NFSv4 Replication for Grid Storage Middleware

Jiaying Zhang
Center for Information Technology Integration
University of Michigan at Ann Arbor
jiayingz@umich.edu

Peter Honeyman
Center for Information Technology Integration
University of Michigan at Ann Arbor
honey@citi.umich.edu

ABSTRACT
Sharing data in scientific collaborations that involve many institutions around the world demands a large-scale storage system that is reliable and efficient, yet at the same time, convenient to use. This paper presents a replicated file system that supports mutable replication with strong consistency guarantees, high failure resiliency, and good scaling properties. The principal concern with such a system is usually its performance penalty. Using experiment evaluation, we show that in simulated wide area networks, the proposed replicated file system maintains a significant performance advantage over a single server system. At the same time, it provides comparable and often better performance than GridFTP, the de facto standard file transfer application that is often used to manually synchronize shared data.

Categories and Subject Descriptors
C2.4 [**Distributed Systems**]: Client/server; C4 [**Performance of Systems**]: Design studies.

General Terms
Design, Experimentation, Performance.

Keywords
Mutable replication, Consistency, Global name space, Distributed File System, NFSv4, Grid computing.

1. INTRODUCTION
The scientific community is seeing an increasing demand for global collaborations, spanning disciplines from high-energy physics, to climatology, to genomic [23-25]. Applications in these fields demand computational resources far beyond the scope of a single organization, and require access to massive amounts of storage. This imposes new challenges in data access, processing, and distribution.

Driven by the needs of scientific collaborations, the emerging Grid infrastructure [1, 2] aims to connect globally distributed resources to form a shared virtual computing and storage system, offering a model for solving large-scale computation problems. Sharing in Grid computing is not merely file exchange but also entails direct access to computers, software, data, and other resources, as is required by a range of collaborative scientific problem-solving patterns. To make such sharing simple and effective demands data access schemes that are scalable, reliable, and efficient.

MGC '06, November 27, 2006 Melbourne, Australia

The primary data access method used today in the Grid infrastructure is GridFTP [3]. Engineered with Grid applications in mind, GridFTP has many advantages: automatic negotiation of TCP options to fill the pipe, parallel data transfer, integrated Grid security, and resumption of partial transfers. In addition, because it runs as an application, GridFTP is easy to install and support across a broad range of platforms.

On the other hand, GridFTP does not offer sophisticated distributed data sharing, which impedes the convenient use of globally distributed resources for scientific studies. For example, in a common Grid use-case, a scientist wants to run a simulation on high performance computing systems and analyze results on a visualization system. With the Grid technologies available today, the scientist submits the job to a Grid scheduler, such as Condor-G [4]. The Grid scheduler determines where to run the job, prestages the input data to the running machines, monitors the progress of the running job, and when the job is complete, transfers the output data to the visualization system through GridFTP. The output data is reconstructed in the visualization site, and the results are returned to the scientist after reconstruction.

The scenario has the advantage of enabling the scientist access to more computing resources and speeding up his simulation. However, the whole process is performed in a batch mode. The scientist cannot view intermediate results before the entire scheduled job is complete. Scientific simulations are often iterative and interactive processes, so the need to wait for hours or days to examine experimental results is very inconvenient. To overcome this disadvantage, the Grid infrastructure requires more flexible data distribution and sharing in its middleware.

To facilitate Grid computing over wide area networks, we developed a replicated file system that provides users high performance data access with conventional file system semantics. The system supports a global name space and location independent naming, so applications on any client can access a file with a common name and without needing to know where the data physically resides. The system supports mutable (i.e., read/write) replication with explicit consistency guarantees, which lets users make data modification with ease, safety, and transparency. The system provides semantics compatible with POSIX, allowing easy deployment of unmodified scientific applications. We implemented our design by extending NFSv4, the emerging standard protocol for distributed filing [5]. In latter discussions, we refer to this system as **NFSv4.r**.

