

On Being Close to the Data :

Executing Code in a Replica Management System

J. M. Wozniak, P. Brenner, D. Thain, A. Striegel, J. A. Izaguirre

Dept. of Computer Science & Engineering

University of Notre Dame

Notre Dame, IN 46556 USA

{ *jwozniak, pbrenne1, dthain, striegel, izaguirr* } @nd.edu

Abstract

The goal of scientific grid computing is to enable researchers to attack problems that were previously impossible. Cooperative storage systems have been assembled to provide users with access to surprisingly large amounts of useful space, but locating, accessing, and efficiently employing such distributed data sets in scientific simulation or analysis is another monumental problem. In this paper, we describe simple techniques compatible with existing software that may be used on clusters or grids to access catalogued, replicated data sets: software tools that allow user code to use a replica system as a scientific file system. The importance of operating in-place, close to the data, is demonstrated by the performance improvements gained by using data locations as a guide when scheduling computation. A new replica management system, called GEMS¹, is proposed to improve the research capabilities of two important computing infrastructures: university networks (groups of clusters) and volunteer computer resources (Internet computing).

I. INTRODUCTION

Modern shared computational clusters and grids offer users a great deal of capacity for code execution. When multiple independent jobs are executed in such systems, the total system may obtain a performance speedup that is linear in the number of compute nodes *after input data has been distributed to the nodes and before output data is recovered*. The data movement mechanism - the manner in which data is moved among file servers and compute resources - is

¹Project supported by: NSF DBI-04500667

limited by the network, and individual file servers are limited by performance specifications and network connectivity. Even on small compute clusters, batches of jobs that operate on sizeable files will quickly find that a centralized storage device will become the bottleneck.

Clusters often are built on a simple, time-tested model: a head node for job submission and an array of compute nodes, all connected over a high-speed, reliable link to a dedicated storage service such as an NFS [17] installation, or others. Jobs with heavy data loads, however, can quickly cause congestion in such a setting. Larger scientific tasks that require large batches of thousands of jobs or more will easily overwhelm the centralized service, resulting in poor performance for all jobs. Users cannot combine such resources into scientific grids because the single data source is not scalable to large numbers of tasks, is not searchable, and does not support the sharing of data across organizational domains.

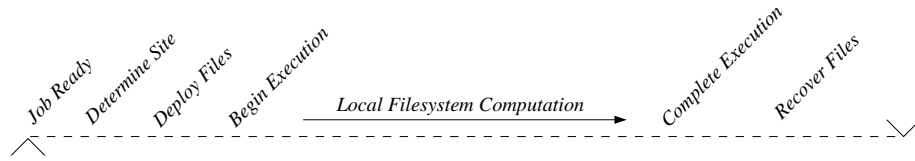
A potential solution to this problem is to utilize multiple data access points at different locations in the network. Multiple file servers can be used to provide file access to the compute system. If multiple data routes exist between file servers and compute servers, the capacity to move data can scale with the compute capacity.

Developing such a distributed file service involves dealing with several challenges, including logical name resolution, efficient matchmaking, and data movement. First, in order to gain access to data, jobs must be able to resolve given logical file names into physical file locations, i.e., where the actual file may be obtained. Jobs must also be able to store new data in the system in a permanent way - not just in temporary space on the compute node - somewhere in the distributed system. Users often may require the use of a predetermined storage location, which means that a job must be able to obtain a new logical name and a corresponding physical location that is based on user request.

Second, a mechanism for *data movement* must be developed to support such a system, which involves gaining access to the data associated with a physical location. Traditionally, centralized data services move whole files to and from computation sites or respond to smaller file operations over the network. In a distributed data service, jobs must also navigate the access control complexities of the various file servers, and be able to access data from one or more of the file servers in accordance with the system policy or computation method in effect.

Additionally, jobs gain a performance improvement from being *close to the data*. A low latency, high bandwidth connection between a compute node and a file server will have clear

Traditional file staging model:



Replica-aware computation model:

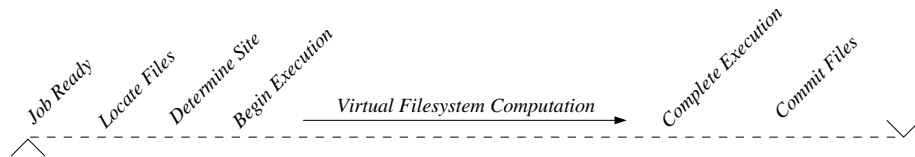


Fig. 1. In the traditional, file staging model, the site is determined before data files are considered, because all data is assumed to come from a centralized source. In the distributed, replica-based computation model, the replica locations *are* considered, and a virtual filesystem is used to connect jobs to data services, because the files are not explicitly copied over the network.

benefits if the job is constrained by data operations. If the compute network and the storage network occupy the same physical machines, a significant performance improvement can be obtained by matching jobs to machines that have the required data sets.

A replica management system called GEMS [24], [25] has been developed to meet the storage needs of users running batches of data-intensive scientific simulations. This alternative approach *recognizes the replica creation process as an asynchronous data pre-staging process*, in which data sets are moved to potential compute sites in advance. In our model, the exploitation of compute and storage site collocation allows reads and writes directly to the local file server. Upon job completion, output data is committed to the system, replicated by the system in an asynchronous manner, and actively maintained by auditing services. This overlay system allows users to employ a vast network of independently operated storage servers *reliably* and *without data staging*.

In the next section, the properties of a distributed file service are analyzed in more detail. In Section III, a replica management system is used to construct a system for distributed computation. Cases are described in which the new system would be beneficial in Section IV-A, and performance comparisons using the implementation are provided for such cases in Section IV-B. Related work is considered in Section V, and conclusions may be found in Section VI.

