极客大学

极客大学 Java 进阶训练营
第 14 课
超越分库分表-高可用与读写分离

KimmKing

Apache Dubbo/ShardingSphere PMC

个人介绍

极客大学

Apache Dubbo/ShardingSphere PMC

前某集团高级技术总监/阿里架构师/某银行北京研发中心负责人

阿里云 MVP、腾讯 TVP、TGO 会员

10 多年研发管理和架构经验

熟悉海量并发低延迟交易系统的设计实现

极客大学

目录

极客大学

1. 从单机到集群

单机 MySQL 数据库的几个问题

随着数据量的增大，读写并发的增加，系统可用性要求的提升，单机 MySQL 面临：

1、容量有限，难以扩容

2、读写压力，QPS 过大，特别是分析类需求会影响到业务事务

3、可用性不足，宕机问题

还有没有其他问题？

单机 MySQL 的技术演进

极客大学

读写压力	多机集群	主从复制
高可用性	故障转移	MHA/MGR/Orchestrator
容量问题	数据库拆分	分库分表
一致性问题	分布式事务	XA/柔性事务

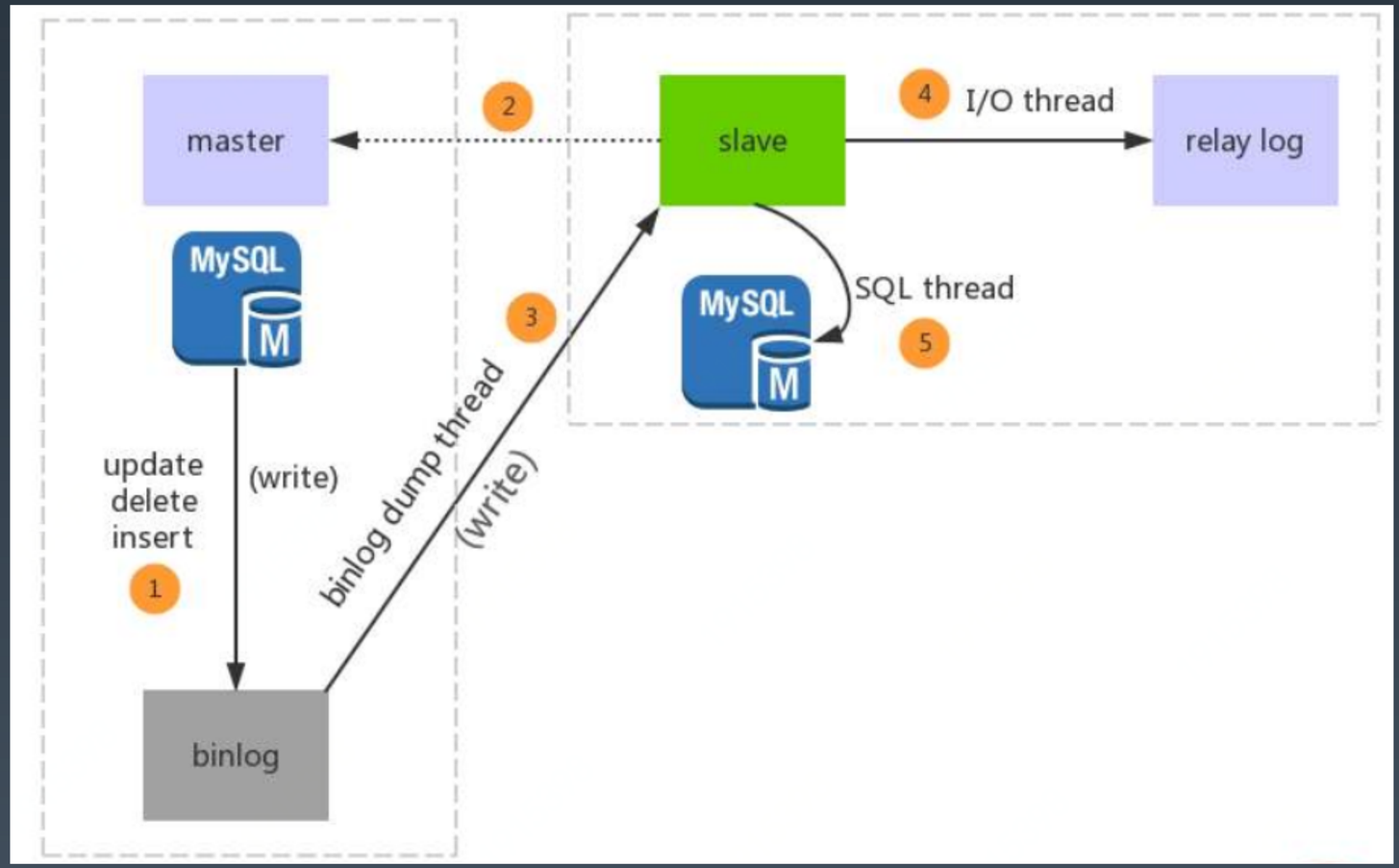
2. MySQL 主从复制

主从复制原理

核心是

1、主库写 binlog

2、从库 relay log



2000年，MySQL 3.23.15版本引入了复制

2002年，MySQL 4.0.2版本分离 IO 和 SQL 线程，引入了 relay log

2010年，MySQL 5.5版本引入半同步复制

2016年，MySQL 在5.7.17中引入 InnoDB Group Replication

主从复制原理

Binlog 格式

- ROW
- Statement
- Mixed

```
SET TIMESTAMP=1504417835/*!*/;
BEGIN
/*!*/;
# at 1678
#170903 13:50:35 server id 2  end_log_pos 1780 CRC32 0x1bb72319       Query   thread_id=5exec_time=0     error_code=0
SET TIMESTAMP=1504417835/*!*/;
insert into t1 values(999)
/*!*/;
# at 1780
#170903 13:50:35 server id 2  end_log_pos 1811 CRC32 0x558e0368       Xid = 37
COMMIT/*!*/;
```

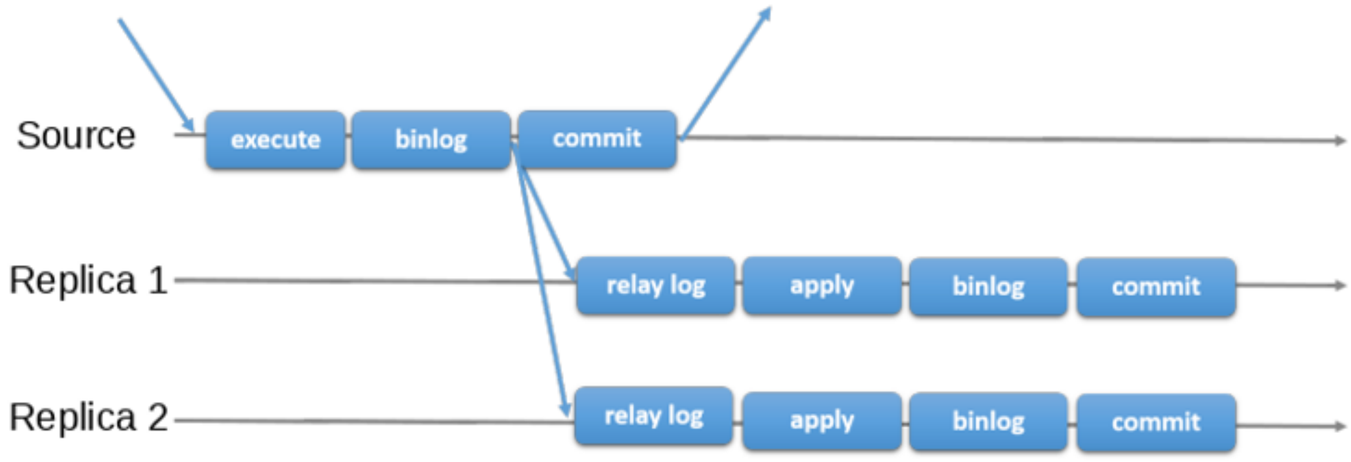
