2008－MPS－70　(9)
2008／7／14

Evaluations of Parallel double Divide and Conquer on a 16–core computer

Taro Konda, Hiroki Toyokawa and Yoshimasa Nakamura†‡
† Department of Applied Mathematics and Physics,
Graduate School of Informatics,
Kyoto University
Yoshida Honmachi, Sakyo-ku, Kyoto, JAPAN
‡ SORST, JST

Abstract ***For bidiagonal SVD, double Divide and Conquer was proposed. It first computes singular values by a compact version of Divide and Conquer. The corresponding singular vectors are then computed by twisted factorization. The speed and accuracy of double Divide and Conquer are as good or even better than standard algorithms such as QR and the original Divide and Conquer. Moreover, it shows high scalability even on a PC cluster, distributed memory architecture. This paper presents evaluations of parallel double Divide and Conquer for singular value decomposition on a 16–core architecture.***

1 Introduction

A new algorithm, double Divide and Conquer(dDC), improves the parallelism of algorithm with twisted factorization for SVD[1] and EVD[2]. It adopts Divide and Conquer(D&C) to parallelize the section of singular/eigen value computation. It is as fast and accurate as I–SVD. Moreover, high scalability on a PC Cluster is shown[3].

Recently, multi/many–core computers are distributed to the market. This architecture is not only the way to speed-up the computer but also power-effective approach. We can expect that multi/many–core architecture will become common year by year in high performance computing area. One of the features of this architecture related the performance of parallel program is shared cache structure. Some/all cores on the same CPU shares same cache memory. High performance will be expected when we use this share cache structure effectively. In this paper, we evaluate parallelism of dDC for SVD on a 16–core computer with 4 Quad–core AMD Opterons, in some matrix sizes and types.

2 Singular Value Decomposition

An arbitrary $m \times n$ rectangular matrix A is converted to an upper bidiagonal matrix B by Houshölder transformation[4, 5] as follows.

$$A = \hat{U} \begin{pmatrix} B \\ 0 \end{pmatrix} \hat{V}^T \text{ or } \hat{U} \begin{pmatrix} B & 0 \end{pmatrix} \hat{V}^T, \qquad (1)$$

where $\hat{U}$ and $\hat{V}$ are suitable orthogonal matrices. This paper is mainly concerned with SVD of the bidiagonal matrices B.

Let T_s be a positive definite symmetric matrix that consists of a $n \times n$ bidiagonal matrix B such that $T_s = B^T B$. And let λ_k $(k = 1, \cdots, n)$ be eigenvalues of the matrix T_s. Then σ_k, singular values of B, is given by $\sigma_k = \sqrt{\lambda_k}$ $(k = 1, \cdots, n)$.

3 double Divide and Conquer

The double Divide and Conquer(dDC) for singular value first computes singular values by a "compact" Divide and Conquer, then, it computes the corresponding singular vectors by twisted factorization described in section 3.2.

3.1 Divide and Conquer for the Computation of Singular Values

Given an $n \times (n+1)$ upper bidiagonal matrix

$$B = \begin{pmatrix} b_1 & b_2 & & & \\ & b_3 & \ddots & & \\ & & \ddots & b_{2n-2} & \\ & & & b_{2n-1} & b_{2n} \end{pmatrix}, \qquad (2)$$

its SVD is

$$B = U \begin{pmatrix} \Sigma & 0 \end{pmatrix} V^T, \tag{3}$$

where U is an $n \times n$ orthogonal matrix whose columns are left singular vectors. V is an $(n+1) \times (n+1)$ orthogonal matrix whose columns are right singular vectors. Σ is an $n \times n$ nonnegative definite diagonal matrix and 0 is a column of zero elements.

The $n \times (n+1)$ upper bidiagonal matrix B is partitioned into two submatrices as

$$B = \begin{pmatrix} B_1 & 0 \\ b_{2k-1}\mathbf{e}_k{}^T & b_{2k}\mathbf{e}_1{}^T \\ 0 & B_2 \end{pmatrix} \tag{4}$$

for a fixed k such that $1 < k < n$, where B_1 is a $(k-1) \times k$ lower bidiagonal matrix, B_2 is an $(n-k)\times(n-k+1)$ upper bidiagonal matrix and $\mathbf{e}_j$ is the j–th unit vector of appropriate dimension. The parameter k is usually taken to be $\lfloor n/2 \rfloor$.

