Performance Comparison of a Set of Periodic and Non-Periodic Tridiagonal Solvers on SP2 and Paragon Parallel Computers

*Xian-He Sun**

Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803-4020

Stuti Moitra

Scientific Applications Branch
NASA Langley Research Center
Hampton, VA 23681-0001

Abstract

Various tridiagonal solvers have been proposed in recent years for different parallel platforms. In this paper, the performance of three tridiagonal solvers, namely, the parallel partition LU algorithm, the parallel diagonal dominant algorithm, and the reduced diagonal dominant algorithm, is studied. These algorithms are designed for distributed-memory machines and are tested on an Intel Paragon and an IBM SP2 machines. Measured results are reported in terms of execution time and speedup. Analytical study are conducted for different communication topologies and for different tridiagonal systems. The measured results match the analytical results closely. In addition to address implementation issues, performance considerations such as problem sizes and models of speedup are also discussed.

*This work was supported in part by the NationalAeronautics and Space Administration under NASA contract NAS1-1672 and byLouisiana Education Quality Support Fund.

1 Introduction

Distributed-memory parallel computers dominate today's parallel computing arena. These machines, such as the Kendall Square KSR-1, Intel Paragon, TMC CM-5, and the recently announced IBM SP2 and Cray T3D concurrent systems, have successfully delivered high performance computing power for solving certain of the so-called "grand-challenge" problems [1]. Despite initial success, parallel machines have not been widely accepted in production engineering environment. On a parallel computing system, a task has to be partitioned and distributed appropriately among processors to reduce communication cost and to achieve load balance. More importantly, even with a careful partitioning and mapping, the performance of an algorithm might be still unsatisfactory, since conventional sequential algorithms may be serial in nature and may not be implemented efficiently on parallel machines. In many cases, new algorithms must be introduced to increase parallelism and to take advantage of the computing power of the scalable parallel hardware.

Solving tridiagonal systems is a basic computational kernel of many scientific applications. Tridiagonal systems appear in multigrid methods, Alternating Direction Implicit (ADI) method, wavelet collocation method, and in line-SOR preconditioners for conjugate gradient methods [2]. In addition to solving PDE's, tridiagonal systems also arise in digital signal processing, image processing, stationary time series analysis, and in spline curve fitting. Because its importance, intensive research has been carried out on the development of efficient parallel tridiagonal solvers. Many algorithms have been proposed [3, 4, 5]. In general, parallel tridiagonal solvers require global communications which makes them inefficient on distributed-memory architectures. The algorithm given by Lawrie and Sameh [6], the algorithm given by Wang [7], and the *Parallel Partition LU* (PPT) algorithm, the *Parallel Diagonal Dominant* (PDD) algorithm proposed by Sun, Zhang, and Ni [3] are designed for medium and coarse grain computing, i.e. for the case of $p < n$ or $p << n$, where p is the number of processors available and n is the order of the linear system. They are substructuring methods. These algorithms partition the original problem into sub-problems. The sub-problems are solved in parallel, and then the solutions of the sub-problems are combined to obtain the final solution.

Among the above substructuring methods, the PPT algorithm has a similar computation and communication complexity as Wang's algorithm and has a similar substructure as the algorithm of Lawrie and Sameh. The PDD algorithm, designed for strictly diagonally dominant problems, is the most efficient, when it is applicable. Recently, Sun has extended the PDD algorithm, and the PPT algorithm, for solving periodic systems and proposed a variation of the PDD algorithm, the *Reduced PDD* algorithm [2]. The Reduced PDD algorithm maintains the minimum communication provided by the PDD algorithm but has a reduced operation count. It has a smaller operation count than the conventional sequential algorithm for many applications. While sequential algorithm requires more operations for solving periodic systems, the three parallel algorithms basically have the same

operation count for solving periodic and non-periodic systems. In this paper, the performance of the PPT algorithm, the PDD algorithm, and the Reduced PDD algorithm are carefully examined. Operation counts are presented for comparison with best sequential algorithms for both periodic and non-periodic systems. Communication complexities are studied for three different communication topologies: 2-D torus, multi-stage Omega network, and hypercube. Implementation is conducted on two distributed-memory computers: the Intel Paragon and the IBM SP2, for solving both periodic and non-periodic systems. Speedup over the best sequential algorithm is compared with speedup over the uniprocessor processing of the parallel program. The influence of problem size on performance and the usefulness of different models of speedup are also discussed. Experimental results match analytical results well. Experimental and theoretical results show that the PDD and the Reduced PDD algorithm are efficient and scalable. They are good candidates for distributed-memory machines.

This paper is organized as follows. Section 2 introduces the three parallel algorithms. Section 3 provides analytical comparisons of the three algorithms in terms of computation and communication, for solving periodic and non-periodic systems, and for solving single systems and systems with multiple right-hand-sides. Related sequential algorithms are also discussed. Section 4 presents experimental results on an Intel Paragon and an IBM SP2 multicomputer. Performance comparison and considerations of the three algorithms on the two parallel platforms are also discussed. Finally, Section 5 gives conclusions and final remarks.

2 Parallel Tridiagonal Algorithms

The PPT, PDD, and Reduced PDD algorithms are introduced in the following four sectons. They are first introduced for solving non-periodic systems and then extended for solving periodic system. Interested readers may refer [3] and [2] for details of the algorithms, especially for accuracy analysis and for extending these algorithms for general banded linear systems.

2.1 A Partition Method for Parallel Processing

A tridiagonal system is a linear system of equations

$$Ax = d, \tag{1}$$

where $x = (x_1, \cdots, x_n)^T$ and $d = (d_1 \cdots, d_n)^T$ are n-dimensional vectors, and A is a tridiagonal matrix with order n:

$$A = \begin{bmatrix} b_0 & c_0 & & & & & \\ a_1 & b_1 & c_1 & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & a_{n-2} & b_{n-2} & c_{n-2} & \\ & & & & a_{n-1} & b_{n-1} & \end{bmatrix} = [a_i, b_i, c_i] \tag{2}$$

To solve Eq. (1) efficiently on parallel computers, we partition A into submatrices. For convenience we assume that $n = p \cdot m$, where p is the number of processors available. The matrix A in Eq. (1) can be written as

$$A = \tilde{A} + \Delta A,$$

where $\tilde{A}$ is a block diagonal matrix with diagonal submatrices $A_i (i = 0, \cdots, p-1)$. The submatrices $A_i (i = 0, \cdots, p-1)$ are $m \times m$ tridiagonal matrices. Let e_i be a column vector with its ith $(0 \leq i \leq n-1)$ element being one and all the other entries being zero. We have

$$\Delta A = [a_m e_m, c_{m-1} e_{m-1}, a_{2m} e_{2m}, c_{2m-1} e_{2m-1}, \cdots, c_{(p-1)m-1} e_{(p-1)m-1}] \begin{bmatrix} e^T_{m-1} \\ e^T_m \\ \cdot \\ \cdot \\ e^T_{(p-1)m-1} \\ e^T_{(p-1)m} \end{bmatrix} = V E^T,$$

where both V and E are $n \times 2(p-1)$ matrices. Thus, we have

$$A = \tilde{A} + V E^T.$$

Based on the matrix modification formula originally defined by Sherman and Morrison [8] for rank-one changes, and assuming that all A_i's are invertible, Eq. (1) can be solved by

$$x = A^{-1} d = (\tilde{A} + V E^T)^{-1} d \tag{3}$$

$$x = \tilde{A}^{-1} d - \tilde{A}^{-1} V (I + E^T \tilde{A}^{-1} V)^{-1} E^T \tilde{A}^{-1} d. \tag{4}$$

Let

$$\tilde{A} \tilde{x} = d \tag{5}$$

$$\tilde{A}Y = V \tag{6}$$

$$h = E^T\tilde{x} \tag{7}$$

$$Z = I + E^TY \tag{8}$$

$$Zy = h \tag{9}$$

$$\Delta x = Y y. \tag{10}$$

Equation (4) becomes

$$x = \tilde{x} - \Delta x. \tag{11}$$

In Eqs. (5) and (6), $\tilde{x}$ and Y are solved by the LU decomposition method. By the structure of $\tilde{A}$ and V, this is equivalent to solving

$$A_i[\tilde{x}^{(i)}, v^{(i)}, w^{(i)}] = [d^{(i)}, a_{im}e_0, c_{(i+1)m-1}e_{m-1}], \tag{12}$$

$i = 0, \cdots, p-1$. Here $\tilde{x}^{(i)}$ and $d^{(i)}$ are the ith block of $\tilde{x}$ and d, respectively, and $v^{(i)}, w^{(i)}$ are possible nonzero column vectors of the ith row block of Y. Equation (12) implies that we only need to solve three linear systems of order m with the same LU decomposition for each i $(i = 0, \cdots, p-1)$.

