Parallel Computational Fluid Dynamics: Current Status and Future Requirements [1]

Horst D. Simon[2], William R. Van Dalsem[3], and Leonardo Dagum[2]
NASA Ames Research Center
Mail Stop T045-1
Moffett Field, CA 94035

April 8, 1994

Abstract

One of the key objectives of the Applied Research Branch in the Numerical Aerodynamic Simulation (NAS) Systems Division at NASA Ames Research Center is the accelerated introduction of highly parallel machines into a full operational environment. In this report we discuss the performance results obtained from the implementation of some computational fluid dynamics (CFD) applications on the Connection Machine CM-2 and the Intel iPSC/860. We summarize some of the experiences made so far with the parallel testbed machines at the NAS Applied Research Branch. Then we discuss the long term computational requirements for accomplishing some of the grand challenge problems in computational aerosciences. We argue that only massively parallel machines will be able to meet these grand challenge requirements, and we outline the computer science and algorithm research challenges ahead.

Keywords: parallel architectures, MIMD, SIMD, computational fluid dynamics.
AMS Subject Classification 76-08, 65W05, 65N99.
CR Subject Classification G.1.8, J.2, C.1.1, C.1.2.

[1]Sections of this report appeared as [43] and [42].

[2]The author is with the Applied Research Branch, Mail Stop T045-1. The author is an employee of Computer Sciences Corporation. This work is supported through NASA Contract NAS 2-12961.

[3]The author is with the Applied Computational Fluids Branch, Mail Stop T045-2.

1 Introduction

One of the key tasks of the Applied Research Branch in the Numerical Aerodynamic Simulation (NAS) Systems Division at NASA Ames Research Center is the accelerated introduction of highly parallel and related key hardware and software technologies into a full operational environment (see [35]). ¿From 1988 - 1991 a testbed facility has been established for the development and demonstration of highly parallel computer technologies. Currently a 32K processor Connection Machine CM-2 and an 128 node Intel iPSC/860 are operated at the NAS Systems Division. This testbed facility is envisioned to consist of successive generations of increasingly powerful highly parallel systems that are scalable to high performance capabilities beyond that of conventional super computers. In the last two years a number of large scale computational fluid dynamics applications have been implemented on the two testbed machines, and the potential of the parallel machines for production use has been evaluated. Beyond that, a systematic performance evaluation effort has been initiated (see [7, 2, 3]), and basic algorithm research has been continued.

In this report we will first give a brief description of the capabilities of the parallel machines at NASA Ames. Then we will discuss some of the research carried out in the implementation of computational fluid dynamics (CFD) applications on these parallel machines. We focus here on those applications where we have more detailed knowledge because of our own involvement: 3D Navier-Stokes multi-block structured grid codes, an explicit 2D Euler solver for unstructured grids, and a simulation based on particle methods. Finally we will outline the computational requirements for large scale aerosciences grand challenge applications by analyzing one such application at NASA Ames. In the last section we offer some preliminary conclusions on the performance of current parallel machines for CFD applications, as well as the potential of the different architectures for production use in the future. Another summary of the experience with parallel machines at NASA Ames is given by D. Bailey in [5]. A more comprehensive survey of the NASA Computational Aerosciences Program with more emphasis on the applications is given in [22].

2 Parallel Machines at NASA Ames

2.1 Connection Machine

The Thinking Machines Connection Machine Model CM-2 is a massively parallel SIMD computer consisting of many thousands of bit serial data processors under the direction of a front end computer. The system at NASA Ames consists of 32768 bit serial processors each with 1 Mbit of memory and operating at 7 MHz. The processors and memory are packaged as 16 in a chip. Each chip also contains the routing circuitry which allows any processor to send and receive messages from any other processor in the system. In addition, there are 1024 64-bit Weitek floating point processors which are fed from the bit serial processors through a special purpose "Sprint" chip. There is one Sprint chip connecting every two CM chips to a Weitek. Each Weitek processor can execute an add and a multiply each clock cycle thus performing at 14 MFLOPS and yielding a peak aggregate performance of 14 GFLOPS for the system.

The Connection Machine can be viewed two ways, either as an eleven dimensional hypercube connecting the 2048 CM chips or as a ten dimensional hypercube connecting the 1024 processing elements. The first view is the "fieldwise" model of the machine which has existed since its introduction. This view admits to the existence of at least 32768 physical processors (when using the whole machine), each storing data in fields within its local memory. The second is the more recent "slicewise" model of the machine, which admits to only 1024 processing elements (when using the whole machine), each storing data in slices of 32 bits distributed across the 32 physical processors in the processing element. Both models allow for "virtual processing", where the resources of a single processor or processing element may be divided to allow a greater number of virtual processors.

Regardless of the machine model, the architecture allows interprocessor communication to proceed in three manners. For very general communication with no regular pattern, the router determines the destination of messages at run time and directs the messages accordingly. This is referred to as general router communication. For communication with an irregular but static pattern, the message paths may be pre-compiled and the router will direct messages according to the pre-compiled paths. This is referred to as compiled communication and can be 5 times faster than general router communication.

Finally, for communication which is perfectly regular and involves only shifts along grid axes, the system software optimizes the data layout by ensuring strictly nearest neighbor communication and uses its own pre-compiled paths. This is referred to as NEWS (for "NorthEastWestSouth") communication. Despite the name, NEWS communication is not restricted to 2-dimensional grids, and up to 31-dimensional NEWS grids may be specified. NEWS communication is the fastest. An analysis of the communication speed of the CM can be found in [29].

The I/O subsystems connect to the data processors through an I/O controller. An I/O controller connects to 8192 processors through 256 I/O lines. There is one line for each chip but the controller can only connect to 256 lines simultaneously and must treat its 8K processors as two banks of 4K each. Each I/O controller allows transfer rates of up to 40 MB per second. In addition to an I/O controller there can be a frame buffer for color graphics output. Because it is connected directly to the backplane rather than through the I/O bus, the frame buffer can receive data from the CM processors at 256 MB per second. The system at NASA Ames has two frame buffers connected to two high resolution color monitors and four I/O controllers connected to a 20 GB DataVault mass storage system.

The Connection Machine's processors are used only to store and process data. The program instructions are stored on a front-end computer which also carries out any scalar computations. Instructions are sequenced from the front end to the CM through one or more sequencers. Each sequencer broadcasts instructions to 8192 processors and can execute either independent of other sequencers or combined in two or four. There are two front end computers at NASA Ames, a Vax 8350 and a Sun 4/490, which currently support about 100 users. There are two sequencer interfaces on each computer which allow up to four dedicated processes. In addition, the system software supports the Network Queue System (NQS) and time sharing through the CM Time Sharing System (CMTSS).

