NAS Parallel Benchmark Results

David H. Bailey and Eric Barszcz
NASA Ames Research Center

Leonardo Dagum and Horst D. Simon
Computer Sciences Corporation

The NAS Parallel Benchmarks focus on eight important aspects of highly parallel supercomputing for aerophysics applications. This article reports benchmark results for several computers, some of which have not been published before.

The Numerical Aerodynamic Simulation (NAS) Program at NASA Ames Research Center is dedicated to advancing the science of computational aerodynamics. One of the program's key goals is to demonstrate by the year 2000 an operational computing system that can simulate an entire aerospace vehicle system within several hours of computing time. We believe that solving this grand-challenge problem will require a computer system that can perform scientific computations at a sustained rate about 1,000 times faster than 1990-generation supercomputers. Such a computer will most likely employ hundreds or even thousands of processors operating in parallel.

Several commercial highly parallel systems have computing power roughly comparable to conventional supercomputers (and even greater on some special problems), but there is little reliable data on the performance of such systems on state-of-the-art computational aerophysics problems. In general, the science of performance evaluation has not kept pace with advances in parallel computer hardware and architecture. There is not even a generally accepted benchmark strategy for highly parallel supercomputers.

In our view, the best benchmarking approach for highly parallel supercomputers is the "paper and pencil" benchmark. The idea is to specify a set of problems only algorithmically. Even the input data must be specified only on paper. Naturally, we must specify the problem in sufficient detail that a unique solution exists, and the required output must be brief yet detailed enough to certify that the problem has been solved correctly.

Table 1. Standard operation counts and current Y-MP/1 megaflops rates.

Benchmark	Operation count	Mflops on Y-MP/1
Embarrassingly Parallel	2.668×10^{10}	211
Multigrid	3.905×10^{09}	176
Conjugate Gradient	1.508×10^{09}	127
3D FFT PDE	5.631×10^{09}	196
Integer Sort	7.812×10^{08}	68
Lower-Upper Diagonal	6.457×10^{10}	194
Scalar Pentadiagonal	1.020×10^{11}	216
Block Tridiagonal	1.813×10^{11}	229

But the implementation details should be left to the programmer as far as possible.

To this end, we have devised the NAS Parallel Benchmarks: a set of eight benchmark problems, each focusing on some important aspect of highly parallel supercomputing for aerophysics applications.[1] Some extension of Fortran or C is required for implementations, and reasonable limits are placed on the use of assembly code and the like, but programmers are otherwise free to use language constructs that give the best performance on the system being studied. The choice of data structures, processor allocation, and memory use are generally left to the discretion of the implementer.

The eight problems consist of five kernels and three simulated computational fluid dynamics (CFD) applications.[1,2] The kernels are relatively compact problems that can be implemented fairly readily and provide insight into the general levels of performance that can be expected for particular types of numerical computations.

The simulated CFD applications usually require more effort to implement, but they are more indicative of the types of actual data movement and computation required in state-of-the-art CFD application codes. For example, a data structure that is very efficient in an isolated kernel on a specific system might be inappropriate if incorporated into a larger application. By comparison, the simulated CFD applications require data structures and implementation techniques that are more typical of real CFD applications.

The results in this article

This article reports benchmark performance results for the Y-MP, Y-MP EL, and C-90 systems from Cray Research; the TC2000 from Bolt Baranek and Newman; the Gamma iPSC/860 from Intel; the CM-2, CM-200, and CM-5 from Thinking Machines; the CS-1 from Meiko Scientific; the MP-1 and MP-2 from MasPar Computer; and the KSR-1 from Kendall Square Research.

The results for the MP-1 and -2, the KSR-1, and the CM-5 have not been published before. Many of the other results are improved from previous listings, reflecting improvements both in compilers and implementations. Efforts are underway to port the benchmarks to other systems, so we hope to have more results in the future.

For each benchmark, the performance ratios compare the best performance to date with the best time on one processor of a Cray Y-MP. We do not cite performance rates in millions of floating point operations per second (megaflops). Instead, actual run times (or the performance ratios) should be used to compare systems and implementations. Readers who want to compute megaflops figures should use the standard flop counts in Table 1. These counts were determined using the hardware performance monitor on a Y-MP, and we believe they are close to the minimal counts required for these problems. Because the Integer Sort Benchmark does not involve floating-point operations, we selected a value approximately equal to the number of integer operations required, which let us compute performance rates analogous to megaflops rates. The table also contains megaflops rates calculated in this manner for the fastest implementation on one processor of the Y-MP.

