Sustainable Adaptive Grid Supercomputing: Multiscale Simulation of Semiconductor Processing across the Pacific

Hiroshi Takemiya*, Yoshio Tanaka*, Satoshi Sekiguchi*
Grid Technology Research Center,
National Institute of Advanced Industrial Science and Technology, Japan

Shuji Ogata+
Graduate School of Engineering, Nagoya Institute of Technology, Japan

Rajiv K. Kalia**, Aiichiro Nakano**, Priya Vashishta**
Department of Computer Science, Department of Physics & Astronomy,
Department of Chemical Engineering & Materials Science,
University of Southern California, USA

Abstract

We propose a reservation-based sustainable adaptive Grid supercomputing paradigm to enable tightly coupled computations of considerable scale (involving over 1,000 processors) and duration (over tens of continuous days) on a Grid of geographically distributed parallel supercomputers. The paradigm is demonstrated for an adaptive multiscale simulation application, in which accurate but compute-intensive quantum mechanical (QM) simulations are embedded within a classical molecular dynamics (MD) simulation only when and where high fidelity is required. Key technical innovations include: 1) an embedded divide-and-conquer algorithmic framework to maximally expose data and computation localities for enhanced scalability; 2) a buffered-cluster hybridization scheme to adaptively adjust MD/QM boundaries to maintain the model accuracy; and 3) a hybrid Grid remote procedure call (GridRPC) + message passing interface (MPI) Grid application framework to combine flexibility (adaptive resource allocation and migration), fault tolerance (automated fault recovery), and efficiency (scalable management of large computing resources). We have achieved an automated execution of multiscale MD/QM simulation on a Grid consisting of 6 supercomputer centers in Japan and the US (in total of 150 thousand processor-hours) for the dynamic simulation of implanted oxygen atoms in a silicon substrate, in which the number of processors changes dynamically on demand and resources are allocated and migrated dynamically according to both reservations and unexpected faults. The simulation results reveal a strong dependence of the oxygen penetration depth on the incident oxygen-beam position, which is useful information to further advance SIMOX (separation by implanted oxygen) technique to fabricate high speed and low power-consumption semiconductor devices.

Keywords: Grid application, multiscale simulation, molecular dynamics, quantum mechanics, density functional theory, Grid remote procedure call, message passing interface

* (h-takemiya, yoshio.tanaka, s.sekiguchi)@aist.go.jp
+ ogata@nitech.ac.jp
** (rkalia, anakano, priyav)@usc.edu

SC2006 November 2006, Tampa, Florida, USA
0-7695-2700-0/06 $20.00 ©2006 IEEE

1 Introduction

Grid computing [1] is playing increasingly more critical roles in advancing science and engineering [2-4]. The use of a Grid of geographically distributed parallel supercomputers to scientific computing has traditionally been limited to embarrassingly parallel applications such as replica-based Monte Carlo simulations [5], because of the limited network bandwidth and large latency of Grid. Fortunately recent extensions [6,7] of divide-and-conquer algorithms promise to make a wide variety of tightly coupled parallel applications scalable even in Grid environments. One promising simulation approach has emerged at the forefront of computational sciences, with broad applications in biology, materials science, and nanotechnology. In the multiscale simulation [8,9], accurate but compute-intensive quantum mechanical (QM) simulations are embedded to handle chemical reactions (below a length scale of 10^{-8} m) within a classical molecular dynamics (MD) simulation to describe large-scale atomistic processes (up to a length scale of 10^{-6} m), only when and

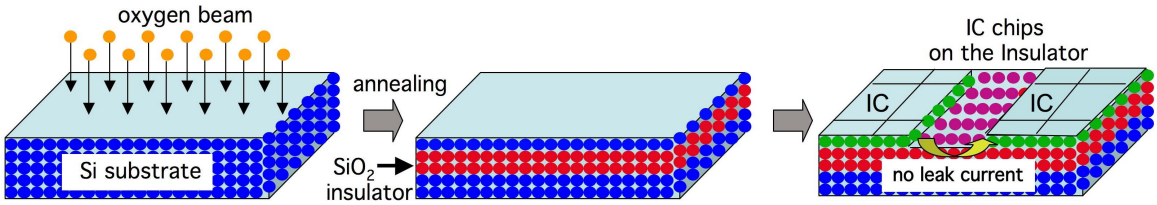

Figure 1: Schematic of the SIMOX technique to create SOI structures. The IC chips fabricated on the insulator layer exhibit excellent performance due to the absence of leak current.

where high fidelity is required. For example, we have developed a highly scalable multiscale MD/QM simulation algorithm based on a divide-and-conquer approach, thereby achieving a parallel efficiency as high as 0.94 on three Linux clusters in the US and Japan [10].

Our previous multiscale MD/QM simulation [10] was implemented using the Grid-enabled message passing interface (MPI), MPICH-G2 [11]. However, recent advancements in multiscale simulation technologies have made this approach inadequate, posing several technical challenges:

Flexibility: Our latest adaptive multiscale MD/QM simulation approach allows the size of embedded QM simulations to change automatically during the simulation so as to maintain the model accuracy. This will require the number of processors change dynamically on demand, and accordingly, resources be allocated and migrated dynamically, which was not possible with the MPICH-G2 implementation.

Scalability: Advanced multiscale simulations are run on thousands of processors for months. While such "sustained" supercomputing is typically performed on a limited number of highest-end computers, it is extremely difficult to obtain such sustained access to large computational resources, especially for university-based researchers. While a Grid could in principle provide requisite computing resources, it is a challenge to achieve scalability on thousands of processors distributed globally (hence large latency ~ 100 ms and low bandwidth < Gbps) for tightly coupled parallel applications.

Fault tolerance: It is necessary to automatically recover from faults, the probability of which increases for thousands of processors distributed over wide-area networks during months of sustained Grid computing.

To address these challenges, we propose a ***reservation-based sustainable Grid supercomputing paradigm***, in which supercomputers that constitute the Grid are changed dynamically according to a reservation schedule. Key technical features of our approach include:

- An embedded divide-and-conquer algorithmic framework to maximally expose data and computation localities, thereby enabling highly scalable MD and QM simulations in Grid environments.
- A buffered-cluster hybridization scheme to enable adaptive multiscale simulations, in which the size of embedded QM simulations is automatically changed during the simulation so as to maintain the model accuracy.
- A hybrid Grid remote procedure call (GridRPC) + MPI Grid application framework to combine flexibility and scalability.

