Mysql 数据库优化总结

----飞鸿无痕

说明：本文的环境为 CENTOS 5.5 64 Bit /Mysql 5.1.50

简介：使用 Mysql 有一段时间了，期间做了不少关于 Mysql 优化、设计、维护的工作，这两天有时间做一下简单的总结，方便自己回忆，同时也希望能对大家有点帮助.

I 硬件配置优化

- CPU 选择：多核的 CPU，主频高的 CPU
- 内存：更大的内存
- 磁盘选择：更快的转速、RAID、阵列卡，
- 网络环境选择：尽量部署在局域网、SCI、光缆、千兆网、双网线提供冗余、0.0.0.0 多端口绑定监听

II 操作系统级优化

- 使用 64 位的操作系统，更好的使用大内存。
- 设置 noatime

[zhangxy@dowload_server1 ~]$ cat /etc/fstab

```
LABEL=/          /          ext3    defaults,noatime        1 1
/dev/sda5       /data       xfs     defaults,noatime        1 2
```

- 优化内核参数

```
net.ipv4.tcp_keepalive_time=7200
net.ipv4.tcp_max_syn_backlog=1024
net.ipv4.tcp_syncookies=1
net.ipv4.tcp_tw_reuse = 1
net.ipv4.tcp_tw_recycle = 1
net.ipv4.neigh.default.gc_thresh3 = 2048
net.ipv4.neigh.default.gc_thresh2 = 1024
net.ipv4.neigh.default.gc_thresh1 = 256
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.forwarding = 1
net.ipv4.conf.default.proxy_arp = 0
net.ipv4.tcp_syncookies = 1
net.core.netdev_max_backlog = 2048
net.core.dev_weight = 64
```

```
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
net.ipv4.tcp_rfc1337 = 1
net.ipv4.tcp_sack = 0
net.ipv4.tcp_fin_timeout = 20
net.ipv4.tcp_keepalive_probes = 5
net.ipv4.tcp_max_orphans = 32768
net.core.optmem_max = 20480
net.core.rmem_default = 16777216
net.core.rmem_max = 16777216
net.core.wmem_default = 16777216
net.core.wmem_max = 16777216
net.core.somaxconn = 500
net.ipv4.tcp_orphan_retries = 1
net.ipv4.tcp_max_tw_buckets = 18000
net.ipv4.ip_forward = 0
net.ipv4.conf.default.proxy_arp = 0
net.ipv4.conf.all.rp_filter = 1
kernel.sysrq = 1
net.ipv4.conf.default.send_redirects = 1
net.ipv4.conf.all.send_redirects = 0
net.ipv4.ip_local_port_range = 5000    65000
kernel.shmmax = 167108864
vm.swappiness=0
```

- 加大文件描述符限制

  Vim /etc/security/limits.conf

  加上

```
*         soft    nofile  65535
*         hard    nofile  65535
```

- 文件系统选择 xfs

```
/dev/sda5       /data        xfs      defaults,noatime         1 2
```

III Mysql 设计优化

III.1 存储引擎的选择

- Myisam：数据库并发不大，读多写少,而且都能很好的用到索引，sql 语句比较简单的应用，TB 数据仓库
- Innodb：并发访问大，写操作比较多，有外键、事务等需求的应用，系统内

存较大。

III.2 命名规则

- 多数开发语言命名规则：比如 MyAdress
- 多数开源思想命名规则：my_address
- 避免随便命名

III.3 字段类型选择

字段类型的选择的一般原则：

- 根据需求选择合适的字段类型，在满足需求的情况下字段类型尽可能小。
- 只分配满足需求的最小字符数，不要太慷慨。

原因：更小的字段类型更小的字符数占用更少的内存，占用更少的磁盘空间，占用更少的磁盘 IO，以及占用更少的带宽。

III.3.1 整型：

见如下图:

类型	字节	最小值	最大值
		(带符号的/无符号的)	(带符号的/无符号的)
TINYINT	1	-128	127
		0	255
SMALLINT	2	-32768	32767
		0	65535
MEDIUMINT	3	-8388608	8388607
		0	16777215
INT	4	-2147483648	2147483647
		0	4294967295
BIGINT	8	-9223372036854775808	9223372036854775807
		0	18446744073709551615

