The Method Of Parallel Recognition And Parallel Optimization Based On Data Dependence With Sparse Matrix

Navid Bazrkar, Payam Porkar

Abstract: for application programs in scientific and technological fields have grown increasingly large and complex, it is becoming more difficult to parallelize these programs by hand using message-passing libraries. To reduce this difficulty, we are researching the compilation technology for serial program automatic parallelization. In this paper, the author puts forward a kind of parallel recognition algorithm in parallelization compiler with sparse matrix to reduce memory consumption and time complexity. In the algorithm the author adopts the idea of the medium grain parallel.

Index Terms: sparse matrix, medium grain parallel, parallel recognition, Parallel Optimization, Data Dependence

1 INTRODUCTION

In recent two decades, due to the rapid development of basic industries, computer hardware performance and softwaredesign level has significantly improved. For this reason, application programs in scientific and technological fields have grown increasingly large and complex. Thus, parallel computing has been one of the most active branches in the development of computer science. Furthermore, the material basis of parallel computing is the high performance parallel computer and the main task of Parallel compiler [1] is to generate parallel codes for the high-performance parallel computer, so that the performance of parallel compiler plays an important role during the high-performance parallel computer executing efficiently. With the development of parallel computer application and technology, parallel compiling system has become a very important part of system software in modern high-performance computer. For it is the most important content in parallel compiler [3] technology to auto-parallelize serial programs, more and more software designers become to pay more attention to the parallel recognition. Consequently, parallelization compiler becomes the main approach to overcome the difficulties such as programming on parallel computer and software transportation.

- *PayamPorkaris currently faculty memberin Damavand university, PH-00989122303971.*
  E-mail: PayamPorkar@damavandiau.ac.ir
- *Department of computer science, Islamic Azad university of Damavand,Tehtan, Iran.*
- *NavidBazrkaris currently have a master degree in computer science, PH-00989360550892.*
  E-mail: navid.bazrkar@damavandiau.ac.ir
- *Department of computer science, Islamic Azad university of Damavand,Tehtan, Iran*

At the same time, it plays an important role in using the accumulated heritage of serial program over a long period of time. Furthermore, it can greatly reduce not only the burden of programmer but also the cost of parallel program development. In this paper, the proposed method of program auto-parallelization based on data dependence is a kind of implementation on serial program auto parallelization technology with sparse matrix. This method can identify the blocks in serial program that can be computed in parallel while it scans the source code based on data dependence, at the same time, it can make the serial programs to parallel execute by decomposingthe serial programs into many subtasks, so that it solves the problem about the heritage of serial programs running on high performance computer. [9] In this paper we use of sparse matrix in parallel recognition algorithm for reduce memory usage.

2 PARALLEL RECOGNITION ALGORITHM

The key in auto-parallelization of serial program is parallel recognition. It is well known there is certain dependence among the various parts of a program, and the parallel computing is to execute these parts in parallel under the condition of not destroying these kinds of relationships, so as to shorten the running time of the whole program. As a result, the theory and technology on the analysis of dependence is the foundation of program parallelization, and it is also the foundation of parallel compiler. [9]

2.1 Parallel grain classification.

Parallel grain has been kept an eye in parallel recognition, and it is the premise of dependent analysis. The greater the parallel grain is, the smaller the overheads of synchronization and communication among processes after program parallelization are, and the higher the speed-up ratio in parallel is. But if we adopt greater parallel grain, the whole parallelism and running speed of program will be decreased. Parallel mechanism consists of three levels parallel grain, the coarse grain parallel, the medium grain parallel, the fine grain parallel. If the parallelizable unit is procedure, this parallel processing is called the coarse grain parallel [2]. In the same way, when the parallelizable unit is loop body that the parallel processing is called the medium grain parallel, and the parallel processing is called the fine grain parallel when the sentence is the basic element in parallel processing. After theoretically analyzing and experimenting on parallel grain, we have a conclusion that the medium grain is the best selection in parallel computing. The one is that adopting coarse grain parallelism can reduce the synchronization overheads and the difficulties of data-dependent recognition, but it can restrict the degree of program parallelization, the other is that adopting fine grain parallelism can improve the degree of program parallelization, but it not only can increase the synchronization overheads but also can make the difficulties of data-dependent recognition larger. Therefore, the author pays attentionto the parallel recognition and the parallel processing among blocks of serial program.[10]

