



# Linux 下的MySQL调优

叶金荣

MySQL专家组核心成员

2008-1-9

[imysql@imysql.cn](mailto:imysql@imysql.cn)



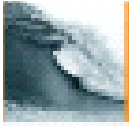
## 主要内容

- 需求：为什么要调优
- 分析：怎么找到软肋
- 实战：如何调优
- 总结



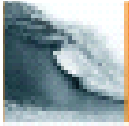
# 为什么要调优

- 老板要求
- 客户投诉
- 机器发飙
- 自己主动



# 机器发飙

- ✓ 网站或客户端打开非常慢，而webserver负载低，或打开静态页面很快，那就很可能是数据库的问题
- ✓ `load average >= 5`
- ✓ `lowait >= 10`
- ✓ `vmstat procs(r b)` 值较高
- ✓ `top`中CPU的idle很小，`sys`或`wait`较高
- ✓ 服务器的swap严重
- ✓ mysql的内存命中率很低，例如 `myisam_key_read_hit_ratio` 或 `innodb_buffer_hit_ratio` 较低



## 瓶颈定位

- **vmstat,iostat,top**等系统级别的工具
- **explain**
- **slow query**
- **show status/show processlist/show engine innodb status**
- 其他, 如**mysqlreport, profiling**



## 瓶颈定位 – vmstat/iostat

```
Linux 2.6.9-55.0.9.EL (01_10_10.11.17.201_000) 12/21/2008
```

```
avg-cpu:  %user   %nice    %sys %iowait  %idle
           12.76    0.00    4.56    8.77   73.90
```

```
Device:   rrqm/s  wrqm/s   r/s    w/s  rsec/s  wsec/s   rkB/s   wkB/s  avgrq-sz  avgqu-sz   await  svctm  %util
cciss/c0d0  0.19    1.80  44.65 618.89 1127.95  783.54  563.97  391.77    2.88     0.25    0.38   0.15  10.16
```

procs		-----memory-----				---swap--		-----io----		--system--		----cpu----			
r	b	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa
0	0	0	41	6	773	0	0	564	3592	15	1	13	5	74	9
1	1	0	46	6	768	0	0	211	15147	3451	4511	10	4	48	38
1	0	0	47	6	767	0	0	1754	3751	3940	5070	14	4	64	17
1	0	0	44	6	769	0	0	195	3246	4046	5767	10	3	76	10
<u>2</u>	<u>1</u>	0	41	6	773	<u>0</u>	<u>0</u>	<u>204</u>	<u>4179</u>	4349	6655	11	3	76	<u>9</u>



# 瓶颈定位 - **Explain**

## **Explain** 都能提供什么信息呢？

- ✓ 表的读取顺序
- ✓ 每个表都是如何读取的
- ✓ 可能用到哪些索引， 实际使用了哪些索引
- ✓ 表是如何引用的
- ✓ 查询优化器从每个表中预计读取的记录数
- ✓ 其他额外信息， 例如是否使用了内存表， 是否引发排序等



# 瓶颈定位 - Explain

## 1. 单表主键检索

```
mysql> EXPLAIN SELECT * FROM yejr WHERE id = 1010864466;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | yejr | const | PRIMARY | PRIMARY | 4 | const | 1 | 
```

## 2. 两个表主键左连接

```
mysql> EXPLAIN SELECT * FROM yejr LEFT JOIN yejr1 USING (id) WHERE id > 1010864466;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | yejr | range | PRIMARY | PRIMARY | 4 | NULL | 1752 | Using where |
| 1 | SIMPLE | yejr1 | eq_ref | PRIMARY | PRIMARY | 4 | yejr.yejr.id | 1 | 
```

## 3. 三个表主键左连接

```
mysql> EXPLAIN SELECT * FROM yejr LEFT JOIN yejr1 USING (id) LEFT JOIN yejr2 USING(id) WHERE id > 1010864466;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows | Extra |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | yejr | range | PRIMARY | PRIMARY | 4 | NULL | 1752 | Using where |
| 1 | SIMPLE | yejr1 | eq_ref | PRIMARY | PRIMARY | 4 | yejr.yejr.id | 1 |
| 1 | SIMPLE | yejr2 | eq_ref | PRIMARY | PRIMARY | 4 | yejr.yejr.id | 1 |
```





