The Parallelization of the direct and Iterative Schemes for Solving Boundary Layer Problem on Heterogeneous Cluster Systems

Norma Alias[1] , Norhafiza Hamzah[2], Norsarahaida S. Amin[2], Noriza Satam[2], Zarith Safiza Abd. Ghaffar[2] and Roziha Darwis[2]

[1] Ibnu Sina Institute for Fundamental Science Studies, Universiti Teknologi Malaysia, 81310 Skudai, Johor Bahru, Malaysia.
norma@ibnusina.utm.my

Department of Mathematics, Faculty of Science, Universiti Teknologi Malaysia, 81310 Skudai, Johor Bahru, Malaysia.
nsarah@mel.fs.utm.my,
{norhafizahamzah, norizasatam, roziha.darwis, zarithsafiza.ag}@gmail.com

Abstract. In this paper, we present iterative schemes, specifically the iterative schemes: conjugate gradient, and Gauss-Seidel as well as direct schemes: LU factorization and Gauss elimination for solving boundary layer problem. The aim of this paper is to offer reasonable assessments and contrasts on behalf of the numerical experiments of these two schemes. The sequential and parallel programming is developed using a C programming language under Linux environment, while the parallel programming is running using the Parallel Virtual Machine (PVM) on a heterogeneous cluster systems. The analysis of the results are conducted in terms of numerical and parallel performance evaluations namely execution time, speedup, efficiency, effectiveness and temporal performance. The results prove that the iterative methods of conjugate gradient and Gauss-Seidel method are the alternatives scheme for solving the large scale computation.

Keywords: Parallel, Keller-box, high performance computing, performance analysis.

1 Introduction

The numerical solution methods for linear systems of equations, $Ax = b$, are broadly classified into two categories, direct methods, and iterative methods [1]. The most reliable and simplest solvers are based on direct methods, but the robustness of direct solvers comes at the expense of large storage requirements and execution times, while the iterative techniques exhibit problem-specific performance and lack the

generality, predictability and reliability of direct solvers. Yet, these disadvantages are outweighed by their low computer memory requirements and their substantial speed especially in the solution of very large matrices [2]. However, direct methods have been recommended for solving large sparse linear systems when, among other reasons, the system is ill-conditioned [3].

Direct methods obtain the exact solution in finitely many operations and are often preferred to iterative methods in real applications because of their robustness and predictable behavior. However, as the size of the systems to be solved increases, they often become almost impractical due to the phenomenon known as fill-in [1]. The fill-in of a sparse matrix is a result of those entries which change from an initial value of zero to a nonzero value during the factorization phase.

Although iterative methods for solving linear systems find their origins in the early nineteenth century especially by Gauss, the field has seen an explosion activity stimulated by demand due to extraordinary technological advances in engineering and sciences [4]. According to [5], the term 'iterative methods' refers to a wide range of techniques that use successive approximations to obtain more accurate solutions to a linear system at each step. Beginning with a given approximate solution, these methods modify the components of the approximation, until convergence is achieved. They do not guarantee a solution for all systems of equations. Within the context of the previous studies of direct methods, there is no advantages. Direct method can be used as a preconditioner of iterative methods for symmetric definite or indefinite problems that provided the tolerance parameter is somewhat relaxed [17]. However, when they do yield a solution, they are usually less expensive than direct methods [1].

2 Problem Statements

In aerodynamics, the details of the flow within the boundary layer are important for many problems including the skin friction drag on an object, the heat transfer in high speed flight, and wing stall which is the condition when aircraft wings will suddenly lose lift at high angles to the flow. The simplified Navier-Stokes equations, known as Prandtl's boundary layer equations are

$$\frac{\partial u}{\partial x}+\frac{\partial \upsilon}{\partial y}=0, \tag{4}$$

