Regular Paper

Implementation and Evaluation of Multiple GridRPC Services for Molecular Dynamics Simulations of Proteins

Takashi Amisaki† and Shin-ichi Fujiwara†

This paper reports a protein-simulation grid that uses grid remote procedure calls (GridRPCs) to a special-purpose cluster machine for molecular dynamics simulations. The grid was implemented using Ninf-G, Torque, LAM, and the Globus Toolkit. To avoid the inefficiency of a single GridRPC session using all the nodes of the cluster, we designed the grid so that it works efficiently when multiple GridRPC sessions share the cluster. This was done by putting the dedicated nodes (PCs with special computation boards) under the management of the Torque system, thus enabling the manager to dynamically configure a cluster with the requested number of dedicated nodes. In addition, a new job type was added to the Globus toolkit and new backend procedure was added to Ninf-G. The Ninf-G stub was separated from processes that actually perform the force evaluation on the dedicated nodes. Simulations for two proteins gave promising results. Simulations performed using a four-node cluster and a 100-Mbps LAN for GridRPC sessions were 4.6–17.0 times faster than the same simulation performed on the local client PC, while their communication overhead was less than 20% of total execution time. Even when the the four-node cluster machine was shared between two distinct simulations of proteins, the two GridRPC communications did not interfere with each other. This showed the efficacy of multiple GridRPC sessions.

1. Introduction

Proteins are essential biomolecules, and understanding their properties is important for understanding diseases and designing therapeutic agents. Extensive efforts are therefore being devoted to analyzing their structures and functions. The three-dimensional structures of many proteins, for example, have been determined experimentally by using X-ray diffraction methods and nuclear magnetic resonance spectroscopy. The structures of a number of proteins are also being investigated in structural genomics projects.

Among the computational approaches being developed are molecular simulations that provide results complementary to those of conventional experiments. Molecular dynamics (MD) simulations, for example, have been used not only for predicting the structure of a protein and for analyzing the kinetics of protein folding but also for estimating the free energy of drug binding. MD simulations of large proteins, however, require vast computational resources which are not widely available.

Grid computing may circumvent this problem because it can deliver the great computational power of expensive parallel computers to remote researchers. There have, in fact, been several reports of studies on grid computing for molecular simulations. These studies can be roughly divided into two categories. Efforts in the first category aim at building a user interface for legacy application programs such as CHARMM [1] and AMBER [2]. A user interface can be implemented as a portal on a Web server [3] or as a workflow management system executed on a client computer [4]. Such interfaces for MD simulations must enable users to monitor the progress of the remote computation, and we also have already reported a grid-enabled application [5] that lets a user monitor the time evolution of a trajectory being produced by a MD application running on remote machines.

The second category of the studies focuses on the computational grid itself, that is, on parallel molecular simulations using distributed resources or remote resources. Studies exploring the use of parameter sweep applications are typical of this category [6]~[8]. There are also reports of molecular simulations analyzing the kinetics of protein folding by using ensemble/distributed dynamics or replica-exchange dynamics, in both of which a number of nearly independent MD trajectories are simulated by nodes exchanging only a small amount of data [9],[10]. To our knowledge, however, there are few reports of general classical MD simulations on grids even though there is much de-

† Tottori University

mand for such simulations. The most time-consuming part of such simulations is the evaluation of nonbonded interactions (e.g., Coulombic forces). In general, it is not easy for this evaluation to be efficiently parallelized in a slow-speed network.

There are currently at least two possible ways of using grids to evaluate nonbonded interactions: by using the message-passing interface (MPI) on grids [11] and by using the grid remote procedure calls (GridRPCs) [12),13)]. The latter method was used in a grid-enabled MD application reported previously [5] because it enabled force evaluations to easily be made by a high-performance special-purpose cluster machine [14]. That application provided remote clients with the computing power of special computation boards that calculate Coulombic interactions at high speed. The boards were plugged into every node of a PC-cluster to form a dedicated cluster. Dedicated boards generally offer high performance at a cost much less than that of extremely expensive supercomputers, but a cluster system that can simulate larger molecular systems is also extremely expensive because it needs a large number of nodes (and hence large number of the boards) in order to keep the simulation time sufficiently short. Because a dedicated cluster system is only valuable for a particular task, a large-scale dedicated cluster need to be shared by many clients. Grid-enabled use of the dedicated cluster thus seems attractive, provided that the application programming interfaces (APIs) to the cluster are designed appropriately.

A GridRPC function using a dedicated cluster to calculate nonbonded interactions was implemented on a minimal platform for a grid prototype by using Ninf-G [13)], and the possibility of using the GridRPC approach for the MD simulations was examined [5]. Although the results were promising, some potential problems were identified.

One of the most problematic issues was that of the utilization of the special-purpose cluster machine. The machine was idle for much of a GridRPC session while waiting for completion of the transmission of huge amount of data about the coordinates and forces. An effective grid system should benefit clients without wasting its computational resources.