II. MOTIVATION: PROPERTIES OF DATA MOVEMENT

In this section, two idealized computation models are compared: those based on single file server architectures and those based on distributed replica servers. To demonstrate the inability of single file server architectures to support large numbers of data-intensive jobs, microbenchmarks are performed to quantify the scaling properties of a single file service.

A. Computation Models

Figure 1 offers a comparison of two computation models: a traditional file staging model and a distributed file service model. In the file staging model, data sets are distributed to the compute node just prior to job start. The data movement time must be added to the turnaround time for the job, and during execution file access is available on a local temporary copy. A distributed model is then shown, in which replicas of a data set are assumed to exist on a variety of nodes. Jobs in the replica model access data through a virtual filesystem that translates standard file access operations into operations to a local or network-accessible file.

In the file staging model, the “Determine Site” step implies only that the chosen site is compatible with the given program. This step typically does not address the need for nearby access to data, since the traditional model assumes a single file server will be used.

In the distributed model, job scheduling and file locations are considered immediately. The required file access must be analyzed and jobs must be colocated with data resources, if possible. Thus, *multiple* sites are determined, as multiple sites may be chosen for input, output, and computation. The resulting aggregate computation for a batch of jobs may be thought of as a “switchboard”, with bipartite sets of compute servers and file servers connected by virtual filesystem operations. Upon job completion, new files, which are created in-place on replica servers, are not explicitly copied to a permanent location, but are committed to the replica system, enabling asynchronous replication in accordance with replication policy.

The replica-based model increases the scalability of the storage system by:

- 1) decreasing the load on any given file service;
- 2) allowing flexibility in scheduled computation and file server access;
- 3) demonstrating graceful degradation in the presence of file server or compute server failure.

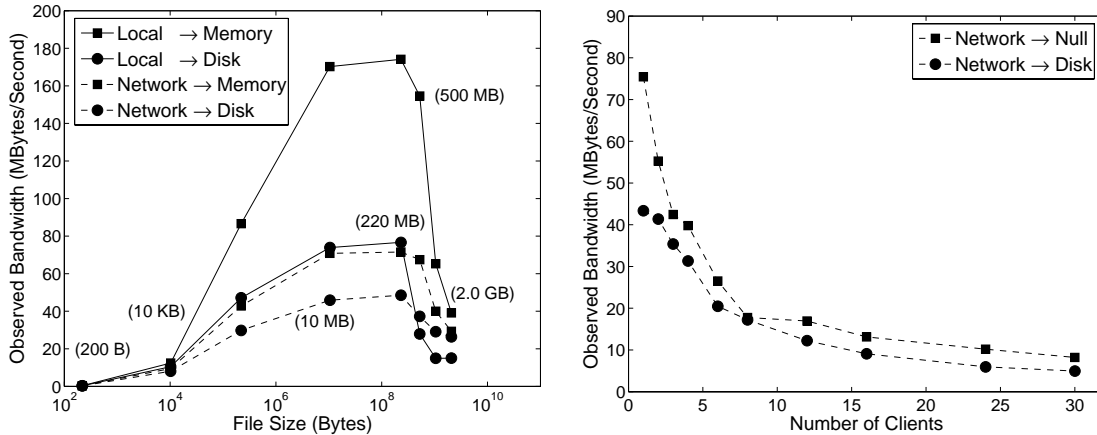


Fig. 2. Microbenchmarks: Observed bandwidth for Chirp clients on a LAN.

B. Data Access Rates

In the previous subsection we claim that matching jobs to file locations will produce a performance benefit, but in practice, on high speed local networks, such results must be verified experimentally. In the following presentation, we report the observed bandwidth of file access modes relevant to our discussion for files of varying sizes. We demonstrate four modes: Local Server to Memory, Local Server to Local Disk, Network Server to Memory, and Network Server to Local Disk. The first two modes represent file access rates for a job that is not collocated with its data locations, and the second two modes represent file access rates for a job that is collocated with a file server that has the necessary data. When “Disk” is the target, that indicates that the files were actually copied to a client disk, and the “Memory” target indicates that the data was delivered to the client and discarded without making a disk copy.

Results were compiled and averaged over multiple test from a sampling of 32 otherwise idle machines, each with Pentium III 1.4 GHz processors and 1 GB of physical memory, running Linux 2.6.9 and the Chirp [22] personal file server, all connected by a 1000Mb switch. Multiple copies per pair of machines were performed to indicate the ability of servers to keep files in memory. The results are shown in Figure 2 in the left frame.

The observed bandwidth results demonstrate that read access to a local replica is clearly better than access to a remote replica. Results in this plot are particularly clear for moderately large files of size ranging from 2 MB to 100 MB, in which the performance is constrained only by the

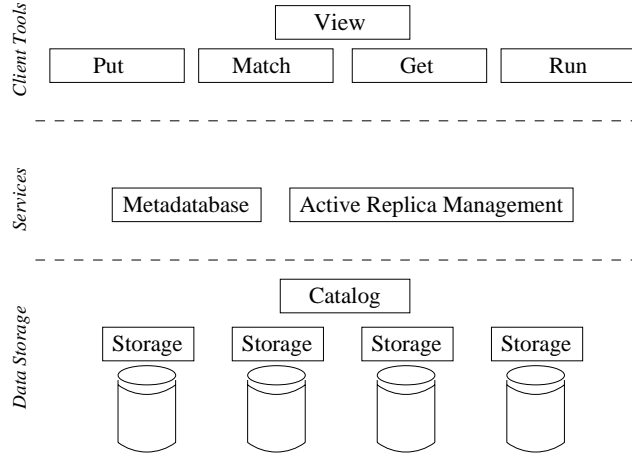


Fig. 3. Replica Management System Framework.

performance of the local disk. For very large files over 500 MB and up to 2 GB, the performance is constrained by the ability of the server to keep the whole file in memory between transfers, reducing performance. Making copies of very large local files is a very slow process because the whole file does not fit in memory, requiring multiple reads and writes with intervening seeks: this behavior may be observed on the right hand side of the Local to Disk line.

