MySQL高可用之MHA的实现及大规模运维实践

黄华亮@京东金融(2014-2015)

DTCC

2015中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2015

大数据技术探索和价值发现

高可用

- **定义：**
  衡量正常运行的百分比。

- **覆盖面：**
  系统
  硬件(磁盘、存储、电源)
  网络
  应用
  缓存
  中间层
  数据库

概述

1.MHA简介

2.项目介绍及wiki

3.MHA架构

4.MHA集中化管理

5.Managre配置

6.切换参数

7.差异Binlog和Relay Log补偿

8.MHA优缺点

9.MHA切换流程

10.对MHA的改进

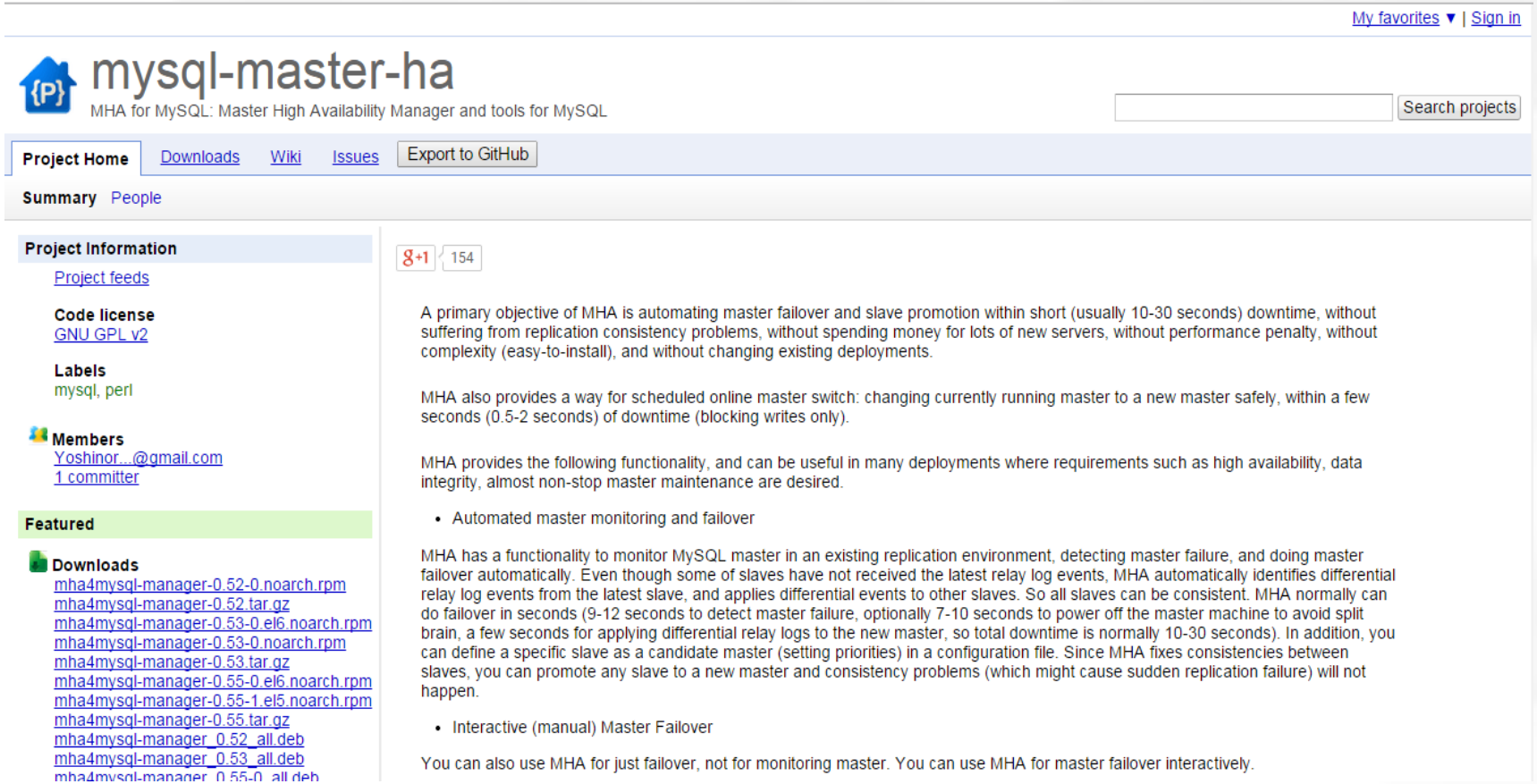
11.我们遇到的问题及解决方法

MHA简介

- 全称： Master High Available Manager
- 开源： https://code.google.com/p/mysql-master-ha/
- 语言： perl
- 关于作者： http://yoshinorimatsunobu.blogspot.jp

地区	日本
介绍	I am a database engineer at Facebook. Before joining Facebook, I was a principal database and infrastructure architect at DeNA. My primary responsibility at DeNA was to make our database infrastructure more reliable, faster and more scalable. Before joining DeNA, I worked at MySQL/Sun/Oracle as a lead MySQL consultant in APAC for four years. You can contact me on Yoshinori.Matsunobu_at_gmail.com (replace _at_ with @).

项目介绍及Wiki



My favorites ▼ | Sign in

mysql-master-ha

MHA for MySQL: Master High Availability Manager and tools for MySQL

Search projects

Project Home | Downloads | Wiki | Issues | Export to GitHub

Summary | People

Project Information

Project feeds

Code license
GNU GPL v2

Labels
mysql, perl

Members
Yoshinor...@gmail.com
1 committer

Featured

Downloads
mha4mysql-manager-0.52-0.noarch.rpm
mha4mysql-manager-0.52.tar.gz
mha4mysql-manager-0.53-0.el6.noarch.rpm
mha4mysql-manager-0.53-0.noarch.rpm
mha4mysql-manager-0.53.tar.gz
mha4mysql-manager-0.55-0.el6.noarch.rpm
mha4mysql-manager-0.55-1.el5.noarch.rpm
mha4mysql-manager-0.55.tar.gz
mha4mysql-manager_0.52_all.deb
mha4mysql-manager_0.53_all.deb
mha4mysql-manager_0.55-0_all.deb

+1 154

A primary objective of MHA is automating master failover and slave promotion within short (usually 10-30 seconds) downtime, without suffering from replication consistency problems, without spending money for lots of new servers, without performance penalty, without complexity (easy-to-install), and without changing existing deployments.

MHA also provides a way for scheduled online master switch: changing currently running master to a new master safely, within a few seconds (0.5-2 seconds) of downtime (blocking writes only).

MHA provides the following functionality, and can be useful in many deployments where requirements such as high availability, data integrity, almost non-stop master maintenance are desired.

