操作系统

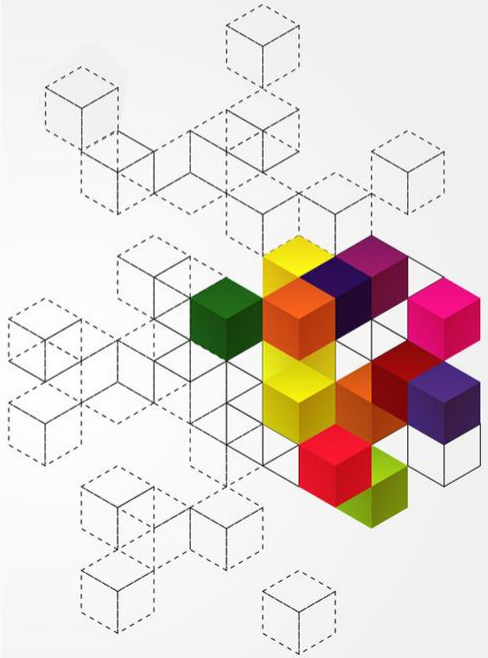
Operating system

胡燕

大连理工大学

主存管理回顾

分区分配 - 碎片问题
分页 - 分页硬件 – 页表
分段 - 段表



内容纲要

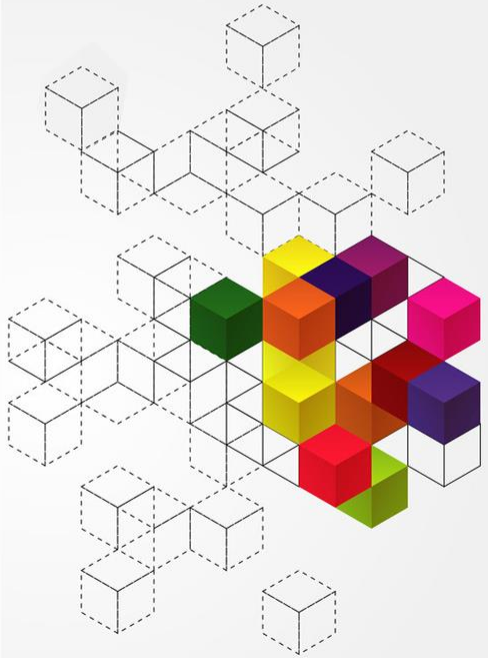
9.1 虚存概念

一、 虚存思想

二、 引入虚存的意义

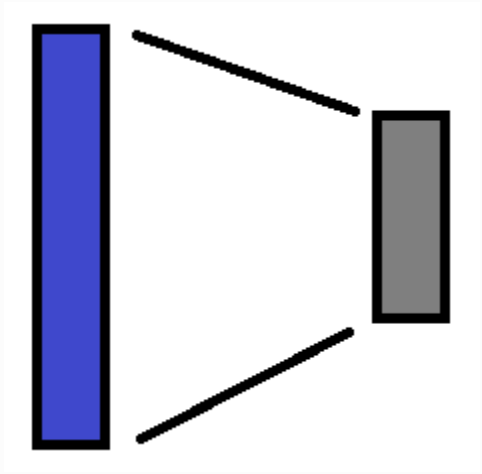
三、 按需调页

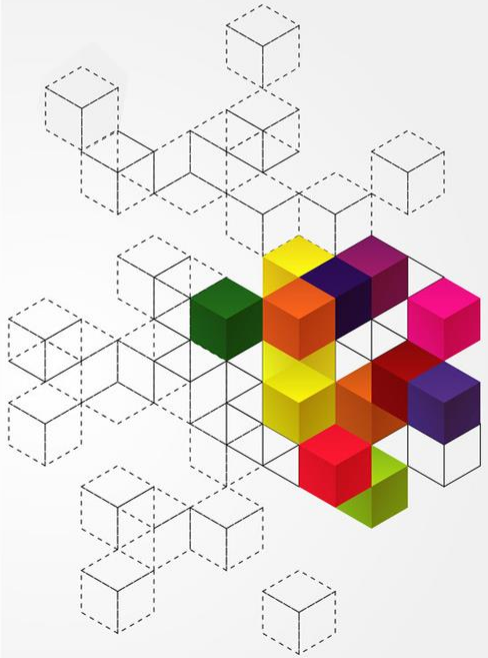
四、 虚存所需硬件支持



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

大逻辑空间

小物理空间

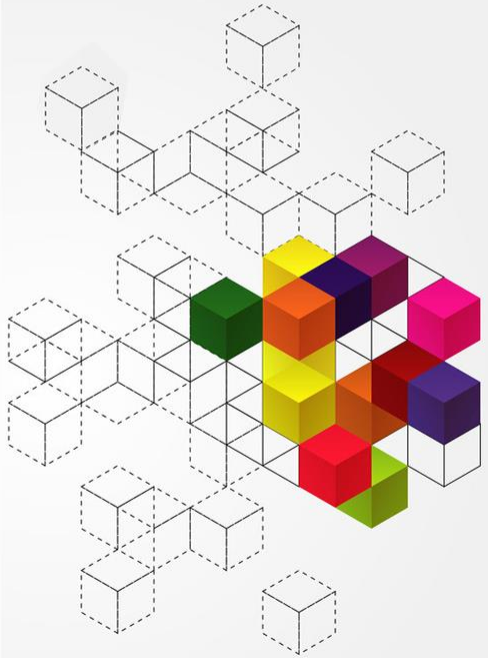
支点：局部性原理

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

程序的局部性原理：程序在执行时呈现出局部性规律，即在一段时间内，整个程序的执行仅限于程序中的某一小部分

课后任务：程序局部性分为哪几类，请进行调研并形成总结报告。（下节课5.20，课前留些时间请几位同学利用word文档或ppt的形式台上分享）



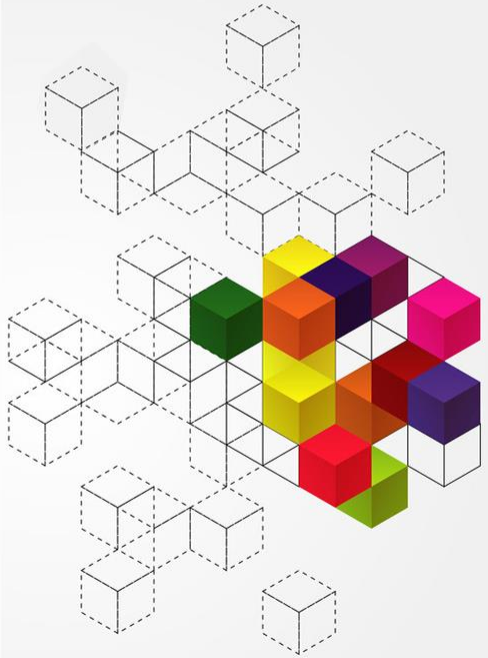
一、虚存思想

- **实现虚存机制，所需解决的核心问题**
  - 如何用少量的物理内存，支撑起大容量的逻辑地址空间

解决手法：

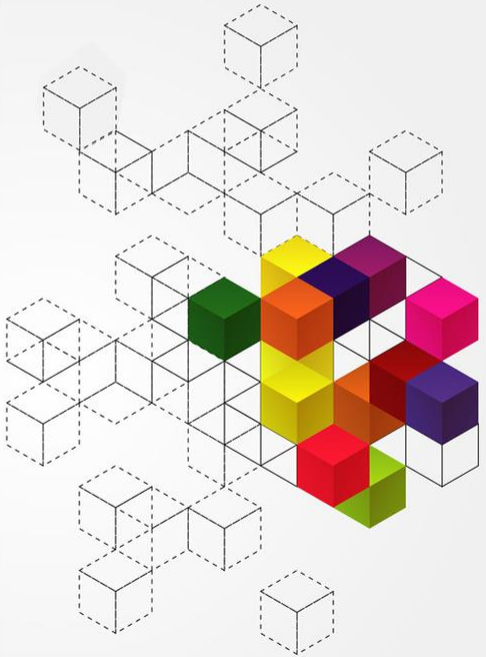
按需分配　demand driven

惰性加载　lazy loading



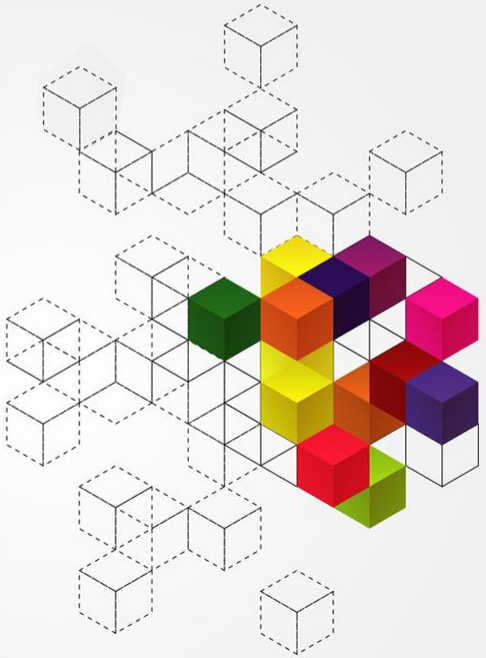
一、虚存思想

- **虚存机制的基本工作原理**
  - 按需分配：进程执行过程中，根据当前执行的局部，根据需要分配物理内存空间
  - 惰性加载：
    - 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）
    - 仅加载所需的逻辑空间内容到为进程分配的物理内存



