RC 20046 (03/01/95)
Computer Science/Mathematics

IBM Research Report

Performance of NAS Parallel Benchmark LU on IBM SP Systems

Vijay K. Naik

IBM Research Division
T.J. Watson Research Center
Yorktown Heights, New York

IBM Research Division
Almaden · T.J. Watson · Tokyo · Zurich

Performance of NAS Parallel Benchmark LU on IBM SP Systems*

Vijay K. Naik

IBM T. J. Watson Research Center, P. O. Box 218,
Yorktown Heights, NY 10598, USA. vkn@watson.ibm.com

Abstract

Abstract: In this paper, we describe and compare performance of the NAS Parallel Benchmark LU on five different parallel systems spanning two generations of the IBM Scalable POWERparallel architectures. The family of SP systems we consider are: the SP1-TB0, SP1-TB2, SP2-TN, SP2-TN2, and SP2-WN. We describe some of the highlights of our parallel implementation for distributed memory systems. We also discuss some of the optimization steps we used to realize good performance on individual processors and to reduce communication overheads.

1 INTRODUCTION

In 1991, the Numerical Aerodynamic Simulation (NAS) program at NASA Ames Research Center announced a set of applications and kernels for benchmarking highly parallel supercomputers [1]. In the recent past, performance results of these benchmarks characterizing various parallel platforms have been reported by many vendors as well as by researchers at NASA and elsewhere [2, 3]. The depth of these benchmarks and the fact that these capture the essence of typical large scale Computational Fluid Dynamics (CFD) applications, have made these benchmarks popular not only for the purpose of evaluating parallel supercomputing systems, but also in demonstrating the viability of novel software and architectural concepts. As a results, the NAS Parallel Benchmark suite has become one of the most widely used benchmarks in many areas of high performance parallel computing.

The NAS benchmark suite consists of eight problems: five kernels and three simulated CFD applications [1]. The kernels are relatively simple solvers and are meant to provide insight into the performance of particular types of numerical computations. The three simulated CFD

*To appear in the Proceedings of Parallel CFD'94, Elsevier Science, 1995

applications are more complex and their computational requirements closely resemble those in the state-of-the-art CFD application codes [1]. However, absent from the application benchmarks are complex boundary conditions and I/O that are typically present in many real CFD applications. For that reason, the three application benchmarks are also referred to as pseudo applications. The three application benchmarks are: the Scalar Pentadiagonal (SP), Block Tridiagonal (BT), and Lower-Upper Diagonal (LU) benchmarks. In [4] and [5], we have presented performance and implementation of the SP and BT benchmarks on distributed memory systems.

In this paper, we present the performance of LU benchmark on the IBM Scalable POWERparallel 1 (SP1) and IBM Scalable POWERparallel 2 (SP2) architectures. As far as we know, our implementation of the LU benchmark, as described in this paper, has delivered the best performance on the SP1 and SP2 systems. Moreover, as of the writing of this paper, the performance of our implementation on SP2 (with wide nodes) with more than 16 processors is significantly better than the performance of any other implementation on any other parallel system with the same number of processors [2, 3]. One objective of this paper is to present some of the salient points of our implementation so other studies, including nonbenchmarking type, may benefit from this work.

In the next section, we describe the mathematical and the numerical problem solved by the LU benchmark. We outline the modifications we made to the original implementation in Section 3. In Section 4, we briefly describe the IBM SP systems used in this study. The performance results are presented in Sections 5 and Section 6 concludes the paper.

2 BENCHMARK LU

The LU benchmark represents the computations associated with a newer class of implicit CFD algorithms that use the symmetric, successive overrelaxation (SSOR) method to compute the implicit operators. An example of an application that is based on this newer class of algorithms is INS3D-LU [6]. In this section, we describe some of the salient points of this benchmark. Most of the discussion in this section is based on Chapter 3 of [1].

