Parallel I/O Evaluation Techniques and Emerging HPC Workloads: A Perspective

Sarah Neuwirth
Institute of Computer Science
Goethe-University Frankfurt, Germany
s.neuwirth@em.uni-frankfurt.de

Arnab K. Paul
National Center for Computational Sciences
Oak Ridge National Laboratory, USA
paula@ornl.gov

***Abstract*—Emerging workloads such as artificial intelligence, big data analytics and complex multi-step workflows alongside future exascale applications are anticipated future HPC workloads, which will result in a more diverse I/O system workload and even less predictable I/O behavior and access patterns. Along with the ever increasing gap between the compute and storage performance capabilities, the in-depth understanding of extreme-scale I/O behavior and the I/O performance modeling and prediction are essential tools of the large-scale I/O evaluation process for addressing the needs of extreme-scale hybrid workloads. In this survey article, we focus on the state-of-the-art of the I/O behavior and performance analysis process for HPC systems in a 5-year time window and identify future research challenges.**

***Keywords*-Parallel File Systems and Storage, Large-scale I/O, HPC, Performance Evaluation, I/O Characterization**

I. Introduction

High Performance Computing (HPC) applications are evolving to include not only traditional scale-up modeling and simulation bulk-synchronous workloads but also scale-out workloads like artificial intelligence (AI), data analytics methods, deep learning, big data and complex multi-step workflows [1]. Exascale workflows are projected to include multiple different components from both scale-up and scale-out communities operating together to drive scientific discovery and innovation.

With the often conflicting design choices between optimizing for write-intensive vs. read-intensive workloads, having flexible I/O systems will be crucial to support these emerging hybrid workloads. Another performance aspect is the intensifying complexity of parallel file and storage systems in large-scale cluster environments. Storage system designs are advancing beyond the traditional two-tiered file system and archive model by introducing new tiers of temporary, fast storage close to the computing resources with distinctly different performance characteristics. The changing landscape of emerging hybrid HPC workloads along with the ever increasing gap between the compute and storage performance capabilities reinforce the need for an in-depth understanding of extreme-scale I/O and for rethinking existing data storage and management evaluation techniques and strategies.

In this article, we present a holistic survey of the current research on large-scale I/O evaluation and characterization techniques in the context of HPC systems. Our contributions are two-fold. First, we provide a detailed taxonomy of I/O performance evaluation strategies together with a snapshot of the current I/O analysis, modeling, and prediction research. We believe this knowledge is useful to the whole scientific community, as applications often observe poor performance due to bottlenecks in the parallel I/O system. Second, we aim to identify current and future research challenges with regard to the emerging exascale computing systems and more complex hybrid HPC workloads. We focus on a 5-year time window from 2015 to 2020. By exploring the most recent publications of the last five years, we aim at answering the following questions:

- What are the main techniques used by HPC researchers for characterizing and analyzing the I/O behavior?
- What effects do emerging workloads have on current research strategies?
- What are the key I/O research topics in the evaluation of emerging HPC workloads at large-scale?

II. Background and Related Work

Proportional to the scale increases in HPC systems, many scientific applications are becoming increasingly data-intensive, and parallel I/O has become one of the dominant factors impacting large-scale application performance.

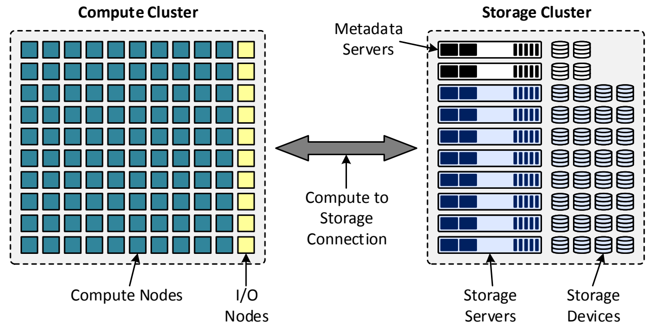


Fig. 1: HPC system with a center-wide parallel file system.

Figure 1 presents an overview of a typical HPC compute cluster backed by a center-wide parallel file system. The compute nodes run the client applications and are typically connected over a high-performance network fabric such as InfiniBand. I/O nodes handle requests forwarded by the scientific applications, potentially integrate a tier of solid-state devices to absorb the burst of random or high volume operations, so that transfers to/from the staging area from/to the traditional parallel file system can be done more efficiently. The connection to the storage cluster is often times through a secondary, slower fabric such as 10GB Ethernet. The storage cluster hosts the server-side of the parallel file system and usually comprises of metadata servers, storage servers, and storage devices.

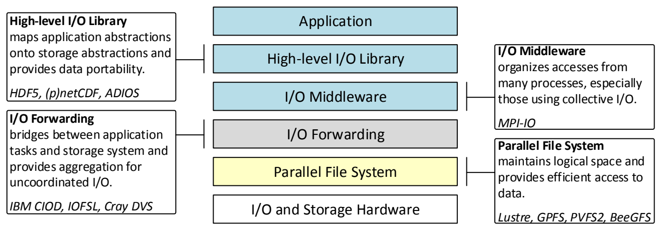

High-level I/O Library
maps application abstractions onto storage abstractions and provides data portability.
HDF5, (p)netCDF, ADIOS
I/O Forwarding
bridges between application tasks and storage system and provides aggregation for uncoordinated I/O.
IBM CIOD, IOFSL, Cray DVS
Application
High-level I/O Library
I/O Middleware
I/O Forwarding
Parallel File System
I/O and Storage Hardware
I/O Middleware
organizes accesses from many processes, especially those using collective I/O.
MPI-IO
Parallel File System
maintains logical space and provides efficient access to data.
Lustre, GPFS, PVFS2, BeeGFS

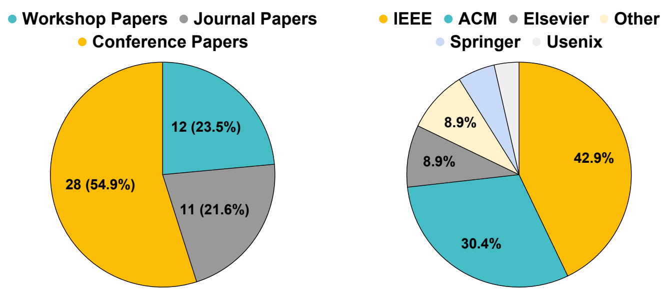
Fig. 2: Parallel I/O architecture [2].

