Getting Started with DBProxy

搜狐技术部 – DBA平台组

概述

- 面临的问题
- DBProxy介绍
- DBProxy实践
- 下一步计划

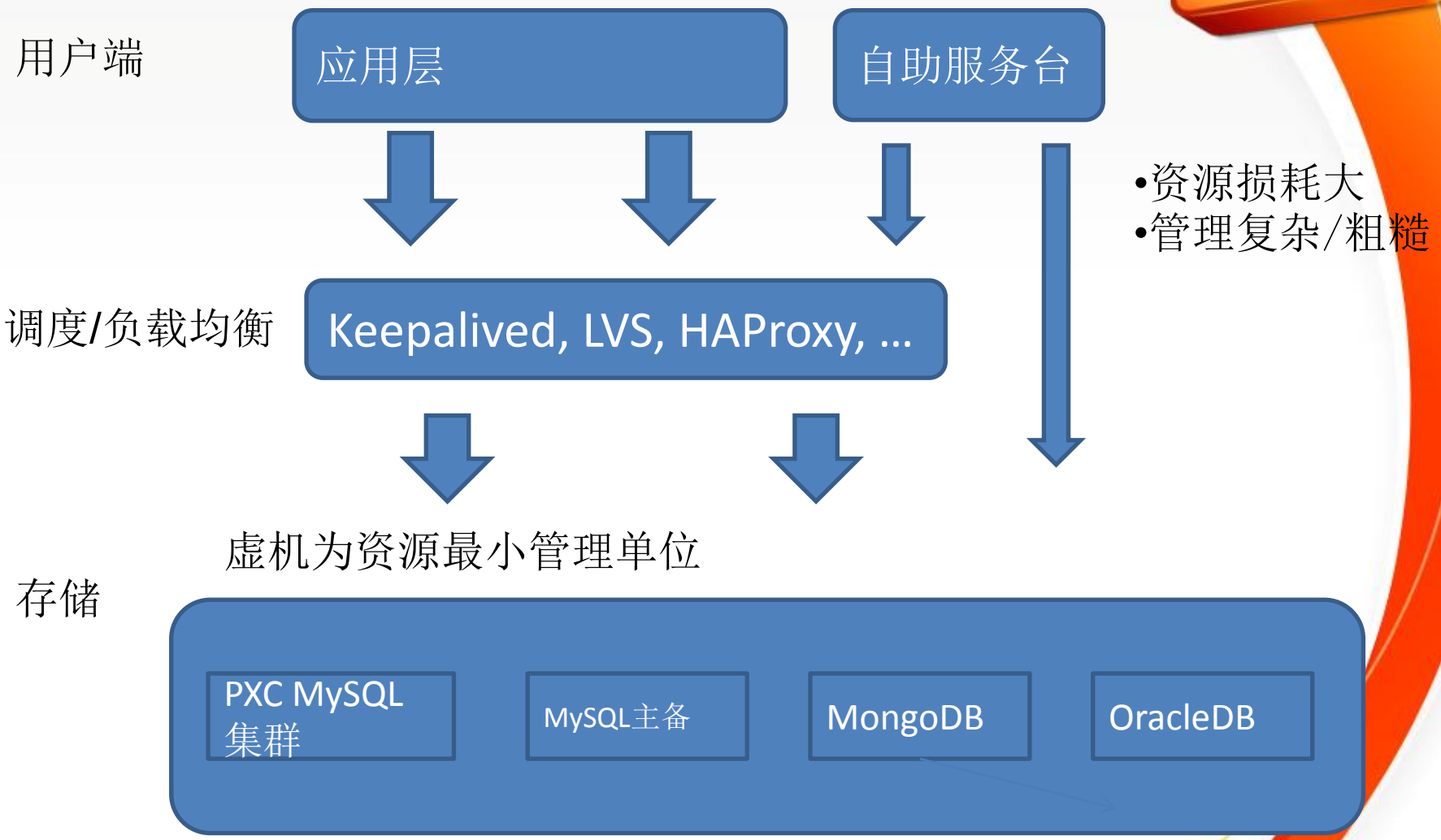
我们面临的问题

- 客户端机器太多，数据库连接不够啊
- 后面的MySQL太多，有主库从库等，而且老变，能否简单/透明点呢
- 某些SQL使用资源太狠了，怎么办呢？
- 数据快速增长，是否能做横向扩展呢
- 成本压力？
- .....

