MySQL High Availability

MMM & MHA in DP

卢钧轶

Agenda

- Why we need a HA solution
- MMM in DP
- MHA in DP

When We Need to Switch Server

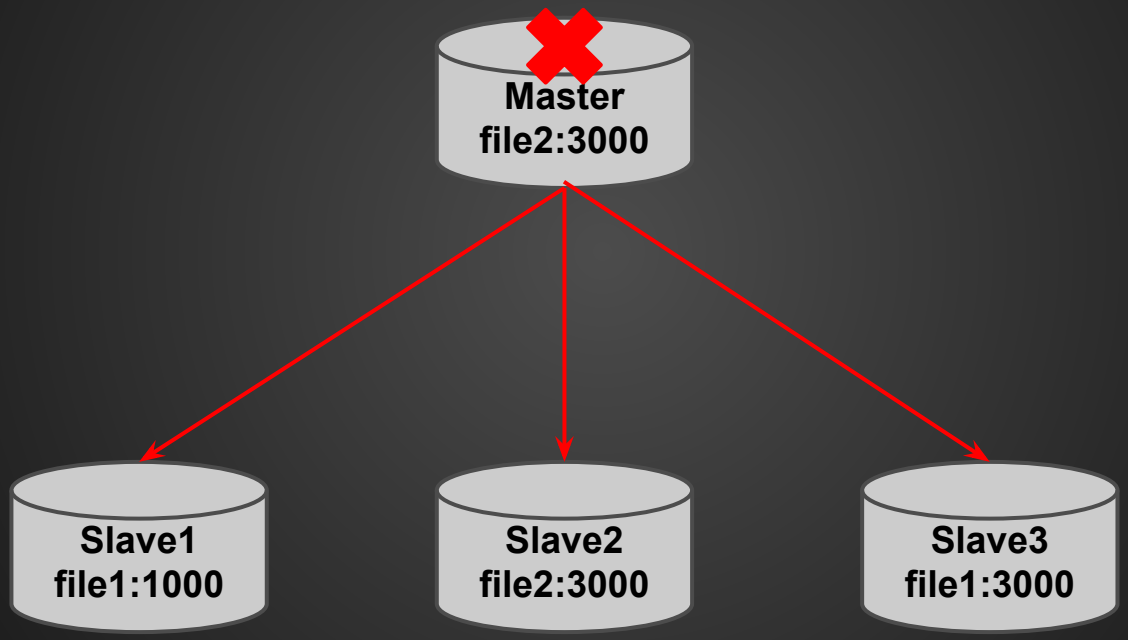
- Master Server down
- Master Server maintenance
  - Logical ： DDL on big table
  - Physical: memory replacement

Why We Need A Failover Solution

Failover without tool:

- Notify DBA for MySQL Failure
- Kill old Master
- Find the latest Slave as Master Candidate
- Execute change Master to on each Slave
- Direct APP to new Master IP & restart

Difficulties For MySQL Failover



MySQL HA options

Stand-by

- MMM
- MHA + VIP
- DRDB + (heartbeat,keepalive)

Proxy

- MHA + HAProxy

Cluster

- MySQL Cluster
- Galera

Why We Chose MMM in 2010

	MMM	MHA
Entire HA solution	Yes	only Master failover
VIP	Built-in support	Needs user defined script
CLI interface	All included in mmm_control	Several Perl script
Deployment	CPAN+Monitor+Agent	CPAN+Manager+Agent + SSH authorization

MMM in DP

Used in Production since 2010

15+ M-M / M-M-S Clusters

Modifications in Source Code

https://github.com/cenalulu/mysql-mmm

MMM

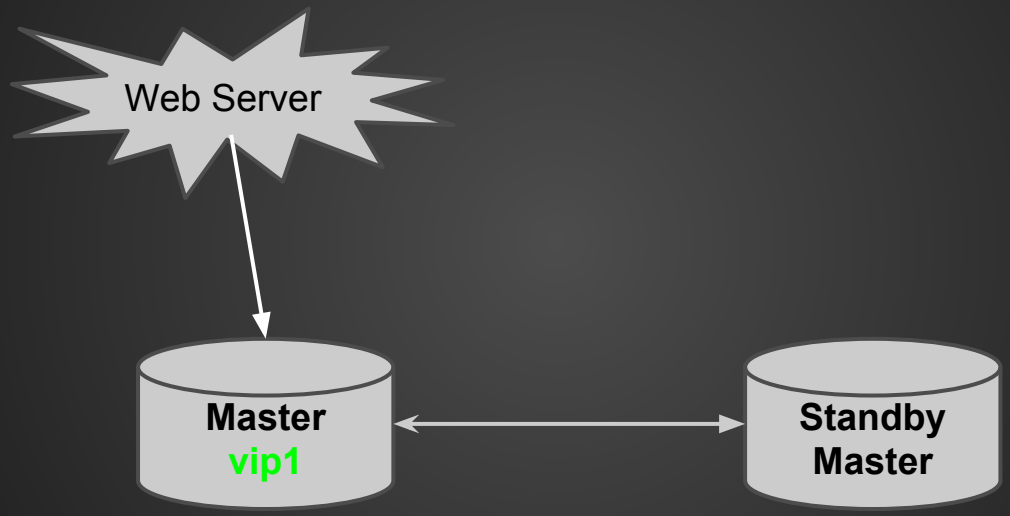
Is

- VIP based HA solution
- Message between Monitor & Agent
- Auto Failover for both M/S

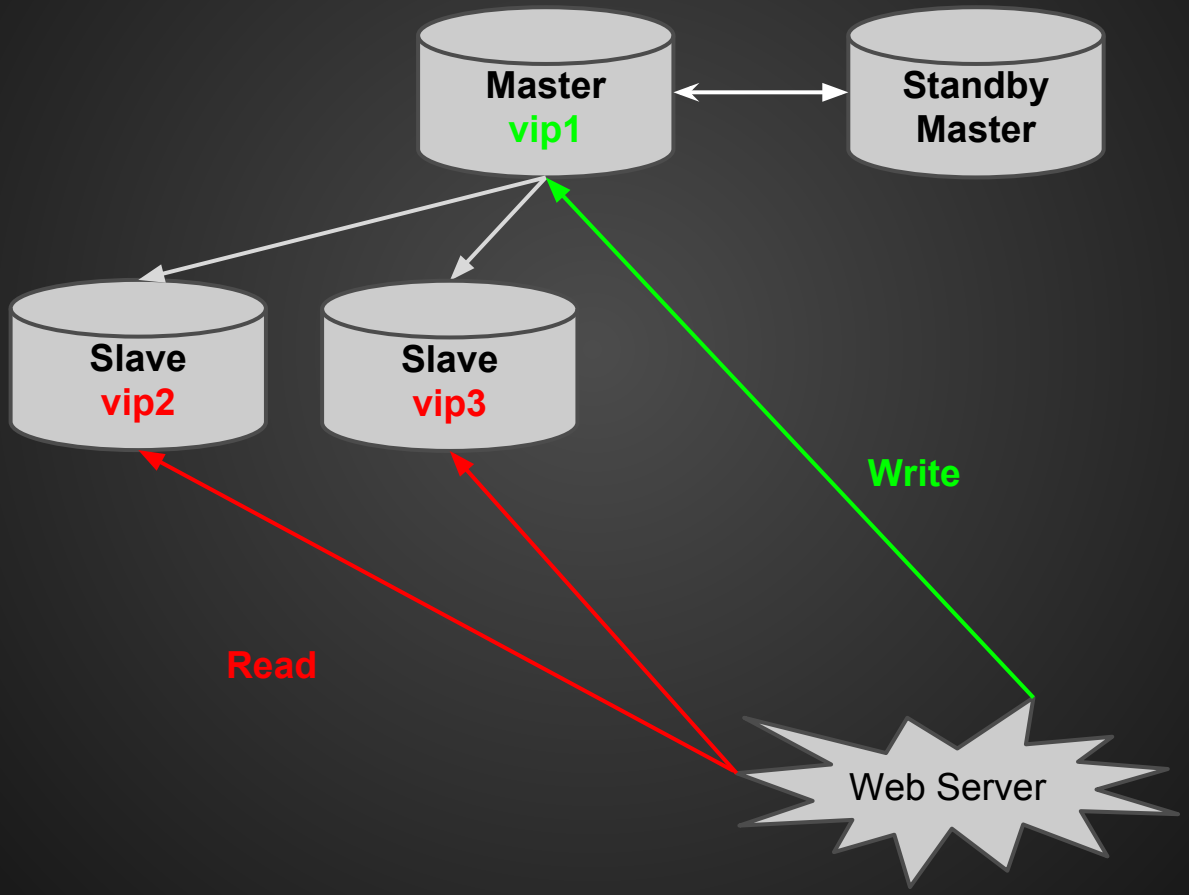
Is Not:

- SQL router
- Load Balancer
- Active (latest code release in 2010-05-07)

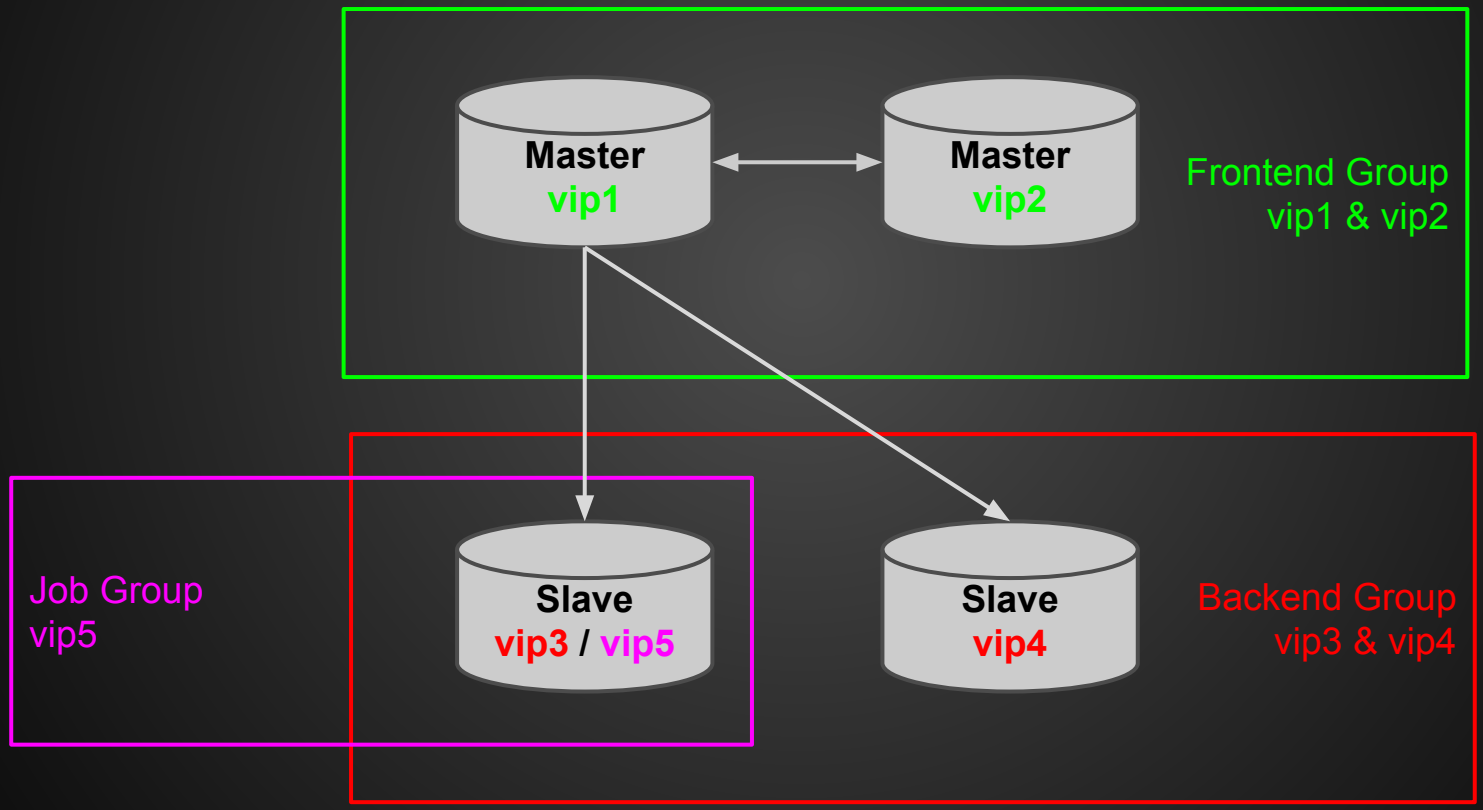
Simple Usage



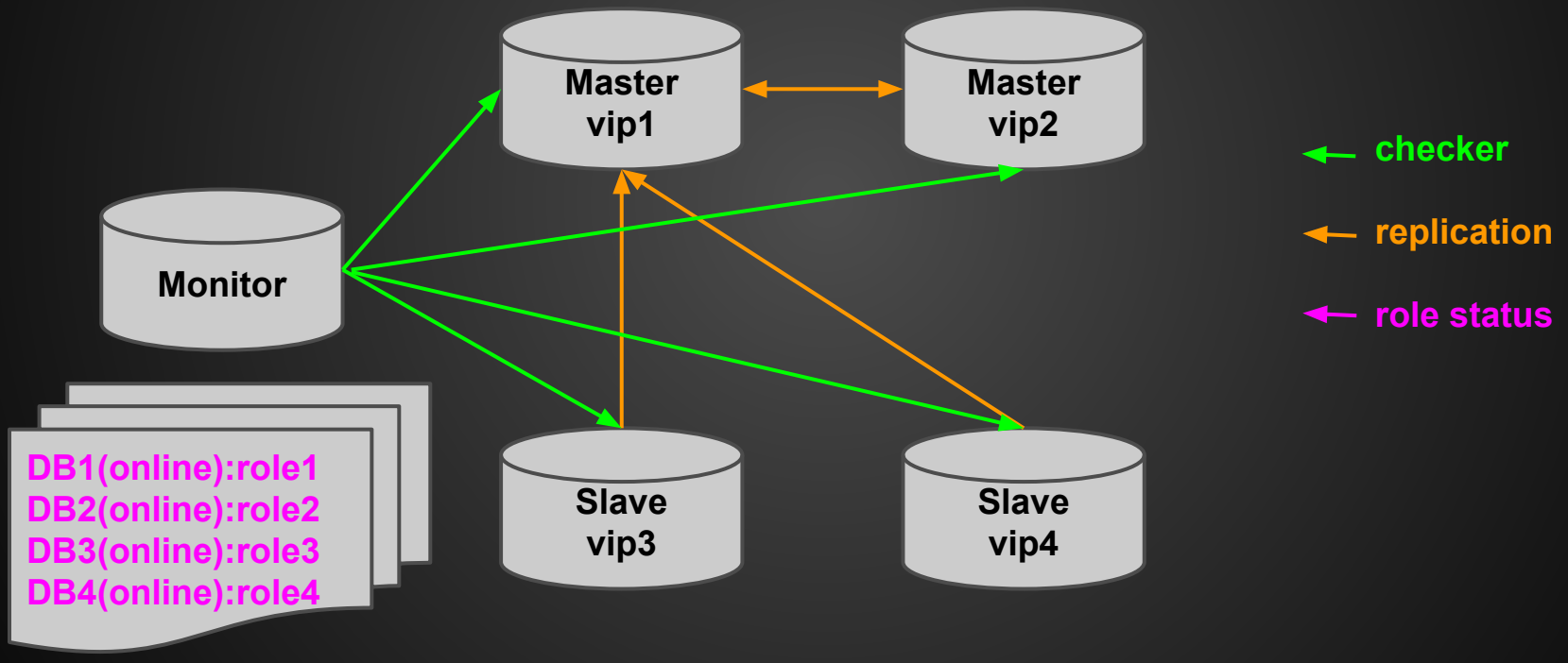
Basic Usage


Master
vip1
Standby
Master
Slave
vip2
Slave
vip3
Write
Read
Web Server

Role Based Machine Group



MMM architecture

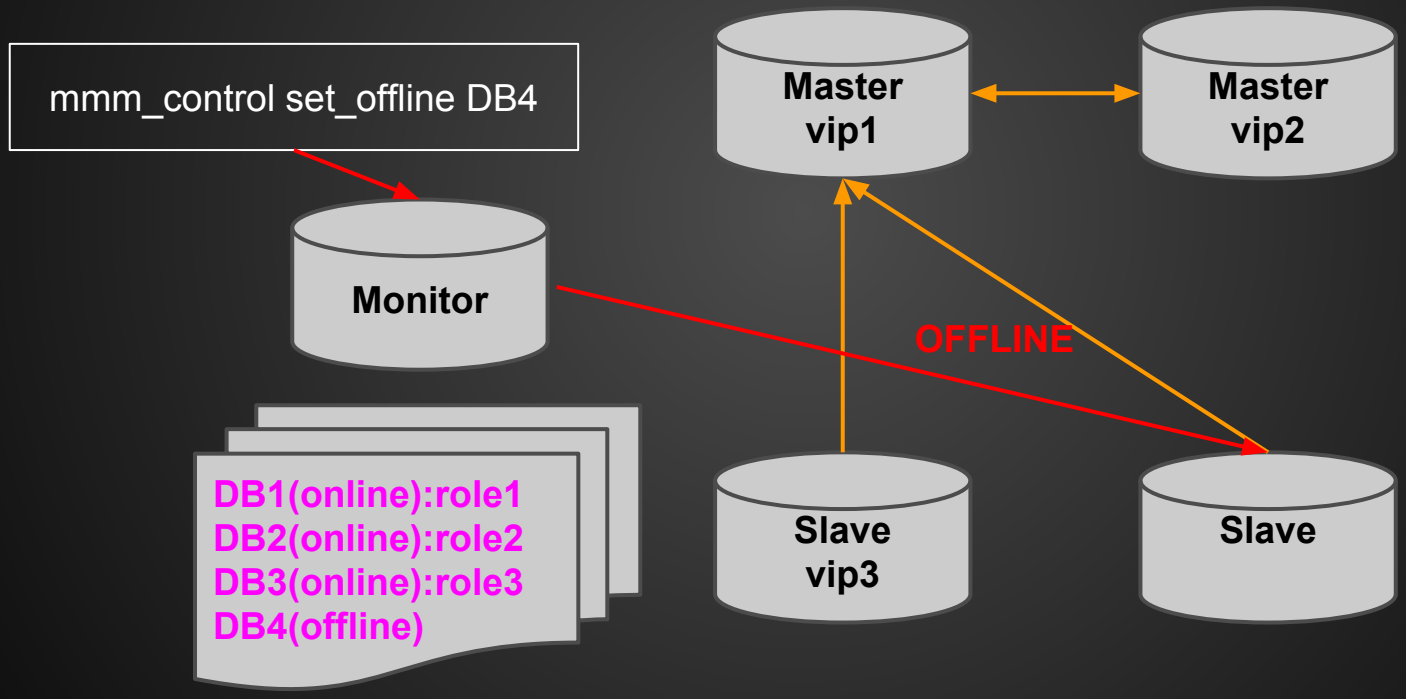


MMM Monitor Logic

```
while (!$main::shutdown) {
        $self->_process_check_results();
        $self->_check_host_states();
        $self->_process_commands();
        $self->_distribute_roles();
        $self->send_status_to_agents();

        # sleep 3 seconds, wake up if command queue gets filled
        lock($command_queue);
        cond_timedwait($command_queue, time() + 3);
}
```

