嵌入式系統之儲存設計

Storage Design for Embedded Systems

蔡亮宙 副教授
南台科技大學
ljtsai@mail.stut.edu.tw

曾文偉
南台科技大學
e041984@gmail.com

陳冠宇
南台科技大學
tasean833@gmail.com

摘要

嵌入式系統大多有其特殊的應用領域，對於可能需要儲存檔案時，常以 Flash 記憶體為其優先考量，但是若考慮其有限的重複寫入次數與大容量的儲存需求時，硬碟式設備還是有其特定的需求領域。本文以醫療系統及其檔案儲存為切入點，利用在嵌入式系統應用中儲存檔案的容量變化不大的特性，實現一個使用硬碟 FAT 格式的驅動程式，名為 FAT+。

基本上 FAT+並未更動任何 FAT 格式，它主要是在驅動程式中依據應用領域的不同，固定讀寫單位為兩種不同大小的區塊，在我們的原型系統中顯示，FAT+可以有效的消除或減低檔案儲存的破碎問題，不只可以加速讀寫效能，而且可將其視為一般的 FAT 設備而在其他不同的設備上讀取。

關鍵詞：嵌入式系統、檔案系統、FAT

Abstract

In most cases, embedded systems are dedicated for specific applications. When data storages are required, flash memory devices are likely to be first considered. However, when the rewrite counts increase and the storage scales rise, hard disk can be another good choice to play the role of storing data. This article is motivated by medical devices with embedded data storage. Based on the phenomenon that most file sizes of embedded applications are limited without drastic variation, we implement a disk driver, named FAT+, for hard disks using FAT format.

Basically, FAT+ does not involve any format modification on FAT. There are 2 sizes of write units for primitives issued from upper layers, i.e. the unit of large block is used for big files and the unit of small block is used for small files respectively. The prototype shows that FAT+ can effectively eliminate or reduce storage fragmentation and therefore increase the read write performance. Furthermore, it still keeps compatibility with FAT format and can be read by generic FAT facilities.

Keywords: Embedded systems、File systems、FAT

簡介

在嵌入式系統中的儲存裝置，也是一種相當特殊的領域，因此儲存裝置本身是否被作業系統支援也是需要被考慮的。除此之外，依照不同的需求，也可能需要選擇不同的檔案系統格式，如JFS[5]、JFFS2[6]、Ext3[7]、cramfs[8]、romfs[9]等適用於嵌入式系統的檔案系統，都是可能需要被使用的。

例如JFFS2功能就是在管理MTD[10]裝置上所實作的日誌型檔案系統。相較於其他Flash儲存方案，JFFS2並沒有提供讓傳統檔案系統也可以使用 Flash 的Translation layer，它只會直接在MTD裝置上實做日誌結構的檔案系統。cramfs在嵌入式的環境之下，記憶體和 Flash 資源都需要節約使用。如果使用 ramdisk 方式來使用檔案系統，那麼在系統運行之後，首先要把 Flash 上的映像檔解壓縮到記憶體中，構造起 ramdisk 環境，才可以開始運行程式。而uClinux系統採用romfs檔案系統，這種檔案系統相對於一般的ext2[11]檔案系統要求更少的空間。空間的節約來自於兩個方面，首先核心支援romfs檔案系統比支援ext2檔案系統需要更少的代碼，其次romfs檔案系統相對簡單，在建立檔案系統Superblock需要更少的儲存空間。

目前常看到的儲存裝置大多是硬碟與快閃記憶體裝置，如果以效能及移動性則是以快閃記憶卡為主，但是在成本與容量上還是以硬碟優於快閃記憶體裝置。但是快閃記憶體裝置在嵌入式系統中，所能進行重複讀寫操作約為10萬次且儲存空間也不足。而本文以醫療系統以及檔案儲存為主，因此選用硬碟作為實現的儲存裝置，主要就是因為存放空間大於快閃記憶卡。

由於目前數位相機所拍攝的照片超過1MB與多媒體影片超過10MB以上甚至到幾GB都有可能，尤其在某些特定的應用或許更高，如醫療資訊、數位攝影機的檔案，當讀取大檔案時如果檔案儲存位址分散的很凌亂，此時硬碟磁頭讀取的時間也相對的增加，如果使用傳統FAT 檔案配置方式解決方法就是磁碟重組。本文以醫療資訊系統為例，目前已大量使用 PACS/HIS 系統來管理院內資訊與病患的相關影像檔案，表 1 是醫療院所每日所得到的醫療資訊影像的統計，每種儀器所產生的醫療資訊影像檔案大小都是一定，在儲存上可藉由此特性配合FAT+作儲存的最佳化。而依照每日檢查人次的數量變化量來調整參數以便於儲存。

例如：由表得知CT斷層掃描所得到的影像單張為 0.5MB，CR (Computed Radiography)每次量測為8MB，當儲存裝置用來儲存此兩種醫學影像格式時可以將大區塊單元規劃為 8MB，小區塊單元規劃為 0.5MB，並依照每日檢查人次的數量來調整大區塊單元與小區塊單元分配的比例，如圖 1。

512KB (32 Cluster 小區塊單元)*20

8MB (512Cluster 大區塊單元)*1

※Cluster:16KB

圖 1 醫療資訊磁碟配置

表 1 醫療資訊影像統計表[4]

