NSDF-Services: Integrating Networking, Storage, and Computing Services into a Testbed for Democratization of Data Delivery

Jakob Luettgau[1], Heberth Martinez[1], Paula Olaya[1], Giorgio Scorzelli[2], Glenn Tarcea[3], Jay Lofstead[4], Christine R. Kirkpatrick[5], Valerio Pascucci[2], Michela Taufer[1]
[1] University of Tennessee, Knoxville, [2] University of Utah, [3] University of Michigan Ann Arbor,
[4] Sandia National Laboratories, [5] University California San Diego

ABSTRACT

The lack of a readily accessible, tightly integrated data fabric connecting high-speed networking, storage, and computing services remains a critical barrier to the democratization of scientific discovery. To address this challenge, we are building National Science Data Fabric (NSDF), a holistic ecosystem to facilitate domain scientists in their daily research. NSDF comprises networking, storage, and computing services, as well as outreach initiatives. In this paper, we present a testbed integrating three services (i.e., networking, storage, and computing). We evaluate their performance. Specifically, we study the networking services and their throughput and latency with a focus on academic cloud providers; the storage services and their performance with a focus on data movement using file system mappers for both academic and commercial clouds; and computing orchestration services focusing on commercial cloud providers. We discuss NSDF's potential to increase scalability and usability as it decreases time-to-discovery across scientific domains.

CCS CONCEPTS

• **Information systems-Data management systems-Data structures**;

KEYWORDS

High-performance computing, Cloud computing, Data democratization, PerfSonar, XRootD

ACM Reference Format:
Jakob Luettgau[1], Heberth Martinez[1], Paula Olaya[1], Giorgio Scorzelli[2], Glenn Tarcea[3], Jay Lofstead[4], Christine R. Kirkpatrick[5], Valerio Pascucci[2], Michela Taufer[1] . 2023. NSDF-Services: Integrating Networking, Storage, and Computing Services into a Testbed for Democratization of Data Delivery. In *2023 IEEE/ACM 16th International Conference on Utility and Cloud Computing (UCC '23), December 4–7, 2023, Taormina (Messina), Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3603166.3632136

1 INTRODUCTION

As scientific data increases exponentially in size and complexity, there is a pressing need for a data fabric to link collaborative platforms, data repositories, and tooling that researchers across scientific domains can adopt to advance their research [1–3]. Currently, scientists must navigate a fragmented landscape of computing providers, conflicting best practices, and technical jargon [4–7] across multiple centers with various resources. Any data fabric must be accessible and tightly integrated to coordinate data movement between geographically distributed teams or organizations [8]. Such coordination requires a suite of services to manage networking, storage, and computing resources across the academic and commercial cloud, lowering the barriers to cloud cyberinfrastructure (CI) and supporting data delivery for scientific discovery. However, effective data delivery remains elusive, limiting the scientific impacts of the available CI. This is particularly true for high-volume/high-velocity datasets and for resource-constrained institutions. Several national reports have stressed the urgency of connecting data sources, computing environments, and scientific investigators by addressing the critical challenge of democratizing data delivery [9–11].

To address this challenge, we are building NSDF, a holistic ecosystem for cloud data management to facilitate domain scientists in their daily research. We work closely with resource providers and users to define the services and software stack necessary to help scientists, educators, and students across domains deploy cutting-edge cloud technology. NSDF comprises networking, storage, and computing services, as well as education, community building, and workforce development initiatives, all of which will democratize data delivery and advance scientific discovery.

In this paper, we present a testbed to validate the concept of NSDF. Figure 1 illustrates the logical structure of the testbed. The testbed integrates a suite of networking (both local and global), storage, and computing services; users access the services through NSDF's entry points across different providers. Furthermore, entry points enable the interoperability of different applications and storage solutions, facilitating fast data transfer and caching among data sources, community repositories, and computing environments. The entry points thus provide the foundation for our NSDF testbed and its services. The contributions of this paper are three-fold:

- We present the logical structure and services of our NSDF testbed.
- We evaluate the performance of the three types of services (networking, storage, and computing) for academic and commercial clouds.
- We discuss the benefits of NSDF services for scientific research.

UCC '23, December 4–7, 2023, Taormina (Messina), Italy

ACM ISBN 979-8-4007-0234-1/23/12…$15.00
https://doi.org/10.1145/3603166.3632136

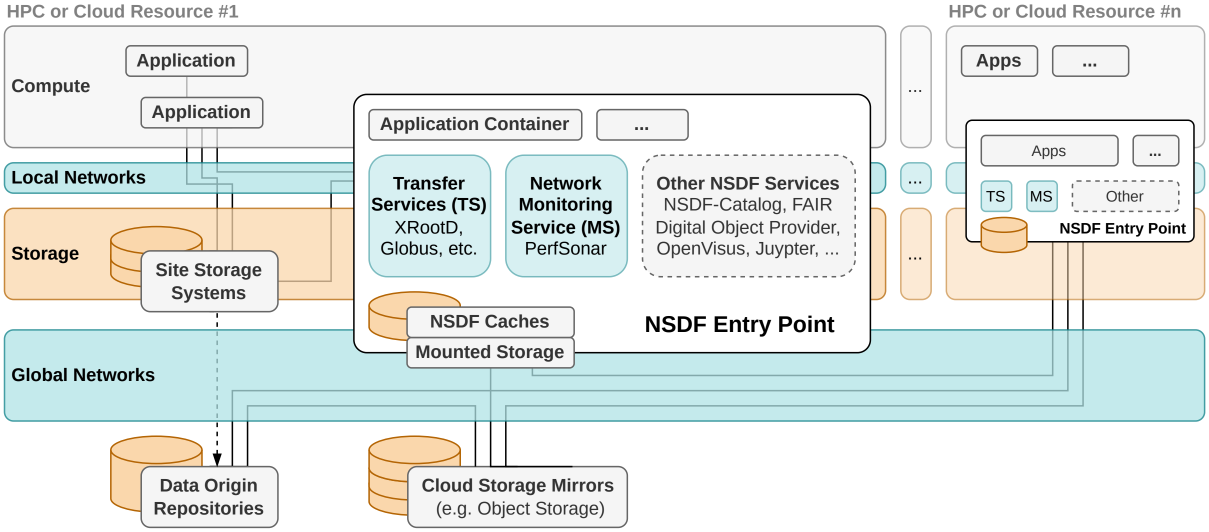


Figure 1: Logical structure of our testbed's networking, storage, and computing services.

2 NSDF SERVICE TESTBED

The networking (local and global), storage, and computing services are at the core of our testbed. We define the three types of services, the critical requirements they address, and their software implementation.

2.1 Networking Services

The need for networking services integrating geographically dispersed data becomes more critical as research becomes increasingly distributed [12]. Funding agencies are increasingly moving away from small-scale research clusters for individual projects. Instead, they foster networking, storage, and computing allocations on large academic clouds and commercial data centers. While this approach offers researchers access to more powerful resources, it also creates additional challenges for sharing data across different platforms and locations. National research and education networks enable researchers to exchange data across some institutions and domains, but sharing data still poses a significant challenge; the process can be prohibitively complex for users.

