TFT-LCD缺陷檢測過程之超大型檔案儲藏庫資訊生命週期管理機制

楊明祥
高雄第一科技大學資訊管理學系

曾守正
高雄第一科技大學資訊管理學系

摘要

TFT-LCD已是台灣最重要的產業之一，由於產品應用範圍已擴大至電視，讓玻璃基板尺寸的需求愈來愈大，也使得玻璃基板或面板缺陷點數直線上升，導致存放這些缺陷點數影像檔的檔案伺服器(Server)容量及效能需求也已經膨脹到讓連當代最新的作業系統都無法負荷的情況。因此，如何重新設計整體伺服器的處理架構以滿足需求，已經是各家TFT-LCD製造公司都要面對的一大挑戰。

本研究採用兩個方法來改善伺服器的效能問題，第一是導入資訊生命週期管理(Information Lifecycle Management, ILM)架構，使用資料庫即時紀錄檔案路徑，供程式刪除過期檔案的搜尋使用，以解決搜尋巨量檔案對伺服器效能所產生的影響。實際驗證的結果顯示：資訊生命週期管理系統架構可以有效改善程式每天清除過期檔案時，所產生的效能嚴重低落問題，而且可以縮短程式執行的時間。實驗結果顯示，當檔案數量很多時，至少可以節省1/3的程式執行時間。另一個方法是參考延伸雜湊(Extendible Hashing)的方法，發展出一種分散式檔案系統，讓所有檔案平均分佈在這些伺服器裡，解決伺服器在空間管理上所衍生的效能低落問題。並以某TFT-LCD製造廠的5代及7.5代廠當作例子，列出兩種不同世代廠的伺服器磁碟空間使用量與建置成本預估。從兩種建置方案的成本預估比較可以發現：使用分散式檔案系統約可節省45%～55%的建置成本，符合TFT-LCD產業降低成本的趨勢，可協助大幅提昇台灣廠商在該產業上的國際競爭力。

關鍵字：延伸雜湊、分散式檔案系統、伺服器效能、資訊生命週期管理

本研究承蒙國科會計畫部份贊助，計畫編號：NSC 96-2416-H-327-008-MY2，特此誌謝。

Information Lifecycle Management for Very Large File Repositories in TFT-LCD Defect Inspection

Ming-Hsiang Yang
Department of Information Management,
National Kaohsiung First University of Science and Technology

Frank S.C. Tseng
Department of Information Management,
National Kaohsiung First University of Science and Technology

Abstract

TFT-LCD manufacturing is a very important industry in Taiwan. Because TFT-LCD panels have been applied to TV products, the glass size of TFT-LCD grows up tremendously. This causes the number of defects in glasses also increase rapidly, which in turn makes the disk space of file servers have to be expanded accordingly. Due to such tough situation, the performance and disk space in file servers may reach the limitation of the underlying operating system. Therefore, how to reconfigure the file system architecture to alleviate such phenomenon has becoming a big challenge for most of the TFT-LCD manufacturers.

In this paper, we will propose two approaches to respectively mitigate the maintenance operation and improve the file server performance. The former approach uses information Lifecycle Management (ILM) architecture to store all file information into a database. Because file searching operations are transformed into database operations, this makes the deletion of expired files be a lightly impact to file servers. The other one develops a distributed file system based on extendible hashing scheme to evenly distribute files to file servers. We have conducted experiments respectively based on examples applied to the processes of the 5th and 7.5th generation TFT-LCD manufacturing factories of a manufacturer in Taiwan to assess the diskspace and investment cost. Based on our study, we found our approach can substantially save approximately 45%~55% establishment cost. We believe that will promote the core competence of the Taiwan's TFT-LCD industry.

Key words : Extendible Hashing, Distributed File System, File Server Performance, ILM (Information Lifecycle Management)

壹、導論

一、研究背景

TFT-LCD乃Thin-Film Transistor Liquid-Crystal Display的縮寫，中文稱為「薄膜電晶體液晶顯示器」，於1960年由美國RCA公司發明。1970年代日本夏普 (Sharp) 將技術引進日本，1973年第一台液晶顯示器的相關產品上市，接著其他廠商紛紛跟進，從此日本掌握了液晶顯示器的市場。1990年代中期南韓成功切入，1990年代末期，台灣結合了日本TFT-LCD廠商的授權及台灣擅長的低成本量產能力，再加上政府「兩兆雙星」政策的推動，短短幾年即追上日、韓，使得TFT-LCD成為台灣繼半導體之後最重要的產業之一，也是第二個年產值突破新台幣一兆元的產業。

TFT-LCD的前段製程如圖1所示，依各段的特性可分為三個部份：

1. TFT(Array)製程：TFT的製程大致上和半導體製程類似，每道製程可分為薄膜、黃光及蝕刻三個模組，其間可能重複數次，它與半導體製程的差別是TFT是將電晶體鍍在玻璃基板上，而半導體則是鍍在矽晶圓上。一般來說，完整的薄膜電晶體至少需要五道製程，其中每次在薄膜、黃光及蝕刻各模組完成一次製程或完成所有的TFT製程之後，還需要對玻璃基板進行量測或檢測的動作，以確認製程品質。在最後檢測後，還需要根據檢測結果，判斷是否要對有問題的玻璃基板進行修補或報廢。

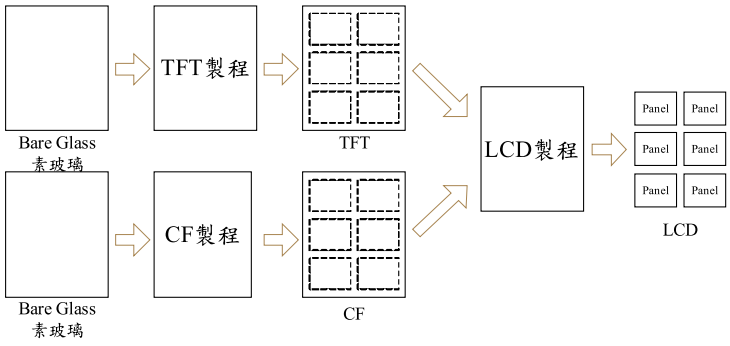

圖1：TFT、CF及LCD製造流程

2. Color Filter(CF)製程：CF的製程大致上也和TFT製程類似，分三次製程，依序將彩色光阻(R, G, B)鍍在玻璃基板上。CF製程大致上分為玻璃濺鍍（Cr濺鍍）、黑色矩陣（Black Matrix，BM製程）、RGB三色製程，及ITO濺鍍等步驟，其中每次在各色完成一次製程或完成所有的CF製程後，也需要對玻璃進行量測或檢測的動作，以確認製程品質。在最後檢測後也需要根據檢測結果，來判斷是否要對有問題的玻璃進行修補或報廢。

3. LCD (Cell)製程：LCD的製程是以前段TFT(Array)為基板，將液晶滴入，再與彩色濾光片結合。LCD製造流程如圖1所示，一開始先進行TFT及CF基板洗淨(Cleaning)、配向膜印刷(PI Printing)、配向膜配向(PI Rubbing)、密封膠塗佈(Sealant Dispense)、最後在TFT注入液晶(One Drop Filling，ODF)、和CF進行組合裝置(Cell Assembly)、切割成最終尺寸(Scriber/Breaker)、偏光膜貼附(Polarizer Attachment)，及最後的點燈測試(Cell Test 1 & 2)。所有面板的品質好壞和等級都會在點燈測試這個程序決定，測試後並對有問題的面板進行修補。

由上述TFT、CF及LCD的製造流程可以發現：其中存在有許多檢測流程，如：TFT製程中的量測及最後檢測，CF製程中的自動檢查機檢查及最終檢查，LCD製程後面的面板測試1及2(Cell Test 1 & 2)，這些測試流程主要的目的是為前面製程的品質把關，及時找出有問題的「在製品」(WIP, Work in Process)，並採取適當的因應措施，如修補或報廢。

如果沒有在發現產品有問題時及時採取適當的措施，會使有問題的產品往下面的製程走，結果將導致品質成本的增加或是良率的降低。所謂品質成本是指因為品質問題付出的成本，如果有一片面板在TFT最前面的製程就產生了問題，到了LCD面板測試才發現這個問題而需要報廢，則這片面板所有使用到的材料，如：CF、偏光板及液晶等，也因而浪費掉了。如果這個問題是可以被接受的，也可能必須以較低的面板等級 (Panel Grade) 及較差的價格出貨。這些結果都會導致面板平均成本的增加，降低公司產品的競爭力。

