Overset Methods, Inc.
A Nonprofit Public Benefit Corporation

PROGRESS 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

Overset Grid Applications on Distributed Memory MIMD Computers

Principal Investigator: Dr. Kalpana Chawla

June 30, 1994

OVERSET GRID APPLICATIONS ON DISTRIBUTED MEMORY MIMD COMPUTERS

Kalpana Chawla
Research Scientist, Overset Methods, Inc.
MS 258-1, NASA Ames Research Center

and

Sisira Weeratunga
Research Scientist, Computer Sciences Corporation
MS T27A-1, NASA Ames Research Center
Moffett Field, California 94035-1000

An extended abstract submitted for presentation at the
33rd AIAA Aerospace Sciences Meeting and Exhibit
Reno, NV
January 9-12, 1995

Abstract

Analysis of modern aerospace vehicles requires the computation of 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 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 communication among the overlapping component grids. This study investigates a strategy for implementing such a static overset grid implicit flow solver on distributed memory, MIMD computers; i.e., the 128 node Intel iPSC/860 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

Justifying Parallelization:

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. The most prominent among a number of such computational strategies being currently

investigated under the umbrella of the national High Performance Computing and Communications Program (HPCCP) is the one 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 of complex shapes 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 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 inter-processor 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 some tasks. 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.

Justifying Overset Grids:

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. Although a vigorous research effort is currently underway in the CFD community to develop unstructured grid flow solvers to deal with such geometrical complications, 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.

Overset Grid Background:

In the overset grid approach, 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 as well as 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.

Inter-Grid Communication:

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) IBGP's, c) donor cells and d) tri-linear interpolation coefficients. For the test cases considered in this study, we used DCF3D[3] 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-uniform distribution with respect to its computational space.

Objectives:

The 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 128-node Intel iPSC/860 and the 208 node Intel Paragon. Each node of the iPSC/860 is equipped with 8 Mbytes of memory as opposed to 32 Mbytes on the Paragon. Two test problems are selected here for the evaluation of the overset grid flow solver. These problems require the solution of the Navier-Stokes equations and the use of multi-zone overset grid topology. The first test problem is the Navier-Stokes simulation of flow past a delta wing with thrust reverser jets, flying in ground effect (the V/STOL configuration). CPU performance is compared for solution of this 4-grid problem on the Cray YMP/C-90, the Intel iPSC/860, and the Intel Paragon. In addition, the parallel I/O performance of the Intel is evaluated to determine if the frequent writing of solution required for the detailed analysis of this unsteady flow problem can keep up with the sustained computational rate of the parallel computer. Also, several strategies used for improving the parallel I/O performance are discussed. The second test problem is the Navier-Stokes simulation of flow past the FLAC (Fighter Lift and Control) wing with deflected leading and trailing edge flaps (the High-Lift configuration). The 20-grid setup of this problem offers an opportunity to evaluate load balancing issues and the grid partitioning strategies for realistically complex geometries.

In the following sections, the parallel implementation strategy is summarized and the geometry of the two selected problems is described along with the boundary conditions. Component grid partitioning strategies based on 1) the aspect ratio of each grid, and 2) minimization of intergrid communication load imbalance associated with each processor, are explored. The CPU and parallel I/O performance are compared for the selected test problems on the Cray YMP/C-90 and the DM-MIMD computers.

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:

$$\mathbf{F}_i(\mathbf{Q}_1^{n+1}, \mathbf{Q}_2^{n+1}, \ldots, \mathbf{Q}_i^{n+1}, \ldots, \mathbf{Q}_N^{n+1}, \mathbf{Q}_i^n) = 0, \ (i = 1, 2, \ldots, i, \ldots, N). \tag{1}$$

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 N is the total number of component grids involved. It should be noted that $\mathbf{F}_i$ may not be a

function of all $\mathbf{Q}_i, (i = 1, 2, \ldots, N)$. The exact functional dependence is determined by the relative overlapping positions of the component grids.

