Theoretical Computer Science 289 (2002) 313–334
Theoretical Computer Science
www.elsevier.com/locate/tcs

Load-balanced parallel banded-system solvers

Kuo-Liang Chung[a,*,1], Wen-Ming Yan[b,2], Jung-Gen Wu[c,3]

[a] Department of Information Management, Institute of Computer Science & Information Engineering, National Taiwan University of Science and Technology No. 43, Section 4, Keelung Road, Taipei 10672, Taiwan ROC
[b] Department of Computer Science and Information Engineering, National Taiwan University, Taipei 100, Taiwan ROC
[c] Department of Information and Computer Education, National Taiwan Normal University, No. 162, Section 1, Hoping E. Road, Taipei 10610, Taiwan ROC

Received June 1999; received in revised form January 2000; accepted May 2001
Communicated by M. Nivat

Abstract

Solving banded systems is important in the applications of science and engineering. This paper presents a load-balancing strategy for solving banded systems in parallel when the number of processors used is small. An optimization-based load-balancing analysis is given to determine how many loads should be assigned to each processor in order to minimize the time requirement. Some experimentations are carried out on the nCUBE 2E multiprocessor to demonstrate the speedup advantage of the proposed load-balancing strategy. The speedup improvement ratio ranges from 47% to 66% (from 12% to 24%) when using 4 (8) processors. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: Banded systems; Load-balancing analysis; nCUBE 2E multiprocessor; Parallel algorithms

1. Introduction

Consider to solve an $n \times n$ banded system

$$A\mathbf{x} = \mathbf{b}, \tag{1}$$

* Corresponding author. Tel.: +886-2-27376771; fax: +886-2-27376777.
E-mail addresses: klchung@cs.ntust.edu.tw (K.-L. Chung), ganboon@csie.ntu.edu.tw (W.-M. Yan), jgwu@ice.ntnu.edu.tw (J.-G. Wu).
[1] Supported by the National Science Council of R.O.C. under contract NSC88-2213-E011-005.
[2] Supported by the National Science Council of R.O.C. under contract NSC87-2119-M002-006.
[3] Supported by the National Science Council of R.O.C. under contract NSC85-2213-E003-003.

PII: S0304-3975(01)00284-5

where $A=(a_{i,j})$ is a banded matrix with r lower nonzero diagonals and s upper nonzero diagonals such that $a_{i,j}=0$ for $j>i+s$ or $i>j+r$. The matrix A is called a banded matrix with bandwidth $r+s+1$. Solving banded systems is important in the applications of science and engineering [4, 8]. Solving Eq. (1) sequentially is time consuming when n is large enough. Parallel processing [10] is a very natural approach to speed up the concerning large amount of computations.

When setting $r=s=m$, Lawrie and Sameh [11] presented an $O(m^2n/p)$-time parallel algorithm for solving a banded positive definite linear system, where p denotes the number of processors used in the multiprocessor. On ensemble architectures such as linear array, Boolean cube, etc., Johnsson [9] presented some time-optimal parallel algorithms for solving Eq. (1). Both results in [11, 9] are algorithmic. The other two efficient parallel methods were presented by Dongarra et al. [6, 7].

Consider a practical situation when n is large enough, but the number of processors used in the multiprocessor is small. That is, p is a small constant, e.g. $p=4$ or 8; the matrix A is of size 2048×2048 or 4096×4096. In this computation-bound case, the communication cost is small and does not dominate the total cost when compared to the computation cost. On the contrary, the computation cost dominates the total cost. For this situation, in the previous results [9, 11], the data of matrix A and vector $\mathbf{b}$ are evenly assigned to each processor in the multiprocessor. The motivation of this research is to determine how many data should be assigned to each processor in order to minimize the time bound requirement in this real computation-bound case.

This paper presents a load-balancing strategy for solving banded systems in parallel when the number of processors used is small. Based on some functional optimization techniques, a nontrivial load-balancing analysis is given to determine how many loads, i.e. data, should be assigned to each processor in order to minimize the time bound requirement. Specifically, we first transfer the load-balancing problem into a minimax problem, then a reduction technique is presented to narrow the solution space such that at least two feasible points but at most eight feasible points are needed to be tested in order to find the optimal solution. To the best of our knowledge, this is the first time that such a tight load-balancing result is derived. Some experimentations are carried out on the nCUBE 2E multiprocessor [13, 14] to demonstrate the speedup advantage of the proposed load-balancing strategy for different r, s, n, and p. Using 4 (8) processors, the speedup improvement ratio ranges from 47% to 66% (from 12% to 24%) when compared to the parallel banded-system solver without load-balancing consideration.

The rest of this paper is organized as follows. Section 2 presents a widely used partition strategy [11, 6, 9, 7] which will be used for solving Eq. (1) in parallel. Section 3 presents a related three-phase parallel algorithm. In addition, the detailed time complexity analysis is also given in the same section. A nontrivial load-balancing analysis is given in Section 4. Some experimental results are illustrated in Section 5. Some concluding remarks are addressed in Section 6.

2. Partition strategy

Suppose we are given p processors. In the assumption of this research, p is assumed to be a small integer, i.e. $n \gg p$. For convenience, we also assume $n \gg p(s+1)$ and $n \gg (r+1)p$. Based on rowwise partition scheme, A, $\mathbf{b}$ and $\mathbf{x}$ are partitioned into

$$A = \begin{pmatrix} A_1 & C_1 & & & & & \\ B_2 & A_2 & C_2 & & & & \\ & B_3 & A_3 & C_3 & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & \cdot & \\ & & & & B_{p-1} & A_{p-1} & C_{p-1} \\ & & & & & B_p & A_p \end{pmatrix},$$

$$\mathbf{b} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_p \end{pmatrix}$$

and

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_p \end{pmatrix},$$

where each A_i is an $n_i \times n_i$ banded matrix with r lower nonzero diagonals and s upper nonzero diagonals;

$$B_i = \begin{pmatrix} 0 & D_i \\ 0 & 0 \end{pmatrix} \quad \text{and } D_i \text{ is an } r \times r \text{ upper triangular matrix,}$$

$$C_i = \begin{pmatrix} 0 & 0 \\ E_i & 0 \end{pmatrix} \quad \text{and } E_i \text{ is an } s \times s \text{ lower triangular matrix.}$$

Thus, we can rewrite $A\mathbf{x} = \mathbf{b}$ as

$$A_1\mathbf{x}_1 + C_1\mathbf{x}_2 = \mathbf{b}_1,$$

$$B_i\mathbf{x}_{i-1} + A_i\mathbf{x}_i + C_i\mathbf{x}_{i+1} = \mathbf{b}_i \quad \text{for } i = 2, 3, \ldots, p-1$$

and

$$B_p\mathbf{x}_{p-1} + A_p\mathbf{x}_p = \mathbf{b}_p.$$

Multiplying A_i^{-1} to both sides of the ith equation for $1 \leqslant i \leqslant p$, yields

$$\mathbf{x}_1 + A_1^{-1}C_1\mathbf{x}_2 = A_1^{-1}\mathbf{b}_1,$$

$$A_i^{-1}B_i\mathbf{x}_{i-1} + \mathbf{x}_i + A_i^{-1}C_i\mathbf{x}_{i+1} = A_i^{-1}\mathbf{b}_i \quad \text{for } i = 2, 3, \ldots, p-1$$

and

$$A_p^{-1}B_p\mathbf{x}_{p-1} + \mathbf{x}_p = A_p^{-1}\mathbf{b}_p.$$

It is easy to know that

$$A_i^{-1}B_i = A_i^{-1}\begin{pmatrix} 0 & D_i \\ 0 & 0 \end{pmatrix} = \left(0 \;\; A_i^{-1}\begin{pmatrix} D_i \\ 0 \end{pmatrix}\right) = \begin{pmatrix} 0 & F_i \end{pmatrix}$$

and

$$A_i^{-1}C_i = A_i^{-1}\begin{pmatrix} 0 & 0 \\ E_i & 0 \end{pmatrix} = \left(A_i^{-1}\begin{pmatrix} 0 \\ E_i \end{pmatrix} \;\; 0\right) = \begin{pmatrix} G_i & 0 \end{pmatrix},$$

where

$$F_i = A_i^{-1}\begin{pmatrix} D_i \\ 0 \end{pmatrix} \quad \text{and} \quad G_i = A_i^{-1}\begin{pmatrix} 0 \\ E_i \end{pmatrix}. \tag{2}$$

