操作系统

Operating system

孔维强

大连理工大学

内容纲要

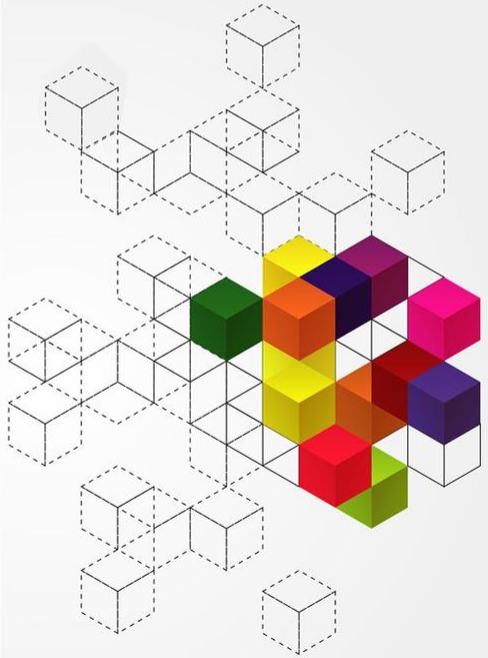
9.1 虚存概念

一、 虚存思想

二、 引入虚存的意义

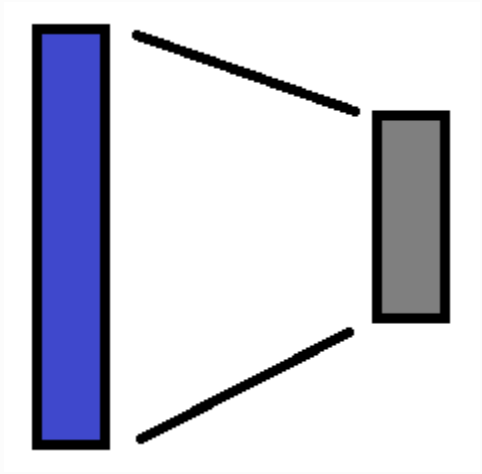
三、 按需调页

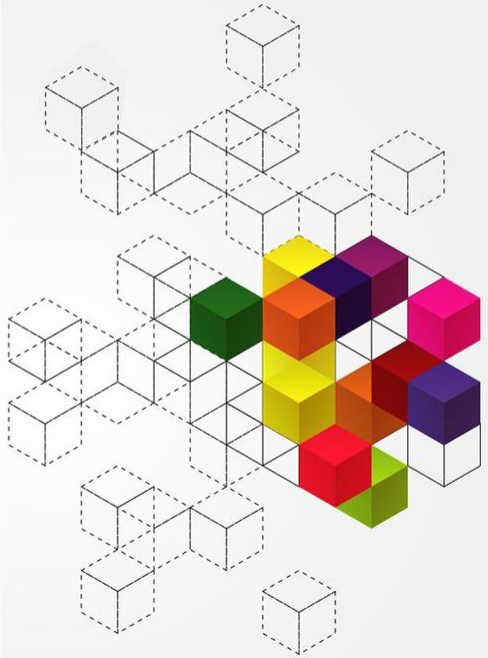
四、 虚存所需硬件支持



一、虚存思想

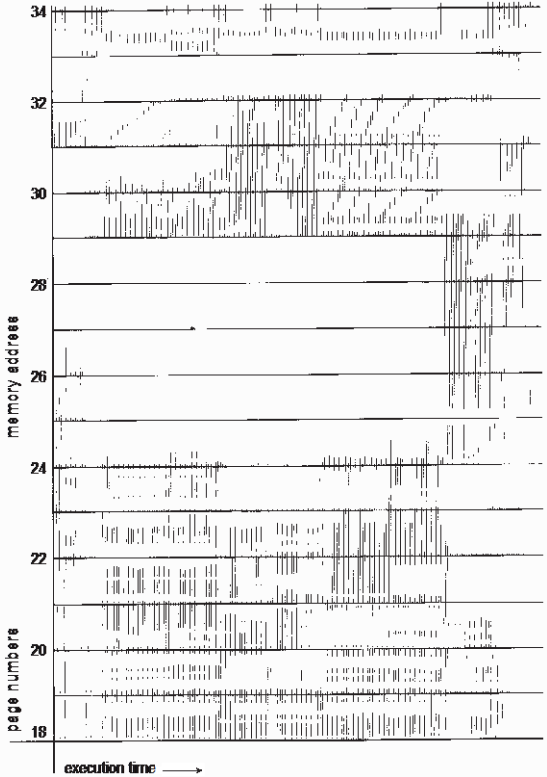
- **虚存机制的核心思想：用较少的物理内存，支撑较大的逻辑地址空间**
  - 大逻辑地址空间的好处：应用编程易
  - 小物理内存空间的好处：占用资源少

