An Algorithm for solving massive matrix inversion in cloud computing systems

Do Hyun Bae
School of Computer Sciences,
University of Seoul,
Seoul, Korea
ilsoo99@uos.ac.kr

Munkhbayar Bayartsogt
School of Computer Sciences,
University of Seoul,
Seoul, Korea
muba@uos.ac.kr

Jin Suk Kim
School of Computer Sciences,
University of Seoul,
Seoul, Korea
kimjs@uos.ac.kr

ABSTRACT

In this paper we introduce a parallel approach to calculate massive matrix inversion. It needs large size of memory to compute with large size of matrices. Because of the memory requirements, we consider an algorithm to optimize memory distribution in cloud computing system. In matrix inversion using Gauss-Jordan algorithm, we found out a lot of regional memory access tendency in the algorithm. We also consider this memory access tendency. To solve these problems, we divide the matrix data as many as numbers of processors which was assigned to calculate matrix inversion. Dividing directions both horizontal and vertical are possible to imply. Matrix inversion has steps, and this step is increase according to the size of the matrix, and previous step calculation results are used at each step calculation results. To do above process, we use a parallel scheduler. Parallel scheduler manages the all processors and synchronizes these processors calculation. Research is focused on solving massive matrix inversion, so we test our research in cloud computing system, and we obtain the progress results.

Categories and Subject Descriptors

C.m [**Computer Systems Organization**]: Miscellaneous

General Terms

Algorithms, Measurement, Performance, Design, Experimentation.

Keywords

Matrix inversion, Gauss-Jordan algorithm, Cloud computing, Distributed computing system.

1. INTRODUCTION

Solving massive matrix inversion is a matter of common interest in research of science and technology area. However, solving massive the matrix inversion needs much resources and time according to the size of matrix. If service a cloud server which calculating massive inverse matrix, many researchers can save much effort to their researches. Recently, social network service is a popular issue, and we are interested in relations between the network and the users. When we imply matrix inversion to social network service then we can find out a relation with users, but this work has not researched yet. In size matter, the number of users who use social network is increase. For example, twitter site obtained 41.7 million user profiles, 1.47 billion social relations, 4262 trending topics, and 106 million tweets as of July 2009, and has been growing fast [13]. Dealing with massive matrix inversion needs complex operation and long operation time. Moreover, because of the memory requirements, it is not easy to process in single. Many researchers present many optimize algorithms and methods. Gauss-Jordan algorithm is one of the optimize algorithm to solve the matrix inversion, and many researchers imply this algorithm to reinforce their research. This Gauss-Jordan algorithm makes the matrix data to linear equation [8]. In other words, we can divide the matrix data by row or column directions. In this works, Block based Gauss-Jordan algorithm [2], [5], [9], [10], [11] is an applied algorithm using Gauss-Jordan algorithm. They divide the matrix by both horizontal and vertical directions such as a chessboard. This algorithm focus on the memory distribution to reduce memory occupancy, but it makes complex dependencies between the blocks [6]. We modify this Block based algorithm, and to solve massive matrix inversion we focused on reducing the dependency between distributed matrix data, and we implement Gauss-Jordan algorithm and use cloud computing system. Also we found out memory access locality. In other words, if we imply chessboard block, some of blocks are used only few steps. We considered distributing the matrix with a set of having same memory access time, so we divide the matrix by processors number, and the dividing direction is only one side. Each processor is connected with Ethernet network, and they communicate each other through the server. In m by m matrix, they exchange a set of data m times. We implement our idea in cloud computing systems, and task-scheduler manages the whole process such as managing processors. Cloud computing is a new paradigm for large scale distributed computing [3]. Actually, definite explanation about the cloud computing has not been settled yet. Doerksen [15] defines that the cloud computing is the user-friendly version of Grid computing. Vaquero et al. [7] studied the cloud computing definition. In that paper, they have been studied more than 20 definitions. Examples of that study, the cloud computing is about User Friendliness, Virtualization, Internet Centric, Variety of Resources, Automatic Adaptation, Scalability, Resource Optimization, Pay per Use, Service SLAs, and Infrastructure SLAs. These features are related with utility computing and service. Also Cloud Computing lies at the large-scale side. Cluster Computing has been more focused on traditional non-service applications. Grid Computing overlaps with all these fields where it is generally considered as lesser scale than Cloud computing system [3]. Summarize the various definitions, the cloud computing system is large scale distributed computing system, and the system is designed as service oriented. Moreover users pay as much as they borrow the servers or services. To implement our algorithm to cloud computing system, we imply a parallel task scheduler. Kim [16] used this scheduler to his research. We named this scheduler "ParTS (Parallel Task Scheduler)". Figure 1 is the configuration of ParTS. ParTS server manages processors and whole inversion process. Because we divide the matrix as many as process numbers and distribute the partitioned matrix data to processors, ParTS Sever have these information to manage the whole process. ParTS Worker manages processor's internal operations and communicates with server. Task Requester provides user interface. All these three parts are connected with Ethernet network.

