大型移动项目MySQL数据库性能优化

张　翔

上海爱可生信息技术有限公司

MySQL高级技术顾问

项目背景

应用环境：

注册用户2亿，同时在线2000多万，活跃用户在230万

连续高压力下单项业务操作的所有数据库响应时间和<0.1s

高访问量压力时的故障快速恢复少于5分钟

每日大量用户LOG IN与LOG OUT（伴随庞大的信息查询）

每日大量的互联网消息通信（自然人与机器人并存）

MSSQL与MySQL同等压力下的极限性能对比

平台环境

数据库环境：

单机MySQL应用环境（目前仅Master提供对外数据库服务）

普通PC64位服务器，共享存储，32G内存，2路4核2.50HZ cpu

150GB的在线应用数据量，上TB的历史数据（统计经分等）

单表最大7亿条记录，最大表容量55G

平均1000个并发的Acitve连接

每秒处理5000个Active的数据库R，2000个Active的 数据库写W

每秒1030次磁盘读，19+M 磁盘数据读取量

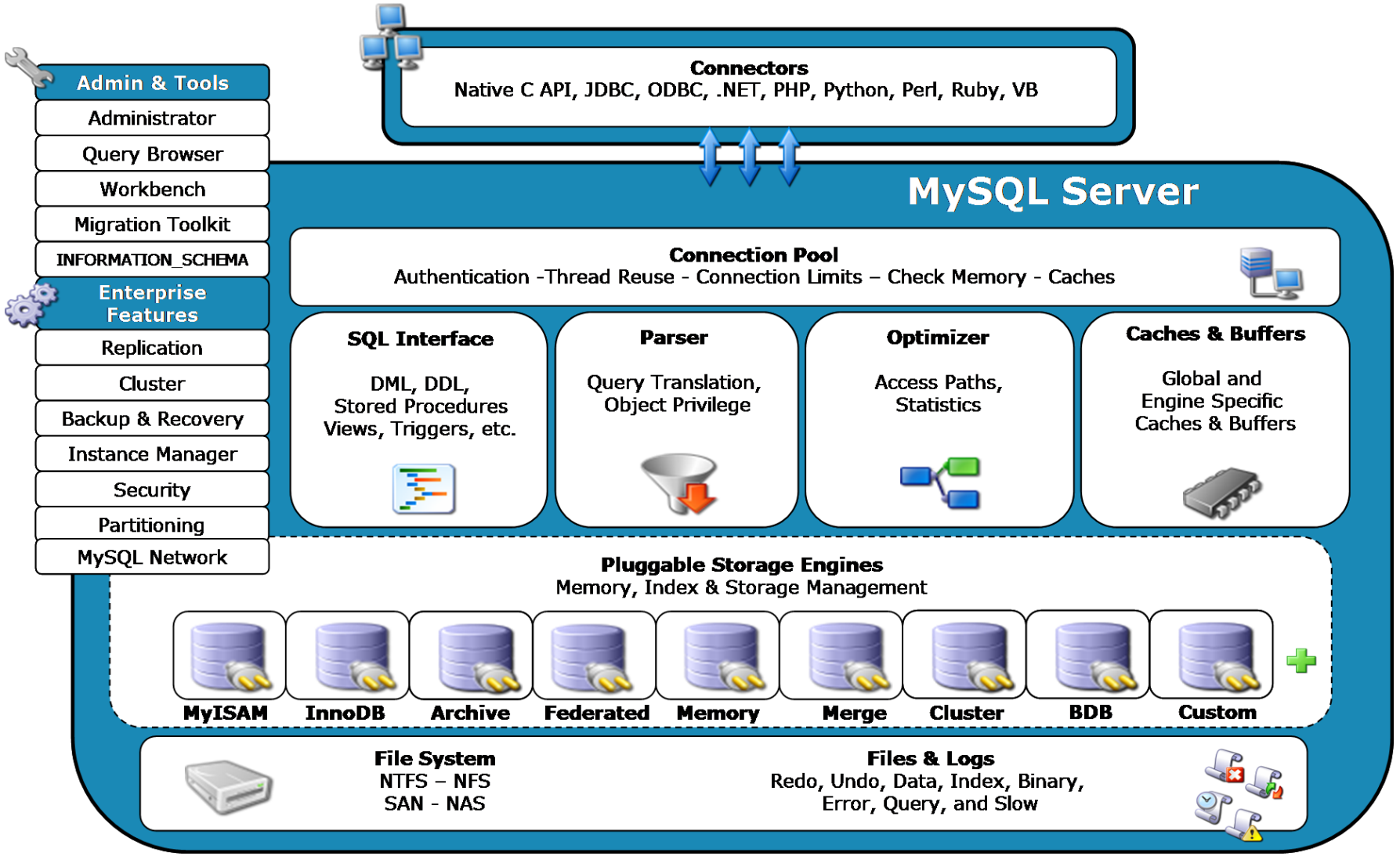
Agenda

- MySQL体系架构
- MySQL设计优化
- MySQL系统优化
- MySQL配置优化
- MySQL语句优化

Agenda

第一层 MySQL体系架构

MySQL体系结构


Admin & Tools
Administrator
Query Browser
Workbench
Migration Toolkit
INFORMATION_SCHEMA
Enterprise Features
Replication
Cluster
Backup & Recovery
Instance Manager
Security
Partitioning
MySQL Network
Connectors
Native C API, JDBC, ODBC, .NET, PHP, Python, Perl, Ruby, VB
MySQL Server
Connection Pool
Authentication -Thread Reuse - Connection Limits – Check Memory - Caches
SQL Interface
DML, DDL,
Stored Procedures
Views, Triggers, etc.
Parser
