34th AIAA Aerospace Sciences Meeting and Exhibit

Parallelization of a Parabolized Navier-Stokes Solver with a Design Optimizer

J. M. Pallis and J. J. Chattot
University of California, Davis, Davis, California 95616
S. L. Lawrence
NASA Ames Research Center, Moffett Field, California 94035

The design of future supersonic aircraft, such as the High Speed Civil Transport (HSCT), will rely heavily on computational methods for aircraft design and the prediction of the complex aerodynamics encountered in flight. Parabolized Navier-Stokes (PNS) equation flow solvers are recognized as efficient and accurate computational tools for the solution of supersonic and hypersonic flowfields, while design optimizers have the potential to be valuable tools within the overall design process. Presently, however, the execution of the flow solver in conjunction with a design optimizer presents a computationally intensive and formidable problem. To meet the challenges and increasing demand for multidisciplinary numerical tools which are faster, more robust and provide greater functionality, alternative strategies are explored to increase computational throughput by coupling a design optimizer and flow solver in a parallel processing environment. To address this problem the parallel processing of a PNS flow solver with a nonlinear constraint design optimizer is investigated as an alternative computational method.

Introduction

To maintain industrial leadership in the design of aircraft in the current global economy, it has become essential to integrate efficient, robust computational tools to provide a reduction in time, cost and a competitive edge in the overall design cycle. The design of future supersonic aircraft, such as the High Speed Civil Transport (HSCT), will rely heavily on the development of computational methods for aircraft design and the determination of the complex aerodynamics encountered in flight.

Parabolized Navier-Stokes (PNS) equation solvers are recognized and utilized as an established computational tool for the solution of supersonic and hypersonic flow fields, while design optimizers have been adopted as a computational efficient instrument within the overall design process. The execution of the flow solver in conjunction with an optimizer presents a computationally formidable

problem.

One approach to increase the computational throughput is to exploit alternative computational methods such as parallel processing. In such cases the original sequential problem is split or decomposed into smaller subproblems. These subproblems are then distributed among several (or many) processors and executed concurrently.

The specific objective of the current research has been to develop and implement a parallel perfect gas version of a Parabolized Navier-Stokes solver (PNS) which can be distributed and executed over multiple computer processors concurrently in conjunction with a design optimizer.

The PNS flow solver utilized is the Upwind Parabolized Navier-Stokes solver (UPS) of Lawrence [4]. UPS is a 3D upwind space marching code which has been used extensively for supersonic and hypersonic conditions for perfect, equilibrium and nonequilibrium gases. The design optimizer of Cheung [2], *II*OWA, is the nonlinear constraint optimizer based on the quasi-Newton method.

In the current research a flow solver is coupled with a design optimizer on a massively parallel processor. Different strategies for implementing these multidisciplinary codes in a parallel environment are discussed. Comparisons between the throughput of a massively parallel processor platform versus a workstation platform are made. Performance tuning strategies for the single processor execution of the flow solver are also presented due to the significant impact these exercises have in the overall computational throughput.

Parabolized Navier-Stokes Equation Formulation

The system of equations used to model the fluids and chemistry are the Parabolized Navier-Stokes equations. The PNS equations are obtained from the full Navier-Stokes equations by neglecting the unsteady terms and the viscous terms which involve derivatives in the streamwise direction. In generalized coordinates the governing equations are:

$$\frac{\partial E}{\partial \xi} + \frac{\partial F}{\partial \eta} + \frac{\partial G}{\partial \zeta} = 0$$

where

$$E = \left(\frac{\xi_x}{J}\right) E_i + \left(\frac{\xi_y}{J}\right) F_i + \left(\frac{\xi_z}{J}\right) G_i$$

$$F = \left(\frac{\eta_x}{J}\right)(E_i - E_v^*) + \left(\frac{\eta_y}{J}\right)(F_i - F_v^*) + \left(\frac{\eta_z}{J}\right)(G_i - G_v^*)$$

$$G = \left(\frac{\zeta_x}{J}\right)(E_i - E_v^*) + \left(\frac{\zeta_y}{J}\right)(F_i - F_v^*) + \left(\frac{\zeta_z}{J}\right)(G_i - G_v^*)$$

