**MySQL查询缓存简介：**

MySQL查询缓存是MySQL中比较独特的一个缓存区域，用来缓存特定Query的整个结果集信息，且共享给所有客户端。为了提高完全相同的Query语句的响应速度，MySQL Server会对查询语句进行Hash计算后，把得到的hash值与Query查询的结果集对应存放在Query Cache中。当MySQL Server打开Query Cache之后，MySQL Server会对接收到的每一个SELECT 语句通过特定的Hash算法计算该Query的Hash值，然后通过该哈希值到Query Cache中去匹配。

* 如果没有匹配，将这个hash值存放在一个hash链表中，并将Query的结果集存放到cache中，存放哈希值链表的每个hash节点存放了相应Quey结果集在cache中的地址，以及该query所涉及到一些table相关信息；
* 如果通过hash值匹配到了一样的Query，则直接将cache中相应的Query结果集返回给客户端。

P.s

目前MySQL Query Cache只会cache select语句，其他类似show ，use的语句不会被cache. MySQL 的每个Query Cache都是以SQL文本作为key来存储的，在应用Query Cache之前，SQL文本不会做任何处理。也就是说，两个SQL语句，只要相差哪怕一个字符(例如大小写不一样，多一个空格，多注释) ，那么这两个SQL将使用不同的Cache地址。**如: 下面三条SQL将会被存储在三个不同的缓存里，虽然他们的结果都是一样的。**

select \* FROM people where name='surfchen';

select \* FROM people where /\*hey~\*/ name='surfchen';

SELECT \* FROM people where name='surfchen';

**缓存规则：**

开启了缓存，MySQL Server会自动将查询语句和结果集返回到内存，下次再查直接从内存中取。

缓存的结果是通过sessions共享的，所以一个client查询的缓存结果，另一个client也可以使用

MySQL Query Cache内容为 select 的结果集, cache 使用完整的SQL字符串做 key, 并区分大小写，空格等。即两个SQL必须完全一致才会导致cache命中。即检查查询缓存时，MySQL Server不会对SQL做任何处理，它精确的使用客户端传来的查询，只要字符大小写或注释有点不同，查询缓存就认为是不同的查询。

prepared statement永远不会cache到结果，即使参数完全一样。在 5.1 之后会得到改善。

where条件中如包含任何一个不确定的函数将永远不会被cache, 比如current\_date, now等。

select \* from foo where date1=current\_date -- 不会被 cache

select \* from foo where date1='2008-12-30' -- 被cache, 正确的做法

太大的result set不会被cache (< query\_cache\_limit)

MySQL缓存在分库分表环境下是不起作用的

执行SQL里有触发器,自定义函数时，MySQL缓存也是不起作用的

**缓存失效：**

在表的结构或数据发生改变时，查询缓存中的数据不再有效。如INSERT、UPDATE、 DELETE、TRUNCATE、ALTER TABLE、DROP TABLE或DROP DATABASE会导致缓存数据失效。所以查询缓存适合有大量相同查询的应用，不适合有大量数据更新的应用。

一旦表数据进行任何一行的修改，基于该表相关cache立即全部失效。

**手动清理缓存手动清理缓存可以使用下面三个SQL：**

FLUSH QUERY CACHE； #清理查询缓存内存碎片

RESET QUERY CACHE；#从查询缓存中移除所有查询

FLUSH TABLES； #关闭所有打开的表，同时该操作会清空查询缓存中的内容

**查询命中率：**

查询缓存命中率：Qcache\_hits/(Qcahce\_hits+Com\_select)，查询缓存命中率多大才是好的命中率，需要具体情况具体分析。只要查询缓存带来的效率提升大于查询缓存带来的额外消耗，即使30%的命中率也是值得。另外，缓存了哪些查询也很重要，如果被缓存的查询本身消耗巨大，那么即使缓存命中率低，对系统性能提升仍然是有好处的。

**MySQL缓存相关的配置参数:**

mysql> show variables like '%query\_cache%';

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

| Variable\_name | Value |

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

| have\_query\_cache | YES | --查询此版本MySQL服务器是否支持缓存机制

| query\_cache\_limit | 1048576 | --可缓存具体查询结果的最大值

| query\_cache\_min\_res\_unit | 4096 | --查询缓存分配的最小块的大小(字节)

| query\_cache\_size | 599040 | --查询缓存的大小

| query\_cache\_type | ON | --是否支持查询缓存

| query\_cache\_wlock\_invalidate | OFF | --控制当有写锁加在表上的时候，是否先让该表相关的 Query Cache失效

