Efficient Computational Strategy for the Large Scale Fluid Analysis (10M Elements Problem)

Yasushi Nakabayashi, Genki Yagawa and Shinobu Yoshimura

University of Tokyo, Japan

ABSTRACT The efficient computational strategy for the large scale finite element fluid analysis was shown in this paper. This strategy originates in the Matrix-Storage Free (MSP) formulation proposed by Okuda et al., which is based on the one-point quadrature method in the hexahedral element and special hourglass controller for the diffusion term. The method reduces computational costs in both CPU time and memory storage. Furthermore, the authors developed pre/post-processing sub-systems for large scale problems. As numerical examples, 10 million elements (approximately 40 million DOFs) problems were solved. Using the efficient domain decomposer based on ParMETIS and Message Passing Interface (MPI), high parallel efficiency was achieved on both MPP (Hitachi SR2201) and PC cluster (DEC Alpha Cluster).

INTRODUCTION

For the computational fluid analysis, the finite element method (FEM), which has been used structural analysis, is attractive in terms of its applicabilities to an unstructured mesh and its simplicity for managing the boundary conditions. The bottleneck of FEM is large memory consumption and high cost of CPU. The one-point quadrature (OPQ) technique proposed by Gresho et al. [1] is quite efficient in saving the computational storage. In this technique, which employs the Q1-P0 element (i.e. velocity-linear/pressure-piecewise constant element), the diffusion and the advection matrices can be computed from the gradient matrix. Therefore the storage needed for the diffusion and advection matrices is unnecessary. Furthermore, using the Matrix-Storage Free (MSP) formulation proposed by Okuda et al. [2][3] the gradient matrix can be computed from nodal informations i.e. nodal coordinates and element-node connectivities. Owing to this formulation, even the storage for the gradient matrix also becomes unnecessary.

Today's parallel computers including MPP, Work Station clusters (WSC) and PC clusters (PCC) have enough power to solve a large scale and complicated problem that was considered impossible in a few years ago. With these progress of hardware, software is also developing for a parallel computer to realize a high performance, however, it is always behind hardware. While many researches are being done in these years, more applications or technique for parallel computers are needed in various fields to bring out the ability of parallel computers. In this paper, considering such trends of computing technology and requirements of solving large scale and complicated problems, an efficient parallel implementation of finite element fluids analysis system was proposed.
The MIMD-type parallelization was performed on the Hitachi SR2201 and DEC Alpha Cluster using Message Passing Interface (MPI). Using MPI library, the developed code became flexible and robust.

According to the recent computer progress, requirements for solving large scale, over one million DOFs, fluid problem is increasing. Tezduyar et al. showed the results of several million DOFs problems [4][5]. There are, however, many difficulties for solving more large problems. At first, CPU cost becomes extremely large. In this paper, parallel processing is one of the most important topics for this reason. Secondly, memory cost also becomes very large. As shown above, MSF formulation is good approach for minimizing memory cost. MSF formulation is also good approach for reducing CPU cost because CPU cost is approximately proportional to DOFs squared (without these scheme DOFs cubed). Finally, input and output data size become giga bytes order (in the case of over ten million DOFs problems). Some operating system (OS) cannot treat over two giga bytes files. Then data decomposition is inevitable for such large scale problems. By decomposing data file, parallel I/O is be available. It is also necessary for performing I/O in the reasonable time. These difficulty are not only in the solving problems but also in the pre/post-processing. In the pre-processing, mesh generation and domain decomposition spend large CPU cost, memory cost and hard disk. In the post processing, data conversion and visualization spend the same resource. Therefore, parallelization, economizing memory cost, data split and parallel I/O are also necessary for pre/post-processing sub-systems.