and

$$
\begin{aligned}
E_i &= (\rho u, \rho u^2 + p, \rho uv, \rho uw, (E_t + p)u)^T \\
F_i &= (\rho v, \rho uv, \rho v^2 + p, \rho vw, (E_t + p)v)^T \\
G_i &= (\rho w, \rho uw, \rho vw, \rho w^2 + p, (E_t + p)w)^T \\
E_v &= (0, \tau_{xx}, \tau_{xy}, \tau_{xz}, u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x)^T \\
F_v &= (0, \tau_{xy}, \tau_{yy}, \tau_{yz}, u\tau_{xy} + v\tau_{yy} + w\tau_{yz} - q_y)^T \\
G_v &= (0, \tau_{xz}, \tau_{yz}, \tau_{zz}, u\tau_{xz} + v\tau_{yz} + w\tau_{zz} - q_z)^T \\
E_t &= \rho[e + \frac{1}{2}(u^2 + v^2 + w^2)]
\end{aligned}
$$

In the above equations ρ is the density; u, v, w are the velocity components in the x, y, z directions; p is the pressure; τ is the viscous stress; e is the internal energy and q is the conduction heat flux. The terms E_v^*, F_v^* and G_v^* indicate that the streamwise derivatives (with respect to ξ) have been neglected.

Parallel Implementation

The UPS code developed by Lawrence [4] has been used as the basis for the development of the parallel perfect gas version. Research conducted by Pallis [5, 6] and Stagg [8] suggest that parallel algorithms for matrix manipulation, implicit schemes, domain decomposition and efficient data routing all contribute to a scalable robust parallel version of UPS.

To initially confirm the applicability of the UPS code to parallelization, an analysis was conducted for the perfect gas portions of the UPS code. This analysis addressed which segments of the code were candidates for fine, medium or coarse grain parallelization and highlighted such areas as candidates for parallel algorithms or concurrent code execution.

Subsequently, the sequential version of the UPS equation solver was decomposed in a zonal fashion into several smaller subproblems. The subproblems were then executed on a parallel computer configuration, specifically a 128 node IBM SP2 utilizing two different interface libraries (PVM and MPI).

Two different approaches were implemented. The first approach is based on the work of Cheung [3] (Figure 1). Utilizing a set of heterogeneous workstations (SGI, HP) and the Parallel Virtual Machine (PVM) [1] communication software, the *II*OWA/UPS application was distributed.

The parallel perfect gas version of the code divides the computational domain in a zonal fashion in the circumferential and normal directions. The subroutines in the single node version which

were the most resource intense, especially as additional parameters were added, were tuned and parallelized for the SP2 platform. Obviously these routines dealt with grid generation and flow solution. It is important to note that there were aspects of the code which lent itself to parallelization but sufficient amounts of code were not executed to make it worth while to incur the overhead of message passing even though those routines possessed a natural division of the workload. Initially, all nodes passed general information regarding the first plane of data. Subsequently, only processors sharing data between node boundaries exchanged data. The code automatically divides the computational domain between 4 to 500 processors. Rearrangement in the flow solver of some of the data in the common areas of storage was performed to take advantage of the cache structure and functionality of the IBM SP/2 processors.

The optimization algorithm of *II*OWA uses a shape perturbation method where each design variable is perturbed to locate a favorable search direction. For each perturbation a new surface grid is generated. The surface grid is used as input to UPS which produces a computational grid, a flow solution and subsequently the value of the objective function, which in this study was the drag coefficient. This process is repeated until a modified body shape which produces a local minimum objective function is obtained.

In this initial approach, *II*OWA operates on one workstation while the new surface grid generation and flow solver UPS operate on the other workstations. In this implementation, there is a workstation assigned (and thus a copy of UPS) executing for each design variable.

The Haack-Adams (H-A) theoretical minimum drag body of revolution is chosen as the test case, since validation can be made against experimental data and computational performance can be compared to the work of Cheung [2]. The H-A body shape is based on classical supersonic slender body theory. The constraints the modified shape must conform to are based on the base area and volume. An addition benefit of utilizing this specific geometry with the PNS flow solver is that the finite base accommodates the space marching methodology of the PNS flow solver.

