Overdrive Controllers for Distributed Scientific Computation

Justin M. Wozniak
University of Notre Dame
Department of Computer Science and Engineering
Notre Dame, Indiana
jwozniak@nd.edu

Abstract

Large distributed computer systems have been successfully employed to solve modern scientific problems that were previously impracticable. Tools exist to bind together notably underutilized high speed computers and sizable storage resources, creating architectures that provide performance and capacity that scales with the quantity of resources invested. In various forms such as Internet computing, desktop grids, and even "big iron" grids, users and administrators find it difficult, however, to obtain aggregate systems that are more reliable than their underlying components and simple to utilize in concert. In this work, we will propose a model for controlling complex distributed systems and its application to the construction of scientific repositories.

1 Introduction

Distributed computer systems seek to overcome natural or economic limits in computer processing speed by linking multiple computers together to create more powerful but less wieldy scientific tools. A variety of existing tools may be used to parallelize numerical computation [1] or archive large data sets [2, 3]. However, the modern phenomenon of overpowered desktops has created university labs, corporate offices, and home gaming machines that contain grossly underutilized computing power and storage space. The creation of new middleware systems [4] that benefit scientific users is an extremely active modern research area. In this proposal, we outline our approach to an important practical problem: the creation of a unified, reliable, secure scientific repository from existing storage resources.

Scientific researchers employing computational systems to perform numerical processing may implement relatively efficient solutions using ordinary workstations. Potential gains in processing power attract these users to distributed computing, where they ideally undergo a *scientific software design process*. First, they must coordinate processing among the compute sites targeted by the application, ensuring that the application can be used in the larger scale computing environment and determining the resources required. Second, they must coordinate data movement and storage for the computation and integrate the storage resources with the framework. Third, they must store and maintain the results in a permanent manner.

Modern computational research that seeks to scrap together *whatever resources are available* may be performed by gluing together existing tools, writing new tools to solve small problems, or building completely new architectures and computational frameworks. Our approach to scalable, widely distributed software systems recognizes the problems associated with solutions that are either too large or too small. Essentially, we recognize the permanence of existing systems and divide the framework into two parts: an existing resource fabric and a new controller: the resource fabric is unchanged, allowing for existing systems and software to function without the controller. For a variety of reasons including simplicity and security, we constrain the controller to perform operations on the resource fabric as an ordinary user. From a software perspective, the controller may be queried or called, allowing new software to be written using controller functionality as methods. The highlight of the framework is that the controller has an internal model of the resource fabric that is used when making policy-based decisions. Such a model will borrow from the mathematical modeling and analysis of computing systems, including operating systems, distributed systems, autonomic systems, and others.

We call such a new system that has these properties a *grid overdrive controller*, and the effort of our research will be investigating and developing this model as well as employing it to solve problems in distributed storage. While the scope of the project focuses on issues related to distributed computer systems, scientific end users are the ultimate target, and we will emphasize ways in which we may aid such developers undergoing the design process.

The overdrive controller approach differs from existing solutions in several ways. While much work in grid computing has enabled resource sharing and secure remote access, a higher level entity may be used to maintain a subset of these resources. Thus the controller may focus on a specific task, such as maintaining specific replicated data sets for a certain group of collaborating chemists on a chosen subset of available resources. Control theory is tapped in the design of the software to clarify the architectural framework, numerically identify the system state and deviations, and prevent potential failures such as data loss or disk full conditions. Existing solutions have given us access and abstractions, we intend to provide utility, management, reliability, and meaning at the user level: atop an unpredictable grid or Internet computing environment.

The remainder of this document is organized in the following way. Section 2 describes the current need for robust, self-managing systems that interact with scientific users in a reliable, understandable way. In Section 3, we describe our intended areas of study. Preliminary results and a review of our work are covered in Section 4. A summary of the expected results of our research in Section 5.