As depicted in Figure 2, parallel I/O systems are inherently complex, particularly in the context of end-to-end I/O paths. For example, at the starting point of a typical I/O path, an application can use a high-level library such as HDF5, for various reasons including portability and improved data management. HDF5 is implemented on top of MPI-IO which, in turn, performs POSIX I/O calls against a parallel file system, such as Lustre. Furthermore, before an I/O request reaches its eventual storage device, it may need to traverse through the compute fabric, and a large-scale storage network fabric.

Hence, providing a comprehensive survey and overview on large-scale I/O evaluation techniques in the context of HPC is invaluable due to the ever increasing gap between the compute and storage performance capabilities. Other surveys such as Boito et al. [3] present an overview about parallel I/O on HPC systems and mainly focus on I/O optimization approaches. Chapp et al. [4] provide an overview of record-and-replay techniques for HPC systems in the context of debugging, reproducibility, and fault-tolerance. The most recent survey by Gupta et al. [5] focuses on open-source tools for I/O monitoring. To the authors' best knowledge, there has been no holistic taxonomy or survey presented targeting the complete life cycle of large-scale I/O performance evaluation and characterization.

III. Survey Technique

In this article, we have followed different guidelines [6], [7] for undertaking a brief methodical survey that focuses on large-scale I/O performance evaluation related research published between 2015 and 2020. We identified articles based on the following process stages: (1) define search keywords, (2) perform search based on keywords, (3) evaluate articles based on abstract and conclusion, (4) exclude articles addressing the same research with common challenge/references, (5) include remaining articles. In the following, we present the source of information and selection criteria.



(a) Paper type distribution. **(b)** Publisher distribution.

Fig. 3: Percentage distribution of included papers.

A. Source of Information

We broadly searched for journal, conference, and workshop research articles in the following databases and search engines:

- IEEE Explore (https://ieeexplore.ieee.org/)
- ACM Digital library (https://dl.acm.org/)
- Springer (https://link.springer.com/)
- ScienceDirect (https://www.sciencedirect.com/)
- Google Scholar (https://scholar.google.com/)

B. Search Criteria and Selection

We defined and used the following keywords for the search in the aforementioned databases: *HPC, large-scale I/O performance evaluation, workload replication and generation, I/O characterization, workload characterization, I/O analysis, I/O traces, proxy applications, I/O kernel applications,* and *emerging workloads*. In the end, we identified 51 research articles to be included in this overview. Figure 3 presents the percentage distribution of paper types and publishers.

IV. Large-Scale I/O Evaluation

This section provides a taxonomy of the iterative process of large-scale I/O performance evaluation based on its different phases and strategies, as depicted in Figure 4. The taxonomy is complemented by a structured survey of existing research work that has been published between 2015 and 2020. Traditionally, the process of understanding I/O behavior and performance for given applications or storage systems is performed iteratively and empirically in a closed loop fashion. The I/O evaluation cycle consists of three main phases: (1) *Measurements and Statistics Collection*, (2) *Modeling and Prediction*, and (3) *Simulation*. The following sections describe each phase, provide an overview of existing strategies and tools, and outline the interplay and feedback loop between the different phases.

A. Measurements and Statistics Collection

The first phase focuses on the performance measurement and data monitoring either conducted on real-world computing environments or through simulation. The empirical data is needed to characterize the I/O behavior of different workloads and to identify issues; the processing of the obtained data is typically performed by the modeling and prediction phase.

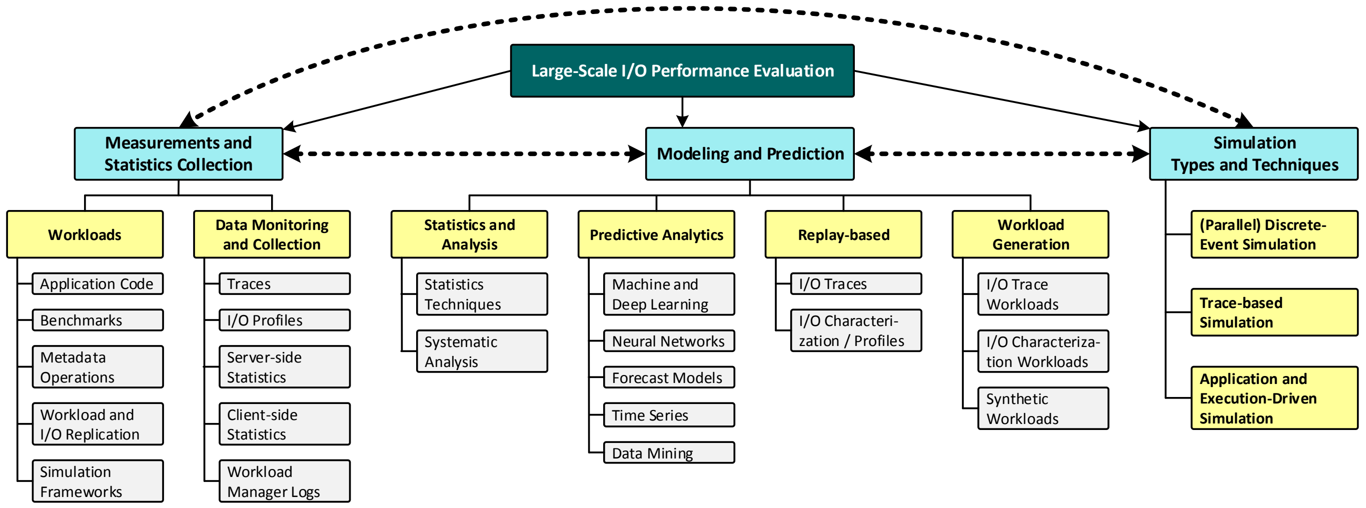


Fig. 4: Phases of the iterative large-scale I/O performance evaluation process. ***Note:*** *The dashed arrows emphasize the iterative performance analysis cycle and feedback loop between different phases.*

1) Workloads: In order to collect I/O performance data, the first step is to characterize the experiment or workload since most applications spend a considerable amount of time gathering input or producing output. Multiple different workload and benchmark strategies can be differentiated:

- ***Application Code:*** The most accurate workload is the application code itself, which cannot always be used or accessed for characterization purposes. Some applications are from domain scientists whose codebase cannot be easily accessed, while others are too huge or take too long to run to profile their I/O requests on smaller testbeds. In these cases, the following strategies can be used.
- ***Benchmarks:*** There is a variety of synthetic and application benchmarks available [8], [9]. *Synthetic benchmarks* try to combine operations in proportions that will yield a representative measure of the actual I/O performance capabilities by mimicking different file access patterns while *application benchmarks* try to mimic the I/O behavior of specific applications.
- ***Metadata Operations:*** Metadata performance can be a limiting factor for parallel file systems. Benchmarks stressing the metadata services such as *mdtest* [8] provide a measure to quantify file and directory based operations.
- ***Workload and I/O Replication:*** Typically, the I/O evaluation is performed in an iterative manner. Results from previous experiments can be used to replicate and mimic the I/O behavior of different applications. *Proxy applications* [10] are manually derived from large-scale application codes and require in-depth understanding and/or access to the source code. *I/O Skeletons* [11]–[14] and auto-generated benchmarks [15] for given applications are created by utilizing a model of the application derived from the properties of its regular diagnostic and/or checkpoint output. An example is the tool *Skel* [14], which generates I/O skeletons for applications that rely on ADIOS to describe the data that may need to be written, read, or processed outside of the running application. Finally, the *Record-and-Replay* strategy relies on a set of lossless and scalable I/O traces or profiles and corresponding replay tools [16]–[19]. Either through dynamic or static I/O analysis, the collected I/O traces are analyzed to extract detailed I/O access information and fed back into replay tools to replicate the I/O behavior of the original application or generate an often times portable application benchmark. The *ScaleIO Framework* [16], [17], for example, can be used to gather I/O traces on a small system, to analyze the traces and extrapolate them, and then finally enable I/O replay to verify the correctness of the projected extrapolation of the I/O behavior.
- ***Simulation Frameworks:*** If researchers do not have access to a real test system, simulator tools and I/O workload simulation [13], [20], [21] can be feasible alternatives. Depending on the used simulation framework, simulations either rely on workload generation based on I/O traces, I/O characterization or synthetic workloads. Snyder et al. [20] provide a detailed discussion about the flexibility, accuracy, and breadth of use of system simulations based on the different workload sources.

2) Data Monitoring and Collection: Besides the workload description, information about the actual workload run needs to be collected in order to characterize the application run and system utilization for further analysis, modeling, and prediction. Here, different types of data and metrics can be of interest. For example, performance metrics such as the time spent in I/O operations, total execution time of an application, or read and write throughput can provide insights on the I/O behavior of an application or storage system.

In principle, there are two ways to collect performance information: ***traces*** and ***profiles***. *Profiles* store I/O characterization information, i.e., statistics, including: number of function invocations, average execution time of a function, file access patterns, or floating-point operations performed.

Darshan [22] and its recent extensions *DXT* [23] and *tf-Darshan* [24] are examples of I/O characterization tools that may be used to derive representative I/O workloads for a given application. Unlike profiles, *traces* record a detailed report of the execution chronology of function and system calls together with a timestamp, which produces much more log data and potentially degrades the system performance while collecting the traces. For example, *Recorder* [25], [26] is a multi-level I/O tracing tool that captures I/O calls at multiple layers of the I/O stack, and *FSMonitor* [27], [28] captures the metadata file system events in storage systems.

In addition to profiles and traces, storage and system system administrators can collect additional ***server-side statistics*** of the file system, e.g., load on the servers and storage devices [29]. Other valuable insights can be gained through the collection of ***client-side hardware statistics*** and ***workload manager logs*** (e.g., from Slurm or TORQUE), if available.

Several different approaches can be identified to capture the I/O performance and characterization of a given workload. For instance, several software implementations of I/O characterization [22], [30]–[33] and tracer [15]–[18], [31], [34]–[37] tools have been proposed to monitor I/O at *job-level*, which focus on the I/O behavior of individual applications. In addition, *storage-system-level* monitoring tools [38]–[40] have been proposed to capture the I/O behavior of the storage system as a whole. Finally, recent work [30], [41]–[46] has proposed to develop all-encompassing and cohesive monitoring systems which can capture *end-to-end I/O behavior* of jobs at each step along their I/O path.

B. Modeling and Prediction

After the initial measurements and data monitoring phase, the empirical performance data needs to be analyzed in order to identify issues, model I/O performance, and predict future I/O behavior for a given application or the storage system. Another aspect of this evaluation phase is the workload generation for further system analysis. The modeling and prediction phase can be divided in four sub-categories.

1) Statistics and Analysis: The traditional modeling approach relies on statistics and data analysis and can be described as the process of extraction of meaningful patterns in data. In simple words, statistics is the science of collecting, classifying, and representing numerical data. Here, data refers to I/O traces, I/O characterization profiles and other logs such as scheduler logs and server-side statistics. Some of the ***statistics techniques*** are arithmetic mean, standard deviation, linear regression, Markov models, hypothesis testing, probability density and cumulative density functions, coefficient of variance, and coefficient of correlation. However, statistics needs extensive efforts of human experts with an in-depth understanding of particular HPC applications or workloads.

When performing a ***systematic analysis***, research studies of I/O behavior can be broadly classified into two categories: (1) focus on the I/O behavior of individual applications [11], [12], [30], [33], [40], [47]–[51], and (2) focus on the I/O behavior of the storage system as a whole [39], [52]–[54]. Analysis work of type (1) describes the I/O behavior of specific applications, such as data transfer rates, I/O periodicity and repetition, and I/O variability of individual jobs. Due to the shared nature of parallel file systems, the I/O behavior of the storage system as a whole is another important performance characteristic. Besides the job-level analysis tools, research work in this area also relies on server-side logs to identify I/O patterns and to correlate the monitoring data of job-level logs with storage-system-level logs. Recent work by Patel et al. [53] introduces the possibility to gain insights about the storage systems through temporal, spatial, and correlative analysis.

2) Predictive Analytics: Predictive analytics is a branch of data analytics to predict future events, but can also be used to identify performance issues and optimization potential. It encompasses a variety of statistical techniques from data mining, predictive modeling, and machine learning that analyze current and historical facts to make predictions about future or otherwise unknown events. With a sufficient amount of training data (e.g., from logs or traces), predictive models can make relatively accurate predictions of the performance, without requiring domain knowledge and human efforts.