Let

$$\mathbf{y}_i = A_i^{-1}\mathbf{b}_i, \tag{3}$$

then we have

$$\mathbf{x}_1 + (G_1 \;\; 0) \quad \mathbf{x}_2 = \mathbf{y}_1,$$

$$(0 \quad F_i)\mathbf{x}_{i-1} + \mathbf{x}_i + (G_i \quad 0)\mathbf{x}_{i+1} = \mathbf{y}_i \quad \text{for } i = 2, 3, \ldots, p-1$$

and

$$(0 \quad F_p)\mathbf{x}_{p-1} + \mathbf{x}_p = \mathbf{y}_p.$$

For any matrix or vector X, let $\bar{X}$ be the one by deleting all the rows of X except the first s rows and let $\underline{X}$ be the one by deleting all the rows of X except the last r rows. Therefore, we have

$$\mathbf{x}_1 + G_1\bar{\mathbf{x}}_2 = \mathbf{y}_1, \tag{4}$$

$$F_i\underline{\mathbf{x}}_{i-1} + \mathbf{x}_i + G_i\bar{\mathbf{x}}_{i+1} = \mathbf{y}_i \quad \text{for } i = 2, 3, \ldots, p-1$$

and

$$F_p\underline{\mathbf{x}}_{p-1} + \mathbf{x}_p = \mathbf{y}_p.$$

Before solving Eq. (4), we first solve the following equations:

$$\underline{G}_1\bar{\mathbf{x}}_2 + \underline{\mathbf{x}}_1 = \underline{\mathbf{y}}_1,$$

$$\bar{\mathbf{x}}_i + \bar{F}_i\underline{\mathbf{x}}_{i-1} + \bar{G}_i\bar{\mathbf{x}}_{i+1} = \bar{\mathbf{y}}_i,$$

$$\underline{F}_i\underline{\mathbf{x}}_{i-1} + \underline{G}_i\bar{\mathbf{x}}_{i+1} + \underline{\mathbf{x}}_i = \underline{\mathbf{y}}_i \quad \text{for } i = 2,3,\ldots,p-1$$

and

$$\bar{\mathbf{x}}_p + \bar{F}_p\underline{\mathbf{x}}_{p-1} = \bar{\mathbf{y}}_p.$$

In terms of matrix form, that is, we first solve the following linear system:

$$\begin{pmatrix}
\underline{G}_1 & I & & & & & & & & \\
I & \bar{F}_2 & \bar{G}_2 & & & & & & & \\
 & \underline{F}_2 & \underline{G}_2 & I & & & & & & \\
 & & I & \bar{F}_3 & \bar{G}_3 & & & & & \\
 & & & \underline{F}_3 & \underline{G}_3 & I & & & & \\
 & & & & \cdot & \cdot & \cdot & & & \\
 & & & & & I & \bar{F}_{p-1} & \bar{G}_{p-1} & & \\
 & & & & & & \underline{F}_{p-1} & \underline{G}_{p-1} & I & \\
 & & & & & & & I & \bar{F}_p &
\end{pmatrix}
\begin{pmatrix} \bar{\mathbf{x}}_2 \\ \underline{\mathbf{x}}_1 \\ \bar{\mathbf{x}}_3 \\ \underline{\mathbf{x}}_2 \\ \vdots \\ \underline{\mathbf{x}}_{p-2} \\ \bar{\mathbf{x}}_p \\ \underline{\mathbf{x}}_{p-1} \end{pmatrix}
=
\begin{pmatrix} \underline{\mathbf{y}}_1 \\ \bar{\mathbf{y}}_2 \\ \underline{\mathbf{y}}_2 \\ \bar{\mathbf{y}}_3 \\ \vdots \\ \bar{\mathbf{y}}_{p-1} \\ \underline{\mathbf{y}}_{p-1} \\ \bar{\mathbf{y}}_p \end{pmatrix}. \quad (5)$$

The solutions of Eq. (5) will be used to solve Eq. (4).

Based on the partition strategy mentioned in this section, in next section we will present a three-phase parallel solver for solving Eq. (1) and the corresponding detailed time complexity analysis.

3. The three-phase method and time complexity analysis

For convenience, we first analyze the time complexity required in solving Eq. (1) sequentially based on the LU-decomposition, i.e. $A = LU$, where L is an unitary lower matrix with r lower nonzero diagonals and U is an upper matrix with s upper nonzero diagonals. The related time complexity will be adopted in the time analysis of the three-phase method. From [8], we have

$$L = (l_{i,j}) \text{ where } l_{i,i} = 1 \text{ and } l_{i,j} = 0 \text{ for } i < j \text{ or } i > j + r,$$

$$U = (u_{i,j}) \text{ where } U_{i,i} = 1 \text{ and } u_{i,j} = 0 \text{ for } j < i \text{ or } j > i + s.$$

Basically, solving Eq. (1) sequentially consists of three steps, namely, the LU-decomposition, solving $L\mathbf{y} = \mathbf{b}$, and solving $U\mathbf{x} = \mathbf{y}$. Here, the time complexity is measured by the number of floating-point operations (FLOPs) and the time required in one subtraction is assumed to be equal to that in one addition.

Lemma 1. *The LU-decomposition of A needs n divisions, the number of multiplications required in this algorithm is between* $[n - \frac{1}{2}(r + s + 1)]r(s + 1)$ *and*

$[n-\frac{1}{2}(r+1)]r(s+1)$, *and the number of additions required in this algorithm is between* $[n-\frac{1}{2}(r+s+1)]rs$ *and* $[n-\frac{1}{2}(r+1)]rs$. *Solving* $L\mathbf{y}=\mathbf{b}$ *takes* $nr-r(r+1)/2$ *multiplications and* $nr-r(r+1)/2$ *additions. Solving* $U\mathbf{x}=\mathbf{y}$ *takes* $n(s+1)-s(s+1)/2$ *multiplication and* $ns-s(s+1)/2$ *additions.*

Proof. See Appendix A. □

The three-phase parallel method for solving Eq. (1) is described in the following three subsections.

3.1. Phase 1

In phase 1, Processor i, $2\leqslant i\leqslant p-1$, wants to obtain the solutions of $\bar{\mathbf{y}}_i$ and $\underline{\mathbf{y}}_i$ by solving $A_i\mathbf{b}_i=\mathbf{y}_i$ (see Eq. (3)). Specifically, Processor 1 (p) wants to obtain only the solution of $\underline{\mathbf{y}}_1$ ($\bar{\mathbf{y}}_p$). Simultaneously, in the same phase, Processor i, $2\leqslant i\leqslant p-1$, also wants to obtain the matrix form of G_i and F_i. Processor 1 (p) wants to obtain only the matrix form of $\underline{G}_1$ ($\bar{F}_p$).

We next analyze the detailed time complexity required in each processor and come to a conclusion that in phase 1, the computation cost required in Processor i, $2\leqslant i\leqslant p-1$, is more heavy than that in Processor 1 or p. This unbalanced phenomenon, which also occurs in phase 3, is an important clue and it leads to this work.

