EFFECTIVELY MANAGING DATA ON A GRID

Catherine L. Ruby and Russ Miller
Center for Computational Research
Department of Computer Science and Engineering
State University of New York, Buffalo, NY 14260
{*clruby, miller*}*@ccr.buffalo.edu*

1 Introduction

Advances in parallel computing have resulted in the emergence of *grid computing*, a field that strives to integrate multiple distributed and heterogeneous resources, which are typically under distinct administrative domains, into a single, coherent system that can be efficiently and effectively used to solve complex and demanding problems in a fashion that is transparent to the end-user [1-3]. Such resources include compute systems, storage systems, sensors, visualization devices, rendering farms, imaging systems, and a wide variety of additional Internet-ready instruments. Grids of various sorts are quite fashionable and are being deployed in many fields, including structural biology, computational chemistry, cancer research, biomedical informatics, astronomy, environmental science, and high-energy physics, to name a few.

Computing with limited resources is a pervasive theme in the field of computational science and engineering, which typically focuses on large-scale problems that rely on simulation and modeling. We live in a digital, data driven society, where demands for the analysis, storage, maintenance, networking, and visualization of data is increasing at an extraordinary rate. Therefore, advances in scalable grid computing are critical to 21st century discovery.

Though inherently scalable in its distributed design, grid computing is in its infancy. That is, it is common for grids to operate under well-controlled environments that predominantly

include large computational clusters, data repositories, and high-end networking. Designing and deploying reliable data systems even within these controlled environments is an essential next step in terms of providing support for the ultimate ubiquitous grid.

A data grid consists of a geographically-distributed network of storage devices intended to house and serve data. Data grids provide the control and management of data used and produced by grid applications, and are often, but not always, closely coupled to computational grids [1, 4, 5]. Distributing files across multiple storage elements can present load issues as usage patterns change over time. Data grids are dynamic systems, and it is essential that as storage network configurations and data usage patterns evolve, load issues can be predicted, analyzed, and addressed to maintain storage network performance. Overcoming the complexities associated with high-utilization distributed data systems, through examining the relationship between data grid performance and its file utilization patterns and storage network architecture, is a key to delivering a reliable and robust data grid.

The *Advanced Computational Data Center Grid* (*ACDC Grid*) is a grid deployed through the *Center for Computational Research* (*CCR*) at the State University of New York at Buffalo (SUNY-Buffalo) [6, 7, 8, 9]. The ACDC Grid integrates several compute elements from within the center, as well as from other institutions across the country. This system supports users running a variety of grid applications, including the *Shake-and-Bake* (*SnB*) [10] application for molecular structure determination and the Princeton Ocean Model (POM) [11] for tracking harmful algal-blooms in the Great Lakes. The ACDC Grid supports many data-intensive grid applications, requiring the deployment of a reliable data grid solution to support the grid and enable continued growth [8].

In this chapter, we will discuss three initiatives that are designed to present a solution to

the data service requirements of the ACDC Grid. We present two data grid file utilization simulation tools, the *Scenario Builder* and the *Intelligent Migrator*, which examine the complexities of delivering a distributed storage network. We then discuss the *ACDC Data Grid*, currently in use in conjunction with the ACDC Grid, and the ways in which the Scenario Builder and Intelligent Migrator enable us to simulate the performance of the ACDC Data Grid over time so that the system can be appropriately tuned for performance.

Deploying a distributed storage element solution to a system such as the ACDC Grid requires an analysis of storage load as data is utilized for computational jobs. The Scenario Builder, developed in conjunction with the ACDC Grid, is a dynamic, web-based tool for creating, simulating, and analyzing virtual data grid environments in an effort to relate simulated file usage to potential production data grid scenarios and predict file utilization across a storage network over time. Through the Scenario Builder, highly-customizable data grid scenarios may be generated and simulated over variable lengths of time in order to examine the demand on the virtual data grid storage network and predict the behavior of a physical data grid solution under similar circumstances.

Data files that are used to execute computational jobs, such as configuration files or data sets for analysis, can present key challenges when located across a distributed storage system. Trafficking files between computational resources as users submit grid jobs can become a difficult task, and users who submit jobs repeatedly can use a significant amount of time and bandwidth to move their files to the correct locations before job execution. Improving on the design of the Scenario Builder, we present the Intelligent Migrator, a system that consists of routines to generate, simulate, and visualize proposed data grid scenarios in an effort to learn about projected patterns of data usage over time. This allows for the effective management and

replication of data to remote locations so that data files are present in advance of when they are needed. By studying the performance of the Intelligent Migrator on generated scenarios, we can evaluate how the learning procedures would perform in a similar physical data system.

The ACDC Data Grid is a data grid serving the needs of the ACDC Grid. In this chapter, we present the ACDC Data Grid, its internal storage infrastructure, and its user interfaces, which provide a seamless environment for file access over its distributed design. The ACDC Data Grid, implemented across several storage repositories within CCR and integrated with compute elements across Western New York, New York State, the Open Science Grid (OSG) [12] and TeraGrid [13], serves the data needs of the grid applications run on the ACDC Grid. The Scenario Builder and the Intelligent Migrator are based on the design and architecture of the ACDC Grid and enable the construction of tailored virtual data grid configurations for the examination of its potential performance in a wide variety of usage scenarios.

The development of data storage solutions is an essential component of ongoing grid research and advances in grid computing. Grid applications demonstrate a growing need for data services, and infrastructures for robust and reliable data grids are a necessity. The Scenario Builder, the Intelligent Migrator, and the ACDC Data Grid are three initiatives designed to meet the complex challenges of creating a distributed, heterogeneous storage framework and serve the ongoing needs of the ACDC Grid.

In this chapter, we will discuss the three data grid initiatives developed for the ACDC Grid. In Section 2, we review related research in the field of grid computing and present an overview of the ACDC Grid initiative. Section 3 discusses the Scenario Builder, both in its design and implementation, as well as in the tools developed for generating tailored data grid scenarios for simulation and visual analysis. We discuss the Intelligent Migrator in Section 4,

including the design of a virtual data grid, its simulation and algorithms for modeling user file utilization over time, and the visualization tools developed for the analysis of the underlying algorithm's performance. In Section 5, we present the design and implementation of the ACDC Data Grid, which is currently providing the data services for the ACDC Grid. In Section 6, we include our final thoughts and discuss the ways in which the Scenario Builder and Intelligent Migrator impact the design of the ACDC Data Grid and enable performance simulations.

2 Related Work

The Scenario Builder, Intelligent Migrator, and the ACDC Data Grid are part of a research initiative to develop critical grid infrastructure. In this section, we will present a background of computational and data grid initiatives. We will then present the Advanced Computational Data Center Grid (ACDC Grid) and the monitoring infrastructure implemented within the Center for Computational Research.

2.1 Grid Computing

As the scientific community embraces simulation and modeling as the 3rd component of research, complementing theory and experimentation, high-end computing has become a key to solving an ever expanding array of scientific questions. Scientific and engineering applications that rely on simulation and modeling in areas that include physics, chemistry, astronomy, biology, manufacturing, and life sciences, to the name a few, have produced a significantly increased need for computing, storage, networking, and visualization, creating the necessity for powerful compute resources and increased collaborations between research facilities. Founded on the principles of parallel computing and built on the maturing public Internet architecture over

the last ten years, grid computing has entered the mainstream as a viable solution to the growing needs of academic and corporate research ventures. Grid research strives to integrate geographically-dispersed sites to provide a single infrastructure for accessing compute and storage resources available throughout the world [3, 14].

Providing scalable and seamless access to distributed compute resources is a key component of grid infrastructure. Significant challenges include providing a viable solution for supporting potentially large and data-intensive applications between grid-enabled endpoints, given unpredictable bandwidth and connectivity limitations. Authentication and security are also critical issues when operating across administrative domains. Furthermore, common computational problems such as resource allocation and scheduling, which are present in a homogeneous computing environment, become more significant challenges in a heterogeneous and distributed setting [3].

Numerous grid initiatives have been spawned in the last several years in an attempt to overcome these challenges and provide a robust and scalable grid solution. Early grid research produced the concept of metacomputing, *i.e.*, integrating large, geographically dispersed supercomputing sites. Endeavors such as the information wide area year (I-WAY) experimented with linking high performance supercomputers by integrating existing high bandwidth networks. Further grid research yielded the second generation of the grid and the recognition of three main grid computing issues, namely (i) heterogeneity, or the ability to link machines with many different architectures or administrative policies, (ii) scalability, in order for grids to grow both in their size and in their support of the expanding scientific applications which require grid services, and (iii) the adaptability of the grid infrastructure to varying degrees of quality-of-service (QoS) between its heterogeneous and distributed computational components. This

second generation resulted in the creation and deployment of grid middleware, such as Globus [15], for authentication and data-transfer capabilities over distributed systems, and Legion [16], which provided seamless resource integration through an object oriented approach. The second generation of grid research also produced core technologies like the Common Object Request Broker Architecture (CORBA) [17] and Grid web portals [18].

The current focus is on developing grids that tie together the middleware discussed above with the overall mission of producing user-friendly grids. The concept of a Virtual Organization (VO) has been established in response to the goal of facilitating collaboration, in which rules regarding the access and sharing of computational resources are established as a response to the definition of a common academic or industrial goal [14]. The CCR, for example, established the Grid Resources for Advanced Science and Engineering (GRASE) VO for research in science, including, but not limited to, biology, chemistry, and engineering [19]. Grid initiatives currently take a more service-oriented approach, with an increased emphasis on enabling distributed collaboration and autonomic methods of failure detection and resolution [18].

2.2 Data Grid Initiatives

As the field of grid computing matures and grid applications evolve, the demand for efficient and reliable storage solutions is growing significantly. Many ongoing computational grid initiatives involve data-intensive applications. Scientific projects in many areas, including CERN's Large Hadron Collider (LHC), is projected to produce petabytes of data per year [3]. Grid applications are requiring and producing growing volumes of data, calling for better storage solutions to facilitate further advances in the field of grid computing [1].

A data grid is a grid solution designed to meet the data storage and service needs of a

computational grid by networking a series of distributed, heterogeneous compute elements to provide a reliable system for serving available data [20]. Data grids house and serve data to grid users, providing the necessary data services for their computational grid-based jobs. The distributed and dynamic nature of a data grid, and the heterogeneous nature of the storage elements that may belong to one of several administrative domains, can present complex challenges in ensuring the reliability and availability of a user's data files [1, 21].

A common approach to meeting these challenges is to provide virtualization services to better manage the data in the storage network. Common abstraction techniques include generating metadata for data files to facilitate straightforward retrieval from a knowledge repository. Others involve constructing a logical namespace for data in order to organize file attributes which may be stored in a database. Emulation and migration techniques serve to reproduce portions of data stored in the storage network and maintain the processes necessary for managing a digital entity. Other endeavors utilize the concept of data replication for managing grid data [1, 5]. These virtualization services attempt to alleviate the difficulties of retrieving data across arbitrarily large and heterogeneous data grids [21].

There are many examples of these virtualization services in grid initiatives around the world. The Storage Resource Broker [22], developed at the San Diego Supercomputing Center, is integrated with an Extensible Metadata CATalog System (MCAT) [23] to provide a logical namespace in a database for the files stored within its infrastructure, in order to easily apply file system operations as well as other extended capabilities. Similarly, the European DataGrid project (EDG) [24] maintains a mapping of global names to local names, and supports caching and streaming as well as data replication to multiple storage devices within the storage network to maintain high data availability [21]. The Grid Physics Network (GriPhyN) [25] utilizes the

concept of Virtual Data and the creation of Petascale Virtual-Data Grids (PVDGs) [26], which catalog data and virtualize information in order to serve needs of data-intensive grid applications [27]. These and other virtualization techniques are solutions to the challenges arising from creating robust and reliable data storage networks to serve data-intensive grid applications.

2.3 The ACDC Grid

The Advanced Computational Data Center Grid (ACDC Grid) [6] is an implementation of a computational grid infrastructure that integrates high-performance compute elements within the CCR with computational resources across the country. It encompasses computing elements with a variety of queue managers, operating systems and Wide Area Network (WAN) connections, and focuses on seamlessly integrating highly heterogeneous compute elements to facilitate computational research [8, 9, 28].

Figure 2.1: The ACDC Grid Web Portal.

The development of the ACDC Grid utilizes a series of Application Programming Interfaces (APIs) to create a robust architecture for integrating compute hardware with grid middleware and user interfaces to the job submission system. The implementation of the ACDC Grid also includes the development of Grid-enabling Application Templates (GATs) [29], which define the procedures for executing a scientific application in a grid environment. Authentication, data transfer and remote execution are supported by the Globus Toolkit [8, 9, 15, 28].

The ACDC Grid initiative encompasses the development of both the infrastructure for submitting and executing jobs, as well as the ACDC Grid Web Portal, the interface through which users access the ACDC Grid. Served by an Apache HTTP Server [30] and written in the PHP hypertext preprocessor scripting language [31] along with JavaScript [32], the web portal provides real-time MySQL [33] database access which supports job submission and portal architecture [8, 9]. Figure 2.1 shows the main ACDC Grid Web Portal page and the point of entry for ACDC Grid users.

The ACDC Grid is a general purpose grid and supports a wide variety of scientific applications. Grid applications include those in computational chemistry, structural biology, and environmental engineering, to name a few *Shake-and-Bake* (*SnB*) [10], for example, is one grid application run on the ACDC Grid that utilizes the grid infrastructure to execute computationally-intensive molecular structure determinations [8, 9, 28].

The ACDC Data Grid serves as the data storage network serving the grid users and grid applications run on the ACDC Grid infrastructure. Section 5 includes a detailed discussion of the design and implementation of this component and its service to the ACDC Grid.

2.4 The ACDC Grid Monitoring Infrastructure

Recent initiatives at the CCR have yielded an extensive monitoring system for grid systems around the world, including the ACDC Grid, the Open Science Grid (OSG) [12], and TeraGrid [13]. This lightweight grid monitoring system collects site functional and operational information for over 150 compute elements. Information is collected through a series of scripts designed to pull site environment information every few minutes from grid resources and use this as a bootstrap for collecting information, including the status of current running and queued jobs, the state of worker nodes, GridFTP [34] activity, bandwidth and network latency statistics, and a variety of other information. The data collected through this system provides the foundation for the historical job information and resource utilization statistics for the Intelligent Migrator,

Figure 2.2: The ACDC Grid Dashboard.

discussed in more detail in Section 4.