Query Translation,
Object Privilege
Optimizer
Access Paths,
Statistics
Caches & Buffers
Global and
Engine Specific
Caches & Buffers
Pluggable Storage Engines
Memory, Index & Storage Management
MyISAM
InnoDB
Archive
Federated
Memory
Merge
Cluster
BDB
Custom
File System
NTFS – NFS
SAN - NAS
Files & Logs
Redo, Undo, Data, Index, Binary,
Error, Query, and Slow

MySQL 存储特性

通过选择存储引擎来更好的适应应用的特殊性能要求

对你来说最重要的是什么?

- 密集读操作
- OLTP（联机事务处理）
- 事务处理
- 性能
- 可伸缩性
- 并发级别
- 索引类型
- 存储利用率
- 高可靠性
- 复制
- 在线备份
- 数据仓库
- 外键
- 占用空间小
- 行级别锁
- 嵌入式
- 表级别锁
- 集群

规划设计-引擎选择

MyISAM 引擎适用场景：

- 数据库端的并发数量不多（20%写 80%读）
- 读操作比较多，而且都能很好的用到索引
- SQL语句比较简单的应用
- 轻松达到TB级数据量存储的数据仓库

规划设计-引擎选择

InnoDB 引擎适用场景：

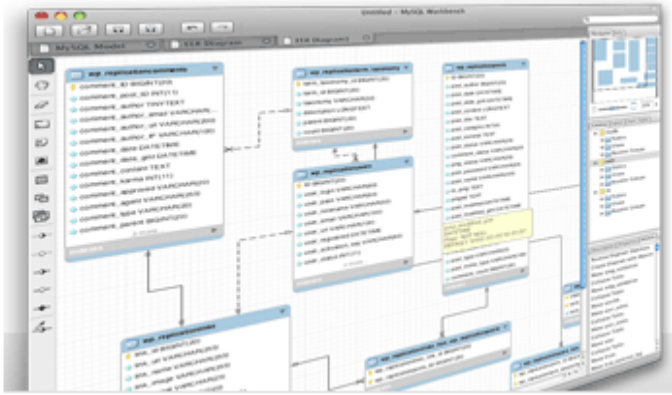
- 数据库端的读写并发数量非常多
- 写操作比较多，TB级数据量应用
- 数据较小、索引不好利用的应用比较多（报表）
- 有外键、事务等需求的应用

Agenda

第二层 MySQL设计优化

MySQL数据库结构----规划设计

- 命名规则
- 字段类型
- 编码选择
- 其他注意的问题



MySQL Workbench 5.1

Visual Database Design

Windows, Linux, Mac OS

规划设计-命名规则

- 按照多数开发语言的命名规则。比如(myCustomer)
- 按照多数开源思想命名规则。比如(my_customer)
- 按照咱们中国人的思想。比如(我的客户)
- 随便的命名。比如(my customer)

