International Journal of Innovation in Science and Mathematics
Volume 2, Issue 1, ISSN (Online): 2347–9051

Cluster Computing in Finite Element Programing to Study the Large Deformation Problems

P. K. Gupta & R. N. Khapre

Abstract — **This paper presents an application of Cluster computing in finite element programming to analyze the large deformation problems. A Cluster of eight PC's of different configurations is developed and used for the present work. Three parallel solvers of Gauss-Seidel Method, Gauss Elimination Method and Matrix Inversion Method are developed and implemented on this Cluster to achieve reduction in computational time in getting the solution of system of linear equations. Performance of these solvers is compared and the most suitable solver is implemented in a finite element software Finite Element Method-Large Deformations (FEMLD) developed on Windows NT platform to analyze the large deformation problems. A typical problem of axial compression of solid cylinder is analyzed using developed software on the developed Cluster and the variation in different components of computational time is obtained. By computing Speedup and Efficiency, the performance of the developed software is also presented.**

Keywords – **Cluster Computing, Finite Element Analysis, Large Deformations, Parallel Computing.**

I. Introduction

Analysis of large deformation metal forming problems is very complex due to its dependency on many process parameters. Finite element method could be used successfully in analyzing such problems [1]. Based on the finite element method, few commercial softwares are available and regularly used to study and understand such complex problems. Most of these softwares are executable on PC and therefore consumes considerable amount of computational time for solving large deformation problems. It is mainly because of iterative solution procedure and large number of elements used to discretize the problem domain to achieve desirable accurate solution. With the help of high computing power of supercomputers and parallel computing technique, now it is possible to save considerable amount of computational time in analyzing such large deformation problems.

Gupta and Mishra [2] measured computational time expenditure in various processes involved in a typical finite element analysis. They found that the major portion of computational time gets consumed in the process of solving system of linear equations generated in the finite element analysis. Hence they developed three parallel solvers using Gauss Seidel Method, Gauss Elimination Method and Matrix Inversion Method to solve the system of linear equations on supercomputer PARAM 10000. After comparing the performance of these three parallel solvers on supercomputer PARAM 10000, they found that Matrix Inversion Method parallel solver is the most efficient parallel solver. Adopting finite element formulation presented by Kobayashi et al. [3], they developed a software (FEMLD) on the platform of supercomputer PARAM 10000 for the analysis of large deformation problems.

Only few research papers are available on the analysis of large deformation problems on supercomputers. Kim and Im [4] and Cheon et al. [5] presented a modified block Jacobi preconditioning technique for analyzing metal forming problems. They used domain decomposition approach in their solver. They carried out their study on the supercomputer CRAY T3E 900. They compared their computational time results with the results obtained from conventional conjugate gradient method and Jacobi preconditioned conjugate gradient method. They found that the modified block Jacobi preconditioning technique is more efficient than the other two techniques.

Since the supercomputers are expensive as compared to the PC's, so PC Clusters could be tested to use in place of the supercomputers as an inexpensive alternative [6]. Very little work has been reported on the implementation of PC Cluster to carry out finite element analysis. Thiagarajan and Aravamuthan [7] have discussed the implementation of high-performance FORTRAN on 32-node Pentium II 350 MHz Linux Cluster. They used two different parallelization strategies on preconditioned conjugate gradient solver for linear elastic finite element analysis. They found that initially Total time reduces with increase in number of computers, but after certain number of computers Total time increases with further increase in number of computers. They found that, the variation of Total time with increasing number of computers is not smooth. The number of computers at which the measured Real time is minimum was called as optimal number of processors. They concluded that this optimal number of processors depends on the problem size. They also discussed the performance of the developed code by measuring the Speedup and found that the Speedup increases curvilinearly with increasing number of computers.

Through this paper an attempt has been made to develop and implement a Windows NT Cluster using Local Area Network (LAN) for analyzing large deformation problems using finite element method. Initially a Cluster of eight computer having dissimilar configurations is developed by connecting them using LAN through HUB of 10 MBPS capacity. On this developed Cluster, three parallel solvers are developed using Gauss Seidel Method, Gauss Elimination Method and Matrix Inversion Method. These three parallel solvers are tested and compared on the developed Cluster and the most suitable parallel solver is employed in a inhouse developed finite element software Finite Element Method-Large Deformations (FEMLD). A sample problem of axial compression of solid cylinder has been solved using the developed software adopting various mesh sizes. The computational time variations are obtained and studied. The Speedup variation is also obtained and presented to study the performance of the developed Cluster.

II. Cluster Setup

Usually LAN connected computers may not have similar configurations, therefore a Cluster consisting of eight computers of different configurations is created. In the developed Cluster, eight computers of IBM Personal Computer 300GL with different processor speeds are used. The Cluster contains four computers having single Intel P II processor of 300 MHz, three computers having single Intel P II processor of 400 MHz, and one computer having Intel P II of 500 MHz. The RAM size of these computers was also different. All computers with 300 MHz processor have RAM of size 64 MB and remaining all computers have RAM of size 128 MB. These computers were connected through a HUB of 10 MBPS capacity. Windows NT 4.0 workstation operating system was installed on all the computers. The developed Cluster is illustrated in Fig. 1. MPICH for Microsoft Windows compiler [8] was also installed on each computer of the Cluster. In addition to this, Microsoft Visual C++ 6.0 compiler was also installed to create executable files. Each computer was configured to execute parallel programs on this developed Cluster.

