行政院國家科學委員會專題研究計畫　成果報告

網路磁碟儲存管理系統

計畫類別：個別型計畫
計畫編號：NSC92-2213-E-034-005-
執行期間：92 年 08 月 01 日至 93 年 07 月 31 日
執行單位：中國文化大學資訊科學系

計畫主持人：戴文彬

報告類型：精簡報告

處理方式：本計畫可公開查詢

中　華　民　國 93 年 11 月 2 日

行政院國家科學委員會專題研究計畫成果報告

網路磁碟儲存管理系統

Network Disk Storage Management Systems

計畫編號：NSC 92-2213-E-034-005
執行期限：92 年 8 月 1 日至 93 年 7 月 31 日
主持人：戴文彬 中國文化大學資訊科學系助理教授

一、摘要

本計畫延續先前之研究，針對在網路多使用者環境下的磁碟資料存取使用特性，研究資料分享、儲存效能、大量儲存、資料安全、網路通訊等的主題，設計一個網路磁碟儲存管理系統(NDSMS)**架構，並且加以實作。**

在本研究中，先設計一個磁碟資料分享技術，考量有多使用者的網路使用環境的資料存取問題，然後實作主要程式的資料結構以及相關效能函數，以作為建置此一管理系統的核心程式，提供有效的集中網路磁碟儲存的管理服務。

利用此一網路磁碟儲存管理系統架構之設計與實作，提供網路上之儲存資料的集合式管理，以達到高效率、高度分享的資料存取服務。

關鍵詞：網路磁碟儲存、儲存管理、Linux **系統**

Abstract

In the project, we devise and implement the Network Disk Storage Management Systems (NDSMS) proposed in [1][2]. The network disk systems are established to store data in multi-user network for various disk usages. The functions of the NDSMS systems focus on the practical usage types of the users. The data structures of the system variables have been devised to provide these functions for data manipulation. Related programs have been implemented as the core kernel of the NDSMS systems.

Keywords: Network Disk Storage, Storage Management, Linux.

二、緣由與目的

本計畫延續先前之研究[1][2]**，提出一個網路磁碟儲存管理系統**(Network Disk Storage Management System, NDSMS)**的架構，將網路環境下多使用者之大量缺乏管理的電腦儲存資料，根據其中的資料使用特性，以磁碟儲存形式為基本單位，加以整合管理。**

本計畫以此一網路磁碟儲存管理系統之設計以及實作為主。作法是以研究現有作業系統作為開發參考[3][4]**，根據網路通訊等資料的儲存特性**[5][6][7]**，設計一連結於網路上的儲存伺服系統，將網路上之儲存空間作集合式的管理，以實現一高效率、高度分享的儲存管理系統架構，並實際予以實作。**

目前相關技術大多著眼於儲存設備集中存放，透過網路存取，得到分享的優點[8][9]**。然而卻未針對儲存資料的有效管理與維護，進行深入的研究與設計。本研究著眼於此，認為有效與合理的資料儲存管理應該以用戶端的磁碟使用**(disk usage)**為單位，以此作為管理的主體。**

不過，此一系統的設計並不容易，主要是用戶端的磁碟使用涉及複雜的各種情況與要求，例如：讓不同檔案系統磁碟同時集中共存、使儲存資料達到高度共享使用、減少使用者間相同資料的重複儲存、提高大量資料的儲存效能、做到有效的資料備份及復原等管理、管理使用端資料將變得非常容易、做到網路資料流量的使用分析…等等。這些不同的情況與要求使得

儲存管理的設計更加困難。

本研究會先設計一個磁碟資料分享技術，考量有多使用者的網路使用環境的資料存取問題，設計主要程式的資料結構(data structure)以及相關效能函數(utility functions)，以作為實作此一管理系統的核心程式，提供有效的集中網路磁碟儲存的管理服務。

