Journal of Computational Science and Technology

Vol. 3, No. 1, 2009

Feasibility Study of Parallel Finite Element Analysis on Cluster-of-Clusters*

Masae MURAOKA** and Hiroshi OKUDA***
**Dep. Quantum Engineering and Systems Science, University of Tokyo,
7-3-1 Hongo Bunkyo-ku Tokyo, Japan
E-mail: muraoka@nihonbashi.race.u-tokyo.ac.jp
***Research into Artifacts, Center for Engineering, University of Tokyo,
5-1-5 Kashiwanoha Kashiwa-shi Chiba, Japan
E-mail: okdua@race.u-tokyo.ac.jp

Abstract
With the rapid growth of WAN infrastructure and development of Grid middleware, it's become a realistic and attractive methodology to connect cluster machines on wide-area network for the execution of computation-demanding applications. Many existing parallel finite element (FE) applications have been, however, designed and developed with a single computing resource in mind, since such applications require frequent synchronization and communication among processes. There have been few FE applications that can exploit the distributed environment so far. In this study, we explore the feasibility of FE applications on the cluster-of-clusters. First, we classify FE applications into two types, tightly coupled applications (TCA) and loosely coupled applications (LCA) based on their communication pattern. A prototype of each application is implemented on the cluster-of-clusters. We perform numerical experiments executing TCA and LCA on both the cluster-of-clusters and a single cluster. Thorough these experiments, by comparing the performances and communication cost in each case, we evaluate the feasibility of FEA on the cluster-of-clusters.

Key words: Computational Mechanics, Numerical Analysis, Finite Element Method, Parallel Computation, Grid Computing

1. Introduction

In the research field of computational mechanics, with the development of high performance computers that allow us to handle more realistic and accurate models, very large scale and highly reliable computations have been archived. The pursuit for even more realism and accuracy accelerates the cycle of innovation of the hardware and the computing environment. With the rapid growth of network infrastructure and the spread of commodity PCs, it has become feasible to utilize collections of computing resources composed of geographically distributed PC clusters for one large computation(1),(2). Furthermore, Grid enhanced MPI (Message Passing Interface), such as MPICH-G2(4) and Grid MPI(5), enable us to run the existing MPI programs on the Grid without any modification. This is a very attractive aspect for application both developers and users of MPI programs.

However, efforts to explore and evaluate the Grid as an effective methodology for enhancing computing resources have been mainly targeted towards asynchronous parallel programs, such as parametric studies and master-worker type programs, which do not require frequent communication among processes. Such applications are developed using the task parallel programming model and efficiently implemented as Grid

*Received 27 Oct., 2008 (No. T1-07-0026)
Japanese Original : Trans. Jpn. Soc. Mech. Eng., Vol.73, No.733, A (2007), pp.981-988 (Received 12 Jan., 2007)
[DOI: 10.1299/jcst.3.77]

applications[9],[10],[11] using RPC (Remote Procedure Call). The implementation of RPC enhanced to the Grid, Grid RPC, such as Ninf[7] and NetSolve[8], has been employed for many Grid applications. While task parallel applications using GridRPC have been successfully adopted, very few MPI programs which require frequent communication, like finite element analysis (FEA hereafter), have been tested and evaluated on the Grid, even though legacy programs that use MPI can be easily ported.

In this study we executed application programs across a test environment constructed using two clusters connected by the Internet. The application programs executed here are tightly connected application (TCA hereafter) and loosely connected application (LCA hereafter), which will be defined in the next section. Both of them are SPMD type programs related to FEA that have been parallelized using the domain decomposition approach. For TCA, in order to investigate the communication cost in more detail, we have also performed communication tests in which we measured the time elapsed for data transfer by the individual MPI functions used in the test applications. We executed these two types of applications on both a single cluster and on the cluster-of-clusters and performed numerical experiment for load balance between clusters by changing the number of processes assigned to each one. Through these numerical experiments we evaluate the feasibility of FEA on cluster-of-clusters.

2. Application classification

We classify FEA applications into two categories according to the communication frequency among the processes. In this section, we illustrate these applications by introducing our test examples.