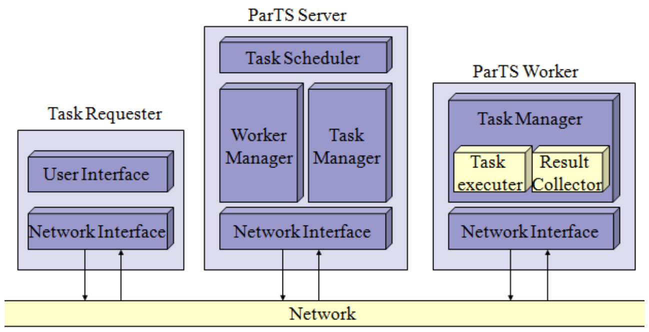

Figure 1: ParTS configuration.

2. RELATED WORKS

In many science research areas, they using system of linear equations applications such as Digital Signal Processing, Geometry, Networks, Temperature Distribution, Heat Distribution , Chemistry, Linear Programming, Games, Estimation, Weather Forecasting , Economics, Image Processing, Video Conferencing, Oceanography and many Statistical analysis [8]. Process of the solving system dealing with linear equation is simple to understand, and it is easy to implement. Gauss-Jordan algorithm deals the matrix inversion into linear equation. Therefore, Gauss-Jordan algorithm is used popularly. Also, we have interests in social network. For example, Newnan [12] imply the matrix inversion to measure centrality of a node in a network. They comment that the calculation is tractable for networks up to about 10000 vertices using typical desktop computing system. Because of the memory requirements and time complexity to solving massive matrix inversion, it is hard to process in single computer. Therefore, we focused on dealing with massive matrix inversion in cloud computing systems. There are some algorithms to solve the massive matrix operation, and they tested that algorithm in various computing system. We focused on memory distribution, and also we consider reducing calculation time. Below subsections describe related works, and we modify them.

2.1 Block based Gauss-Jordan Algorithm

Block-based Gauss-Jordan algorithm [5] is a classical method dealing with massive matrix inversion, and it can be used in weather prediction, aircraft design, and graphic transformation and so on. This algorithm shows efficient memory utilization. Block-based Gauss-Jordan algorithm has some characters. First, matrix data is divided by blocks such as a chessboard. Because of this dividing method, it occur data dependence between different blocks. Second, the numbers of steps are equal to the number of blocks divided into, but total calculation time is not changed compare with basic Gauss-Jordan algorithm calculation. Figure 2 is an example of Block based Gauss-Jordan algorithm.

