Parallel Processing Letters

A CLIENT-SERVER PROTOTYPE FOR APPLICATION GRID-ENABLING TEMPLATE DESIGN

MARK L. GREEN† and RUSS MILLER‡
Center for Computational Research, University at Buffalo
Department of Computer Science and Engineering, University at Buffalo
Buffalo, New York 14260-1800, USA
†*mlgreen@ccr.buffalo.edu*
‡*miller@buffalo.edu*

Received (received date)
Revised (revised date)
Communicated by (Name of Editor)

ABSTRACT

A computational and data grid was developed at the Center for Computational Research in Buffalo, New York, in order to provide a platform to support scientific and engineering applications across a variety of computer and storage systems. This proof-of-concept grid has been deployed using a critical scientific application in the field of structural biology. The design and functionality of the prototype grid is described, along with plans for a production level grid system based on Globus.

Keywords: Grid computing; grid services; grid monitoring tools; heterogeneous grid; runtime optimization.

1. Grid-Enabled Shake-and-Bake Proof-of-Concept

The Advanced Computational Data Center Grid (ACDC-Grid) proof-of-concept has been completed using a client/server framework including a dynamically created HTML grid console for the detailed monitoring of grid jobs. Results from this study are being used in the design of the production-level Globus-based ACDC-Grid that will serve as a follow on. In particular, the proof-of-concept Unix shell scripts and client/server framework will be replaced by HTML, JavaScript, PHP, MySQL, phpMyAdmin, and the Globus Toolkit [1] in the production-level ACDC-Grid.

2. Problem Statement

In order to implement a proof-of-concept ACDC-Grid, we consider as an application a cost-effective solution to the problem of determining molecular crystal structures via direct methods as implemented in a grid setting. We use the *SnB* computer program [2], which is based on the *Shake-and-Bake* [3] method of molecular structure determination, as the prototype application for the template design presented in this paper. *Shake-and-Bake* was developed in Buffalo and is the program of choice for structure determination in many of the 500 laboratories that have acquired it [4,5,6].

In addition, the *SnB* program is well understood by the authors, one of whom is a principle author of the *Shake-and-Bake* methodology [7] and the *SnB* program [8,9]. Finally, *SnB* is a computationally intensive program that can take advantage of the grid's ability to present the user with a large-scale desktop or distributed supercomputer in order to perform computations that are equivalent to parameter studies, which are areas that the grid excels at.

The *SnB* program uses a dual-space direct-methods procedure for determining crystal structures from X-ray diffraction data. This program has been used in a routine fashion to solve difficult atomic resolution structures, containing as many as 1000 unique non-Hydrogen atoms, which could not be solved by traditional reciprocal-space routines. Recently, the focus of the *Shake-and-Bake* research team has been on the application of *SnB* to solve heavy-atom and anomalous-scattering substructures of much larger proteins provided that 3-4$\mathring{A}$ diffraction data can be measured. In fact, while direct methods had been applied successfully to substructures containing on the order of a dozen selenium sites, *SnB* has been used to determine as many as 180 selenium sites. Such solutions by *SnB* have led to the determination of complete structures containing hundreds of thousands of atoms.

The *Shake-and-Bake* procedure consists of generating structure invariants and coordinates for random-atom trial structures. Each such trial structure is subjected to a cyclical automated procedure that includes a Fourier routine to determine phase values from a proposed set of atoms (initially random), determination of a figure-of-merit [17], refining phases to locally optimize the figure-of-merit, computing a Fourier to produce an electron density map, and employing a peak-picking routine to examine the map and find the maxima. These peaks are then considered to be atoms, and the cyclical process is repeated for a predetermined number of cycles.

Trials are continually and simultaneously processed until a solution is discovered by viewing a histogram of final figure-of-merit values. The running time of this procedure ranges from minutes on PCs to months on supercomputers. For each completed trial structure, the final value of the figure-of-merit is stored in a file, and a histogram is produced to determine whether or not a solution is likely present in the set of completed trial structures. A bimodal distribution with significant separation is a typical indication that solutions are present, whereas a unimodal, bell-shaped distribution typically indicates a set comprised entirely of nonsolutions.

The current premise is that the computing framework for this *Shake-and-Bake* procedure need not be restricted to local computing resources. Therefore, a grid-based implementation of *Shake-and-Bake* methodology can afford scientists with limited local computing capabilities the opportunity to solve structures that would be beyond their means.

