行政院國家科學委員會專題研究計劃成果報告

一個基於知識庫技術的高效能平行編譯器(二)

在多處理機系統上一個支援執行時期平行化的平行編譯器

A High Performance Parallelizing Compiler using Knowledge-Base Techniques (II)

A Parallelizing Compiler with Run-Time Parallelization Supports for Multiprocessor Systems

計畫編號：NSC87-2213-E009-023

執行期限：86年8月1日至87年7月31日

主持人：曾憲雄　　國立交通大學資訊科學系教授

一、中文摘要

關鍵詞：多處理機系統、平行編譯器、執行時期平行化、迴圈轉換、偵測階段、執行階段

在上一年計畫中，我們製作出一個 Source Program Behavior Analyzer的雛形即是Practical Parallel Loop Detector 用來偵測程式中可以平行化的迴圈。此一 Analyzer 採用我們於先前研究計畫所製作出的 K-Test 作為 Data Dependence Testing。接著我們使用專家系統的技術來解決平行編譯器上迴圈的排程問題。由於程式中迴圈的特性不一，目前的方法中，沒有一種能夠很有效率的解決排程問題。因此，我們發展一個基於知識庫的平行迴圈排程系統(KPLS)，依照迴圈的特性和資訊加以分析，利用現有的排程演算法，推理一個最適合的排程法，使得排程後的迴圈能在多處理機系統上有效的執行。

我們在研究中發現，迴圈在一程式中存在有大量的平行度，為了此程式平行化，平行編譯器利用靜態資料相依性分析來獲得迴圈的平行度。然而，有些迴圈則無法於編譯時期取得資料相依性的資訊。例如，在稀疏矩陣計算上，陣列述語內通常包含了間接陣列或函式，便無法完成靜態資料相依性分析。故便保守的程式循序的執行，而犧牲了潛在的平行度。因此，在本計畫中，我們提出了一個兩階段(偵測階段及執行階段)的執行時期平行化方法於執行時期擷取出迴圈中潛在的平行度。偵測階段經由建立一 DEF-USE 表而決定出可平行執行的迴圈輪替集合-波前，而此偵測階段本身可以被完全的平行化而減少因決定波前所照成的額外負擔。此外，我們仍然要持續的加強與改進 Parallel Loop Detector 與 KPLS 的功能。以上的研究，在我們已發展的可移植式平行編譯(PFPC)中實際製作出來，並可提昇編譯器產生有效率的執行碼。

二、英文摘要

Keywords : Multiprocessor Systems, Parallelizing Compiler, Run-Time Parallelization, Loop Transformation, Inspector, Executor.

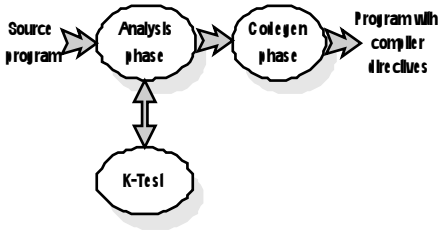
In order to parallelize a sequential loop, a parallelizing compiler must perform a parallel schedule of the loop iterations based on the static data dependence analysis. Some loops, however, may contain the potential parallelism that can not be detected at compile-time. For example, in sparse matrix computations, array subscripts often involve indirection arrays and thus defy static analysis. Conservatively, the loop iterations in such examples will be executed sequentially. Motivated by these concerns, a run-time technique based on inspector-executor scheme is proposed to find available parallelism on loops in this project. Our inspector will be able to determine the wavefronts by building a DEF-USE table. Additionally, the process of inspector for finding the wavefronts can be parallelized fully without any synchronization. Our executor will be able to perform the loop iterations concurrently. For each wavefront in a loop, the auto-adapted function is used to get a tailored thread number rather than using fixed thread number for execution. Experiments will be made to show that our new parallel inspector can handle complex data dependency patterns and reduce itself execution time obviously. Besides, the new partitioning strategy for executor can also improve the performance of run-time parallelization obviously. In this year of project, we will also improve the performance of PPD and KPLS to achieve our exception.

