Parallelization of a Finite-Volume Navier-Stokes solver on a T3D massively parallel system

G. Bärwolff*, K. Ketelsen and F. Thiele***

*TU Berlin, Sekr. HF 1-Hermann-Föttinger-Institut
Straße des 17. Juni 135
D-10623 Berlin, Germany

**CRAY Research GmbH
Riesstrasse 25
D-80992 Munich, Germany

INTRODUCTION

Computational Fluid Dynamics is one of the great challenges in todays supercomputing. To solve high resolution problems cost-effectively, the use of massively parallel processing systems will become necessary. MLET is a finite-volume solution procedure for the 3-D Navier-Stokes equations developed at the Universität der Bundeswehr in Munich and is modified here to examine flow problems with active boundary control. To perform calculations with MLET for Reynolds numbers of interest, three dimensional grids with a couple of million cells are necessary. Running problems of this size leads to very large turnaround times in existing production environments. With the availability of the CRAY T3D at the ZIB[3] in Berlin the authors expected a much better performance of the MLET code, after being properly implemented on this massively parallel system.
This paper describes the transfer of the vector version of MLET to a Massively Parallel Processing (MPP) system. The numerical solution method for the nonstationary 3-D Navier-Stokes equation based on a spatial finite volume discretization and its parallel modification is described briefly. The parallelization strategy is discussed, followed by a description of the domain decomposition and the message passing tools.
The results of a backward facing step problem, using a grid with in excess of 11 million points, are shown as isoline plots. The same example is used to show the performance and scaleability of MLET for MPP as well as parallel vector systems.

1. THE MATHEMATICAL MODEL AND THE SOLUTION METHOD

The basic equations to describe laminar as well as turbulent (DNS or LES) incompressible flow problems are the Navier-Stokes equations:

$$\frac{\partial \vec{u}}{\partial t} + \nabla \cdot \vec{u}\vec{u} \quad = \quad -\nabla p + \nabla(\nu \nabla \vec{u}) + \vec{f} \quad , \qquad (1)$$

and the continuity equation:

$$\nabla \cdot \vec{u} \quad = \quad 0 \quad . \qquad (2)$$

[3]Konrad-Zuse-Zentrum für Informationstechnik

Equations (1) and (2) may be discretized in spatial dimensions by a finite volume method on staggered grids for the velocity components u_i ($\vec{u} = (u_1, u_2, u_3)$) and the pressure p. ν is the effective or molecular viscosity and $\vec{f}$ is a body force vector.
The finite volume discretization results in a system of ode's for the velocity components at every grid point:

$$\frac{\partial \vec{u}_h}{\partial t} + \nabla_h \cdot \vec{u}_h \vec{u}_h \quad = \quad -\nabla_h p_h + \nabla_h (\nu \nabla_h \vec{u}_h) + \vec{f}_h \quad , \tag{3}$$

with the restriction $\nabla_h \cdot \vec{u}_h = 0$. The approximation is conservative and is of second order ($O(h^2)$). The time integration is done either by a leapfrog method or an Adams-Bashforth method. Thus, we have to solve in every time step the equation system:

$$\frac{\vec{u}_h^{n+1} - \tilde{\vec{u}}_h}{\tau} = -\beta \nabla_h p_h^{n+1} \quad , \quad \nabla_h \cdot \vec{u}_h^{n+1} = 0 \, , \tag{4}$$

where $\tilde{\vec{u}}_h$ is a given result of an estimation by a predictor step, τ is the time step and β is a constant depending on the time integration method used (in the case of the Adams-Bashforth method $\beta = 1$, see also [1]).
The equations (4) are equivalent to the equation:

$$-\Delta_h p_h^{n+1} = -\frac{1}{\tau \beta} \nabla_h \cdot \tilde{\vec{u}}_h \quad . \tag{5}$$

$\vec{u}_h^{n+1}$ is then given by an explicit fill-in step following (4). Thus, $(\vec{u}_h^{n+1}, p_h^{n+1})$ can be found either iteratively or by the solution of a Poisson equation for p^{n+1} followed by an explicit fill-in step to get $\vec{u}^{n+1}$. Both possibilities are implemented in the sequential codes (see [1],[2]). In our parallel code implementation the iterative solution method to get $(\vec{u}_h^{n+1}, p_h^{n+1})$ is realized as a solver for the equation system (4). The parallelization of the solution of the system (5) is now under consideration.

