4.3 内存分配与回收机制（一）

西安邮电大学

当我们说一个进程在执行时在说什么？

从操作系统的角度看，一个进程最关键的特征是拥有独立的虚拟地址空间。

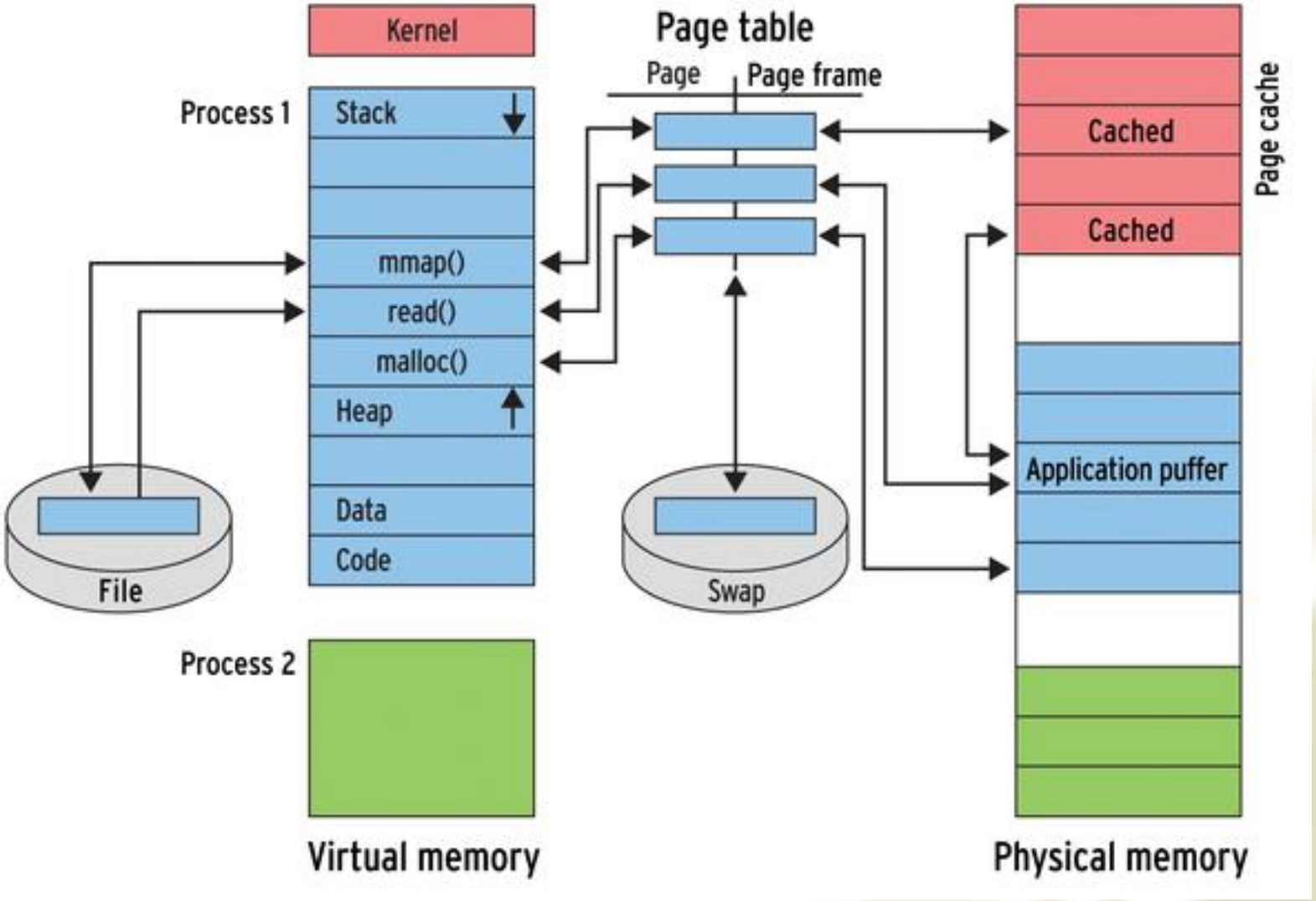
创建并执行一个进程通常需要执行步骤如下：

1、建立可执行文件与虚拟地址空间的映射

当执行一个程序时，加载器读取可执行文件（ELF）的头，建立虚拟空间和可执行文件的映射，调用的是do_mmap()函数，同时虚拟地址空间所需的数据结构mm_struct和vm_area_struct也填充相应的值,如图，左边虚拟内存部分

