窦轶////yi.dou@njupt.edu.cn

Operating Systems

操作系统

外存储设备管理

外存储设备管理

本讲内容

1. 典型外存储设备类型
2. 硬盘的存储空间管理
3. 硬盘的数据访问时间
4. 硬盘驱动臂调度算法

典型外存储设备类型

1 顺序存取存储设备 磁带

磁头(正走,反走,正读,反读,正写,反写,倒带)

- 顺序存取存储设备是严格依赖信息的物理位置进行定位和读写的存储设备
- 具有容量大、稳定可靠、卷可装卸和便于保存等优点
- 读取不方便，找到某一块数据依赖快进和倒带

典型外存储设备类型

2 直接存取存储设备

- 磁盘是一种典型的直接(随机)存取存储设备
- 每个物理记录有确定的位置和唯一的地址，可直接去快速存取任何一个物理块，达到直接存取的目的

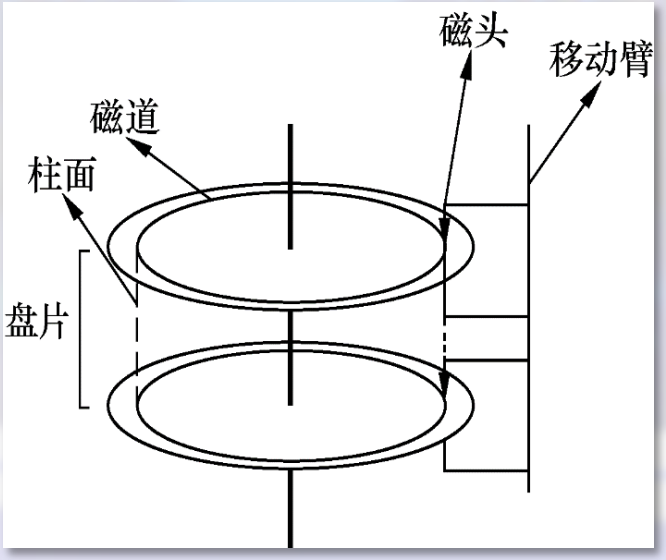
外存储设备管理

本讲内容

1. 典型外存储设备类型
2. 硬盘的存储空间管理
3. 硬盘的数据访问时间
4. 硬盘驱动臂调度算法

硬盘的存储空间管理

- 硬盘常常是多个金属盘面叠在一起，中间有个轴带动所有盘面做高速旋转
- 每个盘面有一个读写磁头，所有的读写磁头都固定在唯一的移动臂上，同时向盘的最里面和最外面移动



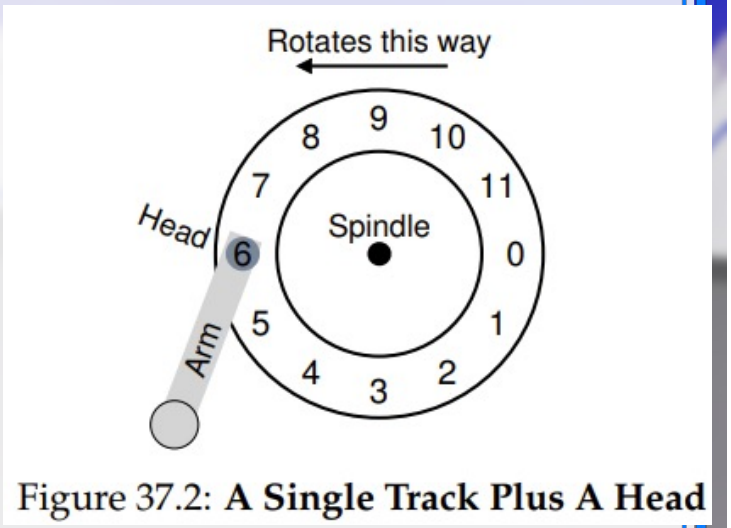
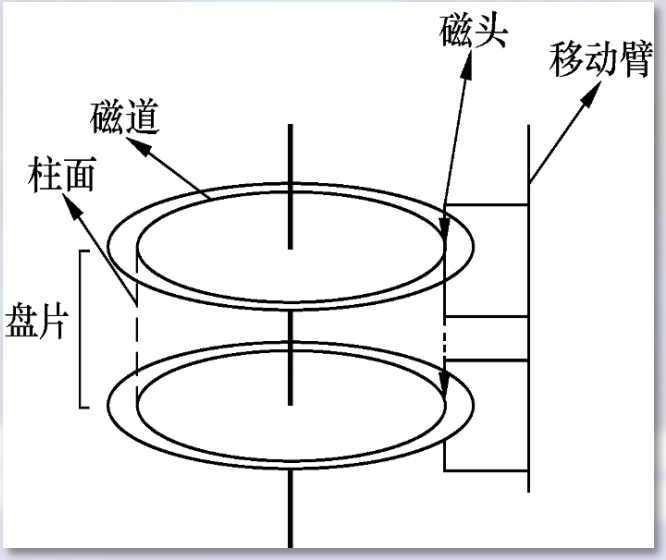


Figure 37.2: **A Single Track Plus A Head**

硬盘的存储空间管理

- 在一个盘面上的读写磁头的轨迹称磁道，在磁头位置下的所有磁道组成的圆柱体称柱面
- 也就是说相同半径的磁道构成为同一个柱面
- 一个磁道可被划分成一个或多个物理块，称为扇区



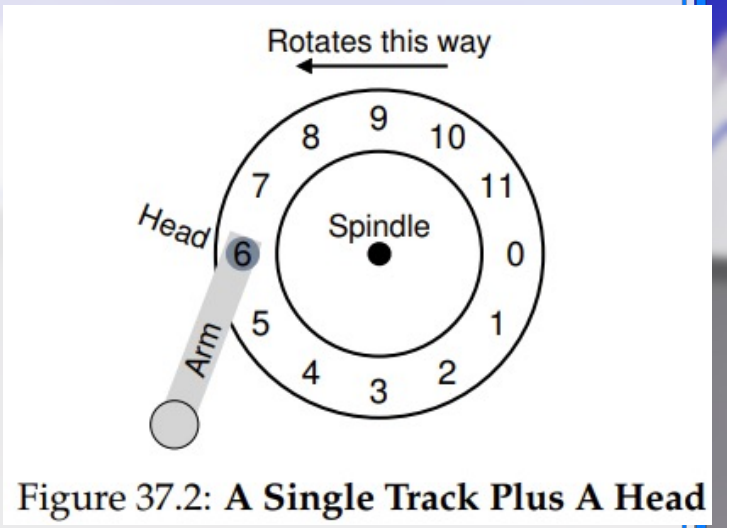
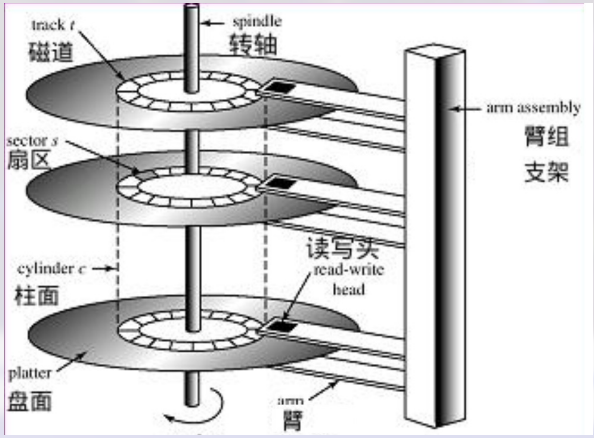


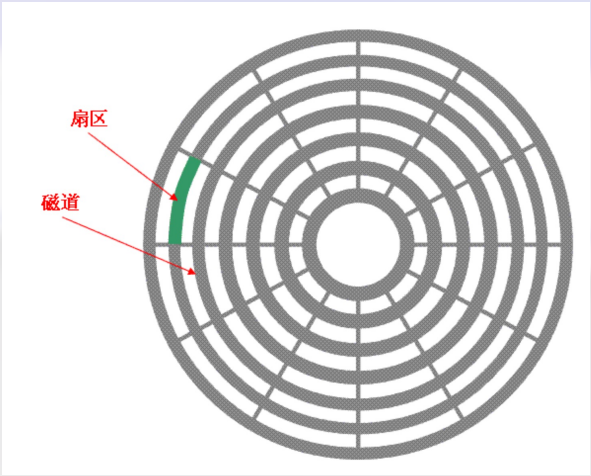
Figure 37.2: **A Single Track Plus A Head**

硬盘的存储空间管理

- 文件的信息通常不是记录在同一盘面的各个磁道上，而是记录在同一柱面的不同磁道上，使移动臂的移动次数减少，缩短存取信息的时间。
- 访问磁盘上的一个物理记录，依赖3个参数：