Using NFSv4.r, the scientist in the example described above can now monitor and control the progress of the simulation in real time. As illustrated in Figure 1, with the support of a global name space, the scientist can run programs on remote machines with the same pathname and without any reconfiguration. By using a replicated file system, the intermediate output of simulation is automatically distributed to the visualization center and the scientist's computer. The scientist can view intermediate results and immediately determine if parameters or algorithms need to be adjusted. If so, he can update them from his local computer and restart the simulation on the remote site, as simply as if he were running the experiment locally. Meanwhile, remote computation nodes can still access data from a nearby server.

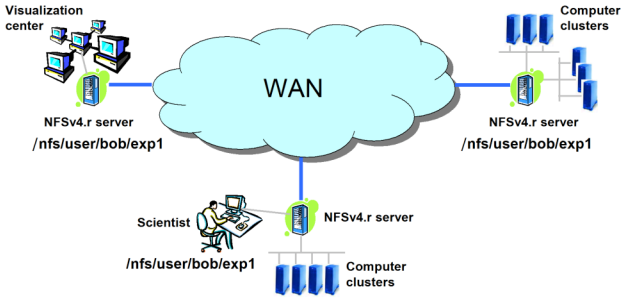

Figure 1. Grid use case scenario.

The remainder of the paper is organized as follows. We first give an overview of the system design in Section 2. The majority of the paper focuses on application evaluation: we examine how the system performs over wide area networks and the benefits it provides to Grid applications. Section 3 presents the experimental results we collected as well as the data analysis. Following that, we review related work in Section 4, and conclude in Section 5.

2. DESIGN

This section describes NFSv4.r, a replicated file system designed to facilitate the data access and management of large-scale scientific applications. Section 2.1 presents the design of a naming scheme that supports a global name space. Then, Section 2.2 describes a replication protocol that provides mutable replication support with strong consistency guarantees.

2.1 Global Name Space

The NFSv4 protocol includes features to support read-only replication using a special file attribute `FS_LOCATIONS`. By the NFSv4 specification, a client's first access to a replicated file system yields the `FS_LOCATIONS` attribute that lists alternative locations for the file system. Complying with the NFSv4 protocol standard, we also use the `FS_LOCATIONS` attribute to communicate replica location information between servers and clients. However, the namespace of NFSv4.r includes extensions to support a single global name space.

By convention, we define a special directory, `/nfs`, as the global root of all NFSv4 file systems. Entries under `/nfs` are mounted on demand. The first time a user accesses any NFSv4 file system, the referenced name is forwarded to a daemon that queries DNS to map the given name to one or more file server locations, selects a file server, and mounts it at the point of reference. The format of reference names under `/nfs` follows Domain Name System [6] conventions. We use the SRV Resource Record [7] to store server location information.

2.2 Consistent Mutable Replication

To meet availability, performance, and scalability requirements, distributed services naturally turn to replication, and file service is no exception. While the concept of file system replication is not new, existing solutions either forsake read/write replication totally [11–13] or weaken consistency guarantees [14, 15]. These compromises fail to satisfy the requirements for global scientific collaborations.

Returning to the example described in Section 1, experiment analysis is often an iterative, collaborative process. The stepwise refinement of analysis algorithms employs multiple clusters to reduce development time. Although the workload during this process is often dominated by read, it also demands the underlying system to support write operations. Furthermore, strong consistency guarantees are often assumed. For example, an executable binary may incorporate user code that is finished only seconds before the submission of the command that requires using the code. To guarantee correctness, the underlying system needs to guarantee that the modified data is successfully transferred to the remote machine where the code is running.

The conventional NFSv4 consistency model, the so-called "close-to-open semantics" [18], guarantees that an application opening a file sees the data written by the last application that writes and closes the file. The strategy has proved to provide sufficient consistency for most applications and users [19].

