國立臺灣大學工學院工程科學及海洋工程學系

碩士論文

Department of Engineering Science and Ocean Engineering

College of Engineering

National Taiwan University

Master Thesis

類視訊壓縮之無線感測網路平行資料存取方法

Video-Like Compression for Parallel Data Access on Wireless Sensor Networks

林東輝

Tung-Hui Lin

指導教授：張瑞益 博士

Advisor: Ray-I Chang, Ph.D.

中華民國 102 年 7 月

July, 2013

誌謝

研究所這段時間雖然不長，但對我影響巨大，很感謝自己能在研究所中學到以前所沒學到的能力。很幸運的能成為張瑞益老師的學生，在研究所的這段時間，能學到與人如何在團隊中溝通協調的能力，學習如何解決問題的能力，這些都是我進研究所前所欠缺的，能在研究所這兩年內得到成長，非常的感激，包含計畫合作的老師、博班的學長育正，學長姐孃瑩、志永、元超，協助幫忙與指導。

在實驗室的生活中，很高興可以遇到很好相處的同伴們，涵仁、禹豪、億鑫、柏安、之盈，在煩悶無聊的時候有你們的陪伴，為生活帶來許多的快樂，不管是實驗和研究，或是出遊與玩樂，一路上有你們的陪伴，讓研究所的這段時間不會沉悶。還有學弟妹們，哲論、以宣、勤昇、玟君，不管是與你們合作或是解決你們的問題，從中我都學習很多，多虧有你們，實驗室的風氣也非常的和樂，希望你們未來能夠學習到更多，有豐碩的成果，也希望你們未來順順利利！

最後能完成這篇論文，我要特別感謝我的指導教授張瑞益老師，謝謝老師的指導與提點，也要感謝實驗室的每一位成員對我的支持與鼓勵，謝謝大家。

林東輝　謹致於

國立台灣大學　工科所　資訊與網路實驗室

中華民國 102 年 7 月

中文摘要

無線感測網路(wireless sensor networks, WSN)是由許多資源有限的感測器所組成，感測器可以蒐集並且監測環境上的變化[1]，無線感測網路能夠滿足環境監測上多樣的需求[2-4]，為了蒐集並儲存這些資訊，無線感測網路必須採取適當的方法來組織並且壓縮感測資料，否則，感測資料必定會佔據大量的存儲空間並且降低資料伺服器(data server)的效能。基於這些原因，我們之前提出類視訊無失真壓縮方法，稱為 VLLC (Video-Like Lossless Compression)，目標在於利用感測資料的特性。空間相關性以及時間相關性來提高資料壓縮率和減少資料壓縮的時間；在類視訊無失真壓縮系統中，會參考相關性將原始的感測資料無失真地轉換並排列成固定格式的資料幀(data frame)，而這些資料幀形成一個 3D 的立體像素(voxel)，並可用 H.264 進行視訊壓縮，由於立體像素結構的關係，資料能被直接存取而不需要解壓縮所有的壓縮檔案。本論文將針對類視訊無失真壓縮提供有效率的資料查詢流程，並提供語法讓使用者查詢感測資料，除此之外，我們還設計平行處理方法以提高壓縮和解壓縮速率，分析並比較不同資料擺放法(data placement)的效能，提高整體效率。在我們的實驗中，對於 4.53GB 的感測資料進行類視訊壓縮能較未壓縮省下超過 92%的資料空間，並且壓縮時間能低於 43 秒。另外，感測資料在 16 台平行處理平台下，採用合適的資料擺放法，將可比隨機擺放節省下 62%的處理時間。

關鍵字：無線感測網路、類視訊壓縮、平行化計算、無失真壓縮

英文摘要

Wireless Sensor Networks (WSNs) consist of groups of resource-restricted sensor nodes that collect sensory data and monitor environmental changes [1]. WSN environment services gather sensory data for various purposes [2-4]. To store the collected information, systems should organize and compress sensory data using proper methods. Otherwise, sensory data will occupy a large amount of storage and decrease the server's performance. In this thesis, we proposed a video-like lossless compression (VLLC), which aims to adopt the spatial correlation of sensory data in WSNs to enhance the degree of space saving and reduce the data compression time. In VLLC, systems will transform and arrange raw data as formatted video frames without loss according to the spatial correlation. The video frames form 3D voxels that can be highly compressed by H.264. Based on the voxel structure, data can be directly accessed without extracting all the compressed data. VLLC provides an efficient processing flow for querying sensory data and a query command that allows clients to access the proposed database. In our experiment, a space saving of more than 92% was achieved, and the data compression time for 4.53 GB of sensory data was less than 43 seconds. Furthermore, VLLC also offers a parallel processing method to enhance compression and decompression speed. To enhance the efficiency of the system, we also analysis and compare the different data placement methods. In our experiments, if we take proper personal data placement method, we will save 62% processing time with 16 personal computers more than random placement.

Keywords ： Wireless Sensor Networks; Video-like lossless compression; Parallel computing; Lossless compression

目錄

圖目錄

表目錄

第1章　　簡介

1.1　研究動機

無線感測網路是由許多資源有限的感測器所組成，有限的電量、計算能力、感測及傳輸的距離、以及無線的通訊能力。基於以上的能力，無線感測網路可以被利用在健康監測、軍事上的用途、或是智慧住宅，無線感測網路的應用同時包含了環境上的監測、監視，以及災害管理等[5]。雖然無線感測網路有著多樣的應用，但感測節點上的資源卻相當有限，有限的電池電力和儲存空間往往是無線感測網路最大的問題，在感測節點上，使用微控制單元 (micro controller unit, MCU) 以及無線通訊模組是耗電的主因，如果更進一步比較感測節點 MCU 計算和無線傳輸的耗電量可以發現，傳輸一個 bit 需耗費 3.91*uJ*，而 MCU 計算一個指令(add、compare)平均需耗費 3.72*nJ*。所以可得知傳輸一個 bit 的耗電量與 MCU 計算 105 個指令耗電量是相同的[6]。而在參考資料[6]發現無線測網路無線傳輸的耗電量往往是整體耗電量的 80%；因此，可以藉由減少傳輸的感測資料的資料量來降低感測節點的整體耗電量，而資料壓縮技術正是降低資料傳輸量相當重要的方法[7, 8]。

在許多的研究上可以發現，相鄰的感測節點蒐集到的感測資料之間存在著空間相關性以及時間相關性[9-11]，空間相關性往往存在自然環境之中；而時間相關性則存在長時間感測資料上的變化，無線感測網路資料壓縮可以藉由參考時間以及空間相關性來提升整體的效能。

1.2 目標與貢獻

在這篇論文中，我們選擇一個複雜度低並且有效率的壓縮/解壓縮方法，並且實作在資料伺服器上。由於在感測資料間存在著空間以及時間相關性這些多媒體的特性，因此我們提出了基於時間相關性以及空間相關性之類視訊壓縮(video-like compression)方法應用在無線感測網路上，這個壓縮方法可以同時提高資料儲存能力與檢索能力。除此之外，為了避免資料失真，我們使用了無失真壓縮方式 。

1.3 論文架構

基於上述的感測網路的特性，考量時間相關性與空間相關性，本研究提出類視訊無失真壓縮方法，稱之為 VLLC (Video-Like Lossless Compression)，VLLC 利用時間相關性與空間相關性來進行有效率的無失真壓縮，除此之外，本研究也提供支援平行處理的範圍查詢功能，提高存取效能。

本論文於第二章，我們將介紹資料壓縮方法的基本概念以及我們系統所使用的方法；在第三章介紹 VLLC 無線感測網路平行系統的系統架構與實作方法；在第四章我們在實驗中將 VLLC 與常見的資料壓縮法如 7-Zip [12]與 WinRAR [13]做比較，並探討資料擺放法對平行處理效能的影響，最後第五章總結與未來工作說明。

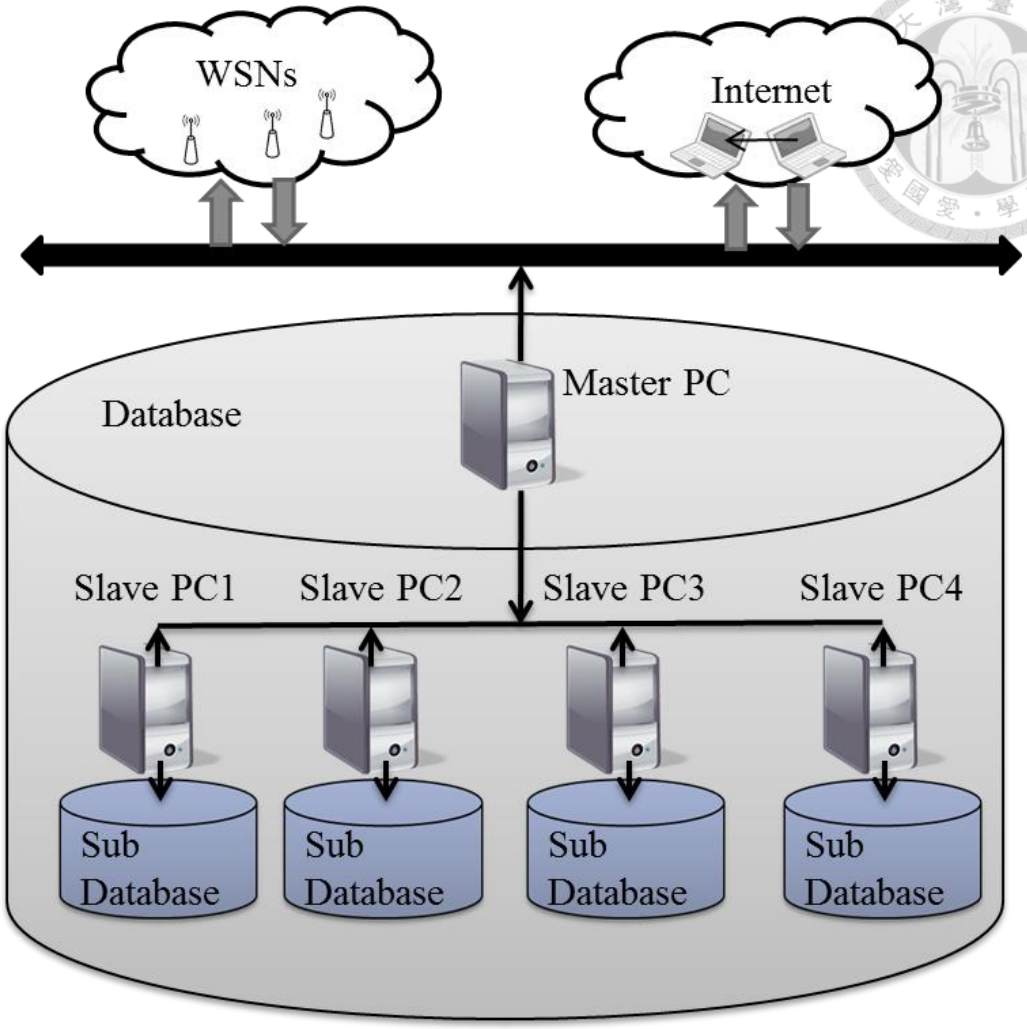


Fig. 1.1 無失真視訊壓縮資料庫之平行運算資料伺服器架構

在我們的方法上，資料伺服器具有平行處理能力，資料伺服器可以平行處理無線感測資料以及平行查詢感測資料，如 Fig. 1.1 所示，一個具有平行處理能力的資料伺服器是由一台 Master PC (master personal computer)和數台 Slave PC (slave personal computer)所組合而成，而每台 Slave PC 都有各自的子資料庫(sub-database)。在資料伺服器中，Master PC 負責控制監控，Master PC 同時也負責 Slave PC 上的感測資料儲存位置與感測資料處理上的決策。

在平行處理上，我們必須考量子資料庫的負荷以及儲存問題，為了使資料庫系統更有實用價值，我們提出了有效率的感測資料查詢服務。當使用者開始進行資料查詢，Master PC 將會分析資料查詢指令，以建立對應個別 Slave PC 的新資料查詢指令，並將對應指令傳給 Slave PC，再由 Slave PC 處理收到的查詢任務，我們定義這段過程所耗費的時間為查詢處理時間。

我們的工作同時包含感測資料的排列演算法(data arrangement)，以及感測資料的排列位置並壓縮，在這演算法中，資料伺服器依據時間將感測資料分成了片狀，而每一個片狀構成了一個資料平面(data plane)，在這些資料平面中，Master PC 會依據感測節點的環境中的擺放位置決定感測資料排列位置，當使用者查詢感測資料，資料伺服器將會依據排列演算法回傳被指定的感測資料。為了提高查詢的效能，影像平面可以分成數塊資料幀進行平行處理，我們定義連續的一段資料幀為 Voxel(立體像素)，並把連續的資料幀當作影片。因此我們可以無失真視訊壓縮來處理所有感測資料的立體像素，同時將這些立體像素存放在資料庫中。

在我們的無線感測資料庫中提供資料查詢的服務，當一個使用者要查詢單一筆資料，系統將會搜尋所有的子資料庫並取出目標的資料幀，並且解壓縮資料幀取得所查詢的感測資料，我們也提供範圍資料查詢(range data query)，在不同的子資料庫之中，Slave PC 從各個子資料庫取出目標資料幀並且解壓縮，Slave PC 解壓縮過目標資料幀，接著再將資料幀傳送到 Master PC，Master PC 最後再整理合併這些解壓縮過的資料幀。

在我們的資料庫系統中，我們利用感測資料的時間與空間相關性提高無失真視訊壓縮整體效能，藉此可以更有效的資料存取，同時我們的無線感測網路資料庫系統能支援平行處理應用在壓縮以及解壓縮，我們將可提出一個穩定並且有效率的無線感測網路資料存取服務。

