SACC 2015中国系统架构师大会
SYSTEM ARCHITECT CONFERENCE CHINA 2015
互联网+ 重塑IT架构

游戏云存储的架构变迁之路

个人简介

- 腾讯游戏互动娱乐事业群(IEG) DBA
- 梁飞龙(英文ID: felixliang)
- 毕业8.8年, 5.8年腾讯DBA经验: DB运维及开发

目录

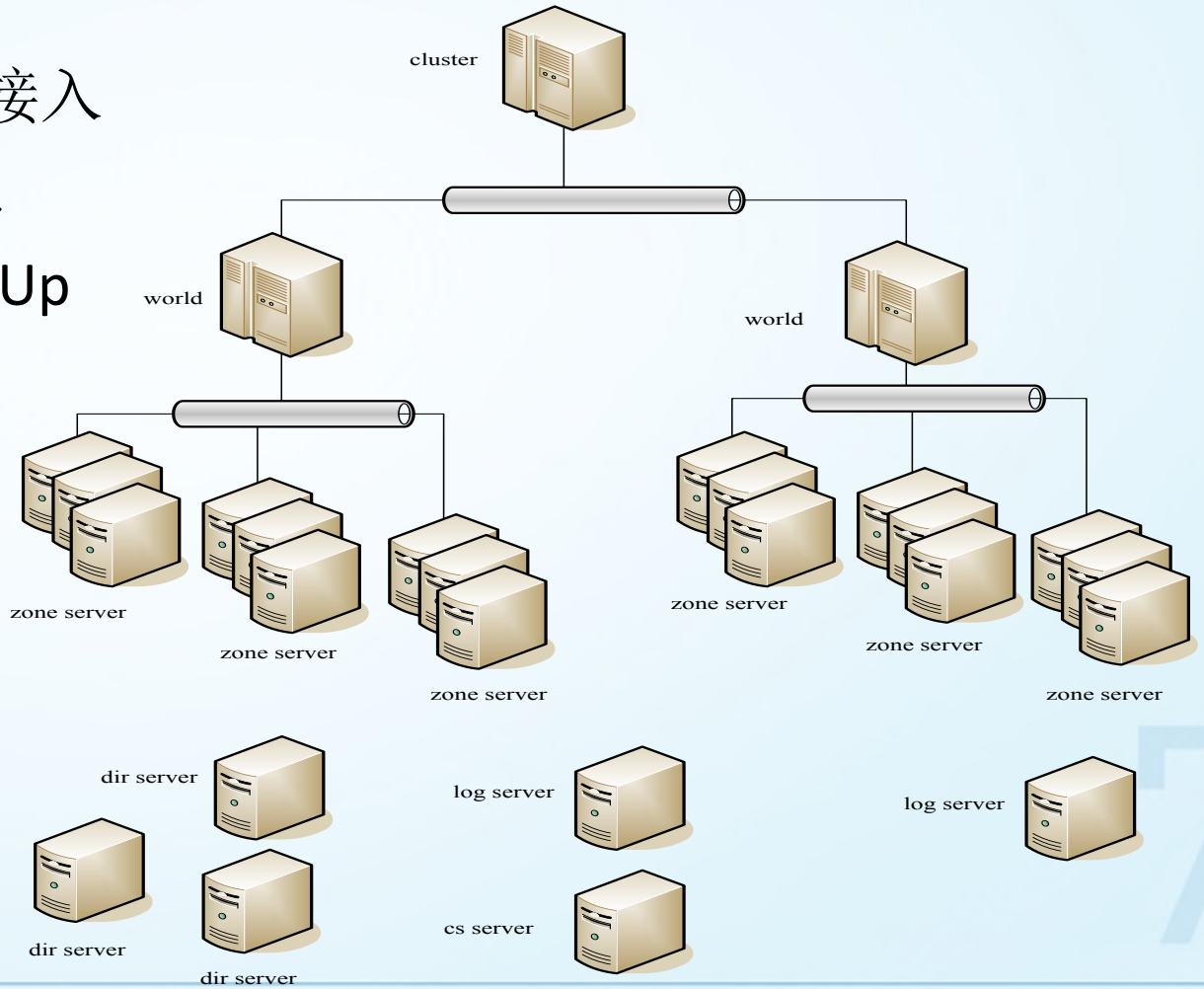
- 单实例时代
  - 游戏DB分布介绍
  - 自动化提供海量DB服务
- 多实例时代
  - 高可用/灵活调度
  - MySQL源码定制
- 分布式时代
  - TSpider动态调度（在线扩容及缩容)

单实例时代，游戏DB分布及架构

- 典型游戏DB分布介绍
  - 大型多人在线游戏(MMOG): 三国/地下城与勇士
  - 高级休闲游戏(ACG): QQ飞车/英雄联盟
  - 平台休闲游戏(PLAT):玫瑰小镇/QQ游戏大厅
- 游戏DB架构简化
  - 核心数据 热备
  - 日志数据 单实例