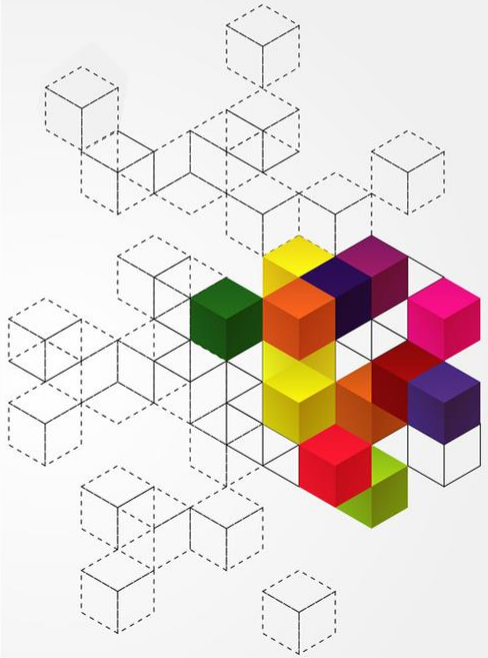




一、虚存思想

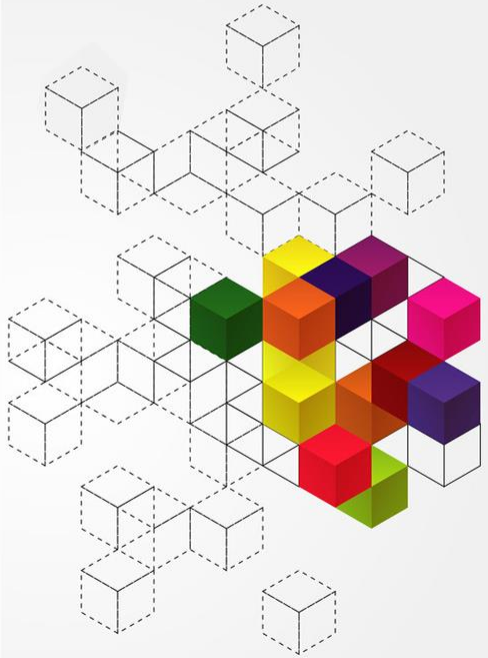
- **虚存机制的理论基础：程序执行的局部性原理**
  - 关键词：Locality





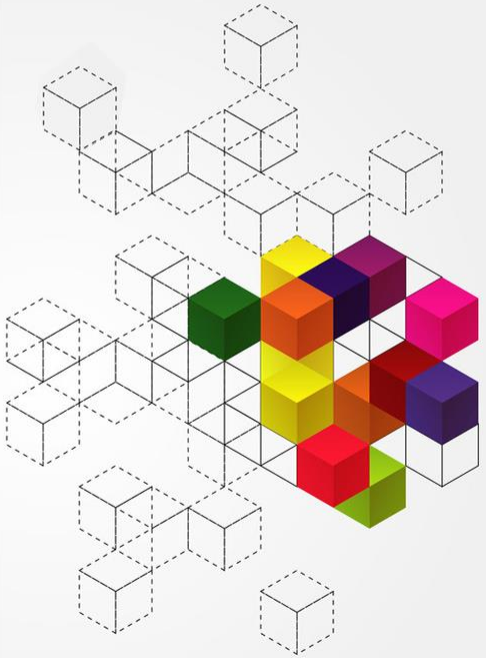
一、虚存思想

- **虚存机制的基本工作原理**
  - 进程初始状态时，并不一次性将逻辑地址空间的所有代码和数据加载入物理内存（不考虑性能因素，甚至可以初始零加载）
  - 进程执行过程中，根据当前执行的局部，根据需要加载所需的逻辑空间内容到物理内存



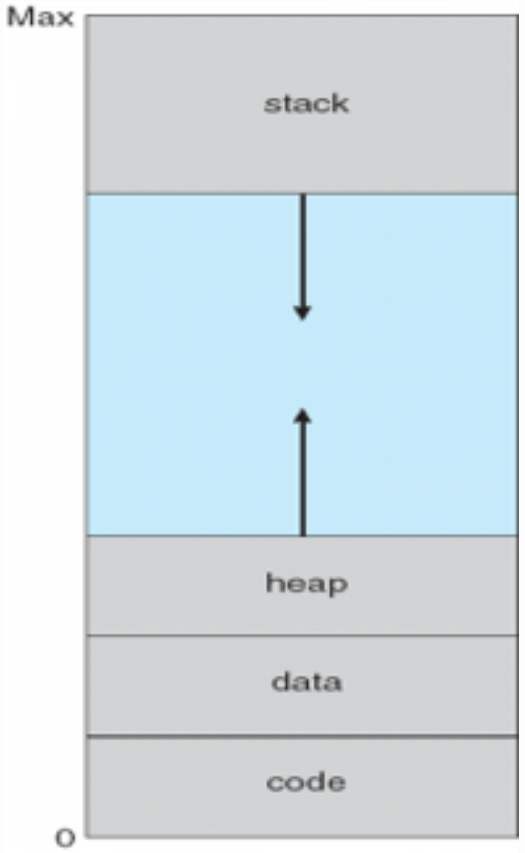
二、引入虚存的意义

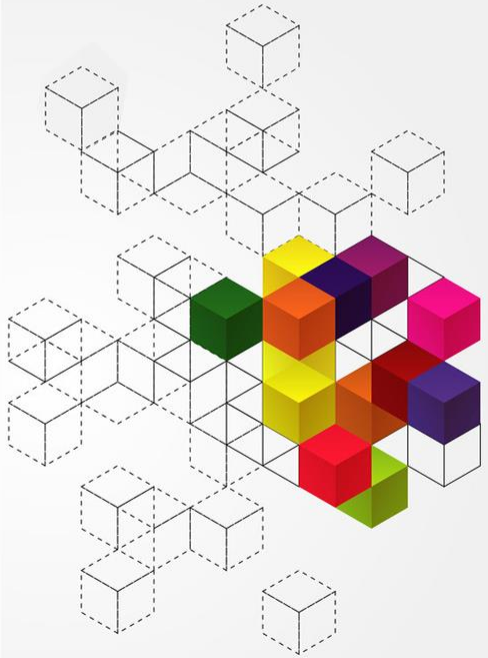
- **虚存机制的优势**
  - 不再需要一次性加载程序的所有代码和一次运行所需的所有数据，可以节省大量物理内存
  - 使得进程的创建更有效率
  - 可以较少IO操作
  - 支持更多的并发进程
  - 方便实施在进程间共享物理内存



