操作系统

Operating system

孔维强

大连理工大学

内容纲要

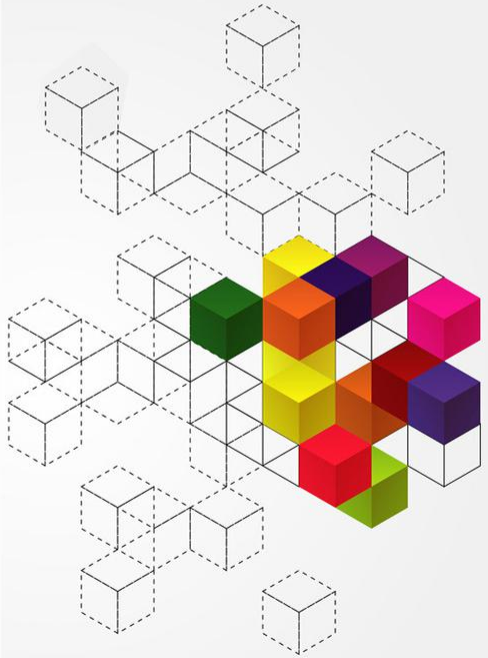
9.6 内核内存分配

一、 内核内存分配需求

二、 伙伴算法

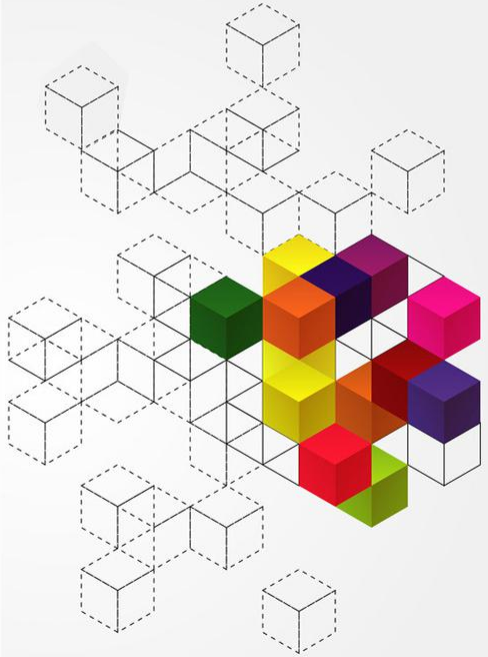
三、 Slab算法

四、 按需分页的一些考虑



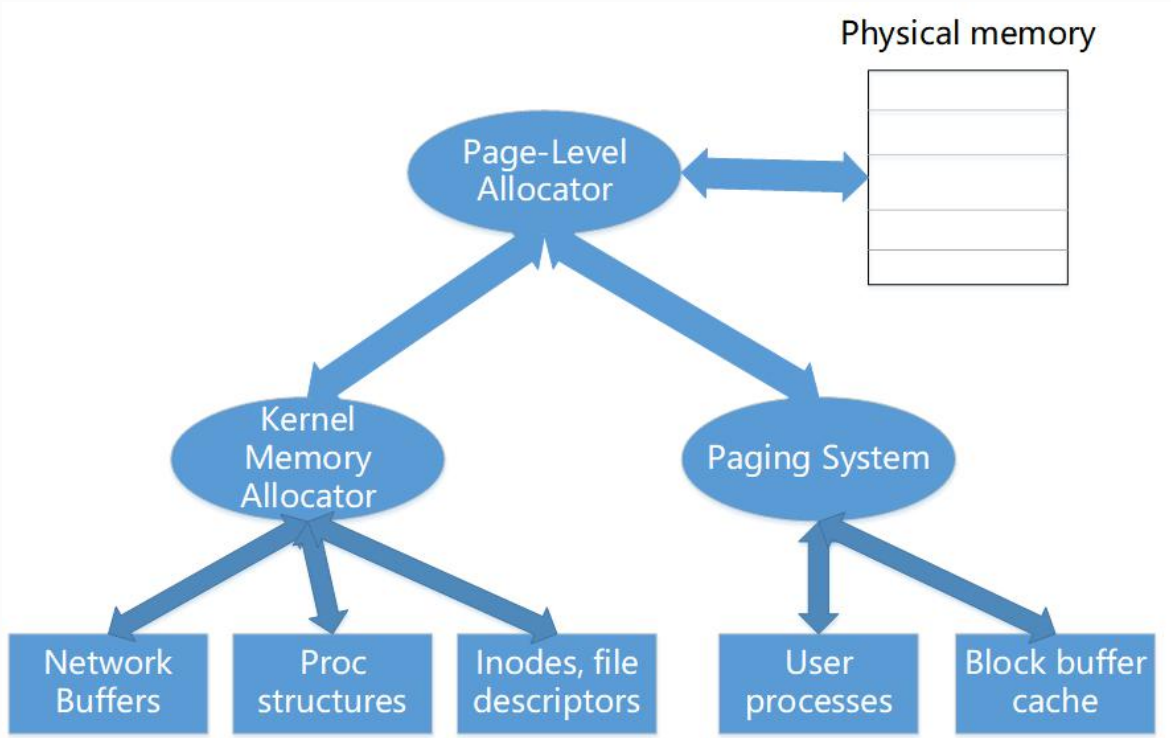
一、内核内存分配需求

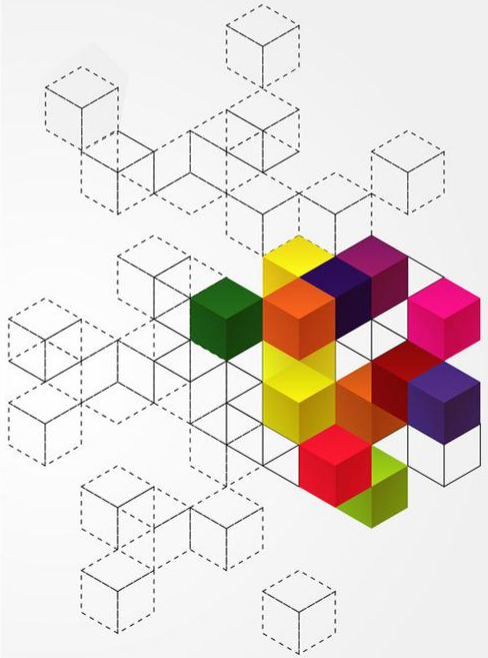
- **操作系统内核的内存分配需求**
  - 与用户态内存分配需求有所区别
  - 内核中有多种固定结构的内核对象，它们通常需要固定大小的内存
  - 内核对象通常要求放在连续的物理内存，以保证内核执行效率



