It takes know-how to retrieve large files over public networks

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

Abstract

Retrieving large data files across public networks is a complicated task. This paper presents the issues and challenges faced by an average user when trying to retrieve large datasets using public, shared resources. Several data retrieval techniques are evaluated from a user's perspective and it is found that users will have drastic differences in performance based on how they select and utilize various servers to complete their requests. Without know-how, a user could potentially spend 20 to 100 times longer on data retrieval than necessary depending on the servers chosen and the retrieval technique utilized. Implementing data retrieval techniques and configuring transfers can be difficult and time-consuming for the average user. There is a significant amount of know-how that is required for a user to successfully and efficiently utilize these retrieval techniques.

1. Introduction

Research communities are creating staggering amounts of data that need to be accessible to users around the world. 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 [6, 8]. Geographically dispersed researchers eagerly await access to the newest datasets. The task of retrieving this data quickly and efficiently is a major undertaking for any user.

In order to improve users' access to large and high-demand datasets, system administrators utilize replication, which reduces access latency and bandwidth consumption [5]. Replication helps to balance load and can improve availability by creating multiple copies of the same data [12]. These copies are distributed amongst servers spread out around the world, allowing users multiple possible access points. For example, CERN's replication strategy for the LHC experimental data utilizes a tiered replica structure. Raw data obtained from their instruments is immediately replicated in a controlled fashion to multiple storage sites in various tiers [8]. End users then have the ability to access these replicas that are distributed around the world.

The datasets being replicated in these distributed systems are extremely large. They can range from multiple terabytes to several petabytes in size. Retrieving portions of these large datasets is a time consuming task that will result in long duration transfers taking tens of hours to several days. When it comes to transfers of this magnitude, a simple "click and wait" methodology is not going to suffice. During the transfer period, any number of situations can occur forcing the transfer to slow down or stop completely. Servers can go offline or become overloaded. Network conditions for the user or the server could also degrade.

Users attempting to retrieve these large scientific datasets are generally using public networks on academic campuses or in research institutions. Even though they possibly have private storage and computation resources, they must utilize a shared connection to the Internet. In a university environment, several thousand users might share this connection and a single user will be limited to only a portion of the bandwidth available to everyone. The conditions of the network can also vary greatly during different times of the day and different months of the academic year. There is no way to guarantee the network conditions at any given time. Due to these types of situations, advanced data retrieval techniques are available to users.

These advanced retrieval techniques allow a user to simultaneously use multiple data sources concurrently. The user is not reliant on one server connection for the entire transfer. A user could retrieve half of a file from one server and the remaining portion from another server at the same time. The number of servers utilized in parallel depends on the algorithm for each technique.

In our experiments, we examine a single user retrieving a large data file over a public network, using a shared

Internet connection. We evaluate several different techniques that a user could potentially utilize to retrieve the data file. We observe their performance, as well as the difficulties that an average user faces when implementing and using these techniques.

We find that users could have drastic differences in performance based on how they select and utilize various servers to complete their requests. Depending on the retrieval technique used and the number of servers involved, a user could find varying performance from 9.2 minutes to 19 hours when trying to retrieve a single 30GB data file. Without know-how, a user could be forced to unnecessarily wait for a data transfer to complete.

During our experiments, we observe the many difficulties and complications that an average user would experience when trying to retrieve these datasets. Implementing and configuring data transfers using these data retrieval techniques can be difficult and time-consuming for the average user. There is a significant amount of know-how that is required for a user to successfully and efficiently utilize these retrieval techniques.

The paper is organized as follows. Our experiments and observations of data retrieval from a user's perspective are detailed in Section 2. Issues and challenges that the average user faces during large file retrieval are presented in Section 3. We then discuss our conclusions in Section 4.

2. Experiments and Observations

In our experiments we observe the process of a single user retrieving a large data file over a public network, using a shared Internet connection. We examine several different techniques that a user could potentially utilize to retrieve the data file. We evaluate their performance, as well as the difficulties that an average user faces when implementing and using these techniques.

Average users have limited capacity for data retrieval, which is governed by their network connection and their Internet service provider. A user may utilize a shared Internet connection, such as an academic campus network. The Internet connection for the entire network is fast, however all of the users on the network are sharing this resource. In our experimental setup, the user's computer is located on an academic campus network and uses a shared high-speed Internet connection.