第2章　　文獻探討

2.1　資料壓縮方法

資料壓縮方法通常會捨棄多餘的資訊來提高壓縮率，資料壓縮基本上分成兩個種類，無失真壓縮以及失真壓縮。影像和視訊失真壓縮允許壓縮時捨棄部分資料，雖然近乎所有失真壓縮的壓縮率比無失真壓縮來得高，節省更多的儲存空間，但是我們某些情況仍需要無失真壓縮，舉例來說，當必須傳輸相當重要感測資料，例如監測病人生命跡象或是傳遞文字資料，在這些情況之中，資料是絕對不能在壓縮中被改變，無失真壓縮都是可逆的，壓縮後的資料都可轉換為原始的資料[14]。在無線感測網路常用的無失真壓縮方法中，已有許多利用感測資料時間及空間相關性來提高壓縮率的方法；JPEG2000[15]、JPEG-LS[16]、SFALIC[17]為感測網路中常見使用空間相關性的無失真壓縮演算法，而在感測網路中使用時間相關性的無失真壓縮演算法則有 SHC[18]、MAHC[19]等等，因此在特定的應用之中，無失真資料壓縮在無線感測網路中值得深入探討與開發。

2.2　資料伺服器

在無線感測資料庫系統裡，資料伺服器負責蒐集並儲存由大量的感測結點傳來的感測資料，同時負責進行資料分析及查詢，資料庫系統在資料伺服器中管理並存放大筆的感測資料。現在普遍在無線感測網路的資料庫分為兩類，集中式資料庫和分散式資料庫兩類[20]，Oracle[21]、SQL server[22]和 MySQL[23]都是集中式資料庫，將資料蒐集資料查詢等工作大都在主機中完成。由於在無線感測網路中，感測節點運算能力以及儲存空間都是分散的，因為分散式的方式更適合無線感測網路，分散式資料庫將所有的感測節點也視為資料庫的其中一部份，而目前

常見的分散式資料庫有 TinyLime[24]、TinyDB[25]和 COUGAR[26]等，這些分散式資料庫都可以做到即時的資料查詢和資料檢索。兩種不同的方式都是設計來處理大量的感測資料。

2.3 時間相關性與空間相關性

若我們需要在資料庫使用資料壓縮法，分析並研究壓縮相關性可以提升感測資料壓縮率，在資料壓縮機制裡，相關性主要可以分為三個種類[27]，資料相關性、時間相關性、空間相關性。變動長度編碼法(run-length coding)就是在資料庫裡相當常見利用資料相關性的壓縮法，而在無線感測網路中，感測節點常用來監測自然環境中的長時間持續的變化，像是溫度、濕度、氣壓等等，這些變化長時間下來通常變動相當緩慢，因此感測環境中同一感測節點，長時間持續所監測的感測資料變動不會太大，因此我們可以得知感測資料存在著時間相關性，此外若是感測節點的距離相當接近，那麼相近的感測節點所監測的感測資料之間的差異也不會太大，感測資料必然存在著空間相關性。由於感測節點存在著時間相關性與空間相關性這些多媒體的特性存在，因此我們決定將視訊壓縮方法應用在感測資料，在類視訊無失真壓縮方法中我們將感測節點每次蒐集到的感測資料當作資料幀上的畫素(pixel)，時間相關性如 Fig. 2.1 所示，資料幀上的畫素如之前所提，感測資料在感測環境中長時間持續下的變動並不大，空間相關性如 Fig. 2.2 所示，我們將在感測環境中，依照位置將相近的感測節點所監測的感測資料排列在一起，藉此提高感測資料的空間相關性，由於我們有效地利用資料幀中時間和空間相關性，因此可藉由時間相關性和空間相關性提高整體感測資料的壓縮率，一旦提高壓縮率必然可以減少感測網路中的資料傳輸量，達到我們延長感測節點壽命的目標。

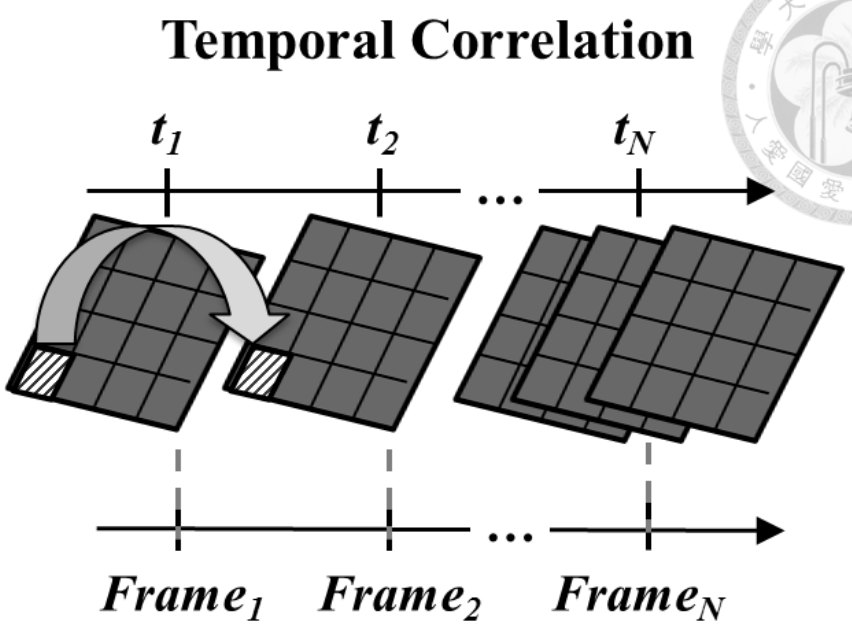


Fig. 2.1 時間相關性

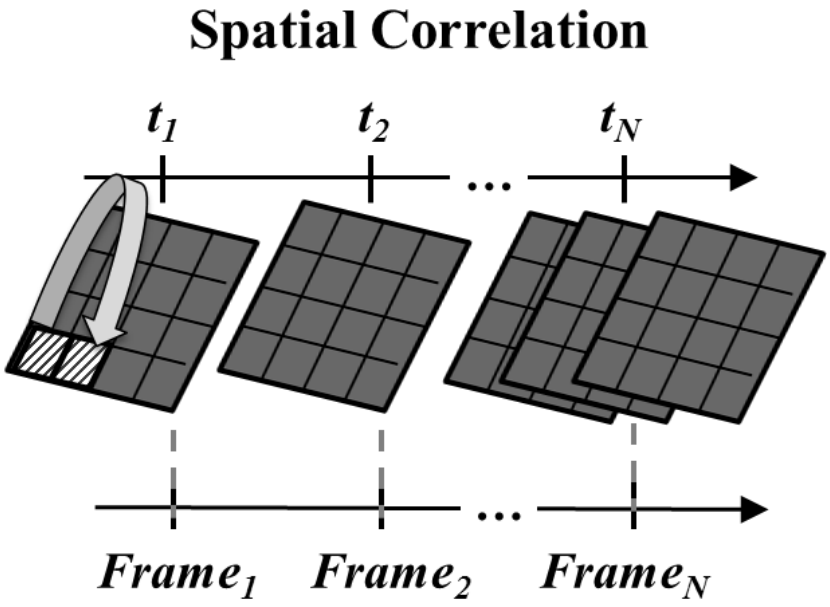


Fig. 2.2 空間相關性

第3章　　系統概觀

3.1　系統架構

在介紹 VLLC 實作的方法前，我們必須先說明在無線感測網路上的資料蒐集。在一般無線感測網路的樹狀結構中[28]，大致上可以分為兩個部分：無線感測網路端與資料伺服器端。而在樹狀結構的無線感測網路之中，如 Fig. 3.1 所示可以簡單地分為三層，最下層通常會擺放著電力有限的終端節點(end node)，終端節點監測並蒐集感測環境中的變化。在中間這一層，超級節點(super node)通常只負責資料傳送或信息轉送。最上層的協調節點(coordinator)必須處理一些相當重要的工作，像是結構管理和資料聚合等等。大部分的協調節點都是連上資料伺服器或是接上電源的，所以較不受到電源或是儲存空間的限制。

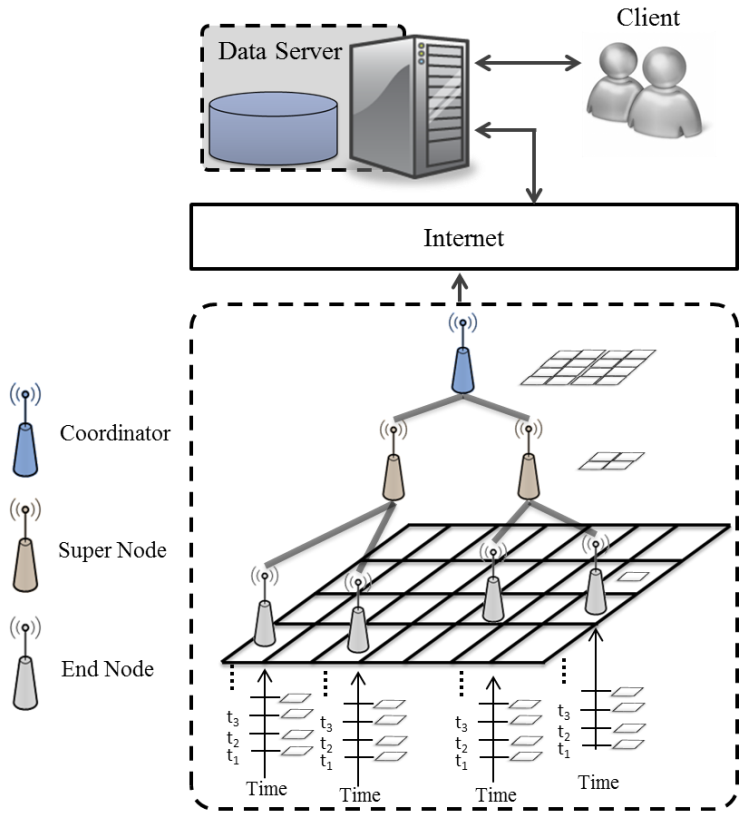
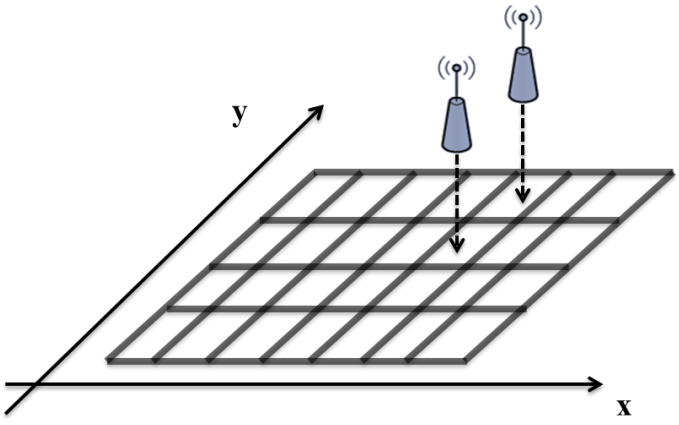


Fig. 3.1 無線感測網路的樹狀架構

在樹狀結構裡，所有的終端節點都與上一層的超級節點相連，而每一個超級節點都與最上層的協調節點相連，在每隔一段固定時間，終端節點傳送感測資料到上層的超級節點，協調節點與超級節點負責協調與管理感測資料並傳送到資料伺服器。而無線感測網路平行系統則設置在無線感測網路的資料伺服器上，管理並組織由下層傳上來的歷史感測資料，使得使用者能夠檢索資料並分析。

如同之前所提，我們會排列整理蒐集到的感測資料，並將感測資料轉為資料幀，而每一個資料幀都會有特定的基準來進行資料排列，為了達到這個基準，每一個終端節點都會分配到一個不同的編號；接著我們系統會對應物理層的排列，因此我們可以簡單並且快速的對應終端節點的編號。如 Fig. 3.2 所示，每個不同的編號都是由系統所分配的，而資料檢索表(mapping table)是用來對應資料排列後的結果。



Lookup.Table				
$Node_1$		$Node_i$		$Node_n$
(X_1, Y_1)		(X_i, Y_i)		(X_n, Y_n)

Fig. 3.2 編號由系統所分配，記錄在 Mapping Table 上

3.2 VLLC 實作方法

有效的節省資料庫儲存空間促進我們打造出更實用且更有彈性的資料庫系統。這個 VLLC 資料庫系統如 Fig. 3.3 所示，系統整合了資料伺服端與感測網路使得我們能提供有效率的資料傳送，資料壓縮以及儲存的服務。從無線感測網路蒐集到感測資料大都是沒有整理過的，我們必須提供一個資料排列方法去整理這些感測資料來提資料的相關性。我們排列感測資料時會依據地理位置在適當的位置進行排列，由於有好的資料相關性，我們將可以達到好的資料壓縮率，如 Fig. 3.3 所示，我們的資料庫系統可以轉換感測資料為影片，接著再將這些影片儲放在緩衝區，一旦達到一定的資料量大小，資料庫系統再進行壓縮工作。

假設感測節點佈置在感測環境中，並定期地在環境中蒐集感測資料，接著以 M 筆感測資料以固定時間為單位。我們再將這些感測資料依照蒐集到的時間分成不同的群組，接著將這些群組放在格式化的平面上，就像是將這些感測資料轉成時間的資料幀，而連續的資料幀也就是我們要壓縮的影片；可以發現到資料排列的規則在搜尋資料檢索表，資料檢索表上保存著每一個感測節點的位置資訊，而資料檢索表也代表著平面的資料量大小，由於感測資料的排列能影響壓縮的相關性，選擇一個適當的資料排列法將可提高資料壓縮率。

