Access Control for a Replica Management Database

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

Paul Brenner
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, Indiana
pbrenne1@nd.edu

Douglas Thain
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, Indiana
dthain@nd.edu

ABSTRACT

Distributed computation systems have become an important tool for scientific simulation, and a similarly distributed replica management system may be employed to increase the locality and availability of storage services. While users of such systems may have low expectations regarding the security and reliability of the computation involved, they expect that committed data sets resulting from complete jobs will be protected against storage faults, accidents and intrusion. We offer a solution to the distributed storage security problem that has no global view on user names or authentication specifics. Access control is handled by a rendition protocol, which is similar to a rendezvous protocol but is driven by the capability of the client user to effect change in the data on the underlying storage. In this paper, we discuss the benefits and liabilities of such a system[1].

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration—*Security, integrity, and protection, Data warehouse and repository*

General Terms

Management, security

Keywords

GEMS, chirp, distributed access control, rendition protocol

1. INTRODUCTION

Recent years have seen an increase in deployment of distributed computation systems in which widely distributed commodity hardware may be cataloged into a unified, high-utilization system. Such widely distributed systems create a massively parallel computing infrastructure, capable of processing vast amounts of user data to serve a variety of applications. Massive batches of such jobs will overwhelm a centralized storage service, necessitating an alternative storage architecture that scales with the computation sites.

A variety of options exist to parallelize and distribute storage over clusters or grids, but several technical and organizational issues must be considered. Current computational environments span multiple administrative domains, such as two or more university clusters combined into a unified resource to increase the utilization of compute resources. A current authorization strategy in such systems is to execute these jobs as a nearly privilegeless anonymous user that has access only to pre-specified remote files over the network.

Distributing storage across the compute system in such a setting presents additional difficulties. Users may be willing to allow the anonymous third-party system to execute jobs on their behalf, but will need to be able to access the storage directly upon data creation. A strategy to solve this problem would be to create a master user list and replicate it to all storage sites, allowing user access over a pre-defined protocol. Managing users in such a system becomes extremely burdensome as the user list would contain members from all collaborating institutions.

More advanced methods have been proposed including grid authentication protocols that take into account the existence of distributed administrative domains. However, choosing one such protocol implies that all users must agree to the protocol, and would find it difficult to fall back on simpler, localized, pre-existing schemes.

Consider the case of an *ad hoc* collaboration between researchers at two distant universities. Each contributes storage servers to the project, and one of the researchers installs a replica management system to synchronize data sets between the sites. This researcher may be unwilling or unable to provide accounts for all eligible users from the other university to interact with the management database.

Similar problems arise in many experiences with cooperative computing, and the premise of the approach presented here originates from typical assumptions and properties of this situation. *Users and authentication methods employed are considered secondary to the ability of users to modify the data sets and servers involved, and the ability to authenticate via a given protocol as a given user is less important than the ability to access a given data set at a given site.* This observation motivates a system that operates at high level, independent of protocols and user names, and can stay within its purpose: the management of distributed data sets. The

[1]This research was supported in part by the National Science Foundation through the grant: NSF DBI-04500667.

StorageSS'06, October 30, 2006, Alexandria, Virginia, USA.

centralized service then acts as little more than a guide, coordinating interaction among users and storage sites. These user-site pairs handle security in a pairwise way, and system changes are propagated up to the management system. The contribution of this paper is to discuss a framework for reasoning about this type of access control system, present a protocol, and discuss the implementation of the method in ongoing scientific work.

As a solution to our considered problem, client tools could be employed that interact with the *local* storage service as a method of proving their identity to the greater system. This allows researchers to simply administer their own machines instead of a grid, and allows the replica system to simply manage replicas instead of users.

In the remainder of this paper, we briefly describe the importance and utility of the replica environment of interest to this work in the following section. We then describe the access control mechanisms available in Section 3. The specifics of the protocol are presented in Section 4, and a brief case study of our target application is discussed in Section 5. Related work is noted in Section 6, and we conclude in Section 7.

