行政院國家科學委員會補助專題研究計畫成果報告

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

＊　　＊

＊　　**領域基底分割問題的平行演算法**　　＊

＊　　＊

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

計畫類別：　**個別型計畫**　　**整合型計畫**

計畫編號： NSC – 89 – 2115 – M – 009 - 012。
執行期間： 88 **年** 08 **月** 01 **日 至** 89 **年** 07 **月** 31 **日。**
計畫主持人： 林朝枝教授　　國立交通大學應用數學系。

執行單位：　　**國立交通大學應用數學系。**

中 華 民 國　89 **年**　9 **月**　12 **日**

行政院國家科學委員會專題研究計畫成果報告

領域基底分割問題的平行演算法

Parallel algorithms for domain-based decomposition problems

計畫編號：NSC-89-2115-M-009-012
執行期限：88 **年** 08 **月** 01 **日 至** 89 **年** 07 **月** 31 **日**
主持人：林朝枝 教授　國立通大學應用數學系
Cjlin@math.nctu.edu.tw

中文摘要

由領域基底分割技巧所先行處理的問題，大都會得到一個要求較特殊線性系統的求解。我們應用已知線性系統的直接解法，設計出平行韻律演算法來求該特殊線性系統的答案。所使用的主要指令為代數學之消去法。其所需執行時間與處理單元數目均少於已知的原來直接解法。

關鍵詞：平行計算機、韻律演算法、 領域分割、線性系統。

Abstract

Two kinds of systolic algorithms are used to solve the linear system AX=B such that it is obtained from the problems by the domain decomposition technique. The basic operation is the elimination law in linear algebra. In fact, it is the application of an existing linear system solver. The numbers of processing elements (PEs) and time steps are less than that in the original solver of a dense linear system AX=B.

Keywords: Parallel computers, systolic algorithms, domain decomposition, linear systems.

1. Introduction

Parallel algorithms are used to solve linear systems for many science and engineering problems. Parallel computers may be divided into two broad cases: distributed memory and shared memory. Systolic algorithm is one of parallel algorithms such that it is the first case. Each processing element (PE) has no direct access to memory of any other PEs, i.e. by message computers.

The domain decomposition techniques appear to be a natural way to distribute the solution of large sparse linear systems across many parallel processes. For a single domain decomposed into two sub-domains (1,2) connected by an interface (3), the partitioned matrix would look like the form:

$$A = \begin{vmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix}$$

Where A_{11} and A_{22} come from the interior of the sub-domains, A_{33} from along the interface, and A_{13}, A_{23}, A_{31}, A_{32} from the interaction between the sub-domains and the interfaces. This A is always symmetry. Here, we do not consider whether it is symmetry or not. In the follows, let A, A_{11}, A_{22} be n by n, n_1 by n_1, n_2 by n_2 matrix respectively.

In many domain decomposition problems, the iterative method of pre-conditioned conjugate gradient techniques is used to solve the linear system Ax=b for b a column vector. See [1,2]. It requires the solution of a linear system for the cross points in the interface area. Thus, the solution process involves some global communication between the processing elements of the used solver. Also, the various pre-conditioners for A will be based on their efficacy and on their parallel limitations. Hence, the choice of a pre-conditioner is critical in domain decomposition problems.

In [3], They identify the costs with the domain decomposition algorithms into three cases: (1) dot product, (2) matrix-vector product, (3) pre-conditioner solver. In our parallel algorithm, we obtain the solution of a linear system AX=B, where B will be any matrix containing m vectors. Our solver is designed by the use of direct method. Also, the cost is only dependent on a assigned statement of arithmetic computation.

2. Domain decomposition method

The basic idea of domain decomposition method is that instead of solving the initial

problem f(u)=g on a domain ¿ , the problem is split into p sub-problems $f_i(u)=g_i$, i=1,2,…,p. where f_i, g_i, are the restrictions of f, g to the sub-domains ¿ $_i$, with ¿ is the union of ¿ $_i$ and a coupling condition at the sub-domain interfaces. Usually, the execution time is the total sum of (1) computation: arithmetic complexity of algorithm; (2) communication: data movement complexity of the parallel process; (3) control part: tasks spawning, synchronization, termination detection of a distributed process. In the single CPU, on a sequential process, there is neither communication over control overhead. Since our solver has no shared memory, it is no control overhead. Also the elapsed execution time would include the data communication time.

