Arbenz, P., Slapničar, I.

On an Implementation of a One-sided Block Jacobi Method on a Distributed Memory Computer

We show that one-sided block Jacobi method for computing the singular value decomposition can be implemented on distributed memory computers with message passing communication model in a highly scalable manner. The algorithm is highly accurate and can also be used to compute the eigenvalue decomposition of a symmetric matrix.

1. The one-sided block Jacobi method

The SVD of a non-singular $n \times n$ matrix A is written as $A = U\Sigma V^T$, where U and V are orthogonal matrices, and Σ is a diagonal matrix whose diagonal entries are the singular values of A. One-sided (implicit) Jacobi method [2] computes the singular values, and the corresponding left singular vectors (the matrix U), by forming a sequence of matrices $A_0 = A$, $A_{k+1} = A_k R_k$, where R_k is a plane rotation chosen to annihilate the element (i,j), $i<j$, of the implicitly defined Gram matrix $G_{k+1} \equiv A_{k+1}^T A_{k+1} = R_k^T A_k^T A_k R_k$. To determine R_k we need to know $[G_k]_{ii}$, $[G_k]_{jj}$, and $[G_k]_{ij} = [A_k]_{:i}^T [A_k]_{:j}$. The former two quantities are are kept and updated in a separate vector, the latter is computed by a scalar product. The sequence A_k converges to the matrix $\bar{A}$, where $\Sigma_{ii} = \|\bar{A}_{:i}\|_2$ and $U_{:i} = \bar{A}_{:i}\ \Sigma_{ii}$. Thus, the initial A is overwritten by singular vectors, which reduces the storage. We use the relative stopping criterion [2]: we rotate only if $|[G_k]_{ij}| > n \cdot \mathit{macheps} \cdot \sqrt{[G_k]_{ii}}\sqrt{[G_k]_{jj}}$, and we stop after an empty cycle.

Since the rotations on non-overlapping index pairs are completely independent, they can be applied simultaneously, so the method is ideal for parallel computation. Modern processors with pipelining features prefer block algorithms rich with matrix multiplications like the BLAS-3 routine `dgemm`. We illustrate the performance of `dgemm` for $(2048 \times m) \times (m \times m)$ matrix multiplication on the Intel i860 RISC processor with peak performance of 50 Mflop/s:

m	1	2	4	8	16	32	64	128	256	512
Mflop/s	11.7	14.8	26.6	34.5	40.5	43.0	44.7	45.3	45.8	45.7

To exploit this property we use a block Jacobi method similar to the one from [1]: A is partitioned into $2p$ column blocks $A = \begin{bmatrix} A^{(1)} & A^{(2)} & \cdots & A^{(2p)} \end{bmatrix}$. Two consecutive blocks are assigned to each processor $i = 0, 1, \ldots, p-1$, and denoted by A_L and A_R. Each processor forms the upper triangle of the Gram matrix of its block columns,

$$G \equiv \begin{bmatrix} G_{11} & G_{12} \\ G_{12}^T & G_{22} \end{bmatrix} = \begin{bmatrix} A_L^T A_L & A_L^T A_R \\ A_R^T A_L & A_R^T A_R \end{bmatrix},$$

applies one sweep of the standard two-sided Jacobi method [5] to G accumulating the rotations in the matrix R, and, finally, multiplies $\begin{bmatrix} A_L & A_R \end{bmatrix} R$. The last step and the forming of G_{12} are performed by `dgemm`. After computation, the blocks A_R circulate along the processor ring according to the caterpillar parallel cyclic strategy [3]. This strategy restores the original layout after every two cycles ($4p$ stages). The algorithm is given in Algorithm 1. Only those columns of U which correspond to non-zero singular values are computed. The convergence is guaranteed since the stopping criterion is at the same time a threshold [5, p. 278]. This criterion also implies high relative accuracy of the singular values computed by the non-blocked version of the algorithm [2]. Numerical tests indicate the same property for our algorithm, although we have not yet a formal proof. The algorithm for an $m \times n$, $m \geq n$ matrix is similar. Since the SVD of A and the eigenvalue decomposition of $A^T A$ are simply related, the algorithm can be easily used to compute the eigenvalue decomposition of a symmetric matrix A (see also [2]).

