國 立 交 通 大 學

資訊學院　資訊學程

碩士論文

CSX 加速器應用在 HPCC 的效能評估

PERFORMANCE EVALUATION OF CSX ACCELERATOR FOR HPCC

研究生:吳育銘

指導教授：鍾崇斌 博士

中 華 民 國 九 十 八 年 一 月

CSX 加速器應用在 HPCC 的效能評估

PERFORMANCE EVALUATION OF CSX ACCELERATOR FOR HPCC

學生：吳育銘　　**Student**：Yuh-Ming Wu

指導教授：鍾崇斌　　**Advisors：**Dr. Chung-Ping Chung

國 立 交 通 大 學

資訊學院　資訊學程

碩士論文

Submitted to College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science

January 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年一月

CSX 加速器應用在 HPCC 的效能評估

學生：吳育銘　　指導教授：鍾崇斌

國立交通大學

資訊學院　資訊學程

摘要

由歷年全球前五百大超級電腦排名中，可以觀察到利用加速器之特性應用於提升運算效能趨勢。本研究採用現有該類解法，由其中選擇一具優勢者，進行其效能評估，並進一步提出改善建議。於傳統上，我們藉由HPL程式來量測高效能系統所產生之計算量。由於系統之多樣化發展。我們有必要更進一步針對電腦系統中，各項元件以不同程式來識別該系統之特性。HPCC為近年來針對處理器、快取記憶體、系統記憶體以至於內部網路連結等，透過不同程式之結果來觀察系統對各種不同性質之計算做出效能上之評量。我們研究將HPCC程式在一般伺服器搭配CSX加速器的系統上進行效能分析，並試圖獲得最佳化結果。同時，建議未來加速器設計上改良之方向。由本研究我們得知加速在HPC領域中應用之特性與其限制用途。

PERFORMANCE EVALUATION OF CSX ACCELERATOR FOR HPCC

Student：Yuh-Ming Wu **Advisors**：Dr. Chung-Ping Chung

Degree Program of Computer Science
National Chiao Tung University

Abstract

From the lists of the world's Top 500 Supercomputers in the past years, we can see the tendency of employing the characteristics of accelerators to enhance operating performance. This research adopts the current solution, choosing one of the superior accelerators to evaluate its performance and further to offer the suggestions of improvement. Traditionally, we measure the computation power in the high performance system by HPL program. However, because of the diverse development of systems, we need to further focus on the characteristic that each item uses different programs to distinguish its system. HPCC has been used in recent years to evaluate performance by observing the system's various computations on the processor, cache system, memory, the interconnection and so on, through the results of using different programs. We used the HPCC Suite to investigate the system performance of a common server with the CSX accelerator and we tried to obtain the optimal result. At the same time, we also made suggestions about future direction of improving accelerators. Through this study, we learned about the characteristics of applying accelerators in the HPC domain and its suitability.

目錄

表目錄

圖目錄

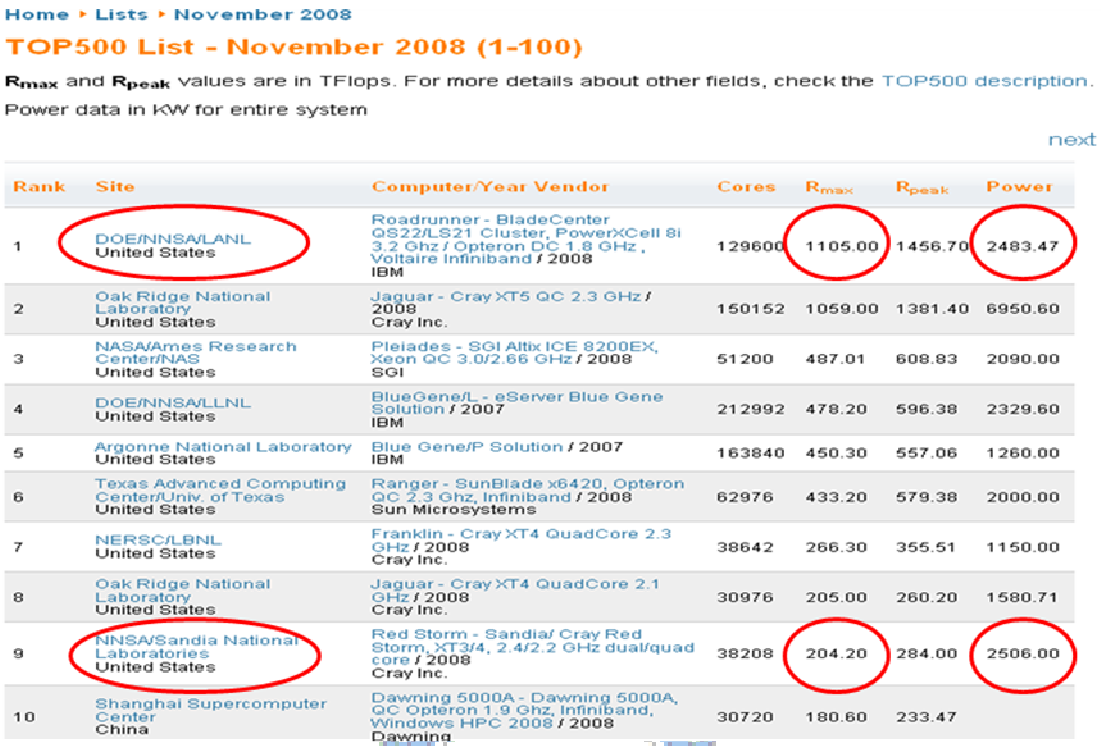
第一章　序論

1-1 前言

高效能運算(High Performance Computing, HPC)已廣泛使用於各種科學模擬的應用領域，相對於日益增加在運算上的需求結果。全球各大超級電腦中心建置之計算量亦不斷提升，在全球前五百大超級電腦(Top500)[1]排名，由圖 1-1 所示自 1993 年迄今其成長速度一直維持著摩爾定律(Moore's Law)[2]每隔 18 個月便增加一倍的速率持續提升運算效能，迄今已邁進單一系統計算總量達到每秒進行 10 的 15 次方双精準度浮點運算(PetaFlops)水準。當建置日益龐大的計算設備時，將面臨到兩個主要的挑戰議題：首先是耗電，若僅使用一般電腦主機之 CPU 來獲取計算量而言，根據美國 Sandia 國家實驗室超級電腦中心之 Red Stom 系統[3]為例，其計算量為 326GFlops，耗電量需求為二百萬瓦(2500KW)；其次該系統所占面積為 4,500 平方呎。

圖 1-1 、Top500 趨勢圖

有鑑於一般 CPU 設計上考量使用之廣泛性，對於進行大量數值運算時，所能提供之運算單元通常在一個 CPU 僅有一個。因此近年來開始考量採用混合式運算(Hybrid Compute)平台，利用運算加速器具有大量運算單元之特性來增加效能並減少成本。以目前在 Top500 首位的美國 Los Alamos 國家實驗室之 Roadrunner 系統[4]採用 IBM Cell B.E 加速器將總運算量達到 1 PetaFlops，耗電量為 2345KW，所佔樓板面積為 5200 平方呎。更有效的增加計算效能並節省耗電量及空間使用。圖 1-2、Top500 系統架構與耗電量比較。



Home ▸ Lists ▸ November 2008

TOP500 List - November 2008 (1-100)

R_{max} and R_{peak} values are in TFlops. For more details about other fields, check the TOP500 description.
Power data in KW for entire system

next

Rank	Site	Computer/Year Vendor	Cores	R_{max}	R_{peak}	Power
1	DOE/NNSA/LANL United States	Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 Ghz / Opteron DC 1.8 GHz , Voltaire Infiniband / 2008 IBM	129600	1105.00	1456.70	2483.47
2	Oak Ridge National Laboratory United States	Jaguar - Cray XT5 QC 2.3 GHz / 2008 Cray Inc.	150152	1059.00	1381.40	6950.60
3	NASA/Ames Research Center/NAS United States	Pleiades - SGI Altix ICE 8200EX, Xeon QC 3.0/2.66 GHz / 2008 SGI	51200	487.01	608.83	2090.00
4	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution / 2007 IBM	212992	478.20	596.38	2329.60
5	Argonne National Laboratory United States	Blue Gene/P Solution / 2007 IBM	163840	450.30	557.06	1260.00
6	Texas Advanced Computing Center/Univ. of Texas United States	Ranger - SunBlade x6420, Opteron QC 2.3 Ghz, Infiniband / 2008 Sun Microsystems	62976	433.20	579.38	2000.00
7	NERSC/LBNL United States	Franklin - Cray XT4 QuadCore 2.3 GHz / 2008 Cray Inc.	38642	266.30	355.51	1150.00
8	Oak Ridge National Laboratory United States	Jaguar - Cray XT4 QuadCore 2.1 GHz / 2008 Cray Inc.	30976	205.00	260.20	1580.71
9	NNSA/Sandia National Laboratories United States	Red Storm - Sandia/ Cray Red Storm, XT3/4, 2.4/2.2 GHz dual/quad core / 2008 Cray Inc.	38208	204.20	284.00	2506.00
10	Shanghai Supercomputer Center China	Dawning 5000A - Dawning 5000A, QC Opteron 1.9 Ghz, Infiniband, Windows HPC 2008 / 2008 Dawning	30720	180.60	233.47	

圖 1-2、Top500 系統架構與耗電量比較

此外，過去長久以來經常用來評估 HPC 系統效能程式 HPL (High Performance LINPAK)在功能上對於 PetaScale 之系統評量上已逐漸不敷使用，因此針對 PetaScale 系統所設計出新一代之效能評量程式 HPCC(High Performance Computing Challenge)[5]亦逐漸取代傳統 HPL 所缺少觀察的系統效能結果。

本篇論文之研究將放在不同類型加速器的介紹，以及針對所選定之 CSX 加速器結合主機伺服器自身的計算能力，應用於 HPCC 效能之提昇及相關改進建議做進一步之探討。

1-2 加速器背景說明

目前主要在市場上應用於 HPC 領域的加速器產品有四種[6]：

1-2.1 FPGA

(例如: Virtex-5[7])：元件可编程邏輯閘陣列(Field Programmable Gate Array ,FPGA)在消費性電子產品市場上常用於嵌入式系統，應用在 HPC 方面於 Cray 或 SGI 超級電腦產品上直接與系統匯流排連接 CPU 與記憶體系統。軟體開發環境使用 VHDL 或 C-Like 編譯器輔助 CPU 在即定點(Fix-point)之數學函式運算上加速其計算效率，尤其在基因演算法上更可達到一般 CPU 100 倍的加速[8] 效能為 352 GMACS，同時具備低功耗(25Watts)等優點。

然而，程式開發人員在使用上需先俱備可重組化硬體方面的知識並熟悉相關演算法轉換成 VHDL 語法，對於一般 HPC 領域程式開發人員所需花費的時間較久。在雙精準度浮點運算(Double Precision)上效能僅有 2.5GFlops。

1-2.2 Cell B.E :

Cell 處理器原本為 IBM 為消費性遊戲機平台所開發出之 Power 系列微處理器[9]，每個 Cell 內含一個 PPE(Power Processor Element)控制程式流程及八個 SPE (Synergistic Processor Element)處理計算工作，在單經度浮點運算(Single Precision)上擁有 204 Gflops 計算能力，功率消耗為 50Watts。

程式開發環境上，SPE 與 PPE 之間資料傳遞需由程式開發人員自行控制記憶體區塊的搬移，且必須確保每個 SPE 的資料量使用在 16KB 以內，才能發揮最大功效。由於在最初設計上，Cell 處理器之目標放在嵌入式系統遊戲機市場，因此架構上以 32 位元運算為主，對於雙精準度浮點運算能力僅有 14.63GFlops。

1-2.3 GPU

(例如 Nvidia GT280) : 利用顯示卡內繪圖處理器提供 HPC 大量平行運算之需求[10]，透過 GPU 自身所擁有的繪圖管線串流(Graphic Pipeline

Streaming)進行多道序列(Multi-threading)運算，在單精度浮點運算上理論值可達到 512GFlops，電量需求則較其他加速器來得高耗電量 240Watts。

現有的在 HPC 程式開發環境有 Nvidia CUDA 及 ATI Boork+兩種，將進行計算之矩陣轉換成 Texture 形式透過呼叫 OpenGL 函式處理。由於一般對與影像處理之精密度僅需使用至單精密度就已滿足產品在繪圖上的需求，因此在雙精密度計算效能僅有 5 GFlops。

1-2.4 CSX 加速器：

由 Clearspeed[11]公司針對 HPC 市場所開發之產品，其特色在低功耗僅 33Watts，完全支援 IEEE 754 所定義之單/雙精度浮點運算皆可達到 50 GFlops ，程式開發環境可直接使用 CSXL 所提供之 BLAS(Basic Linear Algebra Subprograms)[12]或使用完全與現行 C/C++語言相容之 Cn Compiler，另外最大的特點在於 CSX 加速器可與多核心伺服器進行多序列協同運算，其他種類產品一旦分配計算至加速器後伺服器需等待加速器計算完成後程式才能繼續流程的方式更有效運用伺服器本身的計算資源。

因此，考量以上四種產品，CSX 加速器擁 1.有較低的功耗，2.符合 HPC 計算使用的較高雙精準度浮點運算能力，3.針對 HPC 程式開發環之特性，4.連接介面為 PCI-X 或 PCI-E (x8)一般伺服器皆可使用。我們將選定採用 CSX 加速器作為本篇論文的實驗目標。

1-3 CSX 加速器介紹

1-3.1 CSX 硬體架構

如圖 1-2 所示 CSX600 為單指令流多數據流(Single Instruction Multiple Data , SIMD) 多核心輔助處理器(Coprocessor)，資料透過系統匯流排與伺服器相連，架構上主要區分為兩部分 Mono Unit 負責將指令解碼與資料分配

到加速卡上的記憶體，與 Poly Execute Unit 內包含 96 個 PEs (Processing Elements) ，每個 PE 同時執行相同指令，執行完成後透過 PIO (Programming I/O)單元逐一將資料回傳給 Mono Unit 。

Mono Unit 主要功能為 1. 處理非陣列型態資料，2. 控制 CSX 程式執行流程並擷取後解碼，3.控制 CSX 加速器上 DDR 記憶體位址及資料分佈，4. 主記憶體所對應的 PE 區塊並負責傳遞。在運算指令部份可進行 1. 整數及浮點數之加減乘除運算，2.基本邏輯運算(and , or , not , exclusive-or)，3. 算數及邏輯位元移動，4. 資料比較，5. 暫存器資料搬移，6.記憶體存取等指令。指令暫存(Instruction Cache)具有 8Kbytes 空間，資料暫存(Data Cache)具有 4Kbytes 空間。同時最多可進行八道執行序列。

