第11章 I/O管理和磁盘调度

- 主要内容
  - 11.1 I/O设备 ★★★★★
  - 11.2 I/O功能的组织 ★★★★★
  - 11.3 操作系统设计问题 ★★★★★
  - 11.4 I/O缓冲 ★★★★★
  - 11.5 磁盘调度 ★★★★★
  - 11.6 RAID ★★★★★
  - 11.7 磁盘高速缓存 ★★★★★
  - 11.8 UNIX SVR4操作系统的I/O （自学）
  - 11.9 Linux操作系统的I/O （自学）
  - 11.10 Windows操作系统的I/O （自学）

11.1 I/O设备 ★★★★★

- I/O设备类别
  - 人可读
  - 机器可读
  - 通信
- I/O设备差异
  - 数据速率
  - 应用
  - 控制的复杂性
  - 传送单位
  - 数据表示
  - 错误条件

11.2 I/O功能的组织 ★★★★★

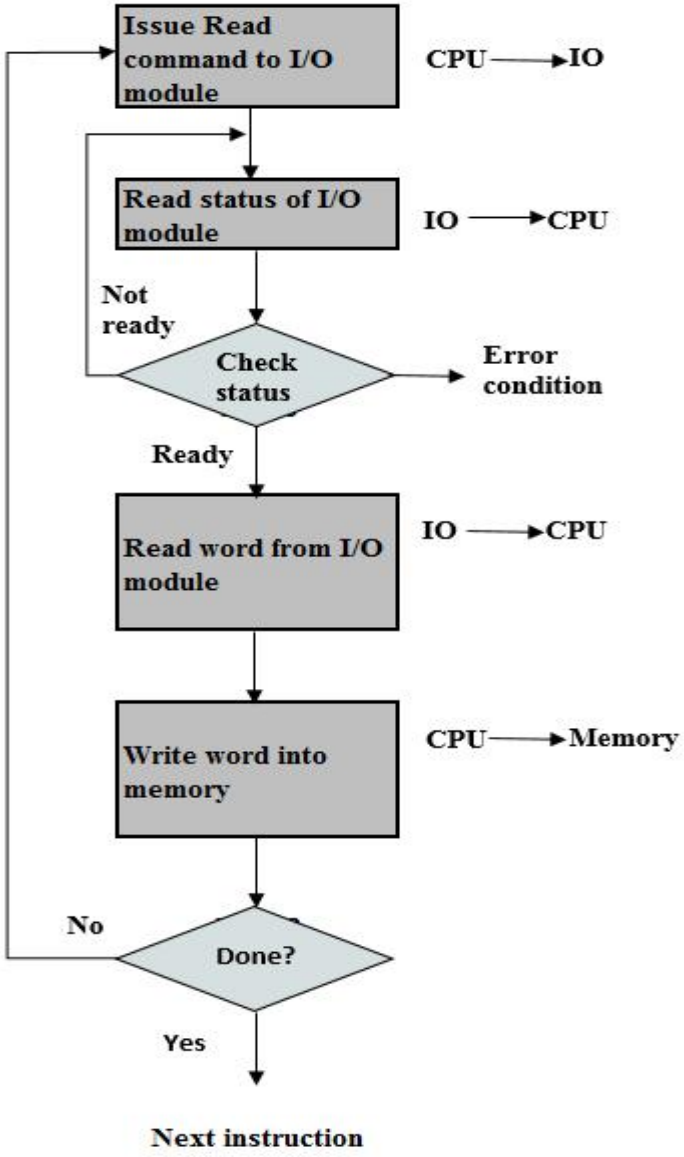
执行I/O的三种技术

- 程序控制I/O
- 中断驱动I/O
- 直接存储器访问（DMA）

11.2 I/O功能的组织 ★★★★★

执行I/O的三种技术

- 程序控制I/O

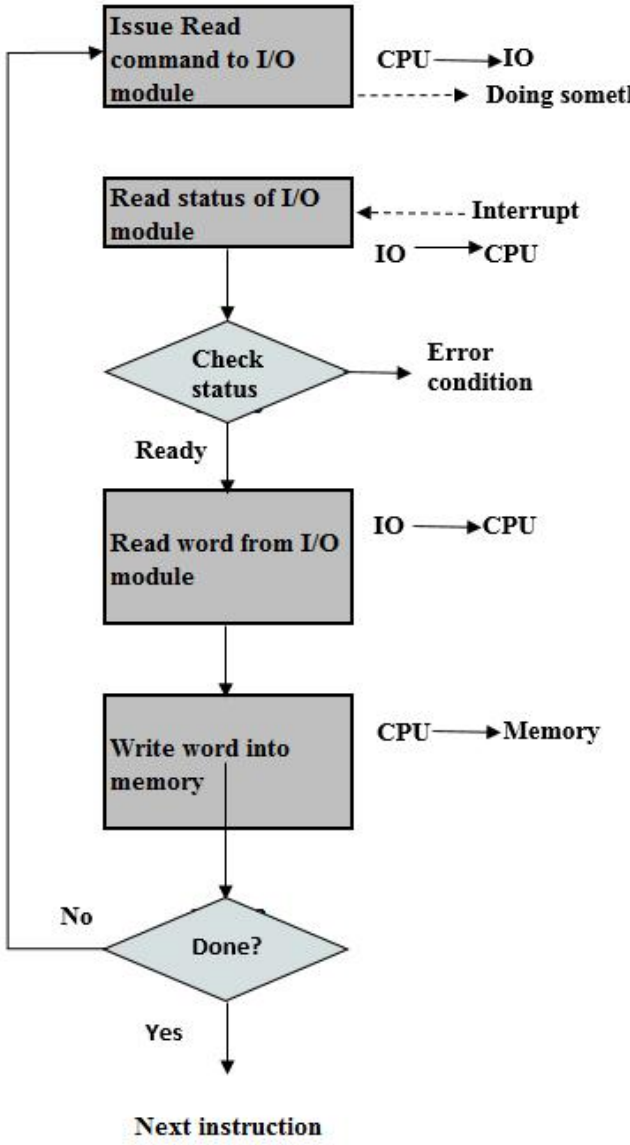


11.2 I/O功能的组织 ★★★★★

执行I/O的三种技术

- 中断驱动I/O

非阻塞IO？

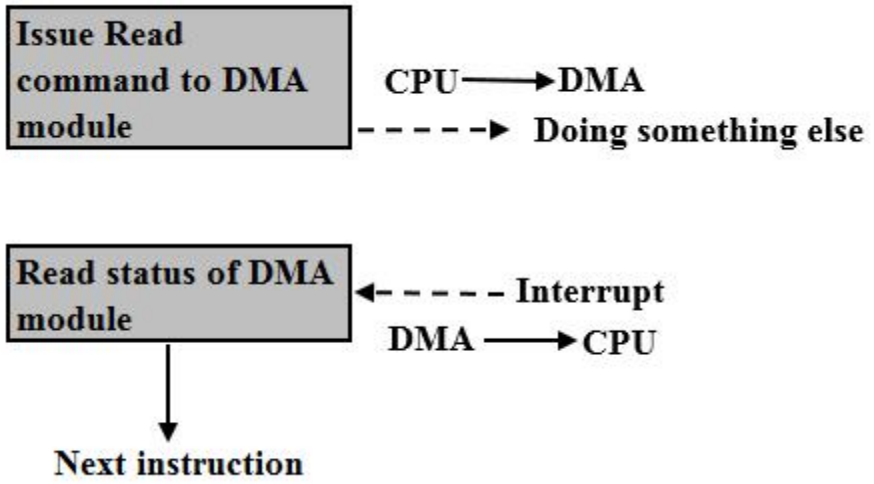


原进程被阻塞

11.2 I/O功能的组织 ★★★★★

执行I/O的三种技术

- 直接存储器访问（DMA）

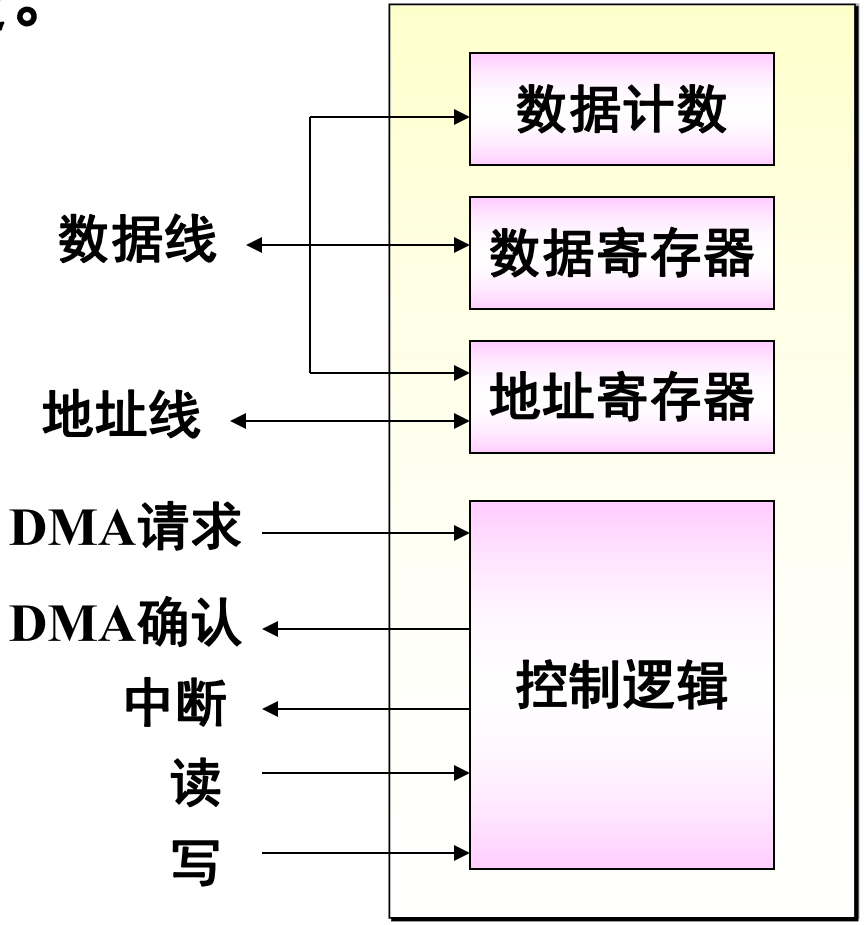


11.2.1 I/O功能的发展 ★★★★★

1. 处理器直接控制外围设备。
2. 增加了处理器或I/O模块，处理器使用非中断的程序控制I/O。【程序控制IO】
3. 配置同2，但采用了中断方式。【中断驱动IO】
4. I/O模块通过DMA直接控制存储器。【DMA】
5. I/O模块具有一个单独的处理器，有专门为I/O设计的指令集。【IO通道】
6. I/O模块有自己的局部存储器，其本身就是一台计算机。【IO处理器】

11.2.2 直接存储器访问 ★★★★★

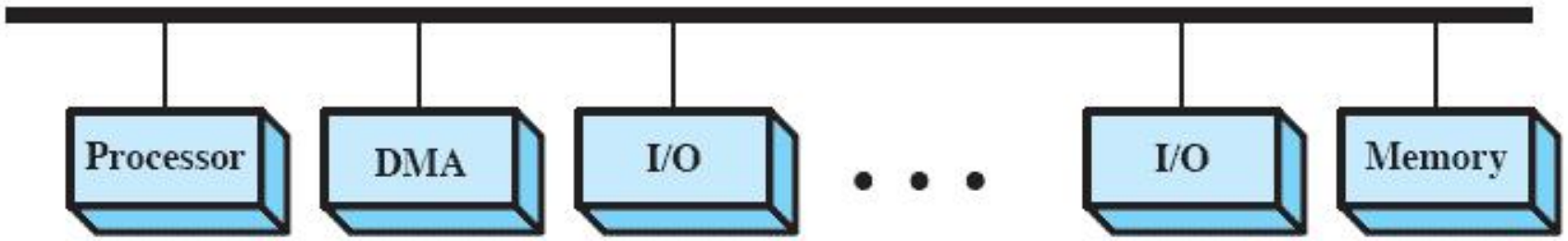
- DMA单元能够模拟处理器，像处理器一样获得系统总线的控制权，利用系统总线与存储器进行双向数据传送。



