# Incorporation of Middleware and Grid Technologies to Enhance Usability in Computational Chemistry Applications

Jerry P. Greenberg, Steve Mock, Mason Katz,Greg Bruno, Frederico Sacerdoti, Phil Papadopoulos, and Kim K. Baldridge

San Diego Supercomputer Center (SDSC)
University of California, San Diego (UCSD)
9500 Gilman Drive, Mail Code 0505, La Jolla, CA 92093-0505, USA
{jpg,mock,mjk,bruno,fds,phil,kimb}@sdsc.edu

**Abstract.** High performance computing, storage, visualization, and database infrastructures are increasing in complexity as research moves towards grid-based computing, often pushing breakthrough computational capabilities beyond the reach of scientists due to the time needed to harness the infrastructure. Hiding the underlying complexity of networked resources becomes essential if scientists are to utilize these resources in a time-effective manner. There are a myriad of solutions that have been proposed, ranging from underlying grid glue, to fully integrated problem solving environments. In this work, we discuss a workflow management system that is fully integrated with emerging grid standards but can be dynamically reconfigured. Through defined XML schema to describe both resources and application codes and interfaces, careful implementation with emerging grid standards and user-friendly interfaces, a "pluggable" event-driven model is created where grid-enabled services can be composed to form more elaborate pipelines of information processing, simulation, and visual analysis.

## 1 Introduction

High Performance Computing (HPC) has dramatically changed scientific research, and enabled advancements to be made at a rate far beyond that conceived a decade ago. HPC brings together the wealth of technology advancements not only in terms of hardware and raw computational speed, but also database technology, visualization infrastructure, algorithms, both efficient for the latest hardware as well as integrated for enhanced capability, networking and remote access through new portal web service technologies. The infrastructure complexity to make these logical components work efficiently together often overwhelms domain scientists. The ultimate goal is to extend and expand efforts in interface simplification in order to build easily reconfigurable web-based workflows that span a variety of technologies and enable complex science to be performed on the grid. Such endeavors uniquely integrate expertise in high-end integration and scientific domain experts to produce an infrastructure that meets the needs of the underlying science driver.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3036, pp. 75–82, 2004.

As scientific algorithms evolve and progress, the number and diversity of computational and grid resources options quickly multiplies. In terms of computational hardware, users may choose from a broad spectrum of possibilities from special purpose, loosely coupled platforms such as a network of PC's spread across an unknown local or remote area, to grids and cluster infrastructures, to highly coupled, highly parallel architectures. In reality, scientific users do not want to differentiate among the vast number of possibilities. Although advances in the capabilities of high-performance computers have made it possible for computational scientists and engineers to tackle increasingly challenging problems, at the same time, it has become considerably more difficult to build, manage, and integrate the software that can achieve the highest performance, use resulting data most efficiently, and/or make it production quality for the community at large. The rate-limiting step in pushing forward advanced research is often the user interface and underlying protocols required to connect to services such as hardware, database, visualization, and software.

## 2 Motivating Application

The most common way to process and analyze data obtained from computational software is via "cut and paste" from output files. Such a process is tedious and in the case that a whole class of analogous calculations is required, the effort may adversely affect the project. An alternative way is via processing data from sequential programs with command language scripts, but here as well, there are several obstacles. For example, a change in the output format of the software program may result in a failure of the script to process the output. Instead, it is more efficient to put structured data output facilities directly into the computational program (e.g., in this work, GAMESS [1]) as well as into any associated output analysis programs (e.g., in this work PLTORB). As well, similar incorporation into the output of other computational codes can be done, potentially opening up opportunities for hybrid integration. In this work, we give an example of such an integration using GAMESS and APBS [2], two molecular based software modeling tools. The result, though more laborious initially then parsing output files in the traditional way, is that dependency on data parsing is removed and the data put into a form that may be used for database storage, querying, and efficient transport over the grid in the form of JAVA objects.

We have begun this process by designing an XML schema specifically for our computational chemistry applications. Initially, the schema provided a template for storing only basic data, such as atomic coordinates, atom types, energies, molecular orbital coefficients and basic input options. We have now also added gradients, Hessian elements, and volumetric grids associated with properties of molecules.

