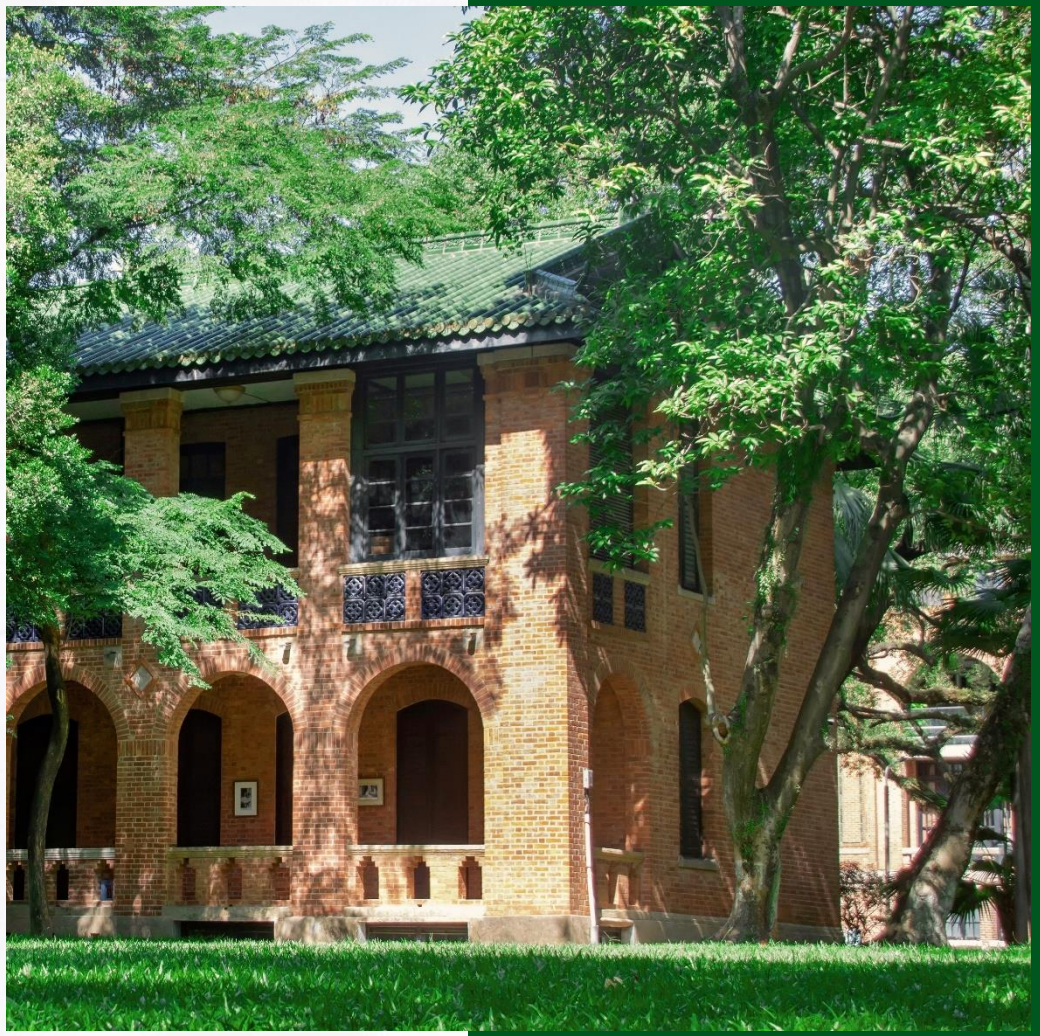
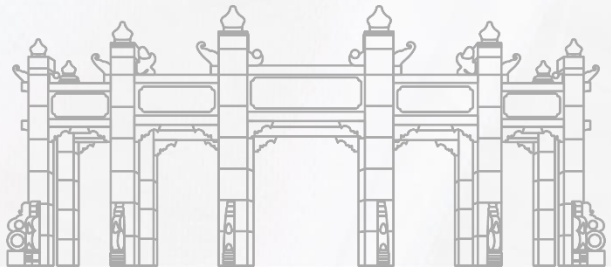
中山大學
SUN YAT-SEN UNIVERSITY

鸿蒙操作系统
虚拟内存





HarmonyOS的虚拟内存管理

1. 虚拟内存使应用程序不必管理的内存空间，能够在进程之间建立共享内存，同时在概念上使用比物理上的可用内存更多的内存。
2. 虚拟内存的特性依赖于硬件提供的MMU，即内存管理单元。
3. 内存管理单元（MMU），是一种计算机硬件单元，它控制了所有的内存访问，主要执行从虚拟内存地址到物理地址的转换。

