分散式計算環境資料平行度的開發與利用

(Exploration and Exploitation of Data Parallelism for Distributed Computing Environment)

計畫編號：NSC87-2213-E-002-043
計畫期限：08/01/97-07/31/98

主持人：王 勝 德　　台灣大學電機工程研究所
Email: sdwang@cc.ee.ntu.edu.tw　Fax: 02-2367-1909

一、中文摘要

(關鍵詞： 大型平行機，資料對齊，平行編譯器，分散式記憶體，程式分割，資料分割)

對於有高平行處理能力的系統，過多的資料傳輸通常會降低平行計算的效果。為了充分利用系統的資源，目前已有許多方面的研究致力於消除不必要的資料傳輸，而我們的對策則是嘗試複製陣列來解決這個問題。

在本報告中，我們提出了一個有系統且有效率的方法來決定該如何複製陣列在處理器上。我們的方法不僅能夠自動決定該整個陣列複製或是部分陣列複製，而且還能夠自動決定該複製到全部的處理器或是部分的處理器，而這些特性是傳統的方法所無法辦到的。為此，我們首先提出了一個新的模板(template)**來處理資料對齊**(data alignment)**的問題，也因為如此，在後來的資料分配上，我們得以全新的觀念來進行。**

英文摘要

(Keywords: Massively parallel processors, Data Alignment, Distributed memory, Program partitioning, Data partitioning, Data alignment)

Writing an effcient program for massively parallel machines (MPP) requires domain knowledge about message-passing and data distribution concepts. In order to efficiently utilize the computing resources, efforts are devoted to eliminating the communication overhead. We propose applying a data replication approach together with data alignment to solve the problem. In the proposed method, data arrays can be replicated in the whole or in part into all processors or subset of processors. The existing alignment approach is not well suited to this kind of problem. Therefore, an abstract computation array is proposed as the template, and data distribution is achieved through partitioning the template among processors. A data array can be fully replicated to eliminate the spatial data dependences if there exists no true dependence across iterations on the data array. Arrays with loop-carried data dependences can be partly replicated if we properly dispatch the iterations to processors. Performance evaluation is carried out in a Cray T3D and the results show that the proposed approach is effective under a restricted loop model.

二、計畫緣由與目的

最近在高平行計算的領域中，分散式系統是越來越受到歡迎。目前已有一些平行編譯器被開發出來，像 ParaScope [CaKe88], Crystal [LiCh91a], Fortran D [HiKe92]**等。但是這些平行編譯器在一些像資料分割的方法或最佳所需處理器的個數等問題，仍需要使用者自行決定；即使其能經由評估資料傳輸的代價來自動決定，也是沒有系統而且複雜，所以目前還有許多研究不斷地在投入。**

在分散式系統中，若所運算的資料不在本地的記憶體內，則必需透過網路到其他處理器的記憶體單元去存取。因為在網路的存取上是十分費時的，因此如何找出好的資料分配是當前的一大課題。然而 Mace [Mace87]**早已證明，在平行計算上，**

即使是一維陣列或兩維陣列的情況，想要找出最佳的資料存放是屬於 NP-完全性的問題(NP-complete)，所以由使用者來自行決定好的資料分配是困難的。為了討論方便，一般都把資料分配分為兩個步驟來處理。首先是將所有的陣列對映到一個共同的資料結構，然後再將此資料結構分配到處理器，如此一來，陣列也相對地會依照對映的關係而被分配到處理器上。前者即是我們所謂的資料對齊(data alignment)，而後者則為資料分配(distribution)問題。

