



Oracle技术嘉年华

Oracle Technology Carnival 2015

稳健●高效●云端 - 数据技术最佳实践

讲师：宋利兵

主题：MySQL-5.7GA中的复制和集群管理



Safe Harbor Statement

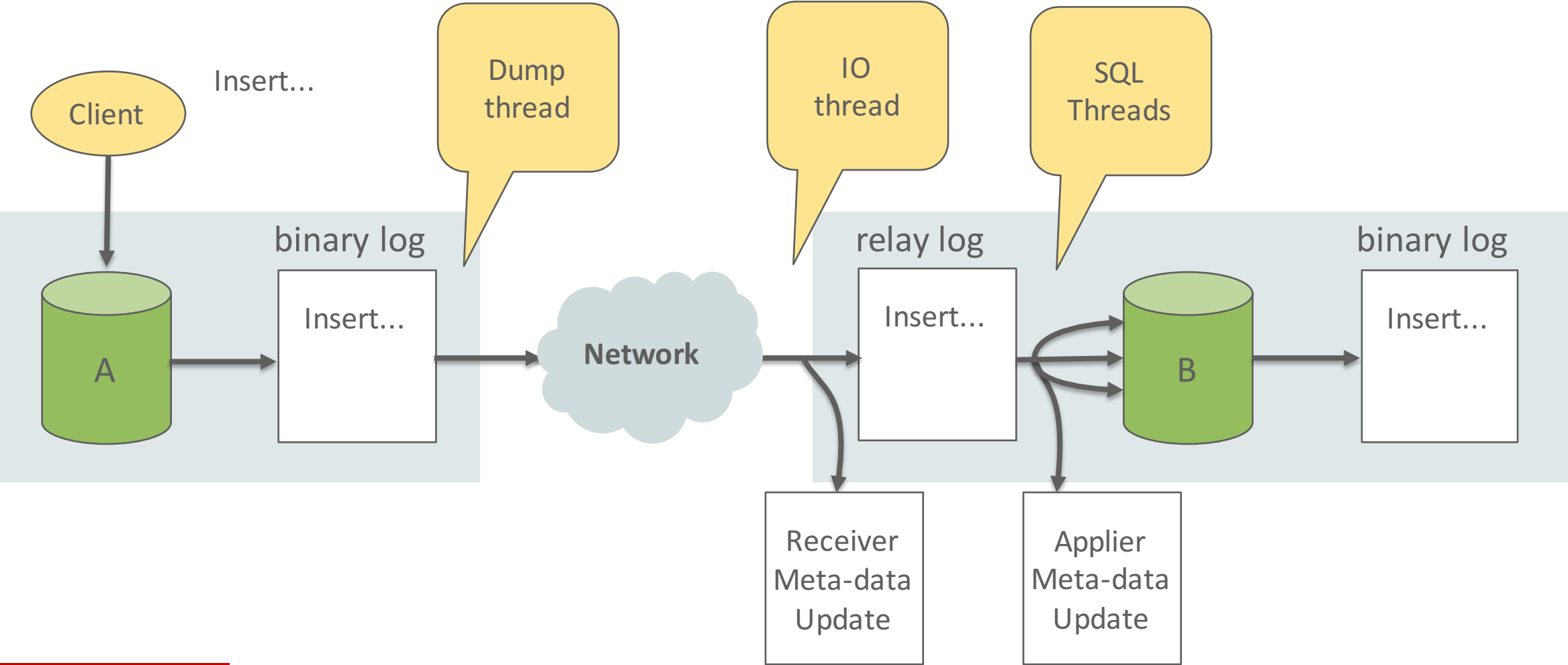
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

目录

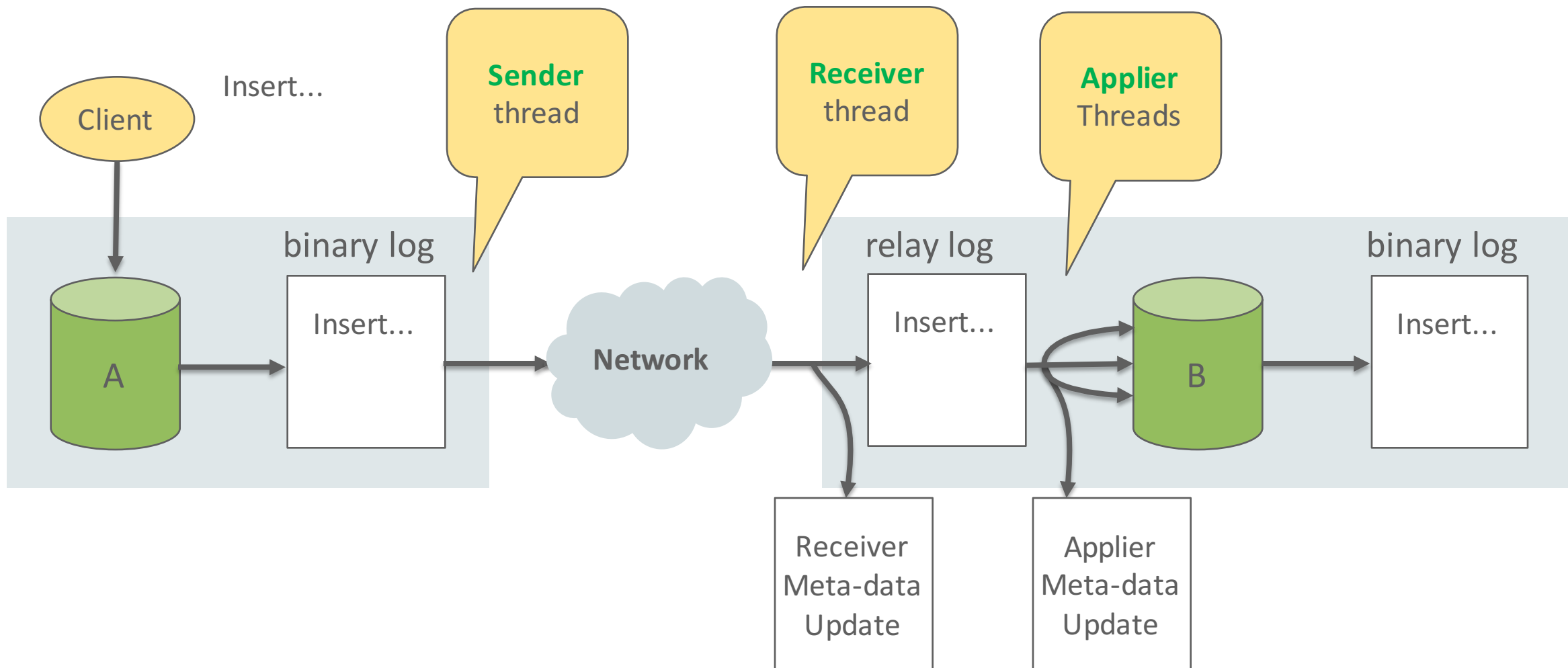
- 1 ➤ MySQL 复制基础
- 2 ➤ MySQL 5.7 GA里的复制新功能
- 3 ➤ Group Replication和MySQL集群管理
- 4 ➤ 复制的开发蓝图

1 MySQL 复制基础

复制基础: 复制的框架



复制基础: 复制的框架



复制基础: 复制的框架

- **Binary Log**

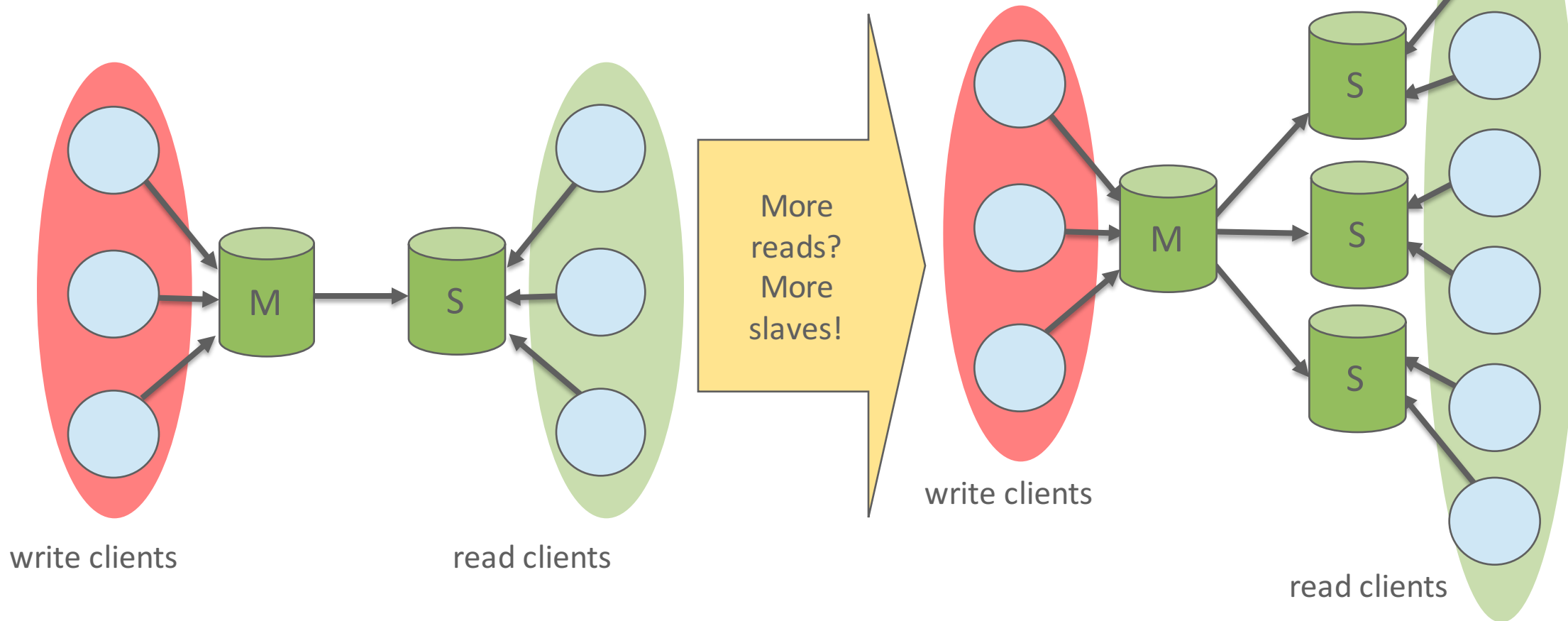
- 基于文件的逻辑日志，记录了Master上数据的变化.
- 语句格式(Statement Format)、行格式(Row Format)、混合模式(Mixed).
- 每个事务会被记录为一组Binlog Events.
- Control events: Rotate, Format Description, Gtid, ...



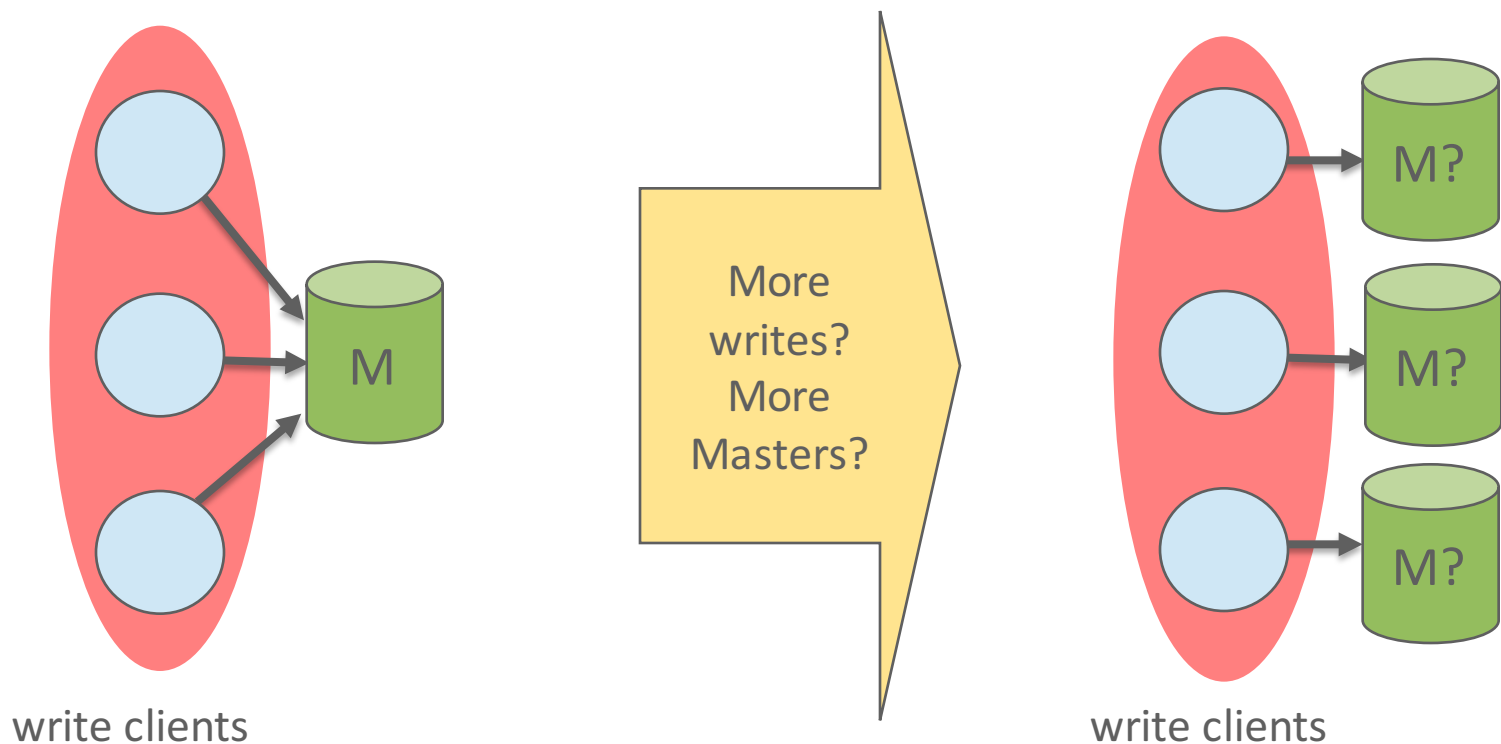
Layout of a
Binary Log File

复制基础: 复制的用途?

读性能的扩展(Read scale-out)



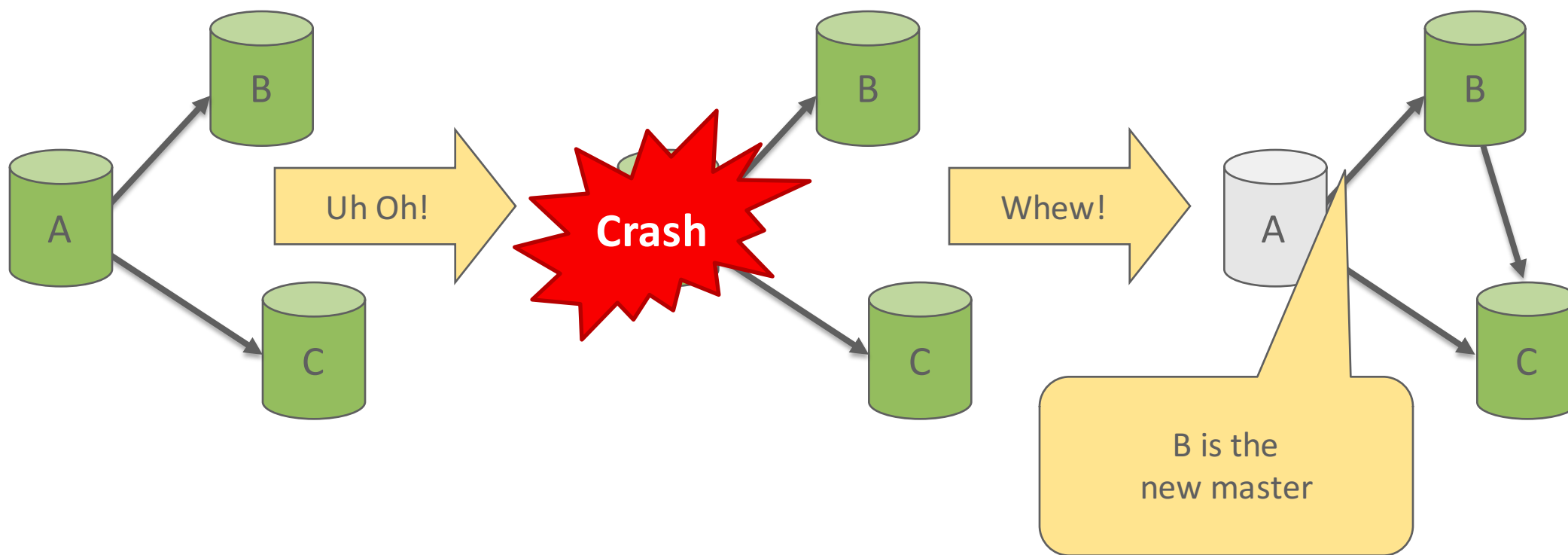
那么写性能的扩展 (**write** scale-out)呢?



MySQL Fabric 通过数据分片来提升MySQL的性能...