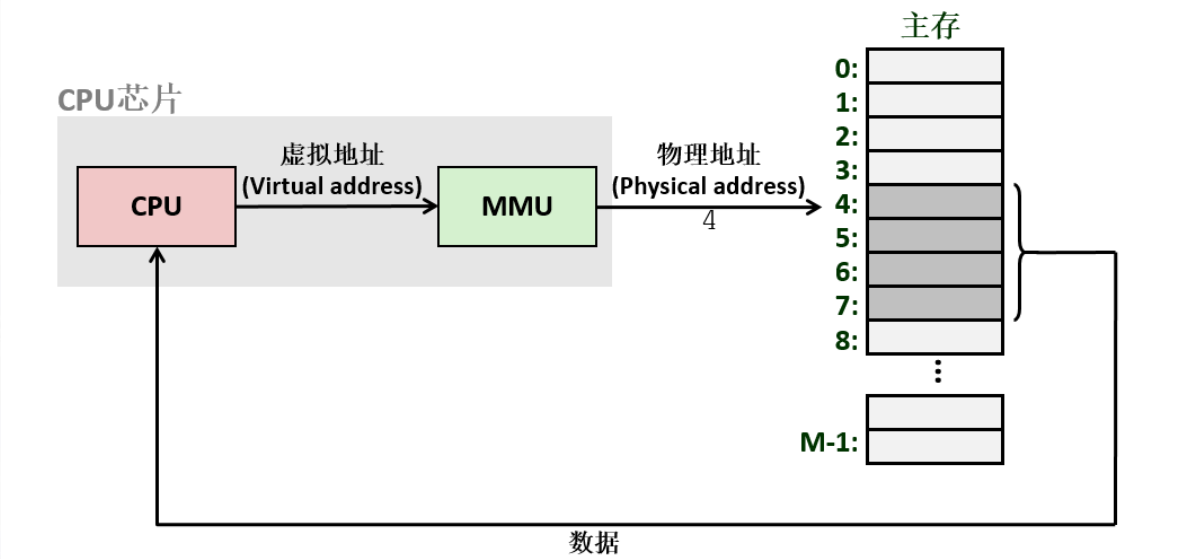


图7-1 CPU、MMU和内存之间的关系

HarmonyOS的虚拟内存管理

1. 访问虚拟内存过程中，HarmonyOS完成的工作：
   - 构建地址映射页表，包括内核页表和用户进程页表。
   - 明确页表的尺度、粒度等设置。
   - 对MMU进行配置。
   - 在进行内存分配和内存释放的过程中，维护地址映射页表。
2. 所有的内存访问都会被MMU拦截，其中快表（TLB）位于MMU中，MMU通过查表的方式来进行地址转换。

