**分类&简介**

硬盘是电脑主要的存储媒介之一，由一个或者多个铝制或者玻璃制的碟片组成。碟片外覆盖有铁磁性材料。硬盘有固态硬盘（SSD 盘，新式硬盘）、机械硬盘（HDD传统硬盘）、混合硬盘（HHD 一块基于传统机械硬盘诞生出来的新硬盘）。SSD采用闪存颗粒来存储，HDD采用磁性碟片来存储，混合硬盘(HHD: Hybrid Hard Disk)是把磁性硬盘和闪存集成到一起的一种硬盘。绝大多数硬盘都是固定硬盘，被永久性地密封固定在硬盘驱动器中。

**机械硬盘**

机械硬盘即是传统普通硬盘，主要由：盘片，磁头，盘片转轴及控制电机，磁头控制器，数据转换器，接口，缓存等几个部分组成。

磁头**可沿盘片的半径方向运动**，加上盘片每分钟几千转的高速旋转，磁头就可以定位在盘片的指定位置上进行数据的读写操作。信息通过离磁性表面很近的磁头，由电磁流来改变极性方式被电磁流写到磁盘上，信息可以通过相反的方式读取。 硬盘作为精密设备，尘埃是其大敌，所以进入硬盘的空气必须过滤。



**原理**

机械硬盘中所有的盘片都装在一个旋转轴上，每张盘片之间是平行的，在每个盘片的存储面上有一个磁头（有多个盘面和多个磁头），磁头与盘片之间的距离比头发丝的直径还小，所有的磁头联在一个磁头控制器上，由磁头控制器负责各个磁头的运动。

硬盘首先在逻辑上被划分为磁道、柱面以及扇区，如图1所示

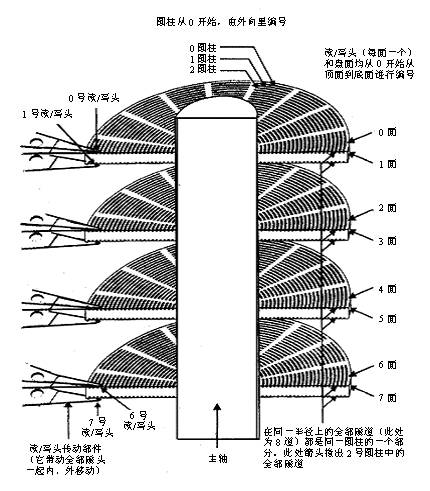


图1

每个盘片的每个面都有一个读写磁头，磁盘盘面区域的划分如图2所示。磁头靠近主轴接触的表面，即线速度最小的地方，是一个特殊的区域，它不存放任何数据，称为启停区或着陆区（Landing Zone），启停区外就是数据区。在最外圈，离主轴最远的地方是“0”磁道，硬盘数据的存放就是从最外圈开始的。那么，磁头是如何找到“0”磁道的位置的呢？有一个“0”磁道检测器，由它来完成硬盘的初始定位。“0”磁道是如此的重要，以致很多硬盘仅仅因为“0”磁道损坏就报废，这是非常可惜的。

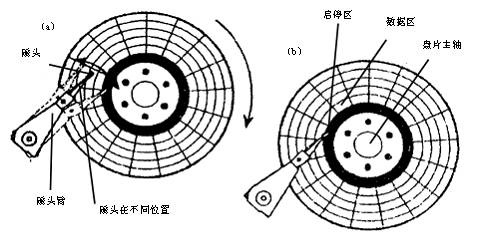


图2

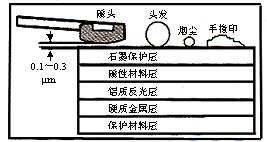
硬盘驱动器采用高精度、轻型磁头驱动/定位系统。这种系统能使磁头在盘面上快速移动，可在极短的时间内精确地定位在由计算机指令指定的磁道上。目前，磁道密度已高达5400Tpi（每英寸磁道数）或更高；

硬盘驱动器内的电机都是无刷电机，在高速轴承支持下机械磨损很小，可以长时间连续工作。高速旋转的盘体产生明显的陀螺效应，所以，在硬盘工作时不宜搬动，否则，将增加轴承的工作负荷。为了高速存储和读取信息，硬盘驱动器的磁头质量小，惯性也小，所以，硬盘驱动器的寻道速度明显快于软驱和光驱。

硬盘驱动器磁头与磁头臂及伺服定位系统是一个整体。伺服定位系统由磁头臂后的线圈和固定在底板上的电磁控制系统组成。由于定位系统限制，磁头臂只能在盘片的内外磁道之间移动。因此，不管开机还是关机，磁头总在盘片上；所不同的是，关机时磁头停留在盘片启停区，开机时磁头“飞行”在磁盘片上方。