2. MOTIVATION FOR COMMODITY SHARED STORAGE

Shared replica management systems have been designed to meet the storage requirements of users requiring a variety of functionality. Users benefit from increased storage space: especially short term storage space, as the shared workspace may increase utilization of the underlying systems. Additionally, replicated data sets are more resilient to hardware failure or loss, as replicas may serve as backup copies. User groups that desire to publish their data inside a virtual organization benefit from catalogued replica systems, which provide a searchable catalog of metadata and allow for data sharing and reuse. Properties of the replica management system relevant to this work have been described previously [17].

The replica management system involved in this work is designed to operate on a network of storage sites that corresponds to a rough overlay of the available computation sites. Client tools include a virtual filesystem adapter which allows for the user to choose the *compute site*, the *input source site*, and the *output destination site* independently. The replica management system may be combined with these compute tools to locate data sources, find sites to safely store output, and recommend a computation site. Once a computation site has been chosen, user-supplied cluster topology information may be used to obtain local or nearby access to data services.

Such systems have a certain typical set of user requirements and assumptions. First, users must basically trust the machines that they are borrowing, the network, and the administrators from whom they obtain these resources. Specifically, our architecture assumes that if a user is willing to delegate computation to a site, then that site may be trusted to serve the output data. Additionally, there is an assumption that the system will be in a partial failure state at all times, with some machines unavailable for a variety of reasons. Middleware computation systems attempt to build this fragile infrastructure into a useful resource through fault-tolerant checkpointing and job restarts; storage solutions such as ours presented here attempt to create a viable, scalable, and secure storage solution through replication and replica management. Finally, the data sets involved are of relatively low value, for example, data transmission is typically performed in the clear. Note that this does not allow us to assume that the data sets are intended to be publically readable. The protocol described in this paper attempts to utilize this fundamentally unreliable and uncontrollable resource fabric into a secure system for communication between previously unauthenticable users and a replica management database while keeping data secure and providing exceptional flexibility.

3. PROPERTIES OF ACCESS CONTROL IN A REPLICA MANAGEMENT SYSTEM

Our relevant system architecture consists of a network of storage devices widely distributed and independently maintained. The storage sites may be configured in various ways, deploying different subsets of the available connection protocols. The storage providers desire to collaborate - to obtain the benefits of a replica system - and thus allow limited access to a centralized service replica management system, or RMS. Users may be able to authenticate to some subset of the available storage sites using one or more protocols. The set of users may be multiplied in a set of pairs $(protocol, name)$, called the set of subjects. Each real-world user corresponds to multiple subjects, based on the accounts and account types available. However, the RMS is not aware of any global list of subjects in advance. Additionally, users may desire to access the data that they may store in the system over more than one protocol.

Centrally, the RMS catalogs a large number of storage devices as they advertise their resources to the system, but cannot manage the lists of users from various domains. The services offered are unable to authenticate users over the network. The immediate result is a simple read-only lookup service that guides users to data sources.

The storage sites implement a variety of authentication protocols, and are drawn from multiple administrative domains. Users may access data in the system by following a metadata lookup to the RMS with a connection to an appropriate replica site, after which they may obtain the required data file. In a large, multiple organization system, users desire to gain access to multiple domains using multiple protocols, which is acceptable because the user/server relationships may be defined by the users.

To effect change in the system, such as to delete a record or to modify a metadata entry for a record, two systems must be protected: the metadatabase and the servers. This represents a challenge to the multiple domain model, because the metadatabase and the storage server may have different conceptions of who the user is. For example, a user may be able to authenticate to a nearby storage site that contains a replica of a data set that the user wishes to delete, but the user is unable to authenticate to the centralized server- a necessary ability to delete a replica set that may be owned by another user and distributed widely.

No global list of users is available to the RMS, since the RMS is unable to implement an authentication test for each subject. However, a user may demonstrate the ability to modify a stored data set by interacting with a storage site. Since the RMS is able to observe such interaction, an indi-

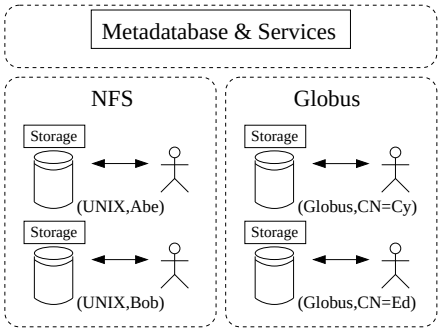

Figure 1: Read-only access controlled independently by local storage servers.