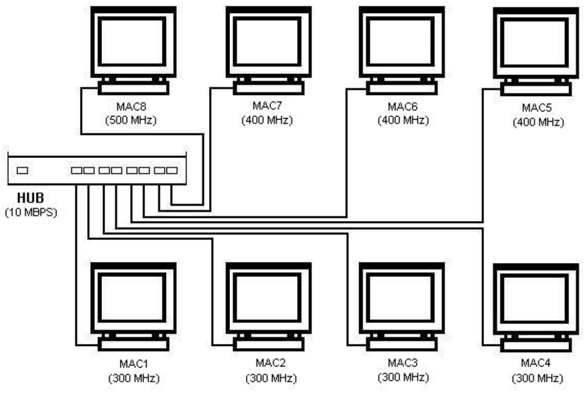


Fig.1. Windows NT cluster of computers with different configurations

III. Development of Parallel Solvers On Cluster

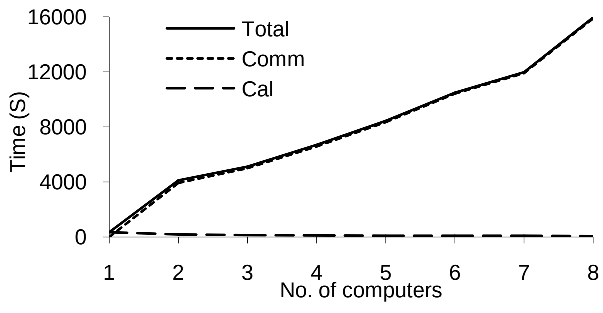
Mathematical methods used to solve system of linear equations [A] {X} = {B} are categorized under two categories namely Direct methods and Iterative methods. In Direct methods, number of computations required to achieve the solution are always know prior to the solution and dependent on number of equations to be handled. Whereas in Iterative methods, number of computations required to achieve the solution of desired accuracy are always unknown prior to actual computations. Here in the development of parallel solvers to solve system of linear equations [A] {X} = {B}, three techniques namely Gauss Seidel Method (GSM), Gauss Elimination Method (GEM) and Matrix Inversion Method (MIM) are used. Based on these techniques, three parallel solvers are developed on Windows NT platform and implemented on the developed Cluster discussed above. All three parallel solvers were used to obtain solution of five different data sets of increasing size (870×870, 1226×1226, 1352×1352, 1722×1722 and 2312×2312) taken from finite element analysis of large deformation problems. Since the developed Cluster is having computers of different processor speeds, so at least one computer with 300 MHz speed was involved in each execution process to maintain the consistency in computational time results. Computational time components, namely Total time (Total), Communication time (Comm.) and Calculation time (Cal.), were measured for all data sets, which were solved by three different parallel solvers with increasing number of computers from one to eight.

A. Gauss Seidel Method (GSM)

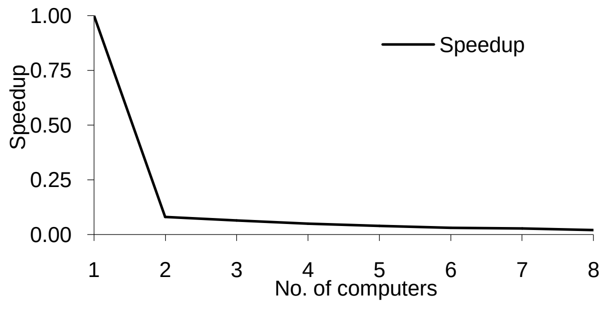
Gauss Seidel Method is an iterative technique to find the solution with desired accuracy. In this method iterations are carried out until the desired accuracy is achieved. It is obvious that as the desired accuracy increases, the number of iterations also increases correspondingly. Initially some guess values of solution vector ($X^{(0)}$) are assumed. Thereafter every equation is divided by its diagonal element. Using guess solution vector, first iteration is carried out to obtain the first solution vector ($X^{(1)}$), which is used for next solution vector. The iterations are carried out until the desired accuracy is achieved. The accuracy is checked by comparing two vectors $X^{(i)}$ and $X^{(i+1)}$.

In development of parallel solver using GSM column wise data distribution was carried out. Initially all computers were given ranks starting from zero. Then the data was uniformly distributed to all computers. If uniform data distribution was not possible, few computers with lower ranks were overloaded with few columns. Each computer operated columns of its share and evaluated sum that was necessary to evaluate the numerical value of each unknown. Then each computer sent this sum to the master computer and master computer added them to get the new value unknown vector $X^{(i)}$. After this master computer broadcasted new value of unknown vector $X^{(i)}$ to all the computers which were used for further calculation. This process continued till current iteration got completed. After every iteration, all computers checked the accuracy of the obtained solution by comparing the old values with new values of unknown vector $\{X\}$. If all values of unknown vector $\{X\}$ were of desired accuracy, then the master computer printed the results and the program get terminated.