二、引入虚存的意义

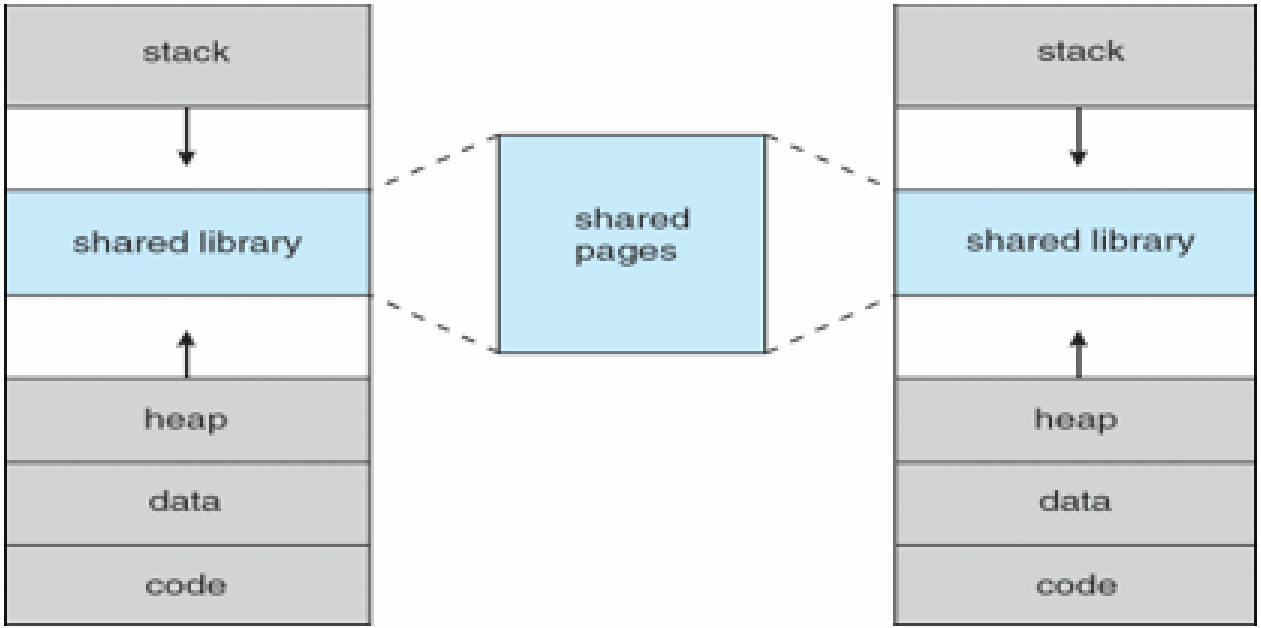
- **形成较大的虚地址空间**
  - 开发人员进行应用开发时，不需要太多考虑对逻辑地址空间大小进行限制的问题
  - 通常，包含代码、数据、运行时堆和栈的逻辑地址空间都会存在较大富余

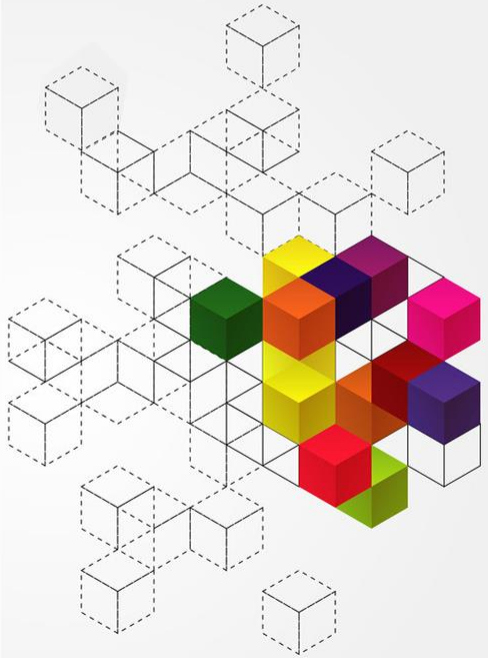




二、引入虚存的意义

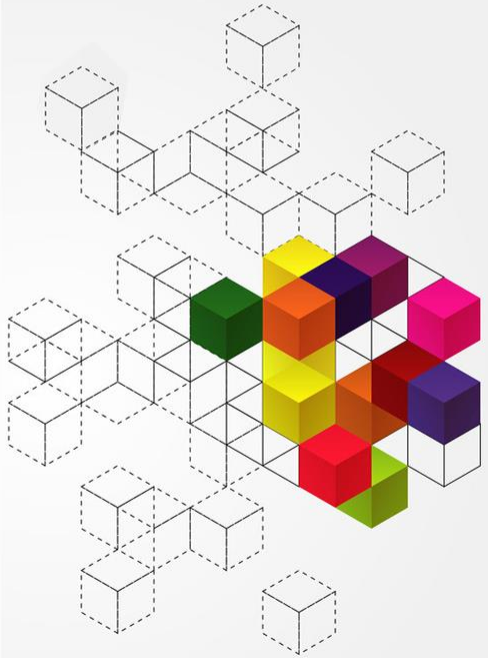
- **便于进程间进行内存共享**
  - 可以将相同一块物理内存区域映射到不同进程的逻辑地址空间