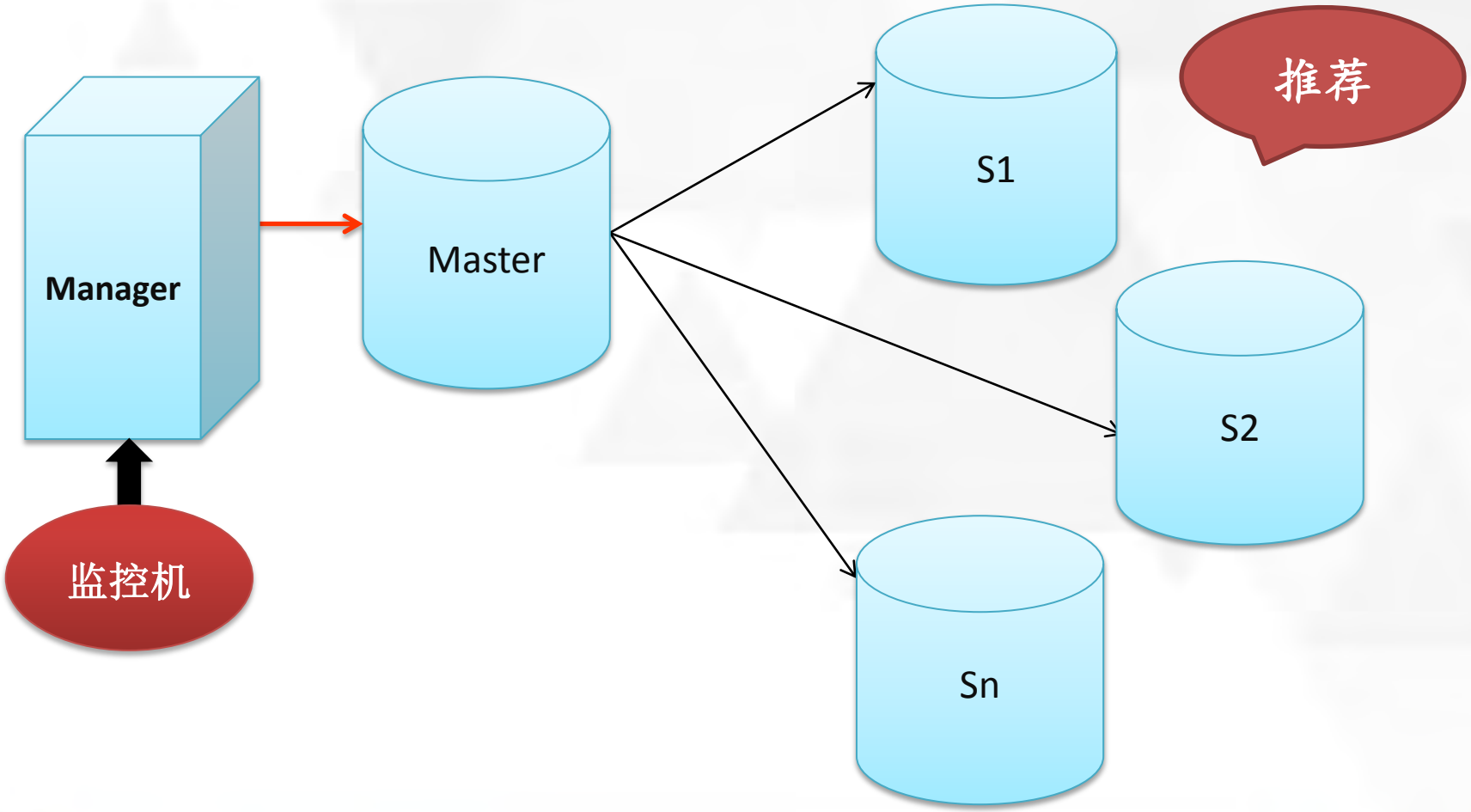
- Automated master monitoring and failover

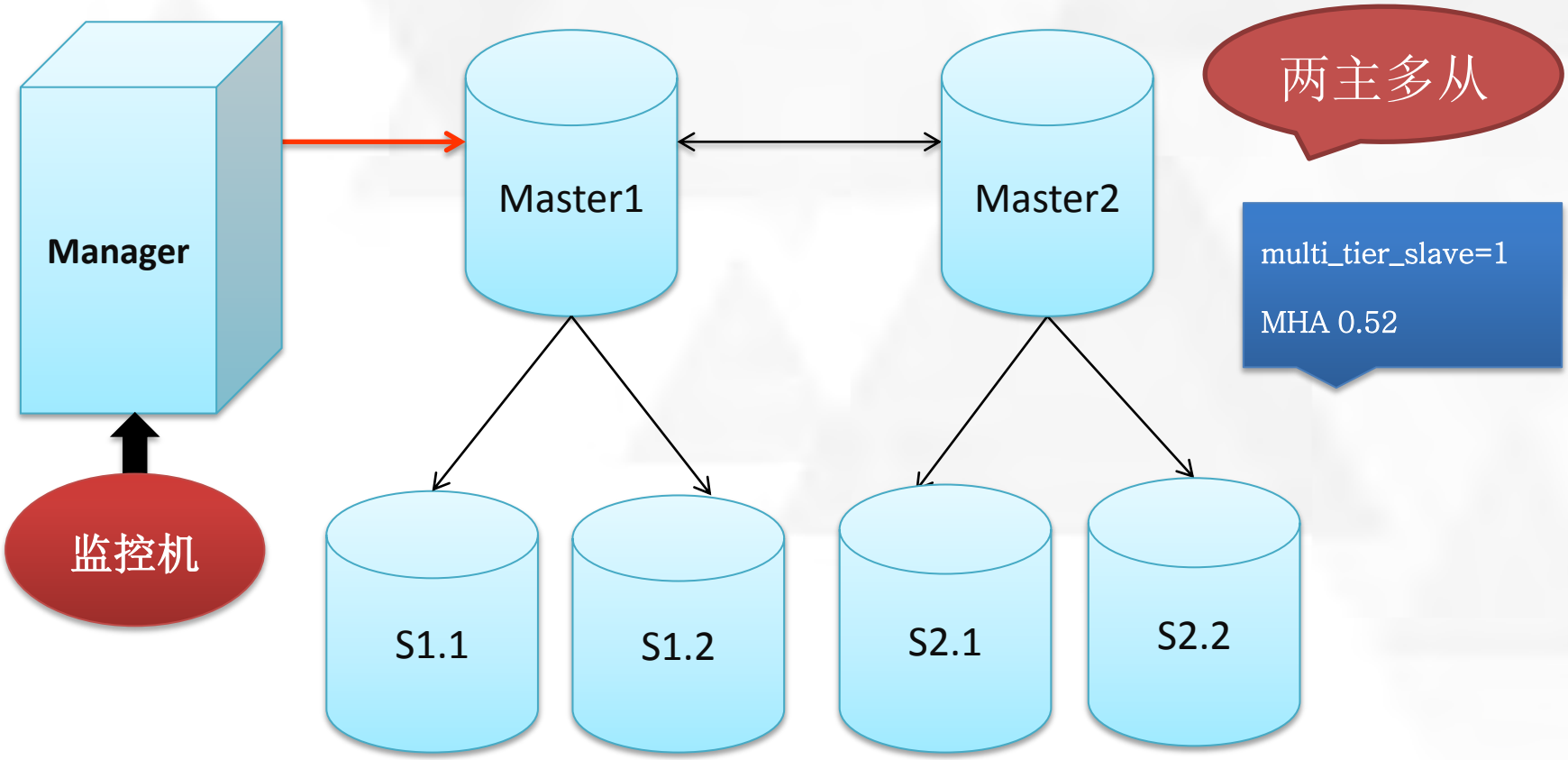
MHA has a functionality to monitor MySQL master in an existing replication environment, detecting master failure, and doing master failover automatically. Even though some of slaves have not received the latest relay log events, MHA automatically identifies differential relay log events from the latest slave, and applies differential events to other slaves. So all slaves can be consistent. MHA normally can do failover in seconds (9-12 seconds to detect master failure, optionally 7-10 seconds to power off the master machine to avoid split brain, a few seconds for applying differential relay logs to the new master, so total downtime is normally 10-30 seconds). In addition, you can define a specific slave as a candidate master (setting priorities) in a configuration file. Since MHA fixes consistencies between slaves, you can promote any slave to a new master and consistency problems (which might cause sudden replication failure) will not happen.