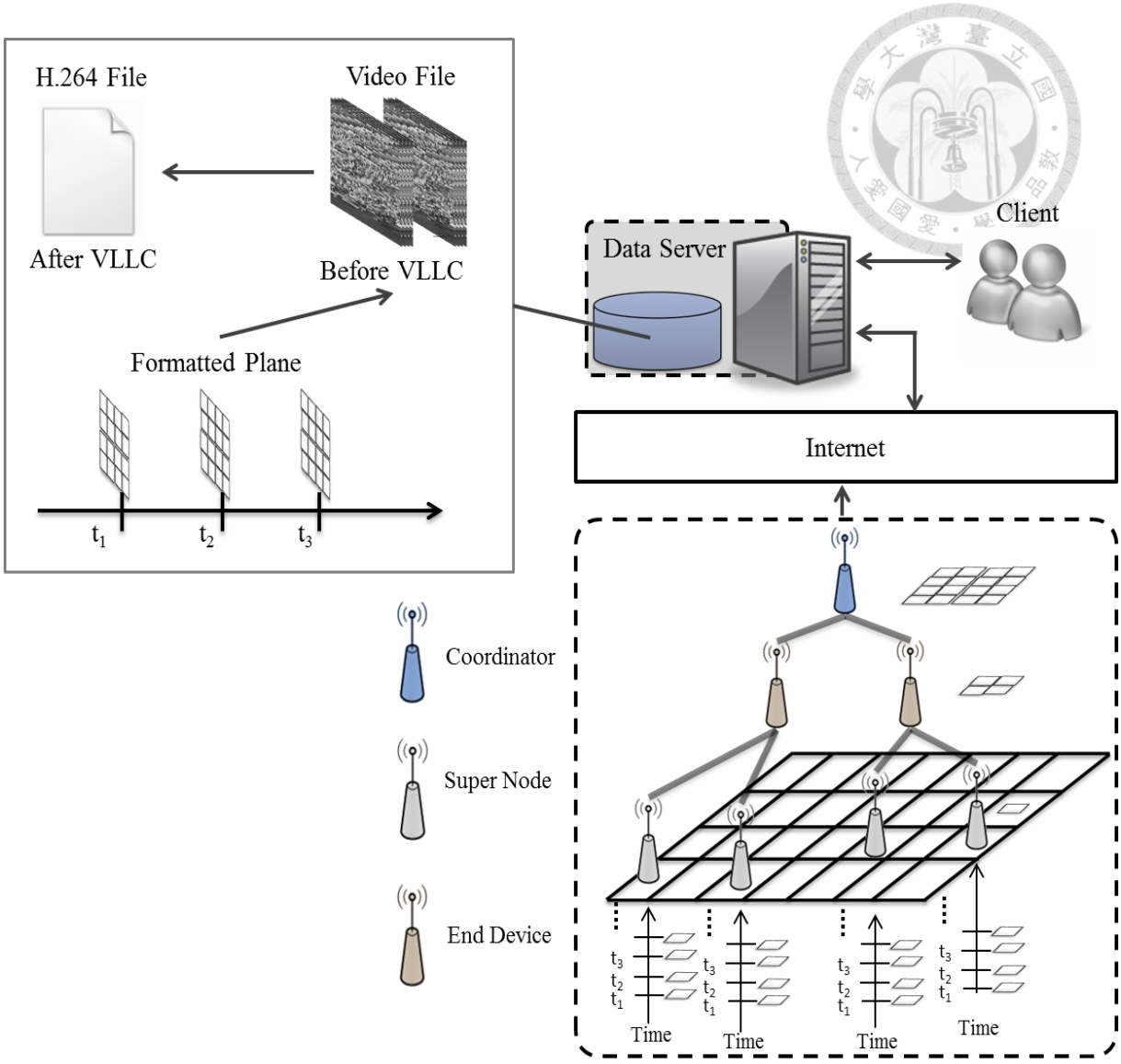


Fig. 3.3 無失真視訊壓縮資料庫系統架構

在資料伺服器端，我們可以根據感測環境中節點的數量決定資料幀的尺寸，若是感測環境裡節點數量不多，則資料幀的尺寸可以縮小，反之，感測節點越多，則需要更大尺寸的資料幀來存放感測資料。連續的資料幀構成了一個面積更大的平面，除此之外，這些資料幀可以調整不同的尺寸，而資料幀的尺寸就代表著影片的解析度(resolution)，系統能保存所有這些在資料排列中所有尺寸變化的資訊。

由於我們提出的資料伺服器提供平行處理，可以同時維持數個子資料庫，所以資料伺服器能夠將這些感測資料分散式地存放感測資料，如同 Fig. 3.4 所示，一個大平面分成了四個子部分，而這四個子部分也就是四個不同的立體像素，四個不同的立體像素可分別存放到不同的子資料庫中，連續一段時間 T 的子平面我們稱之為立體像素，而每個時間片段我們稱之為資料幀。

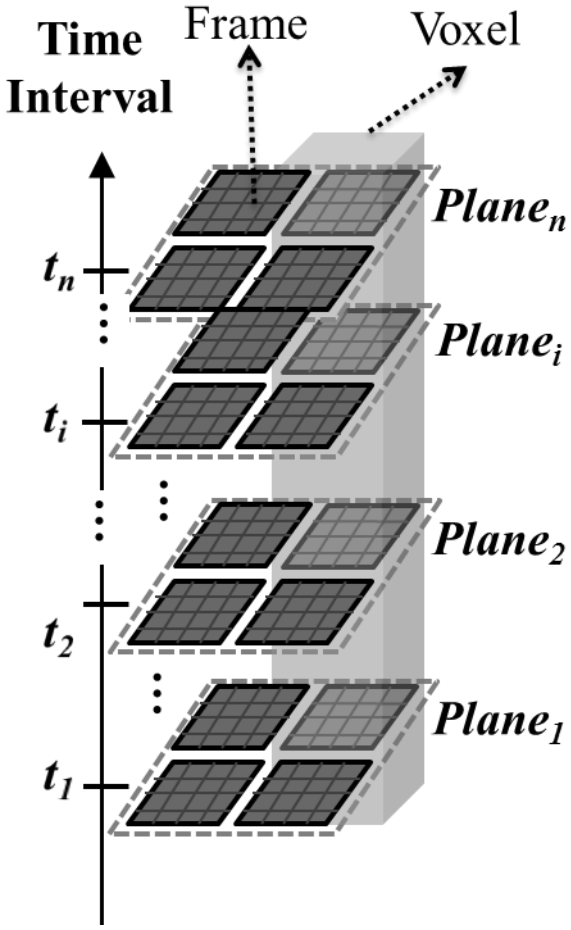


Fig. 3.4 感測資料存放在無失真視訊壓縮法資料庫

而資料幀在系統中，感測資料排列主要是依據感測節點的排列位置，因此資料幀存在著空間相關性，帶著相關性的連續資料幀可以視為一個影片，所以資料庫中的每個立體像素可以使用無失真視訊壓縮進行壓縮工作。

帶著相關性的感測資料有高度的多媒體特性，我們利用這個特性來達到更高資料壓縮效果，在資料排列後，我們使用平行處理分別壓縮不同的立體像素，再將每塊分散的立體像素存放在不同的子資料庫之中，並且採用 H.264 編碼方式在我們的壓縮方法中，除此之外，我們不需要參考所有的資料幀即可解壓縮並取出單一指定的資料幀，這點對於提高查詢的效率上有著非常顯著的幫助

3.3 Query Process in VLLC

3.3.1 Single Query

我們將所有的感測資料當作影片分散地儲存在不同的子資料庫之中，而單一查詢是非常直覺的，資料庫系統在指定的影片中取得目標的資料幀，如 Fig. 3.5 所示，例如單一查詢要在資料庫查詢檔案 $data_i$，當單一查詢開始後，Slave PC 先於資料檢索表找尋感測節點 $node_i$，查詢 $node_i$ 存放在哪一個 Slave PC 的子資料庫之中，接著傳送查詢指令到 Slave PC。Slave PC 負責查詢子資料庫裡對應的立體像素，由於 Frame.offset 存放著資料幀 $Frame_i$ 號碼，Slave PC 使用 Frame.offset 找到 $Frame_i$，接著，解壓縮資料幀 $Frame_i$，從解壓縮的資料幀即可取得目標檔案 $data_i$，最後系統再將 $data_i$ 轉回原來的資料型態。而在單一查詢所花費所有的時間我們稱之為 T_{total}，從子資料庫裡找尋檔案 $data_i$ 的這段時間稱之為 T_f，在另外一方面，我們將壓縮後的 H.264 檔案解壓縮的時間稱之為 T_d，由於目前只查詢到單一個子資料庫，引此查詢總時間很直覺得的推導出來是 $T_{total} = T_f + T_d$。在我們所提出的 VLLC 資料庫系統，我們同時提供感測資料存取的指令，幫助使用者能更快熟悉並使用我們的資料庫。

Query ($(x_i, y_i), t_i$)

在這個例子中，x_i 和 y_i 代表該畫素在資料幀中的位置，而 t_i 則是代表該資料幀在影片的序列，在使用者輸入了資料查詢的命令後，VLLC 資料庫系統將會解析命令，VLLC 資料庫系統便會回傳有著該畫素的資料幀，接下來 VLLC 資料庫解壓縮該資料幀並且將資料幀轉回原本的資料型態。

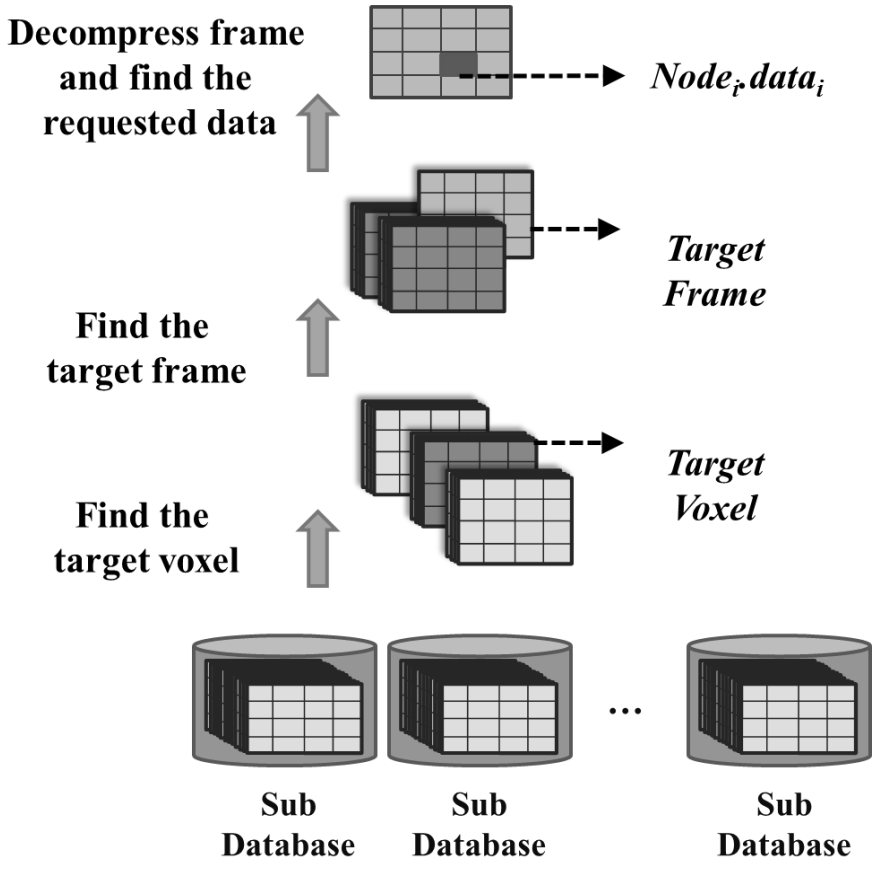


Fig. 3.5 單一查詢在 VLLC 資料庫中

3.3.2 Range Data Query

研究員不只需要單純一筆資料而是需要一些長時間跟固定範圍的大筆資料查詢，無線感測網路的資料用在科學研究或是健康監測的應用上需要範圍查詢，範圍查詢紀錄了一個範圍內的感測節點在一段時間內的環境變化，如 Fig. 3.6 所示，

(x_1, y_1)到(x_2, y_2)形成了一個範圍，而這個範圍跨越了不同的立體像素，這個座標代表著這些感測資料跨越了不同的子資料庫。當使用者要進行範圍查詢，Master PC 第一步查詢資料檢索表，並將查詢指令傳給必要的 Slave PC，Slave PC 選出目標資料幀進行資料解壓縮和資料傳輸，每台 Slave PC 能夠獨立進行這個工作並且支援平行化，最後再由 Master PC 將這些分散的感測資料進行合併的動作，當感測資料合併完成後就是範圍查詢所需要的資料。

