NAS Parallel Benchmarks Version 2.4

Rob F. Van der Wijngaart
Computer Sciences Corporation
NASA Advanced Supercomputing (NAS) Division
NASA Ames Research Center, Moffett Field, CA 94035-1000
wijngaar@nas.nasa.gov

NAS Technical Report NAS-02-007

October 2002

Abstract

We describe a new problem size, called Class D, for the NAS Parallel Benchmarks (NPB), whose MPI source code implementation is being released as NPB 2.4. A brief rationale is given for how the new class is derived. We also describe the modifications made to the MPI implementation to allow the new class to be run on systems with 32-bit integers, and with moderate amounts of memory. Finally, we give the verification values for the new problem size.

1 Introduction

The most recent version of the NAS Parallel Benchmarks [1, 2], which introduced a set of problems collectively called Class C, was released in 1996. Since then, high-performance computer systems have grown significantly in size and capabilities. Cache and memory sizes have increased, clock rates have gone up, compiler technology has improved, network bandwidths have increased, total numbers of processors have gone up, wide-area computations involving multiple large systems (a.k.a. grid computing) has become in vogue. Consequently, the need has arisen to specify more challenging benchmark sizes to rate the performance of computer systems. The new problem suite described here, collectively called Class D, aims to provide that challenge.

2 Rationale

We will not attempt to formulate a reasoning to justify *a posteriori* the various problem sizes that were selected for the NAS Parallel Benchmarks. Little was known at the time of these initial releases about what constitutes rational choices for problem sizes. But we do now adopt a rationale for formulating new classes that are to be run on available high-performance computing systems:

- A benchmark program of the largest problem size should, at the time of its release, run in approximately the same amount of wall clock time on available systems as the previous largest size did at the time of its release.
- The ratio of data set size over cache size for a benchmark program of the largest problem size should, at the time of its release, roughly equal that of the previous largest size at the time of its release.

While these are not very solid rules, they provide us with some decent guidelines for scaleup. Disregarding the very largest systems in the world, as ranked in the list of the 500 best-performing systems [3] executing the LINPACK benchmark, we determined that high-performance computing systems have progressed roughly as follows, relative to the time of the release of NPB Class C:

- system size (number of processors): factor of 2.5
- processor speed: factor of 20
- L2 cache size: factor of 10-30

However, these performance enhancements did not all occur within a single system. For example, the systems occupying the fifth place on the top 500 list in november 1996 and november 2001 contained 3000 and 4000 processors, respectively.

From this we derive the following basic scaleup rule: The data set of each benchmark problem (save Embarrassingly Parallel EP, which does not operate on a data set per se) is increased by a factor of 16 (this particular multiplication factor supports those benchmarks that operate on data sets whose sizes must be powers of two, i.e. Multi-Grid (MG), and Fast Fourier Transform (FT)), and the number of iterations or time steps is increased by a factor of 1.25.

3 Problem parameters

When specifying the parameters of the benchmark problems, it is important to ensure that numerical stability is preserved, but also that none of the benchmarks converges to a steady state prematurely. Convergence means computations can be stopped without affecting correctness of the solution.

We next discuss the changes to the parameters of each individual benchmark problem, including verification values. Any parameters not discussed remain unchanged from the Class C NPB definition, including those required for any initialization of field data.

3.1 Multigrid–MG

The size of the finest of the discretization grids is set to 1024^3 points, and the number of V-cycles to be completed is set to 50. Within the accuracy specified in [1], the norm of the final discrete solution, as defined also in [1], should equal $0.158327506043 * 10^{-9}$.

3.2 Conjugate Gradient–CG

The number of rows in the sparse matrix is set to $1.5 * 10^6$, the number of nonzeroes per row is 21, the number of eigenvalue estimates to be computed (number of outer iterations) is 100, and the eigenvalue shift term λ is 500. Within the accuracy specified in [1], the solution aggregated in the variable ζ, as defined in [1], should equal 52.5145321058.

3.3 Fast Fourier Transform–FT

The size of the discretization grid is set to $2048 \times 1024 \times 1024$ points, and the number of time steps in the evolution of the double-precision complex solution is set to 25. Since FT solutions can be evolved independently, we need to supply verification values for each individually, consisting of the double-precision complex checksum, as defined in [1]. Within the accuracy specified in [1], these values are listed in Table 1.