The problem solved in benchmark LU is that of computing the numerical solution to a synthetic system of five nonlinear partial differential equations (PDE's) that represent some of the key characteristics exhibited by the Navier-Stokes equations. An implicit type of numerical solution is used in solving this system of PDE's. This method is used as a solver in many of the computational fluid dynamics (CFD) programs designed for the numerical solution of 3-dimensional Euler/Navier-Stokes equations using finite-volume or finite-difference discretization on structured grids. The system of five nonlinear PDE's is as follows:

$$\begin{aligned}\frac{\partial \mathbf{U}}{\partial \tau} &= \frac{\partial \mathbf{E}(\mathbf{U})}{\partial \xi} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial \eta} + \frac{\partial \mathbf{G}(\mathbf{U})}{\partial \zeta} + \frac{\partial \mathbf{T}(\mathbf{U}, \mathbf{U}_\xi)}{\partial \xi} \\ &+ \frac{\partial \mathbf{V}(\mathbf{U}, \mathbf{U}_\eta)}{\partial \eta} + \frac{\partial \mathbf{W}(\mathbf{U}, \mathbf{U}_\zeta)}{\partial \zeta} + \mathbf{H}(\mathbf{U}, \mathbf{U}_\xi, \mathbf{U}_\eta, \mathbf{U}_\zeta), \qquad (1)\end{aligned}$$

where $\mathbf{U} = \left[u^{(1)},\ u^{(2)},\ u^{(3)},\ u^{(4)},\ u^{(5)}\right]^T$ is a vector function of temporal variable τ and spatial variables (ξ, η, ζ) that form the orthogonal coordinate system in $\Re^3$. **E**, **F**, **G**, **T**, **V**, **W**, and **H** are 5×1 vector functions whose components are prescribed as functions of **U** or **U** and its derivatives. The boundary conditions are of uncoupled Dirichlet type and the initial values of **U** at $\tau = 0$ are obtained by a transfinite, tri-linear interpolation of the boundary data [1]. The forcing function vector **H** is chosen such that, with the specified boundary and initial conditions, the system of PDE's given by Eq (1) satisfies an exact solution (a fourth order polynomial in ξ, η, and ζ) to **U**. The computations of the LU benchmark seek to obtain a discrete approximation to the steady-state solution of the PDE's, using a pseudo-time marching scheme (two-level, first-order accurate, Euler implicit) and a spatial discretizing procedure based on finite difference approximations (second-order accurate central difference approximations in each of the three coordinate directions). A linear fourth-difference dissipation term is added to the right hand side of Eq (1) so that the numerical scheme converges to a steady-state solution. If $\mathbf{U}^n$ and $\mathbf{U}^{n+1}$ are solutions at time-step n and $n+1$, respectively, and $\Delta\mathbf{U}^n$ is $\mathbf{U}^{n+1} - \mathbf{U}^n$, then the numerical procedure involves solution of a linear system of equations for $\Delta\mathbf{U}^n$ to determine $\mathbf{U}^{n+1}$. The linear system of equations has the following form:

$$(\mathbf{I} + \Delta\tau\mathbf{L})\,\Delta\mathbf{U}^n = \Delta\tau\mathbf{R} \tag{2}$$

where $[\mathbf{L}]$ consists of the flux Jacobian terms corresponding to the vectors **E**, **F**, **G**, **T**, **V**, **W** and the implicitly treated dissipation terms. **R** consists of the spatial difference terms for the vectors **E**, **F**, **G**, **T**, **V**, **W**, the forcing function vector **H**, and the added dissipation terms. In the above equation, the left hand side (LHS) is the implicit part and the right hand side (RHS) is the explicit part.

In LU benchmark, the linear system represented by Eq (2) is solved using the symmetric, successive over-relaxation (SSOR) scheme. Under this scheme, the linear system of equations can be written as:

