Proceedings of the IASTED International Conference
Parallel and Distributed Computing and Systems
November 3-6, 1999 in Cambridge Massachusetts, USA

A Simple Parallel Algorithm for Solving Banded Systems*

Sanguthevar Rajasekaran Jun Luo Peter Sheng

{raj, jluo}@cise.ufl.edu
CISE Department
University of Florida
Gainesville, FL

pete@coastal.ufl.edu
Coastal Engg. Dept.
University of Florida
Gainesville, FL

Abstract

In this paper we consider the problem of solving a banded system of linear equations. We present a simple algorithm. If the number of bands (or diagonals) is d, then the work done is $O(d^2n)$, where n is the number of equations. This algorithm is easily parallelizable on various models. For example, when $d = O(1)$, the algorithm runs in $O(\log n)$ time on an EREW PRAM using $\frac{n}{\log n}$ processors. When implemented on a cluster of p workstations, the algorithm runs in time $O\left(\frac{n}{p}\right)$ (for $d = O(1)$ and $p \leq \frac{n}{\log n}$).

Key Words: tridiagonal systems of equations, parallel algorithms

1 Introduction

The problem of solving a system of linear equations has many applications. An important special case is when the system is banded with d diagonals. When $d = 3$ and $d = 5$, we have a tridiagonal system and a pentadiagonal system, respectively. These two special cases themselves have numerous applications and have been studied extensively. For instance, a number of hydrodynamic models involve the solution of tridiagonal and pentadiagonal systems for their computer simulations.

The input are an $n \times n$ matrix A and an $n \times 1$ column vector b. The problem is to find an $n \times 1$ column vector x such that $Ax = b$. The only nonzero elements of A are along the main diagonal and the diagonals surrounding it. In particular, the number of nonzero diagonals is d. I.e., $A_{ij} \neq 0$ only if $|i - j| \leq \lfloor \frac{d}{2} \rfloor$.

This problem has been studied widely. For a comprehensive coverage on the known algorithms, the reader is refered to the texts [2] and [3]. For solving tridiagonal systems, there exist optimal parallel logarithmic time algorithms. The cyclic reduction algorithm [3] is one such. This algorithm is somewhat complex and applies only when $n = 2^k - 1$ for some integer k. Bar-On [1] has also given an algorithm that has the same performance. This algorithm is also fairly complex.

The algorithm presented in this paper is very simple, can handle any d, and efficiently parallelizes on many model. On the EREW PRAM, it runs in $O(\log n)$ time using $\frac{n}{\log n}$ processors when $d = O(1)$. On a cluster of p workstations, it runs in time $O\left(\frac{n}{p}\right)$ when $d = O(1)$ and $p \leq \frac{n}{\log n}$. The work done by the algorithm of Bar-On [1] is also $O(d^2n)$. When d is a costant, the algorithm of Bar-On and the algorithm presented in this paper have the same processor and time bounds. The algorithms of [1] are complex.

2 The New Algorithm

In this section we provide a detailed description of our algorithm and analyze its performance. We start by describing the algorithm as it applies to a tridiagonal system of linear equations and then extend it to a general d.

*This work is supported in part by an EPA Grant R-825-293-01-0.

$$
\begin{array}{cccccccccccc}
x_1 & x_2 & & & & & & & & & & \\
x_1 & x_2 & x_3 & & & & & & & & & \\
 & x_2 & x_3 & x_4 & & & & & & x_1 & x_4 & \\
 & & x_3 & x_4 & x_5 & & & & & x_1 & x_4 & x_5 \\
\hline
 & & & x_4 & x_5 & x_6 & & & & & & \\
 & & & & x_5 & x_6 & x_7 & & & & x_4 & x_5 \\
 & & & & & x_6 & x_7 & x_8 & & & & x_5 \\
 & & & & & & x_7 & x_8 & x_9 & & & \\
\hline
 & & & & & & & x_8 & x_9 & x_{10} & & \\
 & & & & & & & & x_9 & x_{10} & x_{11} & \\
 & & & & & & & & & x_{10} & x_{11} & x_{12} \\
 & & & & & & & & & & x_{11} & x_{12}
\end{array}
$$