- Interactive (manual) Master Failover

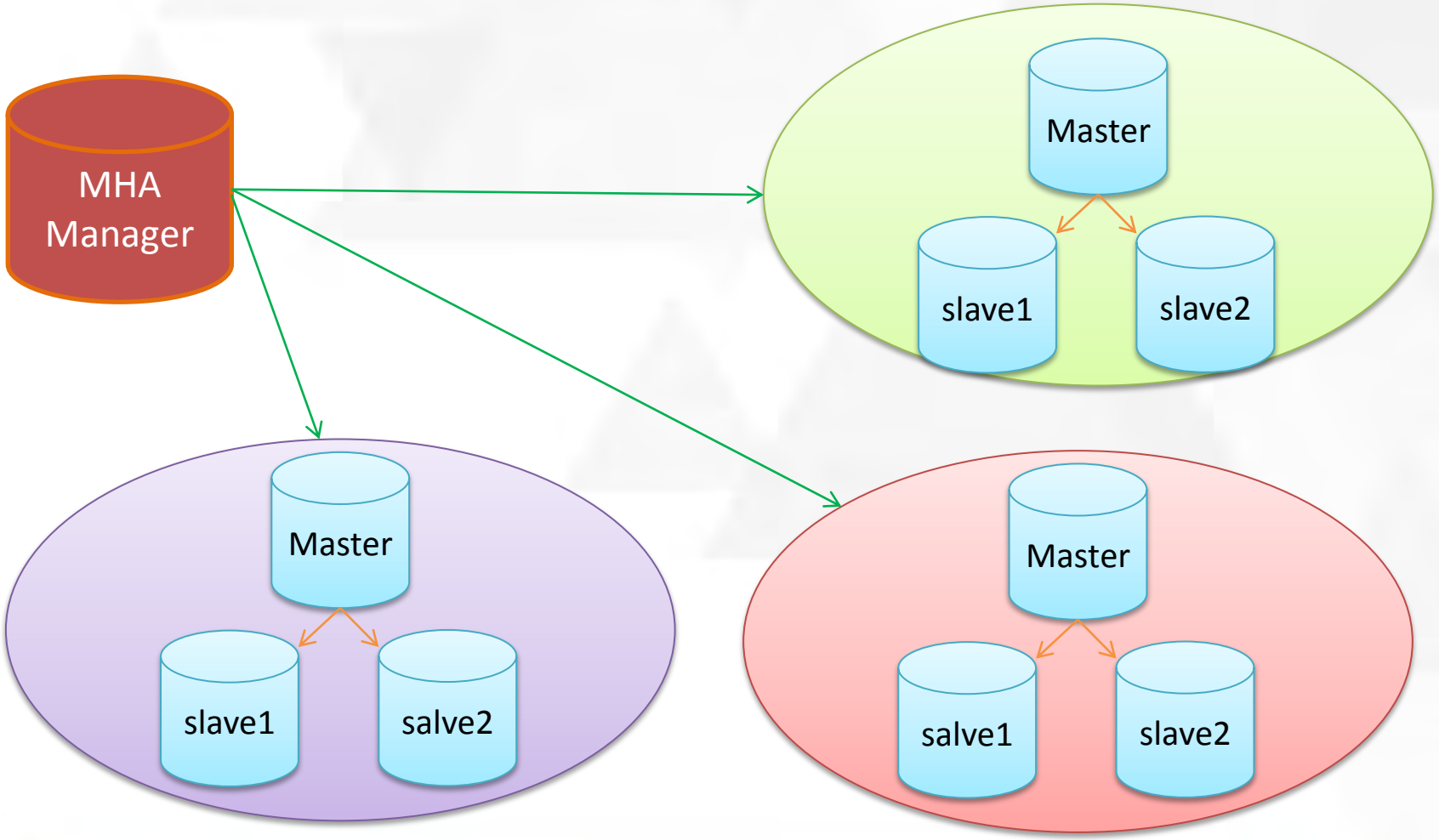
You can also use MHA for just failover, not for monitoring master. You can use MHA for master failover interactively.

MHA架构




Manager
Master1
Master2
两主多从
multi_tier_slave=1
MHA 0.52
监控机
S1.1
S1.2
S2.1
S2.2

MHA组集中化管理


MHA
Manager
Master
slave1
slave2
Master
slave1
salve2
Master
salve1
slave2

Manager配置

■ 重要参数：

配置文件：

```
candidate_master=1
no_master=1
remote_workdir
master_binlog_dir
client_bindir=/usr/local/mysql/bin
client_libdir=/usr/lib/mysql
secondary_check_script
master_ip_failover_script
master_ip_online_change_script
```

```
shutdown_script
report_script
latest_priority
ping_type=select|CONNECT|INSERT
ping_interval
multi_tier_slave
```

■ 切换参数:

masterha_master_switch 命令参数:

```
--master_state=alive|dead
--remove_orig_master_conf
--skip_lock_all_tables
--orig_master_is_new_slave
--running_updates_limit
--running_seconds_limit
```

■常用命令:

masterha_check_ssh

masterha_check_repl

masterha_check_status

masterha_conf_host

masterha_manager

masterha_master_monitor

masterha_master_switch

masterha_secondary_check

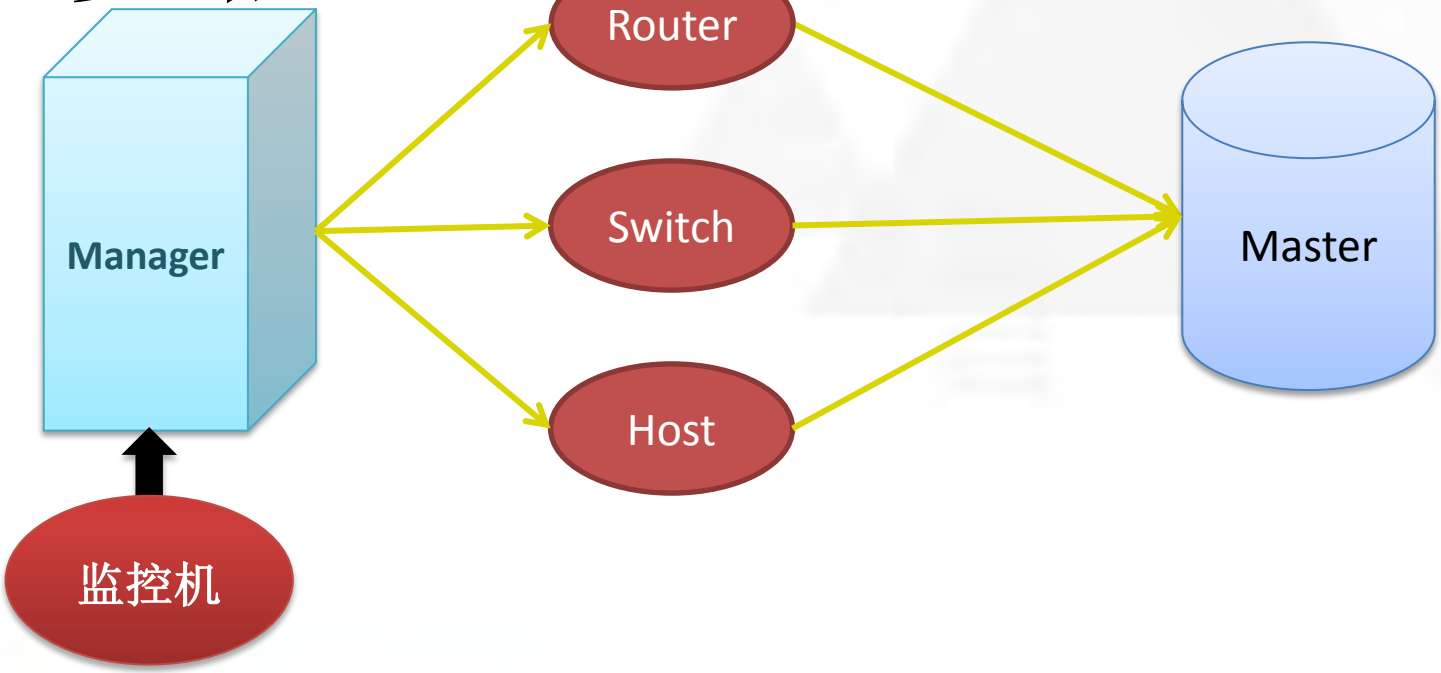
masterha_stop

MHA要点总结

- 至少3个节点
- Manager节点
- Node节点
- Candidate Master
- 心跳(ping|CONNECT|INSERT)
- 多路检测主库
- GTID支持

检测算法

- 单路
- 多路



Binlog补偿和Relay log补偿

New Master(S2)与Master差异Binlog:
Master的
binlog.000010:830 至
binlog.000010:930 之间。

S1与New Master(S2)差异Relay log=
S2的
relaybin.000005:end_log_pos(730)至
relaybin.000005:end_log_pos(830)
之间。

