April

http://blog.sina.com.cn/alwaysabeginner [□□] [□□□□]

个人资料

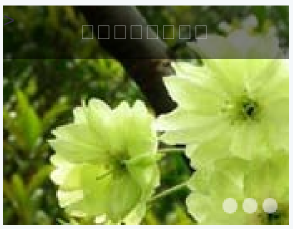
April

Qing □□

加好友 发纸条 写留言 加关注

□□□□□
□□□□□125
□□□□□4,171
□□□□□0

□□□□



□□□□□□□
□□Qing

□□□□□□□
□□Qing

正文 □□□□□大 中 小

MYSQL 查询缓存 (2011-04-12 15:54:36)

□□□ □□ □□□ MYSQL

MySQL 查询缓存保留了查询返回给客户端的完整结果。当缓存命中的时候，服务器马上返回保存的结果，并跳过解析、优化和执行步骤. 缓存也需要开销，只有在节省的资源大于开销的时候，缓存才是真正有效率的，这和服务器的负载相关。

为了保证 Query Cache 中的内容与是实际数据绝对一致，当表中的数据有任何变化，包括新增，修改，删除等，都会使所有引用到该表的 SQL 的 Query Cache 失效。

```
mysql>  SHOW VARIABLES LIKE '%query_cache%';
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| have_query_cache             | YES     |
| query_cache_limit            | 1048576 |
| query_cache_min_res_unit     | 4096    |
| query_cache_size             | 9437184 |
| query_cache_type             | ON      |
| query_cache_wlock_invalidate | OFF     |
+------------------------------+---------+
6 rows in set (0.00 sec)
```

各项参数的意义

query_cache_limit：允许 Cache 的单条 Query 结果集的最大容量，默认是1MB，超过此参数设置的 Query 结果集将不会被 Cache

query_cache_min_res_unit：设置 Query Cache 中每次分配内存的最小空间大小，也就是每个 Query 的 Cache 最小占用的内存空间大小

query_cache_size：设置 Query Cache 所使用的内存大小，默认值为0，大小必须是1024的整数倍，如果不是整数倍，MySQL 会自动调整降低最小量以达到1024的倍数

query_cache_type：控制 Query Cache 功能的开关，可以设置为0(OFF),1(ON)和2(DEMAND)三种，意义分别如下：

0(OFF)：关闭 Query Cache 功能，任何情况下都不会使用 Query Cache

1(ON)：开启 Query Cache 功能，但是当 SELECT 语句中使用的 SQL_NO_CACHE 提示后，将不使用Query Cache

2(DEMAND)：开启 Query Cache 功能，但是只有当 SELECT 语句中使用了 SQL_CACHE 提示后，才使用 Query Cache

query_cache_wlock_invalidate：控制当有写锁定发生在表上的时刻是否先失效该表相关的 Query Cache，如果设置为 1(TRUE)，则在写锁定的同时将失效该表相关的所有 Query Cache，如果设置为0(FALSE)则在锁定时刻仍然允许读取该表相关的 Query Cache。

修改参数

缓存大结果没有太大的益处。可以通过降低query_cache_limit 的值阻止缓存大结果，它有时有助于在碎片和在缓存中保存结果的开销中得到平衡。

```
mysql> set global query_cache_limit=1024;
Query OK, 0 rows affected (0.00 sec)

mysql>  SHOW VARIABLES LIKE '%query_cache%';
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| have_query_cache             | YES     |
| query_cache_limit            | 1024    |
| query_cache_min_res_unit     | 4096    |
| query_cache_size             | 9437184 |
| query_cache_type             | ON      |
| query_cache_wlock_invalidate | OFF     |
+------------------------------+---------+
6 rows in set (0.00 sec)
```

MySQL 检查缓存命中的规则

1、在检查缓存的时候，MySQL 不会对语句进行解析、正则化或者参数化，它精确地使用客户端传来的查询语句和其他数据。只要字符大小写、空格或者注释有一点点不同，查询缓存就认为这是一个不同的查询

2、查询缓存不会存储有不确定结果的查询。因此，任何一个包含不确定函数（比如NOW()或CURRENT_DATE()）的查询不会被缓存。同样地，CURRENT_USER()或CONNECTION_ID()这些由不同用户执行，将会产生不同的结果的查询也不会被缓存。事实上，查询缓存不会缓存引用了用户自定义函数、存储函数、用户自定义变量、临时表、mysql 数据库中的表或者任何一个有列级权限的表的查询

