操作系统

Operating system

孔维强

大连理工大学

内容纲要

9.4 页帧分配

一、 页帧分配策略

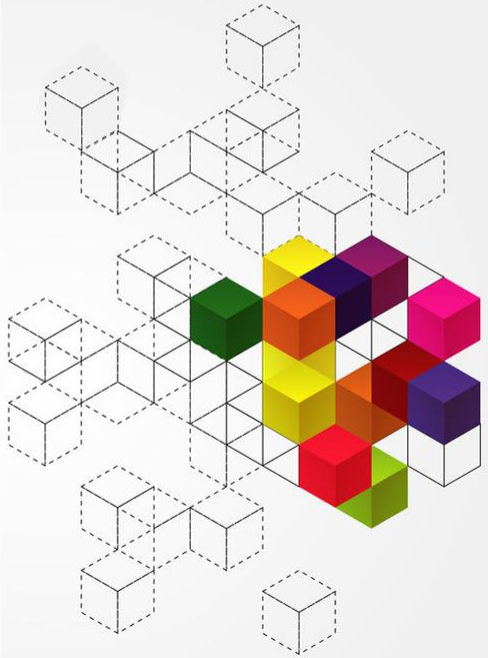
二、 内存抖动

三、 工作集模型简介

四、 工作集模型实现

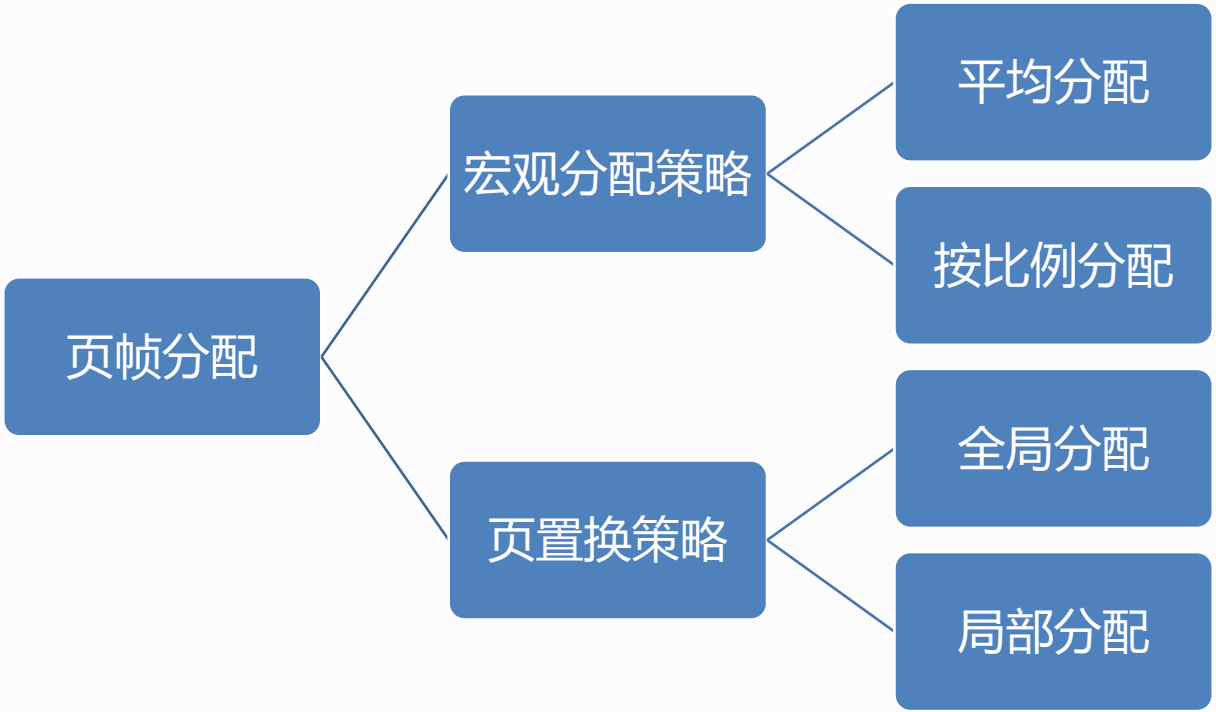
五、 页错率频率

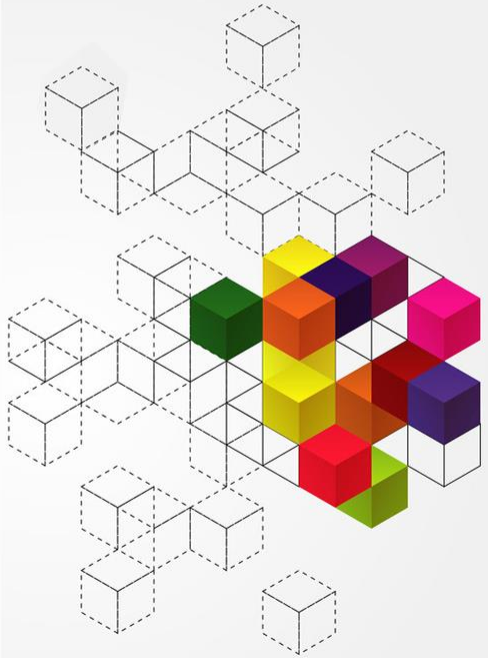
六、 内存映射文件



一、页帧分配原则

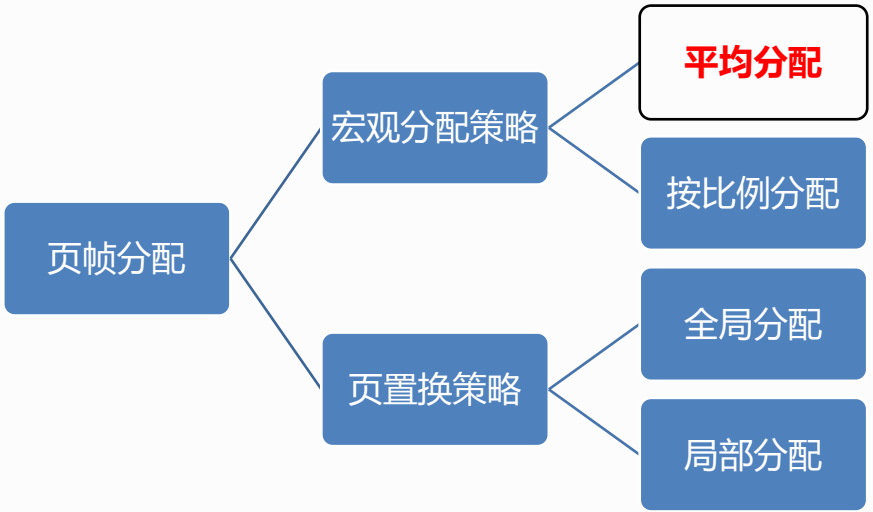
