Tortoise vs. hare: a case for slow and steady retrieval of large files

Abstract

Large file transfers impact system performance at all levels of a network along the data path from source to destination. The increased traffic and contention for shared resources affects all users on the network and results in degraded performance for these users, especially when the network is under high demand. The current trends in large data retrieval have the **hare** *mentality: retrieve data as fast and hard as possible without regard for other users. We find that a single user utilizing these types of techniques on a large, academic network can quickly and significantly affect network performance at multiple levels. The single user's workload is able to increase the local network utilization from a normal of 30 to 90+%. As a solution to this problem, we find that we can minimize performance impact by using the* **tortoise** *approach of slow and steady data retrieval for large files when the network is under high load. By fairly restricting the transfer rates of large file transfers, we surprisingly find improved performance for all users, including the users that had their transfers restricted.*

1. Introduction

Vast amounts of data are stored in clouds around the world and as clouds increase in popularity, their datasets will only continue to grow. Many research communities have opted to utilize cloud infrastructures to store, process, and share large data files [2, 4, 9]. Particle physics and meteorological research groups produce staggering amounts of data that must be cataloged, processed, and made available for users to retrieve [12, 13]. Clouds allow geographically distributed users to access shared datasets that can range in size from terabytes to petabytes [10, 11, 17]. Users require fast and efficient access to these remote data files.

Retrieving large amounts of data over public, shared networks is a complicated and difficult task. Since files from these datasets are extremely large, these transfers have a long duration. The transfers will take a significant amount of time to complete, during which many incidents may occur causing the transfer to degrade or fail.

The current trend of research on large file transfers is geared towards minimizing a user's service time by any means possible. The goal is to increase and maximize user throughput without regard for overall system performance or stability. This is representative of the *hare* mentality: retrieve data as fast and hard as possible. Many recent studies recommend advanced techniques for data retrieval, which attempt to decrease users' service times. Some of these techniques utilize distributed file retrieval (or data co-allocation), which allows a single user to simultaneously utilize multiple resources to service a request. Instead of retrieving data using a single transfer connection, the user would initiate multiple transfers in parallel. In most cases, the user would be retrieving data from multiple locations simultaneously. There are several different types of data co-allocation retrieval techniques [5, 7, 8, 14, 15, 16, 18, 19]. The techniques vary in how they select and utilize the available data sources. Some of these co-allocation techniques will attempt to open connections to all of the available source locations and others will selectively connect to a subset of the sources. These data co-allocation techniques can be characterized as greedy techniques, since they selfishly use resources without regard for other users.

The studies that present these greedy techniques evaluate their performance solely from a single user's perspective. They demonstrate how a single user can increase throughput and minimize service times. They neglect to examine how these techniques affect the network and other users utilizing shared resources. Moreover, these studies are only evaluated on simulators or limited systems, so the performance impacts on real systems are not known.

The users retrieving these large data files from the cloud are often on public networks, using a shared connection to the Internet. These public networks could be located in an academic campus or in a research institution, where there are potentially hundreds to thousands of users utilizing shared network resources. When users retrieve large files over these shared resources, everyone is affected. As the number of users retrieving data increases, the impact on the performance of the entire system multiplies. As the load grows, eventually there will be packet loss and failures. Transfer performance for all users will degrade as the demand rises. This is especially true when users utilize retrieval techniques that attempt to utilize as much bandwidth as possible.

The impact of large file transfers on other users as well as system resources has not been studied on real, actively used networks. In order to fully understand these impacts, we evaluate large file transfers on an actively used local network and examine their impact on other users. We conduct experiments on real systems to examine the effects of these transfers. In our experiments, we find that system and user performance suffer as the number of users retrieving large files increases. All users are affected by the increased workload of large file retrievals. The impact on other users that are sharing the public resources is significant. We find that a typical user could potentially see a 86% degradation in transfer performance when other users are concurrently retrieving large files.