The Connection Machine system was first installed at NASA Ames in June of 1988. Since then the system has undergone a number of upgrades, the most recent being completed in February of 1991. An assessment of the system is given in [40]. Perhaps its greatest strength, from a user standpoint, is the robust system software. This is of critical importance to NASA as it moves its parallel machines into production mode.

2.2 Intel iPSC/860

The Intel iPSC/860 (also known as Touchstone Gamma System) is based on the 64 bit i860 microprocessor by Intel [23]. The i860 has over 1 million transistors and runs at 40 MHz. The theoretical peak speed is 80 MFLOPS in 32 bit floating point and 60 MFLOPS for 64 bit floating point operations. The i860 features 32 integer address registers, with 32 bits each, and 16 floating point registers with 64 bits each (or 32 floating point registers with 32 bits each). It also features an 8 kilobyte on-chip data cache and a 4 kilobyte instruction cache. There is a 128 bit data path between cache and registers. There is a 64 bit data path between main memory and registers.

The i860 has a number of advanced features to facilitate high execution rates. First of all, a number of important operations, including floating point add, multiply and fetch from main memory, are pipelined operations. This means that they are segmented into three stages, and in most cases a new operation can be initiated every 25 nanosecond clock period. Another advanced feature is the fact that multiple instructions can be executed in a single clock period. For example, a memory fetch, a floating add and a floating multiply can all be initiated in a single clock period.

A single node of the iPSC/860 system consists of the i860, 8 megabytes (MB) of dynamic random access memory, and hardware for communication to other nodes. For every 16 nodes, there is also a unit service module to facilitate access to the nodes for diagnostic purposes. The iPSC/860 system at NASA Ames consists of 128 computational nodes. The theoretical peak performance of this system is thus approximately 7.5 GFLOPS on 64 bit data.

The 128 nodes are arranged in a seven dimensional hypercube using the direct connect routing module and the hypercube interconnect technology of the iPSC/2. The point to point aggregate bandwidth of the interconnect system, which is 2.8 MB/sec per channel, is the same as on the iPSC/2. However the latency for the message passing is reduced from about 350 microseconds to about 90 microseconds. The improved latency is mainly a product of the faster execution of the message passing software on the i860 compared to the slower Intel 80386 on the iPSC/2.

Attached to the 128 computational nodes of the NASA Ames system are ten I/O nodes, each of which can store approximately 700 MB. The total capacity of the I/O system is thus about 7 GB. These I/O nodes operate concurrently for high throughput rates. The complete system is controlled by a system resource module (SRM), which is based on an Intel 80386 processor. The SRM originally handled compilation and linking of source programs, as well as loading the executable code into the hypercube nodes and initiating execution. As such, the SRM became a serious bottleneck in the system, due to its slowness in compiling and linking user codes. Intel has since alleviated the problem by providing cross-compilers for Sun and Silicon Graphics workstations and system software to allow remote loading of executable code.

During 1990 the iPSC/860 has been thoroughly investigated at NASA Ames. A first set of benchmark numbers, and some CFD applications performance numbers have been published in [4]. A more recent summary is given by Barszcz in [8]. As documented in [8] from an overall systems aspect the main bottleneck was the SRM, which is not able to handle the demands of a moderately large user community (about 50 to 100 users) in a production environment. Another important result of the investigations was the outcome of a study by Lee [25]. Lee's analysis of the i860 floating point performance indicates that on typical CFD kernels the best performance to be expected is in the 10 MFLOPS range. Finally we mention a two performance studies of the I/O system by Lou [30] and Ryan [39], which measure the I/O performance of the concurrent file system (CFS), the parallel I/O device delivered by Intel.

3 Structured Grid Applications

Structured grid flow solvers, in particular multi-block structured grid flow solvers, are the main class of production CFD tools at NASA Ames. A number of different efforts were directed toward the implementation of such capabilities on parallel machines. One of the first CFD results on the CM-2 was the work by Levit and Jespersen [26, 27], which was recently extended to three dimensions [28]. Their implementation is based on the successful ARC2D and ARC3D codes developed by Pulliam [38]. Work by Barszcz and Chawla [9] is in progress to implement F3D, a successor code to ARC3D, on the CM-2. On the iPSC/860 Weeratunga has implemented ARC2D (for early results see [4]), and work is in progress to implement F3D. Weeratunga also has developed three simulated CFD applications based on structured grid flow solvers for the NAS Parallel Benchmarks, which are described in

Chapter 3 of [7].

The results obtained by Weeratunga, Barszcz, Fatoohi, and Venkatakrishnan on the simulated CFD applications benchmark are indicative for the current performance level of parallel machines on implicit CFD algorithms. Performance results for "kernel" benchmarks do not fully reflect the computational requirements of a realistic, state-of-the-art CFD application. This is because a data structure that is optimal for one particular part of the computation on a given system might be very inefficient for another part of the computation. As a result, the three "simulated CFD application" benchmarks were devised. These three benchmarks are intended to accurately represent the principal computational and data movement requirements of modern implicit CFD applications. They model the main building blocks of CFD codes designed at NASA Ames for the solution of 3D Euler/Navier-Stokes equations using finite-volume/finite-difference discretization on structured grids.

There is one important feature which characterizes these simulated applications from a computational point of view. All three involve approximate factorization techniques, which in turn require the solution of three sets of multiple, independent, sparse, but structured systems of linear equations at each time step. Each of three sets of solves keeps one coordinate direction fixed, and solves the multiple sets of linear systems in the direction of the grid planes orthogonal to the fixed direction. Thus the three dimensional computational grid must be accessed by planes in three different directions. This has a very important implication for distributed memory machines: no single allocation scheme for the three dimensional grid is optimal. In order to carry out the solver phase efficiently in the three different grid directions the grids will have to be redistributed among the processors. The key to an efficient implementation of the simulated application benchmark is then to devise optimal distribution and communication schemes for the transition between the three solve phases at each time step[1].

The first of the simulated applications is the LU benchmark. In this benchmark, a regular-sparse, block (5×5) lower and upper triangular system is solved. This problem represents the computations associated with the implicit operator of a newer class of implicit CFD algorithms, typified at

[1]It should be pointed out that this discussion of the simulated applications does not apply to all production CFD codes at NASA Ames. For example the widely used F3D code, as well as the UPS code, are for example based on a two factor scheme.

Table 1: **Results for the LU Simulated CFD Application**

System	No. Proc.	Time/Iter. (secs.)	MFLOPS (Y-MP)
Y-MP	1	1.73	246
	8	0.25	1705
iPSC/860	64	3.05	139
	128	1.90	224
CM-2	8K	5.23	82
	16K	3.40	125
	32K	2.29	186

NASA Ames by the code INS3D-LU [47]. This problem exhibits a somewhat limited amount of parallelism compared to the next two.

The second simulated CFD application is called the scalar penta-diagonal (SP) benchmark. In this benchmark, multiple independent systems of non-diagonally dominant, scalar, penta-diagonal equations representative of computations associated with the implicit operators of CFD codes such as ARC3D [38] at NASA Ames Research Center. SP and BT are similar in many respects, but there is a fundamental difference with respect to the communication to computation ratio.