Data collected by the ACDC Grid Monitoring Infrastructure is aggregated into an online interface called the ACDC Grid Dashboard [35]. The main Grid Dashboard page provides an overview of the current status of a computational grid, job profiles, gatekeeper load information, GridFTP statistics, and resource storage load data. The presentation of this information is in the form of a variety of dynamic graphics. Clicking on any region of the Grid Dashboard brings the user to a new set of interactive plots with more detailed information on the grid environment, including more in-depth information on specific compute elements and historical information collected from the monitoring infrastructure. The main page of the ACDC Grid Dashboard is featured in Figure 2.2.

The ACDC Operations Dashboard [36], developed as an adjunct to the ACDC Grid Dashboard, provides an interactive and dynamic interface for viewing the operational status of compute elements available to grid users. The Operations Dashboard organizes operational status information in a Site Resource Service Matrix, showcasing the compute elements and their corresponding results from over 30 Site Functional Tests designed to test services offered by a compute element for the VOs it supports. These services range from simple connectivity tests and authentication verifications to the results of GridFTP tests and collecting and presenting virtual organization support information from configuration files. Implemented in Perl [37], and based heavily on the site verification script written by Dr. Craig Prescott of the University of Florida [38], these tests are designed to provide detailed information regarding the functionality of a variety of services on a compute element for the virtual organizations that are being monitored. Figure 2.3 shows a screenshot of the main ACDC Operations Dashboard page.

The Operations Dashboard provides an interactive subset of capabilities through Action

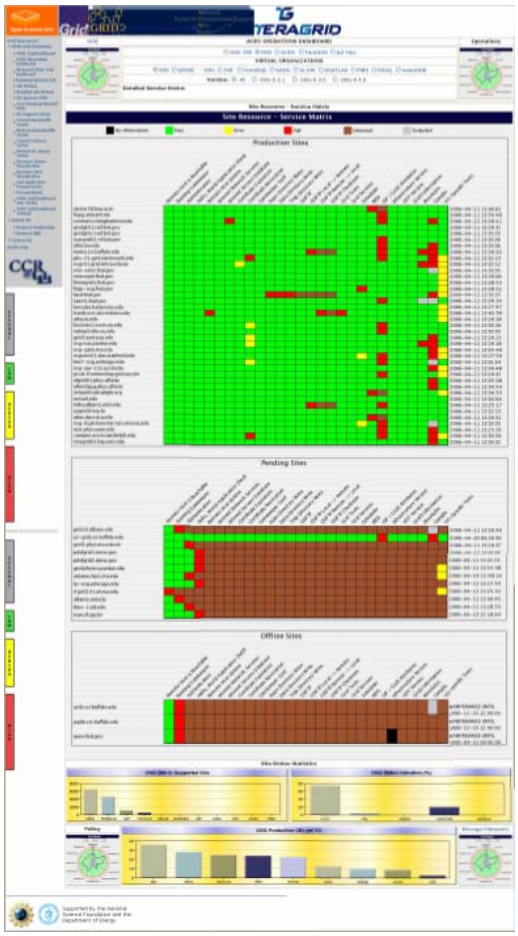


Figure 2.3: The ACDC Operations Dashboard.

Items. Organized in a multi-tiered authentication scheme using user certificates, these Action Items provide a pool of resources for administrators to detect, troubleshoot, and resolve operational issues on their compute elements. These and other popup displays are featured in Figure 2.4.

Work on the ACDC Grid Monitoring Infrastructure serves to provide a wide range of accurate, up-to-date monitoring tools and interactive interfaces to facilitate the management of compute elements in a widely distributed grid setting. The underlying data collection routines and dynamic front-end interfaces provide a collaborative and interactive environment for

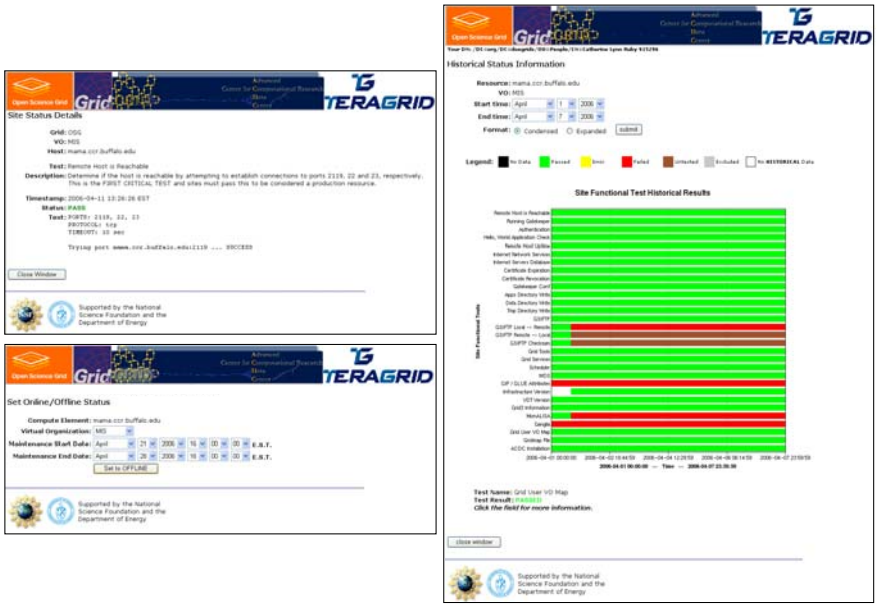


Figure 2.4: Select Action Items and popup displays from the ACDC Operations Dashboard.

publishing and responding to changes across monitored grid infrastructures. Plans have been made to apply the Operations Dashboard architecture to an ACDC Storage Elements Dashboard, which will track the functional status of machines dedicated as storage devices and running Storage Resource Managers (SRMs) [39], a grid middleware component for file management and storage space allocation.

3 The Scenario Builder

In an early effort to model data grid infrastructure and examine storage load as it relates to file utilization over time, the Scenario Builder was created as a dynamic web-based tool for constructing virtual data grid scenarios and simulating file accesses by grid users in order to explore how these would impact the performance of a distributed storage network. The Scenario Builder presents the user with a blank template for generating data grid usage scenarios. It contains simulation routines to model file accesses and the system will present the user with

records of bandwidth and patterns of file distribution based on a user defined scenario, so that the user may evaluate scalability issues that could arise in such a setting.

In this section, we present the design and implementation of the Scenario Builder and its fundamental components, the concept of a scenario, and the architecture of the virtual storage network. We then discuss the Scenario Builder's three usage modes, namely, (i) generation, (ii) simulation, and (iii) visualization. These modes coupled with our Web-based interface provide a concise and simple manner for generating models, simulating usage, and evaluating the performance of the virtual data grid and its response to simulated file utilization. Finally, we will present results and further remarks.

3.1 The Anatomy of a Scenario

The foundation of the Scenario Builder is built on the concept of a scenario, or a virtual user and file utilization model, which is a self-contained representation of a data grid. A scenario

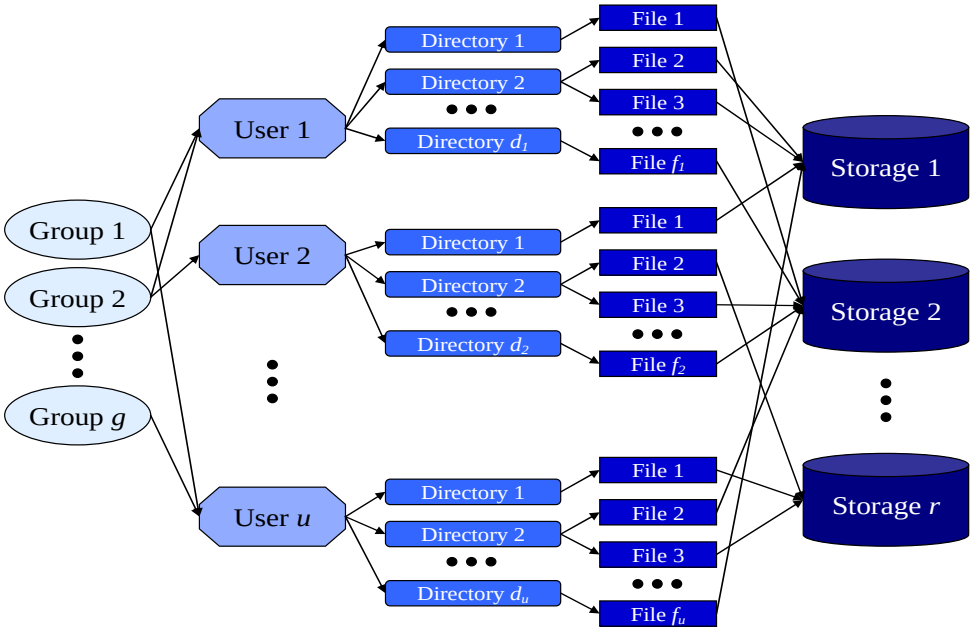


Figure 3.1: The anatomy of a scenario.

consists of users, groups, directories, and files in a virtual data grid. Virtual data grid users belong to groups and own directories that contain files. Files have permission levels (user, group, or public), sizes, creation times, and other common file attributes. Figure 3.1 illustrates these components and their relationships within a virtual data grid scenario.

Internally, a scenario is implemented as a series of MySQL database tables with entries that maintain the elements and their state within the virtual data grid. All scenario components are virtual entities within this database, and any location changes or migrations are simulated by reading or manipulating records in the database tables. This approach presents a lightweight and flexible design, where complex simulations can be run without maintaining physical files or constructing a physical storage network architecture.

3.2 Virtual Network Topology

Another key element of the Scenario Builder is the virtual storage network. This represents the set of storage devices and their associated files, which are available to scenario users. The virtual storage network of the Scenario Builder is itself contained completely within the database, though it is designed to model a physical storage network with the capacity and connectivity constraints found in a physical system. The storage network architecture of the Scenario Builder can be altered to add or remove storage devices or modify system limits by updating its representation in the MySQL database, providing a more lightweight and flexible implementation than by creating a network of physical storage devices for simulation.

The virtual storage network of the Scenario Builder consists of a grid portal node and eleven storage devices. The grid portal node is the main storage repository in the model and represents the starting point of jobs run on the computational grid. When a file is accessed by a

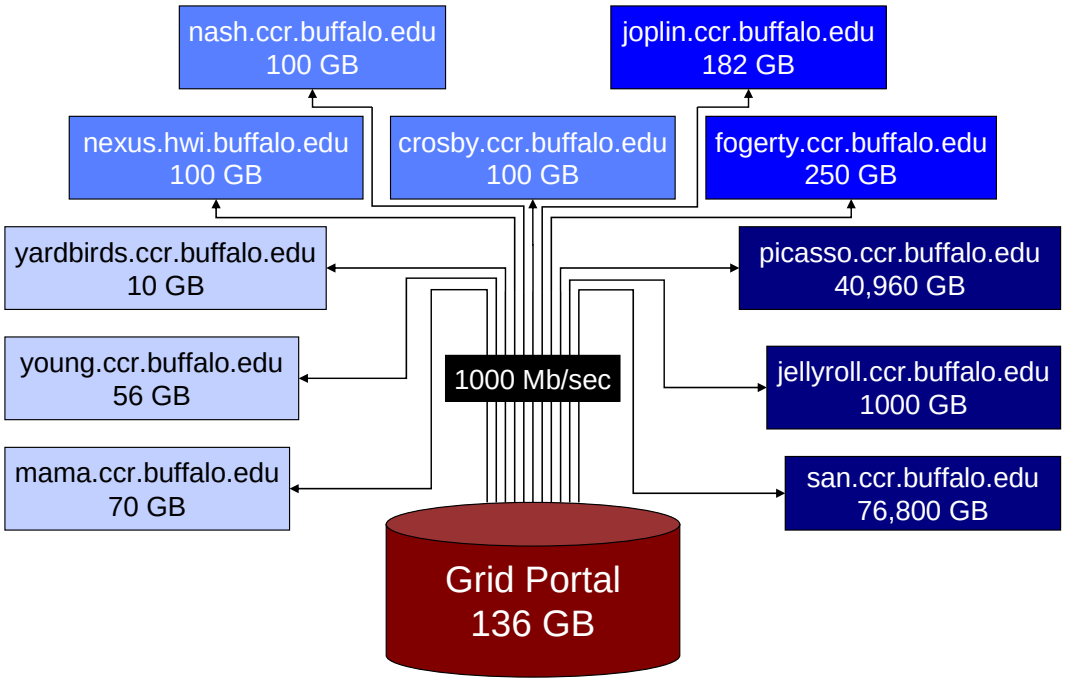


Figure 3.2: Scenario Builder network topology.

user during the simulation routines, its location is updated to the designated grid portal node. As the grid portal node has limited storage capacity, these files must be periodically migrated off of the grid portal node and onto the other storage repositories available. Figure 3.2 represents the topology of the virtual storage network for the Scenario Builder.

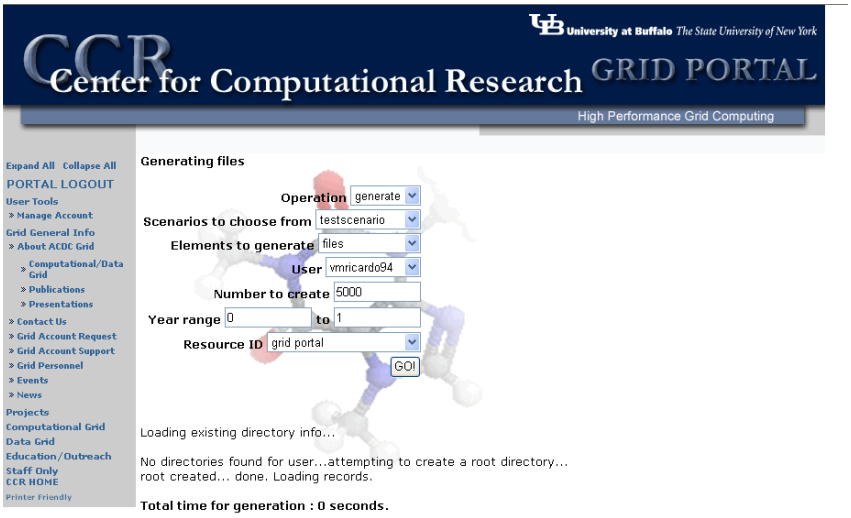
The virtual storage network grid portal node assumes a full duplex gigabit connection to the other eleven storage repositories in the simulation, with no network activity assumed outside of the Scenario Builder simulation. The bandwidth consumed during the periodic migrations (i) to free storage from the grid portal node and (ii) to distribute files across the storage network is considered to be significant, and must be shared between the eleven other storage devices. The bandwidth utilized is calculated based on the random sizes bound to the virtual files and assumes that storage is distributed evenly over the other eleven repositories in proportion to their pre-designated capacities. These imposed storage network limitations, and the topology of the

storage network and its utilization, create the storage and network load scenarios that we monitor. Further discussion is included with the presentation of results in Section 3.5.

