Portable Parallelization of the Iterative Linear Solver Package LINSOL

Hartmut Häfner and Willi Schönauer

Computing Center, University of Karlsruhe, Postfach 6980, D-76128 Karlsruhe, Germany, e-mail: haefner@rz.uni-karlsruhe.de

Keywords: parallelization, sparse matrix-vector multiplication, message passing, MPI (Message Passing Interface)

At the Computing Center of the University of Karlsruhe 'black-box' software packages for the solution of nonlinear systems of elliptic and parabolic PDEs have been developed. All three packages FIDISOL [1], CADSOL [7] (both solving PDEs by the finite difference method) and VECFEM [6] (solving PDEs by a finite element method) use the LINSOL program package solving large (and sparse) linear systems of equations.

LINSOL iteratively solves linear systems of equations

$$Ax = b \text{ , with } r_k = Ax_k - b.$$

Here x_k denotes the iterate and r_k the residual in the k-th step. Different representants of 'CG-type' methods [5] like PRES20, BCG, BICO, QMR, CGNE and CGNR (ATPRES) are included in LINSOL. To compute x_k and r_k from x_{k-1} and r_{k-1} one or two matrix-vector multiplications (MVM) and a few linked triads and scalar products have to be calculated in all these methods. The most time consuming operation is the MVM and thus it is the key operation for the solution process and also for the parallelization; furthermore, the optimal parallelization of the MVM [2] and the scalability of the whole program on a parallel computer determine the internal data structures of the program package. Here the portable parallelization and the optimal internal data structures for vectorcomputers [1] and distributed memory parallel superscalar architectures become the focus of attention.

Two key problems emerge at the parallelization: the first one is the efficient implementation of the parallel matrix-vector multiplication; the second one is the portability of the code on parallel computers, which can be achieved at present only by the 'message passing' paradigm. The first problem is touched on the chapter 'Parallelization of LINSOL' where a few questions of parallelization concerning LINSOL are briefly discussed, and the second one on the chapter 'Portable Message Passing Interface' [4], which shows a way towards portable parallel programming.

1. Parallelization of LINSOL

All vectors of LINSOL like x_k and r_k are distributed in vector parts of the same size onto the p processors. To minimize communication, i.e. to hold the matrix A locally on the single processors, the matrix A is distributed in row blocks of the same size onto the p processors. Additionally the matrix A is subdivided into p logical blocks on each processor corresponding to the physical distribution.

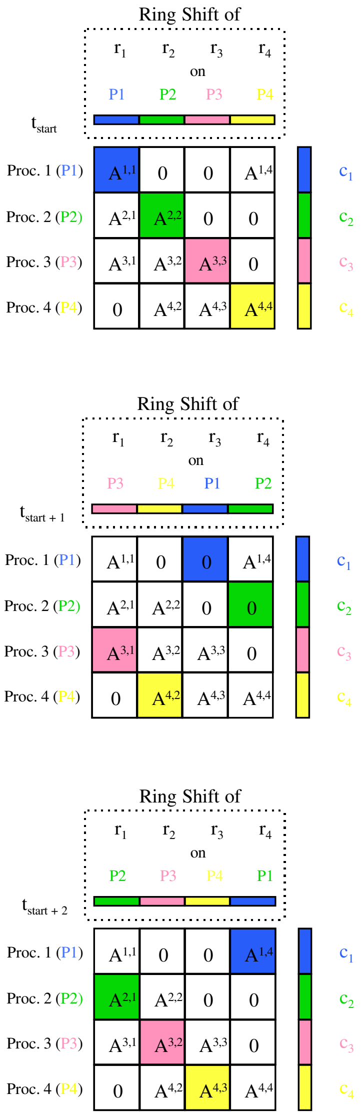

Figure 1: The parallelized matrix-vector multiplication on 4 processors

Thus the MVM $c = A * r$ is computed on each processor i $(i = 1, p)$ with a logical subdivision of the matrix A and a physical subdivision of the vector r in p blocks $(j = 1, p)$:

$$c^i = c^i + \sum_{j=1}^{p} A^{i,j} \cdot r^j.$$

In Fig. 1 the MVM is depicted on 4 processors. Diagonal matrix blocks (e.g. $A^{1,3}$, $A^{2,4}$, $A^{3,1}$, $A^{4,2}$) are computed concurrently, if at least one of these blocks contains non-zero matrix elements. If all diagonal matrix blocks contain no non-zero matrix elements (e.g. in Fig. 1 $A^{1,2}$, $A^{2,3}$, $A^{3,4}$, $A^{4,1}$), the computation of all these diagonal matrix blocks can be skipped, i.e. the number of communication cycles is reduced by one.

This strategy of parallelization of the MVM leads to two-fold scalability:

- we get scalability regarding the memory and
- scalability regarding the computation time (see also section 3).