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+\upsilon\frac{\partial u}{\partial y}=-\frac{1}{\ell}\nu\frac{\partial^2 u}{\partial y^2}, \tag{5}$$

with the boundary conditions

$$\text{x} = 0,\ y=0: u=\upsilon=0; y=\infty: u=U(x,t), \tag{6}$$

where the potential flow $U(x,t)$ is to be considered known [6]. A suitable boundary layer flow must be prescribed over the whole *x, y* region under

consideration for the instant t=0. In the case of steady flow, the system of equations is written as

$$\frac{\partial u}{\partial x}+\frac{\partial \upsilon}{\partial y}=0 \tag{7}$$

$$u\frac{\partial u}{\partial x}+\upsilon\frac{\partial u}{\partial y}=-\frac{1}{\ell}\frac{dp}{dx}\nu\frac{\partial^2 u}{\partial y^2}, \tag{8}$$

Figure 1 shows a boundary layer along a flat plate at zero incidences. Let the leading edge of the plate be at $x=0$, the plate being parallel to the x-axis and infinitely long downstream, Figure 1. We shall consider steady flow with a free-stream velocity, U_∞ which is parallel to the x-axis. The velocity of potential flow is constant in this case, and therefore, $\frac{dp}{dx}\equiv 0$ [6]. The boundary layer equations 7 to 8 become

$$\frac{\partial u}{\partial x}+\frac{\partial \upsilon}{\partial y}=0 \tag{9}$$

$$u\frac{\partial u}{\partial x}+\upsilon\frac{\partial u}{\partial y}=\nu\frac{\partial^2 u}{\partial y^2}, \tag{10}$$

As velocity changes in the stream wise direction, velocity in the other directions will change as well. There is a small component of velocity at right angles to the surface which displacement the flow above it. The thickness of the boundary layer can be defined as the amount of this displacement.

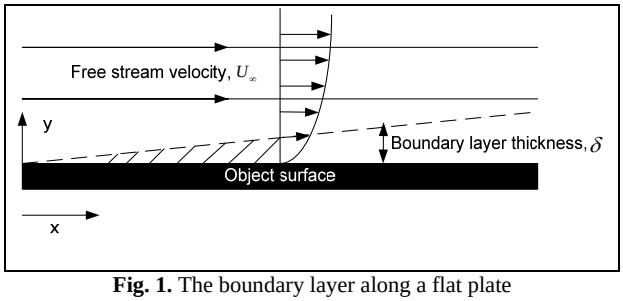

Fig. 1. The boundary layer along a flat plate

3 Numerical Direct and Iterative Methods

Some numerical methods used in this research come from two categories which are direct and iterative methods. Solution using direct method involved LU Factorization and Gaussian Elimination. On the other hand, Conjugate Gradient and Gauss Seidel method represents the solution using iterative scheme.

3.1 Direct Schemes

LU factorization and Gauss Elimination are two numerical simulations of the direct methods under consideration for solving systems of linear equations.

3.1.1 LU Factorization

A block tridiagonal matrix is obtained after we applied the finite difference scheme and Newton's method on the boundary layer equation (11), which is having square matrices blocks in the lower, main, and upper diagonal, where all other blocks is a zero matrices. It is basically a tridiagonal matrix but has submatrices in places of scalars. A block tridiagonal matrix in this case study has the form as follow:

$$[A][\delta] = [r] \tag{11}$$

To solve equation (11), we use LU factorization for decomposing *A* into a product of a lower triangular matrix, *L* and an upper triangular matrix, *U* as follows,

$$[A] = [L][U] \tag{12}$$

The step in which Γ_j, α_j, and W_j are calculated is usually referred to as the forward sweep. Once the elements of *W* are found, equation (11) then gives the solution δ in the so-called backward sweep. Once the elements of δ are found, Newton's method can be used to find the $(i+1)^{th}$ iteration. These calculations are repeated until some convergence criterion is satisfied and calculations are stopped when $\left|\delta v_0^{(i)}\right| < \varepsilon_1$ where ε_1 is a small prescribed value. In this paper, the value of ε_1 is 0.00005.

$$[\alpha_1][W_1] = [r_1], \tag{13}$$

3.1.2 Gaussian Elimination

Objective of Gaussian elimination is to convert the general system of linear equations into a triangular system of equations [1]. The process of Gauss elimination has two parts. The first part is forward elimination reduces a given system to a triangular system. This is accomplished through the use of elementary row operations, which applies the characteristic of linear equations that any row can be replaced by a row that added to another row and multiplied by a constant. The second step uses back substitution to find the solution of the triangular system.