During our experiments, we observe several interesting findings. The first finding of interest is that a single user on a live, academic network can significantly overload network resources and impact all users' performance. One would assume that a user's bandwidth would be limited or controlled in an academic network to ensure that the system remains stable. We repeatedly find the opposite to be true on our testing network. In our experiments, we configure a single user to utilize multiple machines and retrieve several large files in parallel (similar to the greedy retrieval techniques discussed earlier). We find that this concurrent data retrieval increases the load of the entire local subnet substantially, as well as all users utilizing the shared Internet connections. These high utilizations cause dropped packets and latency issues for all users on the network.

Our second interesting finding is that even with ample bandwidth available for all users, large file transfers still have a negative impact on system performance and cause other users to have degraded performance. Depending on the number of concurrent large file transfers, a typical user's retrieval rate could decrease by as much as 68%.

The third interesting finding is that by placing fair retrieval rate restrictions on large file transfers we are able to improve the performance of other users, as well as maintain adequate service for the users retrieving large datasets. By imposing these restrictions, we find that a typical user retrieving small amounts of data could see a significant improvement (60 to 170%) in transfer rates. When restricted properly, the large data transfers only see a 1 to 4% degradation of performance. Slow and steady is a beneficial and effective method for retrieving large files over heavily loaded networks.

Overall, we find that there is significant negative impact to local system performance when users retrieve large data files over shared, public networks. All resources in the system experience increased traffic and heavier workloads. The increased demand affects the performance of all users in the system. Other users are unjustly penalized and observe decreased transfer times and longer service times. By restricting large file transfers however, the system can maintain adequate performance for all users. Slow and steady does win the race after all.

The paper is organized as follows. Our experiments and observations are detailed in Section 2. We summarize our findings in Section 3 and then discuss our conclusions in Section 4.

2. Experiments

We examine the impact that large data transfers have on all user transfer rates and network load. We utilize two testing environments for our experiments. The first environment is a live, actively used network on a university campus. The second is a controlled, isolated environment where we can manage all aspects of the system and the network. There are two types of users that we examine in our experiments: *normal user* and *large user*. A *normal user* is only periodically retrieving small amounts of data (web traffic, email, small ftp transfers, etc.), where as a *large user* is continually retrieving large data files via multiple parallel connections to remote resources using an application such as gridFTP [3, 6].

2.1 Experimental environment: live academic network:

In our live experiments, we utilize an academic network that services over 10,000 users (Figure 1). We utilize one subnet of the network and all of our data transfers are initiated by client machines on this subnet. The subnet connects to the larger network before accessing the shared Internet connections. During all of these experiments, there are other users utilizing the subnet and shared connections to the Internet. We have no control over these other users' workloads. Since conditions can change dramatically during different periods of the day and different days of the week, we run our experiments several times at various times and over the course of

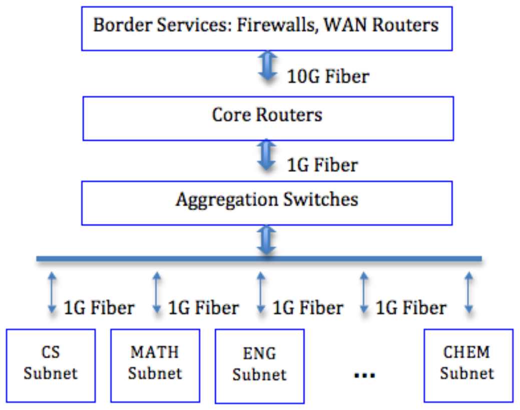

Figure 1: Diagram of live, academic network used in our experiments.

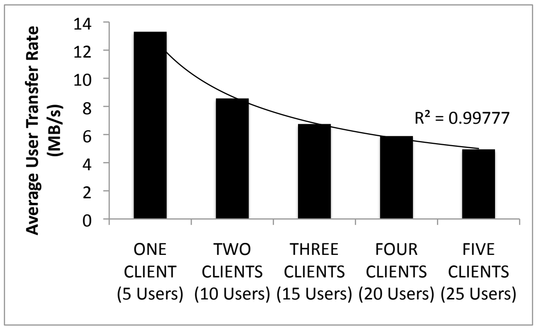

Figure 2: Individual user data transfer rates as the number of clients and user transfers increases.

multiple weeks. We then take an average of all of the data values that we observe.

