海量存储技术课程作业
关于《File Systems Fated for Senescence? Nonsense，Says Science!》论文的学习与整理

专 业 ： 计算机软件与理论
姓 名 ： 纪道敏
学 号 ： 2017202110025

摘　要

本文主要是关于 FAST'17 上的 Conway 等人的论文《File Systems Fated for Senescence？Nonsense，Says Science！》整理与学习。

文件系统往往会面临碎片化问题。由于文件系统在对文件分配相应的存储空间时，由于不能预知文件的后续操作情况，使得文件系统往往不能选择最优的分配方案。使得文件总是在磁盘上分散存储，从而造成寻道时间的增加，影响文件系统的性能。Conway 等人的工作就是围绕这一问题进行的。

Conway 等人引入了 natural transfer size(NTS)的概念，他们认为，当一次 I/O 操作中的数据量达到 NTS 时，此时数据的传输速率会达到总带宽。借由这一发现，他们提出了相关的优化策略，并将其引入到了 BetrFS 文件系统中。他们通过设立不同环境下的实验，比较了 BetrFS 文件系统以及其他主流文件系统在碎片化下和未碎片化下的性能。结果发现，其他文件系统性能受碎片化影响很大，而 BetrFS 文件系统性能基本不受碎片化性能影响。

本文的结构安排如下：第一部分会对有关文件系统碎片化的背景知识进行介绍；第二部分会介绍前面一些已有的碎片化方面的研究；第三部分会介绍 Conway 等人工作的一些相关概念；第四部分会着重对 Conway 等人的实验以及相关结果及结论进行重述；最后一部分会对 Conway 等人的工作进行总结。

目　录

1 背景介绍

在文件创建、移动、删除以及追加等操作中，文件系统会逐渐变得碎片化。

文件系统的碎片化发生在当逻辑上连续的文件块在物理磁盘上被分散存储时，读取这些文件需要额外的增加寻道时间。在机械硬盘上，即便是很少的寻道次数的增加也能对性能造成显著影响。例如，如果一个文件系统将一个 100M 的文件放在 200 个不相关的块上，假设磁盘的带宽是 100MB/s 并且寻道时间是 5ms，那么读取这些数据所消耗的时间将会是理想状态下的两倍。即便是在不进行机械寻道的 SSD 存储介质上，逻辑块上存储局域性的下降也会对性能造成影响。

为了减轻碎片化对性能的影响，一种最新的方法是在文件系统中加入了启发式的分配技术。例如，文件系统会试图将相关联的文件在磁盘上存放的很近，并在磁盘上为未来的文件预置了存储空间。一些文件系统（ext4、XFS、Btrfs 以及 F2FS）也加入了一些防止碎片化的工具和方法，它们会在物理层面上对文件重新进行组织，将这些文件块存放在连续的磁盘区域内。

在以往的相关研究中，出现了关于文件系统碎片化的不同观点。一方面，Smith 以及 Seltzer 在的相关工作表明文件系统在现实的任务中会出现碎片化现象，并且这种现象会导致文件系统的性能下降。而另一方面，也有人认为文件系统碎片化是在文件系统开发过程中已经被解决的问题。例如，在 Linux Syetem Administrator’s Guide 上，文件系统碎片化问题就被认为是不必要在意的问题。

存储技术的改变也能对文件系统碎片化问题有显著的影响。相关研究就表明，碎片化造成的影响会随着硬盘片的增大而加剧。这是因为在寻道时间固定的情况下，带宽的增加速率是磁盘容量增加速率的平方根。从而导致性能的增长并不能跟上存储容量的增长，导致该种情况下的碎片化带来的性能损失显得更为严重。

此外，一种观点认为文件系统碎片化对采用 SSD 技术的磁盘不是个重要的问题。例如，PCWorld 对 SSD 上的 NTFS 文件系统进行了性能测试，对于得到的测试结果，他们认为采用去碎片化的技术并不会对文件系统的性能有着显著提高，反而会在一定程度上影响到 SSD 磁盘的寿命。

在 Conway 等人的文章中，他们对从存储硬件、文件系统设计、数据结构理论等几个方面重新考虑碎片化问题。Conway 等人主要做了以下工作：

①设计了一个简单、快速、轻便的用于解决碎片化的方法和策略；

②发现文件系统碎片化是由于相应的分配策略引起的，并且即便是在SSD上，文件系统的碎片化现象依然会出现，而不像上述观点所描述的那样

③认为文件系统碎片化并不是不可避免的，并给出了几种用于避免碎片化的相关技巧，并将它们整合到实验所用到的系统—BetrFS中。在经过相关的测试后，他们发现，采用了这些技术的BetrFS系统在文件碎片化的解决上比其他文件系统有着更好的表现。甚至，新的BetrFS系统根本不会出现碎片化现象。