In the present study we demonstrate that the utilization of a dedicated machine can be improved, without reducing the benefits of clients, by serving multiple clients simultaneously. We first discuss this approach formally using a simple cost model of a GridRPC session for a MD simulation and then report the development of a practical grid that can process multiple GridRPC sessions simultaneously. A grid implementing this multiple-GridRPC approach needs a resource manager that keeps track of the available nodes of the dedicated machine and dynamically configures a small cluster with the requested number of dedicated nodes. The grid must also be able to handle the communication through multiple GridRPC connections efficiently. In this paper we demonstrate that such a grid can be built using slight modifications of currently available middleware such as the Globus Toolkit [15)], Ninf-G, Torque [16)], and LAM [17)]. We also report the results of numerical experiments confirming the effectiveness of this approach in actual MD simulations.

2. Dedicated Cluster for MD Simulations

The basic idea behind our approach is to use a special cluster system as the main computing resource in grid for MD simulation. The development of this four-node cluster has already been reported in detail [5], so in this section, we briefly describe issues relevant to the current work.

The dedicated cluster is composed of a homogeneous set of compute nodes, each of which is an ordinary PC but has computing boards designed especially for the evaluation of nonbonded pair interactions (i.e., Coulombic and van der Waals forces). The boards used in this work were MD Engine II (MDE-II) (Fuji Xerox, Co., Ltd) boards.

Force evaluation in the cluster is performed in the following way. A spatial decomposition technique is used to partition the evaluation task into sub tasks mapped to the dedicated compute nodes, on each of which they are split into two parts: evaluations of far and nearby interactions. Each node then evaluates far-field interactions by using the fast multipole method (FMM) [18)], while nearest-neighbor interactions are calculated using the MDE-II boards, in which calculations are performed by four custom processors.

Although the four-node dedicated cluster executed the protein simulations examined in this work 5.6–18.4 times faster than an ordinary PC, its parallel efficiency is limited by its hierarchi-

cal architecture. This kind of architecture is not unique to our cluster system and can also be seen in accelerator-type machines. Because high performance is attained at some cost of efficiency, it seems better to share the cluster with multiple jobs rather than carry out serial jobs using all the nodes of the cluster.

3. Performance of Multiple GridRPC Computations

The multiple-GridRPC approach has two advantages with regard to resource utilization. First, because parallel efficiency is generally a decreasing function of the number of nodes, dividing a large dedicated cluster into small sub clusters enables each of the small clusters to execute its own simulation efficiently. Next, dividing a large dedicated cluster keeps the overhead caused by a GridRPC communication at a small fraction of a total execution time. Low communication overhead is also characteristic of a grid molecular simulator using general-purpose clusters [19].

These two advantages can be confirmed using a simple cost model. The cost of a single MD simulation of an N-atom system over a simulation period T can be modeled as

$$\frac{A(N,T)}{p \cdot E(p)} + L(N,T), \tag{1}$$

where A is the amount of computations to be performed in the simulation. It is proportional to T and is of $O(N)$ or greater, depending on the algorithm used for evaluating the nonbonded interactions. The function E represents the parallel efficiency of the dedicated cluster and is the function of the number of nodes, p. The function L represents the amounts of delay caused by communication over the GridRPC connection and is proportional to both N and T.

Consider two MD simulations S_1 and S_2. The simulation costs of S_1 are $(A, L) = (A_1, L_1)$, those of S_2 are $(A, L) = (A_2, L_2)$. We assume that A_1 and A_2 are similar. The total cost of performing the two GridRPC sessions for S_1 and S_2 serially using a $2p$-node cluster is the sum of the costs of the corresponding two single-GridRPC sessions. On the other hand, when the two GridRPC sessions are performed simultaneously each using a p-node cluster, the cost is the greater of the cost of S_1 or that of S_2. It can be shown that

$$\frac{A_1 + A_2}{2pE(2p)} > \max\left\{\frac{A_1}{pE(p)}, \frac{A_2}{pE(p)}\right\} \tag{2}$$

if

$$\frac{E(p)}{E(2p)} > 1 + \frac{|A_1 - A_2|}{A_1 + A_2}. \tag{3}$$

This relation will hold because E is a monotonically decreasing function of p (and A_1 and A_2 are similar to each other). The obvious relation

$$L_1 + L_2 > \max\{L_1, L_2\} \tag{4}$$

also shows the advantage of the multiple-GridRPC approach, but this relation assumes an ideal environment where multiple GridRPC connections never interfere with each other. Because the validity of this assumption should be confirmed in actual situations, we carried out experiments on multiple GridRPC sessions. The results are presented in Section 6.

4. Brief Descriptions of Ninf-G and the Globus Toolkit

This section briefly explains how a Ninf-G job for a MPI program is executed on a grid using the Globus Toolkit, Ninf-G, Torque, and LAM.

Ninf-G provides methods for executing a library function on a grid using an interface like that provided by remote procedure calls. The function may be defined in C or given as an external object file. An example of an IDL file for the latter case is shown at the end of Section 5.4 (Fig. 7). In either case the IDL file must be processed by the ng_gen command to obtain the source code of a Ninf-G stub program, which is the server program of a GridRPC session. If "MPI" backend is specified in the IDL file, the command ng_gen attaches MPI initialization and termination sequences to that function. The stub generated is a Single-Program-Multiple-Data (SPMD) MPI program.

