Replica Traffic Manager for Data Grids

Adam H. Villa and Elizabeth Varki
University of New Hampshire
Department of Computer Science
Durham, NH, 03824 USA

Abstract

Replication is used in data grids to improve the efficiency of user requests. One or more copies of files are stored on storage components, replicas, in the data grid. Currently, users send their requests directly to these replicas. There is no control over a request once it leaves the user. The focus of this paper is controlling workload traffic at data grid replicas, by managing the flow of requests to each replica. We propose the creation of a replica traffic manager that controls workload traffic sent to the individual replicas in the data grid. The traffic manager receives all user requests and manages the traffic for all replicas, by maintaining a certain number of outstanding requests at each replica. When a particular replica is heavily loaded, all incoming requests for that replica would be held in queue at the traffic manager and/or directed to another replica. Once the traffic decreases at the replicas, the queued requests would be immediately forwarded. By limiting the traffic to each replica, the traffic manager has more control over the system than otherwise possible with individual users submitting requests directly to the replicas. In our evaluations, we observe that our replica traffic manager has a significant and beneficial effect on the performance of the data grid. It provides reliable and consistent response times for user requests. In addition, the traffic manager can adjust for overloaded or failing replicas, while ensuring a starvation free environment that provides dynamic load balancing.

1. Introduction

When a user requests a file from a data grid, servicing the request could consume large amounts of bandwidth and resources while the file is transferred from the storage system to the user. As the number of users increases, the consumption of resources could reach maximum capacity, forcing users to wait. A solution to this problem is data replication. Replication is used to reduce access latency and bandwidth consumption [16]. It also helps in load balancing and can improve reliability by creating multiple copies of the same data [19]. One or more copies of particular files are stored on different storage components, replicas, in the data grid. Users can then submit requests to any of these replicas.

Currently, users send their requests directly to the replicas in the grid. There is no control over the request once it leaves the user. Therefore it is possible that certain events could occur that would affect the performance of the data grid. For example, if the same idle replica is selected to serve multiple user requests, it is possible that users submit their requests to the replica at the same time, causing it to become overloaded. The traffic is not managed for any replicas, so the traffic patterns could vary drastically, also affecting system performance.

The replica traffic manager proposed here would work in conjunction with the replica management service, which manages the replicas in the data grid. This management service is a component of the Grid system software, or middleware, which facilitates writing grid applications and manages the underlying grid infrastructure [3]. Specifically, the replica management service is responsible for managing the replication of complete and partial copies of data sets. The service provides several functions: registers new copies in a catalog, allows users to query this catalog to find all existing copies of a particular file or collection of files, and selects the "best" replica for access based on storage and network performance predictions provided by a Grid information service [4, 5]. More detailed information about middleware and replica management can be found in Section 2.

The steps currently taken for a user's request to be serviced in a data grid are described in [5]. When users need data from the data grid, they submit the logical file names of this data to the replica management service. The service determines a list of physical locations for all registered copies of the logical file names, using its replica catalog. The list of replica locations is then sent to a replica selection service, which identifies the source and destination storage system locations for all possible data transfer operations. The replica selection service sends the possible source and destination locations to one or more information services, which provide estimates of transfer performance based on grid measurements and/or predictions. Using these estimates the replica selection service chooses the best replica and returns the location information for the selected replica to the user, who then forwards the request