Conway等人通过现实的应用任务去将五种广泛采用的文件系统（Brtfs，ext4，F2FS，XFS 以及 ZFS）以及研究中所用到的 BetrFS 文件系统碎片化。在这里，Conway等人采用了两种碎片化方法，一种是进行连续的版本控制操作，用以模拟开发者在他的计算机上可能遇到的碎片化问题；另一种是是执行一个邮箱服务的benmark程序，用来模拟服务器可能遭遇的碎片化问题。

Conway 等人采用一下方式对磁盘化的影响进行评估：他们周期性的停止碎片化任务并对文件系统的读取性能进行测试。然而将相应的文件系统移植到一个新的分区，同样对其性能进行测试。通过比较两者之间性能的差异，可以得到碎片化对文件系统的影响大小。

他们发现：

①文件系统的碎片化在机械硬盘和SSD上均有发生。在第一种碎片化方法下，Conway等人发现在机械硬盘下性能损失达到50倍以上，而在SSD上性能损失达到2~5倍。而在第二种碎片化方法下，机械硬盘的性能损失达到4~30倍。

②碎片化的速率是惊人的。ext4文件系统仅在100次git pull操作后，性能下降了2倍以上；而Btrfs和ZFS在300次操作后也陷入了相同情形。

③BetrFS在测试中没有表现出碎片化倾向。除了Btrfs以外，BetrFS在碎片化操作后的性能测试比其他任何文件系统在未碎片化下的性能测试都要好。例如，在第二种碎片化的方法下，未碎片化的ext4文件系统比碎片化后的BeTrfS文件系同性能慢6倍以上。

④碎片化造成的性能损失有时候是惊人的。在第一种碎片化方法的测试中，所有文件系统在读取1GB的数据时，用时都超过了8分钟，甚至有两种需要用时2分钟以上。而BetrFS文件系统仅需要10秒钟。

通过一些小型的benchmark程序，Conway等人试图找到引起碎片化的主要原因。结果表明，

只要有10%的文件在创建的时候与目录结构无关，五种文件系统的吞吐量都达

不到 5MB/s。如果这些文件在拷贝的时候完全无序，只有 XFS 文件系统的吞吐速率能显著超过 1MB/s。而 BetrFS 能够维持在 50MB/s 的速度。

如果应用每次都只进行小规模的写数据，那么物理上这些块会被分散到磁盘上，从而在读回这些文件时对系统的性能造成影响。例如，在一个 Benchmark 中，4KB 的数据会被循环写到 10 个文件中。结果表明，Brtfs 和 F2FS 文件系统在吞吐速率上比这些数据一起被写入的时候慢上 10 倍。ext4 和 XFS 文件系统虽然表现的更加稳定，但是依然会有 2 倍的性能损失。BetrFs 文件系统最稳定，在整个测试中都能维持 2/3 的总带宽。

2 相关工作

之前的有关文件系统碎片的研究主要有以下三个方面：①文件系统的人工碎片化技术②碎片化程度的度量③预防和换件文件系统碎片化的方法

2.1 碎片化文件系统的生成

Smith 和 Selzer 通过研究在 5 个服务器上运行时间为 1 到 3 年的 50 多份文件系统的相关数据，提出了一种用于模拟和评价文件系统碎片化的方法。Smith 和 Selzer 的一个首要目标是用一种可表示的方法对文件系统的碎片化进行评价。

其他有关生成碎片化文件系统的方法也相继被提出。例如，为了评估 NFS 文件系统的性能，文献[]中提出了 TBBT 对 NFS 文件系统进行碎片化。

文件[]创建了一种名为 Impression 的框架，并提供了相关接口参数，使得用户能够根据不同需求对文件系统进行碎片化。

2.2 对碎片化文件系统的度量

Smith 和 Selzer 引入了 layout score 指标来对文件系统碎片化进行研究，layout score 是指在物理层面上被连续存储的文件所占比例。

DOF（degree of fragmentation）则是一种用来研究移动设备上文件系统碎片化的指标。DOF 是指实际存储的物理块数与理想状态下所需物理块数的比例。Layout score 以及 DoF 都是用来评价单一文件被碎片化的程度。

而另外一些研究而在统计方面对多个文件、不同文件大小和类型以及文件系统命名空间进行研究，旨在得到更复杂环境下的文件系统碎片化结果。

2.3 现有的碎片化解决策略

2.3.1 基于柱面或块的组（Cylinder Groups）

FFS 文件系统引入了柱面组的思想。每个组存放这它的 inode 节点以及相应块的位图信息。文件系统会将同一目录下的 inode 节点以及数据块存放在同一柱面组中。这一思想在以后被进一步拓展为块组（Block Groups）以及分配组（allocation groups）。