We also consider supporting close-to-open semantics to be important in Grid data access, as the above example illustrates. To provide such a guarantee, our replication extension to NFSv4 coordinates concurrent writes by selecting a primary server [16]. Unlike the conventional primary copy approach, we do not assign primary servers in advance, but allow any client to choose any relevant server. With no writers, our system has the performance profile of other systems that support read-only replication (e.g., AFS): use a nearby server, support transparent client rollover on server failure, etc. Unlike read-only systems, however, we also support concurrent access with writers. Performance penalties are slight, and are induced only when writers are active.

The system works as follows. When a client opens a file for writing, the chosen server temporarily becomes the primary server for that file. All other replication servers are instructed to forward client requests for that file to the primary server. The primary server distributes updates to other servers during file modification. When the last writer is done, the primary server notifies the other replication servers that it is no longer the primary server for the file. A server becomes the primary server only after it collects acknowledgements from a majority of replication servers. To further guarantee close-to-open semantics, a primary server must ensure that every replication server has acknowledged its role when a written file is closed, so that the subsequent reads on any server can reflect the modified data before close.

For directory updates, the similar procedure is followed except for the handling of concurrent writes. Because a directory update completes quickly, a replication server simply waits for the primary server to relinquish its role if concurrent writes occur on a directory currently being modified.

The election of a primary server could be delayed waiting for acknowledgments from slow or distant replication servers. To amortize the performance penalty, we allow a primary server to

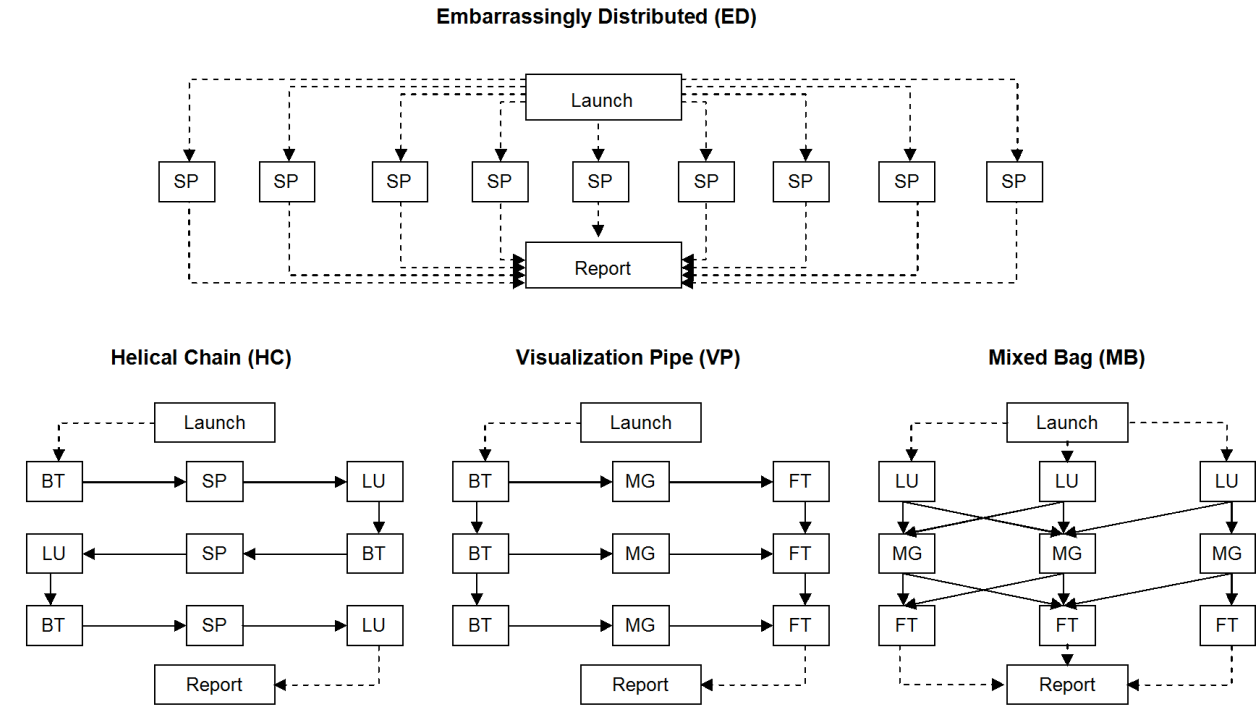

Figure 2. Data flow graphs of the NAS Grid Benchmarks.

