行政院國家科學委員會專題研究計畫成果報告

平行編譯環境中有關不規則相依迴圈平行化方法之探討

A Study of Parallelization Techniques for Irregular Dependence Loops in Parallelizing Compiler Environment

計畫編號：NSC 87-2213-E-009-049
執行期限：87 年 8 月 1 日至 88 年 7 月 31 日
主持人：陳正　　E'mail：cchen@eicpca5.csie.nctu.edu.tw
執行單位：國立交通大學資訊工程研究所

一、中文摘要

本年度的計畫主要是提供一般共享式多處理機系統良好且有效的平行編譯環境。我們不但在迴圈排程方面提供一有效的平行度擷取方式，而且配合資料分解及計算分解的技巧讓多處理機的效能有顯著的提升。

關鍵詞：迴圈排程、平行度、同步、不規則資料相依、共享記憶體多處理機、資料分解、計算分解、靜態排程。

Abstract

This project provides an effective parallel compilation environment for shared memory multiprocessor environment. We not only provide an effective scheduling mechanism but also propose an efficient computation and data decomposition technique to promote the performance of multiprocessor environment.

Keywords：Loop Scheduling, Parallelism, Synchronization, Non-uniform Data Dependence, Shared Memory Multiprocessor, Data Decomposition, Computation Decomposition, Static Scheduling.

二、計劃緣由與目的

一般而言，同步的機制可以用來維持迴圈的相依性限制。另一方面，它也可以整合迴圈排程方式構成一個完整的架構。同時，正確的執行及平衡的工作量分配都將同時被達成。在本年度的計畫中，我們提出了一個新的迴圈排程機制稱為 M 步跳躍方式，用來對不規則相依迴圈在多處理機系統中做排程的動作。根據我們實驗的結果，如果迴圈中存在足夠的平行度，我們的方法將可靠的擷取出適當的平行度，而且平均效果上要比一般的方法在效能上要好 20.29% 左右[2]。

再者由於遠端記憶體存取時間遠大於區域記憶體存取[13]。因此，我們利用計算分解與資料分解之架構計算出迴圈中計算與陣列資料之相關性。並採用廣域資料分析，對整個程序中之迴圈進行陣列資料分析，促使陣列資料分解型態一致，以減少資料重組溝通。同時利用迴圈交換方法，促使迴圈中之計算對陣列資料之存取順序能符合資料之區域性。最後，經由資料型態之分析決定出最後之區塊或循環分配方式，將具有相關性之計算與資料分配到同

一個處理器和其區域記憶體中。初步評估結果顯示，當陣列之下標函數之變數較為複雜時，本方法能經由計算與資料分解之分配方式，降低遠端資料存取順序。而當迴圈之計算順序會破壞陣列資料之區域性時，本方法會利用迴圈交換方式，使其存取順序滿足資料之區域性，進而降阺遠端存取次數。因此，將此方法搭配不規則相依迴圈靜態排程方式，確實可減少迴圈中遠端存取之次數[2]。

三、結果與討論

因為迴圈中擁有大量的平行度，同時執行對於共享式多處理機系統便顯得相當重要。在本計畫中我們提出了一個有效的不規則迴圈排程技巧稱之為 "M 步跳躍方式"[2]。根據我們在 CONVEX SPP 1000 [7-8]的實際多處理機系統執行的結果，我們的方法在實際的程式中確實能有效的提升其效能。根據解 diophantine equations [5]及最小相依距離[5]的概念，我們已經闡釋及決定了細節的跳躍資訊，包括一次可完全平行執行的迴圈數量，以及如何決定跳躍門檻及跳躍間距等。又因為 M 步跳躍並沒有對排程方式作嚴謹的定義，因此必須配合叢集大小控制函數(chunk size control function)[3]才可以有效的運作。初步評估結果顯示我們的方法比較適合引導式迴圈自我排程技巧 (Guided Self Scheduling)[3] ，並能充分發揮其效能。