2、将指令寄存器设置为可执行文件入口，并启动运行。

当我们说一个进程在执行时在说什么？



当我们说一个程序在装载时在说什么？

当我们说一个程序在装载时在说什么？在上述步骤后，执行文件的指令和数据加载进内存，但并没有真正的装入物理内存，只是通过ELF文件头部信息建立起可执行文件与虚拟地址空间的映射关系而已，真正加载过程将在发生缺页异常处理时才进行，装载执行过程如下：

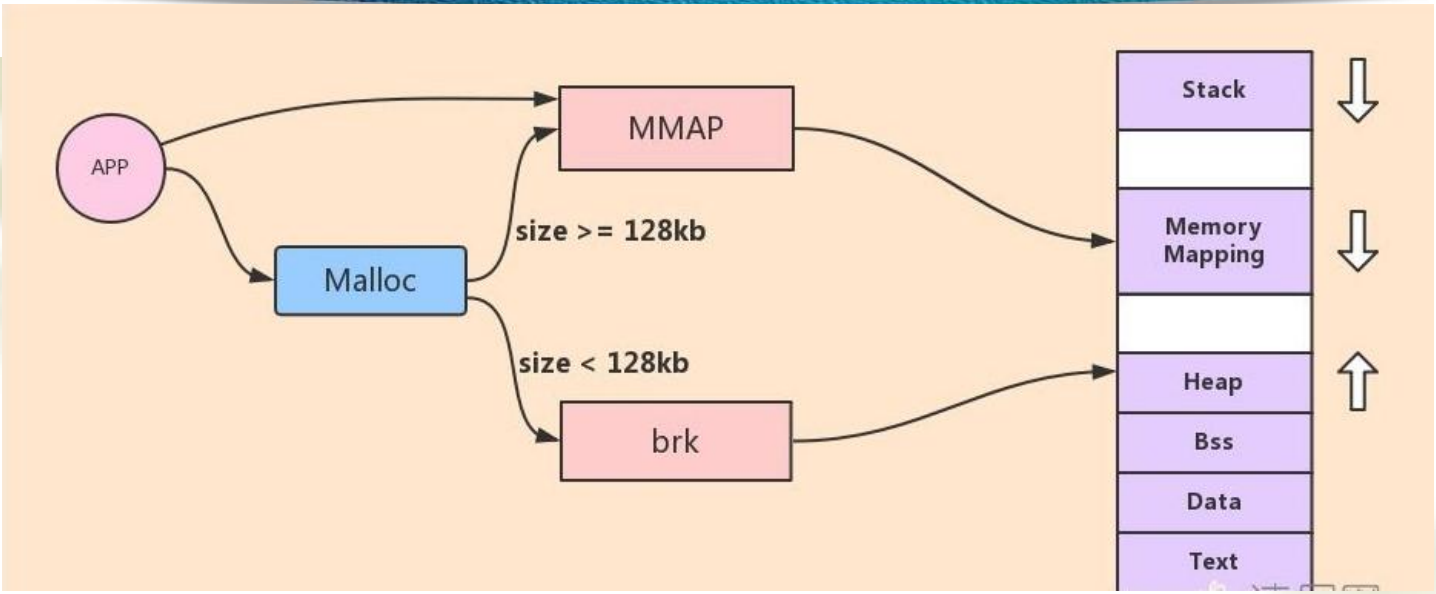
1、内核根据上面建立的映射关系，找到所需的内容在可执行文件中的位置。

2、分配一个物理内存页面，并将可执行文件内容装载到该内存页中。

3、建立该物理页面和虚拟地址空间的映射关系，也就是说填充页表，然后把控制权交换给进程。