3.3 Scenario Generation

Scenario generation takes place through a series of web tools designed to facilitate the simple creation of tailored data grid scenarios through a dynamic and interactive interface. Accessible through the ACDC Grid Web Portal architecture discussed in Section 2, the Scenario Builder generation tool is built using the PHP and Apache architecture used for the portal. The generation suite maintains real-time access to the MySQL database where the scenarios are stored. A scenario is generated by creating the group, user, directory, and file components which comprise the scenarios.

Figure 3.3 features a screenshot from the Scenario Builder generation tool, where 5000



Number	File Name	Dir ID	Res ID	User Name	Grp Name	Creation Time	File Size	Permissions	Scenario
1	Rabbit.xls	1768	100	vmricardo94	cspan20	2005-03-21 23:29:53	2519	2	testscenario
2	Rabbit.ksh	1768	100	vmricardo94	cspan20	2005-05-30 22:51:11	2858	0	testscenario
3	Dozer.m	1768	100	vmricardo94	cspan20	2005-11-09 05:50:36	5321	2	testscenario
4	Smith.txt	1768	100	vmricardo94	cspan20	2005-06-26 16:41:32	9983	0	testscenario
5	Dozer.ppt	1768	100	vmricardo94	cspan20	2005-01-27 19:11:00	77	0	testscenario
6	Dozer.csh	1768	100	vmricardo94	cspan20	2005-05-26 20:13:44	7317	0	testscenario

Figure 3.3: Example of scenario generation.

files were created for a virtual user in the scenario on the grid portal node, with creation times ranging over the year previous to the time of generation. The attributes randomly selected for the file are shown below the form for selecting generation parameters.

3.4 Scenario Simulation

Simulating a virtual data grid scenario represents the core functionality of the Scenario Builder, which models file utilization by the virtual grid users over time. Built through the same interface used for scenario generation, the results of the simulation are used in the evaluation of the performance of a scenario. Users are prompted to enter the user, group, and public access probabilities for scenario data files, as well as the number of years back to initiate the simulation.

A simulation maintains the modified access times of data files, as well as their current locations on the virtual storage network. It also maintains records of bandwidth utilization during migration cycles. Upon creation, a virtual user is randomly assigned two parameters that model file utilization and dictate migration behaviors. The virtual grid user is given a user aging parameter, which represents the probability that the user will access data files within the scenario. The virtual grid user is also given a global migration parameter, which indicates how long files should exist on the grid portal node before being flagged for migration to the other storage elements. These values shape the file utilization and migration patterns observed by the scenario simulation. A virtual user with a high user aging parameter and a low global migration parameter, for example, will access large numbers of files during the simulation while at the same time files will be flagged for migration more frequently, inflating the transfer loads and using more bandwidth than other virtual users possessing different parameters.

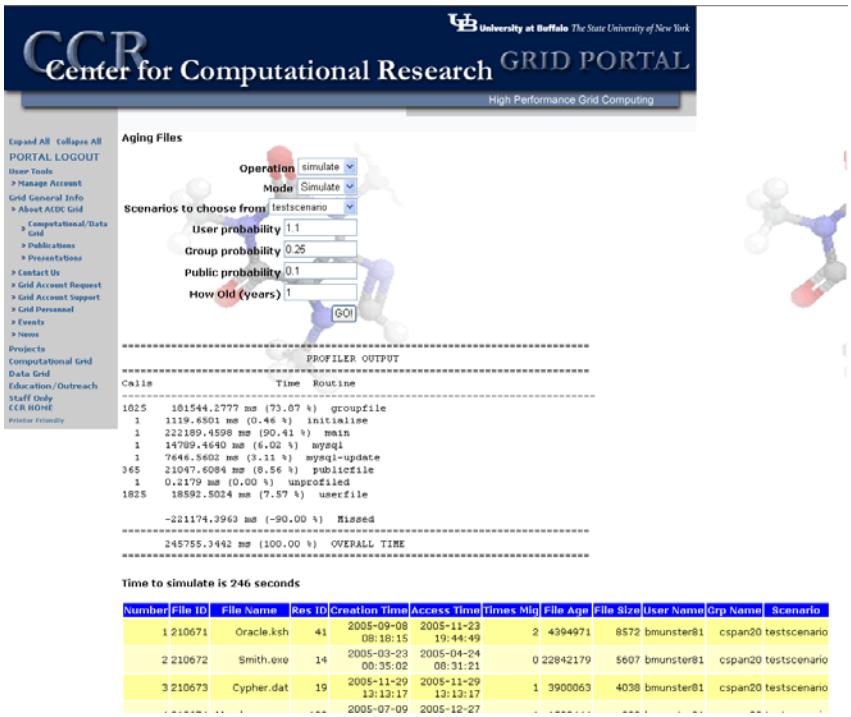


Figure 3.4: Example of scenario simulation.

Simulations run once per virtual day from the starting point to the ending point of the designated time frame. Files are accessed at random according to the specified and randomly chosen access probabilities and placed on the grid portal node. Migrations are used to free space on the grid portal node on a virtual weekly basis or when the grid portal node's utilization exceeds 60%. During a migration cycle, files are flagged if they have gone more days without utilization than the virtual owner's global migration parameter, and are distributed evenly across the other storage devices by their assigned capacities. The bandwidth allocated to a storage device during a migration cycle is proportional to its transfer load during that cycle, and is reallocated as storage devices complete their transfers.

Figure 3.4 is a screenshot from the simulation of a 25,000 file scenario over one year with the default simulation parameters.

3.5 Scenario Visualization and Results

The Scenario Builder is comprised of a series of visualization tools, designed to provide an in-depth view of the configuration of a scenario and the results of the simulation routines. This dynamic Web interface, available with the Scenario Builder generation and simulation tools, provides hundreds of interactive plots, pulling data from the MySQL database tables and organizing it into clickable charts for the evaluation of scenario performance.

Using the visualization suite we can explore the distribution of storage load across the Scenario Builder virtual storage network. Figure 3.5 shows the utilization of storage devices across the virtual storage network after a year-long simulation. The last bar represents the grid portal node which in this example only 15% full, still within reasonable limits, whereas the other eleven bars, representing the other storage devices, are using percentages of their capacities all

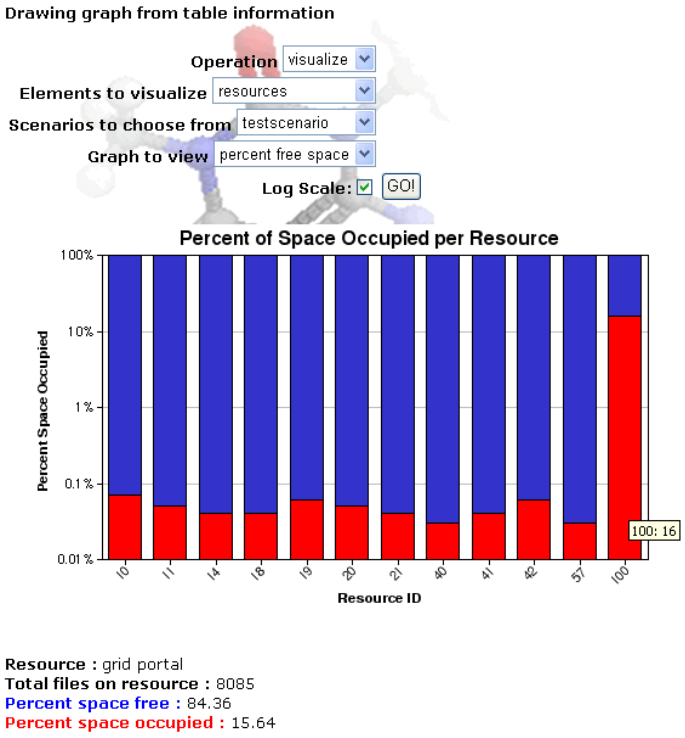


Figure 3.5: Resource capacity chart.

within 0.03% of one another, indicating an even storage load distribution.

We can also examine the network statistics from our scenario simulation to gain an understanding of the bandwidths experienced across the storage network. Figure 3.6 is a screenshot of a visualization chart detailing the bandwidths versus transfer loads over 52 migration cycles of the simulation. The bars represent the total megabytes migrated to each storage element over the time frame of the cycles selected, and the whisker layer indicates the minimum, maximum, and average bandwidths achieved by the storage devices over the same time frame. It is evident from this chart that the storage elements with the highest transfer load also achieved the highest average bandwidth, where the two storage elements with the largest numbers of megabytes transferred observed bandwidths averaging approximately 125 megabytes

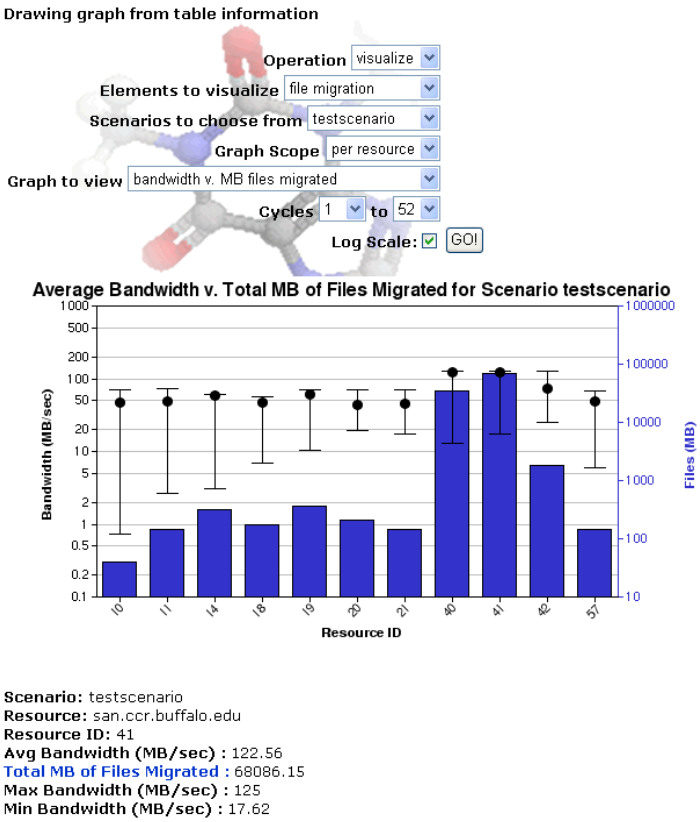


Figure 3.6: Bandwidth versus transfer load.

per second, or the highest possible bandwidth given the gigabit connection. The simulation process is indeed allocating bandwidths proportional to migration transfer loads, and it is evident that for the scenario configuration simulated in this instance that the average bandwidths observed by all storage devices over the entire simulation period are within reasonable limits, or roughly half of the available bandwidth in spite of being shared between up to ten other storage devices.

The distribution of files according to their ages, or the amount of time elapsed since their last access by a grid user within the simulation, is also an important statistic for scenario evaluation. Figure 3.7 shows scenario files by file ages on the grid portal node after a simulation over a virtual year. Each bar represents the number of files accessed on each virtual day, beginning from the leftmost bar which is the last day of the simulation. It is clear from the figure that the largest scenario global migration parameter value of 42 days was enforced, as no files

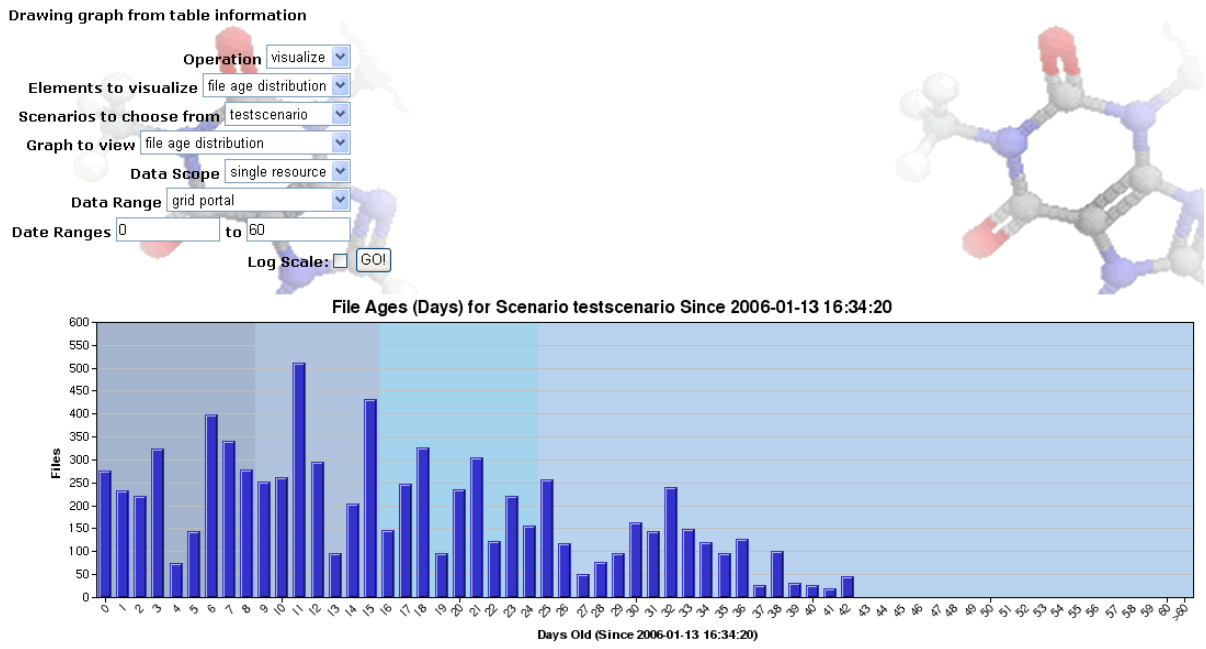


Figure 3.7: File age distributions for a simulated scenario.

exist on the grid portal node that are more than 42 days old. It is also evident from this chart that the file access pattern is not linear. File access appears to occur in bursts for this usage scenario.

Consider a usage scenario where individual files tend to go roughly 50 days between utilization by the virtual users in the scenario. These files would be accessed, moved to the grid portal node, idle on the grid portal node for 42 days, waste valuable storage space, and consume bandwidth in their migration off of the grid portal node, only to be called back to the grid portal node upon being accessed again shortly after. An examination of these usage trends and the application of curve fitting techniques could serve to reduce the wasted storage on the grid portal node and the transfer load to other storage devices, improving the observed performance of the simulated data grid. This type of file usage scenario, and its relation to storage and bandwidth waste, is the basis for the Intelligent Migrator and is discussed in Section 4.