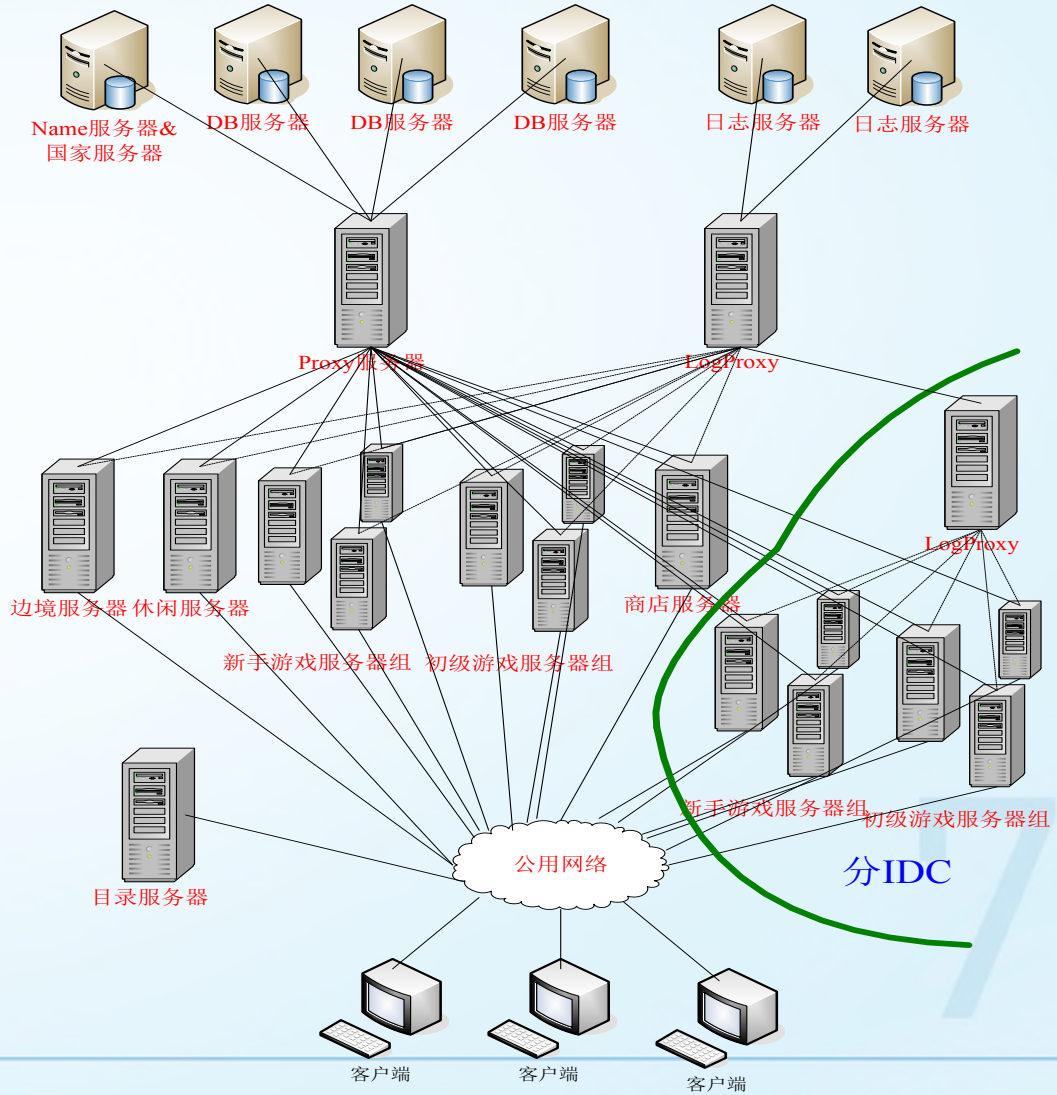
MMOG游戏DB分布

- 部署策略：就近接入
- 切分策略：SET化
- 承载策略：Scale Up



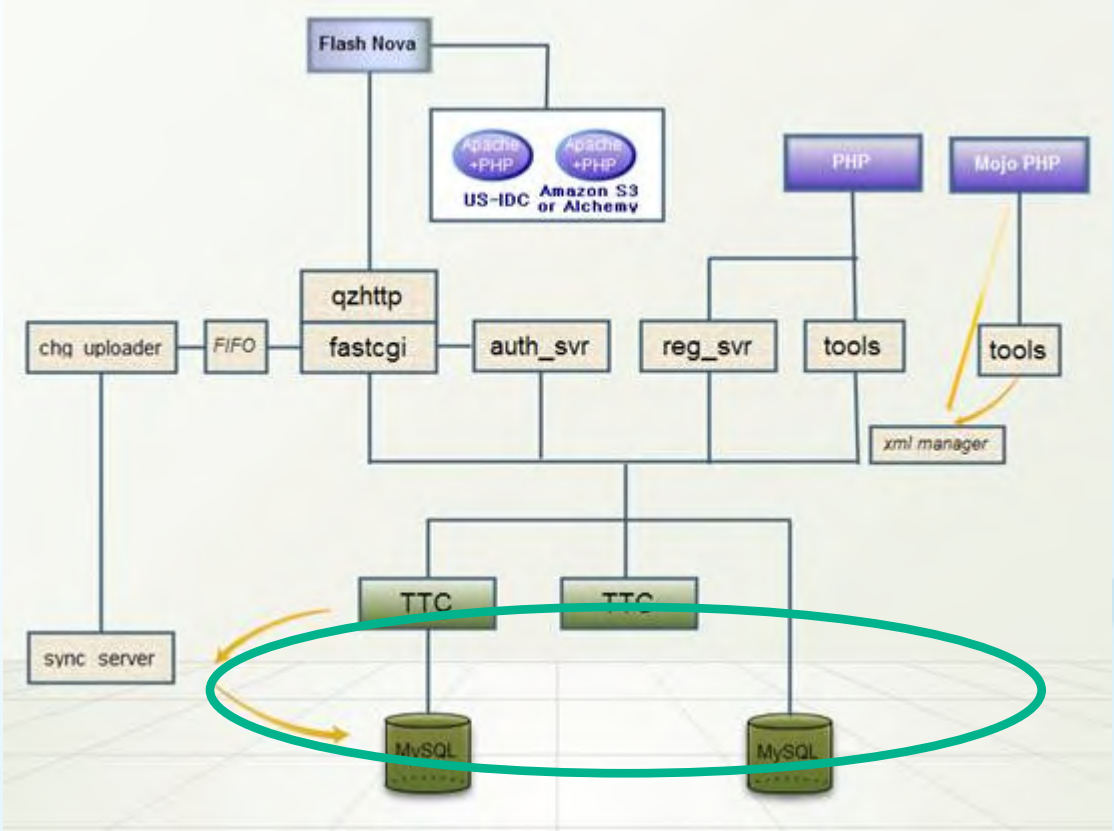
ACG游戏DB分布

- 部署策略：就近接入
- 切分策略：水平 + 垂直
- 承载策略：Scale Out



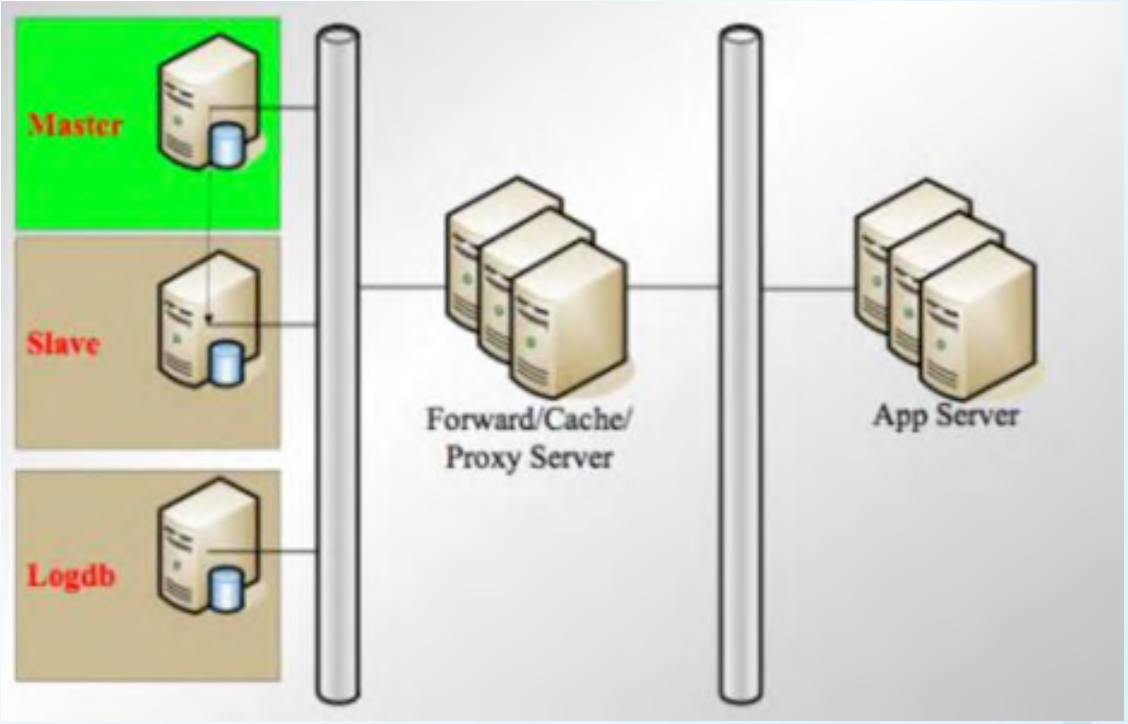
PLAT游戏DB分布

- 部署策略：集中部署，跨IDC容灾
- 切分策略：水平
- 承载策略：Scale Out



游戏DB架构简化

- 核心数据 增加热备
- 日志数据 单实例



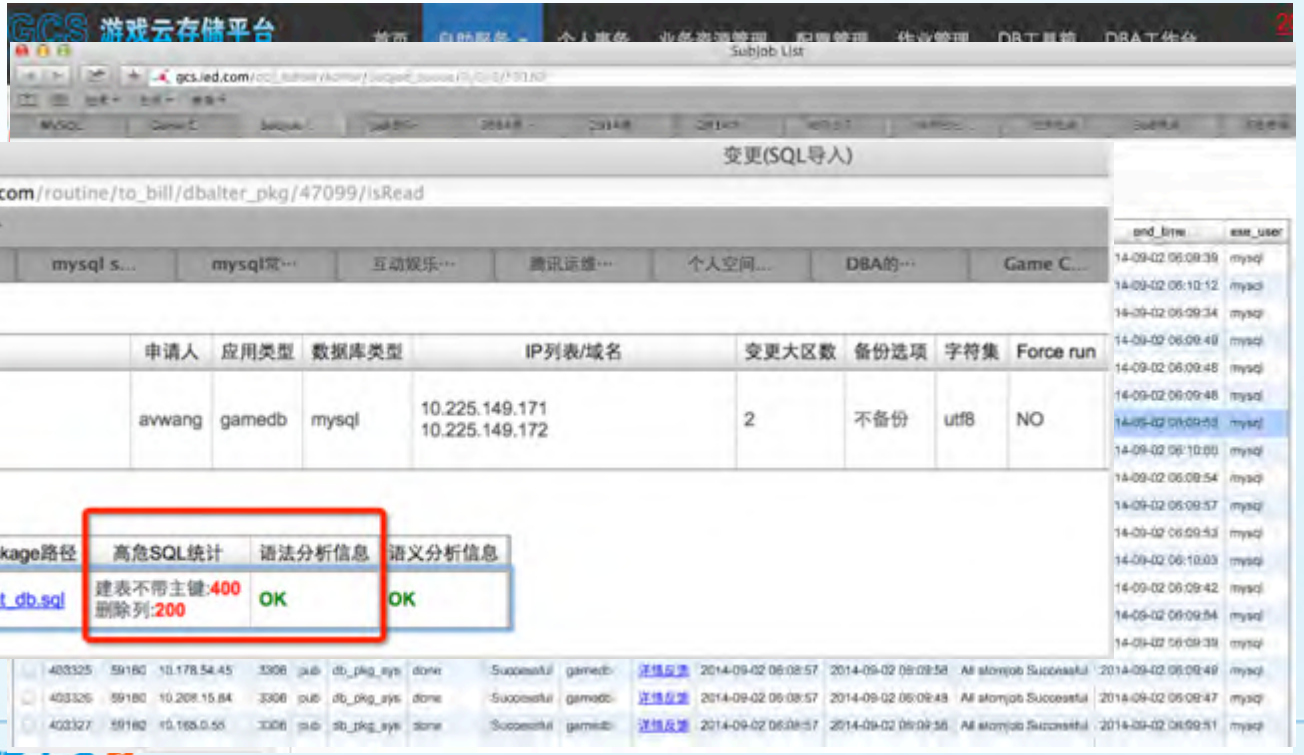
单实例时代，自动化提供海量DB服务

- 效率为王，工具平台解放DBA双手

250+款游戏(端游+手游)、10000+台服务器、20000+个实例

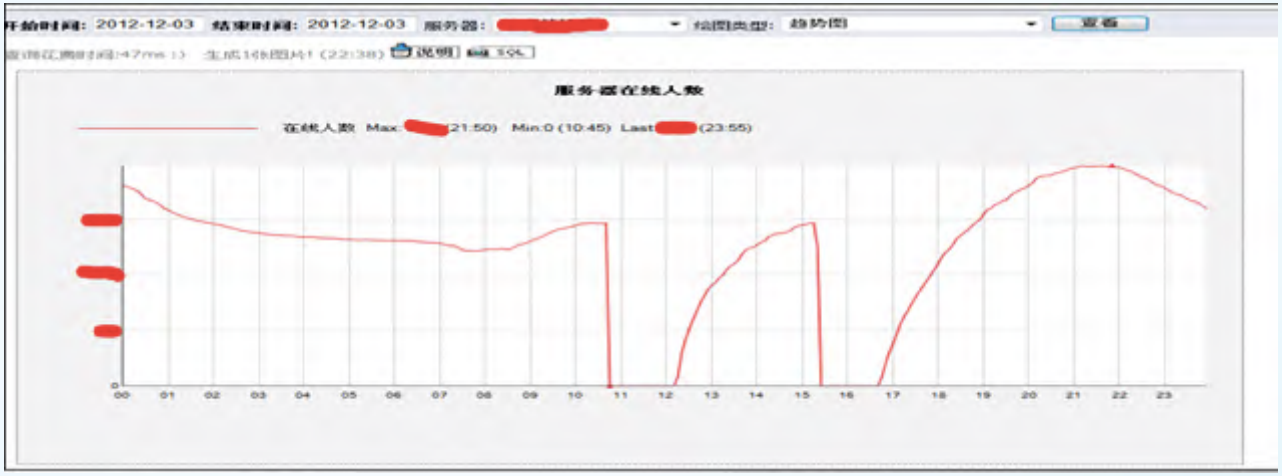
690次SQL变更/月，人均每天支撑3个业务SQL变更，人均管理着500台机器、1000个实例