```
#mysqlbinlog  -vv  mysql-bin.000005

SET TIMESTAMP=1504409214/*!*/;      1.开始事物的时间;
BEGIN
/*!*/;
# at 547           2.sql-event起点 ，改点为事件的起点，是以547字节开始。
#170903 11:26:54 server id 2  end_log_pos 593 CRC32 0x4b8ca1d9  Table_map: `test1`.`t1` mapped to number 219      3.sqlevent 发生的时间点，是事件发生的时间。
# at 593
#170903 11:26:54 server id 2  end_log_pos 633 CRC32 0xacc956c2  Write_rows: table id 219 flags: STMT_END_F    4.server-Id，为master 的server-Id；5.sql-event终点及花费时间，错误码。
BINLOG '
fnarWRMCAAAALgAAAFECAAAAANsAAAAAAAEABXRlc3QxAAJ0MQABAwAB2aGMSw==
fnarWR4CAAAAKAAAAHkCAAAAANsAAAAAAAEAAgAB//7nAwAAwlbJrA==
'/*!*/;

### INSERT INTO `test1`.`t1`
### SET
###   @1=999 /* INT meta=0 nullable=1 is_null=0 */
# at 633
#170903 11:26:54 server id 2  end_log_pos 664 CRC32 0x8100a8d5  Xid = 21
COMMIT/*!*/;
# at 664
#170903 11:28:34 server id 2  end_log_pos 729 CRC32 0x53c4213c  Anonymous_GTID  last_committed=2     sequence_number=3       rbr_only=yes
/*!50718 SET TRANSACTION ISOLATION LEVEL READ COMMITTED*//*!*/;
SET @@SESSION.GTID_NEXT= 'ANONYMOUS'/*!*/;
# at 729
#170903 11:28:34 server id 2  end_log_pos 802 CRC32 0x25df25fb  Query   thread_id=3     exec_time=0  error_code=0
```