Implicit Approach:

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 are linearized about the already known solutions $\mathbf{Q}_i^n, (i = 1, 2, \ldots, N)$. Then the resulting global system of linear algebraic equations are given by:

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

where $\mathbf{A}_{i,j}^n = (\partial\mathbf{F}_i/\partial\mathbf{Q}_j^{n+1})(\mathbf{Q}_1^n, \mathbf{Q}_2^n, \ldots, \mathbf{Q}_N^n)$ and $\mathbf{Q}_i^{n+1} = \mathbf{Q}_i^n + \Delta\mathbf{Q}_i^n, (i = 1, 2, \ldots, N)$. 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 $(\Delta\mathbf{Q}_1^n, \Delta\mathbf{Q}_2^n, \ldots, \Delta\mathbf{Q}_N^n)$ 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 reliability 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 some of 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.

Partitioned Analysis Approach:

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) by evaluating their contributions based on the temporally extrapolated approximations to the relevant elements in vectors $\mathbf{Q}_i^{n+1}, (i = 1, 2, \ldots, N)$. 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), into solution of a series of smaller sub-systems of linear equations represented by its 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 system will be solved through an approach similar to a block-Jacobi scheme. 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-SOR 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 serial implementations of the overset grid flow solvers falls into this category.

A direct consequence of this partitioned analysis approach to solving the system of equations (2) is its 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, the transient response is primarily dominated by the relatively low frequency components and that 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-SOR schemes, the restriction placed on the value of Δt is likely to be more severe in the case of the first variant of the partitioned analysis approach. The restrictions placed on Δt through numerical stability requirements may be alleviated to some degree 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 efficient parallel implementation 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.

Block-Jacobi Approach:

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:

$$\begin{pmatrix} \mathbf{A}^n_{1,1} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^n_{2,2} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{A}^n_{N,N} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{Q}^n_1 \\ \Delta\mathbf{Q}^n_2 \\ \vdots \\ \Delta\mathbf{Q}^n_N \end{pmatrix} = \begin{pmatrix} -\mathbf{R}^n_1 \\ -\mathbf{R}^n_2 \\ \vdots \\ -\mathbf{R}^n_N \end{pmatrix} \tag{3}$$

where the right hand side vectors are defined by:

$$\mathbf{R}^n_i = -\mathbf{F}^n_i(\mathbf{Q}^n_1, \mathbf{Q}^n_2, \dots, \mathbf{Q}^n_i, \dots, \mathbf{Q}^n_N) - \sum_{j=1,(j\neq i)}^{N} \mathbf{A}^n_{i,j}\Delta\mathbf{Q}^p_j = -\mathbf{F}^n_i(\mathbf{Q}^p_1, \mathbf{Q}^p_2, \dots, \mathbf{Q}^n_i, \dots, \mathbf{Q}^p_N) \tag{4}$$

and $\Delta\mathbf{Q}^p_j = \mathbf{Q}^p_j - \mathbf{Q}^n_j, (j = 1, 2, \dots, N)$. 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.

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. In this section, we provide a brief 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 number of processors and an overset grid problem involving a fixed number of grids. In addition, the computational load associated with each grid, largely a function of the number of grid points associated with it, is also assumed to be fixed during the entire simulation. However, we allow for the possibility of a) time marching scheme used for the advancement of the solution and b) the physical effects included in the simulation, within each component grid to be different. The time marching schemes used within each component grid possess a certain degree of concurrency. Although this inherent degree of concurrency may vary from grid to grid, it can generally be exposed through some form of grid partitioning and is commonly referred to as data parallelism. In the block-SOR like variant of the overset grid flow solver, the solution within each component grid is advanced in time in a predetermined sequence. Consequently, in this approach, the degree of exploitable concurrency at any given time is limited to that available within the component grid being processed at the time. This results in a situation where the degree of exploitable concurrency may vary as a function of the processing time. In contrast, the use of the block-Jacobi like variant allows more than one component grid to be advanced in time 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 more than one component grid at a time 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. In addition, on a MIMD computer, the fraction of the total number of processors assigned to each cluster can be made to be directly proportional to the fraction of total computational load associated with the component grid being processed on that cluster of processors. This is generally sufficient to ensure a reasonably good static load balance across all the clusters of processors participating in the solution of the overset grid problem.

