The Grid Enablement and Sustainable Simulation of Multiscale Physics Applications

Yingwen Song[†,††††],Yoshio Tanaka[†,††††],Hiroshi Takemiya[†,††††], Aiichiro Nakano[††], Shuji Ogata[†††,††††], Satoshi Sekiguchi[†]

[†]*National Institute of Advanced Industrial Science and Technology*
Tsukuba, Ibaraki 305-8568, Japan
[††]*University of Southern California*
3651 Watt Way, VHE610 Los Angeles, CA 90089-0242, USA
[†††]*Nagoya Institute of Technology*
Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan
[††††]*CREST, Japan Science and Technology Agency*
{yw-song,yoshio.tanaka, h-takemiya}@aist.go.jp,
anakano@usc.edu, ogata@nitech.ac.jp, s.sekiguchi@aist.go.jp

Abstract

The understanding of H diffusion in materials is pivotal to designing suitable processes. Though a nudged elastic band (NEB)+molecular dynamics (MD)/quantum mechanics (QM) algorithm has been developed to simulate H diffusion in materials by our group, it is often not computationally feasible for large-scale models on a conventional single system. We thus gridify the NEB+MD/QM algorithm on the top of an integrated framework developed by our group. A two days simulation on H diffusion in alumina has been successfully carried out over a Trans-Pacific Grid infrastructure consisting of supercomputers provided by TeraGrid and AIST. In this paper, we describe the NEB+MD/QM algorithm, briefly introduce the framework middleware, present the grid enablement work, and report the techniques to achieve fault-tolerance and load-balance for sustainable simulation. We believe our experience is of benefit to both middleware developers and application users.

1 Introudction

Alumina exhibits a remarkable series of metastable polymorphs other than the most stable alpha-alumina. Alumina in particular gamma-alumina has been used in a variety of industrial applications, where H-diffusion in nanostructured alumina plays important roles. Gamma-alumina is believed to have a spinel structure. The spinel structure possesses a face-centered cubic sublattice of O atoms. The Al_2O_3 stoichiometry necessitates vacant cation positions,locations of which has been a subject of intensive research. The H atoms are expected to diffuse through such vacant positions. Johnsson et al. proposed the nudged elastic band (NEB) method [1] to calculate such a diffusion path in an inhomogeneous material. The NEB method is an efficient method for finding the minimum energy path (MEP) between two given states (i.e., before and after the reaction). Relating to the long MEP, a large number of atoms need to be involved in the NEB calculations. However, since accurate evaluation of the energy profile through MEP usually requires an electronic-structure calculation method such as the density-functional theory (DFT), treating all the involved atoms with such a compute-intensive method is impractical. Our group then has combined the NEB method with a hybrid MD/QM method recently (called NEB+MD/QM method below)[2, 3, 4]. In this method, accurate but computation-intensive quantum mechanical (QM) simulations are embedded within a classical molecular dynamics (MD) simulation only when and where high fidelity is required.

Like other similar methods, the NEB+MD/QM method must take account of a large number of quantum states in the QM computations. It is of course computationally intensive. Though we have parallelized the program and achieved impressive performance on clusters, it cannot yet give results in reasonable time for large scale and complex models in our simulations. We thus decided to harness the power of large numbers of heterogeneous, distributed CPUs provided by a grid infrastructure. The NEB+MD/QM algorithm then has been gridified based on an integrated framework developed by our group.

The motivation of this work is to realize large-scale and long-time simulations which are difficult with convential systems. The *overall contributions* of this paper are as follows.

- We solved our physics problems and got reliable results which are difficult to obtain with conventional HPC methods.
- We achieved fault-tolerance and dynamic load-balance during the long-time simulation through an integrated framework developed by our group[5]. These two issues are critical to the success of computing Grid. As a by-product, the usability of the framework is proved again with this experimental study.
- Experience and lessons learnt in using large-scale computing Grid are reported to share with application community.
- The large-scale and long-time simulations of this study demonstrates that computing Grid can practically bring enormous potential to E-science.