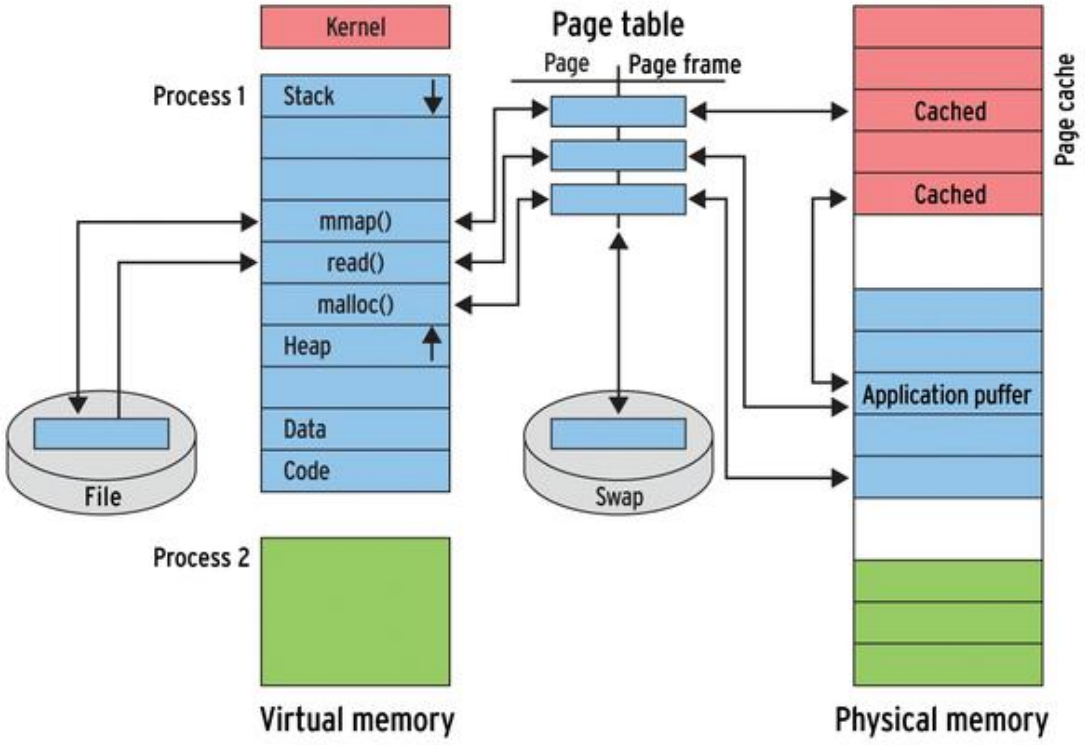
这其中涉缺页异常处理机制已经介绍过，接下来介绍物理内存的分配与回收。

当我们调用malloc()时，内核做什么



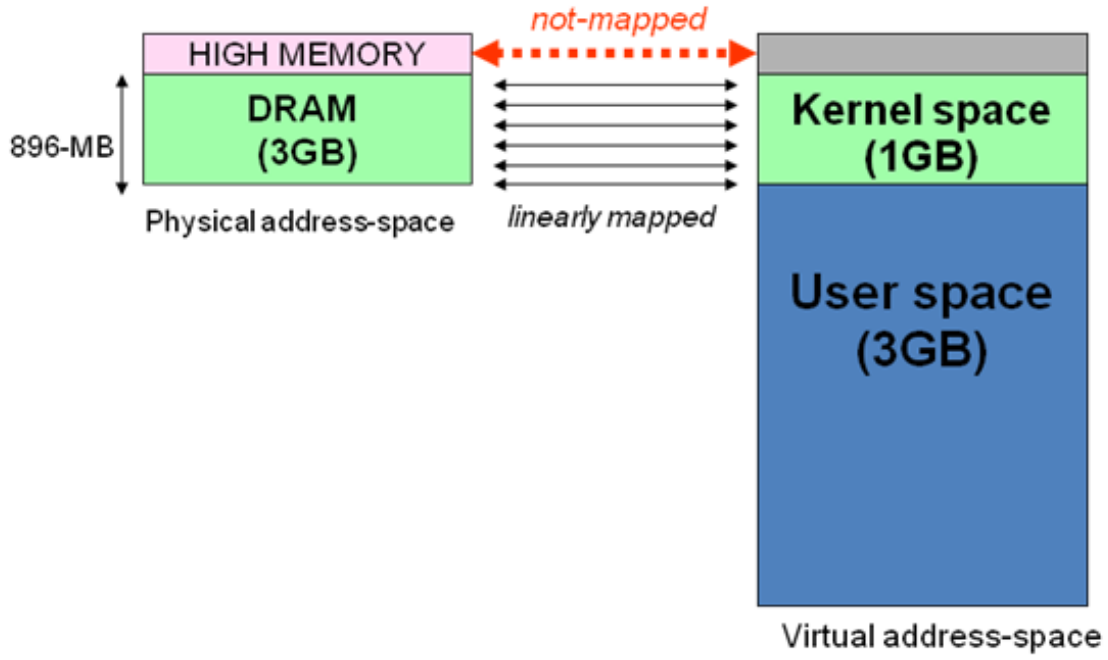
我们还要看一下，当我们在用户程序申请内存时调用malloc()，内核做了什么，实际上，它负责为进程动态的申请一块内存，操作系统从堆中分配一块内存，并把首地址返回给用户。malloc()申请内存大小不一样，最终调用的系统调用也不一样，如图所示，那么，内核是不是立即为进程分配了物理内存？答案是否定的，依然是通过请页机制。

