New Data Distribution for Solving Triangular Systems on Distributed Memory Machines *

Przemysław Stpiczyński

Department of Computer Science, Maria Curie–Skłodowska University
Pl. M. Curie-Skłodowskiej 1, 20-031 Lublin, Poland
przem@hektor.umcs.lublin.pl

Abstract. The aim of this paper is to show that the performance of distributed algorithms for solving triangular systems of linear equations can be improved by using a new data distribution of triangular matrices which provides steady distribution of sub-blocks among processes. The results of experiments performed on a cluster of Itanium 2 processors and on Cray X1 are also presented. The new method is faster than corresponding PBLAS routines `P_TRSV` and `P_TRSM`.

1 Introduction

The problem of solving triangular systems of linear equations belongs to the most important tasks of computational linear algebra and a large number of papers have appeared for handling such systems on distributed memory architectures [5–7]. The most popular solvers, namely `P_TRSV` and `P_TRSM` for solving $Ax = b$ and $AX = B$ (for multiple right hand sides) respectively, belongs to Level 2 and Level 3 PBLAS library designed as a part of Scalapack [2]. They use the block-cyclic data distribution of matrices onto rectangular process grids. The general class of such distributions can be obtained by matrix partitioning like

$$A = \begin{pmatrix} A_{11} & \dots & A_{1m} \\ \vdots & & \vdots \\ A_{m1} & \dots & A_{mm} \end{pmatrix} \tag{1}$$

where each sub-block A_{ij} is $m_b \times m_b$. These blocks are mapped to processes by assigning A_{ij} to the process whose coordinates in a $P \times Q$ grid are

$$loc(A_{ij}) = ((i-1) \textbf{ mod } P, (j-1) \textbf{ mod } Q)\,. \tag{2}$$

When A is a triangular matrix, then only lower (or upper) triangle is distributed and it looks like in Figure 1. It is clear that the block size m_b should be small enough to provide steady distribution. On the other hand, m_b should be large enough to utilize cache memory and provide reasonable performance of BLAS operations performed on sub-blocks of A [3].

* The extended abstract prepared for PARA 2006. The work has been sponsored by the KBN grant 6T11 2003C/06098. The use of Cray X1 from the Interdisciplinary Center for Mathematical and Computational Modeling (ICM) of the Warsaw University is kindly acknowledged.

A_{11}	
$A_{31}\ A_{33}$	A_{32}
$A_{51}\ A_{53}\ A_{55}$	$A_{52}\ A_{54}$
$A_{71}\ A_{73}\ A_{75}\ A_{77}$	$A_{72}\ A_{74}\ A_{76}$
A_{21}	A_{22}
$A_{41}\ A_{43}$	$A_{42}\ A_{44}$
$A_{61}\ A_{63}\ A_{65}$	$A_{62}\ A_{64}\ A_{66}$
$A_{81}\ A_{83}\ A_{85}\ A_{87}$	$A_{82}\ A_{84}\ A_{86}\ A_{88}$

Fig. 1. Block-cyclic distribution of a lower triangular matrix over a 2×2 processor grid.

2 New data distribution

In [8] we introduced a new method distribution symmetric and triangular matrices on orthogonal memory multiprocessors. This method can be adopted for distributing sub-blocks A_{ij} onto $P \times 1$ processor grid. Let us assume that $m = 2P$. Then

$$loc(A_{ij}) = \begin{cases} i-1 & \text{for } i = 1, \ldots, P \\ 2P - i & \text{for } i = P+1, \ldots, 2P. \end{cases} \tag{3}$$

Figure 2 shows the example of such distribution for $P = 4$. Note that each processor holds exactly the same number of sub-blocks of A. Our algorithm which uses (3) is based on the fan-out method presented in [5]. It solves systems of linear equations $AX = B$, where $A \in \mathbb{R}^{m \times m}$ and $B \in \mathbb{R}^{m \times n}$, just like the Level 3 PBLAS routine `P_TRSM`. Note that for $n = 1$, it performs the same computational task as the Level 2 PBLAS routine `P_TRSV`.

A_{11}	$A_{81}\ A_{82}\ A_{83}\ A_{84}\ A_{85}\ A_{86}\ A_{87}\ A_{88}$
$A_{21}\ A_{22}$	$A_{71}\ A_{72}\ A_{73}\ A_{74}\ A_{75}\ A_{76}\ A_{77}$
$A_{31}\ A_{32}\ A_{33}$	$A_{61}\ A_{62}\ A_{63}\ A_{64}\ A_{65}\ A_{66}$
$A_{41}\ A_{42}\ A_{43}\ A_{44}$	$A_{51}\ A_{52}\ A_{53}\ A_{54}\ A_{55}$

Fig. 2. New distribution of a lower triangular matrix over 4 processors.

