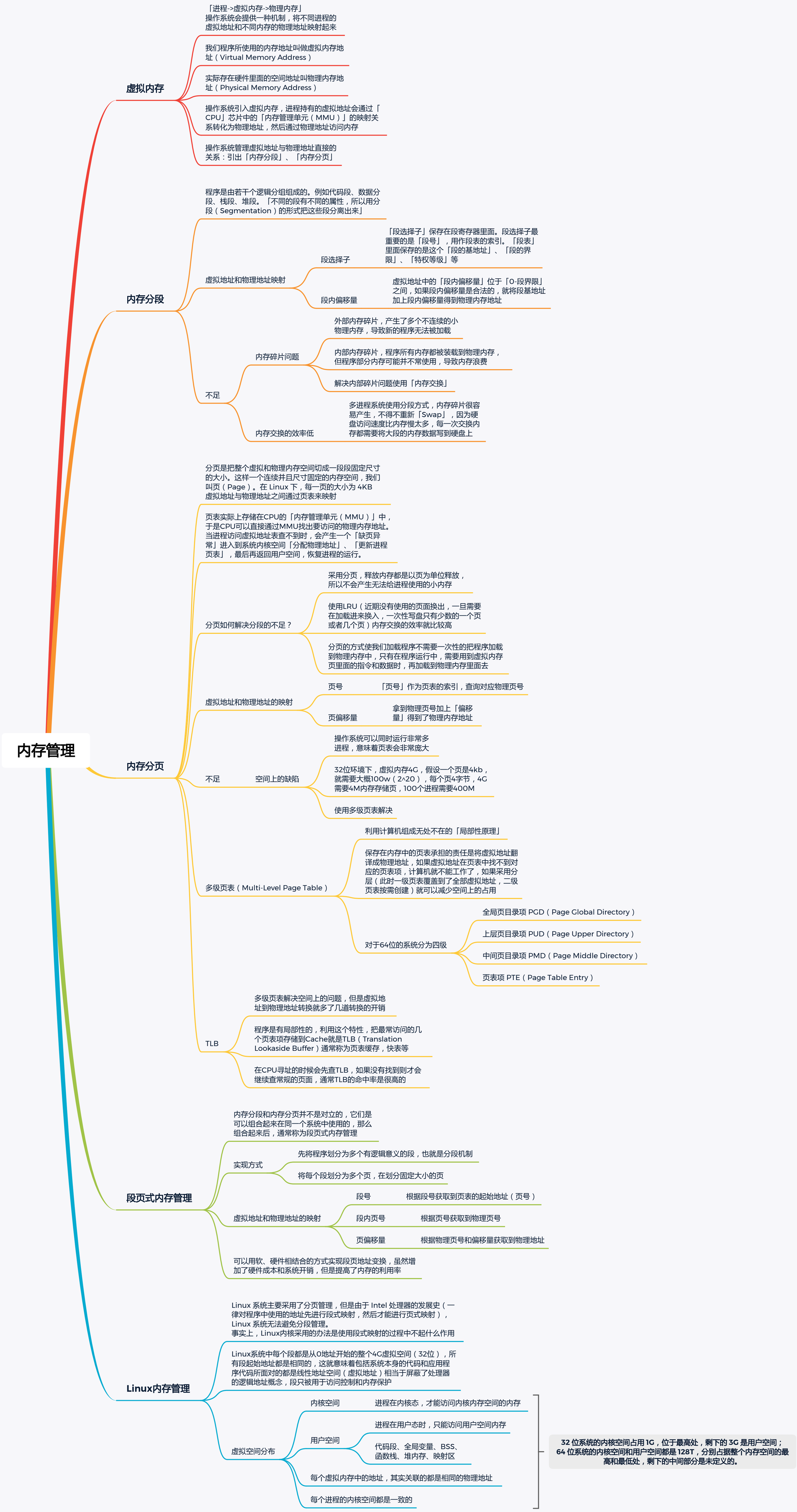
内存管理
虚拟内存
「进程->虚拟内存->物理内存」
操作系统会提供一种机制，将不同进程的
虚拟地址和不同内存的物理地址映射起来
我们程序所使用的内存地址叫做虚拟内存地
址（Virtual Memory Address）
实际存在硬件里面的空间地址叫物理内存地
址（Physical Memory Address）
操作系统引入虚拟内存，进程持有的虚拟地址会通过「
CPU」芯片中的「内存管理单元（MMU）」的映射关
系转化为物理地址，然后通过物理地址访问内存
操作系统管理虚拟地址与物理地址直接的
关系：引出「内存分段」、「内存分页」
内存分段
程序是由若干个逻辑分组组成的。例如代码段、数据分
段、栈段、堆段。「不同的段有不同的属性，所以用分
段（Segmentation）的形式把这些段分离出来」
虚拟地址和物理地址映射
段选择子
「段选择子」保存在段寄存器里面。段选择子最
重要的是「段号」，用作段表的索引。「段表」
里面保存的是这个「段的基地址」、「段的界
限」、「特权等级」等
段内偏移量
虚拟地址中的「段内偏移量」位于「0-段界限」
之间，如果段内偏移量是合法的，就将段基地址
加上段内偏移量得到物理内存地址
不足
内存碎片问题
外部内存碎片，产生了多个不连续的小
物理内存，导致新的程序无法被加载
内部内存碎片，程序所有内存都被装载到物理内存，
但程序部分内存可能并不常使用，导致内存浪费
解决内部碎片问题使用「内存交换」
内存交换的效率低
多进程系统使用分段方式，内存碎片很容
易产生，不得不重新「Swap」，因为硬
盘访问速度比内存慢太多，每一次交换内
存都需要将大段的内存数据写到硬盘上
内存分页
分页是把整个虚拟和物理内存空间切成一段段固定尺寸
的大小。这样一个连续并且尺寸固定的内存空间，我们
叫页（Page）。在 Linux 下，每一页的大小为 4KB
虚拟地址与物理地址之间通过页表来映射
页表实际上存储在CPU的「内存管理单元（MMU）」中，
于是CPU可以直接通过MMU找出要访问的物理内存地址。
当进程访问虚拟地址表查不到时，会产生一个「缺页异
常」进入到系统内核空间「分配物理地址」、「更新进程
页表」，最后再返回用户空间，恢复进程的运行。
分页如何解决分段的不足？
采用分页，释放内存都是以页为单位释放，
所以不会产生无法给进程使用的小内存
使用LRU（近期没有使用的页面换出，一旦需要
在加载进来换入，一次性写盘只有少数的一个页
或者几个页）内存交换的效率就比较高
分页的方式使我们加载程序不需要一次性的把程序加载
到物理内存中，只有在程序运行中，需要用到虚拟内存
页里面的指令和数据时，再加载到物理内存里面去
虚拟地址和物理地址的映射
页号
「页号」作为页表的索引，查询对应物理页号
页偏移量
拿到物理页号加上「偏移
量」得到了物理内存地址
不足
空间上的缺陷
操作系统可以同时运行非常多
进程，意味着页表会非常庞大
