行政院國家科學委員會補助專題研究計畫 ☑成果報告
□期中進度報告

基於圖形處理器之鉅量資料分析系統

Large-Scale Data Processing System on GPU

計畫類別：☑個別型計畫　□整合型計畫
計畫編號：NSC 98-2221-E-009 -074 -MY2
執行期間：98 年 8 月 1 日至 100 年 7 月 31 日

執行機構及系所：國立交通大學資訊工程系

計畫主持人：袁賢銘
共同主持人：
計畫參與人員：宋牧奇、江川彥、邱俊傑、黃聖凱

成果報告類型(依經費核定清單規定繳交)：□精簡報告　☑完整報告

本計畫除繳交成果報告外，另須繳交以下出國心得報告：
□赴國外出差或研習心得報告
□赴大陸地區出差或研習心得報告
□出席國際學術會議心得報告
□國際合作研究計畫國外研究報告

處理方式：除列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

中　華　民　國　100 年 9 月 27 日

第一章 計畫摘要

處理大量資料的能力是在許多領域裡相當基本的。從資料探勘、自然語言分析、地震資料分析、人工智慧到計算物理、化學、生物科學，都需要這樣的能力。然而到目前為止，這樣的分析與模擬都僅限於在一些高效能計算中心中，才有可能被有效率的實現；這些計算中心使用了大量的分散式系統，運用數百台機器及高速的網路設備串接，獲得較高的資料處理能力。然而，這種系統的建置與維護的高成本使得這種運算的資源沒辦法讓每一個研究學者或科學家頻繁的使用，迫使他們得在自己的低效能的機器或叢集上進行研究與資料的分析，甚至花費幾天到幾個星期在等待上。

這些在計算中心才能被實現的問題通常不是受限於I/O 就是受限於運算能力，所以沒有辦法有效的在個人電腦上單單一顆CPU 搭配傳統的硬碟來執行。然而，因為一些最近個人電腦硬體技術上的進步，本計畫希望透過高效能的繪圖晶片與固態硬碟來解決這個問題，以下我們分別簡介這兩者的演進。

從2003 年開始，在高效能運算的領域裡，運用繪圖晶片（GPU）來加速計算，一直是非常熱門的話題。從早期運用3D 渲染函式庫到現在有標準的平行運算開發框架，繪圖晶片運算的技術可以說是一日千里。許多原本需要耗費數天甚至是數個月的複雜計算，透過繪圖晶片的幫忙，在短短的幾秒到幾分鐘內就產生結果。然而，到目前為止，要實際運用繪圖晶片的運算能力，開發一個能在繪圖晶片上運行的應用並不是那麼容易，不是每一個研究學者都有能力去發展的。

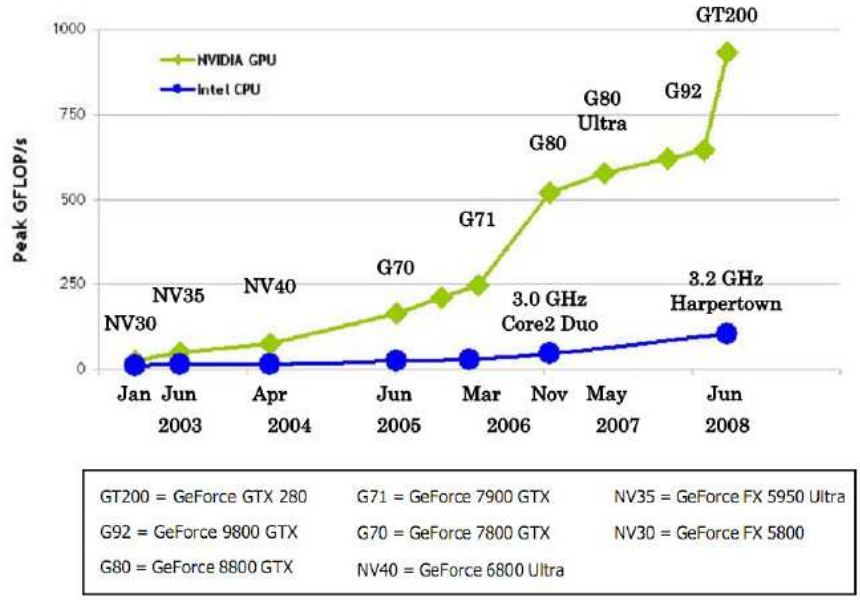
另外，高速的儲存裝置也從2007 年開始萌芽，運用高速的快閃記憶體取代傳統機械式硬碟，稱為固態硬碟（Solid-state Drive，SSD），大幅度的提升I/O 的輸出輸入效率，從以前每秒50~60MB 的讀取速度一舉提升到每秒160~170MB。而在2008 年時，SSD 硬碟價格滑落，開始漸漸的普及到一般消費者，所以在未來幾年內，SSD 硬碟將有機會取代傳統機械式硬碟，成為下一世代個人電腦上儲存裝置的標準。