3.6 Further Remarks

The Scenario Builder represents early work for the ACDC Grid initiative in the area of utilizing simulations to examine file utilization and its relation to the performance of a general purpose data grid. The design of generation, simulation, and visualization routines for examining usage patterns, as well as the concept of a virtual data grid representing an independent data grid scenario completely within a MySQL database, serve as the foundation of the Intelligent Migrator discussed in Section 4. The Scenario Builder's relationship with the design of the ACDC Data Grid, and the ways in which it enables the simulation of its performance, are explored in Section 6.

4 Intelligent Migrator

In this section, we present the Intelligent Migrator, an agent-based system designed to minimize wasted storage and bandwidth on storage devices in a virtual data storage network. Based on the underlying design of the Scenario Builder, and using historical information collected by the monitoring infrastructure discussed in Section 2, the Intelligent Migrator is designed to monitor and observe user file usage and make intelligent decisions on where to place data files such that they will have maximum utilization on remote computational resources. The Intelligent Migrator consists of a series of tools for generating, simulating, and visualizing grid environments and file utilization over computational jobs across several remote resources. By studying the performance of the Intelligent Migrator, we can gain an understanding of the performance of a similar physical system.

4.1 The Anatomy of a Virtual Data Grid

The Intelligent Migrator is based on the concept of a virtual file system. This component designates how file systems will be constructed to model real data grid scenarios, and provides the groundwork for simulating and evaluating its performance in a physical setting. Like the scenario of the Scenario Builder, the virtual file system of the Intelligent Migrator is implemented as a series of tables within a single MySQL database.

A storage element in the Intelligent Migrator is defined as remote storage space in a well defined location that is accessible to a computational resource. Storage elements hold copies, or replicas, of files with a reference to the copy on a central storage repository or network of central storage devices local to the implementation of the data grid. By storing a data file on a storage element bound to a grid-enabled compute element, computational jobs scheduled to run on the

compute element using the data file could utilize the file present on the storage element without staging the file to the compute element initially, saving time and bandwidth to the remote host. In the implementation of the Intelligent Migrator, a central data repository for a virtual data grid consists of a single MySQL database table, housing information on every file in the virtual file system. A storage element is represented by a table where file information can be written when a file is replicated to a remote storage element during the simulation. Upon the generation of a virtual file system for the Intelligent Migrator, one storage element is created for each resource represented by an agent.

A virtual file system generated for the Intelligent Migrator also maintains a set of virtual grid users. These users, like users in a physical grid environment, own files and submit computational jobs using their data files. User templates define the number and size of files that users own, the number of jobs they submit, the types of jobs submitted, and the number of files required for computational jobs, as well as the degree to which users deviate from the file utilization patterns established within the user template.

Grid users in a physical grid do not submit jobs at random, and often adhere to project schedules or demands particular to individual users over varying spans of time. The frequencies and durations of computational jobs submitted by grid users, as well as the compute elements jobs are submitted to, can have a significant impact on the usage of the associated data files and the patterns in which typical usage will follow and be captured by the Intelligent Migrator. The job history collected through the ACDC Grid Monitoring infrastructure developed within the Center for Computational Research serves as the pool of jobs from which virtual file system job histories are generated. Spanning several years of data collection for 68 compute resources, this repository holds valuable computational grid usage information on over 3.5 million jobs,

including runtimes, user and group information, and other job attributes. Jobs for the compute elements supported by defined agents in the Intelligent Migrator are pulled at random from this monitored information and used as the computational jobs are submitted within the simulation routines.

To preserve the natural user job submission patterns present in the real monitored job histories, each user generated in the virtual file system is bound upon creation to a username from the job history pool, where a username in the job history repository represents all grid users who map to that username on the compute element to which the job was submitted. Files are bound to these jobs in the virtual data grid, translating the utilization of a computational grid to the data grid file utilization patterns we wish to examine.

There are several benefits to the architecture outlined in this section and to creating a completely simulated file system environment. A virtual data grid is lightweight, where its properties are housed in a small number of database tables of no more than a few megabytes each and are generated in a manner of minutes. The lightweight design of the file system is also

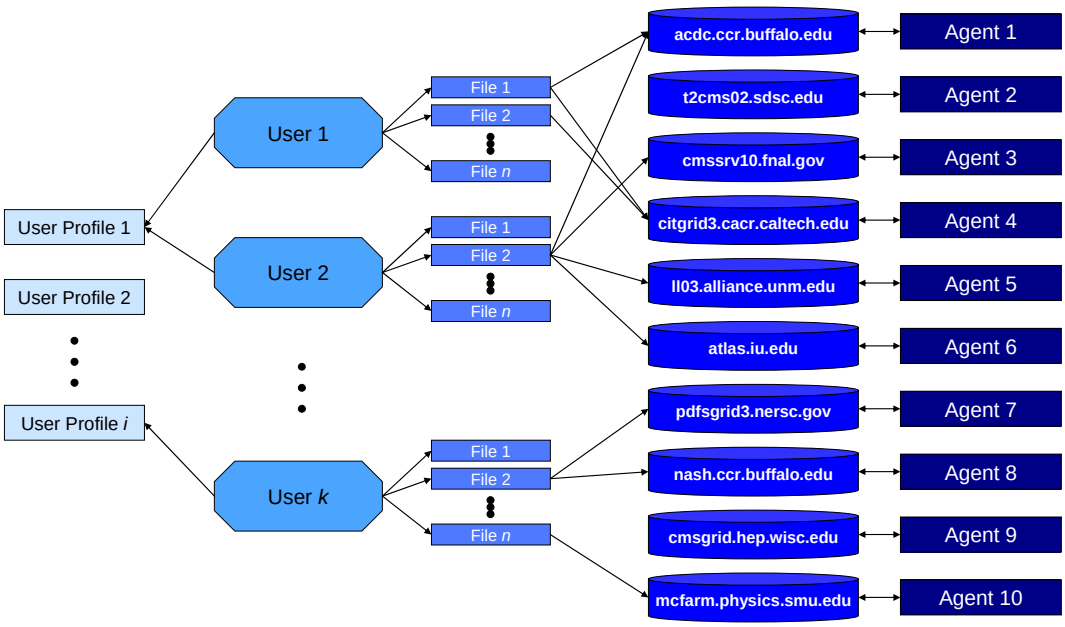


Figure 4.1: The anatomy of a virtual data grid.

highly flexible. Storage system components, distributions, and usage scenarios change at a rapid rate as grid initiatives are born, grow, and mature. The simulations of existing data grid storage scenarios and those that may develop in the future are essential in order to fully analyze the effectiveness of the optimization routines of the Intelligent Migrator. Providing an examination of the performance of the Intelligent Migrator in several usage scenarios is an important step in verifying the underlying intelligence of the system. The virtual file system architecture of the Intelligent Migrator allows for the generation of side-by-side comparisons and analysis of the performance of the optimization routines. Its design is also closely aligned with the design of the ACDC Data Grid at the CCR such that the Intelligent Migrator can be easily deployed on the live system. The anatomy of a virtual data grid and its components is illustrated in Figure 4.1.

4.2 Agent Infrastructure

The study in methods of distributed artificial intelligence has long been investigated as a solution to such optimization problems. Such methods come from fields that include cooperative systems, game theory, and distributed computing. A multi-agent system is an implementation of several small components, or agents, which interact to solve a problem or sub-problems in a large system. Multi-agent systems typically employ four main principles in their design and implementation. First, agents must have some level of autonomy and have some control over actions and their internal state. Second, agents must be able to gather information on their environment and react appropriately. Third, they must be able to interact with other agents, typically with a pre-defined agent communication language or otherwise established protocol. Finally, they must also be goal-oriented, and their actions as a part of a whole system should move toward that goal as the system runs [40, 41].

These multi-agent systems are often designed so that each agent has incomplete information regarding the overall problem to be solved and that there is no global control of the entire system. Similarly, in many multi-agent systems, data is decentralized and the actions of agents are asynchronous with regard to the entire system. Inter-agent communication is often facilitated through defined techniques such as marketplace, auction, or blackboard dialogue styles [40-42].

Multi-agent systems present attractive architectures for system design for several reasons. Partitioning responsibilities among individual agents results in a robust system, often eliminating single points of failure and providing a system more adaptable to failures and new scenarios. On a similar note, they may be easily distributed across parts of the system or even across many machines, lending themselves to growth in the development of distributed systems in many research sectors and even to grid development initiatives [40, 41].

Though there are many advantages to deploying a multi-agent system in a response to a difficult learning problem, there are many challenges that can arise in this type of distributed architecture. The methods of agent communication, for one, must be carefully planned as agent interdependencies may require the sharing of limited resources and a well-defined choreography of handoffs to prevent agent starving due to the lack of global control of the system. Keeping agent information consistent is another challenge, as agents may attempt to solve their sub-problems with out-of-date information from surrounding agents due to limited communication opportunities or the dynamic nature of the system [40, 41].

Multi-agent systems vary widely in their practical applications. Problems such as managing control systems for satellites and communications systems are just two examples for which multi-agent systems are well suited [43]. Multi-agent systems have been used to solve

scheduling problems and manage distributed resources [41]. They have also been applied to many problems relating to the field of grid research. One such project is the development of AGEGC [44], a grid computing model based on the Multi-AGent Environment (MAGE) platform, designed to facilitate matching resource requests with available grid-enabled compute elements. In the AGEGC system, resource agents represent high performance compute elements and act as service providers, with registration and lookup mechanisms in place for locating resource agents and the services they provide.

The Intelligent Migrator uses independent agents to determine where to place files among a pool of remote compute resources with limited storage, given a series of user files. Each learning agent is privy to information regarding the past history of files on the resource it is assigned to, and is capable of making decisions regarding the distribution of data grid files over the limited storage available on its compute element. Agents submit requests to replicate certain files to their storage elements or to remove them, in order to ensure that the files with the highest likelihood of utilization are present for grid users and their computational jobs.

There are many benefits to the multi-agent implementation of the Intelligent Migrator. Most importantly, using an agent-based method simplifies the problem of determining where to place a file to achieve the highest utilization, to evaluating whether a file should be replicated on a per compute element basis. As data files are copied to remote compute elements and not shared among them, optimizing data utilization across n compute elements is, in fact, equivalent to n completely independent sub-problems, as optimal utilization on each resource constitutes optimal utilization across an entire computational grid. Agents are thus free to solve their sub-problems in a self-interested manner, optimizing utilization on their own storage elements and contributing to the solution of the entire problem. As this particular optimization problem is in

fact a composite of independent optimization problems between agents, there is no need for agents in this system to communicate and thus many of the challenges associated with constructing a multi-agent system as discussed previously are avoided.

In addition to simplifying the optimization problem, this type of system is extremely scalable as the underlying computational grid grows and evolves. Agents can be generated or removed as new compute elements are added to or retired from the computational grid. The asynchronous and decoupled nature of multi-agent systems and especially of the architecture proposed in this section lends itself well to parallelization, and agent routines and computations could be integrated into computational grid jobs of their own and processed within the grid infrastructure they support.

4.3 The Anatomy of an Agent

Artificial Neural Networks (ANNs) provide a robust method of solving complex machine learning problems. This supervised hill-climbing technique, commonly used for problems such as approximating real-valued or discrete-valued functions, has been proven to be highly effective in many practical applications. Neural networks are ideal for problems where supervised training sets are available, including training sets that contain noisy or erroneous entries. In addition, neural networks do not require a human understanding of the final learned target function. Changes to a neural network's design, such as its perceptron connection architecture, the encoding scheme of network inputs and outputs, and the energy function used for determining weight delta rules, are often tailored to meet the specific needs of a problem [45].

The Intelligent Migrator appears to be an ideal candidate for the application of artificial neural networks in terms of modeling file utilization in individual agents. Agents within the

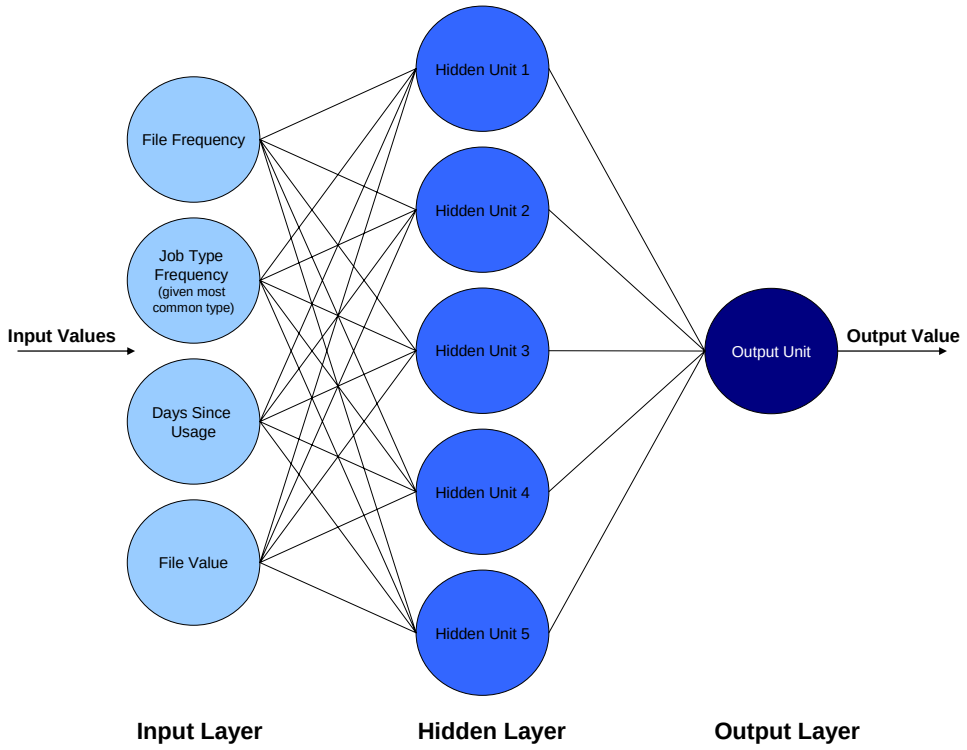


Figure 4.2: Agent neural network topology

Intelligent Migrator each possess a layered artificial neural network that represents the latest learned model of file utilization for the compute element assigned to the agent. Observed file utilization parameters are fed through the neural network file by file, and the magnitude of the output dictates the perceived strength of the file on the remote storage element assigned to the agent. Supervised training sets can be constructed on-the-fly from immediate past observed file utilizations, and data is inherently noisy from natural user deviations from past usage patterns. A fast evaluation of file parameters for the agent decision process is essential to the efficiency of the system, and an understanding of the learned user utilization model is not required for the execution of individual agents within the Intelligent Migrator.

