可維護多個作業環境的管理系統之設計

The Design of a Management System for Multiple Operating Environments

石旭本
國立政治大學 資訊科學系
rudolph@fetnet.net

陳正佳
國立政治大學 資訊科學系
chencc@cs.nccu.edu.tw

摘要

本文主要目的是提出一套全新設計的多作業環境管理系統（COEMS），以解決現有系統，僅著重在單一作業環境的維護、缺少作業環境日誌功能、以及沒有容量管理能力等缺點。COEMS 結合現有的作業環境維護技術、網路儲存技術以及裝置虛擬化技術等優點，將多台用戶電腦的作業環境集中於一或多台伺服器來進行管理與維護，成功克服傳統技術在維護作業環境的困難，使得在網路環境中，維護多台電腦的作業環境的需求變得可行。此外，利用容量管理技術（Volume Management）以及寫入同步複製技術（Copy-On-Write），COEMS 可同時維護多台電腦的作業環境日誌，並可提供資料分享，使得作業環境的維護比傳統技術更簡單有效率。最後，我們以 Linux 作為開發環境，實作出一個 COEMS 原型，來驗證 COEMS 系統設計之正確性以及可行性。

關鍵詞：作業環境維護、網路儲存、裝置虛擬化、容量管理、寫入同步複製。

Abstract

We propose a newly designed management system called COEMS for managing multiple operating environments. Its aim is remedy the shortcomings of existing systems, most of which suffer from the capability of maintaining only a single operating environment (OE), as well as the inabilities of logging changes and sharing volumes of managed OEs. By combining the advantages of current techniques of OE maintenance, network storage and volume virtualization, COEMS now can simultaneously manage multiple heterogeneous OEs from different computers in one or more centered servers, and makes feasible the maintenance of multiple OEs via network. Meanwhile, by virtue of adopted virtual volume management and Copy-On-Write techniques, COEMS is able to maintain the logs and share volumes from different OEs, and thus makes the maintenance of OEs easier and more efficient than traditional techniques. Finally, an actual COEMS prototype is built on the Linux platform to ensure the feasibility and correctness of our design.

Keyword: Operating Environment Maintenance, Network Storage, Virtualization, Volume Management, Copy-On-Write.

一、緒論

電腦的使用，無時無刻都有資料遺失的危險。人為破壞、硬體故障、軟體錯誤、電腦病毒...等因素，都有可能導致儲存裝置失效或是系統當機，而造成重要資料損毀。是故，維護資料的安全以及作業環境的妥善，是維護電腦正常運作的不二法門。

所謂的作業環境是指運作中的電腦系統的瞬時狀態。就軟體角度而言，它包含硬體組態、作業系統、與應用軟體等三部份。作業環境與作業系統的重要差異是多台電腦可使用相同作業系統，但卻構成許多不同的作業環境；此外，同部電腦在不同時點也可構成不同作業環境。而所謂的作業環境維護技術，是指能夠對作業環境進行管控，以增加作業環境的完整性以及安全性，並且在作業環境遭遇意外時，能夠快速地回復至正常運作狀態的相關技術。

作業環境的數量通常與電腦數量成正比。這是因為作業環境是存在於個別的電腦主機之中，而每部電腦在系統組態、作業系統、與安裝之應用軟體等方面多少都會有所不同，因而其作業環境也不盡相同。相較於作業系統之具備一致性，多台電腦的作業環境之維護因其數量與歧異性而變得更加繁瑣與耗時。舉例而言，若有 50 台電腦需要安裝 Office XP，假設平均每台需要 20 分鐘安裝，則需要花費 1000 分鐘或者約 17 個工作小時才能完成所有電腦的安裝。

為克服作業環境維護的繁瑣與耗時，已有許多維護技術被開發出來。這類的技術通常具有將某個時間點的作業環境的狀態保存下來的能力（即快照(Snapshot)作業環境）；有些技術甚至能夠將作業環境隨著時間的連續狀態變化保存下來（即作業環境日誌）。而當意外發生時，這類的技術能夠快速地將毀損的作業環境的資料狀態回溯到意外發生之前的某個時間點的狀態（即回溯(Rollback)作業環境）。而功能更強大的維護作業環境的技術，甚至能夠管理不同作業環境之間的資料狀態的分享（即作業環境之間的 Image Sharing）。

作業環境的傳統維護模式有二。一種是平時僅維護少數幾個具有相同或不同作業系統的原始或非原始作業環境，當有安裝需求或既有作業環境遭遇意外時，則利用快速複製技術重裝特定作業環境。這種方式的最大缺點是重裝的作業環境與原先作業環境可能具有許多差異，以致於會導致原來作業環境的資料遺失。因此除非可以忍受資料遺失，或遇到無法挽回的損壞，否則此種模式並不建議單獨採用。另一種模式則是利用系統提供的功能將某個時點的作業環境狀態備份下來，而當意外發生時，則將毀損的作業環境回溯到意外發生之前的某個時點的正常狀態。此種模式的特點是每部電腦均需個別維護且互相獨立，因此我們無法動態的將某部電腦的作業環境分享給其他電腦。值得一提的是實際維護多部電腦時通常可同時使用兩種模式而達到個別模式無法得到的功能，例如我們可複製某部電腦的瞬時狀態再快速安裝至其他多部電腦。然而此種方式仍然無法克服安裝時間與電腦數量成正比之夢魘。

目前主流的作業環境維護技術包括有：Norton Ghost [16]、Venturcom BXP（Diskless Windows 2000, XP）[21]、Windows Live CD & Linux Live CD [2][12]、DRBL（Diskless Remote Boot in Linux）[7]、LTSP（Linux Terminal Server Project）[13]。這些作業環境維護技術，雖然能夠達到維護作業環境的效果，但是如同表 1 所做的各項比較所示，尚有加強和擴充的空間。以下分別就三個方向，說明現存作業環境維護技術所存在的問題。

(一) 著重於單一作業環境的維護

此類技術以 Ghost [16]與 Live CD [2][12]為代表。其操作模式屬前面所提之第一種，亦即此類工具支援將作業環境快照為一影像檔以為備份，而當系統出意外時，則可快速將備份重裝至其他或原先電腦。

以這類技術維護的電腦系統，其作業環境大部分都儲存在電腦本身直接連接的個別儲存裝置（Hard Disks, FLASH Disks or Bootable CDs）。因而在多台電腦的使用環境中，相關作業環境將分散於不同的主機。而這導致的最大維護問題是：當意外發生時，管理人員就必須花費大量的時間，逐一地依照個別主機的狀況來維修作業環境。作業環境的分散及差異，使得維護多台電腦的作業環境的工作變得困難而且耗時。

雖然，有越來越多的作業維護技術，利用網路

或者是網路儲存技術，來解決傳統式直接連接儲存所存在的問題。但是，像 BXP[21]、DRBL[7]以及 LTSP[13]這幾個技術所做到的，僅僅是利用網路的互連特性，將單一且一時的作業環境分享給多台電腦而已，並沒有做更進一步的運用，實在相當可惜。

