Molecular Structure Determination on a Computational and Data Grid

Russ Miller & Mark L. Green
Department of Computer Science and Engineering
Center for Computational Research
9 Norton Hall
State University of New York
Buffalo, NY 14260

Abstract: The focus of this chapter is on the design and implementation of a critical computer program in structural biology onto two computational and data grids. The first is the Buffalo-based ACDC grid, which uses facilities at SUNY-Buffalo and several research institutions in the greater Buffalo area. The second is Grid2003, an international grid established late in 2003 primarily for physics and astronomy applications. We present an overview of the ACDC Grid and Grid2003, focusing on the implementation of several new tools that we have developed for the integration of computational and data grids, lightweight job monitoring, predictive scheduling, and opportunities for improved Grid utilization through an elegant backfill facility. A new computational framework is developed for the evolutionary determination an efficient implementation of an algorithm to determine molecular crystal structures using the Shake-and-Bake *methodology. Finally, the grid-enabled data mining approach that we introduce is able to exploit computational cycles that would otherwise go unused.*

Introduction. The Grid is a rapidly emerging and expanding technology that allows geographically distributed and independently operated resources (CPU cycles, data storage, sensors, visualization devices, and a wide variety of Internet-ready instruments) to be linked together in a transparent fashion [1-3]. The power of the Grid lies not only in the aggregate computing power, data storage, and network bandwidth that can readily be brought to bear on a particular problem, but on its ease of use.

Grids are now a viable solution to certain computationally- and data-intensive computing problems for reasons that include the following.

1. The Internet is reasonably mature and able to serve as fundamental infrastructure for network-based computing.
2. Network bandwidth, which is doubling approximately every 12 months, has increased to the point of being able to provide efficient and reliable services.
3. Motivated by the fact that digital data is doubling approximately every 9 months, storage capacity has now reached commodity levels, where one can purchase a terabyte of disk for roughly the same price as a high-end PC.
4. Many instruments are Internet-aware.
5. Clusters, supercomputers, storage and visualization devices are becoming more mainstream.
6. Major applications, including critical scientific community codes, have been parallelized in order to increase their performance (faster turnaround time) and capabilities (handle larger data sets or provide finer resolution models).

7. Driven by the fact that science is a collaborative activity, often involving groups that are not co-located, collaborative environments (*i.e.*, *collaboratories*) are moving out of the alpha phase of development and into at least beta testing.

For these and other reasons, grids are starting to move out of the research laboratory and into early-adopter production systems. The focus of grid deployment continues to be on the difficult issue of developing high quality middleware.

Grids have recently moved from academic settings to corporate thrusts. Numerous grid projects have been initiated (GriPhyN, PPDG, EU DataGrid, NASA's Information Power Grid, TeraGrid, Open Science Grid, and iVDGL, to name a few). However, the construction of a real general-purpose grid is in its infancy since a true grid requires coordinated resource sharing and problem solving in a dynamic, multi-institutional scenario using standard, open, general-purpose protocols and interfaces that deliver a high quality of service.

Many types of computational tasks are naturally suited to grid environments, including data-intensive applications. Grid-based research and development activities have generally focused on applications where data is stored in files. However, in many scientific and commercial domains, database management systems play a central role in data storage, access, organization, and authorization for numerous applications. Part of our research effort is targeted at enabling systems that are more accessible within a grid framework.

As Grid computing initiatives move forward, issues of interoperability, security, performance, management, and privacy need to be carefully considered. In fact, security is concerned with various issues relating to authentication in order to insure application and data integrity. Grid initiatives are also generating best practice scheduling and resource management documents, protocols, and API specifications to enable interoperability. Several layers of security, data encryption, and certificate authorities already exist in grid-enabling toolkits such as Globus Toolkit 3 [4].

Molecular Structure Determination. *SnB* [5-7] is a computer program based on the *Shake-and-Bake* [8-9] method of molecular structure determination from X-ray diffraction data. It is the program of choice for solving such structures in many of the hundreds of laboratories that have acquired it. This computationally intensive procedure is ideally suited to an implementation on a computational and data grid. Such an implementation of *SnB* allows for the processing of a large number of related molecular trial structures [10].

The Shake-and-Bake *algorithm for molecular structure determination was listed on the IEEE poster "Top Algorithms of the 20th Century."* 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 3-4Å 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 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 computing a Fourier Transform to determine phase values from the proposed set of atoms (initially random), determining a figure-of-merit [11] associated with these phases, refining the phases to locally optimize the figure-of-merit, computing a Fourier Transform to produce an electron density map, and employing a peak-picking routine to examine the map and find the maxima. These peaks (maxima) are then considered to be atoms, and the cyclical process is repeated for a predetermined (by the user) number of cycles.

The running time of *SnB* varies widely as a function of the size of the structure, the quality of the data, the space group, and choices of critical input parameters, including the size of the Fourier grid, the number of reflections, the number and type of invariants, and the number of cycles of the procedure used per trial structure, to name a few. Therefore, the running time of the procedure can range from seconds or minutes on a PC to weeks or months on a supercomputer. Trial structures are continually and simultaneously processed, with the final figure-of-merit values of all structures stored in a file. The user can review a dynamic histogram during the processing of the trials in order to determine whether or not a solution is likely present in the set of completed trial structures.

SnB has recently been augmented with a data repository that stores information for every application of *SnB*, regardless of where the job is run. The information is sent to the repository directly from *SnB* in a transparent fashion. This information is then mined in an automated fashion in order to optimize 17 key *SnB* parameters in an effort to optimize the procedure for solving previously unknown structures, as discussed later in this chapter.

SnB has also been augmented with a 3D geographically distributed visualization tool so that investigators at geographically distributed locations can collaborate in an interactive fashion on a proposed molecular solution. Further, the tool is being generalized to handle standard formats.

Grid Computing in Buffalo. The *Advanced Computational Data Center Grid* (*ACDC-Grid*) [10,12-14], which spans organizations throughout Western New York, is a heterogeneous grid initially designed to support *SnB*. ACDC-Grid is part of Grid3+, the IBM NE BioGrid, and serves as the cornerstone for our proposed WNY-Grid. ACDC-Grid incorporates an integrated computational and data grid, lightweight job monitoring,

predictive scheduling, and opportunities for improved Grid utilization through an elegant backfill facility. The following projects and packages deliver unique and complementary components that allow for the systematic expansion of the ACDC-Grid.

- **Globus Toolkit 3** [15] provides APIs and tools using the Java SDK to simplify the development of OGSI-compliant services and clients. It supplies database services and MDS index services implemented in Java, GRAM [16] service implemented in C with a Java wrapper, GridFTP [17] services implemented in C, and a full set of Globus Toolkit 2 components based on version 2.4. The Globus Toolkit 3 Java provides C bindings for application development and integration with the existing grid application base. The recently proposed **Web Service-Resource Framework (WS-RF)** provides the concepts and interfaces developed by the OGSI specification exploiting the Web services architecture [18-21]. These specifications enable define the conventions for managing state so that applications discover, inspect, and interact with stateful resources in standard and interoperable ways [22-23].
- **The Python Globus (pyGlobus) project** [24-26] generated a Python object-oriented interface to the Globus Toolkit versions 2.2.4 and 2.4. This provides high-level scripting language access to the entire Globus toolkit with similar performance to the underlying Globus Toolkit. Integration with Python offers high-performance scientific computing access to Numerical Python [27], Scientific Python [28], the netCDF library [29], Message Passing Interface (MPI) [30], Bulk Synchronous Parallel programming (BSPlib) [31-32], and the SciPy library [33]. The **pyGridWare** project [34] provides a migration path for the pyGlobus users that need a pure Python implementation for developing automated client side tooling to interact with Globus Toolkit 3 implementation of OGSI. Whereas, **Perl** provides several different Web services implementations [35] based on SOAP and XML-RPC. The OGSI standard uses SOAP, where the best Perl module for SOAP support is SOAP::Lite [36]. The OGSI::Lite [37] package is a container for grid services that facilitates writing services in the Perl scripting language. Exporting a Perl class as a grid service can inherit the required standard OGSI classes and communicate using the SOAP::Lite package. These packages add tremendous flexibility to the ACDC-Grid enterprise grid service development effort.
- **Microsoft's .NET technology** for supplying Grid Services [38-39] to the UK e-Science community is projected to result from a collaboration between Microsoft [40] and National e-Science Centre (NeSC) [41]. The project objectives include developing an implementation of OGSI using .NET technologies and developing a suite of Grid Service demonstrators that can be deployed under this .NET OGSI implementation. The University of Virginia Grid Computing Group is developing **OGSI.NET** that provides a container framework for the .NET/Windows grid-computing world [42]. This project can bridge the gap between OGSI compliant frameworks that primarily run on Unix based systems to inter-operability with Windows based platforms within the ACDC-Grid.
- **OptimalGrid** is middleware released by IBM that aims to simplify the creation and management of large-scale connected, parallel grid applications [43]. OptimalGrid manages problem partitioning, problem piece deployment, runtime

management, dynamic level of parallelism, dynamic load balancing, and system fault tolerance and recovery. The SETI@home project [44] and the Folding@home protein-folding project [45] are examples of applications, similar in granularity to applications discussed herein, that can utilize the OptimalGrid infrastructure. These applications work in a simple "scatter/gather" mode and have no requirement for communication between the grid nodes participating in the computation.

The ACDC-Grid has been developed with critical grid components that allow for the deployment of a general-purpose regional enterprise grid residing over generally available IP networks. The *Shake-and-Bake* method of molecular structure determination, as instantiated in *SnB,* has been used as the prototype application in the development of our general-purpose grid. There are many reasons why *SnB* was chosen, including the fact that it is an important scientific code, it is widely distributed, both *Shake-and-Bake* and *SnB* were developed in Buffalo by members of the Hauptman-Woodward Medical Research Institute and the State University of New York at Buffalo, and that one of the co-developers of *Shake-and-Bake* and *SnB* is a member of the leadership team of the ACDC-Grid, which means that we have access to the knowledge base associated with *SnB* as well as all of its internals.

To date, the result of our general-purpose grid effort has been the successful deployment of a campus grid involving a variety of independent organizations throughout SUNY-Buffalo and a Western New York Grid (WNY-Grid), which provides a seamless and transparent mode of operation for grid users in the greater Buffalo region. The WNY-Grid also provides a unique framework for education, outreach, and training of grid technology and its application in the Western New York region. Finally, it should be noted that we are in the process of widening the reach of WNY-Grid in order to develop a New York State Grid (NYS-Grid). While the NYS-Grid is in its infancy, we have already secured commitments for participation by a variety of institutions in Western New York, the Southern Tier, Upstate New York, and New York City. Some of these nodes will be brought on-line in early 2005.

Center for Computational Research (CCR). The majority of the work presented in this chapter was performed at the Center for Computational Research, SUNY-Buffalo. The Center maintains a wide variety of resources that were used during various phases of the ACDC-Grid implementation, including the following.

1. Compute Systems. A 3TF peak Dell Pentium4 system with Myrinet; A 6TF peak Dell PentiumIII system with fast Ethernet; A 3TF IBM Blade Server; A 64 processor SGI Origin 3800; A 64 processor SGI Origin 3700 (Altix); A SUN cluster with Myrinet; An IBM SP; An 18 node Dell P4 visualization cluster; A heterogeneous bioinformatics system; Several SGI Onyx systems; Networks of workstations.
2. Storage Systems. A 40TB RAID5 HP SAN system with 190TB of backup tape front-ended by 64 Alpha processors that is directly connected to CCR's high-end compute platforms; Several NAS systems, some of which are targeted at CCR's Condor flocks.

3. Visualization Systems: An 11'×8' Tiled Display Wall with 20 projectors; A FakeSpace ImmersaDesk R2; An SGI Reality Center 3300W; Several Access Grid Nodes; Miscellaneous PC-based visualization systems.
4. Networking. SUNY-Buffalo is an Internet2 member and a participant in the Abeline network. CCR is directly connected to Internet2.

ACDC-Grid Overview. The development of the heterogeneous ACDC-Grid infrastructure has flourished recently with funding from an NSF/ITR. A variety of applications are available on ACDC-Grid, as are a variety of critical tools that we have developed. An overview of the ACDC-Grid effort follows.

1. Grid Core Infrastructure. The core infrastructure for the ACDC-Grid includes the installation of standard grid middleware, the deployment of an active Web portal for deploying applications, dynamic resource allocation so that clusters and networks of workstations can be scheduled to provide resources on demand, a scalable and dynamic scheduling system, and a dynamic firewall, to name a few.
2. Grid Monitoring, Scheduling, and Mining. The ACDC-Grid provides an efficient and lightweight grid monitoring system, a sophisticated predictive job scheduler that integrates past performance of users with the knowledge of availability of compute resources and knowledge of the location of the requisite data, a backfill mechanism that allows the ACDC-Grid to maximize utilization while minimizing interference with job scheduling, and a grid-enabled mechanism for data mining.
3. Data Grid and Storage Services. The ACDC-Grid Data Grid has been developed from the ground up to transparently integrate with the ACDC-Grid Computational Grid and provide the user with a representation of their data that hides critical details, such as location, making the Grid appear as a single entity to the user. That is, from the user's point of view, they have access to their data and computational resources upon which to process their data. However, the user does not need to know the location of the data or computational resources. This development included the design, analysis, and implementation of a data grid scenario manager and simulator. The Data Grid is able to utilize historical information in order to migrate data to locations that are most efficient for its analysis.
4. Applications and Collaborations. The SUNY-Buffalo Grid Team has been working closely with a number of highly-visible grids, including the International Virtual Data Grid Laboratory, Grid3+ and its technical workgroups, Open Science Grid and its technical workgroups, the Northeast Bio-Grid, MCEER, NEES, NSF funded educational grid projects at SUNY-Buffalo, the NSF/NIH supported Grid-enabled *Shake-and-Bake* package, transport modeling to support algal bloom tracking for event monitoring and response management, evolutionary aseismic design & retrofit (EADR), and OSTRICH, a general purpose software tool for parameter optimization.

Monitoring. An effective and efficient grid monitoring system was developed during the early stages of the prototype ACDC-Grid. This monitoring system was critical to the grid development group and proved useful to early application adopters. The *ACDC-Grid monitoring system* exploits the development of robust database servers. The monitoring

system utilizes a MySQL database server, which can maintain millions of records and hundreds of simultaneous connections in a fast and stable manner. In fact, the ACDC-Grid monitoring system currently contains statistics for over 300,000 computational jobs completed on CCR's heterogeneous compute platforms and over 1,600,000 jobs completed on the Grid3 multi-institutional computational resources. The ACDC-Grid monitoring infrastructure has proven to be robust and scalable, but lacks the necessary service-based tooling to be incorporated into a large general-purpose grid infrastructure. Therefore, our current efforts are targeted at a second-generation monitoring service that is more tightly integrated and configured with the unique computational resource it monitors. We believe that this second generation system will provide an order of magnitude more scalability, from tens of thousand to hundreds of thousand of servers.

The current ACDC-Grid monitoring system includes the following features.

1. Running/Queued Jobs. The ACDC-Grid monitoring system provides summary and statistics of currently running or queued jobs on Grid3. Summary charts are compiled based on total jobs, CPU hours, or runtime for either a user or group (*i.e.*, virtual organization (VO)) over an individual resource, subset of resources, or the entire grid. Each interactive chart provides the ability to display detailed job information.
2. Job History. The ACDC-Grid monitoring system provides detailed historical job information including CPU consumption rates and job production rates for either an individual user or a group over a subset of grid resources. To date, ~1,600,000 jobs that have run on Grid3 since October 2003. Summary charts are compiled from usage data based on user jobs or VOs for a given range of dates over a given set of resources. Statistics such as total jobs, average runtime, total CPU time consumed, and so forth, are dynamically produced from the available database. Each interactive chart allows for detailed information to be displayed.
3. ACDC Site Status. The ACDC-Grid monitoring system generates dynamic ACDC site status logs, reporting successful monitoring events as well as specific Grid3 site errors corresponding to monitoring event failures.

Scheduling. The *ACDC-Grid predictive scheduler* uses a database of historical jobs to profile the usage of a given resource on a user, group, or account basis [46-54]. Determining accurate quality of service estimates for grid-enabled applications can be defined in terms of a combination of historical and runtime user parameters in addition to specific resource information. Such a methodology is incorporated into the ACDC-Grid Portal, which continually refines the predictive scheduler parameters based, in part, on the data stored by the monitoring system.

Workload also plays a significant role in determining resource utilization. The native queue schedulers typically use the designated job wall-time for managing resource backfill (*i.e.*, small pockets of unutilized resources that are being held for a scheduled job). However, such systems may also use a weighted combination of node, process, and wall-time to determine a base priority for each job and subsequently modify this priority

in order to impose a fair share resource policy based on historical usage. The backfill system will allow a job with lower priority to overtake a job with higher priority if it does not delay the start of the prioritized job. The *ACDC-Grid predictive scheduler* uses historical information to better profile grid users and more accurately determine execution times. Our prototype predictive scheduling system is based on statistical principles [55] that allow jobs to more effectively run in a backfill mode.

We consider the aforementioned shared- and distributed-memory computational resources at SUNY-Buffalo's Center for Computational Research (CCR). The ACDC-Grid Portal executes many grid-enabled scientific applications on several of the Center's heterogeneous resources concurrently. Several applications have inter-dependent execution and data requirements that require reliable knowledge of job start and completion times.