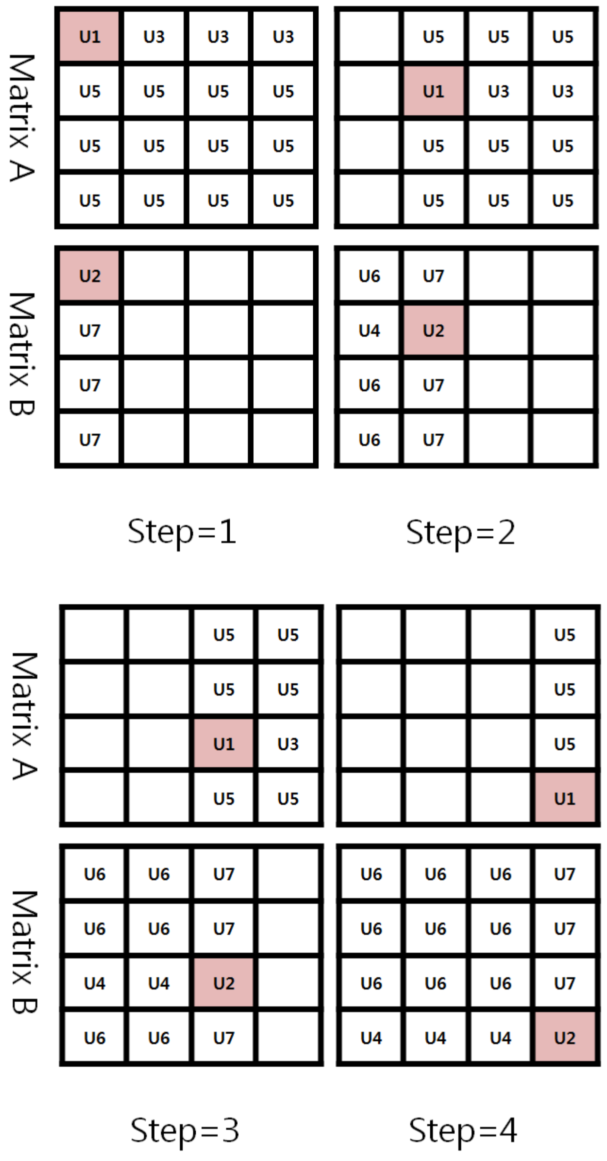

Figure 2: Matrix updating for 4x4 blocking.

Matrix A and B are NxN matrix, and matrix A is the original matrix. Matrix B will be the result of the matrix inversion of A. Matrix A and B are partitioned into a matrix of 4×4 blocks, and one block is nxn matrix(n=N/S). This algorithm is consists of two large processes: one is inter-process and the other is intra-process. Inner-process is similar with basic Gauss-Jordan algorithm. The annotation 'U' denotes the block update. U1, U3, and U5 are processed in matrix A, and U2, U4, U6, U7 are processed in matrix B. U3 to U7, intra-process is occurred. In other words, blocks which denoted U3, U4, U5, U6, and U7 at Figure 2 needs other block's data. Figure 3 denote the data dependencies in intra-process. U1 and U2 data is needed to U3, and the result of U3 is needed to process U5 at the matrix A. Otherwise, in matrix B, U1 and U2 data is needed to update4, and the result of U4 is needed to process U6.

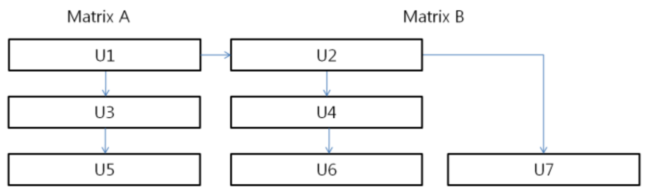

Figure 3: Data update dependency in Intra-process.

Although Black-based Gauss Jordan algorithm presents efficient memory utilizations by nxn block distribution, it shows complex data dependency. This algorithm running time is O(N^3) and following equation shows details.

$$\mathrm{b} \times (\mathrm{n}\beta \times b^2 \times \left(\frac{\mathrm{n}}{\mathrm{b}}\right)^2) \qquad (1)$$

In above equation, b means the number of blocks in b X b blocks, and β means data communication time between blocks. It eventually shows as following equation.

$$\mathrm{bn}^3\beta \qquad (2)$$

This equation shows that block based Gauss-Jordan algorithm don't guarantee the time performance. Moreover, to adapt this algorithm to cloud computing system, hard efforts are needed. In other words it is not easy to adapt to cloud computing system. In section 3, we suggest an algorithm to solving massive matrix inversion in cloud computing system. We modify this Block based Gauss-Jordan algorithm, and we are focused on reducing communication complexity and data dependency.

