Overset Methods, Inc.
A Nonprofit Public Benefit Corporation

OMI 02-93

FINAL REPORT
for NASA GRANT NCC 2-806
SUBMITTED TO

NASA Ames Research Center
Computational Aerosciences Branch

Point of Contact:
Dr. Terry L. Holst

By

Overset Methods, Inc.
262 Marich Way
Los Altos, CA 94022

Aerodynamic Optimization Studies on Advanced Architecture Computers

Principal Investigator: Dr. Kalpana Chawla

February 28, 1995

Design Optimization: Sensitivity Analysis of the Euler Equations as Discretized in ARC3D

K. Chawla
Research Scientist, Overset Methods, Inc.
NASA Ames Research Center
Moffett Field, CA

INTRODUCTION

The computer age has finally advanced to a stage where multi-discipline optimization can be entertained in the aerospace arena. The various disciplines to be optimized together within the aerospace field include not only the technical design areas such as aerodynamics, structures, propulsion, and controls, but also the whole engineering process itself. As various disciplines are interdependent, the assumption is that the optimized result of a single discipline must not dictate inputs for another discipline in a sequential fashion leading to the design of the final product. Instead, the disciplines must be optimized in parallel, and optimization iterations must be carried out across disciplines until satisfactory design is obtained. Engineering processes such as manufacturing are to be included for optimization with the objective of minimizing the costs and reducing the time associated with the production of the end item. This work is focused on aerodynamic optimization, however, the selected approach is viable for multi-discipline optimization as well.

In the field of aerodynamic optimization, there are three distinct technical areas. The first area concerns formulation of geometry functions. These functions either sit atop the existing design or represent the design itself. Perturbations of these functions allow new designs. Wagner functions, polynomials, sine bumps, B-splines are some examples of geometry parameterization. It is critical to select a class of functions which will allow the formulation of as complete a design space as possible. This area of optimization is not addressed in this work.

After suitable functions have been chosen to represent and optimize the geometry, we need to determine the direction in which the design variables should be changed to obtain the optimum design. For example, if the designer is trying to minimize an objective function such as Drag, the optimization process requires determination of sensitivity of drag to the change in design variable, viz. the sensitivity gradient. The present work is targeted to efficient evaluation of these sensitivity gradients.

Finally, the third area of optimization deals with the determination of the size of perturbation applied to the design variable so that one reaches the optimum design via. the path of steepest descent. A number of well developed tools are already available to accomplish this task.

Traditionally, optimization has been carried out to improve the design of a single discipline using

- Direct methods,
- Indirect methods,

- and Inverse methods.

In direct optimization techniques, one begins with the objective function, i.e. the quantity to be optimized, on a baseline configuration. Sequentially, one design parameter at a time is changed while others are kept fixed to determine the direction in which the design parameter should be varied to maximize the objective function. Constraints satisfying the final design requirements can be applied during the optimization process. This type of approach allows control over final design configuration and will be suitable for multi-discipline optimization problems.

In the inverse method approach, a target result, e.g. pressure distribution, is specified and the geometry is modified till the target value is attained. This approach, and indirect methods, however, do not allow complete control over the final configuration, and as a result have lost popularity in the recent past.

In this research effort, a "direct" optimization method is implemented on the Cray C-90 to improve aerodynamic design. It is coupled with an existing implicit Navier-Stokes solver, OVERFLOW[1], to compute flow solutions. The optimization method is chosen such that it can accommodate multi-discipline optimization in future computations. In this work, however, only single discipline aerodynamic optimization will be included.

APPROACH

The approach to carrying out multi-discipline aerospace design studies in the future, especially in massively parallel computing environments, comprises of choosing 1) suitable solvers to compute solutions to equations characterizing a discipline, and 2) efficient optimization methods. In addition, for aerodynamic optimization problems, 3) smart methodologies must be selected to modify the surface shape.

Solver:

An implicit code to solve the Navier-Stokes equations, OVERFLOW, is chosen as it has already been mapped to a number of parallel environments such as the TMC CM-5[2], the Intel iPSC-860[3], the Intel Paragon, and a network of workstations[4], thus offering a number of possibilities for mapping the optimization component. A number of options, such as Pulliam and Chaussee's diagonalized scheme[5], Block Beam-Warming scheme[6], and Steger's partially flux split algorithm[7] are coded in OVERFLOW. The first two options are from the ARC3D code. In the OVERFLOW code, the ARC3D options are written for an overset grid framework thus allowing the possibility of design optimization of configurations that do not lend themselves easily to single grid topologies. Additionally, a number of related disciplines such as dynamics[8,9], controls[10], and propulsion[11] have already been coupled with this code, permitting multi-discipline optimization studies in the future.

Optimization Method:

As outlined in the introduction, the "direct" optimization technique is selected to improve the design. The crux of the problem in this method is to determine the sign of the sensitivity of the objective function to the variation of a design variable. The conventional "brute-force" approach can be extremely expensive for design problems with a large number of design parameters requiring as many or more complete solutions at each optimization step as the number of design variables. Recently, however, substantial progress has been made in determination of sensitivity of the objective function to the design variables using

both the continuum and discrete approaches.[12-17] In these new approaches, the direction of design variable change is determined from analytical/quasi-analytical expressions as opposed to the complete solutions of governing equations. Solution of an algebraic set of equations to determine the complete set of sensitivity derivatives is required for a given optimization step. Consequently, for problems where the number of design variables is large, as in realistic wing aerodynamic optimization problems, and multi-discipline optimization problems, this will prove to be an efficient approach. In the present research effort, the discrete quasi-analytical approach to computing the gradient of objective function with respect to variations in design variables, is being coupled with the flow equations.

Only the Euler subset of the complete flow equations in 2-D is being considered at this time. The Euler equations in generalized curvilinear coordinates are given by:

$$\partial_\tau Q + \partial_\xi E + \partial_\eta F = 0 \qquad (1).$$

In Eq. (1), τ is time, ξ and η are the curvilinear coordinates, Q is the vector of conservative variables, and E and F, are the inviscid flux vectors.
In vector form

$$Q = J^{-1}\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ e\end{bmatrix} \qquad E = J^{-1}\begin{bmatrix}\rho U\\ \rho u U + \xi_x p\\ \rho v U + \xi_y p\\ (e+p)U - \xi_t p\end{bmatrix}$$

More details of these terms can be found elsewhere.[5] To obtain an implicit solution to Eq. (1), E and F are linearized in time resulting in 4×4 flux jacobian matrices $A = \frac{\partial E}{\partial Q}$ and $B = \frac{\partial F}{\partial Q}$ and the following equation:

$$\Big[I + hA_\xi + hB_\eta\Big]\Delta Q = -h\Big\{E_\xi + F_\eta\Big\} \qquad (2)$$

In ARC3D's[5] approach to solving the above equation, E_ξ, F_η, A_ξ, and B_η are central differenced. A combination of 2nd and/or 4th order smoothing is added to both sides of the equation to make the numerical scheme stable. The left hand side is approximated by either 1) factorization into two directionally independent matrices with 2nd order smoothing added to each direction yielding block tri-diagonal matrix systems, or, 2) diagonalization of the factorized matrices with combination 2nd/4th order smoothing yielding scalar penta-diagonal matrix systems. At steady state, ΔQ vanishes, and we are left with:

$$R = \partial_\xi E + \partial_\eta F - k\frac{\partial^4 Q}{\partial \xi^4} - k\frac{\partial^4 Q}{\partial \eta^4} = 0 \qquad (3a)$$

in the interior, and

$$R = Q_b - f(Q_o, X, \beta) = 0 \qquad (3b)$$

at the boundaries, where k is constant and β is a vector of design variables. Q_b and Q_o are values of Q at the boundaries and the interior respectively. X is the vector of physical co-ordinates of the grid. Compared to the ARC3D implementation, here, 2nd

order smoothing and spectral radius scaling have been neglected thus resulting in only the fourth order smoothing terms in Eq. (3a).

For the optimization problem, in general, the real goal is to find the sensitivity of the objective function to the variation of design variables. The objective function may be the value of vector Q, or a function there of, at a certain location in the flow field, or a quantity that is obtained by integration of Q on a boundary, e.g. C_L. To determine the sensitivity of the Euler equations to the variation in design variables, R in general may be expressed as:

$$R(Q(X,\beta),\quad X(\beta),\quad \beta) = 0 \tag{4}$$

Similarly, the jth objective function C_j can expressed as:

$$C_j = C_j(Q(\beta),\quad X(\beta),\quad \beta) \tag{5}$$

The goal, however, is to find sensitivity of the j_{th} objective function C_j to the design variables β_k. Differentiating the above eq. yields:

$$\frac{dC_j}{d\beta_k} = \left[\frac{\partial C_j}{\partial Q}\right]\left\{\frac{\partial Q}{\partial \beta_k}\right\} + \left[\frac{\partial C_j}{\partial X}\right]\left\{\frac{\partial X}{\partial \beta_k}\right\} + \left\{\frac{\partial C_j}{\partial \beta_k}\right\} \tag{6}$$

Eq. (6) comprises of five terms. Evaluation of the first and third terms, viz. the single row matrices $\left[\frac{\partial C_j}{\partial Q}\right]$ and $\left[\frac{\partial C_j}{\partial X}\right]$ depends upon the selection of the objective function C_j. For example, let the objective function be expressed as

$$C_j = \sum \xi_x p$$

Then, p, the pressure can be easily represented in terms of components of vector Q which can then easily be differentiated w.r.t. Q to determine $\left[\frac{\partial C_j}{\partial Q}\right]$. Similarly, ξ_x can be expressed in terms of spatial coordinates so that C_j can be differentiated w.r.t. X to determine $\left[\frac{\partial C_j}{\partial X}\right]$. The fourth term, $\left\{\frac{\partial X}{\partial \beta_k}\right\}$, is the grid sensitivity term. For problems, where design parameters are either not geometric, or are such that an existing grid simply needs to be translated/rotated, the grid sensitivity term is easy to compute. However, for problems, where the design parameters are such that new surfaces have to be defined, this term is evaluated by a brute finite-difference approach. Assumption is made that selected objective functions will not have an explicit dependence on β_k thus negating the need to evaluate the 5th term i.e. $\left\{\frac{\partial C_j}{\partial \beta_k}\right\}$.

To evaluate the fourth term, $\left\{\frac{\partial Q}{\partial \beta_k}\right\}$ in eq.(6), eq.(4) is differentiated with respect to β_k, the k_{th} design variable, following the procedure outlined in Refs. 14-16, to yield:

$$\left[\frac{\partial R}{\partial Q}\right]\left\{\frac{\partial Q}{\partial \beta_k}\right\} + \left[\frac{\partial R}{\partial X}\right]\left\{\frac{\partial X}{\partial \beta_k}\right\} + \left\{\frac{\partial R}{\partial \beta_k}\right\} = 0 \tag{7}$$

The assumption that chosen design variables β_k are such that R does not depend upon β_k leads to the following system of equations that needs to be solved to determine $\left\{\frac{\partial Q}{\partial \beta_k}\right\}$:

$$\left[\frac{\partial R}{\partial Q}\right]\left\{\frac{\partial Q}{\partial \beta_k}\right\} = -\left[\frac{\partial R}{\partial X}\right]\left\{\frac{\partial X}{\partial \beta_k}\right\} \tag{8}$$

In the interior, $\left[\frac{\partial R}{\partial Q}\right]$ comprises of A_ξ, B_η and Frechet derivatives of smoothing terms with respect to Q. At the boundaries $\left[\frac{\partial R}{\partial Q}\right]$ is evaluated using Frechet derivatives. For example, Frechet derivative of a fixed inflow boundary would result in contribution of identity matrices to the main diagonal for rows corresponding to the boundary. Similarly, an extrapolation boundary condition would result in contribution of identity matrices to the main diagonal for rows corresponding to the boundary. In addition the same rows would also get contribution of negative identity matrices at locations where solution is extrapolated from. The second order two-point central differenced stencil of the flux jacobians and the five-point central differenced stencil of the smoothing term result in a $(4 \times jmax \times kmax) \times (4 \times jmax \times kmax)$ matrix system of nine diagonals.

Evaluation of $\left[\frac{\partial R}{\partial X}\right]$ is made simple by writing the flux jacobians E and F in the transformed curvilinear co-ordinates in terms of the flux jacobians in the original physical co-ordinates and then using Frechet derivatives. The resulting matrix for $\left[\frac{\partial R}{\partial X}\right]$ is $(4 \times jmax \times kmax) \times (2 \times jmax \times kmax)$ with four diagonals where each diagonal entity is a block 4×2 matrix.

