Report from Dagstuhl Seminar 21332

Understanding I/O Behavior in Scientific and Data-Intensive Computing

Edited by

Philip Carns[1], Julian Kunkel[2], Kathryn Mohror[3], and Martin Schulz[4]

1 **Argonne National Laboratory, USA**
2 **Universität Göttingen / GWDG, DE**
3 **Lawrence Livermore National Laboratory, USA**
4 **TU München, DE**

──── **Abstract** ────

Two key changes are driving an immediate need for deeper understanding of I/O workloads in high-performance computing (HPC): applications are evolving beyond the traditional bulk-synchronous models to include integrated multistep workflows, in situ analysis, artificial intelligence, and data analytics methods; and storage systems designs are evolving beyond a two-tiered file system and archive model to complex hierarchies containing temporary, fast tiers of storage close to compute resources with markedly different performance properties. Both of these changes represent a significant departure from the decades-long status quo and require investigation from storage researchers and practitioners to understand their impacts on overall I/O performance. Without an in-depth understanding of I/O workload behavior, storage system designers, I/O middleware developers, facility operators, and application developers will not know how best to design or utilize the additional tiers for optimal performance of a given I/O workload. The goal of this Dagstuhl Seminar was to bring together experts in I/O performance analysis and storage system architecture to collectively evaluate how our community is capturing and analyzing I/O workloads on HPC systems, identify any gaps in our methodologies, and determine how to develop a better in-depth understanding of their impact on HPC systems. Our discussions were lively and resulted in identifying critical needs for research in the area of understanding I/O behavior. We document those discussions in this report.

1   Executive Summary

Philip Carns (Argonne National Laboratory, USA, carns@mcs.anl.gov)
Julian M. Kunkel (Universität Göttingen / GWDG, DE, julian.kunkel@gwdg.de)
Kathryn Mohror (Lawrence Livermore National Laboratory, USA, kathryn@llnl.gov)
Martin Schulz (TU München, DE, schulzm@in.tum.de)

Dagstuhl Seminar 21332, "Understanding I/O behavior in scientific and data-intensive computing," brought together computer scientists from around the world to survey how I/O workloads are measured and analyzed on high-performance computing (HPC) systems, identify gaps in methodologies, and debate how to best apply this technology to advance HPC productivity. The hybrid, week-long event attracted 10 physical and 25 virtual attendees. They included representatives from seven countries spanning a variety of career levels in academia, industry, and government. The diversity of perspectives, combined with an intense week-long seminar format, offered an unprecedented opportunity for researchers to share ideas and spark new collaborative opportunities.

The seminar agenda was structured as a combination of full-group plenary sessions and subgroup breakout sessions. The plenary sessions were used to discuss high-level issues, vote on subtopics to investigate, relay results from breakout sessions, and present "lightning" talks that highlighted key issues in the community. The breakout sessions employed small groups (roughly five people each) to follow up in "deep dive" discussions on specific subtopics. This format enabled attendees from numerous time zones to remain productively engaged throughout the week. We also found it to be successful in facilitating discussion despite the COVID-19 safety considerations that prevented us from assembling at a single venue. The final day of the seminar was devoted to recording seminar findings in a timely manner while subject matter experts were still available for consultation.

Over the course of the seminar, the attendees converged on six high-level topics for deep dive discussions that are covered in this report.

- **Tools: Cross-Cutting Issues** (Section 4.1) explored common challenges in development of tools for understanding HPC I/O.
- **Data Sources and Acquisition** (Section 4.2) addressed how to acquire various forms of raw I/O instrumentation from production systems.
- **Analysis** (Section 4.3) focused on how to interpret I/O instrumentation once acquired.
- **Enacting Actionable Responses** (Section 4.4) investigated how to best utilize the outcomes from I/O analysis.
- **Data Center Support** (Section 4.5) focused on strategies for facility operators to facilitate better understanding of I/O behavior.
- **Community Support** (Section 4.6) explored the unique characteristics of the I/O analysis community and how to foster its growth.

This report presents a separate summary for each deep dive topic, including a survey of the state of the art, gaps, challenges, and recommendations. The report concludes in Section 5 with a summary of cross-cutting themes and recommendations produced by the seminar as a whole. We found that understanding I/O behavior in scientific and data-intensive computing is increasingly important in an era of evolving workloads and increasingly complex HPC systems and that several cross-cutting challenges must be addressed in order to maximize its potential.

2   Table of Contents

3 Brief Summaries of All Breakout Reports

Dagstuhl Seminar *Understanding I/O Behavior in Scientific and Data-Intensive Computing* explored a variety of subtopics in small breakout sessions over the first two days in order to identify key issues for broader discussion. This section briefly summarizes the findings of each of these early breakout sessions.

3.1 Tuesday Reports

The Tuesday breakout groups were organized around topics. Each group brainstormed ideas, raised questions, attempted to answer existing questions, and identified one most important question for the specific topic. Then, after 30 minutes, each group moved to the next topic. A document on each topic from each group was kept, providing the collective knowledge of all the attendees. The last group working on each topic summarized it and presented it in the plenary meeting.

3.1.1 I/O Workflow Analysis

The most important question identified was how to set up a proper abstraction level for workflow analysis. This covered the definition of workflows, the means to describe workflows and their characteristics, the analysis of workflows, and the optimization of workflows. The groups found that different workflow systems may define workflows differently. Typically, characteristics cover jobs, job steps, or tasks, organized hierarchically or as a directed acyclic graph with data or task dependencies, which may span high-performance computing, edge computing, and the cloud, and at multiple sites. The participants felt that a unanimous definition of workflow was not needed for I/O analysis and optimization.

A key issue raised was that a community standard for workflow specification does not yet exist. A portable and abstract representation would be useful for the community. Users should specify their workflows using such a description, and systems should be able to infer (learn) such descriptions using monitoring systems.

In the analysis stage, the perspective is generally system-centric and application-centric. A holistic view covering individual applications with their workloads but also the emerging workflows across sites was identified as important for tools. Having workflow knowledge, systems could utilize node local storage and burst buffers and improve job scheduling. However, the community is not yet able to exploit workflow specifications because of the lack of descriptions and monitoring systems that cover workflows.

3.1.2 Tools for I/O Analysis

The discussion started by analyzing the state-of-the-art tools for I/O performance monitoring and identifying the gaps, limitations, and unanswered questions for the I/O performance monitoring and analysis tools. There was consensus that many different I/O monitoring tools are available (at application, file system, and storage system levels). Obtaining an integrated view of the I/O performance is challenging, however, because the results are from diverse performance-monitoring technologies and often target different users (e.g., system administrators vs. applications developers). Moreover, the user typically has no access to monitoring or not enough knowledge to interpret the profiling results.

With regard to next-generation tools, the consensus was that I/O tools are needed for monitoring and analyzing applications using I/O on beyond-POSIX file systems and for different consistency models of I/O operations. In addition, tools are needed for supporting new storage heterogeneous platforms and emerging HPC applications, not necessarily using MPI. A big challenge ahead for I/O monitoring tools is the large amount of data collected. The tools will likely select only certain parts of the codes to be profiled or traced, disregarding data collection.

An open and challenging question is how tools can help understand the application and system behavior causing contention and performance degradation (during the workshop, this was called "I/O Weather" of the shared I/O infrastructure).

A second open question is how to help users and application developers understand application I/O performance in a context that they can utilize to improve performance. One possible option identified was the possibility for users having a report about their job's I/O performance with hints of varying difficulty levels for improving performance.

Further, standardization efforts for I/O tools were seen as important. Such actions might include common data format, core functionality, and open standards for usability and acceptance by the user community.

3.1.3   Changing Workloads and Their Requirements

Workloads can be defined as the I/O access patterns that hit the system partially or as a whole. This definition can be defined abstractly, and workloads can be also collected as traces. Nevertheless, characterizing workloads is difficult even for a specific application workflow, since the spatial and temporal granularity of the workflow can significantly change over its lifetime. Investigating the aggregate over HPC systems becomes even more complex, since many scientific applications typically run concurrently on them in an uncoordinated way.

Workload analysis (and I/O system optimizations) could be supported by information from applications and users about the intended use of I/O. Nevertheless, currently no formalism is available on exchanging this information; and many applications are also not documented well enough to derive it. In many cases, the information is restricted to whether an application is performing checkpoints or not.

Nevertheless, the HPC and data center communities expect that workloads are changing. One reason is the introduction of new storage interfaces, such as the S3 object storage interface. The impact of such emerging interfaces has not been thoroughly investigated yet. Also unclear is how hybrid systems consisting of POSIX-based parallel file systems and object stores will interact and whether existing applications will produce significant new access patters when using new interfaces. Another reason to expect new workloads is the widespread adoption of deep learning, which might lead to an increase in random accesses.