For Processor 1, the procedure of this phase is listed below:

Step 1: Factor A_1 as L_1U_1
Step 2: /* Solve $\underline{\mathbf{y}}_1$ from $A_1\mathbf{y}_1=\mathbf{b}_1$ */
  Step 2.1: Solve $\mathbf{z}_1$ from $L_1\mathbf{z}_1=\mathbf{b}_1$
  Step 2.2: Solve $\underline{\mathbf{y}}_1$ from $U_1\mathbf{y}_1=\mathbf{z}_1$
Step 3. /* Solve $\underline{G}_1$ from $A_1G_1=\begin{pmatrix}0\\E_1\end{pmatrix}$ */
  Step 3.1: Solve R_1 from $L_1R_1=\begin{pmatrix}0\\E_1\end{pmatrix}$
  Step 3.2: Solve $\underline{G}_1$ from $U_1G_1=R_1$

Looking at the above pseudocodes, from Lemma 1, Step 1 needs n_1 divisions, $[n_1-\frac{1}{2}(r+s+1)]r(s+1)$ to $[n_1-\frac{1}{2}(r+1)]r(s+1)$ multiplications, and $[n_1-\frac{1}{2}(r+s+1)]rs$ to $[n_1-\frac{1}{2}(r+1)]rs$ additions. Step 2.1 needs $[n_1-\frac{1}{2}(r+1)]r$ multiplications and $[n_1-\frac{1}{2}(r+1)]r$ additions.

Since $\underline{\mathbf{y}}_1$ consists of the last r entries of $\mathbf{y}_1$, the number of additions required in Step 2.2 is less than $1+2+\cdots+(r-1)=\frac{1}{2}(r-1)r$ and the number of multiplications required in Step 2.2 is less than $1+2+\cdots+r=\frac{1}{2}r(r+1)$.

Since E_1 is an $s\times s$ lower triangular matrix, the number of additions and multiplications required in Step 3.1 is less than $s\underbrace{(r+r+\cdots+r)}_{s}=rs^2$.

Since $\underline{G}_1$ consists of the last r rows of G_1 and R_1 has s columns, the number of additions required in Step 3.2 is less than $s[1+2+\cdots+(r-1)]=\frac{1}{2}(r-1)rs$ and the number of multiplications required in Step 3.2 is less than $s[1+2+\cdots+r]=\frac{1}{2}r(r+1)s$.

Totally, in phase 1, Processor 1 needs n_1 divisions, $[n_1-\frac{1}{2}(r+s+1)](rs+r)+[n_1-\frac{1}{2}(r+1)](rs+r)$ to $[n_1-\frac{1}{2}(r+1)](rs+2r)+\frac{1}{2}r(r+1)+rs^2+\frac{1}{2}r(r+1)s$ multiplications, and $[n_1-\frac{1}{2}(r+s+1)]rs+[n_1-\frac{1}{2}(r+1)]r$ to $[n_1-\frac{1}{2}(r+1)](rs+r)+\frac{1}{2}r(r-1)+rs^2+\frac{1}{2}(r-1)rs$ additions.

For Processor i, $i=2,3,\ldots,p-1$, the procedure performed in phase 1 is shown below:

Step 1: Factor A_i as L_iU_i

Step 2: Solve $\mathbf{y}_i$ from $A_i\mathbf{y}_i=\mathbf{b}_i$

Step 3: Solve G_i from $A_iG_i=\begin{pmatrix}0\\E_i\end{pmatrix}$

Step 3.1: Solve R_i from $L_iR_i=\begin{pmatrix}0\\E_i\end{pmatrix}$

Step 3.2: Solve G_i from $U_iG_i=R_i$

Step 4: Solve F_i from $A_iF_i=\begin{pmatrix}D_i\\0\end{pmatrix}$

Step 4.1: Solve S_i from $L_iS_i=\begin{pmatrix}D_i\\0\end{pmatrix}$

Step 4.2: Solve F_i from $U_iF_i=S_i$

In the above procedure, from Lemma 1, Step 1 needs n_i divisions, $[n_i-\frac{1}{2}(r+s+1)]r(s+1)$ to $[n_i-\frac{1}{2}(r+1)]r(s+1)$ multiplications, and $[n_i-\frac{1}{2}(r+s+1)]rs$ to $[n_i-\frac{1}{2}(r+1)]rs$ additions. Step 2 needs $n_i(r+s+1)-\frac{1}{2}r(r+1)-\frac{1}{2}s(s+1)$ multiplications and $n_i(r+s)-\frac{1}{2}r(r+1)-\frac{1}{2}s(s+1)$ additions.

Since E_i is a $s\times s$ lower triangular matrix, the number of additions and multiplications required in Step 3.1 is less than $s\underbrace{(r+r+\cdots+r)}_{s}=rs^2$.

Since all the rows of R_i except the last s rows of R_i are equal to zero, and R_i has s columns, Step 3.2 needs $s[n_i-\frac{1}{2}(r+1)]r$ multiplications and $s[n_i-\frac{1}{2}(r+1)]r-n_is$ to $s[n_i-\frac{1}{2}(r+1)]r-[n_i-s]s$ additions.

Since D_i is an $r\times r$ upper triangular matrix, Step 4.1 needs $[n_i-\frac{1}{2}(r+1)]r^2$ multiplications and $[n_i-\frac{1}{2}(r+1)]r^2-n_ir$ to $[n_i-\frac{1}{2}(r+1)]r^2-[n_i-r]r$ additions.

Since S_i has r columns, Step 4.2 needs $r[n_i-\frac{1}{2}s](s+1)$ multiplications and $r[n_i-\frac{1}{2}(s+1)]s$ additions.

Totally, in this phase, Processor i, $2\leqslant i\leqslant p-1$, needs n_i divisions, m_{l} to m_{r} multiplications, and a_{l} to a_{r} additions, where

$$\begin{aligned}m_{\mathrm{l}}=&[n_i-\tfrac{1}{2}(r+s+1)]r(s+1)+n_i(r+s+1)-\tfrac{1}{2}r(r+1)-\tfrac{1}{2}s(s+1)\\&+s[n_i-\tfrac{1}{2}(r+1)]r+[n_i-\tfrac{1}{2}(r+1)]r^2+r[n_i-\tfrac{1}{2}s](s+1),\end{aligned}$$

$$m_{\rm r} = [n_i - \tfrac{1}{2}(r+1)]r(s+1) + n_i(r+s+1) - \tfrac{1}{2}r(r+1) - \tfrac{1}{2}s(s+1)$$
$$+ rs^2 + s[n_i - \tfrac{1}{2}(r+1)]r + [n_i - \tfrac{1}{2}(r+1)]r^2 + r[n_i - \tfrac{1}{2}s](s+1),$$

$$a_{\rm l} = [n_i - \tfrac{1}{2}(r+s+1)]rs + n_i(r+s) - \tfrac{1}{2}r(r+1) - \tfrac{1}{2}s(s+1) + s[n_i - \tfrac{1}{2}(r+1)]r$$
$$- n_i s + [n_i - \tfrac{1}{2}(r+1)]r^2 - n_i r + r[n_i - \tfrac{1}{2}(s+1)]s,$$

$$a_{\rm r} = [n_i - \tfrac{1}{2}(r+1)]rs + n_i(r+s) - \tfrac{1}{2}r(r+1) - \tfrac{1}{2}s(s+1)$$
$$+ rs^2 + s[n_i - \tfrac{1}{2}(r+1)]r - [n_i - s]s + r[n_i - \tfrac{1}{2}(s+1)]s.$$

For Processor p, the procedure performed in phase 1 is listed below:

Step 1: Factor A_p as U_pL_p
Step 2: Solve $\bar{\mathbf{y}}_p$ from $A_p\mathbf{y}_p = \mathbf{b}_p$
  Step 2.1: Solve $\mathbf{z}_p$ from $U_p\mathbf{z}_p = \mathbf{b}_p$
  Step 2.2: Solve $\bar{\mathbf{y}}_p$ from $L_p\mathbf{y}_p = \mathbf{z}_p$
Step 3: Solve $\underline{F}_p$ from $A_pF_p = \binom{D_p}{0}$
  Step 3.1: Solve S_p from $U_pS_p = \binom{D_p}{0}$
  Step 3.3: Solve $\bar{F}_p$ from $L_pF_p = S_p$

Step 1 needs n_p divisions, $[n_p - \frac{1}{2}(r+s+1)](r+1)s$ to $[n_p - \frac{1}{2}(s+1)](r+1)s$ multiplications, and $[n_p - \frac{1}{2}(r+s+1)]rs$ to $[n_p - \frac{1}{2}(s+1)]rs$ additions. Step 2.1 needs $[n_p - \frac{1}{2}(s+1)]s$ multiplications and $[n_ps - \frac{1}{2}(s+1)]s$ additions.