- 要解决的基本问题：**如何为每个进程分配物理内存，应该分配多大的物理内存?**



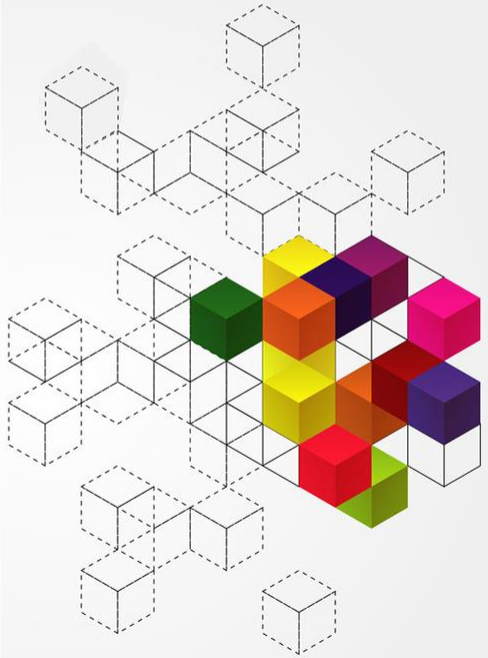


一、页帧分配原则

- 宏观分配策略1：**平局分配（Equal Allocation）**
  - **每个用户进程被分配相同数量的页帧**



如有100个帧
5个进程，
则每个进程20帧



一、页帧分配原则

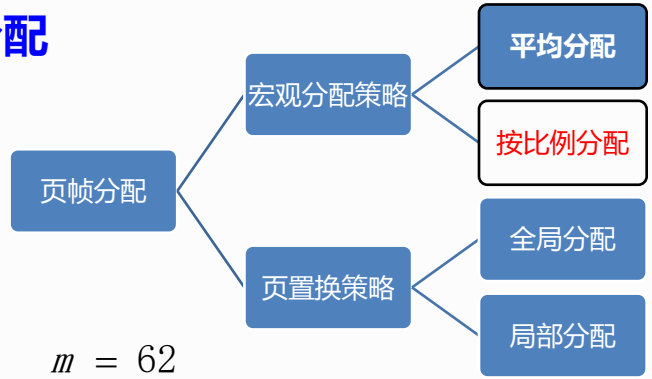
- 宏观分配策略2：**按比例分配（Priority Allocation）**
  - **按照进程的大小，成比例分配**