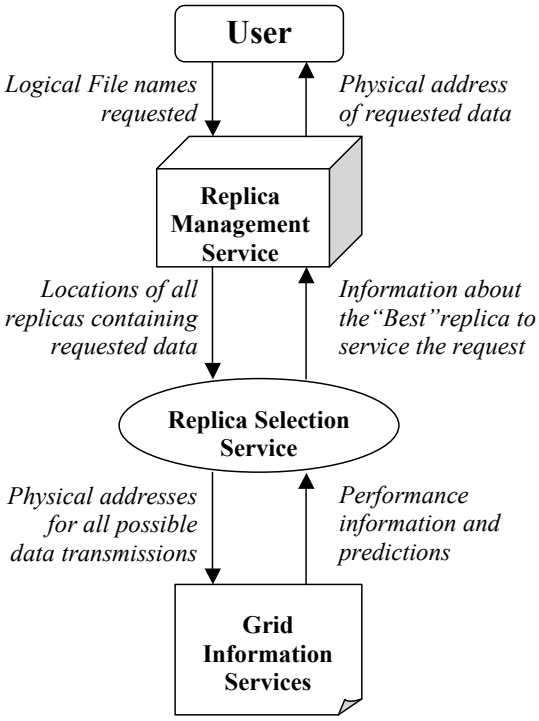

Figure 1. Determining a replica to service a user's request

to the replica. Figure 1 illustrates this process.

The replica traffic manager that we propose would take control after users submit their requests to the replica management service. It would receive all user requests and manage the traffic for each replica in the data grid. When a replica is heavily loaded, all incoming requests for that replica would be forwarded to a less loaded replica and/or held in queue at the traffic manager. Once the traffic decreases at a replica, any queued requests would be immediately forwarded.

By controlling the traffic to each replica, the traffic manager has more control over the system than otherwise possible with individual users submitting requests directly to the replicas. Users with specific quality of service (QoS) requirements can also be satisfied. The traffic manager can adjust for overloaded or failing replicas. It can also ensure a starvation free environment and provide dynamic load-balancing.

The focus of this paper is controlling workload traffic at replicas in a data grid, by managing the flow of requests to the individual replicas in the system. The contributions of this paper are:

- showing the need for a replica traffic manager whose job is managing the traffic to each replica in the data grid and
- evaluating possible traffic controlling policies for disk I/O.

The paper is organized as follows. Section 2 lists related work. Section 3 describes the replica traffic manager. Section 4 explains our evaluations. Conclusions and future work are presented in section 5.

2. Related Works

Grid computing has emerged as a framework for supporting complex computations over large data sets. Applications suitable for grid computing often involve large amounts of data and/or computing and frequently require secure resource sharing across organizational boundaries, and thus are not easily handled by today's Internet and Web infrastructures [17]. Large, complex tasks can easily be completed in a grid environment, since grids are able to efficiently share and manage computing resources [10]. Grids also provide coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [12]. Virtual organizations [3] could be departments in a small university that are in the same physical location or could be large groups of people from various offices that are dispersed around the world. These virtual organizations can be large or small, static or dynamic, and can come together for a particular event and then be disbanded once the event transpires [3].

In some virtual organizations, extremely large data sets with gigabytes or terabytes of data are shared between geographically distributed users. In the near future, several physics projects will produce multiple petabytes of data per year [17]. Users need efficient and reliable access to these data sets. In [9], the authors propose design principles for a data management architecture, a data grid, that satisfies these needs. In a data grid, data may be stored in different locations and on different devices. Users do not need to be aware of the specific low-level mechanisms required to access data from a particular location. They should be presented with a uniform view of data and with uniform mechanisms for accessing that data [9]. Data grids provide a unified interface for all data repositories in an organization and allows data to be queried, managed, synchronized, and secured [17, 10]. Data grids have been adopted as the next generation platform by many scientific communities that need to share, access, transport, process and manage large data collections distributed worldwide [23].

Replication is used in data grids to reduce access latency and bandwidth consumption [16]. It also helps in load balancing and can improve reliability by creating multiple copies of the same data [19]. There are several replication strategies used in data grids. These strategies can be separated into two categories: static and dynamic. In static replication systems, replicas are specifically created on storage components and remain in place. In contrast, dynamic replication automatically creates and deletes replicas in order to follow continually changing system parameters and user access patterns, which keep the performance high and resource usage in reasonable limits [19, 20].

