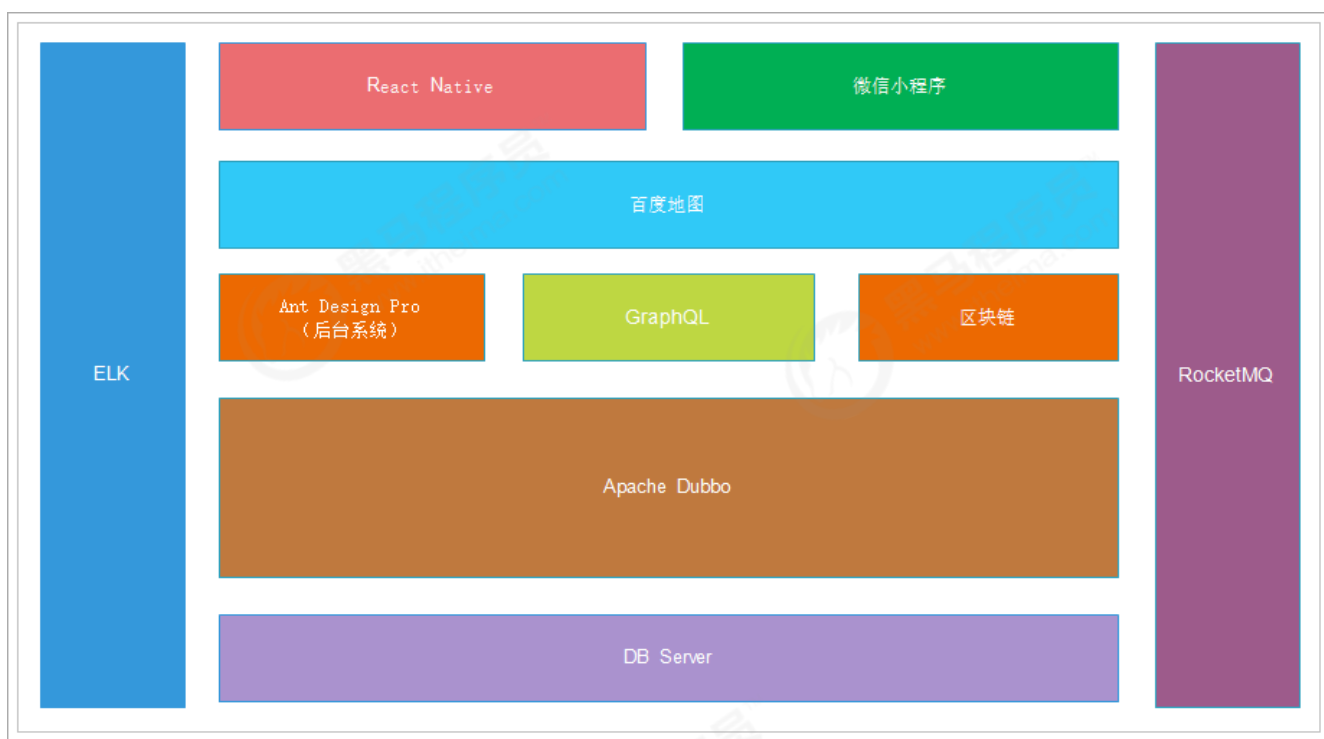


课程介绍

- 分析目前系统架构中的数据库层存在的问题
- 分析MySQL数据库的集群方案
- 学习主从复制（读写分离）架构方案
- 掌握MyCat数据库中间件的使用
- 掌握HAProxy复制均衡的使用
- 掌握PXC集群的使用
- 多种集群架构的综合应用

1、系统架构存在的问题



在我们的系统架构中，DBserver方面我们只是使用了单节点服务，如果面对大并发，海量数据的存储，显然单节点的系统架构将存在很严重的问题，所以接下来，我们将实现MySQL的集群，来应对大并发、海量数据存储等问题。

2、MySQL数据库的集群方案

2.1、读写分离架构

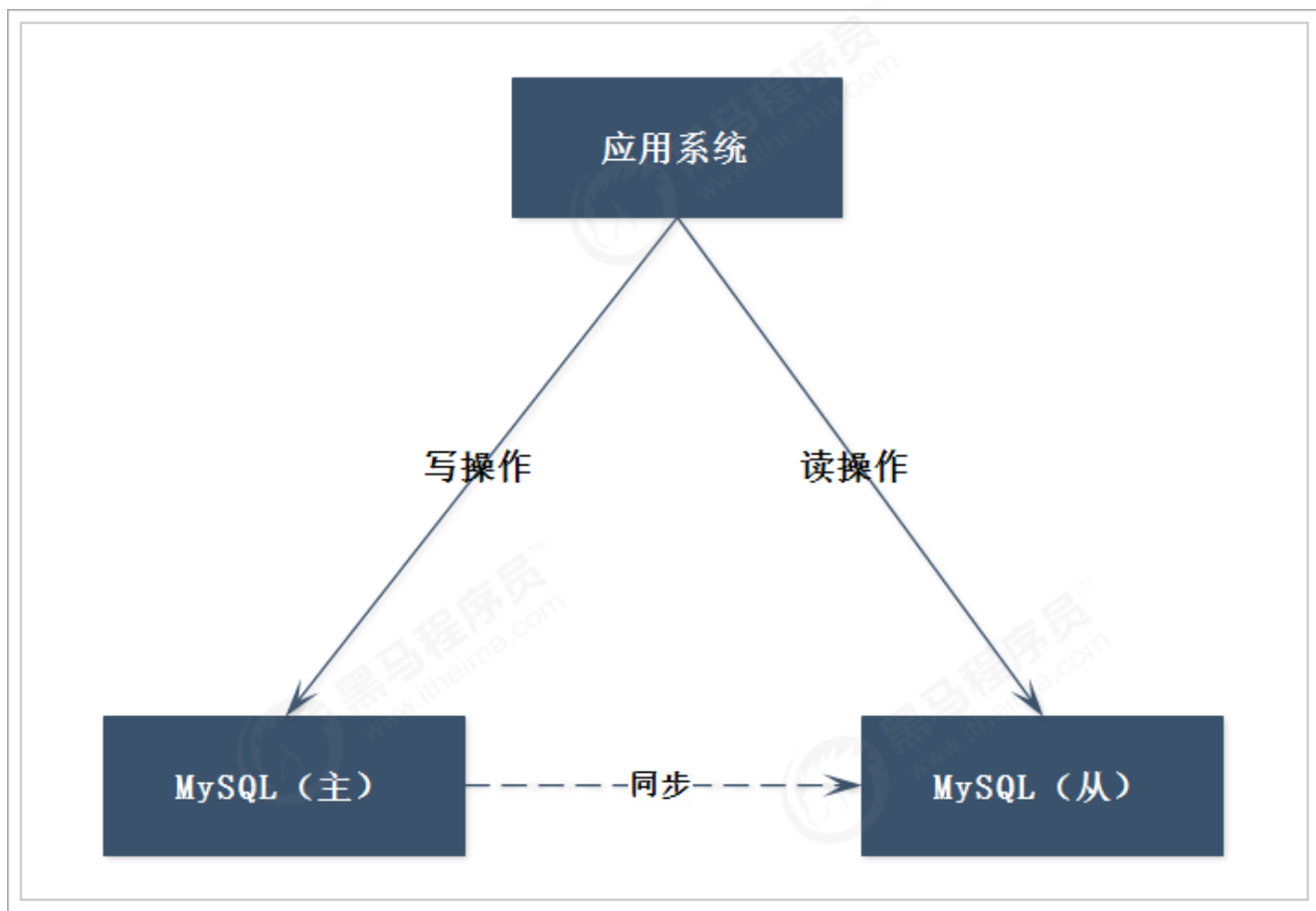
2.1.1、说明

我们一般应用对数据库而言都是“读多写少”，也就说对数据库读取数据的压力比较大，有一个思路就是说采用数据库集群的方案：其中一个为主库，负责写入数据，我们称之为：写库；其它都是从库，负责读取数据，我们称之为：读库；

那么，对我们的要求是：

1. 读库和写库的数据一致；
2. 写数据必须写到写库；
3. 读数据必须到读库；

2.1.2、架构



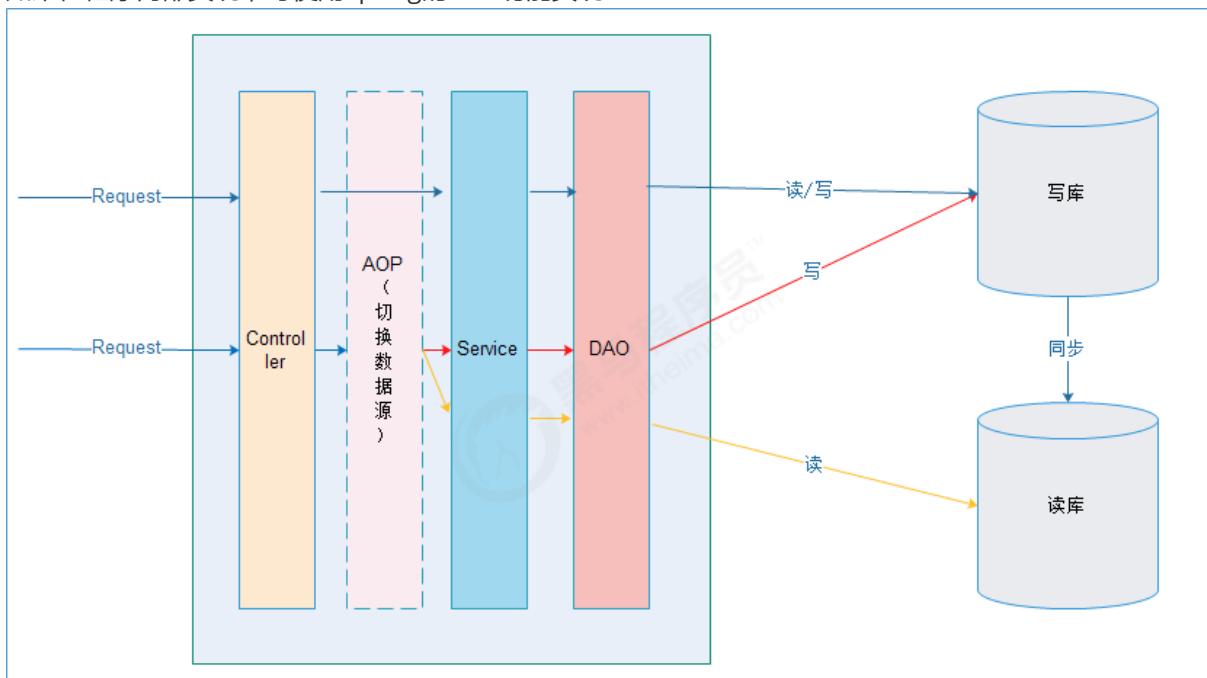
从该系统架构中，可以看出：

- 数据库从之前的单节点变为多节点提供服务
- 主节点数据，同步到从节点数据
- 应用程序需要连接到2个数据库节点，并且在程序内部实现判断读写操作

这种架构存在2个问题：

- 应用程序需要连接到多个节点，对应用程序而言开发变得复杂
 - 这个问题，可以通过中间件解决

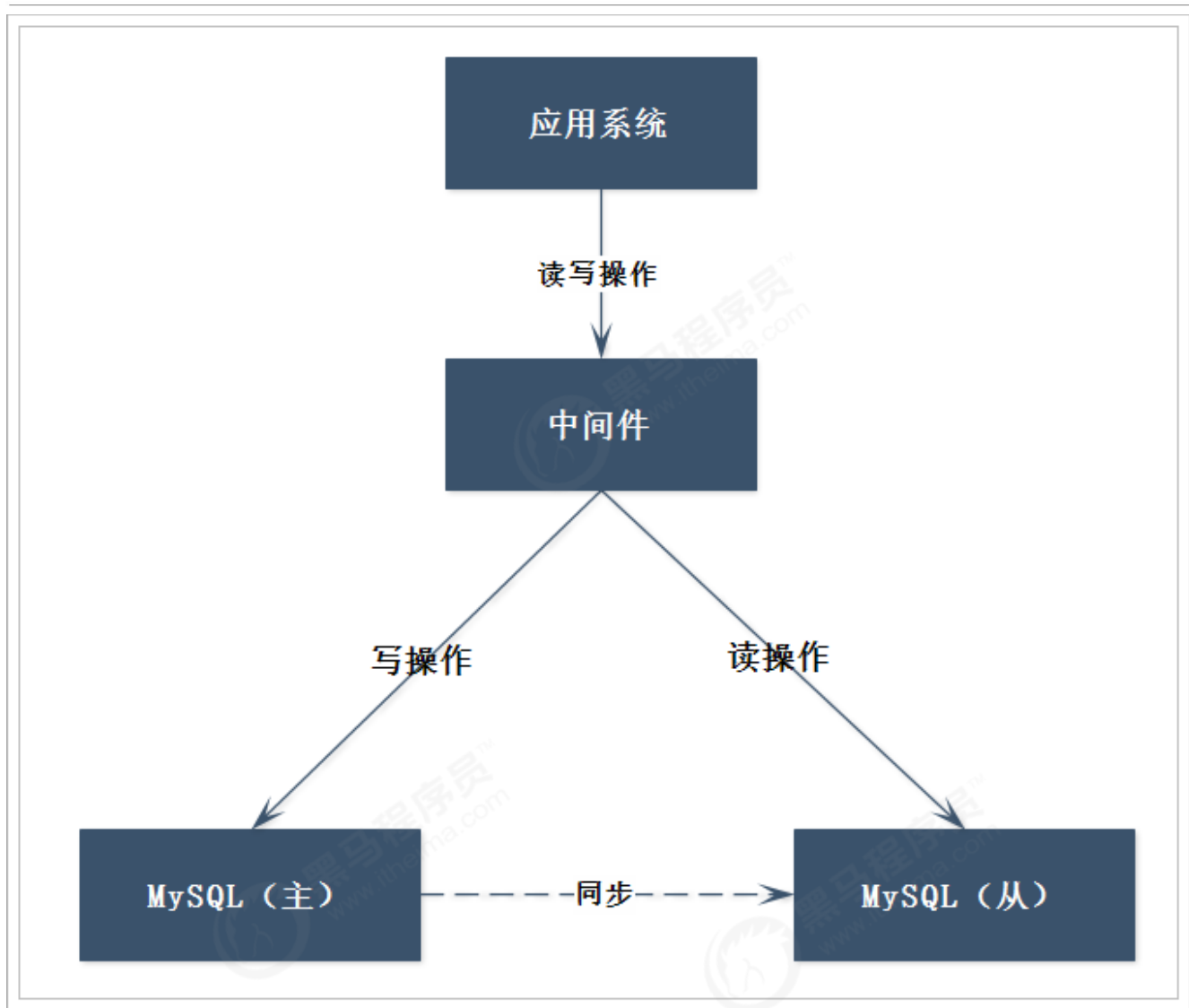
- 如果在程序内部实现，可使用Spring的AOP功能实现



- 主从之间的同步，是异步完成，也就意味着这是 弱一致性
 - 可能会导致，数据写入主库后，应用程序读取从库获取不到数据，或者可能会丢失数据，对于数据安全性要求比较高的应用是不合适的
 - 该问题可以通过PXC集群解决

