各种提高文件系统性能的方法

文件系统的性能1

文件系统的性能问题

磁盘服务

→ 速度成为系统性能的主要瓶颈之一

设计文件系统应尽可能减少磁盘访问次数

提高文件系统性能的方法：

目录项(FCB)分解、当前目录、磁盘碎片整理

块高速缓存、磁盘调度、提前读取、合理分配磁盘空间、信息的优化分布、RAID技术

提高文件系统性能的方法 后面呢，是我们下面要介绍的各种方法

块高速缓存(BLOCK CACHE)

又称为文件缓存、磁盘高速缓存、缓冲区高速缓存
是指：在内存中为磁盘块设置的一个缓冲区，保存了磁盘中某些块的副本

- 检查所有的读请求，看所需块是否在块高速缓存
- 如果在，则可直接进行读操作；否则，先将数据读入块高速缓存，再拷贝到所需的地方
- 由于访问的局部性原理，当一数据块被读入块高速缓存以满足一个I/O请求时，很可能将来还会再次访问到这一数据块

我们把它放到内存，这样的话速度就提高了 下面我们讨论，如何实现块高速缓存

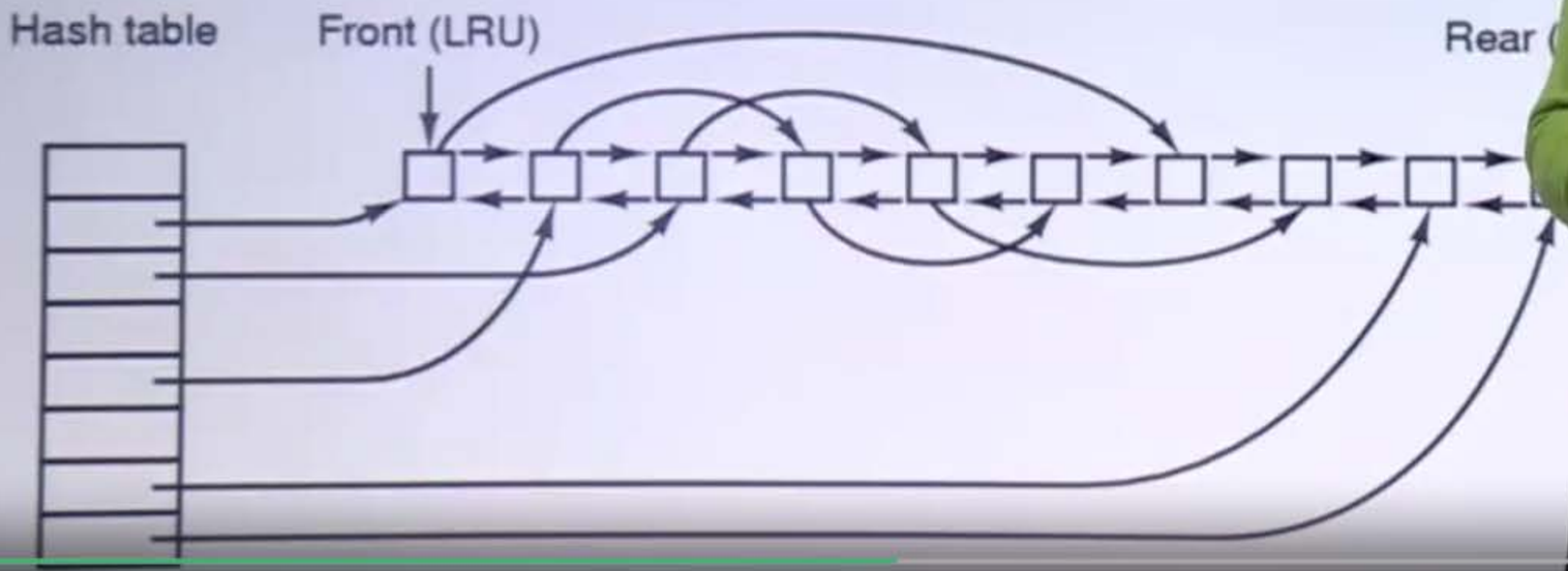
