PERFORMANCE ENHANCEMENT OF PARALLEL SYMMETRIC GENERALIZED EIGENVALUE SOLVER

Si Hyong PARK [1], Jong Geun MOON [1], Wan Il BYUN[1] and Seung Jo KIM[1,2]

1) *School of Mechanical and Aerospace Engineering, Seoul National University, Seoul 151-742, KOREA*
2) *Flight Vehicle Research Center, Seoul National University, Seoul 151-742, KOREA*

Corresponding Author: Seung Jo KIM, sjkim@snu.ac.kr

ABSTRACT

We discuss the performance enhancement techniques for a parallel block Lanczos eigen-solver which deals with a symmetric generalized eigenvalue problem. Approaches proposed in this research are suited for distributed memory parallel environments. Performance tuning of a eigen-solver covers the efficient implementation of block Lanczos iteration and the optimization of the linear equation solver incorporated with the eigen-solver. We propose a block Lanczos iteration equipped with effective mass multiplication algorithm and apply some kinds of concepts to enhance the performance of a linear equation solver. Factorization and triangular system solving phases are simultaneously considered for optimization of a linear equation solver. After individual routines are optimized respectively, they are applied to a single eigen-solver code.

MOTIVATION

The eigenvalue solver considered in the present research is a generalized one which can handle the equation with the mass matrix obtained from the well-known finite element method. The equation has the form as

$$\lambda \mathbf{M}x = \mathbf{K}x \tag{1}$$

Since the stiffness and the mass matrix are symmetric semi-definite ones, the well-known Lanczos iteration is applicable to Equation (1). Exactly speaking, the shifted invert version such as Equation (2) should be utilized due to presence of the mass matrix.

$$\frac{1}{\alpha}\mathbf{L}^T x = \mathbf{L}^T(\mathbf{K} - \sigma\mathbf{M})^{-1}\mathbf{L}\mathbf{L}^T x \tag{2}$$

$\mathbf{L}$ is a lower triangular matrix which can be obtained from factorization of the form $\mathbf{M} = \mathbf{L}\mathbf{L}^T$, and α is the eigenvalue inverted after shifting by σ. Equation (2) can be solved by the $\mathbf{M}$ orthogonal block Lanczos iteration, which will be not explained in this extended abstract. However, it should be noted that the important floating point operations during the iteration are composed of three phases. The first one is related with the basic Lanczos operations composed of matrix multiplication, inner product and orthogonalization using QR factorization. Between such operations, one linear equation solving routine is required. Since the linear equation solving routine handles the same matrix $\mathbf{K}$ and various RHS(right-hand

side), one Cholesky factorization and repeated triangular system solving are the second and the third phases. From the aspect of the flow of eigen-solver algorithm, the Cholesky factorization is located before the block Lanczos iteration, and the basic Lanczos iterative operations are coupled with the repeated triangular system solving routine.
In the present research, a multi frontal algorithm which is based on the finite element mesh structure and domain partitioning[1] is adopted for the linear equation solver. Firstly, if the Lanczos vectors are assigned through processors according to distribution of RHS in the multi frontal solver as proposed in Reference[2], the only communication during the basic Lanczos operations is collective one such as MPI_ALLREDUCE of message passing interface library. If such a communication is implemented effectively, the linear equation solver is the only bottleneck in the parallel eigenvalue computation. For performance enhancement of the multi frontal linear equation solver, three kinds of tuning techniques are proposed. The first is to reduce communication in the Cholesky factorization phase, and the second is to make triangular system solving routine be composed of matrix multiplications which are efficient in parallel environments due to the data independency. Finally, the linear equation solver can be tuned by communication topology parameters.