2. THE PRINCIPLES OF PARALLELIZATION

MLET is a 3-D CFD code using a regular grid. Domain decomposition has been used to parallelize the code. A grouping has been done in the X- and Y-directions while the Z-direction remains local. Figure 1 shows the distribution of the problem to different Processing Elements (PE's).
With a two dimensional domain decomposition it is much easier to adapt the problem size to the number of PE's. For example, on an MPP with 256 PE's, the number of cells must be a multiple of 16 in the X- and Y-directions. In the case of a one-dimensional domain decomposition the number of cells in one direction must be a multiple of the number of PE's.
The iterative solver for the equation system (4) is a pressure velocity iteration method described in [1]. The method is of the form (for simplification, the subscript h of $\vec{u}$ and p to mark grid functions is not written in the following formulas)

$$p_i := p_i + \zeta \, \delta p \quad , \tag{6}$$

$$u_{1_i} := u_{1_i} + \eta \, \delta p \quad , \tag{7}$$

$$u_{1_{i-1}} := u_{1_{i-1}} - \eta \, \delta p \quad , \tag{8}$$

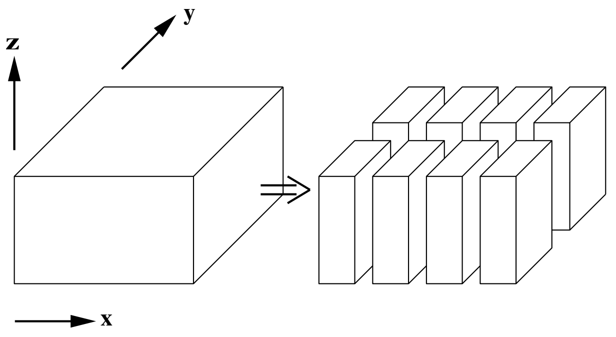

Figure 1: Domain decomposition

with $i = 1, 2, ..., N$ as the subscript of the X-direction. δp is a linear function of the divergence of the current cell $(\nabla_h \cdot \vec{u}_h)$, which means δp is a function of the actual values of u_{1_i}, $u_{1_{i-1}}$, u_{2_j}, $u_{2_{j-1}}$, u_{3_k} and $u_{3_{k-1}}$. For the Y- and the Z-directions we have analogous iteration formulas.
With the formulas above, which have to be calculated for every cell, we have a conflict at the interface between the subdomains of two neighbouring PE's. The original step-by-step iteration is a sequential one and means that the most recent value of u_{1_n} is needed to get $u_{1_{n+1}}$.
This recurrency leads to a conflict at the boundaries because a PE cannot begin with the calculation before the previous one has finished the calculation of all its u values.
To eliminate the recurrency of this algorithm, the computation is done in two passes, first for the odd cells and in the second stride for the even. This chessboard-like iteration allows us to do synchronized iterations in subdomains with an interchange of boundary values between an odd (black) and an even (white) iteration stride. This modification of the iteration algorithm allows us to work in parallel, because in the first iteration pass on every PE only "old" information of neighbouring PE's is needed. In the second pass the "new" values of the neighbouring PE's are available because of the interchange of boundary values done between the two iteration strides.
Other modifications of the numerical solution method are not necessary for the parallel implementation of the method. The modification of the iteration method was validated in the sequential code on the Y-MP.
To exchange boundary information between the subdomains "explicit shared memory message passing" has been used. This way of communication takes into account the global address space of the T3D and allows communication between PE's with **high bandwidth** and **little latency**.
All communication is done in a set of communication subroutines to allow an easy port to other message passing methods like, for example, PVM. Most of the subroutines can be used without any changes in the parallel code. We have to modify only those routines which handle the setting of the boundary conditions, because it has to be decided if a boundary of a PE subdomain is a real boundary of the original global integration domain or not. Further routines which are responsible for the computation of global sums and global maxima must be modified.
Strategies which use a virtual global memory concept like Cray's FORTRAN extension