二、引入虚存的意义

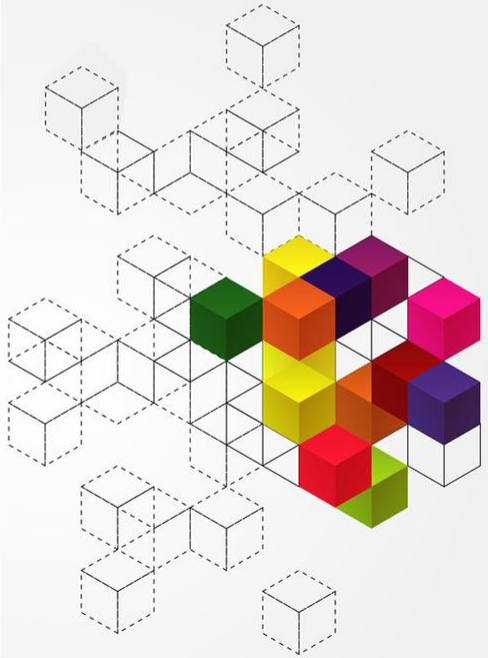
- **虚存机制的优势**
  - 不再需要一次性加载程序的所有代码和一次运行所需的所有数据，可以节省大量物理内存
  - 使得进程的创建更有效率
  - 可以较少IO操作
  - 支持更多的并发进程
  - 方便实施在进程间共享物理内存



二、引入虚存的意义

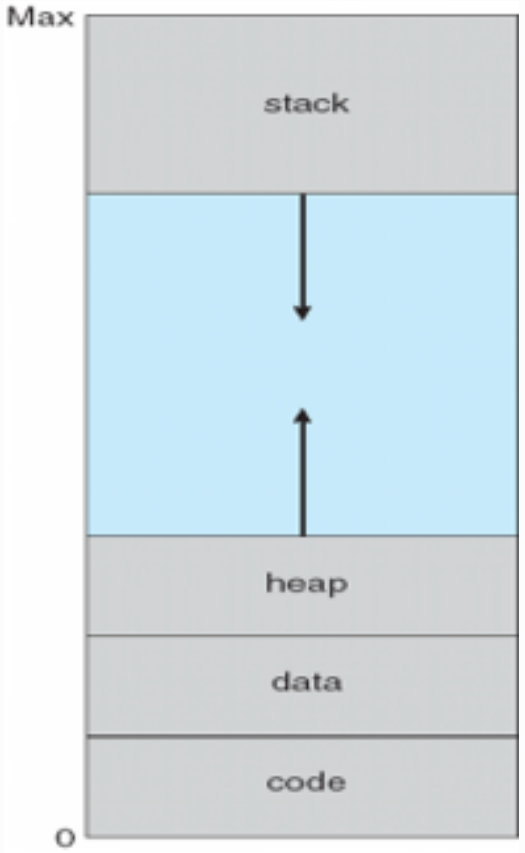
- **慕课堂讨论1：谈谈对操作系统中引入虚存所带来优势的理解**

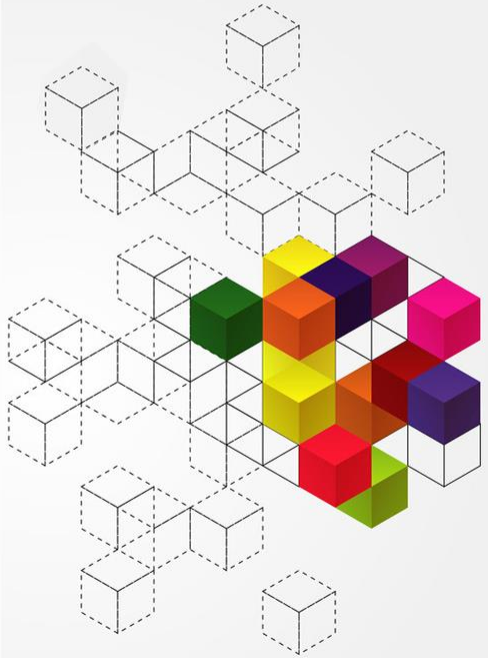
引入虚存的优势：
（1）不再需要一次性加载程序的所有代码和一次运行所需的所有数据，可以节省大量物理内存
（2）使得进程的创建更有效率
（3）可以较少IO操作
（4）支持更多的并发进程
（5）方便实施在进程间共享物理内存
请谈谈对这些方面你的理解。



二、引入虚存的意义

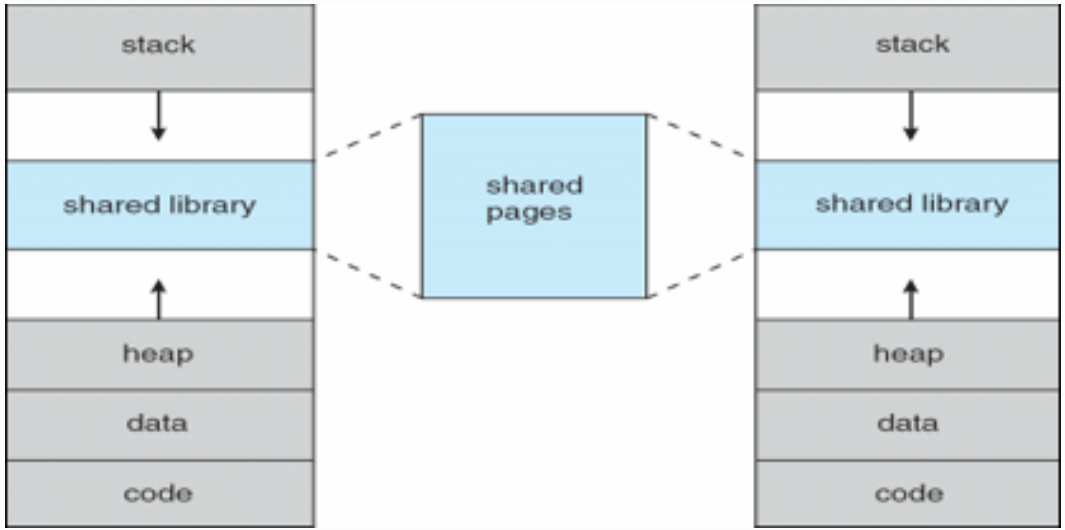
- **形成较大的虚地址空间**
  - 开发人员进行应用开发时，不需要太多考虑对逻辑地址空间大小进行限制的问题
  - 通常，包含代码、数据、运行时堆和栈的逻辑地址空间都会存在较大富余

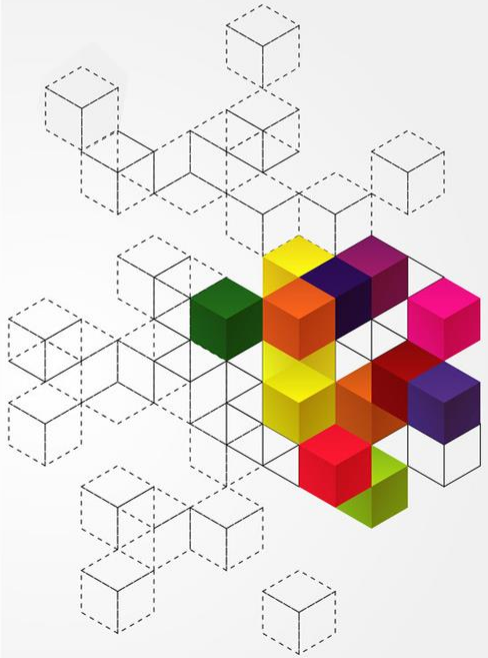




二、引入虚存的意义

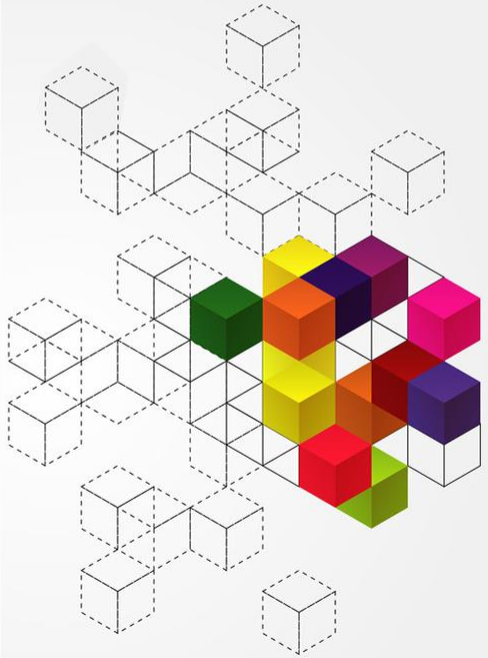
- **便于进程间进行内存共享**
  - 可以将相同一块物理内存区域映射到不同进程的逻辑地址空间：仅需将需共享的物理页内容映射到进程的虚地址空间即可





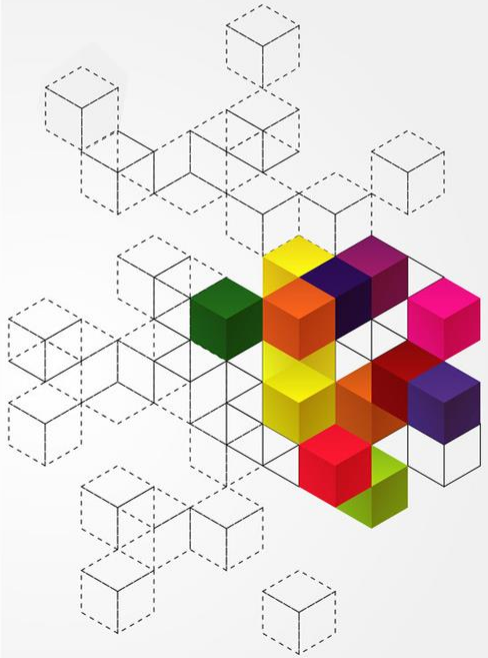
三、按需调页

- **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)
  - 合适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