We have achieved an automated execution of multiscale MD/QM simulation on a Grid consisting of 6 supercomputer centers in Japan and the US (in total of 150 thousand processor-hours), in which the number of processors change dynamically on demand and resources are allocated and migrated dynamically according to both reservations and unexpected faults.

The Grid simulation addresses a problem of significant technological importance. Modern design of high-performance devices focuses on controlling structures at diverse length scales from atomic to macroscopic, and thus multiscale simulations are expected to play an important role in scaling down engineering concepts to nanometer scales. An example is the SIMOX (separation by implantation by oxygen) technique for fabricating silicon on insulator (SOI) structures consisting of a thin layer of Si separated from the bulk substrate by a thin insulator layer of SiO_2 (Fig. 1). The integrated circuits fabricated on the thin Si layer can operate at a high clock speed with a low power supply, due to the absence of leak current through the insulator layer, and hence are suitable for portable products, such as laptops, hand-held devices, and other applications operated by battery power. Oxygen beams with mean implantation energy of $10^{2\text{-}3}$ keV have been used to fabricate SiO_2 layers at the depth of 10^{-5} m. Advancement of the SIMOX technique to create a thinner Si layer of 10^{-8} m width on the SiO_2 layer will require microscopic understanding of the migration processes of oxygen at a much smaller implantation energy, $10^{2\text{-}3}$ eV. Our Grid MD/QM simulation for implantation of oxygen atoms toward a Si substrate has revealed a strong dependence of the oxygen penetration depth on the incident beam position.

This paper is organized as follows. In the next section, we describe our scalable and adaptive multiscale MD/QM simulation algorithms. Section 3 discusses the hybrid GridRPC + MPI Grid application framework. Results of test-bed experiments based on the reservation-based Grid supercomputing paradigm are given in Sec. 4, and Sec. 5 contains conclusions.

2 Scalable and adaptive multiscale simulation algorithms

We have designed a scalable multiscale MD/QM simulation algorithm based on an embedded divide-and-conquer (EDC) algorithmic framework [12]. In EDC algorithms, spatially localized subproblems are solved in a global embedding field, which is efficiently computed with tree-based algorithms. Examples of the embedding field are the electrostatic field in MD simulations and the self-consistent Kohn-Sham potential in QM simulations based on the density functional theory (DFT) [13,14]. We have used the EDC framework to design linear-scaling algorithms for: 1) DFT calculation on adaptive multigrids [7]; 2) chemically reactive MD based on a fast ReaxFF (F-ReaxFF) algorithm [12]; and 3) classical MD based on a space-time multiresolution MD (MRMD) algorithm [15]. The EDC-DFT algorithm employs multigrid preconditioning [16] of an iterative solver for electronic wave functions represented on real-space grid points [17], which are augmented with adaptively generated fine grids around the atoms to accurately operate ionic pseudopotentials.

In recent benchmark tests on 1920 Intel Itanium2 processors, we have demonstrated 1.4 million-atom (0.12 trillion grid points) DFT, 0.56 billion-atom F-ReaxFF, and 18.9 billion-atom MRMD calculations. The EDC algorithms expose maximal data and computation localitites and consequently have achieved parallel efficiency as high as 0.953 on 1920 processors [12].

The EDC framework has also been used to design scalable multiscale MD/QM simulation algorithms, in which divide-and-conquer DFT calculations (or a number of QM cluster calculations) are embedded in an MD simulation. In our additive hybridization scheme [18-23], the total energy is a linear combination of MD and QM energies,

$$E = E_{\mathrm{MD}}^{\mathrm{system}} + \sum_{\mathrm{cluster}} [E_{\mathrm{QM}}^{\mathrm{cluster}}(\{\mathbf{r}_{\mathrm{QM}}\}) - E_{\mathrm{MD}}^{\mathrm{cluster}}(\{\mathbf{r}_{\mathrm{QM}}\})], \quad (1)$$

where $E_{\mathrm{MD}}^{\mathrm{system}}$ is the classical MD energy for the entire system, $E_{\mathrm{QM}}^{\mathrm{cluster}}$ is the QM energy for an atomic cluster, and $E_{\mathrm{MD}}^{\mathrm{cluster}}$ is the MD potential energy of the cluster. In Eq. (1), $\{\mathbf{r}_{\mathrm{QM}}\}$ is the set of positions of the QM atoms. Other physical quantities, such as interatomic forces, are derived from Eq. (1) as a linear combination as well. This modular hybridization scheme not only allows the reuse of existing MD and QM codes but also minimizes the interdependency and communication between MD and QM modules.

We have also developed a buffered-cluster method [23] for accurate hybridization of the MD and QM regions. In this method, buffer atoms are introduced on the surface of the cluster for the calculations of $E_{\mathrm{QM}}^{\mathrm{cluster}}$ and $E_{\mathrm{MD}}^{\mathrm{cluster}}$ to minimize errors arising from the finite size of the QM region. The positions of the buffer atoms in calculating $E_{\mathrm{MD}}^{\mathrm{cluster}}$ are determined to minimize $E_{\mathrm{MD}}^{\mathrm{cluster}}$ for a given set of $\{\mathbf{r}_{\mathrm{QM}}\}$, which are exploited to set buffer atoms in calculating $E_{\mathrm{QM}}^{\mathrm{cluster}}$ in a similar manner. The buffered-cluster method is applicable to a wide range of ceramics and semiconductor materials for any reasonable choice of the QM region. The accuracy of the buffered-cluster method has been tested for crystalline Si and alumina systems, where little difference around the MD-QM boundaries was found in both relaxed configurations of the atoms and recoil forces on them due to their trial displacements. The insensitivity of the atomic forces to the choice of the QM region in the buffered-cluster method allows the QM region to be redefined adaptively during a hybrid simulation run [20-23].

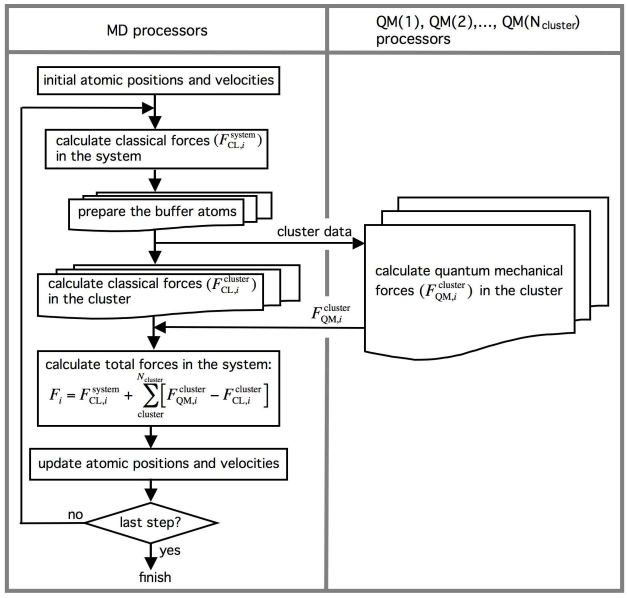

Figure 2: A flowchart of parallel computation in the multiscale MD/QM simulation.