As a final goal, a high-performance and portable FORTRAN parallelizing compiler for shared-memory multiprocessor systems will be developed.

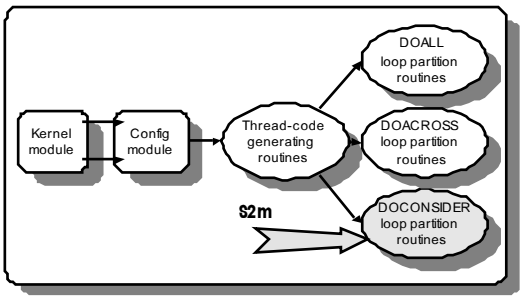
三、計畫緣由與目的

由於 VLSI 技術快速發展使的微處理器的計算效能大幅提昇，目前的個人電腦以能有效應付一般的文書處理及簡單的計算工作。但是對於一些需要大量計算的工作，例如 Grand Challenge 等問題，單一微處理機的計算能力又顯得不足。倘若一部平行電腦之處理器個數可由數個擴充到數千個，使用者可依目前的經費，暫時採購具較少處理器個數的平行電腦，往後依經費狀況及需要再加以擴充。而一個具延展性的平行程式，可依處理器數目的多寡，發揮其最佳的執行效能。在處理器數目改變時，程式並不需要隨著變動，節省了程式再發展的成本。因此可研展性 Scaleable Computing 的研究，可以使整個平行電腦發揮其最佳功能，亦能保障已發展之平行程式的適用性。

平行編譯器的目的是利用平行性分析和程式分割技術自動地循序執行的程式轉換成適合在平行計算機系統上可平行執行的程式。在上一年計畫中，我們使用 Lex 和 Yacc 製作出一個 Source Program Behavior Analyzer(圖一)的雛形即是 Practical Parallel Loop Detector 用來偵測程式中可以平行化的迴圈。此一 Analyzer 採用我們於先前研究計畫所製作出的 K-Test 作為 Data Dependence Testing，並整合於我們自行製作出的可移植式 FORTRAN 平行編譯器(PFPC)，在這編譯器中，利用 Parallel Loop Detector(PPD)做為 FORTRAN 平行編譯器的前段部份，加上自行設計的 Single-to-Multiple Threads Translator(圖二)，利用來偵側出程式中可平行的部份輸入一個循序的 FORTRAN 程式，並 PPD 的輸出中可平行化的程式部分，轉換成 Mach 3.0 的 C-Threads 函式，最後再經由 GUN gcc 編譯器，配合 Mach 3.0 的 C-Threads 函式庫，產生可在多處理機系統上執行的執行碼。



圖一: The structure of PPD



圖二: The structure of S2m

接著，我們人工智慧的理念延伸以及擴充至解決迴圈排程問題。所謂平行迴圈排程問題為針對可平行的迴圈，我們之作適當的分割，再分別這些切割好的迴圈安排至各個處理器，來平行執行，如何作最佳的安排即為平行迴圈排程問題。因此我們利用人工智慧的方法設計一個適合的專家系統。依照迴圈本身的特性及現有已發展的迴圈排程演算法，作一整合，使得各種迴圈其安排的方式能更富有彈性，更符合迴圈的特質。迴圈安排的好壞在執行中(Run-Time)其需要耗費的額外代價是相當大的，所以我們嚐試利用專家系統的概念，來完成迴圈排程的最佳化。