Replicas created by either static or dynamic strategies are managed by a replica management service. This service is a component of the Grid system software, or middleware, which facilitates writing grid applications and manages the underlying grid infrastructure [3]. The infrastructure used in most data grids is coordinated by the Globus Toolkit middleware services. The Globus Toolkit contains code contributed by many organizations. It is the result of the Grid community's attempts to solve real problems that are encountered by real application projects [1]. It contains components that are useful in addressing problems that arise during implementations of Grid applications and systems. These components are generalized so that they function within a wide variety of applications [1].

One of these components is the replica management service, which is responsible for managing the replication of complete and partial copies of data sets. The service provides several functions: registers new copies in a catalog, allows users to query this catalog to find all existing copies of a particular file or collection of files, and selects the "best" replica for access based on storage and network performance predictions provided by a Grid information service [4, 5].

An important service provided by the replica management system is replica selection. Selecting the appropriate replica to service a user's request is a crucial and sometimes complicated task. In general however, the "best" possible replica is typically selected by using network latencies and storage system service times as predictors for estimated transfer time. Most replica selection policies focus heavily on network traffic and bandwidth. These policies consider network conditions to be the limiting factor in the service time for a user's request. Storage systems, however, play an important role in servicing a request and therefore should be given increased consideration.

The electromechanical nature of storage devices limits their performance at orders of magnitude lower than microprocessor and memory devices and, thereby, creates I/O bottlenecks [11]. In [21], the authors found that disk I/O, using high-end RAID servers, can account for up to 30% of the transfer time in grid transfers and the effect of disk I/O would be even more significant in lower-end disk systems. Additionally, trends in disk storage and networking suggest that disk I/O will be of great importance in the future. Disk capacity has improved 1,000 fold in the last fifteen years, but the transfer rate has improved only 40 fold in the same time period [14]. The ratio between disk capacity and disk accesses per second is increasing more than 10 times per decade, which implies that disk accesses will become even more important [14]. In contrast, network bandwidth continues to grow. Gilder [13] predicted that network bandwidth would triple every year for the next 25 years and his prediction has been accurate so far [14]. Network speed and bandwidth will continue to grow at a much faster rate than disk throughput, and thus disk I/O is a critical component of service time for data grid requests.

The Globus Replica Management architecture provides a service for selecting the "best" replica for access based on storage and network performance predictions provided by a Grid information service [4]. The replica selection process allows an application to choose a replica, from among those in a replica catalog, based on its performance and data access features [22]. Several studies analyze different methods of replica selection. The following [8, 18, 7, 24] presents a few of these varying selection techniques.

The replica management system for the European Data Grid is detailed in [8]. The European Data Grid utilizes a Replication Optimization Service (ROS), whose function is the selection of the best replica of a data file for a given request. They take into account the location of the computing resources and network latencies. Network monitoring services provide the ROS with network latencies, which are used to calculate expected transfer times.

In [18], the authors describe an optimization technique that considers both disk throughput and network latencies when selecting the best replica. Disk throughput however, is not a good indicator of traffic. They utilize the k-nearest neighbor rule as a predictive technique for selecting replicas. When a new request arrives, all previous data is analyzed to find a subset of previous file requests that are similar to the new request, which are the k nearest neighbors. The technique then uses these previous requests to predict the best site that can hold the replica.

Several studies take a completely different approach to replica selection. The following are two examples of such studies. Cai et al. [7] present a peer-to-peer replication location service that allows for self-organization, fault-tolerance and improved scalability. They utilize a distributed hash table to aid in the best replica selection process. In [24], the authors suggest the use of a co-allocation technology for servicing requests. Instead of selecting a single best replica to service the request, the co-allocation scheme selects several replicas and sends a portion of the request to each replica. This technique enables the client to download data from multiple locations in parallel, while reducing the idle time spent waiting for the slowest server.