三、結果與討論

3.1 架構設計

在 NDSMS 架構中，必須要將儲存裝置集中管理，並能提供使用者要求所需的儲存空間以供使用。計畫中的做法如下：NDS 把儲存裝置看做是一個 volume 裝置，而每一個 volume 裝置皆被格式化為上層管理系統的資料格式，並且使用上層系統管理者來管理。當使用者向 NDSMS 要求建立所需要的儲存空間時，上層系統管理者會從被集中管理的 volume 裝置中，配置一個符合使用者要求的大小的邏輯磁碟區域，並把這個邏輯磁碟區域的進入點通知 NDSMS。日後使用者就可通過邏輯磁碟區域的進入點來存取或操作屬於使用者的邏輯磁碟區域。

要設計此一功能的磁碟儲存管理，必須要設計符合磁碟資料儲存特性的機制：分享式讀寫 SRW (Sharing Read/Write)。SRW 機制是把儲存空間看成是資料存放連結區塊，使不同的邏輯磁碟區域分享共同的資料，並保存各邏輯磁碟區域之間的差異性及共通性。

運用 SRW 機制來對邏輯磁碟區域進行在邏輯磁碟區域上最主要的幾項操作：

- 複製操作：一邏輯磁碟區域經複製後，仍能正常讀取到以前的資料，後來寫入的資料則會儲存於別處，而不會影響到原來資料的完整性。

- 回復操作：將一指定的邏輯磁碟區域的資料狀況恢復到上一次複製操作後的狀況。

- 清除操作：清除指定的邏輯磁碟區域各個版本間資料狀況的差異性，保留最後的共通性而成為一新的複製版本。

- 邏輯磁碟區域分享：一邏輯磁碟區域經複製操作後，其資料狀況就獲得了完整的保留，因而分享給其他的邏輯磁碟區域。

- 快速的建置與恢復邏輯磁碟區域：利用複製操作、回復操作、清除操作使因人為疏失而破壞的資料，在短時間內恢復。

利用 SRW 處理機制，使得資料的分享、儲存、備份變得更有效率，把儲存空間更徹底的使用。如此使得建置或管理電腦使用環境變得容易而且有效率，實現集中式網路磁碟儲存的管理。

3.2 系統實作

關於系統實作的主要程式之資料結構，分項討論如下：

- 超級區塊結構：整體管理系統的資料結構是由 ds_super_block 來表示的，在 super block 裡存放了這個管理系統重要資訊，用以存取系統中的任何磁碟資料。其中，重要屬性如：總體區塊數、總體保留區塊數、總體未使用的區塊數、總體未使用的 dnode 數、總體 dnode 數、、…等。(dnode 討論於後)

```
struct ds_super_block
{
    __u32   s_dnodes_count;
    __u32   s_blocks_count;
    __u32   s_r_blocks_count;
    __u32   s_free_blocks_count;
    __u32   s_free_dnodes_count;
    __u32   s_first_data_block;
    __u32   s_log_block_size;
    __u32   s_blocks_per_group;
    __u32   s_remain_blocks_count;
    __u32   s_dnodes_per_group;
    __u32   s_first_dno;
    __u16   s_dnode_size;
    __u16   s_block_group_nr;
```

```
    __s8     s_ds_name[  16  ];
    ......
};
```

• 存取磁碟資料的節點資料結構：每個磁碟資料皆由 dnode **來表示，從** ds_super_block **可以取得任何一個磁碟資料的** dnode**，進行讀寫動作以及相關操作。其中，重要屬性如：所屬的** disk **的編號、分配給該** dnode **的區塊階層樹中樹葉的總數、…等。**

```
struct ds_dnode
{
    __u16    d_disk_no;
    __u32    d_blocks;
    __u32    d_path[  DS_N_BLOCKS  ];
    __u16    d_mode;
    __u32    d_size;
    __u32    d_atime;
    __u32    d_ctime;
    __u32    d_mtime;
    __u32    d_dtime;
    __u16    d_gid;
    __u16    d_links_count;
    __u32    d_max_blocks;
    ......
};
```