规划设计-字段类型

- 整型
- 浮点类型
- 时间日期类型
- 字符类型

TINYINT、INT、BIGINT

FLOAT、DOUBLE

DECIMAL、NUMERIC

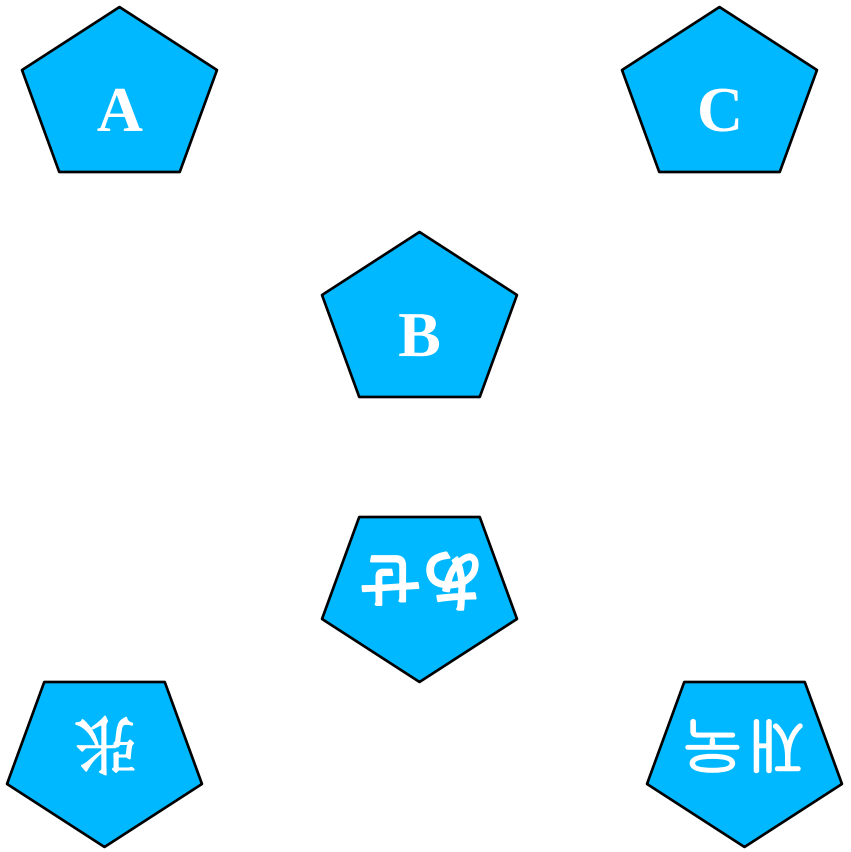
DATETIME、DATE、TIMESTAMP

VARCHAR、CHAR