在 F2FS 文件系统中，磁盘会被首先分成许多相应的段。在磁盘操作的过程中，

一些写操作会被合并在一起，并且一些含有脏数据的段会在找到其他可写段之前被填满。

组思想是用来提高目录局域性的一种很好的方法，它会将临近存储空间预留给同一目录下的其他数据。但是，当磁盘的存储空间有限，同一目录下的文件依然会被分散到整个磁盘上。

2.3.2 Extents

除了 F2FS 以及 BetrFS 文件系统外，所有 Conway 等人测试的文件系统中都使用了 extents 策略。Extents 策略能够降低磁盘的管理开销，并且通过选择更大的 extents 进行存储能够提升大文件的存储局部性。例如，ZFS 文件系统就采用了首次适应算法来对可用的 extents 进行选择。

2.3.3 延迟分配

大多数现代文件系统中都采用了延迟分配策略。所谓延迟分配策略，是指文件系统不会对相应的写操作马上分配存储空间，而是等待相应的缓冲区被写到磁盘块上。通过延迟分配策略，当一个文件被追加时，文件系统可以为其预先分配一个更大的 extents。但是，延迟分配策略可能会造成相关操作的时延过长，从而对一些实时性要求很高的文件系统不利。因此，延迟分配策略只能保证在大的时间尺度上预防文件系统碎片化。

2.3.4 对小文件及元数据进行打包

对于包含许多小文件的目录，一个重要的优化方法就是对这些小文件及元数据进行整合，一起存放在相应的块或 extents 上。在 Btrfs 文件系统中，小的文件会被分为一片或多片，让后在 B-trees 里面进行打包。而这些小文件通过其对象标识符进行索引。所以，对于包含许多先文件的目录，它的存储局部性取决于这些对象标识符的接近度。

BetrFS 采取键值对方式对数据及元数据进行存储。这些键值对会被存放在 B^{ε}-tree 中，B^{ε}-tree 的节点空间非常大，通常为 2~4MB。键值对根据排序顺序在节点里进行打包，而这些节点会周期性的被写入到磁盘上。

3 Conway 等人的工作框架

3.1 Natural Transfer Size

Conway 等人发现当一些 I/O 操作的数据量比较大时，磁盘设备的带宽往往能够达到速率上限。这是因为一般情况下，执行顺序的 I/O 操作速度要比随机的 I/O 操作快得多。为了避免这样的情况，Conway 等人定义了 natural transfer size（NTS），它表示一次 I/O 操作中所需要传送的连续数据量。读操作在读取超过 NTS 的数据量时速度会接近带宽。

图 3.1 是 SSD 以及 HDD 有效带宽关于读取数据大小的函数图像，可以看出，SSD 以及 HDD 的 NTS 都是 4MB。

对于不同存储设备，造成顺序操作和随机操作上的速度差异的原因是不同的。对于 HDD 而言，额外的寻道时间是主要因素；而对于 SDD 而言，相关的原因需要由生产厂商说明。但无论如何，SSD 依然会出现上述现象，尽管这种速率上的差别在 SSD 上显得小的多

因此，为了避免磁盘碎片化，文件系统应该避免将大型文件分成比 NTS 小的文件。并且，它应该将一些小型文件组成超过 NTS 大小的数据，再一起进行存储。

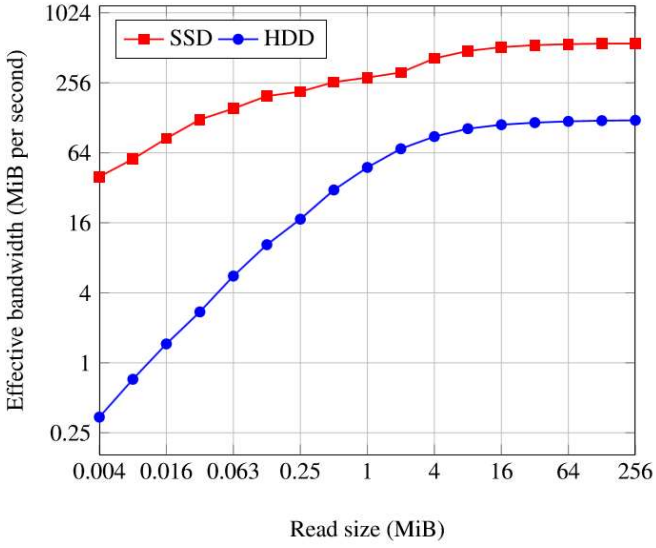


图 3.1 HDD 和 SDD 的有效带宽

3.2 分配策略

现在主流文件系统可以被分为三种：①B-tree-based② update-in-place③log-

structured。

①B-Tree 的碎片化情况由其叶子节点大小决定。如果叶子的大小比 NTS 小得多，B-Tree 也会变得碎片化，从而导致相应文件系统的碎片化。

然而，让叶子节点和 NTS 一样大会增加 write amplification。Write amplification 指的是需要改变的数据与要写回到磁盘数据的大小比例。在极端的情况下，叶子节点上一个字节的改变会导致整片叶子节点内的数据都要被写回到磁盘上，从而大大增加了磁盘的开销。因此，B-tree 的叶子节点往往较小。这又会导致文件系统的碎片化。