s_i = size of process p_i

$S = \sum s_i$

m = total number of frames

a_i = allocation for $p_i = \frac{s_i}{S} \times m$



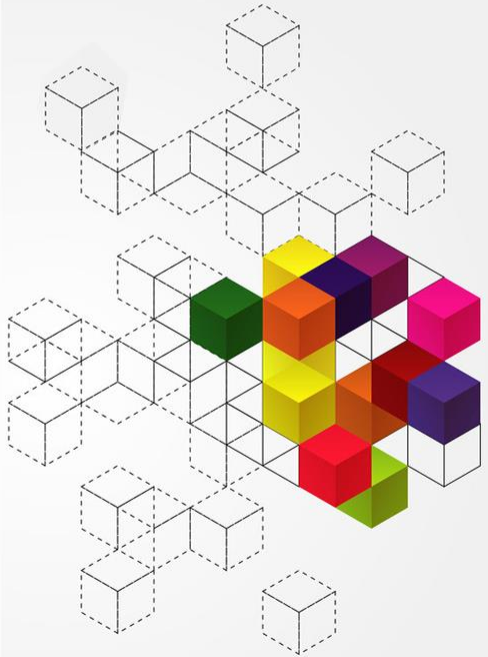
$m = 62$

$s_1 = 10$

$s_2 = 127$

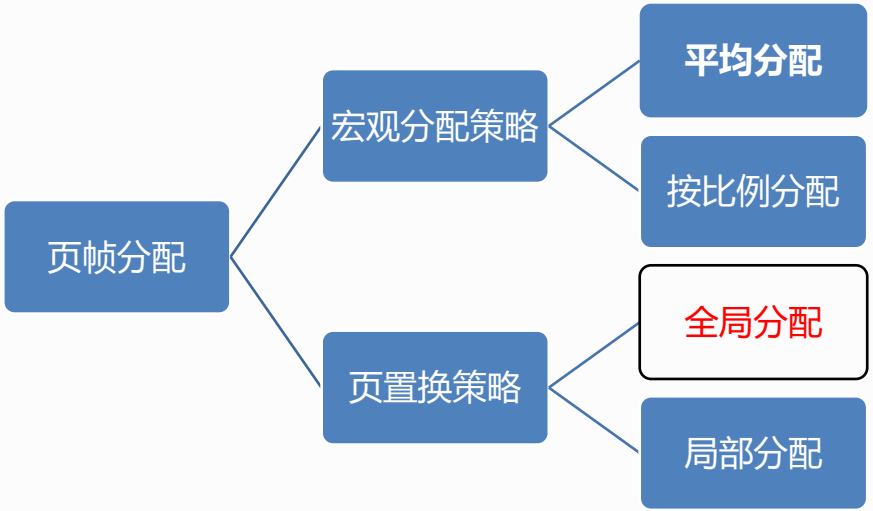
$a_1 = \frac{10}{137} \times 62 \approx 4$

$a_2 = \frac{127}{137} \times 62 \approx 57$

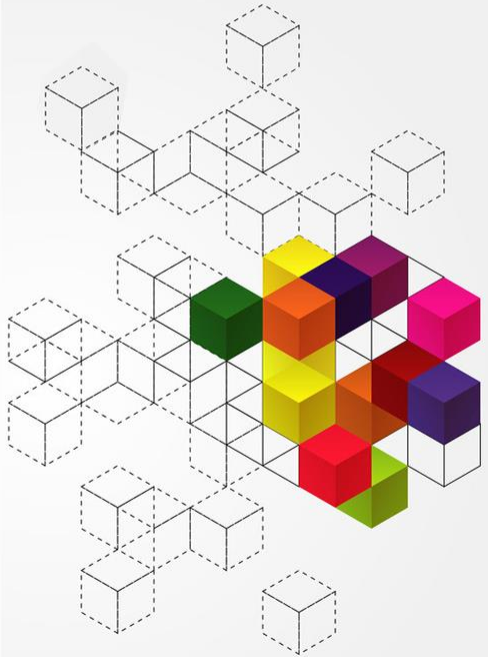


一、页帧分配原则

- 页置换策略1：**全局置换（Global Replacement）**
  - **当需要进行页置换时，选择牺牲页帧的范围为所有进程的已分配物理页框集合**

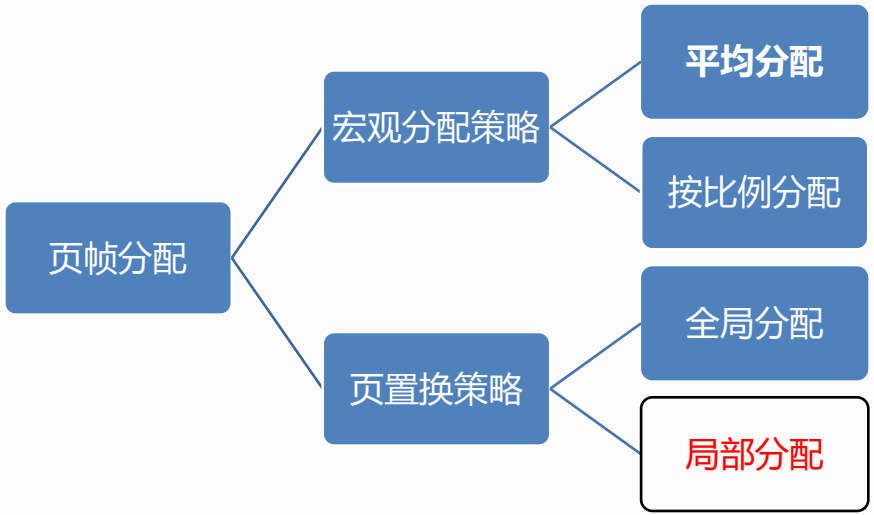


进程的执行时间可能有非常大的变化（依赖于其他进程）

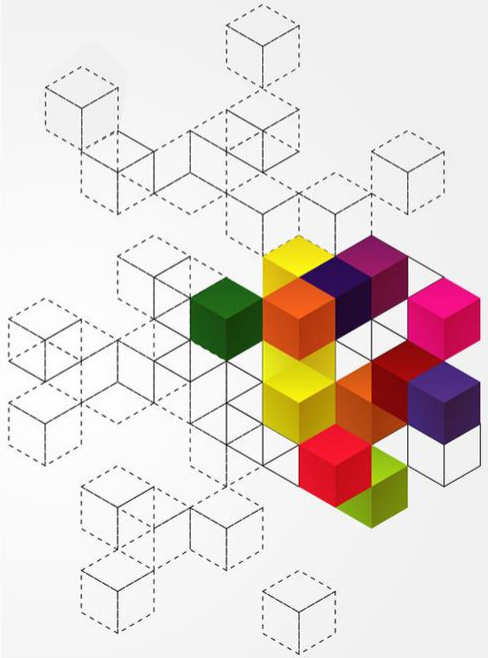


一、页帧分配原则

- 页置换策略2：**局部置换（Local Replacement）**
  - **当需要进行页置换时，选择牺牲页帧的范围为当前发生置换的进程的已分配物理页框集合**