物理内存管理



请页机制可以为进程请求物理内存，那么物理内存在内核中究竟是如何管理和分配的？接下来将从物理内存的逻辑模型开始说起。

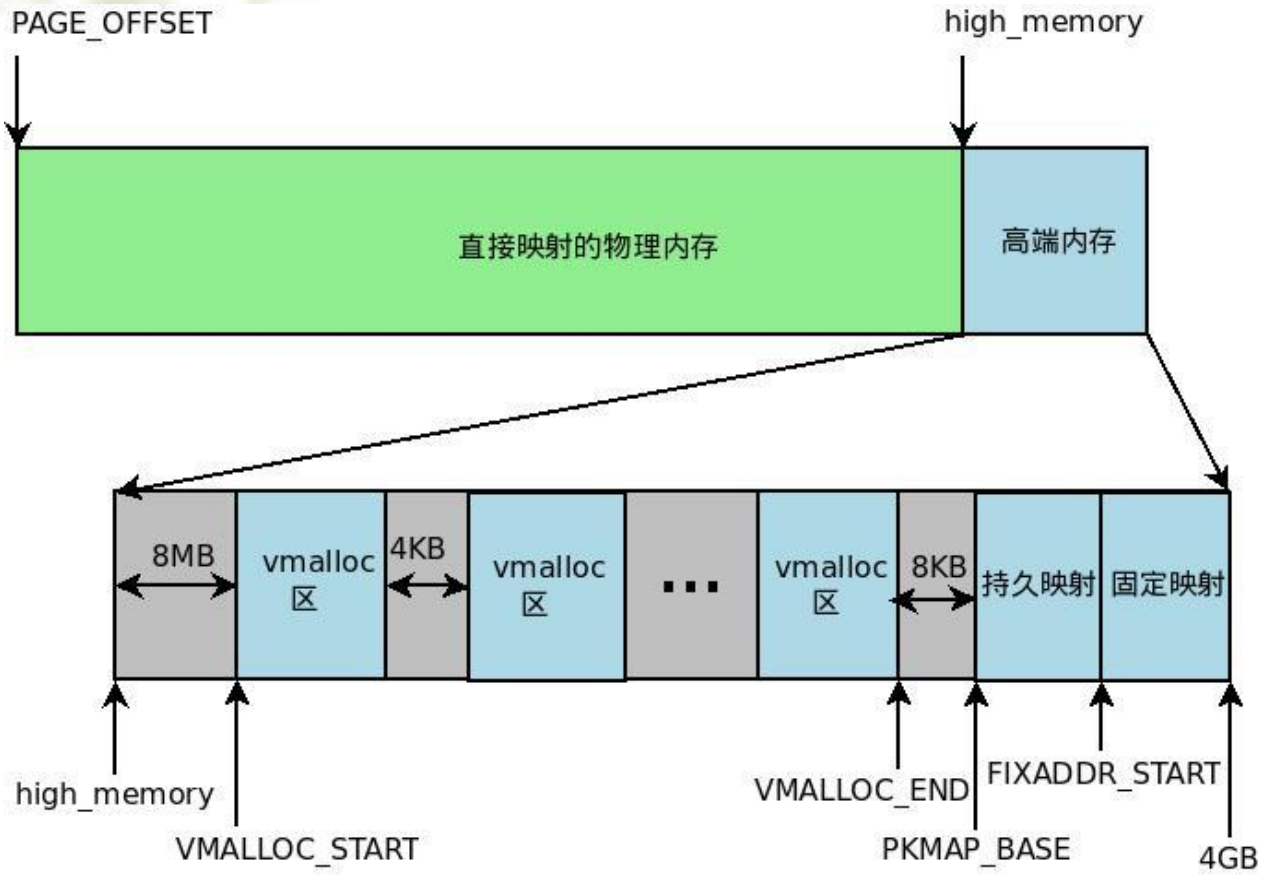
内核空间的划分



首先，我们看一下内核空间的划分。在X86-32体系架构上，内核空间的地址范围是PAGE_OFFSET（3GB）到4GB。内核空间的第一部分试图将系统的所有物理内存线性地映射到虚拟地址空间中，但最多只能映射high_memory（默认为896M）大小的物理内存。大于high_memory的物理内存将映射到内核空间的后部分。

内核空间的划分图示

按照这样的映射规则，OM到high_memory的物理内存称为低端内存，大于high_memory的物理内存称为高端内存。



从图中可以看出，内核采用了三种机制将高端内存映射到内核空间：

1. 永久内核映射
2. 固定映射
3. vmalloc机制。

内核虚拟地址和物理地址的转换

那么，内核虚拟地址和物理地址如何进行转换？

内核为线性映射的内存区提供物理地址和虚拟地址的转换函数：

__pa(vaddr)：返回虚拟地址vaddr对应的物理地址。

__va(paddr)：返回物理地址paddr对应的虚拟地址。

内核地址空间是从PAGE_OFFSET开始的，因此上述两个地址转换函数的源码实现如下：

```
#define __pa(x) ((unsigned long)(x)-PAGE_OFFSET)
#define __va(x) ((void *)((unsigned
    long)(x)+PAGE_OFFSET))
```