In 2-D, there are advantages to solving eq. (8) directly, however, in 3-D, only iterative approaches will be promising. Delta formulation of this equation, where approximations similar to those made for the flow solver[5] can be used, is being looked into. Till date, in the current work, eq. (8) is solved using Cray library routines SSGETRF and SSGETRS, where the first routine factorizes the matrix and the second routine solves it using pivoted Guassian elimination technique. Only non-zero entries are stored to solve the above system. It should be pointed out that the system may be solved explicitly as well such that an inverse does not need to be computed leading to substantially less storage requirements. $\left\{\frac{\partial Q}{\partial \beta_k}\right\}$ thus found is substituted back in eq. (6) to yield the sensitivity gradient.

The sensitivity gradient can also be found by formulating the adjoint equation. Here, we add λR to eq. (6), where λ is the Lagrange multiplier. Then λ is to be determined such that

$$\left[\frac{\partial C_j}{\partial Q} + {\lambda_j}^T \frac{\partial R}{\partial Q}\right]\left\{\frac{\partial Q}{\partial \beta_k}\right\} = 0 \tag{9}$$

thus negating the need to determine $\left\{\frac{\partial Q}{\partial \beta_k}\right\}$. Eq. 9 expressed as:

$$\left[\frac{\partial R}{\partial Q}\right]^T \lambda_j = -\left[\frac{\partial C_j}{\partial Q}\right]^T \tag{10}$$

is the adjoint equation and has to be solved for λ. Components required to be evaluated for this equation are same as for the direct method before. Selection of which method should be used to solve a particular problem depends upon the number of constraints versus the number of design variables. Once λ is known, the sensitivity derivative is evaluated by computing:

$$\frac{dC_j}{d\beta_k} = \left[\frac{\partial C_j}{\partial X} + \lambda_j{}^T \frac{\partial R}{\partial X}\right]\left\{\frac{\partial X}{\partial \beta_k}\right\} = 0 \tag{11}$$

RESULTS AND STATUS

The derivation of the adjoint and sensitivity equations, and implementation in OVERFLOW is complete. The approach to testing this optimization implementation is based on the fact that the direct "brute-force" optimization method works for practical design problems but is limited because of the expense incurred when the number of design variables is large.[19] If a faster method of determining sensitivity of the objective function to the design variables can be implemented, then the existing optimization procedures can be followed by simply replacing the evaluation of the sensitivity derivative with the current approach.

In an effort to test the sensitivity approach, a simplified problem of design of a quasi-1-d convergent-divergent wind-tunnel for a given throat velocity was devised. For the Euler equations, in the absence of any vorticity generation, this reduces to the solution of a very simple equation, viz. $VA = constant$, where V is velocity and A is cross-section area. NPSOL[18], a nonlinear optimizer, based on a sequential quadratic programming algorithm, was used to drive the optimization procedure. With velocity as the objective function and area as the design variable, NPSOL could be used to compute sensitivity of velocity to area change by evaluating $VA = constant$, viz. the "brute-force" method. Also, one could supply the sensitivity to NPSOL by differentiating $VA = constant$. Both approaches were tried in this rather simplistic example to yield the same solution and to show the benefit of supplying the analytical expression for the gradient.

In the next step, a shock-free 2-d convergent-divergent wind-tunnel example is chosen to test the accuracy and efficiency of the current approach. Thrust at exit is chosen as the objective function. Consequently, corresponding to Eq. (3b), $\left[\frac{\partial R}{\partial Q}\right]$ and $\left[\frac{\partial R}{\partial X}\right]$ are derived for inflow, outflow, symmetry, and tangency boundaries using Frechet derivatives. A two-dimensional rectangular grid was generated and flow was computed with the assumption of symmetry on the top surface as only the bottom half of the wind-tunnel is being solved. The lower boundary of the grid was allowed to move by placing a sine-bump[20] at the throat to perturb the baseline rectangular design. Sensitivity gradients computed using the direct and the adjoint approaches match identically, however, vary slightly when compared with the "brute-force" method. To resolve the differences, the left- and right-hand sides of eq.(8) were computed both numerically and analytically. The numerical evaluation of the right-hand side of eq. (8), viz. the vector obtained by carrying out matrix vector multiplication of $\left[\frac{\partial R}{\partial X}\right]\left\{\frac{\partial X}{\partial \beta_k}\right\}$ is done as follows. Compute R_1 to convergence on a grid corresponding to β_1. Compute R_2 by carrying out a restart solution on a new grid corresponding to β_2 (perturbed value of design variable β_1) for one step. $(R_2 - R_1)/(\beta_2 - \beta_1)$ then is the

numerical value of the right-hand side of eq. (8). Similarly, the numerical evaluation of the left-hand side of eq. (8) proceeds by computing Q_1 to convergence on a grid corresponding to β_1 and Q_2 to convergence on a grid corresponding to β_2. R_1 is determined as before. However, R_2 is now determined by supplying Q_2 as the restart and running the solution for one time step. $(R_2 - R_1)/(\beta_2 - \beta_1)$ then is the numerical value of the left-hand side of eq. (8). Figures 1 and 2 show the excellent comparisons obtained between the numerical and analytical evaluations of left- and right-hand sides of eq. (8). In addition notice that solution of Fig. 2 is indeed negative of solution of Fig. 1.

Once, the accuracy of the sensitivity gradients is ascertained, the existing frame-work used by the "brute-force" practitioners can be used as before to carry out the design process. Now, however, when a program such as NPSOL requires evaluation of sensitivity gradient to determine the optimum design, rather than computing flow solution to convergence to determine sensitivity gradient, one calls the new routine to determine the same.

The work carried out so far in this project indicates that this approach is much more efficient compared to the "brute-force" method for 2-D problems. Based on the flowsolver experience, it seems probable that iterative techniques similar to those used in the flowsolver can be used in 3-D for the optimization problem as well. However, work needs to be carried out in that area.

Compared to the Analytical representation of the adjoint equation as opposed to the discrete representation as used here, following points are noted:

1) Analytical representation may be more accurate by definition.

2) Implementation of boundary conditions, especially where discontinuous conditions exist, may be extremely hard for the analytical approach. Discrete representation on the other hand allows for evaluation of $\left[\frac{\partial R}{\partial Q}\right]$ at boundaries using Frechet derivatives of discrete boundary conditions in a straight-forward manner.

3) Either the analytical or the discrete approach can be carried out using implicit or explicit techniques. In the aerodynamic optimization literature, analytical approach used by Jameson et. al. (13) has been carried out by using explicit techniques where as the discrete approach used by others has always been carried out using implicit techniques. With in the discrete area, advantages of the explicit approach need to be explored. Note that when an implicit flowsolver has been used in conjunction with the adjoint problem, one has the advantage of having already computed some of the $\left[\frac{\partial R}{\partial Q}\right]$ terms as required by the time linearizations of the governing equations.

A number of other issues need to be explored to understand the pros and cons of the new approaches. For example, is accurate evaluation of sensitivity gradient necessary at each optimization iteration step? Or, could one reach the final design by carrying out semi-converged solutions of optimization iterations? Is the final design path dependent in that case? For the discrete implementation of the adjoint method, does the adjoint equation need to be based exactly on the governing equation, or could some assumptions be made to simplify, for example, the smoothing terms? It is hoped that other researchers in the near future will explore these issues and provide answers so that a concrete assessment can be made of the viability and pros and cons of the various approaches. Finally, feasibility of using ADIFOR (21) should be checked for computing derivative of the complete smoothing

terms with respect to Q.

REFERENCES

1 Buning, P.G. and Chan, W.M., "OVERFLOW/ F3D User's Manual, Version 1.5," NASA/ARC, Nov. 1990.

2 Jesperson, D.C., and Levit, C.,"A Multiple-Grid Navier-Stokes Code for the Connection Machine CM-2," Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, Norfolk, VA, 1993.

3 Ryan, J.S., and Weeratunga, S., "Parallel Computation of 3-D Navier-Stokes Flowfields for Supersonic Vehicles," AIAA Paper 93-0064, January 1993.

4 Smith, M.H., and Pallis, J.M.,"Medusa – An Overset Grid Flow Solver for Network-Based Parallel Computer Systems," AIAA Paper 93-3312, July 1993.

5 Pulliam, T.H. and Chaussee, D.S., "A Diagonal Form of an Implicit Approximate-Factorization Algorithm," Journal of Computational Physics, Vol. 39, February 1981, pp. 347-363.

6 Beam, R. and Warming, R., "An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form," Journal of Computational Physics, Vol. 22, 1976, pp. 87-110.

7 Steger, J. L., Ying, S. X., and Schiff, L. B., "A Partially Flux Split Algorithm for Numerical Simulation of Compressible Inviscid and Viscous Flow," Workshop on Computational Fluid Dynamics, Institute of Non-Linear Sciences, University of California, Davis, CA, 1986.

8 Benek, J.A., Buning, P.G., and Steger, J.L., "A 3-D Chimera Grid Embedding Technique," AIAA Paper 85-1523, July 1985.

9 Barszcz, E., Weeratunga, S., and Meakin, R., "Dynamic Overset Grid Communication On Distributed Memory Parallel Processors," AIAA Paper 93-3311, July 1993.

10 Atwood, C.A.,"Computation of a Controlled Store Separation from a Cavity," AIAA Paper 94-0031, January 1994.

11 Smith, M.H., Chawla, K., and Van Dalsem, W.R., "Numerical Simulation of a Complete STOVL Aircraft in Ground Effect," AIAA Paper 93-4880, December 1993.

12 Sobieszczanski-Sobieski, J., "The Case for Aerodynamic Analysis," Sensitivity Analysis in Engineering, NASA CP-2457, 1986, pp. 77-96.

13 Jameson, A., "Aerodynamic Design via Control Theory," Journal of Scientific Computing, Vol. 3, 1988, pp. 233-260.

14 Korivi, V. M., Taylor, A. C., Newman, P. A., Hou, G. W., Jones, H. E., "An Incremental Strategy for Calculating Consistent Discrete CFD Sensitivity Derivatives," NASA TM 104207, 1992.

15 Baysal, O., Eleshaky, M., "Aerodynamic Sensitivity Analysis Methods for the Compressible Euler Equations," Journal of Fluids Engineering, Vol. 113, 1991, pp. 681-688.

16 Taylor, A, C, Hou, G. W., Korivi, V. M., "Discrete Shape Sensitivity Equations for Aerodynamic Problems," AIAA Paper 91-2259, 1991.

17 Reuther, J. and Jameson, A., "Control Theory Based Airfoil Design for Potential Flow and a Finite Volume Discretization," AIAA Paper 94-0499, January 1994.

18 Gill, P. E., Murray, W., Saunders, M., and Wright, M. H., "User's Guide for NPSOL (version 4.0): A Fortran Package for Nonlinear Programming," Systems Optimization Lab Report 86-2, Stanford, CA, 1986.

19 Reuther, J., Van Dam C. P., and Hicks, R., "Subsonic and Transonic Low-Reynolds-Number Airfoils with Reduced Pitching Moments," Journal of Aircraft, Vol. 29, 1992, pp. 297-298.

20 Kennelly, R. A., "Improved Method for Transonic Airfoil Design-by-Optimization," AIAA Paper 83-1864, 1983.

21 Bischof, C., Carle, A., Corliss, G., Griewank, A., and Hovland, P., "ADIFOR: Generating Derivative Codes from FORTRAN Programs," Scientific Programming, Vol. 1, 1992, pp. 1-29.

