A Study on the Efficient m-step Parallel Generalization

Sun Kyung Kim *

* School of Computer and Information Technology, Daegu University, Korea
skim@daegu.ac.kr

Abstract - It would be desirable to have methods for specific problems, which have low communication costs compared to the computation costs, and in specific applications, algorithms need to be developed and mapped onto parallel computer architectures. Main memory access for shared memory system or global communication in message passing system deteriorate the computation speed. In this paper, it is found that the *m*-step generalization of the block Lanczos method enhances parallel properties by forming *m* simultaneous search direction vector blocks. QR factorization, which lowers the speed on parallel computers, is not necessary in the *m*-step block Lanczos method. The *m*-step method has the minimized synchronization points, which resulted in the minimized global communications compared to the standard methods.

Keywords: Block Lanczos Algorithm, QR Factorization, m-step Parallelization

1 Introduction

Memory contention on shared memory machines constitutes a severe bottleneck for achieving their maximum performances [11]. The same is true for communication costs on a message passing system [10]. It would be desirable to have methods for specific problems, which have low communication costs compared to the computation costs. This is interpreted as a small number of main memory access for the shared memory systems and a small number of global communications for the message passing systems. It also reduces the need for frequent synchronizations of the processors. Linear algebra algorithms, which are implemented efficiently on parallel computers, have also been studied [3,6,8]. Many important scientific and engineering problems require the computation of a small number of eigenvalues of symmetric large sparse matrices. One of the commonly used algorithm for solving a multiple eigenvalue problem is the block Lanczos algorithm. QR factorization process in the block Lanczos method is bottleneck when this method is implemented on parallel computers because QR factorization process requires many synchronization points[1,9]. In this paper we introduce the *m*-step block Lanczos method that need no *QR* factorization process. In the *m*-step method, *m* consecutive steps of the standard method are performed simultaneously. This means, for example, that the inner products needed during *m* steps of the standard method can be performed simultaneously.

2 The parallel Block Lanczos method

2.1 The Block Lanczos method

The block Lanczos algorithm for computing extreme multiple eigenvalues of symmetric matrices is based on the block Lanczos recursion for the block tridiagonalization of a real symmetric matrix. The block Lanczos is as follows:

Algorithm 2.1 The block Lanczos algorithm

$X_1 \in IR^{n\times p}$ given, with $X_1^T X_1 = I_p$.

$M_1 = X_1^T A X_1$

For j = 1 **until** Convergence **Do**

$C_j = AX_j - X_j M_j - X_{j-1} B_{j-1}^T \; (X_0 B_0^T = 0)$

$X_{j+1} B_j = C_j$ (QR factorization process)

$M_{j+1} = X_{j+1} A X_{j+1}$

EndFor

At the beginning of the j-th pass through the loop we have

$$A[X_1,...,X_j] = [X_1,...,X_j]T_j + C_j[0,...,0,I_j],$$

where $T_j = \begin{bmatrix} M_1 & B_1^T & & & \\ B_1 & M_2 & & & \\ & . & . & . & \\ & & . & . & . \\ & & & B_{j-1} & M_j \end{bmatrix}$

The eigenvalues of the block tridiagonal matrix T_j are called Rits values of large sparse matrix A. For many matrices and for relatively small j several of the extreme multiple eigenvalues of A, that is several of the algebraically-largest or algebraically-smallest of the eigenvalues of A, are well approximated by eigenvalues of the corresponding matrices T_j. The steps to implement the algorithm 2.1 on parallel computer is as follows:

Select Q_1 of size $n \times p$ where p is at least as large as the number of eigenvalues desired and the columns of Q_1 are orthonormal.

For $j = 1$ **until** Convergence **Do**

1. compute AQ_j
2. compute (AQ_j, Q_j)
3. $M_j = (AQ_j, Q_j)$: $p \times p$ matrix
4. $C_j = AQ_j - Q_j M_j - Q_{j-1} R_{j-1}^T$ $(Q_0 R_0^T = 0)$
5. Apply a modified Gram-Schmidt orthogonalization to the columns of C_j
   (that is, $Q_{j+1} R_j = C_j$:QR factorization)

EndFor

Next, compute the p algebraically-largest eigenvalues of T_j. The Ritz vector $Q_j y (=Z)$ obtained from an eigenvector y of a given T_j is an approximation to a corresponding eigenvector of A.