2.2 The basic concept involved

2.2.1 Several definitions.

Definition 1 Dependence: In the program, if event or action A must take place before event or action B takes place, we say that event or action B depend on event or action A, and this relationship is called dependence relation or dependence. Dependence relation is usually divided into two types that are control dependence and data dependence. Control dependence often leads to that the program will change its executing sequence; otherwise data dependence is caused by reading or writing the same data. Because of the major factor that impacts program parallelization is data dependence, the dependence referred in this paper is usually data dependence.

Definition 2 Blocks: Blocks is the program element that there is only one entrance and only one exit. The character of the blocks is that the whole codes of blocks either must be executed or must not be executed [1]. In other words, during the process of running the blocks, transfer-out or transfer-in is not permitted. In this paper we use the symbol just like Bi to represent the blocks.

Definition 3 Dependence of the blocks: Assuming Bi and Bj are all blocks, if there is at least one of the output variables of Bj belongs to the input variables of Bi , or there is at least one of the output variables of Bi belongs to the input variables of Bj , we say that Bi is relevant to Bj , and this relationship of Bi and Bj is denoted as data dependence or dependence[5].

2.2.2 The data structures needed in establishing mathematical model.

Input Variable Matrix Mi : The input variable matrix Mi is a n×m matrix, the row of Mi is composed by blocks Bi (i=1,2,3…n) that are identified in the serial program, and the column of Mi is composed by variables aj (j=1,2,3…m) that are referred in the serial program. The matrix Mi describes whether the variable aj belongs to the input variables of Bi The value of Mi is: All tables and figures will be processed as images. You need to embed the images in the paper itself. Please don't send the images as separate files.

$$\text{Mi(i,j)} = \begin{Bmatrix} 0, & a_j \notin BiI \\ 1, & a_i \epsilon BiI \end{Bmatrix} \quad (1)$$

Where BiI is the set of the input variables Bi

Output Variable Matrix Mo :

The output variablematrix Mo is a n × m matrix, the row of Mo iscomposed by blocks Bi (i=1,2,3…n) that areidentified in the serial program, and the column ofMois composed by variables aj (j=1,2,3…m) that are referred in the serial program. The matrix Modescribes whether the variable aj belongs to the output variables of Bi . The value of Mois:

$$\text{Mo (i,j)} = \begin{Bmatrix} 0, & a_j \notin Bio \\ 1, & a_i \epsilon Bio \end{Bmatrix} \quad (2)$$

where B io represents the set of the output variables ofBi .

Dependence Matrix Md :

The matrix Md is a n×n matrix, and its rows and columns are all composed by blocks. The matrix Md describes whether the relationship of Bi and Bj is dependence. Then

$$\text{M d} = \begin{Bmatrix} 0 \\ n(n > 0) \end{Bmatrix} \quad (3)$$

If the relationship of Bi and Bj is dependence,

Md (i,j) = n, otherwise, Md (i,j) = 0.

Dependence Variable Matrix Mc :

The matrix Mc is a n × n matrix, and its rows and columns are all composed by the blocks. The matrix Mc lists the whole variables referred at the same time in Bi and Bj .

So that is

$$\text{Mc(i,j)} = \begin{Bmatrix} 0 \\ n \end{Bmatrix} \quad (4)$$

where Mc (i,j) = 0 represents that Bi and Bj is independence, once the relationship of Bi and Bj is dependence, the label of the variables referred in Bi and Bj will be saved in Mc (i,j). If dependence exists between the blocks Bi and the blocks Bj , the blocks Bi and the blocks Bj cannot be parallel processing. According to the definition of dependence, we can demonstrate this conclusion easily.[10]

2.3 Algorithm description

It is clear that if dependence exists between the blocks Bi and the blocks Bj , Bi and Bj cannot be executed parallel. Therefore, the main idea of this algorithm is discriminating whether dependence exists among blocks. According to the proposed definitions and conclusions, we know that the dependence discriminations of blocks can be realized by inquiring the proposed matrixes. Furthermore, inquiring the dependence variable matrix Mc , we can get the information of those shared variables of blocks, whose relationship is dependence.[9]

Step 1 Take source program as input, identify the blocks and sign the labels for blocks along their order such as B1 … Bn .