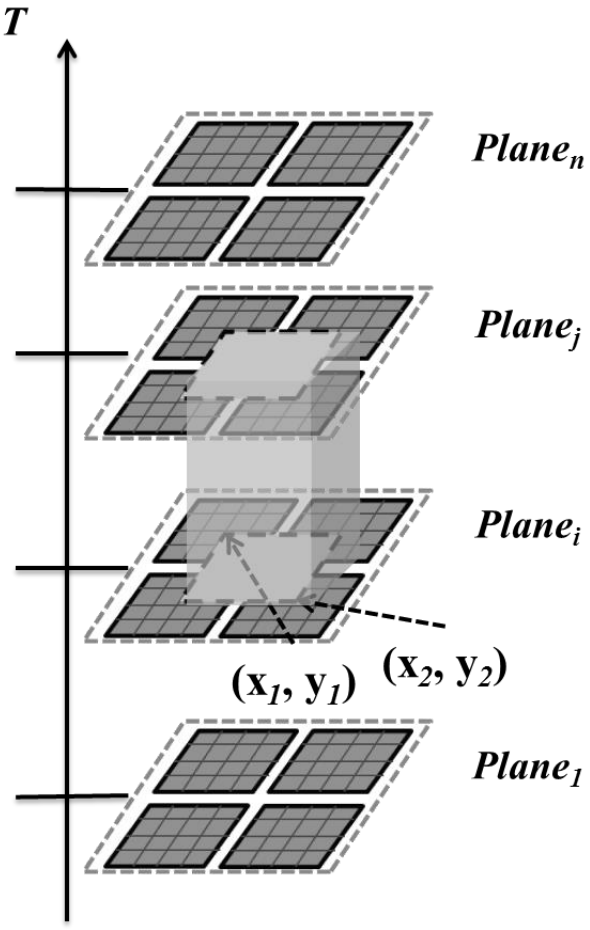


Fig. 3.6 範圍查詢在 VLLC 資料庫中

Query ($(x_1, y_1), (x_2, y_2), t_i$)

當一個使用者想要去觀察某時間點特定的區域的變化，由於我們的系統之中排列感測資料是按照感測節點的位置，因此當使用者輸入兩個座標(x_1, y_1)和(x_2, y_2)，等同代表了在平面裡選擇了一個矩形的區塊，而 t_i 如同上一個的例子代表該資料幀在整個影片裡的序列，如同在範圍查詢裡所提的，這個範圍可能會涵蓋到不同的子資料庫，VLLC 資料庫收到使用者輸入的查詢命令，並且把查詢命令解析後，將查詢命令傳給涵蓋到的子資料庫，Slave PC 再將所有的目標資料幀解壓縮並取出所需的感測資料。

Query ($(x_1, y_1), (x_2, y_2), (t_i, t_j)$)

在無線感測網路資料庫中，使用者不僅僅是要分析特定的區域於某個時間點上的變化，而是要分析特定的區域長時間上的變化，因此只要再多增加時間點 t_j，有了時間點 t_j，VLLC 資料庫則傳回連續的資料幀並進行解壓縮，最後再將資料幀的感測資料轉回原來的資料型態。

3.3.3 Query Scenario

由於我們的目標是用將無線感測網路端以及資料伺服器端整合成一個完整的資料庫系統，我們必須考慮到所有的資料查詢時可能發生的情況，VLLC 資料庫整合了資料伺服器端與無線感測網路端，終端節點蒐集到固定數量的感測資料時，進行無失真壓縮並傳送，終端節點儲存一定資料量大小的感測資料，若資料超過時效則可直接捨棄，超級節點只負責資料傳送或信息轉送，資料伺服器整合由無線感測網路端傳送來的感測資料，不須解壓縮直接儲存在資料庫中，所有的感測

資料儲存在資料伺服器中。

為了有效壓縮資料，終端節點會收集滿緩衝區(buffer)大小的資料(m 筆資料，設為 m 單位時間)再往超級節點送。但是，使用者查詢隨時可能發生。我們假設目前時間為 c，查詢時間範圍是[t_L, t_R]，而 t 則代表前一次收集資料的時間，t_L 是起始的時間而 t_R 是結束的時間，則 t 與[t_L, t_R]有 Fig. 3.7 以下三種可能關係，由於我們在查詢時可能所需的感測資料仍在無線感測網路端，因查詢時而上傳過的感測資料，希望不再重複上傳，所以下一次上傳的 m 筆資料如下，其上傳時間為 t’。

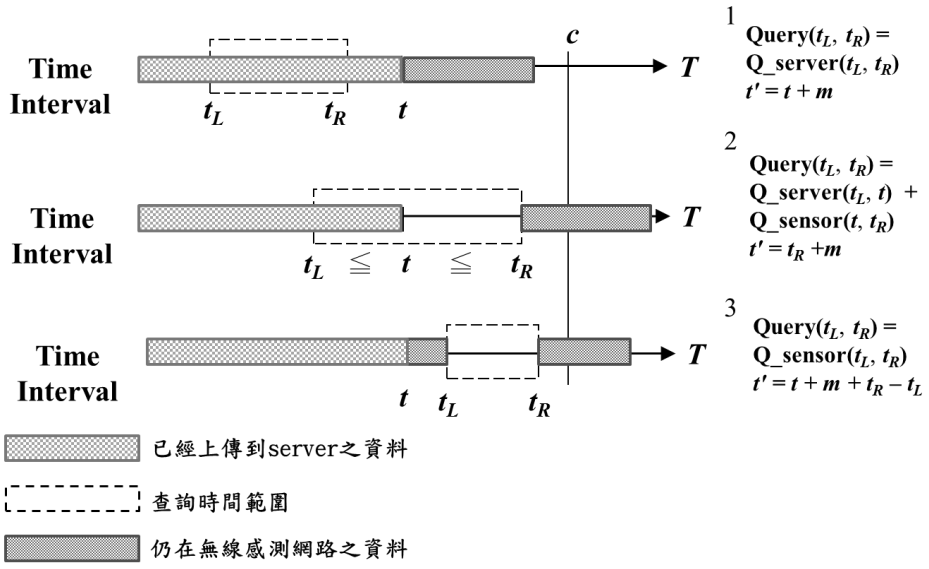


Fig. 3.7 感測資料查詢情節

Query(t_L, t_R) = Q_server(t_L, t_R)

在 Fig. 3.7 情節 1 中，所查詢的感測測資料剛好都在資料伺服器中，這個情節比較單純，資料庫系統只須回傳在資料伺服器中的檔案所以上傳時間為 t’＝t＋m。

Query(t_L, t_R) = Q_server(t_L, t) + Q_sensor(t, t_R)

在 Fig. 3.7 情節 2 中，所查詢的感測資料並沒有完全地存放在資料伺服器中，我們必須向無線感測網路端進行查詢，而無線感測網路端必須將目標資料幀壓縮後上傳到資料伺服器，最後資料伺服器端再將兩邊的資料組合；並一起回傳給使用者，其下一次上傳時間為 $t' = t_R + m$ 。

Query(t_L, t_R) = Q_sensor(t_L, t_R)

在 Fig. 3.7 情節 3 中，所查詢的感測資料不在資料伺服器中，而是全部都在無線感測網路端上，系統必須從無線感測網路查詢所需的感測資料，由無線感測網路端將感測資料壓縮；再往資料伺服器端進行回傳，並把感測資料回傳給使用者，最後將所查詢感測資料儲存在資料伺服器中，由於所查詢的 t_L 到 t_R 已存放在資料伺服器中，t 到 t_L 可以在回傳查詢的感測資料直接存放在資料伺服器中，所以查詢後的下一次上傳仍舊，是上傳 m 單位感測資料，所以下一次上傳的時間仍為 $t' = t_R + m$。

3.4 Parallel computation in data server

若 Slave PC 是在單核心的平台上執行，資料查詢將會是一串序列的工作，但如果我們的 Slave PC 是雙核心或是多核心的平台，我們則可以使用平行化處理，像是解壓縮可以分成數個任務來進行，而每個任務可以分配給不同的個人計算機，因此我們使用平行化處理在無線感測網路資料庫系統之中，如此一來我們可以減少資料解壓縮的時間，並且提高資料伺服器的性能。

假設我們的資料庫是運行在四台個人計算機的平台上，當一個使用者進行資料查詢，如 Fig. 3.8 所示要查詢的資料可以分成四個子部分，將每個子部分同時分給個人計算機進行處理，在經過平行化的解壓縮和轉換後，再將這些分離的部分

進行組合，我們可以參考資料檢索表得到使用者所需求的檔案，當一個高解析度的資料幀進行解壓縮的任務，系統將會將這資料幀拆成小的子部分，再將每個子部分一一分配給個人計算機，如此一來可以降低處理時間並優化查詢的效能。

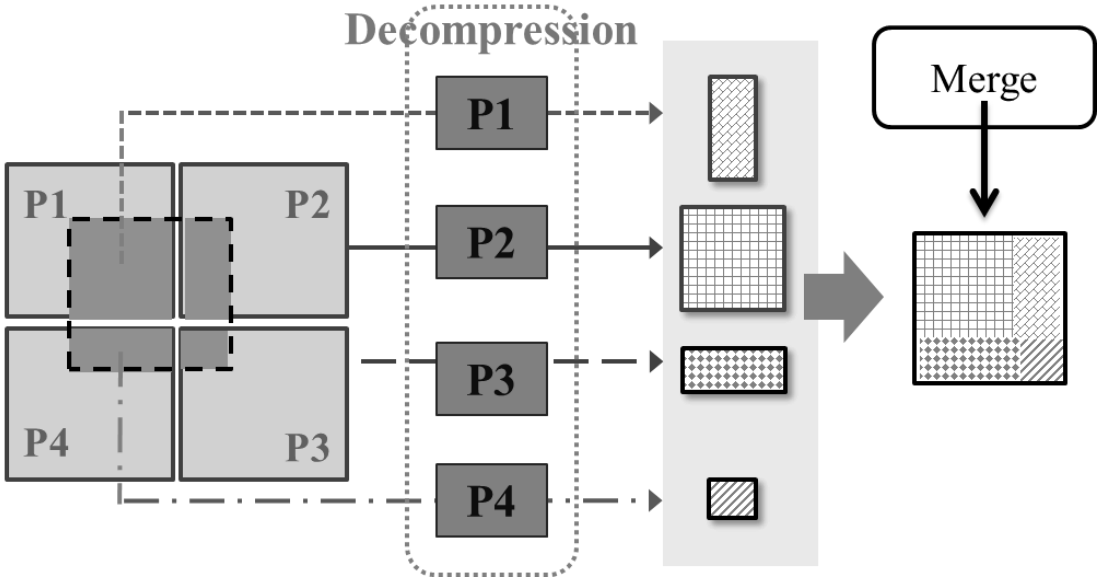


Fig. 3.8 平行化解壓縮運行在四台個人計算機的平台上

我們設定解壓縮一個子部分所花的單位時間稱之為 tp，如 Fig. 3.9 所示，個人計算機 P1 需要 2 個 tp 單位時間來完成解壓縮的工作、個人計算機 P2 需要 4 個 tp 單位時間來完成、個人計算機 P3 需要 2 個 tp 單位時間完成、個人計算機 p4 則只需要 1 個 tp 單位時間完成，Fig. 3.9 這個例子可以得知這樣的擺放方式需要 4 個 tp 單位才能合併我們所查詢的感測資料。

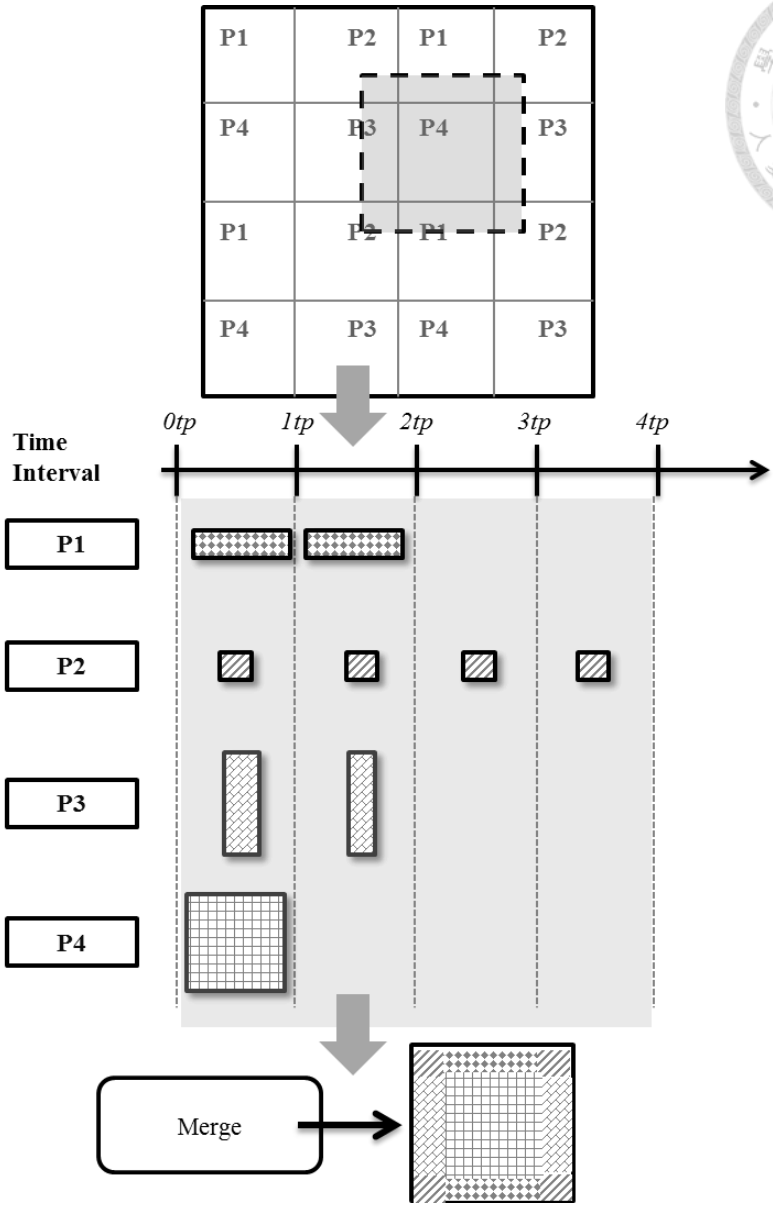


Fig. 3.9 平行解壓縮運算在未優化的資料擺放法上

但是如果擺放方式不同，我們便可以改變或是提高查詢效能，如 Fig. 3.10 所示，最大的解壓縮單位時間為 3 個 tp，比上一個模組的例子少了 1 個 tp 的單位時間，並且 Fig. 3.10 的單位時間就是最佳化的單位時間，3tp 單位時間，所以由這兩個例子可以得知，只要有適當的個人資料擺放法，我們將可以得到更好的處理時間和效率。