从这两个宏定义可以看出，这完全是一种线程关系。

物理内存管理机制

基于物理内存在内核空间中的映射原理，物理内存的管理方式也有所不同。内核中物理内存的管理机制主要有以下四种：

伙伴算法：负责大块连续物理内存的分配和释放，以页框为基本单位。该机制可以避免外部碎片。

per-CPU页框高速缓存：内核经常请求和释放单个页框，该缓存包含预先分配的页框，用于满足本地CPU发出的单一页框请求。

slab缓存：负责小块物理内存的分配，并且它也作为高速缓存，主要针对内核中经常分配并释放的对象。

vmalloc机制：vmalloc机制使得内核通过连续的线性地址来访问非连续的物理页框，这样可以最大限度的使用高端物理内存。

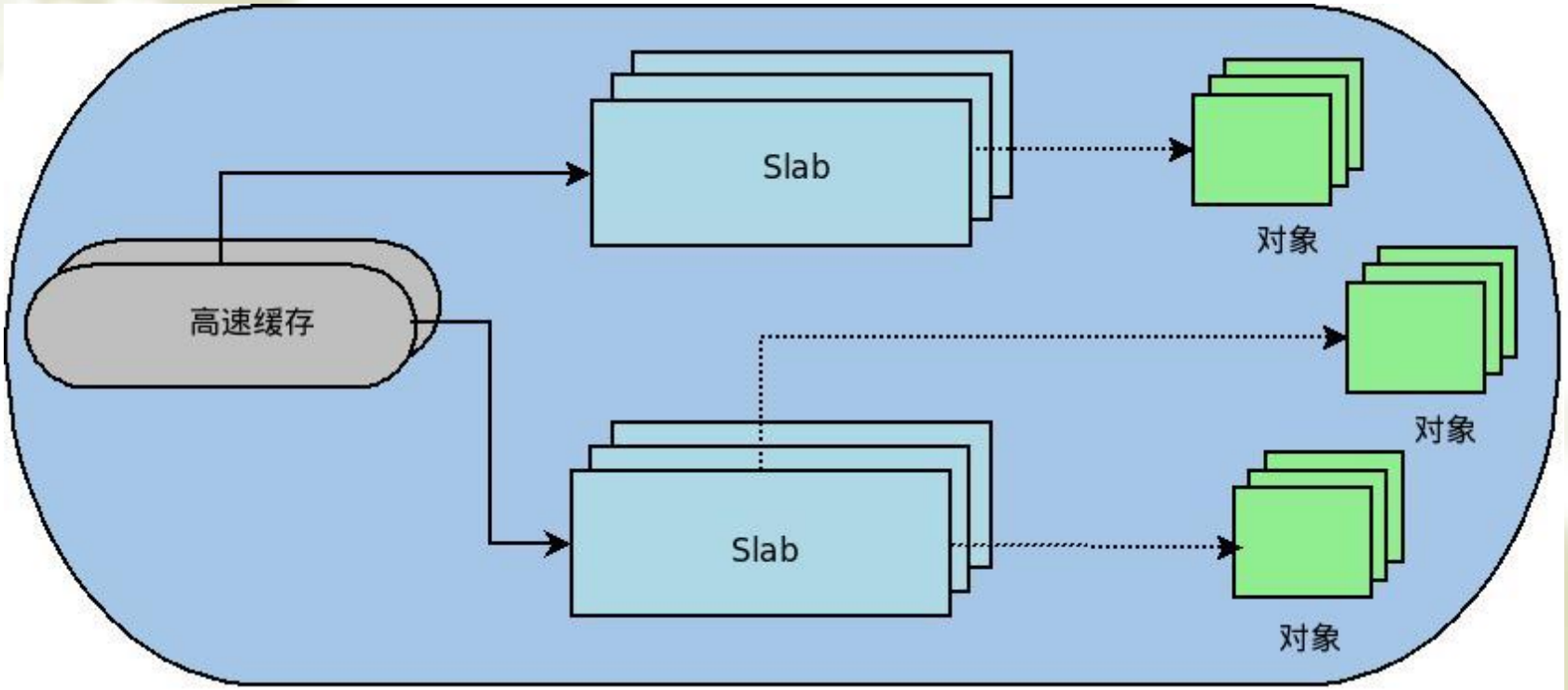
物理内存管理机制

Slab 分配机制—分配小内存

伙伴算法负责大块连续物理内存的分配和释放，以页框为基本单位，将在下一讲介绍，那么分配小块内存怎么办呢？slab分配器就是为此而提出的。slab 分配器最初是为了解决物理内存的内部碎片而提出的，它将内核中常用的数据结构看做对象。slab分配器为每一种对象建立高速缓存。内核对该对象的分配和释放均是在这块高速缓存中操作，如图所示。