3.4 Integer Sort–IS

We do not provide a specification of the Integer Sort benchmark for Class D.

3.5 Embarrassingly Parallel–EP

The Embarrassingly Parallel benchmark problem is the only one within NPB in which the amount of work is not proportional to the data set size. Hence, we simply increase the work by a factor of 16 over that of the Class C problem, that is, we compute 2^{36} pairs of random numbers.

In keeping with the NPB Message Passing Interface (MPI) implementation, version 2.3, the only quantity that is checked in the verification test is the vector of aggregates of the random numbers, i.e. $(\sum_{k=1}^{10} X_k, \sum_{k=1}^{10} Y_k)$, where X_k and Y_k are as defined in [1]. Within the accuracy also specified in [1], the vector of aggregates must equal $(1.982481200946593 * 10^5, -1.020596636361769 * 10^5)$.

3.6 Block-Tridiagonal–BT

The size of the discretization grid is set to $408 \times 408 \times 408$, and the number and the size of the time steps to 250 and $2.0 * 10^{-5}$, respectively. Verification values for the residual and error norms, as computed according to [1], and within the accuracy also specified in [1], are listed in Table 2.

3.7 Scalar-Pentadiagonal–SP

The size of the discretization grid is set to $408 \times 408 \times 408$, and the number and the size of the time steps to 500 and $3.0 * 10^{-4}$, respectively. Verification values for the residual and error norms, as computed according to [1], and within the accuracy also specified in [1], are listed in Table 3.

3.8 Lower-Upper Symmetric Gauss-Seidel–LU

The size of the discretization grid is set to $408 \times 408 \times 408$, and the number and the size of the time steps to 300 and 1.0, respectively. Verification values for the residual and error norms, as computed according to [1], and within the accuracy also specified in [1], are listed in Table 4. The value of the discrete surface integral, again as computed according to [1] and within the accuracy specified in [1], is $0.8334101392503 * 10^2$.

4 FT Implementation details

The NPB, version 2.3, MPI implementation of FT had to be changed somewhat to accommodate the larger data set size of Class D. The changes have largely to do with the desire to be able to run the NPB on systems that do not support 64-bit integers. Addressing very large arrays and defining large integer constants is a problem on such systems. We (partially) circumvent these problems by making use of the fact that the MPI implementation uses domain decomposition and reduced local address spaces for parallelization. Hence, if the problem is solved on a large enough number of processors, all integers required can be represented in 32 bits. The minimum number of processors on which the problem can be solved successfully depends slightly on the system's representation of signed integers. Usually, four to eight processors suffice.

Avoiding integer overflow

The NPB version 2.3 MPI implementation of FT evaluates the number of grid points per processor by computing the total grid size and dividing the result by the number of processors, i.e. $npts = (nx \times ny \times nz)/nprocs$. To void integer overflow, we do not compute and store the total grid size explicitly, but make use of the fact that the grid cross-sectional size is large, and that all grid dimensions as well as the number of processors employed are powers of two: $npts = ((nx \times ny)/nprocs) \times nz$.

Furthermore, the function $ipow46$, which is used in the initialization of the solution of FT, may suffer errors if applied unmodified to solve the Class D problem. The reason is that one of its integer arguments may assume very large values for numbers of processors that are large with respect to the number of grid points in the third spatial dimension. In that case, the integer argument is close to twice the total grid size. We remedy this situation by pre-factoring the integer argument into a modest power of two ($2nx$) and a leftover term, wherever $ipow46$ is called.

Reducing memory required

Completion of the FT benchmark involves the evolution of the Fourier transform of the discrete solution by multiplying each element with some exponential $e^{\tau(i,j,k)*t}$, where τ is a function of the discrete coordinates (i,j,k) of the grid point, and t signifies the sequence number of the time step. In NPB, version 2.3, the exponential was stored in a table for all values of i, j, k, and t that might occur. This approach minimizes the number of evaluations of transcendental functions, and the number of branchings required to compute the function τ. However, it comes at the expense of having to store on each processor an array of size $t_{max}(nx^2 + ny^2 + nz^2)/4$, where t_{max} is the total number of time steps in the computation (this particular size follows from the shape of the grid-dependent portion of the argument of τ, see [1, p.24]). Clearly, this approach is not scalable, and leads to space problems on systems with modest amounts of memory. We solve the problem by only reserving space for, and precomputing, the term $e^{\tau(i,j,k)}$ for those points of the grid for which the calling processor is responsible. We applied the same method to the smaller problem sizes of FT and found that, on an SGI Origin 3000 with MIPS R14000 processors, the extra cost for on-the-fly multiplication of the solution with the term $e^{\tau(i,j,k)}$ was almost completely offset by the reduction in memory accesses to the very large original array tabulating $e^{\tau(i,j,k)*t}$; no significant performance degradation was observed.

