A Workflow Management and Grid Computing Approach to Molecular Simulation-Based Bio/Nano Experiments

Karpjoo Jeong[1], Dongwook Kim[1], Moon Hae Kim[1],
Suntae Hwang[2], Seunho Jung[3], Youngho Lim[4], and Sangsan Lee[5]

[1] College of Information and Communication, Konkuk University, Seoul, Korea
{jeongk, dwkim, mhkim}@konkuk.ac.kr
[2] Department of Computer Science, Kookmin University, Seoul, Korea
sthwang@kookmin.ac.kr
[3] Department of Microbial Engineering, Konkuk University, Seoul, Korea
shjung@konkuk.ac.kr
[4] Department of Applied Biology & Chemistry, Konkuk University, Seoul, Korea
yoongho@konkuk.ac.kr
[5] Supercomputing Center, Korea Institute of Science and Technology Information, Daejon, Korea
sslee@hpcnet.ne.kr

Abstract. In this paper, we propose an approach to molecular simulation-based experiments which combines workflow management and grid computing techniques to address both the computing issue due to the challenging computation requirement from molecular simulation and the management issue due to distributed, heterogeneous computing platforms. We present a workflow management system customized for such experiments and explain how this workflow system can be integrated with computational grids.

Keywords: Molecular Simulation, Grid Computing, Workflow Management

1 Motivations

1.1 Molecular Simulation-Based Analysis of Bio/Nano Materials

Molecular simulation is an effective approach to computing and analyzing behaviours of biologically or chemically relevant molecular structures (e.g., proteins, DNA, RNA) easily by software tools such as CHARMM and GAUSSIAN[2, 9] without going through long, complicated, labor-intensive real experiments. In spite of such apparent advantage, molecular simulation has not been widely used because it has a challenging requirement for computation which can be satisfied only by expensive parallel computers.

Compuational grids which aim at combining computing resources from a number of organizations into large scale parallel/distributed computing platforms give us an economical solution to address the challenging computing issue

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2660, pp. 1117–1126, 2003.

of molecular simulation[5, 6, 8]. However, in order to make grid computing techniques effective for molecular simulation-based experiments, the following issues must be addressed:

- *Single System Image*. Compuational grids are, in nature, wide-area heterogeneous distributed systems. Complicated issues involved in heterogeneous distributed computing must be hidden from scientists whose knowledge about distributed computing is limited. That is, the single system image is needed.
- *Workflow Management*. Molecular simulation-based experiments are composed of various complicated tasks such as molecular structure modelling, model verfication, simulation progress monitoring, molecular simulation, simulation result verfication. They form a complex workflow in which there are inter-task dependencies and whose entire execution may take a couple of months. Managing those tasks properly is already a big burden on scientists and computational grids enable much more concurrent experiments and create much more management burdens on scientists. Therefore, the workflow mangement support is crucial for effective molecular simulation-based experiments on computational grids.

1.2 Brief Review of Conventional Workflow Systems

Workflow is used to express a complex process as a set of interconnected, smaller, less complicated component tasks[11]. In various areas such as industrial and administrative process management and design processes, workflow has successfully been applied. Technically, workflow management implies the automated coordination, control, and inter-task communication.

A workflow instance is an activity involving the coordinated execution of multiple tasks performed by different processing entities[15]. These tasks could be manual, automated, either created specifically for the purpose of the workflow application being developed, or possibly already existing as legacy programs[14]. A workflow process is an automated organizational process involving both human (manual) and automated tasks.

There are many commercial workflow management systems and research prototypes developed for various applications[18]. These systems are either customized for specific application domains or designed as general purpose systems. Both classes of workflow systems fail to satisfy requirements for molecular simulation-based experiments. Apparently, the former class is not suitable and the latter class requires furthermore customization for such experiments.

In addition, these workflow systems are not designed to work with computational grids. In the grid computing area, there are currently efforts on application-specific groupware systems, but major focuses are still on general purpose middleware systems[5, 6]. In [1], Buyya, Branson, Giddy, and Abramson proposed a World Wide Grid-based virtual laboratory system for drug designs. Although it is aimed at molecular simulation, this system is aimed at processing a large number of independent docking tasks by computational grids and assumes

no human interventions. However, our work assumes both human interventions and the dependency among molecular simulation tasks.