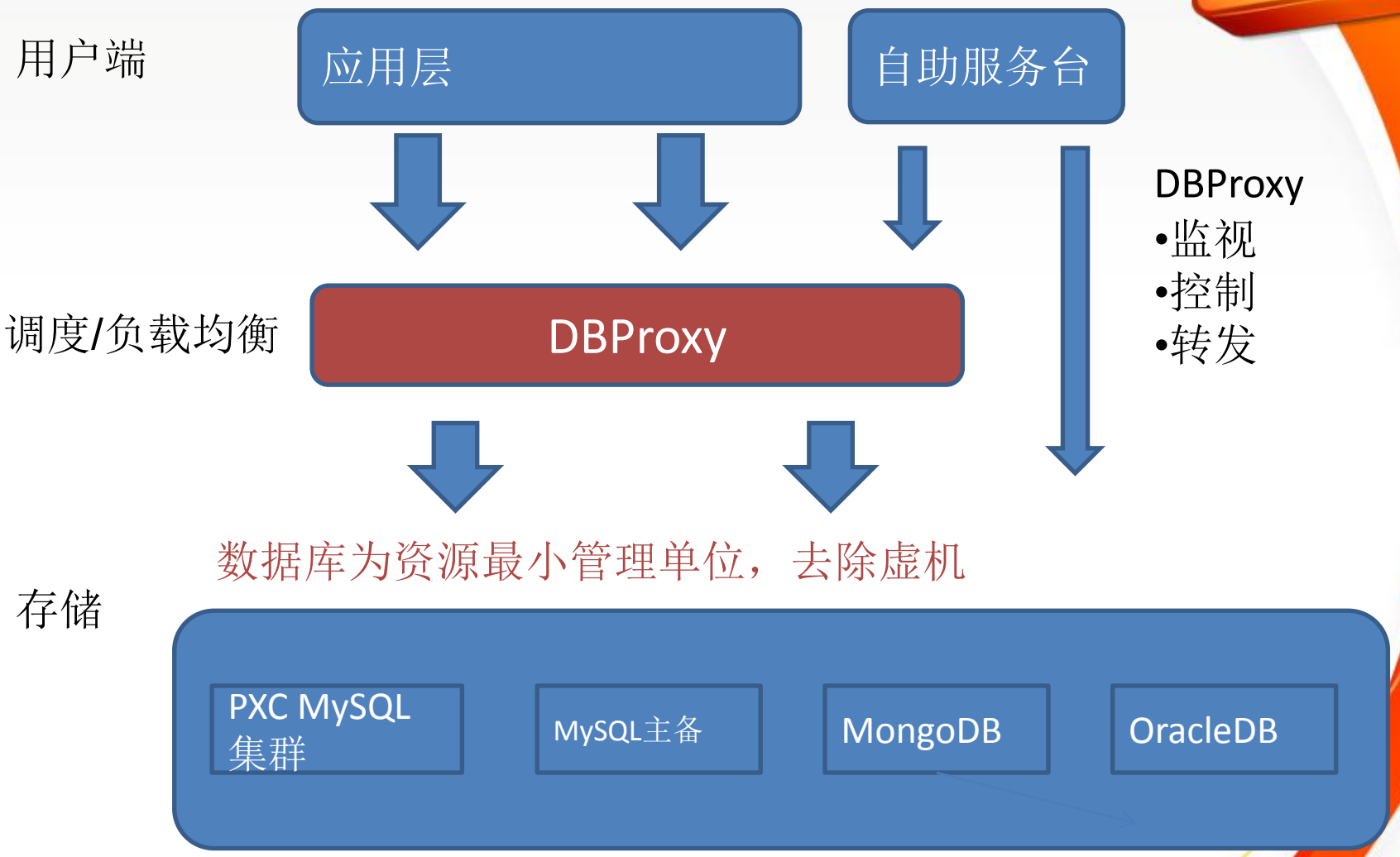
概述

- 面临的问题
- DBProxy介绍
- DBProxy实践
- 下一步计划

原来数据库云平台框架


用户端
应用层
自助服务台
•资源损耗大
•管理复杂/粗糙
调度/负载均衡
Keepalived, LVS, HAProxy, …
虚机为资源最小管理单位
存储
PXC MySQL 集群
MySQL主备
MongoDB
OracleDB

新数据库云平台框架


用户端
应用层
自助服务台
DBProxy
•监视
•控制
•转发
调度/负载均衡
DBProxy
数据库为资源最小管理单位，去除虚机
存储
PXC MySQL 集群
MySQL主备
MongoDB
OracleDB

介绍

- DBProxy是介于MySQL客户端和多台服务器之间，可以监视/控制/转发客户端的请求以及服务器返回的结果
- 参考了开源产品
  - MySQL Proxy 0.8.3
  - Spock Proxy
- 使用标准C
- 应用透明，减少运维代价

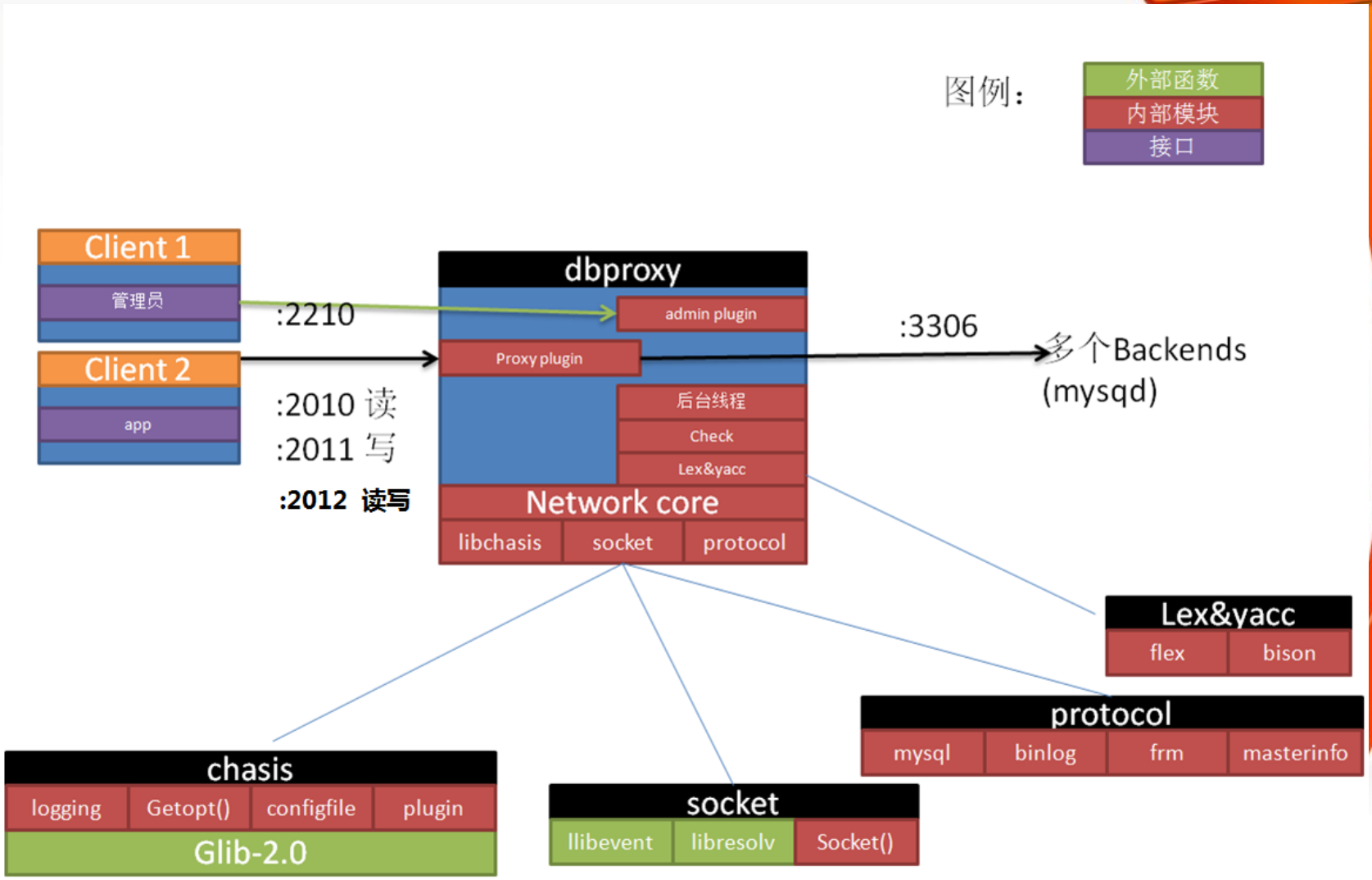
对应用透明

- 兼容MySQL协议，可以用任何5.1/5.5客户端连接
- 支持JDBC, PHP, ODBC, C 驱动， c3p0, DBCP连接池
  - JDBC, PHP,C驱动业务线已经在用或测试过
- 连接池提高PHP短连接性能
- 负载均衡提高读性能,支持动态扩展
- 自动实现读写分离