In the following discussion, we summarize some of the advantages and disadvantages associated with each approach. A more detailed description can be found in Ref. 4. The two factors that have the most influence over this issues are the variation of the degree of exploitable concurrency and the associated 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 the nature of the physical/numerical boundary conditions applied, as well as the number of IGBP's and donor cells associated with each grid. In most realistic overset grid problems, primarily due to the underlying geometrical complexity, there is a significant variation of both the computational load and the available degree of concurrency among the participating component grids. In some extreme 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 given 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. In contrast, 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 invariably result in an implementation with no idle processors at any time during the simulation and with 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 according to Amdahl's law would have a higher parallel efficiency compared with the case of processing the same grid on all of the available processors. In addition, the task granularity associated with the implementation would also be higher, resulting in reduced overheads and still higher parallel

efficiencies. 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 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). This non-iterative time integration scheme was adopted due to the concurrency it affords across all the participating component grids. The total number of available processors is first partitioned into N groups, where N is the number of component grids involved. Then, each component grid is assigned to one of the distinct groups of processors. The number of processors assigned to each group as a fraction of the total processor count is directly related to the fraction of estimated computational load associated with the grid assigned to that group of processors. Since it is not possible to assign a fraction of a processor to a group, this method of processor assignment is not likely to produce perfect static load balance in most cases. In addition, on some DM-MIMD computers, the number of processors that can be assigned to a group is subject to some constraints. For example, on the Intel iPSC/860, number of processors in each group need to be a power of two. This can lead to further complications in achieving a good static load balance across the component grids. 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 from each of the N 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 different implicit time integration schemes associated with each component grid can be carried out independent of one another. This is a direct consequence of the software modularity afforded by the overset grid approach. However, when the governing equations and the implicit time integration scheme used are the same for more than one component grid, only one implementation of that type of flow solver is necessary. Due to the MIMD nature of the architecture, each cluster of processors is capable of executing the same SPMD implementation of the solver for different component grids. A detailed description of the data parallel implementation issues pertaining to the numerical algorithm used for the time integration of the flow field can be found in Ref. 5.

DM-MIMD implementation of OVERFLOW[6], an implicit Navier-Stokes solver utilizing the diagonalized form of block Beam-Warming scheme[7] is used to solve the conventional dependent variable vector $[\rho, \rho u, \rho v, \rho w, e]^T$ where ρ is density, u, v, and w are velocity components, and e is the total energy per unit volume. The following tasks are performed in the diagonalized scheme to obtain a new solution after one step:

1 Enforcing the boundary conditions.

2 Formation of the R.H.S. comprising of the 2nd-order central-differenced Euler and viscous terms, and the 2nd/4th order central-differenced smoothing terms.

3 Multiplication of diagonal, 5X5 block similarity transformation matrices.

4 Formation and inversion of the scalar penta-diagonal systems.

5 Update of the solution.

The version of the parallel implementation used for this study is based on the one-way pipelined Gaussian elimination algorithm for the solution of the multiple, independent scalar pentadiagonal systems encountered during the inversion of the approximately factored equations. This was chosen primarily due to the low memory requirements associated with it.

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 interpolation and exchange of 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 of processors 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 requires 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 different 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 the completed paper. The establishment of this inter-group communication pattern as well as the actual exchange of intergrid boundary data was achieved through the use of interpartition communication software libraries developed for this purpose on the Intel iPSC/860[8] and the Intel Paragon[9]. This software mechanism extends the facilities available for direct processor-to-processor communication within a cluster of processors to direct processor-to-processor communication between processors belonging to different clusters of processors. As a result, completely independent software implementations for the component grid time integration schemes can be easily interfaced, thereby preserving a high degree of software modularity on the DM-MIMD computers.