3.2 Iterative Schemes

Iterative schemes, on the other hand, do not modify matrix A. Rather, they involve the matrix only in the context of matrix-vector product operations. The term "iterative methods" refers to a wide range of techniques that use successive approximations to obtain more accurate solutions to a linear system at each step [5]. Beginning with a given approximate solution, these methods modify the components of the approximation, until convergence is achieved. They do not guarantee a solution for all systems of equations. However, when they do yield a solution, they are usually less expensive than direct methods.

The conjugate gradient method (CG) is an algorithm for the numerical solution of particular systems of linear equations. As it is an iterative method, so it can be applied to sparse systems that are too large to be handled by direct methods.

Another popular iterative scheme to solve a system of linear equation is Gauss Seidel. In this work, Gauss Seidel method is used to solve the problem sequentially. For parallel purpose, we applied other generation of Gauss Seidel namely red-black Gauss Seidel which is more efficient when implements into a parallel machine [12].

4 Formulation of Parallel Algorithm

The formulation of parallel direct and iterative methods is based on the domain decomposition technique. where domain A is decompose into subdomains and being distributed to all processors. For the direct method, the subdomians are overlapping, so we used a pipeline configuration, but for iterative methods, the data dependencies are low, so the subdomain can be easily distributed to the processors [13].

4.1 Gauss Elimination

The backward data distribution of Gauss elimination to the parallel processors was designed: the data is divided by rows block based on the upper triangular matrix (U). The P_1 to P_p are the processes involved in the parallel implementation. As it is an

upper matrix, the data on process P_p needs to be calculated first by P_1, and then the results will be passes to the next processor, P_2. This process will continue up to P_n for solving the large scale of the linear system.

4.2 LU Factorization

Parallel LU factorization is using the same technique as parallel Gauss elimination, but the computation is including the U and lower triangular (L) matrices respectively. The computational complexity is extremely expensive for solving the backward and forward substitution of two linear systems. The calculation is started with L matrix then continues by U matrix [14].

4.3Gauss Seidel Red-Black (GSRB)

Gauss seidel red-black decompose domain Ω to two subdomain on red grid *R*, Ω^R and subdomain on black grid, *B*,$_{\omega}$. Ω^R is an approximate solution on odd grid and Ω^B is an approximation solution on even grid. The computation is first done on Ω^R and followed by computation on Ω^B .The decomposition of domain Ω to these two subdomain makes the computation on grid i^{th} is independent and easy to be implemented on parallel computer system.

4.4 Conjugate Gradient

The implementation of parallel CG can be developing directly without much modification on sequential CG. The non-overlapping subdomains of CG make it easy to distribute the data equally among all processors [15]. The CG method used for solving symmetric positive definite linear systems [16].

5 Numerical Results

Table 1 and 2 show numerical analysis on direct and iterative methods respectively. The analysis are in terms of execution time, mean square error (MSE), root mean square error (RMSE), number of iteration for iterative methods, and maximum error (Max. Err) for m =20,000size of matrix.

From Table 1, we can see that execution time of Gauss elimination is less than LU factorization. This may caused by the computational complexity of LU is greater than Gauss elimination, as LU needs to calculate L and U matrices while Gauss elimination only involved U matrix, and so the waiting time is lower than LU method. There's only a slightly different of MSE, RMSE and maximum error for both

methods. This proved that the direct method is really accurate and the result is nearly to the exact solution.

