NAS Parallel Benchmark Results

D. H. Bailey
E. Barszcz

NAS Applied Research Branch
NASA Ames Research Center
Moffett Field, CA 94035

L. Dagum
H. D. Simon

Computer Sciences Corp.
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

The NAS Parallel Benchmarks have been developed at NASA Ames Research Center to study the performance of parallel supercomputers. The eight benchmark problems are specified in a "pencil and paper" fashion. This paper presents performance results of various systems using the NAS Parallel Benchmarks. These results represent the best results that have been reported to us for the specific systems listed. They represent implementation efforts performed by personnel in both the NAS Applied Research Branch of NASA Ames and in other organizations.

1. Introduction

The Numerical Aerodynamic Simulation (NAS) Program, which is based at NASA Ames Research Center, is dedicated to advance the science of computational aerodynamics. One key goal of the NAS organization is to demonstrate by the year 2000 an operational computing system capable of simulating an entire aerospace vehicle system within a computing time of one to several hours. It is currently projected that the solution of this grand challenge problem will require a computer system that can perform scientific computations at a sustained rate approximately one thousand times faster than 1990 generation supercomputers. Most likely such a computer system will employ hundreds or even thousands of processors operating in parallel.

At the present time, there are several commercial highly parallel systems available with computing power roughly competitive with conventional supercomputers (even greater on some special problems). Unfortunately, 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, the problem has to be specified in sufficient detail that a unique solution exists, and the required output has to be brief yet detailed enough to certify that the problem has been solved correctly. But the details of the implementation should be left to the programmer as far as possible.

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

The eight problems consist of five "kernels" and three "simulated computational fluid dynamics (CFD) applications". Each of these is defined fully in [3]. The five kernels are relatively compact problems, each of which emphasizes a particular type of numerical computation. Compared with the simulated CFD applications, they can be implemented fairly readily and provide insight as to the general levels of performance that can be expected on these specific types of numerical computations.

The simulated CFD applications, on the other hand, 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, in an isolated kernel a certain data structure may be very efficient on a certain system, and yet this data structure would 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.

Space does not permit a complete description of these benchmark problems. A more detailed description of these benchmarks, together with the rules and restrictions associated with the benchmarks, may be found in [2]. The full specification of the benchmarks is given in [3].

Sample Fortran programs implementing the NPB on a single processor system are available as an aid to implementors. These programs, as well as the benchmark document itself, are available from the following address: NAS Systems Division, Mail Stop 258-8, NASA Ames Research Center, Moffett Field, CA 94035, attn: NAS Parallel Benchmark Codes. The sample codes are provided 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. These codes have been validated on a number of computer systems ranging from conventional workstations to supercomputers.

In the following, each of the eight benchmarks will be briefly described, and then the best performance results we have received to date for each computer system will be given in Tables 2 through 9. These tables include memory requirements, run times and performance ratios. The performance ratios compare individual timings with the current best time on that benchmark achieved on one processor of a Cray Y-MP. The run times in each case are elapsed time of day figures, measured in accordance with the specifications given in [3]. Memory requirements are currently available for only some of these implementations. We hope to have complete information for these columns in future editions of this paper.

Note that performances rates are not cited in millions of floating point operations per second (megaflops) in these tables. We suggest instead that the actual run times (or, equivalently, the performance ratios) be examined when comparing different systems and implementations. For those who wish to compute megaflops figures for the NAS Parallel Benchmarks on any system, we insist that they be computed using the standard floating point operation (flop) counts given in Table 1. Table 1 also contains megaflops rates calculated in this manner for the current fastest implementation on one processor of the Cray Y-MP.

With the exception of the Integer Sort benchmark, these standard flop counts were determined by using the hardware performance monitor on a Cray Y-MP, and we believe that they are close to the minimal counts required for these problems. In the case of the Integer Sort benchmark, which does not involve floating-point operations, we selected a value approximately equal to the number of integer operations required, in order to permit the computation of performance rates analogous to megaflops rates. We reserve the right to change these standard flop counts in the future if deemed necessary.