儀器類別	檔案大小	每日檢查人次	片量大小
CT	512x512x12	40	20Mbytes
MRI	512x512x12	60	20Mbytes
NM	128x128x12	30-60	1-2Mbytes
DSA	512x512x12	30-300	15-180Mbytes
CR	2048x2048x12	2	16Mbytes
Digitized X-ray Film	2048x2048x12	2	16Mbytes
Digitized	4096x4096x12	4	1286Mbytes
US	512x512x24	20-120	50-90Mbytes
Endoscope	512x512x24	20-60	15-45Mbytes
Cardiac X-ray	512x512x12	3600	2700Mbytes

一、相關研究

DICOM 與 PACS

DICOM (Digital Imaging and Communications in Medicine) 是一種通用的標準協定，應用在醫學影像的處理、儲存、列印、傳輸上。其影像檔案可分做兩個部份來探討，一個是檔案格式、另一個為檔案內容。圖 1.1 為一典型的 DICOM 影像檔案格式[12]。檔頭(header)，記錄影像屬性。如病患姓名、病歷號碼...等，是由許多的基本資料單位組織而成，即資料元素(data element)所構成。

PACS (Picture Archiving and Communication System) 是指將各種醫療攝影裝備(Modality)中攝得的影像，通過 CR (Computed Radiography)數位化後，儲存在硬體等相關儲存媒介，並通過網路傳輸到各個終端機，使得診間(OPD)、病房(IPD)、開刀房(OR)、急診(ER)等，只要是有工作站的地方，皆可即時查詢就醫者影像的系統。

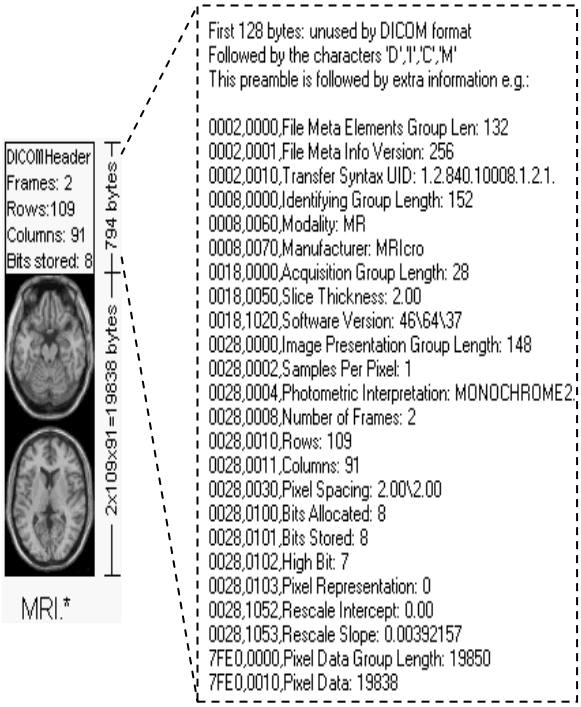

圖 1.1 典型的 DICOM 影像檔案格式

網路儲存裝置

早期的資料儲存方式，大都是採用硬碟為主要的儲存媒體，對於網路上的檔案共用及資料的存取，皆需透過檔案伺服器這個角色，此種資料的儲存架構，我們稱之為直接附加儲存裝置DAS(Direct Attached Storage)架構。在過去，大多採用 DAS 以從事儲存的任務，但是在資料量大幅增加與網路技術進步的趨勢下，DAS 逐漸地已經無法滿足對儲存工作的需求，因而有 SAN(Storage Area Network)及 NAS(Network Attached Storage) 的出現。

NAS 是一種透過網路進行檔案存取與共享的儲存設備。它可以提供無限制容量擴充的網路資料分享機制，只要連上網路任何節點，各種不同作業平台的電腦系統便可藉由 NAS 設備分享資訊。因此內部的其他伺服器便不須同時兼負資料存取的動作，而有更大空間去進行其他的工作。NAS 特色是建置容易、低成本，因此經常被應用在 UNIX 中的 NFS[13]或 Windows NT 的 CIFS[14]中的文件共享工作。原則上，NAS 適合存放不需要運算的資料，例如電子郵件與檔案。

SAN 是將許多儲存裝置從區域網路獨立出來成為另一個網路，最大的特色就在於實現伺服器與儲存設備之間多對多的高速連接。經由光纖通道 (Fibre Channel)或是 iSCSI[15][16]技術，通過光纖通道交換機、或 Gigabit 交換機連接至伺服器和磁碟陣列系統，所構成的專屬儲存的區域網路。SAN 架構如圖 1.2 所示基本定義為獨立、跨平台的資料存取區域，也就是資料的存取可以與現有存取連接媒體 (網路或 Host attached Storage，如SCSI)無關，可以達到最佳的傳輸速度。