BLOB、TEXT、ENUM

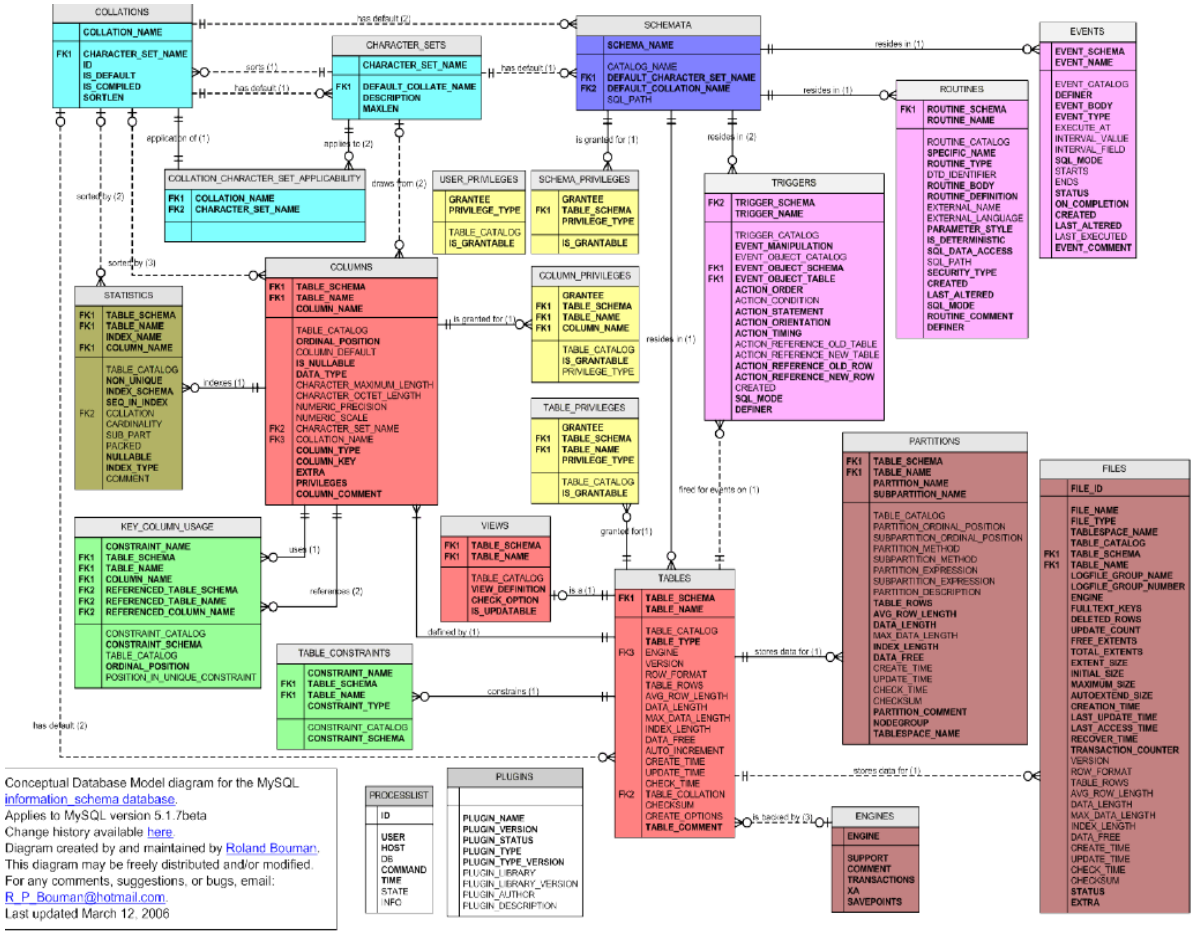
规划设计-编码选择

- 单字节？
  - 西欧文字
  - 中欧、南欧、中东、等等
  - latin1
- 多字节？
  - 中国、韩国、日本
  - utf-8
- 只考虑汉字？
  - 中国
  - gbk 、gb2312



