Understanding Data Motion in the Modern HPC Data Center

Glenn K. Lockwood,* Shane Snyder,† Suren Byna,* Philip Carns,* Nicholas J. Wright†
*Lawrence Berkeley National Laboratory, †Argonne National Laboratory
*{glock, sbyna, njwright}@lbl.gov, †{ssnyder, carns}@mcs.anl.gov

***Abstract*—The utilization and performance of storage, compute, and network resources within HPC data centers have been studied extensively, but much less work has gone toward characterizing how these resources are used in conjunction to solve larger scientific challenges. To address this gap, we present our work in characterizing workloads and workflows at a data-center-wide level by examining all data transfers that occurred between storage, compute, and the external network at the National Energy Research Scientific Computing Center over a three-month period in 2019. Using a simple abstract representation of data transfers, we analyze over 100 million transfer logs from Darshan, HPSS user interfaces, and Globus to quantify the load on data paths between compute, storage, and the wide-area network based on transfer direction, user, transfer tool, source, destination, and time. We show that parallel I/O from user jobs, while undeniably important, is only one of several major I/O workloads that occurs throughout the execution of scientific workflows. We also show that this approach can be used to connect anomalous data traffic to specific users and file access patterns, and we construct time-resolved user transfer traces to demonstrate that one can systematically identify coupled data motion for individual workflows.**

***Index Terms*—data movement, storage, workflows**

I. Introduction

High-performance computing (HPC) has historically been dominated by modeling and simulation workflows whose I/O needs are largely driven by checkpoint/restart. However, the role of HPC is rapidly expanding to include large-scale data analysis as a result of both increased data generation rates from modern scientific instruments and the emergence of artificial intelligence as a technique to rapidly extract insight from large volumes of data [1]–[5]. Hence, characterizing the requirements and performance of I/O in modern HPC centers now requires a more holistic examination of data movement.

A large body of knowledge exists on data motion between compute and storage from the perspectives of both applications and systems [6]–[11] because this form of I/O blocks forward progress on running jobs. However, data-driven workflows rely on data movement not only between compute and storage systems, but also between storage systems and between storage and external networks as depicted in Figure 1. Today's HPC facilities may have three or more tiers of storage for hot, warm, and cold data, and data moves between these tiers as an integral part of scientific workflows [12]–[17]. These workflows may also utilize a diversity of resources across multiple facilities as different stages of data processing require

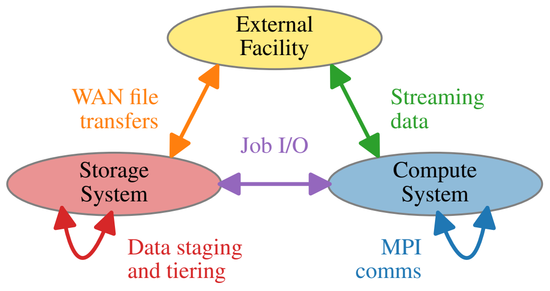

Fig. 1. Data sources and sinks in scientific computing (nodes) and the transfer vectors along which data moves between them (edges).

computational resources that are better suited for accelerators or processors not deployed universally.

The end result of this architectural diversity is a wide range of data paths that are of growing importance to the overall time to solution for modern scientific workflows. Historically, most research has focused on the data path between the primary compute and hot storage tier, and systematic studies of secondary storage systems have examined such systems in isolation. For example, the ways in which users interact with archival tape systems was found to vary widely over time and defy a singular definition of steady-state workload that holds true at multiple time scales [18]. More recent analysis of NCAR's tape archive established a methodology by which the archive's user recall rate and internal data movement workloads informed an optimal ratio of disks to tapes [19]. In both cases, however, the motivations behind the observed user behavior were not explored in the greater context of scientific workflow, and transfers originating from within their respective data centers were not systematically distinguished from transfers originating from the outside. As a result, such studies of secondary storage tiers have not clearly defined how their findings can be applied to optimize entire workflows that operate throughout the data center.

The performance of transfers between storage and the wide-area network (WAN) have also been studied with increasing enthusiasm as networking technologies and methodologies [20] have advanced to the point where geographically distributed workflows and large, public datasets are now enabling new scientific discovery [2], [3], [21]. Globus, GridFTP-based transfer tools, and bbcp are ubiquitous high-performance data transfer tools used to this end [1]–[3], and characterizing the ways in which Globus and GridFTP are used in multisite workflows and distributed HPC environments is the subject of growing interest [12], [22], [23]. As with the efforts to characterize user interactions with archival storage systems, however, studies of wide-area data transfers have explored only the storage-WAN and WAN-storage components of data transfer.

In this work, we take a holistic approach to understanding data motion and illustrate how data moves through the entire data center—storage, compute, and WAN. We present several methods that allow us to infer how data moves along the data transfer vectors illustrated in Figure 1 using readily available tools. We then identify gaps in existing monitoring data that, if addressed, will enable a complete understanding of how data moves throughout the execution of scientific workflows. Using data transfer records from the National Energy Research Scientific Computing Center (NERSC) and the Energy Sciences Network (ESnet) between May 1, 2019 and August 1, 2019, we demonstrate new insights that arise from resolving data transfers on the basis of data ownership, data source and destination, and time.

II. Methods

A. Model for Data Motion

Rather than approaching data movement by examining the storage systems at one or both ends of a transfer, we describe all movement of data in terms of data transfers. Each data transfer can be represented as a *transfer record*, a simple abstraction that possesses several essential attributes:

1) Source storage target, source host(s), and source site
2) Destination storage target, destination host(s), and destination site
3) Start and end time
4) Volume of data moved
5) Protocol or tool used to moderate the data transfer
6) Owner of the data being transferred

We define sources and destinations in terms of three components: storage target (a persistent storage system), host (a network endpoint that controls access to a target), and site (a collection of hosts) because not all transfers occur between two well-defined storage systems. For example, a standard job checkpoint operation has one or more source hosts (compute nodes), but the source target is DRAM and not a storage system; in such cases, the source target would be undefined despite the source host being defined. These three components are collectively used to determine the transfer vector to which each transfer record belongs. For example, we assert that a transfer involving two different sites must be a WAN transfer, and transfers that involve a tape controller host must involve a tape storage target. Defining the heuristics that map targets/hosts/sites to transfer vectors requires knowledge of how different storage systems and hosts are connected to form data paths within a data center, and this map will be different for each data center.

B. System Descriptions

We examine data transfers between a variety of systems within the NERSC data center as well as transfers to the outside world, all of which map to nodes and edges in Figure 1.