應因繪圖晶片效能的提升及高速運算的需求，以及高速儲存裝置的興起，所以本計畫希望能運用消費者市場中標準的硬體元件，建置一套低成本的高效能運算叢集，並開發此運算叢集上之軟體系統，提供簡單的且專為資料分析設計之語言介面，方便科學家或各個領域的研究學者，運用繪圖晶片上的運算能力及高速的I/O 系統，在極短的時間內完成他們的資料分析或處理，達到高效能及簡化軟體開發的目標，而不需要了解底層繪圖晶片的架構與高速 I/O 系統的設計。

第二章 研究計畫內容

以下我們先介紹本計畫的幾個背景，分別是視覺運算，高速I/O 裝置，以及大量資料處理。近年來，因為遊戲對繪圖晶片（GPU）性能的要求持續成長，繪圖晶片的運算速度愈來愈快，尤其是在某些特定的領域上已經大幅度的超越一般處理器（CPU）的運算效能，所以從2003 年開始，許多的研究學者，開始將一些非繪圖相關的運算，移植到繪圖晶片上執行，而獲得了非常多倍效能的成長。而在今年全球最大高階繪圖晶片廠商NVIDIA 也舉辦首屆的NVISION 研討會NVISION08，希望透過NVISION，告訴全世界：視覺運算的時代來了！

繪圖晶片與一般處理器的運算效能比較如下所示：



從這張圖中我們可以很明顯的發現繪圖晶片在運算效能上的優勢，以NVIDIA 最新的GeForce GTX280 系列來說，單張顯示卡對於單精度浮點數的運算效能就接近1 Teraflops，是傳統處理器效能的10~20 倍以上。而且從效能成長的曲線看來，繪圖晶片的成長速度是遠超過中央處理器的成長速度的（甚至超越了Moore's Law）。

繪圖晶片的高效能運算，為那些需要非常大量計算的應用，帶來了許多新的契機。從最基本的BLAS（Basic Linear Algebra Subprograms）及FFT（Fast Fourier Transform）的函式庫[2]、數值運算[3]，到影像辨識[4][5]、流體模擬[6]、電流分析[7]；這些繪圖晶片上的應用都一再的證明了繪圖晶片的運算能力是非常泛用的，可以用來處理繪圖之外的運算的。然而，要將軟體移植到繪圖晶片上並不是那麼容易的。除了開發環境的不同之外，運算模型（Computation Model）的不同是很大的原因。

中央處理器提供的是序列化的模型（Sequential Model），意即同一時間只有一個程式計數器（Program Counter），來控制程式的運作，每一次都處理一個指令或一筆資料。當然這個模型對於最近幾年的中央處理器來說，已經不是那麼的精確，因為現在的中央處理器都會搭載兩個到四個核心，可以在同一時間一次執行兩個到四個不同的程式，以增加多工的效率；而在單一核心上都使用了大量的快取以及複雜的猜測執行邏輯，來加快處理的速度；但是在繪圖晶片上則是採用完全不同的概念。繪圖晶片使用了平行運算的模型，以NVIDIA最新的GTX 280 晶片為例，單一晶片上，有240 個核心[8]，而在ATI 的HD 4800系列上，更是高達800個核心[9]。在繪圖晶片上，沒有大量的快取設計，也沒有複雜的猜測執行以及分支預測，只有大量且簡單的運算核心，這就是為什麼繪圖晶片能提供比中央處理器快上10 到20 倍的運算效能。這樣平行運算的架構，帶來了許多的好處。相較於傳統的序列化執行的模型，要提升效能只能依賴處理器時脈的提高或是更大量的快取，在平行運算的架構下，效能的提升變的是一件十分容易的事，只要增加核心的數目，運算能力就能等比例的提升，但是同時他也帶來了一些困難。因為運算模型的不同，平行化的架構迫使程式設計師要開始思考如何將原本一個程式或演算法，拆開成數百個或是數千個，甚至是上萬個