Fig. 2 (a) and (b) show typical variation in components of computational time and Speedup with increasing number of computers for GSM parallel solver for data set of size 1226×1226. It can be seen from Fig. 2 (a) that Total time increases with increase in number of computers. Communication time variation is exactly similar to the Total time variation. Communication time is marginally lesser than the Total time at every number of computers. Reduction in Calculation time can also be seen with increase in number of computers. From Fig. 2 (b) it can be observed that Speedup reduces drastically from one to two number of computers. With further increase in number of computers, Speedup continuously decreases but with reduced rate.



(a)



(b)

Fig.2. (a) Computational time (b) Speedup variation with number of computers for Gauss-Seidel Method for data set of size 1226 × 1226

When other data sets of increasing sizes were solved, it was found that Total time increases with increase in number of computers employed for obtaining their solution. Fig. 3 shows the variation in Speedup with number of computers for all data sets under consideration. It can be seen that as data size increases, the corresponding performance of GSM parallel solver also improves. This improvement is insignificant, as Speedup remains less than 1.0. Table I shows the number of iterations carried out to obtain solution of various data sets. It can be seen that number of iterations required to solve a particular data set are independent of data size.

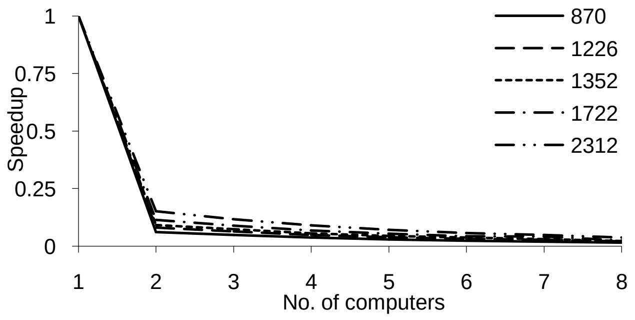


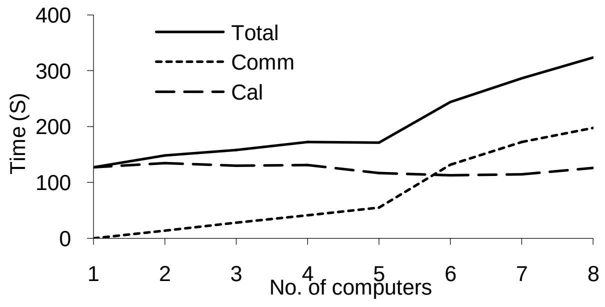
Fig. 3 Variation in Speedup for Gauss-Seidel Method for various data sets

Table I: Number of iteration for various data sizes carried out by Gauss-Seidel Method solver

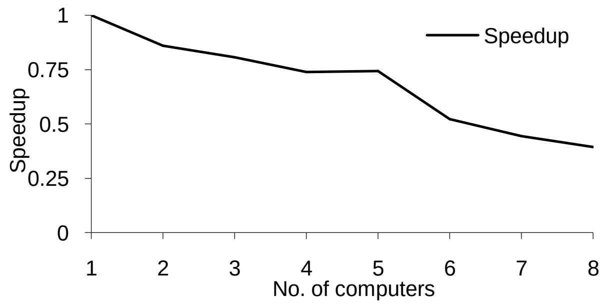
Data size	No. of iterations	Status
870 × 870	1083	Solution found
1226 × 1226	1790	Solution found
1352 × 1352	1803	Solution found
1722 × 1722	2238	Solution found
2312 × 2312	3076	Incomplete

B. Gauss Elimination Method (GEM)

Gauss Elimination Method falls under the category of direct method of solving system of linear equations. In this method, the system of linear equations is reduced to an equivalent upper triangular matrix, which can be solved by method of back substitution. For parallel implementation of GEM, row wise data distribution was carried out. Initially the range of data to be handled by each computer was decided. If data distribution was not even, then the remaining data was distributed to the computers with lower ranks. The operations were started from first row and stopped at the last row. Initially the diagonal element of current row was converted to unity, then the elements of lower diagonal were converted to zero by every computer simultaneously. When the complete matrix reduced to upper triangular matrix, then the unknowns were evaluated by method of back substitution. This process is data dependent and the entire unknowns cannot be evaluated simultaneously. Therefore the computer with highest rank started this process. It evaluated all the unknowns of its share; thereafter broadcasted them to all the computers, which were then utilized by other computers to evaluate the unknowns of their share.



(a)



(b)

Fig.4. (a) Computational time (b) Speedup variation with number of computers for Gauss Elimination Method for data set of size 1226 × 1226

Variation of different components of computational time and Speedup with increasing number of computers for data set of size 1226 × 1226 obtained using GEM parallel solver is shown in Fig. 4. It can be seen from Fig. 4 (a) that Total time increases with increase in number of computers. The increase is gradual from one to five number of computers. After five number of computers sudden increase in Total time can be seen till eight number of computers. The variation of Communication time is exactly similar to variation of Total time with significantly lesser magnitude. It can also be observed from this figure that Calculation time decreases with increase in number of computers. This reduction in Calculation time is significantly less with increasing number of computers. Fig. 4 (b) shows variation in Speedup with increasing number of computers. It can be seen that Speedup decreases with increase in number of computers.

It was observed that the variation in components of computational time is very much identical for all the data sets. In all the cases Total time increases with increase in number of computers. Fig. 5 shows the variation in Speedup with increasing number of computers for all data sets under consideration. It can be seen from this figure that the performance of the GEM parallel solver improves with increase in data size. Still the performance of GEM parallel solver is poor as the Speedup remains below 1.0 even after employing higher number of computers.