Ninf-G is built on top of the Globus Toolkit, which comprises of software tools and libraries for monitoring and discovery services, for resource allocation and management, and for the security infrastructure of a grid. Of these, the Globus resource allocation manager (GRAM) is most relevant to the subsequent discussion [☆]. A Ninf-G client uses GRAM functions to send a request for executing the stub program on the server machine. When the client issues a `grpc_initialize()` with the Ninf-G configu-

[☆] In this work, we used the Globus toolkit version 2.4.3, corresponding to the pre-Web service components in version 4.x.

(a)

```
<SERVER>
  hostname mdegk.med.tottori-u.ac.jp
  mpi_runNoOfCPUs 2
  jobmanager jobmanager-pbs
</SERVER>
```

(b)

```
#! /bin/sh
#PBS -l nodes=2
lamboot -ssi boot tm
mpirun C _stub_pi_trial "--client=
    pen68.med.tottori-u.ac.jp:32796" ...
lamhalt
exit 0
```

Fig. 1 Files used or generated when executing a sample MPI program on a grid using the Globus Toolkit, Ninf-G, Torque, and LAM. (a) An example of a Ninf-G configuration file. The number following the directive `mpi_runNoOfCPUs` is the number of nodes to be used. (b) Skeleton of the PBS batch job script built by the Globus jobmanager. In this case the name of the stub executable is `_stub_pi_trial`. The example is involved in the Ninf-G distribution package.

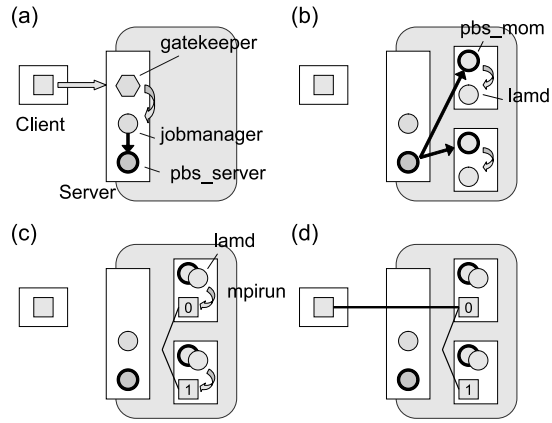

Fig. 2 Schematic representation of a grid that executes MPI jobs using the Globus Toolkit, Ninf-G, Torque, and LAM. (a) `Grpc_initialize` and local job submission. (b) Boot process of a LAM machine. (c) Invocation of SPMD processes using the mpirun command. (d) `Grpc_call` using the connection established between the client and a Ninf-G stub (rank-zero process).

ration file shown in **Fig. 1** (a), the request is processed by a GRAM gatekeeper running on the server `mdegk.med.tottori-u.ac.jp`. The gatekeeper is usually launched by inetd, the superserver for Internet services. When a GRAM request arrives, the gatekeeper creates an jobmanager that uses a qsub command to submit the job to the Torque batch-queueing system server (i.e., pbs_server) [**Fig. 2** (a)]. For this purpose the jobmanager uses a Perl module "pbs.pm" ☆ to create a job-script file that contains the sequence of lamboot, mpirun, and lamhalt commands [Fig. 1 (b)].

The script file first boots a LAM machine by using the TM boot scheme. That is, the lamd process on each node is launched by the node manager (pbs_mom) instead of by rsh/ssh commands [Fig. 2 (b)]. The LAM machine is booted on the nodes that are allocated by the resource manager, with the number of nodes being specified in the directive `mpi_runNoOfCPUs` in the configuration file [Fig. 1 (a)]. The script file then executes the mpirun command to invoke a SPMD process on each node [Fig. 2 (c)]. Finally, a connection between the client and server (i.e., the rank-zero process) is established [Fig. 2 (d)]. The address of the client is passed as an argument to the mpirun command. The SPMD processes stay alive during this GridRPC session and handle all `grpc_call`s.

☆ The file is located at `$GLOBUS_LOCATION/lib/perl/Globus/GRAM/JobManager/pbs.pm`, where `$GLOBUS_LOCATION` indicates the top of the installation tree.

5. Implementation

The organization of our grid for MD simulations is shown schematically in **Fig. 3**. It is basically the same as the grid for MPI jobs that described in Section 4, but to accommodate GridRPC calls to the dedicated cluster, we modified Ninf-G and the Globus Toolkit.

5.1 Client Side Program

In this work, we selected AMBER 7.0 as the MD application program and modified it for the purpose of this work. One of the most important modifications was to replace a call for the evaluation of nonbonded interactions (i.e., `get_nb_energy()`) with a call for a custom-made routine: `anbicalc()`. This C function makes Ninf-G's `grpc_call` to a remote library function named `anbig()`.

The GridRPC-enabled AMBER can be executed by using the same configuration file shown in Fig. 1 (a) and treating the number following the directive `mpi_runNoOfCPUs` as the number of dedicated nodes.

