UNIX 的快速文件系统

马歇尔●K●麦克库克，威廉●N●乔伊，萨穆尔●J●勒夫勒和罗伯特●S●法布里
计算机系统研究组

描述了 UNIXTM 文件系统的重新实现。通过使用更灵活的分配策略，重新实现可以提供更高的吞吐率，该分配策略允许更好的引用位置，并且可以适应各种外围设备和处理器特性。新的文件系统对顺序访问的数据进行了群集，并提供了两种块大小，以允许快速访问大型文件，而不会浪费大量空间存储小型文件。遇到的文件访问速度比传统的 UNIX 文件系统快十倍。讨论了程序员界面长期需要的增强。其中包括一种在文件上放置咨询锁的机制，跨文件系统的名称空间扩展，使用长文件名的能力以及对资源使用进行管理控制的规定。

类别和主题描述符：D.4.3［操作系统］：文件系统管理-文件组织；目录结构；访问方法；D.4.2［操作系统］：存储管理分配/取消分配策略；辅助存储设备；D.4.8［操作系统］：性能评估；运营分析；H.3.2［信息系统］：信息存储文件的组织

通用术语：测量，性能

其他关键字和短语：UNIX，文件系统组织，文件系统性能，文件系统设计，应用程序接口

1.介绍

本文介绍了从原始 512 字节 UNIX [1] 文件系统到使用 4.2 Berkeley Software Distribution 发布的新系统的文件更改。它介绍了更改的动机，实现这些更改的方法，设计决策的依据以及对新实现的描述。在讨论之后，将总结所获得的结果，未来工作的方向以及对程序员可用的功能进行的添加和更改。

[1]UNIX 是 AT&T 贝尔实验室的商标。

这项工作是在美国国家科学基金会（NASDAQ：MCSSO-05144）和美国国防部高级研究计划局（DoD）根据美国海军电子系统司令监督的 ARPA 第 4031 号命令下进行的，合同号为 N00039-82-C-0235。
作者目前的地址：加利福尼亚大学伯克利分校电气工程和计算机科学系计算机科学系 MKMcKusick 和 RS Fabry，CA 94720；WN Joy， 太阳微系统公司， 2550 加西亚大道， 山景，CA 94043;SJ Leffler，Lucasfilm Ltd.，PO Box 2009，San Rafael，CA 94912。

ACM Transactions on Computer Systems，第一卷。2，第 3 号，1984 年 8 月，第 181-197 页。

在 PDP-11[2] 上运行的原始 UNIX 系统具有简单而优雅的文件系统功能。文件系统 I / O 由内核缓冲；数据传输没有对齐限制，并且所有操作都显示为同步。所有到磁盘的传输都以 512 字节的块为单位，可以将其任意放置在文件系统的数据区域内。实际上，除了可用磁盘空间之外，没有其他限制会影响文件的增长[14, 18]。[3]

当在 VAX-11 上与其他 UNIX 增强功能一起使用时，原始的 512 字节 UNIX 文件系统无法提供许多应用程序所需的数据吞吐率。例如，诸如超大规模集成（VLSI）设计和图像处理之类的应用程序需要对大量数据进行少量处理，并且需要具有来自文件系统的高吞吐量。将文件从文件系统映射到大型虚拟地址空间的程序也需要高吞吐率。在文件系统中分页和分页数据可能经常发生[5]。这就要求文件系统提供比原始 512 字节 UNIX 更高的带宽，而该 UNIX 只能提供最大磁盘带宽的大约 2%或每臂每秒 20 KB 的速度[21, 16]。

已对原始 UNIX 文件系统进行了修改，以提高其性能。由于 UNIX 文件系统接口已广为人知，并且固有速度并不慢，因此此开发保留了抽象，只是更改了底层实现以提高其吞吐量。因此，系统的用户并未面临大规模的软件转换。

文件系统性能方面的问题已在文献中得到了广泛解决。参见[15]进行调查。Ferrin [4]已经完成了提高 UNIX 文件系统性能的先前工作。UNIX 操作系统从 Multics（大型，高性能操作系统）[3]中汲取了许多想法。其他工作包括 Hydra [1]，Spice [19]和用于 LISP 环境的文件系统[17]。在[11]中对磁盘的物理延迟进行了很好的介绍。

2. 旧文件系统

在贝尔实验室开发的文件系统（“传统”文件系统）中，每个磁盘驱动器都分为一个或多个分区。这些磁盘分区中的每个分区都可以包含一个文件系统。文件系统从不跨越多个分区。[4] 文件系统由其超级块描述，该超级块包含文件系统的基本参数。这些包括文件系统中数据块的数量，最大文件数量的计数，指向空闲列表的指针，文件系统中所有空闲块的链接列表。

文件系统中包含文件。某些文件被区分为目录，并且包含指向可能本身就是目录的文件的指针。每个文件都有一个

---

[2] DEC，PDP，VAX，MASSBUS 和 UNIBUS 是 Digital Equipment Corporation 的商标。

[3] 实际上，文件的大小被限制为小于大约 1 GB。

[4] 在这里，“分区”是指磁盘驱动器上物理空间的细分。在传统文件系统中，就像在新文件系统中一样，文件系统实际上位于可能重叠的逻辑磁盘分区中。例如，可以使用这种重叠方式来允许程序复制包含多个文件系统的整个磁盘驱动器。

与之关联的描述符称为索引节点。索引节点包含描述文件所有权的信息，标记文件的最后修改和访问时间的时间戳以及指向文件数据块的索引数组。在本节中，我们假设文件的前 8 个块直接由存储在 inode 本身中的值引用。[5]inode 也可能包含对包含更多数据块索引的间接块的引用。在具有 512 字节块大小的文件系统中，单个间接块包含另外 128 个块地址，双重间接块包含另外 128 个间接块的地址，三重间接块包含 128 个其他间接块的地址。