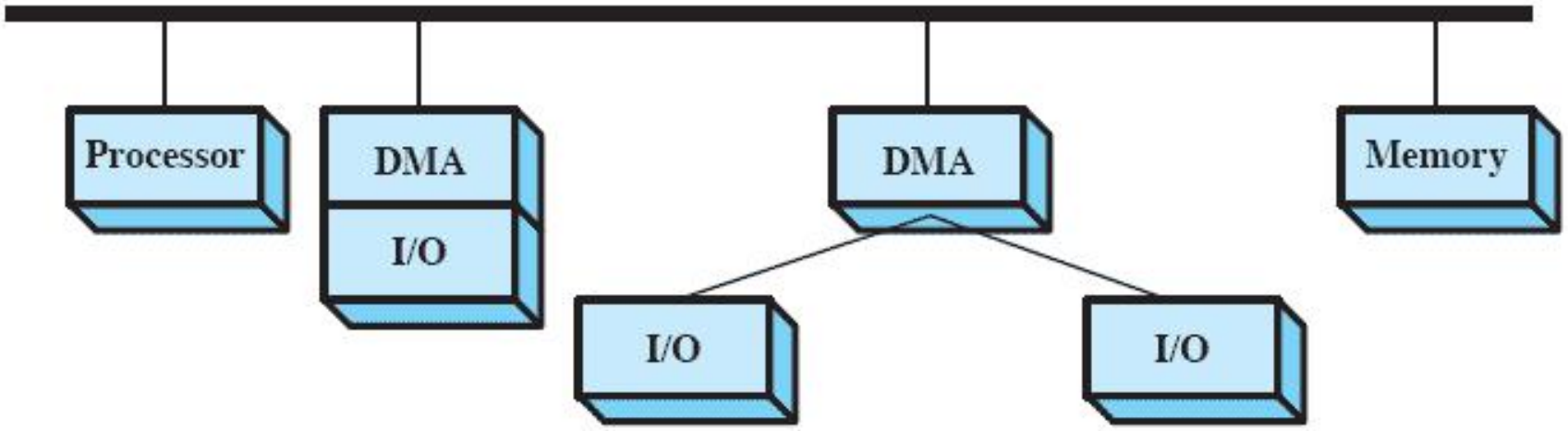
DMA技术工作流程

- 当处理器想读或写一块数据时，通过向DMA模块发送以下信息来给DMA模块发出一条命令：
  - 请求读或写操作的信号，通过读写控制线发送。
  - 相关的I/O设备地址，通过数据线发送。
  - 从存储器中读或往存储器中写的起始地址，在数据线上传送，并由DMA模块保存在其地址寄存器中。
  - 读或写的字数，通过数据线传送，并由DMA模块保存在其数据计数寄存器中。
- 处理器继续执行其工作；
- DMA模块直接从存储器中或往存储器中传送整块数据，一次传送一个字；
- 传送结束后，DMA模块给处理器发送一个中断信号。