$$[\mathbf{K}^n]\,[\Delta\mathbf{U}^n] = [\mathbf{RHS}^n] \tag{3}$$

where, $\mathbf{K}^n = \mathbf{I} - \Delta\tau\left[D_\xi\mathbf{A}^n + D_\xi^2\mathbf{N}^n + D_\eta\mathbf{B}^n + D_\eta^2\mathbf{Q}^n + D_\zeta\mathbf{C}^n + D_\zeta^2\mathbf{S}^n\right]$ and $\mathbf{A}^n$, $\mathbf{B}^n$, $\mathbf{C}^n$, $\mathbf{N}^n$, $\mathbf{Q}^n$, and $\mathbf{S}^n$ are Jacobian matrices at step n. The resulting **K** matrix has a (5×5) block septa-diagonal structure. The point-SSOR iteration scheme is applied to solve Eq (3) as: $\omega(2-\omega)(\mathbf{D}^n + \omega\mathbf{Y}^n)(\mathbf{I} + \omega(\mathbf{D}^n)^{-1}\mathbf{Z}^n)\,[\Delta\mathbf{U}^n] = [\mathbf{RHS}^n]$, where $\mathbf{D}^n$ is main block-diagonal of $\mathbf{K}^n$, $\mathbf{Y}^n$ is strictly lower triangular (3 sub-block-diagonals of $\mathbf{K}^n$), $\mathbf{Z}^n$ is strictly upper triangular (3 super-block-diagonals of $\mathbf{K}^n$), and ω is the over-relaxation factor.

A functional description of the LU benchmark is shown in Figure 1. In that figure, the SSOR algorithm is applied in Steps *i* and *ii*. In Step *iv*, **RHS** is computed. These three steps constitute the computationally intensive sections of the LU benchmark. In both Steps *i* and *ii*, coefficient matrices are computed first. $([\mathcal{A}_l], [\mathcal{B}_l], \ldots, [\mathcal{G}_l]$ represent the coefficients in the l-th block row,

Set `Boundary values of` $\mathbf{U}_{i,j,k}$ `for` $(i,j,k) \in \partial D_h$
Set `Initial values of` $\mathbf{U}^0_{i,j,k}$ `for` $(i,j,k) \in D_h$
Compute `forcing function vector,` $\mathbf{H}^*_{i,j,k}$ `for` $(i,j,k) \in D_h$
Initialize $[\mathbf{RHS}]_{i,j,k}$ `at` $\tau = 0$ `for` $(i,j,k) \in D_h$

for $\tau = 1$ **to** `Maxsteps` **do**

i. **Perform** block lower-triangular solution:

i(a). **Setup** `lower-triangular coefficient matrix:`
Compute from `Jacobian matrices:`

$$[\mathcal{D}_l] = \mathbf{I} + \Delta\tau(2/h_\xi^2)\left[\mathbf{N}(\mathbf{U}^{\tau-1}_{i,j,k})\right] + \Delta\tau(2/h_\eta^2)\left[\mathbf{Q}(\mathbf{U}^{\tau-1}_{i,j,k})\right] + \Delta\tau(2/h_\zeta^2)\left[\mathbf{S}(\mathbf{U}^{\tau-1}_{i,j,k})\right],$$

$$[\mathcal{A}_l] = -\Delta\tau(-1/2h_\zeta)\left[\mathbf{C}(\mathbf{U}^{\tau-1}_{i,j,k-1})\right] - \Delta\tau(1/h_\zeta^2)\left[\mathbf{S}(\mathbf{U}^{\tau-1}_{i,j,k-1})\right],$$

$$[\mathcal{B}_l] = -\Delta\tau(-1/2h_\eta)\left[\mathbf{B}(\mathbf{U}^{\tau-1}_{i,j-1,k})\right] - \Delta\tau(1/h_\eta^2)\left[\mathbf{Q}(\mathbf{U}^{\tau-1}_{i,j-1,k})\right],$$

$$[\mathcal{C}_l] = -\Delta\tau(-1/2h_\xi)\left[\mathbf{A}(\mathbf{U}^{\tau-1}_{i-1,j,k})\right] - \Delta\tau(1/h_\xi^2)\left[\mathbf{N}(\mathbf{U}^{\tau-1}_{i-1,j,k})\right],$$

i(b). **Solve** `block lower triangular system for` $[\Delta\mathbf{U}_1]_{i,j,k}$ `for` $(i,j,k) \in D_h$:

$$(\mathbf{D}^{\tau-1} + \omega\mathbf{Y}^{\tau-1})\,\Delta\mathbf{U}_1 = [\mathbf{RHS}].$$

ii. **Perform** block upper-triangular solution:

ii(a). **Setup** `upper-triangular coefficient matrix:`
Compute from `Jacobian matrices:`

