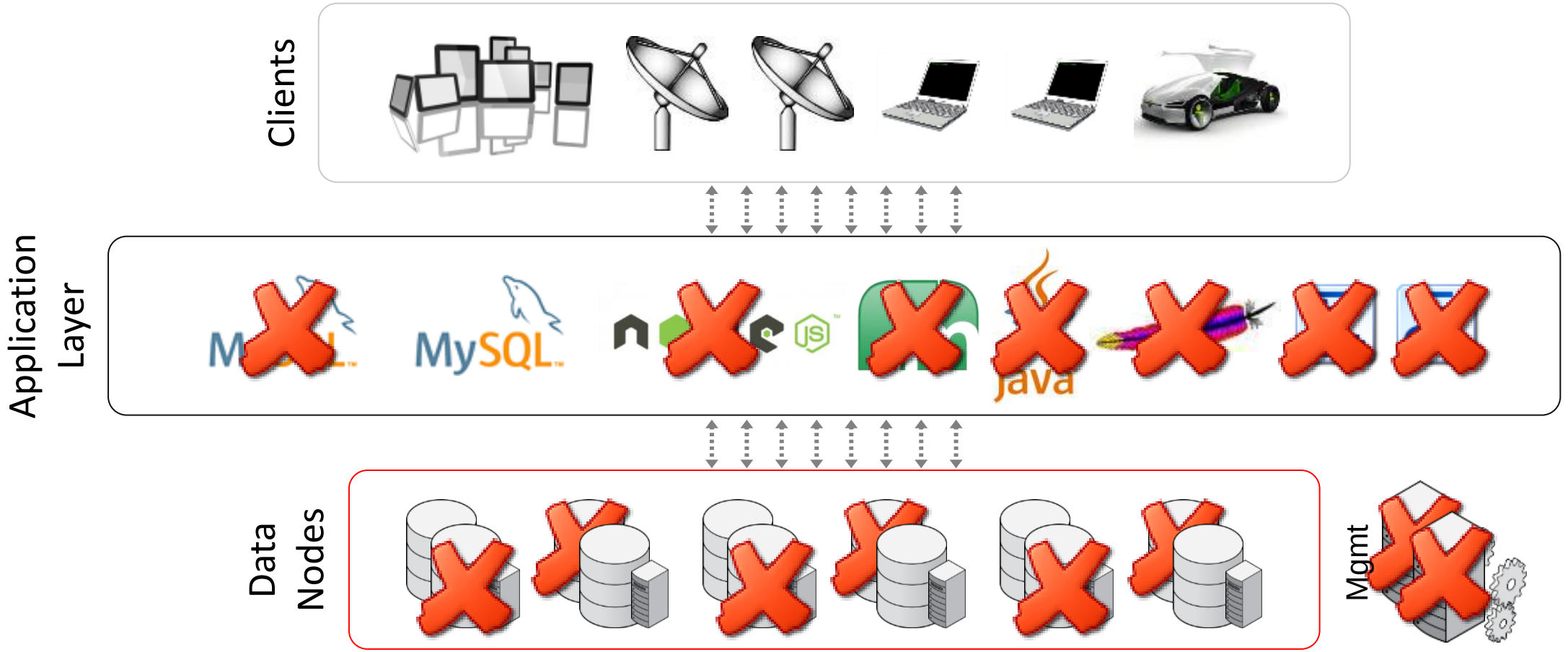



- 当节点回来时,会自动找一个现有的节点自动同步数据



# MySQL集群自动同步(回复的)数据节点

## 以Local Check Point+Global Check Point为回复基础

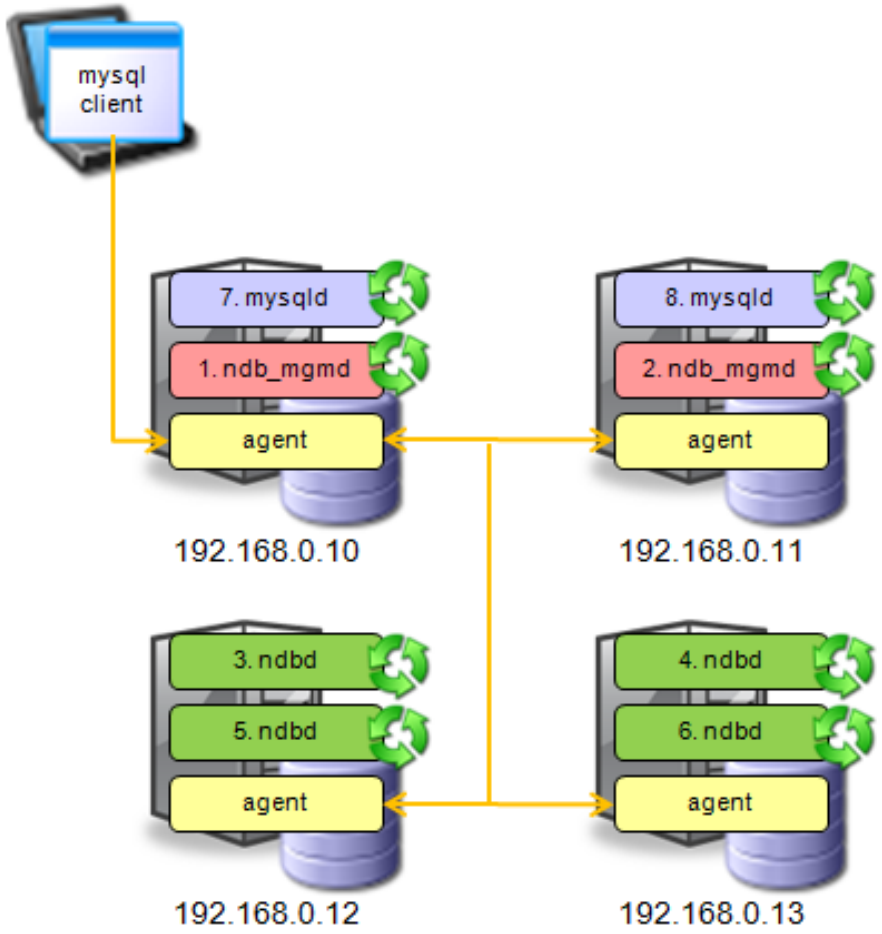


# MySQL Cluster Manager

1. 自edelivery.oracle.com 下载 MCM/Cluster package :
2. Unzip
3. 运行代理器, 定义, 建立和启动 Cluster!

```
$> bin\mcmd --bootstrap
MySQL Cluster Manager 1.1.2 started
Connect to MySQL Cluster Manager by running "D:\Andrew\Documents\MySQL\mcm\bin\mcm" -a NOVA:1862
Configuring default cluster 'mycluster'...