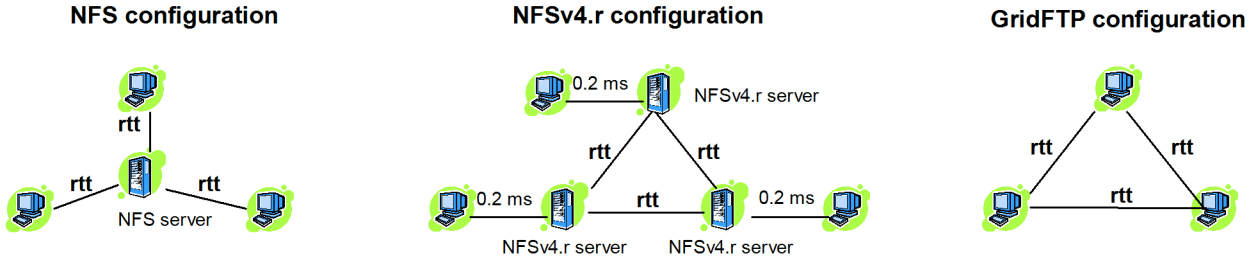

Figure 3. NGB evaluation experiment setup.

Table 1. Amount of data exchanged between NGB tasks

Class	Helical Chain			Visualization Pipe				Mixed Bag	
	BT→SP	SP→LU	LU→BT	BT→MG	MG→FT	BT→BT	FT→FT	LU→MG	MG→FT
S	169K	169K	169K	34K	641k	169K	5.1M	34K	641K
W	1.4M	4.5M	3.5M	271K	41M	1.4M	11M	702K	41M
A	26M	26M	26M	5.1M	321M	26M	161M	5.1M	321M

assert control at various granularities: a file, a directory and its constituent entries, or the entire subtree rooted at a directory. The preliminary results we collect show that with this strategy, the overhead of replication control is negligible even for update-intensive applications. For more details on the above algorithm, we refer readers to a related paper [17].

3. EVALUATION

In this section, we explore the performance of NFSv4.r with the NAS Grid Benchmarks over simulated wide-area networks. We measured all the experiments presented in this paper with a prototype implemented in Linux 2.6.16 kernel. Servers and clients all run on dual 2.8GHz Intel Pentium4 processors with 1024 KB L2 cache, 1 GB memory, and Intel 82547GI Gigabit Ethernet card onboard. The number of bytes NFS uses for reading (rsize) and writing files (wsize) is set to 32768 bytes. We use Netem [20] to simulate the network latencies. To focus on evaluating the performance impact caused by WAN delays, we do not simulate packet loss in our measurements, and enable the `async` option (asynchronously write data to disk) on the NFS servers. All numbers presented are mean values from five trials of each experiment; standard deviations (not shown) are within five percent of the mean values.

Table 2. Times of executing NGB on a single computing node with a local ext3 file system

Class	S				W				A			
Benchmark	ED	HC	VP	MB	ED	HC	VP	MB	ED	HC	VP	MB
Time (s)	2	1	9	6	217	31	83	101	1380	223	930	870