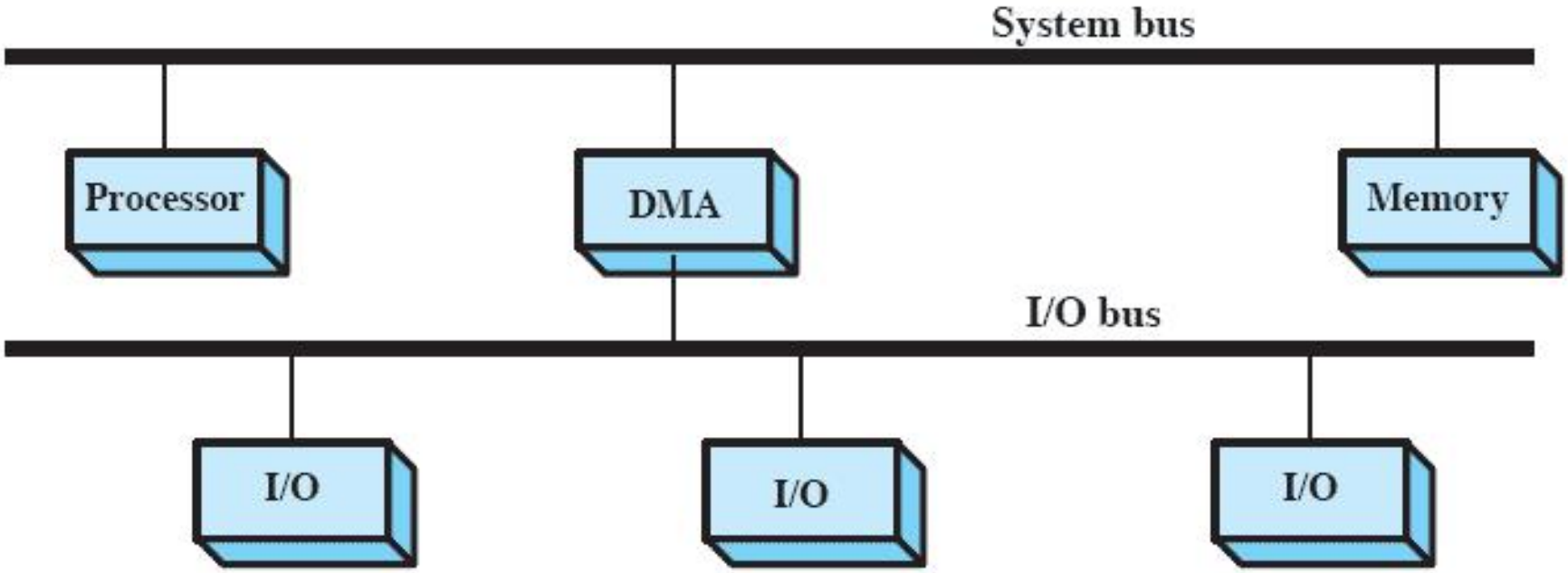
DMA机制的配置方法



(a) Single-bus, detached DMA



(b) Single-bus, Integrated DMA-I/O



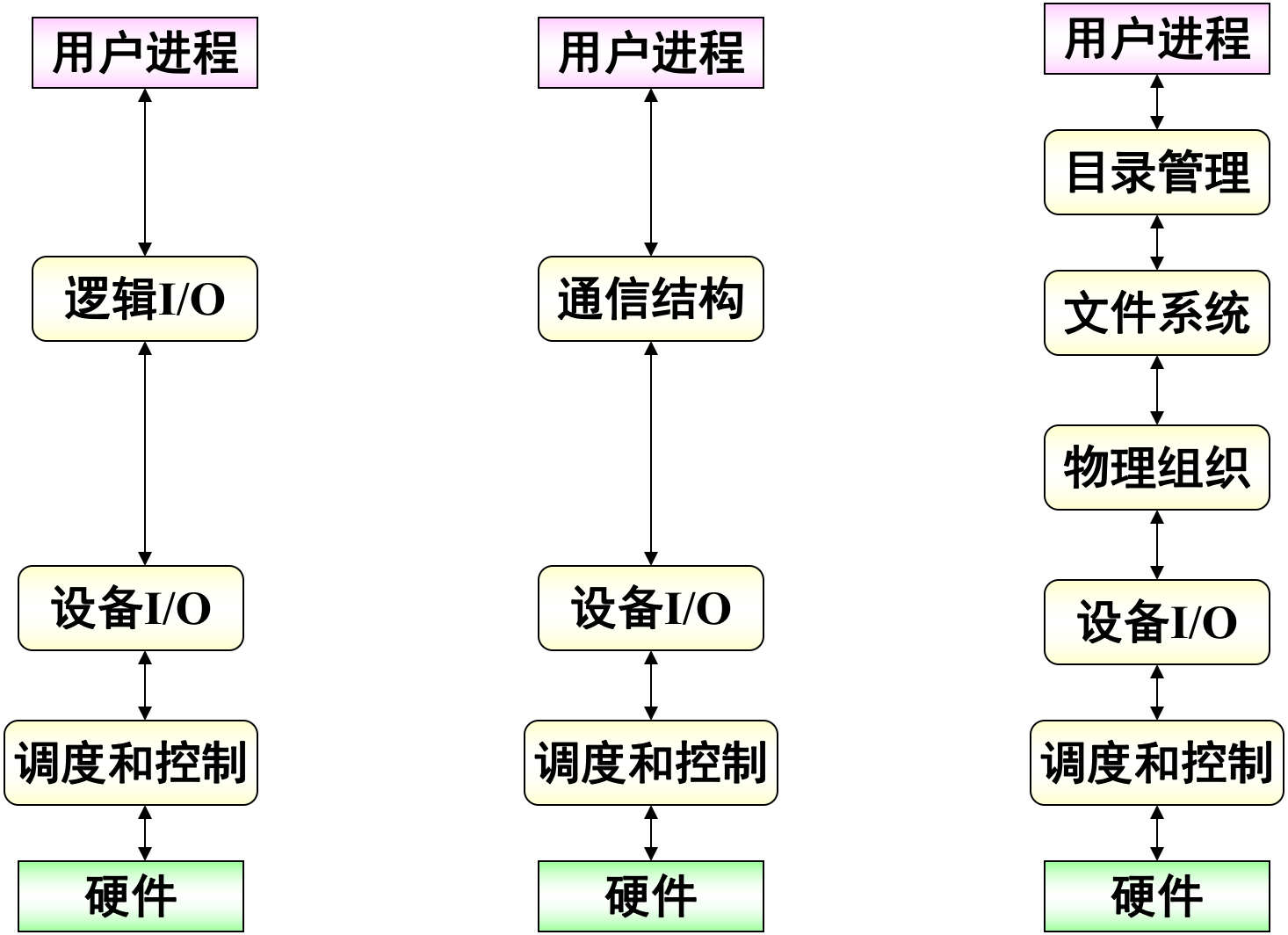
(c) I/O bus

11.3 操作系统设计问题★★★★★

11.3.1 设计目标

- 效率
  - 多道程序设计
  - 交换技术
- 通用性
  - 处理器看待I/O设备的方式
  - 操作系统管理I/O设备和I/O操作的方式

11.3.2 I/O功能的逻辑结构 ★★★★★

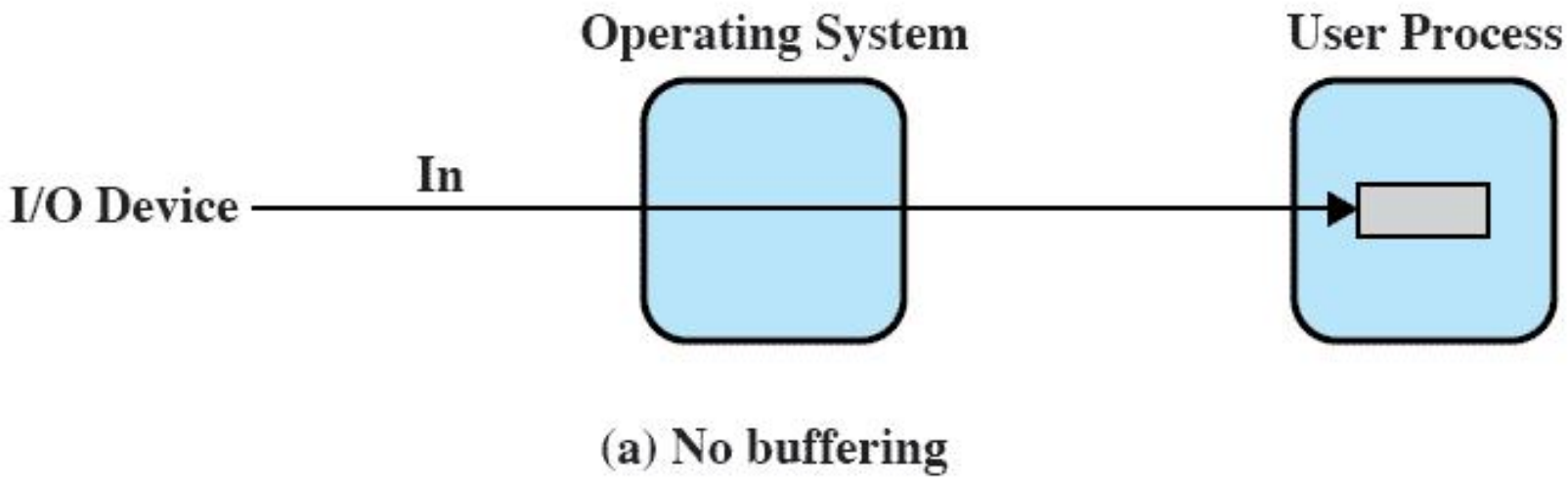


(a)逻辑外部设备　(b)通信端口　(c)文件系统

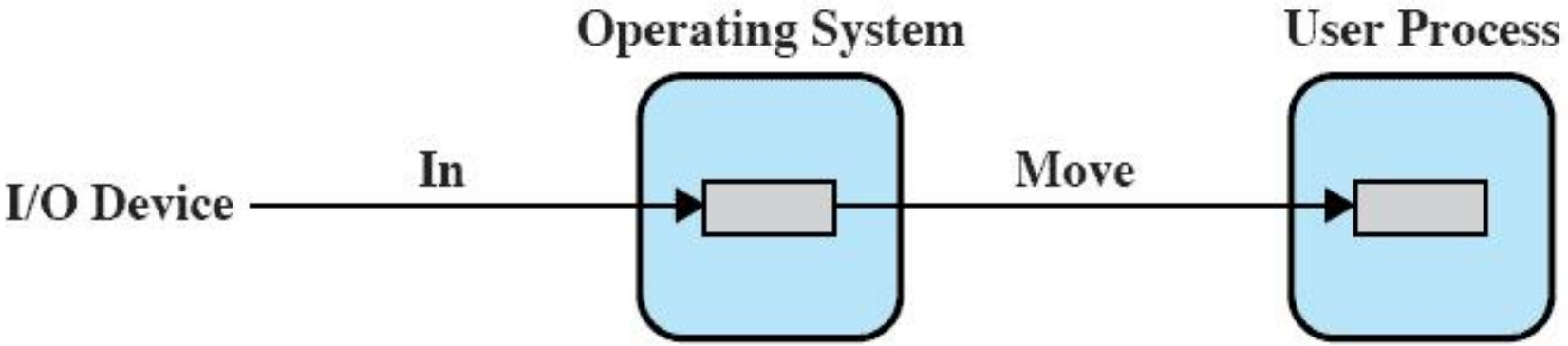
11.4 I/O缓冲 ★★★★★

- 引入缓冲的目的
  - 改善中央处理器与外围设备之间速度不配的矛盾
  - 提高CPU和I/O设备的并行性
- 例：某个用户进程需要从磁盘中读入多个数据块，对磁盘单元执行一个I/O命令，并等待（忙等或进程挂起）数据传送完毕。存在的问题：
  - 程序被挂起，等待相对比较慢的I/O完成。
  - 干扰了操作系统的交换决策。
- 面向块的I/O设备
  - 磁盘、U盘
- 面向流的I/O设备
  - 打印机、通信端口、鼠标、终端等

11.4.1 单缓冲★★★★★

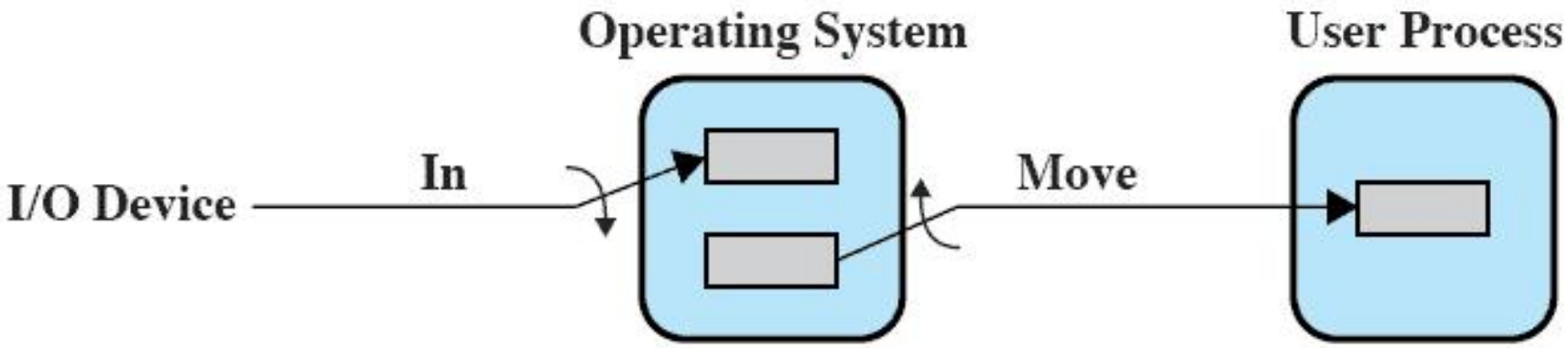


(a) No buffering



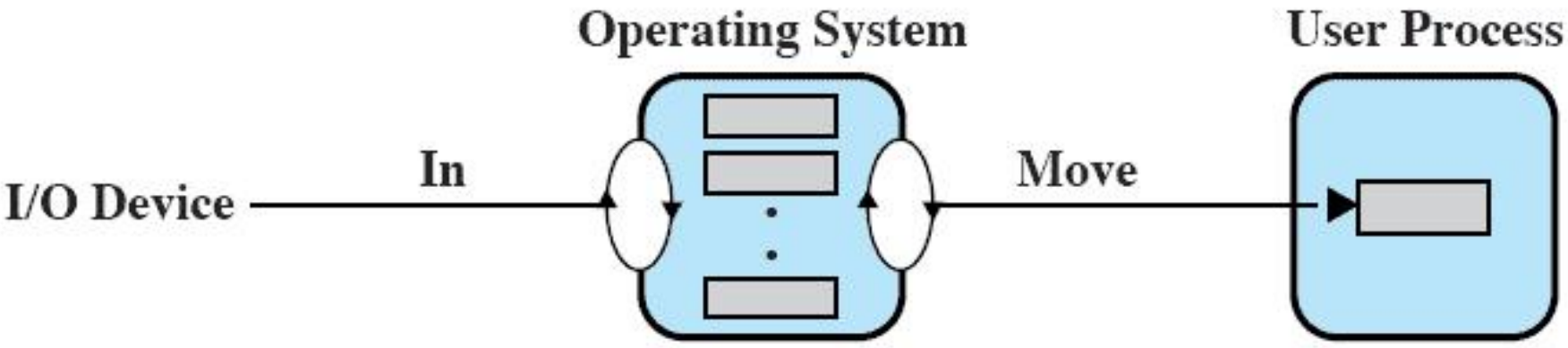
(b) Single buffering

11.4.2 双缓冲★★★★★



(c) Double buffering

11.4.3 循环缓冲 ★★★★★

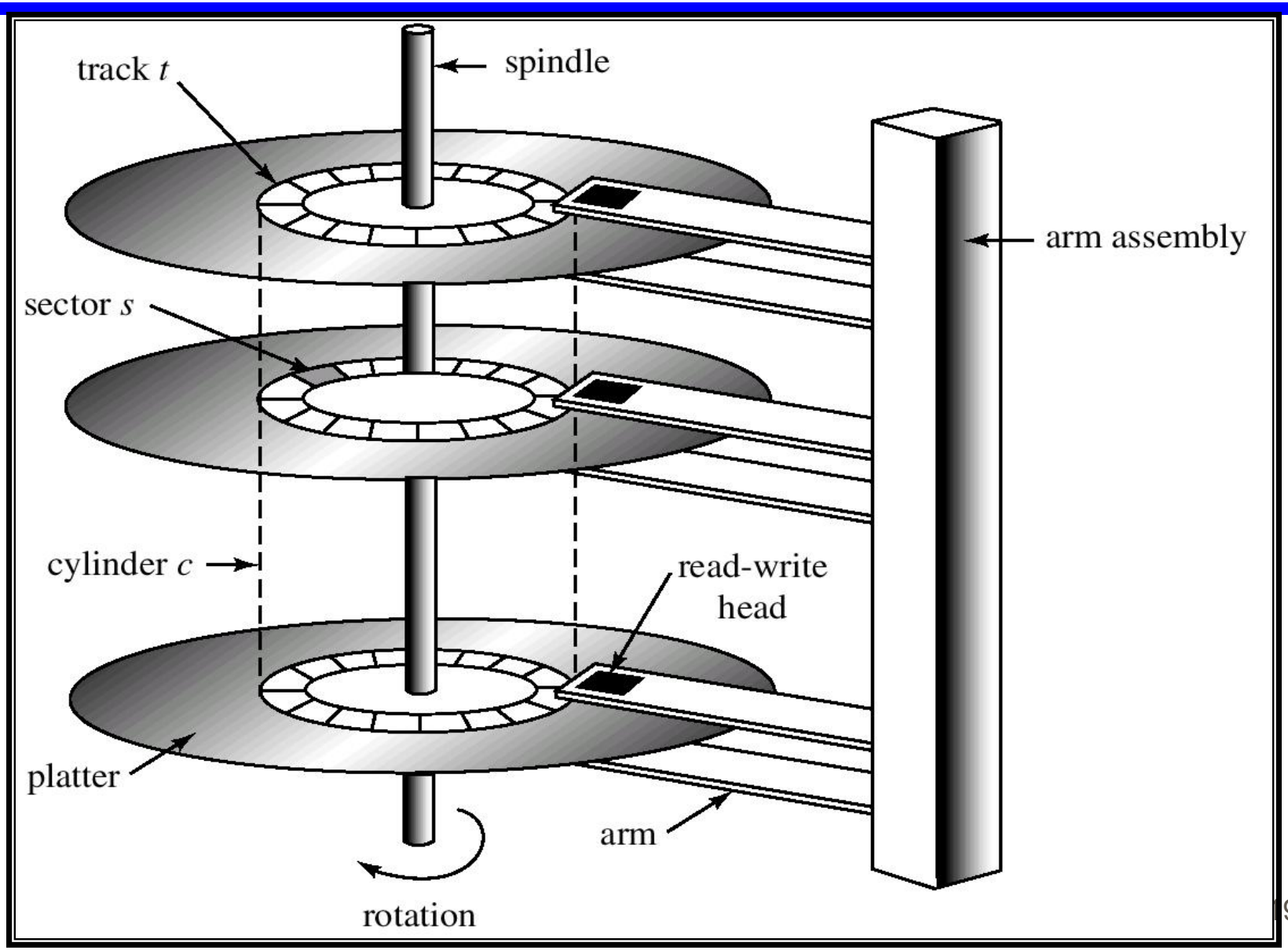


(d) Circular buffering

11.4.4 缓冲的作用★★★★★

- 在多道程序设计环境中，当存在多种I/O活动和多种进程活动时，缓冲是提高操作系统效率和单个进程性能的一种方法。
- 但当进程的平均需求大于I/O设备的服务能力时，缓冲再多也不能让I/O设备与这个进程一直并驾齐驱。

11.5 磁盘调度★★★★★


track t
spindle
arm assembly
sector s
cylinder c
read-write
head
platter
arm
rotation

11.5.1 磁盘性能参数★★★★★

- 寻道时间
  - 将磁头臂移到指定磁道所需要的时间。
- 旋转延迟
  - 将磁盘的待访问地址区域旋转到读/写磁头可访问的位置所需要的时间。
- 传输时间
  - 读或写操作的数据传输所需的时间。

$$T = \frac{b}{rN}$$

- T：传输时间
- b：要传送的字节数
- r：旋转速度
- N：一个磁道中的字节数

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

- T_a：平均存取时间
- T_s：平均寻道时间

时序比较

- 磁盘
  - 平均寻道时间：4ms，转速：7500rpm，每个磁道500个扇区，每个扇区512字节。
- 读取
  - 文件：包含2500个扇区，大小为1.28M。
- 文件顺序组织
  - 文件占据了5个相邻磁道的中的所有扇区。

