第43 - 44讲 I/O缓冲技术

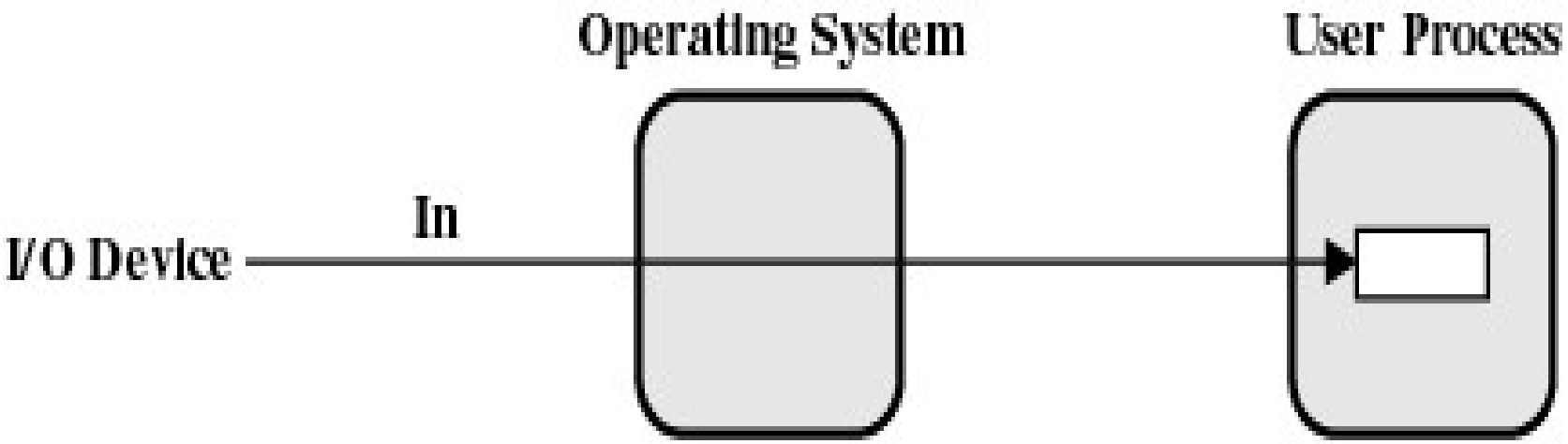
§4.4 I/O Buffering

引　　例

- *假定*：一个用户进程将从磁带上读入若干块数据（每块 100 字节）；数据块装入用户进程地址空间中的一个数据区，其虚拟地址为 1000 ～ 1099　（ 100 字节）