我們的評估結果顯示我們的排程技巧會因為受測標竿程式的平行度而有顯著的差異。如果其中包含足夠的平行度，則我們的 M 步跳躍方式將非常穩定的擷取開發足夠的平行度，而且不會產生嚴重的同步負擔。和目前任何迴圈切割技巧最顯著的差別是我們的方法成功的刪除了所謂的多同步平行動作(multi barrier synchronization) 以及較早的釋放出可完全平行執行的迴圈元素(iterations) 。而可達成此效果的原因在於本方法並不依賴間隔區塊同步(cross-block synchronization primitives)和延遲的機制。本方法利用了動態調整跳躍資訊(hopping information)[2]以便維持相依關係的正確性。

另一方面，隨著多處理機系統的發展，具有 Non-Uniform Memory Access time (NUMA)特性的分散式共享記憶體多處理機系統 (Distributed-Shared Memory Multiprocessor System)愈來愈受重視，它兼顧了分散與共享記憶體的特性。在 NUMA 系統，每個處理機都有自己的記憶體模組 (Local Memory)，其他的則為遠端記憶體模組 (Remote Memory)。遠端的 (Remote)資料存取時間會比區域性的 (Local)資料存取時間長很多；所以如果資料放置不恰當，系統會花很多時間在傳遞資料，嚴重影響系統效能。因此，如何依據程式的計算 (Computation)以及資料依存關係，將資料 (data)適當地分配 (Decomposition)到各個處理機上，以達到系統的最佳效能，是平行編譯器的一個重要課題[15]。

目前，這方面的研究大多是探討在規則資料相依迴圈 (Uniform Data Dependence Loop)中的資料分配。但是在不規則資料相依迴圈 (Non-uniform Data Dependence Loop)[4, 6]中對於資料分配的研究卻很少。原因在於規則資料相依迴圈在做資料分配時不必考慮資料和位置的關係。反

觀在不規則資料相依迴圈中資料相依關係會因迴圈所在位置而有差異。因此，資料分配上就變的非常複雜。以往遇到較為複雜的資料相依向量時，就必須使用動態分配的方法，但分配動作會增加程式的執行時間。若無法有效分配資料，反而會降低整體的效能。因此，我們採用靜態分配來分配迴圈中陣列之資料。

我們針對在不規則資料相依迴圈中資料相依關係、迴圈所在位置及資料使用形態 (pattern)加以觀察，找出不規則資料相依迴圈中資料分配的方法。再依據迴圈位置排程來決定位置分配函數 (iteration decomposition function)，並利用整體平行度與區域性最佳化 (global optimizations for parallelism and locality)[14]方法，求出相對的資料分配函數(data decomposition function)。接著將程序中每個迴圈所對應出的資料分配函數，進行廣域(global)之整合分析，並配合區塊 (block)或是環狀 (cyclic)分配方式[14]，將陣列資料依據資料分配函數配置 (allocation)於分散式共享記憶體系統中每個處理機的記憶體模組中。同時，我們也將本計畫所提之方法實作在由Stanford Compiler Group 所發展，名為 SUIF[16]的一套平行編譯環境，並將編譯後的結果在 CONVEX SPP 1000 分散式共享記憶體多處理機系統進行模擬評估。

我們利用不規則相依迴圈靜態排程方法中之一種索引同步方法 (Index Synchronization Method)為我們評估之靜態排程方式。並利用其方法所採用之二個程式模型 (Program model) 和 實 際 應 用 程 式 Fishpack[12]、Eispack[10]和 Linpack[9, 11]中之四個不規則相依迴圈為評估對象。根據評估結果，本方法的確能減少遠端記憶體存取次數。

綜合上述，則我們可以知道，本方法將計算分解單位所對應之資料分解單位加以分配，的確能減少快取記憶體失誤率。但如果陣列中之下標函數不是如此複雜；如在下標函數中各層迴圈變數之係數皆不大於 1，則我們之對應資料分解單位分配方法的效果便不明顯。因為，此時我們的分解方式，常會和只進行資料區塊分配方法之分解方式一致。但如果當原本迴圈中的迴圈元素對資料的存取方式會破壞資料之區域性時，本方法會對此迴圈加以分析比較，觀察其是否可進行迴圈交換，並經由迴圈之交換來發揮資料之區域。因此，如果能夠將我們的分配方法實際應用在資料分配方法上，便能減少遠端記憶體存取次數，進而降低程式執行時間。