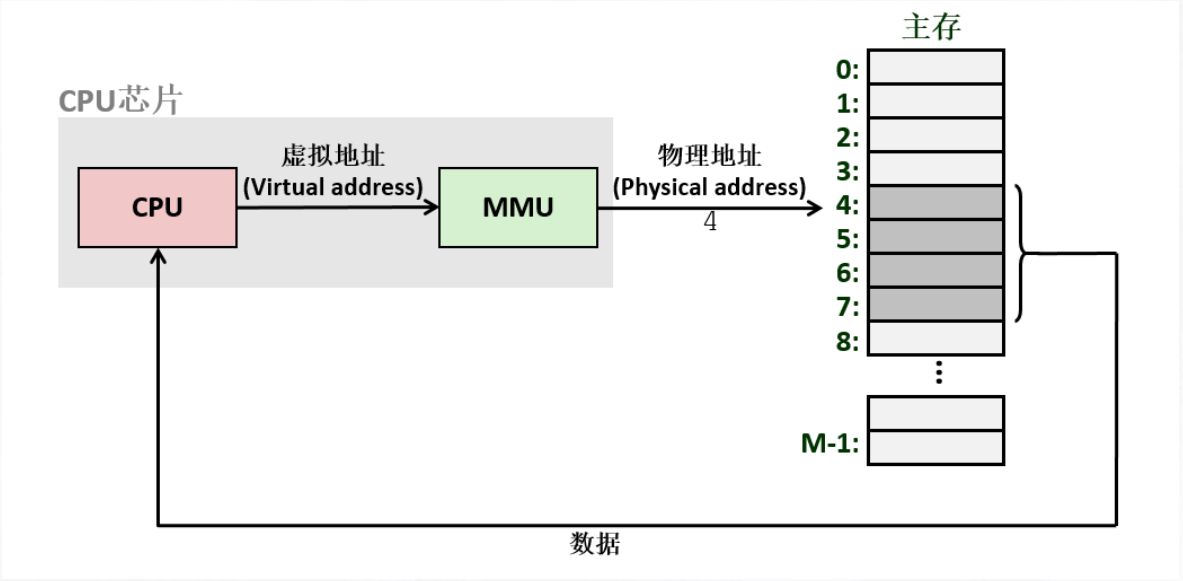


图7-1 CPU、MMU和内存之间的关系

HarmonyOS的虚拟内存管理

虚拟内存管理中，虚拟地址空间是连续的，但是其映射的物理内存并不一定是连续的。CPU访问虚拟地址空间的代码或数据时存在两种情况：

1. CPU访问的虚拟地址所在的页，如V0，已经与具体的物理页P0做映射，CPU通过找到进程对应的页表条目，根据页表条目中的物理地址信息访问物理内存中的内容并返回。
2. CPU访问的虚拟地址所在的页，如V2，没有与具体的物理页做映射，系统会触发缺页异常，系统申请一个物理页，并把相应的信息拷贝到物理页中，并且把物理页的起始地址更新到页表条目中。此时CPU重新执行访问虚拟内存的指令便能够访问到具体的代码或数据

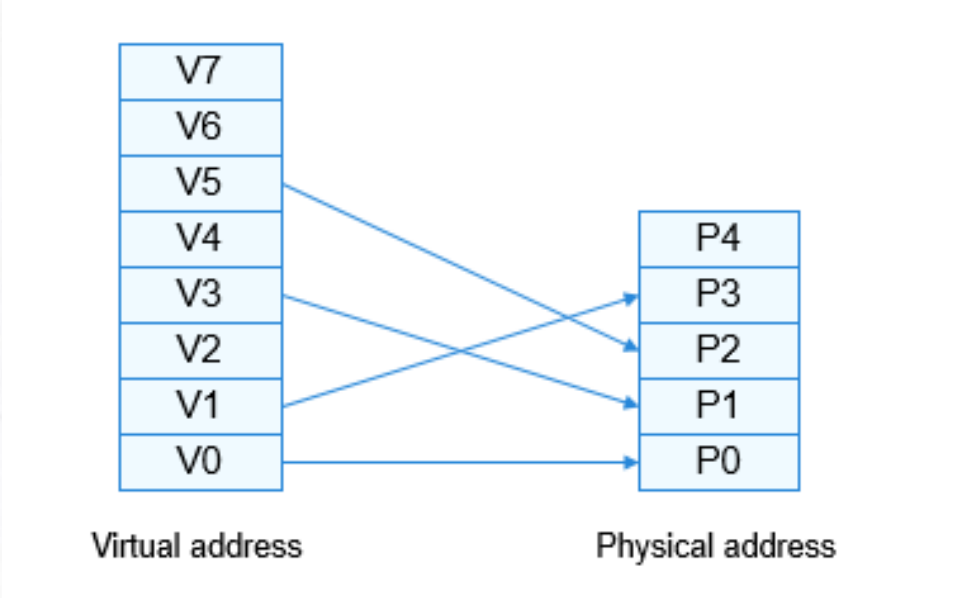


图7-2 HarmonyOS的虚拟内存映射

HarmonyOS的虚拟内存管理

HarmonyOS的虚拟内存运行过程

1. 用户程序加载启动时，会将代码段、数据段映射进虚拟内存空间，此时并没有物理页做实际的映射；
2. 程序执行时，CPU访问虚拟地址，通过MMU查找是否有对应的物理内存，若该虚拟地址无对应的物理地址则触发缺页异常，内核申请物理内存并将虚实映射关系及对应的属性配置信息写进页表，并把页表条目缓存至TLB，接着CPU可直接通过转换关系访问实际的物理内存；
3. 若CPU访问已缓存至TLB的页表条目，无需再访问保存在内存中的页表，可加快查找速度。