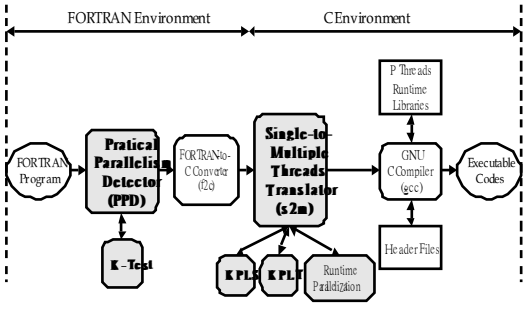
我們在研究中發現，迴圈在一程式中存在有大量的平行度，為了此程式平行化，平行編譯器利用靜態資料相依性分析來獲得迴圈的平行度。然而，有些迴圈則無法於編譯時期取得資料相依性的資訊。例如，在稀疏矩陣計算上，陣列述語內通常包含了間接陣列或函式(圖三)，便無法完成靜態資料相依性分析。故便保守的程式循序的執行，而犧牲了潛在的平行度。因此，在本計畫中，我們提出一個兩階段(偵測階段及執行階段)的執行時期平行化方法於執行時期擷取出迴圈中潛在的平行度。偵測階段經由建立一 DEF-USE 表而決定出可平行執行的迴圈輪替集合-波前。此外，此偵測階段本身可以被完全的平行化而減少因決定波前所照成的額外負擔。此一研究，在我們已發展的可移植式平行編譯(PFPC)中實際製作出來，提昇編譯器產生有效率的執行碼。

```
DO I = 1, N
    DO J = 1, I
S1:    A(3 * I) = A(I + 2) + 100
S2:    aa = aa + sin(J) * tan(J)
    ENDO
ENDO
```

```
DO I = 1, N
    DO J = 1, I
S1:   A(B(I)+J) = I+J+10
S2:   aa = aa+sin(J) * tan(J)
    ENDO
ENDO
```

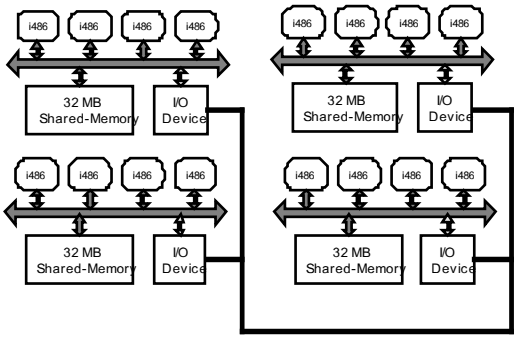
圖三: **Two loop examples for run-time parallelization.**

在本計畫中，我們承襲上面的專家系統精神，再沿用於平行迴圈的事前轉換處理，就一般的 **Program** 而言，並非所有的程式均可以平行化，在針對某些特定的 **Domain** 時，有些程式在經過適當的處理時，平行化卻可以提高，所以為了增加平行度，這前置作業是必須的，且在做過適切的轉換後，更有利於接著要做的資料相依性的測試、平行迴圈的排程和平行程式碼的轉換。未來我們應用於各種不同的 **Domain** 來加以處理，做平行迴圈的轉換 **(Parallel Loop Transformation)**，根據迴圈的資料特性和問題的範籌來建立知識庫，因為之間的變因太大，無法針對各種情況做最佳的安排，這也就是為什麼我們要採用專家系統，來完成所要的資料判定。此一研究，已在我們發展的可移植式平行編譯(如圖四)中實際製作出來，實驗結果並顯示可使編譯器產生有效率的執行碼。



圖四: **The structure of PFPC**

若增加太多的 **CPU** 於一部多處理機上，整個系統的 **Cost** 可能太高，因此有 **NUMA(Non-Uniform Memory Access Architecture)**的模式出現(如圖五)，由好幾部多處理機串接所形成的，採取這種做法一方面可以減低 **Hardware** 上的 **Cost**，另一方面又可以提高系統的 **Performance**，因此在接下來的計劃中，我們移植我們的平行編譯器至 **Cluster** 上。**Message Passing** 是其主要的 **Overhead** 來源，為了減少網路上傳輸的花費且提高資料的再使用性(**Reusable**)，此時一個好的迴圈排程方式就相當重要了，因此我們會根據資料的同質性 **(Affinity)**，來加以分析處理。另外在資料的相依性及迴圈的平行化的部份也朝向 **Cluster** 發展。