Poly Execute Unit 負責將 Mono Unit 所分配之資料傳送到每個 PE 後進行計算，對於 Poly 執行單元而言處理資料型態為一維陣列均勻分散至 96 個 PE 來執行，所有 PE 平行處理在相同之指令下之個別的資料後將再依序透過 PIO 傳回加速卡上之記憶體上，每個 PE 含有 6KB 記憶體及 128 Bytes 暫存器檔案。Mono Unit 所解碼後對於 PE 的指令型態為超長指令字(Very Long Instruction Word, VLIW)，功能上每個 PE 具有 1. 運算邏輯單元(arithmetic-logic unit, ALU)，2. 乘加法器單元(Multiply-Accumulate Unit)，3. 六十四位元浮點運算單元(64bit Floating Point Unit, FPU)，4. 可程式化輸出入單元(PIO)，5. 鄰近 PE 資料傳遞路徑(Swazzle Path)，6. 除法及平方根單元(Divide/Square root Unit)，在 PE 的指令集方面支援有：1.浮點數之加減乘除運算，2.基本邏輯運算(if else)，3. 算數及邏輯位元移動，4. 資料緩衝控制，5. 暫存器資料搬移，6. 鄰近暫存器存取。其中每個暫存器資料寬度為 64bit (8 bytes)。

整體效能 CSX600 時脈為 210MHz，每個 PE 約可提供 0.5 浮點運算能力，內部資料頻寬約為每秒 1TB。消耗功率為 25 瓦(Watts)，每瓦功耗可提供相當於 2GFlops 效能。

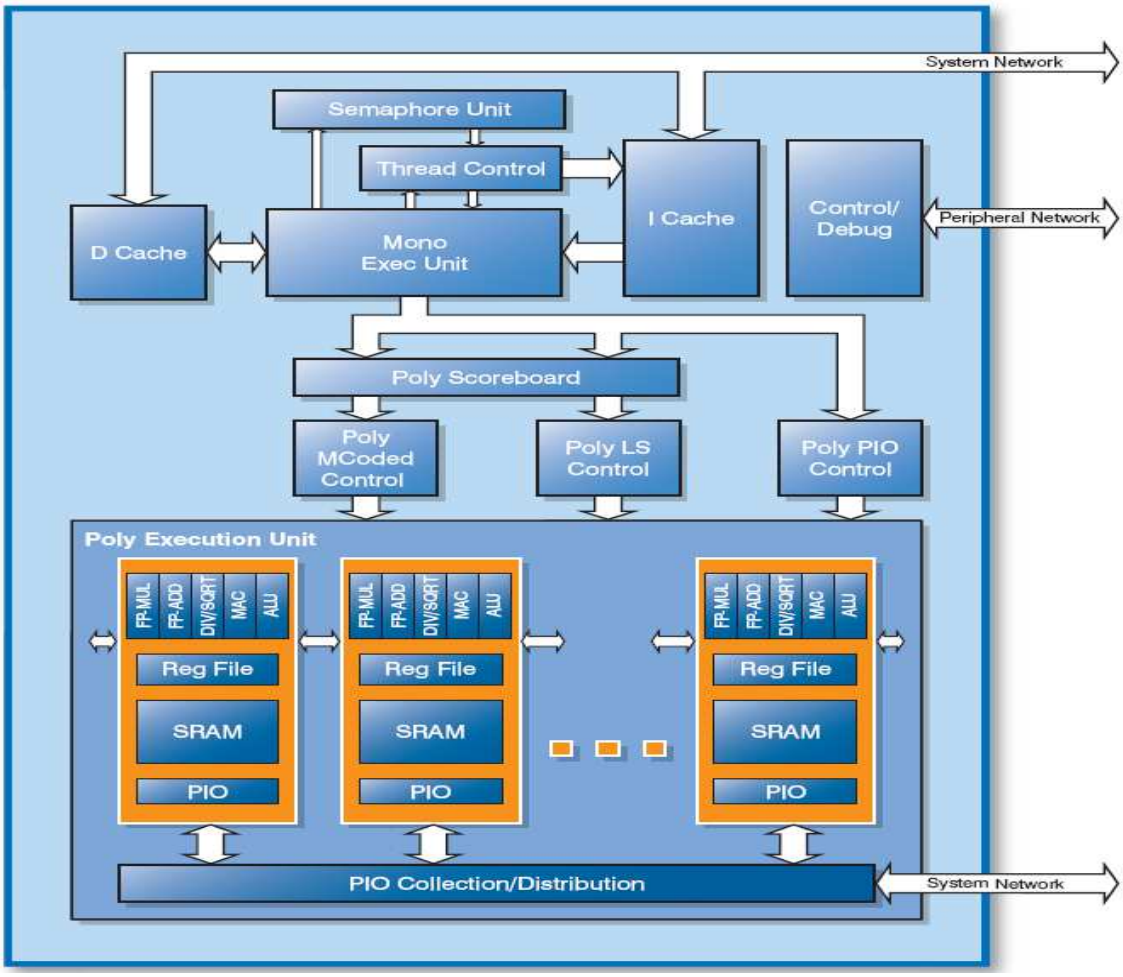


圖 1-3、 CSX600 硬體架構

1-3.2 CSX 使用介面

CSX 如圖 1-3 所示提供兩種方式供使用者利用：1. CSXL 函式庫提供 BLAS 或 LAPACK[13]直接配合市面上現有數學函式庫，如 Intel Math Kernel Library(MKL)[14]或 AMD Core Math Library (ACML)[15]，在無須修改原始程式碼的狀況下呼叫 BLAS 並使用動態連結將數學函式庫中 CSX 可提供加速的功能經由一般編譯器以平行序列(Pthread)的方式，同時結合多序列計算環

境，在程式執行時宣告伺服器與加速器之計算分配比例及其原來所使用之數學函式庫之動態連結目標檔，即可自動在 CSX 所支援之計算核心時將分配資料於伺服器與加速器進行多核心序列運算(Multi-threads Computing)。2. 藉由 CSX 所提供 API 為 Cn(C extended with mono and poly keywords)編譯器針對所需要在加速器執行的程式碼與主程式分開編譯，最後以 C++語言物件的方式傳遞所需參數，藉以在主程式中呼叫使用。

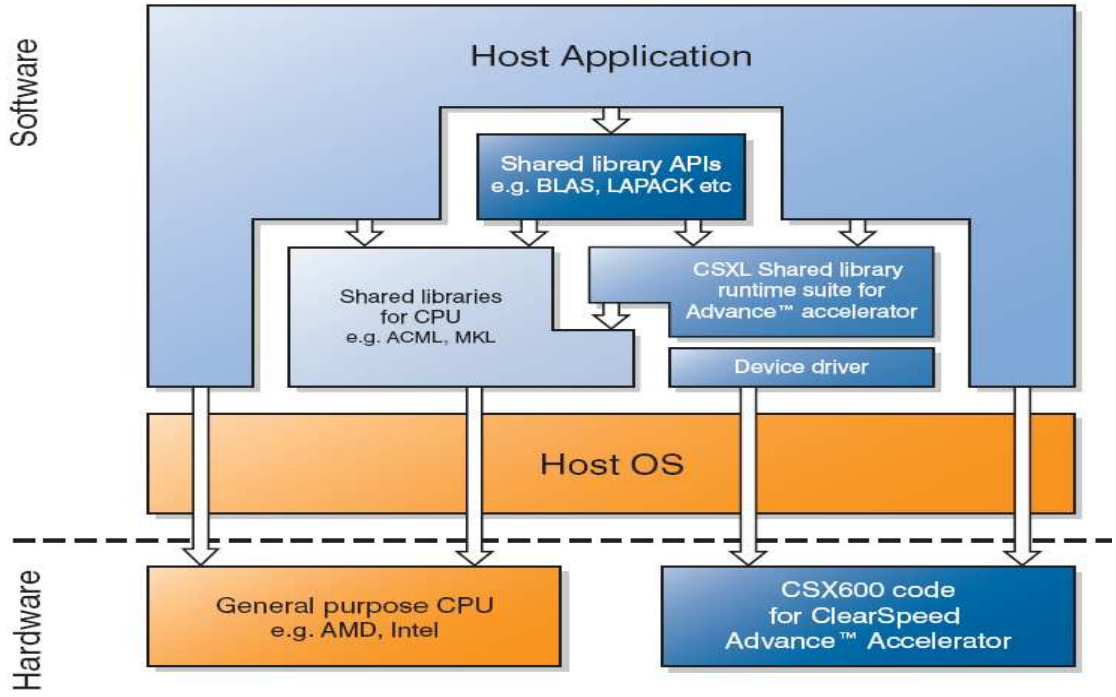
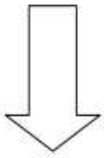


圖 1-4、 CSX 軟體介面

在撰寫 Cn 程式時比較需注意到的特色為，CSX 加速器的 96 PEs 為同時運算之單元，因此資料每次以 96 一維陣列為單位一次執行[16]，範例如下：

```
void daxpy(double *c, double *a, double alpha, uint N) {
  uint i;
  for (i=0; i<N; i++)
    c[i] = c[i] + a[i]*alpha;
}
```



```
void daxpy(double *c, double *a, double alpha, uint N) {
  uint i;
  poly double cp, ap;
  poly int pe_num=get_penum();
  for (i=0; i<N; i+= get_numpes()) {
    memcpym2p(&cp, &c[i+pe_num], sizeof(double));
    memcpym2p(&ap, &a[i+pe_num], sizeof(double));
    cp = cp + ap*alpha;
    memcpyp2m(&c[i+pe_num], &cp, sizeof(double))
}
```

在範例中 get_numpes 為 PE 數量，在迴圈中將 a , c 陣列中資料切割為 96 之倍數一次傳遞平行運算後並傳回至記憶體。當變數宣告時資料為 poly 或 mono 型態編譯器才會將該參數使用於 CSX 加速器上意義上可將 mono 變數視為單一數值 poly 變數視為一維陣列來使用，其餘在程式開發上與標準 C/C++語法相同並在伺服器端執行。

1-3.3 加速器與伺服器合作計算方式

Clearspeed 在 CSX 加速器產品上提供 CSXL 函式庫，利用其所提供通用之 BLAS 及 LAPACK 功能。對於原本已使用 MKL/ACML 函式庫產品計算之程

式，直接連結無須更改程式碼即可設定使用 CSX 加速器。當設定使用 CSXL 之功能時需指定伺服器端於執行時之計算比例，當程式呼叫到 CSXL 提供之函式庫時，CSXL 會將指定比例的計算資料分配於 CSXL 之函式庫與伺服器端的 MKL/ACML 函式庫同時運算。在伺服器端之 CPU 與 CSX 加速器皆進行相同之運算功能自動將負載平均分散以達到最佳之效能使用。如圖 1-4。

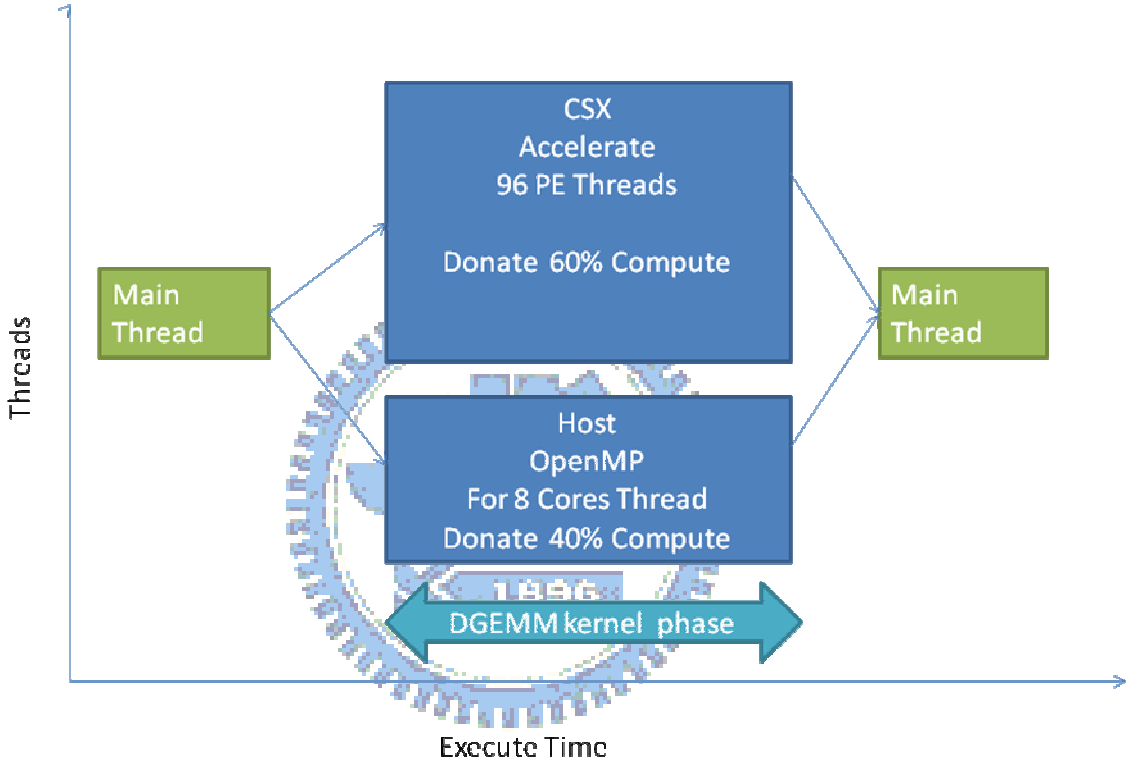


圖 1-5、 CSX 與伺服器執行序列

1-4 HPCC Suite 介紹

1-4.1 HPCC 背景說明

HPCC 是美國國防部先進研究計畫署的高效能技術電腦計畫(DARPA HPCS, Defense Advanced Research Projects Agency, High Productivity Computing Systems)所提出新一代效能評估程式以補強 HPL 功能，藉以評估系統計算效能具有 1TeraFlops、記憶體使用 1TeraBytes、I/O 存取達到 1TeraBytes/Sec 等能力，因此有別於其他效能評估程式只有單一衡量指

標，特別結合若干效能評估程式成為單一執行檔，涵蓋處理器、記憶體、訊息傳遞等因素，企圖模擬應用領域程式並且考量空間(Spatial)和時間(Temporal)，目標成為 2010 年應用於超級電腦系統評估之效能依據指標[17]。