$$
\begin{array}{cccc}
x_8 & x_9 & x_{12} \\
 & x_9 & x_{12}
\end{array}
$$

Input equations — After elimination

Figure 1: An Example for the Algorithm

Let $p \geq 1$ be the number of processors available. We have to solve the system $Ax = b$, where A is an $n \times n$ matrix and b is an $n \times 1$ column vector. For simplicity assume that n is an integral multiple of p. There are three *stages* in the algorithm. In the first stage, the equations are assigned to the processors evenly. The first processor gets the first $\frac{n}{p}$ equations, the second processor is assigned the next $\frac{n}{p}$ equations, and so on. Each processor eliminates as many unknowns as possible from out of the equations it got. Clearly, at the end of this stage, each processor will have two equations for a total of $2p$ equations. These $2p$ equations form a tridiagonal system. In the second stage, the new system is solved. This can be done either in parallel or employing a single processor. In the third stage, each processor back-substitutes the known values to get the values of all of its unknowns.

Note that if all the equations involving an unknown are with the same processor, the unknown can be eliminated in $O(1)$ time. Thus the total work done in the first stage is $O(n)$. The work done in the second stage is also $O(n)$. Call this algorithm Band.

Example. We present an example before proceeding further. Consider the case $n = 12$ and $p = 3$. Let the unknowns be $x_1, x_2, \ldots, x_{12}$. Each processor is assigned four equations. The first processor eliminates the unknowns x_2 and x_3. The second processor eliminates the unknowns x_6 and x_7 and the third processor eliminates the unknowns x_{10} and x_{11}.

In the second stage we have a system of 6 equations involving the unknowns x_1, x_4, x_5, x_8, x_9, and x_{12}. These equations also form a tridiagonal system. Once this system is solved, a back-substitution will complete the algorithm. ♣

3 PRAM Implementation

Now we show that the algorithm Band can be implemented optimally on a PRAM in logarithmic time. We assume the EREW PRAM model. This model consists of a collection of p synchronous processors [2]. Each processor is a Random Access Machine. The processors communicate with the help of a common memory. For instance if processor i wants to communicate with processor j, i writes its message in common memory cell j in one step and j reads it in the second step. There are many versions of the PRAM depending on how read and write conflicts are resolved. In the EREW PRAM, we assume that no two processors can access the same common cell at the same time either for the purpose of reading from or writing into.

To begin with, we employ $p = n$ processors. The algorithm runs in time $O(\log n)$ time. Later we indicate how to reduce the processor bound to $\frac{n}{\log n}$. There are two *phases* in the algorithm. In the first phase unknowns are eliminated in stages so that only $O(1)$ of them remain at the end. This system is solved in $O(1)$ time. In the second phase, the known solutions are back-substitutued to get the values of all the unknowns.

Algorithm PBand0

Elimination

> Consider a complete binary tree T with $\frac{n}{4}$ leaves. It helps to assume that there is a processor at each node of the tree. Each leaf is assigned four of the input equations. The leaves start by eliminating all the unknowns they can and send the remaining eqauations up the tree. Each internal node, upon receiving equations from its children, eliminates unknowns

and sends the remaining equations (at most two) to its parent.

When the root gets equations from its children, it solves those equations completely.

Back-Substitution

The root starts this phase by sending the values it obatained by solving its equations. It sends (at most) two values each to its children. Any internal node, upon receiving values from its parent, back-substitutes and knows the values of four unknowns. It sends two values each to its children.

When the leaves perform back-substituion, the values of all the unknowns are known.

Lemma 3.1 *Algorithm* PBand0 *runs in* $O(\log n)$ *time.*

Proof. The correctness of the algorithm is quite evident. The height of the complete binary tree T is $O(\log n)$. During the elimination phase, the time spent at each level of the tree is only $O(1)$ and hence this phase takes $O(\log n)$ time.

In the back-substitution phase also, the time spent at each level of the tree is only $O(1)$. Thus the whole algorithm runs in $O(\log n)$ time.

The processor bound of the above algorithm can be reduced to $p = \frac{n}{\log n}$ using a standard technique as follows.

Algorithm PBand

Step 1. Assign $2 \log n$ equations to each of the first $\frac{n}{2 \log n}$ processors. Each of these processors eliminates all the unknowns it can from its equations in $O(\log n)$ time.

Step 2. Now we have a tridiagonal system with $\frac{n}{\log n}$ equations. Solve this system using **Algorithm** PBand0.

Step 3. Each processor performs back-substitution to obtain the values of its $2 \log n$ variables.