2.2、中间件

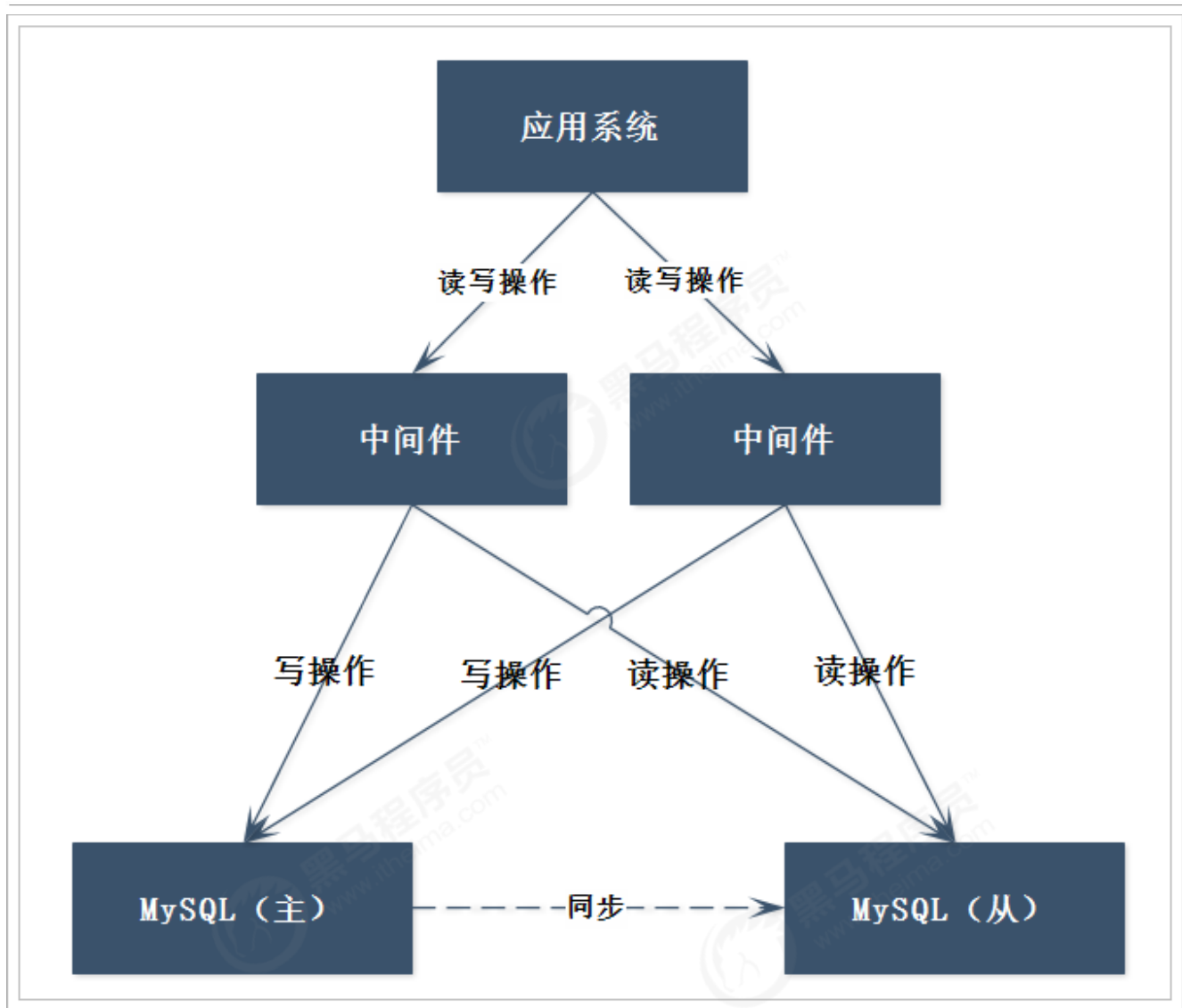
通过上面的架构，可以看出，应用程序会连接到多个节点，使得应用程序的复杂度会提升，可以通过中间件方式解决，如下：



从架构中，可以看出：

- 应用程序只需要连接到中间件即可，无需连接多个数据库节点
- 应用程序无需区分读写操作，对中间件直接进行读写操作即可
- 在中间件中进行区分读写操作，读发送到从节点，写发送到主节点

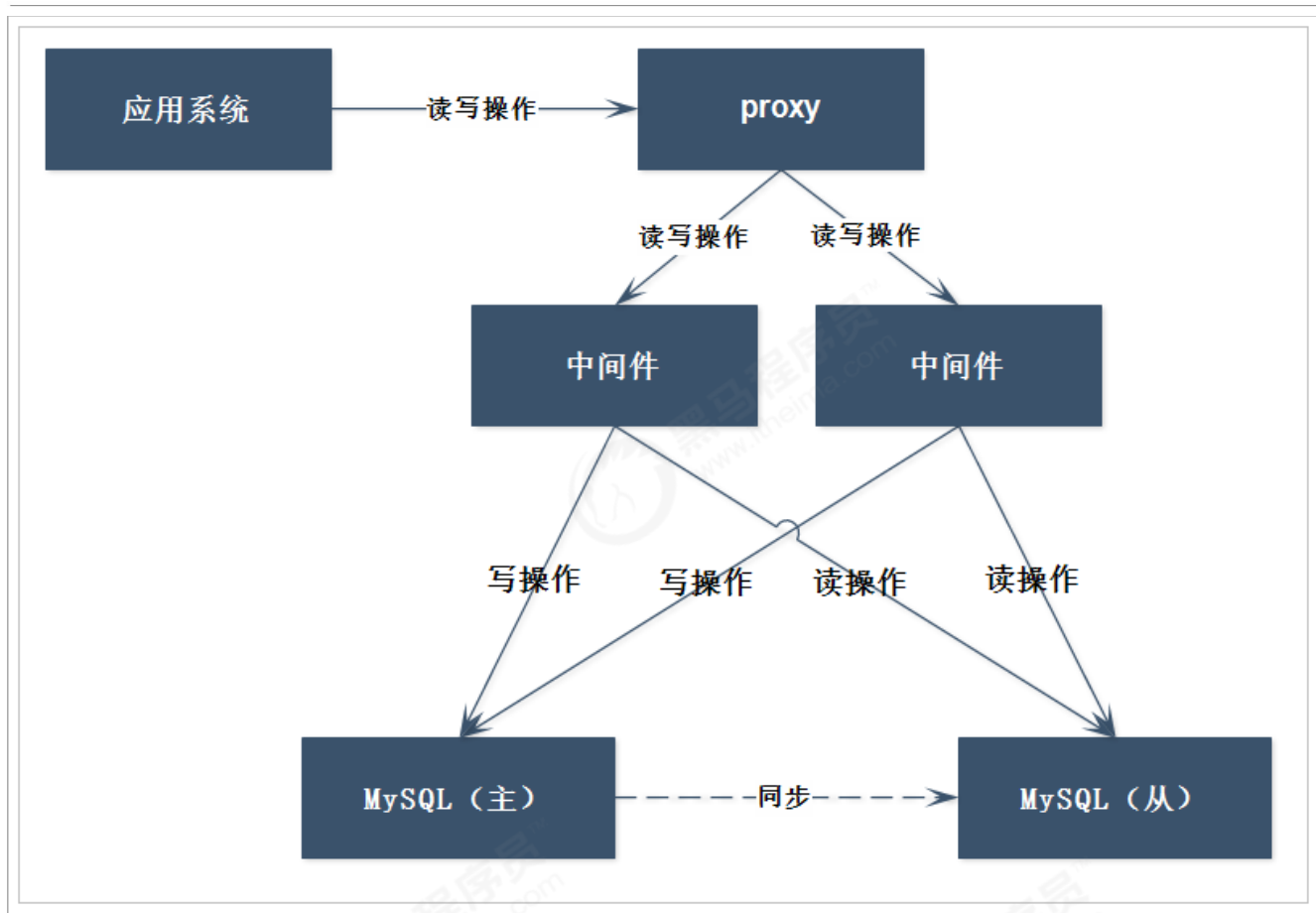
该架构也存在问题，中间件的性能成为了系统的瓶颈，那么架构可以改造成这样：



这样的话，中间件的可靠性得到了保证，但是也带来了新的问题，应用系统依然是需要连接到2个中间件，又为应用系统带来了复杂度。

2.3、负载均衡

为了解决以上问题，我们将继续优化架构，在应用程序和中间件之间增加proxy代理，由代理来完成负载均衡的功能，应用程序只需要对接到proxy即可。



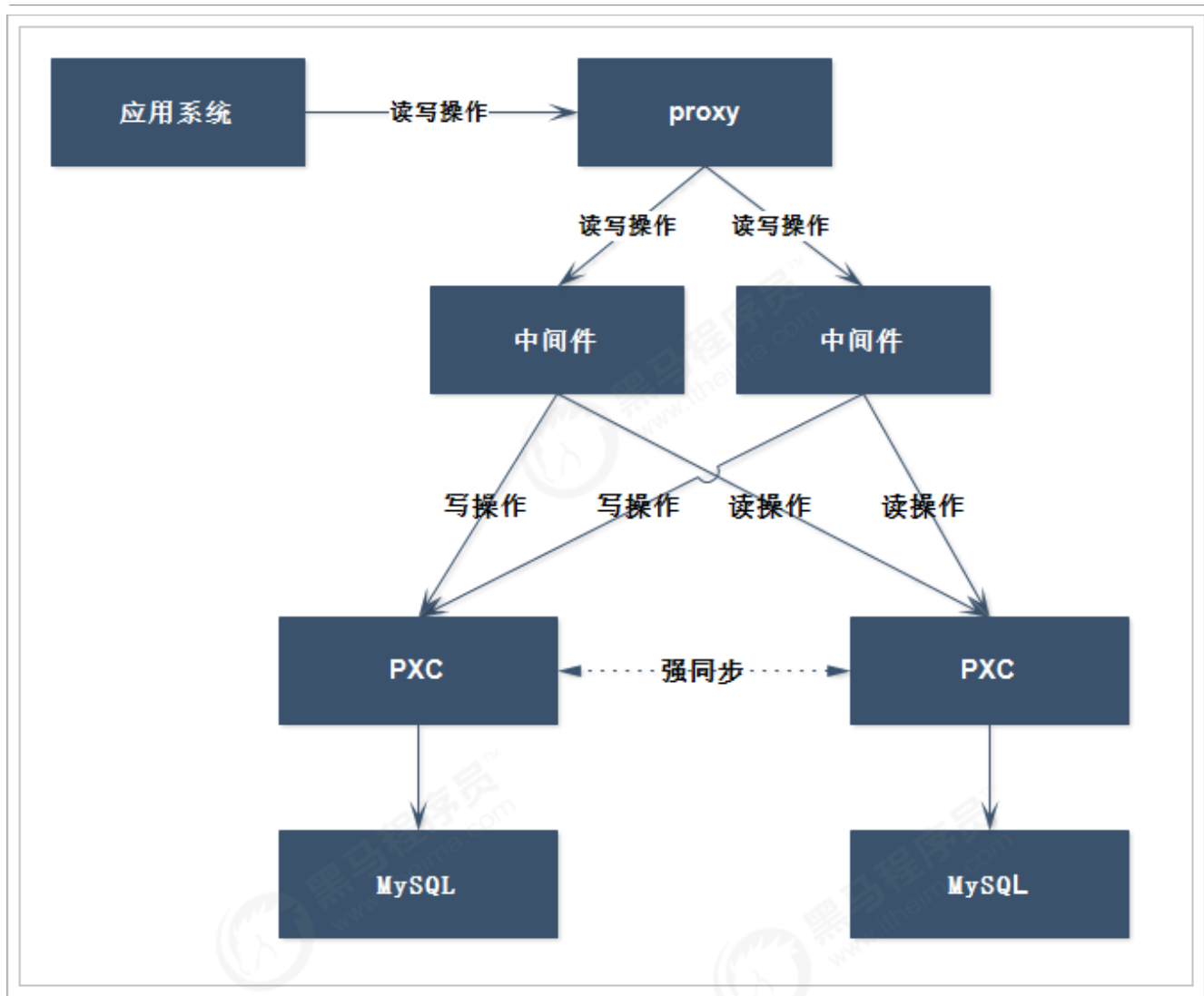
至此，主从复制架构的高可用架构才算是搭建完成。

2.4、PXC集群架构

在前面的架构中，都是基于MySQL主从的架构，那么在主从架构中，弱一致性问题依然没有解决，如果在需要强一致性的需求中，显然这种架构是不能应对的，比如：交易数据。

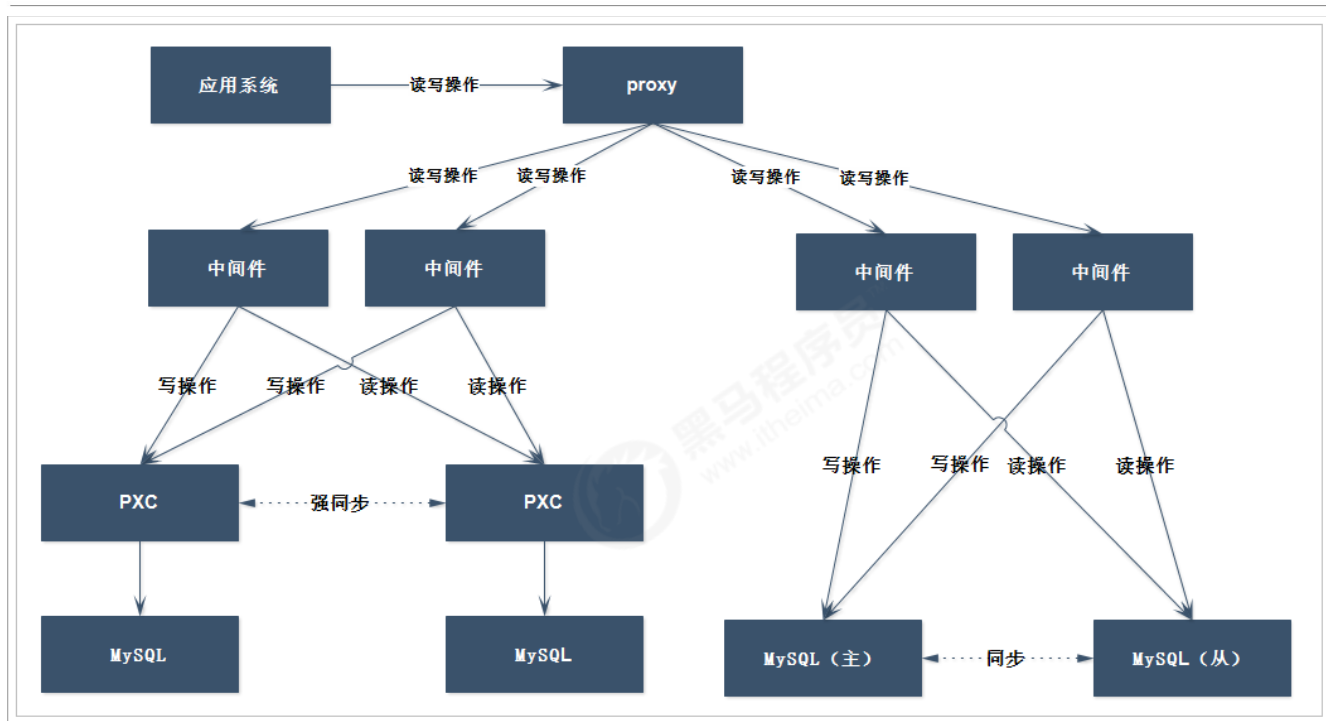
PXC提供了读写强一致性的功能，可以保证数据在任何一个节点写入的同时可以同步到其它节点，也就意味着可以存其它的任何节点进行读取操作，无延迟。

架构如下：



2.5、混合架构

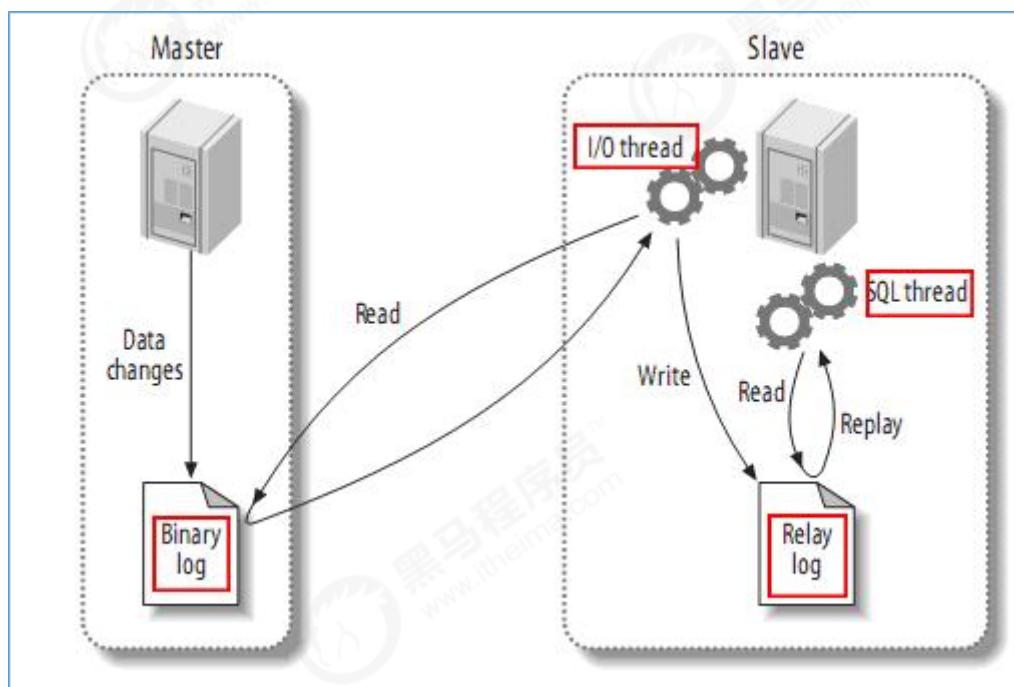
在前面的PXC架构中，虽然可以实现了事务的强一致性，但是它是通过牺牲了性能换来的一致性，如果在某些业务场景下，如果没有强一致性的需求，那么使用PXC就不合适了。所以，在我们的系统架构中，需要将这两种方式综合起来，这样才是一个较为完善的架构。



