OneDataShare:
A Vision for Cloud-hosted Data Transfer Scheduling and Optimization as a Service

Asif Imran, Md S Q Zulkar Nine, Kemal Guner, and Tevfik Kosar
Department of Computer Science and Engineering,
University at Buffalo, State University of New York, Buffalo, NY 14260
{asifimra, mdsqzulk, kemalgne, tkosar}@buffalo.edu

Keywords: Managed file transfer, cloud computing, throughput optimization, protocol translation, data management.

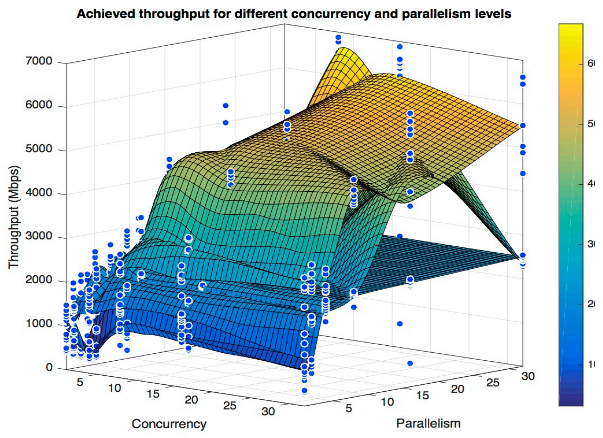
Abstract: Fast, reliable, and efficient data transfer across wide-area networks is a predominant bottleneck for data-intensive cloud applications. This paper introduces *OneDataShare*, which is designed to eliminate the issues plaguing effective cloud-based data transfers of varying file sizes and across incompatible transfer end-points. The vision of *OneDataShare* is to achieve high-speed data transfer, interoperability between multiple transfer protocols, and accurate estimation of delivery time for advance planning, thereby maximizing user-profit through improved and faster data analysis for business intelligence. The paper elaborates on the desirable features of *OneDataShare* as a cloud-hosted data transfer scheduling and optimization service, and how it is aligned with the vision of harnessing the power of the cloud and distributed computing. Experimental evaluation and comparison with existing real-life file transfer services show that the transfer throughout achieved by *OneDataShare* is up to 6.5 times greater compared to other approaches.

1 INTRODUCTION

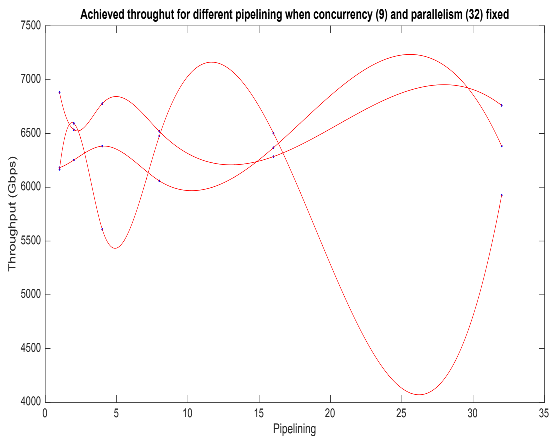
Cloud services have gained tremendous popularity amongst the general public for its low cost, high availability, and elasticity. Millions of data files of varying sizes and formats are transferred from one storage to another every day using the cloud, irrespective of their locations. However, the benefits of cloud computing cannot be fully utilized due to the limitations in data transfer mechanisms, which have become a main bottleneck restraining full utilization of the cloud's strength. For example, transfer of a 1 TB dataset over to a cloud storage may take several weeks despite the high-speed networks available (Garfienkel, 2007). For this reason, many IT companies and academic research labs prefer sending their data through a shipment service provider such as UPS or FedEx rather than using wide-area networks (Cho and Gupta, 2011). Some cloud storage providers (e.g., Amazon S3) even provide import/export services in which the users can ship multiple hard drives via FedEx to the storage provider, and the provider copies data to the cloud storage directly (Zhao et al., 2013).

As data has become more abundant and data resources become more heterogeneous, accessing, sharing and disseminating these data sets become a bigger challenge. Using simple tools to remotely logon to computers and manually transfer data sets between sites is no longer feasible. Managed file transfer (MFT) services such as Globus Online (Chard et al., 2017), PhEDEx (Egeland et al., 2010), Mover.IO (Mover.io, 2017), B2SHARE (A. et al., 2015), and RSSBUS (RSSBUS, 2017) have allowed users to do more, but these services still rely on the users providing specific details to control this process, and they suffer from shortcomings including low transfer throughput, inflexibility, restricted protocol support, and poor scalability. Table 1 shows a comparative analysis of different data transfer and scheduling services available today.