Since $\bar{\mathbf{y}}_p$ consists of the first s entries of $\mathbf{y}_p$, the number of additions required in Step 2.2 is less than $1+2+\cdots+(s-1) = \frac{1}{2}(s-1)s$ and the number of multiplications required in Step 2.2 is less than $1+2+\cdots+s = \frac{1}{2}s(s+1)$.

Since D_p is an $r \times r$ upper triangular matrix, the number of additions and multiplications required in Step 3.1 is less than $r\underbrace{(s+s+\cdots+s)}_{r} = r^2s$.

Since $\bar{F}_p$ consists of the first s rows of F_p and S_p has r columns, the number of additions required in Step 3.2 is less than $r[1+2+\cdots+(s-1)] = \frac{1}{2}rs(s-1)$ and the number of multiplications required in Step 3.2 is less than $r[1+2+\cdots+s] = \frac{1}{2}rs(s+1)$.

Totally, in phase 1, Processor p needs n_p divisions, $[n_p - \frac{1}{2}(r+s+1)](rs+s) + [n_p - \frac{1}{2}(s+1)]s$ to $[n_p - \frac{1}{2}(s+1)](rs+2s) + \frac{1}{2}s(s+1) + r^2s + \frac{1}{2}rs(s+1)$ multiplications, and $[n_p - \frac{1}{2}(r+s+1)]rs + [n_p - \frac{1}{2}(s+1)]s$ to $[n_p - \frac{1}{2}(s+1)](rs+s) + \frac{1}{2}s(s-1) + r^2s + \frac{1}{2}(s-1)rs$ additions.

3.2. Phase 2

In phase 2, Processor 2, Processor 3, ..., and Processor p first send their $\underline{F}_i$'s, $\bar{F}_i$'s, $\underline{G}_i$'s, $\bar{G}_i$'s, $\underline{\mathbf{y}}_i$'s, and $\bar{\mathbf{y}}_i$'s to Processor 1. Processor 1 collects these data to form a reduced linear system and solves Eq. (5) alone sequentially. Processor 1 performs the following procedure in this phase.

Step 1: Accumulate the data $\bar{\mathbf{y}}_i$, $\underline{\mathbf{y}}_i$, $\bar{G}_i$, $\underline{G}_i$, $\underline{F}_i$, and $\bar{F}_i$ from Processors $2,3,\ldots,$ and $p-1$; $\bar{\mathbf{y}}_p$ and $\underline{F}_p$ from Processor p.

Step 2: Solve the reduced system (see Eq. (5)).

Step 3. Distribute the data $\bar{\mathbf{x}}_{i+1}$ and $\underline{\mathbf{x}}_{i-1}$ to Processor i for $i=2,3,\ldots,p-1$; $\bar{\mathbf{x}}_p$ to processor p.

Setting $n=p\times(r+s)$ in Lemma 1, this phase needs less than $p(r+s)(rs+2r+s+1)$ multiplications and less than $p(r+s)(rs+r+s)$ additions in Step 2. Note that the communication cost required in Step 1 and that required in Step 3 is negligible since in our assumption the number of processors used in the multiprocessor is small. The communication cost required in the two steps, i.e. Steps 1 and 3, is a fixed value and does not dominate the total cost required in the parallel banded-system solver. Therefore, the communication cost factor is ignored since it does not affect our load-balancing analysis.

3.3. Phase 3

Phase 3 is an update phase. We want to solve all the $\mathbf{x}_i$'s for $1\leqslant i\leqslant p$ by using Eq. (4) and the temporary solutions $\underline{\mathbf{x}}_1$, $\bar{\mathbf{x}}_p$, $\underline{\mathbf{x}}_i$, and $\bar{\mathbf{x}}_i$ for $2\leqslant i\leqslant p-1$.

For Processor 1, from the first equation in Eq. (4), we have

$$\mathbf{x}_1+G_1\bar{\mathbf{x}}_2=\mathbf{y}_1.$$

Multiplying A_1 to both sides, yields

$$A_1\mathbf{x}_1+A_1G_1\bar{\mathbf{x}}_2=A_1\mathbf{y}_1=\mathbf{b}_1.$$

We then have

$$\begin{aligned}A_1\mathbf{x}_1&=\mathbf{b}_1-A_1G_1\bar{\mathbf{x}}_2\\&=\mathbf{b}_1-\begin{pmatrix}0\\E_1\end{pmatrix}\bar{\mathbf{x}}_2\\&=\mathbf{b}_1-\begin{pmatrix}0\\E_1\bar{\mathbf{x}}_2\end{pmatrix}.\end{aligned}$$

From the LU-decomposition of A_1, $A_1=L_1U_1$, let

$$L_1\mathbf{z}_1=\mathbf{b}_1.$$

In phase 3, Processor 1 first solves $\mathbf{v}$ from

$$L_1\mathbf{v}=\begin{pmatrix}0\\E_1\bar{\mathbf{x}}_2\end{pmatrix},$$

then solves $\mathbf{x}_1$ from $U_1\mathbf{x}_1=\mathbf{z}_1-\mathbf{v}$. In this phase, Processor 1 needs about $n_1(s+1)$ multiplications and n_1s additions.

For Processor i, $2\leqslant i\leqslant p-1$, from the second equation in Eq. (4), we have

$$F_i\underline{\mathbf{x}}_{i-1}+\mathbf{x}_i+G_i\bar{\mathbf{x}}_{i+1}=\mathbf{y}_i.$$

Processor i for $i=2,3,\ldots,p-1$ just performs

$$\mathbf{x}_i \leftarrow \mathbf{y}_i - F_{i-1}\underline{\mathbf{x}}_{i-1} - G_{i+1}\bar{\mathbf{x}}_{i+1}.$$

Processor i for $i=2,3,\ldots,p-1$ needs about $n_i(r+s)$ multiplications and $n_i(r+s)$ additions.

By the same argument as in Processor 1, for Processor p, from the last equation in Eq. (4), we have

$$F_p\underline{\mathbf{x}}_{p-1} + \mathbf{x}_p = \mathbf{y}_p.$$

We further have

$$A_p\mathbf{x}_p + A_pF_p\underline{\mathbf{x}}_{p-1} = A_p\mathbf{y}_p = \mathbf{b}_p.$$

That is, we have

$$\begin{aligned} A_p\mathbf{x}_p &= \mathbf{b}_p - A_pF_p\underline{\mathbf{x}}_{p.0-1} \\ &= \mathbf{b}_p - \begin{pmatrix} D_p \\ 0 \end{pmatrix}\underline{\mathbf{x}}_{p-1} \\ &= \mathbf{b} - \begin{pmatrix} D_p\underline{\mathbf{x}}_{p-1} \\ 0 \end{pmatrix}. \end{aligned}$$

Using the LU-decomposition of A_p, $A_p=U_pL_p$, let

$$U_p\mathbf{z}_p = \mathbf{b}_p.$$

Processor p first solves $\mathbf{v}$ from

$$U_p\mathbf{w} = \begin{pmatrix} D_p\underline{\mathbf{x}}_{p-1} \\ 0 \end{pmatrix},$$

then solves $\mathbf{x}_p$ from $L_p\mathbf{x}_p=\mathbf{z}_p-\mathbf{w}$. Processor p needs about $n_p(r+1)$ multiplications and n_pr additions.

After analyzing the number of FLOPs required in each processor in the three-phase method mentioned in this section, the load-balancing analysis is given in the next section.

4. Load-balancing analysis

For simplifying the load-balancing analysis, let the computation time for one addition be one time unit; the computation time for one division be a time unit; the computation time for one multiplication be b time unit.

