THE NAS PARALLEL BENCHMARKS

D. H. Bailey[1], E. Barszcz[1], J. T. Barton[1], D. S. Browning[2], R. L. Carter, L. Dagum[2], R. A. Fatoohi[2], P. O. Frederickson[3], T. A. Lasinski[1], R. S. Schreiber[3], H. D. Simon[2], V. Venkatakrishnan[2] and S. K. Weeratunga[2]
NAS Applied Research Branch
NASA Ames Research Center, Mail Stop T045-1
Moffett Field, CA 94035
Ref: *Intl. Journal of Supercomputer Applications*, vol. 5, no. 3 (Fall 1991), pg. 66–73

Abstract

A new set of benchmarks has been developed for the performance evaluation of highly parallel supercomputers. These benchmarks consist of five "parallel kernel" benchmarks and three "simulated application" benchmarks. Together they mimic the computation and data movement characteristics of large scale computational fluid dynamics applications.

The principal distinguishing feature of these benchmarks is their "pencil and paper" specification — all details of these benchmarks are specified only algorithmically. In this way many of the difficulties associated with conventional benchmarking approaches on highly parallel systems are avoided.

[1]This author is an employee of NASA Ames Research Center

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

[3]This author is an employee of the Research Institute for Advanced Computer Science (RIACS). This work is supported by the NAS Systems Division via Cooperative Agreement NCC 2-387 between NASA and the Universities Space Research Association.

1 Introduction

The Numerical Aerodynamic Simulation (NAS) Program, which is based at NASA Ames Research Center, is a large scale effort to advance the state of computational aerodynamics. Specifically, the NAS organization aims "to provide the Nation's aerospace research and development community by the year 2000 a high-performance, operational computing system capable of simulating an entire aerospace vehicle system within a computing time of one to several hours" ([4], page 3). The successful solution of this "grand challenge" problem will require the development of computer systems that can perform the required complex scientific computations at a sustained rate nearly one thousand times greater than current generation supercomputers can now achieve. The architecture of computer systems able to achieve this level of performance will likely be dissimilar to the shared memory multiprocessing supercomputers of today. While no consensus yet exists on what the design will be, it is likely that the system will consist of at least 1,000 processors computing in parallel.

Highly parallel systems with computing power roughly equivalent to traditional shared memory multiprocessors exist today. Unfortunately, for various reasons, the performance evaluation of these systems on comparable types of scientific computations is very difficult. Little relevant data is available for the performance of algorithms of interest to the computational aerophysics community on many currently available parallel systems. Benchmarking and performance evaluation of such systems has not kept pace with advances in hardware, software and algorithms. In particular, there is as yet no generally accepted benchmark program or even a benchmark strategy for these systems.

The popular "kernel" benchmarks that have been used for traditional vector supercomputers, such as the Livermore Loops [12], the LINPACK benchmark [9, 10] and the original NAS Kernels [7], are clearly inappropriate for the performance evaluation of highly parallel machines. First of all, the tuning restrictions of these benchmarks rule out many widely used parallel extensions. More importantly, the computation and memory requirements of these programs do not do justice to the vastly increased capabilities of the new parallel machines, particularly those systems that will be available by the mid-1990s.

On the other hand, a full scale scientific application is similarly unsuitable.

First of all, porting a large program to a new parallel computer architecture requires a major effort, and it is usually hard to justify a major research task simply to obtain a benchmark number. For that reason we believe that the otherwise very successful PERFECT Club benchmark [11] is not suitable for highly parallel systems. This is demonstrated by only very sparse performance results for parallel machines in the recent reports [13, 14, 8].

Alternatively, an application benchmark could assume the availability of automatic software tools for transforming "dusty deck" source into efficient parallel code on a variety of systems. However, such tools do not exist today, and many scientists doubt that they will ever exist across a wide range of architectures.

Some other considerations for the development of a meaningful benchmark for a highly parallel supercomputer are the following:

- Advanced parallel systems frequently require new algorithmic and software approaches, and these new methods are often quite different from the conventional methods implemented in source code for a sequential or vector machine.
- Benchmarks must be "generic" and should not favor any particular parallel architecture. This requirement precludes the usage of any architecture-specific code, such as message passing code.
- The correctness of results and performance figures must be easily verifiable. This requirement implies that both input and output data sets must be kept very small. It also implies that the nature of the computation and the expected results must be specified in great detail.
- The memory size and run time requirements must be easily adjustable to accommodate new systems with increased power.
- The benchmark must be readily distributable.