②在采用 update-in-place 策略的文件系统中，由于数据的更新马上就要在磁盘上进行体现，磁盘上数据的连续性非常难以维持。而在上文中 Conway 等人提到，延迟分配策略可以在一定程度上避免磁盘碎片化情况的出现。因此，在这种文件系统中，碎片化的现象也会出现。

③B^{ε}-tree 的每个节点都有一个级联日志，它记录着文件系统的变化。每一次节点出现溢出现象，它就会被刷到下一节点，由下一节点进行存储。这似乎会导致数据被重写多次，从而造成不必要的开销。然而，由于日志的存在，磁盘会对日志记录的内容一起执行，从而可以尽可能将相关数据存储在连续区域上。因此，B^{ε}-tree 中的叶子节点要比 B-tree 的大的多。在 Conway 等人采用的 BetrFS 文件系统中，B^{ε}-tree 的叶子节点被设置成了 4MB，这与 NTS 相一致。从而避免了文件系统碎片化。

④Log-structured merge trees（LSMs）在一定程度也可以避免碎片化。这是因为与 B^{ε}-tree 一样，其叶子节点大小也可以与 NTS 相匹配。

3.3 评价指标

为了在实验中对文件系统的碎片化程度进行衡量，Conway 等人引入了两个相关指标：

①Recursive grep test 指的从文件系统根目录下开始进行递归检索所需要的物理时间。它在一定程度上可以文件内的存储局域性（对大型文件）以及文件间的存储局域性（对包含很多小文件的目录）。

②Dynamic layout score 是对 Smith 和 Seltzer 提出了 layout score 的变体。前者主要是指在逻辑块空间上文件被连续存储所占的比例。Conway 等人将这一概念拓

展到动态的 I/O 模式上。它指在一个任务期间内，被连续存储文件所占的比例。它的值越大，说明对于一个给定任务，数据或元数据的存储局域性越好。

4 相关实验

4.1 关于碎片化的 microbenchmark 实验

Conway 等人将碎片化分为两种，文件间的碎片化以及文件内的碎片化，并对这两种分别进行了相关实验，旨在获取单一碎片化对文件系统性能的影响。

4.1.1 文件内碎片化

当一个文件体积逐渐增大的时候，磁盘上的当前块可能没有空间对其进行存储，从而导致该文件的其它部分被存储到磁盘的其他位置。这就是文件内的碎片化。

在这一部分的实验中，Conway 等人首先在创建 10 个具有一定初始大小的文件，然后将 0~100 个 4KB 的随机数据追加到各个文件末尾，直到每个文件的大小达到 400KB。在第一轮实验的时候，实验文件的初始大小为 400KB，所以整个文件是被连续写入的。然后再接下来的每轮实验中，逐渐减小文件的初始大小。在每次数据完全写完后，需要等待 90 秒，以确保文件系统处于不工作状态。

图 4.1 所示的是实验中这些文件系统在 SSD 和 HDD 上的性能表现。在 HDD 上，layout score 与文件系统性能相关（-0.93）；而在 SDD 上，所有的文件系统表现相近。这是因为这些文件系统的性能会被操作系统的预读操作所掩盖。因此，Conway 等人禁止了预读操作，重复进行了实验并将相关结果呈现在图 4.1（c）中。然后可以看出，性能与 layout score 之间的关系更加明显了（-0.67）。

图 4.2 展示的是各种文件系统的可视化结果。文件的每一个块会用一个竖条表示。连续的块会被合并在一起，并用一个带有颜色的长方形表示。可视化的结果表明：ext4 文件系统会尽可能将文件和接下来的文件顺序放在一起，而 Btrfs 和 F2FS 文件系统则将每轮的数据插入到顺序数据的末尾。ext4 和 XFS 文件系统都试图将文件存放在更大的 extents 中，尽管这些 extents 会随着实验的进行逐渐减小。

然而，这种可视化方法对于 BetrFS 文件系统并不适用，这是因为这些文件数据量很小，BetrFS 可以将他们完全存放在一个叶子节点中，并将他们一次连续读取。这在 HDD 上仅需要一次寻道。