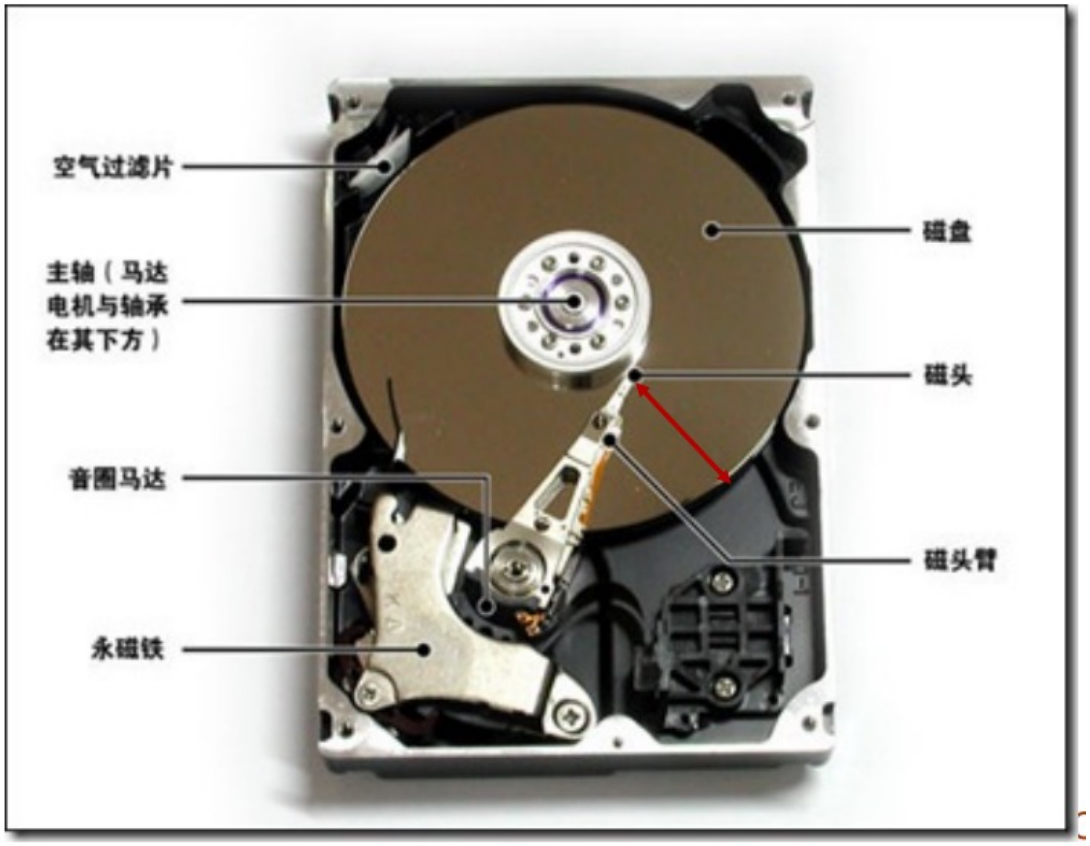
  柱面号，磁头号，块号





磁盘、磁道、扇区

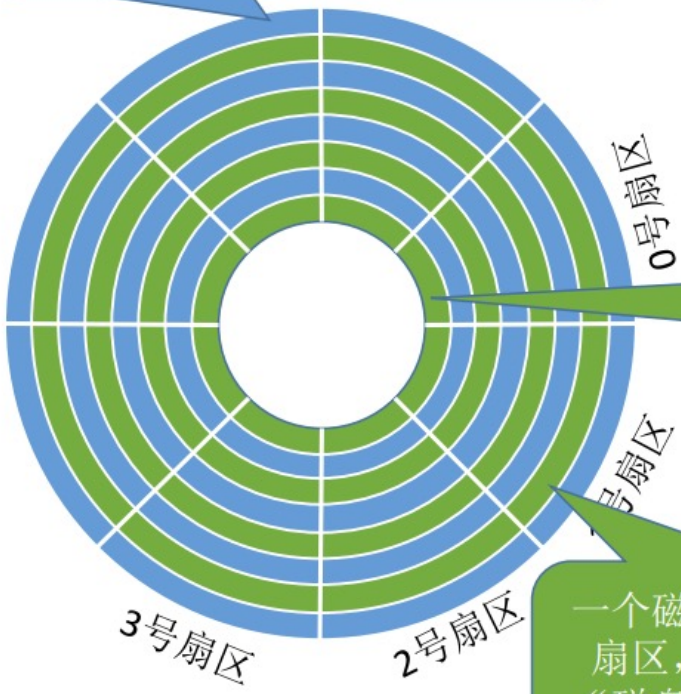
磁盘的表面由一些磁性物质组成，可以用这些磁性物质来记录二进制数据



磁盘、磁道、扇区

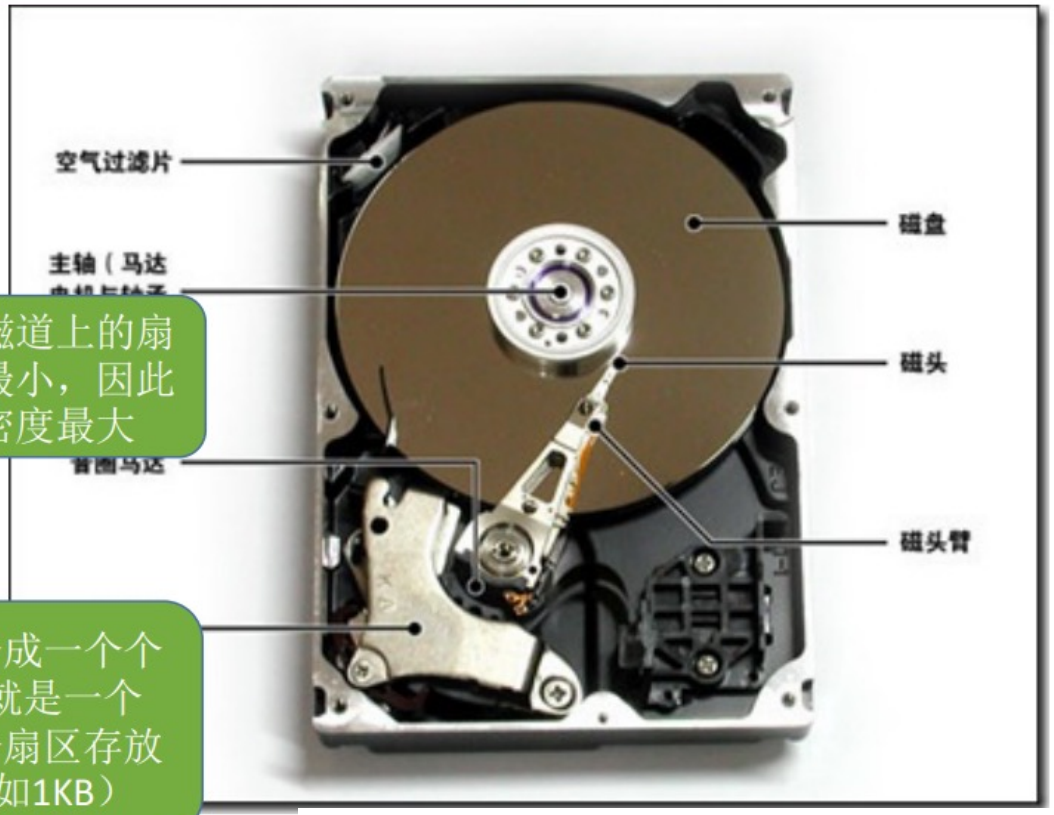
磁盘的表面由一些磁性物质组成，可以用这些磁性物质来记录二进制数据

磁盘的盘面被划分成一个个磁道。这样的一个“圈”就是一个磁道

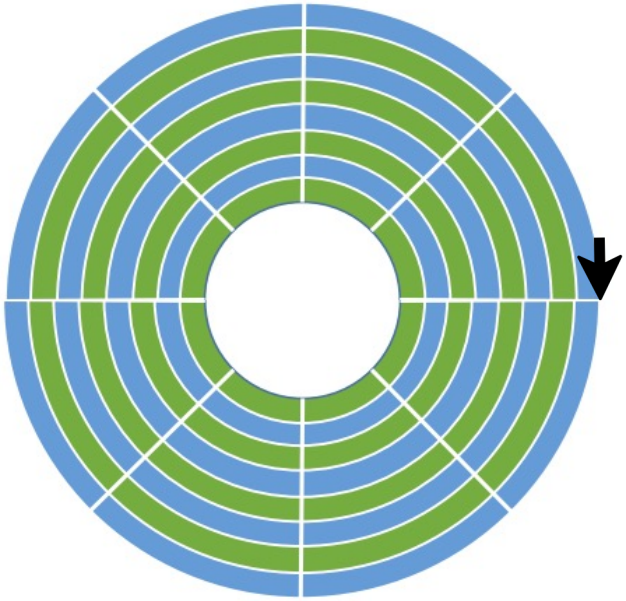


最内侧磁道上的扇区面积最小，因此数据密度最大

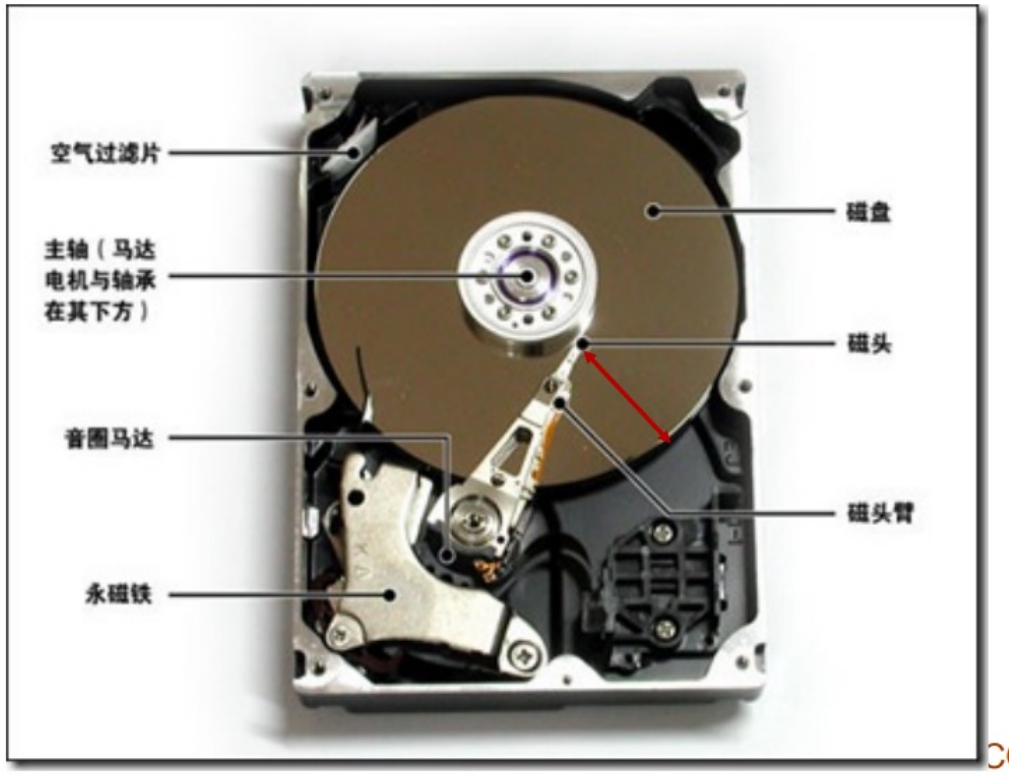
一个磁道又被划分成一个个扇区，每个扇区就是一个“磁盘块”。各个扇区存放的数据量相同（如1KB）



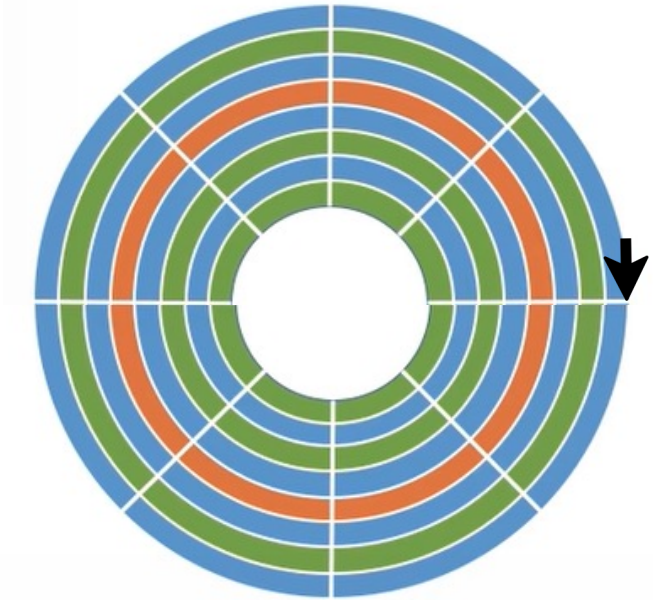
如何在磁盘中读/写数据



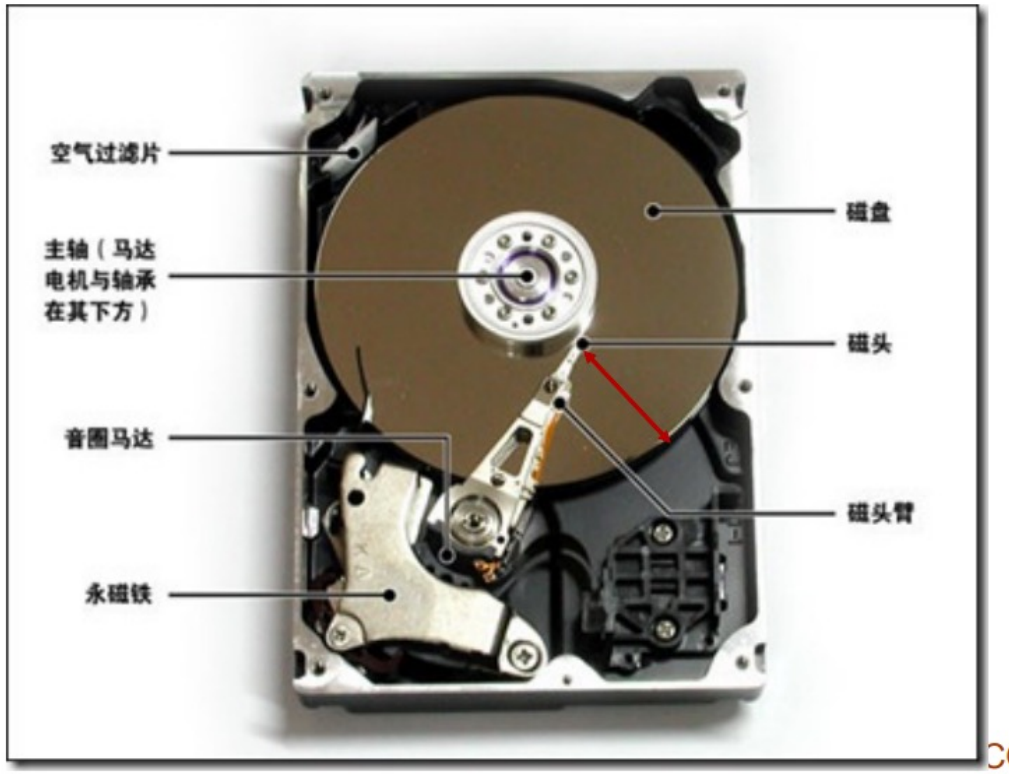
需要把“磁头”移动到想要读/写的扇区所在的磁道。磁盘会转起来，让目标扇区从磁头下面划过，才能完成对扇区的读/写操作。



