JAST, Vol. 4, No. 1, pp 17–23

JAST
Journal of Aerospace Science and Technology

A Message-Passing Distributed Memory Parallel Algorithm for a Dual-Code Thin Layer, Parabolized Navier-Stokes Solver

M. Sarvalisha[1], O. Abouali[2], M. M. Alishahi[3]

In this study, the results of parallelization of a 3-D dual code (Thin Layer, Parabolized Navier-Stokes solver) for solving supersonic turbulent flow around body and wing-body combinations are presented. As a serial code, TLNS solver is very time consuming, and takes a large part of memory due to the iterative and lengthy computations. Also for complicated geometries, an exceeding number of grid points are required, which results in larger serial computation times. Therefore, parallelizing this code would bring about large saving in computer time and memory. In this study, a cluster of 16 computational nodes with 2.4 and 2.8 GHz, P4 CPU has been used. Also MPI library is used for communicating data among processors. Domain is partitioned in a 1-D form in longitudinal, radial and circumferential directions, and the results are compared with those of serial computations. There are several methods for data communication among processors, such as blocking send and non-blocking send. The performance of each method is evaluated and the best method for the problem at hand is determined. The results are compared in terms of run time, speed-up and efficiency for executing the parallel code on 1, 2, 3, 4, 8, 12 and 16 processors. Also the parallel results are compared with serial results and the correctness of the parallel code is proved for each case. The effect of different partitioning direction and their interaction with the turbulence modeling is studied and the best choice is shown. The limitations of using Baldwin-Lomax turbulence model in a parallel program are discussed and a solution is presented.

INTRODUCTION

Solution of large scale CFD problems using Reynolds Averaged Navier-Stokes (RANS) equations is an exhaustive task. Thus, researchers have used simpler forms of Navier-Stokes equations to reduce the computation. Abouali *et.al.* [1, 2, 3] have presented a dual code strategy, a combination of Thin Layer Navier-Stokes equations (TLNS) and Parabolized Navier-Stokes equations (PNS) as a simpler substitute for RANS equations. Using this dual code, supersonic turbulent flow around a missile can be efficiently resolved. The flow field around the nose is solved by TLNS code and the rest of the body up to wings juncture is handled by PNS code. The modeling is switched from PNS to TLNS in the vicinity of wings. This dual-code strategy reduced the required memory and computer time for solving 3-D supersonic flows over wing-body configurations. However, this strategy is not quite useful for solving 3-D problems with complicated geometries, which require a large number of grid points. To remedy this problem, one may employ fast computers with large memories. However, such hardware is very expensive. The other choice is to use parallel processors together with a compiler that accommodates data communication among processors. MPI (Message Passing Interface) is one of the best libraries around, which facilitates data communication among processors [4]. For this task, a cluster of 54 computational nodes with 2.4 and 2.8 GHz, P4 CPU

1. *MSc. Student, Dept. of Mech. Eng., Shiraz Univ., Shiraz, Iran.*
2. *Assistant Professor, Dept. of Mech. Eng., Shiraz Univ., Shiraz, Iran.*
3. *Professor, Dept. of Mech. Eng., Shiraz Univ., Shiraz, Iran.*

at HPCC, Shiraz University was at hand. But only 16 computational nodes of this cluster are used in this work. When several nodes execute a parallel code, the first node has to communicate the data more than the others. Thus, to decrease the run time, it was decided to use a faster node with 2.8 GHz, P4 CPU as the first node and the other slower nodes (2.4 GHz, P4 CPU) as ordinary computational nodes. This study is concentrated on parallelizing TLNS (Thin layer Navier-Stokes) solver rather than PNS (Parabolized Navier-Stokes), since the required memory and computer time of the PNS solver is much less than the TLNS solver.

Many studies include parallel programming of flow problems. Rviz-Calavera and Hirose [5] have presented a parallel time-accurate Euler code to calculate unsteady transonic flow around wings. This code has been implemented on a distributed memory parallel machine with about 140 computational nodes. Two different grids namely $80 \times 16 \times 30$ and $160 \times 32 \times 30$, have been considered. This problem was solved on 1 up to 32 machines and speed-up was about 6 when 32 machines were used on a larger grid.