Each end user (10,000+) shares the multiple high-speed Internet connections servicing the network. Network workload conditions vary throughout the day, as end users share the public resources. Figure 1 illustrates the variations in the user's transfer rate when retrieving a 1MB file from a remote server over the course of several weeks. Since the traffic on local and wide area networks

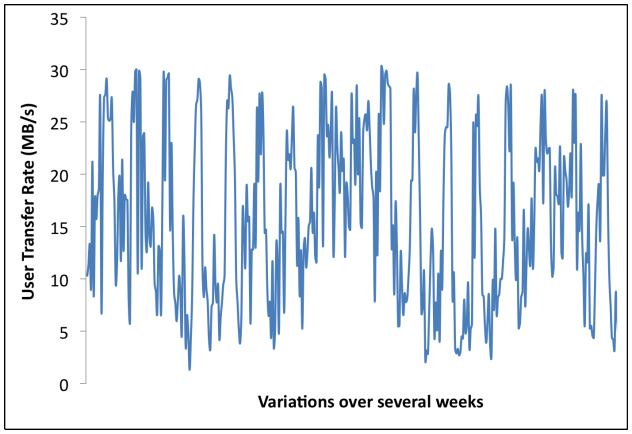

Figure 1: Variations in user transfer rates when retrieving a 1MB file from a remote server over the course of several weeks.

can vary, as well as server workloads, we repeat our experiments several times over the course of three months. We present the average values for all data transfers.

We examine the performance of retrieving a 30GB data file over public networks, using a shared Internet connection. The data file being retrieved is replicated on thirty different servers located around the world. These servers are public servers not under our control and are concurrently servicing other users' requests. The user has the ability to retrieve the file from any of these servers.

2.1 Normal Data Retrieval

Normally, the user is faced with the decision of choosing a server from a listing of available servers to service their request. The user has no knowledge about the potential performance of any given server. In our experiments, we retrieve the data file from each server independently in order to observe the differences in service times that a user would experience. We begin our experiments by retrieving the desired 30GB data file from each one of the available 30 servers independently. We find that the data retrieval performance for each server varies greatly. Figure 2 illustrates the marked differences in the service times for each server. The fastest file transfer occurred in 11.7 minutes, while the slowest file transfer took over 19 hours. The median service time for all servers is 75.5 minutes.

During the transfers, the user's network utilization is monitored. We find that the user's retrieval capacity was not fully utilized during any of the transfers and was especially low for the transfers with the longest service times. This indicates that the bottleneck of the longest transfers lies with either the connection between

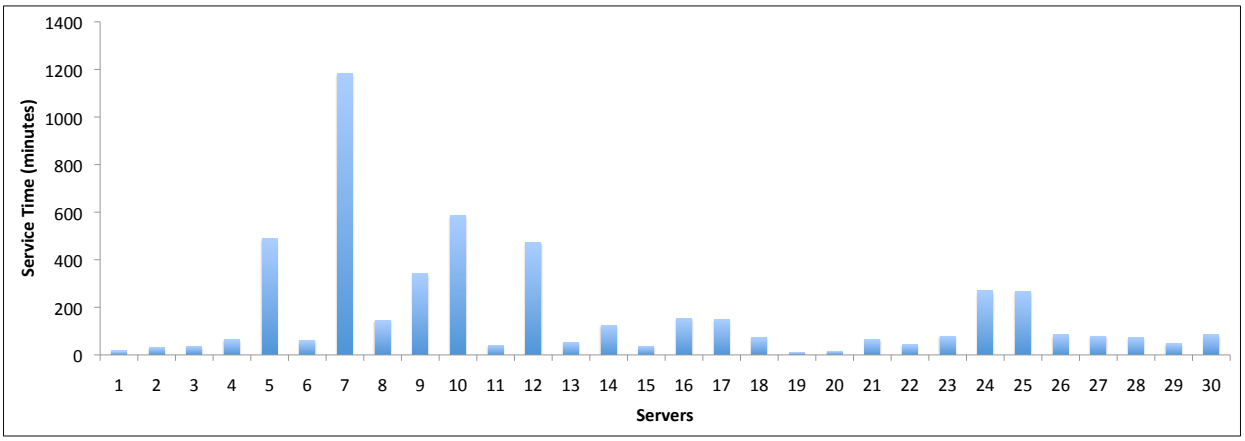

Figure 2: Normal Data Retrieval: Service times (minutes) for each server when retrieving the entire 30GB data file independently.

the server and the user or with the server itself.