如何在磁盘中读/写数据

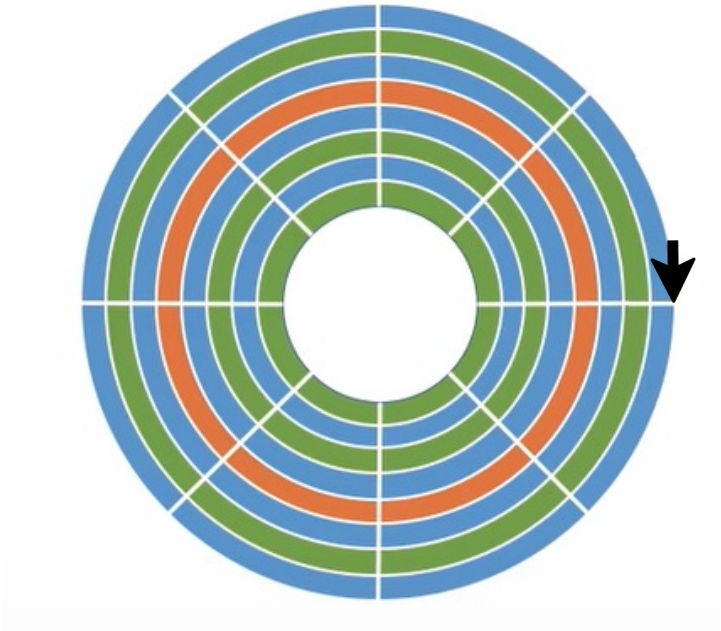


需要把“磁头”移动到想要读/写的扇区所在的磁道。磁盘会转起来，让目标扇区从磁头下面划过，才能完成对扇区的读/写操作。

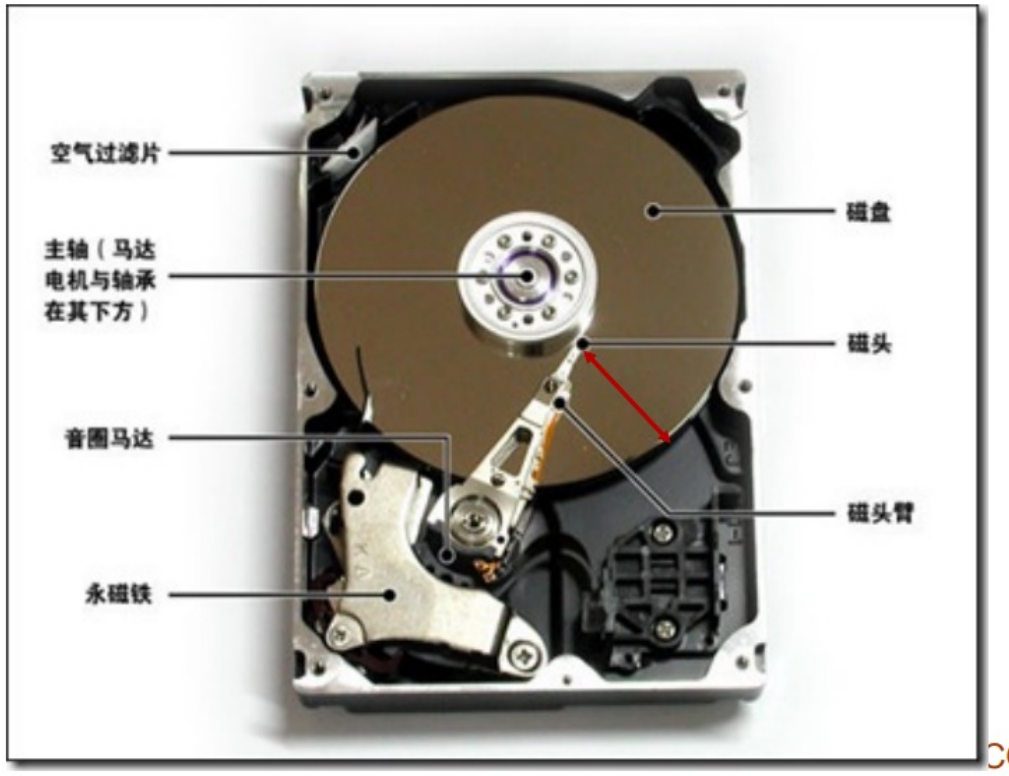


COM

如何在磁盘中读/写数据

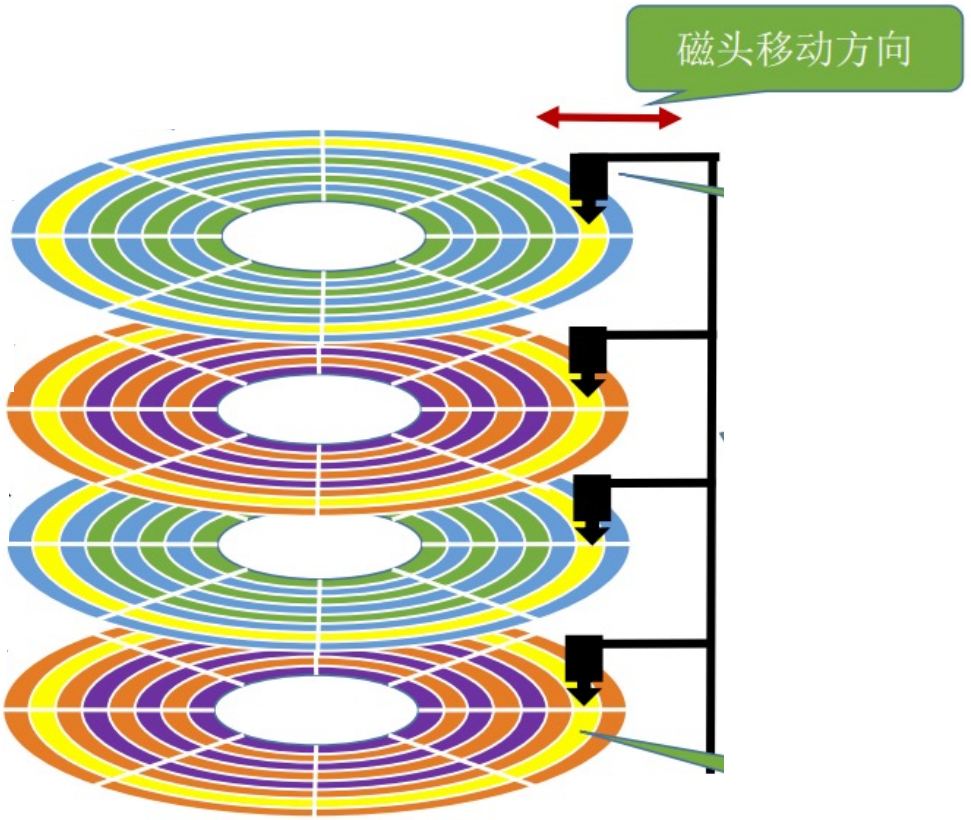


需要把“磁头”移动到想要读/写的扇区所在的磁道。磁盘会转起来，让目标扇区从磁头下面划过，才能完成对扇区的读/写操作。

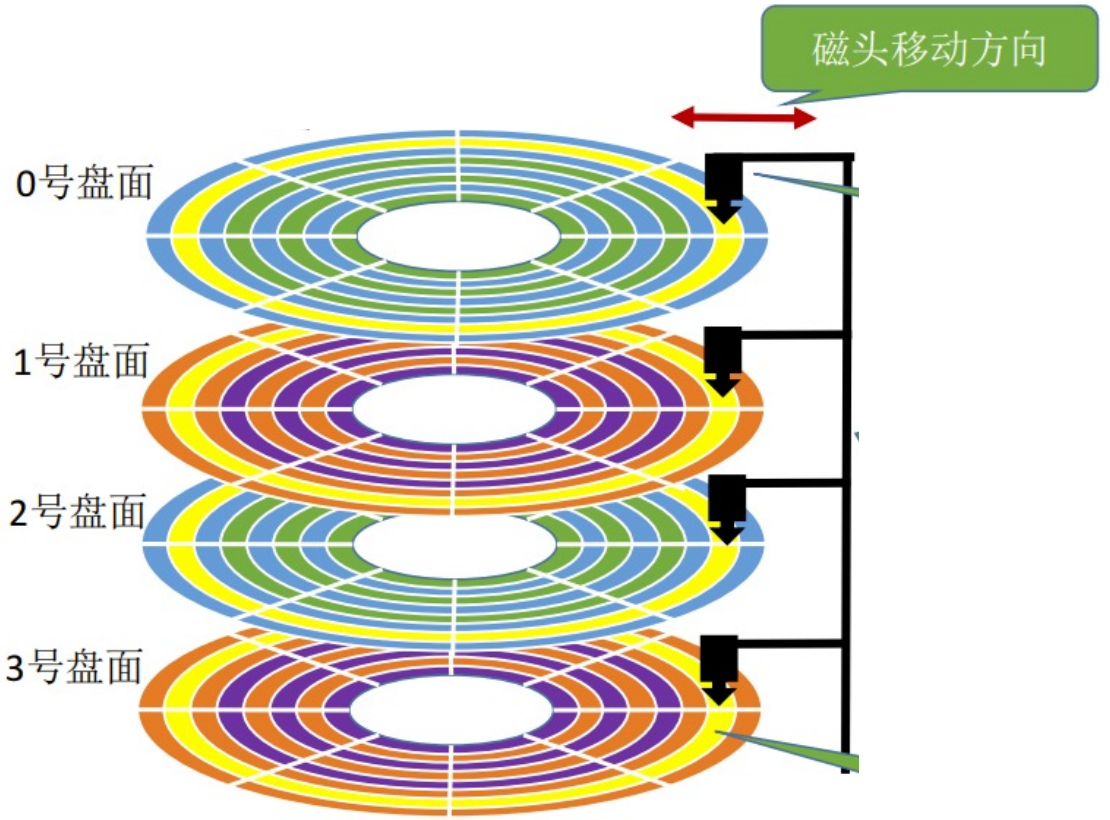


磁盘磁道的编号是(从外向内)依次由小到大进行编号的。

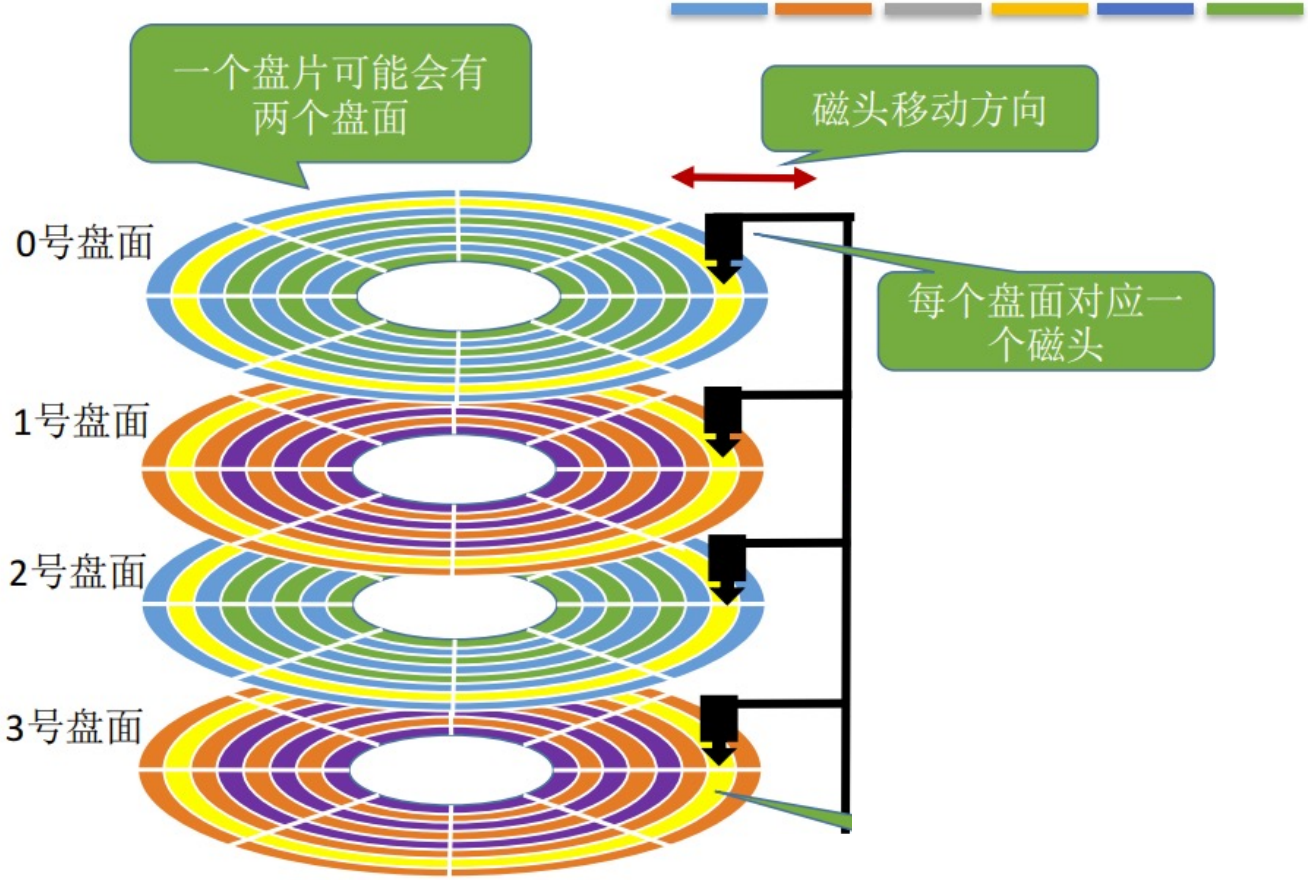
磁盘的物理地址



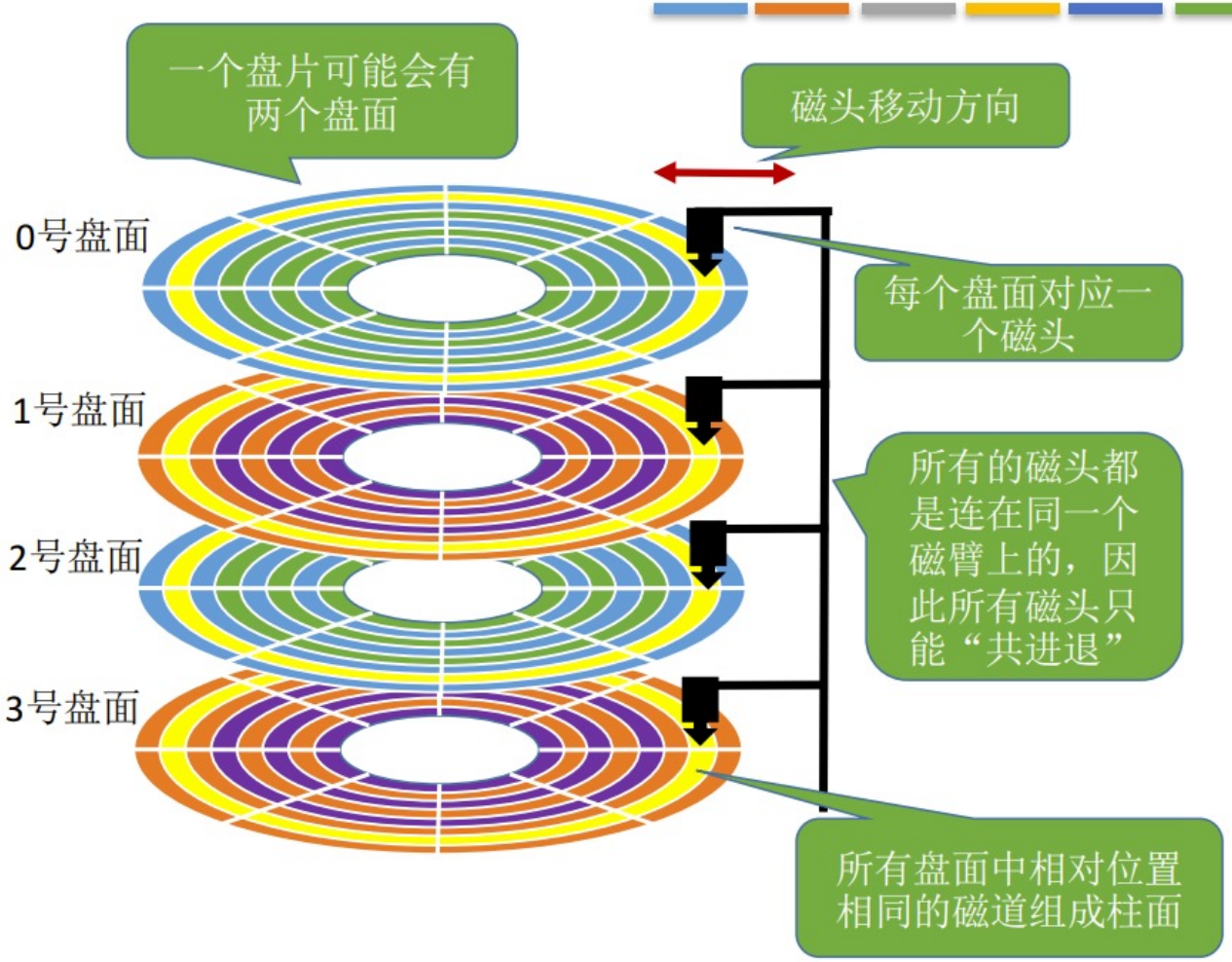
磁盘的物理地址



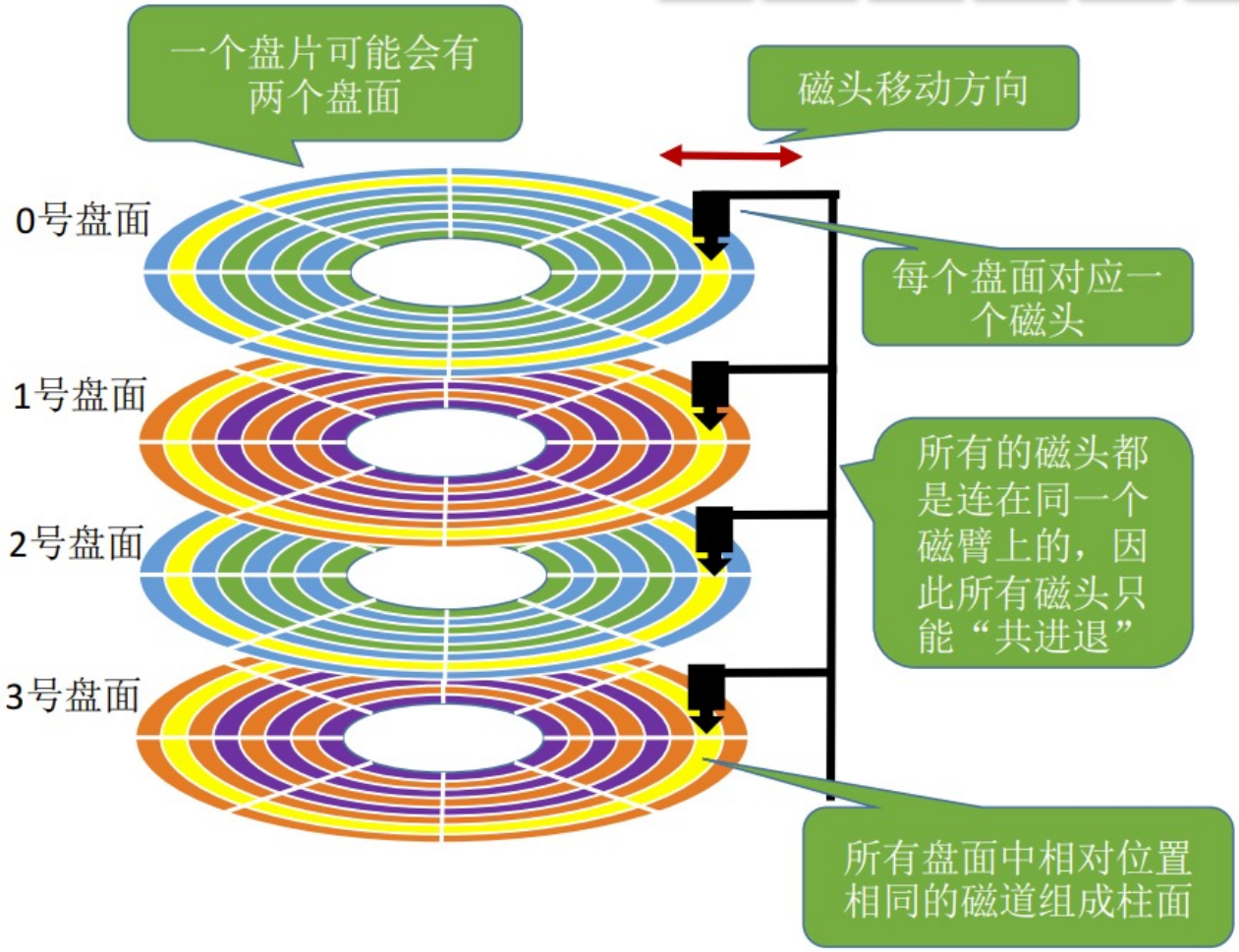
磁盘的物理地址



磁盘的物理地址

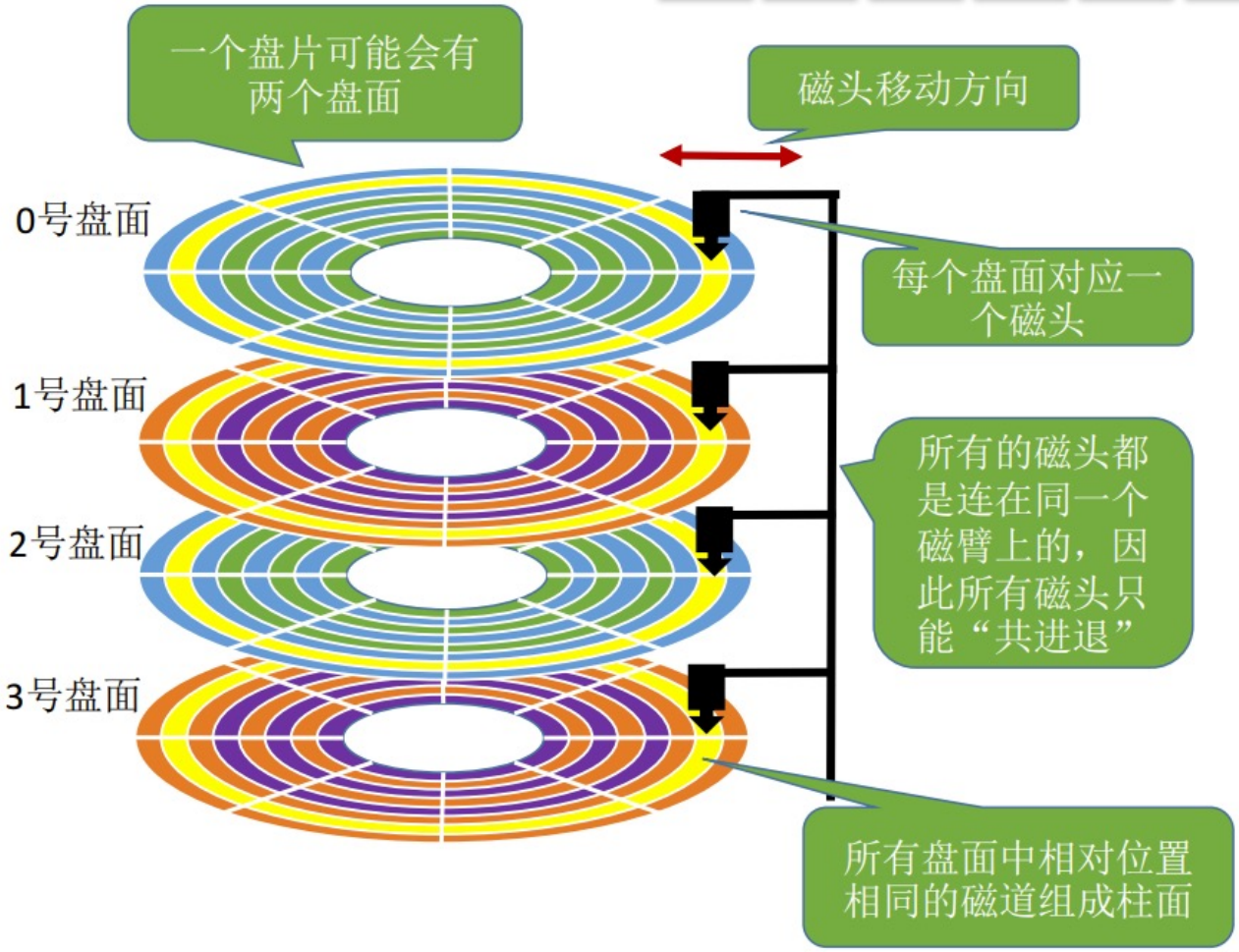

一个盘片可能会有两个盘面
磁头移动方向
0号盘面
1号盘面
2号盘面
3号盘面
每个盘面对应一个磁头
所有的磁头都是连在同一个磁臂上的，因此所有磁头只能“共进退”
所有盘面中相对位置相同的磁道组成柱面