The rest of the paper is organized as follows. After examining the related work briefly in section 2, we introduce the sequential and parallel NEB+MD/QM method in section 3, discuss the grid enablement work in section 4, and report the experimental details in section 5. The conclusions are given in the last section.

2 Related Work

Grid enablement of existing applications has been a prime area of research in recent times and numerous solutions have been proposed. The most simple one is to run MPI-based applications in an MPI-aware grid environemnt. Such typical MPI middleware includes MPICH-G2[6] and GridMPI[7]. At present, most reported studies[8] are actually based on this method. But this approach is not fault tolerable, and is difficult to obtain resources from multiple sites in the same time frame for large scale computations.

In order to overcome the above problems, the GridRPC[9] standard has been defined by OGF and there are currently two reference implementations available: Ninf-G[10] and NetSolve[11]. Our group has also proposed a hybrid MPI+GridRPC programming model to combine the advantages of both MPI and GridRPC. Using this method, we performed some large-scale experiments over TeraGrid, but most of the work such as resource reservation and error recovering was done manually[12]. We thus have developed an integrated framework[5] to provide users with APIs to develop fault-tolerant, resource-aware, and self-load-balancing applications. This framework is not a workflow, and is fundamentally different from other approaches such as Triana, Kepler, Taverna, Petri nets, Skyflow reviewed by Fox et al.[13].

Though there have been some grid-enabled applications reported in computational bio-informatics[14], computational fluid dynamics[15], and e-Science[16] etc., their executions were usually performed in a regional grid or campus test-bed, furthermore the execution time was not so long. Different from other applications in time and scale, this work enables a physics application to run in a cross-continent grid environemnt on top of our framework which makes it uniquely different from previous work[12]. In addition, this work focuses on the application perspective, while our another work[5] discusses the middleware perspective.

3 NEB+MD/QM Method

3.1 Sequential NEB+MD/QM Algorithm

In the NEB+MD/QM method, an elastic chain of images (or states in the phase space) of the whole system is generated between the two end-states, and all the intermediate image-slices are optimized simultaneously in a concerted way toward the minimum energy state which includes the potential energy of the elastic chain. The NEB method is used to evaluate the energy barrier of a reaction process through variation of the atomistic energy among the image-slices. However, a large number of atoms need to be involved in the NEB calculations, and the accurate evaluation of the energy profile through MEP usually requires computation-intensive methods such DFT (density-functional theory). In order to reduce computing efforts and to keep sufficient accuracy in calculating the interaction energy within an image-slice, our group has proposed an accurate and robust MD/QM coupling method by introducing the buffer atoms between the QM and MD regions. That is, the QM region selected in an image-slice is treated with DFT, while the rest of the system with the classical MD method. In the NEB+MD/QM method, the QM region changes adaptively between the image-slices to trace the reacting atoms. Figure 1 schematically shows the computation algorithm. Along the reaction path, a number of image-slices are introduced that form an elastic chain with virtual springs, which are relaxed with the NEB method toward MEP. In each image-slice, the QM region is coupled with the MD region.

3.2 Parallel NEB+MD/QM Method

One of the major advantages of the NEB+MD/QM Method is the ease of parallelization; this scheme has been parallelized with MPI since the image slices can