- DB服务举例



单实例DB管理的痛点

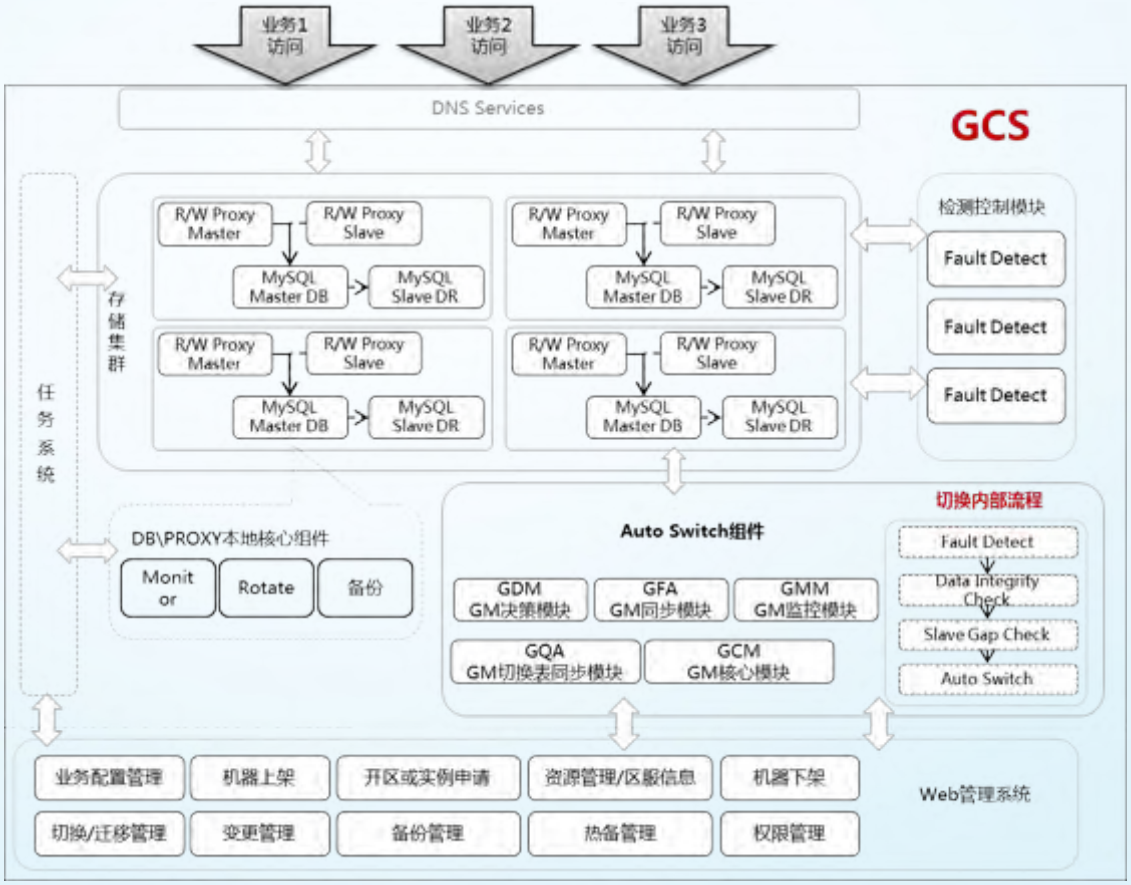
- 硬件故障影响用户时间长



- DB无法实现调度，DB Scale耗费停机时间长
- DB加字段耗时长，无透明DB压缩方案

多实例时代，高可用/灵活调度

- 游戏云存储架构1.0

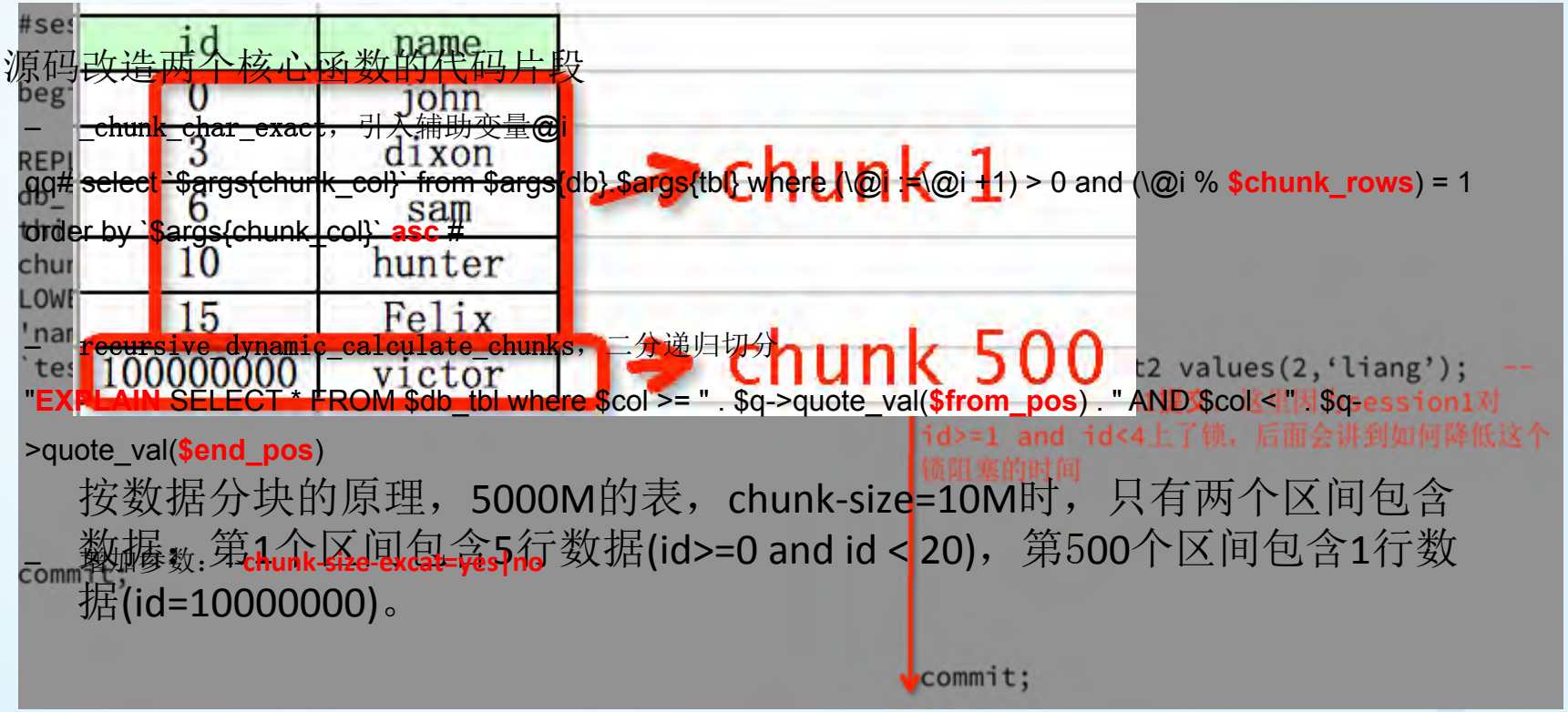


Auto Switch组件

- MySQL-Proxy扩展(基于0.8.2)
  - 去掉lua扩展，提升性能
  - 扩展ADMIN接口
    - refresh_backends\refresh_users\refresh_connlog
    - show processlist\show balances
- 监控
  - 多点监控，IDC内/IDC外
  - 进程探测、SSH探测及Touch文件
  - Double Check
- 切换前检查
  - 数据一致性检查
  - Slave状态检查
  - Time Delay