From the time bound required in Phase 1 for each processor, we know the two facts: (1) each Processor i, $2\leqslant i\leqslant p-1$, has the same time requirement when $n_2=n_3=\cdots=n_{p-1}$, and (2) the terms without involving n_i in the time complexity expressions,

$1 \leqslant i \leqslant p$, can be ignored because of $n_i \gg r, s$. Therefore, we set $n_1 = m_1$, $n_2 = n_3 = \cdots = n_{p-1} = m$, and $n_p = m_2$. Then, Processor 1 needs m_1 divisions, $m_1(rs + 2r)$ multiplications, and $m_1(rs + r)$ additions. Processor i, for $2 \leqslant i \leqslant p - 1$, needs m divisions, $m(r^2 + 3rs + 3r + s + 1)$ multiplications, and $m(r^2 + 3rs)$ additions. Processor p needs m_p divisions, $m_p(rs + 2s)$ multiplications, and $m_p(rs + s)$ additions.

From the parallel processing viewpoint, the time bound required in phase 1 is given by

$$\max(m_1 c_1, mc_2, m_2 c_3)$$

time unit, where

$$c_1 = a + (rs + 2r)b + (rs + r),$$
$$c_2 = a + (r^2 + 3rs + 3r + s + 1)b + (r^2 + 3rs)$$

and

$$c_3 = a + (rs + 2s)b + (rs + s).$$

By the same argument, the time bound required in phase 3 is given by

$$\max(m_1 d_1, md_2, m_2 d_3)$$

time unit, where

$$d_1 = (s + 1)b + s,$$
$$d_2 = (r + s)b + (r + s)$$

and

$$d_3 = (r + 1)b + r.$$

We thus wish to minimize the average computation time $f(m_1, m, m_2)$ for each equation, where

$$f(m_1, m, m_2) = \frac{\max(m_1 c_1, mc_2, m_2 c_3) + \max(m_1 d_1, md_2, m_2 d_3)}{m_1 + (p - 2)m + m_2}.$$

Let $t_1 = m_1/m$ and $t_2 = m_2/m$, then we have

$$f(m_1, m, m_2) = F(t_1, t_2) = \frac{\max(t_1 c_1, c_2, t_2 c_3) + \max(d_1 t_1, d_2, d_3 t_2)}{t_1 + (p - 2) + t_2}.$$

In what follows, an optimization-based load-balancing analysis is given to determine how many loads should be assigned to each processor in order to minimize $F(t_1, t_2)$. On the other hand, we want to determine the values of t_1 and t_2 such that the value of $F(t_1, t_2)$ is minimal. Once such point (t_1, t_2) is obtained, the values $n_1, n_2, \ldots,$ and n_p can also be obtained.

Before obtaining (t_1, t_2) to minimize $F(t_1, t_2)$, we need the following two lemmas.

Lemma 2. *Suppose G is an open region and is included in* $\{(t_1,t_2)\colon t_1>0, t_2>0\}$. *Let* $g(t_1,t_2)=(at_1+bt_2+c)/(t_1+t_2+d)$ *for* $a,b,c,d\geqslant 0$ *be in D. Then either g is constant in this region or minimal value of g does not exist at any interior point in D.*

Proof. Differentiating g with respect to t_1 and t_2, respectively, we have

$$\frac{dg}{dt_1}=\frac{(t_1+t_2+d)a-(at_1+bt_2+c)}{(t_1+t_2+d)^2}=\frac{(a-b)t_2+ad-c}{(t_1+t_2+d)^2}$$

and

$$\frac{dg}{dt_2}=\frac{(t_1+t_2+d)b-(at_1+bt_2+c)}{(t_1+t_2+d)^2}=\frac{(b-a)t_1+bd-c}{(t_1+t_2+d)^2}.$$

Suppose $a\neq b$. Solving $dg/dt_1=0$ and $dg/dt_2=0$, we obtain $t_1=(bd-c)/(a-b)$ and $t_2=(c-ad)/(a-b)$. Unfortunately, since $t_1+t_2=bd-ada-b=-d$, (t_1,t_2) does not belong to the given region D, so minimal value of g does not exist at any interior point in D in this case. Considering another case, suppose $a=b$. If $c\neq bd$, the two equations $dg/dt_1=0$ and $dg/dt_2=0$ have no solution, so minimal value of g also does not exist at any interior point in D in this case. Considering the remaining case, suppose $a=b$ and $c=bd$. Then g is constant in this region. We complete the proof. □

Lemma 3. *Suppose I is an open interval which is included in* $\{t\colon t>0\}$. *Let* $g(t)=(at+b)/(ct+d)$, $a,b,c,d\geqslant 0$ *on I, then either g is constant in this interval or minimal value of g does not exist at any interior point in I.*

Proof. Since $g'(t)=((ct+d)a-(at+b)c)/(ct+d)^2=(ad-bc)/(ct+d)^2$, if $ad\neq bc$, then $g'(t)\neq 0$. It implies that minimal value of g does not exist at any interior point. If $ad=bc$, then g is constant in this interval. We complete the proof. □

For solving this minimization problem, the numerator of $F(t_1,t_2)$ can be handled as follows. Consider the first class which consists of the following nine disjoint open 2-D regions, say $G_1,G_2,\ldots,G_9$. Each region is formed by the intersection of some open half-plane.

$$G_1=\{(t_1,t_2)\colon\ c_1t_1>c_2, c_1t_1>c_3t_2, d_1t_1>d_2, d_1t_1>d_3t_2\},$$

$$G_2=\{(t_1,t_2)\colon\ c_2>c_1t_1, c_2>c_3t_2, d_1t_1>d_2, d_1t_1>d_3t_2\},$$

$$G_3=\{(t_1,t_2)\colon\ c_3t_2>c_1t_1, c_3t_2>c_2, d_1t_1>d_2, d_1t_1>d_3t_2\},$$

$$G_4=\{(t_1,t_2)\colon\ c_1t_1>c_2, c_1t_1>c_3t_2, d_2>d_1t_1, d_2>d_3t_2\},$$

$$G_5=\{(t_1,t_2)\colon\ c_2>c_1t_1, c_2>c_3t_2, d_2>d_1t_1, d_2>d_3t_2\},$$

$$G_6=\{(t_1,t_2)\colon\ c_3t_2>c_1t_1, c_3t_2>c_2, d_2>d_1t_1, d_2>d_3t_2\},$$

$$G_7=\{(t_1,t_2)\colon\ c_1t_1>c_2, c_1t_1>c_3t_2, d_3t_2>d_1t_1, d_3t_2>d_2\},$$

$$G_8 = \{(t_1,t_2)\colon\ c_2 > c_1t_1, c_2 > c_3t_2, d_3t_2 > d_1t_1, d_3t_2 > d_2\},$$

$$G_9 = \{(t_1,t_2)\colon\ c_3t_2 > c_1t_1, c_3t_2 > c_2, d_3t_2 > d_1t_1, d_3t_2 > d_2\}.$$

From G_1, suppose the point $(t_1,t_2)\in G_1$. Then we have $F(t_1,t_2)=(c_1+d_1)t_1/(t_1+(p-2)+t_2)$. In general, considering each G_i for $1\leqslant i\leqslant 9$ and $(t_1,t_2)\in G_i$, it yields to $F(t_1,t_2)=(\alpha_i t_1+\beta_i t_2+\gamma_i)/(t_1+(p-2)+t_2)$.

By Lemma 2, the minimal value of function $F(t_1,t_2)$ does not exist at any interior point in the region unless that function is constant. In fact, if the function is constant, both the interior point in one G_i and the boundary point of the same G_i make the function constant. Further, we consider the next class.