一个 150 MB 的传统 UNIX 文件系统由 4 MB 的 inode 和 146 MB 的数据组成。该组织将 inode 信息与数据分开；因此，访问文件通常会导致从文件的索引节点到其数据的漫长查找。在单个目录中的文件通常不会在 4 兆字节的 inode 中分配连续的插槽，从而导致在对目录中多个文件的 inode 执行操作时访问许多非连续的 inode 块。

数据块到文件的分配也是次优的。传统的文件系统每次磁盘事务传输的数据量永远不会超过 512 字节，并且经常发现下一个顺序数据块不在同一柱面上，因此会在 512 字节传输之间进行强制查找。小块大小，系统中有限的预读以及许多寻求的结合严重限制了文件系统的吞吐量。Berkeley 在 UNIX 文件系统上的第一项工作试图同时提高可靠性和吞吐量。通过对关键文件系统信息进行修改来提高可靠性，从而使崩溃后程序可以完全完成或对其进行彻底修复[6]。通过将基本块大小从 512 字节更改为 1024 字节，文件系统性能提高了两倍以上。增长的原因有两个：每次磁盘传输访问的数据量是原来的两倍，并且大多数文件都可以在不需要访问间接块的情况下进行描述，因为直接块包含的数据量是原来的两倍。从此以后，具有这些更改的文件系统将称为旧文件系统。性能的提高强烈表明增加块大小是提高吞吐量的好方法。尽管吞吐量增加了一倍，但是旧文件系统仍仅使用磁盘带宽的百分之四。主要的问题是，尽管最初对空闲列表进行了排序以实现最佳访问，但是随着创建和删除文件后，空闲列表迅速变得混乱。最终，空闲列表变得完全随机，从而导致文件在磁盘上随机分配其块。这迫使在每次访问块之前先进行查找。尽管旧文件系统在初次创建时提供的传输速率最高为每秒 175 KB，但由于数据块放置的这种随机化，经过数周的合理使用后，该传输速率已降至每秒 30 KB。除了转储，重建和还原文件系统外，无法恢复旧文件系统的性能。正如丸山[9]所建议的那样，另一种可能的方法是定期重组磁盘上的数据以恢复本地性。

[5]实际数字可能因系统而异，但通常在 5-13 范围内。

3. 新文件系统组织

在新的文件系统组织中（与旧的文件系统组织中一样），每个磁盘驱动器包含一个或多个文件系统。文件系统由其超级块描述，该文件位于文件系统磁盘分区的开头。由于超级块包含关键数据，因此可以对其进行复制以防止催化性损失。这是在创建文件系统时完成的。由于超级块数据不会更改，因此不需要引用副本，除非磁头崩溃或其他硬盘错误导致默认超级块不可用。

为了确保仅使用两个间接级别就可以创建高达 2^{32} 字节的文件，文件系统块的最小大小为 4096 字节。文件系统块的大小可以是大于或等于 4096 的 2 的任意幂。文件系统的块大小记录在文件系统的超级块中，因此可以在磁盘上同时访问具有不同块大小的文件系统。同一系统。块大小必须在创建文件系统时确定。如果不重新构建文件系统，则无法更改它。

新的文件系统组织将磁盘分区划分为一个或多个称为柱面组的区域。圆柱组由磁盘上的一个或多个连续的圆柱组成。与每个柱面组相关联的是一些簿记信息，其中包括超级块的冗余副本，用于索引节点的空间，描述柱面组中可用 bloc.ks 的位图以及描述柱面组内数据块使用情况的摘要信息。柱面组中可用块的位图替换了传统文件系统的空闲列表。对于每个柱面组，在文件系统创建时都会分配一个静态的 inode 数。默认策略是为柱面组中的每个 2048 字节空间分配一个索引节点，并期望此节点数将远远超过所需量。

所有气缸组簿记信息都可以放置在每个气缸组的开头。但是，如果使用此方法，则所有冗余信息都将放在最前面。破坏顶部磁盘的单个硬件故障可能会导致超级块的所有冗余副本丢失。因此，汽缸组簿记信息从汽缸组开始处以变化的偏移量开始。计算每个连续汽缸组的偏移量，使汽缸组的偏移量比前一个汽缸组的起始位置大约偏移一个磁道。通过这种方式，冗余信息会螺旋下降到数据包中，从而可以丢失任何单个轨道，圆柱体或拼盘而不会丢失超级块的所有副本。除第一个柱面组外，柱面组起点与柱面组信息起点之间的间隔用于数据块。[6]

---

[6] 虽然看起来第一个柱面组可以使用其超级块放置在“已知”位置，但是这对于块大小为 16 KB 或更大的文件系统不起作用。这是因为要求磁盘的前 8 KB 保留用于引导程序，并且另外要求柱面组信息必须在文件系统块边界上开始。要在文件系统块边界上启动柱面组，块大小大于 8 KB 的文件系统必须在引导块末尾与柱面组开始之间留一个空白。在不知道文件系统块大小的情况下，系统将不知道要使用哪个舍入函数来查找第一个柱面组的开始。

表 I. 浪费空间量与块大小的关系

已用空间 (兆字	% 浪费	组织
775.2	0.0	仅数据，文件之间没有分隔
807.8	4.2	仅数据，每个文件均以 512 字节边界开始
828.7	6.9	数据+索引节点，512 字节块 UNIX 文件系统
866.5	11.8	数据+索引节点，1024 字节块 UNIX 文件系统
948.5	22.4	数据+索引节点，2048 字节块 UNIX 文件系统
1128.3	45.6	数据+索引节点，4096 字节的块 UNIX 文件系统