3. Statement of Approach

The Center for Computational Research (CCR) at the University at Buffalo serves as an ideal testbed for producing a grid-based implementation of *SnB*. Furthermore, CCR provides a diverse set of computational platforms, including the following systems.

- 4000 processor Dell PentiumIII/Xeon Cluster (DNA RNA)

- 64 processor SGI Origin 3800 (CROSBY)
- 78 processor IBM SP2 (STILLS)
- 150 processor SGI-Intel PentiumIII Cluster (NASH)
- 82 processor Sun Ultra 5 Cluster (YOUNG)
- 4 processor DEC alpha (MOONGLOWS)
- 16 processor IBM 340 Cluster (MAMAS PAPAS)
- 3 processor solar powered G4 briQ Cluster (BRIQ)
- 604 processor Dell Pentium4 Cluster (JOPLIN)
- IBM 44P Workstation (COASTERS)
- Dual Processor SGI Octane (THEDOORS)
- 6 Processor SGI Onyx (CREAM)

The Client/Server (*i.e.*, master/worker) configurations are designed for creating a simple grid that can be used to determine the necessary features of the final ACDC-Grid implementation. The proof-of-concept grid configuration and definitions include the following (refer to Figure 1).

- The *Grid Server* is the unique master grid process on the ACDC-Grid. Users submit *SnB* jobs to the Grid Server. In addition, the Grid Server is used to automatically configure and register *Platform Servers*, which control individual compute platforms. Communication between the Platform Servers and the Grid Server provide the status of the compute platform required for load balancing the entire grid. The Grid Server also maintains the *SnB* trial database, including records of where trials have been sent for processing and results of trials that have been completed.
- A *Platform Server* is a process used to register and configure Node Servers, which manage *SnB* worker processes. A Platform Server is also used to manage Block Queues (*i.e.*, dispatched sets of *SnB* jobs) and report the status of Node Servers to the Grid Server. There is one Platform Server running per computational grid resource (platform) configured.
- A *Node Server* is a process that runs on a platform production node and manages the *SnB* application worker processes. The number of workers executed by the Node Server is proportional to the number of processors of the production node.
- A series of Grid, Platform and Node Servers using TCP socket, and SSH secure tunneling/port forwarding facilities are used to configure the computational grid.
- Each platform uses native queuing software (PBSpro [10], OpenPBS [11], Loadleveler [12], and LSF [13]) to submit the requested Node Server processes.
- The *Node Server* receive *SnB* trials to process from a Platform Server, after the trials are complete the server transmits the trial results back to the Platform Server.
- The *Platform Servers* receive *SnB* trials from the Grid Server and transmits all Node Server trial solutions to the Grid Server.

The Grid Server is a master process used to manage and control all of the grid resources. This process can dynamically remove/add/display/control all grid

resources. Note that it is important for the user to request resources for the shortest time possible so that the resources can be released from the ACDC-Grid back into production in its native mode. Each platform queuing system determines which production nodes are available for grid resources for the requested time. The Grid Server may attempt to add or delete resource production nodes during the lifetime of the grid, as needed, or when computational resources become available. The Grid Server is the only process that the end user communicates with directly. The Grid Server can add or remove resources by using one of the three types of configurations shown in Fig. 1.