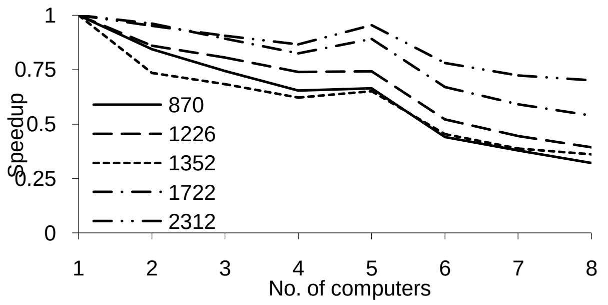


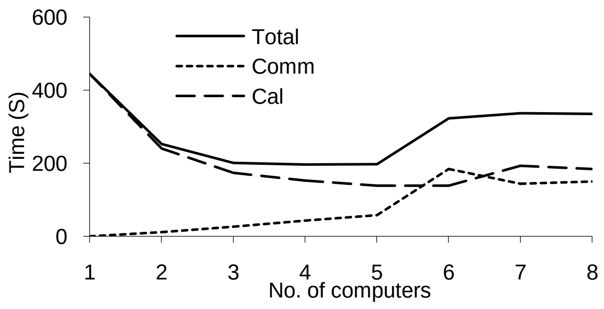
Fig.5. Variation in Speedup for Gauss Elimination Method for various data sets

C. *Matrix Inversion Method (MIM)*

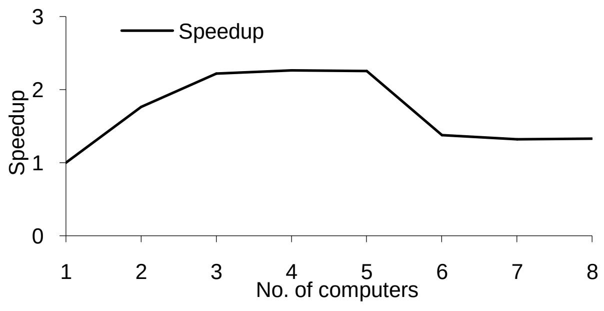
Matrix Inversion Method is a direct method that is also adopted for obtaining the solution of the system of linear equations [A] {X} = {B}. In this method inverse of the matrix [A] is calculated. Then the solutions are computed using expression {X} = $[A]^{-1}$ [B]. For parallel implementation, all computers were given their ranks starting from zero. Then the range of data to be handled by each computer was decided. Row wise data distribution was carried out as described in Gauss Elimination Method. After proper data distribution among the computers, an Identity matrix [I] of size [A] was created by all computers. In the process of matrix inversion, column wise operations were carried out. Every non-diagonal element of matrix [A] was converted to zero and every diagonal element of matrix [A] was made unity.

Whatever operations were carried out on matrix [A], same operations were also carried out on matrix [I] simultaneously. Each computer operated only those rows, which were designated to it to achieve less computational time. After finding the inverse of matrix [A], the unknown vector {X} was calculated by multiplying $[A]^{-1}$ with {B}. At this juncture, each computer was having elements of vector {X} those belong to its share. Then each computer broadcasted these elements of vector {X} to the all other computers so that every computer should have complete vector {X}.

The variation in components of computational time obtained by MIM parallel solver for data set of size 1226 × 1226 is shown in Fig. 6 (a). It can be seen from this figure that Total time reduces with increase in number of computers. Continuous reduction in Total time can be seen for one to five number of computers. After that, for further increase in number of computers, Total time increases slightly and remains almost constant. Continuous reduction in Calculation time with increase in number of computers can also be seen in this figure. It can also be observed that Communication time increases with increase in number of computers. Fig. 6 (b) shows variation in Speedup with increase in number of computers for data set of size 1226 × 1226. It can be seen that Speedup increases gradually from one to three number of computers. After three number of computers, Speedup remains almost constant till five number of computers. Sudden decrease in Speedup can be observed from five to six number of computers. Thereafter Speedup remains almost constant and more than 1.0.



(a)



(b)

Fig.6. (a) Computational time (b) Speedup variation with number of computers for Matrix Inversion Method for data set of size 1226 × 1226

When data set of increasing sizes were solved, it was observed that Total time reduces with increase in number of computers employed for obtaining solution of all data sets under consideration. Fig. 7 shows the variation in Speedup obtained on Windows NT Cluster for increasing data sets. It can be observed that the performance of MIM parallel solver improved significantly with increase in data size. Maximum Speedup was observed at five number of computers for all data sets under consideration. Highest Speedup of 3.16 was obtained for data set of size 2312 × 2312 at five number of computers.