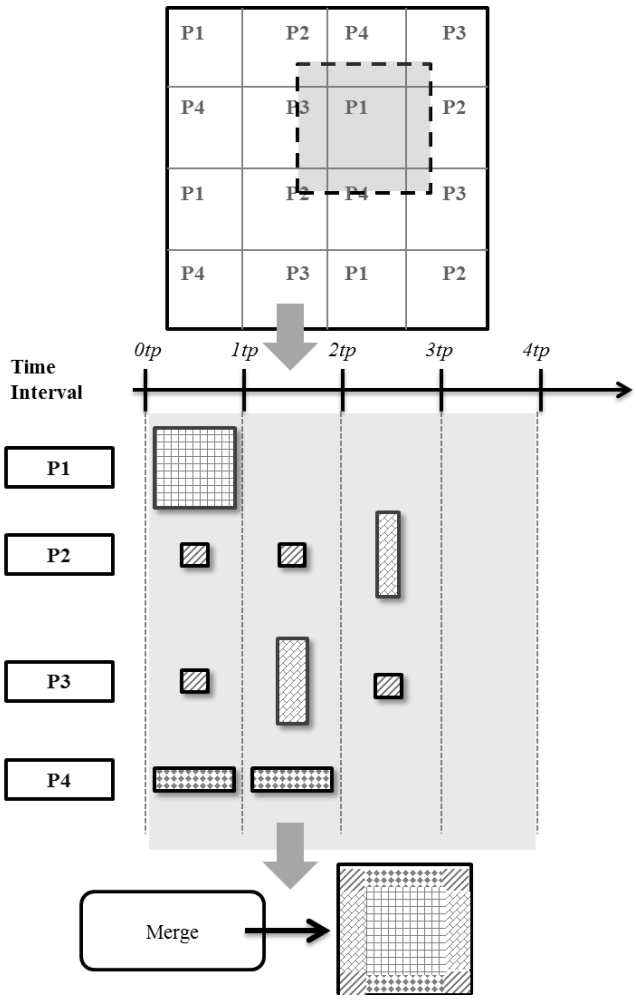


Fig. 3.10 平行解壓縮運算在優化的資料擺放法上

Fig. 3.9 和 Fig. 3.10 為直覺的資料擺放法的例子，資料擺放法自從 1979 年已有許多的發表[29]，現今已有許多研究探討資料擺放法。啟發式擺放法主要可以分為兩個種類；靜態擺放法和動態擺放法[30]，大部分的靜態擺放法需要有完整的資訊，而動態擺放法可以根據工作負擔進行調整，在其他研究中也提出基本的擺放法[29, 31, 32]，最近已有許多的演算法應用在不同的平台和硬體[33-35]，這些擺放演算法都可以應用在我們的 VLLC 資料查詢中。

第4章 模擬結果

4.1 類視訊無失真壓縮

在這次的實驗裡，我們的資料伺服器使用單核心的平台，而測試資料是紀錄 2003 到 2006 的每日大氣中的二氧化碳濃度，而這份測試資料有著高度的空間相關性，非常的適用於我們的系統，而資料幀的每個時間區段為 24 小時。

在 Table 4.1 中，我們比較 VLLC 和其他著名的壓縮法的效能，選出最有名和最常見的幾種壓縮方法，WinRAR 和 7-Zip，並且分別比較資料節省空間和進行資料壓縮所花費的時間。在這次的實驗中，原始的感測資料大小為 4544.3MB，如 Table 4.1 所示，7-Zip 壓縮等級調整為最高時的資料壓縮率為 9.18%，而 WinRAR 的資料率為 18.29%，而我們提出的 VLLC 的資料壓縮率為 7.18%，在資料壓縮方面的比較，7-Zip 的壓縮時間高達 VLLC 的壓縮時間的 27 倍，而 VLLC 比 7-Zip 省下更多的資料儲存空間。

Table 4.1 壓縮方法效能比較

	COMPRESSION METHODS			
	7-Zip*	7-Zip	WinRAR	VLLC
Sensory Data Size	4544.3 MB			
Database Size	417.2 MB	442.6 MB	831.0 MB	226.1 MB
Space Savings	90.82%	90.20%	81.71%	92.82%
Compression Time	1156 (s)	776 (s)	124 (s)	42.8 (s)
	* Compression level: Maximum			

接下來的實驗中，比較的是解壓縮的時間，從 Fig. 4.1 可看出，雖然 7-Zip 在總解壓縮時間上贏過 VLLC，但 VLLC 有隨機存取上的優勢。接著 Fig. 4.2 是要比較 VLLC 與 7-Zip 存取時間的比較，由於 7-Zip 分為結實壓縮與非結實壓縮，結實壓縮是將要壓縮的檔案當作同一個資料流，而非結實壓縮中，檔案會被看成各自的資料流，為了實驗的完整性，在存取時間的比較上，VLLC 將與 7-Zip 的結實壓縮與非結實壓縮分別做比較。

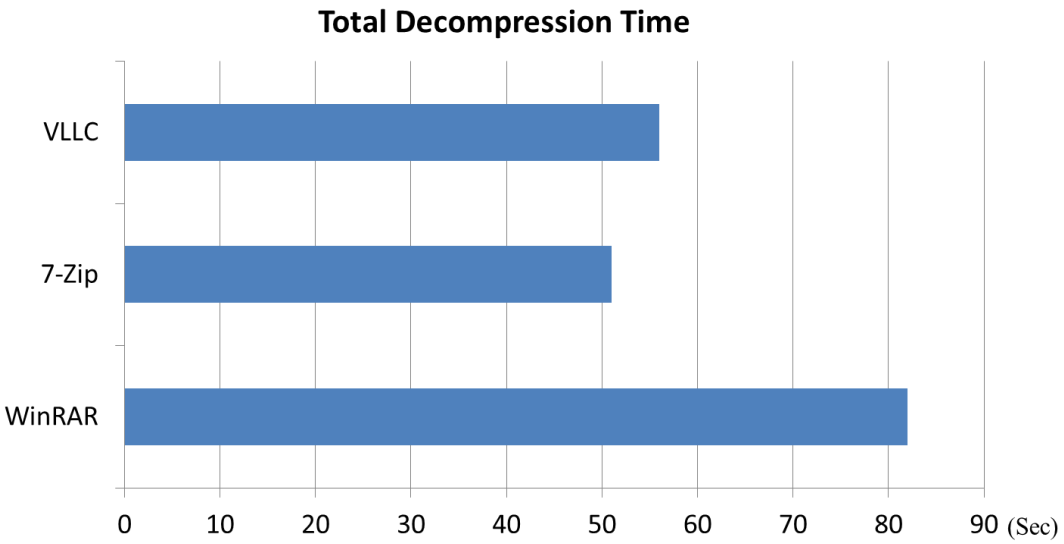


Fig. 4.1 總解壓縮時間比較

在 Fig. 4.2 與 Fig. 4.3 中，7-Zip 壓縮分別為非結實與結實做了不同索引(index)數目之比較，由於檔案的資料幀數目最多為 1461，因此我們將實驗的索引數設定為 2 到 1461 個索引分別比較，從 Fig. 4.2 與 Fig. 4.3 中可看出，由於非結實壓縮將不同的檔案視為不同的資料流，所以非結實壓縮解壓縮的資料量越小，解壓縮的速度越快，而結實壓縮適合解壓縮大筆資料量。若是 7-Zip 增加索引數目，則無論是結實壓縮或是非結實壓縮，皆可觀察出整體的解壓縮時間皆提高，原本從 Fig. 4.1 可得知，若 7-Zip 只有一個索引，總解壓縮時間可低於 VLLC，若 7-Zip 增加索引數目，則總解壓縮時間將高於 7-Zip。

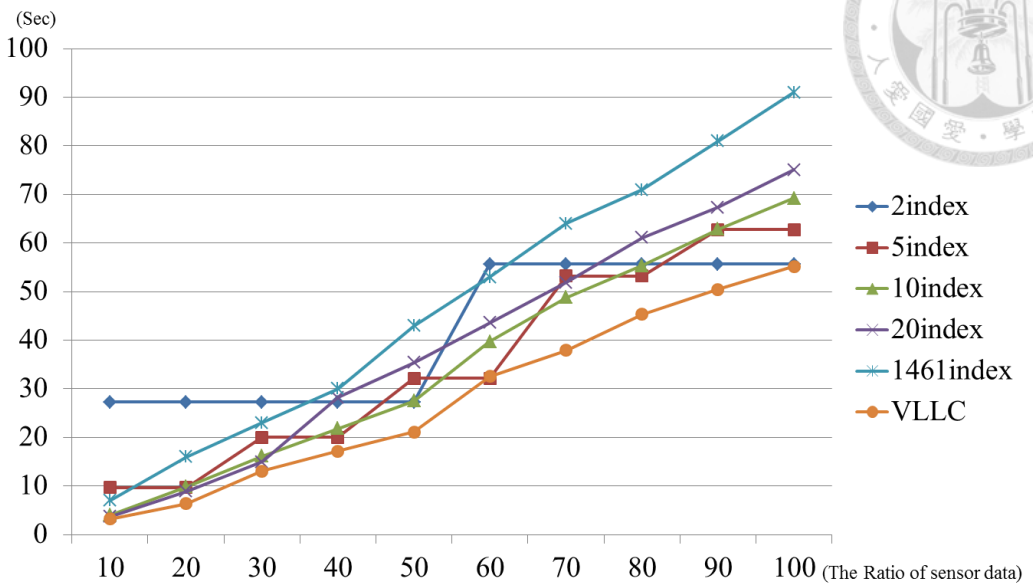


Fig. 4.2 從檔案起點開始往後解壓縮時間比較 (非結實)

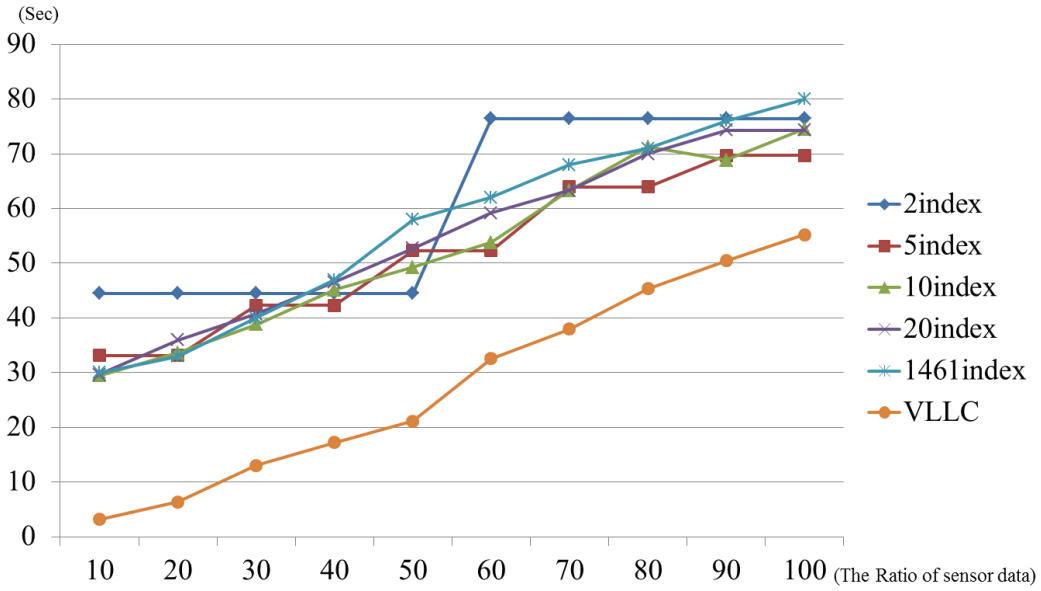


Fig. 4.3 從檔案起點開始往後解壓縮時間比較 (結實)

而在隨機存取時間比較中，實驗設定為從檔案中的不同起點往後固定解壓縮10%的資料量，Fig. 4.5 可觀察出，7-Zip 的結實壓縮並不適用於隨機存取。而在Fig. 4.4 中可看出，索引數目 2 與索引數目 5 明顯高於其他幾組，原因是索引數目 2 與索引數目 5 分別要解壓縮 50%與 20%的資料量，所以解壓縮時間明顯多於其他

幾組。雖然索引數目 10、索引數目 20 與 VLLC 之隨機存取時間無明顯差距，但是索引數目若增加至 100 後，可觀察出隨機存取時間明顯提升，一旦索引數目增加至 1461 時，7-Zip 存取時間明顯高於 VLLC。從 Fig. 4.7 中可看出 7-Zip 平均隨機存取時間與索引數的關係，因此得知當提高 7-Zip 的索引數目，將會降低 7-Zip 隨機存取的效能。根據以上的實驗可得知，VLLC 在壓縮時間與壓縮率均贏過 7-Zip 與 WinRAR，並且在資料的隨機存取能力遠勝 7-Zip，因此在實務上，VLLC 的實用性絕對是大於 7-Zip 與 WinRAR。