3. The dense linear system solver

We review the solver for a dense linear system. Ax=b. See [4]. Under the directed method, the solver uses n(n+1) PEs to form a two dimensional systolic array. It requires 4n time steps. We would extend this solver to solve AX=B with B being b by m matrix. Its PEs number is also n(n+1). Its needs 4n+m-1 time steps. A time step is independent of the problem size n and m. The major instruction in our algorithm is the assignment statement of the form $a_{out}=a_{in}-P.b_{in}$ which is used to modify the value of a_{in} to a_{out} by the values of b_{in} and in a register P.

4. The solved strategy for domain decomposition problems

We apply the result of Section 3 to solve the linear system AX=B, where A is a spares matrix which is come from the domain decomposition problem. Here, unlike in [4], we assume that the pivot equation is not necessary to be exchanged. That is, the diagonal element $|a_{ii}|$ of A is large enough. Otherwise, some modified control instructions would be considered in our systolic algorithm. We apply the solver of our dense linear system with different sizes n_1 and n_2.. A more two dimensional systolic array is used to eliminate the first n_1+n_2 variables on sub-domains from the remaining $n_3=n-n_1-n_2$ equations which are the associated variables in the interface. After this process, once again we apply the solver of dense linear system with an array of $n_3(n+1)$ PEs to obtain the solution X..

5. The used systolic arrays

First, we need two arrays as shown in [3] to solve the n_1 and n_2 variables. See Figures 1 and Figure 2.Thus, we need $n_1(n_1+1)$ and $n_2(n_2+1)$ PEs. Then, these n_1+n_2 output of Figures 1 and 2 are carried into a linear array with $(n-n_1-n_2)(n_1+n_2)=n_3(n_1+n_2)$ PEs to eliminate the first n_1+n_2 variables in the remaining n_3 equations. See Figure 3. That is, the output of b-links in

Figures 1 and 2 are the input of c-links of Figure 3. Finally, the output of c-links and d-links are the input a-links of Figure 4 to solve the linear system AX=B. See Figure 4. In fact, the PEs of Figure3 can be used again in the Figure 4. Thus, the total PEs used is less than n(n+m) of the original dense linear system.

6. The systolic algorithms

The main instruction of PE is defined as follows.

(1) The first, second and fourth arrays are the same as in [3].

(2) The third array is the form. $c_{out}=c_{in}$; $d_{out}=d_{in}-c_{in}*P$. where the P in PE(i,j) is $a_{n1+n2+I,\ j}$, of the matrix A. The total time steps is $4n_1+2n_2+4n_3+m$. It is less than the original dense linear system.

7. Conclusions

Under the domain decomposition technique, many problems, such as the partial differential equations, would be reduced into a sparse linear system. This linear system can be considered as a special form with the sub-domains and their interfaces. These linear systems are always solved by the used of iterative method. Here, under the experiment of the designed systolic algorithm in a dense linear system, we use the directed method to present systolic algorithms to solve this special form linear system. There are less PEs and time steps than that appear in the solver of a dense linear system. We hope that this design consideration can be applied to solve some other problems.