(二) 缺少維護作業環境日誌的能力

現存的作業環境維護技術只著重於維護少數個別單一且一時的靜態作業環境，而缺乏維護由大量動態作業環境所構成的作業環境日誌的能力。雖然，我們可以利用 Ghost 將作業環境隨著時間的變化快照為多個影像檔案的靜態技術，來達到日誌的效果；但是，以靜態技術維護個別作業環境日誌，所需的額外的工作反而會造成日誌維護的不便以及困難。而 BXP 雖然能夠集中管理同質的作業環境，但是以檔案的方式來管理作業環境的作法缺乏彈性，也缺少日誌的能力。至於 Live CD、LTSP 以及 DRBL 則因為本身架構的關係，更是缺少維護作業環境日誌的能力。

(三) 缺少容量管理的能力

現有作業環境維護技術多著重在維護個別作業環境的正常運作之功能需求上，而少有考量如何使個別作業環境共享共用儲存容量。然而網路儲存的特性，除了能夠解決傳統式直接連接儲存所存在的作業環境個別維護問題之外，藉由網路集中管理儲存裝置，更可以解決資料分散的問題，此外利用裝置虛擬化技術，更可以使多部電腦的作業環境共享共用儲存容量。雖然 BXP 也使用了相同的技術（網路儲存技術、裝置虛擬化技術）來管理多台同質的電腦的作業環境，但是卻使用檔案的形式來管理作業環境，缺乏類似 SAN 中以區塊(Block)為單位的容量管理（Volume Management）能力[3][6][9][15][17]，所以尚不能對作業環境進行快照、回溯等操作，是其美中不足的地方。

本研究即是針對現今作業環境維護工具均具有前述三大問題的某些缺點，而提出的一套新的解決方案。我們稱此方案為 COEMS (Client Operating Environment Management System, COEMS）[1]。COEMS 主要的設計概念為結合現有作業環境維護技術、網路儲存技術、以及裝置虛擬化技術，集中多台用戶電腦的作業環境於網路上的伺服器，以便於在網路環境中維護以及管理多台電腦的作業環境。同時 COEMS 利用容量管理技術（Volume Management）以及寫入同步複製技術（Copy-On-Write），以維護多台電腦的作業環境的日誌與資料分享，使得作業環境的維護比傳統技術更簡單更有效率。

二、COEMS 之系統架構

如圖 1 所表示，COEMS 是由三種不同腳色的網路節點組成: COEMS Manager、COEMS Server 以及 COEMS Client（以下簡稱 Client）。COEMS Client 即是一般的電腦系統，其作業環境是由 COEMS 所維護；COEMS Server 則是用於儲存與維護 Client 端的作業環境；至於 COEMS Manager，則是用於整個 COEMS 的管理與操作。

而 COEMS 的三種網路節點則分別是由以下五種元件的部份部署而成：Network Bootstrap Program（NBP）、Network Volume Service（NVS）、Virtual Device Driver（VDD）、Client Volume Manager（CVM）以及 Virtual Volume Manager（VVM）。有關這 5 個元件的功能與設計將延至下節再加以說明。 COEMS 中的 Client 部署有 NBP 元件、NVS 元件與 VDD 元件，使得 Client 能夠透過網路開機，並且利用裝置虛擬技術，來存取 COEMS Server 中的作業環境；COEMS Server 部署有 NVS 元件與 VVM 元件，使得 COEMS Server 能夠集中儲存並且維護多台 Client 的作業環境；而 COEMS Manager 中則部署有 NVS 元件、與 CVM 元件，使得 COEMS Manager 能夠透過網路來對 COEMS Server 進行操作管理的請求，以達到管理整個 COEMS 之效果。

三、COEMS 之主要元件架構

COEMS 是由五種元件部署而成。以下針對各元件的任務、功能、組成、與運作方式加以具體說明。

(一) NBP 元件

NBP 元件（圖 5），主要負責於 Client 開機階段，透過網路與 COEMS Server 溝通以取得作業環境的儲存資源，並且使用裝置虛擬化技術，將作業環境的儲存資源虛擬成為本地端的儲存裝置，藉此使得 Client 能夠順利地完成開機程序。而 NBP 元件，其實就是 PXE [8]中的 Network Bootstrap Program，只要完全依照 PXE 標準（V2.1 或更新的版本）來設計，就可以在 X86 架構下，實現網路開機。

當電腦開機時，在完成 POST（Power-On-Self-Test）階段後，BIOS 中 PXE 會透過 DHCP/BOOTP 來尋找可以提供網路開機的伺服器，並且下載網路開機程式（即 NBP）來進行網路開機。經過簡單的電腦初始化後，NBP 利用 PXE APIs 透過網路向 COEMS Server 取得作業環境的儲存資源，並且使用裝置虛擬化技術，將作業環境的儲存資源虛擬成為本地端的儲存裝置。之後 NBP 試圖載入 Bootstrap Loader ，並將執行權交給 Bootstrap Loader 以進行 OS 開機。 NBP 於開機階段會將存取要求（即 INT 13），經由網路重導到 COEMS Server 處理，藉此使得 Bootstrap Loader 能夠順利地將 OS 載入，並在完成開機程序後，將整個電腦的控制權交給 OS。

(二) NVS 元件

NVS 元件，位於 ISO 網路七層中的 Session Layer，主要負責提供 COEMS 成員網路溝通的服務，並且處理網路磁碟服務協定（Network Volume Service Protocol, NVSP）[1]。NVSP 是 COEMS 成員間共通的通訊協定，使用 UDP 作為傳輸協定，包含支援網路儲存、管理作業環境的網路協定。

NVS 元件是由以下兩個模組所組成的： (1) Session Manager：其主要的功能，就是維護 COEMS 成員之間的網路溝通狀態的詮釋資料（COEMS Metadata）。(2) NVSP Handler：其主要的功能，就是在網路環境中，使用 NVSP 來傳遞作業環境的操作請求與執行結果。

由於虛擬化技術的關係，Client 的作業環境的儲存資源，是被 COEMS Server 中的 VVM 所管理的。因此，就如同圖 2 所表示，每當 Client 或 CVM 需要對作業環境進行操作時，就必須透過網路並且使用 NVSP，來跟 COEMS Server 中的 VVM 要求其所要執行的操作，並且等待 VVM 將其要求處理完畢，將操作結果傳送回來，如此一個操作才能算是完成。為了完成整個操作要求，需要 NVS、VDD、CVM 以及 VVM 這些元件的相互合作，整個程序可以分為以下幾個步驟：