One test case used the Haack-Adams body at Mach 2.5, angle of attack at 0^o , and a Re of 9 x 10^6. The step size used was 1% of the body length with the grid consisting of 50 points in the normal direction and 21 points circumferentially. A sting has also been added to the geometry. Both, inviscid and viscous cases were performed and the wave drag coefficient CD was used as the objective function. Three, four, five and six design variables were tested.

Although the SP2 implementation significantly improved the overall execution time for this geometry, obviously as the grid size increases the UPS execution time increases. Since UPS execution time is the major contributor to the overall execution time of the *II*OWA/UPS system, a second strategy has been developed (Figure 2).

In this implementation, a single copy of UPS is split into multiple parts with a zonal decomposition, distributing the flow solver over four (or more) SP2 nodes. Similar work on a hypercube architecture by Pallis [6] demonstrated promise for a faster processor like the IBM SP2. Thus for each design variable, multiple parallel UPS codes operate concurrently. In Figure 2, for each of the 5 design variables, UPS is split into 4 parts; thus 20 nodes are needed.

In the current study this implementation was brought over to a homogenous node IBM SP2 (IBM Scalable POWER2 parallel system) (Figure 2). The IBM SP2 at the NASA Ames Research center is a 160 node (processor) distributed memory system composed of RS6000/590 workstations connected via a high speed switch. The architecture of each processor is capable of executing six instructions per clock cycle. Operating at 66.7 MHz, the peak rating of each processor is 266 MFLOPS. Communications between the processors is achieved by explicit message passing.

Experience with single node performance on the SP2 suggested an execution time reduction over the workstation configuration. The IBM SP2 has a large primary cache system. Past benchmarks revealed that the hacall segment of the code executed in a few seconds, while the UPS segment executed, on the average, in 6 minutes. Thus, UPS was tuned to take advantage of the caches and architecture of the 590 workstations. Code and memory rearrangement as well as loop unrolling resulted in an execution time of approximately 3 1/2 minutes per UPS execution.

As with many coupled distributed systems, there is a common file system on the IBM SP2 system. Thus, when referencing a data file, all nodes can utilize that file. This may inherently create an I/O bottleneck. For example, if all nodes require the use of an input file and all nodes attempt to access that file concurrently throughout the code execution, overall performance will suffer. Therefore, in the IIOWA/UPS execution, local memory on each 590 workstation was utilized as much as possible, to prevent this situation.

RESULTS

The workstation implementation utilizing PVM executed in approximately 8 hours on a set of 5 dedicated machines [3]. Although the CRAY version executes in 2.5 hours the important contribution of the workstation implementation is its cost effectiveness. The workstation implementation utilizes otherwise unused and wasted computer resources [7]. For small business and industry this type of coupling can be very cost effective.

The wallclock time for the same H-A benchmark on the IBM SP2 platform is approximately 2 hours. In all cases, the CPU resources were dedicated to the execution of the IIOWA/UPS codes. The IBM SP2 platform clearly provides superior throughput.

To study the speedup of the decomposed UPS version conical geometries at 0 degrees angle of attack was used at supersonic velocities (Mach 2.5). The grid consists of 200 points in the normal direction and 400 points circumferentially. For a 6 node parallel execution the speed up was 61% of the original single version executed on the HP workstation. For the viscous case the speedup was 48%.

The decomposed version was then coupled with the design optimizer using the H-A test case. Both inviscid and viscous cases were performed with the drag coefficient C_D used as the objective function. The IBM SP2 implementation executes in less than 50% of the workstation implementation on a set of 6 dedicated nodes. Although this is a significant improvement further improvements were sought.

Preliminary results with the decomposed UPS version have demonstrated a further reduction in the execution time. In the inviscid test case, 24 nodes were used coupled with the design optimizer. This particular test case executed in 43.5% of the original workstation code.

Conclusions

A parallel version of UPS for perfect gas calculations has been developed for the IBM SP2 at the NASA Ames Research Center. The parallel version has been benchmarked against the sequential version to validate proof of concept for perfect gas calculations, speed and accuracy. The parallelization allows improved computational throughput and the ability to study more complex geometries and conditions. Tuning exercises have contributed a significant reduction in the overall execution of this application.