Transferring large datasets with heterogeneous file sizes (i.e., small and large files mixed) causes inefficient utilization of the available network bandwidth. Small file transfers may cause the underlying transfer protocol not reaching the full network utilization due to short-duration transfers and connection start up/tear down overhead; and large file transfers may suffer from protocol inefficiency and end-system limitations. Application-level TCP tuning parameters such as buffer size, pipelining, parallelism, and concurrency are effective in removing these bottlenecks, mainly when used together and in correct combinations. These tunable transfer parameters play a significant role in improving the achievable transfer throughput as shown in Figure 1. Here we can



(a)



(b)

Figure 1: This figure shows the variation in achievable throughput in 10 Gbps XSEDE network between Stampede (TACC) and Gordon (SDSC). Cubic spline surface is constructed to interpolate throughput for the whole parameter space. (a) Shows the impact of concurrency and parallelism parameters on the throughput. The colors represent the throughput in Mbps, blue being the lowest and yellow being the highest throughput. The blue dots represent actual measured throughput values for the corresponding parameters. (b) The impact of pipelining on throughput is demonstrated.

see, same transfer can achieve different throughputs for different parameter values. Figure 1(a) shows the joint effect of concurrency and parallelism over a single transfer, where Figure 1(b) shows the effect of pipelining. However, setting the optimal levels for these parameters is a challenging task and an open research problem. Poorly-tuned parameters can either cause underutilization of the available network bandwidth or overburden the network links and degrade the performance of the end-to-end data transfer due to increased packet loss, end-system overhead, and other factors.

In this paper, we present the vision of *OneDataShare* as a cloud-hosted data transfer scheduling and optimization service. OneDataShare has three major goals: *(i)* optimization of end-to-end data transfers and reduction of the time to deliver the data; *(ii)* interoperation across heterogeneous data resources (both streaming and at-rest) and on-the-fly inter-protocol translation; and *(iii)* predicting the data transfer time and decreasing the uncertainty in real-time decision-making processes. OneDataShare's data transfer optimization, interoperability, and prediction services are being implemented completely at the application-level, not requiring any changes to the existing infrastructure nor to the low-level networking stack, although drastically increasing the end-to-end performance of data transfers and data-intensive applications depending on data transfer. Experiments demonstrated that *OneDataShare* achieved throughput increase of up to 6.5 times compared to existing services.

2 RELATED WORK

The motivation for a platform-independent cloud-to-cloud data transfer system has been specified in (IBM, 2017b). Important gridlocks like local-area storage bottleneck, WAN transport bottleneck, session restore limitation and slow throughput of data object transfers are limiting the data transfer performance. Hence, Aspera (IBM, 2017a) urged for development of a fundamental system that will cater to these issues. They identified some of the important characteristics which a cloud based data object transfer mechanism need to exhibit. Since the majority of cloud based storage are object storage, separation of file data and metadata together with replication across commodity storage are key to the success of effective data transfer (IBM, 2017b). Also, there is a need to address the challenges faced when data objects are transferred across different continents via provisioned cloud instances.

Rclone is developed as a data transfer solution which is installed at the client end and accessed through the command line interface (RClone, 2017). It contains an array of commands which can be used to transfer data using a variety of protocols and APIs like SFTP, Dropbox, and Amazon S3. Although the application can conduct data transfers at a faster rate, however it requires extensive configuration procedures before it is ready to be used. Cloud based data transfer via GUI which shows the directories of remote machines in one dashboard is achieved by CloudFuze (CloudFuze, 2016). The platform uses FTP for data transfer and implements an interface in

General Overview			Optimization information				
Name	Source	Cloud	Parameters	Fsize	Interface	Protocols	Metadata
Globus	Open	Cloud	Parallelism, Pipelining, Concurrency	All	GUI	GridFTP, FTP	Yes
PhEdEX	Open	Client	Compression	All	GUI	FTP, SCP, SFTP	Yes
Waarp	Open	Client	None	All	Cmd line	FTP	No
DivConq	Open	Cloud	Compression	All	Cmd line	HTTP(S), SFTP	No
fileXhub	Open	Cloud	Distributed decoupling	Large	Cmd line	FTP, HTTP, SFTP	No
LDR	Open	Client	None	Medium	Cmd line	HTTP, FTP	No
Mover.io	Closed	Cloud	Compression, Parallelism	All	GUI	FTP, SFTP, SMTP	Yes
IBM Aspera	Closed	Cloud	HDD striping, Compression, FASP	All	GUI	FASP	Yes
MoveIt	Closed	Client	CPU Freq, Parallelism, Pipelining	Medium	GUI	FTP, SFTP, HTTP	Yes
Gscape EFT	Closed	Client	None	All	GUI	FTP, HTTP	No
RSSBus	Closed	Cloud	HDD Stripe, TCP buffer	All	GUI	FTP, HTTP,	Yes
RClone	Open	Client	TCP buffer	All	Cmd line	SFTP, Amazon S3, Google Drive	Yes
Serv-U	Closed	Cloud	None	All	GUI	FTP	No
CloudFuze	Closed	Cloud	Caching	All	GUI	SFTP, Dropbox, Amazon S3	No

Table 1: Comparison among different managed file transfer services.

which the users can drag and drop the data files from the source to destination. However, the solution is proprietary and requires payment for each bit of data which is transferred. Next, server to server data transfer with the ability to share data via emails was showcased by Serv-U (Serv-U, 2017). Despite its cloud hosted transfer capabilities, the speed of transfer remains a lot to be desired which is identified here.

