mysql 中的内存使用

mysql 的内存管理其实是比较复杂的，小结下，分为两类：

1 线程独享内存

2 全局共享内存

先说线程独享内存：

线程栈信息使用内存(thread_stack)：主要用来存放每一个线程自身的标识信息，如线程 id，线程运行时基本信息等等，我们可以通过 thread_stack 参数来设置为每一个线程栈分配多大的内存。

排序使用内存(sort_buffer_size)：MySQL 用此内存区域进行排序操作（filesort），完成客户端的排序请求。当我们设置的排序区缓存大小无法满足排序实际所需内存的时候，MySQL 会将数据写入磁盘文件来完成排序。由于磁盘和内存的读写性能完全不在一个数量级，所以 sort_buffer_size 参数对排序操作的性能影响绝对不可小视。排序操作的实现原理请参考：MySQL Order By 的实现分析。

Join 操作使用内存(join_buffer_size)：应用程序经常会出现一些两表（或多表）Join 的操作需求，MySQL 在完成某些 Join 需求的时候（all/index join），为了减少参与 Join 的“被驱动表”的读取次数以提高性能，需要使用到 Join Buffer 来协助完成 Join 操作(具体 Join 实现算法请参考：MySQL 中的 Join 基本实现原理)。当 Join Buffer 太小，MySQL 不会将该 Buffer 存入磁盘文件，而是先将 Join Buffer 中的结果集与需要 Join 的表进行 Join 操作，然后清空 Join Buffer 中的数据，继续将剩余的结果集写入此 Buffer 中，如此往复。这势必会造成被驱动表需要被多次读取，成倍增加 IO 访问，降低效率。

顺序读取数据缓冲区使用内存(read_buffer_size)：这部分内存主要用于当需要顺序读取数据的时候，如无发使用索引的情况下的全表扫描，全索引扫描等。在这种时候，MySQL 按照数据的存储顺序依次读取数据块，每次读取的数据快首先会暂存在 read_buffer_size 中，当 buffer 空间被写满或者全部数据读取结束后，再将 buffer 中的数据返回给上层调用者，以提高效率。

随机读取数据缓冲区使用内存(read_rnd_buffer_size)：和顺序读取相对应，当 MySQL 进行非顺序读取（随机读取）数据块的时候，会利用这个缓冲区暂存读取的数据。如根据索引信息读取表数据，根据排序后的结果集与表进行 Join 等等。总的来说，就是当数据块的读取需要满足一定的顺序的情况下，MySQL 就需要产生随机读取，进而使用到 read_rnd_buffer_size 参数所设置的内存缓冲区。

连接信息及返回客户端前结果集暂存使用内存(net_buffer_size)：这部分用来存放客户端连接线程的连接信息和返回客户端的结果集。当 MySQL 开始产生可以返回的结果集，会在通过网络返回给客户端请求线程之前，会先暂存在通过 net_buffer_size 所设置的缓冲区中，等满足一定大小的时候才开始向客户端发送，以提高网络传输效率。不过，net_buffer_size 参数所设置的仅仅只是该缓存区的初始化大小，MySQL 会根据实际需要

自行申请更多的内存以满足需求，但最大不会超过 max_allowed_packet 参数大小。

批量插入暂存使用内存(bulk_insert_buffer_size)：当我们使用如 insert …values(…),(…),(…)… 的方式进行批量插入的时候，MySQL 会先将提交的数据放如一个缓存空间中，当该缓存空间被写满或者提交完所有数据之后，MySQL 才会一次性将该缓存空间中的数据写入数据库并清空缓存。此外，当我们进行 LOAD DATA INFILE 操作来将文本文件中的数据 Load 进数据库的时候，同样会使用到此缓冲区。

临时表使用内存(tmp_table_size)：当我们进行一些特殊操作如需要使用临时表才能完成的 Order By，Group By 等等，MySQL 可能需要使用到临时表。当我们的临时表较小（小于 tmp_table_size 参数所设置的大小）的时候，MySQL 会将临时表创建成内存临时表，只有当 tmp_table_size 所设置的大小无法装下整个临时表的时候，MySQL 才会将该表创建成 MyISAM 存储引擎的表存放在磁盘上。不过，当另一个系统参数 max_heap_table_size 的大小还小于 tmp_table_size 的时候，MySQL 将使用 max_heap_table_size 参数所设置大小作为最大的内存临时表大小，而忽略 tmp_table_size 所设置的值。而且 tmp_table_size 参数从 MySQL 5.1.2 才开始有，之前一直使用 max_heap_table_size。