■New Master补偿：

Diff Binlog

■Slave补偿：

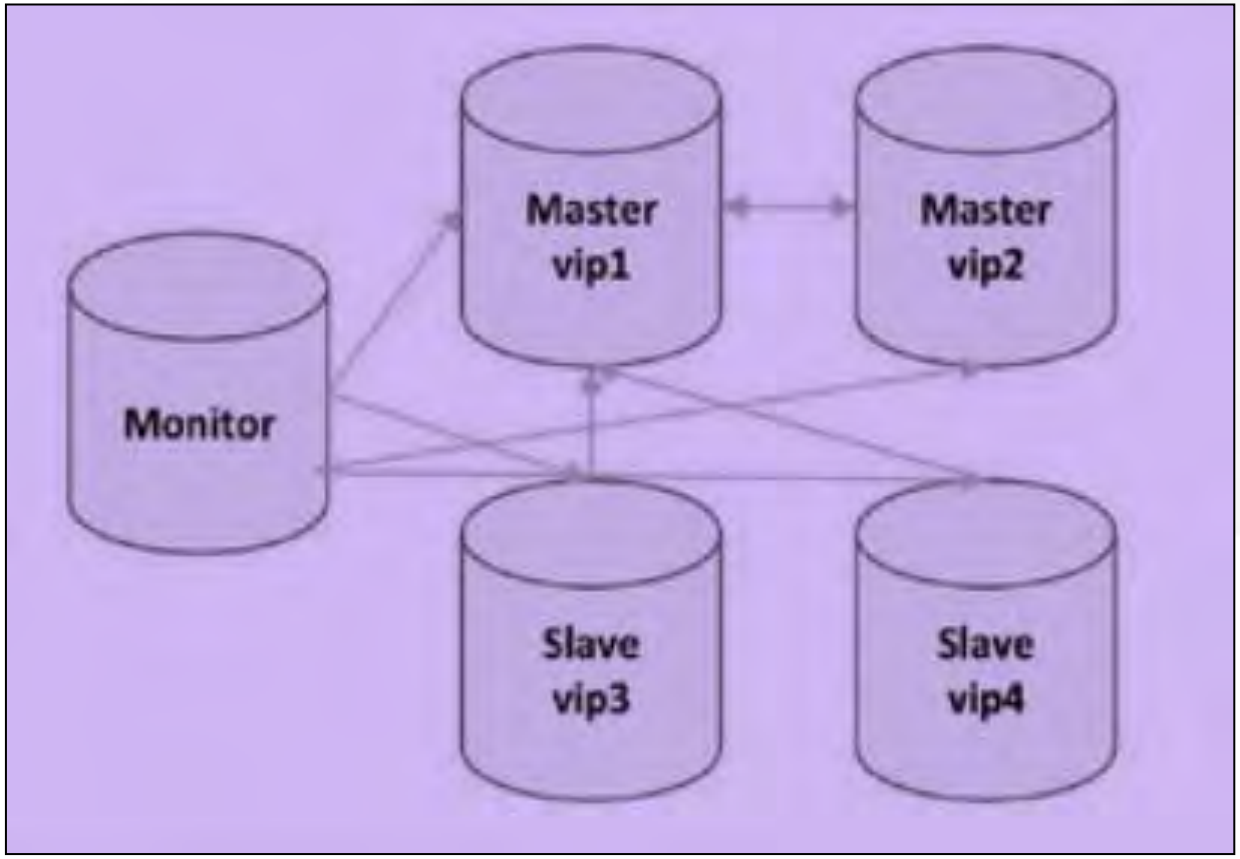
Diff Binlog + Diff Relay Log

补偿差异Relay Log实现

■Candicater Master Relay Log purge:

```
#Candicater master purge relay logs at 5am(MySQL 3306 instance)
0 3,11 * * * /bin/sh /home/mysql/admin/bin/mha_purge_relaylog.sh 3306
```