一、内核内存分配需求

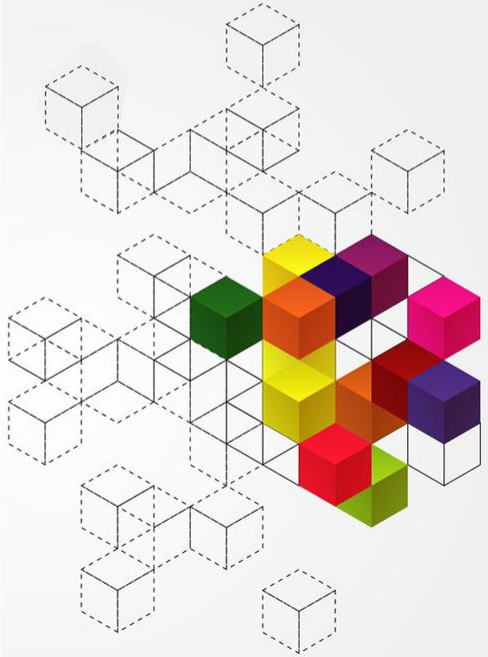
- OS页面级内存分配器与内核内存分配器



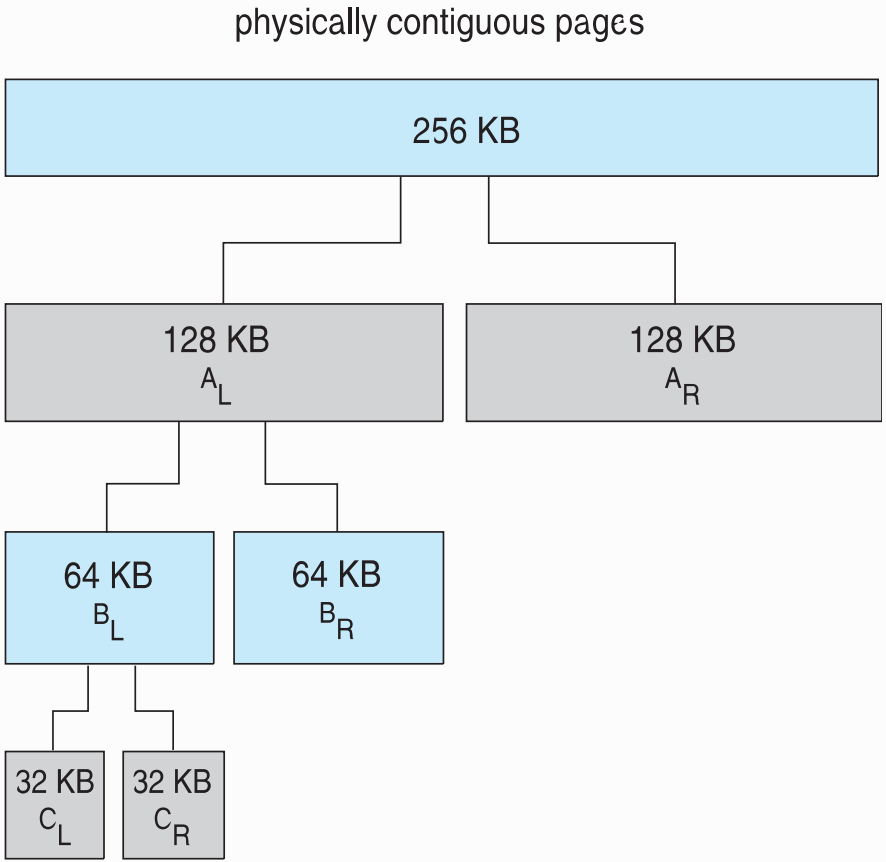


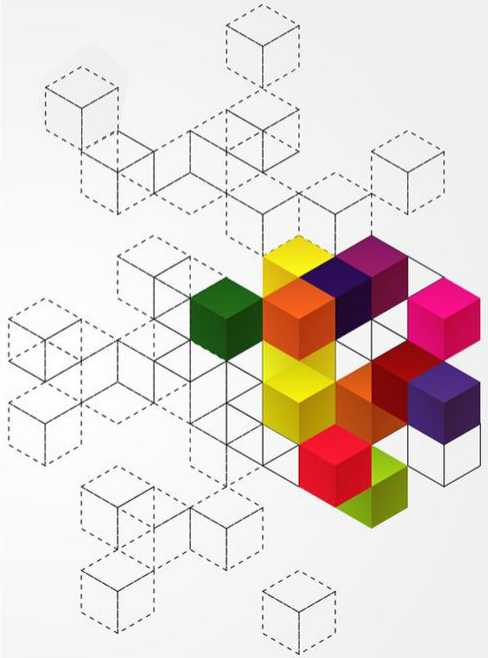
二、伙伴算法(Buddy Algorithm)

- **基本思想：从物理上连续的内存区域分配连续的内存块**
  - Create small buffers by repeatedly halving a large buffer (buddy pairs) and coalescing adjacent buffers when possible
  - Requests rounded up to a power of two