Comparison of Numerical and Analytical $\left[\frac{\partial R}{\partial X}\right]\left\{\frac{\partial X}{\partial \beta_k}\right\}$

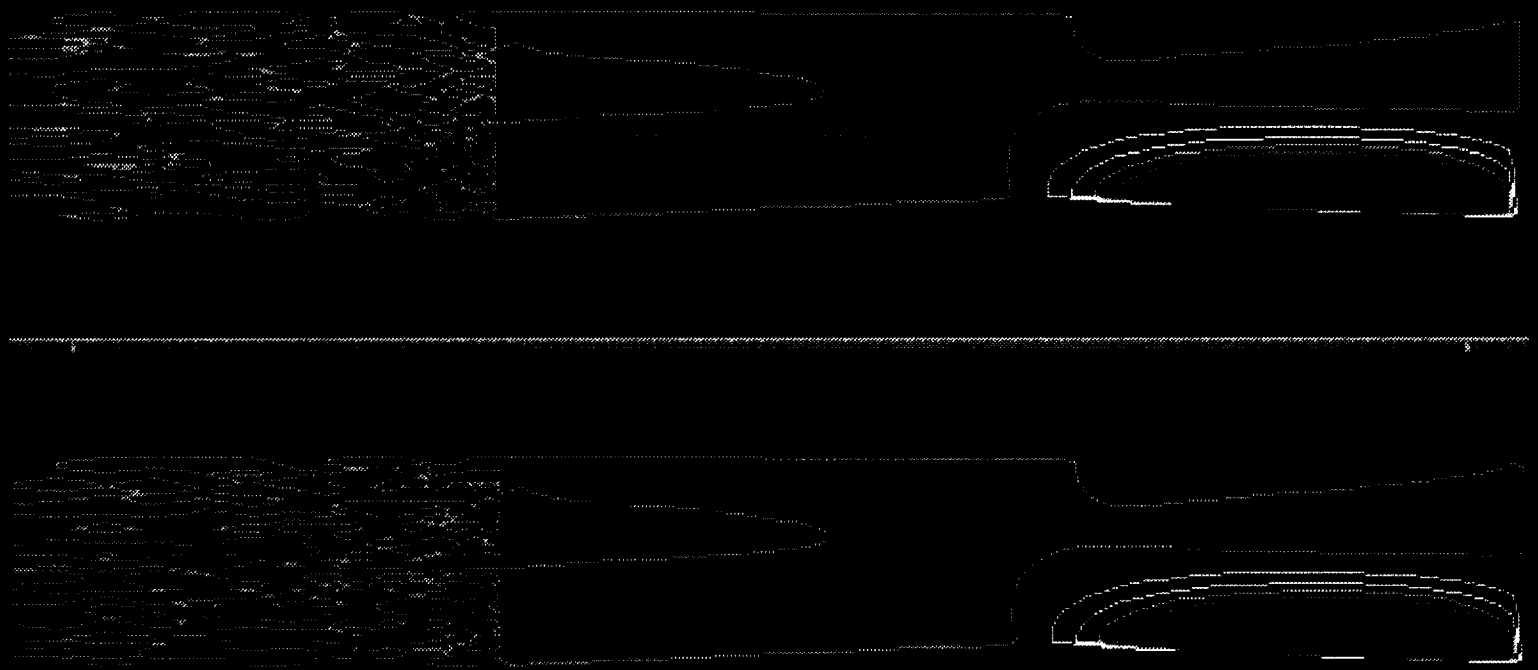

Figure 1

Comparison of Numerical and Analytical $\left[\frac{\partial R}{\partial Q}\right]\left\{\frac{\partial Q}{\partial \beta_k}\right\}$

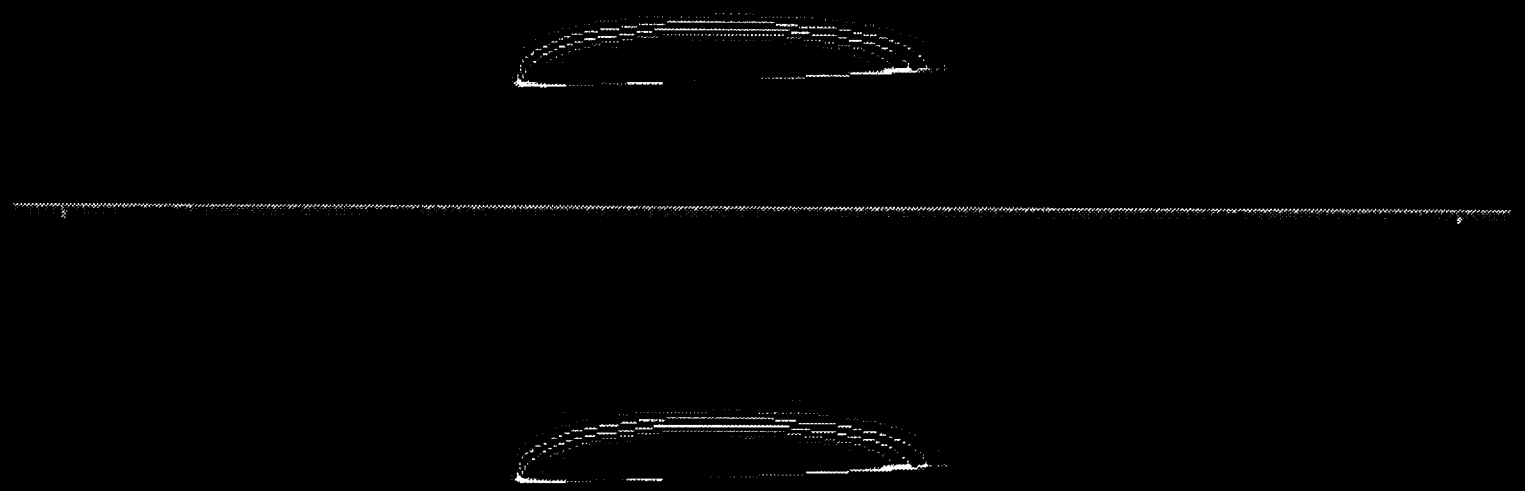

Figure 2

AIAA 95-0573
Overset Grid Applications on Distributed Memory MIMD Computers

Sisira Weeratunga
Computer Sciences Corporation
NASA Ames Research Center

and

Kalpana Chawla
Overset Methods Inc.
NASA Ames Research Center
Moffett Field, CA 94035-1000

33rd Aerospace Sciences Meeting and Exhibit
January 9-12, 1995 / Reno, NV

OVERSET GRID APPLICATIONS ON DISTRIBUTED MEMORY MIMD COMPUTERS

Sisira Weeratunga*
Computer Sciences Corporation
MS T27A-1, NASA Ames Research Center
and
Kalpana Chawla†
Overset Methods, Inc.
MS 258-1, NASA Ames Research Center
Moffett Field, California 94035-1000

Abstract

Analysis of modern aerospace vehicles requires the computation of viscous flowfields about complex 3-D geometries composed of regions with varying spatial resolution requirements. Overset grid methods allow the use of proven structured grid flow solvers to address the twin issues of geometrical complexity and the spatial resolution variation by decomposing the complex physical domain into a collection of overlapping subdomains. This flexibility is accompanied by the need for irregular intergrid boundary data communication among the overlapping component grids. This study investigates one of the strategies for implementing such a static overset grid implicit flow solver on distributed memory, MIMD computers; i.e., the 160 node IBM SP2 and the 208 node Intel Paragon. Performance data for two composite grid configurations characteristic of those encountered in present day aerodynamic analysis are also presented.

Introduction

The complexity of Computational Fluid Dynamics (CFD) simulations attempted at present is very closely related to the sustained CPU performance of the readily available computer resources. Simplified, 2-D flow analysis simulations can be carried out using conventional high performance workstations on a regular basis. However, 3-D unsteady, viscous flow analysis still requires the very best of computing hardware. Most of the current generation of vector supercomputers such as the Cray C-90 and the NEC SX-3 are fully capable of providing the compute resources required for such simulations. However the high cost of such machines and their consequent limited availability have spurred efforts aimed at seeking more cost effective approaches to high performance, numerically-intensive computing. Most of the ongoing efforts in this area are carried out under the umbrella of the national High Performance Computing and Communications Program (HPCCP). One such approach is based on the exploitation of the relatively high degree of concurrency and the spatial data locality inherent in numerical algorithms used for aerodynamic simulations. Under these conditions, distributed memory, multiple instruction, multiple data (DM-MIMD) computers offer excellent long-term prospects for greatly increased computational speed and memory at a cost that may render the 3-D flow analysis around complex geometries on a routine basis more affordable. Among the recent advances in computer hardware technologies that lend credence to such expectations are the advent of mass-produced high-performance Reduced Instruction Set Computing (RISC) microprocessor chips, high density Dynamic Random Access Memory (DRAM) chips and high-speed interconnect networks that are easily scalable to the level of hundreds of nodes. The essential remaining ingredient required for the success of this mode of computing is the development and implementation of underlying numerical algorithms in a manner that is conducive to retaining high parallel efficiencies when the number of processors used range at least in the low hundreds. This often requires a complete top down analysis of the entire numerical scheme in search of

*Research Scientist, Member AIAA.
†Research Scientist, Member AIAA.

exploitable concurrency associated with various algorithmic phases and a complete understanding of the essential data dependencies. This is followed by the design of a parallel implementation strategy that is capable of achieving a near optimal load distribution among all participating computational nodes and simultaneously attempts to minimize the interprocessor communication costs. Such considerations usually require changes in one or more of the following: a) the scheduling of various tasks associated with the underlying numerical scheme, b) the manner in which the associated data is organized and c) the algorithms used to perform certain subtasks. This sometimes leads to radical re-engineering of the existing serial implementations. A further complicating factor in this endeavor is the lack of advanced software development tools for the current generation of DM-MIMD computers comparable to those found on vector supercomputers to aid in the program development effort.

An inescapable fact when computing flowfields around modern aerospace vehicles is the associated geometrical complexity. This is further compounded by the presence of regions with widely varying resolution requirements surrounding such vehicles. A vigorous research effort is currently underway in the CFD community to develop solution adaptive, unstructured grid flow solvers to deal with such geometrical and physical complications. However, their suitability for high Reynolds number flow simulations over complicated geometries is yet to be firmly established. On the other hand, the use of well proven structured grid flow solvers in combination with the overset grid approach [1, 2] has proven to be a viable alternative to the fully unstructured grid approach for simulating high Reynolds number flows around complex configurations.

In the overset grid approach [3], the complex aircraft configuration is first decomposed into a set of components, each with a relatively simple geometry. This is followed by the independent meshing of each such component using logically structured curvilinear meshes. To ensure adequate spatial resolution of the flow field, additional overset grids can be used in critical regions based on a-priori knowledge of the flow field. Finally, these component grids are overlaid to yield a larger composite grid that can be used to compute flow fields around complex configurations. Such an approach gives rise to a locally structured but globally unstructured flow solver.

Overlaying of grids in this manner results in embedding of outer boundaries as well as the solid body regions of one grid within the computational domains of the other grids. As a result, the grid points belonging to the latter grids that lie within an embedded solid body region along with some prescribed neighborhood around it are 'blanked-out', i.e., excluded from the computation. Such points are commonly referred to as hole points and the grid points that lie in the fringes of these blanked-out regions form artificial interior boundaries. They are used to impose the influence of the embedded solid body upon the surrounding component grid. The union of the outer boundaries of the embedded minor grids and the artificial interior boundaries delineated by the blanked-out regions form the inter-grid boundaries of the composite grid system. The values of flow field variables at grid points lying on these inter-grid boundaries need to be obtained through interpolation from the solutions of the other component grids in which they are embedded in.

The interpolation process required to compute values of flow field variables at grid points lying on the inter-grid boundaries serves to communicate the influence of the solution on one grid to those on the other grids. In practice, this intergrid boundary point (IGBP) data interpolation process is carried out at the beginning of each time step of the time marching scheme adapted for the flow solvers used within each component grid and is referred to as the intergrid communication. The intergrid communication scheme seeks the necessary interpolation data from the hexahedral computational cell of the donor grid that contains the IGBP in question and such cells are referred to as the donor cells. Therefore the overset grid approach requires the identification of the following entities in all the component grids: a) hole points, b) IGBP's, c) donor grids and donor cells and d) tri-linear interpolation coefficients. For the test cases considered in this study, we used DCF3D[4] software running on a workstation to accomplish this task as a preprocessing step. It should be noted that this intergrid communication process can have a highly irregular structure depending upon the relative positioning of the component grids. The distribution of the IGBP's within the computational space of each component grid is generally very non-uniform. In addition the corresponding set of donor cells may be distributed among multiple donor grids. Conversely each donor grid may be contributing data to IGBPs belonging to many other component grids. Finally, just as in the case of the IGBP's, the donor cells within a component grid can have a highly non-homogeneous distribution with respect to its computational space.

The primary objectives of this study are three fold: a) design of a scalable parallel implementation strategy for the overset grid, implicit flow solvers on DM-MIMD computers when the number of processors range in the hundreds, b) development of intergrid communication data structures and inter-processor communication strategies for its implementation on the DM-MIMD computers and c) validation of the parallel implementation strategy and the assessment of its scalability as well as the overall performance potential through the use of realistic composite grid

configurations. Two DM-MIMD computer testbeds were chosen for this validation and evaluation process, viz. the 160-node IBM SP2 and the 208 node Intel Paragon. Two test problems are selected here for the performance evaluation of the overset grid flow solver. These problems require the solution of the Navier-Stokes equations and the use of multiple overset grid topology. The first test problem is the 4-grid simulation of viscous flow past a delta wing with thrust reverser jets, flying in ground effect (the Powered-Lift configuration). The second test problem is the simulation of viscous flow past the FLAC (Fighter Lift and Control) wing with deflected leading and trailing edge flaps (the High-Lift configuration). This 20-grid setup offers an opportunity to evaluate load balancing issues and the grid partitioning strategies for realistically complex geometries.

In the following sections, the concurrent algorithms for overset grid problems and their parallel implementation strategy is summarized. This is followed by descriptions of the computational test beds and the geometry/boundary conditions of the selected test problems. Then we present the results of our experiments along with some analysis.