The above block Lanczos procedure with no reorthogonalization has minimal storage requirements and can therefore be used on very large matrix A, if only approximations to eigenvalues are required. However, the inner products cannot be performed simultaneously because of steps 2, 5 and inner products require global communication among all processors on message passing systems. These force several accesses of vectors Q_j, D_j, AQ_j per one iteration of above procedure. For shared memory systems, main memory accessing may be slow. So in the section 2.2, the m-step block Lanczos method is proposed to perform all inner products needed for m iterations of above procedure. For QR factorization, $p(p+1)/2$ inner products should be separately repeated $2p-1$ times and so communication time rate is high for this QR factorization process. In the next section we introduce m-step block Lanczos method which has much less synchronization points and need no QR factorization process.

2.2 m-step block Lanczos method

One way that can lead to build a new parallel block Lanczos algorithm is to perform m steps of the standard algorithm simultaneously in parallel using a set of $m \times p$ linearly independent vectors. In this case, the set of $m \times p$ linearly independent vectors $\overline{V_k}$ is spanned by left $[V_k^1, AV_k^1, \ldots, A^{m-1}V_k^1, V_k^1 \in IR^{n \times p}]$.

Remark 1

If $\overline{V_i}$ and $\overline{V_j}$ are orthogonal for $i \neq j$, $V_k = [\overline{V_1}, \overline{V_2}, \ldots, \overline{V_k}]$ can be decomposed into $Q_k * R_k$, where $Q_k = [\overline{Q_1}, \overline{Q_2}, \ldots, \overline{Q_k}]$ and $R_k = \text{diag}\,[\overline{R_1}, \overline{R_2}, \ldots, \overline{R_k}]$.

Remark 2

If T_j is a symmetric tridiagonal matrix generated by the standard block Lanczos algorithm and $T_k = R_k^{-1} T_j R_k$ where $j = m*k$, $\overline{T_k}$ becomes a non-symmetric matrix similar to T_j as follows:

$$\overline{T_k} = \begin{bmatrix} G_1 & E_1 & & & \\ F_1 & G_2 & E_2 & & \\ & . & . & . & \\ & & & & E_{k-1} \\ & & & F_{k-1} & G_k \end{bmatrix}$$

where $G_i = [G_i^1, \ldots, G_i^m]$, $G_i^j \in IR^{mp \times p}$ and $E_i = [E_i^1, \ldots, E_i^m]$, $E_i^j \in IR^{mp \times p}$ for $i = 1, \ldots, k$, and $j = 1, \ldots, m$. Here F_i is a specially shaped matrix of which the block in the upper right corner is a triangular matrix with zero elements otherwise. This upper triangular matrix arises from the QR factorization of V_i^l, together with orthonormal vectors. The following is an example of F_i for $p = 3$, and $m = 2$:

$$F_i = \begin{bmatrix} 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & 0 & * \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Since $\overline{T_k}$ is similar to T_j with $j=k*m$, they have the same eigenvalues. Note that V_i^l is not orthogonal, but it does not induce any problem in constructing the algorithm. The reason is that, as given in Remark 1, $\overline{V_i}$ and $\overline{V_j}$ are orthogonal for $i \neq j$ and $\overline{V_i} = [V_i^1, V_i^2, \ldots, V_i^m]$ is a set of linearly independent vectors that span

$[V_i^1, AV_i^1, ..., A^{m-1}V_i^1, \ V_k^i \in R^{n\times p}]$. Based on the above analysis, a new parallel block Lanczos algorithm can be constructed as follows:

Algorithm 2.2 The *m*-step block Lanczos algorithm

$\overline{V}_0 = 0, \overline{V}_1 = [V_1^1, AV_1^1, A^2V_1^1, ..., A^{m-1}V_1^1]$

For $k = 1$ **until** Convergence **Do**

Select G_k, E_{k-1}

so that $\overline{V}_k$ is orthogonal to $\overline{V}_{k-1}$.