Step 2 Scan the variable table produced in compiling, build n × m input variable matrix Mi in accordance with the quantity of blocks and the quantity of variables that are saved in variable table, where n represents the quantity of blocks and m represents the quantity of variables.

Step 3 Scan the variable table produced in compiling, build n × m output variable matrix Mo in accordance with the quantity of blocks and the quantity of variables that are saved in variable table, where n represents the quantity of blocks and m represents the quantity of variables.

Step 4 According to the formula Md = Mi × M_o^T to build n × n dependence matrix Md .
1. A = 0
2.for (i=1; i=n; i++)
3.for (j=1; j=n; j++)
4.for (k=1; k=m; k++)
5.A=A+ M i (i,k)× M o (j,k); end for;
6. Md (i,j)= A;
7.A=0;end for;
8.end for;

Step 5 Build n×n dependence variable matrix Mc .
1.for (i=1; i=n; i++)
2.for(z=1; z=n; z++)
3.for(j=1; j=m; j++)
4.ifMi (i,j) × Mo (z,j)=1
5.then A[j]=1
6.else A[j]=0;end if; 7.end for;
8. Mc (i,z)= A[m];end for;
9.A[m]=0; 10.end for;

Step 6 Inquire Md , if Md (i,j) > 0 (i=1,2….n; j=1,2….n) data dependence exists between Bi and Bj, or Bi and Bj can run in parallel.[9]

2.4 Example of the algorithm

We use an example to demonstrate the above parallel recognition method. There is a program segment such as:

```
y=1;
t1: if a && b then x=0;
else y=0;
x=x+1;
y=y-1;
While ( x + y  > 0 )
{x=x-1 ;}
z=0;
```

Step 1: The blocks recognized are:
B1 : y=1; B2 :a && b B3 :x=0; B4 :y=0; B5 : x=x+1; y=y-1; B6 : x + y > 0 B7 : x=x-1 B8 :z=0;

Step 2: Build M i, M o and Md ,inquire Md and get the conclusion of dependence among the blocks.

Table1.*Mi***and** *Mo*

	a	b	x	y	z		a	b	x	Y	z
B1						B1				1	
B2	1	1				B2					
B3						B3			1		
B4						B4				1	
B5			1	1		B5			1	1	
B6			1	1		B6					
B7			1			B7			1		
B8						B8					1

The left is the input variable matrix M i , the right is the output variable matrix M o

Table2. *Md (dependency matrix)*

	B1	B2	B3	B4	B5	B6	B7	B8
B1								
B2								
B3								
B4								
B5	1		1	1	2		1	
B6	1		1	1	2		1	
B7			1		1		1	
B8								

The result is that there are data dependence between B5 and B1 , B3 , B4 , B5 , B6 , B7 , so they cannot be computed in parallel. Similarly, B6 cannot execute when one of B1 , B3 , B4 , B5 , B7 , is running and B7 can also not run when one of B3 , B5 , B7 is running. It is clear that the result is the same as the result by manual analysis. Scheduling overheads and communication overheads the author presents many simple strategies. [10]

2.5 Disadvantages of the proposed algorithm

Because the matrix element, are mostly zero in large scale program ,for this reason to reduce memory consumption, we use the sparse matrix.

3. Convert to sparse matrix

In this method , first we Convert M i and Mo Table to sparse matrix (blocks Bi (i=1,2,3…n) that areidentified in the serial program and variable a is 1 , b is 2 … , z is 5):

Table2.*Mi*

block	variable	Value
B2	1	1
B2	2	1
B5	3	1
B5	4	1
B6	3	1
B6	4	1
B7	3	1

Table3. Mo

block	variable	Value
B1	4	1
B3	3	1
B4	4	1
B5	3	1
B5	4	1
B7	3	1
B8	5	1

Then we compare the block in Mi and Mo matrix, if Exist a same block in Mi and Mo matrix , then we compare the variable in Mi and Mo matrix , if Exist a same variable in Mi and Mo matrix then we conclude dependence exists between the blocks Bi and the blocks Bj , Bi and Bj cannot be executed parallel.