The third simulated CFD application is called the block tri-diagonal (BT) benchmark. In this benchmark, multiple independent systems of non-diagonally dominant, block tri-diagonal equations with a (5×5) block size are solved (for a related discussion of the parallel implemenation of ARC3D see also [34]).

Performance figures for the three simulated CFD applications are shown in Tables 1, 2 and 3. Timings are cited in seconds per iteration. In all three tables results are reported for grids of size $64 \times 64 \times 64$. A complete solution of the LU benchmark requires 250 iterations. For the SP benchmark, 400 iterations are required. For the BT benchmark, 200 iterations are required. The MFLOPS in these tables for the parallel machines are based on an operation count established for the sequential version of the program.

Table 2: **Results for the SP Simulated CFD Application**

System	No. Proc.	Time/Iter. (secs.)	MFLOPS (Y-MP)
Y-MP	1	1.18	250
	8	0.16	1822
iPSC/860	64	2.42	122
CM-2	8K	9.75	30
	16K	5.26	56
	32K	2.70	109

Table 3: **Results for the BT Simulated CFD Application**

System	No. Proc.	Time/Iter. (secs.)	MFLOPS (Y-MP)
Y-MP	1	3.96	224
	8	0.57	1554
iPSC/860	64	4.54	199
CM-2	16K	16.64	54
	32K	9.57	94

4 Unstructured Grid Applications

We discuss here work on an unstructured upwind finite-volume explicit flow solver for the Euler equations in two dimensions that is well suited for massively parallel implementation. The mathematical formulation of this flow solver was proposed and implemented on the Cray-2 by Barth and Jespersen[10]. This solver has been implemented on the CM-2 by Hammond and Barth [20], and on the Intel iPSC/860 by Venkatakrishnan, Simon, and Barth [46].

The unstructured grid code developed by Barth is a vertex-based finite-volume scheme. The control volumes are non-overlapping polygons which surround the vertices of the mesh, called the "dual" of the mesh. Associated with each edge of the original mesh is a dual edge. Fluxes are computed along each edge of the dual in an upwind fashion using an approximate Riemann solver. Piecewise linear reconstruction is employed which yields second order accuracy in smooth regions. A four stage Runge-Kutta scheme is used to advance the solution in time. Fluxes, gradients and control volumes are all constructed by looping over the edges of the original mesh. A complete description of the algorithm can be found in [10, 20]. It is assumed that a triangularization of the computational domain and the corresponding mesh has been computed.

In both implementations the same four element wing cross-section test case has been used. The test case unstructured mesh includes 15606 vertices, 45878 edges, 30269 faces, and 949 boundary edges. The flow was computed at a freestream Mach number of .1 and 0 degrees angle of attack. The code for this test case runs at 150 MFLOPS on the NAS Cray Y-MP at NASA Ames, and requires 0.39 seconds per time step. In the Cray implementation, vectorization is achieved by coloring the edges of the mesh.

4.1 SIMD Implementation of Unstructured Solver

For the implementation on the CM-2 Hammond and Barth [20] used a novel partitioning of the problem which minimizes the computation and communication costs on a massively parallel computer. The following description follows [20] closely. In a mesh-vertex scheme, solution variables are associated with each vertex of the mesh and flux computation is performed at edges of the non-overlapping control volumes which surround each vertex.

In conventional parallel implementations this operation is partitioned to be performed edge-wise, i.e., each *edge* of the control volume is assigned to one processor (edge-based). The resulting flux calculation contributes to two control volumes which share the particular edge.

In the partitioning used by Hammond and Barth, each *vertex* of the mesh is assigned to one processor (vertex-based). Flux computations are identical to the edge-based scheme but computed by processors associated with vertices. Each edge of the mesh joins a pair of vertices and is associated with one edge of the control volume.

One can direct an edge (i,j) to determine which vertex in the pair computes the flux through the shared edge of the control volume, (k', j'). When there is a directed edge from i to j, then the processor holding vertex j sends its conserved values to the processor holding vertex i, and the flux across the common control volume edge is computed by processor i and accumulated locally. The flux through (k', j') computed by the processor holding vertex i is sent to the processor holding vertex j to be accumulated negatively. Hammond and Barth show that their vertex-based scheme requires 50% less communication and asymptotically identical amounts of computation as compared with the traditional edge-based approach.

Another important feature of the work by Hammond and Barth is the use of fast communication. A feature of the communication within the flow solver here is that the communication pattern, although irregular, remains static throughout the duration of the computation. The SIMD implementation takes advantage of this by using a mapping technique developed by Hammond and Schreiber [21] and a "Communication Compiler" developed for the CM-2 by Dahl [17]. The former is a highly parallel graph mapping algorithm that assigns vertices of the grid to processors in the computer such that the sum of the distances that messages travel is minimized. The latter is a software facility for scheduling completely general communications on the Connection Machine. The user specifies a list of source locations and destinations for messages and enables one to fully utilize the large communication bandwidth of the machine.

Hammond and Barth have incorporated the mapping algorithm and the communication compiler into the flow solver running on the CM-2 and have realized a factor of 30 reduction in communication time compared to using naive or random assignments of vertices to processors and the router. Originally, using 8K processors of the CM-2 and a virtual processor (VP) ratio of

2, Hammond and Barth carried out 100 time steps of the flow solver in about 71.62 seconds. An improved implementation by Hammond in [19] resulted in 43 seconds per 100 time steps, which is equivalent to 136 MFLOPS. This does not include setup time.

4.2 MIMD Implementation of Unstructured Solver

Similar to the SIMD implementation one of the key issues is the partitioning of the unstructured mesh. In order to partition the mesh Venkatakrishnan et al. [46] employ a new algorithm for the graph partitioning problem, which has been discussed recently by Simon [41], and which is based on the computation of eigenvectors of the Laplacian matrix of a graph associated with the mesh. Details on the theoretical foundations of this strategy can be found in [37]. Detailed investigations and comparisons to other strategies (cf. [41]) have shown that the spectral partitioning produces subdomains with the shortest boundary, and hence tends to minimize communication cost.

After the application of the partition algorithm of the previous section, the whole finite volume grid with triangular cells is partitioned into P subgrids, each subgrid contains a number of triangular cells which form a single connected region. Each subgrid is assigned to one processor. All connectivity information is precomputed, using sparse matrix type data structures.

Neighboring subgrids communicate to each other only through their interior boundary vertices which are shared by the processors containing the neighboring subgrids. In the serial version of the scheme, field quantities (mass, momentum and energy) are initialized and updated at each vertex of the triangular grid using the conservation law for the Euler equations applied to the dual cells. Each processor performs the same calculations on each subgrid as it would do on the whole grid in the case of a serial computation. The difference is that now each subgrid may contain both physical boundary edges and interior boundary edges, which have resulted from grid partitioning. Since a finite volume approach is adopted, the communication at the inter-processor boundaries consists of summing the local contributions to integrals such as volumes, fluxes, gradients etc.