An open question is whether workload characteristics are really actually changing and on what systems and what the corresponding evidence is. If workloads are changing, then it is not obvious whether such changes are equally true for Tier-1, Tier-2, and Tier-3 systems. One therefore must collect and store a broad set of workload traces for different HPC sites to compare short- and long-term trends on HPC I/O usage.

3.1.4   Data Center Support

After the brainstorming, numerous questions were raised involving data center support to understand I/O behavior in scientific and data-intensive computing. We grouped the questions into five themes: (1) data center requirements to fulfill its role; (2) data center

functionality for I/O optimization; (3) user applications compared with those of system administrators; (4) community and standardization for data centers; and (5) open challenges such as POSIX. This breakout helped inspire further discussion about the data center and community for the I/O.

3.1.5   Storage System Design

Important questions about new storage systems are how their design can help capture application I/O behavior and how it can help predict behavior. Both features are needed in order to support application performance portability. In general, these issues should be addressed in a co-design scheme where we work with parallel file system developers to specify application I/O behavior requirements, share application I/O patterns and workload descriptions, and discuss predictive models for the next generation of storage systems. While we have been able to specify the information we need from a new storage system (instrumentation data, system view and description) and what we can provide from I/O behavior analysis (high-level access patterns, mapping of application I/O activity to file systems and components of the storage system), a number of questions have been left open. For example, how much information about file system internals is needed by the user, and can an artificial intelligence model learn I/O behavior sufficiently well?

3.2   Wednesday Reports

The Wednesday workgroups were organized to cover the topics from Day 1 plus hot topics suggested by attendees after the discussion of Day 1 topics. Then a voting took place, and people were grouped according to their interest. Topics were covered in two longer sessions with varying groups and attendees asked to brainstorm issues, identify challenges, and collect solutions. The resulting reports were presented to the whole group for discussion, which then led to the following six core topics forming this report.

3.2.1   I/O Workflow Analysis

The analysis of I/O in workflows is a complex task. The workflows are often developed organically as the domain's needs grows. This development has led to a proliferation of workflow engines such as Pegasus, Swift, Fireworks, Kepler, Sandia Analysis Workbench, and Dask and domain-specific workflow engines (Galaxy, Taverna, etc.). However, our profiling tools are often limited to the scope of a specific application and lack the temporal correlation between multiple applications participating in a workflow. To achieve a cohesive analysis of workflow, we need to extend the profiling capabilities beyond an application, capture temporal data dependencies between applications, and present a standardized format for representing the I/O behavior. Additionally required are analysis tools that can operate on the standardized I/O logs and extract key I/O behaviors. For analysis, one could potentially leverage existing routines presented in workflow engines (typically used for load balancing, task scheduling, etc.) and extend them with Python-based wrappers to expose the data frame directly to the user.

3.2.2   Tools for I/O Analysis

Many tools have been developed to acquire I/O instrumentation data on HPC systems, with each instrumenting applications, libraries, file systems, and other storage subsystem components. These tools typically come with corresponding analysis components geared

toward providing understanding of HPC I/O behavior, with presentation of this data tailored to application users, facilities staff, or I/O researchers. However, existing I/O analysis tools suffer from a number of shortcomings that impede their ability to provide meaningful insights into I/O behavior. One of the biggest shortcomings is that existing analysis tools are not well suited to aggregating data from multiple instrumentation sources into a holistic systemwide view, which is critical to understanding I/O behavior. Communication-related problems with tools also exist: these tools do not usually have mechanisms for communicating feedback (i.e., to users or libraries) and are often intended for certain types of I/O experts (researchers, system admins, etc.), making them incomprehensible to many users. Developers of I/O analysis tools should think more explicitly about interoperability with other tools in order to enable a more holistic I/O analysis tool ecosystem. Additionally, tool developers should rethink communication strategies to ensure they are providing users with meaningful feedback and are communicating with users in terms that are understandable and relevant to the analysis task at hand.

3.2.3   Standardization of Workflow Specification and Characterization

An I/O workflow usually starts from something simple (e.g., a sequential producer-consumer model) and then grows organically. Many workflow representations and engines were developed to meet the requirements of advanced features, such as fan-in/fan-out control and in situ analysis. However, the lack of a standardized workflow representation binds the analysis tools to one specific workflow engine. The lack of such representation also makes it difficult for users to switch between different workflow engines.

Designing an appropriate intermediate representation for an I/O workflow is not easy. In the same way that users compose a workflow, the standard should also start from the simplest case and add more features gradually. Existing workflow languages (from non-I/O fields, e.g., CWL) may serve as a good starting point. This representation should not be too generic and thus not put too much burden on the users. The ability to support hints such as I/O patterns between two interacting components is useful because it allows the implementation to perform various optimizations accordingly. Such hints can be platform dependent and thus should be made optional to allow portability. Both portability and reproducibility help encourage the standard adoption so we should keep them in mind when designing the workflow representation.

3.2.4   How to Better Engage Users in I/O Performance Tuning ("gamifying" I/O tuning)

Engaging users to tune I/O is a challenging task for several reasons, including low understanding of I/O performance, lack of analysis tools that are easy to use, and less importance given to I/O. Three main classes of users were identified in this breakout session: application developers/users, system administrators, and I/O library developers. Typical issues reported by these stakeholders are poor metadata or I/O performance in their application or library, variation of performance for the same I/O pattern, and increased workload. To improve participation of these users via "gamifying" I/O tuning, the workgroups discussed various incentive strategies. These potential strategies include granting rewards for fixing I/O problems, rewarding usage of efficient high-level libraries, comparing a user's I/O performance with that of other users with the same I/O pattern, and showing a history of performance with previous runs. Having policies that limit usage of resources when a problem is not fixed (e.g., writing file per process by large-scale applications that burdens the metadata servers)

was also discussed as a potential strategy. Improvements needed for facilities and the I/O community to increase user engagement in I/O tuning include easily understandable metrics to demonstrate benefit to the applications and I/O libraries, targeting of applications and workflows that provide the most benefit for the effort, continuous instrumentation, effective communication strategies, and I/O-tuning cookbooks.

3.2.5 Evaluation of Application Semantics and Matching Them to the Appropriate File System

The first question was whether tools could extract I/O semantics of applications matched for a specific file system. Some tools and automated approaches, including some benchmarking, are available. They should be used more, at least for the top 10 of the I/O-intensive applications in every center. Doing so would allow some pragmatic benefit.

Such tools will not be completely sufficient, however, because they observe only what an application is actually doing. They cannot capture what an application *should* do in terms of I/O, for example, when unnecessary information is written or when too frequent checkpoints are made. Thus, a structured dialog with either application users or developers cannot just be replaced. This can cover the observed I/O behavior as well as the intended one.

The groups also discussed the danger of assuming that the parallel file system would provide all the strict POSIX semantics. Such a provision could lead to slowdown for the application or the entire parallel file system. It might also lead to silent data corruption, which is even more critical. Therefore, the current situation remains unacceptable.

3.2.6 Sustaining the Tool Ecosystem and What Tools to Focus On

A variety of research tools exist for understanding I/O. While building a prototype for a new tool is easy, however, it is extremely hard to sustain the development and maintain the software such that it is useful for the community in the long run.

The groups identified the following methodology to discuss the topic: (1) investigate success stories (and failed attempts) for sharing of tools in order to identify common schemes; (2) discuss challenges; (3) identify and discuss approaches to mitigate issues; and (4) propose next steps.

The following tools and approaches were discussed: VI-HPS, IO500, IOR+MDTest, TAU, Darshan, MPI, SLURM, and Allinea tools – DDT/Map, Ellexus Breeze / Mistral.

Successful approaches have a combination of funding, luck, commitment to use software due to networking or initial collaboration projects, unfunded time commitment, research (and publication) opportunities, and community support. Here DOE/DOD centers provide better than do EU centers; presumably working for the "same boss" makes it easier to work on a shared roadmap.

Challenges that inhibit the usage of a tool include a lack of roadmaps, lack of documentation, lack of communication, and closed-source software (e.g., available only in an internal GitLab).

The proposed actions were as follows:
1. Document the history of various tools to learn from it (could be a publication).
2. Organize regular meetings covering operational and development aspects of tools.
3. Raise awareness of publication venues for software products – and publish there.
4. Try to get commitments from data centers for tool sustainment.
5. Join VI-HPS for analysis tools.
6. Establish joint steering for VI4IO, and add information and schedules to webpage.
7. Investigate whether software developers (instead of researchers) could be engaged in tool development.