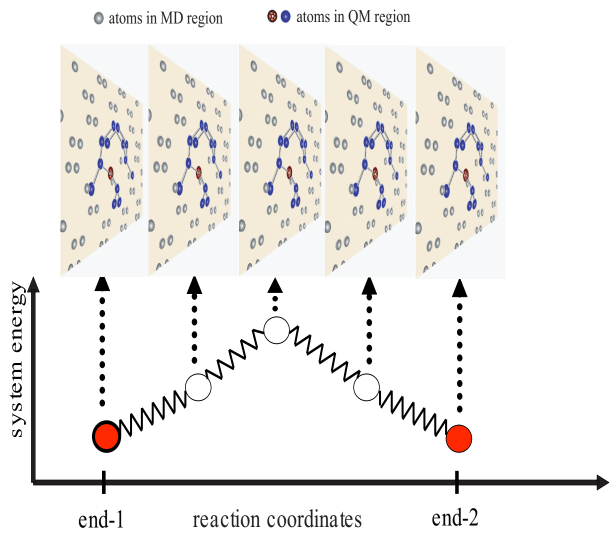

Figure 1. Schematic view of the NEB+MD/QM algorithm

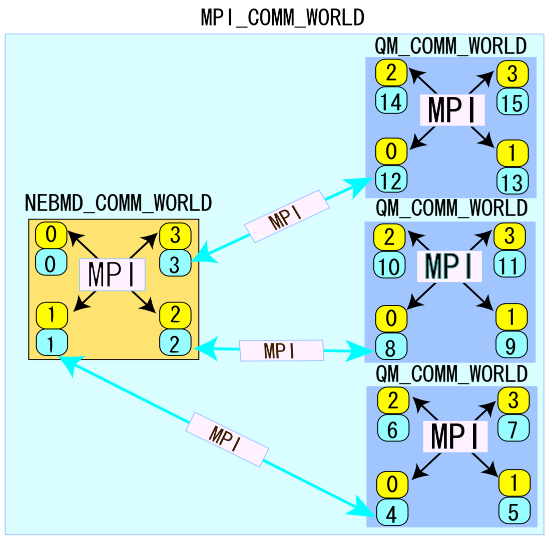

Figure 2. MPI based parallelization of the NEB+MD/QM method

be computed in parallel fashion. In order to take advantage of the computation hierarchy of this scheme and to meet the structure of this program, a two-tier parallelization strategy has been employed in parallelizing this program. In the upper level parallelization, the global communicator (MPI_COMM_WORLD) is split into some sub-communicators, i.e. one NEBMD_COMM_WORLD and many QM_COMM_WORLDs. One process in the NEBMD_COMM_WORLD is called NEB process, and the others are called MD processes. The NEB process sends data and control signals to, and receives results from the MD processes. It plays as a master role in the computation. The number of QM_COMM_WORLD is equal to the number of MD processes, since each QM_COMM_WORLD is remotely controlled by an MD process. Physically, each MD region is associated with an MD process, and each QM region is mapped to one QM_COMM_WORLD. The inter-subcommunicator communications and the intra-subcommunicator ones are schematically shown in Figure 2. In this figure, the numbers in the upper ovals are local ranks and the ones in the lower ovals represent ranks in the global communicator MPI_COMM_WORLD. Though this parallel program has demonstrated very good performance, the models it can handle is not large enough to meet our needs on a convential single-system supercomputer.

4 Grid Enablement of NEB+MD/QM Method

4.1 Middleware Supporting the Grid Enablement

The grid enablement of the NEB+MD/QM method is based on an integrated framework which supports sustainable execution of applications implemented with MPI+GridRPC programming model. It includes a resource allocator (GRPLib) and an application internal task scheduler (AITS). The resource allocator can provide applications with reliable computing resources, and the AITS is designed to ensure computation tasks on all sites to execute sustainably and cooperatively. It incorporates certain fault-tolerant ability by running failed GridRPC calls repeatedly and feeding back GRPLib with site reliability. The AITS is also capable of some degree of load-balancing by letting one task yield powerful computing resources to another one. Since we have already reported the details of the framework elsewhere[5], the focus here will be on the framework's interface and functionality.

4.1.1 Resource Allocator

The GRPLib is a three-tier SOA (service-oriented architecture) system, i.e., service layer, brokerage layer, and client layer. The three layers represent resource provisioning, resource allocation, and resource utilization respectively. The service layer lying at the bottom provides the functionalities. A main service available at present is Maui-based[17] reservation service which guarantees us the availability of computing resources. The middle layer can route reservation requests to the services on remote sites and automate the resource matching to meet a specific resource request. A LDAP database is used to store reservation and allocation data in this layer. The client layer provides functions to interact with the framework and is deployed as a library. These functions encapsulate calls to the client stubs of the middle layer service. At present, functions for resource reservation and allocation are available. A web portal written in PHP is also provided to help users manage reservations and allocations. The reservation functions are usually invoked through the web portal.