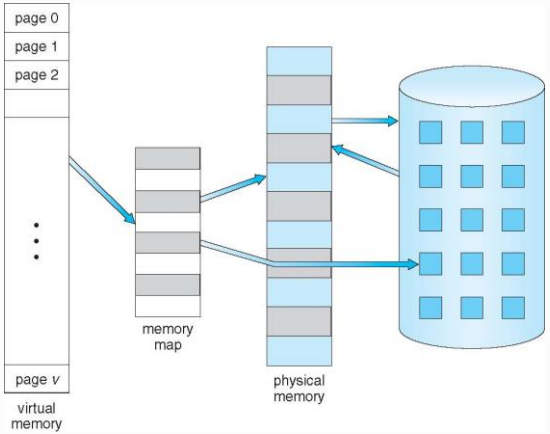
三、按需调页

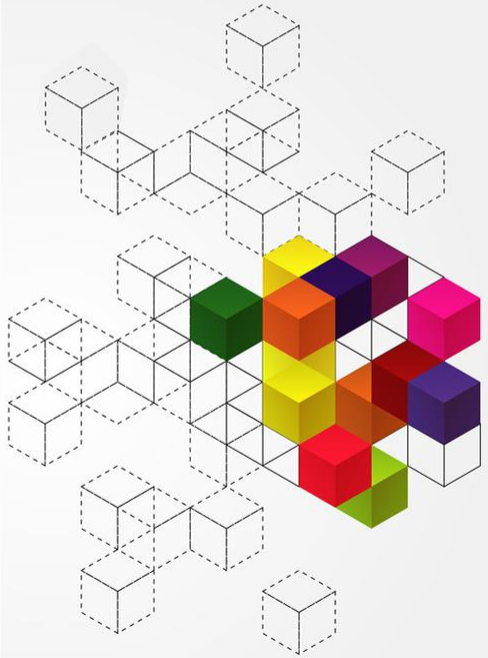
- **虚存实现的关键技术：Demand Paging**
  - 仅当页面被需要（访问）时才将其加入内存
    - 更少的I/O需求，无不必要的I/O
    - 需要更少的内存
    - 更快的响应时间 etc.
  - 页面被需要->访问（reference）
    - 无效访问，则退出
    - 不在内存，则加载至内存
  - 与swap相似，但不是针对整个进程的交换
  - 亦被称作Lazy swapper或pager



三、按需调页

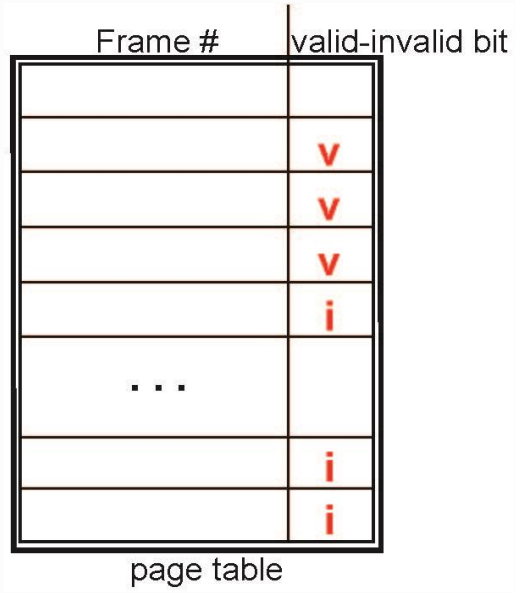
- **虚存实现的关键技术：Demand Paging**
  - 虚存的本质：内存中的页和磁盘上的块结合在一起，完成内存扩充(逻辑地址空间被扩充成一个非常的地址空间)
  - 何时需要将外存中的块数据，通过较慢的IO操作传入内存页，是该项技术的关键