The run times for each benchmark are elapsed time-of-day figures.[1] Memory requirements are available for only some of these implementations; we hope to have complete information in the future.

Whenever possible, we have credited the people and organizations who contributed the performance results. Results that do not cite an individual were contributed by the vendor's staff. In the citations, NAS denotes the NAS Applied Research Branch at NASA Ames, including NASA civil servants and Computer Sciences Corporation contractors.

Results of the Embarrassingly Parallel Benchmark.

System	Problem size	No. of processors	Memory (Mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	2^{28}	1	4.9	126.2	1.00
		8	4.9	15.9	7.95
Y-MP EL		1	4.9	550.5	0.23
		4	4.9	141.2	0.89
C-90		1	4.9	47.6	2.65
		4	4.9	12.4	10.20
		16	4.9	3.2	39.56
TC2000		64	1	284.0	0.44
Gamma iPSC/860		32	1	102.7	1.23
		64	1	51.4	2.46
		128	1	25.7	4.91
CM-2		8K	1	126.6	1.00
		16K	1	63.9	1.97
		32K	1	33.7	3.74
		64K	1	18.8	6.71
CM-200		8K	1	76.9	1.64
		16K	1	39.2	3.22
		32K	1	20.7	6.10
		64K	1	10.9	11.58
CM-5		16	1	42.4	2.98
		32	1	21.5	5.88
		64	1	10.9	11.62
		128	1	5.4	23.49
		256	1	2.7	46.84
		512	1	1.4	90.47
CS-1		16		116.8	1.08
MP-1		4K		248.0	0.51
		16K		69.3	1.82
MP-2		16K		22.4	5.63
KSR-1		32		69.8	1.81
		64		34.9	3.62
		96		23.4	5.39
		128		18.1	6.97

The Embarrassingly Parallel Benchmark

The Embarrassingly Parallel Benchmark is typical of many Monte Carlo applications: Two-dimensional statistics are accumulated from a large number of Gaussian pseudorandom numbers, which are generated according to a scheme that is well-suited for parallel computation. Since it requires almost no communication, this kernel benchmark in some sense estimates the upper limits of a system's floating-point performance.

Not all systems perform well on this problem, which may stem from the fact that the benchmark requires references to several mathematical intrinsic functions — such as the Fortran routines AINT, SQRT, and LOG — for which some systems are evidently not highly optimized. The benchmark's memory requirement was minimal on all systems.

(Gamma iPSC/860 results: J. Baugh of Intel's Supercomputer Systems Division. CM-2, -200, and -5 results: J. Richardson of Thinking Machines. MP-1 and -2 results: J. McDonald of MasPar.)

The Multigrid Benchmark

This simplified multigrid kernel solves a 3D Poisson partial differential equation. The problem is simplified in that it has constant rather than variable coefficients, as in a more realistic application. This code is a good test of both short- and long-distance communication, although the communication patterns are highly structured (as opposed to the Conjugate Gradient Benchmark, discussed next).

(Gamma iPSC/860 results: Boeing Computer Services. CM-2 and -200 results: J. Richardson of Thinking Machines.)

Results of the Multigrid Benchmark.

System	Problem size	No. of processors	Memory (Mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	256^3	1	56.7	22.22	1.00
		8	56.7	2.96	7.51
Y-MP EL		1	56.7	89.19	0.25
		4	56.7	32.11	0.69
C-90		1	56.7	8.65	2.57
		4	56.7	2.42	9.18
		16	56.7	0.96	23.14
Gamma iPSC/860		128		8.6	2.58
CM-2		16K		45.8	0.49
		32K		26.0	0.85
		64K		14.1	1.58
CM-200		16K		30.2	0.74
		32K		17.2	1.29
CS-1		16		42.8	0.52
MP-1		16K		12.0	1.85
MP-2		16K		4.36	5.10
KSR-1		32		20.6	1.08

The Conjugate Gradient Benchmark

This kernel benchmark uses a conjugate gradient method to compute an approximation to the smallest eigenvalue of a large, sparse, symmetric positive definite matrix. This problem is typical of unstructured grid computations in that it tests irregular long-distance communication and uses sparse matrix-vector multiplication. Judging by the results, the benchmark's irregular communication requirement is a challenge for all systems.