2.1 Tightly connected applications

In FEA, the domain decomposition approach is generally employed for parallelization. A whole domain is divided into several sub-domains which are distributed to different processors and processed in parallel. The linear system obtained by discretizing the partial differential equations using FEM is assembled partially by each process. Therefore, the matrix vector operations related to nodes and elements belonging to only one sub-domain can be completed in isolation. However, matrix vector operations involving nodes and elements shared with neighbors cannot be completed as such. For example, the inner product needs collective communication among all the processes and the matrix vector multiplication also needs neighbor-neighbor communication. In FEA, those operations take place quite often. We classify such application requiring frequent communication among all the processes into TCA. Execution of TCA across remote sites, like on the Grid, incurs a high communication cost due to frequent communication via the Internet. This has discouraged researchers to utilize the Grid for such applications. However, utilizing the Grid becomes an inevitable choice when the simulation scale is too large to execute on a single resource, even when a big loss in performance is certain. So our first priority in the utilization of the Grid for TCA is to exploit the scalability of the Grid for running very large simulations rather than performance.

2.2 Loosely connected applications

In contrast, "loosely connected" is understood here as representing infrequent communication. Applications such multi-physics problems, multi-phase problems, and steering visualizations are classified into this category. They are composed by a number of individual program modules, each of which might be regarded as a TCA. However, although the modules constantly interact, the communication is quite infrequent. We therefore categorize such applications into the LCA group. As an example, in the simulation of very large scale fluid-structure interaction problems, both modules for fluid and structure can be individually seen as TCA, but they communicate and synchronize with each other only once in a certain number of steps.

3. Construction of the Cluster-of-Clusters on the Grid

In order to construct our test environment, we reconfigured our lab's cluster machine, which had been previously used in the local network only, so that it could be accessed from the external network. Our test environment is composed of this cluster machine and the one belonging to the Grid Technology Research Center, AIST (Advanced Industrial Science and Technology), which was already operated as a Grid node. In this section, we describe the Grid middleware and network configuration we used to make our cluster machine a Grid-enabled cluster.

3.1 Grid middleware

The grid middleware is the fundamental software system to build the Grid environment. Globus Toolkit[3)] is a widely used Grid middleware, seen to be as the de-facto standard. Globus Toolkit provides low level services which hide the complex technology needed for utilizing the distributed heterogeneous resources, such as user and resource authentication, job initiation and management, directory services, and so on. Though the latest version of Globus Toolkit is ver. 4.0.8, we used Globus Toolkit ver. 2.4.3 which is both supported by MPICH-G and has been extensively used for scientific computation. MPICH-G is an implementation of the MPI standard enhanced for the Grid environment. By using MPICH-G, one can port existing MPI programs to the Grid without any modification.

3.2 Network configurations

One of the clusters of our test environment is a Linux cluster which had been previously used only in our local network. We reconfigured it to be accessible to / from the external network.

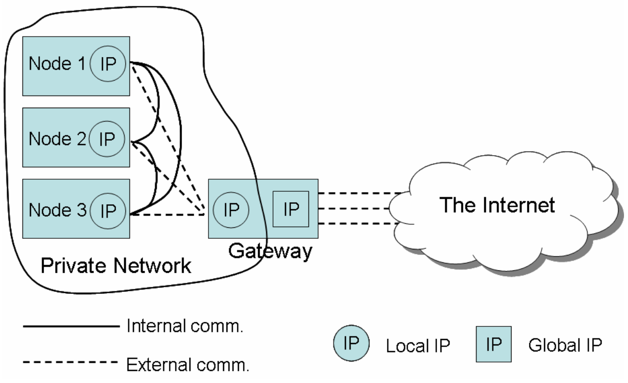


Fig. 1 Network configuration of PC cluster on Grid

We transformed the control node of our cluster into a gateway by adding a network interface to the external network, as shown in Fig. 1. The rest of the computing nodes remained in the private network and have to communicate with the external network via the gateway. Since all communication packets pass through the network interface on the gateway machine, communication to and from the remote sites becomes very inefficient.

Nevertheless we selected this configuration for the following three reasons. (1) We can control the security configurations (e.g. firewall) for all nodes on the gateway machine. (2) This cluster is also operated as a local cluster and it is preferred to remain unchanged as much as possible. (3) Our first priority in the utilization of the Grid is to explore the scalability obtained from collections of heterogeneous resources rather than performance.