顺序访问

- 读第一个磁道的时间
  - 平均寻道：4ms
  - （平均）旋转延迟：4ms

    $$\frac{1}{2} * \frac{1}{\frac{7500}{60}} * 1000 = \frac{60}{7500} * 500 = 4(ms)$$

  - 读取500个扇区：8ms

    $$\frac{500*512B}{\frac{7500}{60}*500*512B} * 1000 = 8(ms)$$

- 读取其余磁道的时间
  - 4+8=12ms
- 读取整个文件总的时间（假设可以不需寻道时间读取其余磁道）
  - （4+4+8）+12*4=64ms

随机访问

- 对于每个扇区
  - 平均寻道：4ms
  - 旋转延迟：4ms
  - 读1个扇区：0.016ms

$$\frac{512B}{\frac{7500}{60}*500*512B} * 1000 = \frac{120}{7500} = 0.016(ms)$$

- 读取整个文件总的时间
  - （4+4+0.016）×2500=20040ms

➢ 从磁盘读扇区的顺序对I/O的性能有很大的影响。

11.5.2 磁盘调度策略★★★★★

- FIFO
- SSTF
- SCAN
- C-SCAN

1、FIFO调度

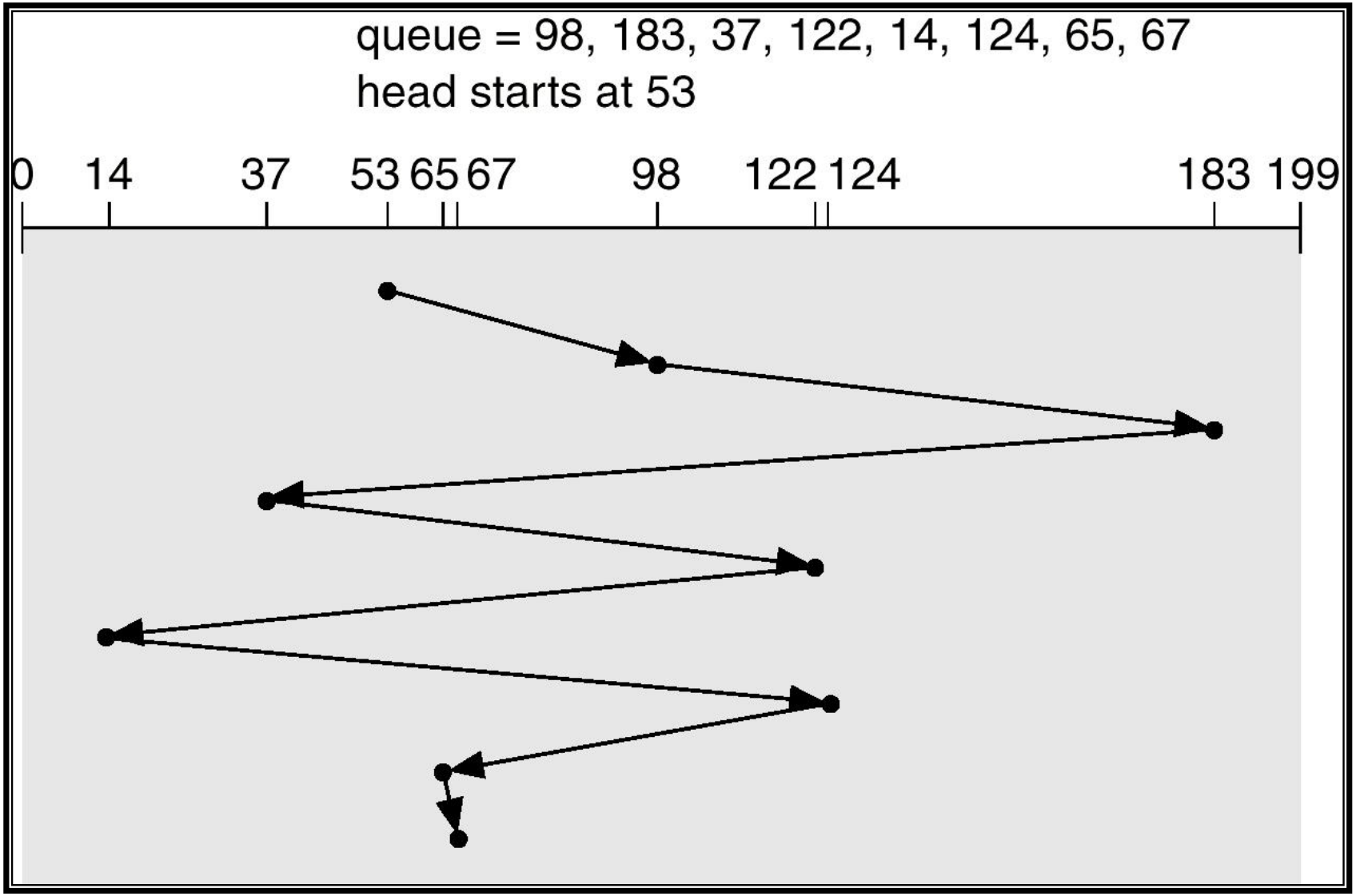
- 先进先出（ FIFO ）：按请求到来的顺序进行服务
- 优点：比较公平
- 缺点：通常不能提供最快的服务

有一个磁盘队列，其I/O请求顺序如下：

98，183，37，122，14，124，65，67

磁头开始位于53

FIFO：平均寻道长度为640/8=80



2、SSTF调度

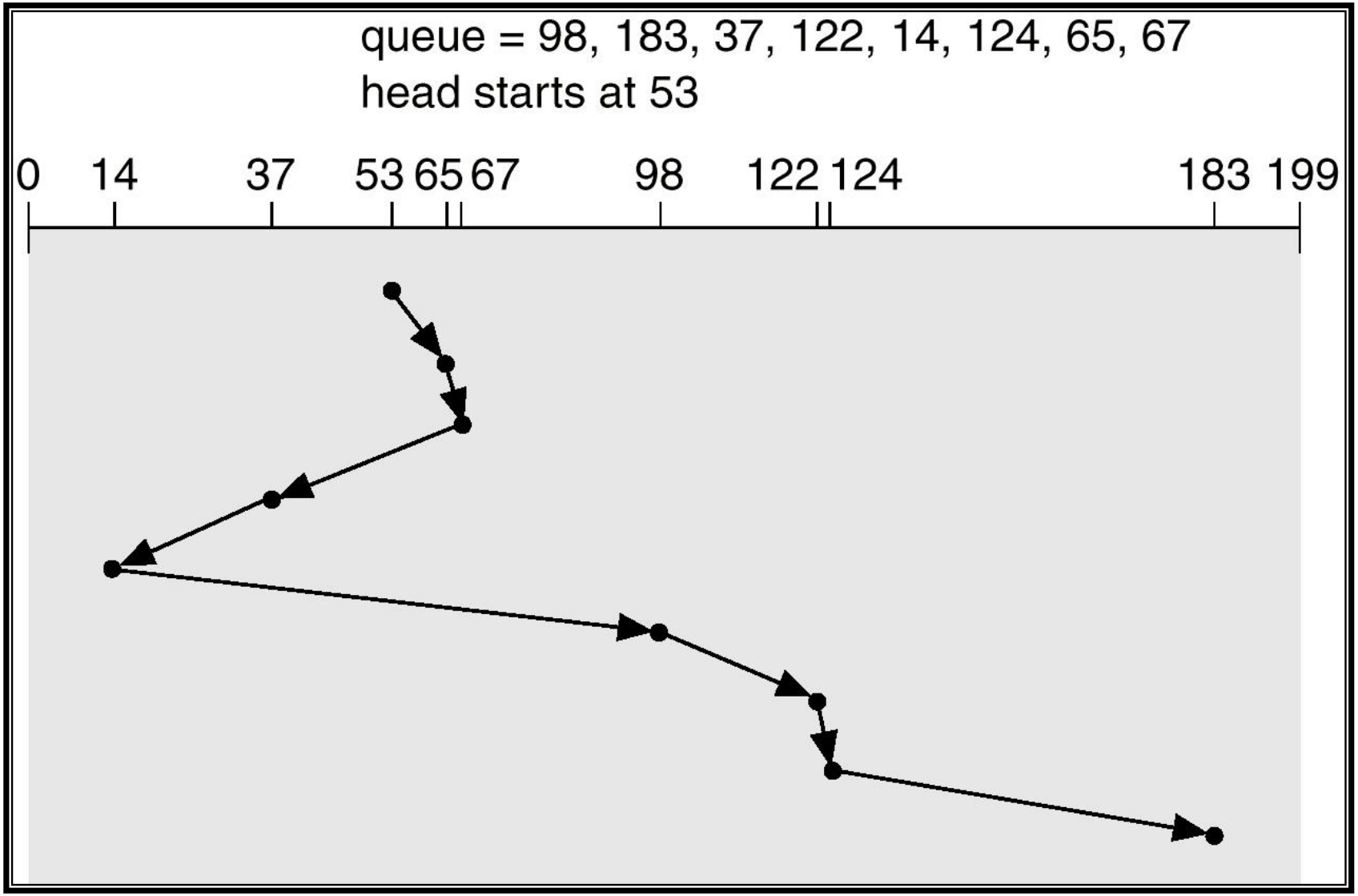
- 最短寻道时间优先算法（SSTF）
  - 从当前磁头位置选择最短寻道时间的请求，即选择与当前磁头位置最近的待处理请求。
- 优点：较FCFS大大提高了性能
- 缺点：可能会导致一些请求得不到服务，并不是最佳。

有一个磁盘队列，其I/O请求顺序如下：

98，183，37，122，14，124，65，67

磁头开始位于53

SSTF：平均寻道长度为236/8=29.5



3、SCAN调度

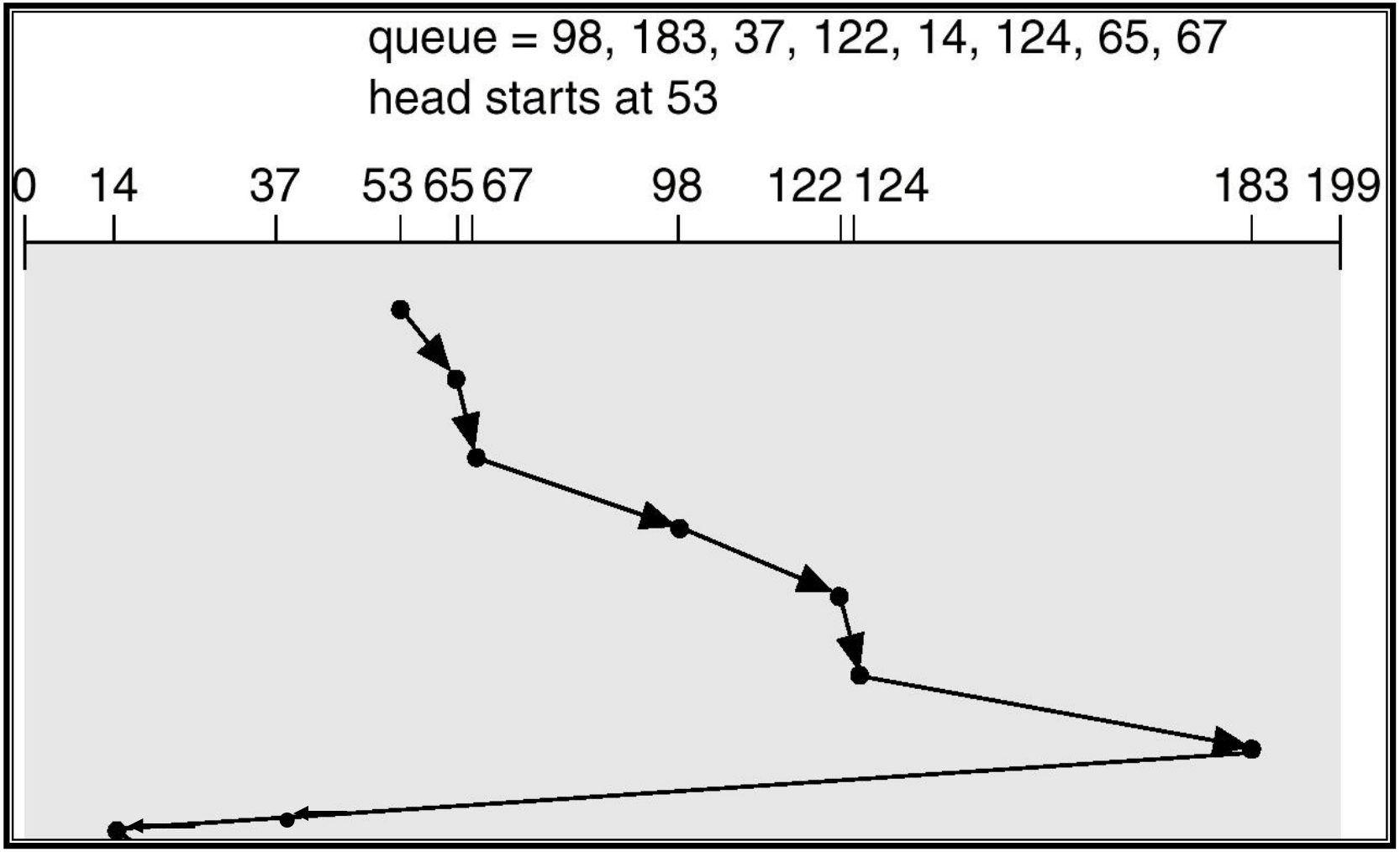
- SCAN算法又称电梯算法
  - 磁臂从磁盘的一端向另一端移动，同时当磁头移过每个柱面时，处理位于该柱面上的服务请求。当到达另一端时，磁头改变方向，处理继续。
- 需要知道磁头的当前位置和磁头移动的方向。
- 某些请求处理可能不及时。

