文件系统的宏观结构

前两篇笔记分别记录了 硬盘 和 闪存 两大类存储设备的物理特性。 这些存储设备，在操作系统中被抽象为块设备（block device）， 是因为他们不同于随机存储设备（Random Access Memory），需要按块为单位读写。 文件系统构建在块设备抽象之上，需要根据存储设备的物理特性做设计和优化，满足性能和安全性两方面的考量。

对传统硬盘而言，寻址时间依赖于地址跳转间隔，地址跳转越远寻址时消耗的时间越长。 因此设计文件系统时要尽量避免远距离的地址跳转，让读写尽量连续。对 SSD 而言，虽然没有了跳转时间的考量，但是随机寻址的写入仍然会加重 FTL 地址翻译的负担， 并且增加垃圾回收时写放大的程度，所以对 SSD 也是同样，按地址连续写入时的性能远优于随机写入。 现代硬盘上内外圈的读写性能也会影响到文件系统设计，外圈柱面的连续读写能比内圈柱面上快很多倍， 甚至一些硬盘的外圈连续读取速度能高于 SSD 的读取速度， 所以在硬盘上写入数据时将经常读写的数据（比如文件系统元数据）放入低端地址能有效提升性能。

现代文件系统通过合理安排存储设备上的数据结构布局，来为这些读写情况优化性能，这种优化叫地理布局优化（ geometry based optimization）。 本篇笔记想讨论一下文件系统在存储设备上安排数据时如何做地理布局优化。文件系统设计中这种优化分两方面： 安排数据布局的「宏观结构（macro structures）」和实际分配地址的「分配器（allocators）」算法。

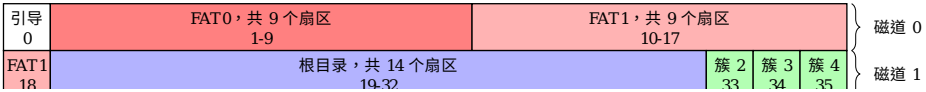
「宏观结构（macro structures）」是盘上数据结构（on-disk structures）中， 关于如何分割并分配整个地址空间的那些数据结构。「分配器（allocators）」则是在文件系统代码中负责 在宏观结构里分配地址安放数据的具体算法。数据结构和算法相互配合而最优化性能， 所以考察文件系统性能时也应当从这两者的设计开始考虑。

值得区别的是「宏观结构」是特定文件系统记录的方式，所以和文件系统代码的实现无关， 不同操作系统下实现同一个文件系统要支持同样的宏观结构布局，也难以在保持兼容性的前提下修改宏观结构。 而「分配器算法」则属于具体文件系统实现的代码，可能在使用同样盘上结构的情况下优化算法、改善分配器性能。 不同操作系统下实现的同一个文件系统也可以采用不同的算法。

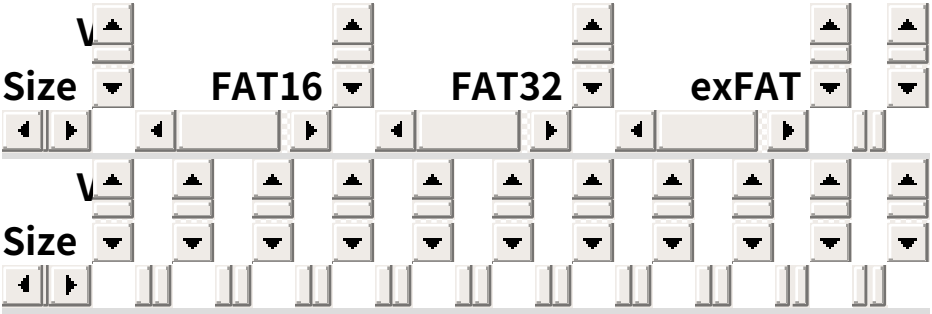
DOS文件系统与文件分配表（FAT12 FAT16 VFAT FAT32）

FAT 系文件系统最初为软盘驱动器设计，

比如 3.5 吋 1.44M 软盘使用 FAT12 ，每簇对应一个 512B 扇区， FAT表项是 12bit ，于是最多可以有 \(2^{12}=4096\) 个簇。 之前的文章讲过 软盘上共有 \(2 \times 80 \times 18 = 2880\) 个扇区，所以减去最开始的 33 个非数据区的扇区，软盘上的 FAT12 有 2847 个簇，于是存一个文件分配表需要至少 \(\frac{2847 \times 12\texttt{bit}}{8\texttt{bit}}=4270.5\) 字节的空间，不到 9 个扇区。按正确大小画出软盘上的 FAT12 大概这样：



注意上图 FAT12 中非数据区占用空间，引导扇区+两份 FAT 表+根目录全部位于软盘的首个柱面上， 这是微软有意如此设计的，从而在读写 FAT 的时候磁头位于最外圈的磁道，不需要移动磁头。 FAT12 用于软盘，可用空间不会很大，所以可以一个簇对应一个扇区也不会导致很大的 FAT 表。 相比之下用于硬盘的 FAT16 的大小可以很大，于是需要增加每簇中包含的扇区数量，让 FAT 表尽量放入首个柱面。