3.2.7 I/O Performance Tuning – Actionable Strategies

I/O performance tuning can be seen from the viewpoint of different stakeholders: users (who have the goal of tuning their application and who can affect knobs only for their own job), administrators (who care about overall efficiency and stability and who can affect systemwide parameters), and developers (who design libraries and runtime systems for efficient I/O usage). Because of time limits we focused on the first role, the user.

The groups classified possible actions by their ease of execution or deployment, using the metaphor of picking fruit from a tree:

1. Picking fruit from the ground
   The first line of defense is datacenter-driven training for users, given by experts who often have simple rules of thumb that can help in many scenarios.
2. Picking low-hanging fruit
   Since the number of experts is often limited, some of their support can be automated. This can include automatic and machine-readable system configurations, summaries of current system state, and possibly interactive visualization.
3. Picking fruit from the middle of the tree
   The resulting data could be interpreted for users, providing basic feedback on how well their application used I/O, for example, with a red/yellow/green light indication. Such a report could be given in the job output file, sent via email or made available via a web form. Ideally, it could be completed with simple suggestions for improvement based on expert rules.
4. Picking high-hanging fruit
   The data could then be further used to automate basic guidance, for example, by providing mapping information or visualizing the application components and workflow on system data.
5. Picking fruit out of the sky
   As the final step, tuning could be fully automated, including system and state detection (from Step 2), output evaluation (from Step 3), workflow integration and basic guidance (from Step 4), and ultimately automatic mapping and tuning.

3.2.8 I/O Performance Analysis

Differentiating this topic from I/O performance tuning, variability, and tools is difficult. One approach is to look at I/O performance analysis from both the application and system levels and the available monitoring and analysis data depending on the access permission (e.g., end users, admins, experts, support). Currently, continuous monitoring and profiling provide I/O characterization, traces, server-side statistics and logs, scheduler logs, and job metadata. Various, and often datacenter-dependent, visualization tools such as dashboards and scripts help with the interpretation of the performance data, but they can be difficult to understand and respond to accordingly, especially for end users.

During the breakout session the following open research questions were identified. At the application level: Can I/O tools provide I/O efficiency metrics (e.g., for end users as a score)? Is there a way to describe boundaries of good/appropriate/poor performance? Can we detect the root cause of individual application performance degradation and slowdowns? At the system level: Can monitoring tools be preventive (detect an I/O system overload before system performance degradation)? Can jobs with problematic and performance-degrading I/O patterns be identified?

One idea of this breakout session was the creation of an application I/O utilization "score" that enables users and administrators to understand utilization characteristics at a glance in the context of system capability. This score could be based on a unification of the output of monitoring tools, which typically provide information to system administrators, and performance-tracing tools, which provide information for the users.

3.2.9   Scalable and actionable I/O log/trace analysis

Logs and traces of I/O operations are integral to analyzing and improving I/O behavior. One obstacle when processing logs and traces is the amount of data generated by multiple nodes, each with multiple processes doing I/O operations. Since this data collection is expensive in terms of both additional overhead during the program runtime and long-term storage, preprocessing or in situ data reduction and compression should be considered. The analysis of these traces should supply the user with a description of the I/O behavior. This can include efficiency metrics (such as percentage of available bandwidth used), usage over time, and some general classifications (e.g., possibly I/O bound). A further result of the analysis could be recommendations for the user: steps to take to improve the I/O pattern of the program. An evaluation of currently available tools showed that while many offer some form of this analysis, they usually need some form of expert knowledge. Ideally there should be a recommendation that is suitable for nonexpert users. In order to devise these recommendations. artificial intelligence methods may play a role, but that has to be evaluated on its own.

3.2.10   Open-Source Community-Supported Complete I/O Software Research and Development

Open source, community driven research software is important for supporting I/O understanding. Vendor-provided software often suffers due to lack of needed features and inability to verify methodologies for obtaining results due to being closed source. That said, there are many challenges in developing open source scientific software. Many times researchers are only funded to develop prototypes, and the tools that are made available are not robust or easily portable to new systems. It is very challenging for researchers to obtain funding to pay for development and maintenance, and the projects are maintained (if they are maintained at all) by small research groups instead of by the larger community.

We advocate for increased funding for maintenance and support of open-source tools and interfaces. The funding can support documentation, open data formats, customer support, porting of code to new platforms, and community input for development of new features. In particular for I/O understanding, we recommend:

- The establishment of user groups or virtual institutes, e.g., for I/O monitoring software;
- Better communication across the community about the goals of the software product, the state of the software, and how community members can contribute;
- Software availability on public platforms like GitHub;
- Reward or motivation for community members to volunteer their efforts to software products;
- Reproducibility testing for software, e.g., publishing test cases and expected results;
- Testing with standard benchmarks.

3.2.11 POSIX Replacement with Low-Level Specifications

A common notion in the HPC I/O community is that we should invent a more appropriate alternative to the POSIX I/O API and the POSIX semantics. The working groups concurred.

What is needed is a separate API that is simplistic, offering only a few well-defined typical I/O tasks. A novel API would clearly communicate this notion to developers. A transparent approach intercepting existing POSIX calls is bound to lead to confusion and disappointment.

Some of the simplifications that will lead to eventual advantages are the following:

- No directory notion but only a flat per-job data set
- No individual permissions checks but uniform permissions for the data set
- Forbiddance to read something written by the same job (because this should be communication)
- Introduction of a few mutual exclusive flavors for typical I/O activities per file or data object.

The reward for such a new API will be delayed, that is, until HPC applications can use it on top of production HPC file systems. The implementation should therefore start with a demonstration layer on top of existing parallel file systems, SQLite, S3, or PMEM. Then, the optimization potential for file system implementers can be demonstrated. Only after they adopt it will HPC applications be able to profit from the new approach.

3.2.12 I/O Performance Variability

The variability of I/O performance is the main reason for different execution times of similar user jobs. Many possible factors can cause this variability—not only that storage is a shared medium—and figuring out the real cause can be difficult. For example, the cause can be hardware issues (RAID rebuild after disk failure, failed redundant components such as servers or RAID controllers, reduced speed on network links), changed software versions (application, libraries, file systems), or bad configurations or bad I/O implementations. Improving the I/O, for example by using optimized striping options or by avoiding conflicting I/O (do not write to the same area from many clients, create files in separate directories), lessens this variability in many cases. Another option is to use dedicated resources, namely, a private parallel file system that uses node-local storage (e.g., BeeOND, GekkoFS).

3.2.13 Artificial Intelligence for Storage and I/O Systems

Work has been done successfully in artificial intelligence (AI) for storage, notably by vendors and researchers for predicting drive failures and characterizing/categorizing HPC applications in terms of I/O read and write characteristics. It is currently unclear, however, where AI is better suited than simpler statistical techniques for performing such characterizations. The groups recommended analyzing use cases and available data to determine applicability and possible performance gains if AI is utilized in this area for providing both understanding and automated feedback to applications and system components.

4 Deep Dive Topics

After two days of breakout sessions, which were described above, the group identified a series of topics to be investigated in more depth by individual groups. The following section describe the results of these group discussions (with a complete list of group members).
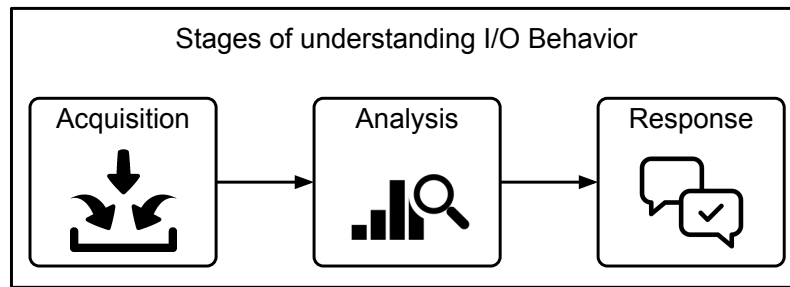
◾ **Figure 1** The process of understanding HPC I/O behavior can be decomposed into three core steps as defined by Ahlgren et al. [1]. The *acquisition* step includes components that produce instrumentation data and instrumentation data acquisition methods. The *analysis* step refers to visualization and other methods of deriving interpretations from data. The *response* step encompasses the enacting of policy or tuning changes.