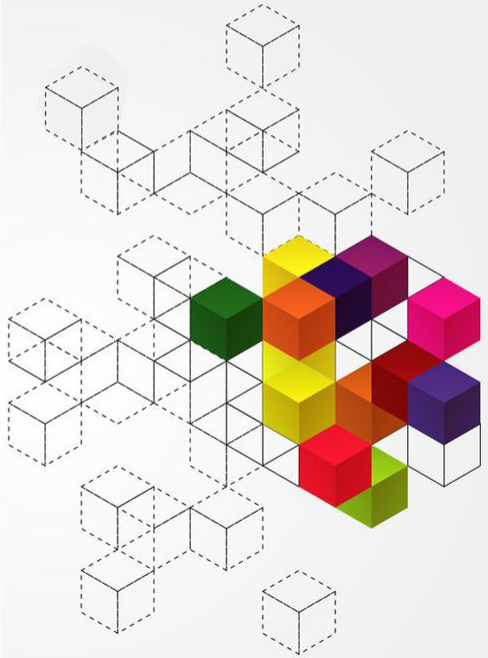
二、伙伴算法


physically contiguous pages
256 KB
128 KB
A_L
128 KB
A_R
64 KB
B_L
64 KB
B_R
32 KB
C_L
32 KB
C_R



二、伙伴算法

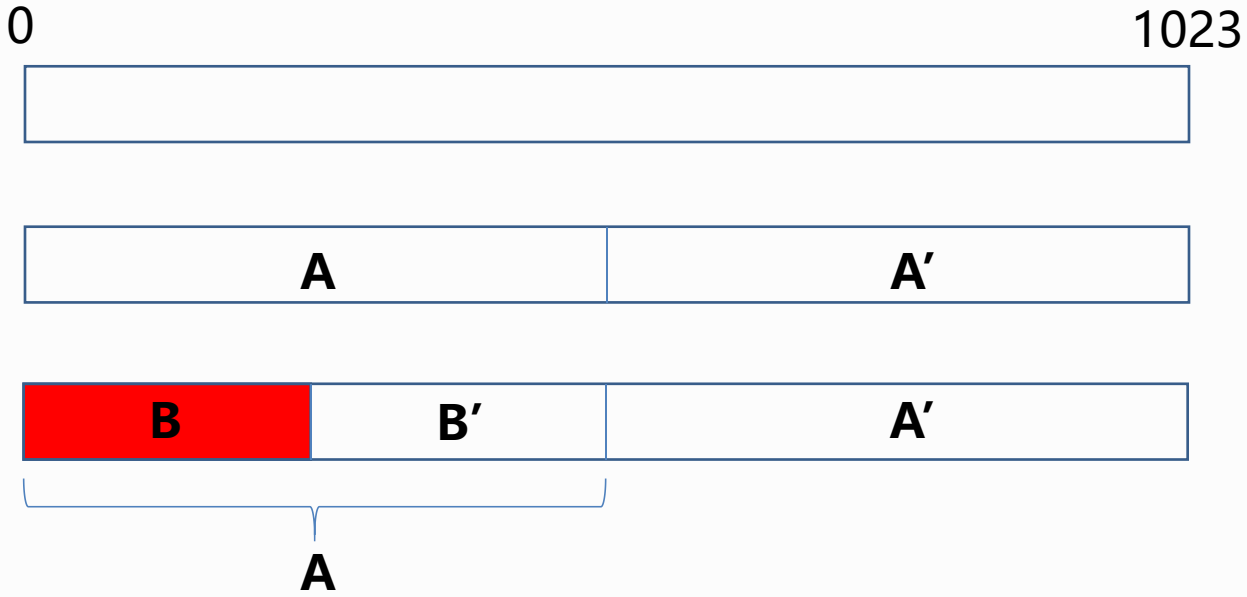
- **Buddy System Example**
  - Minimum allocation size = 32 Bytes
  - Initial free memory size is 1024
  - Use a bitmap to monitor 32 Byte chunks
    - Bit set if chunk is used
    - Bit clear if chunk is free
  - Maintain freelist for each possible buffer size
    - Power of 2 buffer sizes from 32 to 512
    - Sizes = {32, 64, 128, 256, 512}
  - Initial one block = entire buffer

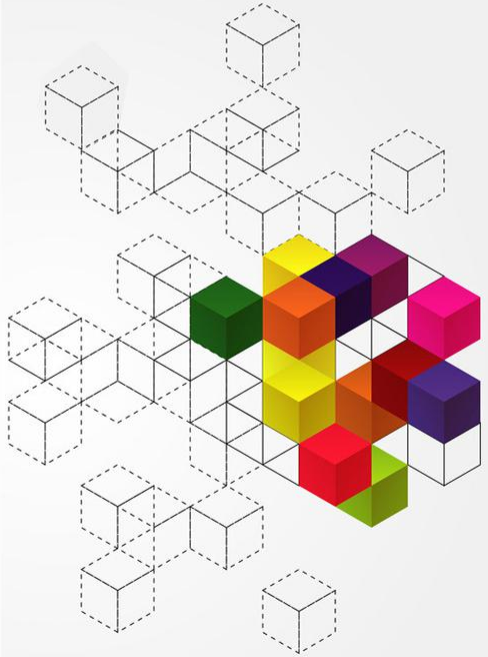


二、伙伴算法

- 伙伴算法示例

Allocate(256)