The current schema may be viewed at http://www.sdsc.edu/~jpg/nmi/gamess.xsd. The production of XML data documents based on this schema was accomplished by putting into GAMESS calls to a library of C functions that produce a JAVA object. We do this by using the "Castor" SourceGenerator [3] to create JAVA source that maps XML elements to JAVA classes. By linking GAMESS to the XML/C library together with the JNI (Java Native Interface)[4] library which allows one to call JAVA methods from C, the data may be stored as a JAVA object and may be marshaled into an ascii XML file as well.

As an example, consider a call within GAMESS to send the coordinates and name of one atom to the GAMESS JAVA object:

```
call output_coord(ANAM(IAT),BNAM(IAT),ILEN,ZNUC,X,Y,Z)
```

The code which prints out an ascii XML document is as follows (see the schema):

```
fprintf("fp,"<ATOM_POSITION>\n");
fprintf(fp,"      <XCOORD>%s</XCOORD>\n",fixupdouble(*x,"%lf",string));
fprintf(fp,"      <YCOORD>%s</YCOORD>\n",fixupdouble(*y,"%lf",string));
fprintf(fp,"      <ZCOORD>%s</ZCOORD>\n",fixupdouble(*z,"%lf",string));
fprintf(fp,"   </ATOM_POSITION>\n");
```

The equivalent for writing to the JAVA object is:

```
callmethodn("g_add_atom ");
callmethodd("g_set_xcoord",x);
callmethodd("g_set_ycoord",y);
callmethodd("g_set_zcoord",z);
```

That is, the "g_add_atom" method instantiates a new "ATOM" and "ATOM_POSITION" object, and the subsequent calls add the data for this atom to the object.

The "g_add_atom" method is given below:

```
public static void g_add_atom() {
  atom = new ATOM();
  atom_position = new ATOM_POSITION();
  system_state.addATOM(atom);
  atom.setATOM_POSITION(atom_position);
}
```

Now we not only have a way to store our data in a rational fashion, but also a method for delivering it to other programs. Consider the GAMESS auxiliary program PLTORB3D, which uses the calculated wavefunction to create a 3D orthogonal grid of molecular orbital values. In the original implementation, separate files for the atomic basis set and input options, as well as molecular orbital coefficients had to be "cut out" of one of the GAMESS output files. Then, PLTORB3D produced a file which the visualization program QMView [5, 6] could read in order to display contours and isosurfaces. With modifications to PLTORB3D, an XML file is "unmarshalled", new data is added to the JAVA object and a new XML document is "marshaled" that contains all the original data plus the volume data. This file (or JAVA object) may be stored and retrieved in order to set up a new GAMESS run, or as part of a collection queried from a database.

## 3 Middleware Methodology

The work described above is rather specific to computational chemistry and in particular, to GAMESS related calculations. Below, we will describe our efforts to develop a general system for facilitating scientific workflows over the grid. For example, in the above discussion of possible GAMESS related workflows, we left out any details of submitting individual GAMESS jobs or PLTORB3D jobs to compute platforms or how jobs are submitted in succession. What is lacking is software that integrates the individual programs, submits jobs to particular platforms, monitors the whole process and stores results.

Previous efforts to facilitate the submission of jobs to remote platforms included the SDSC portals which are based on the NPACI Gridport [7] software. The portals provides low level tools to GLOBUS[8] for secure access to remote platforms and to the SRB (Storage Resource Broker)[9] for storing data collections. Through the web sites built with Gridport, users were shielded from the complexity of the remote platforms used for computations and archival storage. However, building portals is non-trivial and involves the intricacies associated with writing html files and cgi scripts. Additionally, once built, portals are not easily reconfigurable.

To address these problems, we are creating a general Scientific Workflow as part of the National MiddleWare Initiative [10]. The Workflow project will facilitate the building of scientific workflows from smaller tasks using web and grid services. The project is designed to shield both users and application developers from the intricacies of the grid. Resulting infrastructure will provide "hooks" to connect to user interfaces thus exposing that interface to a variety of other user interfaces.