目前在文獻上，已經有兩種技巧分別在討論如何處理資料對齊的問題。其中一種是設法找出一組平行的超平面(hyperplanes)來對齊陣列，這包括了 Ramanujam 和 Sadayappan [RaSa91], Chen 和 Sheu [ChSh94], Darte 和 Robert [DaRo94]等。Ramanujam 和 Sadayappan 分析資料陣列的參考樣本(reference patterns)，並把其位移向量視為 k 維度空間上的點，然後設法找出一個能包含所有位移向量的 k-1 維度超平面，以得到一個無需資料傳輸的分割。Chen 和 Sheu 更進一步針對具有相同參考函數(reference function)的陣列，以分割迴空間(iteration space)成區塊的方法，來得到一個在區塊間無需資料傳輸的分割。他們也考慮了複製陣列的與否而得到不同的分割。Darte 和 Robert 則是引進了資料傳輸圖(communication graph)來有系統地描述資料對齊的問題。這個模型更見一般性，因為在這個模型中，除資料陣列外，每個指令的執行也需要分配，因此無須一味地再遵守"擁有才計算"的規則("the owner computes" rule)。

另一種是直接取陣列各維度間的關係來處理資料對齊問題，這包括了 Li 和 Chen [LiCh91b], Gupta 和 Banerjee [GuBa92], Hovland 和 Ni [HoNi93]等，而我們的研究也是屬於這一類。Li 和 Chen 已經把對齊問題定義成在找一組陣列下標間範圍(index domain)的映射，以使得到資料參考的最大一致性(uniformity)。他們也歸納了四個簡單的映射函數來處理對齊問題。而 Gupta 和 Banerjee 先分析程式中的每個迴圈(loop)，而得到概括的計算時間函數與傳輸時間函數，然後再嘗試將兩個函數做最佳的調整，以得到最快的程式執行時間。Hovland 和 Ni 則視尋找好的資料分配為分析資料相關的延續，並且提出了利用擴展的資料存取描述子(augmented data access descriptors)來處理。

三、資料分割與分配方法

在處理陣列下標間的範圍對齊時，我們先考慮重排、嵌入、投影等函數，然後再考慮移位、反射等函數。對於找重排、嵌入、投影等函數，我們主要是利用 Li 和 Chen 所提出之分割元件親和力圖(component affinity graph)的方法。但是在決定多出來的維度時，對於嵌入函數，我們是採用填"1"的方法，並且由其所產生的函數，我們稱之為部分對齊映射(partial alignment mapping)；而對於投影函數，我們則是採用摺疊映射的方法。至於找反射函數，我們是採取樣板比對(patterns matching)的方法，而找移位函數時，則是由中間位移量來決定。

接下來是計算陣列的分配。因為我們的模板是一個計算陣列，所以在分配階段，我們可以視為在分配迴給處理器執行。這個觀點賦予了傳統的分配不同的意義。我們的方法主要是針對產生"單一程式多處理資料"(single program multiple data)的平行碼而設計的，而且產生的迴圈，其外圍是平行層(parallel levels)，內部是連續層(serial levels)，所以只要知道迴圈中那幾層是平行的、那幾層是連續的，就可以知道計算陣列該如何分配到處理器上了。

我們發現迴圈中的連續層是由跨迴圈的資料相關(loop-carried data dependences)以及前面所說的部分對齊映射來共同決定的。對前者而言，若有跨迴圈的資料相關發生在迴圈的第 c 層，則意指在第 c 層以及其下的所有層的迴都必需依序執行。當然我們可以事先做迴圈交換(loop interchange)以得到最大的平行度，但是必需滿足一些條件[ZiCh91]。至於部分對齊映射，其表示有 1 出現在對齊映射的模板的維度上，若我們將該維度所對應的迴圈層平行化，則與該維度之第一個處理器做資料傳輸，是無

可避免的，因此在該維度所對應的迴圈層的迴也必需依序執行。

在我們的方法中，對於必需依序執行的迴，我們都將之分配到同一個處理器去執行，對於其他的迴，我們則儘量讓他們在更多的處理器去平行執行。所以迴圈中那幾層是平行的、那幾層是連續的即可得知。之後，在分配映射中，我們將連續層所對應的模板維度摺疊(collapsed)，而平行層所對應的模板維度則予以完全分配到處理器上。

