Azerbaijan Journal of High Performance Computing, Vol 1, Issue 1, 2018, pp. 66-86
https://doi.org/10.32010/26166127.2018.1.1.66.86

Data Migration for Large Scientific Datasets in Clouds

Akos Hajnal[1], Eniko Nagy[1], Peter Kacsuk[1] and Istvan Marton[1]

[1]*Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary, akos.hajnal@sztaki.mta.hu, eniko.nagy@sztaki.mta.hu, peter.kacsuk@sztaki.mta.hu*

*Correspondence: Peter Kacsuk, nstitute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary, peter.kacsuk@sztaki.mta.hu

Abstract

Transferring large data files between various storages including cloud storages is an important task both for academic and commercial users. This should be done in an efficient and secure way. The paper describes Data Avenue that fulfills all these conditions. Data Avenue can efficiently transfer large files even in the range of TerraBytes among storages having very different access protocols (Amazon S3, OpenStack Swift, SFTP, SRM, iRODS, etc.). It can be used in personal, organizational and public deployment with all the security mechanisms required for these usage configurations. Data Avenue can be used by a GUI as well as by a REST API. The papers describes in detail all these features and usage modes of Data Avenue and also provides performance measurement results proving the efficiency of the tool that can be accessed and used via several public web pages.

Keywords: data management, data transfer, data migration, cloud storage

1. Introduction

In the last 20 years collecting and processing large scientific data sets have been gaining ever increasing importance. This activity requires first of all large storage systems, where the large scientific data can be stored. In the 00's many storage systems with different kind of access protocols (GridFTP [1], iRODS1, SRM [2], LCG File Catalogs [3], etc.) were introduced for this purpose. This heterogeneity caused a lot of problem for scientists if they wanted to access data stored in different kind of storage systems.

They had to learn the different protocols (APIs) and develop the data processing software according to these APIs. Usually, data processing softwares were developed for one particular API, that caused problem if the software had to access data stored in another kind of storage.

The other problem was transferring large scientific data sets among different

type of data storages. For example, scientists sometimes form new research consortium, that requires using a new storage that is easier to access by all consortium member. In the 00's, grid systems were very popular among scientists to share data and computing capacity. However, with the appearance of cloud systems it became obvious that clouds offer many advantages that make the use of grid systems obsolete. Therefore, scientists had to migrate both applications and data from grids to clouds. This migration also requires a tool that enables fast and efficient transfer of data among grid and cloud storages.

As clouds became more and more popular, scientific researchers and commercial companies store more and more data in cloud-based storages. Meanwhile, this approach has a lot of advantages, and there are several drawbacks too. One of the biggest drawback becomes clear when the owner of the data decides to move the data to another cloud or some kind of other storage. Cloud providers are typically reluctant to develop tools by which data from their cloud could be migrated to an external storage. However, sometimes a company decides to migrate the data to another cloud providers cloud due to economic or security reasons. Scientists can also find sometimes useful to migrate data for example, from a commercial cloud to an academic cloud to reduce cost, or vice versa, to transfer data from the academic cloud to a commercial cloud to get larger storage capacity.

All these problems mentioned above can be solved by a tool that is on one side prepared to efficiently execute large data transfer and can access many different storage types and on the other side provides a uniform API. Such a tool is Data Avenue that was developed by MTA SZTAKI. In this paper we describe in detail how the problems mentioned above can be solved by Data Avenue, and we also show the performance of transferring large data sets by Data Avenue.

The outline of the paper is the following. After the Introduction Section 2 describes the main features of Data Avenue including the possible usage modes and their security aspects. Section 3 overviews the GUI of Data Avenue designed for humans and Section 4 introduces the REST API to be used by software systems like scientific workflows. In Section 5 we show a concrete use case offered for the users of MTA Cloud (the cloud system of the Hungarian Academy of Sciences). Section 6 shows the results of performance measurements in order to prove the efficiency of data transfer by Data Avenue. In Section 7 we compare Data Avenue with similar tools and finally in the Conclusions we summarize the main messages of this paper.

2. Data Avenue

Data Avenue was originally motivated to allow grid applications to access diverse storage resources from different distributed computing infrastructures (DCIs) without the need of installing additional software, or using proprietary, storage-specific data transfer tools in application codes respectively [4], Previously, distributed applications, which were willing to read/store greater amount of data, must had been adapted at code-level to the particular storage provided in that particular DCI (e.g. access to the specific GridFTP site was part of application code using specific libraries deployed in all the worker nodes of the DCI). Therefore, jobs of these applications could not

be executed in other infrastructures, possibly providing different type of local storage (e.g., iRODS). Porting/migrating of these applications were very time-consuming, error prone, and tedious task, moreover, installation of a storage-specific software (such as AWS CLI for Amazon S32-compliant storages) was administratively prohibited on worker nodes in most of these clusters for security reasons.

Data Avenue offered a solution for these problems by providing a uniform interface (HTTP-based, REST and SOAP API) for the clients to access a wide range of storage types, where all data were mediated between the client and the storage by Data Avenue, but without needing the clients to be aware of how to communicate with a specific storage resource. Then, jobs could communicate with a Data Avenue host only, using simple HTTP tools (such as curl and wget commands, which were available at every worker node), which then guaranteed to forward/fetch data to/from the actual storage resource, using the appropriate storage protocol. This kind of data bridging service made possible that jobs could now be submitted to different DCIs and still had access to the same or new storages through Data Avenue to read, write, or exchange data respectively. Data Avenue was also used in WS-PGRADE/gUSE [5] workflow system to generalize data access to any type of remote storage resources supported by Data Avenue.

Data Avenue on the client side is connected over HTTP with clients, and on the storage side is connected with different types of storage resources using the appropriate storage-specific protocols (SFTP, SRM, iRODS, S3). Data Avenue was built using a plugin architecture, where plugins, so called ''adaptors", allow Data Avenue to connect to specific types of storages (e.g. S3 adaptor is used to connect to any S3-compliant storage). Higher-level Data Avenue services do not merely allow of connecting a client to a single storage resource but also enable transferring data between the same or different types of storage resources (e.g., from S3 to SFTP or vice versa) by connecting different adaptors' input and output streams. Data transfer is an asynchronous process that runs in the Data Avenue host, without the need of involving client's machine at all. This data transfer service can be used to move even a huge amount of data, by issuing a single command to Data Avenue and then just polling transfer status until it is completed. Since Data Avenue only streams data from input and output channels opened to the different storages, the amount of data to be forwarded is not limited by the (disk or memory) capacity of the Data Avenue host.

Data Avenue was recently further developed to be able to connect to more and more cloud storages (Amazon S3 and OpenStack Swift3 are tested, Google Cloud Storage and Microsoft Azure are under testing). In this way, Data Avenue has become a potential tool not only to migrate data from legacy grid storages to recent cloud storages (e.g. GridFTP to Swift), but also to migrate data between cloud storages. Data Avenue is planned to be extended by services to synchronize data between different storages (transfer incrementally), and perform scheduled transfers (e.g. to allow nightly backups).

2.1. Data Avenue Deployment Types

Data Avenue was developed in a modular fashion that allows of different setups

and configurations depending on the number of users, expected data transfer load, performance, availability, scalability, cost, or other aspects. The simplest configuration is composed of a single Data Avenue server instance and a database that provides persistence; in the most complicated case, we use replicated HTTPS proxies, replicated Data Avenue servers, and replicated database servers (detailed below).