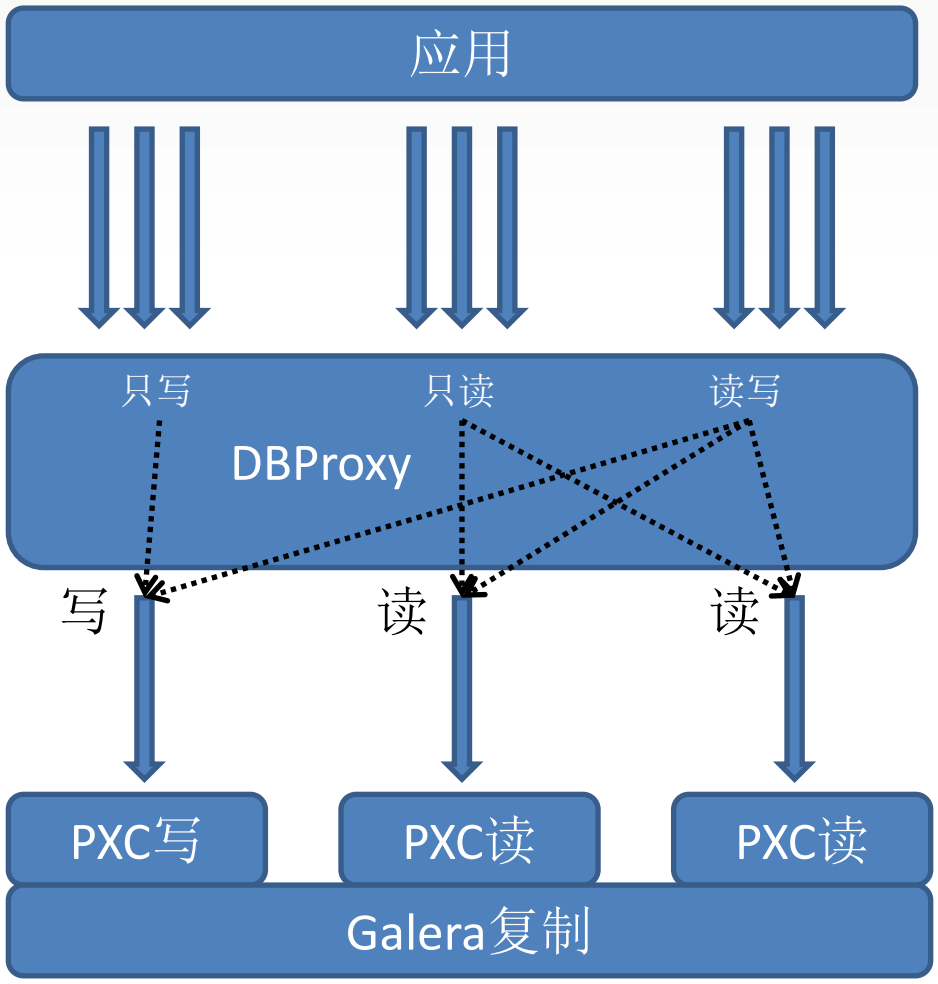
减少运维代价

- 现有高可用方案可简单的切换到DBProxy
- 对外提供一个读写分离的服务端口
- 数据库读负载均衡、故障转移自动化管理
- 连接池和连接复用减小了数据库资源使用
- 提供资源管理，对SQL语句过滤、限制
- 提供统计信息，SQL语句响应时间、等待事件

内部架构

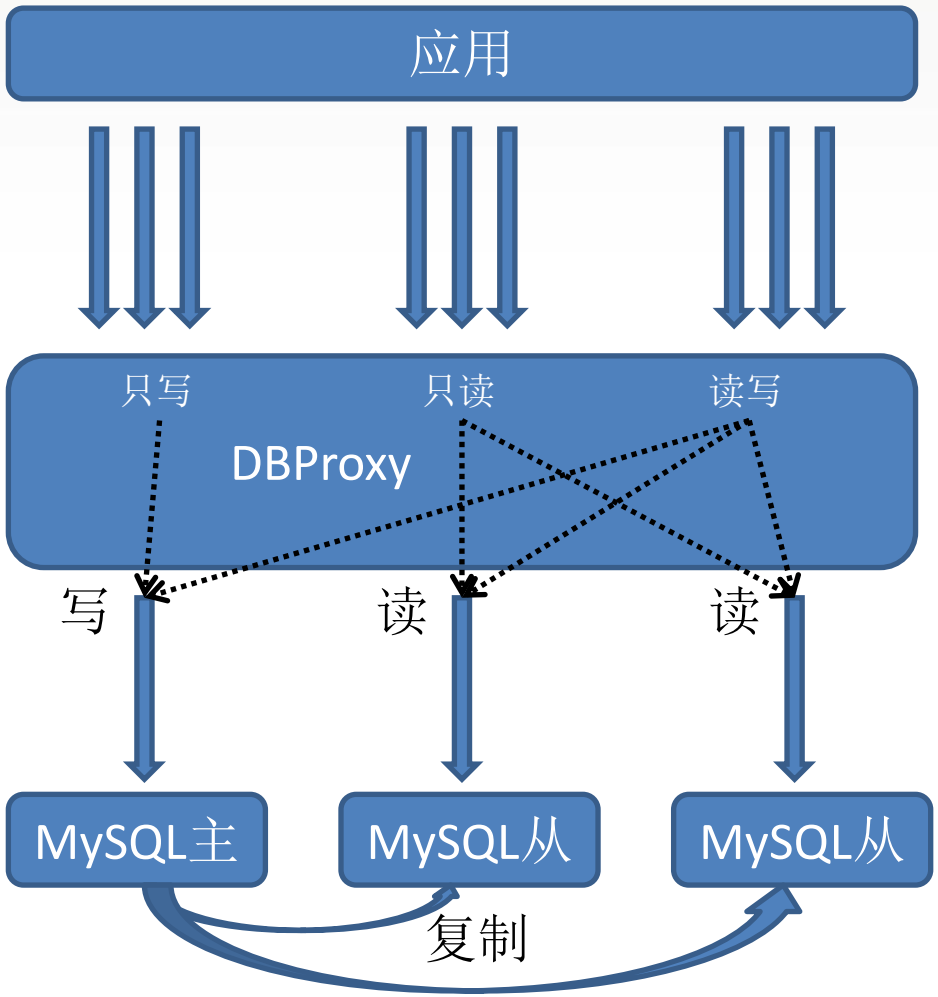

图例：
外部函数
内部模块
接口
Client 1
管理员
Client 2
app
:2210
:2010 读
:2011 写
:2012 读写
dbproxy
admin plugin
Proxy plugin
后台线程
Check
Lex&yacc
Network core
libchasis
socket
protocol
:3306
多个Backends
(mysqd)
Lex&yacc
flex
bison
protocol
mysql
binlog
frm
masterinfo
chasis
logging
Getopt()
configfile
plugin
Glib-2.0
socket
llibevent
libresolv
Socket()

