

FUTURE DIRECTIONS IN COMPUTING AND CFD*

F. Ron Bailey[†]

and

Horst D. Simon[‡]

NASA Ames Research Center, Moffett Field, California 94035

Abstract.

In recent years CFD on massively parallel machines has become a reality. In this paper we summarize some recent trends both in high performance computing, and in CFD using parallel machines. We discuss the long term computational requirements for accomplishing some of the large scale problems in computational aerosciences, and current hardware and architecture trends. We present performance results obtained from the implementation of some CFD applications on the Connection Machine CM-2 and the Intel iPSC/860 at NASA Ames Research Center. Finally 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.

1. Introduction

In the nineteen seventies and eighties significant advances occurred in computational fluid dynamics (CFD) as a result of improvements in processing speed and storage capacity of new

generations of computers as well as improvements in algorithms. As a result CFD has become an ever more powerful tool in the design of aerospace systems. The high computational demands of aerospace applications have been the driving force behind the recent rapid development of CFD. Moreover, CFD has been an equally successful modeling tool in a variety of other fields, such as automotive engineering. Now, in the 1990's it has become apparent that conventional supercomputers cannot sustain CFD's continued high rate of advancement. Thus CFD has reached a critical juncture, since it is becoming more and more apparent that future growth in computational speed will be increasingly the result of parallel processing technology.

In the last several years a wide variety of parallel machines have become available for exploring the issues of using parallelism in scientific computing in general and CFD in particular. Most of the early ("zero-th generation") machines that appeared between 1983 and 1987 were rather experimental in nature, and served mainly for research investigations in areas such as algorithms, languages, and operating systems for parallel computing. In 1988 and 1989 several members of a first generation of *parallel supercomputers* became available. We want to use the term "supercomputer" here because these parallel supercomputers such as the Connection Machine CM-2, the Intel iPSC/860, the NCUBE2, and others are in their larger configurations are comparable both in memory and peak computational speed to the performance of the most powerful conventional supercomputers, e.g., the Cray Y-MP. However, it is well

*Copyright © 1992 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Government purposes. All other rights are reserved by the copyright owner.

[†]Director of Aerophysics, Fellow AIAA

[‡]Research Scientist. The author is an employee of Computer Sciences Corporation. This work is supported through NASA Contract NAS 2-12961.

known that these machines are still very deficient in their systems aspects, for example in their ability to handle a large number of users. Now, in 1992, we are at the threshold of a second generation of parallel supercomputers, which offer considerable improvements in computational power over the previous generation as well as an improved software and user environment.

Because of their considerable potential computational power, parallel supercomputers are increasingly considered as an alternative to the more conventional supercomputers based on a small number of powerful vector processors. Researchers who want to advance the state of the art in CFD are thus faced with the issue of taking a serious look at parallel processing, in particular at the massively parallel, distributed memory machines, which promise to lead to the most advances in computational speed in the 1990s.

In spite of many difficulties initial performance results for parallel machines are extremely encouraging. In some instances (see for example the work at UTRC described by Egolf¹) researchers have demonstrated the potential of massively parallel systems for CFD and are moving (or will soon move) research applications into the engineering community for production use. In this paper we will present the case for massive parallelism by first demonstrating the need for more computational power arising from the requirements of Grand Challenge Applications. 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 resources which could become available in the near future. A grand challenge for CFD will be discussed in section 2. We then demonstrate in section 3 that the performance increases of microprocessors indicate that massively parallel systems based on commodity chips will be the technology leading to Teraflops level performance. Finally we will summarize the experience with parallel machines at NASA Ames Research Center, and present a critique of the current state of par-

allel computing for computational areoscience applications.

2. The Case for More Computer Power

2.1. Requirements for Design

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.²

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 1.

Table 1: Requirements for Flow Simulation

Solution Mode	Turn around time in hours	Required Performance in GFLOPS
Proof of concept	1000 – 100	0.3 – 3
Design	10 – 1	30 – 300
Automated Design	0.1 – 0.01	3,000 – 30,000

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.

