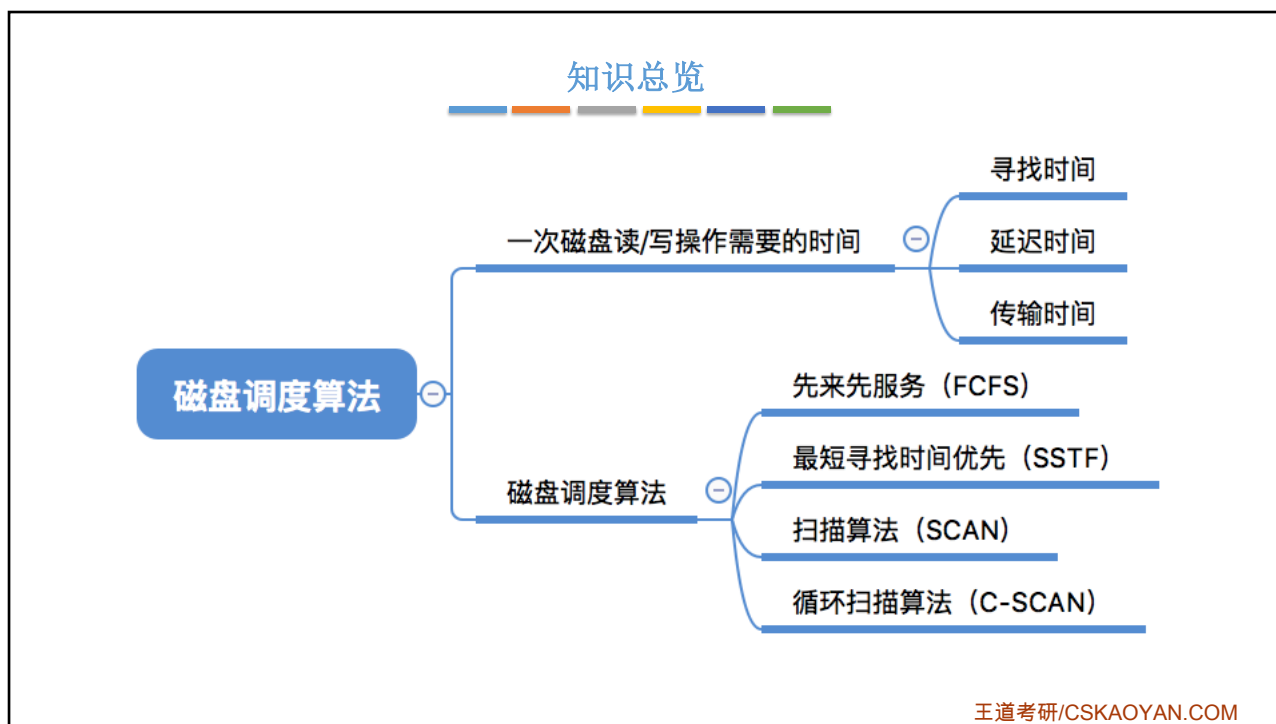


本节内容

磁盘调度算法

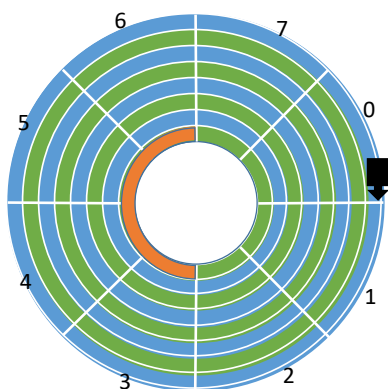
王道考研/CSKAOYAN.COM

1



2

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：

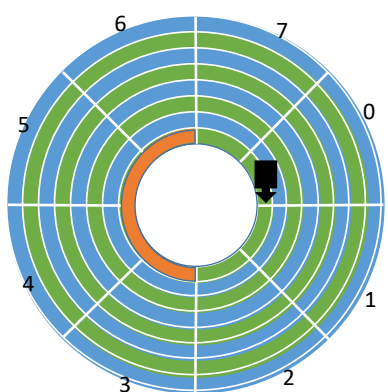
寻道时间 $T_s = s + m \cdot n$

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

王道考研/CSKAOYAN.COM

3

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 $T_s = s + m \cdot n$

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 $T_R = (1/2) \cdot (1/r) = 1/2r$