4.1.2 Ninf-G Extension

To make the reserved resource accessible from a Ninf-G based application. The Ninf-G implementation must be extended to support this feature. Consequently, we have added three attributes to simplify reservation. *GRPC_HANDLE_ATTR_RESVID* is used to pass reservation ID to the reservation-aware GRAM of a remote gatekeeper, *GRPC_HANDLE_ATTR_JOBMAXWALLTIME* attribute is used to specify time limit, while *GRPC_HANDLE_ATTR_JOBHOSTCOUNT* is used to specify node number. They can also be set with attributes *reservation_id*, *job_maxWallTime*, and *job_hostCount* respectively in the configuration file. These extended attributes enable us to invoke a remote server to run on the reserved resources. In addition, we can achieve greater flexibility by retrieving values for the extended attributes from the GRPLib since the server can be launched on the allocated site that we do not need to have prior information. This makes the grid behavior like a virtual machine from the client's point of view.

4.1.3 Application Internal Task Scheduler

The AITS is the most important component in the framework to ensure sustainable execution of large Scale and long time (LSLT) applications. It provides end users with two sets of API functions, and a program skeleton.

One API set is developed to schedule and manage computation tasks. The task sequence is stored in a structure called task-table which contains the task array, task number, and max stage number. The management of a single task and the task pool results in two categories of functions: those for a single task and those for the task-table. At any time, each task is in one of four states: *TASK_INIT*, *TASK_ACTIVE*, *TASK_SUSPEND*, and *TASK_TERMINATE*. They represents initialization, computation, suspension, and termination respectively. If a task is not permitted to begin a stage until its neighbors have finished the previous stage and made dependent data ready, it will be placed in suspension state and the occupied resources will be freed accordingly. The freed resources can then be used by another task.

The other set of functions called Soft-GRPC API are actually wrappers of some Ninf-G functions. They are enhanced with fault-tolerance feature by trying to call the lying Ninf-G functions *MaxRetries* times until the call is successful.These functions can also make the grid infrastructure behavior like a virtual machine by requesting on-demand resources from the GRPLib. These resources are usually reserved automatically with the GRPLib portal or manally in advance.

The resource allocator, the task management API, and the Soft-GRPC API are integrated by an AITS skeleton which implements the hybrid MPI+GridRPC programming models. With this integrated skeleton, grid-enabled applications can take advantage of the efficiency of MPI and the flexibility of Ninf-G. The robustness is achieved by repeatedly trying a failed call and utilizing reliability evaluated and availability guaranteed resources which are allocated by the GRPLib.

4.2 Grid Enablement of the NEB+MD/QM Method

Though we can run the parallelized program on an MPI-aware grid system, it has some problems as has been mentioned above. To overcome the fault-intolerable problems of MPI, we have gridified the program with GridRPC + MPI programming model. This model enables us to implement flexibility in the sense of adaptive resource allocation and task migration, fault tolerance in the sense of automated fault recovery, and efficiency in the sense of scalable management of large computing resources. This approach combines two complementary programming models, MPI and GridRPC, to satisfy the above three requirements. First, GridRPC provides functions for dynamic execution of server-side 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 requirements of flexibility and fault tolerance. On the other hand, MPI is used to support efficient execution of a parallel application on a single system.

Figure 3 shows a schematic of the system configuration, when applying the combined GridRPC + MPI approach to our hybrid NEB and multiscale MD/QM simulation. The simulation is performed through loosely coupled computations for different time instances. Each of the computation programs is a parallel MD/QM implemented with hybrid GridRPC + MPI. The simulation is controlled by a process called NEB master which is one of the processes in the client MPI pool. The other client processes are called MD processes and are actually GridRPC Clients. The NEB