There are research efforts on scientific workflow systems[4, 14, 17, 20]. Although they address various issues involved in scientific workflows and biological research processes, they do not address those issues specifically involved in molecular simulation-based experiments. or furthermore issues on grid computing at all.

For these reasons, workflow systems customized for molecular simulation-based experiments and designed to support computational grids must be needed in order to make molecular simulation a widely-applicable effective technique for various bio/nano research work.

2 Workflow Model for Molecular Simulation-Based Experiments

We propose a workflow model for the molecular simulation-based experiment. The model is designed to be simple because the fundamental structure of the experimental process is not complicated and this simple design makes the system easy to use. Instead of one of existing workflow systems, we decided to develop a custom workflow system because most scientists need to use conventional simulation software tools (i.e., *legacy programs* widely applied and these tools must be integrated in a seamless fashion. Furthermore, we aim at design our workflow system to support grid computing.

The workflow model is defined to consist of:

- a set of *Programs*. We assume simulation tools to be registered in the system. The management and control information about each software are stored in the workflow system:
  - `Program ID`. We assume a system-wide unique name for each program.
  - `Interaction type`. A program can be run as a background process or requires GUI interactions with the user. The latter class of programs must be run locally or requires local GUI emulation facilities for remote execution (e.g., thin client features).
  - `Invocation Parameters`. They can be default values or must be required at invocation time.
  - `I/O file specification` . File names, file types (e.g., file extensions) and file content viewers are specified.
- a set of *Tasks*. We assume each task to be associated with a program which is pre-registered in the workflow system. That is, the task is defined as the function of the program. The information about each task is:
  - `Task ID`. We assume a task to have a system-wide unique name.
  - `Program ID`. The name of the program to execute is specified.
  - `Invocation parameter values`. Values for invocation parameters are given.

  - `Activation Type`. According to the activation condition, tasks are grouped into two classes: *automatically-started* and *manually-started*. The former class of tasks are is immediately initiated as soon as they are ready (e.g., all tasks prerequisite to the task are finished). When a task in the manually-started class is ready, the workflow system sends the user a notification (e.g., an email message or a popup message in the client GUI, *etc.*) and waits for him or her to ask for invocation.
  - `Monitoring Flag`, `Monitoring Program ID` and `Output File Names`. If this flag is on, then the workflow system shows the user the progress of the current task. Currently, we simply assume the progress of the program to be available as appended data in output files. In addition, we assume a monitoring program which is registered. This monitoring program periodically read newly appended data in specified output files and visualize them.
- a set of *Task Dependencies*.
  In our model, tasks are inter-connected and outputs from one task are used as inputs for another task. We call this a *task dependency*. A task can have multiple parent nodes. A task can be started only after all the parent tasks are finished.
  In addition to the execution order, a task dependency defines the *mapping information* between the output files of the parent task and the input files of the child task.

We define a *workflow instance* to be a workflow definition for a certain experiment. A workflow instance consists of a set of tasks and a set of task dependencies.

3 Workflow Management System for Molecular Simulation-Based Experiments

The workflow management system consists of:

- `Workflow Engine`. The workflow engine manages all the running workflow instances created by a number of users. The engine schedules tasks for each workflow instance.
- `Workflow Client` (GUI). The workflow client provides the user with a GUI that allows the user to create, monitor, and guide workflow instances for experiments. This client is written in JAVA and can be invoked on any platform.
- `Workflow Instance Databases`. All the workflow instances are managed by the central repository. Updates to workflow instances are transactional and therefore resilient to failure.
- `Task Template Repository`. Tasks frequently used are saved as templates in the repository. The user can retrieve and update templates for new experiments.

- `Program Repository`. Simulation tools are registered in the repository. Tasks are basically an execution of a certain simulation software and when the user defines tasks, he or she retrieves info about programs from the program respository and uses it to update task templates.
- `Program Adaptor`. Simulation programs require input files and produce output files. In order to make the workflow system independent of specific simulation programs, a program adaptor for each simulation program is built to allow the workflow system to manipulate input and output files in an uniform way.
- `Interface to Computational Grids`. The workflow system can delegate the execution job of molecular simulation software to a computation grid in a user-transparent way. The workflow system is designed to interoperate with computational grids through the Globus middleware.

