A Multi-color Inverse Iteration for a High Performance Real Symmetric Eigensolver

Ken Naono[1], Yusaku Yamamoto[1], Mitsuyoshi Igai[2], Hiroyuki Hirayama[2], and Nobuhiro Ioki[3]

[1] Hitachi,Ltd., Central Research Laboratory
[2] Hitachi ULSI Corp.
[3] Hitachi,Ltd., Software Division

Abstract. An implementation of a real symmetric eigensolver on parallel nodes is described and evaluated. To achieve better performance in the inverse iteration part, a multi-color framework is introduced, in which the orders of the orthogonalizations are rescheduled so that the inverse iterations are executed concurrently. With the blocked tridiagonalization and backtransformation, our real symmetric eigensolver shows good performance and accuracy both on the MPP SR2201 and on the newly developed hybrid machine SR8000.

1 Introduction

In this paper, we treat an implementation of an eigensolver for dense real symmetric matrices that consists of tridiagonalization, bisection, inverse iteration and backtransformation. In each part, we adopted the existing algorithms, improved them from implementative point of views and produced an eigensolver of the matrix library MATRIX/MPP(03-00) for the hybrid[1] machine SR8000 [1]. For the tridiagonalization, the blocked method [2] [3] and lowering byte/flop techniques [4] were found to be successful. The bisection part can also be effectively parallelized [5]. When it comes to calculating a lot of eigenvectors, however, it is known that its parallel computation based on the conventional inverse iteration performs poorly because of the reorthogonalizations [6].

In 1997, Dhillon [7] proposed a new algorithm that solves each eigenvector in $O(N)$ time and produces automatically orthogonal eigenvectors without any reorthogonalization. The algorithm was implemented in the latest LAPACK(version 3.0) [8] subroutine dstegr, but the Dhillon's algorithm does not always work well when the relative gaps of eigenvalues are very small. In such cases, users have to use the conventional inverse iteration 'dstein' and endure poor performance. Furthermore, the ScaLAPACK [9] 'pdstein' allocates the clusters on the processing nodes, which usually results in the biased workload.

In this paper, we describe a new framework for the parallel computation of eigenvectors. Our framework, which we call a multi-color inverse iteration, was

[1] hybrid = combination of SMPs(symmetric multiple processors) and MPPs(massively parallel processors)

A. Bode et al. (Eds.): Euro-Par 2000, LNCS 1900, pp. 527–531, 2000.

published locally in Japan [10]. It is based on the theory of the conventional inverse iteration with reorthogonalizations [11] [12]. One feature of our framework is that the orders of reorthogonalizations are rescheduled with coloring so that dependent eigenvalues are differently colored. Another is that the eigenvectors are evenly distributed over the nodes. Our framework enables some part of eigenvectors to be solved concurrently even though the reorthogonalizations are executed.

2 The Multi-color Inverse Iteration

The inverse iteration with reorthogonalizations is usually described as follows.

$$(T - e_i I)v_i^{k+1} = v_i^k, \qquad k = 0, 1, ..., \tag{1}$$

$$v_i^k := v_i^k - \sum_{j \in O_i} (v_i^k, v_j)v_j. \tag{2}$$

where T is a real symmetric tridiagonal matrix, I is the unit matrix, e_i is the i-th eigenvalue, v_i is the corresponding eigenvector and v_i^k is its k-th iterate. O_i denotes the indices set of eigenvectors against which v_i^k is reorthogonalized.

In the ScaLAPACK pdstein, the indices set O_i^{stein} is $\{\ j \in \mathcal{N}\ ;\ j < i,\ |e_j - e_i| < eps\ \}$, where 'eps' is the reorthogonalization criterion. So, all the eigenvalues in Fig.1, [2] for example, are gathered in one group and the eigenvectors are allocated in one node.

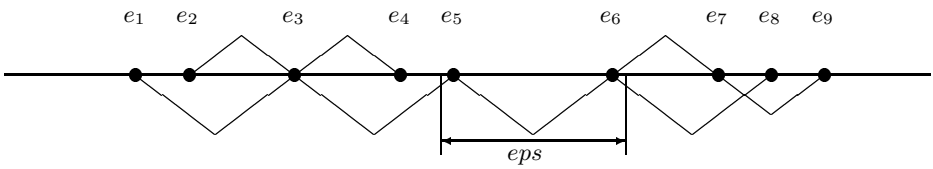

Fig. 1. The reorthogonalization criterion '*eps*' and 'connected' eigenvalues

In the multi-color inverse iteration, we find out the eigenvectors which can be solved independently. For that, we give colors to all the eigenvectors on conditions that connected eigenvectors should be colored differently. Table 1 gives a simple and easily implemented algorithm to color. The result is in the 0th stage of Fig.2. The calculations of eigenvectors with the same color have no data dependency with each other and can be done in parallel, because the eigenvectors need not be reorthogonalized with each other.