有一个磁盘队列，其I/O请求顺序如下：

98，183，37，122，14，124，65，67

磁头开始位于53，向大方向移动

SCAN：平均寻道长度为299/8=37.375



4、C-SCAN调度

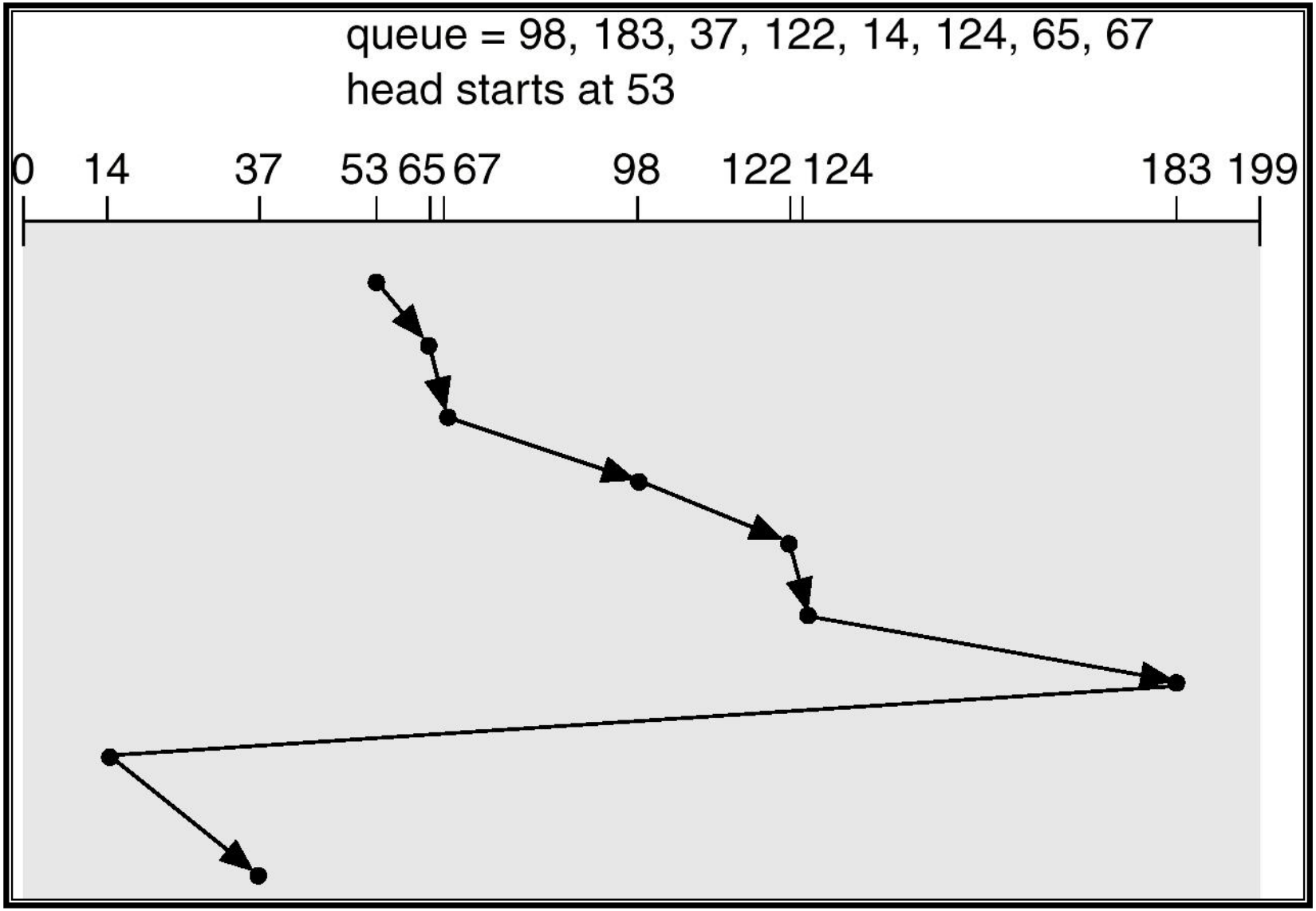
- SCAN调度的变种
  - 将磁头从磁盘一端移到另一端，随着移动不断的处理请求。不过，当磁头移到另一端时，马上返回，返回时不处理请求。

有一个磁盘队列，其I/O请求顺序如下：

98，183，37，122，14，124，65，67

磁头开始位于53，向大方向移动

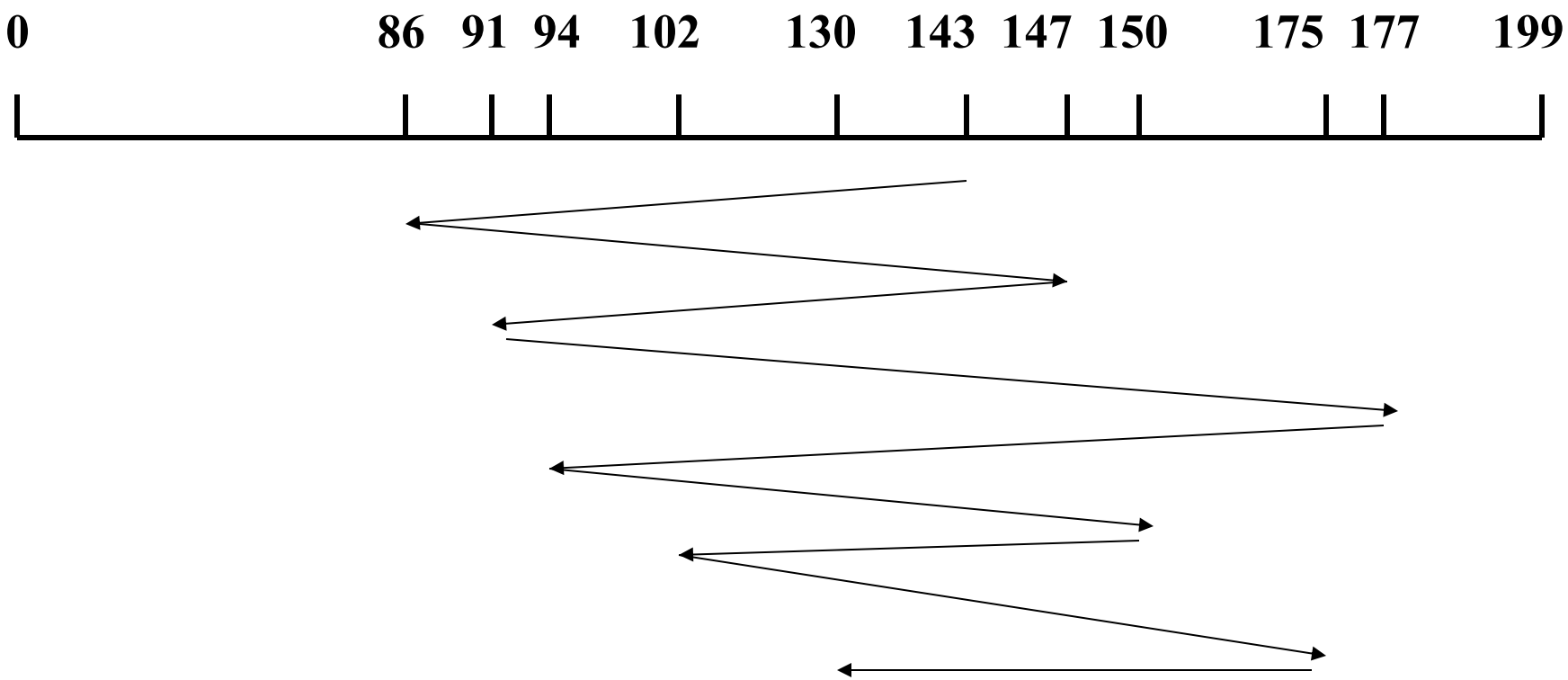
C-SCAN ：平均寻道长度为322/8=40.25



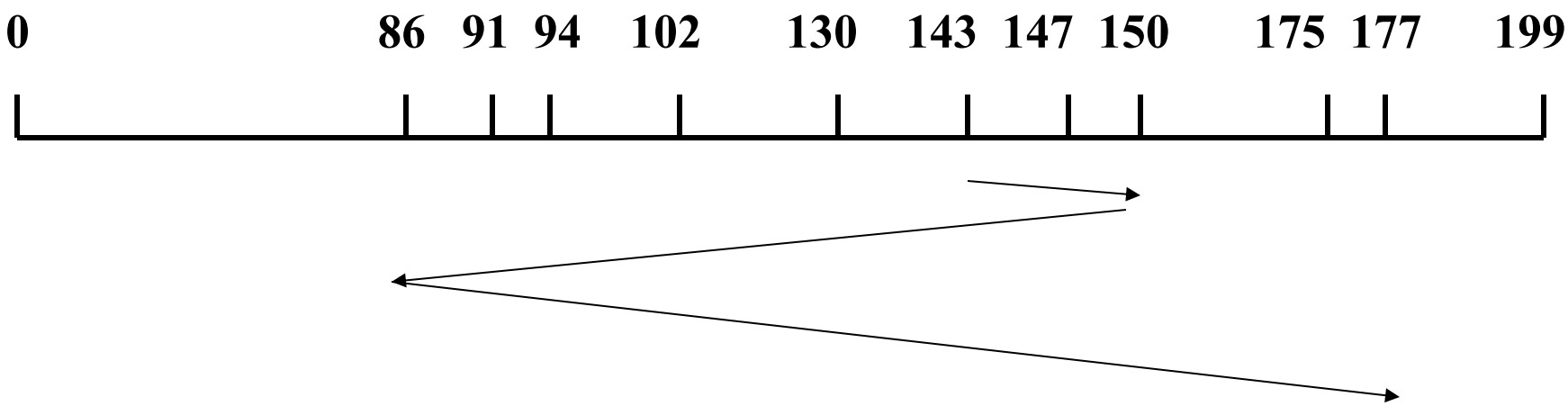
✓调度算法练习

- 假设移动头磁盘有200个磁道（0—199）。目前正在处理143号磁道上的请求，而刚刚处理结束的请求是125号，如果下面给出的顺序是按FIFO算法排成的等待服务队列顺序：86，147，91，177，94，150，102，175，130那么，用下列各种磁盘调度算法来满足这些请求所需的平均寻道长度是多少？（1）FIFO（2）SSTF（3）SCAN（4）C-SCAN