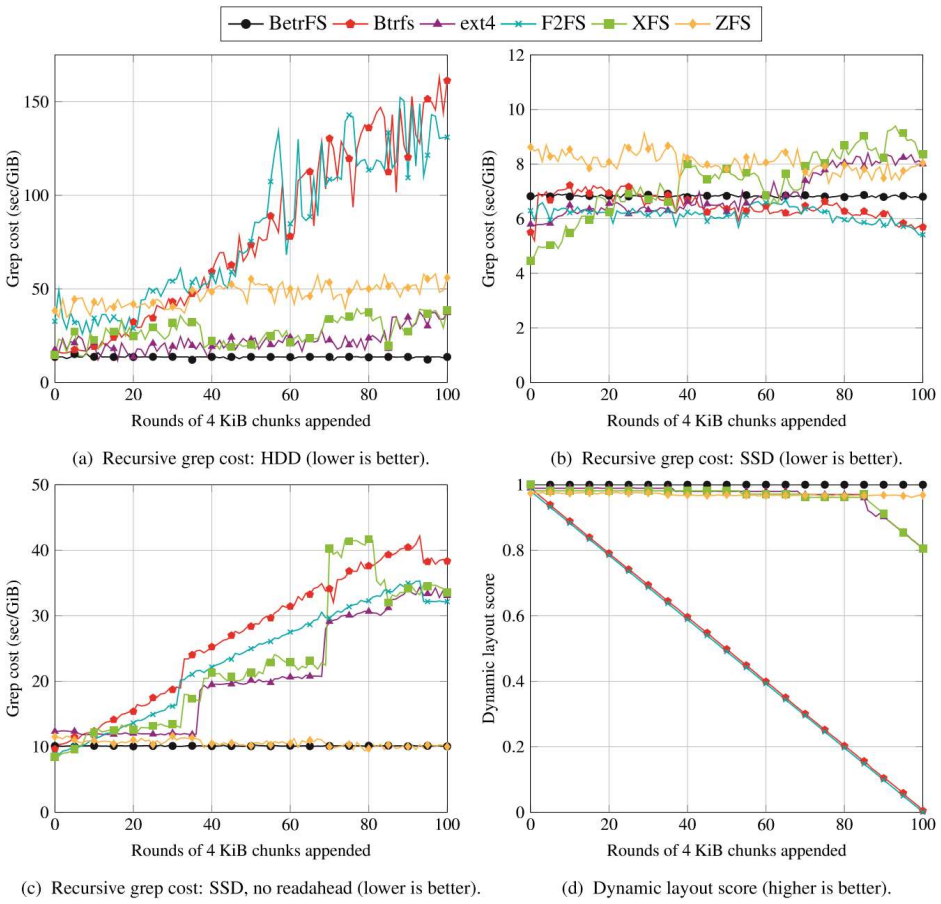


图 4.1 文件内碎片化实验结果

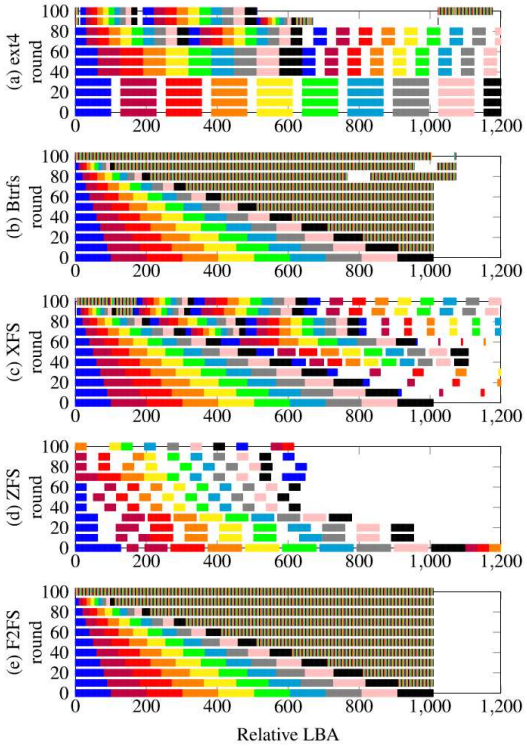


图 4.2 文件内碎片化实验可视化结果

4.1.2 文件间的碎片化

许多任务需要读取一些相关联的文件，一般来说，这些文件会被存放在同一目录下。文件间碎片化发生在这些文件在目录树上距离很近却在逻辑块空间上相隔较远。

在这节实验中，Conway 等人主要目的是研究命名空间创造顺序对文件间存储局域性的影响。为了更真实地模拟实际中地情况，他们从 github.com 上得到 Tensorflow 的目录结构，并将里面的每个文件的数据用 4KB 的随机数据进行替换。在实验中，首先会创建一个关于这些文件的顺序列表以及两个随机组合的乱序列表。在每一轮测试中，benchmark 会复制所有文件，并创建相关的目录。然后，在第 n 轮的时候，它会交换前 n%乱序列表上文件的顺序。因此，在第一轮的时候，文件复制是顺序执行的，然而在接下来的轮次里，文件复制的顺序将会逐渐变得随，直到在最后一轮顺序完全随机。

图 4.3 展示的是该部分的实验结果。可以看出，在 HDD 上，除了 BetrFS 以及 XFS 以外，其他文件系统都出现了比较陡峭、明显的性能下降，尽管此时只有一部分的文件处于乱序状态。甚至，F2FS 的性能因为太差以至于无法在图中表示出来，它在 100 轮的时候 4000 秒以上的时间才能读取 1GB 的数据。XFS 文件系统在一定程度上可以保持稳定的性能，但是即便是在一个完全有序的复制操作中，它的速度也仍然是带宽的 13~35 倍慢。BetrFS 则表现的最稳定，速度一般可以达到带宽的 1/3。