Drikakis [6] has presented a parallel upwind method for 3-D laminar and turbulent incompressible flows. Third order interpolation was used for the computation of primitive variables at the cell faces. The algorithm was parallelized using shared memory and message-passing models. The grid included 224600 points and 1 to 8 nodes were used. The efficiency of the shared memory system was about 75% (speedup=6), but this value for the distributed memory system was about 37% (speedup=2.96) when 8 machines were used.

Bonisch and Ruhle [7] have presented the application of a parallel 3-D simulation code for Euler and Navier-Stokes supersonic flows around re-entering space vehicles on distributed memory parallel computers. The parallelization approach used a domain decomposition method for the subdivision of the numerical grid and MPI as the message passing environment to obtain a portable application. The new parallel application was developed from an existing sequential code with just a few extensions within the source code, which did not alter the numerical efficiency of the sequential algorithm. Also the measured maximum speed-up was high, in other words, with 110592 grid points, executing on 1 up to 512 nodes was approximately linear and equaled to about 400 for 512 machines.

As it is clear from the mentioned references, the speedup and efficiency of the parallelized codes vary to a large extent and is case dependent. The numerical details of the algorithm usually have a strong effect on the success of the parallel code. In this study, the proper parallelization model for enhancement of efficiency and speed-up of a TLNS code are investigated. It is shown that improper parallelization together with use of Baldwin-Lomax turbulence model causes some unnecessary drawbacks that can be prevented.

GOVERNING EQUATIONS

The conservative form of Navier-stokes equations in generalized coordinates are:

$$\tilde{Q}_t + (\tilde{E} - \tilde{E}_v)_\xi + (\tilde{F} - \tilde{F}_v)_\eta + (\tilde{G} - \tilde{G}_v)_\zeta = 0, \qquad (1)$$

$$\begin{aligned}
\tilde{Q} &= Q/J, \\
\tilde{E} &= \frac{\xi_x}{J}E + \frac{\xi_y}{J}F + \frac{\xi_z}{J}G, \\
\tilde{E}_v &= \frac{\xi_x}{J}E_v + \frac{\xi_y}{J}F_v + \frac{\xi_z}{J}G_v, \\
\tilde{F} &= \frac{\eta_x}{J}E + \frac{\eta_y}{J}F + \frac{\eta_z}{J}G, \\
\tilde{F}_v &= \frac{\eta_x}{J}E_v + \frac{\eta_y}{J}F_v + \frac{\eta_z}{J}G_v, \\
\tilde{G} &= \frac{\zeta_x}{J}E + \frac{\zeta_y}{J}F + \frac{\zeta_z}{J}G, \\
\tilde{G}_v &= \frac{\zeta_x}{J}E_v + \frac{\zeta_y}{J}F_v + \frac{\zeta_z}{J}G_v, \qquad (2)
\end{aligned}$$

where Q is the conservative variable vector and E, F, G, E_v, F_v and G_v are inviscid and viscous fluxes, respectively. Variables ξ and ζ are in streamwise and circumferential direction, η is normal to the wall direction, and J is Jacobian of the transformation. Thin layer approximation can now be applied to the transformed Navier-Stokes equations. According to this approximation, all viscous terms containing partial derivatives with respect to ξ can be neglected. The resulting thin layer equations may be written as:

$$\tilde{Q}_t + \tilde{E}_\xi + (\tilde{F} - \tilde{F}_v)_\eta + \left(\tilde{G} - \tilde{G}_v\right)_\zeta = 0. \qquad (3)$$

It should be emphasized that for geometries such as wing-body combinations, $\tilde{G}_v$ in circumferential direction should be retained. Roe upwind method is used for calculation of inviscid fluxes and central differencing is used for discretization of viscous fluxes. Equation (3) may be expressed in the semi-discrete conservation law form given as, [3],

$$\begin{aligned}
&\left(\hat{Q}_{j,k,l}\right) + \left(\hat{E}_{j+1/2,k,l} - \hat{E}_{j-1/2,k,l}\right) + \\
&\left((\hat{F} - \hat{F}_v)_{j,k+1/2,l} - (\hat{F} - \hat{F}_v)_{j,k-1/2,l}\right) + \\
&\left(\left(\hat{G} - \hat{G}\right)_{j,k,l+1/2} - \left(\hat{G} - \hat{G}_v\right)_{j,k,l-1/2}\right) = 0, \qquad (4)
\end{aligned}$$

where $\hat{E}$, $\hat{F}$, $\hat{G}$ are numerical fluxes on the bounding sides of a cell.

