Implementation of Parallel Matrix Multiplication Algorithms Based on Distributed System Using MPI

May Thiri Win
University of Computer Studies (Banmaw)
maythiri123@gmail.com

Khin Soe Moe Kyaw
University of Computer Studies (Banmaw)
khinsoemoekyaw@gmail.com

Abstract

Matrix computations are basic to a wide variety of problems in scientific and engineering fields. Due to the complexity of some scientific problems, it needs to process a large-scale dense matrix which brings great challenges to compute in a reasonable time. Nowadays, High Performance Computing (HPC) technologies are promising the ability to process a large amount of data with high speed. This paper proposed the implementations of parallel algorithms for matrix multiplication based on distributed system. The experimental results helps to reduce the computational time of large-scale matrix multiplication, and may be useful for the development of technical systems which are essentially need to use matrix multiplication calculation.

Keywords: *MPI, Large-scale matrix, Parallel matrix multiplication algorithm, Distributed computing.*

1. Introduction

Matrix calculations are widely used in a wide variety of process, mathematical modeling, phenomena and systems [3]. Matrix multiplication is the basic operation of linear algebra and the dominant computational part of many scientific applications, such as solving linear equations, integrating systems of differential equations of both ordinary and partial derivatives.

A large amount of software products for processing matrices is constantly growing and new economical storage structures for matrix are being developed, various highly machines, dependent implementations of algorithms are being created, theoretical studies are conducted to search for faster matrix calculation methods [4]. Due to the complexity of modern scientific problems and modeling systems, some models are needed to setting up large scale and its cause to deal with a very large matrix and, lead to costs a lot of time. For such large-scale matrix calculations problems, a single thread of executions cannot be used effectively. The growth of computational power of computer systems, the emergence of modern supercomputer, workstation cluster made it possible to solve many discrete mathematics problems that require large amount of computation to be performed in a reasonable operation time.

Nowadays, parallel computing technologies are widely used in various scientific fields such as quantum physics, modular physics, plasma physics, biology, ecology, social science, computer science, medicine, industry, etc. However, developing parallel applications that are robust and provide good speed-up across current and future multiprocessors is a challenging task [1]. The key point of using parallel computing techniques is to develop the compactable algorithm accordingly to the problem and computing architectures. In this paper, the parallel algorithms of matrix multiplication based on distributed computing system are described. The proposed algorithms are implemented using Message Passing Interface (MPI), the parallelization method most commonly used for distributed memory architectures.

2. Matrix Multiplication Algorithm

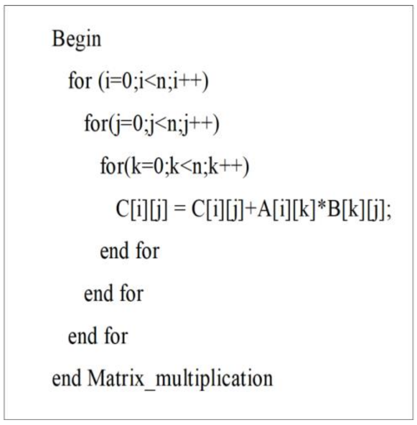
Let's $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times 1}$. Then the multiplication of these two matrices $C = AB$ can be expressed as follow:

$$c_{ij} = \sum_{k=0}^{n-1} a_{ik}\, b_{kj}, 0 \le i \le m, 0 \le j \le l \qquad (1)$$

As described in (1), the sequential matrix multiplication algorithm includes three nested loops and the pseudo code can be described as shown in figure 1.

Each element of the resulting matrix C is a scalar product of the corresponding rows of matrix A and column of matrix B:

$$c_{ij} = a_i, b^T{}_j\,, a_i = a_{i0,}a_{i1,} \dots, a_{i(n-1),}$$
$$b^T{}_j = b_{0j,}b_{1j,} \dots, b_{(n-1)j}{}^T$$



```
Begin
  for (i=0;i<n;i++)
    for(j=0;j<n;j++)
      for(k=0;k<n;k++)
        C[i][j] = C[i][j]+A[i][k]*B[k][j];
      end for
    end for
  end for
end Matrix_multiplication
```

Figure 1. Pseudo code of matrix multiplication algorithm

Matrix multiplication algorithm can be considered as iterative and focused on the sequential calculation of the row of matrix C. When performing one iteration of the inner loop, the first element of the first row of the resulting matrix is calculated, and after completing the one iteration of the outer loop, the complete elements of the first row of matrix C is calculated.

