局部置换算法没有考虑进程访存差异

有可能增加一个物理页面，缺页率会大幅下降！

FIFO 页面置换算法：假设初始顺序 a->b->c

物理页面数：3　　缺页次数：9

时间		0	1	2	3	4	5	6	7	8	9	10	11	12
访问页面			a	b	c	d	a	b	c	d	a	b	c	d
驻留集合	0	a	a	a	a	d	d	d	c	c	c	b	b	b
	1	b	b	b	b	b	a	a	a	d	d	d	c	c
	2	c	c	c	c	c	c	b	b	b	a	a	a	d
缺页状态						●	●	●	●	●	●	●	●	●
缺页状态						●								

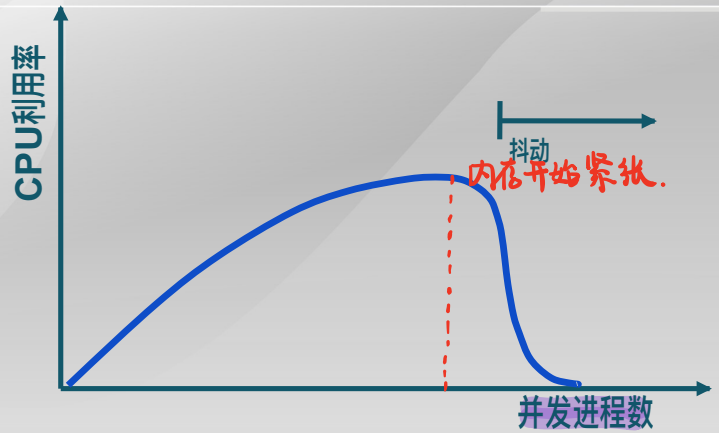
b->c->d->a　a->a->b->c　a->b->c->d　c->d->a->b　a->b->c->d

但如果给这个进程4个物理页面，缺页次数仅为1.

全局置换算法

- 思路
  - 全局置换算法为进程分配可变数目的物理页面
- 全局置换算法要解决的问题
  - 进程在不同阶段的内存需求是变化的⇒如何度量？
  - 分配给进程的内存也需要在不同阶段有所变化⇒按需分配
  - 全局置换算法需要确定分配给进程的物理页面数→不是一成不变的！

CPU利用率与并发进程数的关系



- CPU利用率与并发进程数存在相互促进和制约的关系
  - 进程数少时，提高并发进程数，可提高CPU利用率
  - 并发进程导致内存访问增加
  - 并发进程的内存访问会降低了访存的局部性特征，切换之后执行不相关的等⇒缺页率↑
  - 局部性特征的下降会导致缺页率上升和CPU利用率下降

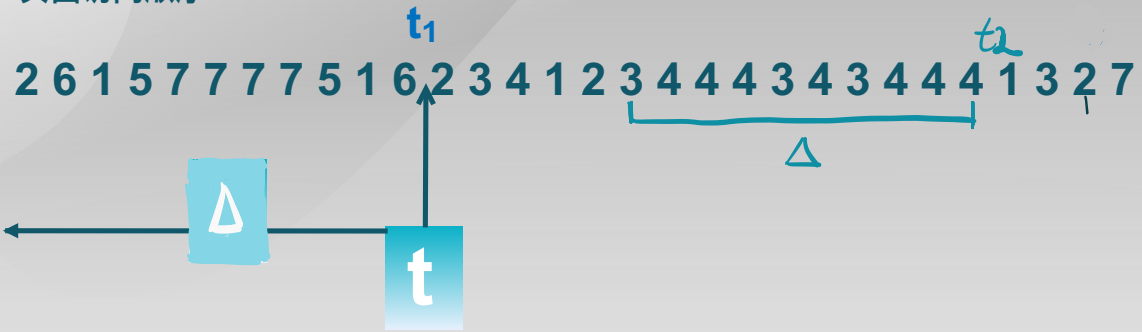
工作集

一个进程当前正在使用的逻辑页面集合，可表示为二元函数W(t, Δ)

- t是当前的执行时刻
- Δ 称为工作集窗口（working-set window），即一个定长的页面访问时间窗口
- W(t, Δ)是指在当前时刻 t 前的时间窗口中的所有访问页面所组成的集合
- | W(t, Δ) | 指工作集的大小，即页面数目