2.2. Requirements for Multidisciplinary Analysis

The CFD 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 as multiplicative factors of additional requirements over the single-discipline baseline CFD simulation.³

Because these factors are multiplicative computational resource requirements can increase rapidly for multi-disciplinary computations. In order to demonstrate this point, the corresponding factors for multi-disciplinary

V/STOL aircraft design have been established and combined with the numbers for the baseline external aerodynamics prediction.³ Very quickly Gword and near TFLOP requirements arise. The details are given in Table 2.³

Table 2: Flops and Run-time Memory Requirements for 5 Hour Run.

	Mwords	GFLOPS
Base CFD	200	60
Structural thermal analysis	× 2	× 2
Propulsion inlet/nozzle simulations	× 2	× 2
engine perf. deck	× 2	× 2
Controls cntrl. law integr. integration	× 1	× 1
thrust vector control	× 2	× 2
Total	2000	600

It should be noted that the factors in Table 2 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.

3. Why Massively Parallel Computers?

Over the last three to four years there has been a dramatic increase in the level of acceptance of massively parallel machines by the scientific user community. Users of massively parallel machines are no longer seen as a “lunatic fringe”, but rather as people on the forefront of scientific computing in 1990s. Public opinion on parallel machines has almost completely reversed itself. Today, in 1992, skepticism about the practical use of parallel processing has been replaced by almost universal acceptance of the belief that a massively parallel machine will deliver performance at the Teraflops level by the

end of the decade, and that almost all Grand Challenge scientific computing will be carried out on massively parallel machines.

The interaction between computer architecture and hardware advances on one side and the requirements of large scale scientific applications on the other side are multifaceted and complex. Currently the rapid growth and the increased acceptance of massively parallel computers might be equally attributed to the increased demands of the scientific user community as exemplified in the Grand Challenge Problems⁴, which have been forcing us to consider parallel computers more seriously.

Clearly there have been some fundamental changes in the relative performance of VLSI based microprocessors and custom designed vector processors used in the more traditional supercomputers. Hennessy and Jouppi estimate that microprocessors have been improving in CPU performance at a rate of between 1.5 to 2 times per year during the last six to seven years, whereas improvement rates for mainframes were only about 25 % per year.⁵ The authors also quote two major factors which have contributed to the high growth rate of microprocessor performance:

- The dramatic increase of the number of transistors available on a chip.
- Architectural advances, including the use of RISC ideas, pipelining, and caches.

In Figure 1 these trends are summarized by comparing the performance of both microprocessors and of single processor of vector supercomputers of the last decade. Obviously the single processor performance of vector supercomputers has improved only by about an order of magnitude within the last decade. In contrast microprocessors have experienced a dramatic jump in their floating point capabilities around 1987. If we restrict ourselves only to the Intel family of microprocessor alone, we see about two orders of magnitude improvement from the Intel 80387 to the Intel i860. This large performance again can be directly

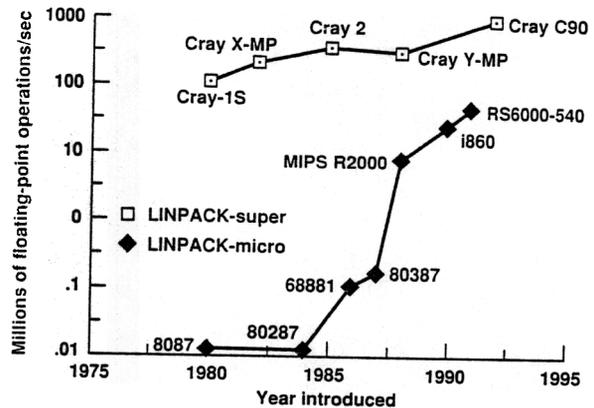


Figure 1: Comparison of Supercomputer and Microprocessor Floating Point Performance

correlated to the performance of the corresponding parallel systems. The current Intel iPSC/860, which uses the i860 processor, is widely regarded as the first true supercomputer produced by Intel, whereas its predecessor the iPSC/2 was at best an experimental research machine.