因此，多數TFT-LCD廠都會透過層層的缺陷檢測(Defect Inspection)來及時找出有問題的面板，其方式大都透過檢查機使用自動掃瞄並照相比對的方法，來找出這些有問題的面板。因此這些缺陷檢測流程便會產生大量的缺陷資訊(Defect File)及缺陷影像檔(Image File)。由於這些檔案必須存在一段時間，直到這些面板出售之後的保固期間為止，而且大部份檔案必須集中存放，以供其他機台使用或是供資料分析之用。故目前最簡單的解決方案就是採用許多檔案伺服器(File Server)，以存放工廠量測及檢測機台的缺陷資訊及缺陷影像。

每一個缺陷資訊會存放機台測試或判斷面板等級會用到的所有資料，格式是文字檔，該類型檔案會針對每一片玻璃產生一個缺陷資訊檔案。而缺陷影像檔案，即缺陷點的照片，主要格式為jpg及bmp，該類型檔案一個缺陷點至少一張，所以機台產生檔案的數量會和玻璃基板尺寸及良率成正比。

由表1的資料可以看出當市場需求面板尺寸愈來愈大時，如果玻璃基板尺寸愈大則產能愈大，假設每平方單位的缺陷點數量是固定的，於是玻璃基板尺寸愈大的結果是整片玻璃基板缺陷點的總數量增加。再加上面板廠這幾年為了降低人力作業的需求，極力導入自動檢查、等級判斷及修補等機制，使得機台照像的動作由抽檢、改為全檢，再利用程式取代人員做判斷，更是使照片數量以相當可怕的倍數大量成長，讓以往可能10片玻璃才照相抽檢一片的情況，現在變成10片都需要照相檢查。

表1：各世代廠玻璃基板尺寸及不同面板尺寸切割數量

廠房	玻璃基板尺寸		32吋	37吋	40吋	42吋	47吋
G5	1100	1300	3	2	2	2	2
G5.5	1300	1500	6	3	2	2	2
G6	1500	1850	8	6	4	3	2
G7	1870	2200	12	8	8	6	4
G7.5	1950	2250	12	8	8	8	6

資料來源：http://www.auo.com.tw

隨著LCD面板尺寸愈來愈大，檔案數量也呈倍數成長，因此檔案伺服器將漸漸產生以下問題：

1. 資料儲存設備(Storage)投資成本呈倍數增加：從早期四代廠數百Giga Bytes，暴增到五代廠以後數十Tera Bytes。
2. 單一硬碟機空間需求超過資料儲存設備的能力：不管是DAS(Direct Attach Storage)或是SAN(Storage Area Network)，RAID都會有硬碟機數量的限制，以IBM的中階儲存設備產品為例，一個RAID最多只能有30幾個硬碟機，如果使用單個硬碟機容量為146GB的RAID，最大容量只能達到4.3TB。
3. 檔案數量太多，效能要求超過作業系統能力，因而影響產能：根據Microsoft的建議，檔案伺服器處理小檔案時的效能較差，而且即使CPU增加也無法改善。
4. 檔案系統(File System)大小超過作業系統限制：不管是使用一般的Windows、Linux或是直接使用NAS(Network Attached Storage)都會有單一磁碟空間大小的限制。在Microsoft 2003 Server Cluster的環境下，每個分割區(Partition)大小上限為2 Tera Bytes。而NAS則是根據內部使用何種作業系統而定。
5. 刪除過期檔案的機制會影響效能：由於檔案系統效能不佳，刪除過期檔案將會耗費相當多的時間，而且在執行這些動作時還會影響效能，導致工廠生產效率也受到影響。例如：若有一個檢測機台正常時候顯示一個影像檔需要500mSec，且每天可檢測2000片面板，則當檔案伺服器效能變慢，在顯示一個影像檔需要1000 mSec時，則該機台產能會減半，變成每天僅能生產1000片面板，影響產能甚鉅。

二、研究動機

TFT-LCD檢測流程機台產生數以千萬計的大量檔案數量，根據我們檢視國內知名的C公司製造廠所得到的結論是：目前並沒有很好的方法，可把每天超過保留期限的檔案找出來，並備存(Archive)到備份伺服器(Backup Server)；也沒有解決方案可以對這些檔案伺服器執行漸增式備份(Incremental Backup)，因為搜尋出這些檔案的速度，比檔案進來的速度還慢，而且搜尋的過程更會導致DISK I/O效率快速下降。其結果便是：

1. 如果這些檔案保留期限縮得太短，當後端機台需要用到前端機台產生的檔案，但卻找不到時，前端機台只好重新拍照產生檔案，如此將會延誤該批面板的生產時程，且佔用機台的產能，導致產能下降。

2. 由於需要定期清除過期的檔案，但使用傳統搜尋檔案建立日期的方法在檔案數量太大的伺服器上會耗費非常多的時間，甚至比檔案進來的速度還慢，嚴重影響伺服器效能。
3. 檔案數量需求量暴增的結果是投資成本直線上升，因為傳統的Windows無法解決作業系統的限制，所以可能需要改採用非Windows的作業，例如：IBM AIX，但是如此投資成本將會更高。在面板產業獲利日益困難的今天，如何降低成本是在評估系統架構時的一個重要考量因素。

三、研究目的

本研究的目的為找出方案，解決上述問題，研究的主題主要有以下三個方向：

1. 探討TFT-LCD檢測資訊生命週期管理(Information Lifecycle Management, ILM)的使用，我們規劃先將所有檔案位置記在資料庫中，把搜尋檔案的需求轉成透過資料庫的查詢找到檔案的明確位置，再直接去該位置操作該檔案，不用經過檔案系統搜尋，也就是利用資料庫當作整體檔案的索引，再配合資料庫表格索引對效能的協助，提昇查詢效能，降低搜尋過期檔案時對效能所造成的影響。同時依據檔案的利用率高低，將檔案分開存放在不同的貯存空間。
2. 探討使用分散式檔案系統架構(Distributed File System)存放檔案，以改善作業系統分割區(Partition)大小的限制，並採用成本較低的小型伺服器，滿足使用者容量及效能的需求。
3. 可行性驗證及研究整體架構的效能評估。

本論文的架構簡介如下：第二節為相關文獻探討。第三節說明如何利用資料庫型態的檔案系統來實現本論文所探討的資訊生命週期管理架構，以及分散式檔案系統架構。在第四節中，我們利用實際在C公司生產環境所收集到的資料，存到資料庫，再將這些資料進行分析，並依據這些結果來評估、驗證本論文所提出之方法的可行性。最後在第五節提出總結，並討論未來可能的研究方向。

貳、文獻探討

一、資訊生命週期管理系統

全世界電子資料每年以超過30% 的速度成長，龐大的資料管理工作對企業造成全新的挑戰，再加上最近幾年來審計資料安全管控及公司行為監管上的需求，如：沙賓法案(Sarbanes-Oxley)、及遵守健康保險可攜性與責任法案(HIPAA)等，更增加了資訊管理的複雜性(Chen, 2005)。

為因應如此大量的資料需求及企業對於成本、效能、可靠度，及可用度的要求（黃華，楊德志，張建剛，2004），儲存媒體的廠商便提出了「資訊生命週期管理」的概

念(Information Lifecycle Management, ILM)。根據儲存網路產業協會(Storage Network Industry Association, SNIA)的資料管理論壇(Data Management Forum, DMF)所定義：「資訊生命週期管理」就是一種基於符合整體企業目標與成本效益的規範下，所制定的資訊基礎架構管理規範。

Peterson(2004)以圖2說明了不同生命週期的資訊價值也不同，如此可以改善系統資源使用率及自動提高資訊的價值，使系統了解什麼資料在什麼時候是重要的，所以正確的政策可以套用在正確的時間，根據資料價值決定資料的片斷要放在那裡。如果缺乏資訊的估價方式，將使資訊生命週期管理只是一個理想化的想法，而不實際(Reiner *et al*., 2004)。

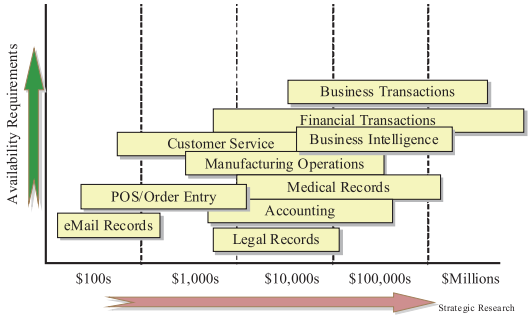

圖2：資訊的價值 (Peterson, 2004)