5 Conclusions

A new problem size was defined for all NAS Parallel Benchmarks, except for the Integer Sort. The new set of problems is collectively called Class D, whose specification coincides with the release of a new MPI implementation, called Version 2.4, which is available at:
`www.nas.nasa.gov/Research/Software/swinstructions.html`.
The Class D codes can be run on systems with 32-bit integers and with MPI 1.0 bindings. NAS invites performance results for all classes, but especially for Class D, to be submitted to: `npb@nas.nasa.gov`.

References

[1] D. Bailey, E. Barscz, J. Barton, D. Browning. R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, S. Weeratunga. *The NAS Parallel Benchmarks.* NAS Technical Report RNR-94-007, NASA Ames Research Center, Moffett Field, CA, 1994.

[2] D.H. Bailey, T. Harris, W.C. Saphir, R.F. Van der Wijngaart, A.C. Woo, M. Yarrow. *The NAS Parallel Benchmarks 2.0.* NAS Technical Report NAS-95-020, NASA Ames Research Center, Moffett Field, CA, 1995.

[3] `http://www.top500.org/`

Table 1: Checksum values for FT, Class D

Time step	Real part	Imaginary part
1	512.2230065252	511.8534037109
2	512.0463975765	511.7061181082
3	511.9865766760	511.7096364601
4	511.9518799488	511.7373863950
5	511.9269088223	511.7680347632
6	511.9082416858	511.7967875532
7	511.8943814638	511.8225281841
8	511.8842385057	511.8451629348
9	511.8769435632	511.8649119387
10	511.8718203448	511.8820803844
11	511.8683569061	511.8969781011
12	511.8661708593	511.9098918835
13	511.8649768950	511.9210777066
14	511.8645605626	511.9307604484
15	511.8647586618	511.9391362671
16	511.8654451572	511.9463757241
17	511.8665212451	511.9526269238
18	511.8679083821	511.9580184108
19	511.8695433664	511.9626617538
20	511.8713748264	511.9666538138
21	511.8733606701	511.9700787219
22	511.8754661974	511.9730095953
23	511.8776626738	511.9755100241
24	511.8799262314	511.9776353561
25	511.8822370068	511.9794338060

Table 2: Verification values for BT, Class D

Component	Residual norm	Error norm
1	$0.2533188551738 * 10^5$	$0.3100009377557 * 10^3$
2	$0.2346393716980 * 10^4$	$0.2424086324913 * 10^2$
3	$0.6294554366904 * 10^4$	$0.7782212022645 * 10^2$
4	$0.5352565376030 * 10^4$	$0.6835623860116 * 10^2$
5	$0.3905864038618 * 10^5$	$0.6065737200368 * 10^3$

Table 3: Verification values for SP, Class D

Component	Residual norm	Error norm
1	$0.1044696216887 * 10^5$	$0.5089471423669 * 10^1$
2	$0.3204427762578 * 10^4$	$0.5323514855894 * 10^0$
3	$0.4648680733032 * 10^4$	$0.1187051008971 * 10^1$
4	$0.4238923283697 * 10^4$	$0.1083734951938 * 10^1$
5	$0.7588412036136 * 10^4$	$0.1164108338568 * 10^2$

Table 4: Verification values for LU, Class D

Component	Residual norm	Error norm
1	$0.4868417937025 * 10^5$	$0.3752393004482 * 10^3$
2	$0.4696371050071 * 10^4$	$0.3084128893659 * 10^2$
3	$0.1218114549776 * 10^5$	$0.9434276905469 * 10^2$
4	$0.1033801493461 * 10^5$	$0.8230686681928 * 10^2$
5	$0.7142398413817 * 10^5$	$0.7002620636210 * 10^3$