Now, suppose that SVD of the B_i is given by

$$B_i = U_i \,(D_i\ 0)\,(V_i\ \mathbf{v}_i)^T. \tag{5}$$

Let $\mathbf{l}_1$ be the last row of V_1, ψ_1 be the last element of $\mathbf{v}_1$, $\mathbf{f}_2$ be the first row of V_2 and ϕ_2 be the first element of $\mathbf{v}_2$. By substituting (5) into (4), we then obtain

$$B = \begin{pmatrix} 0 & U_1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & U_2 \end{pmatrix} \begin{pmatrix} b_{2k-1}\psi_1 & b_{2k-1}\mathbf{l}_1 & b_{2k}\mathbf{f}_2 & b_{2k}\phi_2 \\ 0 & D_1 & 0 & 0 \\ 0 & 0 & D_2 & 0 \end{pmatrix}$$
$$\times \begin{pmatrix} \mathbf{v}_1 & V_1 & 0 & 0 \\ 0 & 0 & V_2 & \mathbf{v}_2 \end{pmatrix}^T. \tag{6}$$

If Givens rotation is applied to make $b_{2k}\phi_2$ zero, then we get

$$B = \tilde{U} \begin{pmatrix} M & 0 \end{pmatrix} \begin{pmatrix} \tilde{V} & \tilde{\mathbf{v}} \end{pmatrix}^T, \tag{7}$$

where

$$\begin{aligned} \tilde{U} &\equiv \begin{pmatrix} 0 & U_1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & U_2 \end{pmatrix}, \\ M &\equiv \begin{pmatrix} r_0 & b_{2k-1}\mathbf{l}_1 & b_{2k}\mathbf{f}_2 \\ 0 & D_1 & 0 \\ 0 & 0 & D_2 \end{pmatrix}, \\ \tilde{V} &\equiv \begin{pmatrix} c_0\mathbf{v}_1 & V_1 & 0 \\ s_0\mathbf{v}_2 & 0 & V_2 \end{pmatrix}, \quad \tilde{\mathbf{v}} \equiv \begin{pmatrix} -s_0\mathbf{v}_1 \\ c_0\mathbf{v}_2 \end{pmatrix}, \\ r_0 &= \sqrt{(b_{2k-1}\psi_1)^2 + (b_{2k}\phi_2)^2}, \\ c_0 &= \frac{b_{2k-1}\psi_1}{r_0}, \quad s_0 = \frac{b_{2k}\phi_2}{r_0}. \end{aligned} \tag{8}$$

Thus the matrix B is reduced to $\begin{pmatrix} M & 0 \end{pmatrix}$ by the orthogonal transformations $\tilde{U}$ and $\begin{pmatrix} \tilde{V} & \tilde{\mathbf{v}} \end{pmatrix}$ [6].

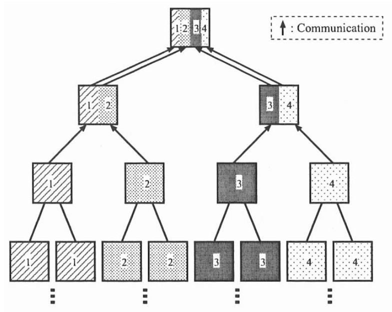

Figure 1: Parallel model of SVDC ($P = 4$).

The above D&C process can be simplified when only singular values are desired. From (7) and (8), B is written as

$$\begin{aligned} B &= \tilde{U} \begin{pmatrix} M & 0 \end{pmatrix} \begin{pmatrix} \tilde{V} & \tilde{\mathbf{v}} \end{pmatrix}^T \\ &= \tilde{U} \begin{pmatrix} U_M \Sigma V_M^T & 0 \end{pmatrix} \begin{pmatrix} \tilde{V} & \tilde{\mathbf{v}} \end{pmatrix}^T \\ &= U\Sigma \begin{pmatrix} \tilde{V} V_M & \tilde{\mathbf{v}} \end{pmatrix}^T \\ &= U\Sigma \left(\begin{pmatrix} c_0\mathbf{v}_1 & V_1 & 0 \\ s_0\mathbf{v}_2 & 0 & V_2 \end{pmatrix} V_M \quad \begin{pmatrix} -s_0\mathbf{v}_1 \\ c_0\mathbf{v}_2 \end{pmatrix} \right)^T, \end{aligned} \tag{9}$$

thus

$$\begin{aligned} f &= \begin{pmatrix} c_o\phi_1 & f_1 & 0 \end{pmatrix} V_M, \\ l &= \begin{pmatrix} s_o\psi_2 & 0 & l_2 \end{pmatrix} V_M, \\ \phi &= -s_0\phi_1, \quad \psi = c_0\psi_2, \end{aligned} \tag{10}$$

where $\mathbf{f}_1$ is the first row of V_1, ϕ_1 is the first element of $\mathbf{v}_1$, $\mathbf{l}_2$ is the last row of V_2, ψ_2 is the last element of $\mathbf{v}_2$, $\mathbf{l}$ is the last row of V, ψ is the last element of $\mathbf{v}$, $\mathbf{f}$ is the first row of V and ϕ is the first element of $\mathbf{v}$.