Cori is the principal computing platform at NERSC and is comprised of over 12,000 compute nodes tightly coupled on a high-speed network (HSN). Cori has twelve login nodes into which users can log in, submit jobs, and initiate data transfers to and from other storage systems. Data transfers originated within the HSN can reach the center-wide network through ten software-defined networking (SDN) nodes [24]. Cori is the only compute system considered in this study, and all compute-storage and storage-compute transfers discussed hereafter come from or go into this system.

Cori is directly attached to three large-scale storage systems: a 27 PiB Lustre file system ("cscratch") that is mounted only on Cori nodes and data transfer nodes, a 15 PiB Spectrum Scale file system ("project") that is mounted throughout the center, and a 1.6 PiB DataWarp burst buffer that is directly integrated into Cori's HSN. An additional home file system is also mounted center-wide, but users are limited to a 40 GiB quota on this file system, and it is neither a high-capacity nor high-performance storage resource.

NERSC also provides a tape-based HPSS [25] archive (named "archive") that was provisioned with 390 PB of slot capacity at the time of this study. Although the vast majority of the archive capacity is tape based, there is a 4 PiB hard disk drive cache to which writes are buffered before being written to tape. User access to this archive system occurs through an HPSS gateway node that provides four interfaces (hsi, htar, ftp, parallel ftp), and users can transfer data to/from the archive from both internal NERSC systems or from the wide-area network. The archive also allows users to store and retrieve data directly from the WAN using Globus, but this specific set of transfers was not included in this study, and the resulting error is quantified in Section III-B.

For parallel data transfers managed by Globus and user-provided parallel copy tools, NERSC maintains a pool of ten data transfer nodes (DTNs). A subset of these nodes are accessible for user login, and all DTNs are capable of transferring files between NERSC's file systems (storage-storage) or to/from the outside world (storage-WAN and WAN-storage). NERSC also provides service nodes as part of Spin, its container-as-a-service platform. This platform allows users and staff to deploy services in support of workloads both internal (such as providing databases for workflows) and external (such as science gateways) to the center. Because these containers are user managed, however, their role in data movement is not well defined; hence, no transfer logs from these systems were included in this study.

C. Data Sources

Transfer records were created from a variety of systems within the NERSC data center:

- **Globus transfer log files** from the ten data transfer nodes were used to generate storage-WAN, WAN-storage, and storage-storage transfer records.
- **HPSS hsi, htar, ftp, and parallel ftp logs** from the HPSS gateway node were used to generate storage-WAN, WAN-storage, and storage-storage transfer records.
- **Darshan logs** from Cori were used to derive compute-storage and storage-compute transfer records.

We defined the mapping between log file entries and transfer records such that each file transfer that appeared in Globus and HPSS logs generated to a single transfer record. However, each Darshan log corresponds to a single invocation of an MPI application that may access many files over time, so for each such log, we generated a single transfer for each unique file path and transfer direction (read or write). Parallel I/O operations (where multiple nodes were all performing I/O to the same file) were aggregated into a single transfer. The transfer volume was the sum of all bytes read or written to that file during that job by the application, and the transfer's start and end times were defined as the times at which the first and last read or write operation were issued by the application. Therefore, some I/O may have been overstated if application I/O was able to make use of node-local page cache, and repeatedly opening and closing a file of the same name still generated only a single transfer record per direction.

To quantify the amount of data that was not captured by the transfers studied here, we also collected ground-truth measurements of data in and out of each storage system directly. The absolute number of bytes read from and written to the storage systems was obtained using the Lustre Monitoring Tool for Lustre, mmperfmon for Spectrum Scale, and HPSS daily reporting for HPSS. The ground-truth data volumes for the burst buffer were measured using NERSC's daily smartctl monitoring data.

The burst buffer's ground-truth data does contain additional reads and writes attributed to file system-level metadata updates because smartctl returns host-initiated, block-level counters rather than user-initiated, file-level measurements. Similarly, the ground-truth values for external transfers were collected from ESnet's SNMP REST API [26] and include protocol framing overheads that are transparent to users. However, we do not expect these overheads to amount to a significant volume of data relative to user-initiated data transfers in either the burst buffer or WAN ground-truth values. The size distribution of files at rest was obtained for the cscratch and project file systems by using the Robinhood database and Spectrum Scale's Information Lifecycle Management (ILM) policy manager, respectively.

All transfers at NERSC between May 1, 2019 and August 1, 2019 were included in this study, resulting in a total of 194,019,683 transfer records reflecting 78.6 PiB of data moved. This choice of time reflects our desire to analyze a representative amount of the NERSC workload while balancing the computational cost of rapidly prototyping different analysis techniques. We used the pytokio library [11] and its connector interfaces for Darshan, LMT, mmperfmon, HPSS logs, smartctl, and the ESnet SNMP REST API to convert the diversity of data sources into uniform transfer records or scalar ground-truth values. This data processing and analysis was implemented using the Dask distributed analysis library [27].

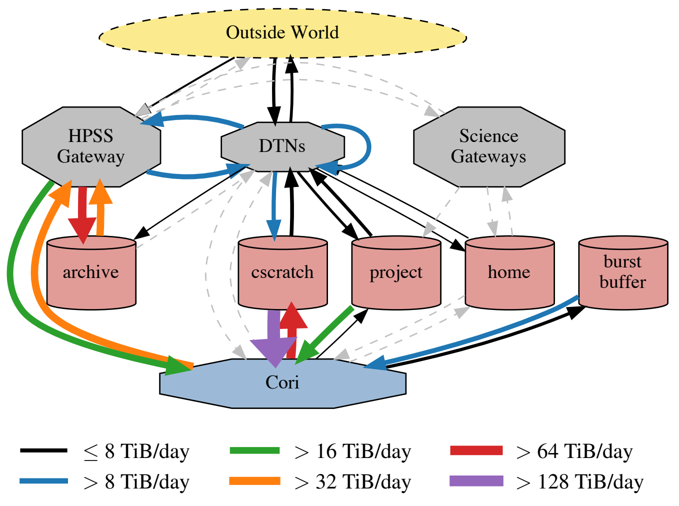

Fig. 2. Directed graph generated from transfers between May 1, 2019 and August 1, 2019. Edges whose daily I/O rates are less than 1.0 TiB/day are dashed; edges whose sources or destinations could not be precisely determined are not drawn. "Science Gateways" are service nodes that may host internally or externally facing services for users. "Cori" includes compute nodes, login nodes, and SDN nodes.

