24th International Conference
ENGINEERING MECHANICS 2018
Svratka, Czech Republic, May 14 –17, 2018
Paper **#344,** pp. 109–112, doi: 10.21495/91-8-109

EVALUATION OF LINEAR SYSTEM EQUATIONS SOLVERS ON MULTICORE ARCHITECTURES

M. Bošanský [*], **B. Patzák** [**]

Abstract: *The aim of this paper is the evaluation of an existing parallel, linear equation solver to solve large-scale, sparse, non-symmetric systems of linear equations as a part of the finite element software. Traditional finite element solvers run in a serial mode and are limited by available resources. The parallel approach allows to take an advantage of the distributed memory to forming large system matrices and multiple processing units to achieve significant speedups. In this paper, we study differences between the sequential and parallel solutions. Parallel approach contributing shared and distributed memory is represented by SuperLU DIST solver. SuperLU DIST is primary based on distributed memory model with using Message Passing Interface. SuperLU DIST can be also used as the hybrid parallel model combining shared memory model or with graphical computing units.*
The performance of parallel solvers is tested on problems from solid mechanics. The two type of benchmark problems discuss in this paper and computations of this problems are based on OOFEM which is a free finite element code with object oriented architecture for solving mechanical, transport and fluid mechanics problems that operates on various platform. The efficiency of hybrid SuperLU solver is com pared to the efficiency of pure shared memory SuperLU solver and to distributed memory solver in PETSc.

Keywords: linear equation solver, parallelization, memory, communication, threads

1. Introduction

The past decades have witnessed exceptional advances in computational science and engineering that were powered by development and increases in the power and pervasiveness of computer and communications. Capitalization on those advances by developing techniques for modern computers enables solution of large and complex problems. The traditional serial computers permit run of simulation codes only sequentially and often have limited available resources represented by single processing unit and available memory, see (Patzak, 2010). Parallelization can significantly reduce the required solution time by more efficient use of modern hardware. Parallel processing is based on idea of the partitioning the work into the group of smaller task which can be solved simultaneously by use of multiple computing resources.

Parallel computers can be classified, for example, by type of memory architecture. The shared, distributed and hybrid memory systems exist. In the shared memory system, all computing units can access the same physical global address space. Communication between processing units via global memory is implicit and transparent and realized through the shared memory bus performance of which is often limiting factor for scalability with increasing number of processing units. In distributed memory systems, the memory is distributed to individual processing units and there is no global shared memory. The explicit communications between processing units is needed to establish data exchange for processor data exchange. It is usually the task of the programmer to explicitly define how and when data is communicated. The cost of communication compared to shared memory model can be very high, on the other hand, the advantage is that the overall memory is scalable with increasing number of processors. Hybrid systems combine the features of shared and distributed memory systems, providing global, shared memory for reasonably small number of

[*] Michal Bošanský: Czech Technical University in Prague, Faculty of Civil Engineering, Thákurova 7, 166 29 Praha 6, Czech Republic, michal.bosansky@fsv.cvut.cz
[**] Bořek Patzák: Czech Technical University in Prague, Faculty of Civil Engineering, Thákurova 7, 166 29 Praha 6, Czech Republic

processing units that are combined into the distributed memory system. Somewhat different way in parallel computing is based on using graphics processing unit together with a traditional processors to accelerate the solution process. Graphics processing unit computes intensive portions of the solution while the remainder of the code still runs on the processor. Hybrid systems distribute the algorithm, inside each system, between multi-core processors and graphical processing units.

The design of parallel algorithms requires partitioning the problem into a set of tasks, that can be processed simultaneously. One of the most important goal in parallel computing is scalability of computation. The scalable parallel algorithm allows to achieve decreasing execution time by using increasing number of processing units, ideally in linear trend. The ideal scalability is difficult to obtain due to the overhead cost of parallel algorithm (synchronization and communication) and due the fact that some parts of the parallel computation are essentially sequential. Moreover, despite obtained speed-up, the parallel computing allows to solve large, complex problems that cannot be solve at single, well equipped machine.

SuperLU is a general purpose library for the direct solution of a large sparse non symmetric system of linear equations (Li, 1999). SuperLU library provides serial, multithreaded (shared-memory), and distributed memory version. The implementation and performance of the multithreaded version has been reported by the authors in (Bosansky, 2017a,b). In this paper, performance of the distributed version of SuperLU based on Message Passing Interface (MPI) (Pacheco, 1998) is evaluated and compared to other solvers. SuperLU DIST can be also used as the hybrid parallel model combining shared memory model with using OpenMP (Barney, 2014) or with graphical computing units (GPU) based on CUDA (Kloeckner, 2014)