整体架构



- 连接复用
- 负载均衡
- 故障转移
- 读写分离
- 目前支持PXC集群数据库

未来支持MySQL复制



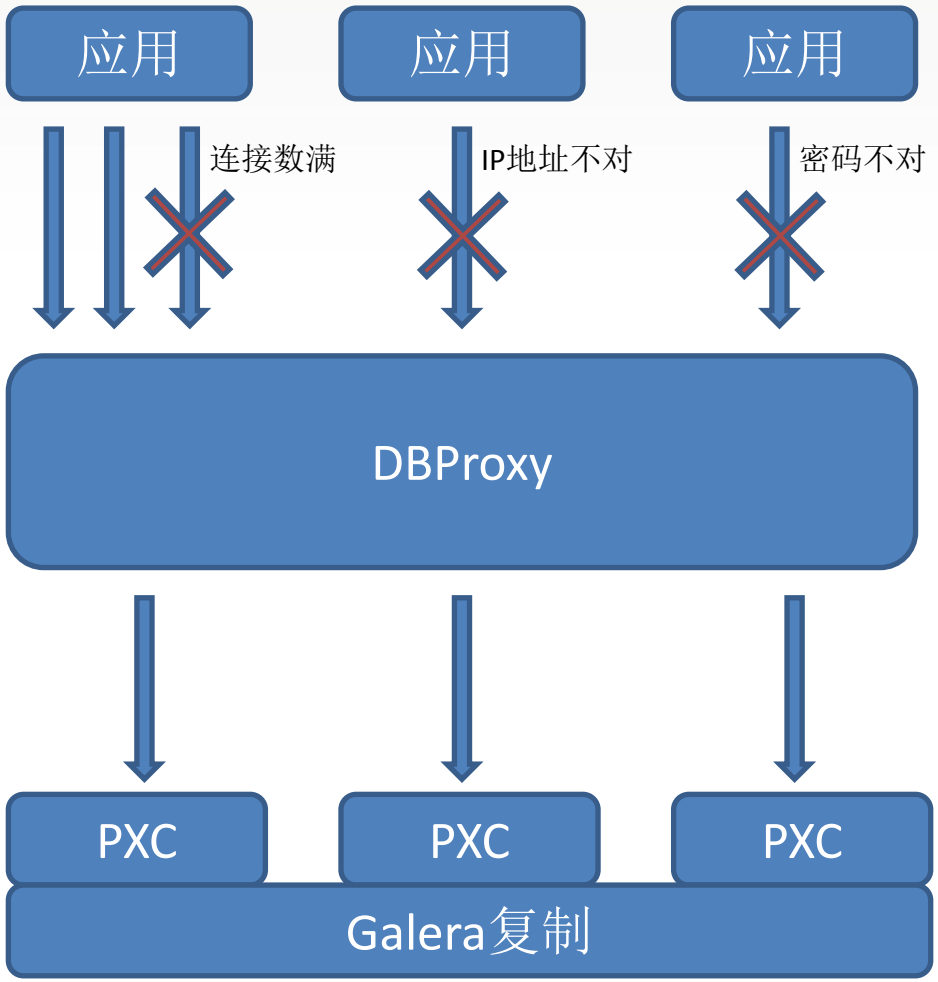
功能介绍

- 协议透明
- 前端连接限制、用户管理
- SQL监视和阻断
- 负载均衡、读写分离、故障转移
- 后端连接池和连接复用

协议透明

- 协议透明
  - 支持MySQL 5.1 / 5.5
- 应用无需知道具体后端数据库所在

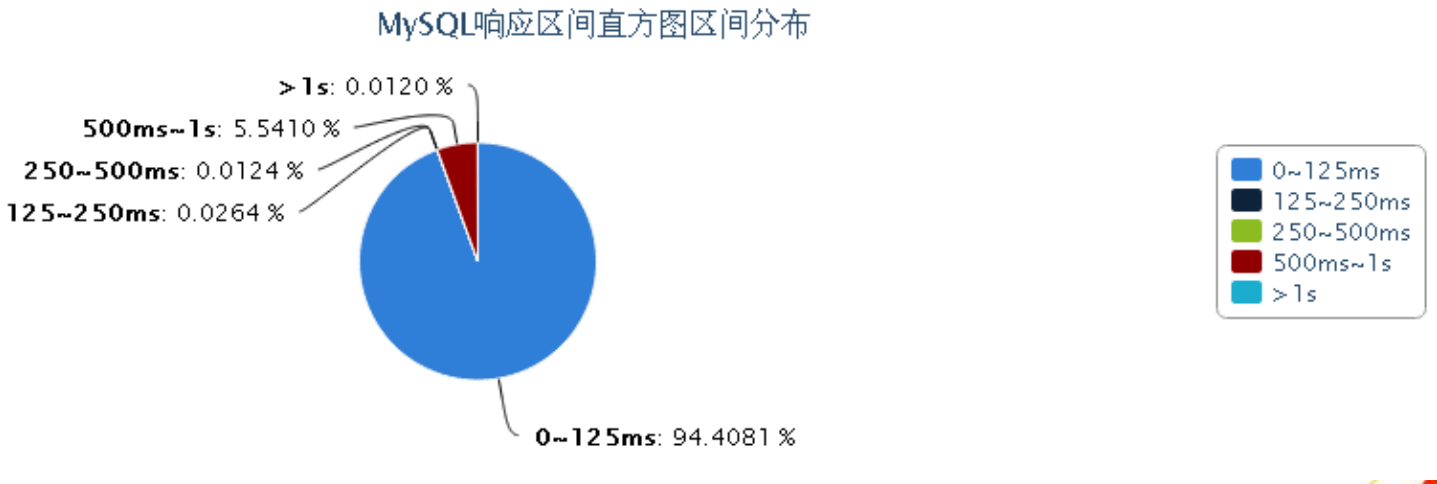
前端连接限制/用户管理



- 限制一个账号的并发连接数

SQL监视-背景

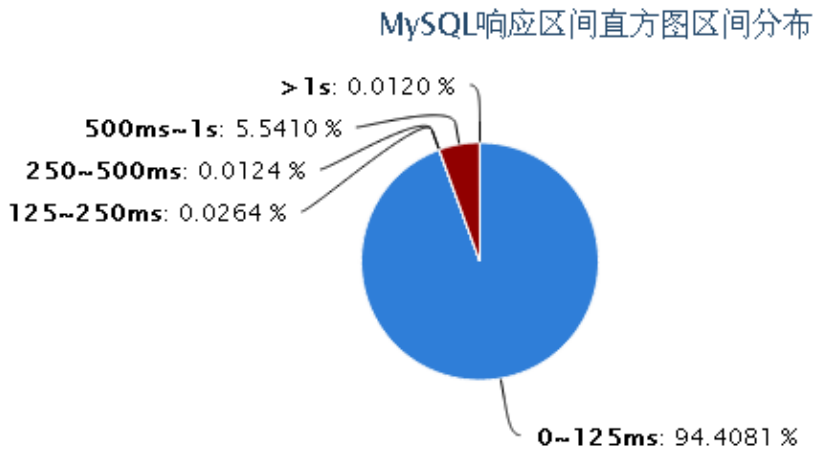
- MySQL中80%的问题是SQL导致的！
- MySQL本身统计数据不够细，收集影响大
  - SlowLog
  - TCPDump

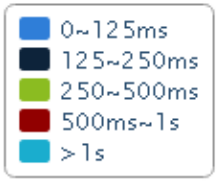


SQL监视

```
mysql> ShowQueryResponseTime;
+-------------------+--------+-------+--------------------------------+-------+----------+
| Time              | User   | Db    | Sql                            | Count | Total    |
+-------------------+--------+-------+--------------------------------+-------+----------+
| 0.001000~0.010000 | proxy1 | test  | select * from test where a=?   | 2     | 0.003513 |
| 0.010000~0.100000 | proxy1 | test  | select * from test where a=?   | 1     | 0.013403 |
| 0.001000~0.010000 | proxy1 | test  | select ?                       | 4     | 0.016922 |
| 0.001000~0.010000 | proxy  | test2 | select @@version_comment limit ? | 1   | 0.005449 |
| 0.001000~0.010000 | proxy  | test2 | select ?                       | 3     | 0.009984 |
| 0.001000~0.010000 | proxy  | test1 | select @@version_comment limit ? | 2   | 0.015030 |
| 0.001000~0.010000 | proxy  | test1 | select ?                       | 5     | 0.019605 |
| 0.001000~0.010000 | proxy  | test  | select @@version_comment limit ? | 2   | 0.010099 |
| 0.001000~0.010000 | proxy  | test  | select ?                       | 6     | 0.027194 |
| 0.001000~0.010000 | proxy1 | test2 | select ?                       | 1     | 0.003582 |
| 0.001000~0.010000 | proxy1 | test1 | select @@version_comment limit ? | 4   | 0.020319 |
| 0.001000~0.010000 | proxy1 | test1 | select ?                       | 2     | 0.008974 |
| 0.010000~0.100000 | proxy1 | test  | select @@version_comment limit ? | 7   | 0.351421 |
| 0.010000~0.100000 | proxy1 | test  | select ?                       | 5     | 0.312646 |
| 0.010000~0.100000 | proxy  | test2 | select @@version_comment limit ? | 6   | 0.169532 |
| 0.010000~0.100000 | proxy  | test2 | select ?                       | 7     | 0.238122 |
| 0.010000~0.100000 | proxy  | test1 | select @@version_comment limit ? | 5   | 0.189049 |
| 0.010000~0.100000 | proxy  | test1 | select ?                       | 4     | 0.168559 |
| 0.010000~0.100000 | proxy  | test  | select @@version_comment limit ? | 6   | 0.279458 |
| 0.010000~0.100000 | proxy  | test  | select ?                       | 4     | 0.144074 |
| 0.010000~0.100000 | proxy1 | test2 | select @@version_comment limit ? | 6   | 0.175733 |
```