Solving Eq. (9) is the major computation involved in the conquer part of our algorithms. Different approaches have been proposed for solving Eq.(9), which results in different algorithms for solving tridiagonal systems [3].

Note that I is an identity matrix. $I + E^T\tilde{A}^{-1}V$ and Z are pentadiagonal matrices of order $2(p-1)$. We introduce a permutation matrix P such that

$$P\ z\ =\ (z_1,\ z_0,\ z_3,\ z_2, ...,\ z_{2p-3},\ z_{2(p-2)}\ )^T \qquad \text{for all } z \in R^{2(p-1)}$$

Eq. 4 then becomes

$$x = \tilde{A}^{-1}d - \tilde{A}^{-1}VP(P + E^T\tilde{A}^{-1}VP)^{-1}E^T\tilde{A}^{-1}d. \tag{13}$$

The intermediate matrix $Z' = P + E^T\tilde{A}^{-1}VP$, then, is a tridiagonal matrix of order $2(p-1)$. The modified solving sequence become:

$$\tilde{A}\tilde{x} = d \tag{14}$$

$$\tilde{A}Y = VP \tag{15}$$

$$h = E^T\tilde{x} \tag{16}$$

$$Z' = P + E^TY \tag{17}$$

$$Z'y = h \tag{18}$$

$$\Delta x \quad = \quad Y y. \tag{19}$$

where equations 15 and 17 are modified to include permutations. Theses permutations make the intermediate matrix Z a tridiagonal system and lead to a reduced computation cost.

2.2 The Parallel Partition LU (PPT) Algorithm

Based on the matrix partitioning technique described previously, using p processors, the PPT algorithm to solve (1) consists of the following steps:

Step 1. Allocate $A_i, d^{(i)}$, and elements $a_{im}, c_{(i+1)m-1}$ to the ith node, where $0 \leq i \leq p-1$.

Step 2. Solve (12). All computations can be executed in parallel on p processors.

Step 3. Send $\tilde{x}_0^{(i)}, \tilde{x}_{m-1}^{(i)}, v_{m-1}^{(i)}, v_0^{(i)}, w_{m-1}^{(i)}, w_0^{(i)}$ to all the other nodes from the ith node to form the matrix Z' and vector h (see Eq.s (7) and (8)) on each node. Here and throughout the subindex indicates the component of the vector.

Step 4. Use the LU decomposition method to solve $Z'y = h$ (see Eq. (9)) on all nodes simultaneously. Note that Z' is a $2(p-1)$ dimensional tridiagonal matrix.

Step 5. Compute (19) and (11). We have

$$\Delta x^{(i)} = [v^{(i)}, w^{(i)}] \begin{pmatrix} y_{(2i-1)} \\ y_{2i} \end{pmatrix}$$

$$x^{(i)} = \tilde{x}^{(i)} - \Delta x^{(i)}$$

Step 3 requires a global total-data-exchange (all-to-all broadcast) communication[1].

2.3 The Parallel Diagonal Dominant (PDD) Algorithm

The matrix Z in Eq. (9) has the form

[1] The total-data-exchange communication can be replaced by one data-gathering communication plus one data-scattering communication. However, on most communication topologies (include 2-D mesh, multi-stage Omega network, and hypercube), the latter has a higher communication cost than the former [9].

$$Z = \begin{bmatrix} 1 & w_{m-1}^{(0)} & 0 & & & & & \\ v_0^1 & 1 & 0 & w_0^{(1)} & & & & \\ v_{m-1}^{(1)} & 0 & 1 & w_{m-1}^{(1)} & 0 & & & \\ & \cdot & \cdot & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & 1 & 0 & w_0^{(p-2)} \\ & & & & v_{m-1}^{(p-2)} & 0 & 1 & w_{m-1}^{(p-2)} \\ & & & & & 0 & v_0^{(p-1)} & 1 \end{bmatrix}$$

In practice, for a diagonally dominant tridiagonal system, the magnitude of the last component of $v^{(i)}$, $v_{m-1}^{(i)}$, and the first component of $w^{(i)}$, $w_0^{(i)}$, may be smaller than machine accuracy when $p << n$. In this case, $w_0^{(i)}$ and $v_{m-1}^{(i)}$ can be dropped, and Z becomes a diagonal block system consisting of $(p-1)$ 2×2 independent blocks. Thus, Eq.(9) can be solved efficiently on parallel computers, which leads to the highly efficient *parallel diagonal dominant* (PDD) algorithm.

Using p processors, the PDD algorithm consists of the following steps:

Step 1. Allocate $A_i, d^{(i)}$, and elements $a_{im}, c_{(i+1)m-1}$ to the ith node, where $0 \le i \le p-1$.

Step 2. Solve (12). All computations can be executed in parallel on p processors.

Step 3. Send $\tilde{x}_0^{(i)}, v_0^{(i)}$ from the ith node to the $(i-1)$th node, for $i = 1, \cdots, p-1$.

Step 4. Solve

$$\begin{bmatrix} 1 & w_{m-1}^{(i)} \\ v_0^{(i+1)} & 1 \end{bmatrix} \begin{pmatrix} y_{2i} \\ y_{2i+1} \end{pmatrix} = \begin{pmatrix} \tilde{x}_{m-1}^{(i)} \\ \tilde{x}_0^{(i+1)} \end{pmatrix}$$

in parallel on the ith node for $0 \le i \le p-2$. Then send y_{2i} from the ith node to the $(i+1)$th node, for $i = 0, \cdots, p-2$.

Step 5. Compute (10) and (11). We have

$$\Delta x^{(i)} = [v^{(i)}, w^{(i)}] \begin{pmatrix} y_{(2i-1)} \\ y_{2i} \end{pmatrix}$$

$$x^{(i)} = \tilde{x}^{(i)} - \Delta x^{(i)}$$

In all of these, one has only two neighboring communications.

2.4 The Reduced PDD Algorithm

The PDD algorithm is very efficient in communication. However, the PDD algorithm has a larger computation count than the conventional sequential algorithm, Thomas algorithm [10]. The Reduced PDD algorithm is proposed in order to further enhance computation [2].

In the last step, Step 5, of the PDD algorithm, the final solution, x, is computed by combining the intermediate results concurrently on each processor,

$$x^{(k)} = \tilde{x}^{(k)} - y_{(2k-1)}v^{(k)} - y_{2k}w^{(k)},$$

which requires $4(n-1)$ sequential operations and $4m$ parallel operations, if $p = n/m$ processors are used. The PDD algorithm drops off the first element of w, w_0, and the last element of v, v_{m-1}, in solving Eq. (9). In [2], we have shown that, for symmetric Toeplitz tridiagonal systems, when m is large enough, we may drop off $v_i, i = j, j+1, \cdots, m-1$, and $w_i, i = 0, 1, \cdots, j-1$, for some integer $j > 0$, while maintaining the required accuracy. If we replace v_i by $\tilde{v}_i$, where $\tilde{v}_i = v_i$, for $i = 0, 1, \cdots, j-1$, $\tilde{v}_i = 0$, for $i = j, \cdots, m-1$; and replace w by $\tilde{w}$, where $\tilde{w}_i = w_i$ for $i = j, \cdots, m-1$, and $\tilde{w}_i = 0$, for $i = 0, 1, \cdots, j-1$; and use $\tilde{v}, \tilde{w}$ in Step 5, we have

Step 5'

$$\Delta x^{(k)} = [\tilde{v}, \tilde{w}] \begin{pmatrix} y_{(2k-1)} \\ y_{2k} \end{pmatrix}$$

$$x^{(k)} = \tilde{x}^{(k)} - \Delta x^{(k)}. \tag{20}$$

It only requires $4j/p$ parallel operations. Replacing Step 5 of the PDD algorithm by Step 5', we get the Reduced PDD algorithm [2]. In general, j is quite small. For instance, when error tolerance ϵ equals 10^{-4}, j equals either 10 or 7 when λ, the magnitude of the off diagonal elements equals $\frac{1}{3}$ or $\frac{1}{4}$ respectively, the diagonal elements being equal to 1. The integer j reduces to 4 for $0 < \lambda \leq \frac{1}{9}$. In general how to determine the value of j is a state-of-art. For symmetric Toeplitz tridiagonal systems, however, a formula is derived in [2] to determine the truncation number j automaticly based on the diagonal dominance of the matrix and the desired accuracy. Interested readers may refer [2] for accuracy analysis of the PDD and reduced PDD algorithm.