4 Implementation Status and Experiments

4.1 Workflow System Implementation

Implementing groupware systems like a workflow system is challenging because those systems are required to support heterogeneous platforms and fault tolerance. Without the support for those features, groupware systems are not able to integrate a variety of legacy systems or may lose the entire intermediate work on a single failure. In order to address these issues, we use a fault-tolerant distributed computing system called Persistent Linda [12] as an implementation computing system.

The Linda model which Persistent Linda is based on allows arbitrary processes on distributed heterogeneous platforms to communicate and sychronize via virtual shared memory called tuple space[3]. Tuple space is a bag of tuples which are almost equivalent to database tuples in terms of how to access them. Due to ease of coordination and the support for heterogeneous platforms, Linda has been used for various groupware systems[7, 19, 21]. Persistent Linda extends the Linda model to support transactional persistence.

The workflow engine (in fact, a Persistent Linda application system) maintains workflow instances (i.e., tasks and task dependencies for an experiment) in the persistent tuple space and keep track of which tasks are ready to run according to task dependencies. For ready tasks that require an user notification, it first informs the user which tasks are ready to run. For those which can be automatically started, it immediately invokes them.

In Linda, program invocation is executed as tuple space access operations. Figure 1 illustrates how the workflow engine works. The workflow engine creates an invocation tuple for a ready task. Then, the program adaptor for the program required by the task retrieves the tuple and invokes the program. The program adaptor is designed to handle invocation details about the program.

In the workflow system, a simple shared file system is implemented on Persistent Linda. This shared file system is used to manage I/O files for simulation programs in a simple way. In this file system, metadata about I/O files is stored

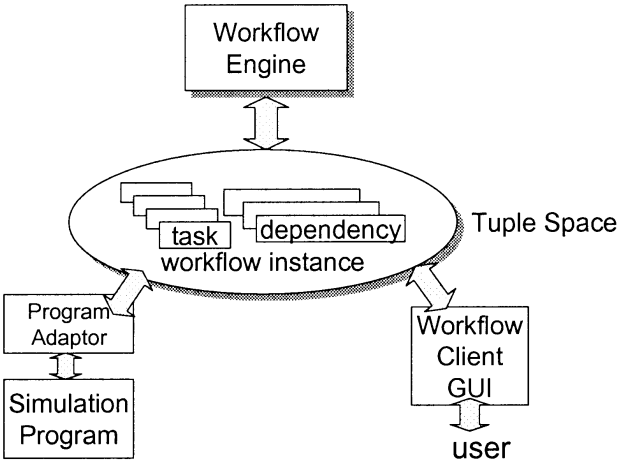

Fig. 1. How the Workflow Engine Works

in tuple space, but real data is managed as files. Program adaptors access I/O files through tuple space.

The workflow client is implemented in JAVA so that it can be run on any platform. The client displays the state of an workflow instance, gives notifications, and allows the user to examine intermediate files and to terminate tasks.

There are several molecular simulation software packages[2, 9]. Currently, our implementation of the workflow system is focused on the `CHARMM` software packages and its relevant tools. Therefore, current program adaptors are developed for those tools. `CHARMM` is one of the most popular molecular simulation package that uses the `CHARMM` force field to model the energetics, forces and dynamics of biologogical molecules by the classical method of integrating Newton's equations of motion.

For `CHARMM`, crucial input files are `inp`, `psf`, `crd` files and important output files are `out`, `trj` files. The program adaptor for `CHARMM` is designed to handle those files automatically.

4.2 Extensions to Computational Grids

In the design described in Section 4.1, the workflow engine directly interacts with program adaptors. In this section, we present how the workflow system supports computational grids.

In the grid computing-based design, the workflow engine and program adaptors can be located in different organizations which require authentication for remote communication. In such wide area environments, the workflow engine interacts with program adaptors via the *grid gateway system*. The grid gateway system consists of:

- `Gateway Agent`. The gateway agent plays the role of the delegate of a program adaptor. It receives an invocation request from the workflow engine, passes the request to the gateway server and waits for results from the server. When it receives results, then it sends them to the workflow system.
- `Gateway Server`. The gateway server manages program adaptors and waits for invocation requests from gateway agents on a number of organizations. When the server receives a request for the execution of a program, it invokes the corresponding program adaptor and waits for the adaptor to return results. When it receives results, it sends them back to the requesting agent.
- `Authentication Server`. For security, gateway agents and servers must use authentication keys from the same authentication server (`Globus CA server`).