- SQL语句响应时间的统计直方图
  - 细化到用户、数据库、标准化语句

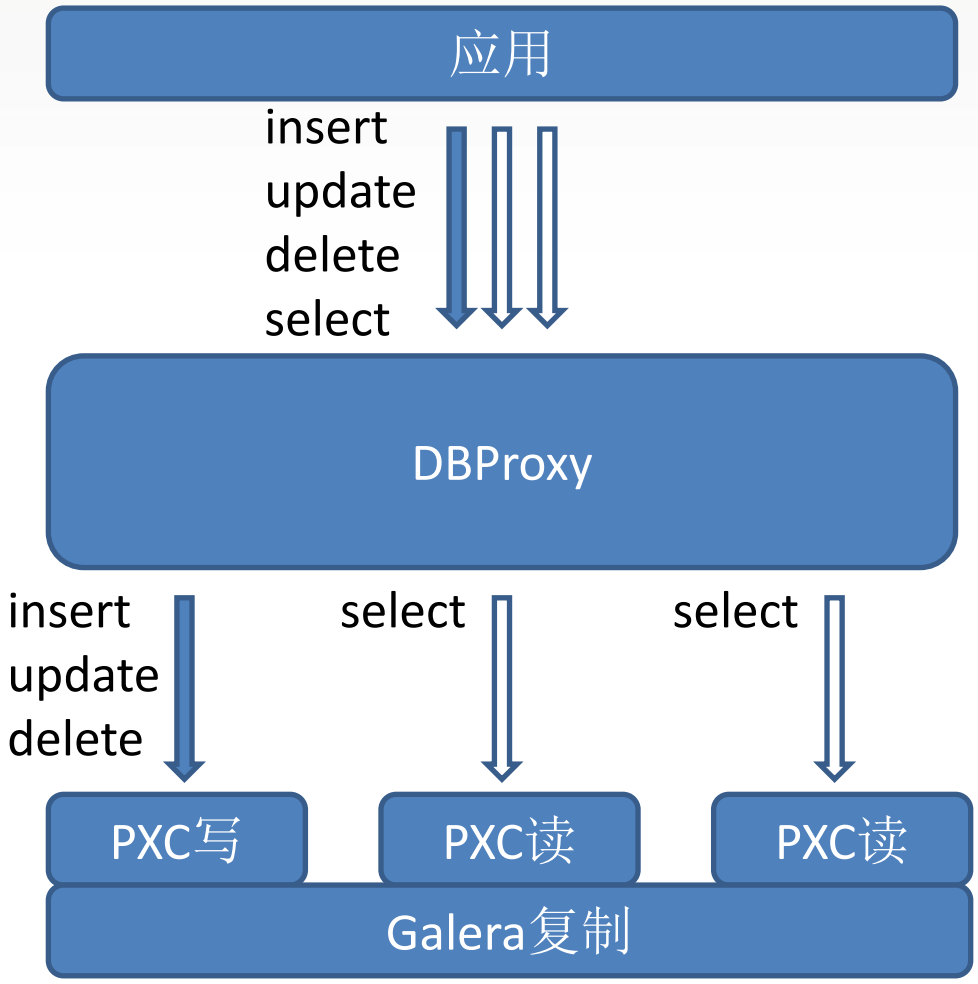




SQL阻断

- 阻断某条SQL
  - select * from t1 where id=1
- 阻断某类SQL
  - select * from t1 where id=?
  - 下列SQL会阻断
    - select * from t1 where id=1
    - select * from t1 where id=2
- 高级SQL阻断
  - 执行时长
  - 并行执行数量

读写分离