2.5 Periodic Tridiagonal Systems

Many PDE's arising in real applications have periodic boundary conditions. For instance, to study a physical phenomenon in an infinite region, we often model only a small subdomain, applying periodic boundary conditions on the boundary. The resulting linear systems have the form of

$$A = \begin{bmatrix} b_0 & c_0 & & & & & a_0 \\ a_1 & b_1 & c_1 & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & a_{n-2} & b_{n-2} & c_{n-2} \\ c_{n-1} & & & & & a_{n-1} & b_{n-1} \end{bmatrix},$$

and are called *periodic tridiagonal systems*. On sequential machines, periodic tridiagonal systems are solved by combining the solutions of two different right-hand-sides [11], which increases the operation count from $8n - 7$ to $14n - 16$.

The partition method introduced in Section 2.1 can be extended to periodic tridiagonal systems. The difference is that, for periodic systems, the matrix Z becomes a periodic system of order $2p$:

$$Z = \begin{bmatrix} 1 & 0 & w_0^{(0)} & 0 & & & & & & v_0^{(0)} \\ 0 & 1 & w_{m-1}^{(0)} & 0 & & & & & & v_{m-1}^{(0)} \\ 0 & v_0^1 & 1 & 0 & w_0^{(1)} & & & & & \\ & v_{m-1}^{(1)} & 0 & 1 & w_{m-1}^{(1)} & 0 & & & & \\ & & \cdot & \cdot & \cdot & \cdot & \cdot & & & \\ & & & \cdot & \cdot & \cdot & \cdot & \cdot & & \\ & & & & \cdot & \cdot & 1 & 0 & w_0^{(p-2)} & \\ & & & & & v_{m-1}^{(p-2)} & 0 & 1 & w_{m-1}^{(p-2)} & 0 \\ w_0^{(p-1)} & & & & & & 0 & v_0^{(p-1)} & 1 & 0 \\ w_{m-1}^{(p-1)} & & & & & & 0 & v_{m-1}^{(p-1)} & 0 & 1 \end{bmatrix}$$

The dimension of Z is slightly higher than in the non-periodic case. It changes from $2(p-1)$ to $2p$. When $p << n$, the influence of the order increase is negligible. In fact, periodic systems simply make the load on the 0th and (p-1)th processor identical to the load on all of the other processors, in Step 2 and in Step 5 of the parallel algorithms. For the PPT algorithm, Step 3, the communication step, remains the same ($v_0^{(0)}$, $v_{m-1}^{(0)}$, $w_0^{(p-1)}$, $w_{m-1}^{(p-1)}$ are equal zero in solving non-periodic systems), Step 4 has a minor operation count increase. For diagonally dominant, periodic systems, the reduced system Z is also periodic.

$$Z = \begin{bmatrix} 1 & 0 & & & & & & v_0^{(0)} \\ 0 & 1 & w_{(m-1)}^{(0)} & & & & & \\ & v_0^{(1)} & 1 & 0 & & & & \\ & & 0 & 1 & \cdot & & & \\ & & & \cdot & \cdot & \cdot & & \\ & & & & \cdot & \cdot & w_{m-1}^{(p-2)} & \\ & & & & & v_0^{(p-1)} & 1 & 0 \\ w_{m-1}^{(p-1)} & & & & & & 0 & 1 \end{bmatrix}$$

The parallel computation time remains the same for the PDD and the Reduced PDD algorithm. The only change is in communication. For periodic systems, the communication at Step 3 changes from one dimensional array communication to ring communication. The communication time is also unchanged for any architecture supporting ring communication. Figure 1 shows the communication pattern of the PDD and Reduced PDD algorithm for periodic systems.

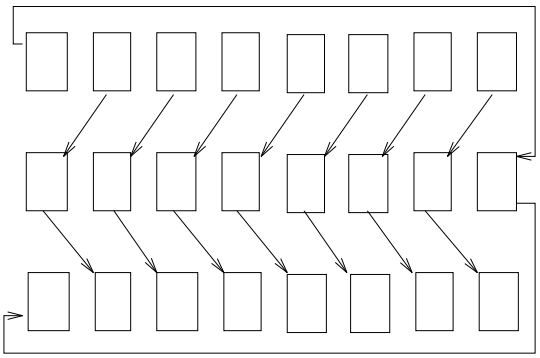

Figure 1. Communication Pattern for Solving Periodic Systems.

3 Operation Comparison

Table 1 gives the computation and communication count of the tridiagonal solvers under consideration for solving non-periodic systems. Tridiagonal systems arising in many applications are multiple right-hand-side (RHS) systems. They are usually "kernels" in much larger codes. The computation and communication counts for solving multiple RHS systems are listed in Table 1, in which the factorization of matrix $\tilde{A}$ and computation of Y are not considered (see Eq.(5) and (6) in Section 2). Parameter $n1$ is the number of RHS. Note that for multiple RHS systems, the communication cost increases with the number of RHS. For the PPT algorithm, the communication cost also increase with the ensemble size. The computational saving of the Reduced PDD algorithm is not only in step 5, the final modification step, but also in other steps. Since we only need j elements of vector v and w for the final modification in the Reduced PDD algorithm (see Eq. (20) in Section 3), we only need to compute j elements for each column of V in solving Eq. (6). Formulas for computing the integer j for particular circumstances can be found in [2]. The best sequential algorithm is the conventional Thomas algorithm [11], the LU decomposition method for tridiagonal systems.

Communication cost has a great impact on overall performance. For most distributed-memory computers, communicate time with nearest neighbors is found to vary linearly with problem size. Let S be the number of bytes to be transferred. Then the transfer time to communicate with a neighbor can be expressed as $\alpha + S\beta$, where α is a fixed startup time and β is the incremental transmission time per byte. Assuming 4 bytes are used for each real number, Steps 3 and 4 of the PDD and Reduced PDD algorithm take $\alpha + 8\beta$ and $\alpha + 4\beta$ time respectively on any architecture

System	Algorithm	Computation	Communication
Single system	best sequential	$8n-7$	0
	the PPT	$17\frac{n}{p}+16p-23$	$(2\alpha+8p\beta)(\sqrt{p}-1)$
	the PDD	$17\frac{n}{p}-4$	$2\alpha+12\beta$
	the Reduced PDD	$11\frac{n}{p}+6j-4$	$2\alpha+12\beta$
Multiple right sides	best sequential	$(5n-3)\cdot n1$	0
	the PPT	$(9\frac{n}{p}+10p-11)\cdot n1$	$(2\alpha+8p\cdot n1\cdot\beta)(\sqrt{p}-1)$
	the PDD	$(9\frac{n}{p}+1)\cdot n1$	$(2\alpha+8n1\cdot\beta)$
	the Reduce PDD	$(5\frac{n}{p}+4j+1)\cdot n1$	$(2\alpha+8n1\cdot\beta)$

Table 1. Comparison of Computation and Communication (Non-Periodic)

which supports single array topology. The communication cost of the total-data-exchange communication is highly architecture dependent. The listed communication cost of the PPT algorithm, in Table 1, 2, and 3, is based on a square 2-D torus with p processors (i.e. 2-D mesh, wrap-around, square) [12]. With a hypercube or multi-stage Omega network connection, the communication cost would be $\log(p)\alpha+12(p-1)\beta$ and $\log(p)\alpha+8(p-1)n1\cdot\beta$ for single systems and systems with multiple RHS respectively [3, 13].

If boundary conditions are periodic, tridiagonal systems arising in scientific applications are periodic tridiagonal systems. Computation and communication counts for solving periodic systems are listed in Table 2. The conventional sequential algorithm used is the periodic Thomas algorithm [11]. Compared with Table 1, we can see, while the best sequential algorithm has a increased operation count, the parallel algorithms have the same operation and communication count for both periodic and non-periodic systems, except for the PPT algorithm which has a slightly increased operation count. However, for the PDD and Reduced PDD algorithm, the communication is given for any architecture which supports Ring communication, instead of 1-D array. Notice that when $j < n/2$, the Reduced PDD algorithm has a smaller operation count than that of Thomas algorithm for periodic systems with multiple RHS.