3. PROCESSOR BASED GAUSS-JORDAN ALGORITHM

We modified Block based Gauss-Jordan Algorithm [2], and we named it as Processor based Gauss-Jordan Algorithm. This Algorithm is similar with Block based Algorithm, but we considered at reducing the complexity and data dependency. First of all, we divide the matrix as many as processors numbers, and dividing direction is only one of horizontal and vertical. See Figure 4. This figure is an example data distribution of mxm matrix with p processors. Divide the matrix only one direction to reduce the data dependencies between the data partitions. In Block based algorithm, they divide the matrix both horizontally and vertically such as a chessboard, but this dividing method makes it more complex. Because of the intra-process, each block must communicate with other blocks every step. Therefore, to reduce the communication time between blocks, we distribute partitioned data to the processors which are assigned in cloud computing system. In Gauss Jordan algorithm, the memory access time is different according to matrix location. Figure 5 is an example which is represents the cumulative access according the matrix location at last step. This is an example 8x8 matrix inversion using Gauss Jordan algorithm.

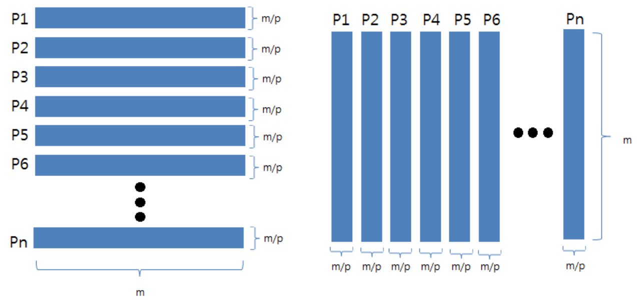

Figure 4: Matrix data distribution in PbGJ algorithm (m = size of the matrix, p = number of processors, P = processor assigned).

In matrix A which has original matrix data, a set of right side of the matrix data is accessed until the end step, but a set of left side of the matrix data is accessed only at the first step. On the other hand in matrix B, a set of left side of the matrix data is accessed until the end step, but right side of the matrix is accessed at the last step. Because of this access tendency, when we divide matrix both horizontally and vertically like as Block based Gauss Jordan, some blocks which are located left side of the matrix A are less accessed than right side of the blocks. To destroy this tendency, we assign both matrix A and matrix B data partitions to each processor.

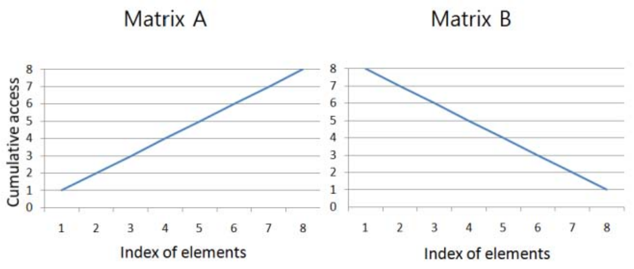

Figure 5: Cumulative access time at step 8 in 8x8 matrix.

We divide the original matrix as many as processors numbers, and assign the partition to each processor. If merge the whole partitions which is owned by processors, it is same as original matrix. Algorithm 1 is shows PbGJ procedure. In Processor based Gauss Jordan algorithm, there is only one main processor, and this main processor is changed according to matrix data. ParTS server has information about matrix distribution and which processor has the data must be used in that step. Therefore, ParTS server decides the main processor. This main processor sends a set of data to ParTS server, and then server send this data set to all processor. All processors receive this data set, and using this received data set, each processor doing matrix operation. First of all, main processor makes the first element to 1, and other elements located same line divide by first element. The main processor sends a set of data to ParTS server, and ParTS server sends the data set to all processor. If a processor receives the data, this processor starts matrix operation. According to step number, the main processor is changed. ParTS server has the information about the matrix index which is assigned to the processors. Then, ParTS server decides the main processor according to the step number. We tested as step number and matrix index equally. According to the status above, ParTS server manages the whole processing.