STRATEGY OF PERFORMANCE ENHANCEMENT

The distributing scheme of Lanczos vectors is dependent on the data structure of the multi frontal solver. This means that there is a portion of vectors which is overlapped through some of processors as shown in Figure 1. The unknowns located in the common interface of two domains are shared by two processors, and the mass matrix of the domain belonging to each processor is computed without communication. Therefore, some entities of the mass matrix of each processor are not fully summed values. Using this setting of data allocation, the basic Lanczos operations can be conducted with only a collective communication routine. In the present research, we utilize MPI_ALLREDUCE function without any modification. Though the algorithm of MPI_ALLREDUCE is known as a hybrid one without detailed information, performance is feasible in the case of small amount of data.

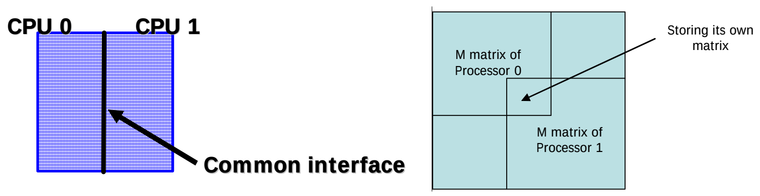

Figure 1. Data sharing interface and mass matrix structure

As explained above, performance tuning of the multi frontal linear equation solver is conducted from three points of views. The first one is to reduce communication occurring in the Cholesky factorization. It is notable that the naming convention of subroutines used in the eigen-solver is based on those of LAPACK(Linear Algebra Package) and BLAS(Basic Linear Algebra Subprograms). As shown in Figure 2, the symmetric frontal matrix is composed of three matrix entities, and three routines are required to factorize the frontal matrix. In parallel matrix operations based on panel communication which are adopted in the present research, each routine performs panel or block communications which are column(row) broadcasting or reducing type. Considering the detailed communication pattern, it is apparent that some parts of broadcasting or reducing are duplicated. For example, column broadcasting of panels of $\mathbf{K}_{11}$ in *POTRF* is also present in *TRSM*. If combining three routines into one is possible, duplicated communications will be avoidable. More detailed technique for combining will not be explained herein. The next is to tune the triangular system solving routines. Operations in the triangular system solving are generally composed of *TRSM* and *GEMM* with $\mathbf{F}_1$ and $\mathbf{F}_2$. Since *TRSM* routine has data-dependent algorithmic flow, there may be idle processors between panel broadcasting. One idea to resolve this bottleneck is converting *TRSM* into *TRMM* by computing inverse of $\mathbf{K}_{11}$ using *TRTRI* routine. Although *TRMM* requires the same number of floating point operations and communication volume as *TRSM* does, the LCM(least common multiple) concept can apparently reduce the idling time. The LCM concept which is originally proposed in the research of *GEMM* [3] can be applied too in any case of data-independent communication pattern. In this approach, performance negotiation between computing time for inverse of $\mathbf{K}_{11}$ and gains by using *TRSM* should be taken into account. The final performance tuning can be conducted by topology control in panel or block communication. We implemented two kinds of topology which are “increasing ring” and “split ring”.