3.1. .优化存储利用率

对数据进行布局，以便可以在单个磁盘事务中传输更大的块，从而大大提高了文件系统的吞吐量。例如，请考虑新文件系统中由 4096 字节数据块组成的文件。在旧文件系统中，此文件将由 1024 字节的块组成。通过增加块大小，每个文件事务中新文件系统中的磁盘访问最多可以传输四倍的信息。在大文件中，可以从同一柱面分配几个 4096 字节的块，以便在需要查找之前甚至可以进行更大的数据传输。

较大块的主要问题是大多数 UNIX 文件系统由许多小文件组成。统一的大块大小会浪费空间。表 I 显示了文件系统块大小对文件系统中浪费的空间量的影响。为获得这些数字而测量的文件位于我们的一个分时系统中，该系统具有大约 1.2 GB 的在线存储。这些度量基于包含约 920 MB 格式化空间的活动用户文件系统。浪费的空间计算为磁盘上不包含用户数据的空间的百分比。随着磁盘上块大小的增加，浪费迅速增加，对于 4096 字节的文件系统块，浪费达到了无法忍受的 45.6%。

为了能够使用大块而不会造成不必要的浪费，必须以更有效的方式存储小文件。新文件系统通过将单个文件系统块划分为一个或多个片段来实现此目标。文件系统片段大小是在创建文件系统时指定的；每个文件系统块都可以分为 2、4 或 8 个片段，每个片段都是可寻址的。这些片段大小的下限受磁盘扇区大小（通常为 512 字节）的约束。与每个圆柱组关联的块图记录了碎片级别上圆柱组中的可用空间。为了确定块是否可用，将检查对齐的片段。图 1 显示了一个来自 4096/1024 文件系统的地图。

映射中的每一位都记录了片段的状态；“ X”表示该片段正在使用中，而“ 0”表示该片段可供分配。在此示例中，片段 0-5、10 和 11 正在使用，而片段 6-9 和 12-15 是空闲的。相邻块的片段即使足够大也不能用作完整块。在此示例中，片段 6-9 无法分配为完整块。只有片段 12-15 可以合并为一个完整的块。

在块大小为 4096 字节，片段大小为 1024 字节的文件系统上，文件由零个或多个 4096 字节的数据块表示，并且可能

映射中的位片段	XXXX	00	OOXX	OOOO
编号块编号	0-3	4-7	8-11	12-15
	0	1	2	3

图 1 在 4096/1024 文件系统中块和片段的布局示例。

单个碎片块。如果必须对文件系统块进行分段以获取少量数据的空间，则可以使该块的其余分段可用于分配给其他文件。例如，考虑一个存储在 4096/1024 字节文件系统中的 11,000 字节文件。该文件将使用两个完整大小的块以及另一个块的一个三片段部分。如果在创建文件时没有包含三个对齐片段的块，那么将拆分一个完整大小的块，以产生必要的片段和一个未使用的片段。可以根据需要将此剩余的片段分配给另一个文件。

程序执行写系统调用时，将空间分配给文件。每一次数据写入文件后，系统将检查文件大小是否已增加。[7] 如果需要扩展文件以容纳新数据，则存在以下三种情况之一：

(1) 在已分配的块或片段中有足够的空间来容纳新数据。新数据将写入可用空间。

(2) 该文件不包含零散的块（并且文件中的最后一个块包含的空间不足以容纳新数据）。如果在已分配的块中存在空间，则该空间将填充新数据。如果新数据的其余部分包含多于一个完整的数据块，则会分配一个完整的块，并在其中写入新数据的第一个完整的块。重复此过程，直到剩余的新数据块不足为止。如果要写入的剩余新数据少于一个完整的块，则找到具有必要片段的块；否则，将找到一个完整的块。其余的新数据将被写入所定位的空间。

(3) 该文件包含一个或多个片段（这些片段的空间不足以容纳新数据）。如果新数据的大小加上片段中已有数据的大小超过了整个块的大小，则会分配一个新块。片段的内容被复制到块的开始，块的其余部分被新数据填充。然后，该过程如上述（2）所述继续进行。否则，如果要写入的新数据少于一个完整的块，则将找到具有必要片段的块或一个完整的块。附加了新数据的现有片段的内容将写入分配的空间。

一次将文件扩展为一个片段的问题在于，随着片段块扩展为完整块，数据可能会被复制多次。如果用户程序一次写入一个完整的块（文件末尾的部分块除外），则可以最大程度地减少片段重新分配。由于具有不同块大小的文件系统可能位于同一系统上，因此文件系统接口已

[7] 程序可能正在覆盖现有文件的中间数据，在这种情况下，空间已经被分配。

扩展为应用程序提供最佳大小的读取或写入。对于文件，最佳大小是在其上访问文件的文件系统的块大小。对于其他对象，例如管道和套接字，最佳大小是基础缓冲区大小。标准输入/输出库（大多数用户程序使用的软件包）使用此功能。某些系统实用程序（例如，归档程序和装载程序）也使用此功能，它们进行自己的输入和输出管理，并且需要尽可能高的文件系统带宽。

根据经验，在 4096/1024 字节的新文件系统组织中浪费的空间数量与在 1024 字节的旧文件系统组织中浪费的空间大致相同。具有 4096 字节块和 512 字节片段的文件系统具有与 512 字节块 UNIX 文件系统相同的浪费空间。新文件系统使用的空间比 512 字节或 1024 字节的文件系统少，用于为大文件建立索引信息，而为小文件占用相同的空间。这些节省被需要使用更多空间来跟踪可用空闲块的需求所抵消。当新文件系统的碎片大小等于旧文件系统的块大小时，最终结果大约是相同的磁盘利用率。