Artificial neural networks have been applied to problems ranging from visual and speech recognition tools to scheduling systems and autonomous vehicle control. Pomerleau's ALVINN (Autonomous Land Vehicle in a Neural Network) system at Carnegie Mellon uses visual sensory

input to train a neural network to making steering decisions [46]. Neural networks have also been applied to the knapsack problem and similar optimization problems [47]. In [48], an interesting discussion focuses on dynamically constructing generic neural networks of clusters of interlinked perceptrons to represent a system's state and possible solutions, with some configurations achieving convergence on a solution on the order of 10^8 times faster than sequential heuristic search techniques to solve similar problems.

The anatomy of an Intelligent Migrator agent is contained within the definition of the parameters of its agent type, including, but not limited to, the dimensions of the neural network model within the agent, the amount of simulated time over which to gather file parameters for calculating input values, and the training parameters used to tune the model on observed file utilizations. Agent types bring flexibility to the system, providing a means of customizing agents as new types of compute elements enter the already heterogeneous system. The agents deployed in this discussion of the Intelligent Migrator possess a single layered neural network with four inputs, five hidden nodes and a single output node. The input arguments fed to the neural network for a single file include the number of times the file was utilized on the remote resource, the number of times a job of the type most frequently associated with the file has been submitted by the user, the number of days since the file was last used by the user on the remote resource, and the current value of the file on the storage associated with the resource, or its value on the main storage repository if it is not present on storage local to the compute element. Figure 4.2 illustrates the topology of the agent's internal artificial neural network.

4.3 Generating a Virtual Data Grid

Virtual data grid generation is the first step in utilizing the Intelligent Migrator to

simulate file utilization and determine the accuracy and effectiveness of the underlying algorithms. Implemented as a PHP script through a command line prompt, this procedure constructs the components that comprise a complete virtual file system. Upon the generation of a virtual data grid, a MySQL database is created based on the user designated virtual file system name. Intelligent Migrator users then specify, in pairs, the user profile template number, which is discussed in Section 4.1, and the number of users to generate for each user profile template type. Information regarding the number and attributes of files to create, the number of jobs to select, and file binding information, is pulled from the user template definitions in the MySQL database and used in generating the virtual file system.

The computational jobs run for virtual users in the generated data grid are pulled from job information collected by the ACDC Grid Monitoring infrastructure, as discussed in Section 2. The virtual users created are bound at random to usernames from the job pool, and jobs are drawn from the pool for each virtual grid user, though the order of the jobs is maintained to preserve the natural grid utilization patterns present in our job history records. As jobs are pulled from the job history pool available to the Intelligent Migrator, they are assigned to a job type, representing a grid application, according to user profile template information.

Grid users in a real grid setting do not access their files at random. There are often sets of files that a user tends to select for use with certain computational jobs, and these may or may not overlap with files submitted with other computational jobs. File utilization is not uniformly random, as there are files accessed for computational job submission with a much higher frequency than others. We model this in the Intelligent Migrator by ordering a virtual user's files and choosing a random starting point among them for each job type they will submit. For each file bound to a job, we walk randomly in either direction a small distance from this point and

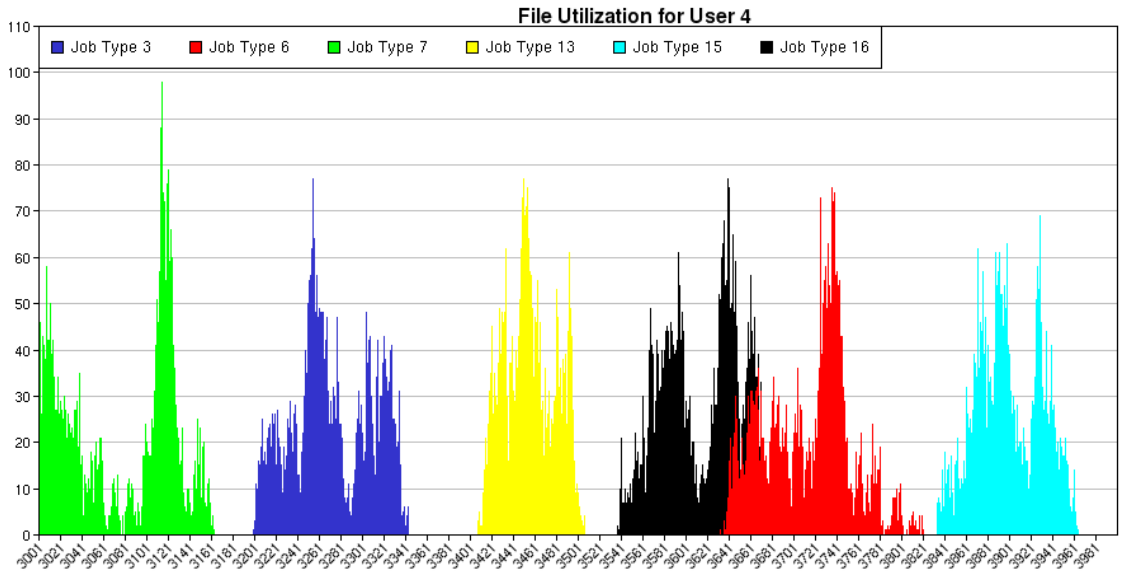


Figure 4.3: Intelligent Migrator file utilization example.

select files. The file furthest from our starting point becomes the starting point in the event of the next job submission of the same type. Choosing files in this manner approximates a multi-modal function for file utilization, where small pools of files are utilized for computational jobs of the same type and represent the same scientific grid application. An example of generated file utilization for a virtual user submitting jobs of six types is shown in Figure 4.3, charted using the Intelligent Migrator visualization tools discussed in Section 4.6. In order to ensure that it is possible to correctly select and place all files required during a day onto the correct network storage element, the virtual data grid is examined and repaired such that on any given day for any given compute element, the total size of the files required does not exceed the capacity of the compute element. Any file that exceeds this limit is discarded.

After the virtual data grid is constructed, the neural network is created for each agent present in the Intelligent Migrator system. The dimensions of the layered acyclic neural network, discussed in more detail in Section 4.3, are pulled from the definition of the agent type possessed by each agent and initialized to random values and stored in the MySQL database

definition of the virtual data grid. These random weights represent the starting point of the neural network for each agent upon the initialization of the simulation procedure.

4.5 Simulating a Virtual Data Grid

The simulation process of the Intelligent Migrator uses the virtual data grids tailored to a specific storage scenario and simulates file utilization over time. The simulation procedure is implemented as a series of PHP scripts initiated through the command line, with full access to the MySQL database where agent information, user profile templates, and the virtual data grids are defined. The Intelligent Migrator simulation routine is designed to run in two modes. The first mode, or base case, is the most simplistic and replicates data to compute elements in a purely reactionary fashion. The second mode, built upon the functionality of the first mode, initiates the intelligent agents that perform additional replications based on observed past utilization and predicted future file usage on Intelligent Migrator storage elements. Simply by specifying the virtual data grid to run, simulations modify the virtual data grids to keep performance histories, submit and perform replication requests, and train the neural networks for each agent as file usage patterns are observed over time.

The generic simulation procedure begins at a random point in the job history of the virtual data grid and runs until the last submitted computational job is processed. The simulation proceeds once per virtual day and examines the computational jobs submitted over the next 24 hours. A submitted job for a resource is associated with data files, and these files must be present on the storage element of their associated compute elements in order to run successfully with the job execution. All files not present on storage elements for the compute elements they are required on are copied to these storage elements during the current simulation round. If a

storage element is at its capacity, the oldest data on the storage element is removed to free space for the required files. This step proceeds once daily until the simulation is complete.

This reactionary implementation provides the base case for comparison of the Intelligent Migrator's performance. Replicating files only as they are needed provides no analysis of past utilization or any attempt at replicating files before they are needed. Files are only present before they are required if there is high overlap in file utilization day-to-day, and any files which are removed from the storage element to make room for new required data must be re-replicated to that storage if needed in the future. The agent-based approach for learning file usage patterns is applied in the second mode of virtual data grid simulations.

There are two key differences between the intelligent mode of virtual data grid simulations and the generic simulation procedure. The intelligent mode utilizes assessed file values to assign weight and importance to files and to build a rough model of file utilization over time. File values appreciate as they are used by their owners and depreciate over time as they are passed over for other files. It is assumed that files with higher values are more valuable to their owners for submission with computational jobs. Appreciation and depreciation parameters are defined in the specification of an agent's type and are a function of recent file utilization on the file repository or on remote storage elements. In the intelligent mode, the file value is used in assessing which files to remove from remote storage when there is insufficient storage space on a compute element's storage device. In the generic mode of simulations, files are removed from a storage element based on their age, where the oldest files are removed with the highest priority.

Secondly, and more importantly, agents are deployed in the simulation of a virtual file system using the intelligent mode. These agents, possessing neural networks accepting file parameters and returning an assessment of future file worth on a given compute resource, are

utilized during the virtual days in the simulations to boost the performance of the Intelligent Migrator over the generic simulation mode. Before the daily replication of required files to storage elements, each agent in the storage network is consulted and given the opportunity to replicate files to the storage element beforehand. The agents gather past file utilization information and determine the pool of files to replicate by considering previous file utilization as well as file values on the repositories and remote storage devices. At the end of a simulated day, each agent's accuracy is measured over the previous day and stored in the database. The agent is only permitted to replicate files to its associated storage element if its latest test accuracy is over 70%.

As file utilization and job submission patterns change over the course of the simulated time period, agents are retrained to represent the latest model of perceived file usage to maintain the accuracy of their neural networks. After agents make their replications and the required files are replicated to the necessary storage elements, agents are retrained if their accuracy decays to below the 70% threshold.

The application of file values and agent learning models to the general simulation routines presents two models for the evaluation of the algorithms designed for the Intelligent Migrator. The generic case provides a basis for comparison with the intelligent approach, allowing for an assessment of the accuracy of the agents on a virtual data grid and the benefits of applying the intelligent model to a similar live data system.

4.6 Intelligent Migrator Visualization and Results

The Intelligent Migrator encompasses a suite of visualization charts designed for an analysis of the results of the internal learning algorithms. These Web tools, utilizing dynamic

charts similar to those found in the Scenario Builder, provide information on-the-fly from the MySQL databases for the virtual data grids created. Graphs displaying the correct responses of the underlying implementation, and the test accuracy of the neural network, to name a few, can be generated by specifying the virtual data grid to evaluate.

In this discussion of results, we present four generated virtual data grids, each possessing different attributes for comparison. Each virtual data grid is associated with the same ten agents, representing ten different compute elements from across the grid initiatives monitored by the ACDC Grid Monitoring Infrastructure. Three user profile templates were established for these examples and applied to the virtual data grids created. Each user profile template creates users who possess 1000 files randomly sized roughly between 100 kilobytes and 1 megabyte. The users submit up to 10,000 jobs chosen from six job types. The three user profile templates differ in that among user profiles one, two, and three, users are limited to submitting a maximum of 10, 15, or 20 files per job, respectively. Each user is given a five percent chance of deviating from these figures and submitting anywhere between 0 and twice the number of files in their individual limit. We generated four virtual data grids such that one grid has ten users of profile type one, the second experiment has ten users of profile type two, the third has ten users of profile type 3, and the fourth has fifteen users, five of each profile type. Each virtual data grid was run both in the general simulation mode as well as in the intelligent mode using multiple agents with artificial neural networks.

Many factors indicate the performance of a simulation of the Intelligent Migrator. For example, in the intelligent case, it is important to examine the accuracy of the neural networks. However, most importantly, we wish to look at the hit rate for a simulated system. A hit is when a file is requested for submission with a computational job and it already exists on the storage

system associated with the compute resource, such that the file does not need to be copied to the compute resource with the job, and time and bandwidth are spared in the transaction. The more hits during a simulation, the more efficient the storage network, and the more intelligent the system simulated by the Intelligent Migrator. A hit rate is the number of hits versus the number of requested files over a period of time.

The results of simulations of the four virtual data grid systems consistently show that the general mode simulation is slightly more effective than the intelligent mode in correctly replicating data to remote storage. Performance in the first three virtual data grids yielded hit rates within or close to 1% of one another between the two simulated scenarios, with a steadily increasing hit rate percentage as the number of files submitted per user increases, between a 16-17% hit rate in the first virtual data grid and an 18-19% hit rate in the third virtual data grid. This may be attributed to an increased instance of repetition in file use, as more files are chosen for each job over the three user profile types while the number of files available to each user remains constant. The fourth virtual data grid, consisting of a mix of the three user profiles, and with the lowest instance of file repetition, faired worst of the four, yielding a 12-13% hit rate between the intelligent and general cases.