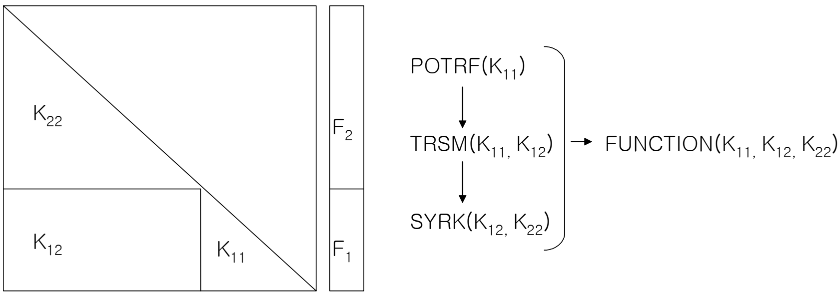

Figure 2. Frontal matrix and factorization routines

PRELIMINARY RESULTS

As preliminary tuning procedures, we conducted performance comparison with various topology settings and analyzed the effect of computing inverse of $\mathbf{K}_{11}$. Topology comparison was performed with a testing code which can execute each routine separately. The combined factorization routine requires 5 row or column panel broadcasting procedures which form 32 topology sets showing in Table 1, where the character ‘s’ and ‘i’ mean the initial letters of split ring and increasing ring respectively. Simple comparison of elapsed time reveals that the topologies ‘isssi’ and ‘issii’ commonly show best performance. In Table 2, for the case of TRTRI routine which has 3 panel or column broadcasts, the topologies ‘sss’ and ‘iss’ are outstanding. We chose ‘issii’ and ‘iss’ topologies, which will be enough to optimize the Cholesky factorization and partial inverting routines. Since the routines *TRMM* and *GEMM* which are used in the triangular system solving stage are known to be most efficient with the increasing ring topology using the LCM concept, case study is not required.

Table 1 Topology comparison of combined factorization time(sec)

Topology sets															
sssss	ssssi	sssis	sssii	ssiss	ssisi	ssiis	ssiii	sisss	sissi	sisis	sisii	siiss	siisi	siiis	siiii
issss	isssi	issis	issii	isiss	isisi	isiis	isiii	iisss	iissi	iisis	iisii	iiiss	iiisi	iiiis	iiiii
Matrix dimension = 32000 x 32000, processor map = 8 x 8, block size = 100															
608	595	606	595	632	620	631	620	617	603	616	602	641	624	641	625
606	592	604	592	631	617	631	617	617	602	617	599	640	622	637	623
Matrix dimension = 8000 x 8000, processor map = 4 x 4, block size = 100															
40	40	40	40	40	40	40	40	40	40	40	40	40	40	41	40
40	40	40	40	40	40	40	40	40	40	40	40	40	40	41	40

Table 2 Topology comparison of TRTRI time(sec)

Topology sets							
sss	ssi	sis	sii	iss	isi	iis	iii
Matrix dimension = 32000 x 32000, processor map = 8 x 8, block size = 100							
123	153	159	190	123	153	158	189
Matrix dimension = 8000 x 8000, processor map = 4 x 4, block size = 100							
9	10	11	11	9	10	10	11

The effect of partially computing inverse of $\mathbf{K}_{11}$ is shown in Table 3, where the performance results of three vibration problems are compared with the normal algorithm using the forward elimination and backward substitution. Although the time taken for factorization is longer for the case of partial inverting algorithm, triangular system solving with matrix multiplication ($TRMM$, $GEMM$) is efficient enough to reduce the total elapsed time. Such an effect will be more apparent for computation of a large number of eigenvalues.

Table 3 Comparison of the algorithms with and without partial inverting $\mathbf{K}_{11}$

	80 x 80 x 80 (20 loop)	800 x 800 x 1 (13 loop)	1260 x 1260 x 1 (13 loop)
Normal Elimination-Substitution algorithm	Fact : 490 sec Tri solve : 1532 sec Lanczos : 31 sec	Fact : 44 sec Tri solve : 110 sec Lanczos : 12 sec	Fact : 122 sec Tri solve : 254 sec Lanczos : 27 sec
Partial inverting algorithm	Fact : 544sec Tri solve : 1355sec Lanczos : 30 sec	Fact : 47 sec Tri solve : 100 sec Lanczos : 11 sec	Fact : 135 sec Tri solve : 232 sec Lanczos : 28 sec

REFERENCES

1. Kim, J. H. and Kim, S. J., “Multifrontal Solver Combined with Graph Partitioner,” *AIAA Journal*, Vol. 37, 1999, pp. 964-970.
2. Park, S. H, Moon, J. J and Kim, S. J., “Design of Parallel Block Lanczos Code based on Data Structure of Multifrontal Solver,” *KSIAM 2005 Conference in Spring*, Seoul Korea, May 2005.
3. Choi, J., “A Fast Scalable Universal Matrix Multiplication Algorithm on Distributed-Memory Concurrent Computers,” Proc. Of the IPPS, pp. 310-314, April 1997.