为了使布局策略有效，文件系统不能保持完全满。对于每个文件系统，都有一个称为可用空间保留的参数，该参数给出了应该释放的文件系统块的最小可接受百分比。如果可用块的数量降至此级别以下，则只有系统管理员才能继续分配块。即使安装了文件系统并处于活动状态，此参数的值也可以随时更改。在文件系统中测量的第 4 节中显示的传输速率保持在不足 90%的已满状态（保留 10%）。如果可用块的数量降至零，则由于文件系统无法在文件中本地定位文件，文件系统吞吐量往往会减少一半。如果文件系统的性能由于过度填充而降低，则可以通过删除文件来恢复它，直到可用空间量再次达到最小可接受水平为止。一旦有足够的可用空间，就可以通过移动其数据来恢复在可用空间很少的情况下创建的文件的访问速率。在比较表 I 中提供的组织时，必须将可用空间保留添加到浪费的百分比中。因此，旧的 1024 字节 UNIX 文件系统中的浪费百分比与具有以下功能的新 4096/512 字节文件系统大致相当：可用空间设置为 5%。（与旧文件系统相比，浪费了 11.8%，在新文件系统中浪费了 6.9%，再加上 5%的保留空间。）

3.2. 文件系统参数化

除了最初创建空闲列表以外，旧文件系统会忽略底层硬件的参数。它没有有关大容量存储设备或与之交互的硬件的物理特性的信息。新文件系统的目标是参数化处理器功能和大容量存储特性，以便可以以最佳配置相关方式分配块。使用的参数包括处理器的速度，对大容量存储传输的硬件支持以及大容量存储设备的特性。磁盘技术在不断改进，并且给定的安装可以在一个单一的设备上运行几种不同的磁盘技术

处理器。每个文件系统都经过参数设置，因此可以适应放置它的磁盘的特性。

对于大容量存储设备（例如磁盘），新文件系统尝试在与同一文件中的前一个块相同的柱面上分配新块。最佳地，这些新块也将在旋转方向上定位良好。“旋转最佳”块之间的距离变化很大；根据系统特性，它可以是连续的块，也可以是旋转延迟的块。在具有 I / O 通道且在大容量存储传输请求之间不需要任何处理器干预的处理器上，由于连续的磁盘旋转，通常可以访问两个连续的磁盘块而不会浪费时间。对于没有 I / O 通道的处理器，主处理器必须发出中断并准备进行新的磁盘传输。服务该中断和安排新磁盘传输的预期时间取决于主处理器的速度。

每个磁盘的物理特性包括每个磁道的块数和磁盘旋转的速率。分配例程使用此信息来计算跳过一个块所需的毫秒数。处理器的特性包括服务中断和计划新磁盘传输的预期时间。给定分配给文件的块，分配例程将计算要跳过的块数，以使文件中的下一个块在开始新磁盘传输所需的预期时间量内进入磁盘头下方。操作。对于顺序访问大量数据的程序，此策略将等待磁盘放置自身的时间减至最少。

为了简化寻找最佳旋转块的计算，汽缸组摘要信息包括汽缸组中不同旋转位置处可用块的计数。区分了八个旋转位置，因此对于典型的每分钟 3600 转驱动器，摘要信息的分辨率为 2 毫秒。超级块包含一个称为旋转布局表的列表向量。向量由旋转位置索引。向量的每个分量针对其旋转位置中包含的每个数据块，在块图中列出索引。当寻找可分配的程序段时，系统首先通过汇总计数查找具有非零程序段计数的旋转位置。然后，它使用旋转位置的索引来找到适当的列表，以用于仅对块图的相关部分进行索引以找到空闲块。

即使在挂载并激活文件系统的情况下，也可以随时更改用于定义在同一柱面上完成数据传输与启动另一个数据传输之间的最小毫秒数的参数。如果将文件系统的参数设置为以 2 毫秒的旋转间隔布置块，然后将磁盘组移至具有处理器需要 4 毫秒来安排磁盘操作的系统，则吞吐量将由于磁盘丢失而急剧下降几乎每个街区都有转数。如果最终的目标机器是已知的，则即使文件系统最初是在其他处理器上创建的，也可以为其参数化。即使事先不知道移动情况，也可以在移动磁盘后重新配置旋转布局延迟，以便根据新主机的特性完成所有进一步的分配。

3.3 布局政策

文件系统布局策略分为两个不同的部分。顶级策略是全局策略，这些策略使用文件系统范围的摘要信息来做出有关新 inode 和数据块的放置的决策。这些例程负责确定新目录和文件的位置。它们还列出了旋转最优的块布局，并决定何时对新的气缸组强制进行长时间搜索，因为当前的气缸组中剩余的块不足以进行合理的布局。全局策略例程下面是使用本地最佳方案对数据块进行布局的本地分配例程。

改善文件系统性能的两种方法是增加引用的局部性，如 Trivedi [20]所述，以最大程度地减少查找等待时间，以及改善数据的布局，以使 Nevalainen [10]所述的更大的传输成为可能。全局布局策略试图通过聚集相关信息来提高性能。他们无法尝试本地化所有数据引用，但还必须尝试在不同的气缸组之间散布不相关的数据。如果尝试进行过多的本地化，则本地柱面组可能会用完空间，从而迫使数据分散到非本地柱面组。极端地讲，完全本地化会导致单个庞大的数据集群，类似于旧文件系统。全局策略试图平衡在分散无关数据的同时对同时访问的数据进行本地化的两个冲突目标。