HPCC 主要構成程式為 HPL、DEGMM、PTRANS、STREAM、FFT、RandAccess 及 b_eff 分別針對於電腦系統在計算上所使用到各環節元件校能進行測試。如圖 1-5。

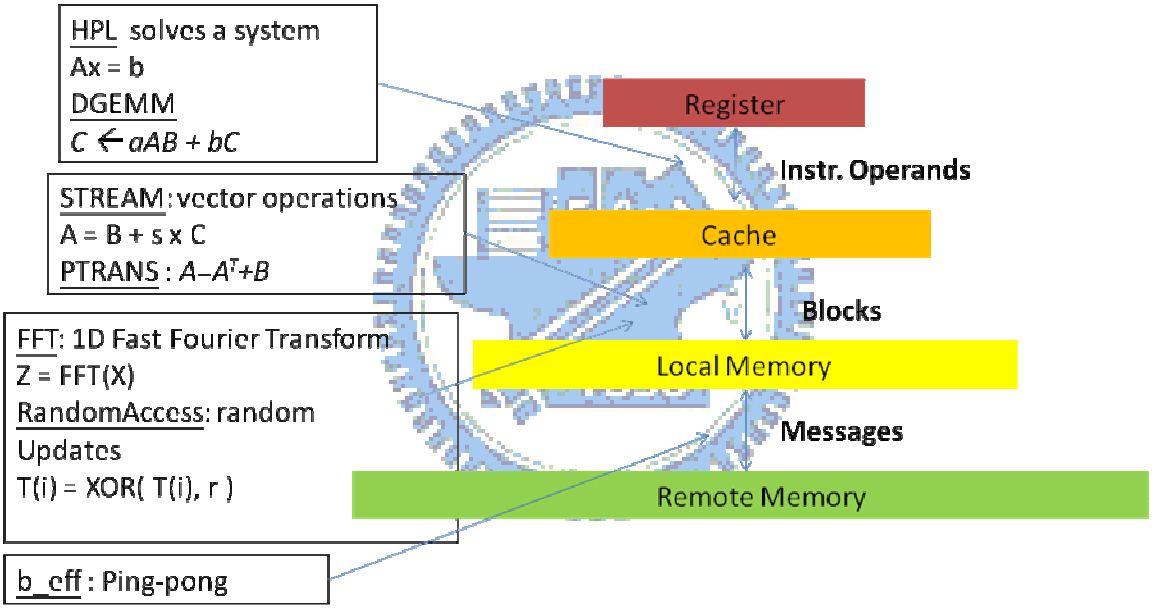


圖 1-6、 HPCC 各校能項目對應

HPL 廣泛地被應用在 Top500 Linpack 系統計算效能評比，這可稱為趨近最高理論值(Toward Peak Performance)工具，STREAM 是量測主記憶體頻寬的工具，RandomAccess 是量測主記憶體隨機更新速率，PTRANS 是矩陣轉換(Matrix Transpose)平行程式，藉以衡量整個系統的訊息傳遞效能， b_eff 用來評估 MPI 訊息傳遞的效能，將這幾個效能評估程式整合為單一個執行檔，以 HPL 最佳效能之矩陣大小為基礎，涵蓋 Local, Embarrassingly Parallel, and Global 三項類別。

1-4.2 HPCC 特性分析

HPCS 所定義高效能計算系統應具有的四項生產性(Productivity)意義:1.高效能(Performance)運算題目在所可接受的時間內完成，2.可程式化(Programmability) 縮短將所需之計算方法開發為可執行程式之時間，3.可移植性(Portability) 與一般 HPC 平台具有共通性，4.高穩定性(Robustness)提供已知的技術減少硬體及軟體之錯誤。

對於用途廣泛的 HPC 計算領域中，一個具備可以應付不同運算型態的系統必須考量到整體架構的平衡性，並且避免在計算過程中所使用到的每一個設備造成整體效能的瓶頸，已使得運算順利運行。因此，考量自處理器單元、記憶體以至於每個伺服器間的連結設備皆須保持在平衡的設計。

然而，各種系統設備的元件在設計的特色上皆有許多不同，更進一步組合成為可運作之 HPC 系統後使得每一套系統皆有其不同之特徵；因此，當我們需要去了解一套 HPC 系統之特色時，HPCC 使用七種簡易的程式以便於觀察該系統之效能。

HPCC 所使用之軟體包含有 HPL 目標在解決系統線性方程式(system of linear equations)，核心運算為 LU 分解法(LU factorization with row pivoting)求 N 組線性聯立方程式，其結果為每秒所能計算之雙精度浮點運算值，由於 HPL 資料運算最多大比例發生在暫存器與 Cache 之間傳遞，無法發揮記憶體與計算單元間傳遞之效能。因此 HPCC 對此增加其他程式以評估其效能。

DEGMM 為 BLAS Level 3 函式庫中所提供之標準功能，用來量測系統雙精準度下標準矩陣相乘之運算效能。

PTRANS 利用標準之記憶體分散演算法(standard distributed memory algorithm)計算之平行程式，程式需要所有計算單元成雙成對同時交換彼此

的訊息，藉以衡量整個系統的訊息傳遞效能，所量測出的傳遞速度以 GB/s 為單位。

STREAM[18]包含 4 個模組：COPY, SCALE,ADD, and TRIAD 量測主記憶體頻寬(*GB/s*)，輸出結果包括單機(只使用 1 計算單元)與 Embarrassingly Parallel 兩種類型，以下分別使用 "*SN*" 和 "*EP*" 來表示，其中 *EP* 表示所有的計算單元同時執行此程式，但是彼此不做訊息交換，此種類型的計算對共享記憶體架構衝擊較大。

RandomAccess 量測記憶體隨機位置之整數更新速率以 *GUPS* 為單位，分成三種模式，單機、Embarrassingly Parallel、透過 MPI 介面執行 "All-to-All" 訊息交換。

FFT 是一維的離散傅立葉變換（Fast Fourier Transform），輸出結果包括單機、Embarrassingly Parallel、透過 MPI 協同工作版本。

MPI [19]訊息傳遞之效能分析，分成兩部份，其一是傳統點對點(Ping-Pong)；另一種為 Ring 方式，此種方式又分為兩種：Naturally Ring 和 Randomly Ring，前者是依照 MPI_COMM_WORLD 順序排列，而後者採取隨機方式，量測出指標有最大 Ping-Pong Latency、最小 Ping-Pong 頻寬、Naturally Ring 之 Latency 和頻寬、10 組 RandomlyRing 幾何平均值(Latency 和頻寬)，其中以 8 *bytes* 訊息來量測 Latency，以 2 *MB* 訊息來量測頻寬。

1-4.3 HPCC 評量模式

基於以上七種程式以標準 C 語言寫成以便於快速於所有平台上編譯執行，HPCC 結果又分為三種類型 1.單機(Local)為伺服器上單一 CPU 所測得之結果，2.內在平行(Embarrassingly Parallel)為同一作業系統內之分享式記憶體(Share Memory)伺服器使用 OpenMP 平行方式之結果，3.整體(Global) 透過 MPI(Message Passing Interface)方式與透過內部網路連結所有伺服器之結

果。由於論文研究之標的為加速器與伺服器之間運算結果，CSX 加速器資料傳遞僅與伺服器記憶體之間，對於透過使用 MPI 呼叫處理器方式反而會增加資料傳遞上的負擔，因此研究之實驗將著重於內在平行 EP 效能結果做為評估分析之依據。

由過去文獻所提供之研究僅探討 CSX 加速器使用 HPL 或單一領域程式後之結果，我們將更進一步透過 HPCC 程式來廣泛檢視 CSX 加速器在 HPC 領域應用上適合之用途以及在現有之加速器設計上尋求改善之建議。

第二章 研究目標與方向

2-1 研究目的

本篇論文研究目的為：

一、於一般多核心計算主機上使用 CSX 加速器，透過 HPCC 程式執行混合式平行序列(於伺服器執行多核心序列同時於加速器執行 SIMD)[20]，觀察並調整以得到各項旗標程式最佳結果。以期獲得 CSX 加速器在伺服上面對對不同計算類型時效能之增益。

二、並且經由實驗過程中所獲得之經驗，建議改善現有 CSX 加速器架構以及未來可應用之領域。

2-2 研究目標

在研究標的實驗平台上，使得 HPCC 程式獲得最佳化結果，並分析加速器在 HPC 領域使用上，對於不同種類之計算模式應用時，建議的使用方式以及加速器的設計上可改善的方向。因此我們將研究的目標逐項分成以下四個方向來進行：

a. 伺服器端與加速器端各別單元本身所能產生之實際 HPCC 計算效能。

b. HPCC 各別程式之計算核心之識別及分析，並設法將計算核心應用於 CSX 加速器上使用。

c. 找出伺服器與加速器配合使用下可獲得之 HPCC 最佳效能。

d. 透過結果來分析改善加速器設計的方向。

2-3 研究方向

為達成上述目標，因此在研究方向上更進一步描述。考量所使用的研究方法為何。

2-3.1 HPCC 各別效能結果

對於各別可提供計算之單元(伺服器與加速器)，獲得實際 HPCC 最佳之實驗結果。因此尋求建立適合的計算環境(Software Stack)，作為整個研究基礎的依據。

在建構的過程中主要的三個元素：作業系統(Operation System)、編譯器(Compiler)、數學函式庫(Math Libraries)。考量 CSX 加速器軟硬體所支援之作業系統主要為 Red Hat Enterprise Linux，在編譯器方面在 Linux 上一般所常用之 GCC / PGI / Intel 皆在考量的範圍內，對於可相容之數學函式庫有 AMD ACML(AMD Core Math Library)、Intel MKL (Math Kernel Library)、GOTOBlas、ATLAS(Automatically Tuned Linear Algebra Software)。基於伺服器端所採用為 IA(Intel Architecture) 平台以及同時需使用 LAPACK 函式庫狀況下，我們將採用 RHEL 及 MKL 為基礎並對於不同版本之編譯器進行 HPCC 效能上比較。主要對於伺服器與加速器之理論值與實際值做出效率的比較。

2-3.2 HPCC 計算核心分析

HPCC 在本研究中，我們將著重於程式中資料型態為陣列資料結構之迴圈的運算上，由其指出可在 Embarrassingly Parallel 模式來進行之程序，對於伺服器而言可進行 OpenMP 平行化，對於加速器而言可進行 SIMD 執行之特性。以期獲得在一台伺服器內最高之效能。以下分別就各個 HPCC 程式，說明分析的結果：

HPL 計算核心功能為 HPL_pdgesv()及 HPL_pdtrsv 分別為對於 N-1xN 矩陣進行因數分解及交換函數運算，將矩陣使用 LU 分解法切割資料為 N/NB 數量之區塊，每個區塊以平行方式運算。其中運算時主要之計算工作則呼叫 BLAS 函式庫中 dtrsm 及 dgemm 計算複雜度為 O(N^3)。在 CSXL 所提供之函數庫中已經提供這兩項功能可直接使用。

PTRANS 程式當中之核心計算為使用 BLAS Level 1 之 DAXPY 及 DCOPY 功能計算複雜度為 O(N^2)。

DEGMM 程式本身即為 BLAS Level3 之功能，CSXL 函式庫中亦有提供，可直接使用於加速器之運算當中。DGEMM 之計算複雜度為 O(N^3)。

STREAM 程式之運算核心分別為 c[j] = a[j]，b[j] = scalar*c[j]，c[j] = a[j]+b[j] 及 a[j] = b[j]+scalar*c[j] 等四種之一維陣列搬移與加法運算，其運算複雜度為 O(N)。

RandomAccess 程式之運算核心為 Table[ran[j] & (TableSize-1)] ^= ran[j]，其計算複雜度為 O(N)。

FFT 程式主要之計算核心為使用 FFTW 函式庫所提供之 zfft1d 功能，計算複雜度為 O(n log (n))。

我們將針對以上 HPCC 當中每個程式之主要計算核心執行方式進行研究，並研究其在 CSX 加速器以何種方式進行上輔助運算可以達到最大效能之增加，以及相較於伺服器的執行方式具效能增益為何。

同時在最佳的執行結果中，將針對 CSX 加速器在能量消耗與效能增加的數據上加以分析。圖 2-1 為 HPCC 各項程式在各領域上所使用趨勢。

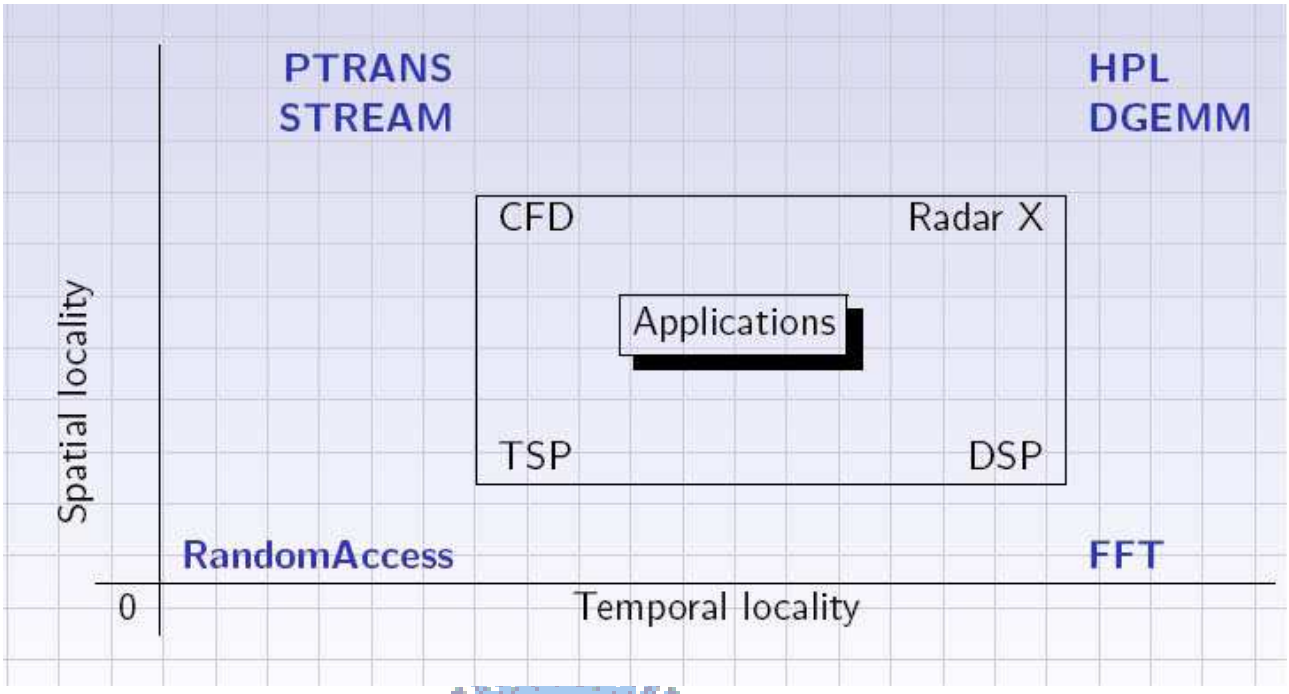


圖 2-1、HPCC 各項程式在各領域上所使用趨勢

2-3.3 混合式 HPCC 運算研究方向

除了利用 CSXL 提供現有之數學函式庫(BLAS,LAPACK)，在未提供之數學函式庫上，使用將現有之功能經由組合之方式以達成相同功能函式的呼叫。

分析 HPCC 各程式所使用之核心計算迴圈萃取出來，使用 Cn 語法將該功能直接應用在加速器上，並透過 CSPX 介面於編譯時連結使用。由於 CSX 加速器使用 PCI-X/E 為連接伺服器介面，因此在處理資料搬移上盡量使用非同步方式傳遞，以使得計算工作與資料傳輸工作得以並列進行[21]。