In our view, the only benchmarking approach that satisfies all of these constraints is a "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. The person or persons implementing the benchmarks on a given system are expected to solve the various problems in the most appropriate way for the specific system. The choice of data structures, algorithms, processor allocation and memory usage are all (to the extent allowed by the specification) left open to the discretion of the implementer. Some extension of Fortran or C is required, 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.

To this end, we have devised a number of relatively simple "kernels", which are specified completely in [6]. However, kernels alone are insufficient to completely assess the performance potential of a parallel machine on real scientific applications. The chief difficulty is that a certain data structure may be very efficient on a certain system for one of the isolated kernels, and yet this data structure would be inappropriate if incorporated into a larger application. In other words, the performance of a real computational fluid dynamics (CFD) application on a parallel system is critically dependent on data motion between computational kernels. Thus we consider the complete reproduction of this data movement to be of critical importance in a benchmark.

Our benchmark set therefore consists of two major components: five parallel kernel benchmarks and three simulated application benchmarks. The simulated application benchmarks combine several computations in a manner that resembles the actual order of execution in certain important CFD application codes. This is discussed in more detail in [6].

We feel that this benchmark set successfully addresses many of the problems associated with benchmarking parallel machines. Although we do not claim that this set is typical of all scientific computing, it is based on the key components of several large aeroscience applications used by scientists on supercomputers at NASA Ames Research Center. These benchmarks will be used by the Numerical Aerodynamic Simulation (NAS) Program to evaluate the performance of parallel computers.

2 Benchmark Rules

2.1 Definitions

In the following, the term "processor" is defined as a hardware unit capable of integer and floating point computation. The "local memory" of a processor refers to randomly accessible memory with an access time (latency) of less than one microsecond. The term "main memory" refers to the combined local memory of all processors. This includes any memory shared by all processors that can be accessed by each processor in less than one microsecond. The term "mass storage" refers to non-volatile randomly accessible storage media that can be accessed by at least one processor within forty milliseconds. A "processing node" is defined as a hardware unit consisting of one or more processors plus their local memory, which is logically a single unit on the network that connects the processors.

The term "computational nodes" refers to those processing nodes primarily devoted to high-speed floating point computation. The term "service nodes" refers to those processing nodes primarily devoted to system operations, including compilation, linking and communication with external computers over a network.

2.2 General Rules

Implementations of these benchmarks must be based on either Fortran-77 or C, although a wide variety of parallel extensions are allowed. This requirement stems from the observation that Fortran and C are the most commonly used programming languages by the scientific parallel computing community at the present time. If in the future other languages gain wide acceptance in this community, they will be considered for inclusion in this group. Assembly language and other low-level languages and constructs may not be used, except that certain specific vendor-supported assembly-coded library routines may be called (see section 2.3).

We are of the opinion that such language restrictions are necessary, because otherwise considerable effort would be made by benchmarkers in low-level or assembly-level coding. Then the benchmark results would tend to reflect the amount of programming resources available to the benchmarking organization, rather than the fundamental merits of the parallel system. Certainly the mainstream scientists that these parallel computers are intended to serve will be coding applications at the source level, almost certainly in Fortran C, and thus these benchmarks are designed to measure the performance that can be expected from such code.

Accordingly, the following rules must be observed in any implementations of the NAS Parallel Benchmarks:

- All floating point operations must be performed using 64-bit floating point arithmetic.
- All benchmarks must be coded in either Fortran-77 [1] or C [3], with certain approved extensions.
- Implementation of the benchmarks may not mix Fortran-77 and C code — one or the other must be used.
- Any extension of Fortran-77 that is in the Fortran-90 draft dated June 1990 or later [2] is allowed.
- Any extension of Fortran-77 that is in the Parallel Computer Fortran (PCF) draft dated March 1990 or later [5] is allowed.
- Any language extension or library routine that is employed in any of the benchmarks must be supported by the vendor and available to all users.
- Subprograms and library routines not written in Fortran or C may only perform certain functions, as indicated on the next section.
- All rules apply equally to subroutine calls, language extensions and compiler directives (i.e. special comments).

2.3 Allowable Language Extensions and Library Routines

The following language extensions and library routines are permitted:

- Constructs that indicate sections of code that can be executed in parallel or loops that can be distributed among different computational nodes.

