# *Data Avenue: Remote Storage Management for Science Gateways*

# Contents

# Introduction

- Several scientific **communities** use **distributed computing infrastructures** for their compute-intensive problems, DCIs used in astronomy & astrophysics, biomedical sciences, earth sciences, agriculture, physics, etc.
- Such calculations would last extremely long to compute on desktop computers, parallelization and utilization of DCIs can drastically reduce the overall computation time
- Scientific applications can be designed as a workflow of tasks in a way that each task can be run in a different DCI, to further improve parallel processing
- There are tasks that can be run in specific DCIs, others can freely be spread across multiple DCIs. (? An example can be PS applications, where thousands of tasks are run on different parameters, where the more parallelization and more DCIs involved the less overall computation time.)

# Introduction

- There are **different types of DCIs** (clusters, supercomputers, grids, clouds). Submission of workflow tasks to different DCIs is not trivial. Desktop workflow systems should be able to submit jobs to various DCIs. There are science gateways that allow of job submission. As an example, WS-PGRADE/gUSE portal framework is capable of submitting jobs to several DCIs. This problem is solved.
- Jobs in workflows process data, consumes inputs and produces outputs. Staging of data should be solved. Moreover, as jobs must pass data to each other this transfer has to solved even if jobs run in different DCIs.

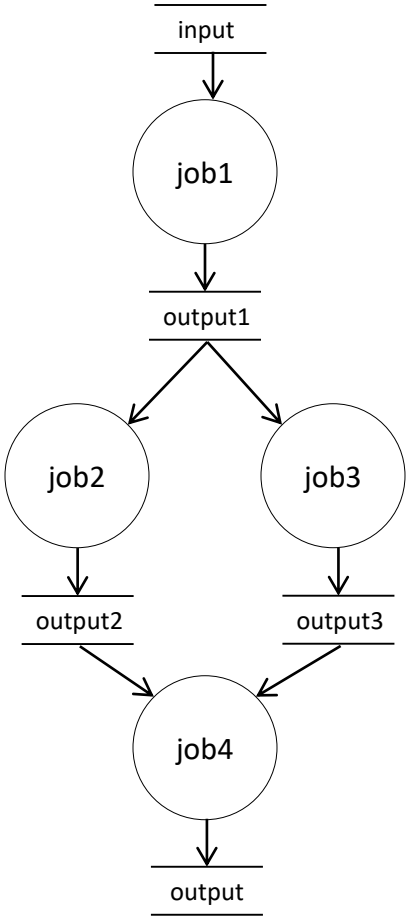
# Introduction

- D**iverse storage resources** are used in the different DCIs where inputs/outputs of scientific computations are stored
- Melyik dci-ban,milyen storage? Pl. gListe srm, Globus gridftp, cloud-okban s3
- It is not a trivial problem, how to transfer data from/to DCI storage by the workflow enactment system, neither to transfer data from DCI to another for a subsequent calculation
- This paper proposes a generic solution for this problem and shows a concrete implementation of the idea using Data Avenue. This could be used for many different workflow systems. As a concrete example we show how to use together with WS-PGRADE workflows.

# Data management problems in workflows

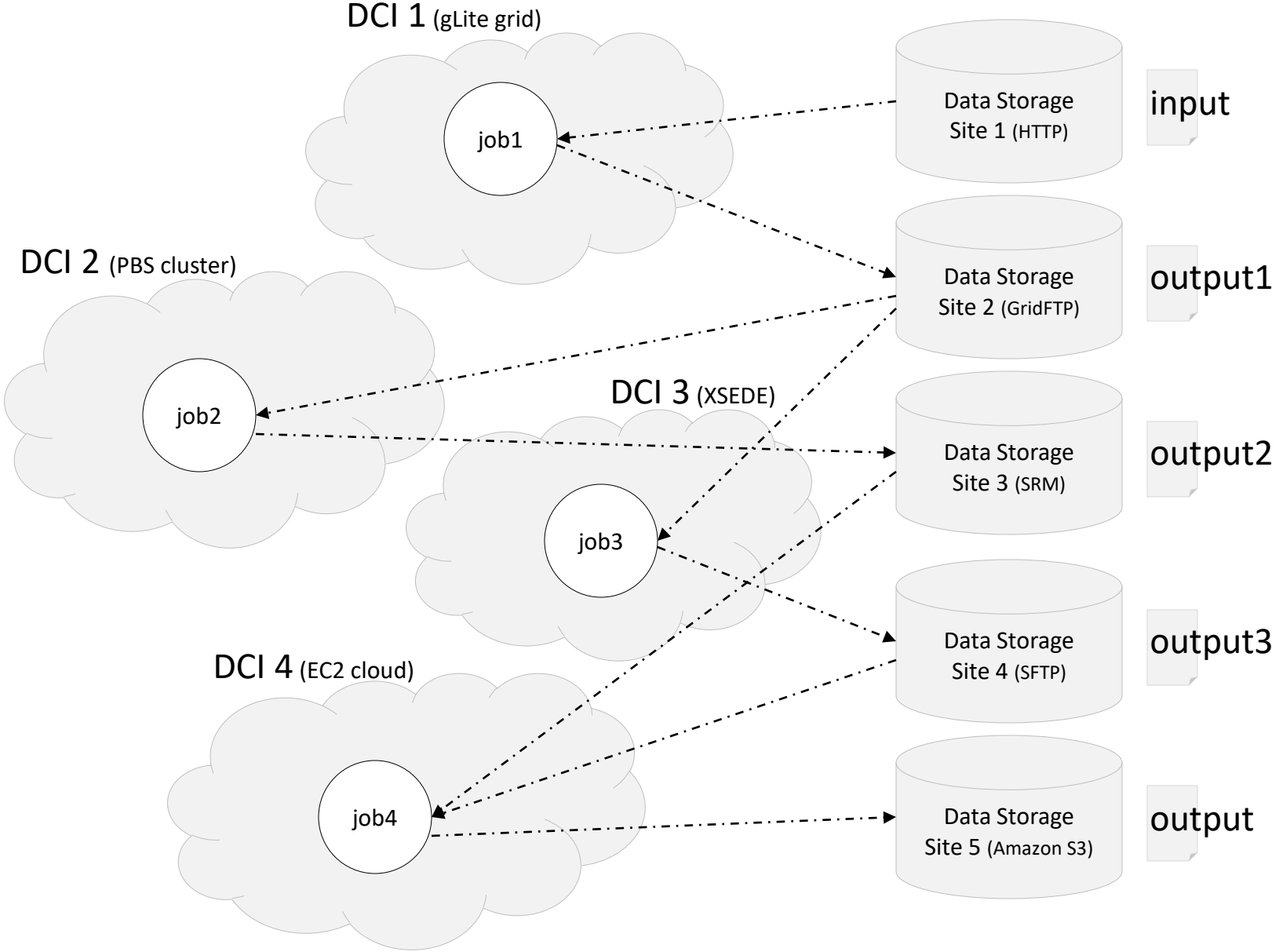
- Scientific applications can be designed as workflows
- Workflows can be formalized using directed acyclic graphs, where nodes represent jobs and edges represent (data-) dependencies between jobs
- Dependencies prescribe certain ordering how different tasks can be executed and allow parallelism (data-flow based execution), that is, all jobs having all constraints fulfilled can be performed simultaneously
- Jobs have inputs and outputs; we refer them to as simply files, but they can be constants, query results, etc.
- When data is passed between jobs these are called intermediate files
- Example:

# Data management problems in workflows



# Data management problems in workflows

- Workflow enactment systems (such as desktop workflow applications, e.g., [] or science gateways, e.g. gUSE) allow jobs to be run in different DCIs
- We specify where jobs to run during job configuration
- Potentially, each job can be configured to run in a different DCI
- Also, ideally, jobs' inputs and outputs can be configured to use different storages where inputs/outputs reside
- A fully flexible, ideal workflow system enables configuring the following workflow:


DCI 1 (gLite grid)
job1
DCI 2 (PBS cluster)
job2
DCI 3 (XSEDE)
job3
DCI 4 (EC2 cloud)
job4
Data Storage Site 1 (HTTP)
input
Data Storage Site 2 (GridFTP)
output1
Data Storage Site 3 (SRM)
output2
Data Storage Site 4 (SFTP)
output3
Data Storage Site 5 (Amazon S3)
output

# Data management problems in workflows

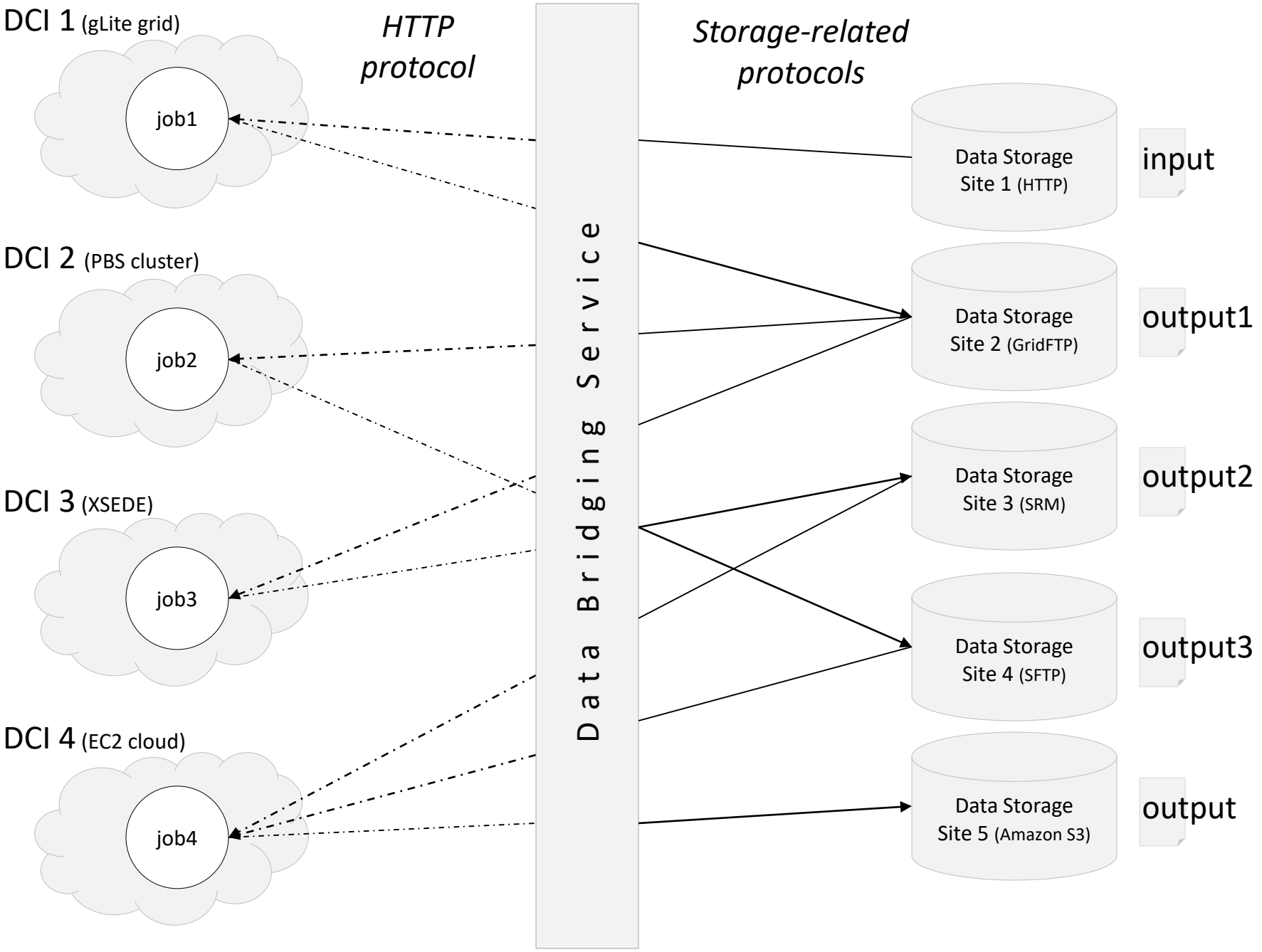
- However, although DCI configuration and job submission to different DCIs is a solved problem (e.g., WS-PGRADE supports 20 DCIs), configuring storages for job inputs/outputs has limitations
- For example, it is not possible to access cloud storages (e.g. Amazon S3) from GT DCIs (no API available on worker nodes), or conversely, jobs running in cloud VMs cannot access GridFTP storages used by some other DCIs
- Although there is possibility to set up the necessary APIs to DCI worker nodes or cloud images, in practice, system administrators do not allow to make modifications on the software stack of compute sites (for security reasons), and modification of cloud images used by VMs may also not be possible.
- Moreover, to provide this flexibility, all kinds of DCIs should be extended with capability to access all kinds of storages, which would unnecessarily increase the software stack on compute sites (which need to be installed, maintained, etc.).
- Last but not least, user credentials (required to access storages) are exposed to worker nodes (passwords, proxies have to be transferred to the compute sites)