Numerical solution of many engineering problems leads to solution of the sparse linear systems of equations. This is also the case of the Finite Element Method (FEM) which is based on converting system of partial differential equations to an algebraic system of equations using decomposition of problem domain into sub-domain (elements). In linear elasticity problem, the algebraic equations represent the discrete equilibrium equations at nodes of the computational mesh and have the following form

$$\mathbf{K} * \mathbf{r} = \mathbf{f}, \tag{1}$$

where $\mathbf{K}$ is the global stiffness matrix, $\mathbf{f}$ is global vector of external forces, and $\mathbf{r}$ is vector of unknown displacements. One of the key feature of the FEM is that the stiffness matrix is typically positive definite, symmetric matrix with sparse structure, which is a consequence of using interpolation and test functions with limited nonzero support. Effective FEM algorithm for solving linear system of equations should exploit the advantage of symmetry and sparse structure to lower the system memory and CPU requirements. There exist number of different storage schemes for sparse matrices. SuperLU library operates with compressed row and compressed column format of the sparse matrix on input (Pissanetzky, 1984).

The efficiency of direct SuperLU solver is compared Conjugate gradient solver from PETSc library (Balay, 1997).

2. Implementation

The comparison was made using the OOFEM (Patzak, 2000) finite element solver which has been extended to support the SuperLU distributed version. The interface to SuperLU DIST library consists of new classes to represent SuperLU DIST (direct solver based on distributed memory model) and sparse matrix distributed storage. The interfaces to SuperLU MT (direct solver based on shared memory) and PETSc (iterative solver based on distributed memory) already exists. The SuperLU driver advantageously uses symmetry of the matrix in the linear system. SuperLU MT ordering of the columns of matrix is optimized using multiple minimum degree algorithm applied to the structure of $A^T + A$. The sparse matrix and the right hand side vector are distributed among all the processes using the distribution based on block rows. That is, each process owns a block of consecutive rows of matrices. Distributed sparse matrix is stored in a compressed row format.

3. Results

The distributed memory programming model based on the solver SuperLU DIST have been evaluated using a benchmark problem of a 3D finite element model of a nuclear containment (Jete 250k) and a benchmark

problem of 3D finite element model of a porous microstructure (Micro 250k). The nuclear containment mesh consists of 87320 nodes and 959700 tetrahedral elements with linear interpolation. The total number of equations is 260322. The model of porous microstructure mesh consists of 85184 nodes and 79507 brick elements with linear interpolation. The total number of equations was 255552. Both structures were loaded with self-weight, resulting in a nonzero contribution of every element to the external load vector. Both benchmark problems are characterized by different sparsity of the system matrix. The model of porous microstructure has significantly more nonzeros members than the model of nuclear containment, which is a really sparse problem. Number of nonzeros members in the model of nuclear containment is $\sim 4,3 * 10^{-6}$ and number of nonzeros members in the model of porous microstructure is 1528773527. The individual approaches were tested on Linux workstation (running Ubuntu 14.04 OS) with the two CPU Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz and 126GB RAM. Each of the two CPU units consists of eight physical and sixteen logical cores, allowing up to thirty-two threads to run simultaneously on one workstation. All the tests fit into a system memory.

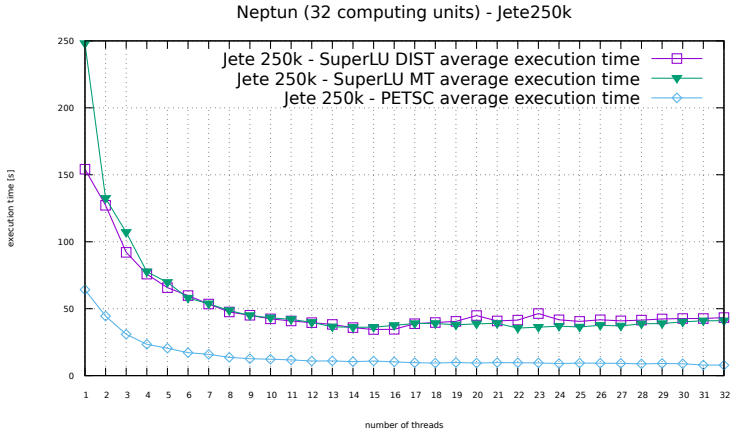

Fig. 1: Executions times of linear system solution of direct SuperLU MT (based on shared memory), direct SuperLU DIST (based on distributed memory) and iterative PETSc (based on distributed memory) solver.