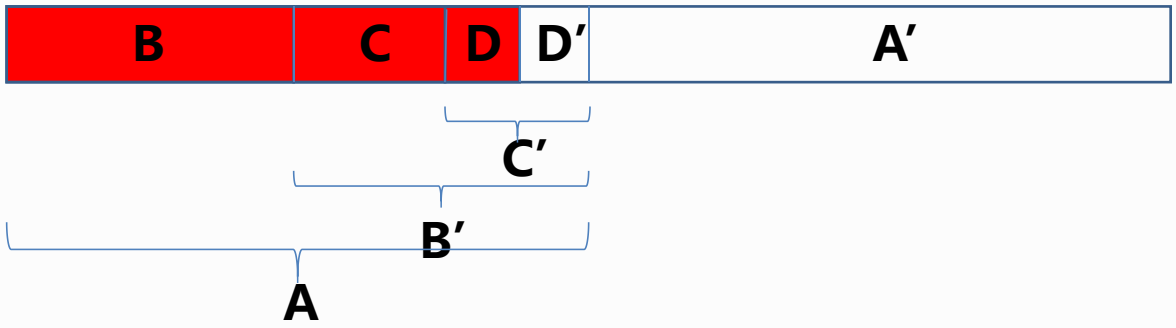


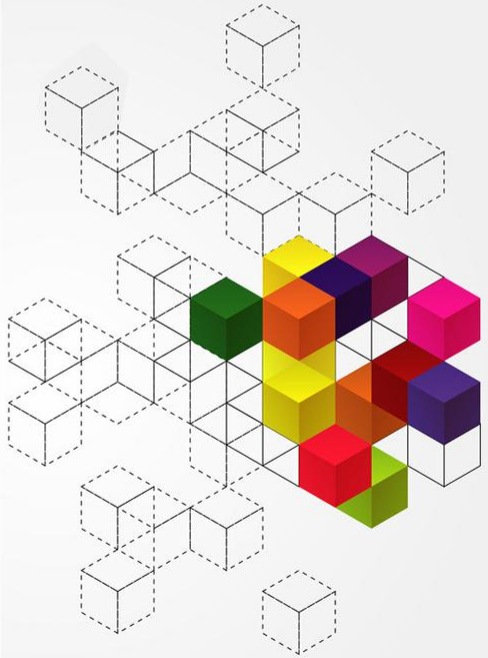


二、伙伴算法

- 伙伴算法示例

→ Allocate(128) → Allocate(64)

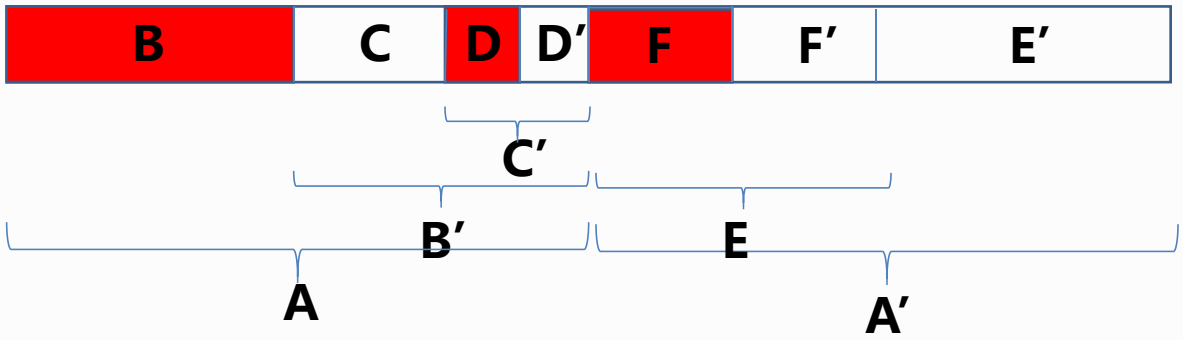


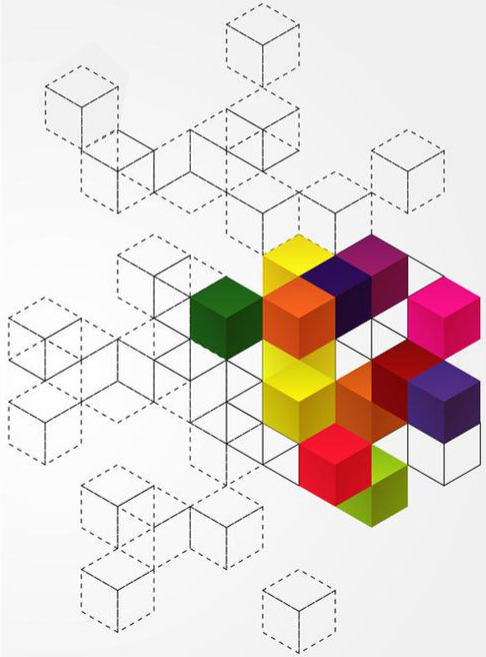


二、伙伴算法

- 伙伴算法示例

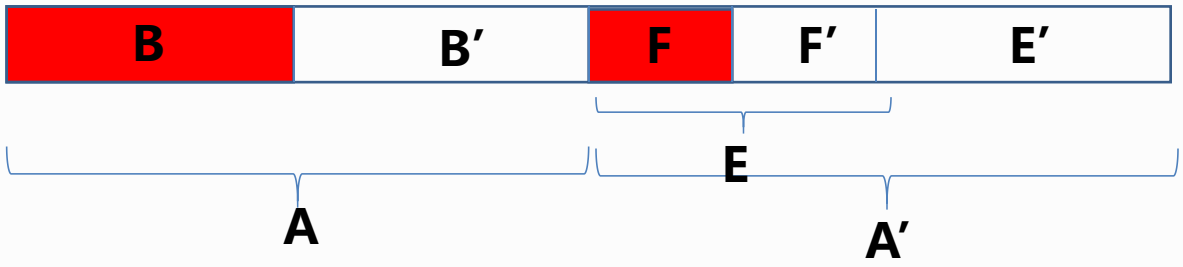
release(D,64)