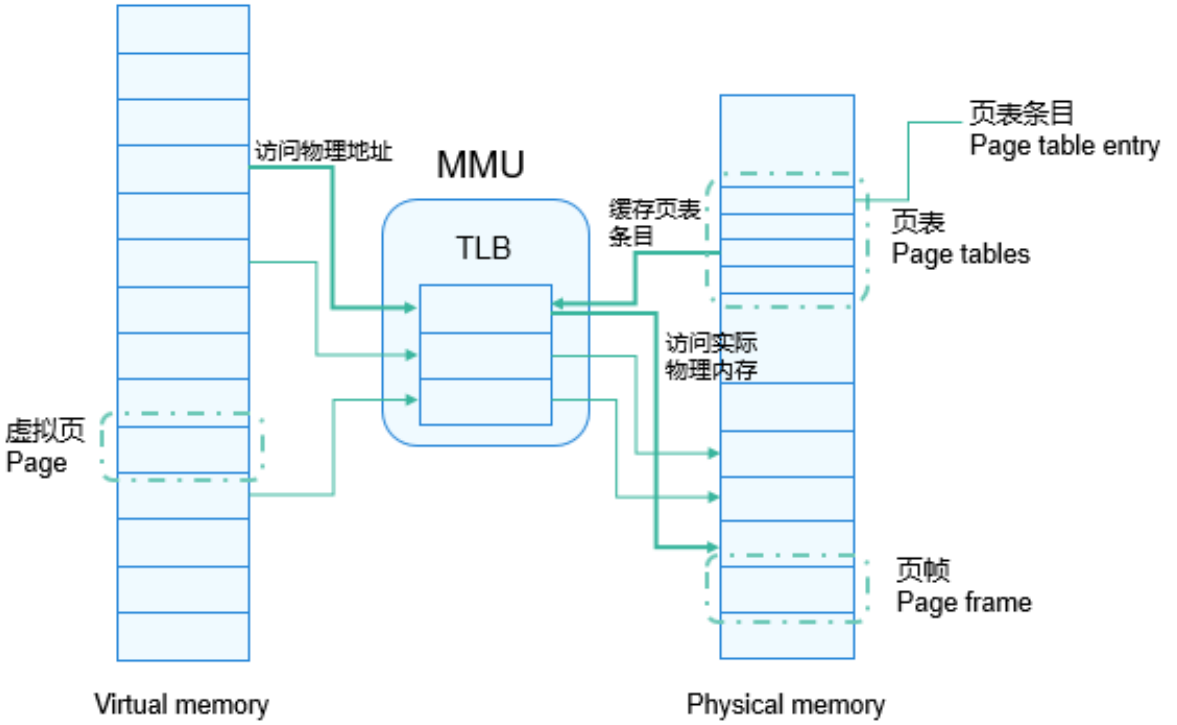


图7-3 HarmonyOS的虚拟内存运行过程

HarmonyOS的虚拟内存技术

1. LiteOS-m不支持虚拟内存技术。
2. LiteOS-a是一个32位的操作系统，因此虚拟地址空间的总规模是32位的地址空间，即4G。
3. LiteOS-a将这4G进行拆分，其中3G留给内核页表，其余1G留给进程使用。3G的内核页表会被所有的进程所共享。

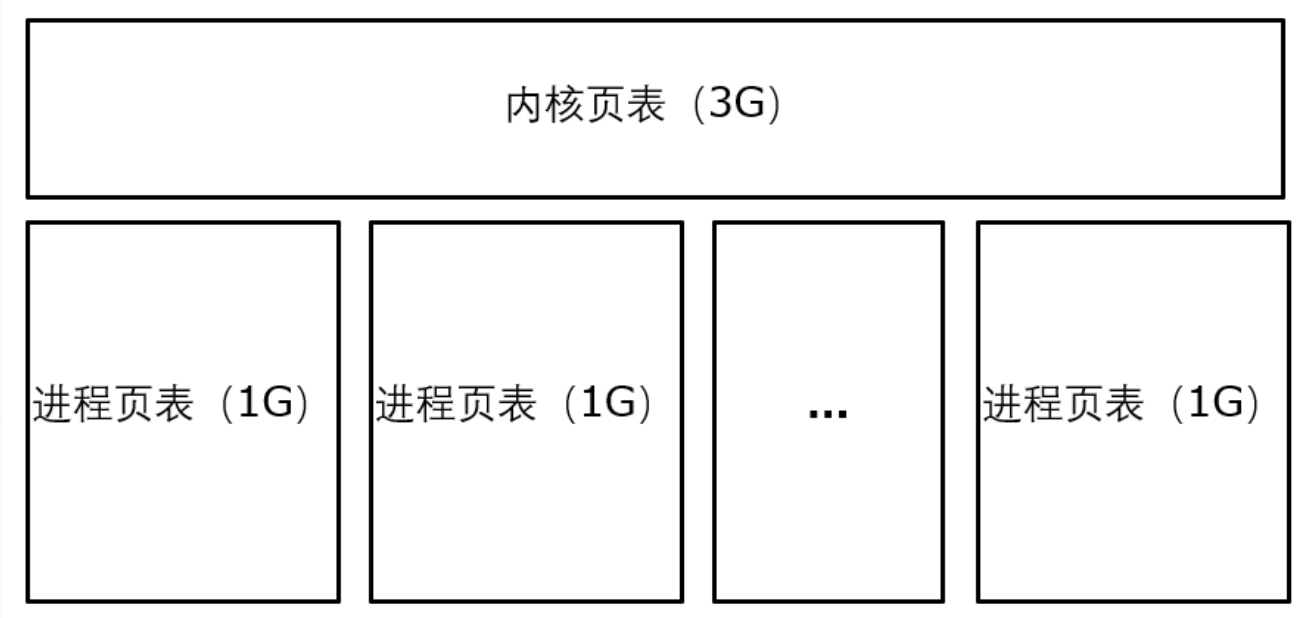


图7-4 LiteOS-a的整体页表布局

HarmonyOS的虚拟内存技术

内核页表：

1. LiteOS-a中，内核页表的初始化：
   - 初始化内核固有空间。
   - 初始化内核堆空间。
   - 初始化内核页空间控制块。
   - 建立内核映射表。
   - 初始化共享内存。
2. 内核空间分配：
   - 使用kmalloc申请的空间采用直接映射的方式。
   - 除了kmalloc使用的空间外，都是使用二级页表映射。
3. 内核级进程统一使用内核内存映射关系，所有的内核态进程共享。