This set of experiments examines the impact that large users have on system resources as well as each others' performance. We begin our experiments with a single client containing five large users that are each retrieving large data files. We then continue to add additional clients, each with five large user transfers. All of these clients are located on the same subnet and utilize a single connection to the larger network and the shared Internet connection. We find that as the number of user transfers increases, the average transfer rate for each user decreases. As Figure 2 shows, the transfer rate steadily decreases for each additional client containing more user file transfers. We see a 36% decrease in average user transfer rates when the number of user transfers increases from 5 to 10. Plotting a trend line with a R-squared value of 0.9977, which indicates a good fit, predicts that as the number of users increases the average user transfer rate will continue to decrease.

The total combined transfer rates for all users is displayed in Figure 3. As the number user workloads in-

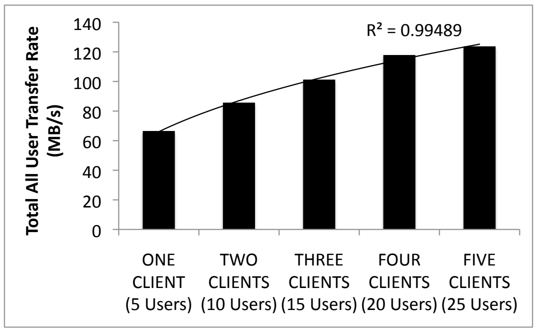

Figure 3: Total combined user transfer rates as the number of servers and user transfers increases.

creased, the total transfer rate also increased. From 5 users to 10 users, the total transfer rate increased from 66.5 to 85.7 MB/s, a 29% increase. However, an individual user's transfer rate dropped by 36% (as shown in Figure 2). This shows the negative impact that large users have on system performance. When increasing the workload from 20 to 25 users, the total transfer rate only increased by 5%. This suggests that our experimental workload is beginning to fully utilize and overwhelm the network. If more users were added, we expect to see a decrease in total transfer rates as a result of overloading system components and dropping packets.

During our live experiments, we were contacted by the local subnet support team as well as the university's telecommunication department. Our experiments, with a total of only 25 concurrent user transfers, were able to significantly impact the service of the subnet as well as the general Internet connections. The subnet's link to the rest of campus normally has a load of about 30%. During our experiments, we increase the load to 90+%, which resulted in dropped packets and service problems for all users, including users on other campus subnets. We also impacted the performance of users on other subnets by monopolizing access to the shared Internet connections. If there were even more user workloads retrieving large files, we expect to see far worse conditions.

Analysis: The results from the first experimental environment are surprising. We expected that on a heavily used and well-managed network our large file transfers would be controlled and restricted. We did not think that our experimental workload would be able to affect the performance of our subnet, let alone other campus subnets, and the general shared Internet connections so dramatically. As the number of users accessing and retrieving shared datasets in clouds increases, the demand on academic networks will increase. System administrators need to be prepared for this type of workload and be able to maintain a well-balanced system in spite

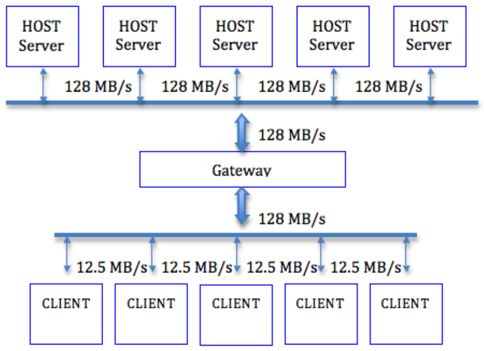

Figure 4: Controlled environment setup.