关于实现

块高速缓存满时需要进行置换

- 块高速缓存的组织
- 块高速缓存的置换（修改LRU）
- 块高速缓存写入策略

该块是否不久后会再次使用

该块是否会影响文件系统的一致性



提前读取

- 思路：每次访问磁盘，多读入一些磁盘块
- 依据：程序执行的空间局部性原理
- 开销：较小(只有数据传输时间)
- 具有针对性

可以连续读入几块，这叫提前读取 在介绍了块高速缓存

WINDOWS 的文件访问方式(1/3)

- 不使用文件缓存
  - 普通方式
  - 通过Windows提供的FlushFileBuffer函数实现
- 使用文件缓存
  - 预读取。每次读取的块大小、缓冲区大小、置换方式
  - 写回。写回时机选择、一致性问题
- 异步模式
  - 不再等待磁盘操作的完成
  - 使处理器和I/O并发工作

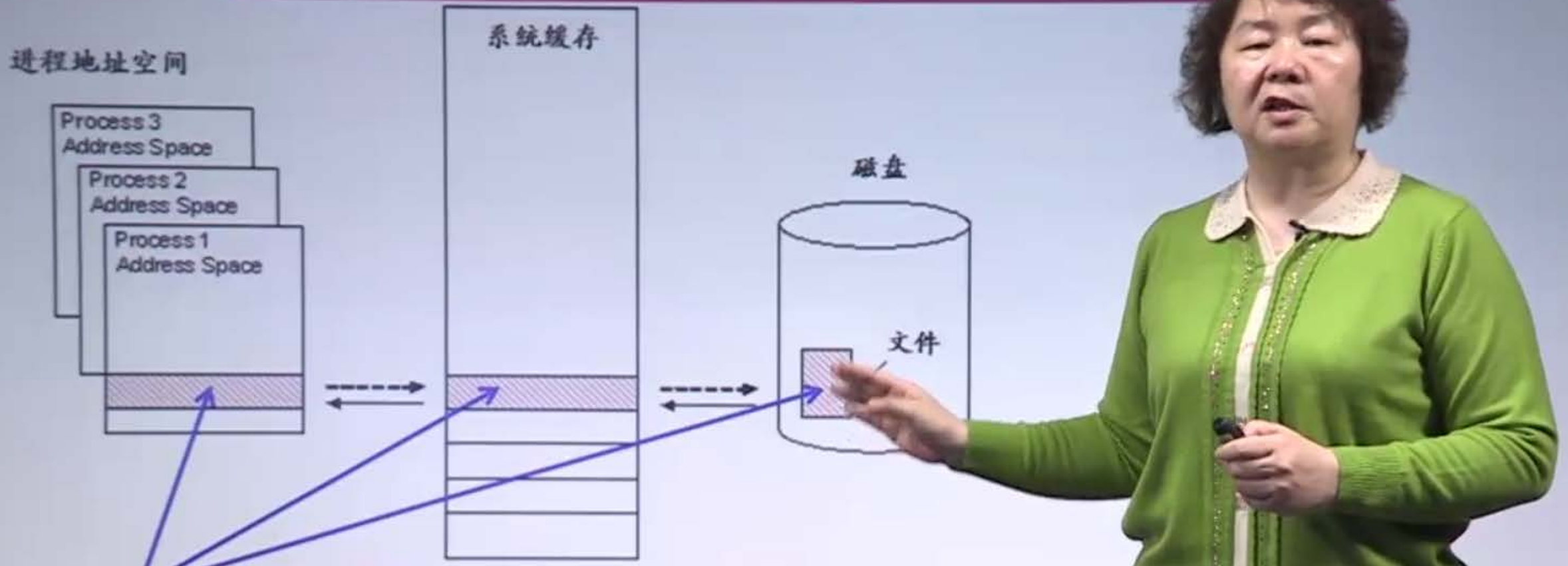
WINDOWS 的文件访问方式(2/3)