FINITE ELEMENT FORMULATION

Governing Equations and Galerkin Method

In this study, three-dimensional incompressibility viscous flow was considered. Governing equations for such a flow are the Navier-Stokes equations and incompressibility constraint equation as follows:

\[
\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{u}
\]  
\[
\nabla \cdot \mathbf{u} = 0
\]

where \( \mathbf{u} \), \( p \) and \( Re \) are the velocity, the pressure divided by the fluid density and the Reynolds number, respectively. The initial/boundary-value problem is considered with the condition as follows:

\[
\mathbf{u} = \mathbf{u}_0 \quad \text{on } \Gamma_u
\]

\[
\tau = \tau_0 \quad \text{on } \Gamma_\tau
\]

where \( \mathbf{u}_0 \) and \( \tau_0 \) are the Dirichlet and Neumann boundary condition value, respectively, enforced on the corresponding boundary \( \Gamma_u \) and \( \Gamma_{\tau} \). Here, \( \tau \) is defined as follows:

\[
\tau = -p\mathbf{n} + \nu(\mathbf{n} \cdot \nabla)\mathbf{u}
\]

Given an initial velocity field, which satisfies Eq.(2) and appropriate boundary conditions for \( \mathbf{u} \), Eqs.(1) and (2) can be solved, in principal, for \( \mathbf{u} \) and \( p \) as functions of space and time. The finite element spatial discretizations of Eqs.(1) and (2) are performed using the Galerkin method with the mixed interpolation formulation as follows:

\[
\mathbf{u} = \sum_{n=1}^{N_{\text{node}}} \Phi_n \mathbf{u}_n
\]
where \( \Phi_n \) is a \( C^0 \) piecewise trilinear basis function with respect to node \( n \). \( \Psi_e \) is a \( C^{-1} \) piecewise constant basis function (unity on element \( e \) and zero on all other elements). \( N_{\text{node}} \) and \( N_{\text{elem}} \) are the total number of the nodes and elements, respectively. Using Eqs.(6) and (7), Eqs.(1) and (2) are expressed as follows:

\[
M \dot{u} + B(u)u - Cp + Du = f
\]  
(8)

\[
C^t u = 0
\]  
(9)

\( M, B, C \) and \( D \) are mass, advection, gradient and diffusion matrices, respectively. \( f \) is nodal external force vector. For the time marching method, Marker-and-Cell (MAC) algorithm with Adams-Bashforth (AB) time integration was employed. In this method, Eq.(8) and (9) are transformed into the explicit scheme of velocity field and the discretized Poisson’s equation for the pressure as follows:

\[
u_{n+1} = u_n - \Delta t M^{-1} (-Cp_{n+1} + B(u_n)u_n + Du_n - f_n)
\]  
(10)

\[
(C^t M^{-1} C) p_{n+1} = -\frac{1}{\Delta t} C^t u_n + C^t M^{-1} (B(u_n)u_n + Du_n - f_n)
\]  
(11)

where \( \Delta t \) and subscript \( n \) denotes time increment and time step corresponding to the time \( n\Delta t \), respectively. After Eq.(11) is solved using the Conjugate Gradient (CG) method preconditioned by the diagonal scaling, the velocity field of the next time step is calculated by Eq.(10).

Matrix-Storage Free Formulation

In the FEM formulation, the mass, the advection, the gradient and the diffusion matrices of Eqs.(10) and (10.1) are all integrated in element-wise. The mass matrix is diagonally lumped, which is an explicit time marching of velocity field (see Eq.(10)), as follows:

\[
M_{\alpha\beta}^e = \int_{Q^e} \Phi_\alpha \Phi_\beta d\Omega = \delta_{\alpha\beta} \int_{Q^e} \Phi_\alpha d\Omega
\]  
(12)

where subscripts \( \alpha \) and \( \beta \) denote the number of node in element \( e \), which varies from one to eight in hexahedral element. \( Q^e \) denotes the volume of element \( e \). By virtue of this lumping, the inverse of the mass matrix in Eq.(10) becomes no longer necessary, and the amounts of memory consumption can be economized. The other matrices, the advection, the diffusion and the gradient matrix, otherwise, can not be lumped, an integration in the element is required. Contribution from the element \( e \) of the the advection, the diffusion and the gradient matrix are described as follows:

\[
B_{\alpha\beta}^e = \int_{Q^e} \Phi_\alpha \frac{\partial \Phi_\beta}{\partial x_j} d\Omega
\]  
(13)