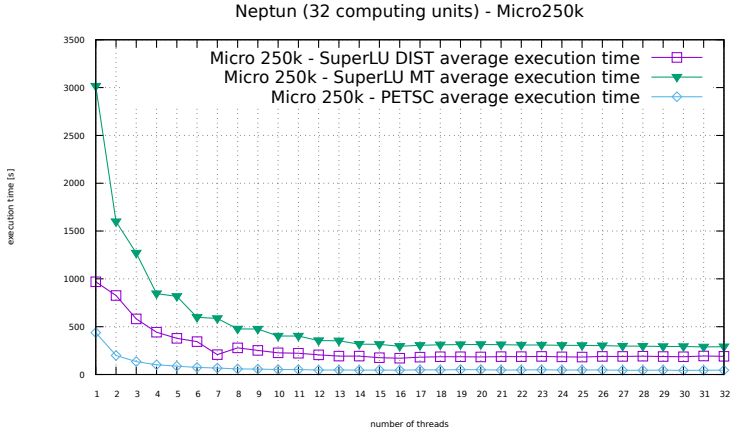

Fig. 2: Executions times of linear system solution of direct SuperLU MT (based on shared memory), direct SuperLU DIST (based on distributed memory) and iterative PETSc (based on distributed memory) solver.

Execution times to solve linear system of equations using the compared solvers are presented in Fig. 1 and Fig. 2. In case of iterative conjugated gradient solver from PETSc, the convergence criteria based on relative solution error equal to 10^{-6} has been used. The reported executions times have been obtained as an average of five consecutive runs.

It can be clearly observed, that solution times are reduced with increasing number of threads. However the SuperLU MT and SuperLU DIST solvers needed considerably more time to solve problem than iterative solver PETSc on both benchmark problems. This can be due to the used tolerance for the iterative solver,

however the used tolerance is absolutely sufficient for engineering applications and the ability of any iterative solver to terminate solution when user defined tolerance is reached is an additional advantage. On the other hand, direct solvers can perform much better when solving problems with multiple right hand sides, for example. In Micro250k example, the efficiency of SuperLU MT is not as good as of SuperLU DIST and it seems that distributed version can better handle systems with less dense profile, however, this should be confirmed by more testing. The computational times for 16 threads and more are similar and no further speed-up is achieved. This is most likely due to the fact the benchmark problems are relatively small and using larger number of processors leads unfavorable ratio of communication and data exchange over actual computation. This is typical behavior for shared memory systems, where all the communication and data exchange is served by shared memory bus, with limited bandwidth. This, however, affects all evaluated solvers, as MPI is using shared memory communication on SMP system internally as well.

It is not possible to make any general conclusion from the presented results, but it is clear, that the choice of optimal solver and its parameters (tolerance, preconditioner) depends on actual problem and available hardware architecture.

4. Conclusion

In this work, performance of the SuperLU solver based on distributed system memory is evaluated and compared. The evaluation and scalability was demonstrated on two benchmark problems. The results show that there are significant differences between performance of direct and iterative solvers for the considered benchmarks. The SuperLU MT and SuperLU DIST solvers needed considerably more time to solve problem than iterative solver PETSc on both benchmark problems. However, the presented conclusions are related to particular selection of benchmark problems. In general, the possibility of using SuperLU DIST based on hybrid parallel model (combination of shared and distributed memory) is considered as an advantage. It is therefore important to have different possibilities available in a general purpose finite element code.

Acknowledgments

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS16/038/OHK1/1T/11 - Advanced algorithms for nu- merical modeling in mechanics of structures and materials.

References

Balay, S., Gropp, W. D., McInnes, L. C. and Smith, B. F. (1997) *Modern Software Tools in Scientific Computing*. Birkhauser, pp 163–202.

Barney, B. (2014) *OpenMP - An Application Program Interface.* Lawrence Livermore National Lab., Berkeley, online: https://computing.llnl.gov/tutorials/openMP/.

Bosansky, M. and Patzak, B., (2017), Evaluation of different approaches to solution of the direct solution of large, sparse systems of linear equations. *Advanced Materials Research*, Vol 1144, pp 97-101, doi: 10.4028/www.scientific.net/AMR.1144.97.

Bosansky, M. and Patzak, B., (2017), On Parallelization of Linear System Equation Solver in Finite Element Software. In: *Engineering Mechanics 2017*, Brno University of Technology, Brno, pp 206-209.

Kloeckner, A. (2014) *GPU Technology*. Lawrence Livermore National Lab., Berkeley, online: https://hpc.llnl.gov/data-vis/vis-software/gpu-cuda .

Demmel, J. W., Gilbert, J. R. and Li, X. S., (1999), *SuperLU users guide*. Technical report, Lawrence Livermore National Lab., Berkeley, doi: 10.2172/751785.

Pacheco, P. S. (1998) *A User's Guide to MPI.* Technical report, University of San Francisco, San Francisco.

Patzak, B. (2000) *OOFEM*. online: http://www.oofem.org

Patzak, B. (2010) *Parallel computations in structural mechanics*, Czech Technical University in Prague, Prague.

Pissanetzky, S. (1984) S*parse Matrix Technology.* Accademic Press.