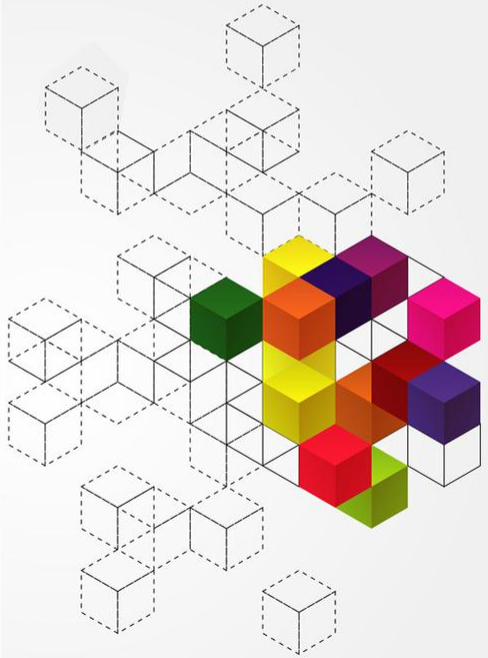




二、伙伴算法

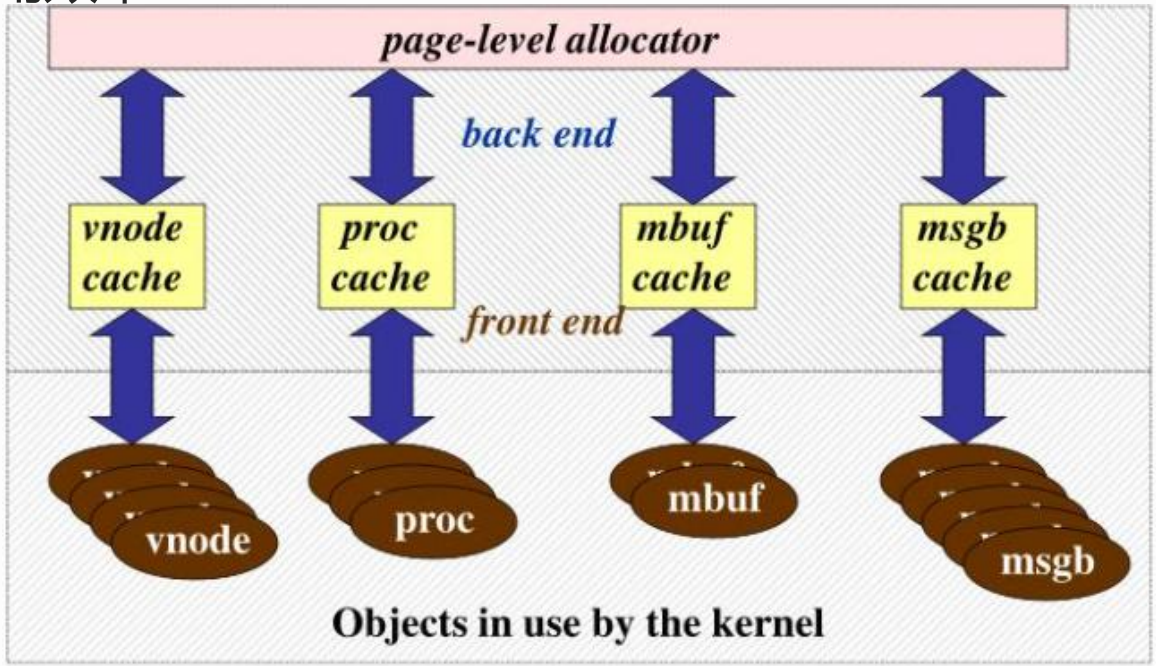
- 伙伴算法示例

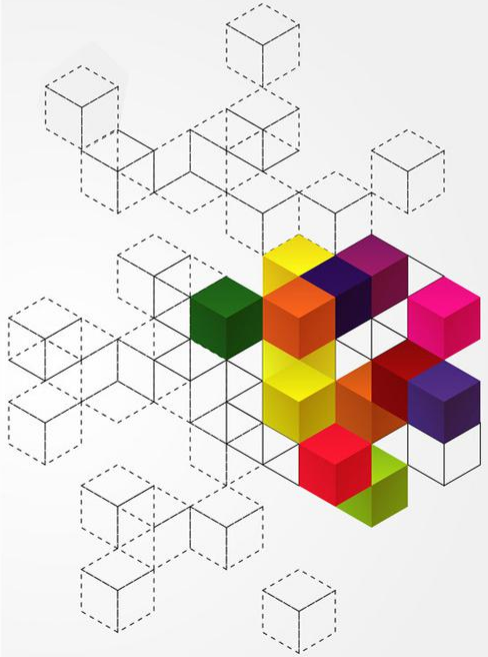




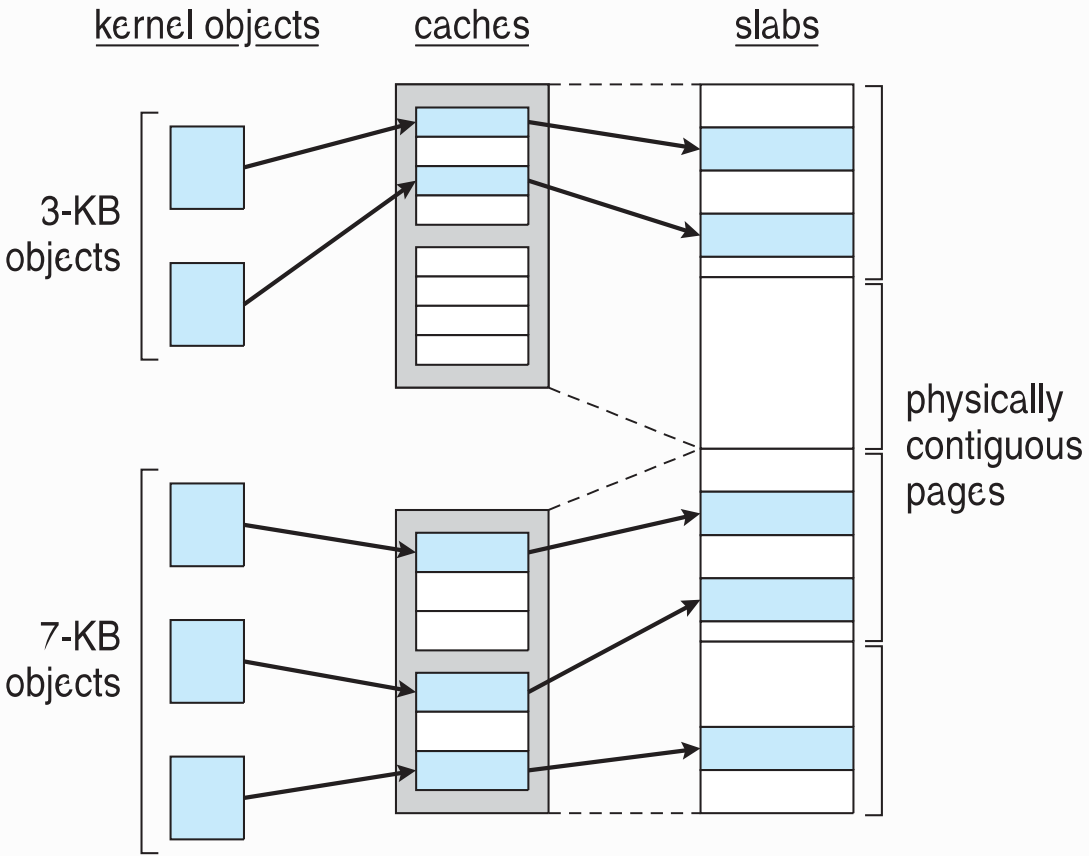
三、Slab算法

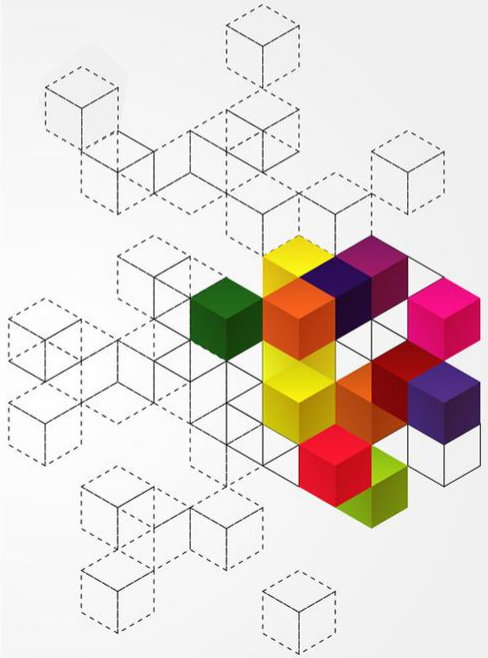