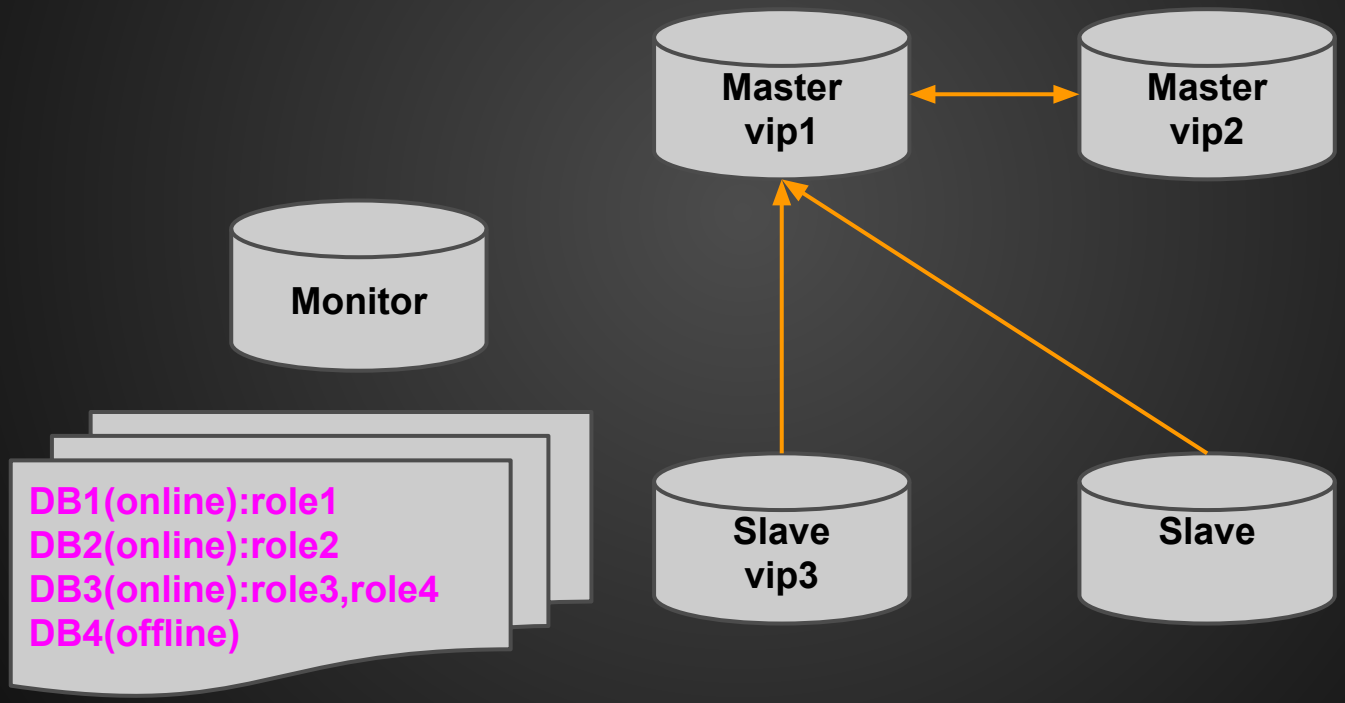
MMM Slave Switch

process_command



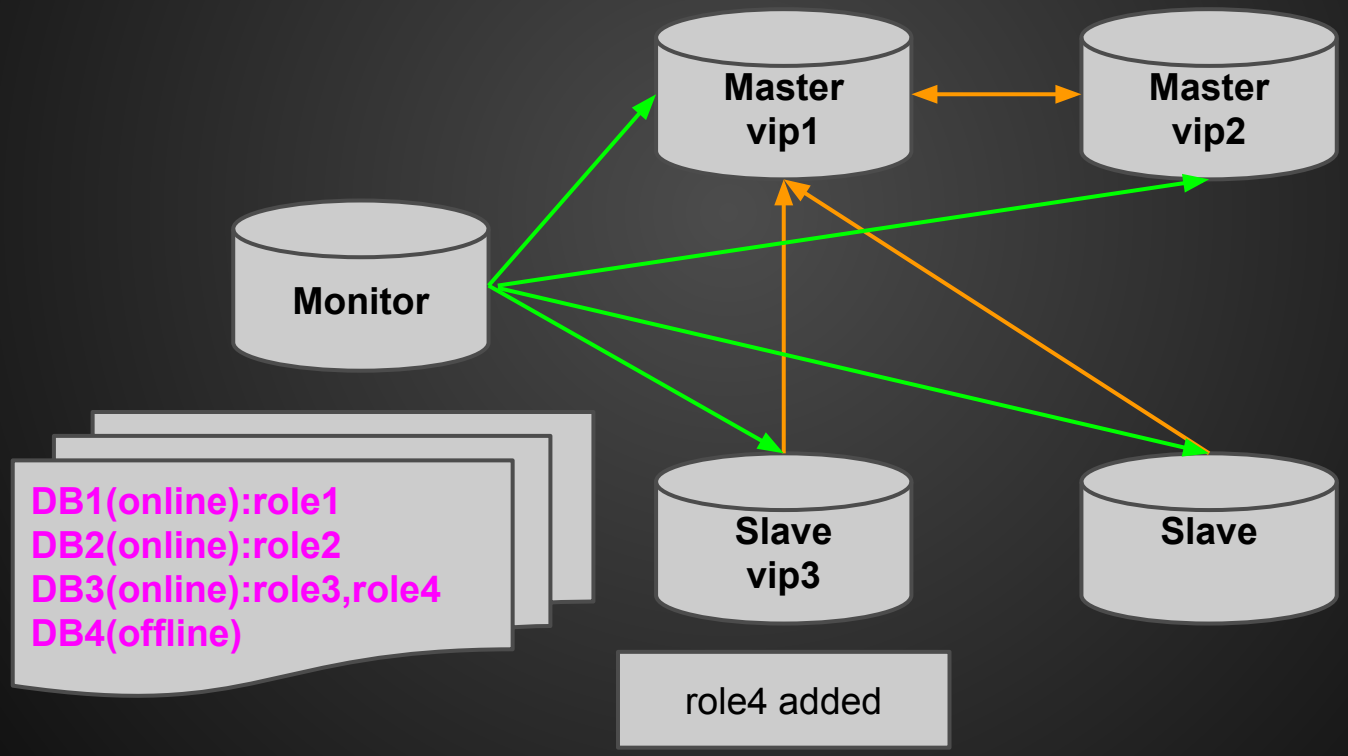
MMM Slave Switch

distributed_roles

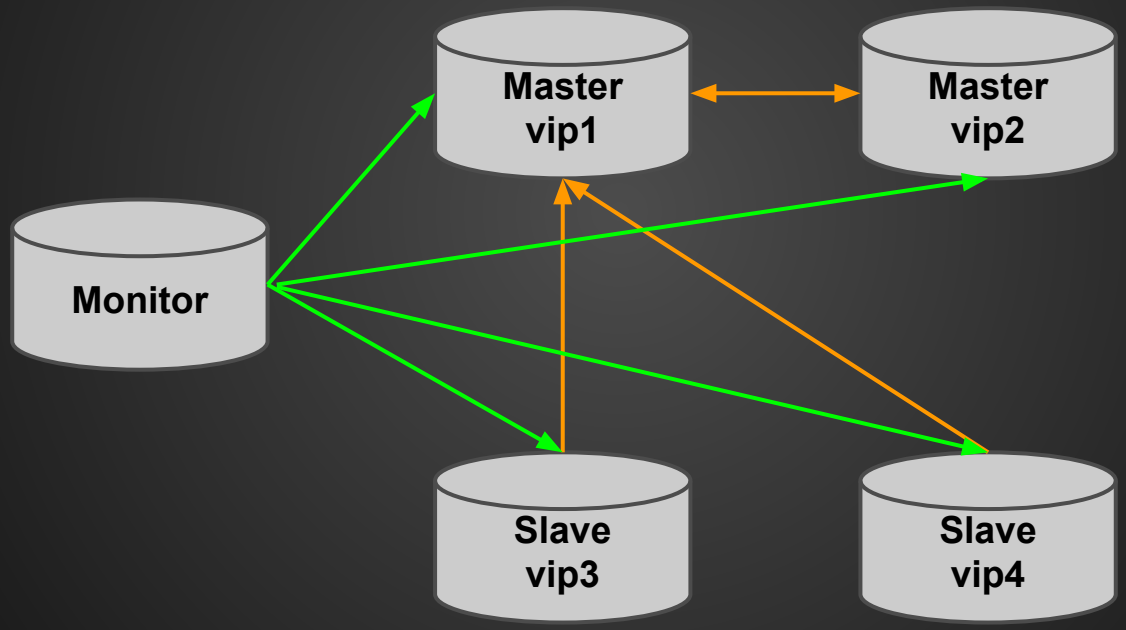


MMM Slave Switch

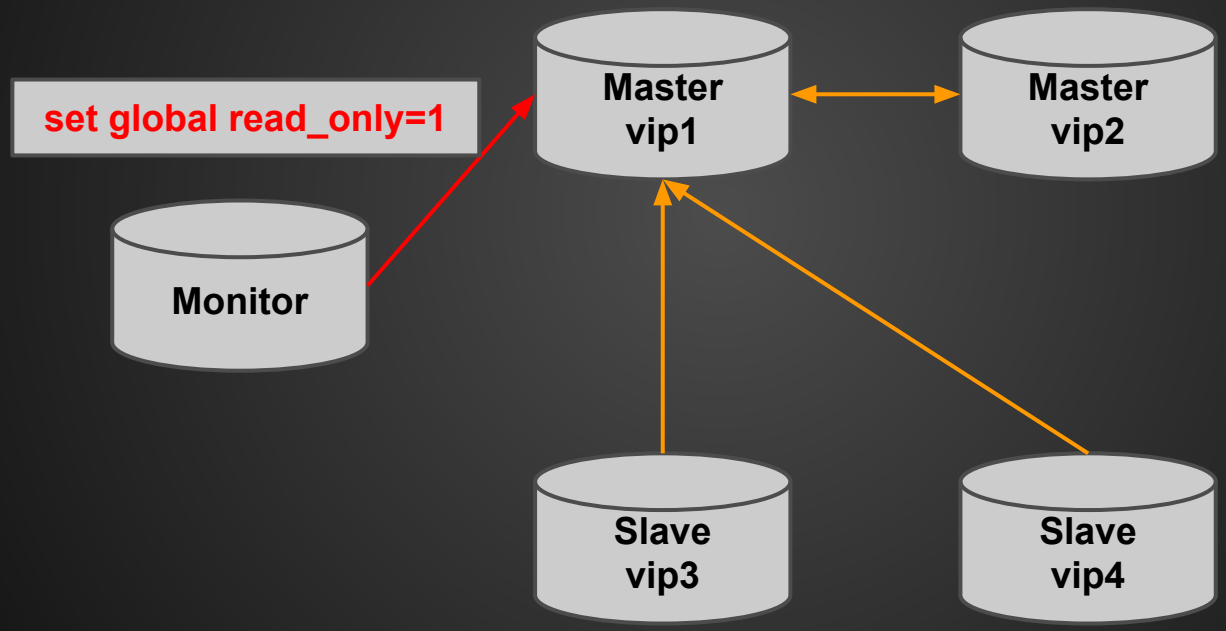
send_status_to_agents



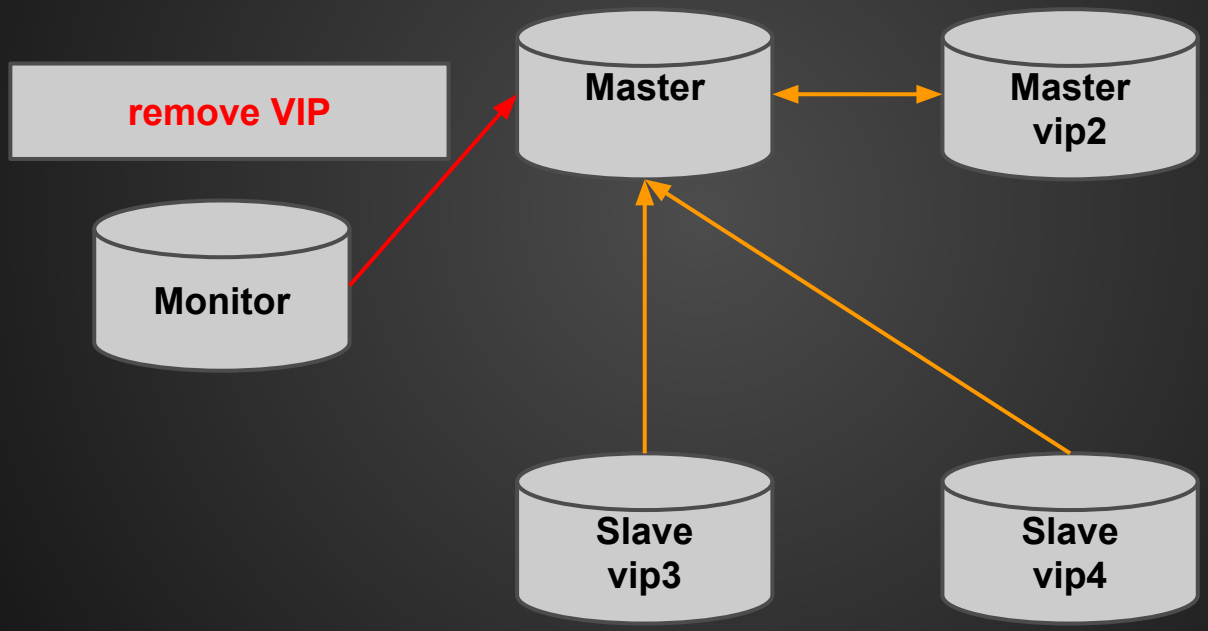
How MMM Do Master Mark-Down



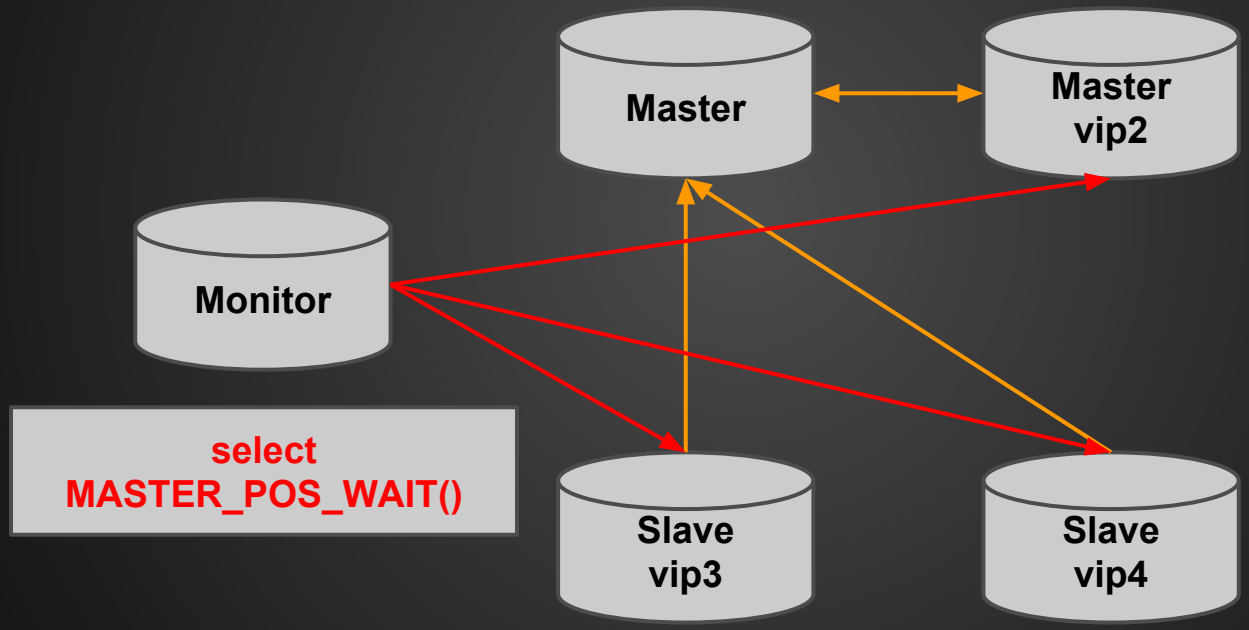
How MMM Do Master Mark-Down



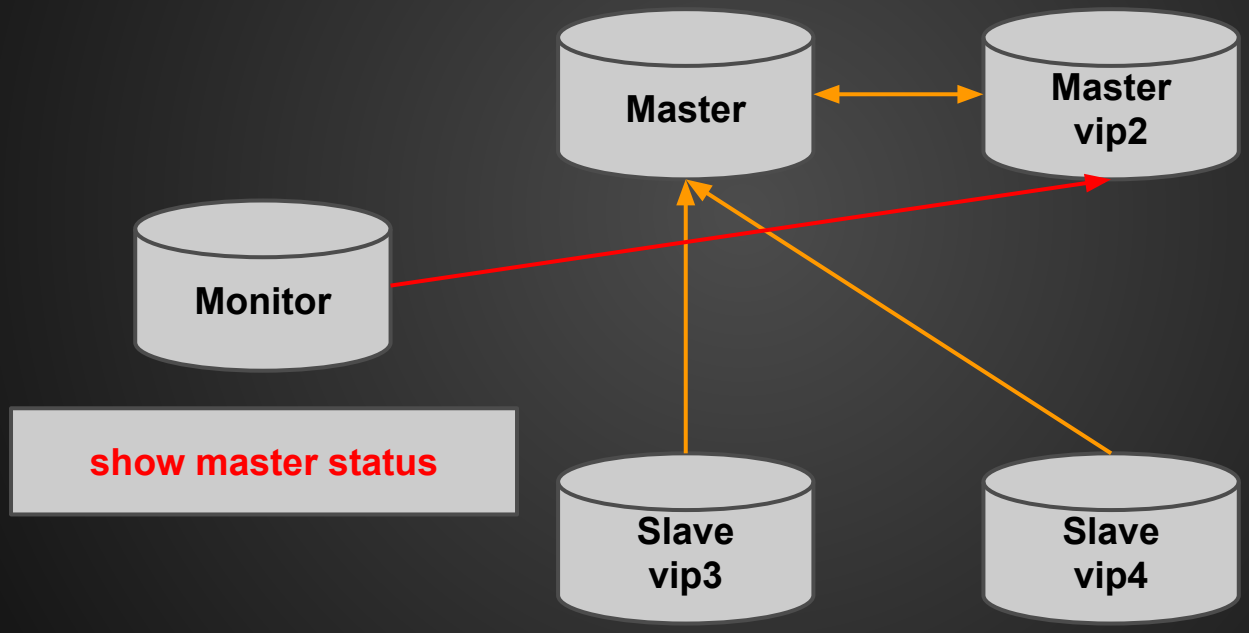
How MMM Do Master Mark-Down



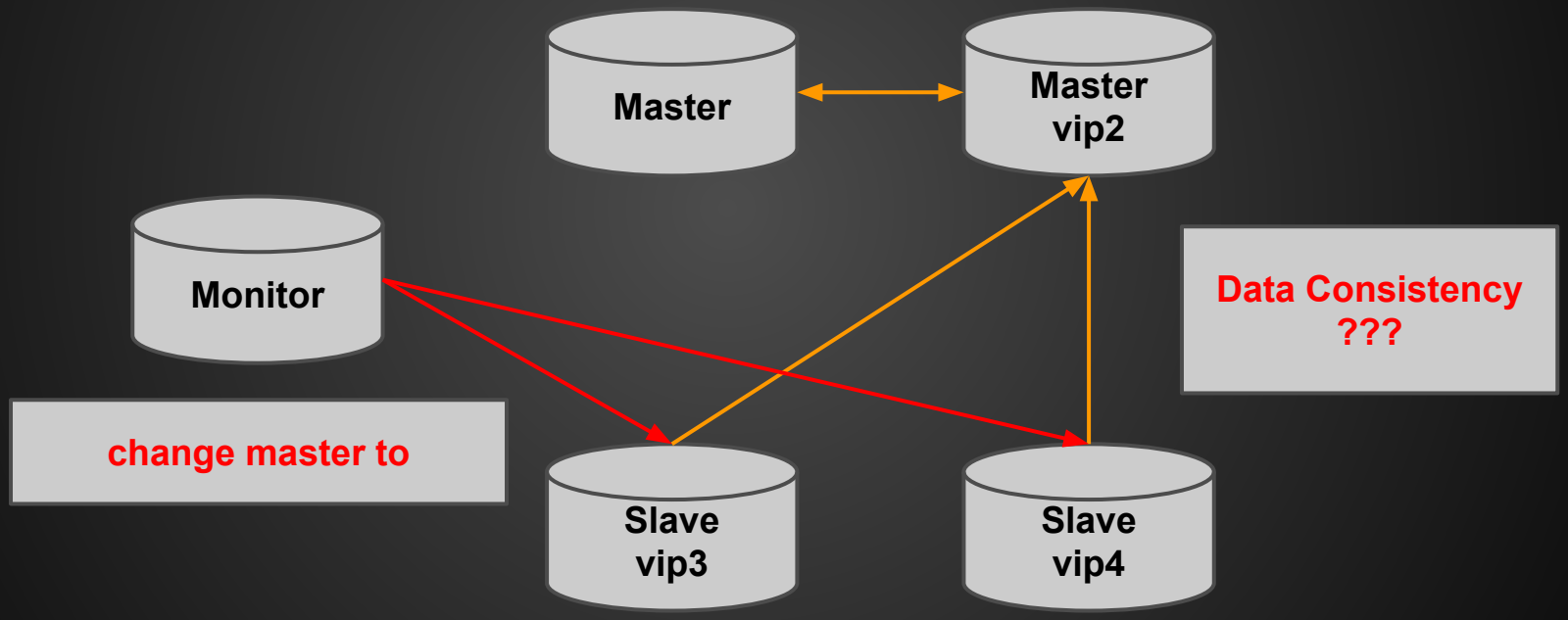
How MMM Do Master Mark-Down



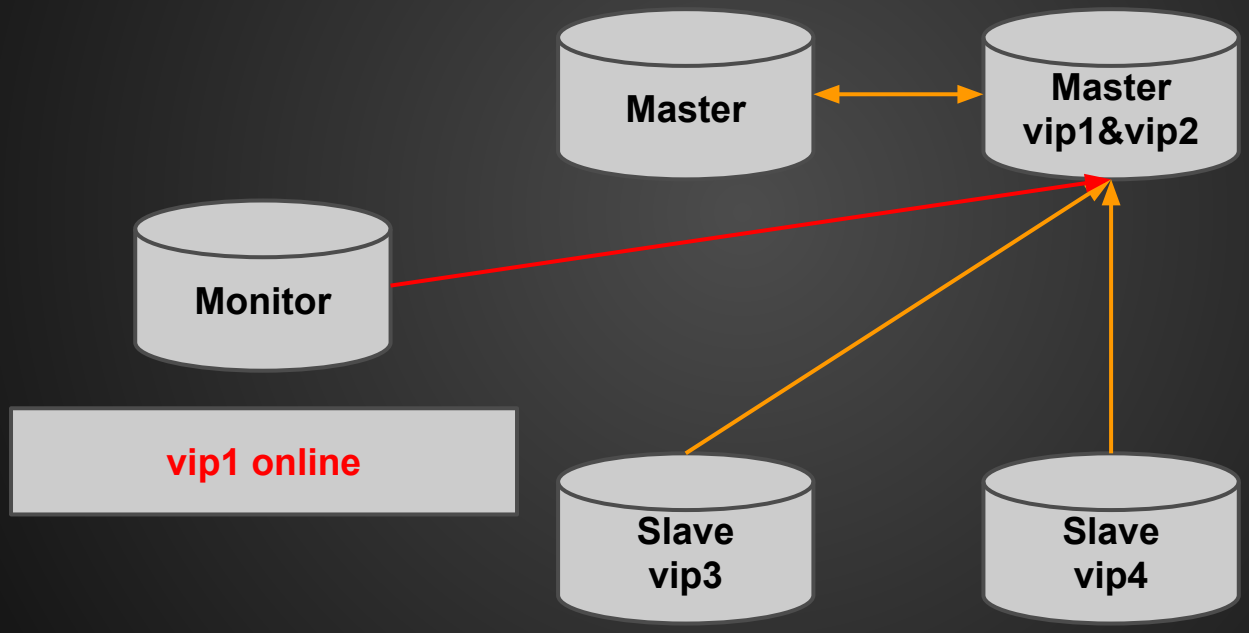
How MMM Do Master Mark-Down



How MMM Do Master Mark-Down



How MMM Do Master Mark-Down



Cons of MMM

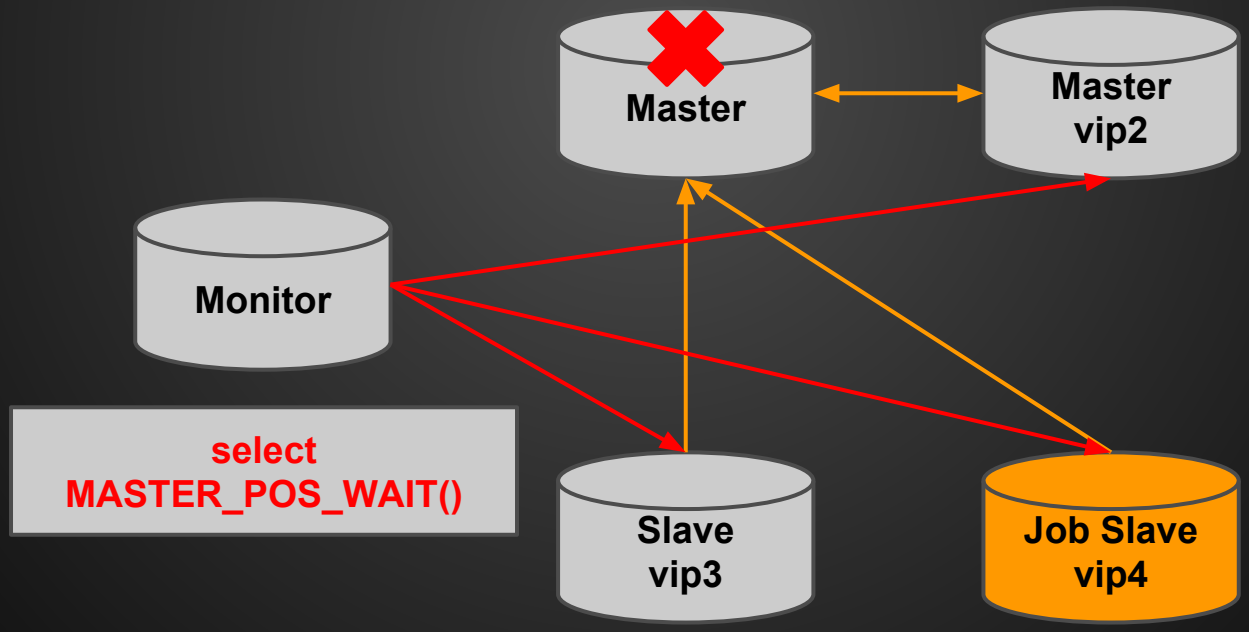
- Global read_only is hard to get on busy servers
- Slow slave bring more downtime
- Change master on slaves is sequential
- Communication between agent & monitor is based on message