规划设计-其他问题

- 默认值
- 索引
- 反范式



规划设计-命名规则

第三层 MySQL系统优化

MySQL数据库系统----选型优化

越好的机器性能越好？

文件系统越好性能越好？

网络越好性能越好？

版本越新性能越好？

MySQL数据库系统—硬件与系统环境

多核的CPU，主频高的CPU

更大的内存，MySQL是个很喜欢内存的数据库

共享磁盘, RAID阵列, ISCSI, NAS, 本地磁盘, SSD

使用合适的文件系统

- XFS
- ZFS
- NTFS
- EXT3

MySQL数据库系统—网络环境

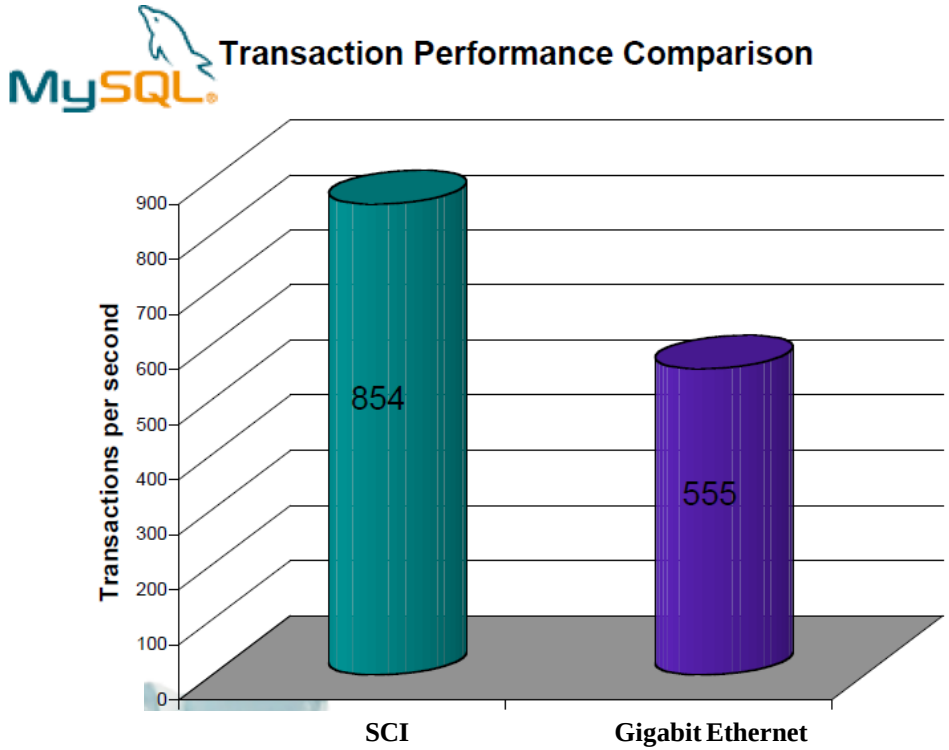
尽量将数据库整体系统部署在局域网内

使用专有的网络协议

- SCI
- 光缆

保证网络的安全冗余

- 双网线，提供安全冗余
- 0.0.0.0多端口绑定监听

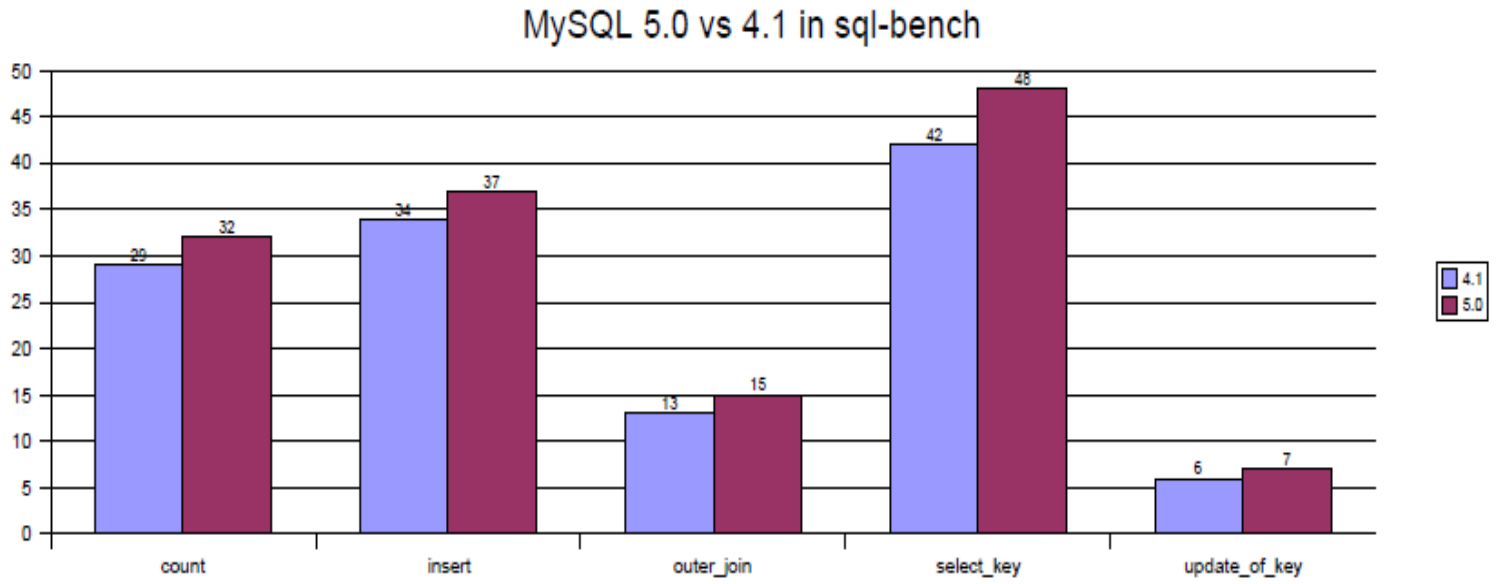


MySQL数据库系统—软件环境

开启MySQL复制，实现读、写分离，负载均衡

获得推荐的最新GA版本，利用BUG修复提升性能

利用分区新功能进行大数据的数据拆分，等等



规划设计-命名规则

第四层 MySQL配置优化

MySQL数据库配置----全局参数设置

key_buffer_size

MyISAM索引缓冲，根据（key reads / Key_read_requests ）判断

innodb_buffer_pool_size

InnoDB数据、索引、日志缓冲 最重要的引擎参数，根据（hit riatos和FILE I/O）判断

wait_time_out

线程连接的超时时间，尽量不要设置的很大

max_connections

允许服务器最大连接数，尽量不要设置很大

thread_concurrency

线程并发利用数量 （cpu+disk）*2， 根据（OS中显示的请求队列和tickets ）判断

注意：全局参数设置一经设置，随服务器启动预占用资源

MySQL数据库配置----线程参数设置

sort_buffer_size

获得更快的--ORDER BY,GROUP BY,SELECT DISTINCT,UNION DISTINCT

read_rnd_buffer_size

当根据键进行分类操作时获得更快的--ORDER BY

join_buffer_size

Join连接使用全表扫描连接的缓冲大小，根据（ Select_full_join ）判断

read_buffer_size

全表扫描时为查询预留的缓冲大小，根据（ Select_scan ）判断

tmp_table_size

临时内存表超出设置，转化为磁盘表，根据（ Created_tmp_disk_tables）判断

注意：线程参数设置的小影响性能，设置的大会导致服务器swap

InnoDB ---- 专有优化参数

innodb_log_file_size (默认5M)

记录InnoDB 引擎redo log 的文件

较大的值意味着较长的故障崩溃恢复时间

Innodb_flush_method (默认 fdatasync)

Linux系统可以使用O_DIRECT处理数据文件，避免OS级别的Cache

O_DIRECT模式提高数据文件和日志文件的IO提交性能

innodb_flush_log_at_trx_commit (默认1)

0 表示每秒进行进行一次A和B操作。