用户对磁盘的访问通过访问文件缓存来实现

- 由Windows的Cache Manager实现对缓存的控制
  - 读取数据的时候预取
  - 在Cache满时，根据LRU原则清除缓存的内容
  - 定期更新磁盘内容使其与Cache一致（1秒）

- Write-back机制
  - 在用户要对磁盘写数据时，只更改Cache中的内容，由Cache Manager决定何时将更新反映到磁盘

WINDOWS 的文件访问方式(3/3)



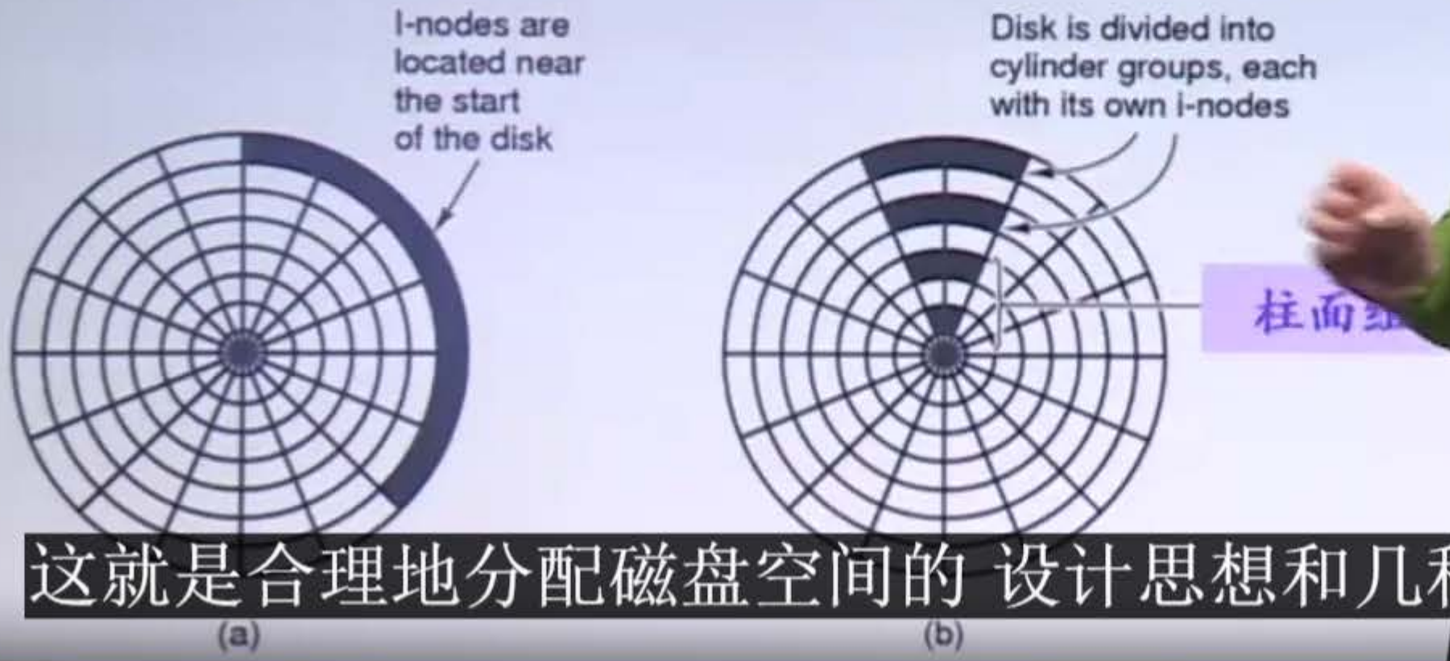
阴影部分为需要访问的数据，数据在磁盘、系统缓存和进程地址空间有3份拷贝，通常下用户对数据的修改并不直接反映到磁盘上，而是通过write-back机制由lazy writer定期地更新到磁盘