  请问：如果从磁带上读进一块数据并直接把它送入用户进程的工作区，会有什么问题？

No Buffering



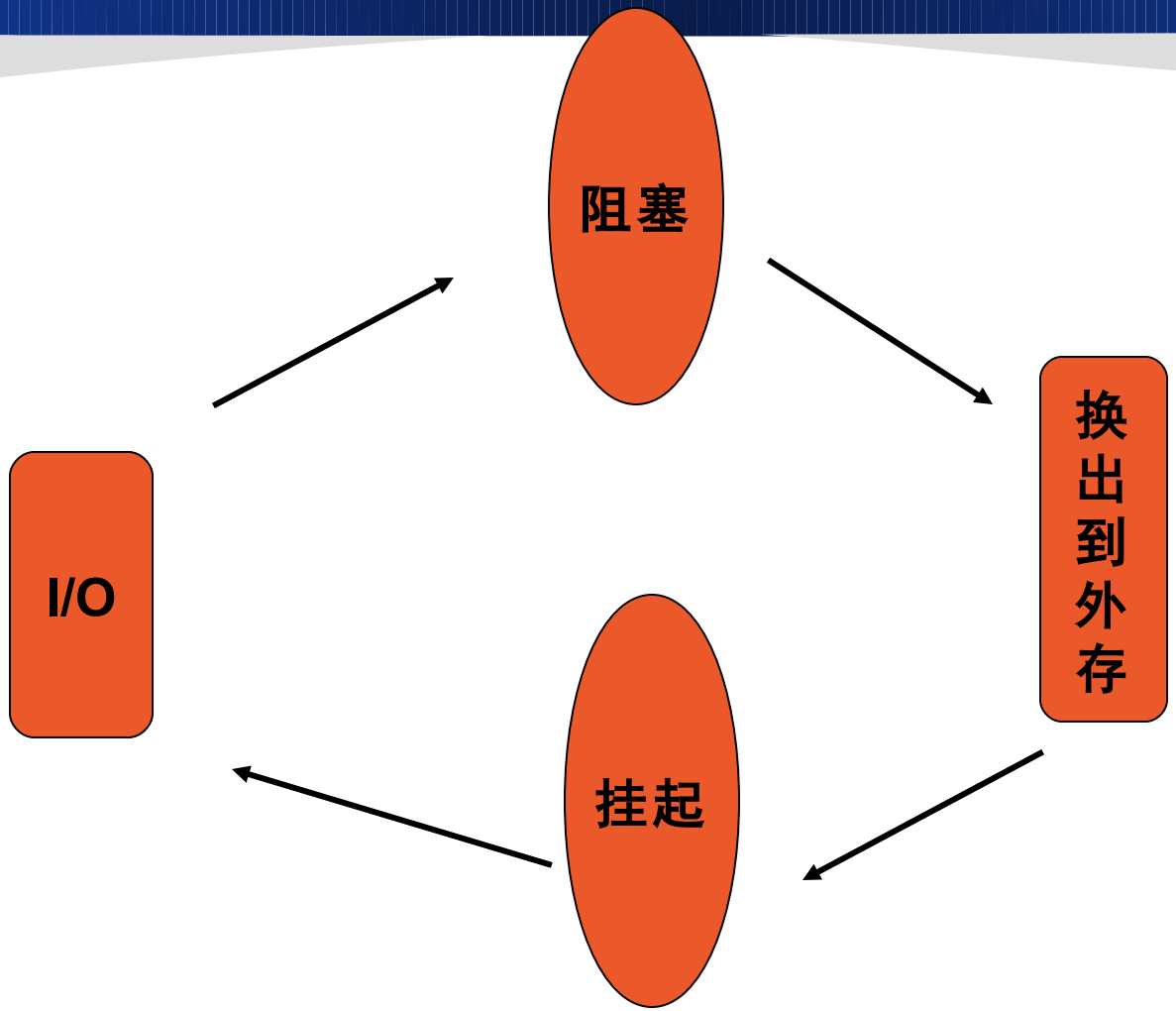
(a) No buffering

Problems

- Processes must wait for I/O to complete before proceeding。显然，用户进程的运行速度将受制于磁带机的工作速度

- 这种 I/O 方法会影响 OS 的交换策略。若 OS 决定挂起该用户进程，其虚地址 1000~1099 必须驻留在内存，否则，数据可能丢失，故进程不能被完整地交换出内存

- 可能会导致*单进程死锁*


阻塞
换出到外存
挂起
I/O

提高用户进程运行效率的方法

许多 OS 通过引入 *I/O Buffering* 来提高用户进程的运行效率、缩短其周转时间。

I/O Buffering

- 核心思想：在内存中建立 I/O 缓冲区
- 缓存从输入设备流入内存的数据
- 缓存从内存流向输出设备的数据

I/O Buffering

- **Block-oriented**
  - Information is stored in fixed sized blocks
  - Transfers are made a block at a time
  - Used for disks and tapes
- **Stream-oriented**
  - Transfer information as a stream of bytes
  - byte：每个 buffer 缓存一个字符 (字节)，键盘输入等
  - line：每个 buffer 缓存一个字符串或长度可变的多个字节，用于显示输出、打印输出等