During our related work search, we did not encounter any component, currently used in data grid environments, that is similar to our replica traffic manager. We also find that replica selection policies only use limited storage performance information when making decisions. Most policies only examine disk service time when deciding storage performance. These policies do not consider workload traffic, which is extremely influential on disk performance. For example, a fast disk, which provides smaller mean seek times, might have a large number of outstanding requests, whereas a slower disk with slightly higher seek times could have only a handful of waiting requests. Most policies would send requests to the fast disk, even though the slower disk could service the requests in less time. Traffic is a very important factor that must be considered when making scheduling decisions. In this paper, we evaluate how a traffic manager impacts the performance of a data grid.

3. Replica Traffic Manager

We propose the creation of a replica traffic manager that controls workload traffic sent to the individual replicas in the data grid. The traffic manager receives all user requests and manages the traffic for each replica in the data grid, by setting a maximum limit on the number of outstanding requests at each replica. When a replica is heavily loaded, all incoming requests for that replica would be directed to another available replica. If all replicas are equally loaded, then the requests are held in queue at the traffic manager. Once the traffic decreases at the replicas, the queued requests would be immediately forwarded.

Our replica traffic manager would take control after users submit their requests to the replica management service. First, the management service determines all of physical locations for the requested data. Using estimates from information services, the replica selection service then chooses an appropriate replica to service the request. The management service finally forwards the request with physical location information about the selected replica to the replica traffic manager.

When the traffic manager receives a request, it analyzes the workload traffic of the replica selected to service the request. If the traffic is low, the request is immediately forwarded. The request is placed in the replica's outstanding request queue and is serviced by its disk controller. If the traffic is heavy however, the traffic manager tries to locate another replica that could service the request. The traffic manager attempts to balance the traffic at each replica when the traffic is heavy. If all replicas are heavily loaded, then the request is placed in the traffic manager's queue. When traffic decreases, requests are forwarded from the traffic manager to the appropriate replicas for servicing. The traffic manager queue is serviced in a FCFS manner, which can be considered as a worst-case scenario. A FCFS policy is generally regarded as an inefficient scheduling policy for disks, however when requests have specific quality of service requirements, it allows us to efficiently satisfy them. FCFS is shown to be an effective scheduling policy under similar circumstances [15].

We evaluate our replica traffic manger in the following section.

4. Evaluations

In our preliminary evaluations, we evaluate the performance impact of our replica traffic manager on disk I/O, since hard disk drives are bottleneck devices and can greatly affect the system performance of a data grid [11]. We find that the replica traffic manager produces a significant impact on the performance of disk I/O.

4.1 Evaluation Setup

We evaluate the performance impact of a replica traffic manager by using a storage system simulator, DiskSim [6]. We choose to model each replica as a single disk, since many replicas in data grids are simple workstations with available storage. It also provides for a worst-case modeling of advanced storage devices, like RAID arrays that would be more efficient than a single disk. The specifications for the hard disk drive used in our model are presented in Table 1.

Disk Specifications	
Name:	Cheetah 9.2 LP
Model #	ST39102LW
Storage Capacity:	9.1 GB
Rev. per minute:	10,000 RPM
Interface:	Ultra2 SCSI
Scheduling Policy:	SCAN

Table 1. Hard disk drive specifications [2]

In our evaluations, we model a simple data grid composed of two replicas that contain the same data and have the same hardware. We evaluate the use of our replica traffic manager compared to unmanaged traffic when servicing requests in this data grid.

In unmanaged traffic, users submit their requests directly to any of the replicas in the data grid and there is no control over the routing of their requests. The users submit their requests to the replica with the shortest queue.

The storage system simulator creates different user workloads where there are between 1 and 100 outstanding requests always present in the data grid. The requests submitted by all users are read requests for random data locations. The sizes of these requests follow an exponential distribution, where the mean request size is 512MB. We choose this request size for our preliminary evaluations, since many organizations are sharing large datasets, possibly several terabytes.