Baldwin-Lomax turbulence modeling with some modifications is used in this code. Since implementation of this turbulence model affects the efficiency of parallelization to some extent, the model is explained in the following in detail. In this model μ_t (the eddy viscosity coefficient) is defined as, [3],

$$\mu_t = \min[(\mu_t)inner, (\mu_t)outer]. \tag{5}$$

For the inner layer, the Prandtl-Van Driest formula is used to determine μ_t which is defined as,

$$(\mu_t)inner = \rho l^2 |\Omega|, \tag{6}$$

where l, the mixing length, given as,

$$l = ky[1 - \exp(-y^+/A^+)], \tag{7}$$

where, k and A^+ are constants and equal to 0.41 and 26 respectively, $|\Omega|$ is the magnitude of the local vorticity vector and is defined as,

$$|\Omega| = \sqrt{\left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial z} - \frac{\partial w}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial x} - \frac{\partial u}{\partial z}\right)^2}. \tag{8}$$

y^+ is defined as,

$$y^+ = \frac{\sqrt{\rho_w \tau_w}}{\mu_w} y, \tag{9}$$

where ρ_w, τ_w and μ_w are the density, shear stress and viscosity coefficient at the wall, respectively, and y is the normal distance from the wall. In the outer region, for attached boundary layers, the turbulent eddy viscosity (μ_t) is defined as:

$$(\mu_t)_{outer} = K\, C_{cp}\, \rho\, F_{wake}\, F_{Kleb}(y), \tag{10}$$

where K and C_{cp} are constants equal to 0.0168, and 1.6 respectively, and $F_{Kleb}(y)$ is the Klebanoff intermittency factor,

$$F_{Kleb} = \left[1 + 5.5\left(\frac{C_{Kleb}\, y}{y_{\max}}\right)^6\right]^{-1}. \tag{11}$$

Also;

$$F_{wake} = \min\begin{cases} y_{\max} F_{\max} \\ C_{wk}\, y_{\max}\, u_{Diff}^2 / F_{\max} \end{cases} \tag{12}$$

where C_{kleb} is the Klebanoff constant equal to 0.3, C_{wk} is the wake constant equal to 0.25 and,

$$u_{Diff} = \left(\sqrt{u^2 + v^2 + w^2}\right)_{\max} - \left(\sqrt{u^2 + v^2 + w^2}\right)_{\min}. \tag{13}$$

$F_{\max}$ and $y_{\max}$ are computed from the function:

$$F(y) = y\,|\Omega|\,[1 - \exp(-y^+/A^+)], \tag{14}$$

such that the peak value of $F(y)$ between the wall and the outer flow field is defined as $F_{\max}$ and the value of y at this station is defined as $y_{\max}$.

This Baldwin-Lomax model causes certain problems when applied in flows around slender bodies at incidence. In the lee-ward separated flow region, it becomes difficult to determine the correct value of $F_{\max}$, which is necessary for evaluating $\mu_{t,outer}$. In an attached flow, there is only one maximum for $F(y)$ in the radial direction and $F_{\max}$ can be simply found. When separated flow occurs, two maxima for $F(y)$ are encountered. The first peak occurs in the boundary layer and a second layer peak exists due to the presence of vortex sheet. If the Baldwin-Lomax model is used to obtain $F_{\max}$, the second maximum in $F(y)$ is obtained. This results in large values of $\mu_{t,outer}$, causing a distortion or washout of the features in the computed flow. Some modifications of Baldwin-Lomax are usually applied to remedy this problem. For each axial station, a maximum value for the scaling length $(y_{max})_w$ is defined as 1.8 times the value of y_{max} on the windward ray. If the two peaks in $F(y)$ merge into one abnormally large peak (or a peak cannot be found at all), the value of $F_{\max}$ is taken from the previous roll angle and y_{max} is set equal to $(y_{max})_w$ [1-3]. As will be shown shortly, implementation of this condition changes with the direction of partitioning, and hence the efficiency of the overall parallelization varies.