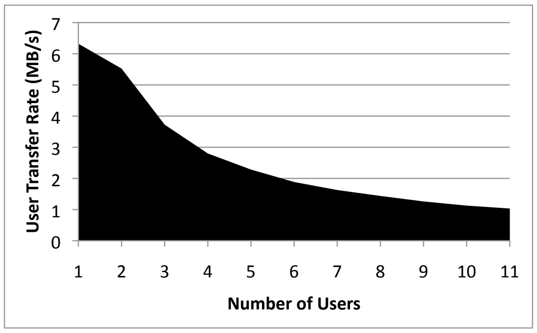

Figure 5: User's data transfer performance as the number of user data requests increases on a single client.

of the demanding requests.

2.2 Experimental controlled environment

Since we are unable to view the exact configuration and complete performance of the live network, we create a controlled testing environment where we setup and monitor all aspects of the system. We create a network with two subnets, *host* and *client*, each containing five servers. The *host* subnet's servers store the data files and service the users' data requests. The *client* subnet's servers allow users to request data files from the host servers. The *client* and *host* subnets are connected by a single gateway that mimics the shared Internet connection of a typical academic network. All traffic between the two subnets must pass through this gateway device, which is an Anue network emulator [1]. The device allows us to monitor the traffic load and control the speed on the connection between the two subnets. We configure the subnets' maximum transfer speeds to 1 Gbit/s (128 MB/s) for the host subnet and 100 Mbit/s (12.5 MB/s) for the client subnet. This mimics the typical faster server connections and slower client connections observed in most academic environments. A diagram of the experimental setup is shown in Figure 4.

We develop three experimental setups for this environment. We vary the bandwidth of the gateway connection in the different setups to create various bottleneck scenarios. In the first setup, the gateway is configured to have ample bandwidth for all of the clients. In the second and third setups, the gateway's bandwidth is restricted so that clients must compete for access. The third setup adds an additional configuration that restricts large file transfers. We compare the performance of user transfer rates and network gateway load for all of these setups. We again examine the performance of the two types of users: normal user and large user. A normal user is only periodically retrieving small amounts of data, where as a large user is continually retrieving large data files via multiple parallel connections to remote resources.

2.2.1 First setup: Bottleneck at client

In the first experimental setup, we configure the gateway connection to have an available bandwidth of 1 Gbit/s (128 MB/s). Since there are only five clients, each using a maximum 100 Mbit/s (12.5 MB/s), the gateway is able to accommodate each client's traffic and is not a bottleneck. Each client has a maximum bandwidth of 12.5 MB/s, which is the bottleneck for the users' transfers.

We begin our evaluations by examining the performance of large user data transfer rates, as the number of large users transferring files on a single client machine increases. In this setup, only one client is generating traffic that utilizes the gateway connection. We incrementally add user transfers on the same client and observe the changes in transfer rate performance. As Figure 5 indicates, the performance of a user's transfer is negatively affected as other transfers are added. With each additional transfer, the transfer rate for each user decreases. The data transfers compete for the available bandwidth of the client. Figure 6 shows that the client's bandwidth was completely utilized with only two users. If a large user creates two parallel transfer connections, then everyone on the client is affected, including the large user. As the number of users increases, we notice a drop off in the total transfer rate for entire client, which indicates the opening too many large file transfers is detrimental to system performance. Network load at the gateway is 10%.

We continue our experiments by observing the performance of user file transfers when multiple large users on multiple client machines are retrieving data. We configure each client to have 10 users transferring large data

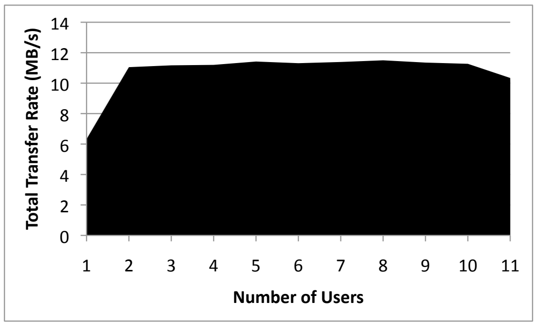

Figure 6: Total transfer rate for all data transfers on a single client.