1. 當操作要求產生時，Client 中的 VDD 或者是 CVM 就必需要呼叫 NVS 的 NVSP Handler 的介面函式，來為其處理操作要求。
2. 當 NVSP Handler 接收到要求之後，會向 Session Manager 要求作業環境的 COEMS Metadata。如果 Session 並不存在，則初始化一個新的 Session。如果作業環境的狀態不允許接受目前的操作，則跳到步驟 8。
3. NVSP Handler 依據 COEMS Metadata 將操作要求封裝成 NVSP 封包，透過網路傳送給正確的 COEMS Server。
4. 當 COEMS Server 中的 NVS 接收的 NVSP 封包後，首先 NVSP Handler 會檢查該封包是否為合法的要求封包，並且向 Session Manager 要求相對應的作業環境的 COEMS Metadata。
5. NVSP Handler 依據 COEMS Metadata 向 VVM 提出 Virtual Volume 的操作要求（指呼叫 VVM 所提供的 Virtual Volume 的操作介面）。
6. VVM 處理 Virtual Volume 的操作要求，並傳回操作結果給 NVSP Handler。
7. NVSP Handler 依據操作結果更新 COEMS

Metadata，並將操作結果封裝成 NVSP 封包，透過網路送回給 Client 的 NVS。

8. Client 的 NVSP Handler 依據操作要求的結果更新 COEMS Metadata，並通知給 VDD 或者是 CVM 元件。

(三) VDD 元件

VDD 元件，主要負責處理遠端存取的工作。由於現今的作業系統在設計時，大都採用分層的概念，所以，只要在不變動溝通介面的情況下，不同層的實作是可以分離而不會相互影響[18][20]。因此，只要不變動檔案系統和裝置驅動程式（Device Driver）之間的溝通介面，在作業系統之中建立不真實存在的儲存裝置（Virtual Device），是可行的。所以嚴格地說，VDD 其實就是 COEMS 之中 Virtual Volume 專用的裝置驅動程式。

如圖 3 所表示，VDD 主要負責處理 Client 端對作業環境的 I/O 要求。當存取作業環境時，VDD 將 I/O 要求經由網路重導到 COEMS Server 進行處理。COEMS Server 在 I/O 要求處理完畢後，將 I/O 要求的結果送回給 VDD，完成整個存取資料的程序。以下是 VDD 處理 I/O 要求的詳細步驟：

1. 當檔案系統需要對作業環境（指 Virtual Volume）進行存取時，如果所請求的 Block 存在於快速緩衝區（Buffer）之中，則直接傳回該 Block，並結束該請求；否則檔案系統會透過呼叫驅動程式的系統介面，來將存取 Block 的要求加入目的裝置的驅動程式（指 VDD）的 I/O Queue 之中，等待驅動程式處理。
2. 當作業系統察覺 VDD 有 I/O 要求需要處理，會將 VDD 的佇列處理程序排班，使得 I/O 要求能夠被處理。當 VDD 獲得執行權時，會將佇列中的 I/O 要求依序重導到 COEMS Server 的 VVM 進行處理（呼叫 NVS 的 NVSP Handler 來處理操作要求）。
3. 待 COEMS Server 的 VVM 將要求處理完畢後，將其操作結果傳送回給 VDD（經由 NVS）。
4. VDD 依據操作結果，修改 I/O 要求的資訊，藉此通知檔案系統進行更進一步的處理。

(四) CVM 元件

CVM 元件是 COEMS 的使用者介面，而為了能夠在不同的作業系統執行，CVM 可採用 JAVA 為開發語言。如圖 4 所表示，CVM 主要負責將 Client 的作業環境的狀態資訊呈現給使用者；並且接受使用者對 Client 的作業環境的維護操作要求，將其重導到 COEMS Server 的 VVM 進行處理，待 VVM 將要求結果送回，再將操作的結果呈現給使用者。以下是 CVM 處理使用者操作要求的詳細步驟：

1. 當使用者需要對 Client 的作業環境進行操作時，可以透過與 CVM 的 GUI 互動，將操作要求通知 CVM。
2. CVM 將操作要求，重導到 COEMS Server 的 VVM 進行處理（呼叫 NVS 的 NVSP Handler 來處理操作要求）。
3. 待 COEMS Server 的 VVM 將要求處理完畢後，將其操作結果傳送回給 CVM（經由 NVS）。
4. CVM 依據操作結果改變作業環境的呈現，藉此通知使用者操作要求的結果，並等待使用者後續的要求。

(五) VVM 元件之設計

VVM 元件，是最重要的 COEMS 元件，主要負責集中管理以及維護 Client 的作業環境。如圖 6 所表示，VVM 將 COEMS Server 的儲存空間虛擬化成為一個 Virtual Volume Pool，並管理其中所有的 Virtual Volume。而 Virtual Volume 支援建置、刪除、設定、存取、Snapshot、Rollback、Image Sharing…等操作，以達到管理及維護作業環境的功效；並將外界真實的電腦以及作業環境，虛擬化成為 Client 以及 Virtual Volume 等可接受管理之物件，藉由對 Client 以及 Virtual Volume 實施管理操作，來反映真實的電腦與作業環境的情形，以達到管理效果。

在設計上，VVM 是由 Matrix、SFS、Virtual Volume Pool 與 Client Manager 這 4 個模組所組成的，以下 4 個小節，分別介紹這 4 個模組的功能，以及模組之間如何分工合作，來達到管理及維護作業環境的目的。

A. Matrix 模組

Matrix 主要負責管理 COEMS Server 中的儲存裝置的儲存資源分配。Matrix 可以將多個獨立的儲存裝置的儲存資源整合成為 Logical Block Space（LBS），並將其虛擬成一個大型的儲存資源－儲存母體（Storage Matrix）。而 LBS 則是由數個儲存裝置所共同組合而成的，其組合的數量依照 LBS 的模式而有所不同，LBS 的模式有 Normal、RAID-0、RAID-1、RAID-5…等。圖 7 所表示的是一個容量為 300G 的 RAID-0 模式的 Storage Matrix，其每一個組成的 LBS 都是 RAID-0 模式（2 顆硬碟作 mirror）。

Storage Matrix是由數個LBS所組合而成，並且可以隨時加入新的LBS來擴充容量。由於目前 Matrix使用 30bits來定址Storage Matrix，而Storage Matrix的Block size為 64K，因此Storage Matrix最大的定址空間為 2^{30}*64K=64T。

B. SFS 模組

一般來說，檔案系統能夠妥善地管理儲存空間，並且規劃安排資料的存放，使得資料能夠正確地儲存於儲存空間之中。因此，VVM 採用 Super File System（SFS），來規劃管理 Matrix 中的 Storage Matrix，並且分配儲存空間供給作業環境存放資料。而 SFS 名稱的由來，是因為 SFS 就像是所有的檔案系統的超檔案系統一樣。

如圖 8 所表示，可以看得出來 SFS 的架構受到 Linux 的 ext2 檔案系統 [19]相當深的影響，是一個索引方式的檔案系統，並且把資料區塊分成區塊群組（Block Group）。為了避免資料分散（Fragment），SFS 儘可能將 Virtual Volume 的資料區塊集中在同一個 Block Group 之中，以便把搜尋區塊時所需的時間降至最低（區塊之間的距離比較靠近），存取資料的速度也因而加快。每一個 Block Group 包含有：一個超級區塊（Super Block）的備份、所有 Block Group 的描述詞（Group Descriptor）的備份、區塊位元對映圖（Block Bitmap）、儲體索引節點的位元對映圖（Volume Index Node Bitmap, Vnode Bitmap）、儲體索引節點表格（Volume Index Node Table, Vnode Table）、以及資料區塊。