4.1 Deep Dive Topic: Tools: Cross-Cutting Issues

Fahim Chowdhury (Florida State University, USA, fchowdhu@cs.fsu.edu),
Hariharan Devarajan (Lawrence Livermore National Laboratory, USA, hariharandev1@llnl.gov),
Ann Gentile (Sandia National Laboratories, USA, gentile@sandia.gov),
Jay Lofstead (Sandia National Laboratories, USA, gflofst@sandia.gov),
Kathryn Mohror (Lawrence Livermore National Laboratory, USA, mohror1@llnl.gov),
Devesh Tiwari (Northeastern University, USA, d.tiwari@northeastern.edu),
Chen Wang (University of Illinois at Urbana-Champaign, USA, chenw5@illinois.edu)

The heterogeneity and hierarchical nature of storage resources in HPC systems dictate the need for a holistic understanding of an application's I/O behavior in large-scale HPC systems. A compute-centric capturing of application behavior is not sufficient to capture the I/O behavior in these applications. First, compute resources are often isolated to a single node, with the scheduler providing exclusive access to these resources. In contrast, storage resources are often shared (e.g., shared burst buffers, I/O forwarders, and even global parallel file systems). This isolation increases the complexity of capturing accurate I/O behavior due to interference and colocation variability. Second, most middleware libraries and users have access to the compute cluster where the application runs but no access to storage resources. For instance, monitoring a Datawarp burst buffer at a system level or accessing Lustre logs to collect observed I/O behavior is extremely challenging. Third, compute-centric monitoring and tracing look at capturing individual elements of interest. However, the application's I/O behavior is often aggregated across the whole system (e.g., complex multijob workflows, producer-consumer paradigms). These factors dictate the need for a holistic set of tools spanning across software and hardware stack layers and providing a complete picture of the I/O behavior in modern HPC systems.

To achieve this holistic view of the I/O behavior across the whole HPC system, one can decompose this problem into three core steps as defined by Ahlgren et al. [1]. The *acquisition* step represents acquiring information from a collection of inputs such as instruments, monitors, profilers, and tracing libraries. This step also handles the actual data represented as instrumentation data, activity log data, application traces, and so on. The *analysis* step refers to extracting meaningful information from all the data collected by using visualization plots, interpretations, I/O timeline, and so on. This step converts the unprocessed data

collected from various sources into analyzed I/O behavior for different jobs in the HPC system. The *response* step provides actionable items to improve the efficiency of storage systems to accelerate or support complex applications by enacting policies and procedures for tuning the I/O subsystem.

4.1.1   State of the Art

In the past decade scientists have built several tools for understanding the I/O behavior of applications in the three stages mentioned above.

4.1.1.1   Tools for *Acquisition*

The tools in the *acquisition* step aim to capture information about the application and system depending on the target layer of the software stack. These tools can be categorized as profiling, tracing, and monitoring tools. **Profiling** tools, including Darshan [13], Vampir [2], and TAU [14], record I/O operations for different interfaces such as STDIO, POSIX, and MPI-IO, or even higher-level I/O libraries such as HDF5 [4], pNetCDF [3], and ADIOS [4]. These tools report aggregate activities of an individual job running on the HPC system. This information can often be collected by using a low-overhead probing mechanism within the application. Additionally, in order to reduce variance within the information collected, they are run multiple times at similar times of the day to account for temporal variance.

Tracing tools collect operation-level information of I/O within the applications. This information is often collected during the software's runtime with detailed granularity and is nonaggregated over the application or system software. These tools are often built for a specific layer of the software stack, such as an application, higher-level I/O library, or storage system logs. Examples of these tools include Recorder [7], Darshan (with DXT) [5], Score-P [6], and Vampir [2].

Monitoring tools are passive tools that hook into existing applications to extract system-level information. These tools are built from a system-centric point of view and are often deployed at the system software layer. These tools use a passive probing mechanism to efficiently monitor the I/O activity, hardware health, and even the overall system. Examples of these tools, such as LDMS [10], DCDB [11], PIKA [8], TACCStats [9], and Beacon [12], are widely seen in many clusters, supercomputers, and data centers. These tools collect systemwide resource utilization and performance counter information from well-known sources (e.g., `/proc`), which can include I/O and storage-related information, and make it available for runtime display and postprocessing diagnostic analysis. There is widespread interest in using information from monitoring systems to understand the effects of the system on application performance. Hence, much work has enabled low overhead to support subsecond data collection, although not at the fidelity of typical application profiling tools. The tools aim to explain the I/O behavior of a single application/job.

4.1.1.2   Tools for *Analysis*

The tools in *analysis* aim to consume the activity/log data produced by the acquisition tools to extract meaningful information about the application/system. Currently, these are built as companion tools for existing acquisition tools. For instance, Darshan has its own postprocessing tools called PyDarshan [13] and VaniDL [14]. Recorder is accompanied by recorder-viz [15], which extracts information from recorder traces and visualizes them for the user. Additionally, we have a collection of multirun analysis tools such as IOMiner [19], TOKIO [18], Gauge [24], and ARCHER-LASSI [20]. Moreover, some generic visualization

and analysis toolkits further enhance analysis capabilities of the existing analysis tools such as Elastisearch [21] with Grafana [22] for Mistral [23], pandas [25] DataFrames for Darshan with generic plotting libraries, and Splunk [26] as a general log and regex parser to look for patterns. All these tools aim to extract useful information from existing acquisition tools. Hence, they also target a single application/job as their primary use case.

4.1.1.3 Tools for *Response*

The tools to provide *response* to the various I/O stakeholders are, at best, limited. We can categorize these tools as human-centric or automated. For human-centric tools, some recent work suggests providing hints through visualization plots and bottleneck analysis to assists developers and system admins to improve the I/O performance. Examples of these tools are monitoring tools providing a visual dashboard to look at activity data with simple drill up and drill down visualizations to manually catch I/O bottlenecks or using Vidya [16], a compiler-based approach to extract code structures and highlight code improvements for the user.

Some automated tools, for example Apollo [17], can provide feedback to middleware libraries to improve I/O decisions such as data placement, scheduling, and data prefetching. These tools are in the early stage of development and require a lot of polishing to be utilized on production machines.

The biggest theme in the state-of-the-art tools is the lack of support for a holistic view of I/O. Here, holistic means looking at several layers of the software stack and multiple applications running in a workflow. A plethora of tools have been developed for different scopes of information. However, these tools are incompatible with each other in all three stages needed to understand the I/O behavior accurately.

4.1.2 Gaps

In this subsection we look at the individual steps involved in understanding I/O behavior and identify the gaps in modern workflows.

The tools for data acquisition lack support for capturing I/O behavior on complex systems and workflows. We identify three major gaps in acquiring complete information from modern systems.

4.1.2.1 Low-Fidelity Acquisition

First is *low-fidelity acquisition*. The diversification of tools has led to lower fidelity of collected data, affecting the collection of required features to understand I/O behavior. This further results in non-reproducible results logs. For instance, data acquisition through current state-of-the-art tools is not reproducible on the same system. Furthermore, the acquired logs are tightly coupled with the system architectures and cannot be portably transferred to other machines with similar architectures. This gap is even wider when we consider multiple applications across geodistributed clusters working together.

4.1.2.2 Missing Hierarchical Scope

The second gap is *non-hierarchical scope*. Current state-of-the-art tools do not consider I/O at multiple levels of scope, such as process, job, workflow, and system. This gap often leads to restrictive analysis capabilities, which hinder the ability to observe I/O bottlenecks at different levels and their propagation within the system. This is even direr with the rise of several high-level I/O libraries because I/O operations by the application do not match the I/O seen by the underlying storage system.

4.1.2.3   Missing Compatibility of Logs

The third gap is *noncompatible logs*. Every tool has its own format for collecting and storing I/O information. These representations are often motivated based on the level of information collection and scope of the data acquisition. However, these representations are not compatible with each other. Hence, users analyzing multiple logs must build custom merging tools for their use cases, leading to restrictive information about their I/O behavior.

4.1.2.4   Gaps in Data Analysis

The tools for I/O analysis are closely tied with the acquisition tools. This leads to the following four gaps in data analysis.

First, *incompatible formats* cannot be associated with analysis because of the different scope and target of collected data. Hence, stitching these logs together is a critical gap that needs to be filled in the analysis space.

Second, the *complexity of analysis* is limited to simple statistical graphs. The analysis needs to be evolved to support complex I/O analysis such as performance bottleneck identification, I/O roofline, and error and fault detection.

Third, *standardization of analysis* is nonexistent in the current tools. The standardization needs to provide common API, functionality, and visualizations that analysts as a community can extend.

Fourth, tools for the *response* have several gaps that need to be addressed for a cohesive understanding of I/O behavior. These gaps can be divided into three categories.