四、資料陣列的複製

資料陣列的複製可以彌補陣列下標間範圍對齊的不足，因其導致無資料傳輸的執行，所以不僅增加了資料的區域性(data locality)，也改進了平行度。然而資料陣列的複製與否取決於發生在該陣列的資料相關。在共享式記憶體系統中，資料相關決定了迴的執行順序，我們稱此種資料相關為時間性的相關，其包括了跨迴圈的資料正相關(true dependences)、反相關(anti dependences)及輸出相關(output dependences)。在分散式系統上，假如資料陣列不在本地的記憶體內，則需要靠資料傳輸來取得其值，我們稱此種資料相關為空間性的相關，其包括了所有的資料相關。由以上可知，時間性的相關限制了迴圈的平行性，而空間性的相關則限制了平行計算的加速值。

為了加入陣列複製，我們的資料分割策略如下：

(1)陣列下標間的範圍對齊，

(2)陣列的複製，

(3)計算陣列的分配。

如前面所說的，非跨迴圈正相關陣列可以整個地被複製，而跨迴圈正相關陣列則必需視迴的分配而定。但是我們並不需要每次都複製整個陣列到全部的處理器上，只要使時間性的相關和空間性的相關剛好被消掉就可以。

在複製非跨迴圈正相關陣列方面，我們的方法是：(1)把部分對齊映射轉成重複映射，也就是說把部分對齊映射中的模板有"1"的維度換成"*"。例如對齊映射是 $A(i,k+1) \rightarrow T(i,1,k)$，則轉成的重複映射為 $A(i,k+1) \rightarrow T(i,*,k)$。(2)補償不對齊存取，也就是說把不完全對齊維度及其共維度均換成"*"。例如對齊映射是 $C(j,i) \rightarrow T(i,j)$，而陣列 C 的第 1 維是不完全對齊維度，則結果為 $C(*,i) \rightarrow T(i,*)$。

然而在複製跨迴圈正相關陣列方面，因為跨迴圈的資料正相關是無法利用陣列的複製來予以消除的，所以我們必需先知道跨迴圈的資料正相關是發生在迴圈的那些層。在那些層以及其下的各層是屬於連續層，而其他的層則是屬於平行層。在平行層上，陣列是可以複製的，所以我們就可以應用前面所說之複製非跨迴圈正相關陣列的方法。

當資料被複製之後，有時我們需要再把它們重新分配(redistribution)，以下我們提供一個方法來處理資料的收集：(1)對於沒被複製的陣列，我們直接從陣列的分配映射即可得知。(2)對於被複製的陣列，假如該陣列在迴圈內是唯讀的，則我們就可以從複製維度的第 1 個處理器來收集。但若該陣列在迴圈內曾被寫過，則我們就從執行最後寫入的那個迴圈的處理器來收集。

五、實作與實驗

考慮下列迴圈

```
do i = 1, n
  do j = 1, n
    B(i,j) = A(i+2,j+1)
    A(i+1,j) = C(j,i)
    A(i,j+1) = C(j+1,i)
    D(i,j) = B(i,j)
  end do
end do
```

，

若不複製陣列時，因為有資料相關存在，所以資料分配為

$A(*,*) \rightarrow P(1)$

$B(*,*) \rightarrow P(1)$

$C(*,*) \rightarrow P(1)$

$D(*,*) \rightarrow P(1)$,

而且迴圈必需依序執行。但是在經我們前面所提的方法來複製陣列，因為所有的陣列均為非跨迴圈正相關陣列，所以最後可得資料分配為

$A(*,*) \rightarrow P(*,*)$

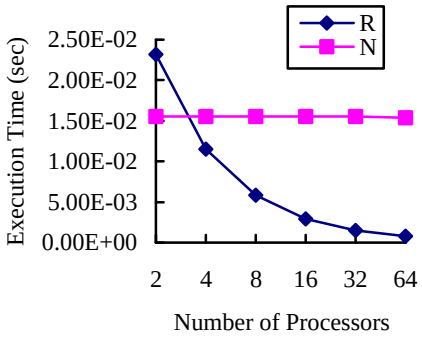
$B(i,j) \rightarrow P(i,j)$
$C(*,i) \rightarrow P(i,*)$
$D(i,j) \rightarrow P(i,j)$，