Algorithm 1: Processor based Gauss-Jordan

```
For s = 1 to m do
(Main processor)
    For j = s to m do
        A(s,j) = A(s,j) / A(s,s)
    End_For
    For j = 1 to s do
        B(s,j) = B(s,j) / A(s,s)
    End_For
    send follow data to all processors
      1.  s (step no.)
      2.  For i = 1 to m A(s, i) End_For
      3.  For i = 1 to m B(s, i) End_For

(All processors)
    For k=1 to m do
        For j = s to m do
                A(k,j) = A(k,j) - A(s,j)*A(k,s)
        End_For
    End_For
    For k=1 to m do
        For j = 1 to s do
                B(k,j) = B(k,j) - B(s,j)*B(k,s)
        End_For
    End_For
End_For
```

Figure 6 is an example of 8x8 matrix inversion using PbGJ. Step 5 at the figure shows the process that changing main processor. Although main processor is changed, other processors don't care about this change. Since all processor receive data set from ParTS server, only main processor, which was changed, sends correct data set. In nxn matrix, step n is the last one. If step reach the last, matrix B, which one processor owned, is a set of fragments of the inverse matrix result. On the other hand, matrix A is a set of fragments of identity matrix. If merge the data of the matrix B, that is the matrix inversion of matrix A.

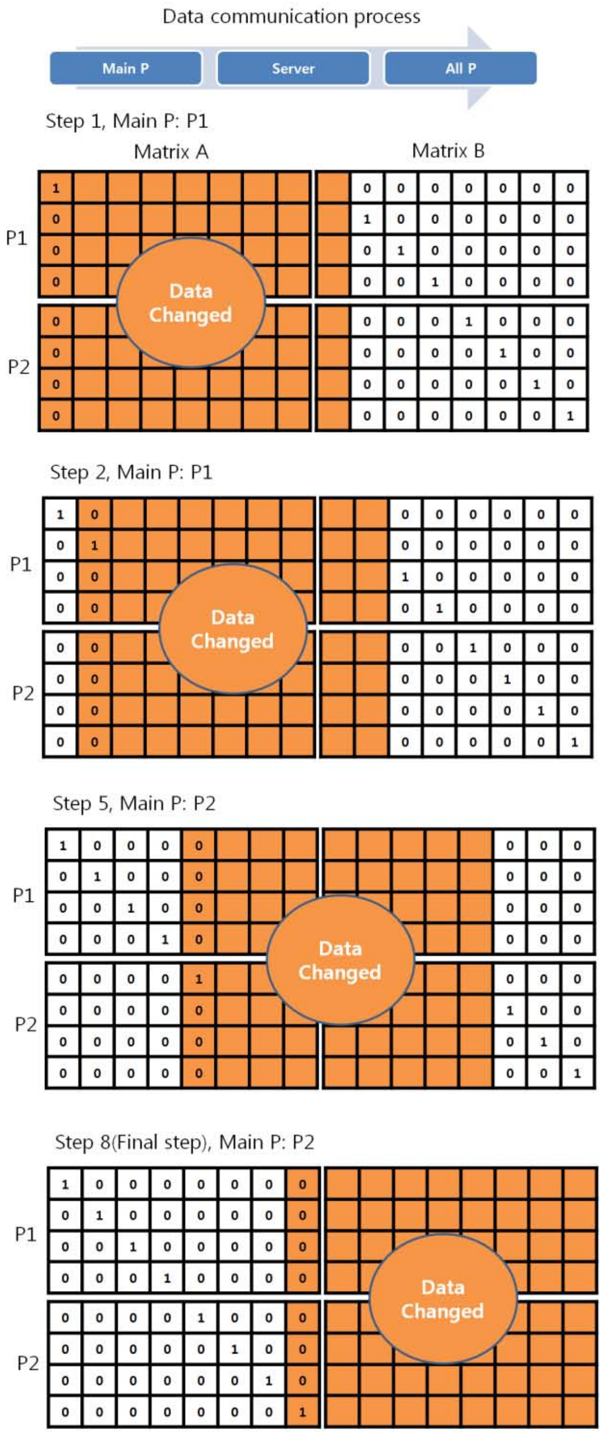

Figure 6: PbGJ matrix inversion in 8x8 sizes of matrix.