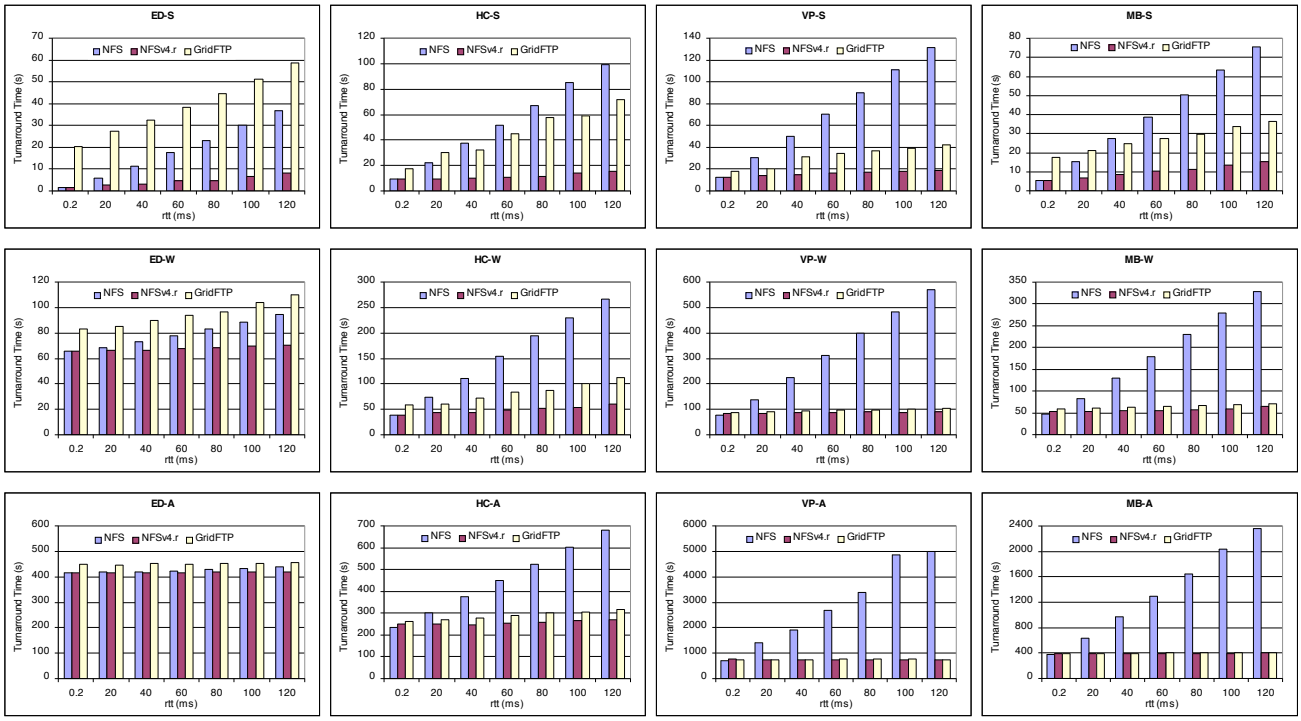

Figure 4. Turnaround times (seconds) of NGB on NFS, NFSv4.r, and GridFTP.

3.1 NAS Grid Benchmarks

The NAS Grid Benchmarks (NGB), released by NASA, provide an evaluation tool for Grid computing. The benchmark suite evolves from the NAS Parallel Benchmarks (NPB), a toolkit designed and widely used for benchmarking on high-performance computing. An instance of NGB comprises a collection of slightly modified NPB problems, each of which is specified by class (mesh size, number of iterations), source(s) of input data, and consumer(s) of solution values. The current NGB consists of four problems: Embarrassingly Distributed (ED), Helical Chain (HC), Visualization Pipe (VP), and Mixed Bag (MB).

ED, HC, VP, and MB highlight different aspects of computational Grid. ED represents parameter studying applications that constitute multiple independent runs of the same program, but with different input parameters. It requires virtually no communication, and all the tasks in it execute independently. HC represents long chains of repeating processes; tasks in HC execute sequentially. VP simulates logically pipelined processes, like those encountered when visualizing flow solutions as the simulation progresses. The three tasks included in VP fulfill the role of flow solver, post processor, and visualization, respectively. MB is similar to VP, but introduces asymmetry. Different amounts of data are transferred between different tasks, and some tasks require more work than others do. Figure 2 illustrates the Data Flow Graph (DFG) for each of these benchmarks. The nodes in the graph, indicated by the rectangular boxes, represent computational tasks. Dashed arrows indicate control flow between the tasks, and solid arrows indicate data as well as control flow. Launch and Report do little work; the former initiates execution of tasks, and the later collects and verifies computation results.