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
  - 指令重启



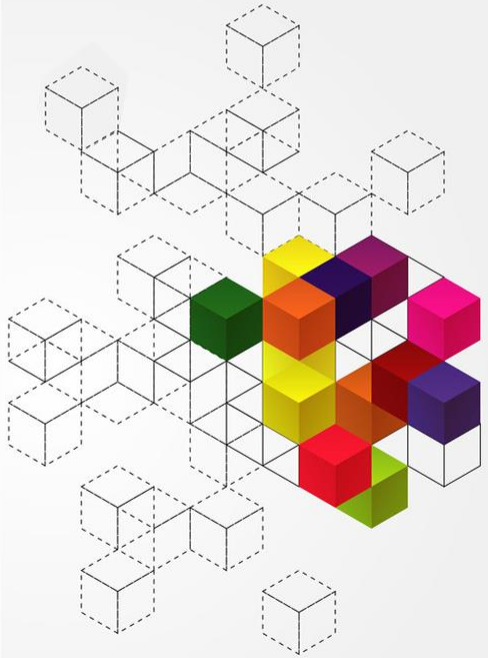
四、虚存所需硬件支持1：页表有效位

Frame #	valid-invalid bit
	v
	v
	v
	v
	i
….	
	i
	i

page table

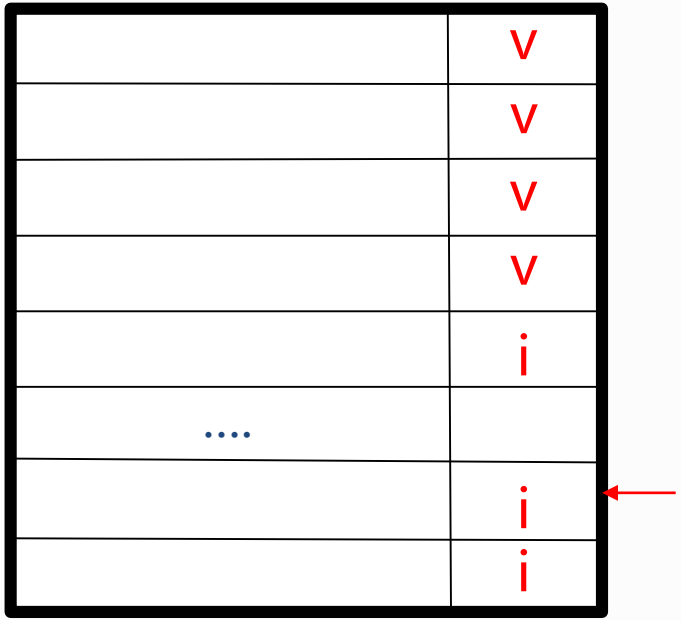
虚存机制下，进程逻辑地址空间内多数为invalid页面

页地址翻译过程中，需要MMU在地址翻译过程中根据有效位进行判断逻辑页面是否在物理内存

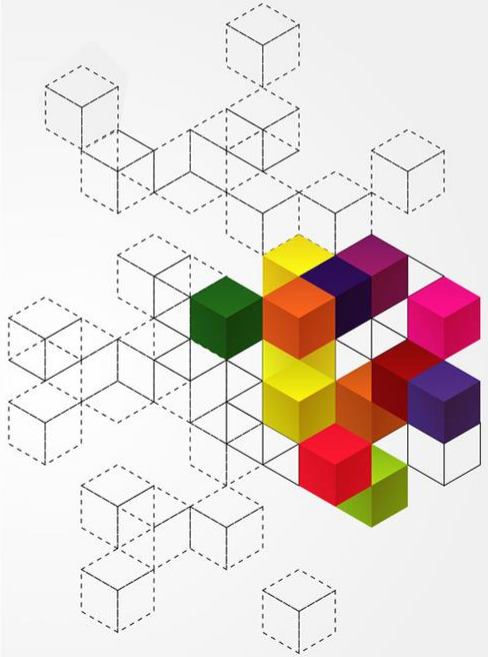


四、虚存所需硬件支持

- **慕课堂讨论2：当CPU访问某个页表状态标为i的页时候，系统应如何处理?**

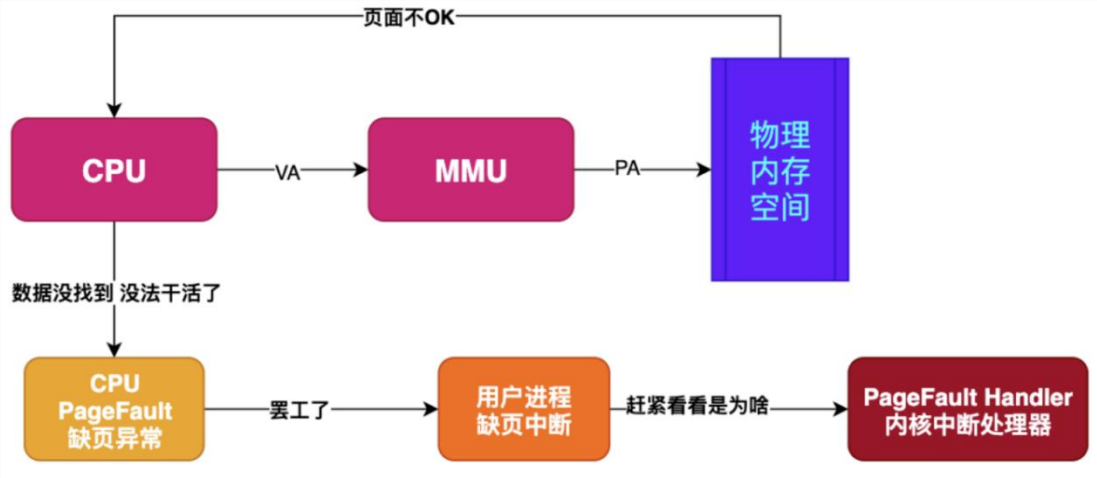


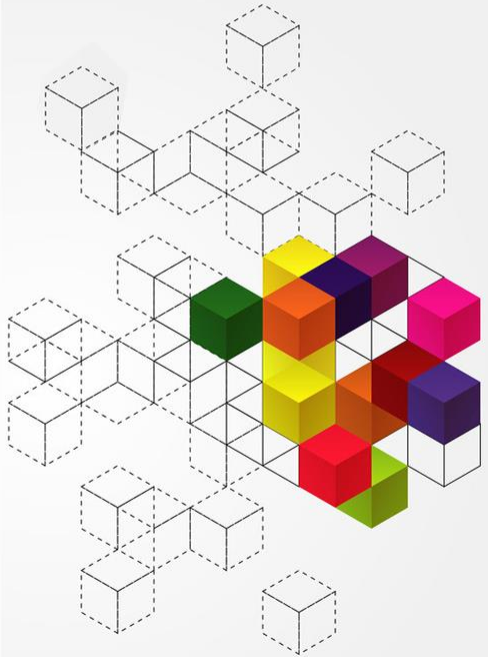
page table



四、虚存所需硬件支持

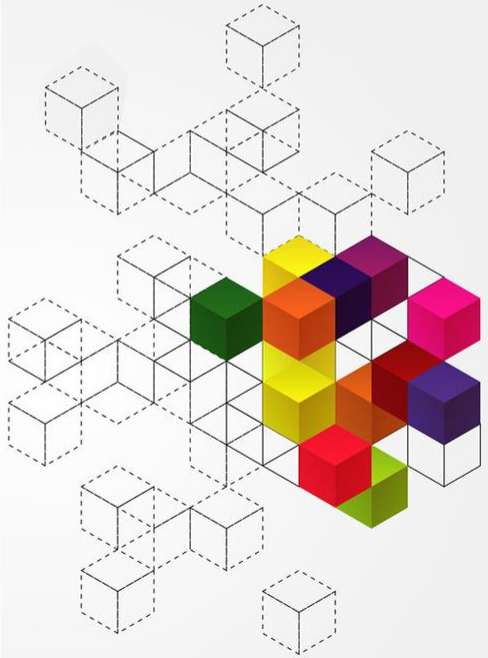
- **Page Fault(页故障，缺页异常)**
  - 当CPU访问有效位标为i的地址时，产生缺页异常