Starting default cluster 'mycluster'...
Cluster 'mycluster' started successfully
 ndb_mgmd NOVA:1186
 ndbd NOVA
 ndbd NOVA
 MySQLd NOVA:3306
 MySQLd NOVA:3307
 ndbapi *
Connect to the database by running "D:\Andrew\Documents\MySQL\mcm\cluster\bin\MySQL" -h NOVA -P 3306 -u root
```

# MCM: 升级 Cluster




```
MySQL> upgrade cluster
 --package=7.5 mycluster;
```

# MySQL 是什么?

- 各组件独立,没有共用
- 运作自动化,不需人工介入
- 故障移转对应用程式透通的

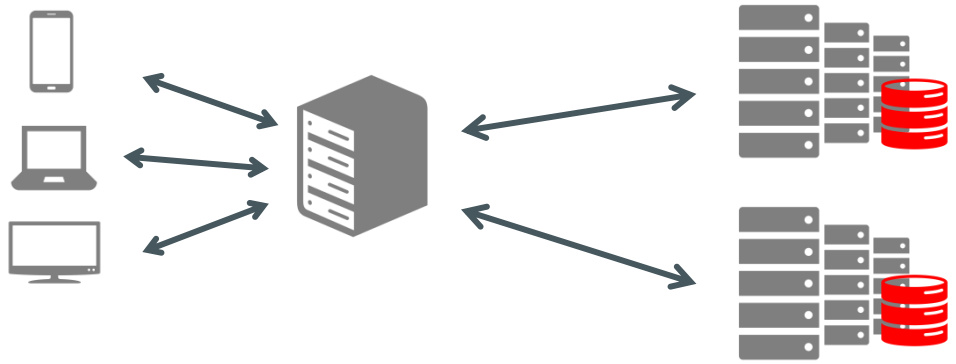
# MySQL 是什么?