The Data Avenue server is a web application that can be run in a web application container (such as Tomcat) and as database MySQL server can be used (accessed via JDBC driver by Data Avenue server). When the Data Avenue server is accessed remotely, it is required to have an HTTPS proxy installed (such as HAProxy or Nginx) in front of Data Avenue server to encrypt sensitive data (e.g. passwords or other sensitive data contents) exchanged between the client and the Data Avenue server. Note that HTTP proxies serve too goals: perform load balancing and terminate HTTPS connection. Load balancing is done by selecting one of the backends in a round-robin way to which the inbound network traffic is forwarded to. There are options to use other selection methods (e.g. based on some load or other metric-based criteria), but due to their overhead (gathering metrics, choosing) they might perform worse than simple round-robin. HTTPS termination means that the proxy encrypts the data exchanged between the client and proxy to protect against eavesdropping, while the connection between the proxy and the backend is unencrypted, which resides within a safe, private network.

When using Data Avenue services from program codes or shell scripts REST and SOAP APIs are available directly on the Data Avenue server through the HTTPS proxy. The graphical user interface (GUI) of Data Avenue was developed using pure JavaScript (using Angular framework and Bootstrap JS libraries), which makes the GUI usable from within any browser regardless of the browser is running on a PC or a mobile device. The GUI running in the browser uses REST calls to the Data Avenue server to perform operations. Every instance of Data Avenue service can provide the GUI. Depending on whether behind the HTTPS proxy there are replicated Data Avenue servers or not. the GUI communicates with a single or multiple Data Avenue servers, respectively, in a load-balanced fashion.

We differentiate three fundamental configurations depending on usage scenarios:

1. Personal desktop PC or server host deployment
2. Company/organization-level deployment
3. Public/scalable service deployment

In die personal configuration, we assume a single user wishing to access remote storage resources (manage, upload/download data from/to local disk) from his/her local desktop PC. The data traffic to convey is relatively little and no massive number of concurrent transfers is expected. There is no need to authenticate the user, keep track of transfers for later accounting; all resources (PC or a remote host) is under his/her authority, free or charged individually after resource usage. For this usage scenario. Data Avenue can be installed on the local PC using the dockerized version of Data Avenue (a single docker-compose command starts up a Data Avenue and a database container). No additional configuration is needed.

Right after that the user can open Data Avenue GUI from browser at URL http://localhost:8080/dataavenue (see Figure 1/a).

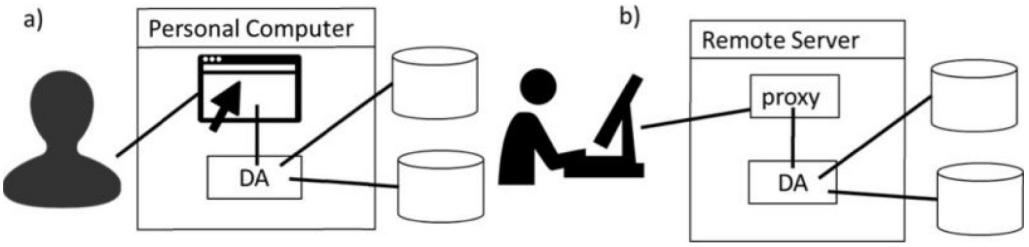

Figure 1: Personal DA

Note that in this case no encryption is needed between the browser and the Data Avenue server, as these data stays within the same machine (loopback), whereas the connection between the Data Avenue server and the remote storage resources are still encrypted by the storage access protocols themselves (e.g. HTTPS at S3). The user can transfer data between any storages, including local disk (upload/download) but transfers will be interrupted when the local PC is turned off.

If the user wishes to transfer larger amount of data which might take several hours or days, he/she has the potential to deploy Data Avenue on a remote server host (which is constantly up) locally or using a virtual machine (VM) in a private/public cloud, as shown in Fig. 1 .b. The same docker-compose command can be used, but in this case, an HTTPS proxy container is also launched, which connects to the Data Avenue server internally, and Data Avenue GUI is served at address https://server-host/dataavenue/. The user now can start longer running transfers, which keep running on the server host even if the user closes the browser or turns off his/her PC.

The drawback of the previous configurations is that when transferring a greater amount of data or performing sever al transfers simultaneously the hardware of a single host might not be able to serve the load and the performance of these transfers might degrade (when CPU or network capacity are reached). For this reason, when not only a single user but a group of people (e.g. member s of some organization) are going to use data transfer (see Fig. 2), Data Avenue can be deployed as a service at university/company-level. To pr epar e for concurrent load both HTTP proxies and Data Avenue servers are replicated on a fixed number of hosts (components deployed individually), calculated based on the expected load/usage. which might be adjusted later manually (new hosts are added or superfluous hosts removed). The university/- company might obtain an official certificate for the domain name (in Domain Name Service (DNS) under which all HTTP proxies are registered). This configuration is shown in Fig. 2. In this multi-user environment, Data Avenue is able to keep track user activities (transfer histories) by using Data Avenue's access key system. Data Avenue access keys are like passwords and usernames combined: for each user a unique identifier is given and all Data Avenue services can be used by showing a valid access key, which is verified at each request. (Note that Data Avenue access key just gives permission to talk to Data

Avenue, which differs from credentials required by Data Avenue to connect to a remote storage, e.g. S3 access key and secret key.) By connecting data transfers and user access keys in the database, each user's transfer history (what copied to where, date, number of bytes transferred, etc.) can be looked back or used for accounting, respectively. Organizations central authentication system (e.g. LDAP) can be connected to Data Avenue access keys.

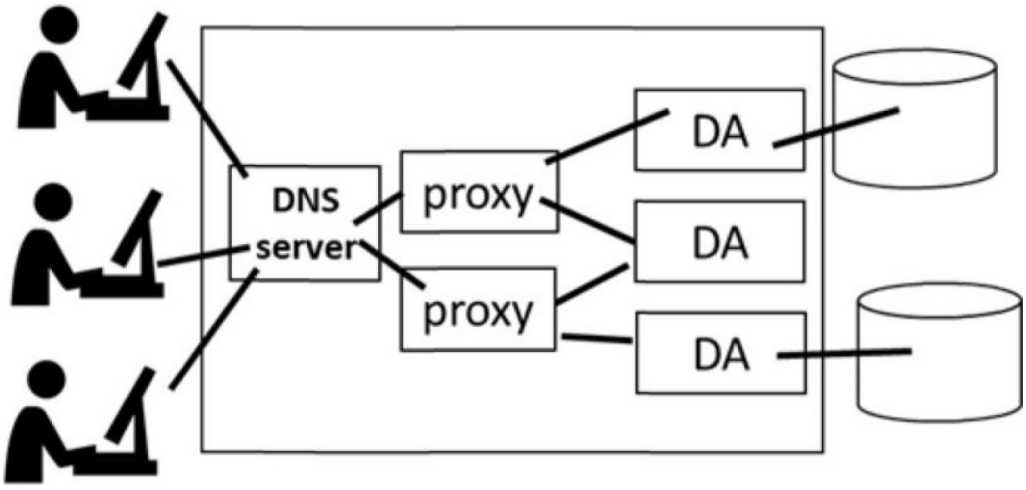

Figure 2: Multi-user DA

In the last setup, the number of users (public service) and/or the level of load varies largely in time. To setup several hosts to prepare for peak loads might waste resources in idle times, to use little hosts causes degraded data transfer throughput. To find the trade-off between cost and performance a possible solution is to use a dynamically changing infrastructure, which grows or shrinks automatically depending on the current load. This configuration is illustrated in Fig. 3. MiCADO [6] is a potential tool to build and control such an infrastructure, but other tools might also work equally. Metrics to control scaling in and out events include CPU load, heap memory usage, and network traffic.