To illustrate the performance penalties involved in serving multiple jobs from the same file server, the effect of multiple client machines accessing a single large file from a file server was plotted in the right frame. In this experiment, multiple machines on the same local network simultaneously begin accessing a 500MB file from a server chosen from the network. The “Memory” and “Disk” targets have the same meaning as in the previous experiment. The results show that performance quickly degrades as the number of concurrent client connections increases, even though the data being served is fixed.

III. REPLICA-AWARE COMPUTATION

Having described the performance issues involved with centralized storage services, we now offer a framework in which the distributed storage problems described previously may be approached. Our replica management system consists of a three-level architecture: a toolset of client programs, a searchable metadatabase and storage management system, and the distributed storage servers, as shown in Figure 3.

A. Storage Benefits of Replica Management Systems

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 [24].

B. Computation in a Replica System

An adapter has been previously developed [22] to treat the whole distributed file system as a single filesystem. By trapping file operations from a running process, the adapter may allow the process to access files from remote servers in a transparent way. For example, a user could execute the script shown in Figure 4.

```
> IN=/remote/host1.nd.edu
> OUT=/remote/host2.nd.edu
> adapter simulator $IN/input.config $OUT/output.data
```

Fig. 4. Example Simulation Script using a Virtual Filesystem

This adapter would begin executing the `simulator` subprocess, trapping its file operation. When the `simulator` performs a file operation, such as an input read or an output write, the operation is captured by the adapter and forwarded over the network to the appropriate service.

Such syntax provides an elegant scripting tool and 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 tools to locate data. A new external system is needed to locate input sites, find sites to safely store outputs, and obtain a compute site that is not already occupied. To obtain good performance, the three sites must be collocated.

Clients of the replica management system are responsible for mapping file locations in the searchable, database-like namespace to the logical namespace of the virtual filesystem. In this work, we call the entries in replica namespace the *abstract* filenames, as opposed to the *virtual*

filenames compatible with the adapter. *Name resolution* maps abstract names in a `/<Data set label>/path/file` format to virtual filenames in a `/protocol/host/path/file` format. The replica management system provides the additional ability to obtain data set labels by searching over the metadata, providing the ability to perform computation in a completely application-oriented way, as shown in Figure 5.

```
> KEY=$( Match reagents=acidbase )
> Run --input HCl /$KEY/hcl --input NaOH /$KEY/naoh
    --output NaCl salt --output HOH water
    reagents=saltwater
    --exec transmute HCl NaOH to NaCl HOH
```

Fig. 5. Example Simulation Script using Abstract Data Locations

The first line of the script uses the `Match` client to locate the data set label required to obtain the necessary input files for the `transmute` task. This key may then be used to specify abstract file locations to the `Run` client. The second line invokes the `Run` client to resolve the abstract file names to virtual file names compatible with the adapter, and then to submit the user task, `transmute`, to an available compute system, using the adapter to actually perform the file operations. The result of this script is a new entry in the replica system, containing two files, `salt` and `water`, which may be located by using the key-value pair `reagents=saltwater`.

Clearly, more complex tasks would involve very lengthy command lines. Since existing job schedulers already require users to explicitly define input and output files, simple extensions to the syntax of these job scheduler scripts may be preprocessed by the `Run` client to provide a more familiar syntax for job submitters.

C. Job Submission Methods for Replica Access

As seen above, a central service maintains a database of replica locations. This service may be queried in three ways for the purposes of this paper: to map metadata tags to data set labels, to map data set labels to a set of file names, or to map a dataset identity and file name to a storage site. Once the site has been obtained, the data source may now be accessed in a location independent way as described above.

While a replica management system does not spawn computation itself, we demonstrate a tool to interface with existing computation systems to access and create data in a compatible

way. In this section, we outline four *computation modes*, that is, ways to effectively perform computation utilizing replica access, including:

- 1) Local Computation on Remote Data; which allows the a local workstation to access remote data sources and create new data in the replica system over a virtual filesystem. This is a simple prototyping technique that does not rely upon a batch scheduler or job submission system.
- 2) Scheduled Computation on Remote Data; which interfaces with an existing scheduler to create jobs that access data over a virtual filesystem. It utilizes the scheduler's ability to prefer target computation sites, improving the performance of the system by targeting sites that have the required data sets in their file server.
- 3) Remote Computation on Remote Data; which utilizes the ability to directly send jobs to remote systems for processing. This method relies on special functionality available in the file server, but could be extended to other systems through more commonly available tools.
- 4) Multiple Name Resolution; which is a more complex scheduled method, guiding the matchmaking process with respect to data locations. Additional functionality is packaged with the job to enable more efficient and fault-tolerant use of remote resources upon job arrival.

1) Local Computation on Remote Data (LCRD): In many common cases, the user simply desires to run a single job on the local workstation. The LCRD model enables users to start new jobs that require data access to the replica management system. This mode is based upon a typical command consisting of an operation on abstract dataset identities, comprising the input and output locations. These abstract arguments, which do not specify the actual data location, are translated by the replica system into the physical file locations required by the virtual filesystem adapter, in a manner analogous to shell parameter parsing and expansion. Thus a task that requires access to remote, abstracted data sets may be translated into a local task operating on data that is virtually local. An example execution of this method is shown in Figure 6, in the LCRD frame.