• 區塊群組描述結構：其中，重要屬性包括如：存放區塊位元對映圖的區塊編號、存放 dnode **位元對映圖的區塊編號、索引節點陣列**(dnode **對映表**)**的第一個區塊編號、未使用的區塊個數、未使用的** dnode **個數、區塊中擁有的目錄個數、…等。**

```
struct ds_group_desc
{
    __u32    bg_block_bitmap;
    __u32    bg_dnode_bitmap;
    __u32    bg_dnode_table;
    __u16    bg_free_blocks_count;
    __u16    bg_free_dnodes_count;
    __u16    bg_used_dirs_count;
    __u16    bg_pad;
    ......
};
```

• 磁碟資料的 disk **進入點結構：其中，重要屬性包括如：**disk **的編號、存取權限、**disk **所選擇的** image **型態、存放紀錄** srw **的** dnode **編號、、、…等。**

```
struct ds_disk_entry
{
    __u16    de_disk_no;
    __u16    de_mode;
    __u16    de_image_type;
    __u32    de_origin_disk_no;
    __u32    de_srw_dno;
    __u32    de_disk_size;
    ......
};
```

• 記憶體中管理超級區塊之結構：其中，重要屬性包括如：管理系統所屬設備的識別碼、區塊大小、區塊可紀錄的 dnode **個數、每個區塊群組的區塊個數、每個區塊群組的** dnode **個數、指到根目錄在** buffer **的位置、…等。**

```
struct super_block
{
   struct list_head s_list;
    dslock_t          s_lock;
    u32               s_dev;
    S8                s_part[  15  ];
    S8                s_ds_name[];
    u32              s_blocksize;
    u8               s_blocksize_bits;
    u32              s_dnodes_per_block;
    u32              s_blocks_per_group;
    u32              s_dnodes_per_group;
    u32              s_dtb_per_group;
    u32              s_gdb_count;
    u32              s_desc_per_block;
    u32              s_groups_count;
    u32              s_dnode_size;
    u32              s_first_dno;
    u32              s_addr_per_block_bits;
    u32              s_desc_per_block_bits;
    u32              s_ver_per_block;
    u32              s_entry_per_block;
    ......
   struct buffer_head     * s_sbh;
   struct ds_super_block * s_dssb;
   struct buffer_head ** s_group_desc;
    struct dnode           * s_root_dnode;
    struct list_head          s_disk_entrys;
    u8                         s_state;
```

```
    u16             s_active_disk_count;
    u16             s_disk_count;
    ......
};
```

• **記憶體中管理磁碟資料** dnode **之結構：其中，重要屬性包括如：在** dnode_hashtable **中的連結、設備代碼、磁碟資料** dnode **的編號、**inode **的許可權、**inode **在做** IO **時的區塊大小、**dnode **所在的區塊群組、…等。**

```
struct dnode
{
  struct list_head    d_hash;
    u32               d_dev;
    u32               d_dno;
    u16               d_mode;
    u32               d_blksize;
    u32               d_block_group;
    u32               d_next_alloc_goal;
    u32               d_prealloc_block;
    ......
};
```

• **記憶體中管理磁碟** disk **之結構：其中，重要屬性包括如：在** disk_hashtable **中的連結、在** super block **中的** disk **快取連結、**device **代碼、**disk **存取權限、**disk **的編號、版本的數量、…等。**

```
struct disk
{
    struct list_head        dk_hash;
    struct list_head        dk_list;
    dslock_t                dk_lock;
    u32                     dk_dev;
    u16                     dk_mode;
    u16                     dk_state;
    u8                      dk_lba2b;
    u32                     dk_disk_no;
    u32                     dk_max_block;
    u32                     dk_version_count;
    struct dnode            * dk_path;
    struct dnode            * dk_srw;
    struct super_block      * dk_sb;
    struct ds_disk_entry    * dk_entry;
    struct buffer_head      * dk_buffer;
    ......
};
```

• **記憶體緩衝區結構：其中，重要屬性包括如：在** hash **中的連結、在** lru **中的連結、在** dirty buffer **中的連結、狀態** flag**、實際的資料連結、、、…等。而狀態** b_flag **有多種狀態，表示** buffer **的不同操作使用情況。**

```
struct buffer_head {
    struct list_head    b_hash;
    struct list_head    b_lru;
    struct list_head    b_dirty;
    volatile char       b_flag;
    unsigned int        b_dev;
    unsigned long       b_blocknr;
    unsigned short      b_size;
    unsigned short      b_rtimes;
    dslock_t            b_lock;
    char                * b_data;
    ......
};
```