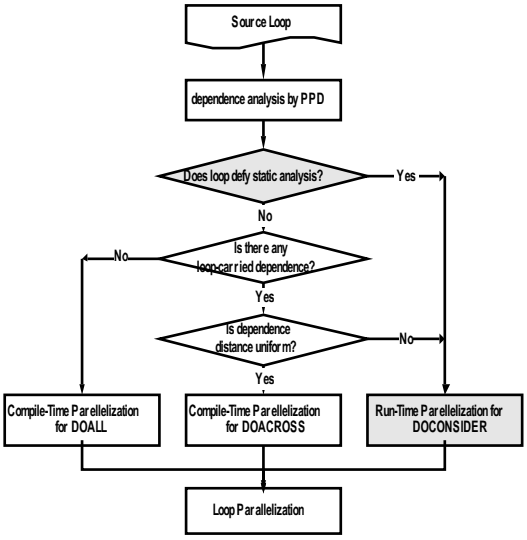
圖五: **An example of NUMA machine**

由於這個領域的研究對平行計算有非常大的影響，所以非常值得深入探討。平行編譯器的完成，有助於加速一般的計算，因此在就其未來的走向而言，可以和高速計算領域結合，減少所需花費的時間，就在國家高速電腦中心，已有訂定出相關的八大領域，如：化工化學分析氣體分子、流體力學、環保計畫的模擬等項，預計未來可以提高各項的研究成果。參與者可以培養系統整體觀的能力，畢竟軟、硬體需要充份配合，才能發揮其各自的效益，以提高總體的性能。本研究可促使多處理機之間作快速並正確的資料傳輸及交換，以及加強程式編繹時與執行時之平行化，以充份利用系統硬體資源。這些皆是高性能計算機系統關鍵組件及關鍵技術，可培養系統級的研究能力及人力，並促進我國產業的升級。

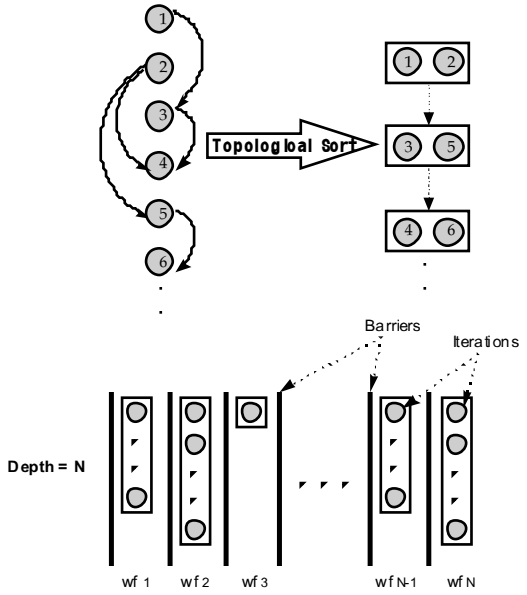
四、研究方法與成果討論

我們在研究中發現，迴圈在一程式中存在有大量的平行度，為了此程式平行化，平行編譯器利用靜態資料相依性分析來獲得迴圈的平行度。然而，有些迴圈則無法於編譯時期取得資料相依性的資訊。例如，在稀疏矩陣計算上，陣列述語內通常包含了間接陣列或函式，便無法完成靜態資料相依性分析。故便保守的程式循序的執行，而犧牲了潛在的平行度(如圖六)。因此，在本計畫中，我們提出一個兩階段(偵測階段及執行階段)的執行時期平行化方法於執行時期擷取出迴圈中潛在的平行度。偵測階段經由建立一 **DEF-USE** 表而決定出可平行執行的迴圈輪替集合-波前(如圖七)。此外，此偵測階段本身可以被完全的平行化而減少因決定波前所照成的額外負擔。這種兩階段的方法，也在我們已發展的可移植式平行編譯

(PFPC)中實際製作出來，並達到預期之研究目標。



圖六: The data flow for loop parallelization.