The computation counts given in Table 1 and 2 are for general tridiagonal systems. For symmetric Toeplitz tridiagonal systems, a fast method proposed by Malcolm and Palmer [14] has a smaller computation count than Thomas algorithm for systems with single RHS. It only requires $5n+2k-3$ counts for arithmetic, where k is a decay parameter depending on the diagonal dominancy of the system. Formulas are available to compute the upper and the lower bounds of parameter k [14]. The computational savings of Malcolm and Palmer's method is in the LU decomposition. For systems with multiple RHS, in which the factorization cost is not considered, the Malcolm and Palmer's method and Thomas method have the same computation count. Table 3 gives the computation and communication counts of the PDD and the Reduced PDD algorithms based on Malcolm and

System	Algorithm	Computation	Communication
Single system	best sequential	$14n - 16$	0
	the PPT	$17\frac{n}{p} + 16p - 7$	$(2\alpha + 8p\beta)(\sqrt{p} - 1)$
	the PDD	$17\frac{n}{p} - 4$	$2\alpha + 12\beta$
	the Reduced PDD	$11\frac{n}{p} + 6j - 4$	$2\alpha + 12\beta$
Multiple right sides	best sequential	$(7n - 1) \cdot n1$	0
	the PPT	$(9\frac{n}{p} + 10p - 3) \cdot n1$	$(2\alpha + 8p \cdot n1 \cdot \beta)(\sqrt{p} - 1)$
	the PDD	$(9\frac{n}{p} + 1) \cdot n1$	$(2\alpha + 8n1 \cdot \beta)$
	the Reduce PDD	$(5\frac{n}{p} + 4j + 1) \cdot n1$	$(2\alpha + 8n1 \cdot \beta)$

Table 2. Comparison of Computation and Communication (Periodic)

Algorithm	Matrix	Best sequential	Parallel Algorithm	
			Computation	Communication
PDD Algorithm	Non-periodic	$5n + 2k - 3$	$14\frac{n}{p} + 2k$	$2\alpha + 12\beta$
	Periodic	$11n + 2k - 12$	$14\frac{n}{p} + 2k$	$2\alpha + 12\beta$
Reduced PDD Alg.	Non-periodic	$5n + 2k - 3$	$8\frac{n}{p} + 2k + 6j$	$2\alpha + 8\beta$
	Periodic	$11n + 2k - 12$	$8\frac{n}{p} + 2k + 6j$	$2\alpha + 8\beta$
PPT Algorithm	Non-periodic	$5n + 2k - 3$	$14\frac{n}{p} + 2k + 16p - 19$	$(2\alpha + 8p\beta)(\sqrt{p} - 1)$
	Periodic	$11n + 2k - 12$	$(14\frac{n}{p} + 2k + 16p - 3)$	$(2\alpha + 8p\beta)(\sqrt{p} - 1)$

Table 3. Computation and Communication Counts for Symmetric Toeplitz Systems

Palmer's algorithm. The computation counts of the two algorithms are reduced by the fast method for solving the sub-systems. Table 3 presents computation and communication counts for solving systems with a single RHS only. For systems with multiple RHS, the computation counts remain the same as in Table 1 and 2 for all the periodic and non-periodic systems.

4 Experimental Results

The PDD and the Reduced PDD algorithms were implemented on the 48-node IBM SP2 and 72-node Intel Paragon available at NASA Langley Research Center. Both the SP2 and Paragon machines are distributed-memory parallel computers which adopt message-passing communication paradigms and support virtual memory. Each processor (node) of the SP2 is either functionally equivalent to a RISC System/6000 desktop system (thin node) or a RISC System/6000 deskside system (wide node). The Paragon XP/S supercomputer uses the i860 XP microprocessor which includes a RISC integer core processing unit and three separate on-chip caches for page translation, data, and instructions. The Langley SP2 has 48 wide nodes with 128 Mbytes local memory and peak performance of 266 MFLOPS each. In contrast, the Langley Paragon has 72 nodes with 32 Mbytes

of local memory and peak performance of 75 MFLOPS each. The heart of all distributed memory parallel computers is the interconnection network that links the processors together. The SP2 High-Performance Switch is a multi-stage packet switched Omega network that provides a minimum of four paths between any pair of nodes in the system. The processors of Intel Paragon are connected in a two-dimensional rectangular mesh topology. The diameter of the 2-D mesh topology will increase with the number of processors. For the SP2, the measured latency (start time), α, is 45 microseconds and the measured transmission time per byte, β, is 2 micorseconds. For Paragon, the measured latency and transmission time per byte is 50 microseconds and 6 microseconds, respectively.

As an illustration of the algorithm and analytical results given by previous sections, a sample matrix is tested. This sample matrix is a diagonal dominant, symmetric, Toeplitz system

$$A = \begin{bmatrix} 1 & \frac{1}{3} & & & & & \\ \frac{1}{3} & 1 & \frac{1}{3} & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \frac{1}{3} & 1 & \frac{1}{3} \\ & & & & & \frac{1}{3} & 1 \end{bmatrix} \quad \text{or} \quad A = \begin{bmatrix} 1 & \frac{1}{3} & & & & & \frac{1}{3} \\ \frac{1}{3} & 1 & \frac{1}{3} & & & & \\ & \cdot & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & \frac{1}{3} & 1 & \frac{1}{3} \\ \frac{1}{3} & & & & & \frac{1}{3} & 1 \end{bmatrix}$$

for non-periodic and periodic system respectively. $j = 17$ has be chosen for the Reduced PDD algorithm to reach the single precision accuracy, 10^{-7}.

Speedup is one of the most frequently used performance metrics in parallel processing. It is defined as sequential execution time over parallel execution time. Parallel algorithms often exploit parallelism by sacrificing mathematical efficiency. To measure the true parallel processing gain, the sequential execution time should be based on a commonly used sequential algorithm. To distinguish it from other interpretations of speedup, the speedup measured versus a commonly used sequential algorithm has been called *absolute speedup* [15]. Another widely used interpretation is the *relative speedup* [15], which uses the uniprocessor execution time of the parallel algorithm as the sequential time. Relative speedup measures the performance variation of an algorithm with the number of processors, which is commonly used in scalability studies. Both Amdahl's law [16] and Gustafson's scaled speedup [17] are based on relative speedup. In this study we first use relative speedup to study the performance of the PDD and Reduced PDD algorithm, then, the absolute speedup is used to compare these two algorithms with the conventionally used sequential algorithm.

Since execution time varies with communication/computation ratio on a parallel machine, the problem size is an important factor in performance evaluation, especially for machines supporting virtual memory. Virtual address space separates the user logical memory from physical memory.

This separation allows an extremely large virtual memory to be provided (with a much slower memory access time) on a sequential machine when only a small physical memory is available. If the problem size is larger than physical memory, data has to be swapped in from and out to secondary memory, which may lead to inefficient sequential processing and unreasonably high speedup. If the problem size is too small, on the other hand, when the number of processors increases, the work load on each processors will drop quickly, which may lead to extremely high communication/computation ratio and unacceptably low performance. As studied in [15], the right choice of initial problem size is the problem size which reaches an appropriate portion of the asymptotic speed, the sustained uniprocessor speed corresponding to the main memory access [15]. The nodes of SP2 and Paragon have different processing powers and local memory sizes. For a fixed 1024 RHS, following the asymptotic speed concept, the order of matrix for SP2 has been chosen to be 6400 and the order of matrix for Paragon has been chosen to be 1600. Figures 2 and 3 show the measured speedup of the PDD algorithm solving the sample periodic system when the large problem size, $n = 6400$, is solved on Paragon and the small problem size, $n = 1600$, is solved on SP2. For comparison, ideal speedup, where speedup equals p when p processors available, is also plotted with the measured speedups. As indicated above, the large problem size leads to an unreasonable superlinear speedup on Paragon and the small problem size leads to a disappointing low performance on SP2.

