Co-allocation in Data Grids: A Global, Multi-user Perspective

Adam H. Villa and Elizabeth Varki

University of New Hampshire, Durham, NH 03824 USA
{ahvilla,varki}@cs.unh.edu

Abstract. Several recent studies suggest that co-allocation techniques can improve user performance for distributed data retrieval in replicated grid systems. These studies demonstrate that co-allocation techniques can improve network bandwidth and network transfer times by concurrently utilizing as many data grid replicas as possible. However, these prior studies evaluate their techniques from a single user's perspective and overlook evaluations of system wide performance when multiple users are using co-allocation techniques. In our study, we provide multi-user evaluations of a co-allocation technique for replicated data in a controlled grid environment. We find that co-allocation works well under low-load conditions when there are only a few users using co-allocation. However, co-allocation works very poorly for medium and high-load conditions since the response time for co-allocating users grows rapidly as the number of grid users increases. The decreased performance for co-allocating users can be directly attributed to the increased workload that their greedy retrieval technique places on the replicas in the grid. Overall, we determine that uninformed, blind utilization of greedy co-allocation techniques by multiple users is detrimental to global system performance.

1 Introduction

Research communities around the world are creating massive amounts of data that need to be accessible to users in various locations. A major creator of such scientific data is the particle physics community. The Large Hadron Collider (LHC), a high energy particle accelerator at CERN, is expected to produce tens of petabytes of raw data annually in 2008 [1,2]. Geographically dispersed researchers are eagerly anticipating access to these datasets. The task of providing fast and efficient data access to these users is a major undertaking for many grid computing research groups.

Replication is used in data grids to help improve users' access to such large and high-demand datasets, by reducing access latency and bandwidth consumption [3]. Replication also helps in load balancing and can improve reliability by creating multiple copies of the same data [4]. 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

S. Wu, L.T. Yang, and T.L. Xu (Eds.): GPC 2008, LNCS 5036, pp. 152–165, 2008.

system parameters and user access patterns, which keep the performance high and resource usage in reasonable limits [4,5]. For example, CERN's replication strategy for the LHC experimental data utilizes a tiered replica structure. Raw data obtained from their instruments is immediately written to the center Tier-0 and is then replicated in a controlled fashion to multiple Tier-1 storage sites [2]. There are several Tier-2 replicas associated with each Tier-1 site that automatically receive copies of the replicated data. End users have the ability to access these Tier-2 replicas that are distributed around the world.

Replicas created by either static or dynamic strategies are managed by a replica management service, a component of Grid middleware. A 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 [6,7]. Selecting the appropriate replica to service a user's request is a complicated and crucial task, in order to minimize a user's response time.

Even with replication and sophisticated replica management services, retrieving these large data files can be extremely time-consuming, especially for users with limited or congested network access. Network latency is a problem experienced by many users. Researchers find that the long latency of data transfer on the Internet often makes it difficult to ensure high-performance access to data and that download speeds are often limited by bandwidth traffic congestion [8].

In order to increase the performance of accessing replicated data, researchers developed the co-allocation technique, which allows a single user to simultaneously utilize multiple resources to service requests. Normally, when users want to retrieve a data file from a remote grid resource, they contact the replica management service to receive a listing of available replicas that contain the specified data. The users would then select a single replica from the listing that would best service their request. Using co-allocation, however, the users could instead utilize many or all of the available replicas. The users would then issue requests for portions of data file from these replicas. The requests would be serviced in parallel and therefore the longest response time that any user would experience would be determined by the slowest replica to service any one of the partial data requests.

Several recent studies, presented in Section 2, develop different co-allocation techniques. Some of these techniques utilize information services, like the Network Weather Service [9], to determine how to split a given request amongst available replicas in order to maximize throughput and decrease network transfer times. Other techniques use past request histories or heuristics to determine a replica's workload. Since network performance can vary greatly, some co-allocation techniques dynamically adjust the amount of data requested from each replica, as data retrieval progresses.

These recent studies compare the efficiency of their new co-allocation technique with previous work and evaluate the performance of their techniques from

a single user's perspective. They do not address situations where multiple users in different locations are simultaneously utilizing their techniques, nor do they discuss the effects that these additional users would have on overall grid performance.