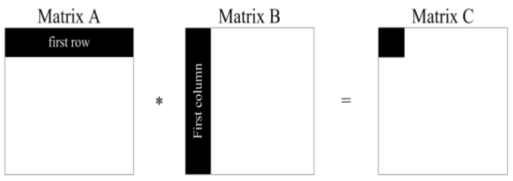


Figure 2. First iteration of the inner loop are used to calculated the first element of the first row of the resulting matrix

Since each element of the resulting matrix is a scalar product of the row and column of the original matrices, to calculate all elements of the matrix C of size $n \times n$, it is necessary to perform $n^2(2n - 1)$ scalar operations and the computational time.

$$T_1 = n^2.(2n - 1).\tau$$

Where τ is the execution time of one elementary scalar operation. Matrix multiplication algorithm involves performing $n.m.l$ multiplication and the same number of calculating addition operations of the elements of the original two matrices. For simplicity, we will assume that both matrix A and B are square. An analysis of possible ways of parallel execution of this task can be carried out by analogy with the consideration of the problem of matrix-vector multiplication.

3. Parallel Matrix Multiplication Algorithms

A large-scale matrix multiplication problem requires a large number of operations (scalar multiplication and additions) to solve it. The resulting vector involves repeating the same type of operations by multiplying the rows and the vector of the matrix. When constructing parallel methods for performing matrix multiplication, along with the consideration of sets of rows and columns, the block representation of matrices is widely used. The choice of matrix separation method leads to the determination of a specific parallel computing method; the existence of different data distribution schemes generates a number of parallel matrix computing algorithms [2].

Let's the number of processor be $p = k^2$, the number of row and column of the matrix is $k = \sqrt{p}$, i.e. $n = mk$. Imagine the original matrices A, B and the resulting matrix C in the form of sets of rectangular blocks of size $m \times m$. Then the operation of matrix multiplication and in block form can be represented as follow:

$$\begin{pmatrix} A_{11} & A_{12} & \ldots & A_{1k} \\ \cdots & \cdots & \cdots & \cdots \\ A_{k1} & A_{k2} & \ldots & A_{kk} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} & \ldots & B_{1k} \\ \cdots & \cdots & \cdots & \cdots \\ B_{k1} & B_{k2} & \ldots & B_{kk} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} & \ldots & C_{1k} \\ \cdots & \cdots & \cdots & \cdots \\ C_{k1} & C_{k2} & \ldots & C_{kk} \end{pmatrix}$$

Where each blocks C_{ij} of the matrix C is determined in accordance with the expression:

$$C_{ij} = \sum_{l=1}^{k} A_{il} B_{lj}$$

When analyzing the matrix multiplication in the form of the calculations of the blocks, a possible approach for parallel computing can consists in allocating different processors for calculations related to obtaining separate blocks. Using this approach allows to get effective parallel methods for multiplying bock-represented matrices.

In this paper, the analysis of parallelization methods will be supplemented by considering the organization of parallel computing depending on the number of processors available for use and also will be developed using the following two facts:

- the operations of multiplying the individual rows of the first matrix by a column vector of second matrix when performing calculations are independent and can be performed in parallel;
- multiplying each row by the whole matrix includes independent multiplication operations and can also be performed in parallel.

The maximum required number of processors is determined by the value:

$$p = n^2$$

The use of such a number of processors Q can be divided into n groups:

$$Q = \{Q_1, Q_2, Q_3, \dots, Q_n\}$$

Each computational group is performing computational tasks in parallel and the results are recombining into the root node. Following the factors described above, parallel matrix multiplication algorithms can be considered into two ways: (1) the method based on splitting both two matrices into the blocks and (2) the method in which only first matrix is splitting into the blocks which are accordingly to the number of computing nodes.

3.1. The First Parallel Matrix Multiplication Algorithm α

Parallel matrix multiplication performing on distributed computing offers two possible ways. The first method is based on the idea of splitting both matrices into the blocks accordingly to number of computing nodes. From the definition of matrix multiplication, it follows that the calculation of all elements of the matrix C can be performed independently. As a result, a possible approach for parallel computation is used as a base subtasks procedures for determining one element of the result matrix C. To carry out all the necessary calculations, each subtask must perform calculations on the elements of one row of the matrix A and one column of the matrix B. The total number of subtasks obtained with this approach is equal to n^2 (according to the number of elements of the matrix C).