2 Background

In a cooperative storage model, multiple users with multiple resources attempt to combine them into a unified system. While the traditional methods described above are still required - data access and movement for scientific jobs - new problems arise as the system takes on several new properties: the system lacks a central authority [5]; the cooperation stems from application background and is defined by the users, thus requiring administrative abilities to be granted to end users; and the complexity of such large systems requires the use of additional abstractions [6].

Specifically, our approach to scientific computing in a cooperative storage environment involves the construction of a distributed database of datasets, in which user data is replicated over the cooperative network. Such a database is influenced by previous work in replica location, as in the Replica Location Service [7]. Additionally, users of such systems typically rely upon meaningful metadata lookups given by a metadata catalog such as the MCAT [8] to obtain the logical dataset of interest.

Many other systems have used a distributed storage fabric to obtain new utility in reliability and performance. Extending the notion and method of disk striping and parity disks [9] to networks of storage services, Zebra [10] stripes data across multiple servers. A further extension is OceanStore [11], which stripes replicated data across a potentially global network of untrusted participating servers. In contrast to disk striping, full file replication is performed on untrusted servers by Farsite [12].

Grid computing attempts to solve data storage problems and more, including security, importing legacy computing technologies, managing virtual organizations, and high performance. At the core of grid computing is the ability to perform remote operations securely through the Globus Security Infrastructure [13], a public key system. Multiple organizations may be coordinated to access resources using the virtual organization abstraction [14]. Grid computing attempts to provide a high quality of service [15] while managing resources well, that is, providing job scheduling [16], negotiating network management [17], and maintaining high utilization [18] of the available hardware.

Fault management of storage systems takes three forms: fault detection, fault resiliency, and fault recovery. Fault detection on dynamic grid storage has important, as ordinary RAID is insufficient [19, 20]. The Globus Heartbeat Monitor [21], for example, employs unreliable failure detectors [22] to detect problems and trigger a correction. Replica creation is employed for data resiliency by several systems, including the Storage Resource Broker (SRB) [23] and the Grid Data Management Pilot (GDMP) [24]. Recent work has focused on reducing the cost of recovery, as in the FARM system [25], however, restoring the loss of a given amount of data will always require a transfer of that size. As a result, systems like OceanStore emphasize parity based recovery models [26].

3 Research Plan

Much scientific data may be tabulated and stored using the database model. However, scaling these databases with high throughput commodity computing systems is not well understood. Users that have massive computation requirements may submit thousands of jobs to a job scheduler, and run hundreds in parallel. We propose to employ a new database system that manages scientific data in the database model while coordinating with the requirements of weighty computational projects. The new system, called GEMS (Grid-Enabled Molecular Simulation) will combine an accessible database of user-specified scientific information with an underlying distributed system for the management of large scientific data sets. The system must meet the practical design requirements of ease of scientific use, ability to function in a dynamic resource environment, and flexible access control. Assuming that a multitude of storage devices are available, what is desired is a front-end interface and controller for an external, scalable cooperative storage network.

The construction of a large commodity scientific database of this type is a resource management problem in which a multitude of storage providers must be conglomerated into a unified resource. System disks must be managed to prevent the occurrence of disk full conditions in the

worst case, and load balanced to prevent overuse of some systems, with respect for the fact that the systems are borrowed. Replicated data sets are much more difficult for users to manage, so the system has to provide ease of use tools and allow for reasonable default settings. Users must be informed of potential storage sites for their datasets, so that they may restrict replication to trusted resources. Simplified but meaningful abstractions must be provided to allow the proper, efficient use of the distributed system when programming. While a great deal of previous work has been done properly managing local disks or coordinating data placement on remote disks, our new system intends to *manage remote disks* to balance storage loads and prevent disk full conditions.

A simple replica management system performs basic operations to detect and handle faults. Faults or storage failures may observed by querying the storage servers. Faults are handled by dynamically creating replicas, ensuring that the user specified replica count is maintained. However, in the presence of continuous server appearance and disappearance, called *churn* [27], the amount of data that would need to be transmitted to maintain user-specified replica counts would be very large, which requires analysis from the perspective of bandwidth economics, data preservation, and other constraints [28].