The multiscale MD/QM simulation algorithm has been implemented on parallel computers, by first dividing processors into the MD and QM calculations (task decomposition) and then using spatial decomposition within each task. The additive hybridization scheme makes the MD and QM subtasks entirely independent except for the exchange of cluster-atom coordinates and calculated forces, as shown in the flowchart in. 2. The MD processors compute the energy and forces of the entire system and send the atomic coordinates of the QM clusters with the buffer atoms to each of the QM processor groups. Subsequently, the MD and QM processors independently perform the MD and QM computations on the atomic clusters. The QM energy and forces are then returned to the MD processors, where the total energy and corresponding forces are calculated and the equations of motion are integrated to update the atomic positions and velocities. The communications between the MD and QM processors are

minimal, since the MD processors only need to send several hundred atomic coordinates to each QM cluster, which in return sends back the calculated several hundred force components.

3 Grid implementation

3.1 Requirements for large-scale, long-run, and adaptive applications

Our goal is to enable Grid applications that: 1) require hundreds to thousands of processors distributed over wide areas; 2) run over a long period, typically for a month to a year; and 3) change problem sizes during the simulation. To support such large-scale, long-run, and adaptive Grid applications, the following three requirements should be satisfied:

Flexibility: The application should be able to allocate computing resources dynamically and switch target computing resources according to their availability, which changes dynamically during the simulation. In addition, the application should be able to adjust computing power, i.e., add or subtract computing resources, according to its computational needs.

Fault tolerance: The application should detect not only explicit faults such as systems of clusters getting down and networks disconnected, but also implicit faults such as a job stuck in a queue for a long time. Furthermore, the application should recover from these faults automatically.

Scalability: The application should be able to manage a large number of computing resources efficiently.

3.2 Combined GridRPC + MPI approach

Although several Grid programming models have been proposed previously, it is not trivial for these models to satisfy all the three requirements at the same time. For example, Grid-enabled MPI does not provide mechanisms for flexibility and fault tolerance. Due to the lack of these mechanisms, applications using Grid-enabled MPI suffer from troubles such as co-allocation and system down.

In order to implement a flexible, fault tolerant and scalable Grid application, we propose a new programming approach, combining GridRPC with MPI. GridRPC [24] is a programming model based on a remote procedure call (RPC) mechanism tailored for the Grid. When viewed at a very high abstraction level, the programming model provided by GridRPC is that of standard RPC plus asynchronous, coarse-grained parallel tasking. At a more practical level, GridRPC provides a variety of features that hide the dynamic, insecure, and unstable aspects of the Grid from programmers. By providing simple, yet powerful, client-server-based frameworks for programming on the Grid, GridRPC has been used successfully in various Grid applications such as Monte Carlo simulations of cellular micro-physiology [25], short- to medium-term weather forecasting on an international Grid test bed [26], and molecular simulations using replica exchange Monte Carlo methods [27].

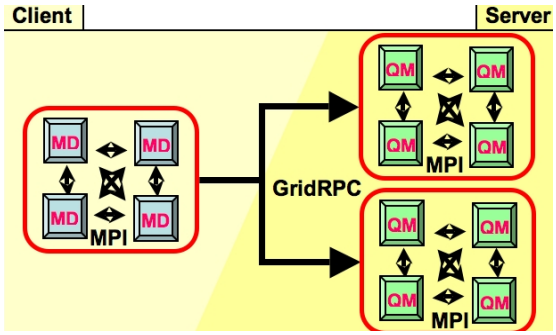

Figure 3: Program structure of the Grid-enabled multiscale MD/QM code based on the combined GridRPC + MPI approach. The simulation is performed by loosely coupling an MD client and QM servers using GridRPC. Each simulation is performed in parallel on a cluster using MPI. MD client also behaves as a GridRPC client to enable dynamic allocation, migration, and fault recovery of QM servers.

The proposed approach combines two complementary programming models, MPI and GridRPC, to satisfy the three requirements. First, GridRPC provides functions for dynamic execution of server programs, for detection of network/server errors, and for time outing to avoid unexpected waiting for a long time. These functions are used to satisfy the first two requirements, flexibility and fault tolerance. On the other hand, MPI is used to support efficient execution of a Grid application run in parallel on clusters.

Figure 3 shows a schematic of the system configuration, when applying the combined GridRPC + MPI approach to our multiscale MD/QM application. Each QM or MD simulation is allocated a cluster and is executed in a highly parallel manner using MPI. At the same time, MD simulation is tailored as a GridRPC client to allocate QM simulation dynamically and recover from faults of QM simulations. Such configuration allows us to: 1) change the target cluster and the number of processors; 2) reduce the possibility of critical faults; and 3) provide high performance of highly parallel MD/QM simulations.

3.3 Implementation of Grid-enabled hybrid MD/QM code

On the basis of the combined GridRPC + MPI approach, we have implemented the Grid-enabled multiscale MD/QM code using Ninf-G [28,29] and MPICH [30], which are widely used implementations of GridRPC and MPI, respectively.