In realistic configurations, there is a high probability for the existence of highly non-uniform distributions of IGBPs and donor cells with respect to the computational spaces of the component grids. Since the computational spaces of the grids are independently partitioned using a 3-D uni-partitioning scheme, it is almost impossible to ensure the eqi-distribution of the IGBPs or the donor cells involved among the processors in different clusters. As a result, it is very likely that significant load imbalances would exist within each group of processors as well as across the groups of processors. This load imbalance would be tolerable, as long as the time spent on intergrid communication is a small fraction of the time required by the component grid time integration scheme that takes the longest time to complete.

Test Problems

The V/STOL Configuration

The computational setup of this configuration[10] 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 height, h, of the delta wing above the ground plane is $h/b = 0.25$, where b is the wing span. Symmetry about the $y = 0$ plane passing through the center line of the delta wing is assumed. The 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).

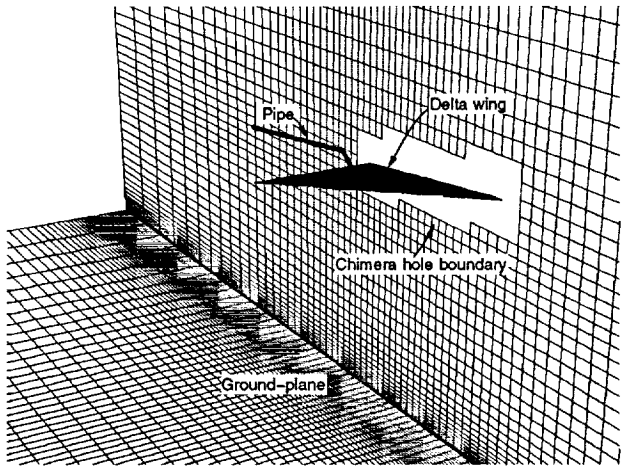

Fig. 1: The ground-plane grid.

The data transfer between the grids is achieved using chimera intergrid communication[3] scheme whereby the first three grids make holes in the fourth grid. The first three grids receive solution information at their outer boundaries from the fourth grid, and the fourth grid receives solution information at the hole boundaries from the first three grids. Solution is not computed within the hole boundaries of the fourth grid.

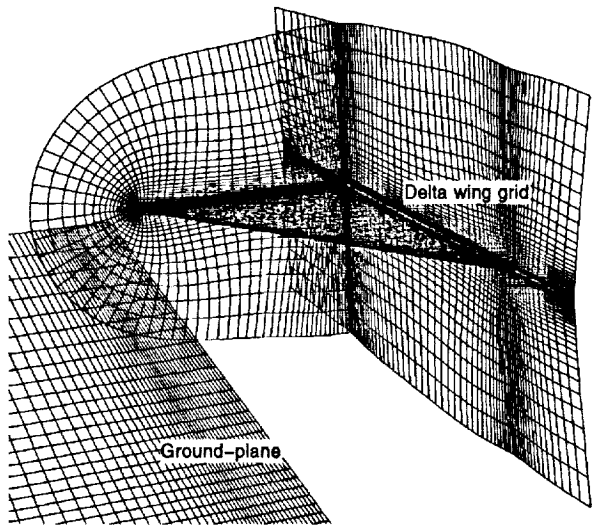

Fig. 2: The delta wing grid.

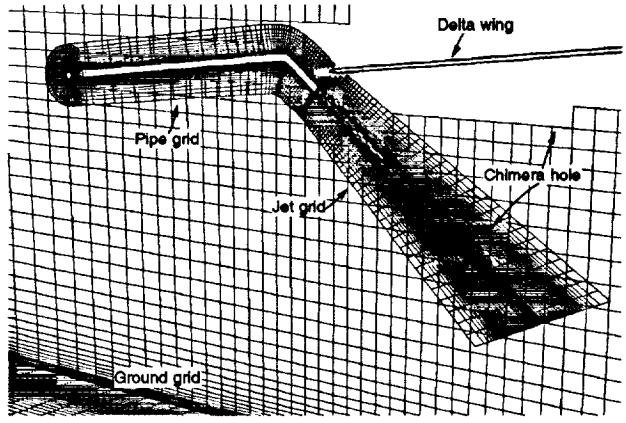

Fig. 3: The pipe and the jet grids.