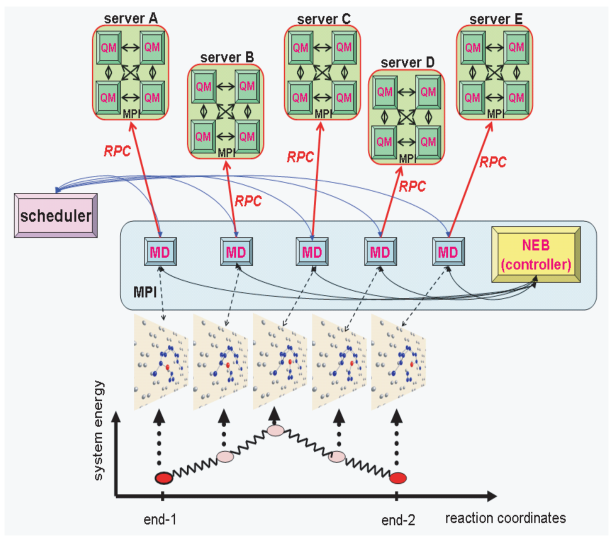

Figure 3. Sructure of the Grid-enabled NEB+MD/QM method

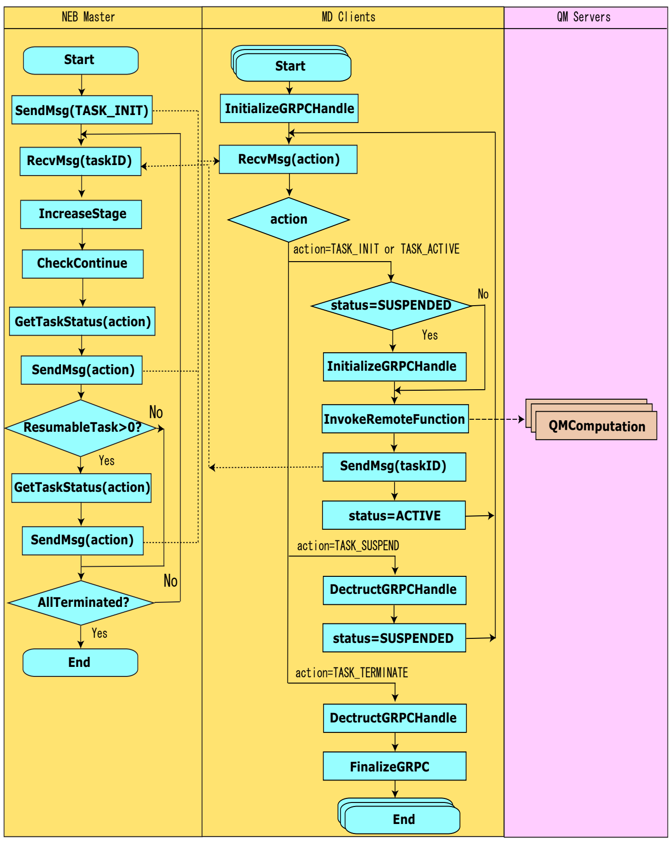

Figure 4. Implementation of the Grid-enabled NEB+MD/QM method

master monitors the computation progress and sends action signals to the MD processes accordingly. Once an MD process receives an instruction, it will take corresponding actions. If the instruction is *TASK_ACTIVE*, it will invoke remote QM computation. The server-side QM programs are parallelized with MPI. This configuration allows us to: (1) request resources dynamically; (2) reduce the possibility of critical faults; and (3) gain maximum performance from highly parallel MD/QM simulations.

The resource allocator API, the task management API, and the Soft-GRPC API are called to implement the AITS. Figure 4 shows the AITS structure for the grid enablement of the NEB+MD/QM algorithm. As can be seen, the server application is the QM module and the client application is composed of NEB and MD modules. Both the QM module and the client application are parallelized with MPI. In the client process pool, the NEB process monitors the statuses of all the tasks and repeatedly signals actions to the MD processes until all tasks have terminated, while the MD processes retrieve messages and take corresponding actions in a loop. If an MD process receives a *TASK_ACTIVE* action, it will invoke remote QM server and wait for the results. If the action received is *TASK_SUSPEND*, the MD process will free the allocated resource, set its status to *SUSPENDED*, and wait to be resumed. The *TASK_TERMINATE* action instructs the MD process to exit the iteration, free the allocated resource, and finalize the Ninf-G and GRPLib modules. With this integrated skeleton, our grid-enabled applications can take advantage of the efficiency of MPI and the flexibility of Ninf-G. The robustness is achieved by repeatedly trying a failed call and utilizing reliability evaluated and availability guaranteed resources which are allocated by the GRPLib.

