Grid Computing in Buffalo, New York

Mark L. Green* and Russ Miller**

* Center for Computational Research, University at Buffalo, 9 Norton Hall, Buffalo, New York 14260; Civil, Structural and Environmental Engineering, University at Buffalo, 240 Ketter Hall, Buffalo, New York 14260, (716) 645-6500 x522, mlgreen@ccr.buffalo.edu

** Center for Computational Research, University at Buffalo, 9 Norton Hall, Buffalo, New York 14260; Department of Computer Science and Engineering, Box 602000, University at Buffalo, Buffalo, New York 14260, (716) 645-6500 x502, miller@buffalo.edu

Abstract

A computational and data grid developed at the Center for Computational Research in Buffalo, New York, will provide a heterogeneous platform to enable scientific and engineering applications to run in a Buffalo-centric grid-based setting. A proof-of-concept heterogeneous grid has been developed using a critical scientific application in the field of structural biology. The design and functionality of the prototype grid web portal is described, along with plans for a production level grid system based on Globus. Several projects covering a collaborative expansion of this system are also summarized with respect to the core research being investigated. This expansion involves researchers located across the United States who are interested in analyzing and grid-enabling existing software applications and grid technology.

Development of Parallel Computing

In 1970s, the VAX timeshare resources dominated the computing scene, but these machines required high initial capital expenditures and very expensive annual maintenance costs. These timeshare resources also required very highly skilled users who were capable of dealing with CPU power constraints, restricted memory scenarios, and modest disk storage systems. In the first half of the 1980s, UNIX workstations became available and were more widely deployed than the VAX timeshare resources. However, while these early workstations continued to provide only limited CPU power, memory, and disk storage, they were much more accessible and user friendly than the VAX timeshares of the 1970s.

More powerful UNIX desktop workstations, networking Ethernet, High Performance Parallel Interface (HPPI) and Fiber Distributed Data Interface (FDDI) networking, as well as distributed file systems and servers were deployed during the second half of the 1980s. In an effort to utilize these relatively high-powered CPUs, with significantly increased memory and disk storage devices, a number of parallel languages, including P4, Express, and LINDA, began to appear [1,7]. See Table 1.

During the early 1990s, desktop workstations began to be incorporated into distributed computing systems. Further, the capabilities of CPUs, memory, and disk storage increased rapidly during this period. Asynchronous Transfer Mode (ATM) used for Wide Area Networks (WAN) allowed networks to efficiently carry services of the future. Network and computer performance increased by 1000 times and standards such as Message Passing Interface (MPI), High Performance Fortran (HPF), and Distributed Computing Environment (DCE) began to emerge [8].

PVM. Parallel Virtual Machine (PVM) is a software package that permits a heterogeneous collection of Unix and/or Windows computers connected together by a network to be used as a single parallel computer. The first version of PVM was written during the summer of 1989 at Oak Ridge National Laboratory. This initial version of PVM was used internally and not released publicly. Based on the internal success of PVM, Version 2 of the code was redesigned and written from scratch during February 1991 at the lab's sister institution, the University of Tennessee, Knoxville. Version 2 of the code was publicly released in March of 1991. This version was intended to clean up and stabilize the system so that external users could reap the benefits of this parallel computing middleware. PVM Version 3 was redesigned from scratch, and a complete rewrite started in September 1992, with first release of the software in March 1993. While similar in spirit to version 2, version 3 includes features that did not fit the old framework, including fault tolerance, better portability and scalability. Three subsequent versions of PVM were released over the next 9 years. The current version of the system is entitled PVM Version 3.4.4, which was released in September of 2001. Concurrent development of XPVM provided a graphical interface to the PVM console commands and information, along with several animated views to monitor the execution of PVM programs. These views provide information about the interactions among tasks in a PVM program in order to assist in debugging and performance tuning. XPVM Version 1.0 was released in November of 1996 and the latest XPVM Version is 1.2.5, released April 1998 [2].

MPI. The specification of the Message Passing Interface (MPI) standard 1.0 [3] was completed in April of 1994. This was the result of a community effort to try and define both the syntax and semantics of a core message-passing library that would be useful to a wide range of users and implemented on a wide range of Massively Parallel Processor (MPP) platforms. Clarifications were made and released in June 1995, where the major goals were portability, high performance, "common practice", features process model, point-to-point communication, collective operations, and mechanisms for writing safe libraries. All major computer vendors supported the MPI standard and work began on MPI-2, where new functionality, dynamic process management, one-sided communication, cooperative I/O, C++ bindings, Fortran 90 additions, extended collective operations, and miscellaneous other functionality were added to the MPI-1 standard [4]. MPI-1.2 and MPI-2 were released at the same time in July of 1997. The main advantage of establishing a message-passing standard is portability. One of the goals of developing

MPI is to provide MPP vendors with a clearly defined base set of routines that they can implement efficiently or, in some cases, improve scalability by providing hardware support. Local Area Multi-computer (LAM) development followed as an MPI programming environment and development system for heterogeneous computers on a network. LAM-MPI 6.1 was released in June 1998 and further development of a graphical user interface continued as XMPI 1.0 was released in January 1999. With LAM-MPI, a dedicated cluster or an existing network computing infrastructure can act as one parallel computer solving one problem and be monitored with a graphical user interface [5].

Development of Grid Computing

Integrating computational resources into parallel and distributed systems has become common practice since the early 1990s. A (computational) grid can be defined as a computing system in which computational resources, including computing, storage, databases, devices, sensors, and tools, are organized into a cohesive distributed system that spans multiple geographic and administrative domains. In this section, we provide a very brief history of grid computing, focusing on the capabilities of several toolkits and software packages that are critical to the Center for Computational Research Grid (CCR-Grid) that is the subject of this paper. In order to provide a framework for the discussion that follows in this section, it is important to know that the CCR-Grid system will leverage many of the communication, authentication, concurrency, security, system monitoring, and error handling capabilities available in Globus, a critical public domain grid software package that has become the *de facto* standard in academic (and many industrial) settings.

Globus. The Globus project was established in 1996. It focuses on enabling the application of various grid concepts, predominantly in the domain of computational science and engineering. The Globus project is centered at Argonne National Laboratory, the University of Southern California, and the University of Chicago, although numerous other research groups and laboratories have made significant contributions over the past several years. Groups all over the world are using the open source Globus Toolkit to build cost-effective computational platforms. An example of these efforts include designing smart instruments, where, for example, a microscope can be coupled to supercomputers, users, and databases, all available over a grid, where each of the items are physically located at distinct locations and each such item is governed by a distinct, though cooperating, management scenario. Another popular use for grids is to provide large-scale desktops, where from a desktop, a user has access to a nearly ubiquitous computational grid that provides access to computationally-intensive disciplinary packages (e.g., computational chemistry).

The open source Globus Toolkit [25] consists of a set of components, implemented as APIs (application programmer interfaces) written in the C programming language, that are designed to be useful for developing grid applications. The major components follow (from www.globus.org).

- The Globus Resource Allocation Manager (GRAM) provides resource allocation and process creation, monitoring, and management services. GRAM implementations map requests expressed in a Resource Specification Language (RSL) into commands that may be submitted to local schedulers (e.g., Loadleveler, PBS, LSF).
- The Grid Security Infrastructure (GSI) provides a single "sign-on, run-anywhere" authentication service. GSI supports local control over access and the mapping from a global user identity to a local user identity.
- The Monitoring and Discovery Service (MDS) is an extensible information service that combines data discovery mechanisms with Lightweight Directory Access Protocol (LDAP). MDS provides a uniform framework for providing and accessing system configurations and status information, including server configurations, status of networks, and locations of replicated databases.
- The Global Access to Secondary Storage (GASS) implements a variety of automatic and programmer-managed data movement and data access strategies, enabling programs running at remote locations to read and write local data.
- Nexus and globus_io provide communication services for heterogeneous environments.
- The Heartbeat Monitor (HBM) allows system administrators or users to detect the failure of system components or application processes.

With the November, 2001 release of Globus Toolkit 2.0, eight firms (Compaq, Cray, SGI, Sun, Veridian, Fujitsu, Hitachi, and NEC) announced that they will develop optimized forms of the toolkit for their operating platforms in an effort to eventually provide a secure, distributed, multi-vendor grid computing system. Three other companies (Entropia, IBM, and Microsoft) simultaneously announced expansions of previous commitments to the Globus Project. Platform Computing has also recently released a supported version of the Globus Toolkit. In addition, IBM has recently joined in the development of the next-generation Globus Toolkit 3.0, to be based on Open Grid Services Architecture (OGSA) [23] specifications drafted by Foster, Tuecke, Kesselman and IBM colleagues.