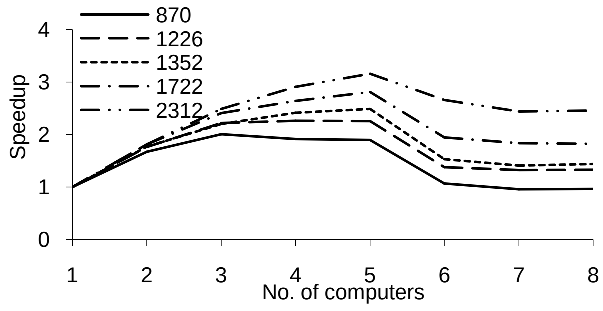


Fig.7. Variation in Speedup for Matrix Inversion Method for various data sets

IV. Comparison of Parallel Solvers

From Fig. 3, 5 and 7, it can be observed that for all data sets, each solver gave its best performance when five number of computers were employed. Fig. 8 shows variation in the Speedup with increasing data size for all three parallel solvers when five computers were employed. It can be observed from this figure that Speedup obtained by MIM parallel solver is maximum whereas by GSM parallel solver is minimum among all three parallel solvers. It can also be observed that GEM and GSM parallel solvers give Speedup less than 1.0 for all data sets under consideration. Hence MIM parallel solver is adopted and implemented in the developed finite element code FEMLD.

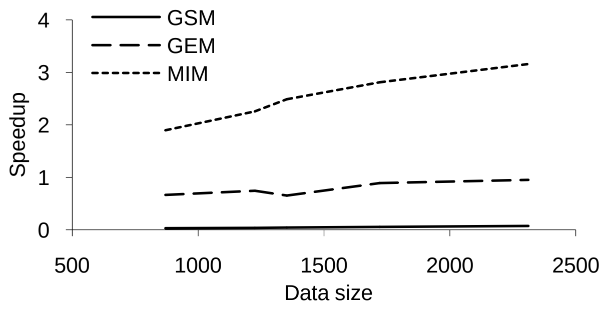


Fig.8. Comparison of four parallel solvers

V. Finite Element Procedure

In the finite element formulation employed to develop the software variational method is used for the derivation of the basic equation for finite element analysis.

$$\pi = \int_v E(\dot{\varepsilon}_{ij})dV - \int_{S_F} F_i u_i dS \tag{1}$$

where $E(\dot{\varepsilon}_{ij})$ is the work function, F_i represents surface traction, u_i is the velocity and S indicate the surface.

The stiffness equation can be written as

$$\frac{\partial\pi}{\partial v_i} = \sum_j \left(\frac{\partial\pi}{\partial v_i}\right)_{(j)} = 0 \tag{2}$$

where i and j represents node and element numbers respectively

Equation (2) can be rewritten in the form

$$\mathbf{K}\,\Delta \mathrm{v} = \mathrm{f} \tag{3}$$

where

$$\mathrm{K} = \frac{\partial^2\pi}{\partial v_i v_j}$$

$$= \int_v \frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}} P_{ij} dV + \int_v \left(\frac{1}{\dot{\bar{\varepsilon}}}\frac{\partial\bar{\sigma}}{\partial\dot{\bar{\varepsilon}}} - \frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}^2}\right)\frac{1}{\dot{\bar{\varepsilon}}} P_{ik} v_k v_m P_{mj} dV + \int_v KC_j C_i dV \tag{4}$$

In which $\Delta\mathbf{v}$ is the velocity correction term at nodal points and

$$\frac{\partial\pi}{\partial v_i} = \int_v \frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}} P_{ij} v_j dV + \int_v KC_j v_j C_i dV - \int_{sf} F_j N_{ji} dS \tag{5}$$

In the above expressions, $\bar{\sigma}$ is the effective stress, $\dot{\bar{\varepsilon}}$ is the effective strain rate, K represents penalty constant, N is the shape function matrix. Matrix P_{ij} is computed as

$$\mathbf{P} = \mathbf{B}^{\mathrm{T}}\,\mathbf{D}\,\mathbf{B} \tag{6}$$

where **B** is the strain rate matrix and **D** represents effective strain rate coefficient matrix.

Vector C is computed as

$$C^T = \{1,1,1,0\}\mathbf{B} \tag{7}$$

Penalty constant (K) is introduced to keep the volumetric strain nearer to zero. To identify the rigid regions where further deformation is not possible a term limiting strain rate $(\dot{\bar{\varepsilon}}_0)$ is introduced. The boundary conditions along the interface of deforming material and die are mixed and can be expressed as

$$S = S_u + S_F + S_C \tag{8}$$

where S_u represents the velocity boundary conditions. They are imposed in form of nodal velocities and element shape functions generated automatically during solution procedure. S_F represents traction boundary conditions that can be imposed in form of nodal-point forces. The term S_C represents the surface. To include the frictional effect between the deforming body and rigid die, Coulombs law $f_s = \mu p$ is used. (where μ is the coefficient of friction and p is the die pressure)

The strain rate components are calculated as

$$\dot{\varepsilon} = \mathbf{B}\,\mathbf{v} \tag{9}$$

where **v** is the velocity at nodal points

The effective strain rate is computed using following expression

$$\dot{\bar{\varepsilon}} = \left(\dot{\varepsilon}^T\,\mathbf{D}\,\dot{\varepsilon}^{1/2}\right) \tag{10}$$