Type	NPES	PE_X*PE_Y	Iteration time	Mflops	Fact or
			[sec]		C90
C90	1		4380	338	1.0
C90	4		1323	1119	3.31
T3D	32	8*4	3433	431	1.27
T3D	64	8*8	1737	852	2.52
T3D	128	16*8	873	1695	5.01
T3D	256	16*16	458	3195	9.56
T3D	512	32*16	345	4291	12.69

Table 1: Performance and scaleability of MLET

CRAFT have the advantage of fewer modifications to the code. Especially the handling of boundaries is done implicitly, i.e. the developers do not have to take care of boundary values. The payoff is a performance degradation because the use of shared arrays requires more global access of data than exchanging boundaries via message passing.
Every PE handles the restart and plot files according to its domain. Simple pre- and postprocessing steps have been implemented which split the global files into domains and vice versa before and after running a T3D program.

3. THE PERFORMANCE AND SCALEABILITY

The performance of the parallel version of MLET is shown for a reference job. This performance is also compared with single and multiple CPU versions on a CRAY C90.

The reference job computes 100 time steps of a FV-grid with 11083776 cells. The dimension of the problem is 516x132x164. An equation system with approximately 45 million unknowns has to be solved for every time step. The solver uses an average of 25 iterations.

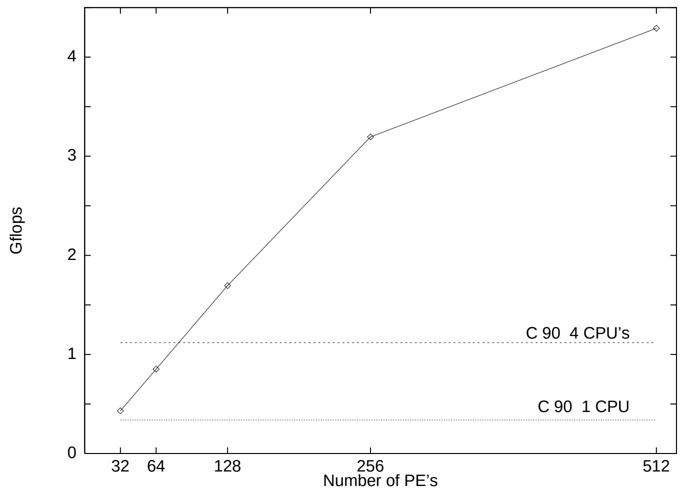

Figure 2: Performance of MLET

Table 1 shows the runtime and performance of the reference jobs on different configurations of a T3D and C90. In case of the T3D, the grouping information, i.e. the

number of PE's in the X- and Y-directions is also shown. Huge CFD problems are not only very computationally intensive, they also need a huge amount of memory. The test job requires approximately 1.3 gigabytes of memory.
To show the performance and scaleability of the parallel version of MLET, several runs of a three-dimensional backward facing step flow problem with periodic boundary conditions (reference job) have been done on different numbers of PE's. Comparison of these runs with the sequential CRAY C90 version is shown in Figure 2.

4. THE RESULTS OF THE SOLVED REFERENCE PROBLEM

The geometry of the flow region is rectangular backward-facing step channel. The Reynolds number calculated using the step height is approximately equal to 3000. In the spanwise direction periodic boundary conditions are used. The inflow profile is a block profile with $u_1 = u_x = 1$ and $u_2 = u_3 = 0$. At the bottom of the channel the no-slip boundary condition is assumed. At the top of the region boundary conditions like $\frac{\partial \vec{u}}{\partial \vec{n}} = 0$ with the outer normal vector $\vec{n}$ are used. The outflow boundary condition is set to $\frac{\partial \vec{u}}{\partial x} = 0$.
The structured grid consists of 516 cells in the X-direction, 132 cells in the Y-direction (spanwise) and 164 cells in the Z-direction. This fine grid allows us to perform a DNS and thus a subgrid-scale model is not necessary.
To get a good first and second order statistic ($\overline{\vec{u}}$, $\overline{u_i' \, u_j'}$, and so on) we need a few hundred thousand flow realizations or instantaneous flow fields or time steps respectively.