合理分配磁盘空间

分配磁盘块时，把有可能顺序存取的块放在一起

→ 尽量分配在同一柱面上，从而减少磁盘臂的移动次数和距离

例子：



这就是合理地分配磁盘空间的 设计思想和几种解决的参考。

0:00 下面，我们介绍如何提高文件系统的性能 这部分内容比较多，所以我们分三部分介绍 关于文件系统的性能问题 主要集中在磁盘服务上 因为访问磁盘比访问内存慢得多 两者的速度差异非常巨大，因此 由于磁盘的服务，它的速度 就成为了制约文件系统性能的因素 因此，作为操作系统的设计者 在设计文件系统、实现文件系统时 必须要考虑如何尽可能地减少磁盘访问的次数 那么，提高文件系统性能的方法有非常多 那么这里头呢，是每一个局部的细节都会对整体 有影响，因此我们在每一个环节 都必须考虑如何减少磁盘访问的次数，下面 是提高文件系统的各种方法，其中 目录项分解法、当前目录 以及磁盘的碎片整理 都是我们前面已经介绍过的 提高文件系统性能的方法 后面呢，是我们下面要介绍的各种方法 我们要介绍的第一种 提高文件系统性能的方法叫做块高速缓存 它又被称为文件缓存 磁盘高速缓存，或者是缓冲区高速缓存 它的基本思想呢，是指在内存当中 为磁盘块设置一个缓冲区 在这个缓冲区当中，保存了磁盘中的若干块的 副本，实际上，也就是说这块 内存的区域，逻辑上呢，是属于磁盘的 那么，当对文件系统进行操作的时候 我们来看一下，如何利用这个块高速缓存 我们有大量的读块的操作 当每次有读请求来的时候 系统呢，会首先检查这些读请求 看看它所需要的数据块是否 保存在块高速缓存当中 如果这些块已经 保存在块高速缓存当中，那么就可以直接 把这个块从块高速缓存 拷贝到它的地址空间 否则的话呢，要先从磁盘上把这个数据块先读入高速缓存 再拷贝到所需要的地方，那么这块区域 啊，我们就称之为块高速缓存 块高速缓存的操作方式呢，就是按这样一种方式进行 为什么要引入这种技术 那么主要的原因就是程序访问的局部性原理 因为当一个数据块被读入块高速缓存，啊 满足了这一次的 I/O 请求后，很可能 它的内容会再次被访问到，因此呢 我们把它放到内存，这样的话速度就提高了 下面我们讨论，如何实现块高速缓存 首先我们来看一下，在内存当中，块高速缓存是如何组织的 在块高速缓存当中，有若干个数据块 这些数据块呢，我们首先呢，把它按照双向链表的方式，组织成一个链 当要访问一个数据块的时候，就把它从这个链中摘出来 然后把它挂接到链尾，但是呢 我们知道，当有一个读请求来了之后，我们根据要读的数据块的块号 要在块高速缓存当中看一看这个数据块是不是已经 进入了块高速缓存，如何快速地能够判断出这一点呢 我们可以通过一个散列函数 那么把这个块号，按照散列函数进行散列，得到一个散列值 因此呢，我们会按照散列值呢，组织成一个散列表 那么这样的话呢，我 任何一个数据块来了之后，那我通过散列 在表里头找到对应的那一项，如果这一项没有，就说明这一块还没有进入 块高速缓存，如果有