本研究參考資訊生命週期管理的理論基礎，將檔案伺服器的檔案依使用者需求分類，讓線上伺服器(Online File Server)只存放價值較高的資料，其他的搬移到離線伺服器(Offline File Server)或磁帶中儲存。

二、分散式檔案系統

TFT-LCD檢測流程機台存放資料的檔案伺服器中，最重要的角色就是資料儲存設備，然而一般傳統式的資料儲存設備存在著成本昂貴、單點失效(Single Point of Failure, 簡稱SPOF)、容量和效能等問題(Jones *et al*., 2000)。針對這些問題，目前已有許多不同的分散式儲存系統(Distributed Storage System)技術陸續被提出(Verma *et al*., 2005; Ghemawat *et al*., 2003)。

分散式檔案系統一般分為採用儲存區塊等級(Block-Level)及檔案系統等級(File-System-Level)兩種(Lee, 1996)。儲存區塊等級(Block-Level)操作的對象是不同的單位資料區塊，各區塊之間並沒有任何資訊關係，因此維護詮釋資料(Metadata)將會導致記憶體負荷較大，而且區塊之間存取的操作較不準確，限制了最佳化(Optimizations)的可能性(Flouris, 2005)。

儲存區塊等級的儲存系統(Block-Level Storage Systems)包括RAID-II、TickerTAIP、Logical Disk、Loge、Mime、Petal、AutoRAID及Swift，而檔案系統等級 (File-System-Level) 的則有*x*FS、Zebra、Echo及AFS等幾種 (Anderson *et al*., 1996)。以下分別就Swift及

Petal兩種分散式檔案系統加以探討。

（一）Swift分散式檔案系統架構

Long(1994)提出一種稱為Swift的分散式檔案系統架構，其系統元件可分為分散代理程式(Distribution Agent)，資料儲存設備中介模組(Storage Mediator)，及資料儲存設備代理程式(Storage Agents)。整體系統架構如圖3所示。

本研究所提出的分散式檔案系統架構參考Swift系統元件的分散代理程式(Distribution Agent)，將檔案分派至不同的伺服器，以分散各伺服器的負載。

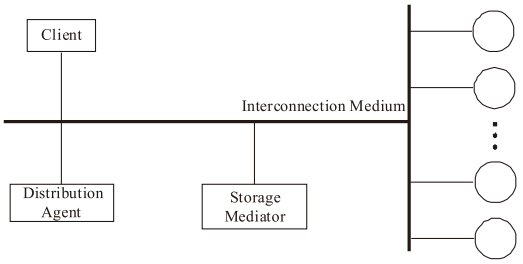

圖3：Swift架構的組成元件 (Long, 1994)

（二）Petal分散式檔案系統架構

Lee (1996) 另外提出了一種稱為Petal的分散檔案系統，其實體架構如圖4所示：

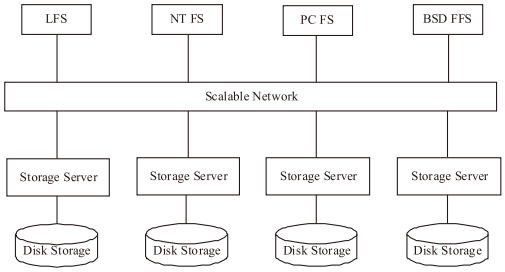

圖4：Petal的實體架構 (Lee, 1996)

Petal的資料存取方式採用串聯非叢集資料置換方案(Chained-Declustering Data Placement Scheme)，每一份資料區塊(Block)都會有二份複本，分成主要(Primary)和次要(Secondary)兩份。當系統收到讀取資料的需求時，會從主要或次要複本讀出資料，如果成功則回傳所需資料，如果失敗則回傳錯誤碼，然後重試別台伺服器，但是在寫資料時要優先考慮寫入主要複本(Primary Copy)，此時會先將該資料元件(Data Element)標示為忙碌(Busy)，然後再把寫入需求(Write Request)送給自己和次要複本(Secondary Copy)，如果都成功，則清除忙碌紀錄(Busy Bit)，並回報結果是成功還是失敗。如果在寫入主要複本的過程發生問題，則可以利用忙碌紀錄來回復系統，以解決一致性的問題，然後將資料寫到次要複本，並將資料元件(Data Element)標示為損壞狀態(Stale)，待主要複本回復後，再將這些資料寫入主要複本，如此可以解決死結(Deadlock)的問題。

Petal是一種易於管理的分散式儲存系統(Distributed Storage System)，提供了虛擬磁碟機(Virtual Disk)給使用者，也就是提供可統一集中存取的儲存空間。跟一般磁碟機相同，Petal提供了以區塊(Block)為讀寫單位的存取方式，但不同的是：Petal可依據需求提供更大的實體空間，而且其選擇性的複製功能也滿足了高可用性(High Availability)的需求。

三、雜湊方案探討

雜湊法(Hashing)是資料結構中常見的資料搜尋方法，但是也存在著資料溢出(Overflow)及擴充籃子(Bucket)的問題。尤其當資料量增加或減少時缺乏自動調整的彈性，導致存取效能常會因為資料量的增加而變差。有鑒於此，Fagin et al. (1979)提出了延伸雜湊(Extendible Hashing)的概念：透過兩層式的目錄結構將資料分類，第一層為目錄頁面(Directory Page)，主要是取出虛擬關鍵值(Pseudo Key)的前幾碼構成目錄，作為資料的第一次分類依據，並用以存放資料位置等資訊；第二層為分葉頁面(Leaf Page)，為資料實際存放的地方，如圖5所示。此方法在分類前會先採用一般的雜湊函數產生出虛擬關鍵值(Pseudo Key)，它和傳統雜湊不同的是該虛擬關鍵值為唯一、長度固定，而且僅使用關鍵值的部份資料，例如：關鍵值若為0100100，虛擬關鍵值如果取前4碼就是0100。

延伸雜湊在每個頁面都有保留一個資訊，稱為深度(Depth)，例如：目錄深度(Directory Depth)為2表示有$2^d=2^2=4$個指標指向分葉頁面，如果目錄頁面愈多，深度就需要愈大。分葉頁面也有深度，稱為本身深度(Local Depth)，存放分葉頁面的深度，例如：深度若為2，則表示這些頁面的前面2碼都是一樣的。

延伸雜湊在插入資料時，如果分葉(Leaf)滿了，就把原來的分葉頁面分開成二個分葉，如果分葉不夠用，或是本身深度等於目錄深度，就會把目錄的數量變成原來的二倍，再把分葉分開。將目錄變成二倍所需的成本並不高，因為大部份的分葉頁面不會被更動到。

在搜尋的時候，只要將關鍵值經過相同的雜湊計算公式求出虛擬關鍵值，再根據深度，取前面幾碼，然後根據其值及指標，就可以找到資料所在的分葉頁面。使用延伸雜湊(Extendible Hashing)的分頁失敗(Page Fault)次數不會多於2次，也就是說：最多只要2次讀寫動作就確保可以找到資料。

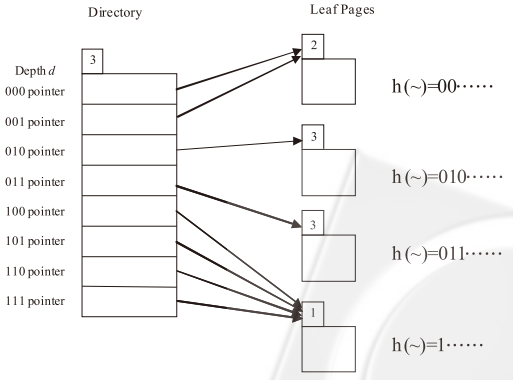

圖5：Extendible hashing的概念說明 (Fagin et al., 1979)

另外有一種由Enbody & Du(1988)根據動態雜湊所提出的動態雜湊方案(Dynamic Hashing Schemes)，目的是為了解決空間使用率及平順擴充等問題，包括目錄方式(Directory schemes)及無目錄方式(Directoryless Scheme)。其中目錄方式即為延伸雜湊，而無目錄方式，則是假設檔案是存放在連續空間，如果需要擴充時，則不採用目錄方式所使用的分裂處理，而是以依序再加入一個頁面的方式來處理。