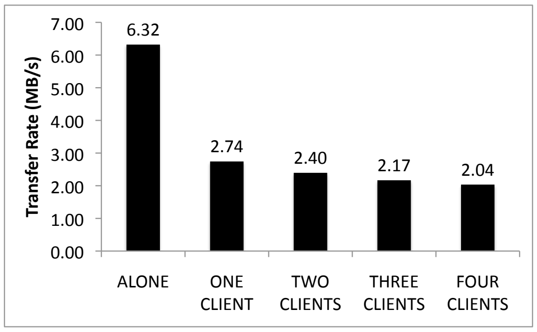

Figure 7: Individual, normal user's transfer rate changes as other clients retrieve large data files for multiple large users.

files. We begin with only one client receiving data and then work up to all five clients. Each client retrieves data from a unique host server. We find that each client achieves the same transfer rate for all of its transfers. Each machine utilizes its maximum bandwidth available, however the gateway connection can handle all five of the clients. The total transfer rate for a machine is not affected. Network load at the gateway increases by 10% for each additional server, with a maximum load of 50%.

We continue our evaluations of user data retrieval performance by creating an experiment that examines how large users' data retrievals affect a normal user. In this experiment, we isolate one of the five clients and configure it to only have a single normal user's data transfer. This single data transfer represents a typical user utilizing the network. We examine the performance of this user as the other four clients concurrently retrieve large data files for multiple large users. Figure 7 compares the normal user's transfer rates as other clients initiate their multiple large file transfers. We are find that this normal user's transfer performance greatly decreases when the other clients are retrieving large data files. This is surprising since the normal user's client has no other transfers. When just one client is actively retrieving large data files for multiple large users, the normal user is dramatically affected. The normal user's retrieval rate decreases by 57%. As the number of other clients increases, the normal user's performance continues to degrade. Even though the gateway connection can handle all of the clients' traffic, the shear number of packets transmitting through the connection limits the normal user's performance.

We find that even though the gateway connection is able to accommodate the bandwidth of all clients, the large data transfers have a negative impact on users' performance. This result is surprising since the gateway device has amble bandwidth for multiple clients. A normal user on one client machine is significantly impacted by the shear volume of data being transferred by other client machines.

2.2.2 Second setup: Bottleneck at gateway

In our second experimental setup, we limit the bandwidth of the gateway connection to 155 Mbit/s (19.44 MB/s). This creates a bottleneck at the gateway device, which simulates a heavily loaded public network, such as the academic network examined in the first experimental environment. Now each client will be competing for available bandwidth. This is typical of an average academic network, where the shared Internet connection must be shared amongst thousands of users. There is not enough bandwidth available to dedicate a share to each user.

We repeat the experiment where each client has 10 active large user transfers and we increase the number clients requesting data. Figure 8 shows the total transfer rates achieved by each client as the number of clients competing for bandwidth increases. The transfer rate for each machine decreases as the competition for the gateway connection increases. The graph also shows the total transfer rate achieved for all clients, which shows that multiple clients will completely utilize all of the available bandwidth at the gateway connection. The clients cannot utilize more bandwidth than is available to them.

We again examine how large users' data retrievals affect a normal user. In this experiment, we isolate one client that has only a single normal user requesting data. This single data transfer represents a typical user requesting and retrieving small amounts of data. We examine the performance of this normal user as the other four clients concurrently retrieve large data files for multiple large users. Now that the gateway connection is a bottleneck, the other clients affect the normal user even more. Figure 9 shows the change in trans-

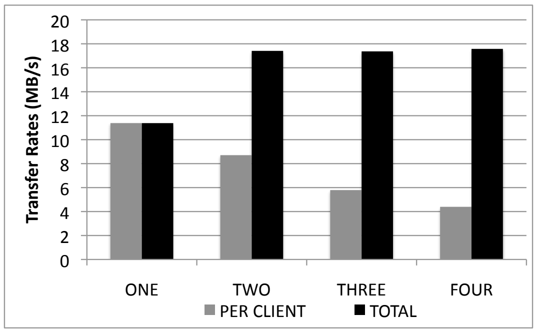

Figure 8: Comparison of client transfer rates as the number of clients increases (shown in grey, left). Also shown (in black, right) is total transfer rate achieved for all clients.