In Grid-enabling the code, we have first extracted the code section to calculate QM forces in the clusters from the original multiscale MD/QM simulation code and have tailored it as a GridRPC server code. The code for

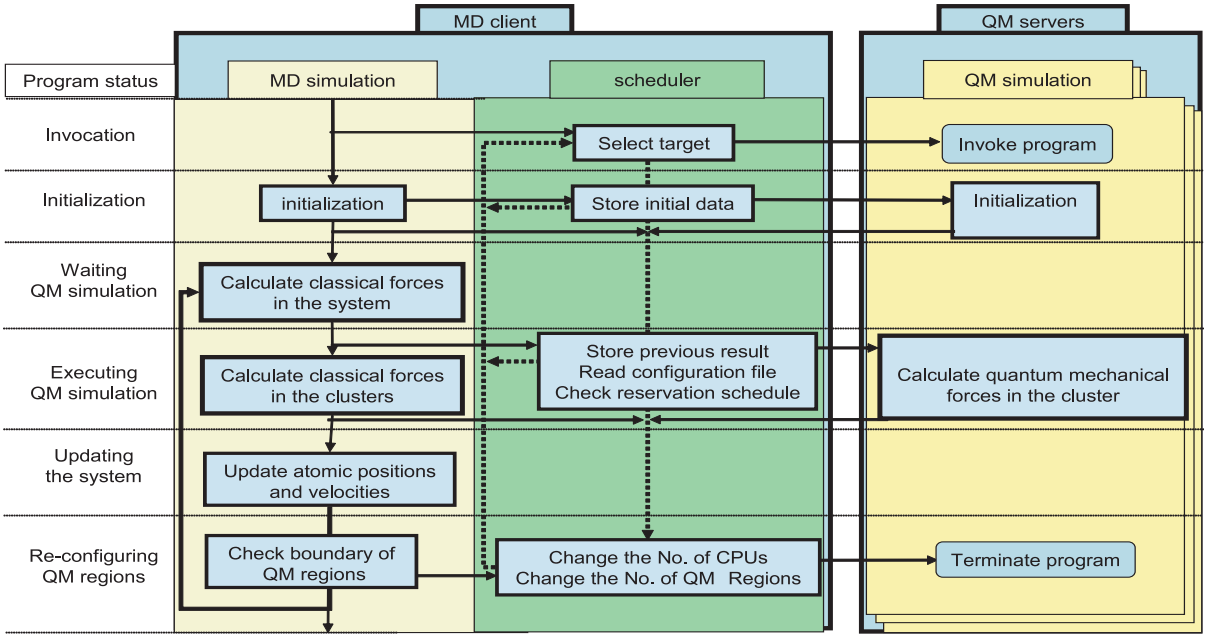

Figure 4: A flowchart of the Grid-enabled multiscale MD/QM simulation. Horizontal lines between MD client and QM servers denote GridRPC calls. A Dotted line shows a feed back loop of fault recovery, dynamic allocation, migration, and adjustment of computing power.

initialization, in which information such as the number of processors used and the rank of each process are set, is also included in the server code. In the next step, scheduling code has been added to the MD code to implement mechanisms for flexibility and fault tolerance as follows.

Flexibility

Mechanisms for flexibility are implemented using dynamic invocation functions of Ninf-G. They are triggered when QM simulation exceeds reservation time, or when QM simulation requires more processors than currently available due to the growth of the QM region. For the former case, the scheduling layer checks the reservation time in the configuration file at the beginning of QM simulation phase. For the latter case, MD client redefines QM regions, checks the resulted number of QM atoms, calculates the necessary number of processors, and notifies the change to the scheduler after updating the atomic positions and velocities.

When the above events take place, the scheduler rolls back the status to the invocation phase to select a target cluster, denoted as a dotted line in Fig. 4. Target machine is determined by considering both the computational cost of the QM simulation and the performance of clusters.

If the new target cluster is different from the previous one, the code is newly invoked and restarted on it. In order to continue simulation from the last time step, the scheduler keeps the initialization data and the snapshot data. In case of migration, the self-consistent field (SCF) calculation is initialized randomly so as to minimize the size of snapshot data, which is transferred between the MD client and the QM server at every time step. As a result, the amount of snapshot data is reduced to few MB (otherwise about 1GB data would have been transferred for the complete snapshot).

Fault tolerance

We implement mechanisms for fault detection and fault recovery. The fault detection mechanism is implemented using two kinds of Ninf-G functions. The first function detects explicit errors such as unexpected termination of server processes and network troubles through network disconnection between the client and servers. Another error-detection function is based on time outing. Ninf-G provides three kinds of timeout mechanisms: invocation timeout, execution timeout, and heartbeat timeout. The first mechanism detects timeout for server process invocation. If a remote program is not activated within a time specified by the user, Ninf-G returns an error to the client program. Execution timeout detects an excessive execution time. The last mechanism is used to detect the degradation of network performance. When the heart beat timeout is set, a server program sends keep-alive messages to the client periodically. If the client does not receive the message over a specified period, Ninf-G returns an error. All timeout mechanisms can be used by setting attributes such as job_MaxWallTime in the configuration file. We set these attributes and implement error check routines for all the Ninf-G functions in the scheduling code.

When an error is detected, the scheduler starts fault recovering. As in the migration of QM simulation, the scheduler selects the target cluster and restarts the QM simulation in the roll back loop. Machine selection algorithm is, however, different from that used for migration. The scheduler first tries to re-allocate the code

Table 1: Computing resources used for QM simulations in the Grid experiment.

	Cluster	Site	Max No. of CPUs used
1	P32	AIST	128×6
2	M64	AIST	128×2
3	F32	AIST	128×2
4	NCSA	NCSA	128
5	TCS	PSC	128×2
6	USC	USC	128×4
7	ISTBS	U-Tokyo	128
8	Presto	TITECH	128×2

on the same cluster, because allocating the code on a different machine would use a random initial guess, which results in a larger computational cost for the QM simulation. If re-allocation fails, the scheduler tries to switch the cluster as in the migration case.

4 Grid test-bed experiments

4.1 Target simulation

The target simulation is SIMOX semiconductor processing, in which five oxygen atoms with energy 240 eV is impacted on different crystalline positions of a silicon substrate consisting of 110,000 atoms. Initially, five QM regions are defined, four of which contain one oxygen atom and 12 silicon atoms, and the last region contains one oxygen atom and 14 silicon atoms. As oxygen atoms penetrate deeper into the substrate, the number of silicon atoms, which interact with oxygen atoms, increases. In order to maintain model accuracy, each QM region size should grow as the simulation proceeds.

4.2 Grid test bed

The SIMOX simulation has been performed on a Grid consisting of six supercomputer centers in Japan and the US: National Institute for Advanced Industrial Science and Technology (AIST), University of Tokyo (U-Tokyo), Tokyo Institute of Technology (TITECH), National Center for Supercomputing Applications (NCSA), Pittsburgh Supercomputing Center (PSC), and the University of Southern California (USC). Table 1 lists the computing resources used in our experiment. These clusters were used to run QM servers, while one small cluster was dedicated for a MD client.