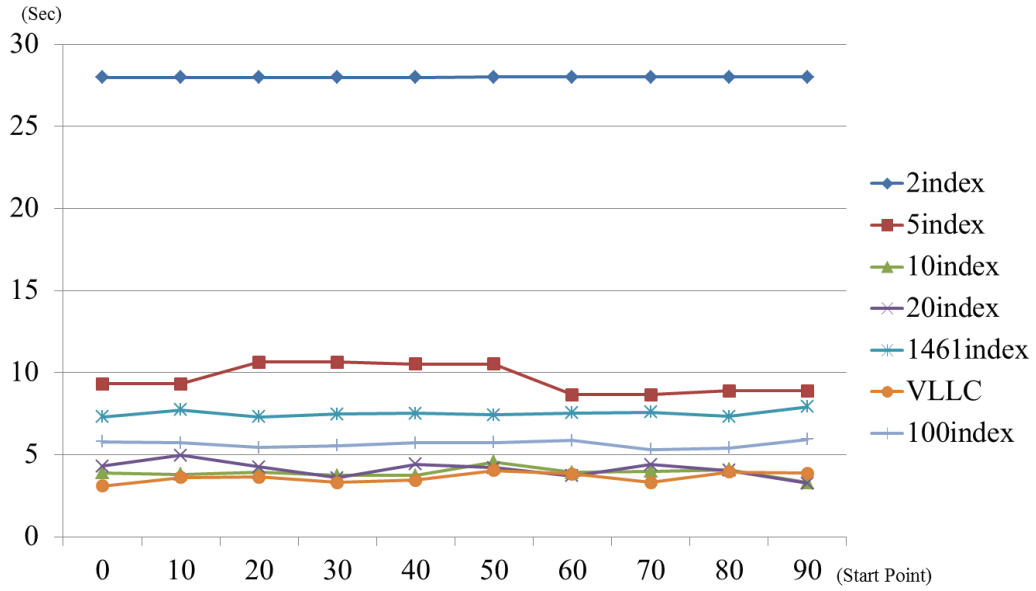


Fig. 4.4 從檔案不同點開始往後解壓縮 10%資料時間比較 (非結實)

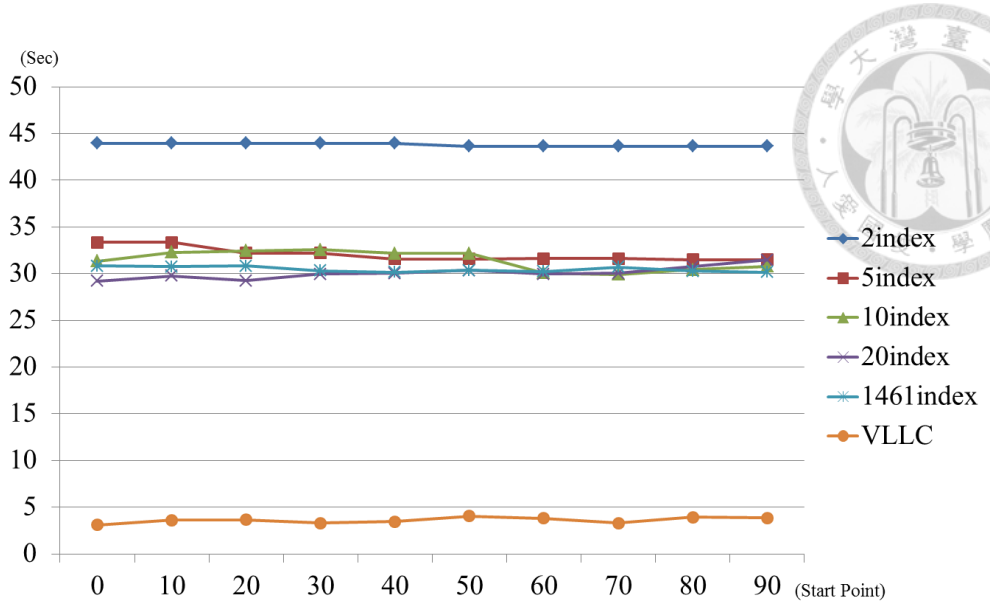


Fig. 4.5 從檔案不同點開始往後解壓縮 10%資料時間比較 (結實)

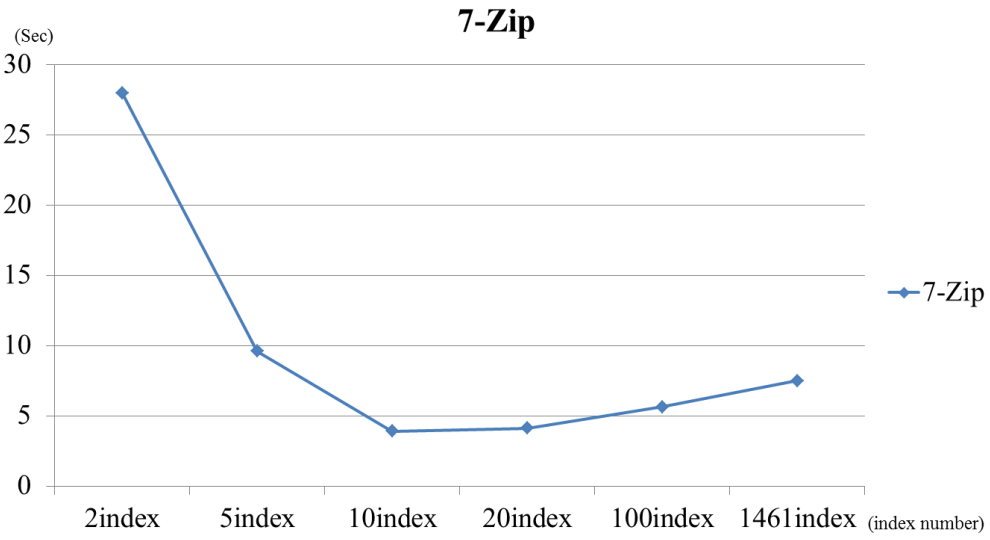


Fig. 4.6 7-Zip 不同所引數之平均隨機存取時間 (非結實)

如我們之前所提到的，我們提出了平行化的資料壓縮與解壓縮的功能，因此在 Table 4.2 我們將測試資料進行平行化的處理，而測試資料與 Table 4.1 的相同，我們將資料幀的解析度大小作調整，第一組數據為原始的大小，第二組數據是調整為原始比例的二分之一，由於是將原始的資料幀調整原始比例的二分之一，所以資料幀數目是原本的四倍，第三組數據則是調整為原始比例的三分之一，而實驗的結果如 Table 4.2 所示。由 Table 4.2 可看出，若是我們將資料幀的解析度縮小，增加資料幀的數目，無失真壓縮後，整體資料量將會越來越大。

Table 4.3 是整理出取單張資料幀所需的解壓縮時間，以及解壓縮所有資料的解壓縮時間。雖然將分工的數目增加，整體的資料量大小會越來越高，但是由 Table 4.3 可看出，第一組未進行平行處理分工的解壓縮時間需 42.801 秒，而第二組的解壓縮時間最大值為 18.217 秒，第三組的解壓縮時間最大值為 7.951 秒，由此可以觀察出來，分工數目增加，雖然無失真壓縮後的感測資料量會因此提升，但解壓縮時間卻可以降低，在這一方面可以提高資料查詢的效能。同時 Table 4.2 和 Table 4.3 兩張表格能從中觀察出一個趨勢，若壓縮率較低則解壓縮時間就會較多，以兩個表中的 Voxel 2 為例，Voxel 2 壓縮的比較差，解壓縮的時間也會因此提高，在此可看出資料相關性的重要性，如果我們能在資料擺放的方法上提高資料的相關性，對於提高資料庫的效能將有極大的幫助。

Table 4.2 感測資料在不同尺寸的資料幀進行壓縮比較

	FRAME SIZE (M BY N)		
	1920x1080	960x540	640x360
Number of Voxels	1	4	9
Total Frame Number	1461	5844	13149
Size Before Compression	4544.3 MB		
Voxel 1	220.8 MB	96.0 MB	42.2 MB
Voxel 2		75.9 MB	65.4 MB
Voxel 3		75.3 MB	50.4 MB
Voxel 4		70.7 MB	44.8 MB
Voxel 5			40.5 MB
Voxel 6			41.2 MB
Voxel 7			37.3 MB
Voxel 8			33.7 MB
Voxel 9			26.5 MB
Size After Compression	220.8 MB	311.1 MB	383.0 MB

Table 4.3 感測資料在不同尺寸的資料幀進行解壓縮比較

	FRAME SIZE (M BY N)		
	1920x1080	960x540	640*360
	Single Frame Decompression Time		
Voxel 1	0.307 s	0.115 s	0.095 s
Voxel 2		0.092 s	0.102 s
Voxel 3		0.112 s	0.099 s
Voxel 4		0.107 s	0.097 s
Voxel 5			0.099 s
Voxel 6			0.090 s
Voxel 7			0.085 s
Voxel 8			0.084 s
Voxel 9			0.080 s
	Total Decompression Time		
Voxel 1	42.801 s	18.217 s	5.570 s
Voxel 2		14.645 s	7.951 s
Voxel 3		15.261 s	6.135 s
Voxel 4		14.701 s	6.623 s
Voxel 5			5.468 s
Voxel 6			5.611 s
Voxel 7			4.931 s
Voxel 8			4.428 s
Voxel 9			3.972 s

4.2 資料擺放法

如之前所提，資料擺放法將會影響平行處理的效能，在資料擺放法實驗裡，我們則是要探討資料擺放法，以 Fig. 4.7 為例，整個系統平台有無數個 Voxel，系統有 9 台個人計算機，當使用者進行了 3x3 大小的範圍查詢，由 Fig. 4.7 可看出 P1 分配到 2TP、P2 分配到 2TP、P3 分配到 1TP、P4 分配到 3TP、P6 分配到 1TP、此例中範圍查詢所需的解壓縮單位時間為 3TP，理想情況下，最佳解壓縮所需的單位時間為 1TP。

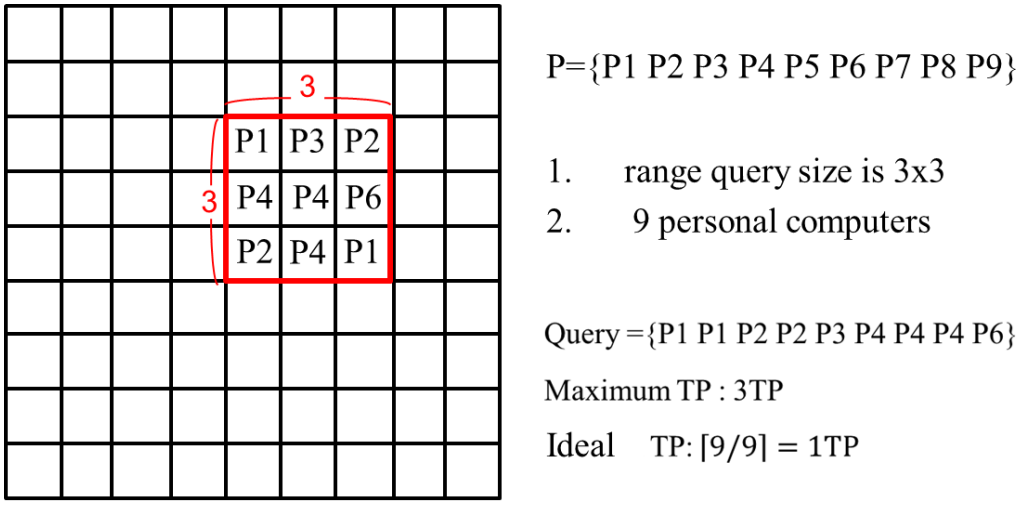


Fig. 4.7 範圍查詢範例

接下來我們將探討如何選出理想資料擺放法，實驗的設計為無數個 Voxel 和數台個人計算機下，範圍查詢尺寸由 2x2 到 11x11 的大小，而資料擺放法有 Random、Zigzag、Reverse、Vortex 四種不同的擺放方式，而 Ideal 這條虛線則為理論下，利用鴿籠原理(the pigeonhole principle)計算，只進行一次範圍查詢的最佳解壓縮單位時間(並非實際下的最佳單位時間)。如 Fig. 4.8 所示，Y 軸代表範圍查詢所需的平均解壓縮單位時間(TP)，X 軸代表範圍查詢的尺寸大小。