\[
D_{\alpha\beta}^e = \frac{1}{Re} \int_{Q^e} \frac{\partial \Phi_\alpha}{\partial x_j} \frac{\partial \Phi_\beta}{\partial x_j} d\Omega
\]  
(14)

\[
C_{\alpha i}^e = \int_{Q^e} \frac{\partial \Phi_\alpha}{\partial x_i} \Psi_e d\Omega = \int_{Q^e} \frac{\partial \Phi_\alpha}{\partial x_i} d\Omega
\]  
(15)

where \( \Psi_e \) is the most recent value of the velocity. The subscript \( i \) and \( j \) denote a component of coordination. In the typical finite element formulation, Eqs.(13), (14) and (15)
are integrated using 8-point Gauss’s numerical integration. The integration, however, requires large amounts of CPU cost, then these matrices must be stored on the memory during solving a problem. By using One-Point Quadrature technique proposed by Gresho et al.[7], Eqs.(13) and (14) can be written as:

\[ B_{\alpha\beta}^e = \frac{1}{8} \bar{v}_j(0)C_{\beta j}^e \]  
\[ D_{\alpha\beta}^e = \frac{1}{Re} \frac{C_{\alpha j}^e C_{\beta j}^e}{\Omega^e} \]  

where \( \bar{v}_j(0) \) is the velocity evaluated at the center of the element. Eqs.(16) and (17) give significant meaning that \( B \) and \( D \) can be computed from \( C \), therefore the storage needed for \( B \) and \( D \) can be released. Even in this situation, the storage of the gradient matrix occupies almost half of the total memory. Regarding the gradient matrix, the Matrix-Storage Free formulation proposed by Okuda et al.[7] can be available. Applying the divergence theorem to Eq.(15) and using the one-point quadrature again, we get:

\[ C_{\alpha j}^e = \int_{e} \Phi_{\alpha} n_i d\Gamma = \frac{1}{4} \left( \Gamma^{(a)} n_i^{(a)} + \Gamma^{(b)} n_i^{(b)} + \Gamma^{(c)} n_i^{(c)} \right) \]  

where \((a), (b)\) and \((c)\) denote three surfaces of element \( e \) on which node \( \alpha \) is included. \( \Gamma^{(a)} \) and \( n_i^{(a)} \) are the area and the outward unit vector of the surface \((a)\). An area vector of each surface, which is used to evaluate Eq.(18), can be computed by taking an outer product of vectors of element edges. When the element surface is a flat plane, the above algorithm bears the same result as the exact integration. It is clear that all data for computing gradient matrix can be derived from nodal informations. Hence, even the storage for the gradient matrix also become unnecessary, that is, all matrix component need not be stored on the memory. We can calculate all matrix using MSP formulation whenever needed.

PARALLEL PROCESSING

Messegue Passing Interface
MIMD-type parallel implementation of finite element fluid analysis code is done using MPI. The target machines of this implementation are not restricted because MPI standard libraries absorb the difference between hardware architectures. Generally, using MIMD-type parallelization (e.g. MPI) result in high parallel efficiency compared with SIMD-type parallelization because parallelizable portion of code is increased and users can control more detail operations. On the other hand, implementation procedure is more difficult than SIMD-type implementation. The developed CFD code employs the Conjugate Gradient (CG) method as the solver of pressure Poisson’s equation. Parallel implementation of CG method is extremely simple in the case of using element-by-element scheme. There are only two kind of data communication, exchanging nodal data on the domain boundary and summarizing vector norms of each domain. Three routines of MPI, “MPI_Send”, “MPI_Recv” and “MPI_Reduce”, are sufficient for these communications. In this paper, Hitachi SR2201 and DEC Alpha Cluster were used with the MPI library. The SR2201 employs an innovative three-dimensional crossbar switch to provide high-speed connection among individual processing elements (PE). With this switch, there are only three output lines from any PE; one for each of the crossbars. This simple layout achieves almost the same performance as the configuration which interconnects all the processing elements directly, yet at a much lower cost. The Alpha Clusters are connected via fast
ether cable network, which is not fast compared with SR2201 network. However, using PC and ether network as the virtual parallel processors has the advantage in the point of economical view. In fact, recently, there are many WSC and PCC at the universities, laboratories and companies.