一种可分配资源是 inode。索引节点用于描述文件和目录。同一目录中文件的 inode 经常一起访问。例如，“列表目录”命令通常访问目录中每个文件的索引节点。布局策略尝试将所有文件的 inode 放在同一柱面组的目录中。为确保文件分布在整个磁盘上，目录分配使用不同的策略。新目录放置在圆柱组中，该组的空闲 inode 数量大于平均数量，并且其中已有的目录数量最少。此策略的目的是使 inode 群集策略在大多数情况下都能成功。使用 next-free 策略完成圆柱组内 inode 的分配。尽管这会在圆柱组内随机分配索引节点，但是可以通过 8 到 16 个磁盘传输来读取特定圆柱组的所有索引节点。（由于一个柱面组最多可以包含 2048 个 inode，因此最多需要进行 16 个磁盘传输。）这为访问目录中所有文件的 inode 所需的磁盘传输数设置了一个小而恒定的上限。相反，旧文件系统通常需要进行一次磁盘传输才能为目录中的每个文件获取索引节点。

另一个主要资源是数据块。由于通常一起访问文件的数据块，因此策略例程会尝试将文件的所有数据块放在同一柱面组中，最好放在同一柱面中旋转最佳的位置。在同一柱面组中分配所有数据块的问题在于，大文件将很快用完柱面组中的可用空间，从而迫使溢出到其他区域。此外，使用柱面组中的所有空间会使柱面组中任何文件的将来分配也溢出到其他区域。理想情况下，所有气缸组都不应该完全充满。选择的启发式解决方案是将块分配重定向到其他

文件超过 48 KB 时（其后每兆字节）存储的柱面组。[8] 新选择的柱面组是从剩余可用块数大于平均数量的那些柱面组中选择的。尽管大文件倾向于散布在磁盘上，但是通常必须先执行兆字节的数据，然后才能执行长查找，并且每兆字节一个长查找的成本很小。

全局策略例程调用具有特定块请求的本地分配例程。本地分配例程将始终分配所请求的块（如果空闲），否则它将分配所请求大小的空闲块，该空闲块在旋转上最接近所请求的块。如果全局布局策略具有完整的信息，则它们始终可以请求未使用的块，并且分配例程将简化为简单的簿记。但是，维护完整的信息非常昂贵。因此，全局布局策略的实施使用仅使用部分信息的试探法。如果请求的块不可用，则本地分配器将使用四级分配策略：

(1) 在旋转上使用最接近请求的块的下一个可用块。这里假设磁头切换时间为零。在不是这种情况的磁盘控制器上，可以在构建旋转布局表时考虑在磁盘盘片之间切换所需的时间。但是，这还没有尝试过。

(2) 如果同一气缸上没有可用块，请在同一气缸组内使用一个块。

(3) 如果该气缸组已满，请对气缸组编号进行二次哈希，以选择另一个气缸组以查找空闲块。

(4) 最后，如果哈希失败，请对所有圆柱组进行详尽搜索。

之所以使用二次哈希，是因为它可以在几乎完整的哈希表中找到未使用的时隙[8]。参数化的文件系统至少可保留 10%的可用空间，很少使用此策略。在不保留任何可用空间的情况下运行的文件系统通常只有很少的可用块，因此几乎所有分配都是随机的。在这种条件下使用的策略的最重要特征是该策略必须快速。

4. 表现

最终，上一节中描述的算法有效性的证明是新文件系统的长期性能。

我们的经验研究表明，inode 布局策略是有效的。在本身包含许多目录的大型目录上运行“列表目录”命令（以强制系统访问多个柱面组中的 inode）时，inode 的磁盘访问次数将减少两倍。对于仅包含文件的大型目录，i 的改进更加明显。在这些搜索中，可以将对 i 节点的磁盘访问量减少八分之一。

---

[8] 48 KB 的第一个溢出点是 4096 字节块文件系统上的文件首先需要单个间接块的点。这似乎是重定向块分配的自然的第一步。选择其他溢出点的目的是在文件使用了柱面组中大约 25%的数据块时强制重定向块分配。在观察日常使用的新文件系统时，启发式方法在最小化完全填充圆柱组的数量方面似乎很有效。

表伊拉。新旧 UNIX 文件系统的读取率

文件系统类型	处理器和总线已测量	速度(千字节	读取带宽%	% 中央处理器
旧 1024	750 /独轮车	29	29/983 3	11
全新 4096/1024	750 /独轮车	221	221/983 22	43
新 8192/1024	750 /独轮车	233	233/983 24	29
全新 4096/1024	750 /辆	466	466/983 47	73
新 8192/1024	750 /辆	466	466/983 47	54

表 11b。新旧 UNIX 文件系统的写入率

文件系统类型	处理器和总线已测量	速度(千字节	写带宽%	% 中央处理器
旧 1024	750 /独轮车	48	48/983 5	29
全新 4096/1024	750 /独轮车	142	142/983 14	43
新 8192/1024	750 /独轮车	215	215/983 22	46
全新 4096/1024	750 /辆	323	323/983 33	94
新 8192/1024	750 /辆	466	466/983 47	95

对于访问许多小文件的后台处理程序之类的程序，这是最令人鼓舞的，因为这些程序往往会淹没旧文件系统上的磁盘请求队列。

表 II 总结了新文件系统的测量吞吐量。关于这些测试的运行条件，需要做出一些评论。测试程序测量用户程序可以在文件上进行数据传输而无需对其进行任何处理的速率。这些程序必须读写足够的数据，以确保操作系统中的缓冲不会影响结果。它们也至少连续运行三遍；第一个使系统进入已知状态，第二个使系统稳定并可重复。所使用的测试及其结果在 Kridle [7]中进行了详细讨论。[9] 系统正在运行多用户，但在其他情况下则处于静止状态。CPU 或磁盘臂都没有争用。UNIBUS 和 MASSBUS 测试之间的唯一区别是控制器。所有测试均使用 AMPEX Capricorn 330 MB 温彻斯特磁盘。如表 II 所示，所有文件系统测试运行都是在 VAX 11/750 上进行的。在测量之前，所有文件系统已投入生产至少一个月。在所有测试中，执行了相同数量的系统调用。基本系统调用开销在测试总运行时间中所占的比例很小。