These overall performance effects are significant, since co-allocation increases the workload at the replicas in the system, especially as the number of users utilizing these strategies increases. Instead of a single user issuing a request to a single server, the user could be issuing tens or even hundreds of requests to various servers during the course of file retrieval. This increased workload has a negative effect on the replicas receiving the requests. The impact is even more dramatic for other users in the system that are not using any co-allocation techniques, since co-allocation increases the workload at all of the servers, even though the number of users remains the same.

We find that these recent studies overlook some important issues related to the evaluations of their co-allocation techniques. Firstly, they evaluate their techniques in terms of network transfer time and network throughput. These performance values provide only a limited view of the impact of their techniques. They neglect to examine response times experienced by users. Response time is an important performance value since it includes wait times, which are key indicators of queue lengths at resources in the grid. Without this information, it is difficult to ascertain the conditions of the resources in their experiments. In addition, they do not provide information about replica workloads or the number of users in the grid when their experiments were conducted. This information is important in order to understand and evaluate their results.

The impact of these techniques on the internal systems of the replicas is also not presented in these studies. Prefetching and caching are used by operating systems and storage systems in an effort to decrease costly disk service times. As the number of incoming user requests increases, caches throughout the system will quickly be flushed and any prefetched data will be lost. This will greatly affect disk service times and therefore impact user response times. Disk service time is becoming increasingly important in grid data retrieval, since the network resources involved in transferring the data are becoming increasingly faster and more efficient. 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 [10]. In [11], the authors found that disk I/O, using high-end RAID servers, can account for up to 30% of the service time in grid requests 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 [12]. 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 [12]. 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 [12]. 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.

In addition to examining the impact on the internal systems of a replica, a system-wide evaluation of multiple users utilizing co-allocation techniques has yet to be presented by the grid research community. To the best of our knowledge, we provide the first global, multi-user evaluations of a co-allocation technique for replicated data in a controlled, simulated environment. Our study evaluates the performance effects of multiple users utilizing a co-allocating strategy to retrieve replicated data in a grid environment. We evaluate grid user response times for both co-allocating and normal data retrieval techniques under varying user workloads. We find that there is a significant difference between the response times of both data retrieval techniques. We find that co-allocation works well under low-load conditions when there are only a few users using co-allocation. However, co-allocation works very poorly for medium and high-load conditions since the response time for co-allocating users increases greatly as the number of grid users increases. For example, when there are 85 grid users all utilizing a co-allocating technique, the average response time is 72% larger than the response times would have been if the users simply requested data from a single server. Overall, we determine that the system-wide use of co-allocation techniques for data retrieval can lead to overloading replicated data servers and can be detrimental to global grid performance.

The paper is organized as follows. Related work is presented in Section 2. Our evaluations are detailed in Section 3 and we present our conclusions in Section 4.

2 Related Work

In many research communities, extremely large data sets with gigabytes or terabytes of data are shared between geographically distributed users. Data grids have been adopted as the next generation platform by many communities that need to share, access, transport, process and manage large data collections distributed worldwide [14,15]. Data grids provide a unified interface for all data repositories in an organization and allows data to be queried, managed, synchronized, and secured [16,1].

Grid researchers examine methods for increasing the performance of accessing remote, replicated data. One of these methods is co-allocation, where a single user simultaneously utilizes several resources. There are several recent studies on the topic of co-allocation in grid computing. We present a selection of these studies focusing on the co-allocation of user data requests.

Vazhkudai presents an architecture for co-allocating grid data transfers across multiple connections [17,18]. He illustrates several techniques for downloading data in parallel. The simplest technique is brute force co-allocation, where each replica is assigned an equal portion to service. The author found this method constraining and therefore devised other techniques, including the dynamic load balancing method, which is subdivided into conservative and aggressive load

balancing techniques. The conservative technique monitors the transfer rates of the replicas and automatically adjusts the load for each replica. The aggressive technique progressively increases the amount of data requested from faster replicas and reduces the amount data requested from slower replicas. Overall, the author found that the dynamic methods out performed the static methods. The author's experiments consisted of a single resource requesting data from a small number of servers over the course of several weeks. The results of the experiments were presented as changes in bandwidth (MB/s). The author neglects to mention the workloads of the replicas and overlooks other performance values, such as response time or wait times.