磁盘的物理地址



可用（柱面号，盘面号，扇区号）来定位任意一个“磁盘块”。在“文件的物理结构”小节中，我们经常提到文件数据存放在外存中的几号块，这个块号就可以转换成（柱面号，盘面号，扇区号）的地址形式。

磁盘的物理地址



可用（柱面号，盘面号，扇区号）来定位任意一个“磁盘块”。在“文件的物理结构”小节中，我们经常提到文件数据存放在外存中的几号块，这个块号就可以转换成（柱面号，盘面号，扇区号）的地址形式。

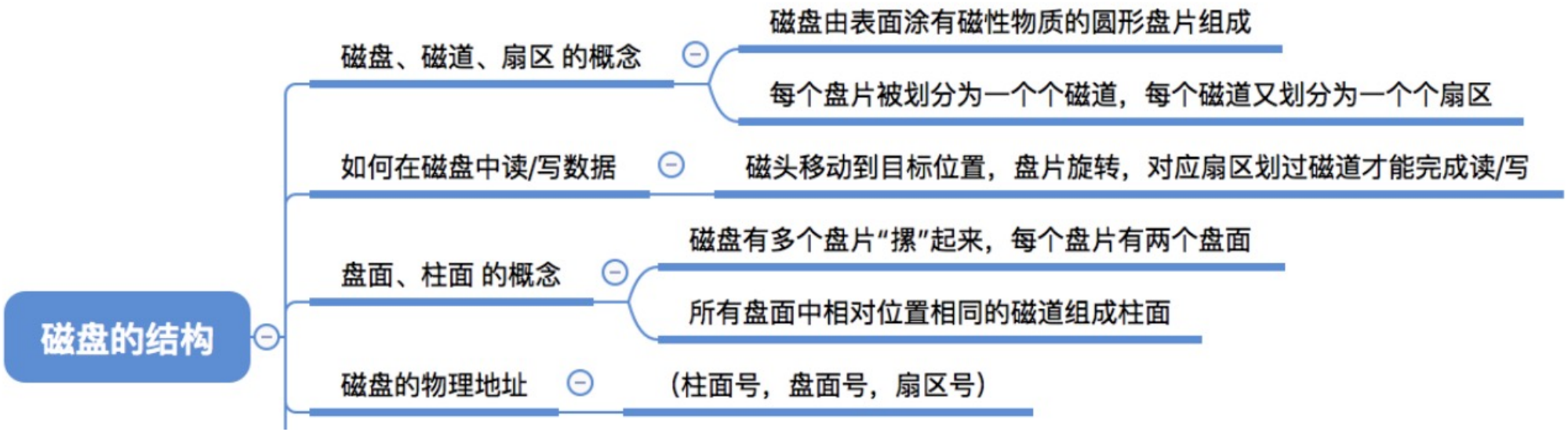
可根据该地址读取一个“块”

①根据“柱面号”移动磁臂，让磁头指向指定柱面；

②激活指定盘面对应的磁头；

③磁盘旋转的过程中，指定的扇区会从磁头下面划过，这样就完成了对指定扇区的读/写。

知识点回顾与重要考点

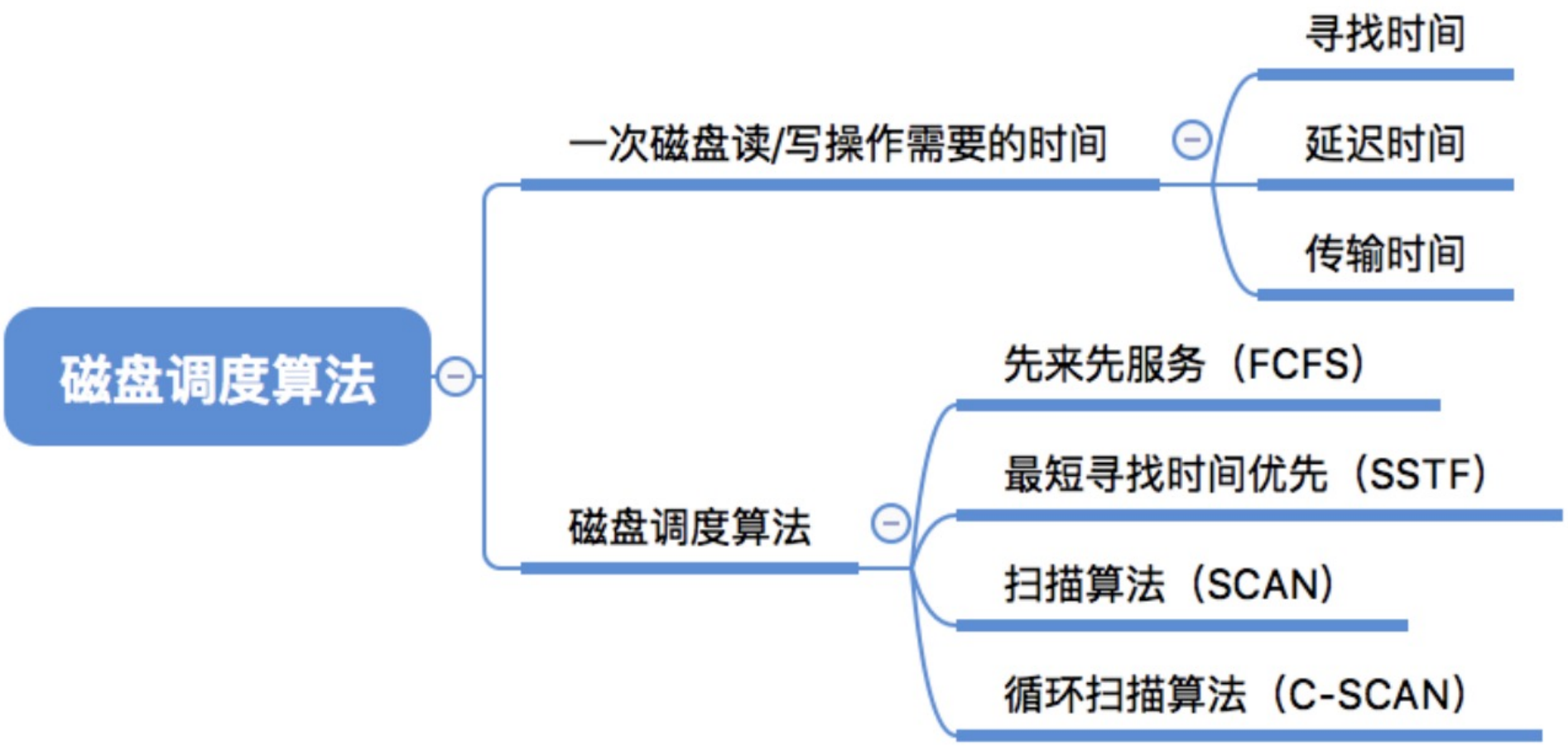

磁盘的结构
磁盘、磁道、扇区 的概念
磁盘由表面涂有磁性物质的圆形盘片组成
每个盘片被划分为一个个磁道，每个磁道又划分为一个个扇区
如何在磁盘中读/写数据
磁头移动到目标位置，盘片旋转，对应扇区划过磁道才能完成读/写
盘面、柱面 的概念
磁盘有多个盘片“摞”起来，每个盘片有两个盘面
所有盘面中相对位置相同的磁道组成柱面
磁盘的物理地址
（柱面号，盘面号，扇区号）

外存储设备管理

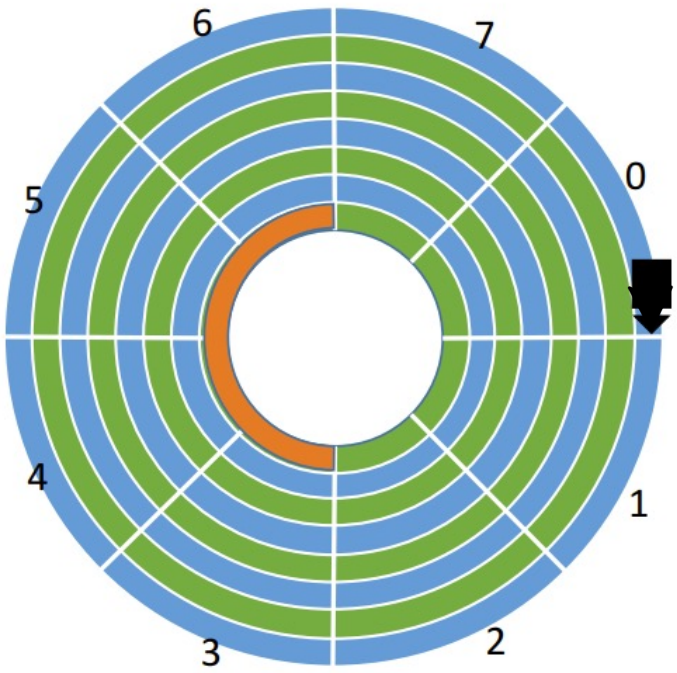
本讲内容

1. 典型外存储设备类型
2. 硬盘的存储空间管理
3. 硬盘的数据访问时间
4. 硬盘驱动臂调度算法

知识总览


磁盘调度算法
一次磁盘读/写操作需要的时间
寻找时间
延迟时间
传输时间
磁盘调度算法
先来先服务（FCFS）
最短寻找时间优先（SSTF）
扫描算法（SCAN）
循环扫描算法（C-SCAN）

一次磁盘读/写操作需要的时间



寻找时间（寻道时间）T_s：在读/写数据前，将磁头移动到指定磁道所花的时间。

①启动磁头臂是需要时间的。假设耗时为 s；

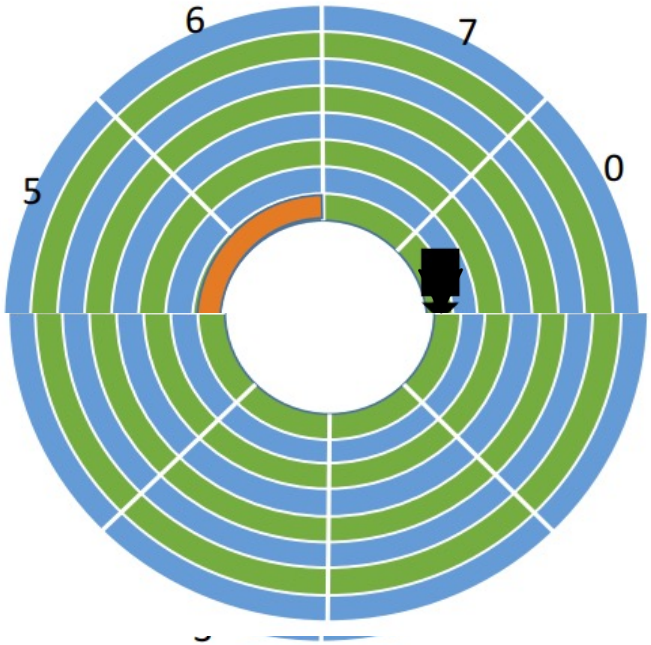
②移动磁头也是需要时间的。假设磁头匀速移动，每跨越一个磁道耗时为 m，总共需要跨越 n 条磁道。则：

寻道时间 $T_s = s + m*n$

现在的硬盘移动一个磁道大约需要0.2ms，磁臂启动时间约为2ms

m是一常数，与磁盘驱动器的速度有关，因此寻道时间将随寻道距离的增加而增大

一次磁盘读/写操作需要的时间



延迟时间T_R：通过旋转磁盘，使磁头定位到目标扇区所需要的时间。设磁盘转速为 r （单位：转/秒，或 转/分），则平均所需的延迟时间 $T_R = (1/2)*(1/r) = 1/2r$