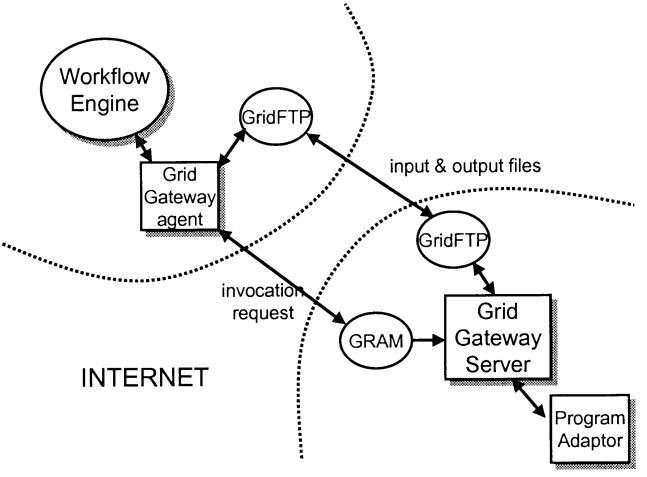

Fig. 2. Design of Computational Grid Support

Figure 2 illustrates how the grid gateway system works with the workflow system. Gateway agents and servers communicate by Globus toolkits such as `GRAM`, `GridFTP`, `Globus CA`, and `MDS`. They are implemented in JAVA in order to support heterogeneous platforms.

4.3 Experiment: Analyzing Cyclo-Oligosacchrarides for Chiral Discrimination

Our workflow system was applied for a real experiment for the chiral discrimination of R- and S-propranolol by β-cyclodextrin[13]. In this experiment, we performed computer aided molecular modeling studies such as Monte Carlo and molecular dynamics simulations.

We used CHARMM as a molecular simulation package, a Linux cluster with 32 nodes as compute server for simulation, and an SGI Octane workstation as platform for modeling, visualization, and verification. Our workflow system provided the user with the single system view. The system automatically invoked CHARMM jobs on remote nodes in the Linux cluster and tranfered input and output files among computers in a user-transparent way.

Figure 3 and 4 shows molecular models and their docking results, respectively.

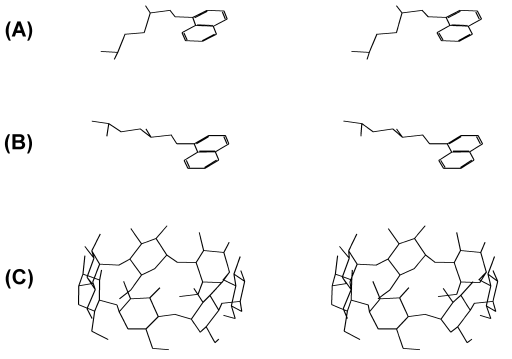

Fig. 3. Stereoview of molecular models used in the MC simulations. (A) (*R*)-propranolol, (B) (*S*)-propranolol, (C) β-CD

.

5 Conclusions and Future Work

In this paper, we assert that the approach combining workflow management and grid computing is an effective way to support large scale molecular simulation-based bio/nano experiments. We present a workflow system customized for those experiments and show how the system is integrated with computational grids in a user-transparent way.

Our experience with a real experiment for the chiral discrimination of *R*- and *S*-propranolol by β-cyclodextrin demonstrated that the workflow system enabled scientists to run many jobs on a Linux cluster concurrently without worrying about complicated distributed computing details.

The current implementation is targeted at the CHARMM molecular simulation package and its related tools. The implementation of the support for other simulation packages such as GAUSSIAN is underway. We are currently planning on

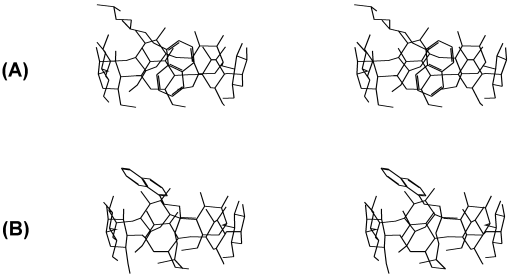

Fig. 4. Stereoview of lowest-energy configurations of the inclusion complexes of both enantiomers of propranolol and b-CD in MC docking simulations. (A) (R)-propranolol-β-CD complex, (B) (S)-propranolol-β-CD complex

large scale experiments (building a database for chiral materials) on computational grids (i.e., Globus environments) in the near future.