故障切换数据一致性保证

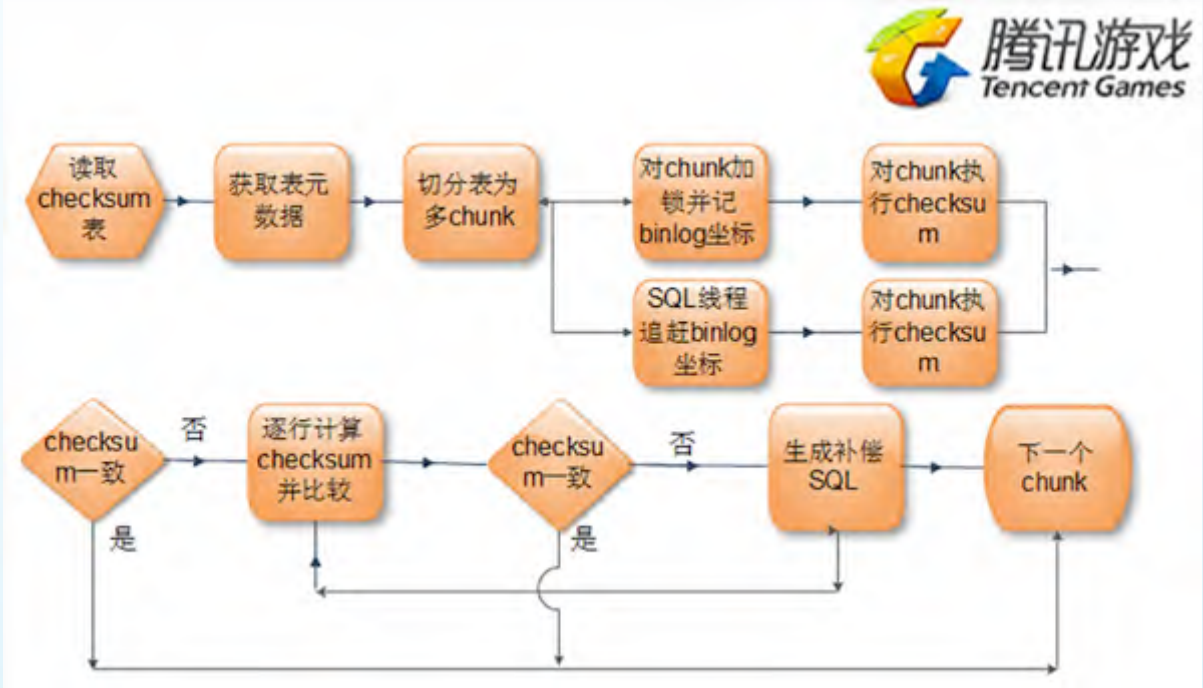
- 数据自动校验pt-table-checksum例行化
  - 数据块切分不均在可重复度隔离级别下的“锁数据”问题
- 源码改造两个核心函数的代码片段



按数据分块的原理，5000M的表，chunk-size=10M时，只有两个区间包含数据，第1个区间包含5行数据(id>=0 and id < 20)，第500个区间包含1行数据(id=10000000)。

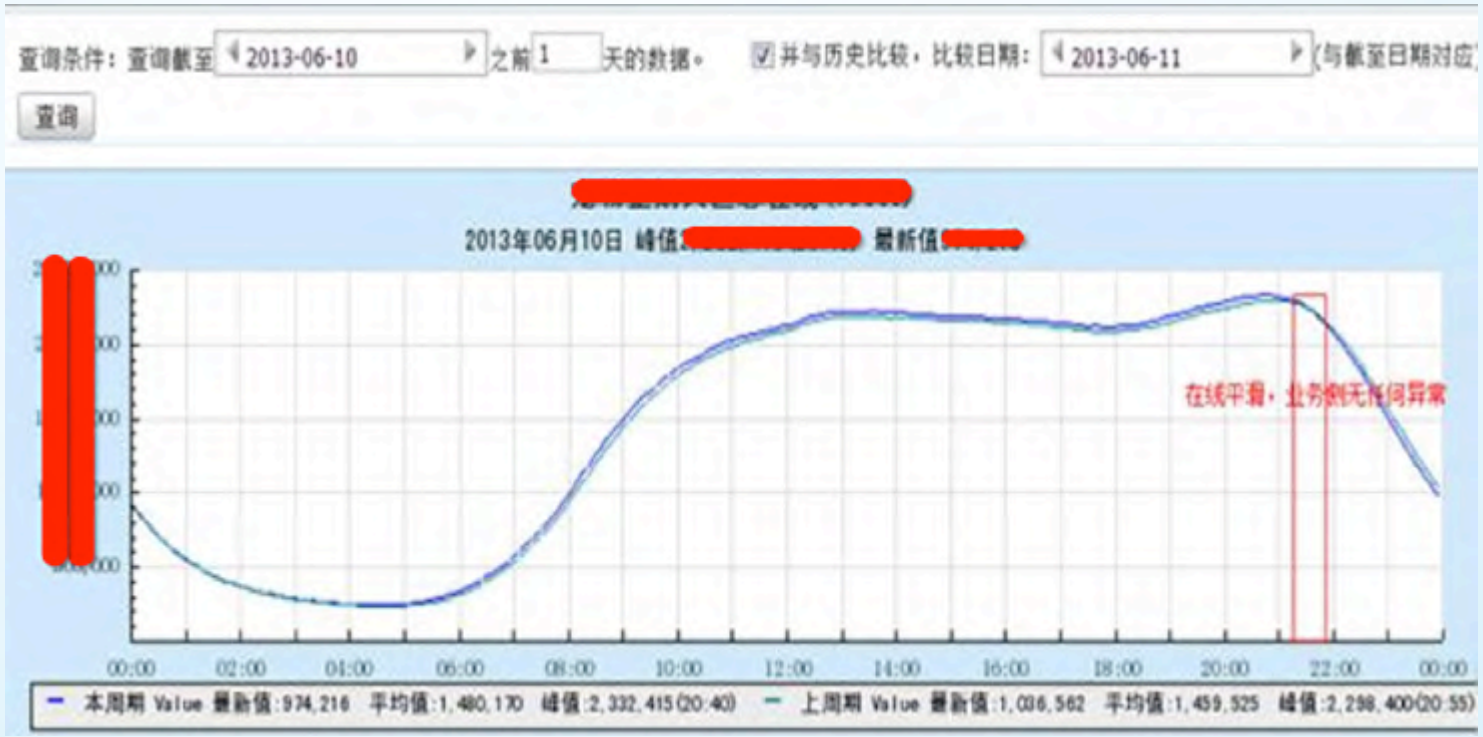
故障切换数据一致性保证

- 数据自动修复pt-table-sync
  - replicate模式下，修复SQL改为直接在Slave上执行
  - replicate模式下，普通索引使用delete+insert方式修补数据
  - replicate模式下，结合--where参数做”差异”数据块修复