From the problem size point of view, speedup can be divided into the *fixed-size speedup* and the *scaled speedup*. Fixed-size speedup fixes the problem size. Scaled-speedup scales the problem size with the number of processors. Fixed-size speedup emphasizes how much execution time can be reduced for a given application with parallel processing. Amdahl's law [16] is based on the fixed-size speedup. The scaled speedup is concentrated on exploring the computational power of parallel computers for solving otherwise intractable large problems. Depending on the scaling restrictions of the problem size, the scaled speedup can be classified as the *fixed-time speedup* [17] and the *memory-bounded speedup* [18]. As the number of processors increases, memory-bounded speedup scales problem size to utilize the associated memory increase. In general, operation count increases much faster than memory requirement. Therefore, in general, the work load on each processor will not decrease with the increase in number of processors in memory-bounded scaleup. Thus, scaled speedup is more likely to get a higher speedup than that of fixed-size speedup. Figures 4 and 5 depict the speedup of the fixed-size and memory-bounded speedup of the PDD and the Reduced PDD algorithm for solving the periodic system, respectively, on the Intel Paragon. From Figs. 4 and 5 we can see that the PDD and the Reduced PDD algorithm have the same speedup pattern. This similarity is very reasonable because these two algorithms share the same computation and communication pattern. It has been proven that the PDD algorithm, and therefore the Reduced PDD algorithm, are perfectly scalable (under the assumption that the number of right-hand-sides is fixed and the order of matrix increases with the number of processors), in terms of isospeed

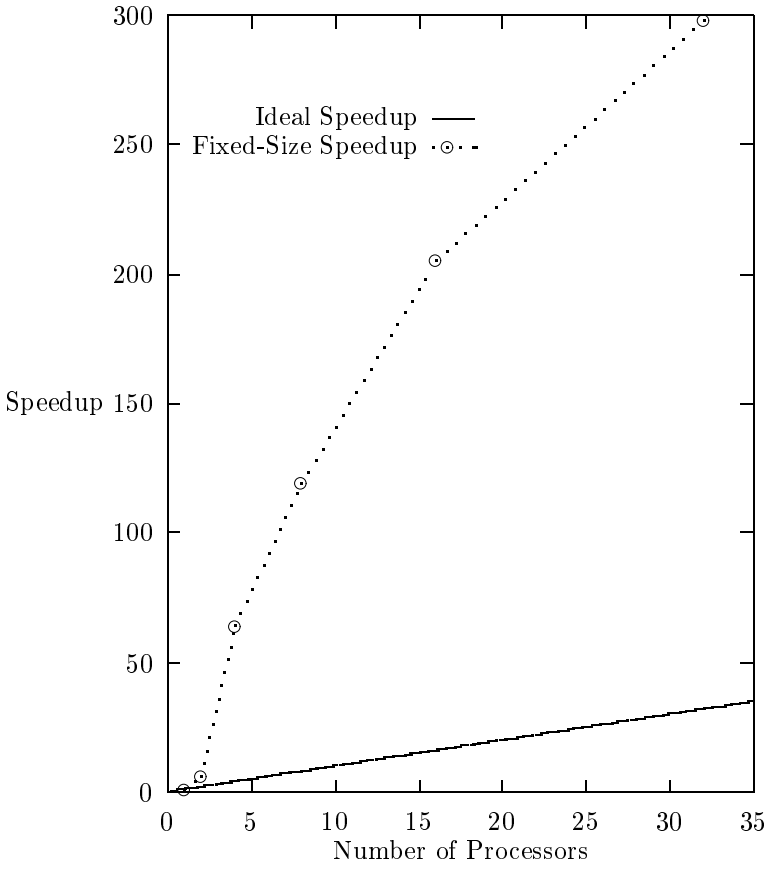

Figure 2. Superlinear Speedup with Large Problem Size on Intel Paragon
1024 System of Order 6400, periodic

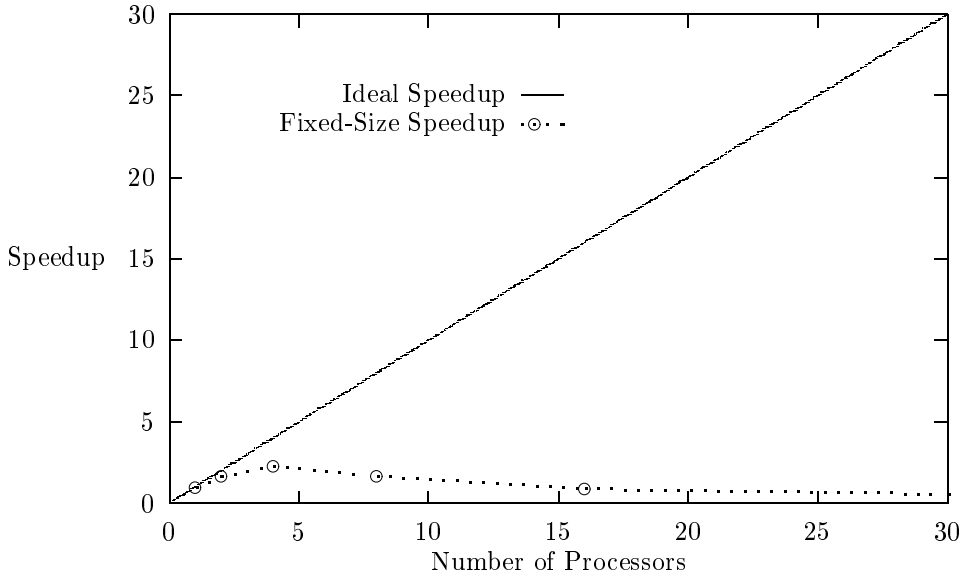

Figure 3. Inefficient Performance with Small Problem Size on SP2
1024 System of Order 1600, periodic

scalability [2], on any architecture which supports ring communication network. However, ring communication cannot be embedded in 2-D mesh topologies perfectly, unless a wrap-around is supported. Thus, the communication cost of the algorithms increases slightly with the increase of the number of processors. The fact that the memory-bounded speedups on the Paragon are slightly below the ideal speedup is very reasonable. The influence of the communication cost has been reflected in the measured speedup. Figure 6 demonstrates the speedups of the PDD algorithm on the SP2 machine. Since the cost of one-to-one communication does not increase with the number of processors on the SP2 multi-stage Omega network, for number of processors from 2 to 32, the PDD algorithm reaches a linear speedup on memory-bounded speedup. The measured speedup is below ideal speedup because there is no communication in uniprocessor processing. In accordance with the isospeed metric [19], the PDD algorithm is perfectly scalable in the multi-stage SP2 machine from ensemble size 2 to 32.

Though the PDD and the Reduced PDD have similar relative speedup patterns, the execution times of the two algorithms are very different. The Reduced PDD algorithm has a smaller execution time than that of the PDD algorithm. For periodic systems, as the sample matrix, the Reduced PDD algorithm even has a smaller execution time than the conventional sequential algorithm. The timing of Thomas algorithm, the PDD algorithm, and the Reduced PDD algorithm on single node of the SP2 and Paragon machine are listed in Table 4. The problem size for all algorithms on SP2 is $n = 6400$ and $n1 = 1024$, on Paragon is $n = 1600$ and $n1 = 1024$. The measured results confirm the analytical results given in Table 1 and 2.

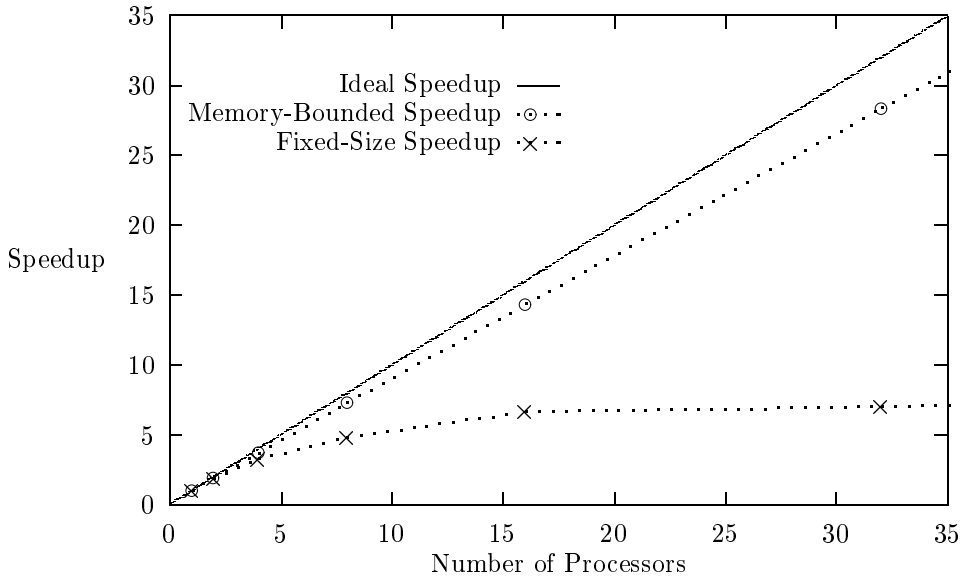

Figure 4. Measured Speedup of the PDD Algorithm on Intel Paragon
1024 System of Order 1600, periodic