The colors also play a role of the priority of computation. First, eigenvectors with $color(i) = 1$ are calculated, second, those with $color(i) = 2$, and so on. The indices set O_i^{multi} is $\{\ j \in \mathcal{N}\ ;\ |e_j - e_i| < eps,\ color(j) < color(i)\ \}$.

[2] The eigenvalues are connected with polygonal lines when the distances between them are less than the reorthogonalization criterion 'eps'.

The eigenvectors are evenly distributed among nodes as in Fig.2. If necessary, each node receive the calculated eigenvectors from other nodes by internodes communication. In the second stage, for example, v_6 is transferred from N1 to N2, and v_7 is calculated by the inverse iteration with reorthogonalizations against v_6 and v_9. Thus the multi-color framework enables effective parallel implementation that is summerized in Table 2.

Table 1. A greedy algorithm for coloring eigenvalues ($e_i \leq e_j$ for $i < j$)

1. Set $color(1) = 1$ and $i = 2$.
2. Set $color(i)$ to a natural number satisfying the following two conditions.
   - As small as possible.
   - For any j with $0 \leq e_i - e_j < eps$, $color(i) \neq color(j)$.
3. $i = i + 1$, and if $i \leq n$ goto 2, otherwise stop.

coloring accomplished (0th stage)

node	N0			N1			N2		
eigenvector	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
$color = 1$	○			○		○			○
$color = 2$		○			○		○		
$color = 3$			○					○	

1st stage

node	N0			N1			N2		
eigenvalue	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
$color = 1$	◎			◎		◎			◎
$color = 2$		○			○		○		
$color = 3$			○					○	

2nd stage

node	N0			N1			N2		
eigenvector	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
$color = 1$	○			○		○			○
$color = 2$		◎			◎		◎		
$color = 3$			○					○	

3rd stage

node	N0			N1			N2		
eigenvector	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
$color = 1$	○			○		○			○
$color = 2$		○			○		○		
$color = 3$			◎					◎	

Fig. 2. Allocation and the multi-color inverse iteration procedure

3 Numerical Tests and Remarks

First, on one node of the SR2201 [3], we compare the residual, orthogonality and time of the multi-color implementation with those of the equivalence of the LAPACK dstein for the [1,2,1] matrix with dimension 2000. The orthogonality is scaled as $||V^T V - I||_F, V_{ij} = (v_i, v_j)$ and the reorthogonality criterion '*eps*' changed from 10^{-6} to 1.0. The result in Table 3 shows that the multi-color inverse iteration performs better and the residual and orthogonality are almost same as those of the equivalence of dstein.

[3] The SR2201 has 300Mflop/s per 1 node and 300MB/s internode bandwidth.

Table 2. A parallel implementation of the multi-color framework

1. Calculate the indices set O_i^{multi} for all i on all nodes.
2. Define the $color(i)$ for all i on all nodes.
3. Do k = 1, Total_Color_Number
   Do i = 1, Total_Eigenvector_Number
   IF (Color(i)=k and My_Node_Number=Eigenvec_Alloc(i))
   (a) Get $v_j \in O_i^{multi}$ from other nodes if necessary.
   (b) Do the inverse iteration performing the following stages alternately.
      i. Solve the linear equation (1).
      ii. Reorthogonalize as in the equation (2) with O_i^{multi}.

Second, we evaluate scalability of the multi-color implementation on the SR2201 for the [1,2,1] matrix with dimension 8000. The result in Table 4 shows that scalability is low and there is room for improvement, especially in eps=1.0e-2. But note that, with the ScaLAPACK pdstein, all eigenvectors in that case fall into one group and no parallelism would be achieved.

We also evaluate the performance and accuracy for a real symmetric eigensolver [4] with the multi-color inverse iteration on the SR8000[4] . Test matrices are the Frank matrix $A_{ij} = min(i, j)$ with the dimension 8000 and 16000. The reorthogonalization criterion used is 10^{-5}. The execution time of each part, and total accuracy(r:residual, o:orthogonality) and performance are shown in Table 5. In both dimensions, the multi-color inverse iteration is confirmed to scale well, and total high performance is achieved.

However, we have to prove rigorously for rescheduling and test on a lot of clustered matrices, which will be our future work. Adoption of 'dtwqds [7]' will be important to solve the low scalability problem.