3、搭建主从复制架构

使用的MySQL版本依然是衍生版Percona，版本为5.7.23。并且通过docker进行搭建服务。

3.1、主从复制原理



mysql主(称master)从(称slave)复制的原理：

- master将数据改变记录到二进制日志(binary log)中,也即是配置文件log-bin指定的文件(这些记录叫做二进制日志事件, binary log events)
- slave将master的binary log events拷贝到它的中继日志(relay log)
- slave重做中继日志中的事件,将改变反映它自己的数据(数据重演)

主从配置需要注意的地方

- 主DB server和从DB server数据库的版本一致
- 主DB server和从DB server数据库数据一致
- 主DB server开启二进制日志,主DB server和从DB server的server_id都必须唯一

3.2、主库配置文件my.conf

```
1 #开启主从复制，主库的配置
2 log-bin = mysql-bin
3 #指定主库serverid
4 server-id=1
5 #指定同步的数据库，如果不指定则同步全部数据库
6 binlog-do-db=my_test
7
8 #执行SQL语句查询状态
9 SHOW MASTER STATUS
10
```

3.3、在主库创建同步用户

```
1 #授权用户slave01使用123456密码登录mysql
2 grant replication slave on *.* to 'slave01'@'127.0.0.1' identified by '123456';
3 #刷新配置
4 flush privileges;
```

3.4、从库配置文件my.conf

```
1 #指定serverid，只要不重复即可，从库也只有这一个配置，其他都在SQL语句中操作
2 server-id=2
3
4 #以下执行SQL：
5 CHANGE MASTER TO
6     master_host='127.0.0.1',
7     master_user='slave01',
8     master_password='123456',
9     master_port=3306,
10    master_log_file='mysql-bin.000006',
11    master_log_pos=1120;
12
13 #启动slave同步
14 START SLAVE;
15
16 #查看同步状态
17 SHOW SLAVE STATUS;
```

3.5、搭建主库

```
1 #创建目录
2 mkdir /data/mysql/master01
```



```
3 cd /data/mysql/master01
4 mkdir conf data
5 chmod 777 * -R
6
7 #创建配置文件
8 cd /data/mysql/master01/conf
9 vim my.cnf
10
11 #输入如下内容
12 [mysqld]
13 log-bin=mysql-bin #开启二进制日志
14 server-id=1 #服务id, 不可重复
15
16 #创建容器
17 docker create --name percona-master01 -v /data/mysql/master01/data:/var/lib/mysql -v
/data/mysql/master01/conf:/etc/my.cnf.d -p 3306:3306 -e MYSQL_ROOT_PASSWORD=root
percona:5.7.23
18
19 #启动
20 docker start percona-master01 && docker logs -f percona-master01
21
22 #创建同步账户以及授权
23 create user 'itcast'@'%' identified by 'itcast';
24 grant replication slave on *.* to 'itcast'@'%';
25 flush privileges;
26
27 #出现 [Err] 1055 - Expression #1 of ORDER BY clause is not in GROUP BY clause and错误解
决方案, 在my.cnf配置文件中设置
28 sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO
,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION'
29
30 #查看master状态
31 show master status;
32
33 #查看二进制日志相关的配置项
34 show global variables like 'binlog%';
35
36 #查看server相关的配置项
37 show global variables like 'server%';
```

master状态：

20				
信息	结果1	概况	状态	
File	Position	Binlog_Do_DB	Binlog_Ignore_DB	Executed_Gtid_Set
mysql-b 648				

14 `show global variables like 'binlog%';`

Variable_name	Value
binlog_cache_size	32768
binlog_checksum	CRC32
binlog_direct_non_transac	OFF
binlog_error_action	ABORT_SERVER
binlog_format	ROW
binlog_group_commit_syr	0
binlog_group_commit_syr	0
binlog_gtid_simple_recover	ON
binlog_max_flush_queue_t	0
binlog_order_commits	ON
binlog_row_image	FULL
binlog_rows_query_log_ev	OFF
binlog_space_limit	0
binlog_stmt_cache_size	32768
binlog_transaction_depen	25000

15
16 `show global variables like 'server%';`
17

Variable_name	Value
server_id	1
server_id_bits	32
server_uuid	444b04d6-249d-11e9-

3.6、搭建从库

```

1  #创建目录
2  mkdir /data/mysql/slave01
3  cd /data/mysql/slave01
4  mkdir conf data
5  chmod 777 * -R
6
7  #创建配置文件
8  cd /data/mysql/slave01/conf
9  vim my.cnf
10
11 #输入如下内容
12 [mysqld]
13 server-id=2 #服务id, 不可重复

```



```
14 sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO
15 ,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION'
16 #创建容器
17 docker create --name percona-slave01 -v /data/mysql/slave01/data:/var/lib/mysql -v
18 /data/mysql/slave01/conf:/etc/my.cnf.d -p 3307:3306 -e MYSQL_ROOT_PASSWORD=root
19 percona:5.7.23
20 #启动
21 docker start percona-slave01 && docker logs -f percona-slave01
22 #设置master相关信息
23 CHANGE MASTER TO
24     master_host='192.168.1.18',
25     master_user='itcast',
26     master_password='itcast',
27     master_port=3306,
28     master_log_file='mysql-bin.000002',
29     master_log_pos=648;
30
31 #启动同步
32 start slave;
33
34 #查看master状态
35 show slave status;
36
```

```
1 mysql> show slave status \G
2 ***** 1. row *****
3         Slave_IO_State: Waiting for master to send event
4         Master_Host: 192.168.1.18
5         Master_User: itcast
6         Master_Port: 3306
7         Connect_Retry: 60
8         Master_Log_File: mysql-bin.000002
9         Read_Master_Log_Pos: 1370
10        Relay_Log_File: dfcb7c833aeb-relay-bin.000002
11        Relay_Log_Pos: 1042
12        Relay_Master_Log_File: mysql-bin.000002
13        Slave_IO_Running: Yes
14        Slave_SQL_Running: Yes
15        Replicate_Do_DB:
16        Replicate_Ignore_DB:
17        Replicate_Do_Table:
18        Replicate_Ignore_Table:
19        Replicate_Wild_Do_Table:
20        Replicate_Wild_Ignore_Table:
21                Last_Errno: 0
22                Last_Error:
23                Skip_Counter: 0
24        Exec_Master_Log_Pos: 1370
25        Relay_Log_Space: 1256
26        Until_Condition: None
```



```
27         Until_Log_File:
28         Until_Log_Pos: 0
29         Master_SSL_Allowed: No
30         Master_SSL_CA_File:
31         Master_SSL_CA_Path:
32         Master_SSL_Cert:
33         Master_SSL_Cipher:
34         Master_SSL_Key:
35         Seconds_Behind_Master: 0
36 Master_SSL_Verify_Server_Cert: No
37         Last_IO_Errno: 0
38         Last_IO_Error:
39         Last_SQL_Errno: 0
40         Last_SQL_Error:
41 Replicate_Ignore_Server_Ids:
42         Master_Server_Id: 1
43         Master_UUID: 444b04d6-249d-11e9-affd-0242ac110002
44         Master_Info_File: /var/lib/mysql/master.info
45         SQL_Delay: 0
46         SQL_Remaining_Delay: NULL
47         Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates
48         Master_Retry_Count: 86400
49         Master_Bind:
50         Last_IO_Error_Timestamp:
51         Last_SQL_Error_Timestamp:
52         Master_SSL_Crl:
53         Master_SSL_Crlpath:
54         Retrieved_Gtid_Set:
55         Executed_Gtid_Set:
56         Auto_Position: 0
57         Replicate_Rewrite_DB:
58         Channel_Name:
59         Master_TLS_Version:
60 1 row in set (0.00 sec)
61
```

可以进行创建数据库、表，插入数据等操作，可以发现实现了主从。

3.7、主从复制模式

```
14 show global variables like 'binlog%';
```

信息	结果1	概况	状态
Variable_name	Value		
binlog_cache_size	32768		
binlog_checksum	CRC32		
binlog_direct_non_transac	OFF		
binlog_error_action	ABORT_SERVER		
binlog_format	ROW		
binlog_group_commit_syr	0		
binlog_group_commit_syr	0		
binlog_gtid_simple_recover	ON		
binlog_max_flush_queue_t	0		
binlog_order_commits	ON		
binlog_row_image	FULL		
binlog_rows_query_log_ev	OFF		
binlog_space_limit	0		
binlog_stmt_cache_size	32768		
binlog_transaction_depen	25000		
binlog_transaction_depen	COMMIT_ORDER		

在查看二进制日志相关参数内容中，会发现默认的模式为ROW，其实在MySQL中提供了有3种模式，基于SQL语句的复制(statement-based replication, SBR)，基于行的复制(row-based replication, RBR)，混合模式复制(mixed-based replication, MBR)。对应的，binlog的格式也有三种：STATEMENT，ROW，MIXED。

STATEMENT模式 (SBR)

每一条会修改数据的sql语句会记录到binlog中。

- 优点是并不需要记录每一条sql语句和每一行的数据变化，减少了binlog日志量，节约IO，提高性能。
- 缺点是在某些情况下会导致master-slave中的数据不一致(如sleep()函数，last_insert_id()，以及user-defined functions(udf)等会出现问题)

ROW模式 (RBR)

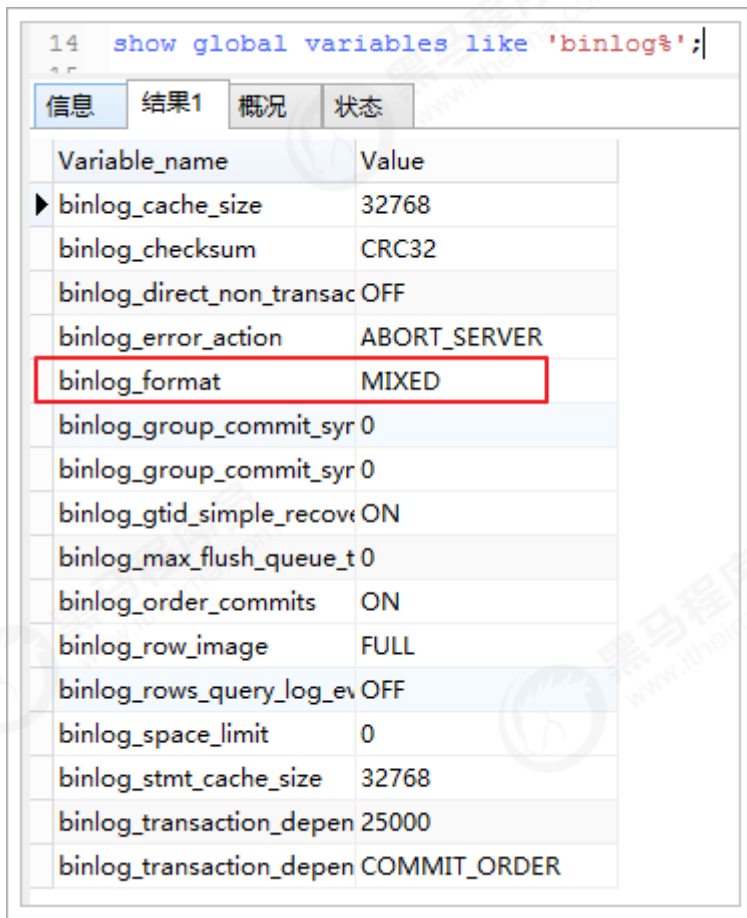
不记录每条sql语句的上下文信息，仅需记录哪条数据被修改了，修改成什么样了。而且不会出现某些特定情况下的存储过程、或function、或trigger的调用和触发无法被正确复制的问题。缺点是会产生大量的日志，尤其是alter table的时候会让日志暴涨。

MIXED模式 (MBR)

以上两种模式的混合使用，一般的复制使用STATEMENT模式保存binlog，对于STATEMENT模式无法复制的操作使用ROW模式保存binlog，MySQL会根据执行的SQL语句选择日志保存方式。

建议使用MIXED模式。

```
1 #修改主库的配置
2 binlog_format=MIXED
3
4 #重启
5 docker restart percona-master01 && docker logs -f percona-master01
6
7 #查看二进制日志相关的配置项
8 show global variables like 'binlog%';
```



14 show global variables like 'binlog%';

信息	结果1	概况	状态
Variable_name	Value		
binlog_cache_size	32768		
binlog_checksum	CRC32		
binlog_direct_non_transac	OFF		
binlog_error_action	ABORT_SERVER		
binlog_format	MIXED		
binlog_group_commit_syr	0		
binlog_group_commit_syr	0		
binlog_gtid_simple_recover	ON		
binlog_max_flush_queue_t	0		
binlog_order_commits	ON		
binlog_row_image	FULL		
binlog_rows_query_log_ev	OFF		
binlog_space_limit	0		
binlog_stmt_cache_size	32768		
binlog_transaction_depen	25000		
binlog_transaction_depen	COMMIT_ORDER		

可以看到，设置已经生效。并且进行测试，同步功能正常。

4、MyCat中间件

4.1、简介

MYCAT 下载 实体书 书籍 测试案例 使用案例 关于我们 开源之星投票 捐赠

Mycat数据库分库分表中间件

国内最活跃的、性能最好的开源数据库中间件！

我们致力于开发高性能的开源中间件而努力！

实体书

Mycat权威指南 »

GitHub地址 »

start »



4,592



加入QQ群

- 一个彻底开源的，面向企业应用开发的大数据库集群
- 支持事务、ACID、可以替代MySQL的加强版数据库
- 一个可以视为MySQL集群的企业级数据库，用来替代昂贵的Oracle集群
- 一个融合内存缓存技术、NoSQL技术、HDFS大数据的新型SQL Server
- 结合传统数据库和新型分布式数据仓库的新一代企业级数据库产品
- 一个新颖的数据库中间件产品

4.2、优势

基于阿里开源的Cobar产品而研发，Cobar的稳定性、可靠性、优秀的架构和性能以及众多成熟的使用案例使得MYCAT一开始就拥有一个很好的起点，站在巨人的肩膀上，我们能看到更远。业界优秀的开源项目和创新思路被广泛融入到MYCAT的基因中，使得MYCAT在很多方面都领先于目前其他一些同类的开源项目，甚至超越某些商业产品。

MYCAT背后有一支强大的技术团队，其参与者都是5年以上资深软件工程师、架构师、DBA等，优秀的技术团队保证了MYCAT的产品质量。

MYCAT并不依托于任何一个商业公司，因此不像某些开源项目，将一些重要的特性封闭在其商业产品中，使得开源项目成了一个摆设。

4.3、关键特性

- 支持SQL92标准
- 支持MySQL、Oracle、DB2、SQL Server、PostgreSQL等DB的常见SQL语法
- 遵守Mysql原生协议，跨语言，跨平台，跨数据库的通用中间件代理。
- 基于心跳的自动故障切换，支持读写分离，支持MySQL主从，以及galera cluster集群。
- 支持Galera for MySQL集群，Percona Cluster或者MariaDB cluster
- 基于Nio实现，有效管理线程，解决高并发问题。
- 支持数据的多片自动路由与聚合，支持sum,count,max等常用的聚合函数,支持跨库分页。
- 支持单库内部任意join，支持跨库2表join，甚至基于caltlet的多表join。
- 支持通过全局表，ER关系的分片策略，实现了高效的多表join查询。
- 支持多租户方案。