Whenever possible, we have tried to credit the actual individuals and organizations who have contributed the performance results cited in the tables. In these citations, NAS denotes the NAS Applied Research Branch at NASA Ames (including both NASA civil servants and Computer Science Corp. contractors); RIACS denotes the parallel systems division of the Research Institute for Advanced Computer Science, which is located at NASA Ames; BBN denotes Bolt, Beranek and Newman; Boeing denotes Boeing Computer Services, Inc.; CRI denotes Cray Research, Inc.; Intel denotes the Supercomputer Systems Division of Intel Corp.; Maspar denotes Maspar Computer Corp.; Meiko denotes Meiko Scientific Corp.; and TMC denotes Thinking Machines, Inc. Where no individual citation is made for a specific model, the results are due to vendor staff.

Unfortunately, the limited space in this report does not permit discussion of the methods used in any of these implementations. However, we have included references to technical papers describing these methods whenever such papers are available. Readers are referred to these documents for full details.

This report includes a number of new results not previously published. The Cray C-90, Cray Y-MP EL, the Maspar MP-1 and MP-2, and the Meiko CS-1 results in particular have not previously been disclosed. In quite a few other instances, results are improved from previous listings, reflecting improvements both in compilers and implementations. Efforts are currently underway to port the NAS Parallel Benchmarks on other systems, and we hope to have some results in the future.

2. The Embarrassingly Parallel Benchmark

The first of the five kernel benchmarks is an "embarrassingly parallel" problem. In this benchmark, two-dimensional statistics are accumulated from a large number of Gaussian pseudorandom numbers, which are generated according to a particular scheme that is well-suited for parallel computation. This problem is typical of many "Monte-Carlo" applications. Since it requires almost no communication, in some sense this benchmark provides an estimate of the upper achievable limits for floating point performance on a particular system.

Results for the embarrassingly parallel benchmark are shown in Table 2. Not all systems exhibit high rates on this problem. This appears to stem from the fact that this benchmark requires references to several mathematical intrinsic functions, such as the Fortran routines AINT, SQRT, and LOG, and evidently these functions are not highly optimized on some systems. The memory requirement for this benchmark was minimal on all systems.

Intel iPSC/860 results are due to J. Baugh of Intel. CM-2 and CM-200 results are due to J. Richardson of TMC. Maspar results are due to J. MacDonald of Maspar.

3. The Multigrid Benchmark

The second kernel benchmark is a simplified multigrid kernel, which solves a 3-D Poisson PDE. This problem is simplified in the sense 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 highly communication, although the communication patterns are highly structured (as opposed to the conjugate gradient benchmark).

Results for this benchmark, for problem size 256^3, are shown in Table 3. Intel results are due to BCS. CM-2 and CM-200 results are due to J. Richardson at TMC.

4. The Conjugate Gradient Benchmark

In this benchmark, a conjugate gradient method is used to compute an approximation to the smallest eigenvalue of a large, sparse, symmetric positive definite matrix. This kernel is typical of unstructured grid computations in that it tests irregular long distance communication and employs sparse matrix vector multiplication.

The irregular communication requirement of this benchmark is evidently a challenge for all systems. Results, for problem size 2.0×10^6, are shown in Table 4. Intel results are due to BCS. CM-2 results are due to J. Richardson of TMC.

5. The 3-D FFT PDE Benchmark

In this benchmark a 3-D partial differential equation is solved using FFTs. This kernel performs the essence of many "spectral" codes. It is a good test of

Benchmark Name	Abbr.	Operation Count	Y-MP Rate
Emb. Parallel	EP	2.668×10^{10}	211
Multigrid	MG	3.905×10^{09}	176
Conjugate Gradient	CG	1.508×10^{09}	127
3-D FFT PDE	FT	5.631×10^{09}	196
Integer Sort	IS	7.812×10^{08}	68
LU Sim. CFD Appl.	LU	6.457×10^{10}	194
SP Sim. CFD Appl.	SP	1.020×10^{11}	216
BT Sim. CFD Appl.	BT	1.813×10^{11}	229

Table 1: Standard Operation Counts and Current Y-MP/1 Megaflops Rates

Computer System	No. Proc.	Memory (mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	1	4.9	126.2	1.00
	8	4.9	15.87	7.95
Y-MP EL	1	4.9	550.5	0.23
	4	4.9	141.2	0.89
C-90	1	4.9	47.60	2.65
	4	4.9	12.37	10.20
	16	4.9	3.19	39.56
TC2000	64	1	284.0	0.44
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
CS-1	16		116.8	1.08
MP-1	4K		248	0.51
	16K		88	1.43