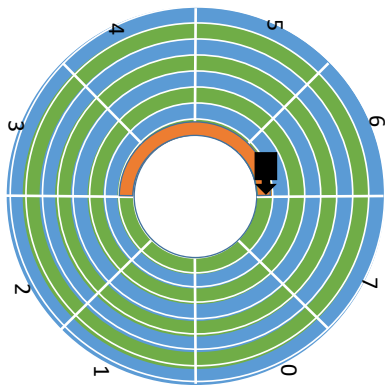
$1/r$ 就是转一圈需要的时间。找到目标扇区平均需要转半圈，因此再乘以 $1/2$

硬盘的典型转速为 5400 转/分，或 7200 转/分

王道考研/CSKAOYAN.COM

4

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 $T_s = s + m \cdot n$

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 $T_R = (1/2) \cdot (1/r) = 1/(2r)$

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 $T_t = (1/r) \cdot (b/N) = b/(rN)$

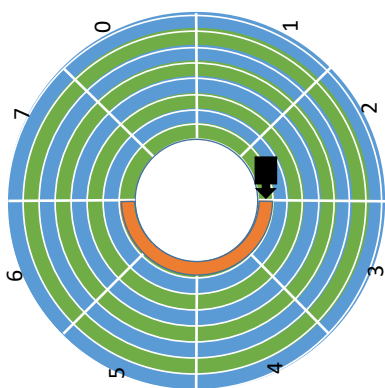
每个磁道要可存 N 字节的数据，因此 b 字节的数据需要 b/N 个磁道才能存储。而读/写一个磁道所需的时间刚好又是转一圈所需要的时间 $1/r$

王道考研/CSKAOTAN.COM

5

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 $T_s = s + m \cdot n$

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 $T_R = (1/2) \cdot (1/r) = 1/(2r)$

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 $T_t = (1/r) \cdot (b/N) = b/(rN)$

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

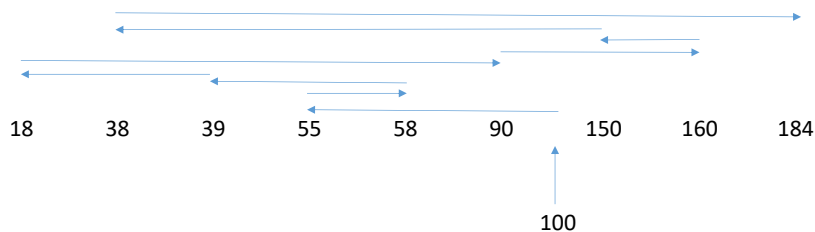
王道考研/CSKAOTAN.COM

6

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，算法性能还算过得去

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。

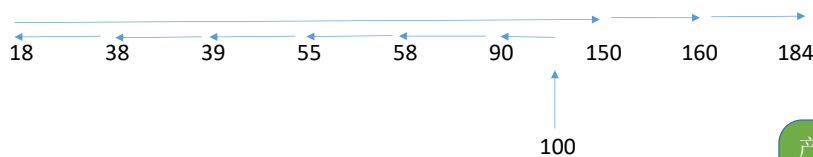
王道考研/CSKAOYAN.COM

7

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

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。

产生饥饿的原因在于：磁头在一个小区域内来回地移动

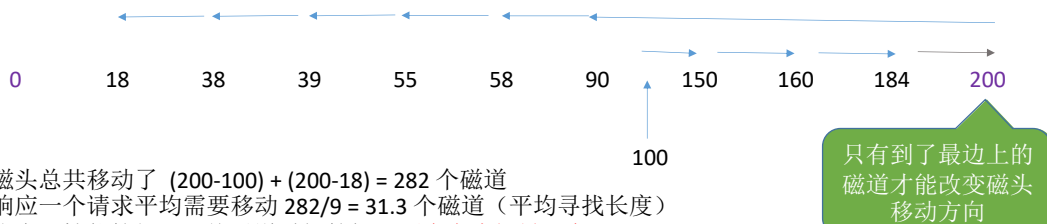
王道考研/CSKAOYAN.COM

8

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。由于磁头移动的方式很像电梯，因此也叫电梯算法。

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，不会产生饥饿现象

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。

② SCAN 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)