First, *response standardization APIs* can enable users and system software libraries such as schedulers, buffering software, and prefetchers to improve the I/O performance of the overall system by providing feedback to these complex systems and libraries.

Second, *I/O quality of service* is lacking in modern storage systems. The response tools can predict I/O expectations based on temporal and spatial information and drive several system optimization algorithms or even I/O policies.

Third, *autotuning capabilities* are not present in the feedback tools. These capabilities can enhance feedback accuracy as these tools learn more about the system and application behaviors.

4.1.2.5   Gaps in understanding I/O behavior

Some gaps in understanding I/O behavior span across all tools and services. These can be categorized as systems, geolocations, and stakeholders. In terms of systems, we see the growth of heterogeneous resources in modern HPC systems. The tools to understand I/O behavior need to evolve past monolithic parallel file system design to heterogeneous storage cluster architectures. The tools have to capture the heterogeneity with a low-overhead, modular design for adapting to new technologies and with a flexible metric system for different hardware. In terms of geodistributed systems, the tools need to address the reproducibility and portability of I/O behavior across these systems. We need to build models that are transferable across machines and are platform-independent.

Moreover, the whole process of understanding I/O behavior needs to cater to the different stakeholders involved: I/O users, I/O practitioners, and I/O researchers. Users require the understanding that their application is getting "good" I/O performance. This is currently not defined and often ignored by the users. I/O practitioners require systemwide information and statistics over multiple applications. These should be presented in the form of visualizations

similar to those provided by compute-centric monitoring tools. I/O researchers require detailed I/O patterns (e.g., overlapping I/O, write-after-write patterns, and file/block size information) or even raw data accesses.

Addressing these gaps across the different stages of tools, multiple application workflows, heterogeneous storage solutions, and multiple stakeholders is essential for meeting the growing I/O challenges in the HPC community.

4.1.3 Challenges

Understanding I/O behavior for multitenant, heterogeneous, and geodistributed systems running complex multistage workflows depends on closing the gaps in tools we described in preceding sections. However, we identified several challenges that the community faces in this path.

4.1.3.1 Multistep Tools

The multistep nature of the tool from acquisition to response makes it hard for researchers to build a cohesive toolkit that can tame the hardware, software, and application workflow complexity all at once. Therefore, we believe approaching this problem in a modular way (i.e., one step at a time) could help us take our first step toward a holistic set of tools for understanding I/O behavior. We discuss the key challenges we envision in each of these three steps as follows.

Data Sources and Acquisition for I/O faces three challenges: collection granularity, quality vs. performance trade-off, and interoperability. First, we have different tools for acquiring a different level of information for each software stack and heterogeneous hardware. However, the nature of collected data varies so diversely that it is challenging to combine different data collected by different acquisition tools.

Second, most acquisition tools are based on the trade-off of performance vs. quality of data collected. Since the required nature of data varies for each stakeholder, it is a persistent challenge to identify the bare minimum set of metrics that acquisition tools should always collect. This decision needs standardization and should not vary based on an HPC site.

Third, the diversification of the tools for data acquisition has resulted in a lot of manual effort to make a group of tools interoperable. Currently, this is done based on the research requirement without any standardization. Researchers build their own parser over existing tools to interpret data for their research. These challenges hinder our development toward a holistic set of tools for data acquisitions for I/O.

Analysis of I/O logs is critical for understanding the environment and the nature of the application. However, currently analysis is tightly coupled with the acquisition step. Hence, building comprehensive analysis tools for understanding I/O is challenging for the following three reasons.

First, *tight coupling with data acquisition tools* makes every analysis engine out there nontransferable because of the unique format for each log. Additionally, analysis tools often are bounded the resolution of data provided by the acquisition tools. We need to move away from non-interoperable tools to a standardized log format for capturing I/O behavior so that we can build analysis tools that can be used and developed by the community (e.g., the sklearn package in Python for machine learning works on any data set that conforms to a particular format).

Second, *a lack of standard analysis definitions* within the tools results in every analysis tool recreating simple statistical functions such as aggregate I/O, observed data access patterns, and small vs large I/O. As we move to more complex analysis, most of the current

analysis tools fall short of providing the required flexibility for data analysts to experiment on. Also, the nature of the standardized analysis is unknown. Some potential venues are application I/O performance, I/O efficiency, I/O bottleneck analysis, parallel vs. serial I/O, detection of producer-consumer patterns, or even I/O performance prediction metrics.

Third, *analysis performance* is often neglected and performed serially. Consuming logs for large-scale application from several different software layers demands smart analysis frameworks such as Dask [27], Spark [28], and Hive [29]. Currently, the tools process these logs serially with no distributed or out-of-core analysis. As the resolution of data increases along with multicomponent logs, smarter analysis engines will be needed that can scale efficiently on modern HPC systems.

To provide meaningful responses to the storage system or users, we need to be able to convert our analysis into "actionable" items. However, actionable items are not well defined in the literature. Some efforts have been made by auto-optimizers such as Vidya and autotuning tools to define them at the application or system level, respectively. However, the actionable items lack standardization. Additionally, the delivery mechanism for response (e.g., online APIs or passive event log) is ambiguous for tools to implement meaningful actions for their middleware libraries and system systems. Another challenge with actionable items is that they require multiple data points (significantly temporally separated) with analytics to identify potential issues/feedback. Single data points are not sufficient because of performance variability. Furthermore, actionable items can have a side-effect on other applications and processes in the system. For instance, if we improve the I/O for one application, do we adversely affect others sharing the resources?

4.1.3.2   Heterogeneous Storage Environment

Heterogeneous storage environments in HPC systems lead to challenges at three levels: application, hardware/software, and system level. The scope of I/O behavior broadens as we go from the application to the system level.

At the application level, acquisition tools attached to the job have low impact with periodic logging. However, this reduces the resolution of the data collected. Full-scale logging has been shown to significantly impact the application's runtime (10–15% in some cases). This leads to a trade-off between the performance and quality of collected data by the tools.

Additionally, as we see a diversification of applications and storage resources, these tools are often utilized to perform best-fit or matching analysis for an application. These new requirements lead to the building of "yet another profiling tool" within the community.

Moreover, modern applications seldom run in isolation. They are generally part of a bigger workflow where several components exchange data among themselves. The application-centric approach in current tools limits their ability to detect and collect I/O behavior across applications, multiple heterogeneous resources (CPU and GPUs), and even different software stacks. The middleware libraries, which accelerate data management within the application, currently profile and manually manage this information within themselves.

At the hardware and software level, we see a growth in several hardware solutions with multiple levels of abstractions through middleware libraries. This complex and deep hardware and software stack (e.g., high-level libraries to low-level I/O interfaces to software abstraction through parallel file systems to the local filesystem) makes collecting I/O data extremely challenging.

Furthermore, several of these resources are allocated dynamically by the scheduler, forcing the tools to adapt a more dynamic approach to resource discovery and collecting information.

Additionally, the heterogeneity of the environment (e.g., faster node-local devices) potentially shifts the trade-off between overhead and data collected within tools.

Furthermore, the performance measurements and optimizations within tools are not portable across different systems because of machine architecture differences such as interconnect, node core and memory count, storage device type and distribution, bandwidth to storage system from compute nodes, and storage system device ages.

External tools such as profilers and tracers are often more costly than storage-level monitoring solutions at the system level. At the storage level, however, these monitoring tools lose the application behavior. Additionally, the logs at the storage level grow rapidly and hence cannot be left running.

Furthermore, since storage is often a shared resource, the potential I/O problems are much broader than the storage device or software. These problems could stem from the interconnect, a memory bus, or other system components. This diversity prompts us to monitor more of the system from the I/O perspective to understand the actual source of issues, since storage device contention, hardware issues (slowly failing components), and latencies are less reliably the source of the problem than they were in the days of hard disk drives.

Assessing the system state and tuning across multiple system components without any isolation is a challenging task.

4.1.3.3 Geodistributed Systems

Tools for understanding I/O behavior in a geodistributed/multisystem setup are challenging from two aspects: the scope of I/O behavior and the nature of application workflows.

In terms of *scope*, the primary challenge is to gain global insights from the system. This is because a performance bottleneck for a particular workflow could be attributed to individual components all the way to the topology of the system. This situation dictates the need for a comprehensive view of the overall system.

Moreover, the limited scope of optimization (e.g., a job or application) can adversely impact other applications in the system. The reason is that I/O tuning is often on a shared resource that can adversely affect other applications if prioritized for one application.