Solution of Overset Grid Problems

As a prelude to the development of a parallel implementation strategy, a brief conceptual overview of the generic mathematical algorithms underlying the overset grid flow solvers is presented in this section. It is assumed that within each component grid, the Navier-Stokes equations along with the relevant physical/numerical boundary conditions are discretized using the appropriate spatial and temporal discretization procedures. This in conjunction with the imposition of the intergrid interpolation conditions at the IGBPs results in a system of nonlinear algebraic equations for each component grid that can be represented by the following generalized vector functions:

$$\mathbf{F}_i(\ \mathbf{Q}_1^{n+1}, \mathbf{Q}_2^{n+1}, \ldots, \mathbf{Q}_i^{n+1}, \ldots, \mathbf{Q}_M^{n+1}, \mathbf{Q}_i^n) = 0, (i = 1, 2, \ldots, i, \ldots, M). \quad (1)$$

where $\mathbf{Q}_i^{n+1}$ is the vector of discrete flow field variables belonging to the $i-$th component grid at the time level $(t + \Delta t)$ and M is the total number of component grids involved. It should be noted that $\mathbf{F}_i$ may not be a function of all $\mathbf{Q}_i, (i = 1, 2, \ldots, M)$. The exact functional dependence is determined by the relative positions of the overlapping component grids.

There are a variety of iterative approaches available for the solution of the system of equations given by Eq. (1). The implicit flow solvers used in this study use a non-iterative time marching scheme for its solution. In this approach, the system of equations is linearized about the already known solutions $\mathbf{Q}_i^n, (i = 1, 2, \ldots, M)$. Then the resulting global system of linear algebraic equations are given by:

$$\begin{pmatrix} \mathbf{A}_{1,1}^n & \mathbf{A}_{1,2}^n & \cdots & \mathbf{A}_{1,M}^n \\ \mathbf{A}_{2,1}^n & \mathbf{A}_{2,2}^n & \cdots & \mathbf{A}_{2,M}^n \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{M,1}^n & \mathbf{A}_{M,2}^n & \cdots & \mathbf{A}_{M,M}^n \end{pmatrix} \begin{pmatrix} \Delta\mathbf{Q}_1^n \\ \Delta\mathbf{Q}_2^n \\ \vdots \\ \Delta\mathbf{Q}_M^n \end{pmatrix} = \begin{pmatrix} -\mathbf{F}_1(\mathbf{Q}_1^n, \mathbf{Q}_2^n, \ldots, \mathbf{Q}_M^n) \\ -\mathbf{F}_2(\mathbf{Q}_1^n, \mathbf{Q}_2^n, \ldots, \mathbf{Q}_M^n) \\ \vdots \\ -\mathbf{F}_M(\mathbf{Q}_1^n, \mathbf{Q}_2^n, \ldots, \mathbf{Q}_M^n) \end{pmatrix} \quad (2)$$

where $\mathbf{A}_{i,j}^n = (\partial\mathbf{F}_i/\partial\mathbf{Q}_j^{n+1})(\mathbf{Q}_1^n, \mathbf{Q}_2^n, \ldots, \mathbf{Q}_M^n)$ and $\mathbf{Q}_i^{n+1} = \mathbf{Q}_i^n + \Delta\mathbf{Q}_i^n, (i = 1, 2, \ldots, M)$. The off-diagonal block matrix elements $\mathbf{A}_{i,j}, (i \neq j)$ of the global Jacobian matrix represent the intergrid coupling effects between component grids i and j through the interpolated values at IGBPs. These block matrix elements are themselves sparse matrices with highly irregular structure. Due to the use of tri-linear interpolation for intergrid communication, they generally have a maximum of eight non-zero elements per row. Again, some of these off-diagonal block matrix elements may be null matrices, depending on the relative locations of the component grids. The diagonal block matrix entries $\mathbf{A}_{i,i}^n$ represent the implicit coupling of variables within a component grid, similar to those found in well known uni-grid flow solvers. The correction vector $(\Delta\mathbf{Q}_1^n, \Delta\mathbf{Q}_2^n, \ldots, \Delta\mathbf{Q}_M^n)$ needed to update the flow variables in all component grids is given by the solution to this large, sparse system of linear equations. There are many approaches available for the solution of this system of linear equations and the method selected should be capable of providing a sufficiently accurate solution with a high degree of reliabiiity in addition to being amenable to efficient implementation on DM-MIMD architectures. In the following paragraphs, we conceptualize some of the available algorithm alternatives for the solution of Eq. (2), and discuss the advantages and disadvantages associated with each such alternative.

The obvious first choice is the fully-coupled approach, where the system of equations (2) is directly inverted. While such a direct inversion scheme would lead to an unconditionally stable time marching scheme for the overset grid problem, it would be prohibitively expensive in terms of computer resource requirements (CPU time and memory), for solving problems of practical interest to the computational aeroscience community. In addition, due to the highly irregular sparsity structure of the coefficient matrix, the direct inversion of Eq. (2) would not lend itself to an efficient implementation on DM-MIMD computers. An alternative avenue within the context of the

fully-coupled approach is to seek a solution to Eq. (2) through a matrix-free iterative scheme, which is designed, if feasible, to be significantly more economical both in terms of memory and CPU time requirements and at the same time be more amenable to efficient implementation on DM-MIMD machines. We defer the consideration of such a solution scheme to future efforts. It should be noted that the use of a geometric multigrid approach to solve Eq. (2) has already been reported in the literature[5].

The alternative to the fully-coupled approach to solving of Eq. (2) is the partitioned analysis. In this approach, some of the off-diagonal block matrix entries, which are responsible for the intergrid coupling effects are moved to the right hand side of Eq.(2). This is effected by evaluating their contributions based on the temporally extrapolated approximations to the relevant elements in vectors $Q_i^{n+1}, (i = 1, 2, \ldots, M)$. These predicted values are usually obtained as a suitable linear combination of their values at the previous time levels, n, $(n-1)$ etc. The primary motivation for this approximation is the resulting decoupling across the inter-grid boundaries, of the solution of the large system of equations represented by Eq. (2). Consequently, the solution of Eq. (2) is accomplished by solving a series of smaller sub-systems of linear equations represented by it's diagonal block matrix entries.

There are two commonly used variants to the partitioned analysis approach. If the effect of all the off-diagonal block matrix entries in Eq. (2) are to be approximately represented on its right hand side, based entirely on the extrapolated values to the discrete field variables required during intergrid communication, then the system will be solved through an approach similar to a block-Jacobi scheme with unequal block sizes. If on the other hand, the matrix in Eqn. (2) or some permuted form of it is reduced to a block lower or upper triangular matrix by approximately representing the effects of only some of its off-diagonal block matrix entries on the right hand side through extrapolation in time, then the underlying system is solved by an approach akin to the classical block-Gauss-Siedel(GS) method. In this staggered approach, the effect of some of the off-diagonal block matrix entries are represented on the right hand side using the most recently computed discrete field values, instead through temporal extrapolation. A majority of the currently available uni-processor and shared-memory multiprocessor implementations of the overset grid flow solvers falls into this category. A third approach, which is a hybrid of the above two approaches is also feasible. In this multilevel method, clusters or groups of component grids are formed first. This is followed by the application of block GS like algorithm within the groups and block-Jacobi like algorithm across the groups.

A direct consequence of this partitioned analysis approach to solving the system of equations (2) is it's conditional stability with respect to the time step size Δt. In order to avoid numerically unstable computations, time step size Δt has to be bounded by a value determined by the highest temporal frequency component present in the solution of the overset grid problem. In addition, the severity of the stability restrictions is also likely to depend on the fraction of the IGBPs relative to the total number of grid points and the characteristics of the flow field in regions where the intergrid boundaries are located. For some overset grid problems, these restrictions are likely to prove to be too severe, giving rise to solution schemes that are unconditionally unstable for all practical purposes. Therefore it is assumed that for the class of overset grid problems of interest to this study; a) the transient response is primarily dominated by the relatively low frequency components and b) the component grids are designed such that the placement of intergrid boundaries in critical flow regions are avoided. Consequently, the partitioned analysis approach is likely to prove to be a cost-effective alternative for solving the system of equations (2). As in the case of block-Jacobi vs. block-GS schemes, the restriction placed on the value of Δt due to numerical stability considerations is likely to be more severe in the case of the first variant of the partitioned analysis approach. Such restrictions placed on Δt may be alleviated to some extent through the use of sub-iterations within a time step.

In spite of the above mentioned drawbacks, the partitioned analysis approach can provide several significant computational and software engineering advantages over the fully-coupled approach. Among these are; 1) ability to use proven and independently developed discretization/solution algorithms within each component grid involved, 2) preservation of high degree of software modularity and 3) excellent prospects for realizing scalable parallel implementations on DM-MIMD computers. Furthermore, within the context of the partitioned analysis approach, incorporation of additional coupled disciplines such as controls, thermal analysis etc. can be accommodated relatively easily.

In this paragraph we examine the algorithmic details of the block Jacobi-variant of the partitioned analysis approach for overset grid problems. This variant is represented by the following system involving a block diagonal coefficient matrix:

$$\begin{pmatrix} \mathbf{A}^n_{1,1} & 0 & \dots & 0 \\ 0 & \mathbf{A}^n_{2,2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{A}^n_{M,M} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{Q}^n_1 \\ \Delta\mathbf{Q}^n_2 \\ \vdots \\ \Delta\mathbf{Q}^n_M \end{pmatrix} = \begin{pmatrix} -\mathbf{R}^n_1 \\ -\mathbf{R}^n_2 \\ \vdots \\ -\mathbf{R}^n_M \end{pmatrix} \quad (3)$$

where the right hand side vectors are defined by:

$$\begin{aligned} \mathbf{R}^n_i &= -\mathbf{F}^n_i(\mathbf{Q}^n_1, \mathbf{Q}^n_2, \dots, \mathbf{Q}^n_i, \dots, \mathbf{Q}^n_M) \\ &\quad - \textstyle\sum_{j=1,(j\neq i)}^{M} \mathbf{A}^n_{i,j}\Delta\mathbf{Q}^p_j \\ &= -\mathbf{F}^n_i(\mathbf{Q}^p_1, \mathbf{Q}^p_2, \dots, \mathbf{Q}^n_i, \dots, \mathbf{Q}^p_M) \end{aligned} \quad (4)$$

and $\Delta\mathbf{Q}^p_j = \mathbf{Q}^p_j - \mathbf{Q}^n_j, (j = 1, 2, \dots, M)$. The temporlly extrapolated values of the discrete flow variables required for the intergrid data interpolation process are given by formulae of the type:

$$\mathbf{Q}^p_j = \sum_{k=0}^{(m-1)} \alpha_k \mathbf{Q}^{n-k}_j,$$

and α_k are appropriately chosen constants. Following algorithmic facts are evident from the above analysis: a) all intergrid data dependencies appear only in the right hand side vectors, b) all intergrid data interpolation and communication requirements can be accomplished concurrently and c) all component grid sub-problems can be solved concurrently.

The solution of component grid sub-problems involves the inversion of block matrices $\mathbf{A}^n_{i,i}, (i = 1, 2, \dots, M)$. Since the component grids are logically structured, these matrices are regular sparse, banded matrices[6], with a relatively high bandwidth. Again, the direct inversion is an option but not viable due to reasons cited before. However, there exist several well-proven approximate inversion schemes that have been developed over last two decades within the context of uni-grid computations. Most of these time marching schemes are generally characterized by: a) low memory requirements and b) amenable to efficient implementation on DM-MIMD architectures. It should be noted that the scheme chosen need not be the same for all component grids. The partitioned analysis approach offers the possibility of using different schemes on different grids, if needed. In this study we have chosen to use the diagonalized form of the block Beam-Warming scheme[6, 7] on all component grids.

Parallel Implementation Strategy

In this section, we provide a brief overview of some of the DM-MIMD architectural features that influenced our parallel implementation strategy followed by an abbreviated discussion of some salient features of the implementation. The DM-MIMD implementation of an overset grid flow solver based on explicit update of the intergrid boundary values presents several options. This is primarily due to the MIMD characteristics of the architecture. Here, we provide a discussion of the options available and the factors influencing the choices. For the remainder of this discussion, we assume a DM-MIMD computer with a fixed but moderate to large number of processors and an overset grid problem involving a fixed number of grids. The system software on the current generation of DM-MIMD computers does not allow dynamic processor allocation/de-allocation, once a job is initiated on a fixed subset of it's processors on a space shared basis. Furthermore, in this study, we preclude the possibility of further subdivision of any of the component grids through the introduction of additional inter-grid boundaries. Although such subdivision may facilitate computationally more efficient and a scalable implementation on DM-MIMD architectures, it's impact on the solution integrity, numerical stability and the overall convergence rate is currently not well understood. However, it should be noted that such implementations for both multi-block and overset grid problems already exist[8, 9, 10]. Also, this approach is a subset of the implementation adapted is this study. The computational load associated with each grid, largely a function of the number of grid points, is also assumed to be fixed during the entire simulation. However, we allow for the possibility that differences may exist across component grids in the following: a) time marching scheme used for the advancement of the solution and b) the physical effects included in the simulation.