主从复制原理

异步复制：传统主从复制--2000年，MySQL 3.23.15版本引入了 Replication

Primary-Secondary Replication（传统主从复制）

异步复制：网络或机器故障，会造成数据不一致



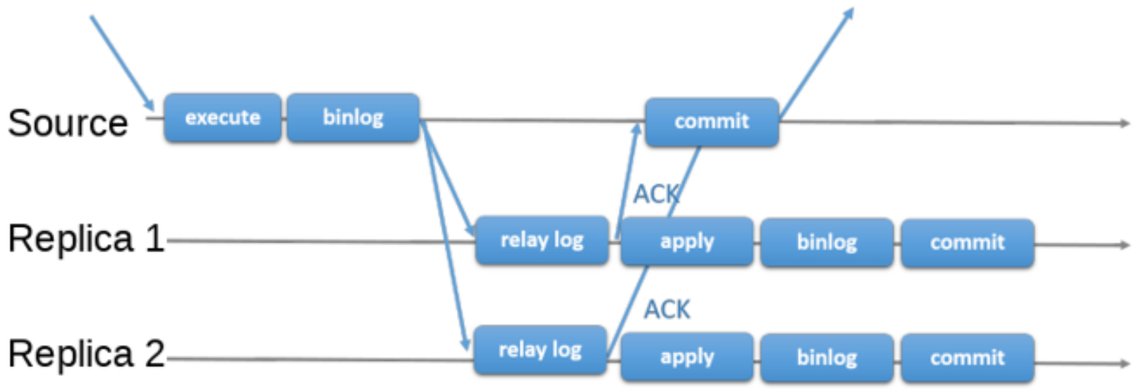
MySQL Asynchronous Replication

主从复制原理

半同步复制：需要启用插件

Primary-Secondary Replication（传统主从复制）

半同步复制：保证Source和Replica最终一致性



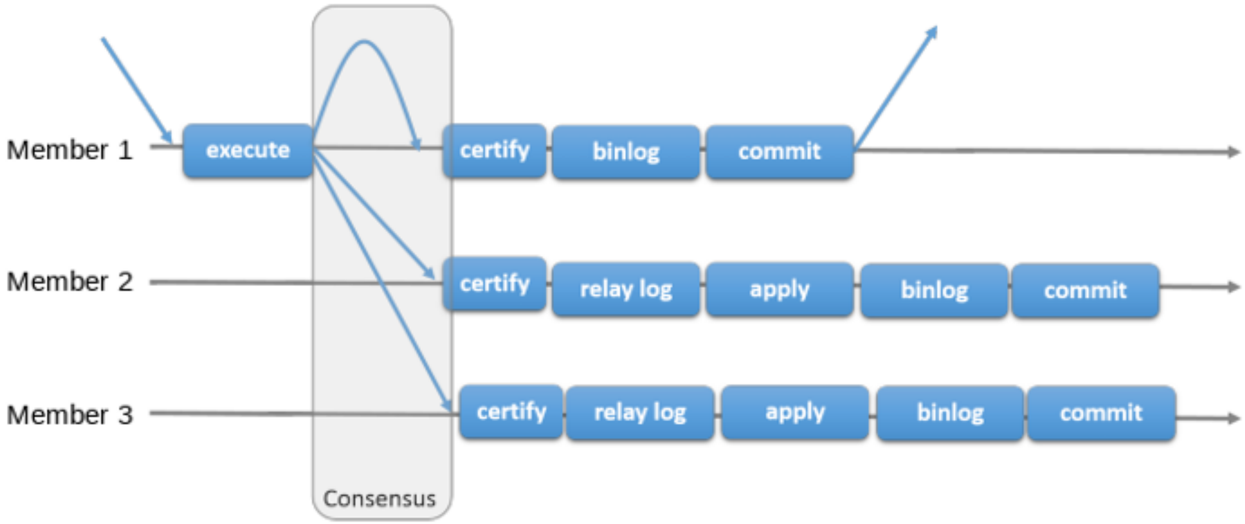
MySQL Semisynchronous Replication

主从复制原理

组复制：

MySQL Group Replication (MGR)

组复制：基于分布式Paxos协议实现组复制，保证数据一致性



主从复制演示

1、本地启动两个 MySQL

2、注意配置文件（思考几种安装、启动方式）

4、演示数据复制操作，创建表和写入、修改数据

还有没有其他问题？

主从复制的局限性

1、主从延迟问题

2、应用侧需要配合读写分离框架

3、不解决高可用问题

3. MySQL 读写分离

主从复制在业务系统里的应用

借助于主从复制，我们现在有了多个 MySQL 服务器示例。

如果借助这个新的集群，改进我们的业务系统数据处理能力？

== > 配置多个数据源，实现读写分离

读写分离-动态切换数据源版本1.0

1、基于 Spring/Spring Boot，配置多个数据源(例如2个，master 和 slave)