Condor. Based on the observation that a large percentage of desktop compute cycles go unused, Condor (http://www.cs.wisc.edu/condor/) was designed to be a robust system that would scavenge unused cycles in networked workstations, while providing the owner of a workstation with the ultimate control as to when their workstation was made available to other users (e.g., evenings, weekends, lunch time, whenever it was not being utilized, etc.). That is, Condor is designed to be a specialized workload management system for compute-intensive jobs, providing a job queue, scheduling policy, priority scheme, resource monitor, and resource management system. So, users may submit their sequential or parallel jobs to Condor, which places the jobs into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion. Condor has been quite successful in managing large networks of workstations and commodity clusters.

In 1988, a workload management system for computationally-intense applications was created by Michael Litzkow, who designed the first version of the Condor Resource Management system. The results of the Remote-Unix (RU) project, directed by David Dewitt, Raphael Finkel, and Marvin Solomon, provided mechanisms for handling environments with heterogeneous distributed resources. The work in the area of Distribute Resource Management (DRM), directed by Miron Livny, was also merged into Condor for load balancing distributed systems. The Condor management policies were provided by the distributed allocation and preemptive scheduling techniques developed by Matt Mutka [12,13].

The recent development of Condor-G, a grid-enabled version of Condor that utilizes Globus to handle issues that arise while coordinating a variety of organizations, including security and resource management, represent a major step forward in grid/distributed computing. That is, Condor-G combines the inter-domain resource management protocols of the Globus Toolkit and the intra-domain resource and job management methods of Condor to allow the user to harness multi-domain resources as if they all belong to one personal domain [14,15].

Legion. The Legion Project [10,26] was initiated in 1996 by Andrew Grimshaw, a faculty member at the University of Virginia. The first public release of the Legion Technology was at Supercomputing '97. In June 1998, Grimshaw joined the Grid Forum as an original Steering Group member. The Grid Forum was a community driven organization for researchers who worked on distributed computing and grid technologies. In 2000, the Grid Forum merged with the Global Grid Forum (GGF) [37] to help promote common practices and interoperability between large-scale grid systems. The goal of Legion is to foster the utilization of standard object representations in the design of distributed systems. The Legion Project capabilities include the following.

- The current release of Legion offers Basic Fortran Support (BFS), which provides a set of Legion directives that can be embedded in Fortran code in the form of pseudo-comment lines.
- The Mentat Programming Language (MPL) is an extension of C++ and is designed to help parallelize applications.
- Applications using PVM can use the Legion core PVM interface to use Legion features in a PVM environment. PVM programs can be registered and run with special Legion tools.
- Legion also has a core MPI interface, which lets MPI users take advantage of Legion features. All MPI features are supported in Legion MPI and there are special Legion tools for registering and running MPI programs.

While Globus and Legion are often thought to have significant overlap in their grid-based computation goals, one may take the view that Globus places an emphasis on providing low-level services, while Legion promotes high-level programming models.

In February 2001, Applied MetaComputing was founded by Grimshaw in an effort to commercialize the Legion technology. In June 2001, Applied MetaComputing officially re-launched as Avaki Corporation and continued the Data and Compute Grid development. Avaki released several versions of Compute Grid and Data Grid throughout 2002 that enhanced many core capabilities [11]. Some of the features include the following.

- Execute parallel or sequential jobs efficiently based on requirements and policies.
- Enhance the current processing infrastructure without disrupting users or modifying applications.
- Establish usage policies that allocate resources optimally across locations and departments.
- Insulate users from system complexity, and reduce the burden on administrators.
- Keep usage policies under control of local resource owners and utilize resources efficiently.

Gridware & Sun Grid Engine. In 1990, Wolfgang Gentzsch founded a consulting and software development company called Genias Software. This company began the development of a prototype in 1992 for a distributed resource management system, later named Codine. From 1995 to 1999, Gentzsch and his staff participated in many early grid-computing projects, including Unicore, Autobench, Medusa, Julius, and Eroppa. In 1999, Genias Software changed its name to Gridware and began the development of interfaces between Codine and open source grid projects, including Globus [18], Legion [10], Punch [9], and Cactus [16]. Sun Microsystems acquired Gridware in July of 2000 and began work on the next generation of Codine, named the Sun Grid Engine software. In September of 2000, Sun delivered Sun Grid Engine 5.2 version for the Solaris™ Operating System. The Linux version was released January 2001 and by June 2001 over 10,000 free downloads of the Sun Grid Engine software were made. Sun established the Grid Engine Project, placing all 500,000 lines of source code into the public domain via CollabNet. Currently, Sun has integrated the Sun Grid Engine system and the Sun Open Net Environment Portal server, so that users can administer a grid through a Web interface. The latest release of Sun Grid Engine software (Sun ONE platform) is version 5.3 with an estimated 5,100 users worldwide [6].

Grid-Enabled *Shake-and-Bake* Proof-of-Concept

The CCR-Grid proof-of-concept has been completed using a Client/Server framework including a dynamically created HTML Grid Console for monitoring parallel grid jobs. Results from this study are being used in the design of the production-level Globus-based CCR-Grid that will serve as a follow on. In particular, the proof-of-concept UNIX shell scripts and Client/Server framework will be replaced by HTML, Java Script, PHP, MySQL, phpMyAdmin, WSDL and the Globus Toolkit in the production-level CCR-Grid.

Problem Statement

In order to implement a proof-of-concept CCR-Grid, we consider as an application a cost-effective solution to the problem of determining molecular crystal structures via direct methods as implemented in a grid setting. We use the program *Shake-and-Bake* (*SnB*) as the application for a variety of reasons. *SnB* was developed in Buffalo and is the program of choice for structure determination in many of the 500 laboratories that have acquired it. In addition, the *SnB* program well understood by the authors, as the second author of this paper is one of the principle authors of the *Shake-and-Bake* methodology and the *SnB* program. Finally, it is a computationally intensive program that can take advantage of the grid's ability to present the user with a large-scale desktop or distributed supercomputer in order to perform computations that are equivalent to parameter studies, which are areas that the grid excels at.

The *SnB* program uses a dual-space direct-methods procedure for determining crystal structures from X-ray diffraction data [19-22]. 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 by *SnB* have led to the determination of complete structures containing *hundreds of thousands* of atoms.

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

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

The current premise is that the computing framework for this *Shake-and-Bake* need not be restricted to local computing resources. Therefore, a grid-based solution to *Shake-*

and-Bake can afford scientists with limited local computing capabilities the opportunity to solve structures that would be beyond their means.

Statement of Approach

The Center for Computational Research at the University at Buffalo serves as an ideal testbed for producing a grid-based implementation of *SnB.* In fact, the gapcon list [35] ranks the Center for Computational Research as the 8th most powerful supercomputing site in the world as of November 28, 2002. Further, CCR provides a diverse set of computational platforms, including the following systems.

- 4000 processor Dell PentiumIII/Xeon Cluster (code name DNA RNA)
- 64 processor SGI Origin 3800 (CROSBY)
- 78 processor IBM SP2 (STILLS)
- 150 processor SGI-Intel PentiumIII Cluster (NASH)
- 82 processor Sun Ultra 5 Cluster (YOUNG)
- 4 processor DEC alpha (MOONGLOWS)
- 16 processor IBM 340 Cluster (MAMAS PAPAS)
- 3 processor solar powered G4 briQ Cluster (BRIQ)
- 604 processor Dell Pentium4 Cluster (JOPLIN)
- IBM 44P Workstation (COASTERS)
- Dual Processor SGI Octane (THEDOORS)
- 6 Processor SGI Onyx (CREAM)

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

- The Grid Server is the unique master grid process on the CCR-Grid. Users submit *SnB* jobs to the Grid Server. In addition, the Grid Server is used to automatically configure and register Platform Servers, which control individual compute platforms, including managing load balancing across the grid in concert with such Platform Servers. The Grid Server also maintains the trial database, including records of where trials have been sent for processing and results of trials that have been completed.
- Platform Server processes are used to register and configure Node Servers, which manage *SnB* worker processes. A Platform Server is also used to manage Block Queues (i.e., dispatched sets of *SnB* jobs) and report the status of Node Servers to the Grid Server. There is one Platform Server running per computational grid resource (platform) configured.
- A Node Server process runs on a platform production node and manages the *SnB* application worker processes. The number of workers executed by the Node Server is proportional to the number of processors of the production node.

- A series of Grid, Platform and Node Servers using TCP socket, and SSH secure tunneling/port forwarding facilities are used to configure the computational grid.
- Each platform uses native queuing software (PBSpro [28], OpenPBS [27], Loadleveler [29], and LSF [30]) to submit the requested Node Server processes.
- The Node Servers receive all trials to process from the Platform Server that they are registered with and then transmit the trial result back to the Platform Server.
- The Platform Servers receive all trials to process from the Grid Server and transmit all Node Server trial solutions to the Grid Server.

The master process, Grid Server, is used to manage and control all of the grid resources. This process can dynamically remove/add/display/control all grid resources. Note that it is important to request resources for the shortest time possible so that the resources can be released from the CCR-Grid back into production in its native mode. Each platform queuing system determines which production nodes are available for grid resources for the requested time. The Grid Server may attempt to add or delete resource production nodes during the lifetime of the grid, as needed, or when computational resources become available. The Grid Server is the only process that the end user communicates with directly. The Grid Server can add or remove resources by using one of the three types of configurations shown in Figure1.