5.2 Resource Management

We attached the static property "mde" to each dedicated node with MDE-II boards by editing the configuration file of the Torque system. This enables the resource manager (pbs_server) to keep track of available nodes

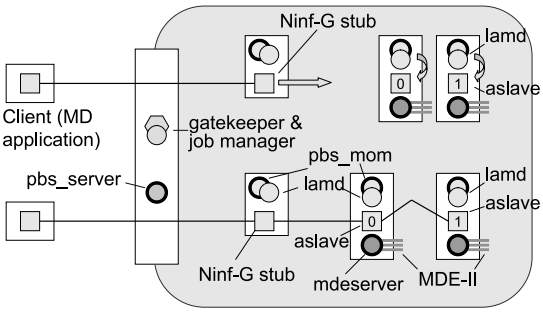

Fig. 3 Organization of the grid. This illustrates two simultaneous GridRPC sessions. The downside stub process has established every connection, while the upside stub is trying to launch "aslave" processes. On the server side, four task groups participate in each GridRPC session: (1) Globus' gatekeeper and jobmanager, (2) processes of the resource management system (pbs_server and pbs_mom), (3) the Ninf-G stub, and (4) SPMD processes (aslave) that calculate non-bonded interactions by using the MDE-II (MD Engine II) boards. The "mdeserver" process running on a node is responsible for the management and mutual exclusion of the MDE-II boards on that node.

and dynamically configure a cluster with the requested number of dedicated nodes[☆]. As will be described later, the property "mde" can be requested by using a PBS resource request statement.

5.3 Ninf-G Stub and Parallel Program for Force Evaluations

In our grid the role of a stub is separated from processes that perform the evaluations of non-bonded interactions (Fig. 3). The task of the force evaluation is instead carried out by the program called "aslave" on the dedicated cluster. The stub process now concentrates on mediating between the client and rank-zero aslave process. This design enabled us to avoid having a direct connection between the special resources and clients. In addition, the site scheduler can place the stub in any node. For example, a couple of stubs can be mapped onto a machine that has access to a high-performance network.

A sketch of the code of the GridRPC library function `anbig()` is shown in **Fig. 4**. The stub process locally calls this function whenever the client sends a request to that function. When the function is called first time, an aslave process is created on each dedicated node by using the `MPI_Comm_spawn()` function. Then and each time the function is called thereafter, the coordinates of the atoms, `xyz`, are sent to the rank-zero aslave process, after which the calculated forces are received from that process and are returned to the calling client through the argument `frc`.

```
int anbig(int cmd, int natoms,
          double xyz[], double frc[])
{
  if (!initialized) {
    MPI_Init(&i, (char ***) MPI_ARGV_NULL);
    MPI_Attr_get(MPI_COMM_WORLD,
        MPI_UNIVERSE_SIZE, &numnodes, &flag);
    nslaves = numnodes - 1;
    Prepare LAM application schema file.
    MPI_Info_create(&info);
    MPI_Info_set(info, "file", schema);
    MPI_Comm_spawn("/path/to/aslave",
        MPI_ARGV_NULL, nslaves, info, 0,
        MPI_COMM_SELF, &aslave, &ierr);
    MPI_Send(&natoms, 1, MPI_INT, 0,
             999, aslave);
    initialized = TRUE;
  }

  if (cmd == EXIT)
    MPI_Send(&EXIT, 1, MPI_DOUBLE, 0,
             999, aslave);

  MPI_Send(xyz, 3 * natoms, MPI_DOUBLE, 0,
           999, aslave);
  MPI_Recv(frc, 3 * natoms, MPI_DOUBLE, 0,
           999, aslave, &ierr);

  return 0;
}
```

Fig. 4 A sketch of the source code of the GridRPC library function. The stub process repeatedly calls this function locally on a node. In the subsequent example of an IDL file (Fig. 7), the code is assumed to be stored in the file `libanbig.c`. The arrays `xyz` and `frc` respectively represent the coordinates and forces. Note that more calling arguments (e.g., atomic charges and species) may be involved in the actual code.

A sketch of the aslave code is shown in **Fig. 5**. The program itself is SPMD. The aslave process with rank zero acts as the coordinator of the SPMD processes. Each aslave process repeatedly receives the coordinates of the atoms (`xyz`) from the stub or rank-zero aslave and then calculates the forces (`frc`) by using the routine `anbi_calc()`, which actually calculates the far-field and nearby interactions by using

[☆] Alternatively, the resource manager could monitor status on a per-board basis rather than a per-node basis. To make this possible, we added a monitoring function to the mdeserver process as shown in Fig. 3). It makes a health check of the MDE-II boards and reports the results to the node manager pbs_mom, which in turn reports the results to the site resource manager pbs_server.

```
int main(int argc, char *argv[])
{

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &mpi_id);
  MPI_Comm_get_parent(&master);

  if (mpi_id == 0)
    MPI_Recv(&natoms, 1, MPI_INT, 0,
             999, stub, &ierr);
  MPI_Bcast(&natoms, 1, MPI_INT, 0,
            MPI_COMM_WORLD);

  for (;;) {
    if (mpi_id == 0)
      MPI_Recv(&cmd, 1, MPI_DOUBLE, 0,
               999, stub, &ierr);
    MPI_Bcast(&cmd, 1, MPI_DOUBLE, 0,
              MPI_COMM_WORLD);
    if (cmd == EXIT) break;

    if (mpi_id == 0)
      MPI_Recv(xyz, 3 * natoms, MPI_DOUBLE, 0,
               999, stub, &ierr);
    MPI_Bcast(xyz, 3 * natoms, MPI_DOUBLE, 0,
              MPI_COMM_WORLD);

    /* force evaluation using FMM & MDE-II */
    anbi_calc(xyz, frc);

    if (mpi_id == 0)
      MPI_Send(frc, 3 * natoms, MPI_DOUBLE, 0,
               999, stub);
  }

  MPI_Finalize();

  return 0;
}
```

Fig. 5 A sketch of the source code of the "aslave" program. The rank-zero process acts as a mediator between the stub and SPMD processes. It also acts as a coordinator for the SPMD processes. The task of force evaluation is performed in accordance with the Replicated-Data Spatial-Decomposition scheme. Notice that the forces are summed up in the routine `anbi_calc()` by using parallel reduction (`MPI_Reduce`).