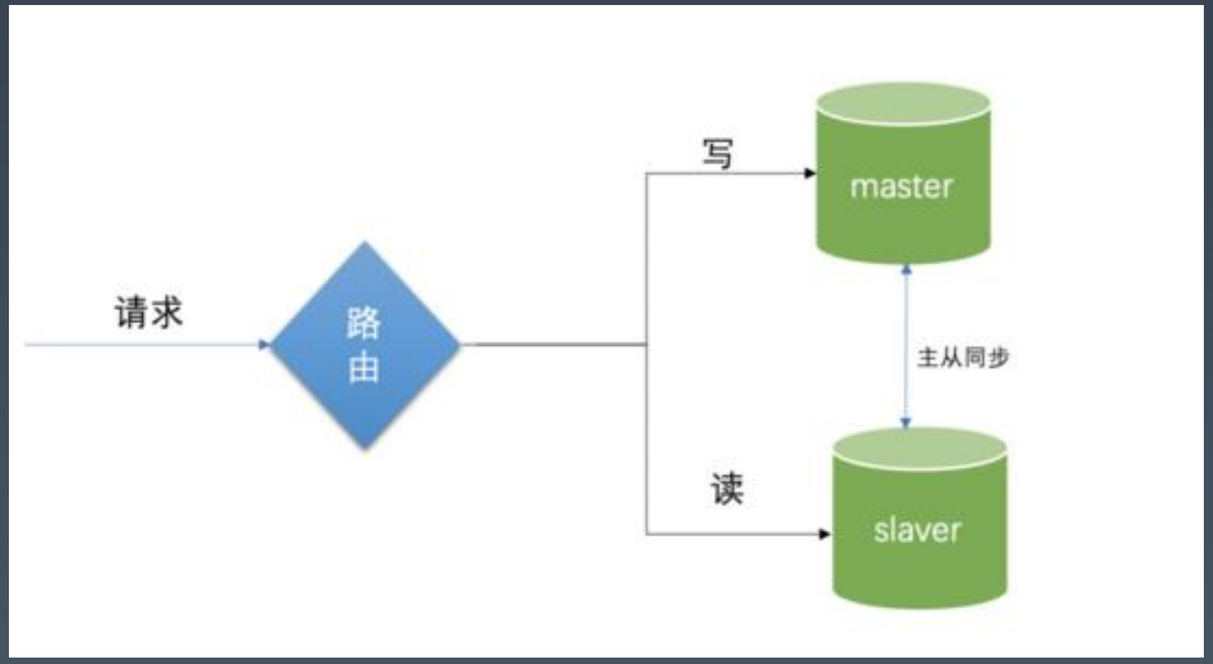
2、根据具体的 Service 方法是否会操作数据，注入不同的数据源,1.0版本