Vnode 主要的功能，是負責記錄資料區塊的分配情形。因此，VVM 利用 Vnode 來記錄每一個作業環境在某一個時間區間的資料狀態。而為了紀錄作業環境的日誌，Vnode 必須能夠紀錄資料狀態的差異性，因此 SFS 引用了虛擬記憶體管理機制“Copy-On-Write”的觀念，來處理 Vnode 的資料區塊的配置與分享。SFS 對待資料區塊就如同虛擬記憶體管理機制對待記憶體一般，一開始 Vnode 可能沒有配置任何的資料區塊或者是與其他的 Vnode 共同分享資料區塊，而直到 Vnode 中某一特定區域有資料寫入時，SFS 才為 Vnode 複製和配置所需的資料區塊。

如圖 9 所表示，SFS 使用 32 bits 的位址空間來定址區塊，但是其中的 MSB（Most Significant Bit）並不使用於定址位址，而是用來記錄資料差異性，稱為 Owner Bit。Owner Bit 主要是供 SFS 辨別是否需要引發 Copy-On-Write 機制，當 Owner Bit 為 1 時，表示其區塊為“專屬資料”，並不需要 Copy-On-Write；相反的當 Owner Bit 為 0 時，則表示該資料區塊是“分享資料”，當要將資料寫入該區塊時，需要 Copy-On-Write 機制來配置新的區塊後，才能寫入資料成為“專屬資料”。SFS 有了記錄資料差異的能力後，Vnode 就可以經由 Spawn 操作來產生 Child Vnode，並將本身所索引的資料區塊的狀態分享給 Child Vnode。Child Vnode 不但能夠完全地索引到 Parent Vnode 所索引到的資料區塊，而且 Child Vnode 其隨後所索引的資料區塊，SFS 會使用 Copy-On-Write 機制來維護 Child Vnode 與 Parent Vnode 之間資料的正確性以及完整性（圖 10）。

因為SFS管理的是Client的作業環境的儲存資

源的分配，所以，跟一般的檔案系統不同的是，SFS 並沒有目錄結構以及檔案結構可以供外界存取，而是直接將 Vnode 的操作介面開放出來。因為缺少了目錄結構以及檔案結構，詮釋資料以及 Vnode 的使用情形並不能夠被記錄下來（就像是一個檔案失去了在檔案系統上的紀錄一樣）。因此，SFS 提供了 Descriptor 結構，可以讓其他模組依自己的格式，來記錄詮釋資料以及 Vnode 的使用情形。

如圖 11 所表示，SFS 模組提供 Descriptor（大小為 2K）結構，供 Virtual Volume Pool 與 Client Manager 模組儲存管理作業環境相關的詮釋資料。SFS 利用一個 Vnode（就像是一般檔案系統中的根目錄）來分配與管理 Descriptor，並且按照 ID 順序來分配儲存空間給 Descriptor。而每一個 Descriptor 都有一個 TYPE 欄位，用來記錄該 Descriptor 是屬於哪一個模組所擁有的（Virtual Volume Pool 或者是 Client Manager）。

C. Virtual Volume Pool 模組

Virtual Volume Pool 主要負責管理所有 Client 的作業環境的儲存資源，在 Virtual Volume Pool 中的每一個 Virtual Volume，就是一個作業環境的儲存資源的抽象（Abstraction）。COEMS 利用這樣的虛擬化（以 Virtual Volume 代表作業環境），將多個 Client 的作業環境集中管理與維護，藉由對 Virtual Volume 實施管理操作，反映真實的 Client 的作業環境的情形，以達到管理之效果，使得多台電腦的作業環境的維護變得可行而且簡單。

如圖 12 所表示，對於每一個 Virtual Volume，Virtual Volume Pool 使用一個 Descriptor 來記錄 Virtual Volume 本身的資訊，並藉此資訊來跟其他模組關聯互動。當一個 Virtual Volume 被建立時，Virtual Volume Pool 會向 SFS 取得一個新的 Descriptor 來記錄 Virtual Volume 的相關資訊。Virtual Volume Pool 對於 Virtual Volume 的管理，除了提供基本的建置、刪除、設定、存取…等操作之外，還支援 Snapshot、Rollback、Image Sharing…等進階操作，才能夠達到管理及維護作業環境的效果，並且維護作業環境的日誌。而隨著 Virtual Volume 的使用，Virtual Volume 的資料狀態（使用 Vnode 來記錄）的日誌資訊也會被記錄在 Descriptor 中。

Virtual Volume Pool 利用 SFS 的 Copy-On-Write 機制，來處理 Virtual Volume 甚至 Virtual Volume 之間的容量管理。透過 Copy-On-Write 機制可輕易地將 Virtual Volume 於某個特定時間點時的資料狀態完整地保存下來，成為一個特定時間點的 Virtual Volume Snapshot。而經由 Snapshot 後 Virtual Volume 使用新的 Vnode 來為 Virtual Volume 提供索引資料區塊的服務（圖 13）。而隨後當有資料需要寫入 Virtual Volume 時，Copy-On-Write 機制就會視情況需要為新的 Vnode 複製和配置所需的資料區塊（圖 14），而不會影響到舊的 Vnode 的索引資訊。

Virtual Volume Snapshot 甚至可以被當成是一個 OE-Image（Operating Environment Image），分享給其他的 Virtual Volume 存取使用，而不會影響原始 Virtual Volume 的資料狀態。如圖 15 所表示，同質的 Client（指電腦硬體、作業系統以及應用軟體的基本需求相同）可以藉由 Image Sharing，來建置新的同質的 Virtual Volume，使得建置 Virtual Volume 變得快速而且容易。經由 Image Sharing 所建立的 Virtual Volume，可透過 Copy-On-Write 機制，來維護本身的資料狀態與 OE-Image 的資料狀態之間的差異（圖 16）。因此，就像是 Snapshot 操作一樣，Image Sharing 並不會影響原始的 Virtual Volume 的資料狀態的完整性。同時，節省了大量的存儲資源（不用浪費儲存空間來儲存許多相同的資料，只需要記錄資料間的差異性即可）。

D. Client Manager 模組

Client Manager 主要負責管理 COEMS 中的 Client，在 Client Manager 中的每一個 Client，就是一個外部的電腦主機的抽象（Abstraction）。COEMS 利用這樣的虛擬化，將多台電腦集中管理與維護。並藉由對 Client 實施管理操作，反映真實的電腦之