FMM and the MDE-II boards, respectively, in accordance with the Replicated-Data Spatial-Decomposition scheme. In this routine, all aslave processes participate in parallel reduction operations for summing up forces. The rank-zero process finally sends the forces back to the stub process.

5.4 Modifications of Ninf-G and the Globus Toolkit

We modified the GRAM job manager in Globus Toolkit 2.4.3 to accommodate the GridRPC calls to the dedicated cluster. Specifically, we added the special job type *mde* to the original ones such as *single*, *mpi*, and *condor* ☆. We also added some codes to the file `pbs.pm` so that the Perl module would generate a job script like that shown in **Fig. 6** instead of the one shown in Fig. 1 (b). Note the following two differences between the two kinds of scripts. One of the differences is that regarding the PBS resource requirement. When a "mde" type of job is specified, the generated script contains a line

```
#PBS -l nodes=1+2:mde
```

indicating a request for two dedicated nodes with MDE-II boards plus one additional node. A Ninf-G stub will be executed on this additional node. The number of dedicated nodes is taken from the value (`mpi_runNoOfCPUs`) specified in the Ninf-G configuration file on the client side.

```
#! /bin/sh
#PBS -l nodes=1+2:mde
lamboot -ssi boot tm
_stub_anbig "--client=
    pen68.med.tottori-u.ac.jp:33427" ...
lamhalt
exit 0
```

Fig. 6 Skeleton of a PBS batch job script generated when the backend MDE is specified.

The other difference is that when the backend MDE is specified the stub process is executed directly rather than being invoked by the mpirun command. As a result, as shown in Fig. 3, only a single stub process is created on the first node of the LAM machine, which may be the node without an MDE-II board as indicated by the PBS resource requirement. The stub process creates an aslave process on each remaining MDE-II node by using the `MPI_Comm_spawn()` function (Fig. 3 and Fig. 4). Notice that it is easy to change the mapping policy so that, for example, the stub can be put on the first MDE-II node together with the aslave process or can be put on another node together with the stubs of other jobs.

To enable the startup of the stub process in the above-mentioned manner, Ninf-G also had to be modified. Every file that handles the backend MPI, except the file `nggenStub.c`, was modified so that a newly added backend "MDE" would be treated in the same manner as the backend MPI. In the file `nggenStub.c` the backend MDE is treated as the backend

☆ These job types are defined in `$GLOBUS_LOCATION/share/globus_gram_job_manager/globus-gram-job-manager.rvf`.

```
Module anbig;
Define anbig(IN int cmd, IN int natoms,
             IN double xyz[3*natoms],
             OUT double frc[3*natoms])
Required "libanbig.o"
Backend "MDE"
Calls "C" anbig(cmd, natoms, xyz, frc);
```

Fig. 7 An IDL file for the backend MDE. In this example, the C function `anbig()` is assumed to be defined in another file `libanbig.c` (see Fig. 4).

NORMAL, rather than MPI, to prevent the ng_gen command from attaching MPI code. Because of this modification, a GridRPC function can no longer be written in an IDL file. The function must be given as an external object file. A simplified version of the IDL file used in this work is shown in **Fig. 7**. The source code of the C function, `anbig()` in this case, is assumed to be defined in the source file `libanbig.c`, which is shown in Fig. 4.

6. Experiments

The main purpose of the tests was to confirm the advantage of multiple GridRPC MD sessions that was implied by the cost model presented in Section 3. Of comparable importance was an examination of the efficacy of the grid when the client and server were connected through a broadband network with a speed of 100 Mbps or higher. In the previous assessment of the performance of our grid-enabled MD application [5], we used a 10-Mbps connection to emulate a low-speed network between the client and server sites. Larger interconnection bandwidths, however, are already likely to be used in such applications. We therefore used 100-Mbps Ethernet for the client/server connection in the work reported here.

6.1 Benchmark System Configuration

The basic configuration was that shown in Fig. 3. The dedicated cluster was composed of four PCs (Pentium 4, 2 GHz, Scientific Linux ver 3.05), each with three MDE-II boards. Two front-end machines were used for executing the stubs. They were PCs with the same specifications as those of the dedicated nodes but without MDE-II boards. These six PCs were interconnected using both Fast Ethernet (100 Mbps) and Gigabit Ethernet (1,000 Mbps) technologies. The latter was used for the LAM/MPI traffic. An additional PC with the same specification was used as the Globus server. It also acted as a NAT (network-address-translation) router between the six PCs and another two client PCs (Pentium 4, 2 GHz). The router and the other PCs were interconnected using 100-Mbps Ethernet technology. Scientific Linux 3.05 was installed on all of the PCs.