The performance of the Intel iPSC/860 on the test problem is given in Table 4. The MFLOPS given are based on operation counts using the Cray hardware performance monitor. The efficiency is computed as

$$Efficiency(\%) = \frac{MFLOPS\ with\ N\ procs}{N\ *\ (MFLOPS\ with\ 1\ proc)} * 100.$$

Table 4: **Performance of Unstructured Grid Code on the Intel iPSC/860**

Processors	secs/step	MFLOPS	efficiency(%)
2	7.39	7.9	86
4	3.70	15.8	86
8	1.94	30.2	82
16	1.08	54.1	74
32	0.59	99.2	67
64	0.31	187.5	64
128	0.19	307.9	52

In summary the performance figures on the unstructured grid code are given in Table 5, where all MFLOPS numbers are Cray Y-MP equivalent numbers.

Table 5: **Performance Comparison of Unstructured Grid Code**

Machine	Processors	secs/step	MFLOPS
Cray Y-MP	1	0.39	150.0
Intel iPSC/860	64	0.31	187.5
	128	0.19	307.9
CM-2	8192	0.43	136

5 Particle Methods

Particle methods of simulation are of interest primarily for high altitude, low density flows. When a gas becomes sufficiently rarefied the constitutive relations of the Navier-Stokes equations (i.e. the Stokes law for viscosity and the Fourier law for heat conduction) no longer apply and either higher order relations must be employed (e.g. the Burnett equations [31]), or the continuum approach must be abandoned and the molecular nature of the gas must be addressed explicitly. The latter approach leads to direct particle simulation.

In direct particle simulation, a gas is described by a collection of simulated molecules thus completely avoiding any need for differential equations explicitly describing the flow. By accurately modeling the microscopic state of the gas, the macroscopic description is obtained through the appropriate integration. The primary disadvantage of this approach is that the computational cost is relatively large. Therefore, although the molecular description of a gas is accurate at all densities, a direct particle simulation is competitive only for low densities where accurate continuum descriptions are difficult to make.

For a small discrete time step, the molecular motion and collision terms of the Boltzmann equation may be decoupled. This allows the simulated particle flow to be considered in terms of two consecutive but distinct events in one time step, specifically there is a collisionless motion of all particles followed by a motionless collision of those pairs of particles which have been identified as colliding partners. The collisionless motion of particles is strictly deterministic and reversible. However, the collision of particles is treated on a probabilistic basis. The particles move through a grid of cells which serves to define the geometry, to identify colliding partners, and to sample the macroscopic quantities used to generate a solution.

The state of the system is updated on a per time step basis. A single time step is comprised of five events:

1. Collisionless motion of particles.

2. Enforcement of boundary conditions.

3. Pairing of collision partners.

4. Collision of selected collision partners.

5. Sampling for macroscopic flow quantities.

Detailed description of these algorithms may be found in [32] and [13]

5.1 SIMD Implementation of Particle Simulation

Particle simulation is distinct from other CFD applications in that there are two levels of parallel granularity in the method. There is a coarse level consisting of cells in the simulation (which are approximately equivalent to grid points in a continuum approach) and there is a fine level consisting of individual particles. At the time of the CM-2 implementation there existed only the fieldwise model of the machine, and it was natural for Dagum [13] to decompose the problem at the finest level of granularity. In this decomposition, the data for each particle is stored in an individual virtual processor in the machine. A separate set of virtual processors (or VP set) stores the geometry and yet another set of virtual processors stores the sampled macroscopic quantities.

This decomposition is conceptually pleasing however in practice the relative slowness of the Connection Machine router can prove to be a bottleneck in the application. Dagum [13] introduces several novel algorithms to minimize the amount of communication and improve the overall performance in such a decomposition. In particular, steps 2 and 3 of the particle simulation algorithm require a somewhat less than straightforward approach.

The enforcement of boundary conditions requires particles which are about to interact with a boundary to get the appropriate boundary information from the VP set storing the geometry data. Since the number of particles undergoing boundary interaction is relatively small, a master/slave algorithm is used to minimize both communication and computation. In this algorithm, the master is the VP set storing the particle data. The master creates a slave VP set large enough to accommodate all the particles which must undergo boundary interactions. Since the slave is much smaller than the master, instructions on the slave VP set execute much faster. This more than makes up for the time that the slave requires to get the geometry information and to both get and return the particle information.

The pairing of collision partners requires sorting the particle data such that particles occupying the same cell are represented by neighboring virtual processors in the one dimensional NEWS grid storing this data. Dagum [15] describes a very efficient sorting algorithm suitable for this purpose. The algorithm makes use of the realization that the particle data moves through the CM processors in a manner analogous to the motion of the particles in the simulation. The mechanism for disorder is the motion of particles, and the extent of motion of particles, over a single time step, is small. This can be used to greatly reduce the amount of communication necessary to re-order the particles.

These algorithms have been implemented in a three-dimensional particle simulation running on the CM-2. The implementation was written in C/Paris and is described in [16]. The code has been used to simulate the flow over a re-entry vehicle using over 3.2×10^7 particles in a grid with 4.5×10^5 cells at a rate of 2.4μsec/particle/time step using all 32K processors. By comparison, a fully vectorized equivalent simulation on a single processor of the Cray YMP runs at 1.0μsec/particle/time step and 86 MFLOPS as measured by the Cray hardware performance monitor. (Note that a significant fraction of a particle simulation involves integer arithmetic and the MFLOP measure is not completely indicative of the amount of computation involved).

5.2 MIMD Implementation of Particle Simulation

The MIMD implementation differs from the SIMD implementation not so much because of the difference in programming models but because of the difference in granularity between the machine models. Whereas the CM-2 has 32768 processors, the iPSC/860 has only 128. Therefore on the iPSC/860 it is natural to apply a spatial domain decomposition rather than the data object decomposition used on the CM-2.

In McDonald's [33] implementation, the spatial domain of the simulation is divided into a number of sub-domains greater than or equal to the desired number of node processes. Communication between processes occurs as a particle passes from one region to another and is carried out asynchronously, thus allowing overlapping communication and computation. Particles crossing region "seams" are treated simply as an additional type of boundary condition. Each simulated region of space is surrounded by a shell of extra cells that, when entered by a particle, directs that particle to the neighboring sub-domains. This allows the representation of simulated space (i.e. the geometry definition) to be distributed along with the particles. The aim is

Table 6: **Performance of Particle Simulation on the Intel iPSC/860**

Processors	μs/prt/step	MFLOPS	efficiency(%)
2	24.4	3.5	97
4	12.5	6.9	95
8	6.35	13.5	93
16	3.25	26.5	91
32	1.63	52.8	91
64	0.85	101	87
128	0.42	215	88

to avoid maintaining a representation of all simulated space which, if stored on a single processor, would quickly become a serious bottleneck for large simulations, and if replicated would simply be too wasteful of memory.