以往的MySQL HA 方案



1. 数据一致性？
2. 延时导致block？
3. 版本的更新？

MHA与MMM的比较

- 数据丢失
- vip
- 脑裂
- 版本维护
- 公司选择
- 高可用方案

MHA的优缺点

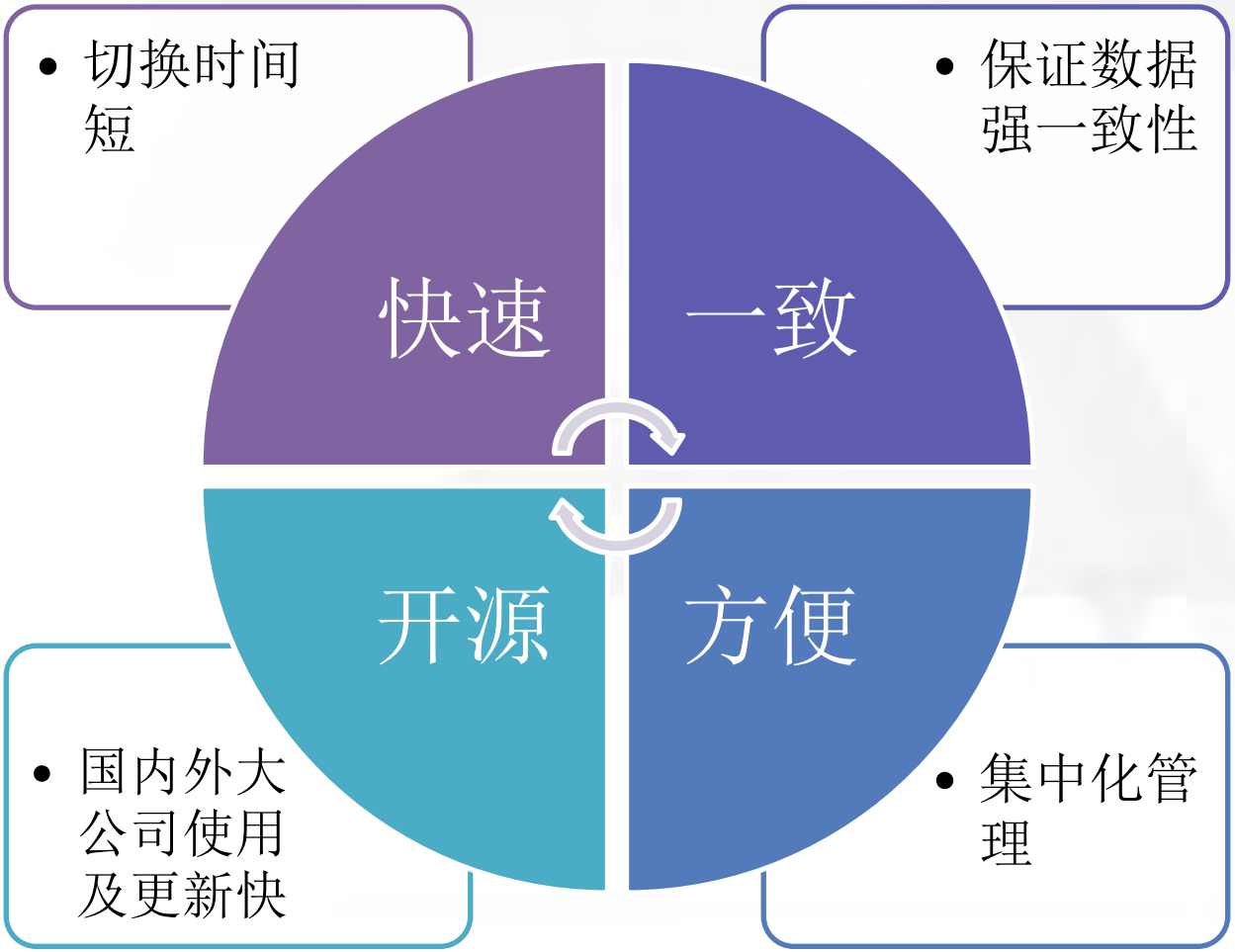
■优点

- 切换时间短
- 前后数据强一致性
- 无脑裂
- 支持多种切换方式
- 支持GTID

■缺点

- 只对Master做了高可用，slave没有
- 依赖SSH及互信
- 接口参数多，配置维护困难

为什么选择MHA



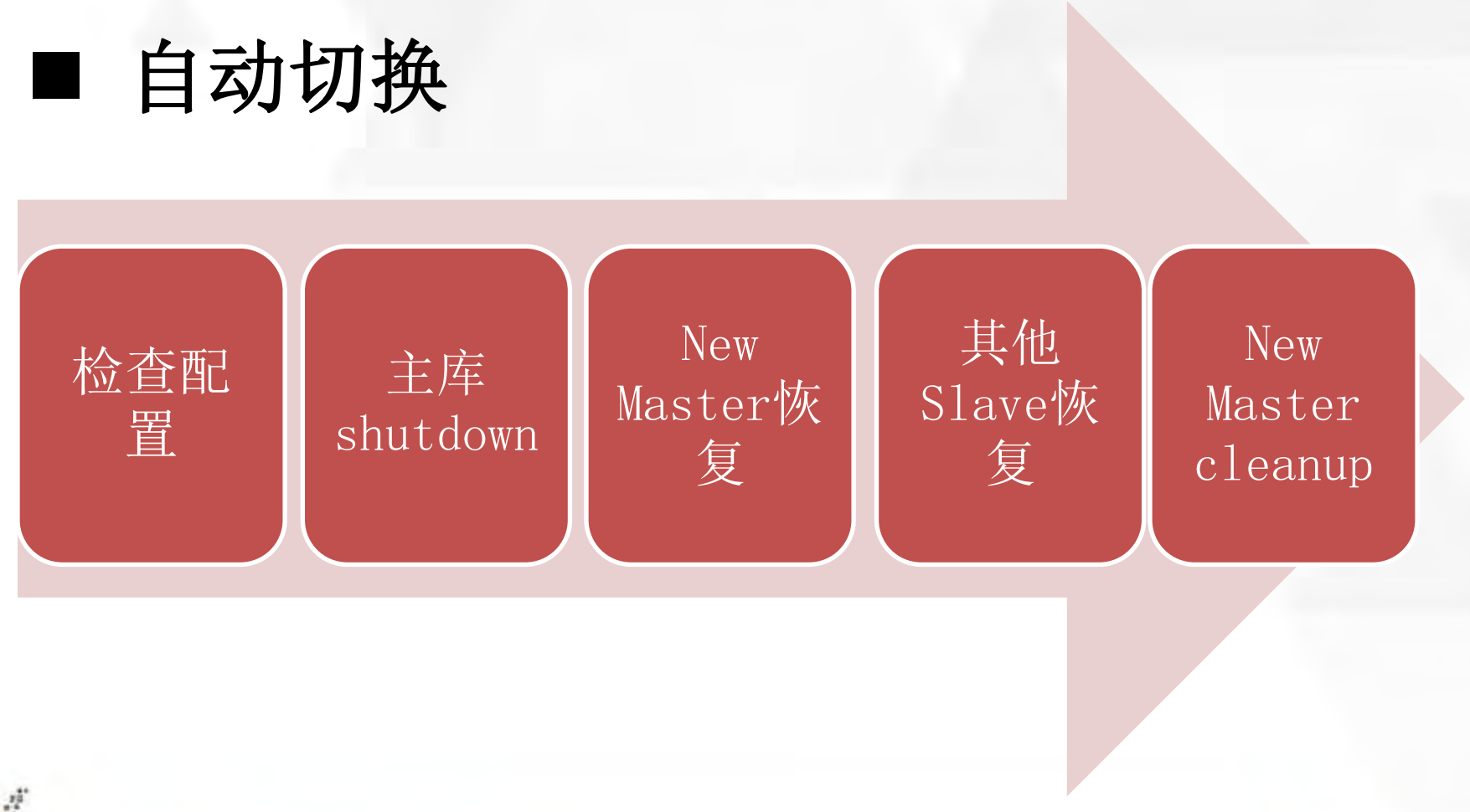
MHA必备条件