We analyze the response time values for the user workload requests. The response time is the time between issuance of the request and the receipt of the requested data. We also examine the range of possible response time values, or standard deviation, as well as maximum values.

4.2 Observations

We observe a significant difference in performance between unmanaged traffic and the use of our replica traffic manager. Unmanaged traffic exhibits a larger range of possible response time values, as well as greater maximum values.

The range of possible response time values for unmanaged traffic is vastly larger than observed with the replica traffic manager. Figure 2 demonstrates this increased variance by showing the response time standard deviations from our evaluations. It is evident that as the number of outstanding requests increases, the response time standard deviation grows much faster when the traffic is unmanaged. For example, when there are 100 requests

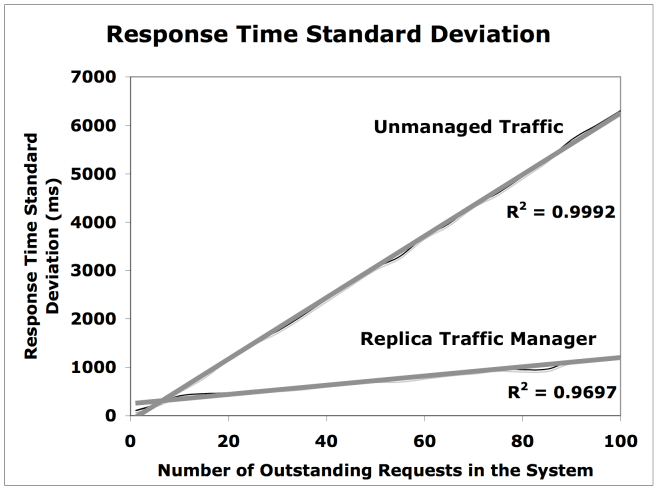

Figure 2. Response time standard deviations (thin black lines) with linear regression lines (thick grey lines).

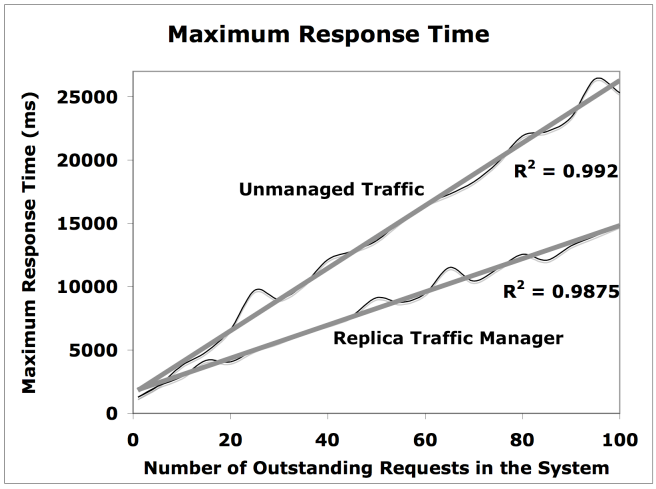

Figure 3. Maximum response time values (thin black lines) with linear regression lines (thick grey lines).

present in the system, the unmanaged traffic produces a response time standard deviation that is 416% larger than when using the traffic manager.

In addition to a larger range of possible values for response time, the unmanaged traffic also exhibits higher maximum values for response time. Figure 3 shows the maximum response time values for our observations. As the number of requests increases, the unmanaged traffic has much larger maximum values. In fact, when there are 100 requests in the system, the maximum response time for the unmanaged traffic is 75% higher than the replica traffic manager's maximum response time.