可以看到每种对象的高速缓存是由若干个slab组成，每个slab是由若干个页框组成的。虽然slab分配器可以分配比单个页框更小的内存块，但它所需的所有内存都是通过伙伴算法分配的。

Slab 分配机制—分配小内存



Slab 分配机制—分配小内存

slab通用缓存

- 通用缓存是针对一般情况，适合分配任意大小的物理内存
- 其接口为kmalloc()

slab专用缓存

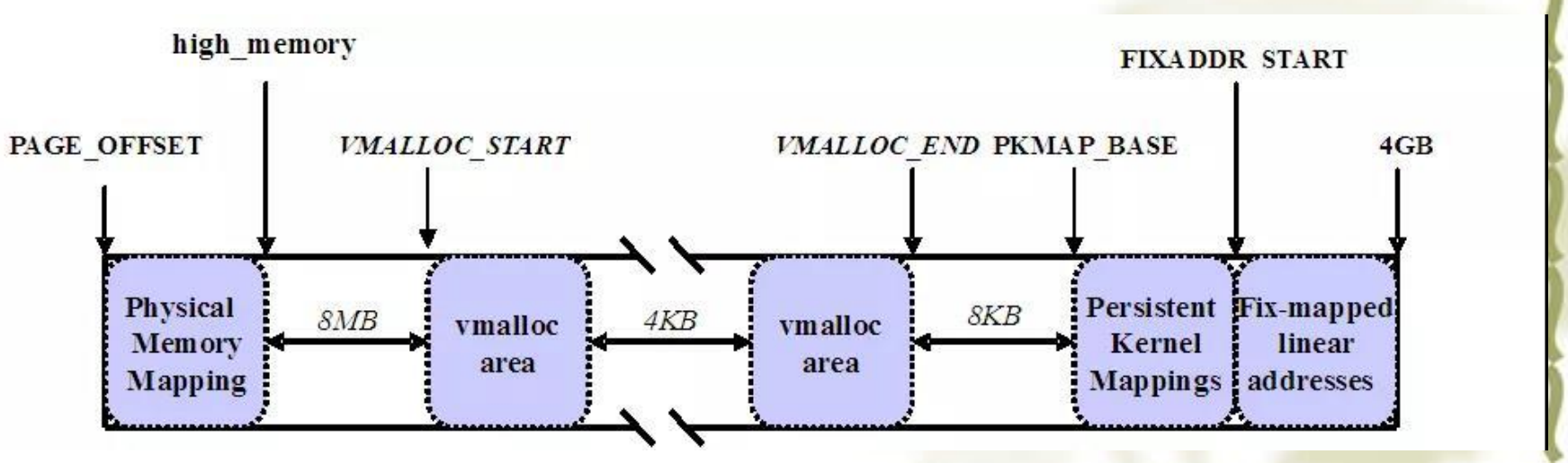
- 专用缓存是对特定的对象。
- 比如为进程控制块创建高速缓存task_struct_cachep