2　全局共享

全局共享内则主要是 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。

查询缓存（Query Cache）：查询缓存是 MySQL 比较独特的一个缓存区域，用来缓存特定 Query 的结果集（Result Set）信息，且共享给所有客户端。通过对 Query 语句进行特定的 Hash 计算之后与结果集对应存放在 Query Cache 中，以提高完全相同的 Query 语句的相应速度。当我们打开 MySQL 的 Query Cache 之后，MySQL 接收到每一个 SELECT 类型的 Query 之后都会首先通过固定的 Hash 算法得到该 Query 的 Hash 值，然后到 Query Cache 中查找是否有对应的 Query Cache。如果有，则直接将 Cache 的结果集返回给客户端。如果没有，再进行后续操作，得到对应的结果集之后将该结果集缓存到 Query Cache 中，再返回给客户端。当任何一个表的数据发生任何变化之后，与该表相关的所有 Query Cache 全部会失效，所以 Query Cache 对变更比较频繁的表并不是非常适用，但对那些变更较少的表是非常合适的，可以极大程度的提高查询效率，如那些静态资源表，配置表等等。为了尽可能高效的利用 Query Cache，MySQL 针对 Query Cache 设计了多个 query_cache_type 值和两个 Query Hint：SQL_CACHE 和 SQL_NO_CACHE。当 query_cache_type 设置为 0（或者 OFF）的时候不使用 Query Cache，当设置为 1（或者 ON）的时候，当且仅当 Query 中使用了 SQL_NO_CACHE 的时候 MySQL 会忽略 Query Cache，当 query_cache_type 设置为 2（或者 DEMAND）的时候，当且仅当 Query 中使用了 SQL_CACHE 提示之后，MySQL 才会针对该 Query 使用 Query Cache。可以通过 query_cache_size 来设置可以使用的最大内存空间。

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”。

MyISAM 索引缓存（Key Buffer）：MyISAM 索引缓存将 MyISAM 表的索引信息缓存在内存中，以提高其访问性能。这个缓存可以说是影响 MyISAM 存储引擎性能的最重要因素之一了，通过 key_buffere_size 设置可以使用的最大内存空间。

InnoDB 日志缓冲区（InnoDB Log Buffer）：这是 InnoDB 存储引擎的事务日志所使用的缓冲区。类似于 Binlog Buffer，InnoDB 在写事务日志的时候，为了提高性能，也是先将信息写入 Innofb Log Buffer 中，当满足 innodb_flush_log_trx_commit 参数所设置的相应条件（或者日志缓冲区写满）之后，才会将日志写到文件（或者同步到磁盘）中。可以通过 innodb_log_buffer_size 参数设置其可以使用的最大内存空间。
注：innodb_flush_log_trx_commit 参数对 InnoDB Log 的写入性能有非常关键的影响。该参数可以设置为 0，1，2，解释如下：

0：log buffer 中的数据将以每秒一次的频率写入到 log file 中，且同时会进行文件系统到磁盘的同步操作，但是每个事务的 commit 并不会触发任何 log buffer 到 log file 的刷新或者文件系统到磁盘的刷新操作；
1：在每次事务提交的时候将 log buffer 中的数据都会写入到 log file，同时也会触发文件系统到磁盘的同步；
2：事务提交会触发 log buffer 到 log file 的刷新，但并不会触发磁盘文件系统到磁盘的同步。此外，每秒会有一次文件系统到磁盘同步操作。
此外，MySQL 文档中还提到，这几种设置中的每秒同步一次的机制，可能并不会完全确保非常准确的每秒就一定会发生同步，还取决于进程调度的问题。实际上，InnoDB 能否真正满足此参数所设置值代表的意义正常 Recovery 还是受到了不同 OS 下文件系统以及磁盘本身的限制，可能有些时候在并没有真正完成磁盘同步的情况下也会告诉 mysqld 已经完成了磁盘同步。

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。