Before starting the experiment, we configured these clusters as 20 virtual clusters, each of which has 128 CPUs, and reserved them manually according to the schedule shown in Table 2. We prepared 5 virtual clusters for QM simulations and saved 1 virtual cluster for possible trouble or the anticipated increase of the number of QM regions. The schedule is divided into 5 phases. In the first 10 days, we allocated all QM simulations on P32 subsystem of the AIST Super Cluster (ASC). Then, M64 subsystem of ASC was added for the simulation in the next 2 days and also added 2 TeraGrid clusters at NCSA and PSC in the third phase. In the fourth phase, we mainly used a cluster at USC. Finally, we used 4 Japanese clusters—P32, M64, Presto, and ISTBS—for 2 days. At the end of each phase, QM simulations were migrated to the next target clusters.

4.3 Results and discussions

The number of simulation time steps performed in the experiment was 270, which corresponds to a simulated time of 54 femtoseconds. The total CPU time amounted to 150,000 processor-hours. It is extremely difficult to execute a program on a single computing resource exclusively for such a long duration. This result suggests that our reservation-based sustainable Grid supercomputing framework provides a viable way to execute large-scale long-run applications.

In the following, we evaluate the results from three points of view, flexibility, fault tolerance, and scalability.

Flexibility

Figure 5 shows the time chart of the experiment in the last 10 days. The blue line denotes the execution of the QM simulation, red, light green, and light blue lines show the failure in the invocation, initialization, and simulation, respectively. As shown in the figure, our application automatically changed the target cluster according to the reservation schedule. It should be noted that the number of used clusters sometimes increased from 5 to 6, e.g. the simulation from the 12th day to the 13th day. This was because one QM region was divided into two due to the increased computational requirement.

Figure 6 shows the time evolution of the number of QM atoms and the number of CPUs used for the simulation,

Table 2: Reservation schedule of the clusters for the Grid experiment

	Phase 0	Phase 1		Phase 2			Phase 3			Phase 4	
	1-9 days	10 day	11 day	12 day	13 day	14 day	15 day	16 day	17 day	18 day	19 day
Cluster 1	P32-1	P32-1	P32-1	P32-1	P32-1	P32-1	USC-1	USC-1	USC-1	ISTBS	ISTBS
Cluster 2	P32-2	P32-2	P32-2	NCSA	NCSA	NCSA	USC-2	USC-2	USC-2	Presto	Presto
Cluster 3	P32-3	M64-1	M64-1	M64-1	M64-1	M64-1	M64-1	M64-1	M64-1	M64-1	M64-1
Cluster 4	P32-4	M64-2	M64-2	TCS-1	TCS-1	TCS-1	USC-3	USC-3	USC-3	P32-1	P32-1
Cluster 5	P32-5	P32-4	P32-4	TCS-2	TCS-2	TCS-2	USC-4	USC-4	USC-4	P32-2	P32-2
Spare	P32-6	P32-3	P32-3	P32-2	P32-2	P32-2	P32-1	P32-1	P32-1	F32-1	F32-1

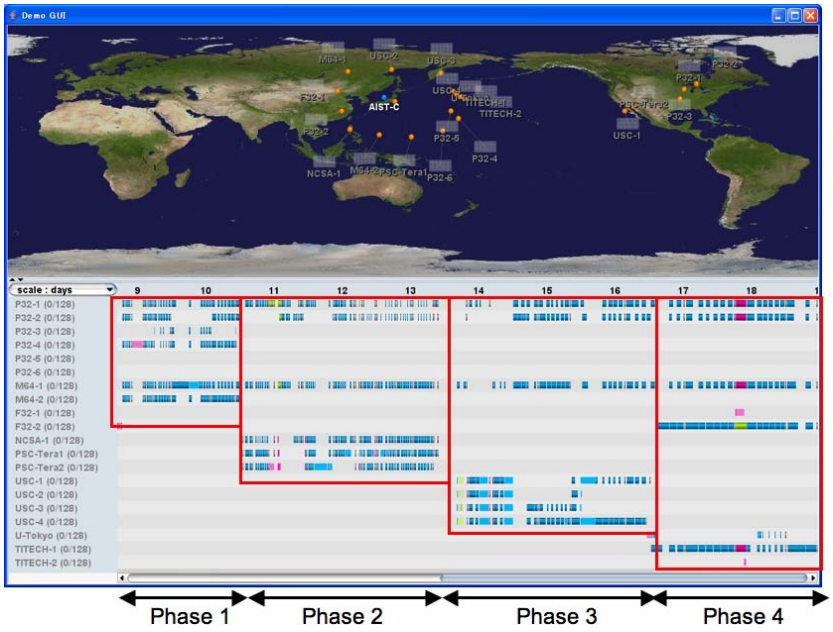

Figure 5: Time chart (bottom) of the multiscale MD/QM simulation of SIMOX, performed on geographically distributed parallel supercomputers in Japan and the US. The blue line denotes the execution of the QM simulation, the red line shows the failure in the initialization phase, and the light blue line the failure in the simulation phase.

where the number of QM atoms is shown as a red line and the number of CPUs as a blue line. Initially, the number of QM atoms was only 62. As the simulation proceeded, however, the number gradually increased to 341. The resulting increase in the computational requirement led to the automatic adjustment of the number of CPUs from 10 up to 702 and the migration of QM simulations 244 times. The abrupt increases of the number of CPUs in Fig. 6 are due to the increase of QM regions from 5 to 6, which are also the results of automatic adjustment. Figures 5 and 6 clearly show that the application is flexible enough to change the target cluster and adjust the number of CPUs automatically.

Fault tolerance

We observed faults in total of 770 times during the experiment, most of which took place in the invocation phase. Causes of the failure included: job manager of Globus was down; MPI process failed to get memories for inter-process communication; batch queue was not activated; and disk usage exceeded the quota limit. Although our application tried to reallocate the QM simulation on the same cluster, the possibility of success in reallocation was very low—less than 5%. It was because most causes of troubles described above were not resolved until we investigated the troubles and manually resolved them. In such cases, the code switched the target to the spare cluster.