进程的工作集示例

页面访问顺序：



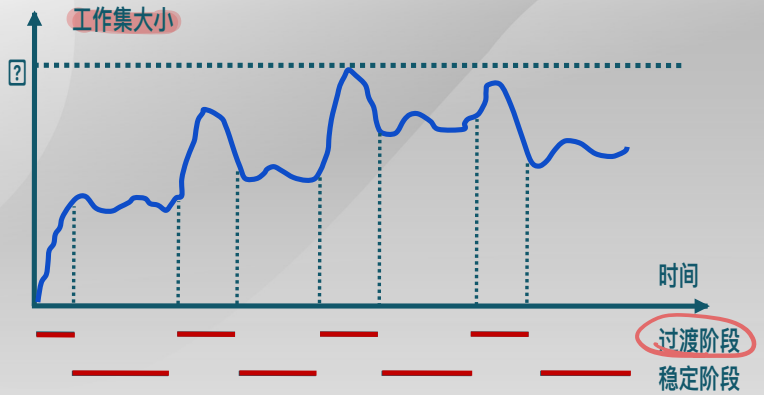
如果 Δ 时间窗口的长度为10，那么：

W(t, Δ) = {1, 2, 5, 6, 7}

W(t_1, Δ) = {1, 2, 5, 6, 7}

W(t_2, Δ) = {3, 4}

工作集的变化



- 进程开始执行后，随着访问新页面逐步建立较稳定的工作集
- 当内存访问的局部性区域的位置大致稳定时，工作集大小也大致稳定
- 局部性区域的位置改变时，工作集快速扩张和收缩过渡到下一个稳定值

常驻集

在当前时刻，进程实际驻留在内存当中的页面集合

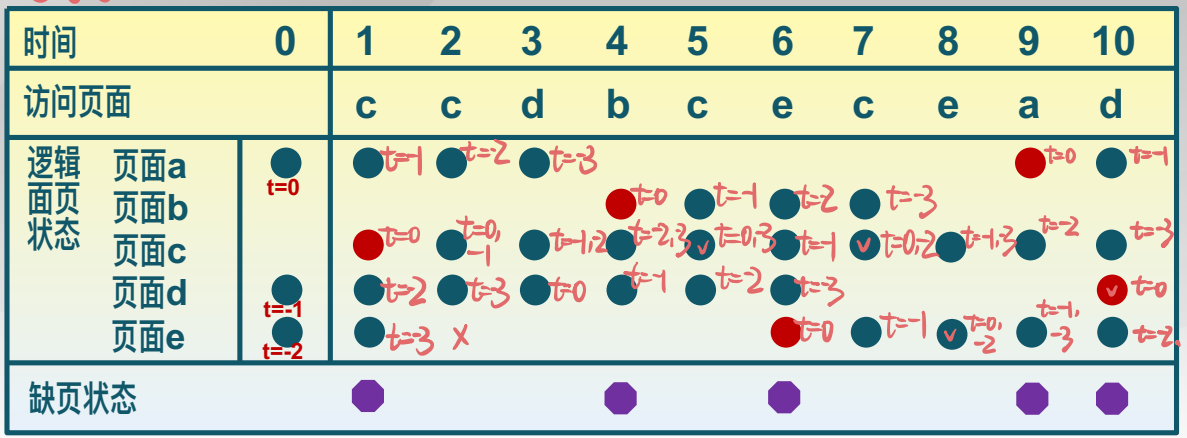
- 工作集与常驻集的关系
  - 工作集是进程在运行中固有的性质.
  - 常驻集取决于系统分配给进程的物理页面数目和页面置换算法.
- 缺页率与常驻集关系
  - 常驻集⊇工作集时，缺页率较少
  - 工作集发生剧烈变动（过渡）时，缺页较多
  - 进程常驻集达到一定大小时，缺页率也不会明显下降.

工作集置换算法

- 思路
  - 换出不在工作集中的页面 $[t-T, t]$
- 窗口大小T
  - 当前时刻前T个内存访问的页引用是工作集，T被称为窗口大小
- 实现方法
  - 访存链表：维护窗口内的访存页面链表
  - 访存时，换出不在工作集的页面；更新访存链表
  - 缺页时，换入页面；更新访存链表

工作集置换算法

τ = 4


时间
0
1
2
3
4
5
6
7
8
9
10
访问页面
c
c
d
b
c
e
c
e
a
d
逻辑页面状态
页面a
页面b
页面c
页面d
页面e
缺页状态

