The Parallel Solution Method of Sparse Triangular System Based on Correlation Decomposition

Li-cui SONG, Song JIN* and Tian-cheng LV
Department of Electronics and Communication Engineering, North China Electric Power University
*Corresponding author

Keywords: Parallel computing, The partial value of the variable, Multi-nuclear multithreading.

Abstract. We propose a parallel solution algorithm based on correlation decomposition. By decomposing the right-hand-side of the linear system and analyzing the dependencies of the variable solution to form multiple independent variable calculation paths. First, the partial value of the variable is calculated in parallel, and then the final result of the variable is calculated by adding the partial value of the variable. Since the variables are calculated without waiting for all the precursor variable to complete the calculation, which greatly improves the parallelism and calculation speed, and communication between multiple parallel tasks only once, which reduces synchronization overhead.

Introduction

The solution of sparse linear triangulation is one of the most important problems in many scientific calculations and engineering techniques. With the improvement of the scale and complexity of scientific and engineering problems, higher requirements are put forward for the solving scale and speed of the sparse triangular system [1]. At present, how to solve the sparse linear triangulation system quickly and efficiently has become the focus of research.

The rapid development of parallel computing technology provides new ideas and powerful tools to solve computationally intensive problems [2]. Sparse data structures lead to two opposite ways to partition data to solve the sparse linear triangulation solution in parallel: fine-grain partitioning and coarse-grain partitioning [3]. Coarse-grained parallel algorithm [4]: the lower triangular matrix is divided into blocks by columns, and calculated in parallel between the blocks, within a single block, the variable is solved in order of dependency. This method significantly improves the computational speed, but how to achieve an optimal blocking strategy is still a problem; fine-grained parallel algorithm [5]: the rows are divided into different layers by the dependency between the lower triangular matrix rows, and the different layers are sequentially executed, which the rows in the same layer are computed in parallel. This method effectively implements parallel solution, but frequent data transfer between layers makes communication overhead large. Aiming at the problems of the above parallel algorithms, we propose a parallel solution method based on correlation decomposition to achieve efficient solution of sparse triangle systems.

In this paper, the parallel algorithm is implemented on multi-cores processor [6]. A sparse linear triangulation system is constructed based on the Florida sparse matrix set [7], and solve by the proposed algorithm. The experimental results show that compared with Nvidia’s cuSPARSE library [9], the proposed algorithm can increase the calculation speed by about 19%. Compared with the serial algorithm, the proposed algorithm can increase the calculation speed by about 72%.

Background

Let us consider the 8×8 sparse linear lower triangular system Lx=y is shown in Figure 1.
The direct method \(^8\) solves the sparse linear lower triangular system with the Equation 1:

\[
x_j = \left( b_j - \sum_{k=0}^{j-1} L_{jk} \times x_k \right) / L_{jj} \quad (0 < j < 8)
\]  

(1)

It can be known from Equation 1 that there is an inherent dependence in solving sparse linear triangular systems. For example, if \( L_{ii} \) is a non-zero value, then the calculation \( x_j \) depends on the precursor variable \( x_i \). According to the dependence relationship of variable solution, we can construct a directed acyclic graph (DAG) represents the order of variable solution \(^9\), as shown in Figure 2. In Figure 2: a circle represents a variable, and an arrow between two circles represents a direct dependency between two variable solutions.

\[\text{Figure 1. The sparse linear lower triangular system.}\]

\[\text{Figure 2. DAG of the lower triangular matrix.}\]

**The Algorithm of Correlation Decomposition**

**Theory of Correlation Decomposition**

Consider the sparse linear trigonometric system of Figure 1. If \( b_i \) is not 0 and the other \( b_i (i = 2, 3 \cdots 8) \) are 0, the process of calculating each variable by equation 1 is as follows:

\[
x_1 = b_1 / L_{11} \quad x_2 = x_3 = x_4 = 0 \quad x_5 = (b_5 - L_{51} \times x_1 - L_{53} \times x_3) / L_{55} = -L_{51} \times x_1 / L_{55}
\]

\[
x_7 = (b_7 - L_{71} \times x_1 - L_{72} \times x_2 - L_{73} \times x_3) / L_{77} = -L_{71} \times x_1 / L_{77} \quad x_8 = (b_8 - L_{85} \times x_5) / L_{88} = -L_{85} \times x_5 / L_{88}
\]