四、檔案伺服器的限制

檔案系統所能存放與即時處理的檔案數量通常會有一定的限制，例如：表2中顯示了Windows 2003 Server的NTFS檔案系統所能承受的限制。雖然看似可以滿足大多數的需求，但是EMC(http://www.emc.com)曾在一份關於DiskXtender for Windows軟體的最佳範例中(EMC Corp, 2006)，建議在一個磁碟機上最好不要超過2千萬個檔案，否則可能會讓NTFS的主要檔案配置表(MFT, Master File Table)產生不穩定的狀況，因此我們將以這個數量為檔案伺服器檔案數量的參考上限值。

表2：Windows 2003 Server檔案系統限制

描述	限制
Basic volume的最大size	2TB
NTFS volume最大size	2^{32} －1個 clusters 最大size為256 TByte－64 KB 預設為16 TB－4 KB
一個NTFS volume最大檔案數	4,294,967,295 (2^{32}－1 file)

資料來源：微軟官方網站

參、研究方法與系統架構

本研究的結果之一為參考資訊生命週期管理系統(Information Lifecycle Management, ILM)的概念，依據資料使用者的需求將檔案分成：線上檔案伺服器(Online File Server) 及離線檔案伺服器(Offline File Server)兩類，並透過自行開發的程式監控檔案伺服器。當有檔案被建立或修改時，能將檔案的資訊即時寫到資料庫，並利用這個資料庫的內容去查詢檔案的日期並刪除過期的檔案，使檔案伺服器的檔案數量能維持一定的數量。

本研究設計一個分散式檔案系統架構，由於機台寫入檔案的數量十分龐大且速度很快，尖峰時刻最高可達每秒40個檔案。如果檔案分配不平均，將會使得檔案的存取集中在少數幾台伺服器，導致這些伺服器負載過重而降低效能，而同一時間內，其它的伺服器卻有閒置的資源沒被使用，造成伺服器資源使用不平均，無法將整體效能完全發揮。

在過去的參考文獻中，我們發現並沒有相關的方法，可以將機台快速寫入檔案的工作，平均分派到各檔案伺服器。本研究採用雜湊函數(Hashing Function)的方式來計算檔案所要存取的位置，透過特定方法設計出來的函數，直接從提供的關鍵值(Key)上，計算

出所要存取資料的位置，以增快搜尋的效率，降低時間成本。

第二部份是參考前面提到的幾種分散式檔案系統架構及延伸雜湊的方法，發展出一種容易實作的分散式檔案系統，並讓它具有分散負載，以及容易擴充的優點。檔案分散的方法主要是利用修改延伸雜湊(Extendible Hashing)的方法，直接利用雜湊函數將要寫入的檔案分散到各個實際存放的伺服器中。

一、資訊生命週期管理系統架構

為了解決檔案伺服器檔案數量過多導致的效能下降，以及災難回復不易等問題，本研究導入資訊生命週期管理系統(Information Lifecycle Management System)的概念，依據各檔案對整體工廠的價值會隨其產生的時間而漸次遞減之效應，將利用價值較高的檔案放在線上檔案伺服器(Online File Server)，而超過一段時間的檔案，因為其利用價值較低，於是搬到離線檔案伺服器(Offline File Server)中。

如果單純使用傳統的應用程式去掃瞄、比對所有檔案建立或修改時間的屬性來決定是否要搬移檔案，則執行時將會對將伺服器效能造成很大的衝擊，導致執行時間拖長，甚至比檔案寫進來的速度還慢。因此，為了減少檔案搬移過程對系統效能的影響，我們使用資料庫來記錄檔案的資訊，機台產生的檔案會先放在線上檔案伺服器，同時監控檔案程式Watcher會監控線上檔案伺服器，將每個新增或修改檔案的建立日期、檔案名稱、路徑，附檔名，及所在伺服器位置等檔案屬性即時存到訊息佇列（Message Queue，簡稱MQ），再透過事件監控程式將佇列的資料取出，記錄到資料庫的表格。

當檔案產生後，搬移檔案的程式Remover每隔一段時間會依據管理者設定的檔案保留規則，及從所對應資料庫表格的t_stamp欄位查詢出該檔案建立日期，如果符合刪除條件，則參考filename及path欄位，組合成檔案名稱及路徑，再將這些過期的檔案搬到離線檔案伺服器上。

當檔案產生的時間超過使用者要求的保留期限時，搬移檔案的程式Remover每隔一段時間會再依據規則，將這些欲刪除的檔案先備存(Archive)到備份伺服器，然後再刪除，刪除的同時也將資料庫的該筆記錄一併刪除。圖6及圖7說明了整個資訊生命週期管理系統的運作流程及架構。

圖6：ILM運作流程

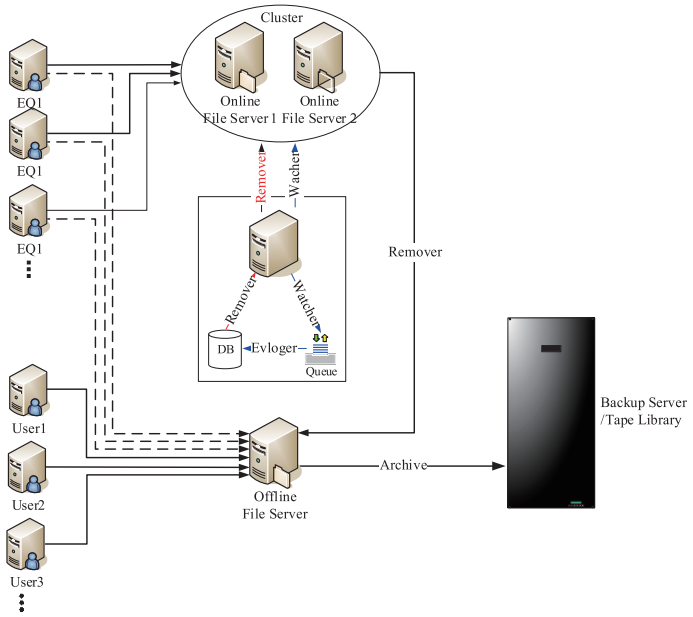

圖7：ILM系統架構

（一）系統架構

在圖7中，Watcher程式對於檔案活動的即時性及效率監控結果，乃是決定本研究架構是否可行的主要關鍵。為探討本研究架構的可行性，我們測試過四種資料處理的方法：第一種方法直接將資料寫入資料庫且每筆交易都交付(Commit)。第二種方法只計算所監控的檔案數目，不做任何處理，目的是為了驗證程式是可以完全監控到所有被送進伺服器的檔案。第三種方法則是每2000筆交付一次。然而對於某些檔案進來數量很多的伺服器而言，若在實際案例中，同時執行多支程式去監控多個目錄，且交付筆數設定太少，則可能會導致檔案的資料漏掉。將交付筆數設為2000筆，雖然不會漏掉資料，但是這樣會變成每次程式都要等新增的檔案累計達到2000筆時才會交付交易，因此會導致表格鎖定(Table Lock)及交易日誌(Transaction Log)暴增的問題經常發生。其中，表格被鎖定的問題會導致其他程式因為無法插入資料到這個表格而需要等待一段時間，如果等待時間過久，則最後會因為超過資料庫設定的最長表格鎖定時間(Lock Timeout)而被強迫結束交易。若是交易日誌暴增，則有可能導致所有資料庫交易失敗，甚至是整體資料庫運作異常。

為了有效同時解決表格效能及鎖定(Lock)的問題，本研究詳細分析後，認為唯有採用具有非同步訊息傳遞特性的中介軟體架構可以有效的解決這些問題(Davies and Broadhurst, 2005)。基於軟體取得便利性及具相關程式開發經驗，本研究的第四種方法最後決定採用IBM MQ Server的訊息佇列(Message Queue)來解決上述問題。實際測試的結果也顯示：佇列效能比資料庫表格一筆一筆交付要來得好，而且沒有表格鎖定的問題。

圖8為本研究程式的架構，我們依據程式的功能分為三個部份說明：

1. Watcher：即時監控檔案伺服器檔案異動，並將異動資訊轉換為訊息，放進佇列。

為了增進Watcher程式效能及減少漏掉的檔案，我們只監控檔案建立事件(Create Event)。

2. Evloger(Event Loger)：每隔一段時間偵測佇列裡面是否有訊息，若有訊息，將訊息轉換為資料庫的資料格式，並將該筆資料寫入資料庫表格。為了減少磁碟機存取(Disk I/O)，加速寫入速度，我們採取每1000筆交付一次的作法。
3. Remover：依指定副檔名（或*.*）及保留時間查詢資料庫，符合條件者可執行搬移 (Move)或刪除(Delete)指令達到資料備存(Archive)的目的，其間並可設定過濾條件 (Filter)，讓程式可以略過某些符合條件設定的路徑檔案，不予以處理。