以上為主要資料結構, 最重要的部份在於 SRW **機制得設計以及實現。**

相關效能函數，較關鍵的幾個討論如下：

• **函數** ds_get_block(dblock)**：計算** dblock **所經過的間接區塊的位移，找出所經過的間接區塊。若中間的區塊找不到，代表可能此** dblock **尚未配置，則對找不到的區塊開始分配新的區塊，直到到達階層深度為止。**

• **函數** ds_get_block(dblock)**：包含取得** dblock **區塊，另外也包括：讀取分享的區塊，得到擁有的區塊等處理。**

• **函數** ds_load_dnode( )**：包含將磁碟資料做載入的操作處理。**

• **函數** ds_merge_dnode( )**：包含將磁碟資料做各版本的** dnode **合併在一起之處理，使最後的** dnode **能完全擁有其** block**。並且要使** block **的擁有權穩定，不易發生錯誤。而且當版本太多時，就會有許多不使用的區塊，所以必須把其清掉，以免佔用空間。**

• 函數 ds_split_dnode( )：包含將磁碟資料做複製的操作處理，可將一磁碟區域當時的資料狀況保存下來。一磁碟區域經複製操作後仍然能夠正常讀取到以前的資料，但是後來寫入的資料則會被 SRW 機制儲存於別處，而不會影響到原來資料的完整性。

• 函數 ds_recover_dnode( )：包含將磁碟資料將一磁碟區域的資料狀況恢復到上一次複製操作後的狀況。一磁碟區域經回復操作後，SFS 會釋放自上一次複製操作後到現在為止所寫入的資料

• 函數 ds_perge_dnode( )：包含將磁碟資料做清除的操作處理，將一磁碟區域的數個指定的複製操作的版本，清除各個版本之間資料狀況的差異性而保留最後的共通性而成為一新的版本。新的版本保留最後共同的資料狀況，而釋放不再參考到的資料。

四、計畫成果自評

本計劃完成之工作項目包括：

- 設計一個網路儲存管理系統架構。
- 設計網路儲存資料高度分享存取(SRW)機制。
- 實際實作出程式資料結構以及相關效能函數，以作為實作此一管理系統的核心程式。

本計劃之成果對學術研究之貢獻如下：提出一個網路儲存管理系統的新架構，並建置實作出相關系統，結果顯示其效能得到驗證。計劃的結果具有實際價值，可提供相關學術研究參考。相關成果已在研討會或期刊上發表[2]，並完成相關的系統實作。

有關後續研究方面，這個新架構能擴展至其他的問題的解決，有進一部探討的必要。特別是透過無線網路存取方式的資料傳輸、同步、維護、備援等問題，將作為本計劃日後的延續研究。

五、參考文獻

[1] 戴文彬, NSC 91-2213-E-034-004-, Linux-based 網路多磁碟儲存系統, 2002.

[2] 戴文彬,石旭本,張志豪, " Linux-based 網路磁碟儲存管理系統", 電子化企業經營管理理論暨實務研討會, E33, 彰化, 2003.

[3] Card, R., Dumas, E., and Mevel, F., The Linux Kernel Book, John Wiley & Sons, 1997.

[4] Beck, M, et al., The Linux Kernel Intervals, Addison-Wesley, 1996.

[5] Kleiman, S., "Vnodes: An Architecture for Multiple File Types in Sun Unix", Proc. Summer USENIX Conf., pp.260-69, 1986.

[6] Knowlton, K. C., "Fast Storage Allocator", Communications of the ACM, 8(10) , pp.623-625, 1965.

[7] McKusick, M., et al., "A Fast File System for Unix", ACM Transactions on Computer Systems, 2(3), pp.181-197, 1984.

[8] Silberschatz, A, and Calvin, P., Operating System Concepts, 4th Ed., Addison-Wesley, 1994.

[9] Tanenbaum, A., Operating Systems: Design and Implementation, Prentice Hall, 1987.