MMM is fundamentally broken -- Baron Schwartz: http://www.xaprb.com/blog/2011/05/04/whats-wrong-with-mmm/

Con #1

Slow slave brings more downtime

VIP1 cannot be accessed during this period



Eliminate Replication Lag

Logical Methods:

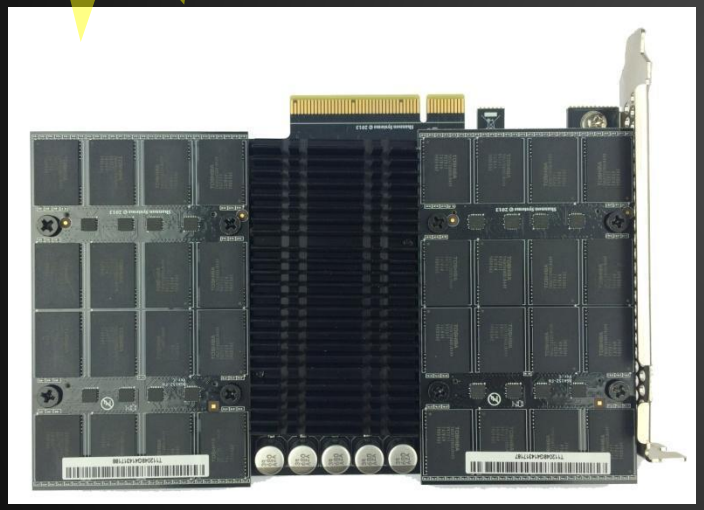
- Chop
- oak-chunk-update
- Avoid peak hour jobs

Physical Methods:

- Prefetch
- Disk upgrade

Shannon Systems Direct-IO™ PCIe SSD

6.4TB 全球单卡最大容量



67/9μs 超低随机读/写延迟

25W 更小峰值功耗 节省运维成本

兼具可靠性和安全性

Smart FTL – Flash Translation Layer
- 热、冷数据动态跟踪
- GC/WL 综合动态平衡，写放大因子最小化
- 双重数据保护机制
- 最大化NAND Flash 寿命

完善容错数据保护机制
- 高达 40bit/1KB ECC或更高
- 读写，擦除出错处理及数据保护
- 页面，坏块出错处理及数据保护

端到端数据保护
- 企业级端到端数据链路保护 (Data path protection)
- 多重数据完整性及正确性校 验

掉电数据保护
- 完善的突发掉电数据保护机制
- 防止系统不正常关机的数据完整性和安全性

过热保护机制
- 防止系统过热对系统造成不可恢复 损伤

Shannon Direct-IO™ PCIe SSD G2T 测试数据

Capacity	800GB	1.2TB	1.6TB	3.2TB/6.4TB
Flash 闪存类型	MLC	MLC	MLC	MLC
Read Bandwidth 读宽带	1.4GB/s	2.0GB/s	2.6GB/s	2.6GB/s
Write Bandwidth 写宽带	1.2GB/s	1.8GB/s	1.8GB/s	1.9GB/s
Random Read Latency(4KB) 随机读延迟	67us	67us	67us	67us
Random Write Latency (4KB) 随机写延迟	9us	9us	9us	9us
Random Read IOPS(4KB) 随机读IOPS	300,000	450,000	590,000	590,000
Random Write IOPS(4KB) 随机写IOPS	310,000	460,000	480,000	480,000
Endurance 寿命	10PB	15PB	20PB	40PB

What Have We Done

However replication lag can still be exist

Improvement:

Only wait for new master to catch up

```
# notify slaves first, if master host has changed
unless ($new_active_master eq $old_active_master) {
    $self->send_agent_status($old_active_master, $new_active_master) if ($old_active_master);
    # save the new master log-pos now, it'll be used when "change master to" is issued for slaves
    my ($new_master_log, $new_master_pos) = split(/:/ , $self->get_master_log_pos($new_active_master));
    DEBUG "Successfully got new master pos:$new_master_log:$new_master_pos";


    # if the master role has changed, notify the new master to take vip
    $self->send_agent_status($new_active_master) if ($new_active_master ne $old_active_master);
    # after the master has taken the writer role, notify all the slaves to do change master
    $self->notify_slaves("$new_active_master,$new_master_log,$new_master_pos");
}
```

Con #2

Global read_only is hard to get on busy servers

1. Invalidates the Query Cache.
2. Waits for all in-flight updates to complete and at the same time it blocks all incoming updates.
3. Closes all open tables and expels them from the table cache. Wait for all SELECT queries to complete. Incoming SELECT queries will get blocked.
4. Blocks COMMITs.

What Have We Done

Manually check long sql before switch

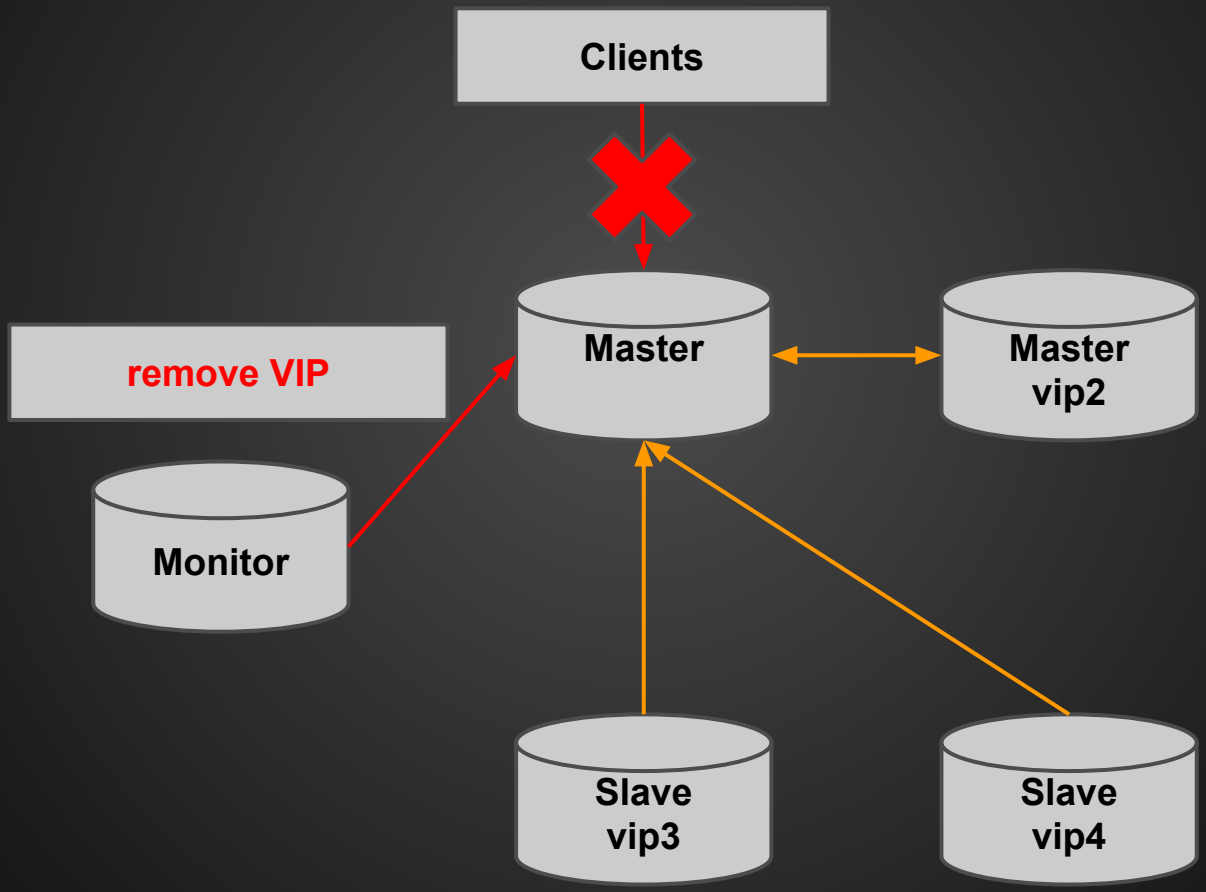
Improvement:

- remove master vip before read_only
- to kill sql before read_only

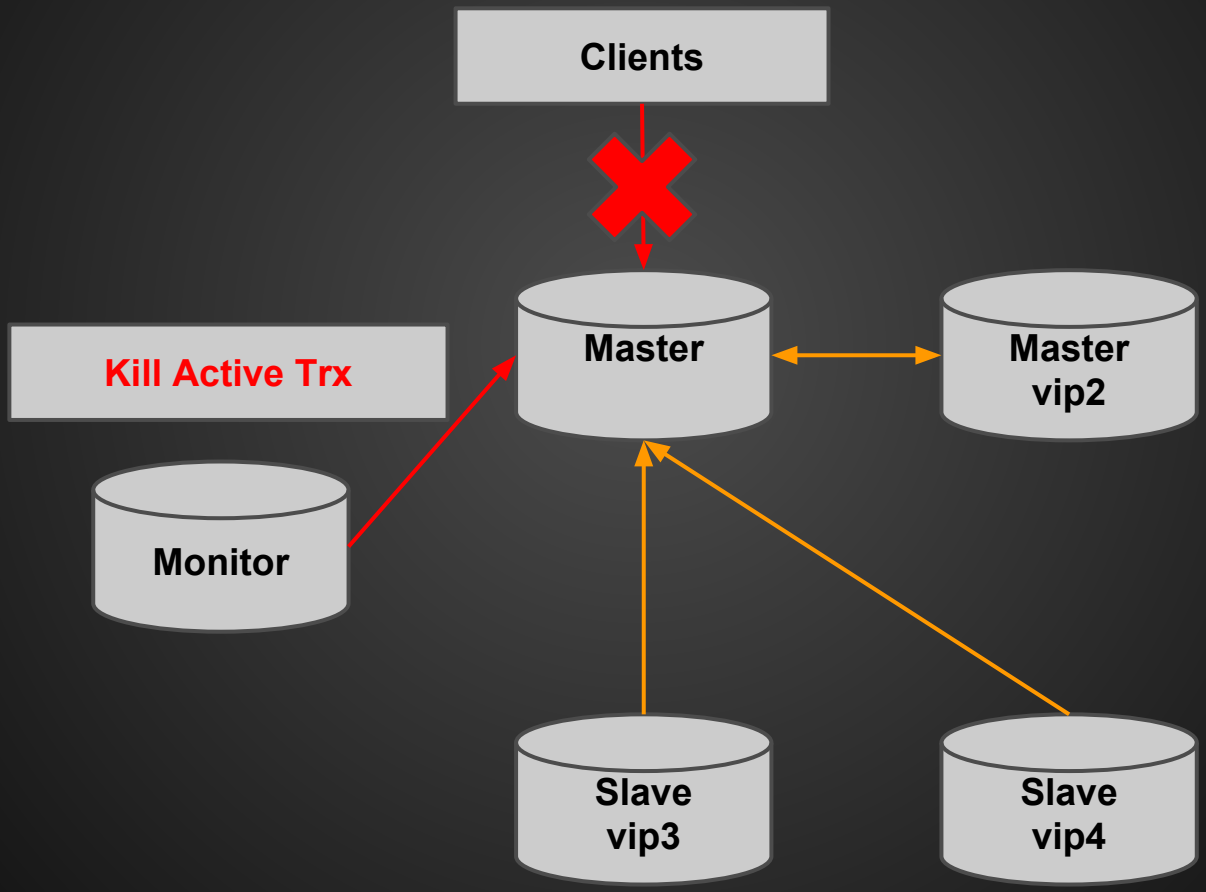
What Have We Done

```
sub del($) {
        my $self = shift;

        my $res;

        $res = MMM::Agent::Helpers::clear_ip($main::agent->interface, $self->ip);
        if (!defined($res) || $res !~ /^OK/) {
                FATAL sprintf("Couldn't clear IP '%s' from interface '%s': %s", $self->ip, $main::a
gent->interface, defined($res) ? $res : 'undef');
        }

        #kill all active process
        #let all uncommited transactions to rollback
        $res = MMM::Agent::Helpers::kill_process();
        if (!defined($res) || $res !~ /^OK/) {
                FATAL sprintf("Couldn't kill active process in MySQL: %s", defined($res) ? $res : '
undef');
        }

       if ($self->name eq $main::agent->writer_role) {
                $res = MMM::Agent::Helpers::deny_write();
                if (!defined($res) || $res !~ /^OK/) {
                        FATAL sprintf("Couldn't deny writes: %s", defined($res) ? $res : 'undef');
                }
        }

}
```