早期的硬盘在每次关机之前需要运行一个被称为Parking的程序，其作用是让磁头回到启停区。现代硬盘在设计上已摒弃了这个虽不复杂却很让人不愉快的小缺陷。硬盘不工作时，磁头停留在启停区，当需要从硬盘读写数据时，磁盘开始旋转。旋转速度达到额定的高速时，磁头就会因盘片旋转产生的气流而抬起，这时磁头才向盘片存放数据的区域移动。盘片旋转产生的气流相当强，足以使磁头托起，并与盘面保持一个微小的距离。这个距离越小，磁头读写数据的灵敏度就越高，当然对硬盘各部件的要求也越高。早期设计的磁盘驱动器使磁头保持在盘面上方几微米处飞行。稍后一些设计使磁头在盘面上的飞行高度降到约0.1μm～0.5μm，现在的水平已经达到0.005μm～0.01μm，这只是人类头发直径的千分之一。气流既能使磁头脱离开盘面，又能使它保持在离盘面足够近的地方，非常紧密地跟随着磁盘表面呈起伏运动，使磁头飞行处于严格受控状态。磁头必须飞行在盘面上方，而不是接触盘面，这种位置可避免擦伤磁性涂层，而更重要的是不让磁性涂层损伤磁头。但是，磁头也不能离盘面太远，否则，就不能使盘面达到足够强的磁化，难以读出盘上的磁化翻转（磁极转换形式，是磁盘上实际记录数据的方式）。

这种硬盘就是采用温彻斯特（Winchester）技术制造的硬盘，所以也被称为温盘。其结构特点如下，磁头在启动、停止时与盘片接触，在工作时因盘片高速旋转，带动磁头“悬浮”在盘片上面呈飞行状态（空气动力学原理），“悬浮”的高度约为0.1μm～0.3μm，这个高度非常小，如下标出了这个高度与头发、烟尘和手指印的大小比较关系，从这里可以直观地“看”出这个高度有多“高”。



**硬盘存储的逻辑结构**

**1.盘面号**

硬盘的盘片一般用铝合金材料做基片，高速硬盘也可能用玻璃做基片。玻璃基片更容易达到所需的平面度和光洁度，且有很高的硬度。磁头传动装置是使磁头部件作径向移动的部件，通常有两种类型的传动装置。一种是齿条传动的步进电机传动装置；另一种是音圈电机传动装置。前者是固定推算的传动定位器，而后者则采用伺服反馈返回到正确的位置上。磁头传动装置以很小的等距离使磁头部件做径向移动，用以变换磁道。

硬盘的每一个盘片都有两个盘面（Side），即上、下盘面，一般每个盘面都会利用，都可以存储数据，成为有效盘片，也有极个别的硬盘盘面数为单数。每一个这样的有效盘面都有一个盘面号，按顺序从上至下从“0”开始依次编号。在硬盘系统中，盘面号又叫磁头号，因为每一个有效盘面都有一个对应的读写磁头。硬盘的盘片组在2～14片不等，通常有2～3个盘片，故盘面号（磁头号）为0～3或0～5。

**2.磁道**

磁盘在格式化时被划分成许多同心圆，这些同心圆轨迹叫做磁道（Track）。磁道从外向内从0开始顺序编号（最外面是0磁道）。硬盘的每一个盘面有300～1024个磁道，新式大容量硬盘每面的磁道数更多。信息以脉冲串的形式记录在这些轨迹中，这些同心圆不是连续记录数据，而是被划分成一段段的圆弧，这些圆弧的角速度一样。由于径向长度不一样，所以，线速度也不一样，外圈的线速度较内圈的线速度大，即同样的转速下，外圈在同样时间段里，划过的圆弧长度要比内圈划过的圆弧长度大。每段圆弧叫做一个扇区，扇区从“1”开始编号，每个扇区中的数据作为一个单元同时读出或写入。一个标准的3.5in硬盘盘面通常有几百到几千条磁道。磁道是“看”不见的，只是盘面上以特殊形式磁化了的一些磁化区，在磁盘格式化时就已规划完毕。

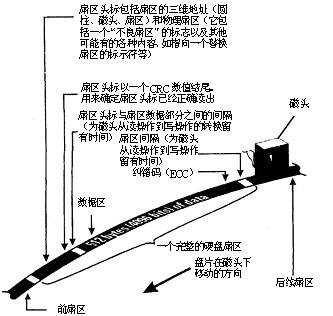
**3.柱面**

所有盘面上的同一磁道构成一个圆柱，通常称做柱面（Cylinder），每个圆柱上的磁头由上而下从“0”开始编号。数据的读/写按柱面进行，即磁头读/写数据时首先在同一柱面内从“0”磁头开始进行操作，依次向下在同一柱面的不同盘面即磁头上进行操作，只在同一柱面所有的磁头全部读/写完毕后磁头才转移到下一柱面，因为选取磁头只需通过电子切换即可，而选取柱面则必须通过机械切换（磁头寻道是一个机械移臂的过程）。电子切换相当快，比在机械上磁头向邻近磁道移动快得多，所以，数据的读/写按柱面进行，而不按盘面进行。也就是说，一个磁道写满数据后，就在同一柱面的下一个盘面来写，一个柱面写满后，才移到下一个扇区开始写数据。读数据也按照这种方式进行，这样就提高了硬盘的读/写效率。