Optimizing this operation is quite simple. First, the output data location is determined. The preferred output location is a server on the local host, but if this is not available or not allowed by the relevant access control policy, a server that is not currently busy will be selected. In the

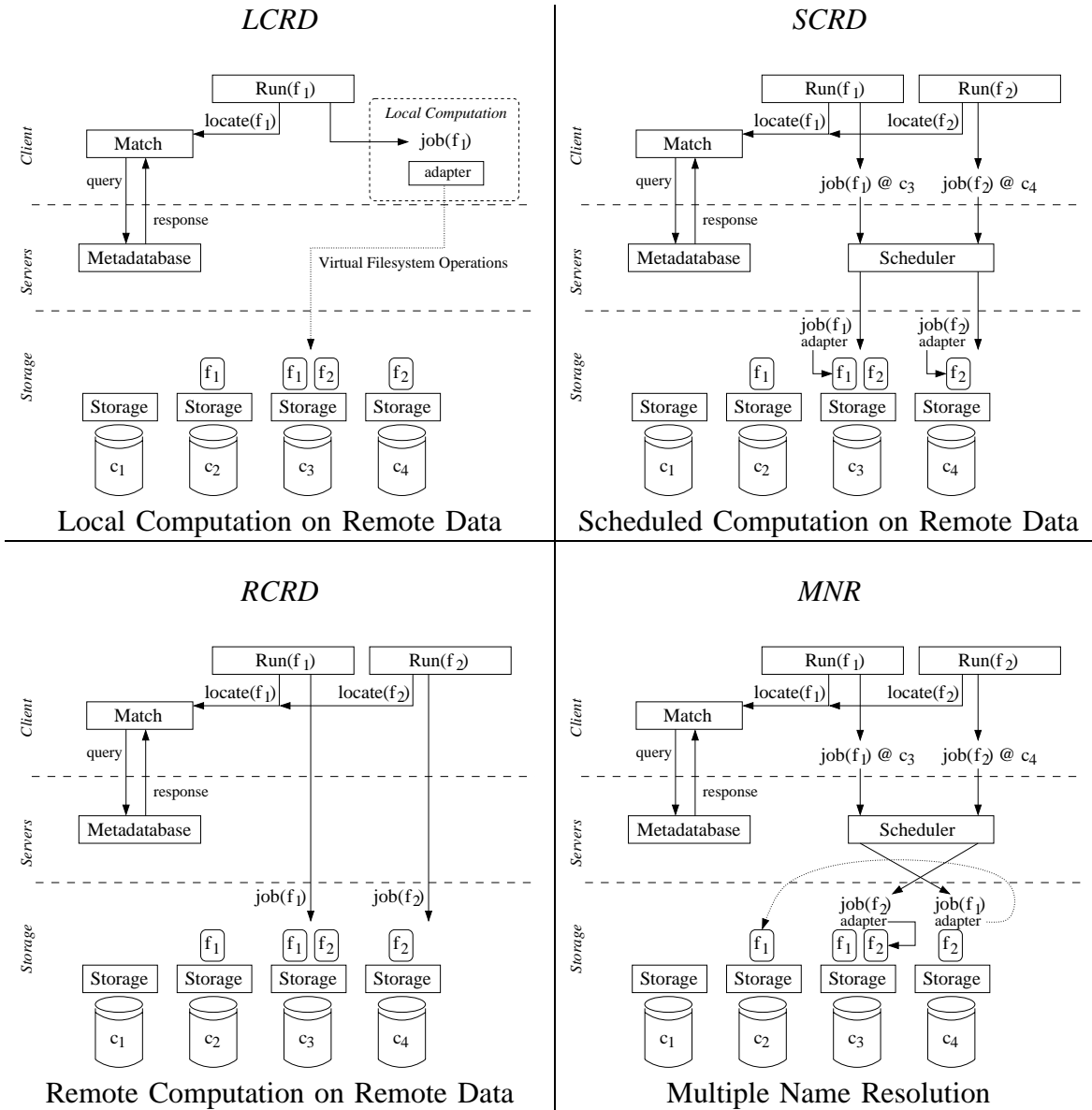


Fig. 6. Computation in a Replica Management System

worst case, a remote, busy machine will be selected. If any required input files from the system have replicas on a local server, these hosts are selected as data sources. Otherwise, a remote, idle service will be selected to provide the file, and if this is not possible the worst case behavior of a busy remote server will be selected.

2) *Scheduled Computation on Remote Data (SCRD)* : Convenient tools on the client end allow clients to employ replica-based computations in existing scheduler submission scripts and

workflows. We augment the specification for scheduled jobs by allowing users to specify input and output locations that reside in the replica management system, and use these locations in the commands and arguments. The Run client translates this augmented submission script into a submission script compatible with the replica system by making necessary substitutions: resolving the abstract data locations into virtual filesystem locations, and ensuring that the resulting job does in fact run atop the adapter. The data sets required by the resulting job are then location-independent because of the adapter, i.e., no data staging is necessary.

As shown in the SCRD frame of Figure 6, jobs are sent to the scheduler with requested destination hosts, illustrated by the “@” markup. The scheduler honors the request and submits the job to the appropriate compute site. The job then proceeds, operating on local data through the adapter.

3) *Remote Computation on Remote Data (RCRD)*: The file server used in this work allows the user to execute jobs inside an isolated “identity box”. The Run client makes use of this technique by sending jobs directly to a file server for computation. The method is similar to the scheduled SCRD method, however, we are missing two important concepts: external centralized matchmaking and real scheduling. To avoid overloading remote hosts, the management system must infer its load on the remote server by determining if the server is currently reserved by another client request. The server may also be excluded if it is being used for file transfer by the replica management system.