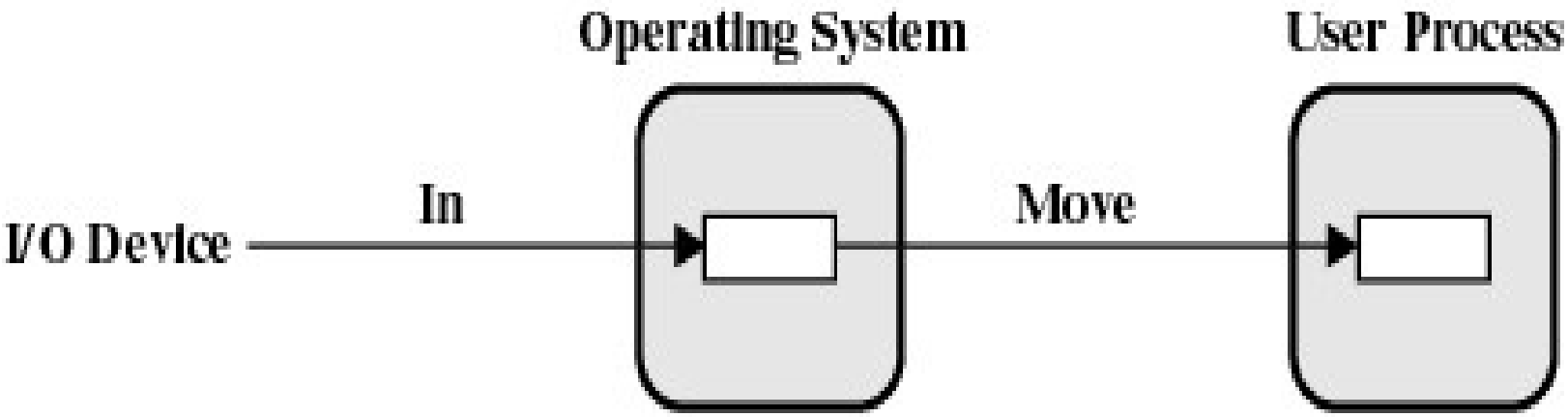
Read Ahead and Write Postponing

- **Read Ahead(提前读)**
  - 用户进程从 I/O 缓冲区取走前一个数据后立即发出对下一个数据的输入请求； OS 将在适当的时候响应该请求，把需要的数据读入 I/O 缓冲区。
- **Write Postponing(延后写）**
  - 当用户进程请求输出数据时， OS 将很快把用户进程请求输出的数据从用户进程的工作区取走，将其暂存在 I/O 缓冲区中； 直到用户进程指定的输出设备空闲时， OS 才把暂存在 I/O 缓冲区中的数据写入用户进程指定的输出设备上。

I/O Buffering

- Single Buffer(单缓冲区)
- Double Buffer(双缓冲区)
- Circular Buffer(循环缓冲区)

Single Buffer



(b) Single buffering

Figure 11.6 I/O Buffering Schemes (input)

Single Buffer

- Operating system assigns a buffer in main memory for an I/O request.

- Block-oriented
  - Input transfers made to buffer
  - Block moved to user space when needed
  - Another block is moved into the buffer
    - Read ahead

Single Buffer

- **Block-oriented**
  - User process can process one block of data while next block is read in.
  - Swapping can occur since input is taking place in system memory, not user memory.
  - Operating system keeps track of assignment of system buffers to user processes.

Single Buffer

- **Stream-oriented**
  - Used a line at time.
  - User input from a terminal is one line at a time with carriage return signaling the end of the line.
  - Output to the terminal is one line at a time.

