TITLE: The NAS Parallel Benchmarks

AUTHOR: David H Bailey[1]

ACRONYMS: NAS, NPB

DEFINITION:

The NAS Parallel Benchmarks (NPB) are a suite of parallel computer performance benchmarks. They were originally developed at the NASA Ames Research Center in 1991 to assess high-end parallel supercomputers [**?**]. Although they are no longer used as widely as they once were for comparing high-end system performance, they continue to be studied and analyzed a great deal in the high-performance computing community. The acronym "NAS" originally stood for the Numerical Aeronautical Simulation Program at NASA Ames. The name of this organization was subsequently changed to the Numerical Aerospace Simulation Program, and more recently to the NASA Advanced Supercomputing Center, although the acronym remains "NAS." The developers of the original NPB suite were David H. Bailey, Eric Barszcz, John Barton, David Browning, Russell Carter, LeoDagum, Rod Fatoohi, Samuel Fineberg, Paul Frederickson, Thomas Lasinski, Rob Schreiber, Horst Simon, V. Venkatakrishnan and Sisira Weeratunga.

DISCUSSION:

The original NAS Parallel Benchmarks consisted of eight individual benchmark problems, each of which focused on some aspect of scientific computing. The principal focus was in computational aerophysics, although most of these benchmarks have much broader relevance, since in a much larger sense they are typical of many real-world scientific computing applications.

The NPB suite grew out of the need for a more rational procedure to select new supercomputers for acquisition by NASA. The emergence of commercially available highly parallel computer systems in the late 1980s offered an attractive alternative to parallel vector supercomputers that had been the mainstay of high-end scientific computing. However, the introduction of highly parallel systems was accompanied by a regrettable level of hype, not only on the part of the commercial vendors but even, in some cases, by scientists using the systems. As a result, it was difficult to discern whether the new systems offered any fundamental performance advantage over vector supercomputers, and, if so, which of the parallel offerings would be most useful in real-world scientific computation.

[1]Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA, dhbailey@lbl.gov. Supported in part by the Director, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy, under contract number DE-AC02-05CH11231.

In part to draw attention to some of the performance reporting abuses prevalent at the time, the present author wrote a humorous essay "Twelve Ways to Fool the Masses," which described in a light-hearted way a number of the questionable ways in which both vendor marketing people and scientists were inflating and distorting their performance results [**?**]. All of this underscored the need for an objective and scientifically defensible measure to compare performance on these systems.

At the time (1991), the only widely available high-end benchmark was the scalable Linpack benchmark, which while it was useful and remains useful to this day, was not considered typical of most applications on NASA's supercomputers or at most other sites. One possible solution was to employ an actual large-scale application code, such as one of those being used by scientists using supercomputers at the NAS center. However, due in part to the very large programming and tuning requirements, these were considered too unwieldy to be used to compare a broad range of emerging parallel systems. Compounding these challenges at this time was the lack of a generally accepted parallel programming model — note this was several years before the advent of the Message Passing Interface (MPI) and OpenMP models that are in common usage today.

For these reasons, the NAS Benchmarks were initially designed not as a set of computer codes, but instead were specified as "paper and pencil" benchmarks defined in a technical document [**?**]. The idea was to specify a set of problems only algorithmically, but in sufficient detail that the document could be used for a full-fledged implementation complying with the requirements. Even the input data or a scheme to generate it was specified in the document. Some "reference implementations" were provided, but these were intended only as aids for a serious implementation, not to be the benchmarks themselves. The original rules accompanying the benchmarks required that the benchmarks be implemented in some extension of Fortran or C (see details below), but otherwise implementers were 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 were (and 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." 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 three 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, and in many other three-dimension physical simulations as well. 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 in three physical dimensions, and thus are more typical of real scientific applications.

Benchmark Rules

Even though the benchmarks are specified in a technical document, and implementers are generally free to code them in any reasonable manner, certain rules were specified for these implementations. The intent here was to limit implementations to what would be regarded as "reasonable" code similar to that used in real scientific applications. In particular, the following rules were presented, and, with a couple of minor changes, are still in effect today (these rules are the current version):

- All floating point operations must be performed using 64-bit floating point arithmetic (at least).
- All benchmarks must be coded in either Fortran (Fortran-77 or Fortran-90), C or Java, with certain approved extensions. Java was added in a recent version of the NPB.
- One of the three languages must be selected for the entire implementation — mixed code is not allowed.
- Any language extension or library routine that is employed in any of the benchmarks must be supported by the system vendor and available to all users.
- Subprograms and library routines not written in Fortran, C or Java (such as assembly-coded routines) may only perform certain basic functions [a complete list is provided in the full NPB specification].
- All rules apply equally to subroutine calls, language extensions and compiler directives (i.e. special comments).

The Original Eight Benchmarks

A brief (and necessarily incomplete) description of the eight problems is given here. For full details, see the full NPB specification document [**?**].