The time marching schemes used within each component grid possess a certain degree of concurrency that can be exploited through data parallelism. In the block-GS like variant of the overset grid flow solver, each component grid is processed in a predetermined sequence. Consequently, the degree of exploitable concurrency at any given instant is limited to the data parallelism available within the component grid being processed at that instant. This results in a situation where the degree of exploitable concurrency may vary as a function of the elapsed time. Such an implementation of an overset grid, implicit flow solver on the Connection Machine CM-2 has already appeared in the literature[11].

In contrast, the use of the block-Jacobi like variant allows more than one component grid to be processed

concurrently. This permits the exploitation of an additional degree of concurrency, available across all or some fixed subset of the component grids involved. This extra level of parallelism is generally exploited through concurrent processing of multiple component grids on distinct clusters of processors and is commonly referred to as task parallelism. Such an implementation allows the simultaneous exploitation of the task parallelism available across the component grids and the data parallelism available within each component grid. However, in order to ensure that a sufficiently good static load balance exist across all the clusters participating in the solution of the overset grid problem, the following condition need to be satisfied. The fraction of the total number of processors assigned to each cluster should be be directly proportional to the fraction of total computational load associated with the component grids being processed on that cluster.

In the following discussion, we summarize some of the advantages and disadvantages associated with each of the approaches. A more detailed description can be found in Ref. 12. The two factors that have a dominant influence over this issue are: a) the variation of the degree of exploitable concurrency and b) the variation of computational load, across the set of component grids. Both of these factors are primarily influenced by: a) the type of mathematical model, b) the nature of the computational algorithms, and c) the number of grid points, used within each component grid. The secondary factors are: a) the nature of the physical/numerical boundary conditions applied, and b) the number of IGBP's and donor cells associated with each grid. In most realistic overset grid problems, there is a significant variation of both the computational load and the available degree of concurrency among the participating component grids. In some cases, this variation could be as much as an order of magnitude or more.

When a fixed number of processors are used to solve an overset grid problem with such a heterogeneous character by processing each component grid in a prescribed sequence, two adverse implications arise. First, in the case of component grids possessing only a reduced degree of concurrency or smaller computational loads, it may not be possible to gainfully utilize all the allocated processors for performing the underlying computational tasks. This would lead to idling of some of the assigned processors. Even when the computational attributes of the component grid are such that all processors can be gainfully utilized, grids with smaller computational loads would incur higher parallel implementation overheads due to reduced task granularity. This would invariably lead to lower parallel efficiency. In addition, one may also be compelled to search for alternative algorithms with higher degree of extractable concurrency that have the potential for being accompanied by higher memory and/or arithmetic overhead as well.

On the contrary, the concurrent computation of either all or a subset of the participating grids on distinct clusters of processors, where the number of processors assigned to each component grid from the fixed pool of processors is decided on the basis of their computational loads and the inherent degree of exploitable parallelism, would likely result in an implementation with minimal idling of processors and higher overall parallel efficiency. This is due to the fact that each individual grid would now be computed using only a fraction of the total available processors, which would invariably lead to higher parallel efficiency compared with the case of processing the same grid on all of the available processors. Also, given the smaller number of processors assigned to individual grids, this approach requires algorithms possessing only a moderate degree of exploitable parallelism.

The secondary factors influencing this choice are; 1) memory requirements for each component grid vs. that available on a fixed number of processors, 2) I/O performance to and from secondary storage devices relative to the sustained computational performance and 3) availability of system software to perform processor-to-processor communication between two processors who are members of two different groups of processors.

A careful consideration of all these factors resulted in our decision to implement the variant of the overset grid approach given by Eqn. (3), at this time. This non-iterative time integration scheme was adopted due to the concurrency it affords across all the participating component grids. In this context, the process of achieving a good load balance across all the processors in the pool assigned to the task is somewhat complicated. Among the factors that hinder our ability are: a) upper bound on the memory available per processor, b) limitations imposed by system software on the number of active processes per processor (currently limited to one), c) each cluster should consist of an integer number of processors, d) additional constraints on the allowable number of processors per cluster that may be imposed by the system architecture and e) restrictions imposed by the mesh partitioning scheme used within a component grid to maintain an acceptable level of parallel efficiency within that component grid. Consequently, the expectation to achieve near-optimal load distribution across the entire pool of processors is unrealistic.

Within the constraints cited above, we adapt the following approach. The pool of processors is assumed to be partitioned into M clusters, where M is the total number of component grids involved. Each such cluster has $P_i, (i = 1, 2, \ldots, M)$ processors and one component grid assigned to it. As a result, we require a minimum of M processors be assigned to the

problem. The actual number required may be higher, due to the limitations on memory available per processor. The total memory available within a cluster should be either equal to or greater than that required by it's component grid solver and the intergrid communication data structures. If the total number of processors in the pool, say P, is sufficiently large, the solution to the following critical path problem can be used to determine the distribution of processors across the M clusters;

$$minimize\ [max\ T_i:\ 1 \le i \le M]\ subject\ to$$

$$\sum_{i=1}^{M} P_i = P : P_i \in I$$

$$\beta_i N_i \le \rho_i P_i$$

where

$$T_i = \eta_i(P_i)\alpha_i \frac{N_i}{P_i}, (i = 1, 2, \ldots, M)$$

Here we have assumed that the cost of intergrid communication is negligible. Also, $N_i, (i = 1, 2, \ldots, M)$ is the number of grid points in the i-th component grid; $\alpha_i, (i = 1, 2, \ldots, M)$ is the normalized work load per grid point per step in the i-th component grid; $\eta_i, (i = 1, 2, \ldots, M)$ is the parallel efficiency of the i-th component grid; $\beta_i, (i = 1, 2, \ldots, M)$ is the memory required in words per grid point for the flow solver used in i-th component grid and $\rho_i, (i = 1, 2, \ldots, M)$ is the memory in words available per processor in the i-th cluster. In general, α_i is a function of the mathematical model and the numerical algorithms used. For a given type of flow solver, η_i is generally a implementation dependent non-monotonic function of P_i for fixed N_i.In addition, it may also depend on the type of boundary conditions used and the ratio of grid dimensions. The critical path problem is somewhat more complicated when more than one process can be assigned to a processor.

It is also possible to use a multi-level approach to load balancing. First, groups of component grids are formed such that the total computational load withing each group is approximately equal. Then processor clusters are assigned to each group along the approach described above. This is followed by formation of sub-clusters within each cluster of processors, again following the same approach.

In this study, we have not attempted to solve the above critical path problem, but instead have sought only to obtain a leading order approximation through the following:

$$P_i = \alpha_i N_i (P / \sum_{i=1}^{M} \alpha_i N_i); i = 1, 2, \ldots, M-1.$$

$$P_M = P - \sum_{i=1}^{M-1} P_i$$

Note that this involves assuming $\eta_i = 1, (i = 1, 2, \ldots, M)$. Once the cluster sizes are known for a given P, it is possible to use a greedy algorithm to adjust the values of $P_i, (i = 1, 2, \ldots, M)$ as P changes.

At the beginning of each new time step, the time marching process starts by simultaneously interpolating and exchanging the temporally extrapolated field data necessary for updating the values at the IGBPs of all component grids. During this data exchange, a subset of processors within each of the M group of processors are participating in inter-processor communications. This is then followed by the simultaneous and independent computation of the updated values of the flow fields in all the participating component grids.

The data parallel, Single Program Multiple Data (SPMD) implementations of the component grid flow solvers can be carried out independently of one another. This is a direct consequence of the software modularity afforded by the overset grid approach. Due to the MIMD nature of the architecture, each cluster is capable of executing the same SPMD implementation of the solver for different component grids. The diagonalized form of the Beam-Warming algorithm found in OVERFLOW[13] formed the basis for the data parallel implementation of the flow solver used in this study. The details of this DM-MIMD implementation can be found in Ref. (14). The version used in this study is based on uni-partitioning of the grid and uses grouped, two-way pipelined Gausssian elimination for the solution of non-periodic pentadiagonal systems. The periodic pentadiagonal systems are solved using fully balanced, one-way pipelined Gaussian elimination algorithm[15].

The only task that requires close interaction and coordination among different clusters of processors from the software implementation point of view is that associated with the tri-linear interpolation and exchange of the flow field data at the IGBPs. This data interpolation and exchange has to be carried out in the context of grid partitioning dictated by the independent, data parallel implementations of the component grid flow solvers within the clusters assigned to them. In addition, this phase of the computation should exploit as much concurrency as possible with minimum of synchronization barriers to maintain the overall efficiency of the parallel implementation. This was accomplished through the use of a distributed, concurrent implementation of the interpolation algorithms and a loosely synchronous approach to interprocessor data communication involving a highly irregular communication pattern. This intergrid boundary data exchange process required the design of a new distributed data structure for the processing of IGBPs and their associated donor cells. Also a procedure for initializing the highly irregular interprocessor communication pattern among processors belonging to differ-

ent groups was required. Further details with regard to the distributed data structures used and the procedure followed for establishing the inter-group communication pattern will be available in a future publication.

The Test bed Architectures

Here we provide a brief description of the two DM-MIMD test bed architectures used in this study: the 160 node IBM SP2 and the 208 node Intel Paragon. The SP2 is essentially a set of IBM RS6000/590 workstations connected by a high performance switch with a topology of an omega network (a hierarchy of cross bar switches). The RS6000/590 workstation is based on 66.7 MHz. POWER2 multi-chip RISC processor, with a theoretical peak performance of approximately 250 Mflops on 64-bit data. Each node has a 128 Kbyte data cache and a minimum of 128 Mbytes of main memory and 2 Gbytes of disk space. ¿From the application software perspective, the interconnection network is capable of moving data between SP2 nodes with a latency of approximately 45 microseconds and a bandwidth of about 34 Mbytes/sec.

The Intel Paragon is composed of 208 compute nodes, each consisting of two 50 MHz. i860/XP RISC microprocessor chips connected by a two-dimensional mesh interconnection network. The theoretical peak performance of the i860/XP is 75 Mflops on 64-bit data, with a 16 Kbyte instruction and data caches. Each compute node has 32 Mbytes of memory with approximately 22 Mbytes available to user application programs. One of the i860/XP chips on each node is used solely for inter-processor communication. The interconnection network moves data with a latency of 120 microseconds and a bandwidth of about 35 Mbytes/sec.

Both test beds currently support message passing programming paradigm. The implementation in this study was layered on the message passing libraries based on the Message-Passing Interface (MPI) standard[16]. The MPI provides a common interface for development of portable message passing applications on distributed memory concurrent computers and networks of workstations. The MPI functionality includes point-to-point and collective communication routines as well as support for process groups and communication contexts. The latter two features are essential for the development of modular applications which incorporate simultaneous use of data and task parallelism.

The Test Problems

In order to evaluate the computational performance of the DM-MIMD implementation of the overset grid flow solver, two realistic test problems that typify the aerodynamic analysis computations carried out using the overset grid approach were used. The first problem, referred to as the Powered-Lift configuration simulates the viscous flow past a delta wing with two jet's in ground effect. The simulation of viscous flow over a Fighter Lift and Control (FLAC) wing with deflected leading and trailing edge flaps was used as the second problem. This is referred to as the High-Lift configuration in the subsequent discussion. In order to assess the impact of the block Jacobi like variant of the partitioned analysis approach on the unsteady flow computations, two test problems were considered. The first one is the viscous flow past a circular cylinder while the Powered-Lift configuration was used as the other.

The Powered-Lift Configuration

The computational setup of this configuration[17] consists of a 60° delta planform in a free stream of Mach number 0.064, at 6.4° angle of attack (α), with two choked jets located at the inboard trailing edge. The jet flow is at a nozzle pressure ratio (NPR) of 1.8 and is exhausted at an angle of 45° to the chord of the delta planform. The delta wing is located at a height h above the ground plane, such that $h/b = 0.25$, where b is the wing span. The flow field symmetry about the $y = 0$ plane passing through the center line of the delta wing is assumed. This geometry is discretized by generating 1) a C-H grid around the delta wing, 2) a cylindrical grid around the jet pipe, 3) a jet trajectory conforming grid, and embedding the three grids in 4) a Cartesian ground plane grid (Figs. 1, 2, 3). This results in a composite grid of nearly 1 million grid points. The interconnectivity among the four grids and the hole points created as a result of overlaying is determined using DCF3D software[4]. The composite grid was found to contain a total of approximately 40,000 IGBP's distributed among it's components. On the delta wing and pipe surface, no slip condition is used along with extrapolation of density and pressure values from those at one grid point above the solid walls. On the ground surface, a moving wall condition is used to reflect the experimental conditions. The free stream conditions are applied on the inflow boundary and the three side surfaces of the background grid, while extrapolation is used at the outflow boundary. At the jet exit plane, the velocity and pressure ratio are set to those corresponding to the experimental conditions.