一块硬盘驱动器的圆柱数（或每个盘面的磁道数）既取决于每条磁道的宽窄（同样，也与磁头的大小有关），也取决于定位机构所决定的磁道间步距的大小。

**4.扇区**

操作系统以扇区（Sector）形式将信息存储在硬盘上，每个扇区包括**512个字节**的数据和一些其他信息。一个扇区有两个主要部分：存储数据地点的标识符和存储数据的数据段，如图所示。



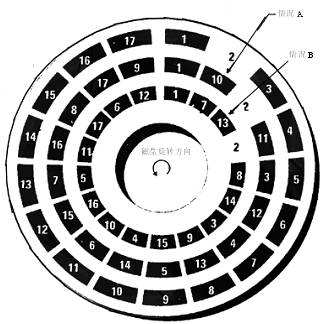
标识符就是扇区头标，包括组成扇区三维地址的三个数字：扇区所在的磁头（或盘面）、磁道（或柱面号）以及扇区在磁道上的位置即扇区号。头标中还包括一个字段，其中有显示扇区是否能可靠存储数据（不良扇区标识），或者是否已发现某个故障因而不宜使用的标记。有些硬盘控制器在扇区头标中还记录有指示字，可在原扇区出错时指引磁盘转到替换扇区或磁道。最后，扇区头标以循环冗余校验（CRC：Cyclic Redundancy Check）值作为结束，以供控制器检验扇区头标的读出情况，确保准确无误。

扇区的第二个主要部分是存储数据的数据段，可分为数据和保护数据的纠错码（ECC）。在初始准备期间，计算机用512个虚拟信息字节（实际数据的存放地）和与这些虚拟信息字节相应的ECC数字填入这个部分。

扇区头标包含一个可识别磁道上该扇区的扇区号。有趣的是，这些扇区号物理上并不连续编号，它们不必用任何特定的顺序指定。扇区头标的设计允许扇区号可以从1到某个最大值，某些情况下可达255。磁盘控制器并不关心上述范围中什么编号安排在哪一个扇区头标中。在很特殊的情况下，扇区还可以共用相同的编号。磁盘控制器甚至根本就不管数据区有多大，只管读出它所找到的数据，或者写入要求它写的数据。

给扇区编号的最简单方法是l，2，3，4，5，6等顺序编号。如果扇区按顺序绕着磁道依次编号，那么，控制器在处理一个扇区的数据期间，磁盘旋转太远，超过扇区间的间隔（这个间隔很小），控制器要读出或写入的下一扇区已经通过磁头，也许是相当大的一段距离。在这种情况下，磁盘控制器就只能等待磁盘再次旋转几乎一周，才能使得需要的扇区到达磁头下面。

显然，要解决这个问题，靠加大扇区间的间隔是不现实的，那会浪费许多磁盘空间。许多年前，IBM的一位杰出工程师想出了一个绝妙的办法，即**对扇区不使用顺序编号**，而是使用一个**交叉因子**（interleave）进行编号。交叉因子用比值的方法来表示，如3﹕1表示磁道上的第1个扇区为1号扇区，跳过两个扇区即第4个扇区为2号扇区，这个过程持续下去直到给每个物理扇区编上逻辑号为止。例如，每磁道有17个扇区的磁盘按2﹕1的交叉因子编号就是：l，10，2，11，3，12，4，13，5，14，6，15，7，16，8，17，9，而按3﹕1的交叉因子编号就是：l，7，13，2，8，14，3，9，15，4，10，16，5，11，17，6，12。当设置1﹕1的交叉因子时，如果硬盘控制器处理信息足够快，那么，读出磁道上的全部扇区只需要旋转一周；但如果硬盘控制器的后处理动作没有这么快，磁盘所转的圈数就等于一个磁道上的扇区数，才能读出每个磁道上的全部数据。将交叉因子设定为2﹕1时，磁头要读出磁道上的全部数据，磁盘只需转两周。如果2﹕1的交叉因子仍不够慢，磁盘旋转的周数约为磁道的扇区数，这时，可将交叉因子调整为3﹕1，如图所示。（磁盘是一直在旋转的，但是磁头读取数据到处理完，需要时间，如果扇区编号连续，很大可能会导致1号扇区读取数据处理完之后，磁盘已经转到3号扇区，如果要接着1号扇区，去读二号扇区，就必须要等磁盘在转一圈才行，交叉因子需要根据磁盘数据读取性能来做配置，可以提升磁盘读取效率）。



上图所示的是典型的MFM（Modified Frequency Modulation，改进型调频制编码）硬盘，每磁道有17个扇区，画出了用三种不同的扇区交叉因子编号的情况。最外圈的磁道（0号柱面）上的扇区用简单的顺序连续编号，相当于扇区交叉因子是1﹕1。1号磁道（柱面）的扇区按2﹕1的交叉因子编号，而2号磁道按3﹕1的扇区交叉因子编号。