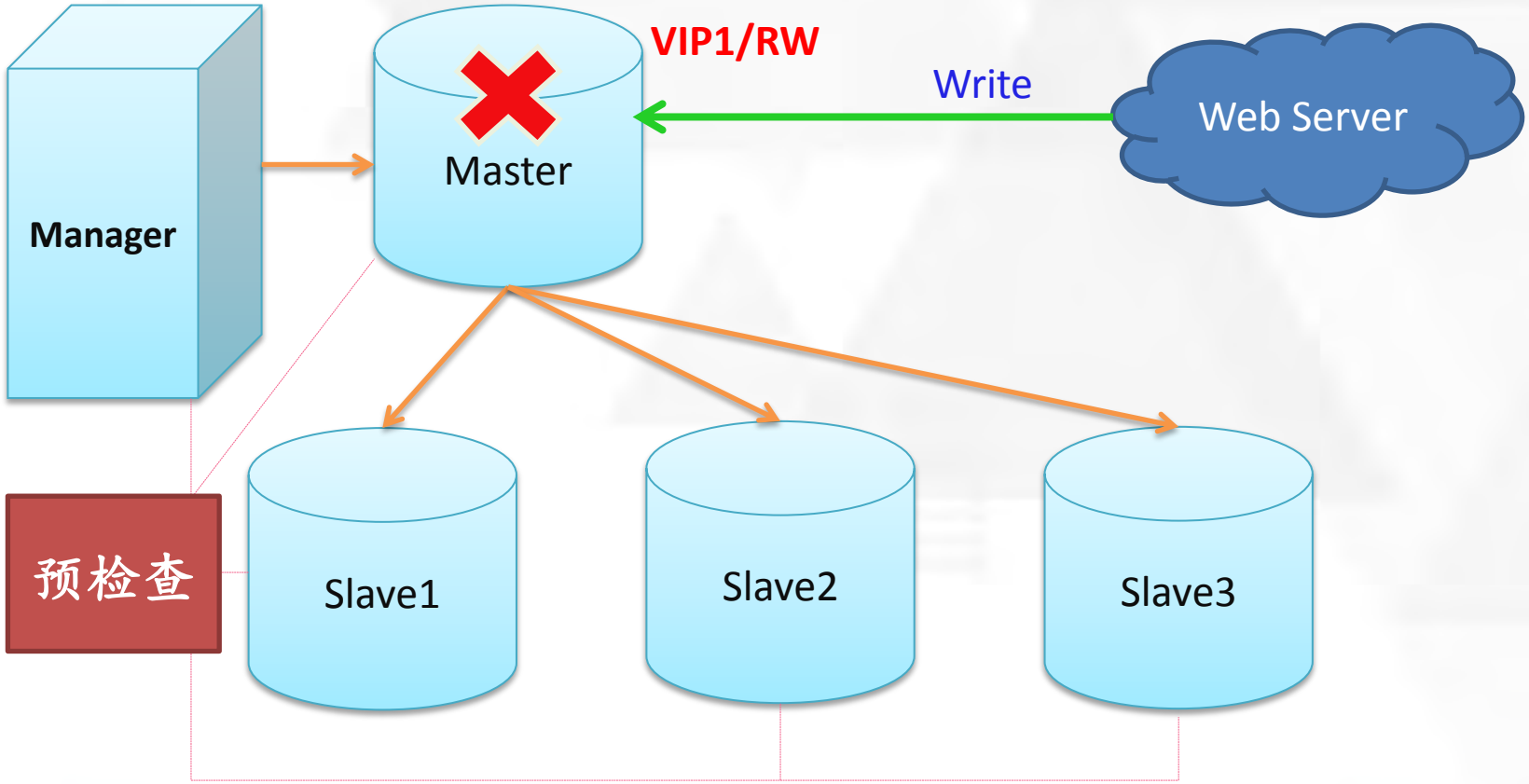
- SSH互信
- OS: Linux only
- 单写
- 复制限制:不支持>=3层复制（0.52版本开始支持）
- MySQL 版本: MySQL 5.0+
- Mysqlbinlog版本>=MySQL版本
- Binlog开启: Master Candidate Masters

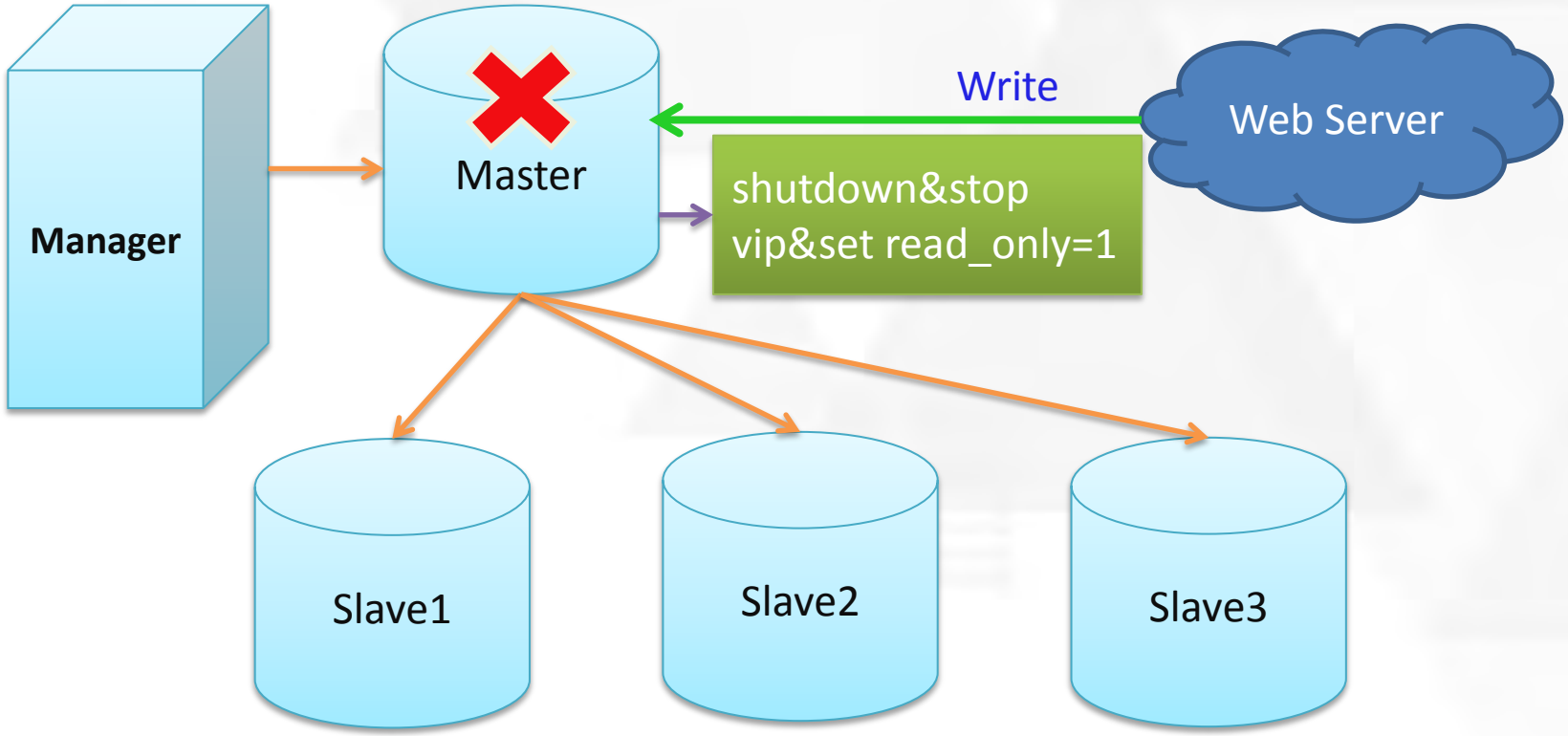
- GTID:0.56支持
- Binlog和relay log过滤规则：所有节点相同
- 复制用户
- Relay Log保留与定期purge
- LOAD DATA:SBR格式不要记录到Binlog