3、改进一下1.1：基于操作 AbstractRoutingDataSource 和自定义注解 readOnly 之类的，简化自动切换数据源

4、改进二下1.2：支持配置多个从库；

5、改进三下1.3：支持多个从库的负载均衡。

今天的作业之一。



读写分离-数据库框架版本2.0

1、分析前一版本“动态切换数据源”有什么问题？

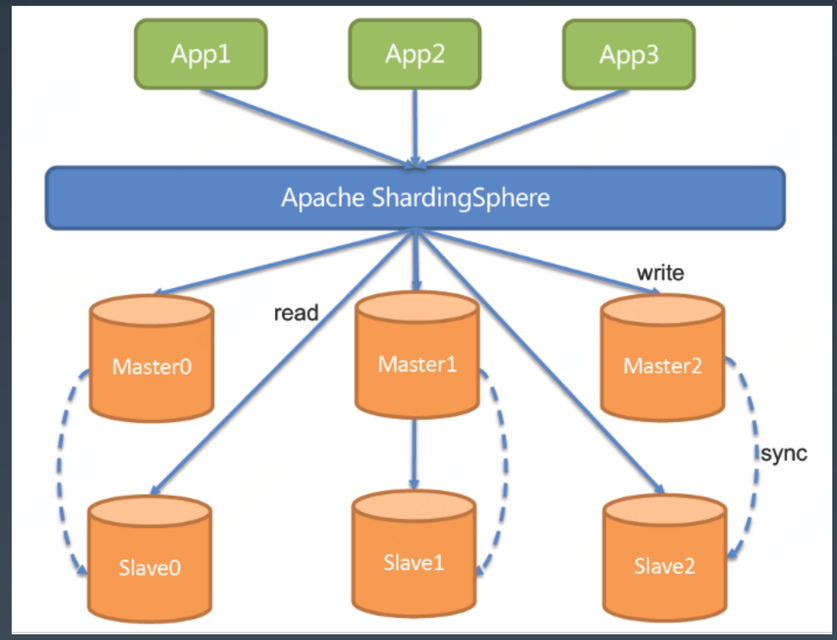
1）侵入性还是较强

2）降低侵入性会导致”写完读”不一致问题

2、改进方式，ShardingSphere-jdbc 的 Master-Slave 功能

1）SQL 解析和事务管理，自动实现读写分离

2）解决”写完读”不一致的问题



今天作业之一：使用 ShardingSphere-jdbc 5.0.0-alpha 实现读写分离配置。

读写分离-数据库中间件版本3.0

1、分析前一版本“框架版本”有什么问题？

1）对业务系统还是有侵入

2）对已存在的旧系统改造不友好

2、改进方式，MyCat/ShardingSphere-Proxy 的 Master-Slave 功能

1）需要部署一个中间件，规则配置在中间件

2）模拟一个 MySQL 服务器，对业务系统无侵入

今天作业之一：使用 ShardingSphere-proxy 5.0.0-alpha 实现读写分离配置。

4. MySQL 高可用

为什么要高可用

1、读写分离，提升读的处理能力

2、故障转移，提供 failover 能力

加上业务侧连接池的心跳重试，实现断线重连，业务不间断，降低 RTO 和 RPO。

高可用定义

高可用意味着，更少的不可服务时间。一般用SLA/SLO衡量。

1年 = 365天 = 8760小时

99 = 8760 * 1% = 8760 * 0.01 = 87.6小时

99.9 = 8760 * 0.1% = 8760 * 0.001 = 8.76小时

99.99 = 8760 * 0.0001 = 0.876小时 = 0.876 * 60 = 52.6分钟

99.999 = 8760 * 0.00001 = 0.0876小时 = 0.0876 * 60 = 5.26分钟

后面的分布式课程讲稳定性，注意关系和区别。

你维护的系统有几个9？99.95%算是几个9？

为什么要高可用

什么是 failover，故障转移，灾难恢复

容灾：热备与冷备

对于主从来说，简单讲就是主挂了，某一个从，变成主，

整个集群来看，正常对外提供服务

常见的一些策略：

1、多个实例不在一个主机/机架上

2、跨机房和可用区部署

3、两地三中心容灾高可用方案

MySQL 高可用0：主从手动切换

如果主节点挂掉，将某个从改成主；
重新配置其他从节点。
修改应用数据源配置。

>> 有什么问题？

1. 可能数据不一致。
2. 需要人工干预。
3. 代码和配置的侵入性。

MySQL 高可用1：主从手动切换

用 LVS+Keepalived 实现多个节点的探活+请求路由。

配置 VIP 或 DNS 实现配置不变更。

>> 有什么问题？

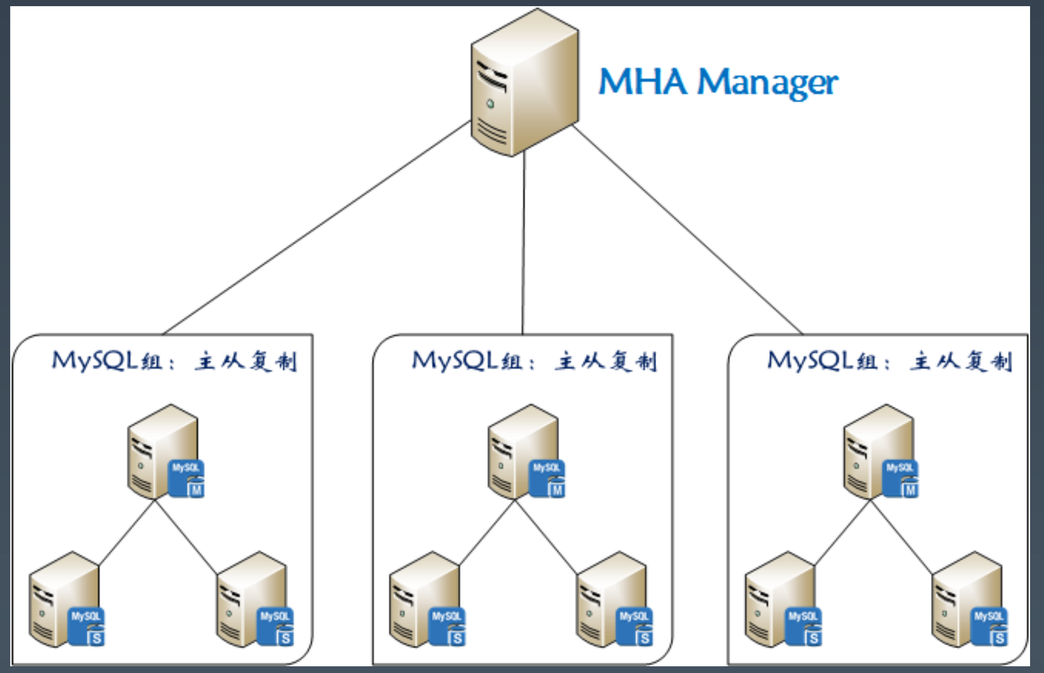
1. 手工处理主从切换
2. 大量的配置和脚本定义

MySQL 高可用2：MHA

MHA（Master High Availability）目前在 MySQL 高可用方面是一个相对成熟的解决方案，它由日本 DeNA 公司的 youshimaton（现就职于 Facebook 公司）开发，是一套优秀的作为 MySQL 高可用性环境下故障切换和主从提升的高可用软件。