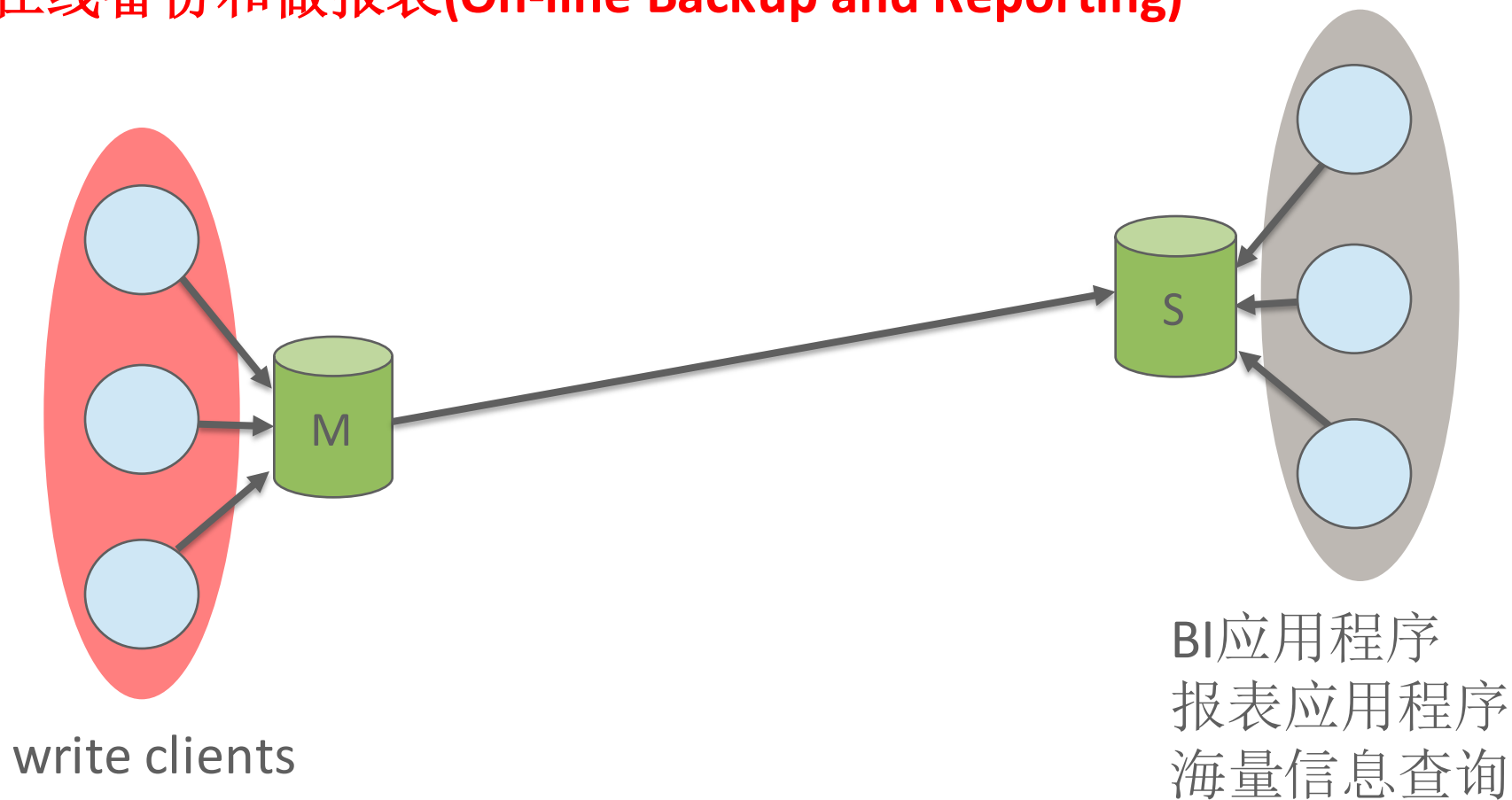
复制基础: 复制的用途?

冗余(Redundancy): 如果主服务器宕机, 可以用一个从服务器来做新的主服务器。



复制基础: 复制的用途?

在线备份和做报表(On-line Backup and Reporting)



复制基础: 复制的用途?

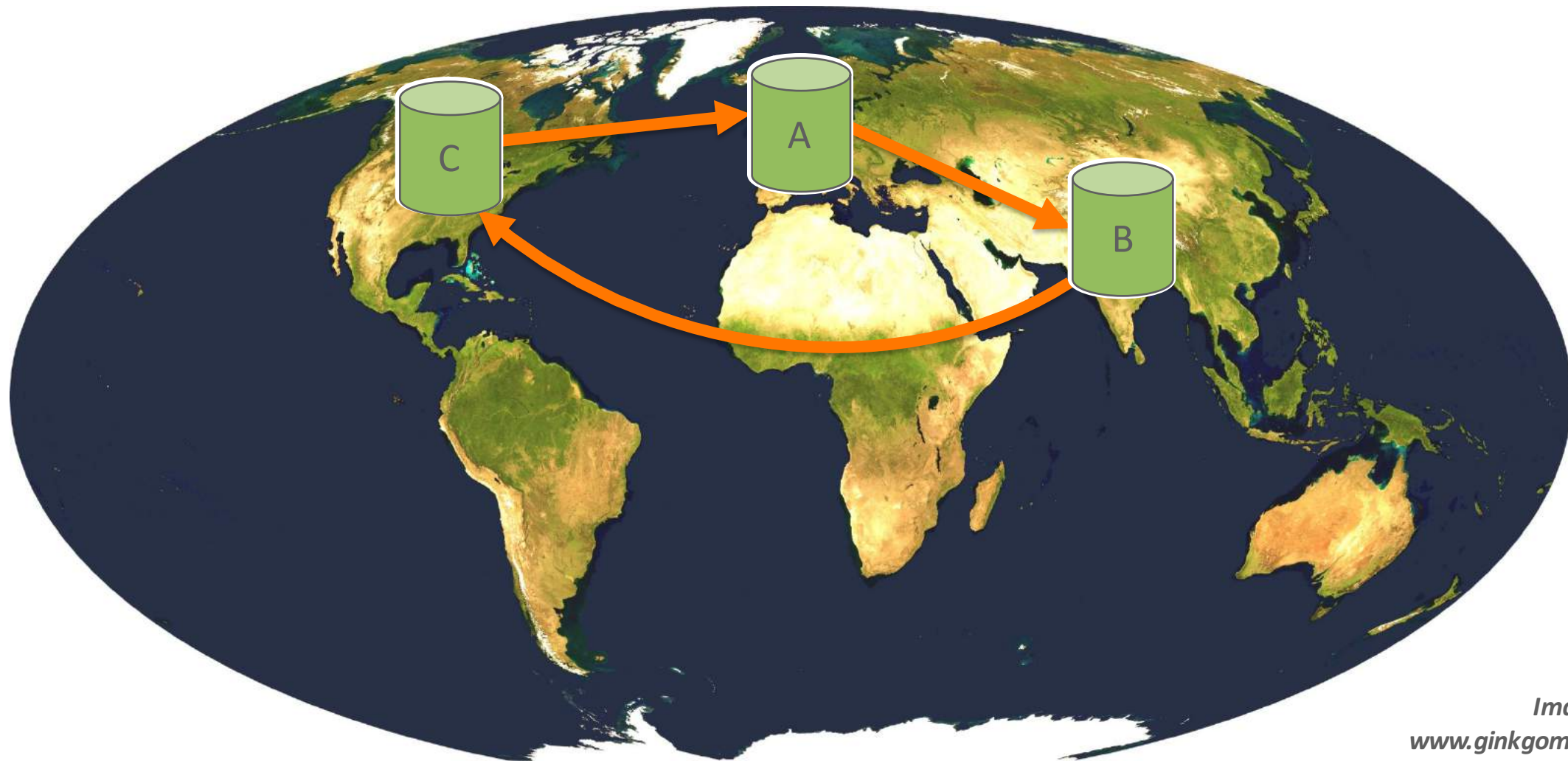


Image from
www.ginkgomaps.com

2 MySQL 5.7 GA里的复制新功能

2015年10月26日MySQL 5.7.9 GA啦

- 包含了**40 replication worklogs**
 - 29 个是关于现有复制的
 - 11 个是关于MySQL Group Replication.
- 有8个MySQL社区提交的patch被合并到MySQL 5.7.
- 复制功能有19个主要的改进
- 对复制的代码有**14处重构**

2 MySQL 5.7 GA里的复制新功能

2.1 GTID 新功能

2.2 多线程复制(Multi-Threaded Slave)新功能

2.3 多源复制(Multiple Source Replication)

2.4 Slave 可用性的改进

2.5 Master 性能的提升

2.6 半同步复制(Semi-synchronous Replication)的新功能

2.7 其他功能

2 MySQL 5.7 GA里的复制新功能

2.1 GTID 新功能

2.2 多线程复制(Multi-Threaded Slave)新功能

2.3 多源复制(Multiple Source Replication)

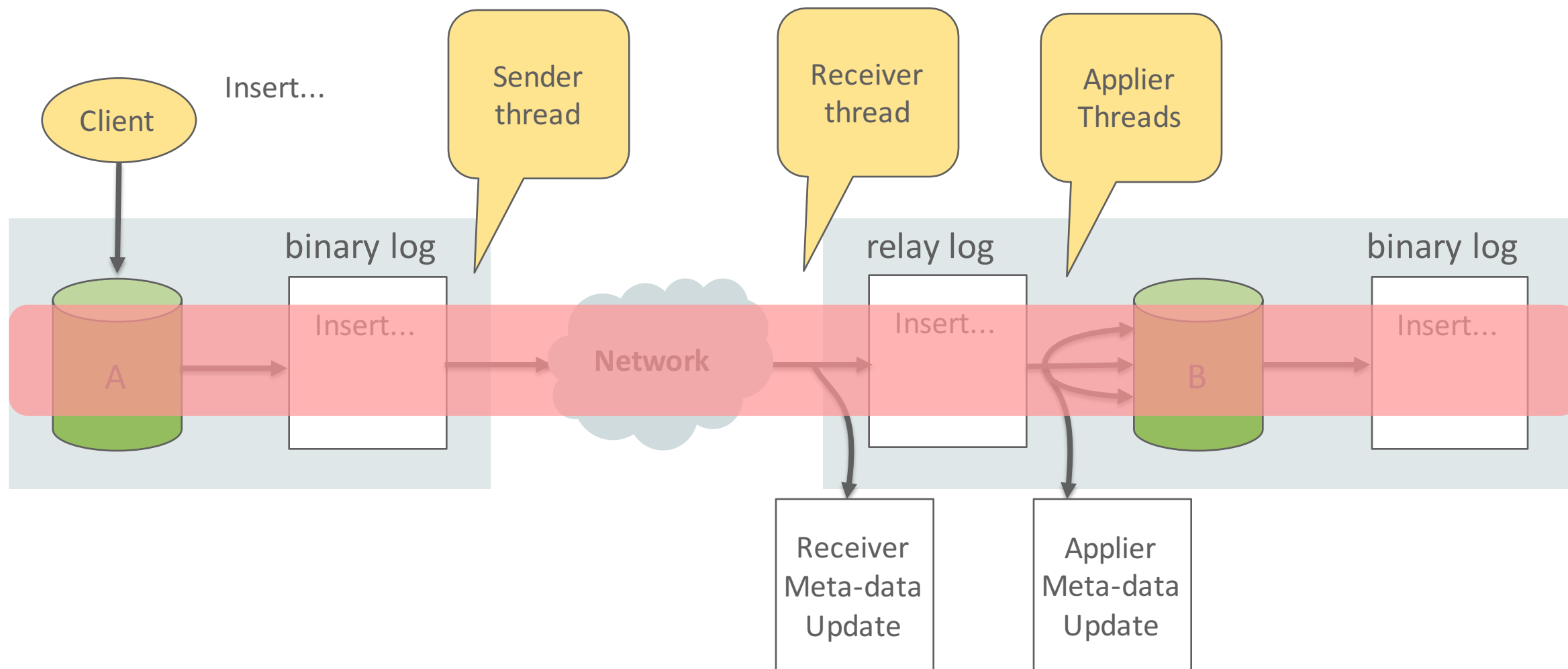
2.4 Slave 可用性的改进

2.5 Master 性能的提升

2.6 半同步复制(Semi-synchronous Replication)的新功能

2.7 其他功能

在线配置GTID功能



在线配置GTID功能

- 整个配置过程在线
 - 在打开或者关闭GTID功能的时候，整个复制集群仍然对外提供读和写的服务.
- 不需要重启MySQL服务器.
- 不需要改变复制拓扑结构.
 - 可以在任何结构的复制集群中在线启用GTID功能.

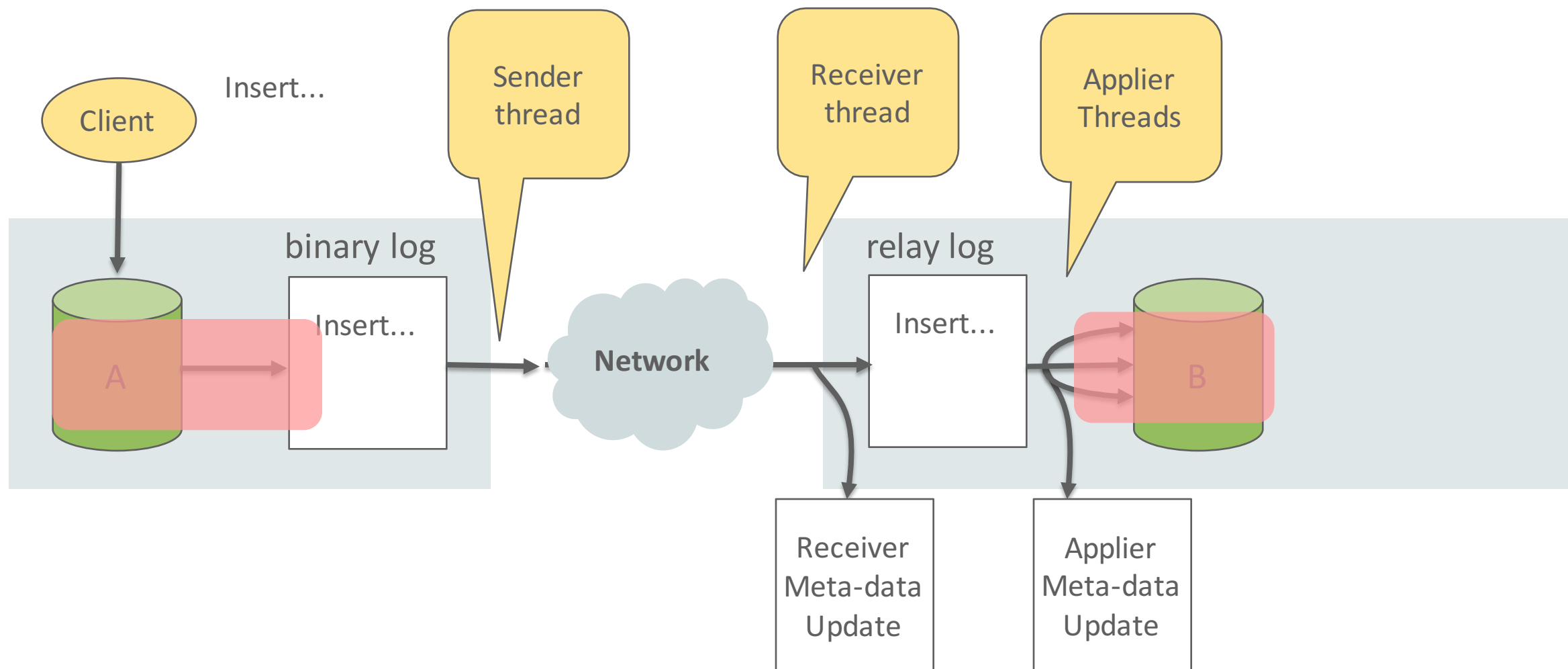
在线配置GTID功能

- 简化了的配置过程:

- 1) SET @@GLOBAL.GTID_MODE = OFF_PERMISSIVE; (在每一个服务器上设置)
- 2) SET @@GLOBAL.GTID_MODE = ON_PERMISSIVE; (在每一个服务器上设置)
- 3) 等一段时间, 让启用前记录的binlog events在所有的服务器上执行完毕。
- 4) SET @@GLOBAL.GTID_MODE = ON; (在每一个服务器上设置)

- 完整配置过程: <http://dev.mysql.com/doc/refman/5.7/en/replication-mode-change-online-enable-gtids.html>

存储GTID到系统表中



存储GTID到系统表中

应用场景

- Slaves 可以在不开启binlog的条件下，使用GTID功能
 - 有些Slave服务器永远不会被当maser来使用, 就不用开启binlog. 但这些服务器仍然可以使用GTID及AUTO_POSITION来简化管理。

存储GTID到系统表中 实现机制

- 在系统数据库mysql中创建了一张表gtid_executed

```
CREATE TABLE gtid_executed(  
    source_uuid CHAR(36) NOT NULL,  
    interval_start BIGINT NOT NULL,  
    interval_end BIGINT NOT NULL,  
    PRIMARY KEY(source_uuid, interval_start)  
);
```

- 事务的GTID在事务提交时插入到该表中去。此操作是该事务的一部分，和事务的其他操作整体保持原子性
- 压缩线程定期压缩该表记录
 - 新系统变量控制压缩周期: gtid_executed_compression_period.