In [19], the authors believe that when several available replicas have almost the same network and disk throughput, choosing a single replica and disregarding the rest is unreasonable and unfair. They therefore developed algorithms that utilize multiple replicas simultaneously. Their workload placement algorithms utilize existing Grid components, such as the replica location service (RLS) [20], the network weather service (NWS) [9] and GridFTP[21]. They developed five algorithms that decide when and which replica should transfer portions of a data file. Their Baseline algorithm simply divides the whole file evenly among all available replicas, while two of their algorithms utilize the NWS to analyze the network throughput of the replicas to make informed scheduling decisions. The final algorithm presented by the authors is NoObserve, which simply uses a fixed-size segment as the basic scheduling unit and each replica is assigned an initial portion to service. When a replica finishes, it is assigned an outstanding portion to service. The authors find that their NoObserve method is characterized by superior performance and simplicity, since it is without infrastructure requirements. In their experiments, the authors have a single resource retrieving data from four servers on a grid system. The results of their experiments are presented as the aggregated bandwidth (KB/s) achieved and the duration (in seconds) of each experiments. The authors do not specify if this duration is only network transfer time or if it includes queue or storage service times, thus its meaning is unclear.

Another co-allocation technique called ReCon (Replica Convoy) is presented in [22]. ReCon simultaneously transfers portions of a source file from several replicas by utilizing several co-allocation algorithms. ReCon's greedy algorithm is similar to the NoObserve [19] algorithm. When a server completes a portion, it is immediately assigned another portion to service. The authors also present a probe-based algorithm for ReCon that uses mechanisms, such as NWS, to test current network performance. In their experiments, they find that the probe-based algorithm provides the best performance. Their experiments consisted of a single machine requesting data from five servers on the PlanetLab grid system. The authors analyze only the transfer times of their algorithms and never specify the workload of the replicas used in their experiments.

The authors of [23,24] attempt to reduce the idle time spent waiting for the slowest replica to service a request by developing a dynamic co-allocation algorithm called Anticipitative Recursive-Adjustment. The algorithm initially

assigns a portion to each replica and then continuously adjusts the workload of each replica by analyzing transfer rates. The algorithm assigns smaller portions of the data file to replicas with slower network connections. In their experiments, the authors examined the bandwidth used by their algorithms and analyzed the differences in transmission times.

In all of the studies presented in this section, we find that the authors developed co-allocation techniques that are inherently selfish and greedy since they utilize as many replicas as possible without consideration for other users. Each user request generates multiple sub-requests and in some cases, the number of secondary requests could be in the hundreds. In addition, these authors analyze their techniques from a single user's perspective and neglect to examine their algorithms from a global, system wide perspective when they are utilized by multiple users. The effects their algorithms have on other users in the grid are also omitted from these studies.

Most of these recent studies examine the results of their experiments in terms of network bandwidth or network transfer times. They focus on changes in network performance values and neglect to examine the effects their algorithms have on service or queue times at the replicas that receive their requests. Many studies also focus on throughput as the most important performance value. However, throughput does not give an accurate representation of overall system performance since throughput can be quite high when a system is overloaded. Response time is a better indicator for user performance.

Co-allocation retrieval mechanisms have yet to be examined from a global, multi-user perspective. Our study provides the first global, multi-user evaluations of a co-allocation technique for replicated data in a controlled grid environment. Our study evaluates the performance effects of multiple users utilizing a co-allocating strategy to retrieve replicated data. We evaluate grid user response times for both co-allocating and normal data retrieval techniques under varying user workloads.

3 Evaluations

In our evaluations we utilize a grid simulator, GridSim [25], in order to conduct repeatable and controlled experiments. The Gridsim simulator also allows us to model realistic network environments [26], including packet schedulers, and components of real data grid systems, such as a replica management service [27]. GridSim also allows us to examine disk I/O overheads, which is an important factor of service time for user requests.

In our experiments, all users and data servers are connected via a high performance router that allows up to 1 GB/s data transfer rates from the data servers and provides users with 100 MB/s data connections. In addition, the only users in the system are those stated in our experiments. Using GridSim, we create an experimental grid environment where ten data servers contain replicated data files. The same files are present on each server and users have the ability to access

any of these servers. Each replicated data server has the same storage hardware, with a fixed average seek time of 9ms.