The workflows are defined by XML documents and the workflow is divided into layers that separate users and application developers from the underlying services. The whole process is managed by a "Workflow Engine" whereby the XML documents defining a job are instantiated as JAVA objects, resources are found, the jobs are executed, and their status is sent to the user (Figure 1).

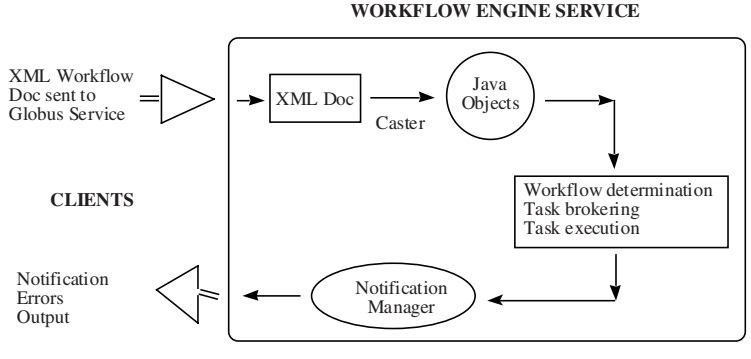


**Fig. 1.** Flowchart of a Scientific Workflow

The Workflow Engine service is not dependent on any particular application, data file format, or operating system. A user may substitute their own user interface as long they adhere to the protocols of the Workflow Engine.

# 4 Building Application Grids

## 4.1 Cluster Infrastructure and Maintenance

For decades, complex and expensive supercomputers were the only resources available for computational chemistry computations on complex molecular systems that involve large numbers of atoms, large basis sets, and/or higher order methods. Besides considerable investment in large-scale hardware itself, such platforms also require an inordinate amount of peripheral infrastructure, such as cooling infrastructure and system administration. In recent years, "commodity" clusters running Linux have become quite popular because they are often affordable by individual research groups, permit scalability to larger numbers of compute nodes, demand little in terms of special equipment or a surrounding facility, and in principle can be administered by the researchers themselves.

In practice however, cluster administration can be complex and time-consuming. There are n-fold copies of the operating system to update and maintain. If the software and the operating system is not maintained, the system may become unstable, endangering the completion of long calculations, security holes may not be patched, and software may not be updated.

The key to rapidly deploying cluster infrastructure is to automate the process of building and managing a cluster, which is itself a single grid end point. In our case the automation of building the grid-enabled clustered endpoint is achieved using the NPACI ROCKS cluster distribution. ROCKS is a cluster-aware Linux distribution that makes it possible to deploy a world class supercomputer in a matter of hours, something that has historically taken much longer.

The philosophy of ROCKS is to make the installation of the operating system the basic management tool. That is, it is easier, when automated, to reinstall all nodes to a known configuration then to determine which nodes are not synchronized. This is the opposite of the maintenance procedures on desktop systems where the operating system is rarely, if ever, reinstalled, or in configuration management tools such as CfEngine[11] that perform maintenance on exiting operating systems installations.

## 4.2 Grid Computing and Middleware

The ROCKS release also contains essential software elements of grid computing. From the end-user perspective, submitting a workflow to the grid involves interaction solely with the web portal interface, a user interface to the middlware described above, or a grid scheduler such as NIMROD[12]. That is, the underlying grid components are hidden from view for day-to-day application operations. However, this underlying system involves substantial complexity and is the focus of several grid efforts today. Figure 2 shows the high level architecture of an application grid. Once a resource is selected, GLOBUS is used to start the jobs on the end point. In addition to providing the connections from a web portal to the grid end points, the grid scheduler continuously monitors all the end points for status information such as CPU utilization, free disk space, and operation system version. This information is gathered us-

ing the GLOBUS Monitoring and Discovery Service (MDS) and allows the scheduler to make reasonable scheduling decisions.

