NAS Parallel Benchmarks, Multi-Zone Versions

Rob F. Van der Wijngaart*, Haoqiang Jin
NASA Advanced Supercomputing (NAS) Division
NASA Ames Research Center, Moffett Field, CA 94035-1000
{wijngaar,hjin}@nas.nasa.gov

NAS Technical Report NAS-03-010

July 2003

Abstract

We describe an extension of the NAS Parallel Benchmarks (NPB) suite that involves solving the application benchmarks LU, BT and SP on collections of loosely coupled discretization meshes. The solutions on the meshes are updated independently, but after each time step they exchange boundary value information. This strategy, which is common among structured-mesh production flow solver codes in use at NASA Ames and elsewhere, provides relatively easily exploitable coarse-grain parallelism between meshes. Since the individual application benchmarks also allow fine-grain parallelism themselves, this NPB extension, named NPB Multi-Zone (NPB-MZ), is a good candidate for testing hybrid and multi-level parallelization tools and strategies.

1 Introduction

The NAS Parallel Benchmarks (NPB) [1] are well-known problems for testing the capabilities of parallel computers and parallelization tools. They exhibit mostly fine-grain exploitable parallelism and are almost all iterative, requiring multiple data exchanges between processes within each iteration. Implementations in MPI [2], Java [4], High Performance Fortran [5], and OpenMP [6] all take advantage of this fine-grain parallelism.

However, many important scientific problems feature several levels of parallelism, and this property is not reflected in NPB. To remedy this deficiency the NPB Multi-Zone (NPB-MZ) versions were created, which are described in this report. Problem sizes and verification values are given for benchmark classes S, W, A, B, C, and D.

2 General Approach

The application benchmarks Lower-Upper Symmetric Gauss-Seidel (LU), Scalar Penta-diagonal (SP), and Block Tri-diagonal (BT) solve discretized versions of the unsteady, compressible Navier-Stokes equations in three spatial dimensions. Each operates on a structured discretization mesh that is a logical cube. In realistic applications, however, a single such mesh is often not sufficient to describe a complex domain, and multiple meshes or *zones* are used to cover it. In the production code OVERFLOW [3] the flow equations are solved independently in each zone, and after each iteration the zones exchange boundary values with their immediate neighbors with which they overlap.

*Employee of Computer Sciences Corporation

2.1 Multi-Zone Mesh Systems

We take the OVERFLOW [3] approach in creating the NPB Multi-Zone versions of LU, BT, and SP, i.e. LU-MZ, BT-MZ, and SP-MZ. In each a logically rectangular discretization mesh is divided into a two-dimensional horizontal tiling of three-dimensional zones of approximately the same aggregate size as the original NPB, as indicated in Figure 1.

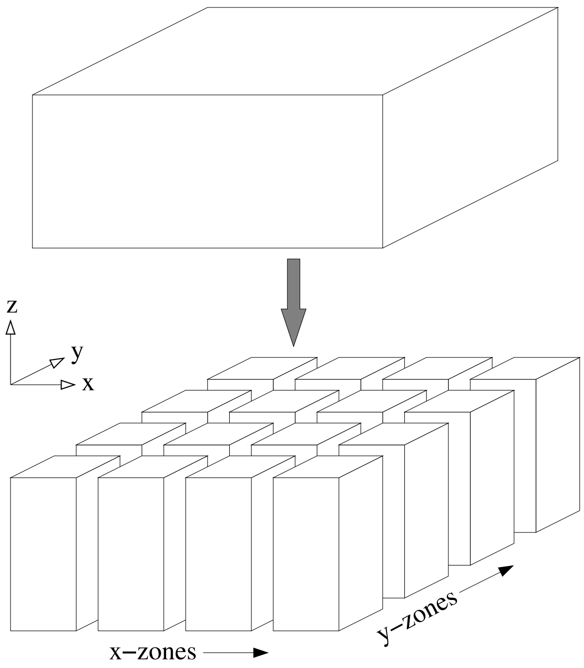

Figure 1: Two-dimensional tiling of three-dimensional mesh (exploded view).