We design experiments that evaluate a simple, straightforward co-allocation technique, similar to the brute force technique in [17] and the baseline algorithm in [19]. Since all of the users and servers in our evaluations have the same network access rates, our co-allocation technique does not have to adjust for fluctuating network conditions. There is also a fixed number of servers that contain the same data accessible for all users, which allows our technique to utilize all servers in the grid. Additionally, our evaluations are conducted in a controlled environment where no outside users are accessing the system, so our technique need not adjust for other users' workloads.

In order to evaluate our co-allocation technique, we create two users groups called *co-allocated* and *whole_file*. Both user groups have different strategies for accessing the replicated data files. The *whole_file* user group attempts to retrieve a desired data file from a single server. The *co-allocated* user group instead attempts to retrieve an equal portion of the data file from all servers. For both user groups, when a user submits a request to server, it is processed in a FCFS manner and priority is not given to any user. When our experiments start, all servers are idle, all network connections are free of traffic and all users starting submitting their requests at the same time.

Retrieving Large Data Files (1 GB)

We begin our evaluations by creating a series of experiments where the number of users in the grid increases from 1 to 140. We limit the number of users to 140, since in many grid environments dynamic replica creation occurs when traffic on a replica is greatly increased. Each user attempts to retrieve a unique 1 GB data file from the replicas or servers according to their retrieval strategy. A user in the *whole_file* group, selects a replica with the fewest number of outstanding requests and then submits a request for the entire 1 GB file from the selected replica. A user in the *co-allocated* group requests a 100 MB portion of the data file from each replica in the system. When any user receives its entire file, the user's overall response time is calculated and then the user is disconnected from the grid. We examine both user groups under these conditions.

We begin our examination by evaluating the average response time experienced by any given user. The response time for a user's request can be computed by the combination of the network service time and the replica service time. The network service time includes the time spent during transmission and communication between resources, whereas the replica service time includes the time spent performing disk I/O and internal communication. Both of these service times include the time elapsed while waiting to use various resources. These wait times depend on the number of outstanding requests and therefore a larger number of outstanding requests will result in longer wait times. The response time gives an accurate view into the current state of the system.

We calculate the average user response time for both user groups as the number of users in the system increases from 1 to 140 and the results are shown in

Figure 1a. We observe that the average response time for the *co-allocated* user group increases at much faster rate than the *whole_file* user group, as the number of grid users increases. When then number of grid users is less than the number of servers, the *co-allocated* group provides a smaller average user response time. When there are a large number of users however, the *whole_file* group provides the lowest average response times. In Figure 1b, we illustrate the percentage increase in user response times for the *co-allocated* group. For example, when there are 135 users, the average user response time for the *co-allocated* group is 78% larger than the *whole_file* group.

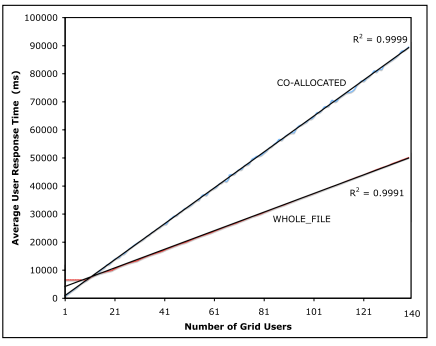
We also evaluate the maximum response times experienced by any given user. In Figure 1c, we show the maximum user response times for both user groups, which are similar. When there are fewer users in the system, the *co-allocated* group provides slightly decreased maximum response times. The difference between the two groups decreases however, as the number of grid users increases. When there are 75 users in the system, the maximum response time for the *co-allocated* group is only 2% lower than the *whole_file* group. In fact, there are several occasions where the *whole_file* group provides smaller maximum response times.

In order to gain further insight into the response time differences between the two groups, we analyze the response time distribution for a specific number of users. In Figure 1d, we illustrate the difference between the response time distributions for both user groups when there are 89 users present in the system. We see that the majority of users in the *co-allocated* group have higher user response times than the *whole_file* users. In fact, 93% of *co-allocated* users have response times greater than 50,000ms. Whereas, 78% of the *whole_file* users have response times less than 50,000ms. The *whole_file* user group also has 23% of users with response times less than 20,000ms.