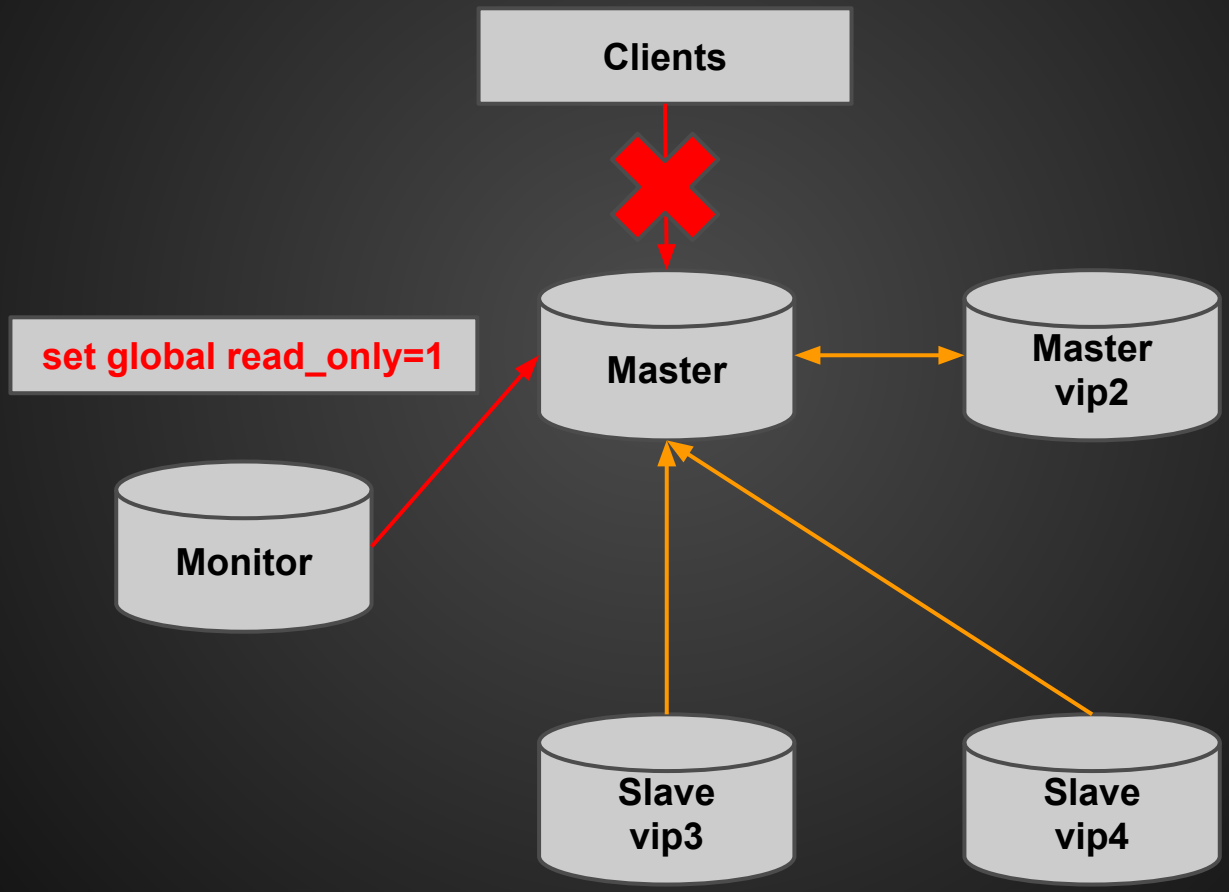
How DP-MMM Do Failover



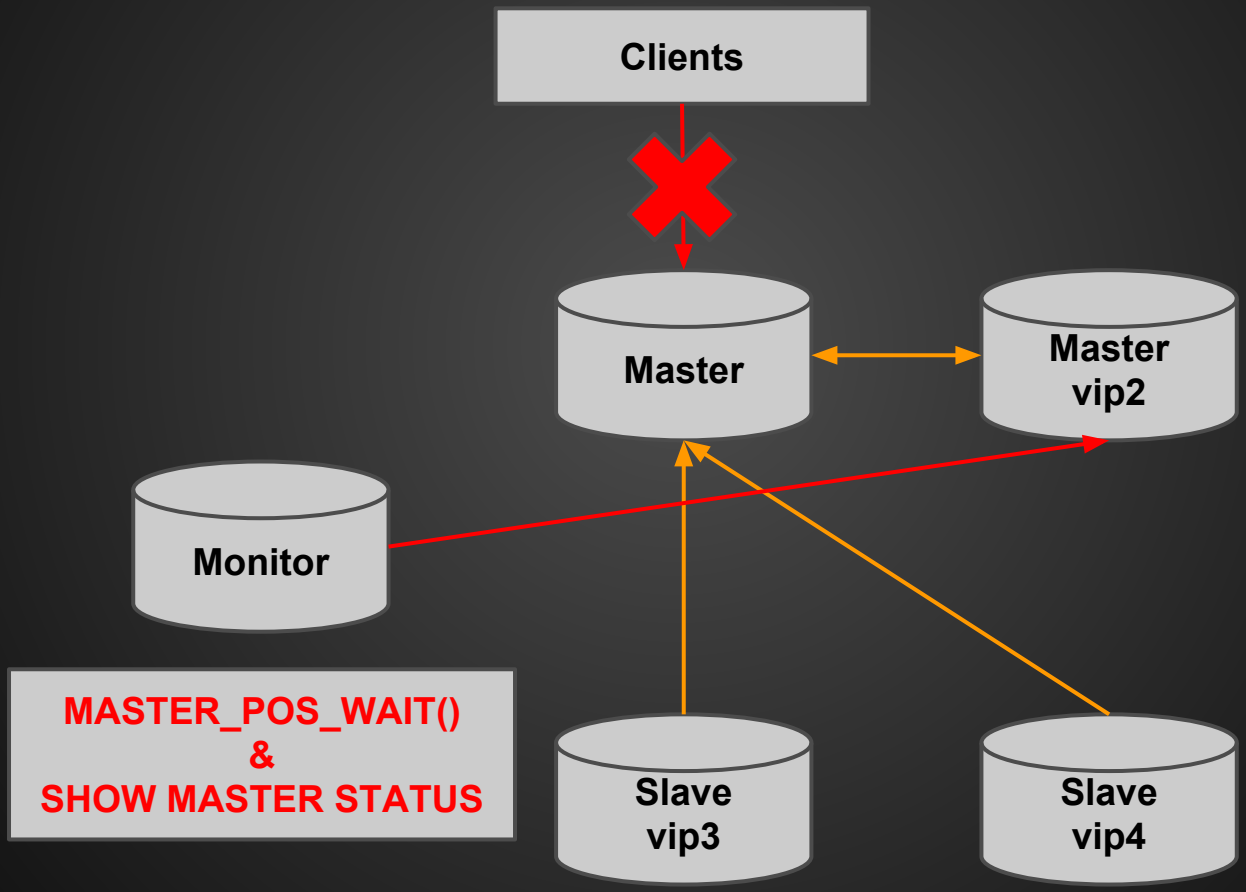
How DP-MMM Do Failover



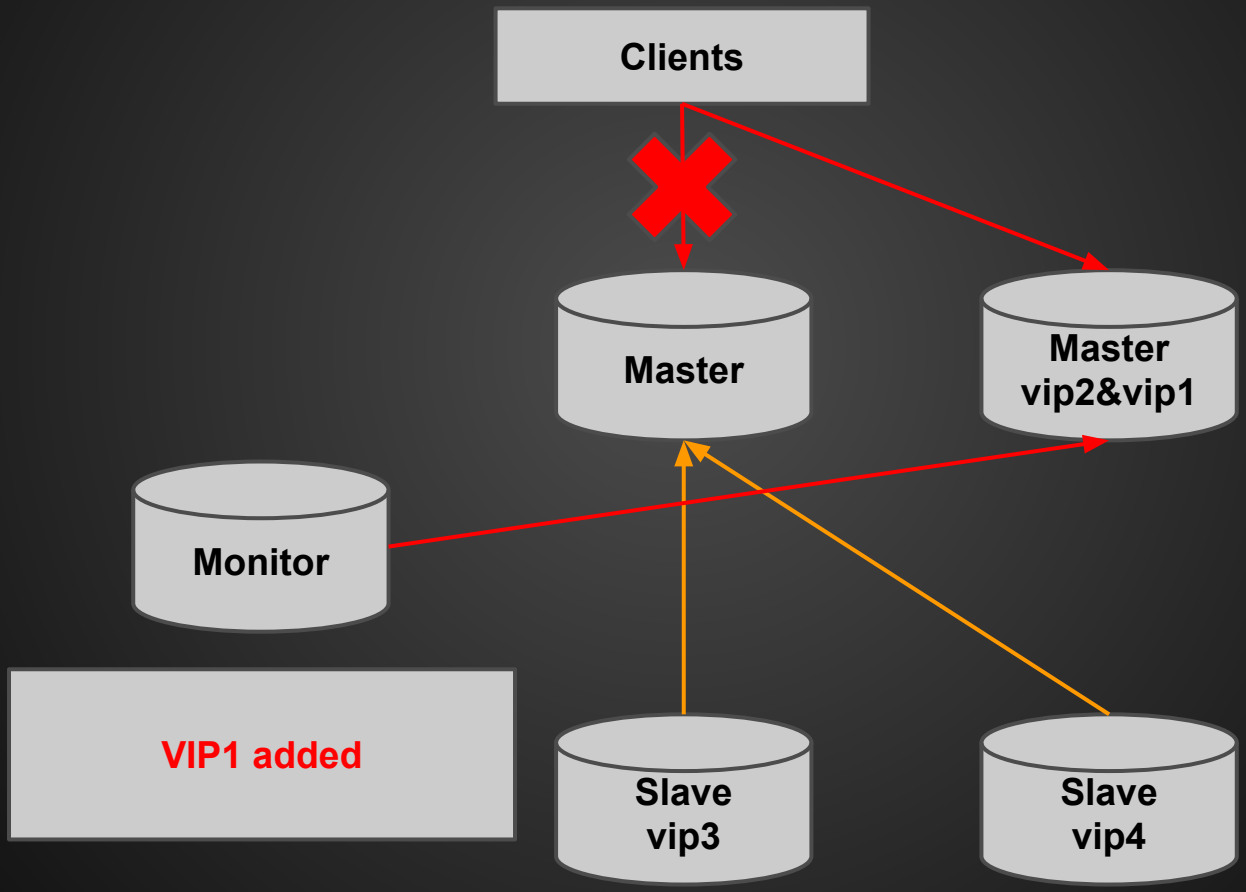
How DP-MMM Do Failover



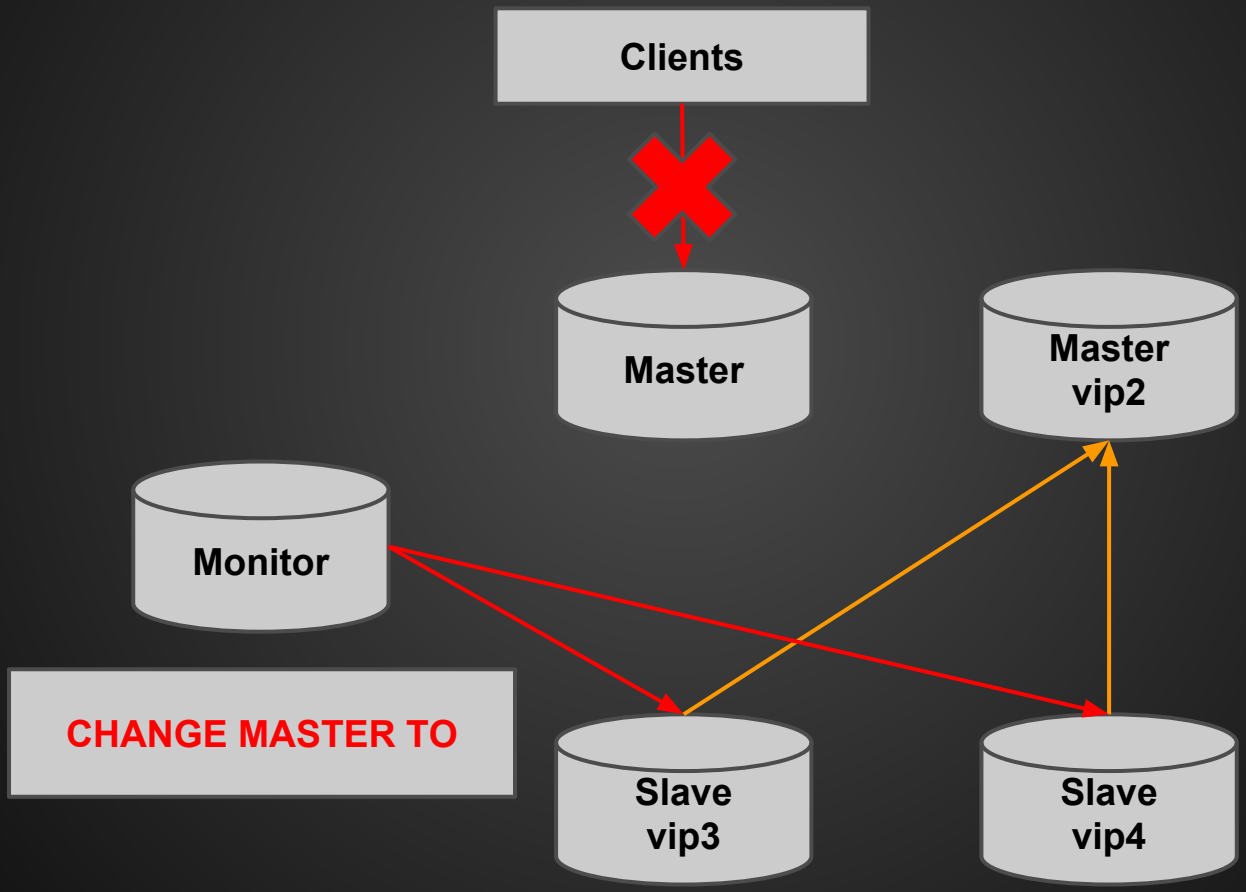
How DP-MMM Do Failover



How DP-MMM Do Failover



How DP-MMM Do Failover



Con #3

Why MMM is fundamentally broken:

- Communication based on message between monitor & agent
- Agent manage its own state
- Data Loss

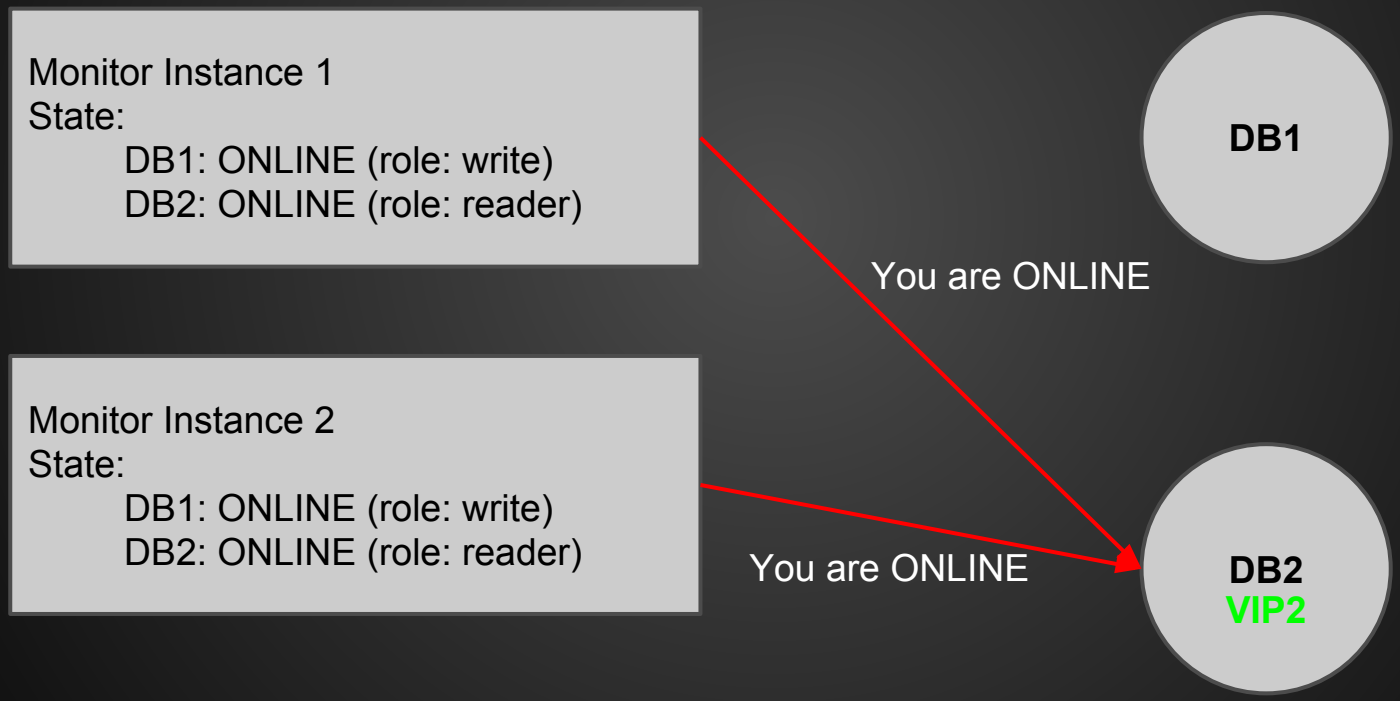
Example of Broken

mmm_control @cluster set_offline db1

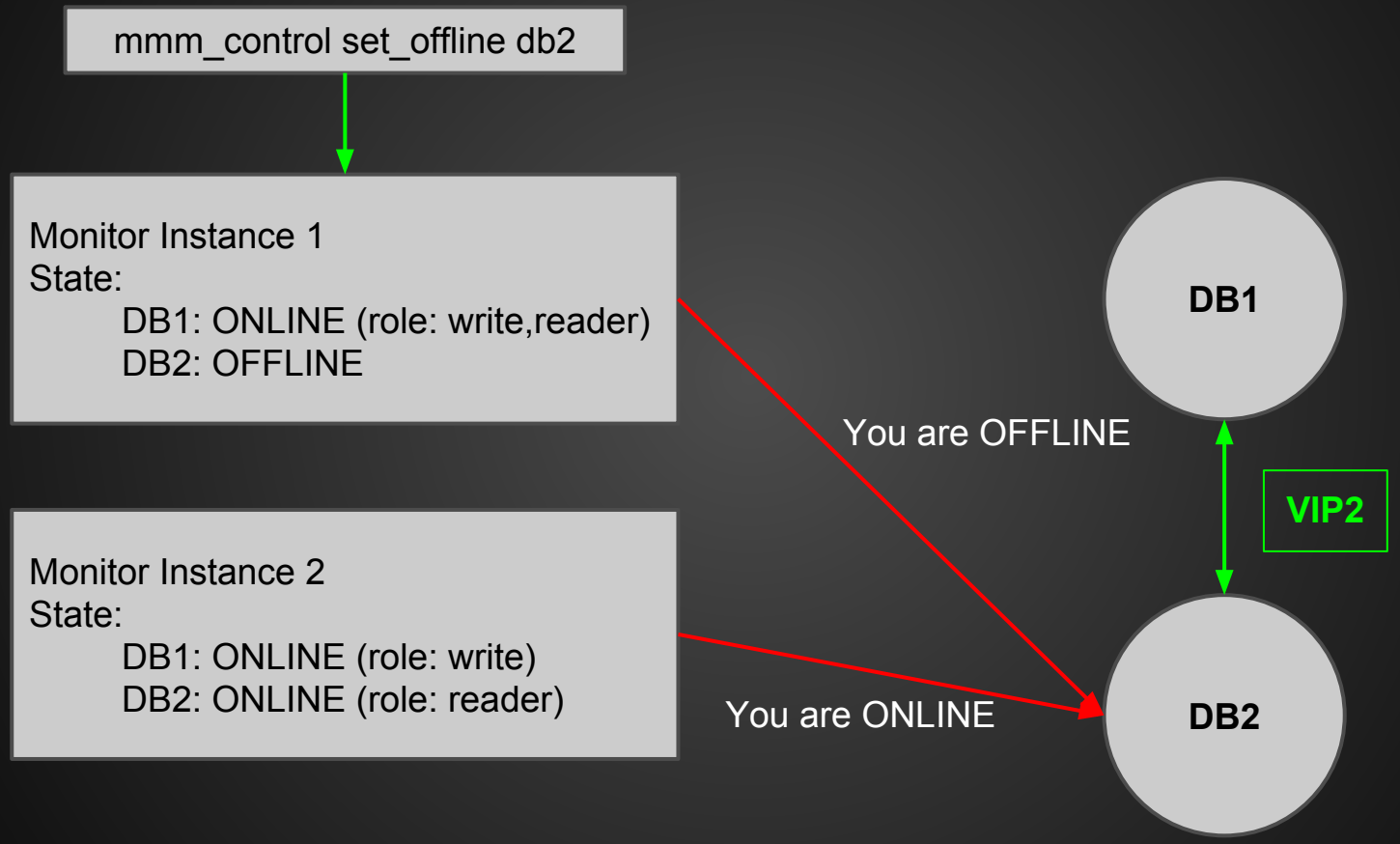
success!

```
2014/01/22 06:35:13  INFO We have some new roles added or old rules deleted!
2014/01/22 06:35:13  INFO Deleted: reader1(10.1.1.227)
2014/01/22 06:35:14  INFO We have some new roles added or old rules deleted!
2014/01/22 06:35:14  INFO Added:   reader1(10.1.1.227)
2014/01/22 06:35:17  INFO We have some new roles added or old rules deleted!
2014/01/22 06:35:17  INFO Deleted: reader1(10.1.1.227)
2014/01/22 06:35:18  INFO We have some new roles added or old rules deleted!
2014/01/22 06:35:18  INFO Added:   reader1(10.1.1.227)
2014/01/22 06:35:21  INFO We have some new roles added or old rules deleted!
2014/01/22 06:35:21  INFO Deleted: reader1(10.1.1.227)
2014/01/22 06:35:22  INFO We have some new roles added or old rules deleted!
2014/01/22 06:35:22  INFO Added:   reader1(10.1.1.227)
2014/01/22 06:35:25  INFO We have some new roles added or old rules deleted!
2014/01/22 06:35:25  INFO Deleted: reader1(10.1.1.227)
2014/01/22 06:35:26  INFO We have some new roles added or old rules deleted!
2014/01/22 06:35:26  INFO Added:   reader1(10.1.1.227)
2014/01/22 06:35:32  INFO We have some new roles added or old rules deleted!
2014/01/22 06:35:32  INFO Deleted: reader1(10.1.1.227)
2014/01/22 06:35:33  INFO We have some new roles added or old rules deleted!
2014/01/22 06:35:33  INFO Added:   reader1(10.1.1.227)
2014/01/22 06:35:36  INFO We have some new roles added or old rules deleted!
2014/01/22 06:35:36  INFO Deleted: reader1(10.1.1.227)
2014/01/22 06:35:37  INFO We have some new roles added or old rules deleted!
2014/01/22 06:35:37  INFO Added:   reader1(10.1.1.227)
2014/01/22 06:35:40  INFO We have some new roles added or old rules deleted!
2014/01/22 06:35:40  INFO Deleted: reader1(10.1.1.227)
2014/01/22 06:35:41  INFO We have some new roles added or old rules deleted!
```

Cause of Broken



Cause of Broken



What Have We Done

Fix:

multiple monitor instance bug.

But we cannot fix its architecture:

mmm_control set_online db1 mean **NOTHING** but confirmation of command

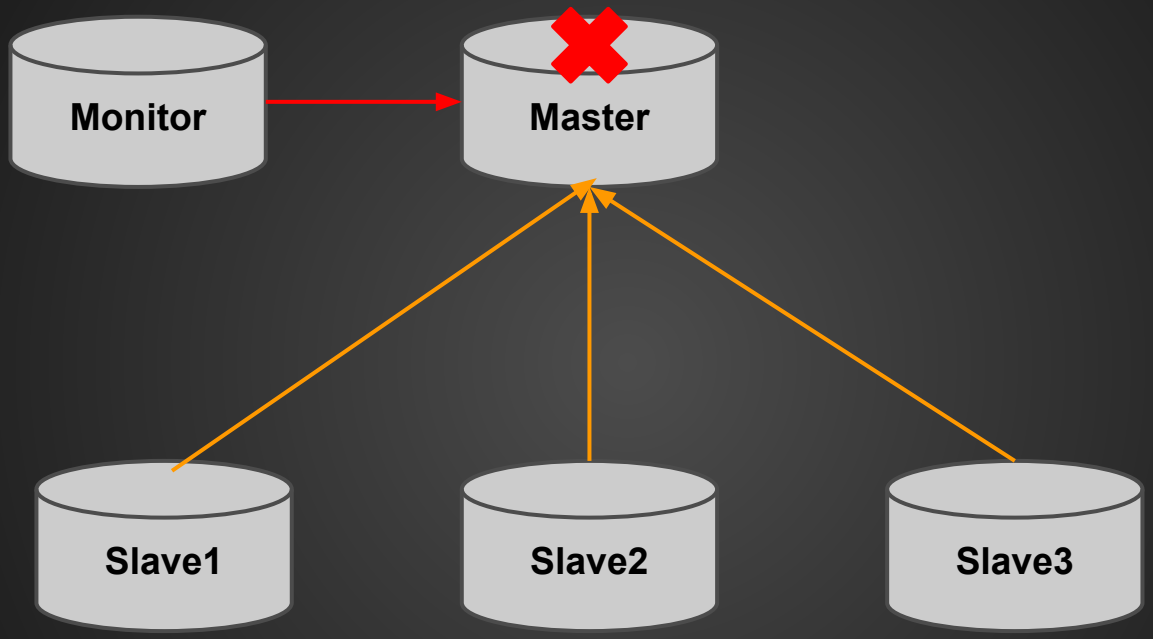
Cons of MMM

- Global read_only is hard to get on busy servers
- Slow slave bring more downtime
- Change master on slaves is sequential
- Communication between agent & monitor is based on message

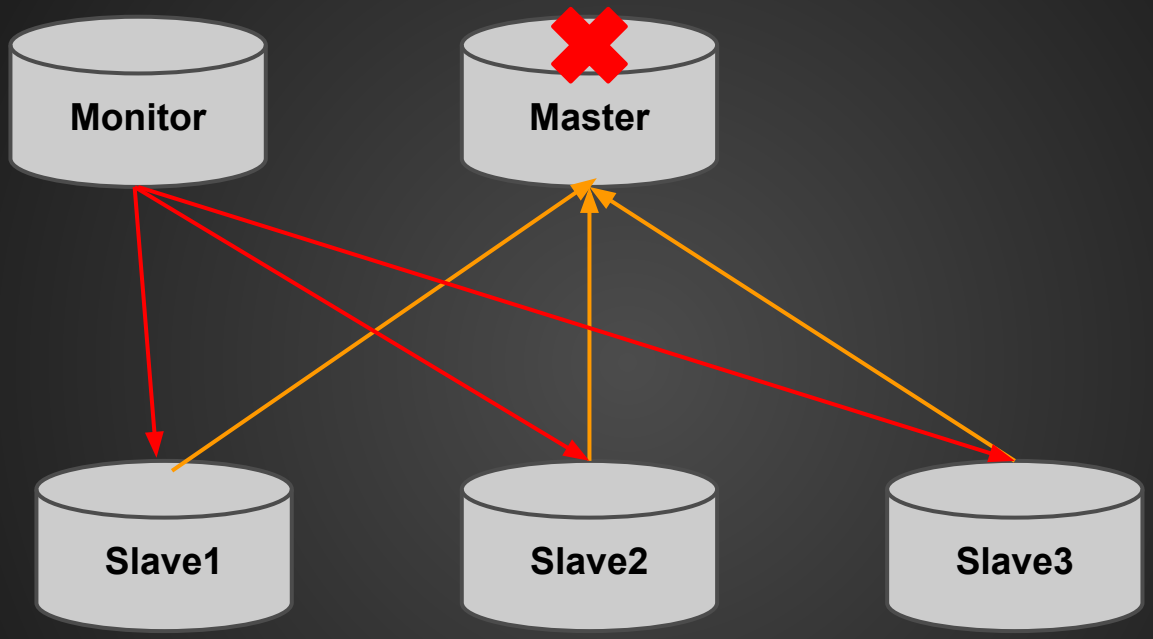
Why We Chose MHA in 2014

	MMM	MHA
Data Loss	Every Time after failover	Almost none
Communicate with DAL	N/A	User defined script
HA Solution without VIP	N/A	Supported