Table 2: Results of the Embarrassingly Parallel (EP) Benchmark

Computer System	No. Proc.	Memory (mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	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
iPSC/860	128		8.61	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		13.1	1.70

Table 3: Results of the Multigrid (MG) Benchmark

Computer System	No. Proc.	Memory (mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	1	10.4	11.92	1.00
	8	10.4	2.38	5.01
Y-MP EL	1	10.4	65.35	0.18
	4	10.4	23.91	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
iPSC/860	128		8.61	1.38
CM-2	8K		25.6	0.47
CM-2	16K		14.1	0.85
CM-2	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

Table 4: Results of the Conjugate Gradient (CG) Benchmark

Computer System	No. Proc.	Memory (mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	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
iPSC/860	64		20.93	1.37
	128		9.72	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		19.6	1.47

Table 5: Results of the 3-D FFT PDE (FT) Benchmark

long-distance communication performance.

The rules of the NAS Parallel Benchmarks specify that assembly-coded, library routines may be used to perform matrix multiplication and one-dimensional, two-dimensional or three-dimensional FFTs. Thus this benchmark is somewhat unique in that computational library routines may be legally employed.

Results, for problem size $256^2 \times 128$, are shown in Table 5. Intel results are due to E. Kushner of Intel. CM-2 and CM-200 results are due to J. Richardson of TMC.

6. The Integer Sort Benchmark

This benchmark tests a sorting operation that is important in “particle method” codes. This type of application is similar to “particle in cell” applications of physics, wherein particles are assigned to cells and may drift out. The sorting operation is used to reassign particles to the appropriate cells. This benchmark tests both integer computation speed and communication performance.

This problem is unique in that floating point arithmetic is not involved. Significant data communication, however, is required. Results, for problem size 2^{23}, are shown in Table 6. Intel results are due to to E. Kushner of Intel. CM-2 results are due to L. Dagum of NAS.

Computer System	No. Proc.	Memory (mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	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	20.1
iPSC/860	32		25.72	0.45
	64		17.26	0.66
	128		13.59	0.84
CM-2	8K		215.1	0.05
	16K		111.5	0.10
	32K		56.0	0.20
MP-1	16K		75	0.15
CS-1	16		62.7	0.18

Table 6: Results of the Integer Sort (IS) Benchmark

Computer System	No. Proc.	Memory (mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	1	32.3	333.5	1.00
	8	32.3	49.50	6.74
Y-MP EL	1	32.3	1449	0.23
	4	32.3	522.3	0.64
C-90	1	32.3	157.6	2.12
	4	32.3	43.94	7.59
	16	32.3	17.62	18.93
TC2000	62		3032	0.11
iPSC/860	64	12	690.8	0.48
	128	16	442.5	0.75
CM-2	8K	14	1307	0.26
	16K	14	850.0	0.39
	32K	14	572.0	0.58
CS-1	16		2937	0.11
MP-1	4K		1958	0.17
MP-2	4K		658	0.51

Table 7: Results for the LU Simulated CFD Application

7. The Three Simulated CFD Application Benchmarks

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

The first of these is the called the lower-upper diagonal (LU) benchmark. It does not perform a LU factorization but instead employs a symmetric successive over-relaxation (SSOR) 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 the code “INS3D-LU”. This problem exhibits a somewhat limited amount of parallelism compared to the next two.

The second simulated CFD application is called the scalar pentadiagonal (SP) benchmark. In this benchmark, multiple independent systems of non-diagonally dominant, scalar pentadiagonal equations are solved. The third simulated CFD application is called the block tridiagonal (BT) benchmark. In this benchmark, multiple independent systems of non-diagonally dominant, block tridiagonal equations with a 5×5 block size are solved.

SP and the third simulated CFD application (BT) are representative of computations associated with the implicit operators of CFD codes such as “ARC3D” at NASA Ames. SP and BT are similar in many respects, but there is a fundamental difference with respect to the communication to computation ratio.