- 基本思想：针对内核数据结构对象的特点，设置不同大小的对象缓冲区，以获得较高的内存分配与使用效率





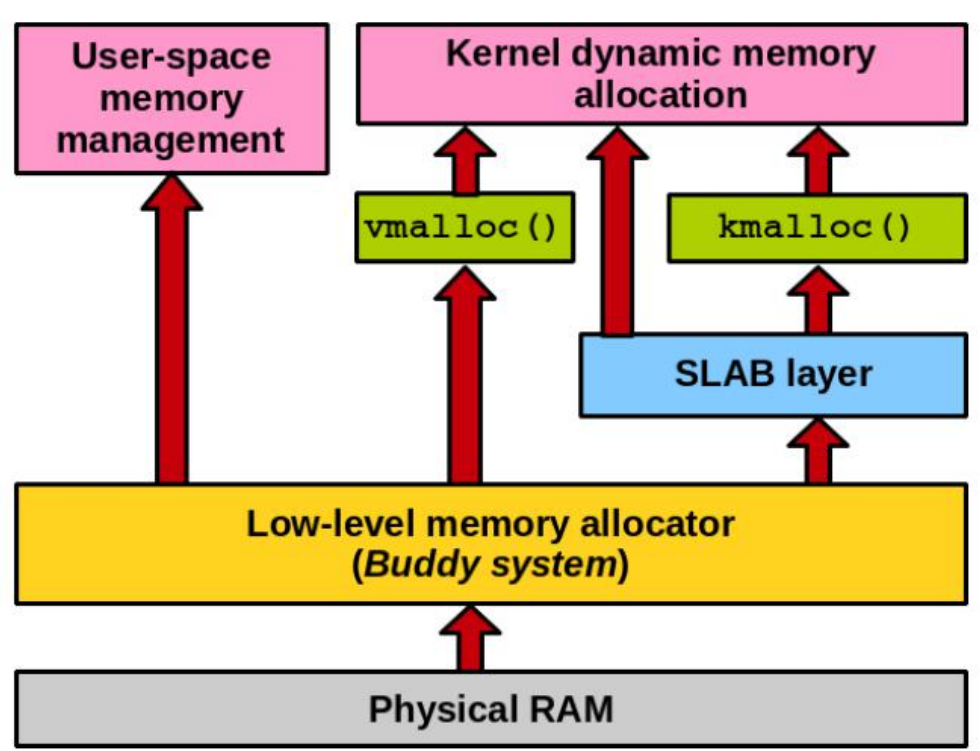
三、Slab算法

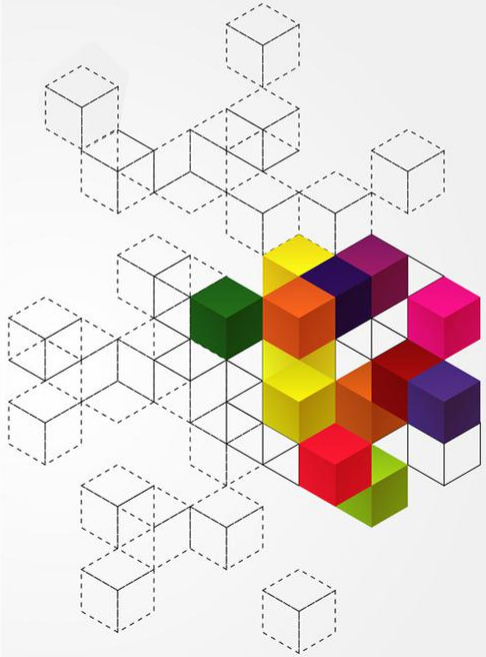

kernel objects
caches
slabs
3-KB objects
7-KB objects
physically contiguous pages