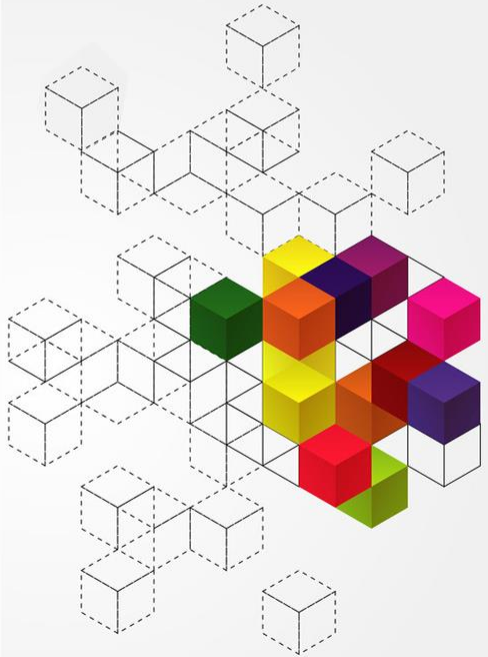


具有更一致的性能表现（可控制自己的行为）



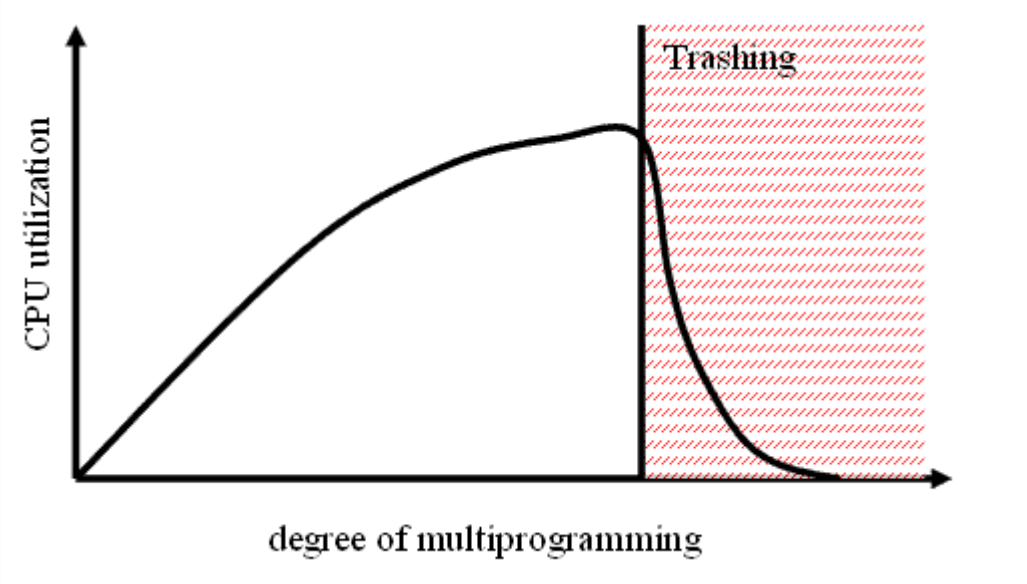
二、内存抖动

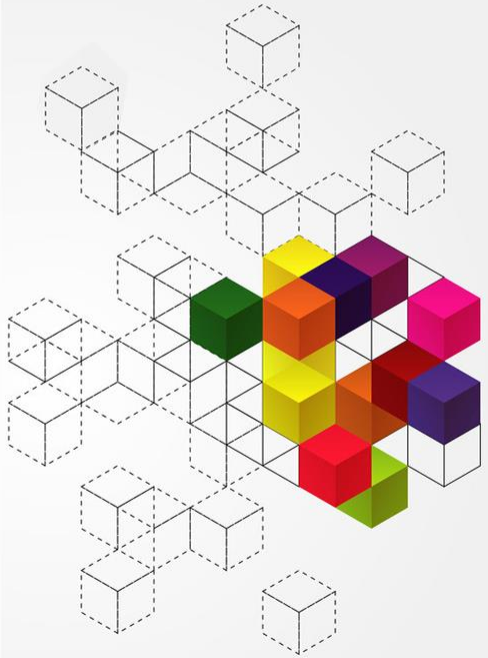
- **Thrashing**
  - 当进程分配的物理内存不足时，导致内存与外存之间的数据交换异常增加的现象，被称为抖动
  - 具体来说，如果进程无足够的页面，页错误率很高
    - 产生页错误、替代已有帧、但替代的帧又被访问，导致：
      - 低CPU使用率
      - OS认为需要增加新任务（multiprogramming）
      - 另一个进程被加入系统
  - 颠簸：进程忙于页面的换入换出，无实质执行，严重的性能问题