6 rows in set (0.02 sec)

* have\_query\_cache

该MySQL Server是否支持Query Cache。

* query\_cache\_limit

MySQL能够缓存的最大查询结果，查询结果大于该值时不会被缓存。默认值是1048576(1MB) 。如果某个查询的结果超出了这个值，Qcache\_not\_cached的值会加1，如果某个操作总是超出，可以考虑在SQL中加上SQL\_NO\_CACHE来避免额外的消耗。

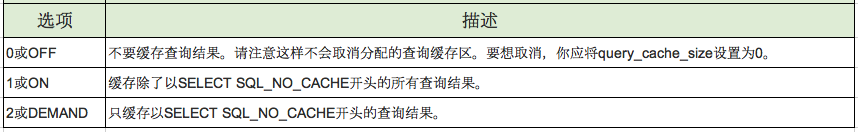
* query\_cache\_min\_res\_unit

查询缓存分配的最小块的大小(字节)。 默认值是4096(4KB)。当查询进行的时候，MySQL把查询结果保存在qurey cache中，但如果要保存的结果比较大，超过query\_cache\_min\_res\_unit的值 ，这时候mysql将一边检索结果，一边进行保存结果，所以，有时候并不是把所有结果全部得到后再进行一次性保存，而是每次分配一块query\_cache\_min\_res\_unit大小的内存空间保存结果集，使用完后，接着再分配一个这样的块，如果还不够，接着再分配一个块，依此类推，也就是说，有可能在一次查询中，mysql要进行多次内存分配的操作。适当的调节query\_cache\_min\_res\_unit可以优化内存如果你的查询结果都是一些small result,默认的query\_cache\_min\_res\_unit可能会造成大量的内存碎片如果你的查询结果都是一些larger result，你可以适当的把query\_cache\_min\_res\_unit调大

* query\_cache\_size

为缓存查询结果分配的内存的数量，单位是字节，且数值必须是1024的整数倍。默认值是0，即禁用查询缓存。请注意如果设置了该值，即使query\_cache\_type设置为0也将分配此数量的内存。

* query\_cache\_type

设置查询缓存类型，默认为ON。设置GLOBAL值可以设置后面的所有客户端连接的类型。客户端可以设置SESSION值以影响他们自己对查询缓存的使用。下面的表显示了可能的值：

* query\_cache\_wlock\_invalidate

如果某个表被锁住，是否返回缓存中的数据，默认关闭，也是建议的。

将该变量设置为0, 一般情况，当客户端对MyISAM表进行WRITE锁定时，如果查询结果位于查询缓存中，则其它客户端未被锁定，可以对该表进行查询。

将该变量设置为1，则可以对表进行WRITE锁定，使查询缓存内所有对该表进行的查询变得非法。这样当锁定生效时，可以强制其它试图访问表的客户端来等待。

**开启关闭缓存:**

* 开启缓存

mysql> set global query\_cache\_size = 600000; --设置缓存内存大小

mysql> set global query\_cache\_type = ON; --开启查询缓存

* 关闭缓存

mysql> set global query\_cache\_size = 0; --设置缓存内存大小为0， 即初始化是不分配缓存内存

mysql> set global query\_cache\_type = OFF; --关闭查询缓存

**MySQL缓存状态查看:**

mysql> SHOW STATUS LIKE 'Qcache%';

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

| Variable\_name | Value |

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

| Qcache\_free\_blocks | 1 | ----在查询缓存中的闲置块，如果该值比较大，则说明Query Cache中的内存碎片可能比较多。FLUSH QUERY CACHE会对缓存中的碎片进行整理，从而得到一个较大的空闲内存块。

| Qcache\_free\_memory | 382704 | ----剩余缓存的大小

| Qcache\_hits | 198 | ----缓存命中次数

| Qcache\_inserts | 131 | ----缓存被插入的次数，也就是查询没有命中的次数。

| Qcache\_lowmem\_prunes | 0 | ----由于内存低而被删除掉的缓存条数，如果这个数值在不断增长，那么一般是Query Cache的空闲内存不足（通过Qcache\_free\_memory判断），或者内存碎片较严重（通过Qcache\_free\_blocks判断）。