7 MiB–8 MiB	N/A	N/A	N/A	N/A	N/A	N/A	4KiB	2K	8KiB	4KiB
8 MiB–16 MiB	512B	32K	64KiB	N/A	N/A	N/A	4KiB	4K	16KiB	8KiB
16 MiB–32 MiB	512B	64K	128KiB	512B	64K	256KiB	4KiB	8K	32KiB	8KiB
32	1KiB	64K	128KiB	512B	128K	512KiB	4KiB	16K	64KiB	8KiB

MiB–64 MiB											
64 MiB–128 MiB	2KiB	64K	128K	1KiB	128K	512K	4KiB	32K	128K	4KiB	
128 MiB–256 MiB	4KiB	64K	128K	2KiB	128K	512K	4KiB	64K	256K	4KiB	
256 MiB–512 MiB	8KiB	64K	128K	4KiB	128K	512K	32KiB	256K	64KiB	8KiB	
512 MiB–1 GiB	16KiB	64K	128K	4KiB	256K	1MiB	32KiB	32K	128K	4KiB	
1 GiB–2 GiB	32KiB	64K	128K	4KiB	512K	2MiB	32KiB	64K	256K	4KiB	
2 GiB–4 GiB	64KiB	64K	128K	4KiB	1M	4MiB	32KiB	128K	512K	4KiB	
4 GiB–8 GiB	N/A	N/A	N/A	4KiB	2M	8MiB	32KiB	256K	1MiB	4KiB	
8 GiB–16 GiB	N/A	N/A	N/A	8KiB	2M	8MiB	32KiB	512K	2MiB	4KiB	
16	N/A	N/A	N/A	16KiB	2M	8MiB	32KiB	1M	4MiB	4KiB	

GiB–32 GiB										
32 GiB–16TiB	N/A	N/A	N/A	N/A	N/A	N/A	128K	1G	512M	4KiB
16 TiB–32 TiB	N/A	N/A	N/A	N/A	N/A	N/A	128K	2G	1GiB	8KiB
32 TiB–64 TiB	N/A	N/A	N/A	N/A	N/A	N/A	128K	4G	2GiB	16KiB
64 TiB–128 TiB	N/A	N/A	N/A	N/A	N/A	N/A	128K	8G	4GiB	32KiB
128 TiB–256 TiB	N/A	N/A	N/A	N/A	N/A	N/A	128K	16G	8GiB	64KiB
> 256 TiB	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

从中我们可以算出默认 FAT 表项的最大数量限制：

Volume Size	FAT1	FAT3	exFA
7 MiB–8 MiB	N/A	N/A	2K
8 MiB–16 MiB	32K	N/A	4K
16 MiB–32 MiB	64K	64K	8K

32 MiB–64 MiB	64K	128K	16K
64 MiB–128 MiB	64K	128K	32K
128 MiB–256 MiB	64K	128K	64K
256 MiB–512 MiB	64K	128K	16K
512 MiB–1 GiB	64K	256K	32K
1 GiB–2 GiB	64K	512K	64K
2 GiB–4 GiB	64K	1M	128K
4 GiB–8 GiB	N/A	2M	256K
8 GiB–16 GiB	N/A	2M	512K
16 GiB–32 GiB	N/A	2M	1M
32 GiB–16TiB	N/A	N/A	128M
16 TiB–32 TiB	N/A	N/A	256M
32 TiB–64 TiB	N/A	N/A	512M
64 TiB–128 TiB	N/A	N/A	1G
128 TiB–256 TiB	N/A	N/A	2G
> 256 TiB	N/A	N/A	N/A

和单个 FAT 表大小：

Volume Size	**FAT**	**FAT**	**exFA**
7 MiB–8 MiB	N/A	N/A	8KiB
8 MiB–16 MiB	64KiB	N/A	16KiB
16 MiB–32 MiB	128KiB	256KiB	32KiB
32 MiB–64 MiB	128KiB	512KiB	64KiB
64 MiB–128 MiB	128KiB	512KiB	128KiB
128 MiB–256 MiB	128KiB	512KiB	256KiB
256 MiB–512 MiB	128KiB	512KiB	64KiB
512 MiB–1 GiB	128KiB	1MiB	128KiB

1 GiB–2 GiB	128KiB	2MiB	256KiB
2 GiB–4 GiB	128KiB	4MiB	512KiB
4 GiB–8 GiB	N/A	8MiB	1MiB
8 GiB–16 GiB	N/A	8MiB	2MiB
16 GiB–32 GiB	N/A	8MiB	4MiB
32 GiB–16TiB	N/A	N/A	512MiB
16 TiB–32 TiB	N/A	N/A	1GiB
32 TiB–64 TiB	N/A	N/A	2GiB
64 TiB–128 TiB	N/A	N/A	4GiB
128 TiB–256 TiB	N/A	N/A	8GiB
> 256 TiB	N/A	N/A	N/A

增加簇大小的原因是为了限制 FAT 表整体的大小，因为在使用 FAT 表的文件系统中，需要将 FAT 表整体装入内存才能满足文件访问和簇分配时的性能，如果读写 FAT 表的范围需要访问磁盘，那么整个文件系统的读写性能将暴跌到几近不可用。到针对闪存优化的 exFAT 上，虽然在 FAT 表外还有额外的簇分配位图（Cluster Bitmap），但是也同样要限制 FAT 表整体大小，减少对 FAT 表区域的随机读写。

早期 Unix 文件系统

柱面-磁头-扇区寻址与地理位置优化

BSD 的 FFS

Linux 的 ext2

物理寻址到逻辑块寻址

Flash 媒介与 FTL