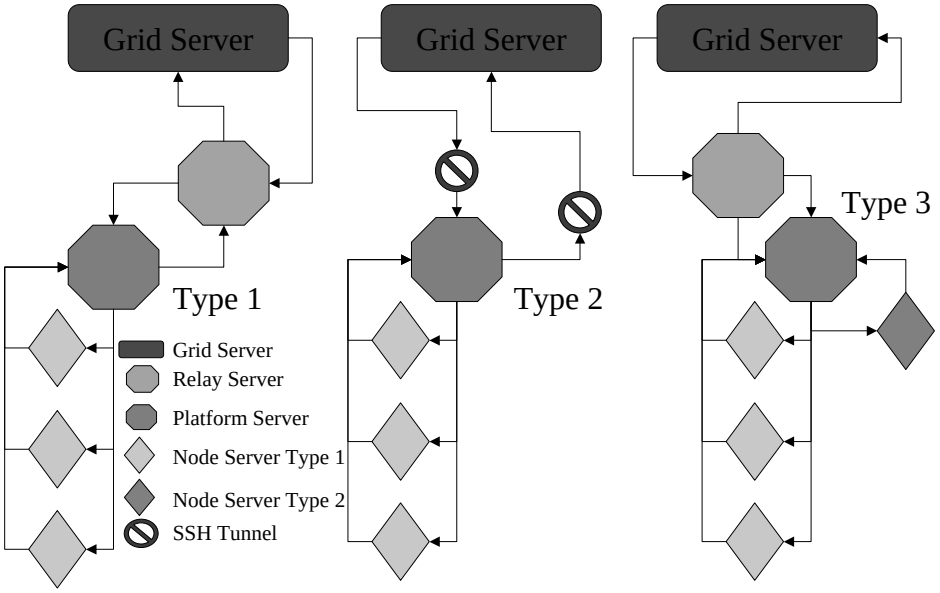

Fig. 1. Client/Server prototype application configuration. For presentation purposes each Type is shown with an associated Grid Server. In practice, only a single instance of the master Grid Server process exists.

The *Client/Server Type 1* is the standard configuration used when the platform production nodes all have the same operating system architecture. The Grid Server communicates through network socket connections to a Relay Server normally located on the front-end of a production platform. The Grid Server and Relay Server must have public Internet addresses to make the necessary network connections and can be located anywhere geographically. The sole purpose of the Relay Server is to communicate information from the public addressed Grid Server and the internally addressed Platform Server. The Platform Server can then setup network socket connections to Node Server processes running on the platform production nodes.

The *Client/Server Type 2* is the firewall cluster configuration used when a Relay

Server cannot connect to the Grid Server directly. The Grid Server communicates with the Platform Server that only has an internal Internet address through SSH encrypted tunnels setup on the cluster firewall machine. For convenience, the SSH tunnels can be setup from the Grid Server remotely and remain installed even after the grid lifetime has expired. The Platform Server can then set up network socket connections to Node Server processes running on the platform production nodes.

The *Client/Server Type 3* is used for a heterogeneous operating system cluster configuration. The Grid Server communicates with the Relay Server and Platform Server as in the standard configuration. The Node Servers in this configuration have different operating system architectures. The Platform Server is solely responsible for managing the heterogeneous Node Servers configuration.

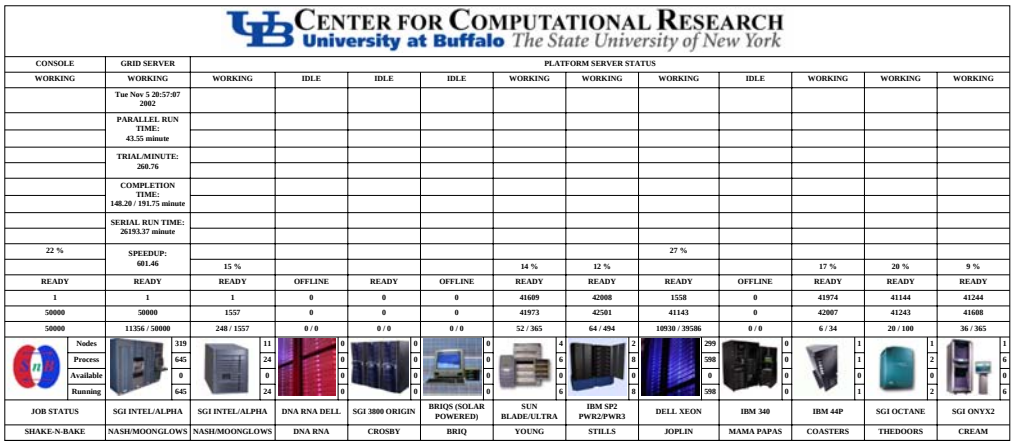

Fig. 2. Grid-enabled application status and job monitoring interface.