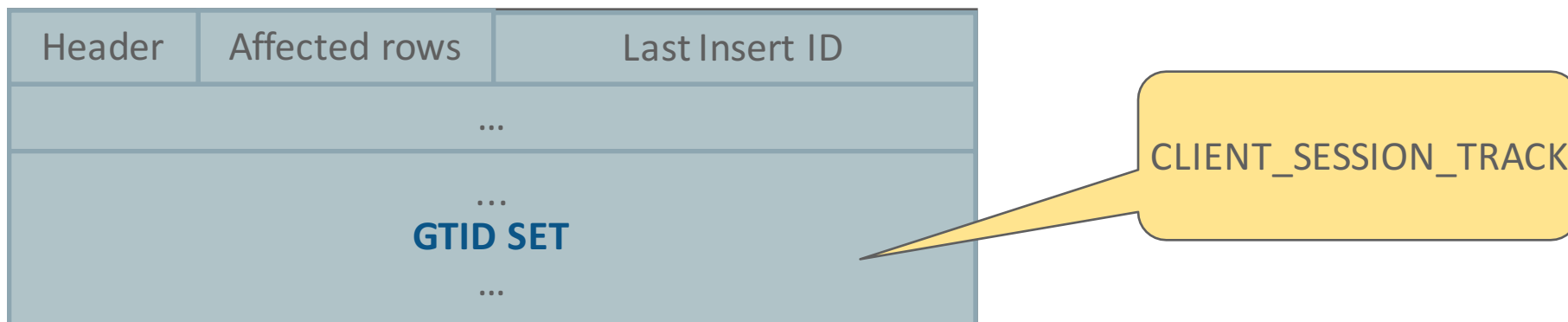
```
mysql> SET GLOBAL gtid_executed_compression_period= N; (N - 事务的数量)
```

存储GTID到系统表中 实现机制

- 当binlog启用时
 - 存储GTID到binlog.
 - 只在binlog文件切换时，将该binlog文件中包含的gtid集合写入表中
 - 性能比每事务写表高
- 当binlog未启用时
 - 每个事务提交时，存储GTID到表中
- 始终crash-safe.

追踪GTID

MySQL OK 包中的GTID



追踪GTID

控制OK包中的GTID内容

- session_track_gtids
 - OFF
 - OWN_GTID
当前事务产生的GTID
 - ALL_GTIDS

等待特定GTID的事务被执行

- `WAIT_FOR_EXECUTED_GTID_SET(gtid_set[, timeout])`

2 MySQL 5.7 GA里的复制新功能

2.1 GTID 新功能

2.2 多线程复制(Multi-Threaded Slave)新功能

2.3 多源复制(Multiple Source Replication)

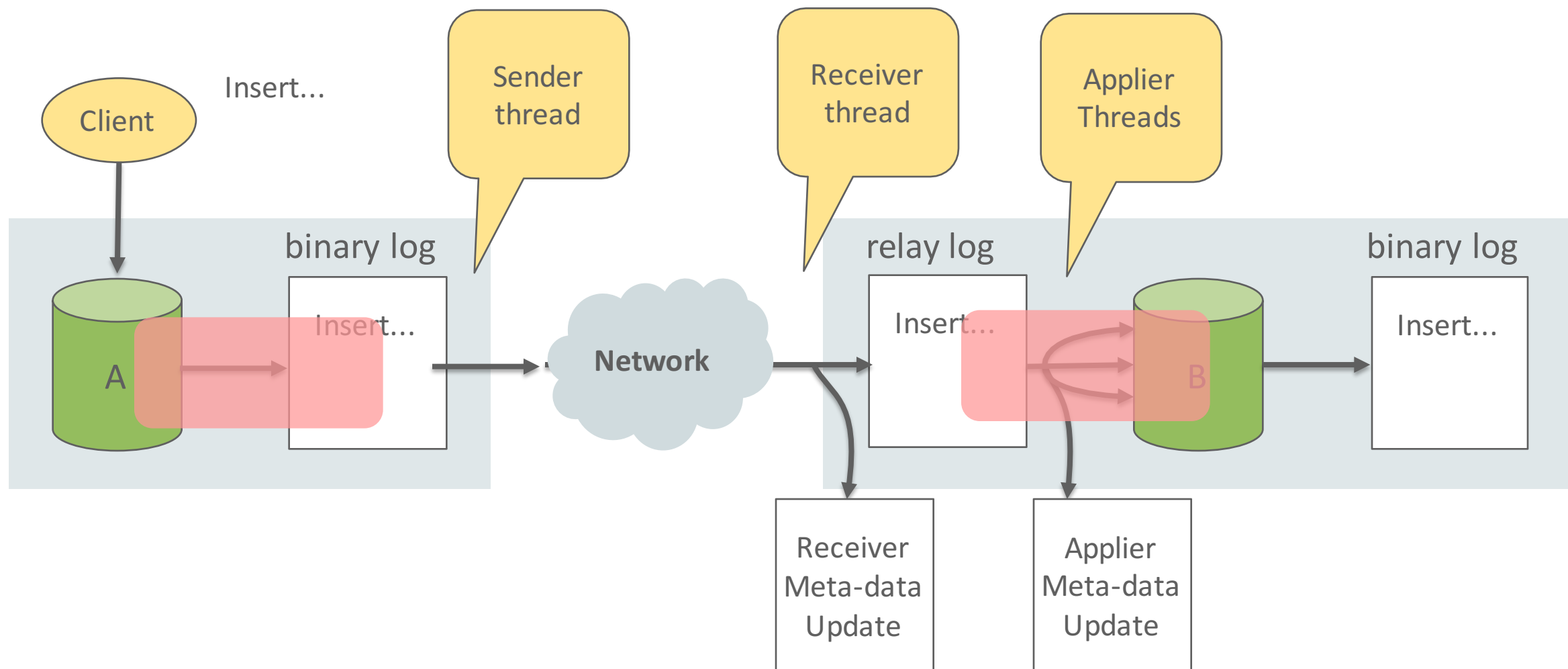
2.4 Slave 可用性的改进

2.5 Master 性能的提升

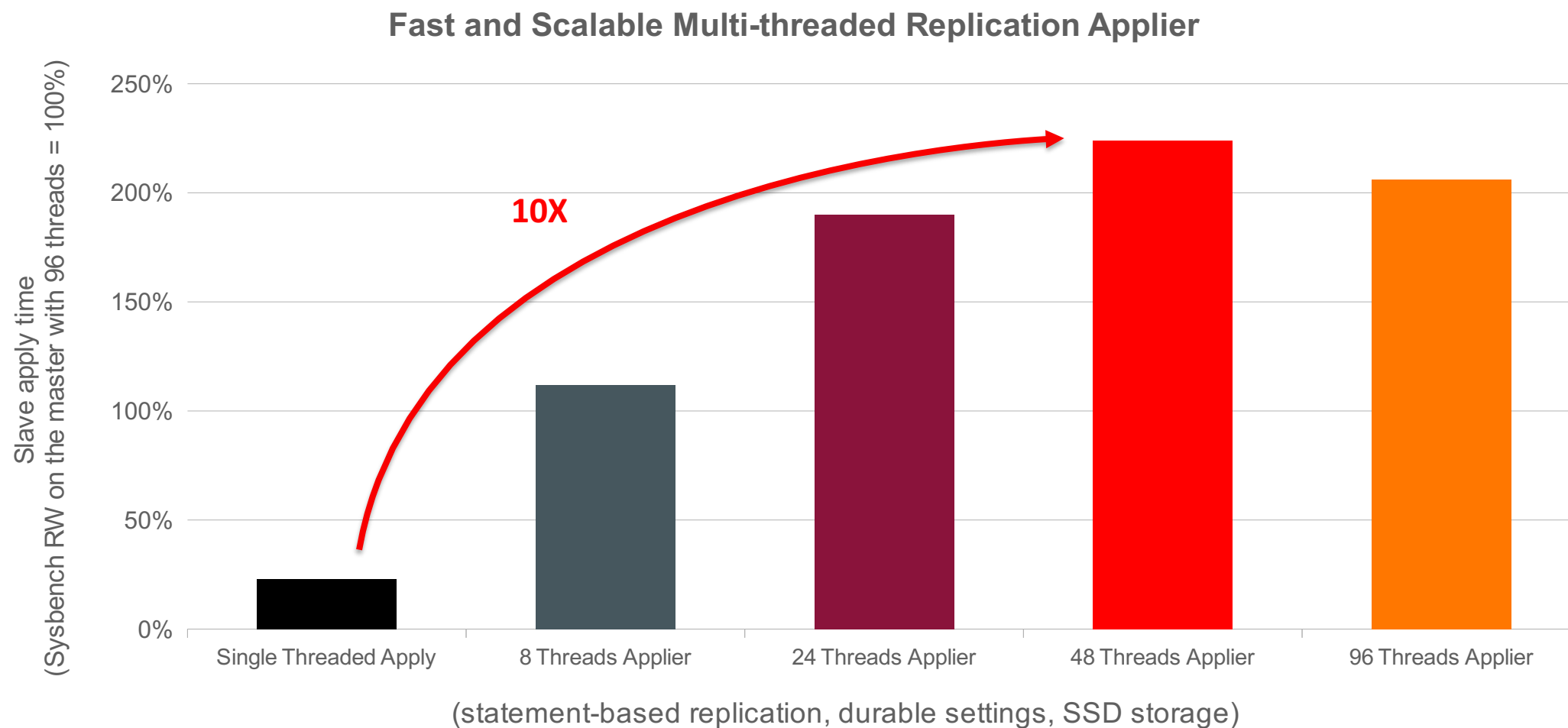
2.6 半同步复制(Semi-synchronous Replication)的新功能

2.7 其他功能

基于锁的并发复制

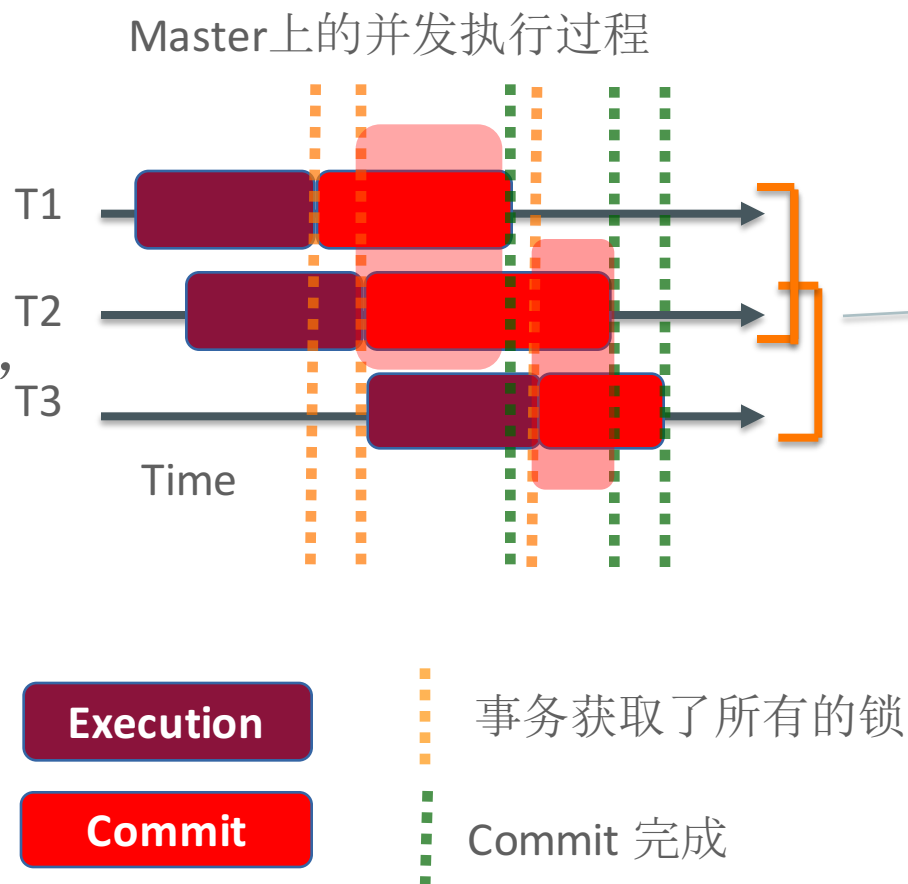


基于锁的并发复制



基于锁的并发复制

- 如果两个并发的的事务，都获取了执行所需要的所有的锁，这两个事务没有锁的冲突，可以在Slave上并发执行那个。
- 当事务开始执行COMMIT语句时，它已经获取了所有的锁。
- 当事务进入prepare阶段时，它已经获取了所有的锁。
- Master上的并发信息，以逻辑时间(Logical clock)的方式记入Gtid_log_event中。



T1 和 **T2** 可并发
T2 和 **T3** 可并发，
但 **T3** 和 **T1** 不能并发。

在Slave上，
T1 和 **T2** 并发执行，
T3 在 **T1** 完成后，
开始执行。

基于锁的并发复制

- 支持语句(statement format)和行(row format)格式的binlog.
- 并发策略通过系统变量控制:

```
mysql> SET slave_parallel_type= [logical_clock|database]
```

- ***logical_clock*** - 采用基于锁的并发策略
- ***database*** - 基于数据库的并发策略(MySQL-5.6上的并发机制)
- 提升Slave性能的相关工作仍在进行中，并不会止步于此.

微调master的Commit过程，获取更好的并发复制性能

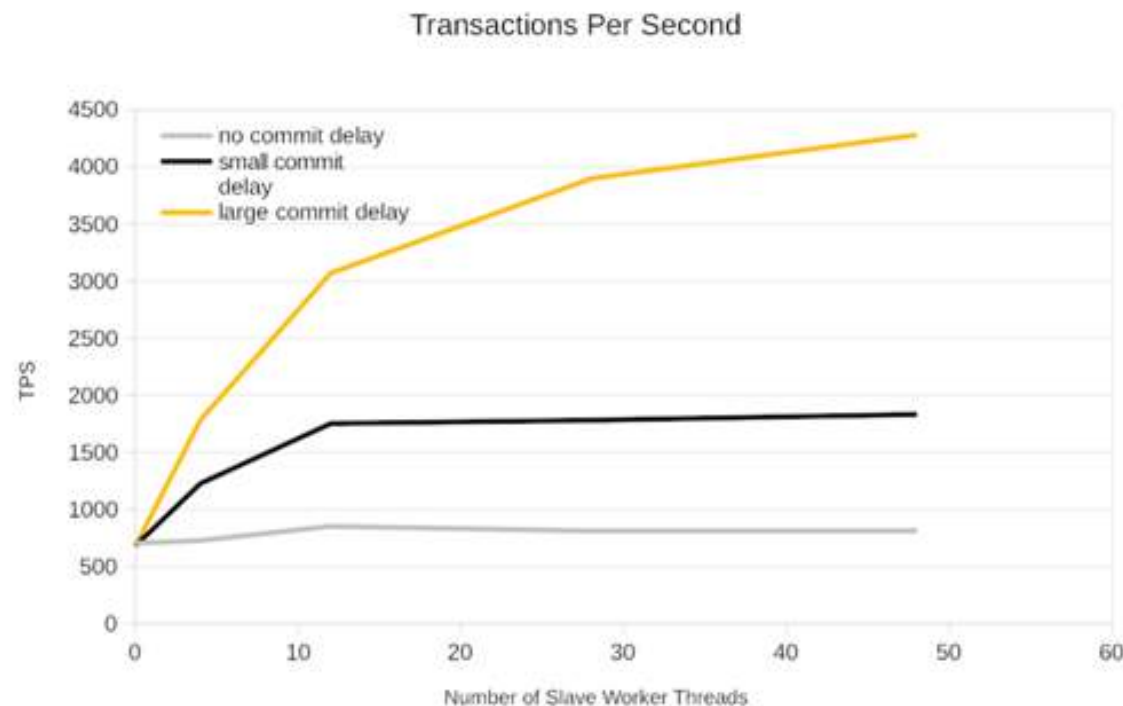
- Master上并发COMMIT的事务越多, Slave的并发性能越好。
- 用来微调的系统变量

`binlog_group_commit_sync_delay`

Binlog sync 之前等待一小段时间

`Binlog_group_commit_sync_no_delay_count`

在等待超时之前，如果有足够数量的事务则，停止等待。



- **6X** slave throughput – large commit delay
- **3X** slave throughput – small commit delay

其他“小”而有趣的改进

- **Multi-threaded applier** 上的slave-transaction-retries
 - Applier 线程可以并发的重做失败的事务。
 - 可以重做的错误类型和单线程复制相同：加锁超时和死锁。
- **Multi-threaded applier** 保持master事务提交顺序
 - Slave_preserve_commit_order
 - ON
 - OFF
 - 仅适用于Logical_clock类型的多线程复制

2 MySQL 5.7 GA里的复制新功能

2.1 GTID 新功能

2.2 多线程复制(Multi-Threaded Slave)新功能

2.3 多源复制(Multiple Source Replication)