An explanation of the development of the ACDC-Grid predictive scheduler is best served by considering a snapshot of the queue for a single computational resource. Table 1 shows 15 running and queued jobs on this resource (Dell P4 cluster with Myrinet) from six users, which initially completely occupy all processors on all nodes (*i.e.*, all 516 processors on the 258 dual-processor nodes). There are seven running jobs and eight queued jobs, where the queue job priority determines a relative rank for corresponding to the order that the queued jobs will start. Note that the user requests the number of nodes, number of processes, and walltime queue parameters for each of the running and queued jobs. This is enough information to completely define the job execution and the native scheduler priority determination.

Table 1. Sample computational resource queue snapshot.

Jobid	User	Nodes	Procs	Walltime	Status
1	user2	32	64	360	running
2	user1	32	64	360	running
3	user1	32	64	360	running
4	user1	32	64	360	running
5	user1	32	64	360	running
6	user3	64	128	500	running
7	user5	34	68	720	running
8	user4	96	192	720	1
9	user5	64	128	360	2
10	user5	64	128	480	3
11	user5	128	256	720	4
12	user6	128	256	720	5
13	user5	128	256	720	6
14	user6	96	192	306	7
15	user5	64	128	480	8

The native queue scheduler uses the designated job walltime for managing resource backfill and estimated job start and end times. Table 2 reports the native queue scheduler

estimates for start and end times for all running and queued jobs. The native scheduler uses a weighted combination of node, process, and walltime to determine a base priority for each job and subsequently modifies the base priority to impose a fair share resource policy. The fairshare value is based on historical usage and can be divided into user, group, and account associated with the job. This scheduling scheme is also based on advanced walltime reservations with backfill, where a job with lower priority can overtake a job with higher priority only if it does not delay the start of the prioritized job. The advanced reservation scheme also makes it possible to allocate resource in the future.

Table 2. Native queue job execution start and end time.

Jobid	Walltime	Starttime	Endtime
1	360	00:05:41	00:11:41
2	360	00:05:41	00:11:41
3	360	00:05:41	00:11:41
4	360	00:05:41	00:11:41
5	360	00:05:41	00:11:41
6	500	00:05:41	00:14:01
7	720	00:07:36	00:19:36
8	720	00:11:41	00:23:41
9	360	00:11:41	00:17:41
10	480	00:14:01	00:22:01
11	720	00:23:41	01:11:41
12	720	00:22:01	01:10:01
13	720	01:11:41	01:23:41
14	306	01:10:01	01:15:07
15	480	01:23:41	02:07:41

The ACDC-Grid predictive scheduler uses a database of historical job executions to provide an improved profile of the usage of a given resource based on a user, group, or account basis. Workload also plays a significant role in determining average system utilization. Users will take advantage of scheduler feedback to determine the type of jobs that have the best turn around time. The users will submit jobs that give them the best service, resulting in a dynamic workload that adjusts to provide near-optimal utilization. Table 3 reports five Genetic Algorithm optimized user profile parameters that were used to determine a more efficient job execution.

Table 3. ACDC-Grid user profile information.

User	Efficiency	Node	Walltime	Job	Age
user1	0.568	33	382	61	17
user2	0.421	44	447	60	38
user3	0.650	64	426	19	23
user4	0.717	96	424	16	30
user5	0.612	44	255	138	35

user6	0.691	19	423	138	20

This methodology is incorporated into the ACDC-Grid Portal, where it continually verifies and evolves the predictive scheduler parameters based on the current computational grid state. The resulting system delivers a self-adapting job start times with a factor of 2-3 times more accurate than the native queue systems.

The ACDC-Grid predictive scheduler backfill algorithm was initially designed to be extensible to a general set of multi-disciplinary applications, though it has only been deployed for the *SnB* application environment. The prototype results have been impressive. Based on predictive analysis, the ACDC-Grid infrastructure determines the length of time idle processors will be available on all computational resources. For example, over a 6 month period, the ACDC-Grid predictive scheduler has allowed 3709 heterogeneous jobs to be completed on an average of 21 processors per job with an average runtime of 7.3 hr consuming a total of 410,000 CPU hrs at the rate of 2250 CPU hrs/day.

The ACDC-Grid predictive scheduler estimates are used for determining whether or not a computational grid resource can meet the quality of service requirements defined by the current workload. If a computational grid resource will not meet the quality of service expectations required, the ACDC-Grid infrastructure will search for a grid resource that can meet the expectations and determine whether or not it is feasible to migrate the job in question to a more suitable resource. The overall computational grid resource statistics are compiled in the ACDC-Grid database and can be queried by grid users and administrators in order to better understand the "state of the grid".

Data Grid. The ACDC-Grid enables the transparent migration of data between various storage element resources while preserving uniform access for the user, where basic file management functions are provided via a platform-independent Web interface, as shown in Figure 1.

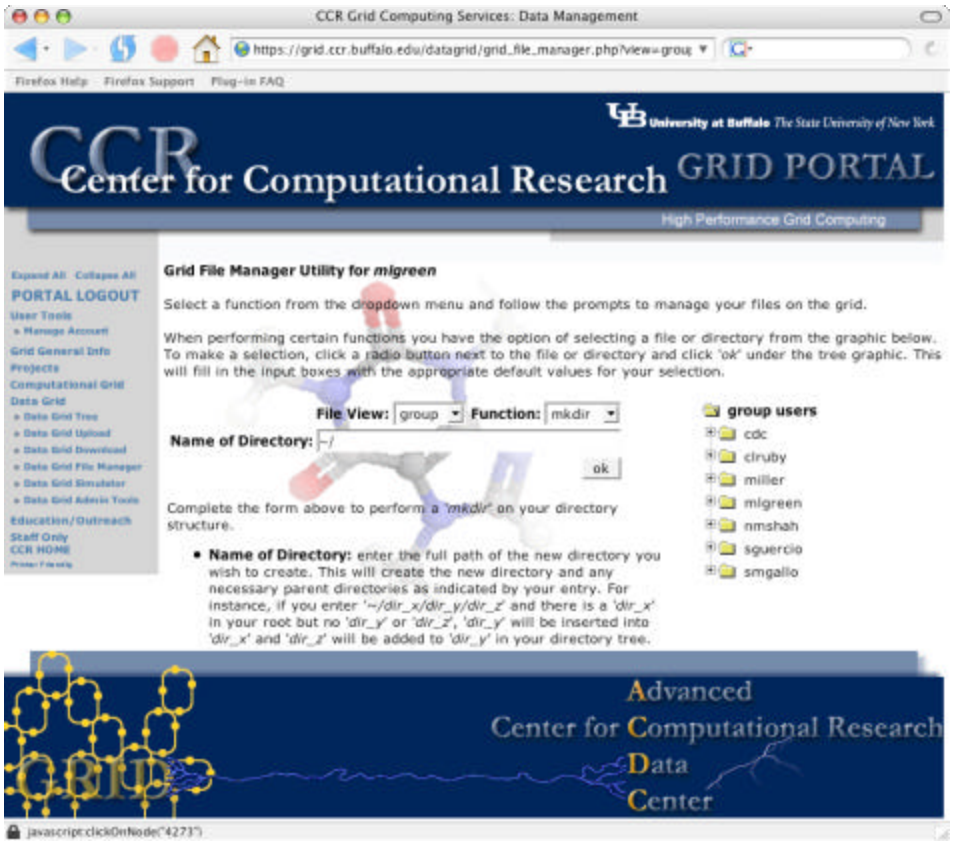


Figure 1. ACDC Data Grid File Manager Web user interface.

We have identified five use cases for the Data Grid file management infrastructure. The infrastructure architecture description for a general View, Edit, Copy, Upload, and Download use case are presented in Figures 2a-b, 2c, 2d-e, respectfully.

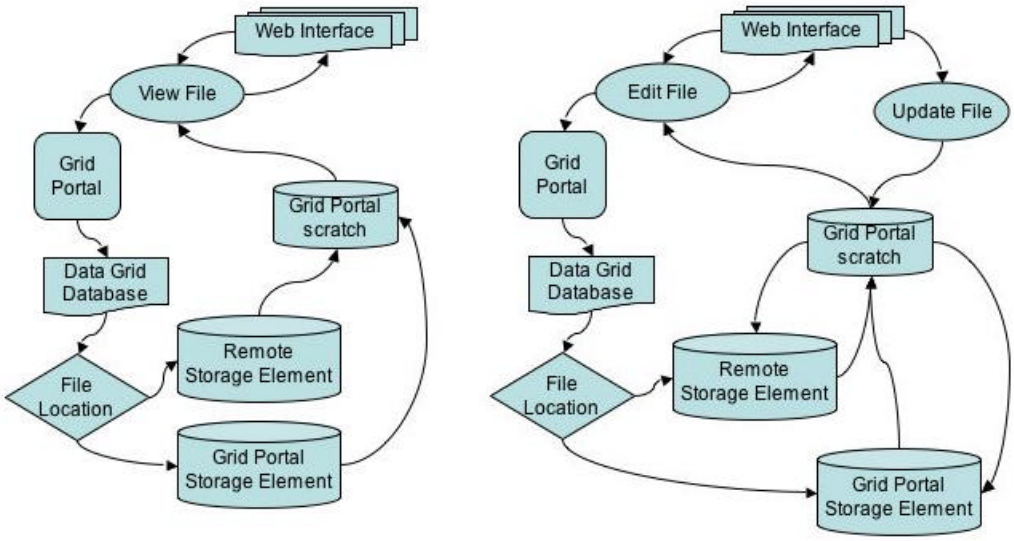


Figure 2a (left) and 2b (right). ACDC Data Grid View and Edit file use case description.

The View use case copies a designated file from the users', group, or public accessible Data Grid to a temporary Grid Portal scratch space and presents the file contents through the Web interface. The temporary copy of the file stored in the Grid Portal scratch space is removed after viewing is complete. The Edit use case copies the from the users, group, or public accessible Data Grid to a temporary Grid Portal scratch space and presents the file for editing through the Web interface. After successfully editing the file, the original file is overwritten by the edited file.

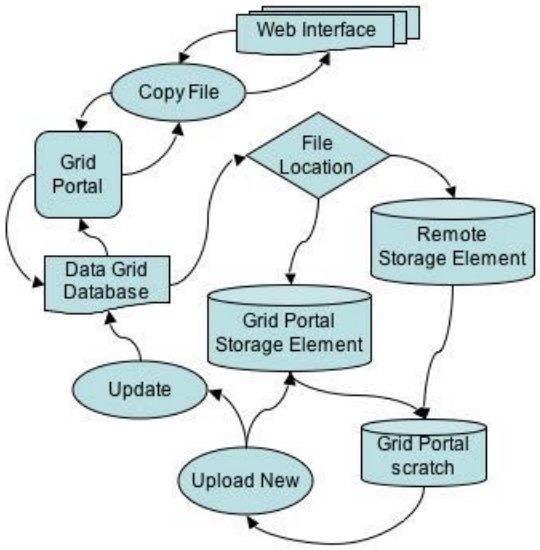


Figure 2c. ACDC Data Grid Copy file use case description.

The Copy use case copies a designated file from the users, group, or public accessible Data Grid to a temporary Grid Portal scratch space and uploads the new file into the Grid Portal Storage Element. The new file attributes are updated in the ACDC Data Grid database after a successful upload into the Storage Element.

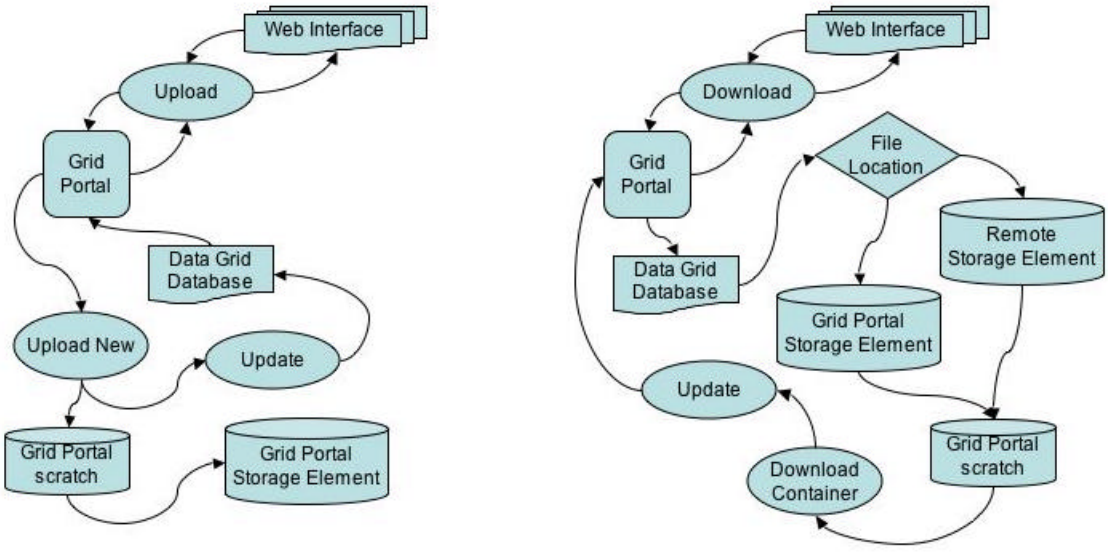


Figure 2d (left) and 2e (right). ACDC Data Grid Upload and Download file use case.

The Upload use case uploads files via the Web interface into a Grid Portal scratch space, applying the directory and file abstractions, and copying the files to the Grid Portal Storage Element. The ACDC Data Grid database is updated with the new directory and file attributes upon successful upload, and the Grid Portal scratch files are deleted. The Download use case assembles the requested files by querying the ACDC Data Grid database for individual file and directory locations into the Grid Portal scratch space. A download container is assembled from the abstracted directory and file attributes obtained from the database and compressed for download. The compressed container is then downloaded to the user through the Web interface.

The gathering of statistical information and the display of such information through a common Web interface are of particular use to developers and administrators. The metadata information and the corresponding data repository for each file are maintained in a global MySQL database table. Algorithms have been implemented to periodically migrate files between repositories in order to optimize usage of resources based on the users' utilization profile. This leads to localization of data files for the computational resources that require them. Conversely, the Chimera Virtual Data System (VDS), which combines a virtual data catalog for representing data derivation procedures and derived data, is used by GriPhyN high-energy physics collaborators [56]. We plan to integrate the Chimera system into the general-purpose ACDC-Grid infrastructure with distributed "Data Grid" services in order to enable on-demand execution of computation schedules constructed from database queries. In addition, this system will provide a catalog that can be used by application environments to describe a set of application programs, and then track all the data files produced by executing those applications.

Storage Resource Managers (SRMs) [57] are middleware components that provide dynamic space allocation and file management on shared storage components of a grid

[58]. SRMs support protocol negotiation and a reliable replication mechanism. The SRM specification standardizes the interface, thus allowing for a uniform access to heterogeneous storage elements [59-62]. The SRM standard allows independent institutions to implement their own SRMs. SRMs provide a flexible policy decision specification process that can be made independently by each implementation for all grid-enabled resources. Furthermore, the tight integration of the computational grid predictive scheduler with the data grid network bandwidth availability statistics is essential for scheduling data migrations for computational jobs. The ACDC-Grid incorporates the Network Weather Service [63] bandwidth and latency information obtained from the computational and data resources into the predictive scheduler algorithms for job staging and execution requirements. Unfortunately, this information is insufficient for determining network bandwidth availability or forecasting essential network statistics. To address this issue, we have deployed software throughout the existing IP networking infrastructure that can be exploited for the development of network forecasting grid services for the ACDC-Grid. This software utilizes the port level network statistics obtained from switches and routers distributed throughout the SUNY-Buffalo network fabric and builds a database for data mining this valuable information. We propose coupling the network information services, predictive scheduler service, and a data grid migration forecasting services into a tool that will achieve improved network and computational resource utilization.

Dynamic Integration of Resources. The ACDC-Grid introduced the concept of dynamic resource allocation during the GRID3 intensive application period during Supercomputing 2003 and Supercomputing 2004. The amount of computational resources provided to the GRID3 user base was dynamically rolled into and out of production on a daily basis. As a proof of concept, for a two-week period, 400 processors of a 600 processor Pentium4 cluster were rolled out of the local CCR pool of resources and into the GRID3 production pool at 8:00 AM, with the inverse procedure taking place at 8:00 PM. The production jobs running on dynamically shared resources were managed through the advanced reservation capabilities of the queuing system [64], thus requiring no administrator intervention in managing the job start or completion. These resources, unlike a similar concept used in Condor flocking, were queue managed and reconfigured on the fly with enhanced grid node security, nfs mounted filesystems, grid user accounts and passwords, grid-enabled software infrastructure, and so forth, and were ready to accept production jobs without system administrator intervention. We are working to extend this automated ACDC-Grid infrastructure to provide on-demand computational resources from multiple IT domain-managed clusters that can be configured by the respective administrators using a grid service.

Grid Research Collaborations. The ACDC-Grid exploits a grid-enabling template framework that includes a dynamically created HTML grid console for the detailed monitoring of computational grid jobs. Results from previous studies have been used in the design of the Globus-based ACDC-Grid that serves researchers at the Center for Computational Research and the Hauptman-Woodward Medical Research Institute, located in Buffalo, NY. In particular, the extensive framework of HTML, JavaScript, PHP, MySQL, phpMyAdmin, and the Globus Toolkit provide a production-level ACDC-

Grid for scientific applications and data integration as required by the applications community. The rapid expansion of the Grid community has facilitated the ACDC-Grid collaboration with many high quality laboratories and testbeds for developing robust and scalable grid infrastructure. The ACDC-Grid has been hardened using grid research collaboration memberships and participation over the past several years.