When broken down over compute resources or users, the intelligent mode in some cases exhibits an improved performance over the general mode. Virtual users may be the sole users of a compute element, or closely match the learned neural network models of the compute resources to which jobs are submitted. Similarly, compute elements represented by agents may experience low variability of file utilization over time or a generated file usage pattern otherwise conducive to an increased performance by the intelligent agent infrastructure, where neural network input arguments are more closely related to future file usage. Figure 4.4 shows the hit

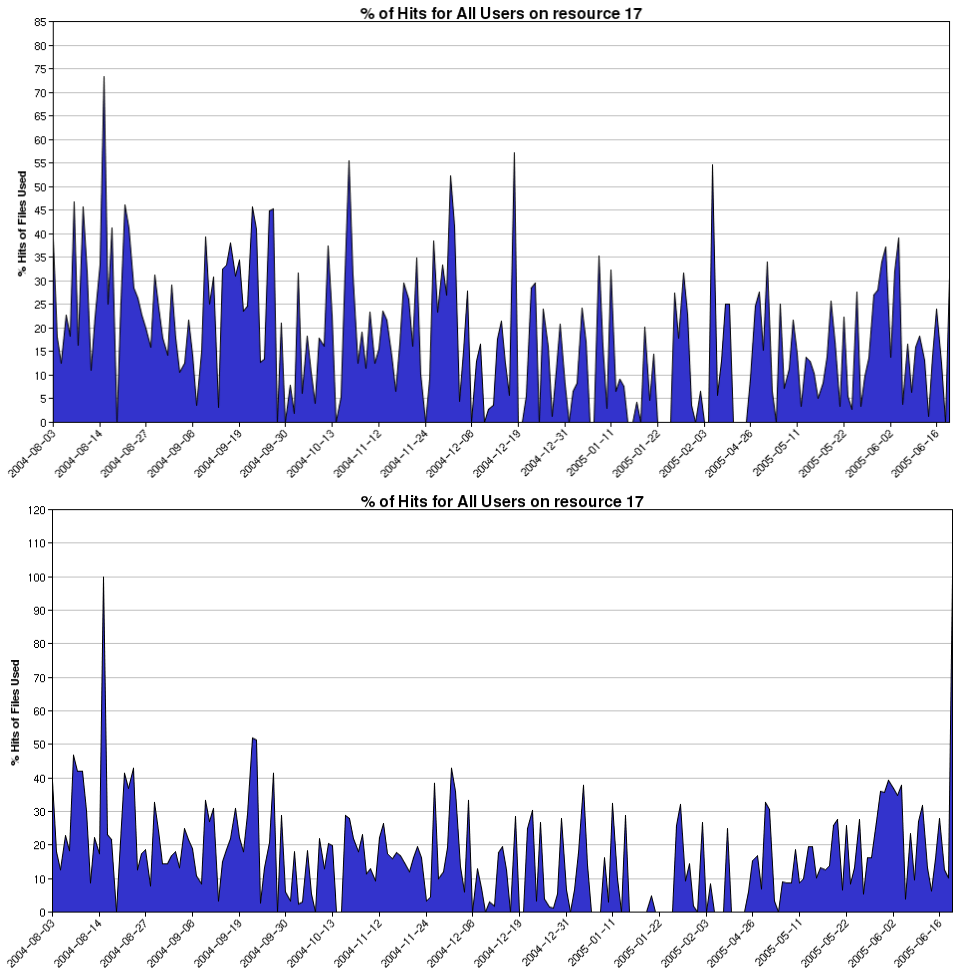


Figure 4.4: Intelligent Migrator hit rates example comparison. Intelligent mode (top), and general mode (bottom).

rates for the same compute resource between the intelligent and general modes. The intelligent mode, featured on the top of the graphic, sustains similar but slightly improved overall performance over the bottom general case.

In spite of the reduced performance of the intelligent replication mode over the general mode, the neural networks implemented and trained for each agent received high test accuracies over the duration of the simulations. Average test accuracies for the neural networks, taken in the 24 virtual hours after the neural network is consulted during the simulation, yield average test accuracies ranging from 80% to 98%, consistent over all virtual data grids utilized in this

analysis in spite of their differing parameters. There are three measures of neural network accuracy, namely, (i) the percentage of positives marked, or the percentage of files submitted to a compute resource which are marked as likely to be required on the remote storage device, (ii) the percentage of negatives marked, or the percentage of files not used which are marked by the neural network as not likely to be required on the remote storage element, and (iii) the overall classification accuracy of the neural network over measures (i) and (ii). Figure 4.5 shows example plots of these three neural network accuracy measures for an agent after a simulation.

Findings conclude, however, that high neural network accuracy does not always

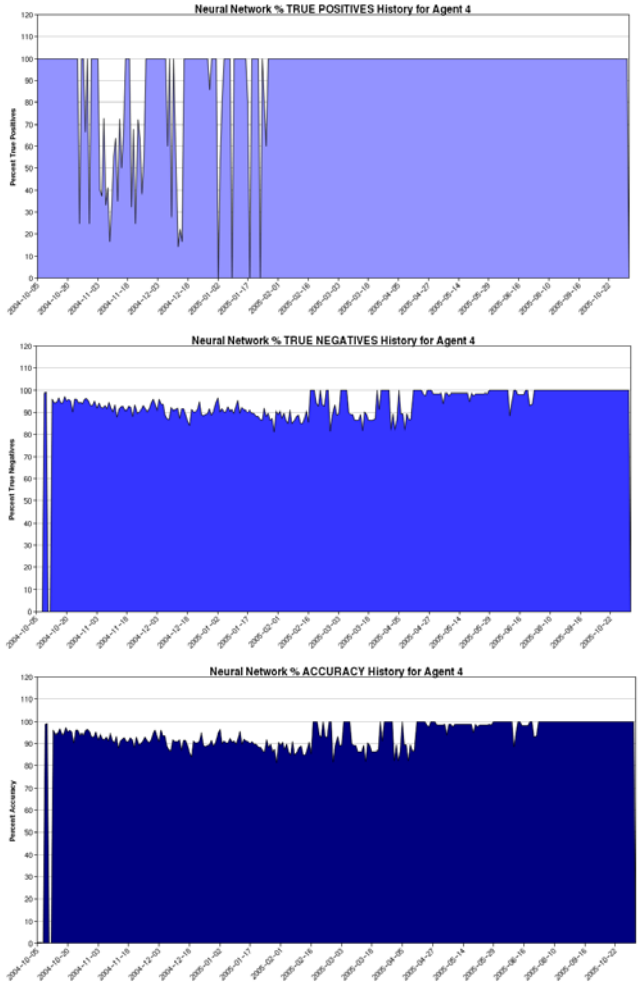


Figure 4.5: Intelligent Migrator neural network accuracy plots for a single agent.

correspond with a high hit rate for an individual agent. The resulting low performances can be attributed to several factors. Flagging too many files for replication and having only the capacity on a remote storage device to replicate a few files that happen to be incorrect can lead to reduced performance. As there are many available files in the system and only a handful of them will be used on any given day, remote storage capacity is reached quickly and files may be left off of the storage element even if they are correctly marked for replication. In addition, as only a handful of files are utilized at a given time, a neural network state which marks no files for replication will have an artificially high overall accuracy as the majority of files, which were not used in actuality, were marked correctly. More closely tuned neural network training and an increased scrutiny over test accuracies could yield an improved performance and correlation between neural network performance and overall system efficiency.

Though the general mode of the Intelligent Migrator performs better than the neural network implementation of the Intelligent Migrator in most cases, the infrastructure presented here represents a framework for the development of learning algorithm enhancements or new approaches to modeling user file utilization. A further examination of usage patterns, job selection, and the learning model could yield substantial improvements in the Intelligent Migrator's performance. For example, changes to the input arguments for the neural network or to its internal structure and training could lead to better results. If the attributes being given for each file to the neural network are not good indicators of the file's future utilization, the neural network will frequently fail to flag the correct data. In addition, smaller training windows or an adjustment to the classification criteria when preparing the supervised training set may more closely tune the learned models. A closer look at the virtual data grid, the simulation routines, and in the ways in which files are chosen, could yield improved performance as well. The

selection of files for submission, for example, may be biased toward supporting the general model for file replication.

The multi-agent learning framework of the Intelligent Migrator represents an attempt to apply a distributed intelligent system to a distributed storage system. Though its initial performance does not yield improvements over the more general file placement case, the Intelligent Migrator presents a solid framework for enhancing the learning models and producing improved results. Further experimentation in the development of these improvements promises avenues for providing reliable and scalable data service to a computational grid and a means for efficiently utilizing storage and network resources.

4.7 Further Remarks

The Intelligent Migrator represents an effort in the ongoing initiative to present scalable and robust data service to the ACDC Grid. The examination of user utilization patterns and algorithms to model them is an important step in efficiently making use of limited storage and boosting the performance of a physical data grid and facilitating improved data service to a computational grid. In Section 6, we will discuss the relationship between the Intelligent Migrator and the design of the ACDC Data Grid.

5 The ACDC Data Grid

The ACDC Data Grid, which serves the ACDC Grid and provides data storage services to grid users, is a network of interconnected and distributed storage resources that are logically linked to provide a single continuous storage environment. In this section we present the ACDC

Data Grid, including the data file virtualization methods designed to provide seamless data access over multiple heterogeneous storage elements. We then discuss the implementation of a storage element and the architecture of the storage network for the ACDC Data Grid. We present the user and administrative Web tools that provide a user-friendly interface with the ACDC Data Grid. Finally, we include current results and present some additional remarks.

5.1 Data Grid File Virtualization

Data grids consist of a network of geographically-distributed heterogeneous storage systems, where storage devices move in to and out of the network and the location of individual files is dynamic. Creating a robust and scalable data grid to provide the necessary data services to a computational grid, and, in particular, to grid users, requires that files be transparently accessible, regardless of their physical location in the storage network. This distributed storage environment should appear as a single homogeneous service to the grid user, and must implement seamless and scalable maintenance of user-imposed naming conventions and directory hierarchies similar to those found on traditional file systems. Maintaining such a flexible file system across an arbitrary storage network is a significant challenge.

In our design of the ACDC Data Grid, we have made a distinction between the information necessary for identifying and locating a data file within a distributed storage network, and the user-imposed naming conventions and directory hierarchy required to provide the impression of a single continuous storage space to grid users. By utilizing a MySQL database, we maintain the file organization imposed by grid users through the web interfaces discussed in Section 5.3, while binding this information to file location records for referring to data internally. Files entering the ACDC Data Grid are stripped of their user-imposed names and

directory locations and permanently assigned to a Globally Unique IDentifier (GUID), which is used as its physical file name. A record in the database is created for each file, mapping its GUID to its user-imposed name, directory, and its location on the storage element in the distributed storage network.

This process in essence creates two entities for each data file entering the data grid, namely, a physical file with a GUID, and a user-based identifier that only exists within the database. The virtual data grid, implemented through calls to the database, creates an image superimposed on the physical storage. The physical data grid can consist of an arbitrary distributed network of directories that are completely transparent to grid users, while the

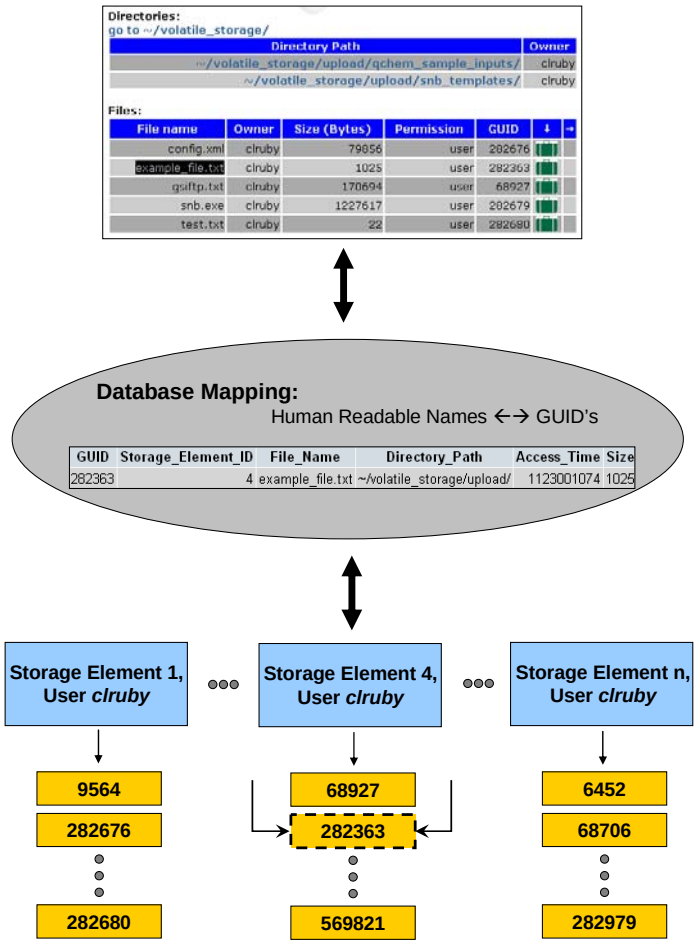


Figure 5.1: Virtual and physical database mapping.

database provides a mapping to the higher-level virtual data grid that provides the complexity and flexibility users require in order to impose their own file naming conventions and organization.

There are several benefits to this approach. The decoupled architecture provides simplified access to files on distributed storage devices. Queries to the database reproduce the user-imposed directory hierarchies as a single directory tree accessible through the Web portal. APIs in place query the database and can retrieve a GUID for a user-defined name and directory location, and the mapping in the database for this GUID yields its physical location on the disk of a storage element. This process is illustrated in Figure 5.1. Files may be added or removed arbitrarily without needing to maintain complex user-imposed directory structures within the storage devices. Users are free to update the names and directory locations of their virtual files in a homogeneous representation within the Web portal with a series of simple, low cost updates to the database without visiting or modifying their physical data files. Creating this virtual image of physical storage provides a seamless interface to an arbitrarily distributed storage network, overcoming the complexities of managing a seamless interface to the data grid over multiple heterogeneous storage devices.

5.2 Storage Network Architecture

A scalable data grid solution must be equipped to grow with evolving demands by grid users and their applications. New storage devices must be easily integrated into the existing storage network, regardless of their architectures or platforms, so that the data grid can scale as demands for data services increase over time. Physical storage devices in the ACDC Data Grid storage network are implemented as designated directories on a grid-enabled compute element.

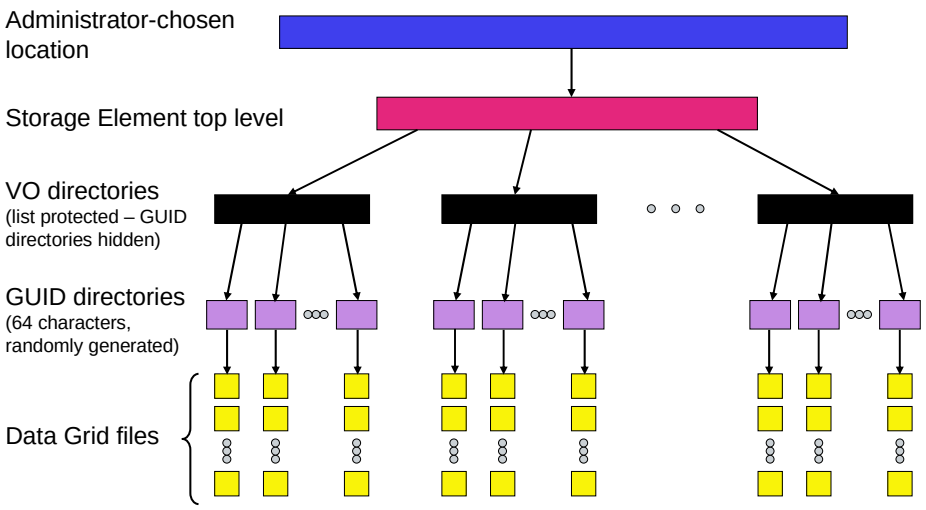


Figure 5.2: The anatomy of a storage element.