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

The Client/Server Type 2 is the firewall cluster configuration used when a Relay Server cannot connect to the Grid Server directly. The Grid Server communicates with the Platform Server that only has an internal Internet address through SSH encrypted tunnels setup on the cluster firewall machine. For convenience, the SSH tunnels can be setup from the Grid Server remotely and remain installed even after the grid lifetime has expired. The Platform Server can then setup network socket connections to Node Server processes running on the platform production nodes.

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

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

- Grid Server
    - Date
    - Parallel Run Time
    - Trial/Minute (rate of progress)
    - Completion Time (estimated remaining time / total completion time)
    - Serial Run Time (estimate)
    - Speedup (estimate)
    - Start Trial Number
    - Finish Trial Number
    - Number of Trials Executed / Total Trials
- Platform Server Status
    - Platform – Picture / Architecture Type / Machine Name
    - Status – Idle / Working / Offline
    - Resources – Nodes / Total Process / Available Process / Running Process
    - Start Trial Number
    - Finish Trial Number
    - Number of Trials Executed / Total Trials
- Console
    - Start Trial Number
    - Finish Trial Number
    - Total Trials (total number of trials to process)
    - Platform Server State – Block Queue / Float / Race

This grid-enabled *SnB* application run shows many of the problems that can be encountered when submitting a job to the CCR-Grid. Since not all of the computational resources are dedicated for grid computing, any time that a user wishes to run a grid-enabled application, a request must be submitted through the batch queue. If the production machines happen to be 100% utilized, the submitted request will be held in the respective platform queue until sufficient production nodes are available.

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

This situation is commonplace when dealing with multiple independent heterogeneous platforms. That is, platforms that become available after an application run has already started are configured and remain "IDLE" until a new application run is issued. The *SnB* application, as shown in Figure 2, was instructed to run 50,000 trials starting from 1 to 50,000 with 9 Platform Servers and 319 nodes yielding 645 processes. From the dynamically created web page, one can see that the application has been running for 43.55 minutes and processing trials at the rate of 260.76 trials per minute. In addition, the estimated total completion time is 191.75 minutes with 148.20 minutes left in the run, and the estimated speedup over aggregate sequential computing is 601.46.

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

- Block Queues
  - Each Platform Server is given a block of trials to process from the Grid Server based on a platform load factor determined from the time required to process one trial. A Platform Server is completely responsible for submitting the assigned trials to its registered Node Servers for processing.
  - The Block Queues account for 85% of the total number of *Shake-and-Bake* trials submitted for processing. The remaining 15% of the trials are allocated to the Float queue.
- Float
  - The *Shake-and-Bake* trials that have been reserved for Float are submitted individually to any Platform Server that requests more trials to process from the Grid Server.
  - The distribution of Float trials to Platform Servers is determined at runtime and is completely dynamic.
  - Any Platform Server in the grid can process the Float trials after their respective Block Queue has been completed.
- Race
  - When a Platform Server requests more trials to processes from the Grid Server and the Float trials have been exhausted, the Grid Server queries its database of results and assigns trials that have been assigned, but not completed, to the requesting Platform Server for processing. This continues until solutions have been obtained and recorded in the Grid Server database for all requested trials.

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

- A platform is still in the process of working on its Block Queue trials.
- Trial results were lost during transmission.
- The platform that processed a trial has gone offline or lost network connectivity without transmitting trial results.

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

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

- *Pt* is the total number of Platform Servers
- *Nt* is the total number of Node Servers
- *Tp* is the platform speed given in seconds
- *Tpm* is the mean platform speed given in seconds
- *Tn* is the node trial time given in seconds
- *Pp* is the number of platform processes

The platform speed is given by

$$Tp_i = \left(\frac{1}{Nt} \sum_{j=1}^{Nt} \frac{1}{Tn_j} \right)^{-1} \quad for \quad i = 1 \cdots Pt,$$

where Tp_i denotes the *i*th Platform Server speed. The Platform Servers mean is calculated by

$$Tpm = \frac{1}{Pt} \sum_{i=1}^{Pt} Tp_i \,.$$

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

$$PLF_i = \frac{Pp_i + Pp_i \left(\frac{Tpm - Tp_i}{Tpm} \right)}{\sum_{j=1}^{Pt} \left[Pp_j + Pp_j \left(\frac{Tpm - Tp_j}{Tpm} \right) \right]} \quad for \quad i = 1 \cdots Pt,$$

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

The bars located in the Platform Server Status columns, as shown in Figure 2, indicate the percentage of the Block Queue that has been completed. For instance, the Grid Server assigned JOPLIN 39586 trials for its Block Queue, beginning at trial number 1558 and ending at trial number 41143. From the snapshot given in Figure 2, we see that JOPLIN has thusfar completed 27% of its queue (10930 trials). The bars located in the Console column indicate the percentage of the total job trials that have been completed. Figure 3 shows the state of the grid-enabled *SnB* application approximately two hours later. In this case, the Float Trials allocated to JOPLIN were completely processed and JOPLIN is now available to accept Race Trials. That is, JOPLIN is prepared to process trials that have been assigned, but not completed, by other platforms. In Figure 3 note that the JOPLIN status is "Race" and 97 % of the total *SnB* application trials have been completed. The total parallel run time is 183.6 minutes with an estimated time to comple tion of 4.27 minutes and an overall estimated speedup of 610.84.

We have chosen this grid-enabled *SnB* application run to illustrate the importance of implementing robust forms of fault tolerance and load balancing. This is essential when using a wide range of heterogeneous computational platforms. The current load-balancing scheme produces a robust fault tolerant system that drives the grid platforms toward 100% usage until the application job has been completed. Further research into more adaptive schemes of load balancing is currently being investigated along with incorporating methods derived from other researchers [34].

CCR Web Portal

The CCR-Grid uses a web portal, as shown in Figure 4, for single point of access to grid-enabled resources. The web server is setup with the Redhat Linux 7.3 operating system and Apache 1.3.26 HTTP server. The Apache server is an open-source HTTP server for UNIX and Windows and it provides a secure, efficient, and extensible platform for the grid web portal. The web portal is designed with Macromedia™ MX, Macromedia™ Homesite, HTML, Java script and PHP scripting language software. The Java script is used for client side form verification and user support functionality. PHP 4.2.3 is configured with the Apache HTTP server for designing the web portal template pages and the creation of dynamic web pages easily and quickly. Furthermore, the PHP code does not have to be compiled as it is interpreted "on-the-fly" by the web server. This results in web pages that can be highly customized for an authenticated user as they browse the web portal for resources.

During the proof-of-concept grid-enabling of *SnB*, it was evident that a fine grained authentication approach would be needed to manage several of the scientific applications. For example, researchers developing the *SnB* application may indeed wish to have all authenticated grid users be able to execute jobs and would like all documentation and database information available. Conversely, other researchers may have proprietary applications (data) that require the computational and data resources made available by the grid, but would like to limit the job execution access to selected departments within the University or possibly researchers located at remote locations. This can be accomplished by creating grid user profiles that contain the users standard information uid, name, organization, address, etc. and more specific information such as group id and access level information, as shown in Figure 5. This information is stored in a database and can be accessed through a PHP initiated query. MySQL 3.23.52 provides the speed and reliability required for this task and it is currently being used as the web portal database provider.

Since there is much more information required for a grid user to effectively use the web portal, the software package PHPMyAdmin 2.3.2 is also installed on the web server. This tool can manage an entire MySQL-server or just a single database from a web interface and perform the following tasks quickly and easily:

- create and drop databases
- create, copy, drop and alter tables
- delete, edit and add fields
- execute any SQL-statement, even batch-queries
- manage keys on fields
- load text files into tables
- create and read dumps of tables
- export and import data to CSV values
- administer multiple servers and single databases
- check referential integrity
- create complex queries automatically connecting required tables
- create PDF graphics of your database layout

The Globus Toolkit 2.2 [18,25] was installed on the web server and its Monitoring and Discovery Service (MDS) is used for monitoring the grid computational resources and the web portal server. The MDS includes a Grid Resource Information Service (GRIS) [18] that obtains platform information and Grid Index Information Service (GIIS) [18] that aggregates the information for an entire grid. MDS provides the necessary tools to build an LDAP-based information infrastructure for the computational grid and OpenLDAP is used to host the grid resource information directory. The LDAP server provides the following types of data:

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

The web portal GIIS aggregates the LDAP-based information of all the grid computational resources and can be monitored remotely. The Softerra LDAP Browser 2.4 is used for remotely viewing this information as it provides a user-friendly interface. This software is a lightweight version of Softerra LDAP Administrator and can be obtained as freeware. The web portal dynamic PHP web pages can also query and display the LDAP-based information for the grid users eliminating the need to use a secondary LDAP browser interface.