Within all zones the LU, BT, or SP problems are solved to advance the time-dependent solution, using exactly the same methods and constants (except for mesh spacing, see Section 2.3) as described in [1, 7]. The mesh spacings of all zones of a particular problem class are identical, and the overlap between neighboring zones is exactly one such spacing, so that discretization points in overlap regions coincide exactly.

2.2 Data Exchange Between Zones

Exchange of boundary values between zones takes place after each time step, which provides the fairly loose coupling of the otherwise independent solution processes within the zones. The data transfer is as follows. Solution values at points one mesh spacing away from each vertical zone face are copied to the coincident boundary points of the neighboring zone. Values at points on zone edges and vertices are not used in the discretization formulas, so these do not need to be copied. The problem is periodic in the two horizontal directions (x and y), so donor point values at the extreme sides of the mesh system are copied to boundary points at the opposite ends of the system. This data transfer process is shown schematically in Figure 2 for a horizontal two-dimensional slice of a mesh system of two by two zones. We only show where data residing on the bottom left slice is copied. No data is copied in the third space dimension, since the problem is not periodic in that direction, nor is the overall mesh system partitioned in that direction. The bottom left slice also receives data from other slices, but that is not shown in the figure.

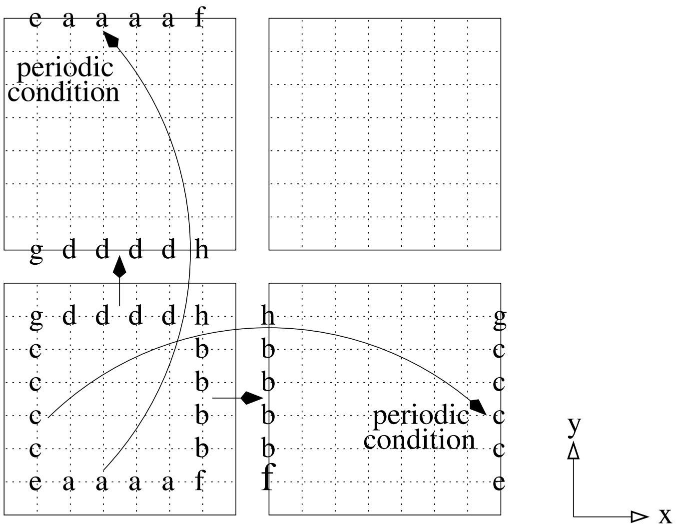

Figure 2: Data transfer among zones within horizontal slice of three-dimensional mesh system. Only copying of data from bottom left zone slice is depicted.

2.3 Aggregate Mesh Sizes

To avoid pathologically shaped zones after partitioning the overall mesh in the two horizontal directions, we change the aspect ratios of the meshes of the original NPB. The total number of points in all zones is kept approximately the same as in the original NPB, except for the smallest problem class (S). There we increase the number of points to make sure that each zone contains enough points to fit the discretization stencil. The overall mesh sizes are listed in Table 1. The mesh spacing (distance between mesh points) used in the initializations and discretizations equals the reciprocal of the average number of mesh cells within each zone in the pertinent coordinate direction. The number of cells of any zone in a certain coordinate direction equals the number of points in that direction minus one. Hence, if the total number of points and the number of zones in a certain coordinate direction are np and nz, respectively, then the mesh spacing in that direction equals $nz/(np-1)$.

	Mesh dimensions		
Class	x	y	z
S	24	24	6
W	64	64	8
A	128	128	16
B	304	208	17
C	480	320	28
D	1632	1216	34

Table 1: Total number of mesh points for all NPB multi-zone problems

3 Individual Benchmarks

The general approach described above covers most aspects of the NPB multi-zone problems. In this section we describe the differences between the three application benchmarks. We note that the selection of different NPB solvers for the new benchmarks is fairly arbitrary. The major difference between the three multi-zone problems lies in the way the zones are created out of the single overall mesh.