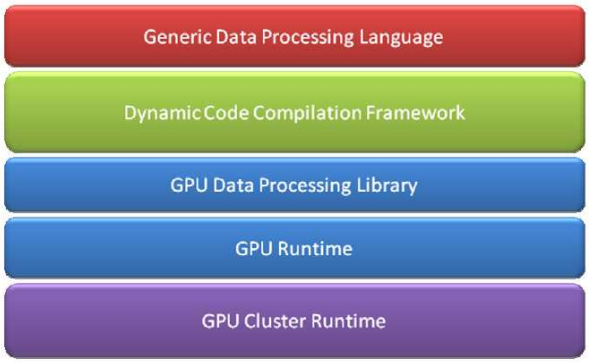
執行緒，然後讓執行緒之間彼此互相合作完成任務。這與一般現在在講「多執行緒」是完全不同的。

所以從2003 年開始，研究學者開始在繪圖晶片上實作一些平行的計算，運用3D 繪圖的函式庫（因為這是當時唯一能跟繪圖晶片溝通的管道），如OpenGL 或DirectX，將一個演算法轉換為3D 渲染的問題，然後搭配GLSL或HLSL 這樣的語言，在頂點著色器（Vertex Shader）及像素著色器（PixelShader）上執行，最後將結果寫入貼圖，再利用OpenGL 或DirectX 的API 下載回來。雖然這樣能夠達到一定的效能提升，但是開發上卻十分不直覺，而且因為是架構於3D 渲染的計算模型之上，所以有許多設計上的限制，因此當時並沒有被廣泛的採用。
但是在2006 年時，隨著軟硬體的技術進步，運用GPU 來進行大量運算開始變的可行。當時，全世界兩大顯示晶片製造商NVIDIA 及ATI 都慢慢的意識到，利用GPU 進行運算的市場是非常可觀，所以他們不約而同的，將他們下一世代的GPU 晶片，設計的更為一般性，讓GPU 不再只是一個專為3D 渲染設計的硬體，而是個非常一般性的高效能運算單元。接著，搭配上軟體開發工具，讓開發者能用他們較為熟悉的語言，來撰寫GPU 上的程式。所以在當時，NVIDIA 推出了CUDA[10]，ATI 則推出了CTM 來因應GPU 上開發的需求[1]；以NVIDIA 的CUDA 為例，開發者可以用他們熟悉的C 語言，來撰寫在GPU 上執行的應用程式，加速了整個GPU 運算軟體開發的效率。ATI 在2007 年及AMD
合併後，AMD/ATI 雖然捨棄了CTM，但也接著推出了下一版的GPU 框架，稱為AMD Stream[12]，同樣也是提供了類似C 語言的開發環境，稱為Brook+[13]，讓開發者能更容易上手，運用AMD/ATI 的GPU 晶片上進行運算。

所以本計畫希望結合這些高速的I/O 裝置以及繪圖晶片的運算能力，設計一個高效能低成本的資料處理叢集系統，同時提供友善的使用者介面，使用者不用了解背後複雜的系統設計，也不需要了解如何在繪圖晶片上撰寫程式，更不需要自行管理這些可觀的計算及儲存資源，只要使用簡單專為資料處理設計的語言，給定相對應的資料，系統就能快速的將資料處理完成。

第三章 研究方法

軟體系統架構



本計畫欲開發的資料處理系統架構圖如上所示。從底層看上來，包含了Generic Data Processing Library，Dynamic Code Compilation Framework，GPU Data Processing Library，GPU Runtime以及GPU Cluster Runtime，我們將先分項簡單說明如下：

本計畫欲開發的資料處理系統架構圖如上所示。從底層看上來，包含了Generic Data ProcessingLibrary，Dynamic Code Compilation Framework，GPU Data Processing Library，GPU Runtime以及GPU Cluster Runtime，我們將先分項簡單說明如下：