三、Slab算法

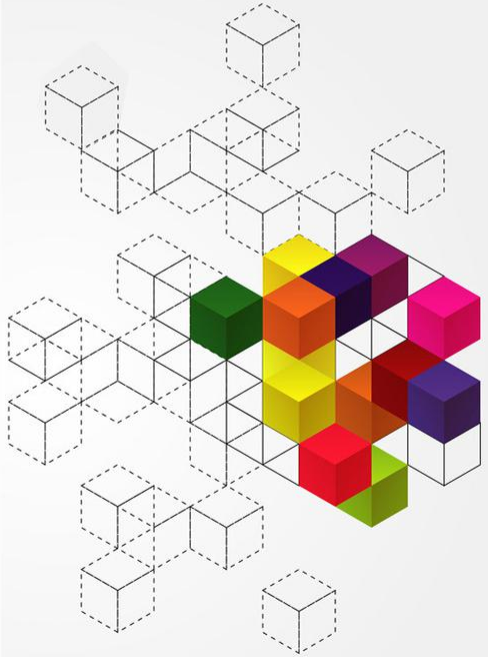
- Linux Kernel Memory Management





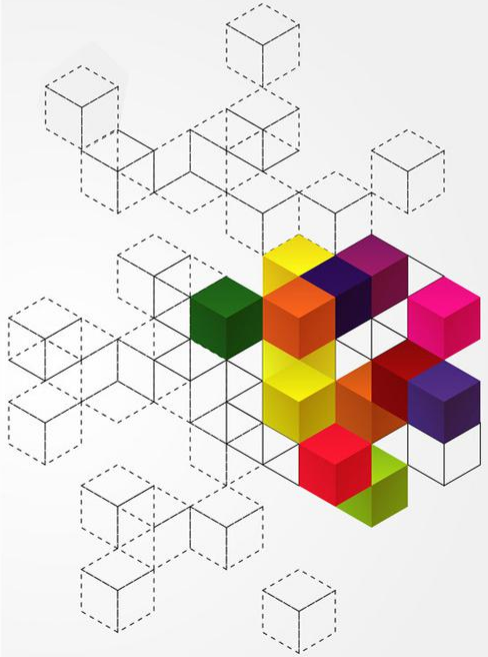
四、按需分页的一些考虑

- Prepaging（预分页）
  - 在程序启动时，减少大量的页错误（例：完全按需分页）
  - 在页面被访问前，预分进程所需的部分或全部页面（例：工作集）
  - 但是如果预分页未被使用，则预分页涉及的I/O操作就浪费了
  - 假设 s 个页面被预分页，其中 α 个页面被实际使用
    - 因 s * α 节省的页错误消耗，是否大于或小于对于 s * (1- α) 个未使用页面进行预分页的消耗？
    - 如果 α 接近0 ⇒ 预分页失败



四、按需分页的一些考虑

- Page Size（页大小）
  - **页面大小的选择必须考虑以下因素：**
    - **1. 碎片**
    - **2. 页表大小**
    - **3. 提取成本**
    - **4. I/O消耗**
    - **5. 页错误的数量**
    - **6. 局部性**
    - **7. TLB的大小与效率**



四、按需分页的一些考虑

- Program structure （程序结构）
  - int[128,128] data;
  - Each row is stored in one page
  - **Program 1**

```
for (j = 0; j <128; j++)
    for (i = 0; i < 128; i++)
        data[i,j] = 0
```

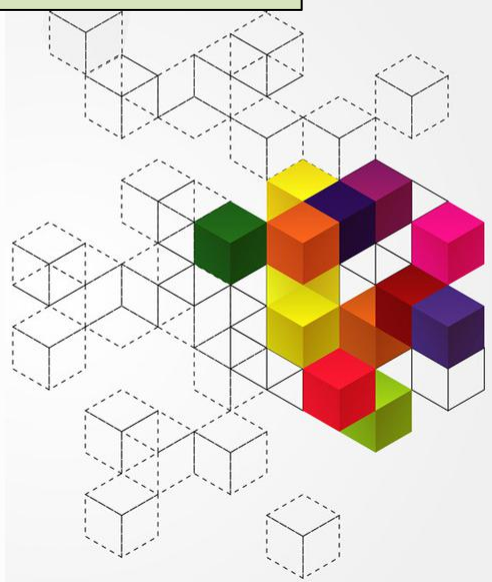
128 x 128 = 16,384 page faults

  - **Program 2**

```
for (i = 0; i < 128; i++)
    for (j = 0; j < 128; j++)
        data[i,j] = 0;
```

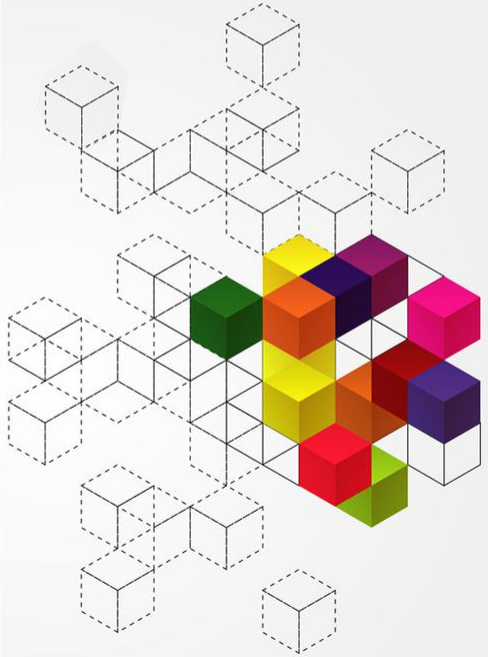
128 page faults

-- Page1 --
data[0,0], data[0,1]..., data[0,127]
-- Page2 --
data[1,0], data[1,1]..., data[1,127]