The division of rows and columns into the blocks in most cases occurs on a continuous basis. With this approach, for horizontal row splitting, for example, matrix B can be represented as:

$$B = \left(B_0, B_1, \dots, B_{p-1}\right)^T, B_i = \left(b_{i_0}, b_{i_1}, \dots, b_{i_{k-1}}\right),\ i_j = ik + j, 0 \le j \le k, k = m/p$$

Where $b_i = (b_{i1}, b_{i2}, \dots, b_{in}), 0 \le i < m$ is ith row of matrix B. The computing diagram of the first parallel matrix multiplication algorithm can be described as follow:

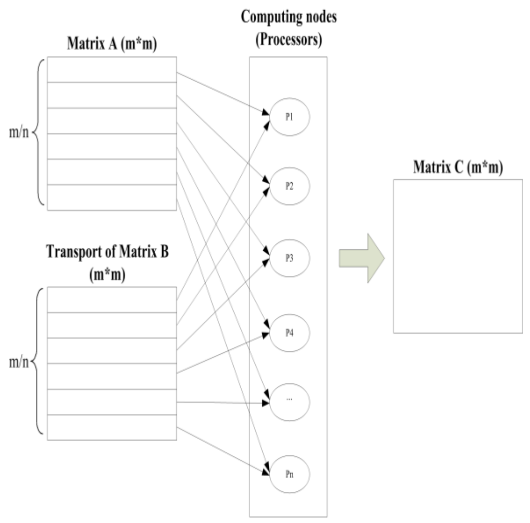


Figure 3. The computing diagram of parallel matrix multiplication algorithm α

3.2. The Second Parallel Matrix Multipli-cation Algorithm β

The second method is considered as the process of solving the independent subtasks of matrix multiplication by the row and the whole columns. In this method, only first matrix is needed to split into the blocks and can reduces the total number of subtasks to n. To perform all the necessary calculations, one of the rows of matrix A and all columns of matrix B must be available in all subtasks. The key point of this method is spreading the matrix B to all subtasks. Each node performs a calculation based on the row and the whole matrix calculation. The computing diagram of the second parallel matrix multiplication algorithm can be described as follow:

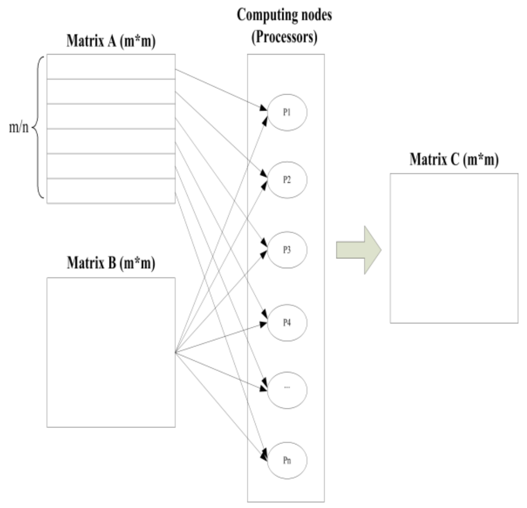
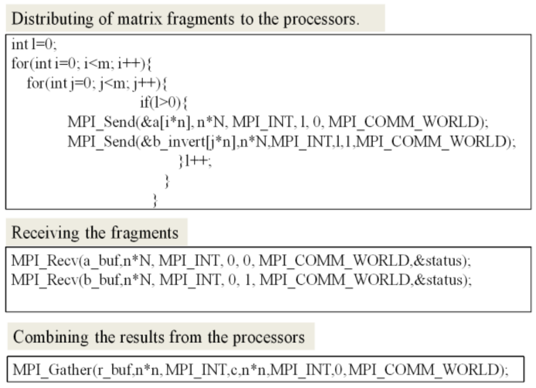


Figure 4. The computing diagram of parallel matrix multiplication algorithm β

3.3. The Implementation of Parallel Matrix Multiplication Algorithms

The implementations of proposed algorithms are focused on the use of parallel computing systems with distributed memory. MPI is used to organize the interaction of processors; to distribute the elements of the multipliable matrices between the processor, when the rows of the matrix A and columns of matrix B are sent to each processor simultaneously. The root note performs splitting the matrices into the blocks, sending and receiving to the computational nodes. To implement these operations MPI_Sent and MPI_Recv functions are used, which are standard routine of the MPI, use to send and receive the message by the destination process. Each computing node receives the corresponding parts from both matrices and performs the matrix multiplication calculation. The results of each computational operation of processors are recombined in the root node using MPI_Gather function.