When retrieving data files, the replica selected can greatly impact a user's performance. The major difficulty for the average user is knowing how to select an appropriate replica. Choosing a lightly loaded server over a heavily loaded server can result in dramatically different completion times for a user. Finding the most efficient replica is a difficult and complicated task. There are many studies [2, 7, 9, 10, 11, 15] that explore different mechanisms for efficient replica selection. All of these mechanisms require the user to implement and configure selection algorithms, which can beyond the skill set of an average user.

2.2 Advanced Data Retrieval

Instead of relying on a single server for the entire data transfer, a user could potentially use multiple servers at the same to transfer the desired data. Many recent studies explore advanced techniques for data retrieval known as distributed file retrieval (or data co-allocation), which allow a single user to simultaneously utilize multiple resources to service a request. Using data co-allocation, users can utilize many or all of the available replicas. The users would issue requests for portions of data file from these replicas. The requests would then be serviced in parallel. The longest service time that any user would experience would be determined by the slowest replica to service any one of the partial data requests.

There are several different types of data co-allocation retrieval techniques. They can be grouped based on how they utilize the available replica severs. We examine the three most common groups of data co-allocation techniques: brute-force, performance-based, and dynamic. In the following sections, we examine the performance differences and user difficulties that we observe for these techniques when used in our experimental setup.

2.2.1 Brute-force Technique

The basic, **brute-force**, data co-allocation technique [13] issues a request for equal sized portions of the file from all available replicas. Every replica that contains the file is utilized and each is responsible for servicing an equal amount of data. There is no consideration given to the performance of replica servers or network conditions. The workload at all servers is increased equally for each co-allocating user.

We evaluate the brute-force technique (BFT) by dividing our file request into 30 equal-sized portions and requesting one portion from each of the 30 replica servers. The requests are serviced concurrently and the data is retrieved from each server in parallel. Since the entire file request is not complete until all of the portions are retrieved, the performance of the request is dependent on the slowest file transfer. Similar to our normal data retrieval observations, we find that the performance of each of the transfers varies greatly. Figure 3 illustrates the differences in the service times for each of the individual file portion retrievals. As with normal data retrieval, server 7 provides the longest service time. The fastest file retrieval finishes in 2.5 minutes and the slowest file retrieval takes 76.8 minutes. Since the data retrieval is not complete until all portions are retrieved, the service time for the entire data file transfer using BFT is 76.8 minutes.

In comparison to our normal data retrieval experiments, the brute-force technique provides improvement over normal data retrieval for some of the servers. The average service time for BFT is almost equal to the median service time for the single server technique. This indicates that the BFT provides improved performance

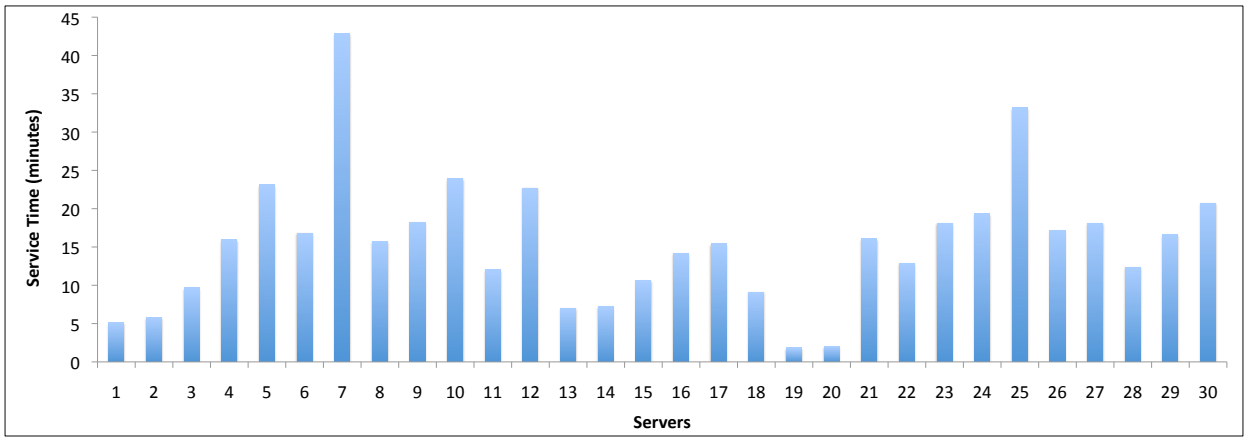

Figure 3: Advanced Data Retrieval - Brute Force Technique: Service times (minutes) for each server when retrieving equal 1GB portions of the 30GB data file.