早期的硬盘管理工作中，设置交叉因子需要用户自己完成。用BIOS中的低级格式化程序对硬盘进行低级格式化时，就需要指定交叉因子，有时还需要设置几种不同的值来比较其性能，而后确定一个比较好的值，以期硬盘的性能较好。现在的硬盘BIOS已经自己解决这个问题，所以，一般低级格式化程序不再提供这一选项设置。

系统将文件存储到磁盘上时，按柱面、磁头、扇区的方式进行，即最先是第1磁道的第一磁头下（也就是第1盘面的第一磁道）的所有扇区，然后，是同一柱面的下一磁头，……，一个柱面存储满后就推进到下一个柱面，直到把文件内容全部写入磁盘。系统也以相同的顺序读出数据。读出数据时通过告诉磁盘控制器要读出扇区所在的柱面号、磁头号和扇区号（物理地址的三个组成部分）进行。磁盘控制器则直接使磁头部件步进到相应的柱面，选通相应的磁头，等待要求的扇区移动到磁头下。在扇区到来时，磁盘控制器读出每个扇区的头标，把这些头标中的地址信息与期待检出的磁头和柱面号做比较（即寻道），然后，寻找要求的扇区号。待磁盘控制器找到该扇区头标时，根据其任务是写扇区还是读扇区，来决定是转换写电路，还是读出数据和尾部记录。找到扇区后，磁盘控制器必须在继续寻找下一个扇区之前对该扇区的信息进行后处理。如果是读数据，控制器计算此数据的ECC码，然后，把ECC码与已记录的ECC码相比较。如果是写数据，控制器计算出此数据的ECC码，与数据一起存储。在控制器对此扇区中的数据进行必要处理期间，磁盘继续旋转。由于对信息的后处理需要耗费一定的时间，在这段时间内，磁盘已转了相当的角度。

交叉因子的确定是一个系统级的问题。一个特定硬盘驱动器的交叉因子取决于：磁盘控制器的速度、主板的时钟速度、与控制器相连的输出总线的操作速度等。如果磁盘的交叉因子值太高，就需多花一些时间等待数据在磁盘上存入和读出。如果交叉因子值太低，就会大大降低磁盘性能。

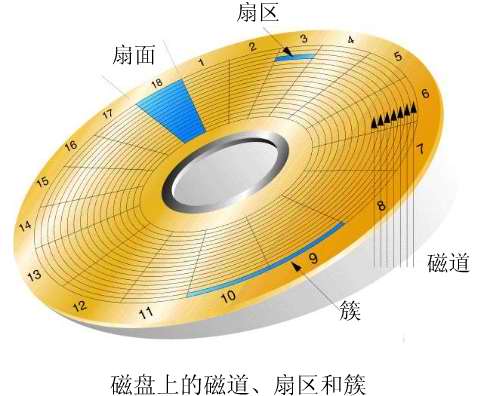
前面已经述及，系统在磁盘上写入信息时，写满一个磁道后转到同一柱面的下一个磁头，当柱面写满时，再转向下一柱面。从同一柱面的一个扇区到另一个扇区，从一个柱面转到下一个柱面，每一个转换都需要时间，在此期间磁盘始终保持旋转，这就会带来一个问题：假定系统刚刚结束对一个磁道前一个扇区的写入，并且已经设置了最佳交叉因子比值，现在准备在下一磁道的第一扇区写入，这时，必须等到磁头转换好，让磁头部件重新准备定位在下一道上。如果这种操作占用的时间超过了一点，尽管是交叉存取，磁头仍会延迟到达。这个问题的解决办法是以原先磁道所在位置为基准，把新的磁道上全部扇区号移动约一个或几个扇区位置，这就是［**磁头扭斜］**。磁头扭斜可以理解为柱面与柱面之间的交叉因子，已由生产厂设置好，用户一般不用去改变它。磁头扭斜的更改比较困难，但是，它们只在文件很长、超过磁道结尾进行读出和写入时才发挥作用，所以，扭斜设置不正确所带来的时间损失比采用不正确的扇区交叉因子值带来的损失要小得多。交叉因子和磁头扭斜可用专用工具软件来测试和更改。更具体的内容这里就不再详述，毕竟现在很多用户都没有见过这些参数。

扇区号存储在扇区头标中，扇区交叉因子和磁头扭斜的信息也存放在这里。最初，硬盘低级格式化程序只是行使有关磁盘控制器的专门职能来完成设置任务。由于这个过程可能破坏低级格式化的磁道上的全部数据，也极少采用。

扇区交叉因子由写入到扇区头标中的数字设定，所以，每个磁道可以有自己的交叉因子。在大多数驱动器中，所有磁道都有相同的交叉因子。但有时因为操作上的原因，也可能导致各磁道有不同的扇区交叉因子。如在交叉因子重置程序工作时，由于断电或人为中断，就会造成一些磁道的交叉因子发生了改变，而另一些磁道的交叉因子没有改变。这种不一致性对计算机不会产生不利影响，只是有最佳交叉因子的磁道要比其他磁道的工作速度更快。