基于 Perl 语言开发，一般能在30s内实现主从切换。

切换时，直接通过 SSH 复制主节点的日志。



>> 有什么问题？

1. 需要配置 SSH 信息
2. 至少3台

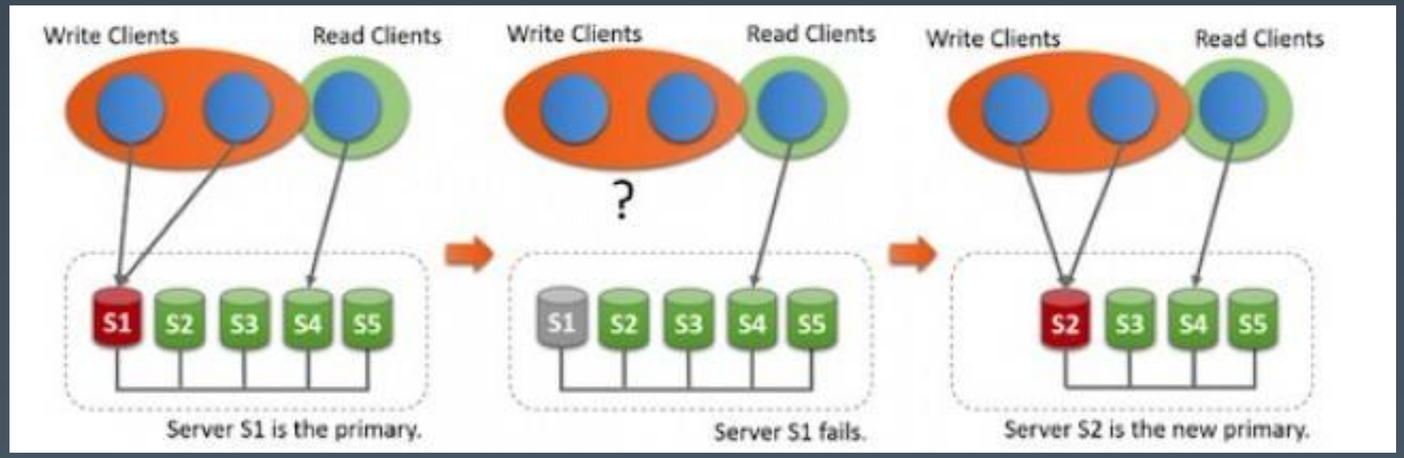
MySQL 高可用3：MGR *

如果主节点挂掉，将自动选择某个从改成主；

无需人工干预，基于组复制，保证数据一致性。

>> 有什么问题？

1. 外部获得状态变更需要读取数据库。
2. 外部需要使用 LVS/VIP 配置。



MySQL 高可用3：MGR *

极客大学

MGR 特点

MySQL Group Replication (MGR)

MGR的特点

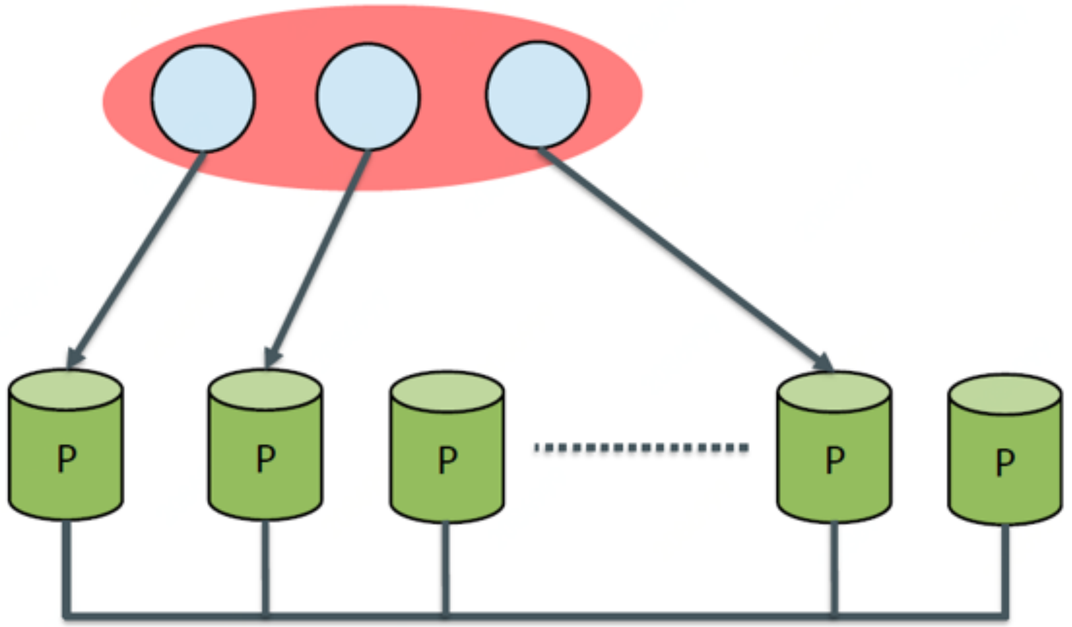
1. 高一致性：基于分布式Paxos协议实现组复制，保证数据一致性；
2. 高容错性：自动检测机制，只要不是大多数节点都宕机就可以继续工作，内置防脑裂保护机制；
3. 高扩展性：节点的增加与移除会自动更新组成员信息，新节点加入后，自动从其他节点同步增量数据，直到与其他节点数据一致；
4. 高灵活性：提供单主模式和多主模式，单主模式在主库宕机后能够自动选主，所有写入都在主节点进行，多主模式支持多节点写入。

MySQL 高可用3：MGR *

适用场景

弹性复制

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.