How MHA Do Failover



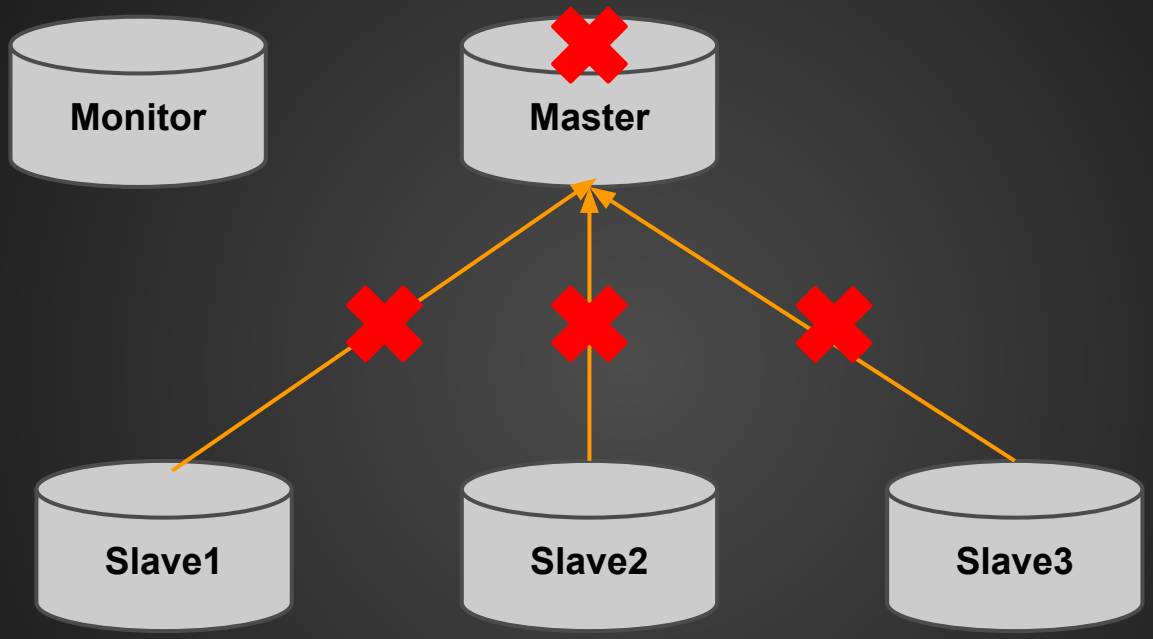
How MHA Do Failover



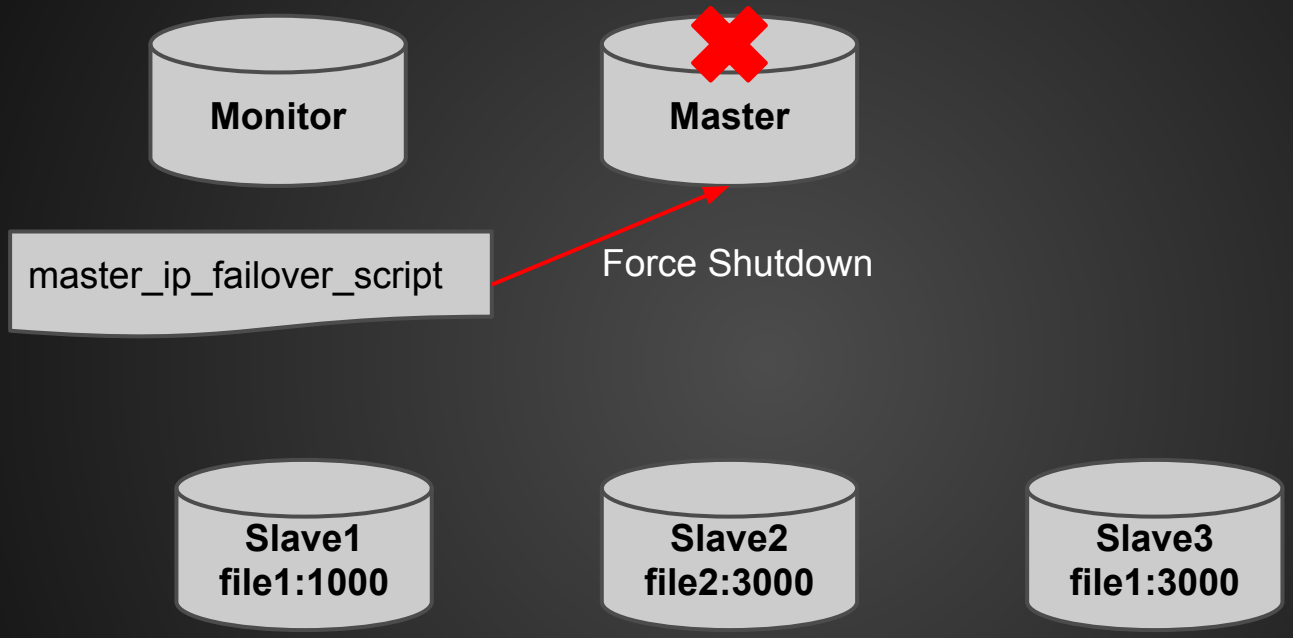
Connect to all nodes and check:

- master is really dead
- all slaves is the slave of dead master
- last failover is ealier than $last_failover_minute

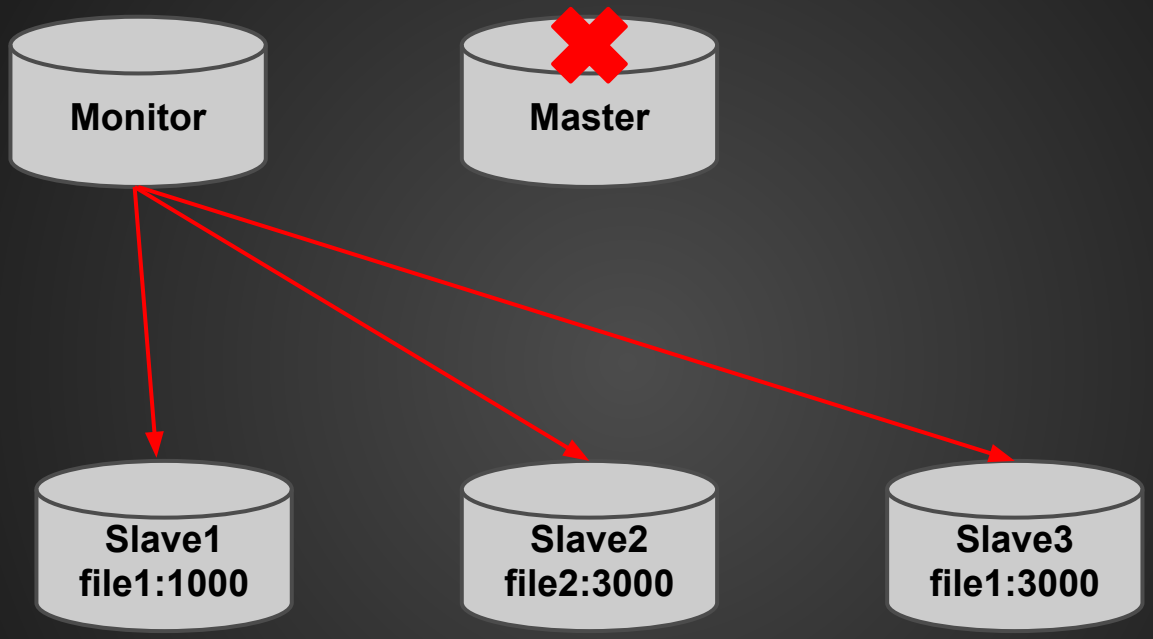
How MHA Do Failover



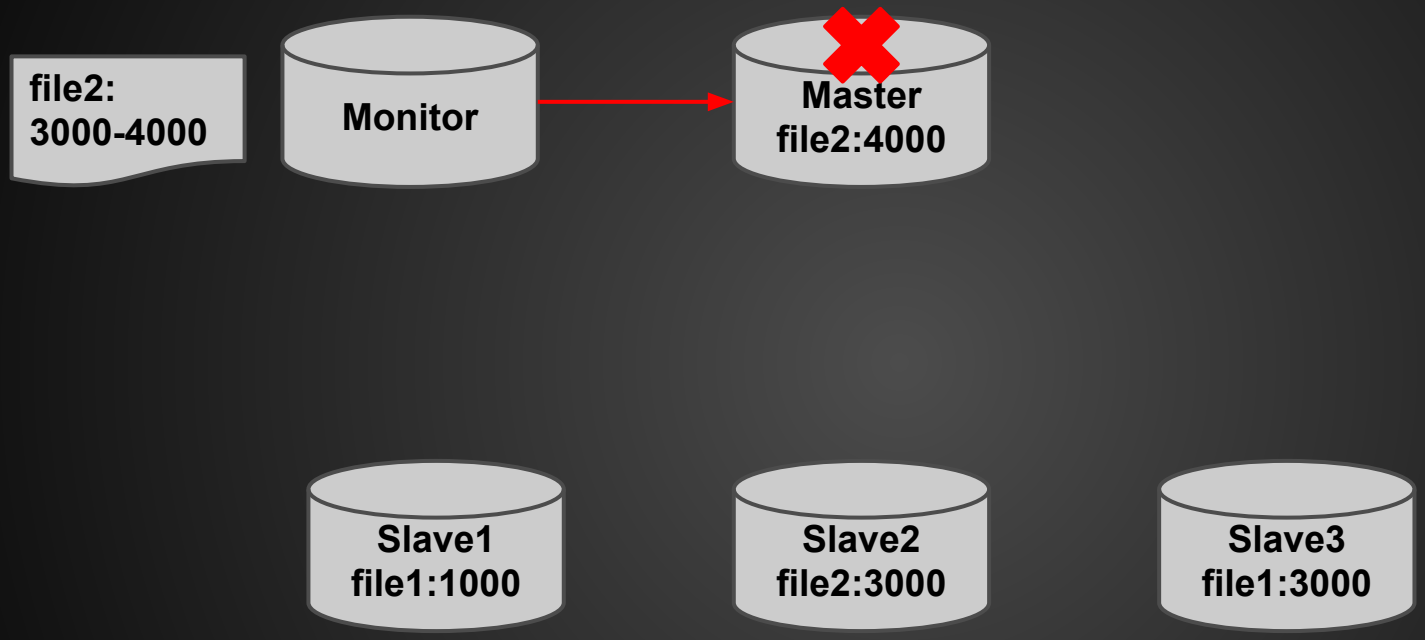
Call master_ip_failover_script



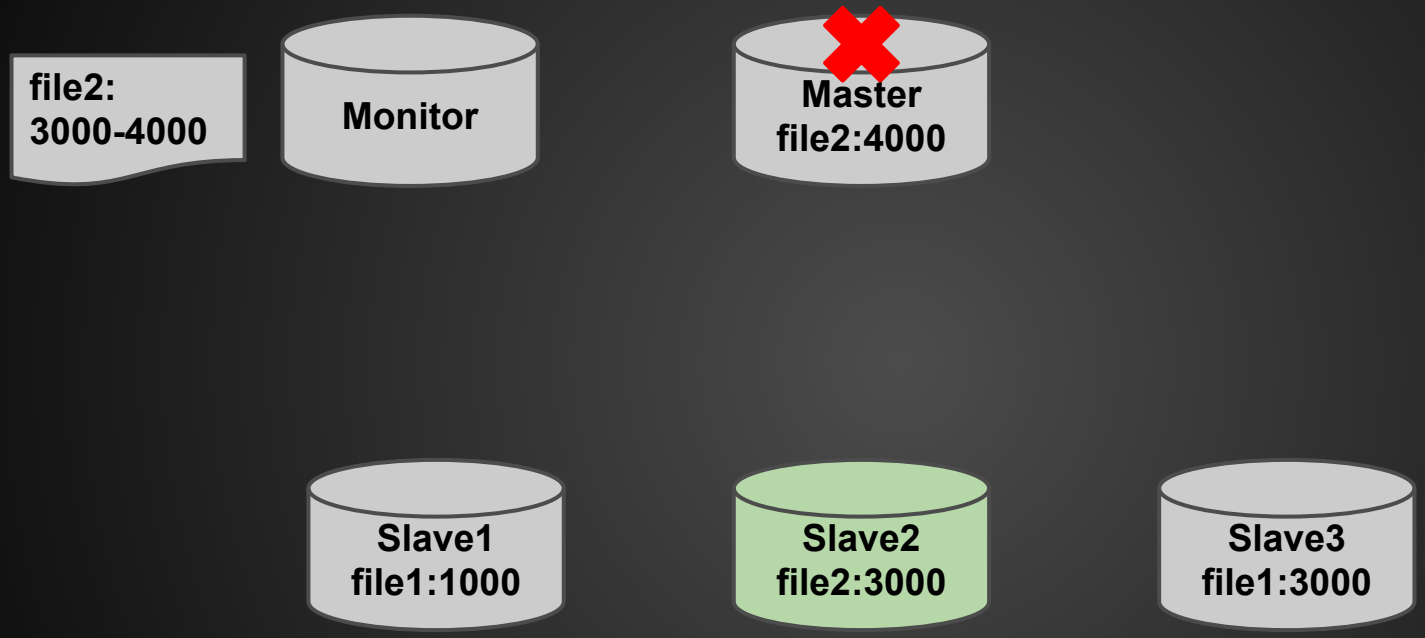
Find Latest Slave Relay Log Position



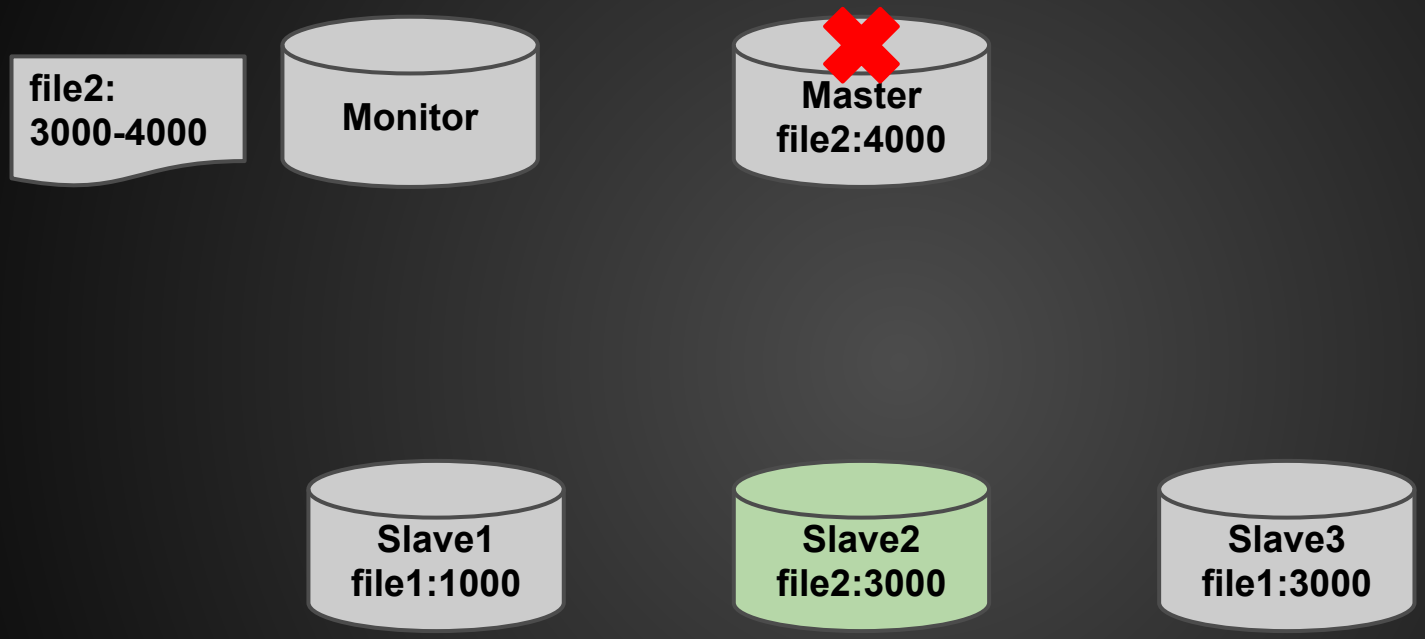
Save Master Binlog if Alive



Find Relay Log Base Slave



Find Candidate Master



Candidate Master Configuration

candidate_master:

this server is prioritized in election

no_master:

this server will not be new master in election

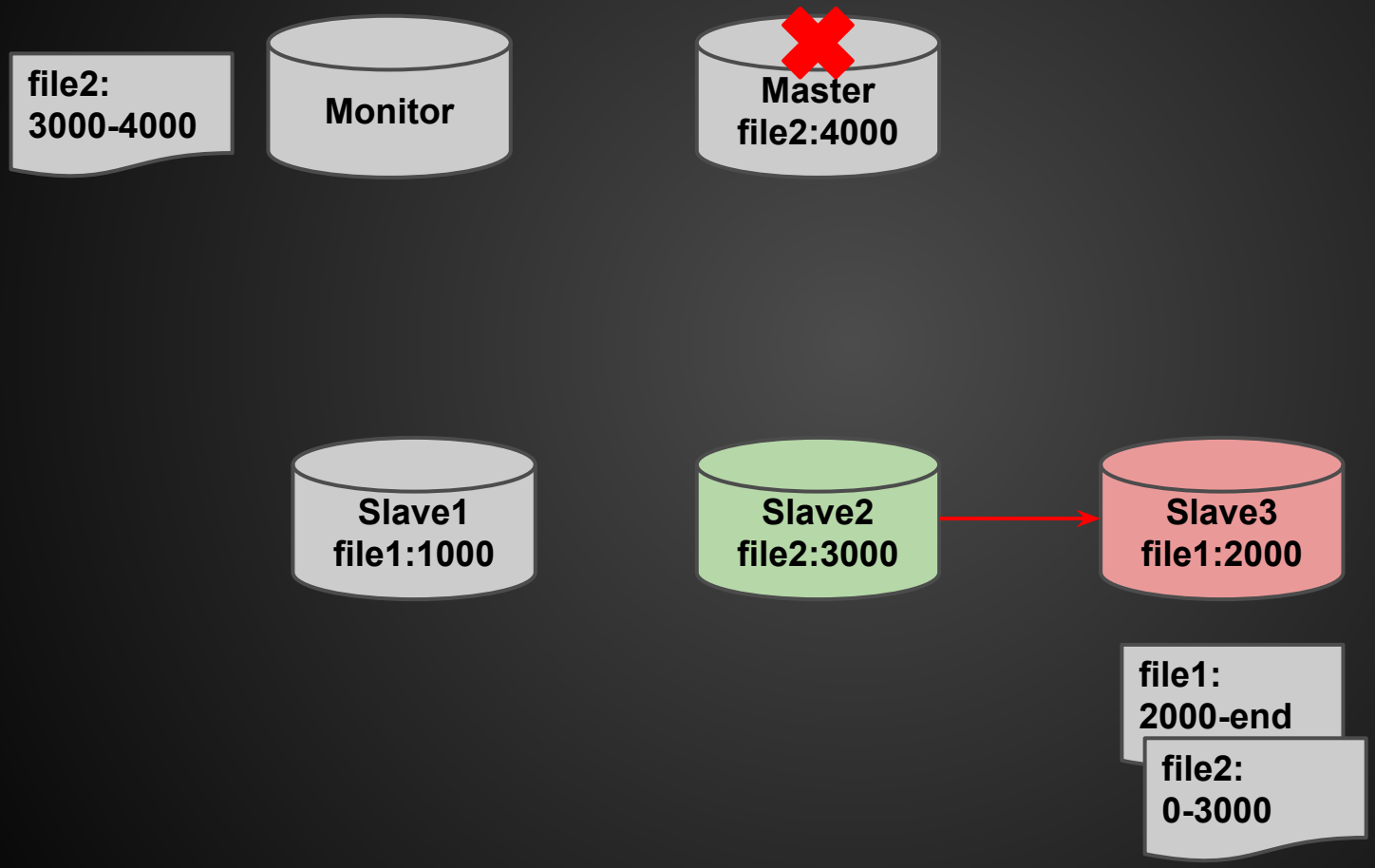
latest_priority:

use order by block_name instead relay_log position in election

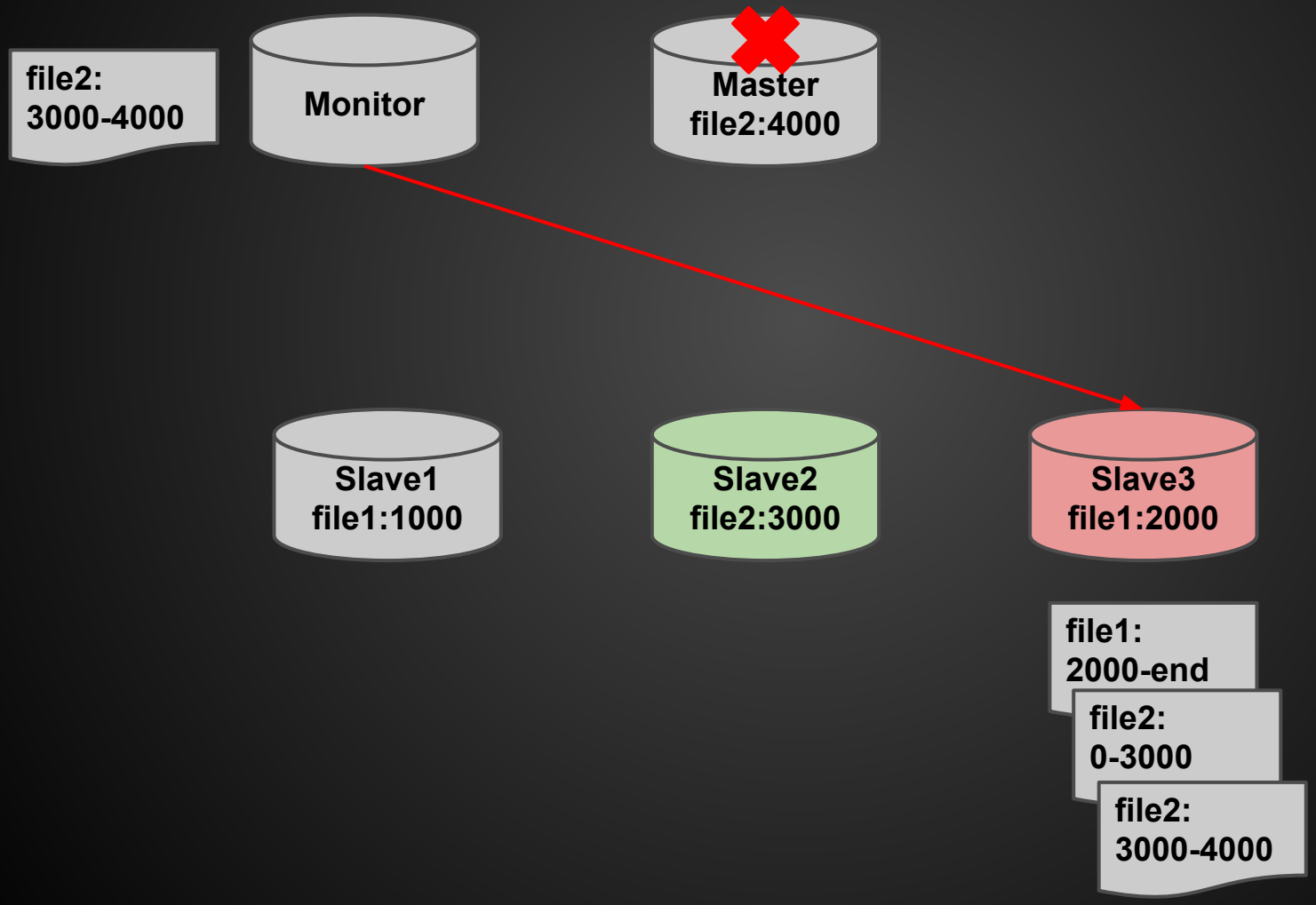
Candidate Master Election Rule

IF nothing special configured
    elect relay log base slave
IF $latest_priority=1
    elect latest slave with candidate_master=1
ELSE
    elect cacndidate_master by name
IF $latest_priority=1
    elect latest slave
ELSE
    elect remaining slave