III. Results and Discussion

A. Transfer Directionality

Because we define transfers in terms of both data volume and directionality, they naturally form a directed graph when edges sharing common sources and destinations are aggregated. Figure 2 shows the result of this process using all transfers observed during this study and represents a concrete example of the notional graph shown in Figure 1. All edges between Cori (blue hexagon) and storage systems (red cylinders) represent storage-compute and compute-storage transfers, and all edges connected to the outside world represent storage-WAN and WAN-storage transfers. The remaining edges are paths followed by storage-storage transfers, which, unlike the other transfer vectors, may take multiple hops. For example, a heavily used data path exists between Cori and the HPSS gateway, but the true storage-storage transfers underpinning this path may involve data transferring from cscratch to Cori, from Cori to the HPSS gateway, and then from the HPSS gateway to the archive system itself. Such a transfer would contribute to all three graph edges.

The visualization of aggregate transfer paths in Figure 2 reveals a number of interesting patterns:

- The home file systems, which are not designed for high performance, are used sparingly on a volume basis. This usage indicates that efforts to limit its load (user education and policies) are effective.

- Similarly, the project file systems (intended to store important data persistently) are subject to a much higher read workload, suggesting that users are indeed using these file systems to store important datasets that are analyzed by multiple jobs.
- The burst buffer appears to be more read-heavy than write-heavy as well, counter to the intuition that it is a write-heavy storage system.
- The tape archive exchanges data predominantly with Cori itself rather than the DTNs, suggesting that the benefits of using a dedicated data transfer node to archive data are not worth the small added effort for users.

This graph-based representation of data motion illustrates both the critical data paths throughout the data center and lightly used data paths that may represent poorly configured services, misinformed users, or areas in need of increased architectural attention.

B. Identifying Gaps in Transfer Data

This transfer-based approach to understanding I/O provides new insights by enabling sources and destinations of data flows to be chained together, but the approach has several limitations as it was applied to create Figure 2. For example, the data sources used to construct transfer records may attribute different semantic meaning to data volume metrics and make it impossible to distinguish a user-level representation of a data transfer (e.g., writing two 4 KiB pages) from a network-level data transfer (e.g., issuing a single 8 KiB RPC) due to intermediate caching. Furthermore, not all mechanisms used to transfer data are logged or loggable; for example, a data transfer that used the `cp` or `rsync` command would not have been captured in any of the data used in this study and would not be represented in Figure 2.

Acknowledging this inability to completely account for every byte transferred leads us to quantify the amount of missing data so that we can better gauge the accuracy of conclusions drawn from this data. Comparing the data transferred to and from each node in Figure 2 with ground-truth data derived from the system-centric (rather than transfer-centric) monitoring data available on each system yields Figure 3, which shows the amount of bytes into and out of each storage system relative to these ground-truth average daily I/O volumes.

Most transfers to and from the three file systems (cscratch, project, and the burst buffer) follow the storage-compute and compute-storage motif, and we rely on Darshan logs to quantify these transfers. As a result, the degree to which jobs at NERSC generate Darshan logs limits how accurately we can represent storage system traffic along these vectors. Studies have shown that only 40% of core-hours are captured by Darshan logs at NERSC [28] as a result of users opting out of Darshan, not using MPI, or not calling `MPI_Finalize`, and we expect these factors to underrepresent the true volume transferred along these paths. Development and administrative efforts are underway to improve future Darshan coverage [29] to address this issue.

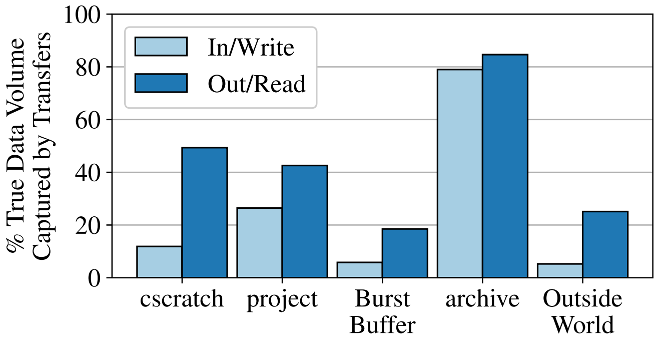

Fig. 3. Completeness of data volumes reported in Figure 2 calculated by comparing the sum of all edges into and out of each node to the ground-truth bytes into and out of each storage system.

Conversely, workloads that make heavy use of client-local page cache may result in spuriously high transfer volumes since Darshan is unable to distinguish I/O that results in a local cache access from I/O that requires a network RPC to the storage system. Thus, while Darshan logs accurately reflect the directionality of transfers, supplementing Darshan logs with an OS-aware tool such as RUR [30], procmon [31], or LDMS [32] could allow a more accurate representation of bytes transferred over the network by quantifying the I/Os that result in both hits and misses of the page cache. For this study, however, it is difficult to assert how much overrepresentation of transfer volume from caching effects offsets underrepresentation of transfer volume from incomplete Darshan coverage.

The burst buffer storage system does not implement client page cache by default, so its transfer volume should not be susceptible to cache-related overrepresentation. However, we find its fraction of missing transfer volume to be even greater than that of cscratch; the reason is that the DataWarp software does not expose detailed logging of its asynchronous data staging activity in any straightforward way. Because DataWarp's asynchronous staging API is the principal mechanism by which users efficiently transfer data between their burst buffer allocation and a persistent storage system, we expect that this lack of staging transfers contributes to the large fraction of missing transfer volume for the burst buffer. In addition, this contributes to some of the lost coverage in cscratch, since cscratch is the only file system that the burst buffer's asynchronous staging transfers can target.

Of similarly low coverage are data transfers to the outside world. Because we capture only external transfers that use Globus or an HPSS interface, the fact that we can account for only between 5% and 25% of the external transfer volume provides strong evidence that a significant amount of data transferred to and from NERSC are either not file based (e.g., use the external-compute/compute-external vector) or do not involve using Globus or HPSS. This fraction is surprisingly low compared with the 40%–65% Globus coverage estimated across all U.S. Department of Energy laboratories [22], but it may reflect NERSC's workload diversity which includes several key experimental workflows that use third-party (non-Globus) tools to transfer data to and from NERSC [1], [12], [33]. Quantifying the degree to which these tools are being used to transfer data is possible by sampling network activity, but such techniques are insufficient to precisely construct discrete transfer records. It would be advantageous to adapt existing tools such as Darshan to provide always-on profiling of the wide-area transfer tools that users are employing in addition to Globus.

Transfers involving the archive are the most completely captured transfers largely because HPSS does not expose data to users through a standard file system interface. Thus, users must interact directly with data transfer daemons, all of which log every file transfer, and the totality of missing transfer volume shown in Figure 3 can be attributed to HPSS transfers that used a specific protocol (Globus) along a specific transfer vector (storage-external) that was not included in this study. Furthermore, the semantics of the HPSS interfaces all map well to a transfer-based model of I/O since they generally do not allow random or partial access to the data objects they store. Object stores' PUT/GET object semantics are similar, and hence object stores would fit well into this transfer-based analysis of data motion. That said, object semantics pose a usability barrier since they limit the ways in which users can access and manipulate data, and any hierarchical storage management interfaces that provide file-based semantics for object-backed data will likely reduce our ability to identify every discrete object transfer event.