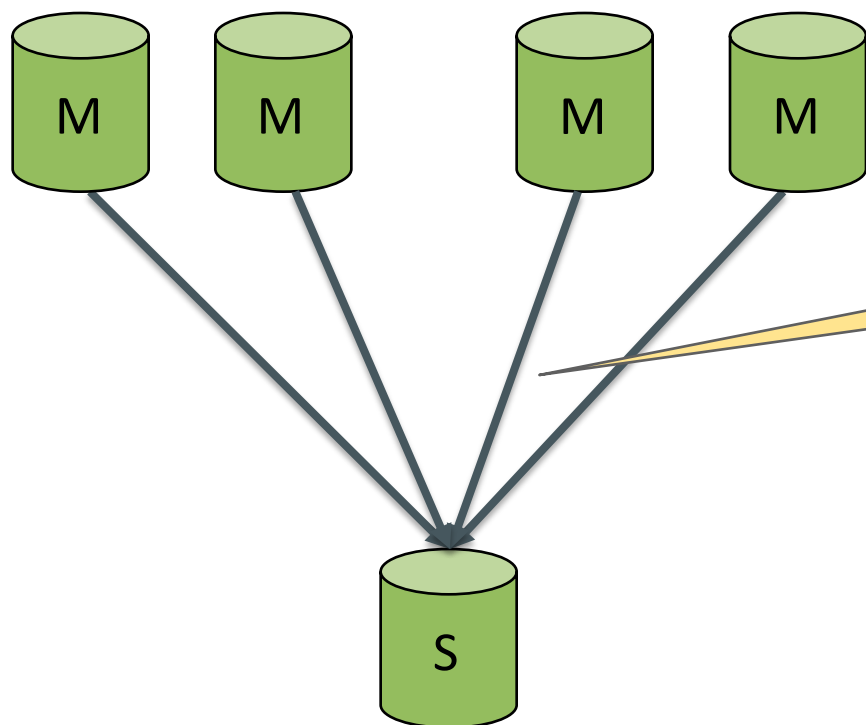
2.4 Slave 可用性的改进

2.5 Master 性能的提升

2.6 半同步复制(Semi-synchronous Replication)的新功能

2.7 其他功能

多源复制



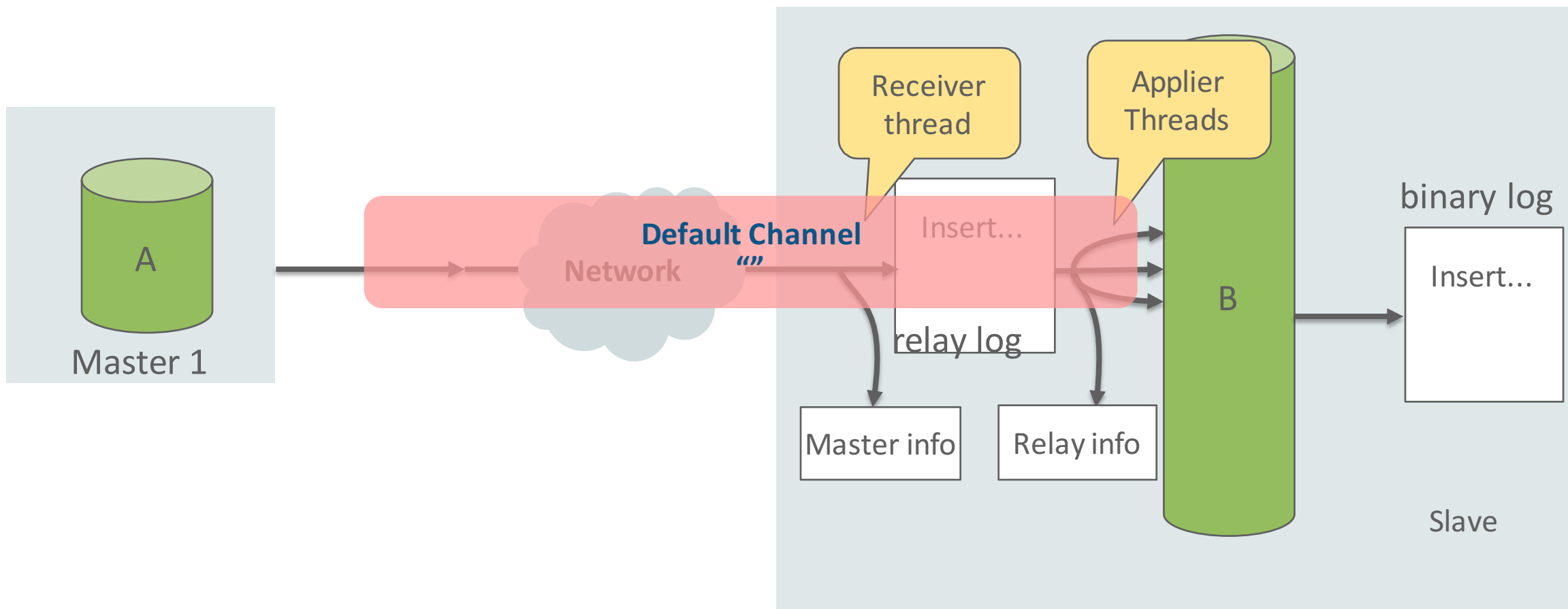
Slave 可以有多个复制源

需要聚合数据到一台数据库服务器上的场景:

- 集成备份;
- 复合数据分析方面的用途;
- 分片数据的合并

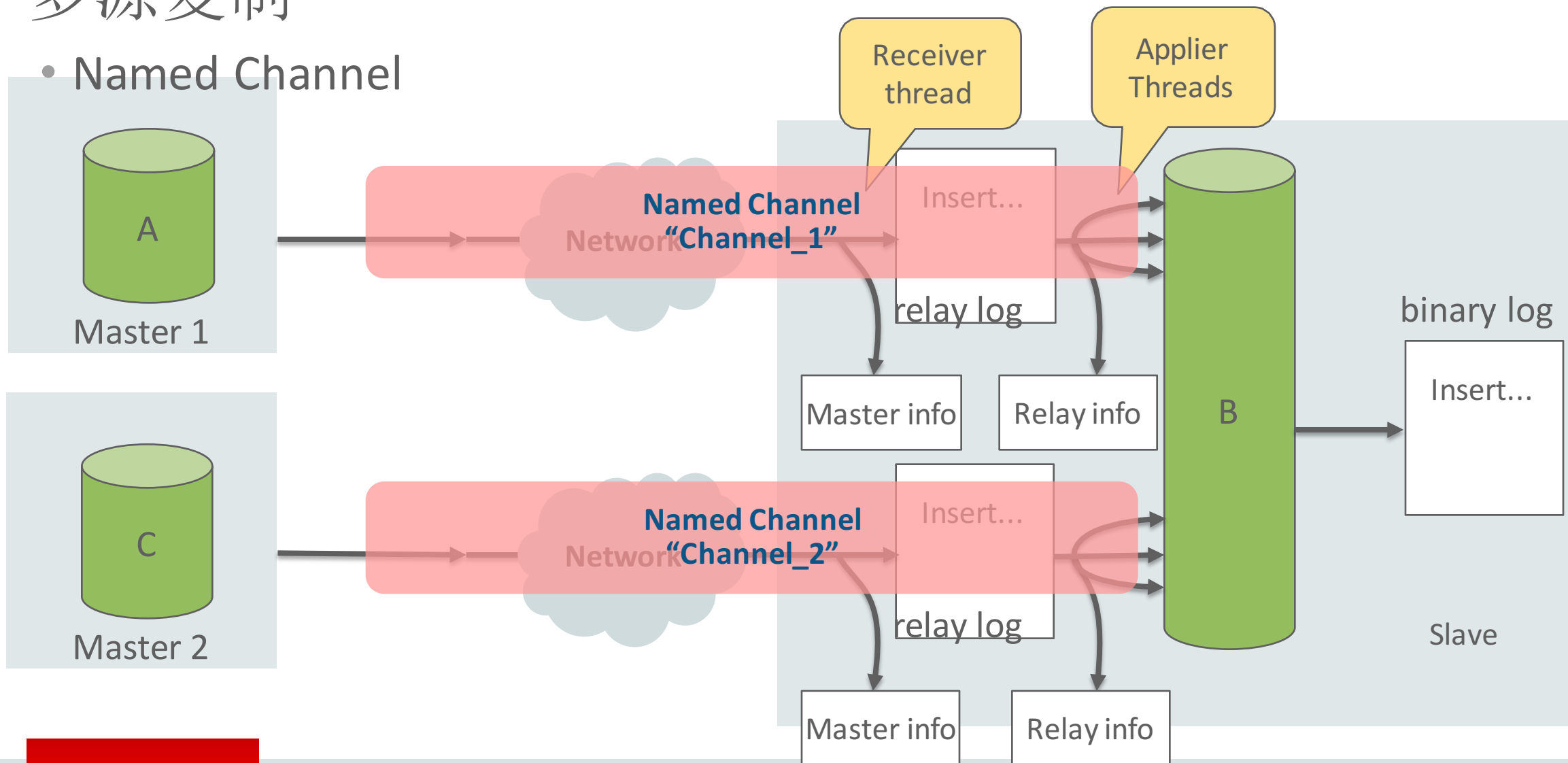
多源复制

- Default Channel



多源复制

- Named Channel



多源复制

- 多个channels(channel包含: receiver thread, relay logs, applier threads) , 每个channel能够独立的运行和停止
- 通过P_S表进行状态监控
 - 下列表中添加了channel_name字段, 不同channel的信息在不同的行中显示。
 - replication_applier_status_by_coordinator
 - replication_applier_status_by_worker
 - replication_connection_status

2 MySQL 5.7 GA里的复制新功能

2.1 GTID 新功能

2.2 多线程复制(Multi-Threaded Slave)新功能

2.3 多源复制(Multiple Source Replication)

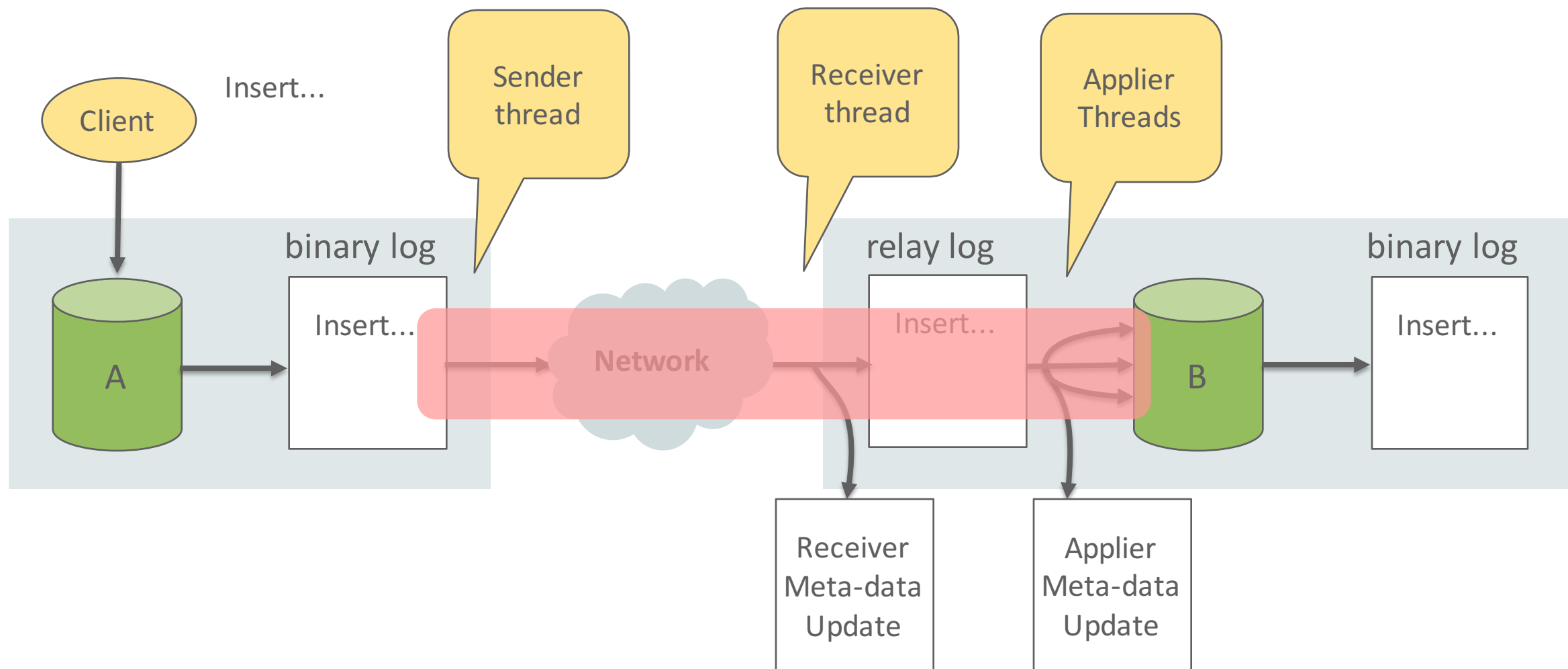
2.4 **Slave** 可用性的改进

2.5 Master 性能的提升

2.6 半同步复制(Semi-synchronous Replication)的新功能

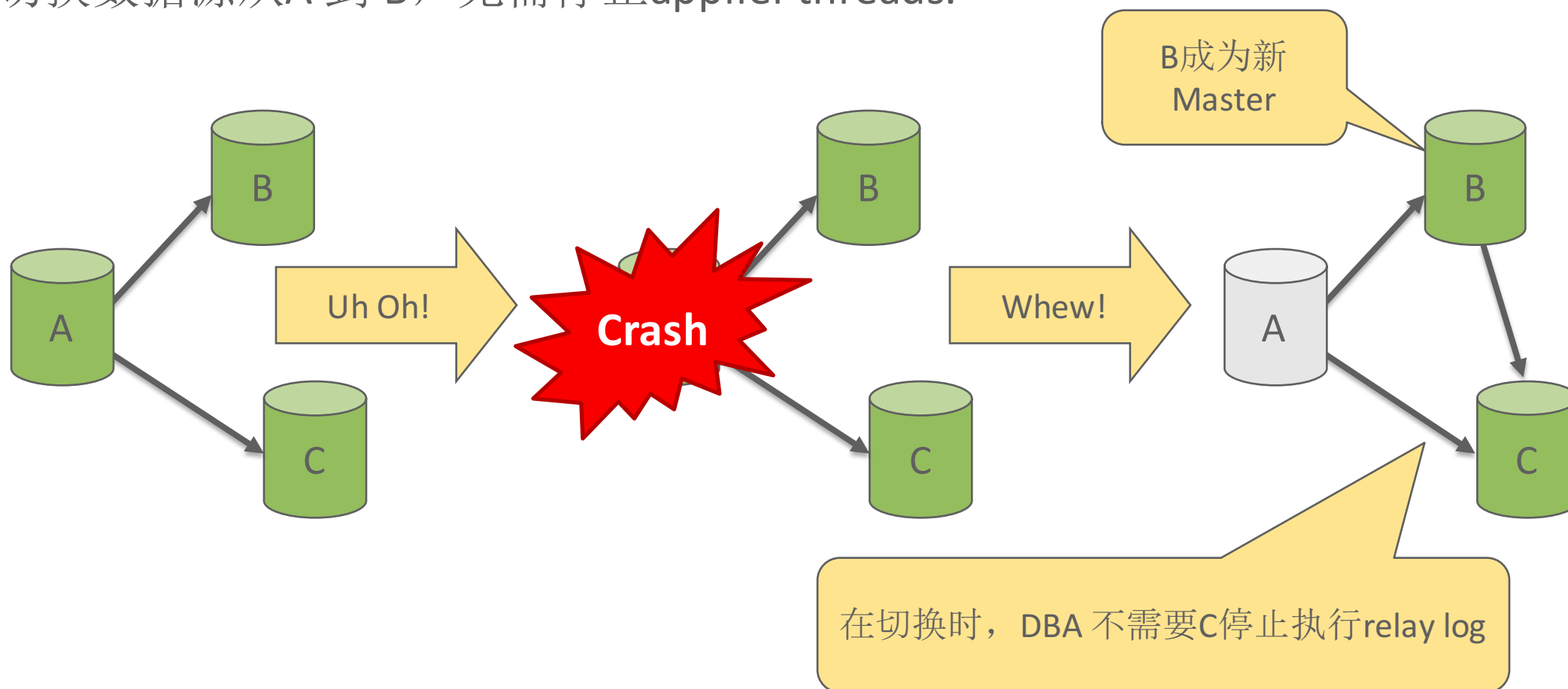
2.7 其他功能

在线配置Receiver/Applier



在线配置Receiver/Applier

切换数据源从A到B，无需停止applier threads.