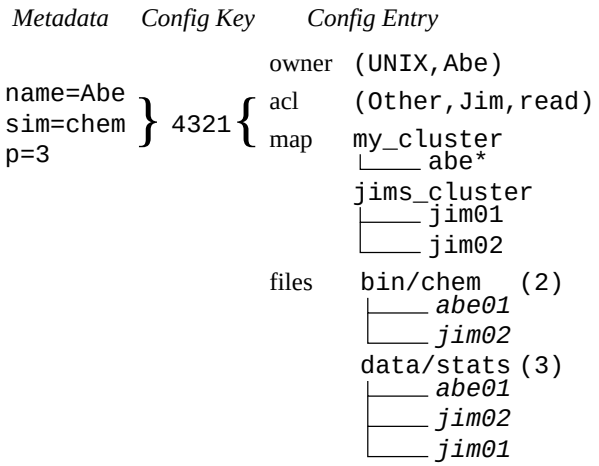

Figure 2: Metadata layout in a replica management system. A user-specified number of copies of each file are stored across the storage map.

rect method of authentication is possible. By demonstrating the ability to change the storage system, the user has authenticated in a meaningful way to the RMS.

4. SYSTEM SPECIFICS

In this section, we develop the specific capabilities of the software components relevant to this work. The system layout consists of a centralized replica management system that can acts as an active replica controller [18], as well as admit and evict storage resources, respond to storage failures, allocate space for new data sets intelligently, propagate and maintain access control lists (ACLs), respond to user queries regarding metadata tags that are associated with all data sets, and act as a replica location service, since replica locations may migrate for a variety of reasons. The storage layer [15] stores the raw data files supplied by the user in the directory structure originally given, enforces the ACLs supplied by the RMS, authenticates users over a variety of protocols. Client tools are provided to perform elementary operations with the system as a unified resource, such as put, get, match, etc.

The data structure of a data set stored in the system is shown as a diagram in Figure 2. Each data set, or *config*, is indexed by a numeric key, but is commonly accessed by a unique metadata lookup query. Configs have an owner and ACL as shown: owners always have full access to the config and may grant full or limited access to other subjects. The config consists of any number of files with their path information, as in an archive. The *storage map* indicates which storage sites are eligible to receive and serve the data files. Additionally, the map indicates the cluster topology to the replication service. The replication service uses this information to split replica locations among available clusters. The map is also used when retrieving data to obtain a nearby replica. This structure increases the performance of data access, and increases data survivability when whole clusters may be offline.

The map takes on critical importance in the security of the data set. Users may use the replica servers as rendezvous locations when gaining access to the config in question, so the servers must be trusted *by the specific user* to enforce the ACLs appropriately.

The system allows the users to create virtual workspaces in which they create storage and grant access control to other users. *The benefit of the methods described in this paper is that the centralized service does not have to maintain a list of these users, or even be able to authenticate them directly over their desired protocol.*

Different operations require different types of authentication in this system. There are four authentication methods that are implemented by the RMS server.

- *none*, to access the public search facility;
- *insert*, to insert a new config into the system;
- *config*, to modify or delete a config;
- *admin*, to rewrite the configuration of the entire system.

The most common use of an archival system is to consume data from it to perform new computation. While the data files are protected on the storage servers, locating this data is a publically available service. Searches may be used to map metadata to configs, or configs to file replica locations. This is equivalent to simply observing a resource catalog.

Data insertion is performed by client tools after authentication using the rendition protocol. The client simply contacts the metadatabase with a request for a challenge, and specifies an desired authentication method. The response to the challenge takes is a rendezvous location on a storage server that the metadatabase administrator and the user trust to allow insertion into the system. The rendezvous location takes the form of a tuple containing an appropriate storage server, a new config key to uniquely identify this data set, and randomly generated code number for this transaction: (*host*, *config*, *code*). The server creates a directory on the given host named `/<config>/RDVS/<code>`, and sets the ACL on this directory to allow the client write access. The client then contacts the storage server and creates a marker file at the given path. Upon completion, the metaserver is notified over the channel, the result is verified, and the channel may be thought of as authenticated.