By comparing the data volume into and out of each node, we can also identify nodes in the data path where transfer records are missing. We define the *incongruency* of each node through which data can transfer as follows:

$$\Delta = \frac{|V_{in} - V_{out}|}{0.5 \cdot (V_{in} + V_{out})} \tag{1}$$

where V_{in} and V_{out} are the bytes into and out of each node, respectively. This is a figure of merit designed to enable relative comparisons among nodes that handle different data volumes in such a way that larger values indicate that more data is entering the node than leaving it (i.e., the node is a data sink) or, conversely, more data is leaving the node than is entering it (the node is a data source). Incongruency is a particularly meaningful quantity for nodes that are expected to be conservative by virtue of the fact that they moderate data transfers instead of sourcing or sinking data. For these conservative nodes, the incongruency should be zero, and deviations from this reflect errors due to uncaptured transfers or data being generated or stored by the node.

Figure 4 shows these incongruencies for non-storage nodes; unsurprisingly, the Cori node is nonconservative because, despite not being a storage system, jobs generate and consume data in inequal proportions. The slight incongruency of HPSS is consistent with the fact that we know that we are missing a subset of HPSS transfers as shown in Figure 3, and the small incongruency for the data transfer nodes indicates high self-consistency in the Globus logs produced by those nodes.

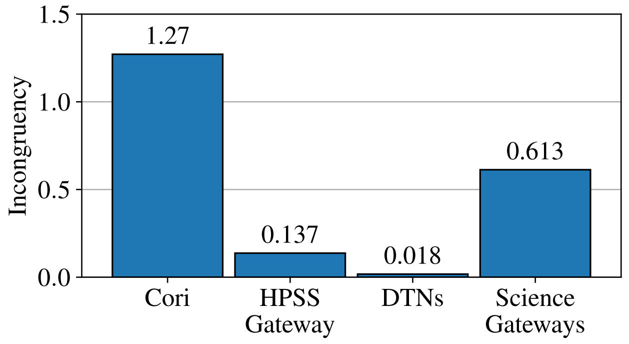

Fig. 4. Incongruency of nodes for stateless components in the data path depicted in Figure 2.

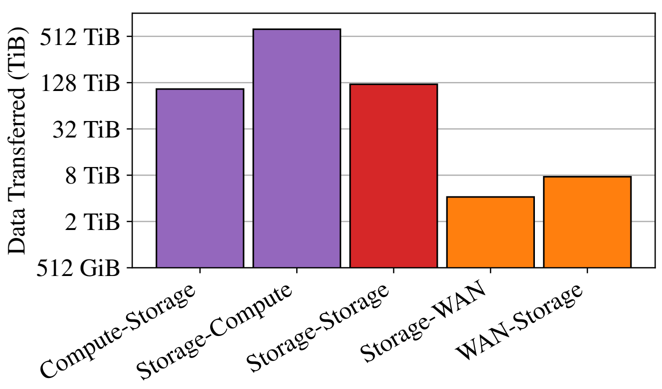

Fig. 5. Average daily I/O volumes for each transfer vector as defined in Figure 1. Because transfer volumes for each vector are derived from different data sources, the errors in each bar are different; see discussion in Section III-B.

Science gateways' high incongruency is unexpected, though; although the absolute volume of data flowing to and from these nodes is small, as shown in Figure 2, the high incongruency of these nodes indicates that they are the source or sink of data that is only partially captured by our transfer records. While this is not entirely unexpected given the wide diversity of roles these nodes play, it does indicate that NERSC's science gateways are not exclusively passive moderators of internal data transfers. Thus, determining whether incongruency is the result of missing records or unexpected data generation or storage requires an understanding of node usage that is not captured by transfer records alone, and further work is required to account for all sources of incongruency.

C. Site-wide Transfer Behavior

Given the known limitations caused by incomplete coverage of data transfers described in Section III-B, we can still draw new insight from the data. For example, we now have a quantitative basis to compare storage-storage and storage-external transfers with compute-storage transfers. Figure 5 contrasts the absolute volumes of each storage vector and is obtained by mapping every edge in Figure 2 to a storage vector. It quantifies the qualitative observation from Figure 2

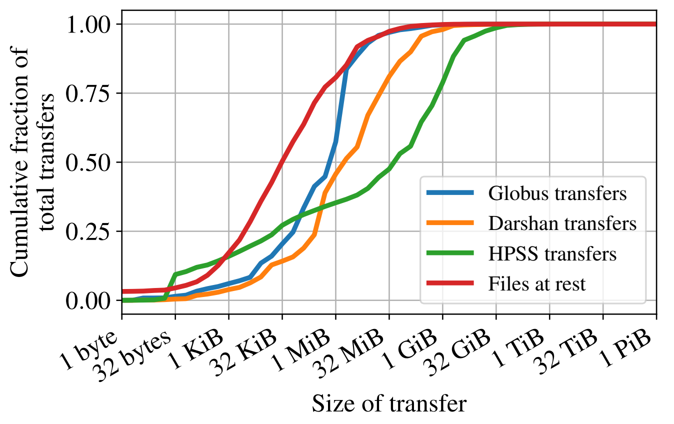

Fig. 6. Cumulative distribution function of data transfer sizes between May 1, 2019 and August 1, 2019.

that storage-compute transfers (i.e., compute jobs reading data from a file system) constitute the majority of bytes transferred within NERSC.

Less obvious from Figure 2 is the fact that storage-storage transfers are of comparable volume to compute-storage transfers. This shows that the need to move data between storage tiers can be as data-intensive as compute jobs themselves are and that storage-storage transfers consume a significant amount of time and network resources in the context of the entire data center. It further highlights the potential benefits of eliminating storage tiers to minimize unnecessary data motion and, combined with the high asymmetry of the compute-storage and storage-compute vectors (6×), challenges conventional wisdom as to the write-intensiveness of HPC I/O. Write-intensive defensive checkpoint I/O is not necessarily the primary driver of storage system traffic. Our study suggests that no canonical I/O workload for HPC applications exists; rather, HPC data center traffic depends on the precise workload mix and storage tiers available.

This conventional view of I/O workloads in HPC has resulted in compute-storage/storage-compute transfers being the subject of considerable optimization. As such, we expect compute-storage transfers to generally demonstrate I/O patterns that are well suited to parallel file systems. Given the relative importance of the other transfer vectors shown in Figure 5, however, storage-storage, storage-WAN, and WAN-storage vectors may warrant an equally critical examination.