- Generic Data Processing Language & Dynamic Code Compilation Framework

就整個系統而言，我們希望這個系統是非常泛用的，能提供給學界與業界在各個領域的專業人士，只要有大量資料分析的需求，便能夠使用。所以我們參考了做產業界大量資料處理最著名的公司，也就是Google，看看他們是如何處理數個PB（1 PetaBytes = 1000 TeraBytes = 1000000 GigaBytes）

的資料。Google 有數套超大型的分散式系統，分佈在全界各地，每天24 小時的處理來自全世界各地的網頁查詢及連結的分析。他們實作了自家的分散式檔案系統，以及分散式的架構，利用上萬台一般等級的個人電腦，可以在短短的六個小時內處理完1PB 的資料[19]；因為資料分析的時間成本減少了，所以Google 在有限的時間內，可以做更多的測試及嘗試，這也是為什麼如今Google 成為了網路世界的霸主的主要原因。但是這樣的架構所耗費的電力及建置的高成本，都不是一般的科學或研究機構可以負擔的起的，不過但其架構卻是十分可取的

所以我們希望能透過底層GPU Cluster Runtime、GPU Runtime、GPU Data Processing Library所提供的功能，開發一套資料處理的框架，其中包含了Generic Data Processing Language，提供使用者一個簡單並泛用的資料分析語言，如Google Sawzall 或是Yahoo Pig Latin 等，然後透過Dynamic Code Generation Framework，將這些使用者所給定之分析程序，即時經過編譯轉換為在GPU 叢集上執行之二進位檔，再交由GPU Runtime 及GPU Cluster Runtime 來執行。為了提供上層Generic Data Analysis Language 在處理資料時所需要的函式，本系統也將實作一套GPU Data Processing Library，在繪圖晶片的運算平台上，實作一個基於繪圖晶片之泛用的資料處理函式庫；其中包含基本之統計功能，平行化之資料結構，字串處理及數值運算函數等。

這部份的函式庫我們將整合現有的已移植到繪圖晶片上的函式庫，在NVIDIA CUDA 上包括CUBLAS（CUDA Basic Linear Algebra Subprograms）及CUFFT（CUDA Fast Fourier Transform）、CUDPP（CUDA Data Parallel Primitive）[26]，或是在AMD Stream上包括了ACML（AMD’s Core Math Library）等；另外也會實作一些常用之2D 或3D 平行資料結構如QuadTree 或Octree，以及一些基本統計的功能。GPU Runtime & GPU Cluster Runtime

GPU Cluster Runtime 提供了一個通訊及執行控管的平台，讓多台配有繪圖晶片的機器能互相溝通合作，結合多台機器的運算及儲存資源，處理更大量的資料。在這邊我們可採用格網的架構或是MPI 的環境，結合本計畫要開發之對繪圖晶片上行程管理的模組，以及自動資料及工作切割的模組，提供上層GPU Runtime 一個統一執行的環境。

關於資料的切割及工作排程，我們將考慮每個節點上之運算單元的效能（註：繪圖晶片的速度及數量），等比例的將資料集合做切割，透過分散式之檔案系統，動態分配到各個節點上，再交由上層的GPU Runtime 進行計算。

而GPU Runtime 為NVIDIA 或AMD 所提供的能在繪圖晶片上執行程式的介面，如NVIDIA CUDA以及AMD Stream。在本計畫中，我們將選擇其中一種平台做為實作目標，也保留整合異質性運算平台的空間。至於是選擇NVIDIA CUDA 或是AMD Stream，這將取決於上層的GPU Data Processing Library 以及Dynamic Code Compilation Framework 在各個平台上實作的可行性及難易度。但就目前看來，對NVIDIA CUDA 平台而言，本實驗室已有相當多的開發經驗，也進行過一些可行性之分析，我們相信CUDA 是足已_____勝任本計畫的需求的；然而，對AMD Stream 來說，雖然其自家的繪圖晶片硬體的架構優於NVIDIA，但是

軟體的成熟度卻還不及NVIDIA，這也是為什麼到目前為止，絕大多數繪圖晶片平行化的應用都還是基於NVIDIA 的解決方案。故本計畫將會進行進一步的評估，選擇一個最適合的平行運算平台