若一块数据从外部设备输入到内存所花费的时间为 T，在内存中移动所花费的时间为 M，被用户进程加工处理所花费的时间为 C，则

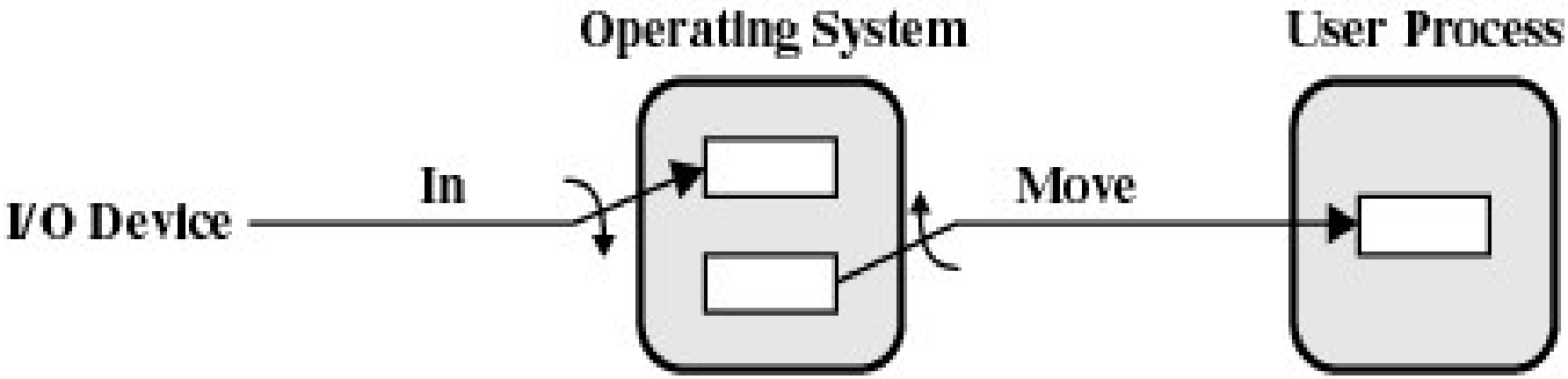
- 在没有使用 I/O 缓冲区的情况下，平均每块数据的处理时间近似为： T + C
- 在使用单 I/O 缓冲区的情况下，平均每块数据的处理时间近似为： max(T,C) + M

- 可见，相对于没有使用 I/O 缓冲区的情形， 引入单 I/O 缓冲区后，用户进程的运行效率得到了提高。
- 然而，如果用户进程在对有关数据进行加工处理时并不释放 I/O 缓冲区， 那么用户进程的性能并不能得到改善。
- 此外，如果 T 远远大于 C， 即外部设备的 I/O 速度比用户进程的计算速度慢得多， 那么即使引入单 I/O 缓冲区，用户进程的性能也几乎没有得到改善。

Double Buffer

- Use two system buffers instead of one.
- A process can transfer data to or from one buffer while the operating system empties or fills the other buffer.

Double Buffer



(c) Double buffering

Double Buffer

- 外部设备和应用进程常常交替引用 Double Buffer

- Double Buffer 也称为 *Buffer Swapping(缓冲交换)*

Single Buffer *vs.* Double Buffer

- 使用双 I/O 缓冲区，即使用户进程在对有关数据进行加工处理时不释放相关的 I/O 缓冲区， 用户进程的性能也能得到改善。

- 与单 I/O 缓冲区类似，如果 T 远远大于 C，即外部设备的 I/O 速度比用户进程的计算速度慢得多，那么即使引入双 I/O 缓冲区，用户进程的性能也几乎没有得到改善。

- 缓和外部设备的 I/O 速度与用户进程的计算速度不匹配的一种有效的办法是，在外部设备和用户进程之间设立多个 I/O 缓冲区

Circular Buffer

- More than two buffers are used.
- Each individual buffer is one unit in a circular buffer.
- Used when I/O operation must keep up with process.

Circular Buffer

(d) Circular buffering

Figure 11.6 I/O Buffering Schemes (input)

Disk Cache

- Buffer in main memory for disk sectors, 不是一种硬件设施
- Contains a copy of some of the sectors on the disk.
- 其组织形式基于程序引用的局部性原理。

工作原理

- 当用户进程请求从磁盘读入一个扇区时，系统首先在 disk cache 中寻找该扇区的副本
  - 如果能够找到，那么系统将从 disk cache 中取出该扇区的副本并返给用户进程；
  - 否则，系统首先从磁盘上读入该扇区并在 disk cache 中为其建立一个副本，然后将该副本返给用户进程。

工作原理 （续）

- 当用户进程请求向磁盘上写出一个扇区时，系统同样首先在 disk cache 中寻找该扇区的副本
  - 如果能够找到，那么系统将根据用户进程的请求修改该扇区的副本；
  - 否则，系统同样首先从磁盘上读入该扇区并在 disk cache 中为其建立一个副本，然后根据用户进程的请求修改该副本。

磁盘高速缓存的数据安全性

Disk Cache 中的数据写出到磁盘：

1. 在系统空闲或需要淘汰被写的缓存空间时进行写。
2. 周期性地进行写。
3. 立即写回，称为“写穿透高速缓存”(write through)，相当于只有读缓存而没有写缓存。