- Constructs that specify the allocation and organization of data among or within computational nodes.
- Constructs that communicate data between processing nodes.
- Constructs that communicate data between the computational nodes and service nodes.
- Constructs that rearrange data stored in multiple computational nodes, including constructs to perform indirect addressing and array transpositions.
- Constructs that synchronize the action of different computational nodes.
- Constructs that initialize for a data communication or synchronization operation that will be performed or completed later.
- Constructs that perform high-speed input or output operations between main memory and the mass storage system.
- Constructs that perform any of the following array reduction operations on an array either residing within a single computational node or distributed among multiple nodes: $+, \times$, `MAX, MIN, AND, OR, XOR`.
- Constructs that combine communication between nodes with one of the operations listed in the previous item.
- Constructs that perform any of the following computational operations on arrays either residing within a single computational node or distributed among multiple nodes: dense matrix-matrix multiplication, dense matrix-vector multiplication and one-dimensional, two-dimensional or three-dimensional fast Fourier transforms. Such routines must be callable with general array dimensions.

3 The Benchmarks: A Condensed Overview

After an evaluation of a number of large scale CFD and computational aerosciences applications on the NAS supercomputers at NASA Ames, five medium-sized computational problems were selected as the "parallel kernels".

In addition to these problems, three different implicit solution schemes were added to the benchmark set. These schemes are representative of CFD codes currently in use at NASA Ames Research Center in that they mimic the computational activities and data motions typical of real CFD applications. They do not include the typical pre- and postprocessing of real applications, nor do they include I/O. Boundary conditions are also handled in a greatly simplified manner. For a detailed discussion on the differences between the simulated application benchmarks and real CFD applications, see Chapter 3 of [6].

Even the five parallel kernel benchmarks involve substantially larger computations than many previous benchmarks, such as the Livermore Loops or Linpack, and therefore they are more appropriate for the evaluation of parallel machines. They are sufficiently simple that they can be implemented on a new system without unreasonable effort and delay. The three simulated application benchmarks require somewhat more effort to implement but constitute a rigorous test of the usability of a parallel system to perform state-of-the-art CFD computations.

3.1 The Eight Benchmark Problems

The following gives an overview of the benchmarks. The first five are the parallel kernel benchmarks, and the last three are the simulated application benchmarks. Space does not permit a complete description for all of these. A detailed description of these benchmark problems is given in [6].

EP: An "embarrassingly parallel" kernel, which evaluates an integral by means of pseudorandom trials. This kernel, in contrast to others in the list, requires virtually no interprocessor communication.

MG: A simplified multigrid kernel. This requires highly structured long distance communication and tests both short and long distance data communication.

CG: 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, employing unstructured matrix vector multiplication.

FT: A 3-D partial differential equation solution using FFTs. This kernel performs the essence of many "spectral" codes. It is a rigorous test of long-distance communication performance.

IS: A large integer sort. This kernel performs a sorting operation that is important in "particle method" codes. It tests both integer computation speed and communication performance.

LU: A regular-sparse, block (5×5) lower and upper triangular system solution. 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". This problem exhibits a somewhat limited amount of parallelism compared to the next two.

SP: Solution of multiple, independent systems of non diagonally dominant, scalar, pentadiagonal equations. SP and the following problem 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.

BT: Solution of multiple, independent systems of non diagonally dominant, block tridiagonal equations with a (5×5) block size.

3.2 The Embarrassingly Parallel Benchmark

In order to give the reader a flavor of the problem descriptions in [6], a detailed definition will be given for the first problem, the "embarrassingly parallel" benchmark:

Set $n = 2^{28}$ and $s = 271828183$. Generate the pseudorandom floating point values r_j in the interval (0, 1) for $1 \leq j \leq 2n$ using the scheme described below. Then for $1 \leq j \leq n$ set $x_j = 2r_{2j-1} - 1$ and $y_j = 2r_{2j} - 1$. Thus x_j and y_j are uniformly distributed on the interval $(-1, 1)$.

Next set $k = 0$, and beginning with $j = 1$, test to see if $t_j = x_j^2 + y_j^2 \leq 1$. If not, reject this pair and proceed to the next j. If this inequality holds, then set $k \leftarrow k + 1$, $X_k = x_j\sqrt{(-2\log t_j)/t_j}$ and $Y_k = y_j\sqrt{(-2\log t_j)/t_j}$, where log denotes the natural logarithm. Then X_k and Y_k are independent

Gaussian deviates with mean zero and variance one. Approximately $n\pi/4$ pairs will be constructed in this manner.