- 支持分布式事务（弱xa）。
- 支持XA分布式事务（1.6.5）。
- 支持全局序列号，解决分布式下的主键生成问题。
- 分片规则丰富，插件化开发，易于扩展。
- 强大的web，命令行监控。
- 支持前端作为MySQL通用代理，后端DBC方式支持Oracle、DB2、SQL Server、mongodb、巨杉。
- 支持密码加密
- 支持服务降级
- 支持IP白名单
- 支持SQL黑名单、sql注入攻击拦截
- 支持prepare预编译指令（1.6）
- 支持非堆内存(Direct Memory)聚合计算（1.6）
- 支持PostgreSQL的native协议（1.6）
- 支持mysql和oracle存储过程，out参数、多结果集返回（1.6）
- 支持zookeeper协调主从切换、zk序列、配置zk化（1.6）
- 支持库内分表（1.6）
- 集群基于ZooKeeper管理，在线升级，扩容，智能优化，大数据处理（2.0开发版）。

4.4、读写分离

MySQL服务部署情况：

主机	端口	容器名称	角色
192.168.1.18	3306	percona-master01	master
192.168.1.18	3307	percona-slave01	slave

server.xml：

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE mycat:server SYSTEM "server.dtd">
3  <mycat:server xmlns:mycat="http://io.mycat/">
4      <system>
5          <property name="nonePasswordLogin">0</property>
6          <property name="useHandshakeV10">1</property>
7          <property name="useSqlStat">0</property>
8          <property name="useGlobleTableCheck">0</property>
9          <property name="sequenceHandlerType">2</property>
10         <property name="subqueryRelationshipCheck">>false</property>
11         <property name="processorBufferPoolType">0</property>
12         <property name="handleDistributedTransactions">0</property>
13         <property name="useOffHeapForMerge">1</property>
14         <property name="memoryPageSize">64k</property>
15         <property name="spillsFileBufferSize">1k</property>
16         <property name="useStreamOutput">0</property>
17         <property name="systemReserveMemorySize">384m</property>
18         <property name="useZKSwitch">>false</property>
19     </system>
    
```



```

20      <!--这里是设置的itcast用户和虚拟逻辑库-->
21      <user name="itcast" defaultAccount="true">
22          <property name="password">itcast123</property>
23          <property name="schemas">itcast</property>
24      </user>
25  </mycat:server>

```

schema.xml :

```

1  <?xml version="1.0"?>
2  <!DOCTYPE mycat:schema SYSTEM "schema.dtd">
3  <mycat:schema xmlns:mycat="http://io.mycat/">
4      <!--配置数据表-->
5      <schema name="itcast" checkSQLSchema="false" sqlMaxLimit="100">
6          <table name="tb_ad" dataNode="dn1" rule="mod-long" />
7      </schema>
8      <!--配置分片关系-->
9      <dataNode name="dn1" dataHost="cluster1" database="itcast" />
10     <!--配置连接信息-->
11     <dataHost name="cluster1" maxCon="1000" minCon="10" balance="3"
12         writeType="1" dbType="mysql" dbDriver="native" switchType="1"
13         slaveThreshold="100">
14         <heartbeat>select user()</heartbeat>
15         <writeHost host="w1" url="192.168.1.18:3306" user="root"
16             password="root">
17             <readHost host="w1r1" url="192.168.1.18:3307" user="root"
18                 password="root" />
19         </writeHost>
20     </dataHost>
21 </mycat:schema>

```

balance属性说明：

负载均衡类型，目前的取值有3种：

1. balance="0", 不开启读写分离机制，所有读操作都发送到当前可用的writeHost上。
2. balance="1", 全部的readHost与stand by writeHost参与select语句的负载均衡，简单的说，当双主双从模式(M1->S1, M2->S2, 并且M1与M2互为主备)，正常情况下，M2,S1,S2都参与select语句的负载均衡。
3. balance="2", 所有读操作都随机的在writeHost、readhost上分发。
4. balance="3", 所有读请求随机的分发到writerHost对应的readhost执行，writerHost不负担读压力，注意balance=3只在1.4及其以后版本有，1.3没有。

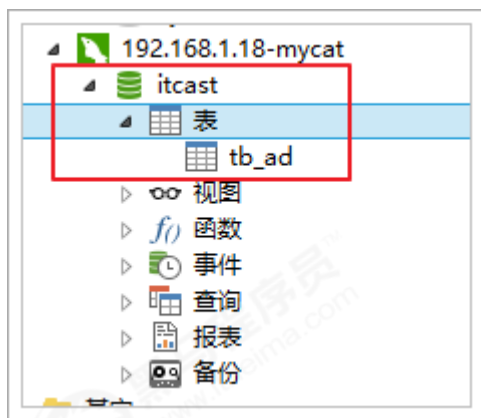
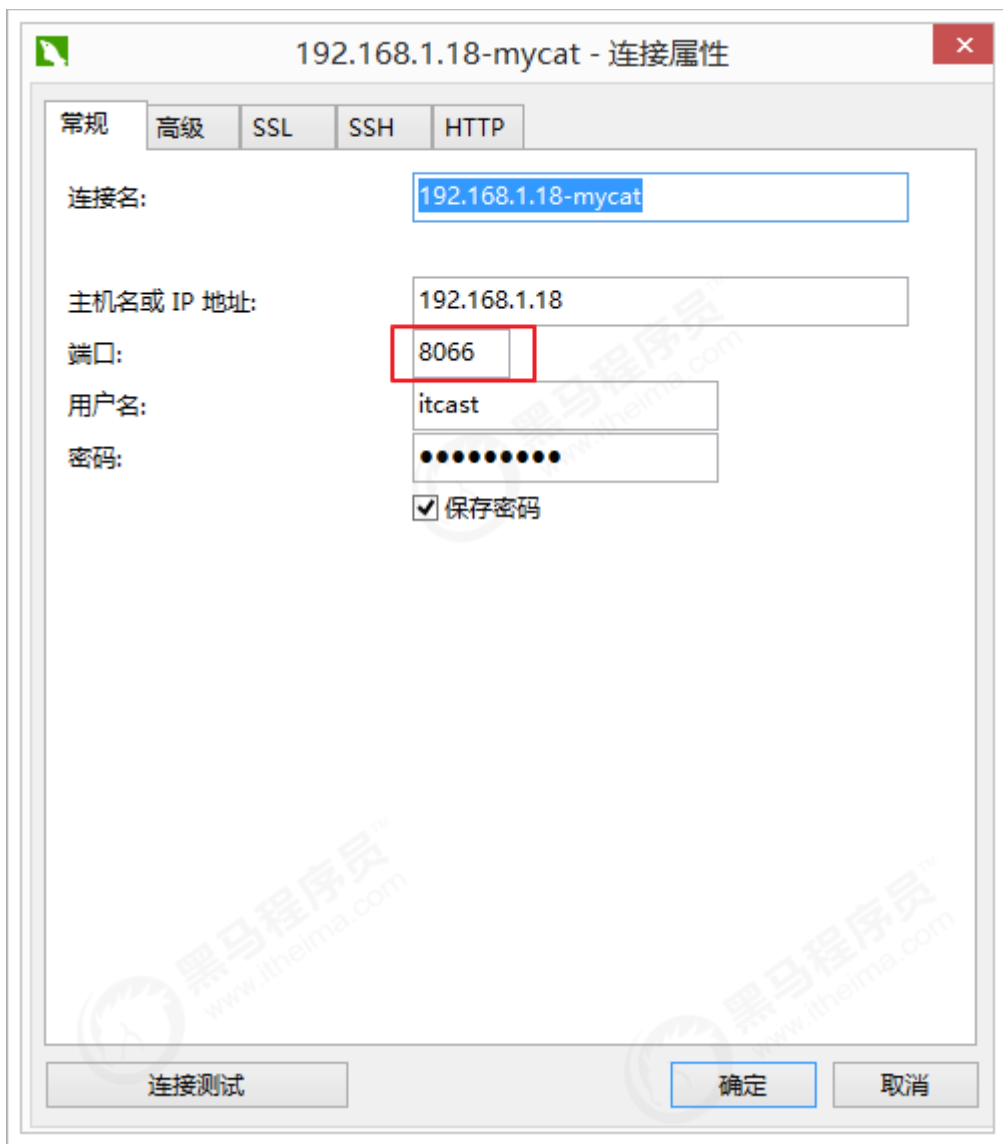
rule.xml :

```

1  <function name="mod-long" class="io.mycat.route.function.PartitionByMod">
2      <property name="count">1</property>
3  </function>

```

测试：



```
1  --创建表
2  CREATE TABLE `tb_ad` (
3    `id` bigint(20) NOT NULL AUTO_INCREMENT,
4    `type` int(10) DEFAULT NULL COMMENT '广告类型',
5    `title` varchar(100) DEFAULT NULL COMMENT '描述',
6    `url` varchar(200) DEFAULT NULL COMMENT '图片URL地址',
7    `created` datetime DEFAULT NULL,
8    `updated` datetime DEFAULT NULL,
```

```

9      PRIMARY KEY (`id`)
10 ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='广告表';
11
12 --测试插入数据
13 INSERT INTO `tb_ad` (`id`, `type`, `title`, `url`, `created`, `updated`) VALUES ('1',
14     '1', 'UniCity万科天空之城', 'http://itcast-haoke.oss-cn-
15     qingdao.aliyuncs.com/images/2018/11/26/15432029097062227.jpg', '2018-11-26 11:28:49',
16     '2018-11-26 11:28:51');
17
18 --测试结果：主库有写入数据，从库会同步数据
    
```

4.5、数据分片

MySQL集群1：

主机	端口	容器名称	角色
192.168.1.18	3306	percona-master01	master
192.168.1.18	3307	percona-slave01	slave

MySQL集群2：

主机	端口	容器名称	角色
192.168.1.18	3316	percona-master02	master
192.168.1.18	3317	percona-slave02	slave

4.5.1、配置master

```

1  #搭建master
2  #创建目录
3  mkdir /data/mysql/master02
4  cd /data/mysql/master02
5  mkdir conf data
6  chmod 777 * -R
7
8  #创建配置文件
9  cd /data/mysql/master02/conf
10 vim my.cnf
11
12 #输入如下内容
13 [mysqld]
14 log-bin=mysql-bin #开启二进制日志
15 server-id=1 #服务id, 不可重复
16 sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO
17 ,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION'
    
```



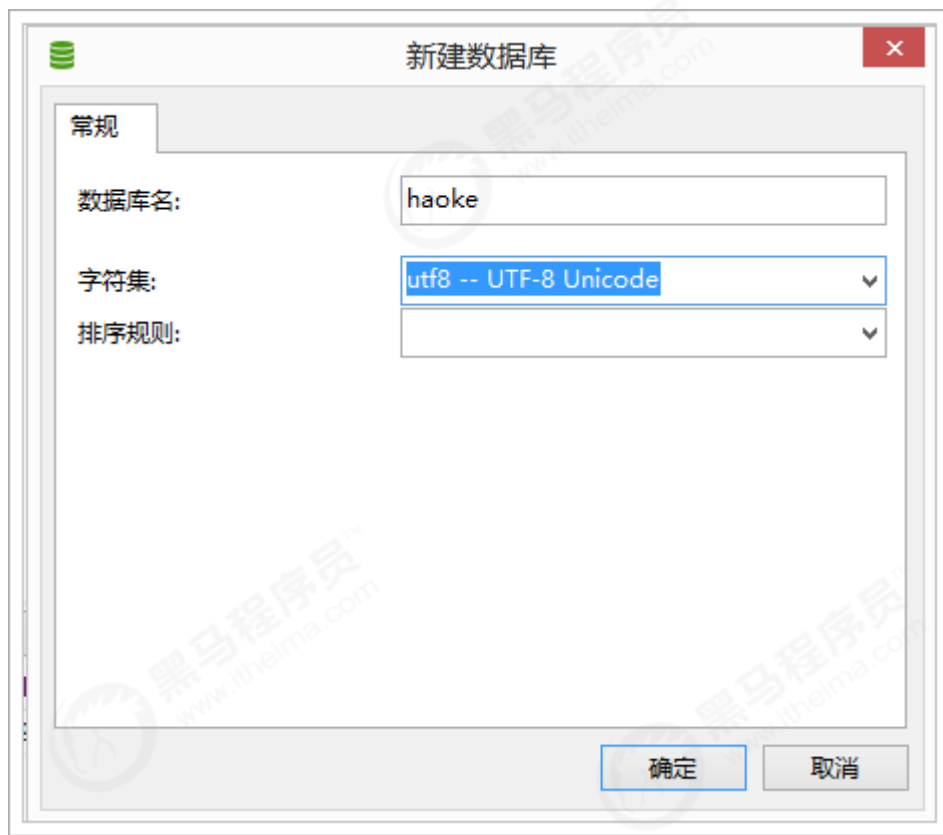
```
17
18 #创建容器
19 docker create --name percona-master02 -v /data/mysql/master02/data:/var/lib/mysql -v
   /data/mysql/master02/conf:/etc/my.cnf.d -p 3316:3306 -e MYSQL_ROOT_PASSWORD=root
   percona:5.7.23
20
21 #启动
22 docker start percona-master02 && docker logs -f percona-master02
23
24 #创建同步账户以及授权
25 create user 'itcast'@'%' identified by 'itcast';
26 grant replication slave on *.* to 'itcast'@'%';
27 flush privileges;
28
29 #查看master状态
30 show master status;
```

4.5.2、配置slave

```
1 #搭建从库
2 #创建目录
3 mkdir /data/mysql/slave02
4 cd /data/mysql/slave02
5 mkdir conf data
6 chmod 777 * -R
7
8 #创建配置文件
9 cd /data/mysql/slave02/conf
10 vim my.cnf
11
12 #输入如下内容
13 [mysqld]
14 server-id=2 #服务id, 不可重复
15 sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO
   ,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION'
16
17 #创建容器
18 docker create --name percona-slave02 -v /data/mysql/slave02/data:/var/lib/mysql -v
   /data/mysql/slave02/conf:/etc/my.cnf.d -p 3317:3306 -e MYSQL_ROOT_PASSWORD=root
   percona:5.7.23
19
20 #启动
21 docker start percona-slave02 && docker logs -f percona-slave02
22
23 #设置master相关信息
24 CHANGE MASTER TO
25     master_host='192.168.1.18',
26     master_user='itcast',
27     master_password='itcast',
28     master_port=3316,
29     master_log_file='xxxxxx',
30     master_log_pos=xxxx;
31
```

```
32 #启动同步
33 start slave;
34
35 #查看master状态
36 show slave status;
```

4.5.3、创建数据库以及表



```
1 CREATE TABLE `tb_ad` (
2   `id` bigint(20) NOT NULL AUTO_INCREMENT,
3   `type` int(10) DEFAULT NULL COMMENT '广告类型',
4   `title` varchar(100) DEFAULT NULL COMMENT '描述',
5   `url` varchar(200) DEFAULT NULL COMMENT '图片URL地址',
6   `created` datetime DEFAULT NULL,
7   `updated` datetime DEFAULT NULL,
8   PRIMARY KEY (`id`)
9 ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='广告表';
```

4.5.4、配置MyCat

schema.xml :

```
1 <?xml version="1.0"?>
2 <!DOCTYPE mycat:schema SYSTEM "schema.dtd">
3 <mycat:schema xmlns:mycat="http://io.mycat/">
4   <!--配置数据表-->
```