Boundary Conditions:

On the delta wing and pipe surfaces, a no-slip condition is used, and on the ground, a moving wall condition is used to match the experimental conditions. For both the solid surfaces, density and pressure values are extrapolated from one grid point above the surface. The in-flow and top boundary conditions are specified as free-stream, and the out-flow is extrapolated. At the jet exit the velocity and pressure ratio are set to the experimental conditions.

The High-Lift Configuration

The computational setup of this configuration consists of the FLAC wing with deflected leading and trailing edge flaps at a Mach no. of 0.18, and a Reynolds number of 2.5 million. To compute the air flow at various flap settings, it was decided to use gridding strategies that would minimize the need to regrid the whole geometry at different flap deflection angles. The selected component grid strategy is shown in Fig. 4a. Grids terminate at boundaries between fixed and moving parts, viz. flaps. Flaps have their own grids so that 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 have to be discretized to account for viscous effects (Fig. 4b). Due to airfoil sections with extremely sharp leading and trailing edges, these tips and the wing tip (Fig. 4c) can not be discretized using standard wrap-around grids. However, they lend themselves easily to polar grids slapped on the tips, with the singularity located at the leading or the trailing edge itself (Fig. 4d). 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. 4c). The whole wing-flap system is thus resolved using 18 grids.

The data transfer between the grids is achieved, once again, using chimera intergrid communication scheme whereby the 18 FLAC grids are embedded in a fine global grid which in turn is embedded in a coarse and bigger global grid. The 18 FLAC grids make holes in the fine global grid and receive solution information at their outer boundaries from this grid and from each other. The hole boundary in the fine global grid receives information from the 18 FLAC grids. The fine global grid makes a hole in the coarse global grid and receives information at its outer boundary from the coarse global grid, whereas the coarse global grid receives solution information at its hole boundary from the fine global grid.

Boundary Conditions:

On all the FLAC wing surfaces, a no-slip condition is used, and density and pressure values are extrapolated from one grid point above the surface. The in-flow and top boundary conditions are specified as free-stream, and the out-flow is extrapolated. Viscous wall condition is used on the yz plane at the root to simulate the wind-tunnel wall corresponding to the experimental set up.

Results

The V/STOL Configuration:

The four overset grids comprising the V/STOL configuration were loaded into four different processor clusters. Various mesh partitionings were explored across the processors in a cluster based on grid aspect ratio and intergrid communication load considerations. It was determined that the grid aspect ratio based partitioning optimizes overall efficiency by far. Excellent load balancing was obtained when 112 nodes were used on the Intel iPSC/860 in the manner listed in table 1.

Increasing the number of nodes for grid 4 adversely affected the load balancing and did not result in improved performance. On the Paragon, 156 nodes were split in the manner shown in table 2. Performance comparisons for this simulation requiring about one million grid points are shown in table 3. It should be pointed out that memory reported for the MIMD machines is that associated with the number of nodes used and not the required memory. Fig. 5 shows the performance of this problem on the Intel Paragon for various numbers of nodes grouped in an optimum fashion.