2.2. Data Avenue Security

Confidentiality and integrity of the data transferred are important aspects in using any tool like Data Avenue.

As described previously, whenever' the Data Avenue server' is accessed remotely, HTTPS connection is established between clients and the Data Avenue server, which guarantees that all data (including Data Avenue commands and credentials) are secured using SSL/TLS. Data Avenue access keys ensure to allow access to Data Avenue services only for' authenticated users. Storage credentials serve for' authenticating to remote storage r esources. We note that Data Avenue does not stor e credentials for the remote resources neither on disk nor in database; they are kept in memory only for the time required (to open connection

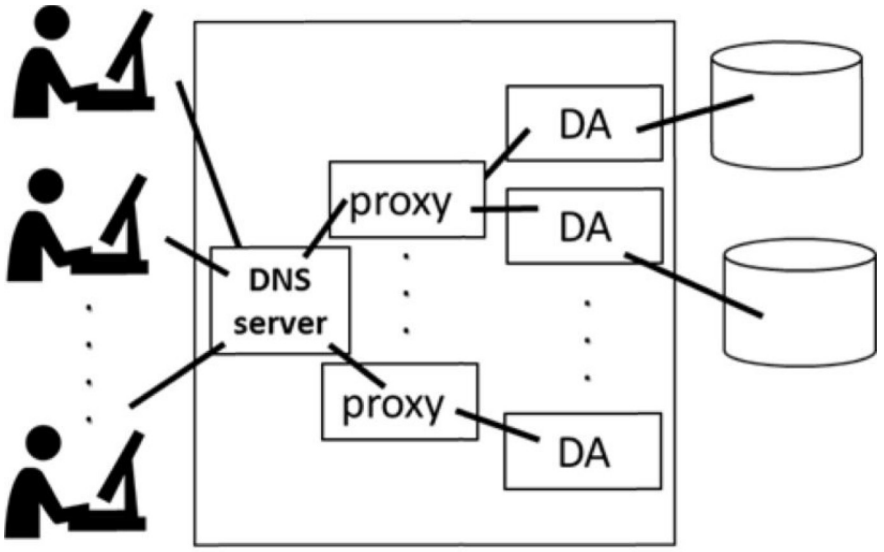

Figure 3: Scalable DA

and authenticate to the remote storage).

The encryption of the communication between the Data Avenue server and the different storage resources is ensured by the related, storage-specific protocol. For example, protocols used to access S3, Swift, GridFTP, SFTP, SRM servers ensure confidentiality of data sent to or received from the storage. When using ''third-party" transfers (e.g., S3 server-side copying within the same- or between different regions, or between GridFTP servers, respectively) the data are not routed through the Data Avenue server, thus use of Data Avenue has no effect on security. We note that Data Avenue does not store any data either in part on disk; data transfers are done using memory buffers only for the time of transfer.

Integrity of the sent data is guaranteed in the case of such storages where the storage-related protocol itself supports integrity checking. For example, when sending data to S3 storages, Data Avenue calculates MD5 hash of the sent data on client side (sender), which finally verified against server-side (receiver) calculated hash. If they do not match, the transfer is considered to be failed. Data Avenue also offers failover mechanisms to recover from temporary failures (e.g., short time network outage). It is done by automatically re-trying data transfers for a (configurable) specified number of times, in a specified re-try delay time, re-trying to transfer the data again up to at most a predefined number bytes. The latter solution guarantees that failed data transfer cost can be escalate.

3. GUI

The aim of the new Data Avenue Graphical User Interface was to serve the needs of users and make Data Avenue more easy to use. By having a GUI, there is no need to use long REST-API commands during data transfer.

In order to use Data Avenue service, the GUI needs to have an accessible Data Avenue service endpoint to be configured before use. Fig. 4 shows the Settings menu, where end-users can configure there their own Data Avenue service (web application name only, if the host serving the GUI also corresponds to a Data Avenue service) and the access key that allows to use Data Avenue services.

In order to access a target storage, end-users should configure their credentials. Fig. 5 shows the needed credentials in the case of an S3 storage (access key, secret key).

Fig. 6 shows the appearance of the DA GUI. It was designed to have a clear, well-transparent design. The main page is separated into two main panels: the two storage's browser. End-users can browse within the source (left window), and the target (right window) storage's file system using DA service. As it can be seen, the buckets/directories and the files within them are appear in a list on the screen. All of the main operations are available while using the GUI: browse within the storage, up- load/download/delete/ copy/rename files/directories/buckets within the target storage.

Fig. 7 shows the list of the launched transfers at the Transfers tab. End-users can check the progress bar and the status of each transfer. The source and the target URIs,

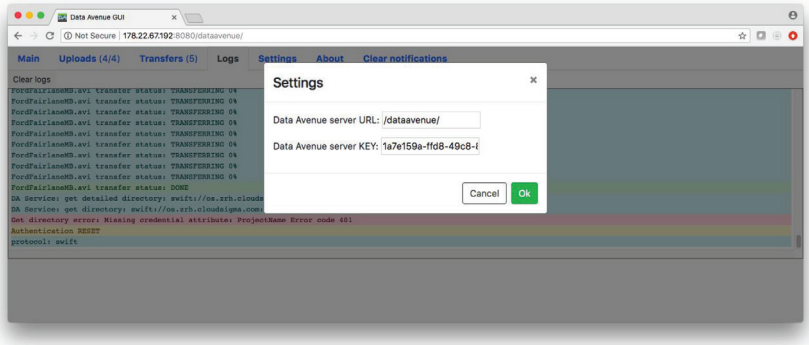

Figure 4: DA GUI settings

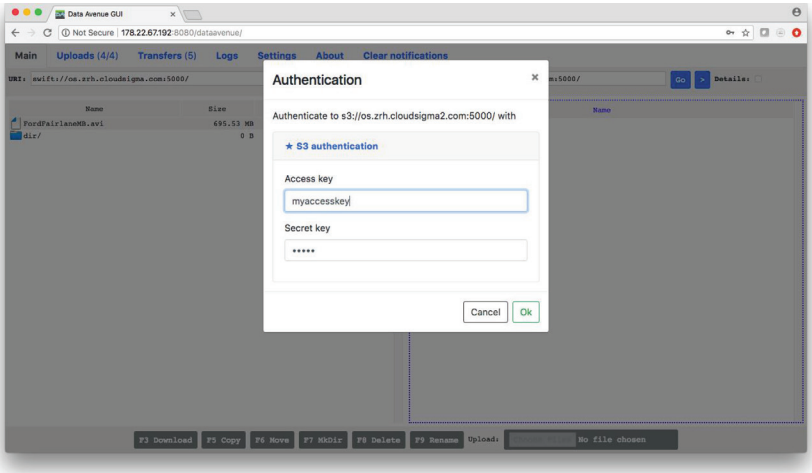

Figure 5: DA GUI S3 authentication

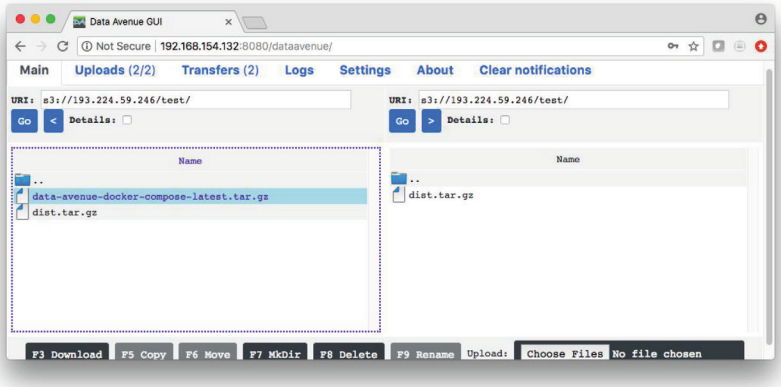

Figure 6: The overlook of the Data Avenue Graphical User Interface

and the size of the file which was moved are also included in the table.

