以 Web-Based 之平行計算環境之設計與實作

Design and Implementation of Web-Based Parallel Computing Environments

*黃圳柏　**邱基峰　*黃文祥　**謝錫堃

*國立高雄應用科技大學電機工程系
**國立成功大學電機工程研究所
Email: adrian@wshlab2.ee.kuas.edu.tw

摘要

本文提出一建構在 Web Base 之平行處理編譯器。此平行處理編譯器，可提供使用者快速編譯之功能，特別是針對需要大量數學運算的使用者。因此使用者只要經由網際網路將所欲執行原始碼上傳至 Web Compiler，Web Compiler 會以平行處理方式將使用者所欲執行原始碼快速執行完畢。

關鍵詞：平行處理、Web-based 編譯器、MPI

一、前言

目前已有許多系統相繼被提出以克服傳統單一處理器執行效能之瓶頸。為了解決此問題，平行處理是目前最為廣泛使用方法。

對於數學家、物理家以及專業程式設計師等等，常常需要使用電腦做大量數學運算，然而這些大量數學運算，執行時間動輒好幾個小時，甚至需要花費一天以上時間，方能將這些運算執行完畢。有鑑於此，本文利用 MPI [8] (Message-Passing Interface)實作出以 Web 為基礎之平行處理編譯器，只要數學家、物理家以及專業程式設計師等等，經由網際網路上傳所欲執行的原始檔(*.c)，此編譯器會自動將執行工作細分給底下多台電腦執行或在一台電腦同時執行多個程序(process)，以平行處理方式將使用者所欲執行的原始檔，快速執行完畢，並將執行結果回應給使用者。

本文第二節為 MPI 之介紹；第三節為系統架構；第四節為實作結果與效能分析；最後本文結論。

二、MPI 之介紹

MPI(Message-Passing Interface)由 MPI Forum (MPIF) 所提出，MPI 為一套標準化之 Message-Passing 平行函式庫，可提供程式設計者平行處理之功能。目前 MPI 可支援 C、C++以及 Fortran 等語言，乃至於 Java 語言[4]MPI 也能夠支援。MPI 可說是目前相當普遍被使用之通訊函式庫。

MPI-1 是最早被 MPIF 提出，MPI-1 主要研究方向集中於 Point to Point Communication Interface 上，因此有關這方面之研究，已有相當成熟度。然而對於 dynamic process control 以及 process management 等議題，MPI-1 並未提出詳細說明。當然 MPI-1 為求迅速，引用了很多現有 Message Passing Systems 上已有的觀念，如 Intel NX/2，Express，nCUBE Vertex ，P4，PARMACS。於 1995-96，MPIF 提出 MPI-2 以補強 MPI-1 不足之地方，因此 MPIF 於 MPI-2 加入 Dynamic Process Management、Parallel I/O 以及 Remote Memory Access 等等之功能。

MPI 的 Programming 方式以 Message-Passing Function Call，屬於 Task Parallelism，Task Parallelism 具有 Unstructured Synchronization and Communication 與 Multiple Threads of Control 等等之特性。

三、系統架構

圖 1 為系統架構圖，此系統主要是建構於 Linux 下，並於每台 PC 上安裝 MPI，以讓每台 PC 具有參與平行處理計算之功能。在此，本文建構五台具平行處理之 PC，詳細實作結果將於第四節作詳細描述。在此五台 PC 中，我們選擇 PC1 為 Web server，其主要提供使用者可上傳所欲編譯原始檔並將編譯與執行之工作細分給所欲參與 PC 數目。因此我們於 PC1 建構一 Web 主畫面，並且撰寫一 CGI 程式，以接收使用者所上傳原始檔之資料，並將此原始檔加入 MPI function call，最後執行由 MPI 所提供之執行檔(mpirun)。當此程式執行後，PC1 即會執行工作細分給所欲指定之 PC 數目。如下所示：

[root@joker /]# mpirun -np 4 matrix

此命令即表示將有四台 PC 將參與此次平行處理之工作。然而，當設定所欲參與 PC 數目多於實際上之 PC 數目時，則某一 PC 將會執行一個以上 process。

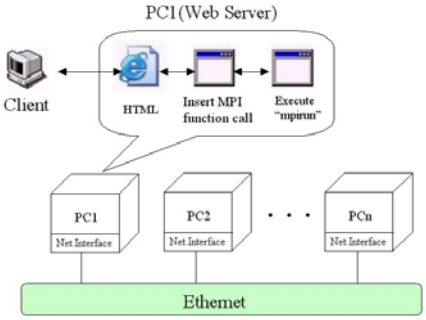

圖 1. 系統架構圖

四、實作結果與效能分析

此章節將細分兩個部分討論。一為針對所實作之 web compiler，做詳細討論並提出此 web compiler 具備哪些功能。二為平行處理效能分析，簡單寫二個程式測試當平行 PC 數目增加時，計算程式執行所耗費時間。此兩個程式分別為矩陣相乘與計算 π。