Theorem 3.1 *A tridiagonal system of linear equations can be solved in time* $O(\log n)$ *using* $\frac{n}{\log n}$ *EREW PRAM processors.*

Proof. Step 1 takes $O(\log n)$ time. Step 2 can be completed in $O(\log n)$ time, in accordance with Lemma 3.1. The time taken by Step 3 is also $O(\log n)$. ♣

4 Implementation on a Cluster of Workstations

Algorithm Band can also be implemented efficiently on a cluster of workstations, for $d = O(1)$. If there are p processors then the run time is $O\left(\frac{n}{p}\right)$, for $p \leq \frac{n}{\log n}$. The use of workstation clusters has become an attractive alternative to the use of supercomputers. One could employ message passing mechanisms such as the PVM, MPI, or JAVA to connect heterogeneous machines to achieve parallelism. Very good speedups have been achieved using clusters.

The idea is very similar to Band. The input equations are with a host. The host partitions the input equations into p even parts and sends a part to every processor. Each processor eliminates as many variables as possible from the equations it receives from the host. After this, each processor will have at most two equations which will be sent to the host. The host solves the $2p$ equations in $O(p)$ time and sends two relevant values to each processor. Each of the processors, upon receiving two values from the host, back-substitutes these values to compute the values of all of its variables. Finally, the processors send their values to the host.

Excluding the time for communications, the total work done by the processors is $O(n) + O(p)$. The work done by each of the slave processors is $O\left(\frac{n}{p}\right)$. Thus the parallel time taken (omitting the communication time) is $O\left(\frac{n}{p} + p\right)$. This will be $O\left(\frac{n}{p}\right)$ if $p \leq \frac{n}{p}$. If $p > \frac{n}{p}$, the host can seek the help of the slaves some more. But in practice, taking into account the communication time also, it may not be necessary to communicate with the slaves more to seek their help.

5 General Banded Systems

In this section we deal with the case $d > 3$. Ideas similar to Band can be applied here as well. Here also we can consider a complete binary tree with $\frac{n}{d-1}$ leaves. Each leaf has $d-1$ equations to begin with. In the elimination phase, leaves start by sending their equations to their parents. Any internal node upon receiving $2(d-1)$ equations, eliminates as many variables as it can, and sends the remaining (at most $d-1$) equations to its parent. When the root gets equations, it solves them completely and the back-substitution phase starts thereafter.

Note that in the original input, each variable occurs in at most d equations. If a processor has all the d equations in which a variable x_i occurs, it can eliminate this variable in $O(d^2)$ time. Thus the time spent at each level of the tree T is $O(d^3)$. We can reduce this time to $O(d)$ preserving the work done, if we have d^2 processors at each node of the tree.

On an EREW PRAM, each of the two phases will take $O\left(d \log\left(\frac{n}{d-1}\right)\right) = O(d \log n)$ time, using $\frac{nd^2}{d-1}$ processors.

The processor bound of the above algorithm can be reduced to $\frac{nd}{\log n}$ using the technique employed for PBand. Each leaf in the tree is assigned $d \log n$ equations to start with. Every processor performs local elimination in $O(d \log n)$ time. At this time the above algorithm is used to solve the resulting $O\left(\frac{n}{\log n}\right)$ equations. After this, back-substitution can be done by the processors locally in time $O(d \log n)$.

The total work done in the modified algorithm is $O(nd^2)$. Thus we get the following

Theorem 5.1 *A banded system with d diagonals can be solved in $O(d \log n)$ time using $\frac{nd}{\log n}$ EREW PRAM processors.*

Note. In [1], Bar-On gives an algorithm for the above problem that runs in time $O(\log d \log n)$ that does $O(nd^2)$ work. In practice since we only have a small number of processors, the most crucial performance measure of any parallel algorithm is the total work done by the algorithm. Of course the number of processors assumed by the parallel algorithm should be reasonably large.

A theorem similar to Theorem 5.1 can be proven for a cluster of workstations also.

6 Conclusions

In this paper we have presented a simple algorithm for solving diagonal system of linear equations. When the number of diagonals d is a constant, our algorithm is asymptotically optimal. For any d, the total work done by our algorithm is $O(nd^2)$.

References

[1] I. Bar-On, A practical parallel algorithm for solving band symmetric positive definite systems of linear equations, *ACM Transactions on Mathematical Software* 13(4), 1987, pp. 323-332.

[2] J. Já Já, *Introduction to Parallel Algorithms*, Addison-Wesley Publishers, 1992.

[3] Lakshmivarahan and Dhall, *Parallel Numeric Algorithms*, Prentice Hall Publishing Company, 1990.