Server to server data transfers in industry environments have been evaluated in (Carroll, 2017). The proponents have analyzed what benchmarks should be followed when transferring data from one server to another at an industry level. They have evaluated and emphasized the importance of logging and timestamping the transfer activity at every stage of the transfer for security and auditing purpose. Additionally, the authors provided a mechanism for identifying nearest server when the transfer needs to be speed effective. However, besides security and speed, how to support multi-protocol data transfers and design criteria for such a distributed data transfer system were not discussed. Also the overhead of time-stamping at every stage has been ignored by the researchers.

Usage of cloud platform to provide data transfer as a service to scientists with increased performance in terms of speed and security has been addressed in (A et al., 2012) which identifies Globus as a probable solution. Recent developments have focused on secured data transfer using Globus with transfer optimization by static parameters (Chard et al., 2017; Liu et al., 2017). The authors showed results of data transfers using static parameters over network via REST APIs, however Globus does not employ dynamic throughput optimization or interprotocol translation.

Scalability and data exchange capabilities of different MFT platforms have been discussed in (Singh et al., 2016). Vertical and horizontal scaling systems have been suggested by the authors in order to address the issue of disk space overflow. *Endpoint Scanning, Auto Recovery and Scheduler* modules have been proposed to address the need for constantly keeping the connectivity up and running to ensure glitch-free data transfer. Single point of scheduling server for multiple MFT communication, configuring one or more properties of servers and updating the entire system about it and using a centralized server for managing configurable job queue have been proposed. However, the issue of achieving effective software designs to incorporate patch updates at run time has been addressed to a limited extent by the authors. Additionally, the throughput of data transfer for a variety of file sizes have not been addressed, however, research has shown that file size is a critical component.

Several MFT solutions present web based dashboards which allow users to specify the sending and receiving entities and also enable selection of the file which will be transferred over the network (Mover.io, 2017). However, many of these solutions are commercial and users need to purchase subscriptions in order to use these services for transferring the data. Also cross protocol data transfers together with optimization of the transfer itself are hardly addressed. Unfortunately, the commercial MFT services do not provide estimation of data arrival time, which is a critical information as it allows the recipient to prepare storage and data analytic tools. Also, automated protocol translation features have been incorporated to a limited extent. Finally, lack of robustness in these services is a common bottleneck to user-friendliness, which ultimately causes users to revert to traditional mechanisms of data transfers like FedEx. A comparative analysis of various MFT services from the perspective of functionality, usability and optimization capability is presented in Table 1.

3 ONEDATASHARE VISION

Efficient transfer of data is still a challenge despite the modern innovations in network infrastructure and availability of large bandwidth Internet (Yildirim et al., 2016). Being able to effectively use these high speed networks is becoming increasingly important for wide-area data transfer as well as for distributed and cloud computing applications which are data intensive. We propose *OneDataShare* as a solution for universal data transfer with the following major goals:

Reduce the time to delivery of the data. Large scale data easily generated in a few days may presently take weeks to transfer to the next stage of processing or to the long term storage sites, even assuming high speed interconnection and the availability of resources to store the data (NSF, 2011). Through *OneDataShare*'s application-level tuning and optimization of TCP-based data transfer protocols (i.e., FTP, GridFTP, SCP, HTTP), the users will be able to obtain throughput close to the theoretical speeds promised by the high-bandwidth networks, and the performance of data transfer will not be a major bottleneck for data-intensive applications any more. *The time to the delivery of data will be greatly reduced, and the end-to-end performance of data-intensive applications relying on remote data will increase drastically.*

Provide interoperation across heterogeneous data resources. To meet specific needs of users (i.e., scientists, engineers, educators and others), numerous data storage systems with specialized transfer protocols have been designed, with new ones emerging all the time (Shoshani et al., 2002). Despite the familiar file system-like architecture that underlies most of these systems, the protocols used to exchange data with them are mutually incompatible and require the usage of specialized software. The difficulties in accessing heterogeneous data storage servers and incompatible data transfer protocols discourage researchers from drawing from more than a handful of resources in their research, and also prevent them from easily disseminating the data sets they produce. *OneDataShare* will provide interoperation across heterogeneous data resources (both streaming and at-rest) and on-the-fly translation between different data transfer protocols. *Sharing data between traditionally non-compatible data sources will become very easy and convenient for the scientists and other end users.*