The NGB instances can run for different problem sizes (denoted *Classes*). For the evaluation results presented in this paper, we use the three smallest Classes: S, W, and A. Table 1 summarizes the amount of data communicated between tasks for these Classes.

A fundamental goal of Grid computing is to harness globally distributed resources for solving large-scale computation problems. To explore the practicality and benefit of using NFS replication to facilitate Grid computing, we compare the performance of running NGB under three configurations, referred as NFS, NFSv4.r, and GridFTP. While our experiments intend to emulate the behavior of three computing *clusters*, with the RTT between each pair of clusters increased from 0.2ms (the RTT measured in our local area network) to 120ms, we actually use three computing *nodes*. This suffices to draw out the performance implications of latency among distant computing sites.

In the NFS configuration, all three computing nodes connect to a single NFS server. In the NFSv4.r configuration, we replace the single NFS server with three replicated NFS servers, with each computing node connected to a nearby server. In the GridFTP configuration, we use GridFTP to transfer data between computing nodes. The GridFTP software we use is `globus-url-`

`copy` from Globus-4.0.2 toolkit. In our experiments, we start eight parallel data connections in each GridFTP transfer, which provides the best-measured performance for GridFTP. We note that the NFSv4.r prototype includes mechanisms of using parallel data connections among replicated NFS servers as well. For the experiments presented in this paper, the performance improvements of using multiple data connections are relatively small. Thus, we report the results measured with a single server-to-server data connection only. Figure 3 illustrates the described experiment setup.

For the GridFTP configuration, we run the NGB tasks using the Korn shell Globus implementation from the NGB3.1 package. In this implementation, a Korn shell script launches the NGB tasks in round robin on the specified computing nodes. Tasks are started through the `globusrun` command with the batch flag set. After a task completes, the output data is transferred to the computing node(s) on which the running tasks require the data as input. A semaphore file is then created to signal task completion. The computing nodes poll their local file systems for the existence of the semaphore files to monitor the status of the required input files. After all tasks start, the launch script periodically queries their completion using `globus-job-status` command.

For the NFS and NFSv4.r setups, we use the modified Korn shell scripts that start NGB tasks in round robin on the specified computing nodes with the `ssh` command. The computing nodes and the launch script poll for the status of the required input data and tasks through semaphore files, as described above.

Figure 4 presents the results of executing NGB on NFS, NFSv4.r, and GridFTP, as the RTT among the three computing nodes increases from 0.2 ms to 120 ms. The data presented is the "measured turnaround" time, i.e., the time between starting a job and obtaining the result. With GridFTP, turnaround time does not include deployment and cleanup of executables on Grid machines. The time taken in these two stages ranges from 10 seconds to 40 seconds, as the RTT increases from 0.2 ms to 120 ms. We note that in Grid computing, deployment and cleanup can sometimes take significant time with large size of executables and input data [22]. Furthermore, in some cases, it is hard for users to determine which files to stage [21]. With NFS and NFSv4.r, on the other hand, there is no extra deployment and cleanup time, because computing nodes access data directly from file servers.

The results in Figure 4 show that the performance with a single NFS server decreases dramatically as the RTT between the server and the computing nodes increases. Except for the ED problem — whose tasks run independently — on large data sets, the experiments take a very long time to execute when the RTT increases to 120 ms. In fact, the times are even longer than those we measured when running the problems on a single computing node without parallel computing. (Table 2 presents the times of executing NGB on a single computing node with a local ext3 file system.) This observation implies that in most cases, it is impractical to run applications on widely distributed clients connected to a single NFS server, even for compute intensive applications.

On the other hand, with NFSv4.r and GridFTP on large class sizes, run times are not adversely affected by increasing RTT. When the class size is small (e.g., the results of Class S), NFSv4.r outperforms GridFTP, because the latter requires extra time to deploy dynamically created scripts and has extra Globus-layer overhead. The experiments demonstrate that it is possible to provide superior file system semantics and easy programmability to WAN-based Grid applications without sacrificing performance.