Grid3. The ACDC-Grid membership in the international Virtual Data Grid Laboratory (iVDGL) provides access to international heterogeneous computing and storage resources for the purpose of experimentation in grid-enabled data-intensive scientific computing. The ACDC-Grid team participates in the (i) iVDGL iGOC, which is used as the central coordination point for grid technical problem resolution, (ii) grid monitoring technical working group, and (iii) grid troubleshooting working group. The iVDGL and other U.S. Grid projects have sponsored several Data Grid activities, including the Grid3 collaboration that has deployed an international Data Grid with participation from more than 28 sites across the United States (including the ACDC-Grid site) and Korea. This facility is operated by the U.S. Grid projects iVDGL, Grid Physics Network (GriPhyN) and the Particle Physics Data Grid (PPDG), and the U.S. participants in the LHC experiments ATLAS and CMS. The Grid3 collaboration uses the Virtual Data Toolkit (VDT) [65] for providing the Grid cyberinfrastructure for the scientific and computer science applications from a variety of disciplines including physics, astrophysics, biology, and astronomy.

The ACDC-Grid Virtual Organization provides computational resources, expertise, users, applications, and core grid job monitoring services for the Grid3 collaboration. The Grid3 resources are used by 7 different scientific applications, including 3 high-energy physics simulations and 4 data analyses in high-energy physics, structural biology (*Shake-and-Bake*), astrophysics, and astronomy. The ACDC-Grid resources processed over 175,000 computational jobs submitted by all of the scientific applications since October, 2003, accounting for over 25% of the total computational jobs processed by the Grid3 resources. The ACDC-Grid resources continue to process computational jobs and provide critical computational job monitoring for the Grid3 collaboration members (ACDC Job Monitoring for Grid3 is at http://acdc.ccr.buffalo.edu).

The International Virtual Data Grid Laboratory (iVDGL) is a global Data Grid that provides resources for experiments in physics and astronomy [66]. Its computing, storage, and networking resources in the U.S., Europe, Asia, and South America provide a unique computational laboratory that will test and validate Grid technologies at international and global scales. The Grid2003 project [67] was defined and planned by Stakeholder representatives in an effort to align iVDGL project goals with the computational projects associated with the Large Hadron Collider (LHC) experiments. See Figure 3.

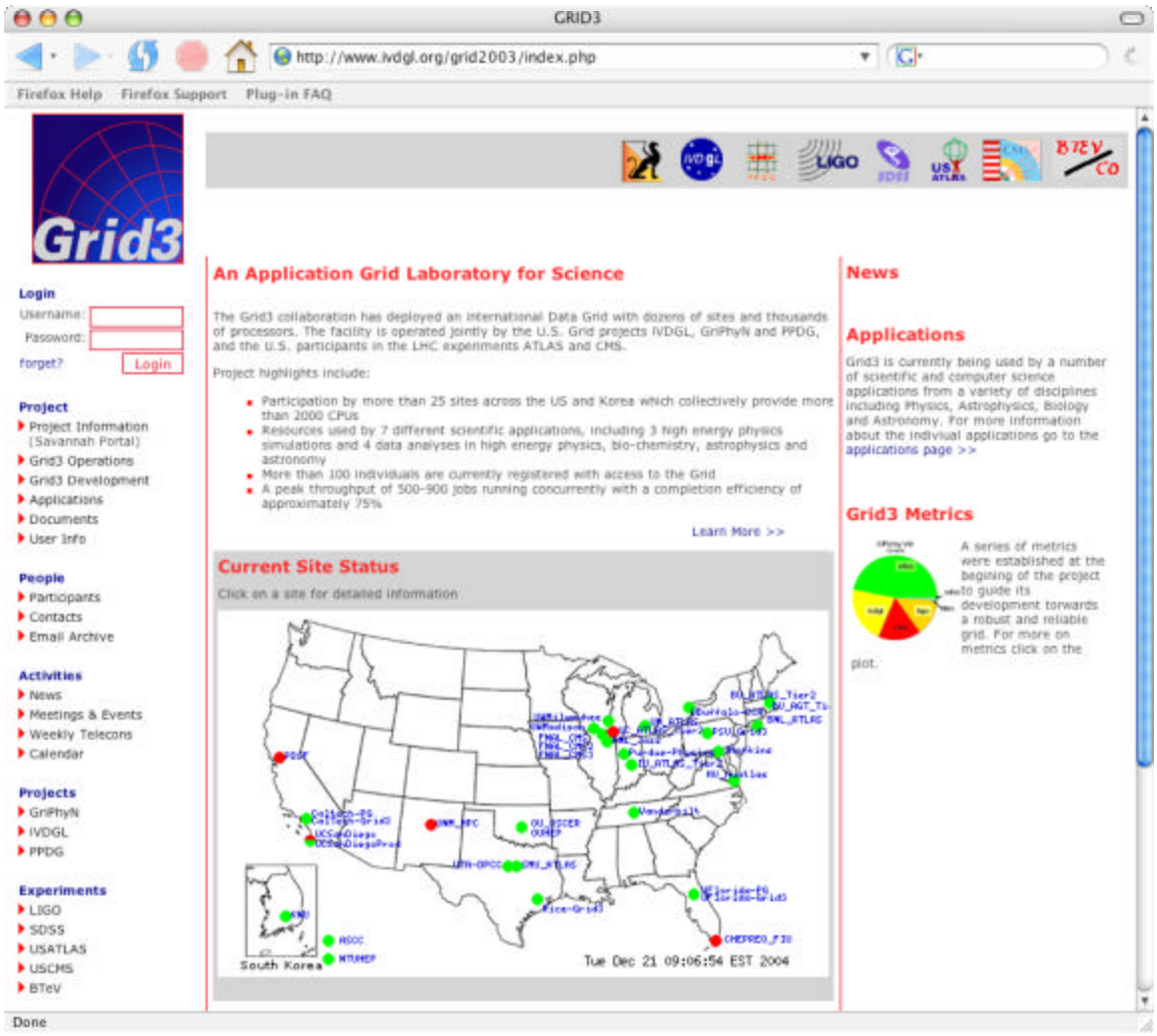


Figure 3. Grid2003 project Web page site catalog and status.

The Grid Laboratory Uniform Environment (GLUE) [68] collaboration was created in Feb. 2002 to provide a focused effort to achieve interoperability between the U.S. physics Grid projects and the European projects. Participant U.S. projects include iVDGL, Grid Physics Network (GriPhyN) [69], and Particle Physics Data Grid (PPDG) [70]. Participant European projects include the European Data Grid (EDG) Project [71], Data Transatlantic Grid (DataTAG) [72], and CrossGrid [73]. Since the initial proposal for the GLUE project, the LHC Computing Grid (LCG) project was created at CERN [74] to coordinate the computing and Grid software requirements for the four LHC experiments, with a goal of developing common solutions. One of the main project goals is deploying and supporting global production Grids for the LHC experiments, which resulted in the Grid2003 "production" grid.

Goals of the Grid2003 Project

The iVDGL Steering Committee set the following broad goals for the Grid2003 project.

- Provide the next phase of the iVDGL Laboratory.

- Provide the infrastructure and services needed to demonstrate LHC production and analysis applications running at scale in a common grid environment.
- Provide a platform for computer science technology demonstrators.

The goals of this project included meeting a set of performance targets, using metrics listed in a planning document. The central project milestone can be summarized as delivery of a shared, multi-Virtual Organization (VO), multi-application, grid laboratory in which performance targets were pursued through deployment and execution of application demonstrations during the period before, during, and after the SC2003 conference in Phoenix (November 16-19). The organization of this project included the creation of teams representing application groups, site administrators, middleware developers, core service providers, and operations. The active period of this project was a 5-month period from July through November 2003. It is interesting to note that subsequent to this period, Grid3 remains largely intact, with many applications running.

The Grid2003 Project deployed, integrated and operated Grid3 with 27 operational processing sites comprising at peak ~2800 CPUs for more than 3 weeks. Progress was made in other areas that are important to the iVDGL mission.

- **Multiple VO grid**. Six different virtual organizations participated and successfully deployed 10 applications. All applications were able to run on sites that were not owned by the host organization. Further, the applications were all able to run on non-dedicated resources.
- **Multi-disciplinary grid**. During the project, two new applications, the *SnB* structural biology application and an application in chemical informatics, were run across Grid3. The fact that these could be installed and run on a Grid infrastructure designed and installed for Particle and Astrophysics Experiments provides the members of iVDGL with confidence that this grid can be adapted to other applications as needed.
- **Use of shared resources**. Many of the resources brought into the Grid3 environment were leveraged facilities in use by other VO's.
- **Dynamic resource allocation**. In addition to resources that were committed 24×7, the Computational Research (CCR) configured their local schedulers to bring additional resources in to and out of Grid3 on a daily basis, satisfying local requirements and Grid3 users.
- **International connectivity**. One site was located abroad (Kyunpook National University, Korea).

Over the course of several weeks surrounding SC2003, the Grid2003 project met its target goals.

1. **Number of CPUS**. With a target of 400 CPUs, Grid2003 successfully incorporated 2163 processors. More than 60% of available CPU resources are non-dedicated facilities. The Grid3 environment effectively shared resources not directly owned by the participating experiments.
2. **Number of Users**. With a target of 10 users, Grid2003 successfully supported 102 users. About 10% of the users are application administrators who do the majority of

the job submissions. However, more than 102 users are authorized to use the resources through their respective VO'S services.

3. **Number of Applications**. With a target of at least 4 physics applications, Grid2003 successfully supported 10 applications, including at least one from each of the five GriPhyN-iVDGL-PPDG participating experiments, the *SnB* program from structural biology, and GADU/Gnare genome analysis. Note that these applications continue to run on Grid3.
4. **Number of sites running Concurrent Applications**. With a target of at least 10 concurrent applications, Grid2003 supported 17 concurrent applications. This number is related to the number of Computational Service sites defined on the catalog page and varies with the application.
5. **Data Transfers Per Day**. With a target of 2-3 TB of data transfer daily, Grid2003 achieved a 4 TB/day transfer rate. This metric was met with the aid of the GridFTP-demo.
6. **Percentage of Resources Used**. With a target of utilizing 90% of the resources, Grid2003 was only able to achieve 40-70% of the resources.
7. **Peak Number of Concurrent Jobs**. With a peak target of 1000 concurrent jobs, Grid2003 was able to support 1100 concurrent jobs. On November 20, 2003 there were sustained periods when over 1100 jobs ran simultaneously.

IBM NE BioGrid. The IBM Northeast Bio-Grid (IBM NE BioGrid) collaboration includes IBM, MIT, Harvard, and the ACDC-Grid. It uses the IBM Grid Toolbox V3 [75] that delivers a set of grid services built with Open Grid Services Architecture (OGSA). OGSA enables the communication across heterogeneous, geographically dispersed environments in addition the IBM General Purpose File System (GPFS) [76] and provides a parallel scalable global filesystem that is used for the ACDC-Grid computational resources. This 4.3TB single filesystem uses 34 servers with 2 hard drives connected by Myrinet and has provided grid-enabled I/O intensive scientific applications bandwidth in excess of 1,800 MB/sec. The IBM NE BioGrid and the Grid3 collaboration use very different Grid cyberinfrastructure middleware for grid-enabled resource communication and computational job executions.

HP GridLite. The ACDC-Grid collaboration with HP on GridLite provides another Grid cyberinfrastructure that we believe will provide core infrastructure for the SUNY-Buffalo ACDC-Campus-Grid that is currently under construction. GridLite will provide a lightweight infrastructure that can easily be deployed on pocketPCs, laptops, PDAs, cellular phones, and other portable devices on the campus. Many of these devices are also being grid-enabled by our NEESgrid [77] collaborators, SUNY-Buffalo's Structural Engineering and Earthquake Simulation Laboratory (SEESL) [78], which is the flagship laboratory in the Multidisciplinary Center for Earthquake Engineering Research (MCEER) [79], which will be an important node of a nationwide "collaboratory" in the NSF's Network for Earthquake Engineering Simulation (NEES). The NEESgrid software release 3.0 [80] point of presence, tele-presence system, and software integration provide the ACDC-Grid with grid services and testbeds for hardening the core grid-enabling instrument and device cyberinfrastructure. The *MCEER, NEES, CSEE, and CCR Collaboration Platform* tightly integrates all of the CCR's common interests and

missions. CCR provides the machine room space to house and maintain a high-powered dual processor server capable of 1) serving a custom Web site with a Gigabit Ethernet connection to the University backbone, 2) serving a Web accessible MySQL database, 3) serving 3D stereo graphics to the SGI 3300W Visualization Display, 4) serving 2D and 3D graphics to the Tiled-Display Wall, 5) serving streaming video to the Access Grid for world-wide presentation, 6) staging and post-processing platform for CCR's Computational Grid (ACDC-Grid) analysis and results, 7) providing a common platform for exchange of information and visualization, and 8) fostering collaborations with other University departments.

Open Science Grid. The ACDC-Grid is a founding participant of the Open Science Grid (OSG), a cooperative venture that brings together laboratory and university facilities, grid technology providers, and the application communities, for the purpose of engineering and building a common Grid infrastructure that will ensure the necessary robust, persistent, computational, and data services needed by laboratories, experiments, and application managers. The OSG provides a framework for coordinating activities with the goal of enabling a common grid infrastructure and shared resources for the benefit of scientific applications. The ACDC-Grid team participate in the a) OSG Security Incident Handling Activity, b) OSG Storage Services Activity, c) OSG-0 Activity, and d) the OSG Blueprint Activity [81]. In accordance with the magnified risk and the circumscribed communities, the Security Incident Handling activity group (SIHag) was established with the goal to reduce risk through the establishment of guidelines, policies, and methods for security incident handling within the OSG and iVDGL communities.

Grid Research Advancements. Several distributed monitoring systems have been designed to track the status of large networked systems. Some of these systems are centralized, where the data is collected and analyzed at a single central host, while others use a distributed storage and query model. Ganglia [82-83] uses a hierarchical system, where the attributes are replicated within clusters using multicast and then cluster aggregates are further aggregated along a single tree. Sophia [84-85] is a distributed monitoring system, currently deployed on Planet-Lab [86], and is based on a declarative logic-programming model, where the location of query execution is both explicit in the language and can be calculated during the course of evaluation. TAG [87] collects information from a large number of sensors along a single tree. IBM Tivoli Monitoring [88] also provides the foundation for additional automated Best Practices via Proactive Analysis Components (PACs) for managing business critical hardware and software including middleware, applications, and databases. A number of existing grid information systems such as MDS2 [89-90], GIS [91], R-GMA [92-93], and Hawkeye [94] each provide a core distributed information management system designed to support a range of applications and services such as scheduling, replica selection, service discovery, and system monitoring. All of these systems use a client-server model in which Information Providers collect or generate data and supply this data to Information Services. We currently work with the Globus Global Information Service (GIS) and the Monitoring and Discovery Service (MDS) working groups through our Grid3 and OSG collaborations. The ACDC-Grid Monitoring and Information Services – Core Infrastructure (MIS-CI) deployed on our grid-enabled resources has been developed through this collaboration, as

shown in Figure 4. The MIS-CI is architected to be a lightweight non-intrusive monitoring and information service that can be throttled by the Resource Provider in a dynamic fashion. The MIS-CI is self-monitoring, secure, and hierarchical in design making it extremely scalable with tuneable information time scales.

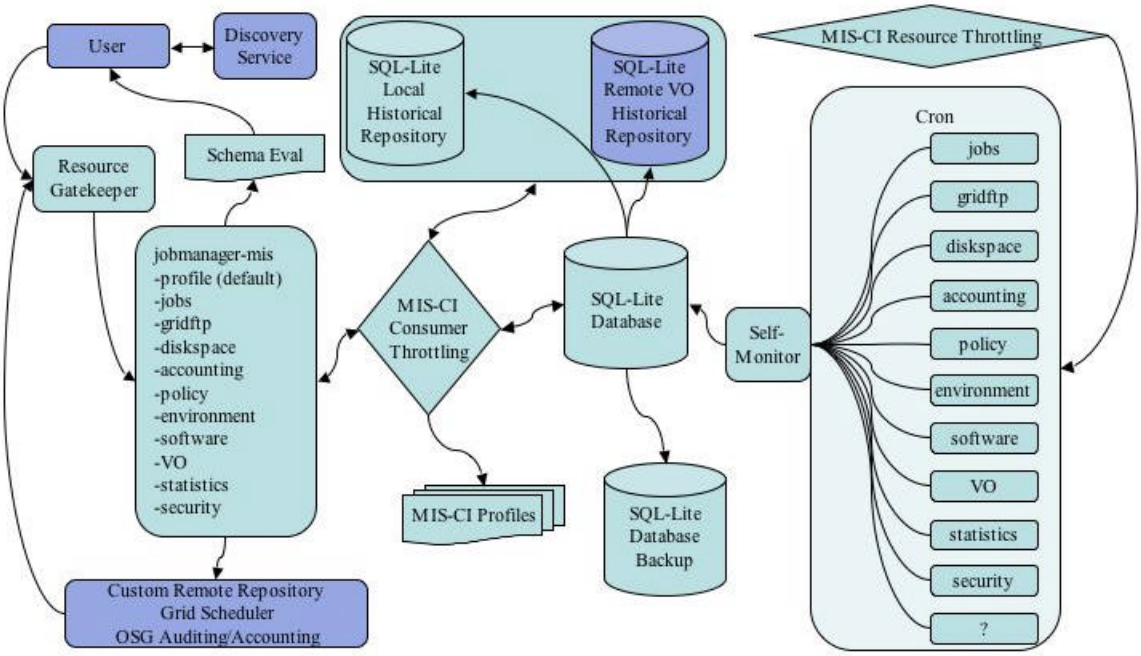


Figure 4. ACDC-Grid Monitoring and Information Services – Core Infrastructure architecture description.