Figure 7 shows the typical behavior of the code in trouble. A vertical red line shows the end of a simulation step. Although the code tried to change target clusters for two QM simulations due to the expiration of reservation time, one of them failed in the invocation phase. After

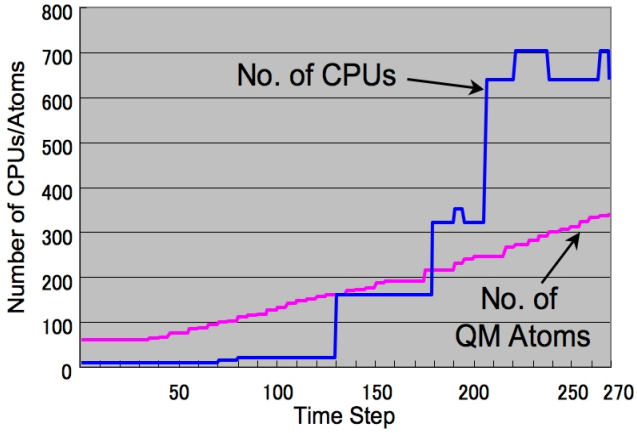

Figure 6: Time evolution of the number of QM atoms and the number of processors used for QM simulations. Blue and magenta lines show the number of CPUs and the number of QM atoms, respectively.

trying to re-allocate the simulation on the same cluster, the code decided to migrate the simulation again to the spare cluster, which fortunately succeeded in invoking the MPI processes.

In the worst cases, the code allocated multiple QM simulations on the same virtual cluster, because the number of available clusters became less than the number of QM regions due to troubles. In such cases, we sometimes stopped and restarted the simulation manually, because serial execution of QM simulations severely degrades the performance. In spite of these accidents, our code continued running over 5 days without intervention in the best case, which corresponds to phase 3 and phase 4 in Fig. 5.

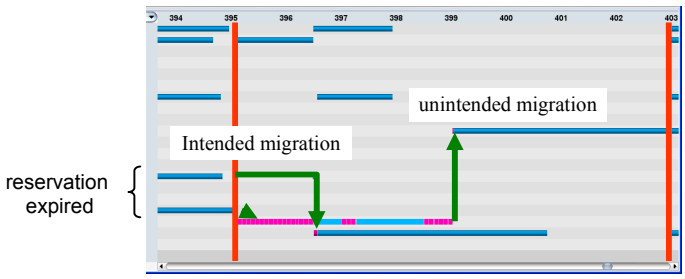

Figure 7: Diagram showing intended and unintended migrations. A vertical red line is the end of a time step.

During the simulation, time outing mechanisms of Ninf-G could detect 110 failures, which amounted to about 15% of the total failures. For example, jobs stuck in a batch queue and exceeding the disk quota limit were detected by the time outing mechanism. This confirms that the detection mechanism for implicit errors is as important as that for explicit errors for sustainable Grid supercomputing.

Scalability

It should be pointed out that managing hundreds of processes efficiently is difficult using only GridRPC, because a single GridRPC client must send data to and retrieve results from many GridRPC servers [26]. Our approach manages hundreds of processes by organizing them into several MPI process groups, which resulted in the management cost less than few percent of the total cost in our experiment. This result suggests that our approach is scalable even if the number of processes increases to thousands, when scalable parallel algorithms are assigned to GridRPC servers.

The execution efficiency of total simulation was about 60%. The main reason of the performance degradation is the load imbalance among QM simulations. Since all the QM simulations need to be synchronized at each time step, most QM simulations have to wait for the completion of the last QM simulation. There are two causes of the load imbalance: the difference in the number of atoms among QM regions and the difference in the performance of target clusters. Since the computational cost of the QM simulation scales as $O(N^3)$, where N denotes the number of QM atoms, different number of QM atoms results in the different simulation time. Even if all QM simulations have the same number of QM atoms, different performance of clusters leads to the different simulation time. In order to alleviate the load imbalance, the scheduler allocated the QM simulation with the larger number of QM atoms on the faster cluster. However, more sophisticated method will be required to fully balance the load.

The costs of fault detection and fault recovery were another source of performance degradation. For example, we observed that the cost of fault detection in the QM simulation phase became non-negligible as the simulation proceeded, because we had to set the value of the execution timeout larger than the simulation time for one time step. Callback function of Ninf-G, which provides a way to send the event back to the GridRPC client, may solve this problem.

4.4 Lessons learned

We gained the following insights through the experiment.

Necessity of systematic cross-site reservation system: In this experiment, fault detection, fault recovery and migration based on a reservation schedule were automated, but the cross-site reservations were performed manually before the experiment was started. In some cases, the manual reservation caused troubles. For example, our submitted jobs were not activated (stuck in a queue) even after the reservation time started. This is not a technical problem, rather a human error. More easy/systematic operation for cross-site run is desirable so as to avoid mis-configuration of the system.

Support for dynamic scheduling: When our application detects faults, it tries to re-allocate the simulation on the same cluster. But the re-allocation often failed, because some causes of the failure, such as batch system was not activated and disc system was full, were not recovered at the re-allocation. We also encountered troubles in MPICH, which quitted the program without releasing resources for IPC and resulted in the memory fault at the re-allocation. Although we reserved a spare cluster for such situations, it is generally difficult for users to reserve spare clusters. Finding and reserving idle clusters dynamically may alleviate the difficulty. There are some on-going researches on providing scheduling systems as a Grid service [31], [32]. Integrating these systems with our application to realize dynamic and fully-automated scheduling will be our future work.

Support for coping with heterogeneity: Although Grid middleware hides heterogeneity in the Grid such as hardware or operating systems, we encountered troubles caused by heterogeneity lying in the system configuration. For example, PSC cluster requires the use of Application Gateway for enabling communication between backend nodes and the Internet. Also, max wall clock time for batch jobs, disk quota limit, firewall, etc., are different from site to site. Although we were able to overcome these differences by adapting Ninf-G, it would not be so easy to do the same for application users. We need to accumulate

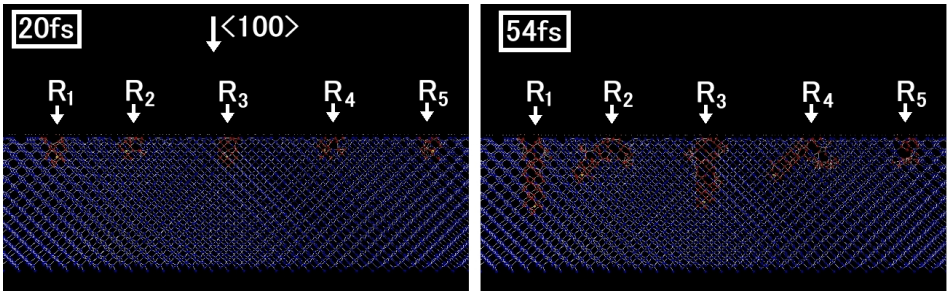

Figure 8: Snapshots of the multiscale MD/QM simulations of SIMOX at time 20fs (left) and 54fs (right), where red and blue spheres are quantum and classical Si atoms, respectively and yellow spheres are oxygen atoms. Five oxygen atoms are injected at the kinetic energy of 240eV perpendicular to Si(100) surface, where all the surface dangling bonds are saturated by hydrogen atoms. Five QM regions R_1-R_5 (indicated with arrows) are assigned, corresponding to the five oxygen atoms. Each QM region expands and changes its shape following the migration of the oxygen atom and the breakage and reconstruction of Si-Si bonds.

and share information on such troubles among not only Grid middleware developers but also application users, and feed them back to the research on innovative technologies such as virtual machine.