For this compare we use of following code:

```
for (int i=0 ;  i<=6  ; i++  )
for (int j=0 ;  j<=6  ; j++  )
if (  A[j][1] == B[i][1] )
{  Z[x][0]=A[j][0]; //output
    Z[x][1]=B[i][0]; //output
     Z[x][2]=1; //output
                    x++ ;  }
```

3.1 New Dependence Matrix Md :

The matrix Md describes whether the relationship of Bi and Bj is dependence. For example; between B5 in Mi block and B1 in Mo block exist a dependence, and these blocks con not be executed parallel.

Table4 .output code** Md **matrix

Mi blocks	Mo blocks	value	variable
B 5	B 1	1	y
B 6	B 1	1	y
B 5	B 3	1	x
B 6	B 3	1	x
B 7	B 3	1	x
B 5	B 4	1	y
B 6	B 4	1	y
B 5	B 5	1	x
B 6	B 5	1	x
B 7	B 5	1	x
B 5	B 5	1	y
B 6	B 5	1	y
B 5	B 7	1	x
B 6	B 7	1	x
B 7	B 7	1	x

4. Parallel Optimization

in [9] the authors used three loop (O(n^3)) for construct M d matrix , but we use two loop (O(n^2)) for construct M d matrix , the best situation will be when the total numbers in sparse Mi matrix is lower the $\frac{n*m}{k}$ (***Mi***$\epsilon\boldsymbol{\Omega}\frac{n*m}{k}$) and Mo matrix is lower the (***Mo***$\epsilon\boldsymbol{\Omega}\frac{m*k}{n}$) ,which this method reduce time complexity . but , the worst situation will be when the total numbers in sparse Mi matrix is higher the $\frac{n*m}{k}$ and Mo matrix is higher the $\frac{m*k}{n}$, assuming that the ***Mi***matrix is N×M and the ***Mo***matrix is M×K,

5. Conclusion

In this paper, the author presents a kind of parallel recognition algorithm to realize automatic parallelization of serial program in parallelization compiler. This algorithm partitions serial program into many blocks and in according with the conditions of the variables referred in blocks, the algorithm can decide whether data dependence exists among the blocks, so that it can recognize the basic units that can parallel compute in medium grain parallel[6]. To implement the algorithm, the author builds four simple matrixes. By matrix operation such as matrix transposition and matrix multiplication, the algorithm can decide the relationship among the blocks [10]. So that, to reduce memory consumption and time of processing, we use the sparse matrix and after that we compare two matrixes for realize dependence exists among blocks. This method in mentioned situation can reduce time complexity and memory consumption.

5. References

[1]. JIN Cheng-zhi. “The Construction Principles and Implementation Techniques of Compilers” , Higher Education Press , Beijing , 2000.

[2]. HU Yan-li, ZHANG Wei-ming. “A dynamic scheduling algorithm of parallel coarse grain tasks in computational grid based on time-balancing strategy”, .Journal of Chinese Computer System , 2008

[3]. SHEN Zhi-yu ,HUZi-ang ,”Methods of Parallel Compilation”, National Defence Industry Press, Beijing , 2001.

[4]. CHEN Guo-liang . “Design and Analysis of Parallel Algorithm ”, Higher Education Press , Beijing , 2002 .

[5]. Stanford Compiler Group. “SUIF Compiler System Version1.0 ,US” . Stanford University, 1994.

[6]. LI Jing, ZANG Bin-yu , “Automatic Parallelism Detection for One Kind of Irregular Problems”, Journal of Software , 2002

[7]. M. Grieble, “Automatic Parallelization of Loop Programs for Distributed Memory Architectures”, FMI, University of Passau, 2004.

[8]. L. N. Pouchet, C. Bastoul, A. Cohen. “Iterative Optimization in the Polyhedral Model: part I, One Dimensional Time”, 2007.

[9]. Zhao Yan , Lei Liu , Li Ma . “ The Method of Parallel Optimization and Parallel Recognition Based on Data Dependence ” , 2009 IEEE.