各种提高文件系统性能的方法

文件系统的性能2

下面我们继续介绍提高文件系统性能的各种方法

磁盘调度

当有多个访盘请求等待时，采用一定的策略，对这些请求的服务顺序调整安排

→ 降低平均磁盘服务时间，达到公平、高效

公平：一个I/O请求在有限时间内满足

高效：减少设备机械运动带来的时间开销

一次访盘时间 = 寻道时间+旋转延迟时间+传输时间

- 减少寻道时间
- 减少延迟时间

磁盘调度算法(1/9)

例子：假设磁盘访问序列：

98，183，37，122，14，124，65，67

读写头起始位置：53

要求计算：

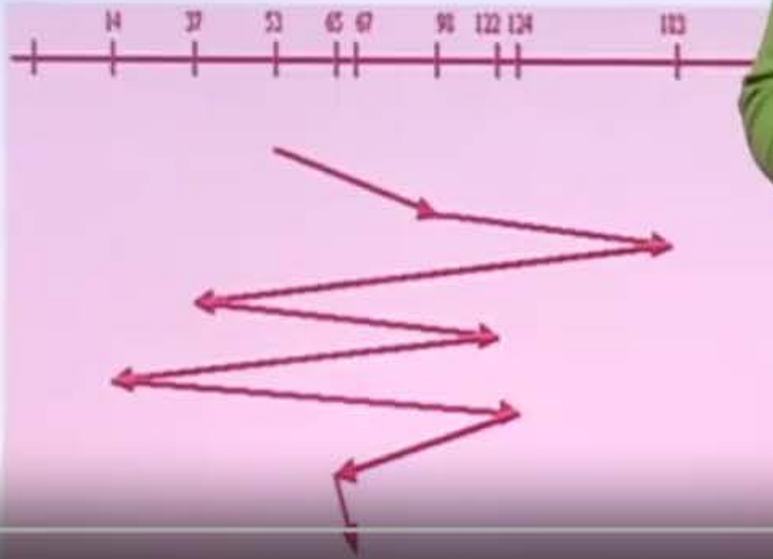
- 磁头服务序列
- 磁头移动总距离（道数）

计算磁头的移动的距离来看看哪一种调度算法比较好

磁盘调度算法(2/9)

- 先来先服务(FCFS)

  按访问请求到达的先后次序服务
- 优点：简单，公平
- 缺点：效率不高，相临两次请求可能会造成最内到最外的柱面寻道，使磁头反复移动，增加了服务时间，对机械也不利



磁盘访问序列：

98，183，37，122，14，124，65，67

读写头起始位置：53

640 磁道 (平均 80)

磁盘调度算法(3/9)

- 最短寻道时间优先(Shortest Seek Time First)

  优先选择距当前磁头最近的访问请求进行服务

  主要考虑寻道优先
- 优点：改善了磁盘平均服务时间
- 缺点：造成某些访问请求长期等待得不到服务

假设磁盘访问序列：

98，183，37，122，14，124，65，67

读写头起始位置：53

236磁道 (平均 29.5)

磁盘调度算法(4/9)

折中权衡
距离、方向

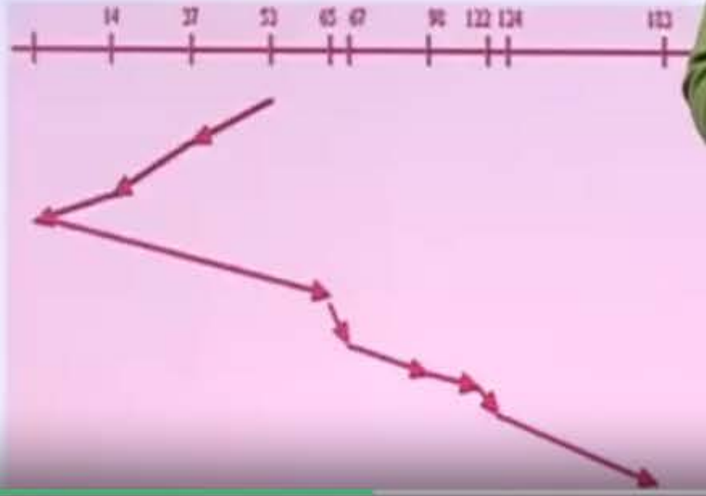
- 扫描算法SCAN（电梯算法）

当设备无访问请求时，磁头不动；当有访问请求时，磁头按一个方向移动，在移动过程中对遇到的访问请求进行服务，然后判断该方向上是否还有访问请求，如果有则继续扫描；否则改变移动方向，并为经过的访问请求服务，如此反复