```
5 <schema name="itcast" checkSQLSchema="false" sqlMaxLimit="100">
6   <table name="tb_ad" dataNode="dn1,dn2" rule="mod-long" />
7 </schema>
8 <!--配置分片关系-->
9 <dataNode name="dn1" dataHost="cluster1" database="itcast" />
10 <dataNode name="dn2" dataHost="cluster2" database="itcast" />
11 <!--配置连接信息-->
12 <dataHost name="cluster1" maxCon="1000" minCon="10" balance="3"
13   writeType="1" dbType="mysql" dbDriver="native" switchType="1"
14   slaveThreshold="100">
15   <heartbeat>select user()</heartbeat>
16   <writeHost host="w1" url="192.168.1.18:3306" user="root"
17     password="root">
18     <readHost host="w1r1" url="192.168.1.18:3307" user="root"
19       password="root" />
20   </writeHost>
21 </dataHost>
22 <dataHost name="cluster2" maxCon="1000" minCon="10" balance="3"
23   writeType="1" dbType="mysql" dbDriver="native" switchType="1"
24   slaveThreshold="100">
25   <heartbeat>select user()</heartbeat>
26   <writeHost host="w2" url="192.168.1.18:3316" user="root"
27     password="root">
28     <readHost host="w2r1" url="192.168.1.18:3317" user="root"
29       password="root" />
30   </writeHost>
31 </dataHost>
32 </mycat:schema>
```

rule.xml :

```
1 <function name="mod-long" class="io.mycat.route.function.PartitionByMod">
2   <property name="count">2</property>
3 </function>
```

重新启动mycat进行测试：

```
1 | ./startup_nowrap.sh && tail -f ../logs/mycat.log
```

4.5.5、测试



```

1 INSERT INTO `tb_ad` (`id`, `type`, `title`, `url`, `created`, `updated`) VALUES ('1',
  '1', 'UniCity万科天空之城', 'http://itcast-haoke.oss-cn-
qingdao.aliyuncs.com/images/2018/11/26/15432029097062227.jpg', '2018-11-26 11:28:49',
'2018-11-26 11:28:51');
2
3 INSERT INTO `tb_ad` (`id`, `type`, `title`, `url`, `created`, `updated`) VALUES ('2',
  '1', '天和尚海庭前', 'http://itcast-haoke.oss-cn-
qingdao.aliyuncs.com/images/2018/11/26/1543202958579877.jpg', '2018-11-26 11:29:27',
'2018-11-26 11:29:29');
4
5 INSERT INTO `tb_ad` (`id`, `type`, `title`, `url`, `created`, `updated`) VALUES ('3',
  '1', '[奉贤 南桥] 光语著', 'http://itcast-haoke.oss-cn-
qingdao.aliyuncs.com/images/2018/11/26/15432029946721854.jpg', '2018-11-26 11:30:04',
'2018-11-26 11:30:06');
6
7 INSERT INTO `tb_ad` (`id`, `type`, `title`, `url`, `created`, `updated`) VALUES ('4',
  '1', '[上海周边 嘉兴] 融创海逸长洲', 'http://itcast-haoke.oss-cn-
qingdao.aliyuncs.com/images/2018/11/26/15432030275359146.jpg', '2018-11-26 11:30:49',
'2018-11-26 11:30:53');
8

```

结果：

对象 tb_ad @haoke (192.168.1.1...						
开始事务 备注 筛选 排序 导入 导出						
id	type	title	url	created	updated	
2	1	天和尚海	http://it	2018-11-26 11:29:27	2018-11-26 11:29:29	
4	1	[上海周边	http://it	2018-11-26 11:30:49	2018-11-26 11:30:53	

对象 tb_ad @haoke (192.168.1.1...)						
开始事务 备注 筛选 排序 导入 导出						
id	type	title	url	created	updated	
1	1	UniCity万科天空之城	http://itcast-haoke.oss-cn	2018-11-26 11:28:49	2018-11-26 11:28:51	
3	1	[奉贤 南桥] 光语著	http://itcast-haoke.oss-cn	2018-11-26 11:30:04	2018-11-26 11:30:06	

查看mycat：

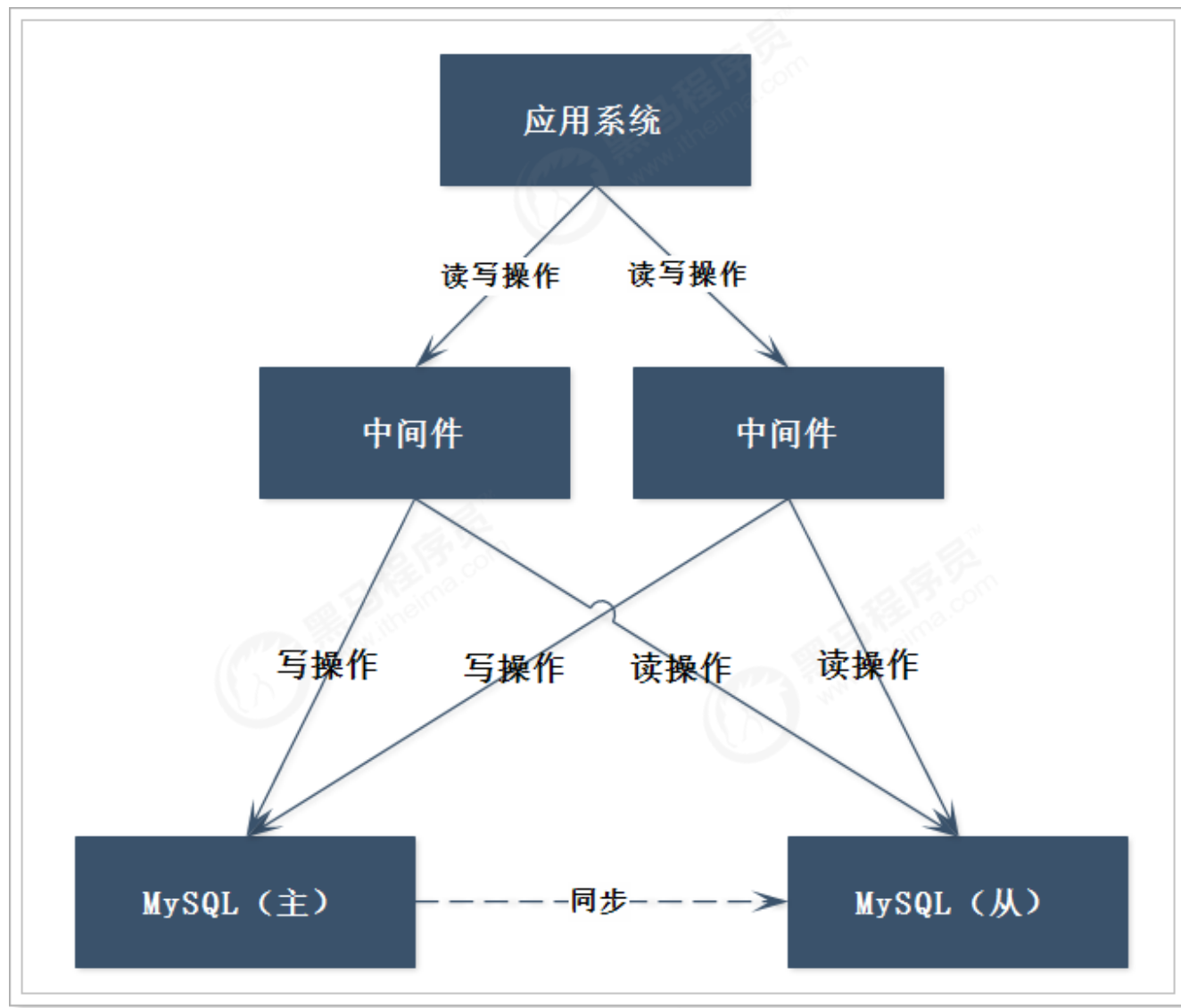
对象 tb_ad @itcast (192.168.1.18-...						
开始事务 备注 筛选 排序 导入 导出						
id	type	title	url	created	updated	
1	1	UniCity万科天空之城	http://itcast-haoke.oss-cn	2018-11-26 11:28:49	2018-11-26 11:28:51	
3	1	[奉贤 南桥] 光语著	http://itcast-haoke.oss-cn	2018-11-26 11:30:04	2018-11-26 11:30:06	
2	1	天和尚海庭前	http://itcast-haoke.oss-cn	2018-11-26 11:29:27	2018-11-26 11:29:29	
4	1	[上海周边 嘉兴] 融创海逸长	http://itcast-haoke.oss-cn	2018-11-26 11:30:49	2018-11-26 11:30:53	

可以看到，数据已经从2个分片中进行了汇总。

4.6、mycat集群

mycat做了数据库的代理，在高并发的情况下，必然也会面临单节点性能问题，所以需要部署多个mycat节点。

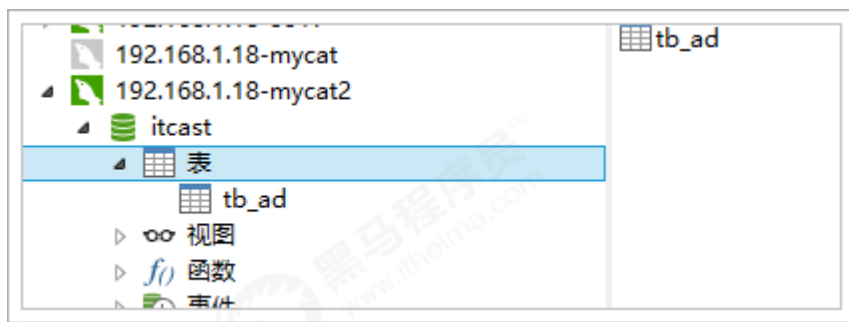
架构：



搭建多节点mycat：

```
1  cp mycat mycat2 -R
2
3  vim wrapper.conf
4  #设置jmx端口
5  wrapper.java.additional.7=-Dcom.sun.management.jmxremote.port=1985
6
7  vim server.xml
8  #设置服务端以及管理端口
9  <property name="serverPort">8067</property>
10 <property name="managerPort">9067</property>
11
12 #重新启动服务
```

```
13 ./startup_nowrap.sh
14 tail -f ../logs/mycat.log
```



多节点的mycat搭建完成。

5、负载均衡

在前面架构中，虽然对mycat做了集群，保障了mycat的可靠性，但是，应用程序需要连接到多个mycat，显然不是很友好的，也就是说缺少负载均衡的组件，接下来我们来了解下HAProxy。

5.1、简介

haproxy

编辑

讨论

HAProxy是一个使用C语言编写的自由及开放源代码软件[1]，其提供高可用性、负载均衡，以及基于TCP和HTTP的应用程序代理。

HAProxy特别适用于那些负载特大的web站点，这些站点通常又需要会话保持或七层处理。HAProxy运行在当前的硬件上，完全可以支持数以万计的并发连接。并且它的运行模式使得它可以很简单安全的整合进您当前的架构中，同时可以保护你的web服务器不被暴露到网络上。

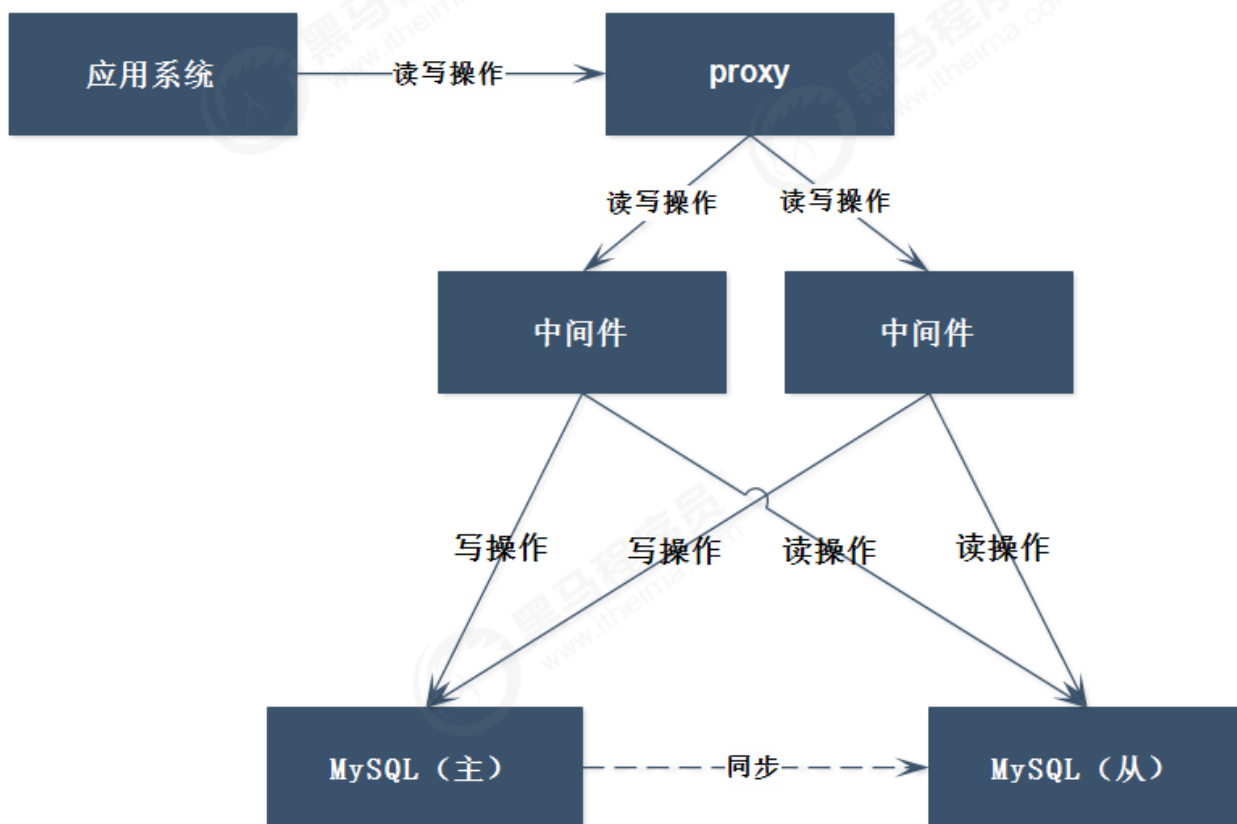
HAProxy实现了一种事件驱动、单一进程模型，此模型支持非常大的并发连接数。多进程或多线程模型受内存限制、系统调度器限制以及无处不在的锁限制，很少能处理数千并发连接。事件驱动模型因为在有更好的资源和时间管理的用户空间(User-Space)实现所有这些任务，所以没有这些问题。此模型的弊端是，在多核系统上，这些程序通常扩展性较差。这就是为什么他们必须进行优化以使每个CPU时间片(Cycle)做更多的工作。

包括 GitHub、Bitbucket[3]、Stack Overflow[4]、Reddit、Tumblr、Twitter[5][6]和 Tuenti[7]在内的知名网站，及亚马逊网络服务系统都使用了HAProxy。

官网：<http://www.haproxy.org/>

关于并发性能，haproxy可以做到千万级的并发。（当然了，运行环境不同，测试结果也不相同的）

5.2、架构



5.3、部署安装HAProxy



我们依然是通过docker进行安装。

```
1 #拉取镜像
2 docker pull haproxy:1.9.3
3
4 #创建目录，用于存放配置文件
5 mkdir /haoke/haproxy
6
7 #创建容器
8 docker create --name haproxy --net host -v /haoke/haproxy:/usr/local/etc/haproxy
   haproxy:1.9.3
```

编写配置文件：

```
1 #创建文件
2 vim /haoke/haproxy/haproxy.cfg
3
4 #输入如下内容
5 global
6     log          127.0.0.1 local2
7     maxconn      4000
8     daemon
9
10 defaults
11     mode          http
12     log           global
13     option        httplog
14     option        dontlognull
15     option http-server-close
16     option forwardfor except 127.0.0.0/8
17     option        redispatch
18     retries       3
19     timeout http-request 10s
20     timeout queue   1m
21     timeout connect 10s
22     timeout client  1m
23     timeout server  1m
24     timeout http-keep-alive 10s
25     timeout check   10s
26     maxconn        3000
27
28 listen admin_stats
29     bind      0.0.0.0:4001
30     mode      http
31     stats uri  /dbs
32     stats realm Global\ statistics
33     stats auth admin:admin123
34
35 listen proxy-mysql
36     bind      0.0.0.0:4002
37     mode      tcp
38     balance   roundrobin
```



```
39 option tcplog
40 #代理mycat服务
41 server mycat_1 192.168.1.18:8066 check port 8066 maxconn 2000
42 server mycat_2 192.168.1.18:8067 check port 8067 maxconn 2000
43
```

启动容器：