二、内存抖动

- **Thrashing**
  - 当进程分配的物理内存不足时，导致内存与外存之间的数据交换异常增加的现象，被称为抖动



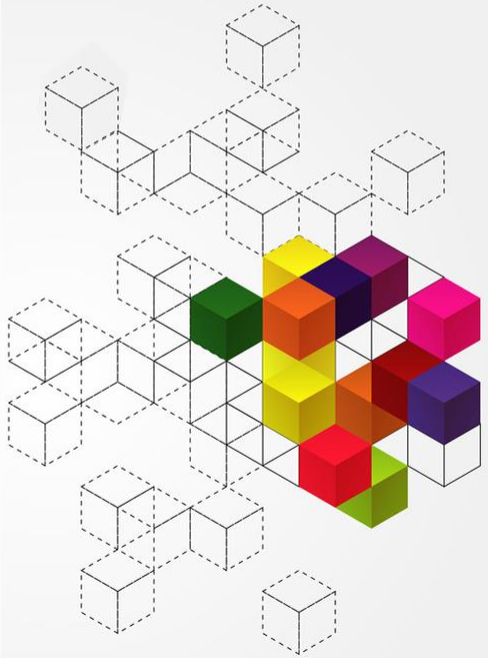


三、工作集模型简介

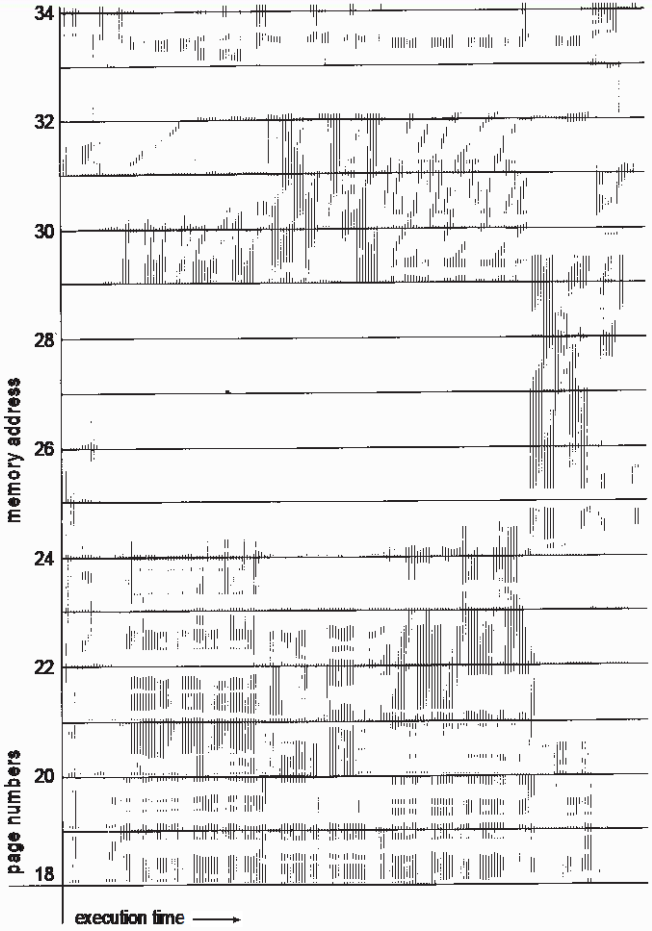
- **为了避免抖动，同时又能够进行有效的物理内存分配，需要对进程局部性进行建模**

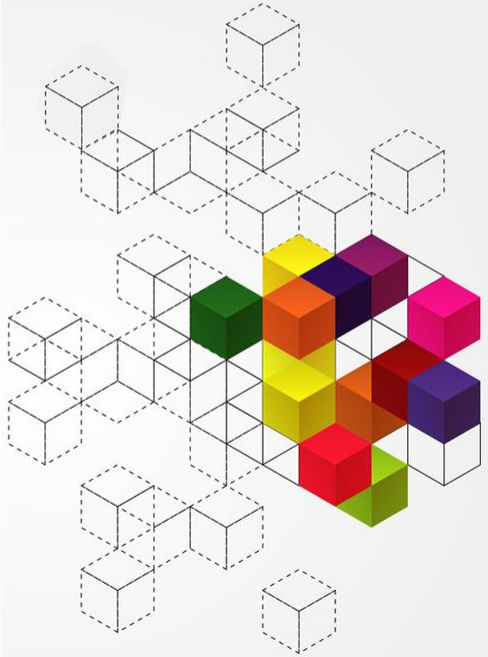
工作集模型是一种局部性建模方法

- **工作集模型基本思想：观察定长时间段内进程访问内存集合，并以此为依据确定该时间段内为该进程分配物理页（页帧）的数量**
  - 工作集用途：刻画程序的局部性信息