To study the performance of parallelized code, figure of merits such as speed-up and the percentage of efficiency are defined as follows:

$$\text{speed up} = \frac{\text{CPU time(single processor)}}{\text{CPU time(multiple processors)}}, \tag{15}$$

$$\%e_p = (\text{speed up/number of processors}) \times 100. \tag{16}$$

These variables can be defined with respect to parallel and serial results. That is, speed-up and efficiency based on parallel and serial modes are defined as:

$$\text{speed up}_{prl} = t_{p,\,total\,iterations}/t_{np,\,total\,iterations}, \tag{17}$$

$$\text{speed up}_{srl} = t_{s,\,total\,iterations}/t_{np,\,total\,iterations}, \tag{18}$$

$$\%e_{p,prl} = t_{p,\,one\,iterations}/t_{np,\,one\,iterations}/np \times 100, \tag{19}$$

$$\%e_{p,srl} = t_{s,\,one\,iterations}/t_{np,\,one\,iterations}/np \times 100, \tag{20}$$

$$\%e_{t,prl} = t_{p,\,total\,iterations}/t_{np,\,total\,iterations}/np \times 100, \tag{21}$$

$$\%e_{t,srl} = t_{s,\,total\,iterations}/t_{np,\,total\,iterations}/np \times 100, \tag{22}$$

where t_p and t_{np} are CPU times for executing the parallel code on a single and multiple processors respectively, and t_s is the CPU time for executing the serial code. Also, e_p is the parallel efficiency and e_t is the total efficiency. Indices prl and srl refer to the parallel and serial modes, and np to the number of processors. Another parameter, granularity, which is the ratio between computation time and communication time for one iteration, is defined as:

$$\text{Granularity} = t_{comp}/t_{comm}. \tag{23}$$

To ensure the correctness of the parallelized code and make an exact comparison between serial and parallel computation, parameters such as sd and sdn are defined as:

$$Sd_i = |\rho_i| + |p_i| + |e_i| + |u_i| + |v_i| + |w_i|\,, \tag{24}$$

$$Sd = \sum_{i=1}^{nc} Sd_i, \tag{25}$$

$$Sdn = \frac{Sd}{nc}, \tag{26}$$

where nc is the number of cells. The variable percent of error defined below is used to show the normalized difference between the parallel and serial results.

$$\%\ error = |Sd_{srl} - Sd_{prl}|\,/Sd_{srl} \times 100. \tag{27}$$

Note that these error norms are meaningful if the same number of iterations is used for different runs.

PARALLELIZATION METHOD

Since the PNS part of the dual-code is a marching algorithm in axial direction, *i.e.*, an ensemble of 2-D problems, it really takes no major computation time. Therefore, based on numerical experimentation, it is decided to concentrate on parallelization of the TLNS part, as mentioned earlier.

For parallelizing the TLNS solver, the domain is partitioned into several sub-domains equal to the number of processors and each sub-domain is assigned to a processor. The discretization algorithm bonds the solution at each point to the solution of six neighbors around it. In other words, each point is dependent on the two neighboring points in each direction. This necessitates communication of data in the boundaries of the sub-domains as shown in Figure 1. That is, the values of variables at the neighbors of each partition interface should be exchanged between adjacent computational nodes. In other words, the inner points of each sub-domain include the points on the boundaries of the neighboring sub-domains. Thus, each processor computes its own inner points and sends the solution on the boundaries of the other sub-domains, and receives its boundary point solution from the neighboring processors.

TLNS code used in this research [1] is a three dimensional code. Thus, the domain can be partitioned in various forms. Due to the structured grid used in this problem, one-dimensional partitioning is used. Figure 2 shows the domain and its one dimensional partitioning in longitudinal, radial and circumferential directions.