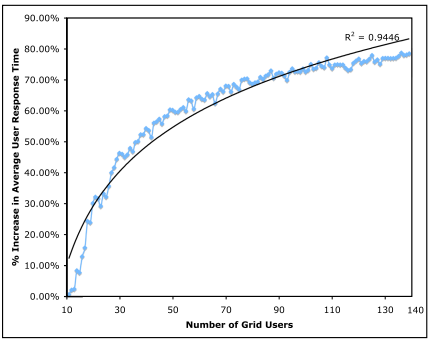
We can attribute the decrease in user response time performance for *co-allocated* users to the increased workload their retrieval mechanism places on the servers in the grid. For every user request, ten user requests are created and one request is sent to all ten servers in the grid. Figure 1e illustrates the dramatic difference in the workload presented to the servers between the *co-allocated* and *whole_file* users. This increased workload directly relates to the decrease in user response time performance that we notice in our evaluations. Even though the co-allocated requests are for smaller file sizes, the shear number of requests waiting at the servers decreases their performance. The queue time for the co-allocated requests grows as the number of grid users increases.

Retrieving Smaller Data Files (100 MB)

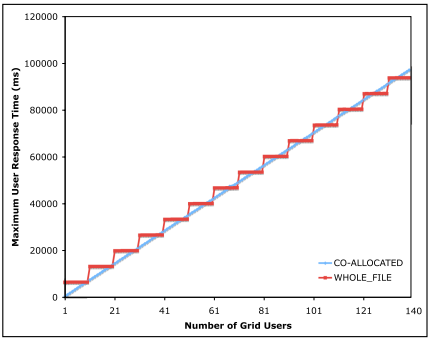
We continue our evaluations by examining the user groups when we decrease the size of the entire data files requested by all users to 100 MB. An evaluation of decreased file size is relevant, since in many instances a user may only require a portion of larger data sets for their computations. We conduct experiments to see how the trends observed with 1 GB data files compare to requesting smaller data files. In this set of experiments all variables are the same except for the



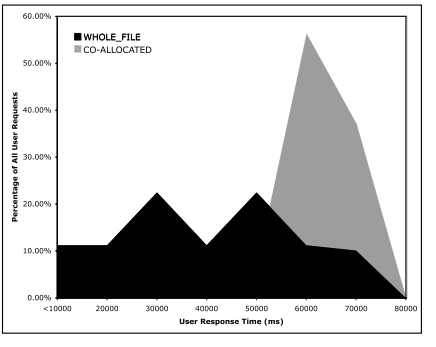
(a) Average user response times



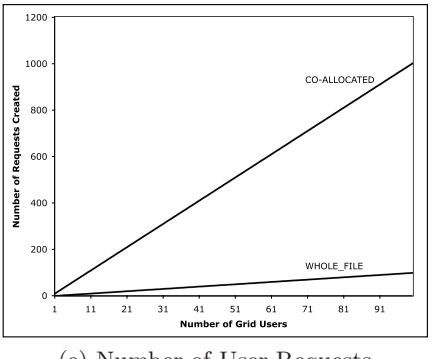
(b) Percentage increase in average user response times



(c) Maximum user response times

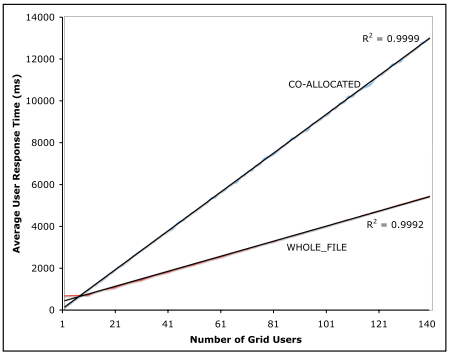


(d) User response time distribution

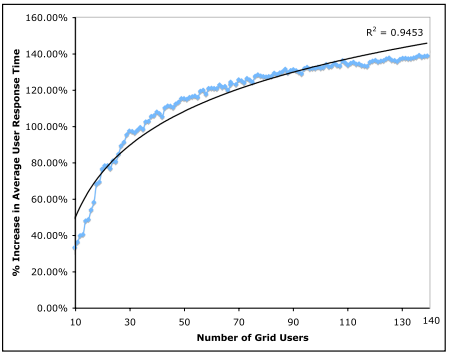


(e) Number of User Requests

Fig. 1. Average response times for both user groups as the number of grid users increases are shown with a tight-fitting, R^2 close to 1, linear trend line (a). The percentage increase for the *co-allocated* users in response time is shown for the number of grid users with an exponential trend line, in black (b). The maximum response times experienced by any grid user (c). The response time distributions for both user groups when there are 89 grid users submitting requests (d). The number of requests generated by each user group, as the number of users increases for both groups (e).