Recent work [55]–[58] introduces I/O performance prediction and I/O behavior emulation based on ***deep learning***, ***artificial neural networks***, and ***machine learning***. Schmid and Kunkel [56], for example, use neural networks to analyze and predict file access times of a Lustre file system from the client's perspective, and show that the average prediction error can be significantly improved in comparison to linear models. Sun et al. [57] instrument machine learning to automatically predict the execution and I/O time of MPI applications with different inputs, at different scales, and without domain knowledge. After collecting data from several automatic executions with different inputs, a random forest machine learning approach is used to build an empirical performance model, which is able to predict the execution and I/O time of the program for new input parameters.

3) Replay-based Modeling: Replay-based modeling relies on historical ***I/O traces*** or ***characterization data***, which contain detailed information about computation and I/O behavior of an HPC application. Through the analysis of these traces, an I/O replication workload can be automatically generated, which is able to replay the I/O behavior of the original application, and in turn is also able to predict the application's I/O performance. Replay-based models can be used for the evaluation of different real-world hardware deployments, but can also be used to generate the workload for storage system simulations. However, a replay-based application can only represent the I/O characteristics of one specific application.

Several research studies [15]–[17], [34], [36], [37] have used replay-based modeling to predict and model HPC workloads. For instance, Hao et al. [15] propose a framework that can automatically generate benchmarks for I/O-intensive MPI applications. The framework takes I/O traces of the original application as the input, performs a trace compressing algorithm based on a suffix tree to reduce the size of traces, and then generates the C code of the corresponding benchmark.

4) Workload Generation: As previously described, various methods are available for replicating and mimicking I/O workloads, including I/O trace replay tools, synthetic and application benchmarks, synthetic workload generators, proxy applications, and application I/O skeletons. Each method offers distinct trade-offs; no technique works best in all scenarios.

I/O workload modeling and generation are major drivers behind the continuous effort of the collection of performance measurements and statistics from different storage systems. At the same time, workload generation plays a crucial part in storage system simulation, which for example allows researchers to evaluate new storage system research using relevant workloads and to directly examine the impact of some workload on a specific storage system implementation. Three major sources of workload information can be distinguished:

- ***I/O Trace Workloads:*** I/O traces [20], [25], [35] provide a detailed report regarding each I/O operation issued by a traced application, including timing information and I/O parameters. These traces can be used to replicate the exact I/O pattern of the original application.
- ***Synthetic I/O Workloads:*** Synthetic I/O workloads [20], [21] are manually designed I/O behavior descriptions used to create desired I/O and access patterns on a storage system. An example is the CODES I/O language [59], which allows researchers to model real or artificial I/O workloads using domain-specific language constructs.
- ***I/O Characterization Workloads:*** I/O profiles [20], [22] provide high-level statistics and capture an accurate picture of application I/O behavior, including properties such as access patterns within files, rather than complete traces.

C. Simulation Types and Techniques

Parallel file and storage system simulation is a powerful tool to examine application I/O behavior without access to a large-scale testbed. For instance, when a new end-to-end I/O behavior monitoring and optimization tool needs to be evaluated, simulation can be especially useful for server-side components as these cannot be easily tested in production environments. Furthermore, in order to analyze or optimize parallel I/O, different workload generation techniques can be used as the input for large-scale storage system simulations. Different simulation types and techniques can be distinguished.

1) (Parallel) Discrete-Event Simulation: Discrete-event simulation is a stochastic mathematical modeling tool, which simulates the behavior and performance of a real-life system as a (discrete) sequence of events in time. Parallel discrete event simulation refers to the execution of a single discrete event simulation program on a parallel computer.

To model the workload for discrete-event simulators, I/O traces, I/O characterization profiles, and synthetic workloads can be used. Parallel discrete-event simulation has been used in several works [20], [21], [33], [37]. For instance, Snyder et al. [20] introduce the *IOWA framework*, which provides an I/O workload abstraction based on different I/O workload generators methods (e.g., Recorder, CODES and Darshan) and workload consumers (such as storage system simulation and I/O replay tool). The work provides a detailed discussion on the different workload generation methods and presents an innovative technique for synthesizing representative I/O workloads from Darshan logs. The CODES simulation framework, which is used in IOWA, is built atop ROSS [60], a high-performance parallel discrete-event simulation system.

2) Trace-based Simulation: Trace-based or trace-driven simulations rely on I/O traces for the workload generation. Traces provide a time ordered record of function and system calls for a given application on a real HPC system. Some of the key advantages of trace-based simulations are the easy validation (i.e., compare with measured results), the accurate workload (traces preserve correlation and interference effects), and less randomness (deterministic input reduces output randomness). However, the traces from one system may not be representative (i.e., may need another trace if workload characteristics change) and the simulations can be costly as they require a high level of detail.

Several research papers have introduced trace-based simulation methodologies [16], [17], [36]. For example, Sangaiah et al. [36] propose *SynchroTrace*, a scalable, flexible, and accurate trace-based multithreaded simulation methodology.

3) Application and Execution-Driven Simulation: The execution-driven simulation model is similar to trace-driven simulation except that the application under study and the simulation are interleaved, i.e., the workload produce and workload consume event streams are interleaved and the application is executed in the simulator. In comparison to trace-based simulations, execution-driven modeling avoids long traces and provides a simpler, more flexible way to study new workloads.

Recent work [51], [58], [61] has introduced the execution-driven model to I/O performance evaluation. For instance, Obaida et al. [51] present *PyPassT*, an analysis based modeling framework built on static program analysis and integrated simulation of target HPC architectures. The prediction framework takes application model and launches it on Performance Prediction Toolkit (PPT) with desired model parameters. During the program execution in the simulation framework, performance counters can be collected which may be used for profiling and further analysis.

V. Emerging HPC Workloads

HPC is no longer solely limited to traditional simulation and modeling workloads. A recent I/O behavior analysis of a year's worth of I/O activity at NESRC [53] has revealed that HPC storage systems may no longer be dominated by write I/O – challenging the long- and widely-held belief that HPC workloads are write-intensive. Emerging HPC workloads now also encompass advanced and big data analytics, machine learning, deep learning, and data-intensive workflows.

A. Advanced Data Analytics and Machine Learning

Emerging HPC workloads are driven by machine learning (ML) and other data analytical techniques that rely on frameworks, such as Apache Spark [62], analytics and ML packages, for example, Pytorch [63] and TensorFlow [64], and domain-specific libraries that traditionally have not been used in HPC. These workloads exhibit largely different kinds of I/O patterns than the traditional simulation based workloads [65], and as a result perform poorly on HPC systems which need to adapt to the changing scenario of HPC workloads [66].