根据满足需求的最小整数为选择原则，能用 INT 的就不要用 BIGINT。
用无符号 INT 存储 IP，而非 CHAR(15)。

III.3.2 浮点型：

类型	字节	精度类型	使用场景
FLOAT(M,D)	4	单精度	精度要求不高，数值比较小
DOUBLE(M,D) （REAL）	8	双精度	精度要求不高，数值比较大
DECIMAL(M,D)(NUMERIC）	M+2	自定义精度	精度要求很高的场景

III.3.3 时间类型

类型	取值范围	存储空间	零值表示法
DATE	1000-01-01~9999-12-31	3 字节	0000-00-00
TIME	-838:59:59~838:59:59	3 字节	00:00:00
DATETIME	1000-01-01 00:00:00~9999-12-31 23:59:59	8 字节	0000-00-00 00:00:00
TIMESTAMP	19700101000000~2037 年的某个时刻	4 字节	00000000000000
YEAR	YEAR(4):1901~2155 YEAR(2)：1970~2069	1 字节	0000

III.3.4 字符类型

类型	最大长度	占用存储空间
CHAR[(M)]	M 字节	M 字节
VARCHAR[(M)]	M 字节	M+1 字节
TINYBLOD，TINYTEXT	2^8-1 字节	L+1 字节
BLOB，TEXT	2^16-1 字节	L+2
MEDIUMBLOB，MEDIUMTEXT	2^24-1 字节	L+3
LONGBLOB，LONGTEXT	2^32-1 字节	L+4
ENUM('value1','value2',...)	65535 个成员	1 或 2 字节
SET('value1','value2',...)	64 个成员	1,2,3,4 或 8 字节

注：L 表示可变长度的意思

对于 varchar 和 char 的选择要根据引擎和具体情况的不同来选择，主要依据如下原则:

1. 如果列数据项的大小一致或者相差不大，则使用 char。
2. 如果列数据项的大小差异相当大，则使用 varchar。

3. 对于 MyISAM 表，尽量使用 Char，对于那些经常需要修改而容易形成碎片的 myisam 和 isam 数据表就更是如此，它的缺点就是占用磁盘空间。
4. 对于 InnoDB 表，因为它的数据行内部存储格式对固定长度的数据行和可变长度的数据行不加区分（所有数据行共用一个表头部分，这个标头部分存放着指向各有关数据列的指针），所以使用 char 类型不见得会比使用 varchar 类型好。事实上，因为 char 类型通常要比 varchar 类型占用更多的空 间，所以从减少空间占用量和减少磁盘 i/o 的角度，使用 varchar 类型反而更有利。
5. 表中只要存在一个 varchar 类型的字段，那么所有的 char 字段都会自动变成 varchar 类型，因此建议定长和变长的数据分开。

III.4 编码选择

单字节 latin1
多字节 utf8(汉字占 3 个字节，英文字母占用一个字节)
如果含有中文字符的话最好都统一采用 utf8 类型，避免乱码的情况发生。

III.5 主键选择原则

注：这里说的主键设计主要是针对 INNODB 引擎
1. 能唯一的表示行。
2. 显式的定义一个数值类型自增字段的主键，这个字段可以仅用于做主键，不做其他用途。
3. MySQL 主键应该是单列的，以便提高连接和筛选操作的效率。
4. 主键字段类型尽可能小，能用 SMALLINT 就不用 INT，能用 INT 就不用 BIGINT。
5. 尽量保证不对主键字段进行更新修改，防止主键字段发生变化，引发数据存储碎片，降低 IO 性能。
6. MySQL 主键不应包含动态变化的数据，如时间戳、创建时间列、修改时间列等。
7. MySQL 主键应当有计算机自动生成。
8. 主键字段放在数据表的第一顺序。

推荐采用数值类型做主键并采用 auto_increment 属性让其自动增长。

III.6 其他需要注意的地方

- NULL OR NOT NULL
  尽可能设置每个字段为 NOT NULL，除非有特殊的需求，原因如下：

1. 使用含有 NULL 列做索引的话会占用更多的磁盘空间，因为索引 NULL 列需要而外的空间来保存。
2. 进行比较的时候，程序会更复杂。
3. 含有 NULL 的列比较特殊，SQL 难优化，如果是一个组合索引，那么这个 NULL 类型的字段会极大影响整个索引的效率。