To address this challenge, we integrate networking services in the NSDF testbed for efficient data sharing and transfer capabilities across networks while hiding the technical complexity of the process. We begin with a testbed of geographically distributed entry points across different academic clouds and research institutions in eight locations in the United States. Figure 2 shows the sites deployed in our testbed and their entry points. We select these sites as entry points because they are heterogeneous in terms of their connections, type of institutions, and research. Thus, they provide a realistic environment for testing and optimizing NSDF services. Each site has at least eight cores, 30 gigabytes of main memory, and 60 GiB of attached storage. This ensures that the entry points can handle large data volumes and effectively support the NSDF services. The testbed includes sites provisioned through CloudLab, Chameleon Cloud, and Jetstream2. Specifically, five sites are provisioned through CloudLab, with two hosts at different locations in Utah (1G and 10G), one host each in Wisconsin (1G), Clemson (1G), and Massachusetts (1G); two sites are provisioned through Chameleon Cloud at TACC (10G) in Texas and CHI (25G) in Illinois; and one host is provisioned on Jetstream2 (10G) in Indiana. The entry points are connected by a high-speed network backbone provided by Internet2 and are designed to interoperate with OSG StashCaches and other resources. For effective networking among sites, we build a software stack that utilizes high-performance data transfer solutions such as Globus [13] and XRootD. This software layer exposes an extensible content delivery network that provides access to data and interoperates with different storage and application solutions in various computing environments. The NSDF testbed allows us to monitor throughput, latency, and routing between entry points over time, identifying areas for improvement and detecting anomalous behaviors. With this extracted knowledge, we automatically coordinate data placement and transfer in the data fabric as we inform users how best to set up NSDF networking services.

2.2 Storage Services

Across cloud platforms, data is generated at unprecedented rates; managing a large amount of data is causing scalability and resilience problems for users. Cloud vendors and users are working on solutions to these problems ranging from hardware and storage technologies (e.g., cloud storage mirrors or file systems) to software and data techniques (e.g., compression). Cloud storage mirror technology, such as object storage, can provide scalable and resilient solutions for cloud data. However, users are often reluctant to move their data to object storage as their legacy applications are optimized to run on local and HPC file systems.

We design a storage service to address the challenge of directly mounting cloud object storage data into a file system. Our solution

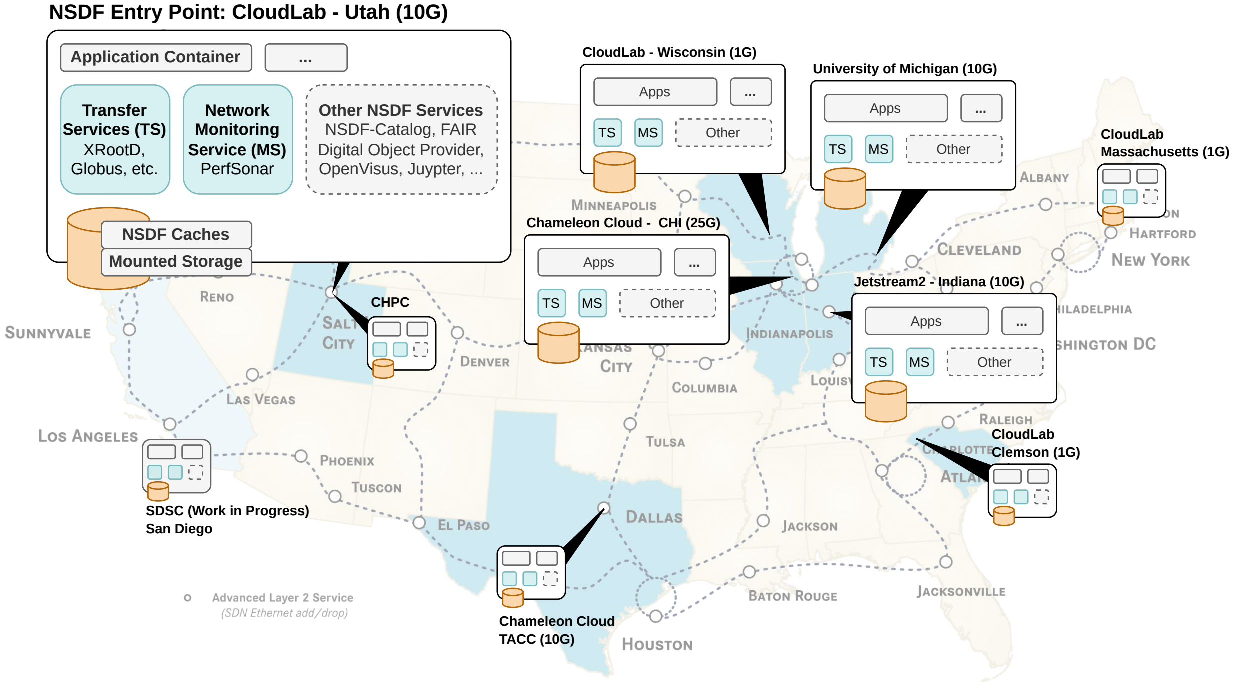

NSDF Entry Point: CloudLab - Utah (10G)
Application Container
...
Transfer Services (TS) XRootD, Globus, etc.
Network Monitoring Service (MS) PerfSonar
Other NSDF Services NSDF-Catalog, FAIR Digital Object Provider, OpenVisus, Juypter, ...
NSDF Caches
Mounted Storage
CloudLab - Wisconsin (1G)
University of Michigan (10G)
CloudLab Massachusetts (1G)
Chameleon Cloud - CHI (25G)
Jetstream2 - Indiana (10G)
CloudLab Clemson (1G)
Apps
TS
MS
Other
CHPC
SDSC (Work in Progress) San Diego
Chameleon Cloud TACC (10G)
Sunnyvale
Reno
Salt Lake City
Denver
Las Vegas
Los Angeles
Phoenix
Tuscon
El Paso
Dallas
Houston
Tulsa
Kansas City
Columbia
Minneapolis
Indianapolis
Louisville
Cleveland
Albany
Hartford
New York
Philadelphia
Washington DC
Raleigh
Charlotte
Atlanta
Jackson
Baton Rouge
Jacksonville
Advanced Layer 2 Service (SDN Ethernet add/drop)

Figure 2: Geographic overview of our testbed with entry points at different locations, including local resources on campuses at the University of Utah, the University of Michigan, and different academic clouds such as Chameleon, CloudLab, and Jetstream2.

is based on Filesystem in USErspace (FUSE) technology on top of S3-compatible object storage. FUSE enables legacy applications to read from and write to files in object storage as though they were from local or HPC file systems. Specifically, FUSE is deployed by mapping software packages that serve as bridges to object storage for legacy applications. Still, users are left with the need to understand the merits and pitfalls of existing packages when mapping object storage to file systems. For effective storage across sites, we extend the NSDF testbed to allow users to mount object storage buckets as file systems on Linux or macOS systems using different mapping packages (i.e., Goofys, GeeseFS, JuiceFS, ObjectiveFS, rclone, s3backer, s3fs; and S3QL) and to measure their performance across cloud platforms. Our FUSE-based service enables the characterization of available mapping packages, provides a set of I/O jobs representative of data patterns on the cloud, and delivers different tests to measure peak performance for various cloud platforms. The FUSE capabilities comprise eight FUSE-based mapping packages on top of S3-compatible object storage. Figure 3 shows the storage services' mapping of data transfer from S3-compatible object storage to the file system. Our testbed allows users to add the installation, mounting, and unmounting actions with a new mapping package. Activities such as creating and deleting are available for deployment. Users can also set the same package with different parameters (e.g., TARGET=geese.v1, geese.v2). Testing can be executed on various cloud platforms by selecting the proper credentials and entry points. Users with workflows that require file systems can thus leverage object storage with minimal changes to their applications when using NSDF storage services.