Continuity and reliability of storage resources is extremely important for emerging workloads that are associated with observational and experimental facilities, for example, the Center for Electron Microscopy [67], and the Advanced Photon Source [68] generating large volumes of data. These facilities currently generate hundreds of megabytes of data per second but are projected to generate tens to hundreds of gigabytes of data per second in the future.

B. Distributed Deep Learning

Recently, deep neural networks (DNNs) have gained tremendous interest due to their potential for solving complex problems, like image recognition, natural languages processing, and autonomous vehicles. With the growth of computation power of processors and accelerators, such as on leadership-class HPC systems, larger datasets can be trained more efficiently with DNNs. However, there need to be efforts on improving I/O support for DNNs to match the increasing computation power. Deep learning (DL) frameworks, such as TensorFlow [64], MXNet [69], and LBANN [70], invoke file read requests to the HPC parallel file systems and form mini-batches of data required for successful training. This is in contrast to the traditional well-structured HPC I/O pattern (for example, checkpoint/restart, multi-dimensional I/O access), as the DL training phase gives rise to highly random small file accesses [71]. The requirement of randomly shuffled input imposes significant pressure to parallel file systems, which are typically designed and optimized for large sequential I/O.

C. Data-Intensive Scientific Workflows

The science drivers for HPC are broadening with the proliferation of high-resolution observational instruments and emergence of completely new data-intensive scientific domains as described above. Scientific workflows [72] that chain the processing and data are becoming critical to manage these on HPC systems. Thus, while providers of HPC resources must continue to support the extreme bandwidth requirements of traditional HPC workloads, centers must now also deploy resources that are capable of supporting the requirements of these emerging data-intensive workflows. In sharp contrast to the traditional highly coherent, sequential, large-transaction reads and writes, data-intensive workflows have been shown to often utilize non-sequential, metadata-intensive, and small-transaction reads and writes [73]. Therefore, the rapid growth in I/O demands coming from data-intensive workflows are demanding new performance and optimization requirements of future HPC I/O subsystems [74].

VI. Key Findings and Research Challenges

Emerging HPC workloads such as artificial intelligence, big data analytics and complex multi-step workflows alongside future exascale applications [75] are anticipated workloads, which will result in a more diverse system-wide workload and even less predictable I/O behavior and access patterns. Preliminary research [24], [43], [48], [56], [58] has shown that I/O evaluation techniques can be significantly improved through the adaption of new techniques such as machine learning algorithms for I/O pattern prediction of complex, hybrid application workloads. The key findings and research challenges can be summarized as follows:

- **New open-source benchmarks and in-depth I/O characterization are needed to evaluate emerging hybrid HPC systems workloads.** The majority of the examined research still relies on synthetic benchmarks such as IOR [76], NPB [77], and HACC-IO [78] or write-intensive, bursty workloads for the evaluation and validation of novel parallel I/O optimization techniques. Only a few case studies exist for emerging workloads [48], [58], [79]. Recent work by Devarajan et al. [80] provides a promising start with the proposal of *DLIO*, a data-centric benchmark for scientific deep learning applications.
- **I/O profiling and characterization tools still lack the ability to capture and analyze emerging HPC workloads.** Parallel file systems in today's supercomputers have been optimized for more traditional HPC workloads, while state-of-the-art performance analysis tools have mainly focused on traditional interfaces such as POSIX I/O, MPI-IO and HDF5. Current I/O tracers and profiling tools lack the means to understand fine-grained I/O performance. In future work, it is essential to develop methods to quantitatively characterize the I/O needs of emerging workloads such as data-intensive workflows to ensure that the design of future HPC I/O subsystems and storage paradigms can address the rapid growth in I/O demand, but also that resources can be deployed with the correct balance of performance characteristics. Chien et al. have recently introduced *tf-Darshan* [24], which enables profiling and tracing of I/O operations in ML workloads using the TensorFlow framework.
- **A better understanding of available simulation frameworks and workload generators is needed.** Most research relies on the simulation or emulation of large-scale storage systems, which makes it more difficult to generate a representative HPC workload behavior. While simulation frameworks such as CODES [59] provide a promising way for researchers to simulate emerging workloads, leveraging the full potential requires the manual design of the I/O behavior description to create the desired I/O and access patterns. This not only intensifies the need for a better understanding of the I/O behavior of emerging HPC workloads, but also the demand for new open-access workload generators or the access to updated I/O traces or characterization datasets that represent the changing landscape of emerging workloads. Finally, standardized workload generation approaches as proposed for example by Snyder et al. [20] will help in

synthesizing representative I/O workloads from I/O traces and characterization logs. However, such techniques are rarely adopted by recent research.

VII. Conclusion

This survey presents a snapshot of the current I/O behavior and performance analysis landscape and identifies areas that require future research. Parallel I/O remains an important topic in the HPC community, motivated by the ever-increasing gap between processing power and storage (i.e., I/O) capabilities and intensified by the emergence of complex hybrid HPC workloads that challenge the long- and widely-held belief that HPC workloads are primarily write-intensive.

More comprehensive and open source workload generation tools will be needed to drive the behavior analysis, modeling, and prediction of future extreme-scale parallel I/O. Furthermore, training events for both storage researchers and domain scientists will help broaden the understanding of available tools and complexities of the parallel I/O stack.

Acknowledgment

This research used resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at the Oak Ridge National Laboratory, which is supported by the Office of Science of the DOE under Contract DE-AC05-00OR22725.

References

[1] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller, *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep*, vol. 15, 2008.

[2] S. Neuwirth, *Accelerating Network Communication and I/O in Scientific High Performance Computing Environments*. PhD thesis, Heidelberg University, Germany, December 2018.

[3] F. Z. Boito, E. C. Inacio, J. L. Bez, P. O. A. Navaux, M. A. R. Dantas, and Y. Denneulin, "A Checkpoint of Research on Parallel I/O for High-Performance Computing," *ACM Comput. Surv.*, vol. 51, Mar. 2018.

[4] D. Chapp, K. Sato, D. Ahn, and M. Taufer, "Record-and-Replay Techniques for HPC Systems: A Survey," *Supercomputing Frontiers and Innovations*, vol. 5, no. 1, 2018.

[5] A. K. Gupta, B. Budroo, S. Shinde, P. Bakshi, P. Joshi, and R. Pol, "Survey of Open Source Tools for Monitoring I/O & Storage Performance of HPC Systems," *International Journal of Computer Techniques*, vol. 8, no. 1, 2021.