- 各组件独立,没有共用
- 运作自动化,不需人工介入
- 故障移转对应用程式透通的

# 新的! MySQL Router

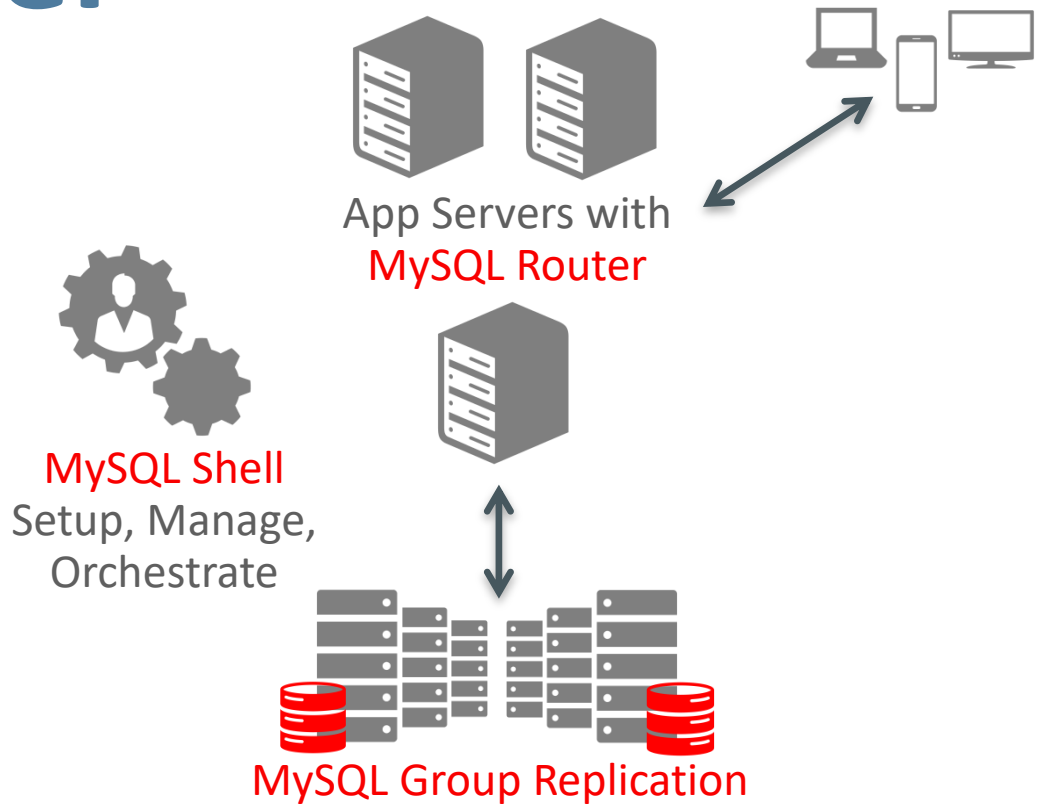
**更容易,和安全的使MySQL 应用更具扩充力**

- 智能的路由 **MySQL**连线和事务, 使您能专注在应用的开发
- 提供跨语言跨平台的支持, 通过分片带来高可用和横向扩充的能力

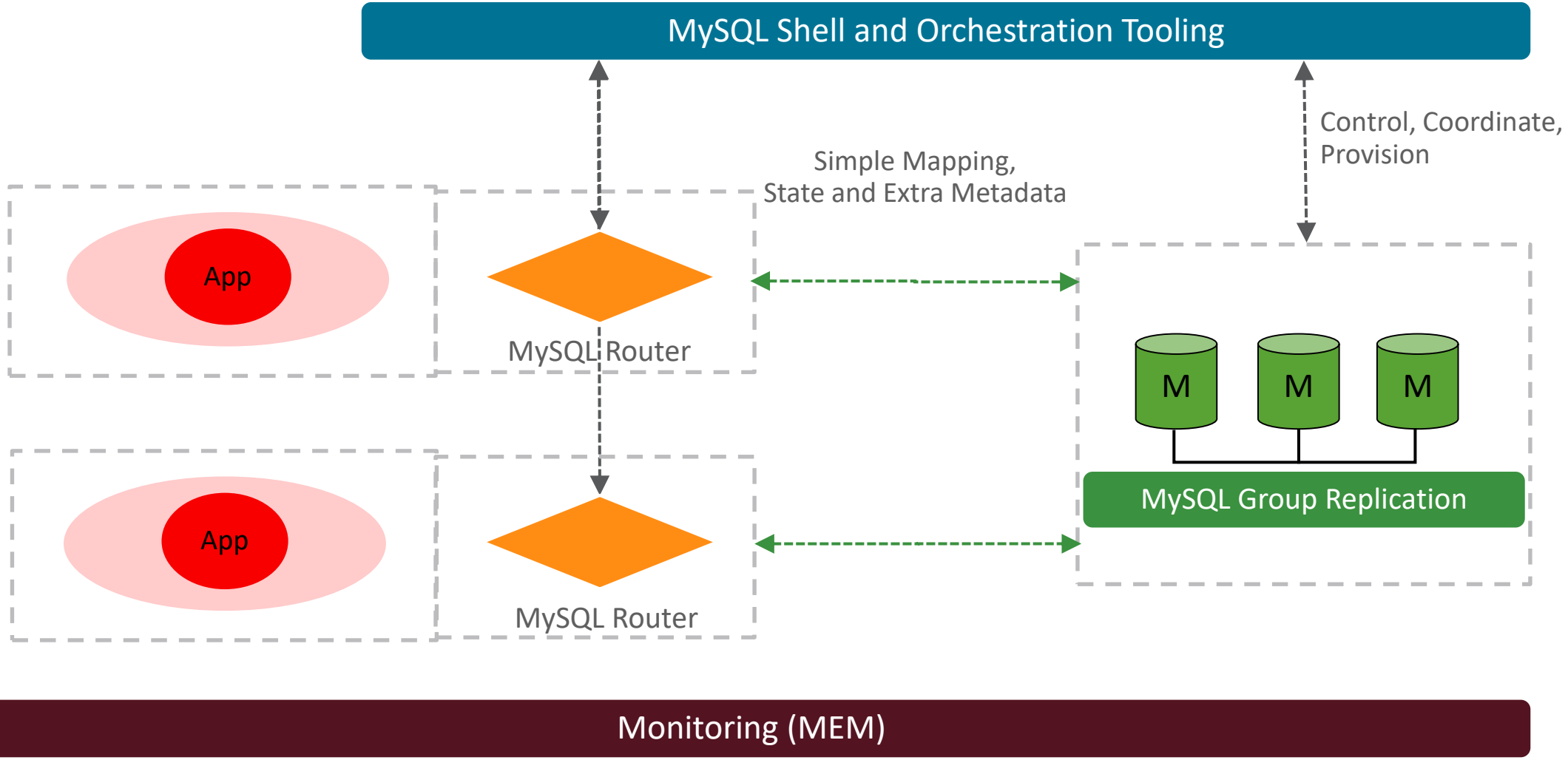


# MySQL InnoDB Cluster

*"High Availability becomes a core first class feature of MySQL!"*

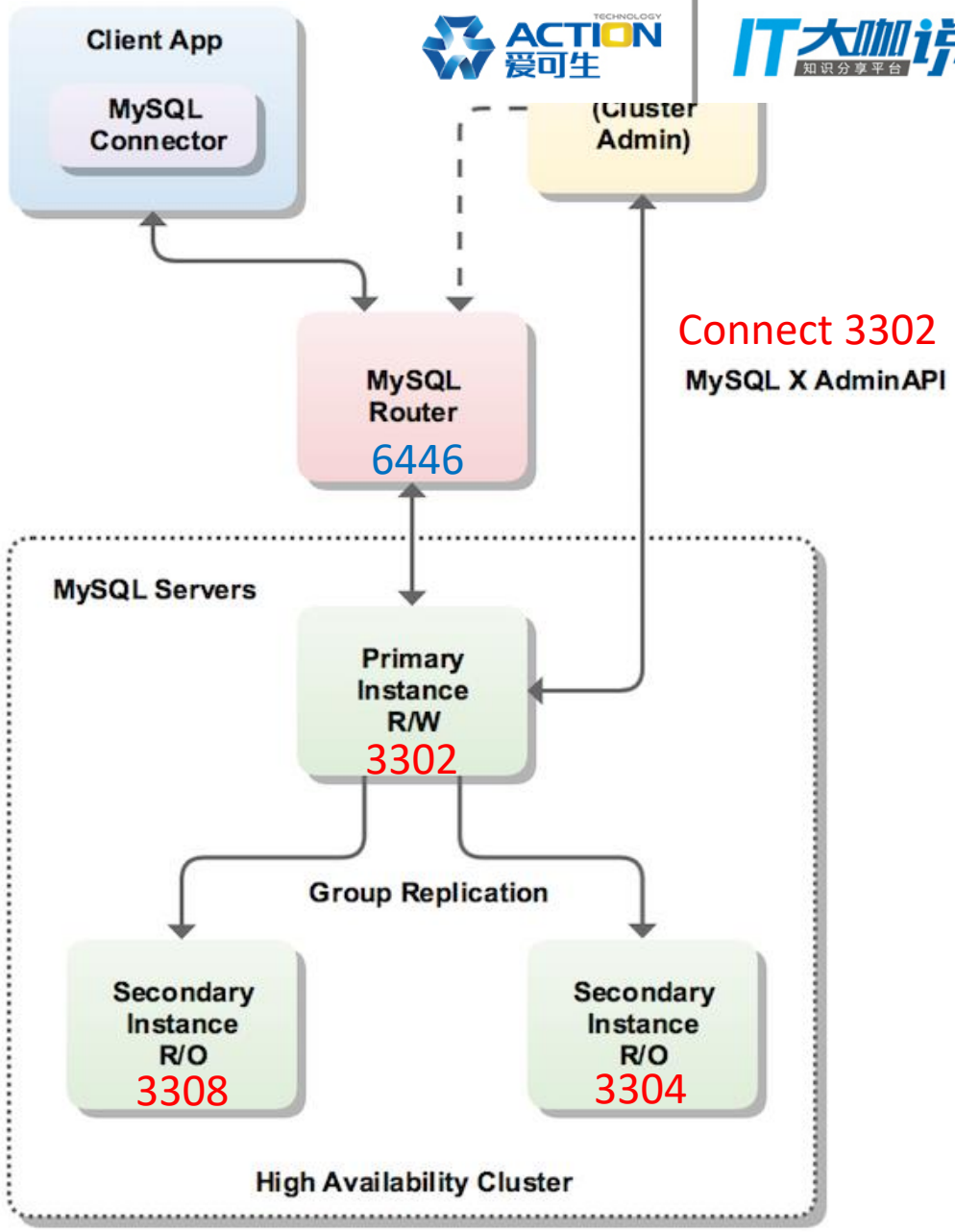