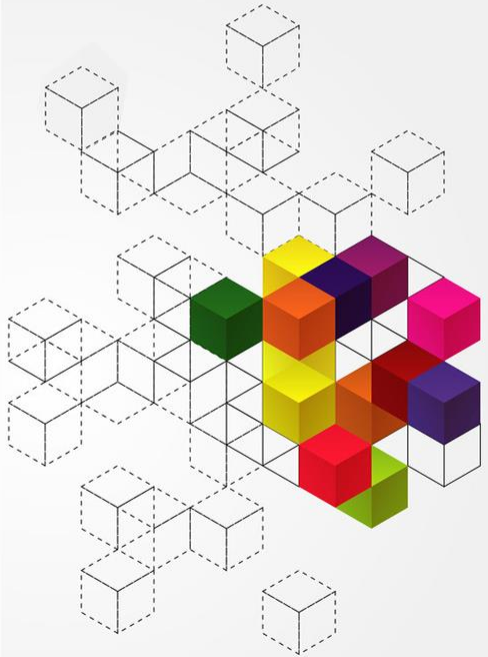




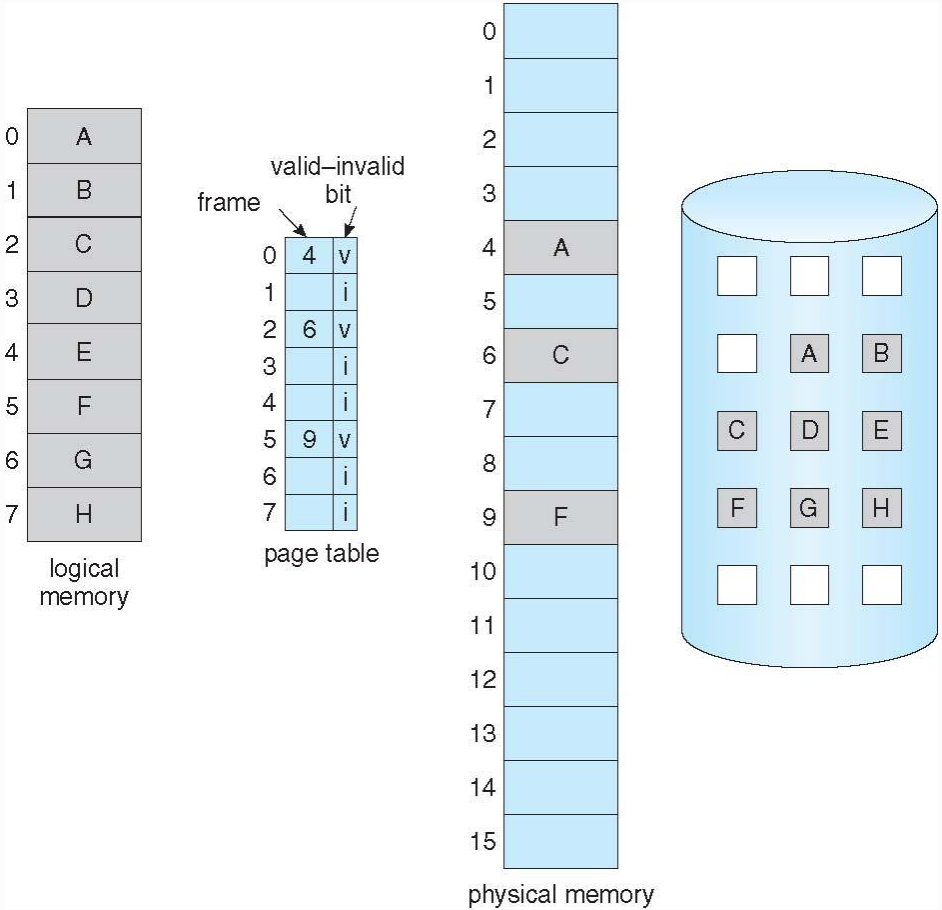
三、按需调页

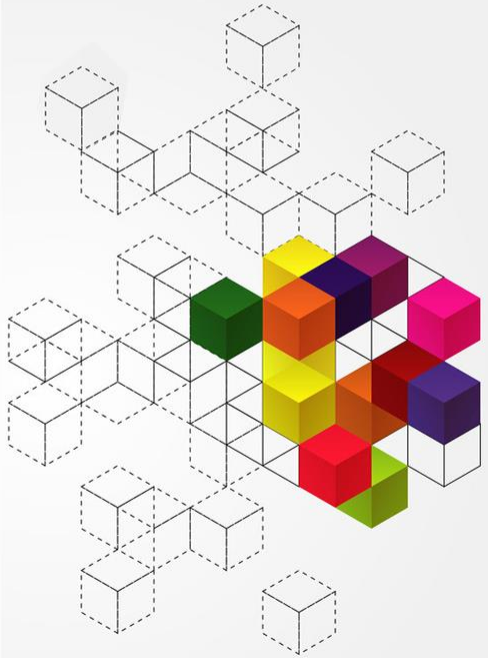
- **有效无效位（valid-invalid bit）**
  - 页表中的每个页表项均带有有效无效位
  - v：有效且在内存；i：无效或不在内存
  - 初始时所有位均被置为 i
  - 在MMU地址翻译时，如果该位为 i，则产生 page fault （页错误）





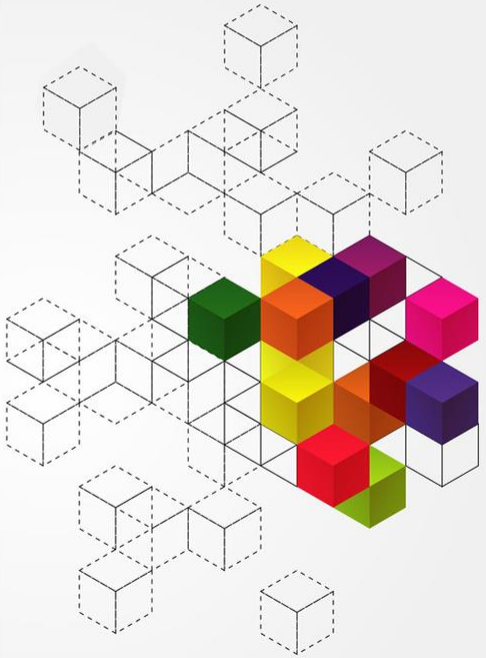
三、按需调页


0 A
1 B
2 C
3 D
4 E
5 F
6 G
7 H
logical memory
valid–invalid bit
frame
0 4 v
1 i
2 6 v
3 i
4 i
5 9 v
6 i
7 i
page table
0
1
2
3
4 A
5
6 C
7
8
9 F
10
11
12
13
14
15
physical memory
A B
C D E
F G H