(Gamma iPSC/860 results: Boeing Computer Services. CM-2 results: J. Richardson of Thinking Machines. nCube-2 results: B. Hendrickson, R. Leland, and S. Plimpton of Sandia National Laboratory.)

Results of the Conjugate Gradient Benchmark.

System	Problem size	No. of processors	Memory (Mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	2.0×10^6	1	10.4	11.92	1.00
		8	10.4	2.38	5.01
Y-MP EL		1	10.4	65.4	0.18
		4	10.4	23.9	0.50
C-90		1	10.4	4.56	2.61
		4	10.4	1.51	7.89
		16	10.4	0.58	20.55
TC2000		40		51.4	0.23
Gamma iPSC/860		128		8.6	1.38
CM-2		8K		25.6	0.47
		16K		14.1	0.85
		32K		8.8	1.35
CM-200		8K		15.0	0.79
CS-1		16		67.5	0.18
MP-1		4K		64.5	0.18
		16K		14.6	0.82
MP-2		16K		11.0	1.08
KSR-1		32		21.7	0.55
nCube-2		1K		6.1	1.96

Results of the 3D FFT PDE Benchmark.

System	Problem size	No. of processors	Memory (Mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	$256^2 \times 128$	1	42.9	28.77	1.00
		8	42.9	4.19	6.87
Y-MP EL		1	42.9	122.6	0.23
		4	42.9	34.9	0.82
C-90		1	42.9	10.28	2.80
		4	42.9	2.58	11.2
		16	42.9	0.91	31.6
Gamma iPSC/860		64		20.9	1.37
		128		9.7	2.96
CM-2		16K		37.0	0.78
		32K		18.2	1.58
		64K		11.4	2.52
CM-200		8K		45.6	0.63
CS-1		16		170.0	0.17
MP-1		16K		18.3	1.57
MP-2		16K		8.0	3.60
KSR-1		32		13.6	2.12
		64		8.4	3.43

The 3D FFT PDE Benchmark

This kernel benchmark solves a 3D partial differential equation using fast Fourier transforms. It performs the essence of many "spectral" codes, and is a good test of long-distance communication.

This benchmark is unique in that computational library routines can be used legally: The rules of the NAS Parallel Benchmarks specify that assembly-coded library routines can be used to perform matrix multiplication and 1-, 2-, or 3D FFTs.

(Gamma iPSC/860 results: E. Kushner of Intel Supercomputer Systems Division. CM-2 and -200 results: J. Richardson of Thinking Machines.)

Results of the Integer Sort Benchmark.

System	Problem size	No. of processors	Memory (Mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	2^{23}	1	31.1	11.46	1.00
		8	31.1	1.85	6.19
Y-MP EL		1	31.1	153.9	0.07
		4	31.1	41.5	0.28
C-90		1	31.1	5.20	2.20
		4	31.1	1.42	8.07
		16	31.1	0.57	20.10
Gamma iPSC/860		32		25.7	0.45
		64		17.3	0.66
		128		13.6	0.84
CM-2		8K		215.1	0.05
		16K		111.5	0.10
		32K		56.0	0.20
CS-1		16		62.7	0.18
MP-1		16K		23.6	0.49
MP-2		16K		18.4	0.62
KSR-1		32		40.2	0.29

The Integer Sort Benchmark

This kernel benchmark tests a sorting operation that is important in "particle method" codes. It is similar to "particle in cell" physics applications, where particles are assigned to cells and may drift out. The sorting operation reassigns particles to the appropriate cells. The benchmark tests both integer computation speed and communication performance. This problem is unique in that floating-point arithmetic is not involved, although significant data communication is required.

(Gamma iPSC/860 results: E. Kushner of Intel Supercomputer Systems Division. CM-2 results; L. Dagum of NAS.)

The simulated CFD applications

The three simulated CFD application benchmarks are intended to accurately represent the principal computational and data movement requirements of modern CFD applications.

(1) The Lower-Upper Diagonal Benchmark does not perform an LU factorization, but instead uses a symmetric, successive overrelaxation numerical scheme to solve a regular-sparse, block (5×5) lower and upper triangular system. This problem represents the computations associated with a newer class of implicit CFD algorithms, typified at NASA Ames by INS3D-LU. This problem exhibits a somewhat limited amount of parallelism compared to the other two simulated CFD applications. A complete solution of this benchmark requires 250 iterations.