假设磁盘访问序列：
98，183，37，122，14，124，65，67

读写头起始位置：53

218 磁道 (平均 27.25)



磁盘调度算法(5/9)

减少了新请求的最大延迟

- 单向扫描调度算法C-SCAN
- 总是从0号柱面开始向里扫描
- 按柱面(磁道)位置选择访问者
- 移动臂到达最后一个柱面后，立即带动读写磁头快速返回到0号柱面
- 返回时不为任何的等待访问者服务
- 返回后可再次进行扫描

对新请求的一个最大的延迟，这样的话我们可以让新请求 尽快地得到服务。

磁盘调度策略(6/9)

克服“磁头臂的粘性”

- **N-step-SCAN策略**
  - 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，新请求添加到其他子队列中
  - 如果最后剩下的请求数小于N，则它们全都将在下一扫描时处理
  - N值比较大时，其性能接近SCAN；当N＝1时，即FIF

这就是 N 步策略算法，那么用 N

磁盘调度策略(7/9)

克服“磁头臂的粘性”

- FSCAN策略
  - 使用两个子队列
  - 扫描开始时，所有请求都在一个队列中，而另一个队列为空
  - 扫描过程中，所有新到的请求都放入另一个队列中
  - 对新请求的服务延迟到处理完所有老请求之后

那么它也可以克服磁头臂的黏性。

磁盘调度算法(8/9)

- 旋转调度算法

  旋转调度：根据延迟时间来决定执行次序的调度

三种情况：

- 若干等待访问者请求访问同一磁头上的不同扇区
- 若干等待访问者请求访问不同磁头上的不同编号的扇区
- 若干等待访问者请求访问不同磁头上具有相同的扇区

再次，啊，经过磁头的时候呢，再由其他的磁头来去读取，啊，这就是旋转、

磁盘调度算法(9/9)

- 解决方案：
  - 对于前两种情况：总是让首先到达读写磁头位置下的扇区先进行传送操作
  - 对于第三种情况：这些扇区同时到达读写磁头位置下，可任意选择一个读写磁头进行传送操作

例子：

请求顺序	柱面号	磁头号	扇区号
①	5	4	1
②	5	1	5
③	5	4	5
④	5	2	8

0:00 下面我们继续介绍提高文件系统性能的各种方法 磁盘调度也是一种提高文件系统性能的方法 它指的是当有多个访盘请求等待的时候 采用一定的策略，把这些 请求，它们的服务顺序做调整，重新安排 通过调整和重新安排，目的是为了降低 这些请求的一个平均的服务时间 达到公平、高效的目的 所谓公平呢，是指的是 一个 I/O 请求能够在有限的时间范围内得到满足，因为我们对多个请求 重新调整顺序，要保证 每个请求都能够在一个合理的范围内得到满足 所谓高效呢，指的是尽量地减少 磁盘机械运动所带来的开销，因为如果 按照一个简单的顺序来 对磁盘的请求进行服务，可能会有更多的 磁臂的移动带来的开销 而通过合理地调整这个顺序 就能够减少这个开销，达到降低平均服务时间的这样一个目的 那么一次访盘请求呢通常是由 三个时间组成：寻道时间、 旋转延迟时间、 和数据传送的时间。 这里头呢我们从 如何减少寻道时间，减少 旋转延迟时间来进行磁盘调度 我们给一个例子 这是一个磁盘访问的序列，这个数字呢代表的是 柱面号或者是磁道号 这是请求到达的先后的次序，也就是说 98 号请求先到达 67 请求是最后到达的，在这样一个范围内 那么读写头，磁头的当前的位置呢是在 53 道上 那么我们来通过计算 这个磁头的这个服务的顺序，以及 计算磁头的移动的距离来看看哪一种调度算法比较好 第一种调度算法呢我们称之为先来先服务调度算法 它是按照访问请求到达的先后次序来 进行服务的。 当然了，这种算法非常简单，也很公平 但是它的效率不高，因为很可能会出现 两次这个访盘的请求 一个在最内道，一个在最外道，造成 完成这个请求的话呢，会从最外到最内道的一个寻道 那么也可能呢会使得磁头臂反复地移动 增加了服务时间，当然对机械也不利，因为那是机械运动 我们来以刚才那个例子来看一下这个结果 我们从这张图的示意中可以看到 当前磁头的位置是在 53 道上 然后我们要先服务 98 那么我们从 53 要先到 98 然后服务 183，所以我们要 大跨度地到 183 道上。 然后 就要到 37 道上，所以我们又大跨度地回到了 37 道 因此我们可以从这个图上可以看到，它是一个磁道的大的跨度比较大 那么跨度大，第一，对机械不利，第二，它的这个反复的移动呢，会增加这个服务时间 我们看数字，那么总的移动的道数，如果把这 所有的请求都服务完成，总的移道的次数呢是在 640 个磁道 那么平均 80 个磁道完成一个服务请求 所以呢时间还是比较长的。 第二种调度算法呢叫最短寻道时间优先 这种调度算法呢是优先选择距离 当前磁头最近的这样一个访问请求来进行服务 它的主旨是要考虑 寻道优先，谁离当前磁头 位置最近，就先服务谁 当然它的优点肯定是改善了磁盘的平均服务时间 但是呢，缺点也会导致 我们说的饥饿现象，因为某些

