Solving the Eigenvalues of Symmetric Sparse Matrices Applying Parallelism

M.A. Trenas
R. Asenjo
E.L. Zapata

September 1996
Technical Report No: UMA-DAC-96/32

Published in:

Euroconference: Supercomputation in Nonlinear and Disordered Systems:
Algorithms, Applications and Architectures
San Lorenzo de El Escorial, Madrid, Spain, September 23-28, 1996, pp. 347-351
(published by World Scientific Pub., Singapore, L. Vazquez, F. Tirado, and I. Martin, Eds., 1997)

University of Malaga
Department of Computer Architecture
C. Tecnologico • PO Box 4114 • E-29080 Malaga • Spain

SOLVING THE EIGENVALUES OF SYMMETRIC SPARSE MATRICES APPLYING PARALLELISM

MARIA A. TRENAS, R. ASENJO, E.L. ZAPATA
Dept. de Arquitectura de Computadores
Universidad de Malaga
email: {maria, asenjo, ezapata}@ac.uma.es

In this work we present a parallelization of Lanczos and Dongarra-Sorensen algorithms. Our trend is to solve the Symmetric Eigenproblem in Sparse Matrices in an efficient way. We have evaluated our results on multiprocessors CrayT3D, and Paramid.

1 The Eigenproblem

There are many applications, both in Science as in Engineering that require to solve the eigenproblem. The particular case of large sparse symnmetric matrices is of special interest. Indeed, this kind of problem arises in a wide range of applications, as are for example, dynamic analysis of large-scale structures, statistical analysis of data, the study of solar convection, etc.

In order to compute the eigenvalues we have proceeded in two main stages: firstly, we tridiagonalize the problem matrix (Lanczos method); and secondly, we compute the eigenvalues using the tridiagonal matrix we have obtained previously (Dongarra-Sorensen algorithm).

Lanczos algorithm[1,5] performs a tridiagonalization of a symmetric matrix $A \in R^{n \times n}$ in such a way that it is very well suited when it is a large and sparse one. The method generates a sequence of T_j with the propierty that the extremal eigenvalues of $T_j \in R^{j \times j}$ are progressively improving approximates of the extremal eigenvalues of A . This way, we arrive to obtain a tridiagonal matrix T with the same eigenvalues than A.

The Lanczos method projects matrix A over a Krylov subspace. This way, the orthogonal matrix Q that verifies the expresion $Q^t AQ = T$ turns out to be an orthonormal basis of this subspace, and its columns are but the the so called Lanczos vectors. For achieving its goal, the algorithm starts with an initial Lanczos vector, q_0 whose norm equals 1, obtaining in each iteration a new Lanczos vector, as pointed out by the recursive expression

$$Aq_j = \beta_{j-1} q_{j-1} + \alpha_j q_j + \beta_j q_{j+1} \qquad \beta_{-1}, q_{-1} \equiv 0$$

for j=0..n-2. Besides a new q_j, as a result of each iteration we will have a new β_j and α_j, elements of the subdiagonal and diagonal, respectively, of matrix

T_j, obtained after j+1 iterations. In exact arithmetic, the process would stop with the encountering of a $\beta_j = 0$, event that would signal the computation of an exact Krylov space $K(A, q_0, j)$.

However, in practice, the algorithm obtained in such a way does not work properly. As we don't work with an exact arithmetic, a loss of orthogonality between the Lanczos vectors occurs that darkens the termination condition and results in problems as the one of the phantom eigenvalues.

This loss of orthogonality is due to the appearance of small values of β_j, indicators of a cancellation in the computed r_j, auxiliar vectors with the same direction than the correspondig vector q_j. For solving this problem, we have chosen to implement a full reorthogonalization, ensuring that each new Lanczos vector we obtain is orthogonal to each of the previously obtained ones.

Once we have applied the Lanczos algorithm to the matrix A, we must solve the eigenproblem associated with the new matrix T. With a view to doing so, we have used the Dongarra-Sorensen algorithm[2].

Dongarra-Sorensen algorithm let us to solve the eigenproblem for symmetric tridiagonal matrices. It is based upon a divide and conquer scheme, very well suited for being parallelized.

The algorithm consists in the successive splitting of the original problem into subproblems of less size and complexity. For this purpose we use rank-1 modifications of the form:

$$T = \begin{bmatrix} T_1 & \rho_k e_1^T \\ \rho_1 e_k^T & T_2 \end{bmatrix} = \begin{bmatrix} \hat{T_1} & 0 \\ 0 & \hat{T_2} \end{bmatrix} + \rho \begin{bmatrix} e_k \\ e_1 \end{bmatrix} \begin{bmatrix} e_k^T & e_1^T \end{bmatrix}$$

Because of the above process, we deal with a computational structure based in a binary tree. Once we have solved the subproblems belonging to the lowerest level of the tree, applying a standard solver for the eigenproblem, as may be *dsteqr* from LAPACK, a recontruction process begins. This process involves obtaning a node's eigenvalues using the information provided by its right and left sons. Traversing the tree in such a way, from the bottom to the top (iterative or bottom-up scheme[3]), and after consecutive recontructions, we arrive to a solution of the original problem, the one associated with the primary node.

Using this technique is advantageous mainly because of the important decrease in the execution time demanded by the routine *dsteqr* as the problem to deal with becomes smaller, and also by reason of the relatively small complexity of the reconstruction process.