狀態，以達到管理之效果。

如圖 17 所表示，每一個 Client Manager 所管理的 Client，就代表一個外部的電腦主機。對於每一個 Client，Client Manager 使用一個 Descriptor 來記錄其電腦硬體、網路相關資訊…等資料。當一個 Client 被加入 COEMS 時，Client Manager 會向 SFS 取得一個新的 Descriptor 來記錄 Client 的相關資訊。而每一個 Virtual Volume 能夠依據 Descriptor 中的 Client 資訊，可以在 Client Manager 中找到擁有自己的 Client。就像如圖 17 中，Virtual Volume Descriptor 與 Client Descriptor 利用 IP 資訊來做關聯，而利用這樣的關連，VVM 能夠將多台電腦與多個作業環境虛擬化，並且集中管理與維護，使 COEMS 能夠在網路環境之中，同時維護與管理多台電腦的作業環境。

四、COEMS 原型之實作情形

我們選擇 Open Source 的 Linux 做為開發平台（RedHat 9，核心版本為 2.4.20），並以核心模組（Kernel Module）的方式，來開發 COEMS 原型。Open Source 的好處是使我們能夠完全地掌握 Linux 核心，並且進行 COEMS 原型的開發；而 Linux 支援動態載入核心模組的功能，更是方便我們開發像 VDD 這類低階的 OS 驅動程式。深入 Linux 核心不但方便我們開發低階的驅動程式，更使我們能夠獲得充分的系統資源，例如記憶體、區塊快速緩衝（Block Buffer）等。

而為求快速開發 COEMS 原型，以驗證 COEMS 之設計是否可行，COEMS 原型並沒有實作 NBP，而是直接在 Linux 上進行所有系統功能面的測試。沒有實作 NBP 的原因，是因為測試系統功能是否正確（指存取 Virtual Volume）並不需要利用到網路開機，而是可以利用簡單的設定以及初始化將執行環境直接設定為 Client 順利完成開機後的情形。所以，我們只需要測試 Client 是否能夠正確地利用 VDD 來存取 Virtual Volume。

如圖 18 所表示，COEMS 原型是完全依照 COEMS 之設計，來實作所有系統功能的（當然除了 NBP 的網路開機功能）。除了 NBP 元件之外，COEMS 原型實作了 4 個 COEMS 元件，分別是 NVS 元件、VDD 元件、CVM 元件以及 VVM 元件。

而經由功能面的測試，我們發現 COMES 的各個元件的確能夠相互合作而達到設計時所預期的功能。NVS 元件能夠於網路環境中，提供 COEMS 成員網路溝通的服務。VDD 元件能夠順利地 I/O 存取要求重導到 COEMS Server 進行處理，並且將 I/O 要求結果通知給檔案系統知道。系統管理者可以透過 CVM 元件經由網路來管理 COEMS 系統中所有的 COEMS Server 的運作，並且進行 Client 的作業環境的管理以及維護的工作。VVM 元件可透過虛擬化技術將 COEMS Server 的儲存空間虛擬化成 Virtual Volume Pool，並將 Client 的作業環境虛擬化成為 Virtual Volume 加以管理，並且能夠同時管理以及維護多台 Client 的作業環境的資料狀態。

五、結論與未來發展

透過測試 COEMS 原型，映證了 COEMS 所設計之功能以及架構是可行而且正確的。COEMS 藉由結合現有的作業環境維護技術、網路儲存技術以及裝置虛擬化技術等優點，將多台用戶電腦的作業環境集中於一或多台伺服器來進行管理與維護，成功克服傳統技術在維護作業環境的困難，使得在網路環境中，維護多台電腦的作業環境的需求變得可行。而利用 Volume Management 以及 Copy-On-Write 技術，COEMS 可同時維護多台電腦的作業環境日誌，並可提供資料分享，使得作業環境的維護比傳統技術更簡單有效率。

雖然 COEMS 原型能夠映證 COEMS 之設計是正確而且可行的，但是，實作完整的 COEMS 系統並且實際應用，仍是必須的。只有透過實際的系統應用，才能發現更多系統的需求以及缺點，進而改進系統的設計。現階段的 COEMS 之功能尚嫌陽

春，仍需要經過不斷地反覆測試、效能分析、以及增加更強大的功能，才能成就一個完美而周全的系統。COEMS 之未來發展，有下列幾個方向：

- COEMS 現階段並沒有使用任何的鑑別資訊來鑑別 Client，也沒有定義任何的 Client 的權限。為了系統以及資料安全，需要為 COEMS 設計一套完整的認證系統，來鑑別 Client 以及管理 Client 的行為。
- 由於COEMS Server集中管理多個Client的作業環境，容易發生因 COEMS Server 當機產生單點錯誤（Single Point of Failure）而影響 Client 對作業環境的存取。因此，為了增進資料安全，我們需要為 COEMS Server 端設計 Virtual Volume 的備份與復原的功能；或者是將多個 COEMS Server 叢集（Clustering）起來相互合作，以避免系統因為單點錯誤而停擺的危機。
- NVSP 現階段的設計過於陽春，沒有考量網路認證、資料加解密、以及品質服務保證（QoS）…等。隨著功能的擴充，我們需要為 COEMS 設計一個功能較為完善的 NVSP。

參考文獻

[1] 石旭本, “用戶端作業環境管理系統之設計”, 國立政治大學資訊科學系研究所, 碩士論文, 2005.

[2] BartPE, http://www.nu2.nu/pebuilder/

[3] Charlotte Brooks et al., "IBM TotalStorage: Introducing the SAN File System", IBM Redbooks, August 2004.

[4] Coleman, S. and R. W. Watson, "The Emerging Paradigm Shift in Storage System Architecture", Proceedings of the IEEE, April 1993.

[5] Coleman, S. S., R. W. Watson, R. A., Coyne, H. Hulen, "The Emerging Storage Management Paradigm", Proceedings of the Twelfth IEEE Symposium on Mass Storage Systems, Monterey, CA., April 1993.

[6] Chang-Soo Kim, Gyoung-Bae and Bum-Joo Shih. "Volume Management in SAN Environment", Proceedings of Eighth International Conference on Parallel and Distributed System, 2001.

[7] Diskless Remote Boot in Linux, http://drbl.nchc.org.tw/

[8] Intel Corporation, "Preboot Execution Environment (PXE) Specification, Version 2.1", ftp://download.intel.com/labs/manage/wfm/download/pxespec.pdf, September 20, 1999.

[9] Jon Tate et al., "Designing and Optimizing an IBM Storage Area Network", IBM Redbooks, August 2002.

[10] Kleiman, S., "Vnodes: An Architecture for Multi File Types in Sun Unix", Summer USENIX conference, 1986, pp.260-69.

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

[12] Knoppix, http://www.knopper.net/knoppix/

[13] Linux Terminal Server Project, http://www.ltsp.org

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

[15] Matthew T. O’Keefe, "Shared file systems and fibre channel", In The Sixth Goddard Conference on Mass Storage System and Technologies in cooperation with the Fifteen IEEE Symposium on Mass Storage Systems, College Park, Maryland, March 1998, pp.1-16.

[16] Norton Ghost, http://www.symantec.com/sabu/ghost/ghost_personal/index.html

[17] Steve Soltis et al., "The design and performance of a shared disk file system for IRIX", in the Sixth Goddard Conference on Mass Storage System and Technologies in cooperation with the Fifteen IEEE Symposium on Mass Storage Systems, College Park, Maryland, March 1998, pp.41-66.

