

# Experience with Massive Parallelism for CFD Applications at NASA Ames Research Center

Horst D. Simon

Computer Sciences Corporation at NASA Ames Research Center  
Mail Stop T045-1, Moffett Field, CA 94035  
U.S.A.

## Zusammenfassung

Eines der Hauptziele des Applied Research Branch in der Numerical Aerodynamic Simulation (NAS) Systems Division am NASA Ames Research Center ist die beschleunigte Einführung von parallelen Hochleistungsrechnern in ein produktionsorientiertes Rechenzentrum. In dieser Arbeit werden die Zielrichtungen des NAS Projekts in Bezug auf Parallelrechner dargestellt. Weiterhin werden die Erfahrungen mit experimentellen Parallelrechnern im NAS Applied Research Branch zusammengefasst. Im Einzelnen wird über Ergebnisse mit Anwendungen in der Strömungsmechanik auf der Connection Machine CM-2 und dem Intel iPSC/860 berichtet. Ergebnisse von Berechnungen mit unstrukturierten Gittern und mit Teilchensimulationen werden dargestellt. Angesichts der Erfahrungen bei NASA wird die zukünftige Entwicklung von Parallelrechnern für die Anwendungen in der Strömungsmechanik diskutiert.

## 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 [30]). 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 [6, 1, 2]), 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. 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 [4]. More details about the NASA Computational Aerosciences Program with more emphasis on the applications are given in [17, 36]. This is an abbreviated version of [36].

## 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 [24].

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 concurrent users. 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 [34]. 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 new 64 bit i860 microprocessor by Intel [18]. 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. This reduction is mainly obtained through the increased speed of the i860 on the iPSC/860 machine, when compared to the Intel 386/387 on the iPSC/2. The improved latency is thus mainly a product of faster execution of the message passing software on the i860.

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. This system handles compilation and linking of source programs, as well as loading the executable code into the hypercube nodes and initiating execution. At present the SRM is a serious bottleneck in the system, due to its slowness in compiling and linking user codes. For example, the compilation of a moderate-sized application program often requires 30 minutes or more, even with no optimization options and no other users on the system.

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 [3]. A more recent summary is given by Barszcz in [7]. As documented in [7] from an overall systems aspect the main bottleneck has been 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 [20]. 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 [25] and Ryan [33], 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 [21, 22], which was recently extended to three dimensions [23]. Their implementation is based on the successful ARC2D and ARC3D codes developed by Pulliam [32]. Work by Barszcz and Chawla [8] 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 [3]), 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 [6].

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 direc-

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

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

tions. 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<sup>1</sup>.

The first of the simulated applications is the LU benchmark. In this benchmark, a regular-sparse, block ( $5 \times 5$ ) 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 NASA Ames by the code INS3D-LU [38]. 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 [32] 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 ( $5 \times 5$ ) block size are solved (for a related discussion of the parallel implementation of ARC3D see also [29]).

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.

---

<sup>1</sup>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.

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 2.** Results for the SP 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

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

#### 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[9]. This solver has been implemented on the CM-2 by Hammond and Barth [15], and on the Intel iPSC/860 by Venkatakrisnan, Simon, and Barth [37].

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 [9, 15]. 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 [15] used a novel partitioning of the problem which minimizes the computation and communication costs on a massively parallel computer. The following description follows [15] 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 [16] and a "Communication Compiler" developed for the CM-2 by Dahl [13]. 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 [14] 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. [37] employ a new algorithm for the graph partitioning problem, which has been discussed recently by Simon [35], 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 [31]. Detailed investigations

and comparisons to other strategies (cf. [35]) 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

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

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

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

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.

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

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

## 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 [26]), 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 [27] and [10]

## 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 [10] 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 [10] 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 [12] describes different sorting algorithms suitable for this purpose. The fastest of these 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 tremendously reduce the amount of communication necessary to re-order the particles.

These algorithms have been implemented in a two-dimensional particle simulation running on the CM-2. At the time of implementation, the CM-2 at NASA Ames had only 64K bits of memory per processor which was insufficient to warrant a three-dimensional implementation. Furthermore, the slicewise model of the machine did not exist and the machine had the slower 32-bit Weitek's which did not carry out any integer arithmetic. Nonetheless, with this smaller amount of memory and fieldwise implementation, the code was capable of simulating over  $2.0 \times 10^6$  particles in a grid with  $6.0 \times 10^4$  at a rate of  $2.0 \mu\text{sec}/\text{particle}/\text{time step}$  using all 32K processors (see [10]). By comparison, a fully vectorized equivalent simulation on a single processor of the Cray YMP runs at  $1.0 \mu\text{sec}/\text{particle}/\text{time step}$  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). Currently, work is being carried out to extend the simulation to three dimensions using a parallel decomposition which takes full advantage of the slicewise model of the machine.

## 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 [28] implementation, the spatial domain of the simulation is divided into a number of sub-domains or regions 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 region. This allows the representation of simulated space (i.e. the geometry definition) to be distributed along with the particles. The aim is 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 modeling of particles, 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. The performance of the code on a 3D heat bath is given in Table 6.

Processors	$\mu\text{s}/\text{prt}/\text{step}$	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

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

At the present time the domain decomposition is static, however work is being carried out to allow dynamic domain decomposition thus permitting a good load balance to exist throughout a calculation. The geometry and spatial decomposition of the heat bath simulation *exaggerated* the area to volume ratio of the regions in order to more closely approximate the communication expected in a real application with dynamic load balancing. The most promising feature of these results is the linear speed up obtained, indicating that the performance of the code should continue to increase with increasing numbers of processors.