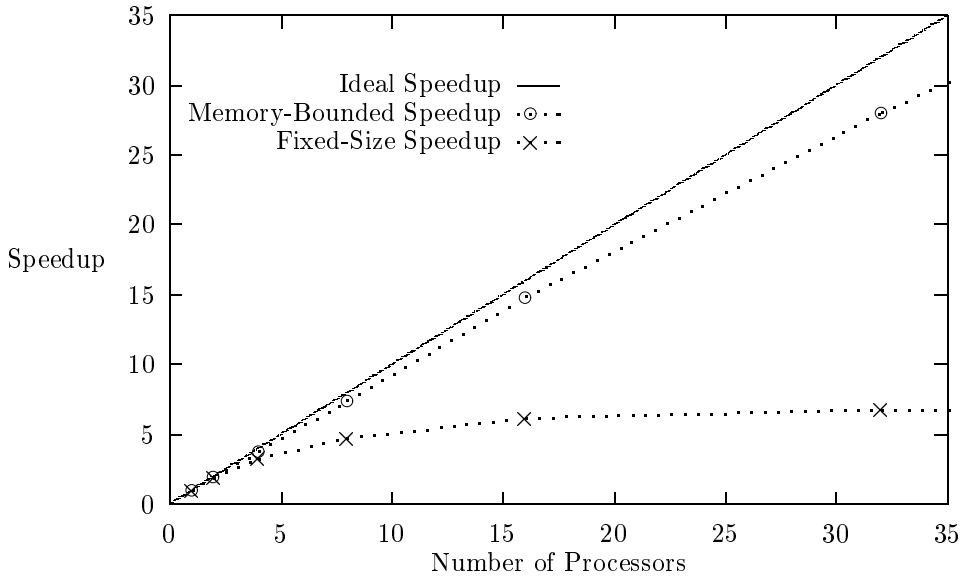

Figure 5. Measured Speedup of the Reduced PDD Algorithm on Intel Paragon
1024 System of Order 1600, periodic

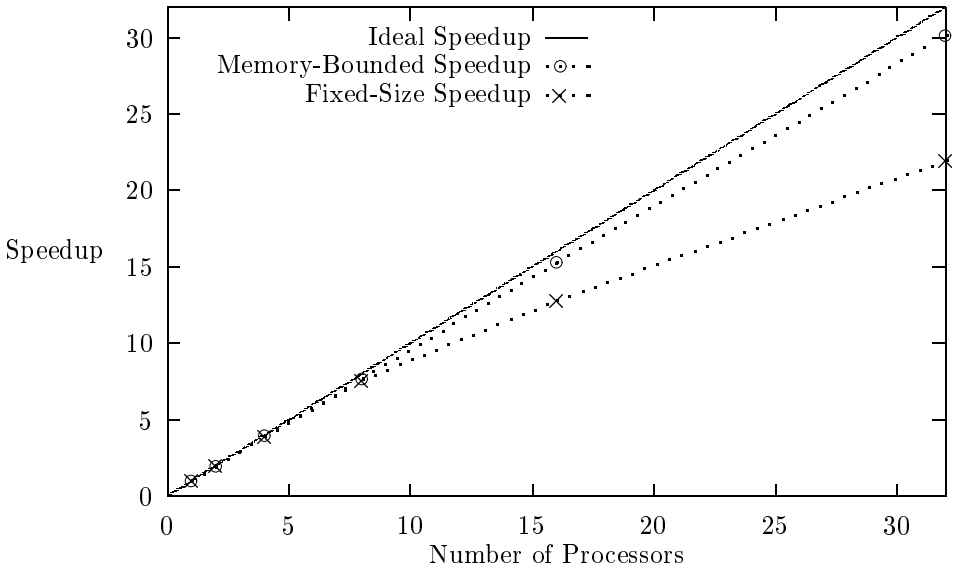

Figure 6. Measured Speedup of the PDD Algorithm on a SP2 Machine
1024 System of Order 6400, periodic

	Size	Thomas Alg.	PDD Alg.	Reduced PDD Alg.
Paragon	1600	0.8265	0.9026	0.6432
SP2	6400	0.7387	0.856	0.5545

Table 4. Sequential Timing (in seconds) on Paragon and SP2 machines

Figures 7 and 8 show the speedup of the PDD and Reduced PDD algorithm over the conventional sequential algorithm, Thomas algorithm, respectively. The PDD algorithm increases computation count for high parallelism. The Reduced PDD reduces computation count by taking advantage of diagonal dominance. Compared to Thomas algorithm, while the absolute speedup of the PDD algorithm is worse than its relative speedup, the Reduced PDD algorithm has a better absolute speedup than its relative speedup. The Reduced PDD algorithm achieves a superlinear speedup over Thomas algorithm. Experimental results confirm that the Reduced PDD algorithm maintains the good scalability of the PDD algorithm and delivers an efficient performance in terms of execution time as well.

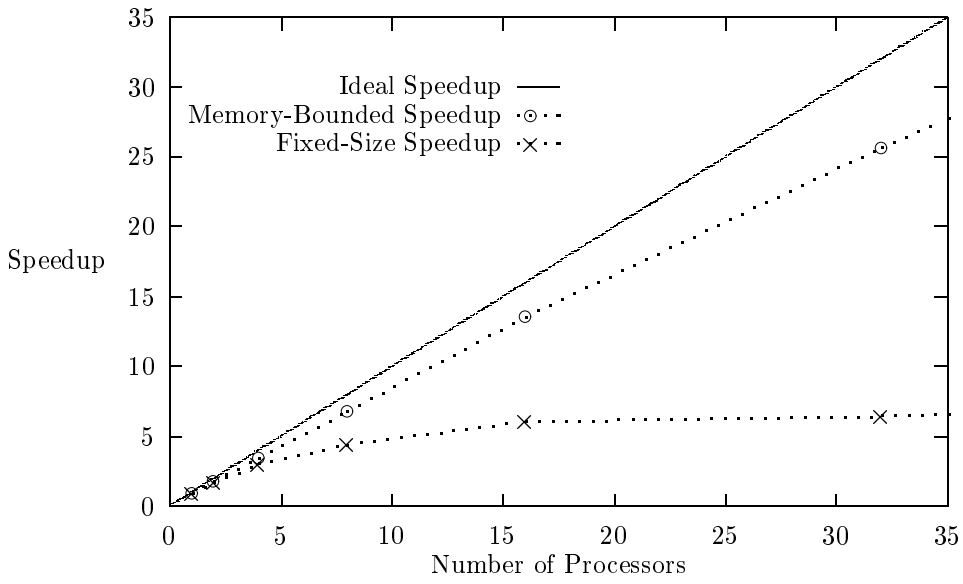

Figure 7. Speedup of the PDD Algorithm Over Thomas Algorithm.
1024 Systems of Order 1600, periodic

Non-periodic systems have also been tested on the Paragon and SP2 machines. As shown in Tables 1 and 2, both the PDD and Reduced PDD algorithm have the same parallel operation count for solving periodic and non-periodic systems. The only difference is that for periodic systems ring communication is required whereas for non-periodic systems 1-D array communication is required. Figure 9 depicts the memory-bounded speedup of the PDD algorithm for solving periodic and non-periodic systems on the Paragon machine. Observing the speedup curves, we can see that speedup is a little higher for non-periodic system than for periodic system. The difference in speedup is due to the nature of the architecture of the Paragon machine: 1-D array can be embedded onto 2-D mesh while ring cannot. The memory-bound speedup of the Reduced PDD algorithm on the SP2 machine is shown in Fig. 10. On a multi-stage Omega network, each processor has a equal access time to all the remote memories. The communication costs for ring communication and 1-D