Because storage-storage and storage-WAN transfers tend to be nonblocking (e.g., Globus transfers are “fire and forget,” burst buffer staging happens asynchronously, and HPSS data movement is managed through a batch queue at NERSC), we hypothesize that users may have less incentive to follow best practices for high-performance parallel I/O and instead initiate small-file transfers that are known to cause suboptimal end-to-end performance [5], [23].

We investigated this hypothesis by calculating the cumulative distribution of data transfer sizes for each transfer type, and the result is shown in Figure 6. Compute-storage transfers generally target larger files, with 50% of all such transfers being of 940 KiB or larger. Globus transfers show similar behavior, with 50% of transfers being 660 KiB or larger, suggesting that data typically transferred via Globus may be subjected to minimal reduction, aggregation, or manipulation between the time it is accessed by parallel applications and the time it is transferred elsewhere. Notably, this median transfer size is an order of magnitude larger than the transfer data presented by Liu et al [22], suggesting either a significant coarsening of wide-area transfer granularity in the past year or a marked deviation between the NERSC workload and the greater national workload.

By comparison, the majority of HPSS transfers target files that are 37 MiB or larger. This may be a reflection of general user sophistication or experience; HPSS presents an object-like interface, and small-file access to HPSS can be orders of magnitude slower than a disk-based file system. Alternatively, these results may be limited by absolute performance, since it is impossible to read a significant amount of small-file data from HPSS in a fixed period of time because of the high latency of tape retrieval. Of the 10.5 PiB of data transfers involving HPSS, however, 60% were writes on average. Because HPSS uses a high-bandwidth disk cache to buffer incoming data, the relative absence of small transfers in Figure 6 is likely the result of user intent, not absolute performance limitation.

When compared with the file size distribution on the file systems at NERSC, Figure 6 shows a sharp contrast between the files that users *move* and those that remain resident on the storage systems: half of the total files on the file system are smaller than 36 KiB. This may be the result of NERSC’s capacity-focused data management policy that does not incentivize users to clean up small files; because NERSC storage systems are generally constrained by capacity rather than inodes, purging, deleting, or migrating large files is the most effective way to stay under allocated resource limits. In addition, the nontrivial number of 0-byte and 1-byte files suggests that many tiny files are kept resident not for the data they store but perhaps as file-based metadata markers or accidental files that have no value. The cost of transferring these tiny files is likely to outweigh the cost of regenerating them on other storage systems, so users are less likely to consciously transfer them to other storage systems.

D. *Understanding Overall User-level Activity*

We have demonstrated the value of differentiating transfer vectors to determine the relative importance of different data paths and areas for potential optimization. In addition to this center-wide characterization of data movement, however, we can attribute individual transfers to the users who own them, and we can develop an understanding of how individual users affect the aggregate behavior presented in Section III-C. To this end, Figure 7 shows the cumulative distribution of per-user data volume along with the protocols used. Of the 991 unique users who were observed, we can see that the majority of volume transferred can be attributed to just three users (labeled Amy, Bob, and Carol) and the remaining 988

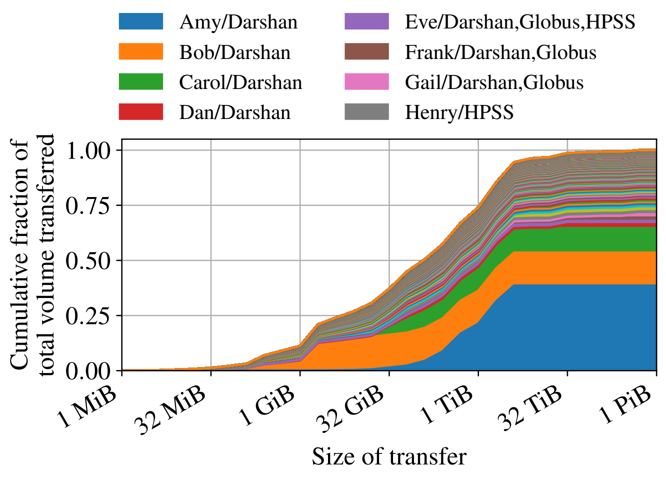

Fig. 7. Cumulative distribution function of data transfer sizes between May 1, 2019 and August 1, 2019 by user. Transfer method(s) employed by the top eight users are shown; and “HPSS” is used to denote the sum of transfers using HSI, HTAR, HPSS FTP, and HPSS pFTP. Note that this CDF is weighted by volume, whereas Figure 6 is weighted by transfer count.

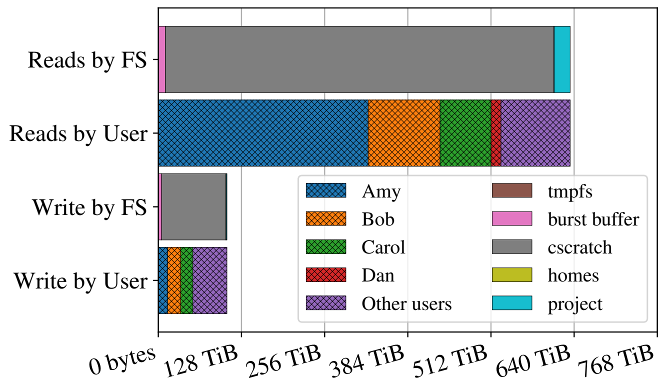

Fig. 8. Distribution of users and storage systems constituting all the compute-storage and storage-compute transfers shown in Figure 5.

users transfer diminishingly small amounts of data overall. All three top users’ transfers use the storage-compute/compute-storage vector, as evidenced by their exclusive appearance in Darshan files. Furthermore, the contributions of all three appear at relatively large transfer sizes, indicating that they are all transferring multiple gigabytes per file.

Our finding that there is an unusually high storage-compute (read) workload (Figure 5) and that most of the transferred volume is owned by Amy, Bob, and Carol (Figure 7) can be further enhanced by decomposing all data transfers by a combination of user and storage system. Figure 8 shows this breakdown and reveals that the overwhelming majority of daily traffic along the storage-compute and compute-storage vectors targets the cscratch file system, countering an intuitive belief that these high read rates may reflect heavy use of the burst buffer. Furthermore, the user-level breakdown of transfers by transfer vector direction shows that the abnormally high transfer volumes owned by Amy, Bob, and Carol are restricted to read (storage-compute) transfers. By comparison, the overall write (compute-storage) transfer volume is more evenly distributed, with Amy, Bob, and Carol representing a minority of transfer volume along that vector.

From this drill down, we have determined that the 6× asymmetry between storage-compute and compute-storage transfers qualitatively illustrated in Figure 2 and quantitatively shown in Figure 5 is indeed anomalous to the degree that it is the result of a small number of users’ workflows that are read-intensive and target one specific storage system; the presence or absence of those users on the system will dramatically affect the observed read/write imbalance on any given day.