本讲小结

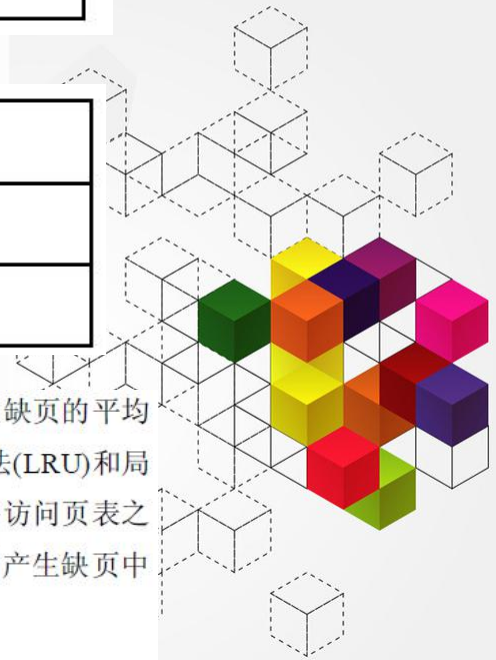
- 内核内存分配需求
- 伙伴算法
- Slab算法



练习1

请求分页管理系统中，假设某进程的页表内容如下表所示：

页号	页框(Page Frame)号	有效位(存在位)
0	101H	1
1	——	0
2	254H	1



页面大小为 4KB，一次内存的访问时间是 100ns，一次快表(TLB)的访问时间是 10ns，处理一次缺页的平均时间 10^8ns(已含更新 TLB 和页表的时间)，进程的驻留集大小固定为 2，采用最近最少使用置换算法(LRU)和局部淘汰策略。假设①TLB 初始为空；②地址转换时先访问 TLB，若 TLB 未命中，再访问页表(忽略访问页表之后的 TLB 更新时间)；③有效位为 0 表示页面不在内存，产生缺页中断，缺页中断处理后，返回到产生缺页中断的指令处重新执行。设有虚地址访问序列 2362H、1565H、25A5H，请问：

(1) 依次访问上述三个虚地址，各需多少时间？给出计算过程。

(2) 基于上述访问序列，虚地址 1565H 的物理地址是多少？请说明理由。

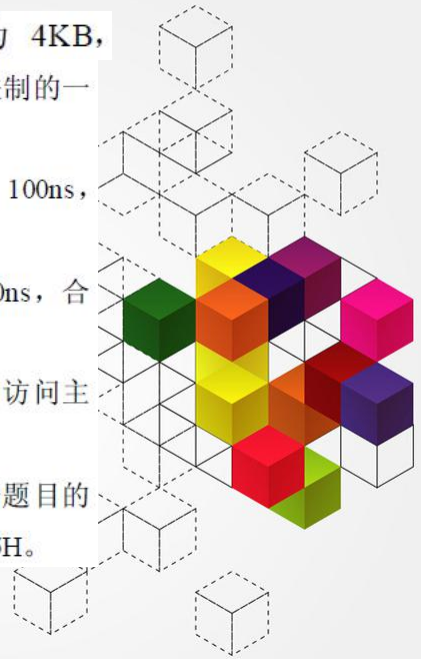
(1) 根据页式管理的工作原理，应先考虑页面大小，以便将页号和页内位移分解出来。页面大小为 4KB，即 2^{12}，则得到页内位移占虚地址的低 12 位，页号占剩余高位。可得三个虚地址的页号 P 如下（十六进制的一位数字转换成 4 位二进制，因此，十六进制的低三位正好为页内位移，最高位为页号）：

2362H：P=2，访问快表 10ns，因初始为空，访问页表 100ns 得到页框号，合成物理地址后访问主存 100ns，共计 10ns+100ns+100ns=210ns。

1565H：P=1，访问快表 10ns，落空，访问页表 100ns 落空，进行缺页中断处理 10^8ns，访问快表 10ns，合成物理地址后访问主存 100ns，共计 10ns+100ns+10^8ns+10ns+100ns=100 000 220ns。

25A5H：P=2，访问快表，因第一次访问已将该页号放入快表，因此花费 10ns 便可合成物理地址，访问主存 100ns，共计 10ns+100ns=110ns。

(2) 当访问虚地址 1565H 时，产生缺页中断，合法驻留集为 2，必须从页表中淘汰一个页面，根据题目的置换算法，应淘汰 0 号页面，因此 1565H 的对应页框号为 101H。由此可得 1565H 的物理地址为 101565H。



46. （8 分）设某计算机的逻辑地址空间和物理地址空间均为 64KB，按字节编址。若某进程最多需要 6 页(Page)数据存储空间，页的大小为 1KB，操作系统采用固定分配局部置换策略为此进程分配 4 个页框(Page Frame)。在时刻 260 前的该进程访问情况如下表所示（访问位即使用位）。

页号	页框号	装入时刻	访问位
0	7	130	1
1	4	230	1
2	2	200	1
3	9	160	1

当该进程执行到时刻 260 时，要访问逻辑地址为 17CAH 的数据。请回答下列问题：

(1) 该逻辑地址对应的页号是多少？

(2) 若采用先进先出（FIFO）置换算法，该逻辑地址对应的物理地址是多少?要求给出计算过程。

(3) 若采用时钟（CLOCK）置换算法，该逻辑地址对应的物理地址是多少？要求给出计算过程 （设搜索下一页的指针沿顺时针方向移动，且当前指向 2 号页框，示意图如下） 。

(1) 由于该计算机的逻辑地址空间和物理地址空间均为 64KB = 2^{16} B，按字节编址，且页的大小为 1K = 2^{10}，故逻辑地址和物理地址的地址格式均为：

页号/页框号（6位）	页内偏移量（10位）

17CAH = 0001 0111 1100 1010B，可知该逻辑地址的页号为 000101B = 5

(2) 根据 FIFO 算法，需要替换装入时间最早的页，故需要置换装入时间最早的 0 号页，即将 5 号页装入 7 号页框中，所以物理地址为 0001 1111 1100 1010B = 1FCAH。

(3) 根据 CLOCK 算法，如果当前指针所指页框的使用位为 0，则替换该页；否则将使用位清零，并将指针指向下一个页框，继续查找。根据题设和示意图，将从 2 号页框开始，前 4 次查找页框号的顺序为 2→4→7→9，并将对应页框的使用位清零。在第 5 次查找中，指针指向 2 号页框，因 2 号页框的使用位为 0，故淘汰 2 号页框对应的 2 号页，把 5 号页装入 2 号页框中，并将对应使用位设置为 1，所以对应的物理地址为 0000 1011 1100 1010B = 0BCAH。