2.3 Computing Services

As computing capabilities are increasingly supplied through academic and commercial cloud providers, porting applications from one cloud platform to another is prohibitively complex. Providers have unique steps for setting up security groups or networks when virtual machines (VMs) are launched. Gathering the public IPs and injecting SSH access credentials require security solutions beyond

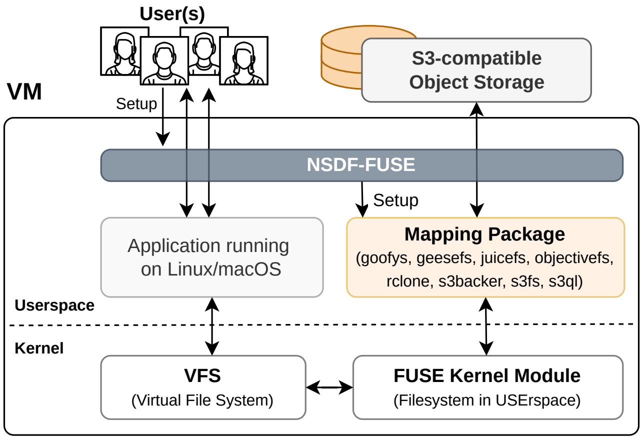

User(s)
VM
Setup
S3-compatible Object Storage
NSDF-FUSE
Setup
Application running on Linux/macOS
Mapping Package
(goofys, geesefs, juicefs, objectivefs, rclone, s3backer, s3fs, s3ql)
Userspace
Kernel
VFS
(Virtual File System)
FUSE Kernel Module
(Filesystem in USErspace)

Figure 3: Data from S3-compatible object storage to file system through FUSE-based mapping packages in our testbed.

the provider's customization. While providers may offer newcomer-friendly web dashboards to monitor and control resources, most also introduce their own vocabulary for different services. Identifying equivalent services is required when moving from one cloud to another. Furthermore, most providers offer command-line interface (CLI) tools or API access in one form or another, but no standard for Identity and Access Management (IAM) has emerged yet. Academic clouds develop high-level abstractions; commercial providers tend to build their stack, hiding details of the underlying open-source infrastructure. Necessary pathways are still missing to leverage underlying academic and commercial infrastructures. There is no universal or standard interface for common actions such as configuration, launching, and termination of virtual resources, so using diverse computing resources effectively imposes a significant technical burden on domain scientists and other users.

To address the orchestration challenges, we design computing services built on a unified API for handling diverse jobs across platforms. We leverage the requirements to augment the NSDF testbed with computing services orchestrating jobs. The API masks different common cloud tasks such as (i) setting up, collecting, and distributing credentials and security settings; (ii) launching VMs and gathering essential information such as public addresses to hand over to higher-level orchestrators such as Ansible [14] or Dask [15]; and (iii) terminating resources once computations or experiments are done. Specifically, the API automates the simultaneous creation, state listing, and tear-down of multiple entry points. It also collects public IPs and accesses credentials required by the higher-level orchestrators. After launching entry points across the NSDF testbed, our computing services automatically generate Ansible inventory files. For effective computing across diverse resources, the API consists of an extendable Python-based toolkit that provides three easy-to-use commands: `create entry point <prefix> −num <N>`, `get entry point <prefix>`, and `delete entry point <prefix>`. Users can configure and register their credentials for different providers using a `vault.yaml`, which our services' CLI utility honors. We configure the NSDF computing services from the CLI and Python scripts. Our computing services thus provide users with an efficient way to orchestrate multiple jobs on the cloud.

3 PERFORMANCE STUDIES

We study the performance of our testbed's networking, storage, and computing services. In our networking study, we set up the geographically distributed testbed across eight entry points at academic cloud sites in Figure 2 and measure latency and throughput over three months. Our storage study examines the features of various object storage mappers in the file system of container environments on single entry points and measures data access performance. Our computing study investigates the performance of various providers' APIs when orchestrating jobs by NSDF. It measures the latency associated with creating and deleting compute resources for different providers.

3.1 Networking Performance

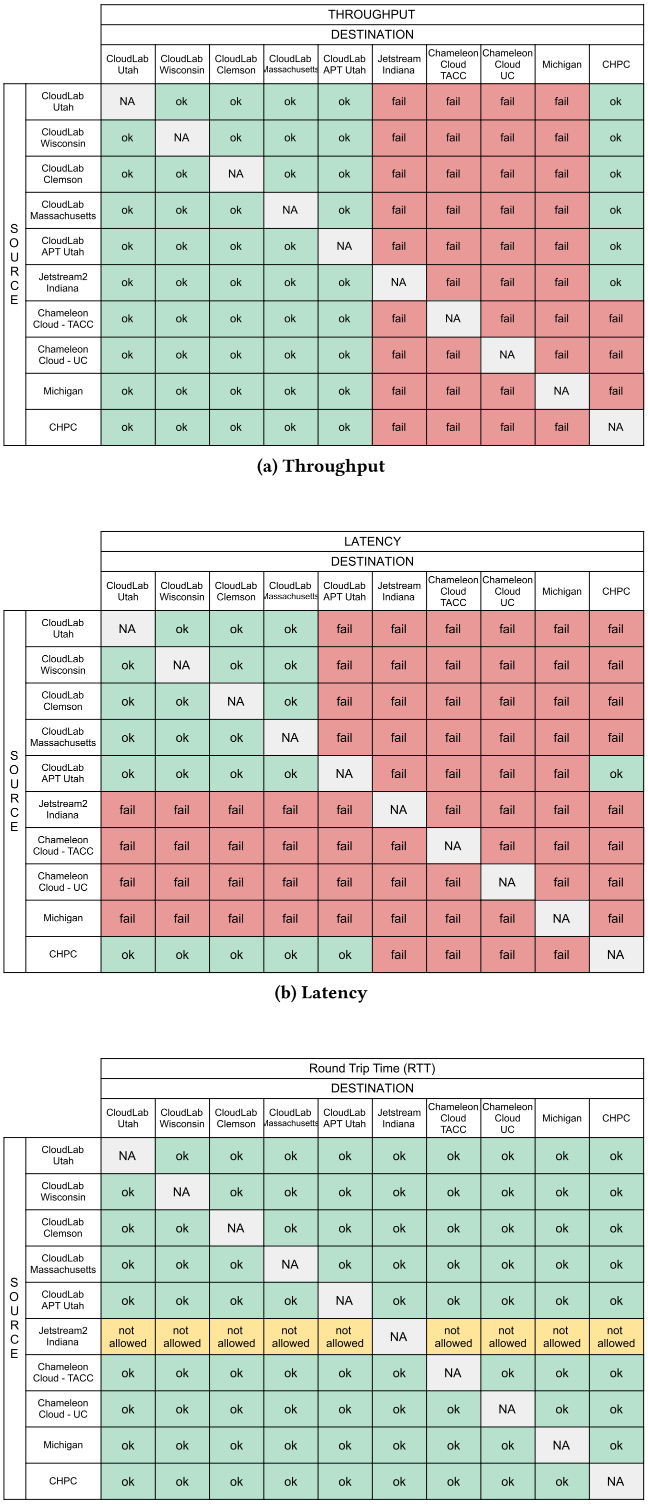
Our networking study deploys the testbed in Figure 2 to continuously sample performance (i.e., throughput, latency, package loss, and traceroute) across the different cloud sites over a period of three months and to quantify performance constraints between entry points. To this end, we use PerfSonar [16] and XRootD [17] to measure point-to-point performance and routing patterns, studying performance variability over time and identifying unexpected performance discrepancies between different locations based on the transfer direction.

Benchmarking with PerfSonar and XRootD. We use PerfSonar and XRootD for measuring and monitoring network performance. We use PerfSonar to collect measurements of throughput (MiB/s), latency (ms), and package loss (percentage) using Round-Trip Time (RTT) as well as characterize the testbed routes between the entry points over three months. We use XRootD [17] to evaluate the performance of our testbed as it transfers different types of scientific data, including metadata and data-heavy project directories. Specifically, we mimic scientific use cases transferring data consisting of multiple files gathered in different folder configurations and different sizes (i.e., 1 MB, 10MB, 1GB) from a client to a server in two Docker containers.