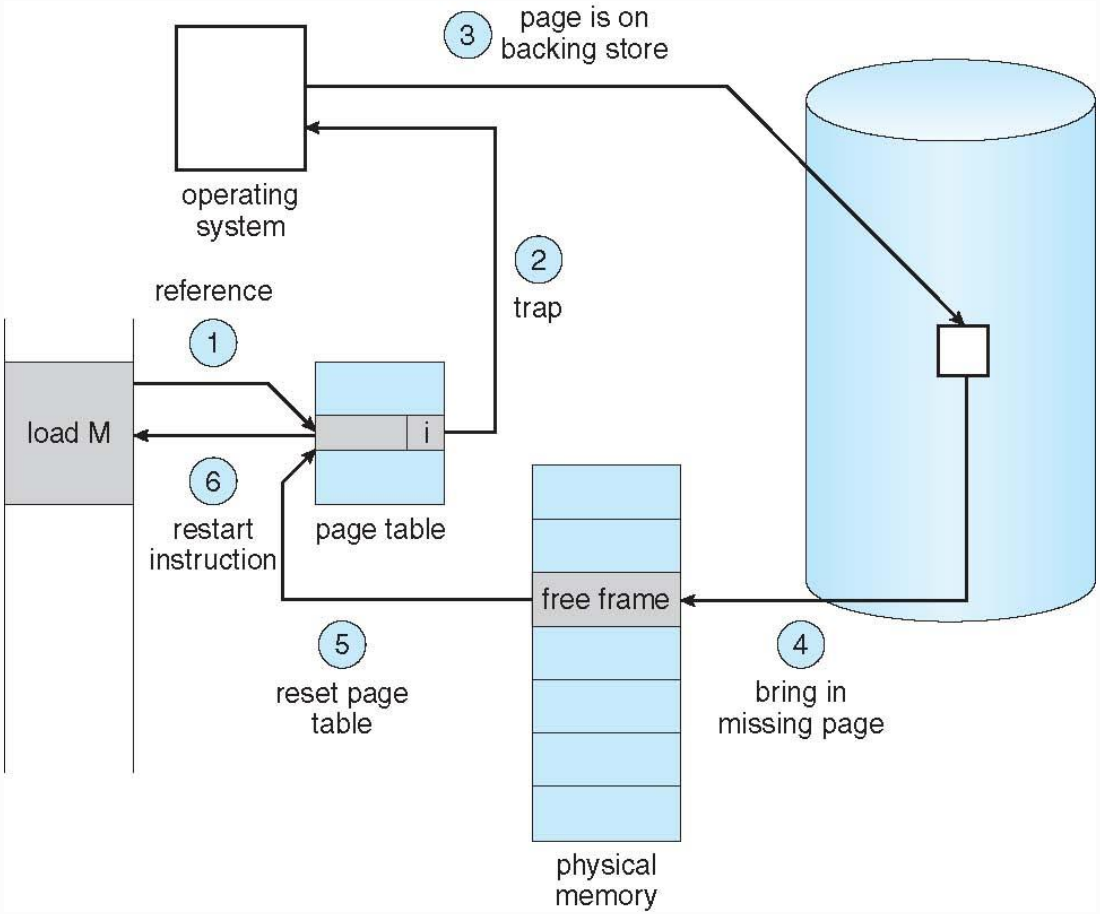


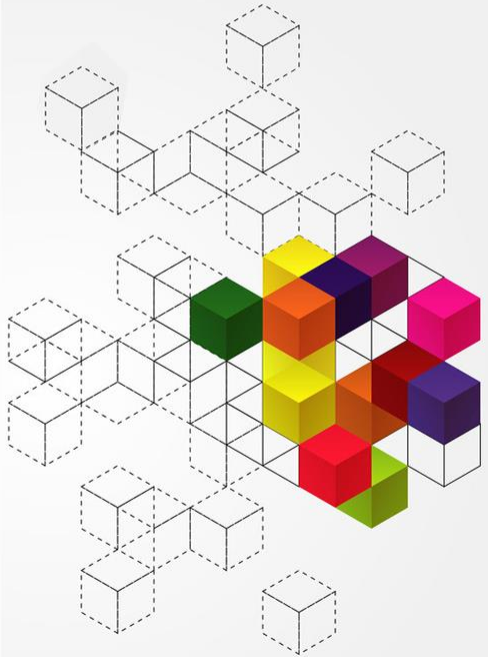
三、按需调页

- **页错误（page fault）**
  - OS查找另一个表（保存在PCB）以决定：
    - 无效访问，则退出
    - 仅是不在内存
  - 查找空闲帧（free frame）
  - 将访问的页面交换至空闲帧
  - 重新设置页表以显示该页现在存在于内存
    - 设置有效无效位为 v
  - 重启导致页错误的指令



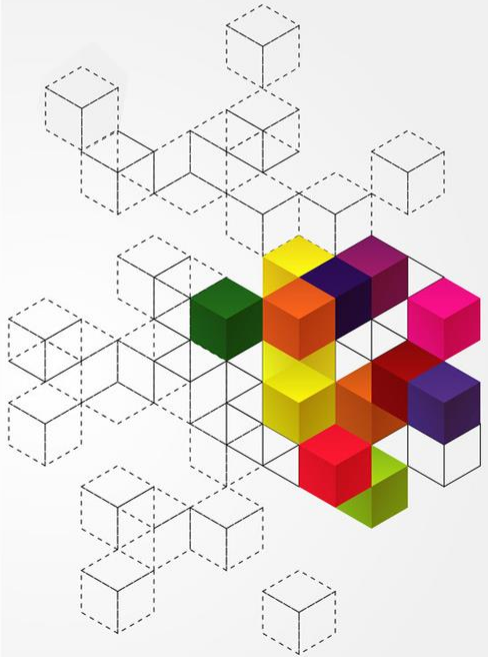
三、按需调页


3
page is on
backing store
operating
system
2
trap
reference
1
load M
i
6
restart
instruction
page table
free frame
5
reset page
table
4
bring in
missing page
physical
memory