| Qcache\_not\_cached | 169 | ----没有被缓存的条数，有三种情况会导致查询结果不会被缓存：其一，由于query\_cache\_type的设置；其二，查询不是SELECT语句；其三，使用了now()之类的函数，导致查询语句一直在变化。

| Qcache\_queries\_in\_cache | 128 | ----缓存中有多少条查询语句

| Qcache\_total\_blocks | 281 | ----总块数

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

**MySQL Query Cache优缺点**

优点Query Cache的查询，发生在MySQL接收到客户端的查询请求、查询权限验证之后和查询SQL解析之前。也就是说，当MySQL接收到客户端的查询SQL之后，仅仅只需要对其进行相应的权限验证之后，就会通过Query Cache来查找结果，甚至都不需要经过Optimizer模块进行执行计划的分析优化，更不需要发生任何存储引擎的交互。由于Query Cache是基于内存的，直接从内存中返回相应的查询结果，因此减少了大量的磁盘I/O和CPU计算，导致效率非常高。

缺点Query Cache的优点很明显，但是也不能忽略它所带来的一些缺点：

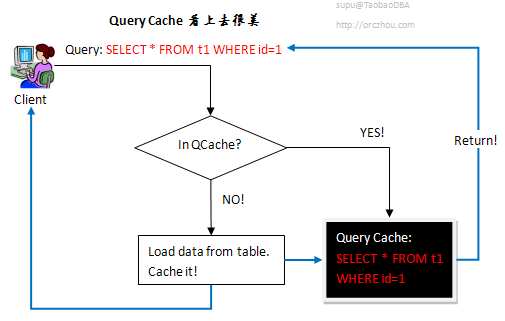
查询语句的hash计算和hash查找带来的资源消耗。如果将query\_cache\_type设置为1（也就是ON），那么MySQL会对每条接收到的SELECT类型的查询进行hash计算，然后查找这个查询的缓存结果是否存在。虽然hash计算和查找的效率已经足够高了，一条查询语句所带来的开销可以忽略，但一旦涉及到高并发，有成千上万条查询语句时，hash计算和查找所带来的开销就必须重视了。

Query Cache的失效问题。如果表的变更比较频繁，则会造成Query Cache的失效率非常高。表的变更不仅仅指表中的数据发生变化，还包括表结构或者索引的任何变化。

查询语句不同，但查询结果相同的查询都会被缓存，这样便会造成内存资源的过度消耗。查询语句的字符大小写、空格或者注释的不同，Query Cache都会认为是不同的查询（因为他们的hash值会不同）。

相关系统变量设置不合理会造成大量的内存碎片，这样便会导致Query Cache频繁清理内存。

**生产如何设置MySQL Query Cache:**



MySQL中的Query Cache是一个适用较少情况的缓存机制。如上图所示，如果缓存命中率非常高的话，有测试表明在极端情况下可以提高效率238%。但实际情况如何？Query Cache有如下规则，如果数据表被更改，那么和这个数据表相关的全部Cache全部都会无效，并删除之。这里“数据表更改”包括: INSERT, UPDATE, DELETE, TRUNCATE, ALTER TABLE, DROP TABLE, or DROP DATABASE等。举个例子，如果数据表posts访问频繁，那么意味着它的很多数据会被QC缓存起来，但是每一次posts数据表的更新，无论更新是不是影响到了cache的数据，都会将全部和posts表相关的cache清除。如果你的数据表更新频繁的话，那么Query Cache将会成为系统的负担。有实验表明，糟糕时，QC会降低系统13%的处理能力。

**如果你的应用对数据库的更新很少，那么QC将会作用显著。比较典型的如博客系统，一般博客更新相对较慢，数据表相对稳定不变，这时候QC的作用会比较明显。**

但是一个更新频繁的BBS系统。下面是一个实际运行的论坛数据库的状态参数：QCache\_hit 5280438QCache\_insert 8008948Qcache\_not\_cache 95372Com select 8104159可以看到，数据库一共往Query Cache中写入了约800W次缓存，但是实际命中的只有约500W次。也就是说，每一个缓存的使用率约为0.66次。很难说，该缓存的作用是否大于Query Cache系统所带来的开销。但是有一点是很肯定的，Query Cache缓存的作用是很微小的，如果应用层能够实现缓存，将可以忽略Query Cache的效果。

**所以，如果经常有更新的系统，想要获得较高tps的话，建议一开始就关闭Query Cache**