The system simulated were those comprising a dihydrofolate reductase (DHFR) molecule surrounded by a sphere of water molecules (N = 20,501) or a human serum albumin (HSA) molecule surrounded by a sphere of water molecules (N = 82,804). The MD application program was the sander module in the AMBER 7.0 package and was modified for the purpose of this work. For the evaluation of electrostatic interaction, two types of FMM computation were examined: one designated FMM-I and using direct calculations for nearest neighbor cells, the other designated FMM-II' and including the second-nearest neighbors in the direct calculations. The prime symbol indicates that, if possible, the cells were aggregated to form a larger cell at the parent level. In both FMM computations, the level of cell hierarchy was 3 and the degree of expansion was 8. Under these conditions, the force calculated using FMM-II' was estimated to be about one significant figure more accurate than that calculated using FMM-I. The direct calculations were performed using the MDE-II boards, as were van der Waals interactions with a cutoff radius of 12.5 Å for the DHFR system and 15.0 Å for the HSA systems.

6.2 Results

Timing results are summarized in **Table 1**, where the values in the column labeled "Eval" are the average times spent on the dedicated cluster for the evaluation of nonbonded (Coulombic and van der Waals) interactions at a single MD step. The values in the "GRPC" and "Total" columns are respectively the average time for a single GridRPC call and the average time to advance a MD step. These values are in seconds and are averaged over 1,000 steps. Each of the values in the column "PE" column is the parallel efficiency of the dedicated cluster. We defined the delay due to the GridRPC communication as the difference between GRPC and Eval, and we listed in the "Delay" column this difference as a fraction of corresponding Total value.

When the four simulations were performed locally on the client PC for comparison, the Total times were 4.65 s when using FMM-I for DHFR, 10.70 s when using FMM-II' for DHFR, 29.90 s when using FMM-I for HSA, and 96.76 s when

Table 1 Timing results for the simulations carried out as single GridRPC sessions.

			Time/step (s)				
Protein	FMM	Nodes	Eval	GRPC	Total	PE	Delay
DHFR	I	1	2.43	2.61	2.64	—	0.07
DHFR	I	2	1.32	1.51	1.54	0.92	0.12
DHFR	I	4	0.80	0.98	1.02	0.76	0.19
HSA	I	1	7.51	7.95	8.09	—	0.06
HSA	I	2	4.04	4.48	4.62	0.93	0.10
HSA	I	4	2.43	2.87	3.00	0.77	0.15
DHFR	II'	1	3.05	3.24	3.27	—	0.06
DHFR	II'	2	1.65	1.83	1.87	0.92	0.10
DHFR	II'	4	0.97	1.16	1.19	0.79	0.16
HSA	II'	1	18.26	18.70	18.83	—	0.02
HSA	II'	2	9.44	9.89	10.02	0.97	0.04
HSA	II'	4	5.13	5.58	5.70	0.89	0.08

using FMM-II' for HSA.

In general, the more nodes that were used, the greater improvement in speed that was obtained on the client side. In the case of the HSA simulation (four nodes, FMM-II'), the GridRPC approach reduced the time required to 1/17 of the time required for local execution on the client PC. This shows the effectiveness of the approach.

As expected, however, parallel efficiency decreased as the number of nodes increased. The fraction of GridRPC communication also increased as the number of nodes increased, although only by 2–19%, an amount that seems acceptable. The increase can be accounted for by the increase in Total time. The difference between GRPC and Eval is almost constant for each molecular system, 0.18–0.19 s for the DHFR simulations and 0.44–0.45 s for the HSA simulations. The values are consistent with the volumes of data transferred, since at every time step the amounts of data transferred along with a single GridRPC call was 7.5 Mbit in the DHFR simulations and 30.3 Mbit in the HSA simulations.

To confirm the advantage of concurrent multiple-GridRPC services that is implied by the cost model, we carried out a 3,000-step DHFR simulation and a 1,000-step HSA simulation simultaneously. Notice that both simulations would take about 4,600 s. Two dedicated nodes were used for each simulation, and FMM-I was used in both simulations. Concerning 2-node computation, the selected pair of simulations was the one of the simulations listed in Table 1 that would maximally interfere with each other. The results for the pair simulations are listed in **Table 2**. The results indicate that the execution time and fraction of delay both coincided with those in the 2-node computation of the single GridRPC session. In other words, the GridRPC communications of the two simulations did not interfere with each other.

Table 2 Timing results for the two simulations performed simultaneously.

		Time/step (s)			
Protein	Steps	Eval	GRPC	Total	Delay
DHFR	3,000	1.32	1.51	1.54	0.12
HSA	1,000	4.04	4.49	4.62	0.10

7. Concluding Remarks

This paper reported the design and implementation of a grid system for MD simulations in the framework of GridRPC as provided by Ninf-G 2.4.0 but with some modifications. The basic concept of our approach was to deliver the computing power of the cluster system for MD simulation through GridRPC calls. Each of the cluster nodes uses special computing boards that calculate nonbonded interactions between the atoms constituting the molecular system. The inefficiency due to a single GridRPC session using all the nodes of the cluster was avoided by designing the grid so that it works efficiently in multiple GridRPC sessions. This was done by putting the dedicated nodes under the management of the Torque system. In addition, a Ninf-G backend procedure was designed so as to handle communication to a client efficiently in the case of multiple GridRPC sessions. As the cost model implies, the multiple-GridRPC approach improves parallel efficiency and the processing/communication ratio in GridRPC calls.