硬體系統架構

前面提到因為視覺運算以及高速I/O 裝置的興起，我們可以利用一些個人電腦消費者市場裡有的元件，建置高效能的資料處理平台。以下我們將說明我們預計運用那些硬體元件搭配，在單一系統上，盡可能運用SSD 硬碟的提高I/O 的讀寫速度，配合高效能的繪圖晶片顯示卡來加速大量資料處理。

繪圖晶片／顯示卡的部份，我們預計將採用新一代的NVIDIA GTX 295 或是ATI 4870X2，主要原因在於：

1. NVIDIA GTX 295/ATI 4870X2 是消費者顯示卡市場中記憶體容量最大（2GB）的，可以在同一時間處理更大的資料量。
2. NVIDIA GTX 295/ATI 4870X2 都提供了兩顆繪圖晶片的設計，透過在NVIDIA CUDA 以及AMDStream 上非同步執行與非同步記憶體拷貝的設計，同一時間我們可以執行6 筆資料的搬移及運算，達到Pipeline 的效果，提高資料處理的效率。
3. NVIDIA GTX 295/ATI 4870X2 都提供了在繪圖晶片上雙精度的浮點運算功能，這在NVIDIA 前一代的顯示卡上是不存在的，且雙精度浮點運算在某些資料分析中是必需的。
4. NVIDIA GTX 295 改善了非連續記憶體存取的效能，對於平行但非連續的資料結構會有較高的讀寫效能。
5. NVIDIA GTX 295 兩個核心共提供了480 個串流處理器；而ATI 4870X2 兩個核心則提供了1600個串流處理器（但因架構上的差異其實不直接代表效能的產出）。大量的串流處理器有助於處理大量資料時，資料流所導致的程式分支。
6. NVIDIA GTX295 使用了512bit GDDR3 的記憶體，ATI 4870X2 使用了256bit GDDR5 的記憶體，使得繪圖晶片到顯示卡上的記憶體頻寬皆突破200GB/s，有助於在MapReduce 的架構下，暫存資料的存取。而在高速I/O 的部份，為了能將繪圖晶片的運算效能發揮到極致，我們希望能將傳統上資料處理

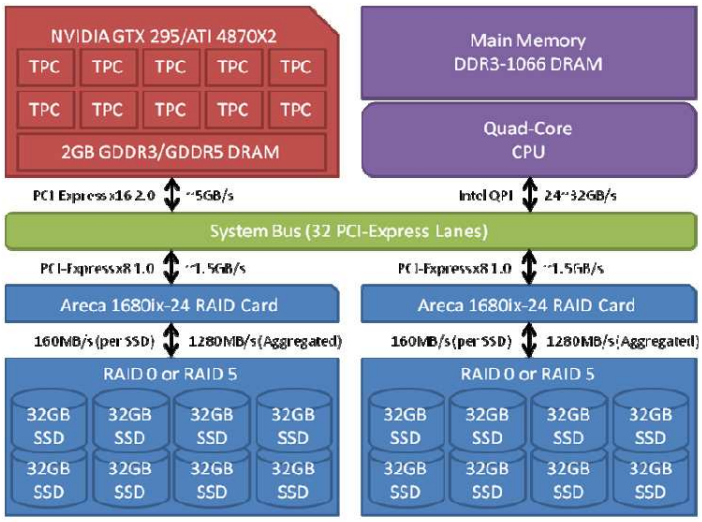
的瓶頸---硬碟的效能，利用日漸普及的高效能SSD 硬碟及磁碟陣列卡（RAID Card），從50~60MB/s 一舉提升到2.4GB/s。但是，要在單一系統內產生這麼高的硬碟讀寫效能是有相當難度的，其中最重要的，莫過於硬體元件的選擇與系統環境的設定。我們參考了許多國內外知名網站的SSD RAID實測，發現把SSD 硬碟組成磁碟陣列最大的問題不在於SSD 的效能，而在於磁碟陣列卡的處理器速度。