[6] Y. Charband and N. J. Navimipour, "Online knowledge sharing mechanisms: a systematic review of the state of the art literature and recommendations for future research," *Information Systems Frontiers*, vol. 18, no. 6, pp. 1131–1151, 2016.

[7] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – a systematic literature review," *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009. Special Section - Most Cited Articles in 2002 and Regular Research Papers.

[8] Virtual Institute for I/O, "Analysis Tools and I/O Benchmarks." https://www.vi4io.org/tools/, 2021. [accessed 9-July-2021].

[9] Rajeev Thakur, "Parallel I/O Benchmarks, Applications, Traces." https://www.mcs.anl.gov/~thakur/pio-benchmarks.html, 2021. [accessed 9-July-2021].

[10] O. B. Messer, E. D'Azevedo, J. Hill, W. Joubert, M. Berrill, and C. Zimmer, "MiniApps derived from production HPC applications using multiple programming models," *The International Journal of High Performance Computing Applications*, vol. 32, no. 4, pp. 582–593, 2018.

[11] S. Herbein, S. McDaniel, N. Podhorszki, J. Logan, S. Klasky, and M. Taufer, "Performance characterization of irregular I/O at the extreme scale," *Parallel Computing*, vol. 51, pp. 17–36, 2016. Special Issue on Parallel Programming Models and Systems Software for High-End Computing.

[12] J. Dickson, S. Wright, S. Maheswaran, A. Herdman, M. C. Miller, and S. Jarvis, "Replicating HPC I/O Workloads with Proxy Applications," in *2016 1st Joint International Workshop on Parallel Data Storage and data Intensive Scalable Computing Systems (PDSW-DISCS)*, 2016.

[13] J. Dickson, S. A. Wright, M. C. Miller, and S. A. Jarvis, "Enabling Portable I/O Analysis of Commercially Sensitive HPC Applications Through Workload Replication," in *2017 Cray User Group Conference (CUG)*, 2017.

[14] J. Logan, J. Y. Choi, M. Wolf, G. Ostrouchov, L. Wan, N. Podhorszki, W. Godoy, S. Klasky, E. Lohrmann, G. Eisenhauer, C. Wood, and K. Huck, "Extending Skel to Support the Development and Optimization of Next Generation I/O Systems," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 563–571, 2017.

[15] M. Hao, W. Zhang, Y. Zhang, M. Snir, and L. T. Yang, "Automatic generation of benchmarks for I/O-intensive parallel applications," *Journal of Parallel and Distributed Computing*, vol. 124, pp. 1–13, 2019.

[16] X. Luo, F. Mueller, P. Carns, J. Jenkins, R. Latham, R. Ross, and S. Snyder, "HPC I/O Trace Extrapolation," in *Proceedings of the 4th Workshop on Extreme Scale Programming Tools*, ESPT '15, ACM, 2015.

[17] X. Luo, F. Mueller, P. Carns, J. Jenkins, R. Latham, R. Ross, and S. Snyder, "ScalaIOExtrap: Elastic I/O Tracing and Extrapolation," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 585–594, 2017.

[18] A. Haghdoost, W. He, J. Fredin, and D. H. Du, "On the Accuracy and Scalability of Intensive I/O Workload Replay," in *15th USENIX Conference on File and Storage Technologies (FAST 17)*, (Santa Clara, CA), pp. 315–328, USENIX Association, Feb. 2017.

[19] A. Haghdoost, W. He, J. Fredin, and D. H. C. Du, "Hfplayer: Scalable Replay for Intensive Block I/O Workloads," *ACM Trans. Storage*, vol. 13, Dec. 2017.

[20] S. Snyder, P. Carns, R. Latham, M. Mubarak, R. Ross, C. Carothers, B. Behzad, H. V. T. Luu, S. Byna, and Prabhat, "Techniques for Modeling Large-Scale HPC I/O Workloads," in *Proceedings of the 6th International Workshop on Performance Modeling, Benchmarking, and Simulation of High Performance Computing Systems*, PMBS '15, 2015.

[21] C. D. Carothers, J. S. Meredith, M. P. Blanco, J. S. Vetter, M. Mubarak, J. LaPre, and S. Moore, "Durango: Scalable Synthetic Workload Generation for Extreme-Scale Application Performance Modeling and Simulation," in *Proceedings of the 2017 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '17, ACM, 2017.

[22] P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, and K. Riley, "24/7 Characterization of petascale I/O workloads," in *2009 IEEE International Conference on Cluster Computing and Workshops*, 2009.

[23] C. Xu, S. Snyder, V. Venkatesan, P. Carns, O. Kulkarni, S. Byna, R. Sisneros, and K. Chadalavada, "DXT: Darshan eXtended Tracing," in *2017 Cray User Group Conference (CUG)*, 2017.

[24] S. W. D. Chien, A. Podobas, I. B. Peng, and S. Markidis, "tf-Darshan: Understanding Fine-grained I/O Performance in Machine Learning Workloads," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 359–370, 2020.

[25] H. Luu, B. Behzad, R. Aydt, and M. Winslett, "A multi-level approach for understanding I/O activity in HPC applications," in *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, 2013.

[26] C. Wang, J. Sun, M. Snir, K. Mohror, and E. Gonsiorowski, "Recorder 2.0: Efficient Parallel I/O Tracing and Analysis," in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 1–8, 2020.

[27] A. K. Paul, S. Tuecke, R. Chard, A. R. Butt, K. Chard, and I. Foster, "Toward scalable monitoring on large-scale storage for software defined cyberinfrastructure," in *Proceedings of the 2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems*, pp. 49–54, 2017.

[28] A. K. Paul, R. Chard, K. Chard, S. Tuecke, A. R. Butt, and I. Foster, "Fsmonitor: Scalable file system monitoring for arbitrary storage systems," in *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 1–11, IEEE, 2019.

[29] A. K. Paul, A. Goyal, F. Wang, S. Oral, A. R. Butt, M. J. Brim, and S. B. Srinivasa, "I/o load balancing for big data hpc applications," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017.

[30] H. Luu, M. Winslett, W. Gropp, R. Ross, P. Carns, K. Harms, M. 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, NY, USA), p. 33–44, ACM, 2015.

[31] S. Snyder, P. Carns, K. Harms, R. Ross, G. K. Lockwood, and N. J. Wright, "Modular HPC I/O Characterization with Darshan," in *2016 5th Workshop on Extreme-Scale Programming Tools (ESPT)*, 2016.