The proof-of-concept ACDC-Grid can be monitored from any HTML 4 compliant web browser. The dynamic HTML front-end reports the following information (Fig. 2):

- Grid Server
  - Date
  - Parallel Run Time
  - Trial/Minute (rate of progress)
  - Completion Time (estimated remaining time / total completion time)
  - Serial Run Time (estimate)
  - Speedup (estimate)
  - Start Trial Number
  - Finish Trial Number
  - Number of Trials Executed / Total Trials
- Platform Server Status

  - Platform - Picture / Architecture Type / Machine Name
  - Status - Idle / Working / Offline
  - Resources - Nodes / Total Process / Available Process / Running Process
  - Start Trial Number
  - Finish Trial Number
  - Number of Trials Executed / Total Trials
- Console
  - Start Trial Number
  - Finish Trial Number
  - Total Trials (total number of trials to process)
  - Platform Server State - Block Queue / Float / Race

Since not all of the computational resources are dedicated to the ACDC-Grid, any time that a user wishes to run a grid-enabled application, a request must be submitted through the batch queue. If a production machine is 100% utilized, then the submitted request will be held in the respective platform queue until sufficient production nodes are available.

Consider the run associated with Fig. 2 which highlights a number of issues that can be encountered when submitting a job to the ACDC-Grid. Note that the user started the grid-enabled *SnB* job before all of the computational resources were configured. The "IDLE" status at the top of the Platform Servers Status area indicates that four of the platforms are not currently processing any trials. The "OFFLINE" status associated with three of the platforms is also on, indicating that the submitted request for production nodes on those platforms has been queued. Conversely, the CROSBY platform (refer to the itemized list of machines presented earlier in this section for a correlation between machine and machine name) shows a "READY" status, which means that after the grid-enabled *SnB* application run was initiated, the queued request for production nodes was submitted.

This situation is commonplace when dealing with multiple independent heterogeneous platforms. That is, platforms that become available after an application run has already started are configured and remain "IDLE" until a new application run is issued. The *SnB* application, as shown in Fig. 3, was instructed to run 50,000 trials numbered 1 to 50,000 with 9 Platform Servers and 319 nodes yielding 645 processes. From the dynamically created web page, one can see that the application has been running for 135.58 minutes and processing trials at the rate of 265.43 trials per minute. In addition, the estimated total completion time is 188.37 minutes with 52.79 minutes left in the run, and the estimated speedup over aggregate sequential computing is 609.45.

Next, we consider the load balancing scheme used in our proof-of-concept analysis. The scheme consists of three phases, namely, Block Queue, Float, and Race. These phases are defined below.

- Block Queues
  - Each Platform Server is given a block of trials to process from the Grid Server based on a platform load factor determined from the time required

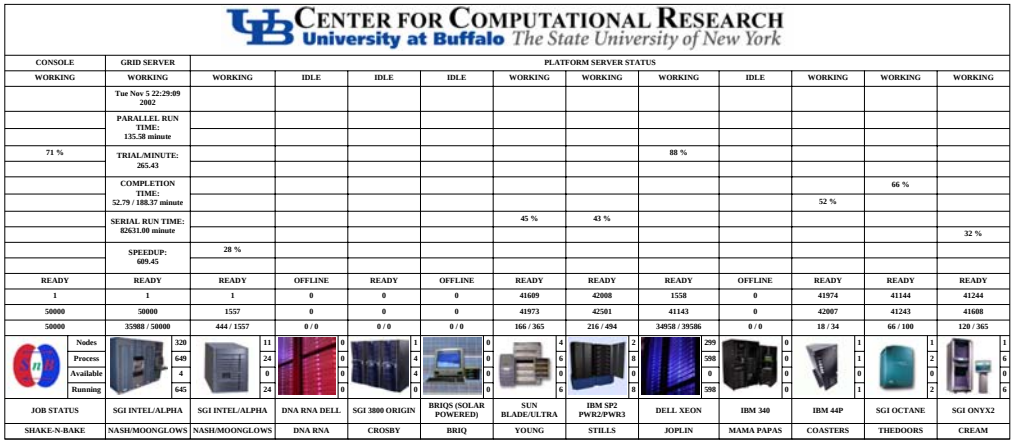

Fig. 3. Grid-enabled application job status and description.

to process one trial. A Platform Server is completely responsible for submitting the assigned trials to its registered Node Servers for processing.
  - The Block Queues account for 85% of the total number of *Shake-and-Bake* trials submitted for processing. The remaining 15% of the trials are allocated to the Float queue.

- Float
  - The *Shake-and-Bake* trials that have been reserved for Float are submitted individually to any Platform Server that requests more trials to process from the Grid Server.
  - The distribution of Float trials to Platform Servers is determined at runtime and is completely dynamic.
  - Any Platform Server in the grid can process the Float trials after their respective Block Queue has been completed.
- Race
  - When a Platform Server requests more trials to processes from the Grid Server and the Float trials have been exhausted, the Grid Server queries its database of results and assigns trials that have been assigned, but not completed, to the requesting Platform Server for processing. This continues until solutions have been obtained and recorded in the Grid Server database for all requested trials.