For the execution of MPI program across different sites on the Internet, each computing node has to be resolved by name, since MPI processes need to establish connections and communicate with each other. The computing nodes insides the local network, having only local IP addresses, require their own global IP address to communicate to computing nodes found outside the gateway. The gateway has Global IP addresses for all the nodes as well as for itself and provides a service for matching between local IP addresses and global IP addresses. This mechanism is implemented by one-on-one NAT and IP masquerading using iptables which is provided by Linux kernel.

3.3 Firewall

In order to securely execute MPI program among the remote sites, the gateway also performs the function of firewall on incoming ports, using iptables. We describe our security policy and configuration which allows incoming packets only on the ports used by Globus Toolkit2 and MPICH-G2.

There are three kinds of ports that are allowed. The first one is for Globus Gatekeeper. Globus Gatekeeper is a daemon program of GRAM (Globus Resource Allocation Manager) which acts as an interface to remote hosts that initiates and manages the jobs. Globus Gatekeeper uses port 2119 by default.

The second one is for Grid FTP. Grid FTP provides a service for secure data movement. It uses port 2811 by default. We use Grid FTP to distribute the FEM mesh files to remote hosts.

The last kind of ports are the ones used by the communication among MPI processes. Once the connections are established, the ports are persistent during for the lifetime of the MPI jobs. Thus the number of ports expected has to be specified before invoking the job. Globus Toolkit handles such situations via an environment variable named GLOBUS_TCP_PORT_RANGE. The value of this variable should be formatted as a pair of values "*min, max*" (a comma separated pair which represents the range of allowed ports). We can restricts ephemeral ports for Globus services using this setting(12).

3.3 Job initiation procedure

A script file described in RSL (Resource Specification Language), which is provided in the Globus Toolkit, is used to invoke the MPI jobs by GRAM. Using the RSL script file, one can specify the resources and jobs in detail. Fig. 2 shows an example of RSL file. As the figure shows, the core syntax of the RSL file is composed by attribute-value couples. For example, the relation "*executable=a.out*" provides the name of an executable. Other keywords are used to specify host name, the number of jobs, working directory and environment variables and so on. This RSL file shows GLOBUS_TCP_PORT_RANGE set in the range of 40000 to 40010.

```
+
( &(resourceManagerContact =“piyo.example.ac.jp ")
   (count=1)
   (label="subjob 0")
   (environment=(GLOBUS_DUROC_SUBJOB_INDEX 0)
               (GLOBUS_TCP_PORT_RANGE "40000,40010")
               (LD_LIBRARY_PATH /usr/local/globus/lib/))
   (directory="/home/muraoka/work_dir")
   (executable="/home/muraoka/work_dir/executable ")
)
```

Fig. 2 Example of RSL file

4. Test applications

As a test application of TCA-type, we use a thermal-fluid analysis code called “MPI-TSLOW”. This is an SPMD-type program which has been developed and executed on massively parallel computers and supercomputers. For LCA-type applications, we use an MxN type steering visualization program. In this section we provide a brief overview of both applications.

4.1 Thermal-fluid analysis MPI-TSLOW

Parallel FE programs using MPI can be regarded as typical TCA-type applications. We use a FE thermal fluid analysis program (called MPI-TSLOW(13),(14) hereafter) as a test application. In this program, the Navier-Stokes equations, the incompressibility constraint equation and the energy conservation equation are solved for velocity, pressure and temperature. To integrate the equations in time, an explicit Euler scheme is employed for velocity and temperature, while pressure is implicitly treated and obtained by solving the Poisson’s equation at each time step. The conjugate gradient (CG hereafter) method with diagonal scaling is used to solve the pressure Poisson’s equation. Since the parallelization is based on the domain decomposition of the finite element mesh, each process has parts of the coefficient matrix and vectors corresponding to the assigned sub-domain. Communication is required at each calculation of the matrix vector multiplication and inner product, which take place quite frequently in iterative methods.

4.2 MxN steering visualization

As a LCA–type test problem, we use an application of steering visualization developed using the MxN communication model (15). The MxN communication model is a technique which enables one to run an application composed of two parallel programs on a distributed environment, such as cluster-of-clusters, efficiently. “MxN” represents the numbers of processes we assign to each program, respectively. This steering visualization application is composed of two SPMD-type parallel programs: time marching FEA simulation and a visualization process which is executed every several time steps of FEA. These parallel programs need to synchronize and communicate with each other constantly but not frequently.