Note that, if the domain is partitioned in circumferential direction, the modification of Baldwin-Lomax model cannot be implemented in the same way as in the serial code. Figure 3 shows this problem when the domain is partitioned into two sub-domains in circumferential direction. For this special case, assume that $F_{\max}$ is frozen at the value used for the previous roll angle. Machines number 1 and 2 start the calculation simultaneously from the planes A and B. If machine 2 requires the value of $F_{\max}$ in the previous cell in circumferential direction, then two cases may occur. Firstly, machine 2 waits until machine 1 computes the value of $F_{\max}$ and sends it to machine 2. This case is similar to the serial execution with the difference that two machines are executing the same code; therefore, more time is taken than in the case of serial computation. The second case is that machine 2 takes the value of $F_{\max}$ from the previous iteration. In this case, some communications among machines occur outside the main iteration loop of the program. This results in a different solution from that of the serial code. However, as will be shown in the results section, parallel solutions remain very close to the serial solutions and parallelization can be efficiently done with a small percentage of error.

For implementation of the Baldwin-Lomax model, the program calculates the values of $F_{\max}$, $y_{\max}$, $\left(\sqrt{u^2+v^2+w^2}\right)_{\max}$ and $\left(\sqrt{u^2+v^2+w^2}\right)_{\min}$ locally in a marching process normal to the wall direction.

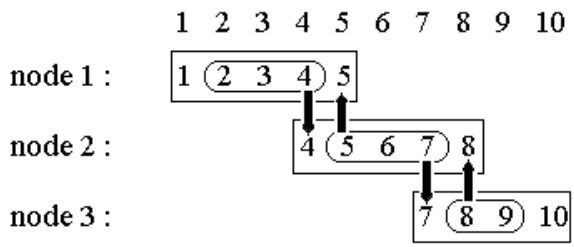

Figure 1. Data communication among 3 processors.

This is idiomatically called the direct method, which necessitates a lot of communication among processors, and increases the run time quite rapidly as the number of processors increases. To overcome this problem, these extreme values in any iteration may be obtained from the previous iteration. This is relevant when the domain is partitioned in radial or circumferential directions. However, when the longitudinal partitioning is used, the above problem does not occure. The best results can be obtained using this type of partitioning.

RESULTS

In this research, the effects of several parameters on the parallelization effectiveness are studied. The first parameter is the kind of send and receive massage. Three methods for communicating data among processors, *i.e.*; blocking send, send-receive and non-blocking send are used. A send or receive is blocking if machines wait until the message is either fully sent or received. In the non-blocking send method, machines do not wait for the send or receive to be completed. This allows the program to continue until a wait command is encountered. This wait command is usually used at the end of a send and receive session to assure receipt of all messages. The send-receive method is a blocking send with the difference that the machines receive and send the data simultaneously, [4]. As shown in Figure 4, the fastest method is non-blocking send for the problem at hand. Thus, non-blocking method with one-dimensional partitioning is used in the next cases. The slope decline in this figure is due to the increased communication load as the number of processors increases.

In this section, the effect of partitioning direction is investigated. Results of the parallelized program for longitudinal, radial and circumferential partitioning are compared. A supersonic turbulent flow field with the Mach number of 3 around a missile with incident angle of 12° and $Re = 6.4 \times 10^6$ is solved on parallel machines. The body is a cone connected to a cylinder.

Table 1. Correctness of the program for different directions of partitioning.

	Par. Direction		
Correctness	Longitudinal partitioning	Radial partitioning	Circumferential partitioning
sd_{prl}	24304.6700	24092.1626	24304.6809
sdn_{prl}	2.5160	2.4940	2.5160
sd_{srl}	24304.6700		
sdn_{srl}	2.5160		
%error	7.26×10^{-8}	0.873448	4.62×10^{-5}

