# 概述

以MySQL的架构来划分内存管理比较合理。即Server层与InnoDB层（Engine 层），而这两块内存是由不同的方式进行管理的。

其中Server层是由mem\_root来进行内存管理，包括Sharing与Thead memory；而InnoDB层则主要由Free List、LRU List、FLU List等多个链表来统一管理 Innodb\_buffer\_pool。

参考：

<https://mp.weixin.qq.com/s/jlFueo-WnR3gILR38uzeIg>

<https://mp.weixin.qq.com/s/yi2_PKpi8ea3lhwEFTe5WA>

MySQL内存分配管理：<http://mysql.taobao.org/monthly/2022/11/02/>

MySQL内存分配与管理：<https://zhuanlan.zhihu.com/p/596425183>

mysql performance storage engine：

<https://www.cnblogs.com/zhedan/p/12488104.html>

剖析 Performance Schema 内存管理：

<https://baijiahao.baidu.com/s?id=1712050278658782079&wfr=spider&for=pc>

IO\_CACHE 源码解析：<http://mysql.taobao.org/monthly/2018/09/03/>

内存分配机制：<http://mysql.taobao.org/monthly/2017/08/06/>

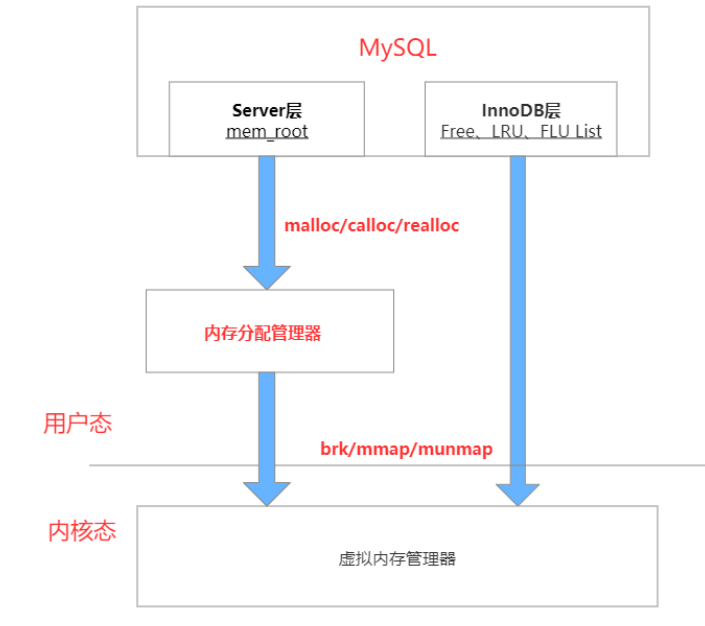
MEM\_ROOT：MySQL Server的内存管理方式。

Buffer pool：InnoDB存储引擎的内存管理方式。

Tcmalloc：Linux系统的内存优化器。

IO\_CACHE：MySQL中的IO\_CACHE的作用就是把连续的文件读写操作，经过缓冲，转化为4K对齐的文件读写操作。

# 原理



## mem\_root

MySQL Server层中广泛使用mem\_root结构体来管理内存，避免频繁调用内存操作，提升性能，统一的分配和管理内存也可以防止发生内存泄漏：

初始化mem\_toot --内存申请

> init\_sql\_alloc --线程分配内存

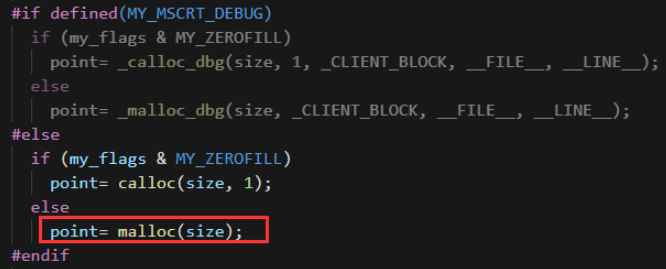
| > init\_alloc\_root alloc\_root

| | > my\_malloc --内存释放

| | | > my\_raw\_malloc free\_alloc

| | | | > malloc

MySQL 首先通过init\_alloc\_root函数初始化一块较大的内存空间，实际上最终是通过malloc函数向内存分配器申请内存空间，然后每次再调用alloc\_root函数在这块内存空间中分配出内存进行使用，其目的就是将多次零散的malloc 操作合并成一次大的malloc操作，以提升性能。