Table 1. Numerical analysis of direct methods

Method	G. Elimination	LU
Execution time (μ second)	577067880	691048023
MSE	3.54E-09	1.94E-09
RMSE	1.25E-17	3.76E-17
Max. Err	4.60E-7	5.20E-7

The numerical analysis of iterative methods is shown in Table 2. From the table, we can see that CG is much better than GSRB. The number of iteration also shows that CG is performing better than GSRB in obtaining the approximation result. So we can say that CG is the best choice among other iterative methods to solve the problems with large size of matrix as it is also easy to implement to the parallel computers.

Table 2. Numerical analysis of iterative methods

Method	G. Elimination	LU
Execution time (μ second)	577067880	691048023
Iteration	120	56
MSE	5.43E-6	9.80E-8
RMSE	2.33E-3	3.13E-4
Max. Err	5.39E-3	5.23E-3

6 Parallel Performance Evaluation

The analysis of the parallel performance evaluations are conducted in terms of execution time, speedup, efficiency, effectiveness and temporal performance. The distributed memory of the heterogeneous cluster systems are supported by Pentium IV, dual core and quad core CPUs for implementing the parallelism of a huge simulation and computational task.

6.1 Software and Hardware

The sequential and parallel programming is developed based on a C programming language under Linux environment, while the parallel programming is running using the Parallel Virtual Machine (PVM) on a heterogeneous cluster systems.

6.2 Execution Time

The execution time is basically the CPU running time during the calculation of the program in micro second. The bigger size of matrices lead to higher calculation complexity that implies the longer time it takes to execute the process. Figure 3 shows the execution time for all methods discussed above. From the graph shows the iterative methods is better than direct methods in terms of execution time.

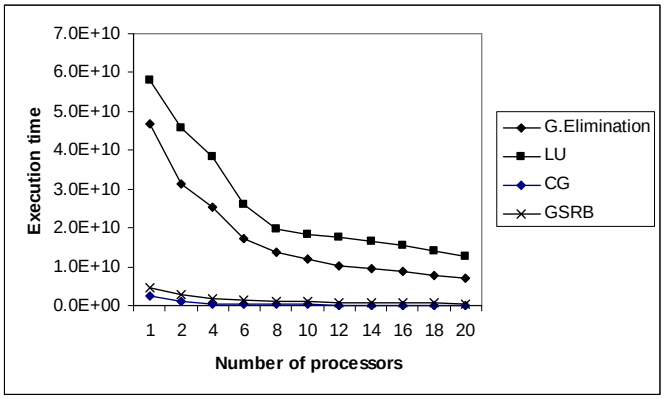

Fig. 3. The execution time vs number of processors

6.3 Speedup

Figure 4 shows the speedup and efficiency for parallel direct methods. Based on the graphs, the speedup is increased when the number of processor is increase. The gradient of speedup is linear on p<12, since of optimum load balancing and data distribution on all processors. For the iterative methods, conjugate gradient shows the great improvement on speedup compared to GSRB method. This proved that conjugate gradient is very suitable to be implemented on parallel computers and to solve large problems.

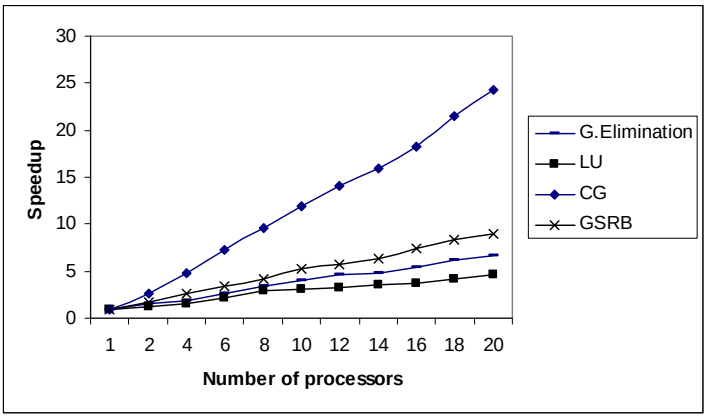

Fig. 4. The speedup vs number of processors

6.4 Efficiency

The efficiency of a parallel program is a measure of processor utilization. The efficiency graph (figure 5) is incline when p>10 for iterative methods, where the processors need additional communication time to send and receive data, while idle time increase as the imbalance of the workload and also caused by the pipeline implementation on parallel algorithms of direct methods.