Our monitoring system is being enhanced to

1. provide a programmatic interface to the ACDC Job Monitoring database for running, queued, or historical jobs, complete with the current site status metrics,
2. provide integration with MonaLisa [95] and the Grid3 site status catalogue for defining difference metrics on job monitoring, resource utilization, and policy metrics,
3. provide integration with the Globus MDS provider development team for providing XML formatted job information and site status metrics,
4. provide integration of our predictive scheduling estimates based on resource policy specifications,
5. provide resource specific CPU availability for Grid3 resources, ACDC-Grid resources, and Virtual Organizations,
6. provide currently available free nodes and predictive scheduling capabilities of job execution start times based on running, queued, and submitted job characteristics, including site policy constraints,

7. provide data grid historical and near real-time estimates of bandwidth and utilization of grid-enabled repositories, and
8. harden the secure, lightweight, scalable distributed hierarchical imbedded MySQL database ACDC-Grid monitoring daemon infrastructure for heterogeneous computational grid hardware resources and heterogeneous grid infrastructure middleware.

This enhanced and hardened service can then be utilized by several other open source applications and included in the Globus, NMI [96-97], and VDT software suites for production grid monitoring efforts.

Grid Research Application Abstractions and Tools. The Grid-enabling Application Templates (GATs) used for porting scientific and engineering applications to the ACDC-Grid use abstraction as the process of combining multiple smaller operations into a single unit that can be referred to by a stage. Each stage is named and may contain a template for imposing fine-grained application input file generation, automated parallelization, intermediate result file monitoring, exception handling, and overall application performance metrics. Using the ACDC-Grid GAT abstraction allows programmers to solve problems at a high level, while deferring non-critical details. This has proved to be an effective problem solving strategy in porting codes from structural biology, earthquake engineering, and the environmental and hydrodynamic domains to the ACDC-Grid infrastructure. The application developers have the ability to drill down into each stage or split stages into logical units for their specific application. For example the *Shake-and-Bake* application uses seven stages in defining a computational and data grid job: 1) Software, 2) Template, 3) General Information, 4) Data Preparation, 5) Job Definition, 6) Review, and 7) Execution Scenario. This GAT defines the grid-enabled software application; required and/or optional data files from the ACDC Data Grid; computational requirements are input or a template defined computational requirement runtime estimate is selected; application specific runtime parameters or default template parameter definitions are used; the grid user accepts the template complete job definition workflow or corrects any part of job definition; and the grid user has the ability to input an execution scenario or select a ACDC-Grid determined template defined execution scenario. After these stages have been completed the grid user can view specific grid job completion status, grid job current state, detailed information on all running or queued grid jobs, grid-enabled application specific intermediate and post processing grid job graphics, as well as plots and tables. Figure 5 describes a typical ACDC-Grid GATs workflow definition and execution. The current GAT workflow is robust enough to handle quite complicated definitions that integrate intermediate job status and statistics on a dynamic basis. The GAT API is used extensively for integration the ACDC Computational Grid with the ACDC Data Grid in a seamless fashion.

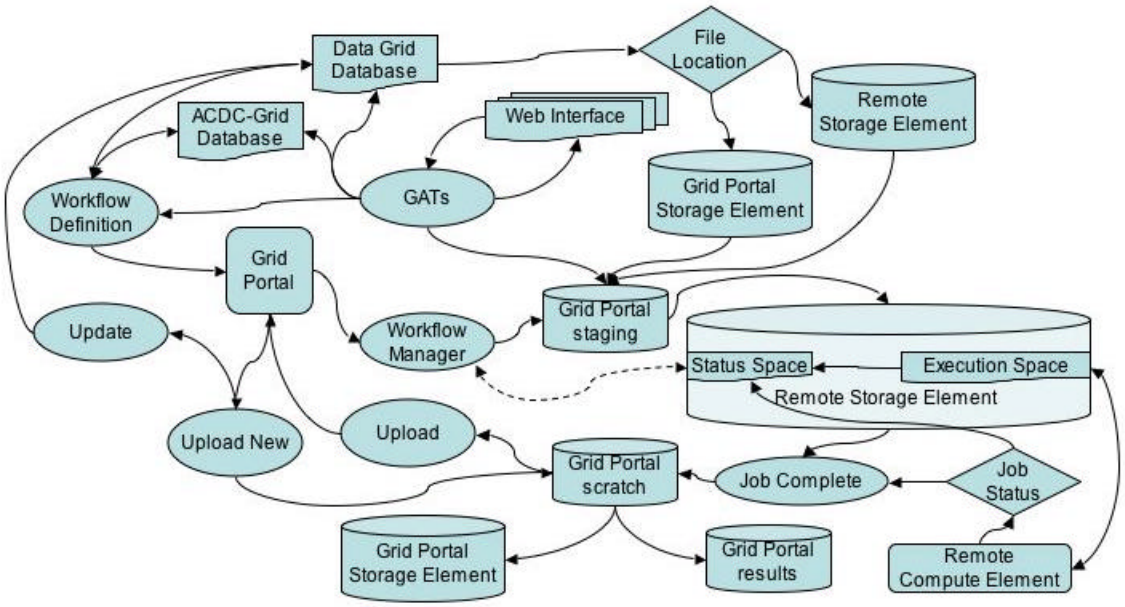


Figure 5. ACDC-Grid Grid-enabling Application Template definition and execution description.

Leveraging our experience with Grid3 and OSG, it is evident that the current Grid security infrastructure is deficient. Specifically, many Grids use a grid-mapfile for mapping remote users to a single local grid user account. This can lead to several potential security problems. We are currently developing infrastructure to mitigate these problems, as shown in Figure 6.

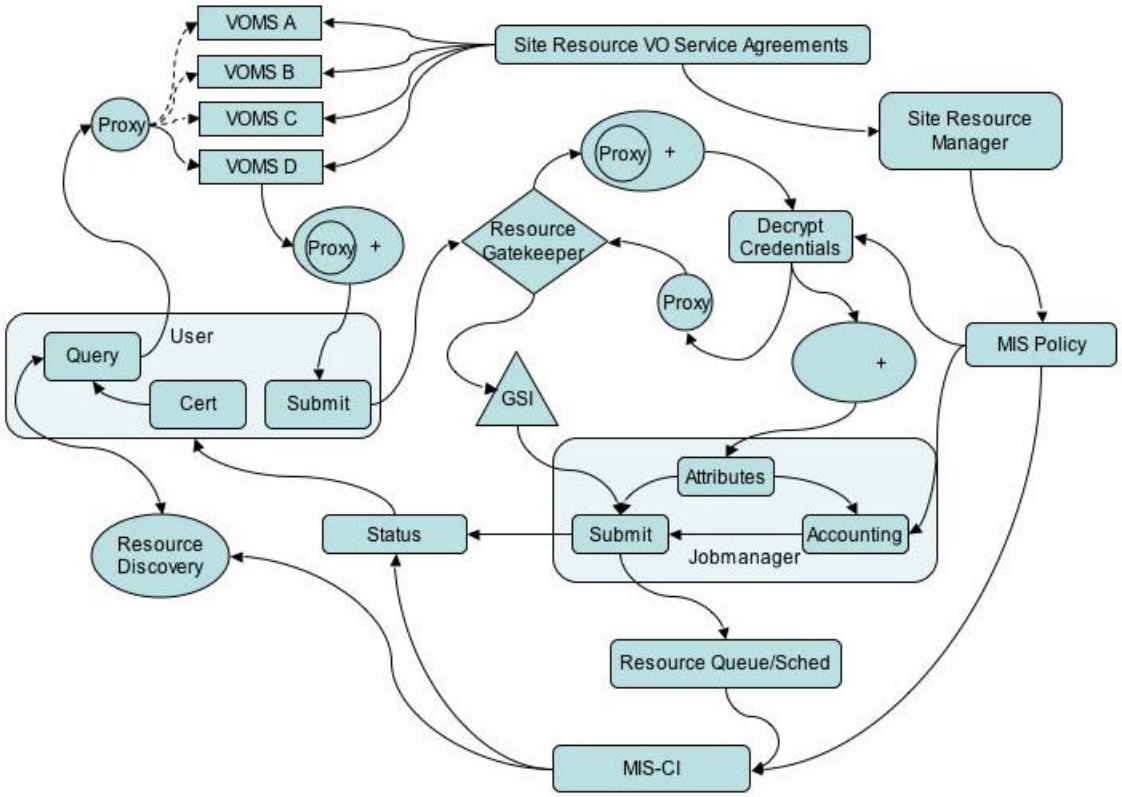


Figure 6. ACDC-Grid Proxy+ enhanced grid security infrastructure.

Optimizing *SnB* on Grids. Genetic Algorithms (GAs) were developed by Holland [98] and are based on natural selection and population genetics. Traditional optimization methods focus on developing a solution from a single trial, whereas genetic algorithms operate with a *population* of candidate solutions. We have constructed a GA to determine an efficient set of *SnB* input parameters in an effort to reduce the time-to-solution for determining a molecular crystal structure from X-ray diffraction data. We use a *population* of candidate *SnB* input parameters. Each member of the population is represented as a string in the population and a fitness function is used to assign a fitness (quality) value for each member. The members in the population obtain their fitness values by executing the *SnB* program with the input parameter values represented by their strings. Using "survival-of-the-fittest" selection, strings from the *old* population are used to create a *new* population based on their fitness values. The member strings selected can recombine using crossover and/or mutation operators. A crossover operator creates a new member by exchanging substrings between two candidate members, whereas a mutation operator randomly modifies a piece of an existing candidate. This procedure of combining and randomly perturbing member strings has, in many cases, been shown to produce stronger (*i.e.*, more fit) populations as a function of time (*i.e.*, number of generations).

We use the Sugal [99] (sequential execution) and PGAPack [100-101] (parallel and sequential execution) genetic algorithm libraries. The Sugal library provided a sequential

GA and has additional capabilities, including a restart function, which proved to be very important when determining fitness values for large molecular structures. The PGAPack library provided a parallel master/slave MPICH/MPI implementation that proved very efficient on distributed- and shared-memory ACDC-Grid compute platforms. Other key features include C and Fortran interfaces, binary-, integer-, real-, and character-valued native data types, object-oriented design, and multiple choices for GA operators and parameters. In addition, PGAPack is quite extensible. The PGAPack library was extended to include restart functionality and is currently the only library used for the ACDC-Grid production work.

The *SnB* computer program has approximately 100 input parameters, though not all parameters can be optimized. For the purpose of this study, 17 critical parameters were identified for participation in the optimization procedure. Eight known molecular structures were initially used to evaluate the genetic algorithm evolutionary molecular structure determination framework performance. These structures are 96016c [102], 96064c [103], crambin [104-105], Gramicidin A [106], isoleucinomycin [107], pr435 [108], Triclinic Lysozyme [109], and Triclinic Vancomycin [110].

In order to efficiently utilize the computational resources of the ACDC-Grid, an accurate estimate must be made in terms of the resource requirements for *SnB* jobs that are necessary for the GA optimization. This includes runs with varying parameter sets over the complete set of eight known structures from our initial database.

This is accomplished as follows. First, a small number of jobs are run in order to determine the required running time for each of the necessary jobs. Typically, this consists of running a single trial for each of the jobs in order to predict the time required for the required number of trials for the job under consideration.

Approximately 25,000 population members were evaluated for the eight known molecular structures and stored in a MySQL database table, as shown in Figure 7.

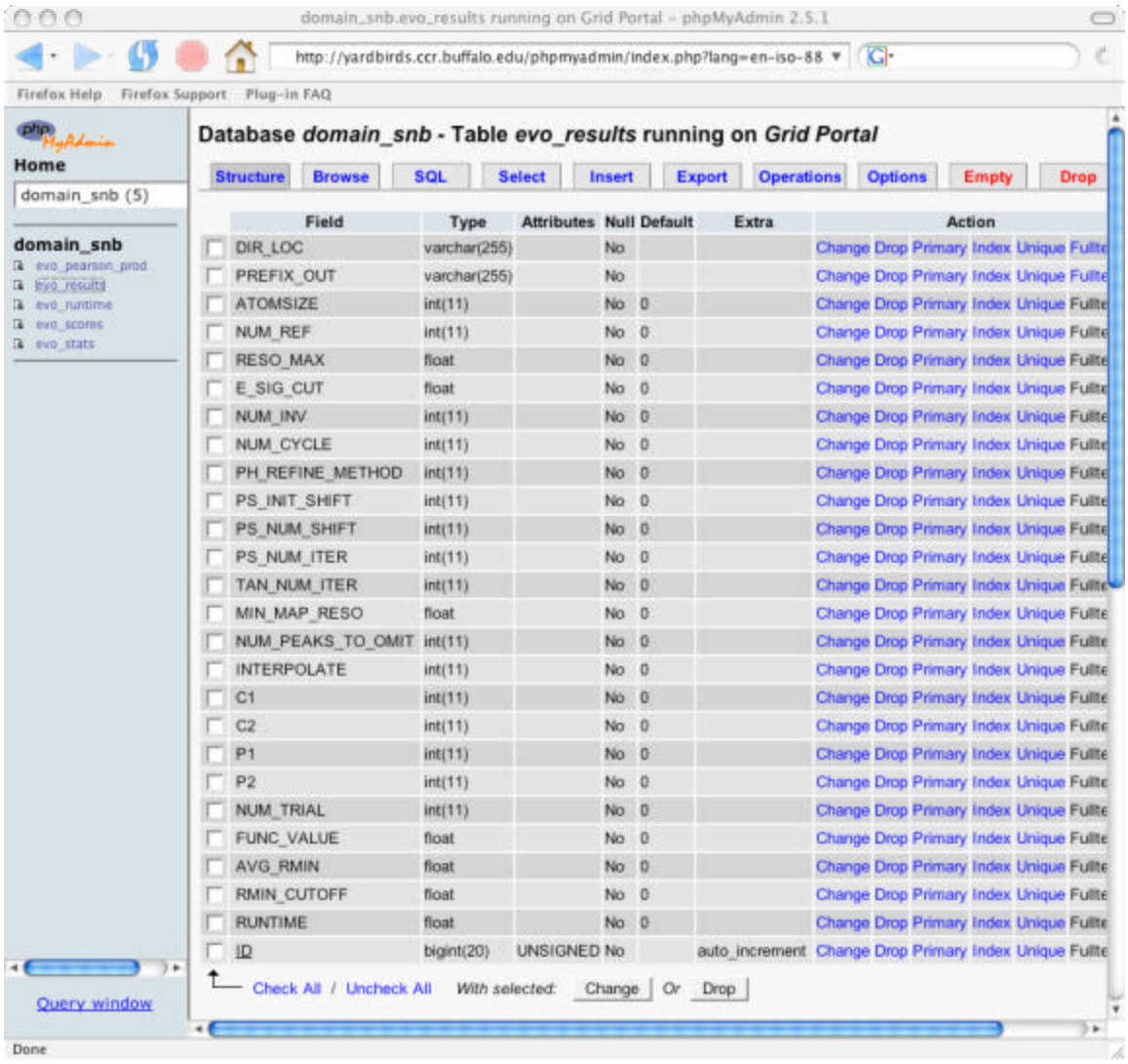


Figure 7. MySQL database table for SnB trial results.

From these trial results, the mean ($\overline{X}^j$) and standard deviations (s^j) are calculated for each input parameter j and used to determine the standard scores (z_i^j) for each trial i,

$$z_i^j = \frac{X_i^j - \overline{X^j}}{s^j},$$

for all i and j, where the trial parameter value for trial i and parameter j is X_i^j. Figure 8 shows the standard scores of the parameters under consideration.