4. RELATED WORK

For its superior read performance, read-only replication has been favored in the current Grid experimental platform [1]. With read-only replication, once a file is declared as shared by its creator, it cannot be modified. An immutable file has two important properties: its name may not be reused and its contents may not be altered. Yet, read-only replication has several deficiencies. First, it fails to support complex sharing behavior, e.g., concurrent writes. Second, to guarantee uniqueness of file names, file creation and retrieval require a special API, which hinders using the software developed in the traditional computing environment for global collaborations.

Various middleware systems have been developed to facilitate data access on the Grid. **Storage Resource Broker (SRB)** [8] provides a metadata catalog service to allow location-transparent access for heterogeneous data sets. **NeST** [9], a user-level local storage system whose goal is to bring appliance technology to the Grid, provides best-effort storage space guarantees, mechanisms for resource and data discovery, user authentication, quality of service, and multiple transport protocol support. The **Chimera** system [10] provides a virtual data catalog that can be used by applications to describe a set of programs, and then track all the data files produced by executing them. The work is motivated by observing that scientific data is often derived from other data by the application of computational procedure, which implies the need for a flexible data sharing and access system.

A common missing feature among these middleware approaches is the absence of fine-grained data sharing semantics. Furthermore, most of these systems provide extended features by defining their own API, so an application has to be re-linked with their libraries in order to use them.

GPFS-WAN [26, 27] combines widely distributed cluster file systems into a common global file system. In operation on the TeraGrid [28], the system demonstrates the promise of using UNIX I/O operations and Grid security for wide-area data access. GPFS-WAN focuses on delivering high throughput for massive data objects, but does not address the cost of individual I/O and metadata operations.

Emerging large-scale scientific collaborations have stimulated the growing research in scientific workload studies. Their findings, described next, provide valuable insights on the design of NFSv4.r.

Thain et al. study the characteristics of six scientific applications [21] whose workloads consist of several pipelines. The studied workloads demonstrate three common behaviors: First, small initial inputs are usually expanded by early stages into large intermediate results, which are often reduced by later stages to small results. Second, although users tend to identify large data collections needed by an application, in a given execution, applications usually selects a small working set. Third, the examined workloads all exhibit large degrees of data sharing.

Holtman et al. investigate the data processing requirements of the CMS experiment, one of the two large particle physics projects being developed at CERN, and the expected workload characteristics during the operation of the Large Hadron Collider [22]. They point out that the CMS workloads will be dominated by reading,

and the stepwise refinement of algorithms will lead to a workload where a series of jobs is run over the same input data, with each job containing the refined code or parameter. Sometimes, CMS applications need to access data randomly from data sets that are too large to stage to every machine in a site. Such use cases require accessing files on a large file system local to the site. Furthermore, CMS expects to access files through regular POSIX I/O calls without re-linking with special libraries.

5. CONCLUSION

The conventional wisdom holds that middleware to support consistent mutable replication in large-scale distributed storage systems is too expensive to consider. Our experiments prove otherwise: in reality, it is both feasible and practical. In this paper, we have focused on the needs of global collaborations: a global name space, which facilitates data sharing, distribution, and management; and a replication control protocol that supports mutable replication with strong consistency guarantees. Experimental evaluation shows that NFSv4.r provides a promising way for accessing and sharing data in Grid computing, delivering superior performance and rigorous adherence to the conventional file system semantics.

6. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their suggestions and gratefully acknowledge the support of NSF Middleware Initiative Grant No. SCI-0438298 and Network Appliance, Inc.

7. REFERENCES

[1] Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., and Tuecke, S. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets. *J. Network and Computer Applications* (2000).

[2] Foster, I. and Kesselman, C. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann (1998).

[3] Allcock, W., Bresnahan, J., Kettimuthu, R., and Link, M. The Globus Striped GridFTP Framework and Server.. *Supercomputing* (2005).