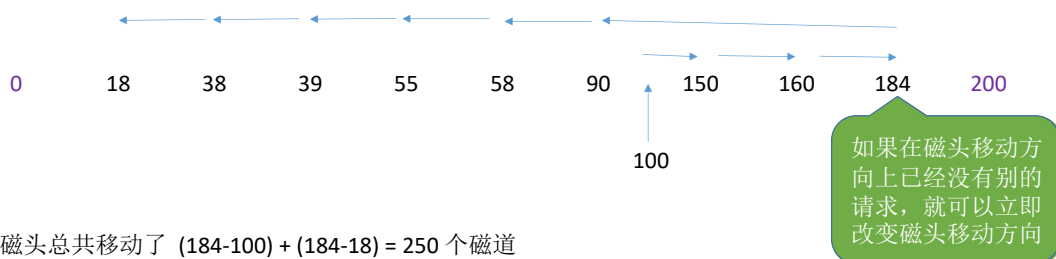
王道考研/CSKAOYAN.COM

9

LOOK 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

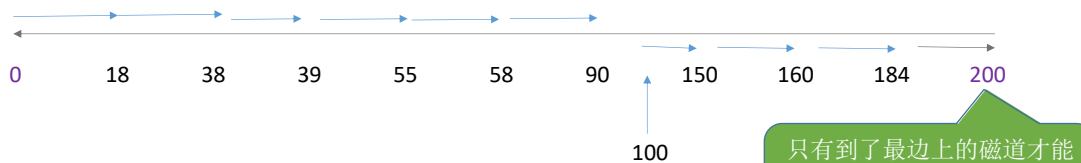
王道考研/CSKAOYAN.COM

10

循环扫描算法 (C-SCAN)

SCAN算法对于各个位置磁道的响应频率不均匀，而 **C-SCAN 算法** 就是为了解决这个问题。规定只有磁头朝某个特定方向移动时才处理磁道访问请求，而**返回时直接快速移动至起始端而不处理任何请求**。

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 $(200-100) + (200-0) + (90-0) = 390$ 个磁道
响应一个请求平均需要移动 $390/9 = 43.3$ 个磁道（平均寻找长度）

优点：比起SCAN来，对于各个位置磁道的响应频率很平均。

缺点：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；并且，磁头返回时其实只需要返回到18号磁道即可，不需要返回到最边缘的磁道。另外，比起SCAN算法来，平均寻道时间更长。

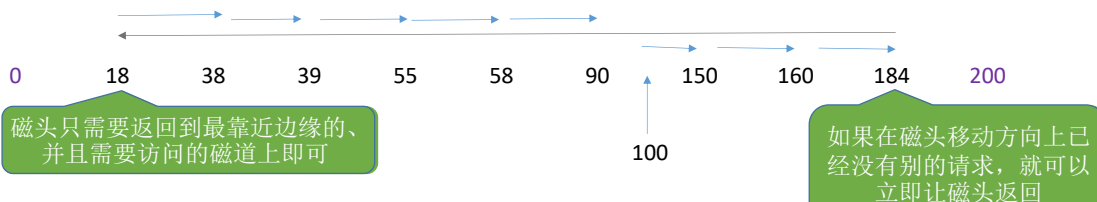
王道考研/CSKAOYAN.COM

11

C-LOOK 调度算法

C-SCAN 算法的主要缺点是只有到达最边上的磁道时才能改变磁头移动方向，并且磁头返回时不一定需要返回到最边缘的磁道上。**C-LOOK 算法** 就是为了解决这个问题。如果磁头移动的方向上已经没有磁道访问请求了，就可以立即让磁头返回，并且磁头只需要返回到有磁道访问请求的位置即可。

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



磁头只需要返回到最靠近边缘的、并且需要访问的磁道上即可

如果在磁头移动方向上已经没有别的请求，就可以立即让磁头返回

磁头总共移动了 $(184-100) + (184-18) + (90-18) = 322$ 个磁道
响应一个请求平均需要移动 $322/9 = 35.8$ 个磁道（平均寻找长度）

优点：比起 C-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

王道考研/CSKAOYAN.COM

12