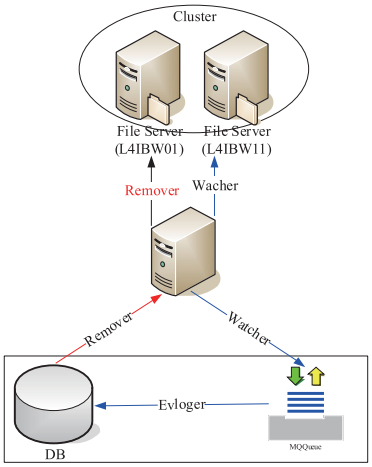

圖8：程式架構

（二）程式原理說明

整體程式採用Microsoft VB .Net程式語言撰寫，配合連結IBM DB2及MQ的函式庫呼叫(Library Function Call)在Microsoft Windows 2003 Server上執行，資料庫管理系統則是採用IBM DB2來實現。

1. Watcher程式說明：本程式使用由.NET所提供的系統類別(Class)即時監控檔案及目錄事件，當有檔案新增或變更名稱，就會立刻發出Event，以透過另一個系統類別，將檔案的相關資訊記錄下來，並組成字串(String)，再呼叫MQ API將這個檔案的資訊寫到佇列去。
2. Evloger(Event Loger)程式說明：本程式使用MQ API將存放在佇列的資料取出，並使用DB2所提供的嵌入式SQL(Embedded SQL)將資料插入到表格裡，如果已經讀取1000筆，就先交付再繼續讀取。
3. Remover程式說明：Remover程式會先過濾掉在KEEP_DIR.ini裡面的目錄不處理，查詢參數指定表格，再以參數所指定刪除的副檔名加上時間當條件，去找出所有符合條件的檔案名稱及路徑，並根據結果將該檔案刪除。保留天數可透過參數指定以天或小時為單位。

圖9為這個架構下新增檔案程式的作業流程，而圖10則為刪除或搬移檔案作業流程。

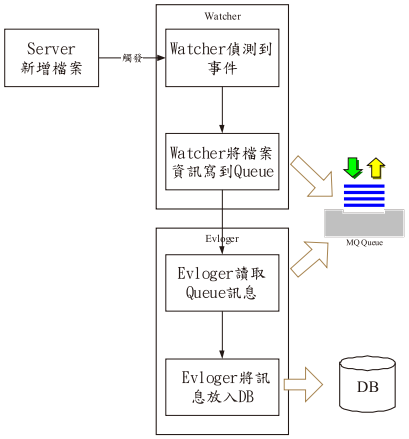

圖9：新增檔案作業流程

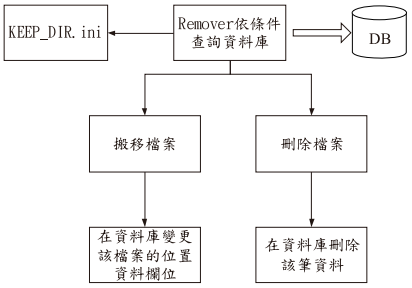

圖10：刪除或搬移檔案作業流程

二、分散式檔案系統架構

本研究參考Swift 的分散代理程式(Distribution Agent)，以及Petal分散磁碟機架構，另外設計出一套分散式檔案系統架構。以下範例使用8台PC伺服器當做最後資料存放的伺服器，再加上一台伺服器執行分散式代理程式(Distribution Agent)，以便讓所有使用者的檔案讀寫都透過該代理程式來進行。

（一）寫入檔案的機制說明

在機台需要寫入時，先把檔案存入暫存的伺服器，再透過程式依據雜湊的計算結果，求出虛擬關鍵值(Pseudo Key)，再轉換成實際檔案所在的位置，將檔案放到目的地伺服器。所有使用者端將資料寫到一台分散代理程式，再透過雜湊表(Hashing Table)的計算，將資料依據計算出來的結果，將資料搬到正確的伺服器中。

例如：假設User 1要將檔案A46A6751AJ06.jpg寫入伺服器中，則透過雜湊函數的計算，算出第9及第10碼為AJ的檔案編碼為9，再計算出虛擬關鍵值為1，故這個檔案應該放入Server 1。如圖11所示。

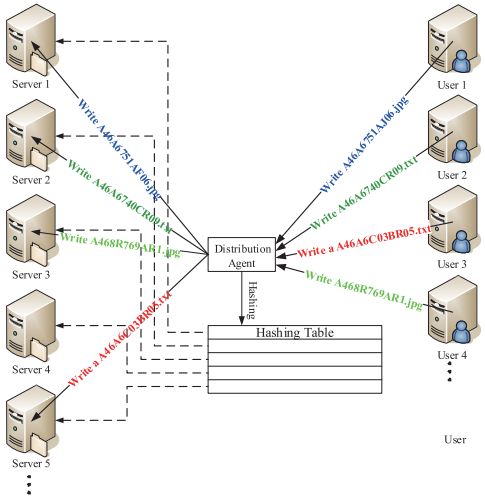

圖11：寫入檔案機制說明

（二）讀取檔案時的機制說明

使用者要檔案時，先對分散代理程式的伺服器發出需求(Request)，分散代理程式會根據雜湊表(Hashing Table)計算方式，算出檔案實際所在的伺服器，再回覆給使用者。使用者再直接存取該台伺服器。

例如：User 1要讀取的檔案名稱為A46A6751AJ06.jpg，透過雜湊函數的計算，我們可以知道第9及10碼為AJ的檔案，編碼為9，再計算出虛擬關鍵值為1，所以該檔案的所在位置為Server 1，User 1即可以直接去Server 1讀取檔案。如圖12所示。

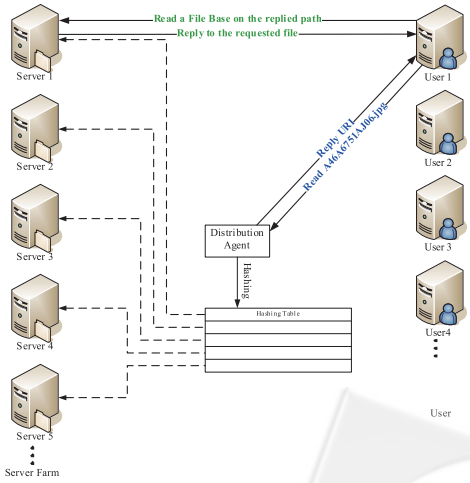

圖12：讀出檔案機制說明

（三）檔案名稱命名規則

在TFT-LCD工廠中，每一片玻璃都有一個唯一的識別碼，稱為玻璃識別碼(Glass

ID)，而玻璃切割後的面板則編有面板識別碼(Panel ID)，它是依據玻璃識別碼再延伸幾碼而形成。玻璃識別碼的編碼方式，前幾碼可能代表了廠別、日期、批號(Lot Number)，玻璃的流水號、玻璃在卡匣載具(Casette)的位置等資訊。

由於面板為機台最小批次處理單位，所以機台所拍照片的檔案名稱之前幾碼便是面板識別碼，例如：A468R769AR11.txt，表示玻璃識別碼為A468R769AR，而面板識別碼A468R769AR11則代表該玻璃所切割而成的其中1片面板之識別碼。

（四）雜湊函數與虛擬關鍵值 (Pseudo Key) 的選擇

本研究採用的雜湊方法乃參考延伸雜湊(Extendible Hashing)所使用的方法，並依據實際需求加修改，我們把經過雜湊函數(Hashing Function)計算出來相同結果的檔案放到同一台伺服器中，類似延伸雜湊方法中的分葉頁面(Leaf Page)。但是我們限制每個目錄頁面(Directory Page)只指向一個分葉頁面（Leaf Page，或稱Bucket），也就是一台伺服器，而且我們假設分葉頁面裡面的資料不會被重新分裝到別台新伺服器，以避免搬動大量的資料影響伺服器效能。所以修改延伸雜湊的部份如下：

1. 每個目錄頁面只指向一個籃子：本研究的架構中每個目錄頁面指向一台伺服器，所以不允許有一個以上的目錄頁面同時指向一個籃子。
2. 籃子(Bucket)夠大，不分裂：每個籃子即為一台伺服器，可以存放很多個檔案，況且以當今各企業級伺服器所擁有的儲存空間與硬體成長速度, 皆足以讓系統忽略碰撞的情況，因此不需要考慮籃子是否已滿，導致要分裂的問題。