在线配置Receiver/Applier

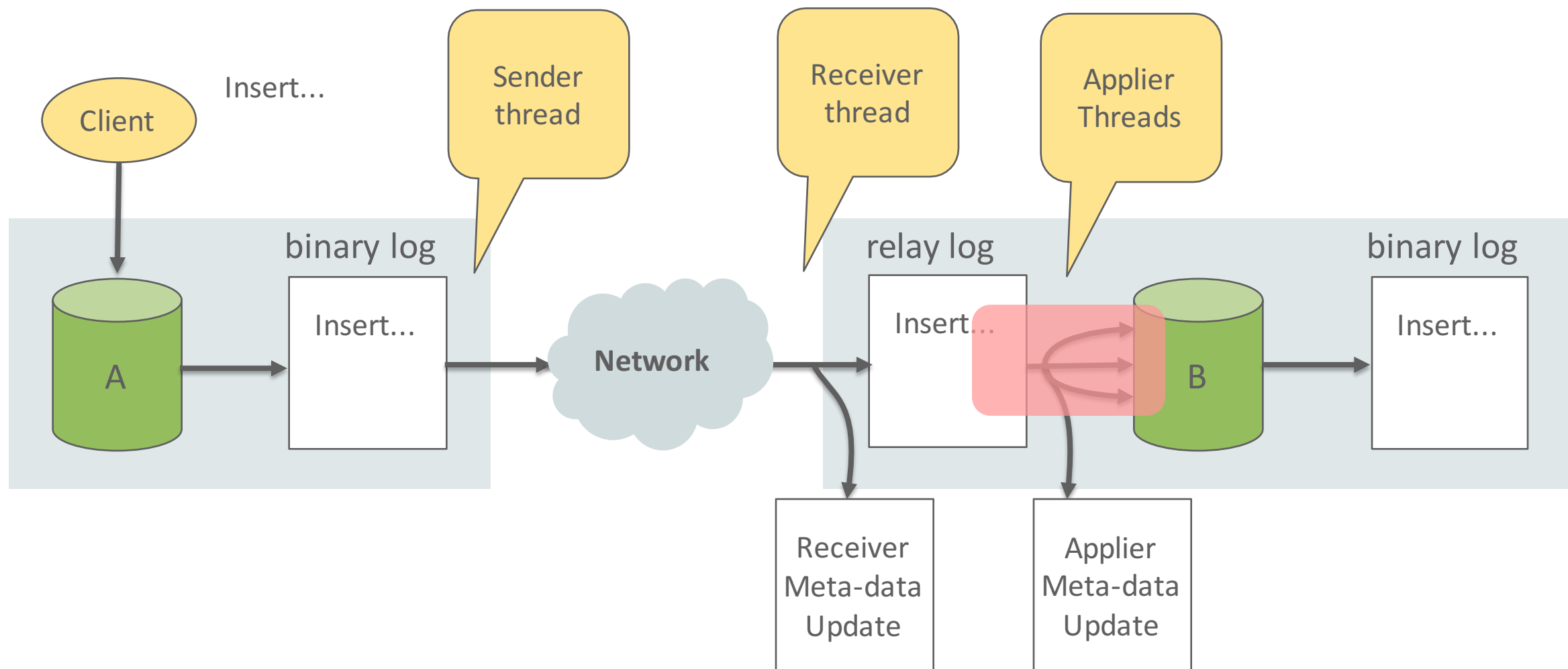
- 切换Master时，Applier线程仍然在运行:

```
mysql> STOP SLAVE IO_THREAD;  
mysql> CHANGE MASTER TO MASTER_HOST='master2', ...;  
mysql> START SLAVE IO_THREAD;
```

- 改变applier参数时，receiver线程仍然在运行:

```
mysql> STOP SLAVE SQL_THREAD;  
mysql> CHANGE MASTER TO MASTER_DELAY=3600, ...;  
mysql> START SLAVE SQL_THREAD;
```

在线变更Replication Filter参数

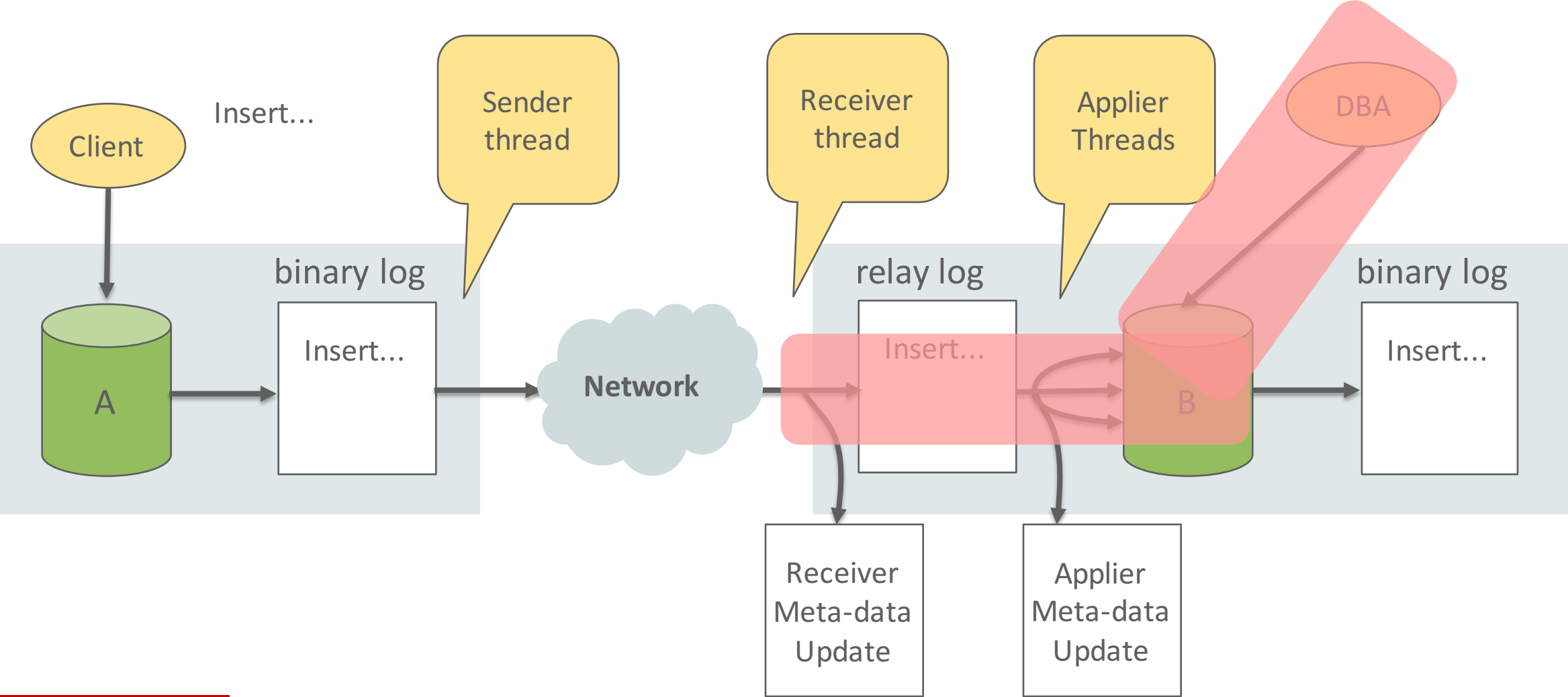


在线变更Replication Filter参数

- 动态变更Replication Filters
 - 不再需要重启MySQL server
 - 支持所有的Filter选项
 - 支持各种字符集

```
mysql> CHANGE REPLICATION FILTER REPLICATE_DO_DB= (db1, db2)
```

改进的复制监控信息

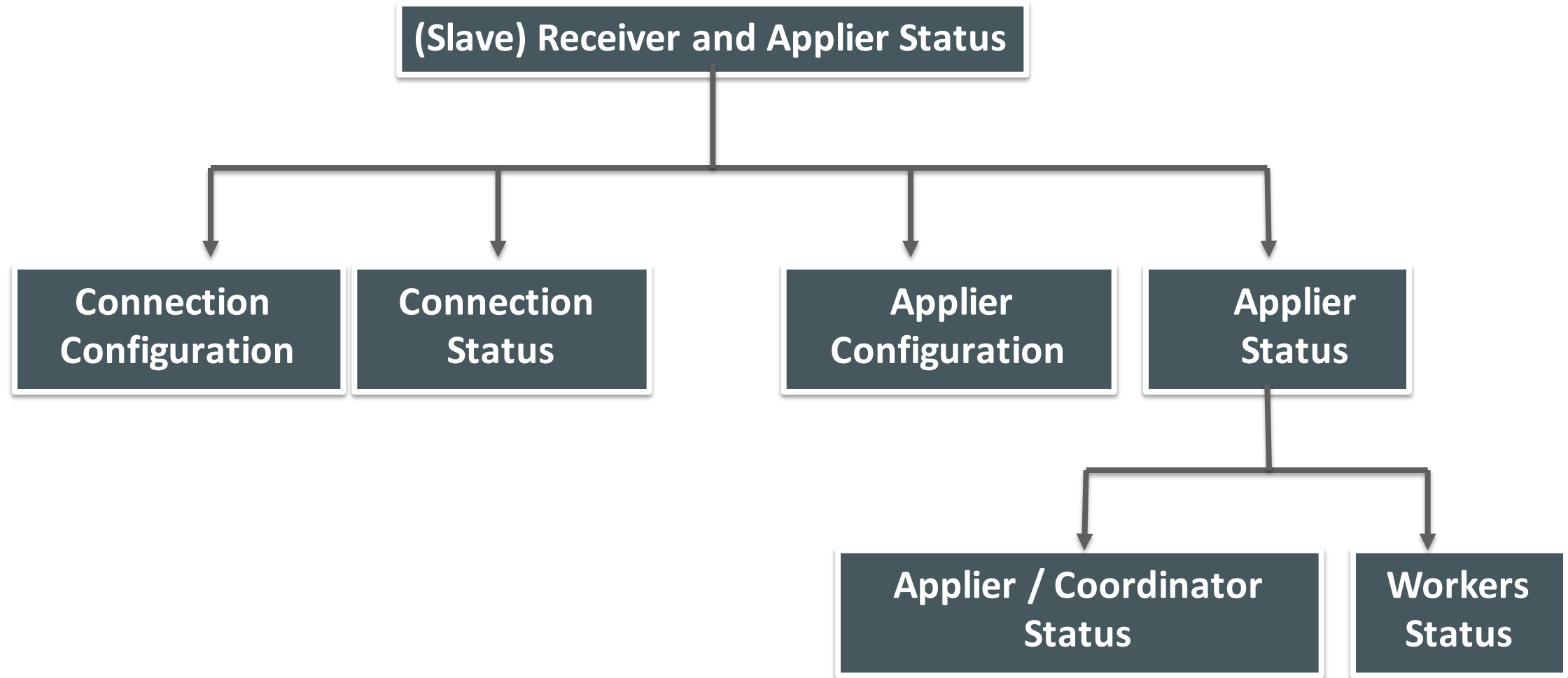


改进的复制监控信息

Replication的信息添加到了**Performance Schema** 中

- 通过SQL进行监控
- 逻辑上无关的信息被放在不同的表中
- 易于扩展和支持新功能
- 更加准确和一致的标识名称
- 支持Master-slave, Multi-source, Group Replication.

改进的复制监控信息



改进的复制监控信息

```
mysql> select * from performance_schema.replication_applier_status_by_worker\G
***** 1. row *****
      CHANNEL_NAME:
        WORKER_ID: 1
        THREAD_ID: 35
      SERVICE_STATE: ON
LAST_SEEN_TRANSACTION: 4ba0eb86-63c3-11e4-92ba-28b2bd168d07:2368
      LAST_ERROR_NUMBER: 0
      LAST_ERROR_MESSAGE:
      LAST_ERROR_TIMESTAMP: 0000-00-00 00:00:00
***** 2. row *****
      CHANNEL_NAME:
        WORKER_ID: 2
        THREAD_ID: 36
      SERVICE_STATE: ON
LAST_SEEN_TRANSACTION: 4ba0eb86-63c3-11e4-92ba-28b2bd168d07:2367
      LAST_ERROR_NUMBER: 0
      LAST_ERROR_MESSAGE:
      LAST_ERROR_TIMESTAMP: 0000-00-00 00:00:00
2 rows in set (0,00 sec)
```


2 MySQL 5.7 GA里的复制新功能

2.1 GTID 新功能

2.2 多线程复制(Multi-Threaded Slave)新功能

2.3 多源复制(Multiple Source Replication)