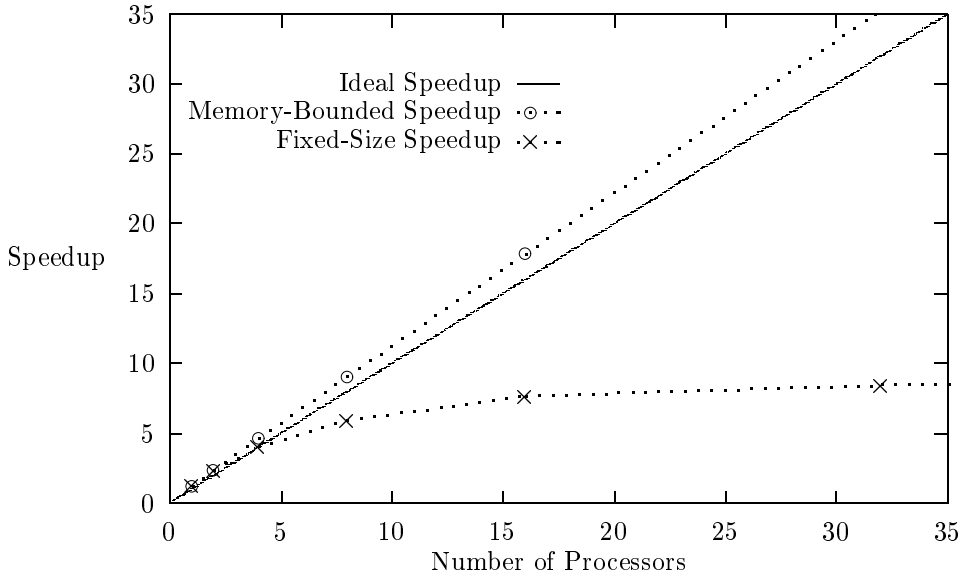

Figure 8. Speedup of the Reduced PDD Algorithm Over Thomas Algorithm.
1024 Systems of Order 1600, periodic

array communication are the same on Omega network. The speedup variation of the Reduced PDD algorithm for solving periodic and non-periodic systems on the SP2 machine is negligible.

The PDD and the Reduced PDD algorithms are perfectly scalable, in the sense that their communication cost does not increase with the order of matrix and ensemble size, and the workload is balanced. The PPT algorithm, however, has a serial processing part and a communication cost which increase with the ensemble size. While the PDD and the Reduced PDD algorithms have similar speedup curves on both the Paragon and the SP2 machines, the PPT has quite different speedup curves on the Paragon and the SP2 machines. Figure 11 shows the scaled and the fixed-size speedup of the PPT algorithm on the SP2 machine. The measured speedup is considerably lower than that of the PDD and the Reduced PDD algorithm. Parallel efficiency is usually defined as speedup divided by the number of processors. Unlike the PDD and the Reduced PDD algorithm, the efficiency of the PPT algorithm decays with the ensemble size. The scaled speedup of the PPT algorithm on the two machines are presented in Fig. 12. From Figs. 11 and 12 we can see that the PPT algorithm cannot reach linear speedup on either machine. However, its speedup on SP2 is much higher than on Paragon. The higher speedup is due to the fact that the SP2 has a larger memory, and therefore a better parallel/serial processing ratio[2]. The higher speedup can also be attributed to the difference of communication complexity of the total-data-exchange communication on a 2-D mesh and on a Omega network topology. The experimental results show

[2]Notice that the operation of the serial portion of the PPT algorithm does not increase with the order of matrix. When the number of RHS is fixed, it only increases with the number of processors

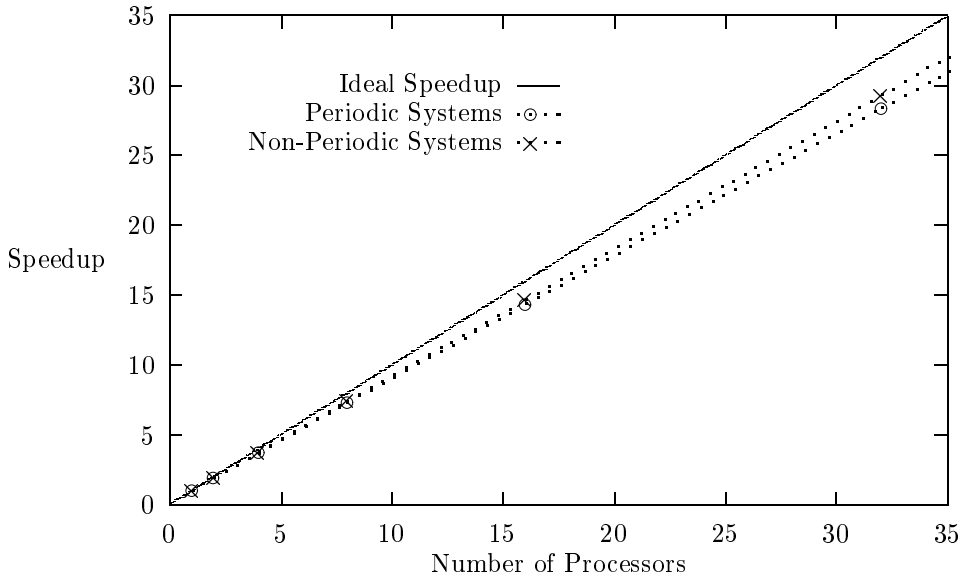

Figure 9. Scaled Speedup of the PDD Algorithm on Paragon.
1024 Systems of Order 1600, periodic & non-periodic

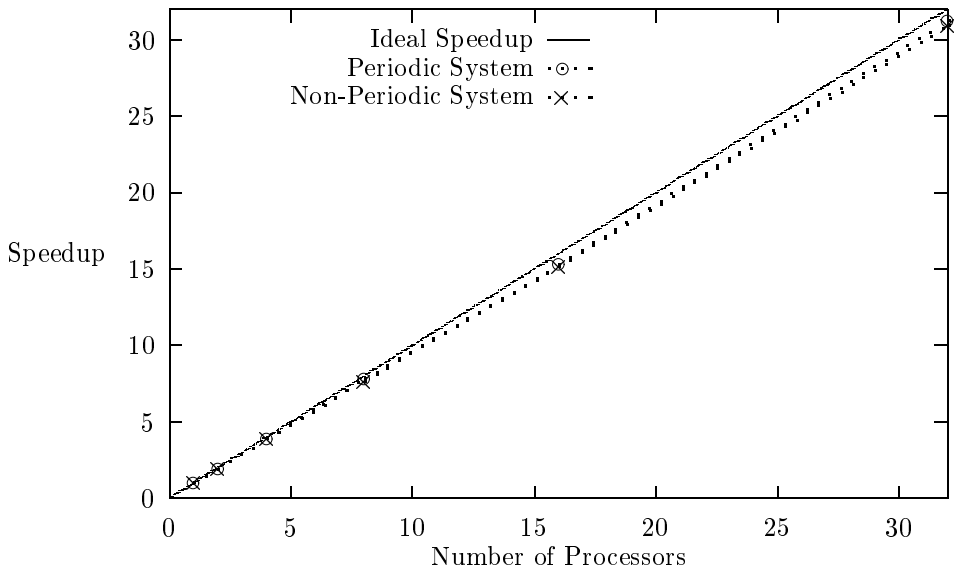

Figure 10. Scaled Speedup of the Reduced PDD Algorithm on SP2.
1024 Systems of Order 6400, periodic & non-periodic

that applications with complicated computation and communication structures are more sensitive to hardware support.