四、計劃結果自評

總體而言，我們的方法大致達成以下幾點目標：

1. M 步跳躍方式的執行效能確實正比於可擷取的平行度。
2. 如果待測標竿程式的平行度足夠，則 M 步跳躍方式將可確實擷取其平行度而且也避免產生無忍受的同步負擔 (scheduling overhead) 。
3. M 步跳躍方式也能針對不規則相依迴圈做有效的排程。
4. 本方法將計算分解單位所對應之資料分解單位加以分配，的確能減少快取記憶體失誤率。

5. **如果能夠將我們的分配方法實際應用在資料分配方法上，便能減少遠端記憶體存取次數，進而降低程式執行時間。**
6. **我們找出有效不規則資料相依迴圈中資料分配的方法。**
7. **我們依據迴圈位置排程來決定位置分配函數，並利用整體平行度與區域性最佳化方法，求出相對的資料分配函數。接進行廣域之整合分析，並將陣列資料依據資料分配函數配置於分散式共享記憶體系統中。**
8. **我們也將本計畫所提之方法實作在由** Stanford Compiler Group **所發展，名為** SUIF **的一套平行編譯環境。**

由以上幾點特性可知本計畫確實可提供平行編譯環境改善排程及配置方式的參考。而且也確實適和建立於目前的平行編譯環境中提供多處理機環境效能的改善。

五、參考文獻

[1] J.L.Lee and C. Chen, A Study on Effective Computation and Data Decomposition Techniques for Non-uniform Dependence Loops, Master Thesis, July, 1999.

[2] H.T. Chua and C.Chen, M-hopping Method: An Efficient Loop Scheduling Scheme for Non-uniform Dependence Loops, Master Thesis, July, 1999.

[3] C. D. Polychronopoulos, "Guided Self Scheduling: A Practical Scheduling Scheme for Parallel Supercomputers", IEEE Tran. Computers, Vol. C-36, No. 12, pp. 1425-1439, Dec. 1987.

[4] D. K. Chen and P. C. Yew, "An Empirical Study on DOACROSS Loops", Proc. Supercomputing, pp-620-632, 1991.

[5] S. Punyamurtula and V. Chaudhary, "Minimum Dependence Distance Tiling of Nested Loops with Non-uniform Dependences", Proc. 6th IEEE Symposium on Parallel and Distributed Computer, pp. 74-81, May 1994.

[6] D. L. Pean, C. C. Wu, H. T. Chua and C. Chen. "Effective Parallelization Techniques for Non-uniform Loops", Proc. of the 21st Australian Computer Science Conf., Perth, pp. 393-404, Feb. 1998.

[7] *Exemplar Architecture,* Convex Computer Corporation, Jun. 1994.

[8] *CONVEX Exemplar Programming Guide,* Convex Computer Corporation, Jun. 1994.

[9] ftp://ftp.ucar.edu/ftp/dsl/lib/linpack/

[10] http://elib.zib.de/netlib/eispack/

[11] http://cm.bell-labs.com/netlib/itpack/index.html

[12] ftp://ftp.ucar.edu/ftp/dsl/lib/fishpak

[13] Lenoski. D. E., et al. "The Stanford DASH multiprocessor," IEEE Comput. Vol. 25, No. 3, Mar, 1993, pp.63-79.

[14] J. M. Anderson and M. S. Lam, "Global optimizations for parallelism and locality on scalable parallel machines," In Proc. the SIGPLAN '93 Conference on Programming Language Design and Implementation, pp. 112-125. , June 1993.

[15] M. Gupta and P. Banerjee, "Demostration of automatic data partitioning technques for parallelizing compilers on multicomputers, " Transactions on Parallel and Distributed Systems, Vol. 3, No. 2, pp.179-193, March 1992.

[16] Robert Wilson, et al., An Overview of the SUIF Compiler System, SUIF Manual, Computer Systems Lab Stanford University.1994.