The effective stress is calculated as

$$\bar{\sigma} = \bar{\sigma}\left(\bar{\varepsilon}, \dot{\bar{\varepsilon}}\right) \tag{11}$$

where $\bar{\varepsilon}$ is the effective strain. The stress components are computed using expression

$$\sigma = \frac{2}{3}\frac{\bar{\sigma}}{\dot{\bar{\varepsilon}}_0}\left(\dot{\varepsilon} - \dot{\varepsilon}_m\right) + 3K\dot{\varepsilon}_m \tag{12}$$

where $\dot{\varepsilon}_m$ is the mean strain rate

The entire deformation process is sub-divided into several number of increments. Stepwise solution for one increment is obtained in the following sequence;

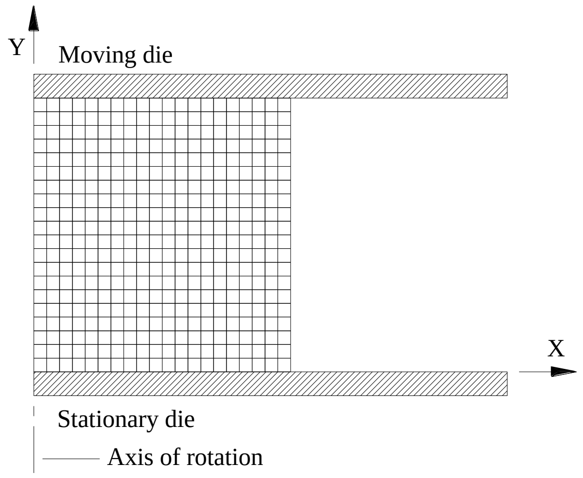
1) Assume guess nodal velocities on all nodes of the discretized problem domain.

2) Generation of stiffness matrix and residual force vector for each element. Assemble them to get stiffness equation for the entire problem domain.
3) Solve the stiffness equation to get the solution velocities.
4) Check the solution velocities with guess velocities through convergence criteria based on two error norms namely velocity error norm $\|\Delta v\|/\|v\|$ and force error norm $\|\partial\pi/\partial v\|$. (where $\|v\| = (v^T v)^{1/2}$ and v is the velocity at nodal points)
5) If convergence criteria's are not satisfied then follow the Direct iteration method or Newton Raphson method till the convergence is achieved.
6) Once the solution gets converge, updated geometry of problem domain is generated. This is followed by strain rate, strain and stress evaluation, which ultimately gives the complete solution of one increment.

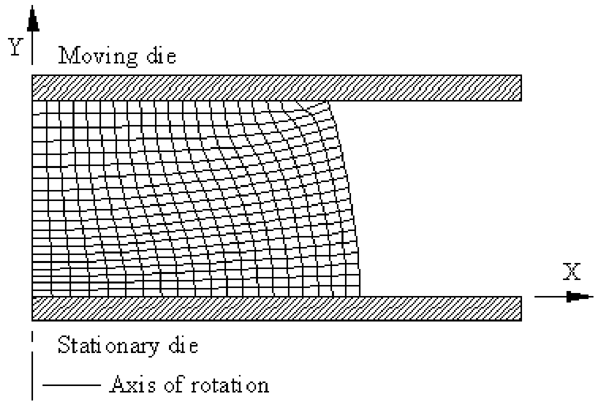
The entire procedure is repeated until the desired deformation is achieved. This procedure can be found in detail elsewhere (Gupta et al. [2]).

VI. Finite Element Analysis of Large Deformation Problem

A software for analyzing large deformation problems FEMLD was developed on the Windows NT platform incorporating MIM parallel solver. This software was developed using flow formulation presented by Kobayashi et al. [3] and four noded rectangular elements were used to discretize the problem domain. With the help of this software, a problem of simple compression of solid cylinder having dimensions, 1 (inch) radius and 1 (inch) height was analyzed on the developed Cluster. The cylinder was compressed between stationary bottom die and moving top die with a velocity of 1 inch/s till 30% reduction in height was achieved in 15 steps. The bottom surface of cylinder was considered as frictionless and for top surface, friction factor of magnitude 0.5 was considered. The error norms (velocity norm and force error norm) were considered as 0.001 and limiting strain-rate value is considered as 0.01 to define the rigid portion of the cylinder. The material behavior was expressed by the equation $\bar{\sigma} = k\dot{\bar{\varepsilon}}^m$, where the values of k and m were taken as 10 Ksi and 0.1 respectively. The cylinder was discretized three times using 400, 625 and 1089 four noded rectangular elements with 441, 676 and 1156 nodes, resulting in global stiffness matrix of size 882 × 882, 1352 × 1352 and 2312 × 2312 respectively. These problems were analyzed on Windows NT Cluster by employing one to eight computers and different components of computational time were measured. It was observed that 84, 36 and 33 iterations were required to analyze these problems. Fig. 9 (a) and (b) shows the undeformed and deformed finite element mesh with 441 four noded rectangular elements respectively.