```
1 #启动容器
2 docker restart haproxy && docker logs -f haproxy
```

5.4、测试

通过web界面进行测试：<http://192.168.1.18:4001/dbs>

HAProxy version 1.9.3, released 2019/01/29

Statistics Report for pid 6

> General process information

pid = 6 (process #1, nbproc = 1, nbthread = 1)
uptime = 0d 11h17m10s
system limits: memmax = unlimited; ulimit-n = 8034
maxsock = 8034; maxconn = 4000; maxpipes = 0
current conns = 1; current pipes = 0/0; conn rate = 1/sec
Running tasks: 1/10; idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:

- Scope:
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:

- Primary site
- Updates (v1.9)
- Online manual

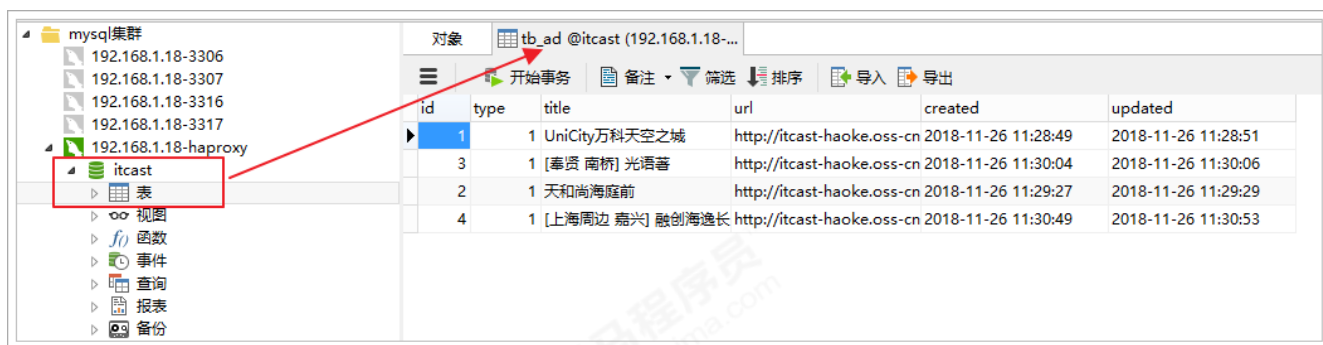
Note: NOC / DRAM = 0 if with no-banking disabled.

admin_stats		Queue		Session rate			Sessions				Bytes		Denied		Errors		Warnings		Server												
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend					1	2	-	1	2	3 000	8			4 245	68 071	0	0	2					OPEN								
Backend		0	0		0	1		0	1	300	4	0	0s	4 245	68 071	0	0		4	0		0	11h17m UP		0	0	0			0	

proxy-mysql		Queue		Session rate			Sessions				Bytes		Denied		Errors		Warnings		Server												
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend					0	1	-	0	1	3 000	1			410	6 628	0	0	0					OPEN								
mycat_1		0	0	-	0	1		0	1	2000	1	1	11h17m	410	6 628	0	0	0	0	0	0	0	11h17m UP	L4OK in 0ms	1	Y	-	0	0	0s	-
mycat_2		0	0	-	0	0		0	0	2000	0	0	?	0	0	0	0	0	0	0	0	0	11h17m UP	L4OK in 0ms	1	Y	-	0	0	0s	-
Backend		0	0		0	1		0	1	300	1	1	11h17m	410	6 628	0	0	0	0	0	0	0	11h17m UP		2	2	0		0	0s	

从页面中，可以看出已经存在了2个mycat代理服务。

通过mysql客户端进行测试：



6、PXC集群

6.1、简介

Percona XtraDB Cluster (简称PXC) 是针对MySQL用户的高可用性和扩展性解决方案，基于Percona Server。其包括了Write Set REplication补丁，使用Galera 2.0库，这是一个针对事务性应用程序的同步多主机复制插件。

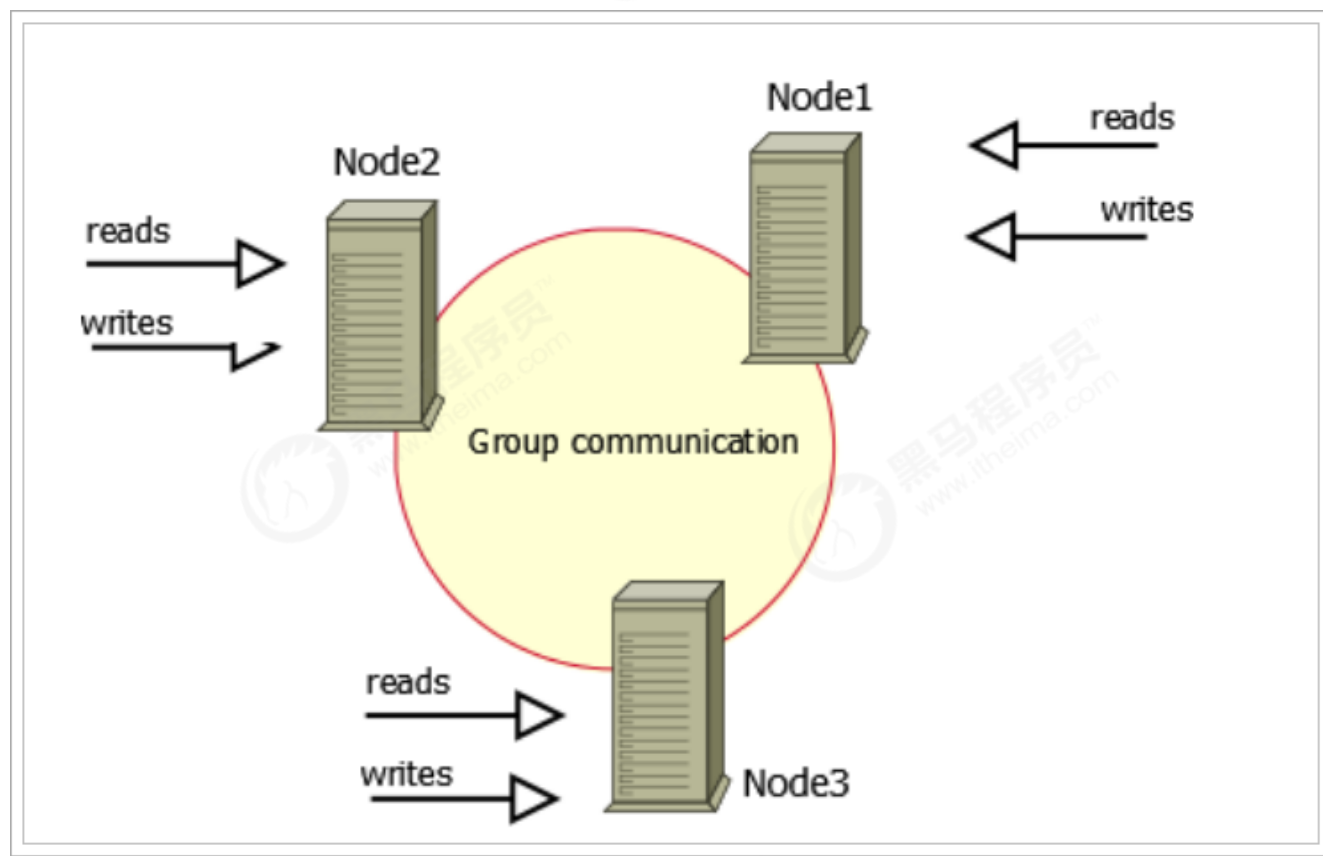
Percona Server是MySQL的改进版本，使用 XtraDB 存储引擎，在功能和性能上较 MySQL 有着很显著的提升，如提升了在高负载情况下的 InnoDB 的性能，为 DBA 提供了一些非常有用的性能诊断工具，另外有更多的参数和命令来控制服务器行为。

Percona XtraDB Cluster提供了：

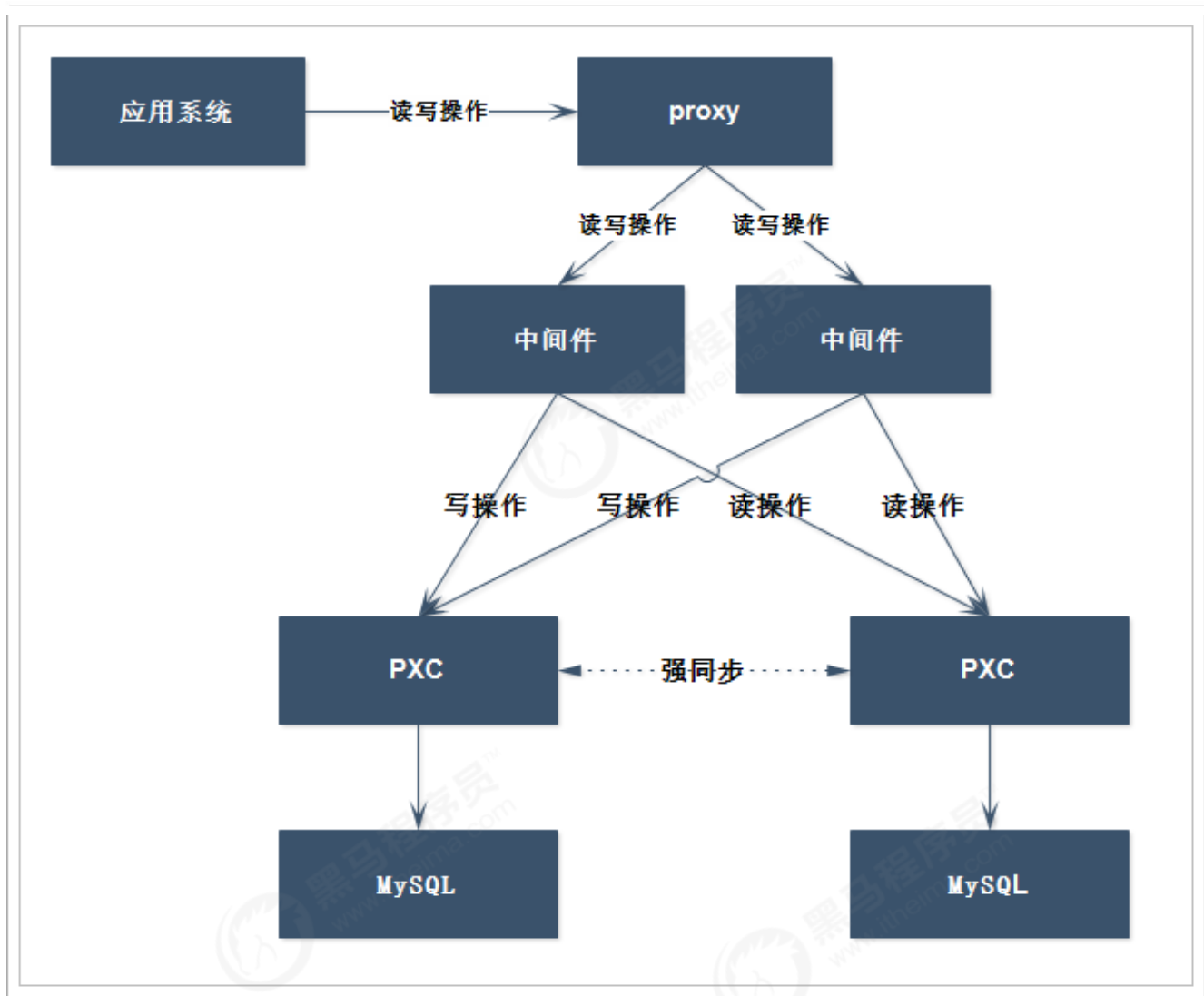
- 同步复制，事务可以在所有节点上提交。
- 多主机复制，你可以写到任何节点。
- 从（slave）服务器上的并行应用事件，真正的“并行复制”。
- 自动节点配置。
- 数据一致性，不再有未同步的从服务器。

官网：<https://www.percona.com/software/mysql-database/percona-xtradb-cluster>

图示：



6.2、架构



6.3、部署安装

接下来，我们部署安装三节点的PXC。

节点	端口	容器名称	数据卷
node1	13306	pxc_node1	v1
node2	13307	pxc_node2	v2
node3	13308	pxc_node3	v3

实施：

```
1 #创建数据卷（存储路径：/var/lib/docker/volumes）
2 docker volume create v1
3 docker volume create v2
4 docker volume create v3
5
6 #拉取镜像
7 docker pull percona/percona-xtradb-cluster:5.7
```



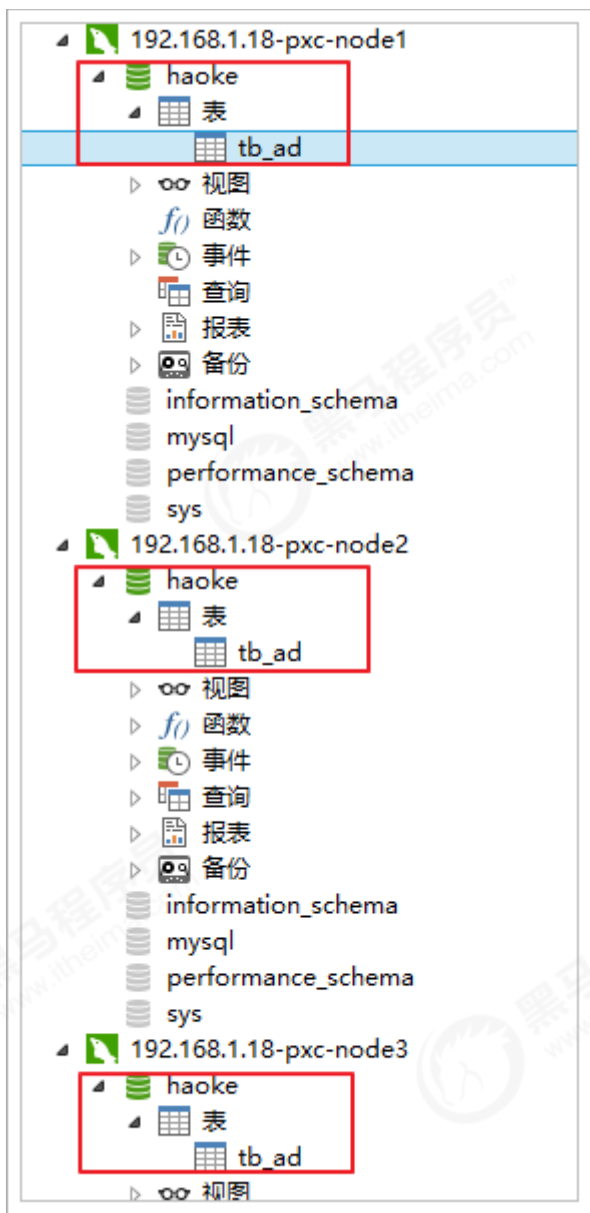

```
8 #重命名
9 docker tag percona/percona-xtradb-cluster:5.7 pxc
10
11 #创建网络
12 docker network create --subnet=172.30.0.0/24 pxc-network
13
14 #创建容器
15 #第一节点
16 docker create -p 13306:3306 -v v1:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e
  CLUSTER_NAME=pxc --name=pxc_node1 --net=pxc-network --ip=172.30.0.2 pxc
17
18 #第二节点 (增加了CLUSTER_JOIN参数)
19 docker create -p 13307:3306 -v v2:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e
  CLUSTER_NAME=pxc --name=pxc_node2 -e CLUSTER_JOIN=pxc_node1 --net=pxc-network --
  ip=172.30.0.3 pxc
20
21 #第三节点 (增加了CLUSTER_JOIN参数)
22 docker create -p 13308:3306 -v v3:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e
  CLUSTER_NAME=pxc --name=pxc_node3 -e CLUSTER_JOIN=pxc_node1 --net=pxc-network --
  ip=172.30.0.4 pxc
23
24 #查看集群节点
25 show status like 'wsrep_cluster%';
```

需要注意的是：先启动第一个节点，等到mysql客户端可以连接到服务后再启动其它节点。

6.4、测试