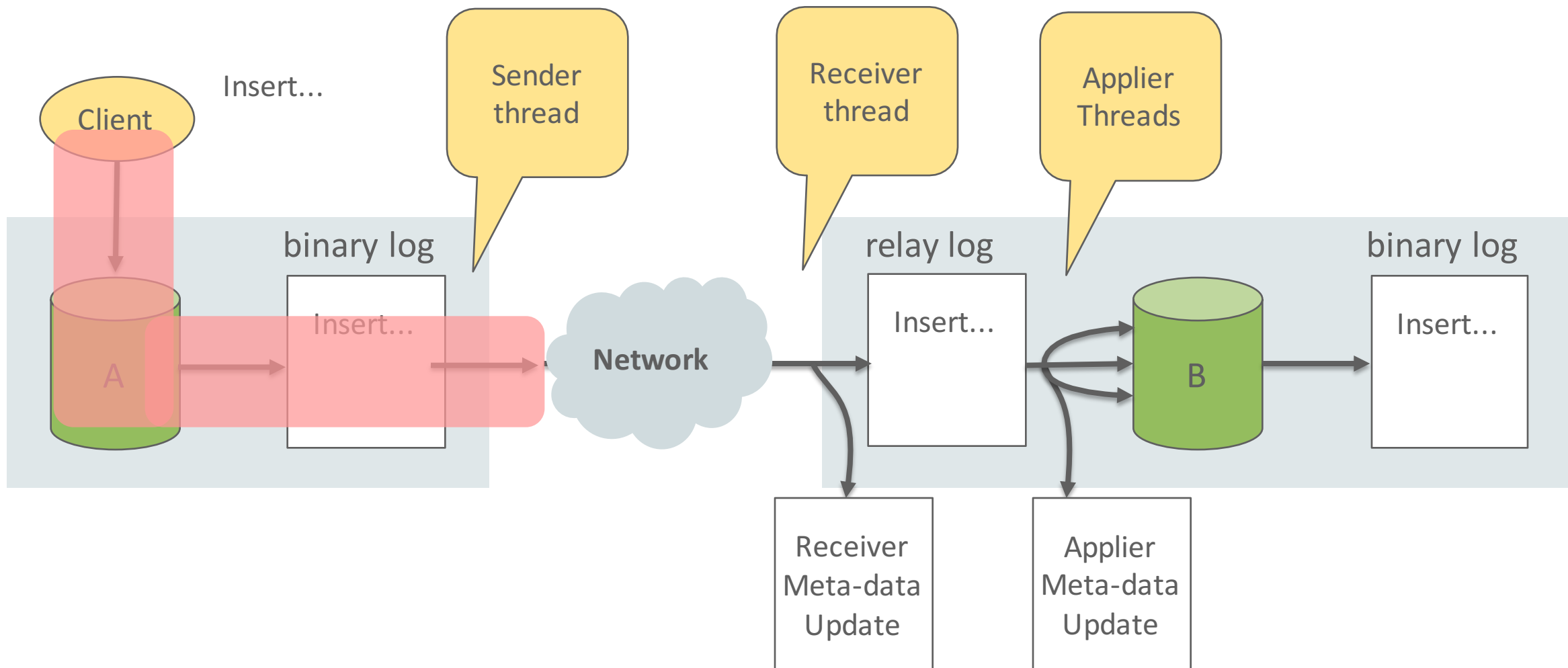
2.4 Slave 可用性的改进

2.5 Master 性能的提升

2.6 半同步复制(Semi-synchronous Replication)的新功能

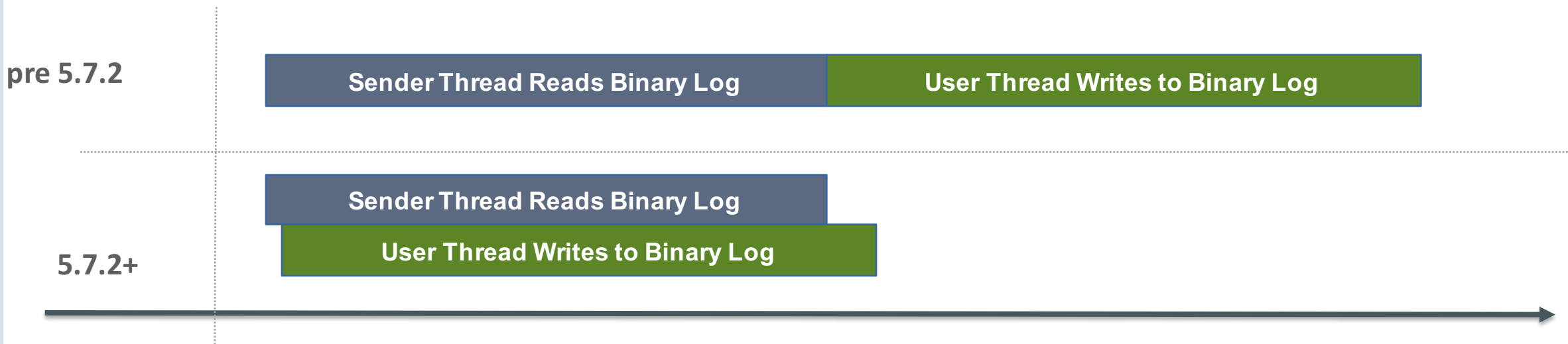
2.7 其他功能

并发读写binlog

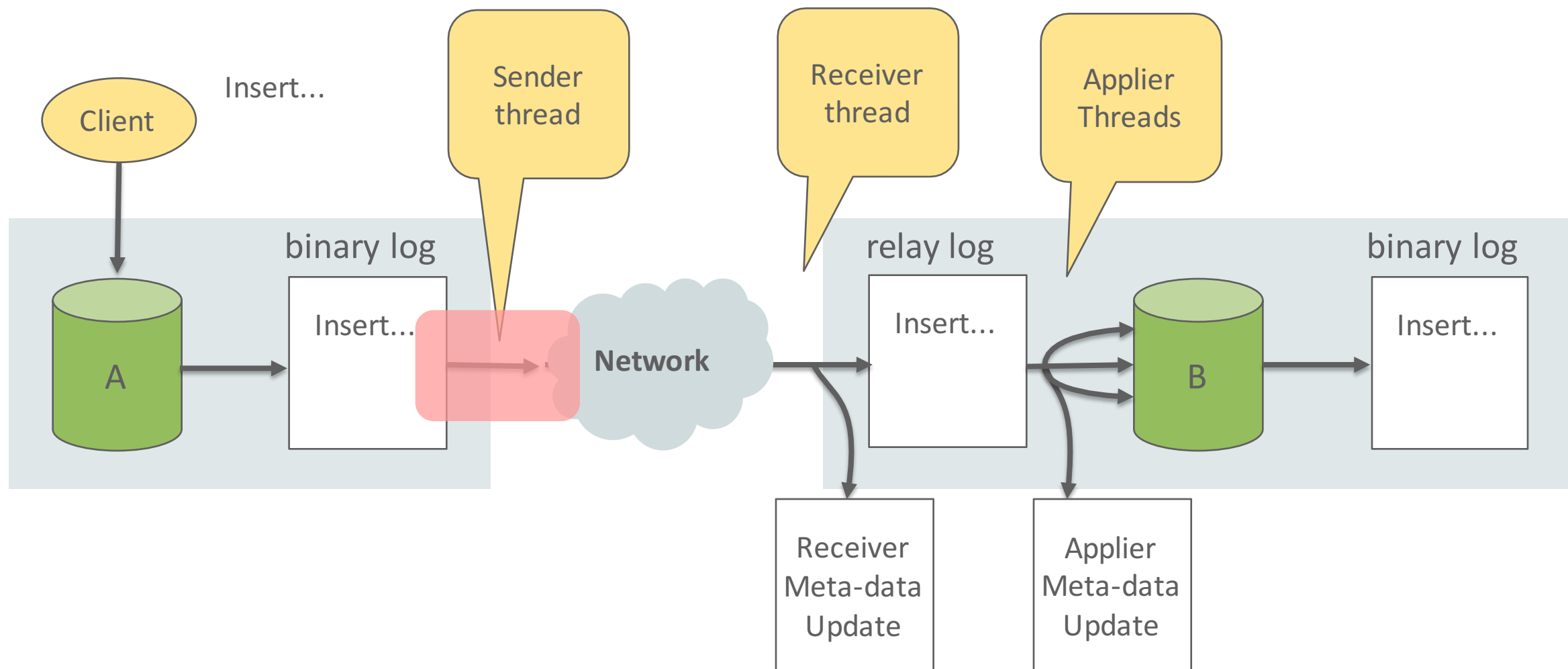


并发读写binlog

- sender 线程在读binlog的同时,用户线程并发的记录binlog日志.
 - Sender线程和用户线程在读写binlog时, 不再互相阻塞对方
 - Sender线程和用户线程都有更好的性能



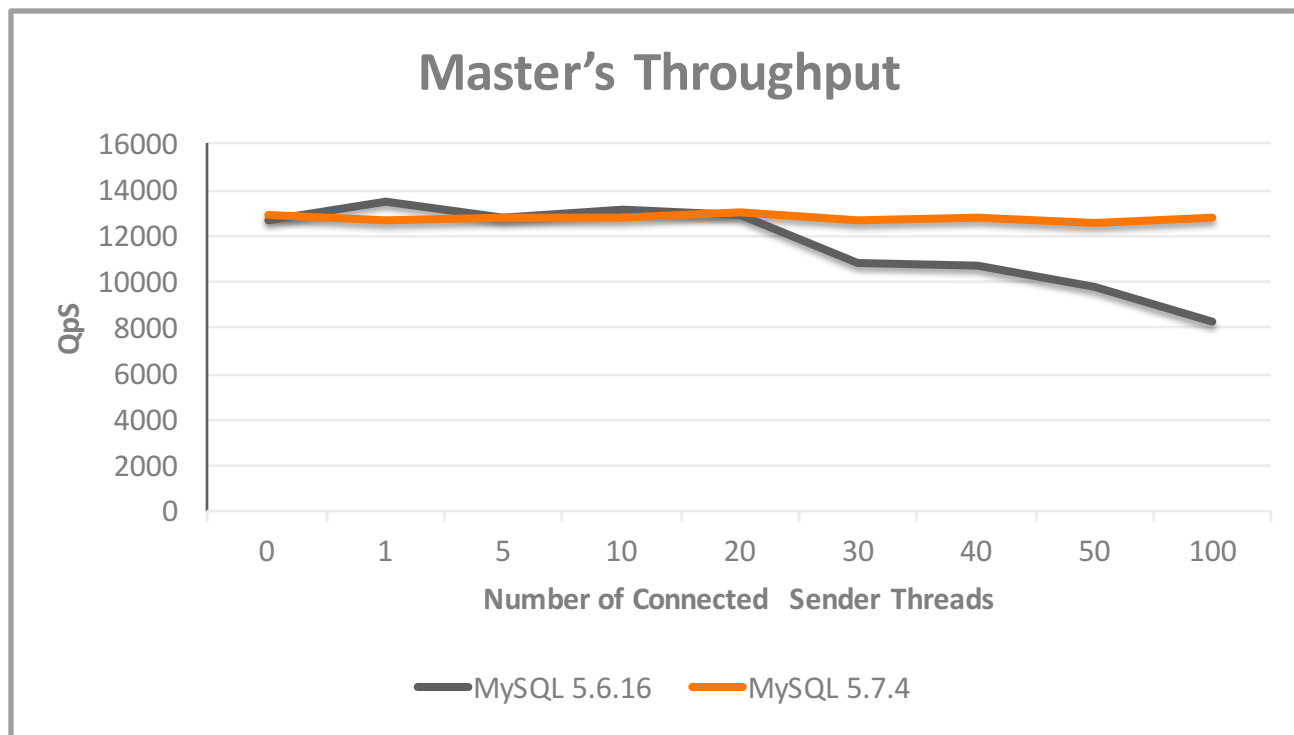
Sender线程的Event缓存管理



Sender线程的Event缓存管理

- 增加了缓存管理功能
- 不再为每个Binlog Event分配和释放内存。
- 当需要更大的Event存储空间时，缓存自动增加。
- 当缓存空间过大时，会自动减少。
- 缓存管理带来的好处：
 - 增加了sender线程的延展性
 - 减少了CPU资源的使用
 - 提高了master负载高峰的应对能力

Master的性能



- 用mysqlslap做的一个简单测试
- 1M 数据, 并发用户=200,
- N sender线程(用mysqlbinlog发起)
- 48 cores HT / 512 GB RAM / HDD

2 MySQL 5.7 GA里的复制新功能

2.1 GTID 新功能

2.2 多线程复制(Multi-Threaded Slave)新功能

2.3 多源复制(Multiple Source Replication)

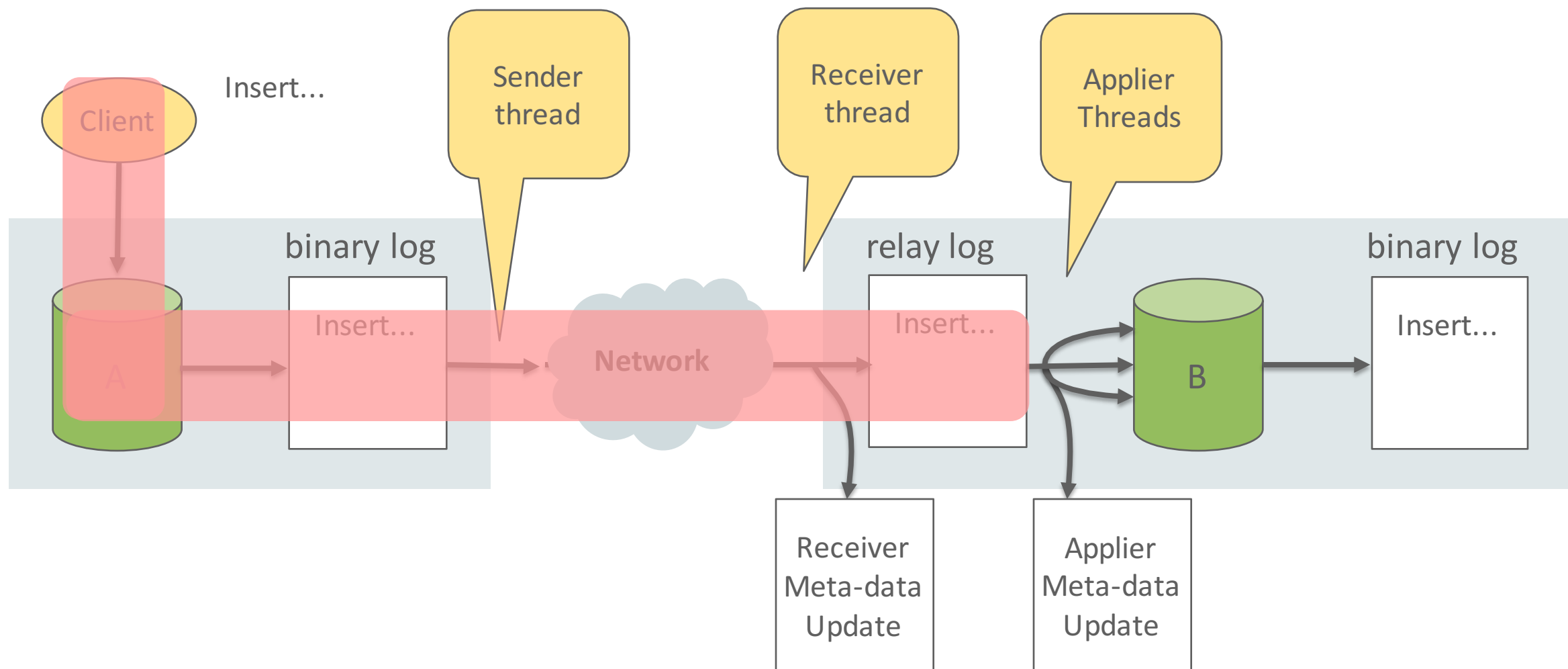
2.4 Slave 可用性的改进

2.5 Master 性能的提升

2.6 半同步复制(Semi-synchronous Replication)的新功能

2.7 其他功能

无数据丢失的半同步复制

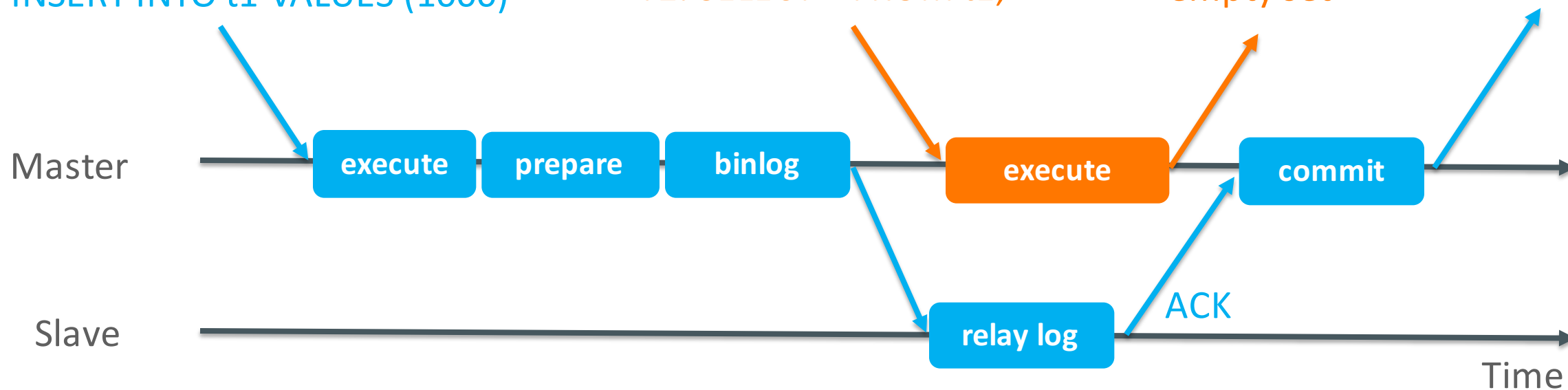


无数据丢失的半同步复制

T1: INSERT INTO t1 VALUES (1000)

T2: SELECT * FROM t1;

empty set

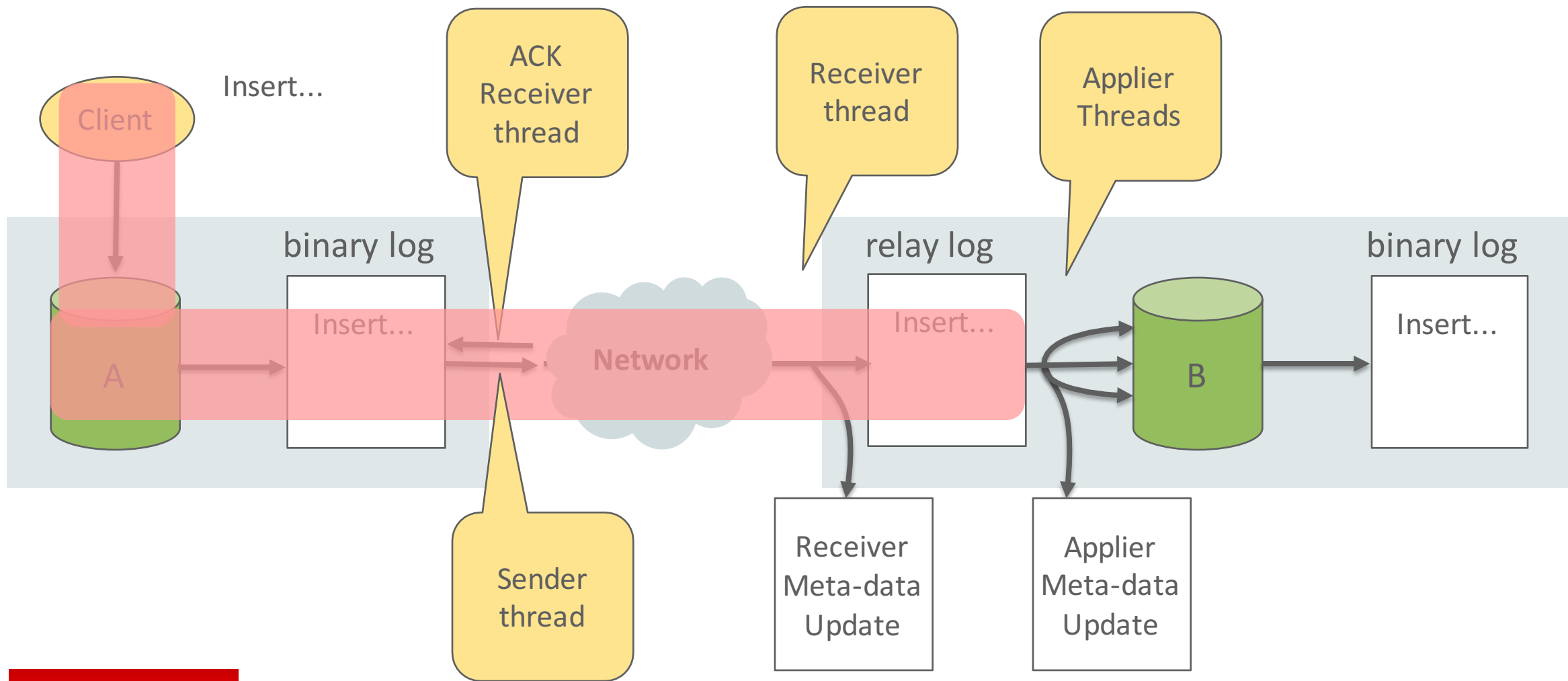


无数据丢失的半同步复制

- Master 在收到slave的应答后才Commit事务(5.6上Master在commit后,才等待Slave的应答).
 - 因此在确认事务复制到Slave上之前, 并发的事务看不到当前事务的数据.
- 当Master出现故障时,所有已经提交的事务都复制到了Slave上.
- 缺省采用无数据丢失的应答等待机制。用户可以选择使用5.6的应答等待机制。