(a) Average user response times



(b) Percentage increase in average user response times

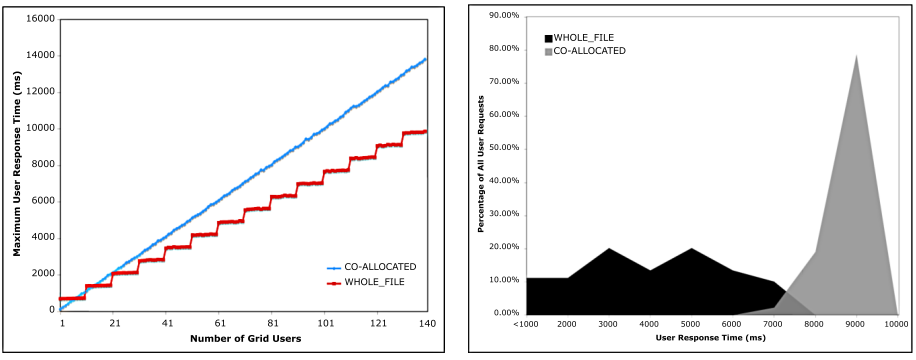
Fig. 2. Average response times for both user groups as the number of grid users increases (a). The users are retrieving a 100 MB data file. Each user group's curve is shown with a tight-fitting, R^2 close to 1, linear trend line. The percentage increase in response time for the *co-allocated* users when they are retrieving a 100MB data file is shown for the number of grid users (b). The curve is shown with an exponential trend line, in black.

requested data file size. A user in the *whole_file* group, selects a replica with the fewest number of outstanding requests and then submits a request for the entire 100 MB file from the selected replica. A user in the *co-allocated* requests a 10 MB portion of the data file from each replica in the system.

We begin by analyzing the average user response times for both groups in Figure 2a. We observe a similar trend in average response time as we did with the larger file size. As the number of grid users increases, the average response time for the *co-allocated* users increases much faster than for the *whole_file* users. There is an even greater difference with the decreased file size, since the transfer time is minimized. For example, when there are 135 grid users, the average *co-allocated* user response time is 138% larger than the average *whole_file* user response time. This greater increase in average user response time is also evident in Figure 2b, which shows the percentage increase in average *co-allocated* user response time over average *whole_file* user response time.

We also find a significant change in the maximum user response times. With larger file size, we find that both user groups have relatively the same maximum user response times. With smaller file size however, we find that there is a noticeable difference between the maximum user response times for both setups as the number of grid users increases. This is illustrated in Figure 3a. We notice that the maximum response time for 130 grid users is 42% larger for the *co-allocated* user group.

To further illustrate the remarkable difference between the two user groups when they are requesting decreased file sizes, we analyze the response time distribution when there are 89 grid users in the system. The response time distributions for both groups are shown in Figure 3b, which demonstrates that 90%



(a) Maximum user response times (b) User response time distribution

Fig. 3. The maximum response times experienced by any grid user, requesting a 100MB file (a). The response time distributions for both user groups when there are 89 grid users submitting requests for 100 MB data files (b).

of *whole_file* user's have response times that are less than any of the *co-allocated* users. In fact, 98% of *co-allocated* users have larger response times than any of the *whole_file* users.

Summary of Evaluations

Overall, we find in our evaluations that the *co-allocated* user group has higher average user response times as the number of grid users increases. The *co-allocated* user group also has response time distributions where a greater percentage of requests have large response times in comparison to the *whole_file* user group. The difference between the two groups is even more distinct when the entire requested data file size is decreased to 100 MB. We find that the maximum and average user response times for *co-allocated* users is drastically larger than the *whole_file* user group. The performance decrease for the *co-allocated* users can be attributed to the increased workload their retrieval mechanism places on the servers in the grid. The increased workload that they generate directly relates to the decrease in user response time performance that we notice in our evaluations.