For such applications consisting of different parallel programs, by using the MxN communication model, both the optimization of the selection of resources according to the program load and the configuration the numbers of processes to be assigned on each resource can be achieved. Furthermore, in our test application the data flow is only from FEA to VIS. Thus the FEA can restart right after transferring the result data to VIS, without waiting for VIS to finish the visualization. As such, the processing time of FEA and VIS overlap and the faster executing process is practically hidden by the one taking longer, as shown in Fig. 3. Since all the processes of FEA and VIS take part in the data transfer process, a global synchronization is performed before and after data transfer. Fig. 3 illustrates the application flow, in which the FEA assigned to cluster A and VIS to cluster B. Data transfers are performed every five time steps of FEA. Therefore the VIS visualizes the result once every five steps.

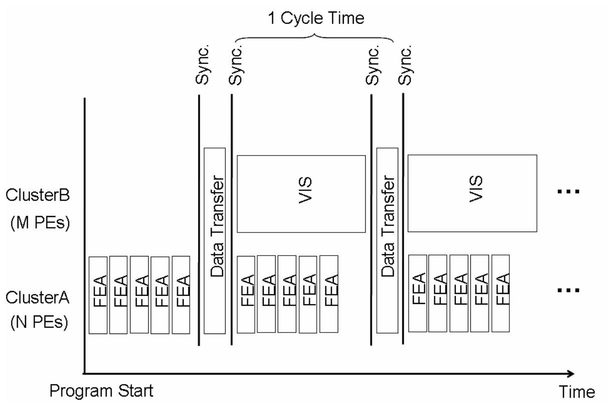
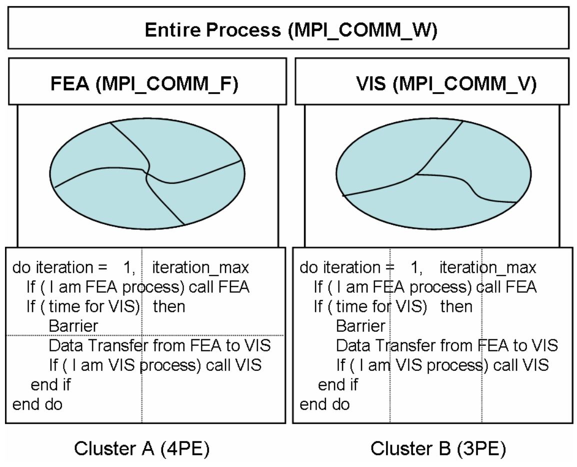


Fig. 3 Time integration algorithm of MxN steering visualization (*M* PEs for cluster A, *N* PEs for cluster B)

Both FEA and VIS have internal communication to perform. We denote the communicator for FEA by MPI_COMM_F, containing *M* processes, while the communicator for VIS, containing *N* processes, is denoted by MPI_COMM_V. Besides them, another communicator containing both the FEA and VIS process, thus having *M+N* processes, called MPI_COMM_W, is created. MPI_COMM_W is used for communication in the data transfer process. Fig. 4 illustrates the case in which the FEA is executed by four processes (four sub-domains) and VIS is executed by three processes (three sub-domains).

In the MPI-based MxN application, every process has its own rank number, from zero to *M+N-1*. At the start of the program, each process finds which application it belongs to (either FEA or VIS) by checking its own rank number. The *M* processes whose rank numbers are from zero to *M-1* belong to FEA while the rest, from *M* to *M+N-1*, belong to VIS.



```
do iteration =  1,   iteration_max
  If ( I am FEA process) call FEA
  If ( time for VIS)   then
     Barrier
     Data Transfer from FEA to VIS
     If ( I am VIS process) call VIS
   end if
end do
```

```
do iteration =  1,   iteration_max
  If ( I am FEA process) call FEA
  If ( time for VIS)   then
     Barrier
     Data Transfer from FEA to VIS
     If ( I am VIS process) call VIS
   end if
end do
```

Fig. 4 MxN steering visualization executed with three MPI communicators

5. Execution scenario

We executed the two types of applications described in the sections above on the test environment consisting of two cluster machines. In this section we describe our test environment and execution conditions for MPI-TSLOW and MxN steering visualization, respectively.