An example of replica access is shown in Figure 6, in the RCRD frame. Two jobs are submitted to the system by the user. Each specifies a target file, f_1 or f_2 . The requests are translated into replica location operations. The request for f_1 is received first, and the response indicates that the file can be accessed on host c_3 . This host is then marked “busy”. The client then submits the job to host c_3 . The second response arrives at the server, which locates the file on c_3 and c_4 . Since c_3 is busy, the job is sent to c_4 .

4) *Multiple Name Resolution (MNR)* : There is a fundamental difference between the LCRD method and methods SCRD, RCRD. The local computation method selects replicas relative to the given computation site, which is immutable. The scheduled and remote computation methods select replicas relative to a variety of potential compute sites.

SCRD and RCRD present challenges of special interest to designers of grid computing systems. We have two essential properties to examine:

- 1) In an environment in which replica locations are free to change or fail, replica locations may need to be re-selected after job deployment.
- 2) Specifically for scheduled jobs, if the job is not deployed to the specific host that optimizes the file transfer, it may be beneficial to re-select replica locations to minimize transfer relative to the actual computation site.

Clearly property (1) is a harder constraint than property (2) but both represent essential design issues.

Of course, if the job is not collocated with the relevant data, the file access is translated into network file access: in many cases, this can become a performance problem. Thus we optimize performance here by determining which eligible compute site minimizes the network consumption required. Since the data may be reached from anywhere through the adapter, this is a soft constraint and corresponds well to the concept of compute site rank, which allows the user to explicitly indicate preferred sites for computation, and is available in many schedulers. By scaling the rank for each eligible host to the inverse of the required network consumption, we can effectively move the system to an efficient operation mode while maintaining high throughput.

A potential solution involves resolving the replica locations twice. A first resolution is performed by the job submission routine, which now selects only the computation site. The computation site is chosen in such a way that the resulting file transfers will be minimized. Then, after deployment, we have a second resolution: the compute job again resolves replica locations, essentially performing an LCRD operation described above. At this point, all replica locations may change due to storage server failure or computation site surprises, but the selection of compute site is fixed, greatly simplifying the choice. This may produce very good throughput at the small but significant cost of a second query to the centralized replica location service.²

In summary, we have a scheduled method similar to the SCRCD but more robust and efficient because of the complex handling of replica locations.

²A second query is not absolutely necessary: results from the first query could be packaged, deployed, and reused. There is a fault-tolerance tradeoff here that is outside the focus of this paper.

Below, the algorithm for the Multiple Name Resolution method is outlined.

Job Submission

The following algorithm is performed by the Run client.

- 1) For each potential compute host c_i , compute the network transfer n_i required to perform computation on that host.
- 2) Compute appropriate ranks and submit the Late Resolution algorithm as a job to the scheduler.

Late Resolution

The following algorithm is performed by the job upon arrival on a compute host.

- 1) Determine the host this task is occupying.
- 2) Locate all required files, and prefer locations that are on this host.
- 3) Resolve abstract file locations to virtual file locations in the user job argument string.
- 4) Execute the user job atop the adapter.

This algorithm is illustrated in Figure 6, in the MNR frame. In a manner similar to the SCRD illustration, the user submits jobs, the replica management system suggests appropriate hosts, and the jobs are sent to the external scheduler. However, the external, replica-unaware scheduler places $\text{job}(f_2)$ on c_3 first, then places $\text{job}(f_1)$ on c_4 . Simply applying the SCRD method here would cause two network file accesses: each job would begin accessing files found on a different host. However, using the MNR, $\text{job}(f_2)$ utilizes the Late Resolution method and obtains access to the local copy of f_2 . $\text{job}(f_1)$ utilizes the Late Resolution method, and cannot locate a local copy of f_1 or the originally preferred copy on c_3 , but is able to locate the copy on c_2 . The net result is that one job obtains access to a local file, and one job must access a file over the network.

D. Summary

Overall, these four methods provide quite a variety of utility for users in various computation environments.

LCRD jobs can be executed in any environment in which a replica location service is available, creating a useful and practical prototyping tool for running simulation in the presence of any replica location system. They even can be submitted to job schedulers, implicitly creating an

“unguided” MNR method. However, they do not give optimal performance because the jobs are not located close to the data.

SCRD jobs provide a useful and often requested additional functionality to existing job schedulers: they allow matchmaking based on replica location. However, once the job arrives, it functions as a LCRD job, meaning that if a different compute site is allocated by the scheduler, all file access must occur over the network, because name resolution has *permanently* occurred³. Practically speaking, these jobs are convenient to construct, because all of the special tools are located in the user environment, i.e., no special functionality is packaged in the submitted job.

RCRD jobs require the user to have compute access to the remote machine, over a system such as SSH or Chirp. There is no ability to delegate to a global scheduling policy, so jobs could execute in a completely haphazard manner. However, jobs and data are guaranteed to be collocated. A special case that would benefit greatly from such a system are Internet computing applications as discussed below, because such applications often require an application-specific job scheduling policy. Additionally these applications must already have a global view on compute host availability, and a method to checkpoint or suspend running jobs based on the activity of the machine owner. The RCRD method would allow such applications to use the data locations as an additional guide in the process.

The MNR is a robust and complex method for job scheduling. By both guiding the job to an appropriate compute site and making corrections upon arrival, it gains the benefits of the replicated data sources and the global view of the scheduler. Practically, it relies upon the ability to package additional code as a wrapper around the user job, which may be a constraint in some cluster or grid environments.