The system model of concern to this work uses ordinary methods to manage an existing system. In the mature field of feedback control, this existing system is called the *plant* [29]. A *controller* is used to maintain the plant in accordance with user requests. We intend to draw from control models to optimize the state of the system when responding to the external disturbances. A diagram of the control model to be developed by our system is shown in Figure 1.

4 Preliminary Results

We began our research in this area by investigating scientific databases for simple, massively parallel batches of parameter sweep tasks [30]. This work allowed for a deeper understanding of typical scientific usage of grid systems, as well as represention of cataloged application data.

A cooperative scientific database called GEMS, for Grid-Enabled Molecular Simulation, has been constructed and is installed on three servers at the University of Notre Dame. Each system makes use of the existing Chirp storage network and thus has access to around 250 storage sites. The software is currently used for molecular dynamics research but could be easily and effectively used by other data intensive computational tasks.

The current system meets many of the requirements previously discussed. From a scientific perspective, it offers a unified view of the storage network to enhance the ability to utilize a variety of remote storage resources in an abstract way, and provides a public, searchable metadata tagging system that may be used to catalog application-specific input and output parameters. It eases the user management of distributed data sets by implementing a "fire-and-forget" replication model in which users may submit data and neglect to fine-tune the replication process, performs active replica monitoring and management, and promotes resource load balancing by observing the state of the storage layer when creating replica sites.

GEMS enhances the ability of users to create distributed applications with a novel combination of functionality. It provides a transaction model for the importation or creation of new data by creating reservations and accepting data set committal at a later date, and it integrates with a personal virtual filesystem [6] to create a convenient I/O subsystem. GEMS maintains replica locations and may derive locality from user-specified topology information on a record-by-record basis. Most importantly for users of distributed job schedulers, GEMS supports computation through client utilities that may create jobs in a variety of methods, including local execution, active server execution, or job scheduler execution. An overview of the GEMS design is shown in Figure 2 a).

Additionally, the system meets the needs of both storage providers and storage consumers [31]. It allows users to gain access to remote machines with survivability delivered through the use of replication. It also respects storage providers by allowing them to restrict access to unknown users, evicting data, and monitoring the usage of their system.

As GEMS relies upon a borrowed, relatively unmanaged storage fabric, storage faults must be handled on a continuous basis. The loss of storage sites or individual files must be detected and promptly rectified to prevent the complete loss of a record. GEMS currently uses an architecture inspired by feedback control systems [32] to handle storage anomalies, which are treated as perturbations from the desired system state. While resources are consumed when performing large replications, other smaller datasets may be lost. Thus, replication must be both prioritized, efficient, and broad to prevent data loss but conservative to avoid overconsumption of resources in the presence of short term outages. We have adapted a model from control system theory using probabilistic concepts to derive a ranking system to handle observed storage problems.

As discussed above, grid computing is distinguished by the ability to work across administrative boundaries, requiring an ability to gain access to remote resources over new protocols. The GEMS system implements an extremely abstract representation of user identities, allowing the storage providers and data owners to coordinate resource sharing and collaboration without having to directly authenticate to