The Uploads tab can be seen in Fig. 8. The progress bar, which shows the actual progress of the upload operation, and the status are also integrated into the spreadsheet here as well.

4. REST API

All Data Avenue services are available programmatically as well through its REST (Representational State Transfer) API from program codes and shell scripts.

The REST API is available at URL https://dataavenue-host/dataavenue/rest/,where "dataavenue-host" is the domain name (or IP address) of the host where Data Avenue has been deployed. Table 1 summarizes REST "resources" (directory, file, attributes, transfer) and the valid HTTP methods (GET/POST/PUT/DELETE) applicable to them, along with the functionality they correspond to.

According to the table, for example, to create a directory we need to send a POST HTTP request to URL https://dataavenue-host/dataavenue/rest/directory; to delete a file we send DELETE HTTP request to URL https://dataavenue-host/dataavenue/rest/file, respectively.

Data Avenue refers to remote files residing on remote storages using URIs (Uniform Resource Identifiers). URIs are of the form: protocol://storage-address/path/file; directory-type URIs end with / symbol. The remote file or directory on which an operation is to be

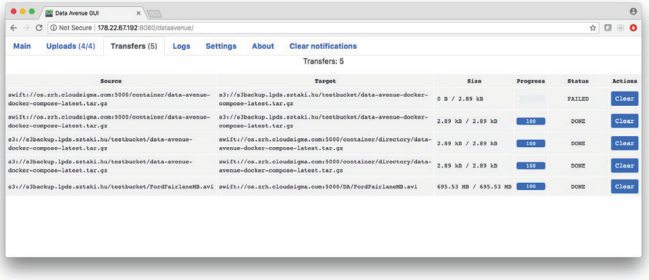

Figure 7: DA GUI transfer

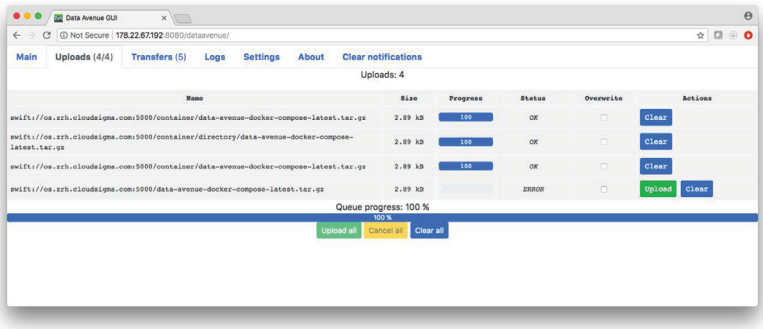

Figure 8: DA GUI upload

Resources	GET	POST	PUT	DELETE
directory	list directory entries	create directory	_	delete directory (recursively)
file	download file contents	upload file (new)	upload file (overwrite)	delete file
attributes	get file or directo-ry attributes	_	modify attributes	get attributes of multiple items
transfer	transfer status	new transfer	-	abort transfer

performed is specified by the ”x-uri” header field. For example, ”x-uri: s3://aws.amazon.com/mytestbucket/myfile.dat” refers to an object myfile.dat in bucket mytestbucket on Amazon S3. Note that Data Avenue presents uniformly different storages to the users regardless of that directories on the target storage are actually buckets (S3), containers (Swift), or sub-directories (SFTP). Currently, protocol string can be one of: http://, https://, sftp://, swift://, s3://, gsiftp://, srm://, irods://, or lfn://.

To access Data Avenue REST API each HTTP call is required to pass a valid access key (also known as ”ticket” or password) as HTTP header field. For example, in header ”x-key: 123e4567-e89b-12d3-a456-426655440000” the value 123e... corresponds to the access key issued to the user.

Credentials that are required by Data Avenue to authenticate to the remote storage determined by x-uri are passed using the ”x-credentials” header field with a value containing a JSON string. For example, ”x-credentials: f Type: UserPass, UserID: accesskey, UserPass: secretkey g” uses ”UserPass”-type authentication for the storage specified by x-uri, with username: ”accesskey” and password: ”secretkey”. Note that different storage types may require different authentication types and fields, so authentication fields may vary correspondingly.

To start a data transfer the x-uri header specified the source file or directory and x-credentials contains credentials for source storage. The target storage (and path) and target credentials are specified in the HTTP request body in JSON format. For example, request body: f target:s3://aws.amazon.com/mytargetbucket/, overwrite: true, credentials:

f Type: UserPass, UserID: targetaccesskey, UserPass: targetsecretkey gg" specifies "targetbucket" as the destination folder, and "targetaccesskey" and "targetsecretkey" as access key and secret key for authentication. As a result, the transfer-start request (POST) returns a "transfer identifier". Sending GET to resource transfer with "path parameter" (postfix added to resource name in the URL) corresponding to this identifier returns the actual transfer status in JSON format. The response contains details such as transfer status (done, running, failed), bytes transferred, date started, etc.

5. Using Data Avenue to move large data sets from different clouds

One of our main targeted user groups is the Hungarian academic research community and their cloud, called MTA Cloud [7]. The MTA Cloud was founded in 2014, when the Wigner Data Center and the Institute for Computer Science and Control (MTA SZTAKI) collaborated to establish a community Cloud for the member institutes of the Hungarian Academy of Sciences. MTA Cloud has currently 56 projects from 17 research institutes including among others the Institute for Nuclear Research, the Research Centre for Astronomy and Earth Sciences and other academic and research institutes in other joint projects e.g. University of Szeged and Eszterhazy Karoly University of Applied Sciences. OpenStack and Docker container based cloud infrastructure combines resources from Wigner and MTA SZTAKI relying on the nationwide academic Internet backbone [8] which has 10 Gigabit network and other federated services, e.g. eduGain and HEXXA for authentication and authorisation. The overall capacity is 1304 virtualized CPU with 4,7 TB memory and 574,75 TB storage facility. Moreover, the expansion in 2017 enabled the use of GPU cards, which can be utilized for parallel and computational scientific applications.

Utilizing (among others) the elasticity, security solutions and easy access offered by cloud computing, changes the way how scientists store the data for their research or the result of their research work. The reproduction of the results of the research is an important aspect of publishing. Therefore, long-term storage of data is a key value, thus, cloud storage usage continuously emerging in the last few years. It is a fact, that cloud providers do not support users taking their data to another cloud storage, they do not provide tools for this purpose. Although, researchers can have difficulties, when they have to move their data to a slower, but at the same time a cheaper private or academic cloud, in case of changed financial circumstances, in order to continue their research.