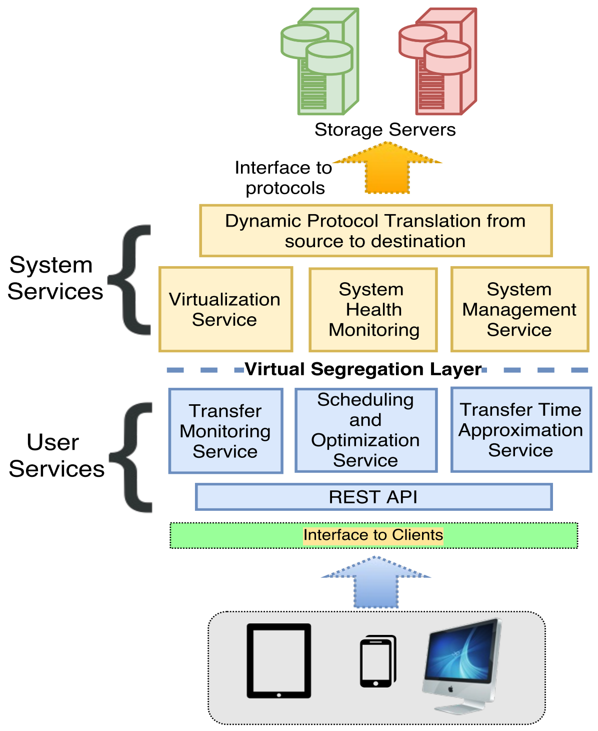
Decrease the uncertainty in real-time decision-making processes. The timely completion of some compute and analysis tasks may be crucial for especially mission-critical and real-time decision-making processes. If these compute and analysis tasks depend on the delivery of certain data before they can be processed and completed, then both the timely delivery of data and the predictive ability for estimating the time of delivery become very important (DOE, 2013). This would allow the researchers/users to do better planning, and deal with the uncertainties associated with the delivery of data in real-time decision making process. *OneDataShare*'s data throughput and delivery time prediction service *will eliminate possible long delays in completion of a transfer operation and increase utilization of end-system and network resources by giving an opportunity to provision these resources in advance with great accuracy.* This will enable the data schedulers to make better and more precise scheduling decisions by focusing on a specific time frame with a number of requests to be organized and scheduled for the best end-to-end performance.

While realizing these goals, *OneDataShare* will make the following contributions to the distributed and cloud computing community: *(i)* implementation of novel and proven techniques (online optimization based on real-time probing, off-line optimization based on historical data analysis, and combined optimization based on historical analysis and real-time tuning) for application-level tuning and optimization of the data transfer protocol parameters to achieve best possible end-to-end data transfer throughput; *(ii)* development of a universal interface specification for heterogeneous data storage endpoints and a framework for on-the-fly data transfer protocol translation to provide interoperability between otherwise incompatible storage resources; *(iii)* instrumentation of end-to-end data transfer time prediction capability, and feeding of it into real-time scheduling and decision making process for advanced provisioning, high-level planning, and co-scheduling of resources; and **(iv)** deployment of these capabilities as part of a stand-alone *OneDataShare* cloud-hosted service to the end users with multiple flexible interfaces.

4 ONEDATASHARE DESIGN

OneDataShare will add significant value to the field of cloud based data transfers due to its adoption of elasticity and sustainability. However, significant features have been specified in *OneDataShare* architecture to ensure that the solution adheres to its promised performance. Incorporation of these features is not only performance critical, but also vital to the acceptance of *OneDataShare* by the general users.

This section identifies some of the most important key design features of *OneDataShare*. Figure 2 provides a visual representation of *OneDataShare*, showing the interim protocols and mechanisms in place and how it achieves improved data sharing and trans-



Storage Servers
Interface to protocols
Dynamic Protocol Translation from source to destination
System Services
Virtualization Service
System Health Monitoring
System Management Service
Virtual Segregation Layer
User Services
Transfer Monitoring Service
Scheduling and Optimization Service
Transfer Time Approximation Service
REST API
Interface to Clients

Figure 2: OneDataShare high-level overview.

fer optimization while providing a thin client interface to the users. *OneDataShare* is designed to support a variety of interfaces for the users which include web interface, smartphone and tablet apps, as well as command-line and file system interfaces. When a user makes a request to *OneDataShare* for a data transfer, the request is submitted to the engine of the *OneDataShare* via RESTful API. *OneDataShare* contains a collection of schedulers, protocol translators, provenance managers and cloud manager. This complex and dynamic collection of modules appears as a black box to the general users of *OneDataShare*. Also, the entire protocol translation, optimization and scheduling of arrival time of the data files is maintained in the cloud, thus ensuring reliability and high availability of service to the users. It is a ubiquitous service as users will only have to pay for the resource requirements of their specific requests, thereby saving unnecessary costs. Status of data transfers and health of the internal components are monitored by the system through the system-level services.

Based on the dynamic nature of *OneDataShare*, it must cater to the demands of the users regarding transfer of data which involves high speed of data delivery, interoperability, and improved estimation of delivery time which are discussed below.

4.1 High Speed Data Delivery

The most desirable feature of *OneDataShare* is high speed delivery of the data through application-level protocol tuning using the underlying network architecture. Prior work on application level tuning of transfer parameters mostly proposed static or non-scalable solutions to the problem with some predefined values for some generic cases (A et al., 2012; Hacker et al., 2002; Crowcroft and Oechslin, 1998; Lu et al., 2005). The main problem with such solutions is that they do not consider the dynamic nature of the network links and the background traffic in the intermediate nodes. In our previous work, we have developed three highly-accurate predictive models (Yin et al., 2011; Yildirim et al., 2011; Kim et al., 2012) which would require as few as three real-time sampling points to provide very accurate predictions for the optimal parallel stream number. These models have proved to have higher accuracy compared to existing similar models (Hacker et al., 2002; Lu et al., 2005) which lack in predicting the parallel stream number that gives the peak throughput. We have analyzed the combined effect of these transfer parameters on end-to-end data transfer throughput, and developed several predictive (offline) and dynamic (online) algorithms to choose the best parameter combination to minimize the delivery time of the data (Yildirim et al., 2016; Arslan et al., 2013; Kim et al., 2015; Yildirim and Kosar, 2012; Arslan et al., 2016).