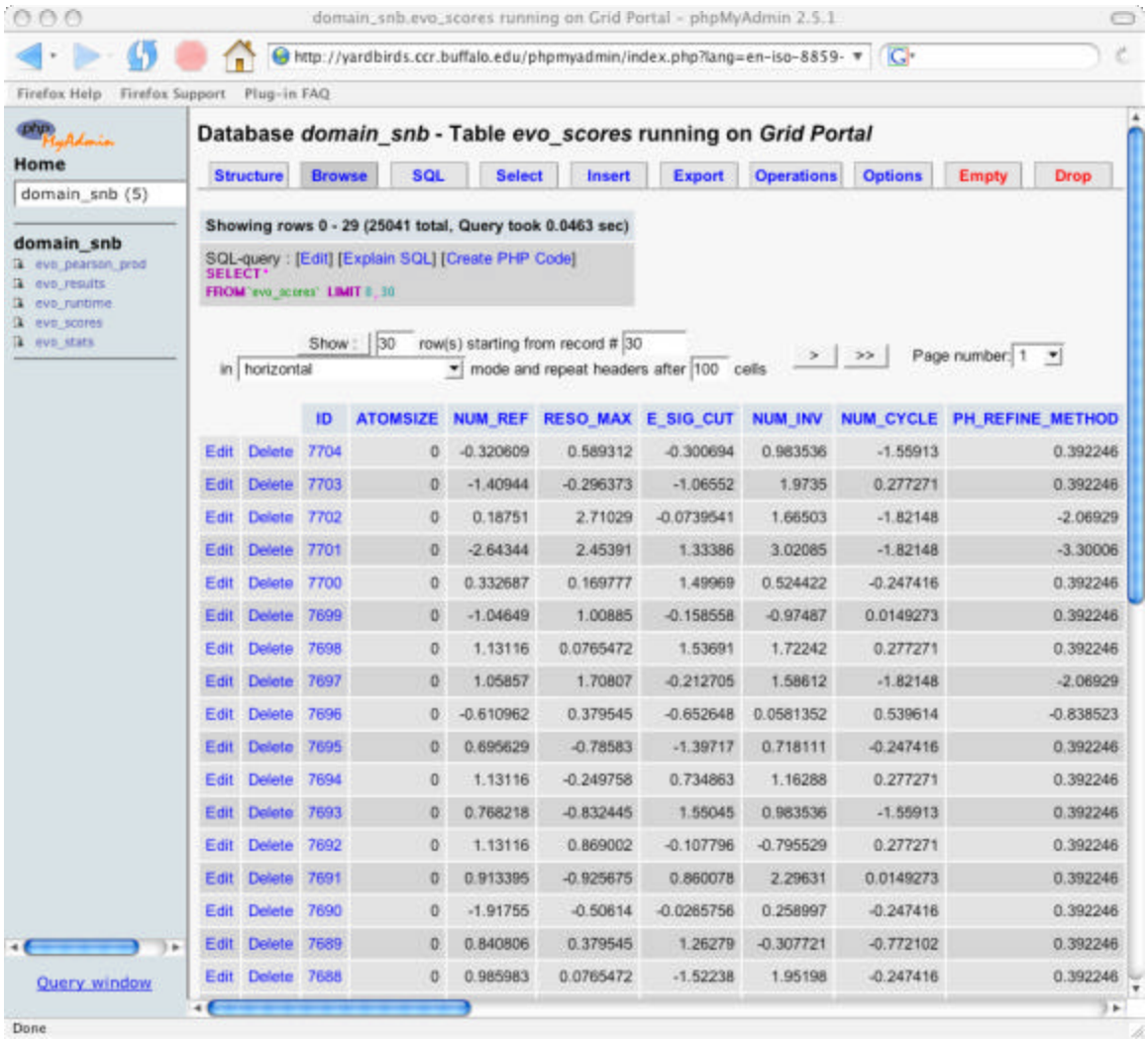


Figure 8. Standard scores for Pearson product-moment correlation coefficient calculations.

The Pearson product-moment correlation coefficients (r_k^j) are calculated for input parameter j and molecular structure k by

$$r_k^j = \frac{\sum z_k^j z_k^{runtime}}{N-1},$$

for all j and k, where N denotes the degrees of freedom and $z_k^{runtime}$ represents the standard score of the GA trial run time. Refer to Figure 9.

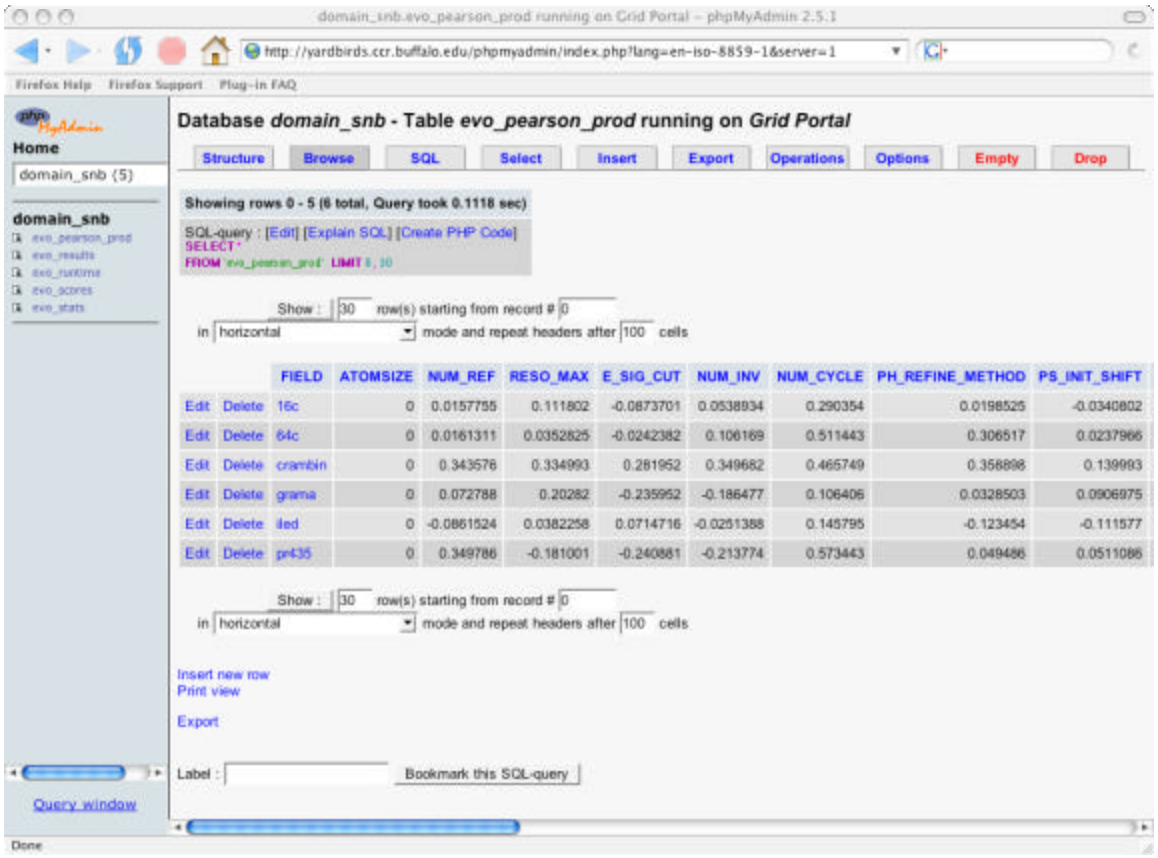


Figure 9. Pearson product-moment correlation coefficient database table.

The input parameters that have the largest absolute magnitude Pearson product-moment correlation coefficient with respect to the observed trial run times are selected and used to form a predictive run time function that is fit using a linear least squares routine

$$X_i^{runtime} = \sum a_j r_k^j \overline{X^j},$$

where the observed $X_i^{runtime}$ trial run time is fit to a selected sub-set of input parameter values j, $\overline{X^j}$ denotes the input parameter value, r_k^j denotes the respective molecular structure k Pearson product-moment correlation coefficient, and a_j denotes the linear least square fit coefficients for each j input parameter. We use this function within the grid-enabled data-mining infrastructure to estimate the maximum number of *SnB* GA generations and the maximum size of the population that would run on a given computational resource within the specified time frame.

The ACDC-Grid infrastructure automatically updates the correlation coefficients based on the availability of new trial data appearing in the *SnB* trial result table. Thus, run time estimates for any given structure continually evolve throughout the GA optimization process.

For example, if there are 50 processors available for 150 minutes on ACDC-Grid compute platform "A", we are interested in determining the maximum number of GA generations and the size of the population that can run on "A" and complete within 150 minutes. Based on this information, the data mining algorithms can make intelligent choices of not only which structures to evaluate, but they can completely define the *SnB* GA job that should be executed. This type of run time prediction is an essential component of our system for providing a level of quality of service. Further, in our experience, this type of run time parameter-based prediction is almost always necessary when queue managed computational resources are employed.

Grid-Enabled Data Mining with *SnB*. The *SnB* grid-enabled data mining application utilizes the ACDC-Grid infrastructure. A typical *SnB* job uses the Grid Portal to supply the molecular structures parameter sets to optimize, the data file metadata, the grid-enabled *SnB* mode of operation (dedicated or back fill), and the *SnB* termination criteria. The Grid Portal then assembles the required *SnB* application data and supporting files, execution scripts, database tables, and submits jobs for parameter optimization based on the current database statistics. ACDC-Grid job management automatically determines the appropriate execution times, number of trials, number of processors for each available resource, as well as logging the status of all concurrently executing resource jobs. In addition, it automatically incorporates the *SnB* trial results into the molecular structure database, and initiates post-processing of the updated database for subsequent job submissions.

The Grid Portal then assembles the required *SnB* application data and supporting files, execution scripts, database tables, and submits jobs for parameter optimization based on the current database statistics. ACDC-Grid job management automatically determines the appropriate execution times, number of trials, number of processors for each available resource, as well as logging and status of all concurrently executing resource jobs. In addition, it automatically incorporates the *SnB* trial results into the molecular structure database, and initiates post-processing of the updated database for subsequent job submissions. Figure 10 shows the logical relationship for the *SnB* grid-enabled data mining routine described.

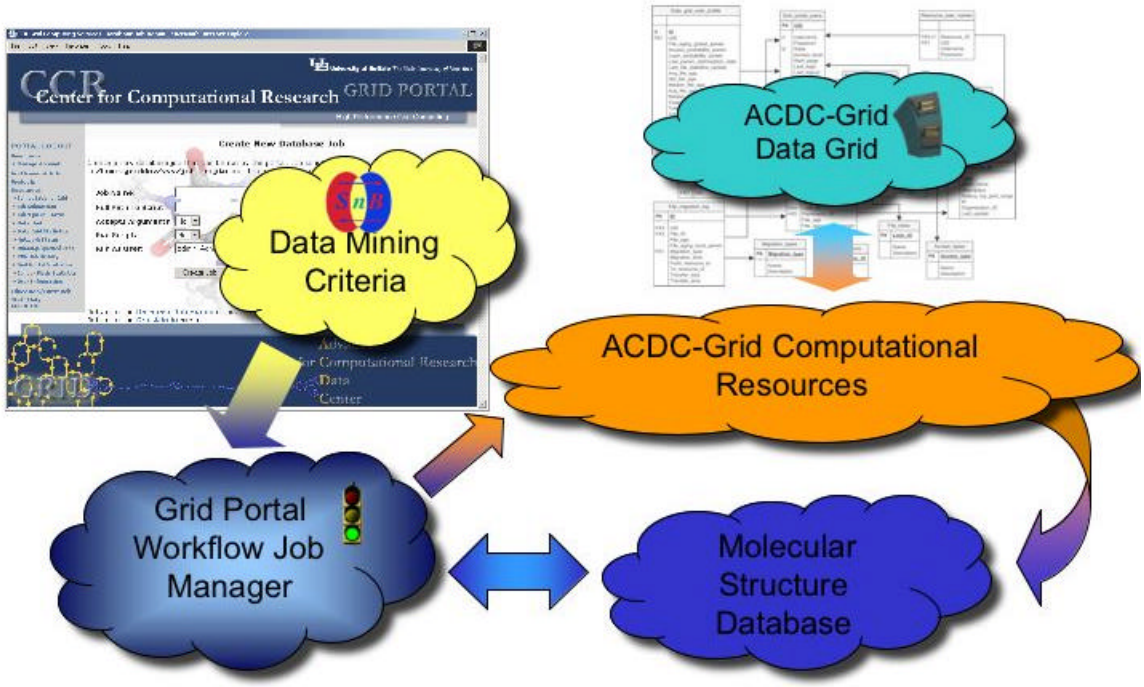


Figure 10. ACDC-Grid grid-enabled data mining diagram.

For example, starting September 8, 2003, a backfill data mining *SnB* job was activated at the Center for Computational Research using the ACDC-Grid computational and data grid resources. The ACDC-Grid historical job-monitoring infrastructure is used to obtain the jobs completed for the period of September 8, 2003 to January 10, 2004, as shown in Figure 11.

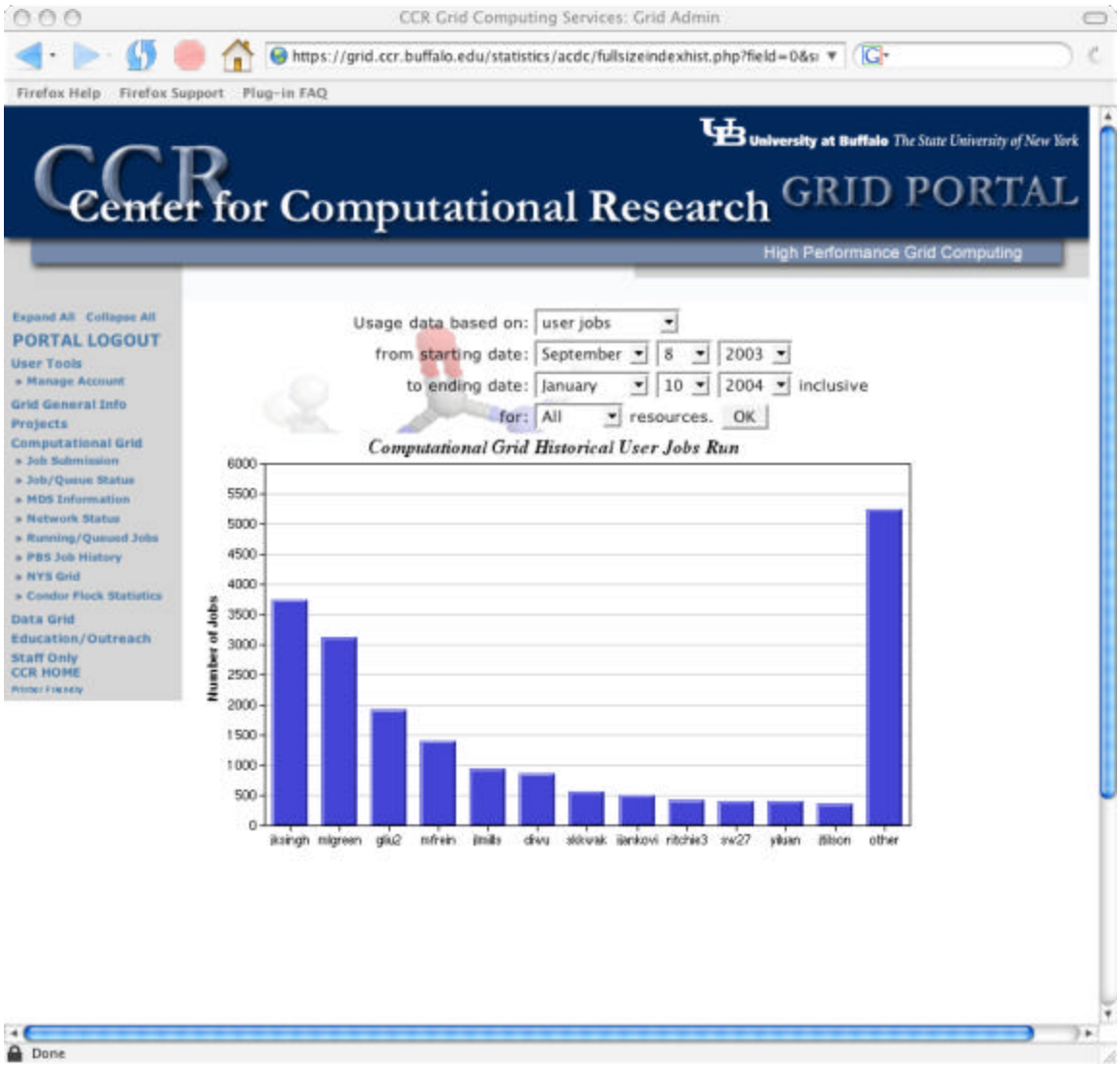


Figure 11. ACDC-Grid job monitoring information for all resources and users.

The activated data mining *SnB* job template is being run by user mlgreen. By hovering over the bar in the chart, as shown in Figure 12, one can see mlgreen's job statistics. Further, notice that 3118 jobs have been completed on the ACDC-Grid resources over this time period. The ACDC-Grid job monitoring also dynamically reports job statistics for the data mining jobs. The total number of jobs completed by all users on all resource is 19,868 where the data mining jobs represent 15.69% of the total. The average number of processes for a data-mining job was 19.65 and the total number of processors used over this period was 433,552, where the data mining jobs accounted for 16.85% of the total. The data mining jobs consumed 291,987 CPU hours, which was 19.54% of the total CPU hours consumed (1,494,352 CPU hours).