We performed benchmark tests on two protein systems to confirm these improvements. The results seemed promising, although we recognize that the scope of our tests was limited in

some respects. In a practical situation, the effects of the multiple-GridRPC approach might be influenced by many factors, such as the number of GridRPC requests, the number of dedicated nodes, and the sizes of molecular systems. The performance of a job scheduler would be also important because the efficiency depends on how evenly jobs are distributed across clusters (see Eq. 3). Nevertheless, the results show the effectiveness of the multiple-GridRPC approach for reducing the time required for MD simulations.

The method of this work was developed under the presumption that a large number of dedicated nodes would be available when actually building the grid. Then the large-scale dedicated cluster would be dynamically divided into small subclusters to circumvent the issue of parallel efficiency. As mentioned previously, we also assumed that the server and clients are connected through a high-speed network. When such is not available, the GridRPC approach could be modified by incorporating an idea that can be found in the work of Boku, et al. [20]. In their design for a gravity calculation system (i.e., HMCS-G), the dedicated cluster was treated as an single-atomic resource and was shared between multiple clients in a time-sharing manner. Multiple sessions were handled by a single stub process. The process also played the role of a data buffer, which dramatically improved efficiency when a large number of GridRPC connections were formed on a low-speed network. This idea can be incorporated into the grid presented in this paper to improve the efficiency in such cases. Specifically, a subcluster of the dedicated cluster may be further shared in a time-sharing manner. Even in such a case, we could basically use the grid presented in this work. Some modification, however, would be needed. We would first have to modify the aslave process so that each process acquires and releases the MDE-II boards at each MD time step. To avoid deadlock, the "mdeserver" process would have to be extended to perform mutual exclusion cluster-wide rather node-wide. In addition, modification of the dedicated board would be required. The current MDE-II boards can store only a limited number of simulation settings (i.e., force field parameters and pair-interaction data), and significant time is required for rewriting them. Hence, for the time-sharing approach to be effective in MD simulations, a method for rapidly switching between many molecular systems should be provided.

In this work, the Ninf-G stub process was separated from SPMD processes on the dedicated nodes for the force evaluation. As a result, it is possible to place stub processes, for example, on other dedicated machines for networking. As mentioned previously, the idea of separating the stub from the SPMD processes can be also seen in the HMCS-G system, where the stub was expected to play the roles of traffic scheduler. In this work, in contrast, the task of traffic scheduling is left to a site-wide manager rather than making a single stub responsible for that task. The design of our grid-enabled MD application may provide an opportunity to efficiently schedule traffic in multiple GridRPC sessions, but such issues are left for future work.

As described previously, there are several application for MD simulations of proteins. Free energy calculations, for example, have been carried out by many researchers. The MD simulation of this type can be performed effectively by using the grid reported in this paper, provided there are stable connections between the clients and the server. Because such an application requires the connections to be maintained throughout millions of GridRPC calls, issues regarding fault tolerance should be addressed before the grid is actually used.

Some MD applications, on the other hand, require many more time steps. To fully sample the dynamics of a protein, for example, a MD simulation should be performed at least over tens of nanoseconds, corresponding to 10^8 GridRPC calls. Such simulations require far more computing resources than were available in the four-node dedicated cluster used in this work. Fortunately, a new generation of computing boards is being brought to market[☆]. Such boards will shorten the execution time dramatically. In such situations, the importance of the multiple-GridRPC approach will further increase.

Acknowledgments This study was supported in part by Grant-in-Aid Scientific Research on Priority Area "Informatics" (Area #006). We are grateful to Mr. Toshiroh Shimada (Fuji Xerox Co., Ltd.) and Mr. Hajime Fukuzawa (NEC Corporation) for their

[☆] To our knowledge, there are currently two new machines: MDG3-system (SGI Japan, Ltd.) built with MDGRAPE-3 boards (RIKEN) and MD Server (NEC Corp.).

assistance in carrying out preliminary benchmark tests, the results of which are not reported in this paper.