5 LSLT Simulation of the NEB+MD/QM Program on TeraGrid

5.1 Testbed Configuration

The simulation of the gridified program was performed on a testbed consisting of clusters provided by both TeraGrid[18] and AIST. The detailed information of the reserved resources is shown in Table 1. The computing resources were guaranteed manually by allowing us to occupy special queues exclusively during the experiment. The available time slots of the resources were stored in the GRPLib database.

Table 1. Computing resources reserved for the experiment

No	Site/Cluster	Grid	CPU Number
1	F32	AIST	35 (NEB + MD)
2	F32	AIST	32 x 6 (192)
3	P32	AIST	32 x 8 (256)
4	NCSA	TeraGrid	32 x 8 (256)
5	SDSC	TeraGrid	32 x 4 (128)
6	Purdue	TeraGrid	32 x 8 (256)
Total			1129

5.2 Experiment

In the experiment, we prepared 34 image slices for different time instances, each of which had different number of QM atoms. The MPI-based NEB and MD computation for different image slices ran on AIST F32 cluster with 35 processes, and each QM server was configured to utilize 16 nodes (32 CPUs). In order to maximize the utilization of TeraGrid resources, we flagged the AIST clusters to *UNAVAILABLE* if all the TeraGrid sites were available. Otherwise, we switched the status of AIST clusters to *AVAILABLE*.

Figure 5 illustrates the status of the execution. The blue (dark grey) bars indicate that the site was used for QM simulations, the red (black) ones represent that the site was not available, while the yellow (light grey) ones signify that the site was not fully available due to some problems. The simulation was started with AIST P32 and F32 clusters. After 10 hours, TeraGrid resources became available, which was automatically detected by the AITS.Figure 6 shows the profile of computed QM number by each site.