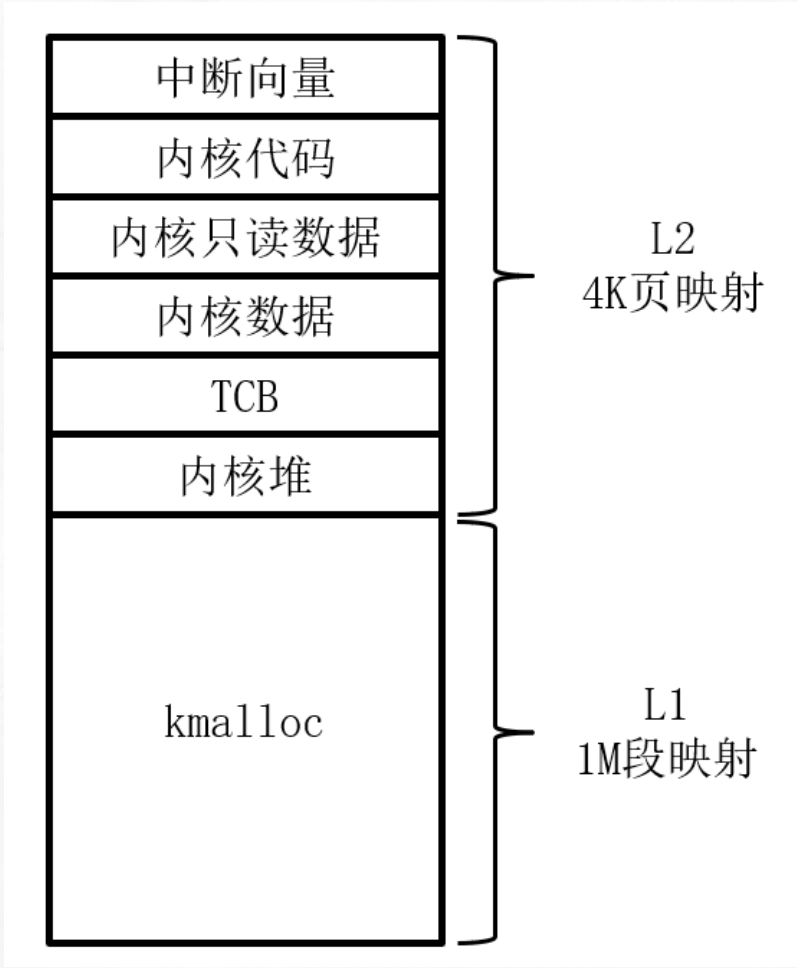


图7-5 LiteOS-a的内核空间分配

HarmonyOS的虚拟内存技术

用户态页表：

1. LiteOS-a中，用户态进程页表的初始化：
   - 申请vmSpace内存。
   - 申请连续的4K物理空间。
   - 初始化vmSpace。
   - 计算得到vmSpace的地址映射关系。
   - 保存数据到PCB。
2. 用户态进程页表的初始化只分配4K的空间，在进程运行的过程中，还会不断地申请新的内存。
3. 新分配的内存也要进行虚拟内存的管理并维护页表。

HarmonyOS的虚拟内存技术

kmalloc

1. kmalloc()函数类似与常见的malloc()函数， kmalloc()用于内核态的内存分配，而malloc()用于用户态。
2. kmalloc()函数在物理内存中分配一块连续的存储空间，且和malloc()函数一样，不会清除里面的原始数据，如果有足够的内存空闲空间，它的分配速度很快。
3. 内核中常用的kmalloc()函数的核心实现是slab机制。kmalloc() 申请的内存位于物理内存映射区域，而且在物理上也是连续的，它们与真实的物理地址只有一个固定的偏移，因为存在较简单的转换关系，所以对申请的内存大小有限制，不能超过128KB。

vmalloc

1. vmalloc() 函数则会在虚拟内存空间给出一块连续的内存区，但这片连续的虚拟内存在物理内存中并不一定连续。由于 vmalloc() 没有保证申请到的是连续的物理内存，因此对申请的内存大小没有限制，如果需要申请较大的内存空间就需要用此函数了。
2. 由于需要建立新的页表，所以它的开销要远远大于kmalloc。