In terms of the Race Trials, the reader may note that there are a variety of situations under which allocated trials were not completed, including, but not limited to, the following.

- A platform is still in the process of working on its Block Queue trials.

- Trial results were lost during transmission.
- The platform that processed a trial has gone offline or lost network connectivity without transmitting trial results.

Thus, for whatever reason the trial result was not received, it will be resubmitted for processing during the Race phase. Therefore, the *SnB* grid application is robust with respect to assigning trials to multiple compute engines and will finish the requested application job as long as one Platform Server with one Node Server remains active and able to process trials.

The number of trials assigned to every Platform Server is determined by initially timing one trial on every available compute engine, this process is initiated automatically by the Grid Server. Specifically, all Node Servers that have registered with a Platform Server are assigned identical *SnB* trials for processing by the Grid Server. The time recorded includes processing and network latency. The Grid Server automatically records the trial solution and timing information for each Node Server. The platform speed (T_p) can now be calculated as the harmonic mean of the respective platform's Node Server solution times. Some variable definitions for the platform speed and load factors follow.

- P_t is the total number of Platform Servers
- N_t is the total number of Node Servers
- T_p is the platform speed given in seconds
- T_{pm} is the mean platform speed given in seconds
- T_n is the node trial time given in seconds
- P_p is the number of platform processes

The platform speed is given by

$$T_{p_i} = \left(\frac{1}{N_t} \sum_{j=1}^{N_t} \frac{1}{T_{n_j}} \right)^{-1} \quad for \quad i = 1 \cdots P_t, \tag{1}$$

where T_{p_i} denotes the ith Platform Server speed. The Platform Servers mean is calculated by

$$T_{pm} = \frac{1}{P_t} \sum_{i=1}^{P_t} T_{p_i}. \tag{2}$$

In order to take into consideration the number of platform processors, a Platform Load Factor (PLF) is calculated as

$$PLF_i = \frac{P_{p_i} + P_{p_i} \left(\frac{T_{pm} - T_{p_i}}{T_{pm}} \right)}{\sum_{j=1}^{P_t} \left[P_{p_j} + P_{p_j} \left(\frac{T_{pm} - T_{p_j}}{T_{pm}} \right) \right]} \quad for \quad i = 1 \cdots P_t, \tag{3}$$

where it should be noted that all PLF values range from $0 - 1$ and the PLF's sum to 1 as a result of the above normalization, where the denominator denotes the "effective number of processes" for all Grid Platform Servers. The PLF values are used to determine the size of each of the Platform Servers Block Queues.

The bars located in the Platform Server Status columns, as shown in Fig. 4, indicate the percentage of the Block Queue that has been completed. For instance, the Grid Server assigned 39586 trials to JOPLIN for its Block Queue, beginning at trial number 1558 and ending at trial number 41143. From the snapshot given in Fig. 3, we see that JOPLIN has thusfar completed 88% of its queue (34958 trials). The bars located in the Console column indicate the percentage of the total job trials that have been completed. Fig. 4 shows the state of the grid-enabled *SnB* application approximately one hour later. In this case, the Float Trials allocated to JOPLIN were completely processed and JOPLIN is now available to accept Race Trials. That is, JOPLIN is prepared to process trials that have been assigned, but not completed, by other platforms. In Fig. 3 note that the JOPLIN status is "RACING" and 97% of the total *SnB* application trials have been completed. The total parallel run time is 183.6 minutes with an estimated time to completion of 4.27 minutes and an overall estimated speedup of 610.84.