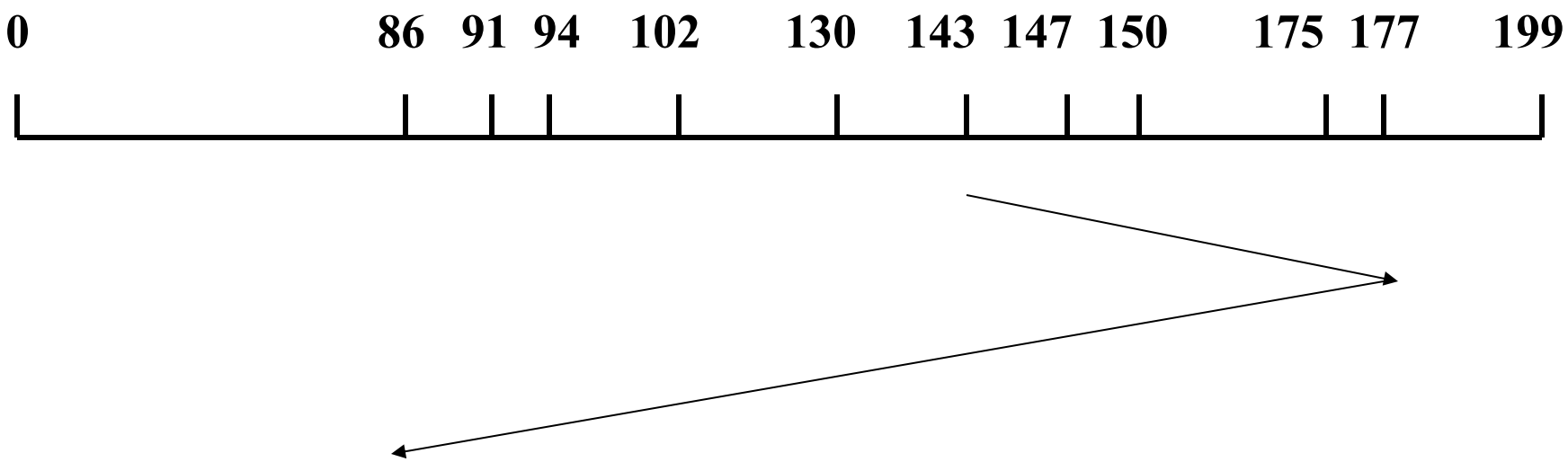
（1）FIFO：565/9=62.78



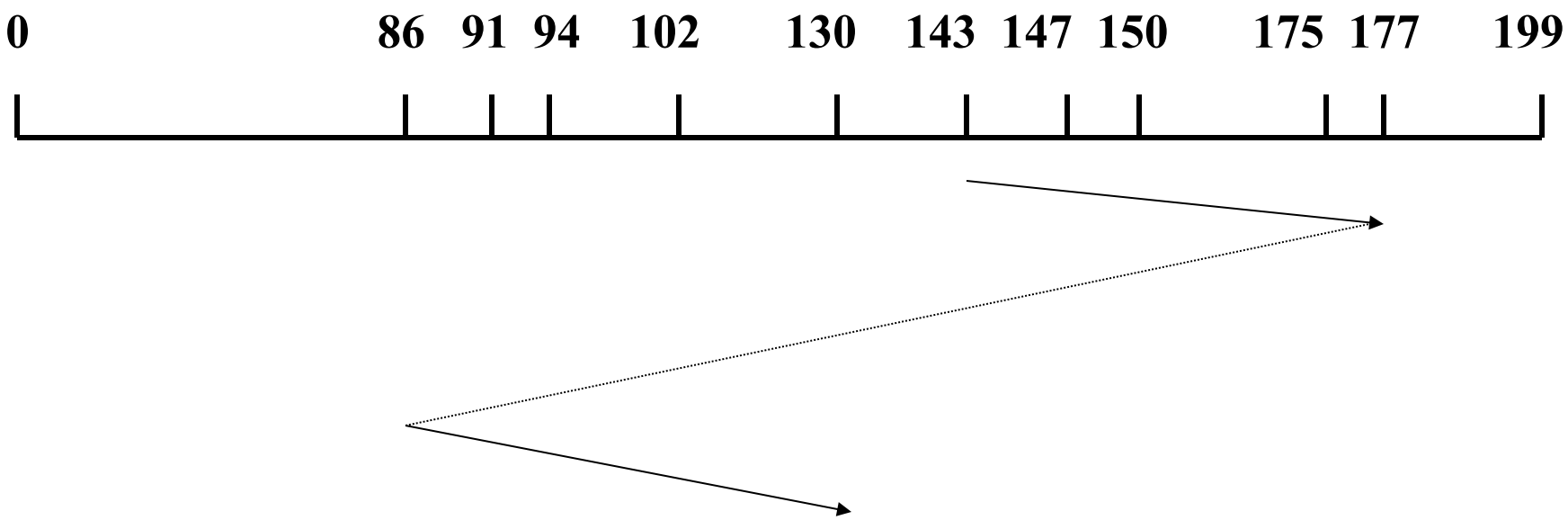
（2）SSTF：162/9=18



（3）SCAN：125/9=13.89



（4）C_SCAN：169/9=18.78



其它磁盘调度方法

- N-step-SCAN
  - 把磁盘请求队列分成长度为N的子队列，每一次用SCAN处理一个子队列。在处理某一个队列时，新请求必须添加到其他某个队列中。
- FSCAN
  - 使用两个子队列。当扫描开始时，所有请求都在一个队列中，而另一个队列为空。在扫描过程中，所有新到的请求都被放入另一个队列中。

磁盘调度算法比较

表 11.3　磁盘调度算法

名　称	说　明	注　释
根据请求者选择		
随机	随机调度	用于分析和模拟
FIFO	先进先出	最公平的调度
PRI	进程优先级	在磁盘队列管理之外控制
LIFO	后进先出	局部性最好，资源利用率最高
根据被请求项选择		
SSTF	最短服务时间优先	利用率高，队列小
SCAN	在磁盘上往复	服务分布比较好
C-SCAN	单向，快速返回	服务变化较低
N 步 SCAN	一次 *N* 个记录的 SCAN	服务保证
FSCAN	*N* 步扫描，*N* 等于 SCAN 循环开始处的队列大小	负载敏感

11.6 RAID ★★★★★

- 独立磁盘冗余阵列（Redundant Array of Independent Disk）
- RAID方案包括了7个级别，从0到6。不同级别表明了不同的设计体系结构，有三个共同特性：
  - RAID是一组物理磁盘驱动器，操作系统把它视为一个单个的逻辑驱动器。
  - 数据分布在物理驱动器阵列中--条带化。
  - 使用冗余的磁盘容量保存奇偶检验信息，从而保证当一个磁盘失效时，数据具有可恢复性。【RAID0,RAID1不支持该特性。】
- 常用的：RAID 0、RAID 1，RAID 5、RAID 6

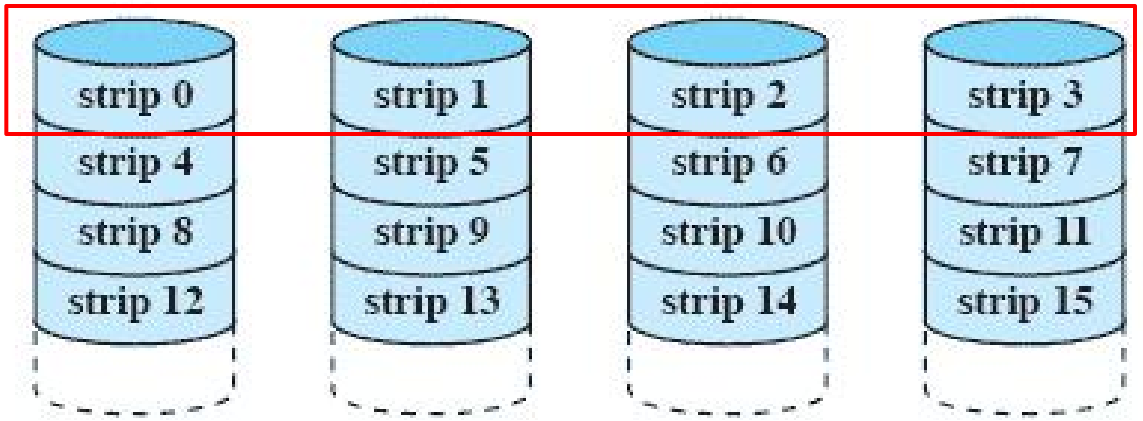
11.6 RAID ★★★★★

表 11.4 RAID 级别

类别	级别	说明	磁盘请求	数据可用性	大 I/O 数据量传送能力	小 I/O 请求率
条带化	0	非冗余	N	低于单个磁盘	很高	读和写都很高
镜像	1	被镜像	$2N$	高于 RAID2、3、4 或 5；低于 RAID6	读时高于单个磁盘；写时与单个磁盘相近	读时最快为单个磁盘的两倍；写时与单个磁盘相近
并行访问	2	通过汉明码实现冗余	$N+m$	明显高于单个磁盘；与 RAID3、4 或 5 可比	所有列出方案中最高的	约为单个磁盘的两倍
	3	交错位奇偶校验	$N+1$	明显高于单个磁盘；与 RAID2、4 或 5 可比	所有列出方案中最高的	约为单个磁盘的两倍
独立访问	4	交错块奇偶校验	$N+1$	明显高于单个磁盘；与 RAID2、3 或 5 可比	读时与 RAID0 相近；写时明显慢于单个磁盘	读时与 RAID0 相似；写时明显慢于单个磁盘
	5	交错块分布奇偶校验	$N+1$	明显高于单个磁盘；与 RAID2、3 或 4 可比较	读时与 RAID0 相近；写时慢于单个磁盘	读时与 RAID0 相似；写时通常慢于单个磁盘
	6	交错块双重分布奇偶校验	$N+2$	所有列出方案中最高的	读时与 RAID0 相近；写时慢于 RAID5	读时与 RAID0 相近；写时明显慢于 RAID5

N 表示数据磁盘数量；m 与 $\log N$ 成比例。

RAID 0

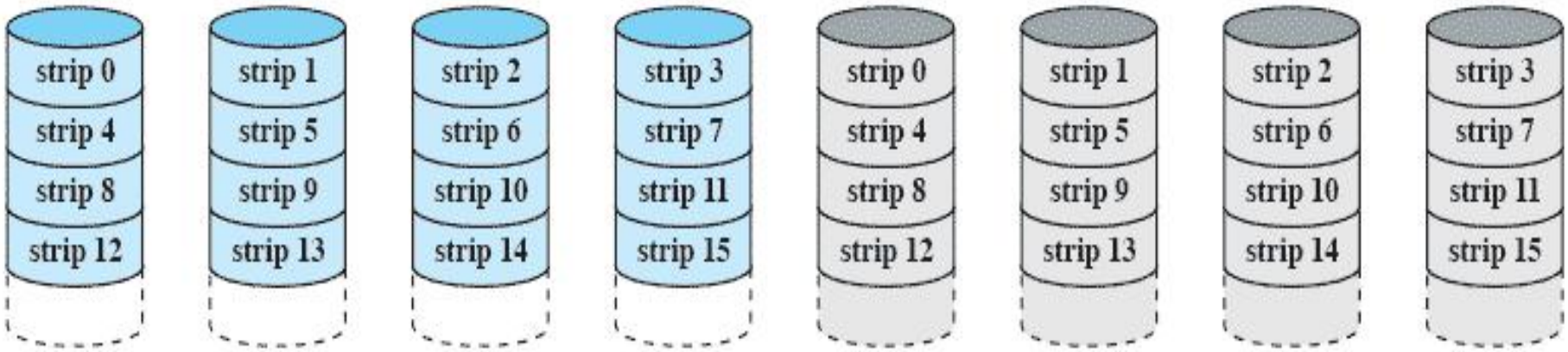


(a) RAID 0 (non-redundant)

条带化

- 一个条带可以是一个物理块、扇区或别的某种单元。条带被循环映射到连续的阵列成员中。
- 一组逻辑上连续的条带，如果恰好一个条带映射到一个阵列成员上，则称为一条条带。
- 优点：如果一个I/O请求由多个逻辑上连续的条带组成，该请求可以并行处理，从而减少I/O传输时间。