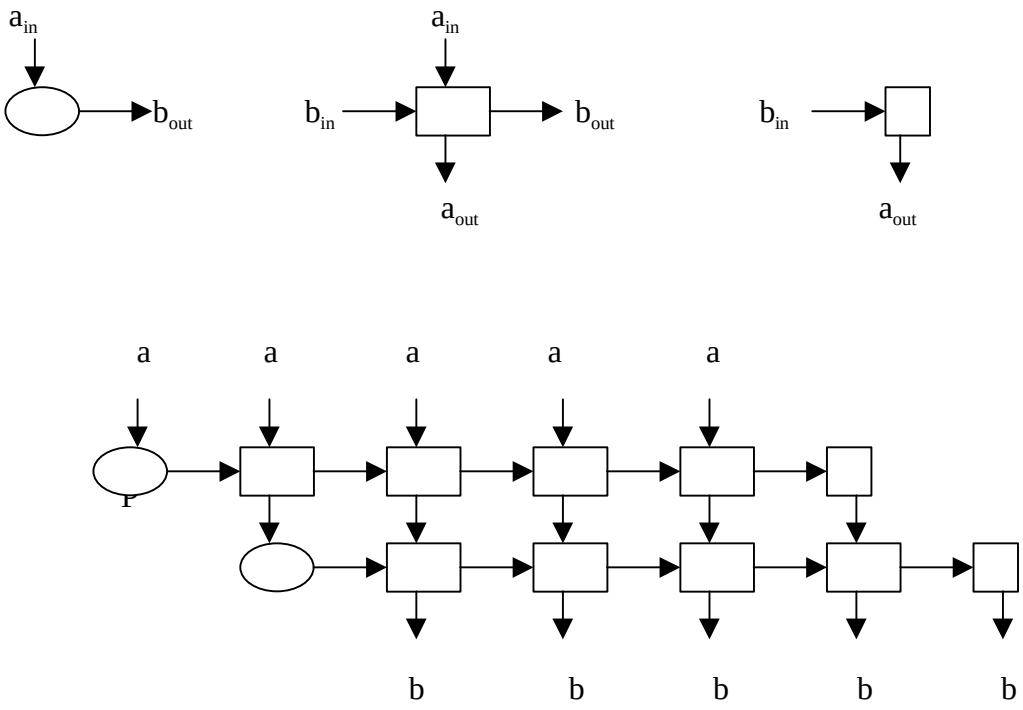
References

[1] J. H. Bramble, J. E. Pasciak and A. H. Schatz, The constructure of pre-conditioners for elliptic problems by sub-structuring, I, *Mathematics of Computation*, 47, July (1986), pp. 103-134.

[2] I. S. Duff, H. A. van der Vorst, Developments and trends in the parallel solution of linear systems, Parallel computing, 25, (1999), pp. 1931-1970.

[3] W. D. Gropp and D. E. Keyes, Domain decomposition on parallel computers, *Domain Decomposition methods*, SIAM, 1989. Pp. 260-268.

[4] C. J. Lin, A systolic algorithm for solving dense linear systems, *Computers and Mathematics with applications*, Vol.32, No.12, pp. 77-91.



Figures 1, 2, 4 Systolic arrays with n_1 (n_1+1), n_2 (n_2 +1) and n_3 (n.+1) PEs. .

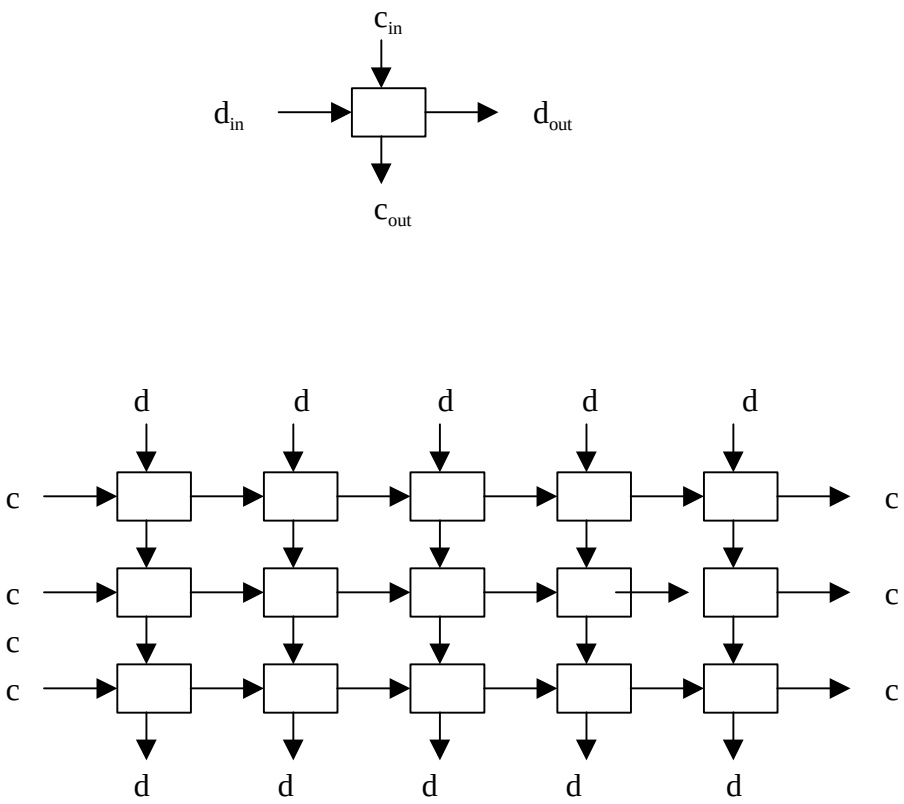


Figure 3. The systolic array with $n_3(n+1)$ PEs.