

Linux 下的 MySQL 调优

2007.12.08

叶金荣

imysql@imysql.cn

<http://imysql.cn>
<http://www.chinaunix.net>

主要内容

- 1、调优概述
- 2、性能瓶颈定位
- 3、调优方法
- 4、其他 TIPS

调优概述

- * 网络、硬件、软件
- * 系统设置
- * 应用程序、架构优化
- * 查询优化、索引

网络、硬件、软件

- 通常硬件是优化的最佳入口，主要是 CPU、硬盘、内存
- 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

ext3 浪费过多的空间而且格式化比较慢

ReiserFS 挂载时间长，而且对于日常操作会产生比较多的页错误

JFS 是 CPU 占用率最低的

XFS 应该是用来作家用和小型商用文件服务器综合起来看最合适的文件系统

系统设置—全局设置

- External-locking # 禁用文件系统外部锁
- Table_cache # 缓存数据表数量
- Thread_cache # 线程缓存
- Back_log # 设定缓存的队列数, 节省连接时的开销
- Slow_query # 设置慢查询
- Query_cache # 查询缓存
- Read_buffer # 看 Handler_read_rnd_next/Com_select 比率
- read_rnd_buffer # 从排序结果中读取记录
- Tmp_table_heap_table # 内存、临时表

系统设置—全局设置(续)

- `Sort_buffer` # 看 `Sort_merge_passes` 值
- `Join_buffer` # 没使用索引 join(需要扫描全表)
- `Wait_timeout` # 超时
- `Skip-name-resolve` # 禁用 dns 解析
- `query_cache_min_res_unit` # 查询缓存分配的最小块的大小
- `query_prealloc_size` # 查询分析器预分配缓存
- `#log-bin` 开启 `log-bin` 会使性能下降 46%

架构、应用程序优化

- 垂直 / 水平切分服务器 / 数据表
- 开启 MySQL 复制，实现读、写分离
- 在复制的基础上，可以在多个从服务器间做到负载均衡
- 采用集群 + 复制
- 需要频繁更新的表，可以分离成母表和子表（内存表）
- 使用一个统计表来保存统计结果，而不是每次都在大表上做统计
- 编写存储过程 / 函数来代替大量的外部应用程序

查询优化、索引

详情请见如下文章中的相关部分

http://imysql.cn/2007_01_27_mysql_optimize_issue

性能瓶颈定位

- Explain
- Profiling
- slow query
- show processlist
- Mysqlreport
- Mysqlar
- Mysql administrator

性能瓶颈定位 — explain

Explain 都提供了些什么信息:

- 数据表的读取顺序
- 各个数据表都是如何读取的
- 使用到了哪些索引
- 数据表之间如何互相引用的
- 查询优化器从每个表中预计读取的记录数

性能瓶颈定位 — profiling

```
mysql> set profiling=1;
mysql> select * from tbl where name= 'user';
mysql> show profiles;
```

```
+-----+-----+
| Query_ID | Duration | Query |
+-----+-----+
.....
|          4 | 0.00039100 | select * from tbl where name = 'user' |
+-----+-----+
```

```
mysql> show profile for query 4;
```

```
+-----+-----+
| Status | Duration |
+-----+-----+
| (initialization) | 0.000005 |
| checking query cache for query | 0.0000660 |
| Opening tables | 0.000014 |
| System lock | 0.000008 |
```

性能瓶颈定位 — profiling(续)

Table lock	0.000028	
init	0.000041	
optimizing	0.000013	
statistics	0.000077	
preparing	0.000015	
executing	0.000005	
Sending data	0.000079	
end	0.000008	
query end	0.000005	
storing result in query cache	0.000005	
freeing items	0.000001	
closing tables	0.000007	
logging slow query	0.000005	
+-----+-----+		

性能瓶颈定位 — profiling(续)

SHOW PROFILES 语法

```
SHOW PROFILE [type [, type] ... ]  
    [FOR QUERY n]  
    [LIMIT n [OFFSET n]]
```

type:

- ALL
- BLOCK IO
- CONTEXT SWITCHES
- CPU
- IPC
- MEMORY 未实现
- PAGE FAULTS
- SOURCE
- SWAPS

性能瓶颈定位 — slow query/show processlist

- `long_query_time` 的单位是秒，默认是 10 秒，通常改成 1 秒
- 或者打个 patch，使得单位可以设定成微秒
- Slow log 文件可以随时清空
- 安装包自带分析工具：`mysqldumpslow`
- 其他工具 `mysql slow log parser`
- 执行 `show processlist` 查看当前运行的查询情况，主要查看运行时间以及状态，确保时间不会很长并且没有锁表等不良状态
- 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