Within each region the sequential or vectorized particle simulation is applied. This decomposition allows for great flexibility in the physical models that are implemented since node processes are asynchronous and largely independent of each other. Recall that communication between processes is required only when particles cross region seams. This is very fortuitous since the particle motion is straightforward and fully agreed upon. The important area of research has to do with the modelling of interaction of particles with solid boundaries and each other, and since this part of the problem does not directly affect communication, particle models can evolve without requiring great algorithmic changes.

McDonald's implementation is fully three-dimensional with dynamic load balancing and chemistry modelling. The performance of the code on a 3D heat bath is given in Table 6. The geometry and spatial decomposition of the heat bath simulation *exaggerated* the area to volume ratio of the regions in order to be conservative in approximating the performance in a real application. The most promising feature of these results is the linear speed up obtained when the problem size is allowed to scale with the number of processors. This indicates that the performance of the code should continue to increase with larger system configurations.

The domain decomposition is dynamic thus permitting a good load balance to exist throughout a calculation. Load balancing is accomplished by allowing a number of sub-domains to exist at each processing node. As the load becomes unbalanced, sub-domains are reassigned to processors in a manner that approximates an equal workload at each node. The balancing operation is repeated a number of times as the solution develops but is unnessecary once a steady state situation is reached. Other simulation costs such as memory usage and communication can also be balanced by appropriately assigning sub-domains to processors. For example, if neighboring sub-domains in the physical domain are assigned to the same processor, communication is not required as a particle moves from one sub-domain to the next.

For the particle methods the corresponding summary of performance figures for all three machines can be found in Table 7. The figures in Table 7 should be interpreted very carefully. The simulations run on the different machines were comparable, but not identical. The MFLOPS are Cray Y-MP equivalent MFLOPS ratings based on the hardware performance monitor. Only 32-bit arithmetic is required in the method however 64-bit arithmetic is used on the Cray systems.

Table 7: **Performance Comparison of Particle Simulation Code**

Machine	Processors	μsecs/particle/step	MFLOPS
Cray 2	1	2.0	43
Cray Y-MP	1	1.0	86
Intel iPSC/860	128	0.4	215
CM-2	32768	2.0	43

6 Grand Challenge Computational Requirements

We would like to contrast now what has been achieved so far with the "Grand Challenges" to be solved on parallel machines in the 1990s. As part of the "Federal High Performance Computing Program", NASA's portion of the "High Performance Computing and Communication Program (HPCCP)" focuses on research and development in areas which show promise to deliver new capabilities to important NASA missions by the late 1990s (for more details see [36]). Two NASA grand challenges have been chosen as focal points for the HPCCP. A grand challenge is a fundamental problem in science and engineering, with broad applications, whose solution would be enabled by the application of high performance computing technology, which could become available in the near future. An important criterion for the selection of grand challenge applications was the breadth of technical considerations presented in a grand challenge, as well as the potential for applying the newly developed technologies beyond the specific problem area. The two NASA grand challenges are:

- integrated, multi-disciplinary simulations and design optimizations of aerospace vehicles throughout their mission profiles.
- multi-disciplinary modeling and data analysis of earth and space science physical phenomena.

The first grand challenge is the focus of the NASA Computational Aero-Sciences (CAS) program [22]. Within this program, activities are focused on the development of multi-disciplinary design tools for the high-speed civil transport (HSCT) and high-performance aircraft (HPA). In the high-performance aircraft area, the primary interest is to develop the capability to predict the performance of next generation fighter concepts operating in the most critical portions of their flight regime. To achieve performance levels beyond present generation vehicles, these next generation fighters designs must include higher levels of system integration than can be obtained with present design tools. Towards this goal, aerodynamic, propulsion system, controls, structural, and even acoustic, analysis modules will be integrated into a single software system. The challenges posed by the development and application of such a multi-disciplinary high-performance aircraft analysis tool will be used to illustrate the computational issues in such grand challenge computations.

6.1 Grand Challenges of the 1990's (An Example)

Powered-lift aircraft utilize a mix of wing-borne and propulsive lift to achieve vertical or short take-off and landings (V/STOL). With careful design, powered-lift aircraft can also out perform conventional aircraft in other portions of the flight envelope via the use of powered-lift features (e.g., vectoring thrust to achieve super maneuverability). Successful powered-lift aircraft designs are developed from a detailed understanding of the interaction of very complex fluid flows (see Figure 1), with all of the major aircraft sub-systems, including the airframe, and propulsion and control systems. Until recently, no computational techniques have been available for the analysis of these complex powered-lift flows [45], and multi-disciplinary interactions [1]. Hence, the design of high-performance powered-lift aircraft has been among the most time-consuming and costly aerospace design activities. As an example, the Harrier was originally conceived in the mid 1950's and is still undergoing significant design studies [18]. Therefore, development of advanced multi-disciplinary analysis tools is being pursued.

A successful computational design tool for high-performance powered-lift aircraft must be able to predict aerodynamic, thermal, and acoustic loads for a vehicle during operations in-ground-effect, transition from jet-borne to wing-borne flight, and in up-and-away flight. Also of key interest is the prediction of engine performance during V/STOL and high-angle-of-attack maneuvers, when inlet flow distortion may degrade thrust or result in engine compressor stall. The V/STOL and transition modes also put severe challenges on the performance of the control system in utilizing the airframe and propulsion systems to retain stable flight.

To model these interactions, at least six computational modules must be integrated (see Figure 2):

- Navier-Stokes
- Engine performance
- Structural Heating

Figure 1: **Harrier Jet in Ground Effect**

- Acoustics
- Control (including pilot model or auto-pilot)
- Aircraft dynamics

Work is presently underway in the Powered-Lift Group of the Applied Computational Fluids Branch at NASA-Ames Research Center towards the Navier-Stokes/Structural Heating/Engine Deck analysis of a Harrier AV-8B in-ground-effect [44]. Work is also underway at NASA- Lewis to develop advanced propulsion system analysis capabilities. Future HPCCP high performance aircraft goals include integrating the aircraft and propulsion analysis tools presently being developed at Ames and Lewis, respectively, into a complete vehicle analysis tool applicable to next generation fighter concepts.