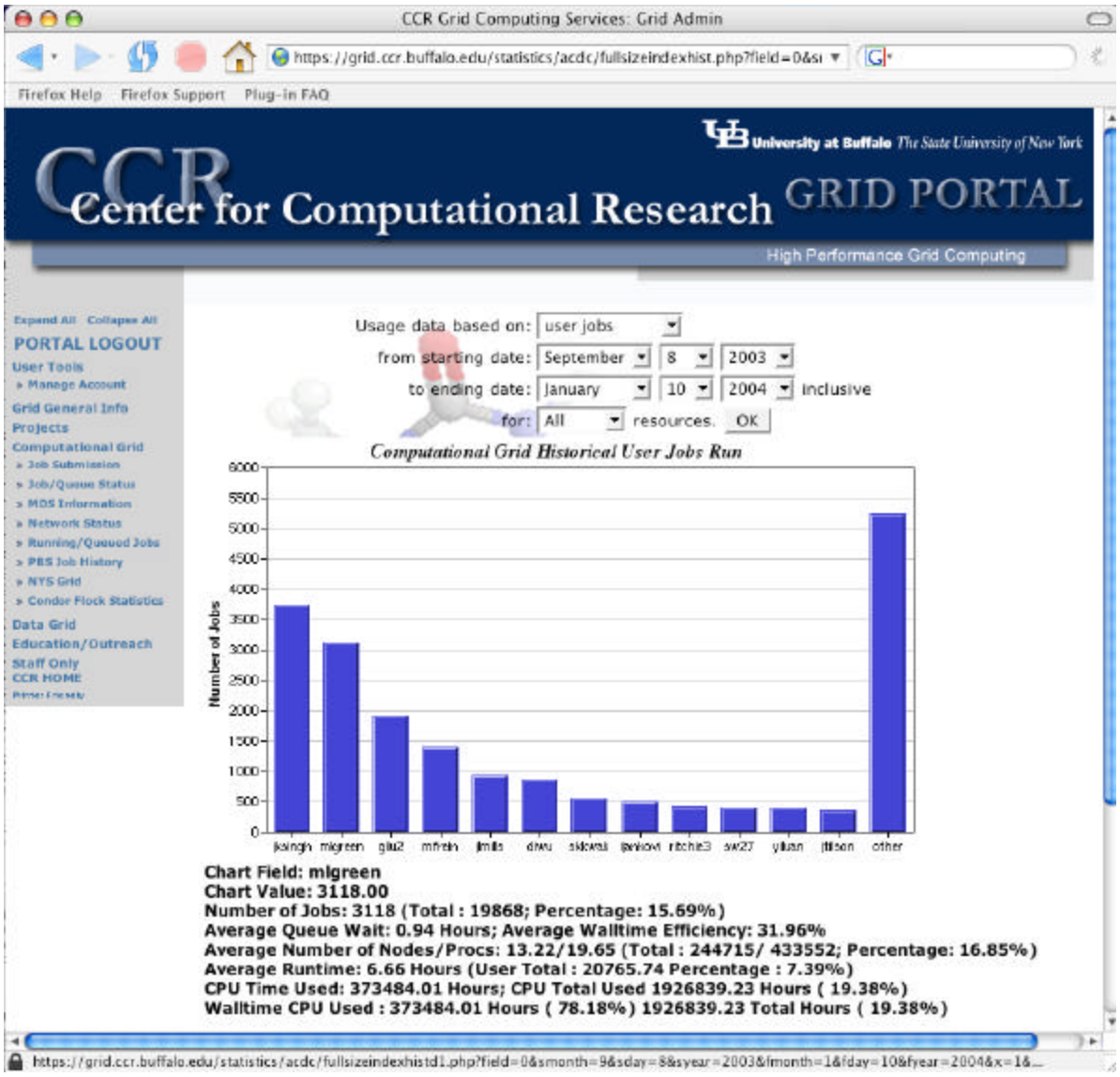


Figure 12. ACDC-Grid job monitoring statistics for user mlgreen.

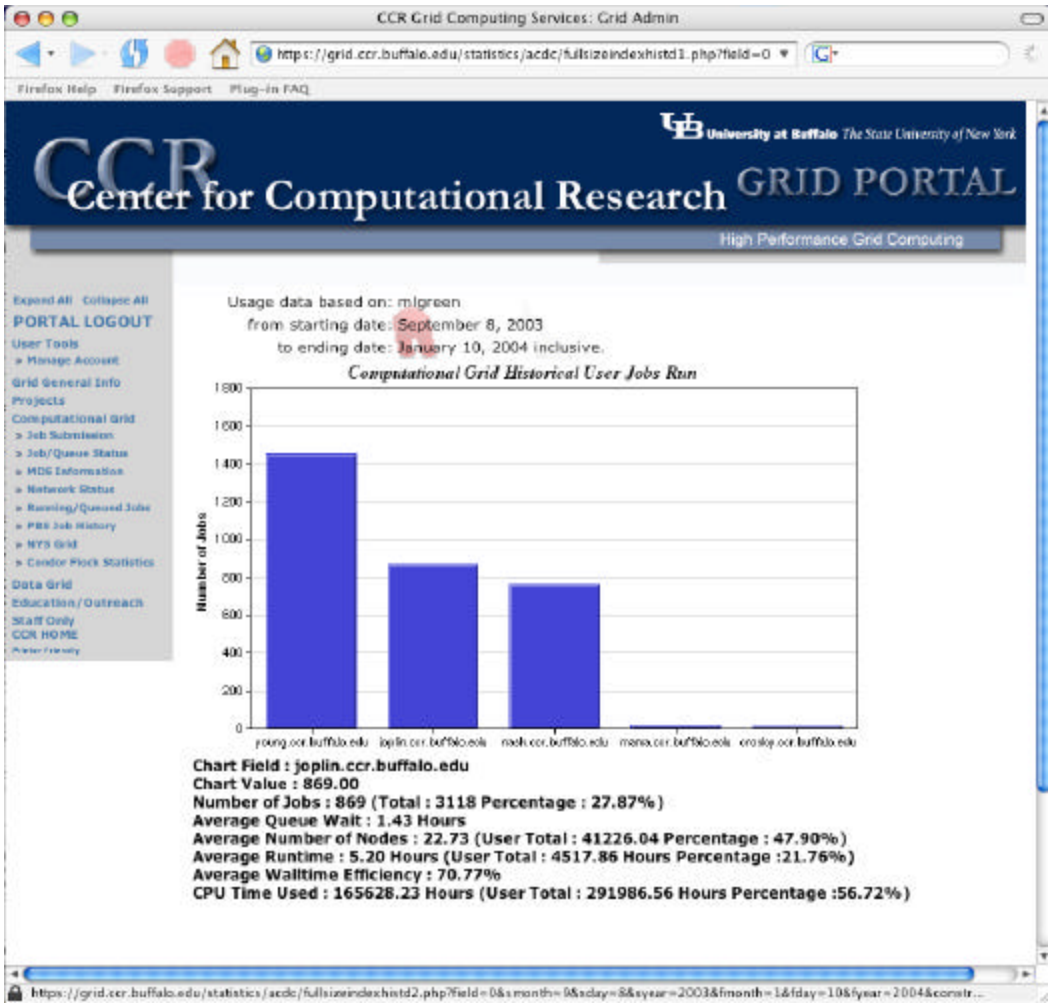
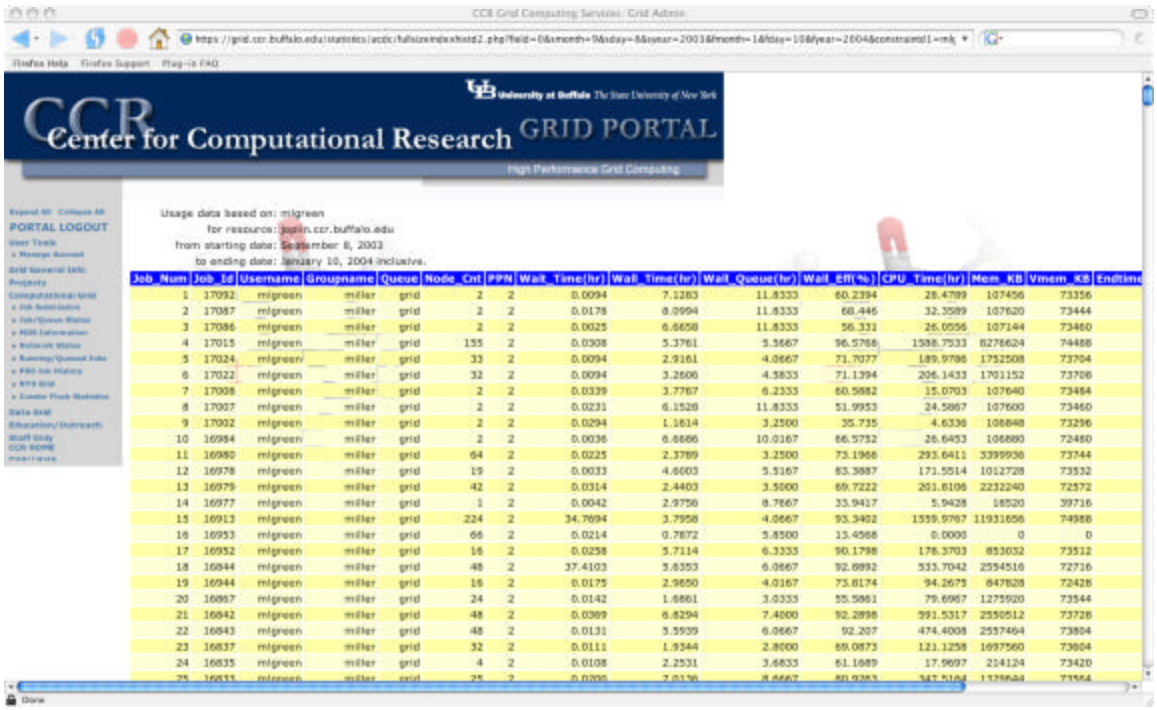


Figure 13. ACDC-Grid job monitoring statistics for user mlgreen.

A subsequent mouse click on the bar chart drills down further describing the jobs completed by user mlgreen. Here, we see five computational resources that processed the 3118 data mining jobs. The statistics for the Joplin compute platform are shown in Figure 13. Note that all statistics are based only on the jobs completed by the mlgreen user. There were 869 jobs processed by the Joplin compute platform representing 27.87% of the 3118 data mining jobs.

Clicking on the bar chart drills down into a full description of all jobs processed by the Joplin compute platform, as shown in Figure 14. The information presented includes job ID, username, group name, queue name, node count, processes per node, queue wait time, wall time used, wall time requested, wall time efficiency, CPU time, physical memory used, virtual memory used, and job completion time/date.



Job_Num	Job_Id	Username	Groupname	Queue	Node_Cnt	PPN	Wall_Time(hr)	Wall_Time(hr)	Wall_Queue(hr)	Wall_Eff(%)	CPU_Time(hr)	Mem_KB	Vmem_KB	Endtime
1	17032	mlgreen	miller	grid	2	2	0.0094	7.1283	11.8333	60.2394	28.4799	107456	73356	
2	17087	mlgreen	miller	grid	2	2	0.0178	8.0994	11.8333	68.446	32.3589	107620	73444	
3	17086	mlgreen	miller	grid	2	2	0.0025	6.6658	11.8333	56.331	26.0556	107144	73460	
4	17015	mlgreen	miller	grid	155	2	0.0308	5.3761	5.3667	96.5768	1388.7533	8276624	74488	
5	17024	mlgreen	miller	grid	33	2	0.0094	2.9161	4.0667	71.7077	189.9786	1752508	73704	
6	17022	mlgreen	miller	grid	32	2	0.0094	3.2606	4.5833	71.1394	206.1433	1701152	73708	
7	17008	mlgreen	miller	grid	2	2	0.0339	3.7767	6.2333	60.5882	15.0703	107640	73484	
8	17007	mlgreen	miller	grid	2	2	0.0231	6.1528	11.8333	51.9953	24.5867	107600	73460	
9	17002	mlgreen	miller	grid	2	2	0.0294	1.1614	3.2500	35.735	4.6336	106848	73296	
10	16984	mlgreen	miller	grid	2	2	0.0036	6.6686	10.0167	66.5752	26.6453	106880	72480	
11	16980	mlgreen	miller	grid	64	2	0.0225	2.3789	3.2500	73.1966	293.6411	3399936	73744	
12	16978	mlgreen	miller	grid	19	2	0.0033	4.6603	5.5167	83.3887	171.5514	1012728	73532	
13	16979	mlgreen	miller	grid	42	2	0.0314	2.4403	3.5000	69.7222	201.6106	2232240	72572	
14	16977	mlgreen	miller	grid	1	2	0.0042	2.9756	8.7667	33.9417	5.9428	18520	39716	
15	16913	mlgreen	miller	grid	224	2	34.7694	3.7958	4.0667	93.3402	1559.9767	11931656	74988	
16	16953	mlgreen	miller	grid	66	2	0.0214	0.7872	5.8500	13.4566	0.0000	0	0	
17	16952	mlgreen	miller	grid	16	2	0.0258	5.7114	6.3333	90.1798	178.3703	853032	73512	
18	16944	mlgreen	miller	grid	48	2	37.4103	5.6353	6.0667	92.8892	533.7042	2554516	72716	
19	16944	mlgreen	miller	grid	16	2	0.0175	2.9650	4.0167	73.8174	94.2675	847828	72428	
20	16867	mlgreen	miller	grid	24	2	0.0142	1.6861	3.0333	55.5861	79.6967	1275920	73544	
21	16842	mlgreen	miller	grid	48	2	0.0369	6.8294	7.4000	92.2898	391.5317	2550512	73728	
22	16843	mlgreen	miller	grid	48	2	0.0131	5.5939	6.0667	92.207	474.4008	2557464	73804	
23	16837	mlgreen	miller	grid	32	2	0.0111	1.9344	2.8000	69.0873	121.1258	1697560	73604	
24	16835	mlgreen	miller	grid	4	2	0.0108	2.2531	3.6833	61.1689	17.9697	214124	73420	

Figure 14. ACDC-Grid job monitoring tabular accounting of completed job statistics.

The ACDC-Grid data mining backfill mode of operation only uses computational resources that are currently not scheduled for use by the native queue scheduler. These resources are commonly called "backfill" as users can run jobs on the associated nodes without affecting the queued jobs. Many queues and schedulers give this information in the X number of nodes available for Y amount of time. The ACDC-Grid infrastructure monitors this information for all of the computational resources and stores this information in a MySQL database table, as shown in Figure 15.

Figure 15 also shows the number of processors and wall time that are available for each resource. Note a value of –1 for the available wall time represents an unlimited amount of time (no currently queued job require the use of these processors). The activated data mining template can obtain the number of processors and wall time available for a given compute platform and then check the status of the platform before determining the actual GA *SnB* data mining job parameters. See Figure 16.

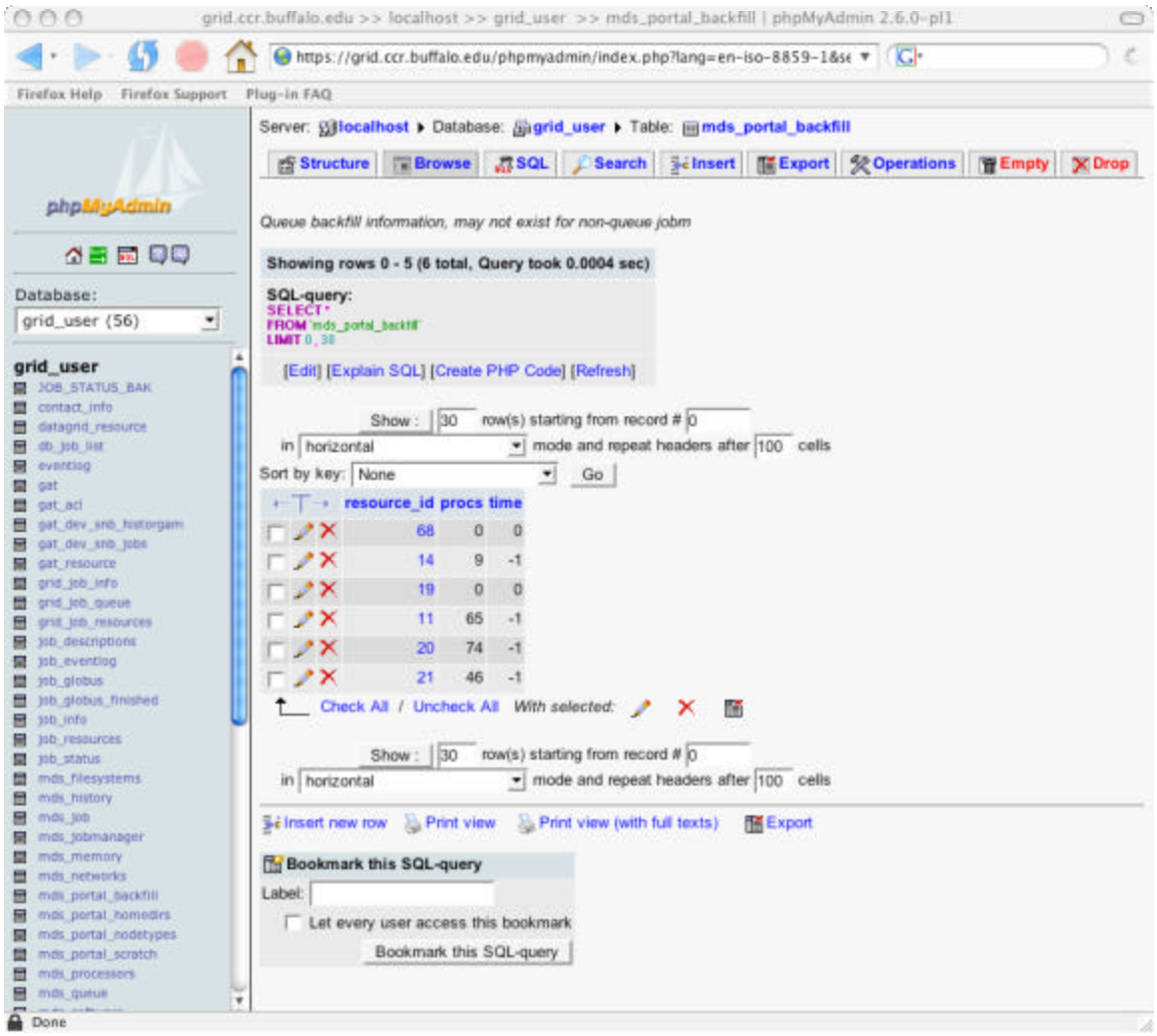


Figure 15. ACDC-Grid backfill information for all resources.