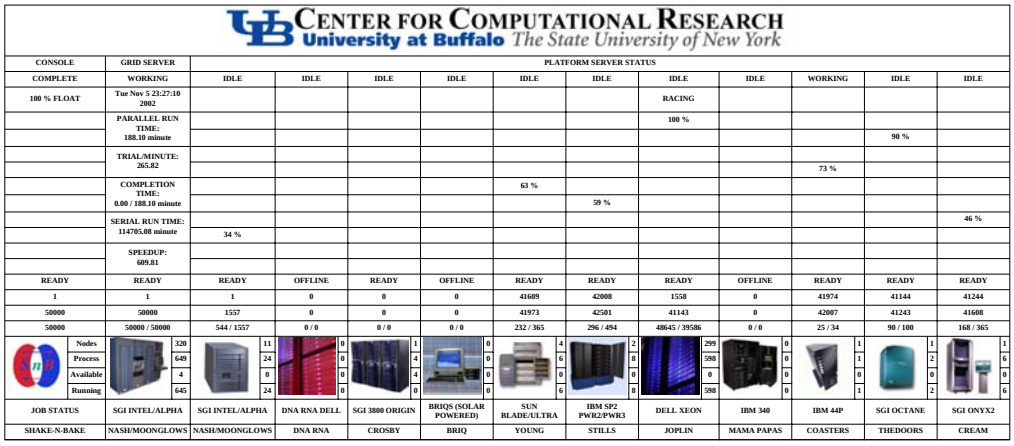

Fig. 4. Grid-enabled application dynamic HTML console and load balancing monitor.

We have chosen this grid-enabled *SnB* application run to illustrate the importance of implementing robust forms of fault recovery and load balancing. This is essential when using a wide range of heterogeneous computational platforms. The current load-balancing scheme produces a robust system that drives the grid platforms toward 100% usage until the application job has been completed. Further research into more adaptive schemes of load balancing is currently being investigated along with incorporating methods derived from other researchers [14].

4. ACDC-Grid Application Template Design

The ACDC-Grid uses a web portal, as shown in Fig. 5, in order to present users with a single point of access to grid-enabled resources. The web server is set up with the Redhat Linux operating system and Apache HTTP server. The Apache

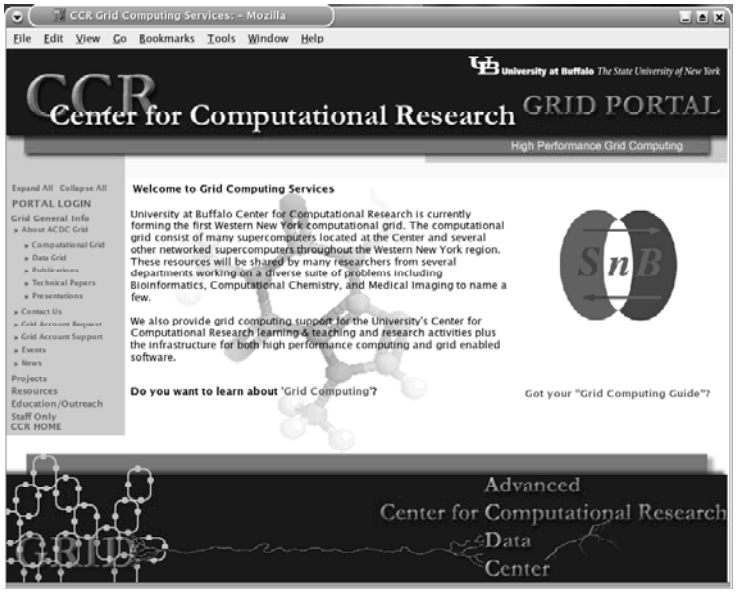

Fig. 5. Advanced Computational Data Center (ACDC) Grid Portal web interface.

server is an open-source HTTP server for UNIX and provides a secure, efficient, and extensible platform for the grid web portal. JavaScript is used for client side form verification and user support functionality. PHP is configured with the Apache HTTP server for designing the web portal template pages and the efficient creation of dynamic web pages. Furthermore, the PHP code does not have to be compiled as it is interpreted "on-the-fly" by the web server. This results in web pages that can be highly customized for an authenticated user as they browse the web portal for resources.