3 Implementation, results and future work

The method has been implemented in Fortran and tested on a cluster of 16 Itanium 2 processors using Intel Fortran Compiler and Math Kernel Library as an efficient implementations of BLAS [3] and on four MSPs of Cray X1. In case of the Itanium cluster, the libraries PBLAS (routines `P_TRSV` and `P_TRSM`) and BLACS [4] (based on MPI, used for communication) have been downloaded from the Netlib. On Cray we have used corresponding routines provided by Cray. We

have compared the performance of the new method with the routine `P_TRSV` (Figure 3) and `P_TRSM` (Figure 4). The new method is much faster than the routine `P_TRSV`. For systems with multiple right hand sides it is also faster than `P_TRSM` for smaller values of n.

In the final version of the paper we are going to present the detailed complexity analysis described in terms of the BSP model of parallel computing [1] of the new method and compare it with the complexity of the corresponding Scalapack routines. Such analysis will show when the new method can be faster.

References

1. Bisseling, R.H.: Parallel scientific computation. A structured approach using BSP and MPI. Oxford University Press (2004)
2. Blackford, L., et al.: ScaLAPACK User's Guide. SIAM, Philadelphia (1997)
3. Dongarra, J., Duff, I., Sorensen, D., Van der Vorst, H.: Solving Linear Systems on Vector and Shared Memory Computers. SIAM, Philadelphia (1991)
4. Dongarra, J.J., Whaley, R.C.: LAPACK working note 94: A user's guide to the BLACS v1.1. http://www.netlib.org/blacs (1997)
5. Heath, M., Romine, C.: Parallel solution of triangular systems on distributed memory multiprocessors. SIAM J. Sci. Statist. Comput. **9** (1988) 558–588
6. Li, G., Coleman, T.F.: A new method for solving triangular systems on distributed-memory message-passing multiprocessors. SIAM J. Sci. Stat. Comput. **10** (1989) 382–396
7. Romine, C., Ortega, J.: Parallel solutions of triangular systems of equations. Parallel Comput. **6** (1988) 109–114
8. Stpiczyński, P.: Parallel Cholesky factorization on orthogonal multiprocessors. Parallel Computing **18** (1992) 213–219

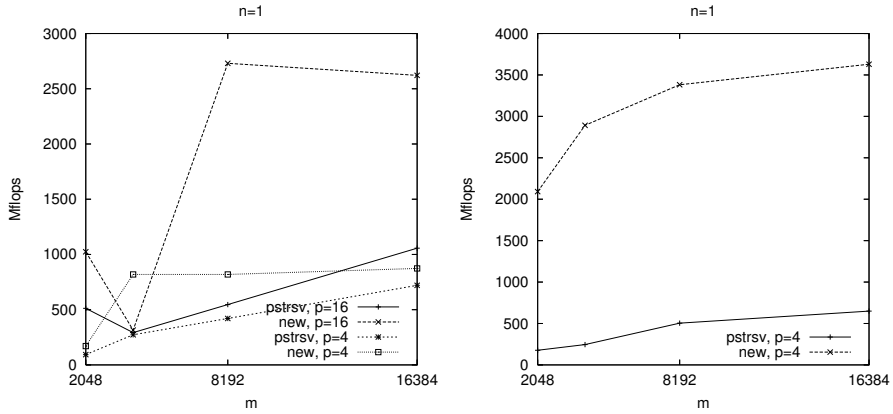

Fig. 3. The performance of the PBLAS routine `PSTRSV` and the new method on a cluster of Itanium 2 (left) and Cray X1 (right) for various matrix sizes (m) and $n = 1$.

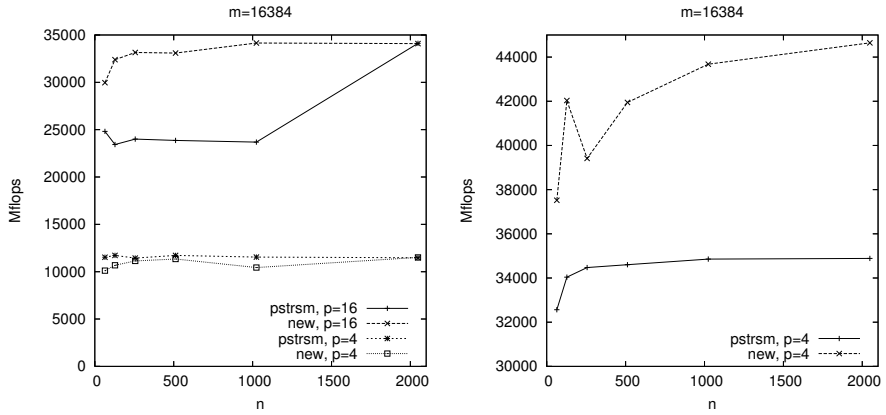

Fig. 4. The performance of the PBLAS routine `PSTRSM` and the new method on a cluster of Itenium 2 (left) and Cray X1 (right) for various number of right hand sides (n) and $m = 16384$.