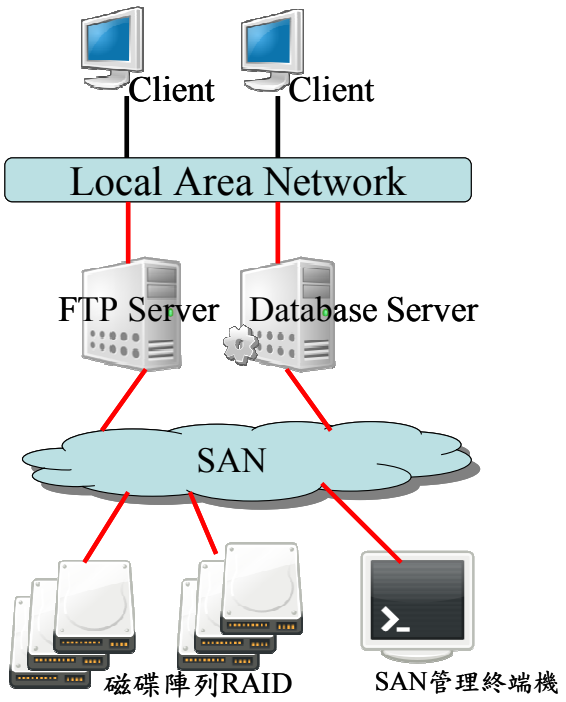

圖 1.2 SAN 架構

儲存設備

對於低價位的 IDE 硬碟而言，它有兩種介面，一為 PATA 另一個為 SATA。PATA 是傳統的並列傳輸介面而 SATA 是串列的傳輸介面，但是這只是傳輸匯流排的介面不同而已，在內部的命令格式也是向下相容，不論介面如何調整內部的驅動程式是不需修改。

IDE 整合驅動電子介面採用 ATA 的通訊協定。早期 IDE 控制介面是以 CHS (Cylinder-Head-Sector)的方式定址[17]，以 16-bit 代表磁柱(cylinders)，4-bit 代表磁頭(heads)，8-bit 代表磁區(sectors)，所以這種 ATA/IDE 介面最多可驅動 65536 Cylinders，16 Heads 與 255 Sectors。另

外，加上 BIOS 的限制，實際上所能驅動的 CHS 數目及最大容量如表 1.1 所示。

表 1.1 CHS 定址模式

	ATA/IDE	BIOS	妥協後
Sectors	255	63	63
Heads	16	255	16
Cylinders	65536	1024	1024
最大容量	127.5GB	7.84GB	504MB

而 LBA (Logical Block Addressing)定址省略了所有 CHS 轉換的動作，直接採取邏輯區塊定址方式。目前硬碟的定址方式都改用 LBA 定址模式，LBA 的定址方式是將原本的 CHS 個別分開的位元全部湊合在一起並且擴充原本保留未使用的暫存器位元使其達到 48 個位元使其定址空間達到 144PTB，LBA 模式使用的是軟體模擬轉換 CHS 模式，因而舊有的驅動程式也是可繼續使用以達到向下相容。

二、系統設計與原理

磁碟配置(Disk Layout)

在 FAT 磁碟配置如圖 2.1 所示，依順序分別是開機磁區(Boot Sector)使用 512Bytes 紀錄也稱作 Super Block，記錄著幾個 FAT 重要的資訊分別為：

(1) 一個磁區(Sector)使用多少個位元組。

(2) 一個叢集(Cluster)使用多少個磁區。

(3) 保留的磁區包含開機磁區。

(4) FAT 配置表數量。

(5) FAT table 使用多少個磁區。

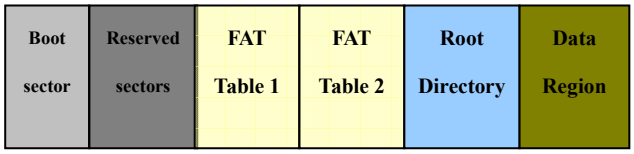

圖 2.1 FAT Disk Layout

由以上訊息可以算出有關 FAT 磁碟配置相關位置：

第一個 FAT Sector = LBA Start (由 MBR 提供) + Number Reserved

第一個 Data Sector = First Fat + (Number of FAT * Sector per FAT)

DataSector = First Data Sector + (cluster number-2) * Sector per cluster

圖 2.2 FAT32 檔案配置方式

在 FAT 檔案配置裡叢集(cluster)為最小單位，上圖則是以 FAT32 配置方式為例子。此圖代表的是 FAT 表裡面以十六進制表示，每一個 32 位元為一個 Cluster，前面兩個 Cluster 被保留，第二個 Cluster 開始使用記錄資料。FAT 是一個串列資料結構(link list)的檔案系統，每個資料記錄著下一個 Cluster 的編號。如果這是最後一個 Cluster 那內容填寫 0xffffffff，如果這個 Cluster 還沒被使用到就填 0。

舉例來說，由圖可得知根目錄的資訊由第 2 個 Cluster 開始放置依序分別為：2－＞9－＞10－＞11－＞17，總共佔用了 5 個 cluster。可由上面所提供的 Data Sector 換算出實際在硬碟的 LBA 位址，其它的檔案也是如此的計算。由此可以明顯的看出，由於 FAT 檔案系統採用的是串列的方式記錄 Cluster，所以在磁碟使用一段時間後會有很嚴重的破碎問題。因此在微軟的作業系統中都有磁碟重整的工具，提供給使用者在使用系統一段時間後，必須將磁碟重整。

有鑑於此，本文提出一個以 FAT32 為基礎的檔案配置方式，改良 FAT32 的檔案配置問題，應用在嵌入式系統內達到效能與相容兼具的檔案配置系統，稱為 FAT+。