Performance figures for the three simulated CFD applications, for problem size 64^3, are shown in Tables 7, 8 and 9. Timings are cited as complete run times, in seconds, as with the other benchmarks. 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.

Intel and CM-2 results are due to S. Weeratunga, R. Fatoohi, E. Barszcz and V. Venkatakrishnan of NAS, except that BT and SP results on the Intel are due to BCS.

8. Other Results

As far as we have been able to determine, the timings presented above all represent runs that fully comply with the rules and restrictions stated in the benchmark document [3]. One of these rules is that except for a short list of mathematical functions, assembly language and assembly-coded library routines may not be used for computation. The exceptions include the standard Fortran intrinsic functions, as well as routines to perform dense matrix multiplication and fast Fourier transforms.

There are several reasons for these restrictions on assembly code. First of all, without restrictions of some sort, an entire benchmark might be implemented in assembly-level code. While such performance re-

Computer System	No. Proc.	Memory (mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	1	9.2	471.5	1.00
	8	9.2	64.60	7.30
Y-MP EL	1	9.2	2026	0.23
	4	9.2	601.9	0.78
C-90	1	9.2	184.7	2.55
	4	9.2	49.74	9.48
	16	9.2	13.06	36.10
TC2000	112		880.0	0.54
iPSC/860	64		667.3	0.71
	128		449.5	1.05
CM-2	8K		3900	0.12
	16K		2104	0.22
	32K		1080	0.44
CS-1	16		2975	0.16
MP-1	4K		1772	0.27
MP-2	4K		668	0.71

Table 8: Results for the SP Simulated CFD Application

Computer System	No. Proc.	Memory (mwords)	Time (sec.)	Ratio to Y-MP/1
Y-MP	1	42.3	792.4	1.00
	8	42.3	114.0	6.95
Y-MP EL	1	42.3	4033	0.20
	4	42.3	1208	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		1378	0.58
iPSC/860	64		714.7	1.11
	128		414.3	1.91
CM-2	16K		3328	0.24
	32K		1914	0.41
CS-1	16		2984	0.27
MP-1	4K		2420	0.33
MP-2	4K		870	0.91

Table 9: Results for the BT Simulated CFD Application

Benchmark	Computer System	No. Proc.	Time (sec.)	Ratio to Y-MP/1
IS	CM-2	16K	35.8	0.32
		32K	21.0	0.55
		64K	14.9	0.77
	CM-200	64K	5.7	2.01
LU	CM-2	16K	868.0	0.38
		32K	546.0	0.61
SP	CM-2	16K	1444	0.33
		32K	917.0	0.51
		64K	640.0	0.74
BT	CM-2	16K	1118	0.71
		32K	634.0	1.25
		64K	370.0	2.14
	CM-200	16K	832.0	0.95
		32K	601.0	1.32

Table 10: Unofficial TMC Results Using Library Routines

sults might be interesting, they would hardly be indicative of the performance that a scientist could reasonably expect on a full-scale application program. One reason that only the above-mentioned routines are allowed is that in our experience only these are generally available on new systems. For more specialized library routines, it is difficult to determine whether they are truly general purpose, i.e. not relying on a specific data layout. Furthermore, even if an assembly-coded library routine can be utilized for an inner computational kernel, this does not help the large mass of additional coding that comprises a full-scale application. In short, the tuning rules for the NPB reflect our expectation (and experience) that real scientific applications consist largely of Fortran or C code, and that usage of library routines is restricted to a handful of widely available mathematical functions.

Nonetheless, some scientists have attempted implementations of the NPB using library routines beyond the ones allowed in [3]. In particular, Thinking Machines, Inc. has obtained performance results using assembly-coded library routines for several of the NPB. Their implementation of the IS benchmark, for example, runs more than twice as fast as reported in Table 6, and their rates for the BT benchmark are nearly three times as fast as reported in Table 9. Some of these results are shown in Table 10 [4].