了，那我们就可以找到这个块的位置 当然了，如果散列值有冲突的话呢，我们再进一步通过链来解决冲突问题 那么第二个，我们要 考虑的问题呢，实际上就是块高速缓存的置换问题 因为，块高速缓存，空间不大 那么，当它满了以后呢，我们要讨论，把哪些数据块置换出去 那么在这个置换过程中，除了刚才我们所说的这样一种策略，我们 每访问一个数据块，就把这数据块从链中摘出来放到链尾，那么这是 完全符合 LRU 最近最少使用，这样一个置换策略的设计思想 但是呢，我们除了精准地来描述这个 LRU 的这样一个场景之外，那么我们还要考虑另外一个因素，就是 我们选择的这一块，是否以后会再次使用 因此我们把会，以后会再次使用的块呢，我们要把它往到链尾放 这是要考虑这样一个问题。 第三个呢，我们要考虑的问题是 块高速缓存的写入策略，那么因为，我们会 在文件系统中，会考虑这样一个问题 如果你的写入不及时，会不会影响文件系统的一致性 比如说，一些重要的数据结构，像 i 节点啊，目录项啊，那我们要把它及时写回到磁盘 那么，关于写入策略，无非是两种，一种呢，就是 在块高速缓存当中，一旦有修改，就立刻把这个数据块写回到磁盘 那么还有一种，就是回写的方式会 积攒一段时间，那么可能，隔一段时间再把相关的内容写回到磁盘 那么不同的操作系统呢，也采用了不同的方案 那么提高文件系统性能的 另外一种办法呢，叫做提前读取 啊，也称之为预取，它的思路呢就是 每次访问磁盘的时候，我会多读入一些磁盘块 那么比如说，我要访问一个数据块 我把它这个块找到之后，我顺带着要把它 后面的几块，把它读进内存，那么主要的依据呢是 依然是程序执行的局部性原理 那么这样做的好处呢，实际上就是，减小了开销 因为你已经找到磁盘块了，顺带着把它后面 几块读入内存，所花费的时间呢，只是数据传输的时间 寻到的时间，旋转延迟时间，其实都可以 省略了，因此呢，它是一个比较经济可行的方案 当然了，这也是有一些针对性，那么对于数据 啊，数据块，那么可以呢，我们多读入几块 那么代码块呢，我们可以少读入几块，但是呢我们 可以连续读入几块，这叫提前读取 在介绍了块高速缓存 和提前读取两种提高文件系统性能的方法之后呢 我们来以 WINDOWS 操作系统为例 介绍一下在 WINDOWS 当中，文件的访问方式 在 WINDOWS 当中的文件访问方式呢，通常有三种 第一种是，不使用文件缓存 第二种是，使用文件缓存 这里的文件缓存，实际上就是我们刚才介绍的 块高速缓存；第三种呢，是异步模式 不使用文件缓存呢，实际上是一个普通的文件访问方式 如果要把在内存的内容 写回到磁盘，必须调用

相应的函数来完成 那么使用文件缓存呢，我们来看一下 要考虑预读取和写回的问题。 那么预读取呢就前面我们所介绍的提前读取的策略。 第三种方式呢叫异步模式。 那么这种模式呢和我们 现在所讨论的主题，文件系统的性能呢关系不大，我们简单介绍一下它的思想。 所谓异步模式呢，就是当一个进程提出访盘请求之后呢不会去等待 磁盘操作的完成，而去继续执行其它的工作，使得 CPU 和 I/O 能够并发工作。 所以异步模式是这样一种、 啊一种编程的开发的方法。 下面我们来着重探讨一下， "用户对磁盘的访问是通过访问文件 缓存方式来实现的"这里头的一些设计思想。 那么在对文件缓存的控制管理方面呢， Windows 会提供一个专门的模块叫 Cache Manager， 由它来完成对文件缓存的管理和控制。 着重呢从以下三个方面来完成：第一个呢是要 考虑在读取数据的时候呢增加一些预取的策略； 而且还要考虑当我的文件缓存满了的时候， 根据 LRU 这样一个算法来清除 在文件缓存中的部分内容；然后我们必须考虑， 因为缓存的内容如何和磁盘保持一致呢， 我们会通过一个定期的更新磁盘 来使得在内存的内容和 磁盘的内容保持一致，那么通常呢是一秒钟来做这样一件事情。 那么每一秒钟呢实际上都有这么一个写回， 把块高速缓存、 文件缓存内容写回到 磁盘，这是我们前面所介绍的写回的策略。 那么有了这样一个策略之后呢，用户在 执行过程中如果要去写盘的话，往磁盘上写 数据的话呢，实际上并不是真正的往磁盘上写这个数据， 而是改变什么呢？在文件缓存当中的相应内容。 然后定期地由 Cache Manager 来决定 什么时候将文件缓存的这些修改的内容反映到磁盘上。 这就是采用了文件缓存之后要考虑的一些问题。 我们再来看一个例子，在这个例子当中我们看一下： 这是磁盘，啊，磁盘上有相应的文件。 这一块儿呢实际上是文件缓存，或者是系统缓存、 块高速缓存， 是把内存的一部分空间 啊，给它划出来，然后用于这样一个用途。 就是逻辑上这块儿空间 属于磁盘，可以用磁盘块儿、 数据块儿 块号到这个空间进行相应的查找。 那么左边呢我们可以看到是用户的地址空间，每个地址空间它要读数据， 这个时候要把相应的数据读到它的地址空间， 由这个进程进行后续的处理。 好，我们来看一下， 采用了文件缓存之后啊， 那我们可以看一下，这里头数据在三个地方啊 有内容啊，同一份数据在三个地方有副本。 一个在原来磁盘上，一个呢是在系统的缓存里头， 文件缓存里头，还有一个呢是在用户的地址空间。 因此当用户在地址空间当中对自己的这个 数据进行相应的修改之后，那么实际上是这个修改的内容 反映到了块高