# MySQL InnoDB Cluster: The Big Picture



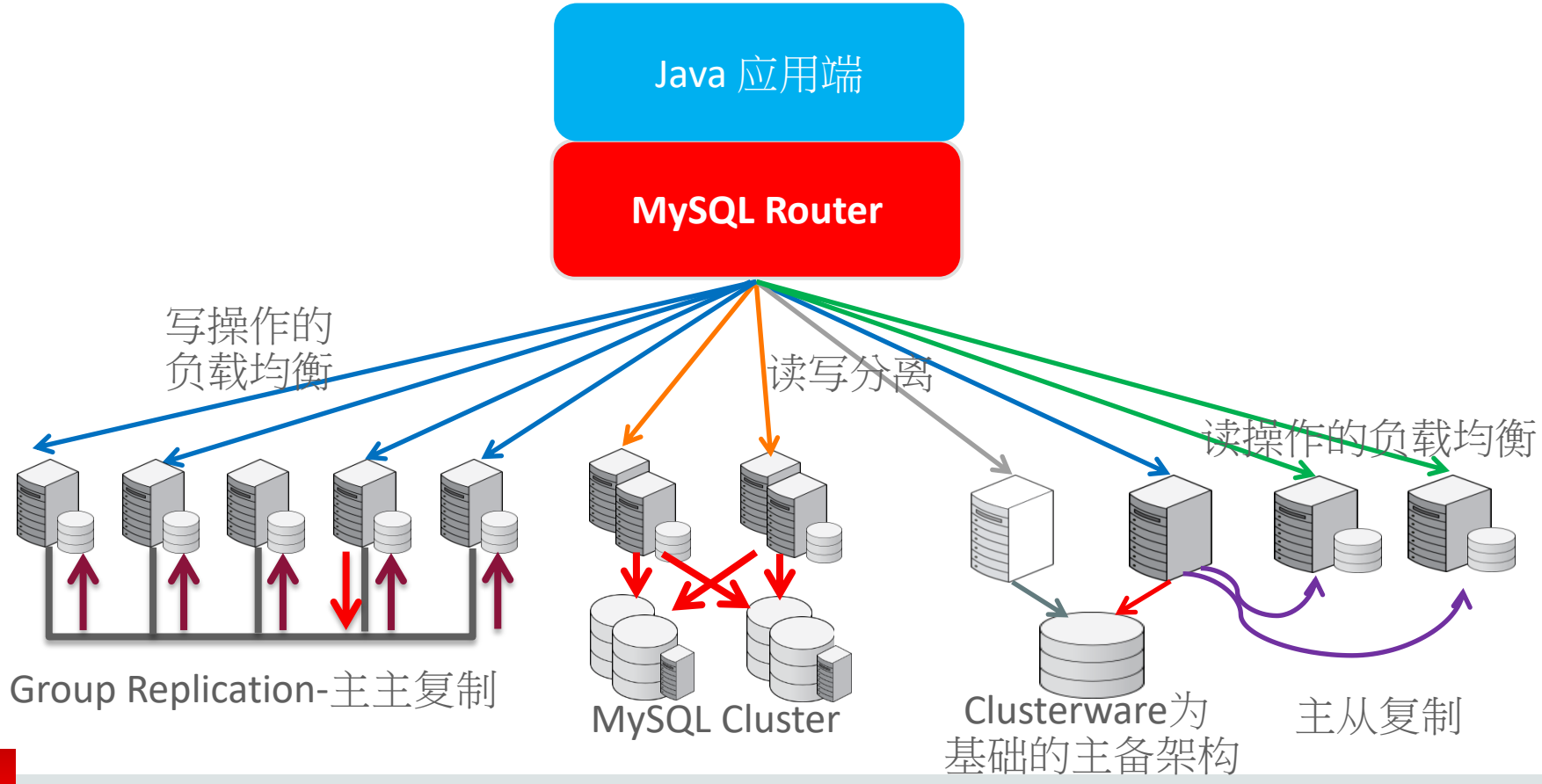
# Demo – InnoDB Cluster
## - mysqlsh –f startInnoDBCluster.js

```
mysql-js> dba.startSandboxInstance(3302)
mysql-js> dba.startSandboxInstance(3304)
mysql-js> dba.startSandboxInstance(3308)
mysql-js> \c root@192.168.56.101:3302
mysql-js> dba.rebootClusterFromCompleteOutage('mycluster')
mysql-js> var cluster=dba.getCluster('mycluster')
mysql-js>  cluster.status()
mysql-js> cluster.describe()
$ <outer-path>/bin/mysqlrouter –bootstrap – directory=<router-path-2-config>
$ <router-path> cat mysqlrouter.conf
$ <router-path-2-config>/start
$ mysql –uroot –p –hlocalhost –P6446
mysql> select @@port;
mysql-js>  dba.stopSandboxInstance(3302)
mysql> select @@port
mysql-js> cluster.status()
```