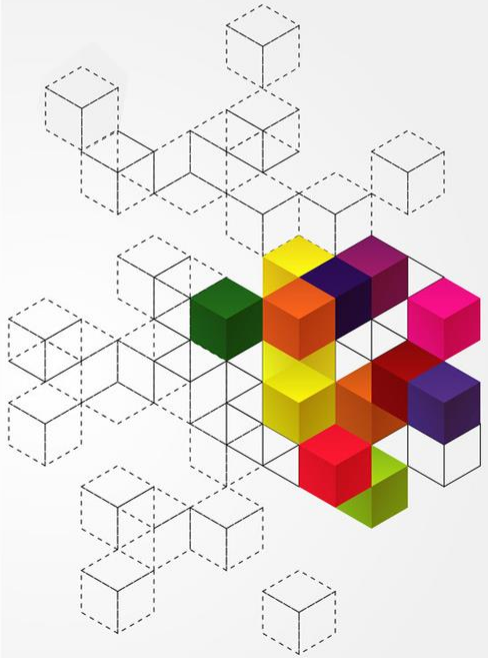
四、虚存所需硬件支持2：指令重启

- **小讨论：虚存机制中，为什么需要硬件支持指令重启?**



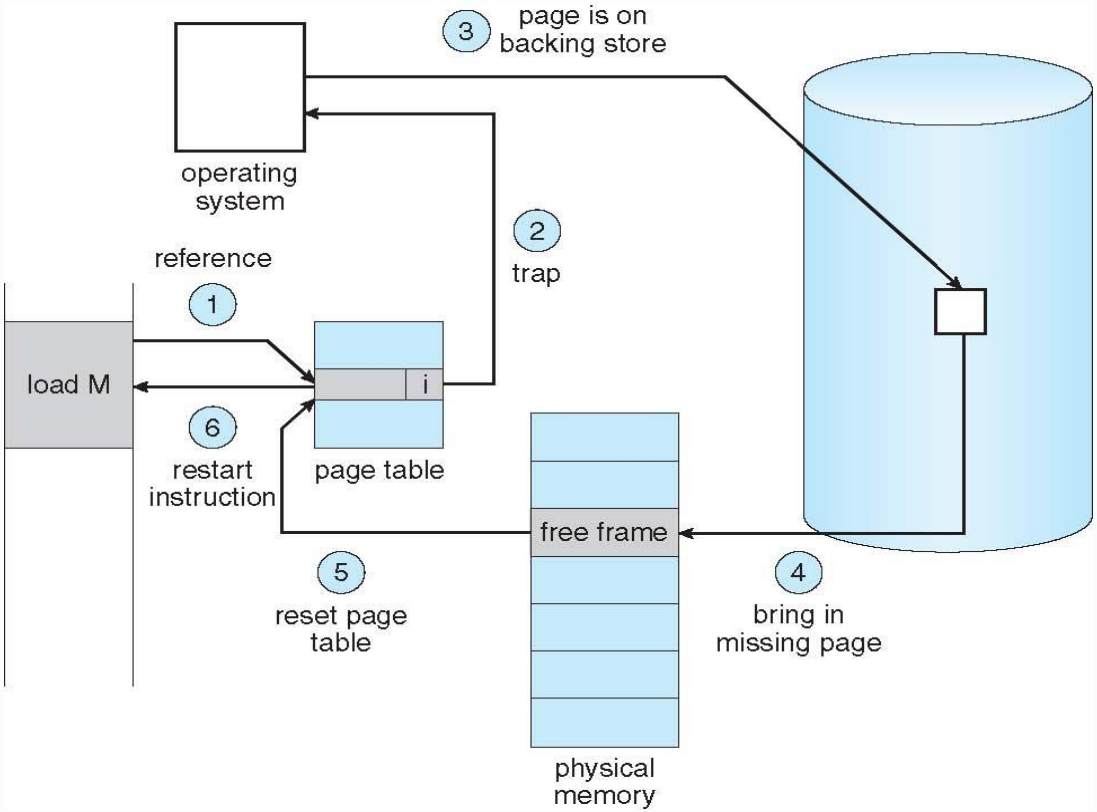
四、虚存所需硬件支持2：指令重启

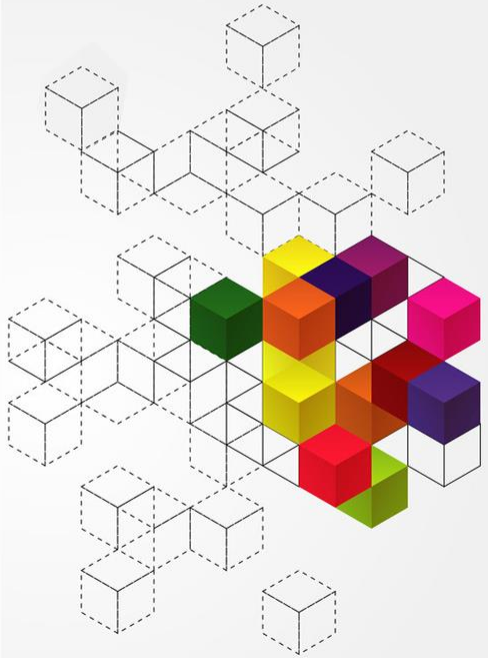
- 虚存机制下，CPU访存产生的缺页异常，大多数是因为逻辑页的内容在磁盘，而尚未被加载到物理内存
- 缺页处理过程中，会调用lazy swapper将导致异常的页面从磁盘加载到物理内存
- 因为此前访存是失败的，因此指令没有被成功执行，原因是缺页。现在不缺了，所以需要重新执行导致缺页的指令，此时它已经可以正确执行