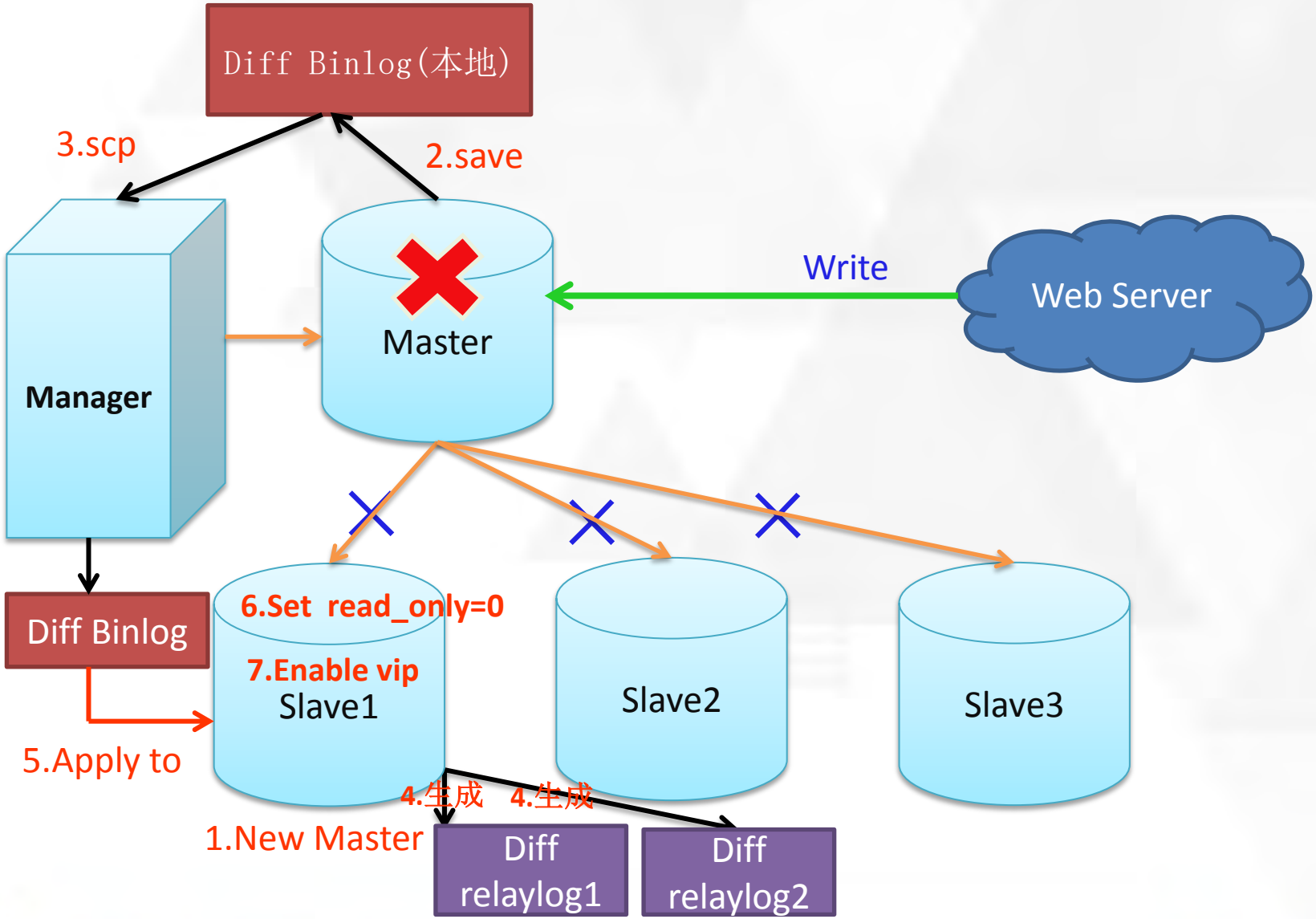
MHA切换流程

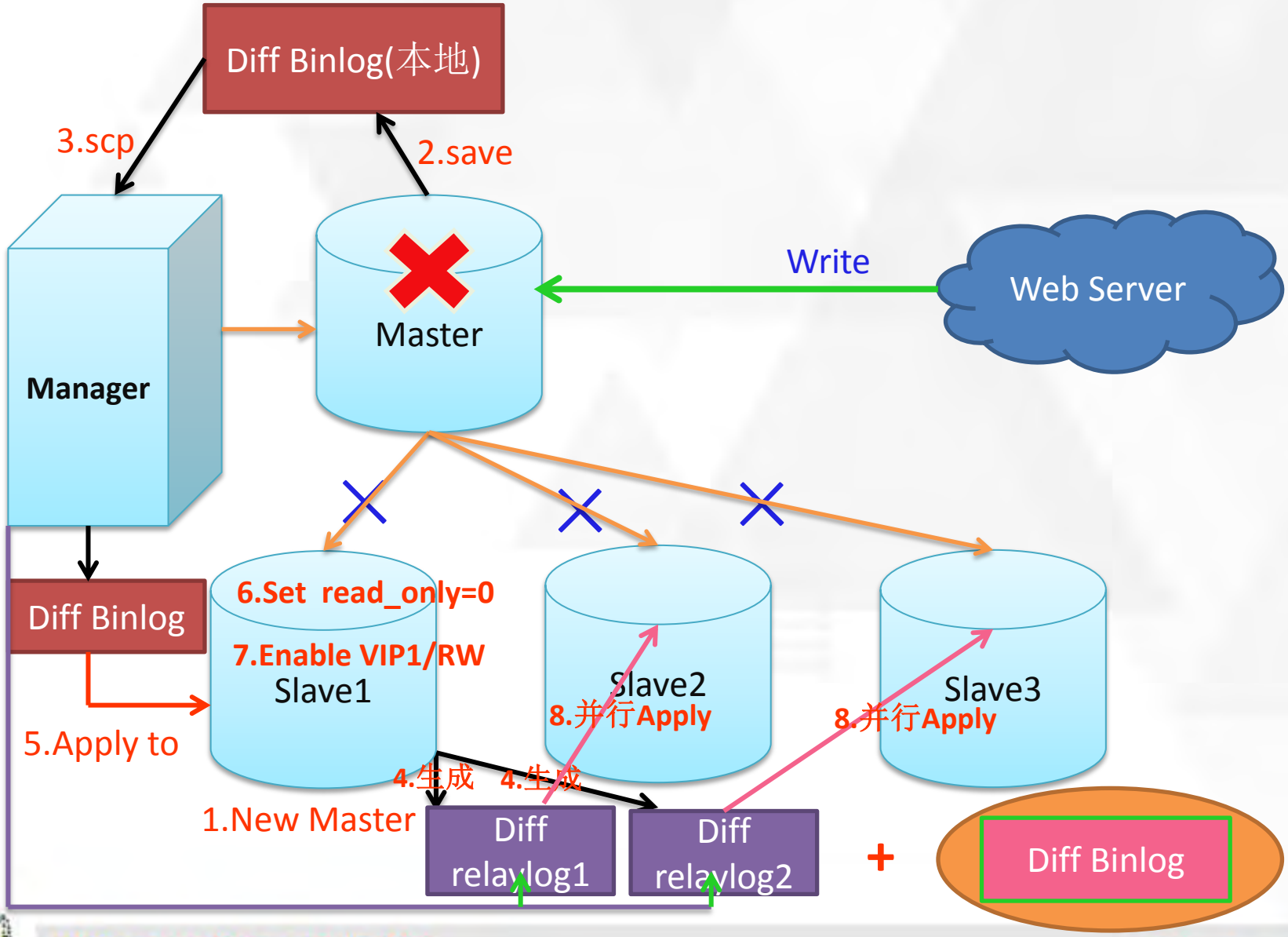
- 自动切换

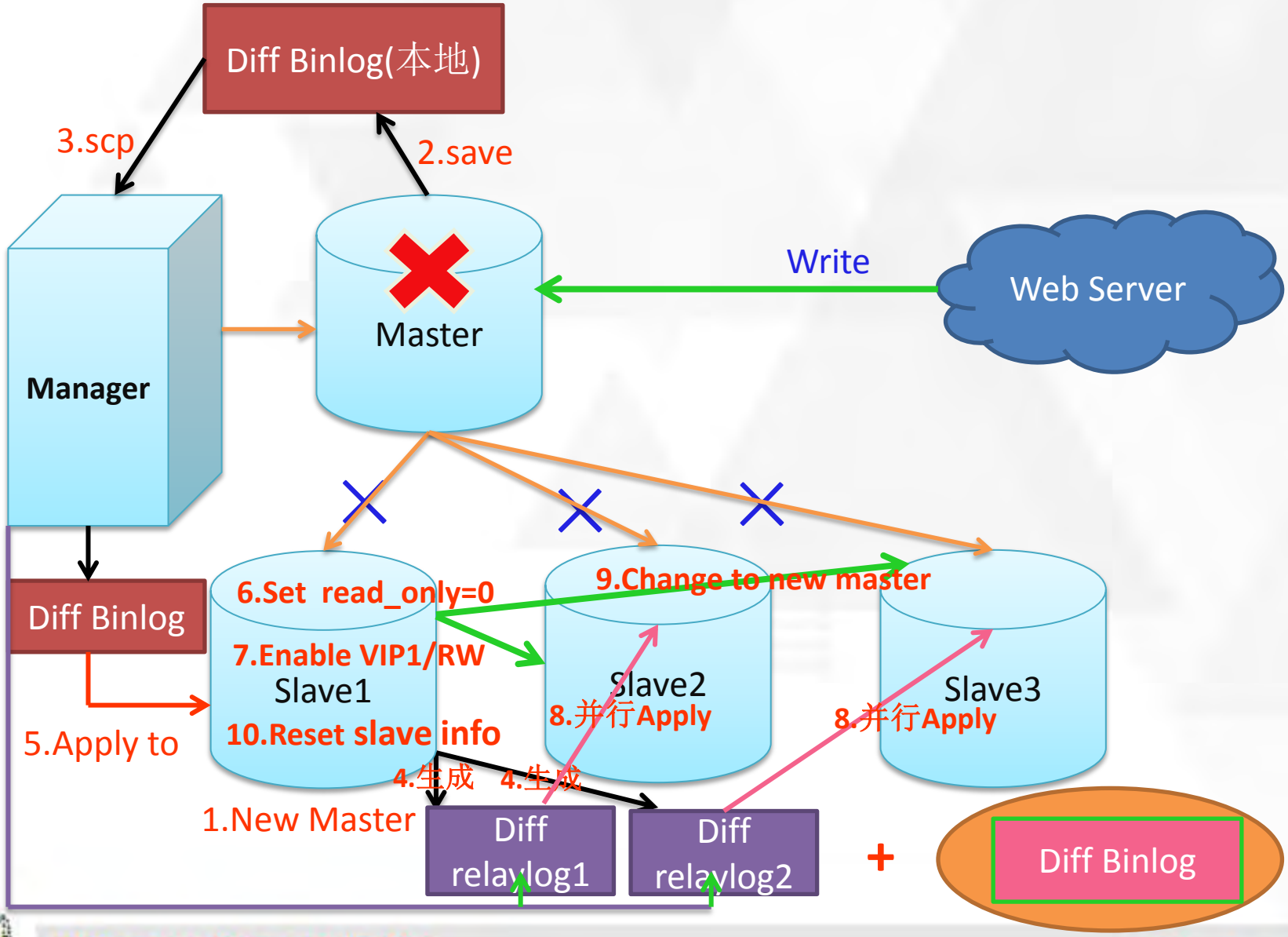



VIP1/RW
Write
Web Server
Master
Manager
预检查
Slave1
Slave2
Slave3

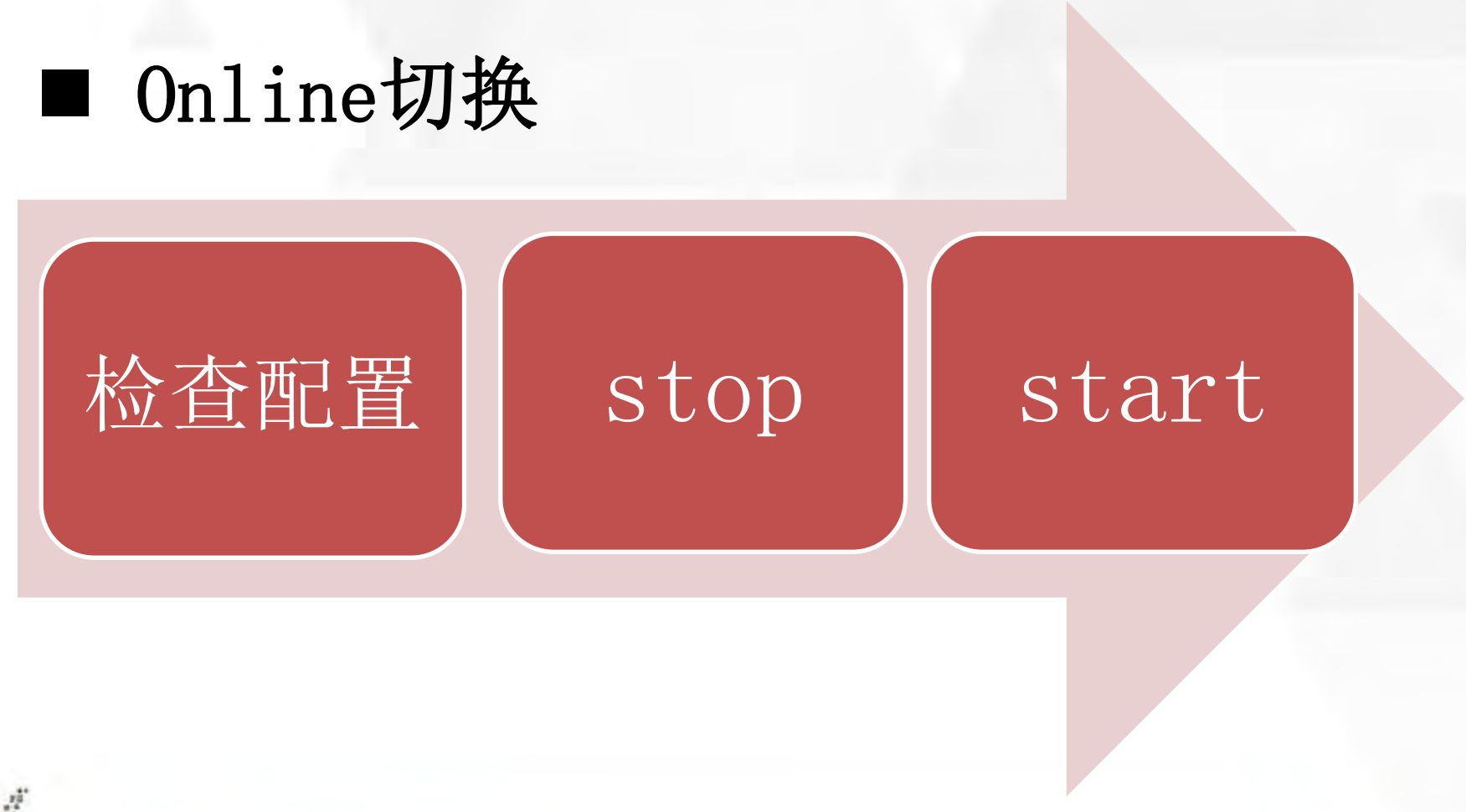

Manager
Master
Write
Web Server
shutdown&stop
vip&set read_only=1
Slave1
Slave2
Slave3