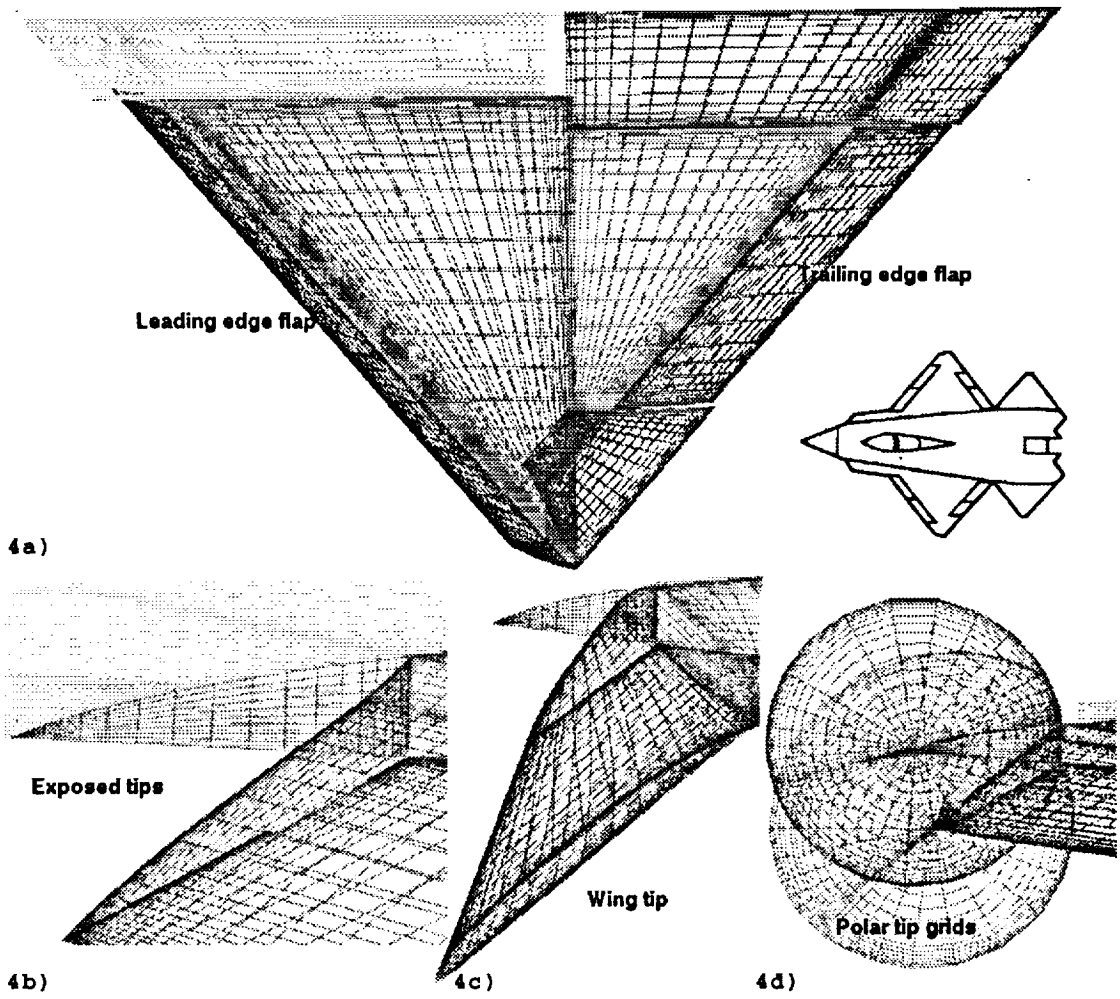

Fig. 4: The FLAC wing grids.