(2) The Scalar Pentadiagonal Benchmark solves multiple independent systems of nondiagonally dominant, scalar pentadiagonal equations. A complete solution requires 400 iterations.

(3) The Block Tridiagonal Benchmark solves multiple independent systems of nondiagonally dominant, block tridiagonal equations with a 5×5 block size. A complete solution requires 200 iterations.

Results for the Lower-Upper Diagonal Benchmark.

System	Problem size	No. of processors	Memory (Mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	64^3	1	32.3	333.5	1.00
		8	32.3	49.5	6.74
Y-MP EL		1	32.3	1,449.0	0.23
		4	32.3	522.3	0.64
C-90		1	32.3	157.6	2.12
		4	32.3	43.9	7.59
		16	32.3	17.6	18.93
TC2000		62		3,032.0	0.11
Gamma		64	12	690.8	0.48
iPSC/860		128	16	442.5	0.75
CM-2		8K	14	1,307.0	0.26
		16K	14	850.0	0.39
		32K	14	546.0	0.61
CM-5		64		336.0	0.99
CS-1		16		2,937.0	0.11
MP-1		4K		1,785.0	0.19
MP-2		4K		562.0	0.59
KSR-1		32		1,041.3	0.32

Results for the Scalar Pentadiagonal Benchmark.

System	Problem size	No. of processors	Memory (Mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	64^3	1	9.2	471.5	1.00
		8	9.2	64.6	7.30
Y-MP EL		1	9.2	2,026.0	0.23
		4	9.2	601.9	0.78
C-90		1	9.2	184.70	2.55
		4	9.2	49.74	9.48
		16	9.2	13.06	36.10
TC2000		112		880.0	0.54
Gamma		64		667.3	0.71
iPSC/860		128		449.5	1.05
CM-2		8K		3,900	0.12
		16K		2,104	0.22
		32K		1,080	0.44
CM-5		64		180.0	2.62
CS-1		16		2,975	0.16
MP-1		4K		1,772	0.27
MP-2		4K		657	0.72
KSR-1		32		377.7	1.25
		64		228.8	2.06
		96		170.2	2.77
		128		150.0	3.14

Results for the Block Tridiagonal Benchmark.

System	Problem size	No. of processors	Memory (Mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	64^3	1	42.3	792.4	1.00
		8	42.3	114.0	6.95
Y-MP EL		1	42.3	4,033	0.20
		4	42.3	1,208	0.66
C-90		1	42.3	356.9	2.22
		4	42.3	96.10	8.25
		16	42.3	28.39	27.91
TC2000		112		1,378	0.58
Gamma		64		714.7	1.11
iPSC/860		128		414.3	1.91
CM-2		16K		3,328	0.24
		32K		1,914	0.41
CM-5		64		176.0	4.50
CS-1		16		2,984	0.27
MP-1		4K		2,396	0.33
MP-2		4K		803	0.99
KSR-1		32		439.0	1.81
		64		239.4	3.31
		96		167.9	4.72
		128		134.5	5.89

These last two benchmarks are representative of computations associated with the implicit operators of CFD codes (such as ARC3D at NASA Ames). They are similar in many respects, but there is a fundamental difference in the communication-to-computation ratio.

(Gamma iPSC/860 and CM-2 results: S. Weeratunga, R. Fatoohi, E. Barszcz, and V. Venkatakrishnan of NAS, except iPSC/860 BT and SP results: Boeing Computer Services. CM-5 results: J. Richardson of Thinking Machines.)

Thinking Machines' results using library routines.

Benchmark	System	No. of processors	Time (sec.)	Ratio to Y-MP/1	Ratio to CM-*x* without library
Integer Sort	CM-2	16K	35.8	0.32	3.11
		32K	21.0	0.55	2.67
		64K	14.9	0.77	
	CM-200	64K	5.7	2.01	
Scalar Pentadiagonal	CM-2	16K	1,444	0.33	1.46
		32K	917.0	0.51	1.18
		64K	640.0	0.74	
Block Tridiagonal	CM-2	16K	1,118	0.71	2.98
		32K	634.0	1.25	3.01
		64K	370.0	2.14	
	CM-200	16K	832.0	0.95	
		32K	601.0	1.32	

Other results

As far as we know, all these timings were taken from runs that fully complied with the rules and restrictions stated in the benchmark document.[1] Two of these rules are that assembly language may not be used, and that assembly-coded library routines may only be used for a restricted set of operations. The allowable exceptions include vendor-supported routines to synchronize processors, communicate data, perform array transpositions, evaluate Fortran intrinsic functions, perform dense matrix multiplication, and compute fast Fourier transforms.