One the config has been created, the ACL is generalized to meet the owner's request allowing access to other eligible subjects. Such subjects do not need to authenticate to the

centralized service as described above. The replication process, coordinated by the metadatabase, propagates the ACL to other eligible storage sites along with the data files.

Metadata modification or config deletion is performed upon satisfaction of the config rendition protocol. In this protocol, the client contacts the metadatabase server and indicates which config is to be affected. The reply takes the form given above, but the host given is selected from the storage servers that currently contain replicas of the stored data. Additionally, the rendezvous directory is placed within the directory allocated for that config. Thus the protocol takes on an element of realism: clients that can demonstrate access to the stored data files are granted access to modify the metadata associated with the config.

The authenticated subject is not a full representation of the user subject in the traditional sense of a login. Since the subject has been authenticated *for* a certain config, the authenticated subject must be considered a limited subject (*protocol*, *name*, *config*), as access has only been granted by the system for the config in question.

If a certain storage site is authorized to serve replicas of a given config, the storage site effectively acts as an authentication authority for the user with respect to the configs which it stores. Thus the site effectively *speaks for* the user with respect to that config. As a result, users must construct storage maps appropriately, weighing the benefits gained from "wider", more distributed maps with the security risks involved in spreading data far and wide. In typical cases, this is equivalent to or easier than constructing a matchmaking script like those used in common computation systems. When the storage network *is* the computation network this is elementary.

A last protocol is available to allow administrators to modify behavior of the whole system during operation, allowing for remote administration. The system configuration specifies a rendezvous location that may be used to satisfy the protocol, and upon satisfaction of the protocol, a new configuration file may be fed into the server.

5. APPLICATION: MOLECULAR DYNAMICS ON TWO UNIVERSITY NETWORKS

The university network is a commonly used tool for scientific research. Many universities have clusters or laboratories with a variety of machines connected over a relatively fast internal local area network. These clusters may be combined into useful high throughput, high utilization systems with the appropriate software. Here we provide an example to motivate the usefulness of the new system.

An engineering building at the University of Notre Dame, which contains over 200 Linux and Solaris machines, has been combined into a Condor [9] system. The default Condor installation package used on campus includes the Chirp [15] file services, totalling over 7 TB of available distributed storage spread over the same machines. Additionally, the University is served by a centralized AFS [8] installation. Users have a large pool of computing and storage resources at their convenience, however, permanent storage and physical file location management must be handled by the individual users.

A replica management system called GEMS [17] that implements the model presented in this paper is used to combine all 200 of these storage servers into a unified system. Since all the storage hosts in question share a network file system, the UNIX authentication protocol is a common choice. Users will typically allow hostname authentication as a fallback access method, and Globus authentication is possible via a local certificate authority.

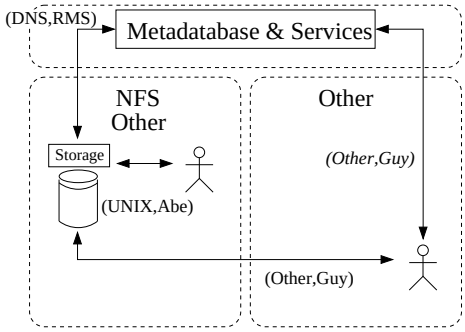

Figure 4: Using a rendezvous point to authenticate a user that the RMS cannot directly authenticate.

The Condor system has been combined with Condor systems at nearby universities to increase the opportunities for resource sharing and collaboration. To unify the storage into a single logical system as well, users would expect to have to be able to authenticate under some uniform protocol. However, the GEMS protocol allows users to interact with the system by authenticating through only the subset of servers that are relevant to the data set in question, relieving the administrative burden of managing users in a global way.

The GEMS system allows replicas of user data sets to be automatically replicated to remote computation sites. Users may use a simple storage map to inform the system that the `*.nd.edu` and, for example, `*.purdue.edu` computation and storage resources are eligible to be used. The map would be used to ensure that data access would be localized to a campus network.

In practice, the GEMS installation could not be immediately configured to authenticate users from multiple universities. However, as individual stake holders allow each other access to storage sites at their respective locales and develop a complex system of user groups and virtual organizations, GEMS is able to grow in parallel with the system, relying on storage providers to serve as the authority on storage access, which is the objective all along.