# Data management problems in workflows

- Two basic approaches exist to enable data transfer between DCI worker nodes and storage resources (when WNs cannot access them natively):
  1. Workflow management system-managed staging: the WMS connects to storages and at each job submission it uploads/downloads data to DCI worker nodes through the appropriate manager node and API (e.g., GT, gLite)
  2. Pegasus Lite approach: it extends WN capabilities to access storages (separate staging jobs do the download) using an additionally installed software component called Pegasus Lite.
- Solution 1 can be very inefficient, as data first moved from the DCI to the host of the workflow management system (desktop/science gateway), then then it is uploaded to the storage, instead of direct connection between the WN and the storage. WMS host may also imply other limitations (disk/file size of VMS host).
- Solution 2 performs direct transfer between the worker node and the storage, but requires changing the software stack of the worker nodes, which is often not possible. In the latter case user credentials (for storages) are exposed to worker nodes, leaves the context of the WMS.
- TODOL EUDAT mit nyujt, web lap, workflow

# Data Management via Data Bridging

- This paper proposes a bridging solution to solve this problem, by introducing a data bridge between DCIs and storages that perform protocol conversion from storage-related protocols from/to HTTP
- Thus, to access storages computing elements only have to access the data bridge over HTTP protocol which DCI worker nodes are aware of
- whereas communication with the storages over the corresponding storage protocol and HTTP tunneling of data are managed by the data bridge


DCI 1 (gLite grid)
job1
DCI 2 (PBS cluster)
job2
DCI 3 (XSEDE)
job3
DCI 4 (EC2 cloud)
job4
HTTP protocol
Data Bridging Service
Storage-related protocols
Data Storage Site 1 (HTTP)
input
Data Storage Site 2 (GridFTP)
output1
Data Storage Site 3 (SRM)
output2
Data Storage Site 4 (SFTP)
output3
Data Storage Site 5 (Amazon S3)
output

# Data Management via Data Bridging

- Further advantage of this solution is that there is a single place (data bridge) where storage communication has to be solved
- As a result, it homogenizes storage access and reduces the problem of implementing all storage access in all DCIs (n^2) to n storages (2n)

