**1、简介：**

查询缓存存储SELECT查询的文本以及发送给客户端的相应结果。如果随后收到一个相同的查询，服务器从查询缓存中重新得到查询结果，而不再需要解析和执行查询。如果你有一个不经常改变的表并且服务器收到该表的大量相同查询，查询缓存在这样的应用环境中十分有用。对于许多Web服务器来说存在这种典型情况，它根据数据库内容生成大量的动态页面。

**注**释：查询缓存不返回旧的数据。当表更改后，查询缓存值的相关条目被清空。

**注**释：如果你有许多**mysqld**服务器更新相同的MyISAM表，在这种情况下查询缓存不起作用。

**注**释：查询缓存不适用于服务器方编写的语句。如果正在使用服务器方编写的语句，要考虑到这些语句将不会应用查询缓存。参见 [25.2.4节，“C API预处理语句”](http://dev.mysql.com/doc/refman/5.1/zh/apis.html#c-api-prepared-statements)。

**2、性能**

下面是查询缓存的一些性能数据。这些结果是在Linux Alpha 2 x 500MHz系统（2GB RAM，64MB查询缓存）上运行MySQL基准组件产生的。

·         如果执行的所有查询是简单的(如从只有一行数据的表中选取一行)，但查询是不同的，查询不能被缓存，查询缓存激活率是13%。这可以看作是最坏的情形。在实际应用中，查询要复杂得多，因此，查询缓存使用率一般会很低。

·         从只有一行的表中查找一行数据时，使用查询缓存比不使用速度快238%。这可以看作查询使用缓存时速度提高最小的情况。

服务器启动时要禁用查询缓存，设置query\_cache\_size系统变量为0。禁用查询缓存代码后，没有明显的速度提高。编译MySQL时，通过在**configure中**使用--without-query-cache选项，可以从服务器中彻底去除查询缓存能力。

1. **3、工作原理**

本节描述查询缓存的工作原理。

下面的两个查询被查询缓存认为是不相同的：

SELECT \* FROM *tbl\_name*

Select \* from *tbl\_name*

查询必须是完全相同的(逐字节相同)才能够被认为是相同的。另外，同样的查询字符串由于其它原因可能认为是不同的。使用不同的数据库、不同的协议版本或者不同 默认字符集的查询被认为是不同的查询并且分别进行缓存。

从查询缓存中提取一个查询之前，MySQL检查用户对所有相关数据库和表的SELECT权限。如果没有权限，不使用缓存结果。

如果从查询缓存中返回一个查询结果，服务器把Qcache\_hits状态变量的值加一，而不是Com\_select变量。

如果一个表被更改了，那么使用那个表的所有缓冲查询将不再有效，并且从缓冲区中移出。这包括那些映射到改变了的表的使用MERGE表的查询。一个表可以被许多类型的语句更改，例如INSERT、UPDATE、DELETE、TRUNCATE、ALTER TABLE、DROP TABLE或DROP DATABASE。

COMMIT执行完后，被更改的事务InnoDB表不再有效。

使用InnoDB表时，查询缓存也在事务中工作，使用该表的版本号来检测其内容是否仍旧是当前的。

在MySQL 5.1中，视图产生的查询被缓存。

SELECT SQL\_CALC\_FOUND\_ROWS ...和SELECT FOUND\_ROWS() type类型的查询使用查询缓存。即使因创建的行数也被保存在缓冲区内，前面的查询从缓存中提取，FOUND\_ROWS()也返回正确的值。

如果一个查询包含下面函数中的任何一个，它不会被缓存：

|  |  |  |
| --- | --- | --- |
| BENCHMARK() | CONNECTION\_ID() | CURDATE() |
| CURRENT\_DATE() | CURRENT\_TIME() | CURRENT\_TIMESTAMP() |
| CURTIME() | DATABASE() | 带一个参数的ENCRYPT() |
| FOUND\_ROWS() | GET\_LOCK() | LAST\_INSERT\_ID() |
| LOAD\_FILE() | MASTER\_POS\_WAIT() | NOW() |
| RAND() | RELEASE\_LOCK() | SYSDATE() |
| 不带参数的UNIX\_TIMESTAMP() | USER() |  |

在下面的这些条件下，查询也不会被缓存：

·         引用自定义函数(UDFs)。

·         引用自定义变量。

·         引用mysql系统数据库中的表。

·         下面方式中的任何一种：

SELECT ...IN SHARE MODE

SELECT ...FOR UPDATE

SELECT ...INTO OUTFILE ...