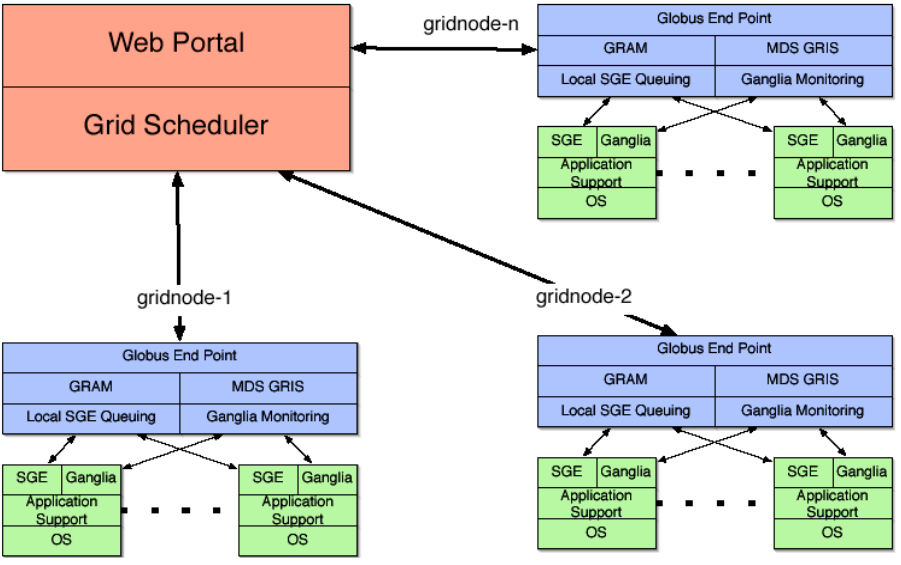


**Fig. 2.** Application Grid Architecture

Within the last year ROCKS has matured from a cluster tool to a grid tool, and now includes the GNSF Middleware Initiative's (NMI) Globus and Certificate Authority software. Because it takes only hours to build a grid-enabled cluster the only remaining step to deploying a grid is the standard Globus certificate exchange between all end points, and the grid scheduler – portal setup.

Figure 3 illustrates the basic system architecture of a ROCKS Linux cluster. The grid scheduler is responsible for submitting jobs to the local Globus Resource Application Manager (GRAM), which next submits the job to the local cluster wide system[13] which contains a current snapshot of the "state" of the cluster. This Ganglia information (memory availability, cluster size, cpu speed etc.) can then be used to feed the local Grid Resource Information Service (GRIS) information about the state and configuration of the cluster. This information is then given to the grid scheduler (Figure 2) to aid in job scheduling decisions.

GAMESS is particularly well suited for running on Linux clusters and will be included in future releases of ROCKS. It does not require any type of parallel software, is built with open source compilers, and scales reasonably well with conventional network interconnects. The rapid building of a cluster, deployment of the operating system and software via ROCKS, and the subsequent running of large scale calculations using GAMESS was demonstrated at Super Computing 2003 [14].

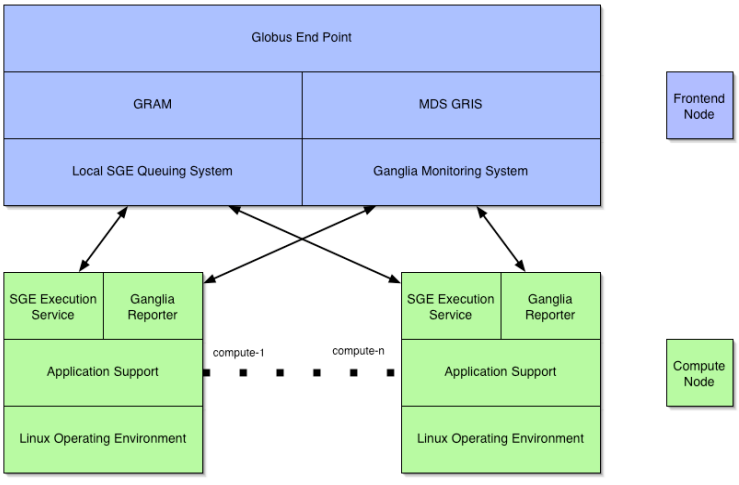


**Fig. 3.** Grid Endpoint Architecture

## 5 Conclusions

The successes of highly efficient, composite software, for molecular structure and dynamics prediction has driven the proliferation of computational tools and the development of first-generation computational chemistry grid-enabled infrastructure. The approach that we are taking, as illustrated in this work, is that of a layered architecture. The layers shield the developer from the complexity of the underlying system and provide convenient mechanisms of abstraction of functionality, interface, hardware, and software. A layered approach also helps with the logical design of a complex system by grouping together the related components while separating them into manageable parts with interfaces to join the layers together.