访问请求会长期等待，得不到服务 前提当然是如果不断地有 访问请求来，那么又采用的是最短寻道 时间优先的这种磁盘调度算法的话呢，就会导致饥饿现象 我们来再看一下这个计算的结果 那么按照最短寻道时间优先，那么离 53 道最近的呢 应该是 65，然后是 67 紧接着呢就是到了 37，所以我们要按这样一个顺序进行服务 那么计算的结果呢是总的移动的道数呢是 236 然后平均起来呢是 29.5，可以满足这样一个要求，那么 平均的服务时间相比于这个先来先服务呢是降低了很多 第三种磁盘调度算法呢，我们称之为扫描算法 有的时候也称之为电梯算法，这种调度算法的基本思想是 当磁盘没有访问请求的时候，那么磁头呢是不动的 如果有访问请求到达了，那么磁头呢是按 一个方向移动，在移动的过程中，对遇到的这种访问请求进行服务 然后再去判断 后续还有没有访问请求，如果后续没有访问请求了呢，其实就 改变移动反向，朝相反的方向移动，然后在移动过程中呢继续为 遇到的访问请求服务，这样是反反复复 我们来看一个例子，那么这个例子呢实际上是给出了 当这么多个请求都到达的时候 那我的磁臂调度算法应该是怎么样处理的 当前磁头在 53 道上，那么现在有很多的 访问请求已经来了，那我们假设这个时候磁臂呢是从 磁道号大的地方往磁道号小的地方移动 因此呢我们可以看到先服务 往这个小的方向移动的话，先服务 37 再服务 14，然后前面没有其它的服务请求的时候呢 就沿着这个相反的方向重新继续地移动，然后把沿路上的各种请求把它完成 那么这就是扫描算法。 那么 计算的结果我们可以看到扫描算法呢 对于这样一个磁盘访问序列呢，它的总的 移动的道数呢是 218 磁道 那么平均起来呢是 27 点多，这是 比较好的结果。 那么 扫描算法呢主要呢是对前面的算法的一种折中、权衡 因为它既考虑了一个距离，同时也考虑了一个方向 所以把距离和方向折中起来考虑，得到的是扫描算法 下面呢我们来看对扫描算法的一种改进，叫做单向扫描算法 这个算法的思想呢是这样，总是从 0 号柱面开始向里扫描 假设 0 号柱面在外面，我们总是从 0 号柱面开始向里扫描 然后按照柱面号或者是磁道的位置来选择访问者 那么当移动臂到达了最后一个柱面的时候 那就，我们原来是说往回方向走，这回不是了，立即带动读写磁头 快速地又返回到了 0 号柱面 在返回的过程中不会为其他的这个访问请求服务 返回之后呢，再重新从 0 号 再往里头去扫描，这就是所谓单向扫描调度算法 那么它的主要的目的呢是为了减少 对新请求的一个最大的延迟，这样的话我们可以让新请求 尽快地得到服务。 我们还可以 有其他的一些策略，比如说叫做 N 步扫描算法 所