References

1. SR8000 HOMEPAGE: http://www.hitachi.co.jp/Prod/comp/hpc/eng/sr81e.html
2. J. J. Dongarra and R. A. van de Geijn: Reduction to condensed form for the Eigenvalue problem on distributed architectures, Parallel Computing Vol. 18, No. 9, pp. 973-982, 1992.
3. H. Y. Chang, S. Utku, M. Salama and D. Rapp: A parallel Householder tridiagonalization stratagem using scattered square decomposition, Parallel Computing Vol. 6, No. 3, pp. 297-311, 1988.
4. K. Naono, Y. Yamamoto, M. Igai, H. Hirayama: High performance implementation of tridiagonalization on the SR8000, Proceedings of the Fourth International Conference/Exhibition on High Performance Computing in Asia-Pacific Region (HPC-ASIA2000), Beijing, China, pp.206-219, 2000.

[4] The SR8000 has 8Gflop/s per 1 node and 1GB/s internode bandwidth.

Table 3. Residual, orthogonality, and performance on 1 node of the SR2201

eps		1e-6	1e-5	1e-4	1e-3	1e-2	1e-1	1e-0
res	multi-color	4.2e-14	4.2e-14	4.2e-14	4.2e-14	4.2e-14	4.2e-14	3.8e-14
	equi-dstein	4.2e-14	4.2e-14	4.2e-14	4.2e-14	4.2e-14	4.1e-14	3.5e-14
ortho	multi-color	4.5e-11	4.5e-11	1.4e-11	4.2e-12	9.7e-13	2.7e-13	8.3e-14
	equi-dstein	4.9e-11	4.9e-11	1.4e-11	4.3e-12	9.2e-13	2.5e-14	8.2e-14
time	multi-color	3.73s	3.75s	3.75s	3.90s	5.67s	19.9s	151.2s
	equi-dstein	5.19s	5.20s	5.22s	5.47s	9.31s	36.4s	211.5s

Table 4. Execution time and speedup rate (in brackets) on the SR2201

eps	4 nodes	8 nodes	16 nodes	32 nodes	64 nodes
1.0e-2	85.8 s (1.00)	70.6 s (1.22)	59.3 s (1.45)	55.0 s (1.56)	55.0 s (1.56)
1.0e-4	16.3 s (1.00)	9.4 s (1.73)	6.2 s (2.63)	6.5 s (2.50)	5.8 s (2.81)
1.0e-6	15.9 s (1.00)	9.0 s (1.77)	5.7 s (2.80)	4.4 s (3.61)	4.0 s (4.00)

(When 4 nodes, the speedup rate is 1.0.)

Table 5. Execution time and accuracy result for Frank matrices on the SR8000

No. of nodes		1	4	16
N=8000	total	440.2 s	120.6 s	46.47 s
acu.	trid.	130.0 s	41.4 s	20.9 s
	bisec.	60.6 s	15.2 s	3.9 s
r:1.47e-8	m-inv.	87.7 s	22.3 s	9.3 s
o:1.04e-10	back.	161.9 s	41.8 s	12.2 s

No. of nodes		1	4	16
N=16000	total	2857.4 s	740.0 s	227.5 s
acu.	trid.	983.1 s	266.5 s	89.9 s
	bisec.	242.3 s	60.6 s	15.2 s
r:1.38e-7	m-inv.	354.1 s	90.1 s	34.0 s
o:2.62e-10	back.	1277.8 s	322.7 s	88.4 s

5. J. Demmel, I. Dhillon, and H. Ren: On the correctness of some bisection-like parallel eigenvalue algorithms in floating point arithmetic, Electronic Trans. Numer. Anal. 3, 116-149, Dec. 1995.
6. J. Choi, J. Demmel, I. Dhillon, J. Dongarra, S. Ostrouchov, A. Petitet, K. Stanley, D. Walker, R.C. Whaley : LAPACK Working Note 95, ScaLAPACK: A Potable Linear Algebra Library for Distributed Memory Computers - Design Issues and Performance, 1995.
7. I. S. Dhillon: A New O(n^2) Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem, Ph.D. thesis, Computer Science Division, University of California, Berkeley, May 1997.
8. LAPACK: http://www.netlib.org/lapack
9. ScaLAPACK: http://www.netlib.org/scalapack
10. K. Naono, M. Igai, Y. Yamamoto: Development of a Parallel Eigensolver and its Evaluation, Proceedings of the Joint Symposium on Parallel Processing 1996, Waseda, Japan, pp9-16, 1996(in Japanese).
11. G. Peters and J.H. Wilkinson : The Calculation of Specified Eigenvectors by Inverse Iteration pp.418-439, in book 'Linear Algebra' edited by J.H.Wilkinson and C.Reinsch, Springer-Verlag, 1971.
12. B. Parlett : The symmetric Eigenvalue Problem, Prentice Hall, Englewood Cliffs, NJ, 1980.