Diff Binlog(本地)
3.scp
2.save
Manager
Master
Write
Web Server
Diff Binlog
6.Set read_only=0
7.Enable vip
Slave1
Slave2
Slave3
5.Apply to
4.生成
4.生成
1.New Master
Diff relaylog1
Diff relaylog2


Diff Binlog(本地)
3.scp
2.save
Manager
Master
Write
Web Server
Diff Binlog
6.Set read_only=0
7.Enable VIP1/RW
Slave1
Slave2
Slave3
8.并行Apply
8.并行Apply
5.Apply to
4.生成
4.生成
1.New Master
Diff relaylog1
Diff relaylog2
+
Diff Binlog


Diff Binlog(本地)
3.scp
2.save
Manager
Master
Write
Web Server
Diff Binlog
6.Set read_only=0
7.Enable VIP1/RW
Slave1
10.Reset slave info
9.Change to new master
Slave2
8.并行Apply
Slave3
8.并行Apply
5.Apply to
4.生成
4.生成
1.New Master
Diff relaylog1
Diff relaylog2
+
Diff Binlog

■ Online切换



■预检查

与自动切换类似

■Stop阶段

1. 在new master上set read_only=1.
2. Wait for N*1000 millseconds等当前naster的连接exit
3. 在当前master库上set read_only=1,阻塞写
4. Wait for M*100 millseconds等当前master事务和查询结束
5. 在当前master stop vip
6. Kill当前master @threads（除了MySQL内部和复制线程）

■start阶段

1. Slave执行change master to new master.
2. 若指定orig_master_is_new_slave,主备互换.
3. 在new master上set read_only=0,打开写.
4. 在new master上start vip.

切换方式

- **自动切换：**

  Manager节点管理多组MHA MySQL主从，每一组一份配置文件，启动一个监控进程。

- **在线切换：**

  主库Running时热切换。

- **手动切换：**

  web页面或命令行调用masterha_switch。

我们做了什么

- Failover脑裂问题

- vip管理

  脚本控制，立即生效

- 切换前预检查

  ssh链路、VIP、配置、DB等

- 回滚方案
- 主库机器宕机切换
- 部署维护平台化

  一键初始化、配置文件自动生成及切换后自动更新

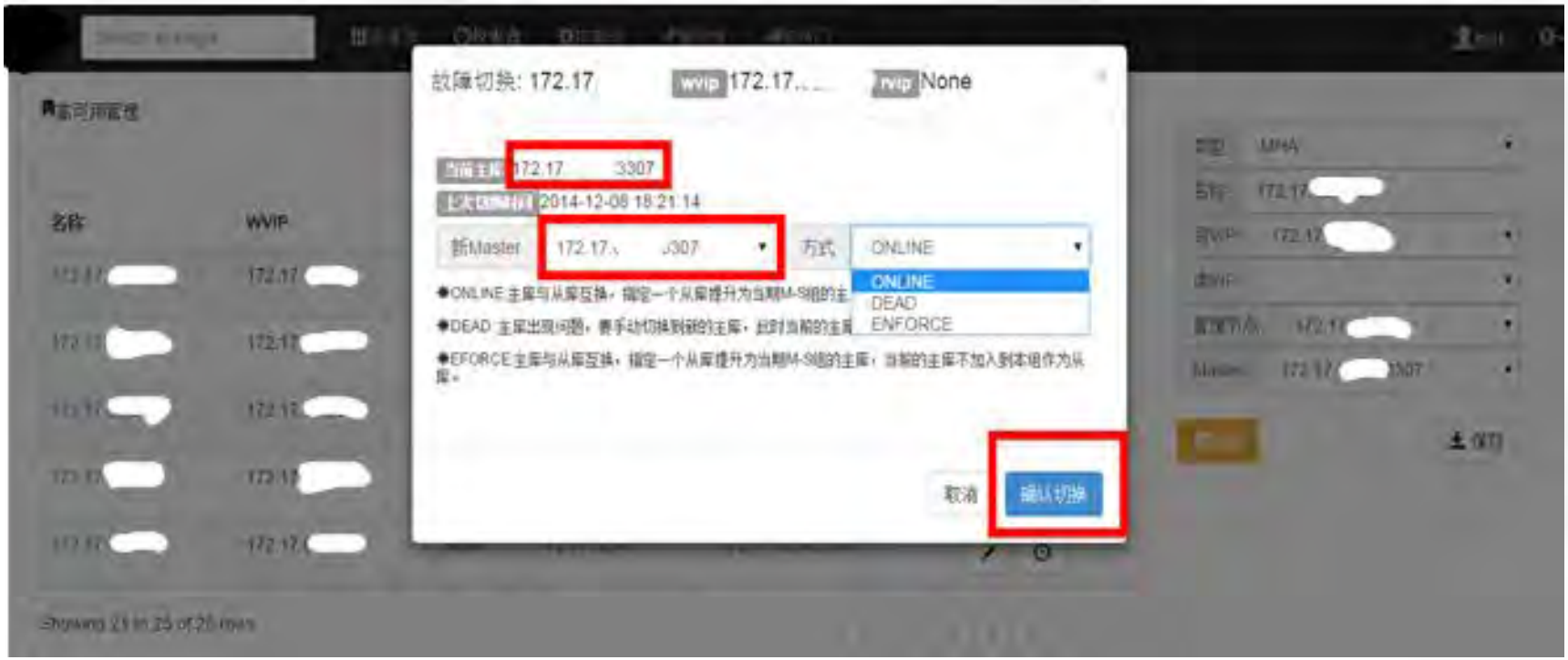
- GTID切换

  有写入非当前主库的Binlog的slave不选举为New Master

- 平台化管理

DB运维管理平台中运维MHA

MHA页面切换


故障切换: 172.17
wvip 172.17.
rvip None
172.17 3307
2014-12-08 18:21:14
新Master
172.17. 3307
方式
ONLINE
ONLINE
DEAD
ENFORCE
取消
确认切换
名称
WVIP


切换信息
当前主库：172.17
切换历史
方式 时间 旧主库 新主库 结果
ONLINE 2014-12-05 11:09:06 172.17 3307 172.17 3307 succes
ONLINE 2014-12-05 11:08:32 172.17 3306 172.17 3506 failed
DEAD 2014-12-04 11:59:59 172.17 3307 172.17 3307 succes

切换日志

模块	时间	耗时	结果	内容
mha	2014-12-05 11:09:07	0	success	1)Configuration Check completed
mha	2014-12-05 11:09:08	0	success	2.1) Set read_only=1 on the new master success
mha	2014-12-05 11:09:11	1	success	2.2)Disabling the VIP on old master success
mha	2014-12-05 11:09:12	0	success	2.3)Killing all application threads on old master success
mha	2014-12-05 11:09:07	5	success	2.4)Executing master_ip_online_change_script to disable write on the current master success
mha	2014-12-05 11:09:12	0	success	2.5)Set read_only=0 on the new master success
mha	2014-12-05 11:09:12	4	success	2.6)Enabling the VIP - 172.17.62.203 on the new master - 172.17.62.42 success
mha	2014-12-05 11:09:16	1	success	2.7)All new slave servers switched successfully
mha	2014-12-05 11:09:17	0	success	5)New master cleanup success
mha	2014-12-05 11:09:07	10	success	6)Switching to new master 172.17.62.42:3307 success

目前生成库MySQL 有40多组主从实例使用MHA并加入自动化运维平台管理，经过多次线上故障切换及切换演练，已经很稳定。

目前面临的问题

- Master主机故障如何补偿差异的Binlog
- MySQL主从版本不一致

■Manger单节点

多路检测并不是完美的解决方法，可以实现仲裁(哨兵)机制更好。

■ 依赖SSH

需要开发Agent组件来替代SSH。

- 接口参数过多
- 切换“中途”失败完全回滚困难

Q&A