在 SSD 上，一半文件系统都表现的很稳定，而另一半文件系统则在完全有序的文件复制操作及 10%无序的文件复制操作间出现了较为明显的下降。这两种情形在 dynamic layout scores 上也显示了出来。尽管 ext4 和 ZFS 性能较为稳定，但是它们比其它文件系统的最好性能要差。而 BetrFS 是唯一表现的又稳定又快地文件系统，它在每轮的实验中性能都要比其他文件系统要好。

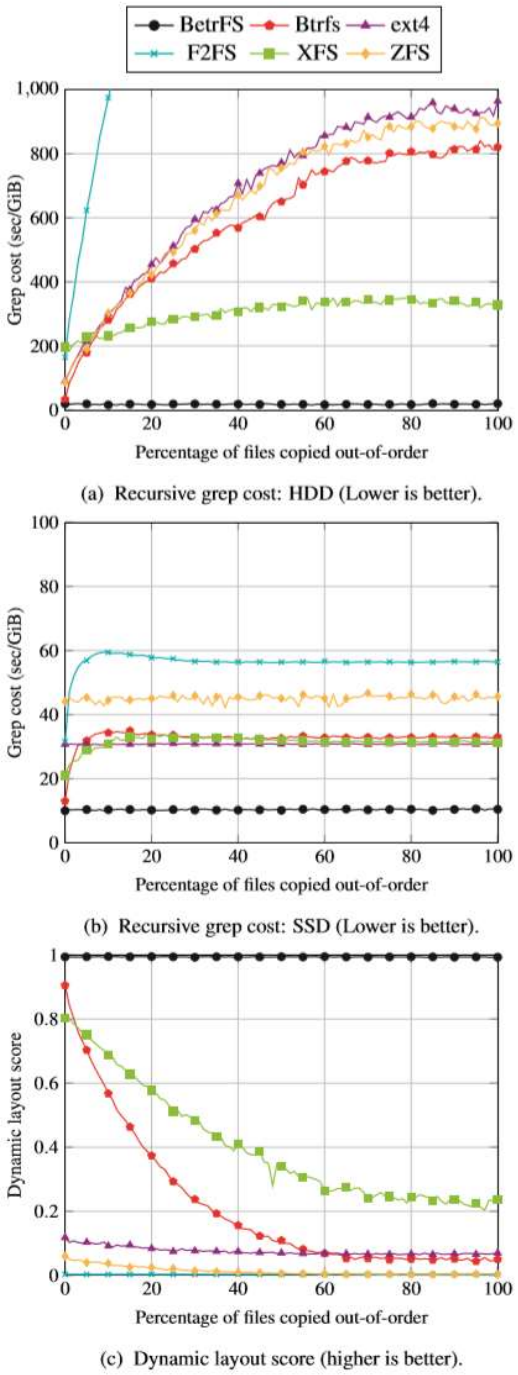


图 4.3 文件间碎片化实验结果

4.2 应用程序碎片化实验

4.2.1 git 程序的碎片化实验

该部分主要是为了模拟开发者计算机文件系统中的碎片化情况。

Git 是一个分布式的版本控制系统。使用者用 pull 操作从其他开发者的代码版本下载到本地，并于自己的代码版本相融合。一般情况下，在跨度很长的工程项目

开发中，git 使用者一天会进行数次 pull 操作。而 git pull 操作会导致新文件的创立以及旧文件的删除，并且 git 也又有一个自己的文件间数据结构，这些数据结构也会随着更新而改变。因此，git 在一定程度上也会造成文件系统的碎片化。

在这节实验中，benchmark 程序会从 Linux git repository 进行 10000 次 pull 操作。每进行 100 次 pull 操作后，benchmark 程序会进行 Recursive grep test 并计算每个文件系统的 dynamic layout score。这个得分将会与一个刚刚格式化后拥有相同文件内容的新分区上的得分进行比较。

如图 4.3（a）所示，除了 BetrFS 文件系统外，其他文件系统都出了比较明显的碎片化问题，它们的碎片化的性能比起未碎片化后的情况要慢上 3~15 倍。所有文件系统的性能都比 BetrFS 慢上 15 倍。在所有的实验中，F2FS 文件系统的碎片化最为明显，这与 dynamic layout scores 曲线相一致，这表明 F2FS 的数据存储局域性较差。

图 4.3（c）展示的是 SSD 上的实验测试结果，Brtfs 以及 XFS 文件系统出现了碎片化现象的特征。尽管这些现象不如在 HDD 那样明显，但是比起 Betrfs，传统文件系统的性能依然要慢上 2~4 倍。甚至，在经过碎片化处理后的 BetrFS，在 HDD 上的性能表现要比在 SSD 上经过碎片化处理后的其他文件系统性能还要好。