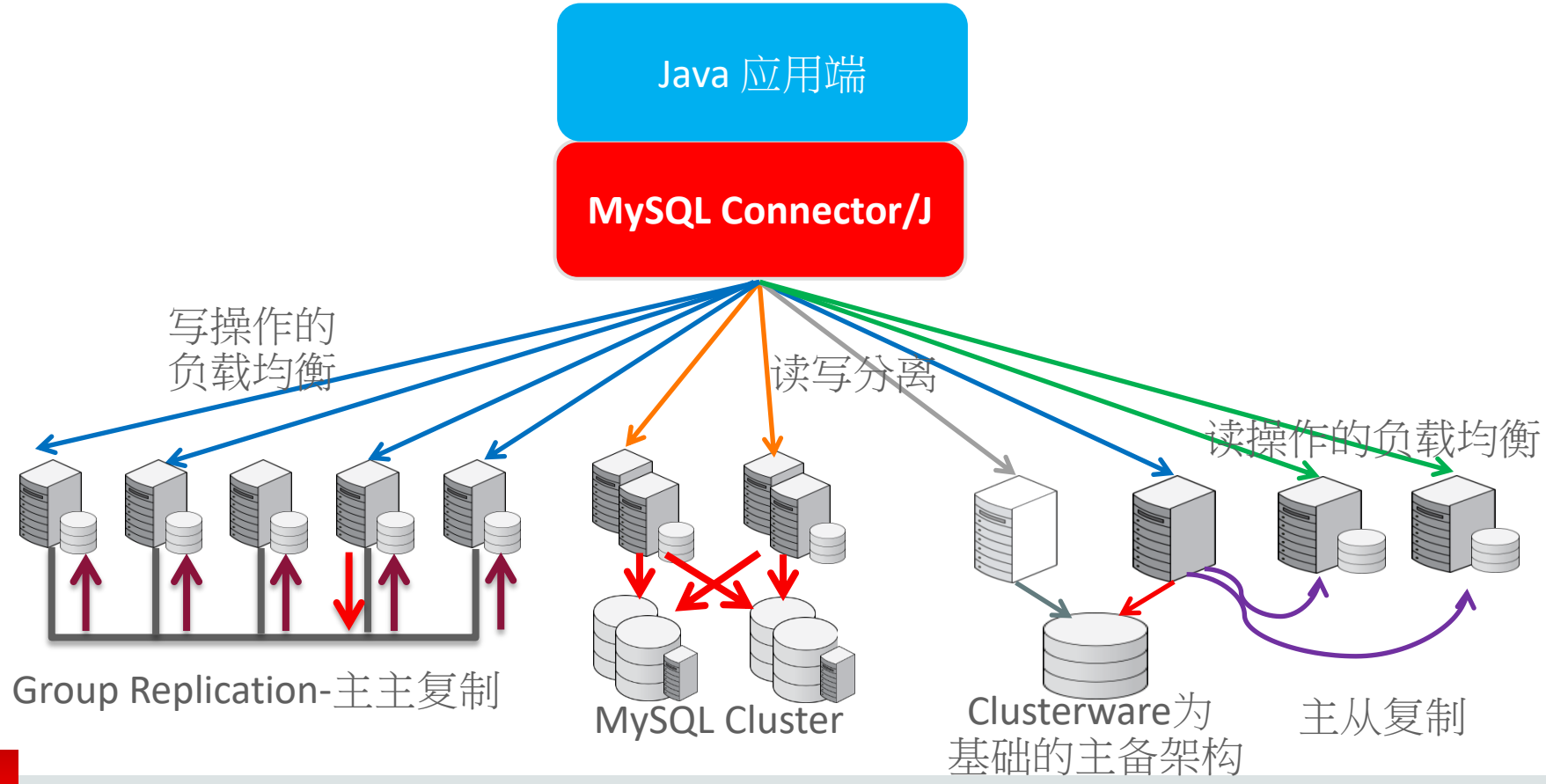
# MySQL HA With Router

- 支援负载均衡,故障移转



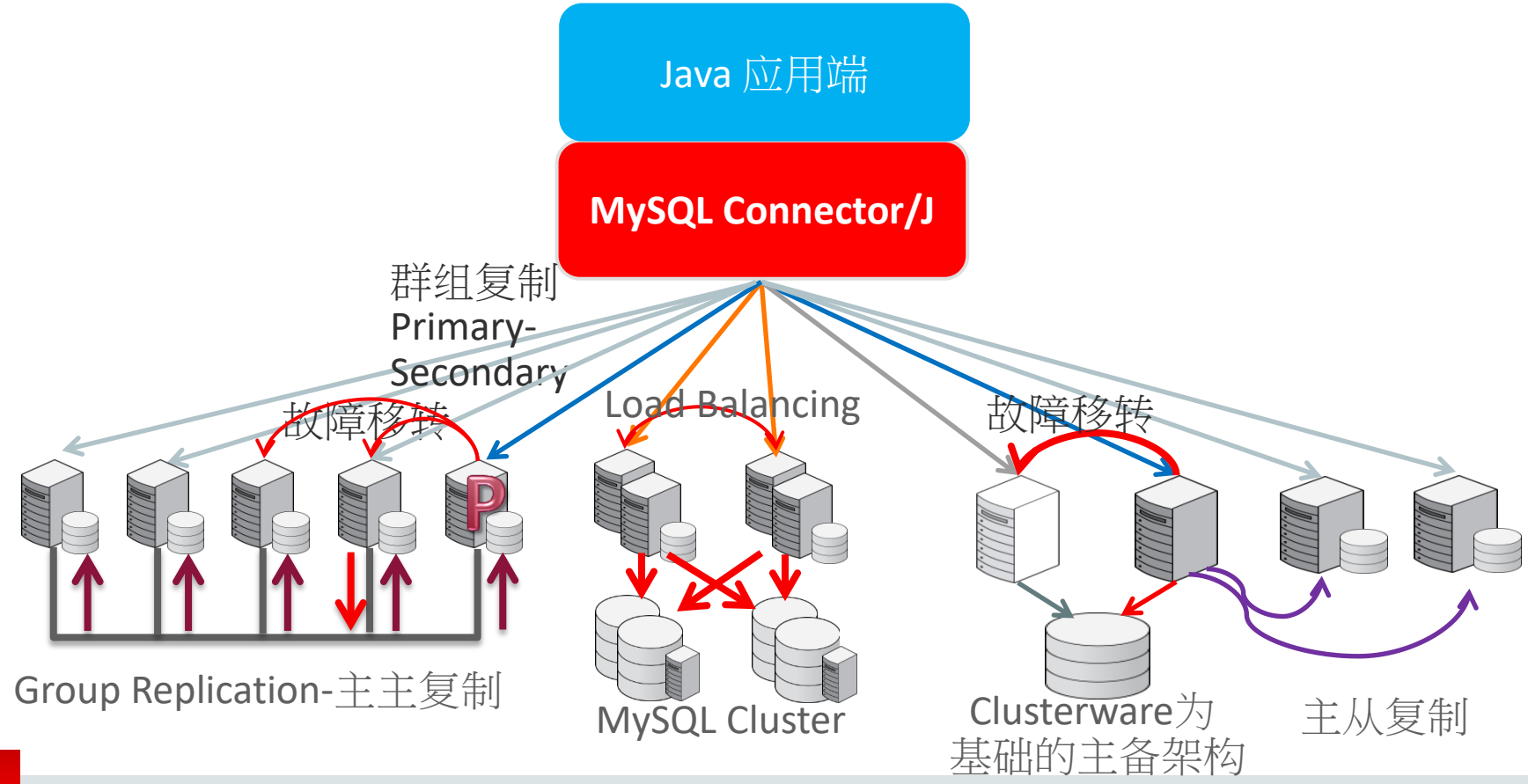
# MySQL HA With Connector/J

## - 支援负载均衡,故障移转



# MySQL HA

## - 支援负载均衡,故障移转



# 故障移转

- JDBC URL format，第一个主机为master，第二个以后为backup:

```
import com.mysql.jdbc.ReplicationDriver;
```

```
jdbc:mysql://[primary-host][:port],[secondary-host1][:port][,[secondary-
  host2][:port]]...[/[database]][?propertyName1=propertyValue1[&propertyName2=pro
  pertyValue2]...]
```

- Connection的faileover属性
  - failOverReadOnly : 为true时，faileover后设Connection.setReadOnly(false)；第二个host仍为read-only mode
  - secondsBeforeRetryMaster