There is currently a migration from Globus Toolkit 2.2 to Globus Toolkit 3.0 [18] for projects that have a lifetime of more than one year and that do not have significant dependence on existing grid infrastructure provided Globus Toolkit 2.2. Since this is the case for the CCR-Grid, several additions will be made to the current web portal software suite in order to incorporate the new Globus Toolkit. Globus Toolkit 3.0 is based on the Open Grid Service Architecture (OGSA) [23] mechanisms, which address architectural issues related to the requirements and interrelationships of Grid Services. A Grid Service is a Web Service that conforms to a set of conventions for controlled, fault resilient and secure management.

The Grid Service clients can be written in any language that has bindings to the Web Services Description Language (WSDL) [24]. The Web Services Description Language for Java Toolkit (WSDL4J) 1.0 allows the creation, representation, and manipulation of WSDL documents describing services. The Hosting Environment that can provide the core capabilities that allow clients to connect to services, service invocation, and resource management is Apache/AXIS 1.1. Apache AXIS is an implementation of the SOAP ("Simple Object Access Protocol") submission to W3C ("World Wide Web Consortium"). SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts.

- SOAP is an envelope that defines a framework for describing what is in a message and how to process it,
- a set of encoding rules for expressing instances of application-defined datatypes, and
- a convention for representing remote procedure calls and responses.

An XML parser in Java called Xerces2 2.2.1 is needed for parsing and generating XML documents for Apache AXIS and is also used during development of the web portal extensively.

Apache Ant 1.5.1 is a Java-based build tool for developing software across multiple platforms and J2EE™ 1.4 are required for Open Grid Service Infrastructure (OGSI) [18] which addresses detailed specifications of the interfaces that a service must implement. J2EE bases them on standardized, modular components, by providing a complete set of services to those components, and by handling many details of application behavior automatically, without complex programming. The J2EE platform is the foundation technology of the Sun ONE platform.

The web portal dynamic PHP web pages that extract information from the LDAP server will now use a collection of classes and resources to process XML with PHP as the Globus Toolkit 3.0 Index Service Data is available only in XML form. There is no need for a separate GRIS as OGSA standard service data elements and FindServiceData interfaces enable any Globus Toolkit 3.0 OGSA service to act as their own GRIS. The Globus Toolkit 3.0 Index Service provides an information aggregation service that is more extensible than the Globus Toolkit 2.2 GIIS, furthermore the Index service and the GIIS are not compatible.

Note the web portal development integrates several software packages and toolkits in order to produce a robust and sustainable system flexible enough to incorporate many different scientific and engineering applications. Each grid-enabled application can be used by all grid users and also use all of the available grid resources or be restricted to a very well defined subset of grid users and resources. The ability to control these aspects is very important as performance prediction and load balancing of grid-enabled scientific applications on the grid can be very challenging, especially when wide ranges of computational power and network latency exist.

CCR Data Grid Design

Early in 2002, the Center for Computational Research developed a Storage Area Network (SAN) to meet critical ubiquitous storage and backup requirements in the Center. An overview of this HP/Compaq/DEC SAN is shown in Figure 6. The system hardware includes 24 TB of native disk storage, 24 HP Alpha servers running a Tru64 Sierra Cluster, 170 TB native (340 TB compressed) tape storage over two 595 slot, and 8 drive SDLT tape libraries. The system is designed to exceed 2.5GB/sec streaming I/O from a Fibre Channel infrastructure capable of delivering 9.6GB/sec of I/O and 777.6GB/hour native (1.555TB/hour compressed) of tape backup throughput. It is expected that this system will be fully operational in 1H03.

Due to the extensive delay in acquiring and implementing this SAN, the data infrastructure requirements for the CCR-Grid are presently being met by the following alternative distributed resource specific solutions:

- a 48 GB General Parallel File System (GPFS) served by the IBM SP2
- a 1 TB Parallel Virtual File System (PVFS) served by a Dell Xeon Cluster
- a 0.5 TB Networked File System (NFS) served by a Dell Xeon 6600 Server
- a 16 TB Redhat Advanced Server file system served by a Dell Xeon Cluster
- a STOREDGE L400 with two 20 GB drives, 21 slots, 21 GB/hour native served by a Solaris E250

Plans for designing and implementing a data grid infrastructure incorporating the new SAN and the current Center hardware are currently underway.

CCR Grid Collaboration Expansion

Several interdisciplinary research projects have successfully used the grid as a tool in order to take advantage of dynamic databases and a variety of compute platforms around the world. In addition to supporting critical applications that can take advantage of a grid, another major focus of the CCR-Grid project is to foster collaborations between researchers at geographically distributed locations. Geographically distributed scientific collaborations are supported by the Access Grid (http://www.accessgrid.org/agdp/) for group-to-group interactions, as well as by a computational grid that will allow scientists to work together in a tightly integrated fashion. Some of the current projects are summarized below, where the geography is shown in Figure 7.

- UCLA, Prakashan Korambath, High Performance Computing Consultant, Academic Technology Services
  - CCR Grid and UCLA Sun Grid connectivity study investigation
    - network latency
    - data Migration Bandwidth
    - grid Load Balancing
    - grid Performance Monitoring
    - interoperability of Globus Toolkit and Sun Grid Engine
    - grid-enabling of scientific and engineering applications
- Hauptman-Woodward Medical Research Institute (HWI), Russ Miller Ph.D., Senior Research Scientist
  - CCR Grid and HWI grid-enabling of *Shake-and-Bake* application [19-22]
    - automated grid application tuning
    - grid-based Data Mining
    - grid-based data warehousing
    - virtual collaboration
    - Genetic Algorithm optimization of grid-enabled *SnB*
- Johns Hopkins University (JHU), Roger Ghanem Ph.D., Director, Center for Uncertainty Analysis and Management
  - Optimal Structural System Design [33]
    - Risk Assessment
    - Risk Mitigation
    - grid-enabled computational models
    - data warehousing and migration
- University of Iowa (UIOWA), Keri C. Hornbuckle, Ph.D., Associate Professor, Department of Civil and Environmental Engineering
  - Fate and transport of persistent orga nic pollutants in nature systems
    - Lake Michigan Mass Balance [32]
    - Lake Michigan Mass Balance lake-wide analysis
    - Lake Michigan atmospheric PCB fate and transport [31,32]
    - grid-enabled atmospheric PCB congener analysis

Future Directions

In the 1980s, the National Science Foundation created the NSFnet that was intended to give scientific researchers easy access to its new supercomputer centers. From this modest beginning, smaller networks merged with NSFnet again and to form what is now known as the Internet. Computational, group-to-group, and data grids are forming in a very similar fashion and are poised to have the same explosion of connectivity. Grid services are paving the way for a broad based organized e-commerce, educational, scientific and data grid framework that can easily span the globe. Soon, the dependence of hard-wired interconnected grid resources will even give way to wireless highly available Grid Services that are accessed through dynamic web portals.

Acknowledgments

Support was provided by the Center for Computational Research at the University at Buffalo and the National Science Foundation under award ACI-0204918.

Bibliography

[1] Louis H. Turcotte, *"A Survey of Software Environments for Exploiting Networked Computing Resources"*, Engineering Research Center for Computational Field Simulation, Mississippi State, MS, June 1993.

[2] A.Geist, A Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam, PVM – Parallel Virtual Machine: A User's Guide and Tutorial For Networked Parallel Computing, MIT Press, Cambridge, MA,1998.

[3] Message Passing Interface Forum. MPI: A message-passing interface standard. Int. J. Supercomput. Appl. And High Performance Comput., Special Issue on MPI 8, ¾ (1994).

[4] Message Passing Interface Forum 1996. MPI-2: Extensions to the Message-Passing Interface, Technical Report, University of Tennessee, Knoxville, 1996.

[5] LAM / MPI Parallel Computing, 2002. http://www.lam-mpi.org

[6] Sun History of the Grid 2002. http://www.sun.co.nz/2002-0708/grid/history.html

[7] R. Butler and E. Lusk. Monitors, messages, and clusters: The p4 parallel programming system. Technical Report Preprint MCS-P362-0493, Argonne National Laboratory, Argonne, IL, 1993.