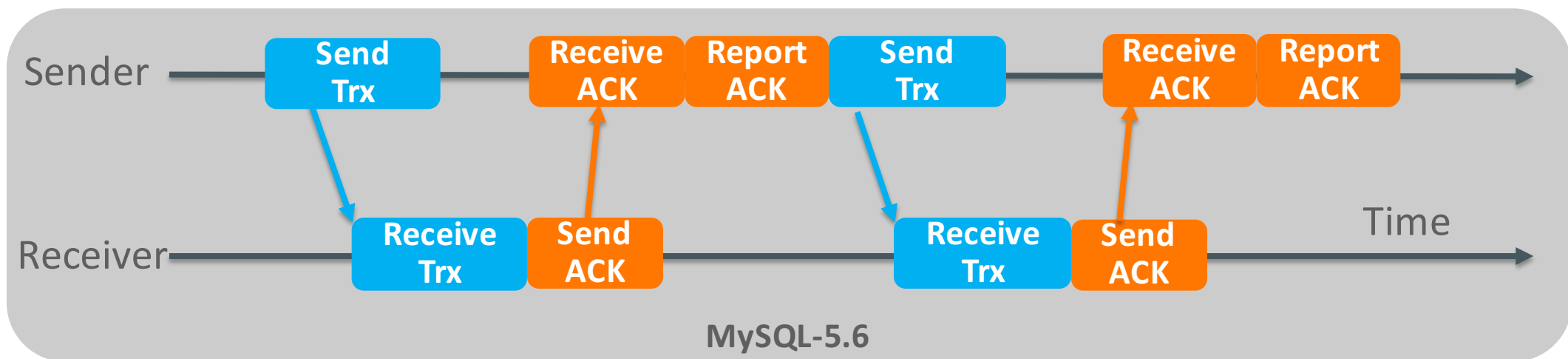
```
mysql> SET rpl_semi_sync_master_wait_point= [AFTER_SYNC|AFTER_COMMIT]
```

更快的半同步复制- ACK接收线程



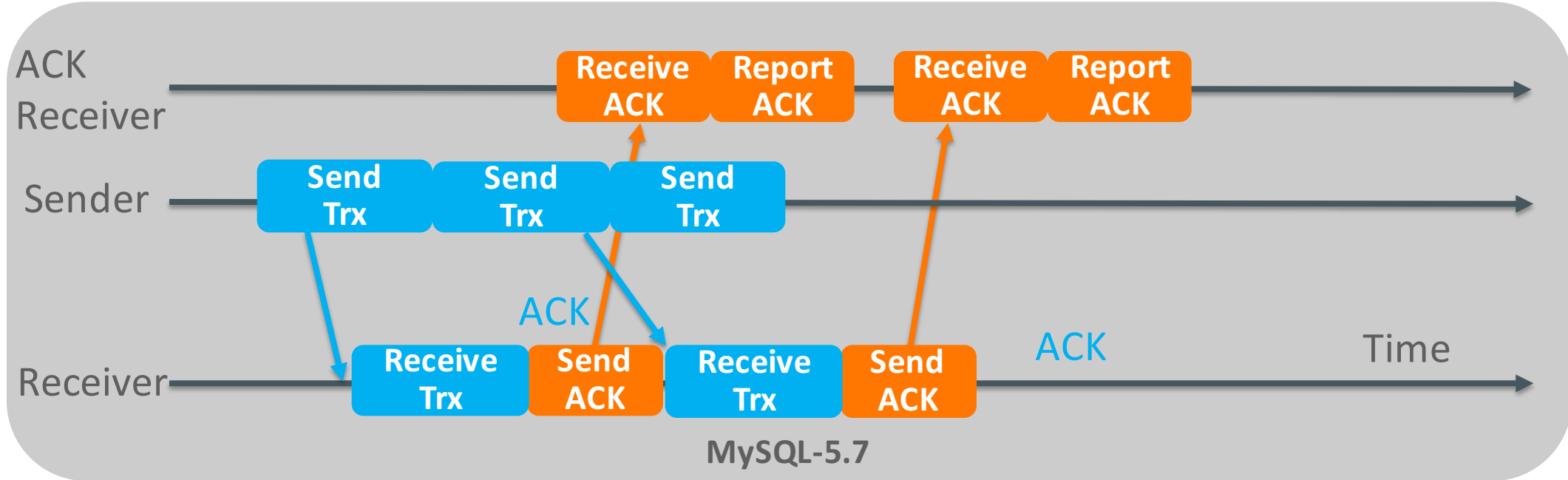
更快的半同步复制- ACK接收线程

- 单工通讯方式，事务发送完毕后，要接收和处理应答。处理完答后才能继续发送下事务的Events。



更快的半同步复制- ACK接收线程

- 创建单独的应答接收线程。
- 变成双工模式。发送和接收互不影响。



更快的半同步复制- ACK接收线程

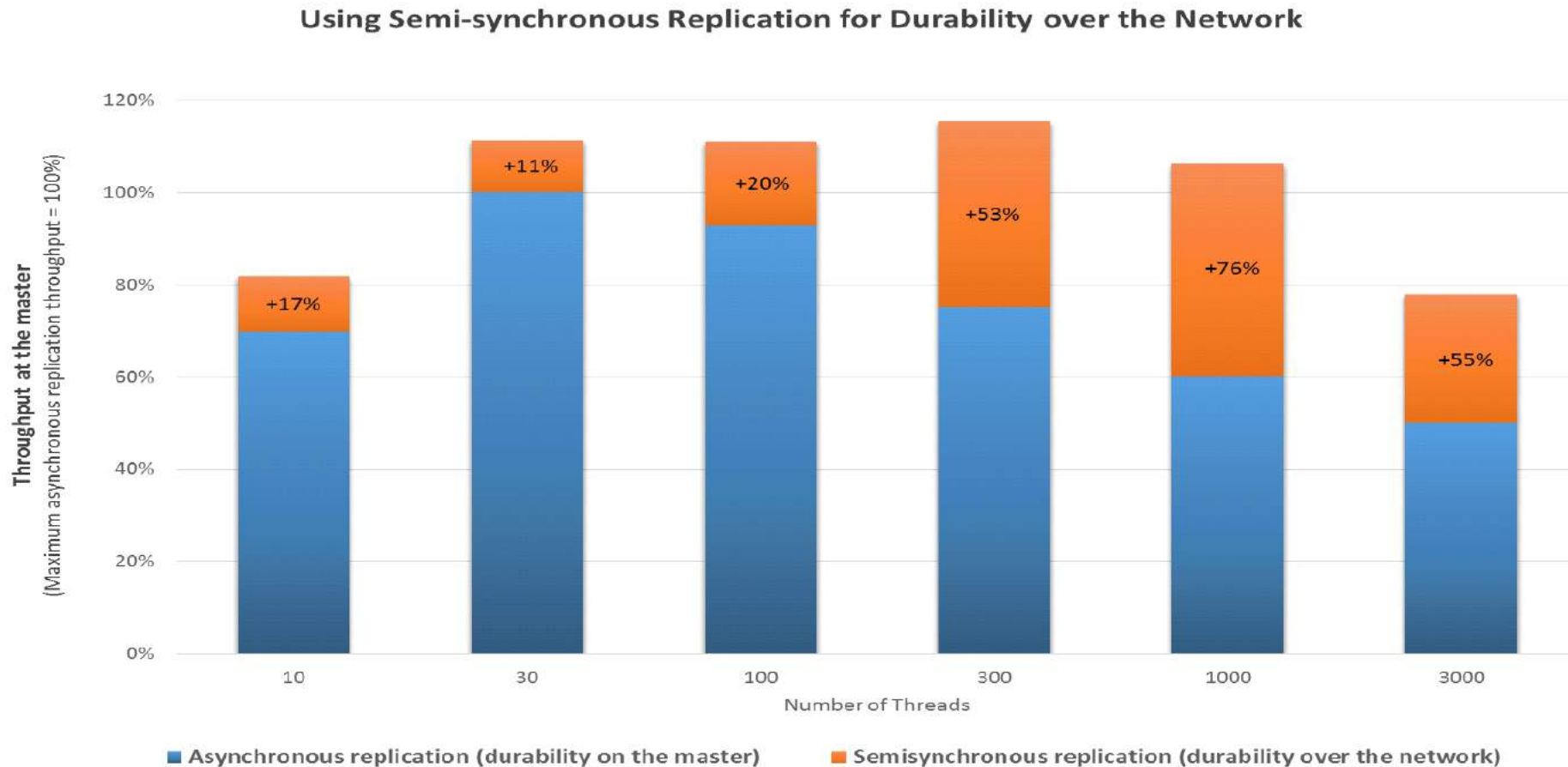
- 启用半同步复制时，应答线程会自动创建。

```
mysql> SET GLOBAL rpl_semi_sync_master_enabled= ON
```

- 关闭半同步复制时，应答线程会自动关闭。

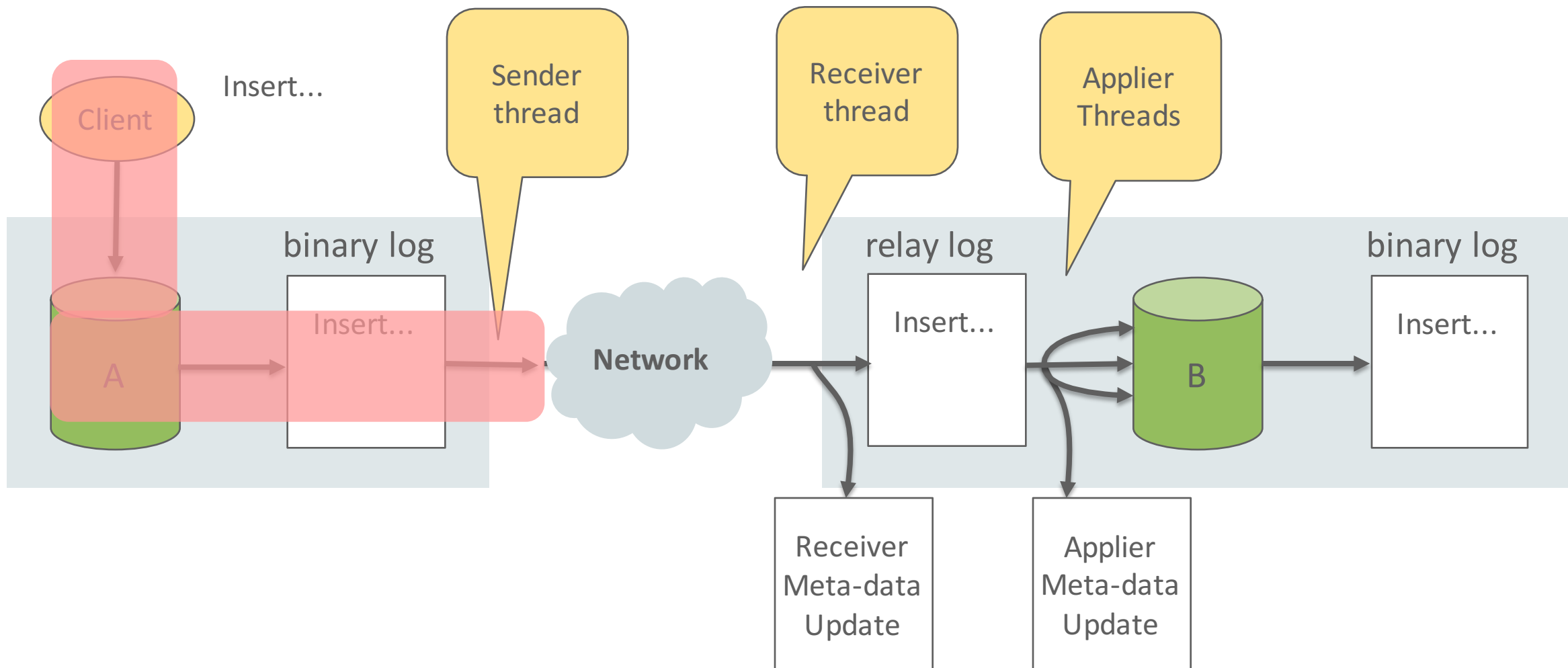
```
mysql> SET GLOBAL rpl_semi_sync_master_enabled= OFF
```

更快的半同步复制: 通过网络持久化数据



蓝色部分代表本地持久化数据的性能(sync_binlog=1).
橘色部分是半同步复制持久化数据的性能比本地持久化数据性能高出的比例(sync_binlog=0).

半同步复制- 等待多个Slave的应答



半同步复制- 等待多个Slave的应答

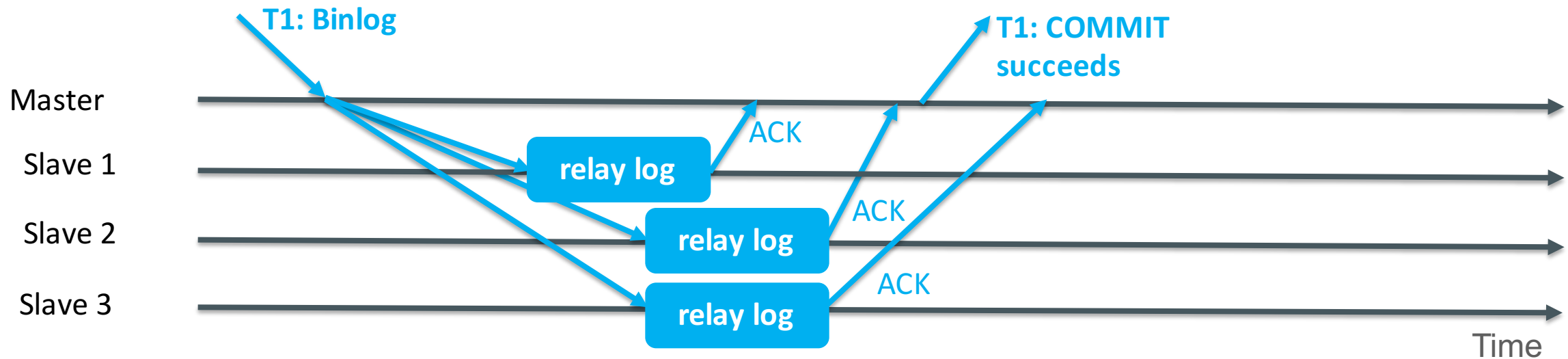
- Master 接收到N个slave的应答后，才**commit** 事务.
- 用户可以设置应答Slave的数量:

```
mysql> SET GLOBAL rpl_semi_sync_master_wait_for_slave_count= N
```

半同步复制- 等待多个Slave的应答

- Master 接收到N个slave的应答后，才**commit** 事务.
- 用户可以设置应答Slave的数量:

```
mysql> SET GLOBAL rpl_semi_sync_master_wait_for_slave_count= N
```



```
mysql> SET GLOBAL rpl_semi_sync_master_wait_for_slave_count= 2
```

2 MySQL 5.7 GA里的复制新功能

2.1 GTID 新功能

2.2 多线程复制(Multi-Threaded Slave)新功能

2.3 多源复制(Multiple Source Replication)

2.4 Slave 可用性的改进

2.5 Master 性能的提升

2.6 半同步复制(Semi-synchronous Replication)的新功能

2.7 其他功能

Binlog XA事务

- XA transaction 被记录成2部分

- Prepare 部分

当执行XA PREPARE 语句时，记录：XA START+DML+XA PREPARE

- Commit 部分

记录'XA COMMIT'

- 两部分都有各自的GTID

- 示例

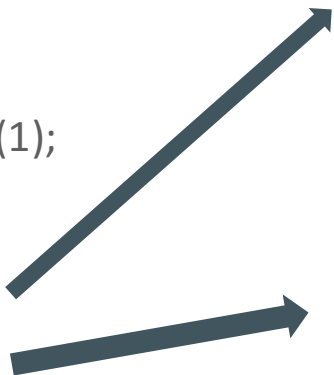
XA START 'xa trx1';

INSERT INTO t1 VALUES(1);

XA END 'xa trx1';

XA PREPARE 'xa trx1';

- **XA COMMIT 'xa trx1';**



```
master-bin.000001 316 Gtid 1 381 SET @@SESSION.GTID_NEXT= 'a_uuid:2'
master-bin.000001 381 Query 1 481 XA START X'736c6231',X",1
master-bin.000001 481 Query 1 579 use `test`; INSERT INTO t1 VALUES(1)
master-bin.000001 579 Query 1 670 XA END X'736c6231',X",1
master-bin.000001 670 XA_prepare 1 710 XA PREPARE X'736c6231',X",1
```