Estimates and studies investigating future trends in microprocessor performance are highly optimistic. A recent study estimates that by the year 2000 a single chip is expected to have four processors with a combined performance of about 1 GFLOPS.⁶ Thus the rate of performance increases for microprocessors is expected to continue to be higher than traditional supercomputers.

Even though these arguments do not address other important aspects of supercomputer systems, such as bandwidth to memory, memory size, I/O devices etc. (see Lundstrom⁷ for a more detailed discussion), we believe that microprocessors will be the pacing item in the future development of massively parallel systems. All indications are therefore that "the attack of the killer micros" is going to be highly successful.⁸

4. Experience with Massively Parallel Computers

The availability of the parallel testbed sys-

tems at NASA's Numerical Aerodynamic Simulation (NAS) facility since 1989 led to a number of CFD and computational aerospace applications to be implemented on these parallel machines at the NAS facility. Here we briefly survey the work carried out by researchers at the Applied Research Branch at NAS.^{3,9,10,11,12,13,14,15,16} Some of the early work on the Connection Machine is summarized in several papers in a book.¹⁷

4.1. NAS Parallel Benchmarks

The NAS Parallel Benchmarks have been developed at NASA Ames Research Center to study the performance of parallel supercomputers.^{9,11} They represent a novel approach to benchmarking in that the eight benchmark problems are specified in a "pencil and paper" fashion. In other words, the complete details of the problem to be solved are given in a technical document, and except for a few restrictions, benchmarkers are free to select the language constructs and implementation techniques best suited for a particular system.

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 mul-

tiplied, 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. (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 based on a two factor scheme.)

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 NASA Ames by the code INS3D-LU.¹⁸ 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 at NASA Ames Research Center.¹⁹

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. SP and BT are similar in many

Table 3: Results for the LU Simulated CFD Application

System	No. Proc.	Time/Iter. (sec)	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 4: Results for the SP Simulated CFD Application

System	No. Proc.	Time/Iter. (sec)	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

pects, but there is a fundamental difference with respect to the communication to computation ratio.

Performance figures for the three simulated CFD applications are shown in Tables 3, 4 and 5. 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 using the hardware performance monitor on the Cray Y-MP. Generally it was found that the performance of the parallel machines at NASA Ames is in the range of about one to two processors of a Cray Y-MP.

Table 5: Results for the BT Simulated CFD Application

System	No. Proc.	Time/Iter. (sec)	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

4.2. 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 which was recently extended to three dimensions.^{13,14,15} Their implementation is based on the successful ARC2D and ARC3D codes developed by Pulliam.¹⁹ Barszcz and Chawla have implemented F3D, a successor code to ARC3D, on the CM-2.¹² On the iPSC/860 Weeratunga has implemented ARC2D, ARC3D and F3D.¹⁰ Weeratunga also has developed three simulated CFD applications based on structured grid flow solvers for the NAS Parallel Benchmarks, which are described in the NAS Parallel Benchmarks.¹¹ It is important to note that he was able to develop the parallel versions of ARC3D and F3D fairly quickly based on the experience and the existing implementation of the simulated application in the NAS Parallel Benchmarks. The NAS Parallel Benchmarks thus have an important second use beyond the performance evaluation of the massively parallel systems: they have proven as a useful template for the development of structured grid CFD applications on parallel machines.

Table 6: ARC2D Performance (MFLOPS)

Problem Size	Number of Processors				
	2	4	8	16	32
192 × 64	5.3	8.5	12.7	16.8	
256 × 80	5.5	9.2	16.1	19.6	
320 × 128		9.9	16.8	25.5	33.8

Table 7: ARC2D Performance (Efficiency)

Problem Size	Number of Processors				
	2	4	8	16	32
192 × 64	92%	74%	55%	36%	
256 × 80	94%	79%	61%	42%	
320 × 128		85%	72%	55%	36%

Details of their implementation on parallel machines as well as final performance results for ARC3D and F3D have not yet been published. In order to demonstrate the potential of parallel machines for production CFD applications, we quote here some earlier results on ARC2D.¹⁰

In Tables 6 and 7 the performance of ARC2D on the iPSC/860 for several different grid sizes is given. The rapid drop off in efficiencies also shows inadequate bandwidth and high latency for both in-processor and interprocessor data paths. These issues are addressed in future machines from Intel such as the Delta and Paragon machines, which substantially increase the interprocessor communication bandwidth. It should be noted that the efficiency remains high, if the grid size is scaled up with the number of processors.