设计问题

- 当用户进程请求从磁盘上读入一个扇区时， 如果系统能够在 disk cache 中找到该扇区的副本，那么系统如何把该副本提交给用户进程？

- 当系统需要从磁盘上读入一个扇区并在 disk cache 中为其建立一个副本时，如果 disk cache 没有空闲空间，那么系统使用何种策略从 disk cache 中选择一个被置换扇区？

向用户进程提交扇区副本的方法

- 如果系统不允许用户进程访问 disk cache，那么系统将把用户进程需要的扇区从 disk cache 中复制到用户进程的工作区。

- 如果系统允许用户进程访问 disk cache ，那么系统将把用户进程需要的扇区副本在 disk cache 中的位置指针传递给用户进程。

扇区置换算法

- **Least Recently Used （ LRU 置换算法）**
  - **The block that has been in the cache the longest with no reference to it is replaced**
- **Least Frequently Used （ LFU 置换算法）**
  - **The block that has experienced the fewest references is replaced**

Least Recently Used

- The cache consists of a stack of blocks.
- Most recently referenced block is on the top of the stack.
- When a block is referenced or brought into the cache, it is placed on the top of the stack.

Least Recently Used

- The block on the bottom of the stack is removed when a new block is brought in.
- Blocks don't actually move around in main memory.
- A stack of pointers is used.

Least Frequently Used

- A counter is associated with each block.
- Counter is incremented each time block accessed.
- Block with smallest count is selected for replacement.
- Some blocks may be referenced many times in a short period of time and then not needed any more.

Frequency-Based Replacement Algorithm

- 系统把 disk cache 中的所有扇区组成一个栈：位于栈顶的扇区最近才被访问过；而位于栈底的扇区最久没有被访问过。

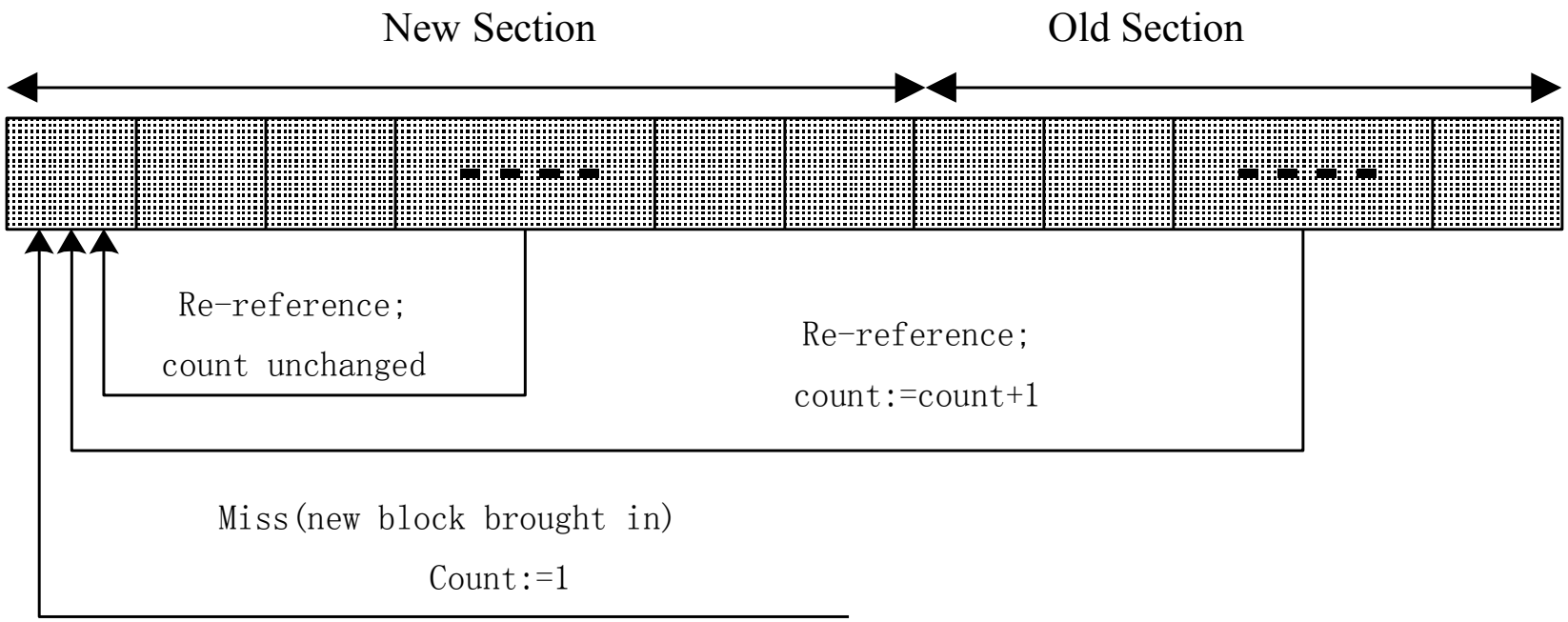
- 系统从中间某个位置把 disk cache 中的栈分成两个部分：靠近栈顶的部分称为 New Section，靠近栈底的部分称为 Old Section 。