Because most of the running time of D&C is consumed for vector update during singular vector computation, SVDC is wholly faster than the normal one.

3.2 Twisted Factorization for the Computation of Singular Vectors

Let us consider the following system

$$(T_s - \hat{\lambda} I)\mathbf{x}^{(\mathbf{k})} = \mathbf{e}_k \gamma_k, \tag{11}$$

where $\mathbf{e}_k$ is the k–th unit vector of appropriate dimension, the k–th element of $\mathbf{e}^{(\mathbf{k})}$ is 1 and γ_k is a residual norm of the k–th equation. Suppose that L is a lower bidiagonal matrix and U is an upper bidiagonal matrix such that

$$\begin{aligned} T_s - \hat{\lambda} I &= L D^{+} L^T \\ &= U D^{-} U^T, \end{aligned} \tag{12}$$

Table 1: Specification of test bed.

CPU	Quad AMD Opteron 1.8GHz
(Cache)	(Shared 2MB)
#Cores/CPU	4
#CPUs	4
#Cores	16
Memory	16GB
OS	openSuSE Linux
(kernel)	(Linux 2.6.22.12–0.1–default)
Compiler	gfortran 4.2.1

where L and U are given by Choleskey decomposition of $T_s - \hat{\lambda}I$ and let the i–th element of the diagonal matrix D^+ and D^- be D_i^+ and D_i^-, and i–th subdiagonal element of L and U be L_i and U_i, respectively, then, factorizations LD^+L^T and UD^-U^T are respectively computed by the algorithms *stdLVvs* and *rdLVvs* [7].

Therefore, the twisted factorization is described as follows.

$$T_s - \hat{\lambda}I = N_k D_k N_k^T, \tag{13}$$

where N_k is the twist factor[8, 9] defined as

$$N_k = \begin{pmatrix} 1 & & & & & & \\ L_1 & 1 & & & & & \\ & \ddots & \ddots & & & & \\ & & L_{k-1} & 1 & U_k & & \\ & & & & \ddots & \ddots & \\ & & & & & 1 & U_{n-1} \\ & & & & & & 1 \end{pmatrix},$$

factoring the matrix from top to bottom and from bottom to top to meet at row k. And D_k is a diagonal matrix such that

$$D_k = \mathrm{diag}\left(D_1^+, ..., D_{k-1}^+, \gamma, D_{k+1}^-, ..., D_n^-\right). \tag{14}$$

When $\hat{\lambda}$ is close to eigenvalues of T_s, singularity of $T_s - \hat{\lambda}I$ is revealed by choosing $\gamma = \arg\min_k |\gamma_k|$.

3.3 Parallel double Divide and Conquer

Here, our concern is a parallel version of dDC. Each piece of twisted factorization can be executed independently. Moreover, each of them has almost equal amount of loading. Thus, we parallelize SVDC more carefully. The original D&C is parallelized naively because a large amount of matrix–matrix products is operated. But SVDC of dDC skips this operation. Figure 1 shows our strategy for parallel SVDC. This is a tree structure of SVDC executed by 4 processors. A leaf of the tree represents a submatrix. A node is a merger process of two submatrices. A number on leaves and nodes shows processor number to compute SVD of a submatrix and merge submatrices, respectively. A line is process transition. In particular, an arrowed line between boxes (leaves and nodes) means communication between processors.

After submatrices are assigned evenly to all processors and the corresponding SVDs are computed, submatrices are merged. The merger process is parallelized when the height becomes more than P, the number of processors. Data Σ_M, $\mathbf{f}$, $\mathbf{l}$, ϕ and ψ are sent and shared by responsible processors. This process continues until the root node is computed. After that, the computed singular values are broadcasted to every processor. Finally, twisted factorization is invoked parallelly.

With multi–core machine, these data (Σ_M, $\mathbf{f}$, $\mathbf{l}$, ϕ and ψ) will be shared in only the shared–memory but shared–cache successfully, although they are formally sent between cores.

4 Numerical Experiments

In this section, we evaluate parallelism of dDC. The following is test matrix.