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.

Machine	Processors	$\mu\text{secs}/\text{particle}/\text{step}$	MFLOPS
Cray 2	1	2.0	43
Cray Y-MP	1	1.0	86
Intel iPSC/860	128	0.4	215
CM-2 (32 bit)	32768	2.0	43

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

## 6 Conclusions

The results in Table 8 summarize most of the efforts discussed in this paper. They demonstrate that on current generation parallel machines performance on actual CFD applications is obtained

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

\*) CM result for 8K processors

**Table 8.** Summary of Performance on Parallel Machines (fraction of single processor Cray Y-MP performance).

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 [11]). 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 <sup>2</sup> range. The question then is (to quote Tom Lasinski [19]): "So why are we still bullish on parallel computers?". The answer, also given in [19], 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.

<sup>2</sup>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" [5].

As outlined in [36], 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.** I wish to thank my colleagues with the NAS Applied Research Branch, whose work has been discussed here: D. Bailey, E. Barszcz, L. Dagum, R. Fatoohi, T. Lasinski, C. Levit, V. Venkatakrisnan, and S. Weeratunga. I also thank T. Barth, D. Jespersen, T. Abeloff, K. Chawla, M. Smith, W. Van Dalsem (all NASA Ames), S. Hammond, and R. Schreiber at RIACS, J. McDonald (MassPar) and P. Frederickson (Cray Research) for their contributions to this summary report.

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.

## References

- [1] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrisnan, and S. Weeratunga. The NAS Parallel Benchmarks. *Int. J. of Supercomputer Applications*, 5(3):63 – 73, 1991.
- [2] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrisnan, 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.
- [3] 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.
- [4] 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.
- [5] D. H. Bailey. Twelve ways to fool the masses when giving performance results on parallel computers. *Supercomputing Review*, pages 54 – 55, August 1991.
- [6] 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.
- [7] E. Barszcz. One year with an iPSC/860. Technical Report RNR-91-01, NASA Ames Research Center, Moffett Field, CA 94035, January 1991.
- [8] 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.
- [9] T.J. Barth and D.C. Jespersen. The design and application of upwind schemes on unstructured meshes. In *Proceedings, 27th Aerospace Sciences Meeting*, January 1989. Paper AIAA 89-0366.

- [10] 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.
- [11] L. Dagum. Lip leakage flow simulation for the gravity probe b gas spinup using psicm. Technical Report RNR-91-10, NASA Ames Research Center, Moffett Field, CA 94035, March 1991.
- [12] Leonardo Dagum. Sorting for Particle Flow Simulation on the Connection Machine. In Horst D. Simon, editor, *Parallel CFD - Implementations and Results Using Parallel Computers*, pages 245 – 270. MIT Press, Cambridge, Mass., 1992.
- [13] 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.
- [14] S. Hammond. *Mapping Unstructured Grid Computations to Massively Parallel Computers*. PhD thesis, RPI, 1992.
- [15] S. Hammond and T. Barth. On a Massively Parallel Euler Solver for Unstructured Grids. In Horst D. Simon, editor, *Parallel CFD - Implementations and Results Using Parallel Computers*, pages 55 – 70. MIT Press, Cambridge, Mass., 1992.
- [16] 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.
- [17] T.L. Holst, M. D. Salas, and R. W. Claus. The NASA computational aerosciences program - toward teraflop computing. In *Proceedings of the 30th Aerospace Sciences Meeting, Reno, NV, Washington, D.C., 1992*. American Institute of Aeronautics and Astronautics. AIAA Paper 92-0558.
- [18] Intel Corporation. *i860 64-Bit Microprocessor Programmer's Reference Manual*. Santa Clara, California, 1990.
- [19] T. A. Lasinski. Massively parallel computing at nas: Opportunities and experiences. Presentation in the NAS User TeleVideo Seminar, March 1991.
- [20] 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.
- [21] 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.
- [22] C. Levit and D. Jespersen. A computational fluid dynamics algorithm on a massively parallel computer. *Int. J. Supercomputer Appl.*, 3(4):9 – 27, 1989.
- [23] 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.
- [24] 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.

- [25] 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.
- [26] 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.
- [27] 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.
- [28] J. D. McDonald. Particle simulation in a multiprocessor environment. Technical Report RNR-91-02, NASA Ames Research Center, Moffett Field, CA 94035, January 1991.
- [29] 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.
- [30] NAS Systems Division, NASA Ames Research Center. *Numerical Aerodynamic Simulation Program Plan*, October 1988.
- [31] A. Pothén, H. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Mat. Anal. Appl.*, 11(3):430 – 452, 1990.
- [32] 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.
- [33] James S. Ryan. Concurrent File System (CFS) I/O for CFD Applications. August 1991.
- [34] R. Schreiber. An assessment of the Connection Machine. In H. D. Simon, editor, *Scientific Applications of the Connection Machine, 2nd edition*, pages 379 – 390. World Scientific, 1992.
- [35] H. D. Simon. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2(2/3):135 – 148, 1991.
- [36] Horst D. Simon, William Van Dalsem, and Leonardo Dagum. Parallel CFD: Current Status and Future Requirements. In Horst D. Simon, editor, *Parallel CFD - Implementations and Results Using Parallel Computers*, pages 1 – 28. MIT Press, Cambridge, Mass., 1992.
- [37] 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.
- [38] S. Yoon, D. Kwak, and L. Chang. LU-SGS implicit algorithm for implicit three dimensional Navier-Stokes equations with source term. Washington, D.C., 1989. American Institute of Aeronautics and Astronautics. AIAA Paper 89-1964-CP.