From the above process of solving variables, we find that only when \( b_i \) is not 0, the partial value of the variables \( x_1, x_5, x_7, x_8 \) can be calculated. If \( x_1 \) is calculated, the partial values of \( x_5, x_7 \) can be calculated simultaneously, and the partial value of \( x_8 \) can be calculated by the partial value of \( x_5 \). Therefore, the above process of solving variables can be divided into two calculation processes.

Calculation process 1: \( x_1 \) calculates \( x_5 \) and then \( x_8 \) is calculated by \( x_5 \).

Calculation process 2: \( x_1 \) calculates \( x_7 \).

The two calculation processes can calculate the variable partial values in parallel. We refer to the divided calculation process as the variable calculation path. In summary, we propose a parallel solution method for linear trigonometric systems: set the right-hand-side \( b_i (i = 1, 2 \cdots 8) \) to have only one term is not 0, according to the dependence of the variable solution, the solution process of the linear triangulation system is divided into multiple variable calculation paths. First, multiple variable
calculation paths is used to calculate the partial value of the variable in parallel, then all the values of the variables are added to calculate the final value of the variables.

Taking the sparse linear trigonometric system of Figure 1 as an example, the parallel method proposed in this paper is applied to solve the problem. First, the process of solving the sparse linear triangulation system is divided into multiple variable calculation paths. Set the right-hand-side \( b(i = 1, 2, \ldots, 8) \) of the sparse triangle system in turn, only one item is not 0. According to the dependence relationship existing in the variable solution, the solution process is decomposed into 16 variable calculation paths, and the obtained paths are as shown in Figure 3.

![Figure 3. The 16 variable calculation paths.](image)

In Figure 3, we use \( N \) to represent the number of paths divided by the same variable; the variables that pass through the path are called other related variables on this path, and are represented by the set \( U \). For example, if \( b_1 \) is not 0, the variable \( x_1 \) is used as the starting point, and two variable calculation paths are divided, \( N = 2 \), which are path 1 and path 2 in Figure 3 respectively; and the set of other related variables on path 1 is \( U = \{7, 8\} \), the set of other related variables on path 2 is \( U = \{5\} \).

After that, the partial values of the variables are calculated in parallel. The 16 variable calculation paths are distributed to 16 threads, and the variable partial values are calculated in parallel between threads; in a single thread, the variable partial value is calculated according to the dependency of the variable calculation path. Specific to a single thread, the partial values are calculated as equation 2 and equation 3:

\[
x_i = b_i / L_{ii} \tag{2}
\]

\[
x_j = (-L_{ik} \times x_k) / L_{jj} \quad (j \in U) \tag{3}
\]

In equation 2: the subscript \( i \) indicates the label of the starting point on a single path; in equation 3: the subscript \( j \) represents the variable number, and the subscript \( k \) represents the precursor variable label of the variable \( x_j \) on a single path, and \( U \) is the other related variable on a single path set.

Finally, after the 16 variables are calculated in parallel, all the values of the variable \( x(i = 1, 2, \ldots, 8) \) are added, and the sum of the sums is divided by the number \( N \) of the paths defined by the variable \( x(i = 1, 2, \ldots, 8) \) as the starting point, and the final solution vector is calculated.