Acknowledgements.

This wok was partially supported by MIC (Ministry of Information and Communication) through National Grid Infrastructure Implementation Project of KISTI (Korea Institute of Science and Technology Information). It was also in part supported by University IT Research Center Project.

References

1. Buyya, R., Branson, K., Giddy, J., Abramson,D.: The Virtual Laboratory: A Toolset for Utilising the World-Wide Grid to Design Drugs, Proceedings of 2nd IEEE International Symposium on Cluster Computing and the Grid, May 2002, Berlin, Germany.
2. Brooks, B.R., Bruccoleri, R.E., Olafson, B.D., States, D.J., Swaminathan, S., Karplus, M.: CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations, Journal of Computational Chemistry, Vol. 4, 1983.
3. Carriero, N. and Gelernter, D.: How to Write Parallel Programs, The MIT Press, 1990.
4. Chin, G., Schuchardt, K., Myers, J., and Gracio, D.: Participatory Workflow Analysis: Unveiling Scientific Research Processes with Scientists, Proc. of the 6th Biennial Participatory Design Conference, 2000.

5. Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International J. Supercomputer Applications, Vol 15., No. 3, 2001.
6. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: An Open Grid Services, Architecture for Distributed Systems Integration, Jan. 2002.
7. Freeman, E., Hupfer, S., and Arnold, K.: JavaSpaces Principles, Patterns and Practice, Addison Wesley, 1999.
8. Frenkel, D. and Smit, B.: Understanding Molecular Simulation, Academic Press, 2002.
9. The Official Gaussian Home Page: http://www.gaussian.com/.
10. Goble, C.: The low down on e-science and grids for biology, Comparative and Functional Genomics, Vol. 2, pp. 365-370, 2001.
11. James, H., Hawick, K., and Coddington, P.: An Environment for Workflow Applications on Wide-Area Distributed Systems, Hawaii International Conference on System Sciences, 2001.
12. Jeong, K., Talla, S., Wyckoff, P., and Shasha, D.: An Approach to Fault Tolerant Parallel Processing on Intermittently Idle, Heterogeneous Workstations, Proc. the 27th International Symposium on Fault-Tolerant Computing, Jun. 1997.
13. Kim, H., Jeong, K., Lee, S. and Jung, S.: Molecular Modeling of the Chiral Recognition of Propranolol Enantiomers by a β-Cyclodextrin, Bulletin of Korean Chemical Society, Vol. 24, No. 1, 2003.
14. Kochut, K., Arnold, J., Sheth, A., Miller, J., Kraemer, E., Arpinar, B., Cardoso, J.: IntelliGEN: A Distributed Workflow System for Discovering Protein-Protein Interactions, International Journal on Distributed and Parallel Databases, Special Issue on Bioinformatics, 2002.
15. Krishnakumar, N. and Sheth, A.: Managing Heterogeneous Multi-system Tasks to Support Enterprise-wide Operations, Distributed and Parallel Database Journal, Vol. 3, No. 2, 1995.
16. Natrajan, A., Crowley, M., Wilkins-Diehr, N., Humphrey, M., Fox, A. and Grimshaw, A.: Studying Protein Folding on the Grid: Experiences using CHARMM on NPACI Resources under Legion, Proceeding of the HPDC Conference, San Francisco, CA, USA, Aug 7-9, 2001.
17. Peleg, M., Yeh, I., and Altman, R.: Modeling biological processes using Workflow and Petri Net models, Bioinformatics, vol. 18, pp. 825-837, 2002.
18. Riempp, G.: Wide-Area Workflow Management, Springer-Verlag, 1998.
19. Sun Microsystems, Jini Specifications, Available from Sun Microsystems WWW Site (http://java.sun.com/products/javaspaces), 1998.
20. Weske, M., Vossen, G., and Medeiros, C.: Scientific Workflow Management: WASA Architecture and Applications, Architecture and Applications, Fachbericht Angewandte Mathematik und Informatik 03/96-I, Universitat Munster, 1996.
21. Wyckoff, P., McLaughry, S., Lehman, T., Ford, D.: TSpaces, IBM System Journal, 1998.