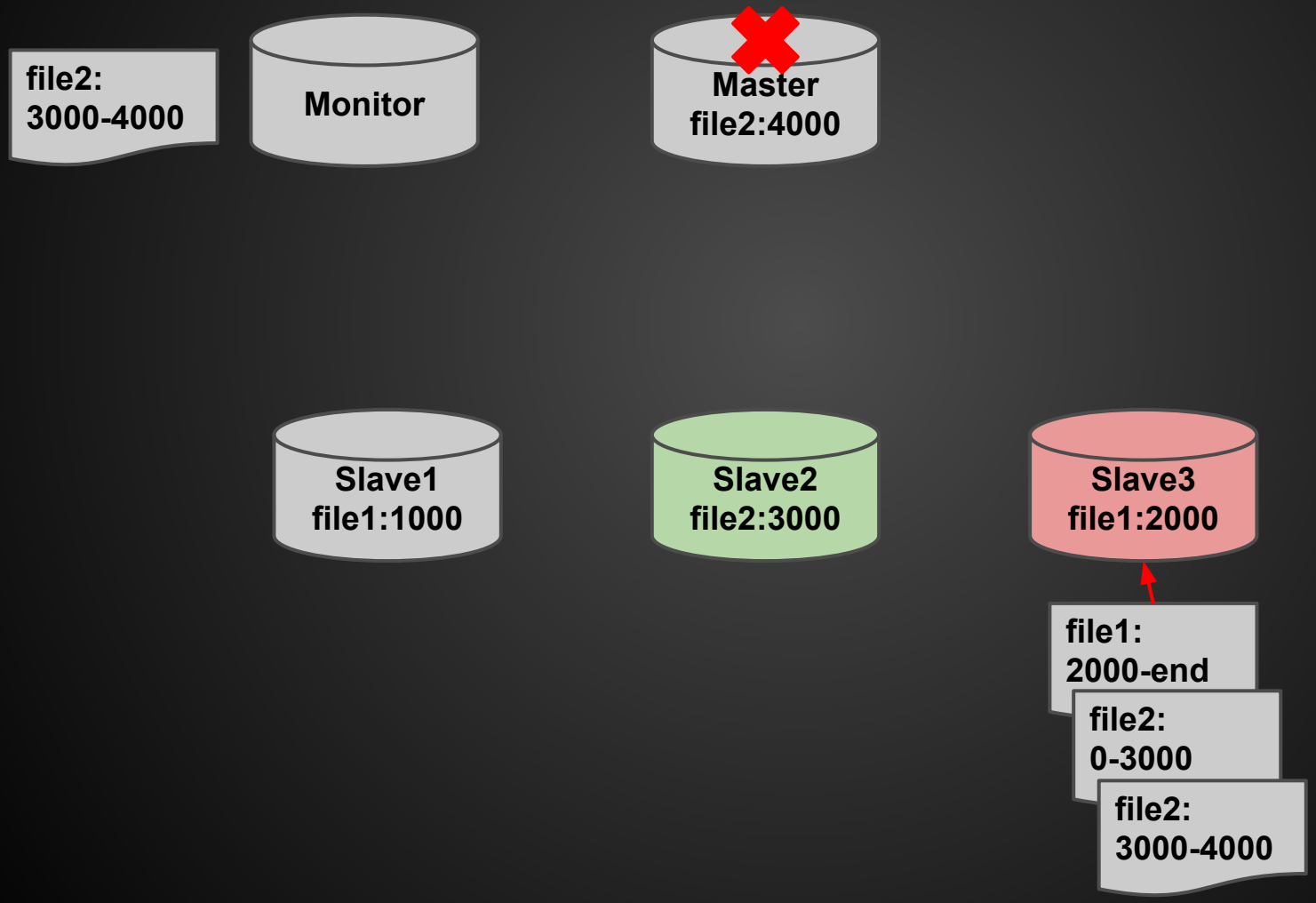
Recover Candidate Master



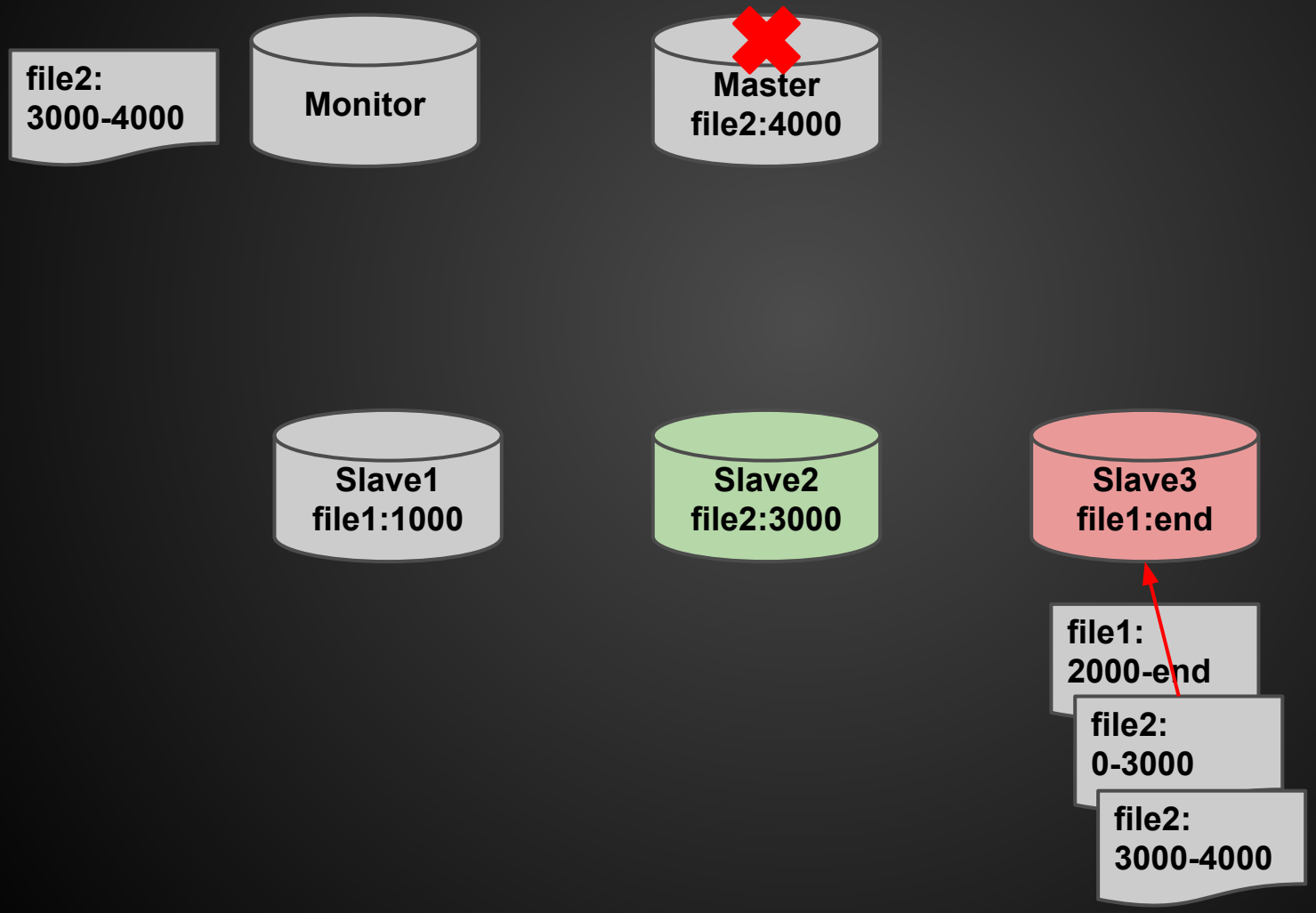
Recover Candidate Master



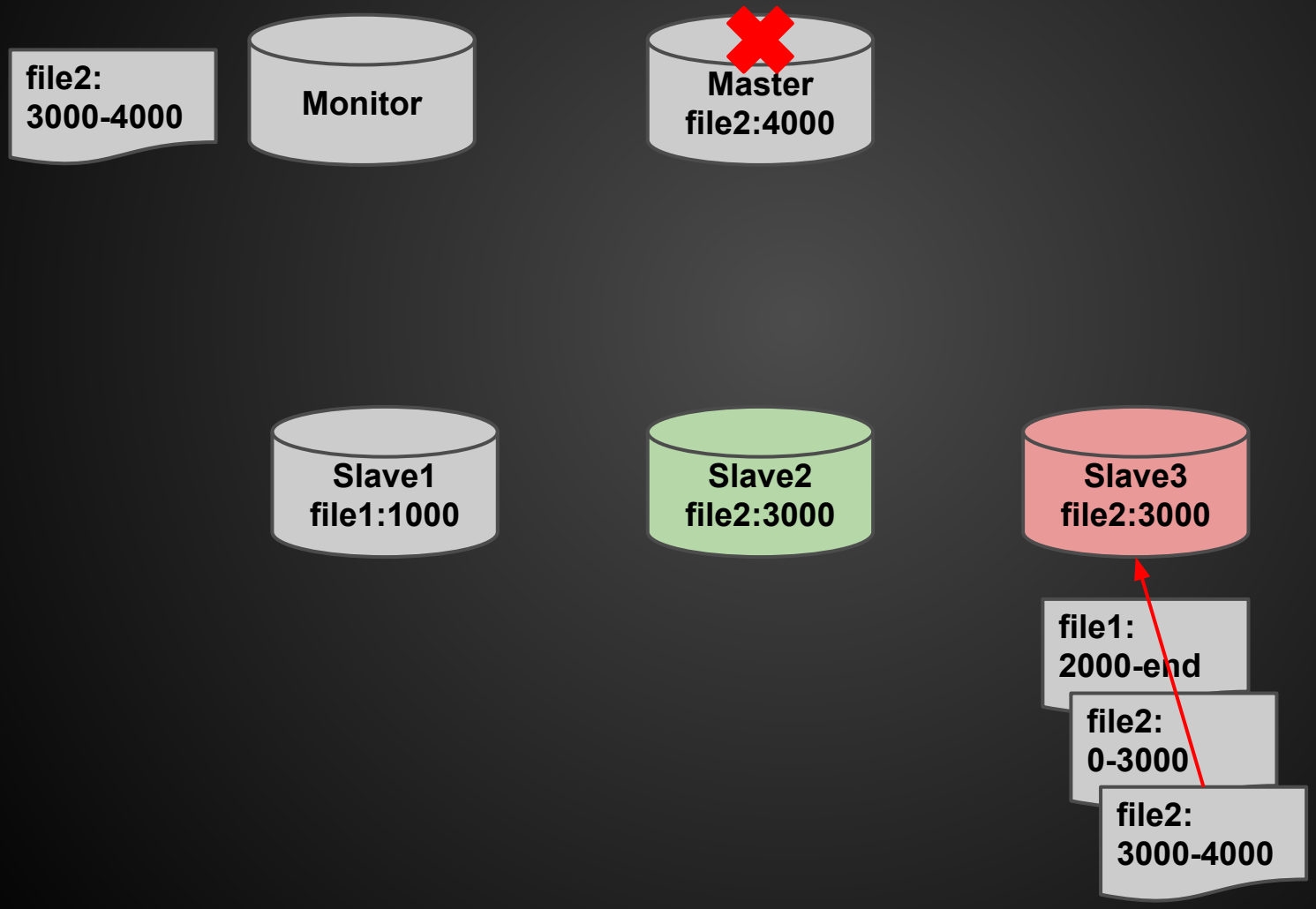
Recover Candidate Master



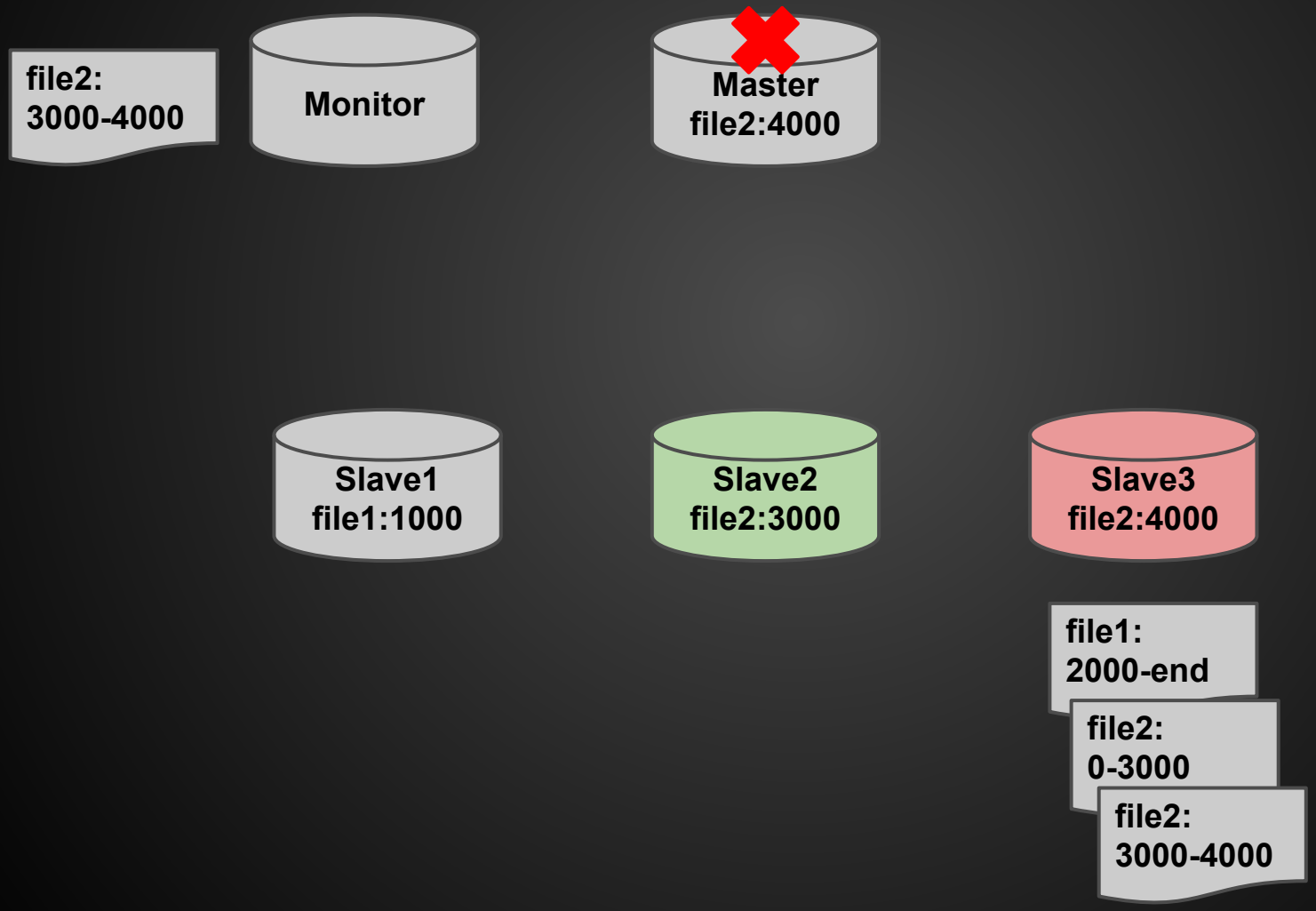
Recover Candidate Master



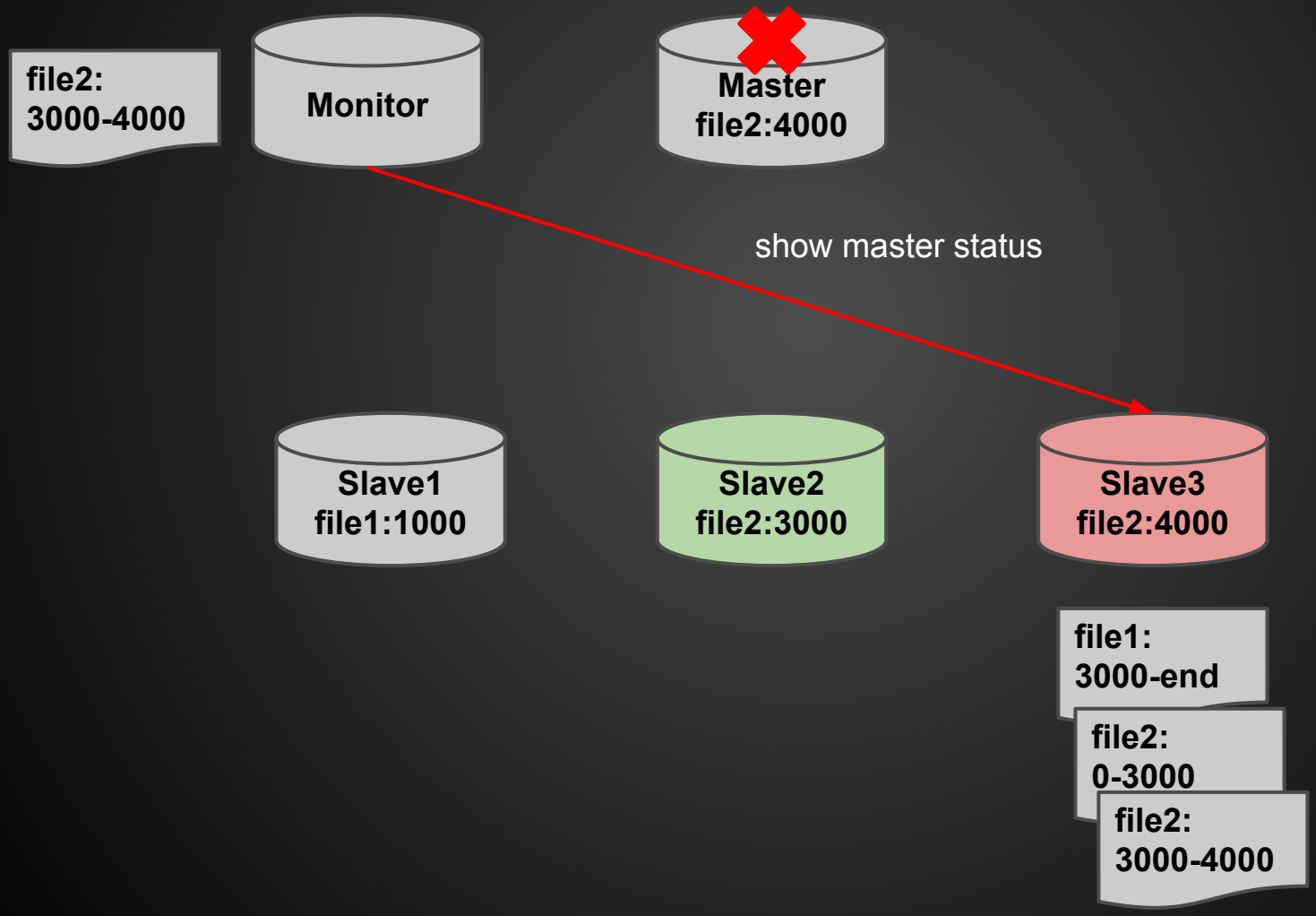
Recover Candidate Master



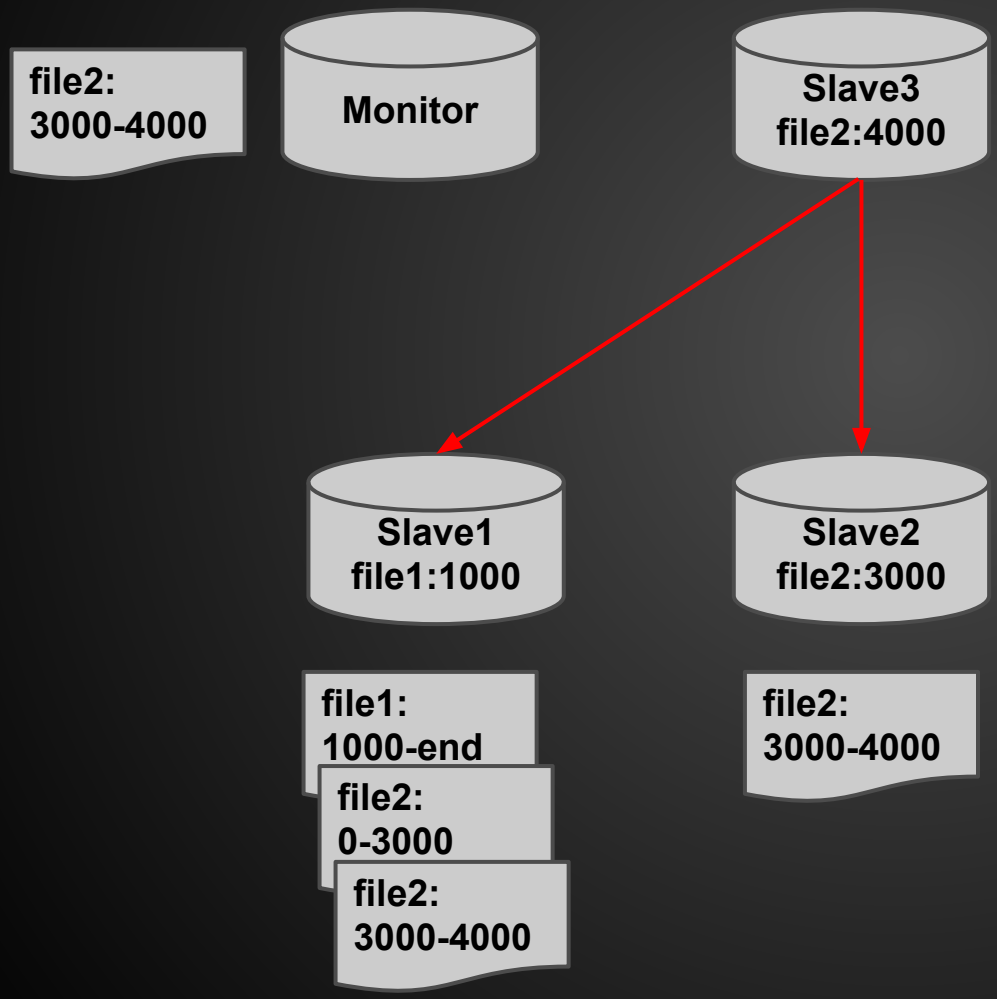
Recover Candidate Master



Recover Candidate Master



Parallel Recover Slaves



Cons of MHA

No Slave Management

CLI Not So Friendly compared to MMM

SSH login without password within all nodes

What Have We Done

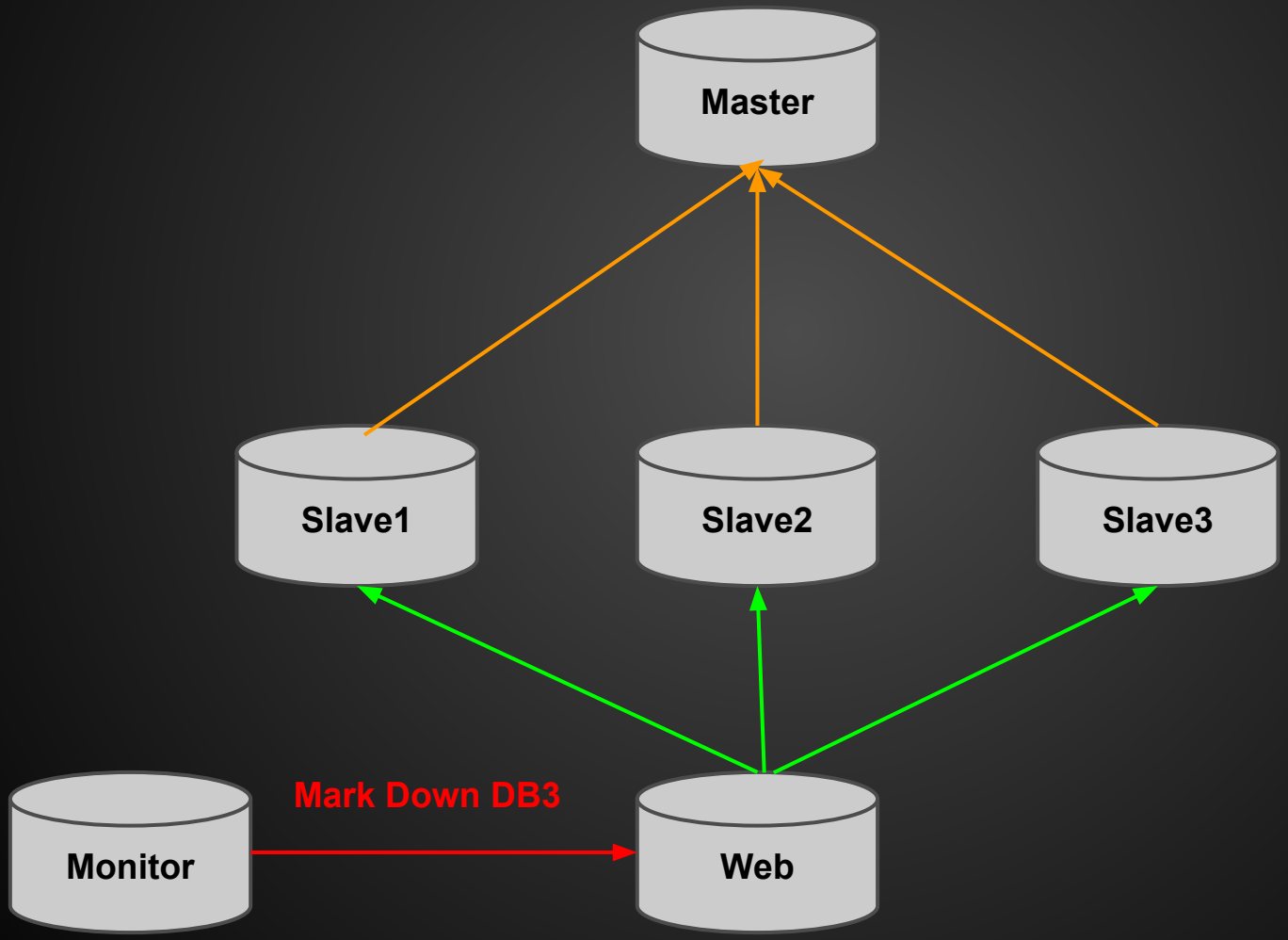
MHA::SlaveFailover

MHA::NodeOnline

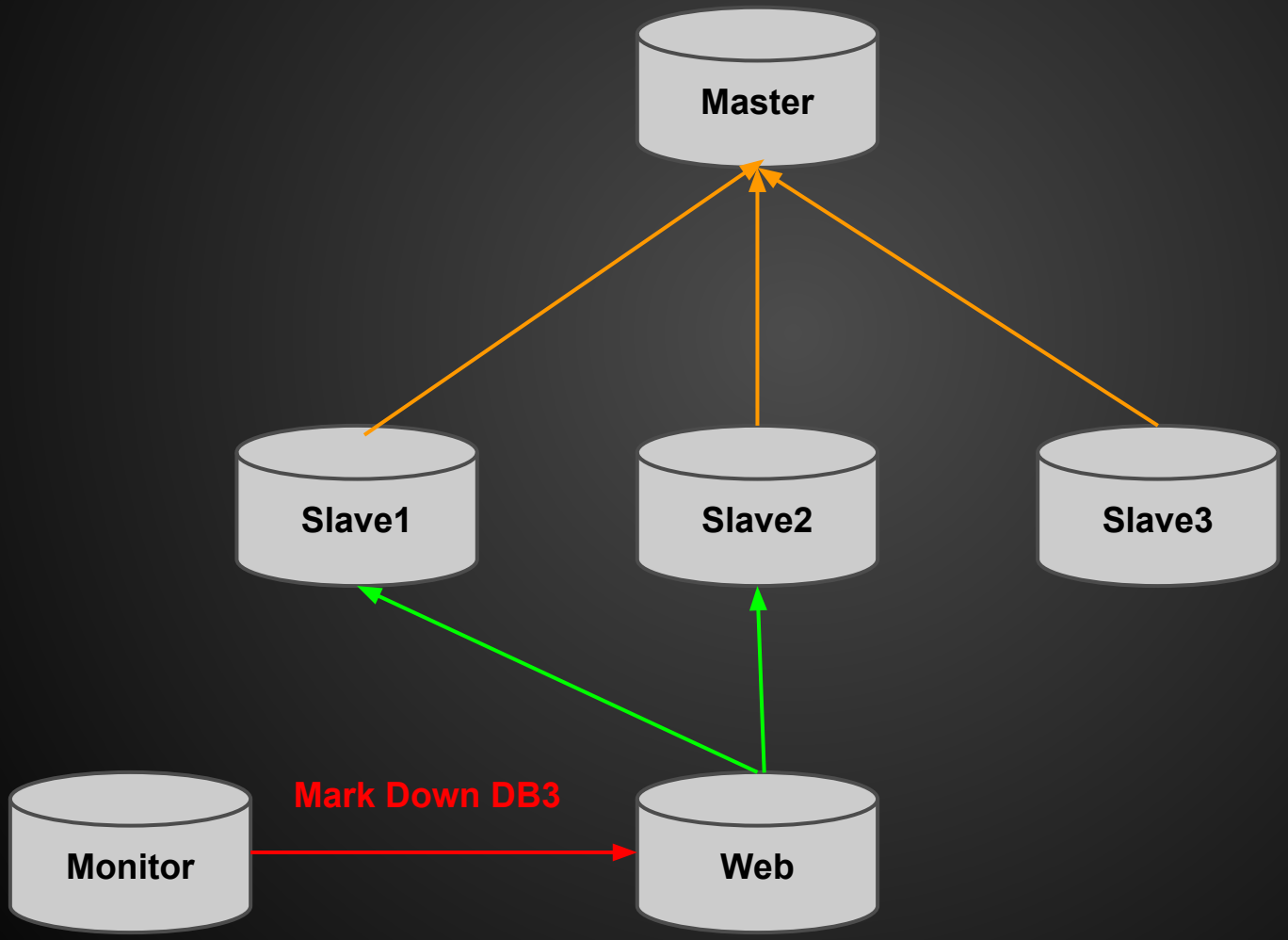
masterha_wrapper

User Defined Script interact with DAL

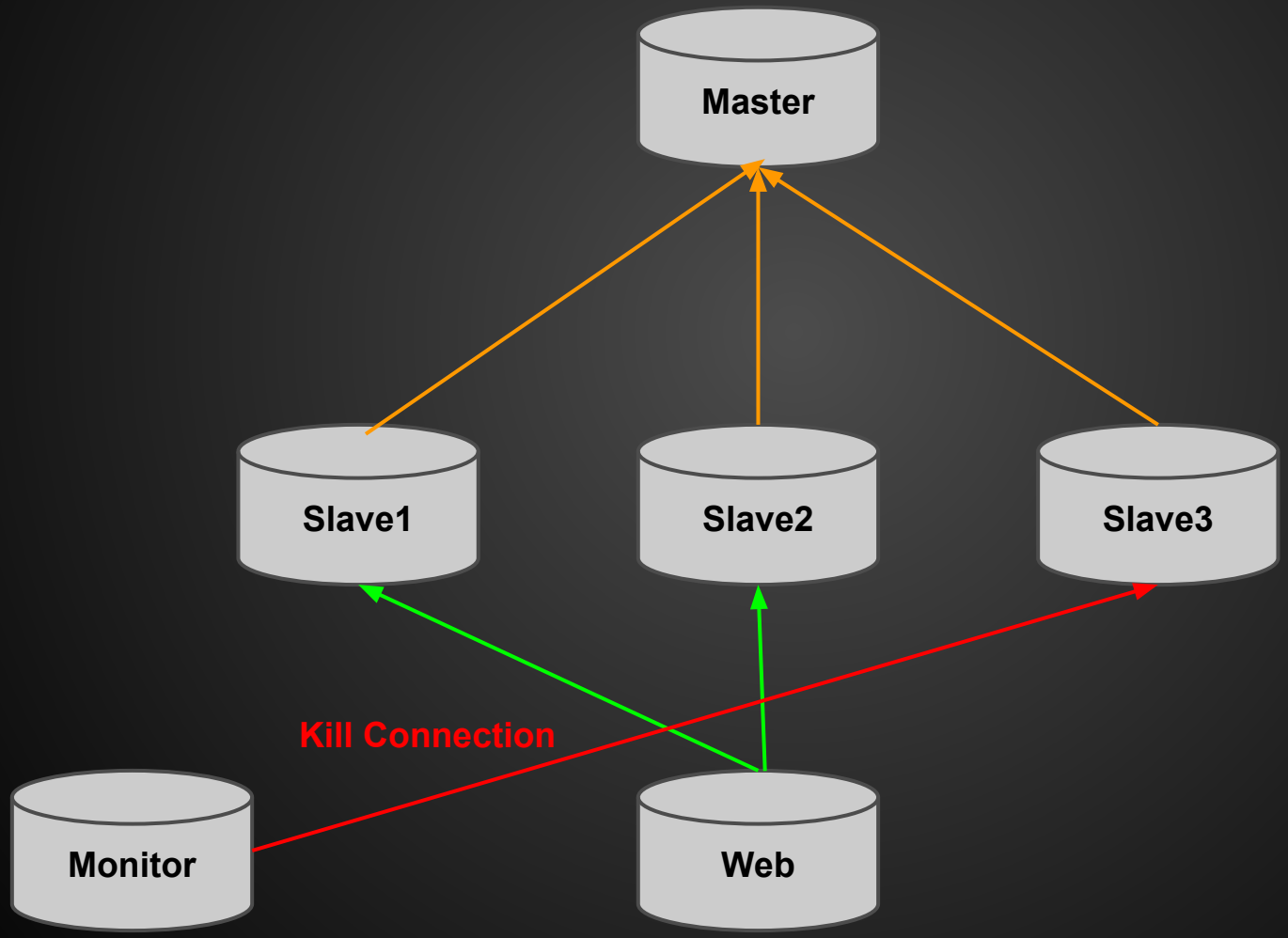
Interact With DAL



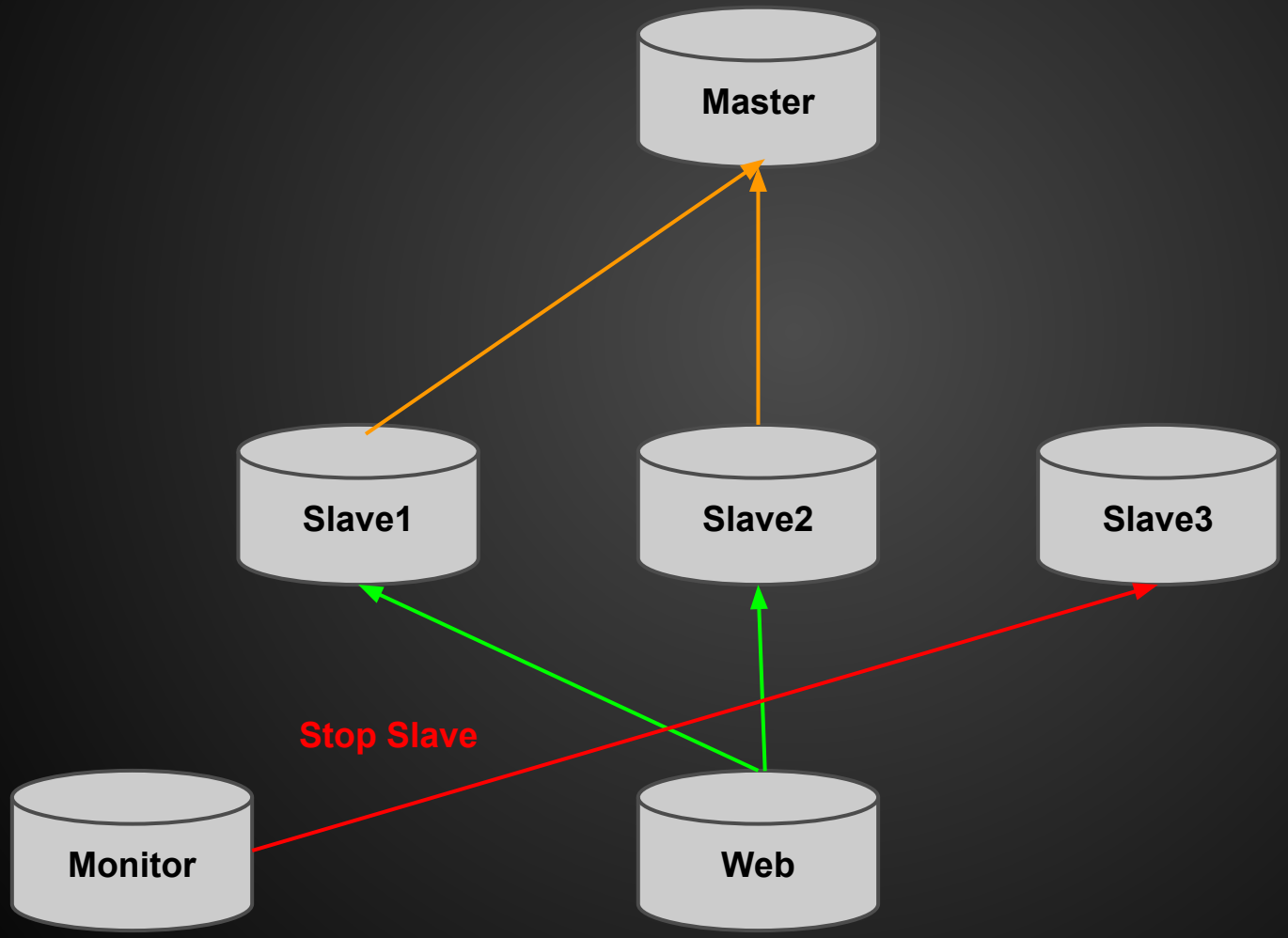
Interact With DAL



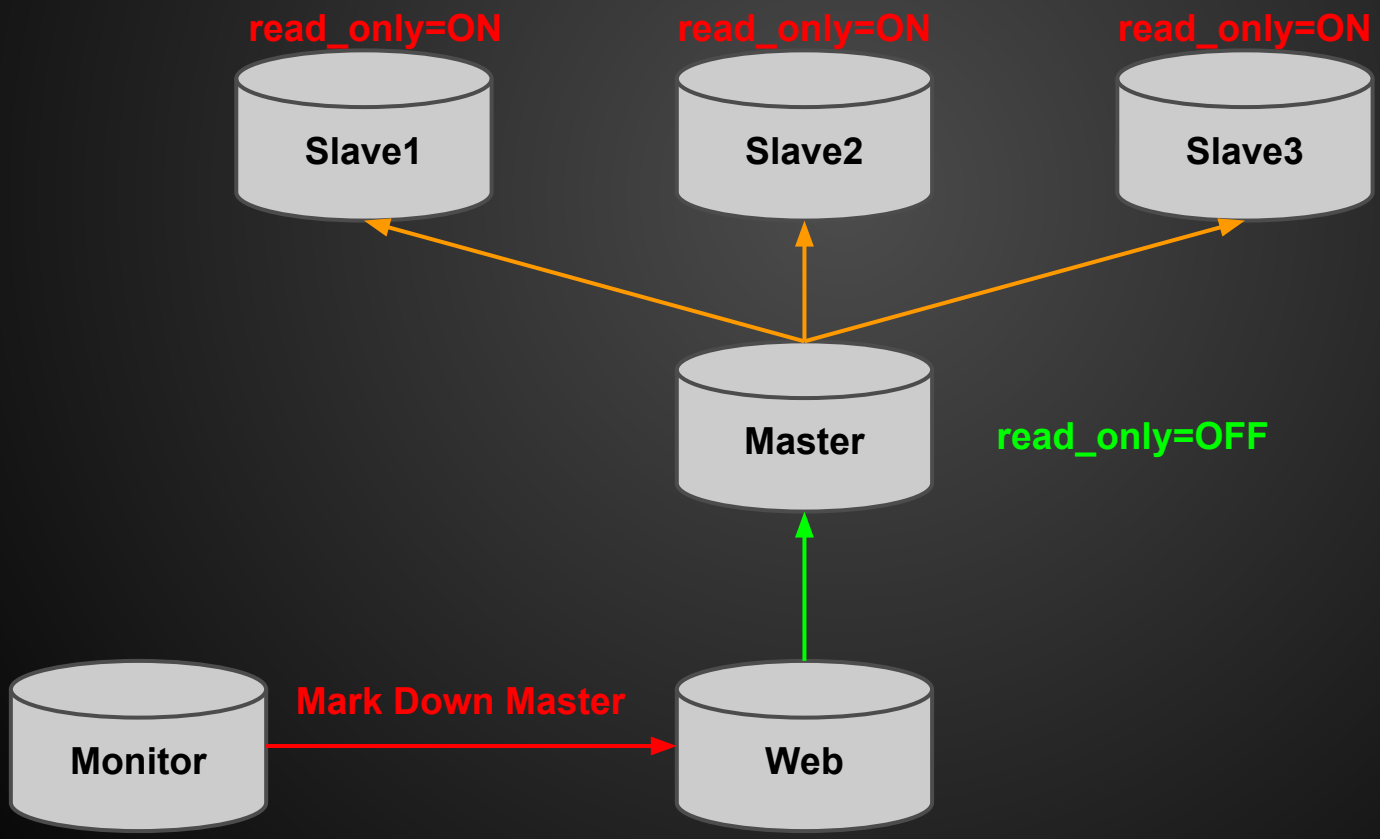
Interact With DAL



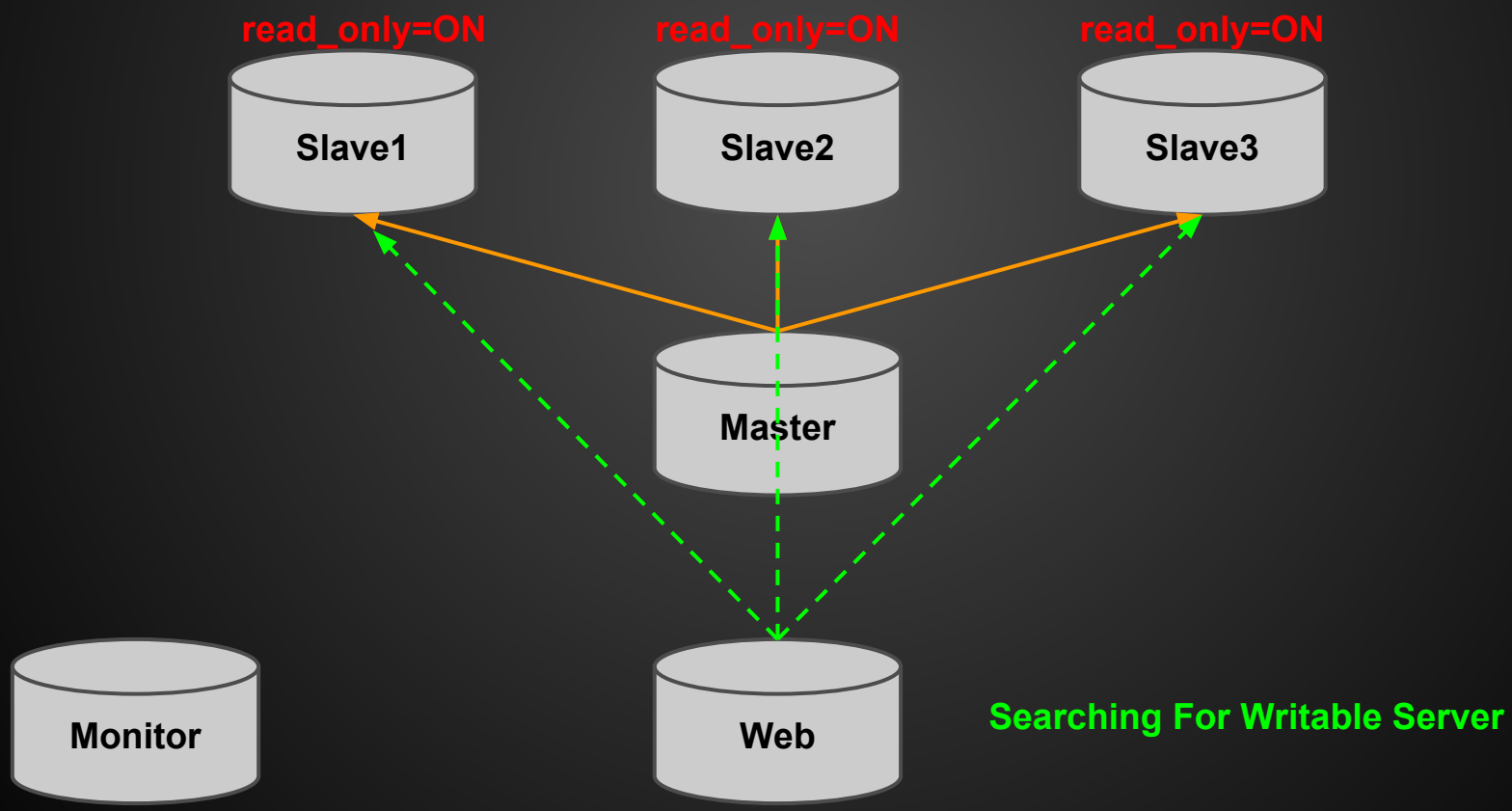
Interact With DAL



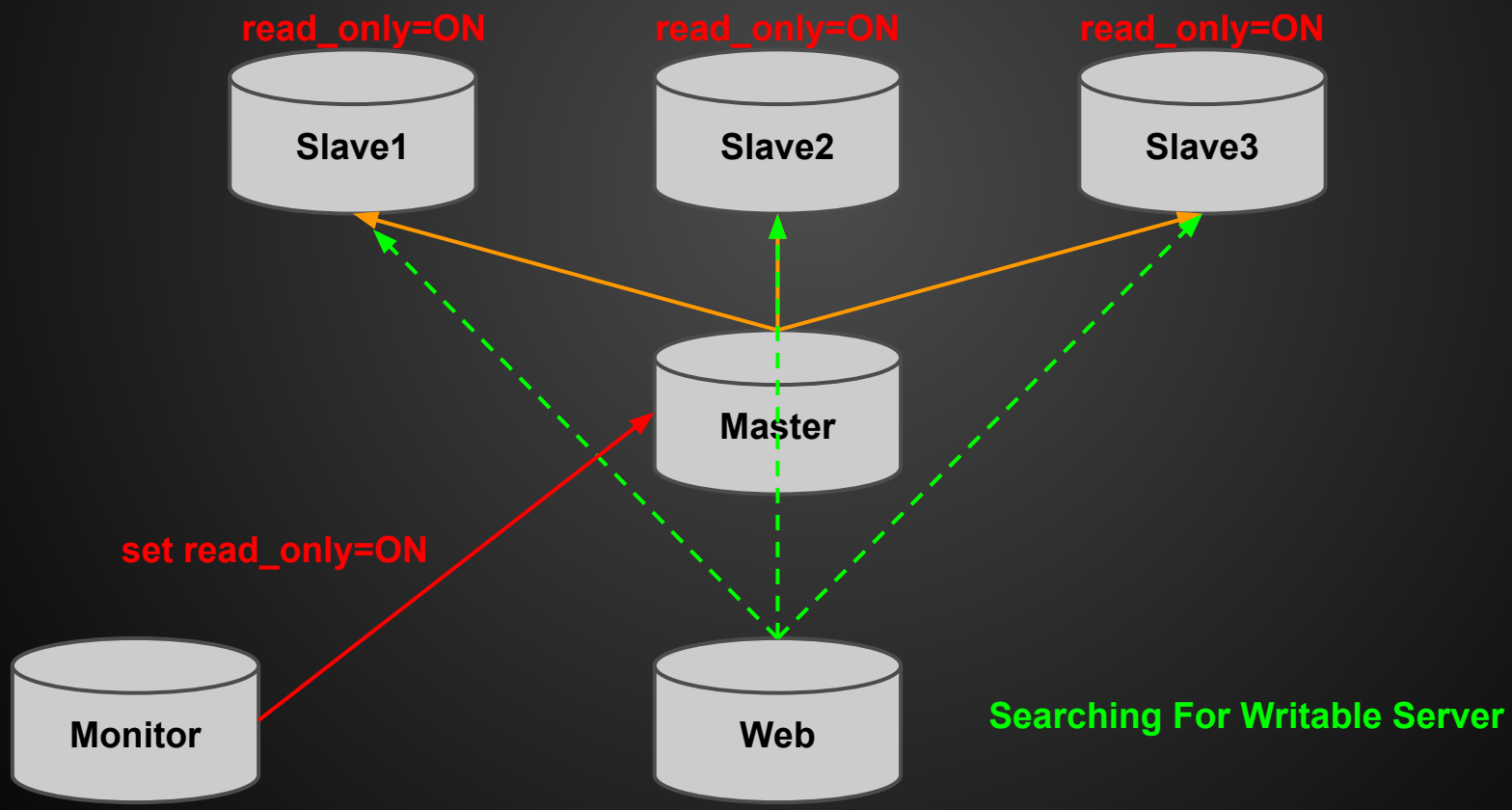