四、虚存所需硬件支持2：指令重启

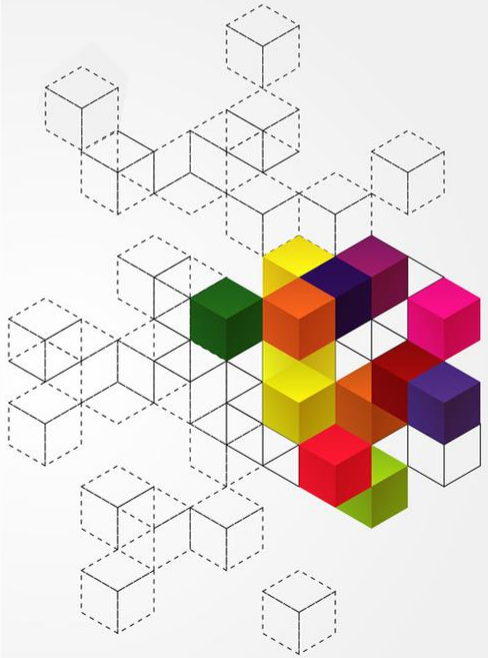
Steps in Handling a Page Fault





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



内容纲要

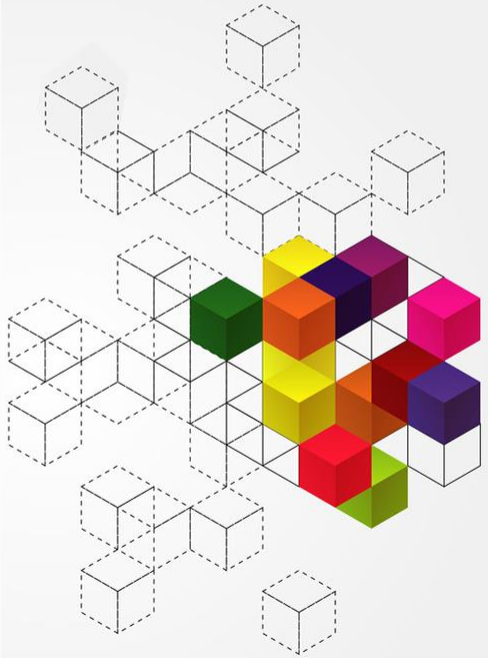
9.2 页置换算法

一、 页置换算法简介

二、 算法1：FIFO

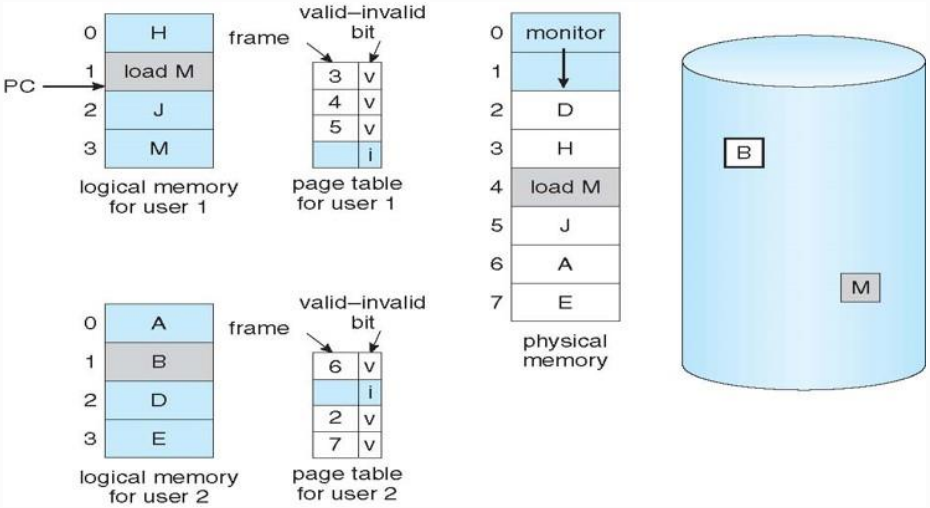
三、 算法2：OPT

四、 算法3：LRU



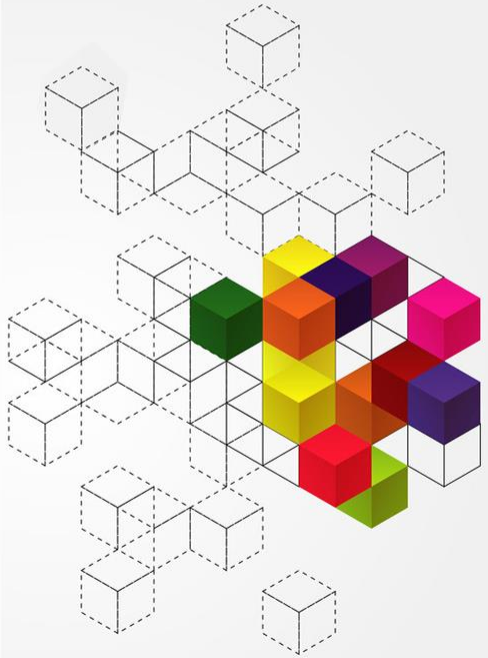
一、页置换算法简介

- **需要页置换的示例**



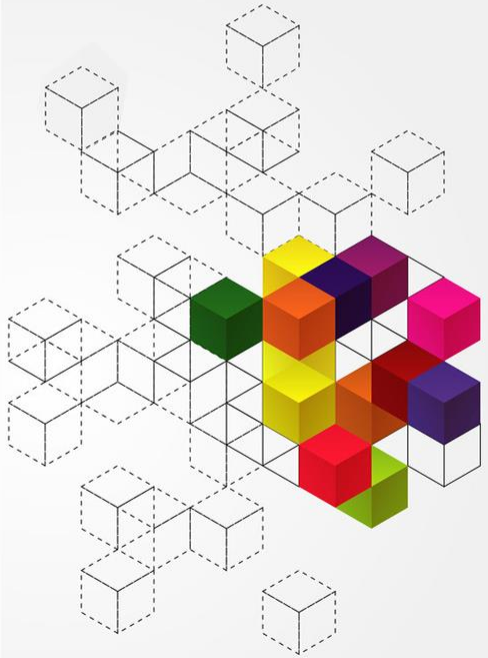
- 物理内存已满，但执行load M操作需要为逻辑页M分配物理内存，需要腾出一个已被占用的物理页

