内存管理器

内存管理器是 Windows 执行体的一部分，因此也位于Ntoskrnl.exe文件中。它是执行体中最大的组件，这也可以体现其重要性和复杂性。HAL 不包含内存管理器的任何部分，内存管理器包含如下这些组件。

1. 一组执行体系统服务，负责虚拟内存的分配、撤销分配和管理，大部分这些服务是通过 Windows API或内核模式设备驱动程序接口暴露的。
2. 一种用于解决硬件检测到的内存管理异常问题，以及让虚拟页面代表进程实现驻留的转换无效 (translation-not-valid)和访问错误 (access fault) 陷阱处理程序(trap handler).
3. 6个关键的顶层例程，每个都运行在System 进程的6个不同内核模式线程之一的内部。
   * **平衡集管理器(KeBalanceSetManager，优先级为 17)**。负责调用内部例程，工作集管理器(MmWorkingSetManager)每秒钟调用一次，并会在可用内存低于某个阈值时调用。工作集管理器驱动着整体内存管理策略，如工作集修剪、老化以及已修改页面的写入。
   * **进程/栈交换器(KeSwapProcessOrStack，优先级为23)**。负责执行进程栈和内核线程栈的换入与换出。平衡集管理器和内核中的线程调度代码会在需要执行换入和换出操作时唤醒该线程。
   * **已修改页面写出器(MiModifiedPageWriter，优先级为18)**。负责将已修改列表中的脏页面重新写回到相应的页面文件。当已修改列表需要缩小时，会唤醒该线程。
   * **已映射页面写出器(MiMappedPageWriter，优先级为18)**。负责将已映射文件中的脏页面写入磁盘或远程存储。如果已修改列表需要缩小，或已映射文件页面已经位于已修改列表并持续超过 5min，此线程会被唤醒。第二个已修改页面写出器的存在是必要的，因为它可以生成页面错误进而导致请求空闲页面。如果没有空闲页面并且只存在一个已修改页面写出线程，系统可能会因为等待空闲页面而陷入死锁。
   * **段取消引用 (segment dereference) 线程(MiDereferenceSegmentThread，优先级为 19)**。负责减小缓存以及页面文件的增长和收缩。举例来说，如果没有虚拟地址空间可用于换页池的增长，该线程会修剪页面缓存，借此将换页池所占用空间释放出来重新使用。
   * **零页面线程(MiZeroPageThread，优先级为0)**。负责将空闲列表中的页面清零借此提供零页面缓存，以满足将来要求零页面错误的需求。某些情况下，内存清零是通过一个速度更快的函数 MiZeroInParallel负责的。

内存管理工作是通过一种名为“页面”的块进行的。这是因为硬件内存管理单元需要以页面为单位进行虚拟和物理地址的转换。因此从硬件层面来看，页面是最小的保护单位。运行 Windows 的处理器可支持两种页面大小:小页面(4KB)和大页面(2MB)。

使用大页面的主要优势在于，当引用同一个大页面中的其他数据时地址转换的速度更快。这是因为在首次引用一个大页面中的任何一个字节时，都会导致硬件的转换旁视缓冲区(Translation Look-aside Buffer，TLB)将必要信息保存在自己的缓存中进而在引用同一个大页面内的其他数据时直接使用。如果使用小页面，为了覆盖相同范围的虚拟地址将需要更多 TLB 项，随着新的虚拟地址需要转换，这会增加项的回收次数因而这也就意味着如果所引用的虚拟地址不在已缓存的小页面范围内，此时将需要重新查询页表结构。TLB 是一种非常小的缓存，因此大页面可以更好地利用这种有限的资源。