圖七: The structure of wavefronts

對平行編譯器的設計者來說，最理想的狀況是，無論在哪一種硬體平台上，都可以不必改變平行編譯器產生的程式碼，硬體上的差異，交給系統負責處理。所以我們有子計畫一，負責移植能利用目前最新的 SMP 和 server Clustering 技術的微核心作業系統 (OSF/1) 到 Intel 的 SMP 平台上並利用 mach 作業系統的支援程式庫。但這並不夠，因為 OSF/1 只支援以訊息傳遞為基礎的行程間通訊機制 (interprocess communication mechanism)，並不能完全滿足平行編譯器的需求。所以，又有子計畫二的出現，提供一套能讓平行編譯器設計者很方便使用的行程間和行程內的通訊機制，以簡化他們的工作。而子計畫四，實作一程式切片系統。可以整合於我們的 Run-time Parallelization Method 以獲得程式切片後的結果 Clustering Parallelizing Compiler Design 其中最重要的因素包括(1)硬體及系統軟體的影響加入此平行化系統之中，(2)加入適當的同步與通訊指令，(3)分析重複計算(duplicate computation)與通訊(synchronized communication)之利弊，以導引平行化編譯工作。(4)產生適當的資料，協助作業系統來對這些程式迴圈作排程(Parallel Loop Scheduling)的工作。本計畫的實作部份，在所購置的 2-CPU 個人電腦叢集(PC Clustering)上實施。

五、計畫成果自評

未來的平行機器走向，是採用不同的機器上之間的相互合作關係，也就是透過網路上資料的傳輸，來達成平行處理、分散計算的目的，也就是架構在非對稱的平行架構(NUMA)系統上，因此我們的 Parallelizing Compiler 也朝向這個方向發展。我們會現有的 Compiler System 從原來的對稱式機器(UMA)移植至平行式機器架構。在 NUMA 的系統下，因為是透過網路的傳輸，其 Overhead 是相當大的，為了減低 Message Passing 的 Cost，先前有關平行迴圈排程的問題就顯得相當重要了，我們利用資料之間的同質性及再使用性(Reusable)，來減少網路上存取的次數，並提高平行化的程度。整個 Compiler System 是以一種專家系統的模式來完成各方向的設計，提供一個更有彈性、更有智慧的方式來完成系統的設計。本計畫已發表相關論文共五篇，其研究成果，在平行編譯器之研究及分析上可提供重要的參考。

參考文獻

[1] W. Blume, R. Eigenmann, K. Faigin, J. Grout, J. Hoeflinger, D. A. Padua, P. Petersen, B. Pottenger, L. Rauchwerger, P. Tu, and S. Weatherford, "Polaris: The next generation in parallelizing compilers," in Proc. 7th Int'l. Workshop Languages and Compilers for Parallel Computing, Lecture Notes in Computer Science, 892:141-154, Springer-Verlag, Ithaca, New York, Aug. 1994.

[2] W. Blume, R. Eigenmann, J. Hoeflinger, and D. Padua, P. Petersen, L. Rauchwerger, P. Tu, "Automatic detection of parallelism: A grand challenge for high-performance computing," IEEE Parallel & Distributed Technology, 2(3):37-47, Fall 1994.

[3] U. Banerjee, R. Eigenmann, A. Nicolau, and D. A. Padua, "Automatic program parallelization," Proc. IEEE, 81(2):211-243, Feb. 1993.

[4] D. F. Bacon, S. L. Graham, and O. J. Sharp, "Compiler transformations for high-performance computing," ACM Computing Surveys, 26(4):345-420, Dec. 1994.

[5] J. Boykin, D. Kirschen, A. Langerman, and S. LoVerso, Programming under Mach, Addison Wesley, 1993.

[6] R. Eigenmann, J. Hoeflinger, Z. Li, and D. Padua, "Experience in the automatic parallelization of four perfect benchmark programs," in Proc. 4th Annual Workshop Language and Compilers for Parallel Computing, Lecture Notes in Computer Science, 586:65-83, Springer-Verlag, Aug. 1991.

[7] G. J. Hwang and S. S. Tseng, "EMCUD: A knowledge acquisition nethod which captures embedded meanings under uncertainty," Int'l. J. Man-Machine Studies., 33:431-451, 1990.