三、按需调页

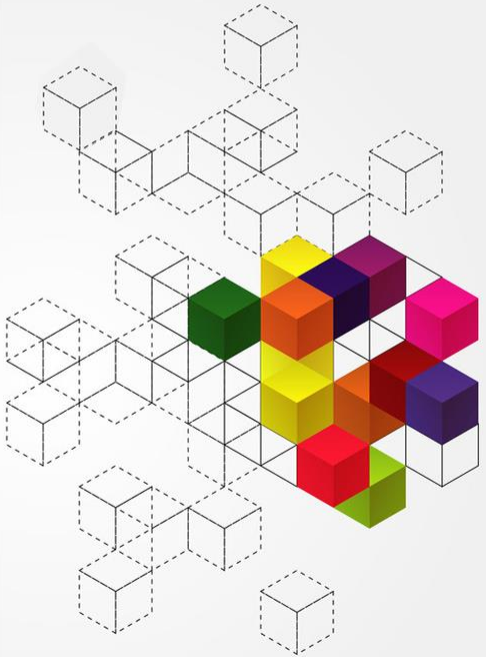
- **完全按需分页（pure demand paging）**
  - 初始时进程无任何页面在内存
  - OS执行进程的第1条指令→页错误
  - 对于所有其他页面的第1次访问→页错误
  - 实际上，一条指令可能访问多个页面→多个页错误
    - 指令：取两个数、相加、存储
    - 过多页错误可由进程访问的locality缓解



三、按需调页

- **按需分页的性能分析（最差情况）**

1. 发生页错误→trap to OS
2. 保存用户寄存器和进程状态
3. 确定中断为页错误
4. 检查页面访问合法、确定页面在磁盘上的位置
5. 从磁盘上读取页面到空闲帧
6. 在等待读取结束前，分配CPU给其他进程
7. 接收到I/O子系统的中断（告知I/O结束）
8. 保存当前执行的其他进程的寄存器和进程状态



三、按需调页

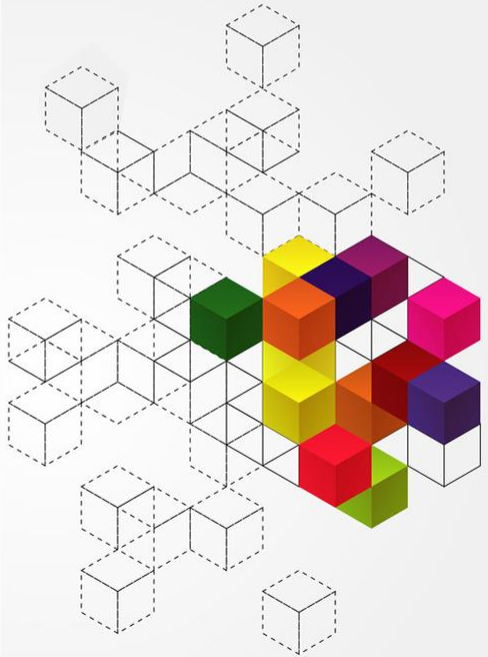
- **按需分页的性能分析（最差情况）**

9. 确定中断来自磁盘

10. 更新页表（及其他相关表格）以说明该页已在内存

11. 等待CPU被分配至该进程

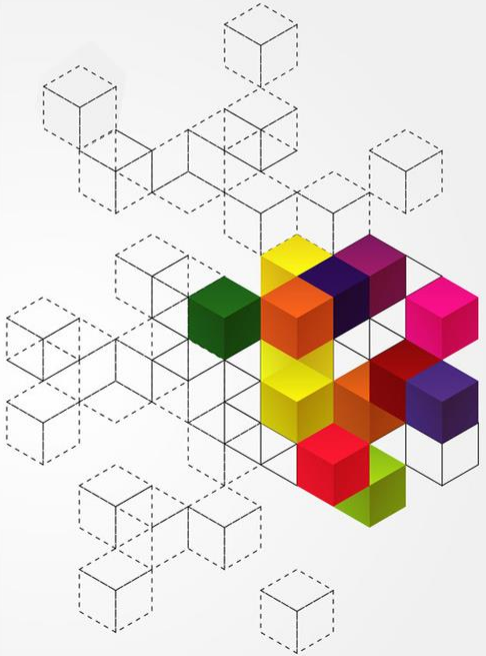
12. 重新加载用户寄存器和进程状态、新页表、然后重启指令



三、按需调页

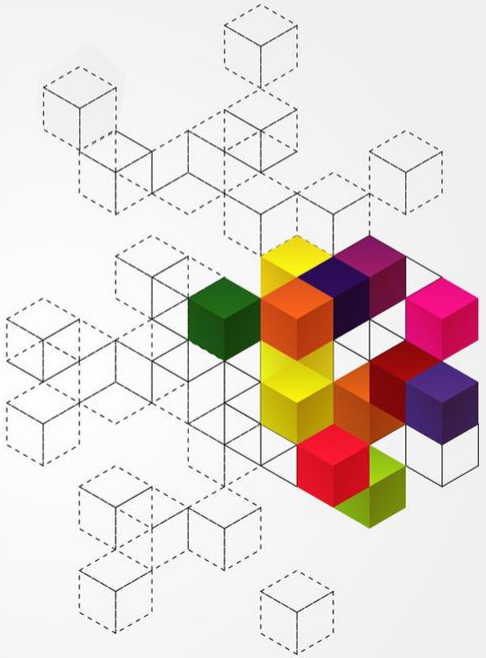
- **有效访问时间Effective Access Time**
  - 三个主要动作:
    - 服务中断—一般中断服务代码仅有几百行
    - 读取页面—很多时间
    - 重启进程—仅很少时间
  - 页错误率$0 \le p \le 1$
    - if p = 0 no page faults
    - if p = 1, every reference is a fault
  - Effective Access Time (EAT)