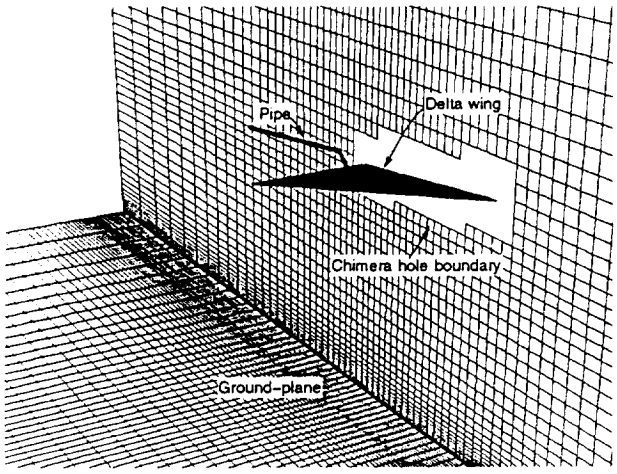

Fig. 1: The ground-plane grid.

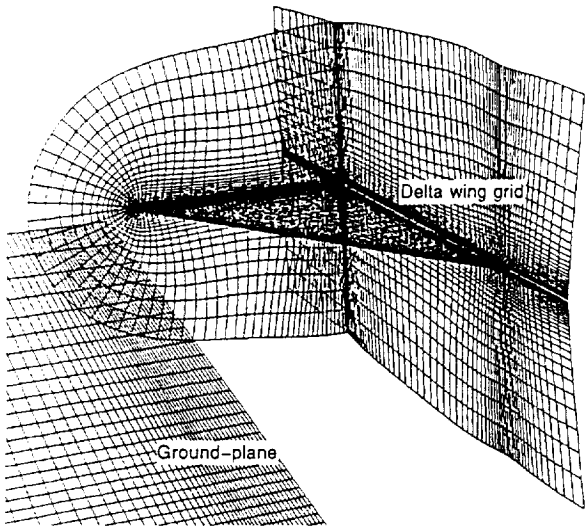

Fig. 2: The delta wing grid.

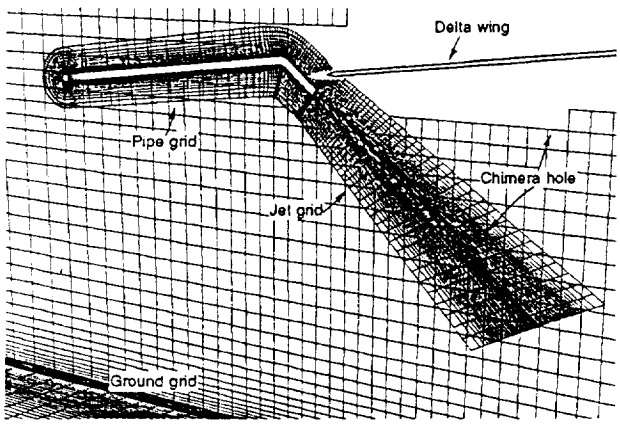

Fig. 3: The pipe and the jet grids.

The High-Lift Configuration

The High-Lift configuration is represented by a computational model to simulate the flow over the FLAC wing with deflected leading and trailing edge flaps at a Mach number of 0.18 and a Reynolds number of 2.5 million. The gridding strategy was designed to facilitate the computation of flow field at different flap deflection angles, with a minimum amount of regridding. This component grid strategy is depicted in Fig. 4. Grids terminate at boundaries between fixed and moving parts, viz. flaps. The flaps are gridded as separate components, so that the flap rotation about a hinge line on the lower surface of the wing can be accomplished by rotating the flap grid. As the flaps rotate, they slide down the upper surface of the wing. The flap tips and the internal wing tips that get exposed when the flaps deflect need to be discretized in a manner that can resolve viscous effects (Fig. 5). Due to the presence of airfoil sections with extremely sharp leading and trailing edges, these tips and the wing tip (Fig. 6) can not be discretized using standard wrap-around grids. However, they lend themselves easily to the use of polar grids placed on the tips, with the singularity located at the leading or the trailing edge itself (Fig. 7). Volume grids are then grown from these polar grids to cover the air gaps. The extremely thin and sharp wing tip is discretized using three grids; two polar grids for the leading and trailing edge areas and one cartesian grid for the region not covered by the polar grids (Fig. 6). The entire wing-flap system is gridded using 18 grids. These 18 component

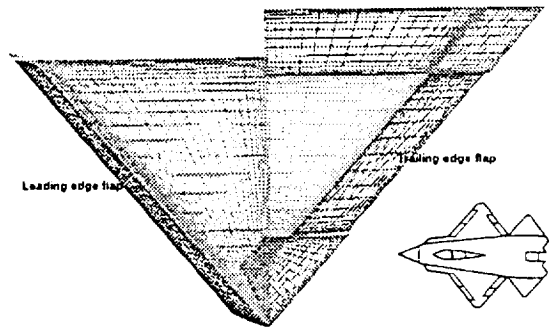

Fig. 4: The FLAC wing grids.

grids are embedded in a fine global grid covering the entire set to facilitate good inter-grid information exchange. The fine global grid is in turn embedded in a coarse grid spanning large extent of the physical space around the wing, resulting in a composite of 20 overset grids with a total of approximately 1.5 million grid points. Again the interconnectivity among the 18 FLAC grids and the 2 global grids as well as the location of their respective hole points are determined using DCF3D. The composite grid was found to contain 140,000 IGBPs. On all the FLAC wing surfaces, a no-slip condition similar to one used in Powered-

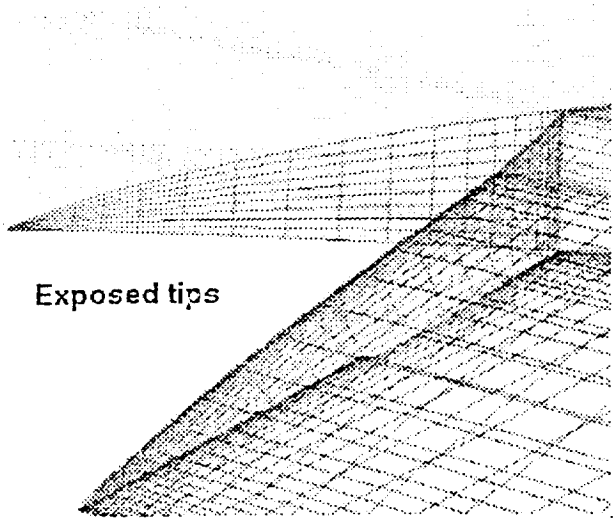

Fig. 5: The FLAC wing exposed tips.

Lift configuration is applied. A no-slip wall condition is used on the yz-plane at the wing root to simulate the wind-tunnel wall found in the experimental set up. The free stream conditions are applied at the inflow boundary and on the remaining side surface of the global coarse grid, while extrapolation is used at it's outflow boundary.

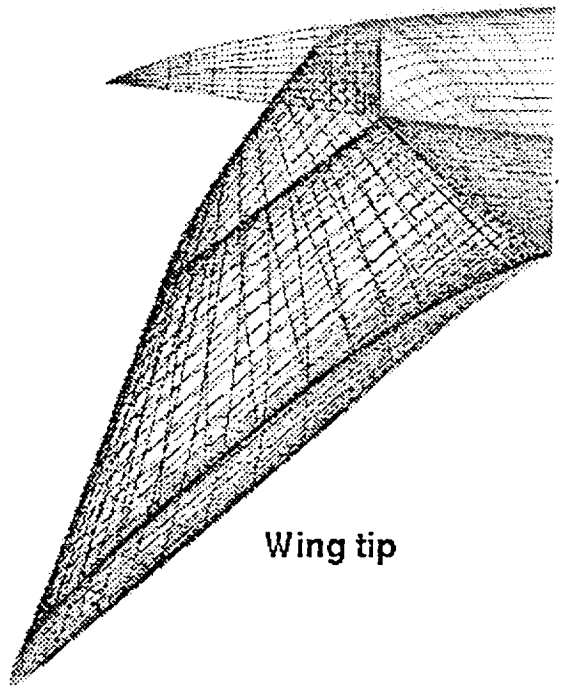

Fig. 6: The FLAC wing tip.

Results and Discussion

In this section, we describe some computational performance data and unsteady flow computation results obtained using the static overset grid flow solver implemented on DM-MIMD computers. All performance data reported are for 64-bit arithmetic and implementations based entirely on FORTRAN. In implementing a general purpose code such as the one used here, issues such as software modularity, extensibility and maintainability cannot be entirely overlooked in favor of computational performance. The code extensibility and maintainability issues precludes the developers from using excessive amount of "creative" programming procedures and instead rely mostly on optimizing compilers for achieving good performance on modern RISC architectures. The general purpose nature of the code and the attendant software modularity requirements are often in conflict with programming techniques that enhance temporal and spatial locality of data. The locality of data is of utmost importance to achieving high performance on hierarchical memory architectures such as those found in modern RISC processors. We have employed a balanced approach, whereby the essential software modularity was retained while attempting to maximize data locality within that context. Another issue confronting the application software developers on RISC architectures

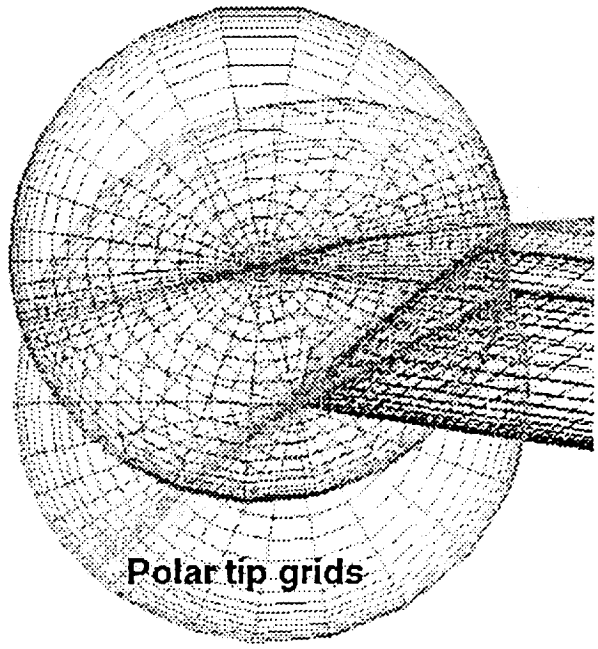

Fig. 7: The polar tip grids.

is the relatively high cost of floating point divide operations and intrinsic functions such as SQRT, when compared with the cost of similar operations on traditional vector supercomputers. When implementing an algorithm such as ARC3D[7], it is always possible to reduce the total floating point operation count in general and the number of floating point divide and SQRT operations in particular, by resorting to the storage of intermediate data arrays. Very often, this is accompanied by increased memory requirements, measured in terms of 64-bit words per grid point. This in turn reduces the size of the problem that can be computed on a fixed number of processors. Therefore judicious compromises are necessary when employing memory vs. time optimizations. The degree to which all of the above tradeoffs are exercised have a profound impact on the observed performance of the implementation.

Fig. 8 shows the performance of the single grid, implicit Navier-Stokes solver on the IBM SP2, for 5 different problem sizes and processor counts varying from 1 to 128. The problem sizes vary between approximately 1/4 million to 4 million grid points. The performance is measured in terms of time per step. The corresponding figure for a single processor of Cray C-90 is approximately 7 microsec/pt/step. Although for a fixed grid size, the time/step continues to decrease as the number of processors is increased, the parallel efficiency also decreases. For the smallest problem size used, the efficiency is only 40% when 128 processors are deployed. This is primarily due to the inevitable increase in parallel implementation overheads as a fraction of the total time, when the number of processors is increased for a fixed problem size. A discussion of various types of parallel implementation overheads and their impact on performance can be found in Ref. (18).

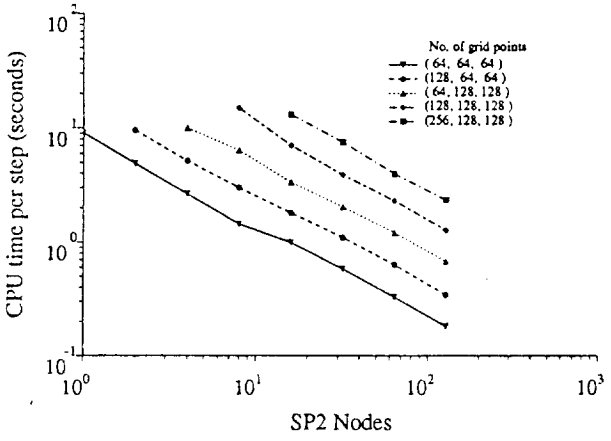

Fig. 8: Single zone performance on the SP2.