高可用故障切换效果

- 机器故障切换，几乎不影响游戏在线



基于Proxy灵活调度

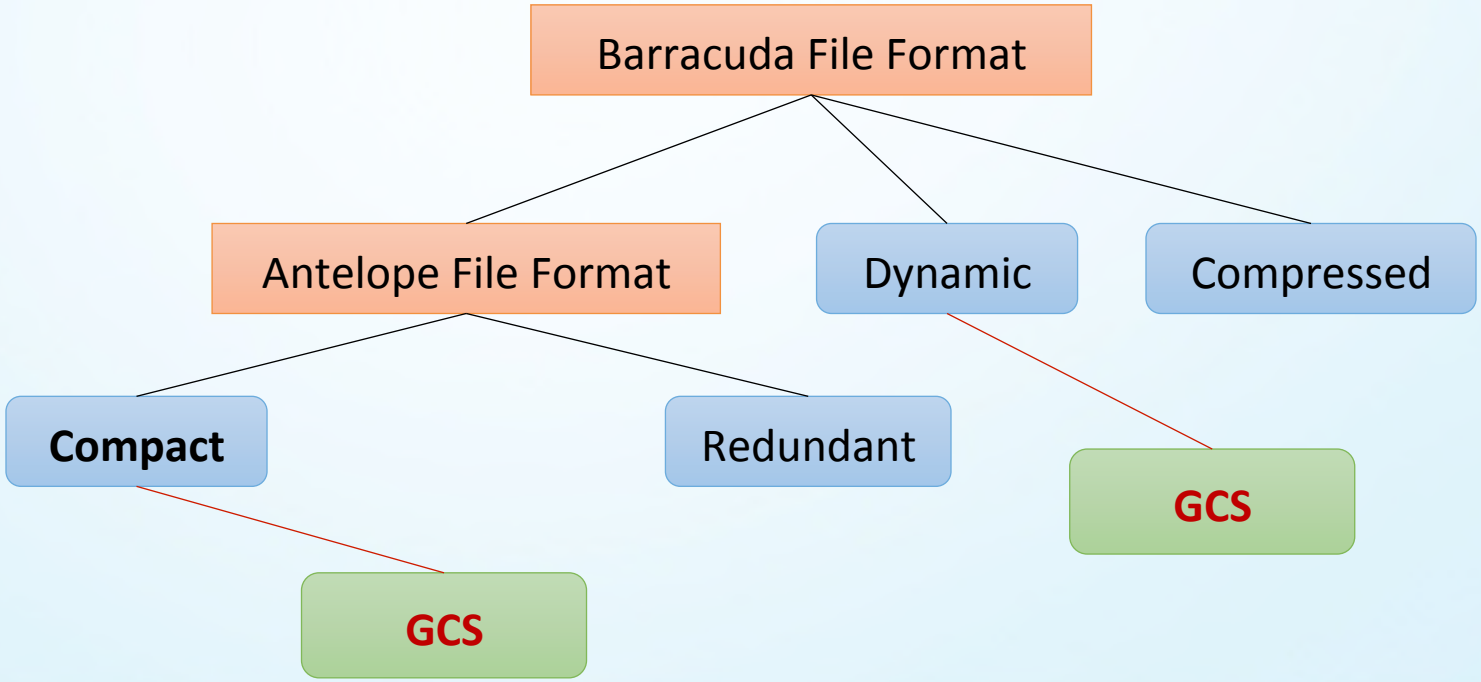
- 业务高峰期调整Proxy的后端backends的效果

多实例时代，MySQL源码定制

- 在线加字段
- 大字段列压缩
- binlog压缩及binlog限速

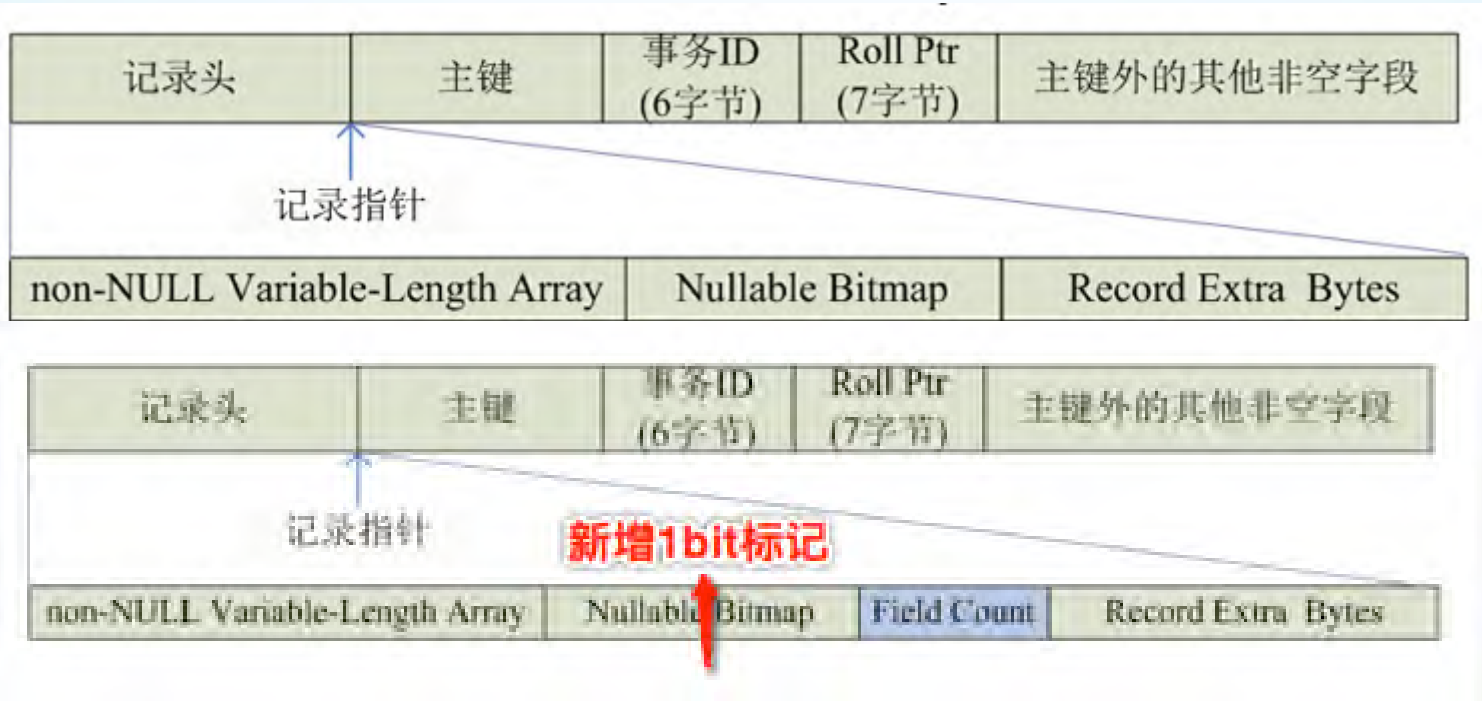
MySQL源码定制之路

- 在线加字段的原理
  - GCS存储格式
    - TMySQL为在线加字段功能新增行格式GCS
    - 原理：扩展原Compact及Dynamic行格式，增加1~2字节控制信息



MySQL源码定制之路

- 在线加字段的原理



select：判断GCS行标记，对比fieldcount大小，实际列数大于Row Field count即为null或者默认值
insert：根据当前表的字段数构造记录中field count和置GCS行标记位
delete：保持不变
update：原地更新不变；非原地更新走delete+insert，会更新为新的field count

MySQL源码定制之路

- 大字段列压缩的背景
  - 游戏内大量使用blob或text来简化数据访问
  - 多实例管理模式下磁盘和内存是瓶颈
  - 无有效的数据压缩方案
    - 依赖于开发进行数据压缩，周期长且太被动
    - MySQL页压缩的空间预留问题及uncompress page问题

MySQL源码定制之路

- 大字段列压缩的实现
  - 适配GCS行格式，支持blob/text类字段
  - 字段内容变为

首字节标记（1 字节）	解压后长度（0-4 字节）	压缩的内容

    - 是否压缩：首字节最高bit，若为1表示已压缩；若为0表示不压缩，同时解压后长度为空。
    - 多种压缩算法：首字节5~6 bit用于可表示4种不同压缩算法，目前只支持zlib（0）
    - 其他特性扩充预留
  - 支持行外存储

MySQL源码定制之路

- 大字段列压缩的用法
  - 建表

```
create table t1 (
  C1 int primary key,
  C2 blob compressed,
  C3 text character set gbk compressed,
  C4 blob
 ) engine = innodb row_format=GCS
```

  - 修改表

```
alter table t1 change c4 c4 blob compressed
```

MySQL源码定制之路

- 大字段列压缩的收益

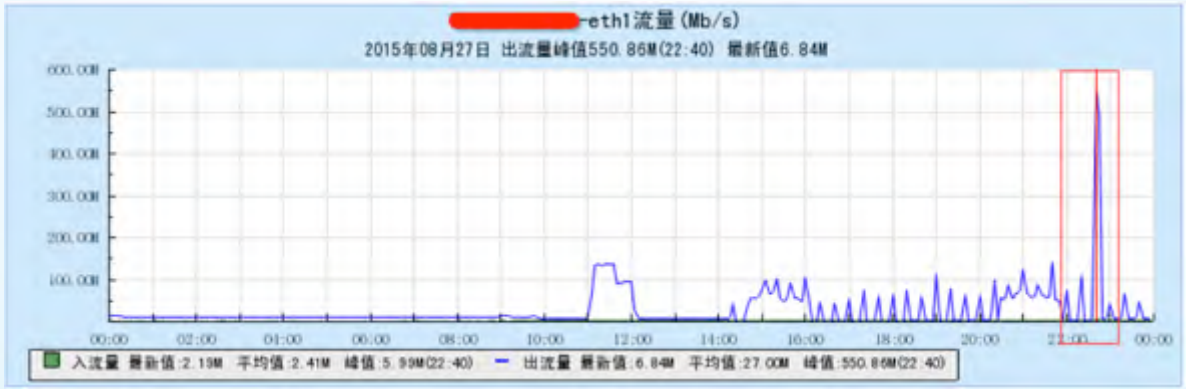
原始数据51G

服务器配置：24core/64Gmem/fusionIO，buffer_pool=32G

	非压缩 原始数据	Innodb页压缩 key_block_size=8	列压缩 （blob/text大字段压缩）
数据量	51G	24G	7.1G
QPS （100个并发update）	1174	1524	3994
IO util	100%	100%	30%
CPU util	15%	45%	50%

MySQL源码定制之路

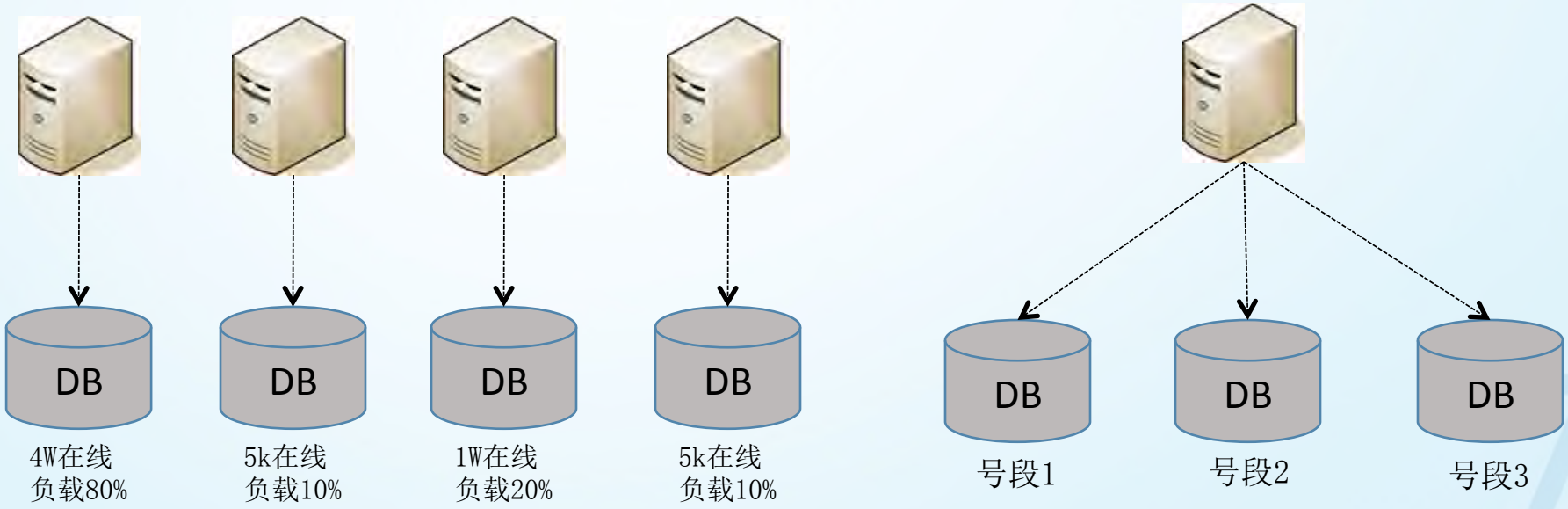
- binlog压缩
  - 单实例最多生成360G/天，本地及异地存储困难(网络传输)
  - 支持语句级/行级binlog压缩，可节省42%~69%的空间
- binlog限速
  - 多实例管理模式下，重做slave造成网络阻塞