The second class consists of the following 12 disjoint open segments or open half-line, say $I_1, I_2, \ldots, I_{12}$.

$$I_1 = \left\{(t_1,t_2)\colon\ c_1t_1 = c_2 > c_3t_2, d_3t_2 > \max\left(\frac{c_2d_1}{c_1}, d_2\right)\right\},$$

$$I_2 = \left\{(t_1,t_2)\colon\ c_1t_1 = c_2 > c_3t_2, d_3t_2 < \max\left(\frac{c_2d_1}{c_1}, d_2\right)\right\},$$

$$I_3 = \left\{(t_1,t_2)\colon\ c_1t_1 = c_3t_2 > c_2, \max\left(\frac{d_1c_3}{c_1}, d_3\right)t_2 > d_2\right\},$$

$$I_4 = \left\{(t_1,t_2)\colon\ c_1t_1 = c_3t_2 > c_2, \max\left(\frac{d_1c_3}{c_1}, d_3\right)t_2 < d_2\right\},$$

$$I_5 = \left\{(t_1,t_2)\colon\ c_3t_2 = c_2 > c_1t_1, d_1t_1 > \max\left(d_2, \frac{c_2d_3}{c_3}\right)\right\},$$

$$I_6 = \left\{(t_1,t_2)\colon\ c_3t_2 = c_2 > c_1t_1, d_1t_1 < \max\left(d_2, \frac{c_2d_3}{c_3}\right)\right\},$$

$$I_7 = \left\{(t_1,t_2)\colon\ d_1t_1 = d_2 > d_3t_2, c_3t_2 > \max\left(\frac{c_1d_2}{d_1}, c_2\right)\right\},$$

$$I_8 = \left\{(t_1,t_2)\colon\ d_1t_1 = d_2 > d_3t_2, c_3t_2 < \max\left(\frac{c_1d_2}{d_1}, c_2\right)\right\},$$

$$I_9 = \left\{(t_1,t_2)\colon\ d_1t_1 = d_3t_2 > d_2, \max\left(\frac{c_1d_3}{d_1}c_3\right)t_2 > c_2\right\},$$

$$I_{10} = \left\{(t_1,t_2)\colon\ d_1t_1 = d_3t_2 > d_2, \max\left(\frac{c_1d_3}{d_1}c_3\right)t_2 < c_2\right\},$$

$$I_{11} = \left\{(t_1,t_2)\colon\ d_3t_2 = d_2 > d_1t_1, c_1t_1 > \max\left(c_2, \frac{d_2c_3}{d_3}\right)\right\},$$

$$I_{12} = \left\{(t_1, t_2)\colon\ d_3 t_2 = d_2 > d_1 t_1, c_1 t_1 < \max\left(c_2, \frac{d_2 c_3}{d_3}\right)\right\}.$$

From I_1, suppose the point $(t_1, t_2) \in I_1$. Then we have $F(t_1, t_2) = c_2/c_1 + (p-2) + t_2$. In general, considering each I_i for $1 \leqslant i \leqslant 12$ and $(t_1, t_2) \in I_i$, yields

$$F(t_1, t_2) = \frac{\alpha_i t_1 + \beta_i}{t_1 + (p-2) + \gamma_i}$$

or

$$F(t_1, t_2) = \frac{\alpha_i t_2 + \beta_i}{t_2 + (p-2) + \gamma_i},$$

where α_i, β_i, and $\gamma_i \geqslant 0$.

By Lemma 3, the minimal value of function $F(t_1, t_2)$ does not exist at any interior point in the open interval unless that function is constant. In fact, if the function is constant, both the interior point in one I_i and the boundary point of the same I_i make the function constant. Finally, we consider the class of boundary points on all I_i's.

The third class consists of the following eight points, say $P_1, P_2, \ldots, P_8$.

$$P_1 = \{(t_1, t_2)\colon\ c_1 t_1 = c_2 = c_3 t_2\},$$

$$P_2 = \{(t_1, t_2)\colon\ d_1 t_1 = d_2 = d_3 t_2\},$$

$$P_3 = \{(t_1, t_2)\colon\ c_1 t_1 = c_2 > c_3 t_2, d_2 = d_3 t_2 > d_1 t_1\},$$

$$P_4 = \{(t_1, t_2)\colon\ c_1 t_1 = c_2 > c_3 t_2, d_1 t_1 = d_3 t_2 > d_2\},$$

$$P_5 = \{(t_1, t_2)\colon\ c_1 t_1 = c_3 t_2 > c_2, d_2 = d_3 t_2 > d_1 t_1\},$$

$$P_6 = \{(t_1, t_2)\colon\ c_1 t_1 = c_3 t_2 > c_2, d_1 t_1 = d_2 > d_3 t_2\},$$

$$P_7 = \{(t_1, t_2)\colon\ c_3 t_2 = c_2 > c_1 t_1, d_1 t_1 = d_2 > d_3 t_2\}$$

and

$$P_8 = \{(t_1, t_2)\colon\ c_3 t_2 = c_2 > c_1 t_1, d_1 t_1 = d_3 t_2 > d_2\}.$$

From Lemmas 2 and 3, we have the following theorem.

Theorem 1. *There exists a point $(\bar{t}_1, \bar{t}_2)$ which belongs to some P_i for $1 \leqslant i \leqslant 8$ such that the function $F(t_1, t_2)$ has its minimal value at point $(\bar{t}_1, \bar{t}_2)$.*

Consequently, there are at most eight points and at least two points from $P_1, P_2, \ldots, P_8$ to be put into the function $F(t_1, t_2)$. We then select the minimal output value. That is, the load-balancing problem discussed in this paper is equal to computing $\mathrm{Min}\{F(t_1, t_2) \mid (t_1, t_2) \in \bigcup_{i=1}^{8} P_i\}$.

Table 1
Values of t_1 and t_2

	t_1	t_2
$r=s=5$	3.27	3.27
$r=4,\ s=6$	3.14	2.84
$r=6,\ s=4$	3.46	3.83
$r=3,\ s=7$	3.06	2.44
$r=7,\ s=3$	3.72	4.67

5. Experimental results

Given different r, s, p, and n, we have implemented the proposed load-balanced parallel banded system solver on the nCUBE 2/E [13, 14] with 1, 2, 4, and 8 processors. Here, the time units required in one division, one multiplication, and one addition are 15, 5, and 1, respectively. That is, $a=15$ and $b=5$. The performance when compared to that one without load-balancing consideration is also demonstrated.

In our implementation, the data sizes, n's, are 2048 and 4096. Five different combinations of r and s are given. Table 1 illustrates the optimal solutions (t_1,t_2)'s, which are found by the load balancing analysis described in Section 4, for different cases.

The time requirement and the speedup for all the cases are listed in Table 2. The performance improvement of using the proposed load-balancing strategy is also listed in Table 2, where $R_p=(S_p(\text{Load balance})-S_p(\text{No load balance})/S_p(\text{No load balance})$. Using 4 (8) processors, the speedup improvement ratio ranges from 47% to 66% (from 12% to 24%) when compared to the parallel banded-system solver without load-balancing consideration.

The number of equations, i.e. the load, assigned to each processor using the proposed load-balancing scheme for each case is listed in Table 3, where n_0, n_i, and n_{p-1} denote the number of equations assigned to Processor 1, Processor i for $2\leqslant,\dots,p-1$ and Processor p, respectively.

6. Conclusions

This paper presents a load-balancing strategy for solving banded systems in parallel. The proposed strategy can speed up the existing parallel solvers for large banded systems significantly when the number of processors used is small. The main contribution of this paper is to present a nontrivial but rather general load-balancing analysis to determine how many loads should be assigned to different processors in order to minimize the time bound required. Some experimentations are carried out on the nCUBE 2E multiprocessor. Using 4 (8) processors, the speedup improvement ratio ranges from 47% to 66% (from 12% to 24%) when compared to the parallel banded-system solver without load-balancing consideration.