与旧文件系统不同，新文件系统的传输速率似乎不会随时间变化。吞吐率与保持的可用空间量紧密相关。表 II 中的测量基于具有 10%可用空间保留的文件系统。综合工作负载表明，当文件系统已满时，吞吐量下降到表 II 中给出的速率的一半左右。

表 II 中给出的带宽百分比是文件系统对磁盘有效利用的度量。传输速率的上限

[9] 与我们使用的读取测试类似的 UNIX 命令是“ cp file / dev / null”，其中“ file”的长度为 8 MB。

通过将磁道上的字节数乘以得出磁盘上的数据。磁盘每秒的转数。通过比较文件系统能够达到的数据速率（以该速率的百分比）来计算带宽。使用此度量标准，旧文件系统只能使用大约 3-5%的磁盘带宽，而新文件系统最多可以使用 47%的带宽。

在新系统中，读写速度都比旧系统快。加速的最大因素是新文件系统使用的更大的块大小。在新系统中分配块的开销大于在旧系统中分配块的开销，但是在新系统中分配的块较少，因为它们更大。最终结果是，两个系统分配的每字节成本几乎相同。

在新文件系统中，读取速率始终至少与写入速率一样快。这是可以预料的，因为内核在分配块时必须比单纯读取它们做更多的工作。请注意，写入速率与 8192 字节块文件系统中的读取速率大约相同。写速率比 4096 字节的块文件系统中的读速率慢。发生写入速度较慢的原因是，内核每秒必须完成两倍的磁盘分配，从而使处理器无法跟上磁盘传输速度。

相反，旧文件系统在写入文件时比读取文件快约 50%。这是因为写系统调用是异步的，并且内核可以更快地生成磁盘传输请求，因此无法将其处理，因此磁盘传输在磁盘缓冲区高速缓存中排队。因为磁盘缓冲区高速缓存是按最小查找距离排序的，所以计划的磁盘写操作之间的平均查找要远远少于如果按照生成数据块的随机磁盘顺序写出数据块的情况。但是，在读取文件时，已读取的系统调用将被同步处理，因此必须按照请求磁盘块的非最佳查找顺序从磁盘中检索磁盘块。这迫使磁盘调度程序进行长时间查找，从而导致较低的吞吐率。

在新系统中，文件的块在磁盘上的排序更为理想。即使读取仍然是同步的，请求也将以更好的顺序呈现给磁盘。即使写操作仍然是异步的，它们也已经以最小查找顺序显示在磁盘上，因此重新排序不会有任何好处。因此，限制旧文件系统的磁盘查找延迟对新文件系统几乎没有影响。分配成本是新系统中导致写入慢于读取的因素。

当前，新文件系统的性能受到内存到内存复制操作的限制，这些操作需要将数据从系统地址空间中的磁盘缓冲区移动到用户地址空间中的数据缓冲区。这些复制操作约占执行 1/0 操作所花费时间的 40%。如果两个地址空间中的缓冲区正确对齐，则可以使用 VAX 虚拟内存管理硬件执行此传输而无需复制。当传输大量数据时，这是特别理想的。我们之所以没有实现，是因为它将以两种主要方式更改文件系统的用户界面：需要用户程序在页面边界上分配缓冲区，并且数据在写入后将从缓冲区中消失。

通过重写磁盘驱动器以将内核缓冲区链接在一起，可以实现更大的磁盘吞吐量。这将允许在单个磁盘事务中读取连续的磁盘块。UNIX 系统上使用的许多磁盘的每个磁道包含 32 或 48 512 字节的扇区。每个磁道恰好容纳两个或三个 8192 字节的文件系统块，或四个或六个 4096 字节的文件系统块。无法使用连续的磁盘块有效地限制了这些磁盘的性能，使其不足可用带宽的 50%。如果不能连续放置文件的下一个块，则到任何盘上的下一个可分配块的最小间距为六分之一半。这意味着没有连续块的最佳可能布局仅使用任何给定轨道的一半带宽。如果每个轨道包含奇数个扇区，则可以通过找到在另一个轨道上的所需旋转位置开始的块来将旋转延迟解析为任意数量的扇区。尚未实现块链接的原因是因为它将需要重写系统中的所有磁盘驱动器，并且当前的吞吐率已经受到可用处理器速度的限制。

当前，一次仅将一个块分配给一个文件。DEMOS 文件系统发现文件正在快速增长时使用的一种技术是一次预分配几个块，如果文件未使用，则在关闭文件时将其释放。通过分批分配，系统可以减少每次写入时分配的开销，并且可以减少使磁盘上的块指针与块分配保持同步所需的磁盘写入次数[13]。由于块分配当前仅占写系统调用所用时间的不到 10%，并且当前的吞吐率已经受到可用处理器速度的限制，因此不包括此技术。

5. 文件系统功能增强

UNIX 文件系统的性能增强不需要对应用程序可见的语义或数据结构进行任何更改。但是，一段时间以来，人们通常希望进行一些更改，但由于需要用户转储和还原所有文件系统，因此尚未引入。由于新文件系统已经要求所有现有文件系统都必须转储和还原，因此这些功能增强功能已在此时引入。

5.1. 长文件名

文件名现在可以具有几乎任意长度。仅读取目录的程序会受到此更改的影响。为了提高到不运行新文件系统的 UNIX 系统的可移植性，引入了一组目录访问例程，以为新旧系统上的目录提供一致的接口。

目录以称为块的 512 字节单位分配。选择此大小，以便可以在单个操作中将每个分配转移到磁盘。块被分解为称为目录条目的可变长度记录。目录条目包含将文件名映射到与其关联的 inode 所必需的信息。目录条目不允许跨越多个块。目录条目的前三个字段是固定长度的，并且包含：索引节点号，