As heuristic approaches are solely based on domain knowledge, an alternative approach would be the acquisition of such knowledge directly from the historical data transfer logs. Heuristic approaches might over-generalize the prediction for some systems, where historical analysis based approaches actually mine useful knowledge from the user data transfer patterns, end node characteristics, and the specific parameter values of the connecting links. Such approach can provide more personalized optimization for the users and the corresponding systems. We have collected production level data transfer logs from XSEDE, a well-known infrastructure for high performance scientific computing. Those transfer logs contain information about end systems, dataset, network links, and the protocol along with parameter settings. Proper data mining techniques on such a rich collection of historical logs should reveal interesting user transfer patterns, intensity of network traffic (e.g., peak or off-peak hours), and end-system specific suitable parameter settings. Even if the data transfer logs are very detailed, it is not possible to collect information for a transfer with many different parameter settings. Only a partial view of the whole parameter space can be extracted. As the par-

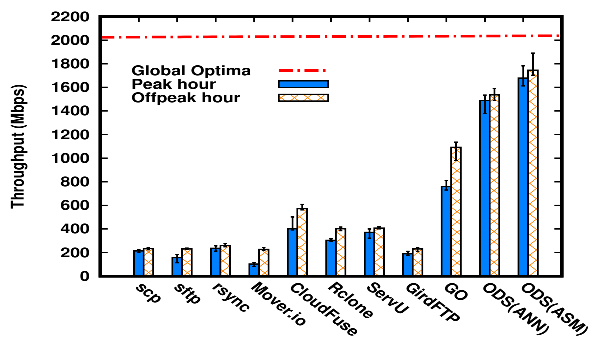


Figure 3: Performance comparison between different data transfer services for both peak and off-peak hours.

tial view of parameter space might not contain the optimal settings, we can use regression or interpolation techniques to predict a more detailed view.

Another important aspect of data transfer optimization is the dynamic load change in the network link. Even if the historical analysis provides more fine-grained parameter tuning than heuristics based approaches, still it might prove to be sub-optimal for a network load that is different from the network load present in the historical logs. Instead of transferring the whole dataset using parameters from historical analysis, some portion of the dataset could be transferred to assess the network load intensity and based on the results the parameters can be tuned effectively. However, deciding the appropriate sample transfer size and number is a critical research issue. Since the network load can change during a long data transfer, dynamic tuning of the parameters on the fly will be a very useful feature.

We introduced historical analysis based approach in ANN+OT (Nine et al., 2015) that uses machine learning techniques to learn optimal parameters from the historical logs. In this model we have used machine learning based techniques to get optimal parameters for different types of network. We have used Artificial Neural Networks (ANN) and Support Vector Machines (SVM), two well-known supervised learning techniques, to learn optimal parameters from the transfer logs. This approach provided considerable improvement over the existing techniques. Even though historical analysis can provide more accurate parameters than heuristic approaches, the dynamic nature of the network might prove those parameters sub-optimal. Therefore, current network condition is also an important factor to set tuning parameters. We have made some progress in real-time sampling in ANN+OT. It performs a series of real-time sampling to assess the current network condition and update parameters.

In our most recent work (Nine et al., 2017), we introduced a two-phase model aiming to reduce the performance degradation due to sampling overhead. It uses a robust mathematical model based offline analysis on the historical logs to interpolate the throughput surface for the parameter space. It stores the most interesting regions of the surface and local maxima points for different network conditions. During online phase, instead of performing sample transfers blindly, it adapts the parameters using guidelines from offline analysis to achieve faster convergence.

In an effort to compare OneDataShare performance to other state-of-the-art solutions, we have run an experiment in which we transferred data using two production level data centers, one located in Washington DC and the other one in San Jose, CA. Here, the data were transferred from the server in Washington DC to the server in San Jose. Both servers were provisioned with Ubuntu Operating System and configured for transferring a common data set using a number of MFT services mentioned here. This testing was conducted based on our study of current MFT services and it is a good indicator of the performance of current MFTs. We tested the MFTs by transferring data during peak and off-peak times of the day. The same dataset was used for all the MFTs. The dataset was 15.4 GB in size and each individual file ranged between 200 MB to 300 MB. The peak time was considered between 9:00 AM till 7:00 PM and off-peak time includes the rest of the hours. We transferred the data using different file transfer services, such as - SCP, Rsync, SFTP, Mover.io, Rclone, CloudFuze, Serv-U, GridFTP, Globus Online (GO) and compared those results with our OneDataShare (ODS) service with two different optimization models - ANN+OT and ASM as shown in Figure 3, where we refer them as ODS(ANN) and ODS(ASM) respectively.