由於我們希望每台伺服器都能有效地被使用，故本研究的架構有考慮到分散負載，希望所有的檔案可以儘量平均分配在所有的目錄(Directory)中。由C公司的TFT-LCD製造工廠之面板識別碼命名規則可以知道：前面1～5幾碼不能用來當鍵值(Key)，因為有可能同一天產生的檔案都集中到某一個目錄，即使使用第6碼，還是沒辦法平均分配，因在工廠裡面板移動是以卡匣載具(Casette)為單位。範例中取樣用的這個廠，一個卡匣載具最多可能有60片玻璃，在十分理想的情形下，這60片玻璃的前8碼有可能相同，要是切開成6片面板，則會有360片面板，每片面板可能會產生10～20個檔案，也就是總共會有3600～7200個檔案，而這些面板移動到各機台的時間將會十分接近，如果只採用這一碼，透過雜湊函數運算出來的虛擬關鍵值會全部相同，如此會導致檔案全部都在同一個目錄，也就是同一台伺服器，如此分散負載的效益將會降低。

例如，有幾個檔案是記錄面板識別碼為A46AG287CC06的相關缺陷資訊，而這幾個檔案是這片面板移動到這台機台時所量測出來的資料，所以時間會非常接近，如果取碼範圍為第1碼到第8碼，則會包括所有檔案名稱為A46AG287開頭的這些玻璃上的所有面板的檔案都會指向同一個目錄。

基於上述原因，至少要使用第9碼以後來決定關鍵值才能使檔案分配較平均，考慮未來擴充性，本研究採用第9及10碼，依編碼規則，此為A～Z，其中A＝1，B＝2，C＝3, …, V＝20（I和O一般來說並不使用，因為容易和數字的1和0混淆）。我們先將面板識別碼的第9及10碼的所有可能組合依照英文的先後順序列出，再分別指定1到60的數字給各

種組合(Gilbert, 1958)，編碼過後所產生的碼即為關鍵值。再將編碼後的數字，也就是關鍵值，除以*n*之後取餘數，再依不同的餘數放入不同的目錄分葉(Directory Leaf)。例如：A46AG287CC06，取第9及10碼，結果為CC，再進行20進位編碼後得到結果為43，如果假設有8個目錄分葉，就除以8，得到餘數為3，故放入Directory 3。

以下用圖13的幾個實際例子來說明修改後延伸雜湊的運作流程：

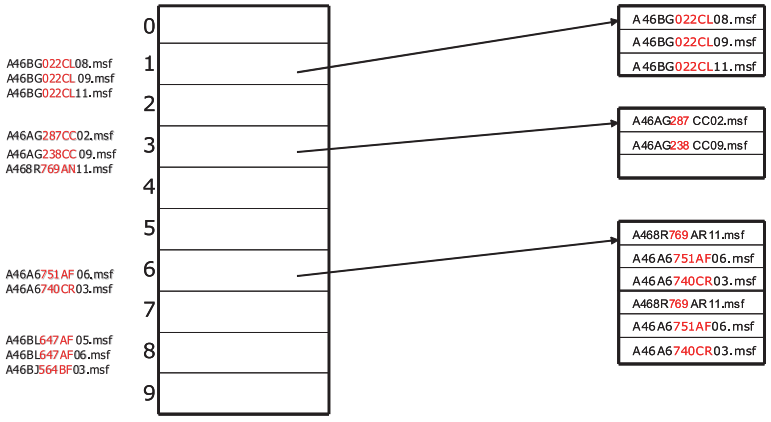

圖13：修改後的Extendible Hashing例子

機台寫入3個檔案檔名為A46BG022CL08.msf、A46BG022CL09.msf及A46BG022CL11.msf，取第9及第10碼編碼，為51，後再進行MOD(10)運算，得到虛擬關鍵值的結果為1，故這3個檔案都會被放到Directory 1。

機台寫入2個檔案檔名為A46AG287CC02.msf，及A46AG238CC09.msf，同樣的方法可以算出虛擬關鍵值的結果為3，故這2個檔案都會被放到Directory 3。

同樣的方法可以算出A468R769AR11.msf、A46A6751AF06.msf A46A6740CR03.msf、A468R769BR11.msf、A46BL647AF06.msf及A46BJ564BF03.msf都會放在Directory 6。

伺服器擴充數量時，只需要更改雜湊函數的變數，也就是本研究使用的MOD的參數。如果伺服器從8台變16台，只需要將MOD(8)改成MOD(16)，即可增加伺服器，並將檔案分配到新的伺服器，原有伺服器上的資料，只要有效保存期限過了，就會清除，所以擴充後只要一段時間，所有伺服器的檔案數就會平均分佈了。而且伺服器擴充數量時，舊有資料不需更動，以免影響正常存取的效能。

肆、驗證與評估

一、資料庫型態的檔案配置表系統架構效能

為了測試程式的效能，本研究交錯測試程式不同的設計方式，以比較不同的結果。

測試的方式為將2組相同數量的檔案複製到受監控的目錄，觀察程式監控到的檔案數量是否符合複製的數量，測試結果如表3所示，第2個欄位為複製的檔案數量，第3個欄位為複製檔案所花費的時間，第4個欄位為檔案的平均寫入數量，也就是第2個欄位除以第3個欄位所得到的結果。各項目為程式檔案資訊放到不同的目標所測試的結果，項目1及2為直接把資料插入表格，且每筆資料都交付(Commit)，結果顯示這樣的方法程式效能不佳，會導致部份檔案的資料沒插入表格。項目3及4為只計算監控到的檔案數，不做任何處理，目的是為了驗證程式是可以完全監控進檔案伺服器的檔案，結果顯示程式可以100%監測到所有的新增檔案。項目5及6則是每2000筆交付一次，加上IBM MQ後，項目7及8的結果顯示效能比一筆一筆交付來得好，而且沒有表格鎖定的問題。為了進一步了解程式的效能極限，我們另外用5000筆資料重複多做了項目9及10的測試，希望多測幾次，讓我們可以知道瓶頸為何？實驗結果證明：這樣的設計至少可以監控到每秒47個檔案寫入，而且幾乎沒有漏掉任何檔案資料，這樣的效能已足以應付最忙的檔案伺服器，而且透過執行多支程式監控不同的目錄可以避開這個限制。

表3：監控程式Watcher執行效能驗證

Item	File數量	時間(sec)	Files/sec	達成率(%)	Comments
1	16971	265	64	41	Insert DB, and commit for each
2	5657	130	43	59	Insert DB, and commit for each
3	16971	275	61	100	不做任何處理
4	5657	165	34	100	不做任何處理
5	16971	440	38	100	DB commit count 2000
6	5657	210	27	100	DB commit count 2000
7	16971	520	32	99	Put message to queue
8	5657	210	27	100	Put message to queue
9	5000	105	47.6	100	Put message to queue
10	5000	105	47.7	100	Put message to queue

由於複製檔案是使用系統指令達到目的，無法控制檔案寫入速度。實驗的結果也受限於主機及資料儲存設備的資源使用狀況，如快取(Cache)或硬碟忙碌程度，使得複製檔案測試的執行時間每次都不相同，導致每秒寫入檔案的數量無法準確控制。不過這樣的限制，並不影響我們對結果的判讀。

二、資訊生命週期管理系統架構執行成效

為了驗證本研究和傳統搜尋檔案條件的方法效能改善的效益，本研究以這兩種方法的程式分別刪除相同數量的檔案來比較結果。我們並且測試了3種數量的檔案數來比較差異。為了模擬新增檔案時所產生的表格資料，我們另外寫了一支程式將受測目錄的所有檔案資訊寫到表格中，以供後面Remover查詢之用。

三次測試的步驟如下：

1. 複製受測檔案到要測試的目錄中，如：E:\ Test5或F:\Test5。
2. 執行data_collect.exe 掃瞄所有檔案的資訊，並且把這些資訊插入到表格中。
3. 執行SQL select count(*) from db2admin.cftest5，比對表格筆數和檔案數量一樣。
4. 確認要刪除的檔案數量：執行SQL“select count(*) from db2admin.cftest5 where t_stamp <= (current timestamp－*n* days)”。
5. 執行remover.exe，附檔名為txt的檔案只保留*n*天。
6. 檢查剩下的檔案數和表格剩餘資料數量是否相等？