To gain further insight into these differences, we

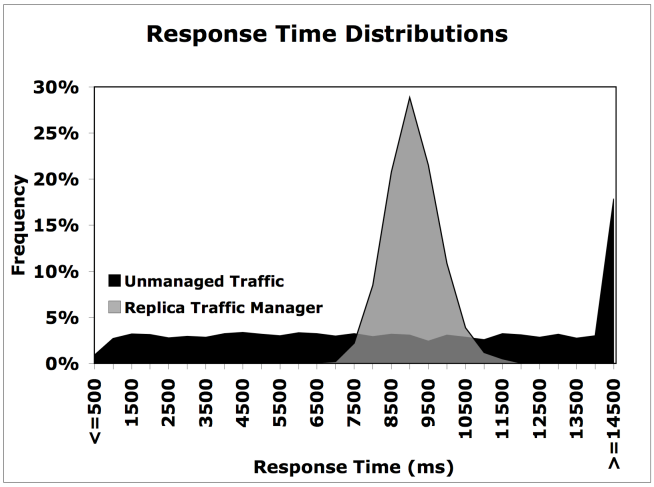

Figure 4. Response time distributions for 80 outstanding requests in the data grid.

closely examine the complete response time distributions when there are 80 requests in the system. Figure 4 demonstrates the difference between these distributions. It is clear that the replica traffic manger produces a much narrower range of possible values for response time. In fact, 91% of all requests can expect response time values within a range of 2000 ms. The unmanaged traffic, however, has a much wider range and 91% of all requests can expect to have response times within a range of 14000 ms. A user has a greater chance of waiting longer when the traffic is unmanaged. For example, when the traffic manager is used, 6% of requests have response times greater than 10000 ms. An unmanaged traffic, however, has a much larger percentage of requests, 41%, that can expect response times of that size.

To further our study, we expand our evaluations to a data grid system that contains four replicas. We observe similar results to our previous findings. The range in response times is extremely larger for unmanaged traffic. Figure 5 demonstrates this increased variance by showing the response time standard deviations for our four-replica evaluations. When there are 100 outstanding requests in the system, unmanaged traffic produces a response time standard deviation that is 303% larger than we observe with our replica traffic manager. In addition, the maximum response time for the unmanaged traffic is 71% higher.

Overall, our evaluations demonstrate that our replica traffic manager has a significant and beneficial effect on the performance of the data grid. The response time standard deviations for the replica traffic manager illustrate the small range of possible response time values in comparison to unmanaged traffic. The traffic manager also provides reliable and consistent response times for users' requests. It ensures that a request will not incur unnecessarily long wait times or be a victim of starvation. It also satisfies re-

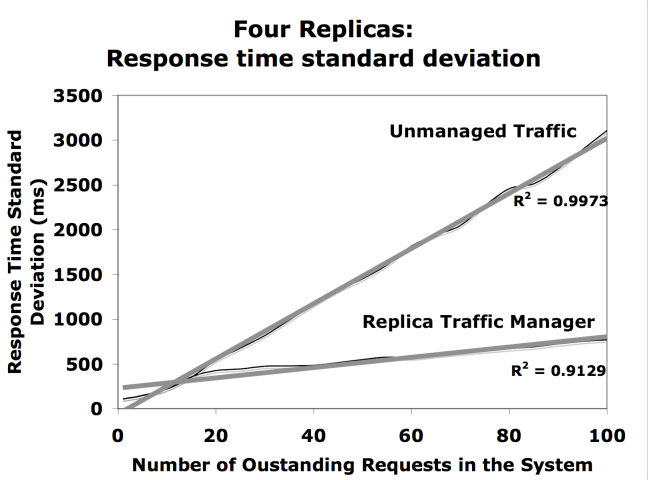

Figure 5. Four replica evaluation: Response time standard deviations (thin black lines) with linear regression lines (thick grey lines).

quests with quality of service requirements that have response time constraints.

4.3 Analysis of Evaluations

In our observations, we found that there is a significant performance difference between unmanaged traffic and our replica traffic manager. The traffic manager provides response time values that fall within a narrow range of values. Unlike unmanaged traffic, which has a vast range of response time values and has much larger maximum response times.