IV. TARGET APPLICATION: MOLECULAR DYNAMICS

An extremely active area of modern research in high performance computing centers around applications in molecular dynamics [18]. Molecular dynamics (MD) is the numerical simulation of bonded atoms and the forces they exert on each other over time. Typical solvated single protein systems on the order of 100,000 atoms require weeks of simulation time on distributed systems to compute motion on the nanosecond time scale. Storage for sufficient post analysis of

³A variation would be to instruct the scheduler that a given job is only eligible to be run on the specified host, and must otherwise wait- which would provide good collocation but would fill job queues in many applications.

this relatively small system is on the order of gigabytes (10^9 bytes) and computational scientists are aggressively working to achieve simulation results on micro and millisecond time scales, rapidly pushing the storage requirements for individual simulations into the terabyte (10^{12} bytes) range. Publicly shared storage repositories for simulations would require petabyte (10^{15} bytes) storage capacity just to facilitate current simulation results.

The combination of MD simulators and modern grid middleware provides the researcher with a powerful tool for performing large numbers of experiments, multiplying the work done and storage required. In fact, the amount of storage required by one user may be much more than is commonly available on one hard disk, and a conventional network of storage server may not be sufficient for a simulation focused research team. Users must look elsewhere to find effective storage.

Additionally, molecular dynamics is an interdisciplinary endeavor which often combines large numbers of researchers from geographically and organizationally distant places. The data from a simulation may be reused to the benefit of all researchers in a given project. This means that users are often interested in sharing data with others which implies that the storage fabric itself is shared. Such shared resources can be federated into large data grids, providing unified access to a very large repository of scientific data.

A. Case Studies

In this section, we discuss two environments in which replica storage systems and job schedulers may be combined using the above principles to increase utilization of existing resources and benefit users by creating useful and efficient virtual workspaces.

1) *The University Network (LAN)*: 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.

For example, an engineering building at the University of Notre Dame, which contains over 200 Linux and Solaris machines, has been combined into a Condor [13] system. The default Condor installation package used on campus includes the Chirp [22] file services, totalling over 7 TB of available distributed storage. Additionally, the University is served by a centralized AFS [10] 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.

New jobs submitted via Condor require that input data must be immediately staged to the host, either explicitly by file transfer or implicitly by AFS. While the local network is capable of serving data for a handful of compute-bound jobs, cases have arisen in which a small numbers of output-heavy simulation jobs have caused debilitating effects on the network. On-demand data staging and transfer in such a setting is not a viable solution.

Consider the following example: user P intends to post-process large output files that have been created by previous simulations. As shown in our introductory timelines, two strategies are available:

Strategy A: P stores the data files in centralized storage. A batch of Condor tasks are submitted to process the data. Each task must stage all of the input data from a centralized file server.

Strategy B: The data files are previously stored in GEMS, and asynchronously replicated by the GEMS system to eligible machines. P then submits a batch of GEMSRUN requests to the GEMS server, which results in a list of potential compute sites that host the required data. The user then specifies those hosts in the Condor submit files via automated client tools, so that the computation proceeds on a host which can serve that data locally.

Such a data management method inherits from the original goals of the Condor system: to improve the utilization and throughput of the whole system by federating existing systems into a flexible, unified resource.

2) *Internet Computing (WAN):* A variety of applications are suited for the “@ Home” computing model, in which volunteers allow their computers to be used for large-scale scientific projects when they would otherwise be idle. Popular examples include SETI @ Home [1], [2] and Folding @ Home [7]. The range of potential applications is limited because of the perceived data movement bottleneck, so applications are typically chosen only if the amount of data transfer is relatively small. The Mersenne Prime project, for example, simply distributes a single number to a host computer, which is then factored by the host, and the primality question is answered by a single bit: prime or composite.

Applications that require heavier amounts of input and output present a list of challenges, including how to move the data to a compute host in the first place. If the host were to disappear, the central server will have done a substantial amount of work for nothing. Once output data is

produced, it must be safely stored to prevent data loss, and it may need to be post-processed or amalgamated with other output data sets.

GEMS offers a strategy to approach these problems. First, by increasing the number of data servers, we decrease the load on the central server. Original input data may then be served once to a set of volunteer hosts, and the replicas will be automatically maintained. Data can then be managed with fairness and load balancing as described previously [24]. Output data would be written to the local disk, and asynchronously replicated. Subsequent post-processing of the output data would be performed by locating existing replicas of the data and computing on the storage sites, without any additional data transfers.

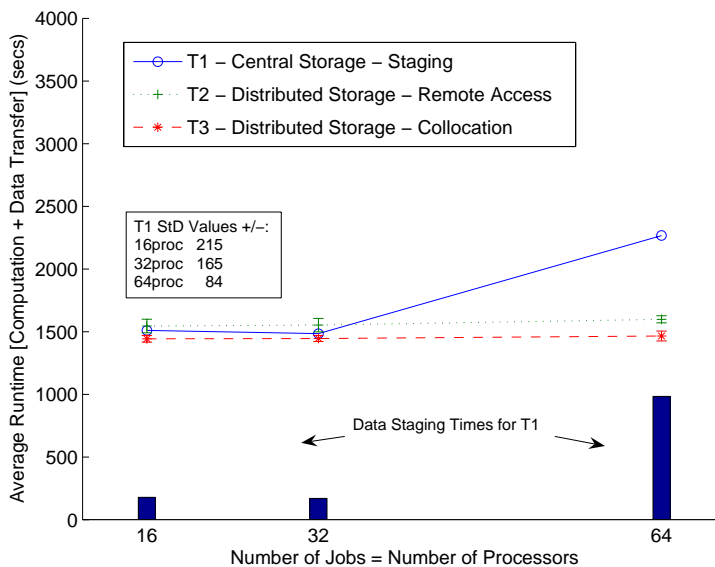
B. Experimental Results

In this section, we describe performance experiments utilizing the replica management system GEMS combined with the Condor job scheduler. We intend to show improvement in the amount of useful work completed for our target application by collocating data and computation with GEMS and performing computation atop the Parrot personal filesystem.