Domain Decomposition
One more important topic in the parallel processing is domain decomposition. In this parallelization style (MIMD-type), two steps of pre-processing are required. They are mesh generation and domain decomposition. The first step, mesh generation, is almost the same as SIMD-type parallelization one except for node/element numbering. Node/element numbering in this step is not necessary for MIMD-type parallelization because it is done in the next step. The second step, domain-decomposition, has many difficult requirement. At first, the cross section of domain boundary should be minimum for decreasing the amount of data communication. Secondly, each domain has the same number of elements, node and boundary conditions for the load balance of each processor. Furthermore, these procedure should be in parallel because it needs large CPU and memory costs. The graph partitioning tool METIS/ParMETIS [6][7] were adopted for the domain decomposition tool, which satisfied these requirement. The procedure of preprocessing can be summerized as three steps. At first, making the shape of analysis domain and attaching the boundary conditions. Secondly, generating the elements/nodes using the Inteligent Local Approach (ILA) [8] or any commercial tool. Finally, decomposition of domain by the METIS or parallel version of it, ParMETIS. The output of these pre-processing should be multi files (i.e. one file for each PE) for parallel I/O described above.

NUMERICAL EXAMPLE

Sample Problem
As the example of large scale problem, we employed two size of square cavity problems. Analysis model is shown in Figure 1. All the walls of the cavity have no-slip boundary conditions (BC) except the top lid, which has the non-dimensional velocity of 1.0 in the
x-direction. Reynolds number is set to 100. In this analysis, the cube was regularly divided into 100 or 216 elements in each direction i.e. total number of elements is 1,000,000 or 10,077,696 and total number of nodes is 1,030,301 or 10,218,313, which corresponds to 4,090,903 DOFs (4 million DOFs) or 40,732,635 DOFs (40 million DOFs).

Both 4 million DOFs and 40 million DOFs problems showed the same result. Figure 2 show the velocity of x-direction on A-A' line (see Figure 1) at the converged state, which corresponds better to the result of Ku et al. [9]. Figures 3 and 4 are velocity hog on plane S1 and S2 (see Figure 1), respectively, which show the appearance of three dimensional effect.

Parallel Efficiency
In this section, parallel performances such as the speedup and scalability are shown. Figures 5 and 6 shows 4 million DOFs problem on Alpha Cluster and SR2201, respectively. Because it is impossible to solve 4 million DOFs problem on one processor, exact speed-up can not be calculated. Therefore, “time-steps / second” was used for estimating approximately speed-up and scalability. In both case, high parallel efficiency was confirmed except for the Alpha Cluster using large number of PEs. The reason of this inefficiency is considered as the saturation of communication because the communication network of Alpha Cluster has not enough band-width. Otherwise, the speed-up became extremally high on SR2201 of 256 PEs. A so-called super-linear curve appeared by the cache effect. Figures 7 and 8 shows the parallel efficiency of 40 million DOFs problem and convergence behavior of velocity field, respectively. Even in the case of 40 million DOFs problem, high parallel efficiency and straight convergence was confirmed.

CONCLUDING REMARKS

Finite element fluid analysis system, which is based on the Matrix-Storage Free formulation, was developed. Furthermore, pre/post-subsystems, which are suitable for large scale problem, were also developed. The developed system was parallelized using MPI
4,090,903 DOFs (Alpha-Cluster)

Figure 5: Alpha Cluster (4,090,903 DOFs)

4,090,903 DOFs (SR2201)

Figure 6: SR2201 (4,090,903 DOFs)

40,732,635 DOFs (SR2201)

Figure 7: SR2201 (40,732,635 DOFs)

Convergence of 40M DOFs cavity flow

Figure 8: Convergence behaviors of cavity flow

4 million DOFs and 40 million DOFs three-dimensional square cavity problem was solved using DEC Alpha Cluster and Hitachi SR2201. The results shows high parallel efficiency and linear scalability except for the case of Alpha Cluster with high number of PEs.

The problems which have more complex shape and unstructured mesh should be solved for the verifications of convergency and accuracy of the developed system.

REFERENCES