谓 N 步扫描算法呢指的是把磁盘的这种请求队列呢划分为 长度为 N 的子队列，每次呢是服务这个子队列当中的这些 访问请求。 在处理一个队列的过程中， 如果还有新的请求到来，就加入到其他的子队当中。 那么，因为每个 队列的长度都是 N，如果最后剩下的一个请求队列长度 小于 N 的时候呢，就把它们一次全服务完成。 那么，N 步的话呢，如果 N 比较大，那么它的这个性能呢，是接近于扫描算法。 如果 N 等于 1 ，其实就退化为 FIFO 算法。 这就是 N 步策略算法，那么用 N 步策略算法主要是为了解决"磁头臂的一个黏性问题"。 我们还可以继续对，啊，调度算法，进行相应的改进。 比如说 FSCAN 策略， FSCAN 策略呢，是使用两个子队列。 那在扫描开始的时候呢，所有的请求呢，都在第一个，啊，子队列里头。 那么另外一个队列呢，现在是空的，在扫描的过程中，如果 有新的请求到来，那都进入到，啊，第二个子队列。 第一个子队列都完成了之后，那么 再去处理第二个子队列，这就是 FSCAN 这个策略。 那么它也可以克服磁头臂的黏性。 刚才我们介绍的呢都是磁臂 的移动，啊，减少它的开销，所以我们这是磁臂的移动的调度算法。 下面我们来看一下，旋转延迟的这个调度算法。 所以旋转调度呢，是根据延迟时间来决定执行次序。 因为我们知道，磁臂带动了磁头，寻道 停在了某个磁道上的时候呢，那么这个盘是转的，那么 等待所要读的这个扇区，经过自己的这个磁头，才 能把这个数据读出来，所以它有一个，啊，旋转的一周， 最慢一周我能读到这个数据块。 所以这个旋转调度呢，是指的对这段时间的一个优化。 那么我们来看看，有三种情形。 第一种情形呢，是说若干个等待访问的请求呢， 它们访问的是同一磁头的不同扇区，就是，磁头相同 但是呢，扇区是不一样的；第二种情况呢，是若干个 等待访问的这个请求呢，是访问不同磁头的 不同扇区，磁头不同、 扇区也不同； 第三种情况呢，是若干等待访问请求， 它们访问的是不同的磁头，但是扇区是同一个。 那么前面我们已经说过，啊， 虽然有多个磁头，任何时刻只能有一个磁头在工作，这当然指的是一个移动头的磁盘，啊。 我们不讨论 SSD 这种固态盘，因此 对于第一种情况和第二种情况来讲呢，比较简单。 因为，它们访问的同一磁道上的不同扇区，尽管磁头可能是相同或者是不同，但是因为扇区， 是不一样的，所以哪个扇区先经过这个磁头， 那就先服务这个要求。 因此呢，前两种情况比较简单，按照扇区，啊， 经过磁头的先后次序来进行服务。 而第三种情况， 比较复杂，因为它是不同磁头，但是要读同一个扇区， 所以呢，我们只能够随机选择一个磁

头，让它去读取相应的内容。 那么其他磁头的这种，扇区的这个服务呢， 要等着这个扇区，在经过了一整圈， 再次，啊，经过磁头的时候呢，再由其他的磁头来去读取， 啊，这就是旋转、延迟调度算法的三种 场景，以及相应的策略。 我们再总结一下，对于前两种场景呢， 就是总是让先到达读写磁头位置的这样的扇区呢，先进行传送工作， 第三种情况呢，因为这些扇区呢，是同时到达读习磁头的位置，所以 可以任意选择一个读写头进行相应的操作，而其他的呢，是要经过 另外一整圈之后，啊，延迟一整圈之后呢，再进行相应的传送操作。 我们可以看一个例子： 这个例子当中呢，柱面号都是 5，就说明已经寻道，啊，到位了。 然后呢，我们来看一下，有四个请求，分别它们的 磁头号和扇区号都给出来了，这四个请求怎么服务呢？ 如果按照我们刚才的策略，我们假设， 现在这个磁头的位置，是从，啊，这个零号扇区这个位置， 或者是 1 号扇区这个位置，所以我们可以看到，那么肯定首先要满足这个 1 号扇区的读写， 因此呢，请求 ① 先完成，但是呢， 2 和 3 呢，我们可以看到，那么它们的 磁头号不一样，但是扇区号是一样的，所以随机的挑一个，比如说挑了 2 了。 那么 3 就得要等一圈之后，因此呢， 在满足这样一个， 啊，这个场景的这样一个 调度算法之后呢，我们可以看到，可能的服务的 顺序呢，是 1 2 4 3 或者是 1 3 4 2 这样 两种可能性。