文件系統的宏觀結構

前兩篇筆記分別記錄了 硬盤 和 閃存 兩大類存儲設備的物理特性。 這些存儲設備，在操作系統中被抽象爲塊設備（block device）， 是因爲他們不同於隨機存儲設備（Random Access Memory），需要按塊爲單位讀寫。 文件系統構建在塊設備抽象之上，需要根據存儲設備的物理特性做設計和優化，滿足性能和安全性兩方面的考量。

對傳統硬盤而言，尋址時間依賴於地址跳轉間隔，地址跳轉越遠尋址時消耗的時間越長。 因此設計文件系統時要儘量避免遠距離的地址跳轉，讓讀寫儘量連續。對 SSD 而言，雖然沒有了跳轉時間的考量，但是隨機尋址的寫入仍然會加重 FTL 地址翻譯的負擔， 並且增加垃圾回收時寫放大的程度，所以對 SSD 也是同樣，按地址連續寫入時的性能遠優於隨機寫入。 現代硬盤上內外圈的讀寫性能也會影響到文件系統設計，外圈柱面的連續讀寫能比內圈柱面上快很多倍， 甚至一些硬盤的外圈連續讀取速度能高於 SSD 的讀取速度， 所以在硬盤上寫入數據時將經常讀寫的數據（比如文件系統元數據）放入低端地址能有效提升性能。

現代文件系統通過合理安排存儲設備上的數據結構佈局，來爲這些讀寫情況優化性能，這種優化叫地理佈局優化（ geometry based optimization）。 本篇筆記想討論一下文件系統在存儲設備上安排數據時如何做地理佈局優化。文件系統設計中這種優化分兩方面： 安排數據佈局的「宏觀結構（macro structures）」和實際分配地址的「分配器（allocators）」算法。

「宏觀結構（macro structures）」是盤上數據結構（on-disk structures）中， 關於如何分割並分配整個地址空間的那些數據結構。「分配器（allocators）」則是在文件系統代碼中負責 在宏觀結構裏分配地址安放數據的具體算法。數據結構和算法相互配合而最優化性能， 所以考察文件系統性能時也應當從這兩者的設計開始考慮。

值得區別的是「宏觀結構」是特定文件系統記錄的方式，所以和文件系統代碼的實現無關， 不同操作系統下實現同一個文件系統要支持同樣的宏觀結構佈局，也難以在保持兼容性的前提下修改宏觀結構。 而「分配器算法」則屬於具體文件系統實現的代碼，可能在使用同樣盤上結構的情況下優化算法、改善分配器性能。 不同操作系統下實現的同一個文件系統也可以採用不同的算法。

DOS文件系統與文件分配表（FAT12 FAT16 VFAT FAT32）

FAT 系文件系統最初爲軟盤驅動器設計，

早期 Unix 文件系統

柱面-磁頭-扇區尋址與地理位置優化

BSD 的 FFS

Linux 的 ext2

物理尋址到邏輯塊尋址

Flash 媒介與 FTL