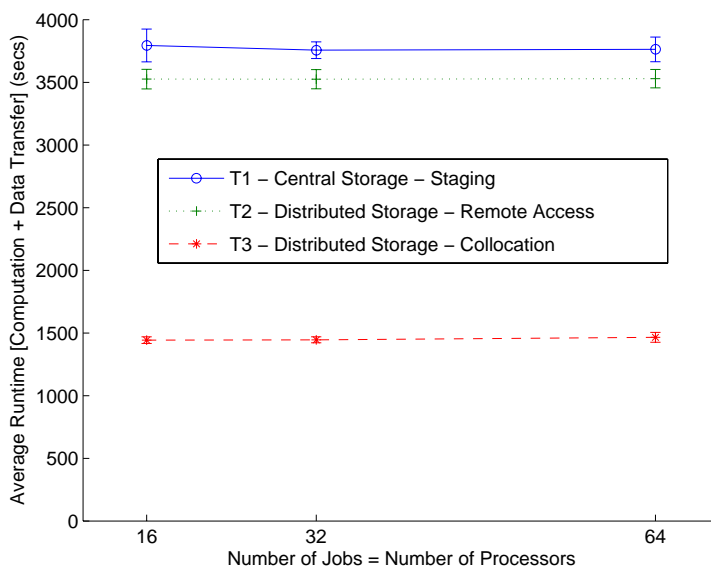
In our experiment, we use VMD [11] to postprocess a 350 MB PROTOMOL [14], [15] MD output trajectory file, performing an RMSD based clustering calculation. This task can be considered input-bound; it is limited by its ability to read the data file. The experiments performed correspond to those environments used by MD researchers that could benefit from replica management systems. We first tested the university network LAN environment as typified by low latency, high bandwidth connections among tightly clustered compute hosts or clusters of hosts. Our experimental testbed typifies this setting, consisting of 32 dual processor (Intel Pentium III 1.4 GHz) machines with 1 GB of RAM, connected by a dedicated 1000Mb interconnect, and running Linux 2.6.9. In this environment we analyzed the average runtime performance of three competing computation and storage models: Staging, Remote Access, and Collocation.

640, 320, and 160 executions of the same postprocessing task were submitted to the cluster and runtimes were averaged with standard deviations indicated by the error bars. As shown in the Cluster Configuration graph of Figure 7, we measured three computation techniques on this testbed.

The “Staging” model stages data from a centralized storage server for each execution of the task, as described by the timelines in Section II. This computation model is shown to be subject



Cluster Configuration



Internet Configuration

Fig. 7. Total job turnaround time for each configuration.

to data transfer scale limitations for processor numbers beyond 32. The observed increase in data staging time when utilizing 64 processors is expected based on the results from Figure 2.

Utilizing the replica-based storage system, the virtual filesystem adapter may be used to gain access to data across the network. “Remote Access” indicates that each VMD job ran atop the adapter, as described in Section III-B, with the constraint that data must be obtained from a *remote* service. Thus each job executed according to the “switchboard” description in Section II-A. This model outperforms the “Staging” method as the data access is parallelized and the application is able to begin processing data as it arrives incrementally.

A third test was performed using “Collocation”, as discussed in Section III-C.4. In this case, jobs are scheduled to only execute on hosts that have the required data file, but externally scheduled in such a way that jobs will not be forced to wait in a queue to gain access to the target compute site. Performance for this test is similar to but slightly better than the “Remote Access” test, indicating that either method is appropriate for computing on high bandwidth clusters. “Collocation” gives the best performance, but is unrealistic in its reliance on access to a single target host. A more relaxed algorithm such as the MNR may be used to capitalize on the good performance from the “Remote Access” test.

A second battery of tests was performed on the same cluster, tuned to perform as an Internet Computing system. The file service protocol on each machine was “throttled” to limit the data bandwidth from any given file server to 1 MB/s. While this testbed does not model all important properties of WAN performance or “@ Home” computing, it is useful to illustrate the idealized performance of the three computation methods.

As shown in the Internet Configuration graph of Figure 7, “Staging” and “Remote Access” perform similarly, as job execution is limited by access to data. Of note is that the implementation of the Staging environment modeled a fast central server with bandwidth much greater than that of the clients. A central server with bandwidth on the 1MB/s scale of the clients would not be practical, yielding projected runtimes of 56000, 112000, and 224000 seconds for the 16, 32, and 64 processor configurations respectively. The performance of “Collocation” is equivalent to that obtained in the “Cluster Configuration”, yielding a speedup of greater than 2 when compared to the Staging and Remote Access models. We see that an SCRD algorithm, as described in Section III-C.2 becomes crucial in order to obtain good performance. On the wide-area network, *jobs must be close to the data.*

V. RELATED WORK

The concepts developed in this work apply to the ongoing task of gluing applications together on the grid, matching users to various resources. Specifically, we address the need to provide access to replica storage systems while negotiating with a scheduler, with the intention of producing a high utilization, high efficiency system.

An existing replica location system offered by the Globus [8], [9] system is the Replica Location Service (RLS) [6] which provides the ability to map logical file names to physical file locations. Systems built upon the RLS must manage user metadata externally. Another Globus project, GASS [5], exemplifies the data staging model of job submission, using cache management strategies to reduce network bandwidth consumption.

The Storage Resource Broker (SRB) [16] 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 [19].

Either of the above 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 [22] 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 [13], GEMS does not require this because of the server-side execution facility provided by Chirp.