Distributing of matrix fragments to the processors.

```
int l=0;
for(int i=0; i<m; i++){
   for(int j=0; j<m; j++){
                              if(l>0){
           MPI_Send(&a[i*n], n*N, MPI_INT, l, 0, MPI_COMM_WORLD);
           MPI_Send(&b_invert[j*n],n*N,MPI_INT,l,1,MPI_COMM_WORLD);
                                    }l++;
                                 }
                              }
```

Receiving the fragments

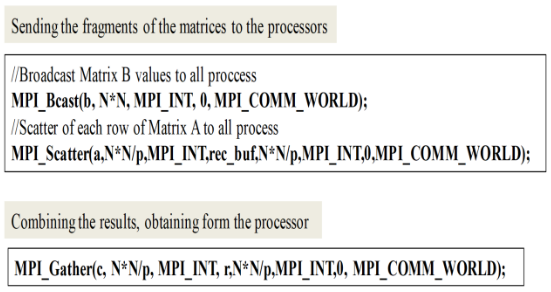
```
MPI_Recv(a_buf,n*N, MPI_INT, 0, 0, MPI_COMM_WORLD,&status);
MPI_Recv(b_buf,n*N, MPI_INT, 0, 1, MPI_COMM_WORLD,&status);
```

Combining the results from the processors

```
MPI_Gather(r_buf,n*n,MPI_INT,c,n*n,MPI_INT,0,MPI_COMM_WORLD);
```

Figure 5. The implementation of parallel matrix multiplication algorithm α

In the second algorithm, only the fist matrix is needed to split accordingly the number of processors. In this case, MPI_Scatter function is used to send data from one process to all other process in a computation environment. To be available the whole matrix B in all subtasks, MPI_Bcast function is used, a routine that broadcasts a message from the process with rank root to all processes of the group, itself included.



Sending the fragments of the matrices to the processors

```
//Broadcast Matrix B values to all proccess
MPI_Bcast(b, N*N, MPI_INT, 0, MPI_COMM_WORLD);
//Scatter of each row of Matrix A to all process
MPI_Scatter(a,N*N/p,MPI_INT,rec_buf,N*N/p,MPI_INT,0,MPI_COMM_WORLD);
```

Combining the results, obtaining form the processor

```
MPI_Gather(c, N*N/p, MPI_INT, r,N*N/p,MPI_INT,0, MPI_COMM_WORLD);
```

Figure 6. The implementation of parallel matrix multiplication algorithm β

4. Experimental Results

The experiments are performed with N = 200,000 on a computer with two intel@Xeon® E5335 processors (2.0) GHz, (8) MB cache and (4) GB RAM, 8 computing cores. Firstly, the run time of a simple serial matrix multiplication algorithm is estimated. After increasing the size of the matrices 1024, the computational time continues to grow linearly. The result of computational experiments of sequential algorithm of matrix multiplication is shown in figure 7.

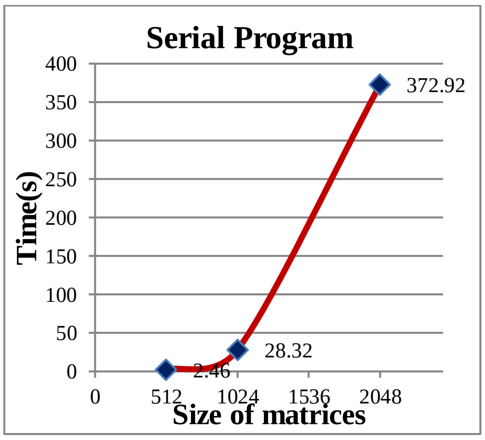


Figure 7. The computational time of serial matrix multiplication algorithm

Parallel matrix multiplication algorithm is processed using 16 processors. When performing the first parallel algorithm, each sub blocks of matrices are sent to the nodes of the computational environment, the receiving blocks are calculated element-wise multiplication. The difference of the second algorithm is that in the subtasks there are matrix-vector multiplication is implemented.