Throughput, Latency, and Package Loss with PerfSonar. PerfSonar cannot always collect metrics in both directions in the point-to-point connections between two entry points because some cloud providers do not support or allow such a test. Figure 4 presents the outcome of our point-to-point performance measurements for throughput (a), latency (b), and RTT (c). Green cells (ok) represent successful metric collections, red cells (fail) represent tests for which PerfSonar is unsupported, and yellow cells represent not allowed tests. For instance, CloudLab allows us to run all tests using PerfSonar and their containerized XRootD clients and servers, suggesting that CloudLab provides a reliable infrastructure for scientific research. However, our tests revealed that hosts on Jetstream2 and Chameleon Cloud, connected through networks utilizing Network Address Translation (NAT), only allow outgoing connections. This means there are connection problems in one direction for 40% of the test locations that employ NAT. An exception is `traceroute` which is blocked by Jetstream2 and fails with a meaningful error message instead of a timeout as indicated in Figure 4. Overall, we collect 32,440 measurements that provide valuable insights into network performance on our testbed.

Understanding peak and average throughput and latency performance between entry points is important for coordinating data movements and lookup requests in our testbed. Figure 5 plots a summary of both the throughput (a) and latency (b) metric measurements for the different entry points. The figure shows that both throughput and latency are subject to performance variability. We occasionally observe transfer speeds above 100 MiB/s for 80% of our entry points. However, only 40% offer an average performance between 50 and 100 MiB/s, with the remainder staying below 50 MiB/s on average. The analysis of latencies is less conclusive at this point, as four of the ten entry points did not allow the directional latency measurement with PerfSonar. Only two hosts consistently had latency below 50 ms and the remaining hosts occasionally suffered from higher tail latencies on some occasions exceeding 1000 ms.

Furthermore, we observe two challenges in managing throughput and latency time-series data. The first challenge is the throughput asymmetry depending on the direction of the data transfer.



		THROUGHPUT									
		DESTINATION									
		CloudLab Utah	CloudLab Wisconsin	CloudLab Clemson	CloudLab Massachusetts	CloudLab APT Utah	Jetstream Indiana	Chameleon Cloud TACC	Chameleon Cloud UC	Michigan	CHPC
SOURCE	CloudLab Utah	NA	ok	ok	ok	ok	fail	fail	fail	fail	ok
	CloudLab Wisconsin	ok	NA	ok	ok	ok	fail	fail	fail	fail	ok
	CloudLab Clemson	ok	ok	NA	ok	ok	fail	fail	fail	fail	ok
	CloudLab Massachusetts	ok	ok	ok	NA	ok	fail	fail	fail	fail	ok
	CloudLab APT Utah	ok	ok	ok	ok	NA	fail	fail	fail	fail	ok
	Jetstream2 Indiana	ok	ok	ok	ok	ok	NA	fail	fail	fail	ok
	Chameleon Cloud - TACC	ok	ok	ok	ok	ok	fail	NA	fail	fail	fail
	Chameleon Cloud - UC	ok	ok	ok	ok	ok	fail	fail	NA	fail	fail
	Michigan	ok	ok	ok	ok	ok	fail	fail	fail	NA	fail
	CHPC	ok	ok	ok	ok	ok	fail	fail	fail	fail	NA

(a) Throughput

		LATENCY									
		DESTINATION									
		CloudLab Utah	CloudLab Wisconsin	CloudLab Clemson	CloudLab Massachusetts	CloudLab APT Utah	Jetstream Indiana	Chameleon Cloud TACC	Chameleon Cloud UC	Michigan	CHPC
SOURCE	CloudLab Utah	NA	ok	ok	ok	fail	fail	fail	fail	fail	fail
	CloudLab Wisconsin	ok	NA	ok	ok	fail	fail	fail	fail	fail	fail
	CloudLab Clemson	ok	ok	NA	ok	fail	fail	fail	fail	fail	fail
	CloudLab Massachusetts	ok	ok	ok	NA	fail	fail	fail	fail	fail	fail
	CloudLab APT Utah	ok	ok	ok	ok	NA	fail	fail	fail	fail	ok
	Jetstream2 Indiana	fail	fail	fail	fail	fail	NA	fail	fail	fail	fail
	Chameleon Cloud - TACC	fail	fail	fail	fail	fail	fail	NA	fail	fail	fail
	Chameleon Cloud - UC	fail	fail	fail	fail	fail	fail	fail	NA	fail	fail
	Michigan	fail	fail	fail	fail	fail	fail	fail	fail	NA	fail
	CHPC	ok	ok	ok	ok	ok	fail	fail	fail	fail	NA

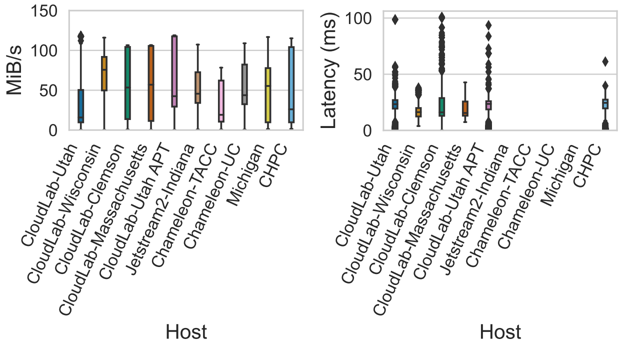
(b) Latency

		Round Trip Time (RTT)									
		DESTINATION									
		CloudLab Utah	CloudLab Wisconsin	CloudLab Clemson	CloudLab Massachusetts	CloudLab APT Utah	Jetstream Indiana	Chameleon Cloud TACC	Chameleon Cloud UC	Michigan	CHPC
SOURCE	CloudLab Utah	NA	ok	ok	ok	ok	ok	ok	ok	ok	ok
	CloudLab Wisconsin	ok	NA	ok	ok	ok	ok	ok	ok	ok	ok
	CloudLab Clemson	ok	ok	NA	ok	ok	ok	ok	ok	ok	ok
	CloudLab Massachusetts	ok	ok	ok	NA	ok	ok	ok	ok	ok	ok
	CloudLab APT Utah	ok	ok	ok	ok	NA	ok	ok	ok	ok	ok
	Jetstream2 Indiana	not allowed	not allowed	not allowed	not allowed	not allowed	NA	not allowed	not allowed	not allowed	not allowed
	Chameleon Cloud - TACC	ok	ok	ok	ok	ok	ok	NA	ok	ok	ok
	Chameleon Cloud - UC	ok	ok	ok	ok	ok	ok	ok	NA	ok	ok
	Michigan	ok	ok	ok	ok	ok	ok	ok	ok	NA	ok
	CHPC	ok	ok	ok	ok	ok	ok	ok	ok	ok	NA

(c) Round-Trip Time (RTT)

Figure 4: Point-to-point Collectable Metrics with PerfSonar.

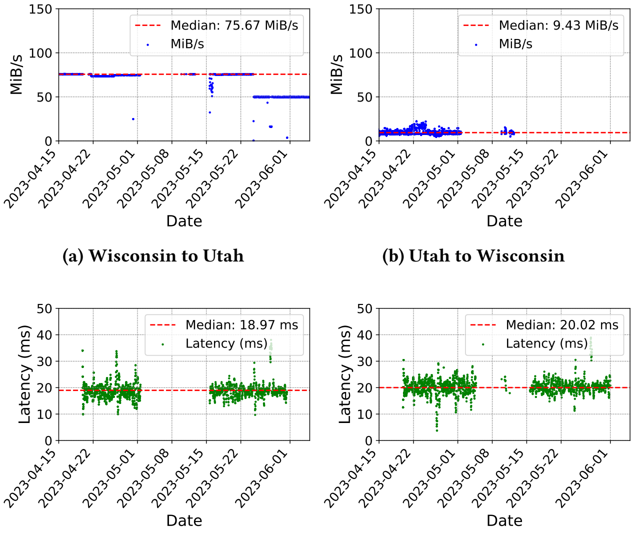
We illustrate an example in Figure 6 showing throughput from Wisconsin to Utah (a) and from Utah to Wisconsin (b), and the associated latency (c) and (d). When transferring data from Wisconsin to Utah, we observe a median throughput of 75.67 MiB/s, but the other direction from Utah to Wisconsin achieves merely 9.43 MiB/s. Interestingly, we do not observe the same asymmetric behavior for latency as shown in Figure 6 (c) and (d). Our networking services need to account for this asymmetry in future work. The second challenge is performance variability, where transfer throughput and latency vary significantly from run to run. We observe variability across point-to-point pairs in our testbed. An example of variability is plotted in Figure 7 for throughput (a) and (b) and latency (c) and (d) between CloudLab entry points at Clemson and Massachusetts.