3.1 LU-MZ

For all problem classes the number of zones in each of the two horizontal dimensions equals four. The overall mesh is partitioned such that the zones are identical in size, which makes it relatively easy to balance the load of the parallelized application. The actual sizes of the zones are listed in Table 2.

Class	dir	# zones	size
S	x	4	6
	y	4	6
	z	1	6
W	x	4	16
	y	4	16
	z	1	8
A	x	4	32
	y	4	32
	z	1	16
B	x	4	76
	y	4	52
	z	1	17
C	x	4	120
	y	4	80
	z	1	28
D	x	4	408
	y	4	304
	z	1	34

Table 2: Zone sizes (in points) for LU-MZ

Class	dir	# zones	size
S	x	2	12
	y	2	12
	z	1	6
W	x	4	16
	y	4	16
	z	1	8
A	x	4	32
	y	4	32
	z	1	16
B	x	8	38
	y	8	26
	z	1	17
C	x	16	30
	y	16	20
	z	1	28
D	x	32	51
	y	32	38
	z	1	34

Table 3: Zone sizes (in points) for SP-MZ

3.2 SP-MZ

As in the case of LU-MZ, the overall mesh is partitioned such that the zones are identical in size. However, the number of zones in each of the two horizontal dimensions grows as the problem size grows. The actual sizes of the zones are listed in Table 3.

3.3 BT-MZ

The number of zones in this benchmark grows with the problem size in the same fashion as in SP-MZ. However, the overall mesh is now partitioned such that the sizes of the zones span a significant range. This is accomplished by increasing sizes of successize zones in a particular coordinate direction in a roughly geometric fashion. An example of the stretched tiling of the overall mesh is shown in Figure 3. Except for class S, the ratio of largest over smallest total zone size is approximately 20. This makes it harder to balance the load than for SP-MZ and LU-MZ if the implementation is to take advantage of multi-level parallelism. The actual sizes of the successive zones for all problem classes are listed in Table 4.

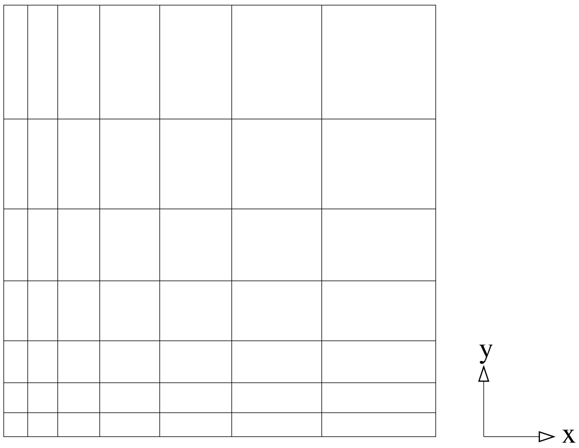

Figure 3: Example of uneven mesh tiling (horizontal cut through mesh system) for the BT-MZ benchmark.

4 Performance Reporting

The paper-and-pencil specification of the original NPB defined a performance result of the individual application benchmarks as the elapse time of a certain segment of the whole code. A quantity that can be derived from that elapse time is the number of millions of floating point operations per second (MF/s). This is necessarily an approximation, since the actual number of floating point instructions executed depends on compiler technology and the presence of specialized hardware (floating point

Class	dir	# zones	sizes
S	x	2	6, 18
	y	2	6, 18
	z	1	6
W	x	4	6, 11, 18, 29
	y	4	6, 11, 18, 29
	z	1	8
A	x	4	13, 21, 36, 58
	y	4	13, 21, 36, 58
	z	1	16
B	x	8	16, 20, 24, 30, 38, 47, 57, 72
	y	8	11, 13, 17, 21, 26, 31, 40, 49
	z	1	17
C	x	16	13, 14, 15, 18, 19, 21, 23, 26, 28, 31, 35, 38, 43, 47, 52, 57
	y	16	8, 10, 10, 12, 12, 14, 16, 17, 19, 21, 23, 26 28, 31, 35, 38
	z	1	28
D	x	32	22, 23, 24, 25, 26, 28, 29, 31, 32, 34, 35, 37, 39, 41, 43, 45, 48, 49, 52, 55, 58, 60, 63, 67, 70, 73, 77, 81, 85, 88, 94, 98
	y	32	16, 17, 18, 19, 20, 20, 22, 23, 24, 25, 26, 28, 29, 30, 33, 33, 35, 37, 39, 41, 43, 45, 47, 50, 52, 54, 58, 60, 63, 66, 70, 73
	z	1	34