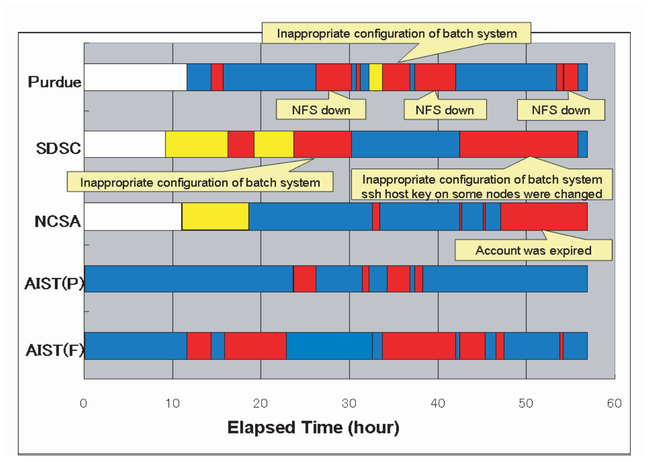

Figure 5. Status of the execution

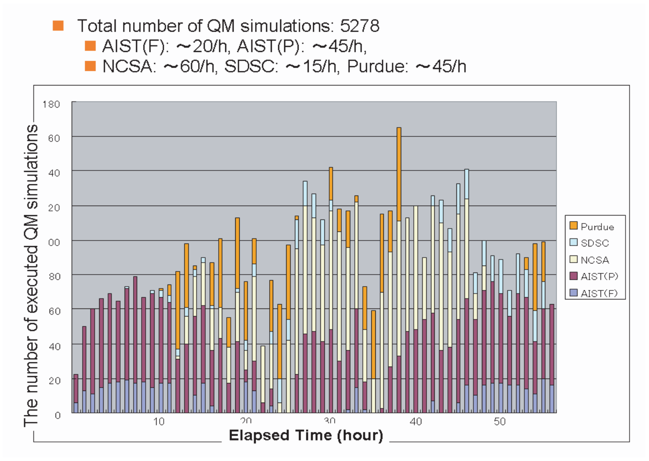

Figure 6. QM number calculated per hour in the experiment

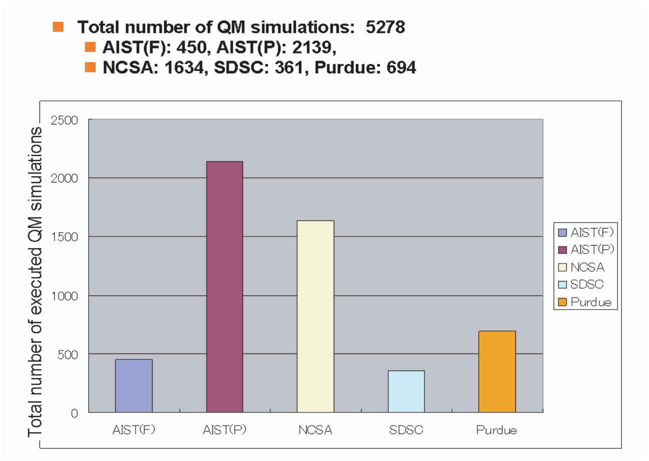

Figure 7. QM number calculated in the experiment

5.3 Results and Discussion

Figure 7 shows the total QM number computed by each site. We reached total 5278 QM simulations. Sites AIST(F), AIST(P), NCSA, SDSC, and Purdue contributed 450, 2139, 1634, 361, and 694 QM simulations respectively. AIST P32 cluster contributed the largest number since it could run 256 CPUs stably. On the other hand, the AIST F32 and SDSC did small ones. The performance of the AIST F32 cluster was relatively low. As for the SDSC site, we used only 128 CPUs and some of which were sometimes unavailable due to the problems mentioned above. Furthermore, the jobs waited a long time in the scheduled queue due to inappropriate configuration. As can be seen from Figure 7, the NCSA site gave the best performance, while AIST(P) and Purdue sites exhibited almost the same performances. However, because the NCSA site was unavailable during the last 12 hours, it contributed less number than the AIST P32 cluster did.

During the two days experiment, we encountered some problems such as *NFS down, system miss-configuration, and account expiration* etc. These problems prevented us from using some TeraGrid sites. However, most of the unavailable durations of AIST sites were made intentionally for increasing the utilization of TeraGrid resources.

This experiment also has demonstrated some interesting points: (1) we achieved total 5278 QM simulations which are very difficult to achieve on a conventional single-system cluster. Thus grid has been proved to have the potential power for ultra-scale computation. (2) The program was capable of fault-tolerance to some extent. Although we encountered some problems, the simulation continuously executed over 48 hours by detecting such faults and then recovering from the errors. (3) The program was able to adjust the load balance automatically. The total QM simulation number of each site depends on the available duration, the available node number, and the performance of the site. (4) The program made the testbeds behavior like a virtual machine, and each site behavior like a node. An MD process invoked computation on a virtual site which was dynamically bound to a real one allocated from GRPLib at runtime. Once the computation was over, the real site might be unbound from the virtual site dependent on if the next stage could be computed immediately or not. Therefore, the QM computation of an image might be migrated from one site to another one.

6 Concluding Remarks

We have gridified a physical application for the simulation of H diffusion in materials based on an integrated framework for grid enablement. This framework mainly consists of three components: a hybrid GridRPC+MPI programming model, a resource allocator, and a application internal task scheduler. The GridRPC+MPI programming model provides the application with flexibility and efficiency, the resource allocator guarantees the application with reliable on-demand resources, and the task scheduler ensures all computation tasks to execute sustainably and cooperatively. Accordingly, the application is capable of fault-tolerance and load-balance in some degree. A two days simulation has been successfully performed over a Trans-Pacific grid infrastructure consisting of the TeraGrid and AIST testbeds. The experiment has yielded very large-scale results which are difficult to achieve with convential single-system supercomputers due to the large scale. At the same time, this work has proved that the framework is practically usable and efficient for grid enablement. We believe this case study can provide helpful tips to readers to successfully accomplish their own grid enablement.

Acknowledgments

We are grateful to TeraGrid executive committee members for providing their computing resources for the Grid experiments. The work at USC was partially supported by ARL-MURI, Chevron-CiSoft, DOE, DTRA, and NSF. The work in Japan was partially supported by JST-CREST.