RAID 1 ★★★★★

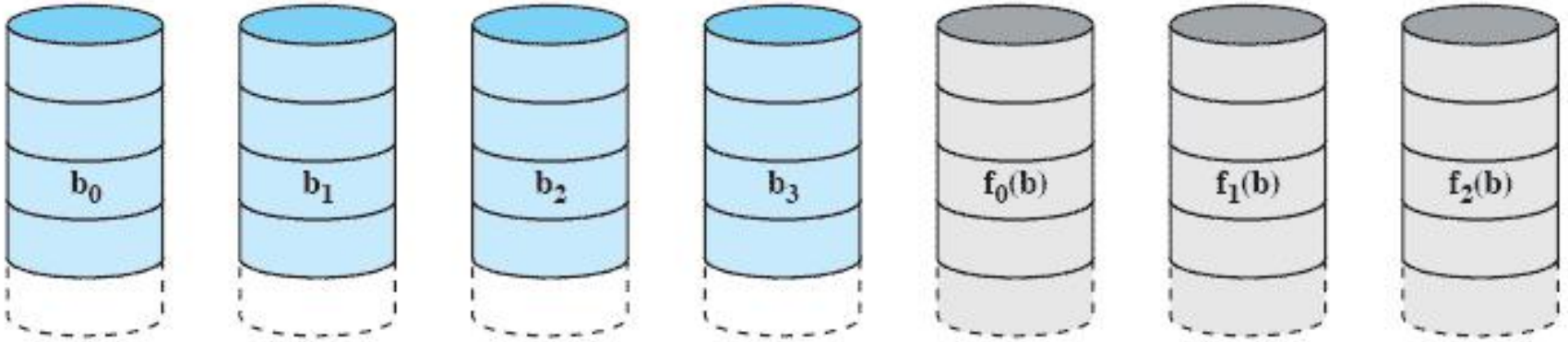


(b) RAID 1 (mirrored)

镜像

- **每个逻辑条带映射到两个单独的物理磁盘上，使得阵列中的每个磁盘都有一个包含相同数据的镜像磁盘。**
- **可靠性较好，但成本较高。**

RAID 2 ★★★★★



(c) RAID 2 (redundancy through Hamming code)

错误校正码

- **每个数据磁盘中的相应位都计算一个错误校正码，这个码位保存在多个奇偶检验磁盘中相应的位中。**
- **成本仍然较高。**

RAID 2 ★★★★★

汉明码（纠错1位）

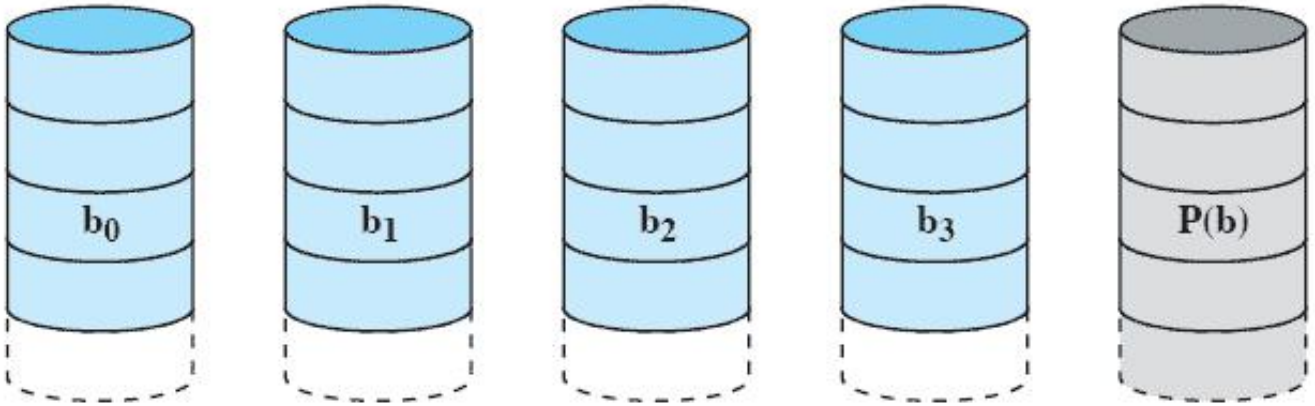
- 1计算校验位数
- 2 确定校验码位置
- 3 确定校验码
- 4 实现校验和纠错

假设用N表示添加了校验码位后整个信息的二进制位数，用K代表其中有效信息位数，r表示添加的校验码位，它们之间的关系应满足：

$$N=K+r\leq 2^{r}-1$$

https://www.cnblogs.com/CheeseIce/p/11079362.html

RAID 3 ★★★★★



(d) RAID 3 (bit-interleaved parity)

奇偶校验位

- **为所有数据磁盘中同一位置的位的集合计算一个简单的奇偶校验位，而不是错误校正码。**
- **以较低的成本保证较好的可靠性。**

例

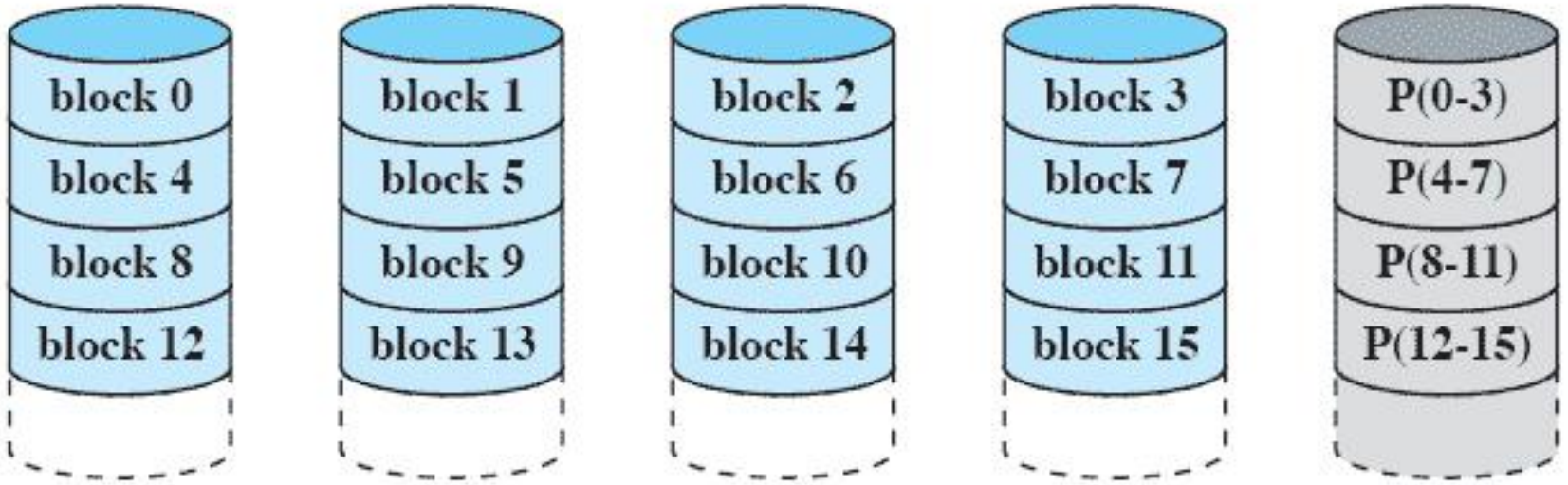
- 考虑一个有5个驱动器的阵列，其中X_0到X_3包含数据，X_4为奇偶校验磁盘，第i位的奇偶校验可计算如下：

$$X_4(i) = X_3(i) \oplus X_2(i) \oplus X_1(i) \oplus X_0(i)$$

- 假设驱动器X1失效，如果给上面的等式两边都加上$X_4(i) \oplus X_1(i)$，则有：

$$X_1(i) = X_4(i) \oplus X_3(i) \oplus X_2(i) \oplus X_0(i)$$

RAID 4 ★★★★★



(e) RAID 4 (block-level parity)

- 每个条的写操作都包含两次读和两次写。
- 奇偶检验驱动器和数据驱动器可以并行更新。
- 由于每次写操作都必须包含奇偶校验，因此奇偶检验磁盘有可能成为瓶颈。

RAID 4 ★★★★★

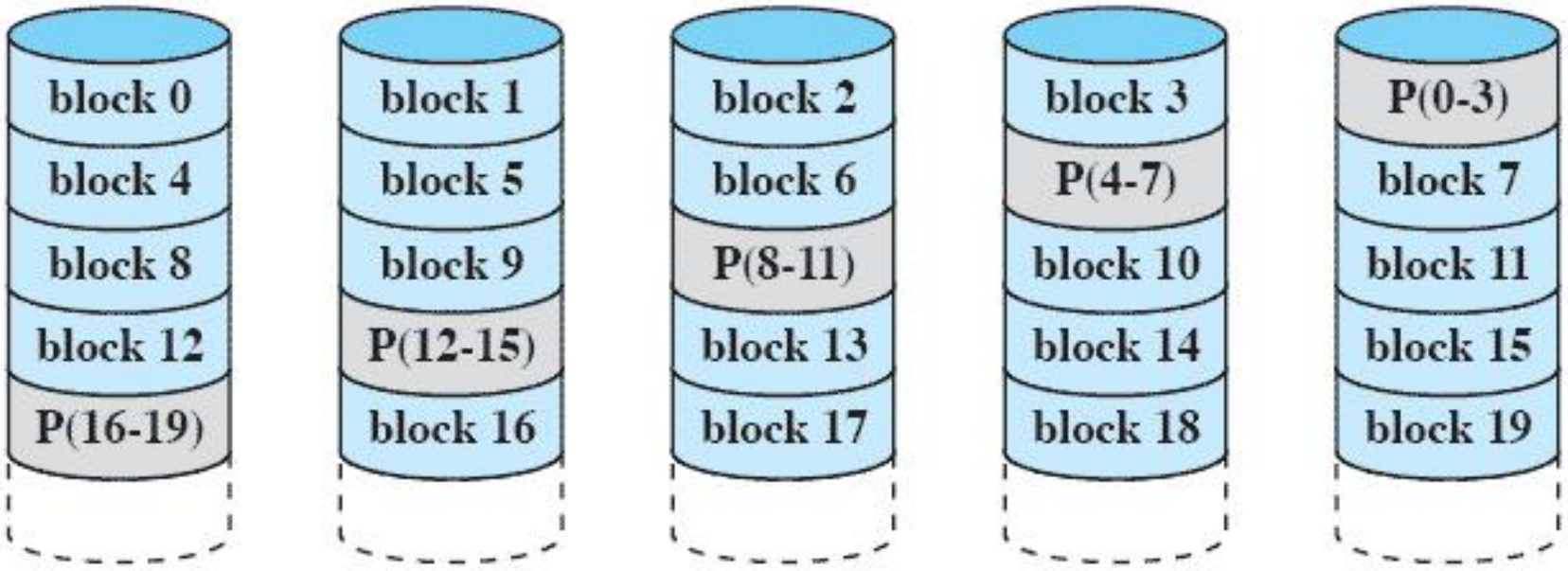
Consider an array of five drives in which X0 through X3 contain data and X4 is the parity disk. Suppose that a write is performed that only involves a strip on disk X1. Initially, for each bit i, we have the following relationship:

$$X4(i) = X3(i) \oplus X2(i) \oplus X1(i) \oplus X0(i) \quad \textbf{(11.1)}$$

After the update, with potentially altered bits indicated by a prime symbol:

$$\begin{aligned} X4'(i) &= X3(i) \oplus X2(i) \oplus X1'(i) \oplus X0(i) \\ &= X3(i) \oplus X2(i) \oplus X1'(i) \oplus X0(i) \oplus X1(i) \oplus X1(i) \\ &= X3(i) \oplus X2(i) \oplus X1(i) \oplus X0(i) \oplus X1(i) \oplus X1'(i) \\ &= X4(i) \oplus X1(i) \oplus X1'(i) \end{aligned}$$