EP. As the acronym suggests, this is an "embarrassingly parallel" kernel — in contrast to others in the list, it requires virtually no interprocessor communication, only coordination of pseudorandom number generation at the beginning and collection of results at the end. There is some challenge in computing required intrinsic functions (which, according to specified rules, must be done using vendor-supplied library functions) at a rapid rate.

The problem is to generate pairs of Gaussian random deviates and tabulate the number of pairs in successive square annuli. The Gaussian deviates are to

be calculated by the following well-known scheme. First, generate a sequence of n pairs of uniform $(0,1)$ pseudorandom deviates (x_j, y_j) (using a specific linear congruential scheme specified in the benchmark document). Then for each j check whether $t_j = x_j^2 + y_j^2 \leq 1$. If so, set $X_k = x_j\sqrt{(-2\log t_j)/t_j}$ and $Y_k = y_j\sqrt{(-2\log t_j)/t_j}$, where the index k is incremented with every successful test; if $t_j > 1$, then reject the pair (x_j, y_j). In this way, the resulting pairs (X_k, Y_k) are random deviates with a Gaussian distribution. The sums $S_1 = \sum_k X_k$ and $S_2 = \sum_k Y_k$ are each accumulated, as are the counts of the number of hits in successive square annuli.

The verification test for this problem requires that the sums S_1 and S_2 each agree with reference values to within a specified tolerance, and also that the ten counts of deviates in square annuli exactly agree with reference values.

MG. This is a simplified multigrid calculation. This requires highly structured long distance communication and tests both short and long distance data communication.

The problem definition is as follows. Set $v = 0$ except at the twenty specified points, where $v = \pm 1$. Commence an iterative solution with $u = 0$. Each of the four iterations consists of the following two steps, in which $k = 8$:

$$\begin{aligned} r &:= vAu \qquad \text{(evaluate residual)} \\ u &:= u + M^k r. \qquad \text{(apply correction)} \end{aligned}$$

Here M^k denotes the following V-cycle multigrid operator: $z_k := M^k r_k$, where if $k > 1$ then

$$\begin{aligned} r_{k-1} &:= Pr_k \qquad \text{(restrict residual)} \\ z_{k-1} &:= M^{k-1} r_{k-1} \qquad \text{(recursive solve)} \\ z_k &:= Qz_{k-1} \qquad \text{(prolongate)} \\ r_k &:= r_k - Az_k \qquad \text{(evaluate residual)} \\ z_k &:= z_k + Sr_k, \qquad \text{(apply smoother)} \end{aligned}$$

else

$$z_1 := Sr_1. \qquad \text{(apply smoother)}$$

The coefficients of the P, M, Q and S arrays, together with other details, are given in the benchmark document.

The benchmark problem definition requires that a specified number of iterations of the above V-cycle be performed, after which the L_2 norm of the r array must agree with a reference value to within a specified tolerance.

CG. In this problem, 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.

The problem statement in this case is fairly straightforward: Perform a specified number of conjugate gradient iterations in approximating the solution z to a certain specified large sparse $n \times n$ linear system of equations $Az = x$. In this problem, the matrix A must be used explicitly. This is because after the original NPB suite was published, it was noted that by saving the random sparse vectors x used in the specified construction of the problem, it was possible to reformulate the sparse matrix-vector multiply operation in such a way that communication is substantially reduced. Therefore this scheme is specifically disallowed.

The verification test is that the value $\gamma = \lambda + 1/(x^T z)$ (where λ is a parameter that depends on problem size) must agree with a reference value to a specified tolerance.

FT. Here, a 3-D partial differential equation is solved using FFTs. This performs the essence of many "spectral" codes. It is a rigorous test of heavy long-distance communication performance.

The problem is to numerically solve the Poisson partial differential equation (PDE)

$$\frac{\partial u(x,t)}{\partial t} = \alpha \nabla^2 u(x,t),$$

where x is a position in three-dimensional space. When a Fourier transform is applied to each side, this equation becomes

$$\frac{\partial v(z,t)}{\partial t} = -4\alpha\pi^2 |z|^2 v(z,t),$$

where $v(z,t)$ is the Fourier transform of $u(x,t)$. This has the solution

$$v(z,t) = e^{-4\alpha\pi^2|z|^2 t} v(z,0).$$

The benchmark problem is to solve a discrete version of the original PDE by computing the forward 3-D discrete Fourier transform (DFT) of the original state array $u(x,0)$, multiplying the results by certain exponentials, and then performing an inverse 3-D DFT. Of course, the DFTs can be rapidly evaluated by using a 3-D fast Fourier transform (FFT) algorithm.

The verification test for this problem is to match the checksum of a certain subset of the final array with reference values.

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

The specific problem is to generate a large array by a certain scheme and then to sort it. Any efficient parallel sort scheme may be used. The verification test is to certify that the array is in sorted order (full details are given in the benchmark document).