每次將資料傳遞進入使用加速器時，將陣列轉換切割為 96 個單位之一維矩陣傳入加速器之 Poly 單元運算以利用最大資源使用，在一次 PE 內部運算過程中若需要資料交換，可使用 PE 之間內部路徑交換，最後結果輸出時利用 PIO 單元雙重緩衝單元(Double Buffer)，以使計算資料輸入與輸出可同時交換至加速器之記憶體。圖 2-1 為 PE 間內部資料交換路徑。

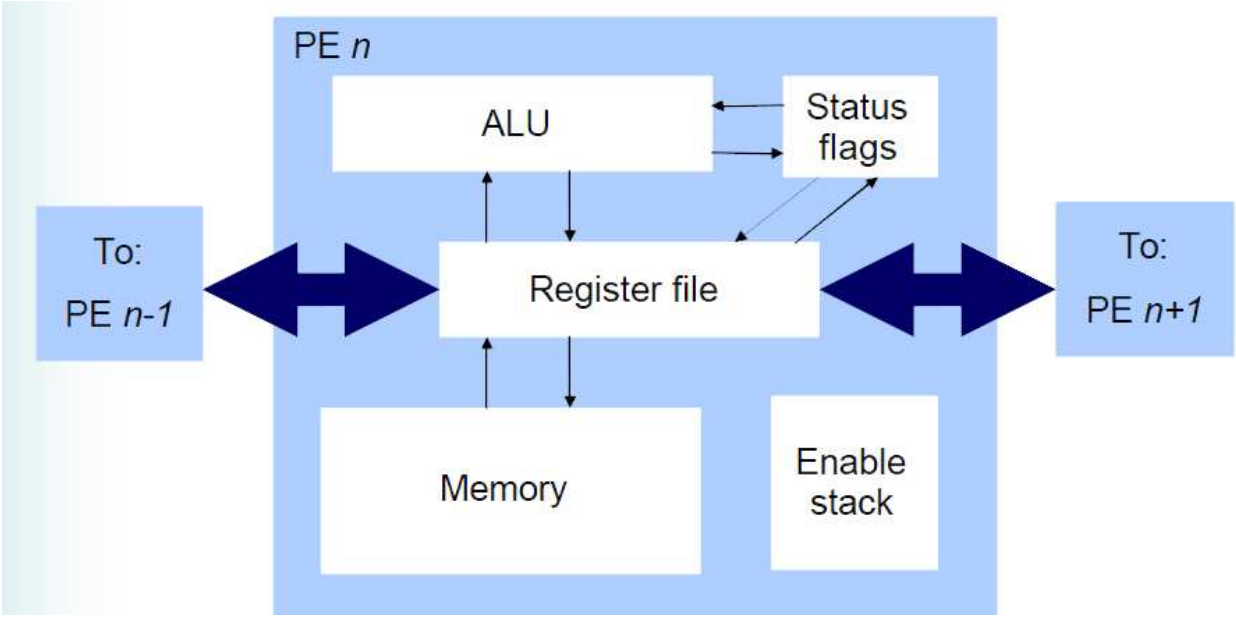


圖 2-2、 PE 內部資料路徑

同時為達到伺服器與加速器之間負載之平衡，首先對於伺服器可產生最佳化之表現上分配不同比例之資料量給予加速器運算。藉以得到在該類型運算時之系統設定參數。並於不同程式間變更時分配比例。圖 2-2 為伺服器與加速器共同執行最佳狀況。

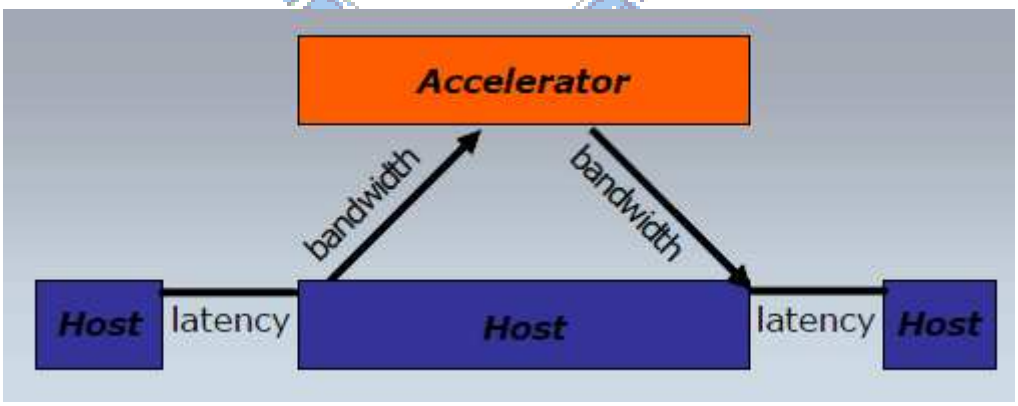


圖 2-3、 伺服器與加速配合計算

由於將一筆資料由伺服器傳遞至加速器運算總共需耗費 56 個時脈延遲時間，但在加速器上完成一筆 64 位元浮點數乘法運算僅需花費 4 個時脈，

同時提供 96 個單元進行平行運算，因此對於計算量選擇是否需要使用加速器來運算，將視資料運算核心迴圈中之計算複雜度以及資料可否大量平行化處理最為主要考量因素。圖 2-3 為伺服器至加速器資料階層圖。

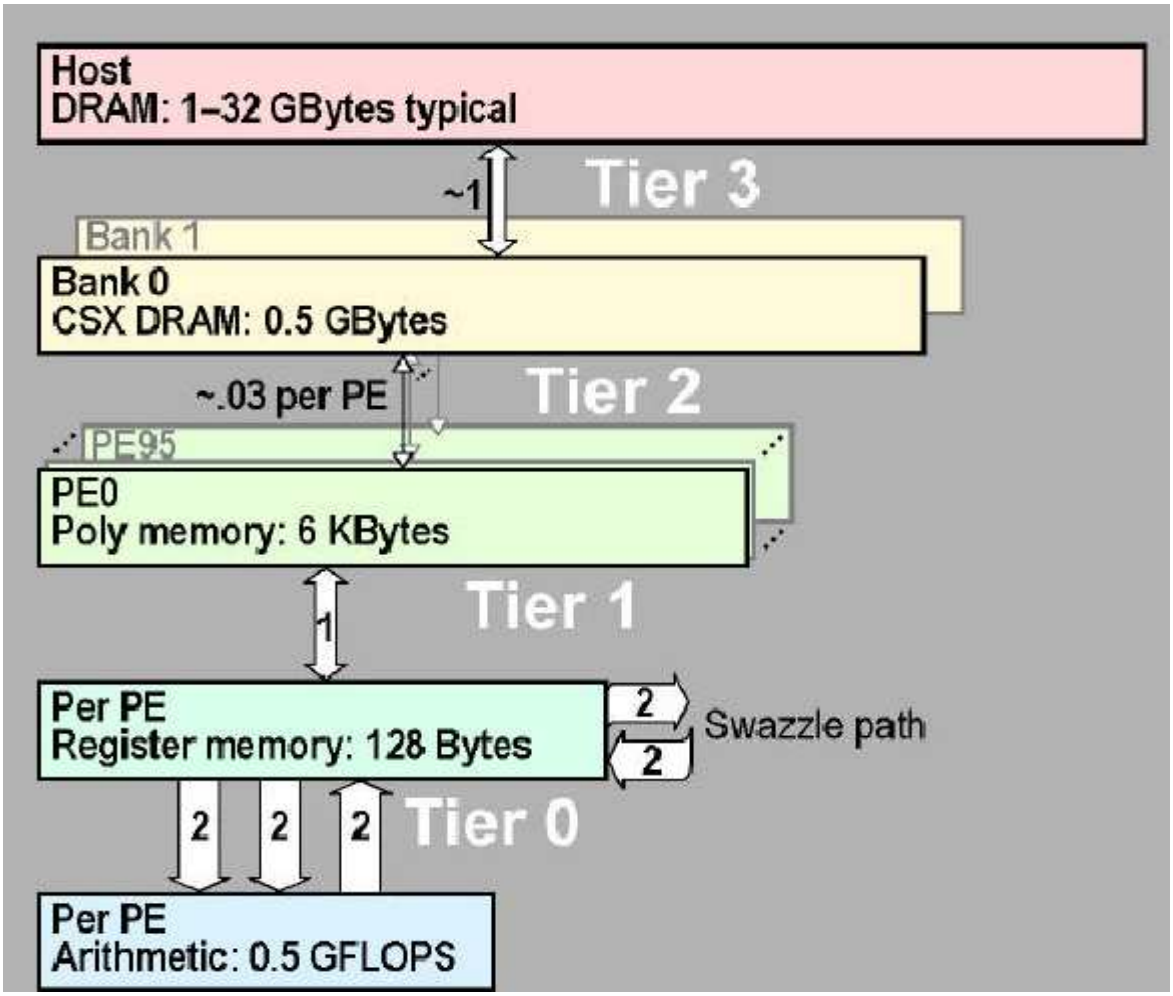


圖 2-4、加速器資料階層

2-4 研究流程

在研究實驗流程中，初步針對在伺服器端與加速器建立可執行 HPCC 程式之計算環境，並決定可穩定且產生正確結果之軟體環境架構及相關編譯時之參數與函式庫呼叫，使得各單項目標控制在最佳化之結果。

對於 HPCC 各項程式所提供之原始碼分析計算核心迴圈內之運算式，嘗試將資料使用在 CSX 加速器提供 SIMD 之方式在同一個指令下對於 96 個 PE 資料進行初步之效能測試，以決定是否將該計算功能使用在加速器上。

逐一對於 CSXL 現有提供之數學函式庫在 HPCC 亦可使用之功能，在同一計算題目下採用不同資料分配比例，以取得該系統上該函式功能，在伺服器端與加速器之間最佳分配結果。並使用該結果比例再逐漸加大計算題目之矩陣，藉以獲取該功能最佳效能結果。 在整個 HPCC 執行過程中，除了需注意 CSX 加速器資料的分配比例外，考量伺服器之多核心對於記憶體存取上每個核心分配之資料頻寬限制，對於伺服器端執行時多序列所使用之 OpenMP (Open Multi-Processing) 計算時序列數(Thread numbers)亦需在實驗中使用不同參數。

因此我們在整個實驗過程中，對於每個效能程式將對於伺服器參數、加速器計算比例以及計算題目的設定三方面進行調校，以獲取整體最佳表現結果。並同時量測計算過程中不同組態下所需耗費之功率。作為後續效能結果分析之參考依據。

第三章 HPCC 實驗及結果分析

3-1 實驗平台介紹

伺服器方面我們採用 HP DL380 G5 機型，內含兩顆 Intel Xeon E5335 四核心 64 位元處理器，記憶體規格為 4GBytes DDR2-667，作業系統為 Redhat Enterprise Linux Version 4 Update 6 (Nahant)，編譯器為 Gnu C/C++/Fortran Version 3.2/4.12/4.2 及 Intel C/C++/Fortran Version 9.1/10.1，數學函式庫方面安裝有 GOTOBlas Version 1.26 、 Intel MKL Version 9.1/10.1 及 ATLAS Version 3.8.2。

CSX 加速器硬體部分我們採用型號為 Clearspeed Advance e620 PCI-Express 8-lanes 內含兩顆 CSX600 多序列陣列處理器(Multi-Thread Array Processors,MTAP)。如圖 3-1 所示。

圖 3-1、 Clearspeed e620 卡

CSX 加速器軟體安裝包含有編譯器 Cn version 3.1 針對於 CSX 加速器上程式之編譯，數學函式庫 CSXL 3.1 包含有部分標準 BLAS(DGEMM, DTRSM, ZGEMM, ZGEMM3M)及 LAPACK(DGEQRF, DGESV, DGETRF, DGETRS, DORGQR, DORMQR, DPOSV, DPOTRF, DPOTR)等功能可供直接呼叫使用，加速器程式介面函式庫 CSPX Version 3.1 用來將加速器子程式與伺服器端主程式連結使用 , CSX 軟體開發工具 SDK Version 3.1，分散複立業轉換函式庫 CSDFT Version 3.1。