There are several reasons for these restrictions on assembly code. Without them, an entire benchmark could be implemented in assembly-level code. While such performance results might be interesting, they would hardly be indicative of the performance that a scientist could reasonably expect on a full-scale application program. In other words, the tuning rules for the NAS Parallel Benchmarks reflect the expectation (and experience) that real scientific applications consist largely of Fortran or C code, and that the use of library routines is restricted to a handful of widely available functions.

Nonetheless, some scientists have tried implementations of the benchmarks using library routines that do not comply with the official rules. In particular, Thinking Machines has obtained performance results using assembly-coded library routines to perform key computations in several benchmarks (see Table 2).[3] Its implementation of the Integer Sort Benchmark on the CM-2, for example, runs more than twice as fast as reported in that section above, and the Block Tridiagonal Benchmark is nearly three times as fast.

Delta iPSC/860

Some of the benchmarks written by the NAS Applied Research branch were ported directly to Intel's Delta iPSC/860 prototype. The Delta machine uses the same microprocessor as the Gamma but has a 2D grid topology and a faster router. Since the 2D topology results in a lower connectivity than is available with the Gamma's hypercube topology, the faster router must try to make up for the additional distance messages might have to travel. Table 2 presents the relative performance between the two machines. We do not present the Delta's absolute performance because the tests did not use the fully optimized codes used for the results in the previous tables. Also, the codes were not tuned for the Delta's mesh topology; there might be a significant performance improvement if this were done.

Table 2. Performance of Intel Delta compared to Gamma Machine (May 1992).

Benchmark	No. of processors	Ratio to Gamma
Embarrassingly Parallel	64	1.03
Integer Sort	64	1.36
	128	1.09
Multigrid	32	0.70
Conjugate Gradient	128	1.05
Lower-Upper Diagonal	16	0.96

Sustained performance per dollar

So far we have not addressed the price differences among these systems. The C-90 system, for example, exhibits superior performance on these benchmarks, and its current purchase price is correspondingly much higher than that of the iPSC/860 and the CM-2.

One way to compensate for these price differences is to compute sustained performance per million dollars. We have done this by dividing the performance ratio by the nominal purchase price (the price of a complete system with a typical set of peripherals). Table 3 shows these figures for the Embarrassingly Parallel and Scalar Pentadiagonal benchmarks on 10 systems. These figures are very approximate and do not necessarily reflect current prices; old system prices typically drop substantially when a vendor introduces a new model. Because the prices are approximate and changeable — and because the memory size, disk capacity, and I/O performance of these systems are certainly not equivalent — the last column is only a very rough indication of sustained performance per dollar.

With some algorithmic experimentation and tuning, respectable performance rates have been achieved on several multiprocessor systems under the NAS Parallel Benchmarks. Except for the Embarrassingly Parallel Benchmark,

Table 3. Approximate sustained performance per dollar.

Benchmark	System	No. of processors	Ratio to Y-MP/1	Nominal cost ($)	Performance per million $
Embarrassingly Parallel	C-90	16	39.56	36M	1.10
	Y-MP	8	7.95	15M	0.53
	Y-MP EL	4	0.89	1.5M	0.59
	iPSC/860	128	4.91	3M	1.64
	CM-2	32K	3.74	5M	0.75
	CM-5	512	90.47	15M	6.03
	MP-1	16K	1.82	0.5M	3.64
	MP-2	16K	5.63	1M	5.63
	CS-1	16	1.08	0.3M	3.60
	KSR-1	128	6.97	6M	1.16
Scalar Pentadiagonal	C-90	16	36.10	36M	1.00
	Y-MP	8	7.30	15M	0.49
	Y-MP EL	4	0.78	1.5M	0.52
	iPSC/860	128	1.05	3M	0.35
	CM-2	32K	0.44	5M	0.09
	CM-5	64	2.62	2.5M	1.05
	MP-1	4K	0.27	0.2M	1.35
	MP-2	4K	0.72	0.3M	2.40
	CS-1	16	0.16	0.3M	0.53
	KSR-1	128	3.14	6M	0.52

- the 16-processor C-90 is the highest performing system tested;
- the 128-processor iPSC/860 system, the 32K-processor CM-2, the 64-node CM-5, and the 16K-processor MP-2 are roughly equivalent to one (in some cases significantly more than one) Y-MP processor; and
- when sustained performance rates are normalized by system prices, the CM-5 and the MP-1 and -2 deliver somewhat more performance per dollar than the C-90.