This gives

$V_{k+1}^1 = AV_m^k - \overline{V}_{k-1}E_{k-1}^m - \overline{V}_k G_k^m$

Select C_k^j so that $\overline{V}_{k-1}$ is orthogonal to $[V_k^1, AV_k^1, ..., A^{m-1}V_k^1]$ which gives

$V_{k+1}^j = A^{j-1}V_{k+1}^1 - \overline{V}_k C_k^j$ for $j = 2, \ldots, m$.

EndFor

We will present the reduction matrix generated by *m*-step block Lanczos method for the special case of $m = 2$ and $p = 2$:

$$\overline{T}_k = \left[\begin{array}{cccc|cccc|cc}
* & * & * & * & * & * & * & * & & \\
* & * & * & * & * & * & * & * & & \\
* & * & * & * & * & * & * & * & & \\
* & * & * & * & * & * & * & * & & \\
\hline
 & & 1 & 0 & * & * & * & * & . & \\
 & & & 1 & * & * & * & * & . & \\
 & & & & * & * & * & * & . & \\
 & & & & & * & * & * & . & \\
 & & & & & & & & & \\
 & & & & . & . & . & . & . & . \\
 & & & & . & . & . & . & . & .
\end{array}\right]$$

Next, we demonstrate how to determine the parameters G_k, E_{k-1}, C_k in the above process:

Scalar 1: $\overline{V}_k{}^T A\overline{V}_k = \overline{V}_k{}^T \overline{V}_k G_k$,

$\overline{V}_{k-1}{}^T A\overline{V}_k = \overline{V}_{k-1}\overline{V}_{k-1}E_{k-1}$

Scalar 2: $0 = \overline{V}_k{}^T A^{j-1}V_k^1 - \overline{V}_k{}^T \overline{V}_k C_k^j$

The inner products $(\overline{V}_k, \overline{V}_k)$, $(\overline{V}_k, A\overline{V}_k)$, $(\overline{V}_{k-1}, A\overline{V}_k)$, $(\overline{V}_k, A^{j-1}V_k^1)$ can be reduced to, (V_k^1, V_k^1), (V_k^1, AV_k^1), ... , $(V_k^1, A^{2m-1}V_k^1)$ in a similar way to the *s*-step Lanczos method[7]. We now formulate the *m*-step block Lanczos algorithm.

Select V_1^1

For $k = 1$ **until** Convergence **Do**

1. Compute $AV_k^1, A^2V_k^1, ..., A^mV_k^1$
2. Compute $2p^2m$ inner products
3. Compute Scalars

   C_{k-1}^j for $j = 2, ..., m$

   E_{k-1}^j, G_k^i for $i = 1, ..., m$
4. Compute

   $V_k^j = A^{j-1}V_k^1 - \overline{V}_{k-1}C_{k-1}^j$

   (with $\overline{V}_0 = 0, C_0^j = 0$)
5. Compute AV_k^m
6. Compute

   $V_{k+1}^1 = AV_k^m - \overline{V}_{k-1}E_{k-1}^m - \overline{V}_k G_k^m$

   (with $E_0^m = 0$)

EndFor

In the above algorithm all the inner products are performed at once in the step 2. Matrix-vector operations are performed in the step 1 and 5, and the other steps are for vector update operation.

3 Analysis of *m*-step iterative methods

During the procedure of the standard block Lanczos algorithm in the Section 2.1, $p^2+p(p+1)/2$ inner products should be separately repeated at $2p$ times. Thus on message passing system, lots of times are needed for global communication, which in turn reduces the efficiency of the parallel system.

However, the new parallel algorithm introduced in Section 2.2 can perform all the needed inner products at once during *m* iterations of the corresponding standard method, the time necessary for inner products is reduced by a factor of $1/(2pm)$ compared to the standard method. The main cause for the increased memory latency time for the shared memory system is too many synchronizing point. In a new parallel algorithm 2.2 such a bottleneck phenomena can be reduced. Table 1 shows the number of vector operations and the data communications during a single iteration of the m-step block Lanczos algorithm and *m* iterations of the corresponding standard method.