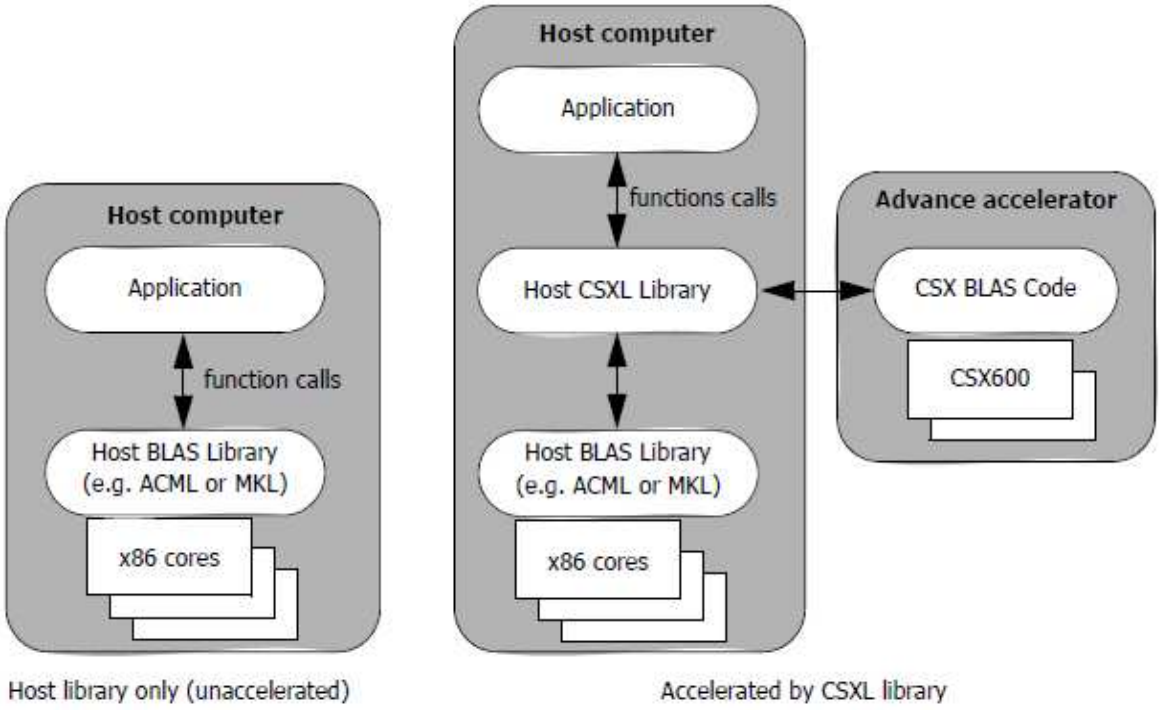


圖 3-2 、CSXL 呼叫 BLAS 示意圖

3-2 初步實驗數據

根據 2-3.1 所敘述之研究方法，我們進行初步實驗確認作業系統、編譯器及數學函式等軟體堆疊(Software Stack)，研究各種組合之可行性後，挑選影響伺服器與加速器各別可獲得之最佳效能結果。伺服器端 CPU 在 HPL 的表現上理論值為 64 GFlops 在實驗各種系統計算環境後，以 Redhat Enterprise Linux 加上 Intel C/C++/Fortran 10.1，配合 Intel MKL 9.1 之組合在伺服器及加速器上所獲得效能最佳。結果如表一所示。

	HPL (GFLOPS)	DGEMM (GFLOPS)	PTRANS (GUP/s)	Random Access (GUP/s)	STREAM Triad (GB/s)	FFT (GFLOPS)
HOST Single cores	7.31	7.46	0.47	0.0105	3.06	0.99
HOST (EP) 8 cores	48.09	56.04	0.46	0.0158	5.36	2.63
Host Use CSX 96 PEs	47.56	64.41	0.49	0.0056	0.04	0.58

表一、伺服器與加速器各別效能

對於伺服器與加速器在比較計算軟體環境後，我們以獲得最佳之 Software Stacks 之組合，初步結果來看在 HPL 方面之效能伺服器在單顆處理器上可到達理論值(8GFlops)之 91%水準，對於 CSX 加速器而言則達到理論值 (50 GFlops) 之 95%效能。

在伺服器方面在使用 OpenMP 之平行化後，效能可以倍數(Speed-up)呈現之程式有 HPL、DEGMM 及 FFT。其程式種類以雙精度浮點運算為效能主要運算之特性。然而對於以記憶體存取之程式(PTRANS、RandomAccess 及 STREAM)，在架構上主要取決於系統晶片組及伺服器上記憶體之既有組態，因此在效能上對於使用處理器數量多寡則無顯著之影響。

在加速器方面使用 SIMD 之結果，在雙精度浮點運算之程式方面(HPL、DGEMM)相當或更勝於使用伺服器八個處理器之結果。然而對於以記憶體存取為主之程式(RandomAccess 及 STREAM)則大幅低於伺服器本身之效能。主要因素為加速器與伺服器所連接之介面 PCI-Express 8 lanes 所影響資料傳遞之時間，對於此特性之程式將大多數執行時間花費在資料傳遞及延遲上所導致。

然而較特別的是 PTRANS，雖然其程式特性主要為記憶體之存取，在伺服器端對於使用處理器數量上無特別的影響，理應在加速器上將花費與 RandomAccess 及 STREAM 相同花費較多時間在資料存取上，但是使用加速器後則有較伺服器上更佳之結果。分析其原因主要在於 PTRANS 程式先等待資料由伺服器記憶體傳遞至加速器之記憶體後，才開始進行相關運算。雖然雙精度浮點乘法運算其效率較伺服器上花費時間少，然而記憶體與加速器間資料傳遞所造成花費較長時間，因此抵銷在運算上所節省之時間。

在 FFT 方面，主要加速器僅使用 1D-FFT 功能，在 HPCC 所使用為 2D-FFT 功能，因此對於非 bit-reverse 之演算法資料交換時，在加速器內部 PE 單元無法與鄰近 PE 使用 Swazzle Path，必須先將資料由 Poly Unit 移出至 Mono Unit 後交換，比其他程式花費較多之額外時間，導致計算所獲得之效能無法在 CSX 加速器上有顯著之幫助。

綜合研究方法與實驗結果後，我們大致可歸納出，計算複雜度大於等於 N^2 之具有雙精度浮點運算程式，若資料型態可簡化為一維陣列時，在加速器運算上可獲得較佳之結果。

在 HPCC 各項程式執行時間比例上，如圖 3-3 及 3-4 所示。對於接下來在功耗方面的評估作為數據之依據。

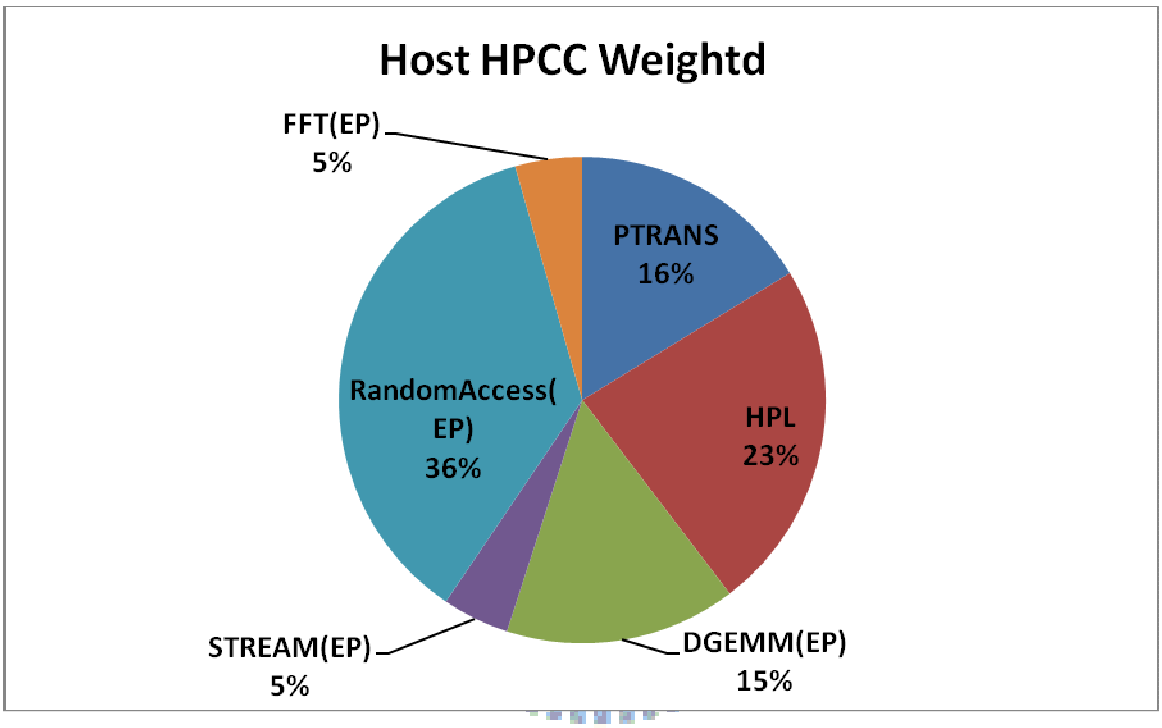


圖 3-3、伺服器各項程式 HPCC 執時時間比例

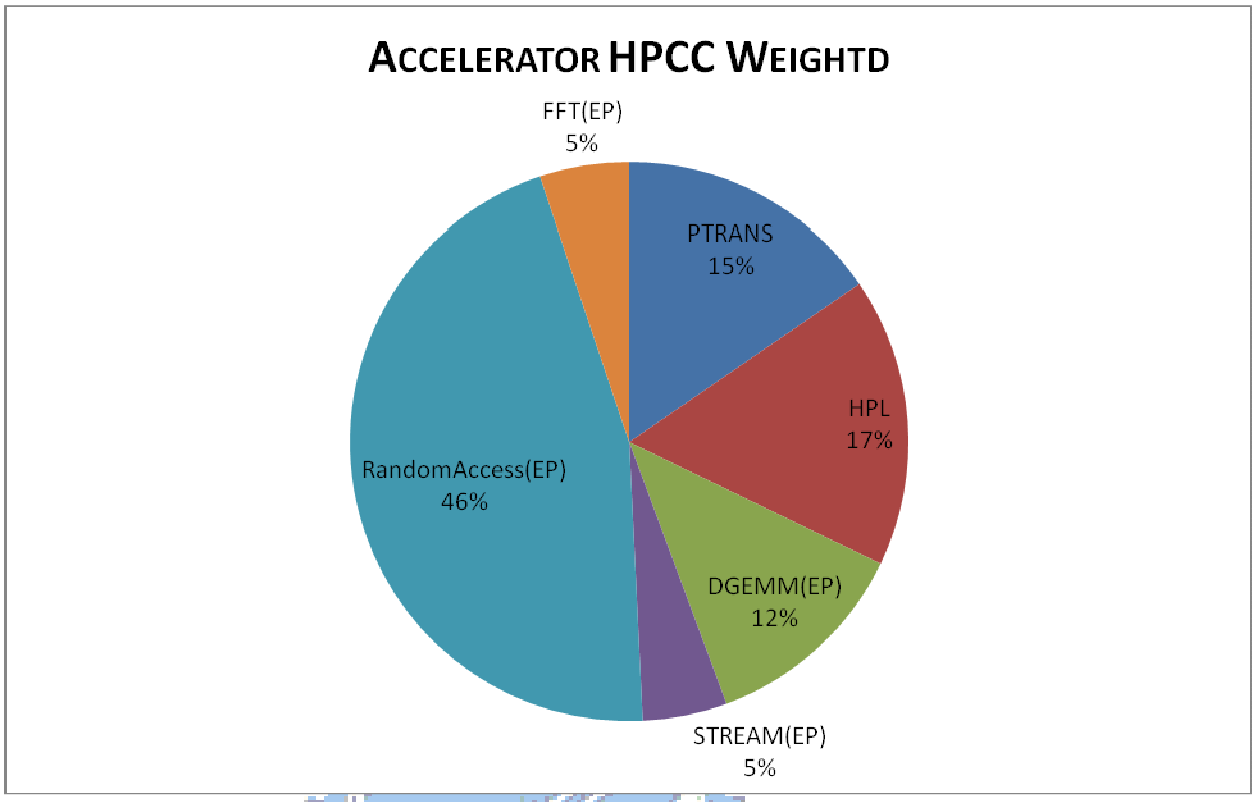


圖 3-4、加速器各項程式 HPCC 執時時間比例

3-3 調整後結果

我們依據 2-3.2 HPCC 計算核心之分析結果，將計算核心部分由伺服器與 CSX 加速器進行混合式平行運算。其餘部分仍以伺服器為主要執行單元。

在伺服器與 CSX 加速器協同運算的結果，如表二所示。我們 HPCC 各個執行結果，在使用 CSX 加速器後相較於僅使用伺服器之結果增加。

尤其在 DGEMM 表現上，其結果更是達到為伺服器加上 CSX 加速器整體理論值(114GFlops)之 87%。在過程中，我們發現到伺服器自身雖有八個處理器可供運算，但在最佳結果卻發生在使用七個處理器。由於我們推斷加速器本身仍需消耗伺服器一個處理器作為資料傳遞使用。因此我們更進一步將計算使用伺服器七個處理器與 CSX 加速器之理論值設定為 106GFlops，我們在整體運算最佳效率可達到理論值之 93%。

當利用 CSX 加速器與伺服器進行混合式平行運算時，在本實驗平台上對於加速器與伺服器資料分配比例分別為佔 60%及 40%計算比重獲得之效能最佳。最主要考量為計算矩陣在記憶體中，傳遞至加速器之效能而對於每套系統使用上會有不同之影響比例結果。

同時，在此結果中再次說明對於使用 CSX 加速器而言，計算複雜度在 O(N^2)(含)以上之雙精度浮點運算，加速器對於伺服器運算效能增加有較明顯改善。

	HPL (GFLOPS)	DGEMM (GFLOPS)	PTRANS (GUP/s)	Random Access (GUP/s)	STREAM Triad (GB/s)	FFT (GFLOPS)
HOST (EP)+ CSX	64.59	99.54	0.74	0.0183	5.40	2.88
Host Speedup	1.34	1.78	1.61	1.16	1.01	1.10

表二、伺服器與 CSX 加速器共同運算結果

3-4 雷達圖

另外，我們使用雷達圖(Kiviat Charts)方式表現，整體系統、伺服器單元與加速器單元效能上之比較。如圖 3-5 所示。

當我們將這三者之效能正規化(Normalized)後，可以更清楚的顯示伺服器與加速器在各項效能上所表現的差異之處。在以計算為主的程式組部份加速器可提供與伺服器相等或更佳之效能，因此在混合式運算時有明顯對效能上的貢獻。從另一方面，對於以記憶體資料傳輸為主之程式組而言，加速器本身執行時，效能表現遠低於伺服器之結果，因此進行混合式運算亦僅有些許效能增加的貢獻。