1/r 就是转一圈需要的时间。找到目标扇区平均需要转半圈，因此再乘以 1/2

硬盘的典型转速为 5400 转/分，或 7200 转/分

转速越高，延迟时间越小，磁盘读写速度越快

硬盘的数据访问时间

2 旋转延迟时间

- 扇区移动到磁头下面所经历的时间
- 例子：硬盘

  旋转速度为 5400 r/min，

  每转需时11.1 ms = 60000 / 5400，

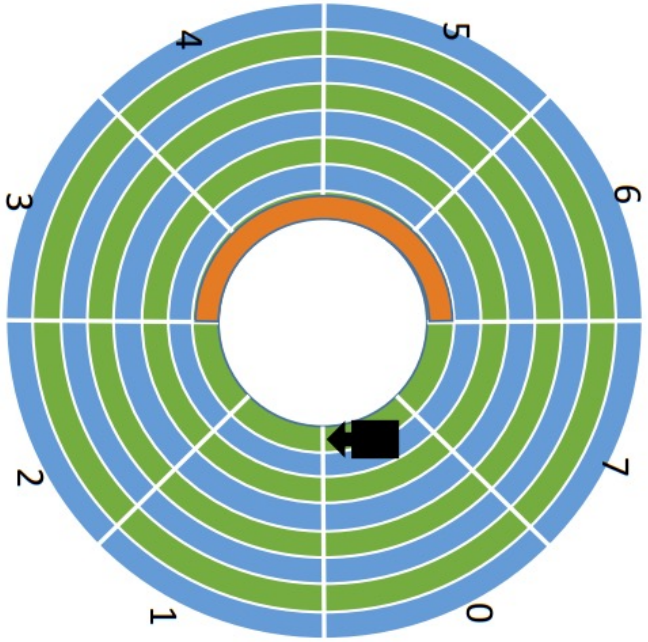
  平均旋转延迟时间为5.55 ms = 11.1 / 2

硬盘的数据访问时间

3 传输时间

- 把数据从磁盘读出或向磁盘写入的传输时间

一次磁盘读/写操作需要的时间

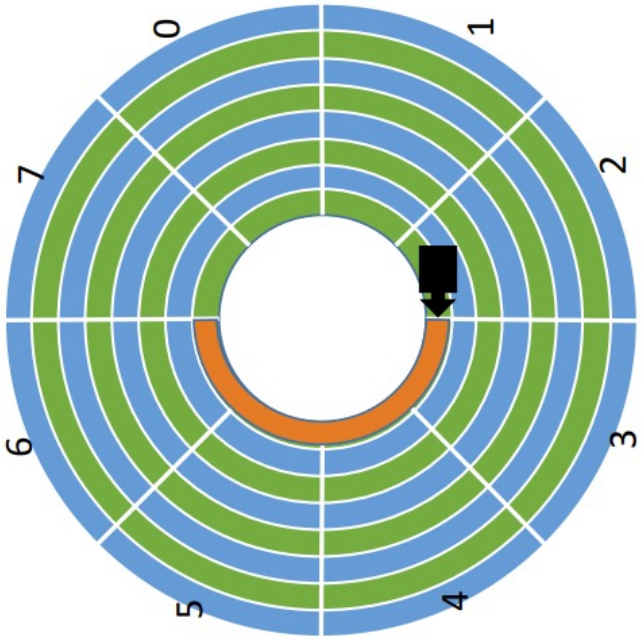


传输时间T_t：从磁盘读出或向磁盘写入数据所经历的时间，假设磁盘转速为 r，此次读/写的字节数为 b，每个磁道上的字节数为 N。则：

传输时间$T_t = (1/r) * (b/N) = b/(rN)$

每个磁道要可存 N 字节的数据，因此 b 字节的数据需要 b/N 个磁道才能存储。而读/写一个磁道所需的时间刚好又是转一圈所需要的时间 1/r

一次磁盘读/写操作需要的时间



寻找时间（寻道时间）T_S：在读/写数据前，将磁头移动到指定磁道所花的时间。

①启动磁头臂是需要时间的。假设耗时为 s；

②移动磁头也是需要时间的。假设磁头匀速移动，每跨越一个磁道耗时为 m，总共需要跨越 n 条磁道。则：

寻道时间 $T_S = s + m*n$

延迟时间T_R：通过旋转磁盘，使磁头定位到目标扇区所需要的时间。设磁盘转速为 r（单位：转/秒，或 转/分），则

平均所需的延迟时间 $T_R = (1/2)*(1/r) = 1/(2r)$

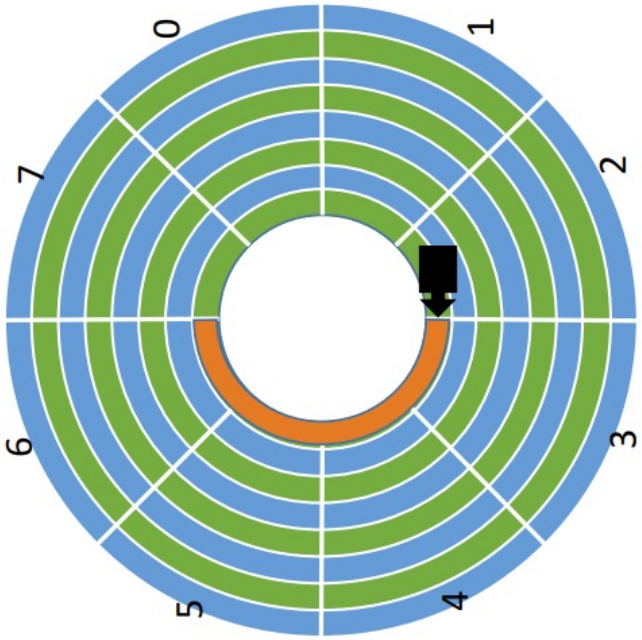
传输时间T_t：从磁盘读出或向磁盘写入数据所经历的时间，假设磁盘转速为 r，此次读/写的字节数为 b，每个磁道上的字节数为 N。则：

传输时间 $T_t = (1/r) * (b/N) = b/(rN)$

总的平均存取时间 $T_a = T_S + 1/2r + b/(rN)$

一次磁盘读/写操作需要的时间

但是操作系统的磁盘调度算法会直接影响寻道时间



寻找时间（寻道时间）T_S：在读/写数据前，将磁头移动到指定磁道所花的时间。

①启动磁头臂是需要时间的。假设耗时为 s；

②移动磁头也是需要时间的。假设磁头匀速移动，每跨越一个磁道耗时为 m，总共需要跨越 n 条磁道。则：

寻道时间 $T_S = s + m*n$

延迟时间T_R：通过旋转磁盘，使磁头定位到目标扇区所需要的时间。设磁盘转速为 r（单位：转/秒，或 转/分），则

平均所需的延迟时间 $T_R = (1/2)*(1/r) = 1/(2r)$

传输时间T_t：从磁盘读出或向磁盘写入数据所经历的时间，假设磁盘转速为 r，此次读/写的字节数为 b，每个磁道上的字节数为 N。则：

传输时间$T_t = (1/r) * (b/N) = b/(rN)$

总的平均存取时间 $T_a = T_S + 1/2r + b/(rN)$

延迟时间和传输时间都与磁盘转速相关，且为线性相关。而转速是硬件的固有属性，因此操作系统也无法优化延迟时间和传输时间

外存储设备管理

本讲内容

1. 典型外存储设备类型
2. 硬盘的存储空间管理
3. 硬盘的数据访问时间
4. 硬盘驱动臂调度算法

硬盘驱动臂调度算法

- 多个进程并发访问硬盘时，应采用一种最佳的驱动臂调度算法，以使各进程对硬盘的平均访问时间最少
- 硬盘访问的时间中，寻道时间比重最大，磁盘调度的目标，是使磁盘的平均寻道时间最少

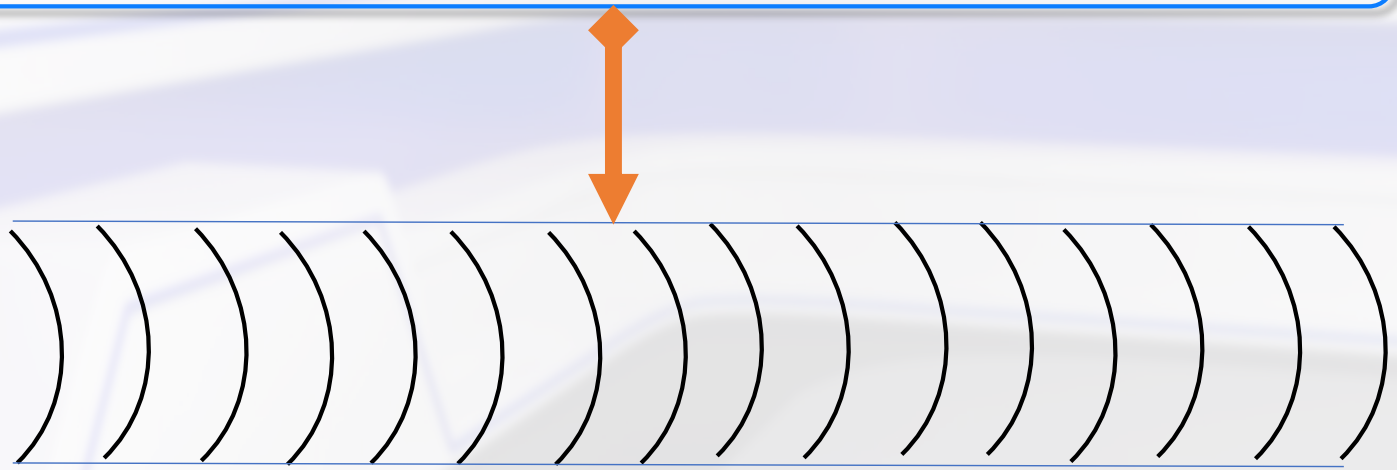
硬盘驱动臂调度算法

- 先来先服务算法
- 最短查找时间优先算法
- 电梯调度算法
- 扫描算法
- 单向扫描算法
- 分步扫描算法

硬盘驱动臂调度算法

硬盘有200个柱面，编号0~199，当前磁头悬停的位置在143号柱面上，并刚刚完成了125号柱面的服务请求，如果请求队列的先后顺序是：

86，147，91，177，94，150，102，175，130。



199… 177.175.. 147.143…130……94..91…86……..0

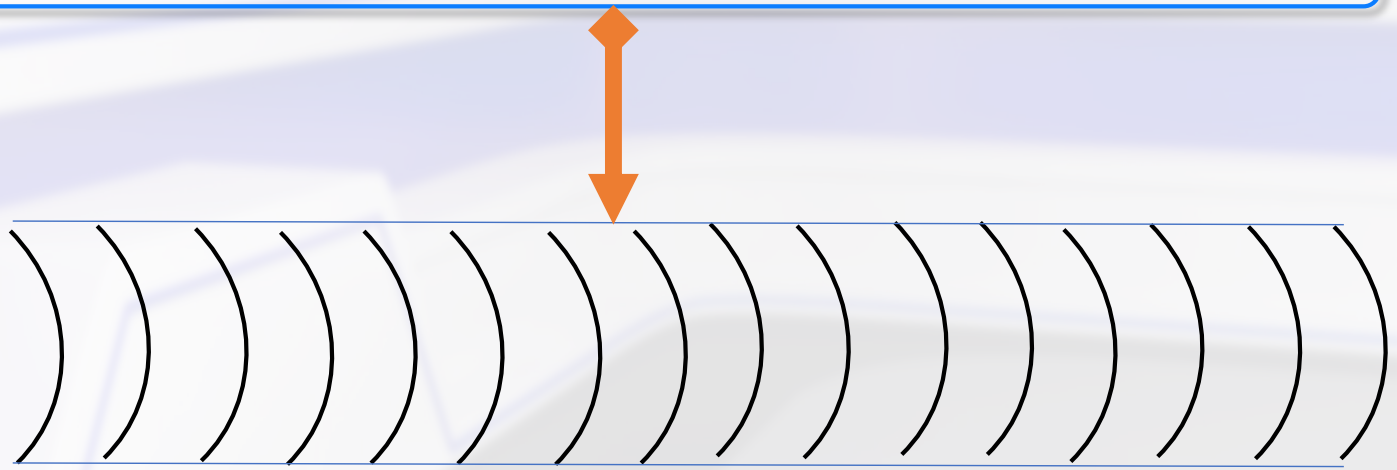
硬盘驱动臂调度算法

1 先来先服务算法

请求队列的先后顺序是：86，147，91，177，94，150，102，175，130。

磁头移动路径为：

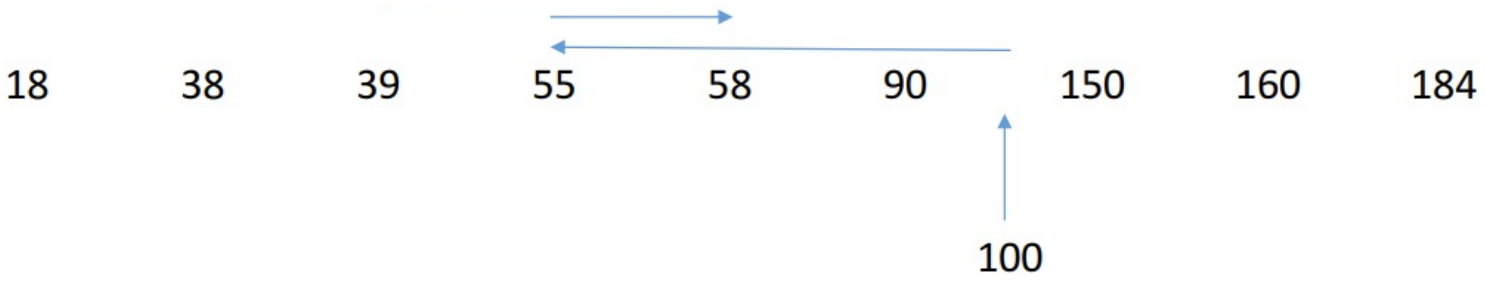
143-86-147-91-177-94-150-102-175-130



先来先服务算法（FCFS）

根据进程请求访问磁盘的先后顺序进行调度。

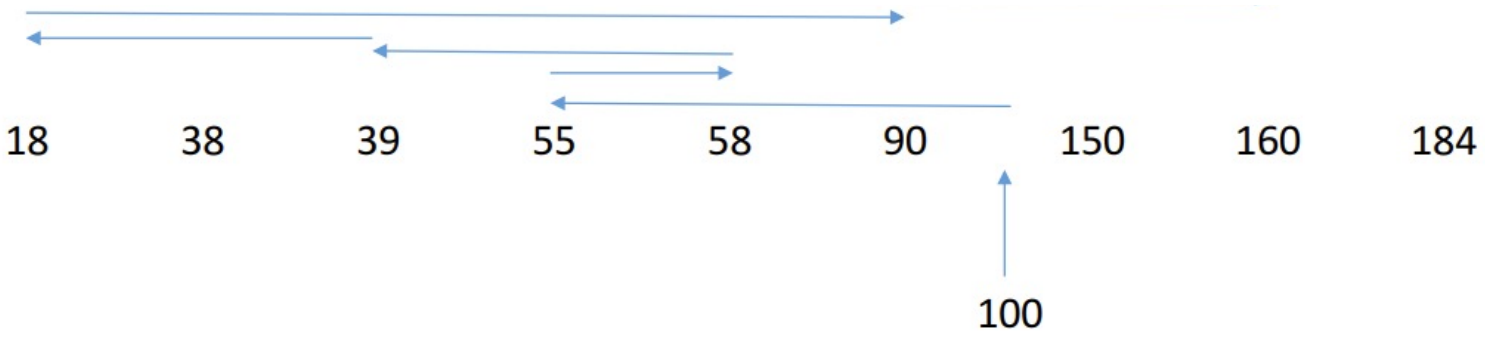
假设磁头的初始位置是100号磁道，有多个进程先后陆续地请求访问 55、58、39、18、90、160、150、38、184 号磁道
按照 FCFS 的规则，按照请求到达的顺序，磁头需要依次移动到 55、58、39、18、90、160、150、38、184 号磁道



先来先服务算法（FCFS）

根据进程请求访问磁盘的先后顺序进行调度。

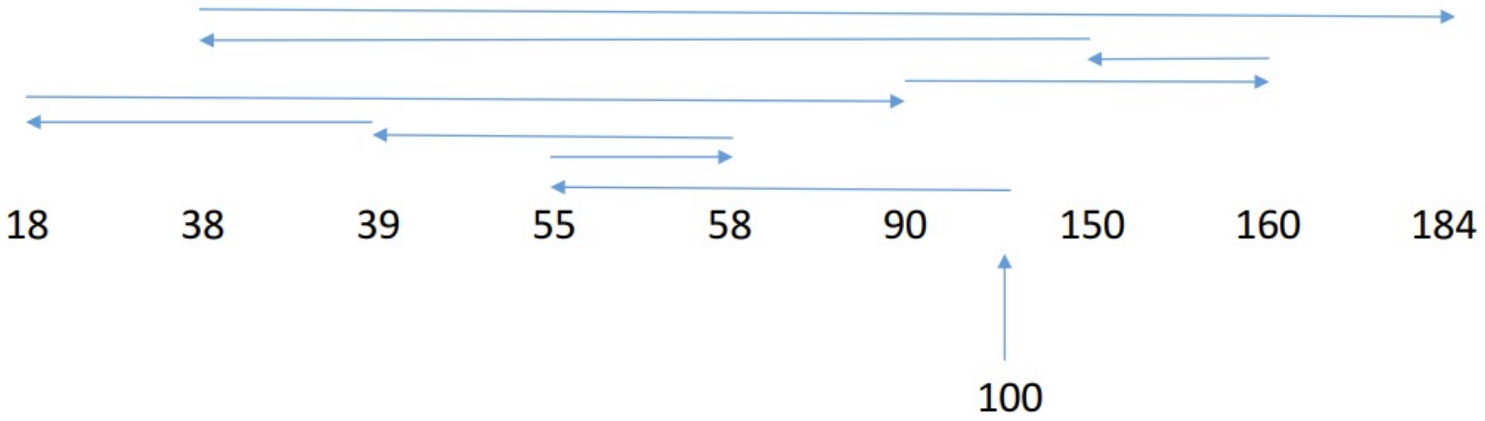
假设磁头的初始位置是100号磁道，有多个进程先后陆续地请求访问 55、58、39、18、90、160、150、38、184 号磁道
按照 FCFS 的规则，按照请求到达的顺序，磁头需要依次移动到 55、58、39、18、90、160、150、38、184 号磁道