4.2.2 邮箱服务器程序的碎片化实验

除了上述的 git 程序碎片化实验之外，Conway 等人还进行了邮箱服务器程序的碎片化实验。实验中采用的程序是 Devecot mail server，它会将每条信息都存储在一个文件里，并且将每个收件箱都放在一个目录下。实验中模拟了 2 个用户的邮件操作，每个用户都有 80 个邮箱用来接收新邮件、删除旧邮件以及检索邮件。

在模拟的每一天内，邮箱服务器大概会进行 8000 多次操作，每一个操作可以等效地看作成删除或插入，这代表这邮箱的接受新邮件或者删除旧邮件操作。每封邮件都是一组随机的字符，长度是在[1,32000]之间服从均匀分布的随机数。每个邮箱初始都会有 1000 封邮件，并且由于插入和删除操作的概率是相等的，所以邮件数量大致在 1000 附近进行微小波动。实验总共会模拟 100 天，在每天过后，都会对每个文件系统的性能进行相应的测试。

图 4.4 显示的是相应的测试结果。尽管所有文件系统在未碎片化的情况下性能表现稳定，但是碎片化后的 ext4、Btrfs、XFS 以及 ZFS 文件系统的性能随着时间增长有着明显的下降趋势。具体而言，ext4 的性能较未碎片化版本下降 4.4 倍，这

比碎片化操作后的 BetrFS 性能要慢上 28 倍。XFS 这一方面的相关数据是 7 和 30。而 ZFS 性能下降的最为明显，在第 20 天的实验测试中，读取 1GB 数据所需要的时间就已经超过了 20 分钟。最后可以看出，BetrFS 文件系统在碎片化操作后性能不会下降。