# Implementation: Data Avenue

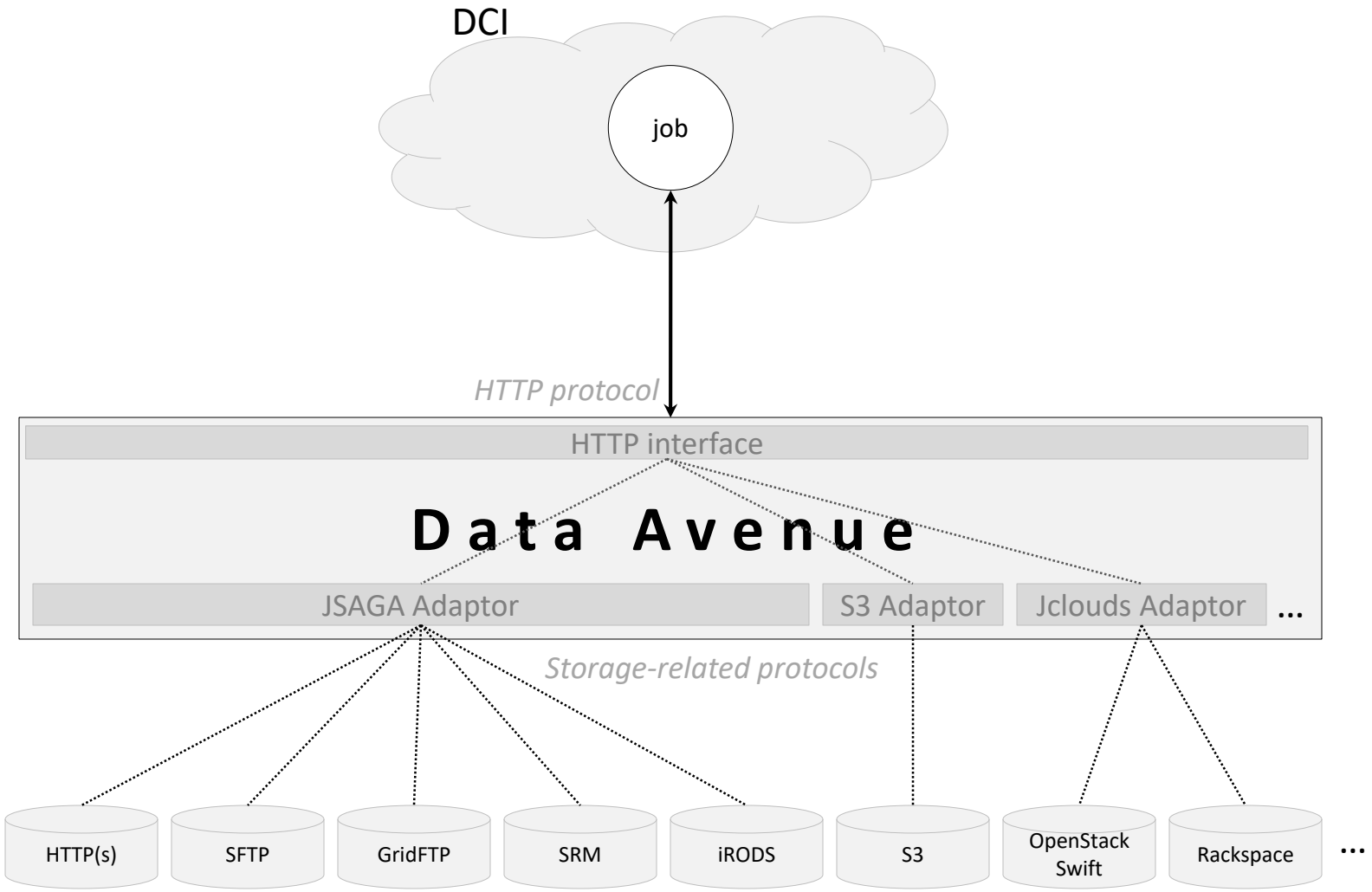
- A data bridge prototype implementation called Data Avenue has been developed
- It is open source, written in Java
- Besides data bridging capability (described later) it offers a set of operations to manage data on storages: list, mkdir, rmdir, rename, delete, permissions
- Data Avenue is a web application that offers web service interface (SOAP and REST)
- Accessible from program codes, Java (API provided) and also UI (portlet-level access to storages, shown later)
- Data Avenue is designed as an extendible plugin architecture, where different storage resources are managed by different plugins, called adaptors
- Adaptor implementations are based on abstraction libraries: JSAGA, Amazon S3 SDK, Jclouds, Google Drive (planned)

# Implementation: Data Avenue

- Data bridging is realized via so called “HTTP aliases”; an alias is an HTTP URL pointing to Data Avenue server that can be read or written over HTTP protocol
- First, an alias must be created for a remote file using Data Avenue operation “createAlias”, then the return URL can be read/written - for the specified period of time
- To create an alias, creator must specify the storage type, internet location, path and filename on the storage, and credentials required to access the storage by Data Avenue
- The returned HTTP URL a unique id (randomly generated UUID) associated with the remote file
- At accessing the HTTP URL Data Avenue performs the necessary protocol conversion (HTTP tunneling), this is transparent for the client that uses that URL