(a) Throughput (MiB/s)

(b) Latency (ms)

Figure 5: Summary distribution of throughput and latency measurements for different entry points in our testbed.



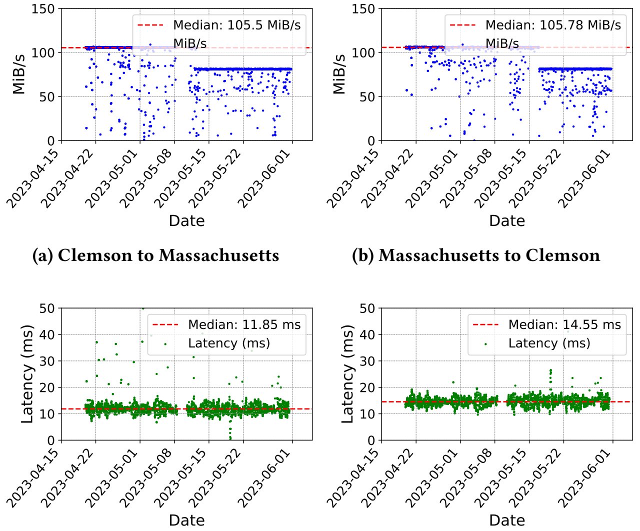
(a) Wisconsin to Utah

(b) Utah to Wisconsin

(c) Wisconsin to Utah

(d) Utah to Wisconsin

Figure 6: Asymmetric throughout and symmetrical latency between Wisconsin to Utah.



(a) Clemson to Massachusetts

(b) Massachusetts to Clemson

(c) Clemson to Massachusetts

(d) Massachusetts to Clemson

Figure 7: Throughput and latency variability between CloudLab entry points at Clemson and Massachusetts.

To automate transfers, our testbed needs to monitor and react to performance variability by notifying the user or adjusting tuning parameters.

Routing Patterns with PerfSonar. The collected data does not allow us to attribute the performance variability to a specific cause. One of the causes could be changed network routing patterns. To rule out route instabilities as a cause of variability, we use PerfSonar to identify the network hops through which we transfer the data and measure the performance. We use this information to generate the routing patterns deployed for our testing. PerfSonar identifies more than 210 network hops. About half of the observed routes include Internet2 (93) or ESnet (13), which are fast backbone networks that quickly enable traffic to be routed across state boundaries. Figure 8 visualizes the superposition of all observed routes for the eight entry points of the testbed. This visualization allows us to verify possible route instabilities. Over our measurements, we observe that routes are stable, with only CloudLab Wisconsin showing alternating routing patterns between Wisconsin and Clemson. Interestingly, we do not observe the same anomaly for the route from Clemson to Wisconsin. Jetstream2 suppresses trace routes entirely (Fig. 4); we, therefore, do not have routing information for this provider. Consequently, we infer that routing instability is not the cause for the performance variability in our testing, leaving e the network contention as a possible cause.

Throughput with XRootD. All entry points in our testbed support containerized deployments so researchers can simplify the deployment and management of their scientific workflows. We leverage containerization to test throughput between clients and servers at different entry points. The client sends the data, and the server receives it. Figure 9 shows the measured throughput for three settings: 1 GiB transferred in a single file, 1 GiB split in 100 files, and 1 GiB split in 1,000 files. For each test, we execute the transfer using a different number of copy jobs (from one to four) and parallel streams (one to fifteen). XrootD imposes these parameter ranges. When copying the data in a single file, we observe variability in performance similar to our observation of PerfSonar. Therefore, we cannot conclude that the number of copy jobs or streams drives performance optimization. However, when splitting the data in multiple files (100 or 1000), the number of copy jobs plays a key role in optimization: the greater the number of jobs, the better the performance. Furthermore, when dealing with a large number of files (100 or 1000), fewer streams result in higher performance. This suggests the importance of integrating parameter adaptability into our testbed.

3.2 Storage Performance

Our storage study examines the peak performance of packages mapping object storage into file system namespaces. The eight mapping packages we use have various features, and our testbed executes jobs with a wide range of I/O patterns on two different commercial clouds.

Feature Comparison. We evaluate the storage performance using eight mapping packages for integrating object storage with file systems on an individual entry point of two different commercial clouds (referred to here as Cloud A and Cloud B). The packages are:

- *Goofys*, an only partially POSIX-compliant tool optimized for high-performance;
- *GeeseFS*, a fork of Goofys focusing on support for small files and metadata operations;
- *JuiceFS*, an optimized tool for shared access and high performance by allowing the use of a dedicated backend server for metadata;
- *ObjectiveFS*, an optimized tool for shared access and automatic scalability and portability;
- *rclone*, a collection of command-line utilities including the option to mount S3 via FUSE;
- *s3backer*, a file system mapping blocks of a single file to objects and mounts this file as a loop device;
- *s3fs*, a file system mapping object names to file paths in the mounted file system; and
- *S3QL*, a file system implemented in Python and designed to favor simplicity and elegance over performance.

Each of the eight packages has different characteristics. We classify them based on seven features. First, all but ObjectiveFS are open source. Second, ObjectiveFS, s3backer, and S3QL have full POSIX support, while the others have partial and limited POSIX compatibility. Third, Goofys, GeeseFS, rclone, and s3fs map file objects from the file system directly to object storage format, while the rest use some fixed-size or sizable chunk data transfer (i.e., file-blocks-objects). Fourth, Goofys, GeeseFS, and s3fs use a location of the file system metadata inferred from the name of the objects (i.e., the object name corresponds to the file system path). The other packages have a custom metadata extra object in the bucket. JuiceFS offers custom metadata in a database server (e.g., Redis dedicated server). Fifth, Goofys, GeeseFS, rclone, and s3fs do not support data compression. The other packages use compression algorithms to minimize transfer time and storage costs. For example, JuiceFS

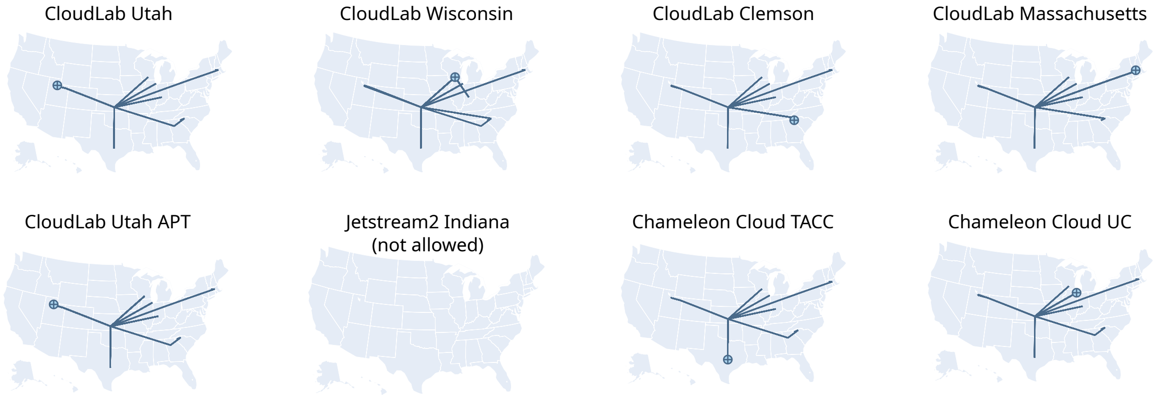


Figure 8: Routing patterns for our testbed testing.