**5.簇**

将物理相邻的若干个扇区称为了一个簇。操作系统读写磁盘的基本单位是扇区，而文件系统的基本单位是簇(Cluster)。在Windows下，随便找个几字节的文件，在其上面点击鼠标右键选择属性，看看实际大小与占用空间两项内容，如大小：15 字节 (15 字节)， 占用空间：4.00 KB (4，096 字节)。这里的占用空间就是你机器分区的簇大小，因为再小的文件都会占用空间，逻辑基本单位是4K，所以都会占用4K。 簇一般有这几类大小 4K，8K，16K，32K，64K等。簇越大存储性能越好，但空间浪费严重。簇越小性能相对越低，但空间利用率高。NTFS格式的文件系统簇的大小为4K。



**硬盘读写数据的过程**

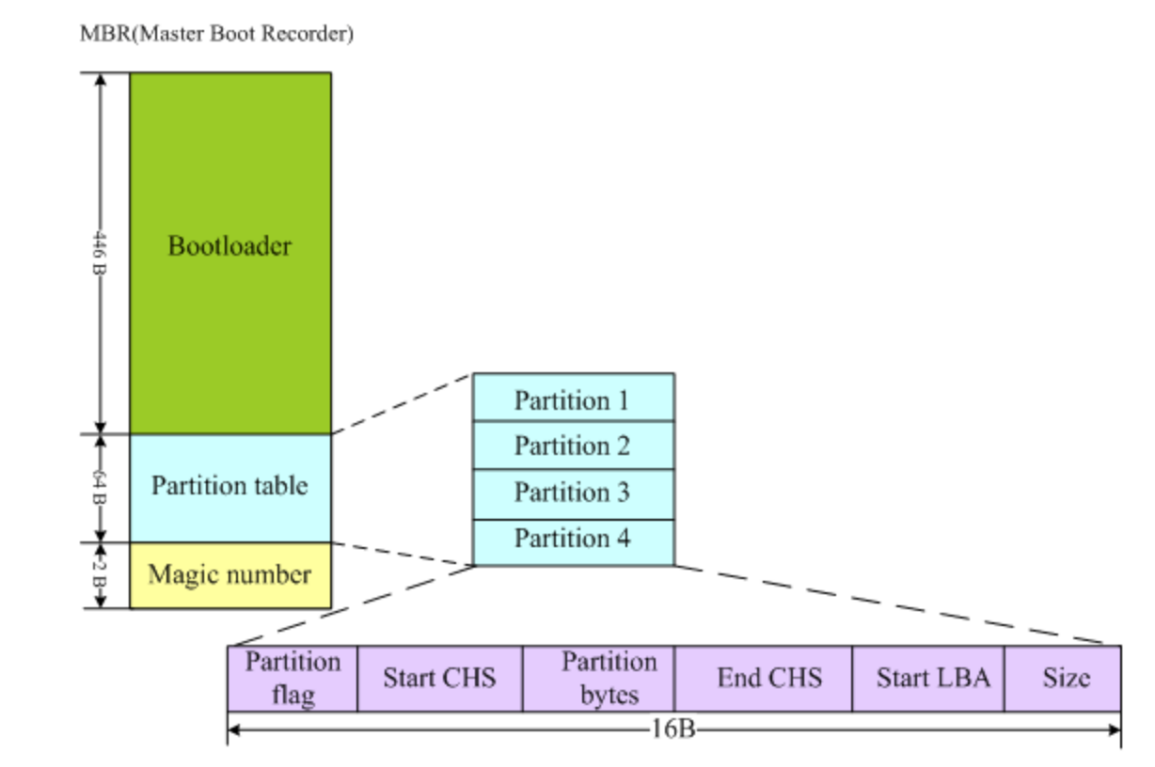
现代硬盘寻道都是采用CHS(Cylinder Head Sector)的方式，硬盘读取数据时，读写磁头沿径向移动，移到要读取的扇区所在磁道的上方，这段时间称为**寻道时间**(**seek time**)。因读写磁头的起始位置与目标位置之间的距离不同，寻道时间也不同。目前硬盘一般为2到30毫秒，平均约为9毫秒。磁头到达指定磁道后，然后通过盘片的旋转，使得要读取的扇区转到读写磁头的下方，这段时间称为**旋转延迟时间(rotational latencytime)**。

一个7200（转/每分钟）的硬盘，每旋转一周所需时间为60×1000÷7200=8.33毫秒，则平均旋转延迟时间为8.33÷2=4.17毫秒（平均情况下，需要旋转半圈）。平均寻道时间和平均选装延迟称为平均存取时间。

所以，最后看一下硬盘的容量计算公式：

硬盘容量=盘面数×柱面数×扇区数×512字节

在Linux启动过程中有一个**MBR**，它是存在于硬盘的0柱面，0磁头，1扇区里，占512字节的空间。这512字节里包含了主引导程序Bootloader和磁盘分区表DPT。其中Bootloader占446字节，分区表占64字节，一个分区要占用16字节，64字节的分区表只能被划分4个分区，这也就是目前我们的硬盘最多只能支持4个分区记录的原因。