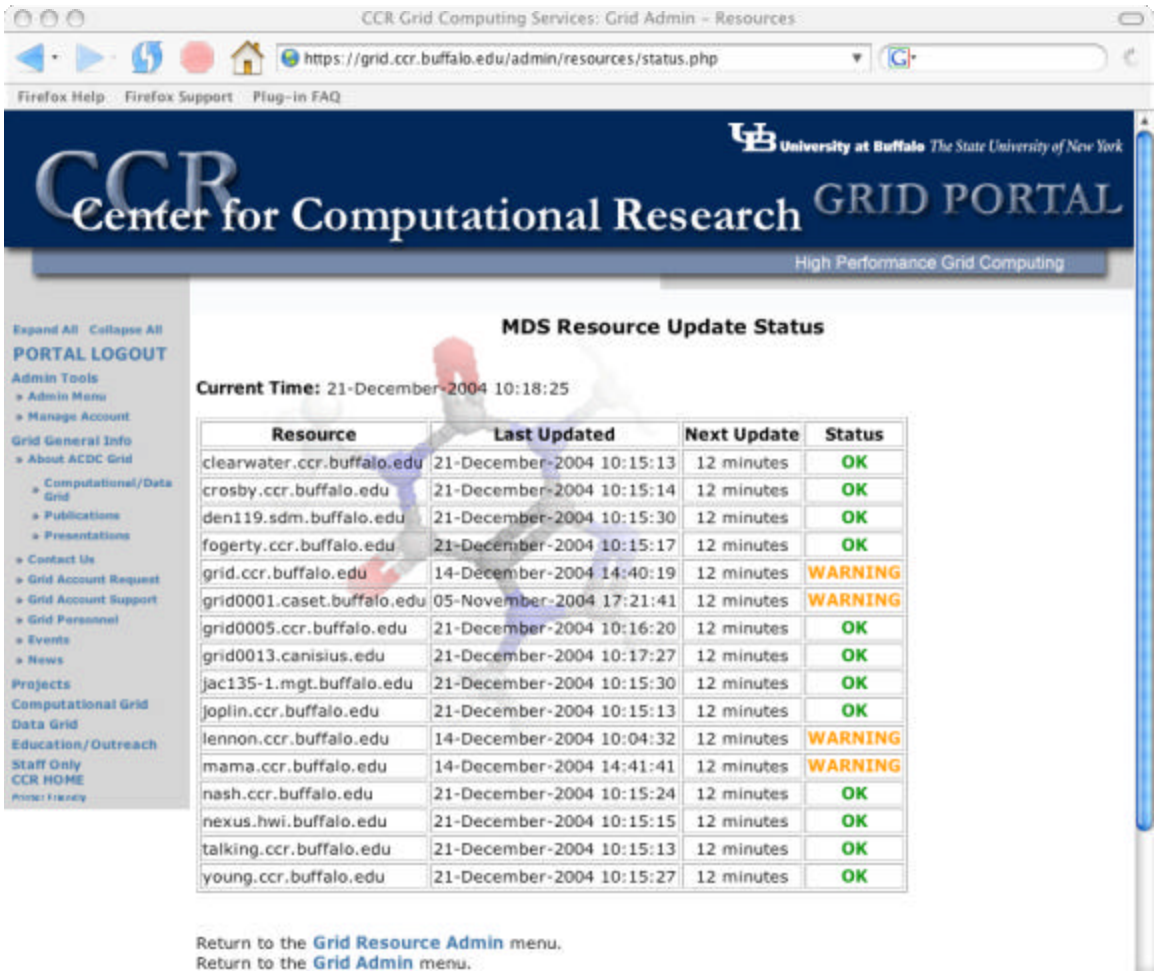


Figure 16. ACDC-Grid computational resource status monitor.

Using the Pearson product-moment fit function derived earlier, the new data mining job run time is estimated based on the current ACDC-Grid *SnB* molecular structure database information. The data mining job template is then executed leading to the migration and submission of the designed data-mining job to the respective ACDC-Grid computational resource.

The activated data-mining template has two options of stopping criteria, as follows.

1. Continue submitting *SnB* data mining application jobs until the optimal parameters have been found based on pre-determined criteria.
2. Continue indefinitely (the data mining template is manually de-activated by the user when optimal parameters are found).

This illustrative example summarizes the evolutionary molecular structure determination optimization of the *Shake-and-Bake* method as instantiated in the *SnB* computer program.

The ACDC data grid complements the ACDC computational grid in terms of managing and manipulating these data collections. As discussed, the goal of the ACDC data grid is to transparently manage data distributed across heterogeneous resources, providing access via a uniform (Web) interface. In addition, we also enable the transparent

migration of data between various resources while preserving uniform access for the user. See Figure 17.

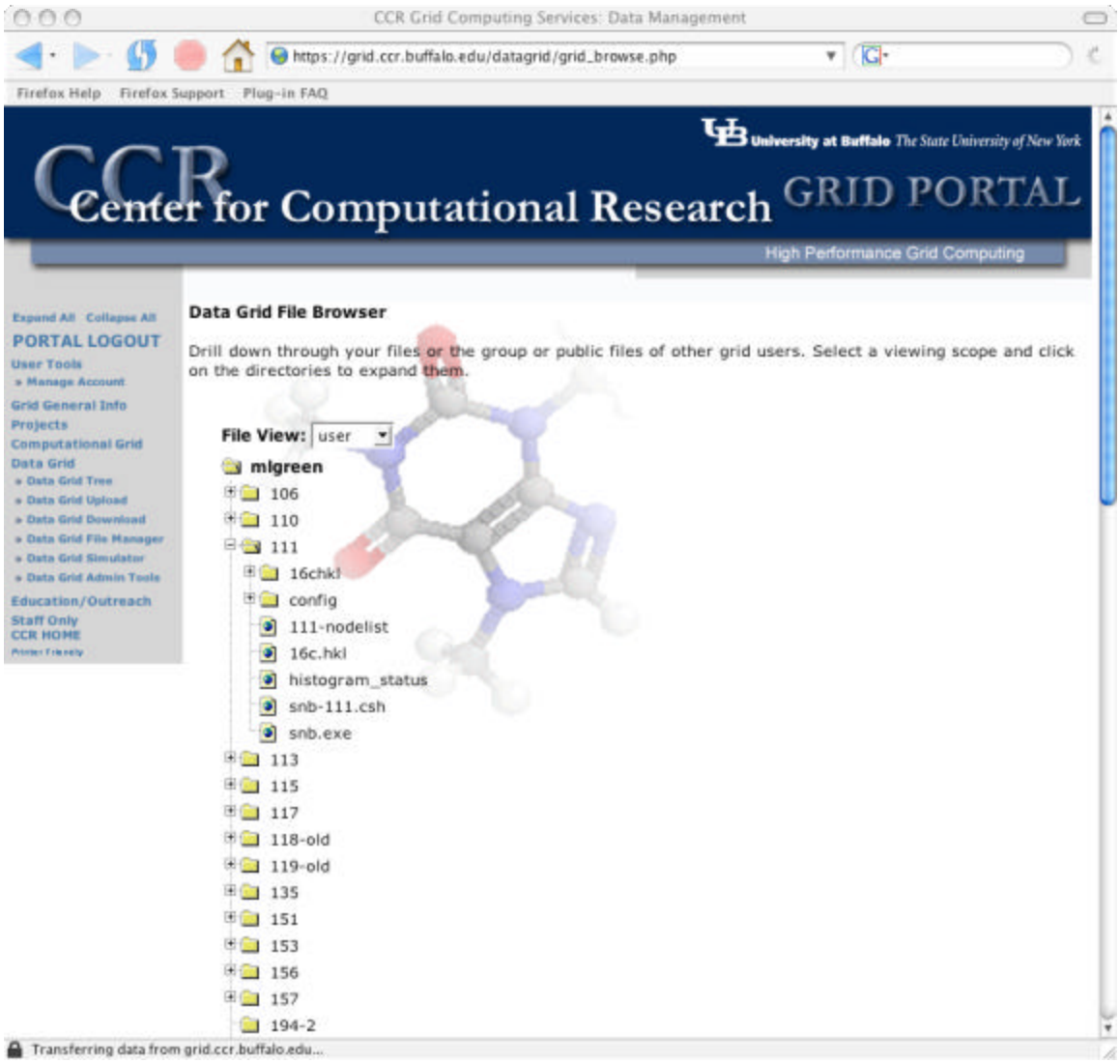
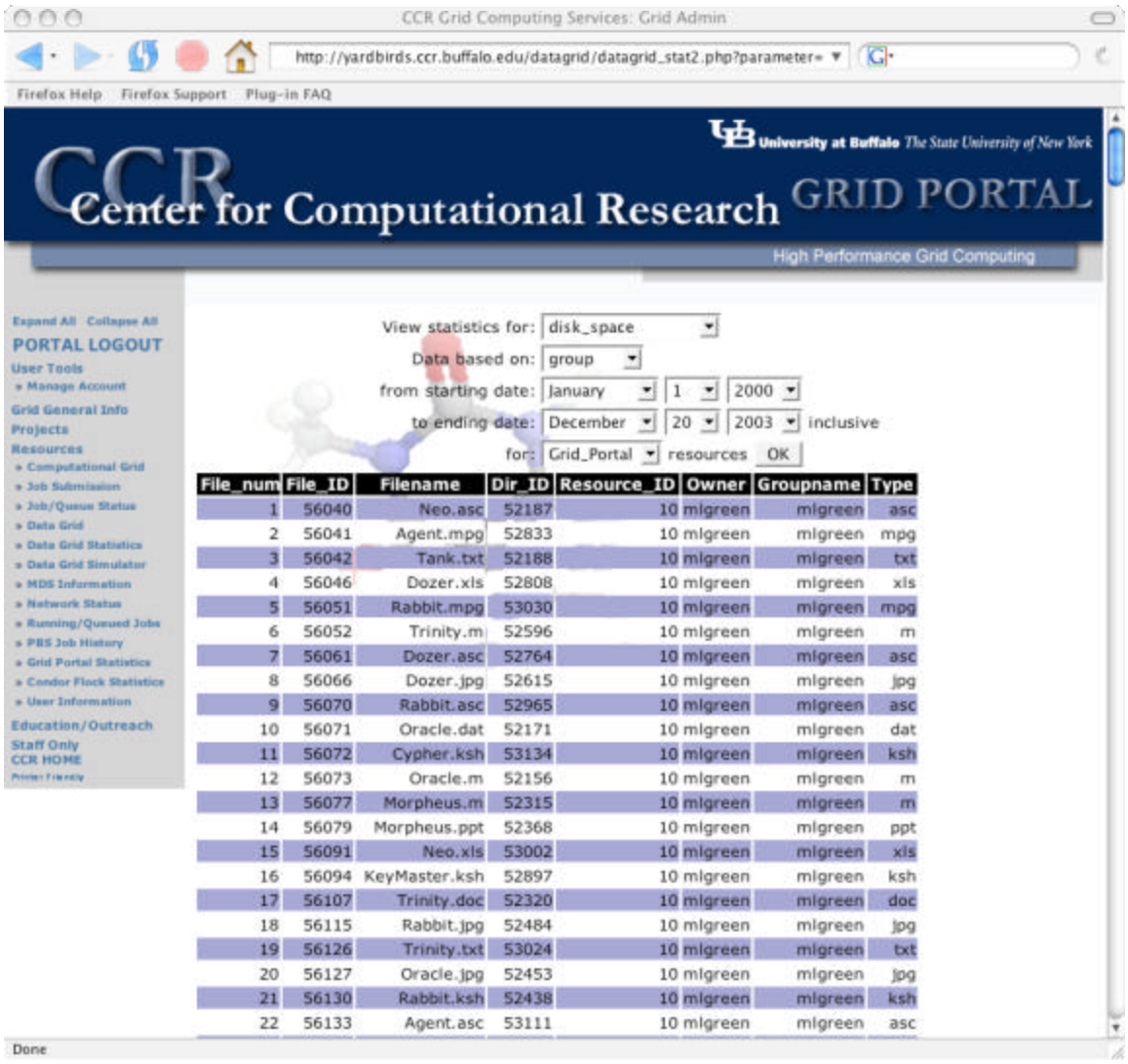


Figure 17. ACDC data grid Java tree view of files.

The hierarchical display does not list the file attribute data, so a list-based display has also been developed that can be used for sorting data grid files based on available metadata (*e.g.*, filename, file size, modification time, owner, etc.), as shown in Figure 18.



File_num	File_ID	Filename	Dir_ID	Resource_ID	Owner	Groupname	Type
1	56040	Neo.asc	52187	10	mlgreen	mlgreen	asc
2	56041	Agent.mpg	52833	10	mlgreen	mlgreen	mpg
3	56042	Tank.txt	52188	10	mlgreen	mlgreen	txt
4	56046	Dozer.xls	52808	10	mlgreen	mlgreen	xls
5	56051	Rabbit.mpg	53030	10	mlgreen	mlgreen	mpg
6	56052	Trinity.m	52596	10	mlgreen	mlgreen	m
7	56061	Dozer.asc	52764	10	mlgreen	mlgreen	asc
8	56066	Dozer.jpg	52615	10	mlgreen	mlgreen	jpg
9	56070	Rabbit.asc	52965	10	mlgreen	mlgreen	asc
10	56071	Oracle.dat	52171	10	mlgreen	mlgreen	dat
11	56072	Cypher.ksh	53134	10	mlgreen	mlgreen	ksh
12	56073	Oracle.m	52156	10	mlgreen	mlgreen	m
13	56077	Morpheus.m	52315	10	mlgreen	mlgreen	m
14	56079	Morpheus.ppt	52368	10	mlgreen	mlgreen	ppt
15	56091	Neo.xls	53002	10	mlgreen	mlgreen	xls
16	56094	KeyMaster.ksh	52897	10	mlgreen	mlgreen	ksh
17	56107	Trinity.doc	52320	10	mlgreen	mlgreen	doc
18	56115	Rabbit.jpg	52484	10	mlgreen	mlgreen	jpg
19	56126	Trinity.txt	53024	10	mlgreen	mlgreen	txt
20	56127	Oracle.jpg	52453	10	mlgreen	mlgreen	jpg
21	56130	Rabbit.ksh	52438	10	mlgreen	mlgreen	ksh
22	56133	Agent.asc	53111	10	mlgreen	mlgreen	asc

Figure 18. ACDC data grid list-based view of sorted user files.

Basic file management functions are available via a platform-independent user-friendly Web interface that includes file transfer capabilities, a simple Web-based file editor, an efficient search utility, and the logical display of files for a given user in three divisions (user/ group/public). Collating and displaying statistical information is particularly useful to administrators for optimizing usage of resources. The ACDC data grid infrastructure periodically migrates files between data repositories for optimal usage of resources. The file migration algorithm depends upon a number of factors, including the following:

- User access time
- Network capacity at time of migration
- User profile
- User disk quotas on various resources

Further, we have the ability to mine log files, which aids in the determination of the following:

- The amount of data to migrate in one cycle

- The appropriate migration cycle length
- The file access pattern of a data grid user
- The access pattern for public or group files

The user global file-aging attribute is indicative of a user's access across their own files and is an attribute of a user's profile. The local file aging attribute is indicative of overall access of a particular file by users having group or public access. The latter is an attribute of a file and is stored in the file management data grid table. During migration, these attributes are used to determine the files that are to be migrated from the grid portal repository to a remote resource repository. Specifically, file migration is a function of global file aging, local file aging, and resource usage (*e.g.*, the previous use of files on individual compute platforms is a factor in determining file migration). By tracking the file access patterns of all user files and storing this information in the associated database tables, the ACDC data grid infrastructure can automatically determine an effective repository distribution of the data grid files. See Figure 19 for a schematic of the physical data ACDC data grid.