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

[19] The Second Extended File system (EXT2), http://www.science.unitn.it/~fiorella/guidelinux/tlk/node95.html

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

[21] Venturcom BXP, http://www.directinsight.co.uk/products/venturcom/bxp.html

表 1：各種作業環境維護技術之比較。

	Norton Ghost	Live CD	Venturcom BXP	LTSP	DRBL
維護作業環境的方式	單一個別	單一個別	多台集中	多台集中	多台集中
維護個別作業環境的能力	優秀	優秀	優秀	優秀	優秀
維護個別作業環境的日誌的能力	不良	沒有	沒有	沒有	沒有
維護多個作業環境的能力	中等	中等	優秀	優秀	優秀
維護多個作業環境的日誌的能力	不良	沒有	沒有	沒有	沒有

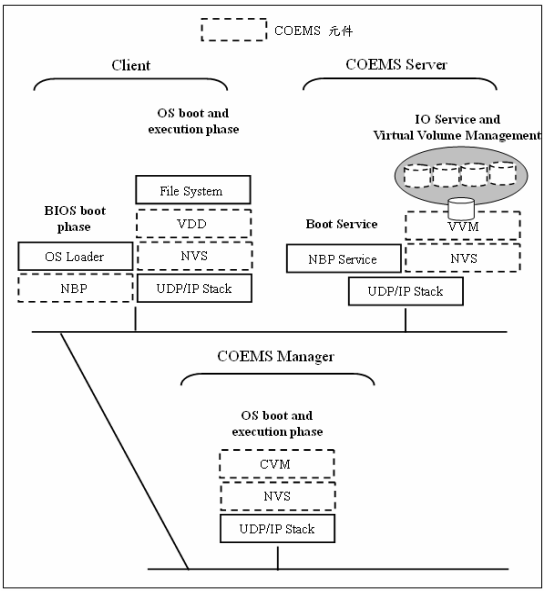

圖 1：COEMS 系統架構。