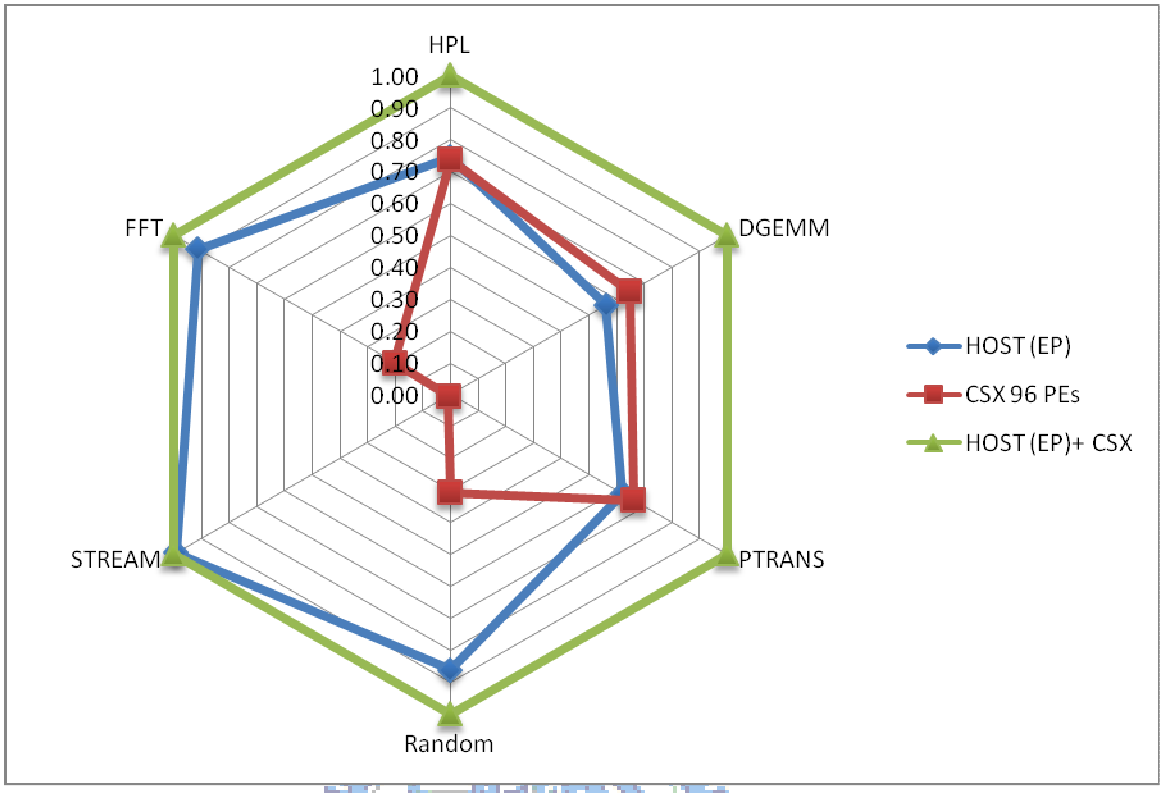


圖 3-5、雷達圖

3-5 功率消耗

當進行 HPCC 各項程式執行時，我們亦同時分別量測在不同情境下，整體電量消耗的功率。以進一步計算在效能方面與能量之間的關係。

經由量測各種狀況後，伺服器在空載(Idle)在功率使用 243 瓦特(Watts)，當空載裝上 CSX 加速卡後功率增加為 266 瓦特。在 HPCC 執行時伺服器為全載(Full Loading)所需功率為 435 瓦特，與加速器共同運算時全載功率為 472 瓦特。當僅使用伺服器在最佳狀態下 HPCC 執行完成所需時間為 499 秒，伺服器與 CSX 加速器之執行時間為 352 秒，因此所花費之時間與消耗的功率比較時，如圖 3-6 所示。

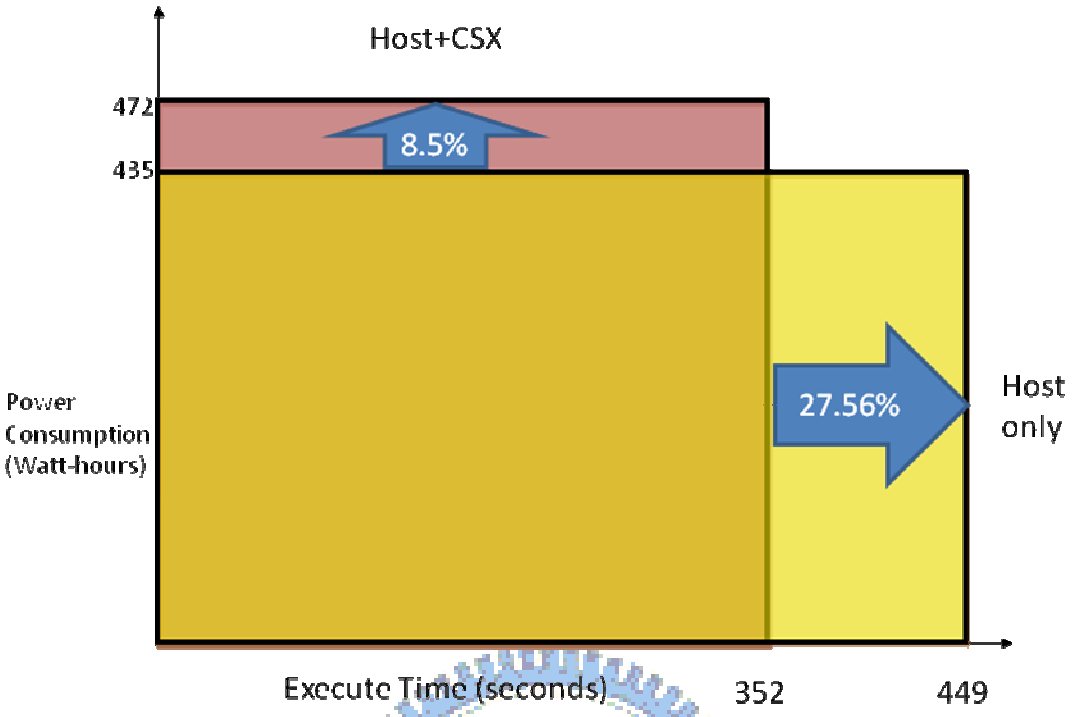


圖 3-6、功率消耗與執行時間比較圖

使用 HPCC 執行所花費之時間後，對於 HPCC 對於伺服器與使用加速器之功率消耗比較，如圖 3-7 所示。

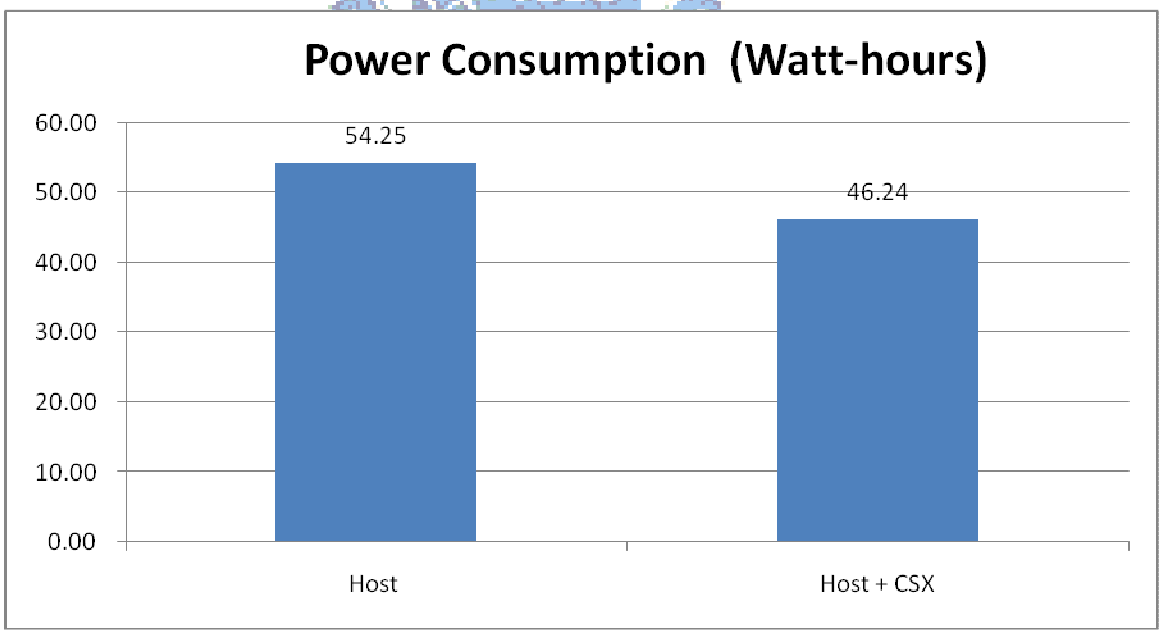


圖 3-7、HPCC 執行所花費功率比較

對此我們可以指出在伺服器加上加速器後，對於 HPCC 的執行上，其功率之使用較未加上加速器更能節少功率消耗。

另外我們進一步分析使用加速器後，對於功率消耗在各程式效能上所增加之比例，如圖 3-8 所示。使用加速器在 HTRANS、HPL、DGEMM 及 RandomAccess 等程式之效能在功率消耗在執行過程中，有降低功率消耗之意義。

然而對於 STREAM 及 FFT 程式而言，所消耗之能量則大於效能增加比例。

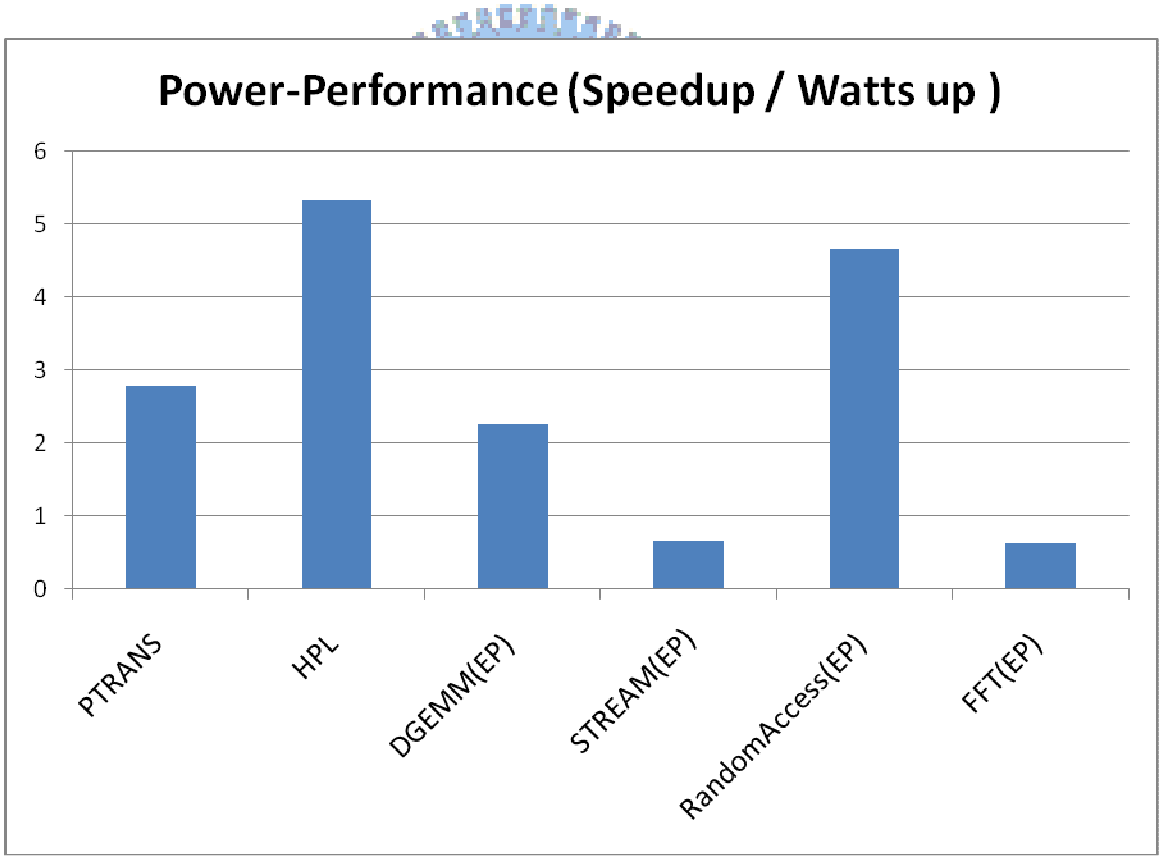


圖 3-8、效能與功率增加比例

3-6 實驗結果分析

就整體而言，當伺服器使用 CSX 加速器後用電負載增加 37Watts，應用在 HPCC 效能表現上，以執行時間來計算效能提升 27%，並且經由瓦特表上所量測到功率消耗只增加 8.5%。

因此 CSX 加速器對於計算密集程式(HPL、DGEMM&PTRANS)在效能上可提供與伺服器相當的結果；協同平行序列計算下最多可增加於單獨使用伺服器 1.78 倍效能，因此該加速器非常適用於密集矩陣的計算模式。

在 HPCC 其中 Stream 的結果僅有 0.04GB/s，由於伺服器與加速器間連結效能為 PCI-Express x8Lanes 理論值為 4GB/s，然而 STREAM 實際效能落差卻與該連結介面相差近百倍之多，推論主要為 CSX 架構上之 mono unit 與 poly unit 之間的傳輸效率造成效能上之瓶頸，因此對於僅記憶體搬移之運算類型使用 CSX 加速器，所增加之效能上相較伺服器並無太大幫助。

第四章 CSX 加速器之建議改善方向

基於我們前面所研究與實驗之結果，在加速器之設計與使用上有以下幾點改良方向與建議：

4-1 改善加速器內部資料傳遞

根據實驗結果，在使用加速上最住要之瓶頸為加速器內部資料在傳遞過程，加速器內之 DRAM 記憶體使用共享示匯流排，其頻寬分散於每個 PE 後，平均對每一個 PE 可提供 45MB/s 效能。

因此，我們建議以下幾個方式來增進 CSX 加速器之使用效能[22]：

4-1.1 群組化 PE 架構，以增加效能：

如圖 4-1 所示，Mono Unit 之 DRAM 透過 CCB(ClearConnect Bus)將資料由 DMA(Director Memory Access)控制依序將資料傳入每個 PE 當中，CCB 之傳遞效能為 3.2GBytes/Sec，因此 96 個 PE 傳遞完成，根據圖 2-3 所示，平均每個 PE 效率約為 30Mbytes/Sec。