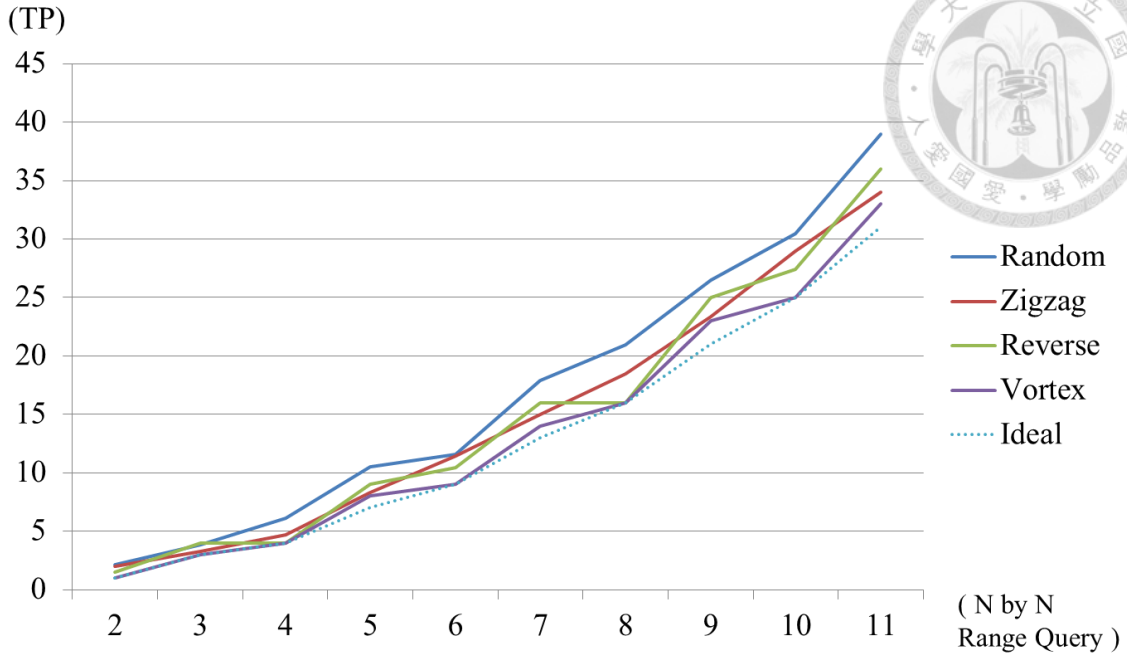


Fig. 4.8 資料擺放法在 4 台個人計算機情況下

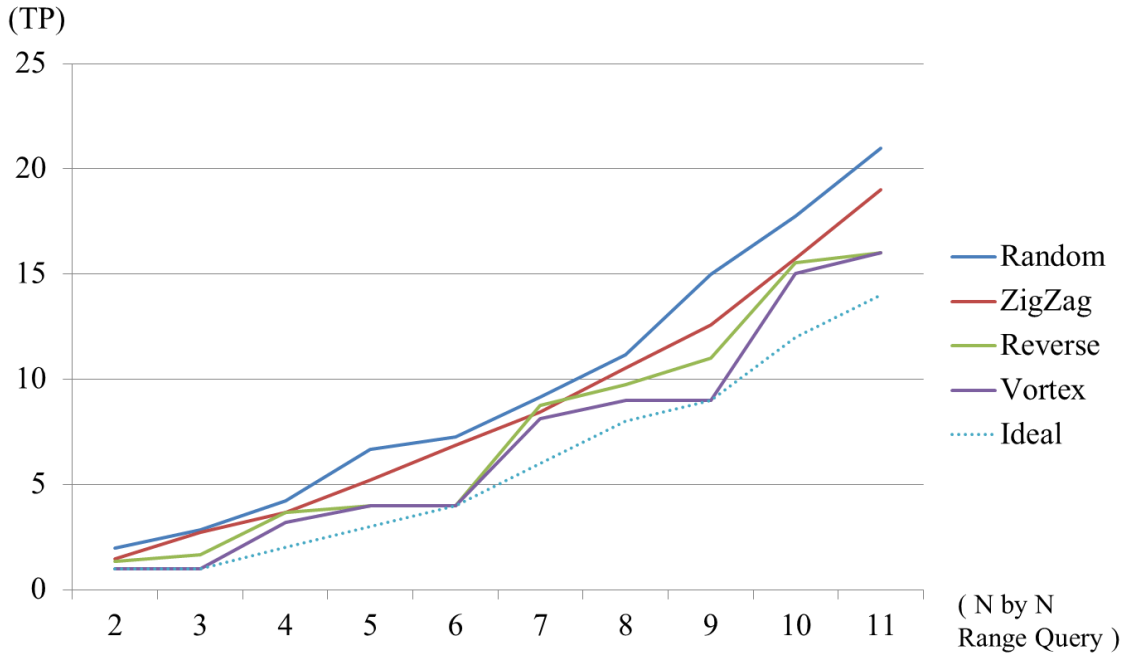


Fig. 4.9 資料擺放法在 9 台個人計算機情況下

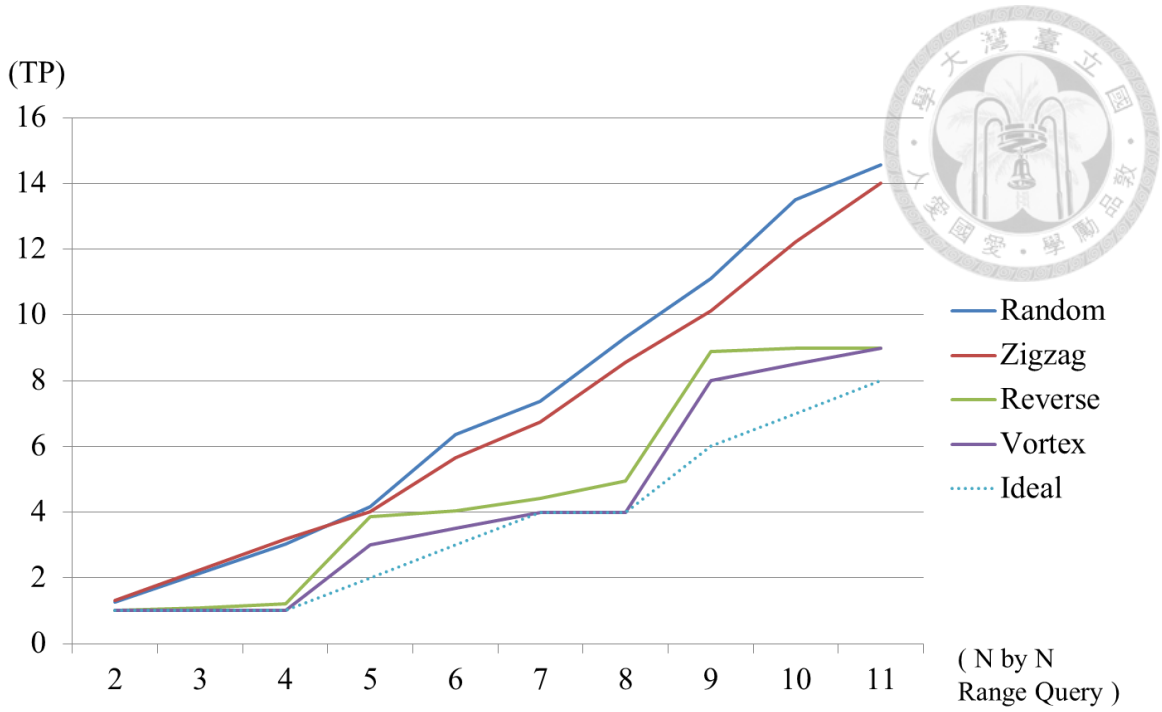
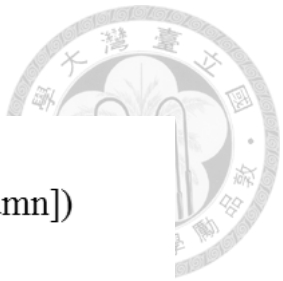


Fig. 4.10 資料擺放法在 16 台個人計算機情況下

從 Fig. 4.8、Fig. 4.9 和 Fig. 4.10 中可看得出來，最差的方法是 Random 擺放，Random 擺放是完全無規則隨意地分配個人計算機，因此可得知合理有效的個人資料擺放法是必須的，由於 Fig. 4.88 的個人計算機數目只有 4 個，不同擺放法的差異並不明顯，因此我們還做了 9 台個人計算機資料擺放法以及 16 台個人計算機資料擺放法的實驗。以之前所做的實驗 Table 4.3 為例，資料幀的尺寸為 1920x1080 來說，單一資料幀解壓縮時間約為 0.307 秒，這 0.307 秒即為一個 TP，若在 16 台計算機分工的情況下，範圍查詢的大小為 11x11，Random 擺放需花費 14.57 個 TP 也就是 4.473 秒與 Vortex 擺放需要 9 個 TP 也就是 2.763 秒，兩種擺放法相差 5.570 個 TP，若能選擇合適的擺放法，每做一次範圍查詢能省下 62%的時間。

隨著個人計算機數目的增加，不同擺放法的效果差異也越來越明顯，Random 擺放仍舊是最差的擺放法，最佳的擺放法為 Vortex 擺放，接下來為 Reverse 擺放、Zigzag 擺放，接著介紹這些擺放法，並且解釋這些不同演算法為何有如此的差異。



```
1.  FUNCTION Zig-Zag(row,  column)
2.  Treat all the voxels as 2d arrays (Voxel[row][column])
3.      LOOP Place Process Until the Plane is full
4.          Set Variable i=1
5.          IF i is odd THEN
6.              While Position Not on the Boundary
7.                  V[row][column] = Process
8.                  Change Current Process to Next Process
9.                  Move From Lower Left To Upper Right
10.             END While
11.             Variable i PLUS 1
12.         ELSE IF i is even THEN
13.             While Position Not on the Boundary
14.                 V[row][column] = Process
15.                 Change Current Process to Next Process
16.                 Move From Upper Right To Lower Left
17.             END While
18.             Variable i PLUS 1
19.         ENDIF
20.     END LOOP
21. END FUNCTION
```

Fig. 4.11 Zigzag 擺放虛擬碼

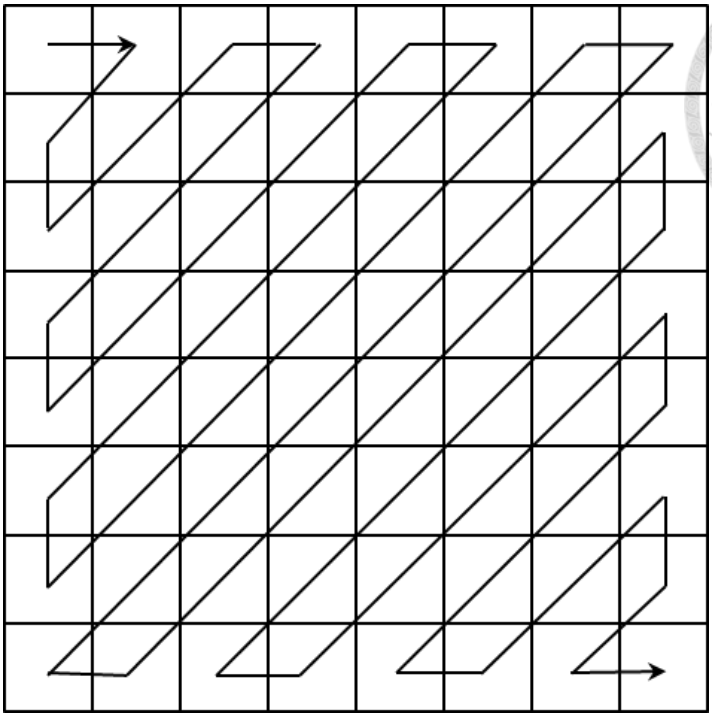


Fig. 4.12 Zigzag 擺放示意圖

Zig-Zag 為 JPEG 無失真資料壓縮上的特別形式，它的特色是以 Z 字形進行掃描。Zig-Zag 將整個平面先由左下往右上掃描，接著在以右上向左下掃描，直到填滿整個平面，而在掃描的同時按照序列分配給個人計算機。

由 Fig. 4.12 我們可以觀察出，Zigzag 擺放並沒有特別針對範圍查詢將個人計算機平均分配，因此效果並沒有特別好，整體時間也只不過比 Random 擺放來的好，所以好的擺範方法要針對範圍查詢的特性，最好的兩個擺放方法 Reverse 和 Vortex 都是針對範圍查詢的特性將個人計算機進行平均分配。

```
1.    Function  Reverse (row, column)
2.        Set Variable a*n = row
3.        Set Variable b*m=column,
4.        Set Variable axb=number of  process // a, n, b, m are integer
5.        Divide the Plane to m*n 2d arrays SubPart[m][n], the size of each sub-part is axb
6.        Treat all the voxels as 2d arrays (Voxel[a][b])
7.         LOOP-1 SET Variable i=1 -> m
8.               LOOP-2 SET Variable j=1 -> n
9.               Move to SubPart[i][j]
10.                   LOOP-3 SET Variable p=1 ->a
11.                       LOOP-4 SET Variable q=1 ->b
12.                       Voxel[p][q]= Process
13.                       Change Current Process to Next Process
14.                       Variable q PLUS 1
15.                       END LOOP-4
16.                   Variable p PLUS 1
17.                   END LOOP-3
18.                Reverse the Order of The Processes
19.              Variable j PLUS 1
20.              END LOOP-2
21.           Variable i PLUS 1
22.           END LOOP-1
23.   END FUNCTION
```