  - queriesBeforeRetryMaster
  - retriesAllDown
  - autoReconnect
  - autoReconnectForPools

# Load Balancing

```
String URL = "jdbc:mysql:loadbalance://" +"localhost:3306,localhost:3310/test?" +
    "loadBalanceConnectionGroup=first&loadBalanceEnableJMX=true";
Class.forName("com.MySQL.jdbc.Driver");
Connection conn DriverManager.getConnection(URL, "root", “secret");

conn.setAutoCommit(false);

Statement stmt = conn.createStatement();

stmt.executeQuery("SELECT SLEEP(1) /* Connection: " + conn + ", transaction: " + trans + " */");

conn.commit();
```

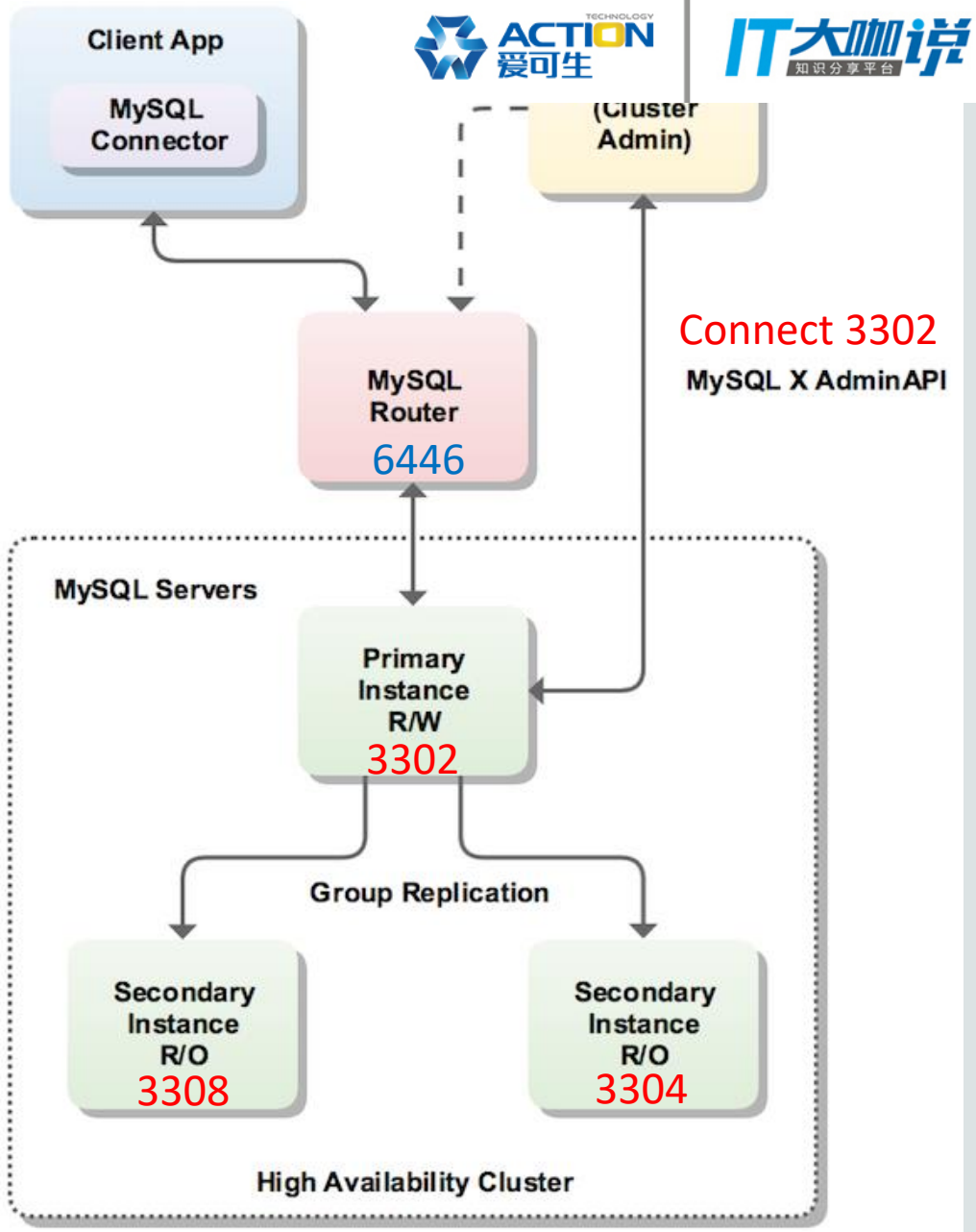
# MySQL 是什么?