SELECT ...INTO DUMPFILE ...

SELECT \* FROM ...WHERE autoincrement\_col IS NULL

最后一种方式不能被缓存是因为它被用作为ODBC工作区来获取最近插入的ID值。

被作为编写好的语句，即使没有使用占位符。例如，下面使用的查询：

char \*my\_sql\_stmt = "SELECT a，b FROM table\_c";

   /\* ...\*/

mysql\_stmt\_prepare(stmt，my\_sql\_stmt，strlen(my\_sql\_stmt));

不被缓存。

·         使用TEMPORARY表。

·         不使用任何表。

·         用户有某个表的列级权限。

1. **4、如何使用查询缓存**

可以在SELECT语句中指定查询缓存相关选项：

SQL\_CACHE：如果query\_cache\_type系统变量的值是ON或DEMAND，查询结果被缓存。

SQL\_NO\_CACHE：查询结果不被缓存。

示例：

SELECT SQL\_CACHE id, name FROM customer;

SELECT SQL\_NO\_CACHE id, name FROM customer;

1. **5、查看查询高速缓冲是否可用**

mysql> **SHOW VARIABLES LIKE 'have\_query\_cache';**

+------------------+-------+

| Variable\_name    | Value |

+------------------+-------+

| have\_query\_cache | YES   |

+------------------+-------+

即使禁用查询缓存，当使用标准 MySQL二进制时，这个值总是YES。

其它几个系统变量控制查询缓存操作。当启动**mysqld**时，这些变量可以在选项文件或者命令行中设置。所有查询缓存系统变量名以query\_cache\_ 开头。

**6、设置查询缓冲区大小**

mysql> **SET GLOBAL query\_cache\_size = 41984;**

mysql> **SHOW VARIABLES LIKE 'query\_cache\_size';**

+------------------+-------+

| Variable\_name    | Value |

+------------------+-------+

| query\_cache\_size | 41984 |

+------------------+-------+

为了设置查询缓存大小，设置query\_cache\_size系统变量。设置为0表示禁用查询缓存。 默认缓存大小设置为0；也就是禁用查询缓存。

当设置query\_cache\_size变量为非零值时，应记住查询缓存至少大约需要40KB来分配其数据结构。(具体大小取决于系统结构）。如果你把该值设置的太小，将会得到一个警告，如本例所示：

mysql> **SET GLOBAL query\_cache\_size = 40000;**

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> **SHOW WARNINGS\G**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

  Level: Warning

   Code: 1282

Message: Query cache failed to set size 39936; new query cache size is 0

mysql> **SET GLOBAL query\_cache\_size = 41984;**

Query OK, 0 rows affected (0.00 sec)

mysql> **SHOW VARIABLES LIKE 'query\_cache\_size';**

+------------------+-------+

| Variable\_name    | Value |

+------------------+-------+

| query\_cache\_size | 41984 |

+------------------+-------+

**7、query\_cache\_type的设置**

mysql> **SET SESSION query\_cache\_type = ON;**

如果查询缓存大小设置为大于0，query\_cache\_type变量影响其工作方式。这个变量可以设置为下面的值：

0或OFF：将阻止缓存或查询缓存结果。

1或ON：将允许缓存，以SELECT SQL\_NO\_CACHE开始的查询语句除外。

2或DEMAND：仅对以SELECT SQL\_CACHE开始的那些查询语句启用缓存。

设置query\_cache\_type变量的GLOBAL值将决定更改后所有连接客户端的缓存行为。具体客户端可以通过设置query\_cache\_type变量的会话值控制它们本身连接的缓存行为。