先来先服务算法（FCFS）

根据进程请求访问磁盘的先后顺序进行调度。

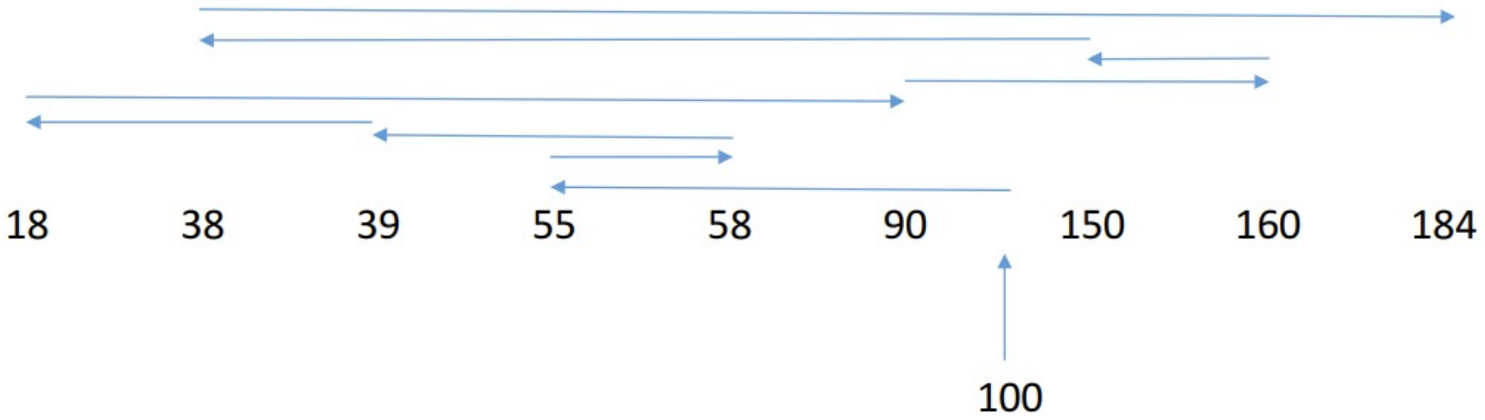
假设磁头的初始位置是100号磁道，有多个进程先后陆续地请求访问 55、58、39、18、90、160、150、38、184 号磁道
按照 FCFS 的规则，按照请求到达的顺序，磁头需要依次移动到 55、58、39、18、90、160、150、38、184 号磁道



先来先服务算法（FCFS）

根据进程请求访问磁盘的先后顺序进行调度。

假设磁头的初始位置是100号磁道，有多个进程先后陆续地请求访问 55、58、39、18、90、160、150、38、184 号磁道
按照 FCFS 的规则，按照请求到达的顺序，磁头需要依次移动到 55、58、39、18、90、160、150、38、184 号磁道

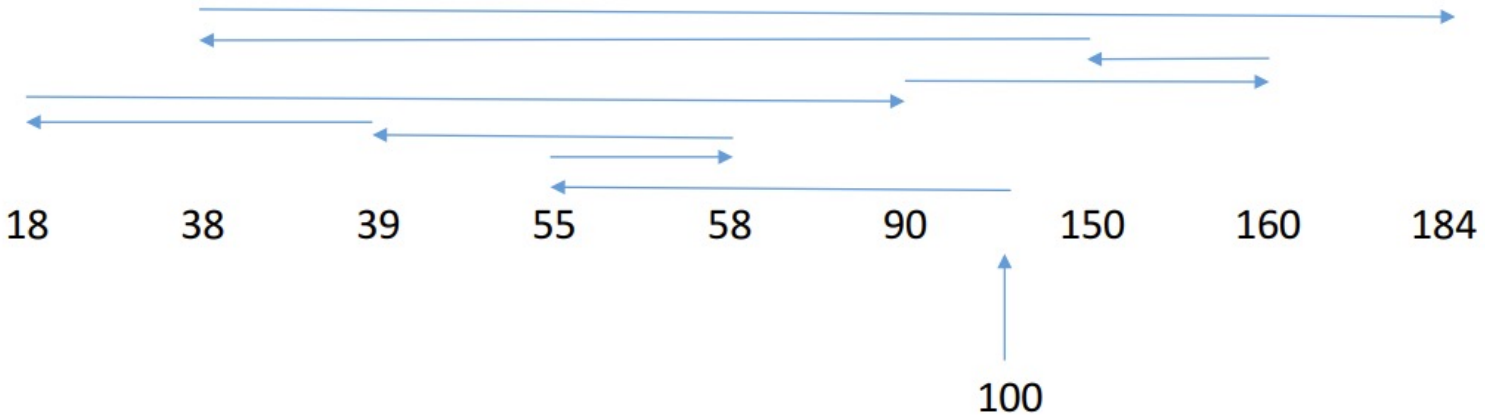


磁头总共移动了 45+3+19+21+72+70+10+112+146 = 498 个磁道
响应一个请求平均需要移动 498/9 = 55.3 个磁道（平均寻找长度）

先来先服务算法（FCFS）

根据进程请求访问磁盘的先后顺序进行调度。

假设磁头的初始位置是100号磁道，有多个进程先后陆续地请求访问 55、58、39、18、90、160、150、38、184 号磁道
按照 FCFS 的规则，按照请求到达的顺序，磁头需要依次移动到 55、58、39、18、90、160、150、38、184 号磁道

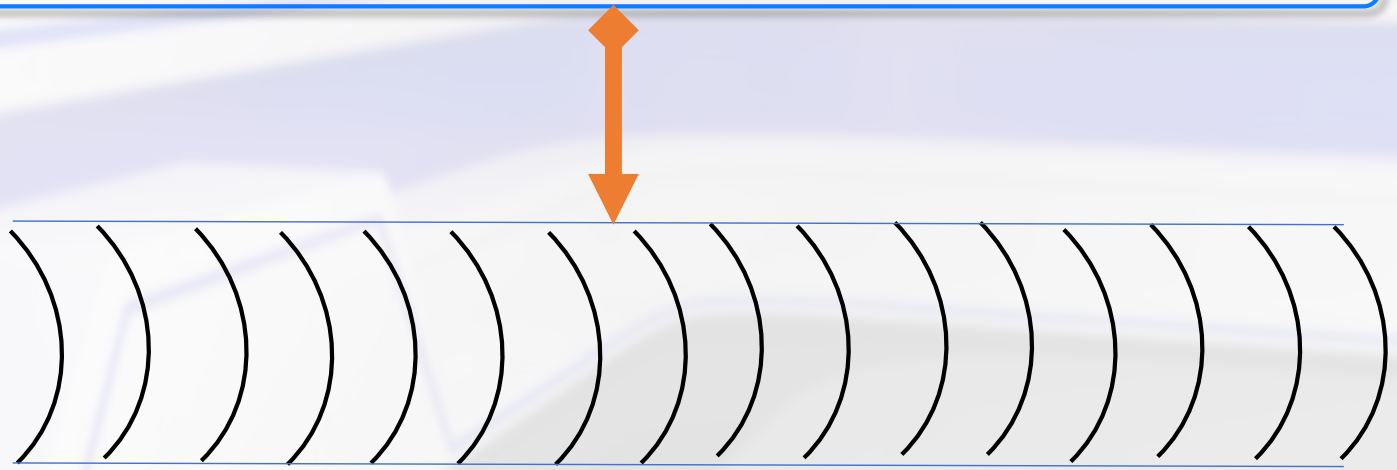


磁头总共移动了 45+3+19+21+72+70+10+112+146 = 498 个磁道
响应一个请求平均需要移动 498/9 = 55.3 个磁道（平均寻找长度）
优点：公平；如果请求访问的磁道比较集中的话，算法性能还算过的去
缺点：如果有大量进程竞争使用磁盘，请求访问的磁道很分散，则FCFS在性能上很差，寻道时间长。

硬盘驱动臂调度算法

2 最短查找时间优先算法

总是先执行查找时间最短的硬盘请求，较先来先服务算法有更好的性能，但会出现饥饿现象，距离远的读写请求可能被长期推迟



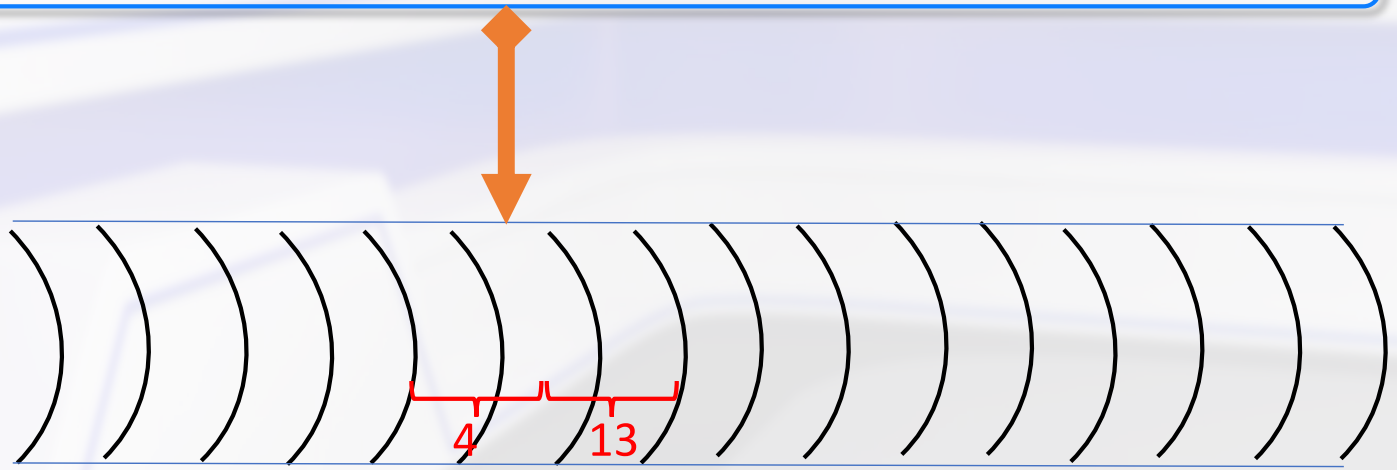
硬盘驱动臂调度算法

2 最短查找时间优先算法

总是先执行查找时间最短的硬盘请求，当前磁头悬停的位置在143号柱面上

磁头移动路径为：

143-147-150-130-102-94-91-86-175-177



199… 177.175.. 147.143…130……94..91…86……..0

硬盘驱动臂调度算法

2 最短查找时间优先算法

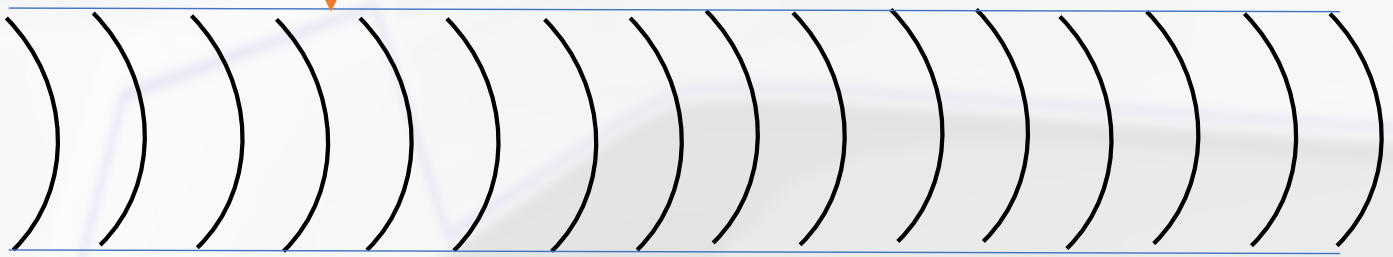
总是先执行查找时间最短的硬盘请求

磁头移动路径为：

25

143-147-150-130-102-94-91-86-175-177

20



199… 177.175.. 147.143…130……94..91…86……..0

硬盘驱动臂调度算法

2 最短查找时间优先算法

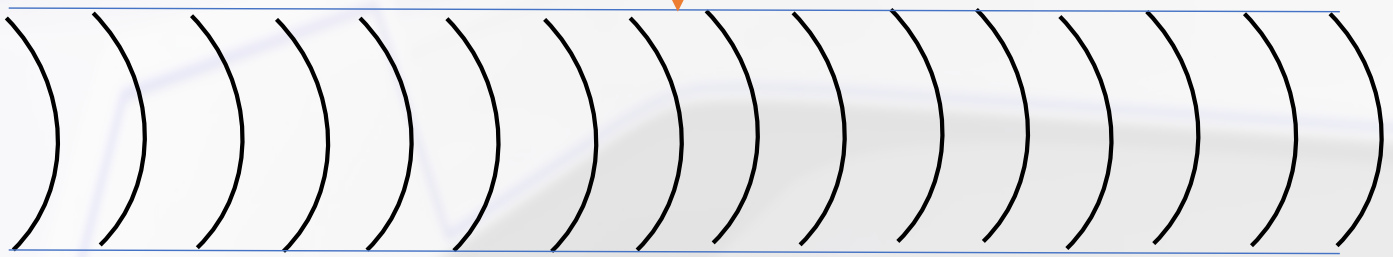
总是先执行查找时间最短的硬盘请求

磁头移动路径为：

45

143-147-150-130-102-94-91-86-175-177

28

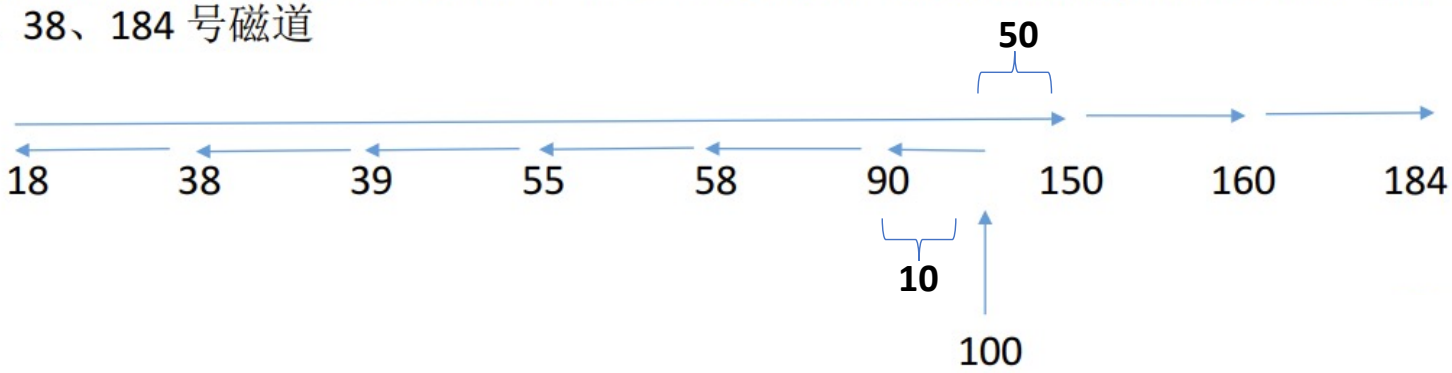


199… 177.175.. 147.143…130……94..91…86……..0

最短寻找时间优先（SSTF）

SSTF 算法会优先处理的磁道是与当前磁头最近的磁道。可以保证每次的寻道时间最短，但是并不能保证总的寻道时间最短。（其实就是贪心算法的思想，只是选择眼前最优，但是总体未必最优）

假设磁头的初始位置是100号磁道，有多个进程先后陆续地请求访问 55、58、39、18、90、160、150、38、184 号磁道

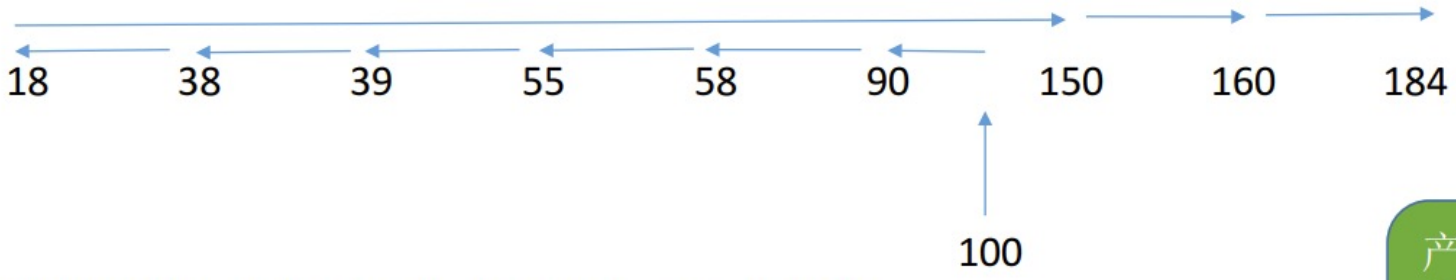


磁头总共移动了 (100-18) + (184-18) = 248 个磁道
响应一个请求平均需要移动 248/9 = 27.5 个磁道（平均寻找长度）

最短寻找时间优先（SSTF）

SSTF 算法会优先处理的磁道是与当前磁头最近的磁道。可以保证每次的寻道时间最短，但是并不能保证总的寻道时间最短。（其实就是贪心算法的思想，只是选择眼前最优，但是总体未必最优）