[32] G. P. Rodrigo, P.-O. Östberg, E. Elmroth, K. Antypas, R. Gerber, and L. Ramakrishnan, "Towards understanding HPC users and systems: A NERSC case study," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 206–221, 2017.

[33] H. Khetawat, C. Zimmer, F. Mueller, S. Atchley, S. S. Vazhkudai, and M. Mubarak, "Evaluating Burst Buffer Placement in HPC Systems," in *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 1–11, 2019.

[34] A. Saif, L. Nussbaum, and Y.-Q. Song, "IOscope: A Flexible I/O Tracer for Workloads' I/O Pattern Characterization," in *High Performance Computing. ISC High Performance 2018 Workshops. Lecture Notes in Computer Science*, vol. 11203, pp. 103–116, Springer International Publishing, 2018.

[35] W. He, D. H. Du, and S. B. Narasimhamurthy, "PIONEER: A Solution to Parallel I/O Workload Characterization and Generation," in *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 111–120, 2015.

[36] K. Sangaiah, M. Lui, R. Jagtap, S. Diestelhorst, S. Nilakantan, A. More, B. Taskin, and M. Hempstead, "SynchroTrace: Synchronization-Aware Architecture-Agnostic Traces for Lightweight Multicore Simulation of CMP and HPC Workloads," *ACM Trans. Archit. Code Optim.*, vol. 15, Mar. 2018.

[37] F. Azevedo, D. Klusáček, and F. Suter, "Improving Fairness in a Large Scale HTC System Through Workload Analysis and Simulation," in *Euro-Par 2019: Parallel Processing* (R. Yahyapour, ed.), pp. 129–141, Springer International Publishing, 2019.

[38] J. M. Kunkel, E. Betke, M. Bryson, P. Carns, R. Francis, W. Frings, R. Laifer, and S. Mendez, "Tools for Analyzing Parallel I/O," in *High Performance Computing* (R. Yokota, M. Weiland, J. Shalf, and S. Alam, eds.), (Cham), pp. 49–70, Springer International Publishing, 2018.

[39] S. S. Vazhkudai, R. Miller, D. Tiwari, C. Zimmer, F. Wang, S. Oral, R. Gunasekaran, and D. Steinert, "GUIDE: A Scalable Information Directory Service to Collect, Federate, and Analyze Logs for Operational Insights into a Leadership HPC Facility," SC '17, (New York, NY, USA), Association for Computing Machinery, 2017.

[40] O. Yildiz, M. Dorier, S. Ibrahim, R. Ross, and G. Antoniu, "On the Root Causes of Cross-Application I/O Interference in HPC Storage Systems," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 750–759, 2016.

[41] S. Di, R. Gupta, M. Snir, E. Pershey, and F. Cappello, "LOGAIDER: A Tool for Mining Potential Correlations of HPC Log Events," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 442–451, 2017.

[42] G. K. Lockwood, N. J. Wright, S. Snyder, P. Carns, G. Brown, and K. Harms, "TOKIO on ClusterStor: Connecting Standard Tools to Enable Holistic I/O Performance Analysis," *2018 Cray User Group Conference (CUG)*, 1 2018.

[43] B. H. Park, S. Hukerikar, R. Adamson, and C. Engelmann, "Big Data Meets HPC Log Analytics: Scalable Approach to Understanding Systems at Extreme Scale," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 758–765, 2017.

[44] G. K. Lockwood, W. Yoo, S. Byna, N. J. Wright, S. Snyder, K. Harms, Z. Nault, and P. Carns, "UMAMI: A Recipe for Generating Meaningful Metrics through Holistic I/O Performance Analysis," in *Proceedings of the 2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems*, PDSW-DISCS '17, (New York, NY, USA), p. 55–60, Association for Computing Machinery, 2017.

[45] B. Yang, X. Ji, X. Ma, X. Wang, T. Zhang, X. Zhu, N. El-Sayed, H. Lan, Y. Yang, J. Zhai, W. Liu, and W. Xue, "End-to-end I/O Monitoring on a Leading Supercomputer," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, (Boston, MA), pp. 379–394, USENIX Association, Feb. 2019.

[46] B. Wadhwa, A. K. Paul, S. Neuwirth, F. Wang, S. Oral, A. R. Butt, J. Bernard, and K. W. Cameron, "iez: Resource Contention Aware Load Balancing for Large-Scale Parallel File Systems," in *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 610–620, 2019.

[47] 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*, pp. 931–943, 2018.

[48] J. Lüttgau, S. Snyder, P. Carns, J. M. Wozniak, J. Kunkel, and T. Ludwig, "Toward Understanding I/O Behavior in HPC Workflows," in *2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS)*, 2018.

[49] T. Wang, S. Snyder, G. Lockwood, P. Carns, N. Wright, and S. Byna, "IOMiner: Large-Scale Analytics Framework for Gaining Knowledge from I/O Logs," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 466–476, 2018.

[50] B. Xie, Y. Huang, J. S. Chase, J. Y. Choi, S. Klasky, J. Lofstead, and S. Oral, "Predicting Output Performance of a Petascale Supercomputer," HPDC '17, p. 181–192, ACM, 2017.

[51] M. A. Obaida, J. Liu, G. Chennupati, N. Santhi, and S. Eidenbenz, "Parallel Application Performance Prediction Using Analysis Based Models and HPC Simulations," in *Proceedings of the 2018 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, SIGSIM-PADS '18, p. 49–59, acm, 2018.

[52] R. Gunasekaran, S. Oral, J. Hill, R. Miller, F. Wang, and D. Leverman, "Comparative i/o workload characterization of two leadership class storage clusters," in *Proceedings of the 10th Parallel Data Storage Workshop*, PDSW '15, ACM, 2015.

[53] T. Patel, S. Byna, G. K. Lockwood, and D. Tiwari, "Revisiting I/O Behavior in Large-Scale Storage Systems: The Expected and the Unexpected," SC '19, ACM, 2019.

[54] A. K. Paul, O. Faaland, A. Moody, E. Gonsiorowski, K. Mohror, and A. R. Butt, "Understanding HPC Application I/O Behavior Using System Level Statistics," in *2020 IEEE 27th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, 2020.

[55] M. Dorier, S. Ibrahim, G. Antoniu, and R. Ross, "Using Formal Grammars to Predict I/O Behaviors in HPC: The Omnisc'IO Approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 8, pp. 2435–2449, 2016.