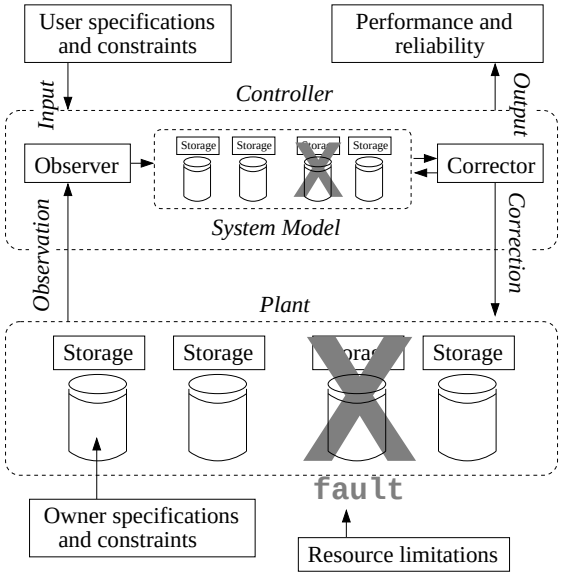

Figure 1. Schematic of a cooperative storage controller (GEMS). The controller actively observes the state of the system and updates its internal model of the system status. Corrections are made in a well defined way, bringing the the system into a state that satisfies user requirements.

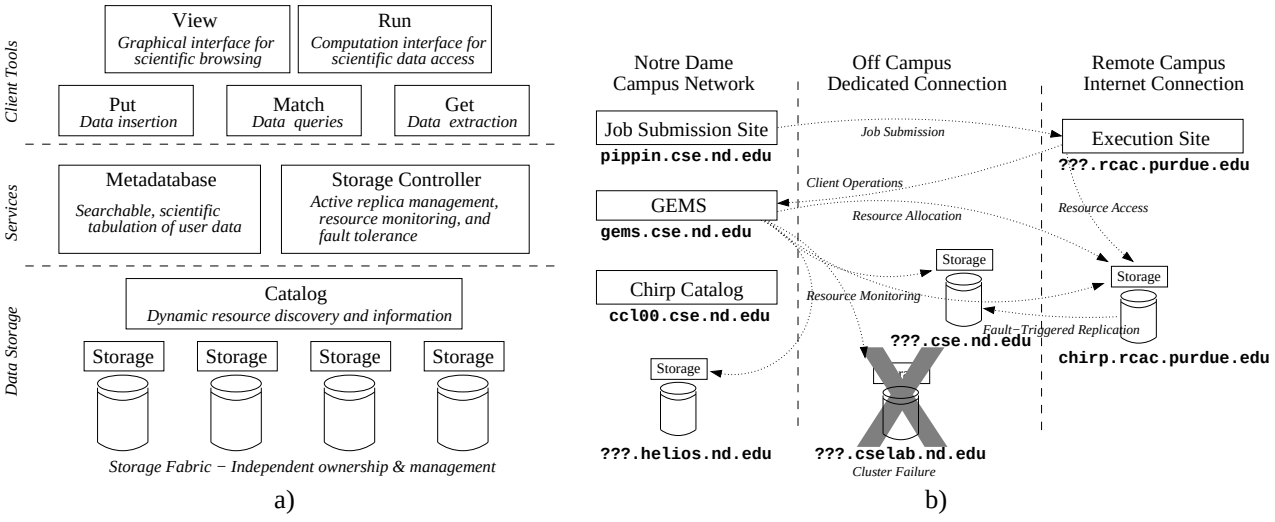

Figure 2. a) Key components of the GEMS system. Automatic services include metadata queries, replica location, and management. The underlying system consists of external grid-enabled file servers. b) Usage scenario of the GEMS system. Local, off-site, and remote campus resources are monitored for failures and automatic replica creation results to ensure dataset survival. Remote jobs executing in a Condor environment obtain free storage space or nearby dataset replica locations via GEMS client tools.

GEMS [33]. A rendition protocol was introduced by which a client may authenticate to the system indirectly through a storage site in a situation in which authentication would otherwise be impossible. This allows for a greatly enhanced administration environment in which users which are essentially unknown to GEMS may configure and share new storage networks, maintaining their own access policies, while gaining the benefits of replication and storage management.

GEMS is intended to be utilized by scientific jobs as they execute on distributed scheduled resources. Current work focuses on refining the methods used to access GEMS storage by jobs running in these existing systems as shown in Figure 2 b). The emphasis here is that GEMS simplifies the distributed storage problem to the point that storage servers may be as widely distributed as the compute servers. By scheduling jobs to access on-site replicas we eliminate the turnaround penalty caused by data staging, thus, replication now has a double benefit: storage reliability and data prestaging.