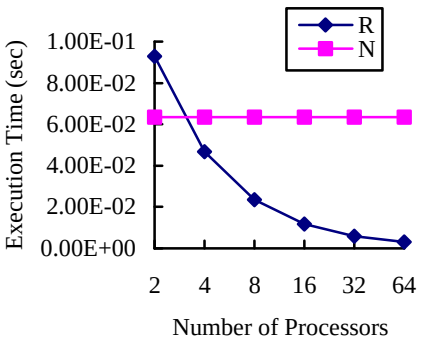
而且迴圈是平行執行如下：

```
doall i = 1, n
  doall j = 1, n
    B(i,j) = A(i+2,j+1)
    A(i+1,j) = C(j,i)
    A(i,j+1) = C(j+1,i)
    D(i,j) = B(i,j)
  end do
end do
```

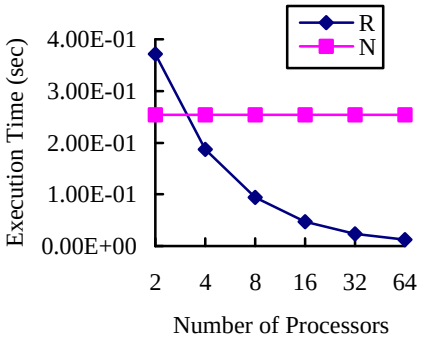
。

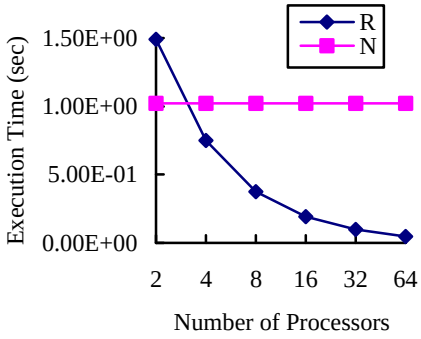
圖 5.1 **是其實驗的比較結果，我們發覺在同一個問題的大小下，當處理器的個數越多時，有複製陣列的，其執行會越來越快，而沒複製陣列的，其執行時間不變。**



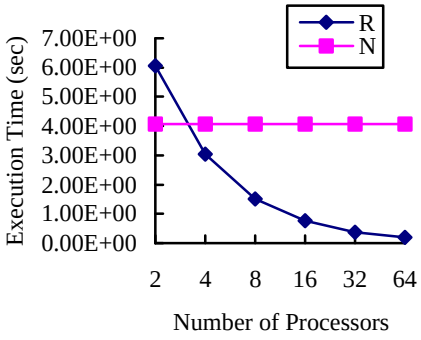


(a) **問題大小是** 64　(b) **問題大小是** 128





(c) **問題大小是** 256　(d) **問題大小是** 512



(e) **問題大小是** 1024

圖 5.1. **應用陣列複製在迴圈平行執行方面的效能比較。**

六、結論與計畫成果自評

在分散式系統中，一個好的資料分割能讓平行迴圈執行的更有效率。在大多數的情形，資料對齊是可以消除資料傳輸，並進而提升平行計算的加速值。但是像矩陣相乘的例子，在經過資料對齊之後，資料傳輸仍是難免的，因此我們想要利用陣列複製來解決這個問題。我們發展了一個定性的方法(qualitative rules)**來決定如何複製陣列，這是有別於利用窮舉評估**