- 令牌限速算法，限制binlog传输速率，譬如控制在100Mb/s以下

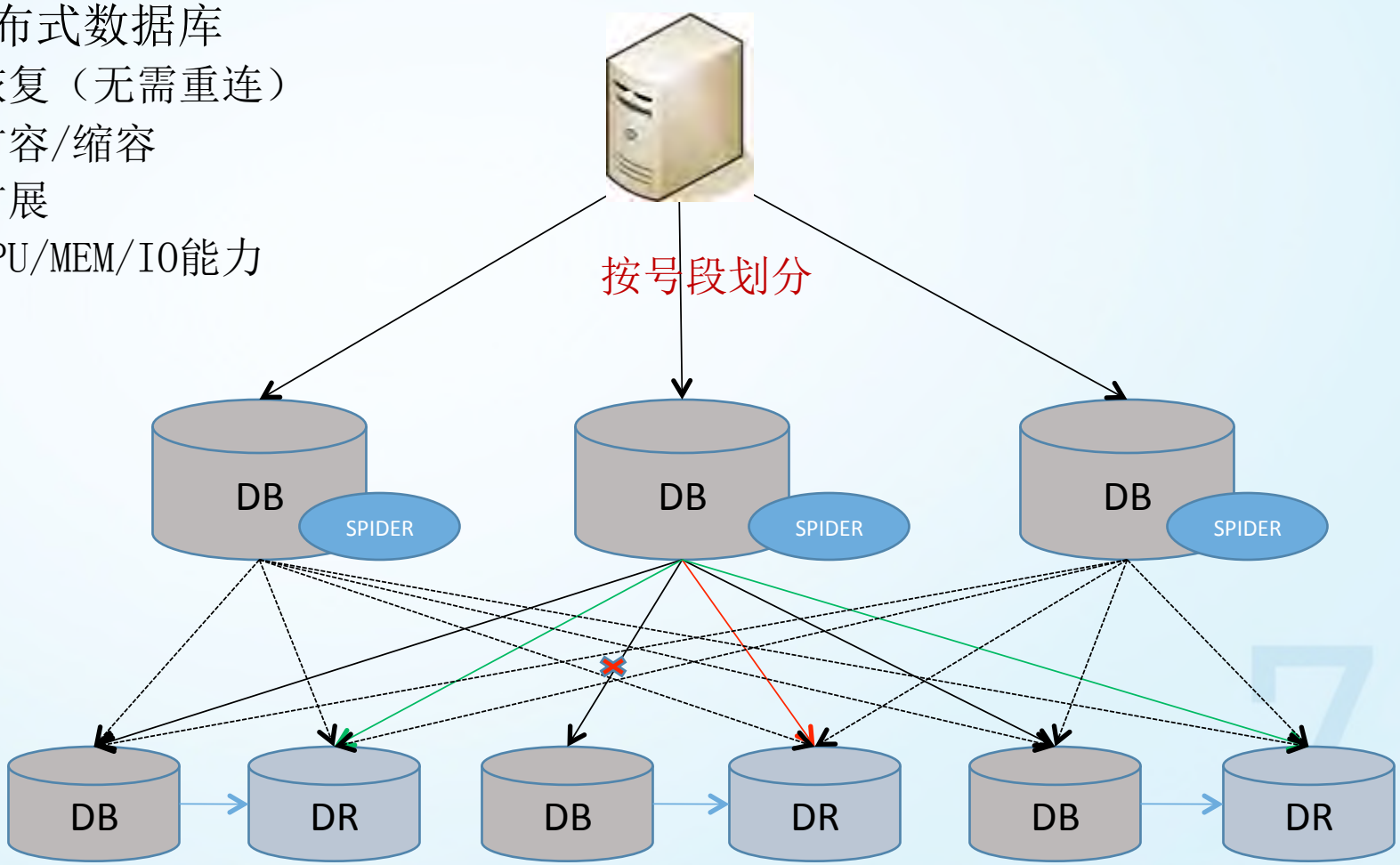
多实例DB管理的痛点

- MySQL实例彼此孤立，无法实现资源共享和负载均衡
- MySQL存在单机性能瓶颈，无有效的动态扩容与缩容机制



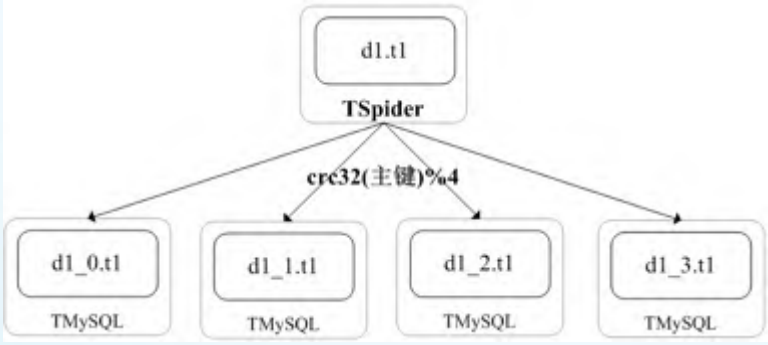
分布式时代，TSpider动态调度

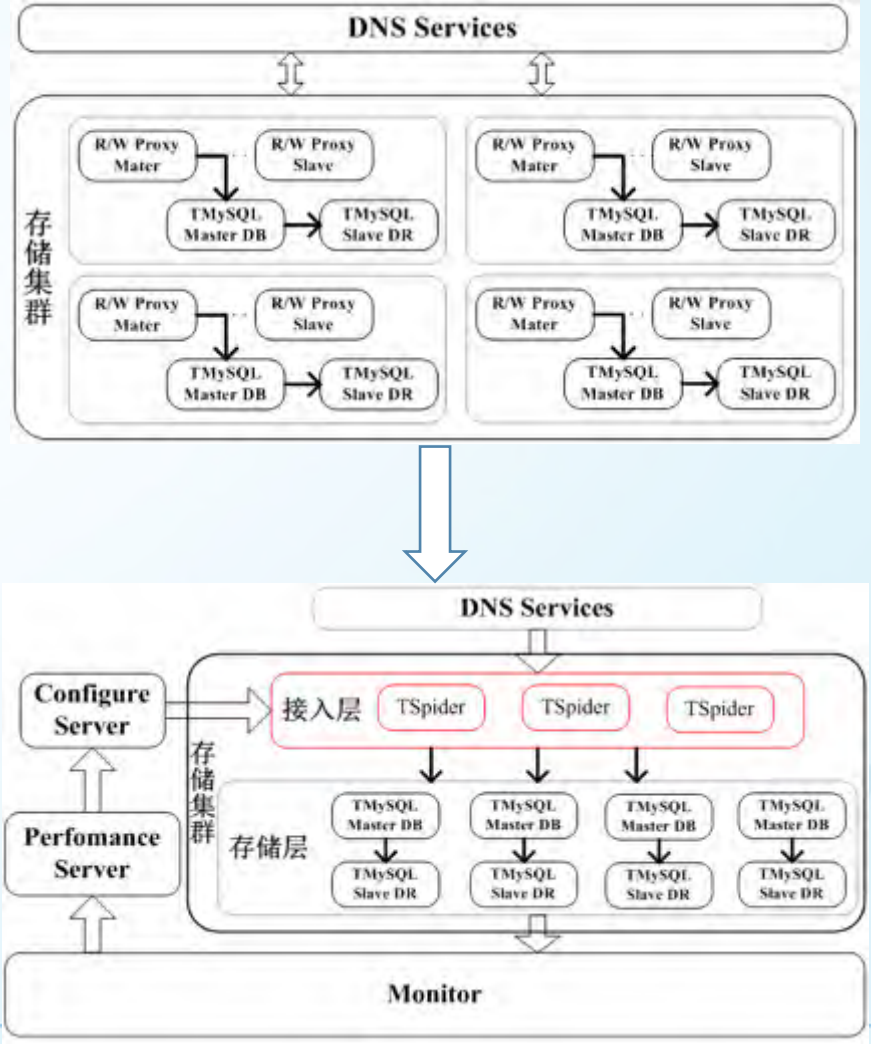
- TSpider分布式数据库
  - 故障恢复（无需重连）
  - 动态扩容/缩容
  - 水平扩展
  - 均衡CPU/MEM/IO能力



