Atitit mysql cache机制总结 Buffer Pool 取代了query cache

# **MySQL之缓存讲解大全**



[wcdzxxgc](https://blog.csdn.net/wcdzxxgc" \t "https://blog.csdn.net/wcdzxxgc/article/details/_blank) 2010-10-19 11:15:31 IMG_257 39 IMG_258 收藏

分类专栏： [Mysql](https://blog.csdn.net/wcdzxxgc/category_8268621.html" \t "https://blog.csdn.net/wcdzxxgc/article/details/_blank) 文章标签： [MySQL](https://www.csdn.net/tags/MtTaEg5sOTYwNC1ibG9n.html" \t "https://blog.csdn.net/wcdzxxgc/article/details/_blank) [Cache](https://www.csdn.net/tags/MtjaUg1sOTAyMjYtYmxvZwO0O0OO0O0O.html" \t "https://blog.csdn.net/wcdzxxgc/article/details/_blank) [SQL](https://www.csdn.net/tags/MtTaEg0sMDg0OTctYmxvZwO0O0OO0O0O.html" \t "https://blog.csdn.net/wcdzxxgc/article/details/_blank) [数据结构](https://www.csdn.net/tags/MtTacg3sNTk3OC1ibG9n.html" \t "https://blog.csdn.net/wcdzxxgc/article/details/_blank) [Oracle](https://www.csdn.net/tags/MtTaMg5sNTY0Ni1ibG9n.html" \t "https://blog.csdn.net/wcdzxxgc/article/details/_blank)

版权

全局共享内存主要是 MySQL Instance（mysqld进程）以及底层存储引擎用来暂存各种全局运算及可共享的暂存信息，如存储查询缓存的 Query Cache，缓存连接线程的 Thread Cache，缓存表文件句柄信息的 Table Cache，缓存二进制日志的 BinLog Buffer， 缓存 MyISAM 存储引擎索引键的 Key Buffer以及存储 InnoDB 数据和索引的 InnoDB Buffer Pool 等等。下面针对 MySQL 主要的共享内存进行一个简单的分析。

****MySQL缓存****

缓存有两个纬度，MySQL层：查询缓存Query Cache（QCache）；存储引擎层：InnoDB\_Buffer\_Pool。

　　1、Qcacche缓存的是SQL语句及对应的结果集，缓存在内存，最简单的情况是SQL一直不重复，那Qcache的命令率肯定是0；

1. buffer pool中缓存的是整张表中的数据，缓存在内存，SQL再变只要数据都在内存，那么命中率就是100%。

连接线程缓存（Thread Cache）：连接线程是 MySQL 为了提高创建连接线程的效率，将部分空闲的连接线程保持在一个缓存区以备新进连接请求的时候使用，这尤其对那些使用短连接的应用程序来说可以极大的提高创建连接的效率。当我们通过 thread\_cache\_size 设置了连接线程缓存池可以缓存的连接线程的大小之后，可以通过(Connections - Threads\_created) / Connections \* 100% 计算出连接线程缓存的命中率。注意，这里设置的是可以缓存的连接线程的数目，而不是内存空间的大小。  
  
表缓存（Table Cache）：表缓存区主要用来缓存表文件的文件句柄信息，在 MySQL5.1.3之前的版本通过 table\_cache 参数设置，但从MySQL5.1.3开始改为 table\_open\_cache 来设置其大小。当我们的客户端程序提交 Query 给 MySQL 的时候，MySQL 需要对 Query 所涉及到的每一个表都取得一个表文件句柄信息，如果没有 Table Cache，那么 MySQL 就不得不频繁的进行打开关闭文件操作，无疑会对系统性能产生一定的影响，Table Cache 正是为了解决这一问题而产生的。在有了 Table Cache 之后，MySQL 每次需要获取某个表文件的句柄信息的时候，首先会到 Table Cache 中查找是否存在空闲状态的表文件句柄。如果有，则取出直接使用，没有的话就只能进行打开文件操作获得文件句柄信息。在使用完之后，MySQL 会将该文件句柄信息再放回 Table Cache 池中，以供其他线程使用。注意，这里设置的是可以缓存的表文件句柄信息的数目，而不是内存空间的大小。  
  
表定义信息缓存（Table definition Cache）：表定义信息缓存是从 MySQL5.1.3 版本才开始引入的一个新的缓存区，用来存放表定义信息。当我们的 MySQL 中使用了较多的表的时候，此缓存无疑会提高对表定义信息的访问效率。MySQL 提供了 table\_definition\_cache 参数给我们设置可以缓存的表的数量。在 MySQL5.1.25 之前的版本中，默认值为128，从 MySQL5.1.25 版本开始，则将默认值调整为 256 了，最大设置值为524288。注意，这里设置的是可以缓存的表定义信息的数目，而不是内存空间的大小。  
  