in comparison to retrieving the file from a single server for 15 of the available 30 servers. Since the BFT technique utilizes all servers regardless of their retrieval capacity, the slower servers will always hinder the performance of the entire data transfer.

There are several difficulties that an average user would face when using the brute-force technique. Initiating and monitoring multiple transfers can be difficult. In our experiments, we utilized 30 concurrent transfers, which proved to be complicated to track. With multiple simultaneous transfers, the task of setting up and monitoring the individual transfers can be overly complex for the average user.

2.2.2 Performance-based Technique

Performance based techniques utilize performance metrics when selecting replicas to utilize. There are two main groups of performance-based techniques: history-based and probe-based. Both of these groups attempt to exploit faster servers by assigning them greater portions of the workload. Depending on the user's choice, the number of servers utilized in parallel can vary from two to possibly all of the available servers.

In **history-based techniques** [13, 14], the retrieval algorithms address the fact that each transfer between a replica and the client has varying transfer rates. These techniques adjust the amount of data retrieved from each replica by predicting the expected transfer rate for each replica. The algorithms create forecasts of future performance based on transfer history with network and disk load data. Historically faster servers are assigned to deliver larger portions of the file and slower servers are assigned smaller pieces.

In **probe-based techniques** [4, 19], the retrieval algorithms utilize network status information to create network throughput predictions. Some of these techniques utilize the Network Weather Service [17], which is a networking monitoring tool that utilizes sensors which gather data on the latency and bandwidth of end-to-end TCP/IP performance. Using these throughput forecasts for each replica server, the algorithms assign portions of data request to each available replica. Replicas predicted to have the best performance are assigned a larger portion of the request workload.

We evaluate a performance-based technique (PBT) by selecting servers using the round-trip time from a network ping and performance information from the transfers that we observed when examining the brute-force technique. We select servers with the lowest ping times and the shortest historical service times first. We vary the number of servers that are used concurrently from two to twenty. As the number of servers utilized increases, we are using slower servers with larger round-trip times and longer historical service times. We compare the overall service times that we experience for the varying number of concurrently utilized servers in Figure 4. We find that as the number of servers increases, the overall service time also increases. When more servers are used, slower servers are required to service portions of the request. The request is not complete until the slow servers finish their portions and thus affect the overall service time. When only the two servers with the best metrics were utilized, the overall time to retrieve the file was the least.

We also observe that as more file transfers are added, the user's available retrieval capacity diminishes. Eventually, there are more file transfers than the user's connection can handle and the transfers will compete for the available retrieval capacity. This can negatively affect faster transfers. Figure 5 illustrates the effects of multiple parallel file retrievals on the transfer rate of the

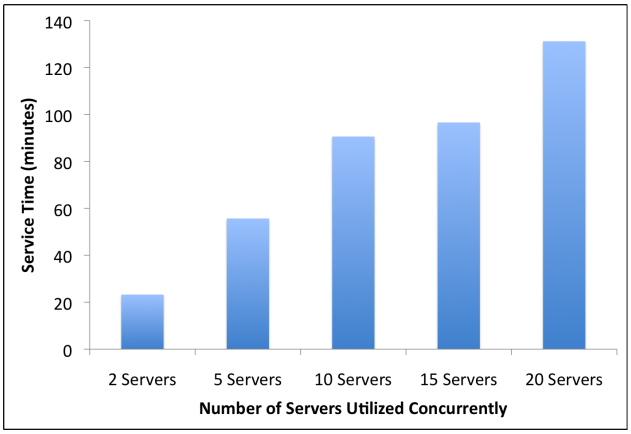

Figure 4: Advanced Data Retrieval - Performance-based Technique: Total service time (minutes) for the file transfer as the number of servers concurrently used increases.