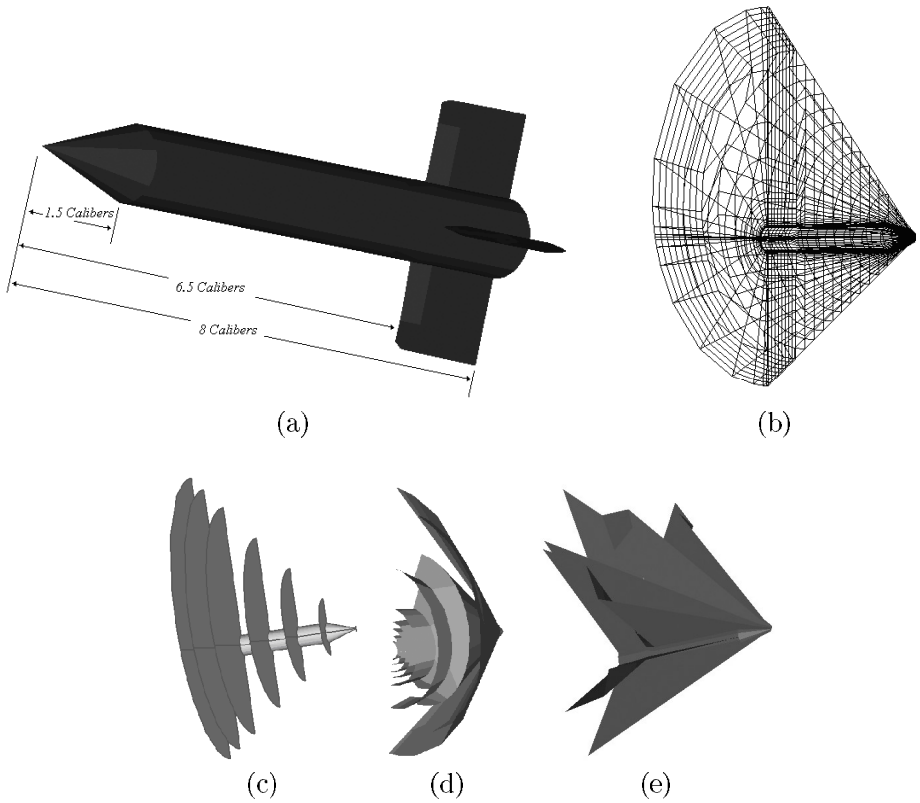

Figure 2. A schematic of the body and the domain partitioned in 1-D form among 6 processors in different directions, (a) geometry of the fin-body, (b) schematic of computational grid, (c) axial partitioning, (d) radial partitioning, (e) circumfrential partitioning.

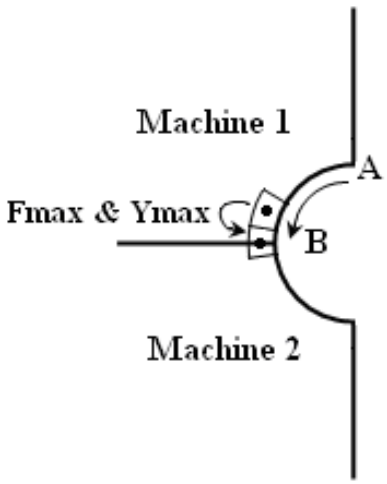

Figure 3. Data communication in circumferential direction between two processors.

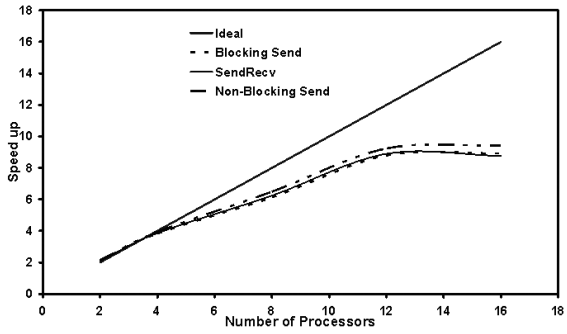

Figure 4. Different methods of communicating data among processors (A grid with 30×20×20 cells and 5000 iterations).

The number of computational grid points is 21×20×23 in longitudinal, radial and circumferential directions, respectively. Because of the symmetry shown in Figure 2 only half of the domain in circumferential direction is considered.