A more abstract example of the protocol in action is shown in Figure 4. In this special case, we demonstrate the additional ability of the system to authenticate users to whom access has been granted on a pre-existing config. In this case, user Abe has stored a config on his server, but has deployed an ACL that allows user Guy full access using the "Other" protocol. Abe's environment is capable of authenticating UNIX users and Other users, however, the metadatabase is unable to authenticate Other users. The config in question has been replicated, perhaps to other domains, so deleting the config must be performed at the RMS level. Since Abe has allowed Guy full access, Guy should be able to perform deletion but is unable to gain config access at the RMS level. However, the client tool that performs deletion using the rendition protocol is able to render a marker file at a rendezvous location on the storage site. This is checked

Rendition Procedure for Storage Access

	Client operations	*Metaserver operations*
1.	Client obtains an anonymous secure channel to the metaserver.	
2.	Client requests access to config c as user n.	
3.		Metaserver issues a rendezvous challenge involving a host h in the map associated with c.
4.	Client authenticates directly to host h using method m.	
5.	Client satisfies challenge.	
6.	Client notifies metaserver that challenge is satisfied.	
7.		Metaserver inspects host h for completion of challenge.
8.		Metaserver notifies client that inspection was successful; the channel is authenticated as $m : n$ *for* c.
9.	Client transmits metadatabase operation regarding config c to metaserver.	

Figure 3: Outline of the Rendition Protocol

by the RMS, which authenticates over a hostname protocol. Thus the connection between the RMS and Guy is indirectly authenticated, as indicated by the italicized subject label *(Other, Guy)*.

6. RELATED WORK

Many grid applications currently rely on Globus [4, 7] for security. A certificate authority model is offered by the Globus Security Infrastructure (GSI) [5]. This model stresses the primality of local access control mechanisms, and maps global users to local users. An existing replica location system offered by the Globus system is the Replica Location Service (RLS) [3] which provides the ability to map logical file names to physical file locations. Systems built upon the RLS must manage user metadata externally.

The Storage Resource Broker (SRB) [11] allows for the construction of data grids, which combine a variety of storage systems into a unified grid. This system provides multiple user-configurable replication techniques. Appropriate metadata is stored in a database [13]. Access control is managed through a distributed user-password list or through GSI.

Either of the above storage systems could be used to support users running biomolecular simulations with the construction of an appropriate client to help collocate the computation and data services. However, as discussed below, the Parrot/Chirp [15] system upon which GEMS relies allows for a great deal of flexibility and ease of use when actually running simulation scripts because of the location independence and virtual filesystem that Parrot creates. Additionally, while one would expect to have to use an external job submission system such as Condor [9], GEMS does not require this because of the server-side execution facility provided by Chirp. Chirp provides a comprehensive ACL system and provides authentication for subjects over Kerberos [10], Globus, UNIX, and hostname.

The Legion system [16] creates a large-scale computing resource. The data services in Legion are designed to scale, making massive use of parallelism to provide enormous aggregate bandwidth to jobs running in the object-based system. Objects in Legion are responsible for their own access control, which is similar to the model presented here. Legion provides a single system image, which is different from our model in which users specify a subset of the system to use: thus, Legion relies heavily on encryption to protect objects from unauthorized access.

Additionally, a great deal of security is provided by network attached secure disks [6]. This technology allows a high performance disk to perform local access control, however, these model does not directly take into account a replica management model in which changes must be propagated to a higher level system.

A proposed API for distributed computing is the Generic Authorization and Access-Control API (GAA-API) [12], which greatly extends access control functionality and attempts to unify authorization and authentication over various protocols. For example, a unified syntax is derived to clearly identify subjects as a combination of user type, authentication type, and user name. The system can thus *understand* and authenticate each possible subject by calling the appropriate routine. In our model, subjects are understood in as much as the storage sites can understand them, and the central system simply accepts the subject as a name.

Another storage system designed for the application area of molecular dynamics is BioSimGrid [14]. Centering on a simulator-independent scientific database, BioSimGrid provides tools to perform analysis on its libraries of simulation data. The software architecture combines a standard database with an underlying SRB storage system. Computation in BioSimGrid is centered around application-specific analysis tools which are run on the centralized system.