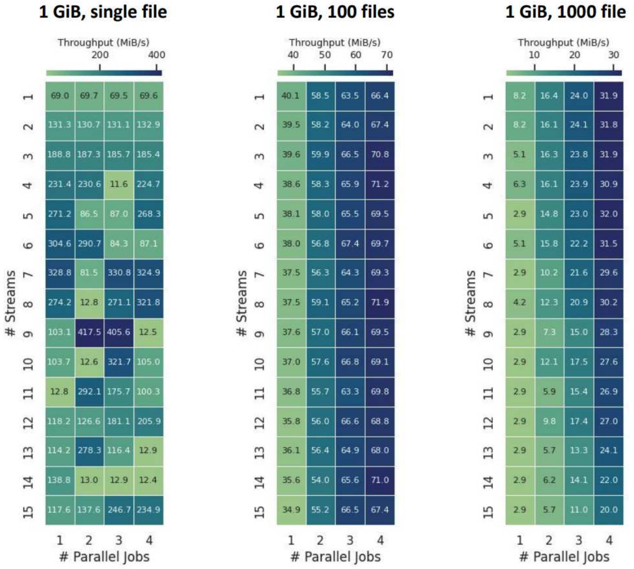


Figure 9: Throughput with different parameter ranges for data transfer with XrootD.

supports the LZ4 and Zstandard; S3QL supports three compression algorithms, LZMA, Bzip2, and zlib; and s3backer supports block-level compression. Sixth, Goofys, rclone, and s3fs do not include any consistency mechanism to define the rules for the order and visibility of read and write in distributed mode to the object storage. GeeseFS and ObjectiveFS require read-after-write consistency: the ability to view changes (read data) right after making those changes (write data). JuiceFS uses close-to-open, which means that when two or more clients read and write the same file simultaneously, the changes made by a client may not be immediately visible to the second client. s3backer supports two mechanisms: it enforces a minimum delay between consecutive PUT or DELETE operations to the same object, and it maintains an internal block MD5 checksum cache that automatically detects and rejects blocks in stale status by GET operations. S3QL supports copy-on-write snapshots. Last, all packages but s3backer and S3QL support concurrent reads (shared access by multiple clients). Only JuiceFS and ObjectiveFS support concurrent writes and updates through the “close-to-open” and “read-after-write." Table 1 summarizes our comparison.

Peak I/O Performance. We use the FUSE-based service to measure the peak I/O performance of the eight packages on the two commercial cloud platforms. We use the mapping packages’ best practices recommended by developers and the cloud community for setting the test environment. Table 2 shows peak I/O performance collected from tests executed across multiple days (to mitigate noisy neighbors in the cloud) and repeated five times for each I/O job.

Application performance varies substantially based on the data access patterns exhibited by the application. To ensure our tests represent a wide range of patterns, we define six I/O jobs that match six common data access patterns (different combinations of sequential vs. random access, access size, and different levels of concurrency). The jobs are as follows:

- **Job 1** Sequential write of eight large files (each file with size 1GB), written sequentially by a single writer;
- **Job 2** Sequential reads of eight large files (each file with size 1GB), read sequentially by a single reader;
- **Job 3** Sequential writes of eight large files (each file with size 1GB), each one written concurrently by one writer (8 writers);
- **Job 4** Sequential read of 8 large files (each file with size 1 GB), each one read concurrently by one reader (8 readers);
- **Job 5** Random writes of 32,768 small files (each file with size 64KB), where each one of 16 writers writes 2,048 files for a total of 128 MiB per writer; and
- **Job 6** Random reads of 32,768 small files (64 KB), where each one of 16 readers reads 2,048 files for a total of 128 MiB per reader.

Each pattern mimics possible I/O operations in real applications on the cloud and at the edge. The results for each of the six jobs and eight mapping packages on the two different cloud platforms are shown in Table 2. We observe that there is no optimal mapping package that provides the highest I/O performance for all data

Table 1: Feature comparison of mapping packages.

Mapping	Open	POSIX	Data	Metadata	Compression	Consistency	Multi-clients	
package	Source	POSIX	mapping	location			Reads	Writes
Goofys	Yes	Partial	Direct	In name	No	None	Yes	No
GeeseFS	Yes	Partial	Direct	In name	No	read-after-write	Yes	No
JuiceFS	Yes	Full	Chunked	In bucket*	Yes	close-to-open	Yes	Yes
ObjectiveFS	No	Full	Chunked	In bucket	Yes	read-after-write	Yes	Yes
rclone	Yes	Partial	Direct	In bucket	No	None	Yes	No
s3backer	Yes	Full	Chunked	In bucket	Yes	PUT or DELETE delay	No	No
s3fs	Yes	Partial	Direct	In name	No	None	Yes	No
S3QL	Yes	Full	Chunked	In bucket	No	copy-on-write	None	No

**JuiceFS offers a dedicated server for the metadata*

Table 2: Peak I/O performance for six jobs on two cloud platforms.

Mapping	Cloud A - Peak I/O performance MiB/s						Cloud B - Peak I/O performance MiB/s					
Package	Job1	Job2	Job3	Job4	Job5	Job6	Job1	Job2	Job3	Job4	Job5	Job6
Goofys	248	546	481	1638	9	28	136	431	356	910	15	78
GeeseFS	248	455	910	585	19	34	136	409	356	146	28	51
JuiceFS	455	327	744	431	13	25	148	47	327	43	11	15
ObjectiveFS	195	315	273	327	41	39	117	240	282	356	62	40
rclone	107	85	372	682	8	16	89	95	372	630	32	47
s3backer	84	81	102	91	62	51	39	130	42	126	29	34
s3fs	74	117	91	136	1	3	34	512	41	585	4	12
S3QL	44	64	56	117	32	9	13	46	6	31	12	9

patterns. Depending on the type of I/O (i.e., read-heavy or write-heavy, sequential or random) in a workflow, the user can deploy our FUSE-based services to test and study their optimal solution. Users deploying our FUSE-based service observe these outcomes. For Job 1 and for Job 2, Juice FS and Goofys enable the highest I/O performance for both cloud platforms. For Job 3 and Job 4, the highest performance is achieved for Cloud A using Goofys. Finally, for Job 5 and for Job 6, the optimal I/O is obtained for Cloud B using ObjectiveFS and Goofys, respectively. Our study suggests that users should consider both the type of I/O job and the cloud platform on which the job is executed when selecting a mapping package.

3.3 Computing Performance

Our computing study evaluates how to orchestrate elasticity for our testbed services, with the goal of allowing application jobs to scale up and down across different cloud sites. We investigate different APIs on single entry points for commercial and academic cloud providers. Specifically, we create and delete VMs across different cloud providers and measure the latency until the resources become available. Our analysis focuses on two aspects. First, we compare features across different commercial and academic cloud computing providers. Second, we study the latency for creating and deleting operations during resource management of the different cloud providers.

Feature Comparison. We highlight challenges encountered when using five different academic and commercial cloud providers and their APIs. Table 3 presents four key characteristics (i.e., credential management, region availability, underlying software stack, and support for custom container images) for the five cloud platforms in our study. When testing the five cloud platforms, we gathered the following observations:

- (Ob.1) Accessibility and ease of use are central concerns that all the providers address through different means. All offer newcomer-friendly web dashboards both to monitor and control resources.
- (Ob.2) Most of the providers introduce their own vocabulary for different services; identifying equivalent services when moving from one cloud to another is a common challenge.
- (Ob.3) Most providers offer CLI tools or API access in one form or another, but no standard for Identity and Access Management (IAM) has emerged yet, requiring special procedures for each provider.
- (Ob.4) Providers have their own unique sequence of steps, such as setting up security groups or networks when VMs are launched.
- (Ob.5) Gathering the public IPs and injecting SSH access credentials require significant customization for each provider.
- (Ob.6) Academic clouds develop high-level abstractions and often offer pathways to leverage the underlying infrastructure; conversely, commercial providers tend to develop their own stack and hide details of the underlying open-source infrastructure.
- (Ob.7) Most providers feature multiple regions either globally or in the US.
- (Ob.8) Some academic clouds enforce a lease-based model that requires users to make reservations before launching resources; on the other hand, commercial clouds usually offer on-demand services.