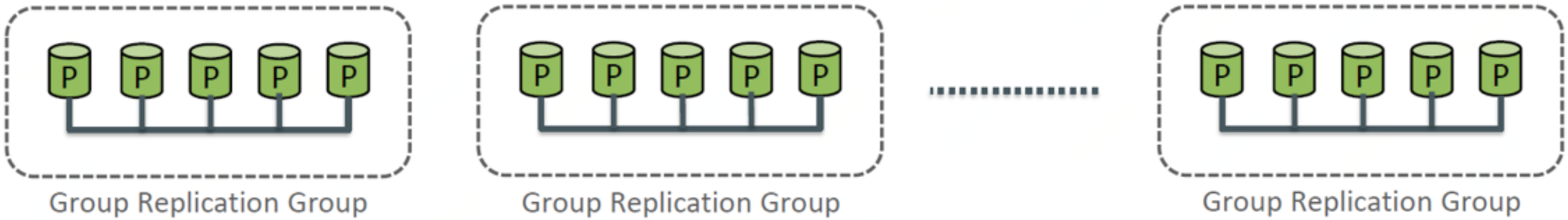


MySQL 高可用3：MGR *

适用场景

高可用分片

Shardingis 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.



MySQL 高可用3：MGR *

MGR 演示。

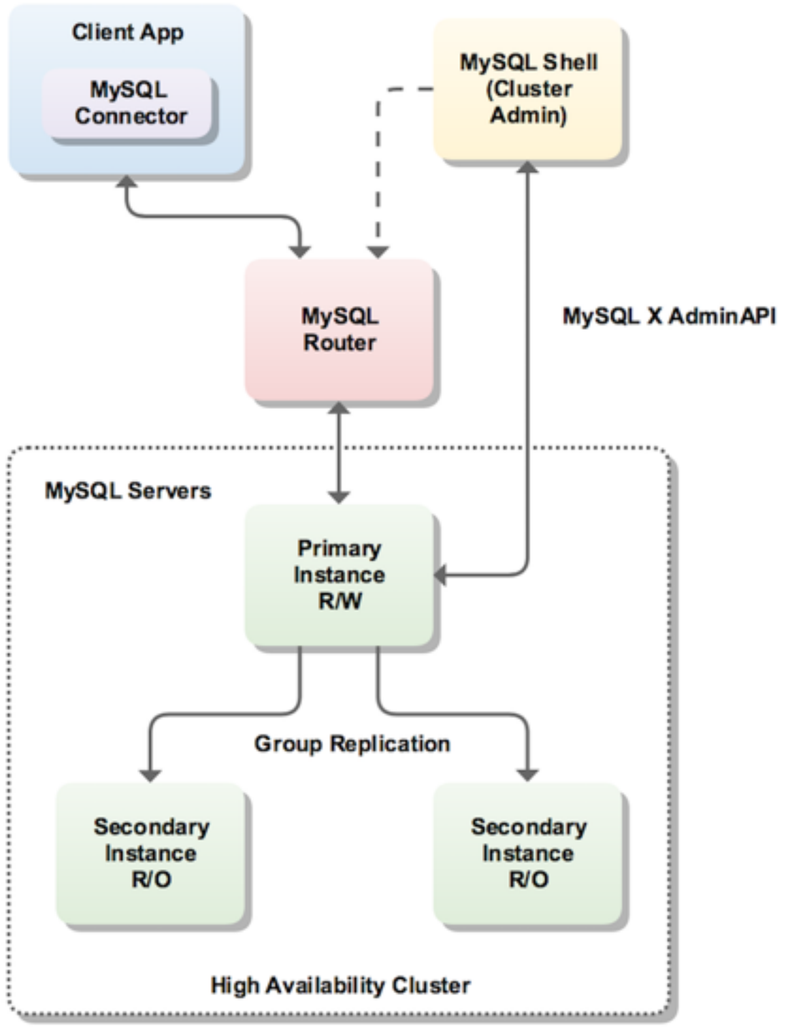
MySQL 高可用4：MySQL Cluster

MySQL InnoDB Cluster

完整的数据库层高可用解决方案

MySQL InnoDB Cluster是一个高可用的框架，它由下面这几个组件构成：

1. **MySQL Group Replication**：提供DB的扩展、自动故障转移
2. **MySQL Router**：轻量级中间件，提供应用程序连接目标的故障转移
3. **MySQL Shell**：新的MySQL客户端，多种接口模式。可以设置群组复制及Router