Table 1 shows the values of sd_{prl}, sd_{srl}, sdn_{prl}, sdn_{srl} and the percent of error for different partitioning directions. With regards to the results presented, the use of the longitudinal partitioning provides the best results compared to the radial and circumferential partitioning. Now to find the fastest method, the domain is partitioned in different directions, and run times are compared for all the cases. Figure 5 shows the comparison of run times for the longitudinal, radial and circumferential partitioning. These results are obtained for the wing-body combination shown in Figure 2 and the flow conditions as explained above. The grid points are 21×21×21 and equal number of grid points are assigned to each computational node.

Next, the effect of partitioning direction on implementation of the turbulence model is presented. As shown in Figure 5, application of direct method in circumferential and radial partitioning increases the run time rapidly as the number of processors increases. The run time increase for radial partitioning is much more than that for circumferential partitioning, particularly, when the number of machines is more than 8 processors. The reason is that the communication load for calculating the real values of $F_{\max}$, $y_{\max}$, $\left|\vec{V}\right|_{\max}$, $\left|\vec{V}\right|_{\min}$ for radial partitioning is much more than that for circumferential partitioning. This is so because in circumferential partitioning the program calculates these values only on the wings and not on the body required in radial partitioning, hence more time for communication. Altogether, the use of the direct method to find the correct values of $F_{\max}$, $y_{\max}$, $\left|\vec{V}\right|_{\max}$, $\left|\vec{V}\right|_{\min}$ causes increased run time as the number of processors increases. To overcome this problem, the values of previous iteration for $F_{\max}$, $y_{\max}$, $\left|\vec{V}\right|_{\max}$, $\left|\vec{V}\right|_{\min}$ can be used. Thus, to find the correct values of $F_{\max}$, $y_{\max}$, $\left|\vec{V}\right|_{\max}$, $\left|\vec{V}\right|_{\min}$ in the whole domain, the communication would be implemented out of the main loop of the program over ξ, ζ and η directions.

Table 2. Results of parallelizing the TLNS code.

Number of processors	$t_{total}(hours)$	$\frac{\text{Second}}{\text{Iteration}}$	Granularity
1 (Serial)	110.092	9.00664	——
1(Parallel)	155.49	12.8146	17.273
2	78.2383	6.4751	16.414
4	39.8264	3.3153	13.518
8	20.4825	1.6741	9.268
12	14.10003	1.2085	6.941
16	10.9261	0.9430	4.685

Table 3. Performance of the TLNS parallel code (based on parallel code).

Number of processors	Speedup_{prl}	$\%e_{t,prl}$	$\%e_{p,prl}$
1 (Parallel)	1	100	100
2	1.9874	99.37	98.95
4	3.9042	97.61	96.63
8	7.5914	94.89	95.68
12	11.0275	91.9	88.36
16	14.2311	88.94	84.93

Table 4. Performance of the TLNS parallel code (based on serial code).

Number of processors	Speedup_{srl}	$\%e_{t,srl}$	$\%e_{p,srl}$
1 (Parallel)	0.7080	70.80	70.2843
2	1.4071	70.355	69.5479
4	2.7643	69.1075	67.9170
8	5.3749	67.1863	67.2505
12	7.8079	65.0658	62.1048
16	10.0760	62.975	59.6938

This reduces the computation time as the number of machines increases. This phenomenon is also shown in Figure 5 for radial and circumferential partitioning.

With view of the results presented, it is clear that the longitudinal partitioning decreases the computation time more than the other types of partitioning. Also, the results obtained using this type of partitioning are more accurate than other results (Table 1).

At this stage, a supersonic turbulent flow with Mach number of 3 around a tangent-ogive with incident angle of 6° and $Re_\infty = 6.4\times 10^6$ is solved in parallel and serial modes and the results are compared with each other. The model consists of a tangent-ogive connected to a cylinder with a total length of 6 calibers and the cylindrical part of 3 calibers. The number of computational grid points is $150 \times 50 \times 40$ in longitudinal, radial and circumferential directions, respectively. Tables 2 through 4 show the speed-up and efficiency obtained from executing the serial and parallel codes on 1 up to 16 nodes. As shown in these tables and in Figure 6, as the number of machines increases, the rate of speed-up decreases, and so does the efficiency. Also, increasing the number of machines causes the computation time and by extension the granularity to decrease.