例如在中，原作者使用Mtron 16G SSD 來做實驗，Mtron 16G SSD 單顆硬碟的效能就高達120MB/s，若使用RAID0 模式將2 顆SSD 硬碟串接起來時，可以高達240MB/s；但是當原作者將5 顆SSD 硬碟串接起來時，發現在他所使用磁碟陣列卡上（Areca ACR-1220）只能達到386MB/s，最後找出問題在於磁碟陣列卡上的I/O 處理器效能不足；接著他換成較高等級的磁碟陣列卡（Acreca ACR-1231）時跑RAID0 模式時，5 顆SSD 硬碟就如預期的完整的提供了600MB/s 的效能。然而，當原作者希望再把極限往上推進，使用9 顆SSD 硬碟時，又一次被I/O 處理器的效能所限制，最後原作者只能得到接近830MB/s 的讀取效能。

所以從他的實驗中我們知道了在建置高效能SSD 硬碟陣列時，磁碟陣列卡上處理器效能的重要性；我們在本計畫當中，選用了較高效能的磁碟陣列卡，來滿足我們高速I/O 的需求。

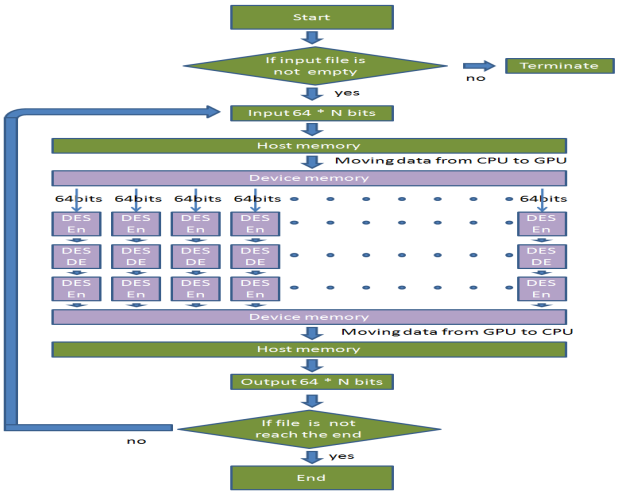
我們不使用傳統在伺服器市場中常見的，使用非常昂貴的SAN（Storage Area Network）[29] 來建置高速的I/O 環境，主要的考量在於可取得性及系統成本。我們希望運用消費者市場

可購得的硬體元件來達成高效能I/O 及高效能運算。
所以我們購買的個人電腦系統硬體架構如下圖所示。我們使用ATI4870X2 來做為我們的系統運算單元，並且在第一年內完成系統評估，實測兩種不同運算單元
之在大量資料處理上之效能；另外，在高速I/O 裝置的部份，我們使用Patriot Wrap Series 之32GB SSD，單一SSD 硬碟可以產生160~170MB/s 的讀取速度，然後利用兩組磁碟陣列卡的RAID0 或RAID5 模式，串接16 顆SSD 硬碟，達到單一陣列1280MB/s，總共2560MB/s 的讀取效能，以產生足夠的I/O 資料量交由繪圖晶片來運算。另外，我們也仔細的衡量了各個元件之間介面的頻寬，並選擇較高效能的磁碟陣列卡(如Acreca 1680ix-24 或Adaptec RAID 5805等），來避免高速I/O 的瓶頸。

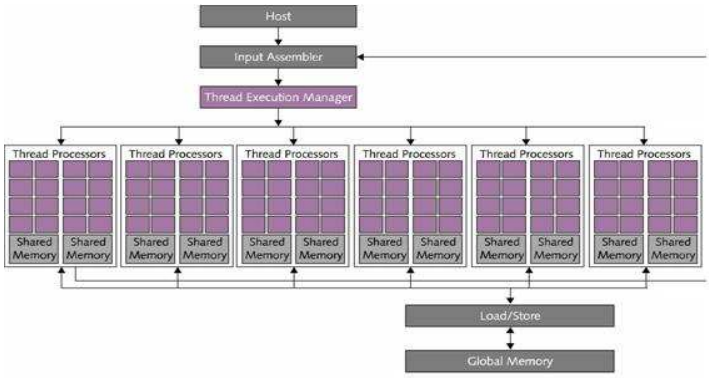


第四章 結果與討論

Accelerating 3-DES performance using GPU



3-DES encryption flow chart in CUDA programming



CUDA architecture

System Design:

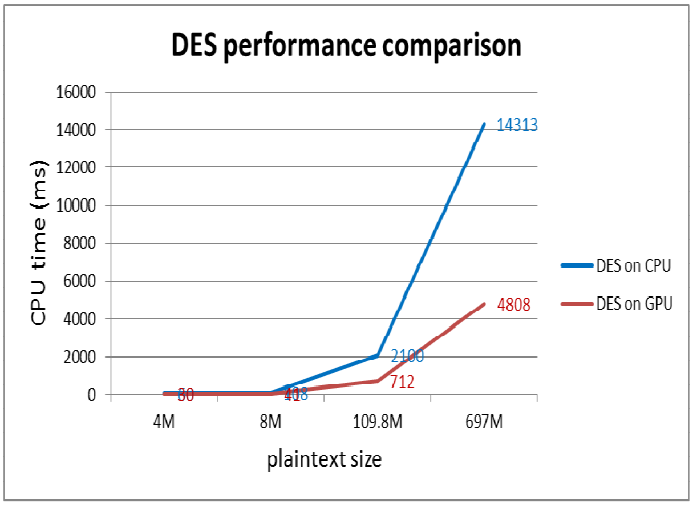
We move the compute-sensitive parts of 3-DES architecture to GPU to speed up whole performance.

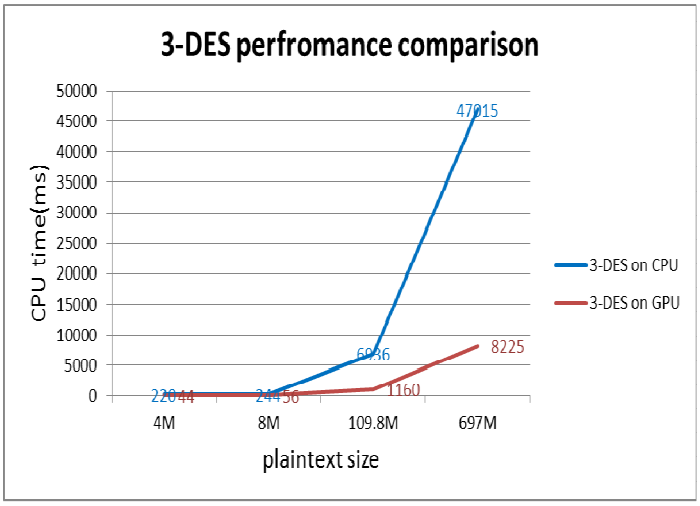
CPU	Intel® Core™2 Quad Processor Q6600 (2.4GHz, quad-core)
Motherboard	ASUS P5E-VM-DO-BP, Intel® X38 Chipset
RAM	Transcend 2G DDR-800
GPU	NVIDIA 9800GT 521MB (ASUS OEM)
HDD	WD 320G w/ 8MB buffer

Hardware configuration

Experimental results:

DES and 3-DES performance successfully improve about more than six times in GPU than in CPU when the file size is larger than 512M bytes.





Implementation of a GPU Based Main Memory Database

Implement several common functions of data base on GPU to improve the performance of data base operations.

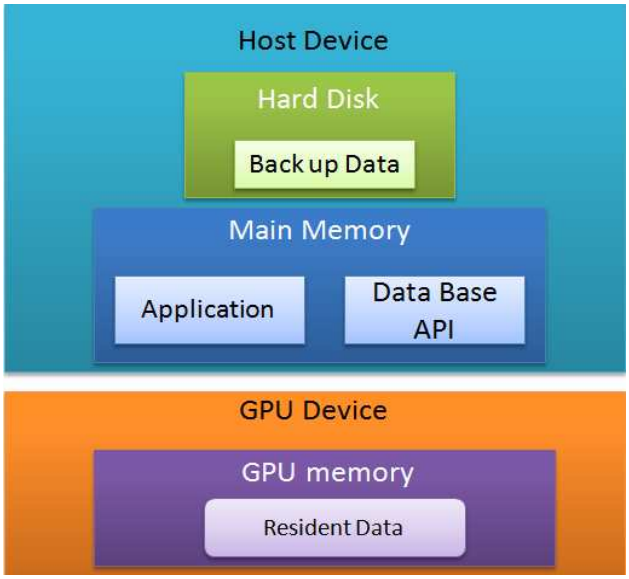
1. Support select, insert, delete, update and the other operations of database.
2. Store entire data base on GPU memory by using 2-D array as data structure to avoid the transmission overhead.
3. Support key-value pair sorting by modifying the sorting function of CUDPP library.
4. Support aggregated functions for eachdata group.