5.1 Test environment

Numerical experiments on the Grid were performed on a collection of two cluster machines connected by the Internet. The former, called the OGT cluster (Alpha21269 667MHz, 10node, 1GB Memory/node, 100Base-X) and located at the University of Tokyo, in Tokyo, Japan, was configured as a Grid-enabled cluster as described in section 3. The later, called the F32 cluster (Intel Xeon 3.06GHz, 128CPU/ 64node, 4GB Memory/node, Gigabit Ethernet) was located at the Grid Technology Center, AIST, in Ibaraki, Japan. The specifications of these cluster machines are quite different in various aspects, not only in architecture and hardware but in network configurations and policy as well. As stated in section 3, the OGT cluster has an interface to the Internet only on the front-end node, which provides a one-on-one NAT for mapping Global IP addresses and Local IP addresses on behalf of the nodes used for computations, which remain in the private network. On the other hand, every node of F32 cluster has an interface to the Internet.

5.2 MPI-TSLOW

We constructed a scenario in which one sub-domain is regarded as one process and assigned one process to each processor element (PE). In order to manage the job execution across the clusters, we used RSL files which describe the resource attributes which we were going to use. We specified the name of the PE with full domain name. The PE described in the file were numbered in the order they appeared in the RSL file, number which corresponds to the rank of MPI process assigned to that particular PE. We specified the number of processes assigned to each cluster by changing the number of PE appearing in the RSL file. For example, in the case that we execute the program on 8 processes and assign six processes to cluster A and two to cluster B, we describe the six PE of cluster A first and the two PE of cluster B afterwards.

5.3 MxN Steering Visualization

As both FEA and VIS are parallel programs based on the domain decomposition method, we regarded one sub-domain as one process assigned to one PE belonging to either of the two clusters. Let *M* be the number of sub domains for FEA and *N* for VIS. We use *M* PE on one cluster for FEA and *N* PE on the other cluster for VIS. All the *M*+*N* processes create one communicator; MPI_COMM_W. The processes whose rank numbered from zero to *M-1* create their own communicator MPI_COMM_F for the computation of FEA while the rest (the processes numbered from *M* to *M*+*N-1*) create another communicator MPI_COMM_V used for VIS. Since the order of PE described in the RSL file corresponds to the number of the ranks in MPI_COMM_W, we can assign FEA and VIS to different clusters by describing *M* PE for the cluster used for FEA first then followed by *N* PE for the cluster used for VIS.

6. Numerical results

In order to see the effects of the communication cost across the sites, we executed numerical experiments, changing the number of processes assigned to each cluster for both MPI-TSLOW and MxN steering visualization. Since MPI-TSLOW requires frequent communication via the Internet, we also performed a basic experiment for evaluating the individual communication cost of the MPI functions used in the application.

6.1 MPI-TSLOW (TCA)

In MPI-TSLOW communication across the sites takes place very frequently. The experiment consisted in first measuring the elapsed time for each MPI function and then carrying on with the numerical experiments using MPI-TSLOW. The test model was a cavity flow problem of 2 000 000 nodes which took 65 iterations to converge. We measured the solving time for the pressure Poisson's equation using the CG method. We executed the same experiment both on a single cluster and on cluster-of-clusters. We the compared the solving time for each case,

6.1.1 Communication test

Communication is required by each calculation of the matrix vector multiplication and each inner product, both taking place quite frequently in iterative methods. The matrix vector multiplication is completed with neighbor- neighbor communication, which is implemented in our program by the MPI_Isend and MPI_Irecv (Isend/Irecv hereafter) routines. The data transmitted by Isend/Irecv is an array of type MPI_DOUBLE_PRECISION, whose size corresponds to the number of nodes shared with the neighboring domains. Depending on the size of the test models, the size of the data reaches a relatively large amount, from KB to MB. On the other hand, the inner product needs collective communication, implemented in our program using MPI_Allreduce (Allreduce hereafter). The data transmitted by the Allreduce operation consists of only one element of type MPI_DOUBLE_PRECISION.