We can attribute this difference in part to the queuing of outstanding requests. When the replica traffic is unmanaged, users submit their requests directly to the replica with the shortest queue. The queue of outstanding requests at these replicas continues to grow as the number of requests increases. The replica's queue is serviced by the disk controller using a SCAN scheduling policy, which chooses requests to service depending on the current position of the disk's read/write heads. Requests are not necessarily serviced in the order in which they arrive at the disk. Therefore, requests will spend a varying amount of time in the queue, which produces large response time standard deviations.

When the replica traffic manager is used however, a small number of requests are present at each replica in the data grid. The lengths of the queues at the replicas' disk controllers are much smaller. This indicates that a request will spend less time in these queues and therefore the variance of queue times is minimized. Meanwhile, all other outstanding requests, waiting to be sent to the replicas, are in queue at the traffic manager. The traffic manager services its queue in a FCFS manner. This scheduling policy produces more consistent queue times for user requests, since requests are serviced in the order in which they arrive. The majority of the variance in response times for the replica traffic manager can therefore be attributed to the varying queue times experienced at the replica.

The variance in outstanding requests' queue times is affected by the scheduling policy servicing the queue. The replica traffic manager produces a narrower range of response times, since the FCFS scheduling policy services requests in fair manner. In addition, as the queue lengths increase, the variance also increases. This is turn affects the response time and its standard deviation. As the demand on replicas increases, the replica management component of the grid middleware will automatically create new replicas to meet the increase in demand. Therefore, extremely long queue lengths at the replica traffic manager should not occur.

By controlling the workload traffic at the traffic manager, we are able to provide reliable and consistent response times. The response time standard deviations are much smaller for the replica traffic manager, which demonstrates the narrower range of possible response times. The maximum response time values are also decreased when the traffic manager is utilized.

5. Conclusions and Future Work

Currently, data grid users send their requests directly storage system replicas. There is no control over the request once it leaves the user. The traffic is not managed for any of the replicas, so the traffic pattern could vary drastically, affecting system performance. Replicas could become overloaded or fail and the performance of the data grid would suffer.

We find that there is a general need for more studies that analyze the impact of storage systems and disk I/O on data grid performance. We propose the creation of a replica traffic manager whose job is managing the traffic to each replica in the data grid. The replica traffic manager would take control after users submit their requests. It would receive all user requests and manage the traffic for each replica in the data grid, by maintaining a certain number of outstanding requests at each replica. When all replicas are heavily loaded, all incoming requests for that replica would be held in queue at the traffic manager. Once the traffic decreases at the replica, the queued requests would be immediately forwarded.

By controlling the traffic to each replica, the traffic manager has more control over the system than otherwise possible with individual users submitting requests directly to the replicas. The traffic manager can adjust for overloaded or failing replicas and can service requests with quality of service requirements. It can also ensure a starvation free environment and provide dynamic load balancing.

In our evaluations, we observe that our replica traffic manager has a significant and beneficial effect on the performance of the data grid. The response time standard deviations for the replica traffic manager illustrate the small

range of possible response time values in comparison to unmanaged traffic. The traffic manager also provides reliable and consistent response times for users' requests. It ensures that a request will not incur unnecessarily long wait times or be a victim of starvation.

For future work, we would like to evaluate the replica traffic manager under varying storage and network conditions. We want to integrate the storage system simulator into a data grid simulator in order to conduct further, detailed evaluations. Real workload traces from actual data grids would also be beneficial to our study.

Acknowledgements

This work was supported by the US National Science Foundation under grant CCR-0093111.

References

[1] The globus alliance, http://www.globus.org/.

[2] Seagate technology inc. cheetah 9lp product manual. Publication number: 83329240, Rev. C., August 1998.

[3] A. Abbas. *Grid Computing: A Practical Guide to Technology and Applications*. Charles River Media, Inc., Hingham, MA, 2004.

[4] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. Secure, efficient data transport and replica management for high-performance data-intensive computing. In *IEEE Mass Storage Conference*, 2001.