6.2 Surface Modeling and Grid Generation Requirements

A major bottleneck in the application of the described computational design tools will be the development of surface modeling and grid generation

Figure 2: **Powered-Lift Integrated Multi-Disciplinary System**

software which allows:

1. Surface model definition in less than 1 week
2. Complete grid generation in less than 1 week
3. Design change/regridding of components in less than 1 day
4. Vehicle deformation (e.g., aero-elastic effects) during computation
5. Relative vehicle motion (e.g., landing/take-off) and effector (e.g., flaps and jets) movement during computation

Tasks 1-3 require the development of powerful interactive software tools on workstation platforms, with Task 2 requiring some distributed processing to a super computer (vector or parallel). These requirements are very challenging, but do not necessarily involve parallel computers, and will not be addressed in detail here.

Tasks 4 and 5 must be performed during the numerical simulation on the parallel computer systems. Accommodating vehicle deformation (Task 4) will require that a parametric representation (e.g., NURBS) of the vehicle surface reside on the parallel computer, and that this geometric representation can be manipulated and sampled dynamically without user intervention. It will also be required that new volume grids be created dynamically, using the deforming vehicle geometry as the new boundary condition for the algebraic or PDE (e.g., elliptic) volume grid generator. Accounting for vehicle and effector movement (Task 5) will be best accommodated using an overset grid technology (e.g. [11, 12]). In this case, as the aircraft moves in relationship to the ground (for example) the grids attached to the aircraft and ground will be in relative motion, and new interpolation stencils must be computed at each iteration. This requires that the nearest-point and interpolation features of the overset-grid technology be ported to the parallel computers. Considering that the technology required for Tasks 4-5 is only in the formative stages of development on vector computers, the challenge of fully-developing this software and implementing it in the parallel environment is formidable.

6.3 Flow Simulation (CFD) Requirements

In 1991, a state-of-the-art simulation of the flow about a Harrier operating in-ground effect required approximately 2.8 million points, 20 Mwords of run-time memory, and about 40 hours of CPU time on a Cray Y-MP running at a sustained speed of approximately 160 MFLOPS. Such a computation solves the Navier-Stokes equations for the viscous flow about the Harrier using, in this case, a simple algebraic turbulence model. The grid was the coarsest possible that would still allow most of the important flow features to be resolved. The predicted flow features are in good agreement with flight flow visualization [44].

It is estimated that to obtain "engineering-accuracy" predictions of surface pressures, heat transfer rates, and overall forces, the grid size will have to be increased to a minimum of 5.0 million points. If the unsteady motion of the flow structures is to be resolved, at least 50,000 iterations will also be required. Also, more advanced turbulence modeling must be included. In summary, we anticipate the following minimum requirements in terms of floating point operations for just the external flow simulation element of future grand challenge computations:

- 5,000,000 grid points
- 50,000 iterations

- 5,000 operations per point per iteration
- 10^{15} operations per problem

The actual computational speed requirements for such a calculation depend on the mode in which the calculation is carried out. In a proof-of-concept mode such a calculation may be carried out only once as a "heroic effort". If this could be done in 100 to 1000 hours turn-around-time, it would translate into a sustained speed between 3 and 0.3 GFLOPS. Design and automated design modes require a much lower turn-around-time and thus result in much higher requirements for computational speed. The corresponding figures are summarized in Table 8.

Table 8: **Requirements for Flow Simulation**

Solution Mode	**Turn-around-time**	**Required Performance**
Proof-of-concept	1000 – 100 hours	0.3 – 3 GFLOPS
Design	10 – 1 hours	30 – 300 GFLOPS
Automated Design	0.1 – 0.01 hours	3 – 30 TFLOPS

These computational requirements are accompanied by a corresponding increase in memory and storage requirements. Approximately 40 storage locations are required per grid point. If all of the computational zones remain in memory, this translates to a requirement for 200 million words of run-time memory (to date, often a desirable feature for parallel systems). For unsteady flow analysis 100-1000 time steps (at 8 words per point) must be stored. This leads to a requirement of 4-40 gwords of "disk" storage per problem.

If we compare these requirements with the computer resources required to address the "grand challenges" of the 1980's (e.g., a 1.0 million point steady Navier-Stokes simulation, on a Cray-2 class machine, of the external flow about an aircraft at cruise) we arrive at Table 9.

We note in particular that a 5000 fold increase in data storage and manipulation capabilities will be required to address CFD grand challenges of the 1990's. A single solution file for a time step will have up to 40 Mwords (320 Mbytes) of data. The above discussion assumes that the computation for advancing the solution one time step can be carried out in about 10 seconds. Even though it is not necessary to store a solution file at every time step, these figures show the need for a sustained I/O bandwidth of at least 40

Table 9: **Proof-of-concept requirements: 1980's vs. 1990's**

	1980's	**1990's**	**Ratio**
100 hr. run time	40 MFLOPS	3000 MFLOPS	75
run-time memory	35 Mwords	200 Mwords	6
"disk" storage	8 Mwords	40000 Mwords	5000

Mbytes/sec. For a more detailed discussion of I/O requirements for parallel CFD see the report by Ryan [39].

6.4 Grand Challenges of the 90's: Multi-disciplinary computations

The discussion in the previous section was restricted to prediction of the external flow about an advanced fighter concept. As explained in subsection 2.1 the Grand Challenge computations of the 90's will be multi-disciplinary, combining computational techniques useful in analyzing a number of individual areas such as structures, controls, and acoustics, in addition to the baseline CFD simulations. It is possible in all these areas to derive estimates for the performance requirements. These estimates are given in Table 10 as multiplicative factors of additional requirements over the single-discipline baseline CFD simulation.

Table 10: **Increase in memory and CPU requirements over baseline CFD simulation**

Discipline	**Memory increase**	**CPU Time increase**
Structural Dynamics		
modal analysis	$\times$ 1	$\times$ 2
FEM analysis	$\times$ 2	$\times$ 2
thermal analysis	$\times$ 2	$\times$ 2
Propulsion		
inlet/nozzle simulation	$\times$ 2	$\times$ 2
engine performance deck	$\times$ 2	$\times$ 2
combustion model, e.g. scramjet	$\times$ 4	$\times$ 10
turbojet engine (full sim.)	$\times$ 10-100	$\times$ 10-100
Controls		
control law integration	$\times$ 1	$\times$ 1
control surface aerodynamics	$\times$ 2	$\times$ 2
thrust vector control	$\times$ 2	$\times$ 2
control jets	$\times$ 2	$\times$ 2
Acoustics	$\times$ 10	$\times$ 10
Numerical Optimization Design	$\times$ 2	$\times$ 10-100

It is clear that computational resource requirements can increase rapidly for multi-disciplinary computations. If the corresponding factors for multi-disciplinary V/STOL aircraft design are extracted from Table 10, and combined with the numbers for the baseline external aerodynamics prediction, quickly Gword and near TFLOP requirements arise. The details are given in Table 11 [36].