**Implementation of Parallel Algorithm**

Considering that the number of processors in the experimental platform is small, and the number of variable calculation paths is particularly large, this results in the serial processing of many variable calculation paths in a processor, which reduces the running speed. In order to reasonably match the number of divided paths and the number of processors, when implementing the parallel solution algorithm in this paper, the method of dividing the variables to calculate the path is as follows: sequentially set the right-hand-side \( b(i = 1, 2, 3, \ldots, n) \) of the sparse triangle system have \( M \) terms are not 0, and start from \( M \) starting points and calculate the partial value of the variable. Then, \( M \) initial values need to be calculated in a single thread, the first starting value is calculated by equation 2, and the
other $M-1$ starting values are calculated by equation 4; equation 3 is transformed into equation 5, which calculates the partial values of other related variables within a single thread:

$$x_t = (b_t - \sum_{k=1}^{l} L_{tk} \times x_k) / L_{tt} \quad (t \in U_t)$$

(4)

$$x_j = (\sum_{k=1}^{l} L_{jk} \times x_k) / L_{jj} \quad (j \in U)$$

(5)

In equation 4, the subscript $t$ represents the other $M-1$ starting variable numbers, and $U_t$ is the set of other $M-1$ starting variables. In equation 5, the subscript $j$ represents the variable number, and $U$ is a collection of other related variables within a single thread.

Implementation of the correlation decomposition algorithm. First, in the main function, the solution process of the sparse triangle system is divided into several calculating paths, and each calculation path is arranged by one thread’s calculation. For convenience of description, let the order of the triangular matrix $L$ be $n$, the number of threads is set to $p$, and $p$ can be divided by $n$, $m=n/p$. The right-hand-side is divided into groups of $m$ values, and the division result is shown in Fig. 4. Fig. 4 divides the right-hand-side into $p$ blocks, each block is numbered $0, 1, 2, \ldots p-1$, and each block is dispatched by the operating system into one thread for calculation, and all threads execute in parallel.

$$B^T = \{b_1, b_2, \ldots, b_m, b_{m+1}, b_{m+2}, \ldots, b_{2m}, b_{2m+1}, b_{2m+2}, \ldots, b_{3m}, \ldots, b_{m-1}, b_m\}$$

Figure 4. Division of the right end item B.

Specific to a single thread, through the dependency of the variable solution, the last column is found from the starting column to form a variable calculation path, and the partial value of the variable is calculated simultaneously. The description of the calculation process within a single thread is shown in Figure 5.

**Calculation within a single thread**

**INPUT:** value, col_ind, row_ptr, Bi, i  
**RETURN:** x  
1: x[i*m]=Bi[i*m]/value[i*m] /* calculates the first starting value */  
2: For t=i*m+1 TO t=(i+1)*m Do  
3: For p=0 TO p=number Do /* number represents the number of predecessors of the variable $x_t$ in this thread*/  
4: SUM=value[ref_ptr[t]+p]*x[col_ind[ref_ptr[t]+p]]+SUM /* summation in Equation 6 */  
5: EndFor  
6: x[t]=(Bi[t]-SUM)/value[t] /* calculates the value of the other m-1 starting variables $x_t$ */  
7: EndFor  
8: For j=(i+1)*m+1 TO j=n Do  
9: For p1=0 TO p1=count Do /* count represents the number of previous variables of this thread variable $x_j$ */  
10: if x[col_ind[ref_ptr[j]+p1]]=0 then  
11: sum=sum-value[ref_ptr[j]+p1] /* x[col_ind[ref_ptr[j]+p1]] *//* summation in Equation 7 */  
12: EndIf  
13: EndFor  
14: x[j]=sum/value[j] /* calculates the value of other related variables $x_j$ in this thread. */  
15: EndFor  

Figure 5. Description of the calculation process within the thread.

Then, after the $p$ threads are calculated, all the calculation results are returned to the main function, and all the values of the variables are added to calculate the final solution vector.

**Case Analysis**

The parallel algorithm adopts Java programming, and chooses the main memory of 16GB Intel(R) Core(TM) i7-4770 3.40 GHz, 4-core 8-thread hardware platform. With the support of Windows
operating system, the Callable interface is applied to solve the sparse linear triangle system. And we evaluate our algorithm against the state-of-the-art cuSPARSE algorithm.