FAT+

傳統的 FAT 檔案配置方式單位都是以 Cluster 為單位，在本文提出一個可兼容大檔案與小檔案，且相容於原有 FAT 檔案配置方式的磁碟配置方法。使用於嵌入式系統上並且可以避免有效減少 FAT 檔案系統的磁碟重組，並且考量用於醫學資訊的儲存上，由於產生的檔案大小並不會差異太大，由預先知道檔案的大小的特點在儲存時將檔案內容存入連續空間內，以利於日後檔案讀取時的效能。

本文將原本 FAT 配置的最小單元由 Cluster 改成由多個或一個 Cluster 所組成的小區塊單元與大區塊單元兩種儲存單位。例如有一顆 20G 的磁碟時常用來存放數位影像，由於數位影像目前大多是 1MB~2MB 大小，可由此將大區塊單元設定為 1MB、小區塊單元為 16KB。當一個數位影像被儲存時，因為大於小區塊的單元的容量將會被存放在大區塊單元中，在大區塊單元使用的是連續配置空間能夠避免外部破碎的產生，也可以使讀取的效率提高。

FAT+ 驅動程式工作方塊如圖 2.3 所示，在此將檔案分為超過 1MB 的大區塊單元以及小於 1MB 的小區塊單元兩種儲存單元，分別規劃在固定的磁區位址。例如起頭由 0 開始，一開始是存放大區塊單元 1MB 佔用掉實際上佔用掉傳統 FAT 檔案配置 64 個 cluster，下一個小區塊單元位於起頭偏移量 1MB 的位址，小區塊單元所佔用的傳統 FAT 檔案配置是 1 個 Cluster 的數量，檔案配置方式可以動態的改變。如圖 2.3 大區塊單元後接著五個小區塊單元，使用者可以將其調整成十個大區塊單元後接著一個小區塊單元，如此的調整可以應用在各種特殊儲存容量空間的最佳化配置上。

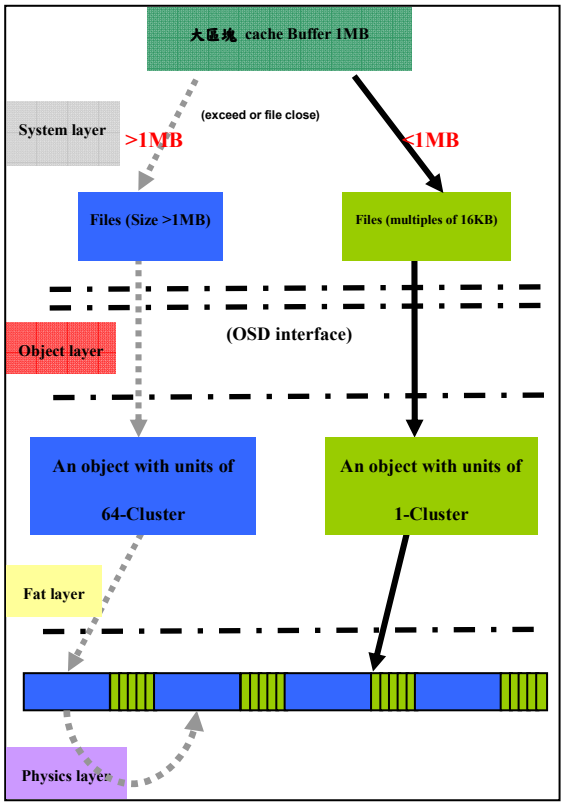

圖 2.3 FAT+ 驅動程式工作方塊

FAT +檔案配置方式規劃分為四個部份：

A. System Layer

一般的應用程式由系統提供的系統呼叫開啟一個檔案開始寫入，假設大區塊為1MB，小區塊為16KB，當使用者連續寫入檔案超過檔案系統快取1MB，檔案系統就把此檔案歸類成為大區塊的儲存磁區。當使用者關閉一個檔案但已寫入完畢而未超過1MB的檔案系統快取時，檔案系統將此檔案歸類成為小區塊的儲存磁區，系統初步規劃使用兩個分類來加快寫入的速度，在應用上如果需要也可加入更多的快取分類已達到應用上的需求。

B. Object Layer

物件層可讓原本規劃的 FAT 檔案配置方式成為 Object Storage Device(OSD)的一部份，可將檔案以物件的方式儲存。而 OSD 具有以下優點：

(1) 以物件為存取單位，解決不同平台間存取sharing data 的問題。

(2) 除了OSD裝置本身可以加入存取控制外，在資料安全控管時透過metadata server來加入憑證可讓系統做更彈性、更複雜的安全管理，不用擔心異質平台不易相容的問題。

(3) 物件為儲存單元，物件的屬性可詳細的描述儲存在裝置中的資訊，透過此特性在OSD裝置中加入對物件的前處理可以讓資料的儲存更有智慧。

例如對於一個文字類型的物件，我們可以在前處理時加入壓縮/解壓縮的功能以達節省空間的目地，而對於多媒體類型的物件可以使物件在儲存裝置中以連續空間的方式配置以增加存取效率，這些都是傳統儲存裝置不易達成的。