- 系统为 disk cache 中的每个扇区设置一个引用计数器。

Frequency-Based Replacement Algorithm

- 当系统从磁盘上把一个新扇区读入 disk cache 中时，把该扇区放在 disk cache 的栈顶，并把该扇区的引用计数器置 1。

- 当 disk cache 中的某个扇区被用户进程访问时，系统将把该扇区从其原来的位置移到 disk cache 的栈顶，同时判断该扇区原来是否位于栈的 New Section：若是，该扇区的引用计数器的值保持不变；否则，该扇区的引用计数器的值被加 1。



(a) FIFO

P480 fig11.10

Frequency-Based Replacement Algorithm

- 当需要从 disk cache 中选择一个被置换扇区时，系统将从位于栈的 Old Section 中的所有扇区中选择引用计数器值最小的那个扇区。

- 如果位于栈的 Old Section 中的所有扇区其引用计数器的值均相等，那么系统将选择位于 disk cache 栈底的那个扇区。

基于频率置换算法的不足

- 一个新的扇区被读入 disk cache 时，它将被置于栈的 New Section 中，其引用计数器被置 1。
- 只要该扇区不离开 New Section，它的引用计数器的值就一直保持为 1。
- 当该扇区最终离开 New Section 时，它的引用计数器的值仍然为 1。

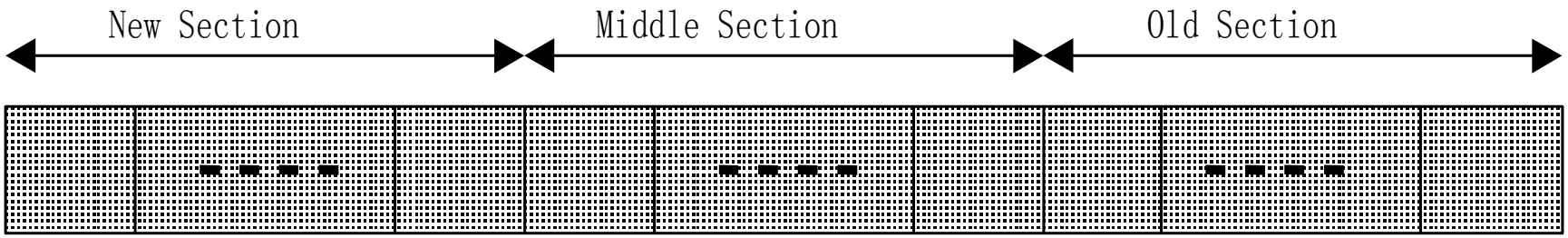
基于频率置换算法的不足 （续）

- 如果此时该扇区不能在系统进行扇区置换之前很快地被用户进程引用，那么该扇区很可能被置换出去。
- 这意味着，一个在离开 New Section 时不能很快被引用的扇区（即使是一个经常被引用的扇区）根本没有机会增加自己的引用计数。
- 显然，这对一个经常被引用但在离开 New Section 时不能很快被引用的扇区是不合理的。