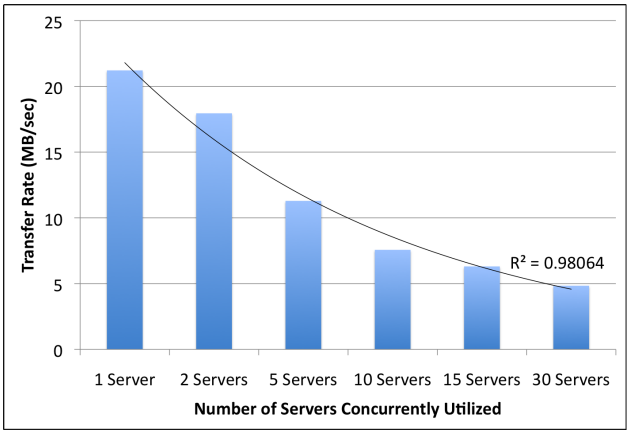

Figure 5: Transfer rates for the fastest server connection observed, as the number of servers concurrently used increases.

fastest connection observed. As the number of concurrent data retrievals increase, there is a decrease in the transfer rate for the fastest file transfer. There is a 77% decrease in the transfer rate when there are 29 other transfers competing for available retrieval capacity.

While probe-based techniques provide improved performance over brute-force techniques, they still attempt to utilize as many servers as necessary without regard for the user's limited retrieval capacity. In many cases these techniques create more transfers than the user's bandwidth can accommodate, which results in transfers competing for bandwidth. This situation diminishes the performance of the overall file transfer.

Users are faced with several difficulties when utilizing performance-based retrieval techniques. Since these techniques are more complex than the brute-force technique, their implementation could prove difficult for the average user. Another issue that a user faces is the problem of stale performance metrics. Network conditions and server workloads are constantly changing, which means that these metric values can quickly become inaccurate. Since these data transfers could potentially take multiple hours to several days, users will need to continuously update their performance metrics for all server connections.

2.2.3 Dynamic Technique

Dynamic techniques [1, 3, 16, 18] attempt to automatically adapt to changing system conditions by requesting small, equally sized, portions of a file from multiple replicas. In many dynamic techniques, each replica is initially assigned one segment. As replicas complete their assigned segments, they are assigned additional portions of the data file to service. Each dynamic technique uses different decision making algorithms on how to schedule these requests, however faster servers will end up transferring larger portions of the file. Any failed or undelivered requests can be automatically rescheduled to other replica servers, potentially created duplicate work. Depending on the specific dynamic technique, the desired data file could be segmented so that a single server could receive tens to hundreds of requests for portions of one file.

One dynamic technique attempts to fully utilize the user's retrieval capacity by dynamically and incrementally increasing the number of servers currently utilized until the user's maximum retrieval capacity is reached. This technique attempts to avoid situations where several transfers are fighting for available bandwidth.

The dynamic technique begins by selecting one server with the smallest round-trip time using a network ping. After the transfer has started, the user's available bandwidth is monitored. The technique then incrementally creates additional data transfers to other servers, as necessary until the user's retrieval capacity is fully utilized. Figure 6 illustrates the decrease in service time for the incremental technique as we approach full utilization of the user's retrieval capacity. The service time for this technique was 9.2 minutes, which was less than the fastest time observed using normal data retrieval.

In comparison to the other techniques, the dynamic technique produces the smallest service time for retrieving the 30GB data file. Figure 7 shows the difference in service times for all of the techniques that we observe. In addition to providing the smallest service time the user, the dynamic technique attempts to involve the smallest number of replica servers and attempts to fully utilize the user's retrieval capacity.