# Implementation: Data Avenue

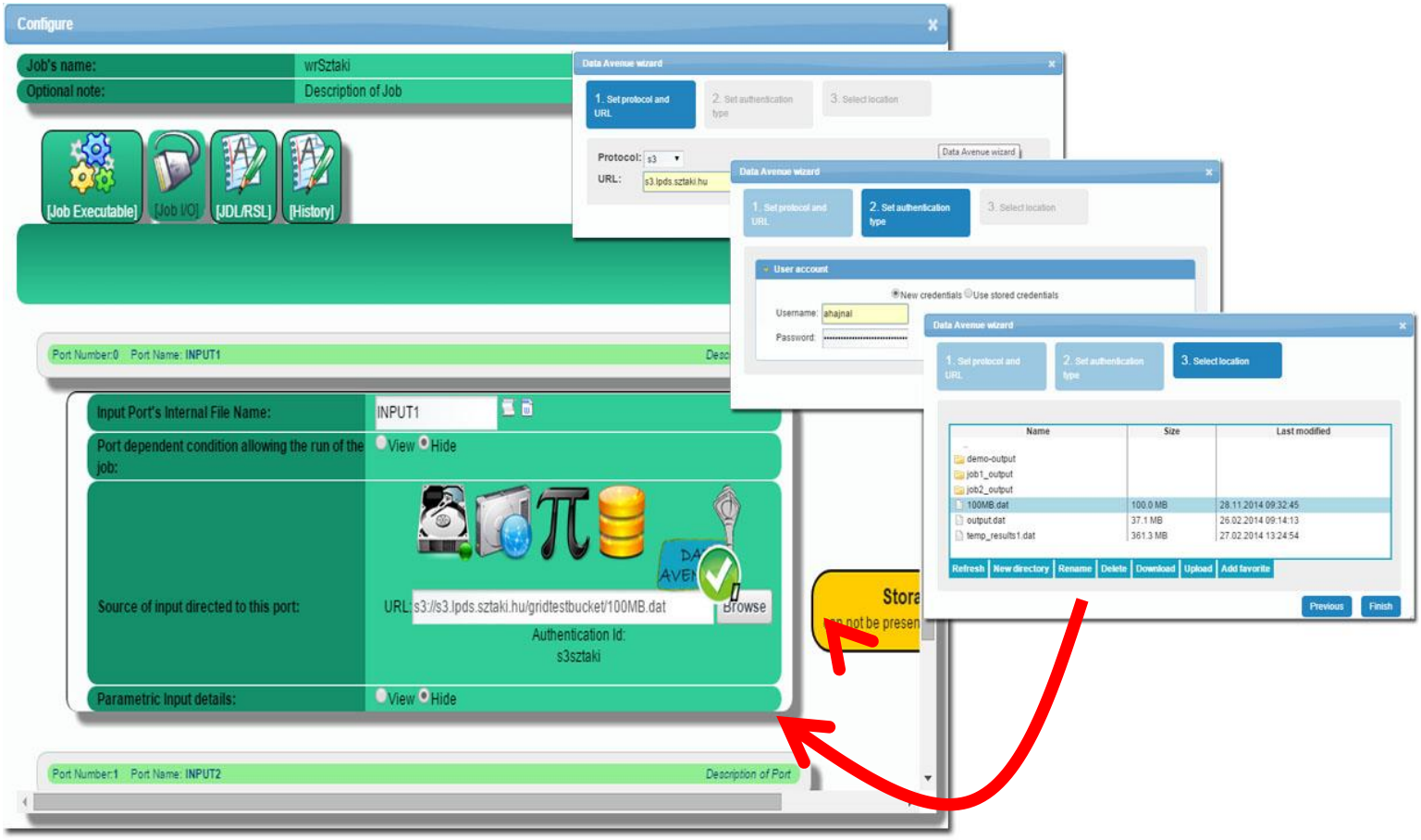
- For example, given a file /path/filename.txt on a GridFTP storage at gridftp-server.com accessible with an x509 proxy, Data Avenue creates an alias: https://data-avenue.eu/blacktop/20b419d9-d287-4a44-a23e-adde1ad455b3
- By reading this URL (HTTP GET), based on the id of the URL (d287-4a44-a23e-adde1ad455b3) Data Avenue can fetch access details of the remote resource (storage address, credentials), stored at alias creation, connect to the storage (gridftp-server.com), read the contents of the remote file (/path/filename.txt) and forward bytes to the client over HTTP
- Note that credentials are not revealed to the clients (worker nodes or other clients can access passwords/proxies used to connect to the storage)
- In the case of S3, it is also possible to create URLs directly pointing to storage resources, which HTTP URLs can be handled in the same way as aliases (see Evaluation section)
- Aliases can be created for any remote file on any storage that Data Avenue can handle, see Figure, in this way any job can access a large number of storage resources over HTTP


DCI
job
HTTP protocol
HTTP interface
Data Avenue
JSAGA Adaptor
S3 Adaptor
Jclouds Adaptor
...
Storage-related protocols
HTTP(s)
SFTP
GridFTP
SRM
iRODS
S3
OpenStack Swift
Rackspace
...

# Implementation: Data Avenue

- Data Avenue services has been made available from WS-PGRADE/gUSE portal framework
- It allows selecting remote inputs and outputs for jobs through Data Avenue services (configuration time)
- A wizard-like user interface enables users to browse storage resources and select inputs/outputs (see next figure)
- When executing workflows, aliases required to access input/output files by jobs are automatically created prior to job submission (runtime)
- Credentials to storages are managed by the portal as well, and delegated to Data Avenue sever only
- We note that Data Avenue is not only usable from WS-PGRADE but more generally applicable due to web service interface