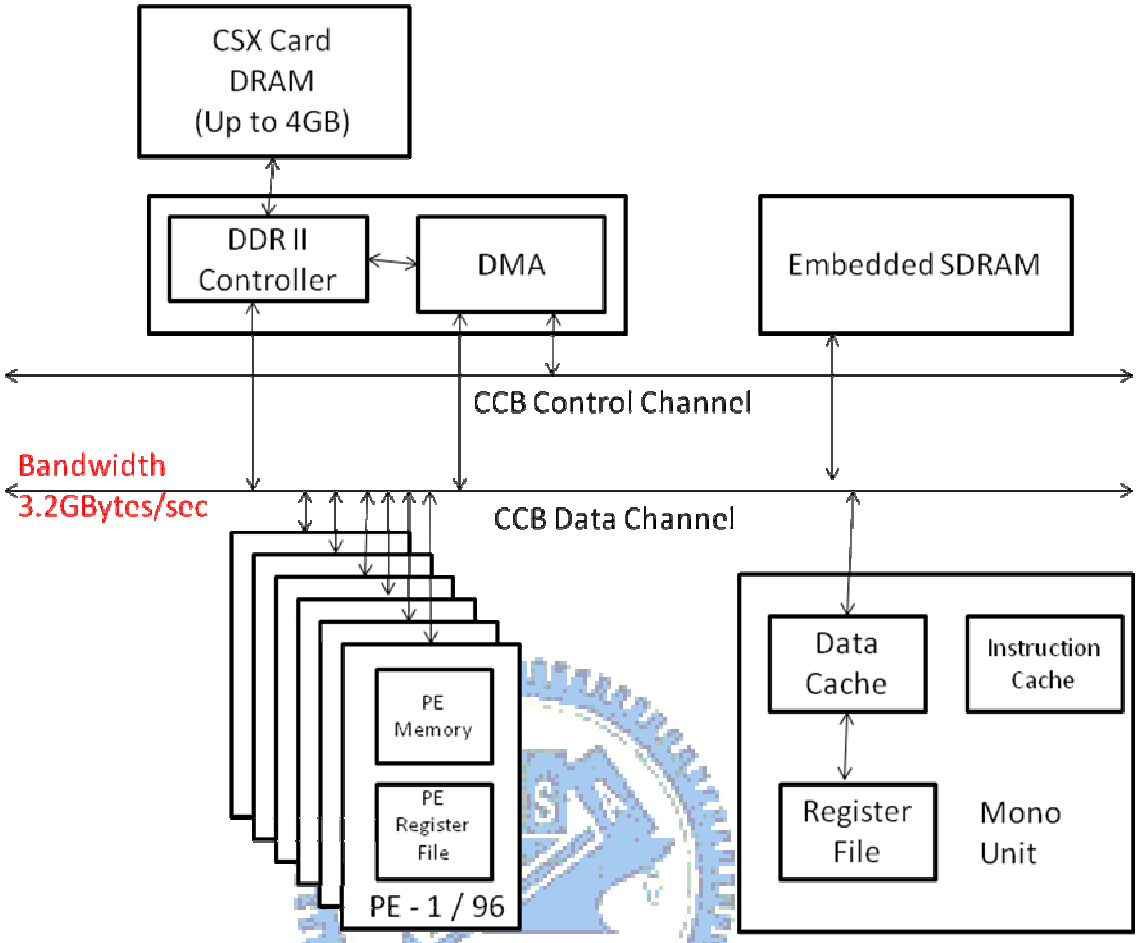


圖 4-1、記憶體資料存取區塊

將數個 PE 單元群組化(Grouping) 使得每個群組資料每次的傳遞效能提升數倍(n times)，我們將每一個群組化後之 PE 群稱為 GPE(Grouping Process Elements) 。每個 GPE 內部利用 PE 內部資料路徑(Swazzle Path)執行管線化(Pipelining)指令。

在架構上，每個 PE 由原來 SIMD(Single Instruction Multiple Data)方式轉換為 SPMD(Single Program Multiple Data)，即意味著原來在 96 個 PE 資料傳遞完成並同步後，所有 PE 指令在一次指令執行後傳回，並同步資狀況下。在 GPE 以管線方式執行多道指令所形成之程式。則使得資料傳遞提升數倍，並且一次可進行數道指令，如圖 4-2 所示。

我們試著以每 8 個 PE 單元群組化計算，每個 GPE 之效能提升至 240 MBytes/Sec。對於記憶體存取之效能預計亦可提升 8 倍。

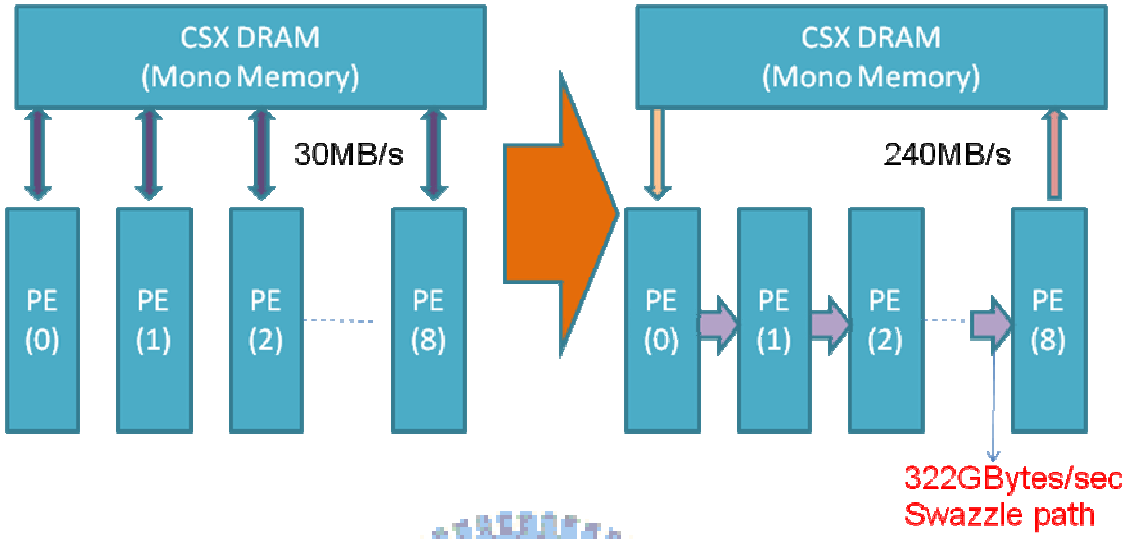


圖 4-2、群組化 PE 示意圖

4-1.2 提升 DDR II 及 CCB 工作頻率

對於每個 PE 所能使用到之頻寬主要取決於，CSX DRAM 使用之工作頻率除以總 PE 個數。因此我們實驗平台所採用之 CSX600 e620 內，DRAM 工作頻率為 400MHhz，整體可提供之效能為 3.2GHz，平均每個 PE 可分配 30Mbytes/Sec。

就以我們所查到之參考資料所示，以一條 512Mbytes 之 DDRII SDRAM ECC 記憶體工作頻在 400MHz 所需功率為 1.825Watts，若換成 DDRII ECC 工作頻率為 800MHz 所需功率為 2.025Watts。由此我們可進一步計算，加速器內之記憶體存取效能增加一倍，所需功率使用增加為約 10%。因此對於 CCB 控制電路工作頻率亦可相同推論之。[23]

再者我們結合 4-1.1 所提出之 GPE 架構，將可大幅增加加速器對於記憶體單元之效能。

4-2 稀疏矩陣求解法(Sparse Matrix Solver)功能應用

有鑑於高效能計算領域中，主要在處理矩陣方面之運算。其中，又可分為密集矩陣求解(Dense Matrix Solver)運算及稀疏矩陣求解(Sparse Matrix Solver)運算。對於真實狀況上大多數高效能計算所處理問題為稀疏矩陣資料型態。

於本研究過程中，HPCC 程式計算過程使用之計算資料型態為密集矩陣。同時，加速器函式庫 CXSL 目前所提供之功能亦為針對密集矩陣計算上使用。

主要之差異點為對於稀疏矩陣在運算過程中，僅對於非零（None-Zero）數值進行計算。因此，以目前本研究平台所提供之計算功能上，對於處理稀疏運算過程中，將有許多結果為零之不必要的計算。

例如：以下 A(9X6)矩陣內，顯示空白元素內之數值為零，其他則為非零數值。(圖 4-3)

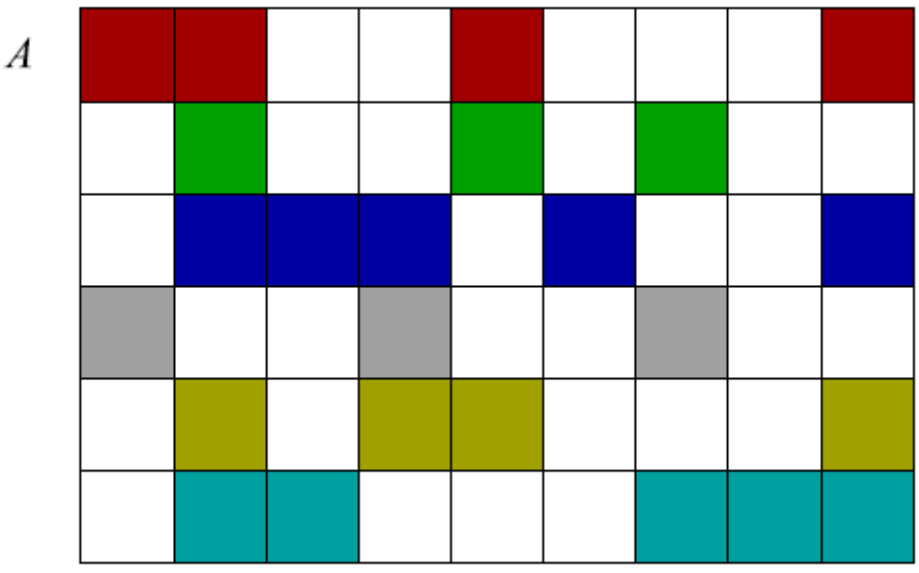


圖 4-3、稀疏矩陣

經壓縮後，將矩陣中非零元素取出，以一維陣列方式表示(圖 4-4)。在計算過程中，僅處理非零元素，以減少不必要結果為零之運算，進而使運算單元資源的利用更有效率；同時，亦可節省對於記憶體空間的使用。

在效能上，因資料處理量的減少，進而達到加速的效果。

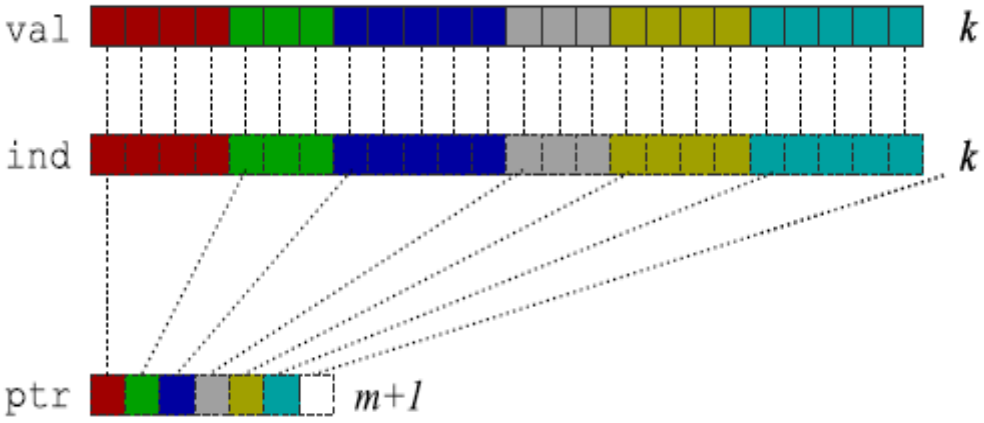


圖 4-4、稀疏陣列轉換後資料格式

以本實驗平台所採用之 Intel MKL 數學函式庫中，對於稀疏矩陣採用 PARDISO 方式對於 Ax=b 方程式求解。步驟如下：

1. 矩陣重排與符號分解(Reordering and Symbolic Factorization)：PARDISO 根據不同陣列類型，計算不同類型的行列交換矩陣 P 與對角矩陣 D，對 A 矩陣進行交換重排。獲得新的矩陣 A=PDAD’P’分解後將包含盡量少的非零元素。
2. 矩陣 LU 分解：對於 PDAD’P’進行 LU 分解。
3. 方程式求解與疊代(Iterative)：根據 LU 分解結果，求解方程式。對於結果若有進步精密度上的需求，使用疊代法進一步提高精密度。
4. 疊代結束。

4-3 提供 Fortran 語言之應用程式化界面

CSX Compiler(Cn) 可相容於目前之 C 或 C++ 程式編譯器，但在高效能計算領域傳統上，迄今仍有大多數應用程式使用 Fortran 語言來編寫，甚至於 Fortran 77 標準語法內連指標資料結構都尚未提供。

然而 CSX 加速器上函式庫及副程式之參數傳遞上，係使用指標與物件資料結構。對於程式開發者而言，需要額外將使用於 CSX 加速器上之程式，再增加轉換後才可提供於Fortran程式呼叫，因此提供 Fortran API(Application Programming Interface)對於高效能計算領域程式開發者而言在參數傳遞方式上，可縮減原來 CSX Cn 編譯器需要另外轉換使用物件呼叫功能的機制，進而減少程式編寫之工作量。

4-4 編譯器自動化資料平行

我們已知 CSX 加速器之 CSXL 函式庫為部分 BLAS 函式庫功能，其運作原理即為將原有伺服器上數學函式庫，同時運算於伺服器與加速器上。

目前 CSX 加速器方面，對其程式撰寫方式與標準 C 或 C++語法相容，在資料型態上，以資料結構定義為 Mono 與 Poly 識別為該變數使用在加速器。其計算方式，將原來使用於迴圈內之運算轉為 SIMD 方式運算，對其運算之語法及功能與原來相同，主要在資料處理的平行化。

由此，我們可進一步對於目前常用於程式內可平行化之標準，例如：OpenMP、TBB (Threading Building Blocks)或其他標準[25]。在編譯過程中，直接將欲平行化之程式部分，亦同時轉換為適合 CSX 加速器執行之目標碼(Object Code)。進而省去需另外針對加速器程式開發之過程，範例如下：

```
#pragma omp parallel for clause clause ...
for (i = e1; i op e2; i = i ops incr) { stmts; }
```

轉換為使用於 CSX 加速器之語法：

```
{ poly int i_p = e_1;
  for (i_p = i_p op_s get_penum() * incr; i_p op e_2; s_1_p )
    { stmts_p; }
}
```