9. Sustained Performance Per Dollar

One aspect of the relative performance of these systems has not been addressed so far, namely the differ-

B'mark	Computer System	No. Proc.	Ratio to Y-MP/1	Perf. per million $
FT	C-90	16	31.60	0.87
	Y-MP	8	6.87	0.27
	iPSC/860	128	2.96	0.99
	CM-2	32K	2.52	0.50
	MP-1	16K	1.47	1.47
	CS-1	16	0.17	0.57
LU	C-90	16	18.93	0.53
	Y-MP	8	6.74	0.27
	iPSC/860	128	0.75	0.25
	CM-2	32K	0.58	0.12
	MP-2	4K	0.51	1.02
	CS-1	16	0.11	0.37

Table 11: Approximate Sustained Performance Per Dollar

ences in price between these systems. We should not be too surprised that the Cray C-90 system, for example, exhibits superior performance rates on these benchmarks, since its current purchase price is 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, i.e. the performance ratio figures shown in Tables 2 through 9 divided by the purchase price in millions. Some figures of this type are shown in Table 11 for two of the benchmarks, the FT benchmark and the LU benchmark and for five different systems. They are based on 36 million, 25 million, 3 million, 5 million, 1 million, 500,000 and 300,000 U.S. dollars, respectively, for the Cray C-90, the Cray Y-MP, the Intel iPSC/860, the CM-2, the Maspar MP-1, the Maspar MP-2 and the Meiko CS-1. These are approximate current prices, obtained from vendor personnel, for complete systems with 16, 8, 128, 32K and 16K, 4K and 16 processors, respectively, with one, two, one, four, one, 0.25 and 0.5 gigabytes of main memory, respectively, and with a typical set of peripherals. Because of the approximate and changeable nature of these prices, and because the memory sizes, disk capacities and I/O performances of these systems are certainly not equivalent, the figures in the last column of Table 11 should be interpreted as only very rough indications of sustained performance per dollar.

10. Conclusions

With some algorithmic experimentation and tuning, respectable NPB performance rates have been achieved on several multiprocessor systems. The 16 processor Cray C-90 system is consistently the highest performing system tested, far surpassing any of the highly parallel systems. The Intel 128 processor iPSC/860 system and the 32K CM-2 system each show promise, but they do not yet demonstrate sustained performance comparable to full Cray systems. Instead, in both cases their rates appear to be equivalent to about one or, in some cases, two Y-MP processors. When sustained performance rates are normalized by system prices, the situation is somewhat different: the highly parallel systems are approximately on a par with the Cray systems.

The Cray NPB performance results uniformly are large fractions (in some cases over fifty percent) of the theoretical peak performance of these systems. By contrast, the NPB performance rates on the highly parallel systems are typically only two to five percent of the theoretical peak performance of these systems. Reasons for the low sustained-to-peak ratios on the highly parallel systems are not hard to identify: immature compilers, insufficient bandwidth between processors and main memory, and insufficient bandwidth between separate processing nodes. Clearly the challenge of the highly parallel vendors is to alleviate these bottlenecks in future editions of their systems.

Some scientists have suggested that the answer to obtaining high performance rates on highly parallel computers is to substitute alternative algorithms that have lower interprocessor communication requirements. However, it has been the experience of the scientists in our research group that a certain amount of long-distance communication is unavoidable for these types of applications. Alternative algorithms that have higher computation rates usually require more iterations to converge to a solution and thus require more overall run time. Clearly it is pointless to employ numerically inefficient algorithms merely to exhibit artificially high performance rates on a particular parallel architecture [1].

References

[1] D. H. Bailey, "Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers", *Supercomputing Review*, August 1991, p. 54 – 55. Also published in *Supercomputer*, September 1991, p. 4 – 7.

[2] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga, "The NAS Parallel Benchmarks", *Intl. Journal of Supercomputer Applications*, v. 5, no. 3 (Fall 1991), pp. 63 – 73.

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

[4] G. Bhanot, K. Jordan, J. Kennedy, J. Richardson, D. Sandee and M. Zagha, "Implementing the NAS Parallel Benchmarks on the CM-2 and CM200 Supercomputers", Thinking Machines Corp, Cambridge, MA 02142.

[5] S. Breit, W. Celmaster, W. Coney, R. Foster, B. Gaiman, G. Montry and C. Selvidge, "The Role of Computational Balance in the Implementation of the NAS parallel Benchmarks on the BBN TC2000 Computer", submitted to *Concurrency*, April 1991.