TSpider分布式数据库

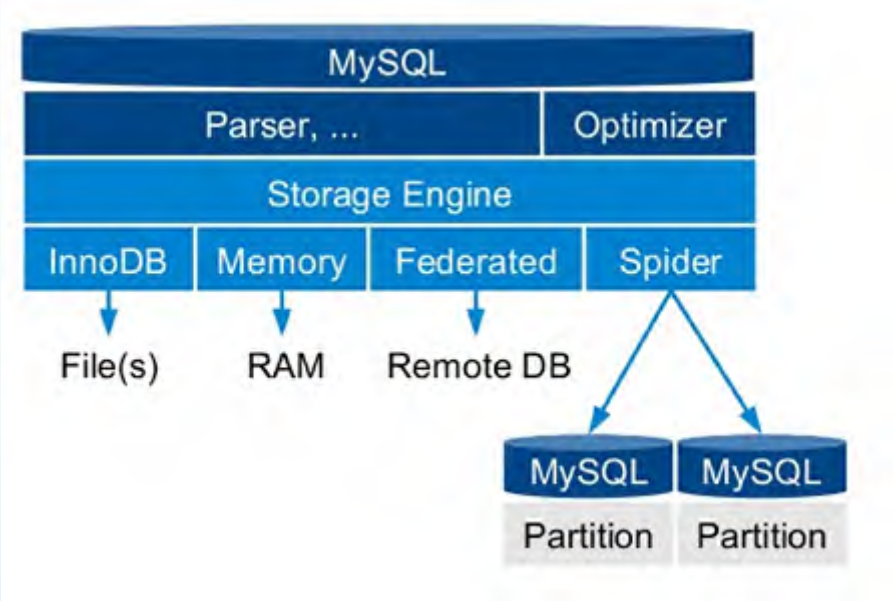
- 游戏云储存的转变
  - 实例管理 -> 集群管理
  - 自动分表，应用透明





TSpider分布式数据库

- Spider存储引擎
  - Kentoku　SHIBA 开发的基于分区表的分布式存储引擎
  - http://spiderformysql.com
- TSpider就是spider 3.1基础上开发而成，进一步提高了性能、稳定性和兼容性，并结合互娱业务特性整合而成的分布式数据库解决方案

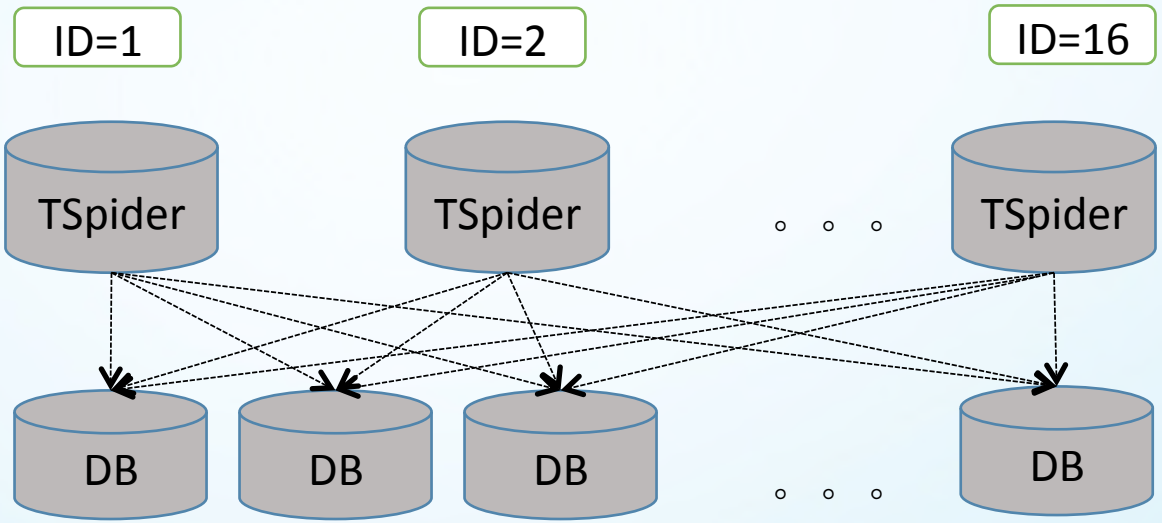


TSpider分布式数据库

- TSpider核心特性
  - 高效全局自增ID
  - 自定义分区策略
  - 线性扩展(在线扩容缩容)
  - 兼容MySQL协议及性能优化
  - SQL并行化

TSpider分布式数据库

- 高效全局自增ID
  - 初始编号不同，自增的步长=集群TSpider最大节点数(**16**)
  - TSpider重启后，需select max获得最大值



- 可保证全局自增且唯一，但不保证+1
- ID生成效率高但存在空洞

TSpider分布式数据库

- 自定义分区策略
  - 一对多的ER关系，按照”一”进行切分，减少跨分区SQL操作

```
Create Table: CREATE TABLE `mail` (
  `id` int(20) unsigned NOT NULL,
  `accountid` int(11) unsigned NOT NULL,
  ……
  PRIMARY KEY (`id`),
  KEY `idx_accountid` (`accountid`),
  KEY `idx_etime` (`etime`)
) ENGINE=SPIDER DEFAULT CHARSET=utf8
```

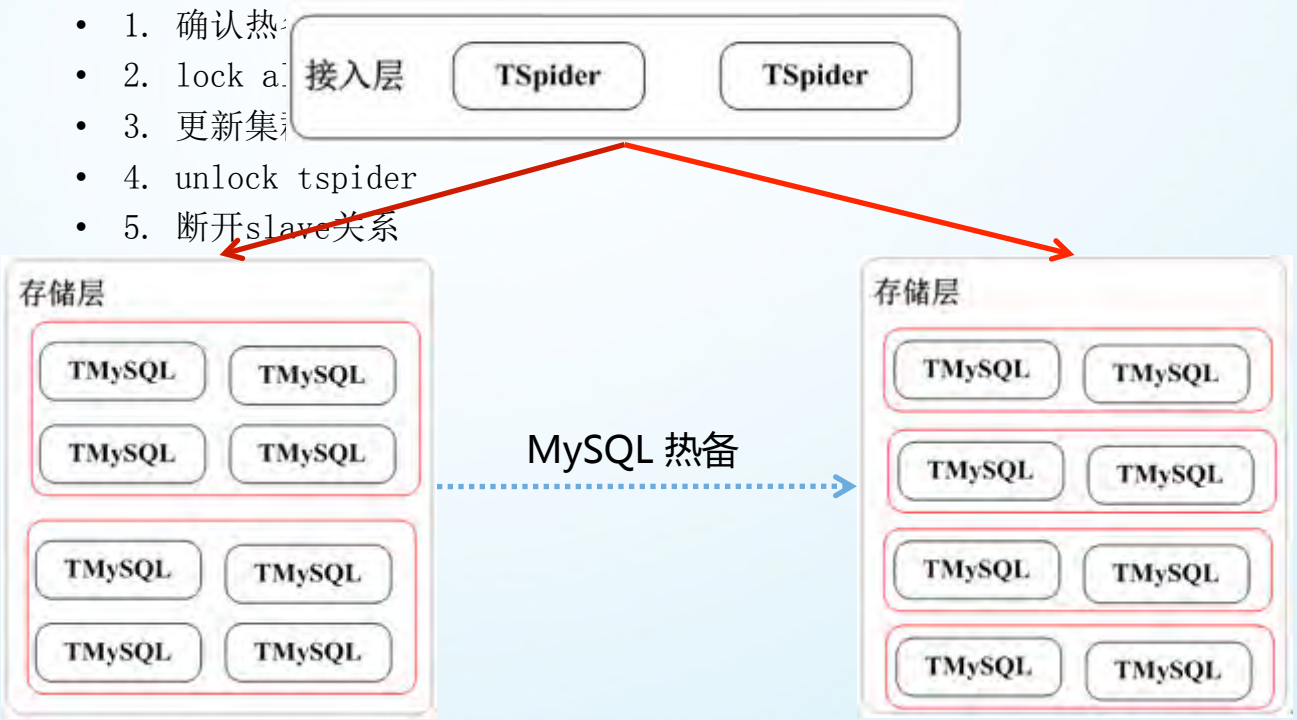
/*! shard_key "accountid" */

用户访问邮件表的SQL

```
# SELECT count(1) FROM mail WHERE accountid = 102393935;
# SELECT id, accountid, state, sender, sendername, type, title, content, etime,
ctime, hasitems, mailitem FROM mail WHERE accountid = 102393935 ;
# DELETE FROM mail where accountid=102393935 and id = 142028645;
```

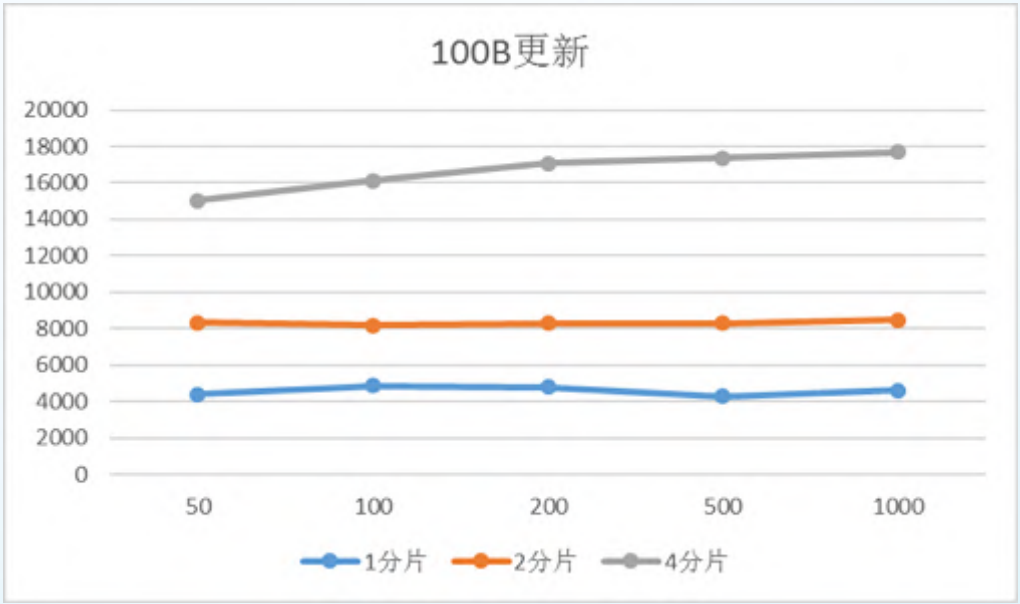
TSpider分布式数据库

- 线性扩展(在线扩容缩容)
  - 例：一个集群8个shard，存储层两个物理机器
  - 扩容一倍
  - 具体步骤
    - 1. 确认热
    - 2. lock a
    - 3. 更新集
    - 4. unlock tspider
    - 5. 断开slave关系