MySQL 高可用4：MySQL Cluster

MySQL InnoDB Cluster

MySQL Shell

MySQL Shell是MySQL团队打造的一个统一的客户端，它可以对MySQL执行数据操作和管理。它支持通过JavaScript，Python，SQL对关系型数据模式和文档型数据模式进行操作。使用它可以轻松配置管理 InnoDB Cluster。

MySQL Router

MySQL Router是一个轻量级的中间件，可以提供负载均衡和应用连接的故障转移。它是MySQL团队为MGR量身打造的，通过使用Router 和 Shell，用户可以利用MGR实现完整的数据库层的解决方案。如果您在使用MGR，请一定配合使用Router和Shell，您可以理解为它们是为MGR而生的，会配合MySQL的开发路线图发展的工具。

MySQL 高可用5：Orchestrator

如果主节点挂掉，将某个从改成主；

orchestrator

一款MySQL高可用和复制拓扑管理工具，支持复制拓扑结构的调整，自动故障转移和手动主从切换等。后端数据库用MySQL或SQLite存储元数据，并提供Web界面展示MySQL复制的拓扑关系及状态，通过Web可更改MySQL实例的复制关系和部分配置信息，同时也提供命令行和API接口，方便运维管理。

特点：

1. 自动发现MySQL的复制拓扑，并且在web上展示；
2. 重构复制关系，可以在web进行拖图来进行复制关系变更；
3. 检测主异常，并可以自动或手动恢复，通过Hooks进行自定义脚本；
4. 支持命令行和web界面管理复制。

MySQL 高可用5：Orchestrator

基于 Go 语言开发，实现了中间件本身的高可用(?!)

优势：

能直接在 UI 界面

拖拽改变主从关系

orchestrator

两种部署方式

orchestrator/raft:

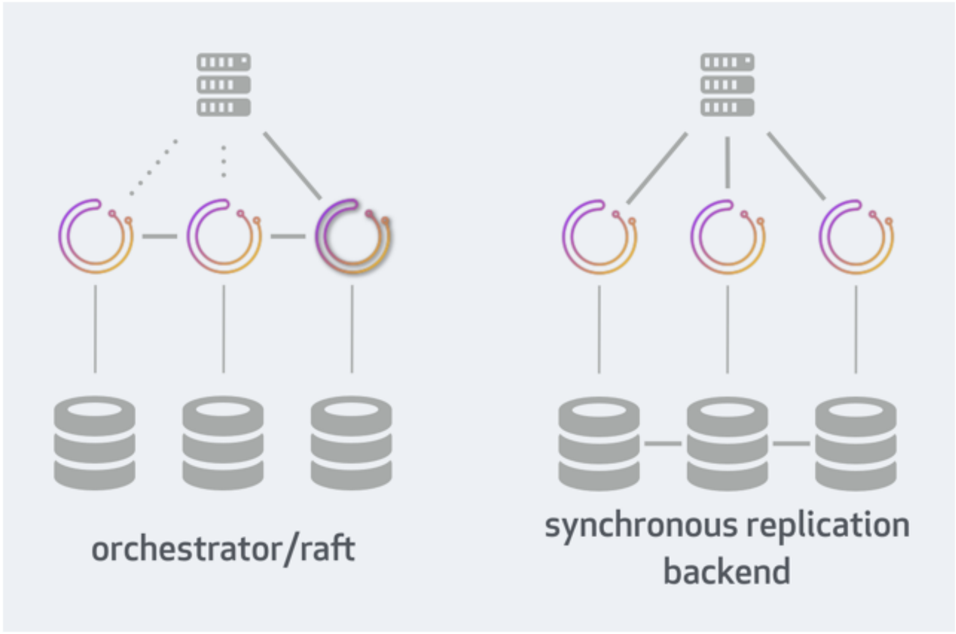
1. 数据一致性由orchestrator的raft协议保证
2. 数据库之间不通信

orchestrator/[galera | xtradb cluster | innodb cluster]:

1. 数据一致性由数据库集群保证
2. 数据库结点之间通信

如果不部署client

1. 使用HTTP (/api/leader-check) 查询并路由到主节点



极客大学

5.总结回顾与作业实践

第 14 节课总结回顾

极客大学

单机数据库技术演进

MySQL 主从复制

MySQL 读写分离

MySQL 高可用

第14节课作业实践

1、（选做）配置一遍异步复制，半同步复制、组复制。

2、（必做）读写分离-动态切换数据源版本1.0

3、（必做）读写分离-数据库框架版本2.0

4、（选做）读写分离-数据库中间件版本3.0

5、（选做）配置 MHA，模拟 master 宕机

6、（选做）配置 MGR，模拟 master 宕机

7、（选做）配置 Orchestrator，模拟 master 宕机，演练 UI 调整拓扑结构

THANKS!