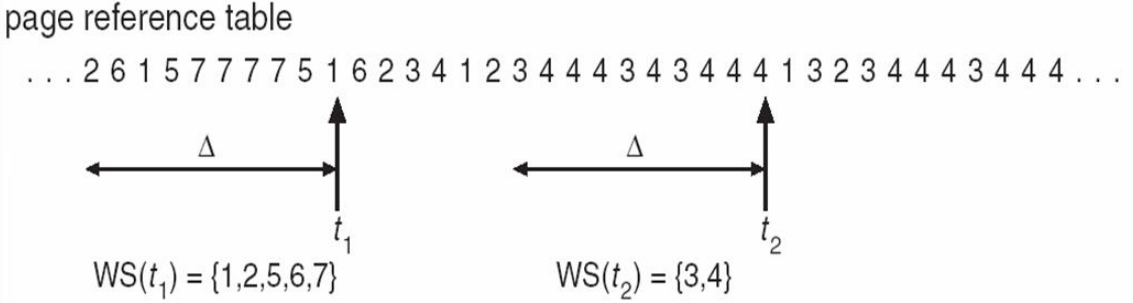
三、工作集模型简介

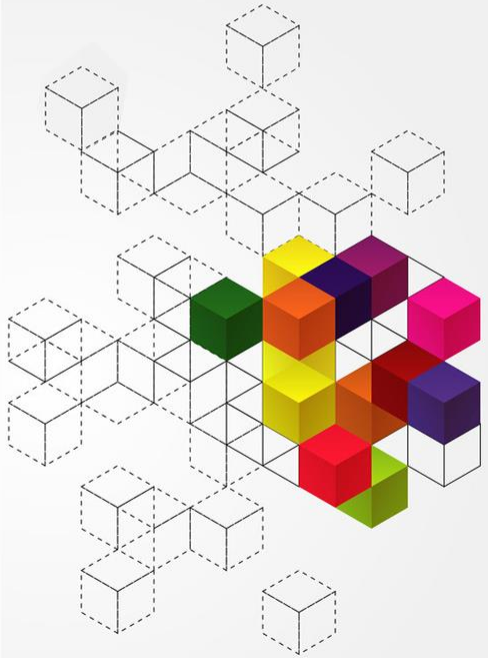

34
32
30
28
26
24
22
20
18
memory address
page numbers
execution time



三、工作集模型简介

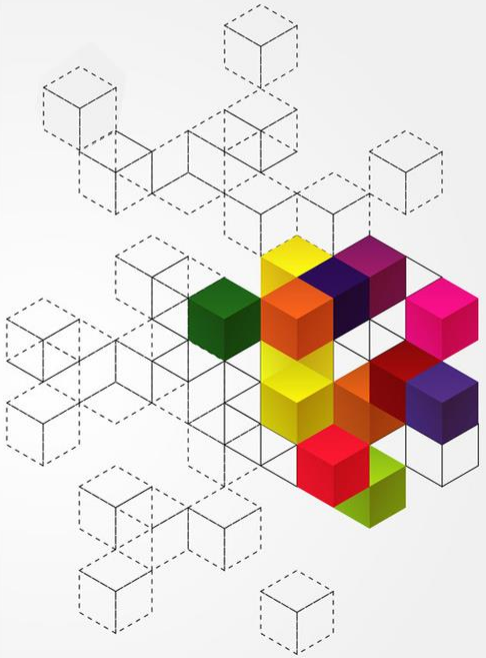
- $\Delta \equiv$ 固定数量的页面访问
- WSS_i (working set of Process P_i) = 在 Δ 内访问的页数量
  - 如果 Δ 过小，则无法包含整个的局部性
  - 如果 Δ 过大，则可能包含多个局部性
  - 如果 $\Delta = \infty \Rightarrow$ 则包含全部程序
- $D = \Sigma\ WSS_i \equiv$ 所有需要的帧 （**局部性的近似**）
- 如果 $D > m \Rightarrow$ 颠簸 (m 为可用帧的数量)
- 策略: 如果 $D >$ m, 则挂起或换出某个进程





四、工作集模型实现

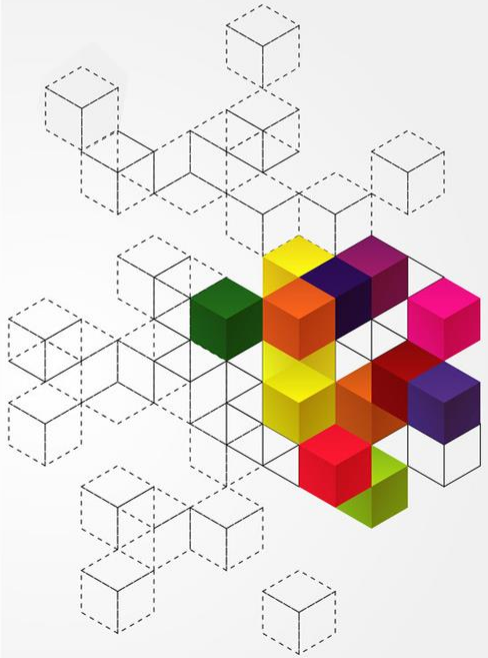
- **实现工作集模型的关键在于：如何随着时间的推移，持续跟踪进程工作集**
- 实现难点：
  - 过于频繁的工作集更新，会对程序执行性能产生较大负面影响



四、工作集模型实现

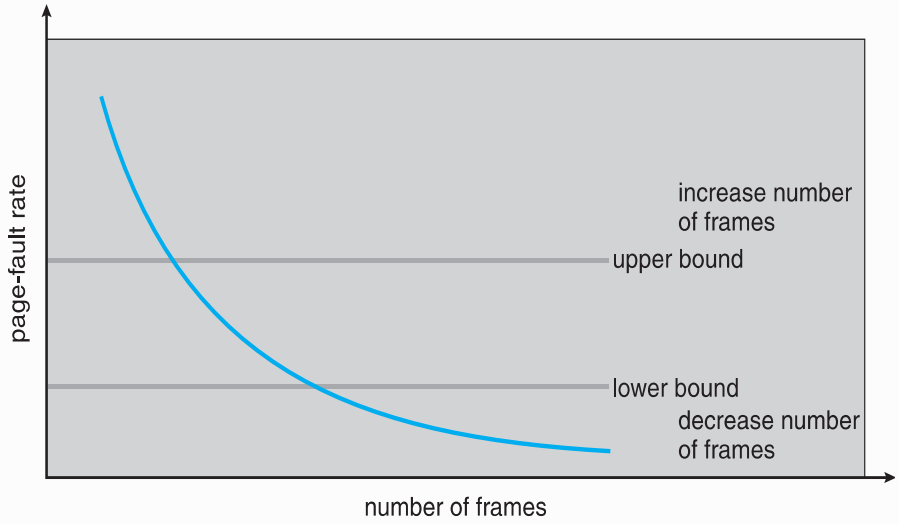
- 实现策略：减少采样频率
- 实际做法：计时器 + 引用位
- 关键参数：工作集窗口Δ，时钟中断间隔ti