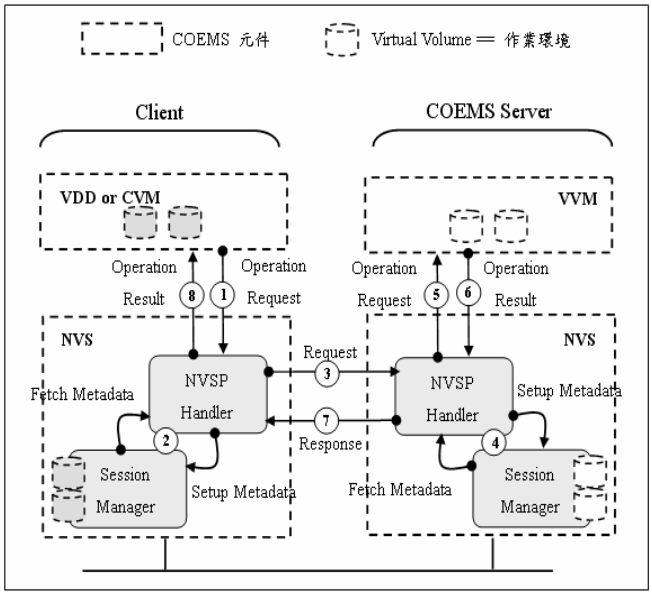

圖 2：NVS 處理操作要求的程序。

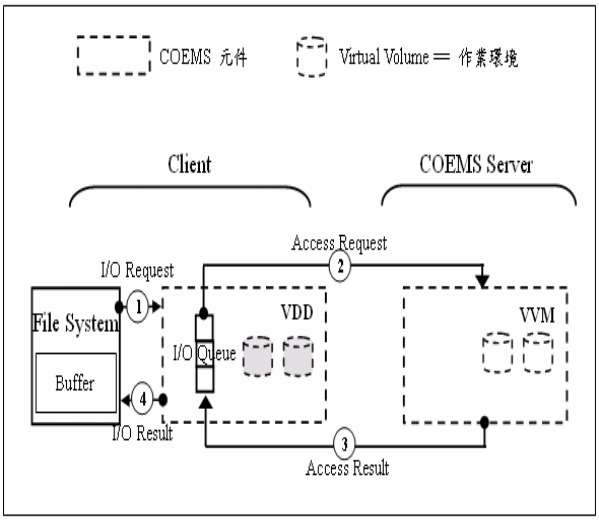

圖 3：VDD 處理 I/O 要求的程序。

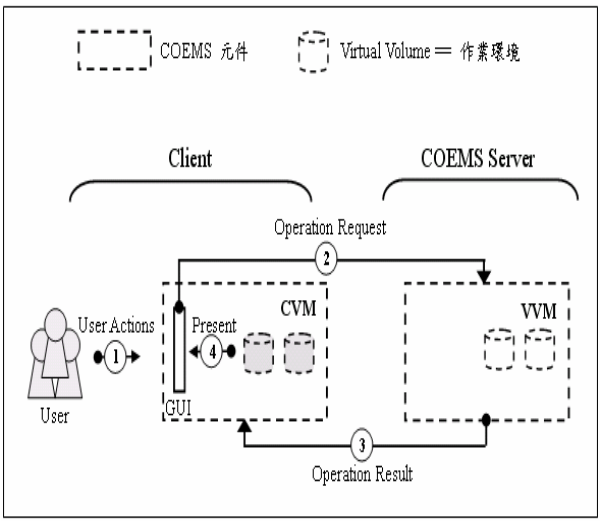

圖 4：CVM 處理使用者操作要求的程序。

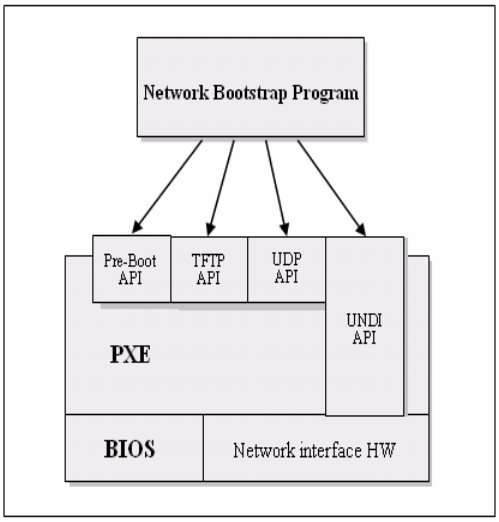

圖 5：PXE and NBP。

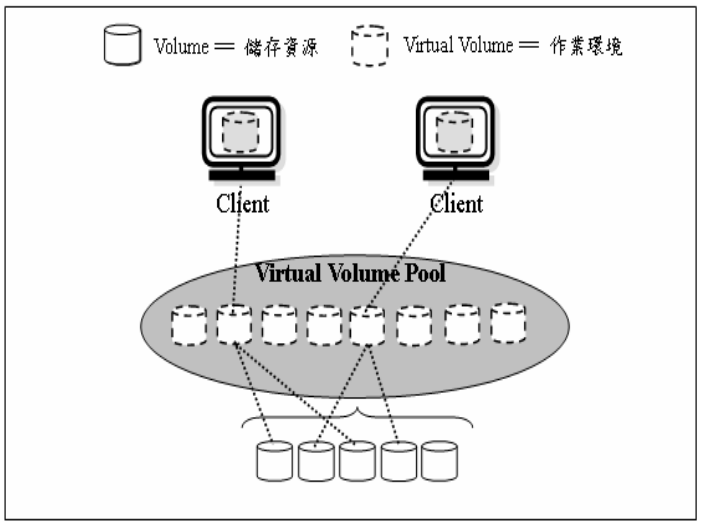

圖 6：VVM 的虛擬化技術。

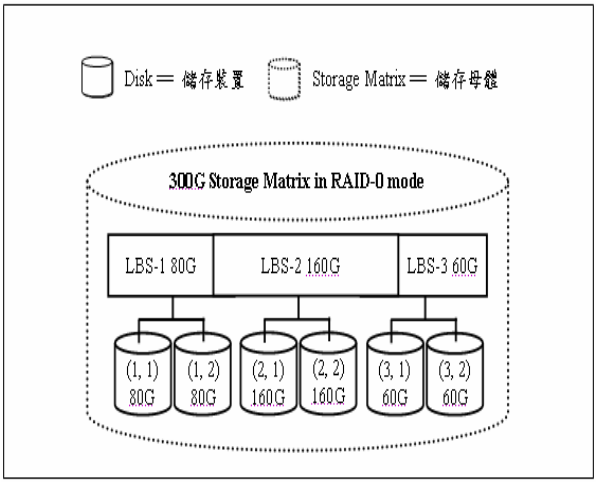

圖 7：Storage Matrix in RAID-0 mode。

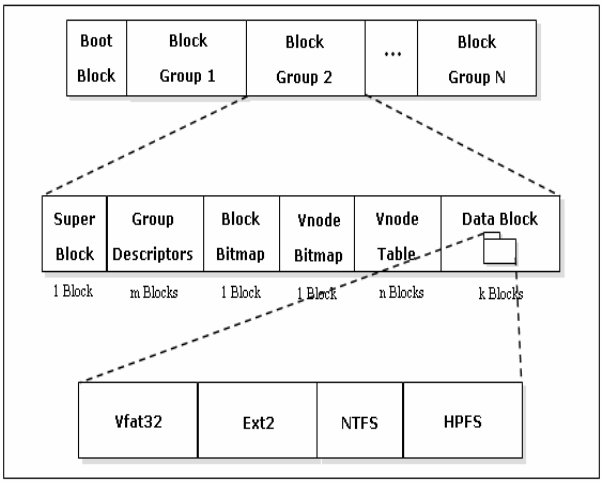

圖 8：SFS Layout。

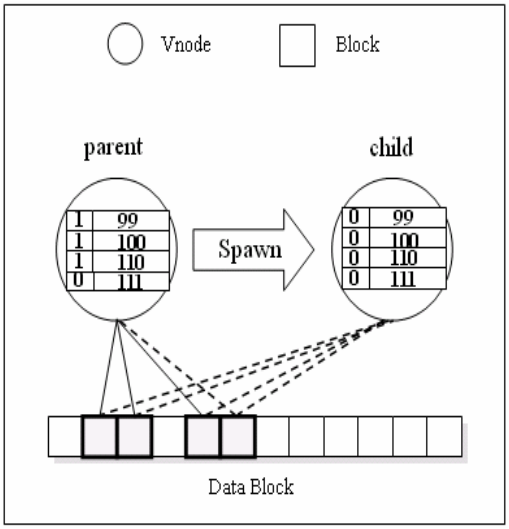

圖 9：Spawn Vnode。

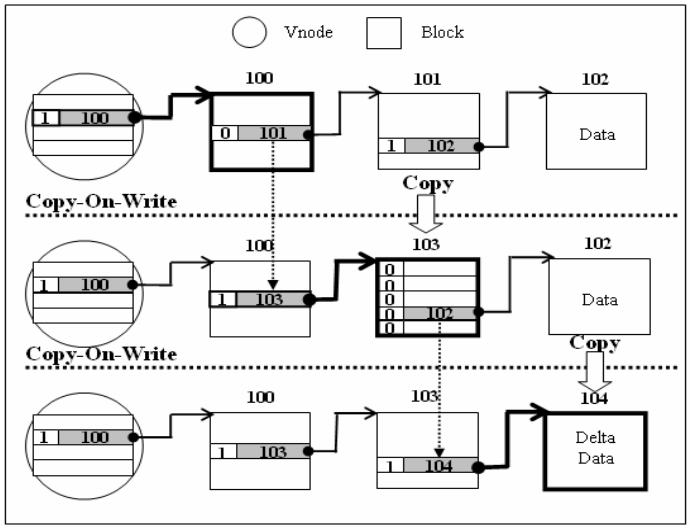

圖 10：Block Level Copy-On-Write。