Table 2
Execution time, speedup, and performance improvement

		Execution time(T_p)/speedup(S_p)					Performance improvement	
		Sequential	No load balance		Load balance			
	n	$p=1$	4	8	4	8	R_4(%)	R_8(%)
	2048	0.54	0.38	0.23	0.24	0.20	59	14
$r=5$		1	1.42	2.35	2.25	2.69		
$s=5$	4096	1.09	0.70	0.41	0.47	0.35	50	17
		1	1.56	2.63	2.33	3.07		
	2048	0.54	0.38	0.23	0.24	0.20	59	17
$r=4$		1	1.40	2.33	2.22	2.72		
$s=6$	4096	1.07	0.74	0.41	0.47	0.35	61	19
		1	1.43	2.60	2.30	3.10		
	2048	0.53	0.36	0.23	0.24	0.21	47	12
$r=6$		1	1.41	2.55	2.18	2.57		
$s=4$	4096	1.06	0.75	0.41	0.47	0.36	61	14
		1	1.41	2.55	2.27	2.92		
	2048	0.50	0.38	0.23	0.24	0.19	63	21
$r=3$		1	1.32	2.19	2.15	2.65		
$s=7$	4096	1.01	0.75	0.41	0.45	0.33	66	24
		1	1.35	2.45	2.24	3.04		
	2048	0.49	0.38	0.23	0.23	0.20	64	13
$r=7$		1	1.28	2.13	2.10	2.40		
$s=3$	4096	0.98	0.75	0.41	0.45	0.36	66	15
		1	1.31	2.37	2.17	2.72		

Table 3
Number of equations to be assigned to each processor

		$p=4$			$p=8$		
	n	n_0	n_i	n_{p-1}	n_0	n_i	n_{p-1}
$r=5$	2048	785	239	785	535	163	535
$s=5$	4096	1569	479	1569	1070	326	1070
$r=4$	2048	803	256	733	533	170	495
$s=6$	4096	1630	513	1460	1070	341	980
$r=6$	2048	761	220	847	532	154	592
$s=4$	4096	1522	440	1694	1065	308	1183
$r=3$	2048	835	273	667	544	178	436
$s=7$	4096	1670	546	1334	1089	356	871
$r=7$	2048	732	197	922	528	142	668
$s=3$	4096	1465	394	1843	1056	284	1336

In fact, the proposed load-balancing result can be applied to the design of load-balanced parallel solvers for solving the banded systems based on some other methods [2, 3], e.g. cyclic reduction method. The main reason of this applicability is that under the situation when the number of processors is small and the size of the system is large enough, the load in the first/last processor is less than that in any other processor, so the optimization-based load-balancing analysis discussed in this paper can be used to determine how many loads should be assigned to each processor in order to minimize the time bound requirement, i.e. $F(t_1, t_2)$. Since the optimization-based load-balancing analysis is rather general, the detailed load-balancing analysis could be derived if we know the related parameters in different parallel machine, e.g. IBM SP/2 supercomputer [3], and different implementation, e.g. ScaLAPACK [5].

Specifically, the spirit of this paper has been applied successfully to plug the load-balanced advantage into the parallel solvers for solving the tridiagonal systems [5, 12]. However, since the tridiagonal system is a special case of the banded system, a simpler load-balancing analysis has been derived in [16]. Experimental results show that the parallel tridiagonal system-solver proposed by Amodio and Mastronardi [1] has 43% (21%) speedup improvement when employing the proposed load-balancing consideration on the nCUBE 2E multiprocessor with 4 (8) processors.

Appendix A. Proof of Lemma 1

In this appendix, we first present the algorithm for solving Eq. (1) sequentially based on the LU-decomposition approach, that is, the banded matrix $A = LU$. Then we analyze the number of FLOPs required in the algorithm.

The formal algorithm for LU-decomposition of A is shown below. Following the notations used in [8], the resulting matrices L and U are obtained according to the order $U(1:1:s+1) = [u_{11}u_{12}\ldots u_{1(s+1)}]$, $L(2:1:2) = [l_{21}l_{22}]$, $U(2:2:s+2)$, $L(3:1:3)$, and so on.

Input: $A = [a_{ij}]_{n\times n}$ is a banded matrix with r lower diagonals and s upper diagonals.

Output: The resulting matrices L and U which are overlapped into $A = [a_{ij}]_{n\times n}$. $L = [l_{ij}]_{n\times n}$ and $U = [u_{ij}]_{n\times n}$ are obtained by setting $l_{ii} = 1$, $l_{ij} = a_{ij}$ for $j < i \leqslant j + r$, $l_{ij} = 0$ for $i < j$ or $i > j + r$, $u_{ii} = 1/a_{ii}$, $u_{ij} = a_{ij}$ for $i < j \leqslant i + s$, and $u_{ij} = 0$ for $j < i$ or $j > i + s$.

for $i \leftarrow 1$ to n
  for $j \leftarrow \max(1, i - r)$ to $i - 1$
    $a_{ij} \leftarrow -a_{jj} * a_{ij}$
    for $k \leftarrow \max(1, j + 1)$ to $\min(n, j + s)$
      $a_{ik} \leftarrow a_{ik} + a_{ij} * a_{jk}$
    end for
  end for

$a_{ii} \leftarrow \dfrac{1}{a_{ii}}$

end for

In the ith iteration of the outer for-loop, it needs m_i multiplications, a_i additions, and one division, where m_i and a_i will be analyzed later. Temporarily, it is said that the above procedure needs $\sum_{i=1}^{n} m_i$ multiplications, $\sum_{i=1}^{n} a_i$ additions, and n divisions. For analyzing the number of FLOPs, let

$$m_i = \begin{cases} (i-1)(s+1) & \text{for } 1 \leqslant i \leqslant r, \\ r(s+1) & \text{for } r+1 \leqslant i \leqslant n-s+1, \end{cases}$$

$$m_{n-s+i+1} = m_{n-s+i} - i \quad \text{for } 1 \leqslant i \leqslant r,$$

$$m_{n-s+r+i+1} = m_{n-s+r+i} - r \quad \text{for } 1 \leqslant i < s-r \tag{A.1}$$

and

$$a_i = \begin{cases} (i-1)s & \text{for } 1 \leqslant i \leqslant r, \\ rs & \text{for } r+1 \leqslant i \leqslant n-s+1, \end{cases}$$

$$a_{n-s+i+1} = a_{n-s+i} - i \quad \text{for } 1 \leqslant i \leqslant r,$$

$$a_{n-s+r+i+1} = a_{n-s+r+i} - r \quad \text{for } 1 \leqslant i < s-r. \tag{A.2}$$

The number of multiplications required in the above algorithm is equal to $\sum_{i=1}^{n} m_i$. From the first and second equalities in (A.1), we have

$$\sum_{i=1}^{n} m_i = \sum_{i=1}^{n} r(s+1) - \sum_{i=1}^{n} [r(s+1) - m_i]$$

$$= nr(s+1) - \sum_{i=1}^{r} [r(s+1) - m_i] - \sum_{i=1}^{s} [r(s+1) - m_{n-s+i}]$$

$$= nr(s+1) - \sum_{i=1}^{r} (r-i+1)(s+1) - \sum_{i=1}^{s} [m_{n-s+1} - m_{n-s+i}]$$

$$= nr(s+1) - \frac{1}{2}r(r+1)(s+1) - \sum_{i=1}^{s} [m_{n-s+1} - m_{n-s+i}].$$

We now further complete the asymptotical analysis of $\sum_{i=1}^{s}[r(s+1)-m_{n-s+i}]$. From the third and fourth equalities in (A.1), we have $m_{n-s+1} + m_n \geqslant m_{n-s+1} = r(s+1)$, $m_{n-s+2} + m_{n-1} \geqslant m_{n-s+1} + m_n \geqslant r(s+1)$, $m_{n-s+3} + m_{n-2} \geqslant m_{n-s+2} + m_{n-1} \geqslant r(s+1)$, and so on. Hence we have

$$\sum_{i=1}^{s} m_{n-s+i} \geqslant \frac{1}{2} rs(s+1).$$

Since each $m_{n-s+i} \leqslant r(s+1)$ for $i=1,2,\ldots,s$, we have

$$\sum_{i=1}^{s} m_{n-s+i} \leqslant rs(s+1).$$

So, it yields

$$0 \leqslant \sum_{i=1}^{s} [r(s+1) - m_{n-s+i}] \leqslant \frac{1}{2} rs(s+1).$$

Finally, we have

$$\left[n - \frac{1}{2}(r+s+1)\right] r(s+1) \leqslant \sum_{i=1}^{n} m_i \leqslant \left[n - \frac{1}{2}(r+1)\right] r(s+1).$$

The number of multiplications required for the above LU decomposition algorithm is between $[n - \frac{1}{2}(r+s+1)]r(s+1)$ and $[n - \frac{1}{2}(r+1)]r(s+1)$.