- 索引

  索引的缺点：极大地加速了查询，减少扫描和锁定的数据行数。
  索引的缺点：占用磁盘空间，减慢了数据更新速度，增加了磁盘 IO。
  添加索引有如下原则：
  1. 选择唯一性索引。
  2. 为经常需要排序、分组和联合操作的字段建立索引。
  3. 为常作为查询条件的字段建立索引。
  4. 限制索引的数据，索引不是越多越好。
  5. 尽量使用数据量少的索引，对于大字段可以考虑前缀索引。
  6. 删除不再使用或者很少使用的索引。
  7. 结合核心 SQL 优先考虑覆盖索引。
  8. 忌用字符串做主键。

- 反范式设计

  适当的使用冗余的反范式设计，以空间换时间有的时候会很高效。

IV Mysql 软件优化

- 开启 mysql 复制，实现读写分离、负载均衡，将读的负载分摊到多个从服务器上，提高服务器的处理能力。
- 使用推荐的 GA 版本，提升性能
- 利用分区新功能进行大数据的数据拆分

V Mysql 配置优化

注意：全局参数一经设置，随服务器启动预占用资源。

- key_buffer_size 参数

  mysql 索引缓冲，如果是采用 myisam 的话要重点设置这个参数，根据（key_reads/key_read_requests）判断

- innodb_buffer_pool_size 参数

  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 参数
  线程连接的超时时间，尽量不要设置很大，推荐 10s
- max_connections 参数
  服务器允许的最大连接数，尽量不要设置太大，因为设置太大的话容易导致内存溢出，需要通过如下公式来确定：

```
SET @k_bytes = 1024;
SET @m_bytes = @k_bytes * 1024;
SET @g_bytes = @m_bytes * 1024;
SELECT
      (
 @@key_buffer_size + @@query_cache_size + @@tmp_table_size+
      @@innodb_buffer_pool_size + @@innodb_additional_mem_pool_size+
      @@innodb_log_buffer_size+
      @@max_connections *

 ( @@read_buffer_size + @@read_rnd_buffer_size + @@sort_buffer_size+
         @@join_buffer_size + @@binlog_cache_size + @@thread_stack
        ) )
/ @g_bytes AS MAX_MEMORY_USED_GB;
```

- thread_concurrency 参数
  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
- sort_buffer_size 参数
  获得更快的--ORDER BY,GROUP BY,SELECT DISTINCT,UNION DISTINCT
- read_rnd_buffer_size 参数
  当根据键进行分类操作时获得更快的--ORDER BY
- join_buffer_size 参数
  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 参数
  全表扫描时为查询预留的缓冲大小，根据 select_scan 判断
- tmp_table_size 参数
  临时内存表的设置，如果超过设置就会转化成磁盘表，根据参数(created_tmp_disk_tables)判断
- innodb_log_file_size 参数(默认 5M)
  记录 INNODB 引擎的 redo log 文件，设置较大的值意味着较长的恢复时间。
- innodb_flush_method 参数(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)
  1. 0 表示每秒进行一次 log 写入 cache，并 flush log 到磁盘。
  2. 1 表示在每次事务提交后执行 log 写入 cache，并 flush log 到磁盘。
  3. 2 表示在每次事务提交后，执行 log 数据写入到 cache，每秒执行一次 flush log 到磁盘。

VI Mysql 语句级优化

1. 性能查的读语句，在 innodb 中统计行数,建议另外弄一张统计表，采用 myisam，定期做统计.一般的对统计的数据不会要求太精准的情况下适用。
2. 尽量不要在数据库中做运算。
3. 避免负向查询和%前缀模糊查询。
4. 不在索引列做运算或者使用函数。
5. 不要在生产环境程序中使用 select * from 的形式查询数据。只查询需要使用的列。
6. 查询尽可能使用 limit 减少返回的行数，减少数据传输时间和带宽浪费。
7. where 子句尽可能对查询列使用函数，因为对查询列使用函数用不到索引。
8. 避免隐式类型转换，例如字符型一定要用’’，数字型一定不要使用’’。
9. 所有的 SQL 关键词用大写，养成良好的习惯，避免 SQL 语句重复编译造成系统资源的浪费。
10. 联表查询的时候，记得把小结果集放在前面，遵循小结构及驱动大结果集的原则。
11. 开启慢查询，定期用 explain 优化慢查询中的 SQL 语句。