Table 4: Zone sizes (in points) for BT-MZ

multiply/add units, division and square root hardware). In NPB 2.3 [2] curve fits were used that best fit data gathered by hardware performance counters on a number of systems available at that time. We use the same curve fits for computing MF/s for each zone of the new benchmark suite. While they are not very accurate since they were derived using meshes with different sizes and aspect ratios, they provide a convenient means of determining the relative performance of different systems. Let n_x, n_y, and n_z be the extents of a zone in the three respective coordinate directions, n_i is the number of timed solver iterations, and T is the elapse time of the measured code segments in the application benchmarks. The following formulas define our performance curve fits.

$$\mathrm{MF/s}_{LU} = \left[1984.8 n_x n_y n_z - 1213.7(n_x + n_y + n_z)^2 + 9257.0(n_x + n_y + n_z) - 144010\right] \frac{n_i}{10^6 T}$$
$$\mathrm{MF/s}_{SP} = \left[881.17 n_x n_y n_z - 520.43(n_x + n_y + n_z)^2 + 3828.2(n_x + n_y + n_z) - 19272\right] \frac{n_i}{10^6 T}$$
$$\mathrm{MF/s}_{BT} = \left[3478.8 n_x n_y n_z - 1961.7(n_x + n_y + n_z)^2 + 9341.2(n_x + n_y + n_z)\right] \frac{n_i}{10^6 T}$$

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] M.J. Djomehri, R. Biswas, M. Potsdam, R.C. Strawn. *An Analysis of Performance Enhancement Techniques for Overset Grid Applications.* NAS Technical Report NAS-03-008, NASA Ames Research Center, Moffett Field, CA, 2003

[4] M. Frumkin, M. Schultz, H. Jin, J. Yan. *Implementation of the NAS Parallel Benchmarks in Java* NAS Technical Report NAS-02-009, NASA Ames Research Center, Moffett Field, CA, 2002.

[5] M. Frumkin, H. Jin, J. Yan. *Implementation of NAS Parallel Benchmarks in High Performance Fortran.* NAS Technical Report NAS-98-009, NASA Ames Research Center, Moffett Field, CA, 1998.

[6] H. Jin, M. Frumkin, J. Yan. *The OpenMP Implementation of NAS Parallel Benchmarks and its Performance.* NAS Technical Report NAS-99-011, NASA Ames Research Center, Moffett Field, CA, 1999.

[7] R.F. Van Der Wijngaart. *NAS Parallel Benchmarks, Version 2.4.* NAS Technical Report NAS-02-007, NASA Ames Research Center, Moffett Field, CA, 2002

Appendix: Verification

The solution on each zone for each of the modified NPB problems is initialized in the same way as the single-zone NPB. Verification values are computed individually for each zone in the same way as for the original NPB, and are then summed over all zones, taking into account that L^2 norms of solution errors and residuals are scaled using the actual number of points in each zone, not the overall number of points. The same error tolerances for verification apply as for the single-zone NPB.