32位环境下，虚拟内存4G，假设一个页是4kb，
就需要大概100w（2^20），每个页4字节，4G
需要4M内存存储页，100个进程需要400M
使用多级页表解决
多级页表（Multi-Level Page Table）
利用计算机组成无处不在的「局部性原理」
保存在内存中的页表承担的责任是将虚拟地址翻
译成物理地址，如果虚拟地址在页表中找不到对
应的页表项，计算机就不能工作了，如果采用分
层（此时一级页表覆盖到了全部虚拟地址，二级
页表按需创建）就可以减少空间上的占用
对于64位的系统分为四级
全局页目录项 PGD（Page Global Directory）
上层页目录项 PUD（Page Upper Directory）
中间页目录项 PMD（Page Middle Directory）
页表项 PTE（Page Table Entry）
TLB
多级页表解决空间上的问题，但是虚拟地
址到物理地址转换就多了几道转换的开销
程序是有局部性的，利用这个特性，把最常访问的几
个页表项存储到Cache就是TLB（Translation
Lookaside Buffer）通常称为页表缓存，快表等
在CPU寻址的时候会先查TLB，如果没有找到则才会
继续查常规的页面，通常TLB的命中率是很高的
段页式内存管理
内存分段和内存分页并不是对立的，它们是
可以组合起来在同一个系统中使用的，那么
组合起来后，通常称为段页式内存管理
实现方式
先将程序划分为多个有逻辑意义的段，也就是分段机制
将每个段划分为多个页，在划分固定大小的页
虚拟地址和物理地址的映射
段号
根据段号获取到页表的起始地址（页号）
段内页号
根据页号获取到物理页号
页偏移量
根据物理页号和偏移量获取到物理地址
可以用软、硬件相结合的方式实现段页地址变换，虽然增
加了硬件成本和系统开销，但是提高了内存的利用率
Linux内存管理
Linux 系统主要采用了分页管理，但是由于 Intel 处理器的发展史（一
律对程序中使用的地址先进行段式映射，然后才能进行页式映射），
Linux 系统无法避免分段管理。
事实上，Linux内核采用的办法是使用段式映射的过程中不起什么作用
Linux系统中每个段都是从0地址开始的整个4G虚拟空间（32位），所
有段起始地址都是相同的，这就意味着包括系统本身的代码和应用程
序代码所面对的都是线性地址空间（虚拟地址）相当于屏蔽了处理器
的逻辑地址概念，段只被用于访问控制和内存保护
虚拟空间分布
内核空间
进程在内核态，才能访问内核内存空间的内存
用户空间
进程在用户态时，只能访问用户空间内存
代码段、全局变量、BSS、
函数栈、堆内存、映射区
每个虚拟内存中的地址，其实关联的都是相同的物理地址
每个进程的内核空间都是一致的
32 位系统的内核空间占用 1G，位于最高处，剩下的 3G 是用户空间；
64 位系统的内核空间和用户空间都是 128T，分别占据整个内存空间的最
高和最低处，剩下的中间部分是未定义的。