(a)

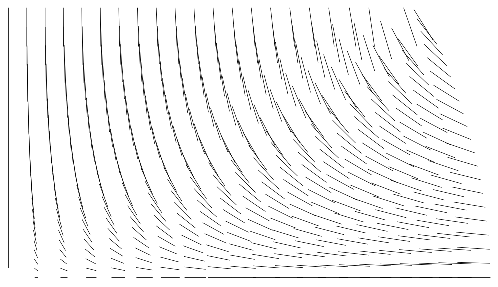


(b)

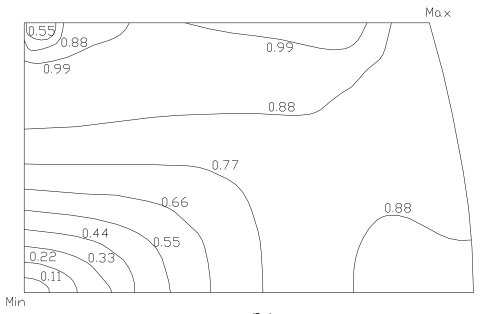
Fig.9. Axial compression of solid cylinder (a) Undeformed shape, (b) Deformed shape

Fig. 10 (a) shows the direction of nodal velocities obtained from FEMLD that illustrates the direction of metal flow during the deformation process. Fig. 10 (b), (c) and (d) show the contours of magnitude of nodal velocity, effective strain-rate and effective strain respectively after 30% reduction in height.

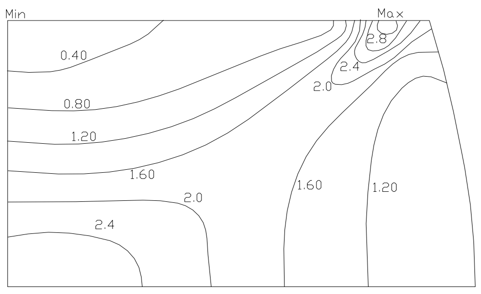
Table II shows the obtained variation in different components of computational time. It can be seen that Total time reduces with increase in number of computers. It reaches to its minimum value at three number of computers for the problem with stiffness matrix of size 882 × 882 and 2312 × 2312 and four number of computers for problem with stiffness matrix of size 1226 × 1226. Further increase in number of computers resulted in increase in Total time.



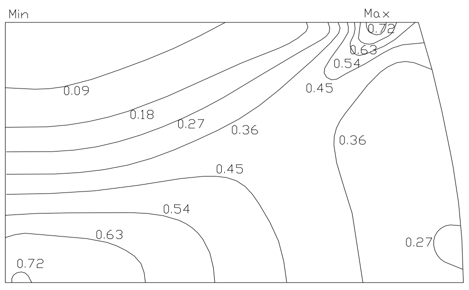
(a)



(b)



(c)



(d)

Fig. 10 (a) Direction of nodal velocity and contours of (b) nodal velocity (inch/s) (c) effective strain-rate and (d) effective strain after 30% reduction in height

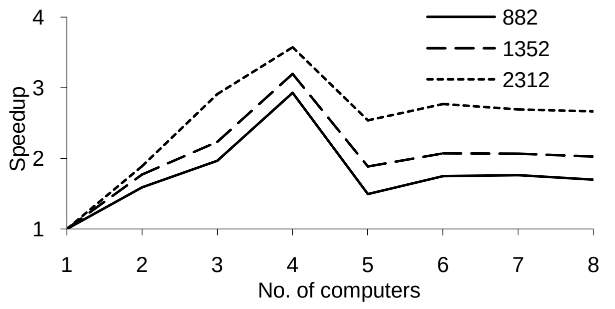


Fig.11. Variation in Speedup for FEMLD for problems with stiffness matrices of increasing size

Fig.11 shows the variation in Speedup obtained by FEMLD on Windows NT Cluster for problem with stiffness matrix of various sizes. Overall it can be seen that Speedup increases with increase in number of computers and reaches to its peak value. Then Speedup starts decreasing with further increase in number of computers. Maximum Speedup of 1.86 was obtained at three number of computers for problem with stiffness matrix of size 2312 × 2312. It can also be seen from this figure that the performance of FEMLD improves with increase in size of stiffness matrix.

Table 2: (a) Computational time and performance of FEMLD on Windows NT cluster for problem with stiffness matrix of size 882 × 882

No. of computers	Total time (s)	Comm. time (s)	Cal. time (s)	Speedup	Efficiency
1	23536.2	0.0	23536	1.00	100.00
2	14781.7	4887.3	9894.4	1.59	79.61
3	11965.1	5005.6	6959.4	1.97	65.57
4	8032.7	2553.6	5479.1	2.93	73.25
5	15732.6	9379.5	6353.0	1.50	29.92
6	13459.9	7058.3	6401.5	1.75	29.14
7	13350.7	7347.1	6003.6	1.76	25.18
8	13861.0	9327.1	4533.8	1.70	21.23

Table 2: (b) Computational time and performance of FEMLD on Windows NT cluster for problem with stiffness matrix of size 1352 × 1352

No. of computers	Total time (s)	Comm. time (s)	Cal. time (s)	Speedup	Efficiency
1	21202.5	0.0	21202	1.00	100.00
2	11960.7	515.7	11444	1.77	88.63
3	9495.6	1184.2	8311.3	2.23	74.43
4	6631.5	1244.8	5386.6	3.20	79.93
5	11241.7	6314.6	4927.1	1.89	37.72
6	10232.6	3280.6	6951.9	2.07	34.53
7	10252.6	3093.6	7159.0	2.07	29.54
8	10464.9	4437.3	6027.6	2.03	25.33

Table 2: (c) Computational time and performance of FEMLD on Windows NT cluster for problem with stiffness matrix of size 2312 × 2312