Our evaluations in GridSim also allow us to examine disk service time as an integral component of overall request service time. Since disk I/O is becoming increasingly important as network technologies improve, it is valuable to have these evaluations where disk service time is computed. The GridSim simulator uses fixed, average values that are based on real disk I/O performance in order to calculate disk service time, such as the seek and rotation times. Since these disk service parameters are static, it is difficult to examine the intricate effects that caching and pre-fetching have on user performance when co-allocation techniques are utilized. We expect that the performance difference between the *co-allocated* and *whole_file* data retrieval techniques would be even more noticeable in real grid systems where disk service time is dynamic, with costly seek and rotation times. The request service time would also be notably affected by caching and pre-fetching mechanisms at many resources throughout the grid.

4 Conclusions

Research communities are generating massive amounts of data that need to be readily accessible to users around the world. The task of providing fast and efficient access to this data is a primary focus for many grid computing research groups. Even with replication and sophisticated replica management services, users still experience delays when accessing large data files from remote servers. Researchers attempt to find new techniques to reduce the amount of time a user must wait to retrieve large data files. The co-allocation technique was developed to serve this purpose.

Current co-allocation techniques for data retrieval from remote, replicated sources have only been evaluated from a single user's perspective. Situations where multiple users in different locations are simultaneously utilizing these techniques have yet to be evaluated. The effects that these additional users would have on overall grid performance are also unknown. For example, the internal systems of data grid replicas would be affected by these additional co-allocating users. Since prefetching and caching are used by a replica's operating systems and storage systems, the increased number of user requests would quickly flush caches throughout the system and any time saving, prefetched data would be lost. This loss will greatly affect disk service times and therefore greatly impact user response times.

Since co-allocation retrieval mechanisms have yet to be examined from a global, multi-user perspective, we develop experiments where we compare users that utilize a straightforward co-allocation technique with users that employ normal data retrieval methods. We evaluate the performance of the co-allocating technique as the number of users in the grid increases, up to 140 users. We find that there is a significant and negative change in performance for the co-allocating users as the number of grid users grows. We find that co-allocation works well under low-load conditions when there are only a few users utilizing the technique. However, we determine that under medium and high workload conditions, co-allocating users have higher average user response times and have response time distributions where a greater percentage of requests have large response times in comparison to normal data retrieval users. When there are 135 grid users all utilizing a co-allocating technique, the average response time is 78% larger than the response times would have been if the users simply requested data from a single server. This decreased performance for co-allocating users can be directly attributed to the increased workload their retrieval mechanism places on the replicas in the grid.

Overall, we find that the global use of co-allocating techniques for replicated data retrieval when done in a greedy or selfish manner is detrimental to user performance in a data grid. Replica workload must be considered when making co-allocated requests in a multi-user environment, in order to prevent overload situations. Only under low utilization conditions should co-allocation be used unchecked.

Clearly, more work is needed to fully understand the specific conditions when co-allocation is beneficial to all grid users. Co-allocation in prior work has been

shown to be very beneficial for single users in isolation. We believe that co-allocation could also be advantageous for multiple grid users, however it must be used cautiously and wisely. More information should be required before users decide which resources they are going to allocate, in order to make knowledgeable and conscientious decisions. There are several possible ways that this task could be accomplished and individual grids might require different techniques. One possible solution could be the creation of a co-allocation management service that monitors the workloads present at each of the replicas in the grid. A user would then have to either contact this service for replica status or receive allocation suggestions from the service before the user could start issuing requests. Another possible solution could be to simply allow a user to make a best guess decision based on existing replica information. These suggested techniques and any other mechanisms for controlling user co-allocation need to be closely examined in order to see how they would intricately affect global grid performance. Since data sets will continue to grow at incredible rates, new co-allocation mechanisms need to be carefully developed and closely evaluated from all perspectives in order to provide users with fast and efficient data access.