- 各组件独立,没有共用
- 运作自动化,不需人工介入
- 故障移转对应用程式透通的

# 演示 MySQL最新高可用方案 - InnoDB Cluster

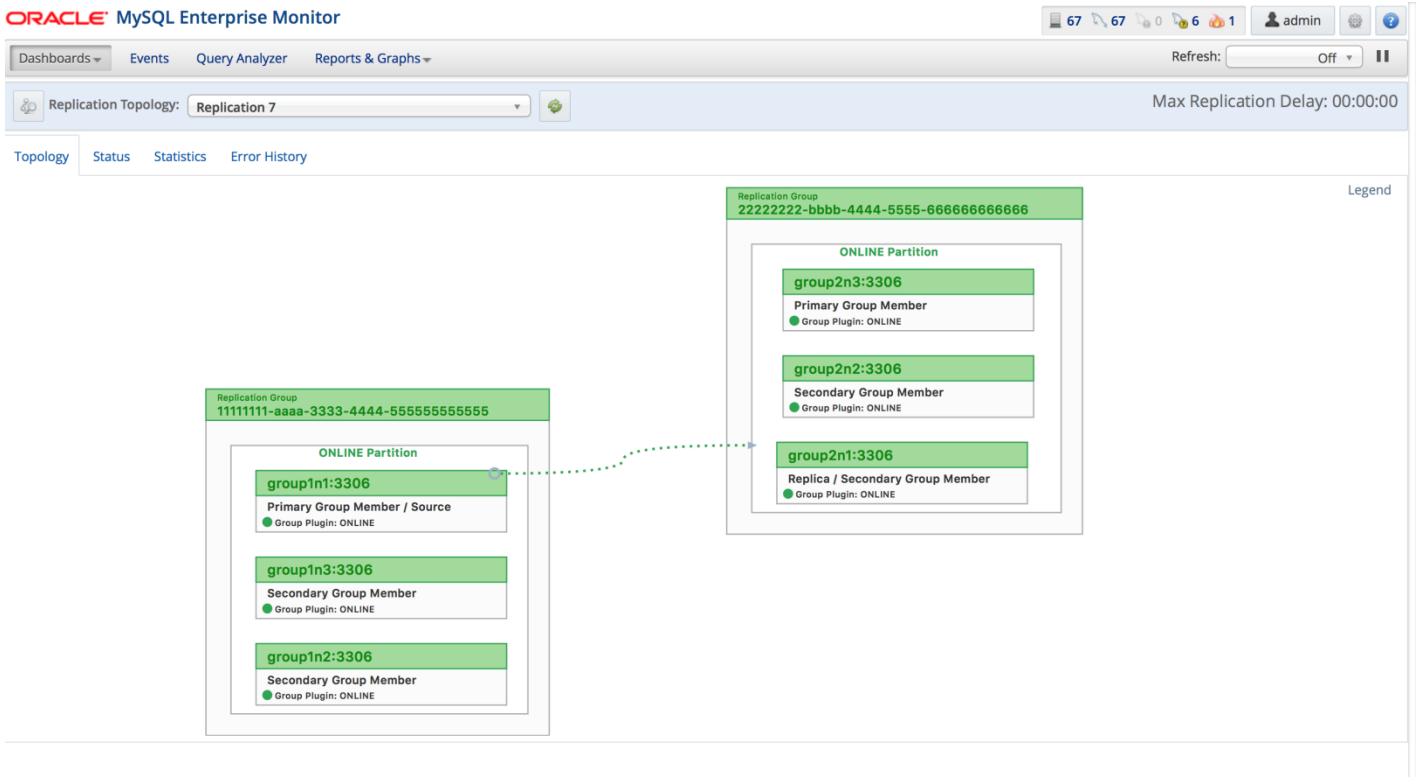
# Demo architecture – InnoDB Cluster

- IP of the guest OS is 192.168.56.101
- OS level username: ubuntu, password:Welcome1
- Passwords for router and all root account of member instances are Welcome1
- Run recreateInnoDBCluster.sh to bootstrap all the member instances and MySQL Router
- Demoreadme shows the steps of conflict resolution by group replication and fail-over by InnoDB Cluster



# MySQL Enterprise Monitor

- Out-of-box supports Group Replication / InnoDB clusters
  - Topology views
  - Detailed metrics and graphs
  - Best Practice advice



谢谢您

# 请您指教