During the proof-of-concept grid-enabling of *SnB*, it was evident that a fine grained authentication approach would be needed to manage grid users and several of the scientific applications data and computational requirements. The following grid-enabling template was designed to provide a flexible and extensible environment for application development and management:

- Software designation
  - ACDC-Grid Web Portal authentication
  - Grid-enabled application identification
- Data requirements
  - ACDC-Grid Web Portal authentication
  - Automated Data Grid resource management

  - Grid user upload/download data files
  - Data Grid management interface
- Resource information/management
  - Monitoring and Discovery Service (MDS) information query
  - ACDC-Grid Computational resource management
  - Historical Computational resource statistics
- Job definition
  - Grid user profile management
  - Computational requirement definition
  - Predictive estimate of resource consumption
- Job verification/review
  - Verify application integrity
  - Verify resource integrity
  - Verify Data Grid file access
  - Review proposed ACDC-Grid enabled job definition
- Scenario manager
  - Query updated ACDC-Grid resource status
  - Query updated ACDC Data Grid status
  - Present execution scenario matrix
  - Select/Reject/User define execution scenario
- Execute ACDC-Grid application
  - Stage grid-enabled job files
  - Submit execution scenario
  - Create/Update job monitoring database

The grid-enabling template is created from the grid user profile that contains the users standard information uid, name, organization, address, etc., and more specific information such as group id and access level information. This information is stored in a database and can be accessed through a PHP initiated query. Additionally, each grid-enabled scientific application profile contains information about specific execution parameters, required data files, optional data files, computational requirements, etc. and statistics on application historical ACDC-Grid jobs for predictive runtime estimates. MySQL provides the speed and reliability required for this task and it is currently being used as the ACDC-Grid Web Portal database provider.

The grid-enabled version of *SnB* defined the core functionality that many scientific and engineering applications require for execution in the ACDC-Grid environment. We have identified that sequentially defining milestones for the user to complete intuitively guides them through the application workflow:

- **Software**: user chooses a grid-enabled software application.
- **Data**: user selects the required and/or optional data files from the ACDC Data Grid.
- **Resources**: user defined computational requirements are input or a template defined computational requirement runtime estimate is selected.
- **Job Definition**: user defines application specific runtime parameters or accepts default template parameter definitions.
- **Review**: user accepts the template complete job definition workflow or corrects any part of job definition.
- **Scenario**: user inputs a execution scenario or selects a template defined execution scenario.

Each item of the job definition workflow is then stored in the MySQL web portal database so the grid user may use/modify any previously created workflow in creating new job definitions. The job definitions can also be accessed via batch script files for executing hundreds of similar workflows in an automated fashion. For example, a grid user would first define/save a relatively generic job workflow template for the grid-enabled application and then use the batch script capabilities to change the job definition workflow data files or application parameters and execute a series of new grid jobs.

The Globus Toolkit 2.2 [1,15] was installed on the web server and its Monitoring and Discovery Service (MDS) is used for monitoring the grid computational resources and the web portal server. The MDS includes a Grid Resource Information Service (GRIS) [15,16] that obtains platform information and Grid Index Information Service (GIIS) [15,16] that aggregates the information for an entire grid. MDS provides the necessary tools to build an LDAP-based information infrastructure for the computational grid and OpenLDAP is used to host the grid resource information directory. The LDAP server provides the following types of data:

- platform type and instruction set architecture
- operating system (host OS) name and version
- CPU information (type, number of CPUs, version, speed, cache, etc.)
- physical memory (size, free space, etc.)
- virtual memory (size, free space, etc.)
- network interface information (machine names and addresses)
- file system summary (size, free space, etc.)

5. Future Directions

There is much more information required for a grid user to effectively use the web portal that is beyond the scope of this paper. We will list several areas of interest that are currently under development at the Center for Computational Research:

- **ACDC-Grid Web Portal**: The web portal provided the infrastructure required for defining the job template workflows discussed in this paper. The portal is being designed around a central database that contains the "state" of the ACDC-Grid, including all of the grid user and resource information, network status, Data Grid status, running/queued jobs, historical jobs, and so forth. The "state" of the ACDC-Grid can be queried by grid users with appropriate access level authentication or the grid-enabled application job template infrastructure providing up-to-date resource information.

- **ACDC-Grid Job Monitoring**: The ACDC-Grid Job Monitoring system is being designed to be an extremely light-weight and non-intrusive tool for monitoring applications and resources on computational grids. It also provides a historical retrospective of the utilization of such resources, which can be used to track efficiency, adjust grid-based scheduling, and perform a predictive assignment of applications to resources. The intelligent management of consumable resources, including computational cycles, requires accurate up-to-date information. The ACDC Job Monitoring system provides near real-time snap shots of critical computational job metrics, which are stored in a database and utilized by dynamic web pages that it generates for the user.