Most of the current data transfer services achieve a fraction of the achievable throughput due to their sub-optimal protocol parameter choices. SCP and Rsync are very popular file transfer tools, however, their performance is quite marginal. SFTP protocol itself has many performance issues for long RTT high speed networks. It can be observed in Figure 3 that all data transfer mechanisms display a performance between 83.28 Mbps to 1900 Mbps. Mover.io MFT has the lowest thoughput of 83.28 Mbps. Mover.io uses SFTP to transfer the data, however, its limitation of scaling horizontally when multiple files need to be transferred was the prime bottleneck for the observed throughput (Mover.io, 2012). Additionally, Mover.io service providers have mentioned slow transfer speeds when their platform is used to transfer files via SFTP (Mover.io, 2012). Following Mover.io, tests were conducted for CloudFuze, which is a cloud based file transfer application providing low overhead and high speed (CloudFuze, 2016). CloudFuze provided a throughput of 402 Mbps during peak hours and 572 Mbps during off-peak hours. It provided fast and secure file migrations which deployed a user-friendly interface for the clients. Next, Rclone was tested; Rclone is a client application to sync files between two servers. It does not provide a GUI, however it supports a wide variety of storage platforms including DropBox, Microsoft Azure, WebDAV, and Yandex (RClone, 2017). Rclone yielded a throughput of 304 Mbps and 402 Mbps during peak and off-peak hours respectively. Afterwards another proprietary cloud based file transfer mechanism called Serv-U (Serv-U, 2017) was tested. It supports file transfers and provides a user interface which can be used to upload, transfer and download data into the servers. During data transfers, it was seen that the throughput of Serv-U is 371 Mbps during peak hours. During off-peak hours, it gives a throughput of 407 Mbps. GridFTP overcomes many issues of FTP based protocols, such as data channel reuse, pipelining, parallel streams, concurrent file transfers. We can see that GridFTP and Globus Online perform better than SCP, Rsync or SFTP. However, performance can be increased by tuning the protocol parameters, like - pipelining, parallelism, concurrency, and tcp-buffer size. One of the two optimization algorithms used in *OneDataShare*, ODS(ANN), shows 2x performance increase compared to GO during peak hours. Our second optimization algorithm, ODS(ASM), achieved the highest throughput among all tested tools during both peak and offpeak hours.

4.2 Interoperability

OneDataShare ensures that data sent using *Protocol X* can be delivered at the recipient in a different protocol (i.e. Protocol Y) without much effort from the user end. This protocol translation mechanism is a black box to the user and implements a fast, effective and timely solution to on-the-fly interoperability between different transfer protocols. In Figure 4, it is seen that a user from Location A wants to transfer a file to a system at Location B using GridFTP protocol. However, the user from location B will accept the data file using a protocol which is not compatible with GridFTP (e.g., Dropbox).

Placing the burden of protocol translation on the user will provide extra burden on them. Users may not be adept at protocol translation mechanisms as they are not used to technological perspectives of converting data from one protocol to the next. It is very common that large volumes of data are transferred by users with no background of computing knowledge, hence, carrying out protocol translation manually by users will impose burden on them and discourage them to use the network for transferring data.

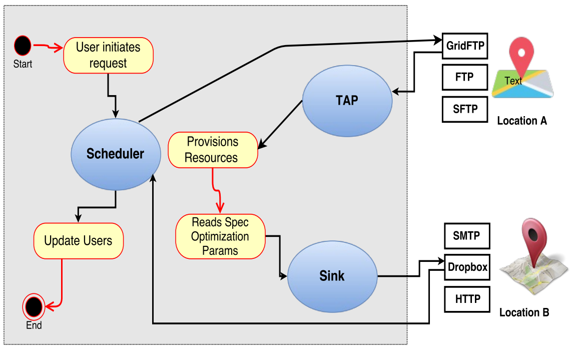


Figure 4: OneDataShare's interoperability components.

OneDataShare uses the state of the art *Tap* and *Sink* mechanisms for protocol translation as shown in Figure 4. When the process is passed through the protocol translation gateway and the server is triggered, instantaneously the server reads the incoming file properties and contents, analyzes the incoming protocols used in the tap and writes the data in memory. After that, the data bytes are re-commissioned into a desirable format of the protocol accepted at the recipient, which is Dropbox in this example. Finally, the file is delivered.

In this framework, the readable resources implement the *Tap* operation to acquire a data *tap* which will emit data into a data *sink*; and the write-able resources implement *Sink* operation to acquire a data *sink* which will drain data from a data *tap*. The life cycle of *OneDataShare* inter-protocol translation begins with the user triggering a transfer to the transfer manager known as *tap*. The *tap* is one of the core modules which is responsible for accessing the files source and preparing the protocol translation for successful transfer of the file to its destination

The *Tap* is the frontier interface, responsible for abstracting the back-end components of OneDataShare, which may pose significant complexity if exposed to general users. The *Tap* provisions a unified single point opening wedge to the system. The UI is a dashboard which presents all the functionalities which can be harnessed by the users. The dashboard is also responsible for transmitting the selected functionalities and optimization to the back-end engine. Afterwards, provisioning of the computing and network resources are done. The system obtains authorized file access permissions and loads all child directories from which the transfer files may be selected. Next, the *Tap* receives this input and ultimately enables the deployment of the process which places the data files into the provisioned resources, attaches protocol related information which primarily constitutes of the names of source and destination protocols and transmits these journals to *Sink*.