These directories contain a directory for each Virtual Organization (VO) users of the ACDC Data Grid, which at this time consists of the Grid Resources for Advanced Science and Engineering (GRASE) VO. Each grid user possesses a GUID directory within the VO directory of the VO the user is associated with, and all files belonging to the grid user on the storage element are stored directly within this GUID directory. As no two data grid files share a GUID, there is no chance of namespace overlap within this flat directory. An illustration of the structure of a storage element in the ACDC Data Grid storage network is shown in Figure 5.2.

Storage element information, including the names of the machines hosting them, the directories in which they are located, and the ways in which they are accessed, is stored within a MySQL database. These grid-enabled storage devices are accessed using the Globus Toolkit, and though storage elements within the administrative domain of the CCR are mounted and accessed directly to enhance internal performance, these can be changed to use Globus through modifications to database entries. Given the GUID of a file to access, the full physical path to this GUID can be constructed given the owner of the file, the owner's associated VO, and the storage element it is located on, allowing for seamless access in spite of arbitrarily distributed

data files. The directory implementation of a storage element provides simple access in spite of the differing underlying architectures of the physical storage devices.

There are three types of storage elements implemented within the ACDC Data Grid storage network, namely (i) persistent, (ii) volatile, and (iii) replica storage devices. Persistent and volatile storage elements are located within the administrative domain of the CCR and the ACDC Grid. Persistent storage elements are directories that are recoverable in the event of a device failure, and are intended for user files that are critical and are not easily reproduced. Volatile storage, which is more abundant than physical storage, is implemented on devices where data recovery is not guaranteed. A data file exists exactly once on one of the persistent or volatile storage devices within the CCR.

Replica storage space, on the other hand, is implemented on grid-enabled compute elements beyond the scope of ACDC Grid administrative control. Inherently volatile, the data

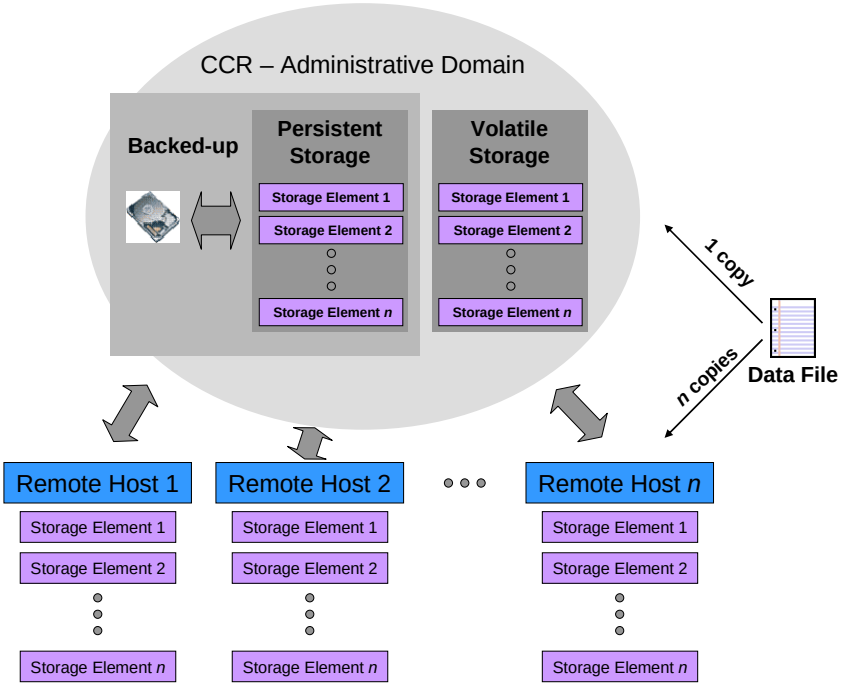


Figure 5.3: Persistent, volatile, and replica storage organization.

files stored in these storage elements are copies, or replicas, of data files existing on persistent or volatile storage devices within the CCR. This storage network architecture introduces the potential for saving storage space and bandwidth used in repeatedly submitting files with computational jobs.

Figure 5.3 illustrates the persistent, volatile and replica storage space design of the ACDC Data Grid.

5.3 Web Tools for the ACDC Data Grid

A series of Web tools has been developed to facilitate user data management as part of the ACDC Data Grid implementation. Users may upload, manage and retrieve data from the data grid using the ACDC Grid Portal. This Web interface is designed to deliver seamless access to data files, concealing the distributed storage network and providing the impression of a single, coherent data repository.

Data files are added to a user's home space automatically as results from computational jobs, or by uploading them through the data grid upload interface. Users may upload single files or drag-and-drop entire directory structures from their local machines through the Rad Inks Rad Upload [49] Java™ applet [50], which is embedded in the Web tool. Users also have the ability to view and manage the files they own on the data grid. Files are displayed using the user-imposed virtual directory structure that is navigated by a series of links to all sub-directories of the current viewing directory. Files may be selected for a number of different tasks, including renaming, copying, removing, permissions modification, viewing, and editing. Users may also replicate their files to a number of remote storage elements available to them and manage their replicas through a similar file browse interface.

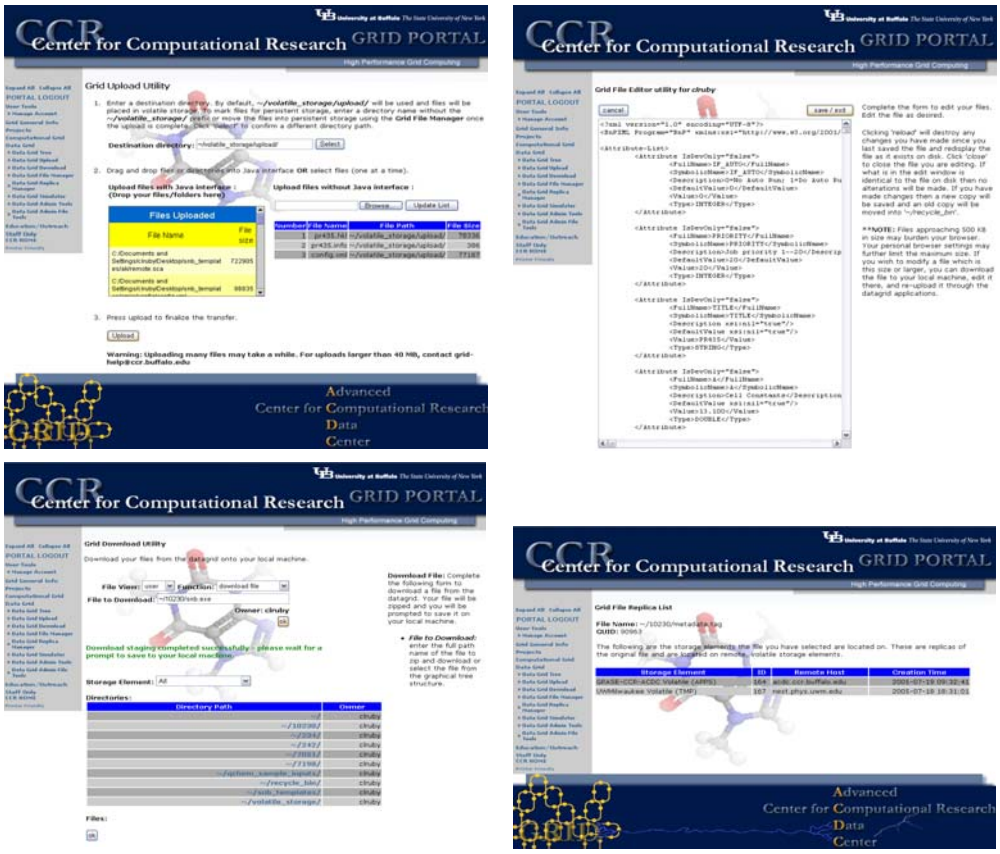


Figure 5.4: Select ACDC Data Grid user interface tools.

In addition to being able to view and manipulate files in the data grid, users may use the web tools in place to select files to be used in scientific grid applications. Also available is a data grid file download interface, or a tool for extracting data grid files from the ACDC Grid Portal by selecting files or a directory to be zipped and made available to download to a user's local machine. Figure 5.4 includes several screenshots of user tools available for managing data files on the ACDC Data Grid.

Administrators have access to other Web tools designed for managing the ACDC Data Grid behind the scenes. These tools give data grid administrators access to many capabilities, including the ability to ensure the consistency between the virtual and physical data grid representations, physically pull files into the data grid, manually migrate files, and view file

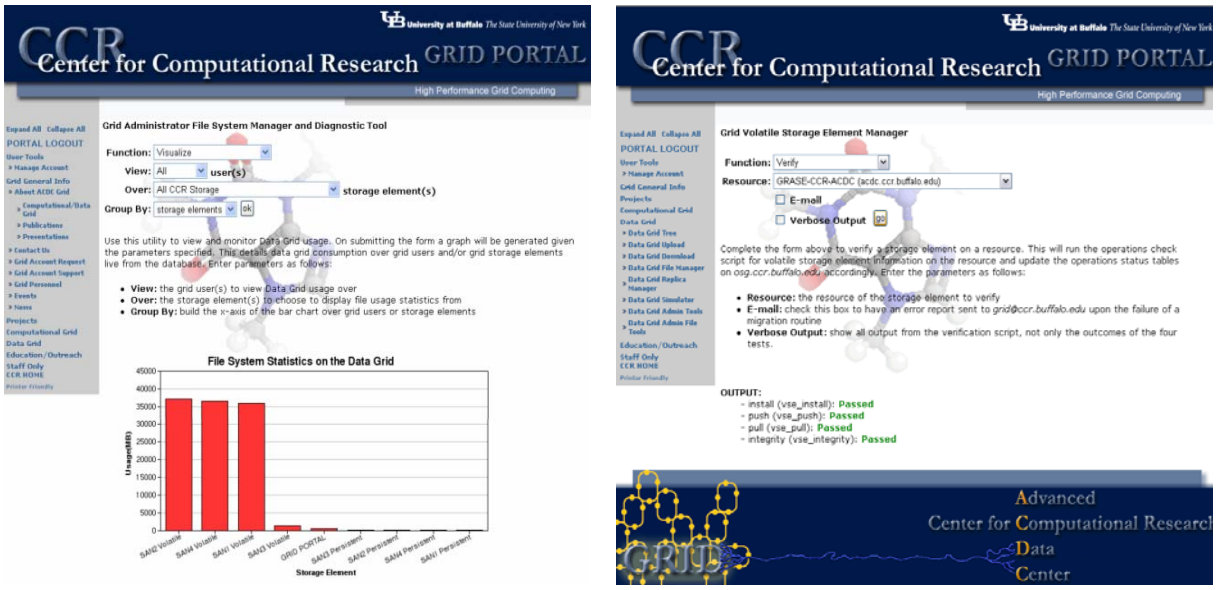


Figure 5.5: Select ACDC Data Grid administrative interface tools.

distributions across the available storage repositories, to name a few. In addition, replica storage elements can be installed, managed, or removed on storage devices remote to the CCR. Figure 5.5 shows examples of the administrative tools available for managing the ACDC Data Grid.

5.4 Results

The ACDC Data Grid is used in conjunction with the ACDC Grid and serves the data requirements of grid users submitting computational jobs to more than ten grid applications supported by the ACDC Grid in a variety of scientific fields, including environmental engineering, computational chemistry and structural biology, to name a few. The ACDC Data Grid consists of a production version and a development version, and currently supports 24 grid users and nearly 500 GB of data in over 1,000,000 data files. Since August of 2005, nearly 400,000 files totaling 70 GB of data have been staged for submission with computational jobs.

There are currently nine persistent and volatile storage elements installed throughout the

Center for Computational Research. Four 2 TB volatile storage elements and four 500 GB persistent storage elements are installed on each of four Hewlett-Packard® Storage Area Network (SAN) AlphaServer™ GS1280s. A ninth 380 GB persistent storage element is installed on the Dell® four processor hyper-threaded machine running the ACDC Grid Portal. Multiple replica storage elements are installed on compute elements throughout the Open Science Grid and TeraGrid.

5.5 Further Remarks

The ACDC Data Grid is a part of the ongoing initiative within the CCR to provide scalable and reliable data services to the ACDC Grid. Improvements to its implementation are often made to strengthen its design and ability to serve the needs of grid users and their computational jobs. One key improvement planned for the ACDC Data Grid is to further integrate replicated data files with the job submission process. Automating a procedure for utilizing replicated data files present on a remote computational resource, and thus obviating the need to include them when staging a job and its associated files for submission to the resource, would enhance the performance of the ACDC Data Grid and fully integrate the remote replica storage repositories with the storage established within the CCR. Moving the ACDC Data Grid forward in this direction is an important step in further developing its distributed design and in ensuring scalable data service to the ACDC Grid.

6 Conclusions

The Scenario Builder, the Intelligent Migrator, and the ACDC Data Grid represent three

ongoing initiatives for developing a robust and reliable data system for the Advanced Computational Data Center Grid. The Scenario Builder and the Intelligent Migrator are two applications that explore file utilizations and their effects on the performance of a data grid through generated models and simulations, whereas the ACDC Data Grid is the current implementation of the production data grid serving the ACDC Grid. In this section, we present the ways in which the Scenario Builder and the Intelligent Migrator relate to the design of the ACDC Data Grid and enable simulations of its performance. We then present our concluding remarks.

6.1 Enabling Simulations with the Scenario Builder

The Scenario Builder represents an early model of the ACDC Data Grid and contains elements from the data grid's early design. There are many similarities between these two data service initiatives, as well as a few key differences.

The virtual storage elements implemented in the Scenario Builder represent the persistent and volatile storage local to the CCR in the ACDC Data Grid. The grid portal node of the Scenario Builder is analogous to the machine running the ACDC Grid Portal, which is where computational jobs are staged with the data grid files they require. As in the virtual storage network of the Scenario Builder, the ACDC Data Grid utilizes the grid portal node as a central location for files accessed for use in computational jobs, where data may be initially present on any storage device within the CCR.

There are, however, important differences between the implementations of the Scenario Builder and the ACDC Data Grid. Files brought to the grid portal for submission with computational jobs are copied to the grid machine, not migrated as is done in the Scenario

Builder. As such, these copies may be removed from the staging areas of the ACDC Grid portal machine without damaging grid user data integrity, as the original data files remain intact in their original storage locations. Thus, there is no need for periodic migrations of data from the grid portal node to the outlying data repositories managed by the CCR.

The Scenario Builder represents early work in the initiatives at the CCR to provide robust data service to the ACDC Grid. In order to enable accurate simulations of the ACDC Data Grid, the Scenario Builder's underlying virtual storage network must be updated to accurately represent the current approach of the data grid. The storage network must be rearranged to include the ACDC Data Grid storage devices that are currently in use, and the storage and network limitations of the ACDC Data Grid must be updated within the Scenario Builder network definition in the MySQL database tables.