## 瓶颈定位 - 续

- ✓ 将 **LONG\_QUERY\_TIME** 设为最小值；建议打补丁，这样单位可以设成微秒，并可查看详细执行计划
- ✓ 执行 **SHOW [GLOBAL] STATUS/PROCESSLIST** 查看当前运行状态，从结果中发现可能的问题
- ✓ 执行 **SHOW ENGINE INNODB STATUS** 查看 **INNODB** 的状态
- ✓ 另外,要定期检查多余的索引以及没有使用索引的慢 查询
- ✓ 利用 **mysqlreport** 产生可读性更强的报告
- ✓ 利用 **Profiling** 剖析一次查询瓶颈所在
- ✓ 其他工具，包括 监控工具，**linux** 自带工具等



# MySQL调优的几种途径

- 硬件、网络、软件
- MySQL参数设置
- 应用程序、架构优化
- 查询优化、索引



## 硬件、网络、软件

- ✓ 通常硬件是优化的最佳入口，主要是CPU、内存、磁盘、网络
- ✓ 客户端和服务器的一个高速的局域网内
- ✓ 通常，新版本的效率不如旧版本，但是可以利用新版本的新功能来从另一方面得到性能上的提升
- ✓ 编译优化，采用静态编译等
- ✓ 使用更稳定高效的内核
- ✓ 使用合适的文件系统，推荐使用xfs([高级文件系统实现者指南](#))



# MySQL参数设置

参数名	说明
<b>Key Buffer</b>	MyISAM索引缓冲
<b>Query Cache</b>	查询结果缓存
<b>Sort Buffer</b>	排序缓冲
<b>Read Buffer</b>	全表扫描缓冲
<b>Join Buffer</b>	连接查询缓冲
<b>Slow Query</b>	设置慢查询,打上msl补丁
<b>Tmp Table</b>	内存表, 还需要注意max_heap_table_size
<b>Innodb Buffer</b>	InnoDB最重要的设置,包括日志缓冲



## 应用程序、架构优化

- ✓ 垂直/水平切分服务器/数据库、表
- ✓ 开启MySQL复制，实现读、写分离
- ✓ 在复制的基础上，增加负载均衡
- ✓ 采用集群+复制(MySQL 6.0+)
- ✓ 频繁更新的表，可以分离成父表和子表(内存表)
- ✓ 用统计表保存定时统计结果，而不是在大表上直接统计
- ✓ 编写存储过程/函数来代替大量的外部应用程序交互



## 查询优化、索引

- ✓ 确保索引合理利用，尽量使用 联合索引
- ✓ 适当加大查询缓存(query cache)
- ✓ 尽量减少交互次数
- ✓ 尽量使用固定格式的 **SQL** 语句，查询语句中少用运算或函数
- ✓ 缩短每个事务
- ✓ 使用适当的字段 类型；适当的长度，有需要的时候再扩充
- ✓ 分解复杂查询为 多个小查询
- ✓ 字符型字段采用前缀索引
- ✓ 其他 .....



# 调优方法

## 一、选择合适的引擎

### ■ MyISAM


这个是默认类型,基于传统的**ISAM**类型,它是存储记录和文件的标准方法。与其他存储引擎比较, **MyISAM**具有检查和修复表格的大多数工具。**MyISAM**表格可以被压缩,而且它们支持全文搜索。它们不是事务安全的,而且也不支持外键。

### ■ InnoDB

**ACID**、外键、日志修复。**InnoDB**表格速度很快。如果需要一个事务安全的存储引擎或者是需要大量并发的**INSERT**或**UPDATE**,应该使用**InnoDB**表。

### ■ NDB

支持事务,用于**cluster**,实现高可用,但性能仍欠佳。



## 调优方法 – 续(其他)

- ✓ 不直接执行 **COUNT(\*)** – innodb
- ✓ 多个操作放在一起提交，但要注意事务不能太大
- ✓ 日志文件并非越大越好，需要考虑恢复和检查点
- ✓ 左连接时把数据量小的表放在前面
- ✓ **innodb\_flush\_log\_at\_trx\_commit** 可以尝试设置为 **2**，甚至是 **0**
- ✓ 导入数据时关闭 **AUTOCOMMIT** 以及 **UNIQUE\_CHECKS**、**FOREIGN\_KEY\_CHECKS**
- ✓ 复杂的查询总是先用**EXPLAIN**来分析一下
- ✓ 定期执行**OPTIMIZE TABLE**整理碎片
- ✓ 用**char**来代替**varchar**，**MyISAM**是这样，**InnoDB**则相反





# 一起讨论！

参考&推荐：

**MySQL官方手册(英文在线)**

**高性能MySQL(第2版)**