MySQL Server并不是完全由一个mem\_root结构体管理所有Server层内存（这个跟Innodb\_buffer\_pool不同），**不同的线程会产生不同的mem\_root来管理各自的内存，不同的mem\_root之间互相没有影响**。

说明：在初始化MEM\_ROOT的时候会通过参数KEY（PSI\_memory\_key类型）来区别不同线程的内存。比如如果需要打开一个表，可以使用的KEY是key\_memory\_table\_share或key\_memory\_TABLE。

Server层的内存管理相较于InnoDB层来说复杂的多，也更容易产生内存碎片，很多MySQL内存问题都出自于此。

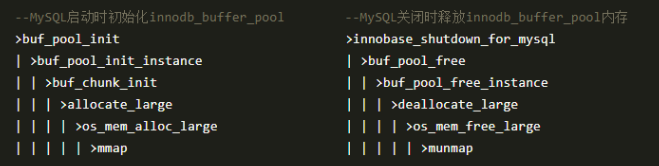
## Innodb\_buffer\_pool

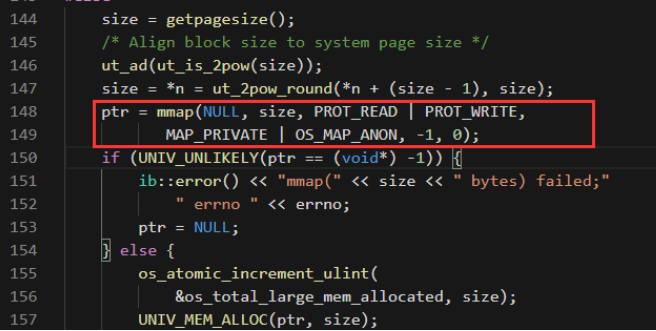
参考：

InnoDB内存管理：<http://mysql.taobao.org/monthly/2021/01/06/>

innodb\_buffer\_pool\_size在线修改：<http://mysql.taobao.org/monthly/2018/03/06/>

MySQL5.7开始支持Innodb\_buffer\_pool动态调整大小，每个buffer\_pool\_instance都由同样个数的chunk组成，每个chunk内存大小为 innodb\_buffer\_pool\_chunk\_size，所以Innodb\_buffer\_pool以 innodb\_buffer\_pool\_chunk\_size为基本单位进行动态增大和缩小。





可以看到，Innodb\_buffer\_pool内存初始化是通过mmap()方式直接向操作系统申请内存，每次申请的大小为innodb\_buffer\_pool\_chunk\_size，最终会申请Innodb\_buffer\_pool\_size大小的文件映射段动态内存。这部分内存空间初始化后仅仅是虚拟内存，等真正使用时，才会分配物理内存。

根据之前 Linux 下内存分配原理，mmap() 方式申请的内存会在文件映射段分配内存，而且在释放时会直接归还系统。

仔细想下，Innodb\_buffer\_pool的内存分配使用确实如此，当Innodb\_buffer\_pool初始化后，会慢慢被数据页及索引页等填充满，然后就一直保持Innodb\_buffer\_pool\_size大小左右的物理内存占用。除非是在线减少Innodb\_buffer\_pool或是关闭MySQL才会通过munmap()方式释放内存，这里的内存释放是直接返回给操作系统。

Innodb\_buffer\_pool的内存主要是通过Free List、LRU List、FLU List、Unzip LRU List等4个链表来进行管理分配。

Free List：缓存空闲页

LRU List：缓存数据页

FLU List：缓存所有脏页

Unzip LRU List：缓存所有解压页

PS：源码全局遍历下来，只有innodb\_buffer\_pool与online ddl的内存管理是采用mmap()方式直接向操作系统申请内存分配，而不需要经过内存分配器。

## IO\_CACHE

# 源码分析

# 对比

## PostgreSQL内存管理

## LevelDB内存管理/arena