TSpider分布式数据库

- 线性扩展(性能)
  - 集群整体QPS随分片数增加而线性提升



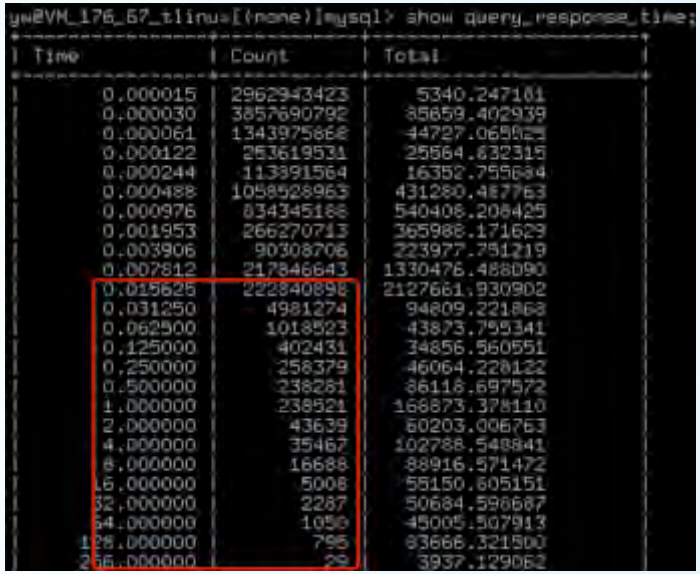
TSpider分布式数据库

- 兼容MySQL协议
  - 支持99.99%SQL语句（join/sum/group by/order by/in/not in/limit/case when)
  - 支持所有mysql连接协议，如：CLIENT_FOUND_ROWS
- 性能优化
  - 减少网络流量　　-> select limit m, n
  - 增加direct SQL　-> delete/update limit n
  - 智能下推　　-> select c1, count(*) from t1 where group by c1
- 执行计划相关
  - 定期统计远程表信息 show table status
  - 禁用index merge，减少SQL分发数量 select * from t1 where key1 = 1 or key2 = 2

```
gw@VM_151_4_tlinux[(none)]mysql> show variables like 'optimizer_switch'\G
*************************** 1. row ***************************
Variable_name: optimizer_switch
        Value: index_merge=off,index_merge_union=off,index_merge_sort_union=off,index_merge_intersection=off,engine_condition_pushdo
wn=on,index_range=off
1 row in set (0.00 sec)
```

TSpider分布式数据库

- SQL并行化
  - 某实例2%的SQL超过10ms
  - 绝大多数是跨分区SQL导致



```
yw@VM_176_67_tlinux[(none)]mysql> show query_response_time;
+----------------+-------------+-----------------+
| Time           | Count       | Total           |
+----------------+-------------+-----------------+
|       0.000015 | 2962943423  |     5340.247181 |
|       0.000030 | 3857690792  |    85659.402939 |
|       0.000061 | 1343975868  |    44727.065[illegible] |
|       0.000122 |  253619531  |    25564.632315 |
|       0.000244 |  113891564  |    16352.755684 |
|       0.000488 | 1058528963  |   431280.467763 |
|       0.000976 |  834345188  |   540406.208425 |
|       0.001953 |  266270713  |   365988.171629 |
|       0.003906 |   90308706  |   223977.751219 |
|       0.007812 |  217846643  |  1330476.488090 |
|       0.015625 |  222840898  |  2127661.930902 |
|       0.031250 |    4981274  |    94809.221868 |
|       0.062500 |    1018523  |    43873.795341 |
|       0.125000 |     402431  |    34856.560551 |
|       0.250000 |     258379  |    46064.228122 |
|       0.500000 |     238281  |    86118.697572 |
|       1.000000 |     238521  |   166873.378110 |
|       2.000000 |      43639  |    60203.006763 |
|       4.000000 |      35467  |   102788.548841 |
|       8.000000 |      16688  |    88916.571472 |
|      16.000000 |       5008  |    55150.605151 |
|      32.000000 |       2287  |    50684.596687 |
|      64.000000 |       1050  |    45005.507913 |
|     128.000000 |        795  |    83666.321500 |
|     256.000000 |         29  |     3937.129062 |
```

- TSpider到RemoteDB的延迟，16节点至少可节省2ms

```
[mysql@VM_176_67_tlinux /data/mysqllog/25000]$ ping 10.219.19.241
PING 10.219.19.241 (10.219.19.241) 56(84) bytes of data.
64 bytes from 10.219.19.241: icmp_seq=1 ttl=60 time=0.133 ms
64 bytes from 10.219.19.241: icmp_seq=2 ttl=60 time=0.145 ms
64 bytes from 10.219.19.241: icmp_seq=3 ttl=60 time=0.144 ms
64 bytes from 10.219.19.241: icmp_seq=4 ttl=60 time=0.146 ms
64 bytes from 10.219.19.241: icmp_seq=5 ttl=60 time=0.145 ms
```

TSpider分布式数据库

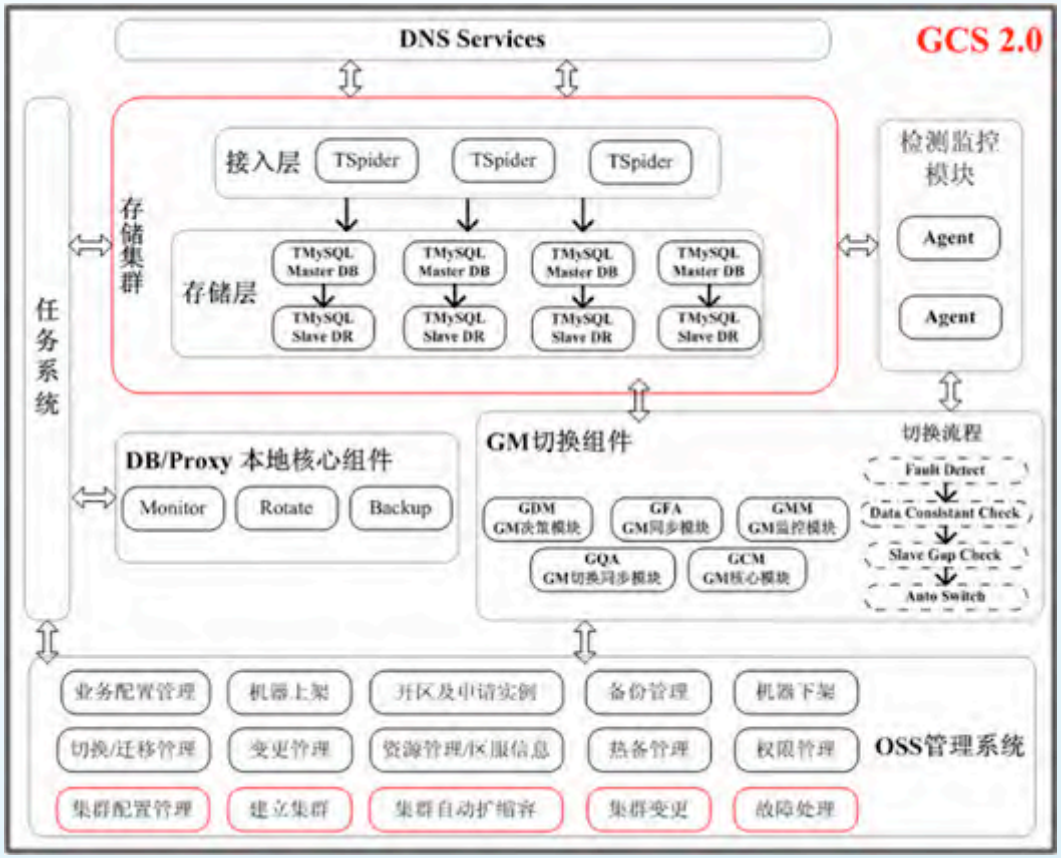
- SQL并行化
  - 程序伪代码

```
Serial mode()
{
    my partitions[] = prune
    spider_append_sql;
    foreach pt in(paritions
    {   // 每个分片串行执行que
        exec_query(pt->quer
        spider_db_store_res
    }
}
```

```
Parallel_mode()
{
    my partitions[] = prune_partitions(); // 算出query涉及哪些分区
    spider_append_sql;                    // 拼出待执行的sql语句
    foreach pt in(paritions)
    {   // 每个分片放到单独线程同时执行query并store result
        create_thread(spider_bg_conn_action);
    }
    // 线程执行逻辑
    spider_bg_conn_action(pt)
    {
        exec_query(pt->query)
        spider_db_store_result(pt->result_list)
    }
    // 合并处理各个分片执行的结果
    foreach pt in(paritions)    {
        process(pt->result_list);
    }
}
```

TSpider分布式数据库

- 游戏云存储新架构2.0



FAQ

团队技术博客

tencentdba.com

欢迎加入我们

felixliang@tencent.com

THANKS

SequeMedia 盛拓传媒 | ITPUB