Additionally, multitenant nodes (e.g., fat nodes, CPU + GPU applications, or in situ computations) require tools to span multiple applications because of the shared nature of the resources. Application workflows span multiple systems that could be potentially geographically distributed. The tools for analyzing such I/O behaviors need to be cross-platform and highly portable. Analyzing the I/O over architecturally different HPC sites is extremely challenging, even if application behavior is collected.

Furthermore, the data dependencies from workflow make temporal relations of data access important. However, the capturing resolution of tools along with clock skewness makes this analysis extremely hard.

Furthermore, the current tools do not support or provide enough information for workflow-specific analysis. We need to move toward more portable, holistic, and robust tools that address these challenges.

4.1.4 Next Steps and Recommendations

In this subsection we present four groups of high-priority next steps that can help drive the tool community forward. First, the primary goal of the tools is to provide insight. The community should put priority **focus on analysis** as opposed to getting stuck in optimization of design, particularly in data collection techniques.

Table 1 Tools Chapter Summary. Gaps/Challenges: green/yellow/red for increasing difficulty/large problems. Gaps and Challenges exist for all categories. For Next Steps: green/yellow/red for least to most important.

		Gaps	Challenges	Next Steps
Multistage	Acquisition	yellow	green	green
	Analysis	red	yellow	red
	Response	red	red	red
Heterogeneous Environment		yellow	green	yellow
Geodistributed/Multisystem		red	red	red
IO Stakeholders	Admin	yellow	yellow	yellow
	Users	red	red	red
	Researcher	green	green	green

Second, **agreement on the relevant I/O performance metrics** is required in order to ensure that the tools are capturing the necessary information to accurately characterize I/O behavior.

Third, success of tools relies on the adoption of tools by stakeholders. **Development of a compelling set of use cases** is essential to demonstrate potential benefit to users and system administrators. A particular challenge in the application of tools is that a single user may derive limited benefit in performance tuning of I/O within that user's application. Given the interdependence of applications due to shared resources, bigger gains may be obtained by using tools to understand applications' I/0 behavior and use that information to enable the system (through enacting fixes at the hardware, middleware, I/O libraries, system software, and subsystem levels) to automatically manage the overall system efficiency. Aggregate user performance and workload throughput may be the level at which substantial value from tuning may be realized.

Furthermore, a **balance of tools** is needed to provide the production hardening required for use and deployment while still supporting exploratory development needed for research to thrive.

References

1 V. Ahlgren et al. *Generic Monitoring System Requirements*. 2018 IEEE International Conference on Cluster Computing (CLUSTER), 2018, pp. 532–542, doi: 10.1109/CLUSTER.2018.00069.

2 Holger Brunst and Matthias Weber. Custom hot spot analysis of HPC software with the Vampir performance tool suite. In Alexey Cheptsov, Steffen Brinkmann, José Gracia, Michael M. Resch, and Wolfgang E. Nagel, editors, *Tools for High Performance Computing 2012*, pages 95–114, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

3 Jianwei Li, Wei keng Liao, Alok Choudhary, Robert Ross, Rajeev Thakur, William Gropp, Rob Latham, Andrew Siegel, Brad Gallagher, and Michael Zingale. Parallel netCDF: A high-performance scientific I/O interface. *SC Conference*, 0:39, 2003.

4 Jay Lofstead and Robert Ross. Insights for exascale I/O APIs from building a petascale I/O API. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 87:1–87:12, New York, NY, USA, 2013. ACM.

5 Cong Xu, Shane Snyder, Omkar Kulkarni, Vishwanath Venkatesan, Philip Carns, Suren Byna, Robert Sisneros, and Kalyana Chadalavada. "dxt: Darshan extended tracing". In *CUG*, 2017.

6 Jan Frenzel, Kim Feldhoff, Rene Jaekel, and Ralph Mueller-Pfefferkorn. Tracing of multi-threaded Java applications in Score-P using bytecode instrumentation. In *ARCS Workshop 2018; 31th International Conference on Architecture of Computing Systems*, pages 1–8. VDE, 2018.

7 Chen Wang, Jinghan Sun, Marc Snir, Kathryn Mohror, and Elsa Gonsiorowski. Recorder 2.0: Efficient parallel I/O tracing and analysis. In *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1–8. IEEE, 2020.

8 Robert Dietrich, Frank Winkler, Andreas Knüpfer, and Wolfgang Nagel. Pika: Center-wide and job-aware cluster monitoring. In *2020 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 424–432. IEEE, 2020.

9 Todd Evans, William L Barth, James C Browne, Robert L DeLeon, Thomas R Furlani, Steven M Gallo, Matthew D Jones, and Abani K Patra. Comprehensive resource use monitoring for HPC systems with TACC stats. In *2014 First International Workshop on HPC User Support Tools*, pages 13–21. IEEE, 2014.

10 Steven Feldman, Deli Zhang, Damian Dechev, and James Brandt. Extending LDMS to enable performance monitoring in multi-core applications. In *2015 IEEE International Conference on Cluster Computing*, pages 717–720. IEEE, 2015.

11 Alessio Netti, Micha Müller, Axel Auweter, Carla Guillen, Michael Ott, Daniele Tafani, and Martin Schulz. From facility to application sensor data: modular, continuous and holistic monitoring with DCDB. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–27, 2019.

12 Bin Yang, Xu Ji, Xiaosong Ma, Xiyang Wang, Tianyu Zhang, Xiupeng Zhu, Nosayba El-Sayed, Haidong Lan, Yibo Yang, Jidong Zhai, et al. End-to-end I/O monitoring on a leading supercomputer. In *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, pages 379–394, 2019.

13 Argonne National Laboratory. PyDarshan: HPC I/O characterization tool. `https://www.mcs.anl.gov/research/projects/darshan/docs/pydarshan/index.html`.

14 Hariharan Devarajan, Huihuo Zheng, Xian-He Sun, and Venkatram Vishwanath. Understanding I/O behavior of scientific deep learning applications in HPC systems. 2020.

15 UIUC. Recorder-Viz. `https://github.com/wangvsa/recorder-viz`.

16 Hariharan Devarajan, Anthony Kougkas, Prajwal Challa, and Xian-He Sun. Vidya: Performing code-block I/O characterization for data access optimization. In *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*, pp. 255–264. IEEE, 2018.

17 Neeraj Rajesh, Hariharan Devarajan, Jaime Cernuda Garcia, Keith Bateman, Luke Logan, Jie Ye, Anthony Kougkas, and Xian-He Sun. Apollo: An ML-assisted real-time storage resource observer. In *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing*, pp. 147-159. 2020.

18 Glenn K Lockwood, Nicholas J Wright, Shane Snyder, Philip Carns, George Brown, and Kevin Harms. Tokio on ClusterStor: connecting standard tools to enable holistic I/O performance analysis. Technical report, Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2018.

19 Teng Wang, Shane Snyder, Glenn Lockwood, Philip Carns, Nicholas Wright, and Suren 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. IEEE, 2018.

20 Karthee Sivalingam, Harvey Richardson, Adrian Tate, and Martin Lafferty. Lassi: metric based I/O analytics for HPC. In *2019 Spring Simulation Conference (SpringSim)*, pages 1–12. IEEE, 2019.

21 Oleksii Kononenko, Olga Baysal, Reid Holmes, and Michael W Godfrey. Mining modern repositories with Elasticsearch. In *Proceedings of the 11th working conference on mining software repositories*, pages 328–331, 2014.

22 Daniel Thalmann. An interactive data visualization system. *Software: Practice and Experience*, 14(3):277–290, 1984.

23 Julian Kunkel, Eugen Betke, Matt Bryson, Philip Carns, Rosemary Francis, Wolfgang Frings, Roland Laifer, and Sandra Mendez. Tools for analyzing parallel I/O. In *High Performance Computing: ISC High Performance 2018 International Workshops, Frankfurt/-Main, Germany, June 28, 2018, Revised Selected Papers*, volume 11203, page 49. Springer, 2019.

24 Del Rosario, Eliakin, Mikaela Currier, Mihailo Isakov, Sandeep Madireddy, Prasanna Balaprakash, Philip Carns, Robert B. Ross, Kevin Harms, Shane Snyder, and Michel A. Kinsy. Gauge: An interactive data-driven visualization tool for HPC application I/O performance analysis. In *2020 IEEE/ACM Fifth International Parallel Data Systems Workshop (PDSW)*, pp. 15–21. IEEE, 2020.

25 Wes McKinney et al. pandas: a foundational Python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9, 2011.

26 Jon Stearley, Sophia Corwell, and Ken Lord. Bridging the gaps: Joining information sources with Splunk. In *SLAML*, 2010.

27 Matthew Rocklin. Dask: Parallel computation with blocked algorithms and task scheduling. In *Proceedings of the 14th Python in Science conference*, vol. 130, p. 136. Austin, TX: SciPy, 2015.