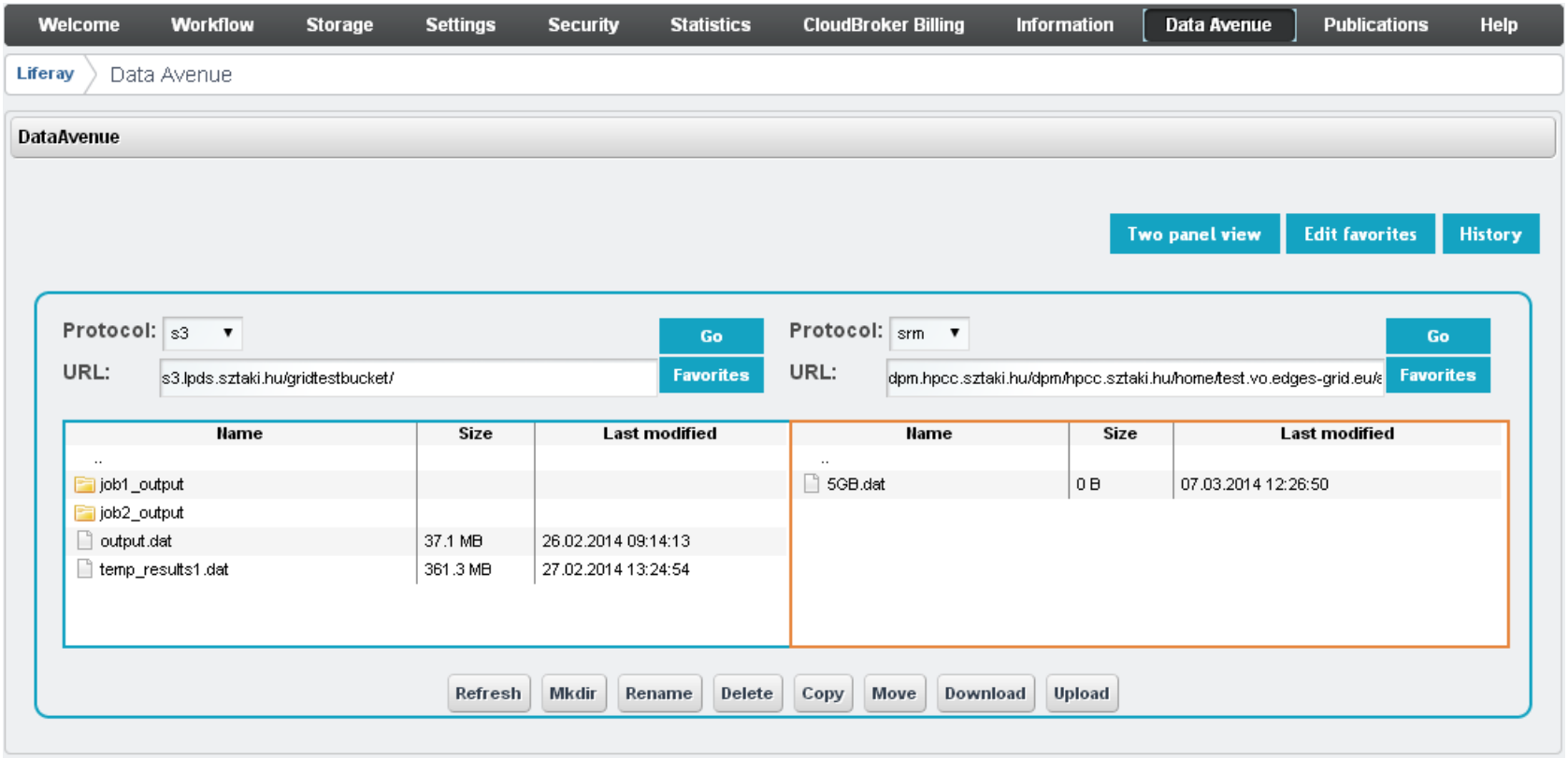
# Implementation: Data Avenue



# Implementation: Data Avenue

- GUI is also provided for end-users to access/organize data on storages, which is integrated into the portal (portlet-level access)
- Users don’t have to leave the portal environment to manage remote storages (install and use other tools)
- On the GUI users can initiate transfers between storages even of different types, which does not load the portal (Data Avenue managed transfers)

# Implementation: Data Avenue



# Implementation: Data Avenue

- A single Data Avenue instance with limited CPU and network capacity can be a bottleneck, even in the case of a single portal (scalability concerns)
- However, multiple data bridge components can be deployed with load balancing facility
- These instances can be started statically (a constant number of instances per portal, based on estimated load) or
- Even new instances can started dynamically on demand in a cloud on extreme loads
- This possibility solves scalability concerns