REFERENCES

1. Beguelin, A., Dongarra, J., Geist, A., Manchek, R., Sunderam, V., "A User's Guide to PVM (Parallel Virtual Machine)", ORNL/TM-11826, 1992.

2. Cheung, S., Aaronson, P., and Edwards, T., "CFD Optimization of a Theoretical Minimum-Drag Body," AIAA Paper 93-3421, August 1993.

3. Cheung, S., "Aerodynamic Design: Parallel CFD and Optimization Routines," AIAA Paper 95-1748, June 1995.

4. Lawrence, S. L., Tannehill, J. C. and Chaussee, D. S., "Upwind Algorithm for the Parabolized Navier-Stokes Equations," AIAA Journal, Vol. 27, Sept. 1989, pp. 1175-1183.

5. Pallis, J. M. and Chattot, J. J., "Implementation of a Nonequilibrium Flow Solver on a Massively Parallel Processor," 14th International Conference on Numerical Methods in Fluids, July, 1994.

6. Pallis, J. M. and Chattot, J. J., "Parallel Implementation of a Chemical Nonequilibrium Flow Solver", CERCA, September, 1994.

7. Smith, M. H. and Pallis, J. M., "MEDUSA - An Overset Grid Flow Solver for Network-Based Parallel Computer Systems," AIAA Paper 93-3312, June 1993.

8. Stagg, A. K., Cline, D. D., Carey, G. F. and Shadid, J. N., "Parallel, Scalable Parabolized Navier-Stokes Solver for Large-Scale Simulations," AIAA Journal, Vol. 33, No. 1, January 1995, pp. 102-108.

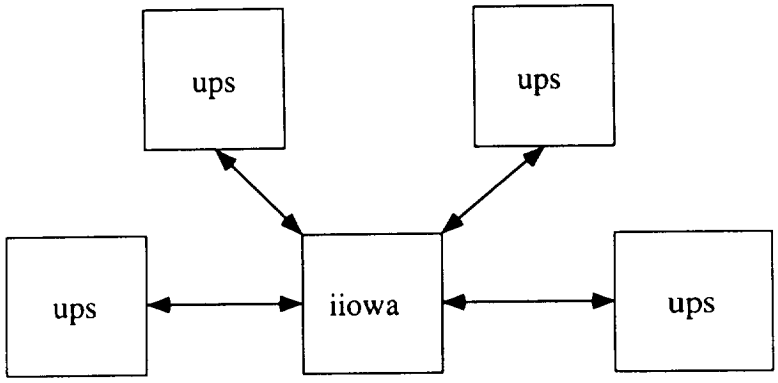

Figure 1 IBM SP2 Configuration

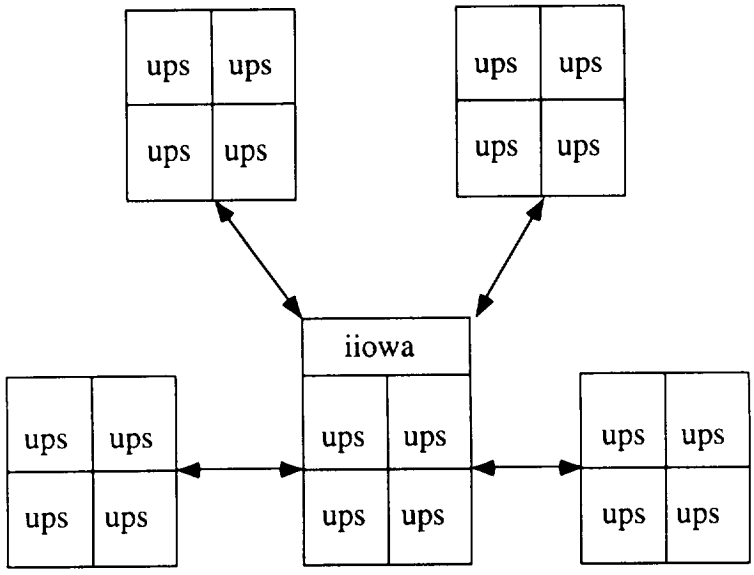

Figure 2 Multiple UPS for each Design Variable