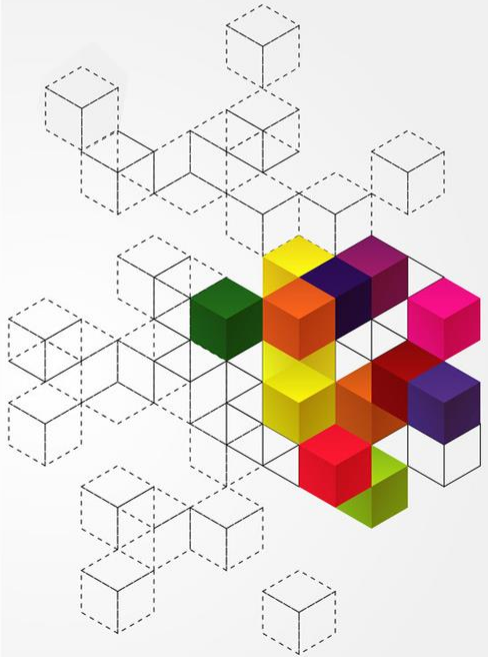
Δ=10000，ti=5000
为每个页面维护一个引用位，以及2个额外的位
2个额外的位用来记录在过去的两个长为5000的时间区间内页面有没有被引用
时钟中断发生时，如果这3个位至少有一个值为1，则对应的页面在工作集中



五、页错误频率

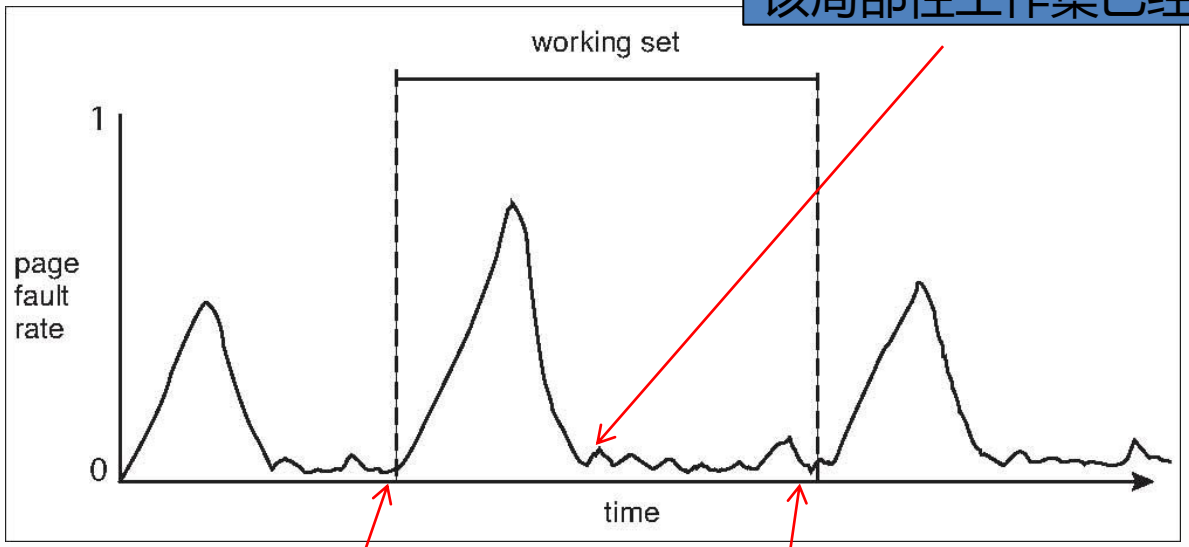
- 比工作集模型更加直接
- 建立“可接受的”页错误评率 page-fault frequency (PFF)，采用局部替换方法
  - 如果错误率过低，则减少进程的帧
  - 如果错误率过高，则增加进程的帧