关键问题：牺牲哪个物理页



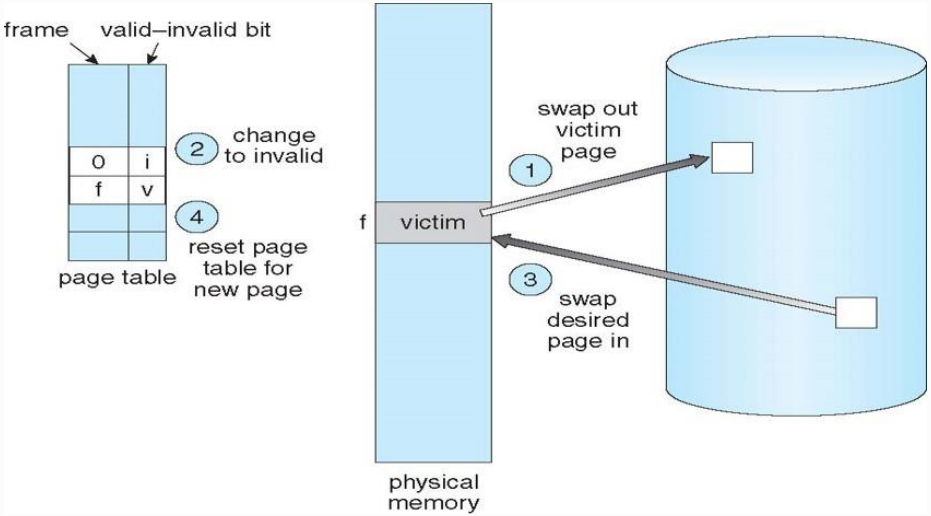
一、页置换算法简介

- **根据不同的牺牲页帧的选择策略，引入不同的页置换算法**
  - FIFO
  - OPT
  - LRU

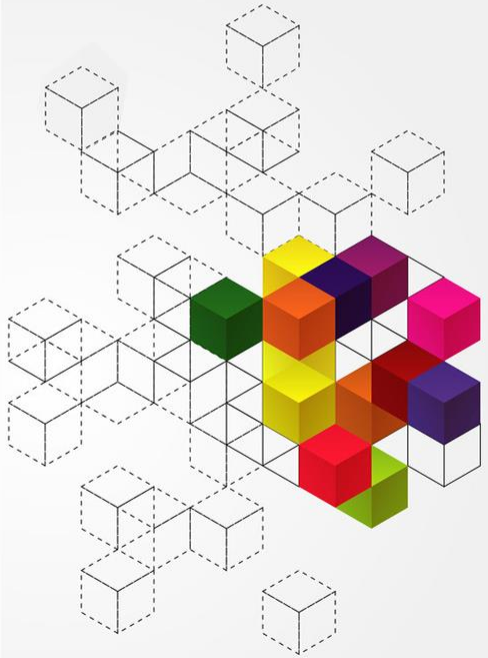


一、页置换算法简介

- **页置换关键步骤详解**

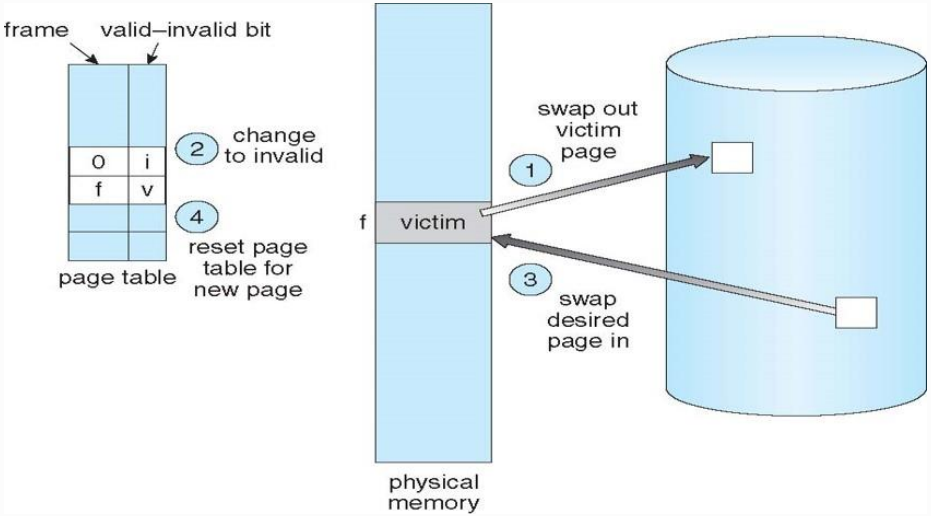


1. 选择牺牲页帧f，将其中的逻辑页换出

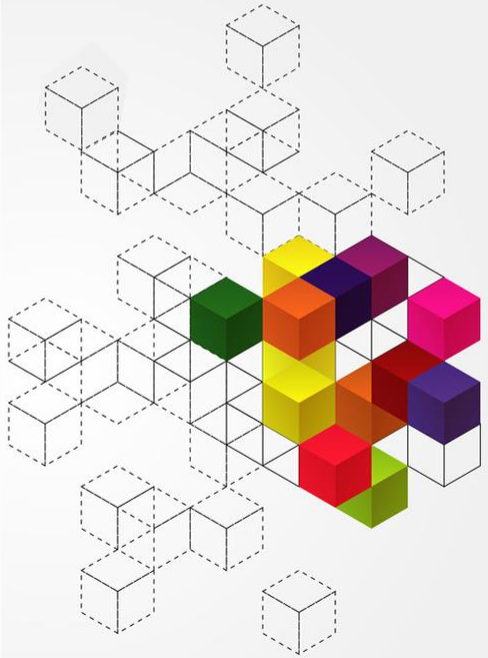


一、页置换算法简介

- **页置换关键步骤详解**

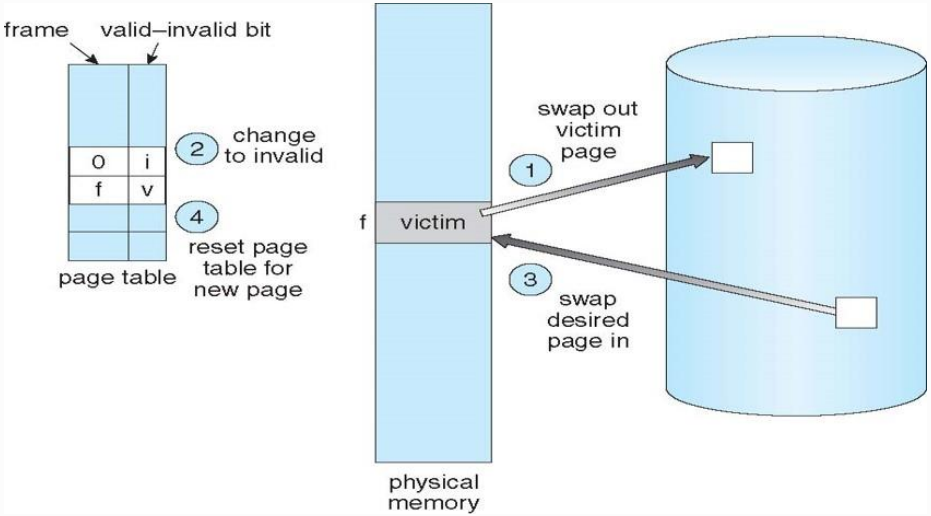


2. 将被换出的被牺牲的逻辑页对应的页表项置为invalid

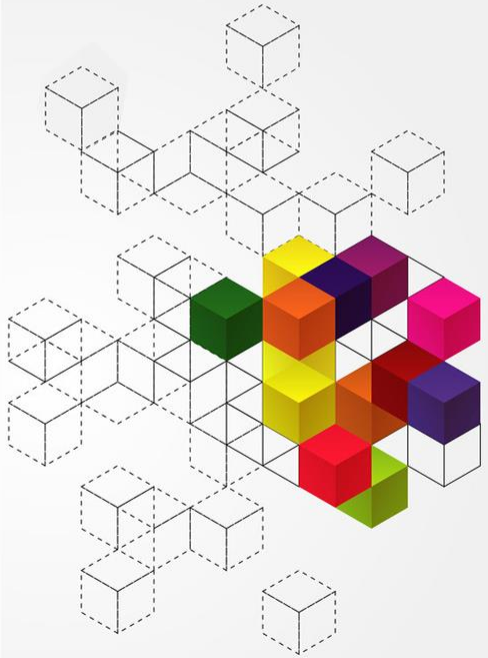


一、页置换算法简介

- **页置换关键步骤详解**

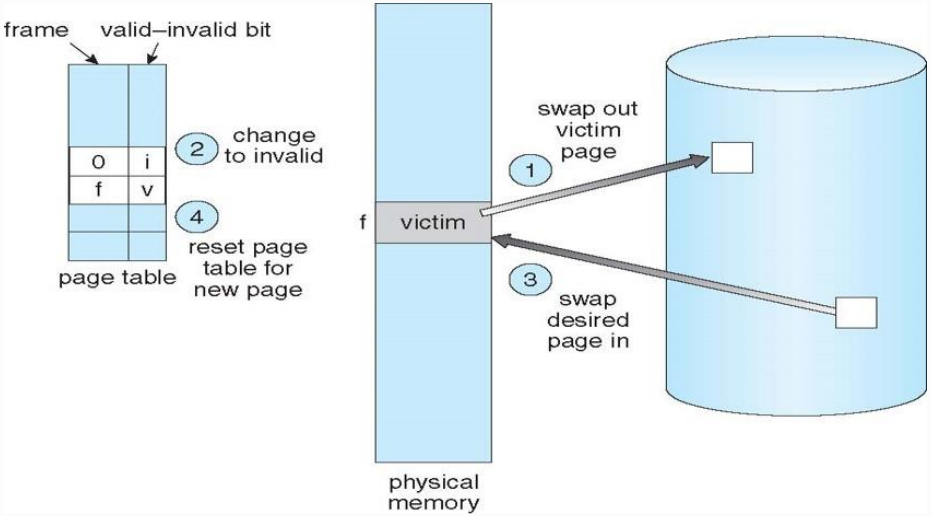


3. 将需要的页面调入页框f

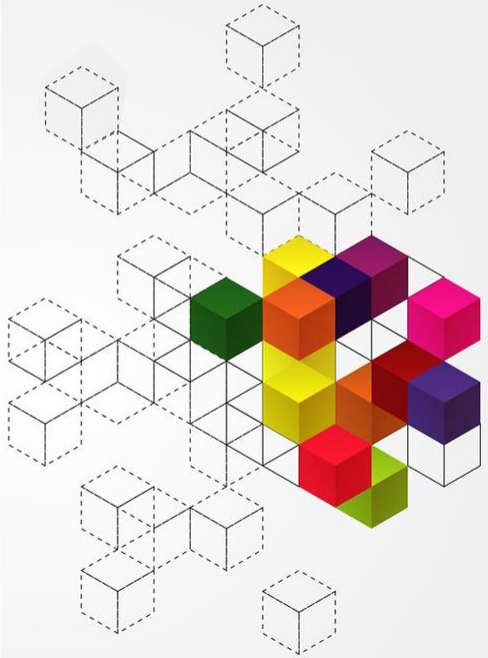


一、页置换算法简介

- **页置换关键步骤详解**

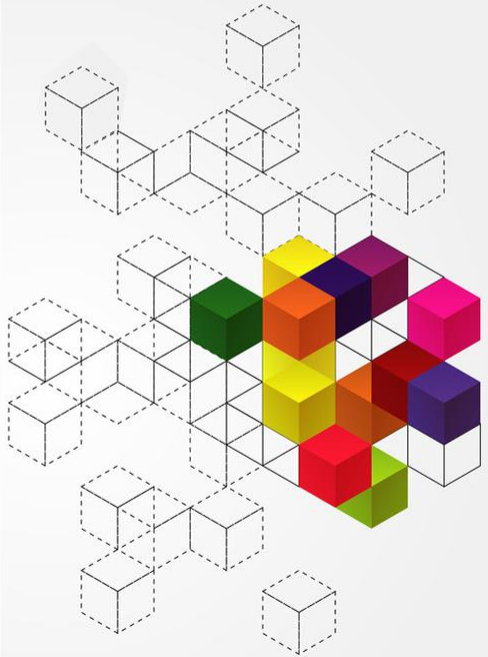


4. 更新被调入页的页表项



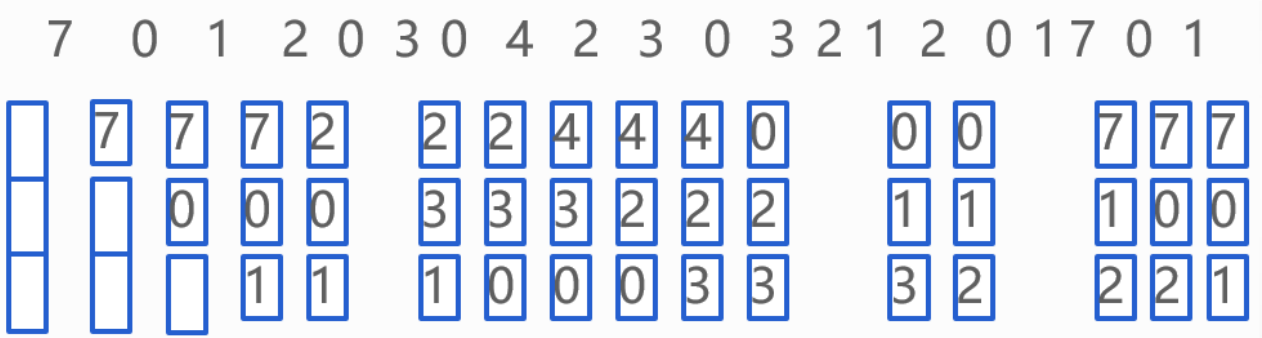
一、页置换算法简介

- **包含页置换的按需调页算法步骤**
  - 1.从磁盘上获取所需的逻辑页
  - 2.在分配页框时，如果没有空闲页框可用了，则选择一个牺牲页。如果选中的牺牲页内有数据被修改过，那么必须将页框内的数据写回磁盘
  - 3.将需要的页面调入已经空闲的牺牲帧，并同时更新页表
  - 4.重启引发页故障的指令