C. FAT Layer

FAT 配置層相似於傳統的 FAT 檔案配置，遵循傳統的 FAT 規格將其相關的檔案訊息寫入相對應的資料結構中。如此一來，此硬碟不管拿到 PC 上或者在本系統上都可以正確的讀取檔案，符合向下相容的需求。唯一的差異在於，本系統上將資料寫入磁碟中採用的是FAT+配置方式。FAT layer 是將系統層所提供的大區塊單元與小區塊單元資料存放到實體磁碟上的作業程序。

D. Physics Layer

實體層對應於硬式磁碟機的驅動程式，此驅動程式接受 FAT layer 所下達的磁區做寫入或者讀取的動作，如果執行的平台使用在移動式電子設備對應的儲存裝置是快閃記憶卡(Flash Card)只需修改系統相關實體層的驅動程式模組，就可與 FAT layer 銜接。

FAT+ Disk Layout

在嵌入式的應用中，這種應用層的特定性與單一性尤其顯著，有許多嵌入式系統只做單一工作，即使不是特地的嵌入式應用，人為的工作職場也是決定儲存內容的重要因素，例如工業用 PC 依工作場合所儲存的資料性質大小變化不大，醫學儀器所得資料，(如 CT 圖、X 光片等)，在醫院內大量產生，其資料物件的大小並非亂數，有其固定的格式可循。圖 2.4 說明了此種可變性，必須使用前審慎評估，一經設定，此檔案系統在使用中是不能更改的。圖 2.5 更進一步說明小資料區塊可以是介於 0 與 1 之間，亦即幾個大區塊之後才出現一個小區塊，當小區塊數量為 0 時，此檔案系統只存在大區塊。

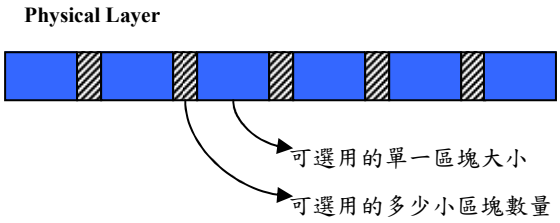

圖 2.4 可選用的區塊參數設定

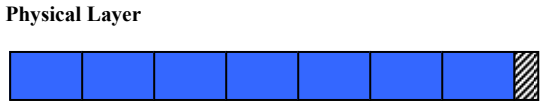

圖 2.5 斜線區塊數量可以小於 1

例如儲存裝置應用在多媒體相簿時，電子相簿內照片必然佔據大部分的儲存空間，每張相片也許介於 512KB~2MB 之間，在每張相片隨之附加一個簡短的文字說明，每一個說明檔在 16KB 以內，當大於 16KB 的檔案將寫入到大區塊單元，在實體層上可以規劃如圖 2.6 所示，兩個大區塊單元後緊跟著兩個小區塊單元。

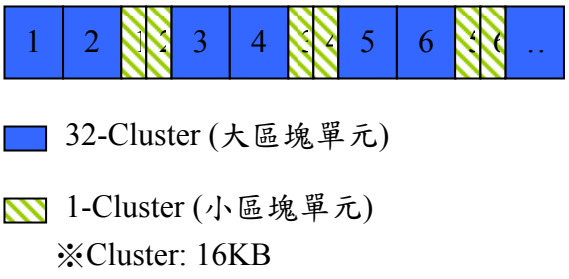

圖 2.6 多媒體相簿磁碟配置

又例如此平台是一個影片播放儲存裝置，在裡面的影片有上百 MB，大區塊單元可使用 10MB 或者更大的單元量來儲存，並且連續配置多個大區塊單元，也可將小區塊單元省略配置。單配置大區塊單元，影片播放儲存裝置磁區分配如圖 2.7 所示。

圖 2.7 影片播放器檔案配置

藉由 FAT+所提供的大區塊單元與小區塊單元兩種儲存單元，將固定檔案大小的檔案存放在特定的位址。假如大區塊單元內的資料被刪除，下次存進的資料還是大區塊單元，不會被小區塊單元取代，可以有效減少檔案刪除/新增後傳統 FAT 檔案配置產生嚴重的破碎情況。但 FAT+並非如此的完善，有許多小缺失是採用 FAT+檔案配置會發生的情形，例如檔案內部破碎問題，可藉由調整 FAT+對於平台應用的參數來改善。

FUSE 檔案系統

FUSE (Filesystem in user space) [18] 檔案系統提供使用者一個開發 Linux 檔案系統的框架，給開發者於使用者空間上可以快速的開發與測試檔案系統，在 Linux kernel 2.6.14 以後的版本已正式納入主流的核心樹(kernel source tree)內，初期雛型系統的 FAT+檔案系統就是架構在此之上。圖 2.8 為 FUSE 架構，FUSE 檔案系統架構在兩個主要的工具，一個是建在 Linux 核心模組內的 FUSE 檔案系統模組，另一個是架構於提供 FUSE 檔案系統相關函式給開發者使用的 FUSE 函式庫。

FUSE 函式庫實現的檔案操作大多與 Linux 核心中 VFS 的檔案操作相關，如此的設計可以方便開發者以後將檔案系統納入 Linux 核心內部，本文所提出的 FAT+檔案系統也是實現在 FUSE 上。除了開發方便外，還有利於將程式碼移植到其他平台上執行。