$$[\mathcal{E}_l] = -\Delta\tau(1/2h_\xi)\left[\mathbf{A}(\mathbf{U}^{\tau-1}_{i+1,j,k})\right] - \Delta\tau(1/h_\xi^2)\left[\mathbf{N}(\mathbf{U}^{\tau-1}_{i+1,j,k})\right],$$

$$[\mathcal{F}_l] = -\Delta\tau(1/2h_\eta)\left[\mathbf{B}(\mathbf{U}^{\tau-1}_{i,j+1,k})\right] - \Delta\tau(1/h_\eta^2)\left[\mathbf{Q}(\mathbf{U}^{\tau-1}_{i,j+1,k})\right],$$

$$[\mathcal{G}_l] = -\Delta\tau(1/2h_\zeta)\left[\mathbf{C}(\mathbf{U}^{\tau-1}_{i,j,k+1})\right] - \Delta\tau(1/h_\zeta^2)\left[\mathbf{S}(\mathbf{U}^{\tau-1}_{i,j,k+1})\right],$$

ii(b). **Solve** `block upper triangular system for` $[\Delta\mathbf{U}^{\tau-1}]_{i,j,k}$ `for` $(i,j,k) \in D_h$:

$$(\mathbf{I} + \omega[\mathbf{D}^{\tau-1}]^{-1}\mathbf{Z}^{\tau-1})\,\Delta\mathbf{U}^{\tau-1} = \Delta\mathbf{U}_1.$$

iii. **Update** `solution to time-step` τ:

$$\mathbf{U}^\tau = \mathbf{U}^{\tau-1} + \Delta\mathbf{U}^{\tau-1}$$

iv. **Compute** $[\mathbf{RHS}]^\tau_{i,j,k}$ `for` $(i,j,k) \in D_h$
end for

FIGURE 1
A functional outline of the LU benchmark.

where each block is a 5×5 matrix.) The lower and upper triangular solves sweep through the entire grid and the diagonal block inversion is performed using Gaussian elimination. For the explicit computations of Step *iv*, a 13-point spatial discretization is used and the resulting computations are equivalent to performing sparse block (5×5) matrix-vector multiplication.

3 IMPLEMENTATION OF BENCHMARK LU

The starting point for our parallel implementation was the sequential Fortran 77 implementation of the LU benchmark (written by S. Weeratunga) provided by NASA Ames as an example implementation. For our parallel implementation, we modified this code significantly to extract parallelism and to get good floating-point performance. In this section, we present some of the key modifications we made in our implementation.

In each iteration of the original implementation, 5×5 coefficient blocks $[\mathcal{A}_{i,j,k}], \ldots, [\mathcal{G}_{i,j,k}]$ are computed separately at each grid point. $[\mathcal{A}_{i,j,k}]$ is computed from $\mathbf{U}_{i,j,k-1}$, $[\mathcal{B}_{i,j,k}]$ is computed from $\mathbf{U}_{i,j-1,k}$ and so on. Following the setup, in Step *i* of each iteration, a sparse block lower triangular system is solved as:

$$[\mathbf{V}_{i,j,k}] \leftarrow [\mathcal{D}_{i,j,k}]^{-1} \left[[\mathbf{RHS}]_{i,j,k} - \omega \left([\mathcal{A}_{i,j,k}][\mathbf{V}_{i,j,k-1}] + [\mathcal{B}_{i,j,k}][\mathbf{V}_{i,j-1,k}] + [\mathcal{C}_{i,j,k}][\mathbf{V}_{i-1,j,k}] \right) \right]$$

and, in Step *ii*, a sparse block upper triangular system is solved as:

$$\Delta\mathbf{U}_{i,j,k} \leftarrow [\mathbf{V}_{i,j,k}] - \omega [\mathcal{D}_{i,j,k}]^{-1} \left[[\mathcal{E}_{i,j,k}]\Delta\mathbf{U}_{i+1,j,k} + [\mathcal{F}_{i,j,k}]\Delta\mathbf{U}_{i,j+1,k} + [\mathcal{G}_{i,j,k}]\Delta\mathbf{U}_{i,j,k+1} \right].$$