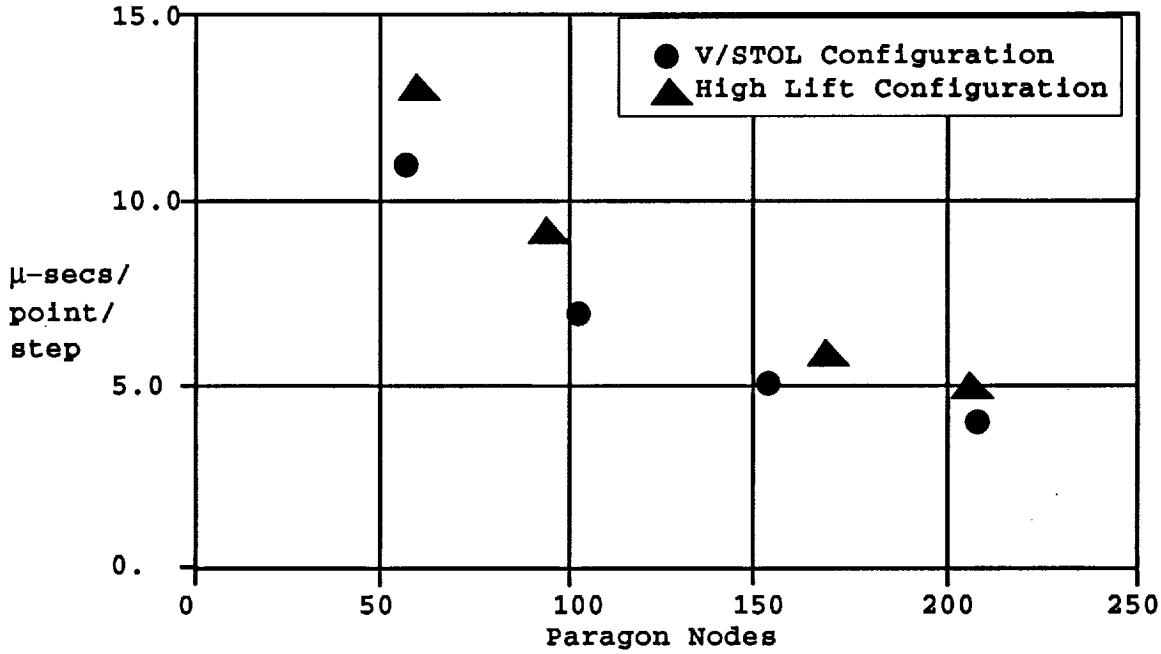

Fig. 5: The Performance on the Paragon.

I/O Performance on the Intel iPSC/860

The unsteady jet flow associated with the VSTOL configuration requires that for detailed analysis, solution be written to the disk every so often. The sustained I/O performance of the iPSC/860 was measured at 0.3 mbytes/sec for solution files written out in the standard plot3d format i.e

$$((((q(i,j,k,n), i = 1, imax), j = 1, jmax), k = 1, kmax), n = 1, 5)$$

where 5 corresponds to the 5 dependent variables. Writing a comprehensive solution file to the disk implies that all nodes must write the solution values corresponding to the grid points that are solved on them. Further, these solutions must be written at specific location in the solution file to make it comprehensive. It is evident then that some time can be saved by writing all 5 dependent variables corresponding to a grid point in one shot rather than writing one dependent variable for all grid points and then the next one, and so on. As a result, solution was saved in a modified format (q(5,i,j,k)) that favors the iPSC/860 architecture and the output performance was measured at 1.1 mbytes/sec. This is substantially better than the performance obtained with the standard format, however, it is deemed inadequate for unsteady problems requiring frequent output of solution. Reasons for poor I/O performance on the iPSC/860 and fixes implemented to improve the same will be summarized in the full paper.

The High Lift Configuration:

The twenty overset grids comprising the High-Lift configuration were loaded into twenty partitions on the Paragon. Again, various grid partitionings were explored across the processors in a group based on grid aspect ratio and intergrid communication load equi-distribution considerations. Once again, grid aspect ratio based partitioning provides far superior performance compared to the intergrid communication load equi-distribution based partitioning. The partitioning used for a total of 207 nodes allocated for this problem is tabulated in table 4. CPU performance obtained using other optimum partitionings used for higher number of nodes are shown in Fig. 5.

Conclusions

Overset multi-zone Navier-Stokes computations comprising of a large number of grids have been performed on two DM-MIMD architectures. An overset grid implicit flow solver has been successfully implemented on two DM-MIMD parallel computers, the Intel iPSC/860 and the Intel Paragon demonstrating the validity of the chosen parallel implementation strategy.

Two realistic test problems comprising of widely varying grid sizes, one with four grids, and the other with twenty grids, were used to evaluate the efficacy of the parallel implementation strategy and the overall performance of the overset grid implicit flow solver. For both the cases, it is shown that grid aspect ratio based partitioning yields optimum performance. Intergrid communication load equi-distribution based partitioning yields sub-optimal performance indicating that intergrid communication costs are indeed a very small fraction of the cost to advance the component grid solutions through one time step.