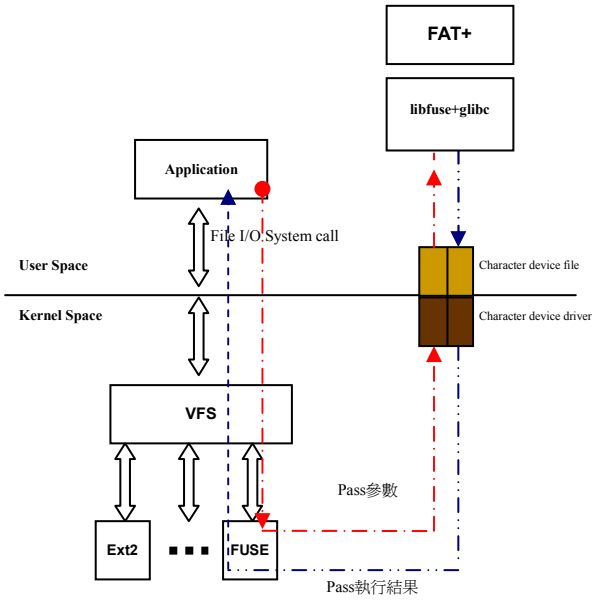

圖 2.8 為 FUSE 架構

核心內的 FUSE 模組是一個標準的 Linux kernel module，不同於一般本地端檔案系統，一般的檔案系統在接受來自 VFS 的命令後，即向更下層的 block device driver 請求讀寫的動作以取得期望的檔案資訊。而 FUSE 則是將 VFS 傳下來的參數 Pass 到 FUSE 實作的一個模擬字元裝置上，而此字元裝置會被映置到使用者空間可以操作的/dev/fuse 裝置檔案，以後來自 VFS 送向 FUSE 的要求都將轉送到使用者空間可操作的/dev/fuse 裝置檔案，接下來想在使用者空間開發檔案系統的工作者，即可透過/dev/fuse 與核心中的 FUSE 檔案系統通訊，可以取得 VFS 傳送給核心內 FUSE 的資訊，對可以將處理完資訊的結果回傳給核心內 FUSE 模組。使用者空間有一個 FUSE 專屬的函式庫，用來抽象化檔案系統的操作函式，藉此解析/封裝參數與核心內的 FUSE 模組溝通，使開發者再撰寫檔案系統程式時感覺像在核心內開發。

FUSE 函式庫提供開發者提供相關的檔案系統操作函式，此資料結構名稱為 fuse_operations 以下將說明此資料結構中需實現的檔案操作函式:

一、取得屬性(getattr) ：此函式將會傳入所要得知屬性的檔案路徑，判斷此檔案路徑是否存在。

二、取得檔案屬性(fgetattr) ：此函式將會傳入所要得知屬性的檔案名稱，判斷此檔案是否存在。

三、讀取目錄(readdir) ：此函式將會傳入所要讀取的目錄名稱，判斷此目錄是否存在。

四、讀取(read) ：函式會傳入要讀取的檔案名稱、長度與偏移量，並且提供一個位址來存放讀取的檔案資料。

五、寫入(write) ：此函式會傳入要寫入的檔案名稱、寫入資料的位址、長度與偏移量。

其他還有:建立(create)、開啟(open)、釋放(release)、開啟目錄(opendir)，等等。

三、平台建置與測試

硬體目標平台、開發環境與工具

本文採用的是 ATMEL 公司所生產的 AT91SAM9260-EK 開發版。開發版上有一組 RJ45 網路介面、三組 RS232 通訊埠、USB2.0 主機端、客戶端介面、JTAG 介面、64MB SDRAM、256MB NAND Flash 與 AT91SAM 系列 ARM base 的微控制器，控制器內含 ARM926EJ-S 處理器，此處理器包含了 8K byte 指令快取與 8K byte 資料快取，在 190MHz 操作頻率可達到 210 MIPS 的效能，微控制器內含 8K bytes SRAM 、32K bytes ROM、SDRAM、NAND Flash、Compact Flash 控制器，周邊控制器包含了 USB Full Speed Host and Device 介面、10/100 乙太網路、影像掃描

介面、多媒體儲存介面(MMC)、序列埠(UART)介面、SPI。

使用 FUSE 建置 FAT+

FAT+實作過程考慮到能否輕易的移植到任何的系統上，在實作過程中將程式碼分為三大部分如圖 2.4 所示，實體層的驅動程式對於硬式磁碟機、SD/CF 記憶卡等裝置的驅動模式皆不相同。在整個架構中最底層使用抽象的操作方式，以讀取或寫入 Sector 為單位，此層可實作磁碟快取，以減低因為磁碟操作時所需的等待時間。FAT+的磁碟配置演算方式位於中間層，與硬體裝置無關，在此將判斷寫入或讀取的資料位於磁碟何處，並依照 FAT+所規劃的大區塊單元與小區塊單元對磁碟裝置存取。最上層與 FAT+銜接的應用程式可使用 FUSE 檔案系統所提供的 Framework 來與 Linux 檔案系統連接，也可提供一般的應用程式直接操作；FAT+實現的部分有：建立/刪除資料夾、建立/刪除檔案、資料夾轉換、列出資料夾內容、讀取/寫入檔案內容，本文中 FUSE 實現了 FAT+的列出資料夾內容(取得檔案屬性)，讀取寫入檔案內容，如此就可以透過 FAT+將資料寫入進而符合本文所提出 FAT+的磁碟規劃。