28 Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud 10*, no. 10-10 (2010): 95.

29 Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. Hive: a warehousing solution over a Map-Reduce framework. *Proceedings of the VLDB Endowment 2*, no. 2 (2009): 1626-1629.

4.2   Deep Dive Topic: Data Sources and Acquisition

Stefano Markidis (KTH Royal Institute of Technology – Stockholm, SE)
Sivalingam Karthee (Huawei Technologies – Reading, GB)
Sandra Mendez (Barcelona Supercomputing Center, ES)
Roland Laifer (KIT – Karlsruher Institut für Technologie, DE)
Osamu Tatebe (University of Tsukuba, JP)
Michèle Weiland (EPCC, The University of Edinburgh, GB)

This section explores the extraction of the raw information that guides our understanding of I/O behavior in scientific and data-intensive computing. Potential data sources include both hardware (e.g., disks, networks, and compute nodes) and software (e.g., file systems, libraries, and applications). The scope of data acquisition can vary from application-level profiles all the way to full-scale data center telemetry. The following sections summarize the state of the art in data sources and acquisition, identify gaps and challenges, and propose recommendations for how to address them.

4.2.1 State of the Art

4.2.1.1 Benchmarks

Several benchmark programs have been developed to evaluate storage systems with various metrics. Benchmark programs measure the performance under several I/O access patterns. An excellent list of parallel I/O benchmarks, applications, and traces is available at `https://www.mcs.anl.gov/~thakur/pio-benchmarks.html` and for benchmarks at `https://www.vi4io.org/tools/benchmarks/`.

Benchmarks can be divided into two categories: microbenchmarks and application benchmarks. Microbenchmarks measure performance using a simple access pattern. Typical microbenchmark tools are IOR and MDtest, available at `https://github.com/hpc/ior`. IOR is a parallel I/O benchmark that can be used to measure the bandwidth of parallel storage systems using various interfaces and access patterns. MDtest tests the peak metadata rates of storage systems under different directory structures. The IO500[1] combines these benchmarks into meaningful patterns.

Application benchmarks measure the I/O performance of real applications. Application benchmarks include DLIO for scientific deep learning workloads, WRFIO for the Weather Research and Forecasting model, and MACSio to mimic I/O workloads for a wide variety of real applications.

4.2.1.2 Workload Generators

A workload generator creates a complex benchmark that exhibits the temporal and spatial performance characteristics of a combination of applications representing the total workload of a system. Workload generators analyze the jobs that are running on a system, for instance by taking a snapshot of that system over a fixed period of time, and creating a "condensed" version of that workload. The aim is for the generated workloads to exhibit performance characteristics similar to real workloads; this is particularly important at the system specification and acceptance testing stages. For I/O performance, the main objective of testing a system using a generated workload as opposed to a single benchmark application is to understand the impact of the workload on shared resources, namely, the network, the storage subsystem, and the (parallel) file system. Unlike benchmarks, workload generators create situations where multiple different applications that form part of the workload compete for resources and thus create contention. Workloads can span a broad range of application areas (traditional HPC as well as emerging application areas such as AI), performance characteristics and I/O patterns.

One example of an existing workload generator is the Kronos [1] tool, developed by the European Centre for Medium Range Weather Forecasts as part of the NEXTGenIO project.

4.2.1.3 Monitoring Data Access and Retrieval

Monitoring tools provide information about the HPC systems state such as the jobs running, state of the different components, and workload. From the point of view of I/O, all this information is collected along the I/O path. As shown in Figure 2, different components of HPC systems are sources of the data, depending on the scope of the analysis, granularity, and components selected. Because of the complexity and different information that can be obtained in each element, different monitoring tools are run for collecting information.

---

[1] `https://io500.org`
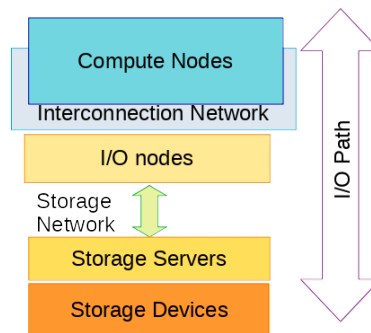
■ **Figure 2** I/O hardware along the I/O path. Each component is a source for data acquisition.

HPC centers have different monitoring tools and in several cases have their own implementation tools that provide data logs at different levels. Data can be obtained from compute nodes, storage network, I/O nodes, storage nodes, and storage devices. For example, at the compute node level, we can monitor information related to memory, frequency, CPU usage, hardware counters, and so on. Although there is an effort to provide I/O data at this level, it depends on the file system and tools provided to capture the information for each job, file, application, or user.

4.2.1.4   Storage System Health Checks

Checking the health of the storage system is an important data source for understanding the I/O behavior. For example, if a Lustre file system is down, the I/O of all applications will hang and then will normally continue after the file system is available again. Or if a part of the system is degraded, for example, a RAID rebuild after disk failure, failed redundant components such as servers or RAID controllers, or reduced speed on network links, this might have a huge impact on the I/O performance. Such issues are likely with huge storage systems. Sometimes the issues are healed automatically; for instance, if a storage server hangs and does not respond to heartbeat messages, it might be automatically killed and restarted.

4.2.1.5   High-Level Libraries and APIs for I/O

Large-scale HPC applications use high-level libraries and APIs to optimized read and write operations from/to the parallel file systems. Such high-level HPC libraries allow for coordinating these operations across several processes, reading and writing operations to shared files, and provide means to optimize the efficiency of the operations, for example, by collecting several requests from multiple processes and merging them.

MPI-I/O is among the most used approaches for parallel I/O. MPI-I/O was first featured in MPI-2 and released in 1997 [2]. MPI is a convenient setting for parallel I/O since write and read operations can be conceived as send and receive operations. Moreover, MPI provides mechanisms for collective operations exploiting communicators and noncontiguous data access with MPI derived datatypes. MPI-I/O implementations, such as ROMIO, provide two-phase and data-sieving optimization techniques for parallel I/O on HPC systems [3].

Among other important HPC libraries for parallel I/O are HDF5 and NetCDF. HDF5 (Hierarchical Data Format) is an open-source library that supports parallel I/O [4]. It provides a machine-independent data storage format and user-defined datatypes and metadata.

NetCDF (Network Common Data Form) is a software library that provides machine-independent data formats to support operations on array-oriented scientific data. NetCDF is used largely by the weather and climate simulation community.

4.2.1.6 Metrics

The most common I/O metrics are as follows:
- Bandwidth (read & write)
- I/O operation per second (IOPs)
- Metadata operations (stat, open, close, create, remove, etc.)
- Latency, which can be useful as an indication whether the system is overloaded or something is broken

Other metrics can be more specific, for example, measuring time for file system operations such as creating a small file or checking the queue depth of storage devices. At the device level, some storage arrays are able to report the worst time for read and write operations on their devices.

Depending on the point of the view of the I/O analysis, metrics report different values. For example, tools such as Darshan capture information related to all the files open by the application but cannot obtain data at the system level. If the analyst needs information at the system level (e.g., at the storage node level), other data must be collected in order to obtain the appropriate metric. Having a metric in each component could be advantageous because we could identify the slowest I/O component and the possible source of an I/O bottleneck, but it is not trivial to measure the same metrics in each component.

Furthermore, metrics depend on the I/O patterns, and it is not trivial to define a value to have an indicator of poor or appropriate I/O performance. For example, if one needs to have an indicator of I/O performance for an specific I/O system, one should use benchmarks configured for specific patterns. For this situation, the IO500 score is a good indication of performance for a range of access patterns.

4.2.2 Gaps

A policy is needed for making the I/O telemetry and log collection to be on at all times except for extreme conditions. Networking telemetry, for example, is on, but I/O profiling is either opt-in or opt-out in most cases.

4.2.2.1 Benchmarks and Workload Generators

Numerous benchmarks are regularly used on HPC systems (from HPL and HPCG to STREAM and IOR) to derive the performance of the whole system in production. However, the practice of using workloads (in particular, generated workloads that represent a specific system load) to assess the performance of HPC systems is uncommon. For I/O performance in particular, this is a problem because single benchmarks cannot replicate the type of shared resource contention that workloads will inevitably encounter. This is a clear gap in the evaluation of I/O subsystems.

4.2.2.2 Monitoring Data Access and Retrieval

Data from the whole system is needed in order to have a holistic view of the I/O behavior. A monitoring tool or a set of monitoring tools could provide a holistic view of the I/O behavior at different levels or granularity. For example, if a job opened a file at runtime, one could

track how the resources were being used by such a job for that file. This will require a common log data format for the monitored data that considers the granularity and level of the data.