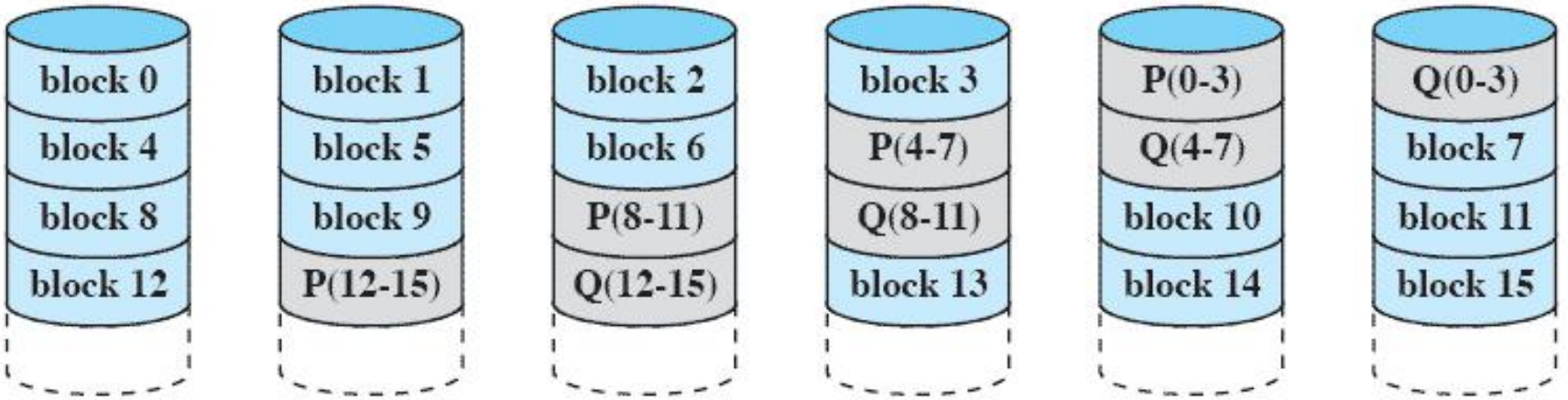
RAID 5 ★★★★★



(f) RAID 5 (block-level distributed parity)

- 避免RAID 4中一个奇偶校验的潜在I/O瓶颈。

RAID 6 ★★★★★



(g) RAID 6 (dual redundancy)

- 提供数据极高的可用性。
- 写性能损失。

11.7 磁盘高速缓存★★★★★

- 磁盘高速缓存是内存中为磁盘扇区设置的一个缓冲区，它包含有磁盘中某些扇区的副本。
- 当出现一个请求某一特定扇区的I/O请求时，首先进行检测，以确定该扇区是否在磁盘高速缓存中。
  - 如果在，则该请求可以通过这个高速缓存来满足；
  - 如果不在，则把被请求的扇区从磁盘读到磁盘高速缓存中。

11.7.1 设计考虑因素

- 当一个I/O请求从磁盘高速缓存中得到满足时，磁盘高速缓存中的数据必须传送到发送请求的进程。
  - 在内存中把数据从磁盘高速缓存传送到分配给该用户进程的存储空间中。
  - 使用一个共享内存，传送指向磁盘高速缓存中相应项的指针。
- 当一个新扇区被读入磁盘高速缓存时，必须置换出来一个已存在的块。置换策略选择：
  - 最近最少使用（LRU）
  - 最不常使用页面置换算法（LFU）：置换集合中访问次数最少的块。（给每个块关联一个计数器）
  - 基于频率的置换算法（新老区；新中老区）

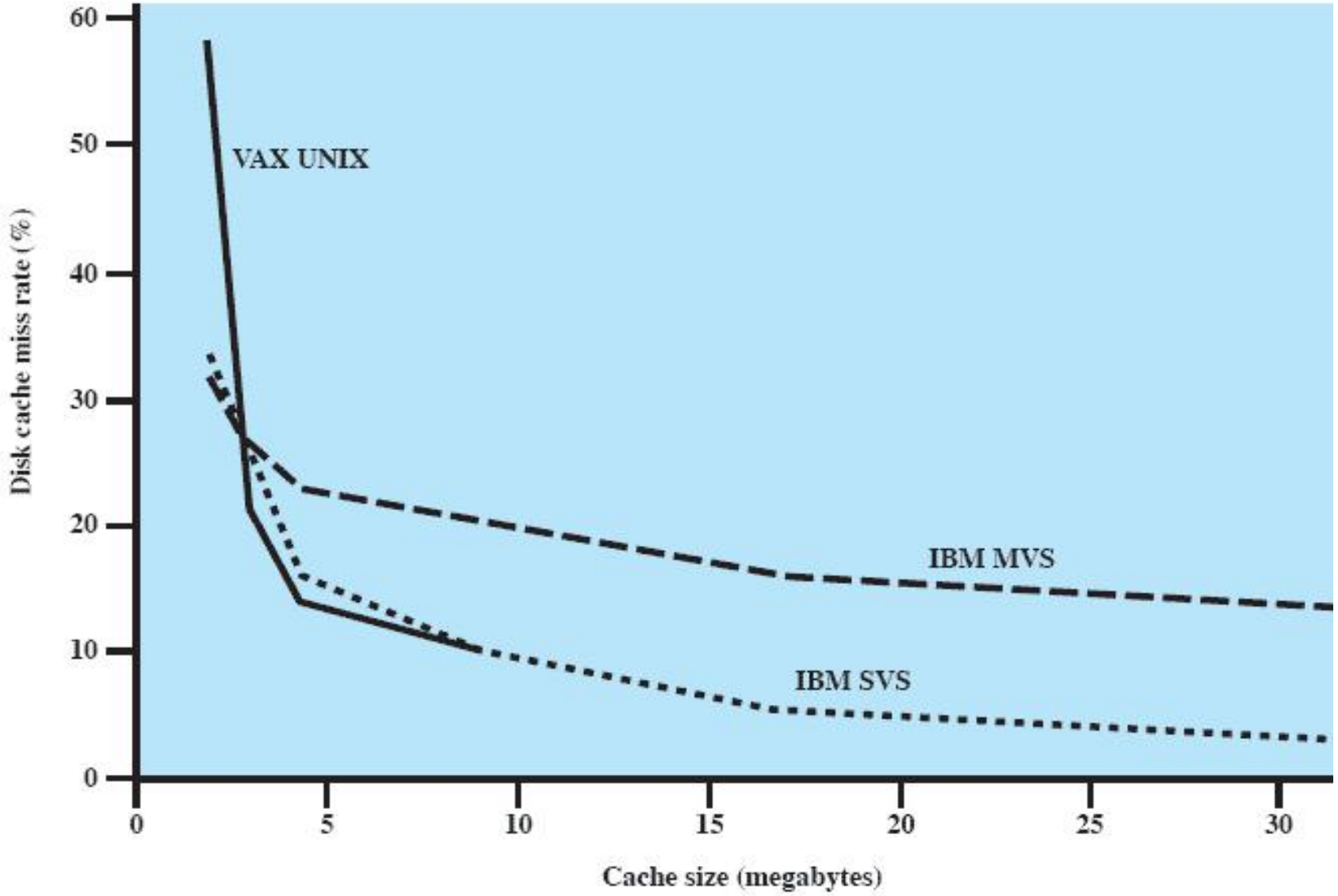


Figure 11.10 Some Disk Cache Performance Results Using LRU

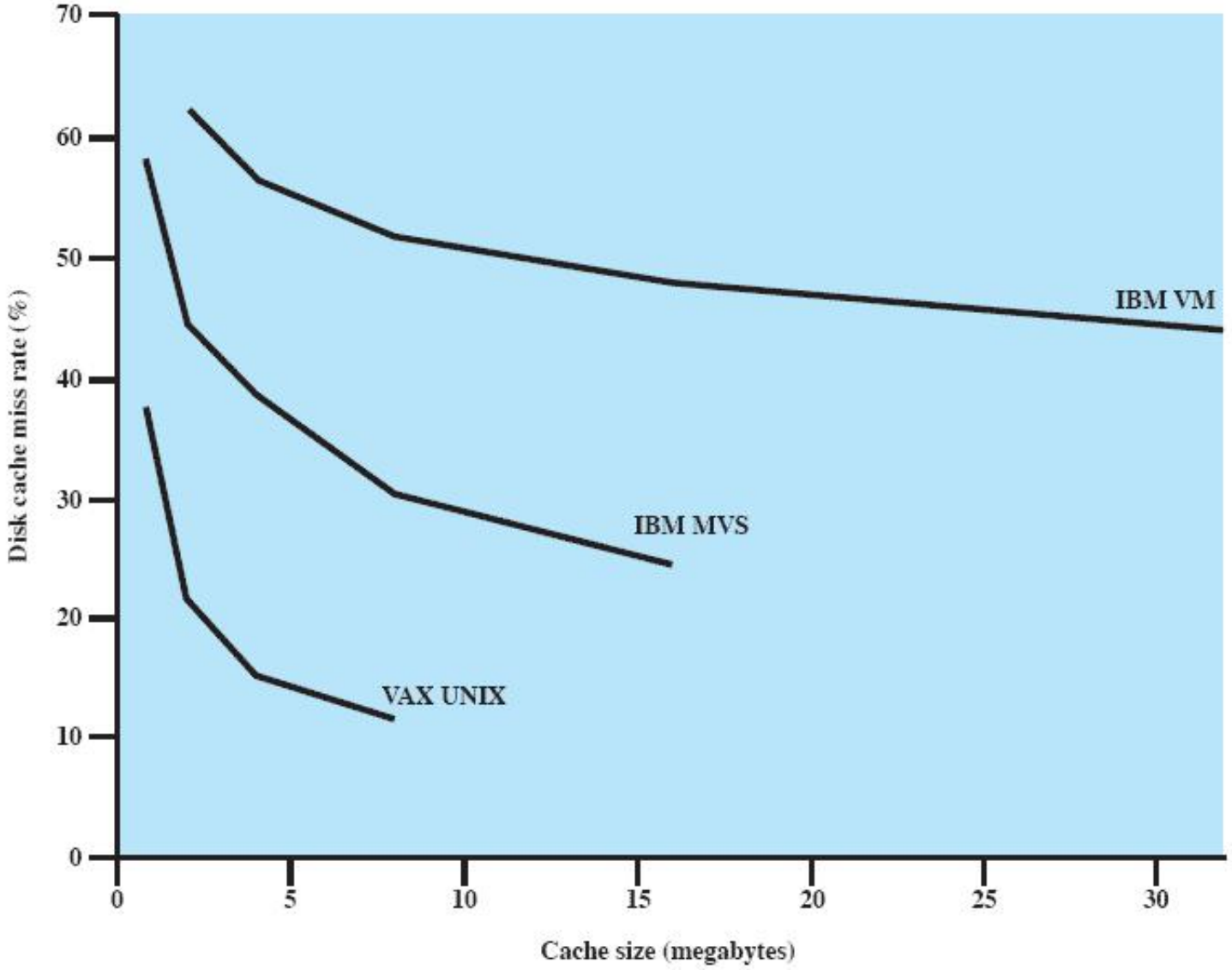


Figure 11.11 Disk Cache Performance Using Frequency-Based Replacement [ROBI90]

非阻塞IO?(异步IO)

Asynchronous I/O enables an application to issue an I/O request and return control immediately to the caller, before the I/O has completed; additional interfaces enable an application to determine whether various I/Os have completed.

- **Want to Read a file on Mac**

(1)Fill the structure before read a file

```
struct aiocb {
    int             aio_fildes;    // File descriptor
    off_t           aio_offset;    // File offset
    volatile void   *aio_buf;      // Location of buffer
    size_t          aio_nbytes;    // Length of transfer
};
```

非阻塞IO?(异步IO?)

(2)Read the file

```
int aio_read(struct aiocb *aiocbp);
```

(3) an application can periodically poll the system via a call to aio_error() to determine whether said I/O has yet completed.

```
int aio_error(const struct aiocb *aiocbp);
```

To remedy the polling issues, some systems provide an approach based on the interrupt. This method uses UNIX signals to inform applications when an asynchronous I/O completes, thus removing the need to repeatedly ask the system.

作业

- 复习题　11.1
- 习题　11.3 （要求按我们上课讲的方式做，即分四种算法，每个要画轨迹图和计算平均寻道长度）