(exhaustively evaluating)資料傳輸代價而來決定如何複製陣列的方法，所以我們的方法有系統且有效率。

在資料分割上，我們創造了一個抽象的計算陣列來當作模板以處理對齊問題，其中模板的每個元素均相當於迴圈中的每一個迴。所以在分配模板時，我們可以視為在分配迴給處理器執行。這個觀點賦予了傳統的分配不同的意義，而且特別有利於處理陣列的複製。我們也利用了資料相關分析(data dependence analysis)來自動決定如何複製資料陣列，因此在平行編譯器上，要應用我們的方法，是十分容易的。

將來，我們可以把硬體的條件考慮進來，如記憶體的大小、實際的處理器個數等等。而區塊大小的問題也是在模板分配方面急待解決的，因為它不僅會增進快取記憶體的使用效率，而且還會影響整體平行計算的加速值。

本計畫部份成果已發表在 Journal of Information Science and Engineering [WaJw98]，成果算是相當不錯。

七、參考文獻

[AgKr95] A. Agarwal, D. A. Kranz, and V. Natarajan, “Automatic partitioning of parallel loops and data arrays for distributed shared-memory multiprocessors,” *IEEE Trans on Parallel and Distributed Systems*, vol. 6, no. 9, Sep. 1995.

[CaKe88] D. Callahan and K. Kennedy, "Compiling programs for distributed-memory multiprocessors," *Journal of Supercomputing*, vol. 2, pp. 151-169, Oct. 1988.

[ChSh94] T. S. Chen and J. P. Sheu, “Communication-free data allocation techniques for parallelizing compilers on multicomputers,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 5, no. 9, pp. 924-938, Sep. 1994.

[DaRo94] A. Darte and Y. Robert, “On the alignment problem,” *Parallel Processing Letters*, vol. 4, no. 3, pp. 259-270,1994.

[GuBa92] M. Gupta and P. Banerjee, “ Demonstration of automatic data partitioning techniques for parallelizing compilers on multicomputers,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 179-193, Mar. 1992.

[GuBa93] M. Gupta and P. Banerjee, “PARADIGM: A compiler for automatic data distribution on multicomputers,” *ACM International Conference on Supercomputing*, Tokyo, Japan, July 1993.

[HiKe92] S. Hiranandami, K. Kennedy, and C. Tseng. “Compiling Fortran D for MIMD distributed-memory machines,” *Communications of the ACM*, vol. 35, no. 8, pp. 66-80, August 1992.

[HoNi93] P. D. Hovland and L. M. Ni, "A model for automatic data partitioning," in *Proc. of the 1993 Int. Conf. on Parallel Processing*, vol. II, Aug. 1993, pp. 251-259.

[LiCh91a] J. Li and M. Chen, "Compiling communication-efficient programs for massively parallel machines," *IEEE Trans. on Parallel and Distributed Systems*, vol. 2, no. 3, pp. 361-376, Jul. 1991.

[LiCh91b] J, Li and M. Chen, “The data alignment phase in compiling programs for distributed-memory machines,” *Journal of Parallel and Distributed Computing*, vol. 13, no. 2, pp. 213-221, Oct. 1991.

[Mace87] M. Mace, Memory Storage Patterns in Parallel Processing. Boston, MA: Kluwer Academic, 1987.

[MPhi95] M. Philippen, “Automatic alignment of array data and processes to reduce communication time on DMPPs,” *The 5th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, vol. 30, no. 8, pp. 156-165, Aug. 1995.

[RaSa91] J. Ramanujam and P. Sadayappan, “Compile-time techniques for data distribution in distributed memory machines,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 2, no. 4, pp. 472-482, Oct. 1991.

[WaJw98] S. -D. Wang and W. -D Jwo, “Replication and Partitioning for Data Arrays in Distributed Memory Systems,” Journal of Information Science and Engineering, vol. 14, pp. 281-298, 1998.

[ZiCh91] H. Zima and B. Chapman, Supercompilers for Parallel and Vector Computers, New York: ACM Press, 1991.