条目的大小，以及条目中包含的文件名的长度。条目的其余部分为可变长度，并且包含一个以 0 终止的空文件名，填充到 4 字节边界。当前目录中文件名的最大长度为 255 个字符。

通过使一个或多个条目在其条目大小字段中累积可用空间来记录目录中的可用空间。这将导致目录条目大于保存条目名称以及固定长度字段所需的目录条目。分配给目录的空间应始终通过合计其条目的大小来完全解决。当从目录中删除条目时，通过将前一个条目的大小增加已删除条目的大小，其空间将返回到同一目录块中的前一个条目。如果目录块的第一个条目是空闲的，则该条目的索引节点号设置为零以指示未分配。

5.2 文件锁定

旧的文件系统没有锁定文件的规定。需要同步文件更新的进程必须使用单独的“锁定”文件。一个进程将尝试创建一个锁定文件。如果创建成功，则过程可以继续进行更新；否则，过程将继续进行。如果创建失败，则该过程将等待并重试。该机制具有三个缺点。通过遍历创建锁的尝试来处理消耗的 CPU 时间。由于系统崩溃而留下的锁必须手动删除（通常在系统启动命令脚本中）。最后，始终允许以系统管理员身份运行的进程创建文件，因此被迫使用其他机制。尽管可以解决所有这些问题，但是解决方案并不简单，因此已添加了一种锁定文件的机制。

最通用的方案允许多个进程同时更新文件。Peterson [12]中讨论了其中的几种技术。一种更简单的技术是使用锁序列化对文件的访问。为了获得合理的效率，某些应用程序需要能够锁定文件的片段。低音[2]已在 Onyx 文件系统中实现了字节级别的锁定。但是，对于标准系统应用程序而言，锁定文件粒度的机制就足够了。

锁定方案分为两类，一种是使用硬锁，另一种是使用咨询锁。咨询锁和硬锁之间的主要区别在于执行的程度。当程序尝试访问文件时，总是强制执行硬锁。咨询锁仅在程序请求时才应用。因此，咨询锁定仅在访问文件的所有程序都使用锁定方案时才有效。使用硬锁时，内核中必须实现一些替代策略。使用咨询锁时，策略留给用户程序。在 UNIX 系统中，允许具有系统管理员特权的程序优先于任何保护方案。因为许多需要使用锁的程序也必须以系统管理员的身份运行，所以我们选择实施咨询锁，而不是创建与 UNIX 原理不一致或系统管理程序无法使用的其他保护方案。

文件锁定功能允许合作程序对文件应用建议性共享或独占锁定。只有一个进程可能对文件拥有排他锁，而可能存在多个共享锁。共享锁和排他锁

不能同时存在于文件中。如果在另一个进程持有排他锁时请求任何锁，或者在另一个进程持有任何锁时请求排他锁，则锁定请求将阻塞，直到获得该锁为止。因为共享锁和排他锁仅是建议性的，所以即使一个进程已获得文件的锁，另一个进程也可能会访问该文件。

锁仅适用于打开的文件。这意味着无需关闭并重新打开文件即可操纵锁。例如，当进程希望应用共享锁，读取一些信息并确定是否需要更新，然后应用排他锁并更新文件时，这很有用。

如果无法立即获得锁定，则对锁定的请求将导致进程阻塞。在某些情况下，这是不令人满意的。例如，只想检查是否存在锁的过程将需要一种单独的机制来查找此信息。因此，如果无法立即获得锁定，则进程可以指定其锁定请求应返回一个错误。能够有条件地请求锁定对于希望为假脱机区域提供服务的“守护程序”进程很有用。如果守护程序的第一个实例锁定了进行假脱机的目录，则以后的守护程序进程可以轻松地检查是否存在活动的守护程序。由于仅在存在锁定进程时才存在锁定，因此在进程退出后或系统崩溃时，锁定文件永远不会保持活动状态。

几乎没有尝试进行死锁检测。系统完成的唯一死锁检测是，已对其施加锁的文件必须尚未具有相同类型的锁（即，两次连续的应用相同类型锁的调用中的第二个将失败）。

5.3. 符号链接

传统的 UNIX 文件系统允许同一文件系统中的多个目录条目引用单个文件。每个目录条目都将文件名“链接”到索引节点及其内容。链接概念是基础。索引节点不驻留在目录中，而是单独存在，并由链接引用。删除到一个 inode 的所有链接后，该 inode 会被释放。这种引用索引节点的样式不允许跨物理文件系统进行引用，也不支持机器间链接。为了避免这些限制，添加了类似于 Multics [3]所使用的方案的符号链接。

符号链接被实现为包含路径名的文件。当系统在解释路径名的组成部分时遇到符号链接时，该符号链接的内容会附加在该路径名的其余部分之前，并且会对该名称进行解释以生成最终的路径名。在 UNIX 中，相对于文件系统层次结构的根目录或相对于进程的当前工作目录指定路径名。相对于根指定的路径名称为绝对路径名。相对于当前工作目录指定的路径名称为相对路径名。如果符号链接包含绝对路径名，则使用绝对路径名，否则相对于文件层次结构中链接的位置来评估符号链接的内容。

通常，程序不想知道它们正在使用的路径名中有符号链接。但是，某些系统实用程序必须能够检测和操纵符号链接。提供三个新的系统调用功能

检测，读取和写入符号链接；七个系统实用程序需要更改才能使用这些调用。

在将来的 Berkeley 软件发行版中，可能可以使用路径名引用位于远程计算机上的文件系统。发生这种情况时，将可能创建跨机器的符号链接。

5.4. 改名

创建现有文件的新版本的程序通常会将新版本创建为临时文件，然后使用目标文件的名称重命名该临时文件。在旧的 UNIX 文件系统中，重命名需要对系统的三个调用。如果程序在两次调用之间被中断或系统崩溃，则目标文件只能保留其临时名称。为了消除这种可能性，已添加了重命名系统调用。重命名调用以保证目标名称存在的方式进行重命名操作。重命名适用于数据文件和目录。重命名目录时，系统必须进行特殊的验证检查，以确保目录树结构不会因创建循环或无法访问的目录而损坏。如果将父目录移入其后代之一，则会发生这种损坏。验证检查需要跟踪目标的后代目录以确保它不包括要移动的目录。