7. CONCLUSION

> **ren · di · tion** *n.* 1. The act of submitting for approval [1]; 2. An explanation of something that is not immediately obvious [2].

The protocol in this paper results from several observations about the properties of such widely distributed replica management systems: the wide variety of users, access methods, and administrative domains; the wall of separation between centralized metadata and user data files; and an implicit trust of storage providers, that storage providers will keep the data sets secure. The purpose of replication here is not for security purposes but for reliability, availability, and performance purposes. This model is consistent with the security model of computation systems because in such systems users must trust some number of sites to create correct data sets.

The new method presented herein attempts to authenticate users in a limited way, avoiding difficulties caused by squeezing users into a single authentication protocol, as well as the administrative challenge of managing all users and resources as a whole. Thus the user identity of a client connection to the system is never "immediately obvious", and an explanation is provided in the form of a file operation submitted for approval.

A replica management system called GEMS[2] has been constructed to improve the utility of large scale compute and storage networks for researchers performing molecular dynamics simulation. Such research is commonly performed on university networks, and can be migrated to Internet computing on volunteered resources.

8. ADDITIONAL AUTHORS

Additional authors: A. Striegel (email: `striegel@nd.edu`), and J. A. Izaguirre (email: `izaguirr@nd.edu`).

9. REFERENCES

[1] *The American Heritage Dictionary of the English Language, Fourth Edition*. Houghton Mifflin Company.

[2] *WordNet 2.0*. Princeton University.

[3] A. L. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman, and R. Schwartzkopf. Performance and scalability of a replica location service. In *Proceedings of the International Symposium on High Performance Distributed Computing*, 2004.

[4] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11, 1997.

[5] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. *ACM Conference on Computers and Security*, 1998.

[6] Garth A. Gibson and Rodney Van Meter. Network attached storage architecture. *Communications of the ACM*, November 2000.

[7] The Globus Alliance. `http://www.globus.org` .

[8] J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham, and M. West. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems*, 6, 1988.

[9] M. Litzkow, M. Livny, and M. Mutka. Condor - A hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, 1988.

[10] B. Clifford Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32, 1994.

[11] A. Rajasekar, M. Wan, R. Moore, G. Kremenek, and T. Guptill. Data grids, collections and grid bricks. In *20th IEEE / 11th NASA Goddard Conference on Mass Storage Systems and Technologies*, 2003.

[12] Tatyana Ryutov, Grig Gheorghiu, and Clifford Neuman. An authorization framework for metacomputing applications. In *Proceedings of Cluster Computing*, 1999.

[13] G. Singh, S. Bharati, A. Chervenak, E. Deelman, C. Kesselman, M. Manohar, S. Patil, and L. Pearlman. A metadata catalog service for data intensive applications. In *Proceedings of Supercomputing*, 2003.

[14] K. Tai, S. Murdock, B. Wu, M. Ng, S. Johnston, H. Fanghor, S. J. Cox, P. Jeffreys, J. W. Essex, and M. S. P. Sansom. BioSimGrid: towards a worldwide repository for biomolecular simulations. *Org. Biomol. Chem.*, 2, 2004.

[15] D. Thain, S. Klous, J. Wozniak, P. Brenner, A. Striegel, and J. Izaguirre. Separating abstractions from resources in a tactical storage system. In *Proceedings of Supercomputing*, 2005.

[16] Brian S. White, Michael Walker, Marty Humphrey, and Andrew S. Grimshaw. LegionFS: A secure and scalable file system supporting cross-domain high-performance applications. In *Proceedings of Supercomputing*, 2001.

[17] J. M. Wozniak, P. Brenner, D. Thain, A. Striegel, and J. A. Izaguirre. Generosity and gluttony in GEMS: Grid-Enabled Molecular Simulation. In *Proceedings of the International Symposium on High Performance Distributed Computing*, 2005.

[18] J. M. Wozniak, P. Brenner, D. Thain, A. Striegel, and J. A. Izaguirre. Applying feedback control to a replica management system. In *Proceedings of the 38th Southeastern Symposium on System Theory*, 2006.

[2]GEMS is an open source project, available at `http://sourceforge.net/projects/gems-nd`.