Table 3: Characteristics of the different cloud service providers, both academic and commercial.

Provider	Type	Credentials	Multi-Region	Stack	Custom Images
AWS	Commercial	Token+Secret	Yes (Int.)	Custom	Yes
Chameleon	Academic	Token	Yes (US)	CHI on OpenStack	Yes*
CloudLab	Academic	Certificate	Yes (US)	Custom	Yes
Vultr	Commercial	Token+IP-Whitelist	Yes (Int.)	Custom	Yes
JetStream	Academic	Token	Yes (US)	Atmosphere on OpenStack	No*

The lack of standard APIs and credential management poses a significant challenge for NSDF. Our testbed addresses the challenge using a special wrapper that creates a unified API for various providers.

VM Creation and Deletion Latency. We evaluate our unified API in terms of its latency by launching varying numbers of VMs {1, ..., 16} with each of the different providers and measuring the latency to complete different actions {create, delete}. To create a VM, we measure the time until logging onto the nodes using SSH succeeds. Our measurements number of VMs requested is plotted in Figure 10. The task of launching between one and 16 VMs usually completes within 10 minutes. We observe that the amount of resources requested in most cases does not have an effect on the overall launch latency in our experiments. We also observe that the two commercial providers, AWS and Vultr, often launch resources more reliably than their academic counterparts. Our API enables spawning resources across different cloud providers, thus supporting users as they create ad hoc compute clusters using different clouds and higher-level orchestration.

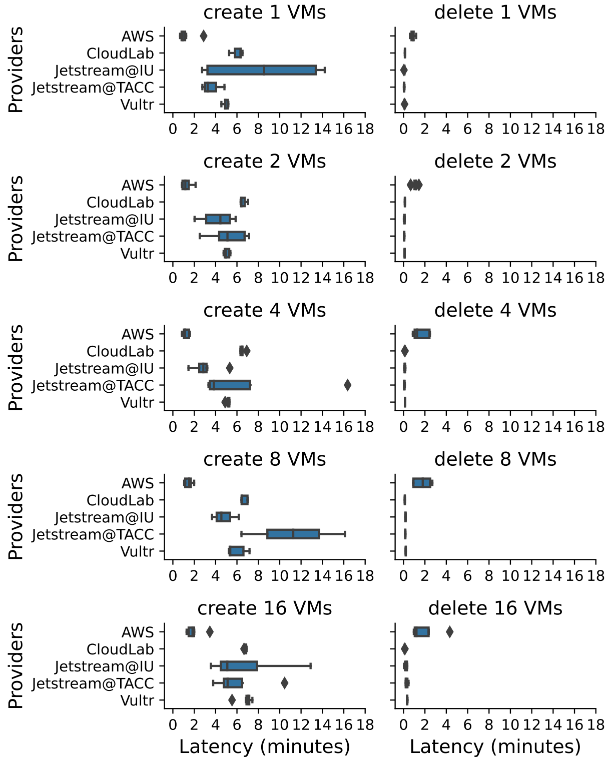


Figure 10: Latency in seconds to complete the *create* and *delete* commands using the NSDF-Cloud CLI tools.

4 BENEFITS ACROSS SCIENTIFIC DOMAINS

As scientific data increases exponentially, individual researchers from domains as different as economics, evolutionary biology, materials science, and bioinformatics all encounter similar data scalability and usability challenges. Moreover, the challenges faced by individual researchers mirror those at major multi-institutional projects such as IceCube Neutrino Observatory [18–20], Laser Interferometer Gravitational-Wave Observatory (LIGO) [21], Virgo interferometer [22], and Kagra observatories [23]. Producing, managing, and repeatedly processing large amounts of data is an ongoing challenge. For instance, IceCube uses workflows based on cloud bursts [18, 24] for which a typical job is configured to use less than 100MB of unique input data to produce 2.5GB of unique output in about 5 fp32 TFLOP-hours of compute time. This workflow dominates the IceCube simulation activities, and variants of it run routinely on OSG, Pacific Research Platform (PRP), and various NSF HPC resources. Our NSDF services can enable researchers at IceCube and elsewhere to use platforms such as Open Science Network (OSN) pods [25], the NSF Virtual Data Collaboratory (VDC) [26], and the Open Science Grid (OSG) [27, 28] for distributed high throughput computing for open science [21].

Our testbed demonstrates the potential for NSDF to address these challenges, increasing scalability and usability while decreasing time-to-solution for multiple domains. Ultimately, we believe this will democratize data delivery and advance scientific discovery. For instance, the Materials Genome Initiative (MGI) [**?**] reflects a national priority to accelerate materials development. By emphasizing data-centric approaches to the integration of computational and experimental methods (i.e., using integrated experiments, simulation, and data) [29, 30], MGI aims to cut materials development time substantially, reduce the cost to develop new materials, and accelerate the scientific understanding of complex materials phenomena. In the past, it has taken 10 to 20 years to design and implement a new material for an engineering application [31, 32]. Today, the ever-pressing needs of society require that materials be developed at a much more accelerated pace. Through our services, MGI can reduce the timeline for materials design, testing, and validation loop from years to months, with immeasurable benefits to society. The discovery and deployment of innovative materials, facilitated by NSDF, can meet challenges in critical areas such as energy, security, environment, transportation, and public health, ultimately addressing many of NSF's 10 Big Ideas, including Growing Convergence Research and Making the Quantum Leap [33]. Particle physics such as MINERVA [34], NuMI Off-axis Appearance (NOVA) experiment [35], and Deep Underground Neutrino Experiment (DUNE) [36], or the search for transients with the Dark

Energy Survey (DES) [37], similarly require the increased scalability and usability with decreased time-to-solution that NSDF can provide.

5 CONCLUSION

In this paper, we present a testbed as a prototype of NSDF. Our testbed integrates networking, storage, and computing services that users access through entry points with different providers. We evaluate the performance of networking, storage, and computing services for academic and commercial clouds. We anticipate that NSDF will ultimately increase the usability and scalability of scientific applications on the cloud as it simultaneously increases the speed of discovery. For our networking service study, our testbed spans eight locations and collects performance metrics that allow us to identify constraints when moving data across research sites in the United States. Our storage service study evaluates tuning opportunities and I/O performance of FUSE-based file systems on top of S3-compatible object storage. Our benchmarks enable a comprehensive analysis of different mapping packages depending on a specific I/O pattern and cloud platform. Our computing service study assesses both academic and commercial clouds, performing common tasks such as creating and deleting compute resources. We leverage the obtained latency for create and delete operations to understand how NSDF can elastically scale and orchestrate its own service workloads as well as user applications. In future work, we will extend the testbed with additional data points and monitor it over longer periods and with additional workflows. We will also compare the performance of NSDF with other transfer tools, such as GridFTP, Globus, and S3 clients, to enable optimization.

ACKNOWLEDGMENTS

This research is supported by the National Science Foundation (NSF) awards #1841758, #2028923, #2103836, #2103845, #2138811, #2127548, #2223704, #2330582, #2331152, #2334945; DoE award DE-FE0031880, the Intel oneAPI Centers of Excellence at the University of Utah, the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the DoE and the NNSA, and UT-Battelle, and LLC under contract DE-AC05-00OR22725. Results presented in this paper were obtained in part using resources from ACCESS TG-CIS210128; CloudLab PID-16202; Chameleon Cloud CHI-210923; Fabric; and IBM Shared University Research Award.