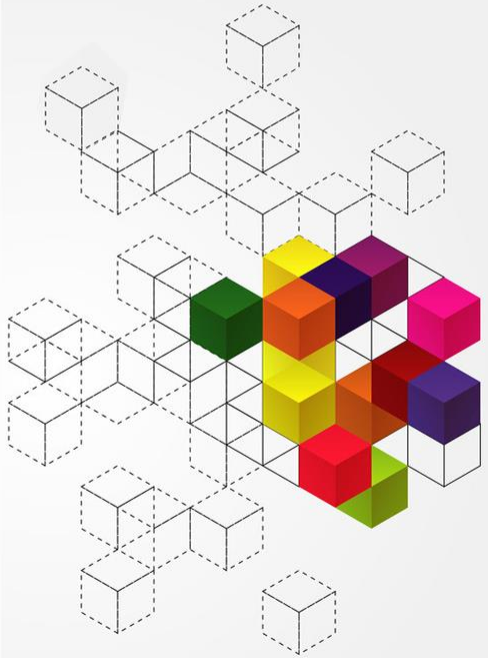


二、页置换算法1：FIFO

- **每次选择在页帧内停留最久的页面将其换出**

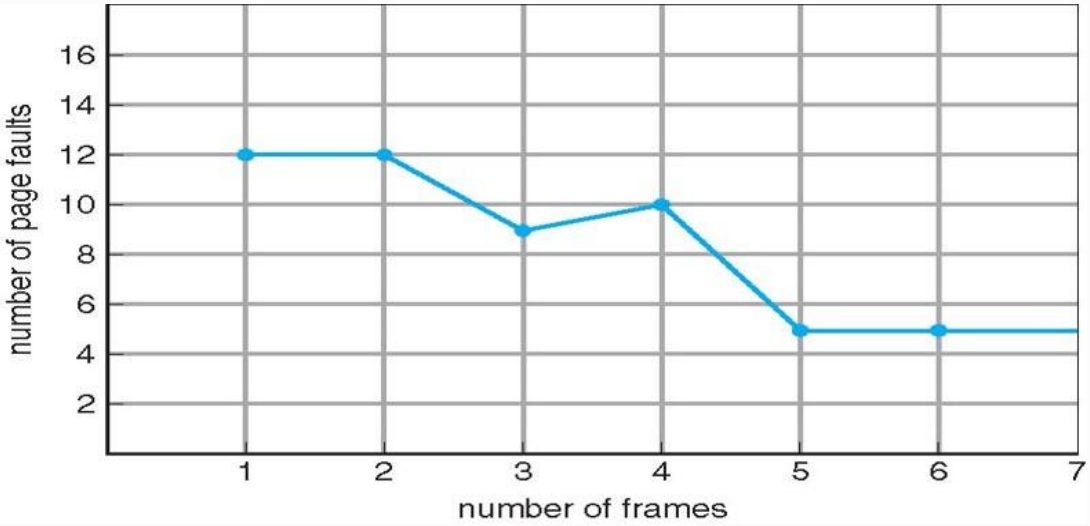


- 进程拥有3个物理页框
- 20次内存访问发生了15次页故障，其中包含12次页置换

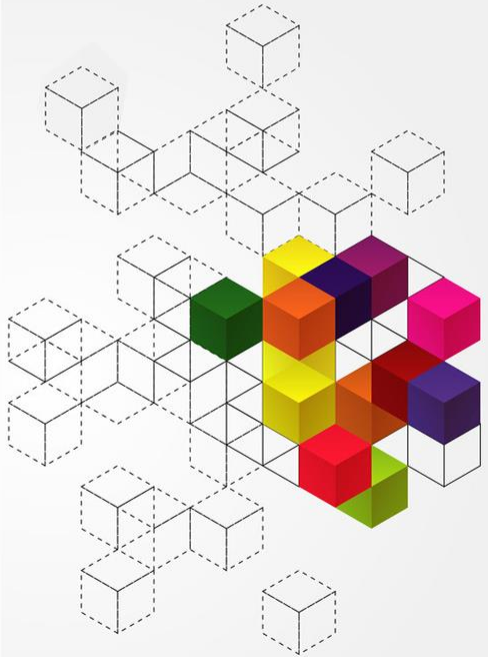


二、页置换算法1：FIFO

- **Belady异象（Belady Anomaly）**

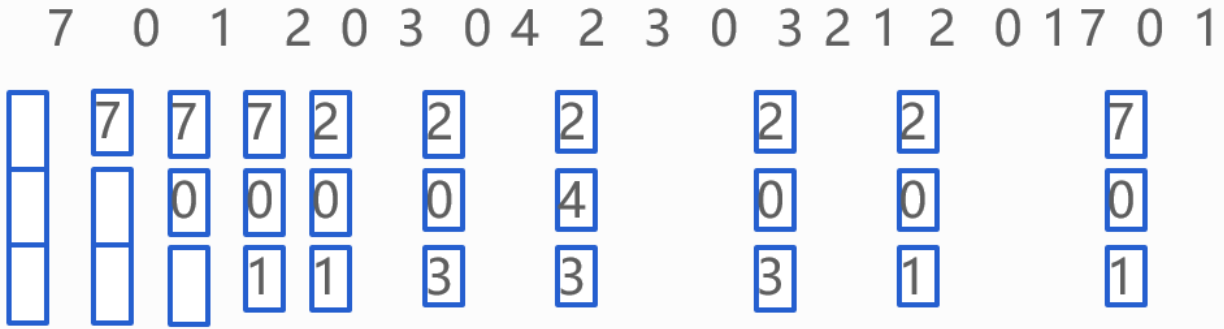


- Reference String： 1,2,3,4,1,2,5,1,2,3,4,5
- FIFO算法不仅导致页故障率偏高，且不够稳定
  - 该示例中，从3个页帧增加到4个页帧，页故障率反而增加