即，如果你将硬盘分成4个主分区的话，必须确保所有的磁盘空间都被使用了(这不是废话么)，一般情况下我们都是划分一个主分区加一个扩展分区，然后在扩展分区里再继续划分逻辑分区。当然，逻辑分区表也需要分区表，它是存在于扩展分区的第一个扇区里，所以逻辑分区的个数最多也只能有512/16=32个，并不是想分多少个逻辑分区都可以。  
注意，我们所说的扩展分区也是要占用分区表项的。例如，如果我们的硬盘只划分一个主分区和一个逻辑分区，此时的分区表的排列如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Device Boot*** | ***Start*** | ***End*** | ***Blocks*** | ***Id*** | ***System*** |
| ***/dev/sda1*** | *1* | *19* | *152586* | *83* | *Linux* |
| ***/dev/sda2*** | *20* | *2569* | *20482875* | *83* | *Extended* |
| ***/dev/sda5*** | *2570* | *19457* | *4128705* | *82* | *Linux* |

主分区为1号分区，扩展分区占用了2号分区，3和4号扩展分区被预留了下来，逻辑分区从5开始编号依次递增，这里我们只划分了一个逻辑分区。

**磁盘读写数据所花费的时间**

在了解了硬盘的基本原理之后，不难推算出，磁盘上数据读取和写入所花费的时间可以分为三个部分。

* **寻道时间**

即磁臂移动到指定磁道所需要的时间，这部分时间又可以分为两部分：

寻道时间=启动磁臂的时间+常数\*所需移动的磁道数

其中常数和驱动器的的硬件相关，启动磁臂的时间也和驱动器的硬件相关

* **旋转延迟**

指的是把扇区移动到磁头下面的时间。这个时间和驱动器的转数有关，我们通常说的7200转的硬盘的转就是这个。

平均旋转延迟=1/(2\*转数每秒)

比如7200转的硬盘的平均旋转延迟等于1/2\*120≈4.17ms

旋转延迟只和硬件有关。

* **传输时间**

传输时间指的是从磁盘读出或将数据写入磁盘的时间。

这个时间等于：所需要读写的字节数/每秒转速\*每扇区的字节数

**磁盘调度算法**

通过上面硬盘读写数据所分的三部分时间不难看出，大部分参数是和硬件相关的，操作系统无力优化。只有所需移动的磁道数是可以通过操作系统来进行控制的，所以减少所需移动的磁道数是减少整个硬盘的读写时间的唯一办法。

因为操作系统内可能会有很多进程需要调用磁盘进行读写，因此合理的安排磁头的移动以减少寻道时间就是磁盘调度算法的目的所在，几种常见的磁盘调度算法如下。

* **先来先服务算法(FCFS)：**这种算法将对磁盘的IO请求进行排队，按照先后顺序依次调度磁头。这种算法的特点是简单，合理，但没有减少寻道时间
* **最短寻道时间算法(SSFT)：**这种算法优先执行所需读写的磁道离当前磁头最近的请求。这保证了平均寻道时间的最短，但缺点显而易见：离当前磁头比较远的寻道请求有可能一直得不到执行，这也就是所谓的“饥饿现象”。
* **扫描算法(SCAN)**：这种算法在磁头的移动方向上选择离当前磁头所在磁道最近的请求作为下一次服务对象，这种改进有效避免了饥饿现象，并且减少了寻道时间。但缺点依然存在，那就是不利于最远一端的磁道访问请求。
* **循环扫描算法(CSCAN)：**也就是俗称的电梯算法，这种算法是对最短寻道时间算法的改进。这种算法就像电梯一样，只能从1楼上到15楼，然后再从15楼下到1楼。这种算法的磁头调度也是如此,磁头只能从最里磁道到磁盘最外层磁道。然后再由最外层磁道移动到最里层磁道，磁头是单向移动的，在此基础上，才执行和最短寻道时间算法一样的，离当前磁头最近的寻道请求。这种算法改善了SCAN算法，消除了对两端磁道请求的不公平。

**局部性原理**

所谓的局部性原理分为时间和空间上的。由于程序是顺序执行的，因此当前数据段附近的数据有可能在接下来的时间被访问到。这就是所谓的空间局部性。而程序中还存在着循环，因此当前被访问的数据有可能在短时间内被再次访问，这就是所谓的时间局部性原理。

因此在了解了局部性原理之后，我们可以通过以下几个手段来减少磁盘的IO。

* **提前读(Read-Ahead)**

提前读也被称为预读。根据磁盘原理我们不难看出，在磁盘读取数据的过程中，真正读取数据的时间只占了很小一部分，而大部分时间花在了旋转延迟和寻道时间上，因此根据空间局部性原理，MySql每次读取数据的时间不仅仅读取所需要的数据，还将所请求数据附近的数据进行读取。这在MySql中被称为预读。MySql通过预读可以有效的减少IO请求。

* **延迟写(Delayed write)**

同样，根据时间局部性原理，最近被访问的数据有可能再次被访问，因此当数据更改之后不马上写回磁盘，而是继续放在内存中，以备接下来的请求读取或者修改，是减少磁盘IO的另一个有效手段，在MySql中，实现延迟写是buffer pool,当一个修改请求被commit之后，并不会立刻写回磁盘，而是将修改的页标记为“脏”，然后根据某种机制通过checkpoint或lazy writer写回磁盘（如果数据未写入硬盘之前，直接kill进程会导致内存中的数据丢失）