HarmonyOS的虚拟内存技术

LiteOS-a的晚分配晚映射机制：

1. 当进程申请更多内存时，需要对物理页框进行调度，会带来一定的时间开销。同时有些进程申请完内存之后并不会立即访问或者只访问其中的一部分，如果直接为其分配完整的物理页框则可能会浪费有限的物理内存，同时带来不必要的分配开销。

2. LiteOS-a使用晚分配晚映射机制来解决上述问题。
   - 当用户进程申请更多的内存时，操作系统只会对更新对应的虚拟内存表，而不会立即为进程分配对应的物理块，因此在TLB和页表中也并不存在对应的映射关系。
   - 只有当进程访问到该申请的内存块时，操作系统通过MMU触发缺页中断，才会为该虚拟内存块分配对应的物理内存，同时建立对应的映射关系。

HarmonyOS的虚拟内存技术

LiteOS-a的晚分配晚映射机制：

- 当用户请求更多内存时，内核仅更新堆虚拟内存表并快速返回一个值给用户进程。
- 此时，实际上并没有分配新的物理内存页，页表上也没有建立对应的物理映射。
- 该过程对用户进程是透明的，用户进程可以继续向下执行，而不需要等待操作系统为其分配物理页框。

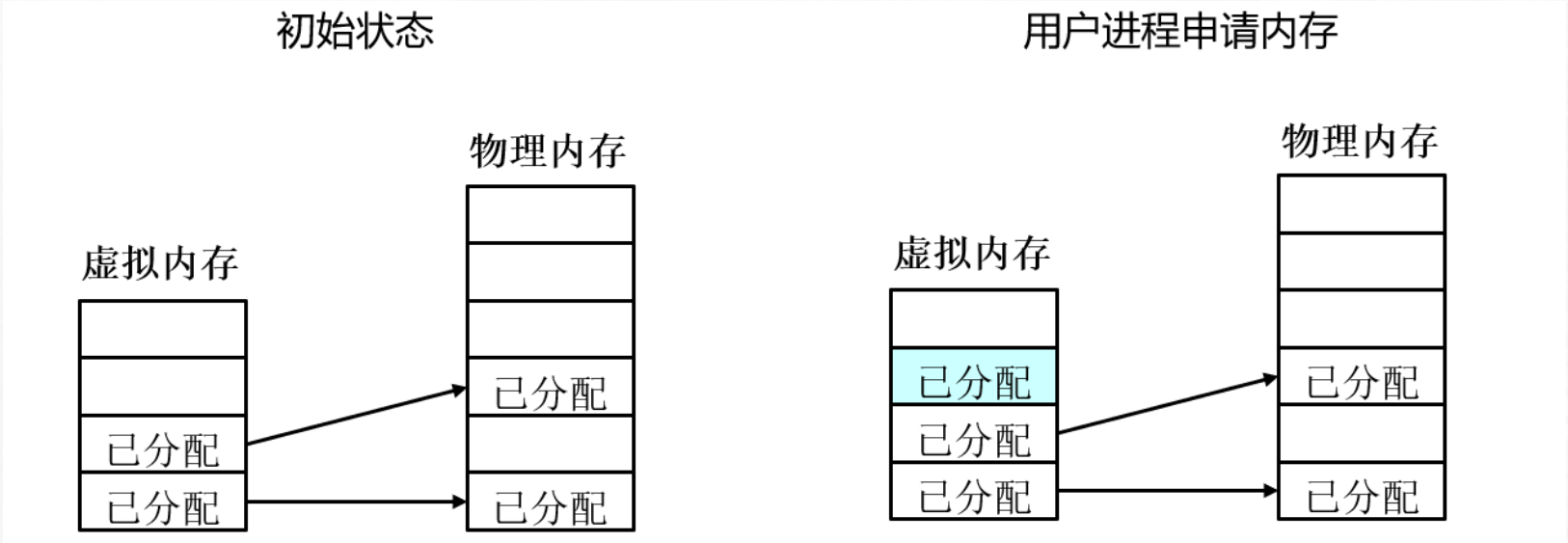


图7-6 LiteOS-a的晚分配晚映射机制

HarmonyOS的虚拟内存技术

LiteOS-a使用晚分配晚映射机制：

- 当程序访问新分配的虚拟地址所在的页面，就会触发pagefault中断，从而进入相应的中断处理程序。
- 中断处理程序查找页表，显示该页未分配，内核会先分配一块物理内存，进行虚拟内存和物理内存的映射，并更新页表。