[8] M. C. Hsiao, S. S. Tseng, C. T. Yang, and C. S. Chen, "Implementation of a portable parallelizing compiler with loop partition," in Proc. Int'l. Conf. Parallel and Distributed Systems, ICPADS'94, Hsinchu, Taiwan, R.O.C., pp. 333-338, Dec. 1994.

[9] K. Hwang, Advanced Computer Architecture: Parallelism, Scalability, Programmability, MIT Press and McGRAW-Hill Inc., 1993.

[10] S. H. Kao, C. T. Yang, and S. S. Tseng, "Run-time parallelization for loops," in Proc. 29th 1996 Hawaii International Conference on System Sciences, HICSS-29, vol. 1, pp. 233-242, Hawaii, Jan. 1996.

[11] D. J. Lilja, "Exploiting the parallelism available in loops," Computer 27(2):13-26, Feb. 1994.

[12] K. Loepere, Mach 3 Kernel Principles, Open Software Foundation and Carregie Mellon University, 1992.

[13] K. Loepere, Mach 3 Server Write's Guide, Open Software Foundation and Carregie Mellon University, 1992.

[14] S. Leung and J. Zahorjan, Extending the Applicability and Improving the Performance of Runtime Parallelization, Technical Report UW-CSE-95-01-08, Department of Computer Science and Engineering, University of Washington, Jan. 1995.

[15] O'Reilly & Associates, Inc., Guide to OSF1: A Technical Synopsis, O'Reilly & Associates, Inc., Sebastopol, CA, June 1991.

[16] C. D. Polychronopoulos Parallel Programming and Compilers, Kluwer Academic Publishers, MA, 1988.

[17] W. C. Shih, C. T. Yang, and S. S. Tseng, "Knowelwdge-based data dependence testing on loops," in Proc. 1994 International Computer Symposium, Hsinchu, Taiwan, R.O.C., pp. 961-966, Dec. 1994.

[18] M. Wolfe, High-Performance Compilers for Parallel Computing, pp. 137-162, Addison-Wesley Publishing, New York, 1995.

[19] Chao-Tung Yang, Shian-Shyong Tseng, and Chang-Sheng Chen, "The anatomy of parafrase-2," Proc. of the National Science Council Republic of China (Part A), 18(5)450-462, Sep. 1994.

[20] Chao-Tung Yang, Shian-Shyong Tseng, Cheng-Der Chuang, and Wen-Chung Shih, "Using knowledge-based techniques for parallelization on parallelizing compilers," in Proc. EURO-PAR'95, Lecture Notes in Computer Science, 966:417-428, Springer-Verlag, Stockholm, Sweden, Aug. 1995.

[21] Chao-Tung Yang, Shian-Shyong Tseng, and Ming-Chang Hsiao, "A model of parallelizing compiler on multithreading operating systems," International Journal of Modelling and Simulation, vol. 18, no. 1, 1998..

[22] Chao-Tung Yang, Cheng-Tien Wu, and Shian-Shyong Tseng, "PPD: A practical parallel loop detector for parallelizing compilers," in IEICE Trans. Information and Systems, Vol. E79-D, No. 11, 1545-1561, Nov. 1996.

[23] Chao-Tung Yang, Shian-Shyong Tseng, Cheng-Der Chuang, and Wen-Chung Shih, "Using knowledge-based techniques on loop parallelization for parallelizing compilers," Parallel Computing, .vol. 23, no. 3, pp. 291-309, May 1997.

[24] Chao-Tung Yang, Shian-Shyong Tseng, Ming-Chang Hsiao, and Shih-Hung Kao, "A portable parallelizing compiler with loop partitioning for multithreading operating systems," Proc. of the National Science Council Republic of China (Part A), vol. 23, no. 3, 1998.

[25] H. P. Zima and B. Chapman, Supercompilers for Parallel and Vector Computers, Addison-Wesley Publishing and ACM Press, New York, 1990