Tables (1) and (2) show the total number of processors used and the sizes of the processor clusters assigned to each component grid for a series of runs on the IBM SP2, for the Powered-Lift and the High-Lift configurations respectively. The tables also show the sizes of each component grid and the mesh partitioning used for the data parallel implementation of the implicit flow solver within each component grid. This mesh partitioning was chosen to provide best possible performance of the flow solver, given the size of the processor cluster assigned to a particular component grid. For the implicit flow solver implementation used in this study, it was found that for a fixed number of processors, the best performance was generally realized when following guidelines were adhered to during the mesh partitioning process: a) each coordinate direction is partitioned into two segments before any direction is assigned more than two partitions, b) ratio of partitions in each coordinate direction matches as closely as possible to the ratio of grid points in each coordinate direction. For a given number of processors, it is not always possible to follow both guidelines exactly, but the first guideline always takes precedence over the second whenever possible.

As a result of this mesh partitioning strategy, no attempt was made to equi-distribute the IGBPs or donor cells among all participating processors. Tables (3-6) show the distribution of IGBPs and donor cells among processors assigned to different grids of the Powered-Lift configuration, where the number of processors used is 28. As can be seen from the tables, the distribution of IGBPs and donor cells among processors is highly non-uniform. It is quite evident from the tables that a given processor can act in one of the following four modes during intergrid communication process: a) only as a recepient of donor cell data from other processors, b) only as a provider of donor cell data to other processors, c) a combination of (a) and (b), d) none of the above. As a result, the processor load during intergrid communication process can be highly unbalanced.

Consider the case when processors are acting in mode (a). The number of grids and processors supplying donor cell data can vary widely among the active set of processors. In addition, the number of donor cells supplied by each donor processor can also have a large variation. Now consider the case when processors are acting in mode (b). Again the number of grids and associated processors receiving donor cell data from a donor processor can vary widely across the active set of processors. Also, the number of donor cells supplied to each of the recepient processors can also be very different. As a consequence, during the intergrid communication phase, the number and length of messages received as well as the messages sent by a processor can have a large variation across the active set. In addition, the sources of the incoming messages and the destinations of the outgoing messages is widely dispersed across the entire active processor set. This results in a highly irregular interprocessor communication pattern with a wide variation in processor

load.

As can be seen from Table (1), Powered-Lift configuration is an example of an overset grid problem with: a) a relatively small number of component grids and b) the sizes of the largest and smallest grids differ only by a factor of two. As a result, it is possible to realize reasonably good static load balance among component grid flow solvers, even with only a moderately sized pool of processors. On the otherhand, the High-Lift configuration is an example of an overset grid problem with: a) a moderate number of component grids and b) the sizes of the largest and the smallest grids differ by a factor as much as 15. Consequently, it is quite difficult to realize good static load balance among the component grid flow solvers without a relatively large pool of processors. Not all entries in Tables (1) and (2) represent cases where the component grid solver loads are in balance across clusters of processors. The first entry in both tables represent the smallest pool of processors that can be used to solve the problems. It is 6 and 22 for the Powered-Lift and High-Lift configurations respectively. Due to the wide disparity in the grid sizes, the High-Lift configuration needs a minimum of 105 processors to achieve a good static load equi-distribution across all component grids. Some of the other entries in the tables represent assignment of processors to the clusters based on a greedy algorithm, i.e., the cluster with the heaviest load at a given time getting the most number of additional processors as the size of the pool assigned to the problem is increased. Although this approach does not guarantee a good load balance across the entire pool of processors, it ensures that the additional processors are put to best possible use.

Fig. 9 shows the performance of the overset grid flow solver on the IBM SP2 for Powered-Lift and High-Lift configurations, as the size of the pool of processors assigned to the problems is increased. Similar data for the Intel Paragon is shown in Fig. 10. Although most of the cases depicted in these figures are not anywhere near a balanced load state, the data indicates continued reduction in time required to complete the task as the number of processors assigned to the task is increased. However, this does not imply that all the processors are optimally utilized. In all cases, the time per step is determined by the slowest component grid. Load imbalances result in idling of processors belonging to other component grids to varying degrees.

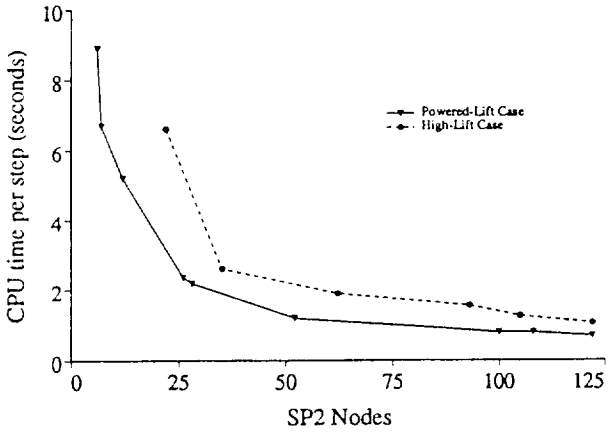

Fig. 9: Performance on the SP2 for Powered- and High-Lift configurations.

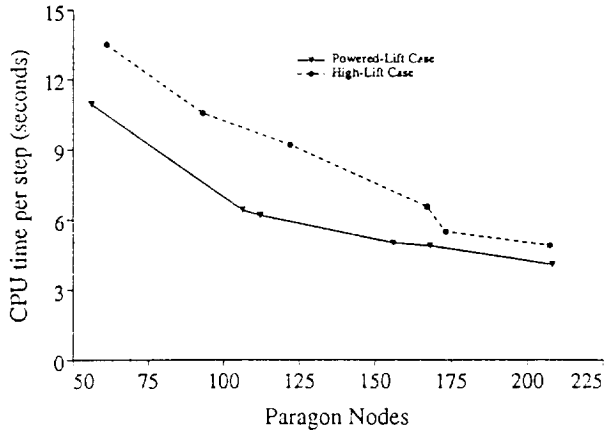

Fig. 10: Performance on the Paragon for Powered- and High-Lift configurations.

Table (7) shows the time per step required for the intergrid data communication process for the Powered-Lift configuration. The size and distribution of the pool of processors used is same as those in Table (1). This time was measured by introducing a synchronizing barrier across the entire pool of processors before the intergrid communication process was initiated to eliminate any processor idle time being included under the intergrid communication costs. The data indicates that the cost of intergrid communication itself is always less than 4% of the total time per step, irrespective of the size of the processor pool. Similar results are obtained for the High-Lift configuration, even though the number of component grids is 20. This validates our assumption that the cost of intergrid communication is negligible in spite of the highly non-uniform load distribution encountered during this phase of the computation. It also attests to the efficacy of the approach used for accomplishing the intergrid communications. Table (7) also indicates the maximum idle time for any processor in the pool of processors assigned to the problem. This provides an indication of the worst possible load imbalance that exists across the entire set of active processors.

In this section, we discuss the results of the preliminary experiments carried out to investigate the effect of block-Jacobi like variant of partitioned analysis approach implemented in this study, on the computation of unsteady flow fields. Fig. 11 shows the time trace

of the lift force acting on a circular cylinder obtained using the DM-MIMD version of the overset grid solver and the Cray C-90 implementation of a similar solver [13]. Three overset grids were used for this computation. The amplitude and frequency of the lift histories show good agreement, indicating that: a) the non-dimensional time step size of 0.125 used for this computation and b) locations of the component grid overlap regions, are such that the use of block-Jacobi variant does not lead to any deleterious effects on the solution. It should be noted that this is a problem with only one dominant, relatively low frequency component in the solution and therefore does not pose serious challenges to the partitioned analysis approach.

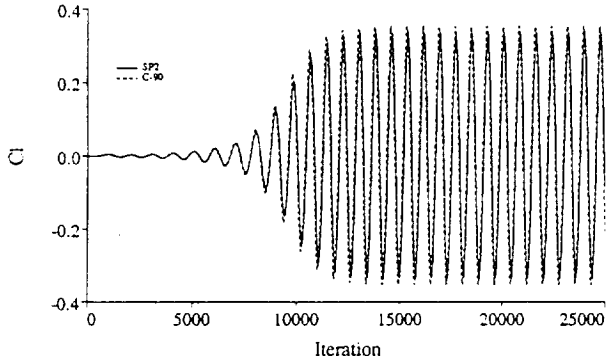

Fig. 11: Lift history for the circular cylinder.

Next, Fig. 12 compares the lift histories of the delta wing in the Powered-Lift configuration, obtained through use of above two implementations. Initially they show good agreement, but as the flow fields develop, some discrepancies between the two traces begin to emerge. Our speculation is that once the jet impinges the ground plane, it is likely that pressure waves are generated that bounce back and forth between the underside of the delta wing and the ground plane, resulting in highly non-linear interactions. Such interactions appear to produce components with frequencies high enough to produce differences in the solutions obtained through the two approaches. In order to examine what effect the time step size would have on the solution, we repeated the experiment twice, each time reducing the time step size by a factor of two. The lift histories obtained are shown in Figs. 13, 14. Both approaches show some differences and they do not follow similar trends. Although no firm conclusions can be drawn from this preliminary investigation, it indicates that the use of block-Jacobi approach can lead to discrepancies at least in some unsteady flow computations.

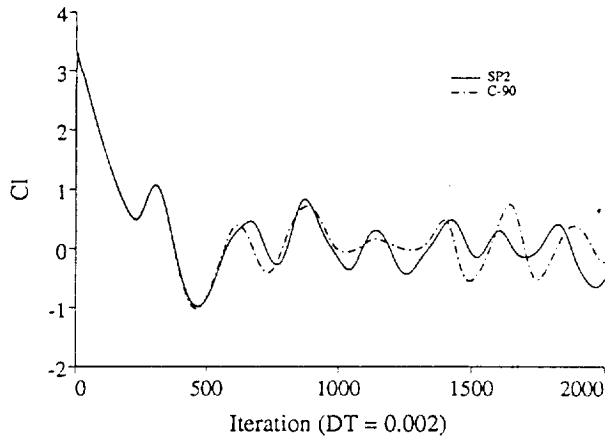

Fig. 12: Delta wing lift history on C-90 and SP2.

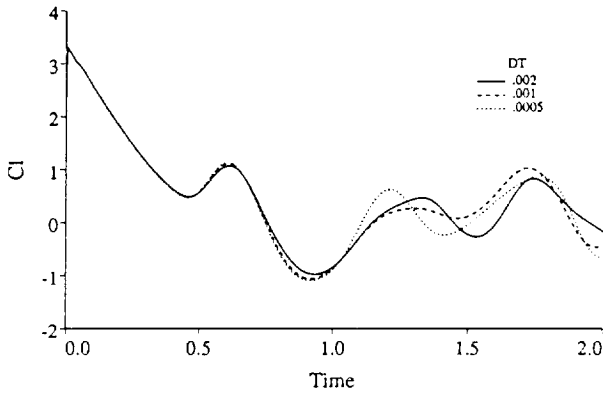

Fig. 13: Delta wing lift history on SP2.

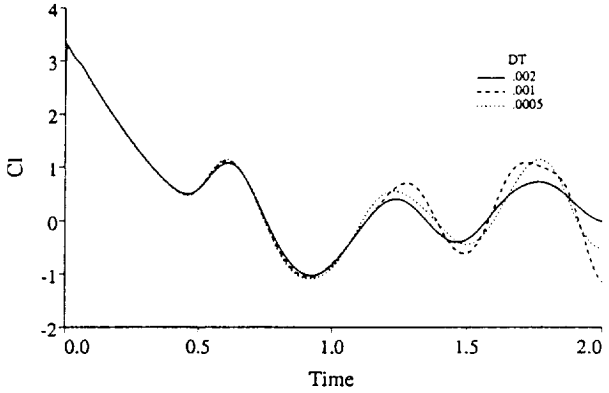

Fig. 14: Delta wing lift history on C-90.

Conclusions

We have successfully implemented an implicit overset grid flow solver on DM-MIMD architectures based on partitioned analysis approach. The implementation facilitates the simultaneous exploitation of data parallelism available within each component grid and task parallelism available across the composite of overset grids. This has the potential to enhance the scalability of the implementation, especially for problems with widely disparate component grid sizes. As a result of using the MPI standard, the implementation was shown to be completely portable across two DM-MIMD architectures, the IBM SP2 and the Intel Paragon. The software architecture adapted for the implementation allows complete modularity and the possibility of deploying different flow solvers on different component grids, if necessary.