References

[1] H. Jónsson, G. Mills, and K. W. Jacobson: Nudged elastic band method for finding minimum energy paths of transitions, in Classical and Quantum Dynamics in Condensed Phase Simulations, edited by B.J. Berne, G. Ciccotti, and D.F. Coker (World Scientific, Singapore), pp. 385, 1998.

[2] S. Ogata, E. Lidorikis, F. Shimojo, A. Nakano, P. Vashishta, and R.K. Kalia: Hybrid finite-element/molecular-dynamics/electronic-density-functional approach to materials simulations on parallel computers, Comp. Phys. Comm. 138, 143, 2001.

[3] T. Kouno and S. Ogata: Activation Energy for Oxygen Diffusion in Strained Silicon: A Hybrid Quantum-Classical Simulation Study with the Nudged Elastic Band Method, J. Phys. Soc. Jpn. 77 (2008) 54708-1-10.

[4] S. Ogata: Buffered-cluster method for hybridization of density- functional theory and classical molecular dynamics: Application to stress-dependent reaction of H2O on nanostructured Si, Phys. Rev. B. 72 (2005) 045348-045364.

[5] Y. Song, Y. Tanaka, H. Takemiya, H. Nakada, S. Sekiguchi, Aiichiro Nakano, and Shuji Ogata: The Development and Evaluation of an Integrated Framework Supporting Sustainable Execution for Large-Scale Computations on Grids, International Journal of Computational Science, in printing.

[6] N. Karonis, B. Toonen, and I. Foster: A grid-enabled implementation of the message passing interface, Journal of Parallel and Distributed Computing, Vol.63 , No.5 , 2003, pp. 551-563.

[7] M. Matsuda et al.: Performance evaluation of the GridMPI (in Japanese), Distributed and Cooperative Processing (SWoPP 2004), Aomori, Japan , 2004.

[8] Y. Song et al.: Grid enablement and performance evaluation for simulators of nanoscale measurements of material properties, in Proceedings of HPCAsia 2005, Beijing, China, 2005, pp.195-204.

[9] K. Seymour, H. Nakada, S. Matsuoka, J. Dongarra, C. Lee, and H. Casanova: Overview of GridRPC: a remote procedure call API for grid computing, in Grid Computing - GRID 2002, LNCS 2536, 2002, pp. 274-278.

[10] Y. Tanaka, H. Nakada, S. Sekiguchi, T. Suzumura, and S. Matsuoka: Ninf-G: a reference implementation of RPC based programming middleware for grid computing, Journal of Grid Computing, Vol 1. No. 1, 2003, pp. 41-51.

[11] S. Agrawal, J. Dongarra, K. Seymour, and S. Vadhiyar: NetSolve: past, present, and future - a look at a grid enabled server, in Grid Computing: Making the Global Infrastructure a Reality, Berman, F., Fox, G., Hey, A. eds. Wiley Publishing, 2003.

[12] H. Takemiya,Y. Tanaka, S. Sekiguchi et al.: Sustainable adaptive grid supercomputing: multiscale simulation of semiconductor processing across the Pacific, In Proceedings of SC2006, ACM/IEEE Supercomputing 06, Tampa, Florida, 2006.

[13] G. Fox and D. Gannon: Workflow in Grid Systems, Concurrency and Computation: Practice & Experience, Vol. 18 , No. 1, pp. 1009-1019, 2006;

[14] A. Krishnan: Distributed Workflows in Bioinformatics, in Grid Computing for Bioinformatics and Computational Biology, edited by E. Talbi and A. Y. Zomaya(Wiley-Interscience, USA), 2007.

[15] X. Lin, X. Sun, X. Lu, Q. Deng, M, Li, H. Liu, Y. Qi, and L. Chen: Recent Advances in CFD Grid Application Platform, in proceedings of 2004 IEEE International Conference on Services Computing, pp. 588-591, 2004.

[16] M. Alfredsson et al.: eMinerals: Science Outcomes enabled by new Grid Tools, in Proceedings of the UK e-Science All Hands Meeting, UK, 2005.

[17] Clusterresources Homepage: http://www.clusterresources.com/.

[18] C. Catlett et al.:TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications, HPC and Grids in Action, Advances in Parallel Computing, Amsterdam, 2007.