[4] Frey, J., Tannenbaum, T., Livny, M., Foster, I., and Tuecke, S. Condor-G: A computation management agent for multi-institutional grids. *HPDC* (2001).

[5] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and Noveck, D. Network File System (NFS) version 4 Protocol. RFC 3530 (2003).

[6] Mockapetris, P. Domain Names – Concepts and Facilities. RFC 1034 (1987).

[7] Mockapetris, P. Domain Names – Implementation and Specification. RFC 1035 (1987).

[8] Baru, C., Moore, R., Rajasekar, A., and Wan, M. The SDSC Storage Resource Broker. *CASCON Conf.* (1998).

[9] Bent, J., Venkataramani, V., LeRoy, N., Roy, A., Stanley, J., Arpaci-Dusseau, A. C., Arpaci-Dusseau, R. H., and Livny, M. Flexibility, Manageability, and Performance in a Grid Storage Appliance. *HPDC* (2002).

[10] Foster, I., Voeckler, J., Wilde, M., and Zhao, Y. Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation. *Scientific and Statistical Database Management Conf.* (2002).

[11] Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., and Tuecke, S. Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. *IEEE Mass Storage* (2001).

[12] Satyanarayanan, M., Howard, J. H., Nichols, D. A., Sidebotham, R. N., Spector, A. Z., and West, M. J. The ITC distributed file system: principles and design. *Op. Sys. Rev.* (1985).

[13] White, B. S., Walker, M., Humphrey, M., and Grimshaw, A. S. Legionfs: a secure and scalable file system supporting crossdomain high-performance applications. *Supercomputing* (2001).

[14] Kumar, P. and Satyanarayanan, M. Supporting application specific resolution in an optimistically replicated file system. In *Workshop on Workstation Op. Sys.* (1993).

[15] Satyanarayanan, M., Kistler, J.J., Kumar, P., Okasaki, M.E., Siegel, W.H., and Steere, D.C. Coda: A highly available file system for a distributed workstation environment. *IEEE Trans. on Computers* (1990).

[16] Zhang, J. and Honeyman, P. Naming, Migration, and Replication for NFSv4. *SANE* (2006).

[17] Zhang, J. and Honeyman, P. *Hierarchical Replication Control.* Tech. Rep. CITI-06-03, Center for Information Technology Integration, Ann Arbor (2006).

[18] Pawlowski, B., Shepler, S., Beame, C., Callaghan, B., Eisler, M., Noveck, D., Robinson, D., and Thurlow, R. The NFS version 4 protocol. *SANE* (2000).

[19] Pawlowski, B. Juszczak, C., Staubach, P., Smith, C., Lebel, D., Hitz, D. NFS Version 3 Design and Implementation. *USENIX Summer Tech. Conf.* (1994).

[20] Hemminger S. Netem – Emulating Real Networks in the Lab. *LCA2005* (2005).

[21] Thain, D., Bent, J., Arpaci-Dusseau, A. C., Arpaci-Dusseau, R. H., and Livny, M. Pipeline and batch sharing in grid workloads. *HPDC* (2003).

[22] Holtman, H. *CMS data grid system overview and requirements*. The Compact Muon Solenoid (CMS) Experiment Note 2001/037, CERN, Switzerland (2001).

[23] The Large Hadron Collider (LHC). http://lhc.web.cern.ch/lhc/.

[24] The Educational Global Climate Modeling (EdGCM). http://edgcm.columbia.edu/.

[25] The Basic Local Alignment Search Tool (BLAST). http://www.ncbi.nlm.nih.gov/BLAST/.

[26] Andrews, P., Jordan, C., and Pfeiffer, W. Marching Towards Nirvana: Configurations for Very High Performance Parallel File Systems. *HyperIO workshop* (2006).

[27] Andrews , P., Jordan C., and Lederer, H. Design, Implementation, and Production Experiences of a Global Storage Grid. *IEEE Mass Storage* (2006).

[28] TeraGrid. http://www.teragrid.org/