The user difficulties associated with dynamic tech-

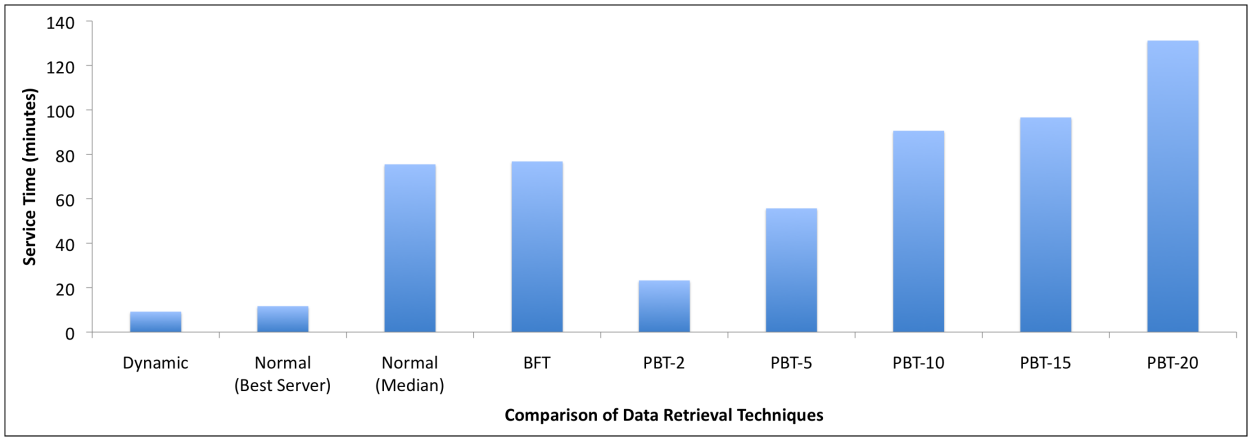

Figure 7: Comparison of service times (minutes) for all data retrieval techniques observed.

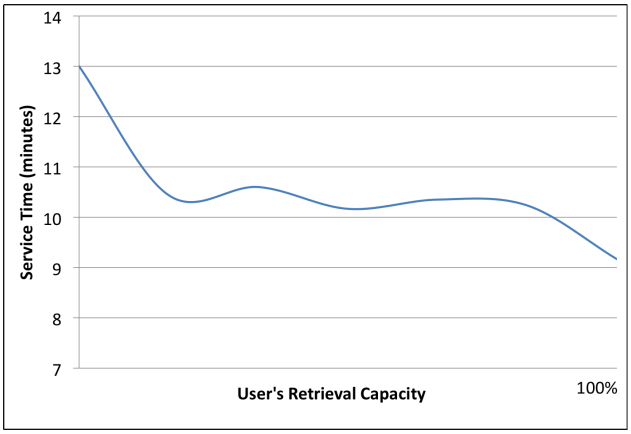

Figure 6: Incremental Distributed File Retrieval: Changes in service time (minutes) as the user's retrieval capacity approaches its maximum utilization.

niques are numerous. Many of these techniques are quite complicated and their algorithms are complex. Implementing them for automated use would require significant time for even an experienced programmer. The average user would find this task to be insurmountable. In addition, there are many aspects of these algorithms that are left for the user to decide and control. We detail some of these issues and challenges in the next section.

In summary, we find that advanced file retrieval provides improved performance in comparison to normal data retrieval. There are several advanced techniques available for the user to utilize. Choosing an appropriate and viable technique however is a difficult task. Implement, configuring and utilizing these techniques can be a challenge for even for the experienced user.

3. Issues and Challenges

Research communities are generating massive amounts of data that need to be readily accessible to thousands of users around the world. These users require fast and efficient access to data files.

Using public networks and shared resources to retrieve large amounts of data is a daunting task for any user. It requires significant know-how for a user to configure and maintain the data transfers needed to transport these massive data sets.

Retrieving large data files (GB, TB, PB) is a complicated and time-consuming process. These long duration transfers could take tens of hours to several days and a normal "one click and wait" method will not suffice. During the course of the transfer, servers may go off-line and network conditions may change that either hinder or stop the transfer completely. The user needs to know how to maintain the data transmission until completion.

Advanced retrieval techniques allow users to utilize multiple resources simultaneously. A user could utilize some or all of the available replicated copies of the data set. Depending on the specific advanced technique, these replicas can be selected and used to varying extents during the course of the data transfer.

In our evaluations, we transfer a large data file over a public, shared academic network from servers around the world. We find that users could have drastic differences in performance based on how they select and utilize various servers to complete their requests. Using normal data retrieval, where the user selects and relies on a single server to complete the entire request, a user could have a wide range of potential service times from 11.7 minutes to 19 hours depending on the replica server selected. Using advanced retrieval techniques, users can potentially reduce these service times. The brute-force technique utilizes all available replicas and produces a

service time of 76.8 minutes, which is almost equal to the median service time of normal data retrieval. The probe-based technique using only two servers with the best probe times yields a service time of 23.3 minutes, a significant improvement. The dynamic technique that minimizes the number of servers involved and fully utilizes the user's retrieval capacity provides a service time of 9.2 minutes, which is best observed time for all techniques.