* **优化物理分布 <<<!!!**

根据磁盘原理不难看出，如果所请求的数据在磁盘物理磁道之间是连续的，那么会减少磁头的移动距离，从而减少了寻道时间。因此相关的数据放在连续的物理空间上会减少寻道时间。MySql中，通过**聚簇索引**使得数据根据主键在物理磁盘上连续，从而减少了寻道时间。

——————————————————————————————————————————————————————－

附录-闪存：Flash存储器

分为NAND Flash (=NON AND 与非) 和 NOR Flash (=NOT OR 或非) 两种主要的非易失闪存技术。只是用来存储少量的代码并且需要多次擦写时，NOR闪存更适合一些。而NAND则是高数据存储密度的理想解决方案。

NAND 与 NOR 性能比较

Flash闪存是非易失存储器，可以对称为**块**的存储器单元块进行擦写和再编程。任何Flash器件的写入操作只能在空或已擦除的单元内进行，所以大多数情况下，在进行写入操作之前必须先执行擦除。NAND器件执行擦除操作是十分简单的，而NOR则要求在进行擦除前先要将目标块内所有的位都写为0。

由于擦除NOR器件时是以64～128KB的块进行的，执行一个写入/擦除操作的时间为5s，与此相反，擦除NAND器件是以8～32KB的块进行的，执行相同的操作最多只需要4ms。

执行擦除时块尺寸的不同进一步拉大了NOR和NAND之间的性能差距，统计表明，对于给定的一套写入操作（尤其是更新小文件时），更多的擦除操作必须在基于NOR的单元中进行。当选择存储解决方案时，设计师必须权衡以下的各项因素：

* NOR的读速度比NAND稍快一些。
* NAND的写入速度比NOR快很多。
* NAND的擦除速度远比NOR快。
* NAND的擦除单元更小，相应的擦除电路更加简单。
* NAND的实际应用方式要比NOR复杂的多。
* NOR可以直接使用，并在上面直接运行代码，而NAND需要I/O接口，因此使用时需要驱动

NAND 与 NOR接口差别

NOR Flash带有SRAM接口，有足够的地址引脚来寻址，可以很容易地存取其内部的每一个字节。

NAND器件使用复杂的I/O口来串行地存取数据，各个产品或厂商的方法可能各不相同。8个引脚用来传送控制、地址和数据信息。

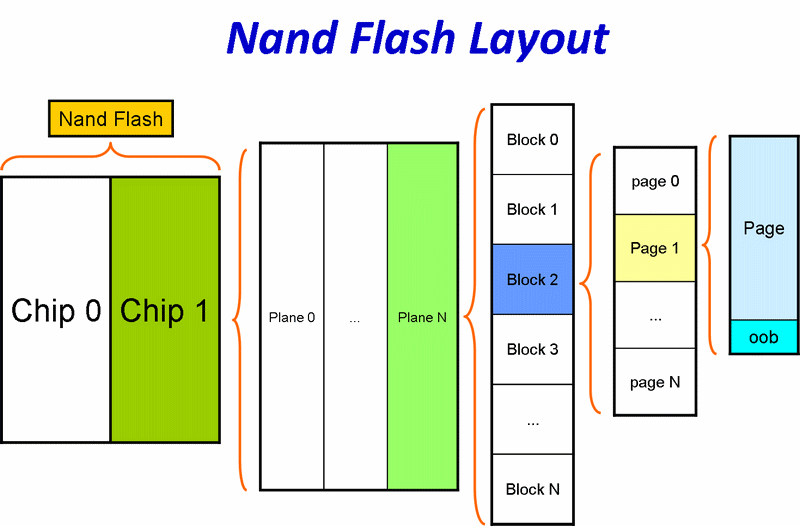
NAND读和写操作采用512字节的块，这一点有点像硬盘管理此类操作，很自然地，基于NAND的存储器就可以取代硬盘或其他块设备。NOR的特点是芯片内执行(XIP, eXecute In Place),这样应用程序可以直接在Flash闪存内运行,不必再把代码读到系统RAM中。

NOR的传输效率很高,在1～4MB的小容量时具有很高的成本效益,但是很低的写入和擦除速度大大影响了它的性能。

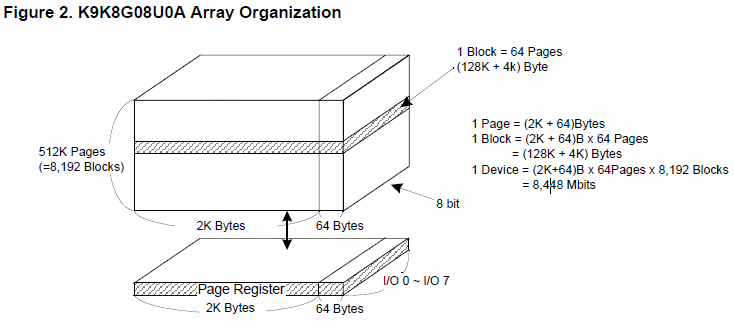
NAND结构能提供极高的单元密度,可以达到高存储密度,并且写入和擦除的速度也很快。应用NAND的困难在于Flash的管理需要特殊的系统接口。