In our implementation, we merged the coefficient computations with the lower triangular solve step and did not compute any coefficients before the upper triangular solve. Specifically, in the lower triangular solve step, while computing the unknowns $[\mathbf{V}_{i,j,k}]$ on the k-th plane, $[\mathcal{A}]$ values for the $(k+1)$-th plane are computed and while computing the unknowns $[\mathbf{V}_{i,j,k}]$ on the j-th line, $[\mathcal{B}]$ values for the $(j+1)$-th line are computed. After computing $[\mathbf{V}_{i,j,k}]$, $[\mathcal{C}]$ is computed for the $(i+1)$-th grid point and $[\mathcal{D}]$ is computed just prior to computing $[\mathbf{V}_{i,j,k}]$. While these coefficients are computed, we also set the block coefficients in the upper triangular matrix. By doing so, we avoid some of the duplication of computations such as those in setting up $[\mathcal{E}_{i-1,j,k}]$ and $[\mathcal{C}_{i+1,j,k}]$, $[\mathcal{F}_{i,j-1,k}]$ and $[\mathcal{B}_{i,j+1,k}]$, and those in computing $[\mathcal{G}_{i,j,k-1}]$ and $[\mathcal{A}_{i,j,k+1}]$. Note that with these modifications, $[\mathbf{U}_{i,j,k}]$ loaded only once.

We also note that there are a significant number of zeros in the 5×5 coefficient blocks used in the coefficient computations. (Problem specification introduces these zeros in the Jacobians.) For example, $[\mathcal{D}]$ has 12 nonzeros and is lower triangular and all other coefficient blocks have 18 nonzeros. We take advantage of the sparsity structure of the coefficient matrices to reduce operation count and memory requirements. The lower triangular structure of $[\mathcal{D}]$ further allows us to reduce the operation count of computing $[\mathcal{D}]^{-1}[\mathbf{V'}_{i,j,k}]$. Thus, in the original implementation, the operation count of this inversion is 50 daxpy's, 9 divisions, and 10 multiplies (in each solve). We reduced this to 10 daxpy's and 5 divisions in the lower triangular solve, and to 10 daxpy's and 5 multiplies in the upper triangular solve. The sparsity structure also allowed us to reduce the memory requirements for the coefficients to 66 variables per grid point from the original 100 variables per grid point.

For the parallel implementation of the LU benchmark, we use SPMD style of programming and take advantage of the available data parallelism. We note that the SSOR part of the LU benchmark has only a limited degree of parallelism. In all our implementations of LU benchmark, for which we have reported performance, we used a 2-D uni-partitioning method for data distribution [7]. In our parallel implementation, in each sweep, we first compute triangular solves at the grid points along the ξ direction. Following this, the next grid-line (η direction) is computed. Clearly, to extract the available parallelism, a message must be sent whenever a grid-point on the partition boundary is computed. For LU, 5 words are sent or received for each grid point on the partition boundary. When message latency cost is high, such frequent communication leads to high communication overhead. We hide latency by sending one message per several line solves in the ξ direction. In the η direction, one message is sent for each line segment on the partition boundary. Note that for the partitioning scheme we chose, there is no communication along the ζ direction.

4 IBM SP ARCHITECTURE

For this study, performance measurements were made on five different parallel platforms belonging to the IBM SP1 and SP2 architectures. The SP1 and SP2 architectures differ primarily in the individual processor architecture. The processors of SP1 belong to the RS/6000 POWER architecture and those of SP2 belong to RS/6000 POWER2 architecture. We considered two types of SP1 systems and three types of SP2 systems. The two SP1 systems differed in their communication adapter where as the three SP2 systems differed in the type of POWER2 processors used. Readers interested in the details of various aspects of the POWER and POWER2 architectures should refer to the articles that appear in [8] and [9], respectively. Throughout this paper we refer to the two SP1 platforms as SP1-TB0 and SP1-TB2. (TB0 and TB2 are the communication adapters used in these systems.) We point out here that, in the literature, SP1-TB2 is sometimes referred to as SP2 thin nodes/POWER. In this paper, the three SP2 systems are referred to as SP2-TN, SP2-TN2, and SP2-WN. (TN stands for “thin nodes”, TN2 for “thin nodes 2”, and WN for “wide nodes”.) Table 1 summarizes the characteristics of the individual processors of these five systems. Since the individual processors of SP1-TB0 and SP1-TB2 are identical, we have grouped these two systems together in that table. The SP2-TN2 nodes can be fitted with L2-cache. However, the SP2-TN2 system for which we report performance in this paper, had no L2 cache. We further note that, for this study, we used an SP2-WN system with 256KB data cache on each node with 256-bit wide memory bus. On all systems, except on SP2-TN, each processor had 128 MB of memory. On the