We can also leverage fine grained data analysis to gain greater insight into the behavior of the largest data consumers. By knowing that Amy, Bob, and Carol are exclusively using compute-storage transfers and that those transfer records are derived from Darshan logs, we can capitalize on the fact that Darshan stores both the total number of bytes read and written to every file and the maximum offset within each file to which I/O was issued. Comparing the count of total bytes read with the maximum offset read for each file read by each user, we find that Amy, Bob, and Carol were rereading the same files 140, 50, and 580 times on average, respectively.

Whether this excessive rereading of the same files was intentional or not is unclear, but the combination of high reread activity and the knowledge that these I/Os were targeting a Lustre file system indicates that these workflows were making heavy use of the Lustre client page cache. Hence, some component of the high read skew shown in Figure 8 probably did not require network transfers, and the resulting load on the file system was only a fraction of the transfer workload observed by Darshan. However, the ground-truth data in Figure 4 shows a similarly disproportionate read coverage, and therefore some component of the heavy read workload from these users did reach the cscratch storage system itself despite the availability of client caching.

E. Tracing Data within Workflows

In addition to identifying the largest users of a single transfer vector, the transfer data also allow us to trace how a single user’s data moves throughout the data center over time. Figure 9 shows all the different data transfers attributed to another individual user over time; we generate this time series by assuming a constant transfer rate over the duration of each transfer and applying its transfer volume over the entire length of time that transfer was active. In this case, the user demonstrates an intuitive pattern of data movement during a workflow that involves coupling data movement between tiers to reading from and writing to the compute system.

Although not explicitly resolved in Figure 9, the shaded region at higher temporal resolution reveals that this user always performs a storage-storage data movement before issuing compute-storage and storage-compute transfers of comparable volume. This indicates that the workflow is moving data from a colder tier to a warmer tier before being analyzed. The ratio of data volumes transferred reflects a 2:1 read:write ratio

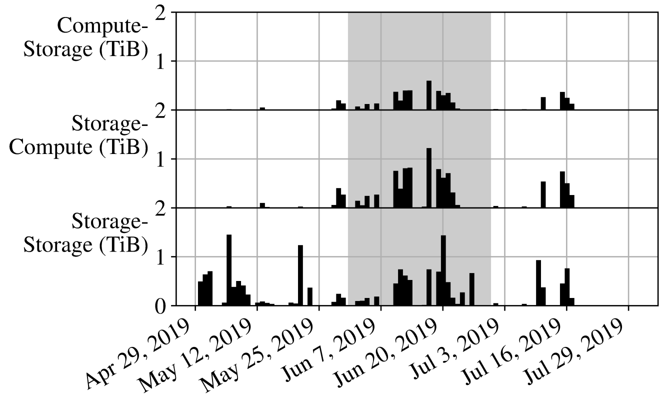

Fig. 9. Data transfers from a single user over a three-month period grouped by transfer vector direction. The period from June 1 to July 1 (shaded) shows high correlation across all vectors.

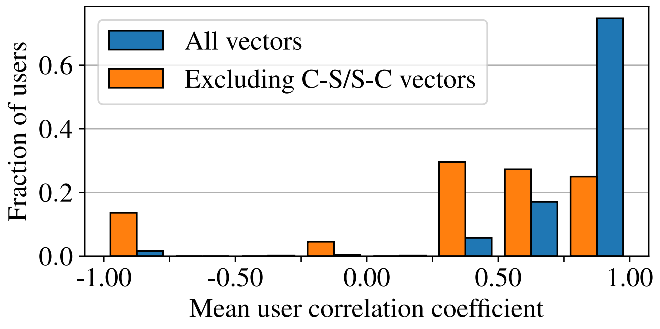

Fig. 10. Distribution of mean correlations for all users that transferred data along multiple vectors. "Excluding C-S/S-C vectors" are the mean user transfer correlations when the compute-storage and storage-compute vectors are disregarded. High coefficients suggest highly regular behavior whereby a data transfer along one vector is usually accompanied by a transfer along another vector within the same calendar day.

during computation, further suggesting that this workflow is performing a data reduction.

The Pearson correlation between transfers occurring along each of the three vectors shown in Figure 9 is high during the shaded region ($R = 0.778$). However, the correlation between daily compute-storage and storage-storage transfers diminishes considerably ($R = 0.360$) for this same user when calculated over the full three-month window. Thus, while the components of this user's workflow that involve compute-storage and storage-compute transfers are tightly coupled to storage-storage transfers, other pieces of the larger end-to-end workflow involve storage-storage transfers that are not followed by computation. This lack of predictability over longer time scales is likely a reflection of a human in the loop of an otherwise highly regular workflow.

To determine whether this workflow regularity is the rule or the exception for user behavior, we can generalize our correlation analysis to all users by calculating correlation coefficients between each vector for each user (as demonstrated in Figure 9) and then taking the mean of those vector-pair correlations on a per-user basis. We call this the "mean user transfer correlation," and it is a figure of merit that describes how often a user transfers data along two or more transfer vectors within a single day—or, broadly speaking, how tightly coupled a user's data transfers are throughout the data center. Of the 1,562 unique users that transferred data during the time studied, 439 transferred data only along one vector and therefore could not be attributed a mean user transfer correlation. Of the remaining 1,123 users, only 486 of them showed any statistically significant correlation between any of the transfer vectors they employed, indicating that the relatively strong correlation shown by the user in Figure 9 is not the typical case.

Of those 486 users, a significant number (75%) showed strong mean user transfer correlations, as shown in Figure 10. However, decomposing these mean user transfer correlations into correlations between individual vector pairs reveals that the strongest correlations for most of these users occurred between compute-storage and storage-compute transfers. In the context of application workflow, this result is not surprising; we expect that applications that perform compute-storage transfers (i.e., write data) will also perform storage-compute transfers (i.e., read data) within the same day. When we omit the compute-storage:storage-compute pair when calculating mean user transfer correlations, however (also shown in Figure 10), we find that only 44 users of the total 486 show strongly correlated behavior between different transfer vectors.

We thus establish that users generally do not exhibit high regularity over long periods of time outside of the intuitive correlation between users' applications reading and writing on the same calendar day. This unpredictability in data transfers along different vectors suggests a high degree of user interactivity in initiating data transfers outside of any automated processes that may be moderating data transfers between individual jobs.

IV. Conclusion

Characterizing the I/O demands of scientific workflows using data transfers enables new insights into the ways in which scientific workflows utilize resources across the entire data center. Users structure data transfers in markedly different ways from how they retain data at rest in that data transfers tend to involve larger files while data at rest comprise many smaller files. The amount of data transferred between storage tiers is also significant; during the three months examined, the amount of data moved between storage tiers outside of jobs was approximately equal to the volume of data written from HPC jobs to file systems. These results indicate the need for better I/O monitoring of data transfer tools that operate between jobs within scientific workflows.