Data Avenue can be utilized in many ways. We have selected from this set, the deployment of a personal Data Avenue service in an academic cloud, since data movement from public clouds is a relevant issue here.

Fig. 9 shows the overall architecture of this use case. In the first scenario, a Data Avenue service and an S3 storage is established within the MTA academic cloud. With the help of Data Avenue service, all of the important files can be transfered from a public cloud, such as Amazon cloud, to the local S3 storage which is running in MTA Cloud. In the second scenario, Data Avenue service can be utilized by transferring data from the local S3 storage to the virtual machine which processes data.

End-users can establish a Data Avenue service in the cloud either using Docker [9] or automatically with an orchestration tool. Optionally, end-users can use the Data Avenue service which is running in SZTAKI, and operate with the GUI of the Data Avenue [10],

instead of establishing their own Data Avenue service. Data Avenue can move data across clouds, with different type of API access, if they are publicly available.

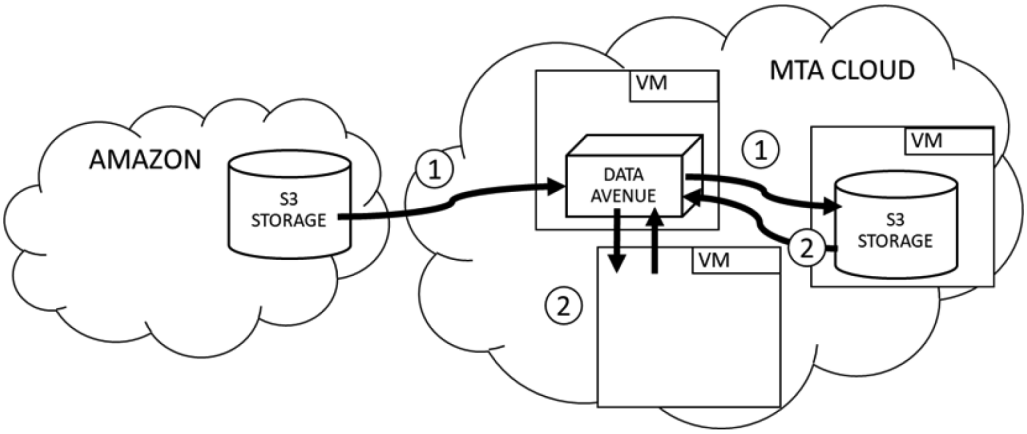

Figure 9: Use case architecture.

The use of Data Avenue service is available for MTA Cloud users. Although, MTA Cloud has no S3 storage available, therefore, users should make an S3 storage for their own, using volumes, which can be attached to a running virtual machine. There is an opportunity to request volumes in clouds, which can be attached to a running virtual machine. End-users can use attached volumes in many different ways, we recommend the Ceph [11] storage solution in order to have a full value private S3 storage within our project. Ceph is an open-source storage system, designed to provide highly scalable object-, block- and file-based storage under a unified storage cluster. It uses CRUSH (Controlled Replication Under Scalable Hashing) algorithm to ensure data distribution across the whole cluster, so all of the nodes within the cluster can retrieve data quickly, and in addition, it eliminates the centralized network bottleneck. While delivering high-performance and extraordinary data scalability (thousands of client hosts or KVMs accessing to exabytes of data), it replicates and rebalances data within the cluster dynamically.

After setting the configuration of the ceph network which is installed on a virtual machine, credentials of the storage node, Data Avenue can start moving all of the important data from the outer clouds (e.g. Amazon [12]) storage to the established S3 storage within the MTA Cloud. After a completed data transfer, virtual machines can use the new S3 storage as a full value storage, containing exactly the same data, which were on the Amazon cloud. Once the expensive computations are ready in the academic cloud, there is an opportunity for transferring back the data of the results to a public cloud using again Data Avenue. After the data migration, the data can be accessible for the researchers within the MTA Cloud, using Data Avenue service, and our new Ceph based, S3 storage.

The advantages of this solution are as follows. It is cloud independent, therefore it can be established in many different type of clouds (private, public, hybrid, academic). Furthermore, Data Avenue can communicate between several types of cloud storage

types, therefore it can perform data transfer between two different types of cloud and cloud storage flawlessly.

6. Performance measurements

For all the measurements the OpenStack-based (Mitaka) MTA Cloud was used as the target cloud. Fig. 10 and Fig. 15 show the architecture, which was used for testing Data Avenue. The architecture in the MTA Cloud consists of two node types: Data Avenue and storage node. For measuring upload and download performance the architecture shown in Fig. 10 was used, where there was only one S3 storage was used. The second, architecture in Fig. 15 shows the architecture, which was used during measuring the transfer rate performance of Data Avenue, where two S3 storage node is needed.

On the Data Avenue node, the Data Avenue application is running, and on the storage nodes, an S3 ceph storage is running. The virtual machines had "ml .medium" flavour (2 VCPU, 4GB RAM, 40 GB disk), based on Ubuntu 16.04 OS images with cloud-init support, and 2048 GB volume was attached to each S3 storage node. Docker [9] and Ceph [11] (version 10.2.10) was used to build up the components. In order to make Data Avenue able to manage large files as well (file size >50GB), we used 3GB memory heap for Apache Tomcat 7 [13] on the Data Avenue node.

6.1. Data Avenue-mediated upload/download transfer rate vs. object size granularity

In this section we examine data upload and download functionality provided by Data Avenue from two different perspectives:

- What is the overhead of transferring data through the intermediate node (Data Avenue) instead of direct connection between the client and the storage?
- Does the granularity of the transferred data (file sizes) affect the transfer rate?

As described earlier, Data Avenue offers a uniform interface for clients, which is accessible over plain HTTP (or secure HTTP protocols), thus, it can be used by simple command-line tools, like curl or wget, whereas, forwarding clients data to the storage, or vice versa, over the proprietary protocol, is done by Data Avenue. Clearly, this mediation service has performance overhead for three reasons: a) the time to transfer a single file includes the connection establishment cost between the client and Data Avenue host in contrast to directly connecting to the storage, b) data are not directly transferred between the client and the storage but through an intermediate node (Data Avenue), c) the connection between the client and Data Avenue is single-threaded (data are passed over a single HTTP connection) while proprietary storage connectivity tools (such as AWS CLI) might use multiple threads to download/upload data. This overhead applies to each individual transfer (REST API is stateless, meaning that Data Avenue does not re-use storage connections from previous transfers). Reason a) involves a (slight) constant penalty for each transfer, compared to direct connection, which might be significant when passing files with small size, but less relevant at longer overall transfer time; reasons b) and c) imply proportional overhead during the whole transfer.

To compare rates of direct and Data Avenue-mediated data transfers, a measurement to transfer files of different sizes, using both Data Avenue and a proprietary tool (AWS CLI) while using S3 storage was designed. Fig. 10 depicts this configuration. We generated files with random content with the fohowing sizes: 1 MB, 10 MB, 100 MB, 1 GB, 10 GB, 100 GB. These are first uploaded to the S3 storage, then downloaded. We

measured the transfer time with the total of 100GB data of each file size, i.e. 1 MB file had been uploaded 100000 times, 10 GB ten times, one after the other, respectively. Each experiment of transferring 100 GB file in different data packages was executed ten times to calculate average transfer time from ten measurements. From these results we derived the transfer rate value (MB/s) for each granularity, file size. We used curl [4] (version 7.47.0) to communicate with Data Avenue REST API service, AWS CLI (version 1.14.7) in order to connect to the S3 storage directly. In our experiments all the hosts: S3 storage, Data Avenue server, client machine, were virtual machines resided in the same cloud (MTA Cloud). As S3 storage Ceph-Rados gateway was used, which was deployed in a Docker container, without HTTPS (plain HTTP was used to eliminate data encryption overhead during this measurement in both direct and indirect connections). Download rate includes the time of saving the received data to local disk to reflect real-world usage; similarly, at upload, the file to be transferred is read from local disk.