Table 11: **Flops and Run-time Memory Requirements for 5 Hour Run.**

	Mwords	**GFLOPS**
Base CFD	200	60
Structural		
thermal analysis	$\times$ 2	$\times$ 2
Propulsion		
inlet/nozzle simulations	$\times$ 2	$\times$ 2
engine performance deck	$\times$ 2	$\times$ 2
Controls		
control law integration	$\times$ 1	$\times$ 1
thrust vector control	$\times$ 2	$\times$ 2
Total	2000	600

It should be noted that the factors in Table 10 are based on the assumption that the physical frequencies introduced because of the multi-disciplinary integration can be resolved with the time steps required by the aerodynamics simulation. Additional compute time may be required if the multi-disciplinary system exhibits higher frequency modes which must be resolved.

7 Conclusions

Table 12: **Summary of Performance on Parallel Machines**
(fraction of single processor Cray Y-MP performance)

Application	CM-2 32K proc.	iPSC/860 128 proc.
Structured grid (LU)	0.76	0.91
Unstructured grid*	0.91	2.05
Particle methods	0.50	2.50

$^{*)}$ result for 8K processors

The results in Table 12 summarize most of the efforts discussed in this paper. They demonstrate that on current generation parallel machines performance on actual CFD applications is obtained which is approximately equivalent to the performance of one to two processors of a Cray Y-MP. All applications considered here are not immediately parallelized and both on SIMD and MIMD machines considerable effort must be expended in order to obtain an efficient implementation. It has been demonstrated by the results obtained at NASA Ames that this can be done, and that super computer level performance can be obtained on current generation parallel machines. Furthermore the particle simulation code on the CM-2 is a production code currently used to obtain production results (see [14]). The iPSC/860 implementation should be in production use by the end of 1991.

Our results also demonstrate another feature which has been found across a number of applications at NASA Ames: massively parallel machines quite often obtain only a fraction of their peak performance on realistic applications. In the applications considered here, there are at least two requirements which form the primary impediment in obtaining the peak realizable performance from these machines. One of these requirements is for unstructured, general communication with low latency and high bandwidth, which arises both in the unstructured application and in particle codes. The other requirement is for high bandwidth for a global exchange as it occurs in array transposition. This is important for the structured grid problems, since three dimensional arrays have to be accessed in the direction of the three different grid planes. Neither the CM-2 nor the iPSC/860 deliver the communication bandwidth necessary for these CFD applications. Experience has shown that CFD applications require on the order of one memory reference per floating point operation and a balanced system should have a memory bandwidth comparable to its floating point performance. In these terms, current parallel systems deliver only a fraction of the required bandwidth.

It spite of these promising results all the high expectations for parallel machines have not yet been met. In particular we do not believe that there is or will be a 10 GFLOPS sustained performance parallel machine available before 1993. Even on the new Intel Touchstone Delta machine the applications described here will perform at best in the 1 - 2 GFLOPS [2] range. The question then is (to quote Tom Lasinski [24]): "So why are we still bullish on parallel computers?". The answer, also given in [24], is: "Parallel computers have a tremendous growth potential." Even if we assume that current machine such as the CM-2 and the Intel iPSC/860 achieve only 1/50 of their peak performance on parallel CFD applications, we can extrapolate to the near future and predict a great increase in performance. In 1995 a machine based on commodity microprocessors with 160 MHz, three results per clock period, and 2048 processors is entirely likely and feasible. Such a machine would have approximately 1 TFLOPS peak performance. Even at 1/50 of this peak performance, we would be able to perform CFD calculations at a level of 20 GFLOPS sustained. With improvements in hardware, software, and algorithms we should be able to obtain even better performance.

As outlined in Section 6, these significant increases in compute power are essential to accomplishing the computational Grand Challenges of the 1990's. Even detailed single discipline computations will require GFLOP performance, with the multi-disciplinary simulations becoming just feasible on the most advanced systems of the 1990's.

Acknowledgment. We wish to thank our colleagues with the NAS Applied Research Branch, whose work has been discussed here: D. Bailey, E. Barszcz, R. Fatoohi, T. Lasinski, C. Levit, V. Venkatakrishnan, and S. Weeratunga. We also thank T. Barth, D. Jespersen, T. Abeloff, K. Chawla, M. Smith, (all NASA Ames), S. Hammond, and R. Schreiber at RIACS, J. McDonald

[2]Researchers at NAS are aware that there are claims about multiple GFLOPS performance on these systems. However, the discriminating reader is encouraged to study the recent note by D. Bailey on "Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers" [6].

(Maspar Corp.) and P. Frederickson (Cray Research) for their contributions to this summary report.

References

[1] P. A. Abeloff, W. R. Van Dalsem, and F. C. Dougherty. Thermal Interaction between an Impinging Hot Jet and a Conducting Solid Surface. 1990. AIAA Paper 90-0299.

[2] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga. The NAS Parallel Benchmarks. *Int. J. of Supercomputer Applications*, 5(3):63 – 73, 1991.

[3] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga. The NAS Parallel Benchmarks - Summary and Preliminary Results. In *Proceedings of Supercomputing '91, Albuquerque, New Mexico*, pages 158 – 165, Los Alamitos, California, 1991. IEEE Computer Society Press.

[4] D. Bailey, E. Barszcz, R. Fatoohi, H. Simon, and S. Weeratunga. Performance Results on the Intel Touchstone Gamma Prototype. In David W. Walker and Quentin F. Stout, editors, *Proceedings of the Fifth Distributed Memory Computing Conference*, pages 1236 – 1246, Los Alamitos, California, 1990. IEEE Computer Society Press.

[5] D. H. Bailey. Experience with Parallel Computers at NASA Ames. Technical Report RNR-91-07, NASA Ames Research Center, Moffett Field, CA 94035, February 1991.

[6] D. H. Bailey. Twelve Ways to Fool the Masses when Giving Performance Results on Parallel Computers. *Supercomputing Review*, pages 54 – 55, August 1991.

[7] D. H. Bailey, J. Barton, T. Lasinski, and H. Simon (editors). The NAS Parallel Benchmarks. Technical Report RNR-91-02, NASA Ames Research Center, Moffett Field, CA 94035, January 1991.

[8] E. Barszcz. One Year with an iPSC/860. Technical Report RNR-91-01, NASA Ames Research Center, Moffett Field, CA 94035, January 1991.

[9] E. Barszcz and K. Chawla. F3D on the CM-2. In T. Pulliam, editor, *Compendium of Abstracts, NASA CFD Conference, March 1991*, pages 56 – 57. NASA Office of Aeronautics Exploration and Technology, March 1991.

[10] T.J. Barth and D.C. Jespersen. The Design and Application of Upwind Schemes on Unstructured Meshes. 1989. AIAA Paper 89-0366.

[11] J. A. Benek, P. G. Buning, and J. L. Steger. A 3-d Chimera Grid Embedding Technique. 1985. AIAA Paper 85-1523.

[12] J. A. Benek, T. L. Donegan, and N. E. Suhs. Extended Chimera Grid Embedding Scheme with Application to Viscous Flows. 1987. AIAA Paper 87-1126-CP.