性能瓶颈定位

— mysqlreport

```
3  _Key_-----
4 Buffer used    380.00k of 512.00M  %Used:   0.07
5   Current      59.32M           %Usage:  11.59
6 Write ratio    0.93
7 Read ratio     0.00

38 _ Query Cache-----
39 Memory usage  17.81M of  32.00M  %Used:  55.66
40 Block Fragmnt 13.05%
41 Hits          16.58k      8.02/s
42 Inserts       48.50k      23.48/s
43 Prunes        33.46k      16.20/s
44 Insrt:Prune   1.45:1      7.28/s
45 Hit:Insert    0.34:1

47 _ Table Locks-----
48 Waited        1.01k      0.49/s  %Total:   1.24
49 Immediate     80.04k     38.74/s
```

性能瓶颈定位

(续)

mysqlreport

```
51_ Tables-----
52 Open          107 of 1024    %Cache: 10.45
53 Opened        118    0.06/s

59 _ Created Temp-----
60 Disk table    10    0.00/s
61 Table         26    0.01/s
62 File          3    0.00/s

64 _ Threads-----
65 Running        55 of 77
66 Cache          0          %Hit: 0.5
67 Created        201    0.10/s
68 Slow           0    0.00/s
```

性能瓶颈定位 — mysqlar/mysql administrator

- mysqlar 需要和 rrdtool 配合，产生一个类似 cacti 那样的各种状态曲线图
- mysql administrator 是一个 gui 工具，windows 和 linux 的版本都有，简单易懂。主要是关注 “Health” 部分，还可以通过它来监控 mysql 复制是否正常工作等

调优方法 — 选择合适的存储引擎

- 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

支持 ACID 特性，还支持外键。InnoDB 表格速度很快。如果需要事务安全的存储引擎或者是需要大量的 INSERT 或 UPDATE，应该使用 InnoDB 表。

- NDB

支持事物，用于 mysql cluster，实现高可用。

调优方法 — 系统设置

下面的配置基本上能承受 1000qps 的压力 ,myisam+innodb

- `external-locking = FALSE`
- `back_log = 1024`
- `max_connections = 1200`
- `max_connect_errors = 1024`
- `open_files_limit = 4096`
- `max_allowed_packet = 24M`
- `read_rnd_buffer_size = 4M`
- `read_buffer_size = 4M`
- `join_buffer_size = 4M`
- `sort_buffer_size = 2M`
- `query_cache_limit = 2M`

调优方法 — 系统设置 (续)

- `query_cache_size = 400M`
- `query_cache_min_res_unit=2k`
- `thread_cache_size = 1200`
- `thread_concurrency = 4`
- `thread_stack = 128K`
- `tmp_table_size = 256M`
- `max_tmp_tables = 256`
- `bulk_insert_buffer_size = 4M`
- `binlog_cache_size = 2M`
- `max_binlog_size = 128M`
- `max_binlog_cache_size= 512M`
- `log-queries-not-using-indexes`
- `long_query_time = 1`

调优方法 — 系统设置 (续)

- `innodb_data_home_dir = /usr/local/mysql/data/`
- `innodb_data_file_path = ibdata1:10M:autoextend`
- `innodb_log_group_home_dir = /logs/mysql/` #data 和 log 目录分开
- `innodb_file_per_table= 1`
- `innodb_buffer_pool_size = 1300M`
- `innodb_log_file_size = 256M`
- `innodb_log_buffer_size = 16M`
- `innodb_lock_wait_timeout = 100`
- `innodb_flush_log_at_trx_commit = 2` # 设成 0, 大致会快 4.4 倍
- `innodb_flush_method = 'O_DIRECT'`
- `set-variable="transaction-isolation=READ-COMMITTED"`
- `innodb_file_io_threads = 4`
- `innodb_thread_concurrency = 4`
- `innodb_log_files_in_group = 3`
- `innodb_max_dirty_pages_pct = 90`

其他 TIPS — 针对 innodb

- 不直接执行 `select count(*) from table_name;`
- 多个操作放在一起提交，但要注意事务不能太大
- 日志文件越大越好，不过恢复时更慢
- 加大日志缓冲 `innodb_log_buffer_size`
- 能使用 `varchar` 则不使用 `char`，以节省空间
- `innodb_flush_log_at_trx_commit` 可以尝试设置为 0，甚至是 2
- 导入数据时关闭 `autocommit` 以及 `unique_checks`、`foreign_key_checks`
- 定期执行 `ALTER TABLE xx ENGINE=innodb` 整理碎片

其他 TIPS — 针对 myisam 及其他

- 酌情决定是否使用长连接
- 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
- 尽量使用 char 来代替 varchar ，前提是空间够大
- 每个字段都指定默认值，而不是 null
- 字段长度越短越好
- 查询条件中不使用函数
- 多用存储过程

Questions ?

Thank you!

imysql@imysql.cn
imysql@gmail.com
okyejr@gmail.com

<http://imysql.cn>