Class	m	Residual norm	Error norm	surface integral
S	1	$0.3778579699366 * 10^1$	$0.2429480066305 * 10^2$	$0.4964435445706 * 10^2$
	2	$0.3120418698065 * 10^0$	$0.9072817470024 * 10^1$	
	3	$0.8386213407018 * 10^0$	$0.1032621825644 * 10^2$	
	4	$0.4452165980488 * 10^0$	$0.9256791727838 * 10^1$	
	5	$0.7808656756434 * 10^1$	$0.1639045777714 * 10^2$	
W	1	$0.8285060230339 * 10^3$	$0.7514670702651 * 10^2$	$0.3781055348911 * 10^3$
	2	$0.5753415004693 * 10^2$	$0.9776687033238 * 10^1$	
	3	$0.2023477570531 * 10^3$	$0.2141754291209 * 10^2$	
	4	$0.1586275182502 * 10^3$	$0.1685405918675 * 10^2$	
	5	$0.1733925947816 * 10^4$	$0.1856944519722 * 10^3$	
A	1	$0.1131574877175 * 10^4$	$0.1115694885382 * 10^3$	$0.5904992211511 * 10^3$
	2	$0.7965206944742 * 10^2$	$0.1089257673798 * 10^2$	
	3	$0.2705587159526 * 10^3$	$0.2905379922066 * 10^2$	
	4	$0.2129567530746 * 10^3$	$0.2216126755530 * 10^2$	
	5	$0.2260584655432 * 10^4$	$0.2501762341026 * 10^3$	
B	1	$0.1734656959567 * 10^5$	$0.1781612313296 * 10^4$	$0.6107041476456 * 10^4$
	2	$0.1238977748533 * 10^4$	$0.1177971120769 * 10^3$	
	3	$0.4123885357100 * 10^4$	$0.4233792871440 * 10^3$	
	4	$0.3613705834056 * 10^4$	$0.3577260438230 * 10^3$	
	5	$0.3531187871586 * 10^5$	$0.3659958544012 * 10^4$	
C	1	$0.4108743427233 * 10^5$	$0.3429276307955 * 10^4$	$0.1125826349653 * 10^5$
	2	$0.3439004802235 * 10^4$	$0.2336680861825 * 10^3$	
	3	$0.9961331392486 * 10^4$	$0.8216363109621 * 10^3$	
	4	$0.8321426758084 * 10^4$	$0.7143809828225 * 10^3$	
	5	$0.7463792419218 * 10^5$	$0.7057470798773 * 10^4$	
D	1	$0.3282253166388 * 10^6$	$0.6620775619126 * 10^4$	$0.2059421629621 * 10^5$
	2	$0.3490781637713 * 10^5$	$0.5229798207352 * 10^3$	
	3	$0.8610311978292 * 10^5$	$0.1620218261697 * 10^4$	
	4	$0.7004896022603 * 10^5$	$0.1404783445006 * 10^4$	
	5	$0.4546838584391 * 10^6$	$0.1222629805121 * 10^5$	

Table 5: Verification values for LU-MZ. m signifies vector component.

Class	m	Residual norm	Error norm
S	1	$0.7698876173566 * 10^1$	$0.9566808043467 * 10^1$
	2	$0.1517766790280 * 10^1$	$0.3894109553741 * 10^1$
	3	$0.2686805141546 * 10^1$	$0.4516022447464 * 10^1$
	4	$0.1893688083690 * 10^1$	$0.4099103995615 * 10^1$
	5	$0.1369739859738 * 10^2$	$0.7776038881521 * 10^1$
W	1	$0.1887636218359 * 10^3$	$0.2975895149929 * 10^2$
	2	$0.1489637963542 * 10^2$	$0.1341508175806 * 10^2$
	3	$0.4851711701400 * 10^2$	$0.1585310846491 * 10^2$
	4	$0.3384633608154 * 10^2$	$0.1450916426713 * 10^2$
	5	$0.4036632495857 * 10^3$	$0.5854137431023 * 10^2$
A	1	$0.2800097900548 * 10^3$	$0.3112046666578 * 10^2$
	2	$0.2268349014438 * 10^2$	$0.1172197785348 * 10^2$
	3	$0.7000852739901 * 10^2$	$0.1486616708032 * 10^2$
	4	$0.5000771004061 * 10^2$	$0.1313680576292 * 10^2$
	5	$0.5552068537578 * 10^3$	$0.7365834058154 * 10^2$
B	1	$0.5190422977921 * 10^4$	$0.5469182054223 * 10^3$
	2	$0.3655458539065 * 10^3$	$0.4983658028989 * 10^2$
	3	$0.1261126592633 * 10^4$	$0.1418301776602 * 10^3$
	4	$0.1002038338842 * 10^4$	$0.1097717156175 * 10^3$
	5	$0.1075902511165 * 10^5$	$0.1260195162174 * 10^4$
C	1	$0.5886814493676 * 10^5$	$0.6414069213021 * 10^4$
	2	$0.3967324375474 * 10^4$	$0.4069468353404 * 10^3$
	3	$0.1444126529019 * 10^5$	$0.1585311908719 * 10^4$
	4	$0.1210582211196 * 10^5$	$0.1270243185759 * 10^4$
	5	$0.1278941567976 * 10^6$	$0.1441398372869 * 10^5$
D	1	$0.7650595424723 * 10^6$	$0.8169589578340 * 10^5$
	2	$0.5111519817683 * 10^5$	$0.5252150843148 * 10^4$
	3	$0.1857213937602 * 10^6$	$0.1984739188642 * 10^5$
	4	$0.1624096784059 * 10^6$	$0.1662852404547 * 10^5$
	5	$0.1642416844328 * 10^7$	$0.1761381855235 * 10^6$