LU. This performs a synthetic computational fluid dynamics (CFD) calculation by solving regular-sparse, block (5×5) lower and upper triangular systems.

SP. This performs a synthetic CFD problem by solving multiple, independent systems of non diagonally dominant, scalar, pentadiagonal equations.

BT. This performs a synthetic CFD problem by solving multiple, independent systems of non diagonally dominant, block tridiagonal equations with a (5×5) block size.

These last three "simulated CFD benchmarks" together represent the heart of the computationally-intensive building blocks of CFD programs in most common use today for the numerical solution of three-dimensional Euler/Navier-Stokes equations using finite-volume, finite-difference discretization on structured grids. LU and SP involve global data dependencies. Although the three benchmarks are similar in many respects, there is a fundamental difference with regard to the communication-to-computation ratio. BT represents the computations associated with the implicit operator of a newer class of implicit CFD algorithms. This kernel exhibits somewhat more limited parallelism compared with the other two.

In each of these three benchmarks, the same high-level synthetic problem is solved. This synthetic problem differs from a real CFD problem in the following important aspects:

1. Absence of realistic boundary algorithms.
2. Higher than normal dissipative effects.
3. Lack of upwind differencing effects.
4. Absence of geometric stiffness introduced through boundary conforming coordinate transformations and highly stretched meshes.
5. Lack of evolution of weak solutions found in real CFD applications during the iterative process.
6. Absence of turbulence modeling effects.

Full details of the these three benchmark problems are given in the benchmark document, as are the verification tests.

Evolution of the NAS Parallel Benchmarks

The original NPB suite was accompanied by a set of "reference" implementations, including implementations for a single-processor workstation, an Intel Paragon system (using Intel's message passing library) and a CM-2 from Thinking Machines Corp. Also, the original release defined three problem sizes: Class W (for the sample workstation implementation), Class A (for what was at the time a moderate-sized parallel computer), and Class B (for what was at the time a large parallel computer). The Class B problems were roughly four times larger than the Class A problems, both in total operation count and in aggregate memory requirement.

After the initial release of the NPB, the NASA team published several sets of performance results, for example [**?**, **?**], and in the next few years numerous teams of scientists and computer vendors submitted results.

By 1996, several of the original NPB team had left NASA Ames Research Center, and, for a while, active support and collection of results lagged. Fortunately, some other NASA Ames scientists, notably William Saphir, Rob Van der Wingaart, Alex Woo and Maurice Yarrow, stepped in to continue support and development. These researchers produced a reference implementation of the NPB with MPI constructs, which, together with a few minor changes, was designated NPB 2.0. Then a Class C problem size was defined as part of NPB 2.2.

Another enhancement was the addition of the "BT I/O" benchmark. In this benchmark, implementers were required to output some key arrays to an external file system as the BT program is running. Thus, by comparing the performance of the BT and BT I/O benchmarks, one could get some measure of I/O system performance. This change, together with the definition of Class D problem sizes, was released in 1997 and designated NPB 2.4.

In 1999, NASA Ames researchers Haoqiang Jin, Michael Frumkin and Jerry Yan released NPB 3.0, which included an OpenMP reference implementation of the NPB.

In 2002, Rob van der Wingaart and Michael Frumkin released a grid version of the NPB. In the next year or two several additional minor improvements and bug fixes were made to all reference implementations, and Class E problem sizes were defined. Reference implementations were released in Java 3.0 and High-Performance Fortran, and a "multi-zone" benchmark, reflective of a number of more modern CFD computations, was added. These changes were designated NPB 3.3, which is the latest version on the NASA website.

Full details of the current version of NPB, as well as the actual documents and reference implementations, are available at:
`http://www.nas.nasa.gov/Resources/Software/npb.html`

RELATED REFERENCES:
The Linpack benchmark
The HPC Challenge benchmarks

References

[1] D. H. Bailey, "Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers," *Supercomputing Review*, Aug. 1991, pg. 54–55. Also published in *Supercomputer*, Sep. 1991, pg. 4–7. Available at `http://crd.lbl.gov/~dhbailey/dhbpapers/twelve-ways.pdf`.

[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), pg. 63–73.

[3] 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," Technical Report RNR-94-007, NASA Ames Research Center, Moffett Field, CA 94035, Jan. 1991, revised 1994, available at `http://www.nas.nasa.gov/News/Techreports/1994/PDF/RNR-94-007.pdf`.

[4] David H. Bailey, Eric Barszcz, Leo Dagum and Horst D. Simon, "NAS Parallel Benchmark Results," *Proceedings of Supercomputing 1992*, Nov. 1992, pg. 386–393.

[5] David H. Bailey, Eric Barszcz, Leo Dagum and Horst D. Simon, "NAS Parallel Benchmark Results," *IEEE Parallel and Distributed Technology*, Feb. 1993, pg. 43–51.