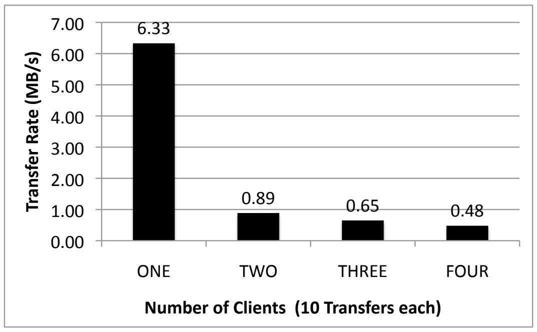

Figure 9: Normal user's transfer rate changes as the number of other clients retrieving large data files for multiple large users increases. (Limited gateway connection bandwidth)

fer rates for the normal user as the number of other large user client machines increases. With just one other client, the normal user's transfer rate decreased by 86%. As the number of other clients retrieving large files increases, the normal user is further affected by the increased contention at the gateway connection.

We find that with a bottleneck at the gateway connection, where the bandwidth to the host subnet is restricted to a fraction of the bandwidth of the *client* subnet, large data transfers have a significant impact on user performance. This corroborates our previous experiment and shows how large file transfers can significantly impact a normal user.

2.2.3 Third setup: Bottleneck at gateway with restricted large file transfers

In the third experimental setup, we continue to limit the gateway connection as in the second setup, which causes a bottleneck at the gateway device. This time we also impose transfer rate restrictions on the large users'

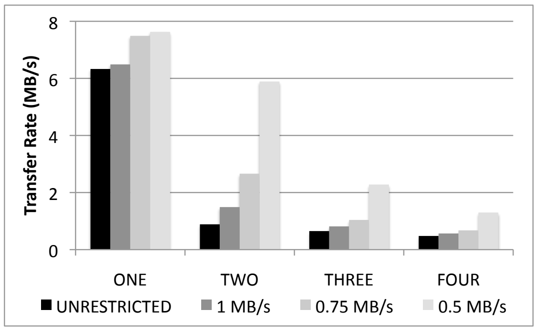

Figure 10: Compares the performance of a normal user attempting to retrieve data while a varying number of other clients are retrieving large data files for multiple large users. Varying levels of transfer rate restrictions are applied to the large users' data transfers.

data retrievals. We restrict the transfer rate that a large user can retrieve large data files. We vary this limit from 1 MB/s to 0.5 MB/s. When a large user retrieves data at a faster rate than specified, the transfer is forced to wait a certain amount of time before it can continue. Using these forced waits, we can guarantee that the average rate of the large file transfer matches the imposed restriction.

We again examine the transfer performance of a normal user when other clients are retrieving large data files for large users. This time the large users' transfers are restricted to specific rates. The normal user's transfer is not limited. Figure 10 illustrates the changes in the normal user's transfer rates when various retrieval rate restrictions are placed on the large file transfers of the large users. The greater the restriction the better performance that the normal user observes.

Figure 11 shows the changes in transfer rates for an average large file transfer when the different restrictions are in place. When there are one or two clients retrieving large files, there is a noticeable change in transfer rates depending on the degree of restriction. For three or more clients, the difference in performance between the various restrictions is minimal.

Large file transfer restrictions have a significant positive impact on the performance of the normal user's transfer, as shown in Figure 12. As the number of large file transfers increases, the normal user observes a greater increase in performance. When there are 40 simultaneous large users' file transfers being restricted fairly, the normal user saw 171% increase in transfer performance.

Restricting the large data transfers improves a normal user's performance, but is it detrimental to the large users attempting retrieving these large files? We examine the changes in transfer rates for the restricted

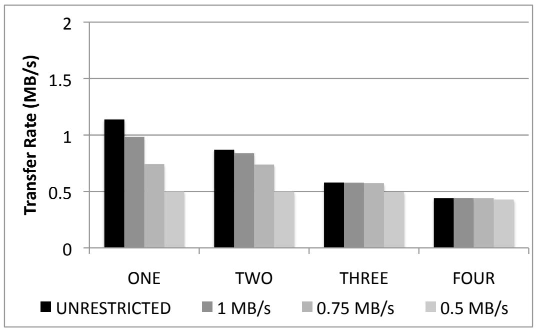

Figure 11: Comparing the individual large data transfer rates on the clients as varying levels of transfer speed restrictions are applied.