第一次測試檔案數為565，第二次為29,623，第三次為757,047。為了讓資料庫在資料筆數多的時候，能有最佳的效能，在執行remover.exe之前，先根據程式所用的SQL建立索引(Index)，程式裡用到的SQL為:

select T_STAMP, PATH, FILENAME, FILE_EXT, FLAG1 from
db2admin.cftest1 where (T_STAMP < (current timestamp－115 days)
and FLAG1 = 'N' and FILE_EXT = '.TXT')"

我們直接利用DB2提供的工具“db2advis”來為我們決定要建立的索引(Index)，從分析結果得知，如果在欄位“FILE_EXT”，“FLAG1”，“T_STAMP”，“FILENAME”，及“PATH”建立一個複合的索引(Index)，程式的執行效能會最好，改善幅度達96.44%，所以我們依據這個工具分析的結果建立這個索引，並且執行bind及runstats的指令將變更後的資料分佈統計資料寫到系統表格，以便程式執行時，DB2資料庫的Optimizer 可以依據正確的統計資料來得到最好的執行計劃(Execution Plan)。以下為建立索引前（圖14）及建立索引後（圖15），執行計劃的比較：從建立索引後的執行計劃中，可以確認建立的索引已正常運作。

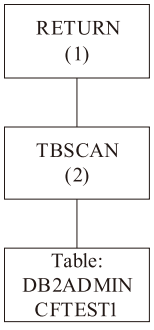

圖14：建立索引前的執行計劃

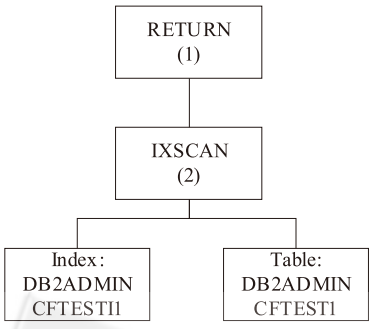

圖15：建立索引後的執行計劃

三次測試結果如表4所示，結果顯示當檔案數量不多時，二種刪除檔案效能差不多，透過資料庫查詢而不掃瞄所有檔案的程式效益不明顯，但是隨著檔案數量的增加效益愈來愈明顯，比掃瞄所有檔案的程式節省了約36%的時間。

表4：不同檔案數的測試結果

	第1次	第2次	第3次
檔案數	565	29,623	757,047
保留天數	110	47	115
Remover執行時間	00:00:17	00:23:13	1:36:49
直接刪除執行時間	00:00:19	00:25:45	2:30:11
剩餘檔案數	321	6,714	159,115
刪除檔案數	244	22,909	597,732

三、雜湊方法驗證

為了驗證雜湊方法可以符合資料平均分佈的需求，本研究採用前面資訊生命週期程式所收集的檔案資訊資料庫來分析可行性，並使用下列的SQL得到數據：select SUBSTR(filename, 9, 2), count(*) from ALEX.L4IBW11 where substr(path, 1, 10) ＝'\IBW04\IBW' or substr(path,1,10)＝'\IBW05\IBW' group by substr(filename, 9, 2)

從得到的結果看來，第1次取樣結果最小為4,612，最大為6,133，根據使用者需求，資料保存最久的時間為45天，所以分配到最多檔案的伺服器檔案數為1,655,910，這樣的檔案數只要使用一般小型的PC伺服器即足以應付。

表5：綜合比較

	MOD(10)			MOD(12)			MOD(16)		
取樣次樣	1st	2nd	3rd	1st	2nd	3rd	1st	2nd	3rd
MAX	6,133	6,715	9,272	5,233	6,143	6,021	5,394	7,001	7,885
MIN	4,612	5,874	6,406	4,121	4,458	4,162	3,538	4,450	5,105
AVG	5,373	6,295	7,839	4,677	5,301	5,092	4,466	5,726	6,495
誤差	14.16%	6.68%	18.28%	11,89%	15.89%	18.26%	20.78%	22.28%	21.40%
Num/4hr	56,026	63,518	78,920	55,955	63,349	62,708	70,480	90,766	106,129
Num/Day	336,156	381,108	473,520	335,730	380,094	376,248	422,880	544,596	636,774

目前資料保留天數依廠別、機台不同，從4小時到180天不等，表5為所得資料的分析比較，結果顯示分到最多檔案和分到最少檔案的伺服器，和平均檔案數相差約介於6.68%和22.28%之間。最後2列為伺服器實際現況，最後第2列為每4小時實際寫入檔案數量，最後第1列則為1天檔案數量的推算，也就是最後第2列的6倍。表7為某TFT-LCD 5.5代廠在2007年的新增檔案的數量統計。參考該廠所有需求比較的結果，此例子的使用機台的資料量為第二大量的機台，如果是最大資料量的機台，則每天是產生的檔案數為4,007,486，如果使用這個資料量並用最多的資料保留天數180天來估計，這些檔案分散在16台伺服器，每台伺服器最多將會存放45,084,217個檔案左右，這樣的資料量，一台PC伺服器搭配Windows 2003 Server仍足以應付。

表6：預計檔案數量

	檔案數量		Disk Space (GB)	
	Image File	Defect File	Online	Offline
File Size	100～1,600KB	<50KB		
TFT	1,065,144	57,792	267	12,794
CF	40,074,486	174,962	4,033	23,701
LCD	412,317	385,767	301	10,980
Total	41,551,947	618,521	4,601	47,475

表7：在最大量的需求下，每台Server的最大資料量

資料天數	10台Server	16台Server	20台Server	32台Server
1天	400,749	250,468	200,374	125,234
8天	3,205,989	2,003,743	1,602,994	1,001,872
30天	12,022,458	7,514,036	6,011,229	3,757,018
180天	72,134,748	45,084,218	36,067,374	22,542,109

伍、結論與建議

一、結論

本研究的資訊生命週期管理系統(Information Lifecycle Management, ILM)為利用程式監聽檔案系統新增檔案或變更名稱的事件，並將這個檔案的資訊組成字串，放到佇列中。另外有一支程式會去把佇列的資料拿出來，插入資料庫的表格。搬移過期檔案的程式先到資料庫的表格，將過期的檔案名稱和路徑找出來，再根據這兩項資料，找到該檔案，搬移到其他的伺服器或磁帶。

第二個部份是提出分散式檔案系統架構，分配檔案的方法是參考延伸雜湊(Extendible Hashing)的方法並略加修改，將使用者要寫入的檔案，依據檔案名稱重新編碼再用雜湊方式，計算出目的伺服器，重新將檔案分散到各伺服器，以達到分散負載的目的。本研究利用分析資料庫資料內容，驗證此方法可以有效地將檔案分散到所有的伺服器。

二、研究貢獻

本研究貢獻有兩個，第一個是利用資訊生命週期管理系統的概念，將伺服器檔案位置即時更新到資料庫，供刪除檔案時查詢用，以解決搜尋檔案對伺服器效能產生的影響，更可利用這個資料庫去分析伺服器的行為，如每分鐘進入伺服器的檔案數、有多少種型態的檔案、檔案寫入狀態是否有異常等。本研究針對分散式檔案系統所進行的檔案分佈情形評估，即是透過查詢該資料庫的資料所整理而成的結果。資訊生命週期管理系

統架構可以改善每天清除過期檔案時，嚴重影響系統效能的問題，而且可以縮短程式執行的時間。結果顯示，當檔案數量多時，至少可以節省1/3的程式執行時間。

第二個是以具有成本效益的分散式檔案系統，解決伺服器效能的問題。表8是以某C公司TFT-LCD 製造商的5代及7.5代廠當作例子，列出2種不同世代廠的伺服器在磁碟空間使用量、建置成本預估，以及使用傳統伺服器配合資料儲存設備，和使用本研究架構的概略建置成本進行比較。在分散式檔案系統架構的伺服器數量是以總共需要的磁碟空間數量，除以每台伺服器能提供的最大磁碟空間數而得到的。每台伺服器的磁碟空間，在硬碟機插槽全部裝滿，且扣除線上備用(Online Spare HD)及RAID-5損失的硬碟機後，可用磁碟空間約為876GB，如此可以算出此例子的5代及7.5代廠各需要38台和85台伺服器。在5代廠中，使用集中式方案成本約需14,000,000，而使用分散式檔案系統只需要約7,600,000。在7.5代廠中，使用集中式方案成本約需38,000,000，而使用分散式檔案系統只需要約1,700,000。從2種方案的參考售價的比較可以發現，使用分散式檔案系統約可節省建置成本約45%～55%的成本。

伺服器實際的售價則與伺服器生產廠商對該公司是否因為長期配合、專案金額較大，或是惡性競爭等因素，使得訂價與實際成交價格之間的價差可能會很大，表8所列舉的價格僅為一般合理的行情價格。