This algorithm running time is $O(N^3)$ on a single processor system, but if we add more than one processor, the running time is decreased. Following equation shows details.

$$n \times (n\omega + \frac{n^2}{p}) \tag{3}$$

In above equation, p means the number of processors, and ω is data communication time between processors. It eventually shows as following equation.

$$n^2\omega + \frac{n^3}{p} \tag{4}$$

If we add more processors, we can obtain more performance in large size of matrix inversion. Therefore, we can reduce execution time as much as adding processors.

4. EXPERIMENT RESULTS

In this section, we will show the experiment results. We experimented solving 30000x30000 size of matrix inversion using PbGJ. The experiment has built by thirty-six workstations with Ethernet connection. The each hardware consists of Intel Q8400 2.66Ghz CPU, 3GB of memory, 500GB hard disk. All these hardware are connected with 1Gbps Ethernet. We modified a scheduler the ParTS(Parallel Task Scheduler) to manage tasks and processors. Our application was corded by C++ in Windows XP based. It was possible to solve more than 30000x30000 size of matrix, but because of the long execution time, our test was limited. For that limitation, we decide the test assigning different numbers of processors at each condition. Table 1 is show the experiment result. We assign single processor, 4 processors and 8 processors at each condition. The condition is consists of 10000x10000, 20000x20000, and 30000x30000 size of matrix inversion.

TABLE I
RESULT OF THE PbGJ CALCULATION

Matrix inversion result with single processor

	Execution time(sec.)	Time for a step(sec.)
10000X10000	11375	1.137
20000X20000	N/A	N/A
30000X30000	N/A	N/A

Matrix inversion result assigned 4 processors

	Execution time(sec.)	Time for a step(sec.)
10000X10000	9955	0.995
20000X20000	49268	2.463
30000X30000	146032	4.868

Matrix inversion result assigned 8 processors

	Execution time(sec.)	Time for a step(sec.)
10000X10000	8614	0.861
20000X20000	38446	1.922
30000X30000	108721	3.624

Because of the memory limit, over 10000x10000x size of the matrix inversion at single processor couldn't be experimented. In our algorithm, we distribute the matrix data as many as processors numbers, and we already apply the data access locality problem. Compare with single and assigned 4 or 8 processors, execution time is not show direct proportion, because of the communication between the processors. This communication time is added at execution time. In other words, execution time is consists of matrix operation time and the communication time. In same size of the matrix, the length of the message is equal at every step. In the test, this communication size is not related with processors number. If the matrix operation time increases at each processor, the ratio of the communication time is decreased relatively. Implementing effective message passing technology is an idea to improve the performance.

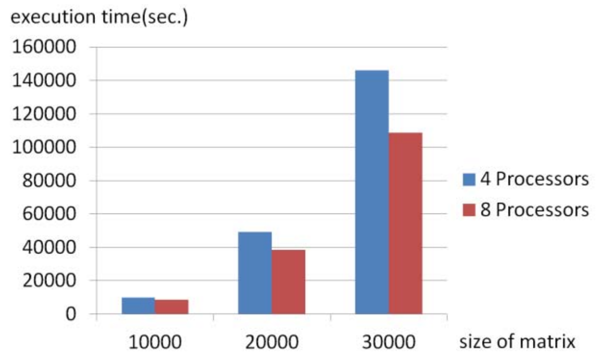

Figure 7: Performance of PbGJ matrix inversion on 4 processors system and 8 processors system.

Figure 7 show the execution time comparison. Test unit assigned 8 processors shows the more performance than 4 processors. Testing our algorithm the size 30000x30000, it takes about 40 hours in assigned 4 processors, but it takes about 30 hours in assigned 8 processors. We found out assigning more processors can obtain high performance to solve massive matrix inversion in cloud computing system. In Shang's paper [5], they experiment increase of the number of blocks affects to the execution time. In that case, the execution time was increased when they divide more pieces of blocks. In our algorithm, it shows more performance when we add more processors in large size of matrix.