5 Summary

The following list summarizes the projected contributions of this work.

- An enhanced understanding of dynamic replica systems, mathematical models that may be used to interpret them, policy that makes effective use of them, and software that manages them through the maintenance of the system model;
- A complete model for overdrive grid controllers that enhance the utility of existing systems by providing new abstract functionality and actively maintaining underlying grid resources;
- New software packages in the GEMS and East project domains.

The GEMS home page is:
`http://gipse.cse.nd.edu/GEMS`.
GEMS is available at:
`http://sourceforge.net/projects/gems-nd`.

6 Notes

Justin Wozniak is in his third year of the PhD program at the University of Notre Dame, working with advisor Dr. Aaron Striegel. Dissertation defense is planned for March 2008.

Another application of the overdrive grid controller model is in deadline driven job scheduling and policy enforcement [34], implemented in software called East, which was not covered here for lack of space.

References

[1] Antoine Petitet, Susan Blackford, Jack Dongarra, Brett Ellis, Graham Fagg, Kenneth Roche, and Sathish Vadhiyar. Numerical libraries and the Grid. *The International Journal of High Performance Computing Applications*, 15(4), 2001.

[2] Randolph Y. Wang and Thomas E. Anderson. xFS: A wide area mass storage file system. In *Workshop on Workstation Operating Systems*, 1993.

[3] Zeyad Ali and Qutaiba Malluhi. NSM: A distributed storage architecture for data-intensive applications. In *Proc. Mass Storage Systems and Technologies*, 2003.

[4] Philip A. Bernstein. Middleware: A model for distributed services. *Communications of the ACM*, 39(2), 1996.

[5] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *Proc. Symposium on Operating Systems Principles*, 2001.

[6] Douglas Thain, Sander Klous, Justin Wozniak, Paul Brenner, Aaron Striegel, and Jesus Izaguirre. Separating abstractions from resources in a tactical storage system. In *Proc. Supercomputing*, 2005.

[7] Ann L. Chervenak, Naveen Palavalli, Shishir Bharathi, Carl Kesselman, and Robert Schwartzkopf. Performance and scalability of a replica location service. In *Proc. High Performance Distributed Computing*, 2004.

[8] Gurmeet Singh, Shishir Bharati, Ann Chervenak, Ewa Deelman, Carl Kesselman, Mary Manohar, Sonal Patil, and Laura Pearlman. A metadata catalog service for data intensive applications. In *Proc. Supercomputing*, 2003.

[9] David Patterson, Garth Gibson, and Randy Katz. A case for redundant arrays of inexpensive disks (RAID). In *Proc. Management of Data*, 1988.

[10] John Hartman and John Ousterhout. The Zebra striped network file system. In *Proc. Symposium on Operating System Principles*, 1993.

[11] Emil Sit, Andreas Haeberlen, Frank Dabek, Byung-Gon Chun, Hakim Weatherspoon, Robert Morris, M. Frans Kaashoek, and John Kubiatowicz. Proactive replication for data durability. In *Proc. International Workshop on Peer-to-Peer Systems*, 2006.

[12] Atul Adya, William Bolosky, Miguel Castro, Gerald Cermak, Ronnie Chaiken, John Douceur, Jon Howell, Jacob Lorch, Marvin Theimer, and Roger Wattenhofer. Farsite: Federated, available, and reliable storage for an incompletely trusted environment. In *Proc. Symposium on Operating Systems Design and Implementation*, 2002.

[13] Ian Foster, Carl Kesselman, Gene Tsudik, and Steven Tuecke. A security architecture for computational grids. *Conference on Computers and Security*, 1998.

[14] Laura Pearlman, Von Welch, Ian Foster, Carl Kesselman, and Steven Tuecke. A community authorization service for group collaboration. In *Proc. International Workshop on Policies for Distributed Systems and Networks*, 2002.