[8] The History of the Development of Parallel Computing, 1994. http://ei.cs.vt.edu/~history/Parallel.html

[9] Nirav H. Kapadia and Jose A. B. Fortes, PUNCH: An architecture for Web-enabled wide-area network-computing, Cluster Computing 2, 1999, pp. 153-164. http://punch.purdue.edu/

[10] Andrew S. Grimshaw, William A. Wulf, James C. French, Alfred C. Weaver, Paul F. Reynolds Jr. UVa CS Technical Report CS-94-20, June 8, 1994

[11] The Avaki Corporation, 2002. http://www.avaki.com/company/history.html

[12] Michael Litzkow, Miron Livny, and Matt Mutka, "Condor - A Hunter of Idle Workstations", *Proceedings of the 8th International Conference of Distributed Computing Systems,* pages 104-111, June, 1988. http://www.cs.wisc.edu/condor/

[13] Michael Litzkow, "Remote Unix - Turning Idle Workstations into Cycle Servers", *Proceedings of Usenix Summer Conference*, pages 381-384, 1987. http://www.cs.wisc.edu/condor/

[14] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steven Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", *Journal of Cluster Computing* volume 5, pages 237-246, 2002. http://www.cs.wisc.edu/condor/condorg/

[15] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steven Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)* San Francisco, California, August 7-9, 2001. http://www.cs.wisc.edu/condor/condorg/

[16] The Cactus Code, 2002. http://www/cactuscode.org/

[17] NPACI & SDSC Online, Volume 6, Issue 14, July 10, 2002. http://www.npaci.edu/online/

[18] The Globus Project, 2002. http://www.globus.org/

[19] H.A. Hauptman, H. Xu, C.M. Weeks, and R. Miller, Exponential *Shake-and-Bake*: theoretical basis and applications, *Acta Crystallographica* **A55**, 1999, pp. 891-900.

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

[21] C.M. Weeks and R. Miller, Optimizing *Shake-and-Bake* for proteins, *Acta Crystallographica* **D55**, 1999, pp. 492-500.

[22] R. Miller, S.M. Gallo, H.G. Khalak, and C.M. Weeks, *SnB*: Crystal structure determination via *Shake-and-Bake*, *Journal of Applied Crystallography* (1994), 27, pp. 613-621.

[23] I. Foster et al., “The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration,” tech. report, Glous Project; http://www.globus.org/research/papers/ogsa.pdf (current June2002).

[24] E. Christensen et al., “Web Services Description Language (WSDL) 1.1,” W3C Note, 15 Mar. 2001; http://www.w3.org/TR/wsdl (current June 2002).

[25] I. Foster and C. Kesselman, “Globus: A Toolkit-Based Grid Architecture,” *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, eds., Morgan Kaufmann, San Francisco, 1999, pp. 259-278.

[26] A.S. Grimshaw and W.A. Wulf, “The Legion Vision of a Worldwide Virtual Computer,” *Comm. ACM*, vol. 40, no. 1, 1997, pp. 39-45.

[27] Open Portable Batch System, 2002. http://www.openpbs.org/

[28] Portable Batch System Professional, 2002.http://www.pbspro.com/

[29] IBM Loadleveler, 2002.
http://www-1.ibm.com/servers/eserver/pseries/library/sp_books/loadleveler.html

[30] Platform LSF 5, 2002. http://www.platform.com/products/wm/LSF/index.asp

[31] Miller, SM, Green, ML, DePinto, JV, Hornbuckle, KC. "Results from the Lake Michigan Mass Balance study: Concentrations and fluxes of atmospheric polychlorinated biphenyls and trans-nonachlor", Environ. Sci. Technol., 35, 2001, pp. 278-285.

[32] Green, ML, Depinto, JV, Sweet, C, Hornbuckle, KC. "Regional spatial and temporal interpolation of atmospheric PCBs: Interpretation of Lake Michigan mass balance data", Environ. Sci. Technol., 34, 2000, pp 1833-1841.

[33] Ghanem, R. and Sarkar, A., ``Mid-frequency structural dynamics with parameter uncertainty," *Computer Methods in Applied Mechanics and Engineering,* Vol. 191, pp. 5499-5513, 2002.

[34] Alt, M., Bischof, H. and Gorlatch, S., “Program development for computational Grids using skeletons and performance prediction,” Parallel Processing Letters, Vol. 12, No. 2 (2002) pp. 157-174

[35] Gunter Ahrendt Purchasing Consulting, 2002. http://www.gapcon.com/listg.html

[36] Gentzsch, Wolfgang, “Response to Ian Foster’s “What is the Grid?””, GRID Today, Vol. 1, No. 8, August 5, 2002

[37] Global Grid Forum, 2002. http://www.gridforum.org/

[38] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke. “Data Management and Transfer in High Performance Computational Grid Environments”. *Parallel Computing Journal*, Vol. 28 (5), May 2002, pp. 749-771.

[39] W. Allcock, J. Bresnahan, I. Foster, L. Liming, J. Link, P. Plaszczac. “GridFTP Update January 2002”. *Technical Report, January 2002*.

[40] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, L. Liming, S. Meder, S. Tuecke. "GridFTP Protocol Specification". *GGF GridFTP Working Group Document,* September 2002.

[41] J. Bester, I. Foster, C. Kesselman, J. Tedesco, S. Tuecke. GASS: A Data Movement and Access Service for Wide Area Computing Systems**.** *Sixth Workshop on I/O in Parallel and Distributed Systems*, May 5, 1999.

[42] S. Vazhkudai, S. Tuecke, I. Foster. "Replica Selection in the Globus Data Grid"**.** *Proceedings of the First IEEE/ACM International Conference on Cluster Computing and the Grid (CCGRID 2001),* pp. 106-113, IEEE Computer Society Press, May 2001.

[43] D. Angulo, I. Foster, C. Liu, and L. Yang. "Design and Evaluation of a Resource Selection Framework for Grid Applications"**.** *Proceedings of IEEE International Symposium on High Performance Distributed Computing (HPDC-11),* Edinburgh, Scotland, July 2002.

[44] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, S. Tuecke; "SNAP: A Protocol for Negotiating Service Level Agreements and Coordinating Resource Management in Distributed Systems". *8th Workshop on Job Scheduling Strategies for Parallel Processing,* Edinburgh, Scotland, July, 2002.

[45] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman. "Grid Information Services for Distributed Resource Sharing"**.** *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, IEEE Press, August 2001.

[46] K. Keahey, V. Welch, "Fine-Grain Authorization for Resource Management in the Grid Environment"**.** *Proceedings of Grid2002 Workshop*, 2002.

[47] L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke. "A Community Authorization Service for Group Collaboration"**.** *Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002.

[48] G. von Laszewski, I. Foster, J. Gawor, A. Schreiber, C. Pena. "InfoGram: A Grid Service that Supports Both Information Queries and Job Execution"**.** *Proceedings of the 11th IEEE International Symposium on High-Performance Distributed Computing (HPDC-11)*, IEEE Press, Edinburg, Scotland, July 2002.

[49] I. Foster, N. Karonis. "A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems"**.** *Proc. 1998 SC Conference*, November, 1998.

[50] B. de Supinski, N. Karonis. "Accurately Measuring MPI Broadcasts in a Computational Grid"**.** *Proc. 8th IEEE Symp. on High Performance Distributed Computing*, pp. 29-37, August 1999.

[51] I. Foster, N. Karonis, C. Kesselman, G. Koenig, S. Tuecke. "A Secure Communications Infrastructure for High-Performance Distributed Computing"**.** *6th IEEE Symp. on High-Performance Distributed Computing*, pp. 125-136, 1997.

Parallel Language	Description
P4	P4 is a library of macros and subroutines developed at Argonne National Laboratory for programming a variety of parallel machines. The p4 system supports both the shared-memory model (based on monitors) and the distributed-memory model (using message-passing) [1,7].
Express	The Express system is a set of libraries for communication, I/O, and parallel graphics. Extended I/O routines enable parallel input and output, and a similar set of routines are provided for graphical displays from multiple concurrent processes. The toolkit is developed and marketed commercially by ParaSoft Corporation [1].
LINDA	LINDA is a concurrent programming model that has evolved from a Yale University research project. The primary concept in Linda is that of a ``tuple-space'', an abstraction via which cooperating processes communicate. This central theme of Linda has been proposed as an alternative paradigm to the two traditional methods of parallel processing: that based on shared memory, and that based on message passing [1].

Table 1: Parallel Computing Languages.