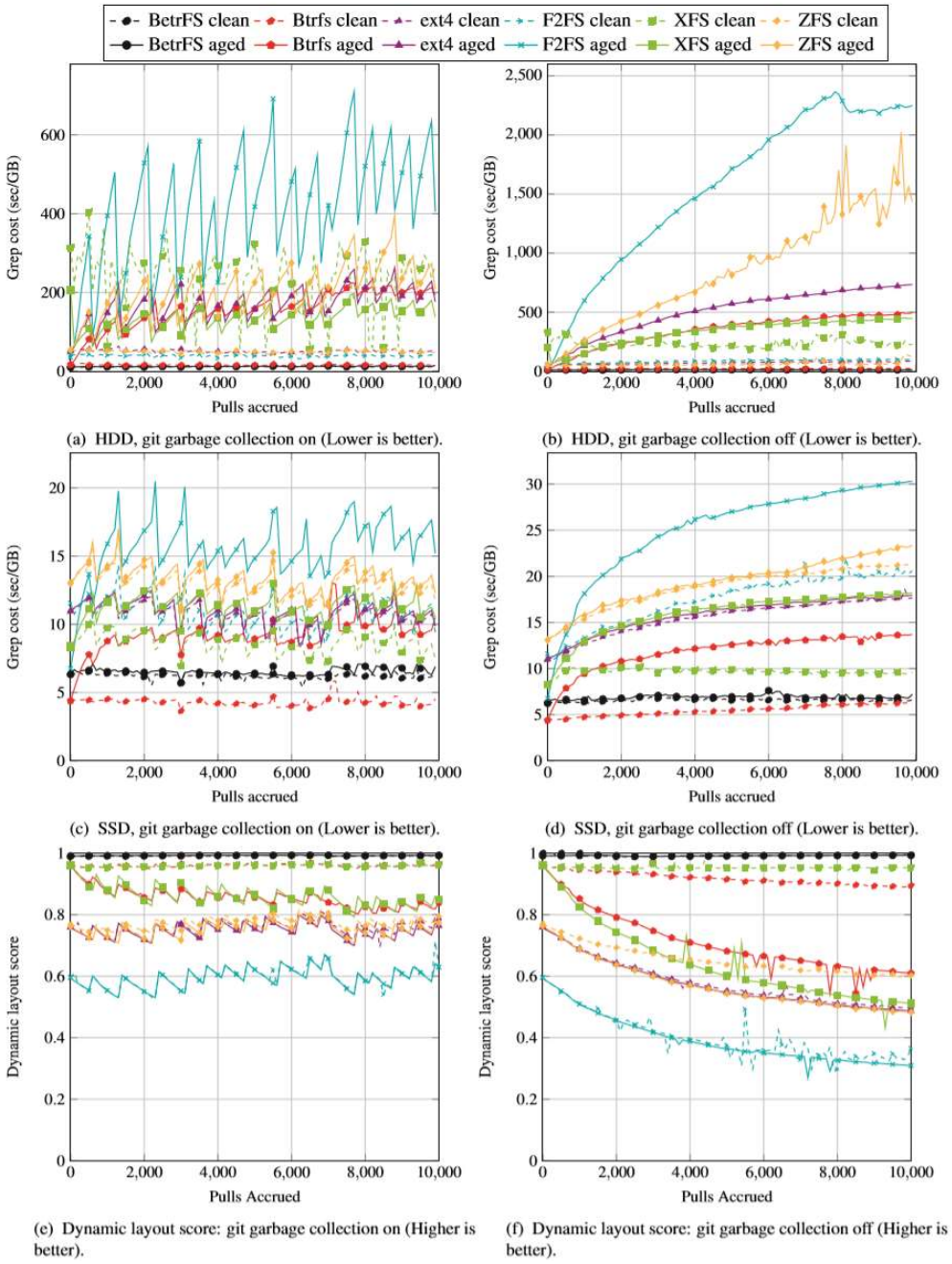
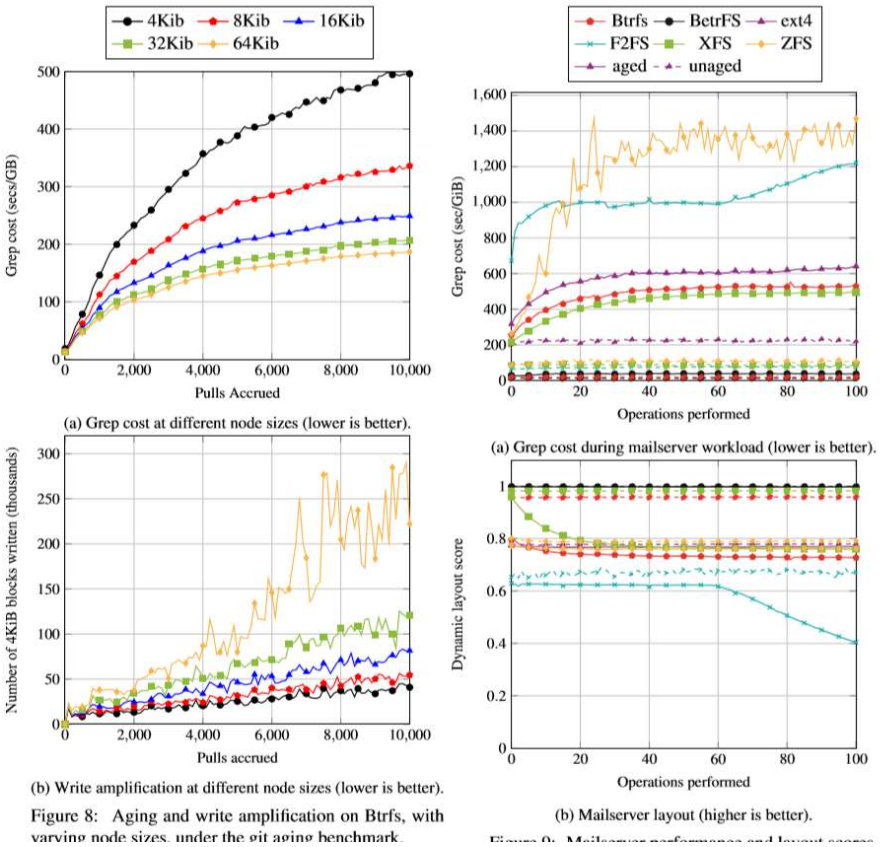


图 4.4 git 程序碎片化结果



(a) Grep cost at different node sizes (lower is better).

(b) Write amplification at different node sizes (lower is better).

Figure 8: Aging and write amplification on Btrfs, with varying node sizes, under the git aging benchmark.

(a) Grep cost during mailserver workload (lower is better).

(b) Mailserver layout (higher is better).

图 4.5 mail server 程序碎片化实验结果

5.总结

Conway 等人的实验结果表明，传统的文件系统在去碎片化方面表现的有较多不足：

①传统文件系统设计者认为要尽可能的避免将同一数据进行多次写操作，但是仅写一次数据会造成文件数据之间逻辑上的乱序。而重写操作可以分更好地实现数据存储局域性。而多次写操作所造成的额外开销可以像许多写优化策略那样，通过批量处理来抵消；

②当前文件系统所采用的启发式搜索策略在保持数据存储局域性上仍显不足，它们所分配的连续数据大小并没有达到 NTS，使得文件系统依然会面临碎片化问题。

尽管 Conway 等人取得了较为理想的实验结果，但是我认为，仍然有些地方是值得去完善的：

①BetrFS 文件系统采用的是 B^{ε}-tree，在实验过程中，其叶子节点大小被固定为 4MB。如果这里对不同大小的叶子节点进行实验测试，通过比较这些不同叶子大小的 BetrFS 性能，可以进一步验证 NTS 的猜想。

②实验中所采取的应用程序，一个是 git，另一个则是 mail server。这些程序更贴近于行业从事者，而距离普通人的使用环境较为遥远。之后的实验可以采取更为大众化的程序来对文件系统碎片化。