流程示意圖，如圖 4-5 所示。

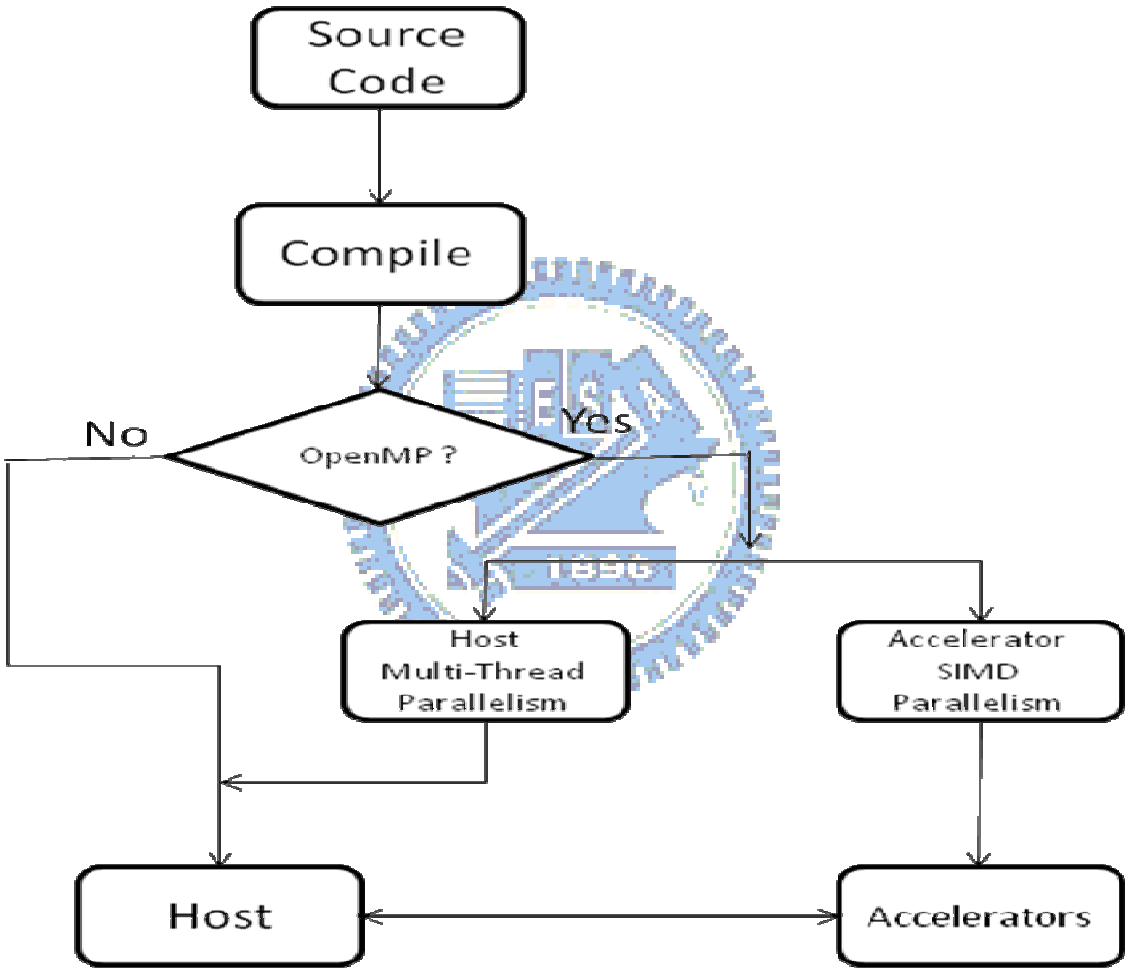


圖 4-5、編譯自動化流程

OpenMP 在編譯器過程，利用在程式中註記(Pragma)方式將所指定程式區塊，進行多個執行序列將迴圈內資料平行運算。相對於加速器使用上，在編譯器的設計上，對於 OpenMP 之註記部份將迴圈內運算之變數資料型態，自動轉換可於加速器辨識使用之 Poly 與 Mono 資料型態。以達到編譯自動化的目的。

第五章 總結

5-1 經驗

5-1.1 CSX 加速器應用經驗

目前市面上，採用加速器之應於在於利用其自身所內含大量陣列形式之運算單元，用以輔助處理大量資料之運算。在本篇研究中，加速器對於高效能計算應用上，採用混合式運算後，對 HPCC 程式之執行上，各個程式皆有輔助效能增加之效果。

在使用上，CSX 加速器有別於其他加速器產品之地方，主要在於可進行伺服器與加速器同時運算。不同於其他加速器將伺服器與加速器分開方式運作。由於使用 CSX 加速器運算時，與伺服器運算之比例需由手動調整。同時，在不同硬體架構或程式執行時，需隨時做參數調整以達到最佳效能。

透過 HPCC 執行後，較可確定計算複雜度大於 N^2 時，效能可獲得較明顯之加速器輔助增加之功效。

另外，於本研究過程中，我們亦嘗試將分子動力學上常用之應用程式：Gaussian03 及 VASP 其中呼叫 BLAS 部份，使用 CSXL 函式庫取代原有 BLAS 功能，結果卻無法在 CSX 加速器上使用，因此對於目前 CSXL 所提供之功能上，仍有許多需要再增加的地方。

且其他加速器所提供之函式庫，大多與目前伺服器所使用之函式庫不同。對於程式開發上，須配合某特定產品之加速器修改原有程式架構。例如：Nvidia GPU 產品開發環境為使用其產牌自身標準之 CUDA(Compute Unified Device Architecture)、ATI GPU 則採用該產牌之標準 Brook+ 等。對於使用者原有程式皆須針對使用之加速器修改原始碼。

5-1.2　加速器使用上限制

有鑑於加速具有為數眾多之計算單元，因此在記憶體之使用上，加速器多半於伺服器外另獨立擁有加速器使用之記憶體，系統運作時伺服器與加速器之資料透過 PCI_Express 介面傳遞彼此記憶體資料，效能上較伺服器直接與記憶體傳遞上，頻寬與延遲時間較差。

更進一步，當加速器記憶體資料傳遞到為數眾多運算處理單元時，加速器上每個運算單元可分配到之資料頻寬，常常成為造成效能上之瓶頸。如圖 5-1 所示。

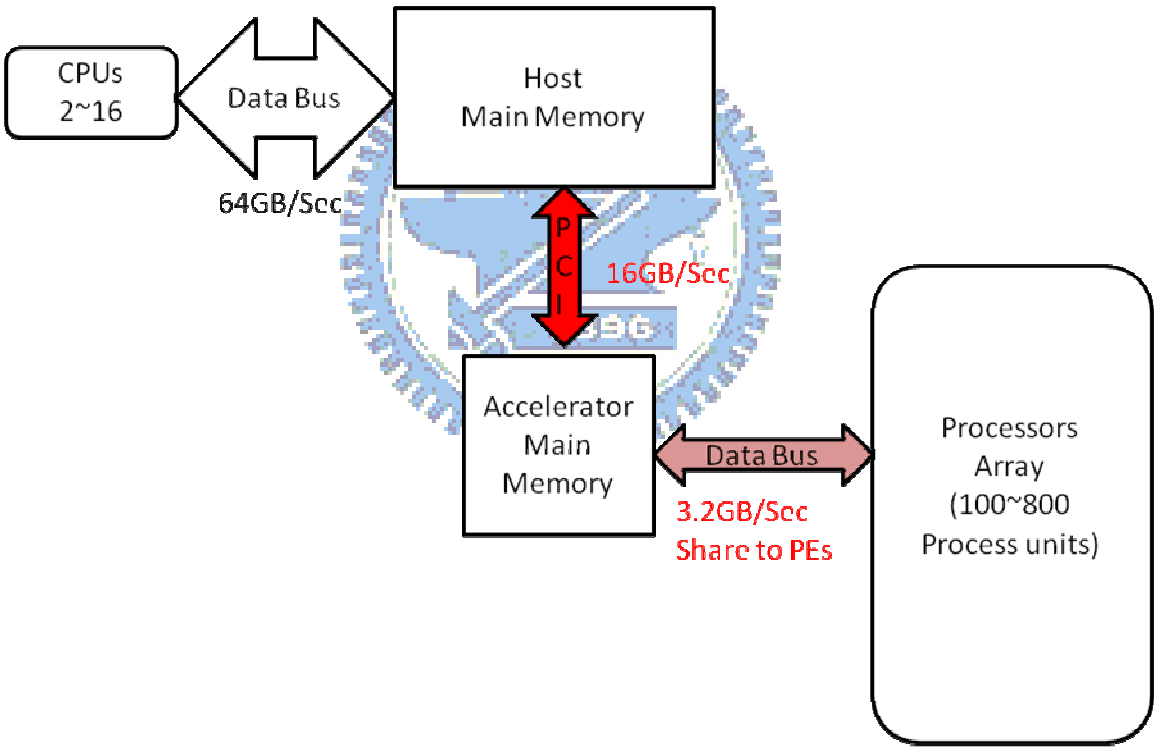


圖 5-1、伺服器與加速器記憶體連接圖

因此使用加速器時，最常需要注意地方為如何節省資料傳遞時所花費的時間。

5-1.3 CSX 加速器應用

藉由本研究實驗，可知建議使用 CSX 加速器之應用，在於計算複雜度在 O(N^2)以上之矩陣雙精度浮點運算可獲得效能上之增益。意即為符合 OpenMP 之兩層以上迴圈運算。於執行時，將最內層迴圈解開(unrolling)以 96 個 SIMD 方式運行。

於本研究實驗系統上，計算過程中伺服器與加速器資料分配比例最佳為 40:60，同時伺服器端佔用一顆處理器核心作為與加速器之資料通訊。

5-2 未來研究方向

5-2.1 標準化加速器語言研究

有鑑於各類加速器產品之多樣化，使用上因應供應商所提供之軟體開發環境標準不同。需特別針對該產品開發其特別之程式才可使用。

因此近來有逐漸將不同加速器上，使用共同開發環境之趨勢。例如：OpenCL (Open Computing Language)標準，為針對異質性平行程式開發環境所建立之一般用途目標(General-purpose)程式語法。依據其標準程式可應用於不同加速器上。

5-2.2 評估使用各種型態加速器，建置計算量在 PetaFlops 等級之系統架構

PetaFlops 為每秒可進行百兆(10^15)次浮點運算之單一高效能計算系統，亦為 Top500 趨勢在計算量上常見之目標。對於目前已知的世界排名前十大超級電腦中有三套採用 IBM Cell 處理器作為加速器系統。另外，亦有採用 Nvidia 及 Clearspeed 產品進入前五百大排名。

並有鑒於加速器計算能力已超越目前處理器效能之摩爾定律成長趨勢，因此利用不同加速器，作為提供計算主要能量之系統架構演討。

5-2.3 評估多加速器系統效能

在本研究中，我們僅使用單一 CSX 加速器卡進行研究。未來我們將進一步在單一伺服器下，使用多張 CSX 加速器卡片進行計算。藉以研究對於效能之影響。

參考文獻：

[1] Top500 Supercomputing Sites, http://www.top500.org

[2] Gordon E. Moore "Cramming more components onto integrated circuits" Electronics, Volume 38, Number 8, April 19, 1965

[3] Texas Advance Computing Center , Ranger User Guide , http://www.tacc.utexas.edu/services/userguides/ranger/

[4] Los Alamos National Laboratory of the Department of Energy's National Nuclear Security Administration , http://www.top500.org/system/9485

[5] Piotr Luszczek, David Bailey, Jack Dongarra, Jeremy Kepner, Robert Lucas, Rolf Rabenseifner, Daisuke Takahashi "The HPC Challenge (HPCC) Benchmark Suite", SC06, Tampa, Florida

[6] Jim Bovay, Brent Henderson , Hsin-Ying Lin, Kevin Wadleigh ,"Accelerators For High Performance Computing Investigation ", High Performance Computing Division Hewlett-Packard Company.

[7] Xilinx, Inc , http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex5/index.htm

[8] Olaf O. Storaasli "Accelerating Genome Sequencing 100X with FPGAs" Future Technologies Group, Oak Ridge National Laboratory

[9] "SYNERGISTIC PROCESSING IN CELL'S MULTICORE ARCHITECTURE", IEEE Computer Society MARCH-APRIL 2006

[10] Owens, J.D.; Houston, M.; Luebke, D.; Green, S.; Stone, J.E.; Phillips, J.C.; "GPU Computing", Proceedings of the IEEE Volume 96, Issue 5, May 2008 Page(s):879 - 899

[11] Clearspeed Technology Inc. http://www.clearspeed.com

[12] BLAS (Basic Linear Algebra Subprograms) , http://www.netlib.org/blas/

[13] LAPACK -- Linear Algebra PACKage , http://www.netlib.org/lapack/

[14]Intel Math Kernel Library , http://www.intel.com

[15]AMD Core Math Library (ACML), http:// www.amd.com/**acml**

[16] Vivek Sarkar "Seminar on Heterogeneous Processors" , Department of Computer Science Rice University

[17]周朝宜 張西亞 王順泰 陳德民 ,"HPCC 做為叢集電腦採購指標之研究", 國家高速網路與計算中心

[18] STREAM: Sustainable Memory Bandwidth in High Performance Computers ,http://www.cs.virginia.edu/stream/

[19] M. Snir, S. Otto, S. Huss-Lederman, D. Walker and J. Dongarra, MPI: The Complete Reference, MIT Press, Cambridge, MA, 1996

[20] Con Brandley and Benedict R. Gaster "Exploiting Loop-Level Parallelism for SIMD Array using OpenMP" ClearSpeed Technical Plc.

[21] Toshio Endo and Satoshi Matsuoka, "**Massive Supercomputing Coping with Heterogeneity of Modern Accelerators**" , Tokyo Institute of Technology/JST, Japan , Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium

[22] Nishikawa, Y.; Koibuchi, M.; Yoshimi, M.; Miura, K.; Amano, H.;" ***Performance Improvement Methodology for ClearSpeed's CSX600"*** , Parallel Processing, 2007. ICPP 2007. International Conference on 10-14 Sept. 2007 Page(s):77 – 77

[23] Kingston Memory Module Specification, http://www.kingston.com

[24] Samuel Williams_†, Leonid Oliker_, Richard Vuduc§, John Shalf_, Katherine Yelick_†, James Demmel ,; "Optimization of Sparse Matrix-Vector Multiplicationon Emerging Multicore Platforms" , SC07 November 10-16, 2007, Reno, Nevada, USA

[25] Yang, Jianfeng; Zheng, Hong; Xie, Yinbo; Wang, Jolly; Bao, Nick; "Adding TBB Contents to the Multi-Core Related Curriculums" , Intelligent Networks and Intelligent Systems, 2008. ICINIS '08. First International Workshop on 1-3 Nov. 2008 Page(s):685 - 688