The charts below show the transfer rates during upload (Fig. 11) and download (Fig. 13) at different granularities. Series DA in the chart show transfer rate values through Data Avenue, series AWS represent values at using AWS CLI.

During upload, we expected slower overall transfer rate values for all file sizes, which came true in file sizes above 10 MB; deceleration is at about 23% in this range. However, in the case of smaller files (1-10 MB), the transfer through Data Avenue proved to be even faster up to 3 times than direct transfer. Fig. 12 shows the proportion of of the different transfer rates as bars at different granularities. The reason could be that Data Avenue chooses different upload methods depending on the size of the object, as it is show in table 2, and Data Avenue uses the Java implementation of AWS SDK, while AWS CLI uses python.

During download, similar relation was experienced. In the range of file sizes above at about 100 MB direct transfer (AWS) exceeded Data Avenue (DA) transfers by 39%. At smaller file granularity, Data Avenue was more efficient, eventually by almost 4 times due to the reasons in case of upload, as shown in Fig. 14.

In this chart, data series ”DA redirect” represent transfer rates using a special (X-Accept-Redirects) option of Data Avenue (applicable only in the case of S3 download).

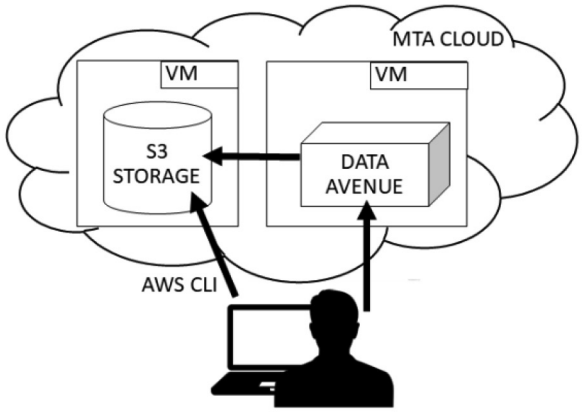

Figure 10: Uploading files using Data Avenue

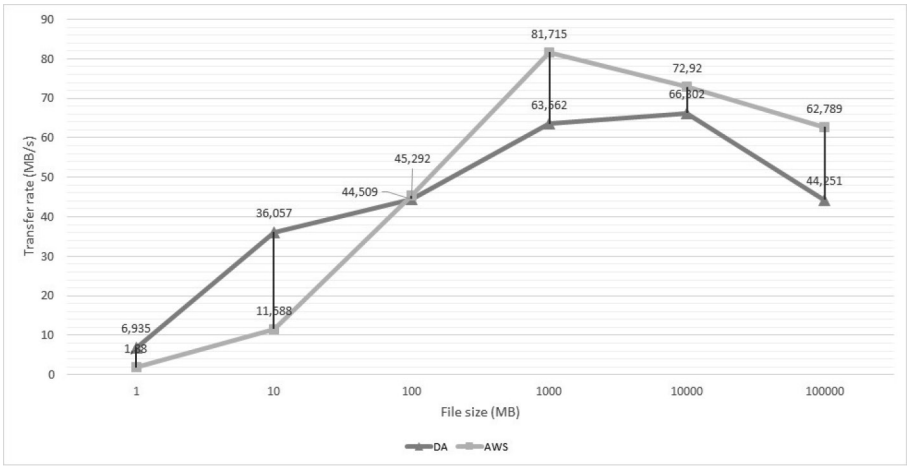

Figure 11: Average upload transfer rate vs. object size granularity

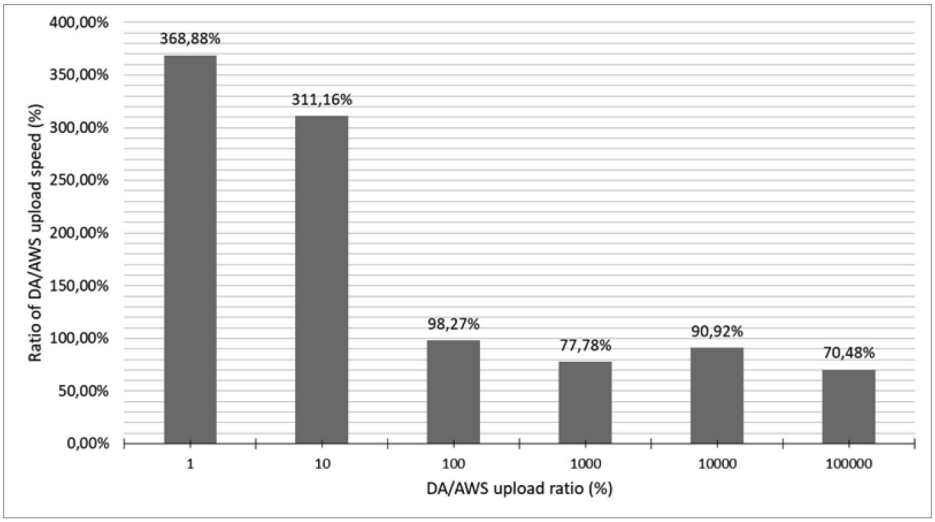

Figure 12: Ratio of DA/AWS upload rate at different granularities

TABLE 2: S3 upload method depending on object size.

Object size	S3 upload method used by Data Avenue
B - 10 MB 50 GB - 100 GB 100 GB - 500GB 500 GB -1 TB TB - 5 TB / unknown size	HTTP PUT (AWS SDK PutObjectRequest) Multipart upload (part size: 10 MiB, threads: 4) Multipart upload (part size: 50 MiB, threads: 2) Multipart upload (part size: 100 MiB, threads: 2) Multipart upload (part size: 500 MiB, threads: 2)

Using this option, Data Avenue only creates a pre-signed URL for the object to be downloaded and immediately redirects the client to the pre-signed URL pointing to the S3 storage (direct connection will then be established), so from now on no data is transferred through the Data Avenue host. (When clients are allowed to connect directly to outer storages, this option is useful; in private networks, where access to Internet is restricted and bridging through Data Avenue is possible, this option will not work.) Using DA redirect option can further increase the throughput compared to through-DA option, as shown in the figure.