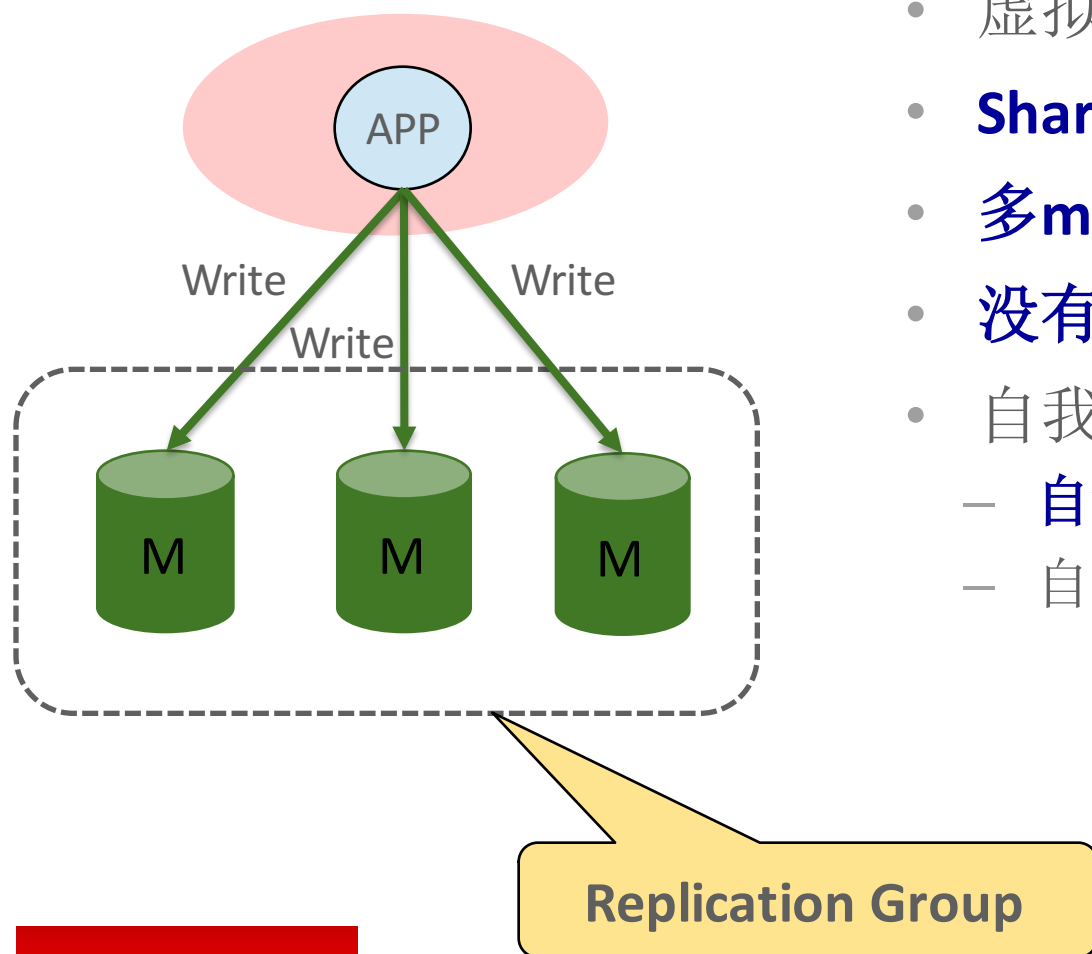
```
master-bin.000001 710 Gtid 1 775 SET @@SESSION.GTID_NEXT= 'a_uuid:3'
master-bin.000001 775 Query 1 869 XA COMMIT X'736c6231',X",1
```

其他“小”而有趣的改进

- 改变了一些变量的缺省值。例如binlog_format=**ROW**， sync_binlog=**1**.
- mysqlbinlog支持SSL连接.
- mysqlbinlog上支持 **Rewrite DB** 规则.

3 Group Replication和MySQL集群管理

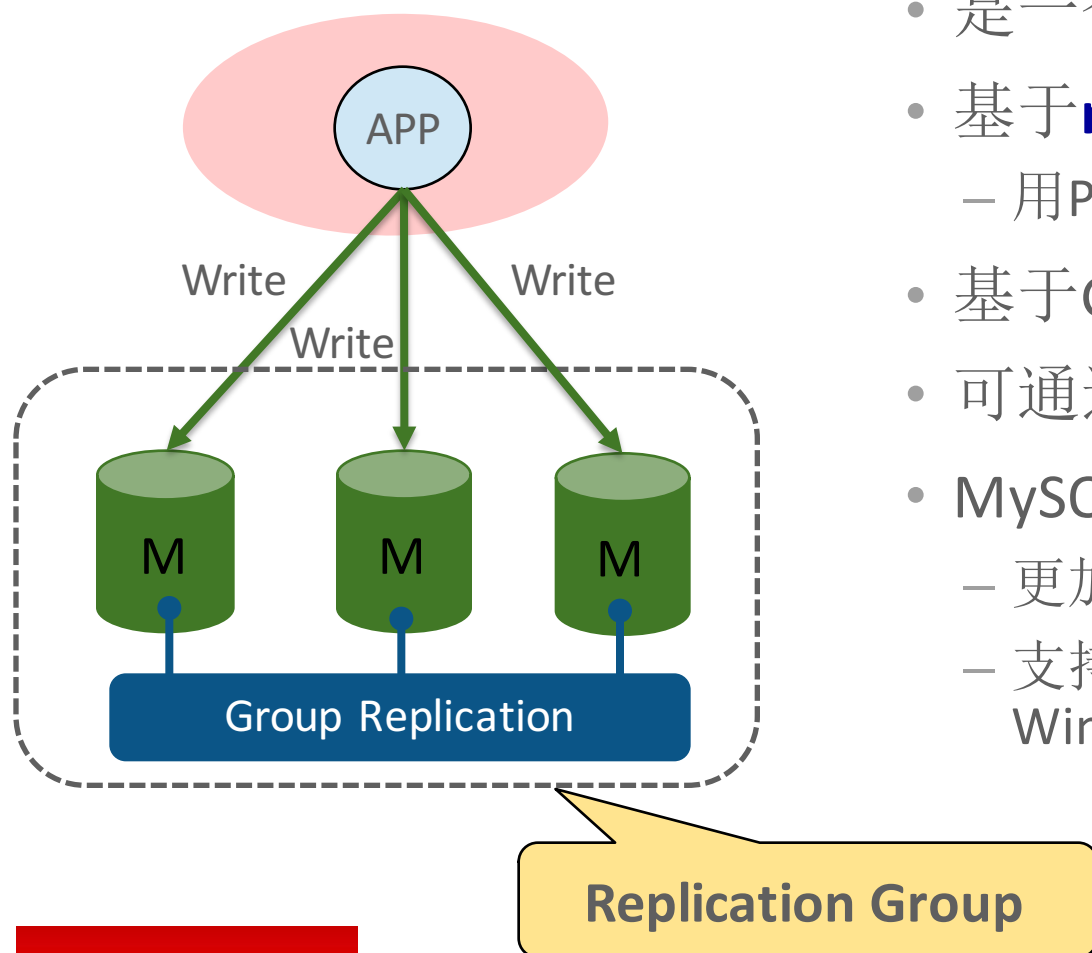
MySQL Group Replication



- 虚拟的同步复制
- **Shared-nothing** state machine replication.
- 多master同时更新
- 没有单点故障，不需要fail-over
- 自我管理
 - 自动化创建同步新节点和移除节点。
 - 自动化故障检测和节点管理。

MySQL Group Replication 的实现

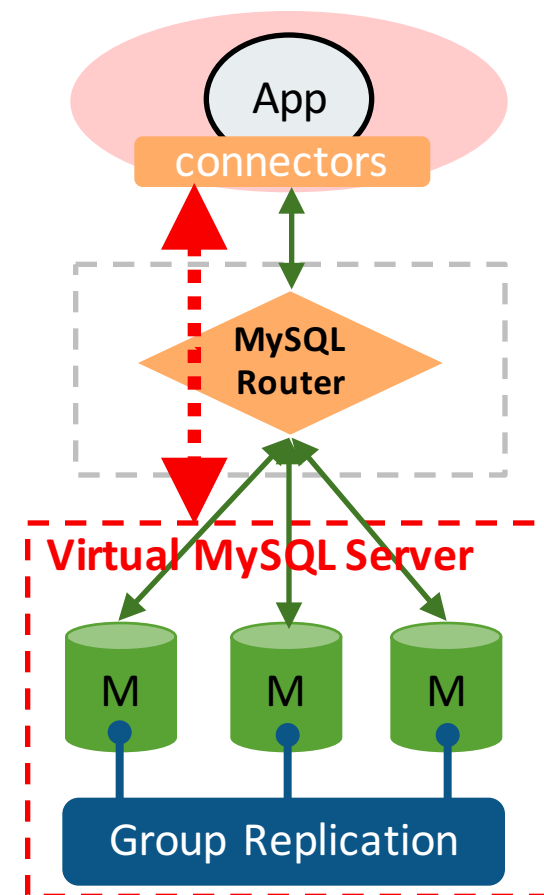
- 是一个 **MySQL plugin**
- 基于 **row format** 和 InnoDB
 - 用 PK 来做冲突检测
- 基于 GTID
- 可通过 performance schema 监控.
- MySQL-5.7.9 加入了 XCom 来替代 corosync
 - 更加容易编译、安装和使用。
 - 支持更多操作系统: Linux, Solaris, FreeBSD, OSX and Windows



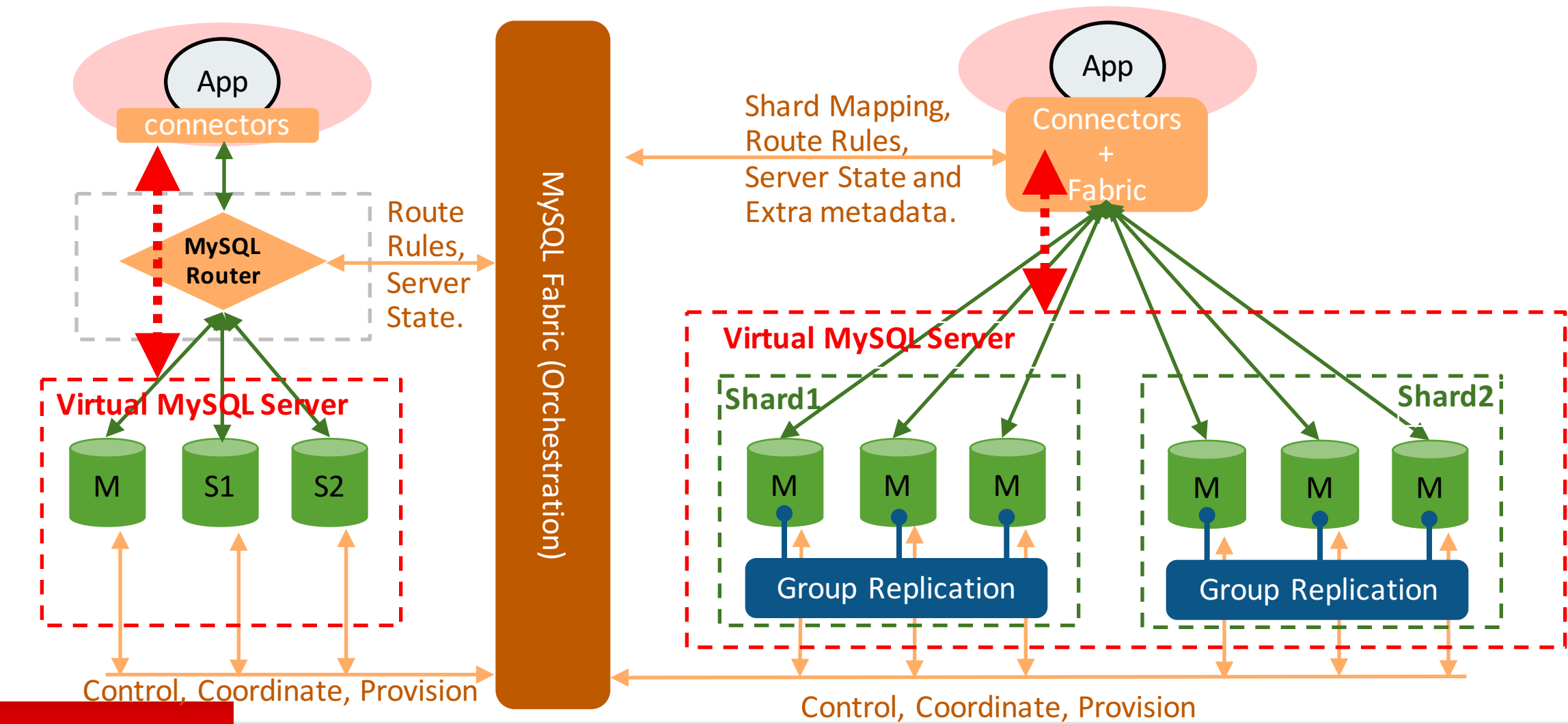
MySQL Router

NEW

- 基于Connection的路由和负载均衡
- 可对读(read-only)和写(read-write)分别路由。
- 平滑的故障切换(fail-over)
- 对用户APP时透明的
 - 不需要升级connectors.
 - 没有connector的语言限制
- 高性能
- 简单易用



管理MySQL大群



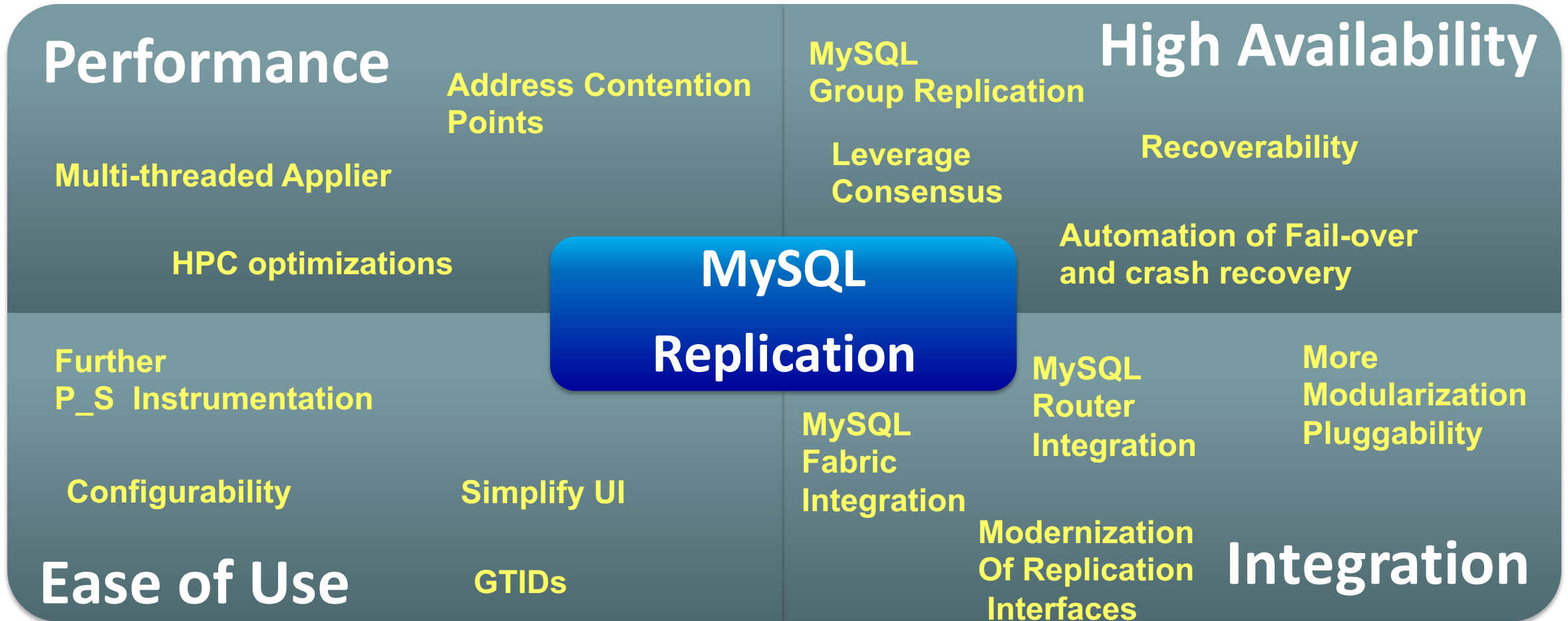
4

复制的开发蓝图

下一步的工作

- **MySQL Group Replication**
 - 快速的发布.
 - 提高性能、稳定性和可用性.
- **MySQL Replication 可用性方面**
 - 将更多的Replication信息添加到P_S表中
 - 简化管理命令，增加更多的在线(on-line)操作.
- **Replication 的性能方面**
 - 持续提高多线程复制的性能.
 - 持续提高复制整个过程各个环节的性能和效率.
- **改进控制中心(Fabric)和MySQL Router之间的交互操作。**

Focus



参考信息

- 安装包
 - <http://dev.mysql.com>
 - <http://labs.mysql.com>
- 手册
 - <http://dev.mysql.com/doc/#manual>
- 开发人员的博客(新闻, 技术信息,其他有趣的内容)
 - <http://mysqlhighavailability.com>

关注微信公众号 获取文档和更新



云和恩墨



恩墨学院



Oracle新闻



Z3 - SQL审核工具提升SQL质量

- 独特的 SQL 视角
 - SQL生命周期管理 - 捕获, 分析, 归档, 形成SQL全周期管理平台;



z3 - SQL审核



zData - 高性能弹性分布式存储解决方案

- 大数据整合与集中面临的平台压力
 - 去“TE”架构, 通过Virtual SAN替代FC SAN
 - 同样的成本获得 20倍+ 的IO性能
 - 动态扩展、高性能的存储解决方案



zData - 分布式存储



Announcing : BayMax 自动化巡检即将开放云服务

- 自动化巡检 - 让DBA去完成那20%最有价值的工作



BayMax自动化巡检

特别感谢

合作伙伴



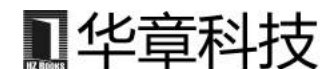
云和恩墨
ENMOTech



Shannon Systems
宝 存 科 技



清华大学出版社





THANKS