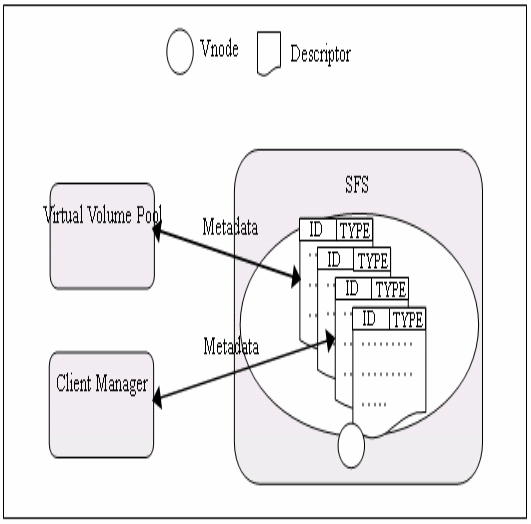

圖 11：Descriptor structure of SFS。

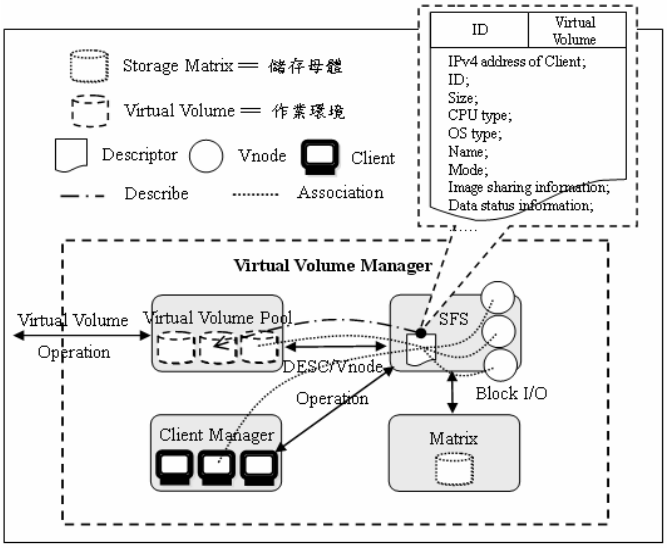

圖 12：Association of Virtual Volume。

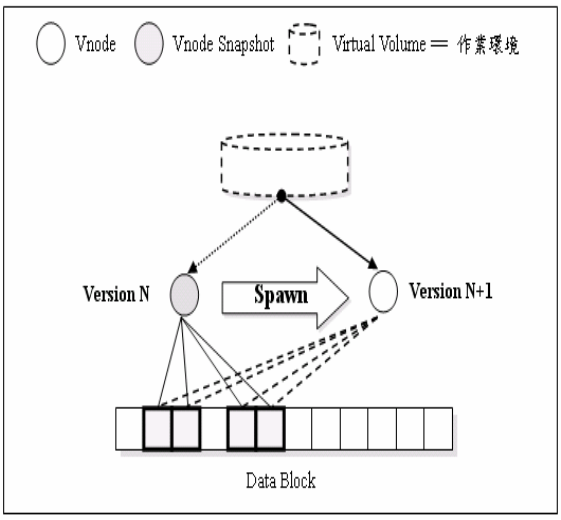

圖 13：Snapshot a Virtual Volume。

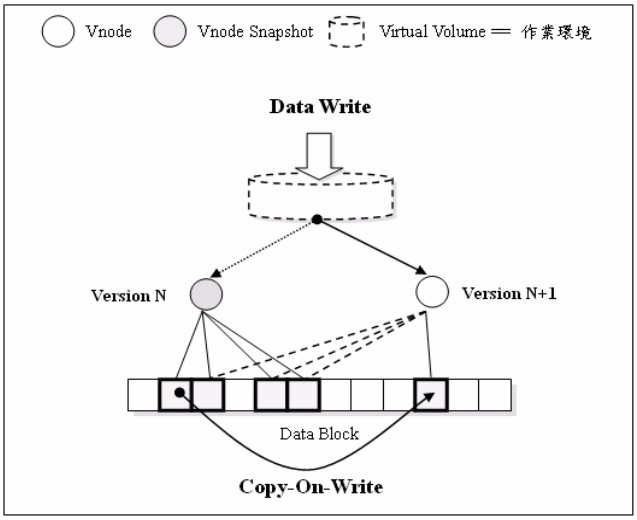

圖 14：Virtual Volume level Copy-On-Write。

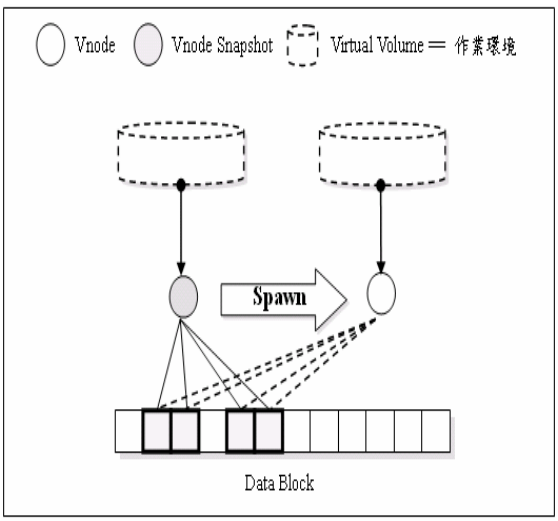

圖 15：Image Sharing。

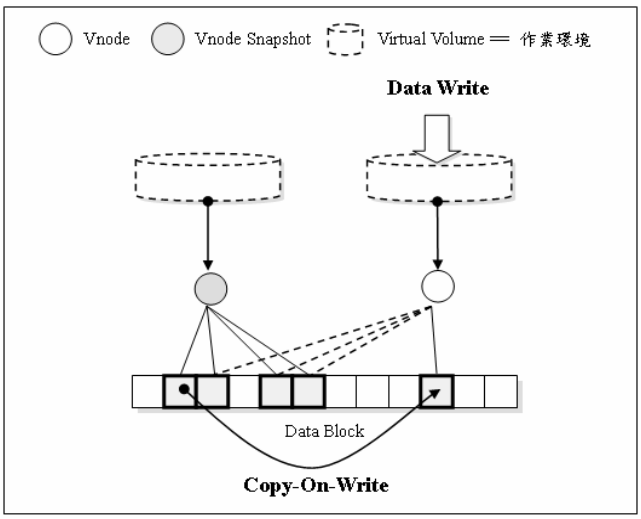

圖 16：Image Sharing with Copy-On-Write。

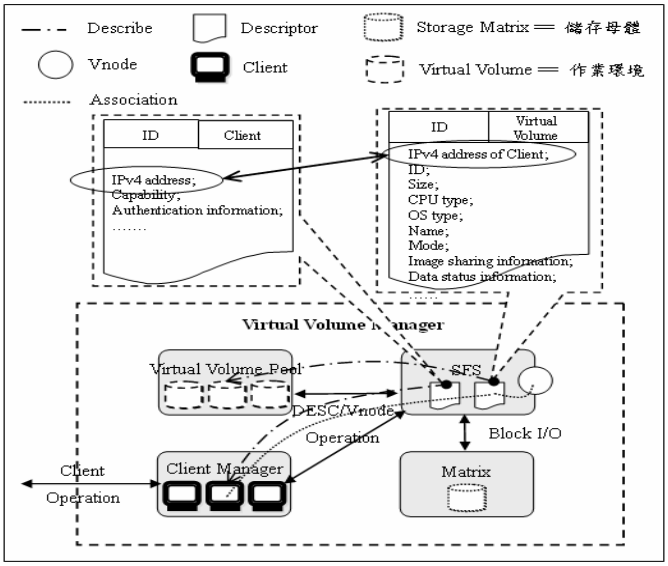

圖 17：Association of Client and Virtual Volume。

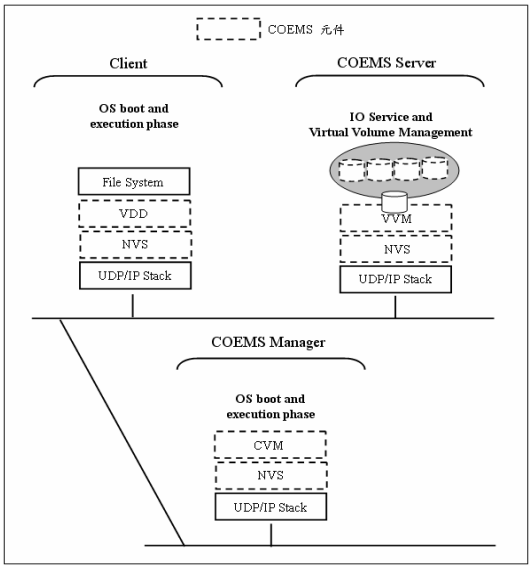

圖 18：COEMS 原型。