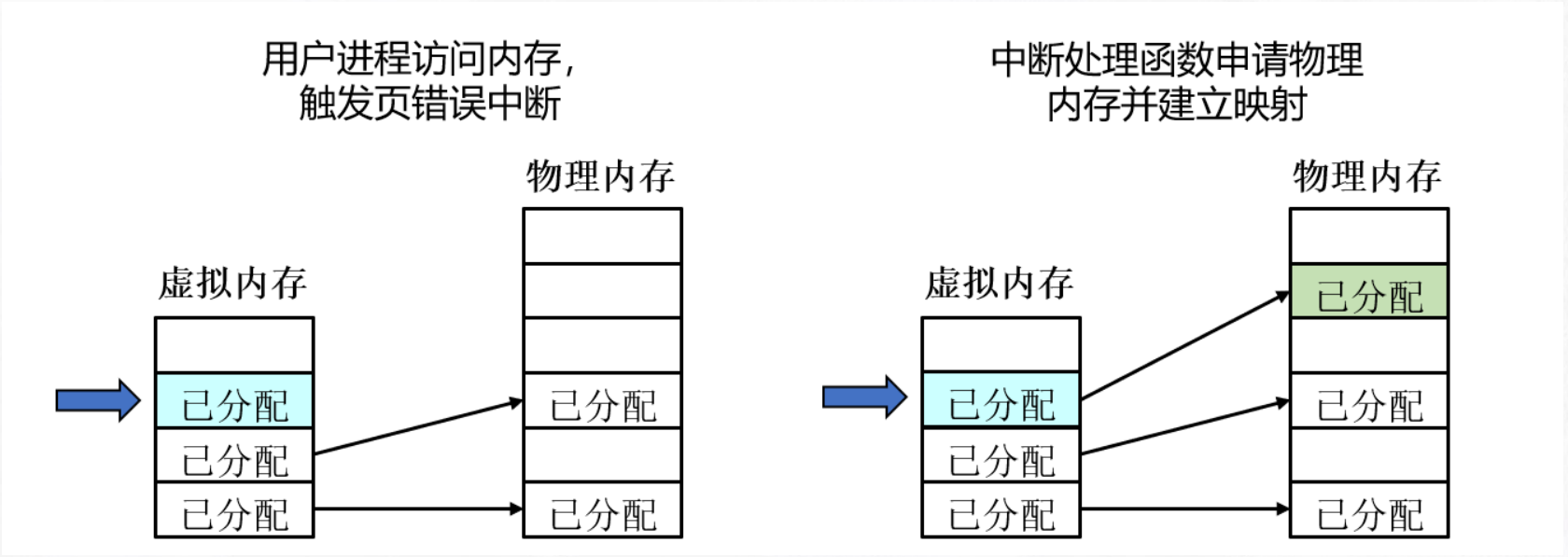


图7-6 LiteOS-a的晚分配晚映射机制

HarmonyOS的虚拟内存技术

Harmony虚拟内存操作接口

1. 虚拟地址区间region相关的操作
   - LOS_RegionFind：在进程空间内查找并返回指定地址对应的虚拟地址区间
   - LOS_RegionAlloc：申请空闲的虚拟地址区间
   - LOS_RegionSize：获取region的大小
2. vmalloc操作
   - LOS_Vmalloc：vmalloc申请内存
   - LOS_Vfree：vmalloc释放内存
   - LOS_IsVmallocAddress：判断地址是否是通过vmalloc申请的
3. 地址校验
   - LOS_IsUserAddress：判断地址是否在用户态空间
   - LOS_IsKernelAddress：判断地址是否在内核空间

- **HarmonyOS应用的生命周期**

1. 应用生命周期
   - onCreate：应用创建。
   - onDestroy：应用销毁，当应用退出时触发。
2. 页面的生命周期
   - onInit：页面初始化。
   - onReady：页面创建完成。
   - onShow：页面显示。
   - onHide：页面消失。
   - onDestroy：页面销毁。