Current restrictions imposed by the system software prevents the assignment of more than one process to a processor for time shared execution. This is seen as a major hindrance to accomplishing the following: a) a good static load balance across the component grids and b) solution of overset grid problems with a large number of disparately sized component grids on a moderate number of processors. In spite of this difficulty, we have been able to demonstrate reductions in total time per time step on two realistic overset grid Navier-Stokes computations with the increasing size of the pool of processors assigned to the problems. The cost of intergrid communications appears to be negligible for the two test configurations used. The failure to realize good static load balance with certain processor counts leads to significant idling of some of the processors assigned to the task. This along with parallel efficiency losses within each component grid flow solver are the major factors limiting the parallel efficiency of the overset grid flow solver.

For simulations involving unsteady flow computations, further studies are needed to gain a better understanding of the impact of using block-Jacobi like variant of the partitioned analysis approach. Future efforts may also be warranted in developing static load balancing software to determine the optimum number and sizes of processor clusters and the assignment of component grids to the clusters, given the size of the pool of processors available to the task. The opportunity also exists, when only a steady state solution of the overset grid problem is required, to explore the possibility of allowing component grid solvers to, proceed through the time marching process in a completely asynchronous manner. Although this may alleviate some load balancing problems, potential exist for the appearance of numerical stability problems.

References

[1] Atwood, C.A. and Van Dalsem, W.R.,"Flowfield Simulation about the SOFIA Airborne Observatory," AIAA Paper 92-0656, AIAA 30th Aerospace Sciences Meeting, Reno, NV, January 1992.

[2] Smith, M., Chawla, K., and Van Dalsem, W., "Numerical Simulation of a Complete Aircraft in Ground Effect," AIAA-91-3293, AIAA 9th Applied Aerodynamics Conference, Baltimore, MD, Sept 23-25, 1991.

[3] Steger, J.L. and Benek, J. A., "On the use of Composite Grid Schemes in Computational Aerodynamics", Compt. Methods Appl. Mech. Engrg, 64 (1987), 301-320.

[4] Meakin, R.L., "Overset Grid Methods for Aerodynamic Simulation of Bodies in Relative Motion," 8th Aircraft/Stores Compatibility Symposium, Oct. 1990.

[5] Henshaw, W.D. and Chesshire, G., "Multigrid on Composite Meshes", SIAM J. Sci. Stat. Comput. 8, No. 6, 914 (1987).

[6] Pulliam, T.H., "Implicit Solution Methods in Computational Fluid Dynamics", Applied Numerical Mathematics, 2 (1986), 441-474.

[7] Pulliam, T. H. and Chaussee, D. S., "A Diagonal Form of an Implicit Approximate Factorization Algorithm", Journal of Computational Physics 39, pp. 347-363, 1981.

[8] Smith, M. and Pallis, J., "MEDUSA: An Overset Grid Flow Solver for Network Based Parallel Computer Systems", AIAA-93-3312-CP, 11th AIAA Computational Fluid Dynamics Conference, July 6-9, 1993.

[9] Sawley, M.L. and Tegner, J.K., "A Data Parallel Approach to Multiblock Flow Computations", Intl. J. of Num. Meth. in Fluids, 19, 707-712, 1994.

[10] Chien, Y.P., Ecer, A., Akay, H.U., Carpenter, F. and Blech, R.A., "Dynamic Load Balancing on a Network of Workstations for Solving Computational Fluid Dynamics Problems", Compt. Methods Appl. Mech. Engrg, 119, 17-33, 1994.

[11] Jespersen, D. and Levit, C., "A Multiple-Grid Navier-Stokes Code for the Connection Machine CM-2", in Proc. of 6th SIAM Conf. on Parallel Processing for Scientific Computing, 1, 3-7, 1993.

[12] Barszcz, E., Weeratunga, S.K. and Pramono, E.,"A Model for Executing Multidisciplinary and Multizonal Programs", RNR Report 93-009, NAS Applied Research Branch, NASA Ames Research Center, Moffett Field, Ca 94035, 1993.

[13] Buning, P.G. and Chan, W.M., "OVERFLOW/ F3D User's Manual, Version 1.5", NASA/ARC, Nov. 1990.

[14] Ryan, J.S., and Weeratunga, S., "Parallel Computation of 3-D Navier-Stokes Flowfields for Supersonic Vehicles", AIAA Paper 93-0064, January 1993.

[15] Eidson, T.M. and Erlebacher, G., "Implementation of a Fully-Balanced, Periodic Tridiagonal solver on a Parallel Distributed Memory Architecture", ICASE Report No. 94-37, May 1994.

[16] "MPI: A Message-Passing Interface Standard", The Intl. J. of Supercomputer Applications and High Performance Computing, 8, No. 3/4, Fall/Winter 1994.

[17] Chawla, K., Van Dalsem, W. R., and Rao, K. V., "Simulation of a Delta Wing with Two Jets in Ground Effect," Computing Systems in Engineering, **1**, pp. 483-494, 1990.

[18] Naik, N.H., Naik, V.K. and Nicoules, M., "Parallelization of a Class of Implicit Finite Difference Schemes in Computational Fluid Dynamics", Intl. J. of High Speed Computing, 5, No. 1, (1993) 1-50.

Procs. #	Grid 1 (70,56,70)	Grid 2 (83,81,47)	Grid 3 (60,71,52)	Grid 4 (69,71,35)
6	(2,1,1) = 2	(2,1,1) = 2	(1,1,1) = 1	(1,1,1) = 1
7	(1,1,2) = 2	(1,2,1) = 2	(1,2,1) = 1	(1,1,1) = 1
12	(2,1,2) = 4	(2,2,1) = 4	(1,2,1) = 2	(1,2,1) = 2
26	(2,2,2) = 8	(2,2,2) = 8	(2,3,1) = 6	(2,2,1) = 4
28	(2,2,2) = 8	(2,2,2) = 8	(2,2,2) = 8	(2,2,1) = 4
52	(2,2,4) = 16	(2,4,2) = 16	(2,3,2) = 12	(2,2,2) = 8
100	(2,2,7) = 28	(4,4,2) = 32	(3,4,2) = 24	(2,4,2) = 16
108	(3,2,5) = 30	(4,3,3) = 36	(3,4,2) = 24	(3,3,2) = 18
122	(4,2,4) = 32	(5,4,2) = 40	(3,5,2) = 30	(2,5,2) = 20

Table 1: Grid partitioning for the Powered-Lift configuration on the SP2.

Grid #	Grid size	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
1	(62,62,62)	(1,1,2) = 2	(1,2,2) = 4	(2,2,2) = 8	(2,2,4) = 16	(2,2,4) =16	(2,2,4) = 16
2	(62,62,62)	(1,1,2) = 2	(1,2,2) = 4	(2,2,2) = 8	(2,2,4) = 16	(2,2,4) =16	(2,2,2) = 8
3	(99,38,30)	(1,1,1) = 1	(1,1,2) = 2	(2,1,2) = 4	(2,2,2) = 8	(2,2,2) = 8	(2,2,2) = 8
4	(49,75,30)	(1,1,1) = 1	(1,1,2) = 2	(2,3,1) = 6	(2,2,2) = 8	(2,2,2) = 8	(2,2,2) = 8
5	(99,38,30)	(1,1,1) = 1	(1,1,2) = 2	(2,2,1) = 4	(2,2,2) = 8	(2,2,2) = 8	(2,2,2) = 8
6	(49,57,31)	(1,1,1) = 1	(1,1,2) = 2	(2,2,1) = 4	(2,2,1) = 4	(2,3,1) = 6	(2,2,2) = 8
7	(79,49,33)	(1,1,1) = 1	(1,2,2) = 4	(3,2,1) = 6	(2,2,2) = 8	(2,2,2) = 8	(4,2,2) = 16
8	(36,68,40)	(1,1,1) = 1	(1,1,2) = 2	(1,2,1) = 2	(2,2,2) = 8	(1,2,3) = 6	(2,2,2) = 8
9	(36,57,30)	(1,1,1) = 1	(1,1,1) = 1	(1,3,1) = 3	(1,2,1) = 2	(1,2,2) = 4	(2,2,1) = 4
10	(36,68,30)	(1,1,1) = 1	(1,2,1) = 2	(1,2,1) = 2	(2,2,1) = 4	(1,3,2) = 6	(2,2,2) = 8
11	(26,57,30)	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,2,1) = 1	(1,2,2) = 4	(1,2,2) = 4
12	(10,32,50)	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,2) = 2
13	(14,32,50)	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,2) = 2
14	(11,32,50)	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,2) = 2
15	(24,55,20)	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,2,1) = 2	(1,2,1) = 2
16	(24,55,20)	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,2,1) = 2	(1,2,1) = 2
17	(24,55,20)	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,2,1) = 2	(1,2,1) = 2
18	(24,55,20)	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,2,1) = 2	(1,2,1) = 2
19	(24,55,20)	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,2,1) = 2	(1,2,1) = 2
20	(24,55,20)	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,1,1) = 1	(1,2,1) = 2	(1,2,1) = 2
	Total	22	35	62	93	105	122

Table 2: Grid partitioning for the high-lift configuration on the Paragon.

ZONE 1								
Proc. ID.	1	2	3	4	5	6	7	8
IGBP	271	360	11	91	405	379	21	98
Donor Zones	1	3	1	1	1	2	1	1
Donor Procs	1	10	1	2	3	8	2	5
Max D.C./D.P.	271	286	11	88	330	259	14	78
Min D.C./D.P.	271	1	11	3	2	1	7	1
Avg D.C./D.P.	271	36	11	46	135	48	11	20
IGBP served	1592	8632	267	2031	4730	1524	725	869
Zones served	2	3	1	1	1	2	1	1
Procs served	4	12	2	4	4	4	4	1
Max IGBP/Proc	1377	1286	233	965	2088	1211	487	869
Min IGBP/Proc	2	2	34	226	746	25	5	869
Avg IGBP/Proc	398	720	134	508	1183	381	182	869

Table 3: Powered-Lift Zone 1 grid-communication details (D.C. = Donor Cells, D.P. Donor Processors).

ZONE 2								
Proc. ID.	1	2	3	4	5	6	7	8
IGBP	960	1272	936	1274	2600	2818	2559	2837
Donor Zones	1	2	1	2	1	2	1	3
Donor Procs	1	6	1	4	3	9	4	11
Max D.C./D.P.	960	491	936	528	2088	1211	1377	1286
Min D.C./D.P.	960	24	936	164	25	30	203	2
Avg D.C./D.P.	960	212	936	319	867	314	640	258
IGBP served	2	778	0	1223	394	2596	424	4853
Zones served	1	3	0	3	1	3	2	3
Procs served	1	8	0	9	4	8	9	11
Max IGBP/Proc	2	248	0	348	330	680	271	868
Min IGBP/Proc	2	3	0	3	6	78	1	6
Avg IGBP/Proc	2	98	0	136	99	325	48	442

Table 4: Powered-Lift Zone 2 grid-communication details.

ZONE 3								
Proc. ID.	1	2	3	4	5	6	7	8
IGBP	935	288	909	272	1377	2225	1246	2100
Donor Zones	1	1	1	1	2	3	2	3
Donor Procs	1	1	1	1	4	6	4	6
Max D.C./D.P.	935	288	909	272	970	903	872	868
Min D.C./D.P.	935	288	909	272	15	6	16	2
Avg D.C./D.P.	935	288	909	272	345	371	312	350
IGBP served	2	720	1	680	53	1027	74	926
Zones served	1	1	1	1	2	2	2	2
Procs served	1	1	1	1	2	2	2	2
Max IGBP/Proc	2	720	1	680	44	913	65	820
Min IGBP/Proc	2	720	1	680	9	114	9	106
Avg IGBP/Proc	2	720	1	680	27	514	37	463

Table 5: Powered-Lift Zone 3 grid-communication details.

ZONE 4				
Proc. ID.	1	2	3	4
IGBP	3448	2448	3219	2330
Donor Zones	2	2	2	2
Donor Procs	6	6	6	6
Max D.C./D.P.	913	1286	820	1188
Min D.C./D.P.	225	28	248	32
Avg D.C./D.P.	575	408	537	389
IGBP served	1667	164	1533	202
Zones served	3	2	3	2
Procs served	7	4	7	4
Max IGBP/Proc	903	104	813	143
Min IGBP/Proc	2	11	1	9
Avg IGBP/Proc	239	41	219	51

Table 6: Powered-Lift Zone 4 grid-communication details.

Proc. No.	6	7	12	26	52	100	108	122
Total time/ step	8.899	6.690	5.198	2.358	1.201	0.800	0.761	0.663
Min. solver petime/step	5.480	5.340	2.770	1.592	1.185	0.738	0.715	0.595
Max. I.G.C. time/step	0.058	0.055	0.053	0.026	0.027	0.014	0.013	0.020
Max. Idle time/step	3.600	1.353	2.626	0.766	0.015	0.062	0.046	0.071

Table 7: Intergrid communication (I.G.C.) and idle time for the Powered-Lift configuration for various no. of processors on the SP2.