Table 6: Verification values for SP-MZ. m signifies vector component.

Class	m	Residual norm	Error norm
S	1	$0.1047687395830 * 10^4$	$0.1775416062982 * 10^3$
	2	$0.9419911314792 * 10^2$	$0.1875540250835 * 10^2$
	3	$0.2124737403068 * 10^3$	$0.3863334844506 * 10^2$
	4	$0.1422173591794 * 10^3$	$0.2634713890362 * 10^2$
	5	$0.1135441572375 * 10^4$	$0.1965566269675 * 10^3$
W	1	$0.5562611195402 * 10^5$	$0.7185154786403 * 10^4$
	2	$0.5151404119932 * 10^4$	$0.7040472738068 * 10^3$
	3	$0.1080453907954 * 10^5$	$0.1437035074443 * 10^4$
	4	$0.6576058591929 * 10^4$	$0.8570666307849 * 10^3$
	5	$0.4528609293561 * 10^5$	$0.5991235147368 * 10^4$
A	1	$0.5536703889522 * 10^5$	$0.6716797714343 * 10^4$
	2	$0.5077835038405 * 10^4$	$0.6512687902160 * 10^3$
	3	$0.1067391361067 * 10^5$	$0.1332930740128 * 10^4$
	4	$0.6441179694972 * 10^4$	$0.7848302089180 * 10^3$
	5	$0.4371926324069 * 10^5$	$0.5429053878818 * 10^4$
B	1	$0.4461388343844 * 10^6$	$0.4496733567600 * 10^5$
	2	$0.3799759138035 * 10^5$	$0.3892068540524 * 10^4$
	3	$0.8383296623970 * 10^5$	$0.8763825844217 * 10^4$
	4	$0.5301970201273 * 10^5$	$0.5599040091792 * 10^4$
	5	$0.3618106851311 * 10^6$	$0.4082652045598 * 10^5$
C	1	$0.3457703287806 * 10^7$	$0.2059106993570 * 10^6$
	2	$0.3213621375929 * 10^6$	$0.1680761129461 * 10^5$
	3	$0.7002579656870 * 10^6$	$0.4080731640795 * 10^5$
	4	$0.4517459627471 * 10^6$	$0.2836541076778 * 10^5$
	5	$0.2818715870791 * 10^7$	$0.2136807610771 * 10^6$
D	1	$0.4250417034981 * 10^8$	$0.9462418484583 * 10^6$
	2	$0.4293882192175 * 10^7$	$0.7884728947105 * 10^5$
	3	$0.9121841878270 * 10^7$	$0.1902874461259 * 10^6$
	4	$0.6201357771439 * 10^7$	$0.1361858029909 * 10^6$
	5	$0.3474801891304 * 10^8$	$0.9816489456253 * 10^6$

Table 7: Verification values for BT-MZ. m signifies vector component.