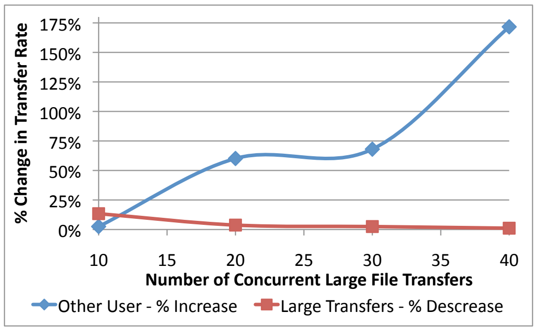

Figure 12: Changes in transfer rates for a single, normal user and for an average large user when large transfers are restricted fairly.

large data transfers in Figure 12. With a 0.5 MB/s restriction, there is only a 2.5% decrease in the transfer rate of a large data retrieval transfer when there are 40 large file transfers competing for the gateway connection. When retrieving a 1 TB file, the unrestricted configuration would take 11 hours and by imposing the 0.5 MB/s restriction the transfer would only take 19 minutes longer. By waiting a fraction of time longer, the restricted setup ensures improved performance for all users and maintains a stable and efficient network environment.

Restricting the large data transfers results in improved performance for normal users not retrieving large data files. When properly configured, the large users' data transfers are minimally impacted by the transfer restrictions.

3. Summary of findings

In our experiments, we find that system and user performance suffer as the number of users retrieving large files increases. All users are affected by the increased workload of large file retrievals. This is expected, however, the surprising result is the degree of the impact. The impact on normal users, which are only sporadically transferring small amounts of data using shared resources, is significant. We find that a normal user could potentially see a 86% degradation in transfer performance when other users are concurrently retrieving large files.

During our experiments, we observe three interesting findings. The first finding is that a single user on a real academic network can significantly overload network resources and impact all users' performance. One would assume that a user's bandwidth would be limited or controlled in an academic network to ensure that the system remains stable. We find that this concurrent data retrieval affects the entire local subnet, as well as all users utilizing the shared Internet connections. Our experimental workload increases the local network utilization from a normal of 30 to 90+%. These high utilizations cause dropped packets and latency issues for all users on the network.

Our second interesting finding is that even with ample bandwidth available for all users, large file transfers still have a negative impact on system performance and cause other users to have degraded performance. Depending on the number of concurrent large file transfers, a typical user's retrieval rate could decrease by as much as 57 to 68%. As the number large file transfers increases, normal users would see their performance continue to degrade due to the increase in traffic and contention for the shared Internet connection.

The third interesting finding is that by placing fair retrieval rate restrictions on large file transfers we are able to improve the performance of other users, as well as maintain adequate service for the users retrieving large datasets. By imposing these restrictions, we find that a typical user retrieving small amounts of data could see a significant improvement (+170% in some instances) in transfer rates. When restricted properly, the large data transfers only see a minor impact on their performance.

Overall, we find that there is significant impact to local system performance when users retrieve large data files over shared, public networks. All resources in the system experience increased traffic and heavier workloads. The increased demand affects the performance of all users in the system. Normal users are unjustly penalized and observe decreased transfer times and longer service times. By restricting large file transfers however, the system can maintain adequate performance for all users.

4. Conclusions

Large amounts of data are stored in clouds around the world and as clouds grow, their datasets will only continue to multiply. The users retrieving these large data files from the cloud are often on public networks, using a shared connection to the Internet. When users retrieve large files over these shared resources, everyone is affected. As the number of users retrieving data increases, the impact on the performance of the entire system multiplies. The impact of large file transfers is significant, as these transfers increase traffic and workload at all levels of the system. Congestion and contention for shared resources will multiply as the workload and demand on these resources increases. As the load grows, eventually there will be packet loss and failures. Transfer performance for all users will degrade as the demand rises.