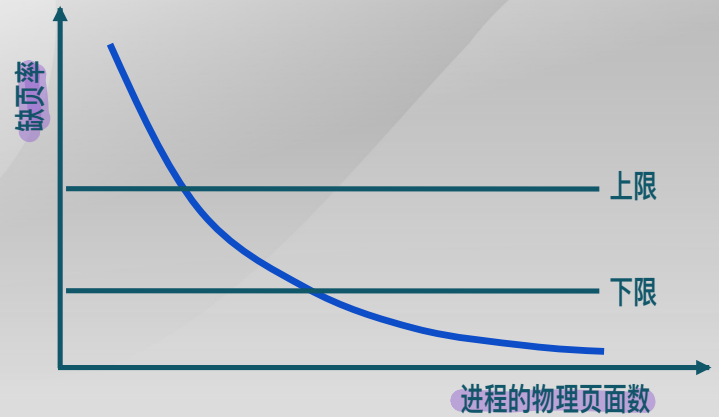
缺页率(page fault rate)

缺页次数 / 内存访问次数　　或　缺页平均时间间隔的倒数

更常用，两次缺页的间隔。

- 影响缺页率的因素
  - 页面置换算法
  - 分配给进程的物理页面数目
  - 页面大小
  - 程序的编写方法（局部性）

缺页率置换算法(PFF, Page-Fault-Frequency)



通过调节常驻集大小，使每个进程的缺页率保持在一个合理的范围内

- 若进程缺页率过高，则增加常驻集以分配更多的物理页面
- 若进程缺页率过低，则减少常驻集以减少它的物理页面数

并发度↓，CPU利用率↓

缺页率置换算法的实现

- 访存时，设置引用位标志
- 缺页时，计算从上次缺页时间t_{last}到现在$t_{current}$的时间间隔
  - 如果 $t_{current} - t_{last} > T$，则置换所有在[$t_{last}$，$t_{current}$]时间内没有被引用的页
  - 如果$t_{current} - t_{last} \leq T$，则增加缺失页到工作集中

缺页率置换算法示例

- 假定窗口大小为 2 T=2

时间		0	1	2	3	4	5	6	7	8	9	10
访问页面			c	c	d	b	c	e	c	e	a	d
逻辑页面状态	页面a	●	●	●	●	×					●	●
	页面b					●	●	●	●	●	×	
	页面c		●	✓	●	●	✓	●	✓	●	●	●
	页面d	●	●	●	✓	●	●	●	●	●	×	●
	页面e	●	●	●	●	×		●	●	✓	●	●
缺页状态			●			●		●			●	●
$t_{cur} - t_{last}$			1			3>2		2=2			3>2	1

剔除没有访问的页

加入缺失的页

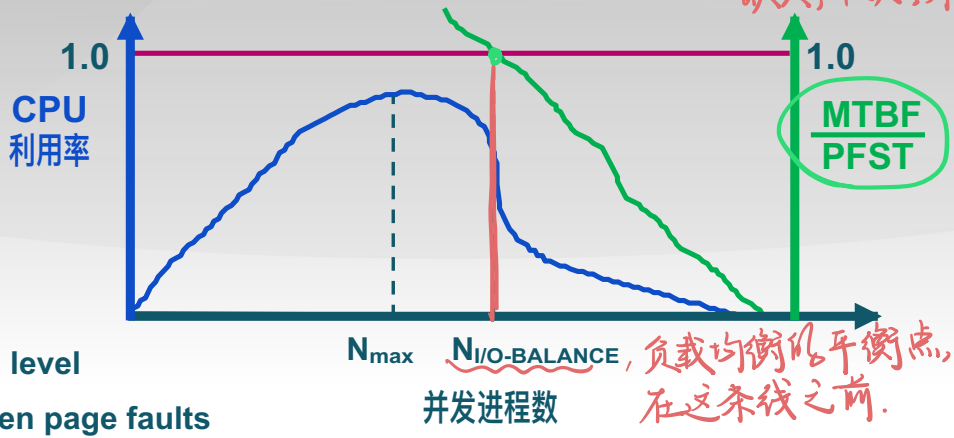
抖动问题(thrashing)

- 抖动
  - 进程物理页面太少，不能包含工作集
  - 造成大量缺页，频繁置换
  - 进程运行速度变慢
- 产生抖动的原因
  - 随着驻留内存的进程数目增加，分配给每个进程的物理页面数不断减小，缺页率不断上升
- 操作系统需在并发水平和缺页率之间达到一个平衡
  - 选择一个适当的进程数目和进程需要的物理页面数

负载控制

通过调节并发进程数（MPL）来进行系统负载控制

- ∑WSi = 内存的大小
- 平均缺页间隔时间(MTBF) = 缺页异常处理时间(PFST)



MPL-multiprogramming level

MTBF-mean time between page faults

PFST-page fault service time