```
1 CREATE TABLE `tb_ad` (
2   `id` bigint(20) NOT NULL AUTO_INCREMENT,
3   `type` int(10) DEFAULT NULL COMMENT '广告类型',
4   `title` varchar(100) DEFAULT NULL COMMENT '描述',
5   `url` varchar(200) DEFAULT NULL COMMENT '图片URL地址',
6   `created` datetime DEFAULT NULL,
7   `updated` datetime DEFAULT NULL,
8   PRIMARY KEY (`id`)
9 ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='广告表';
10
11
12 --插入测试数据
13 INSERT INTO `tb_ad` (`id`, `type`, `title`, `url`, `created`, `updated`) VALUES ('1',
  '1', 'Unicity万科天空之城', 'http://itcast-haoke.oss-cn-
  qingdao.aliyuncs.com/images/2018/11/26/15432029097062227.jpg', '2018-11-26 11:28:49',
  '2018-11-26 11:28:51');
```

效果：



数据：

id	type	title	url	created	updated
1	1	UniCity万	http://it	2018-11-26	2018-11-26 1

6.5、集群的说明

- 尽可能的控制PXC集群的规模，节点越多，数据同步速度越慢
- 所有PXC节点的硬件配置要一致，如果不一致，配置低的节点将拖慢数据同步速度
- PXC集群只支持InnoDB引擎，不支持其他的存储引擎

6.6、PXC集群方案与Replication区别

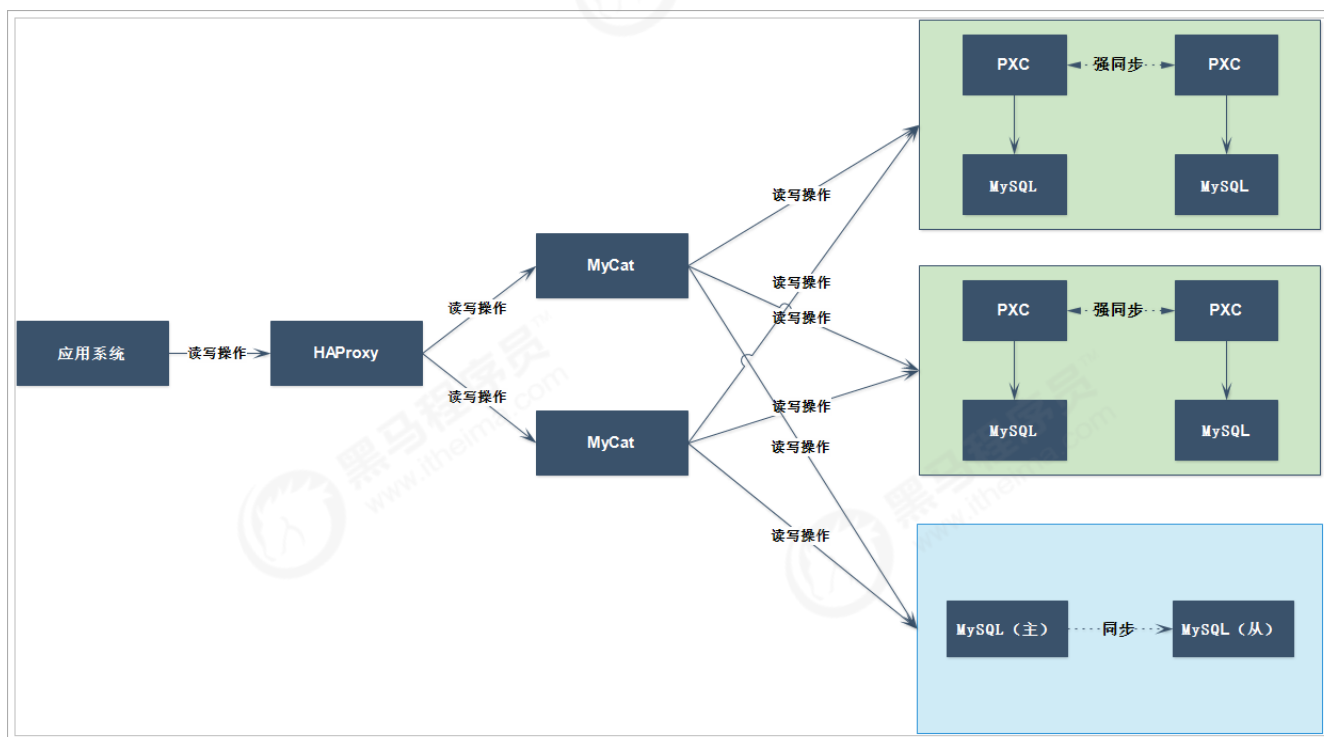
- PXC集群方案所有节点都是可读可写的，Replication从节点不能写入，因为主从同步是单向的，无法从slave节点向master点同步。

- PXC同步机制是同步进行的，这也是它能保证数据强一致性的根本原因，Replication同步机制是异步进行的，它如果从节点停止同步，依然可以向主节点插入数据，正确返回，造成数据主从数据的不一致性。
- PXC是用牺牲性能保证数据的一致性，Replication在性能上是高于PXC的。所以两者用途也不一致。PXC是用于重要信息的存储，例如：订单、用户信息等。Replication用于一般信息的存储，能够容忍数据丢失，例如：购物车，用户行为日志等。

7、综合应用

前面学习了主从架构、Mycat中间件、HAProxy负载均衡、PXC集群架构，在实际的项目中，往往不单单是一种架构，更多的使用的混合架构，下面我们将好客租房项目采用混合架构的方式进行完善数据库集群。

7.1、架构



说明：

- HAProxy作为负载均衡器
- 部署了2个Mycat节点作为数据库中间件
- 部署了2个PXC集群节点，作为2个Mycat分片，每个PXC集群中有2个节点，作为数据的同步存储
- 部署了1个主从复制集群
- 房源数据保存到PXC分片中，其余数据保存到主从架构中

7.2、部署PXC集群

集群一：

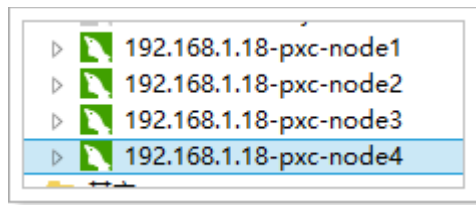
节点	端口	容器名称	数据卷
node1	13306	pxc_node1	haoke-v1
node2	13307	pxc_node2	haoke-v2

集群二：

节点	端口	容器名称	数据卷
node3	13308	pxc_node3	haoke-v3
node4	13309	pxc_node4	haoke-v4

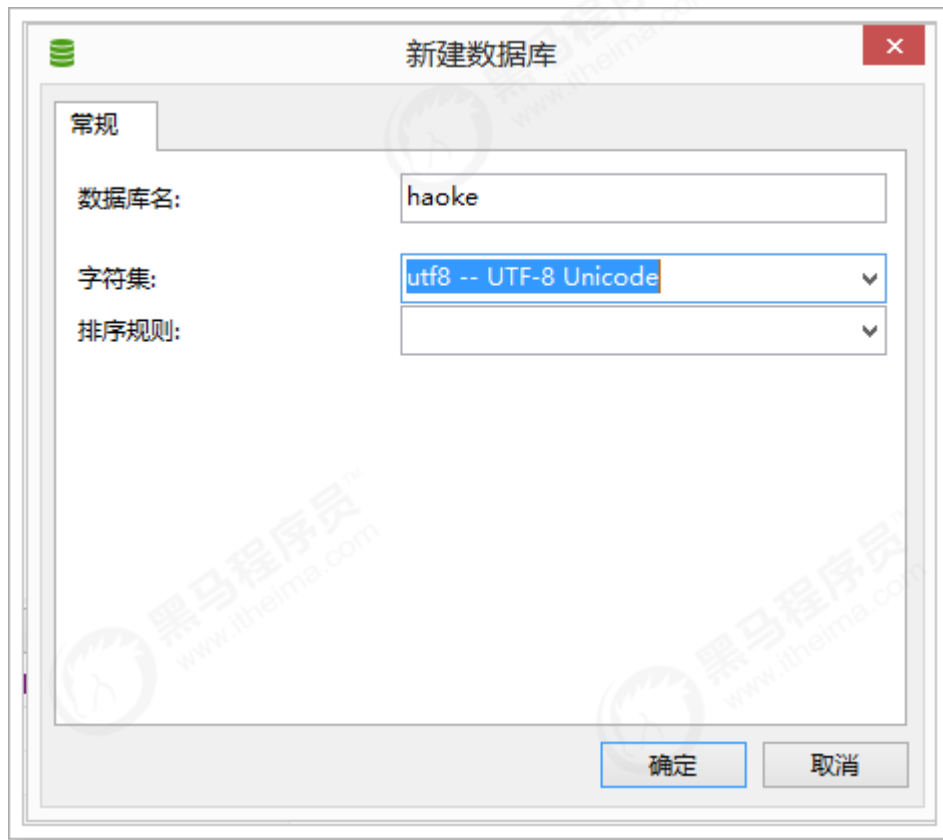
```
1 #创建数据卷 ( 存储路径 : /var/lib/docker/volumes )
2 docker volume create haoke-v1
3 docker volume create haoke-v2
4 docker volume create haoke-v3
5 docker volume create haoke-v4
6
7 #拉取镜像
8 docker pull percona/percona-xtradb-cluster:5.7
9
10 #创建网络
11 docker network create --subnet=172.30.0.0/24 pxc-network
12
13 #创建容器
14 #集群1，第一节点
15 docker create -p 13306:3306 -v haoke-v1:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e
    CLUSTER_NAME=pxc --name=pxc_node1 --net=pxc-network --ip=172.30.0.2 pxc
16
17 #第二节点 ( 增加了CLUSTER_JOIN参数 )
18 docker create -p 13307:3306 -v haoke-v2:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e
    CLUSTER_NAME=pxc --name=pxc_node2 -e CLUSTER_JOIN=pxc_node1 --net=pxc-network --
    ip=172.30.0.3 pxc
19
20 #集群2
21 #第一节点
22 docker create -p 13308:3306 -v haoke-v3:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e
    CLUSTER_NAME=pxc --name=pxc_node3 --net=pxc-network --ip=172.30.0.4 pxc
23
24 #第二节点 ( 增加了CLUSTER_JOIN参数 )
25 docker create -p 13309:3306 -v haoke-v4:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -e
    CLUSTER_NAME=pxc --name=pxc_node4 -e CLUSTER_JOIN=pxc_node3 --net=pxc-network --
    ip=172.30.0.5 pxc
26
27 #启动
28 docker start pxc_node1 && docker logs -f pxc_node1
29 docker start pxc_node2 && docker logs -f pxc_node2
30
31 docker start pxc_node3 && docker logs -f pxc_node3
32 docker start pxc_node4 && docker logs -f pxc_node4
33
34 #查看集群节点
35 show status like 'wsrep_cluster%';
```

结果：



2个集群，4个节点，均启动成功。

分别在2个集群中创建haoke数据库以及房源数据表：



```
1 CREATE TABLE `tb_house_resources` (  
2   `id` bigint(20) NOT NULL AUTO_INCREMENT,  
3   `title` varchar(100) DEFAULT NULL COMMENT '房源标题',  
4   `estate_id` bigint(20) DEFAULT NULL COMMENT '楼盘id',  
5   `building_num` varchar(5) DEFAULT NULL COMMENT '楼号(栋)',  
6   `building_unit` varchar(5) DEFAULT NULL COMMENT '单元号',  
7   `building_floor_num` varchar(5) DEFAULT NULL COMMENT '门牌号',  
8   `rent` int(10) DEFAULT NULL COMMENT '租金',  
9   `rent_method` tinyint(1) DEFAULT NULL COMMENT '租赁方式, 1-整租, 2-合租',  
10  `payment_method` tinyint(1) DEFAULT NULL COMMENT '支付方式, 1-付一押一, 2-付三押一, 3-付  
    六押一, 4-年付押一, 5-其它',  
11  `house_type` varchar(255) DEFAULT NULL COMMENT '户型, 如: 2室1厅1卫',  
12  `covered_area` varchar(10) DEFAULT NULL COMMENT '建筑面积',  
13  `use_area` varchar(10) DEFAULT NULL COMMENT '使用面积',  
14  `floor` varchar(10) DEFAULT NULL COMMENT '楼层, 如: 8/26',  
15  `orientation` varchar(2) DEFAULT NULL COMMENT '朝向: 东、南、西、北',  
16  `decoration` tinyint(1) DEFAULT NULL COMMENT '装修, 1-精装, 2-简装, 3-毛坯',  
17  `facilities` varchar(50) DEFAULT NULL COMMENT '配套设施, 如: 1,2,3',  
18  `pic` varchar(1000) DEFAULT NULL COMMENT '图片, 最多5张',  
)
```



```
19 `house_desc` varchar(200) DEFAULT NULL COMMENT '描述',
20 `contact` varchar(10) DEFAULT NULL COMMENT '联系人',
21 `mobile` varchar(11) DEFAULT NULL COMMENT '手机号',
22 `time` tinyint(1) DEFAULT NULL COMMENT '看房时间, 1-上午, 2-中午, 3-下午, 4-晚上, 5-全天',
23 `property_cost` varchar(10) DEFAULT NULL COMMENT '物业费',
24 `created` datetime DEFAULT NULL,
25 `updated` datetime DEFAULT NULL,
26 PRIMARY KEY (`id`)
27 ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='房源表';
28
```

7.3、部署主从复制集群

master :

```
1 #创建目录
2 mkdir /data/mysql/haoke/master01 -p
3 cd /data/mysql/haoke/master01
4 mkdir conf data
5 chmod 777 * -R
6
7 #创建配置文件
8 cd /data/mysql/haoke/master01/conf
9 vim my.cnf
10
11 #输入如下内容
12 [mysqld]
13 log-bin=mysql-bin #开启二进制日志
14 server-id=1 #服务id, 不可重复
15 sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO
16 ,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION'
17
18 #创建容器
19 docker create --name percona-haoke-master01 -v
20 /data/mysql/haoke/master01/data:/var/lib/mysql -v
21 /data/mysql/haoke/master01/conf:/etc/my.cnf.d -p 23306:3306 -e
22 MYSQL_ROOT_PASSWORD=root percona:5.7.23
23
24 #启动
25 docker start percona-haoke-master01 && docker logs -f percona-haoke-master01
26
27 #创建同步账户以及授权
28 create user 'itcast'@'%' identified by 'itcast';
29 grant replication slave on *.* to 'itcast'@'%';
30 flush privileges;
31
32 #查看master状态
33 show master status;
```

slave :



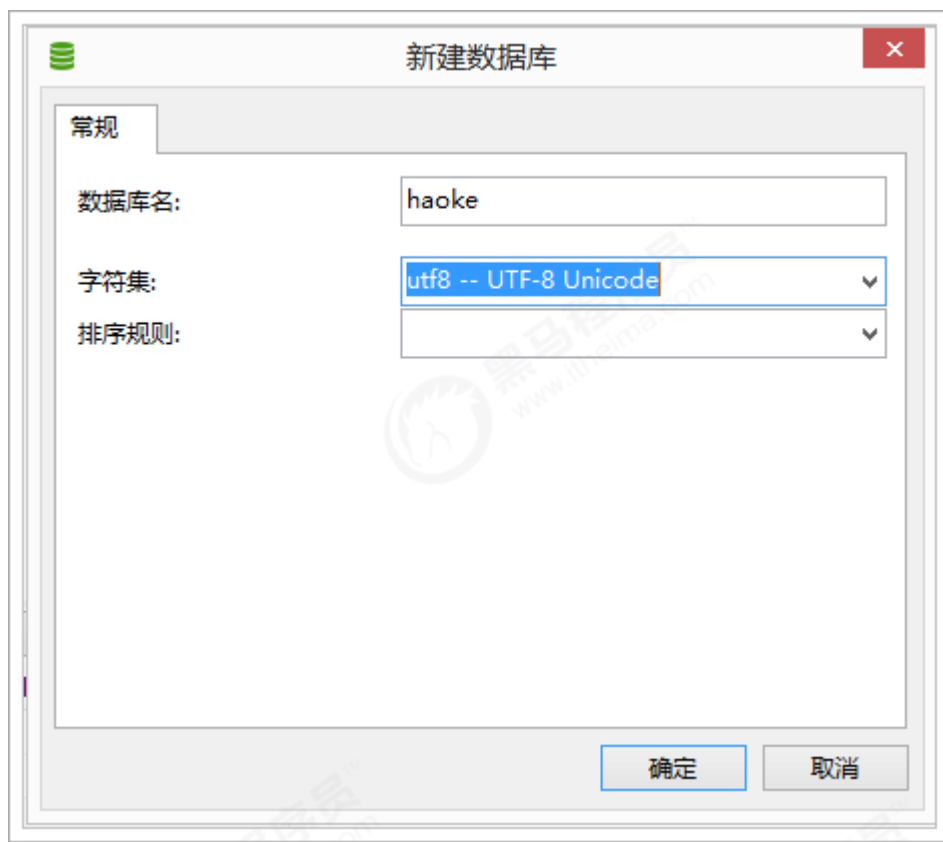
```
1 #创建目录
2 mkdir /data/mysql/haoke/slave01 -p
3 cd /data/mysql/haoke/slave01
4 mkdir conf data
5 chmod 777 * -R
6
7 #创建配置文件
8 cd /data/mysql/haoke/slave01/conf
9 vim my.cnf
10
11 #输入如下内容
12 [mysqld]
13 server-id=2 #服务id, 不可重复
14 sql_mode='STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO
15 ,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION'
16
17 #创建容器
18 docker create --name percona-haoke-slave01 -v
19 /data/mysql/haoke/slave01/data:/var/lib/mysql -v
20 /data/mysql/haoke/slave01/conf:/etc/my.cnf.d -p 23307:3306 -e
21 MYSQL_ROOT_PASSWORD=root percona:5.7.23
22
23 #启动
24 docker start percona-haoke-slave01 && docker logs -f percona-haoke-slave01
25
26 #设置master相关信息
27 CHANGE MASTER TO
28 master_host='192.168.1.18',
29 master_user='itcast',
30 master_password='itcast',
31 master_port=23306,
32 master_log_file='mysql-bin.000002',
33 master_log_pos=648;
34
35 #启动同步
36 start slave;
37
38 #查看master状态
39 show slave status;
```

测试：

Relay_Master_Log_File	Slave_IO_Running	Slave_SQL_Running	Replica
mysql-bin.000003	Yes	Yes	

说明主从搭建成功。

创建haoke数据库以及创建广告表：