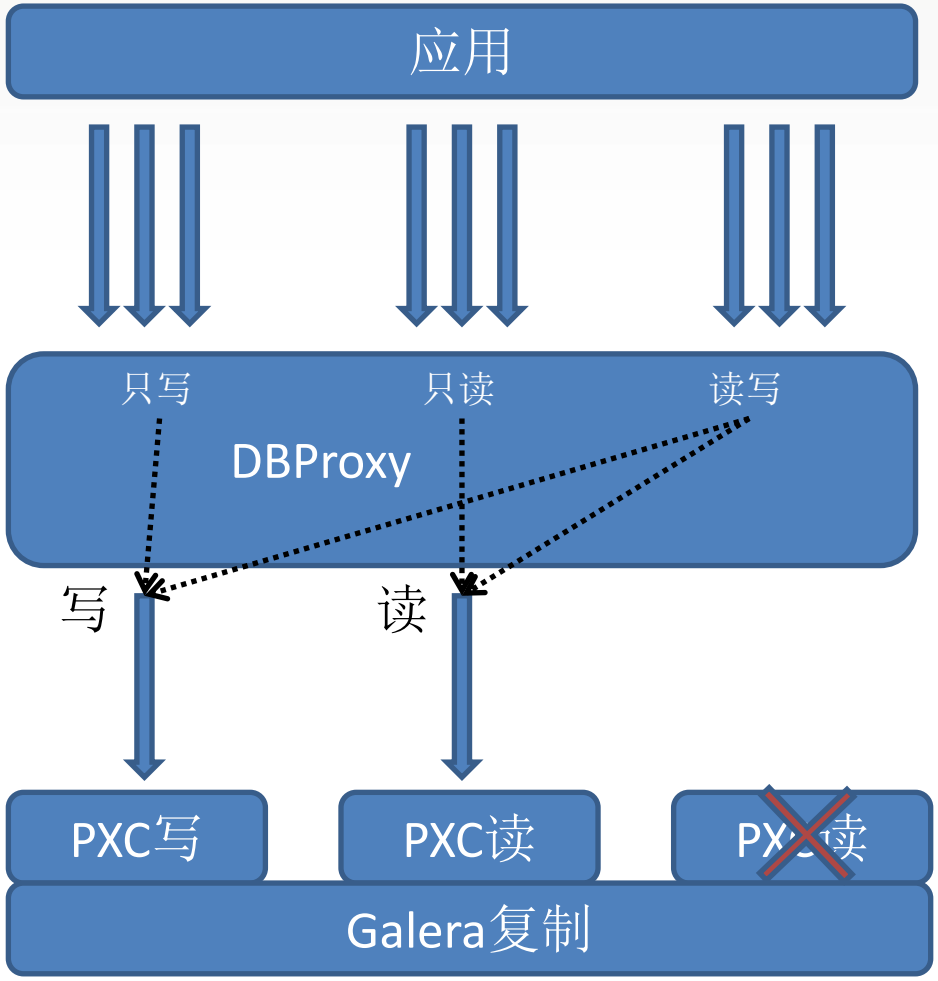


- 读操作路由到读后端
  - 查询语句：select, show, set, desc, use
  - COM_INIT_DB
  - COM_FIELD_LIST
- 写操作、事务操作全部路由到写后端上
  - 事务内
  - prepare
  - 其它语句

负载均衡

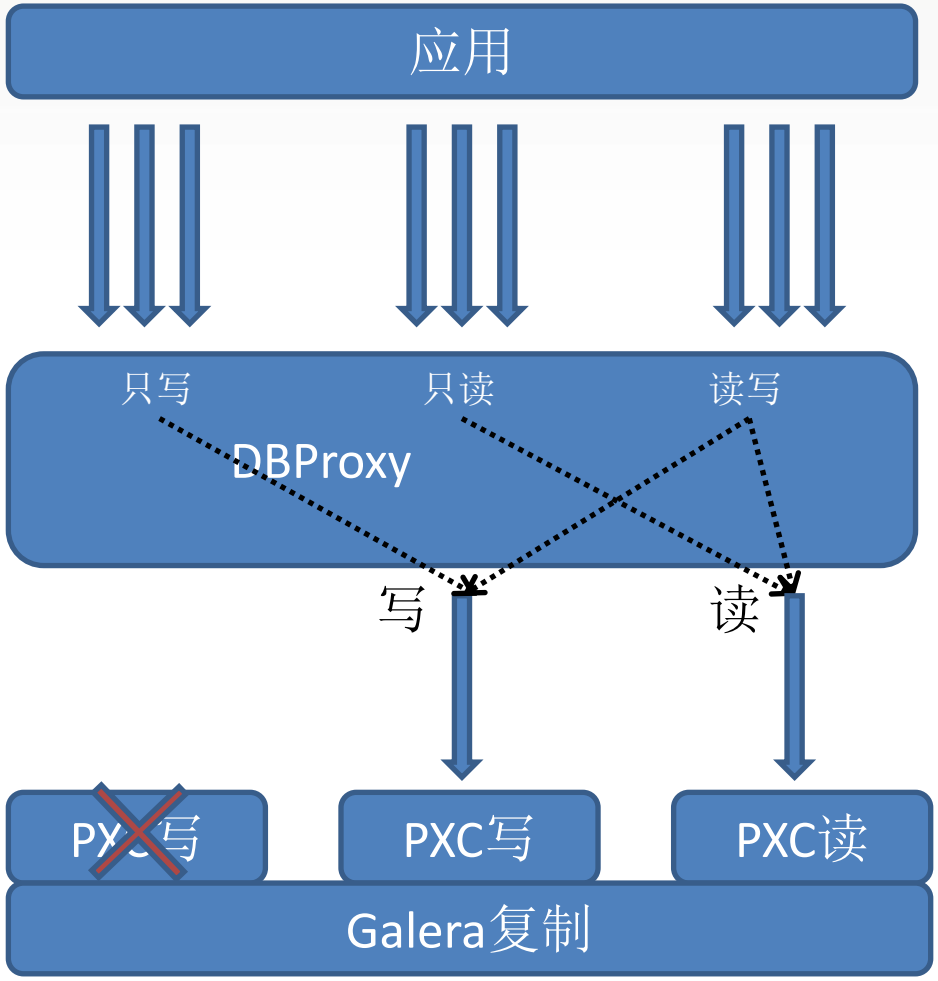
- 单点写，多点读
- 写操作不做负载均衡
- 读负载均衡算法：
  - 最小连接数
  - 加权轮询
- 读操作不发往写后端
  - 只剩一台数据库时，读写在一起