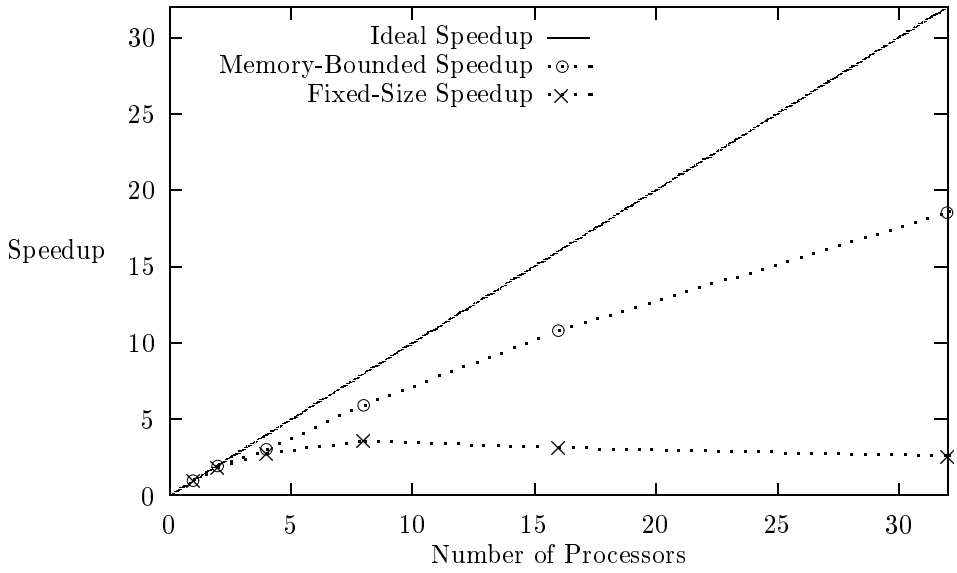

Figure 11. Speedup of the PPT algorithm on SP2 Machine.
1024 Systems of Order 6400, non-periodic

5 Conclusion

Parallel computers offer significantly increased computing power for solving scientific applications. However, utilizing the high computing power in solving actual applications is difficult. Efficient parallel algorithms have to be designed to maximize the parallelism and to minimize the communication. Communication cost is strictly related to the underlying architecture as well as the algorithm. Various algorithms have been proposed on various architectures in recent years. The choice of an algorithm/architecture pair may exhibit a wide range of variations in performance for a given application. In addition, implementation technique and hardware details, that may not be considered in theoretical analysis, may influence the final performance considerably. It is very important that parallel algorithms are compared not only in terms of operation count but also taking implementation details and results into account, especially for distributed-memory machines where communication cost is high.

Tridiagonal systems arise in many scientific applications. They are usually "kernels" in larger codes. Three parallel tridiagonal solvers, the PDD, the Reduced PDD, and the PPT algorithms are studied in detail in this paper. Comparisons of these three algorithms, in terms of best sequential algorithms, in terms of execution time, and in terms of speedup are presented. Experimental mea-

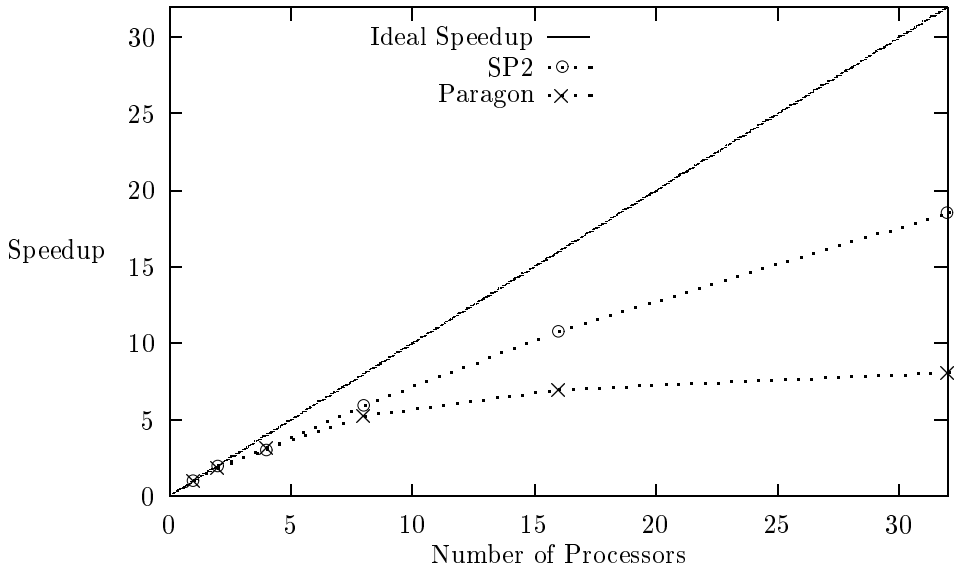

Figure 12. Memory-Bounded Speedup of the PPT algorithm.
1024 Systems, non-periodic

surement and performance evaluations have been conducted on two distributed-memory platforms: Intel Paragon and IBM SP2. Algorithms for both periodic and non-periodic systems are tested. In addition to theoretical analysis, implementation considerations are also discussed. The PPT algorithm is a general tridiagonal solver. It has a serial processing part and requires a all-to-all communication. Implementation comparison on a Paragon and SP2 machine shows that the performance of the PPT algorithm is very sensitive to hardware support and to problem size. The sensitivity probably is true for any algorithm with complicated computation and communication structures. Unlike the PPT algorithm, the PDD and the Reduced PDD algorithms, which have local communication and load balance, reach similar speedup curves on the Paragon and the SP2 machine. For both the PDD and Reduced PDD algorithms the non-periodic systems yield a little better performance than the periodic systems on the Paragon.

The PDD and the Reduced PDD algorithm are designed for diagonally dominant tridiagonal systems. Experimental and theoretical results show that both the PDD and Reduced PDD algorithm are efficient and scalable, even for systems with multiple right-hand-sides. For periodic systems, as confirmed by our implementation results, the Reduced PDD algorithm even has a smaller sequential execution time than that of the best sequential algorithm, when it is applicable. The two algorithms are good candidates for parallel computers. The common merit of these two algorithms is the minimum communication required. This merit makes them even more valuable in a distributed computing environment, such as the environment of a cluster of a network of workstations.

References

[1] Committee on Physical and Mathematical and Engineering Sciences, "Grand challenges: High performance computing and communications," National Science Foundation, 1992.

[2] X.-H. Sun, "Application and accuracy of the parallel diagonal dominant algorithm," *Parallel Computing*, pp. 1241–1267, Aug. 1995.

[3] X.-H. Sun, H. Zhang, and L. Ni, "Efficient tridiagonal solvers on multicomputers," *IEEE Transactions on Computers*, vol. 41, no. 3, pp. 286–296, 1992.

[4] C. Ho and S. Johnsson, "Optimizing tridiagonal solvers for alternating direction methods on boolean cube multiprocessors," *SIAM J. of Sci. and Stat. Computing*, vol. 11, no. 3, pp. 563–592, 1990.

[5] J. Lambiotte and R. Voigt, "The solution of tridiagonal linear systems on the CDC Star-100 computer," *ACM Trans. Math. Soft.*, vol. 1, pp. 308–329, Dec. 1975.

[6] D. Lawrie and A. Sameh, "The computation and communication complexity of a parallel banded system solver," *ACM Trans. Math. Soft.*, vol. 10, pp. 185–195, June 1984.

[7] H. Wang, "A parallel method for tridiagonal equations," *ACM Trans. Math. Software*, vol. 7, pp. 170–183, June 1981.

[8] J. Sherman and W. Morrison, "Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix," *Ann. Math. Stat.*, vol. 20, no. 621, 1949.

[9] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability.* McGraw-Hill, 1993.

[10] J. Ortega and R. Voigt, "Solution of partial differential equations on vector and parallel computers," *SIAM Review*, pp. 149–240, June 1985.

[11] C. Hirsch, *Numerical Computation of Internal and External Flows.* John Wiley & Sons, 1988.

[12] V. Kumar and et al., *Introduction to Parallel Computing: Design and Analysis of Algorithms.* Benjamin/Commings, 1994.

[13] V. Bala and et al., "Ccl: A portable and tunable collective communication library for scalable parallel computers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, pp. 154–164, Feb. 1995.

[14] M. A. Malcolm and J. Palmer, "A fast method for solving a class of tridiagonal linear systems," *Communications of the ACM*, vol. 17, no. 1, pp. 14–17, 1974.

[15] X.-H. Sun and J. Zhu, "Performance considerations of shared virtual memory machines," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1185–1194, Nov. 1995.

[16] G. Amdahl, "Validity of the single-processor approach to achieving large scale computing capabilities," in *Proc. AFIPS Conf.*, pp. 483–485, 1967.

[17] J. Gustafson, "Reevaluating Amdahl's law," *Communications of the ACM*, vol. 31, pp. 532–533, May 1988.

[18] X.-H. Sun and L. Ni, "Scalable problems and memory-bounded speedup," *J. of Parallel and Distributed Computing*, vol. 19, pp. 27–37, Sept. 1993.

[19] X.-H. Sun and D. Rover, "Scalability of parallel algorithm-machine combinations," *IEEE Transactions on Parallel and Distributed Systems*, pp. 599–613, June 1994.