The *Sink* is the delivery manager of data which triggers the recipients of the incoming requests and prepares them for receiving this data. It is a self-protocol-translating system which takes input from the *Tap* and formats the data into forms which are suitable to be accepted at the recipient. It envisages the incoming data and loads it in memory, typically associating them to the resulting protocol, embedding them and mitigating the network transfer requirements based on the optimization parameters as specified by users. It is empowered with a multitude of child processes which cater to the requirements of multiple block data transfers per unit time. More specifically, *Sink* is fully fitted with the required processes of computing, memory and network resources which are used during protocol translations.

This solution presents a number of strengths over other solutions: *i)* neither servers nor clients need to install custom software to take advantage of the protocol translation *OneDataShare* offers; *ii)* support for additional protocols can be added on-the-fly without having to make changes to any other system. Through this framework, *OneDataShare* will provide interoperability and on-the-fly protocol translation between a wide-range of data transfer protocols and storage systems, including but not limited to FTP, GridFTP, HTTP, SCP, SFTP, Rsync, UFTP, UDT, iRODS, SMTP, Dropbox, and Google Drive.

4.3 Transfer Time Estimation

It is seen in the modern world that both scientific research organizations and industries are struggling to estimate the time of arrival of data after it has been transferred. This information is critical as the arrival of data will require provisioning storage for the data itself at the recipient's end. Additionally, the algorithms and software components which will be used to analyze these data need to be prepared before it arrives, and compute resources need to be provisioned.

If storage and compute resources are provisioned long time before the arrival of data then the client needs to pay for the time during which these resources were provisioned but left idle because the data did not arrive. Hence, estimation of arrival time accurately will provide benefits to the users in a multitude of ways which includes time and cost benefits. *OneDataShare* will use dynamic prediction algorithms to estimate arrival time of data to a significant degree of accuracy, thereby allowing the users to provision both storage, computing, software and human resources on time for using the data. It principally ensures that the user is well aware and can plan ahead to better utilize the resources and analyze the data.

Using the prediction models which we have developed in our previous work, *OneDataShare* will provide an "end-to-end data transfer throughput and delivery time estimation service" for external data scheduling and management tools. Prior research on predictive models showed that we can estimate the real-time achievable throughput with as low as 5% error rate on average (Kim et al., 2015; Yildirim and Kosar, 2012; Yildirim et al., 2013). This service will include information regarding available end-to-end network throughput for the user, the total time it will take to transfer a particular dataset, network and end-system parameters that need to be used in order to achieve highest end-to-end throughput.

5 CONCLUSIONS

We have presented our vision for *OneDataShare* as a cloud-hosted data transfer scheduling and optimization service, which aims to remove the burden of managing end-to-end data transfers from the shoulders of users. It en-visions the goal of providing a simple, yet user friendly platform to share data irrespective of size, type, and difference in protocols between sender and receiver. Real life experiments have been conducted to exhaustively compare the performance of *OneDatashare* with other existing services and it is seen that the data transfer throughput of *OneDataShare* is up to 6.5 times greater than the other solutions currently in place. Building *OneDataShare* into a reliable and sustainable system beneficial to all is an area of future research.

ACKNOWLEDGEMENTS

This project is in part sponsored by the National Science Foundation (NSF) under award number OAC-1724898. We also would like to thank Brandon Ross, Vandit S. Aruldas, Mythri Jonnavittula, Ahmad M. Temoor, Dhruva K. Srinivasa, Jimmy Huang, Sourav Puri, Siddharth Ravikumar, and Amardeep Virdy for their contributions in the implementation of different *OneDataShare* components.

REFERENCES

A, B., Bresnahan, J., Childers, L., Foster, I., Kandaswamy, G., Kettimuthu, R., Kordas, J., Link, M., Martin, S., Pickett, K., and Tuecke, S. (2012). Software as a service for data scientists. *Commun. ACM*, 55(2):81–88.

A., S. B., Håkansson, C. J., Laure, E., Livenson, I., Stranák, P., Dima, E., Blommesteijn, D., and van de Sanden, M. (2015). B2share: An open escience data sharing platform. In *e-Science (e-Science), 2015 IEEE 11th International Conference on*, pages 448–453. IEEE.

Arslan, E., Guner, K., and Kosar, T. (November 2016). Harp: predictive transfer optimization based on historical analysis and real-time probing. In *Proceedings of IEEE/ACM Supercomputing Conference (SC16)*.

Arslan, E., Ross, B., and Kosar, T. (August 2013). Dynamic protocol tuning algorithms for high performance data transfers. In *Proceedings of the Int. European Conference on Parallel and Distributed Computing (Euro-Par 2013), Aachen, Germany*.

Carroll, J. (2017). Systems and methods for managed data transfer. US Patent 9,537,834.

Chard, K., Foster, I., and Tuecke, S. (2017). Globus: Research data management as service and platform. In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*, page 26. ACM.

Cho, B. and Gupta, I. (2011). Budget-constrained bulk data transfer via internet and shipping networks. In *8th International Conference on Autonomic Computing*.