基于频率置换算法的改进

把 disk cache 中的栈分成三个部分

- New Section
- Middle Section
- Old Section

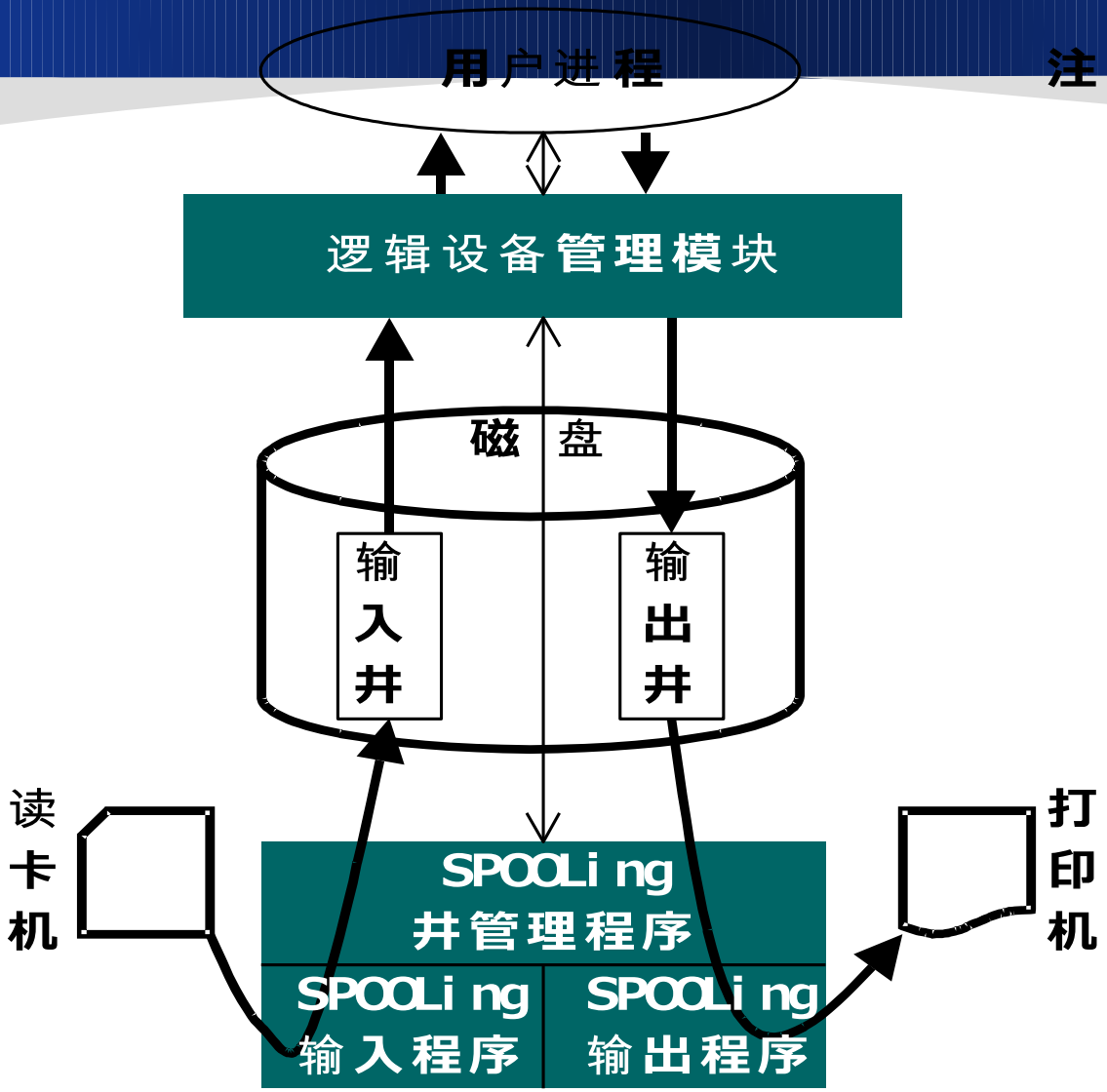


(b) Use of Three Sections

P480 fig11.10

SPOOLing 技术

核心思想是：在快速辅助存储设备中建立 I/O 缓冲区，用于缓存从慢速输入设备流入内存的数据，或缓存从内存流向慢速输出设备的数据。



SPOOLing技术实现原理示意图