EAT = (1 – p) x memory access + p (page fault overhead
+ swap page out
+ swap page in)



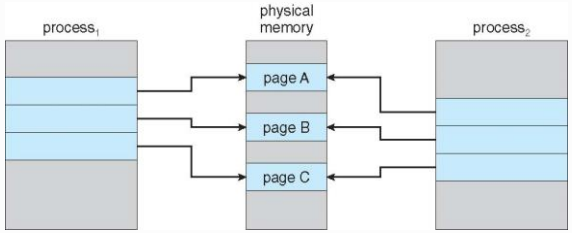
三、按需调页

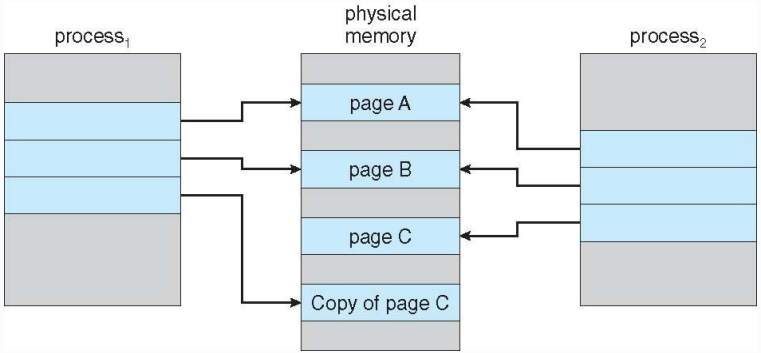
- 内存访问时间 = 200 nanoseconds
- 平均页错误服务时间 = 8 milliseconds
- EAT = (1 – p) x 200 + p (8 milliseconds)
  = (1 – p) x 200 + p x 8,000,000
  = 200 + p x 7,999,800
- 如果每1,000 访问发生一次页错误，则
  EAT = 8.2 microseconds.
- 如果希望性能下降低于 10%
  - 220 > 200 + 7,999,800 x p
    20 > 7,999,800 x p
  - p < .0000025
  - 需低于每400,000 访问发生一次页错误

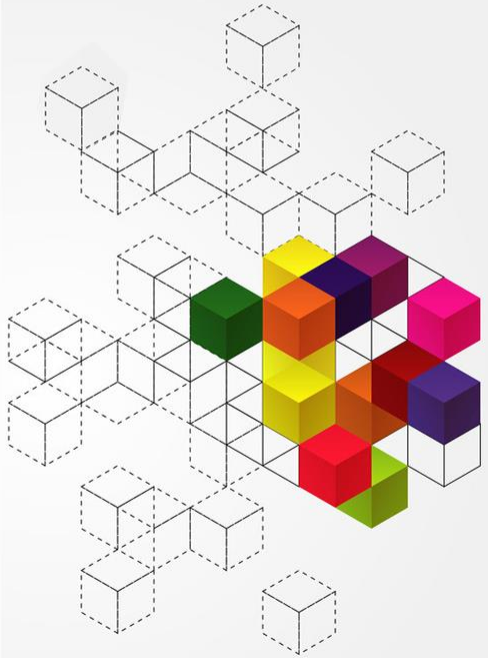


三、按需调页

- **Copy-on-Write (COW)**
  - 允许父进程和子进程在创建时共享内存中的页面
    - 仅当某进程修改了共享页面时，才重新复制被修改的页面
  - COW允许更有效的进程创建（仅复制被修改的页面）

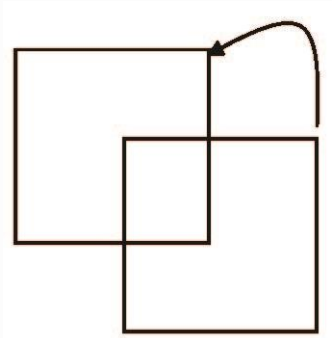




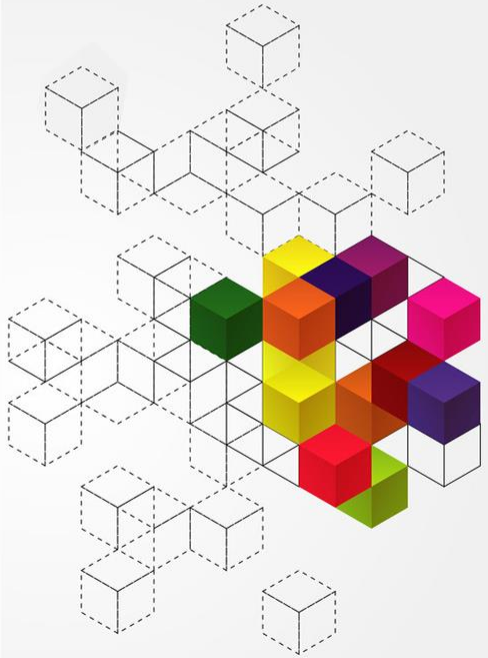


四、虚存所需硬件支持

- **按需调页所需的硬件支持**
  - 页表项中包含有效位
  - 二级内存（交换设备）
  - 指令重启
    - 考虑块移动指令
    - 在部分移动后发生页错误
    - 重启整个操作？
      - 如果源和目的重叠怎么办？
      - 可用临时寄存器存储被修改的数据



这仅是按需调页面临的诸多困难的一个例子



本讲小结

- 虚存思想
- 引入虚存的意义
- 按需调页
- 虚存所需硬件支持