CloudFuze (2016). Why cloudfuze: Quickly connect with powerful providers like google drive, dropbox, or box from a single screen and login. *https://www.cloudfuze.com/why-cloudfuze/*.

Crowcroft, J. and Oechslin, P. (1998). Differentiated end-to-end internet services using a weighted proportional fair sharing tcp. *ACM SIGCOMM Computer Communication Review*, 28(3):53–69.

DOE (2013). Accelerating scientific knowledge discovery (askd) working group report.

Egeland, R., Wildish, T., and Huang, C.-H. (2010). Phedex data service. In *Journal of Physics: Conference Series*, volume 219, page 062010. IOP Publishing.

Garfienkel, S. (2007). An evaluation of Amazon's Grid computing services: EC2, S3 and SQS. Tech. Rep. TR-08-07, Aug 2007.

Hacker, T. J., Noble, B. D., and Atley, B. D. (2002). The end-to-end performance effects of parallel tcp sockets on a lossy wide area network. In *Proceedings of IPDPS '02*, page 314. IEEE.

IBM (2017a). Aspera fasp high-speed transport. http://asperasoft.com/fileadmin/media/Asperasoft.com/ Resources/White_Papers/fasp_Critical_Technology _Comparison_AsperaWP.pdf.

IBM (2017b). Ibm aspera direct-to-cloud storage. https://www-01.ibm.com/common/ ssi/cgi-bin/ssialias?htmlfid=ZZW03322USEN.

Kim, J., Yildirim, E., and Kosar, T. (2015). A highly-accurate and low-overhead prediction model for transfer throughput optimization. *Cluster Computing*.

Kim, J., Yildirim, E., and Kosar, T. (November 2012). A highly-accurate and low-overhead prediction model for transfer throughput optimization. In *Proc. of DISCS'12 Workshop*.

Liu, Z., Balaprakash, P., Kettimuthu, R., and Foster, I. (2017). Explaining wide area data transfer performance. In *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '17, pages 167–178, New York, NY, USA. ACM.

Lu, D., Qiao, Y., Dinda, P. A., and Bustamante, F. E. (2005). Modeling and taming parallel tcp on the wide area network. In *Proceedings of IPDPS '05*, page 68.2. IEEE.

Mover.io (2012). Sftp is moving in. *https://mover.io/blog/2012/08/28/sftp-is-moving/*.

Mover.io (2017). Mover.io cloud file migrations. "url:https://mover.io/services/". Access: 2017-11-01.

Nine, M. S. Q. Z., Guner, K., and Kosar, T. (2015). Hysteresis-based optimization of data transfer throughput. In *Proceedings of the Fifth International Workshop on Network-Aware Data Management*, NDM '15, pages 5:1–5:9, New York, NY, USA.

Nine, M. S. Q. Z., Kemal, G., Ziyun, H., Xiangyu, W., Jinhui, X., and Tevfik, K. (2017). Data transfer optimization based on offline knowledge discovery and adaptive aampling. In *Proceedings of the IEEE International Conference on Big Data*.

NSF (2011). Task force on grand challenges final report.

RClone (2017). Rclone-rsync for cloud storage. *https://rclone.org/commands/rclone_sync/*.

RSSBUS (2017). Rssbus: Load blanced and high availability setups. "url:https://www.rssbus.com/kb/articles/tutorial-high-availability.rst". Accessed: 2017-11-02.

Serv-U (2017). Serv-u managed file transfer and serv-u ftp server. *https://support.solarwinds.com/Success_Center/Serv-U_Managed_File_Transfer_Serv-U_FTP_Server*.

Shoshani, A., Sim, A., and Gu, J. (2002). Storage resource managers: Middleware components for grid storage. In *NASA Conference Publication*, pages 209–224.

Singh, A., Gupta, B., Yatzeck, F., Dixit, S., Mandadapu, S., and Vallamkonda, S. (2016). Managed filed transfer utilizing dynamic horizontal and vertical scaling. US Patent 9,521,187.

Yildirim, E., Arslan, E., Kim, J., and Kosar, T. (2016). Application-level optimization of big data transfers through pipelining, parallelism and concurrency. *IEEE Transactions on Cloud Computing*, 4(1):63–75.

Yildirim, E., Kim, J., and Kosar, T. (2013). Modeling throughput sampling size for a cloud-hosted data scheduling and optimization service. *Future Generation Computer Systems*, 29(7):1795–1807.

Yildirim, E. and Kosar, T. (2012). End-to-end data-flow parallelism for throughput optimization in high-speed networks. *JGC, Springer*, 10(3):395–418.

Yildirim, E., Yin, D., and Kosar, T. (2011). Prediction of optimal parallelism level in wide area data transfers. *IEEE Transactions on Parallel and Distributed Systems*, 22(12).

Yin, D., Yildirim, E., and Kosar, T. (2011). A data throughput prediction and optimization service for widely distributed many-task computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(6).

Zhao, S., Prenger, K., and Smith, L. (2013). Stormbow: A Cloud-Based Tool for Reads Mapping and Expression Quantification in Large-Scale RNA-Seq Studies. *ISRN Bioinformatics*, 2013:1–8.