[56] J. Schmid and J. Kunkel, "Predicting I/O Performance in HPC Using Artificial Neural Networks," *Supercomputing Frontiers and Innovations*, vol. 3, no. 3, 2016.

[57] J. Sun, G. Sun, S. Zhan, J. Zhang, and Y. Chen, "Automated Performance Modeling of HPC Applications Using Machine Learning," *IEEE Transactions on Computers*, vol. 69, no. 5, pp. 749–763, 2020.

[58] F. Chowdhury, Y. Zhu, F. Di Natale, A. Moody, E. Gonsiorowski, K. Mohror, and W. Yu, "Emulating I/O Behavior in Scientific Workflows on High Performance Computing Systems," in *2020 IEEE/ACM Fifth International Parallel Data Systems Workshop (PDSW)*, 2020.

[59] N. Liu, J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn, "On the role of burst buffers in leadership-class storage systems," in *2012 IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–11, 2012.

[60] C. D. Carothers, D. Bauer, and S. Pearce, "ROSS: A high-performance, low-memory, modular Time Warp system," *Journal of Parallel and Distributed Computing*, vol. 62, no. 11, pp. 1648–1669, 2002.

[61] W. Liu, K. Wu, J. Liu, F. Chen, and D. Li, "Performance Evaluation and Modeling of HPC I/O on Non-Volatile Memory," in *2017 International Conference on Networking, Architecture, and Storage (NAS)*, 2017.

[62] Apache Software Foundation, "Apache Spark." https://spark.apache.org/, 2021. [accessed 12-July-2021].

[63] Pytorch. https://pytorch.org/, 2021. [accessed 12-July-2021].

[64] Tensorflow. tensorflow.org, 2021. [accessed 12-July-2021].

[65] P. Xenopoulos, J. Daniel, M. Matheson, and S. Sukumar, "Big Data Analytics on HPC Architectures: Performance and Cost," in *2016 IEEE International Conference on Big Data (Big Data)*, pp. 2286–2295, 2016.

[66] P. Xuan, W. B. Ligon, P. K. Srimani, R. Ge, and F. Luo, "Accelerating big data analytics on HPC clusters using two-level storage," *Parallel Computing*, vol. 61, pp. 18–34, 2017. Special Issue on 2015 Workshop on Data Intensive Scalable Computing Systems (DISCS-2015).

[67] National Center for Electron Microscopy. https://foundry.lbl.gov/instrumentation/ncem-computer-lab/, 2021. [accessed 12-July-2021].

[68] Advanced Photon Source. aps.anl.gov, 2021. [accessed 12-July-2021].

[69] T. Chen, Q. Kou, and T. He, "mxnet: MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems," *R package version*, vol. 1, no. 0, 2017.

[70] B. Van Essen, H. Kim, R. Pearce, K. Boakye, and B. Chen, "LBANN: Livermore Big Artificial Neural Network HPC Toolkit," in *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, MLHPC '15, (New York, NY, USA), ACM, 2015.

[71] F. Chowdhury, Y. Zhu, T. Heer, S. Paredes, A. Moody, R. Goldstone, K. Mohror, and W. Yu, "I/O Characterization and Performance Evaluation of BeeGFS for Deep Learning," in *Proceedings of the 48th International Conference on Parallel Processing (ICPP 2019)*, 2019.

[72] C. Daley, D. Ghoshal, G. Lockwood, S. Dosanjh, L. Ramakrishnan, and N. Wright, "Performance characterization of scientific workflows for the optimal use of Burst Buffers," *Future Generation Computer Systems*, vol. 110, pp. 468–480, 2020.

[73] R. Ferreira da Silva, R. Filgueira, I. Pietri, M. Jiang, R. Sakellariou, and E. Deelman, "A characterization of workflow management systems for extreme-scale applications," *Future Generation Computer Systems*, vol. 75, pp. 228–238, 2017.

[74] R. F. da Silva, H. Casanova, K. Chard, T. Coleman, D. Laney, D. Ahn, S. Jha, D. Howell, S. Soiland-Reyes, I. Altintas, D. Thain, R. Filgueira, Y. N. Babuji, R. M. Badia, B. Balis, S. Caíno-Lores, S. Callaghan, F. Coppens, M. R. Crusoe, K. De, F. D. Natale, T. M. A. Do, B. Enders, T. Fahringer, A. Fouilloux, G. Fursin, A. Gaignard, A. Ganose, D. Garijo, S. Gesing, C. A. Goble, A. Hasan, S. Huber, D. S. Katz, U. Leser, D. Lowe, B. Ludäscher, K. Maheshwari, M. Malawski, R. Mayani, K. Mehta, A. Merzky, T. S. Munson, J. Ozik, L. Pottier, S. Ristov, M. Roozmeh, R. Souza, F. Suter, B. Tovar, M. Turilli, K. Vahi, A. Vidal-Torreira, W. R. Whitcup, M. Wilde, A. Williams, M. Wolf, and J. M. Wozniak, "Workflows Community Summit: Advancing the State-of-the-art of Scientific Workflows Management Systems Research and Development," *CoRR*, vol. abs/2106.05177, 2021.

[75] A. Geist and D. A. Reed, "A survey of high-performance computing scaling challenges," *The International Journal of High Performance Computing Applications*, vol. 31, no. 1, pp. 104–113, 2017.

[76] LANL, "IOR Benchmark." https://github.com/hpc/ior. [accessed 9-July-2021].

[77] NASA Advanced Supercomputing Devision, "NAS Parallel Benchmarks." https://www.nas.nasa.gov/software/npb.html. [accessed 9-July-2021].

[78] S. Habib, V. Morozov, N. Frontiere, H. Finkel, A. Pope, and K. Heitmann, "HACC: Extreme scaling and performance across diverse architectures," in *Proc. SC '13*, IEEE/ACM, 2013.

[79] M. Bae, M. Jeong, S. Yeo, S. Oh, and O.-K. Kwon, "I/O Performance Evaluation of Large-Scale Deep Learning on an HPC System," in *2019 International Conference on High Performance Computing Simulation (HPCS)*, pp. 436–439, 2019.

[80] H. Devarajan, H. Zheng, A. Kougkas, X.-H. Sun, and V. Vishwanath, "DLIO: A Data-Centric Benchmark for Scientific Deep Learning Applications," in *2021 21st IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 81–91, 2021.