Fig. 4.13 Reverse 擺放虛擬碼

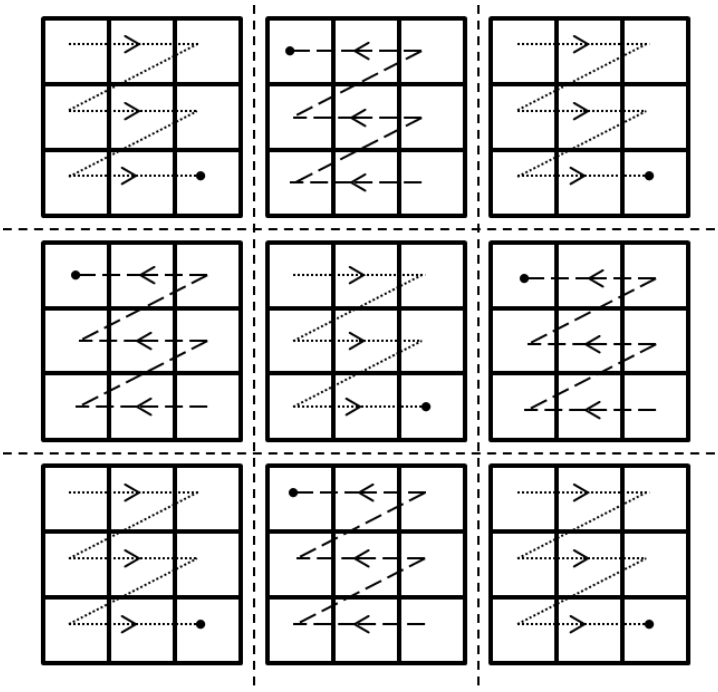


Fig. 4.14 Reverse 擺放示意圖

Reverse 擺放則是有針對範圍查詢的特性做設計，由 Fig. 4.14 所示 Reverse 擺放將整個影像平面按照個人計算機的數目做切割，在 Fig. 4.13 的虛擬碼可得知整個影像平面的長和寬是子平面的倍數，而子平面上的個人計算機個數即為可用個人計算機的總數，因此影像平面可被數台個人計算機平均分配，Reverse 擺放在個人資料擺放法上的特性是將個人計算機先由上到下並由左到右按照順序擺放填滿子平面，接著再相反順序由上到下並由左到右填滿下一個子平面。填滿整個影像平面的方式則是一樣由上到下並由左到右直到填滿整個平面，由於有根據範圍查詢的特性設計擺放，所以 Reverse 擺放為此次實驗次好的資料擺放法。

```
1.  Function  Vortex (row, column)
2.      Set Variable a*n = row
3.      Set Variable b*m=column,
4.      Set Variable axb=number of  process // a, n, b, m are integer
5.      Divide the Plane to m*n 2d arrays SubPart[m][n], the size of each sub-part is axb
6.      Treat all the voxels as 2d arrays (Voxel[a][b])
7.       LOOP-1 SET Variable j=1 -> m
8.           LOOP-2 SET Variable i=1 -> n
9.           Move to SubPart[i][j]
10.          IF Variable j is  odd THEN
11.              Place Processes in the sub-part in a clockwise direction
12.          ELSE IF Variable j is even THEN
13.              Place Processes in the sub-part in a counterclockwise direction
14.          ENDIF
15.          Variable i PLUS 1
16.          END LOOP-2
17.      Variable j PLUS 1
18.      END LOOP-1
19.  END FUNCTION
```

Fig. 4.15 Vortex 擺放虛擬瑪

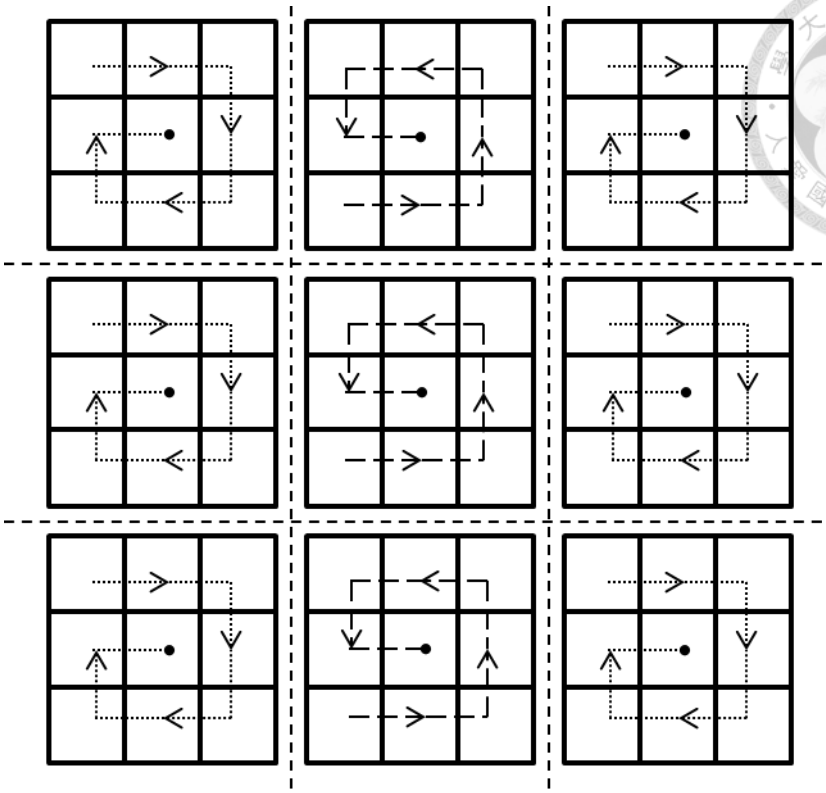


Fig. 4.16 Vortex 擺放示意圖

Vortex 擺放跟 Reverse 擺放一樣是將影像平面切割成許多子平面，主要差別則是 Vortex 在子平面的擺放法，Vortex 擺放將擺放方式的分為順時針擺放逆時針擺放，第一行的子平面進行順時針擺放，下一行的子平面進行逆時針擺放。除此之外，在 Fig. 4.8、Fig. 4.9、可 Fig. 4.10 以觀察出，Vortex 擺放在與子平面大小相同的範圍查詢上均達到單一查詢的最佳時間，也就是與子平面大小相同的範圍查詢的最佳解。所以當本系統使用者使用範圍查詢時，系統能提供良好的資料擺放法，提高資料查詢的效能。

第5章　　結論與未來研究

本論文提出了一個應用在無線感測網路的類視訊無失真壓縮法，幫助無線感測網路資料伺服器處理大量的感測資料，類視訊無失真壓縮法可套用在現有的資料庫，能夠幫助分散式資料庫處理感測資料，提高資料庫的效能。

在此壓縮法中，感測資料先依照蒐集到的時間分類，同一時間點的感測資料放在同一張資料幀上，資料幀上畫素擺放的位置是根據空間相關性所決定的，這些資訊通通會存放在 Mapping Table 上，而資料幀組合起來稱之為一個平面，連續的子平面我們稱之為立體像素，而連續的影片上的同一位置的畫素存在著時間相關性，由於立體像素可視為一個影片，因此我們可以藉由無失真視訊壓縮來降低感測資料的大小，提高感測網路整體存活時間。我們的系統也提供了感測資料的查詢，單一資料查詢、範圍資料查詢，並且在範圍資料查詢提供平行化處理來提高資料查詢的效率。在我們的實驗中指出資料節省空間與資料查詢時間之間的關係，使用者可以權衡兩者之間的效益來調整立體像素的尺寸大小。而在資料壓縮效能的實驗中，我們可以看到無失真視訊壓縮在壓縮感測資料的效能優於知名的壓縮方法 7-Zip 和 WinRAR，無失真視訊壓縮的資料節省空間贏過 7-Zip 和 WinRAR，並且 7-Zip 的壓縮時間是無失真視訊壓縮時間的 12 倍，而 WinRAR 是無失真視訊壓縮的 3 倍。資料庫系統提供感測資料範圍查詢，並支援平行處理來提高範圍查詢的效能。在實驗裡我們探討如何決定良好的資料擺放法，使用者必須根據不同的資料來決定資料擺放法，若使用者沒有採取適當的資料擺放法，當分工的個人計算機數目一旦增多，良好的擺放法與不良的擺放之間效能的差異會越明顯。

在本文中，我們提出了無失真視訊壓縮感測網路資料庫的基本功能與應用，無失真視訊壓縮與感測資料查詢等功能，若需要提高資料庫系統整體的性能與實用性，無線感測網路資料庫必須增加更多必需的功能，像是模糊查詢(fuzzy search)、查詢最佳化、以及感測資料缺值的問題探討，以上這些方法都是我們未來所會考慮增加到我們的資料庫系統中，並且這些方法並有助於提高感測網路資料庫的實用性與效能。

參考文獻

[1] C.F. García-Hernández, P.H. Ibargüengoytia-González, J. García-Hernández, and J.A. Pérez-Díaz, "Wireless Sensor Networks and applications: a survey," International Journal of Computer Science and Network Security, vol. 7, pp. 264-273, March 2007.

[2] D. Culler, D. Estrin, and M. Srivastava, "Guest Editors' Introduction: Overview of Sensor Networks," *Computer,* vol. 37, pp. 41-49, 2004.

[3] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson and D. Culler, "An Analysis of a Large Scale Habitat Monitoring Application," Embedded Networked Sensor Systems, November 2004.

[4] B. Zhou, C. Hu, H.B. Wang, R. Guo and Q.H. Meng, "A Wireless Sensor Network for Pervasive Medical Supervision," International Conference on Integration Technology, March 2007.

[5] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," Communications Magazine, IEEE, vol. 40, pp. 102-114, 2002.

[6] Y. Liang and W. Peng, "Minimizing energy consumptions in wireless sensor networks via two-modal transmission," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 12-18, 2010.

[7] V. Singla, R. Singla and S. Gupta, "Data compression modeling : Huffman and Arithmetic," International Journal of The Computer, vol. 16, pp. 64-48, December 2008.

[8] K.C. Barr and K. Asanovic, "Energy-aware Lossless Data Compression," Transactions on Computer Systems, vol. 24, pp. 250-291, August 2006.

[9] Z. Zhang and O. Berger, "Cluster based data query analysis and optimization for Wireless Sensor Networks," Advanced Communication Technology, February 2008.

[10] S. Zhou, Y. Lin, J. Wang, J. Zhang and J. Ouyang, "Compressing Spatial and Temporal Correlated Data in Wireless Sensor Networks Based on Ring Topology," Lecture Notes in Computer Science, vol. 4016, pp. 337-348, October 2006.

[11] J.M. Miranda, E. Reynaud, F. McGlone, G. Calvert and M. Brammera, "The impact of temporal compression and space selection on SVM analysis of single-subject and multi-subject fMRI data," NeuroImage, vol. 33, pp. 1055-1056, December 2006.

[12] *7-Zip*. Available: http://www.7-zip.org/

[13] *WinRAR*. Available: http://www.rarlab.com/

[14] R. Asraf, M. Akbar and N. Jafri, "Statistical analysis of difference image for absolutely Lossless compression of medical images," Engineering in Medicine and Biology Society, September 2006.

[15] D. S. Taubman and M. W. Marcellin, "*JPEG2000: Fundamentals, Standards, and Practice,*" Published by Kluwer Academic Publishers, 2002.

[16] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *Image Processing, IEEE Transactions on,* vol. 9, pp. 1309-1324, 2000.

[17] R. Starosolski, "Simple fast and adaptive lossless image compression algorithm," *Software: Practice and Experience,* vol. 37, pp. 65-91, 2007.

[18] F. Marcelloni and M. Vecchio, "An Efficient Lossless Compression Algorithm for Tiny Nodes of Monitoring Wireless Sensor Networks," *Comput. J.*, vol. 52, pp. 969-987, 2009.
[19] C. Tharini and P. V. Ranjan, "Design of modified adaptive Huffman data compression algorithm for wireless sensor network," *Journal of Computer Science*, vol. 5, pp. 466-470, 2009.
[20] L.D. Chagas, E.P. Lima and P.F.R. Neto, "Real-Time Databases Techniques in Wireless Sensor Networks," International Conference on Networking and Services, March 2010.
[21] *Oracle*. Available: http://www.oracle.com/tw/index.html
[22] *SQL Server*. Available: http://www.microsoft.com/taiwan/sql/default.mspx
[23] *MySQL: The world's most popular open source database*. Available: http://www.mysql.com
[24] *TINYLIME: LIME FOR SENSOR NETWORKS*. Available: http://lime.sourceforge.net/tinyLime/index.html
[25] *TinyDB: A Declarative Database for Sensor Networks*. Available: http://telegraph.cs.berkeley.edu/tinydb/
[26] *COUGAR: The Network Is The Database*. Available: http://www.cs.cornell.edu/bigreddata/cougar/index.php
[27] M.H. Li, C.C. Lin, C.C. Chuang and R.I. Chang, "Error-bounded data compression using data, temporal and spatial correlations in Wireless Sensor Networks," Multimedia Information Networking and Security, November 2010.
[28] C.C. Lin, C.C. Chuang, C.W. Chiang and R.I. Chang, "A Novel Data Compression Method using Improved JPEG-LS in Wireless Sensor Networks," International Conference on Advanced Communication Technology, February 2010.
[29] J. Valdes, R.E. Tarjan and E.L. Lawler, "The recognition of Series Parallel digraphs," ACM symposium on Theory of computing, 1979.
[30] D.K. Madathil, R.B. Thota, P. Paul and T. Xie, "A Static Data Placement Strategy towards Perfect Load-Balancing for Distributed Storage Clusters," International Symposium on Parallel and Distributed Processing, April 2008.
[31] C.M. Wang and S.D. Wang, "Efficient processor assignment algorithms and loop transformations for executing nested parallel loops on multiprocessors," IEEE Transactions on Parallel and Distributed Systems, vol. 3, pp. 71-82, January 1992.
[32] A.N. Choudhary, B. Narahari, D.M. Nicol and R. Simha, "Optimal Processor Assignment for a Class of Pipelined Computations," IEEE Transactions on Parallel and Distributed Systems, vol. 5, pp. 439-445, April 1994.
[33] H. Back, H.S. Chwa and I. Shin, "Schedulability Analysis and Priority Assignment for Global Job-Level Fixed-Priority Multiprocessor Scheduling," IEEE Symposium on Real-Time and Embedded Technology and Applications, April 2012.
[34] Ch. Xu, X. Chen, R.P. Dick and Z.M. Mao, "Cache Contention and Application Performance Prediction for Multi-Core Systems," IEEE International Symposium on Performance Analysis of Systems & Software, March 2010.
[35] D.Q. Ren and R. Suda, "Investigation on the Power Efficiency of Multi-core

and GPU Processing Element in Large Scale SIMD Computation with CUDA," Green Computing Conference, August 2010.