In this paper, a Florida matrix with different sparsity is used to form a sparse linear lower triangular system. Table 1 is a Florida matrix with a relatively small sparsity, and Table 2 is a Florida matrix with a very small sparsity.

<table>
<thead>
<tr>
<th>matrix name</th>
<th>rows/columns</th>
<th>sparsity(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pwtk</td>
<td>217918</td>
<td>0.02</td>
</tr>
<tr>
<td>SiO2</td>
<td>155331</td>
<td>0.047</td>
</tr>
<tr>
<td>shallow_water1</td>
<td>81920</td>
<td>0.69</td>
</tr>
<tr>
<td>finan512</td>
<td>74752</td>
<td>0.24</td>
</tr>
<tr>
<td>cant</td>
<td>62451</td>
<td>0.86</td>
</tr>
<tr>
<td>gridgena</td>
<td>48962</td>
<td>1.12</td>
</tr>
<tr>
<td>wathen100</td>
<td>30401</td>
<td>1.66</td>
</tr>
<tr>
<td>gyro_m</td>
<td>17361</td>
<td>8.39</td>
</tr>
</tbody>
</table>

Table 2. Florida matrix with very small sparsity.

<table>
<thead>
<tr>
<th>matrix name</th>
<th>rows/columns</th>
<th>sparsity(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>thermomech_dM</td>
<td>204316</td>
<td>0.0039</td>
</tr>
<tr>
<td>xenon2</td>
<td>157464</td>
<td>0.016</td>
</tr>
<tr>
<td>shallow_water1</td>
<td>81920</td>
<td>0.0061</td>
</tr>
<tr>
<td>finan512</td>
<td>74752</td>
<td>0.012</td>
</tr>
<tr>
<td>cant</td>
<td>62451</td>
<td>0.11</td>
</tr>
<tr>
<td>gridgena</td>
<td>48962</td>
<td>0.023</td>
</tr>
<tr>
<td>wathen100</td>
<td>30401</td>
<td>0.054</td>
</tr>
<tr>
<td>gyro_m</td>
<td>17361</td>
<td>0.34</td>
</tr>
</tbody>
</table>

Comparison of Algorithms

In this paper, we use the matrix with small sparsity in Table 1 and the matrix with very small sparsity in Table 2 to form a linear trigonometric system and solve it. In order to reflect the advantages of the proposed algorithm, run a 4-core processor and set 8 threads, and then compare the algorithm of correlation decomposition with the serial algorithm and the cuSPARSE algorithm. After many experiments, the results in Fig. 6 were obtained.

From the graph (a) in Figure 6, it can be clearly seen that compared with the serial algorithm and the cuSPARSE algorithm, the correlation decomposition algorithm has the smallest calculation time and the fastest calculation speed. Compared with the matrix adopted in graph (a), the matrix adopted in graph (b) has the same or similar order, but the sparsity decreases. The calculation time of the three methods are greatly reduced, however, compared with the serial algorithm and the cuSPARSE algorithm, the calculation time of the algorithm in this paper is still minimal, and the calculation speed is still the fastest.
(b) Calculating time for very sparse triangle systems

Figure 6. Calculating time comparison of three different algorithms.

It can be seen that whether the matrix of sparse degree is adopted in graph (a) or matrix of very small sparse degree is adopted in graph (b), the calculation speed of the correlation decomposition algorithm is obviously better than that of serial algorithm and the cuSPARSE algorithm.

Conclusion

In this paper, a sparse linear trigonometric system is constructed based on the Florida sparse matrix set, and the correlation decomposition algorithm is applied to solve those systems. Through comparison with the serial algorithm and the cuSPARSE algorithm, the efficiency of the parallel algorithm in this paper is verified.

Acknowledgements

The work was supported in part by Natural Science Foundation of Hebei province of China under Grant No. F2017502043.

References