1 表示在每次事务提交后执行一次A和B操作。

2 表示在每次事务提交后，执行一次B操作。

A--LOG数据写到CACHE **B--FLUSH LOG 数据刷新到磁盘**

规划设计-SQL优化

第五层 MySQL语句优化

语句优化-读语句

性能差的读语句

```
CREATE TABLE `UserStatus_Log` (
  `LogTime` datetime NOT NULL,
  `UserId` int(11) NOT NULL,
  `MobileNo` bigint(20) DEFAULT NULL,
  `Sid` int(11) DEFAULT NULL,
  `OpType` tinyint(3) unsigned DEFAULT NULL,
  `RequestSource` smallint(6) DEFAULT NULL,
  KEY `IX_PS_UserStatusLog_UserId_LogTime` (`UserId`,`LogTime`),
  KEY `IX_PS_UserStatusLog_Sid` (`Sid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
mysql> select  count(*)  as  total  from UserStatus_Log where  1;
 +----------+
 | count(*)|
 +----------+
 | 524288  |
 +----------+
1 row in set (1.68 sec)

mysql>
```

InnoDB引擎　随记录越大执行越慢

语句优化-读语句

优化替代方法：

```
mysql> create table table_count
    -> ( table_name varchar(64) not null default '' primary key,
    ->   total bigint unsigned not null default 0
    -> ) engine myisam;
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> select count(*) as total from UserStatus_Log where 1 into @total;
Query OK, 1 rows affected (0.36 sec)

mysql>insert into table_count (table_name, total) values ('UserStatus_Log', @total);
Query OK, 1 rows affected (0.00 sec))
```

```
mysql>select * from table_count where table_name = 'UserStatus_Log';
+-----------------------+----------+
| table_name            |  total   |
+-----------------------+----------+
| UserStatus_Log        | 524288   |
+-----------------------+----------+
1 row in set (0.00 sec)
```