开启查询缓存的开销

读取查询在开始之前必须要检查缓存。

如果查询是可以被缓存的，但是不在缓存中，那么在产生结果之后进行保存会带来一些额外的开销。

写入数据的查询也会有额外的开销，因为它必须使缓存中相关的数据表失效。

这些开销相对来说较小，所以查询缓存还是很有好处的。但是，稍后你会看到，额外的开销有可能也会增加。

从缓存中受益最多的查询可能是需要很多资源来产生结果，但是不需要很多空间来保存的类型。所以用于存储、返回和失效的代价都较小。聚集查询，比如从大表中利用COUNT()产生较小的结果，就符合这个范畴。

检查查询缓存使用情况

检查是否从查询缓存中受益的最简单的办法就是检查缓存命中率

当服务器收到SELECT 语句的时候，Qcache_hits 和Com_select 这两个变量会根据查询缓存的情况进行递增

查询缓存命中率的计算公式是：Qcache_hits/(Qcache_hits + Com_select)。

```
mysql> show status like '%Com_select%';
```

□□□

- □□□□
  □□□
- 20130113□□□□□□□□
  □□□
- □□□□□□“□□□”□□□□
  □□□□□□□□

□□□□>>

□□□□□□□□

□□□...

```
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Com_select    | 1     |
+---------------+-------+
1 row in set (0.00 sec)

mysql> show status like '%Qcache%';
+-------------------------+---------+
| Variable_name           | Value   |
+-------------------------+---------+
| Qcache_free_blocks      | 1       |
| Qcache_free_memory      | 9418152 |
| Qcache_hits             | 3       |
| Qcache_inserts          | 9       |
| Qcache_lowmem_prunes    | 0       |
| Qcache_not_cached       | 10      |
| Qcache_queries_in_cache | 9       |
| Qcache_total_blocks     | 21      |
+-------------------------+---------+
8 rows in set (0.00 sec)
```

此时的查询缓存命中率：3/（3+1）=75%；由于个人的测试数据库，查询较少，更行更少，命中率颇高。嘿嘿~~

MySQL 提供了一系列的 Global Status 来记录 Query Cache 的当前状态，具体如下：

Qcache_free_blocks：目前还处于空闲状态的 Query Cache 中内存 Block 数目

Qcache_free_memory：目前还处于空闲状态的 Query Cache 内存总量

Qcache_hits：Query Cache 命中次数

Qcache_inserts：向 Query Cache 中插入新的 Query Cache 的次数，也就是没有命中的次数

Qcache_lowmem_prunes：当 Query Cache 内存容量不够，需要从中删除老的 Query Cache 以给新的 Cache 对象使用的次数

Qcache_not_cached：没有被 Cache 的 SQL 数，包括无法被 Cache 的 SQL 以及由于 query_cache_type 设置的不会被 Cache 的 SQL

Qcache_queries_in_cache：目前在 Query Cache 中的 SQL 数量

Qcache_total_blocks：Query Cache 中总的 Block 数量

分析和调整查询缓存的一个图示

通用查询缓存优化方案

使用多个较小的表，而不是用一个大表，对查询缓存有帮助。这种设计方式使失效策略工作在一个较好的颗粒度上。但是不要让这个想法过度影响架构设计，因为其他的因素能轻易抵消它的好处。

成批地进行写入操作，而不是逐个执行，会有效率得多。因为这种方法只会引起一次失效操作。

我们已经注意到在让缓存失效或清理一个大型缓存的时候，服务器可能会挂起相当长时间。至少在MySQL5.1之前的版本中是这样。一个容易的解决办法就是不要让query_cache_size 太大，256MB 已经太大了。

不能在数据库或表的基础上控制查询缓存，但是可以使用SQL_CACHE 和SQL_NO_CACHE 决定是否缓存查询。也可以基于某个连接来运行或禁止缓存，可以通过用适当的值设定query_cache_size 来开启或关闭对某个连接的缓存。

对于很多写入任务的应用程序，关闭查询缓存也许能改进性能。这样做可以消除缓存那些很快就会失效的查询所带来的开销。要记住在禁用的时候需要把query_cache_size 设置到0，这样就不会消耗任何内存。

如果想让大多数查询都不使用缓存，但是有少部分查询能从缓存中极大地受益，这时可以将全局变量query_cache_type 设置为DEMAND，然后在想使用缓存的查询后面添加SQL_CACHE。尽管这会造成更多的工作，但是可以细粒度地控制缓存。相应地，如果想缓存大部分查询，只排除其中一小部分，就可以使用SQL_NO_CACHE

□□□□RMAN□□□□□□□□□

□□□□mysql□□□□1064□□

□□ [□□□]

□□□□□□□□□...

□□□

□□□□□ □□□□□□□□□□

□□□□ □□□□□□□□□□ □□□□□

□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

< □□□

RMAN□□□□□□□□

□□□ >

mysql□□□□1064□□