This simplicity of the reconstruction process is due to the rank-1 tearing satisfying the interlacing property[1], stating

$$d_1 < \lambda_1 < d_2 < \lambda_2 < \cdots < d_n < \lambda n$$

where λ_i are the eigenvalues we are searching, belonging to the "father" problem, and d_i are the eigenvalues of the son subproblems of that one. Thanks to this propierty we can build an iterative method for the recontruction process. An implementation of this method can be found in LAPACK, in routine *dlaed4*.

2 Parallelization of the algorithms

As refers to the Lanczos algorithm, we have implemented a medium granularity parallelization, at the loop level. We have projected the iterations associated with vectorial operations over the two dimensions of the mesh. With this view, the dense data structures have been distributed in a cyclic way. The sparse matrix follows the BRS distribution scheme [4] assuring this way, the perfect alignement with the dense ones. Because of the data dependences, the extern loop of the algorithm can not be distributed over the processors.

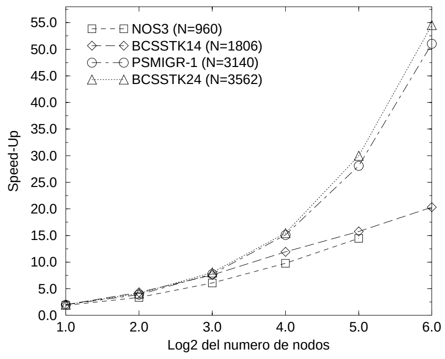
Dongarra algorithm parallelization is based upon its binary tdree structure. In the lowerest level of the tree, we have a number of subproblems equal to the number of available processors (a power of two) to be solve with *dsteqr*. Then we come into an iterative process of consecutive recontructions, traversing the tree from bottom to up. The number of processors that will colaborate in the resolution of the same subproblem will be multiplied by two with each new iteration.

The computers used for the evaluation of our algorithms have been the Cray T3D, as well as the Paramid. Both of them are distributed memory multiprocessors. Matrices from the Harwell-Boeing collection have been selected for testing.

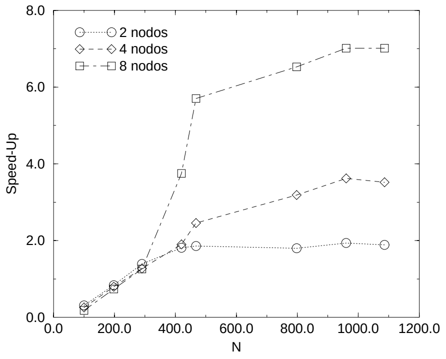
Figure 1 plots the Speed-Up values obtained with our parallel Lanczos algorithm using SHMEM on Cray T3D. We show the mesh configurations that furnish us with the better results for a fixed number of processors. Our measurements showed that the algorithm is scalable in both mesh dimensions, though with a better performance in topologies having the X dimension slightly bigger than the Y one.

The speed-up values obtained with Dongarra algorithm, are displayed also in figure 1. It can be observed that are very close the ideal ones, even using the PVM programming model. The reason is most of the time in the sequential algorithm is spend solving the lowerest level subproblems (appliying routine *dsteqr*), and in the recontruction process.

The time employed in recontruction (mainly routine *dlaed4*) will be approximately divided between the number of available processors: each one of them will apply the iterative method for the computation of a subset of the



(a) Lanczos Speed-Up using SHMEM



(b) Dongarra Speed-Up using PVM

Figure 1: Speed-Up of the parallel algorithms

eigenvalues. And as refers to the *dsteqr* routine it has a behaviour such that solving a problem of half the size than the original one will require less than half the time.

That is why in the sequential version of the algorithm we obtained speed-up values (refered to the direct application of the standard routine *dsteqr*) as high as twenty.

3 Conclusions

We have parallelized two algorithms for computing the eigenvalues of sparse symmetric matrices that apply the Lanczos and Dongarra-Sorensen methods on a mesh topology. As we showed in the above section, the results obtained evaluating those algorithms were satisfactory enough.

For both Lanczos and Dongarra algorithms, we have observed good values in the efficience, as well as a good scalability. And in the particular case of Dongarra algorithm, we have been able to confirm the efficiency of the divide and conquer scheme. We obtained pretty good improvements in the sequential algorithm just by using it.

However, in Lanczos algorithm, full reorthogonalization results in a very high increment of the execution time, due to the introduction of a dense matrix built with the Lanczos vectors. This implies not only a great increment in the memory requirements, but also the introduction of dense matrix-vector products of high computational cost. That is the reason of our working, in this moment with the view of implementing partial or selective reorthogonalization methods, of less cost.

References

1. G.H.Golub, C.F. Van Loan. *Matrix Computations*, second edition. The Johns Hopkins University Press, 1993.
2. J.J. Dongarra, D.C.Sorensen. *A Fully Parallel Algorithm for the Symmetric Eigenvalue Problem.* SIAM J. Sci. Stat. Comput., 2(1987),pp.139-154.
3. S. Sur, W. Bhm. *Analysis of Non-Strict Functional Implementations of the Dongarra-Sorensen Eigensolver.* ACM (1994),pp.412-418.
4. R. Asenjo, L.F. Romero, M. Ujaldn, E.L. Zapata. *Sparse Block and Cyclic Data Distributions for Matrix Computations.* In L. Grandinetti, G. R. Joubert, J. J. Dongarra, and J. Kowallik, editors; High Performance Computing, Technology and Aplications. Elsevier Science 1994, pp.6-8.
5. M.A. Trenas, *Parallel Algorithms for Eigenvalues Computation with Sparse Matrices*, Master Thesis, University of Malaga, November 1995 (in spanish).