Interact With DAL



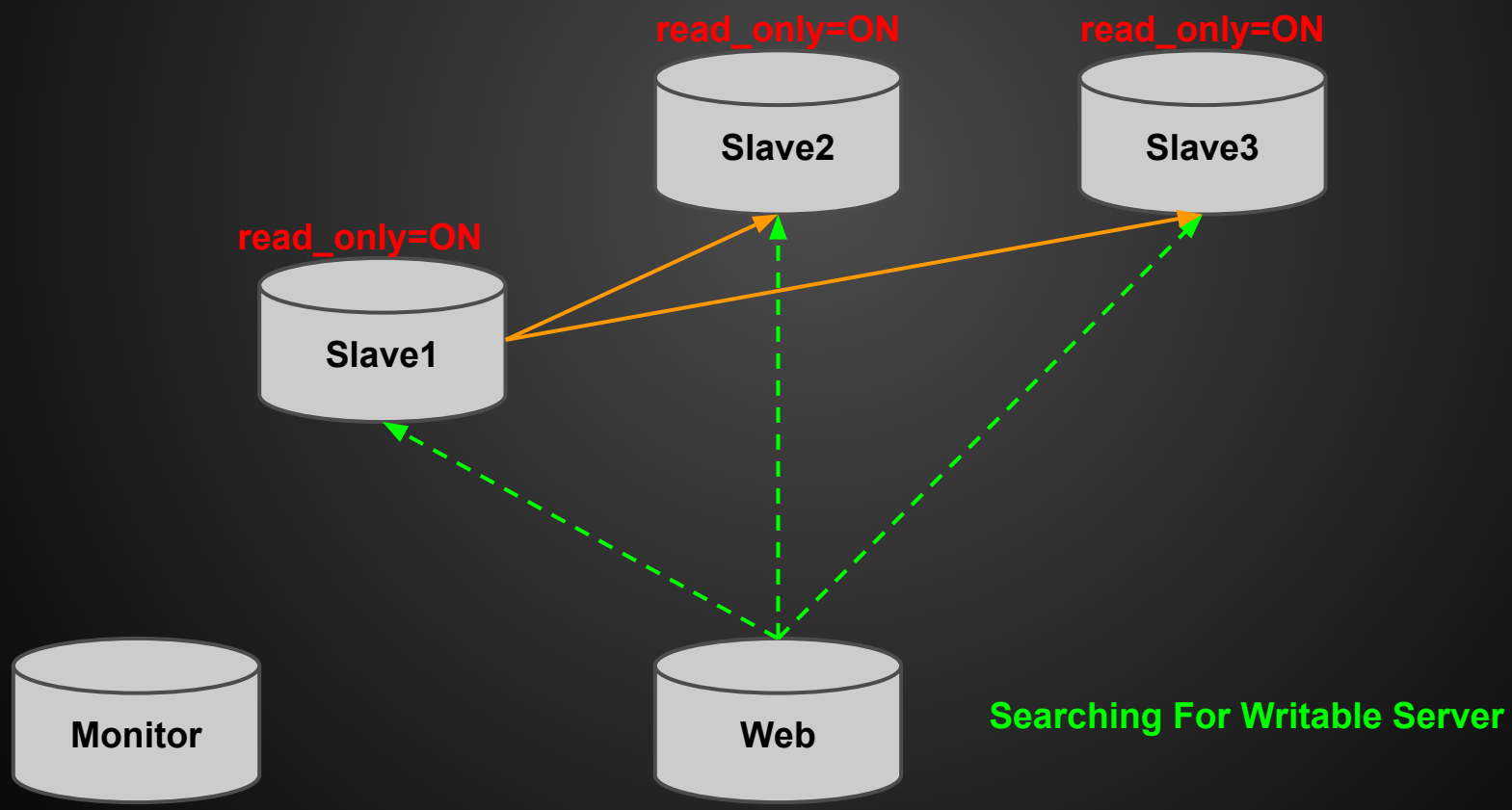
Interact With DAL



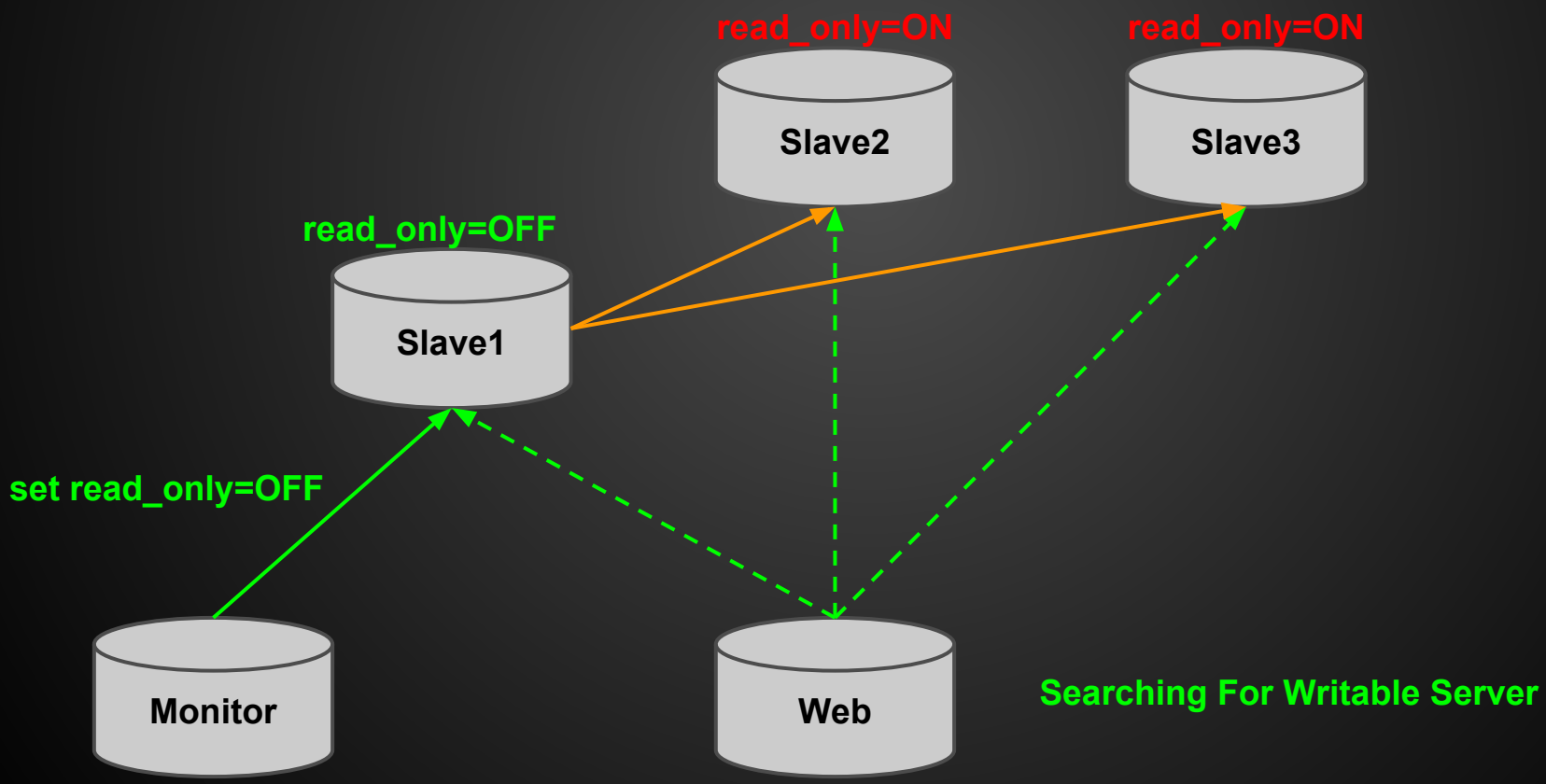
Interact With DAL



Interact With DAL



Interact With DAL



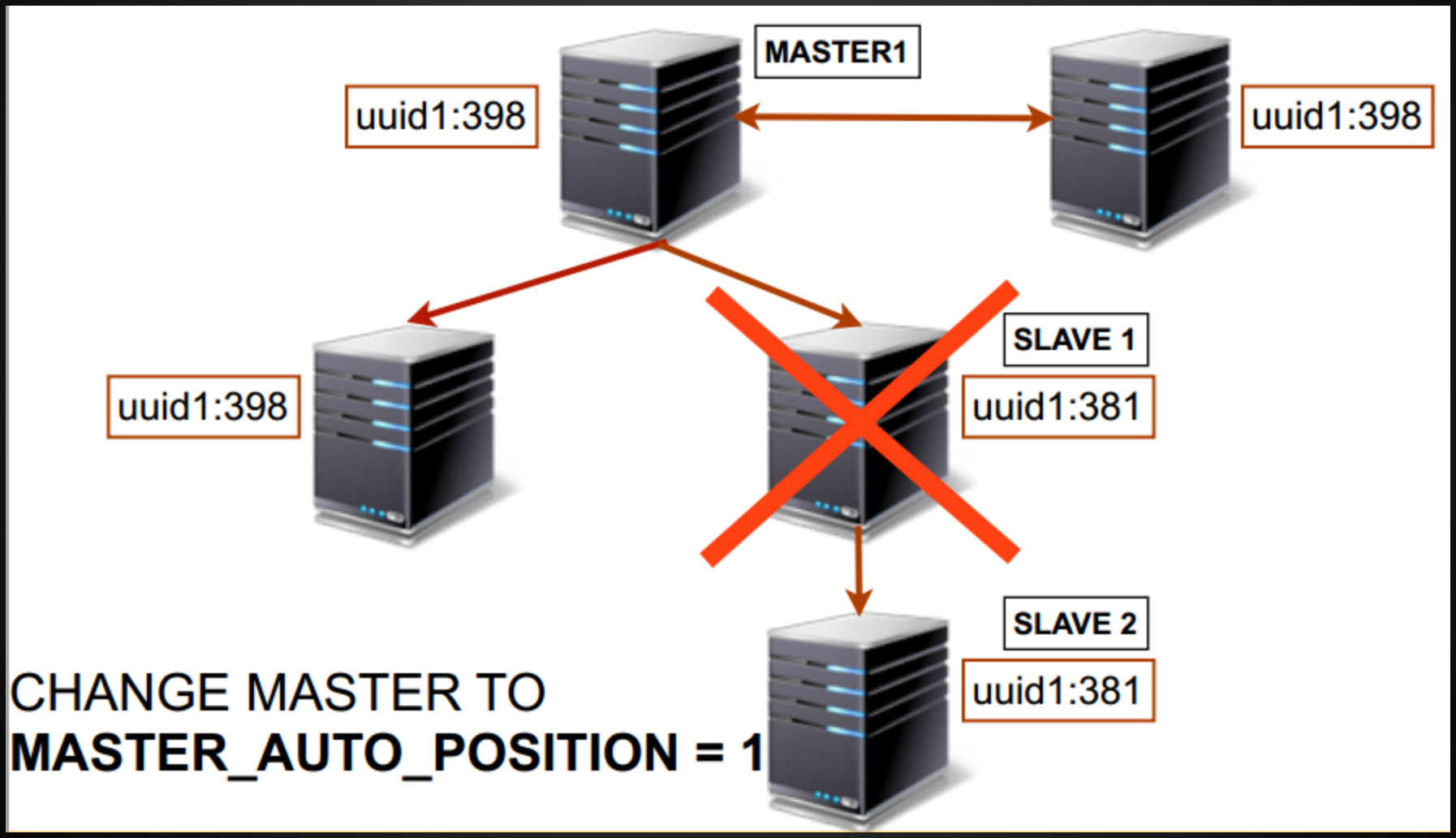
Future

- GTID + mysqlfailover
- Percona lock_binlog_for_backup

GTID

- Possible to identify a transaction uniquely across the replication servers.
- Make the automation of failover process much easier. No calculations!
- At application level it is easier to do WRITE/READ split.
- Development of new automation tools isn’t a pain now.

GTID


MASTER1
uuid1:398
uuid1:398
uuid1:398
SLAVE 1
uuid1:381
SLAVE 2
uuid1:381
CHANGE MASTER TO
MASTER_AUTO_POSITION = 1

Flush Tables With Read Lock

1. Invalidates the Query Cache.
2. Waits for all in-flight updates to complete and at the same time it blocks all incoming updates.
3. Closes all open tables and expels them from the table cache. Wait for all SELECT queries to complete. Incoming SELECT queries will get blocked.
4. Blocks COMMITs.

Reference:
http://www.mysqlperformanceblog.com/2014/03/11/introducing-backup-locks-percona-server-2/

Lock Binlog For Backup

Unlike FTWRL, the LOCK TABLES FOR BACKUP statement:

* does not invalidate the Query Cache;

* never waits for SELECT queries to complete regardless of the storage engines involved;

* never blocks SELECTs, or updates to InnoDB, Blackhole and Federated tables.

http://www.mysqlperformanceblog.com/2014/03/11/introducing-backup-locks-percona-server-2/

Conclusion

Tips for takeaway

When to Use MMM

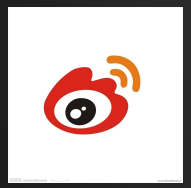
- Simple Master -- Master MySQL deployment environment
- Can afford several seconds data loss
- VIP based HA solution already exist

When to Use MMM - DP version

- Already using MMM
- Slow Slaves included in cluster
- Massive load on master

When to Use MHA

- Cannot afford data loss
- No requirement for Slave High Availability
- Proxy already exists in current deployment
- Interact with client side needed when switch/failover



Q&A

Thanks!

https://github.com/cenalulu/mysql-mmm

卢钧轶/cenalulu @ dianping