A、Web Compiler

Web Compiler 主網頁如圖 2 所示，圖中使用者可直接在此網頁撰寫 C 語言程式，或者可上傳所欲執行 C 語言程式原始碼，一旦上傳完畢之後，其原始碼將顯示在此網頁中，使用者可依其需要再行修改。若欲執行該程式碼，即可點選此網頁左下方之"compile"按鈕，即可編譯。圖 2 為撰寫九九乘法程式之範例，其編譯結果如圖 3 所示。

圖 2. Web Compiler 主網頁

圖 3、Web Compiler 執行結果

當然此 Web Compiler 也會通知編譯失敗回報，即代表此 C 語言原始碼有某些語法錯誤。圖 4 為一範例。此範例中，我們於 printf()函式中引用先前未宣告變數 k 以及沒有在 printf()函式結尾處加上分號，因此在編譯時會發生失敗。當編譯失敗發生時，Web Compiler 會將這些錯誤訊息，顯示在網頁上，以通知使用者，某些地方語法發生錯誤或者引用先前未定義變數等等錯誤，因此使用者即可直接在網頁上修改 C 語言原始碼錯誤之地方，並且順利將此原始碼順利編譯成功。圖 5 為圖 4 編譯失敗之結果。

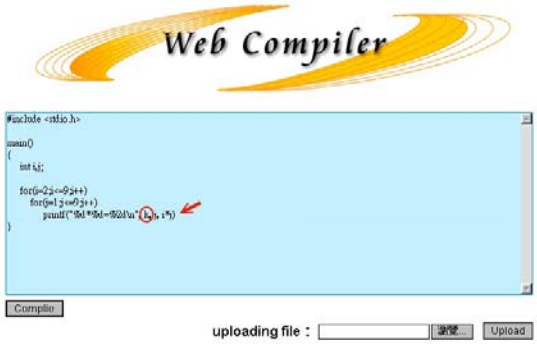

圖 4. 錯誤範例

圖 5. 編譯失敗回報

B 、平行處理效能分析

在此平行處理效能分析章節中，將簡單撰寫二個程式測試當平行 PC 數目增加時，其程式執行所耗費時間有何變化。

512*512 矩陣相乘—矩陣相乘中若矩陣大小越大其所計算時間越久，然而在計算過程當中有許多項目彼此獨立不相依，因此可利用平行計算其結果。此例中每個 CPU 將計算矩陣 A*B=C 當中自行負責之部分，最後將結果矩陣 C 傳回第一顆 CPU 並且累加存放在 D 矩陣得其結果。每個 CPU 所負責的部分為各 CPU 依序讀取 A 矩陣之行且與 B 矩陣相乘得其結果存放 C 矩陣。其示意圖如下：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & \cdots & a_{2N} \\ \cdots & \cdots & \cdots & \cdots & \\ \cdots & \cdots & \cdots & \cdots & \\ a_{N1} & a_{N2} & \cdots & \cdots & a_{NN} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \cdots & \cdots & b_{1N} \\ b_{21} & b_{22} & \cdots & \cdots & b_{2N} \\ \cdots & \cdots & \cdots & \cdots & \\ \cdots & \cdots & \cdots & \cdots & \\ b_{N1} & b_{N2} & \cdots & \cdots & b_{NN} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & \cdots & c_{2N} \\ \cdots & \cdots & \cdots & \cdots & \\ \cdots & \cdots & \cdots & \cdots & \\ c_{N1} & c_{N2} & \cdots & \cdots & c_{NN} \end{bmatrix}$$

$a_{11} \ldots a_{1N}$ 為 CPU0 計算之乘數，B 矩陣為被乘數，因此可得 $c_{11} \ldots c_{1N}$ 值。依此類推，每個參與執行之 CPU 分別計算 C 矩陣之行。最後在將它們相加得其結果。此範例中，我們將矩陣大小設為 512*512。其結果如表 1 與圖 6 所示，圖 6 縱軸代表矩陣運算所耗費時間，橫軸代表參與平行處理 CPU 數目。

CPU 數目	1	2	3	4	5
運算時間(sec)	32.07467	17.2 2833	12.08 3532	9.92 6985	7.325 573

表 1. 512*512 矩陣相乘運算數據表

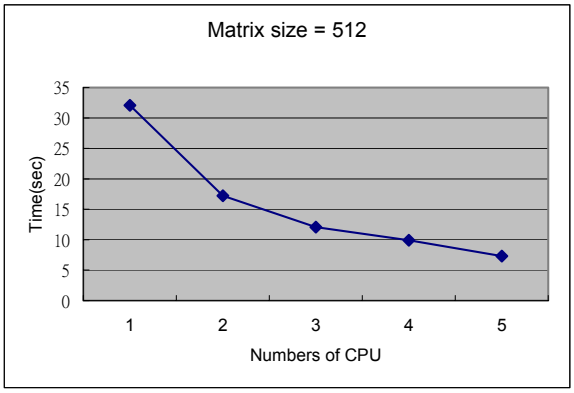

圖 6. 512*512 矩陣數據曲線圖

理想上當參與運算 CPU 數目增為兩倍時、時間能減至二分之一。然而由此例子中可得知 CPU 彼此間通訊也花費不少時間，因此無法達到理想之狀況。

PI 值計算—PI 值為一無理數，因此計算 PI 值乃評估計算機效能最常使用之方法，我們利用計算 PI 值來探討平行計算之效能。圖 7 為上傳 pi.c 原始檔之畫面，其 PI 值運算值如圖 8 所示。其運算所耗費時間數據與數據曲線圖，如表 2 以及圖 9 所示。