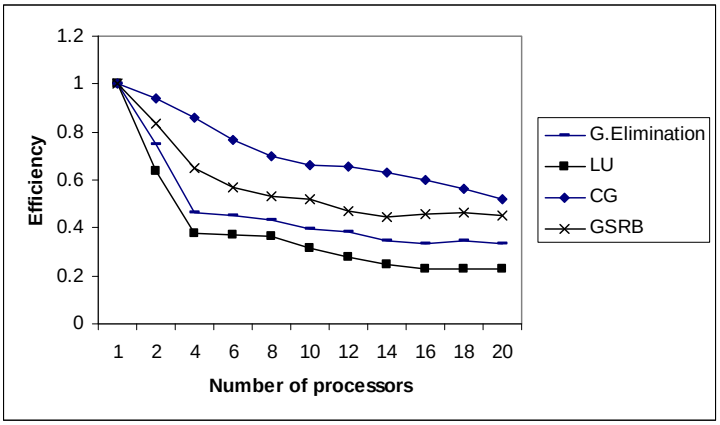

Fig. 5. The efficiency vs number of processors

6.5 Effectiveness

Figure 6 shows that effectiveness increase when the number of processors increases. The formula of the effectiveness depends on the speedup, when the speedup increases, the effectiveness will also increase. The graph shows effectiveness of parallel direct methods is dominant by Gauss elimination. For iterative methods, the effectiveness of CG is much better than GSRB. This proved that CG has a very good performance in solving a large sparse problem.

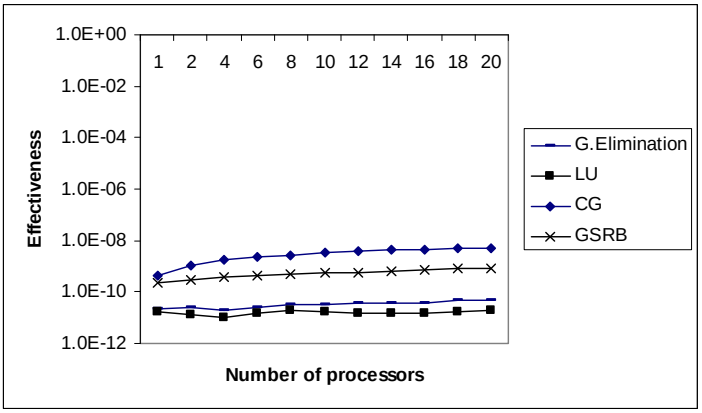

Fig. 6. The effectiveness vs number of processors

6.6 Temporal Performance

Temporal performance is a parameter to measure the performance of a parallel algorithm. The results in Figure 6 shows that the temporal performance of Gauss elimination is better than LU factorization and the iterative schemes are higher than the direct schemes.

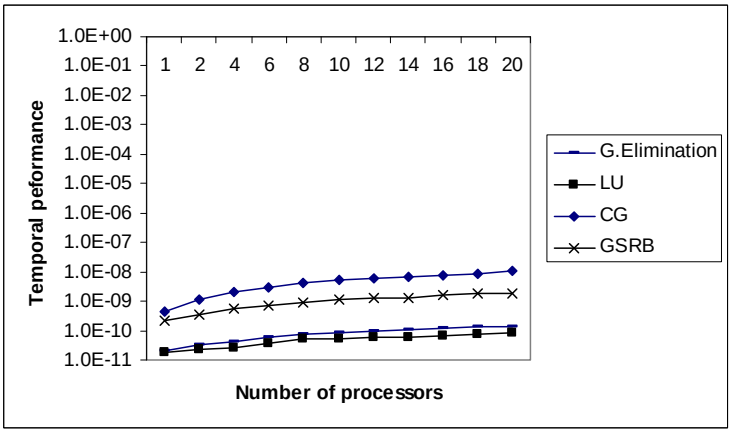

Fig. 7. The temporal performance vs number of processors

7 Conclusion

In this work, we have presented the experimental results illustrating the parallel implementation of iterative and direct method using PVM programming environment on heterogeneous architecture. The contributions of this paper: in terms of the parallel performance evaluations, the parallelization of iterative CG method is the alternative scheme and in term of numerical analysis, the parallelization of direct Gauss elimination method is the alternative scheme for solving the large-sparse matrices of the boundary layer problem. The combinations of the parallel direct and iterative methods for improving the performances results are the suggested directions for future research.