TABLE 1
A summary of the single processor characteristics for the five systems considered in this study

	SP1-TB0/SP1-TB2	SP2-TN	SP2-TN2	SP2-WN
RS/6000	Model 370	Model 390	Model 390H	Model 590
Processor	(POWER)	(POWER2)	(POWER2)	(POWER2)
Clock	62.5 MHz	66.5 MHz	66.5 MHz	66.5 MHz
Data cache	32 KB	64 KB	128 KB	256/128 KB
Instr cache	32 KB	32 KB	32 KB	32 KB
Bus Width	64-Bit	64-Bit	128-Bit	256/128-Bit
Memory	64-256 MB	64-512 MB	64-512 MB	64MB-2GB
Instr (Ops)/cycle	4 (5)	6 (8)	6 (8)	6 (8)
# FPUs	1	2	2	2
# FPRs	32	32	32	32
Peak MFLOPS	125	266	266	266

SP2-TN, each node had only 64 MB of memory. Although the peak performance of the processors of the three SP2 systems is identical, the SP2-WN system tends to deliver higher performance because of other processor features such as cache size and memory bus width. This will be obvious from the results presented in the next section.

All the five SP systems discussed here use identical interconnection network which is based on the IBM's High Performance Switch. The network used is a variation of the omega-type multistage network with the number of stages required to connect n processors being proportional to $\log_4 n$. A buffered wormhole routing strategy is used to move packets from source to destination across the network. All systems, except the SP1-TB0, use TB2 communication adapter card, which is superior in performance to TB0. Table 2 summarizes the key communication characteristics of the five systems. For this study, we found the communication characteristics of SP2-TN2 and SP2-WN systems to be identical. For that reason, those two systems are grouped together in Table 2. The communication protocols used on SP1 systems are different than those on the SP2 systems. However, in all cases, the user interface for application programs was exactly the same. In all our implementations, we used IBM's proprietary External User Interface (EUI) to write SPMD parallel programs with message passing calls. Finally, note that the point-to-point and the exchange bandwidths listed in Table 2 are asymptotic in the sense that these figures represent performance with message length greater than 8KB. Further details on various aspects of the system architectures can be found in articles that appear in [10].

TABLE 2
A summary of communication characteristics for the five systems considered in this study

	SP1-TB0	SP1-TB2	SP2-TN	SP2-TN2/SP2-WN
Adapter card	TB0	TB2	TB2	TB2
Comm Protocol	EUIH	EUIH2	POE	POE
Latency	28 μsec	49 μsec	41 μsec	39 μsec
Pt-to-pt Bandwidth	8.7 MB/sec	31 MB/sec	35 MB/sec	35 MB/sec
Exchange Bandwidth	8.7 MB/sec	44 MB/sec	42 MB/sec	48 MB/sec

5 PERFORMANCE RESULTS

In this section, we summarize the performance results for the LU benchmark on the family of SP platforms described in the previous section. The benchmarking rules stipulate two grid sizes – $64 \times 64 \times 64$ and $102 \times 102 \times 102$. These are referred to as the Class A and Class B problem sets, respectively. In both cases, for the LU benchmark, 250 iterations are performed. Here we report performance results for both Class A and Class B problem sets. SP1-TB0 and SP1-TB2 measurements are from the SP1 system at the IBM T. J. Watson Research Center. The SP2-TN measurements are from the SP2 system at the Maui High Performance Computing Center and the SP2-TN2 measurements were made at the IBM Kingston Laboratory. The SP2-WN measurements are from the SP2 system at NASA Ames Research Center.