Finally, for $0 \leq l \leq 9$ tabulate Q_l as the count of the pairs (X_k, Y_k) that lie in the square annulus $l \leq \max(|X_k|, |Y_k|) < l + 1$, and output the ten Q_l counts. Each of the ten Q_l counts must agree exactly with reference values.

The $2n$ uniform pseudorandom numbers r_j mentioned above are to be generated according to the following scheme: Set $a = 5^{13}$ and let $x_0 = s$ be the specified initial "seed". Generate the integers x_k for $1 \leq k \leq 2n$ using the linear congruential recursion

$$x_{k+1} \quad = \quad a x_k \pmod{2^{46}}$$

and return the numbers $r_k = 2^{-46} x_k$ as the results. Observe that $0 < r_k < 1$ and the r_k are very nearly uniformly distributed on the unit interval.

An important feature of this pseudorandom number generator is that any particular value x_k of the sequence can be computed directly from the initial seed s by using the binary algorithm for exponentiation, taking remainders modulo 2^{46} after each multiplication. The importance of this property for parallel processing is that numerous separate segments of a single, reproducible sequence can be generated on separate processors of a multiprocessor system. Many other widely used schemes for pseudorandom number generation do not possess this important property.

Additional information and references for this benchmark problem are given in [6].

4 Sample Codes

The intent of the NAS Parallel Benchmarks report is to completely specify the computation to be carried out. Theoretically, a complete implementation, including the generation of the correct input data, could be produced from the information in this paper. However, the developers of these benchmarks are aware of the difficulty and time required to generate a correct implementation from scratch in this manner. Furthermore, despite several reviews, ambiguities in the technical paper may exist that could delay implementations.

In order to reduce these difficulties and to aid the benchmarking specialist, Fortran-77 computer programs implementing the benchmarks are available.

These codes are to be considered examples of how the problems could be solved on a single processor system, rather than statements of how they should be solved on an advanced parallel system. The sample codes actually solve scaled down versions of the benchmarks that can be run on many current generation workstations. Instructions are supplied in comments in the source code on how to scale up the program parameters to the full size benchmark specifications.

These programs, as well as the benchmark document itself, are available from the following address: Applied Research Branch, NAS Systems Division, Mail Stop T045-1, 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.

Table 1 lists approximate run times and memory requirements of the sample code problems, based one processor Cray Y-MP implementations. Table 2 contains similar information for the full-sized benchmark problems. The unit "Mw" in tables 1 and 2 refers to one million 64-bit words. Note that performance in MFLOPS is meaningless for the integer sort (IS) benchmark and is therefore not given. An explanation of the entries in the problem size column can be found in the corresponding sections describing the benchmarks in [6].

5 Submission of Benchmark Results

It should be emphasized again that the sample codes described in section 4 are not the benchmark codes, but only implementation aids. For the actual benchmarks, the sample codes must be scaled to larger problem sizes. The sizes of the current benchmarks were chosen so that implementations are possible on currently available supercomputers. As parallel computer technology progresses, future releases of these benchmarks will specify larger problem sizes.

The authors and developers of these benchmarks encourage submission of performance results for the problems listed in Table 2. Periodic publication

Table 1: **NAS Parallel Benchmarks Sample Codes.** (Times and MFLOPS for one processor of the Cray Y-MP)

Benchmark code	Problem Size	Memory (Mw)	Time (sec)	MFLOPS
Embarrassingly parallel (EP)	2^{24}	0.1	11.6	120
Multigrid (MG)	32^3	0.1	0.1	128
Conjugate gradient (CG)	$\approx 10^5$	0.6	1.2	63
3-D FFT PDE (FT)	64^3	2.0	1.2	160
Integer sort (IS)	2^{16}	0.3	0.2	NA
LU solver (LU)	12^3	0.3	3.5	28
Pentadiagonal solver (SP)	12^3	0.2	7.2	24
Block tridiagonal solver (BT)	12^3	0.3	7.2	34

Table 2: **NAS Parallel Benchmarks Problem Sizes.** (Times and MFLOPS for one processor of the Cray Y-MP)