Though the current implementation of the ACDC Grid uses copied data and independent staging procedures for submitting files with computational jobs, and does not require migration cycles to free storage from the grid portal node, the study of these scenarios in the Scenario Builder still provides a useful examination of a storage network response to data access. As the utilization of the ACDC Data Grid grows with the ACDC Grid initiative at the CCR, it may become important in the future to examine how to efficiently distributed files to evenly utilize storage devices as the data grid nears the capacity of its networked storage repositories. If utilization continues to evolve and grow, scalability will become a greater concern in ensuring reliable data service for the ACDC Grid. The results produced by the Scenario Builder and its power for producing storage network projections may prove to be even more relevant in the future as the need for evenly distributing larger volumes of files with respect to limited bandwidth becomes an issue in the growing ACDC Data Grid.

6.2 Enabling Simulations with the Intelligent Migrator

The Intelligent Migrator is intended to model the design and implementation of the ACDC Data Grid by providing a service for simulating the performance of generated virtual data grid usage scenarios and testing the learning and prediction accuracy of agents for the storage network topology. The Intelligent Migrator is based on the replica storage network implemented for the ACDC Data Grid. Recall that the ACDC Data Grid implements storage on compute elements external to the CCR where computational jobs may be submitted. These compute elements are simulated within the Intelligent Migrator, where an agent represents a compute element and an associated storage element. It is assumed within the Intelligent Migrator simulation design that a data file located on a storage element for a compute element is accessible to computational jobs submitted to the compute element, obviating the need to submit the data file to the computational resource when staging the job for submission.

The availability of this infrastructure presents the opportunity for efficiently utilizing remote storage elements with files repeatedly submitted with computational jobs. If the files required for jobs on a compute element are stored on the storage element ahead of time, time and bandwidth are saved in submitting the job as the files do not need to be staged. The Intelligent Migrator seeks to take advantage of this design by implementing learning algorithms which utilize past file usage patterns to predict which files will be required on a compute element in advance. The structure of the MySQL database supporting the virtual data grids of the Intelligent Migrator closely mirrors the database and implementation of the ACDC Data Grid. Agents utilizing the learning algorithms developed within the Intelligent Migrator could be deployed to learn ACDC Grid user file usage patterns and submit physical replication requests

on their behalf with few changes to the Intelligent Migrator implementation.

Though the ACDC Data Grid implements remote storage and replicated data across storage elements bound to compute elements, the ACDC Data Grid is still under development in order to fully utilize file replicas located on remote storage elements. Remote storage can be installed on compute elements using the ACDC Data Grid administrative tools discussed in Section 5. However, jobs submitted to compute elements from the ACDC Grid are typically staged using the persistent or volatile data files stored locally within the CCR. As the ACDC Data Grid is a part of an ongoing initiative to deliver a robust data system to the ACDC Grid, the capabilities utilized in the Intelligent Migrator are a part of future developments for improving the current system.

As the ACDC Data Grid matures, the utilization of data replicas on remote storage elements will serve to improve the efficiency of the data grid system and take advantage of the capabilities examined in the Intelligent Migrator. The learning agent infrastructure of the Intelligent Migrator could serve as a significant contribution to the performance of the ACDC Data Grid as it is enhanced to provide more robust and efficient data service to the ACDC Grid.

6.3 Concluding Remarks

Ongoing initiatives within the Center for Computational Research to provide robust data service to the Advanced Computational Data Center have lead to the development of the Scenario Builder, the Intelligent Migrator, and the ACDC Data Grid. The Scenario Builder and Intelligent Migrator represent two tools for generating a variety of tailored virtual data grid scenarios for simulation and evaluation. Both examine file utilization and its relationship to data grid performance by simulating file accesses and their affect on the storage and network load of

the generated data environments. The Scenario Builder, representing early research in developing data services for the ACDC Grid, focuses on the direct impact of file accesses on a virtual storage network, whereas the Intelligent Migrator builds on the Scenario Builder by seeking to minimize waste by learning usage patterns and projecting future file utilization on remote storage elements. The Scenario Builder and the Intelligent Migrator each provide tools for the generation and simulation of tailored virtual data grid usage scenarios as well as for the interpretation of results with dynamic, interactive, charts through a convenient online interface.

The ACDC Data Grid serves the data requirements of the ACDC Grid by providing users with seamless access to their files distributed across multiple heterogeneous storage devices located both within the CCR and across compute elements in the Western New York Grid, the emerging New York State Grid, the Open Science Grid, and TeraGrid. The ACDC Data Grid implements data virtualization and simple storage element installation procedures and provides a scalable and robust system serving the ACDC Grid. In addition, the ACDC Data Grid provides a set of online tools for grid users. Built on components of the design of the ACDC Data Grid, the Scenario Builder and the Intelligent Migrator provide insight into potential performance issues by enabling the generation and simulation of specific usage scenarios which could arise in a physical setting.

Institutions and organizations around the world are working on grid solutions to problems with significant computational and/or data components. The growth of grid applications has spurred the need for robust and reliable data management services that are typically used to facilitate computation on a large scale. Grid research at the Center for Computational Research, as well as research in providing data service to the Advanced Computational Data Center Grid, represents our role in an international initiative to develop robust and scalable grid

infrastructures, facilitating further computational research and supporting science in the 21st century.

Acknowledgments

We would like thank Dr. Mark L. Green, Steven M. Gallo, Jonathan J. Bednasz, and Anthony D. Kew for their contributions to this work. The development of the Scenario Builder, the Intelligent Migrator, and the ACDC Data Grid are supported by NSF grant ACI-0204918 and the Center for Computational Research, SUNY-Buffalo.

References

1. Chervenak, A., et al., *The data grid: Towards and architecture for the distributed management and analysis of large scientific data sets.* Journal of Network and Computer Applications, 2000. **23**(3): p. 187-200.
2. *Grid Computing*. [webpage] 2006 April 5, 2006 [cited 2006 April 6]; Available from: http://en.wikipedia.org/wiki/Grid_computing.
3. Foster, I., *The Grid: A new infrastructure for 21st century science*, in *Grid Computing - Making the Global Infrastructure a Reality*, F. Berman, G.C. Fox, and A.J.G. Hey, Editors. 2002, John Wiley and Sons Ltd.: West Sussex. p. 50-63.
4. *Data Grid*. [webpage] 2006 March 16, 2006 [cited 2006 April 6]; Available from: http://en.wikipedia.org/wiki/Data_grid.
5. Allcock, B., et al., *Data management and transfer in high-performance computational grid environments.* Parallel Computing, 2002. **28**(5): p. 749-771.
6. *ACDC Grid Computing Services*. [webpage] 2006 April 1, 2006 [cited 2006 April 6, 2006]; Available from: https://grid.ccr.buffalo.edu.
7. *Grid Computing at CCR*. [webpage] 2004 November 3, 2004 [cited 2006 April 6]; Available from: http://www.ccr.buffalo.edu/grid/.
8. Green, M.L. and R. Miller, *Molecular structure determination on a computational and data grid.* Parallel Computing, 2004. **30**(9-10): p. 1001-1017.
9. Green, M.L. and R. Miller, *Evolutionary molecular structure determination using grid-enabled data mining.* Parallel Computing, 2004. **30**(9-10): p. 1057-1071.
10. Miller, R., et al., *SnB: crystal structure determination via Shake-and-Bake.* Journal of Applied Crystallography, 1994. **27**(1994): p. 631-621.
11. *The Princeton Ocean Model*. [webpage] 2004 [cited 2006 April 25]; Available from: http://www.aos.princeton.edu/WWWPUBLIC/htdocs.pom/.
12. *Open Science Grid*. [webpage] 2006 April 13, 2006 [cited 2006 April 13]; Available from: http://www.opensciencegrid.org.
13. *TeraGrid*. [webpage] 2006 April 13, 2006 [cited 2006 April 13]; Available from:

http://www.teragrid.org.
14. Foster, I., C. Kesselman, and S. Tuecke, *The Anatomy of the Grid - Enabling Scalable Virtual Organizations.* International Journal of Supercomputer Applications, 2001. **15**(3): p. 200-222.
15. *Globus Toolkit*. [webpage] 2006 April 3, 2006 [cited 2006 April 6]; Available from: http://www.globus.org/toolkit/.
16. *Legion Worldwide Virtual Computer*. [webpage] 2001 2001 [cited 2006 April 30]; Available from: http://legion.virginia.edu.
17. *CORBA (R) Basics*. [webpage] 2006 January 5, 2006 [cited 2006 May 1]; Available from: http://www.omg.org/gettingstarted/corbafaq.htm.
18. Roure, D.D., et al., *The evolution of the Grid*, in *Grid Computing - Making the Global Infrastructure a Reality*, F. Berman, G.C. Fox, and A.J.G. Hey, Editors. 2003, John Wiley and Sons Ltd.: West Sussex. p. 65-100.
19. *Science Grid This Week*. Image of the Week [webpage] 2006 February 22, 2006 [cited 2006 April 14]; Available from: http://www.interactions.org/sgtw/2006/0222/.
20. Kunszt, P.Z. and L.P. Guy, *The Open Grid Services Architecture, and data Grids*, in *Grid Computing - Making the Global Infrastructure a Reality*, F. Berman, G.C. Fox, and A.J.G. Hey, Editors. 2002, John Wiley and Sons Ltd: West Sussex. p. 384-407.
21. Moore, R.W. and C. Baru, *Virtualization services for Data Grids*, in *Grid Computing - Making the Global Infrastructure a Reality*, F. Berman, G.C. Fox, and A.J.G. Hey, Editors. 2002, John Wiley and Sons Ltd: West Sussex. p. 409-435.
22. *The San Diego Supercomputing Center Storage Resource Broker*. [webpage] 2006 April 7, 2006 [cited 2006 May 1]; Available from: http://www.sdsc.edu/srb/index.php/Main_Page.
23. *MCAT*. [webpage] 1998 May 11, 1998 [cited 2006 May 1]; Available from: http://www.sdsc.edu/srb/index.php/MCAT.
24. *The DataGrid Project*. [webpage] 2006 2004 [cited 2006 April 17]; Available from: http://eu-datagrid.web.cern.ch/eu-datagrid/.
25. *Grid Physics Network*. [webpage] 2006 2006 [cited 2006 May 1]; Available from: http://www.griphyn.org/.
26. *Grid Physics Network - Part 2: Petascale Virtual-Data Grids*. [webpage] 2006 2006 [cited 2006 May 1]; Available from: http://www.griphyn.org/projinfo/intro/petascale.php.
27. Bunn, J.J. and H.B. Newman, *Data-intensive Grids for high-energy physics*, in *Grid Computing - Making the Global Infrastructure a Reality*, F. Berman, G.C. Fox, and A.J.G. Hey, Editors. 2002, John Wiley and Sons Ltd.: West Sussex. p. 859-905.
28. Green, M.L. and R. Miller, *Grid computing in Buffalo, New York.* Annals of the European Academy of Sciences, 2003: p. 191-218.
29. Green, M.L. and R. Miller, *A client-server prototype for application grid-enabled template design.* Parallel Processing Letters, 2004. **14**(No. 2): p. 1001-1017.
30. *Apache HTTP Server*. [webpage] 2006 2006 [cited 2006 April 6]; Available from: http://www.apache.org/.
31. *PHP: Hypertext Preprocessor*. [webpage] 2006 April 6, 2006 [cited 2006 April 6]; Available from: http://www.php.net.
32. *JavaScript.com (TM) - The Definitive JavaScript Resource*. [webpage] 2006 2006 [cited 2006 April 6]; Available from: http://www.javascript.com/.
33. *MySQL AB: The world's most popular open source database*. [webpage] 2006 April 6,

2006 [cited 2006 April 6]; Available from: http://www.mysql.com/.
34. *GridFTP*. [webpage] 2006 2006 [cited 2006 April 13]; Available from: http://www.globus.org/grid_software/data/gridftp.php.
35. *ACDC Grid Dashboard*. [webpage] 2006 [cited 2006 April 19]; Available from: http://osg.ccr.buffalo.edu/.
36. *ACDC Operations Dashboard*. [webpage] 2006 [cited 2006 April 19]; Available from: http://osg.ccr.buffalo.edu/operations-dashboard.php.
37. *perl.com - The Source for Perl*. [webpage] 2006 [cited 2006 April 28]; Available from: http://www.perl.com.
38. Prescott, C., *Site Verify*. 2005. p. Perform grid site verification checks on remote host(s) and report results.
39. Shoshani, A., A. Sim, and J. Gu, *Storage Resource Managers: Essential Components for the Grid*, in *Grid Resource Management: State of the Art and Future Trends*, J. Nabrzyski, J.M. Schopf, and J. Weglarz, Editors. 2003, Kluwer Publishing. p. 329-347.
40. Sycara, K.P., *Multiagent Systems.* Artificial Intelligence Magazine, 1998. **10**(2): p. 79-92.
41. Lesser, V.R., *Cooperative Multiagent Systems: A Personal View of the State of the Art.* IEEE Transactions on Knowledge and Data Engineering, 1999. **11**(1): p. 133-142.
42. Hu, J. and M.P. Wellman, *Online Learning about Other Agents in a Dynamic Multiagent System.* Autonomous Agents, 1998: p. 239-246.
43. Wolpert, D.H., K.R. Wheeler, and K. Tumer, *General Principles of Learning-Based Multi-Agent Systems.* Autonomous Agents, 1999: p. 77-83.
44. Shi, Z., et al., *Agent-based grid computing.* Applied Mathematical Modelling, 2006. **30**(7): p. 629-640.
45. Mitchell, T.M., *Machine Learning*. McGraw-Hill Series in Computer Science, ed. C.L. Liu. 1997, Boston, Massachusetts: McGraw-Hill.
46. Pomerleau, D.A., *Efficient Training of Artificial Neural Networks for Autonomous Navigation.* Neural Computation, 1991. **3**(1): p. 88-97.
47. Ohlsson, M., C. Peterson, and B. Soderburg, *Neural Networks for Optimization Problems with Inequality Constraints - the Knapsack Problem.* Neural Computation, 1993. **5**: p. 331-339.
48. Tsang, E.P.K. and C.J. Wang, *A Generic Neural Network Approach For Constraint Satisfaction Problems.* Neural Network Applications, 1992: p. 12-22.
49. *Rad Inks - Rad Upload*. [webpage] 2006 April 2006 [cited 2006 April 14]; Available from: http://www.radinks.com/upload/.
50. *Sun Developer Network*. Applets [webpage] 2006 2006 [cited 2006 April 14]; Available from: http://java.sun.com/applets/.