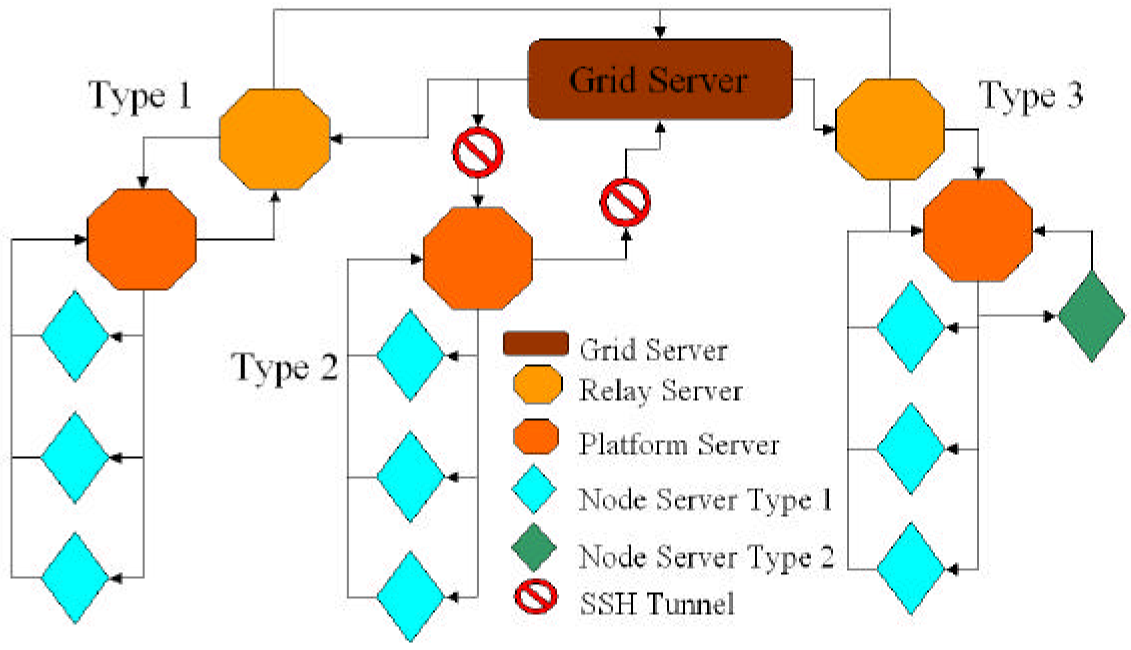
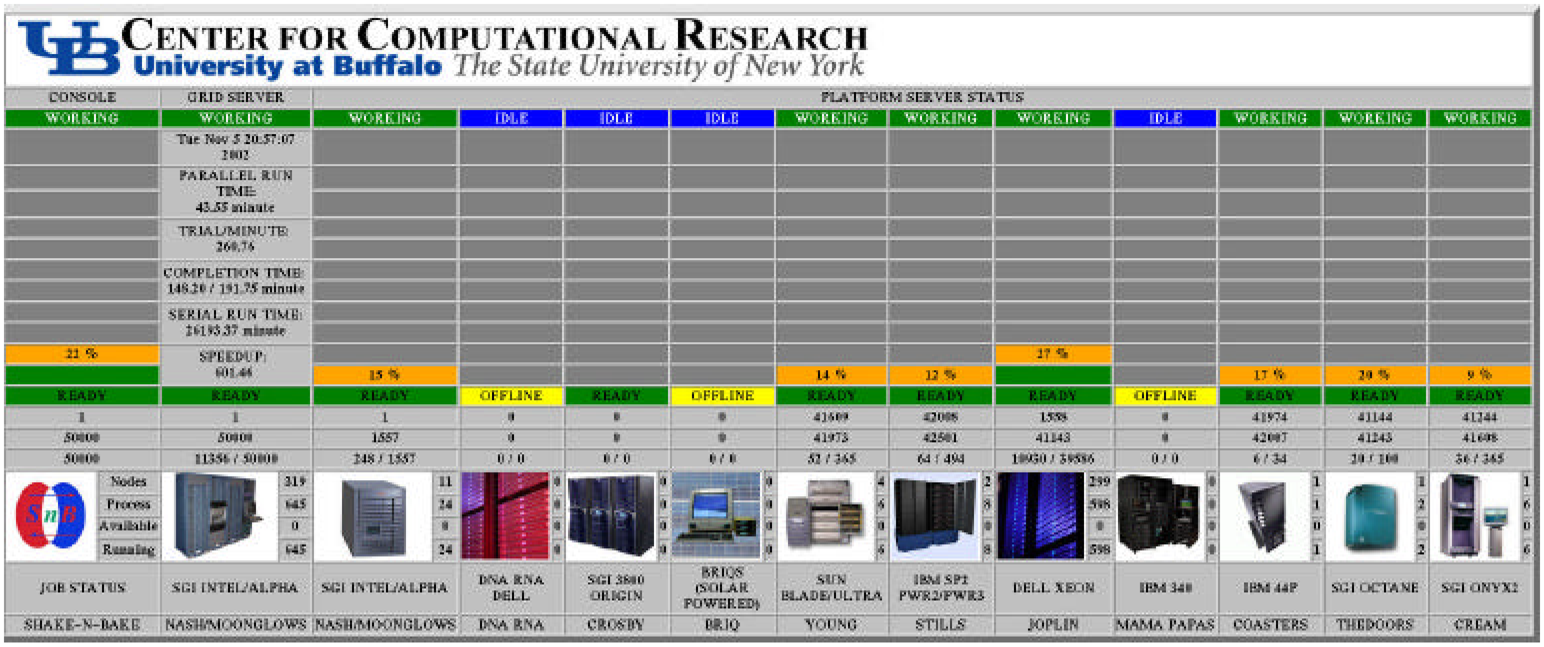


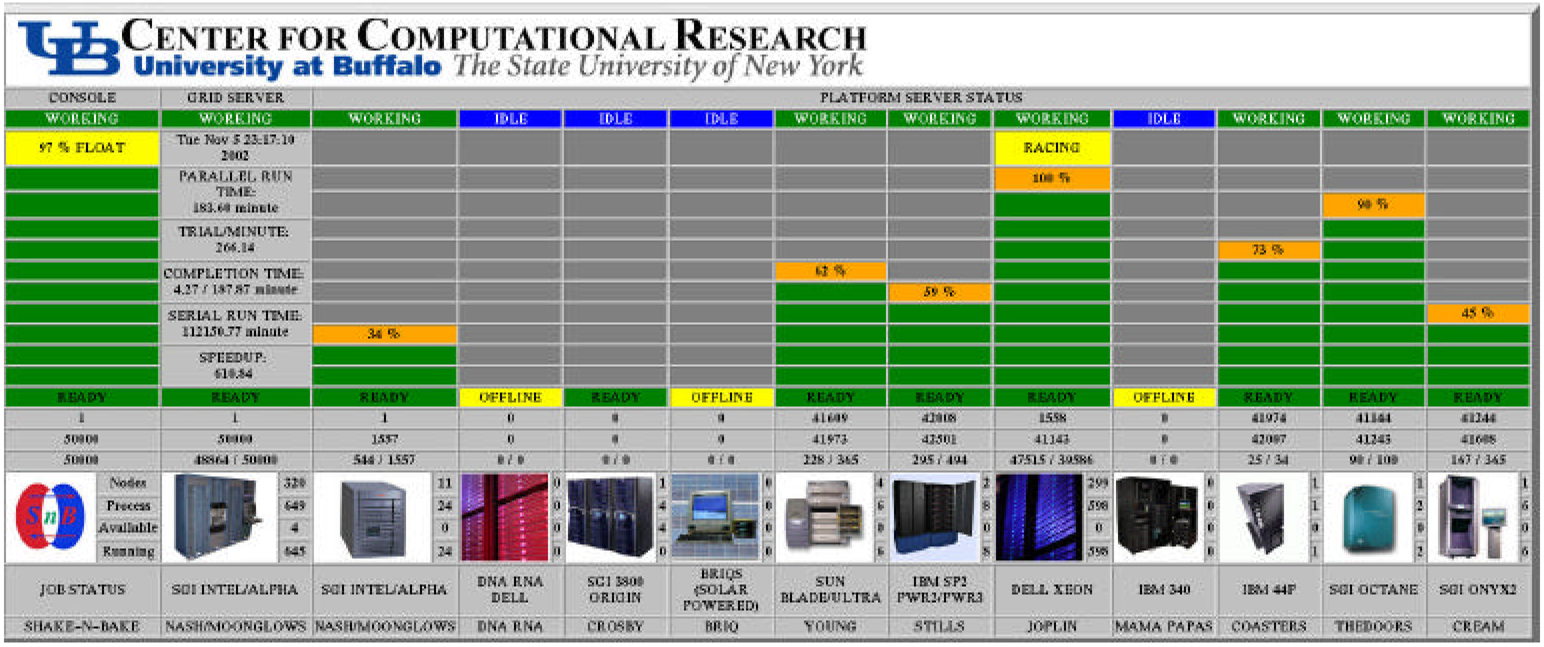
Figure 1: Client/Server configuration types for grid-enabled *Shake-and-Bake* application.