5. CONCLUSION

Solving massive matrix inversion is a matter of common interests in science research area. However, it is not easy to construct the system which calculating large size of matrix inversion to researchers. In this paper, we present a matrix inversion algorithm in cloud computing systems, and test the performance. In future work, it may possible to use as a cloud server which many science researchers can borrow to use in their researches. It is similar with HaaS & SaaS. We were focused on reducing data dependency between the processors, simplified model, and analyzing data access localities. We implement these efforts to the cloud computing system, and we found out more processor makes more performance. Network bandwidth and delay were the factor affects the performance. In the test, communication time inclusive of waiting time for communicate with processors is more occupied than matrix calculation time in each processor. Research to reduce communication time would be one of idea to improve the performance. Also we schedule tasks and manage processors using parallel task scheduler named ParTS. If improve the scheduling performance, it will be a method improving the performance as well. Improving data communication performances and scheduling matters would be involved to the future research.

6. REFERENCES

[1] Hoffa, C., Mehta, G., Freeman, T., Deelman, E., Keahey, K., Berriman, B., and Good, J., “On the Use of Cloud Computing for Scientific Workflows”, Fourth IEEE International Conference on eScience, 2008.

[2] N. MELAB, E.-G. TALBI, and S. PETITON, “A Parallel Adaptive Gauss-Jordan Algorithm”, The Journal of Supercomputing, vol. 17, pp. 167–185, 2000.

[3] Foster, I., Zhao, Y., Raicu, I., and Lu, S., “Cloud Computing and Grid Computing 360-Degree Compared”, Grid Computing Environments Workshop, 2008.

[4] Hayes, B., “Cloud Computing: As software migrates from local PCs to distant Internet servers, users and developers alike go along for the ride”, Communications of the acm, vol.51, no.7, 2008.

[5] Shang, L., Petiton, S., and Hugues, M., “A New Parallel Paradigm for Block-based Gauss-Jordan Algorithm”, Eighth International Conference on Grid and Cooperative Computing, 2009.

[6] Yokoyama, S., Matsumoto, K., and Sedukhin, S., “Matrix Inversion on the Cell/B.E. Processor”, 11th IEEE International Conference on High Performance Computing and Communications, 2009.

[7] Vaquero L., Rodero-Merino, L., Caceres, J., Lindner, M., “A Break in the Clouds: Towards a Cloud Definition”, ACM SIGCOMM Computer Communication Review, vol.39, no.1, 2009.

[8] Rajalakshmi, K. “Parallel Algorithm for Solving Large System of Simultaneous Linear Equations”, IJCSNS International Journal of Computer Science and Network Security, vol.9 no.7, 2009.

[9] Serge, P., and Aouad, L., “Large Scale Peer to Peer Performance Evaluations, with Gauss-Jordan Method as an Example”, LNCS 3019, pp. 938–945, 2004.

[10] Shang, L., Hugues, M., and Serge, P., “A Fine-grained Task Based Parallel Programming Paradigm of Gauss-Jordan Algorithm”, Journal of Computers, vol. 5, no.10, 2010.

[11] Juanjuan, Bai Ling Gao and LidongHe, “Constructing Windows+gcc+mpi+omp and Performance Testing with Gauss-Jordan Elimination Method in Finding the Inverse of a Matrix”, 2010 International Conference On Computer Design And Applications, vol.2 , 2010.

[12] M.E.J. Newman, “A measure of betweenness centrality based on random walks”, Social Networks, vol.27, no.1, pp. 39-54, 2005.

[13] Kwak, H., Lee, C., Park, H., and Moon, S., “What is Twitter, a Social Network or a News Media?”, Proceedings of the 19th international conference on World wide web, 2010.

[14] Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M., “A View of Cloud Computing”, communications of the acm, vol.53, no.4, 2010.

[15] Geelan, J., “Twenty one experts define cloud computing. Virtualization”, Electronic Magazine, 2008, article available at http://virtualization.sys-con.com/node/612375.

[16] Kim, H., Lu, S., Kim, J., and Kim, B., "Parallel, Multistage Model for Enterprise System Planning and Design", IEEE Systems Journal, vol. 4, no. 1, pp. 6-14, 2010.