For the latest generation of parallel computers, only Kendall Square has given us a complete set of benchmark numbers, while Thinking Machines and MasPar have provided a limited set of results. These latest results are rather encouraging, with some systems performing better than a Y-MP/1, even on the challenging Scalar Pentadiagonal and Block Tridiagonal benchmarks. We expect even better performance from these machines as the compilers mature and the implementations are tuned further.

Some scientists have suggested that the answer to obtaining high performance rates on highly parallel computers is to use alternative algorithms with lower interprocessor communication requirements. However, the scientists in our research group have found that a certain amount of long-distance communication is unavoidable for these types of applications. Alternative algorithms with higher computation rates usually require more iterations to converge to a solution and thus require more overall run time. Clearly it is pointless to use numerically inefficient algorithms merely to exhibit artificially high performance rates on a particular parallel architecture.[4]

Acknowledgments

Dagum and Simon's work was funded by the NASA Ames Research Center under contract NAS2-12961.

Sample Fortran programs implementing the NAS Parallel Benchmarks on a single-processor system, as well as the benchmark document itself, are available as an aid to implementors. The sample codes are on Macintosh floppy disks and contain the Fortran source codes, "readme" files, input data files, and reference output data files for correct implementations of the benchmark problems. The codes have been validated on systems ranging from conventional workstations to supercomputers. Contact NAS Systems Division, Mail Stop 258-8, NASA Ames Research Center, Moffett Field, CA 94035, attention: NAS Parallel Benchmark Codes.

References

1. D. Bailey et al., eds., "The NAS Parallel Benchmarks," Tech. Report RNR-91-02, NASA Ames Research Center, Moffett Field, Calif., 1991.
2. D.H. Bailey et al., "The NAS Parallel Benchmarks," *Int'l J. Supercomputer Applications*, Vol. 5, No. 3, Fall 1991, pp. 63-73.
3. G. Bhanot et al., "Implementing the NAS Parallel Benchmarks on the CM-2 and CM200 Supercomputers," tech. report, Thinking Machines Corp, Cambridge, Mass.
4. D.H. Bailey, "Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers," *Supercomputing Rev.*, Aug. 1991, p. 54-55. Also published in *Supercomputer*, Sept. 1991, p. 4-7.

David H. Bailey is with the NAS Applied Research Branch at NASA Ames Research Center. His research interests include parallel numeric algorithms, fast Fourier transforms, probability and statistics, multiprecision computation, computational number theory, and supercomputer performance analysis. Bailey received a PhD in mathematics from Stanford University in 1976, and a BS in mathematics from Brigham Young University in 1972.

Eric Barszcz is with the NAS Applied Research Branch at NASA Ames Research Center, and is a doctoral candidate in computer engineering at the University of California at Santa Cruz. His research interests include parallel numeric algorithms, dynamic load balancing, and locality issues. He received an MS in mathematics in 1983, and a BS in computer science and mathematics and a BA in chemistry in 1981, all from the University of Rhode Island.

Leonardo Dagum works for Computer Sciences Corporation, where his research interests include benchmarking parallel architectures and developing algorithms for Monte Carlo simulations of rarefied gases on homogeneous and heterogeneous parallel architectures. He received a PhD from the Department of Aeronautics and Astronautics at Stanford University in 1990, and a BS in engineering physics from Queen's University in Kingston, Canada, in 1985.

Horst D. Simon works for Computer Sciences Corporation at NASA Ames Research Center, where he is a department manager for three groups of researchers in parallel algorithm development, scientific visualization, and grid generation. His research interests are in high-performance algorithms for vector and parallel machines, especially sparse matrix algorithms, algorithms for large-scale eigenvalue problems, and domain-decomposition algorithms for unstructured domains in parallel processing. Simon received the 1988 Gordon Bell Award for parallel processing research. He received a PhD in mathematics from the University of California at Berkeley in 1982, and a diploma in mathematics from the Technical University of Berlin in 1978.

Readers can contact the authors at NASA Ames Research Center, Mail Stop T045-1, Moffett Field, CA 94035.