With such tools, researchers can begin to ask more complex questions in a variety of contexts over a range of scales, using seamless transparent computing access. As more and more realistic time simulations are enabled extending well into the nanosecond and even microsecond range at a faster turnaround time, and as problems that simply could not fit within the physical constraints of earlier generations of supercomputers become feasible the ability to integrate methodologies becomes more critical. Assembly and modeling can often utilize different computational approaches, and are best optimized for use on different computer architectures. Thus, considerations of porting and optimizing the problem-specific application for both the high-end supercomputers, and a commodity cluster of computers often factor in.

The described technology will help to tie computation with investigator intuition regardless of location, to facilitate scientific investigations by exploiting novel grid capabilities and teraflop hardware speeds, enabling direct user input and feedback. It is anticipated that such infrastructure will impact scientists that potentially need such tools for interdisciplinary research. This will in turn foster development of new modeling, data, computational technologies.

**Acknowledgements.** We acknowledge support from the NSF through DBI-0078296 and ANI-0223043 and from the NIH through NBCR-RR08605.

## References

1. Schmidt, M., Baldridge, K.K., Boatz, J.A., Elbert, S., Gordon, M., Jenson, J.H., Koeski, S., Matsunaga, N., Nguyen, K.A., Su, S.J., Windus, T.L., Dupuis, M., and Montgomery, J.A.: The General Atomic and Molecular Electronic Structure System. J. Comp. Chem. 14 (1993) 1347-1363
2. Baker, N., Holst, M., and Wang, F.: Adaptive multilevel finite element solution of the Poisson-Boltzmann equation II. Refinement at solvent-accessible surfaces in biomolecular systems. Journal of Computational Chemistry 21 (2000) 1343 - 1352
3. The Exolab Group: Castor (2002)
4. Liang, S., The JavaTM Native Interface: Programmer's Guide and Specification. 1 ed. The JAVA series. 1999: Addison Wesley Longman, Inc. 303.
5. Baldridge, K.K. and Greenberg, J.P.: QMView: A Computational 3D Visualization Tool at the Interface Between Molecules and Man. J. Mol. Graphics 13 (1995) 63-666
6. Baldridge, K.K. and Greenberg, J.P.: QMView as a SupramolecularVisualization Tool In: J. Siegel (ed.): Supramolecular Chemistry Kluwer Academic Publishers, Dordrecht Norwell New York London. (1995) 169-177.
7. Thomas, M., Mock, S., Dahan, M., Mueller, K., Sutton, D., and Boisseau, J.R. The Gridport Toolkit: a System for Building Grid Portals. in 10th IEEE International Symp. on High Perf. Comp. San Francisco (2001).
8. Foster, I. and Kesselman C.: Globus: A Metacomputing Infrastructure Toolkit. Intl. J. Supercomputing Applications 11 (1997) 115-128
9. Rajasekar, A.K. and Wan, M. SRB and SRBRack- Components if a Virtual Data Grid Architecture. in Advanced Simulation Technologies Conference. San Diego CA (2002).
10. Papadopoulos, P., Baldridge, K., and Greenberg, J., Integrating Computational Science and Grid Workflow Management Systems To Create a General Scientific Web Service Environment. NSF award (2002)
11. Burgess, M.: Cfengine: a site configurable engine. USENIX Computing Systems 8 (1995)
12. Abramson, D., Lewis, A., and Peachy, T. Nimrod/O: A Tool for Automatic Design Optimization. in The 4th International Conference on Algorithms & Architectures for Parallel Processing. Hong Kong (2000).
13. Sacerdoti, F.D., Katz, M.J., Massie, M.L., and Culler, D.E. Wide Area Cluster Monitoring with Ganglia. in Proceedings of the IEEE Cluster 2003 Conference. Hong Kong (2003).
14. Gannis, M. and Lund, G.: SDSC/NPACI Rocks Team and Sun Create Supercomputer, Run Scientific Applications in Less than Two Hours. http://www.sdsc.edu/Press/03/112403_NPACIRocks.html (2003)