CENTER FOR COMPUTATIONAL RESEARCH
University at Buffalo The State University of New York

CONSOLE	GRID SERVER	PLATFORM SERVER STATUS										
WORKING	WORKING	WORKING	IDLE	IDLE	IDLE	WORKING	WORKING	WORKING	IDLE	WORKING	WORKING	WORKING
	Tue Nov 5 20:57:07 2002											
	PARALLEL RUN TIME: 43.55 minute											
	TRIAL/MINUTE: 260.76											
	COMPLETION TIME: 148.20 / 191.75 minute											
	SERIAL RUN TIME: 26193.37 minute											
21 %	SPEEDUP: 601.46							27 %				
		15 %				14 %	12 %			17 %	20 %	9 %
READY	READY	READY	OFFLINE	READY	OFFLINE	READY	READY	READY	OFFLINE	READY	READY	READY
1	1	1	0	0	0	41609	42008	1558	0	41974	41144	41244
50000	50000	1557	0	0	0	41973	42501	41143	0	42007	41243	41608
50000	11356 / 50000	248 / 1557	0 / 0	0 / 0	0 / 0	52 / 365	64 / 494	10030 / 39586	0 / 0	6 / 34	20 / 100	36 / 365
Nodes	319	11	0	0	0	4	2	299	0	1	1	1
Process	645	24	0	0	0	6	8	598	0	1	2	6
Available	0	0	0	0	0	0	0	0	0	0	0	0
Running	645	24	0	0	0	6	8	598	0	1	2	6
JOB STATUS	SGI INTEL/ALPHA	SGI INTEL/ALPHA	DNA RNA DELL	SGI 3800 ORIGIN	BRIQS (SOLAR POWERED)	SUN BLADE/ULTRA	IBM SP2 PWR2/PWR3	DELL XEON	IBM 340	IBM 44P	SGI OCTANE	SGI ONYX2
SHAKE-N-BAKE	NASH/MOONGLOWS	NASH/MOONGLOWS	DNA RNA	CROSBY	BRIQ	YOUNG	STILLS	JOPLIN	MAMA PAPAS	COASTERS	THEDOORS	CREAM

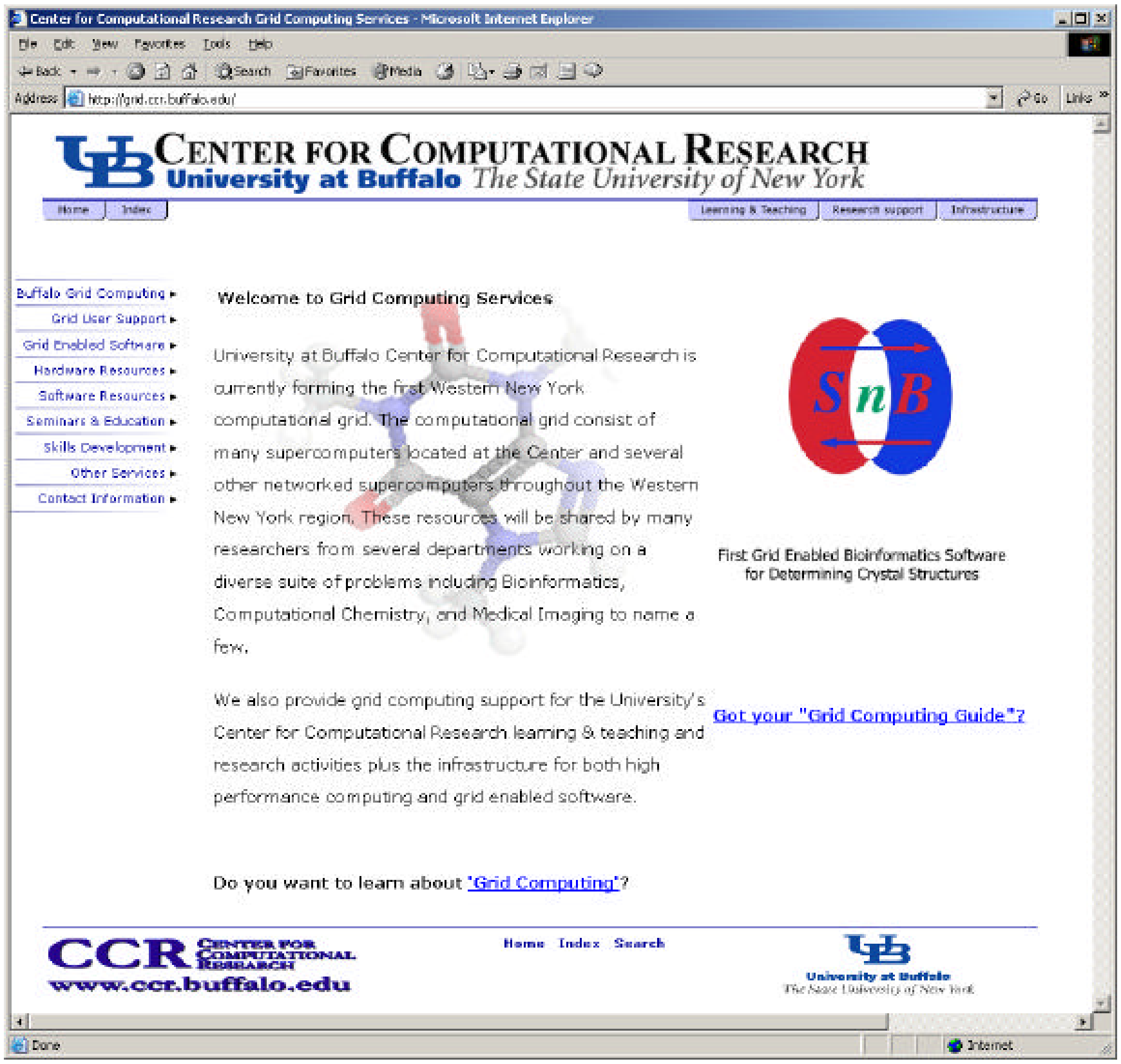
Figure 2: Dynamic HTML front-end for grid-enabled *Shake-and-Bake* application.



CENTER FOR COMPUTATIONAL RESEARCH
University at Buffalo The State University of New York

CONSOLE	GRID SERVER	PLATFORM SERVER STATUS										
WORKING	WORKING	WORKING	IDLE	IDLE	IDLE	WORKING	WORKING	WORKING	IDLE	WORKING	WORKING	WORKING
97 % FLOAT	Tue Nov 5 23:17:10 2002							RACING				
	PARALLEL RUN TIME: 183.60 minute							100 %				
											90 %	
	TRIAL/MINUTE: 266.14											
										73 %		
	COMPLETION TIME: 4.27 / 187.87 minute					62 %						
							59 %					
	SERIAL RUN TIME: 112150.77 minute											45 %
		34 %										
	SPEEDUP: 610.84											
READY	READY	READY	OFFLINE	READY	OFFLINE	READY	READY	READY	OFFLINE	READY	READY	READY
1	1	1	0	0	0	41609	42008	1558	0	41974	41144	41244
50000	50000	1557	0	0	0	41973	42501	41143	0	42097	41245	41608
50000	48864 / 50000	544 / 1557	0 / 0	0 / 0	0 / 0	228 / 365	295 / 494	47515 / 39586	0 / 0	25 / 34	90 / 100	167 / 365
Nodes	320	11	0	1	0	4	2	299	0	1	1	1
Process	649	24	0	4	0	6	8	598	0	1	2	6
Available	4	0	0	4	0	0	0	0	0	0	0	0
Running	645	24	0	0	0	6	8	598	0	1	2	6
JOB STATUS	SGI INTEL/ALPHA	SGI INTEL/ALPHA	DNA RNA DELL	SGI 3800 ORIGIN	BRIQS (SOLAR POWERED)	SUN BLADE/ULTRA	IBM SP2 PWR2/PWR3	DELL XEON	IBM 340	IBM 44P	SGI OCTANE	SGI ONYX2
SHAKE–N–BAKE	NASH/MOONGLOWS	NASH/MOONGLOWS	DNA RNA	CROSBY	BRIQ	YOUNG	STILLS	JOPLIN	MAMA PAPAS	COASTERS	THEDOORS	CREAM

Figure 3: Dynamic HTML front-end for grid-enabled *Shake-and-Bake* application load balancing and fault tolerance.


Center for Computational Research Grid Computing Services - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://grid.ccr.buffalo.edu/
Go Links
CENTER FOR COMPUTATIONAL RESEARCH
University at Buffalo The State University of New York
Home Index
Learning & Teaching Research support Infrastructure
Buffalo Grid Computing
Grid User Support
Grid Enabled Software
Hardware Resources
Software Resources
Seminars & Education
Skills Development
Other Services
Contact Information
SnB
Home Index Search
CCR CENTER FOR COMPUTATIONAL RESEARCH
www.ccr.buffalo.edu
University at Buffalo
The State University of New York
Done
Internet

Welcome to Grid Computing Services

University at Buffalo Center for Computational Research is currently forming the first Western New York computational grid. The computational grid consist of many supercomputers located at the Center and several other networked supercomputers throughout the Western New York region. These resources will be shared by many researchers from several departments working on a diverse suite of problems including Bioinformatics, Computational Chemistry, and Medical Imaging to name a few.

We also provide grid computing support for the University's Center for Computational Research learning & teaching and research activities plus the infrastructure for both high performance computing and grid enabled software.

First Grid Enabled Bioinformatics Software for Determining Crystal Structures

Got your "Grid Computing Guide"?

Do you want to learn about 'Grid Computing'?

Figure 4: University at Buffalo Center for Computational Research Grid Web Portal.