- **Type 1:** All diagonal elements are 2.001 and all subdiagonal elements are 2.0. All singular values are separated to each other. Deflation of dDC seldom occurs (approximately 10% of dimension n).

Here, dDC uses QR to solve submatrices. After singular vectors are computed by twisted factorization, dDC invokes inverse iteration to refine orthogonality of these vectors. The parallel dDC is implemented with MPI[10] in the same way as on PC–cluster. Table 2 shows specification of the test bed, a multi–core computer with 4 Quad AMD Opterons with 16GB memory.

Table 2 shows the best times in second of 10 executions of the type 1 matrix. The number of cores ranges from 1 to 16. Dimensions of tested matrices are n=10,000, n=25,000, n=50,000 and n=75,000. Parallel speed-up S_P and parallel efficiency $E_P = S_P/P$ are also calculated. For n=25,000, E_Ps are more than 1.0 with 2, 4 and 8 cores. These are highly effective parallelization called super linear. It is also observed with 2 cores for n=50,000 and n=75,000. Parallel dDC utilize shared cache memory of multi–core architecture.

Efficiencies of n=25,000 are better than those of n=10,000. But, those of n=50,000 and n=75,000 are worse. It is unusual because ordinal parallel program becomes more effective when the problem size gets increased. It seems more likely

Table 2: Timing, speed–up and parallel ratio of Parallel dDC (Matrix: Type 1).

#Cores	1	2	4	8	16
n=10,000	63.40	32.11	16.62	8.59	4.58
S_P	1.00	1.97	3.81	7.38	13.84
E_P	1.00	0.99	0.95	0.92	0.87
n=25,000	445.60	212.00	105.34	55.30	29.81
S_P	1.00	2.10	4.23	8.06	14.95
E_P	1.00	1.05	1.06	1.01	0.93
n=50,000	1695.32	838.16	428.34	230.40	133.42
S_P	1.00	2.02	3.96	7.36	12.71
E_P	1.00	1.01	0.99	0.92	0.79
n=75,000	3922.11	1955.45	996.39	540.79	328.92
S_P	1.00	2.01	3.94	7.25	11.92
E_P	1.00	1.00	0.98	0.91	0.75

in second: [s]

S_P: Speed–up ratio by P processors

E_P: Parallel Efficiency by P processors

that transfer data size exceed cache size for this large matrix.

5 Conclusions

In this paper, we presented some evaluations of parallel double Divide and Conquer with MPI for bidiagonal SVD on a 16–core computer. We can expect that multi/many–core architecture is not only the way to speed-up the computer but also power-effective approach. Parallel dDC showed high scalability for various matrix sizes. Super linear efficiency is observed for some cases because of good use of shared cache memory of multi–core architecture. However, the efficiency is declined when the matrix is too large for the cache memory. We can say that dDC parallelized with MPI works well on multi/many–core architectures.

References

[1] T. Konda, H. Tsuboi, M. Takata, M. Iwasaki, and Y. Nakamura. Preliminary result of parallel double divide and conquer. *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '06)*, pages 888–894, 2006.

[2] H. Tsuboi, T. Konda, M. Takata, K. Kimura, M. Iwasaki, and Y. Nakamura. Evaluation of a new eigen decomposition algorithm for symmetric tridiagonal matrices. *Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '06)*, pages 832–838, 2006.

[3] T. Konda, H. Tsuboi, M. Takata, M. Iwasaki, and Y. Nakamura. Parallelism of double divide and conquer algorithm for singular value decomposition. *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks*, 2007.

[4] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numeri. Anal.*, 2(2):205–224, 1965.

[5] J. Demmel. *Applied Numerical Linear Algebra.* SIAM, 1997.

[6] M. Gu and S.C. Eisenstat. A divide–and–conquer algorithm for the bidiagonal svd. *SIAM J. Matrix Anal. Appl.*, 16(1):79–92, 1995.

[7] M. Iwasaki, S. Sakano, and Y. Nakamura. Accurate twisted factorization of real symmetric tridiagonal matrices and its application to singular value decomposition. *Trans. Japan. Soc. Indust. Appl. Math.*, 15(3):461–481, 2005.

[8] K. Fernando. On computing an eigenvector of a tridiagonal matrix. part 1: basic results. *SIAM J. Matrix. Anal. Appl.*, 18(4):1013–1034, 1997.

[9] B. Parlett and I. Dhillon. Fernando's solution to wilkinson's problem: An application of double factorization. *Lin. Alg. Appl.*, 267:247–279, 1997.

[10] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI Second Edition.* The MIT Press, 1999.