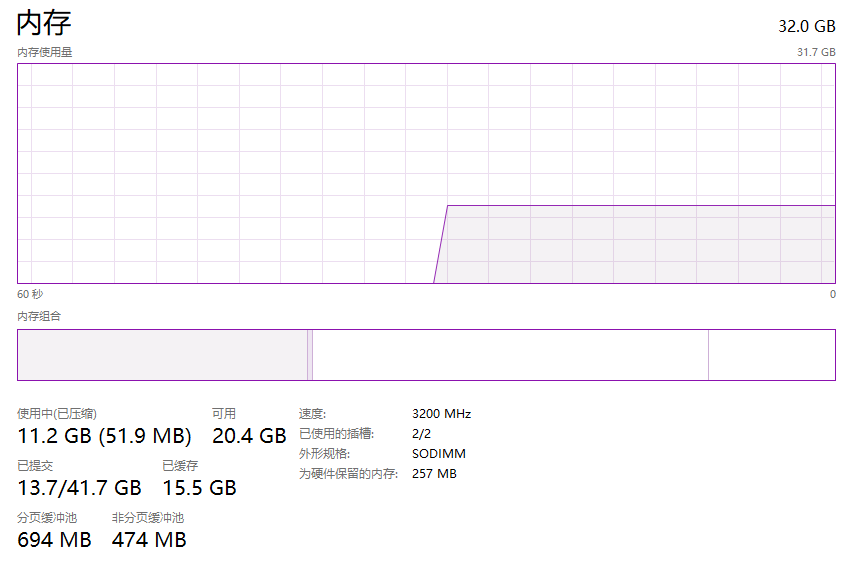
大页面

对于物理内存超过2GB的系统，为了更充分利用大页面，Windows 会利用大页面映射核心操作系统映像 (Ntoskrnl.exe 和 Hal.dll)，以及核心操作系统数据(例如非换页池的初始部分，以及用于描述每个物理内存页面状态的数据结构)。Windows还会自动将I/O空间请求(由设备驱动程序调用 MmMapIoSpace)映射到大页面，但前提是这种请求满足大页面的长度和对齐要求。此外 Windows 还允许应用程序将自己的映像、专用内存及页面文件支撑的内存区映射到大页面(可参考 VirtualAlloc、VirtualAllocEx和 VirtualAllocExNuma函数的MEM\_LARGE\_PAGES 标志。我们还可以指定将其他设备驱动程序映射到大页面为此只需要在注册表 HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management 键下添加一个多字符串注册表值 LargePageDrivers，并分别以 Null 结尾的字符串形式指定驱动程序的名称即可。

内存也始终是不可换页的，因为页面文件系统不支持大页面。由于内存不可换页，调用方必须具备SeLockMemoryPrivilege 特权才能使用大页面进行分配。

大页面机制也存在一个副作用。每个页面(无论巨型页面(1GB)、大页面或小页面)都必须对应一个应用于整个页面的保护。这是因为硬件内存保护是以页面为基础的。如果怀疑遇到内核代码出错的问题，请启用驱动程序验证器 ，这样即可禁止使用大页面。

查看内存使用情况



下表对上图中的信息进行了说明

|  |  |
| --- | --- |
| 任务管理器数值 | 定义 |
| 内存使用量图表 | 该图表的线条高度代表正被 Windows 使用的物理内存数量(无法通过性能计数器获知)线条上方区域等于界面下方的“可用”值。该图表的总高度等于图表右上角显示的内存总数，这代表可被操作系统使用的物理内存总量，其中不包含 BIOS影子页面(shadow page)、设备内存等 |
| 内存组合 | 详细列出了使用中、备用、已修改和可用内存之间的关系 |
| 物理内存总数(图表右上) | 代表可被Windows使用的物理内存量 |
| 使用中(已压缩) | 代表当前正在使用的物理内存数量，已压缩物理内存量会显示在括号中。将鼠标指针悬停在内存组合的第一部分上会显示通过压缩节约的内存数量 |
| 已缓存 | 显示了性能计数器 Memory 类别中如下计数器的数值总和:Cache Bytes、Modified Page List Bytes、Standby Cache Core Bytes、Standby Cache Normal Priority Bytes 以 Standby Cache Reserve Bytes |
| 可用 | 代表可以立即被操作系统、进程和驱动程序使用的内存数量。该值等同于备用、可用以及零页面列表的大小总和 |
| 空闲 | 代表空闲和零页面列表的大小。要查看该信息，请将鼠标指针悬停在“内存组合”图表的最右侧(前提是有足够多的空闲内存区域可供鼠标指针悬停) |
| 已提交 | 这两个数字分别等于性能计数器Committed Bytes和Commit Limit的值 |
| 分页缓冲池 | 这是分页池的总大小，包括空闲和已分配区域 |
| 非分页缓冲池 | 这是非分页池的总大小，包括空闲和已分配区域 |

内部同步

与Windows 执行体的所有其他组件类似，内存管理器也是完全可重入 (reentrant)的。并可支持在多处理器系统中并发执行。也就是说，它可以让两个线程以互不破坏对方数报的方式获取资源。为了实现完全可重入这一目标，内存管理器使用了多种不同的内部同机制，例如自旋锁和互锁指令，借此控制对其自有内部数据结构的访问。

内存管理器必须进行同步访问的部分系统范围内的资源如下。

1. 系统虚拟地址空间中动态分配的部分
2. 系统工作集。
3. 内核内存池。
4. 已加载驱动程序列
5. 页面文件列表。
6. 物理内存列表
7. 映像基址随机化地址空间布局随机化 (Address Space Layout RandomizationASLR)结构。
8. 页面帧编号(Page Frame Number，PFN) 数据库中的每个项

面向每个进程的内存管理数据结构则需要同步下列内容。

1. 工作集锁。更改工作集列表时需要这个锁。
2. 地址空间锁。更改地址空间时需要这个锁