2. The implementation

If the execution of one flop takes $\phi(n,m)$ μs, where $m = n/(2p)$, and the transmission of k doubles ($8k$ bytes) takes $(\sigma + \tau k)$ μs, where σ is the startup time and $1/\tau$ is the bandwidth, then the cost of Algorithm 1 in μs is

$$\begin{aligned} C(n,p) &= (2m^2+m)(2n-1)\phi(n,m) + 8 \cdot 4p\left[6(2m)^3\phi(m,2) + n(2m)(4m-1)\phi(n,2m)\right. \\ &\quad \left. + m^2(2n-1)\phi(n,m) + 4\sigma + 2\tau m(m+1)/2 + 2\tau nm\right]. \end{aligned} \tag{1}$$

Algorithm 1. *(Block-Jacobi method.) Processor i executes the following program:*

```
$G_{11} \leftarrow A_L^T A_L$, $G_{22} \leftarrow A_R^T A_R$, $G_{12} \leftarrow A_L^T A_R$.
repeat until convergence
  for $j = 1, \ldots, 4p$
    if $(j-1) \bmod 2p+1 \neq 2p-2(i-1)$ then
      apply one sweep of Jacobi to $G$ and compute $R$.
      $[\, A_L \quad A_R \,] \leftarrow [\, A_L \quad A_R \,]\, R$.
      Send $A_R$ and $G_{22}$ to next processor, and receive $A_R$ and $G_{22}$ from previous processor.
      $G_{12} \leftarrow A_L^T A_R$.
    else
      Swap $A_L$ with $A_R$, and $G_{11}$ with $G_{22}$.
      Send $A_R$ and $G_{22}$ to next processor, and receive $A_R$ and $G_{22}$ from previous processor.
      $G_{12} \leftarrow A_L^T A_R$.
    end if
  end for
end repeat
```

Here we have used the fact that for larger dimensions ($n = 1000, 2000$) the algorithm converges after approximately 8 double sweeps. The speedup is defined as $S(n,p) = C(n,1)/C(n,p)$. The computational overhead (25 %) versus the non-blocked version is more than compensated by using `dgemm` on larger matrices. If computation and communication can be performed simultaneously (asynchronous communication), then the communication (except startup) can be overlapped by computation by modifying Algorithm 1 as follows: G_{22} is communicated while upper half of $[\, A_L \quad A_R \,]$ is multiplied by R; upper half of A_R is communicated while lower half of $[\, A_L \quad A_R \,]$ is multiplied by R; lower half of A_R is communicated while forming of the new G_{12} starts on the upper half of $[\, A_L \quad A_R \,]$, etc. In this case the cost is given by the maximum between computation with startup and the τ part of (1).

We implemented our algorithms on the Intel Paragon with 96 nodes and the NX message passing system [4] located at ETH Zürich. Each node consists of two i860 processors, the compute and the message processor, thus enabling asynchronous communication. The values for Paragon are $\sigma = 65$, $\tau = 1/8$, while $\phi(n,m)$ is computed from the table of Section 1. For $n = 2048$ the theoretical and measured speedups for Algorithm 1 and its asynchronous version are as follows:

Number of processors	4	8	16	32	64	96
Theoretical – Asynchronous	4	8	16	32	63.8	95.3
Theoretical – Synchronous	3.99	7.97	15.8	31.1	60.2	87.4
Measured – Asynchronous				29.5	49.1	54.7
Measured – Synchronous				28.4	47.5	52.3

For orientation, the asynchronous version takes 630, 342, and 300 seconds for $p = 32, 64, 96$, respectively.

As a conclusion, let us say that our theoretical considerations show that the one-sided block Jacobi method attains nearly optimal speedups, and is therefore highly attractive for parallel computations. We suspect that the relative slowdown for the larger number of processors and the relatively low speedup obtained for the asynchronous version are due to the facts that the NX message passing system is not yet optimally implemented, and the synchronization of the memory accesses of both compute and message processor working at full speed may cause problems.

3. References

1 Arbenz, P., Oettli, M.: Block implementations of the symmetric QR and Jacobi algorithms. Technical Report 178. Institute for Scientific Computing, ETH Zürich, 1992.

2 Demmel, J. W., Veselić, K.: Jacobi's method is more accurate than QR. SIAM J. Matrix Anal. Appl. **13** (1992), 1204–1243.

3 Luk, F. T., Park, H.: On parallel Jacobi orderings. SIAM J. Sci. Statist. Comput. **10** (1989), 18–26.

4 Pierce, P.: The NX message passing system. Parallel Computing **20** (1994), 463–480.

5 Wilkinson, J. H.: The Algebraic Eigenvalue Problem. Oxford University Press, 1965.

Addresses: Peter Arbenz, ETH Zürich, Institut für Wissenschaftiches Rechnen, 8092 Zürich, Switzerland
Ivan Slapničar, University of Split, Faculty of Electrical Engineering, Mechanical Engineering, and Naval Architecture, R. Boškovića b.b., 58000 Split, Croatia