For each communication using the MPI function, we measured the elapsed time by changing the data size from 8B to 8MB. For each data size we obtained the elapsed time by averaging 25 measurements. For this experiment we used MPICH-G2, the same MPI library used for the test applications. The result obtained by Allreduce is plotted in Fig. 5 and the result for Isend/Irecv is plotted in Fig. 6. We focus on the data size implemented in MPI-TSLOW. For Allreduce, we focus on the elapsed time for 8B. While the difference in internal communication time between nodes of F32 and SC is quite small, we observe a difference of an order or magnitude between the intra-cluster case and inter-cluster case. For Isend/Irecv we focus on data sizes ranging from KB to MB. The intra-cluster elapsed time is of course larger than the inner-cluster time, but this difference becomes smaller as the data size increases. Based on the observation obtained here, in the next section we investigate the communication cost in the execution of MPI-TSLOW.

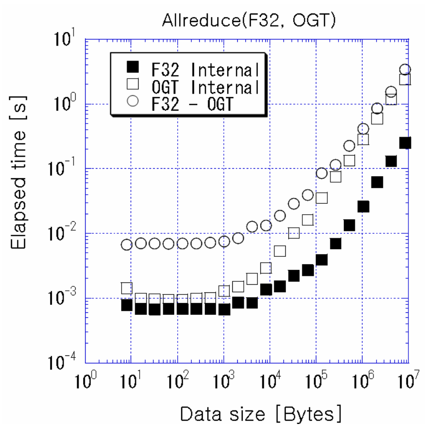


Fig. 5 Data transfer by Allreduce

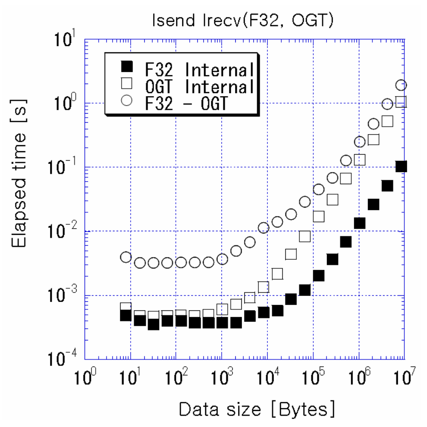


Fig. 6 Data transfer by Isend/Irecv

6.1.2 MPI-TSLOW on Cluster-of-Clusters

We compare the elapsed time for MPI-TSLOW on Cluster-of-Clusters with that for a single cluster. The test model is a 3D cavity flow with 2 000 000 nodes. The size of test model is the maximum size that can be executed by one PE on OGT, whose memory is smaller than that of F32. We executed the MPI-TSLOW under the conditions described below. As we presented in section 5.2, we regard one sub-domain as one process, allocate one process to one processor and assign the same number of processes to each cluster. Under this scenario, we executed MPI-TSLOW changing the number of parallel processes, from two to eight. In each case, we measured the elapsed time for one time step of time marching flow simulation and compared the time obtained for a single cluster to that obtained by using Cluster-of-Clusters. The result is shown in Fig. 7.

First we note that the elapsed time by one PE on OGT is three times longer that that of F32, which is a direct result of the difference in hardware performance between the two. Since the hardware performance is directly reflected in computing time if assigning the same number of sub domains, the total time elapsed is dominated by the computing time of the cluster whose hardware performance is lower. As a result, the time obtained for OGT alone is almost the same with that of the Cluster-of-Clusters.

In regard to the above observation, the next thing that should be noted is that the elapsed time on the cluster-of-clusters is almost the same as that of OGT alone, even if the execution on the cluster-of-clusters includes the communication cost across the sites. In the particular case of two PE, one PE on each cluster, the elapsed time was 127[s] (figure 7). The communication pattern in this case is illustrated in Table 3. According to the results of the communication test described in section 6.1.1, the intra-cluster Allreduce at 8B is faster than the inter-clusters Allreduce by an order of magnitude (Fig. 5). However, even for such a long communication time, of the order of 10^{-2} [s], as the function is only called for 260 times, it sums up to a mere 2[s] which is insignificant if compared to the total elapsed time of 127[s]. On the other hand, focusing on the data size of 185KB for Isend/Irecv, there is no big difference in the elapsed time between OGT alone and the cluster-of-clusters. This difference is, as in the previous case, sums up to insignificant communication cost viewed from OGT side (Fig. 6).

Therefore, for the effective execution of TCA, the reduction of the waiting time occurred at every synchronization, due to the hardware performance, which leads to the reduce of work ratio, it is a primary issue.