The performance results for the Class A problem set are shown in Table 3 and those for the Class B problem set are shown in Table 4. In both cases, the performance is reported on up to 128 processors. In case of Class A, the performance on each system is compared with that on a single processor Cray Y-MP and, in case of Class B, the performance is compared with that on a single processor Cray C-90. The Y-MP and C-90 performance figures are from [3].

In Tables 3 and 4, we also report the MFLOPS (Millions of FLoating-point Operations Per Second) delivered by the IBM SP systems on the two problem sets. For determining the MFLOPS, we have

TABLE 3

Performance Comparisons for Benchmark LU (Class A problem set)

System	No. of Procs.	Time (sec.) for 250 iters.	Ratio to Y-MP/1	Equiv Y-MP/1 MFLOPS
Cray Y-MP	1	333.5	1.00	194.0
	8	49.5	6.74	1,307.6
Cray C-90	1	157.6	2.12	411.3
	16	17.6	18.93	3,672.4
IBM SP1-TB0	8	291.4	1.14	222.0
	16	172.9	1.93	374.2
	32	101.8	3.28	635.6
	64	63.2	5.28	1,023.7
	128	42.5	7.85	1,522.3
IBM SP1-TB2	16	155.5	2.14	416.1
	32	85.7	3.89	754.9
	64	53.4	6.25	1,211.6
	128	38.4	8.68	1,684.9
IBM SP2-TN	16	112.5	2.96	575.1
	32	63.4	5.26	1,020.5
	64	38.2	8.73	1,693.7
IBM SP2-TN2	8	120.8	2.76	535.6
	16	70.9	4.70	912.5
	32	40.1	8.32	1,613.4
	64	24.5	13.61	2,640.8
	128	15.9	20.97	4,069.1
IBM SP2-WN	8	112.5	2.96	575.1
	16	64.6	5.16	1,001.5
	32	36.5	9.14	1,772.6
	64	22.7	14.69	2,850.2
	128	15.2	21.94	4,256.5

used the floating-point operation counts for the LU benchmark as reported in [3] – 64.6×10^9 is the operation count for the Class A problem on Cray Y-MP and 319.6×10^9 is the operation count for the Class B problem on Cray C-90. As explained in [3], these floating-point operation counts were obtained using the Cray performance monitoring hardware for benchmarks that were tuned for the Cray systems. We note here that since our implementations of the LU benchmark were

TABLE 4
Performance Comparisons for Benchmark LU (Class B problem set)

System	No. of Procs.	Time (sec.) for 250 iters.	Ratio to C-90/1	Equiv. C-90/1 MFLOPS
Cray C-90	1	648.5	1.00	493.0
	16	51.6	12.57	6,195.9
IBM SP1-TB0	16	604.8	1.07	528.6
	32	348.1	1.86	918.4
	64	207.5	3.13	1,540.8
	128	130.3	4.99	2,453.6
IBM SP1-TB2	32	304.7	2.13	1,049.3
	64	173.5	3.74	1,842.7
	128	106.0	6.12	3,016.1
IBM SP2-TN	32	227.0	2.86	1,408.4
	64	129.1	5.02	2,476.5
IBM SP2-TN2	8	477.3	1.36	669.8
	16	255.4	2.54	1,251.8
	32	141.3	4.59	2,262.6
	64	82.9	7.82	3,856.6
	128	51.2	12.67	6,244.3
IBM SP2-WN	8	429.8	1.51	743.9
	16	234.4	2.77	1,364.0
	32	129.7	5.00	2,465.0
	64	76.8	8.44	4,162.9
	128	47.8	13.57	6,688.5

done independent of the implementations on the Cray systems, the MFLOPS we report may not be true RS/6000 or SP MFLOPS. Therefore, we refer to these as Equivalent Y-MP and Equivalent C-90 MFLOPS in Tables 3 and 4, respectively.

6 CONCLUSIONS

We have implemented the LU benchmark from the NAS Parallel Benchmark suite for distributed memory systems. In this paper, we have presented performance of this implementation on five different IBM SP platforms. We report performance results on up to 128 processors.