6.2. Migrating large-size data between different storages using Data Avenue

In this section, we investigate data transfer performance of Data Avenue during which

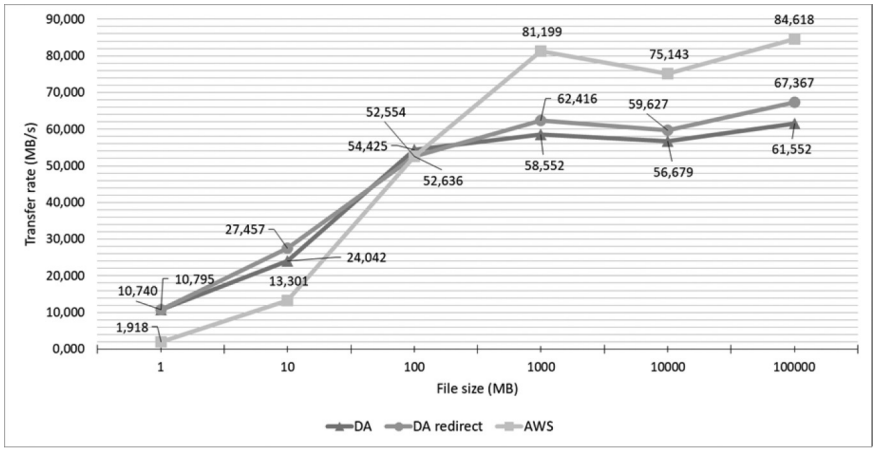

Figure 13: Average download transfer rate vs. object size granularity

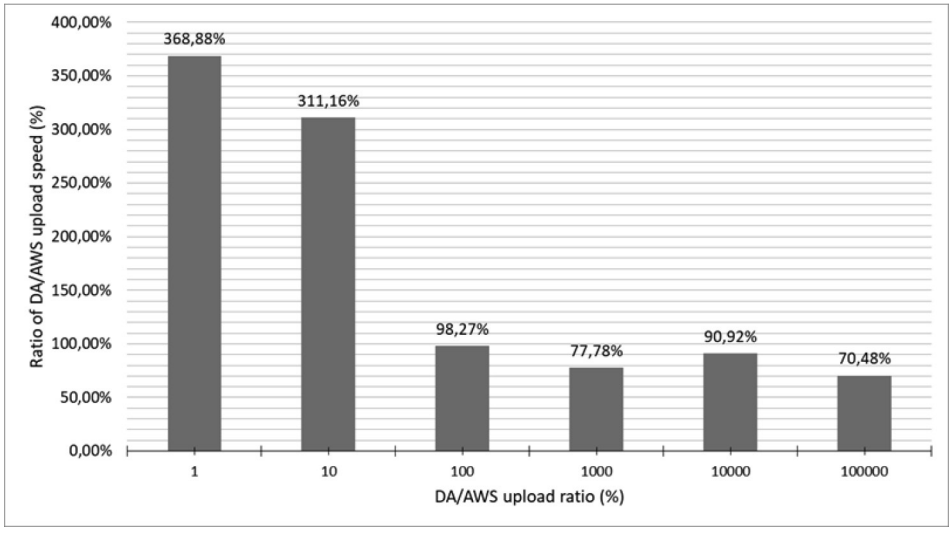

Figure 14: Ratio of DA/AWS download rate at different granularities

we measure the transfer of large amount of data between different storage resources (without having clients machine involved in the transfer). The question we would like to answer is how the total transfer rate relates to the output rate of the source storage and the input rate of the target storage.

The measurement was conveyed by transferring whole buckets from one storage to the other, using Data Avenues copy function. Each bucket contained a total of 100 GBs of data, but the contents of the different buckets composed of objects of different sizes: 1MB, 10MB, 100MB, 1GB, 10GB, 100GB, respectively.

In the this experiment, we transferred data between two storages, both resided in the

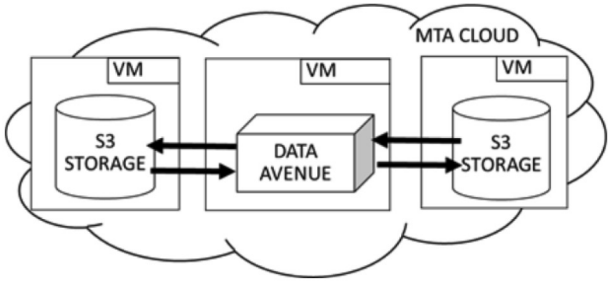

Figure 15: Architecture for testing data transfer in MTA Cloud

same cloud (MTA Cloud), as shown in Fig. 15.

The results of the measurements are shown in Fig. 16, in which we indicated the transfer rate. Fig. 16 shows the ratio of DA and AWS transfer rates. In the case of AWS, the transfer rate is calculated based on the download and upload transfer rates presented in the previous section, and we assumed the copy with AWS CLI is done by first downloading a given file and then uploading it to the target storage. (AWS CLI cannot copy between two storages, therefore this sequential model is applied.) In the case of Data Avenue, download and upload is done in parallel. This reason can explain, why we obtained better results at all granularity. The speedup is significant in the case of smaller files (similarly to uploads and downloads).

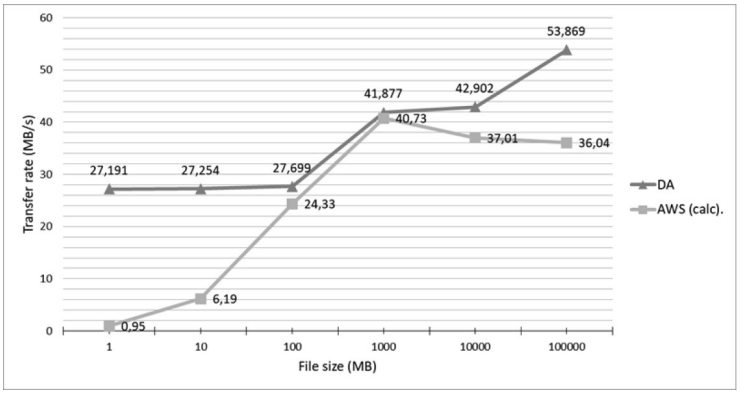

Figure 16: Average copy transfer rate vs. object size granularity

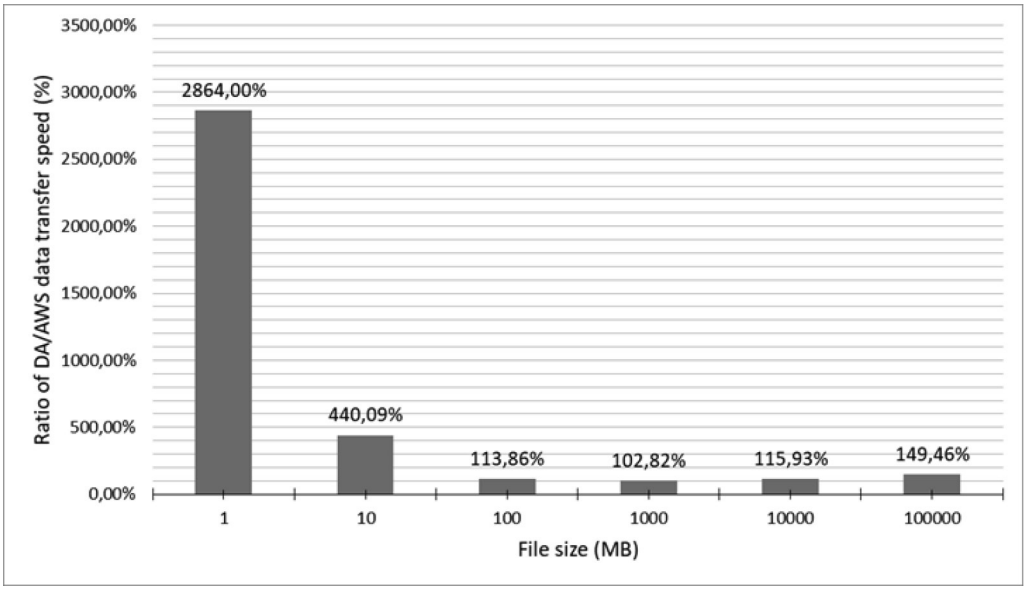

Figure 17: Ratio of DA/AWS data transfer rate at different granularities

7. Related work

Numerous commercial tools and services are available to manage data migration, such as Mover, CloudFuze, Cloudsfer, CloudFASTPATH, Veritas, Comm Vault, Mult-Cloud . They cover most of the popular cloud storage technologies and providers, including Amazon S3, Microsoft Azure, Google Drive, Google Cloud Storage, Drop- Box, OneDrive, Mega, Egnyte, to name a few, allowing to manage storages individually or transfer data between any two of them. Most services ensure data security (end-to-end encryption), reliable data transfer (failover), scheduled tasks (e.g. daily backups), data synchronization (incremental transfers). Most of them are based on a central service (web portal) through which registered users can perform their data migration tasks. CloudFuze, CloudFASTPATH, and Veritas offer hosted service, which allows to deploy the software on premise. However, none of these tools support access to legacy storages such as GridFTP, SRM, iRODS, or open source cloud storage solutions such as OpenStack Swift, which are supported by Data Avenue.