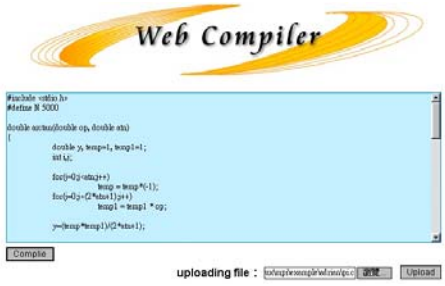

圖 7 上傳 pi.c 原始檔

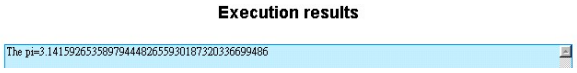

圖 8　π 值運算結果

CPU 數目	1	2	3	4	5
運算時間(sec)	2.18 2818	1.09 2269	0.729 050	0.66 4686	0.543 454

表 2　π 值運算數據表

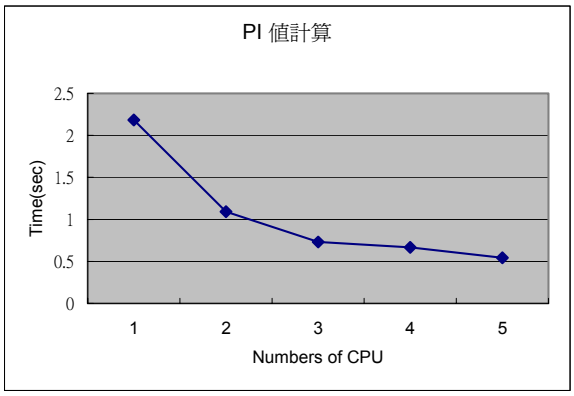

圖 9　π 值數據曲線圖

PI 值為一無理數，因此用來分析平行效能為一可行方法，由此例子可得知參與運算 CPU 數目越多其執行時間越短，然而因其 CPU 通訊時間影響，其效能亦受影響，由實驗結果得知當 CPU 增加時，其 CPU 通訊時間亦漸漸追近計算時間，因此效能上並無法達到倍數成長。

5. 結論與未來工作

本文中，我們提出以 Web 為基礎之 compiler，使用者經由網際網路上傳所欲執行 C 語言原始檔，此 Web Compiler 會以較快速執行速度將這些工作量大之程式執行完畢，意即透過平行處理方式來達成。

在未來研究中，我們將著重於多語言之 Web Compiler，意即使用者所上傳之原始檔屬於 C++、JAVA 等等，並非 C 語言。此 Web Compiler 也可順利編譯與執行成功，並將其結果回傳給使用者。

參考文獻

[1] Daesuk Kwon, Sangyong Han, Heunghwan Kim, “MPI backend for an automatic parallelizing compiler”, Parallel Architectures, Algorithms, and Networks, pp. 152 –157,1999.

[2] Sergei Gorlatch, “Toward formally-based design of message passing programs”, Software Engineering, IEEE Transactions on Volume: 26 Issue: 3, pp.276-288, March 2000.

[3] Ewing Lusk, “Programming with MPI on Clusters”, Cluster Computing Proceedings, 2001 IEEE International Conference, pp.360-362, 2001.

[4] Ricky K.K Ma, Cho-Li Wang, Francis C.M. Lau, “M-JavaMPI: A java-MPI binding with process migration support”, Cluster Computing and the Grid 2nd IEEE/ACM International Symposium CCGRID2002, pp.240-247, 2002.

[5] W.Gropp, E.Lusk, and A.Skjellum, “Using MPI：Portable Parallel Programming with the Message-Passing Interface, 2nd edition”, MIT press, Cambrige, MA, 1999.

[6] R. Butler, W.Gropp, and E.Lusk, “Components and interfaces of a process management system for parallel programs”, Parallel Computing, 2001.

[7] Message Passing Interface Forum, “MPI：A Message-passing Interface standard”, International Journal of Supercomputer Application, 1994.

[8] The Message-Passing Interface (MPI) Standard, http://www-unix.mcs.anl.gov/mpi/.

[9] Rémy Card and Frank Mével, “the LINUX KERNEL book”, WILEY, 1998.

[10] Richard Stones and Neil Matthew, “Beginning Linux Programming”, WROX, October 1999.

[11] William Stallings, “Operating Systems, Fourth Edition”, Prentice Hall, 2001

[12]高速計算環境, http://www.nchc.gov.tw/chinese/07_publication/nchcs/Other/V3N1/p23.html