物理构成

<https://www.cnblogs.com/shangdawei/archive/2013/05/14/3077018.html>



NAND Flash 的数据是以bit的方式保存在memory cell，一般来说，一个cell 中只能存储一个bit。这些cell 以8个或者16个为单位，连成bit line，形成所谓的byte(x8)/word(x16)，这就是NAND Device的位宽。这些Line会再组成Page，(NAND Flash 有多种结构，我使用的NAND Flash 是K9F1208,下面内容针对三星的K9F1208U0M)，每页528Bytes(512byte(Main Area)+16byte(Spare Area))，每32个page形成一个Block(32\*528B)。具体一片Flash上有多少个Block视需要所定。我所使用的三星k9f1208U0M具有4096个block，故总容量为4096\*（32\*528B）=66MB，但是其中的2MB是用来保存ECC校验码等额外数据的，故实际中可使用的为64MB。



NAND flash以页为单位读写数据，而以块为单位擦除数据。按照这样的组织方式可以形成所谓的三类地址：

Column Address：Starting Address of the Register. 翻成中文为列地址，地址的低8位

Page Address ：页地址

Block Address ：块地址

对于NAND Flash来讲，地址和命令只能在I/O[7:0]上传递，数据宽度是8位。

NAND Flash 的 SLC&MLC

存储单元分为两类：SLC（Single-Level Cell 单层单元）和MLC（Multi-Level Cell多层单元）。此外，SLC闪存的优点是复写次数高达100000次，比MLC闪存高10倍。此外，为了保证MLC的寿命，控制芯片都校验和智能磨损平衡技术算法，使得每个存储单元的写入次数可以平均分摊，达到100万小时故障间隔时间(MTBF：Mean Time Between Failures)。

存取原理上SLC架构是0和1两个充电值，即每Cell只能存取1bit数据，有点儿类似于开关电路，虽然简单但却非常稳定。如同电脑的CPU部件一样，要想在一定体积里容纳更多的晶体管数，就必须提高生产工艺水平，减小单晶体管体积。

而MLC架构可以一次 储存4个以上的充电值，因此拥有比较好的存储密度

SLC与MLC的区别：

* **存取次数**。MLC架构理论上只能承受约1万次的数据写入（非数据写入加读取的总次数），而SLC架构可承受约10万次，是MLC的10倍。（这里1W次指全量数据写入，类似于电池的全量充放电）。
* **读写速度**。这里仍存在认识上的误区，所有闪存芯片读取、写入或擦除数据都是在闪存控制芯片下完成的，闪存控制芯片的速度决定了闪存里数据的读取、擦除或是重新写入的速度。（闪存控制芯片效能低，且与闪存之间的兼容性不好，这类产品不仅速度慢而且在数据操作时出错的概率也大）
* **功耗**。SLC架构由于每Cell仅存放1bit数据，故只有高和低2种电平状态，使用1.8V的电压就可以驱动。而MLC架构每Cell需要存 放多个bit，即电平至少要被分为4档（存放2bit），所以需要有3.3V及以上的电压才能驱动。最近传来好消息，英特尔新推出的65纳米MLC写入速 度较以前产品提升了二倍，而工作电压仅为1.8V，并且凭借低功耗和深层关机模式，其电池使用时间也得到了延长
* **出错率**。在一次读写中SLC只有0或1两种状态，这种技术能提供快速的程序编程与读取，非常稳定，就算其中一个Cell损坏，对整体的性能也不会有影响。MLC有四种状态（以每Cell存取2bit为例）， 这就意味着MLC存储时要更精确地控制每个存储单元的充电电压，读写时就需要更长的充电时间来保证数据的可靠性。它已经不再是简单的开关电路，而是要控制四种不同的状态，这在产品的出错率方面和稳定性方面有较大要求，而且一旦出现错误，就会导致2倍及以上的数据损坏，所以MLC对制造工艺和控制芯片有着更高的要求。
* **制造成本**。为什么硬盘容量在成倍增大的同时生产成本却能保持不变，简单点说就是在同样面积的盘片上存储更多的数据，也就是所谓的存储密度增大了。MLC技术与之非常类似，原来每Cell仅存放1bit数据，而现在每Cell能存放2bit甚至更多数据，这些都是在存储体体积不增大的前提下实现的，所以相同容量大小的MLC制造成本要远低于SLC。

综上所述，MLC技术是今后NAND Flash的发展趋势，就像CPU单核心、双核心、四核心一样，MLC技术通过每Cell存储更多的bit来实 现容量上的成倍跨越，直至更先进的架构问世。而SLC短期内仍然会是市场的佼佼者，但随着MLC技术的不断发展和完善，SLC必将退出历史的舞台。