Globus [14] is also a commercial tool which supports GridFTP storages, and now can connect to S3-compliant cloud storages and Google Drive. Globus (as well as Data Avenue) allows of ''third-party" transfers between GridFTP sites, in which case, the data is exchanged directly between the two storages. Data Avenue also use server-side transfers on S3 when copying data between regions or within the same storage.

To our knowledge, only CloudFASTPATH and Globus provide API, which would allow to fetch data from program codes, or upload large result sets back to these cloud storages. Through Data Avenue's REST or SOAP API, all data management operations are possible, which opens the possibility to automate data access from either scripts, workflow systems, and other program codes. Data Avenue is a hosted application, which

can thus support data processing considering data locality, i.e. it can be deployed next to the storage and performing data bridging to application codes or other supported cloud storages if needed. Data Avenue also enables multi-user usage, and it is possible to setup distributed configurations, or to scale on-demand, depending on the customer needs.

CERN's File Transfer Service is a data movement service aims at reliably copying data between different GridFTP storages, which uses third-party copy. FTS does not support cloud storages.

Generic Storage Service (GSS) [15] was developed within the CloudFlow Infrastructure, which provides a technology platform for cloud based workflows in order to support different cloud storage accesses. Basic functionalities such as listing files, creating folders are made directly through the SOAP API. Data transfer between two storages is not supported, and in contrast to Data Avenue, it supports only OpenStack Swift and dedicated HPC storages.

Among the free tools we can only mention DragonDisk14, which can connect to S3-compliant storages and providers including Google Cloud Storage, GreenQloud. DragonDisk can be used to manage data on such storages (upload, download, create buckets, directories, delete, rename, etc.), to copy or synchronize data between different connected storages. Cyberduck15 and Transmit16 are similar desktop applications (commercial). They support most of the popular cloud storage providers Amazon S3, Google Cloud Storage, DropBox, Microsoft Azure, Blackblaze, to mention a few. These are desktop applications, which however cannot be used as a service (to perform long running background transfers) in contrast to Data Avenue.

8. Conclusions

Transferring large data sets among storages is an important task both for scientific applications and in commercial usage. The paper describes a potential solution called Data Avenue. This tool is offered and used in MTA Cloud which is the cloud of the researchers of the Hungarian Academy of Sciences. It is also used for commercial applications in the EU H2020 project Cloudifacturing. Data Avenue has been further developed in the EU H2020 project COLA [16] by MTA SZTAKI and CloudSME UG in order to produce its commercial version.

Data Avenue represents a very flexible solution for data migration in many respects. It provides several configuration possibilities by which both personal usage and organizational and global service can be set up. Its built-in security mechanisms also support all these configuration and usage possibilities. The GUI enables the easy use for humans and the REST API supports the usage by software systems like work- flows. Although Data Avenue was originally developed for grid and cloud systems due to the recent further developments, it is now a good candidate to transfer data among various cloud systems, too. Its Amazon S3 and OpenStack Swift plugins can cover a very large set of academic and public clouds. The recently developed Google Cloud Storage and Microsoft Azure plugins are in test phase and soon will be available for extending the set of clouds that can be served by Data Avenue.

Interested readers can try Data Avenue in several ways according to the different usage modes described in Section 2. Public service deployment can be found at Data

Avenue website. This file transfer service is freely available for everyone based on the GUI of a previous release but in principle very similar to the latest GUI described in this paper. The personal server host deployment can be tried based on the tutorial available on the Occopus [17] web page [18]. This can be tried by anyone who has access to a cloud system. Finally, a similar tutorial is available for the Hungarian academic researchers on the MTA Cloud web page in Hungarian.

Acknowledgement

This work was partially funded by the European COLA - Cloud Orchestration at the Level of Application project under grant No. 731574 (H2020-ICT-2016-1). On behalf of the Data Avenue project we thank for the usage of MTA Cloud that significantly helped us achieving the results published in this paper.

References

[1]. Allcock, W. (2003) GridLTP: Protocol Extensions to FTP for the Grid, Global Grid ForumGFD-R-P.020.

[2]. Shoshani, A. (2002) Storage Resource Management, GGF-4. Retrieved from: https://sdm.lbl.gov/srm-wg/doc/02.02.srm.joint.design/index.htm

[3]. Lemaitre, S., Frohner, A., Baud, J.P., Smith, D., Nienartowicz, K., Abadie, L., Mollon, R. (2007) Recent developments in LFC. CHEP07.

[4]. Hajnal, A., Marton, I., Farkas, Z., Kacsuk, P., Remote storage management in science gateways via data bridging, Concurrency and Computation: Practice and Experience, 27 (16)., 4398-4411.

[5]. Kacsuk, P, Farkas, Z., Kozlovszky, M., Herman, G., Balasko, A., Karoczkai, K., Marton, I. (2012) WS-PGRADE/gUSE generic DCI gateway framework for a large variety of user communities", Journal of Grid Computing, 10(4).

[6]. Kiss, T., Kacsuk, P., Kovacs, J., Rakoczi,B., Hajnal, A., Farkas, A., Gesmier, G., Terstyanszky, G. (2017) 'MiCADOMicroservice-based Cloud Application-level Dynamic Orchestrator, Future Generation Computer Systems.

[7]. MTA Cloud website https://cloud.mta.hu/

[8]. HBONE website https://www.niif.hu/en/hbone_hbone

[9]. Docker website https://www.docker.com/

[10]. Data Avenue website https://data-avenue.eu/

[11]. Ceph website https://ceph.com/

[12]. AWS website https://aws.amazon.com/

[13]. Apache Tomcat website https://tomcat.apache.org

[14]. A. William, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, and I. Foster, The Globus striped GridFTP framework and server, in proceedings of the 2005 ACM/IEEE conference on Supercomputing, IEEE Computer Society, p. 54, 2005.

[15]. Havard Heido Holm, Jon M. Hjelmervik, Volkan Gezer, ''CloudFlow-AnInfrastructureforEngineeringWorkowsintheCloud", UBICOMM 2016 : The 24 Tenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (2016)

[16]. COLA project website https://project-cola.eu/

[17]. Jozsef Kovacs, Peter Kacsuk, ''Occopus: a Multi-Cloud Orchestrator to Deploy

and Manage Complex Scientific Infrastructures", Journal of Grid Computing, Volume 16, Issue 1, pp 1937, 2018

[18]. Occopus website http://occopus.lpds.sztaki.hu/

Submitted 10.02.2018
Accepted 24.05.2018