```
1 CREATE TABLE `tb_ad` (  
2   `id` bigint(20) NOT NULL AUTO_INCREMENT,  
3   `type` int(10) DEFAULT NULL COMMENT '广告类型',  
4   `title` varchar(100) DEFAULT NULL COMMENT '描述',  
5   `url` varchar(200) DEFAULT NULL COMMENT '图片URL地址',  
6   `created` datetime DEFAULT NULL,  
7   `updated` datetime DEFAULT NULL,  
8   PRIMARY KEY (`id`)  
9 ) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8 COMMENT='广告表';
```

7.4、部署Mycat

7.4.1、节点一

```
1 cd /data/  
2 mkdir mycat  
3 cp /haoke/mycat . -R  
4 mv mycat/ mycat-node1
```

server.xml：

```
1 <?xml version="1.0" encoding="UTF-8"?>  
2 <!DOCTYPE mycat:server SYSTEM "server.dtd">  
3 <mycat:server xmlns:mycat="http://io.mycat/">  
4   <system>
```




```
5      <property name="nonePasswordLogin">0</property>
6      <property name="useHandshakeV10">1</property>
7      <property name="useSqlStat">0</property>
8      <property name="useGlobleTableCheck">0</property>
9      <property name="seunceHandlerType">2</property>
10     <property name="subqueryRelationshipCheck">>false</property>
11     <property name="processorBufferPoolType">0</property>
12     <property name="handleDistributedTransactions">0</property>
13     <property name="useOffHeapForMerge">1</property>
14     <property name="memoryPageSize">64k</property>
15     <property name="spillsFileBufferSize">1k</property>
16     <property name="useStreamOutput">0</property>
17     <property name="systemReserveMemorySize">384m</property>
18     <property name="useZKSwitch">>false</property>
19 </system>
20 <!--这里是设置的itcast用户和虚拟逻辑库-->
21 <user name="itcast" defaultAccount="true">
22     <property name="password">itcast123</property>
23     <property name="schemas">haoke</property>
24 </user>
25 </mycat:server>
```

schema.xml :

```
1 <?xml version="1.0"?>
2 <!DOCTYPE mycat:schema SYSTEM "schema.dtd">
3 <mycat:schema xmlns:mycat="http://io.mycat/">
4     <!--配置数据表-->
5     <schema name="haoke" checkSQLschema="false" sqlMaxLimit="100">
6         <table name="tb_house_resources" dataNode="dn1,dn2" rule="mod-long" />
7         <table name="tb_ad" dataNode="dn3"/>
8     </schema>
9     <!--配置分片关系-->
10    <dataNode name="dn1" dataHost="cluster1" database="haoke" />
11    <dataNode name="dn2" dataHost="cluster2" database="haoke" />
12    <dataNode name="dn3" dataHost="cluster3" database="haoke" />
13    <!--配置连接信息-->
14    <dataHost name="cluster1" maxCon="1000" minCon="10" balance="2"
15        writeType="1" dbType="mysql" dbDriver="native" switchType="1"
16        slaveThreshold="100">
17        <heartbeat>select user()</heartbeat>
18        <writeHost host="w1" url="192.168.1.18:13306" user="root"
19            password="root">
20            <readHost host="w1r1" url="192.168.1.18:13307" user="root"
21                password="root" />
22        </writeHost>
23    </dataHost>
24    <dataHost name="cluster2" maxCon="1000" minCon="10" balance="2"
25        writeType="1" dbType="mysql" dbDriver="native" switchType="1"
26        slaveThreshold="100">
27        <heartbeat>select user()</heartbeat>
28        <writeHost host="w2" url="192.168.1.18:13308" user="root"
29            password="root">
```



```
30         <readHost host="w2R1" url="192.168.1.18:13309" user="root"
31             password="root" />
32     </writeHost>
33 </dataHost>
34 <dataHost name="cluster3" maxCon="1000" minCon="10" balance="3"
35     writeType="1" dbType="mysql" dbDriver="native" switchType="1"
36     slaveThreshold="100">
37     <heartbeat>select user()</heartbeat>
38     <writeHost host="w3" url="192.168.1.18:23306" user="root"
39         password="root">
40         <readHost host="w3R1" url="192.168.1.18:23307" user="root"
41             password="root" />
42     </writeHost>
43 </dataHost>
44 </mycat:schema>
```

rule.xml :

```
1 <function name="mod-long" class="io.mycat.route.function.PartitionByMod">
2     <property name="count">2</property>
3 </function>
```

设置端口以及启动：

```
1 vim wrapper.conf
2 #设置jmx端口
3 wrapper.java.additional.7=-Dcom.sun.management.jmxremote.port=11985
4
5 vim server.xml
6 #设置服务端口以及管理端口
7 <property name="serverPort">18067</property>
8 <property name="managerPort">19067</property>
9
10 ./startup_nowrap.sh && tail -f ../logs/mycat.log
```

测试：



测试插入数据：

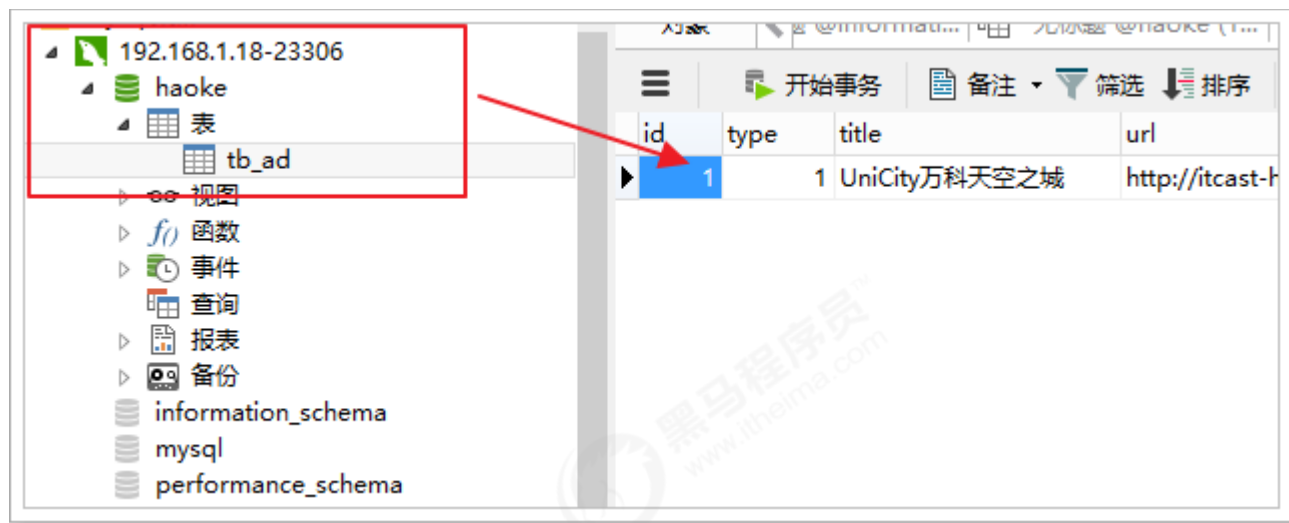


```
1 INSERT INTO `tb_house_resources` (`id`, `title`, `estate_id`, `building_num`,  
  `building_unit`, `building_floor_num`, `rent`, `rent_method`, `payment_method`,  
  `house_type`, `covered_area`, `use_area`, `floor`, `orientation`, `decoration`,  
  `facilities`, `pic`, `house_desc`, `contact`, `mobile`, `time`, `property_cost`,  
  `created`, `updated`) VALUES ('1', '东方曼哈顿 3室2厅 16000元', '1005', '2', '1', '1',  
  '1111', '1', '1', '1室1厅1卫1厨1阳台', '2', '2', '1/2', '南', '1', '1,2,3,8,9', NULL, '这个  
  经纪人很懒，没写核心卖点', '张三', '1111111111', '1', '11', '2018-11-16 01:16:00',  
  '2018-11-16 01:16:00');  
2 INSERT INTO `tb_house_resources` (`id`, `title`, `estate_id`, `building_num`,  
  `building_unit`, `building_floor_num`, `rent`, `rent_method`, `payment_method`,  
  `house_type`, `covered_area`, `use_area`, `floor`, `orientation`, `decoration`,  
  `facilities`, `pic`, `house_desc`, `contact`, `mobile`, `time`, `property_cost`,  
  `created`, `updated`) VALUES ('2', '康城 3室2厅1卫', '1002', '1', '2', '3', '2000', '1',  
  '2', '3室2厅1卫1厨2阳台', '100', '80', '2/20', '南', '1', '1,2,3,7,6', NULL, '拎包入住',  
  '张三', '1888888888', '5', '1.5', '2018-11-16 01:34:02', '2018-11-16 01:34:02');  
3  
4 INSERT INTO `tb_ad` (`id`, `type`, `title`, `url`, `created`, `updated`) VALUES ('1',  
  '1', 'Unicity万科天空之城', 'http://itcast-haoke.oss-cn-  
  qingdao.aliyuncs.com/images/2018/11/26/15432029097062227.jpg', '2018-11-26 11:28:49',  
  '2018-11-26 11:28:51');  
5
```

测试结果：

id	title	estate_id	building_num	building_unit	building_floor_num
2	康城 3室2厅1卫	1002	1	2	3

id	title	estate_id	building_num	building_unit	building_floor_num
1	东方曼哈顿 3室2厅 16000元	1005	2	1	1



7.4.2、节点二

```
1 cp mycat-node1/ mycat-node2 -R
2
3 vim wrapper.conf
4 #设置jmx端口
5 wrapper.java.additional.7=-Dcom.sun.management.jmxremote.port=11986
6
7 vim server.xml
8 #设置服务端口以及管理端口
9 <property name="serverPort">18068</property>
10 <property name="managerPort">19068</property>
11
12 ./startup_nowrap.sh && tail -f ../logs/mycat.log
```

测试结果与节点一相同。

7.5、部署HAProxy

修改配置文件：

```
1 #修改文件
2 vim /haoke/haproxy/haproxy.cfg
3
4 #输入如下内容
5 global
6     log          127.0.0.1 local2
7     maxconn      4000
8     daemon
9
10 defaults
11     mode          http
12     log           global
13     option        httplog
14     option        dontlognull
15     option http-server-close
16     option forwardfor except 127.0.0.0/8
```



```
17 option redispatch
18 retries 3
19 timeout http-request 10s
20 timeout queue 1m
21 timeout connect 10s
22 timeout client 1m
23 timeout server 1m
24 timeout http-keep-alive 10s
25 timeout check 10s
26 maxconn 3000
27
28 listen admin_stats
29 bind 0.0.0.0:4001
30 mode http
31 stats uri /dbs
32 stats realm Global\statistics
33 stats auth admin:admin123
34
35 listen proxy-mysql
36 bind 0.0.0.0:4002
37 mode tcp
38 balance roundrobin
39 option tcplog
40 #代理mycat服务
41 server mycat_1 192.168.1.18:18067 check port 18067 maxconn 2000
42 server mycat_2 192.168.1.18:18068 check port 18068 maxconn 2000
43
```

启动容器：

```
1 #启动容器
2 docker start haproxy && docker logs -f haproxy
```

<http://192.168.1.18:4001/dbs>

HAProxy version 1.9.3, released 2019/01/29

Statistics Report for pid 7

> General process information

pid = 7 (process #1, nbproc = 1, nbthread = 1)
uptime = 0d 0h00m26s
system limits: memmax = unlimited, ulimit-n = 8034
maxsock = 8034; maxconn = 4000; maxpipes = 0
current conns = 4; current pipes = 0/0; conn rate = 2/sec
Running tasks: 1/13; idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:

- Scope:
- Hide "DOWN" servers
- Refresh now
- CSV export

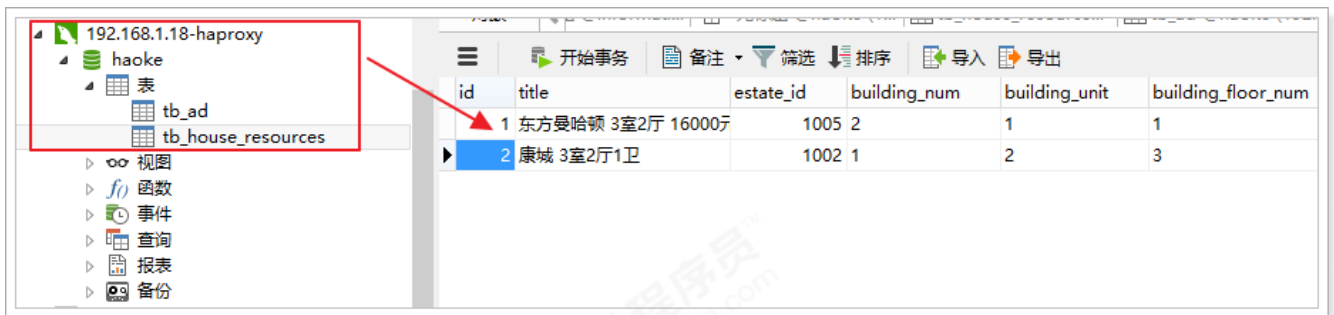
External resources:

- Primary site
- Updates (v1.9)
- Online manual

Note: NOC / DRAIN = 0: With no-connection issued.

admin_stats		Queue			Session rate			Sessions				Bytes		Denied		Errors			Warnings		Server											
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtime	Thrtle	
Frontend					2	2	-	2	2	3 000	4			942	16 756	0	0	1					OPEN									
Backend		0	0		0	1		0	1	300	1	0	0s	942	16 756	0	0	0	1	0	0	0	26s UP		0	0	0		0			

proxy-mysql		Queue			Session rate			Sessions				Bytes		Denied		Errors			Warnings		Server											
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Downtime	Thrtle	
Frontend					0	1	-	2	3	3 000	5			583	10 287	0	0	0					OPEN									
mycat_1		0	0	-	0	1		1	2	2000	3	3	12s	355	6 562	0	0	0	0	0	0	0	26s UP	L4OK in 0ms	1	Y	-	0	0	0s	-	
mycat_2		0	0	-	0	1		1	2	2000	2	2	15s	228	3 725	0	0	0	0	0	0	0	26s UP	L4OK in 0ms	1	Y	-	0	0	0s	-	
Backend		0	0		0	1		2	3	300	5	5	12s	583	10 287	0	0	0	0	0	0	0	26s UP		2	2	0		0			



至此，集群搭建完毕。