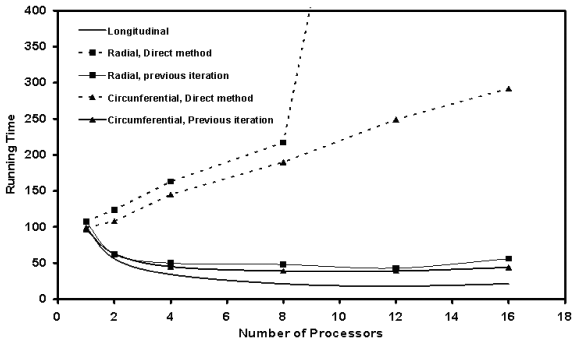

Figure 5. Comparison of the computation time between longitudinal, radial and circumferential direction partitioning (supersonic flow about the wing-body combination with 21×21×21 grid points and 500 iterations).

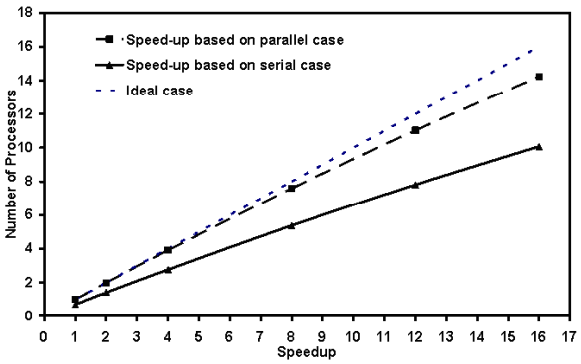

Figure 6. Relation between speed-up and the number of machines (Supersonic flow about wing-body combination with 150×50×40 grid points).

CONCLUSION

In this paper a Thin Layer Navier-Stokes solver is parallelized to work on distributed memory systems. MPI library is used for data communication among processors. The program is parallelized for 1-D partitioning in longitudinal, radial and circumferential directions over a 3-D wing-body combination. The effect of partitioning direction and different kinds of send and receive are studied. Although, no large differences in the performance of the parallel code were observed when blocking and non-blocking send and receive were used, the interaction of partitioning direction with the implementation of the specific turbulence model is noticeable. The problems of using Baldwin-Lomax turbulence model for parallel programming were addressed and a new strategy was introduced to improve parallelization efficiency. The obtained results show that the longitudinal partitioning is the best choice when Baldwin-Lomax turbulence model is used. Also when using longitudinal partitioning together with non-blocking send, the efficiency and speed-up of the parallel code are both noticeable. Therefore, using the present parallel code, computation of 3-D complicated flows would be feasible and efficient.

REFERENCES

1. Abouali O., "Dual-Code Thin Layer, Parabolized Navier-Stokes Solution for Supersonic Flows Over Spinning Wing-Body Configurations", Ph.D. Thesis, Shiraz University, Shiraz, Iran(2003).

2. Abouali O., Alishahi M. M., Emdad H. and Ahmadi G., "Dual-Code TLNS-PNS Strategy for 3-D Supersonic Flows over Spinning Bodies", *Journal of Spacecraft and Rocket*, **40(6)**, PP 893-897(2003).

3. Alishahi M. M., Emdad H. and Abouali O., "3-D Thin Layer Navier-Stokes Solution of Supersonic Turbulent Flow", *Scientia Iranica*, **10(1)**, PP 74-83(2003).

4. Gropp W., Lusk E., Skjellum A., "Using MPI, Portable Parallel Programming with the Message-Passing Interface", Massachusetts Institute of Technology, (1994).

5. Ruiz-Calavera L. P. and Hirose N., "Implementation and Results of a Time Accurate Finite-Volume Euler Code in the NWT Parallel Computer", *Parallel CFD Conference*, Pasadena, CA, U.S.A, PP 183-190(1995).

6. Drikakis D., "Development and Implementation of Parallel High Resolution Scheme in 3D Flows Over Bluff Bodies", *Parallel CFD Conference*, Pasadena, CA, U.S.A, PP 191-198(1995).

7. Bonisch T. and Ruhle R., "Portable Parallelization of a 3-D Flow-Solver", *Parallel CFD Conference*, Pasadena, CA, U.S.A, PP 457-464(1997).