The CPU performance for the selected test problems is compared on the Intel iPSC/860, the Intel Paragon, and the Cray YMP. It is shown that reasonable performance can be obtained on the Intel machines when compared against the YMP. The performance obtained on the Intel Paragon falls short of the performance expected of a post iPSC/860 generation Intel machine, however, the current performance data is based on a code that has not been optimized for the i860XP processor used on the Paragon.

I/O performance is measured on the Intel iPSC/860 for the unsteady V/STOL problem. The standard plot3d format is modified to cater to the 860 architecture to yield a performance three times better than the standard format, however, is still deemed unsuitable for unsteady problems.

Note to the Reviewers

The missing pieces in this work comprise essentially of obtaining the performance of the 20 grid problem on the Intel iPSC/860. This work is currently in progress. Time permitting, performance numbers will be presented for the Intel Delta at Caltech, and the IBM SP2 system to be installed at NAS in July 1994.

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

[4] 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.

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

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

[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] Barszcz, E., "Intercube Communication for the iPSC/860, "Proceedings of the Scalable High Performance Computing Conference, Williamsburg, VA., April, 1992, pp. 307-313.

[9] Fineberg, S. A., "The Map Library–A Flexible Group Mechanism for the Intel Paragon XP/S," Tech. Report RND-93-015, NASA Ames Research Center, November 1993.

[10] 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.

Grid #	Grid size	Group size
1	(70,56,70)	(4,2,4) = 32 nodes
2	(83,81,47)	(4,4,2) = 32 nodes
3	(60,71,52)	(4,4,2) = 32 nodes
4	(69,71,35)	(2,4,2) = 16 nodes

Table 1: Grid partitioning for the V/STOL configuration on the iPSC/860.

grid #	grid size	Group size
1	(70,56,70)	(4,3,4) = 48 nodes
2	(83,81,47)	(4,4,3) = 48 nodes
3	(60,71,52)	(3,4,3) = 36 nodes
4	(69,71,35)	(3,4,2) = 24 nodes

Table 2: Grid partitioning for the V/STOL configuration on the Paragon.

Section	YMP 1 Proc	iPSC/860 112 Proc	Paragon 156 Proc
time/pt./step	14 micro-secs.	7 micro-secs.	5 micro-secs.
grids solved	sequentially	in parallel	in parallel
memory	12 MW	112 MW	4,992 MW
Accuracy	64 bit	64 bit	64 bit

Table 3: Performance Comparisons for the V/STOL Configuration.

Grid #	Grid size	Group size
1	(62,62,62)	(4,4,3) = 48 nodes
2	(62,62,62)	(4,4,3) = 48 nodes
3	(99,38,30)	(3,2,2) = 12 nodes
4	(49,75,30)	(2,3,2) = 12 nodes
5	(99,38,30)	(3,2,2) = 12 nodes
6	(49,57,31)	(2,3,2) = 12 nodes
7	(79,49,33)	(4,2,2) = 16 nodes
8	(36,68,40)	(2,3,2) = 12 nodes
9	(36,57,30)	(2,3,1) = 6 nodes
10	(36,68,30)	(2,4,1) = 8 nodes
11	(26,57,30)	(1,2,2) = 4 nodes
12	(10,32,50)	(1,1,1) = 1 nodes
13	(14,32,50)	(1,1,2) = 2 nodes
14	(11,32,50)	(1,1,2) = 2 nodes
15	(24,55,20)	(1,2,1) = 2 nodes
16	(24,55,20)	(1,2,1) = 2 nodes
17	(24,55,20)	(1,2,1) = 2 nodes
18	(24,55,20)	(1,2,1) = 2 nodes
19	(24,55,20)	(1,2,1) = 2 nodes
20	(24,55,20)	(1,2,1) = 2 nodes

Table 4: Grid partitioning for the high-lift configuration on the Paragon.