In the process of implementation parallel algorithms, the following critical parameters of the developed program should be taken into account – time characteristics, i.e. the time required to implement the task, the total amount of memory occupied by all fragments of the matrices, the size of the actual program implementing this method (table 1).

Table 1. Comparison of the computing time of the implementations of the parallel matrix multiplication algorithms

Matrix Size	First method α			Second method β		
	Number of processors					
	4	16	64	4	16	64
512	0.98	0.45	0.88	0.62	0.29	0.51
1024	7.84	3.52	4.82	6.79	2.78	37.80
2048	63.12	27.9	31.13	93.15	37.81	49.48

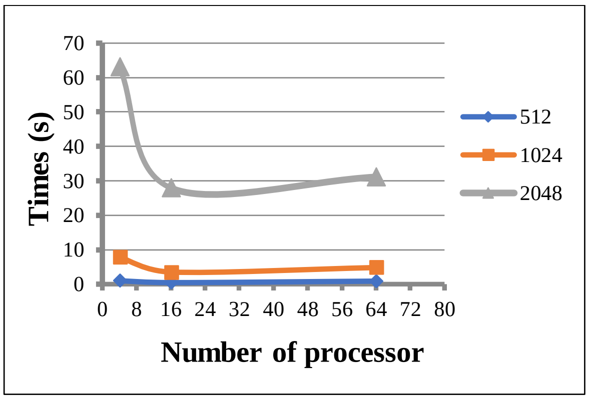


Figure 8. The computational time of implementation of matrix multiplication algorithm α

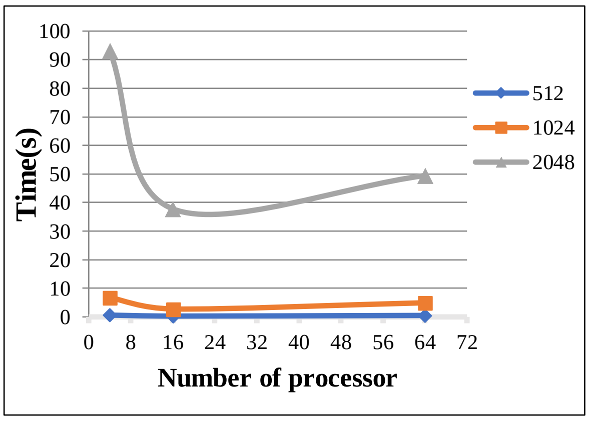


Figure 9. The computational time of implementation of matrix multiplication algorithm β

5. Conclusion

In this paper, the two types of parallel matrix multiplications algorithm based on distributed system are proposed. The first algorithm is considered in the operations of multiplying the individual rows and columns of the matrices, calculations are performed in parallel. The idea of the second algorithm is based on multiplying each row by the whole matrix includes independent multiplication operations and can also be performed in parallel. The implementations are carried out for various matrix sizes and various numbers of processor on a multi-processor system using MPI. The experimental results show that program execution time is inversely proportional to the number of processors. Increasing the number of processors reduces the execution time of the parallel algorithm. Accordingly to the results, it can be assumed that such computationally complex problems as matrix multiplication can be successfully implemented on multiprocessor parallel system using the proposed algorithms.

Acknowledgement

The author would like to express her gratitude to Dr. May Phyo Oo, Pro-Rector of Computer University (Banmaw) and editorial board of UJCIR, for giving a chance to submit paper. The author is also grateful to the reviewers for their useful comments. Finally, the author thanks her colleagues for their suggestions and cooperation.

References

[1] D.Culler, J.P.Singh, A.Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*, Morgan Kaufmann Publishers Inc, San Francisco, California, 1999.

[2] M.J.Chorley & D.W.Walker. (2010). Performance analysis of a hybrid MPI/OpenMP application on multi-core clusters. Journal of Computational Science, 1(3), 168-174.

[3] N.E.E. Theint, S.Soe. “Application of Matrices in Human’s Life”. International Journal of Science and Engineering Applications. Volume 8–Issue 10, 2019, pp. 438-443.

[4] O.D.Joukov, N.D.Rishe “Computer Technology for Solving Large Scale Matrix Problems”. Computational Science and Its Applications — ICCSA 2003. vol 2667. Springer, Berlin, Heidelberg, pp. 848-854.