Users also read far more data than they wrote from their compute jobs during this time, defying the notion of HPC I/O workloads being characteristically write-heavy; this unexpectedly high read-write ratio was found to be the result of three extremely read-intensive users rereading their files dozens to hundreds of times. Moreover, time-resolved tracing of data motion between tiers revealed that one can identify strong correlations between users running data-intensive computational jobs and staging data in a highly predictable way, but the majority of users do not operate exclusively in this mode. Rather, transfer traces suggest that tightly coupled data transfers and compute jobs are interspersed with data transfers that are indicative of a human in the loop, suggesting that few workflows at NERSC are automated at present.

We made several notable observations in the course of this study, but it is difficult to determine how generally applicable they are due to the amount of unaccounted transfer data. In future work, we plan to increase the breadth of our telemetric coverage so that we can compare trends over longer periods of time and across multiple data centers.

Acknowledgments

We thank Damian Hazen, Kristy Kallback-Rose, Lisa Gerhardt, Ravi Cheema, and Nicholas Balthaser for providing insight into and data from the NERSC HPSS archive and DTN resources.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contracts DE-AC02-05CH11231 and DE-AC02-06CH11357. This research used resources and data generated from resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 and the Argonne Leadership Computing Facility, a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

References

[1] S. Parete-Koon, B. Caldwell, S. Canon, E. Dart, J. Hick, J. Hill, C. Layton, G. Pelfrey, D. Shipman, H. Skinner, J. Nam, and J. Wells, "HPC's Pivot to Data," in *Proceedings of the 2014 Cray Users Group*, 2014. [Online]. Available: https://cug.org/proceedings/cug2014_proceedings/includes/files/pap151.pdf

[2] J. Thayer, D. Damiani, C. Ford, I. Gaponenko, W. Kroeger, C. O'Grady, J. Pines, T. Tookey, M. Weaver, and A. Perazzo, "Data Systems for the Linac Coherent Light Source," *Journal of Applied Crystallography*, vol. 49, no. 4, pp. 1363–1369, Aug. 2016. [Online]. Available: http://scripts.iucr.org/cgi-bin/paper?S1600576716011055

[3] D. Y. Parkinson, K. Beattie, X. Chen, J. Correa, E. Dart, B. J. Daurer, J. R. Deslippe, A. Hexemer, H. Krishnan, A. A. Macdowell, F. R. Maia, S. Marchesini, H. A. Padmore, S. J. Patton, T. Perciano, J. A. Sethian, D. Shapiro, R. Stromsness, N. Tamura, B. L. Tierney, C. E. Tull, and D. Ushizima, "Real-Time Data-Intensive Computing," *AIP Conference Proceedings*, vol. 1741, pp. 1–6, 2016.

[4] G. K. Lockwood, D. Hazen, Q. Koziol, S. Canon, K. Antypas, J. Balewski, N. Bathaser, W. Bhimji, J. Botts, J. Broughton, T. L. Butler, G. F. Butler, R. Cheema, C. S. Daley, T. Declerck, L. Gerhardt, W. E. Hurlbert, K. A. Kallback-Rose, S. Leak, J. Lee, R. Lee, J. Liu, K. Lozinskiy, D. Paul, Prabhat, C. Snavely, J. Srinivasan, T. Stone Gibbins, and N. J. Wright, "Storage 2020: A Vision for the Future of HPC Storage," Lawrence Berkeley National Laboratory, Berkeley, CA, Tech. Rep., 2017. [Online]. Available: https://escholarship.org/uc/item/744479dp

[5] R. Kettimuthu, Z. Liu, D. Wheeler, I. Foster, K. Heitmann, and F. Cappello, "Transferring a Petabyte in a Day," *Future Generation Computer Systems*, vol. 88, pp. 191–198, 2018.

[6] P. Carns, K. Harms, W. Allcock, C. Bacon, S. Lang, R. Latham, and R. Ross, "Understanding and Improving Computational Science Storage Access through Continuous Characterization," in *2011 IEEE 27th Symposium on Mass Storage Systems and Technologies (MSST)*. IEEE, May 2011, pp. 1–14. [Online]. Available: http://ieeexplore.ieee.org/document/5937212/

[7] J. Lofstead and R. Ross, "Insights for Exascale IO APIs from Building a Petascale IO API," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '13*, 2013, pp. 1–12. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2503210.2503238

[8] H. Luu, M. Winslett, W. Gropp, R. Ross, P. Carns, K. Harms, Prabhat, S. Byna, and Y. Yao, "A Multiplatform Study of I/O Behavior on Petascale Supercomputers," in *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing - HPDC '15*. New York, New York, USA: ACM Press, 2015, pp. 33–44. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2749246.2749269

[9] Y. Liu, R. Gunasekaran, X. Ma, and S. S. Vazhkudai, "Server-side Log Data Analytics for I/O Workload Characterization and Coordination on Large Shared Storage Systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC'16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 70:1–70:11. [Online]. Available: http://dl.acm.org/citation.cfm?id=3014904.3014998

[10] S. Madireddy, P. Balaprakash, P. Carns, R. Latham, R. Ross, S. Snyder, and S. M. Wild, "Analysis and Correlation of Application I/O Performance and System-Wide I/O Activity," in *2017 International Conference on Networking, Architecture, and Storage (NAS)*. IEEE, Aug. 2017, pp. 1–10. [Online]. Available: http://ieeexplore.ieee.org/document/8026844/

[11] G. K. Lockwood, S. Snyder, T. Wang, S. Byna, P. Carns, and N. J. Wright, "A Year in the Life of a Parallel File System," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '18. Piscataway, NJ, USA: IEEE, Nov. 2018, pp. 931–943. [Online]. Available: http://dl.acm.org/citation.cfm?id=3291656.3291755 https://ieeexplore.ieee.org/document/8665806/

[12] J. Deslippe, A. Essiari, S. J. Patton, T. Samak, C. E. Tull, A. Hexemer, D. Kumar, D. Parkinson, and P. Stewart, "Workflow Management for Real-Time Analysis of Lightsource Experiments," in *2014 9th Workshop on Workflows in Support of Large-Scale Science*. IEEE, Nov. 2014, pp. 31–40. [Online]. Available: http://ieeexplore.ieee.org/document/7019860/

[13] "APEX Workflows Whitepaper," Los Alamos National Laboratory, Lawrence Berkeley National Laboratory, and Sandia National Laboratories, Tech. Rep., 2016. [Online]. Available: https://www.nersc.gov/assets/apex-workflows-v2.pdf

[14] C. S. Daley, L. Ramakrishnan, S. Dosanjh, and N. J. Wright, "Analyses of Scientific Workflows for Effective Use of Future Architectures," in *Proceedings of the 6th International Workshop on Big Data Analytics: Challenges, and Opportunities (BDAC-15)*, Austin, TX, 2015.