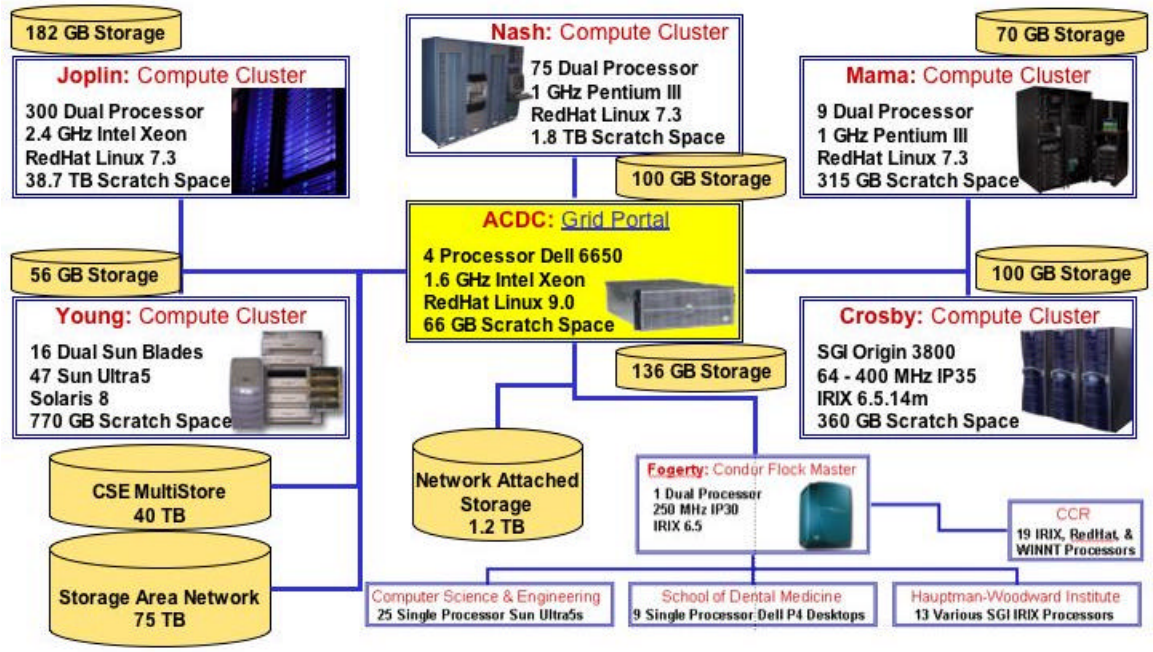


Figure 19. ACDC data grid repository location, network bandwidth, and size.

Support for multiple access to files in the data grid has been implemented with file locking and synchronization primitives. The ACDC data grid also provides security for authentication and authorization of users, as well as policies and facilities for data access and publication. The ACDC data grid algorithms are continually evolving to minimize network traffic and maximize disk space utilization on a per user basis. This is accomplished by data mining user usage and disk space requirements in a ubiquitous and automated fashion.

One advantage of *SnB* is that it can run in either a loosely coupled or tightly coupled fashion, it uses a database management system, it can take advantage of computational steering, it utilizes a geographically distributed interactive back-end visualization system, and it is amenable to an automated backfill mechanism. Currently, we run *SnB* on the ACDC-Grid from either a GUI or Web portal.

Summary. The Grid is a rapidly emerging and expanding technology that allows geographically distributed and independently operated resources (CPU cycles, data storage, sensors, visualization devices, and a wide variety of Internet-ready instruments) to be linked together in a transparent fashion. *SnB* is a computer program based on the *Shake-and-Bake* method of molecular structure determination from X-ray diffraction data. The *Shake-and-Bake* algorithm for molecular structure determination was listed on the IEEE poster "Top Algorithms of the 20^{th} Century." In this chapter, we have discussed the development of tools that allow for an efficient grid-based implementation of *SnB* that is extensible to a wide range of scientific programs.

We introduced the ACDC-Grid, which provides an integrated computational and data grid, lightweight job monitoring, predictive scheduling, and opportunities for improved Grid utilization through an elegant backfill facility. The core infrastructure for the ACDC-Grid includes the installation of standard grid middleware, the deployment of an active Web portal for deploying applications, dynamic resource allocation so that clusters and networks of workstations can be scheduled to provide resources on demand, a scalable and dynamic scheduling system, and a dynamic firewall, to name a few.

Acknowledgments. The authors would like to thank members of the SUNY-Buffalo Grid Team, including Steve Gallo, Naimesh Shah, Jason Rappleye, Cathy Ruby, Jon Bednasz, and Tony Kew, as well as Chuck Weeks, Sam Guercio, Adam Koniak, Martins Innus, Dori Macchioni, Henrique Bucher, and Cynthia Cornelius, for their contributions to the efforts described in this chapter. This research is supported by NSF ITR Grant #0204918 and a post-doctoral fellowship from HP. Computing Resources provided by the Center for Computational Research, SUNY-Buffalo.

References

[1] Grid Computing Info Centre http://www.gridcomputing.com/

[2] Foster, I. and Kesselman, C., *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kauffman Publishers, Inc., San Francisco, 1999.

[3] F. Berman, G. Fox, and T. Hey, *Grid Computing: Making the Global Infrastructure a Reality*, John Wiley & Sons, 2003.

[4] The Globus Alliance, http://www/globus.org

[5] 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.

[6] C.M. Weeks and R. Miller, The design and implementation of *SnB* v2.0, *J. of Applied Crystallography* 32, 1999, pp. 120-124.

[7] J. Rappleye, M. Innus, C.M. Weeks, and R. Miller, *SnB* v2.2: An Example of Crystallographic Multiprocessing, *J. of Applied Crystallography* 35, 2002, pp. 374-376.

[8] 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.

[9] 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 Crystallograpica* A50, 1994, pp. 203-210.

[10] M.L. Green and R. Miller, Grid computing in Buffalo, New York, Annals of the European Academy of Sciences, 2003, pp. 191-218.

[11] 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.

[12] M.L. Green and R. Miller, Molecular structure determination on a computational & data grid, Parallel Computing, Volume 30, Issues 9-10, September-October 2004, Pages 1001-1017.

[13] M.L. Green and R. Miller, Evolutionary molecular structure determination using grid-enabled data mining, Parallel Computing, Volume 30, Issues 9-10, September-October 2004, Pages 1057-1071.

[14] M.L. Green and R. Miller, A client-server prototype for application grid-enabling template design, January, 2004 http://www.cse.buffalo.edu/pub/WWW/faculty/miller/Papers/PPL04-1.pdf

[15] Globus Toolkit 3, http://www-unix.globus.org/toolkit/

[16] Globus Resource Allocation Manager (GRAM), http://www-unix.globus.org/developer/resource-management.html

[17] GridFTP Data Transfer Protocol, http://www.globus.org/datagrid/gridftp.html

[18] K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, S. Tuecke, From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution, http://www.globus.org/wsrf/OGSI%20to%20WSRF%201.0.pdf

[19] Web Services Resource Lifetime, http://www.globus.org/wsrf/WS-ResourceLifetime.pdf

[20] Web Services Resource Properties, http://www.globus.org/wsrf/WS-ResourceProperties.pdf

[21] Web Services Notification, http://www.globus.org/wsrf/WS-Notification.pdf

[22] Modeling Stateful Resources with Web Services, http://www.globus.org/wsrf/ModelingState.pdf

[23] The Globus Alliance: WS-Resource Framework, http://www.globus.org/wsrf/

[24] pyGlobus Project, http://www-itg.lbl.gov/gtg/projects/pyGlobus/

[25] XML Package for Python, http://pyxml.sourceforge.net/

[26] Python Extensions for the Grid (PEG), http://grail.sdsc.edu/projects/peg/introduction.html

[27] Numerical Python, http://www.pfdubois.com/numpy/

[28] Scientific Python, http://starship.python.net/~hinsen/ScientificPython/

[29] netCDF Library, http://www.unidata.ucar.edu/packages/netcdf/

[30] Message Passing Interface (MPI), http://www-unix.mcs.anl.gov/mpi/

[31] Bulk Synchronous Parallel Programming (BSPlib) http://www.bsp-worldwide.org/

[32] R.H. Bisseling, *Parallel Scientific Computation: A structured approach using BSP and MPI*, Oxford University Press, February 2004, 305 pages. ISBN 0-19-852939-2.

[33] SciPy Library, http://www.scipy.org/

[34] pyGridWare Project, http://www-itg.lbl.gov/gtg/projects/pyGridWare/

[35] Integrate Perl into OGSI-based Grid Applications and Services, http://www-106.ibm.com/developerworks/library/gr-perlinf.html

[36] SOAP::Lite, http://www.soaplite.com/

[37] OGSI::Lite, http://www.sve.man.ac.uk/Research/AtoZ/ILCT

[38] MS .NET Grid, http://www.nesc.ac.uk/action/projects/project_action.cfm?title=145

[39] MS.NETGrid Project, http://www.epcc.ed.ac.uk/~ogsanet/

[40] Microsoft, http://www.microsoft.com/

[41] National e-Science Centre, http://www.nesc.ac.uk/

[42] OGSI.NET, http://www.cs.virginia.edu/~humphrey/GCG/ogsi.net.html

[43] Optimalgrid, http://www.alphaworks.ibm.com/tech/optimalgrid

[44] Seti@home, http://setiathome.ssl.berkeley.edu/

[45] Folding@home, http://folding.stanford.edu

[46] MOAB Grid Scheduler (Silver), http://www.supercluster.org/silver/

[47] Sun Grid Engine (SGE), http://gridengine.sunsource.net/

[48] Sphinx Scheduler, http://www.griphyn.org/sphinx/

[49] J.Patton Jones and B.Nitzberg, Scheduling for Parallel Supercomputing: A Historical Perspective of Achievable Utilization, in *Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science 1659, pages 1-16. Springer-Verlag, 1999

[50] Dmitry Zotkin and Peter J. Keleher, Job-length estimation and performance in backfilling schedulers, 8th High Performance Distributed Computing Conference, IEEE, 1999

[51] S.-H. Chiang and M. Vernon, Production Job Scheduling for Parallel Shared Memory Systems, Proc. Int'l. Parallel and Distributed Processing Symp. (IPDPS)

[52] W. Smith, V. Taylor, and I. Foster, Using run-time predictions to estimate queue wait times and improve scheduler performance, in *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), Springer Verlag, 1999 Lect. Notes Comput. Sci.

[53] R. Gibbons, A Historical Application Profiler for Use by Parallel Schedulers, Lecture Notes on Computer Science, pages 58-75, 1997

[54] W. Smith, V. Taylor, and I. Foster, Using run-time predictions to estimate queue wait times and improve scheduler performance, in *Job Scheduling Strategies for Parallel Processing*, D. G. Feitelson and L. Rudolph (eds.), Springer Verlag, 1999 Lect. Notes Comput. Sci.

[55] John Keenan Talyor, Statistical Techniques for Data Analysis, Lewis Publishers, Inc., 121 South Main Street, Chelsea, MI 48118, 1990

[56] Ian Foster, Jens Vöckler, Michael Wilde, Yong Zhao, Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation, *14th International Conference on Scientific and Statistical Database Management* (SSDBM 2002).

[57] Storage Resource Manager, http://sdm.lbl.gov/srm-wg.

[58] Storage Resource Management: Concepts, Functionality, and Interface Specification, Arie Shoshani, Future of Data Grids Workshop, Berlin, 2004

[59] Storage Element Service, http://www.ivdgl.org/grid2003/news/

[60] SRM Joint Functional Design, Version 1 .http://sdm.lbl.gov/srm/documents/joint.docs/SRM.joint.func.design.part1.doc.

[61] SRM joint methods specification version 1 http://sdm.lbl.gov/srm/documents/joint.docs/srm.v1.0.doc.

[62] The Storage Resource Manager Interface Specification, version 2.1, Edited by Junmin Gu, Alex Sim, Arie Shoshani, available at http://sdm.lbl.gov/srm/documents/joint.docs/SRM.spec.v2.1.final.doc.

[63] The Network Weather Service, http://nws.cs.ucsb.edu/

[64] W. Smith, I. Foster, and V. Taylor, Scheduling with advanced reservations, International Parallel and Distributed Processing Symposium (IPDPS '00), May 2000

[65] The Virtual Data Toolkit, http://www.lsc-group.phys.uwm.edu/vdt/

[66] International Virtual Data Grid Laboratory, http://www.ivdgl.org/

[67] GRID2003, http://www.ivdgl.org/grid2003

[68] The Grid Laboratory Uniform Environment (GLUE), http://www.hicb.org/glue/glue.htm

[69] Grid Physics Network GriPhyN, http://www.griphyn.org/index.php

[70] Particle Physics Data Grid, http://www.ppdg.net/

[71] European Data Grid, http://eu-datagrid.web.cern.ch/eu-datagrid/

[72] TransAtlantic Grid 9DataTAG), http://datatag.web.cern.ch/datatag/

[73] CrossGrid, http://www.eu-crossgrid.org/

[74] CERN, http://public.web.cern.ch/public/

[75] IBM Grid Toolbox, http://www-132.ibm.com/webapp/wcs/stores/servlet/CategoryDisplay?storeId=1&catalogId=-840&langId=-1&categoryId=2587007.

[76] IBM General Purpose File System (GPFS), http://www-1.ibm.com/servers/eserver/clusters/software/gpfs.html.

[77] NSF Network for Earthquake Engineering Simulation (NEES) Grid, http://www.nees.org/

[78] Structural Engineering Earthquake Simulation Laboratory (SEESL), http://www.civil.buffalo.edu/Facilities/research.html

[79] Multidisciplinary Center for Earthquake Engineering (MCEER), http://mceer.buffalo.edu/

[80] NEEsgrid software release 3.0, http://www.neesgrid.org/software/neesgrid3.0/.

[81] Open Science Grid Consortium, http://www.opensciencegrid.org/activities/index.html.

[82] Ganglia: Distributed Monitoring and Execution System. http://ganglia.sourceforge.net.

[83] Matthew L. Massie, Brent N. Chun, and David E. Culler. The ganglia distributed monitoring system: Design, implementation, and experience. Parallel Computing, Vol. 30, Issue 7, July 2004

[84] Mike Wawrzoniak, Larry Peterson, and Timothy Roscoe. Sophia: An Information Plane for Networked Systems. In Proceedings HotNets-II, Cambridge, MA, USA, November 2003

[85] Sophia, http://www.cs.princeton.edu/~mhw/sophia/documents.php.

[86] Planetlab. http://www.planet-lab.org.

[87] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. In Proceedings of the 5th Annual Symposium on Operating Systems Design and Implementation (OSDI), December 2002

[88] IBM Tivoli Monitoring. http://www.ibm.com/software/tivoli/products/monitor.

[89] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001

[90] X. Zhang and J. Schopf, Performance Analysis of the Globus Toolkit Monitoring and Discovery Service, MDS2, Proceedings of the International Workshop on Middleware Performance (MP 2004), part of the 23rd International Performance Computing and Communications Workshop (IPCCC), April 2004

[91] Globus Alliance WS information services: Key concepts. http://www-unix.globus.org/toolkit/ docs/3.2/infosvcs/ws/key/index.html, 2004

[92] Z. Balaton and G. Gombás, Resource and Job Monitoring in the Grid, in Proc. of the Euro-Par 2003 International Conference on Parallel and Distributed Computing, Klagenfurt, Austria (2003).

[93] R-GMA, http://www.r-gma.org.

[94] Hawkeye: A monitoring and management tool for distributed systems. http://www.cs.wisc.edu/condor/hawkeye/.

[95] MonaLisa Monitoring, http://gocmon.uits.iupui.edu:8080/index.html

[96] NMI-R4, http://www.nsf-middleware.org/NMIR4/

[97] NMI-R4 New Release, http://www.nsf-middleware.org/Outreach/news_12-16-03.asp

[98] J. Holland. Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, 1992

[99] A. Hunter, SUGAL User Manual v2.0, http://www.dur.ac.uk/andrew1.hunter/Sugal/

[100] D. Levine. PGAPack, 1995 A public-domain parallel genetic algorithm library. Available anonymous ftp from ftp.mcs.anl.gov in the directory pub/pgapack, file pgapack.tar.Z.

[101] D. Levine. *Users guide to the PGAPack parallel genetic algorithm library*. Technical Report ANL-95/18, Argonne National Laboratory, Mathematics and Computer Science Division, June 23, 1995

[102] D. Ho, personal communication: C109 H73 N1

[103] D. Ho, personal communication: C220 H148

[104] M.G. Usha and R.J. Wittebort, Orientational ordering and dynamics of the hydrate and exchangeable hydrogen atoms in crystalline crambin, *J Mol Biol* **208 (4),** pp. 669-678, 1989

[105] M.G. Usha and R.J. Wittebort, Orientational ordering and dynamics of the hydrate and exchangeable hydrogen atoms in crystalline crambin, *J Mol Biol* **208 (4),** pp. 669-678, 1989

[106] D.A. Langs, G.D. Smith. C. Courseille, G. Précigoux, and M. Hospital, Monoclinic uncomplexed double-stranded, antiparallel, left-handed ß$^{5.6}$-helix (↑↓ß$^{5.6}$) structure of Gramicidin A: Alternative patterns of helical association and deformation, *Proc. Natl. Acad. Sci. USA*, Vol. 88, pp. 5345-5349, June 1991.

[107] V. Pletnev, N. Galitskii, G.D. Smith, C.M. Weeks, and W.L. Duax, Crystal and molecular structure of Isoleucinomycin, cyclo[-(D-Ile-Lac-Ile-D-Hyi)3-] (C60H102N6O18), *Biopolymers* **19**, pp. 1517-1534, 1980.

[108] C.M. Weeks and W.L. Duax, 9α-Chlorocortison, an Active Cortisol Derivative, *Acta Cryst.* **B30,** pp. 2516-2519, 1974.

[109] J.M. Hodsdon, G.M. Brown, L.C. Sieker, and L.H. Jensen, Refinement of triclinic Lysozyme: I. Fourier and Least-Squares Methods, *Acta Crystallogr.* **B46**, pp. 54-62, 1990.

[110] P.J. Loll, R. Miller, C.M. Weeks, and P.H. Axelsen, A ligand-mediated dimerization mode for vancomycin, *Chemistry and Biology* **5**, pp. 293-298, 1998.