5.5. 配额

传统上，UNIX 系统试图最大程度地共享所有可用资源。因此，任何单个用户都可以分配文件系统中的所有可用空间。在某些环境中，这是不可接受的。因此，已添加了配额机制来限制用户可以获取的文件系统资源量。配额机制对用户可以分配的索引节点数和磁盘块数都设置了限制。可以为每个文件系统上的每个用户设置单独的配额。资源有硬性限制和软性限制。当程序超出软限制时，将在用户终端上打印警告。除非程序超出其硬限制，否则它不会终止。想法是，用户应在两次登录会话之间保持低于软限制，但在积极工作时他们可能会使用更多资源。为鼓励这种行为，如果用户超出其软限制，则在登录时会得到警告。如果用户无法通过过多的登录会话来解决问题，则最终会通过将其软限制强制为硬限制而受到谴责。

致谢

感谢 Robert Elz 对新文件系统的持续关注，并以合理有效的方式添加了磁盘配额。我们也感谢 Dennis Ritchie 对用户界面进行适当修改的建议。我们感谢 Michael Powell 对 DEMOS 文件系统如何工作的解释。他的许多想法都用于此实现中。彼得 ● 凯斯勒（Peter Kessler）和罗伯特 ● 亨利（Robert Henry）在文件调试不稳定的早期阶段，像真实用户一样表现出特别的称赞。

本来应该。评论的批评和建议极大地促进了论文的连贯性。

参考资料

1. ALMES, G。和 ROBERTSON, G。Hydra 的可扩展文件系统。在第三届软件工程国际会议论文集（乔治亚州亚特兰大，5 月 10 日至 12 日）上，IEEE，纽约，1978 年，第 288-294 页。
2. BASS, J。文件锁定的实现描述。Onyx Systems Inc.，73 E.Trimble Rd，San Jose，CA 95131（1981 年 1 月）
3. FEIERTAG, RJ 和 ORGANICK, EI Multics 输入输出系统。在第三届操作系统原理研讨会上的论文集（帕洛阿尔托，加利福尼亚州，10 月 18 日至 20 日）。ACM，1971，35-41。
4. FERRIN, TE 在版本 7 UNIX 中的性能和鲁棒性方面的改进。（1982 年，加利福尼亚州圣莫尼卡，冬季 Usenix 会议）计算机图形实验室技术。大学药学院第二讲加利福尼亚州旧金山市（1982 年 1 月）。
5. FERRIN, TE 重新探讨了 VMUNIX 的性能问题；登录名：Usenix 协会新闻信）第 7、5（1982 年 11 月），3-6。
6. T. FSCK KOWALSKI, UNIX 文件系统检查程序。科技 EECS 伯克利分校计算机系统研究小组第 9 报告，加利福尼亚州 94720（1983 年 7 月）。
7. KRIDLE, R.和 McKusICK, M.运行 4.2 BSD UNIX 的 VAX 系统的磁盘子系统选择对性能的影响。科技报告 8：计算机系统研究小组，EECS 部门，伯克利，加利福尼亚州 94720，1983 年 7 月。
8. KNUTH, D。计算机编程的艺术。卷 3，排序和搜索。Addison-Wesley，马萨诸塞州雷丁（1975）506-549。
9. K. MARUYAMA 和 S. SMITH。分布式空间磁盘文件的最佳重组。公社 ACM 19，11（1976 年 11 月），634-642。
10. NEVALAINEN, O.和 VESTERINEN, M.通过启发式方法确定顺序文件的阻塞因子。计算 J.20，3（1977 年 8 月），245-247。
11. 马萨诸塞州 PECHURA 和 SCHOEFFLER，JD 估计软盘的文件访问时间。公社 *ACM* 26，10（1983 年 10 月），754-763。
12. PETERSON, G.写作时同时阅读。ACM Trans。程序。郎 Syst。5，1（1983 年 1 月），46-55。
13. POWELL, ML DEMOS 文件系统。在第六届操作系统原理研讨会论文集（西拉斐特，印第安纳州，11 月 16 日至 18 日）上，ACM，1977，33-42。
14. RITCHIE，DM 和 THOMPSON，K。UNIX 分时系统。公社 ACM 17，7（1974 年 7 月），365-375。
15. SMITH，A。输入/输出优化和磁盘体系结构：一项调查。执行。EVA /。1（1981 年 1 月），第 104-117 页。
16. SMITH，A。参考书目，文件和 1/0 系统优化以及相关主题。歌剧 Syst。版本号 *15*，4（1981 年 10 月），39-54。
17. 符号。符号文件系统。Symbolics Inc.，9600 DeSoto Ave.，Chatsworth，CA 91311（1981 年 8 月）。
18. 汤普森，K。UNIX 实现。贝尔系统科技 J. 57，6，part2。（1978 年 7 月至 8 月），1931-1946 年。
19. 汤普森，M。Spice 文件系统。科技代表，CMU-CS-80，卡内基-梅隆大学，计算机科学系，宾夕法尼亚州匹兹堡（1980 年 9 月）。
20. TRIVEDI，KS 优化选择 CPU 速度，设备功能和文件分配。J. *ACM* 27，3（1980 年 7 月），457-473。
21. 白色，RM 磁盘存储技术。科学上午。243，2（1980 年 8 月），138-148。

1983 年 7 月收到；1984 年 2 月修订;1984 年 3 月接受