Tab. 1 Data size and call count during 1 time step execution

	MPI_Allreduce	Isend/Irecv
Data size / Call [B]	8	185 000
Call cnt / Time step [cnt]	260	198

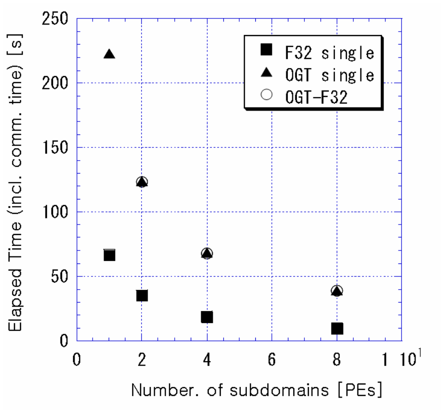


Fig. 7 Intra- and Inter- cluster computations of 2M node cavity flow

6.2 MxN Steering Visualization (LCA)

For LCA, the test model for FEA is a 3D heat transfer problem of 32,000 nodes. We fixed the number of sub domains for FEA at eight and assigned the FEA to the F32 cluster. We assigned VIS is to OGT. By changing the number of sub-domains for VIS, from 2 to 8, we measured the total elapsed time while configuring different load balances between clusters. VIS is performed every 10 steps of FEA.

The experimental results are shown in Fig. 8. The horizontal axis represents the number of sub-domains of VIS and the vertical axis represents the time for each action (FEA, VIS, and data transfer between them) for one cycle. One cycle here means a sequence of 10 steps of FEA followed by the data transfer from F32 to OGT and one VIS process, as shown in Fig. 3. After 10 time steps, after completing the data transfer, VIS starts to visualize the simulation result. At the same time, the FEA can also restart the simulation for the next 10 time steps. Their processing times overlaps. Since the process finishing later hides the elapsed time of the one finishing faster, the measured time for “one cycle” is the time belonging to the longer process (either FEA or VIS) plus data transfer time.

In this test problem, we fixed the number PE for FEA. Thus the processing time for FEA is constant in every case. We can reduce the processing time for VIS by increasing the number of PE, so in this way we can hide the VIS time in the FEA time. As we can see in Fig. 8, the VIS time becomes shorter than the FEA time when increasing the PE count from two to four. For both cases 4 PE and 8 PE, the VIS times are hidden in FEA times. As the FEA time dominates the total time, the reduction of VIS time by increasing the PE count from 4 to 8 doesn’t appear effective. In addition, by increasing the PE count, the data transfer time become larger.

As is often the case with fluid analysis, we sometimes have to increase the number of time steps from tenth of thousands to millions to obtain the steady state. In such a case, even a very small time increase due to imbalanced load assignment would be accumulated and becomes large. Thus it is essential for an application to execute under a load balance that hides the processing time of the other process and also minimizes the cost of data transfer most effectively. It is, however, difficult to find such an effective load balance, that is an optimal choice of M and N, before execution. Besides, the Grid environment is usually shared with many users and the occupancy changes from time to time. Thus the load balancing has to be considered dynamically. One adequate implementation is to run the application with the proper set of M and N for several cycles, measuring the elapsed time for each of FEA, VIS, and data transfer, and introduce a load balancing mechanism to find a better load balance based on the actual performance data obtained from test run. But to change M and N means to change the number of sub domains, thus to repartition the finite element mesh. As the cost of the load balance is accompanied by that of the repartitioning of the mesh and the redistribution of the data, dynamic load-balancing remains an unsolved issue.

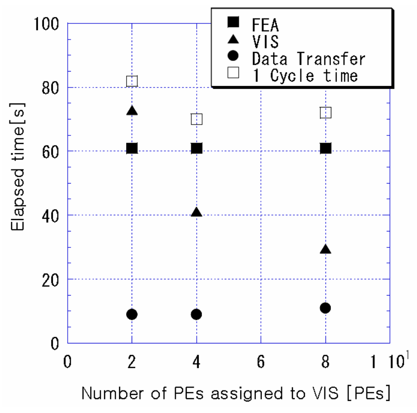


Fig. 8 Elapsed time when varying number of VIS PEs

7. Concluding remarks

We constructed a Grid environment composed of two cluster machines connected by the Internet, which allowed us to execute MPI program across them. By executing two types of FEA, that is, TCA and LCA on the heterogeneous cluster-of-clusters, we reached the following conclusions:

1. Focusing on the communication times of Allreduce at 8B and Isend/Irecv at 185KB, as actually used in MPI-TSLOW, although the difference between the inner-cluster and inter-cluster cases is by an order of magnitude higher for the former communication pattern, the total inter-cluster communication time is negligible in comparison with the computation time. For the later case, the difference between the inner-cluster and inter-cluster cases is also small enough so that the performance of the cluster-of-clusters system is very close to the one of the OGT cluster alone.
2. The key factor to improve the performance of TCA is to realize a load balancing mechanism to compensate the difference of hardware specifications.
3. For LCA, it is also important to introduce a load balancing mechanism between different program modules. Though the calculation time decreases by increasing the number of PEs, the communication cost spent by data transfer between program modules increases, which leads to an increase in the total execution time.

Acknowledgment

This work has been conducted as part of ApGrid activities. We wish to express our gratitude to all of the ApGrid participants, especially AIST. And this study was supported by Toray Industries.

References

(1) I. Foster, C. Kesselman, S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International J. Supercomputer Applications, Vol.15, No.3 (2001).

(2) I. Foster, C. Kesselman, Computational Grids, The Grid: Blueprint for a New Computing Infrastructure, (1999) , Morgan-Kaufman.

(3) I. Foster, C. Kesselman, Globus: A Metacomputing Infrastructure Toolkit, Intl J. Supercomputer Applications, Vol. 11, No.2 (1997), pp.115-128.

(4) I. Foster and Nicholas T. Karonis, A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems, Proc. Supercomputing, (1998).

(5) Grid MPI (http://www.gridmpi.org/index.jsp)

(6) K. Seymour, H. Nakada, S. Matsuoka, J. Dongarra, C. Lee and H. Casanova, GridRPC: A Remote Procedure Call API for Grid Computing, Proceedings of Third International Workshop on Grid Computing, (2002), pp. 274-278.

(7) M. Sato, H. Nakada, S. Sekiguchi, S. Matsuoka, U. Nagashima and H. Takagi, Ninf: a network based information library for a global world-wide computing infrastructure, High-Performance Computing and Networking, Lecture Notes in Computing Science, Vol. 1225, (1997), Springer-Verlag.

(8) H. Casanova and J. Dongarra, NetSolve: A network server for solving computational science problems, International J. Supercomputing Applications and High Performance Computing, Vol. 11, No.3(1997).

(9) H. SHIMOSAKA, M. MIKI, M. SANO, Y. TANIMURA, Y. MIMURA and S. YOSHIMURA, Truss Structure Optimization by using NetSolve, Proceedings of the 5th Symposium on Optimization of JSME (OPTIS'02), (2002), pp.141-146 .

(10) H. Takemiya, K. Shudo, Y. Tanaka, S. Sekiguchi, Development of Grid Applications on Standard Grid Middleware, In Workshop on Grid Applications and Programming Tools GGF8 (2003).

(11) N. Emad, S. A.Shahzadeh Fazeli, and J. Dongarra, An Asynchronous Algorithm on NetSolve Global Computing System, Future Generation Computing Systems, Vol. 22, No. 3(2005), pp.279–290.

(12) Von Welch, Globus Toolkit Firewall Requirements version9(2006), (http://www.globus.org/toolkit/security/firewalls/Globus-Firewall-Requirements-9.pdf)

(13) H.Okuda and S.Kudo, Parallel Finite Element Flow Computations for SMP Clusters, Proc. Sixth Japan-US International Symposium on Flow Simulation and Modeling,(2002), pp.127-130.

(14) H.Okuda, MPI/OpenMP Hybrid Parallel Finite Element Analysis for Incompressible Frow Using Full Nodes of Hitach SR8000/MPP, Supercomputing News, Vol.4, No.3(2002),pp.47-53, Information Technology Center, The University of Tokyo.

(15) L. Chen, K. Nakajima, and I. Fujishiro, Parallel Computational Steering Using MxN Communication Model on HPC-MW, Proceedings of the Conference on Computational Engineering and Science, Vol.9, (2004), pp.867-868.