Recent studies suggest that opening as many concurrent transfers as possible will result the best service times for a user. This is representative of the *hare* mentality: retrieve data as fast and hard as possible. These studies neglect to examine the impact that their greedy techniques have on the performance of all users and the local networks. By attempting to utilize as much bandwidth as possible, a greedy technique not only hinders the performance of all users on the network, but also negatively impacts the greedy, large user. In our live experiments, we find that system and user performance suffer as the number of users retrieving large files in this manner increases. All users are affected by the enlarged workload of large file retrievals.

We find that the opposite approach is more efficient for all users, which is counter-intuitive. We discover that restricting the large file transfers to a specified rate results in superior performance for all users, including the users retrieving the large files. This is representative of the *tortoise* mentality. Since users are transferring extremely large data files, the transfers will have long durations taking tens of hours to several days. We find that this slow and steady approach does indeed "win the race."

Data files are going to continue to grow in size and the demand for these files will increase proportionally. Network administrators will need to consider how to properly handle these large file transfers and minimize the impact that they have on system and users' performance. For future work, we intend to further study fair, dynamic restrictions on large file transfers. By dynamically imposing appropriate transfer restrictions, we can maintain a balanced and sustainable system.

References

[1] Anue systems. *http://www.anuesystems.com*.

[2] Cloud computing: An overview. *Queue*, 7(5):3–4, 2009. Advisor-Creeger, Mache.

[3] W. Allcock and et. al. The globus striped gridftp framework and server. In *Supercomputing*, page 54, 2005.

[4] M. Armbrust and et al. A view of cloud computing. *Commun. ACM*, 53(4):50–58, 2010.

[5] R. S. Bhuvaneswaran and et al. Redundant parallel data transfer schemes for the grid environment. In *ACSW Frontiers*, pages 71–78, 2006.

[6] J. Bresnahan and et. al. Globus gridftp: What's new in 2007. In *GridNets*, October 2007.

[7] R.-S. Chang, M.-H. Guo, and H.-C. Lin. A multiple parallel download scheme with server throughput and client bandwidth considerations for data grids. *Future Generation Computer Systems*, 24(8):798–805, 2008.

[8] J. Feng and M. Humphrey. Eliminating replica selection - using multiple replicas to accelerate data transfer on grids. In *ICPADS*, page 359, 2004.

[9] I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE '08*, pages 1 –10, 12-16 2008.

[10] B. Hayes. Cloud computing. *Commun. ACM*, 51(7):9–11, 2008.

[11] H. Liu and D. Orban. Gridbatch: Cloud computing for large-scale data-intensive batch applications. In *Cluster Computing and the Grid, 2008. CCGRID '08. 8th IEEE International Symposium on*, pages 295 –305, 19-22 2008.

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

[13] C. Nicholson and et. al. Dynamic data replication in lcg 2008. In *UK e-Science All Hands Conference*, Nottingham, September 2006.

[14] S. Vazhkudai. Enabling the co-allocation of grid data transfers. In *GRID '03: Proceedings of the Fourth International Workshop on Grid Computing*, page 44, 2003.

[15] S. Vazhkudai. Distributed downloads of bulk, replicated grid data. In *Journal of Grid Computing*, volume 2, pages 31–42, March 2004.

[16] C.-M. Wang, C.-C. Hsu, H.-M. Chen, and J.-J. Wu. Efficient multi-source data transfer in data grids. In *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, pages 421–424, 2006.

[17] A. Weiss. Computing in the clouds. *netWorker*, 11(4):16–25, 2007.

[18] C.-T. Yang, Y.-C. Chi, and C.-P. Fu. Redundant parallel file transfer with anticipative adjustment mechanism in data grids. In *Journal of Information Technology and Applications*, volume Vol. 1, pages 305–313, March 2007.

[19] X. Zhou, E. Kim, J. W. Kim, and H. Y. Yeom. Recon: A fast and reliable replica retrieval service for the data grid. In *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*, pages 446–453, 2006.