内核空间非连续内存区的分配

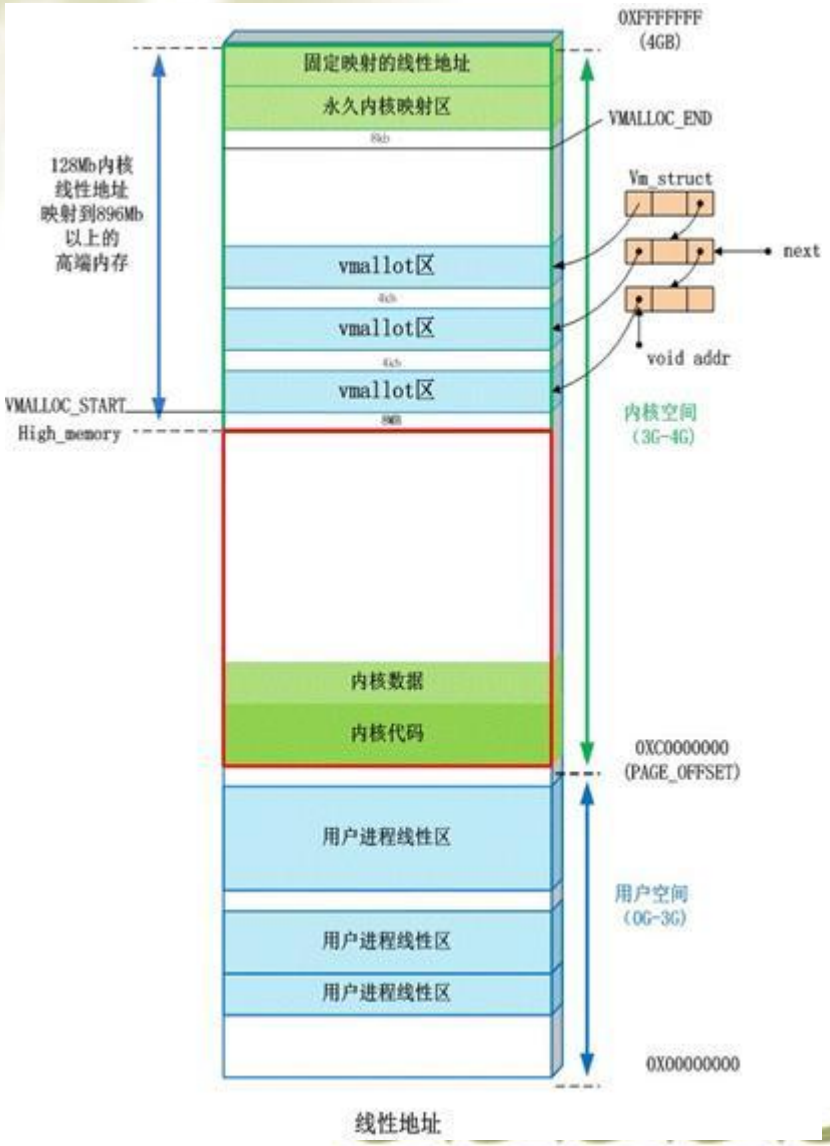
非连续内存处于3G到4G之间的内核空间的高端内存区。

我们知道物理上连续的映射对内核是最好的，但不是总能成功。在分配一大块内存时，可能无法找到连续的内存块。内核使用vmalloc（）接口函数，来分配在虚拟内存中连续但在物理内存中不一定连续的内存。