4.3. 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.²⁰ This solver has been implemented on the CM-2 by Hammond and Barth, and on the Intel iPSC/860 by Venkatakrishnan, Simon, and Barth.^{16,21}

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 the papers by Hammond et al.^{20,21} 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.

4.3.1 SIMD Implementation of Unstructured Solver

For the implementation on the CM-2 Hammond and Barth used a novel partitioning of the problem which minimizes the computation and communication costs on a massively parallel computer.²¹ An important aspect of the work by Hammond and Barth is the use of fast communication. The SIMD implementation takes advantage of this by using a mapping technique developed by Hammond and Schreiber and a “Communication Compiler” developed for the CM-2 by Dahl.^{22,23} The use of these techniques results in 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 resulted in 43 seconds per 100 time steps, which is equivalent to 136 MFLOPS.²⁴ This does not include setup time.

4.3.2 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 Venkatakrisnan et al. employ a new algorithm for the graph partitioning problem, which has been discussed recently by Simon, and which is based on the computation of eigenvectors of the Laplacian matrix of a graph associated with the mesh.^{16,25} Details on the theoretical foundations of this strategy are given by Pothen et al.²⁶ Detailed investigations and comparisons to other strategies have shown that the spectral partitioning produces subdomains with the shortest boundary, and hence tends to minimize communication cost.²⁵

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 pre-computed, 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 8. The MFLOPS given are based on operation counts using the Cray hardware performance monitor. The efficiency E in % is computed as

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

Table 8: Performance of Unstructured Grid Code on the Intel iPSC/860

Proc.	sec/step	MFLOPS	E (%)
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 9, where all MFLOPS numbers are Cray Y-MP equivalent numbers.

Table 9: Performance Comparison of Unstructured Grid Code

Machine	Proc.	sec/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. Future Directions

The results in Table 10 summarize 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 Weeratunga's parallel ARC3D code has been used by Ryan for production runs in connection with HPCC applications. In so far unpublished work Ryan has demonstrated the accuracy and efficiency of the parallel code in several validation cases using single zone Euler computations. For example, for a grid of size $67 \times 60 \times 112$ involving 450,240 gridpoints for a HSCT wing-body combination, he has obtained on 128 processors of the Intel iPSC/860 a performance of about 10 microseconds per gridpoint per iteration, which is somewhat better than a single processor Cray Y-MP performance. This work demonstrates that we have reached about the same performance level on massively parallel machines as we have on traditional supercomputers, even when performing production type calculations.

Table 10: 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
ARC3D		1.99
Unstructured grid*	0.91	2.05

*) result for 8K processors

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 structured and unstructured applications. 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.

Another main issue in massively parallel computing is designing efficient algorithms. One of the key lessons learned is that simpler algorithms such as explicit methods for CFD may parallelize easily and thus lead to very high and impressive MFLOPS figures. However, their high speed cannot overcome (even in the massively parallel setting) their inherent weaknesses in term of convergence. Simply expressed, an explicit algorithm may run 10 times faster on a parallel machine, but still may take 100 times longer to converge. Experience with ARC3D and other implicit algorithms, which involve any of the popular approximate factorization schemes has shown that current implicit algorithms strain the communication capabilities of massively parallel machines, and therefore perform at speeds far below the peak of the parallel machines. The way out of this algorithmic dilemma may be via "hierarchical algorithms".²⁷ These algorithms include approaches such as multigrid schemes, hierarchical basis preconditioners, domain decomposition methods, and adaptive mesh refinement methods. What these algorithms have in common is the fact that just like implicit algorithms they provide immediate global dis-