[15] Ian Foster. What is the Grid? A three point checklist. *GRIDToday*, 2002.

[16] Karl Czajkowski, Ian Foster, Nick Karonis, Carl Kesselman, Stuart Martin, Warren Smith, and Steven Tuecke. A resource management architecture for metacomputing systems. *Lecture Notes in Computer Science*, 1459, 1998.

[17] Ian Foster, Markus Fidler, Alain Roy, Volker Sander, and Linda Winkler. End-to-end quality of service for high-end applications. *Computer Communications*, 27(14), 2004.

[18] James Frey, Todd Tannenbaum, Ian Foster, Miron Livny, and Steven Tuecke. Condor-G: A computation management agent for multi-institutional grids. *Cluster Computing*, 5(3), 2002.

[19] Qin Xin, Ethan Miller, T. Schwarz, Darrell D. E. Long, Scott A. Brandt, and Witold Litwin. Reliability mechanisms for very large storage systems. In *Proc. Mass Storage Systems and Technologies*, 2003.

[20] Guillermo A. Alvarez, Walter A. Burkhard, and Flaviu Cristian. Tolerating multiple failures in RAID architectures with optimal storage and uniform declustering. In *Proc. International Symposium on Computer Architecture*, 1997.

[21] Paul Stelling, Cheryl DeMatteis, Ian T. Foster, Carl Kesselman, Craig A. Lee, and Gregor von Laszewski. A fault detection service for wide area distributed computations. *Cluster Computing*, 2(2), 1999.

[22] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2), 1996.

[23] Arcot Rajasekar, Michael Wan, Reagan Moore, George Kremenek, and Tom Guptill. Data grids, collections and grid bricks. In *Proc. Mass Storage Systems and Technologies*, 2003.

[24] Heinz Stockinger, Asad Samar, Bill Allcock, Ian Foster, Koen Holtman, and Brian Tierney. File and object replication in data grids. In *Proc. High Performance Distributed Computing*, 2001.

[25] Qin Xin, Ethan L. Miller, and S.J. Thomas J. E. Schwarz. Evaluation of distributed recovery in large-scale storage systems. In *Proc. High Performance Distributed Computing*, 2004.

[26] Hakim Weatherspoon and John Kubiatowicz. Erasure coding vs. replication: A quantitative comparison. In *Proc. Workshop on Peer-to-Peer Systems*, 2002.

[27] Bram Cohen. Incentives build robustness in BitTorrent. In *Proc. Workshop on the Economics of Peer-to-Peer Systems*, 2003.

[28] Srikumar Venugopal, Rajkumar Buyya, and Kotagiri Ramamohanarao. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Computing Surveys*, 38(1), 2006.

[29] Stanley Shinners. *Modern Control System Theory and Design*. Wiley Interscience, 1998.

[30] Justin M. Wozniak, Aaron Striegel, David Salyers, and Jesus A. Izaguirre. GIPSE: Streamlining the management of simulation on the grid. In *Proc. Annual Simulation Symposium*, 2005.

[31] Justin M. Wozniak, Paul Brenner, Douglas Thain, Aaron Striegel, and Jesus A. Izaguirre. Generosity and gluttony in GEMS: Grid-Enabled Molecular Simulation. In *Proc. High Performance Distributed Computing*, 2005.

[32] Justin M. Wozniak, Paul Brenner, Douglas Thain, Aaron Striegel, and Jesus A. Izaguirre. Applying feedback control to a replica management system. In *Proc. Southeastern Symposium on System Theory*, 2006.

[33] Justin M. Wozniak, Paul Brenner, Douglas Thain, Aaron Striegel, and Jesus A. Izaguirre. Access control for a replica management database. In *Proc. Workshop on Storage Security and Survivability*, 2006.

[34] Justin M. Wozniak, Yingxin Jiang, and Aaron Striegel. Effects of low-quality computation time estimates in policed schedulers. In *Proc. Annual Simulation Symposium*, 2007.