故障转移 一



- 摘除失效的只读后端
- 保持前端连接（不在用的）

故障转移 二



- 摘除失效的写后端
- 按权重选择一个只读，提升为写
- 保持前端连接（不在用的）

连接复用概述

- 目标：实现类似10：1的连接复用效果
- 策略
  - 相同用户才会复用连接
  - 不需要等用户Client关闭才连接复用
  - Client长时间不使用就可以复用
  - 具体实现：
    - 连接复用时需要重新根据Client属性恢复上下文（比如设置当前数据库）
    - 定义“什么叫Client长时间不使用”

Client长时间不使用—举例

- 事务提交&prepare关闭&缓存一端时间
- 举例
  1. prepare st=“update blog set stat=‘1’ where pubdate=?”
  2. set autocommit=0
  3. bind st,’2013-1-1’
  4. execute st
  5. commit
  6. close st
  7. select row_count()
- 第7步以后就可以复用

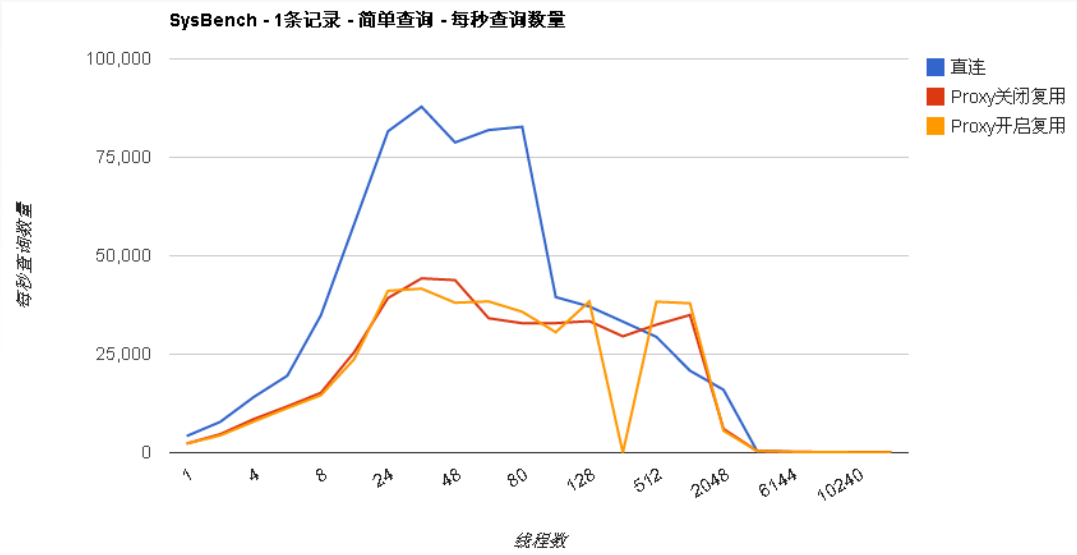
什么叫Client长时间不使用

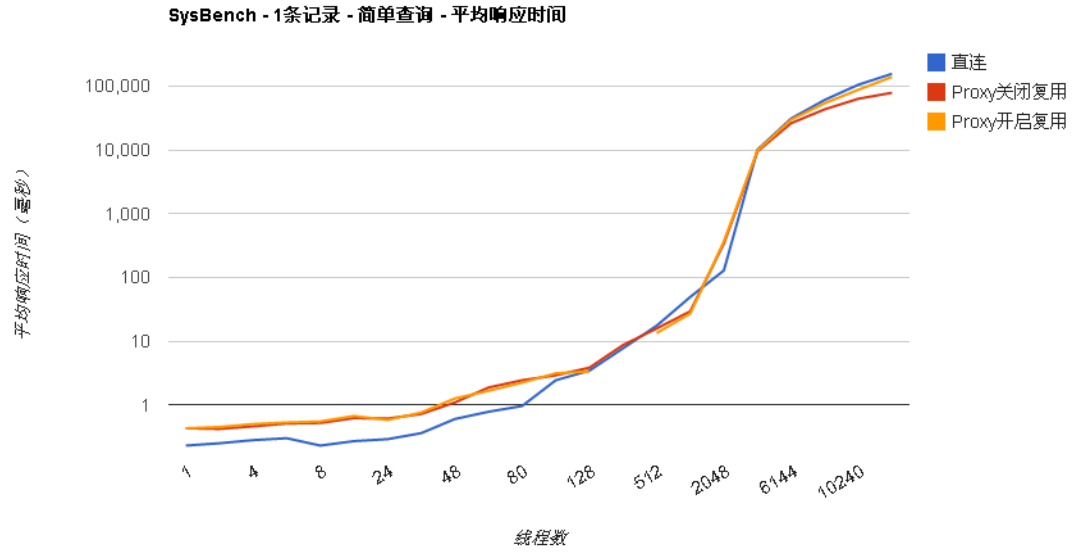
- 下面3种情况都满足才可复用
- 3种情况
  - prepare所有被关闭
  - 事务均已提交
  - 再保留一段时间后
    - 特殊函数last_insert_id()、select_found_rows()等是上下文相关