Acknowledgments

This work was supported in part by Research Management Center, UTM and Ministry of Science, Technology and Innovation of Malaysia (MOSTI) through National Science Fellowship scholarship (NSF).

References

1. M. Rashid and C. Jon, "Parallel iterative solution method for large sparse linear equation systems,"University of Cambridge Comp. Lab, United Kingdom, no. 650, 2005.
2. J. M. George, "a new set of direct and iterative solvers for the tough2 family of codes," presented at the TOUGH workshop 95, Berkeley, 1995, p 293-298.
3. W. E. Louis, "Iterative vs. a directive method for solving fourth order elliptic difference equations," in *proc. ACM national meeting*, 1966, p 29-35.
4. Y. Saad and H.A. van der Vorst, " Iterative solution of linear systems in the 20-th century," *J. Comp. Appl. Math.*, 2000, vol. 123, pp. 1–33.
5. R. Barrett, M. Berry, T.F. Chan, J. Demmel, J.M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. Philadelphia: Society for Industrial and Applied Mathematics, 1994.
6. H. Schlichting, *Boundary layer theory*. New York: McGraw-Hill, 1979, p 127-148.
7. H. B. Keller, *A new difference scheme for parabolic problems. In: Hubbard, B. ed. Numerical solutions of partial differential equations*. New York: Academic Press, 1971, 2: 327-350.
8. H. B. Keller, and T. Cebeci, "A numerical methods for boundary layer flows, I: Two-dimensional laminar flows," in *Proc. 2nd Int. Conf. on Numerical Methods in Fluid Dynamics*, New York: Springer-verlag, 1971.
9. H. B. Keller, and T. Cebeci, "Accurate numerical methods for boundary layer flows, II: Two-dimensional turbulent flows," *AIAA Journal*, 1972. Vol. 10, 1972, pp. 1193-1199.
10. T. Cebeci, A. Smith, *Analysis of turbulent boundary layers*. New York: Academic Press, 1974.
11. B. Wilkinson, M. Allen, *Parallel programming: Techniques and applications using networked workstations and parallel computers*. New Jersey: Prentice hall, 1998.

12. A.G. Sifalakis, S.R. Fulton, E.P. Papadopouloua, Y.G. Saridakis. Direct and iterative solution of the generalized Dirichlet_Neumann map for elliptic PDEs on square domains. Journal of Computational and Applied Mathematics 227 (2009).171-184.
13. C. Bernardi, T. Chacón Rebollo, E. Chacón Vera, D. Franco Coronil. A posteriori error analysis for two non-overlapping domain decomposition techniques. Applied Numerical Mathematics 59 (2009) 1214–1236.
14. Chi-Ye Wu, Ting-Zhu Huang. Stability of block LU factorization for block tridiagonal matrices. Computers and Mathematics with Applications 57 (2009) 339-347.
15. Gonglin Yuana, Xiwen Lu, Zengxin Weia. Conjugate gradient method with descent direction for unconstrained optimization. Journal of Computational and Applied Mathematics 233 (2009) 519-530.
16. Tong-Xiang Gua, Xian-Yu Zuo, Xing-Ping Liu, Pei-Lu Li. An improved parallel hybrid bi-conjugate gradient method suitable for distributed parallel computing. Journal of Computational and Applied Mathematics 226 (2009) 55-65.
17. Yan-Fei Jing, Ting-Zhu Huang. On a new iterative method for solving linear systems and comparison results. Journal of Computational and Applied Mathematics 220 (2008) 74 – 84.