REFERENCES

[1] National Research Council, *Big Data in Materials Research and Development: Summary of a Workshop.* The National Academies Press, 2014.

[2] T. T. Wong, "Building a Materials Data Infrastructure," *The Minerals, Metals & Materials Society (TMS), JOM*, vol. 68, p. 2029–2030, 2016.

[3] N. Zhou *et al.*, "Orchestration of Materials Science Workflows for Heterogeneous Resources at Large Scale," *The International Journal of High Performance Computing Applications*, vol. 37, no. 3-4, pp. 260–271, 2023.

[4] P. Olaya *et al.*, "Building Trust in Earth Science Findings through Data Traceability and Results Explainability," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 34, no. 2, pp. 704–717, 2023.

[5] J. Luettgau *et al.*, "NSDF-Cloud: Enabling Ad-Hoc Compute Clusters Across Academic and Commercial Clouds," in *Proc. of the 31st International Symposium on High-Performance Parallel and Distributed Computing*, pp. 279–280, ACM, 2022.

[6] P. Olaya *et al.*, "NSDF-FUSE: A Testbed for Studying Object Storage via FUSE File Systems," in *Proc. of the 31st International Symposium on High-Performance Parallel and Distributed Computing*, pp. 277–278, ACM, 2022.

[7] P. Olaya *et al.*, "Enabling scalability in the cloud for scientific workflows: An earth science use case," in *Proc. of IEEE 16th International Conference on Cloud Computing (CLOUD)*, pp. 383–393, 2023.

[8] J. Luetgau *et al.*, "Development of large-scale scientific cyberinfrastructure and the growing opportunity to democratize access to platforms and data," in *Proc. of the 11th International Conference: Distributed, Ambient and Pervasive Interactions (DAPI), Held as Part of the 25th HCI International Conference (HCII)*, pp. 378–389, Springer Nature Switzerland, 2023.

[9] National Academies of Sciences and Medicine, *Open Science by Design: Realizing a Vision for 21st Century Research.* The National Academies Press, 2018.

[10] National Academies of Sciences and Medicine, *Future Directions for NSF Advanced Computing Infrastructure to Support U.S. Science and Engineering in 2017-2020.* The National Academies Press, 2016.

[11] Subcommittee On Future Advanced Computing Ecosystem Of The National Science & Technology Council, "Pioneering the Future Advanced Computing Ecosystem: A Strategic Plan," January, 2021.

[12] J. Luettgau *et al.*, "Studying Latency and Throughput Constraints for Geo-Distributed Data in the National Science Data Fabric," in *Proc. of the 32nd International Symposium on High-Performance Parallel and Distributed Computing*, p. 325–326, ACM, 2023.

[13] I. Foster, "Globus Online: Accelerating and Democratizing Science through Cloud-Based Services," *IEEE Internet Computing*, vol. 15, no. 3, pp. 70–73, 2011.

[14] M. DeHaan, "Ansible." Available at https://github.com/ansible/ansible, [Online; accessed 10-30-2023].

[15] M. Rocklin, "Dask: Parallel computation with blocked algorithms and task scheduling," in *Proc. of the 14th Python in Science Conference*, no. 130-136, 2015.

[16] A. Hanemann *et al.*, "PerfSONAR: A Service Oriented Architecture for Multi-domain Network Monitoring," in *Proc. of Service-Oriented Computing - ICSOC 2005*, pp. 241–254, Springer Berlin Heidelberg, 2005.

[17] C. Boeheim *et al.*, "Scalla: Scalable Cluster Architecture for Low Latency Access Using Xrootd and Olbd Aervers," *Technical report, Stanford Linear Accelerator Center*, 2006.

[18] I. Sfiligoi *et al.*, "Running a Pre-exascale, Geographically Distributed, Multi-Cloud Scientific Simulation," in *Proc. of The High-Performance Computing ISC*, p. 23–40, Springer International Publishing, 2020.

[19] I. Sfiligoi *et al.*, "Demonstrating a pre-exascale, cost-effective multi-cloud environment for scientific computing: Producing a fp32 exaflop hour worth of icecube simulation data in a single workday," in *Proc. of Practice and Experience in Advanced Research Computing (PEARC '20)*, p. 85–90, ACM, 2020.

[20] I. Sfiligoi *et al.*, "Pushing the Cloud Limits in Support of IceCube Science," *IEEE Internet Computing*, vol. 25, pp. 71–75, Jan. 2021.

[21] B. Bockelman *et al.*, "Principles, Technologies, and Time: The Translational Journey of the HTCondor-CE," *Journal of Computational Science*, vol. 52, p. 101213, 2021.

[22] J. Abadie *et al.*, "Search for gravitational waves from compact binary coalescence in LIGO and Virgo data from S5 and VSR1," *Phys. Rev. D*, vol. 82, no. 10, p. 102001, 2010.

[23] "Kagra Observatory." Available at https://www.icrr.u-tokyo.ac.jp/en/facility/4219/, [Online; accessed 10-30-2023].

[24] I. Sfiligoi *et al.*, "Managing Cloud Networking Costs for Data-Intensive Applications by Provisioning Dedicated Network Links," in *Proc. of Practice and Experience in Advanced Research Computing (PEARC '21)*, ACM, 2021.

[25] "OpenStorage Network." Available at https://www.openstoragenetwork.org, [Online; accessed 10-30-2023].

[26] M. Parashar *et al.*, "The Virtual Data Collaboratory: A Regional Cyberinfrastructure for Collaborative Data-Driven Research," *Computing in Science & Engineering*, vol. 22, no. 3, pp. 79–92, 2020.

[27] R. Pordes, "The Open Science Grid," *Journal of Physics: Conference Series*, vol. 78, p. 012057, 2007.

[28] L. Bauerdick *et al.*, "Xrootd, Disk-Based, Caching Proxy For Optimization Of Data Access, Data Placement And Data Replication," *Journal of Physics: Conference Series*, vol. 513, no. 4, p. 042044, 2014.

[29] K. Alberi *et al.*, "The 2019 materials by design roadmap," *Journal of Physics D: Applied Physics*, vol. 52, p. 013001, oct 2018.

[30] L. Aagesen *et al.*, "PRISMS: An Integrated, Open-Source Framework for Accelerating Predictive Structural Materials Science," *The Minerals, Metals & Materials Society (TMS), JOM*, vol. 70, 08 2018.

[31] Engineering, Committee and Board, National and Sciences, Division and Council, National, *Integrated Computational Materials Engineering: A Transformational Discipline For Improved Competitiveness And National Security.* The National Academies Press, 10 2008.

[32] National Research Council, *Application of Lightweighting Technology to Military Aircraft, Vessels, and Vehicles.* The National Academies Press, 2012.

[33] "NSF's 10 Big Ideas - Special Report." Available at https://www.nsf.gov/news/special_reports/big_ideas/, [Online; accessed 10-30-2023].

[34] L. Fermi Research Alliance, "MINERvA: Bringing neutrinos into sharp focus." Available at https://minerva.fnal.gov, [Online; accessed 10-30-2023].

[35] L. Fermi Research Alliance, "NOvA: NuMI Off-axis Appearance experiment." Available at https://novaexperiment.fnal.gov/, [Online; accessed 10-30-2023].

[36] L. Fermi Research Alliance, "DUNE: Deep Underground Neutrino Experiment." Available at https://lbnf-dune.fnal.gov/, [Online; accessed 10-30-2023].

[37] R. Kessler *et al.*, "The Difference Imaging Pipeline For The Transient Search In The Dark Energy Survey," *The Astronomical Journal*, vol. 150, 11 2015.