References

1) Brooks, B.R., Bruccoleri, R.E., Olafson, B.D., States, D.J., Swaminathan, S. and Karplus, M.: CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations, *J. Computat. Chem.*, Vol.4, pp.187–217 (1983).
2) Case, D.A., Cheatham, T.E., Darden, T., Gohlke, H., Luo, R., Merz, K.M., Onufriev, A., Simmerling, C., Wang, B. and Woods, R.: The Amber Biomolecular Simulation Programs, *J. Computat. Chem.*, Vol.26, pp.1668–1688 (2005).
3) Wilter, A., Osthoff, C., Oliveira, C., Gomes, D.E.B., Hill, E., Dardenne, L.E., Barros, P.M., Loureiro, P.A.A.G.L., Novaes, R. and Pascutti, P.G.: The BioPAUÁ Project: A Portal for Molecular Dynamics Using Grid Environment, *Brazilian Symposium on Bioinformatics*, pp.214–217 (2005).
4) Jeong, K., Kim, D., Kim, M.H., Hwang, S., Jung, S., Lim, Y. and Lee, S.: A Workflow Management and Grid Computing Approach to Molecular Simulation-Based Bio/Nano Experiments, *Lecture Notes in Computer Science*, Vol.2660, pp.1117–1126 (2003).
5) Amisaki, T. and Fujiwara, S.: Development of a Dedicated PC-Cluster for Molecular Dynamics Simulations and Its Application in Computational Grids (in Japanese), *IPSJ Transactions on Computing Systems*, Vol.45, No.SIG 6(ACS 6), pp.244–253 (2004).
6) Natrajan, A., Humphrey, M.A. and Grimshaw, A.S.: The Legion Support for Advanced Parameter-Space Studies on a Grid, *Future Generation Computer Systems*, Vol.18, pp.1033–1052 (2002).
7) Sudholta, W., Baldridgea, K.K., Enticottc, D.A.C., Garicc, S. and Nguyenb, C.K.D.: Application of Grid Computing to Parameter Sweeps and Optimizations in Molecular Modeling, *Future Generation Computer Systems*, Vol.21, No.1, pp.27–35 (2005).
8) Taufer, M., Crowley, M., Price, D.J., Chien, A.A. and Brooks III, C.L.: Study of a Highly Accurate and Fast Protein-Ligand Docking Method Based on Molecular Dynamics, *Concurrency and Computation: Practice and Experience*, Vol.17, No.14, pp.1627–1641 (2005).
9) Pande, V.S., Baker, I., Chapman, J., Elmer, S.P., Khaliq, S., Larson, S.M., Rhee, Y.M., Shirts, M.R., Snow, C.D., Sorin, E.J. and Zagrovic, B.: Atomistic Protein Folding Simulations on the Submillisecond Time Scale Using Worldwide Distributed Computing, *Biopolymers*, Vol.68, No.1, pp.91–109 (2003).
10) Wang, W., Chen, G., Chen, H. and Yang, S.: A Grid Computing Framework for Large Scale Molecular Dynamics Simulations, *Lecture Notes in Computer Science*, Vol.3032, pp.645–648 (2004).
11) Karonis, N., Toonen, B. and Foster, I.: MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface, *Journal of Parallel and Distributed Computing*, Vol.68, No.5, pp.551–563 (2003).
12) Seymour, K., Nakada, H., Matsuoka, S., Dongarra, J., Lee, C. and Casanova, H.: Overview of GridRPC: A Remote Procedure Call API for Grid Computing, *GRID '02: Proc. Third International Workshop on Grid Computing*, London, UK, Springer-Verlag, pp.274–278 (2002).
13) Tanaka, Y., Nakada, H., Sekiguchi, S., Suzumura, T. and Matsuoka, S.: Ninf-G: A Reference Implementation of RPC-based Programming Middleware for Grid Computing, *Journal of Grid Computing*, Vol.1, No.1, pp.41–51 (2003).
14) Amisaki, T., Toyoda, S., Miyagawa, H. and Kitamura, K.: Development of hardware accelerator for molecular dynamics simulations: A computation board that calculates nonbonded interactions, *J. Computat. Chem.*, Vol.24, No.5, pp.582–592 (2003).
15) Foster, I. and Kesselman, K.: Globus: A metacomputing infrastructure toolkit, *International Journal of Supercomputer Applications*, Vol.11, pp.115–128 (1997).
16) Cluster Resources, Inc.: TORQUE Resource Manager. http://www.clusterresources.com/pages/products/torque-resource-manager.php
17) Burns, G., Daoud, R. and Vaigl, J.: LAM: An Open Cluster Environment for MPI, *Proc. Supercomputing Symposium*, pp.379–386 (1994).
18) Greengard, L.: *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge (1988).
19) Gervasi, O., Dittamo, C. and Laganà, A.: A Grid Molecular Simulator for E-Science, *Lecture Notes in Computer Science*, Vol.3470, pp.16–22 (2005).
20) Boku, T., Sato, M., Onuma, K., Makino, J., Takahashi, H.S.D. and Umemura, M.: HMCS-G: Grid-enabled Hybrid Computing System for Computational Astrophysics (in Japanese), *IPSJ Transactions on Computing Systems*, Vol.44, No.SIG 11(ACS 3), pp.1–13 (2003).

(Received January 27, 2006)
(Accepted May 23, 2006)

Takashi Amisaki was born in 1959 and received Master's and Ph.D. degrees from Osaka University in 1983 and 1992. After graduate school he worked at the Dept. of Hospital Pharmacy in Tottori University before moving to Shimane University in 1991. In 2000, he obtained his current position as a professor at Tottori University. His current research interests include molecular simulations of biological macromolecules, high-performance computing, and biomedical data analysis. He is a member of many professional societies, including the IPSJ, IEICE, ACM, Pharmaceutical Society of Japan, and Biophysical Society of Japan.

Shin-ichi Fujiwara was born in 1976 and received Master's and Ph.D. degrees from Kyoto University in 2000 and 2003. Since 2003 he has been a research associate at Tottori University, where his research interests include molecular dynamics simulations of pharmacokinetic-related proteins. He is a member of the Pharmaceutical Society of Japan, Japanese Society for the Study of Xenobiotics, and Biophysical Society of Japan.