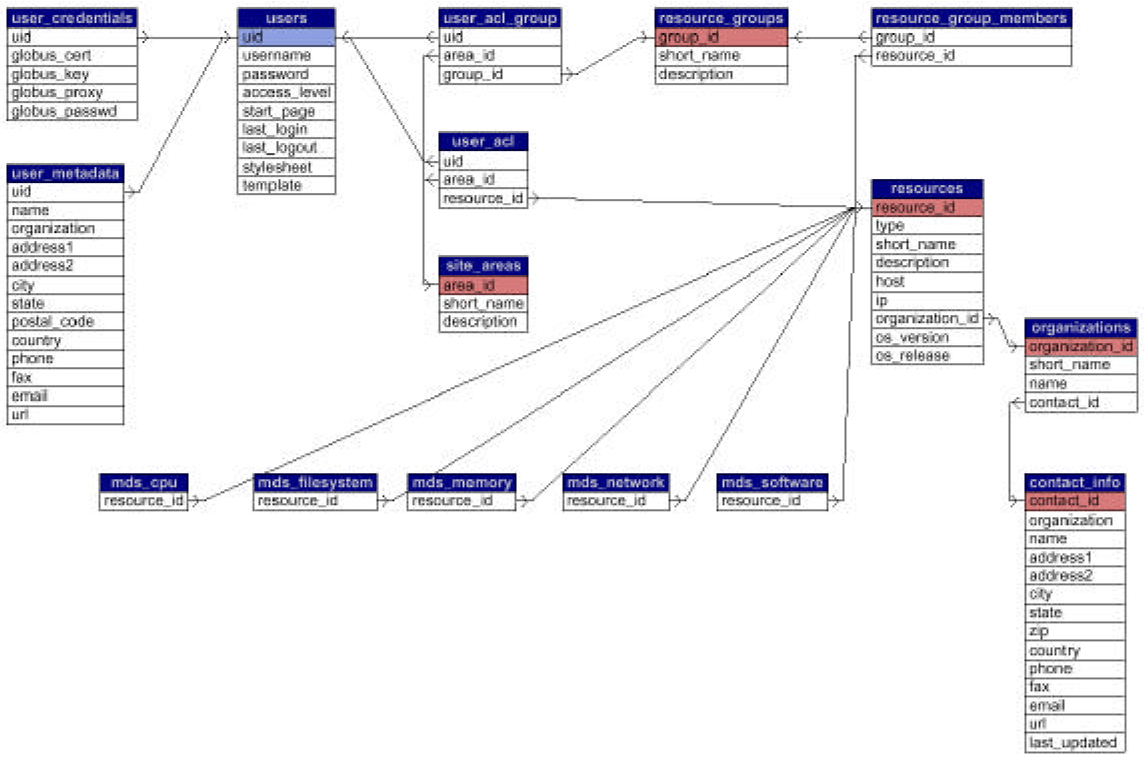


Figure 5: Grid portal CCR Grid database schema (Courtesy of Steven M. Gallo, Database Administrator, Center for Computational Research).

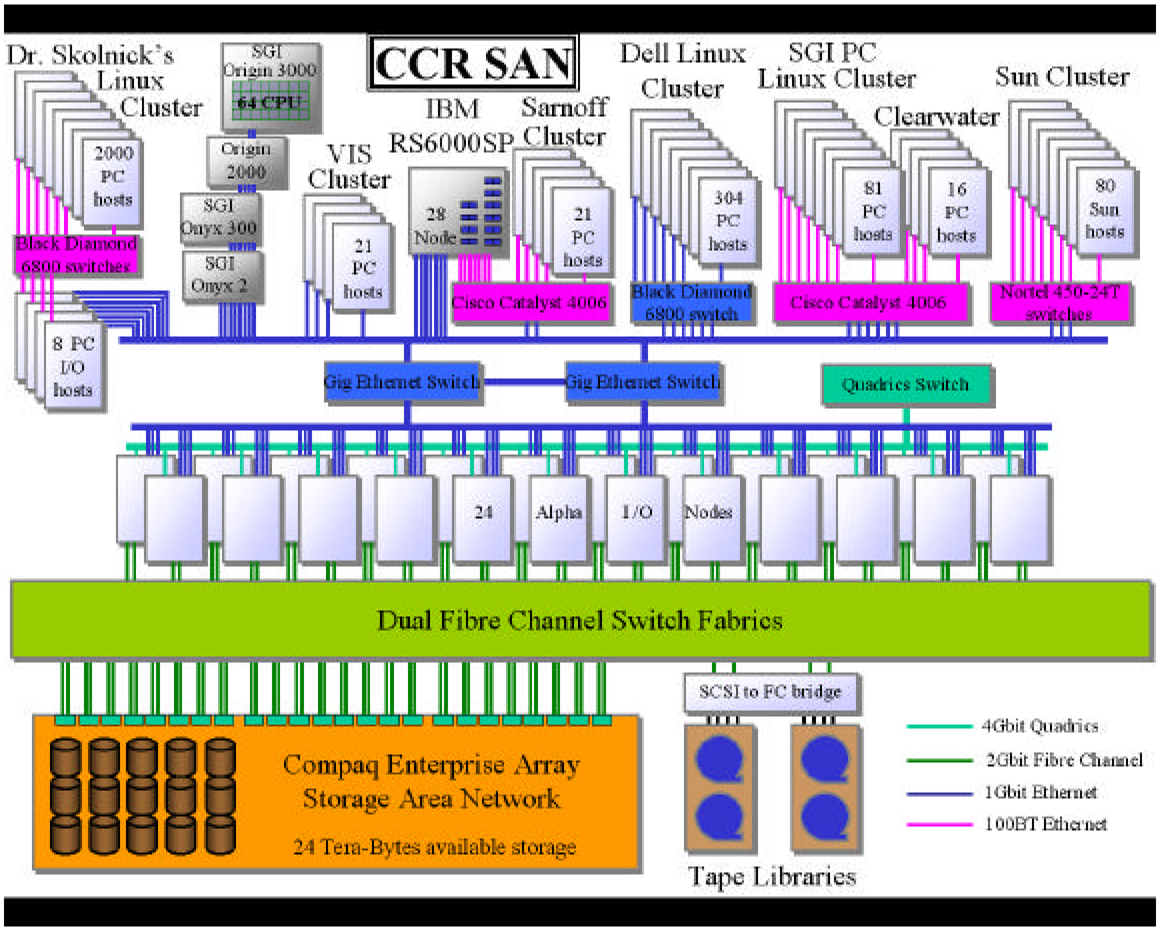

CCR SAN
Dr. Skolnick's Linux Cluster
2000 PC hosts
Black Diamond 6800 switches
8 PC I/O hosts
SGI Origin 3000
64 CPU
Origin 2000
SGI Onyx 300
SGI Onyx 2
VIS Cluster
21 PC hosts
IBM RS6000SP
28 Node
Sarnoff Cluster
21 PC hosts
Cisco Catalyst 4006
Dell Linux Cluster
304 PC hosts
Black Diamond 6800 switch
SGI PC Linux Cluster
81 PC hosts
Clearwater
16 PC hosts
Cisco Catalyst 4006
Sun Cluster
80 Sun hosts
Nortel 450-24T switches
Gig Ethernet Switch
Gig Ethernet Switch
Quadrics Switch
24 Alpha I/O Nodes
Dual Fibre Channel Switch Fabrics
Compaq Enterprise Array Storage Area Network
24 Tera-Bytes available storage
SCSI to FC bridge
Tape Libraries
4Gbit Quadrics
2Gbit Fibre Channel
1Gbit Ethernet
100BT Ethernet

Figure 6: Center for Computational Research Storage Area Network design and integration with CCR Grid computational resources (Courtesy of Tony Kew, Storage Area Network Administrator, Center for Computational Research).

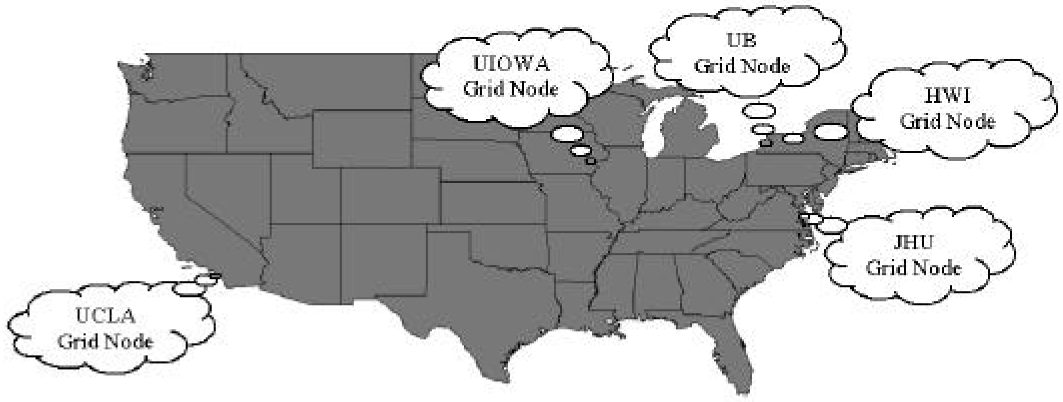


Figure 7: Buffalo-CCR proposed collaborative computational and data grid.