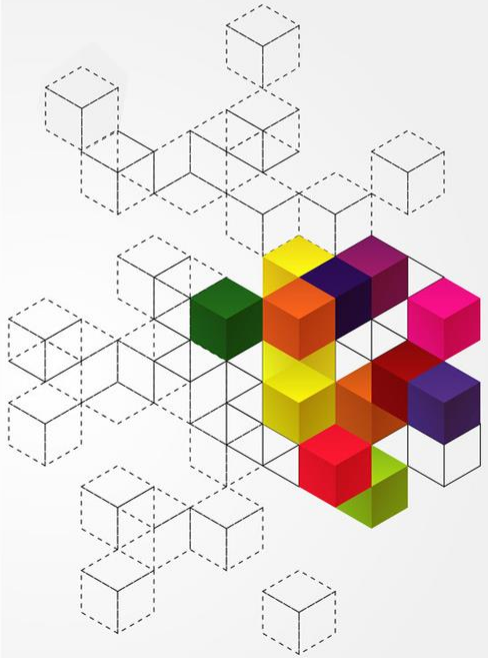


三、页置换算法2：OPT

- **展望未来，最久未被使用的页被选为牺牲页**

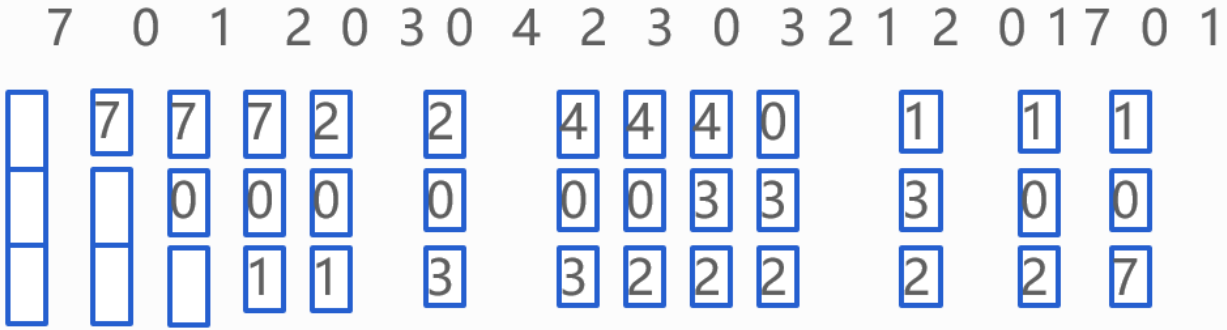


- 进程拥有3个页帧
- 发生9次页故障，其中包含6次页置换
- 问题：实际在进程运行时，未来不可精确预知

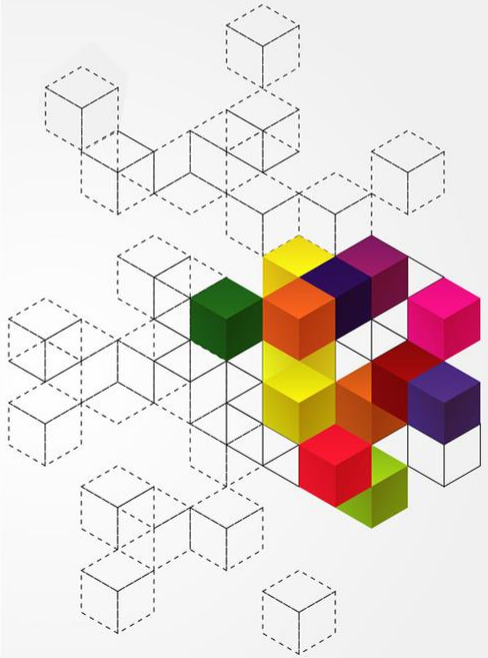


四、页置换算法3：LRU

- **以史为鉴，最近最久未被使用的页被选中置换**

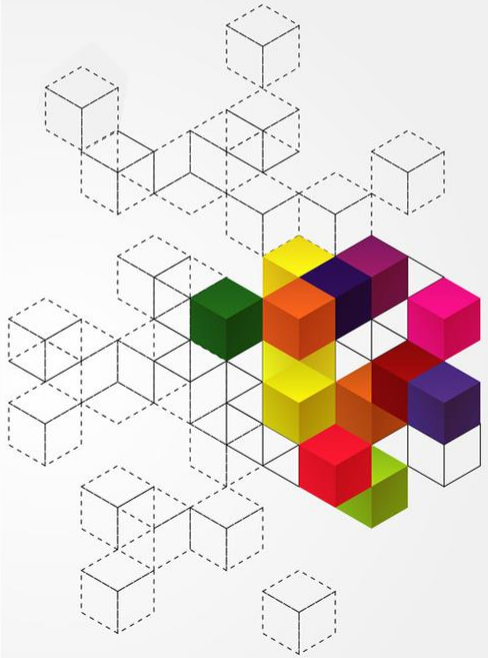


- 进程拥有3个页帧
- 发生12次页故障，其中包含9次页置换



本讲小结

- 多种页置换算法的介绍



内容纲要

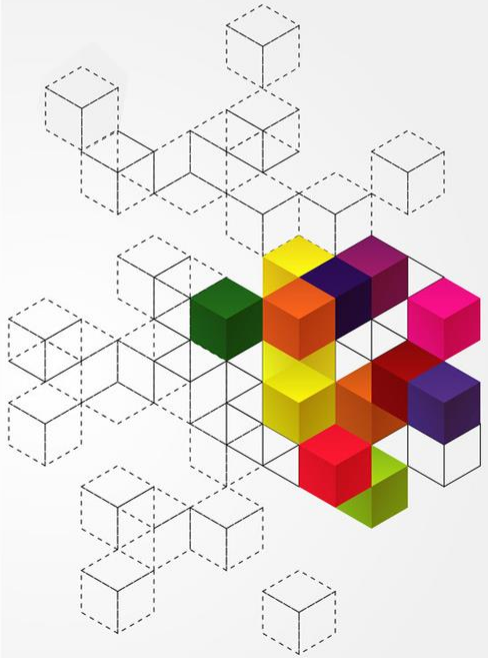
9.3 页置换算法实现

一、 页置换算法实现考虑

二、 近似实现1：附加引用位

三、 近似实现2：时钟算法

四、 近似实现3：增强型二次机会算法

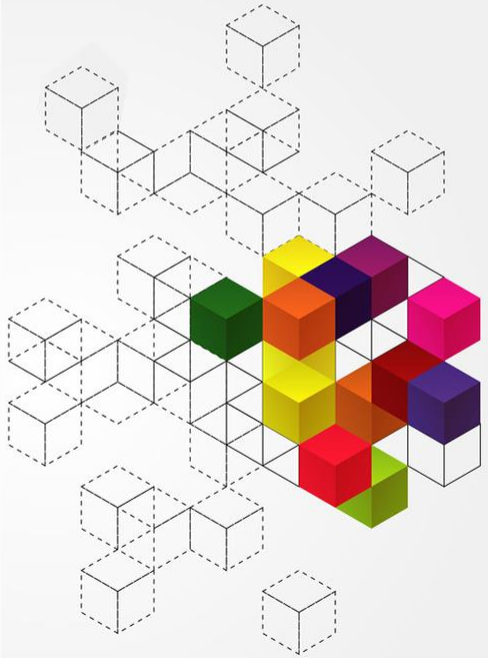


一、页置换算法的实现考虑

- **LRU算法是OPT算法的近似，不会产生Belady异常**
  - 是页置换算法的首选

问题： **使用栈来精确实现LRU算法，可能使得系统内存访问效率下降为原来的1/10**

不可接受！



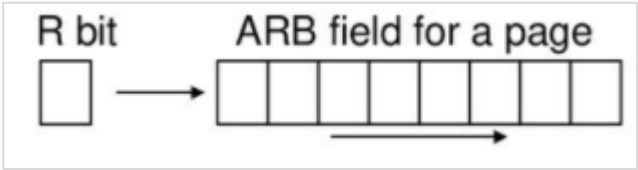
二、近似实现1：附加引用位

LRU精确实现，性能差，不实用

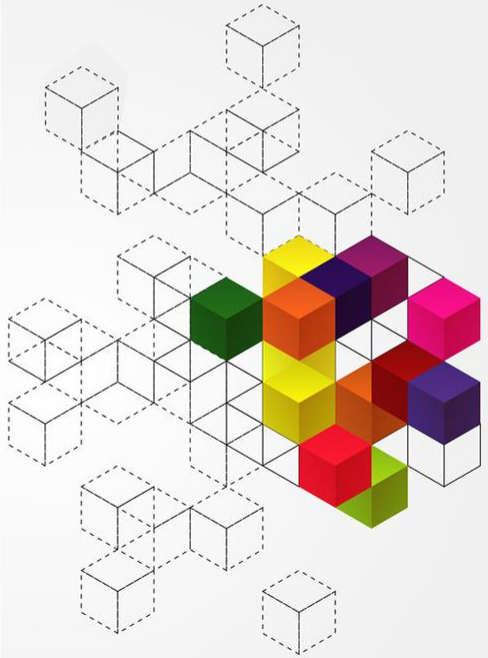
考虑对LRU进行近似实现

策略1：附加引用位

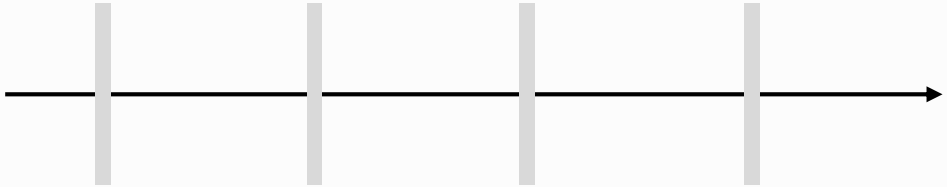
除了页面引用位R外，为每个页面增加额外的引用位
Additional Reference Bit (ARB)



At each timer interrupt, the R bit is shifted from left into the ARB, and the ARB shift to right accordingly

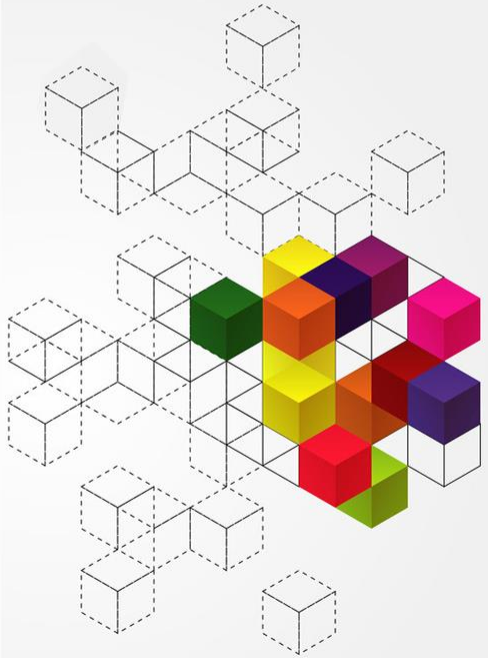


二、近似实现1：附加引用位



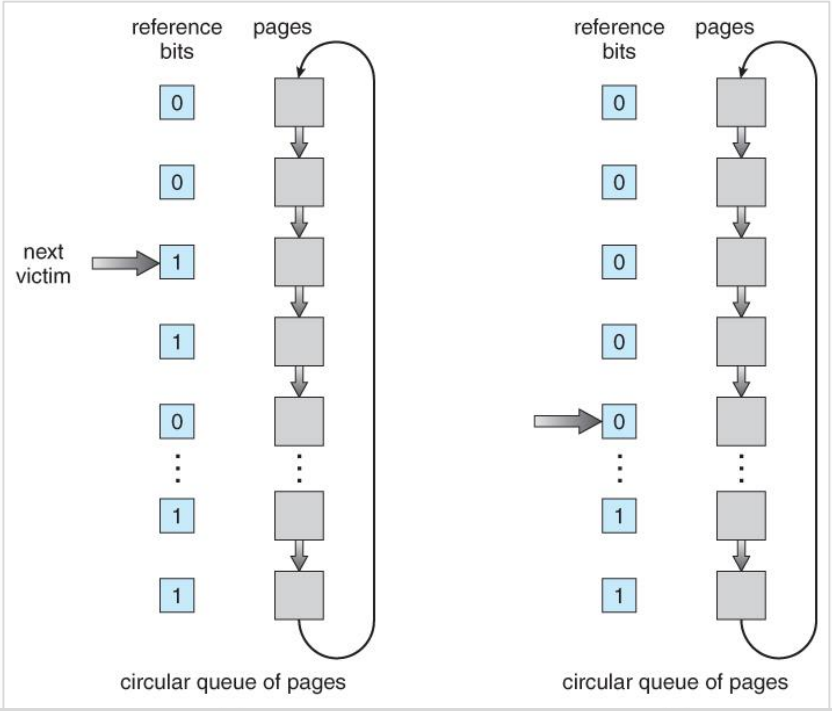
		tick1	tick2	tick3	tick4
R:		0	1	0	1
ARB:	0000000	0000000	1000000	0100000	1010000

When a page is replaced, select the page with least ARB



三、近似实现2：时钟算法

Second-Chance Page Replacement Algorithm

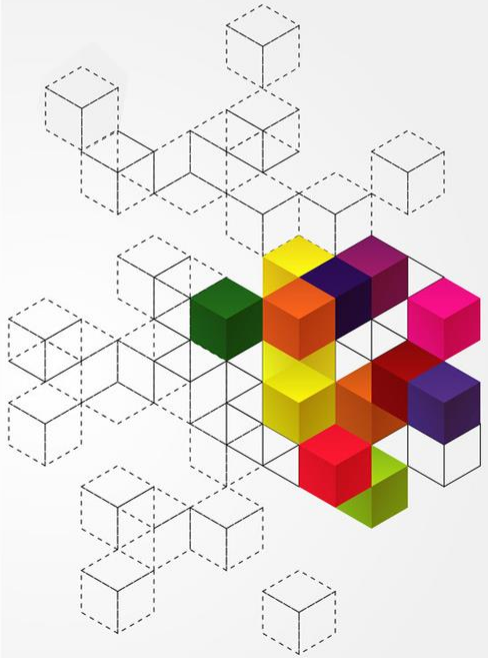


① 页面组织成环

② 每次页置换后，将next victim指针指向被置换的后继页面

③ 如需要进行置换，从当前next victim指向的位置开始，寻找第一个引用为为0的页，搜索途中遇到的引用位为1的改为0

引用位为1的页面，会赢得驻留内存的第2次机会

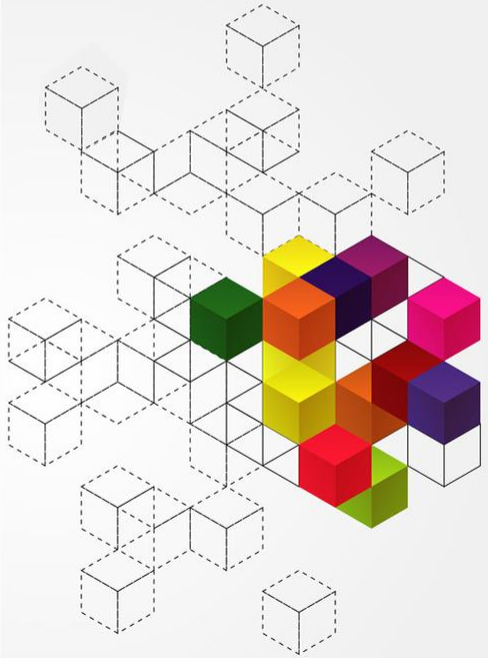


四、近似实现3：增强型二次机会算法

- **原理**：引入引用位(r)和修改位(c)作为有序对

(r,c)	页面描述
(0,0)	最近未使用，也未修改过（置换最佳候选）
(0,1)	最近未使用，但被修改过
(1,0)	最近使用过，但未被修改
(1,1)	最近使用过，也被修改过

置换时，优先选择rc位为00的页面置换，01，10，11次之



本讲小结

- 页置换算法近似实现方法讨论