No. of computers	Total time (s)	Comm. time (s)	Cal. time (s)	Speedup	Efficiency
1	129323	0.0	129323	1.00	100.0
2	68424.9	1445.7	66979	1.89	94.50
3	44447.0	1257.4	43189	2.91	96.99
4	36189.4	3177.3	33012	3.57	89.34
5	50931.6	18001	32930	2.54	50.78
6	46712.8	11584	35128	2.77	46.14
7	47995.8	15905	32090	2.69	38.49
8	48555.8	20939	27616	2.66	33.29

VII. Conclusions

In this paper, an application of Cluster computing technique for analysis of large deformation problems using finite element method is presented. One Windows NT Cluster of eight computers is developed which consists of PC's having different configurations (300 MHz, 400 MHz, and 500 MHz) connected by HUB of 10 MBPS capacity. Three parallel solvers are developed on this Windows NT platform and tested for various sizes of data sets on developed Cluster. It was found that Matrix Inversion

Method parallel solver gives highest Speedup and takes minimum computational time as compared to the other solvers. Hence it was chosen for implementation in the finite element analysis.

A software for finite element analysis of large deformation problems (FEMLD) is developed on the developed Windows NT Cluster. One sample problem has been analyzed using the developed software by increasing the number of computers and its performance on developed Cluster is measured. It was found that, Total time required in finite element analysis was reduced by employing more number of computers of developed Windows NT Cluster. It has been also found that analysis could be carried out in minimum Total time by employing three to five number of computers depending on the size of the finite element mesh. It was also observed that excessive increase in number of computers resulted in increase in Total time. One advantage of using Windows NT Cluster for finite element analysis is that the computers of the Windows NT Cluster can be very easily replaced with the better and faster computers to achieve better performance.

REFERENCES

[1] P. K. Gupta, "An Investigation into Large Deformation Behaviour of Metallic Tubes," Ph.D. Thesis, Indian Institute of Technology, Delhi, India, 2000.

[2] P. K. Gupta, and J. P. Mishra, "Computer Simulation of Large Deformations Process," Research Project Report PI-10. Center for Development of Advanced Computing, Pune, 2004.

[3] S. Kobayashi, S. I. Oh, and T. Altan, "Metal Forming and The Finite-Element Method," Oxford University Press, New York, 1989.

[4] S. Y. Kim, and Y. T. Im, "Parallel Processing of 3D Rigid Viscoplastic Finite Element Analysis using Domain Decomposition and Modified Block Jacobi Preconditioned Technique," *Journal of Materials Processing Technology*, vol. 134, 2003 pp. 254–264.

[5] J. S. Cheon, S. Y. Kim, and Y. T. Im, "Application of Parallel Processing in Three-Dimensional Bulk Forming Finite Element Analysis," *Journal of Materials Processing Technology*, vol. 152, 2004, pp. 106–115.

[6] T. Sterling, "An Introduction to PC Clusters for High Performance Computing," *International Journal of High Performance Computing Applications*, vol. 15, no. 2, 2001, pp. 92-101.

[7] G. Thiagarajan, and V. Aravamuthan, "Parallelization Strategies for Element-By-Element Preconditioned Conjugate Gradient Solver Using High-Performance FORTRAN for Unstructured Finite-Element Applications on Linux Clusters," *ASCE Journal of Computing In Civil Engineering*, vol. 16, no. 1, 2002, pp. 1-10.

[8] MPICH is a high performance and widely portable implementation of the Message Passing Interface (MPI) standard. Available: http://www.mpich.org/

AUTHOR'S PROFILE

Dr P.K. Gupta

is working at the Indian Institute of Technology (IIT) Roorkee. He obtained his doctoral degree in 2001 from Indian Institute of Technology, Delhi, INDIA in the area of Structural Mechanics. His date of birth is 12/01/1966.

He is working as Associate Professor of Structural Engineering in Department of Civil Engineering of Indian Institute of Technology Roorkee, INDIA. His areas of research are: Structural mechanics, finite element analysis, parallel computing and large deformations. He has vast experience in consultancy and teaching at undergraduate and postgraduate levels. He has edited one conference proceedings and published more than 110 research papers in different international and national journals and conferences. He has guided three PhD and more than twenty five masters' theses. Presently five doctoral students are working with him. He is actively engaged in research and consultancy projects of national repute.

Dr. Gupta if life member of Life Member of the different societies which includes: Indian Society for Technical Education , Life Member of Institution of Engineers (India), Life Member of Applied Mechanics Society, I.I.T. Delhi, Life Member of Indian society of theoretical and Applied Mechanics, Life Member of Indian Association for Computational Mechanics. Recently Dr. Gupta was awarded with the Endeavour awards of Australian Government.

Dr. Rajendra N. Khapre

Place: Nagpur, Maharashtra, INDIA.
Date of Birth: 22 August 1978
Educational Background:
Ph.D. (Civil Engineering), BITS, Pilani, Rajasthan, INDIA, 2006.

He is working as ASSOCIATE PROFESSOR at Shri Ramdeobaba College of Engineering & Management, Nagpur. He is published 35 research papers in Journals and Conferences of National and International repute. Presently he is working in the field of Biomedical Engineering with application to dentistry.

Dr. Khapre is the member of Institution of Engineers (INDIA). He is also a Life Member of Indian Society For Technical Education, (INDIA).