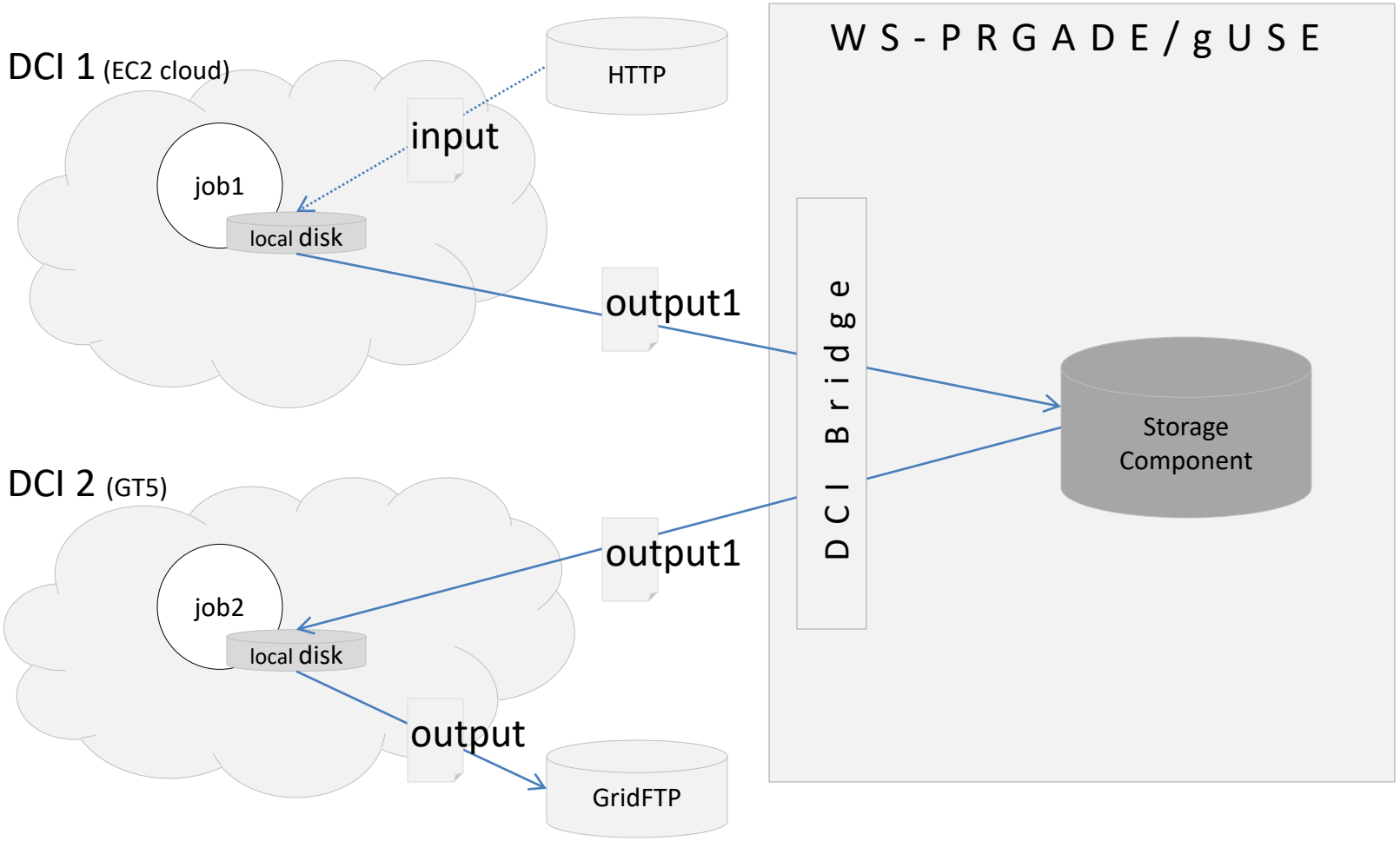
# 5. Evaluation

- Assume that we are given a workflow that converts VM image formats and calculates MD5 hashes
- The workflow has two tasks, source VM images can be downloaded from an HTTP server, and converted images have to be moved to a GridFTP storage
- The first task can only be run in Cloud (because of the required software, QEMU disk image utility), its input is on an HTTP server (say, in Virtual Disk Image format), and this task produces a large-size output (in raw format)
- The second task runs in GT5 infrastructure, and its goal is to perform computation on the output of task1 (i.e., compute MD5 hash of the raw image obtained in task1) and put/publish its outputs (raw image + MD5 hash) on a GridFTP server

# 5. Evaluation

- The first task cannot access any GridFTP server, so it cannot put its output to a GridFTP storage
- GT5 task cannot access storages other than GridFTP servers
- Therefore, this workflow can only rely on WS-PGRADE's (or other workflow enactment system's) internal storage to pass the intermediate file between the tasks
- WS-PGRADE performs transfer from cloud VM's local disk to gUSE internal storage, then from gUSE storage to GT5 WN's local disk
- Note: passing the intermediate file would happen in the same way even if both tasks run in the cloud

# 5. Evaluation



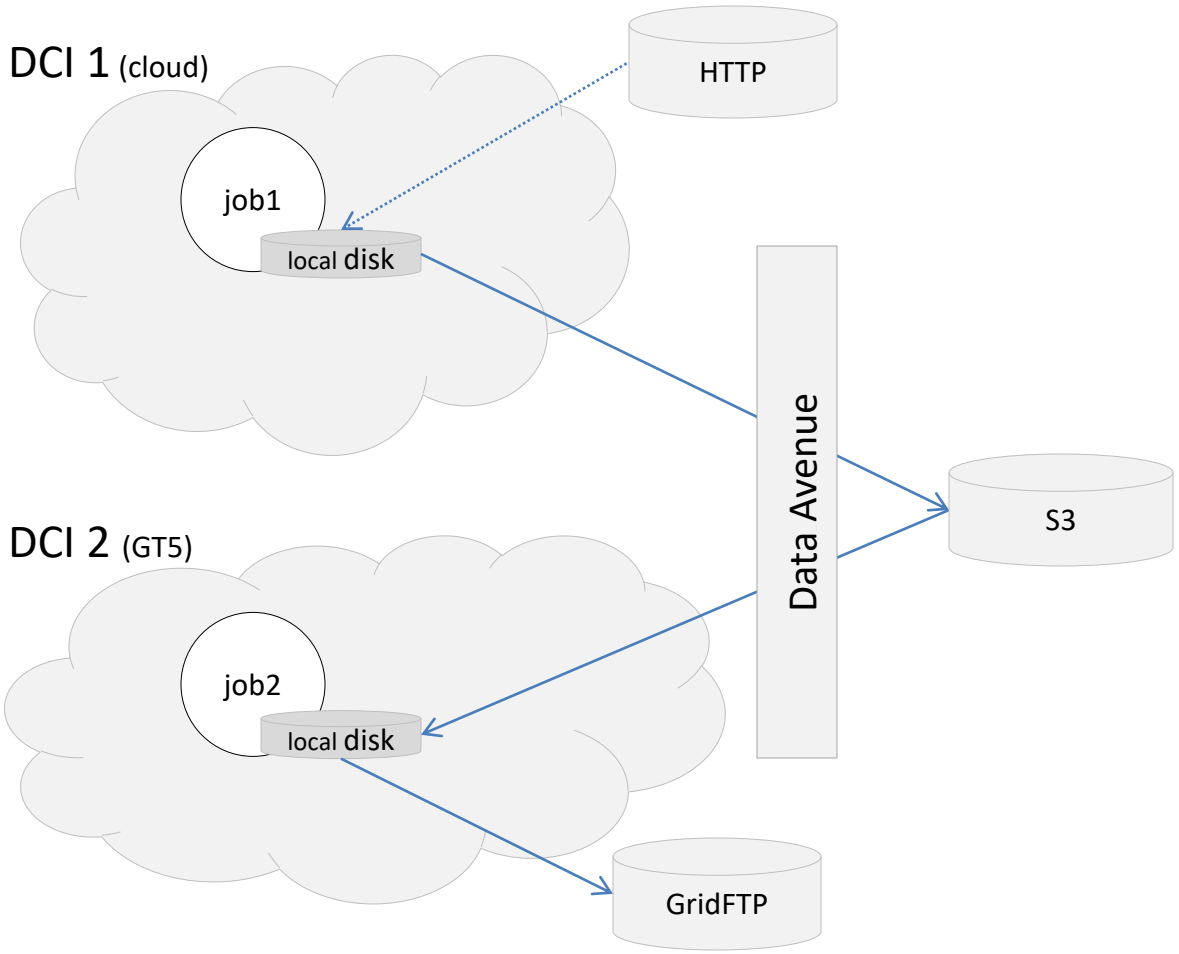
# 5. Evaluation

- However, gUSE storage component has a limited disk space
- As there are many users of the portal, there are quota limits
- All users of the portal uses the same storage component (concurrent load and size requirements)
- Storage is not highly optimized (security, scalability, sync/backup, efficiency, integrity)
- (Storage is typically behind firewall for security reasons, so the upload to GT5 local disk is done in two steps: first storage->DCI Bridge, then from DCI Bridge->GT5 local disk; which further delimits size by disk space of the DCI Bridge, and implies load on DCI Bridge to transfer data. This is the preferred way through plugins.)