Necessity of general API for reservation-based sustained Grid supercomputing: A hybrid GridRPC + MPI approach is not specific to the multiscale MD/QM simulation but general for large-scale long-run applications which loosely couple several numbers of fine-grained parallel programs to calculate the result. For example, parameter survey applications such as genetic algorithm-based optimization [33] as well as multi-disciplinary simulation such as fluid-structure coupled simulation [34] will be good candidates. Both examples execute parallel simulation programs on clusters that constitute the Grid. These simulations, implemented based on our approach, can be flexible, robust and scalable. Simulation programs based on divide-and-concur algorithm with varying computing load such as shock wave simulation [35] based on the adaptive mesh refinement method [36] will be another candidate for our approach. If the simulation program is allocated dynamically using GridRPC from a client that controls the execution of the simulation, the application can be tailored to automatically increase/decrease the number of CPUs according to the computing load. In order to facilitate the construction of such large-scale long-run Grid applications in various scientific domains, it is important to design a general API for reservation-based sustained Grid supercomputing. Such an API enables application programmers to Grid-enable their applications without worrying about low-level logics of resource allocation, migration, fault detection and fault recovery. One of our future works is to provide Grid services for sustainable adaptive Grid supercomputing via a general API which is implemented by integrating various Grid middleware such as Ninf-G and a Grid scheduling system.

4.5 Application results

Present multiscale MD/QM simulation provides useful information to improve the SIMOX technique. We have set the same incident velocity to all the five oxygen atoms but found that the migration of oxygen is highly sensitive to the incident position. In two cases at regions R_1 and R_3 in Fig. 8, the oxygen atoms are rather ballistic and lose only 5-15% of the incident velocity at 54fs, whereas in other three cases (R_2, R_4, and R_5), the oxygen atoms change their direction of motion toward ⟨111⟩ and lose 30-60% of the incident velocity at 54fs, accompanied by recoil motion of a few Si atoms hit by the oxygen atoms. These results indicate that the eventual penetration depths will differ substantially in the five cases. Such high sensitivity of oxygen penetration depth should be taken into consideration in extending the SIMOX technique to lower incident energies.

5 Conclusions

We have demonstrated a large adaptive multiscale MD/QM simulation of considerable scale (involving over 1,000 processors) and duration (over tens of continuous days) on a Grid of geographically distributed parallel supercomputers in Japan and the US, in which the number of processors change dynamically on demand and resources are allocated and migrated dynamically according to both reservations and unexpected faults. The proposed reservation-based sustainable adaptive Grid supercomputing paradigm has been proven to be a promising approach towards sustained Grid supercomputing — when combined with enabling algorithmic and systems techniques: scalable algorithms based on the embedded divide-and-conquer framework; adaptive MD/QM boundary management to maintain the model accuracy based on the buffered-cluster hybridization scheme; and the combined GridRPC + MPI Grid application framework to achieve flexibility (adaptive resource allocation and

migration), fault tolerance (automated fault recovery), and scalability. Such ultrascale atomistic simulations are expected play an important role in the design of future high speed and low power-consumption semiconductor devices, as the present simulation results on SIMOX (separation by implanted oxygen) processing of semiconductor devices indicate.

Acknowledgements

We are grateful to TeraGrid executive committee members for providing their computing resources for the Grid experiment. We would also like to thank administrators at TITECH, U-Tokyo, USC, and AIST for setting up their clusters for us to use exclusively over the reservation period. The work at USC was partially supported by AFOSR-DURINT, ARO-MURI, Chevron-CiSoft, DOE, and NSF. The work in Japan was supported by JST-CREST.

References

[1] FOSTER, I., AND KESSELMAN, C. 2003. *The Grid2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, San Francisco.

[2] ALLEN, G., DRAMLITSH, T., FOSTER, I., KARONIS, N. T., RIPEANU, M., SEIDEL, E., AND TOONEN, B. 2001. Supporting efficient execution in heterogeneous distributed computing environments with Cactus and Globus. In *Proceedings of SC2001*, ACM, New York.

[3] EMMOTT, S., AND RISON, S. 2006. *Towards 2020 Science*. Microsoft, Redmond.

[4] FOSTER, I. 2006. A two-way street to science's future. *Nature*, 440:419.

[5] SHIRTS, M., AND PANDE, V. S. 2000. Screen savers of the world unite. *Science*, 290:1903.

[6] IKEGAMI, T., ISHIDA, T., FEDOROV, D. G., KITAURA, K., INADOMI, Y., UMEDA, H., YOKOKAWA, M., AND SEKIGUCHI, S. 2005 Full electron calculation beyond 20,000 atoms: ground electronic state of photosynthetic proteins. In *Proceedings of SC05*, ACM, New York.

[7] SHIMOJO, F., KALIA, R. K., NAKANO, A., AND VASHISHTA, P. 2005. Embedded divide-and-conquer algorithm on hierarchical real-space grids: parallel molecular dynamics simulation based on linear-scaling density functional theory. *Computer Physics Communications*, 167:151-164.

[8] BROUGHTON, J. Q., ABRAHAM, F. F., BERNSTEIN, N., AND KAXIRAS, E. 1999. Concurrent coupling of length scales: methodology and application. *Physical Review B*, 60:2391-2403.

[9] OGATA, S., LIDORIKIS, E., SHIMOJO, F., NAKANO, A., VASHISHTA, P., AND KALIA, R. K. 2001. Hybrid finite-element/molecular-dynamics/electronic-density-functional approach to materials simulations on parallel computers. *Computer Physics Communications*, 138:143-154.

[10] KIKUCHI, H., KALIA, R. K., NAKANO, A., VASHISHTA, P., IYETOMI, H., OGATA, S., KOUNO, T., SHIMOJO, F., TSURUTA, K., AND SAINI, S. 2002. Collaborative simulation Grid: multiscale quantum-mechanical/classical atomistic simulations on distributed PC clusters in the US and Japan. In *Proceedings of SC2002*, IEEE, Los Alamitos.