表8：兩種不同世代廠的磁碟空間使用及建置成本預評估

玻璃尺寸		G5	G7.5
作業系統		Windows	AIX
On-line需求空間		5,000GB	10,000GB
Off-line需求空間		28,000GB	64,000GB
Server+SAN Storage	訂價	41,000,000	180,000,000
	參考售價	14,000,000	38,000,000
	參考售價折數	約3～5折	約3折
Distributed File System	Server數量	38台	85台
	單台Server訂價	400,000	400,000
	參考總售價	7,600,000	17,000,000
	參考售價折數	約5折	約5折
比較	價差（元）	6,400,000	21,000,000
	價差%	45%	55%

TFT-LCD製造已是一種十分成熟的製造技術，全世界各TFT-LCD製造廠所使用的機台及產品製造流程差異都不大，故此架構不只適用於特定製造廠或特定世代廠，而是全世界所有的TFT-LCD製造廠都可以適用，甚至其他類似產業，如：IC製造或是太陽能電池製造業，如果有類似的問題，也可以依實際需求做部份修改後套用。

三、未來研究

（一）強化資訊生命週期管理系統 (Information Lifecycle Management System)

由於MQ及資料庫主機只有一台，易造成本系統的單點失效。資料庫失效時，如果MQ可以正常作業，則因為資料可以暫存佇列中，對整體作業影響不太，但是如果MQ沒有正常運作，雖然新增檔案仍然可以被程式偵測到，檔案的資料寫到佇列卻會失敗，導致沒有把資料放到佇列，資料庫表格也就不會有這些資料，整個ILM(Information Lifecycle Management)運作形同停擺。

要解決MQ單點失效的問題就要新增一台新的MQ伺服器，實現的方法至少有二個，第一個是利用程式去處理MQ失效的問題，每次程在寫入資料到佇列後，會判斷寫入是否成功，如果寫入失敗，就再把同一筆資料寫到另外一台MQ伺服器。另一個方法則是使用MQ Cluster，當程式寫入第一台失敗的時候，MQ客戶端會自動將連線轉移到第二台。

兩種方法各有優缺點，程式自己處理的優點是較具彈性，所有的東西都可以根據實際需求設定，缺點是程式較複雜，需要較多的判斷來處理異常狀況。使用MQ Cluster的優點則是可讓程式變得較簡單，缺點是用戶端的設定檔因為不是集中在伺服器，會有不易維護的問題。

（二）分散式檔案系統的瓶頸改善

由於分散代理程式(Distribution Agent)只有一台，效能瓶頸及單點失效(SPOF)可能發生在這台伺服器上(Thekkath, 1997)，要解決這個問題，可以使用二台以上的伺服器，再加上Microsoft Windows 2003 Server Cluster，即可解決單點失效的問題，至於效能瓶頸的問題可以用多台分派伺服器來加以改善。

除了使用獨立的分派伺服器外，也可以將雜湊函數整合到使用者的程式中，寫入時由程式直接判斷該檔案要放在那一台機器，讀取時也可以藉由所需檔案名稱及雜湊函數算出該檔案放在那一台機器而直接去該機器讀取。如此除了簡化架構之外，也可以避免分散代理程式(Distribution Agent)效能瓶頸及單點失效的問題。

另外可以參考PASIS架構(Wylie et al. 2000)，將所有的檔案重新分派時，複製為兩份或兩份以上，分別放到不同的伺服器，如此可以解決系統儲存設備單點失效的問題，也因為資料有二處可以存取，可以將線上伺服器改為非叢集架構，且改用等級較低的伺服器，降低成本，不過這樣的運作架構必須先確認複製資料的機制是可以正常運作的。

參考文獻

1. 黃華、楊德志、張建剛，2004，『分布式文件系統』，中國科學院計算技術研究所—資訊技術快報，第 10 期：4～5頁。

2. 展茂光電 (http://www.amtc.com.tw)，技術研發－CF製程。
3. 友達光電 (http://www.auo.com.tw)，TFT-LCD製程介紹。
4. 和鑫光電 (http://www.sintek.com.tw)，研發技術－彩色濾光片製程。
5. Anderson, T. E., Dahlin, M. D., Neefe, J. M., Patterson, D. A., Roselli, D. S., and Wang, R. Y.,“Serverless network file systems,”*ACM Transactions on Computer Systems* (14:1) ACM Press, February 1996, pp. 41-79.
6. Chen, Y.“Information Valuation for Information Lifecycle Management,”*Proceedings of the Second International Conference on Autonomic Computing(ICAC)*, IEEE Computer Society, 2005, pp. 135-146.
7. Davies, S. and Broadhurst, P., *WebSphere MQ V6 Fundamentals* Nov 2005, (available online at http://www.ibm.com/redbooks).
8. Du, D.H.C. and Tong, S.R.,“Multi-Level Extendible Hashing: A File Structure for Very Large Databases,”*IEEE Trans. on Knowledge and Data Engineering* (3:3), Sep. 1991, pp. 357-370.
9. EMC Corp.,“EMC File System Archiving for Windows/Centera Best Practives Guide,”(Rev. 1.0) Apr. 11, 2006.
10. EMC Corporation. (http://www.emc.com/ilm/)。
11. Enbody, R. and Du, D. H. C.,“Dynamic hashing schemes,”*ACM Computing Surveys* (20:2) , June 1988, pp. 85-114.
12. Fagin, R., Nievergelt, J., Pippenger, N., and Strong, R.“Extendible hashing-A fast access method for dynamic files,”*ACM Trans. Database Systems* (4:3), Sep. 1979, pp. 315-344.
13. Flouris, M. D. and Bilas, A.,“Violin: A Framework for Extensible Block-level Storage,”*In 13th NASA Goddard, 22nd IEEE Conference on Mass Storage Systems and Technologies (MSST'05)*, Apr.2005, pp. 83-98.
14. Ghemawat, S., Gobioff, H., and Leung, S.T.,“The Google file system,”*In Proc. 19th Symposium on Operating Systems Principles*, Lake George, New York, 2003, pp. 29-43.
15. Gilbert, E. N.,“Gray Codes and Paths on the *n*-Cube,”*Bell System Tech. Journal* (37), 1958, pp. 815-826.
16. Jones, T., Koniges, A., Yates, R.K.,“Performance of the IBM general parallel file system,”*Proceedings of the 14th International Symposium on Parallel and Distributed Processing (IPDPS'2000),* 1-5, May 2000, pp. 673-681.
17. Lee, E. K. and Thekkath, C. A.“Petal: distributed virtual disks,”*ACM SIGOPS Operating Systems Review* (30:5), Dec. 1996, pp. 84-92.
18. Long, D.D.E., Montague, B. R., and Cabrera, L.,“Swift/RAID: a distributed RAID system,”*Computing Systems* (7:3) Usenix, Summer 1994, pp. 333-359.
19. Microsoft Technet (available online at http://technet2.microsoft.com/WindowsServer/en/library/7e567bb0-59c7-413d-914d-cdcb55c1bbda1033.mspx?mfr=true)
20. Peterson, M.,“Information Lifecycle Management: A Vision for the Future,”*SNIA Data*

Management Forum, Storage Networking Industry Association, March 2004, pp. 4-5.

21. Reiner, D., Press, G., Lenahan, M., Barta, D., and Urmston, R., "Information Lifecycle Management: The EMC Perspective," *Proceedings of the 20th International Conference on Data Engineering*, Apr. 2004, pp. 804-807.
22. Storage Networking Industry Association. (http://www.snia.org)
23. Storage Networking Industry Association. (http://www.snia-dmf.org/ilmi/index.html)
24. Thekkath, C. A., Mann, T., Lee, E. K., "Frangipani: A Scalable Distributed File System," *Proc. Symposium on Operating Systems Principles (SOSP)*, Saint Malo, France, Oct. 1997, pp. 224-237.
25. Verma, A., Sharma, U., Rubas, J., Pease, D., Kaplan, M., Jain, R., Devarakonda, M. and Beigi, M., "An Architecture for Lifecycle Management in Very Large File Systems," *Proceedings of the 22nd IEEE-13th NASA Goddard Conf on Mass Storage Systems and Technology (MSST2005),* Monterey, USA, Apr 2005, pp. 160-168.
26. Wylie, J.J., Bigrigg, M. W., Strunk, J. D., Ganger, G. R., Kiliccote, H., Khosla, P. K., "Survivable information storage systems," *Computer* (33,8) Aug. 2000, pp. 61-68.