[15] C. S. Daley, D. Ghoshal, G. K. Lockwood, S. Dosanjh, L. Ramakrishnan, and N. J. Wright, "Performance Characterization of Scientific Workflows for the Optimal Use of Burst Buffers," *Future Generation Computer Systems*, Dec. 2017. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0167739X16308287

[16] W. Bhimji, D. Bard, K. Burleigh, C. S. Daley, S. Farrell, M. Fasel, B. Friesen, L. Gerhardt, J. Liu, P. Nugent, D. Paul, J. Porter, and V. Tsulaia, "Extreme I/O on HPC for HEP using the Burst Buffer at NERSC," *Journal of Physics: Conference Series*, vol. 898, p. 082015, oct 2017. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/898/8/082015

[17] H.-B. Chen, G. Grider, and D. R. Montoya, "An Early Functional and Performance Experiment of the MarFS Hybrid Storage EcoSystem," in *2017 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, April 2017, pp. 59–66. [Online]. Available: http://ieeexplore.ieee.org/document/7923787/

[18] W. Schroeder, R. Marciano, J. Lopez, M. Gleicher, G. Kremenek, C. Baru, and R. Moore, "Analysis of hpss performance based on per-file transfer logs," in *16th IEEE Symposium on Mass Storage Systems in cooperation with the 7th NASA Goddard Conference on Mass Storage Systems and Technologies. Information-based Access to Storage: Foundation of Information Systems*. Los Alamitos, CA, USA: IEEE Computer Society, March 1999, pp. 103–115. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/MASS.1999.830000

[19] B. Anderson, M. Genty, D. L. Hart, and E. Thanhardt, "Using Data Science to Understand Tape-Based Archive Workloads," in *Proceedings of the 2015 XSEDE Conference on Scientific Advancements Enabled by Enhanced Cyberinfrastructure - XSEDE '15*, vol. 2015-July. New

York, New York, USA: ACM Press, 2015, pp. 1–8. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2792745.2792776

[20] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The Science DMZ," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '13*. New York, New York, USA: ACM Press, 2013, pp. 1–10. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2503210.2503245

[21] W. E. Allcock, R. Wagner, B. S. Allen, R. Ananthakrishnan, B. Blaiszik, K. Chard, R. Chard, I. Foster, L. Lacinski, and M. E. Papka, "Petrel: A Programmatically Accessible Research Data Service," in *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning) - PEARC '19*. New York, New York, USA: ACM Press, 2019, pp. 1–7. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3332186.3332241

[22] Z. Liu, I. Foster, R. Kettimuthu, and N. S. Rao, "Cross-Geography Scientific Data Transferring Trends and Behavior," in *HPDC 2018 - Proceedings of the 2018 International Symposium on High-Performance Parallel and Distributed Computing*, 2018, pp. 267–278.

[23] Y. Liu, Z. Liu, R. Kettimuthu, N. Rao, Z. Chen, and I. Foster, "Data Transfer between Scientific Facilities – Bottleneck Analysis, Insights and Optimizations," in *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, may 2019, pp. 122–131. [Online]. Available: https://ieeexplore.ieee.org/document/8752877/

[24] T. Declerck, K. Antypas, D. Bard, W. Bhimji, S. Canon, S. Cholia, and Y. H. He, "Cori - A System to Support Data-Intensive Computing," in *Proceedings of the 2016*, London, 2016. [Online]. Available: https://cug.org/proceedings/cug2016_proceedings/includes/files/pap171s2-file2.pdf

[25] R. A. Coyne, H. Hulen, and R. Watson, "The high performance storage system," in *Proceedings of the 1993 ACM/IEEE conference on Supercomputing - Supercomputing '93*. New York, New York, USA: ACM Press, 1993, pp. 83–92. [Online]. Available: http://portal.acm.org/citation.cfm?doid=169627.169662

[26] J. Dugan, "ESxSNMP: ESnet eXtensible SNMP System," in *Summer JointTechs 2010*, Columbus, OH, 2010. [Online]. Available: https://www.internet2.edu/presentations/jt2010july/20100713-Dugan-ESxSNMP.pdf

[27] M. Rocklin, "Dask: Parallel computation with blocked algorithms and task scheduling," in *Proceedings of the 14th Python in Science Conference*, K. Huff and J. Bergstra, Eds., 2015, pp. 130 – 136.

[28] T. Wang, S. Byna, G. K. Lockwood, S. Snyder, P. Carns, S. Kim, and N. J. Wright, "A Zoom-in Analysis of I/O Logs to Detect Root Causes of I/O Performance Bottlenecks," in *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE, May 2019, pp. 102–111. [Online]. Available: https://ieeexplore.ieee.org/document/8752753/

[29] S. Snyder, P. Carns, K. Harms, R. Ross, G. K. Lockwood, and N. J. Wright, "Modular HPC I/O characterization with Darshan," *Proceedings of the 5th Workshop on Extreme-Scale Programming Tools (ESPT '16)*, pp. 9–17, 2016.

[30] T. L. Butler, "Using Resource Utilization Reporting to Collect DVS Usage Statistics," in *Proceedings of the 2014 Cray User Group*, 2014.

[31] D. M. Jacobsen, "procmon: Real-Time Process Monitoring on the Cray," in *Proceedings of the 2015 Cray User Group*, 2015.

[32] A. Agelastos, B. Allan, J. Brandt, P. Cassella, J. Enos, J. Fullop, A. Gentile, S. Monk, N. Naksinehaboon, J. Ogden, M. Rajan, M. Showerman, J. Stevenson, N. Taerat, and T. Tucker, "The Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications," in *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, Nov. 2014, pp. 154–165. [Online]. Available: http://ieeexplore.ieee.org/document/7013000/

[33] R. Gerber, W. Allcock, C. Beggio, S. Campbell, A. Cherry, S. Cholia, E. Dart, C. England, T. Fahey, F. Foertter, R. Goldstone, J. Hick, D. Karelitz, K. Kelly, L. Monroe, Prabhat, D. Skinner, and J. White, "DOE High Performance Computing Operational Review (HPCOR): Enabling Data-Driven Scientific Discovery at HPC Facilities," Lawrence Berkeley National Laboratory (LBNL), Berkeley, CA (United States), Tech. Rep., Oct. 2014. [Online]. Available: http://www.osti.gov/scitech/servlets/purl/1163236 http://www.osti.gov/servlets/purl/1163236/