As all vectors are distributed onto the p processors, scalar products are computed via a cascade over all processors and immediately broadcasted to all processors.

2. Portable Message Passing Interface

Designing parallel programs for MIMD computers means that you have to choose between one of the two following programming models:

- 'message passing' paradigm,
- '(virtual) shared memory' programming model.

At present the only possible choice is 'message passing', if portable programs should be implemented. All manufacturers of MIMD computers offer a 'message passing' library, but the problem is that these libraries have no common subset of basic 'message passing' routines. There are two ways to solve this problem.
The first one is to use PVM, PARMACS or MPI. On nearly all MIMD computers PVM, PARMACS and MPI run, but often the performance of proprietary 'message passing' libraries like NX/2 from Intel or MPL/P from IBM is much better than the performance of PVM (currently we have no know-how regarding MPI), i.e. that the theoretically possible performance rates of the communication networks mostly (sometimes) can not be exploited by PVM, and on many parallel computers PARMACS has not been installed.
The second one - which has been chosen by us - is to implement an own Portable Message Passing Interface (P_MPI). In this interface only six basic 'message passing' routines (send and receive asynchronously, send-wait, receive-wait, start-of-communication and end-of-communication) are considered. As send- and receive-routines are part of all 'message passing' libraries, we only have to change a few calls inside of the send- and receive-routines of P_MPI and to set up a few communication parameters like e.g. a table for the logical processor numbers and the according physical processor numbers within the start-of-communication routine to obtain first a portable code and second high performance rates (concerning the communication network) for all MIMD computers.

3. Measurements

We have measured the scalability of the program package LINSOL on many parallel computers. Here we show the scalability of LINSOL regarding the computation time on the INTEL Paragon in Fig. 2. There are two definitions for scalability; we use: a program is scalable regarding the computation time, if the computation time for n unknowns on a single processor is about the computation time for $n * p$ unknowns on p processors. The second (unused) definition is: a program is scalable regarding the computation time, if the computation time for n unknowns on a single processor is about a p-th of the computation time for the same number of unknowns on p processors.

In Fig. 2 NxNxN means that a unstructured FEM-grid with a variing number of grid points in all three dimensions is used; NxNxN/op means that the same grid is used, but the matrix now is bandwidth optimized; 11x11xN/op means that the number of grid points in the first two dimensions are constant (=11) and only the number of grid points in the third dimension is increasing with the number of the processors and the matrix is bandwidth optimized.

Figure 2: The computation time of one matrix-vector multiplication of LINSOL with the same number of unknowns on each processor for a 3-dimensional problem.

The case NxNxN leads to a slightly increasing computation time of one MVM for an increasing number of processors, because the bandwidth of the matrix, i.e. the number of non-zero diagonal matrix blocks, and thus the number of communication cycles between the computation of these blocks is increasing for an increasing number of processors. In the third case (11x11xN/op) the bandwidth of the matrix stays constant for an increasing number of processors and thus the number of communication cycles does not increase. Then the computation time of one MVM does not increase, i.e. LINSOL is scalable regarding the computation time.

References

1. W. Schönauer, Scientific Computing on Vector Computers, North-Holland, Amsterdam, 1987.
2. R. Weiss, C. Roll, H. Häfner, L. Grosz, M. Schmauder, W. Schönauer, Parallelisierung von ODIN-Software fuer iterative Gleichungslöser, in A. Schreiner, E. Schnepf (Eds.), Drittes ODIN-Symposium, Rechenzentrum Universität Karlsruhe, 1994, pp. 117-128.
3. W. Schönauer, H. Häfner, L. Grosz, M. Schmauder and R. Weiss, Parallelization of Black-Box Software for Elliptic and Parabolic PDEs, submitted to publication.
4. H. Häfner, W. Schönauer, Kommunikation auf verschiedenen Parallelrechnern, Auswirkung auf die Parallelisierung, in A. Schreiner, E. Schnepf (Eds.), Drittes ODIN-Symposium, Rechenzentrum Universität Karlsruhe, 1994, pp. 195-206.
5. R. Weiss, Convergence behaviour of generalized conjugate gradient methods, Internal report 43/90, University of Karlsruhe, Computing Center, 1990.
6. L. Grosz, C. Roll, W. Schönauer, Nonlinear finite element problems on parallel computers, in J. Dongarra, J. Wasniewski (Eds.), Parallel Scientific Computing, 1. Internat. Workshop PARA'94, Lecture Notes in Computer Science, Springer Verlag, Berlin 1994, pp. 247-261.
7. M. Schmauder, W. Schönauer, CADSOL - A fully vectorized black-box solver for 2-D and 3-D partial differential equations, in R. Vichnevetsky, D. Knight, G. Richter (Eds.), Advances in Computer Methods for Partial Differential Equations - VII, IMACS, New Brunswick, NJ, 1992, pp. 639-645.