本文中所實現的 FAT+雛型架構於 Linux 系統之上，初期開發在一般 Linux 系統上也實作一個以 FAT+磁碟配置的應用程式，應用程式有一小的測試對 FAT+的檔案做開啟、寫入、讀取、關閉、刪除與建立資料夾等基本操作都有相對應的測試，以便使用 FUSE 系統時能直接加入相對應的函式呼叫。由於 FAT+是架構於 Linux 系統上的裝置，對於磁碟的儲存方式目前是對一個檔案裝置直接操作；在系統上安裝儲存裝置(目前為硬碟)， Unix 架構下任何的周邊都被當作檔案，因此 FAT+只要對此裝置檔案作儲存或者讀取的操作即可對實際的磁碟操作，例如目前的磁碟機位於 IDE 插座的第三個插槽，在 Linux 下將會是/dev/hdc 此裝置檔案。

驗證與測試

本文架構於 Linux 系統之上，由於作業系統提供一抽象層，初期的開發與測試皆在 PC 上，在此將 Linux 系統架安裝在 VMware 軟體所提供的虛擬電腦上，在 VMware 裡模擬一個實體化的硬碟，FAT+的操作都直接對這顆硬碟，也可藉由掛載外部硬碟如 USB 隨身碟來測試與 Windows 系統上 FAT 的相容性。

使用終端機連線至 Linux 系統，另一個終端機執行 FUSE 所架構的 FAT+檔案系統，在此執行 FAT+與 FUSE 銜接的應用程式，這程式為目前實現下的檔案操作方法：

```
static struct fuse_operations pfat_oper = {
.getattr        = pfat_getattr,
.fgetattr       = pfat_fgetattr,
.readdir        = pfat_readdir,
.open           = pfat_open,
.read           = pfat_read,
.write          = pfat_write,
.create         = pfat_create,
.opendir        = pfat_opendir,
.releasedir     = pfat_releasedir,
.release        = pfat_release,
};
```

在這個 fuse_operations 的資料結構內，定義著 FUSE 系統呼叫所對應的檔案操作函式指標，程式開發者需將相對

應的函式在此告知 FUSE 系統，本文目前只實作上述的操作方式，可提供平面式的 FAT 檔案操作，來驗證 FAT+寫入的過程。終端機進入掛載點建立檔案，執行建立檔案開啟檔案等一般檔案操作，在 FUSE 所執行的終端機會顯示一些操作訊息。指令為：

./fusepfat -d /mnt/fuse

程式名稱 (除錯資訊) (掛載點)

驗證是否有正確的將資料以 FAT+所規劃的寫入連續的磁碟空間，將透過 Linux 系統與 FUSE 配合 FAT+分別寫入五個檔案，分別為 a.dat 0.4MB、b.dat 0.5MB、c.dat 1.1MB、d.dat 1.4MB、e.dat 0.6MB 以來驗證 FAT+是否正確。

因此使用(dd)命令分別寫入五個檔案，(if)輸入檔案、(of)輸出檔案、(bs)區塊大小、(count) 區塊大小的數目個數。由 dd 所提供的資訊可得知建立此檔案的時間、拷貝時間與寫入的速度。例如寫入 0.4MB 的檔案，即是寫入 410 個 1K 的檔案：

dd if=/dev/zero of=a.dat bs=1k count=410

在 FAT+規劃裡將第 128 Cluster 開始存放屬於 FAT+的磁碟配置方式起頭為大區塊單元，接下來為小區塊單元，小區塊單元配置起點為，大區塊單元起始點第 128 個 Cluster 偏移 16 個 Cluster 位於第 144 個 Cluster，有此推論以上五個檔案寫入的配置方式如圖 3.1 所示為：

a.dat 0.4MB 配置到小區塊單元，小區塊單元開始由第 144Cluster 連續配置 8 個 Cluster 所以 a.dat 佔用掉 144、145、146、147、148、149、150、151 這個 8 個連續的 Cluster 空間。下次小區塊單元開始的位址為 152 Cluster 再加上大區塊單元的 16 個 Cluster 偏移量，應為第 168 個 Cluster。