[11] KARONIS, N., TOONEN, B., AND FOSTER, I. 2003. MPICH-G2: a Grid-enabled implementation of the message passing interface. *Journal of Parallel and Distributed Computing*, 63:551-563.

[12] VASHISHTA, P., KALIA, R. K., AND NAKANO, A. 2006. Multimillion atom simulations of dynamics of oxidation of an aluminum nanoparticle and nanoindentation on ceramics. *Journal of Physical Chemistry B*, 110:3727-3733.

[13] HOHENBERG, P., AND KOHN, W. 1964. Inhomogeneous electron gas. *Physical Review*, 136:B864-B871.

[14] KOHN, W., AND VASHISHTA, P. 1983. General density functional theory. In *Inhomogeneous Electron Gas*, eds. N. H. March and S. Lundqvist, pages 79-184. Plenum, New York.

[15] NAKANO, A., KALIA, R. K., VASHISHTA, P., CAMPBELL, T. J. OGATA, S., SHIMOJO, F., AND SAIRI, S. 2002. Scalable atomistic simulation algorithms for materials research. *Scientific Programming* 10:263-271.

[16] FATTEBERT, J.-L., AND GYGI, F. 2004. Linear scaling first-principles molecular dynamics with controlled accuracy. *Computer Physics Communications*, 162:24-36.

[17] CHELIKOWSKY, J. R., SAAD, Y., ÖGUT, S., VASHILIEV, I., AND STATHOPOULOS, A. 2000. Electronic structure methods for predicting the properties of materials: Grids in space. *Phyica Status Solidi (b)*, 217:173-195.

[18] DAPPRICH, S., KOMAROMI, I., BYUN, K. S., MOROKUMA, K., AND FRISCH, M. J. 1999. A new ONIOM implementation in Gaussian 98. I. The calculation of energies, gradients, vibrational frequencies, and electric field derivatives. *Journal of Molecular Structure (Theochem)* 461-462:1-21.

[19] OGATA, S., SHIMOJO, F., KALIA, R. K., NAKANO, A., AND VASHISHTA, P. 2002. Hybrid quantum mechanical/molecular dynamics simulations for parallel computers: density functional theory on real-space multigrids. *Computer Physics Communications*, 149:30-38.

[20] BELKADA, R., AND OGATA, S. 2003. Theoretical study of stress corrosion cracking in Si. *Transactions of the Materials Research Society of Japan*, 28:817-820.

[21] OGATA, S., SHIMOJO, F., NAKANO, A., VASHISHTA, P., AND KALIA, R. K. 2004. Environment effects of H_2O on fracture initiation in Si: a hybrid electronic-density-functional/molecular-dynamics study. *Journal of Applied Physics*, 95:5316-5323.

[22] OGATA, S., AND BELKADA, R. 2004. A hybrid electronic-density-functional/molecular-dynamics simulation scheme for multiscale simulation of materials on parallel computers: application to silicon and alumina. *Computational Material Science*, 30:189-194.

[23] OGATA, S. 2005. Buffered-cluster method for hybridization of density-functional theory and classical molecular dynamics: application to stress-dependent reaction of H_2O on nanostructured Si. *Physical Review B*, 72:045348.

[24] SEYMOUR, K., NAKADA, H., MATSIPKA, S., DONGARRA, J., LEE, C., AND CASANOVA, H. 2002. Overview of GridRPC: a remote procedure call API for Grid computing. In *Proceedings of the 3rd International Workshop on Grid Computing*, pp. 274-278.

[25] CASANOVA, H., BARTOL, T., STILES, J., AND BERMAN, F. 2001. Distributing Mcell simulations on the Grid. *Journal of Supercomputing Applications*, 15:243-257.

[26] TAKEMIYA, H., SHUDO, K., TANAKA, Y., AND SEKIGUCHI, S. 2003. Constructing Grid applications using standard Grid middleware. *Grid Computing*, 1:117-131.

[27] IKEGAMI, T., TAKEMIYA, H., NAGASHIMA, U., TANAKA, Y., AND SEKIGUCHI, S. 2003. Accurate molecular simulation on the Grid—replica exchange Monte Carlo simulation for C_{20} molecule. *IPSJ Transaction of Advanced Computing Systems*, 44(SIG11):14-22.

[28] TANAKA, Y., NAKADA, H., SEKIGUCHI, S., SUZUMURA, T., AND MATSUOKA, S. 2003. Ninf-G: a reference implementation of RPC-based programming middleware for Grid computing. *Journal of Grid Computing*, 1:41-51.

[29] TANAKA, Y., TAKEMIYA, H., NAKADA, H., AND SEKIGUCHI, S. 2004. Design, implementation and performance evaluation of GridRPC programming middleware for a large-scale computational Grid. In *Proceedings of the 5th International Workshop on Grid Computing*, pp.298-305.

[30] GROPP, W., LUSK, E., DOSS, N., AND SKJELLUM, A. 1996. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22:789-828.

[31] http://hpcsoftware.teragrid.org/Software/user/show_asset.php?asset_id=189&view=TeraGrid

[32] http://www.g-lambda.net

[33] MIRGHANI, B. Y., TRYBY, M.E., BAESSLER, D. A., KARONIS, N., RAHJITHAN, R., AND MAHINTHAKUMAR, K. G. 2005. Development and Performance Analysis of a Simulation-Optimization Framework on TeraGrid Linux Clusters. In Proceedings of the 6th LCI International Conference on Linux Clusters: The HPC Revolution 2005.

[34] KIMURA, T., AND TAKEMIYA, H. 1999. Distributed Parallel Computing for Fluid-Structure Coupled Simulations on a Heterogeneous Parallel Computer Cluster, International Journal of High Performance Computing Applications, Vol. 13, No. 4, pp.320-333.

[35] BRANICIO, P. S., KALIA, R. K., NAKANO, A., AND VASHISHTA, P. 2006. Shock-induced Structural Transition, Plasticity, and Brittle Cracks in Aluminum Nitride Ceramic: A Molecular Dynamics Study, Physical Review Letters, Vol. 96, pp. 065502: 1-4.

[36] BERGER, M. J., AND OLIGER, J. 1984. Adaptive Mesh Refinement for Hyperbolic Partial Differential Equation, Journal of Computational Physics, Vol. 53, pp. 484-512.