速缓存里头，而不是立即写回到磁盘。通常呢是当定时的由一个特定的，比如说我们说的 Cache Manager，它来决定什么时候把这部分内容写回到磁盘。那么读数据呢是首先把数据从磁盘读入到系统缓存，然后再把内容拷贝到这个用户的地址空间。所以这就是采用了块高速缓存之后在 Windows 当中的一个典型的文件访问方式。下面呢我们来介绍另一种提高文件系统性能的方法：合理地分配磁盘空间。这种方法呢它的基本思想是：在分配磁盘块儿的时候，把有可能顺序存取的磁盘块，把它尽可能地放在一起。比如说我们把这些磁盘块都分配在同一个柱面上，那么在读取这些磁盘块的时候呢，磁臂移动的次数和距离都少了很多。我们来看一个例子。左边这张图呢实际上描述了一个 i 节点区的一个示意，根据我们前面所介绍的 Unix 类的文件系统的布局，i 节点区呢是在磁盘的这个分区的前半部分。那么从这张图上可以看出来，这个 i 节点区呢，是把它划分到了磁盘的外道，啊最外道。那么当读一个文件的时候，我们首先先要找到这个文件所对应的 i 节点。找到 i 节点之后呢，我们再根据 i 节点里头记录的文件的地址呢找到这个文件的其它的数据块。所以我们每次都要先访问 i 节点，再访问这个文件的数据块。如果 i 节点区在磁盘的最外道，而这个文件它的这个数据块恰好又保存在这个磁盘的最里道，因此呢要读这个文件的话，或者是要查找这个文件的话，可能要外道里道，外道里道不断地进行磁臂的移动，它的次数和距离都会增加。那么如何解决这个问题呢？一种可能的办法是把这个 i 节点区不放在磁盘的最外道，而是放在中间的磁道上。那么这样的话呢数据放在外面道上，或者放在里面道上，那么磁臂的移动的距离都不会太大。这是一种解决方法。而第二种方法呢就如这个图右边这个展示的结果，我们来说一下它的思想：它首先把这个磁道，或者是柱面呢分成了若干组，然后呢把 i 节点区呢也划分成若干部分，每一组柱面，或者是每一组磁道呢都有一个 i 节点区。那么如果一个文件啊，它要是保存在某一个柱面组的时候，那么它的 i 节点呢也就放在同一个柱面组上的这个 i 节点区里头。所以这样的话找到 i 节点，再找到它的磁盘块，基本都是在同一个柱面组，不太会引起很长的这种磁臂的移动。这就是合理地分配磁盘空间的设计思想和几种解决的参考。