使用vmalloc（）最好的实例是为内核模块分配内存，因为模块可能在任何时候加载，如果模块数据较多，那么无法保证有足够的连续内存可用。



vmalloc()与 kmalloc()之区别



vmalloc()与 kmalloc()都可用于内核空间分配内存。

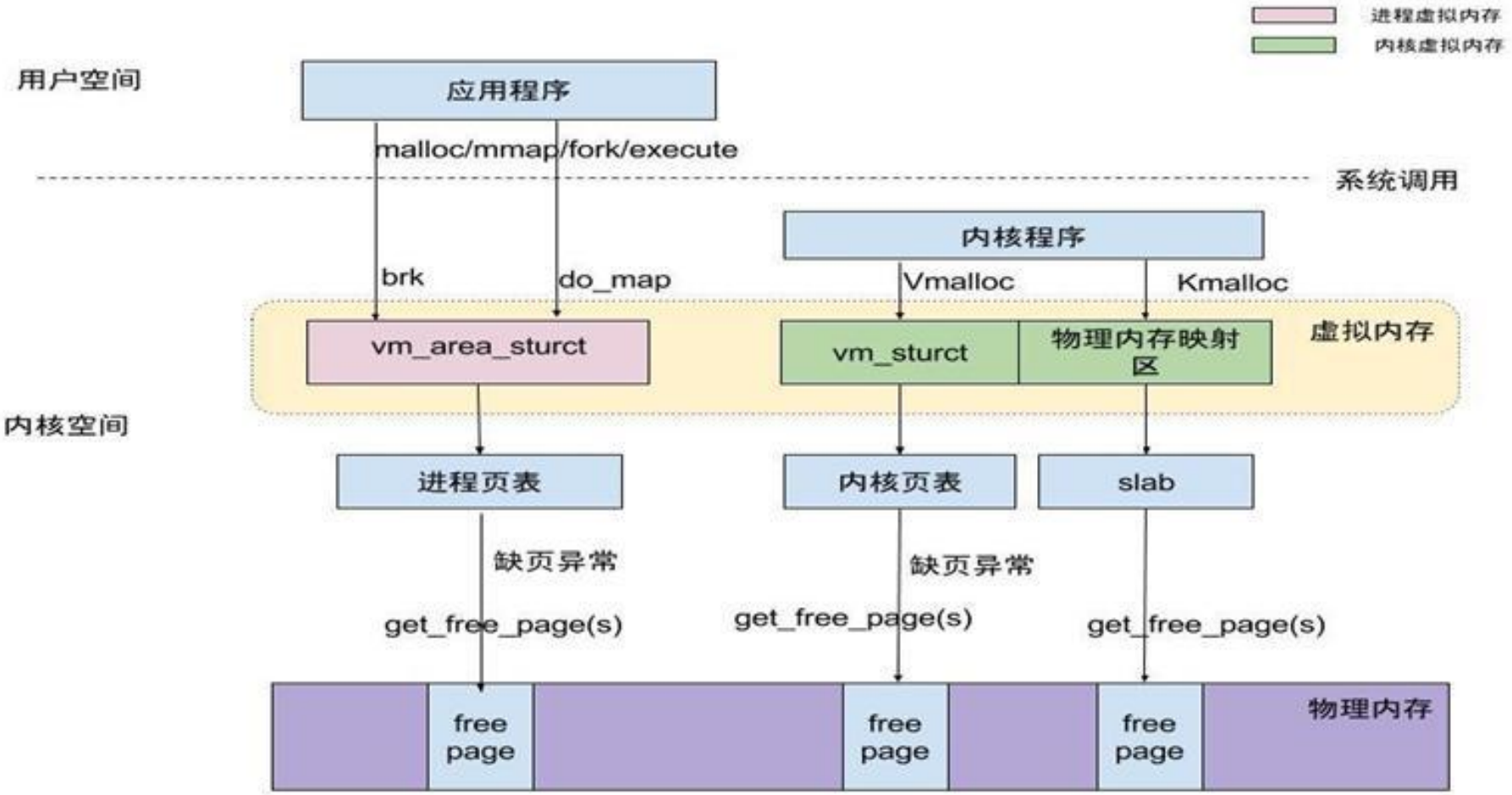
kmalloc()分配的内存处于3GB～high_memory之间，这段内核空间与物理内存的映射一一对应，而vmalloc()分配的内存在VMALLOC_START～VMALLOC_END之间，这段非连续内存区映射到物理内存也可能是非连续的。

vmalloc() 分配的物理地址无需连续，而kmalloc() 确保页在物理上是连续的。

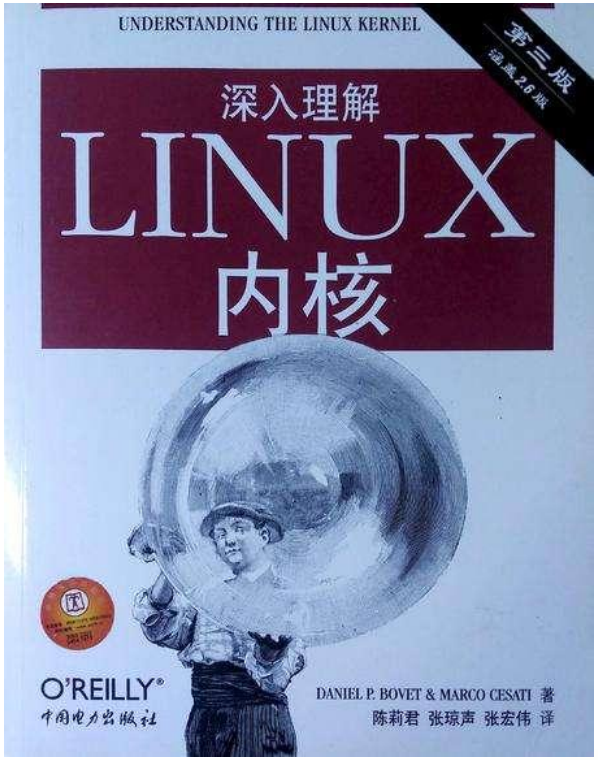
总结：当我们申请内存时内核到底做了什么？

从用户进程发出内存分配请求，到内核最终分配物理内存，这中间内核要做大量的工作，这一讲概要介绍了vmalloc()和kmalloc(),但它们最终都要调用伙伴算法，通过get_free_page()内核函数获得物理内存，下一讲将对伙伴系统进行介绍。

总结：当我们申请内存时内核到底做了什么？



参考资料



深入理解Linux内核 第三版第八章

带着思考离开

1. 用户空间（进程）是否有高端内存概念？
2. 64位内核中有高端内存吗？
3. 在32位和64位系统上，用户进程能访问多少物理内存？内核代码能访问多少物理内存？

谢谢大家！

THANK YOU