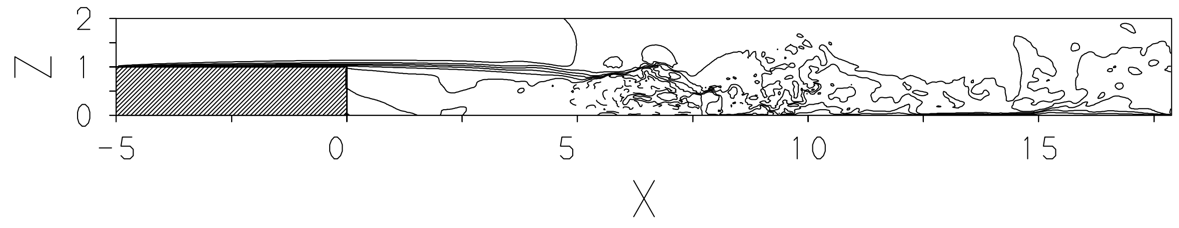

Figure 3: velocity isolines of u_1, $t = 180$

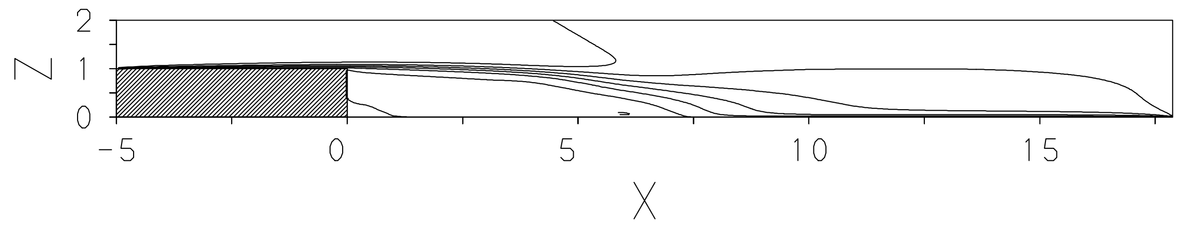

Figure 4: velocity isolines of $\overline{u_1}$, $t = 180$

Using 64 PE's on the CRAY T3D a production chain for the problem described above takes about 600 hours CPU, which was achieved in two months. Due to the amount of memory required, the reference job could not run on the Y-MP production environment. Smaller problems with approximately 2 million grid points took about 10 months on the Y-MP.
The figures 3 and 4 show results of the backward facing s tep problem as isoline plots

of an instantaneous velocity field and the mean velocity field (statistic of first order) in an X-Z cut (symmetry plane, near bottom region).
This problem is a part of a parameter study to control the magnitude of the recirculation region behind the separation edge by blowing with a prescribed frequency.

5. CONCLUSION

This paper has shown the implementation of the MLET code on a massively parallel system. Using domain decomposition, a scalable version has been created which allows the use of the MPP system in connection with comfortable pre- and postprocessing steps transparent to the user.
The development of the parallel algorithm and its implementation results in a performance which allows the solution of large problems, for example flow problems with higher Reynolds numbers and very fine grids to resolve all important flow structures, on the T3D systems. Due to memory and CPU restrictions, it was impossible to handle these kinds of problems on existing vector production environments. Comparing the performance of the T3D with vector computers show that 9 PE's are equivalent to 1 Y-MP CPU and 25 PE's are equivalent to 1 C90 CPU.
The whole port of the code has been accomplished in 4 weeks. The parallel version of MLET is now used for production runs on the T3D for the modeling of turbulent flows on grids with more than 11,000,000 cells and a very fine time resolution to realize periodic stimulations of approximately 50 Hz of the flow over the boundary conditions.

References

[1] Werner, H., Grobstruktursimulation der turbulenten Strömung über eine querliegende Rippe in einem Plattenkanal bei hoher Reynoldszahl, PhD thesis, TU München 1991,

[2] Bärwolff, G. and Seifert, G., Efficient 2D and 3D Navier-Stokes solver, Proceedings of the 5. ISCFD Sendai/Japan, 1993 (Ed. H. Daiguji)