An example of grid scheduling is provided by APST [4]. This system can keep track of the location of files with respect to the available storage, and schedule the execution jobs in an appropriate way relative to the location of the data. This project has a similar goal to the computation model in GEMS: to move the computation to the data. However, GEMS is a replica system, which makes its storage model quite different, and increases its ability to find compute hosts that already have the requisite data files present, reducing the amount of time spent moving files.

Another method of relating jobs to data is developed by the NeST project [3], [21], which allows users to discover remote storage resources and reserve the required amount of space for the data staging process and job output. NeST can discover pre-existing replicas of a data source

to save on file staging costs. NeST is built up from the Condor system, and the matchmaking model is tightly linked to the Condor ClassAd database, whereas GEMS is built up from the replica system.

A comprehensive model for large-scale virtual computing system is the Legion system [23]. 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.

The computation models considered in this paper emphasize the ability to treat a remote system primarily as a storage device, but also as a cycle server. This merges with the long established work on active storage systems, which are based on the ability to perform small amounts of preliminary processing or filtering on the storage device before bringing the data set into the general purpose computation model. While our model assumes the use of fully functional remote machines, active storage architectures often delve into lower level hardware functionality [12].

A storage system designed for the application area of molecular dynamics is BioSimGrid [20]. 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.

VI. CONCLUSION

For scientific users conducting experiments that consist of large batches of data-hungry jobs on commodity systems, centralized storage services constrict the throughput of the overall system and dilate the turnaround time of individual jobs. Our benchmarks indicate the degrading quality of service for larger numbers of simultaneous accesses. Replica management systems, which have already been deployed for other reasons including improving long term storage reliability, may be used to improve storage service rates proportionally with the size of the system.

Personal virtual filesystems offer a convenient, location-independent data access method that is invisible to the user code. The combination of such a distributed filesystem with replica management results in a parallelizable, scalable system that was demonstrated to outperform centralized storage.

By distributing the data sets across the compute network, opportunities occur in which jobs may be colocated with existing input data sets, again greatly improving performance. Various

methods to submit or schedule jobs with respect to existing data sets were described and compared.

A replica management system called GEMS⁴ 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. Experiments show much better performance when accessing data replicas rather than centralized storage.

REFERENCES

- [1] D. P. Anderson. BOINC: A system for public-resource computing and storage. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, 2004.
- [2] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@home: An experiment in public-resource computing. *Communications of the ACM*, 45, Nov 2002.
- [3] J. Bent, V. Venkataramani, N. LeRoy, A. Roy, J. Stanley, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau, and M. Livny. Flexibility, manageability, and performance in a grid storage appliance. In *Proc. IEEE International Symposium on High Performance Distributed Computing*, 2002.
- [4] F. Berman, R. Wolski, H. Casanova, W. Cirne, H. Dail, M. Faerman, S. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Shao, S. Smallen, N. Spring, A. Su, and D. Zagorodnov. Adaptive computing on the grid using AppLeS. *IEEE Transactions on Parallel and Distributed Systems*, 14, 2003.
- [5] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke. GASS: A data movement and access service for wide area computing systems. In *Proceedings of the Sixth Workshop on I/O in Parallel and Distributed Systems*, 1999.
- [6] A. L. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman, and R. Schwartzkopf. Performance and scalability of a replica location service. In *Proc. IEEE International Symposium on High Performance Distributed Computing*, 2004.
- [7] V. Pande et al. Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing. *Biopolymers*, 68, 2003.
- [8] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11, 1997.
- [9] The Globus Alliance. <http://www.globus.org> .
- [10] 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*, February 1988:1613.
- [11] W. F. Humphrey, A. Dalke, and K. Schulten. VMD – Visual Molecular Dynamics. *J. Mol. Graphics*, 14:33–38, 1996.
- [12] K. Keeton, D. A. Patterson, and J. M. Hellerstein. A case for intelligent disks (IDISKs). *SIGMOD Record*, 27(3), 1998.
- [13] M. Litzkow, M. Livny, and M. Mutka. Condor - A hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, June 1988.
- [14] Thierry Matthey. *Framework Design, Parallelization and Force Computation in Molecular Dynamics*. PhD thesis, University of Bergen, Bergen, Norway, 2002.

⁴GEMS is an open source project, available at <http://sourceforge.net/projects/gems-nd>.

- [15] Thierry Matthey, Trevor Cickovski, Scott S. Hampton, Alice Ko, Qun Ma, Matthew Nyerges, Troy Raeder, Thomas Slabach, and Jesús A. Izaguirre. PROTOMOL: An object-oriented framework for prototyping novel algorithms for molecular dynamics. *ACM Trans. Math. Softw.*, 30(3):237–265, 2004.
- [16] 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.
- [17] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon. Design and implementation of the sun network file system. In *Proc. USENIX*, 1985.
- [18] T. Schlick. *Molecular Modeling and Simulation - An Interdisciplinary Guide*. Springer-Verlag, New York, NY, 2002.
- [19] 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 *Proc. Supercomputing*, 2003.
- [20] 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.
- [21] D. Thain, J. Bent, A. Arpaci-Dusseau, R. Arpaci-Dusseau, and M. Livny. Gathering at the well: Creating communities for grid I/O. In *Proc. of Supercomputing 2001*, 2001.
- [22] D. Thain, S. Klous, J. Wozniak, P. Brenner, A. Striegel, and J. Izaguirre. Separating abstractions from resources in a tactical storage system. In *Proc. Supercomputing*, 2005.
- [23] 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 *Proc. Supercomputing*, 2001.
- [24] J. M. Wozniak, P. Brenner, D. Thain, A. Striegel, and J. A. Izaguirre. Generosity and gluttony in GEMS: Grid-Enabled Molecular Simulation. In *Proc. IEEE International Symposium on High Performance Distributed Computing*, 2005.
- [25] 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.