Acknowledgements

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

References

1. Minoli, D.: A Networking Approach to Grid Computing. Wiley Press, Chichester (2005)
2. Nicholson, C., Cameron, D.G., Doyle, A.T., Millar, A.P., Stockinger, K.: Dynamic data replication in lcg 2008. In: UK e-Science All Hands Conference, Nottingham (2006)
3. Lamehamedi, H., Szymanski, B., Shentu, Z., Deelman, E.: Data replication strategies in grid environments. In: ICA3PP (2002)
4. Ranganathan, K., Foster, I.T.: Identifying dynamic replication strategies for a high-performance data grid. In: GRID, pp. 75–86 (2001)
5. Slota, R., Nikolow, D., Skital, L., Kitowski, J.: Implementation of replication methods in the grid environment. Advances in Grid Computing, 474–484 (2005)
6. Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., Tuecke, S.: Secure, efficient data transport and replica management for high-performance data-intensive computing. In: IEEE Mass Storage (2001)
7. Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., Tuecke, S.: Data management and transfer in high performance computational grid environments. Parallel Computing Journal 28, 749–771 (2002)
8. Yang, C.T., Yang, I.H., Chen, C.H., Wang, S.Y.: Implementation of a dynamic adjustment mechanism with efficient replica selection in data grid environments. In: SAC (2006)

9. Wolski, R., Spring, N.T., Hayes, J.: The network weather service. Future Gener. Comput. Syst. 15, 757–768 (1999)
10. Farley, M.: Storage Networking Fundamentals: An Introduction to Storage Devices, Subsystems, Applications, Management, and Filing Systems. Cisco Press (2004)
11. Vazhkudai, S., Schopf, J.M.: Using disk throughput data in predictions of end-to-end grid data transfers. In: GRID, pp. 291–304 (2002)
12. Gray, J., Shenoy, P.: Rules of thumb in data engineering. In: IEEE International Conference on Data Engineering, April 2000, pp. 3–12 (2000)
13. Gilder, G.: Fiber keeps its promise. Forbes (April 7, 1997)
14. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., Tuecke, S.: The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. Journal of Network and Computer Applications, 187–200 (2001)
15. Venugopal, S., Buyya, R., Ramamohanarao, K.: A taxonomy of data grids. ACM Comput. Surv. 38, 3 (2006)
16. DiStefano, M.: Distributed Data Management for Grid Computing. John Wiley and Sons, Inc., Chichester (2005)
17. Vazhkudai, S.: Enabling the co-allocation of grid data transfers. In: GRID (2003)
18. Vazhkudai, S.: Distributed downloads of bulk, replicated grid data. Journal of Grid Computing 2, 31–42 (2004)
19. Feng, J., Humphrey, M.: Eliminating replica selection - using multiple replicas to accelerate data transfer on grids. In: ICPADS, p. 359 (2004)
20. Chervenak, A., Deelman, E., Foster, I., Guy, L., Hoschek, W., Iamnitchi, A., Kesselman, C., Kunszt, P., Ripeanu, M., Schwartzkopf, B., Stockinger, H., Stockinger, K., Tierney, B.: Giggle: a framework for constructing scalable replica location services. Supercomputing, 1–17 (2002)
21. Bresnahan, J., Link, M., Khanna, G., Imani, Z., Kettimuthu, R., Foster, I.: Globus gridftp: What's new in 2007. In: GridNets (2007)
22. Zhou, X., Kim, E., Kim, J.W., Yeom, H.Y.: Recon: A fast and reliable replica retrieval service for the data grid. In: CCGRID, pp. 446–453 (2006)
23. Yang, C.T., Chi, Y.C., Fu, C.P.: Redundant parallel file transfer with anticipative adjustment mechanism in data grids. Journal of Information Technology and Applications (2007)
24. Yang, C.T., Yang, I.H., Li, K.C., Wang, S.Y.: Improvements on dynamic adjustment mechanism in co-allocation data grid environments. The Journal of Supercomputing (2007)
25. Buyya, R., Murshed, M.: Gridsim: A toolkit for the modeling and simulation of scheduling for grid computing. In: CCPE (2002)
26. Sulistio, A., Poduval, G., Buyya, R., Tham, C.K.: On incorporating differentiated levels of network service into gridsim. Future Gen. Computer Systems (2007)
27. Sulistio, A., Cibej, U., Robic, B., Buyya, R.: A tool for modelling and simulation of data grids. Technical Report GRIDS-TR-2005-13, Grid Computing Laboratory, University of Melbourne (2005)