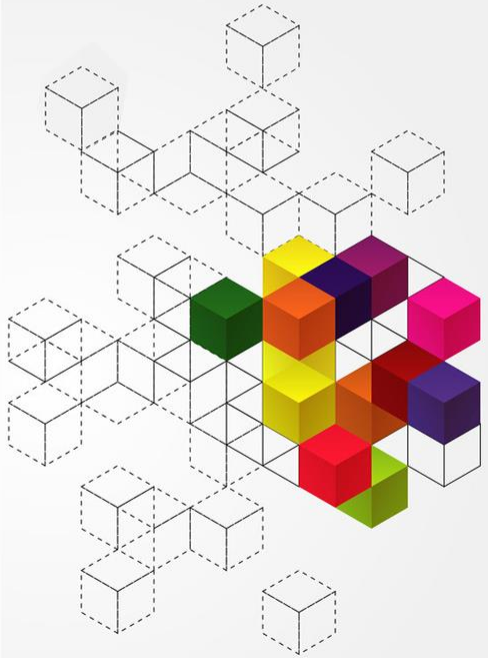
五、页错误频率

- 工作集与页错误率有直接的关系



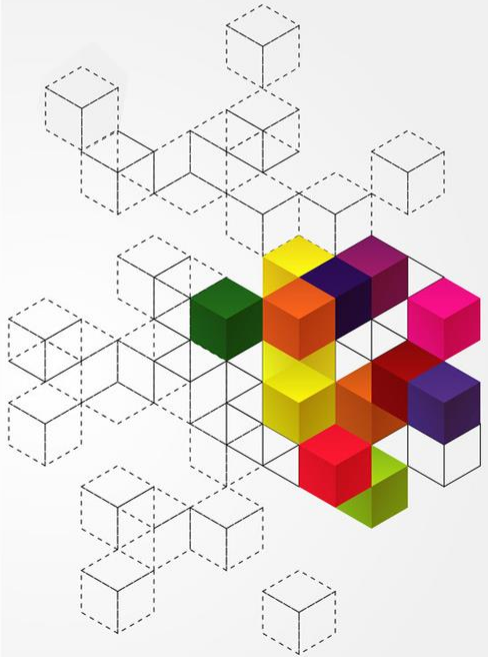
该局部性工作集已经在内存

开始为新局部性启动按需分页

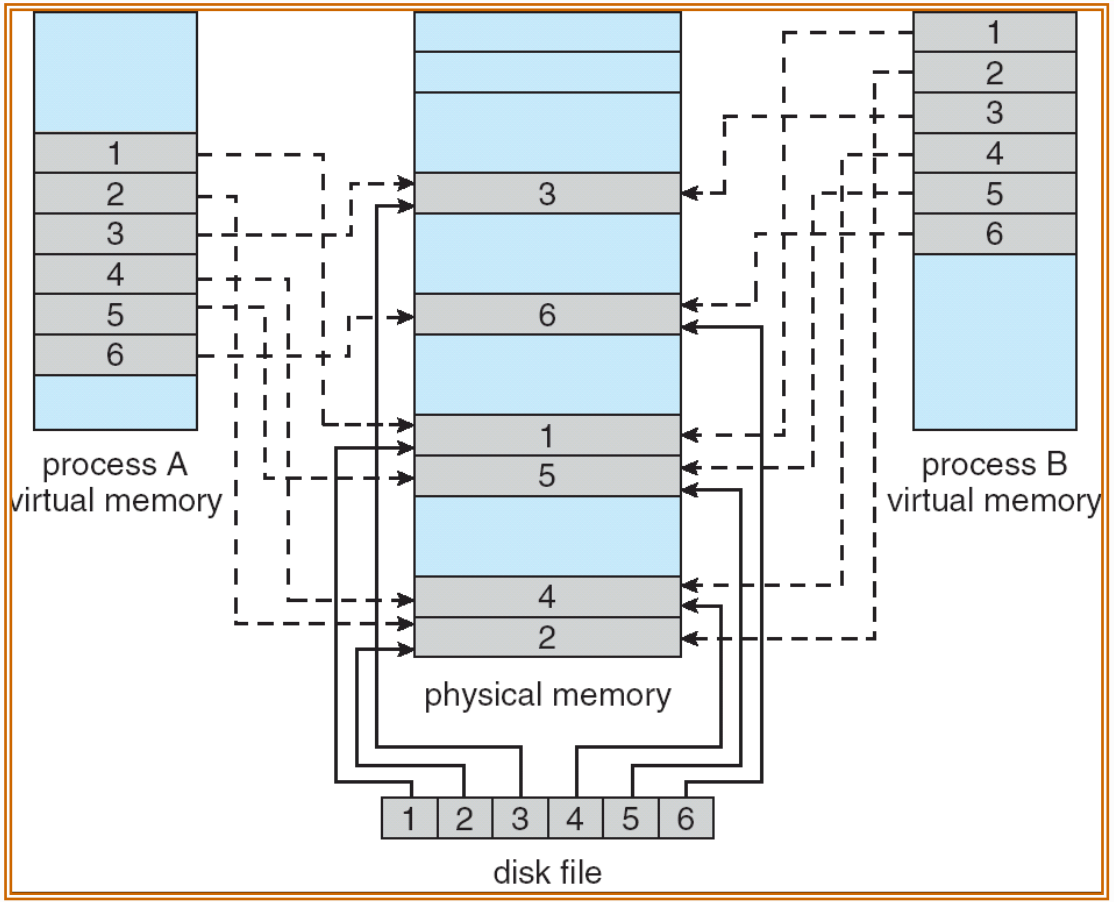


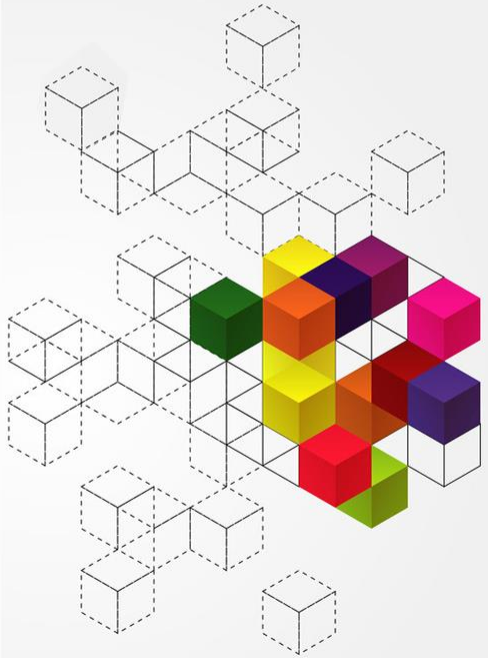
六、内存映射文件

- 内存映射文件（Memory-mapped files）允许文件I/O被当做内存访问处理——将磁盘文件块映射到内存中的页（pages）
- 初始时使用按需分页读取文件，将文件按页面的形式读至物理内存，后续对文件的读/写操作被看做是正常的内存操作。
- 简化了基于read()/write()的文件I/O操作
- 亦允许多个进程将相同的文件映射到内存中相同的帧进行共享。



六、内存映射文件


1
2
3
4
5
6
process A
virtual memory
3
6
1
5
4
2
physical memory
1
2
3
4
5
6
process B
virtual memory
1
2
3
4
5
6
disk file



本讲小结

- 页帧分配策略及实现方法