语句优化-查询分析器

执行性能差的SQL分析结果

```
mysql> explain
    -> select count(*)  as total  from  UserStatus_Log  where  1;

+----+-------------+--------------------+-------+-----------------+--------+---------+----------+---------+--------------+
| id | select_type | table              | type  | possible_keys   | key    | key_len | ref      | rows    | Extra        |
+----+-------------+--------------------+-------+-----------------+--------+---------+----------+---------+--------------+
|  1 | SIMPLE      | UserStatus_Log     | index | NULL            | idx_id | 4       | NULL     | 524288  | Using index  |
+----+-------------+--------------------+-------+-----------------+--------+---------+----------+---------+--------------+
1 row in set (0.01 sec)
```

虽然使用了索引，但是还是进行了全表扫描

语句优化-查询分析器

优化后的SQL执行性能分析

```
mysql> explain
    -> select table_name, total from table_count where table_name = 'UserStatus_Log';

+----+-------------+---------------+--------+---------------+------+---------+------+------+-------+
| id | select_type | table         | type   | possible_keys | key  | key_len | ref  | rows | Extra |
+----+-------------+---------------+--------+---------------+------+---------+------+------+-------+
|  1 | SIMPLE      | table_count   | system | primary       | NULL | NULL    | NULL |    1 |       |
+----+-------------+---------------+--------+---------------+------+---------+------+------+-------+
1 row in set (0.00 sec)
```

因为仅有一行，这行的列值被优化器认为是常数。最多一行匹配，如const表执行，速度非常快

MySQL Query Analyzer

利用GUI工具去捕捉有性能问题的SQL语句，提高优化效率

MyWeb1:13306 Browse Queries

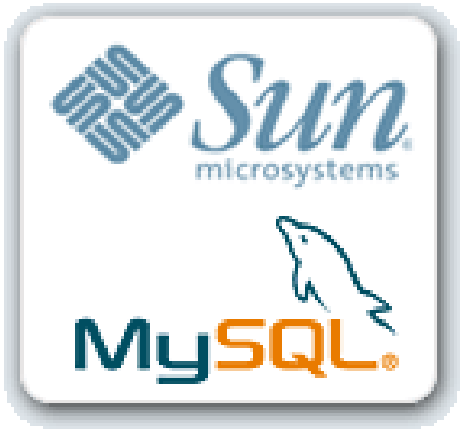
Search Type	Query Search	Database	Time Display	Hours
Contains	SELECT	sakila	Interval	02

Query	Database	Execution (hh:mm:ss.ms) Count	Total	Max
SELECT `p` . * , paymen...t_orders . last_order ;	sakila	47,541	3:43.281	0.531
SELECT `p` . * , paymen... = `p` . customer_id) ;	sakila	11,640	1:30.000	0.141
SELECT COUNT(*) FROM ...E pad > ? AND pad < ? ;	sakila	6,345	23.594	0.234
SELECT hibinstanc0_ . i...frequency IS NOT NULL)	sakila	6,133	23.422	0.156
SELECT continent , regi... , Region WITH ROLLUP ;	sakila	6,123	23.125	0.109
SELECT Country . Name ,...) < City . Population ;	sakila	5,278	1:10.094	0.234

MyWeb1:13306 Browse Queries

Search Type	Query Search	Database	Time Display	Hours
Contains	SELECT	sakila	Interval	02

Query	Database	Execution (hh:mm:ss.ms) Count	Total	Max
SELECT `p` . * , paymen...t_orders . last_order ;	sakila	47,401	1:25.828	0.016
SELECT `p` . * , paymen... = `p` . customer_id) ;	sakila	11,442	1:27.531	0.141
SELECT COUNT(*) FROM ...E pad > ? AND pad < ? ;	sakila	6,313	23.938	0.234
SELECT hibinstanc0_ . i...frequency IS NOT NULL)	sakila	6,278	23.156	0.156
SELECT continent , regi... , Region WITH ROLLUP ;	sakila	6,221	23.125	0.109
SELECT Country . Name ,...) < City . Population ;	sakila	5,553	1:11.875	0.203



谢谢！QA

技术服务热线　400-820-6580

在线服务网站　www.actionsky.com