- **ACDC Data Grid**: The data grid will enable the transparent migration of data between various resources while preserving uniform access for the grid users through the ACDC-Grid Web Portal. Basic file management functions accessible via a platform-independent web interface feature user friendly menus, upload/download functionality, file editing, search and discovery service, sorting and file transfer mechanisms.

- **Grid-enabled Data Mining**: The web portal also provides access to data mining jobs that can be submitted in a dedicated mode (time critical), where jobs are queued on ACDC-Grid resources, or in a back fill mode (non-time critical), where jobs are submitted to ACDC-Grid resource that have unused cycles available. ACDC-Grid job management includes: automatic determination of appropriate execution times, number of trials, and number of processors for each available resource, logging and status of all concurrently executing resource jobs, automatic incorporation of *SnB* trial results into the molecular structure database, and post processing of updated database for subsequent job submissions.

Our experience with the *SnB* prototype and template design has provided a solid foundation for the current grid-enabling infrastructure development. We believe that the template approach will continue to compliment the computational and data grid development work. In the future, we envision an application grid-enabling template API that will provide application developers the interface and tools required to port their existing scientific and engineering applications to a grid environment.

Acknowledgments

Support was provided by the Center for Computational Research at the University at Buffalo and the National Science Foundation under award ACI-0204918. We would also like to acknowledge visualization support from Martins Innus, Scientific Visualization Specialist and Adam Konak, Multimedia Specialist, from the Center for Computational Research, University at Buffalo.

References

[1] The Globus Project, 2002. http://www.globus.org/
[2] R. Miller, S.M. Gallo, H.G. Khalak, and C.M. Weeks, *SnB*: Crystal structure determination via Shake-and-Bake, *J. of Applied Crystallography* (1994), 27, pp. 613–621.
[3] C.M. Weeks, G.T. DeTitta, H.A. Hauptman, P. Thuman, and R. Miller, Structure solution by minimal function phase refinement and Fourier filtering: II. implementation and applications, Acta Cryst. A50, 1994, pp. 210–220
[4] H.A. Hauptman, H. Xu, C.M. Weeks, and R. Miller, Exponential *Shake-and-Bake*: theoretical basis and applications, *Acta Crystallographica* A55, 1999, pp. 891–900.
[5] C.M. Weeks and R. Miller, The design and implementation of *SnB* v2.0, *J. of Applied Crystallography* 32, 1999, pp. 120–124.
[6] C.M. Weeks and R. Miller, Optimizing *Shake-and-Bake* for proteins, *Acta Crystallographica* D55, 1999, pp. 492–500.
[7] G.T. DeTitta, C.M. Weeks, P. Thuman, R. Miller, and H.A. Hauptman, Structure solution by minimal function phase refinement and Fourier filtering: theoretical basis, Acta Cryst. A50, 1994, pp. 203–210.
[8] J. Rappleye, M. Innus, C.M. Weeks, and R. Miller, SnB v2.2: An Example of Crystallographic Multiprocessing, Journal of Applied Crystallography 35, 2002, pp. 374-376.
[9] M.L. Green and R. Miller, Grid computing in Buffalo,New York, Annals of the European Academy of Sciences, 2003, pp. 191-218.
[10] Portable Batch System Professional, 2002.http://www.pbspro.com/
[11] Open Portable Batch System, 2002. http://www.openpbs.org/
[12] IBM Loadleveler, 2002. http://www-1.ibm.com/servers/eserver/pseries/library/sp_books/loadleveler.html
[13] Platform LSF 5, 2002. http://www.platform.com/products/wm/LSF/index.asp
[14] Alt, M., Bischof, H. and Gorlatch, S., "Program development for computational Grids using skeletons and performance prediction," Parallel Processing Letters, Vol. 12, No. 2 (2002) pp. 157-174.
[15] I. Foster and C. Kesselman, "Globus: A Toolkit-Based Grid Architecture," The Grid: Blueprint for a New Computing Infrastructure, I. Foster and C. Kesselman, eds., Morgan Kaufmann, San Francisco, 1999, pp. 259–278.
[16] I. Foster et al., "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration," tech. report, Globus Project; http://www.globus.org/research/papers/ogsa.pdf (current June2002).
[17] H.A. Hauptman, A minimal principle in the phase problem, In Crystallographic Computing 5: from Chemistry to Biology, edited by D.Moras, A.D. Podjarny and J.C. Thierry, pp. 324–332, Oxford: IUCr & Oxford University Press.