例如，一个客户可以禁用自己的查询缓存，方法如下：

mysql> **SET SESSION query\_cache\_type = OFF;**

**8、设置缓存结果的最大值最小值**

**SET GLOBAL query\_cache\_limit=10485760;#10M**

**SET GLOBAL query\_cache\_min\_res\_unit=41984;**

要控制可以被缓存的具体查询结果的最大值，应设置query\_cache\_limit变量。 默认值是1MB。

当一个查询结果（返回给客户端的数据）从查询缓冲中提取期间，它在查询缓存中排序。因此，数据通常不在大的数据块中处理。查询缓存根据数据排序要求分配数据块，因此，当一个数据块用完后分配一个新的数据块。因为内存分配操作是昂贵的(费时的)，所以通过query\_cache\_min\_res\_unit系统变量给查询缓存分配最小值。当查询执行时，最新的结果数据块根据实际数据大小来确定，因此可以释放不使用的内存。根据你的服务器执行查询的类型，你会发现调整query\_cache\_min\_res\_unit变量的值是有用的：

·         query\_cache\_min\_res\_unit默认值是4KB。这应该适合大部分情况。

·         如果你有大量返回小结果数据的查询，默认数据块大小可能会导致内存碎片，显示为大量空闲内存块。由于缺少内存，内存碎片会强制查询缓存从缓存内存中修整（删除）查询。这时，你应该减少query\_cache\_min\_res\_unit变量的值。空闲块和由于修整而移出的查询的数量通过Qcache\_free\_blocks和Qcache\_lowmem\_prunes变量的值给出。

·          如果大量查询返回大结果（检查 Qcache\_total\_blocks和Qcache\_queries\_in\_cache状态变量），你可以通过增加query\_cache\_min\_res\_unit变量的值来提高性能。但是，注意不要使它变得太大（参见前面的条目）。

**9、查询高速缓冲状态和维护**

可以使用下面的语句检查MySQL服务器是否提供查询缓存功能：

mysql> **SHOW VARIABLES LIKE 'have\_query\_cache';**

+------------------+-------+

| Variable\_name    | Value |

+------------------+-------+

| have\_query\_cache | YES   |

+------------------+-------+

FLUSH QUERY CACHE：语句来清理查询缓存碎片以提高内存使用性能。该语句不从缓存中移出任何查询。

RESET QUERY CACHE：语句从查询缓存中移出所有查询。FLUSH TABLES语句也执行同样的工作。

SHOW STATUS：为了监视查询缓存性能，使用SHOW STATUS查看缓存状态变量，例如：

mysql> **SHOW STATUS LIKE 'Qcache%';**

+-------------------------+--------+

|变量名                   |值 |

+-------------------------+--------+

| Qcache\_free\_blocks      | 36     |

| Qcache\_free\_memory      | 138488 |

| Qcache\_hits             | 79570  |

| Qcache\_inserts          | 27087  |

| Qcache\_lowmem\_prunes    | 3114   |

| Qcache\_not\_cached       | 22989  |

| Qcache\_queries\_in\_cache | 415    |

| Qcache\_total\_blocks     | 912    |

+-------------------------+--------+

这些变量的描述见[5.3.4节，“服务器状态变量”](http://dev.mysql.com/doc/refman/5.1/zh/database-administration.html#server-status-variables)。这里描述它们的一些应用。

SELECT查询的总数量等价于：

Com\_select

+ Qcache\_hits

+ queries with errors found by parser

Com\_select的值等价于：

Qcache\_inserts

+ Qcache\_not\_cached

+ queries with errors found during columns/rights check

查询缓存使用长度可变块，因此Qcache\_total\_blocks和Qcache\_free\_blocks可以显示查询缓存内存碎片。执行FLUSH QUERY CACHE后，只保留一个空闲块。

每个缓存查询至少需要两个块（一个块用于查询文本，一个或多个块用于查询结果）。并且，每一个查询使用的每个表需要一个块。但是，如果两个或多个查询使用相同的表，仅需要分配一个块。

Qcache\_lowmem\_prunes状态变量提供的信息能够帮助你你调整查询缓存的大小。它计算为了缓存新的查询而从查询缓冲区中移出到自由内存中的查询的数目。查询缓冲区使用最近最少使用(LRU)策略来确定哪些查询从缓冲区中移出。调整信息在[5.13.3节，“查询高速缓冲配置”](http://dev.mysql.com/doc/refman/5.1/zh/database-administration.html#query-cache-configuration)中给出。