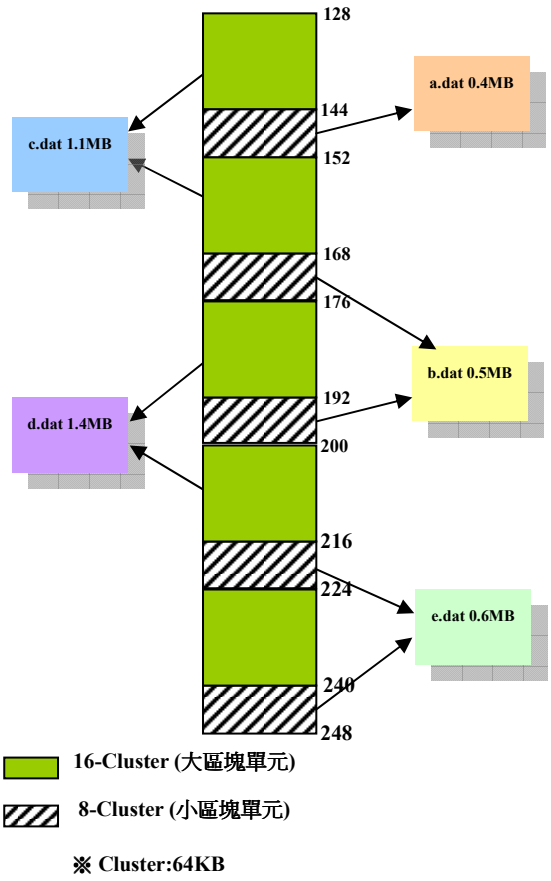

圖 3.1 檔案配置概略圖

由於 Linux 系統本身支援 FAT 檔案系統，可由 Linux 系統來掛載剛剛所寫入的磁區能否相容 FAT 檔案系統，藉此驗證是否可在一般 Windows 作業系統上讀取。

四、結論

一個嵌入式儲存系統而言，儲存的配置對於往後的操作與讀取占了非常關鍵的因素。本文所提出的 FAT+對於既有的檔案系統擁有向下相容的性質，且在儲存配置上又可以對檔案的處理提供參數做最佳化。

本文架構於 Linux 系統之上，一個好作業系統能隱藏平台的硬體相依性，因此在程式開發的過程中可先在一般 Linux 系統測試。對於使用 FUSE 在 User Space 開發程式除了擁有較好的系統資源可以使用外，眾多有用的開發環境亦有其相當助益，但是效能的欠缺，讓它在大部分的應用中，終究無法成為最終的成品。此外在實作的過程考慮到是否能將 FAT+磁碟配置移植到不同的作業系統上，目前除了本文所提出的 Embedded Linux 系統外，也可以將 FAT+移植到嵌入式熱換置系統(Hot Swap System) [1] [2] [3]上執行。

雖然 FAT+對於特定的檔案儲存方式有著比傳統 FAT 檔案系統更為優良的儲存方式，但並不代表 FAT 檔案系統可以因此退而不用，一般的應用場合還是傳統的 FAT 的對於磁碟的使用率比較好，主要是因為在 FAT+使用大區塊單元與小區塊單元的規劃上比較偏向特殊場合應用。然而嵌入式平台通常專為某種特殊用途而規劃設計，在某些用途上資料的存取往往都是固定的大小，因此可以將 FAT+應用在此。

未來工作

本文使用FUSE是為了測試FAT+是可行的而且能在使用者空間能夠方便的實現、開發。但使用FUSE來實現FAT+的效能初步評估很有可能不佳，未來嘗試將原本user space的檔案系統以module的方式移到kernel space。

此外，由於目前儲存技術普遍存在些不易解決的問題，因此下一代的物件儲存技術受到各界的關切，而在前面提到物件儲存技術最少具有下列優點：

(1) 以物件為存取單位，解決不同平台間存取sharing data 的問題。

(2) 除了OSD裝置本身可以加入存取控制外，在資料安全控管時透過metadata server來加入憑證可讓系統做更彈性、更複雜的安全管理，不用擔心異質平台不易相容的問題。

(3) 物件為儲存單元，物件的屬性可詳細的描述儲存在裝置中的資訊，透過此特性在OSD裝置中加入對物件的前處理可以讓資料的儲存更有智慧。

因此未來物件技術以其前瞻性的技術，相較於傳統，物件儲存技術將會是較理想且實際的發展方向。

五、致謝

本文承蒙國科會計畫部分補助，計畫編號 NSC95-2221-E-218-028-，在此致謝。

六、參考文獻

[1] 蔡亮宙，“嵌入式熱換置系統 HS^{2}”，多媒體及通訊系統研討會，2006。

[2] 鄭匡志，“嵌入式系統核心模組與高速網路實作”，南台科技大學碩士論文，2005。

[3] 鄭紹歡，“嵌入式系統之行動任務的實

現”，南台科技大學碩士論文，2006。

[4] 魏如宏，“多層次搜尋方法應用在PACS之研究”，逢甲大學電子工程學系碩士論文，2005。

[5] JFS (http://jfs.sourceforge.net/)

[6] JFFS2(http://sources.redhat.com/jffs2/)

[7] Ext3 (http://en.wikipedia.org/wiki/Ext3)

[8] Cramfs(http://en.wikipedia.org/wiki/Cramfs)

[9] Romfs(http://en.wikipedia.org/wiki/Romfs)

[10] MTD(http://www.linux-mtd.infradead.org/)

[11] Ext2(http://en.wikipedia.org/wiki/Ext2)

[12] DICOM(http://www.sph.sc.edu/comd/rorden/dicom.html)

[13] NFS(http://en.wikipedia.org/wiki/Network_File_System)

[14] CIFS(http://en.wikipedia.org/wiki/CIFS)

[15] RFC 3720 Internet Small Computer Systems Interface (iSCSI)

[16] RFC 3783 Small Computer Systems Interface (SCSI)Command Ordering Considerations with iSCSI

[17] ANSI X3.221-1994

[18] FUSE(http://fuse.sourceforge.net/)