假设磁头的初始位置是100号磁道，有多个进程先后陆续地请求访问 55、58、39、18、90、160、150、38、184 号磁道



磁头总共移动了 (100-18) + (184-18) = 248 个磁道
响应一个请求平均需要移动 248/9 = 27.5 个磁道（平均寻找长度）
优点：性能较好，平均寻道时间短
缺点：可能产生“饥饿”现象

产生饥饿的原因在于：磁头在一个小区域内来回来去地移动

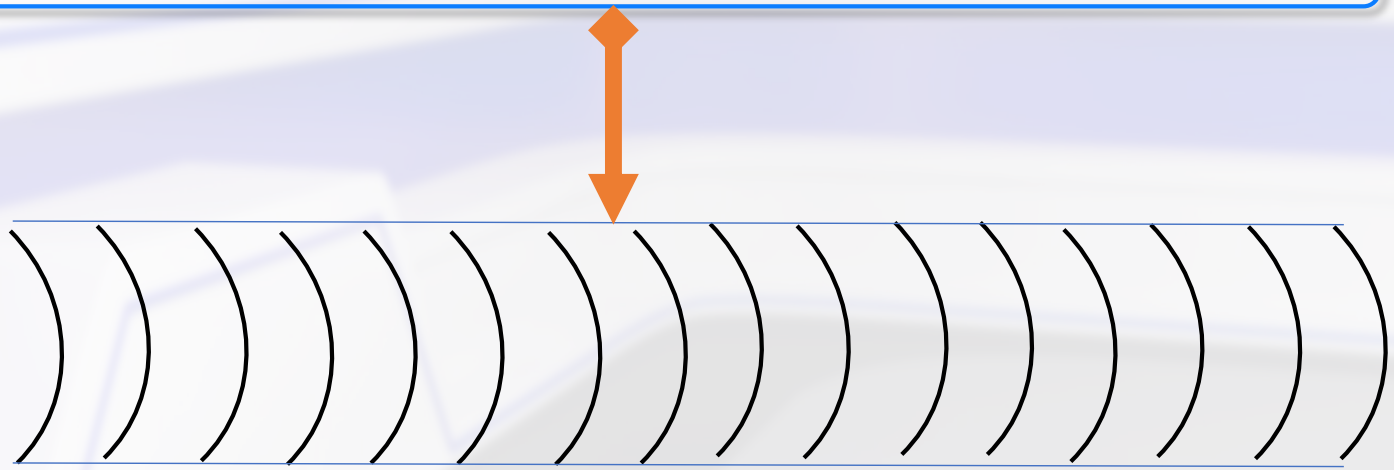
Eg：本例中，如果在处理18号磁道的访问请求时又来了一个38号磁道的访问请求，处理38号磁道的访问请求时又来了一个18号磁道的访问请求。如果有源源不断的 18号、38号磁道的访问请求到来的话，150、160、184 号磁道的访问请求就永远得不到满足，从而产生“饥饿”现象。

硬盘驱动臂调度算法

4 扫描算法

钟摆，来回摆动

臂沿一个方向移动，扫过所有柱面，遇到硬盘请求便进行处理，直到最后一个柱面后，再向相反方向移动，也会出现饥饿现象



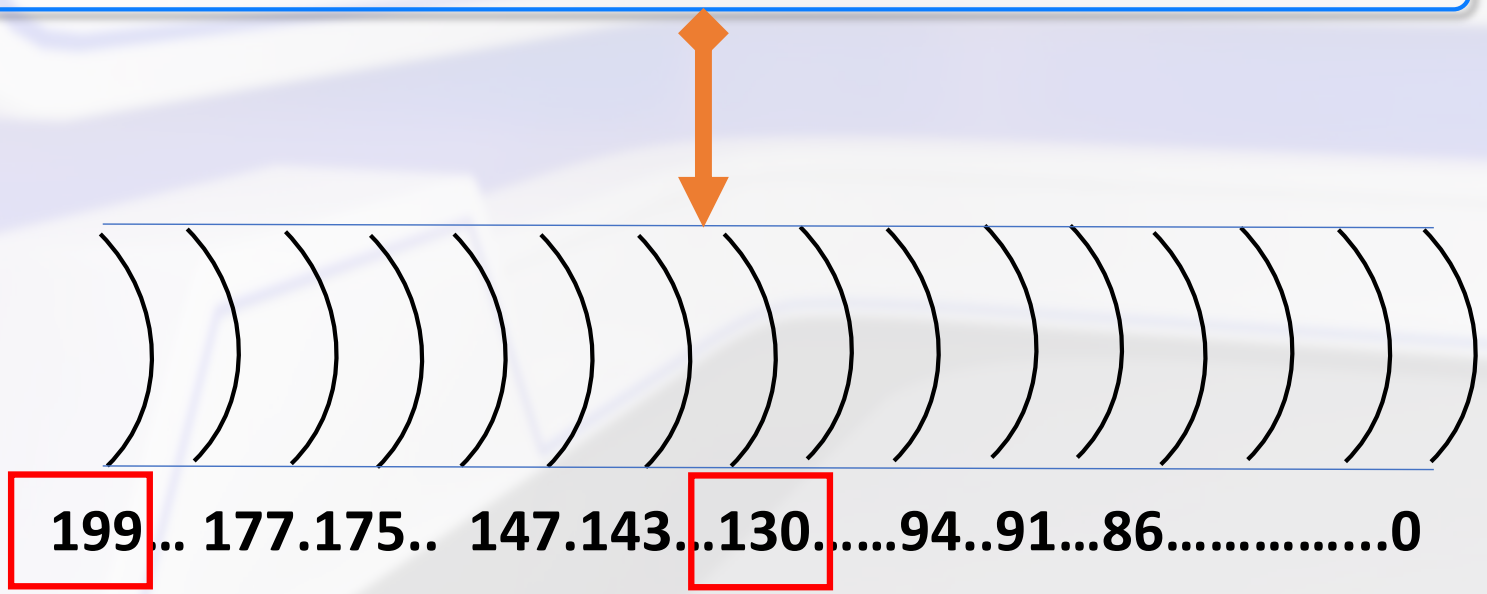
199… 177.175.. 147.143…130……94..91…86……0

硬盘驱动臂调度算法

4 扫描算法

磁头移动路径为：

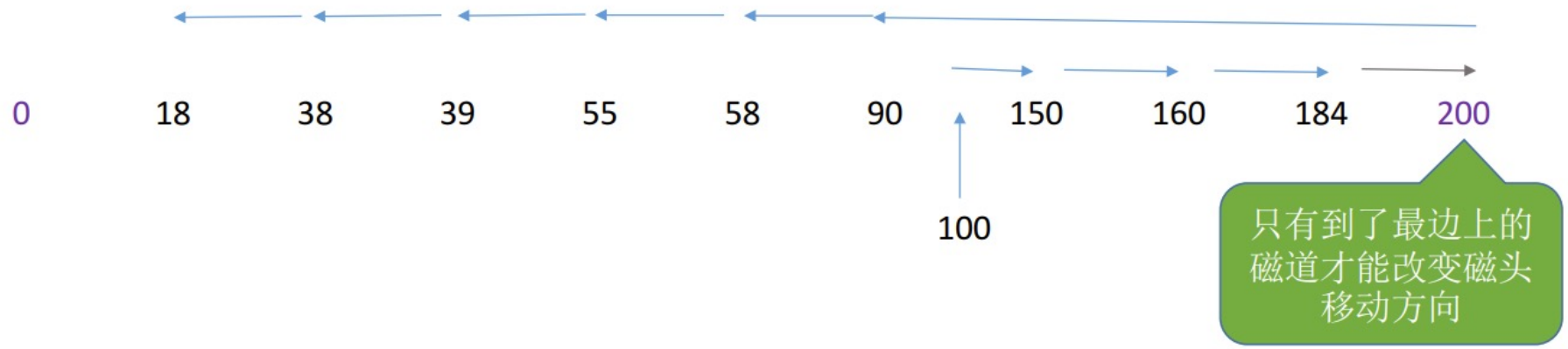
143-147-150-175-177-199-130-102-94-91-86



扫描算法（SCAN）

SSTF 算法会产生饥饿的原因在于：磁头有可能在一个小区域内来回来去地移动。为了防止这个问题，可以规定，只有磁头移动到最外侧磁道的时候才能往内移动，移动到最内侧磁道的时候才能往外移动。这就是扫描算法（SCAN）的思想。

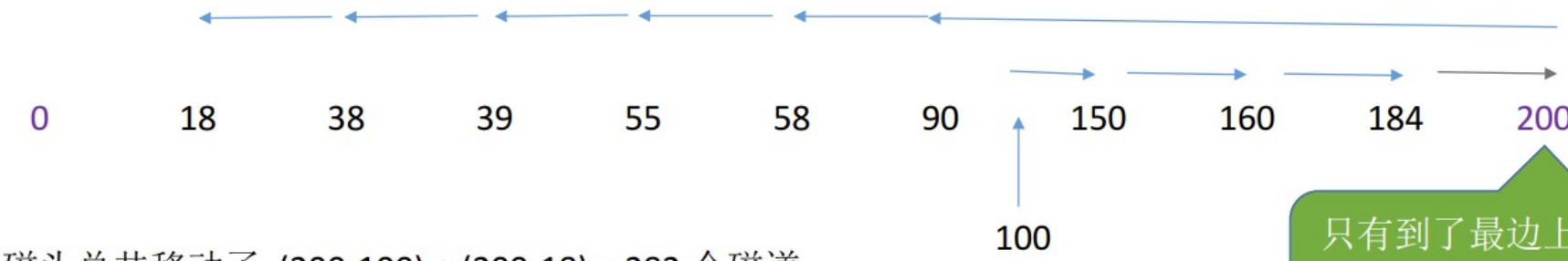
假设某磁盘的磁道为 0~200号，磁头的初始位置是100号磁道，且此时磁头正在往磁道号增大的方向移动，有多个进程先后陆续地请求访问 55、58、39、18、90、160、150、38、184 号磁道



扫描算法（SCAN）

SSTF 算法会产生饥饿的原因在于：磁头有可能在一个小区域内来回来去地移动。为了防止这个问题，可以规定，只有磁头移动到最外侧磁道的时候才能往内移动，移动到最内侧磁道的时候才能往外移动。这就是扫描算法（SCAN）的思想。

假设某磁盘的磁道为 0~200号，磁头的初始位置是100号磁道，且此时磁头正在往磁道号增大的方向移动，有多个进程先后陆续地请求访问 55、58、39、18、90、160、150、38、184 号磁道

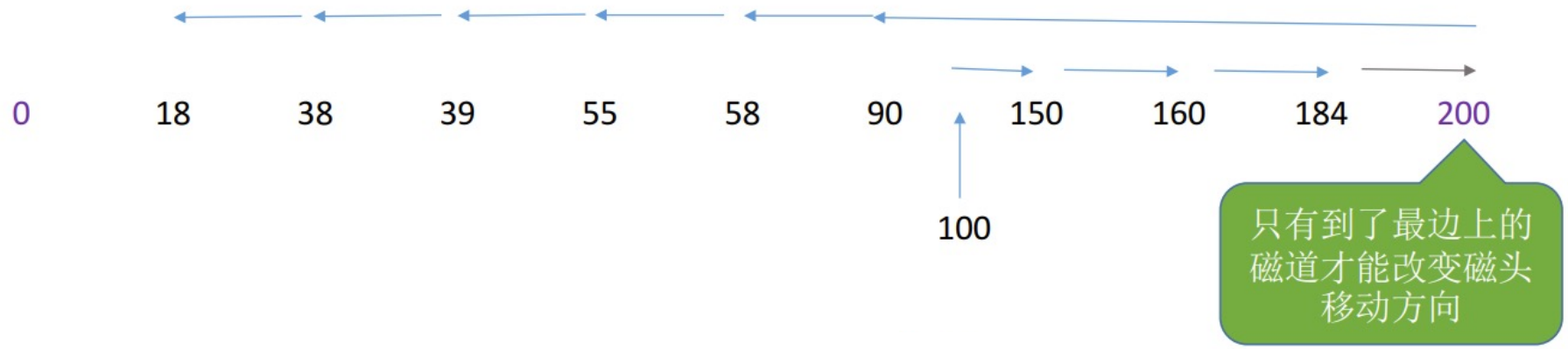


磁头总共移动了 (200-100) + (200-18) = 282 个磁道
响应一个请求平均需要移动 282/9 = 31.3 个磁道（平均寻找长度）
优点：性能较好，平均寻道时间较短

扫描算法（SCAN）

SSTF 算法会产生饥饿的原因在于：磁头有可能在一个小区域内来回来去地移动。为了防止这个问题，可以规定，只有磁头移动到最外侧磁道的时候才能往内移动，移动到最内侧磁道的时候才能往外移动。这就是扫描算法（SCAN）的思想。

假设某磁盘的磁道为 0~200号，磁头的初始位置是100号磁道，且此时磁头正在往磁道号增大的方向移动，有多个进程先后陆续地请求访问 55、58、39、18、90、160、150、38、184 号磁道



缺点：①只有到达最边上的磁道时才能改变磁头移动方向，事实上，处理了184号磁道的访问请求之后就不需要再往右移动磁头了。

②SCAN算法对于各个位置磁道的响应频率不平均（如：假设此时磁头正在往右移动，且刚处理过90号磁道，那么下次处理90号磁道的请求就需要等磁头移动很长一段距离；而响应了184号磁道的请求之后，很快又可以再次响应 184 号磁道的请求了）

LOOK 调度算法

“电梯调度”算法

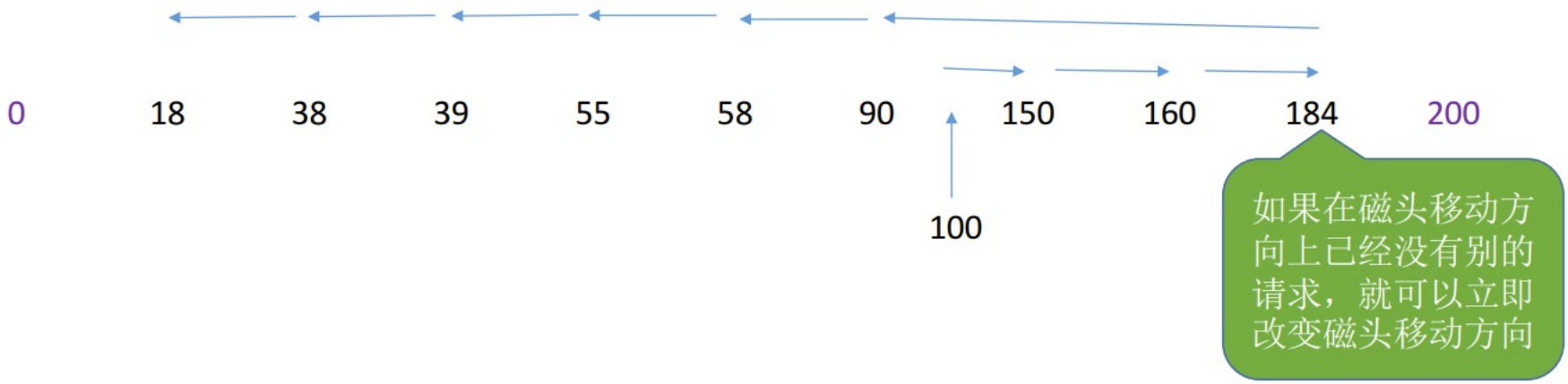
扫描算法（SCAN）中，只有到达最边上的磁道时才能改变磁头移动方向，事实上，处理了184号磁道的访问请求之后就不需要再往右移动磁头了。LOOK 调度算法就是为了解决这个问题，如果在磁头移动方向上已经没有别的请求，就可以立即改变磁头移动方向。（边移动边观察，因此叫 LOOK）