[5] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. Data management and transfer in highperformance computational grid environments. *Parallel Computing Journal*, 28(5):749–771, May 2002.

[6] J. Bucy and G. Granger. The disksim simulation environment, version 3.0. Reference Manual. Technical report, School of Computer Science, Carnegie Mellon University, 2003.

[7] M. Cai, A. Chervenak, and M. Frank. A peer-to-peer replica location service based on a distributed hash table. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 56, Washington, DC, USA, 2004. IEEE Computer Society.

[8] D. Cameron, J. Casey, L. Guy, P. Kunszt, S. Lemaitre, G. McCance, H. Stockinger, K. Stockinger, G. Andronico, W. Bell, I. Ben-Akiva, D. Bosio, R. Chytracek, A. Domenici, F. Donno, W. Hoschek, E. Laure, L. Lucio, P. Millar, L. Salconi, B. Segal, and M. Silander. Replica management in the european datagrid project. *Journal of Grid Computing*, 2(4):341–351, 2004.

[9] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *Journal of Network and Computer Applications*, 23:187–200, 2001.

[10] M. DiStefano. *Distributed Data Management for Grid Computing*. John Wiley and Sons, Inc., 2005.

[11] M. Farley. *Storage Networking Fundamentals: An Introduction to Storage Devices, Subsystems, Applications, Management, and Filing Systems*. Cisco Press, 2004.

[12] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *The International Journal of High Performance Computing Applications*, 15(3):200–222, Fall 2001.

[13] G. Gilder. Fiber keeps its promise. *Forbes*, April 7, 1997.

[14] J. Gray and P. Shenoy. Rules of thumb in data engineering. In *IEEE International Conference on Data Engineering*, pages 3–12, April 2000.

[15] V. Hamscher, U. Schwiegelshohn, A. Streit, and R. Yahyapour. Evaluation of job-scheduling strategies for grid computing. *Grid Computing - GRID 2000: First IEEE/ACM International Workshop, Bangalore, India, December 2000. Proceedings*, pages 191–, 2000.

[16] H. Lamehamedi, B. Szymanski, Z. shentu, and E. Deelman. Data replication strategies in grid environments. In *Proc. Of the Fifth International Conferenceon Algorithms and Architectures for Parallel Processing (ICA3PP'02)*, 2002.

[17] D. Minoli. *A Networking Approach to Grid Computing*. John Wiley and Sons, Inc., 2005.

[18] R. M. Rahman, K. Barker, and R. Alhajj. Replica selection in grid environment: a data-mining approach. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 695–700, New York, NY, USA, 2005. ACM Press.

[19] K. Ranganathan and I. T. Foster. Identifying dynamic replication strategies for a high-performance data grid. In *GRID*, pages 75–86, 2001.

[20] R. Slota, D. Nikolow, L. Skital, and J. Kitowski. Implementation of replication methods in the grid environment. *Advances in Grid Computing - EGC*, pages 474–484, 2005.

[21] S. Vazhkudai and J. M. Schopf. Using disk throughput data in predictions of end-to-end grid data transfers. In *Grid Computing - GRID 2002 : Third International Workshop*, volume 2536/2002, pages 291–304, Baltimore, MD, USA, November 18, 2002.

[22] S. Vazhkudai, S. Tuecke, and I. Foster. Replica selection in the globus data grid. In *Proceedings of the First IEEE/ACM International Conference on Cluster Computing and the Grid (CCGRID 2001)*, pages 106–113. IEEE Computer Society Press, May 2001.

[23] S. Venugopal, R. Buyya, and K. Ramamohanarao. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Comput. Surv.*, 38(1):3, 2006.

[24] C.-T. Yang, I.-H. Yang, C.-H. Chen, and S.-Y. Wang. Implementation of a dynamic adjustment mechanism with efficient replica selection in data grid environments. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 797–804, New York, NY, USA, 2006. ACM Press.