In addition, the number of additions required is equal to $\sum_{i=1}^{n} a_i$. From the first and second equalities in (A.2), we have

$$\begin{aligned}
\sum_{i=1}^{n} a_i &= \sum_{i=1}^{n} rs - \sum_{i=1}^{n} (rs - a_i) \\
&= nrs - \sum_{i=1}^{r} (rs - a_i] - \sum_{i=1}^{s} (rs - a_{n-s+i}) \\
&= nrs - \sum_{i=1}^{r} (r - i + 1)s - \sum_{i=1}^{s} (rs - a_{n-s+i}) \\
&= nrs - \frac{1}{2} r(r+1)s - \sum_{i=1}^{s} (rs - a_{n-s+i}).
\end{aligned}$$

We now give the asymptotical analysis for $\sum_{i=1}^{s} [rs - a_{n-s+i}]$. From the third and fourth equalities in (A.2), we have $a_{n-s+1} + a_n \geqslant rs$, $a_{n-s+1} + a_{n-1} \geqslant a_{n-s+1} + a_n \geqslant rs$, $a_{n-s+2} + a_{n-2} \geqslant a_{n-s+1} + a_{n-1} \geqslant rs$, and so on. Hence, we have

$$\sum_{i=1}^{s} a_{n-s+i} \geqslant \frac{1}{2} rs^2.$$

Since each $a_{n-s+i} \leqslant rs$ for $i=1,2,\ldots,s$, we have

$$\sum_{i=1}^{s} a_{n-s+i} \leqslant rs^2.$$

So, it yields

$$0 \leqslant \sum_{i=1}^{s} [rs - a_{n-s+i}] \leqslant \frac{1}{2} rs^2.$$

Finally, we have

$$\left[n - \frac{1}{2}(r+s+1)\right] rs \leqslant \sum_{i=1}^{s} a_{n-s+i} \leqslant \left[n - \frac{1}{2}(r+1)\right] rs.$$

Consequently, the number of additions required in the LU decomposition algorithm is between $[n - \frac{1}{2}(r+s+1)]rs$ and $[n - \frac{1}{2}(r+1)]rs$.

After discussing the LU decomposition for $A = LU$, we further analyze the number of FLOPs required in solving $L\mathbf{y} = \mathbf{b}$ and $U\mathbf{x} = \mathbf{y}$. We can solve $L\mathbf{y} = \mathbf{b}$ using the forward substitution. The related procedure is shown below:

$y_1 \leftarrow b_1$
for $i \leftarrow 2$ to n
  $y_i \leftarrow b_i$
  for $j \leftarrow \max(1, i-r)$ to $i-1$
    $y_i \leftarrow y_i - l_{ij} * y_j$
  end for
end for

In the ith iteration of the outer for-loop, it needs c_i multiplications and c_i subtractions, where

$$c_i = \begin{cases} i-1 & \text{for } 2 \leqslant i \leqslant r, \\ r & \text{for } r+1 \leqslant i \leqslant n. \end{cases} \tag{A.3}$$

We have assumed that the time required for one subtraction is equal to that required for one addition. The number of additions (multiplications) in the above procedure is equal to

$$\begin{aligned} \sum_{i=2}^{n} c_i &= \sum_{i=2}^{n} r - \sum_{i=2}^{n} [r - c_i] \\ &= (n-1)r - \sum_{i=2}^{r} [r - i + 1] \\ &= nr - \sum_{i=1}^{r} [r - i + 1] \\ &= nr - \frac{r(r+1)}{2} \\ &= [n - \tfrac{1}{2}(r+1)]r. \end{aligned}$$

Afterward, we can solve $U\mathbf{x} = \mathbf{y}$ using forward substitution and the procedure is shown below:

$x_n \leftarrow y_n / u_{nn}$ /* performed by $y_n * a_{nn}$ */
for $i \leftarrow n-1$ downto 1

$x_i \leftarrow y_i$
for $j \leftarrow i+1$ to $\min(i+s, n)$
$x_i \leftarrow y_i - u_{ij} * x_j$
end for
$x_i \leftarrow x_i / u_{ii}$ /* performed by $x_i * a_{ii}$ */
end for

In the ith iteration of the outer for-loop, it needs $d_i + 1$ multiplications and d_i additions, where

$$d_i = \begin{cases} s & \text{for } 1 \leqslant i \leqslant n-s, \\ n-i & \text{for } n-s < i \leqslant n-1. \end{cases} \tag{A.4}$$

The number of additions required in the above algorithm is equal to

$$\begin{aligned} \sum_{i=1}^{n-1} d_i &= \sum_{i=1}^{n-1} s - \sum_{i=1}^{n-1} [s - d_i] \\ &= (n-1)s - \sum_{i=n-s+1}^{n-1} [s-n+i] \\ &= ns - \sum_{i=n-s+1}^{n} [s-n+i] \\ &= ns - \frac{s(s+1)}{2} \\ &= [n - \tfrac{1}{2}(s+1)]s. \end{aligned}$$

The number of multiplications required in the above algorithm is equal to

$$\begin{aligned} \sum_{i=1}^{n-1} (d_i + 1) &= ns - \frac{s(s+1)}{2} + n \\ &= [n - \tfrac{1}{2}s](s+1). \end{aligned}$$

In summary, solving $U\mathbf{x} = \mathbf{y}$ needs $[n - \frac{1}{2}(s+1)]s$ additions and $[n - \frac{1}{2}s](s+1)$ multiplications.

Acknowledgements

The authors are indebted to the anonymous reviewers, Prof. M. Nivat, and S. Smit for their valuable suggestions that lead to the improved version of the paper.

References

[1] P. Amodio, N. Mastronardi, A parallel version of the cyclic reduction algorithm on a hypercube, Parallel Comput. 19 (1993) 1273–1281.

[2] P. Arbenz, A. Cleary, J. Dongarra, M. Hegland, A comparison of parallel solvers for diagonally dominant and general narrow-anded linear systems, UT-CS-99-414, University of Tennessee, 1999, USA.

[3] P. Arbenz, A. Cleary, J. Dongarra, M. Hegland, A comparison of parallel solvers for diagonally dominant and general narrow-banded linear systems II, UT-CS-99-415, University of Tennessee, 1999, USA.

[4] D. Bini, Victor Y. Pan, Polynomial and Matrix Computations, Birkhäuser, Boston, 1994, pp. 208–211.

[5] A. Cleary, J. Dongarra, Implementation in ScaLAPACK of divide-and-conquer algorithms for banded and tridiagonal systems, UT-CS-97-358, University of Tennessee, 1997, USA.

[6] J.J. Dongarra, L. Johnsson, Solving banded systems on a parallel processor, Parallel Comput. 5 (1987) 219–246.

[7] J.J. Dongarra, A.H. Sameh, On some parallel banded system solvers, Parallel Comput. 1 (1984) 223–235.

[8] G.H. Golub, C.F. Van Loan, Matrix Computations, Section 4.3: Banded Systems, Johns Hopkins University Press, Baltimore, 1989, pp. 149–159.

[9] S.L. Johnsson, Solving narrow banded systems on ensemble architectures, ACM Trans. Math. Software 11 (3) (1985) 271–288.

[10] S. Lakshmivarahan, S.K. Dhall, Analysis and Design of Parallel Algorithms: Arithmetics and Matrix Problems, McGraw-Hill, New York, 1990.

[11] D.H. Lawrie, A.H. Sameh, The computation and communication complexity of a parallel banded system solver, ACM Trans. Math. Software 10 (2) (1984) 185–195.

[12] U. Meier, A parallel partition method for solving banded systems of linear equations, Parallel Comput. 2 (1985) 33–43.

[13] nCUBE 2 Processor Manual, nCUBE company, Foster City, CA, 1993.

[14] nCUBE 2 Programmer's Guide, nCUBE company, Foster City, CA, 1993.

[15] H.H. Wang, A parallel method for tridiagonal equations, ACM Trans. Math. Software 7 (1981) 170–183.

[16] W.M. Yan, K.L. Chung, J.G. Wu, Load-balanced parallel tridiagonal system solver, Research Report, Dept. of Information Mgmt. and Inst. of Computer Science and Information Eng., National Taiwan University of Science and Technology, 1999.