二进制日志缓冲区（Binlog Buffer）：二进制日志缓冲区主要用来缓存由于各种数据变更操做所产生的 Binary Log 信息。为了提高系统的性能，MySQL 并不是每次都是将二进制日志直接写入 Log File，而是先将信息写入 Binlog Buffer 中，当满足某些特定的条件（如 sync\_binlog参数设置）之后再一次写入 Log File 中。我们可以通过 binlog\_cache\_size 来设置其可以使用的内存大小，同时通过 max\_binlog\_cache\_size 限制其最大大小（当单个事务过大的时候 MySQL 会申请更多的内存）。当所需内存大于 max\_binlog\_cache\_size 参数设置的时候，MySQL 会报错："Multi-statement transaction required more than 'max\_binlog\_cache\_size' bytes of storage"。

InnoDB 数据和索引缓存（InnoDB Buffer Pool）：InnoDB Buffer Pool 对 InnoDB 存储引擎的作用类似于 Key Buffer Cache 对 MyISAM 存储引擎的影响，主要的不同在于 InnoDB Buffer Pool 不仅仅缓存索引数据，还会缓存表的数据，而且完全按照数据文件中的数据快结构信息来缓存，这一点和 Oracle SGA 中的 database buffer cache 非常类似。所以，InnoDB Buffer Pool 对 InnoDB 存储引擎的性能影响之大就可想而知了。可以通过 (Innodb\_buffer\_pool\_read\_requests - Innodb\_buffer\_pool\_reads) / Innodb\_buffer\_pool\_read\_requests \* 100% 计算得到 InnoDB Buffer Pool 的命中率。  
  
InnoDB 字典信息缓存（InnoDB Additional Memory Pool）:InnoDB 字典信息缓存主要用来存放 InnoDB 存储引擎的字典信息以及一些 internal 的共享数据结构信息。所以其大小也与系统中所使用的 InnoDB 存储引擎表的数量有较大关系。不过，如果我们通过 innodb\_additional\_mem\_pool\_size 参数所设置的内存大小不够，InnoDB 会自动申请更多的内存，并在 MySQL 的 Error Log 中记录警告信息。  
  
这里所列举的各种共享内存，是我个人认为对 MySQL 性能有较大影响的集中主要的共享内存。实际上，除了这些共享内存之外，MySQL 还存在很多其他的共享内存信息，如当同时请求连接过多的时候用来存放连接请求信息的back\_log队列等。

**查询缓存（QueryCache）**

1、关于查询缓存机制

　　开启了缓存，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；

查询缓存会跟踪系统中每张表，若表发生变化，则和该张表相关的所有查询缓存全部失效，这是和buffer pool缓存机制很大的区别；

**3.3 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**

# ****二、存储引擎层-innodb buffer pool****

　　buffer pool是innodb存储引擎带的一个缓存池，查询数据的时候，它首先会从内存中查询，如果内存中存在的话，直接返回，从而提高查询响应时间。Buffer pool是设置的越大越好，一般设置为服务器物理内存的70%。

1、Innodb\_buffer\_pool参数

2、Innodb\_buffer\_pool状态

mysql> show status like '%innodb\_buffer\_pool%';+---------------------------------------+--------------------------------------------------+| Variable\_name | Value |+---------------------------------------+--------------------------------------------------+| Innodb\_buffer\_pool\_dump\_status | Dumping of buffer pool not started || Innodb\_buffer\_pool\_load\_status | Buffer pool(s) load completed at **170430** **7**:**07**:**12** || Innodb\_buffer\_pool\_resize\_status | || Innodb\_buffer\_pool\_pages\_data | **241** || Innodb\_buffer\_pool\_bytes\_data | **3948544** || Innodb\_buffer\_pool\_pages\_dirty | **0** || Innodb\_buffer\_pool\_bytes\_dirty | **0** || Innodb\_buffer\_pool\_pages\_flushed | **176** || Innodb\_buffer\_pool\_pages\_free | **7951** || Innodb\_buffer\_pool\_pages\_misc | **0** || Innodb\_buffer\_pool\_pages\_total | **8192** || Innodb\_buffer\_pool\_read\_ahead\_rnd | **0** || Innodb\_buffer\_pool\_read\_ahead | **0** || Innodb\_buffer\_pool\_read\_ahead\_evicted | **0** || Innodb\_buffer\_pool\_read\_requests | **53710** || Innodb\_buffer\_pool\_reads | **201** || Innodb\_buffer\_pool\_wait\_free | **0** || Innodb\_buffer\_pool\_write\_requests | **45242** |+---------------------------------------+--------------------------------------------------+

　　Innodb\_buffer\_pool\_read\_requests　　#逻辑读(缓存读)请求次数，也是读的请求次数

　　Innodb\_buffer\_pool\_reads　　#从物理磁盘中获取到数据的次数

注：

　　逻辑读就是从buffer pool的读，但是也会包含物理读，因为物理读也要是先将从disk中读取的数据放入buffer pool里，然后再进行逻辑读。所以：总的逻辑读也就是读的请求次数。

*读的命中率=(Innodb\_buffer\_pool\_read\_requests- Innodb\_buffer\_pool\_reads)/ Innodb\_buffer\_pool\_read\_requests*