# 5. Evaluation

- With Data Avenue it is possible to use output storage for task 1 which also serves as input storage for task 2
- This solution avoids use of gUSE internal storage component
- The type of the storage is up to the workflow developer
- The size of the intermediate file is now only limited by the selected storage component
- For example, we can use Amazon S3 for storing the intermediate file:

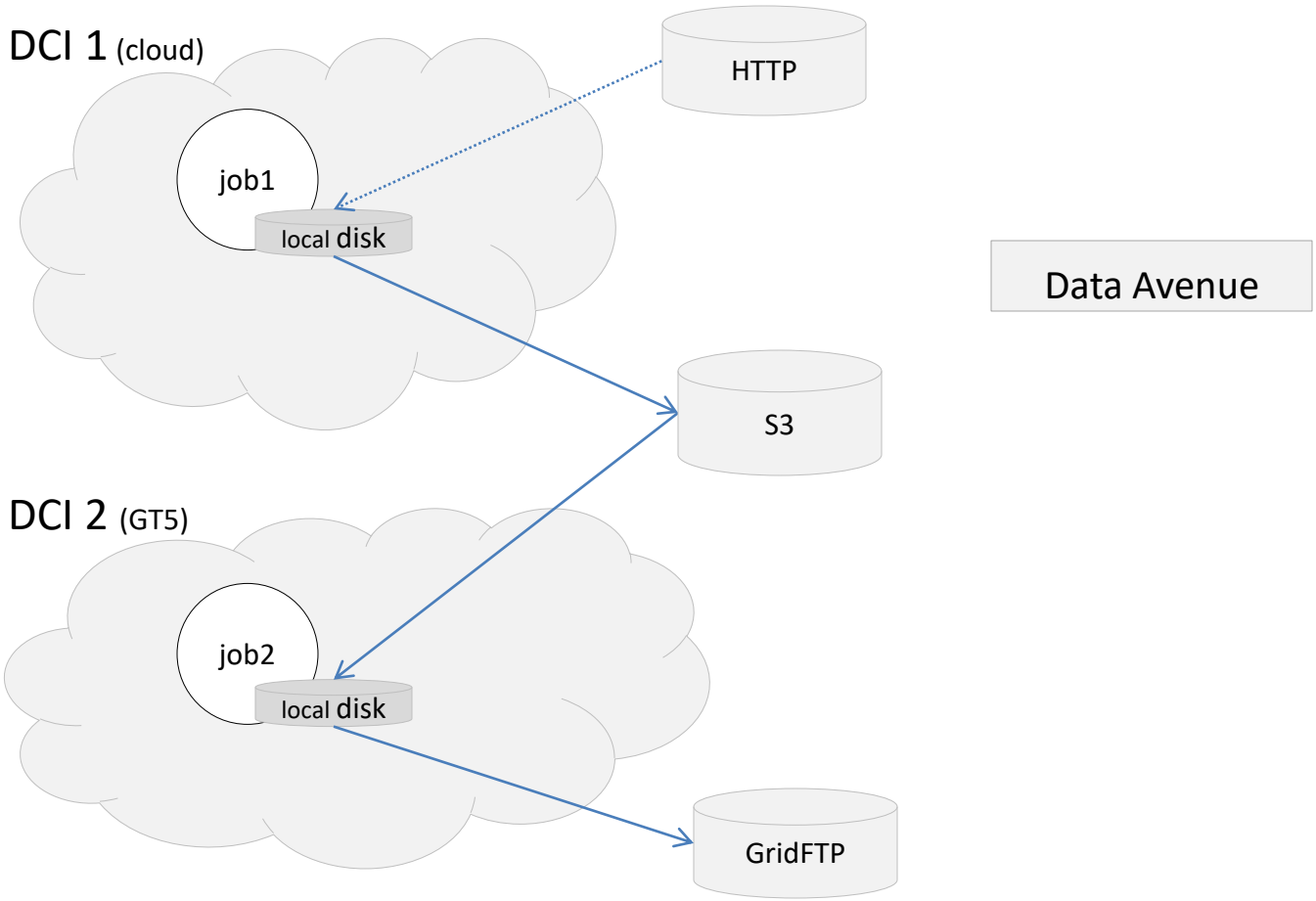
# 5. Evaluation



# 5. Evaluation

- Note: worker nodes in the different DCIs need to know HTTP only (e.g., may use tools wget or curl), no need to know how to handle S3 storages
- Data Avenue receives data from the worker nodes and forwards data towards the S3 server using the appropriate protocol
- WS-PGRADE create HTTP aliases only (less load on portal) and forwards these URLs at job submission
- As a result, workflow execution reduced by ? seconds (due to more efficient data transfer)
- In the case of S3 storages, it is also possible to avoid tunneling through Data Avenue, S3 protocol allows creating pre-signed URLs, which, from clients' point of view work in the same way as HTTP aliases (readable/writable over HTTP)
- Changing the URL what Data Avenue's createAliases operation returns from Data Avenue alias to S3 pre-signed URLs, the data transfer is even more efficient – without changing jobs' code
- It further reduces workflow execution time with ? seconds

# 5. Evaluation



# Related Work

- Kepler
- Taverna
- Pegasus (Lite)
- Galaxy
- Apache Ariavata
- EUDAT

# Conclusions

- Data bridging concept improves flexibility (now input sources can more freely be selected, outputs can more freely be chosen)
- Data bridging concept can improve efficiency (use of high-performance, sophisticated storages may outperform gUSE storages, removes limitation imposed by temporary storage capacity)
- It offers more portability of workflows (can run in other gateways with the same storage resources)