Benchmark code	Problem Size	Memory (Mw)	Time (sec)	MFLOPS
Embarrassingly parallel (EP)	2^{28}	1	151	147
Multigrid (MG)	256^3	57	54	154
Conjugate gradient (CG)	$\approx 2 * 10^6$	12	22	70
3-D FFT PDE (FT)	$256^2 \times 128$	59	39	192
Integer sort (IS)	2^{23}	26	21	NA
LU solver (LU)	64^3	8	344	189
Pentadiagonal solver (SP)	64^3	6	806	175
Block tridiagonal solver (BT)	64^3	6	923	192

of the submitted results is planned. Benchmark results should be submitted to the Applied Research Branch, NAS Systems Division, Mail Stop T045-1, NASA Ames Research Center, Moffett Field, CA 94035, attn: NAS Parallel Benchmark Results. A complete submission of results should include the following:

- A detailed description of the hardware and software configuration used for the benchmark runs.
- A description of the implementation and algorithmic techniques used.
- Source listings of the benchmark codes.
- Output listings from the benchmarks.

6 Acknowledgments

The conception, planning, execution, programming and authorship of the NAS Parallel Benchmarks was truly a team effort, with significant contributions by a number of persons. Thomas Lasinski, chief of the NAS Applied Research Branch (RNR), and John Barton of the NAS System Development Branch (RND), provided overall direction and management of the project. David Bailey and Horst Simon edited the benchmark document and worked with others in the development and implementation of the benchmarks. Eric Barszcz of RNR assisted in the implementation of both the multigrid and the simulated application benchmarks. David Browning and Russell Carter of RND reviewed all problem definitions and sample codes, as well as contributed some text to this paper. Leonardo Dagum of RNR developed the integer sort benchmark. Rod Fatoohi of RNR assisted in the development and implementation of the simulated application benchmarks. Paul Frederickson of RIACS developed the multigrid benchmark and worked with Bailey on the embarrassingly parallel and 3-D FFT PDE benchmarks. Rob Schreiber of RIACS developed the conjugate gradient benchmark and worked with Simon on its implementation. V. Venkatakrishnan of RNR assisted in the implementation of the simulated application benchmarks. Finally, Sisira Weeratunga of RNR was responsible for the overall design of the simulated application benchmarks and also for a major portion of their implementation.

References

[1] *American National Standard Programming Language Fortran X3.9-1978.* American National Standards Institute, 1430 Broadway, New York, NY, 10018, 1990.

[2] *Draft Proposed Fortran 90 ANSI Standard X3J11.159 – 1989.* American National Standards Institute, 1430 Broadway, New York, NY, 10018, 1990.

[3] *Draft Proposed C ANSI Standard X3J3 – S8115.* American National Standards Institute, 1430 Broadway, New York, NY, 10018, 1990.

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

[5] *PCF Fortran Extensions – Draft Document, Revision 2.11.* Parallel Computing Forum(PCF), c/o Kuck and Associates, 1906 Fox Drive, Champaign, Illinois 61820, March 1990.

[6] 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.

[7] D. Bailey and J. Barton. *The NAS Kernel Benchmark Program.* Technical Report 86711, NASA Ames Research Center, Moffett Field, California, August 1985.

[8] G. Cybenko, L. Kipp, L. Pointer, and D. Kuck. *Supercomputer Performance Evaluation and the Perfect Benchmarks.* Technical Report 965, CSRD, Univ. of Illinois, Urbana, Illinois, March 1990.

[9] J. Dongarra. The LINPACK Benchmark: An Explanation. *SuperComputing*, 10 – 14, Spring 1988.

[10] J. Dongarra. *Performance of Various Computers Using Standard Linear Equations Software in a Fortran Environment.* Technical Report MCSRD 23, Argonne National Laboratory, March 1988.

[11] M. Berry et al. The Perfect Club Benchmarks: Effective Performance Evaluation of Supercomputers. *The International Journal of Supercomputer Applications*, 3:5 – 40, 1989.

[12] F. McMahon. *The Livermore Fortran Kernels: A Computer Test of the Numerical Performance Range.* Technical Report UCRL - 53745, Lawrence Livermore National Laboratory, Livermore, California, December 1986.

[13] L. Pointer. *PERFECT Report 1.* Technical Report 896, CSRD, Univ. of Illinois, Urbana, Illinois, July 1989.

[14] L. Pointer. *PERFECT Report 2: Performance Evaluation for Cost-Effective Transformations.* Technical Report 964, CSRD, Univ. of Illinois, Urbana, Illinois, March 1990.