As pointed out in [11], LU benchmark has limited parallelism and the primary difficulty is in finding enough parallelism to keep a large number of processors busy. The available parallelism is fine-grained, requiring frequent communication of small size messages. Thus, for this benchmark, large scale distributed memory systems are particularly ill-suited. In spite of these inherent difficulties, the performance results presented here indicate that the SP architecture delivers respectable performance over the entire range of processor configuration considered here. For example, for the Class A problem set, performance of 128-node SP2-WN is close to 22-times that of a single node Y-MP and it is over 3.25-times higher than the performance of an 8-node Y-MP. Similarly, for the Class B problem set, the performance of 128-node SP2-WN is over 13.5-times the performance of a single node C-90 and it is slightly better than the performance of a 16-node C-90. Note also that the performance of SP2-TN2 is within 5% to 10% of the performance of SP2-WN, over the entire range of processor configurations for which we have performance data. For both systems, when the number of processors is increased from 8 to 128 processors, on Class B problem set, the relative efficiency is close to 60%. This is a respectable figure considering the fact that on the 128-node system less than 20% of the available memory is used on each processor.

Another noteworthy aspect is the performance improvement from SP1-TB0 to SP2-WN, which are the two extremes in the family of SP systems considered here. When the number of processors is identical, performance of SP2-WN is higher by a factor of 2.6 to 2.8. Note that in both cases, the same parallel implementation of the LU benchmark was used, the only difference being the variations in the tuning for single processor performance. As compared to the performance of SP1-TB2, the SP2-WN performance is higher by a factor more than 2.2 which is higher than the ratio of the peak performance of the individual processors.

Acknowledgements: The author is grateful to Bob Walkup for his expert assistance in completing the experiments on the 128-way SP1-TB0 and SP1-TB2 systems. The performance measurements on SP1-TB2 and the measurements on the SP2-TN2 system were completed with the help of Wei-Hwan Chiang and those on the SP2-TN at the Maui High Performance Computing Center were completed with the help of Mark Smith. Thanks to both. The author is also grateful to NASA Ames Research Center and to the helpful staff at NAS for allowing the use of their SP2 system.

References

[1] D. Bailey, J. Barton, T. Lasinski, and H. Simon. The NAS Parallel Benchmarks. NASA Ames Research Center Technical Report No. RNR-91-002, 1991.

[2] D. Bailey, E. Barszcz, L. Dagum, and H. Simon. NAS Parallel Benchmark Results 3-94. Proceedings of 1994 Scalable High Performance Computing Conference, 1994.

[3] D. Bailey, E. Barszcz, L. Dagum, and H. Simon. NAS Parallel Benchmark Results 10–94. NASA Ames Research Center Technical Report No. RNR-94-006, 1994.

[4] V. Naik. Performance of NAS Parallel Application-Benchmarks on IBM SP1. Proceedings of Scalable High Performance Computing Conference, IEEE, 1994.

[5] V. Naik. A Scalable Implementation of NAS Parallel Benchmark BT on Distributed Memory Systems. IBM Systems Journal, vol. 34, no. 2, 1995.

[6] S. Yoon, D. Kwak, and L. Chang. LU-SGS Implicit Algorithm for Three-dimensional Incompressible Navier-Stokes Equations with Source Term. AIAA Paper 89–1964–CP, 1989.

[7] N. Naik, V. Naik, and M. Nicoules. Parallelization of a Class of Implicit Finite Difference Schemes in Computational Fluid Dynamics. International Journal of High Speed Computing, vol. 5, 1993.

[8] Special Issue on IBM RISC System/6000 Processor, IBM Journal of Research and Development, vol. 34, no. 1, 1990.

[9] Special Issue on POWER2 and PowerPC, IBM Journal of Research and Development, vol. 38, no. 5, 1994.

[10] Special Issue on IBM SP2 Systems. IBM Systems Journal, vol. 34, no. 2, 1995.

[11] E. Barszcz, R. Fatoohi, V. Venkatakrishnan, and S. Weeratunga. Solution of Regular, Sparse Triangular Linear Systems on Vector and Distributed-Memory Multiprocessors. NASA Ames Research Center Technical Report No. RNR-93-007, 1993.