Table 11: CFD Convection Diffusion Problem Implemented on NCUBE2

Solver Algorithm	Floating Point Ops.	CPU Time (sec)	MFLOPS
Jacobi	3.82×10^{12}	2124	1800
Gauss-Seidel	1.21×10^{12}	885	1365
Least Squares	2.59×10^{11}	185	1400
Multigrid	2.13×10^{09}	6.7	318

tribution of information. However they capture the global interactions in a more efficient way, which does not require the large amount of global communication as in traditional implicit algorithms. Therefore they appear to be the right compromise class of method for highly parallel architectures. This is demonstrated in the results in Table 11 which show the considerably faster performance of a multigrid algorithm.²⁸ These results also demonstrate the fallacy to measure the performance of a parallel machine by MFLOPS alone, when comparing different algorithms. Because of their potential for parallel machines, it is planned to increase research efforts into these type of hierarchical algorithms at the NAS Applied Research Branch in the near future.

These new algorithmic efforts together with 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 multidisciplinary simulations becoming just feasible on the most advanced systems of the 1990's.

Acknowledgement. Section 2 of this paper is abbreviated version of section 6 in the paper.³ We thank William R. Van Dalsem for providing us with his detailed analysis of future computational requirements.

References

[1] T. Alan Egolf. Scientific applications of the Connection Machine at the United Tech-

nologies Research Center. In H. D. Simon, editor, *Scientific Applications of the Connection Machine*, pages 38 – 63. World Scientific, 1989.

- [2] M. H. Smith, K. Chawla, and W. R. Van Dalsem. Numerical simulation of a complete STOVL aircraft in ground effect. AIAA Paper 91-3293.
- [3] 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.
- [4] Committee on Physical, Mathematical, and Engineering Sciences. *Grand Challenges: High Performance Computing and Communications*, Office of Science and Technology Policy, National Science Foundation, Washington, D.C., 1991.
- [5] J. Hennessy and N. Jouppi. Computer technology and architecture: An evolving interaction. *IEEE Computer*, 24(9):18 – 29, September 1991.
- [6] P. P. Gelsinger, P. A. Gargini, G. H. Parker, and A. Y. C. Yu. Microprocessors circa 2000. *IEEE Spectrum*, pages 43 – 47, October 1989.
- [7] S. F. Lundstrom. Supercomputing systems - a projection to 2000. *Computing Systems in Engineering*, 1(2 - 4):145 – 151, 1990.
- [8] E. D. Brooks and K. H. (eds.) Warren. The 1991 MPCCI yearly report: The attack of the killer micros. Technical Report UCRL-JC-107022, Lawrence Livermore National Laboratory, Livermore, CA 94550, July 1991.
- [9] 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.
- [10] 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.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] C. Levit and D. Jespersen. A computational fluid dynamics algorithm on a massively parallel computer. *Int. J. Supercomputer Appl.*, 3(4):9 – 27, 1989.
- [15] 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.
- [16] V. Venkatakrishnan, H. Simon, and T. Barth. A MIMD implementation of a parallel Euler solver for unstructured grids. *The Journal of Supercomputing*, 1992 (to appear).
- [17] H. D. Simon, editor. *Scientific Applications of the Connection Machine*, Singapore, 1989. World Scientific.
- [18] S. Yoon, D. Kwak, and L. Chang. LU-SGS implicit algorithm for implicit three dimensional Navier-Stokes equations with source term. AIAA Paper 89-1964-CP.
- [19] 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.
- [20] T.J. Barth and D.C. Jespersen. The design and application of upwind schemes on unstructured meshes. Paper AIAA 89-0366.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] S. Hammond. *Mapping Unstructured Grid Computations to Massively Parallel Computers*. PhD thesis, RPI, 1992.
- [25] H. D. Simon. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering*, 2(2/3):135 – 148, 1991.

- [26] 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.
- [27] Tony F. Chan. Hierarchical algorithms and architectures for parallel scientific computing. In S. Gallopoulos and A. Sameh, editors, *Proceedings of the 1990 International Conference on Supercomputing*, pages 318 – 329, New York, 1990. ACM Press.
- [28] J. N. Shadid and R. S. Tuminaro. Iterative Methods for Nonsymmetric Systems on MIMD Machines”. *Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing*, SIAM, Philadelphia, 1992, to appear.