These advanced techniques provide improved performance for users, however they are quite complicated to implement and use. They require significant user involvement and require multiple user decisions that can dramatically affect the performance of the transfer. A user needs know-how in order to make these techniques function properly and efficiently.

There are several complicated features that are left to users to control when configuring these transfers. The first option that the user must control is replica selection. As we've discussed, many distributed systems utilize replication to create multiple copies of data and place them on various servers throughout the system. The user can retrieve data from any of these replicas. Choosing which replica(s) to use is left to the user and can dramatically affect their performance, as we've shown in our experiments. Determining which replica in this "best" at a given time is not a trivial task.

Another configuration option that is frequently left for the user to determine is segment size. In some advanced techniques, the data file is divided into small portions called segments. The segment size is often left for the user to decide and the size chosen can affect the performance of the transfer. Determining the appropriate segment size is not a simple task. If the size is too small, a server may receive hundreds to thousands of requests for portions of a single file. This will result in longer disk service times at a server, as the number of users increases. A server's storage system can best service requests if it has greater knowledge of a user's workload. It can better schedule reading from the hard disks, as well as take advantage of pre-fetching and caching strategies.

A key issue that is not adequately addressed for retrieval techniques is failures. Since we are transferring extremely large data files over long periods of time, we will eventually encounter transfer failures. Many advanced techniques identify that failures can occur and provide mechanisms for issuing new requests, however specific details about the timings of these actions are not addressed and are left for the user to decide. The request re-issue delay is a common problem with these techniques. Determining the appropriate amount of time that the application should wait before issuing a replacement request is a non-trivial task.

4. Conclusions

Overall, setting up and configuring data transfers using advanced data retrieval techniques can be difficult and time-consuming for the average user. There is a significant amount of know-how that is required for a user to successfully and efficiently retrieve large data sets. Ideally, these users should be able to devote their time and attention to data analysis instead of focusing on data transfer. The entire retrieval process needs to be simpler, easier and more efficient for the average user.

We see a clear need for an automated data retrieval tool that allows the user to easily select and retrieve large data files with minimal configuration and monitoring. We want to reduce the number of decisions that a user has to make when starting a data transfer. The retrieval tool would utilize a responsible and efficient data retrieval technique that attempts to minimize the number of servers involved while still providing reliable service.

We also see the need to consider other performance factors besides user service times and data throughput. Retrieving data as quickly as possible for the user without regard to overall system performance can be detrimental to all users. Transferring extremely large files from shared servers over common networks can create a massive burden on system resources. The burden is distributed along the entire data path from the replica servers to the users. All users utilizing these shared resources will be adversely affected. We need to keep in my mind the state of the overall system and be considerate of other users when retrieving data.

References

[1] R. S. Bhuvaneswaran, Y. Katayama, and N. Takahashi. Redundant parallel data transfer schemes for the grid environment. In *ACSW Frontiers '06: Proceedings of the 2006 Australasian workshops on Grid computing and e-research*, pages 71–78, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[2] 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.

[3] 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.

[4] J. Feng and M. Humphrey. Eliminating replica selection - using multiple replicas to accelerate data transfer on grids. In *ICPADS '04: Proceedings of the Tenth International Conference on Parallel and Distributed Systems*, page 359, 2004.

[5] 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.

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

[7] M. Mitzenmacher. The power of two choices in randomized load balancing. *Parallel and Distributed Systems, IEEE Transactions on*, 12(10):1094–1104, Oct 2001.

[8] C. Nicholson, D. G. Cameron, A. T. Doyle, A. P. Millar, and K. Stockinger. Dynamic data replication in lcg 2008. In *UK e-Science All Hands Conference*, Nottingham, September 2006.

[9] M. Rabinovich and O. Spatscheck. *Web Caching and Replication*. Addison-Wesley, 2002.

[10] R. M. Rahman, R. Alhajj, and K. Barker. Replica selection strategies in data grid. *Journal of Parallel and Distributed Computing*, 2008.

[11] 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.

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

[13] 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.

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

[15] 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.

[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] R. Wolski, N. T. Spring, and J. Hayes. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Gener. Comput. Syst.*, 15(5-6):757–768, 1999.

[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.