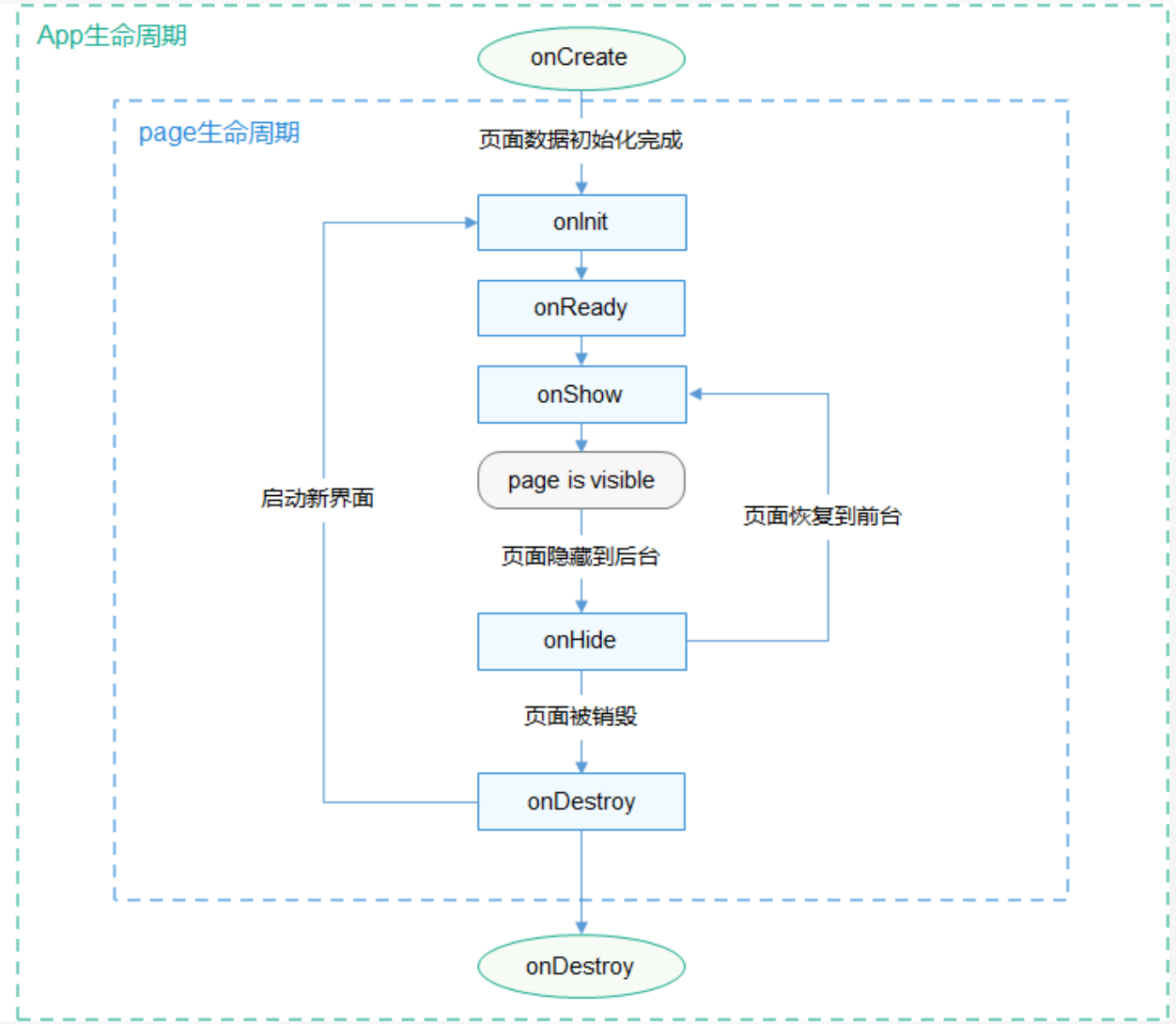


图7-7 HarmonyOS应用的生命周期

HyperHold内存管理引擎

对于内存管理，由于开源生态的不限制，导致应用开发的内存使用野蛮生长。设备长时间使用后，可回收内存越来越低。产生这个问题的原因有两个：

1. 传统内存数据冷热管理，无法感知业务特性
   - 尽管Linux内核提供了很多的内存回收机制，然而每种内存回收都会有相应的系统代价。
   - 比如，回收文件页面后，如果系统需要二次加载这部分数据，需要从底层器件Flash里面把数据读回来，这会引起Flash随机IO读的现象。
2. 传统共享式内存分配，无法感知数据重要性
   - 从内存分配角度看，现在的操作系统基本采用统一接口的分配方式，使得手机里面多个进程或多个业务会共用一块内存区域。
   - 数据回收时，会频繁出现数据搬移，以及内存震荡的现象。这个现象会加重内核管理内存的开销。

HyperHold内存管理引擎

为了解决传统Linux内核的内存问题，HarmonyOS提供了Hyperhold内存管理引擎：

1. Hyperhold内存管理引擎打通了上层系统到内核的调用栈，让内核完整感知到应用的整个生命周期，并结合应用生命周期以及周期内的数据访问特征，对每一块内存数据做合理的内存管理。

图7-8 HyperHold内存管理引擎

- **HyperHold内存管理引擎**

2. 为了降低内核管理内存的开销，HarmonyOS提出了自研的内存压缩体系，包括多线程压缩、自研的压缩算法。
3. 为了进一步扩大可用空间， HarmonyOS在Flash器件上开出了一块可交换区，结合自研的聚合换出和内存标记技术，充分利用Flash器件的性能。

图7-8 HyperHold内存管理引擎

中山大學
SUN YAT-SEN UNIVERSITY

谢谢观看

SUN YAT-SEN UNIVERSITY