Contribution

A GPU based main memory database is proposed.

1. Observation of turning points in the performance comparison with SQLite.
2. Figure out a ratio of No. data in query result to total No. record. When the ratio is exceeded our GPU DB has better performance.

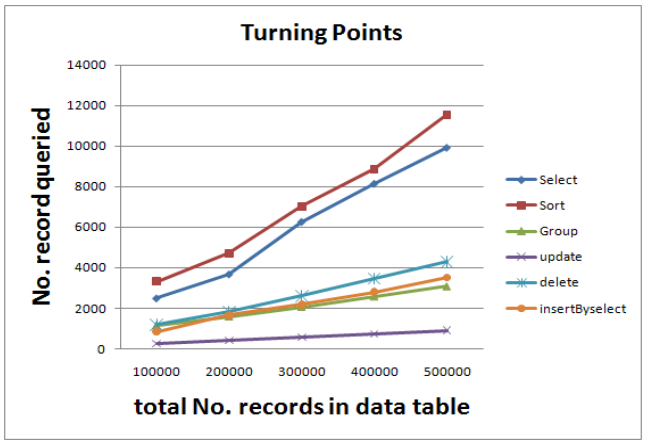
System Architecture



List of the ratios

Function Name	Ratio of No. data queried to total No. records (%)
Selection Query	1.926%
Selection Query and Sorting Data	2.061%
Selection Query and Data Group operations	0.491%
Data Insert According to Selection Query	0.784%
Data Update	0.161%
Data Delete	0.784%

Turning points



第五章 計畫成果自評

截至目前為止，本計畫的目標可以說大部份都已經完成。在繪圖晶片架構下，我們發展了自己的加解密系統提供加速，也發展了CUDADB提供快速且高效能的Database。我們於資料探勘方面，分群演算法是很多應用的前處理，對此在研究過程中，我們發展了一套可以run在GPU上面的K-means演算法，大大提升分群的速度，也在分類上提供了CUDA版本的決策樹，雖然在精確度上面略為下降一點，但在速度上卻有了長足的突破。

這個研究確信可提供低成本高效能的平台以及架構。繪圖晶片上的軟體開發是目前非常新

的技術，也是未來高效能運算的趨勢之一，透過本計畫我們獲得最新的開發經驗，期勉未來配合NVIDIA 或AMD 一起推廣平行運算平台。

低成本之分散式資料處理系統本計畫欲實作之分散式資料處理系統軟體，在未來將提供給各大資料處理中心或有需要之研究中心使用，可以自行購置低成本的硬體元件進行大規模的運算，讓他們可以隨時隨地調用大量的運算及I/O 效能，在極短的時間內完成數據的分析，提升研究的水平，而不需要自行建置大型的分散式系統，亦不需求助於國內各大計算中心，為了分析一筆資料還得經過一層又一層的申請及核覆，提升研究的效率。

目前本計畫在論文的產出方面，成果如下:

國際期刊論文：

1. Yue-Shan Chang & Ruey-Kai Sheu & Shyan-Ming Yuan &Jyn-Jie Hsu " Scaling database performance on GPUs " Information Systems Frontiers A Journal of Research and Innovation © Springer Science+Business Media, LLC 2011 10.1007/s10796-011-9322-0

國際會議論文：

1. Hsiu-Pang Yeh & Yue-Shan Chang & Chia-Feng Lin & Shyan-Ming Yuan "Accelerating 3-DES performance using GPU" CyberC 2011
2. Chun-Chieh Chiu, Guo-Heng Luo & Shyan-Ming Yuan. A Decision Tree Using CUDA GPUs. In Proceedings of Information Integration and Web-based Applications & Services (iiWAS2011),ISBN 978-1-4503-0784-0, Vietnam, 5-7 December, 2011.

國內會議論文：

1.邱俊傑 & 黃聖凱 "基於圖形顯示卡之高效能 K-means 分群"

2011 Symposium on Digital Life Technologies