LOOK 调度算法

“电梯调度”算法

扫描算法（SCAN）中，只有到达最边上的磁道时才能改变磁头移动方向，事实上，处理了184号磁道的访问请求之后就不需要再往右移动磁头了。LOOK 调度算法就是为了解决这个问题，如果在磁头移动方向上已经没有别的请求，就可以立即改变磁头移动方向。（边移动边观察，因此叫 LOOK）

假设某磁盘的磁道为 0~200号，磁头的初始位置是100号磁道，且此时磁头正在往磁道号增大的方向移动，有多个进程先后陆续地请求访问 55、58、39、18、90、160、150、38、184 号磁道

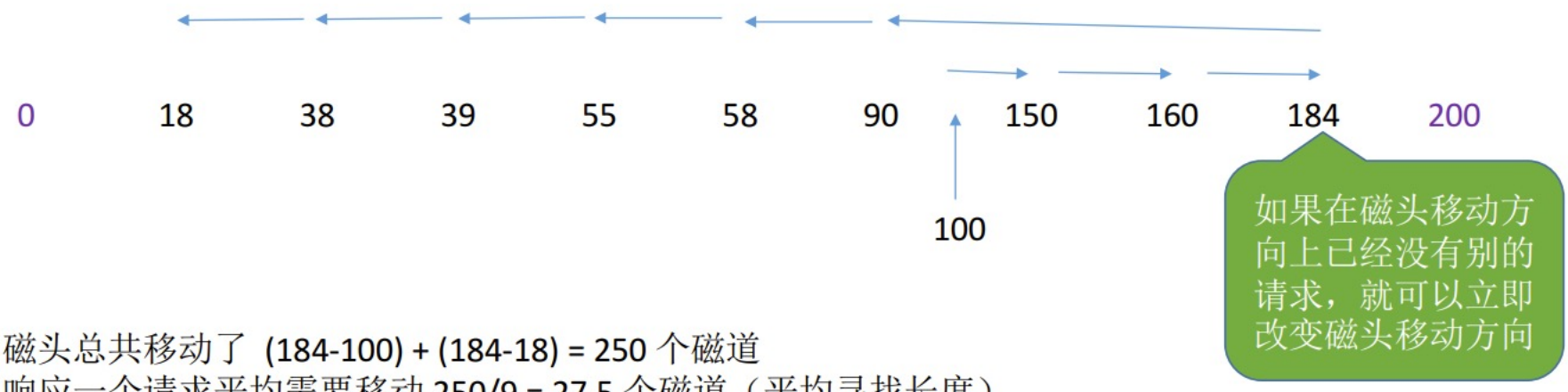


LOOK 调度算法

“电梯调度”算法

扫描算法（SCAN）中，只有到达最边上的磁道时才能改变磁头移动方向，事实上，处理了184号磁道的访问请求之后就不需要再往右移动磁头了。LOOK 调度算法就是为了解决这个问题，如果在磁头移动方向上已经没有别的请求，就可以立即改变磁头移动方向。（边移动边观察，因此叫 LOOK）

假设某磁盘的磁道为 0~200号，磁头的初始位置是100号磁道，且此时磁头正在往磁道号增大的方向移动，有多个进程先后陆续地请求访问 55、58、39、18、90、160、150、38、184 号磁道

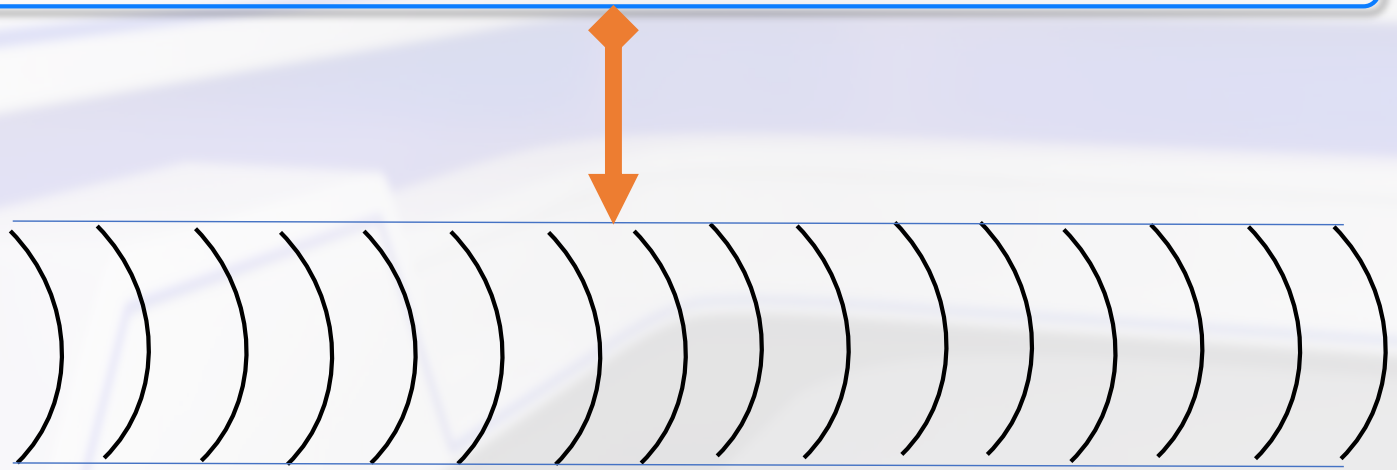


磁头总共移动了 (184-100) + (184-18) = 250 个磁道
响应一个请求平均需要移动 250/9 = 27.5 个磁道（平均寻找长度）
优点：比起 SCAN 算法来，不需要每次都移动到最外侧或最内侧才改变磁头方向，使寻道时间进一步缩短

硬盘驱动臂调度算法

3 “电梯调度”算法

选择沿臂的移动方向最近的柱面，如果方向上没有访问请求时，就改变臂移动方向，使移动频率极小化，也会出现饥饿现象



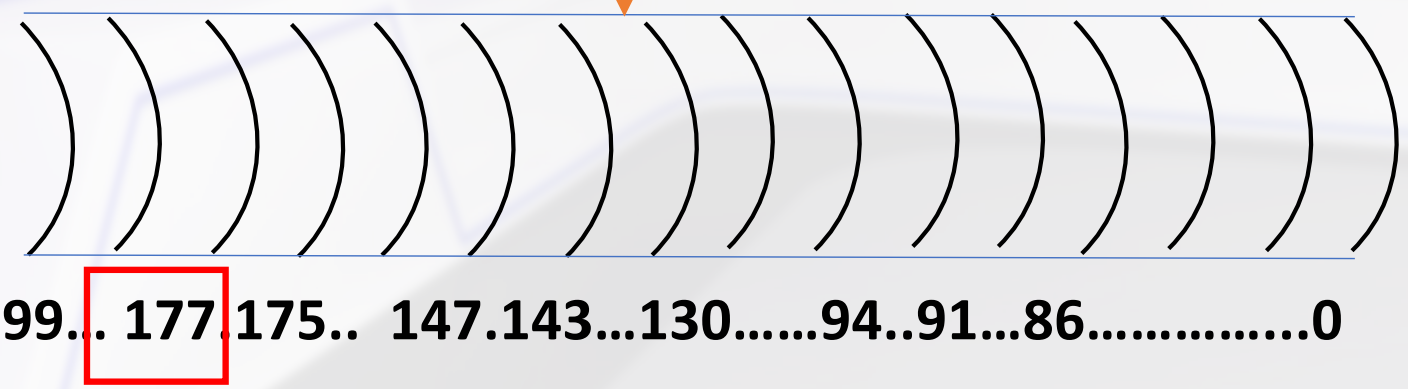
199… 177.175.. 147.143…130……94..91…86……..0

硬盘驱动臂调度算法

3 “电梯调度”算法

磁头移动路径为：

143-147-150-175-177-102-94-91-86

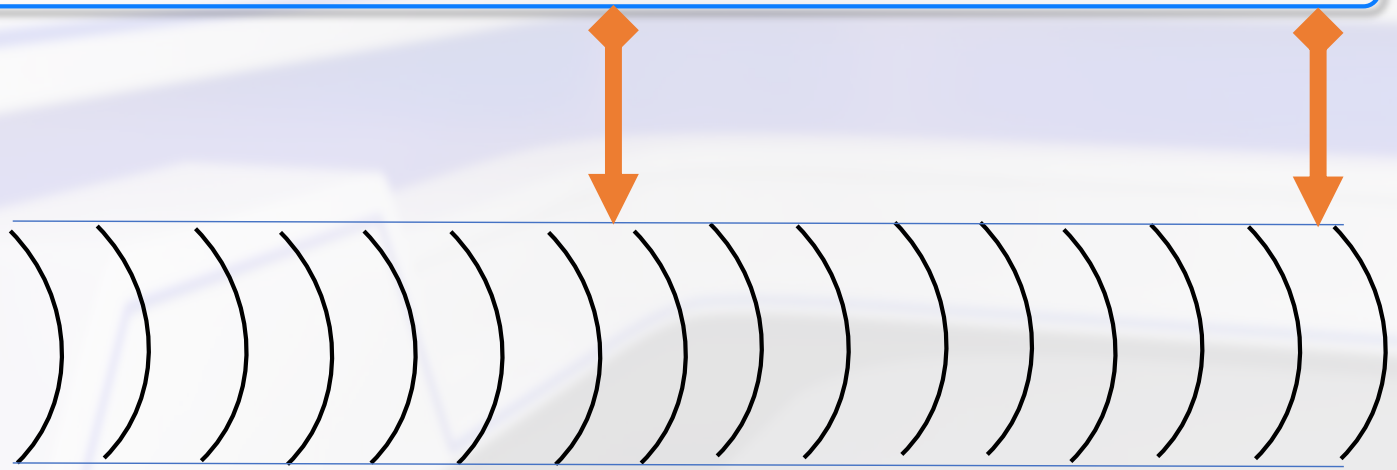


硬盘驱动臂调度算法

5 单向扫描算法

看视频文件、视频服务顺序播放

臂总是从0号柱面至最大号柱面顺序扫描，然后直接返回0号柱面，归途中不再服务，适应不断有大量柱面均匀分布的存取请求



199… 177.175.. 147.143…130……94..91…86……0

硬盘驱动臂调度算法

6 分步扫描算法

磁臂粘着：

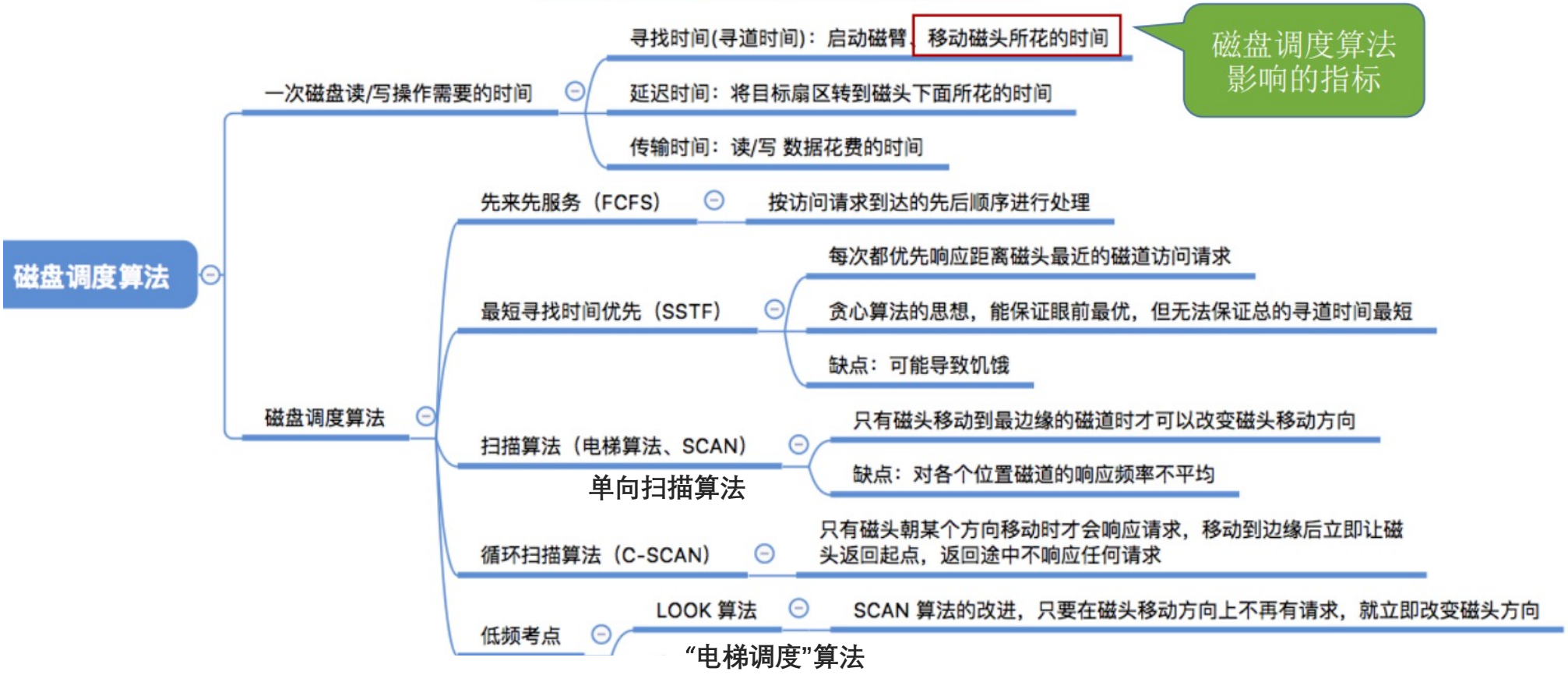
进程对某一磁道有较高的访问频率，反复请求对某一些磁道的I/O操作，从而垄断了整个硬盘设备

将请求队列分成若干长度为N的子队列，磁盘调度将按FCFS算法依次处理这些子队列，每处理一个队列时又是按扫描算法；出现新的磁盘I/O请求，便将新请求进程放入其他队列，可解决磁臂粘着

硬盘驱动臂调度算法

- 先来先服务算法
- 最短查找时间优先算法
- 电梯调度算法
- 扫描算法
- 单向扫描算法
- 分步扫描算法

知识点回顾与重要考点


磁盘调度算法
一次磁盘读/写操作需要的时间
寻找时间(寻道时间)：启动磁臂，移动磁头所花的时间
延迟时间：将目标扇区转到磁头下面所花的时间
传输时间：读/写 数据花费的时间
磁盘调度算法影响的指标
磁盘调度算法
先来先服务（FCFS）
按访问请求到达的先后顺序进行处理
最短寻找时间优先（SSTF）
每次都优先响应距离磁头最近的磁道访问请求
贪心算法的思想，能保证眼前最优，但无法保证总的寻道时间最短
缺点：可能导致饥饿
扫描算法（电梯算法、SCAN）
只有磁头移动到最边缘的磁道时才可以改变磁头移动方向
缺点：对各个位置磁道的响应频率不平均
单向扫描算法
循环扫描算法（C-SCAN）
只有磁头朝某个方向移动时才会响应请求，移动到边缘后立即让磁头返回起点，返回途中不响应任何请求
低频考点
LOOK 算法
SCAN 算法的改进，只要在磁头移动方向上不再有请求，就立即改变磁头方向
“电梯调度”算法

外存储设备管理

- 发出 I/O 请求会导致哪种进程状态演变?

A. 就绪 → 执行　　B. 执行 → 就绪

C. 阻塞 → 执行　　D. 执行 → 阻塞

- 假设磁盘柱面访问序列:
  98,183,37,122,14,124,65,67;读写起始位置为53,
  分别列出采用FCFS, SSTF, SCAN(由外往里), 三种
  不同算法，磁盘柱面访问序列

- 发出 I/O 请求会导致哪种进程状态演变?

A. 就绪 → 执行　　B. 执行 → 就绪

C. 阻塞 → 执行　　D. 执行 → 阻塞

- 假设磁盘柱面访问序列: 98,183,37,122,14,124,65,67;读写起始位置为53, 分别列出采用FCFS, SSTF, SCAN(由外往里), 三种不同算法，磁盘柱面访问序列
- FCFS: 53, 98,183,37,122,14,124,65,67;
- SSFT: 53, 65,67,37,14,98,122, 124, 183;
- SCAN: 53, 65,67,98,122,124,183,199,37,14;