概述

- 面临的问题
- DBProxy介绍
- DBProxy实践
- 下一步计划

性能测试





- 大压力响应时间衰减2~8ms之间
- QPS性能下降25%-50%左右
- CPU负载高，SYS很高，USR较高
- 性能瓶颈，网络、锁

http://wiki.no.sohu.com/index.php/DBProxy%E6%B5%8B%E8%AF%95#.E6.B5.8B.E8.AF.95.E6.8A.A5.E5.91.8A

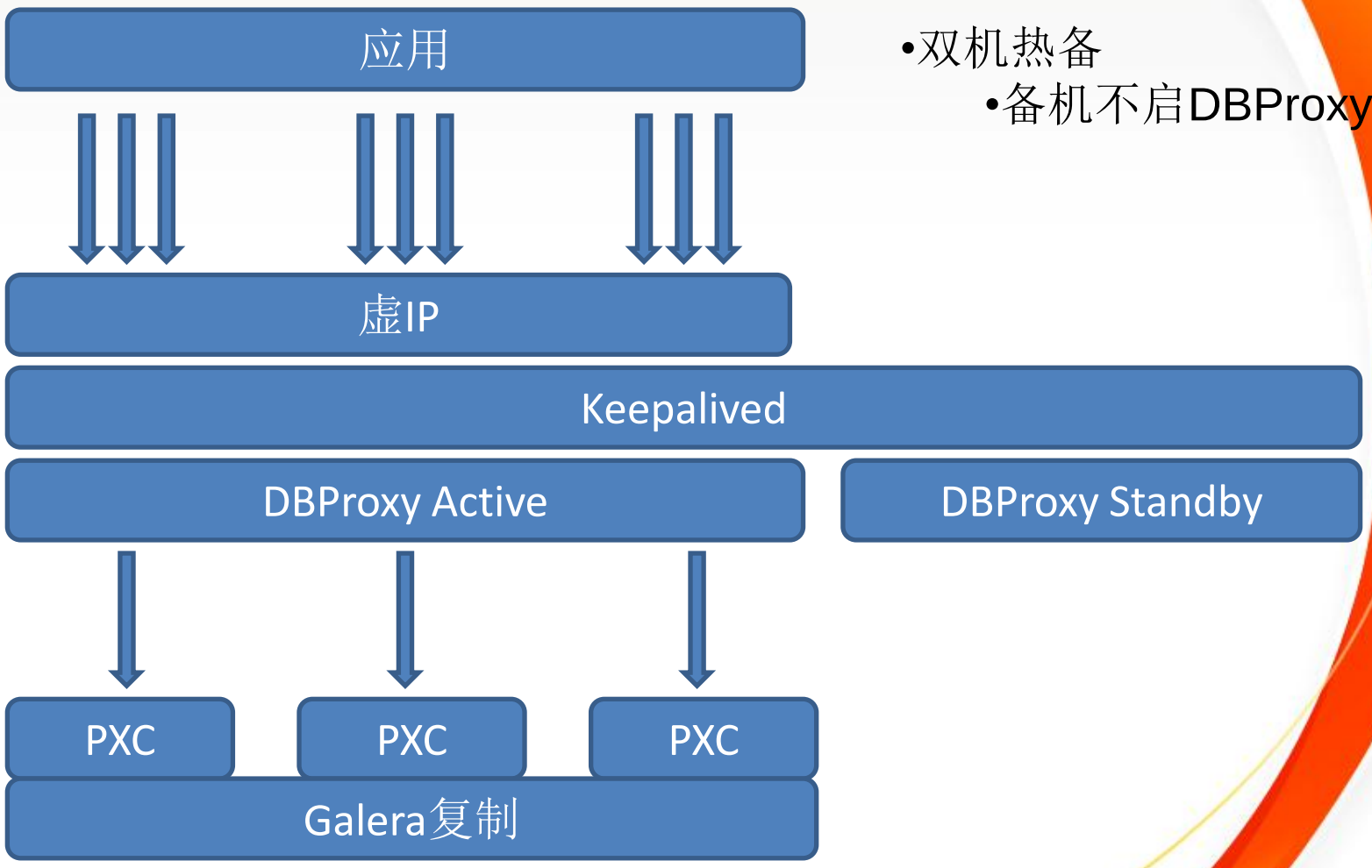
使用限制

- 后端数据库是PXC，未来支持MySQL复制
- 不支持LOAD DATA，多语句查询”insert …; update …; select …”
- 连接复用打开时，不支持程序里使用变量
  - 除了character_set_client/connection/result, autocommit
- JDBC setQueryTimeout()，URL多地址(loadbalance,replication)
- 不支持SSL，压缩，连接属性CLIENT_IGNORE_SPACE

常用命令

- 用户登不上
  - showusers 用户名/IP/最大连接数。还要登后端数据库看
  - showbackends 后端数据库状态
  - showpoolstatus 连接池状态
- 查询结果集出错
  - show/setmultiplexswitch 关闭连接复用
- 语句执行慢
  - ShowQueryResponseTime
- 其它
  - SetLogLevel 修改日志输出级别DEBUG

高可用架构



- 双机热备
  - 备机不启DBProxy

DBProxy高可用

- Keepalived选主、绑定虚IP
- 读写虚IP都绑定在一个VRRP实例上
- 具有虚IP的主机上启DBProxy进程，备机上不启
- 所有数据库都down，就切换
- 通过虚IP去ping网关，避免脑裂，删除虚IP
- 自动同步DBProxy的配置文件etc/*.{cnf,xml}
  - 主到从
  - 用rsync，需要配SSH自动登录

故障回顾

- 2013-11-19 凌晨 某业务线
  - 其中一个备库备份压力大，DBProxy检查超时，这个库状态变成down
  - DBProxy后端检查进程3个，检查频率是10秒，脚本的超时是通过agent模块实现，时间设置太长是30秒。一次超时检查堵30s,最后所有进程都堵在超时这里，正常的库也不能检查了，也变成了down
  - 所有库down，引起高可用切换
  - 重复前面的过程，两台机器之间不断切换

项目文档及其他

TIPS：

$DBPROXY_HOME/etc/mysql-proxy.cnf

dbproxy-collation: 影响到结果的正确性

$DBPROXY_HOME/etc/zabbix_agentd.cnf

Timeout=10 （DBProxy检查间隔）

StartAgents=10 （一个检查任务会占用一个进程）

概述

- 面临的问题
- DBProxy介绍
- DBProxy实践
- 下一步计划

提问