[13] L. Dagum. On the suitability of the Connection Machine for direct particle simulation. Technical Report 90.26, RIACS, NASA Ames Research Center, Moffett Field, CA 94035, June 1990.

[14] L. Dagum. Lip Leakage Flow Simulation for the Gravity Probe B Gas Spinup. 1992. AIAA Paper 92-0559.

[15] L. Dagum. Data Parallel Sorting for Particle Simulation. *Concurrency: Practice and Experience*, (to appear), May 1992.

[16] L. Dagum. Three Dimensional Direct Particle Simulation on the Connection Machine. *J. Thermophysics*, (to appear) 1992.

[17] E. Denning Dahl. Mapping and compiled communication on the Connection Machine system. In David W. Walker and Quentin F. Stout, editors, *Proceedings of the Fifth Distributed Memory Computing Conference*, pages 756 – 766, Los Alamitos, California, 1990. IEEE Computer Society Press.

[18] J. W. Fozard. *The Jet V/STOL Harrier - An Evoluntionary Revolution in Tactical AirPower*. 1978. AIAA Professional Study Series.

[19] S. Hammond. *Mapping Unstructured Grid Computations to Massively Parallel Computers*. PhD thesis, 1992.

[20] S. Hammond and T.J. Barth. On a massively parallel Euler solver for unstructured grids. In Horst D. Simon, editor, *Research Directions in Parallel CFD.* MIT Press, Cambridge(to appear), 1991.

[21] S. Hammond and R. Schreiber. Mapping Unstructured Grid Problems to the Connection Machine. Technical Report 90.22, RIACS, NASA Ames Research Center, Moffett Field, CA 94035, October 1990.

[22] T.L. Holst, M. D. Salas, and R. W. Claus. The NASA Computational Aerosciences Arogram - Toward Teraflop Computing. 1992. AIAA Paper 92-0558.

[23] Intel Corporation. *i860 64-Bit Microprocessor Programmer's Reference Manual.* Santa Clara, California, 1990.

[24] T. A. Lasinski. Massively parallel computing at NAS: Opportunities and experiences. Presentation in the NAS User TeleVideo Seminar, March 1991.

[25] K. Lee. On the Floating Point Performance of the i860 Microprocessor. Technical Report RNR-90-019, NASA Ames Research Center, Moffett Field, CA 94035, 1990.

[26] C. Levit and D. Jespersen. Explicit and Implicit Solution of the Navier-Stokes Equations on a Massively Parallel Computer. Technical report, NASA Ames Research Center, Moffett Field, CA, 1988.

[27] C. Levit and D. Jespersen. A Computational Fluid Dynamics Algorithm on a Massively Parallel Computer. *Int. J. Supercomputer Appl.*, 3(4):9 – 27, 1989.

[28] C. Levit and D. Jespersen. Numerical Simulation of a Flow Past a Tapered Cylinder. Technical Report RNR-90-21, NASA Ames Research Center, Moffett Field, CA 94035, October 1990.

[29] Creon Levit. Grid communication on the Connection Machine: Analysis, performance, improvements. In H. D. Simon, editor, *Scientific Applications of the Connection Machine*, pages 316 – 332. World Scientific, 1989.

[30] Z. C. Lou. A Summary of CFS I/O Tests. Technical Report RNR-90-20, NASA Ames Research Center, Moffett Field, CA 94035, October 1990.

[31] F.E. Lumpkin. *Development and Evaluation of Continuum Models for Translational-Rotational Nonequilibrium.* PhD thesis, Stanford University, Dept. of Aeronautics and Astronautics, Stanford CA 94305, April 1990.

[32] J. D. McDonald. *A Computationally Efficient Particle Simulation Method Suited to Vector Computer Architectures.* PhD thesis, Stanford University, Dept. of Aeronautics and Astronautics, Stanford CA 94305, December 1989.

[33] J. D. McDonald. Particle Simulation in a Multiprocessor Environment. 1991. AIAA Paper 91-1366

[34] V. Naik, N. Decker, and M. Nicoules. Implicit CFD Applications on Message Passing Multiprocessor Systems. In Horst D. Simon, editor, *Parallel CFD - Implementations and Results Using Parallel Computers*, pages 103 – 132. MIT Press, Cambridge, Mass., 1992.

[35] NAS Systems Division, NASA Ames Research Center. *Numerical Aerodynamic Simulation Program Plan*, October 1988.

[36] National Aeronautics and Space Administration. *Project Plan for the Computational Aerosciences Project of the NASA High Performance Computing and Communications Program (NASA HPCCP/CAS)*, May 1991.

[37] A. Pothen, H. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Mat. Anal. Appl.*, 11(3):430 – 452, 1990.

[38] T. H. Pulliam. Efficient Solution Methods for the Navier-Stokes Equations. Lecture Notes for The Von Karman Institute for Fluid Dynamics Lecture Series, Jan. 20 - 24, 1986.

[39] James S. Ryan. Concurrent File System (CFS) I/O for CFD Applications. August 1991.

[40] R. Schreiber. An Assessment of the Connection Machine. Technical Report 90.40, RIACS, NASA Ames Research Center, Moffett Field, CA 94035, June 1990.

[41] H. D. Simon. Partitioning of unstructured problems for parallel processsing. *Computing Systems in Engineering*, 2(2/3):135 – 148, 1991.

[42] Horst D. Simon. Massive Parallelism at NAS. In *Proceedings of the Supercomputing USA Pacific 1991 Conference*, pages 100 – 103, Santa Clara, California, June 1991. Meridian Pacific Group.

[43] Horst D. Simon and Leonardo Dagum. Experience in using SIMD and MIMD parallelism for computational fluid dynamics. In R. Vichnevetsky and J.J.H. Miller, editors, *Proceedings of the 13th IMACS World Congress on Computational and Applied Mathematics*, pages 693 – 697. Criterion Press, Dublin, Ireland, 1991.

[44] M. H. Smith, K. Chawla, and W. R. Van Dalsem. Numerical Simulation of a Complete STOVL Aircraft in Ground Effect. 1991. AIAA Paper 91-3293.

[45] W. R. Van Dalsem, K. Chawla, M. H. Smith, and P. A. Abeloff. Numerical Simulation of Powered Lift Flows. In *Proceedings of the International Powered-Lift Conference*, London, England, August 1990. The Royal Aeronautical Society.

[46] V. Venkatakrishnan, H. Simon, and T. Barth. A MIMD implementation of a parallel Euler solver for unstructured grids. Technical Report RNR-91-24, NASA Ames Research Center, Moffett Field, CA 94035, September 1991.

[47] S. Yoon, D. Kwak, and L. Chang. LU-SGS Implicit Algorithm for Implicit Three Dimensional Navier-Stokes Equations with Source Term. 1989. AIAA Paper 89-1964-CP.