Table 1 Comparison of the numbers of the vector operations and the data communications for block Lanczos method

	standard algorithm	m-step algorithm
inner product	$[p^2+p(p+1)/2]m$	$2p^2m$
vector update	$(2p+1)pm$	$pm(m+1)$
matrix-vector multiplication	pm	$p(m+1)$
Global communication	$2pm$	1
Local communication	m	2

As shown in Table 1, the communication cost can be greatly decreased by introducing more effective algorithm in parallel process.

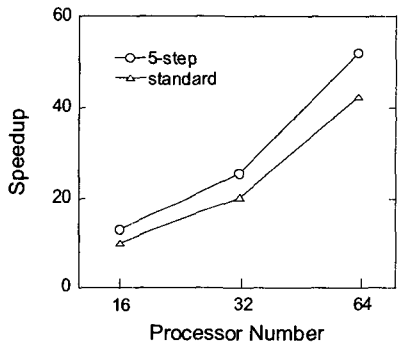

Fig. 1. Perfomance of the standard and the m-step block Lanczos algorithm in Cray T3E.

Fig. 1 shows the efficiency of the 5-step block Lanczos algorithm implemented on MP parallel computer Cray T3E. All the inner products needed for m iterations of the standard method are performed at the same time in the m-step method, so that only one global communication is required and the communication cost is decreased on a message passing system like Cray T3E. But the m-step block Lanczos method needs $p(m+1)$ times of matrix-vector operations compared to pm times in the standard methods. But in the case of vector updates, the number is slightly smaller in the m-step method. The m-step algorithm is also effective in the matrix-vector operations since the communication time between the adjacent processors of Cray T3E can be reduced. While the efficiency of parallelism increases with increasing m and p, but a loss of accuracy for eigenvalues has been observed for large $m > 5$.

4 Conclusions

Parallel processing systems equipped with from a few to many thousand processors are now being used in many areas. As the number of the processors involved in the parallel system is increased, the relative importance of the communication cost grows. In this paper, we proposed new m-step iterative method suitable to reduce the communication cost. The m-step block Lanczos algorithm utilizes a reduced matrix similar to that in the standard block Lanczos method with the same eigenvalues, but m-step method is more effective in the parallel system because a large amount of the inner products can be done at once. This process can reduce the data communication time in a message passing system. The m-step methods also help to reduce the memory latency time in shared memory systems by reducing the synchronization point showing a better data locality. The new m-step method has the better performance compared to the standard method in MP parallel computer Cray T3E.

References

[1] Claus Bendtsen, Per Christian Hansen, Kaj Madsen, Hans Bruun Nielsen, Mustafa Pinar, "Implementation of QR up- and downdating on a massively parallel computer", *Parallel Computing* 21 (1995) 49-61

[2] Jane K. Cullum and Raph A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalues Computation*, Birkhauser Boston, Inc. (1985).

[3] Ameet K. Dave and Iain S. Duff, "Sparse Matrix Calculations on the CRAY-2", *Parallel Computing* 5 (1987) 55-64

[4] James W. Demmel, *Applied Numerical Linear Algebra*, the Society for Industrial and Applied Mathematics press. (1997)

[5] G. H. Golub, C. F. Van Loan, *MATRIX Computations*, Johns Hopkins University Press. (1996)

[6] Inge Gutheil and Werner Krotz-Vogel, "Performance of a Parallel Matrix Multiplication Routine on Intel iPSC/860", *Parallel Computing* 20 (1994) 953-974

[7] Sun Kyung Kim and A. T. Chronopoulos, "A Class of Lanczos Algorithms Implemented on Parallel Computers", *Parallel Computing* 17 (1991) 763-778

[8] K.K. Mathur and S. Lennart Johnsson, "Multiplication of Matrices of Arbitrary Shape on a Data Parallel Computers", *Parallel Computing* 20 (1994) 919-951.

[9] Pontus Matstoms, "Parallel sparse QR factorization on shared memory architectures", *Parallel Computing* 21 (1995) 473-486

[10] Sanjay Ranka, Youngju Won, and Sartaj Sahni, "Programming the NCUBE Hypercube", *Tech. Rep. Csci* No 88-13, Univ. of Minnesota, (1988).

[11] Paul E. Saylor, "Leapfrog Variants of Iterative Methods for Linear Algebraic Equations", *Journal of Computational and Applied Mathematics* 24 (1988) 169-193