4.2.2.3 Storage System Health Checks

System administrators are usually aware of an unhealthy or degraded storage system since they get alerts and have access to corresponding monitoring systems. Users, however, are frequently unaware of such issues. This is a gap since it might help users understand the I/O behavior or the reason for I/O performance variability.

4.2.2.4 High-Level Libraries and APIs for I/O

New and emerging storage systems, such as heterogeneous storage architectures and object stores, require the extension of established parallel I/O libraries and the creation of new approaches to enable the usage of these systems [6].

In particular, high-Level libraries need to consider that multiple storage might be available to an application: for instance, there might be a storage system on a compute node or shared by compute nodes and global storage shared among all the compute nodes. Ongoing research is extending and designing new libraries for these platforms. However, no established approach for these efforts exists. In addition, new storage technologies such as object stores are gaining more space on HPC systems. There exist emerging high-level APIs for programming object stores, such as Intel's DAOS [7] and Seagate Motr [5]. However, it is not clear how MPI I/O and other traditional I/O parallel approaches need to change to support object stores.

4.2.2.5 Metrics

Metrics differ depending of the I/O component monitored, because an I/O operation changes as it is processed along the data path. Therefore metrics cannot be obtained with the same tool in all the I/O components or with the same time interval.

4.2.3 Challenges

4.2.3.1 Benchmarks

Numerous benchmarks and I/O kernels exist that represent different I/O patterns and are used to evaluate the different components of the I/O software stack. Therefore, the main challenge is to provide a guide for selecting an appropriate benchmark or set of benchmarks to evaluate the performance at different levels and for the different I/O analyst roles (user, developer, or administrator).

Also needed are benchmarks for applications such as deep learning or I/O workflow in order to analyze whether we should use existing benchmarks or we need to implement new ones. For example, for deep learning applications several data formats and frameworks exist that present different I/O behavior; therefore a benchmark may be needed that allows selecting the data format and framework to best represent the I/O behavior of the deep learning application.

4.2.3.2   Workload Generators

Benchmarking whole system behavior with workloads is challenging for multiple reasons.

1. Generating the workloads is difficult because, in order for them to be representative, the whole system workload that will inform the generated workload has to be analyzed carefully. This process involves monitoring, recording, and interpreting the temporal and spatial performance characteristics of multiple snapshots of a system under load.

2. Using a generated workload as a mandatory benchmark as part of a procurement would deliver the highest impact, because the characteristics of the benchmark would directly influence the design of the system. Doing so is difficult in practice, however, because vendors often do not have access to large systems in house and thus have to project performance from a small to a large system. Such projection is already difficult with single benchmarks.

4.2.3.3   Monitoring Data Access and Retrieval

The main challenges in monitoring data are as follows.

- Monitor data that allows correlating the I/O performance with computation, communication, or memory performance issues.
- Monitor data that allows tracking I/O performance issues along the data path to identify I/O bottlenecks sources.
- Evaluate how to monitor the influence of the memory usage on the I/O behavior.
- Evaluate what data needs to be monitored in heterogeneous compute nodes such as CPU+GPU, because these nodes have their own I/O techniques to reduce the impact of transfer data between the host and GPU devices.
- Define and decide what and how to monitor data for I/O workflows.

4.2.3.4   Storage System Health Checks

System providers do not report storage-system-related issues for multiple reasons.

- They do not want to report every issue since this looks like the system is frequently in bad shape.
- They do not want to bother users with reduced performance that could also be caused by other jobs competing on the same resources.
- This information is usually available automatically only on internal systems. Site-specific solutions might be required to make this information available for users.

4.2.3.5   High-Level Libraries and APIs for I/O

Developers of libraries for parallel I/O face two main challenges. The first is the rise of heterogeneous systems with local (to compute nodes) and global storage systems. The second challenge is the uptake of new storage technologies, such as object stores, with different data consistency. The main challenge for library developers is understanding how to express parallel I/O in existing or new frameworks to take advantage of these new emerging systems without precluding the possibility of optimization.

4.2.3.6   Metrics

In order to allow I/O analysis by users, administrators, and developers, the metrics to be collected need to be clearly defined. Depending on the role of the I/O analyst, the metrics differ. The information therefore should be shown at different granularity levels such as job, application, file, and file system.

Furthermore, current I/O workflows present anther challenge. Are classical metrics sufficient for measuring I/O performance, or must we define composed metrics or new metrics?

4.2.4 Next Steps and Recommendations

4.2.4.1 Benchmarks

In IO500 a big effort was made to provide a set of benchmarks (IOR and MDtest) to measure the performance of the I/O system and define an appropriate setting of the benchmark parameters. This set is used mainly by system administrators. A similar idea could be done to guide users and developers in the selection of benchmarks.

Benchmarks for deep learning applications exist but are not focused on I/O. They can provide a start to evaluate whether they can be extended for I/O or whether new benchmarks must be developed.

4.2.4.2 Workload Generators

All system benchmarking activities should, to a degree, include generated workloads. In order to achieve this, generating representative and realistic workloads must be simplified. This process relies on better monitoring data that is explicitly tied to the workloads running on the system. At any one point, one should be able to link system behavior to workload and ideally to specific jobs in the workload. Using this information, one then can generate workloads that can expose specific and measurable system behavior.

4.2.4.3 Monitoring Data Access and Retrieval

As recommendations, we can start by considering the following:

- Identifying the information required by users, developers, and administrators to understand I/O behavior and identify I/O performance issues. The information that still is not being monitored could be added to current monitored data.
- Selecting tools that provide compatible data log format to facilitate the analysis and visualization of I/O behavior along the I/O path.
- Providing or selecting monitoring tools that can show the information considering the role of the I/O analyst (user, developer, or administrator).

4.2.4.4 Storage System Health Checks

As a first step system providers need to decide if and which storage-related issues they want to expose to their users and if they want to provide the additional effort to set up such a system. For example, they could create a web page that reports the issue with timestamps when they existed and possible impacts.

Even if system providers do not want to expose such issues, they could run regular performance checks to get system health and performance data over time and make this data available to their users. Users could then compare their application performance variation with the I/O system performance data.

4.2.4.5 High-Level Libraries and APIs for I/O

A recommendation for parallel I/O developers is to investigate the extension of established libraries, such as MPI I/O and HDF5, to use emerging heterogeneous systems and object stores efficiently. Having support in established libraries would allow porting HPC applications to new I/O solutions without using other APIs.

4.2.4.6 Metrics

Some work that defines useful metrics such as application I/O "risk" (LASSi) or I/O efficiency at application level (POP2) can be considered by HPC centers and users in the I/O analysis process.

I/O efficiency usually is measured as bandwidth relative to an empirical maximum. However, users sometimes require the I/O efficiency related with the whole application running, as is done in the POP2 project.

By using an I/O risk metric one can identify cases such as OpenFOAM that could show a "high risk" because it is (mainly) metadata intensive.

It could also useful to define a metric that assesses performance relative to system capability and capacity at the time of the job running. Also useful would be a metric for I/O workflow, for example identifying how often files are used.

References

1 https://github.com/ecmwf/kronos

2 Thakur, Rajeev, Ewing Lusk, and William Gropp. Users guide for ROMIO: A high-performance, portable MPI-IO implementation. No. ANL/MCS-TM-234. Argonne National Lab., IL (United States), 1997.

3 Thakur, Rajeev, William Gropp, and Ewing Lusk. "Data sieving and collective I/O in ROMIO." In Proceedings. Frontiers' 99. Seventh Symposium on the Frontiers of Massively Parallel Computation. IEEE, 1999.

4 Folk, Mike, Gerd Heber, Quincey Koziol, Elena Pourmal, and Dana Robinson. "An overview of the HDF5 technology suite and its applications." In Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases, pp. 36-47. 2011.

5 Narasimhamurthy, Sai, et al. "The SAGE project: a storage centric approach for exascale computing." In Proceedings of the 15th ACM International Conference on Computing Frontiers. 2018.

6 Wei-der Chien, Steven, Stefano Markidis, Rami Karim, Erwin Laure, and Sai Narasim-hamurthy. "Exploring scientific application performance using large scale object storage." In International Conference on High Performance Computing, pp. 117–130. Springer, Cham, 2018.

7 Lofstead, Jay, Ivo Jimenez, Carlos Maltzahn, Quincey Koziol, John Bent, and Eric Barton. "DAOS and friends: a proposal for an exascale storage system." In SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 585-596. IEEE, 2016.
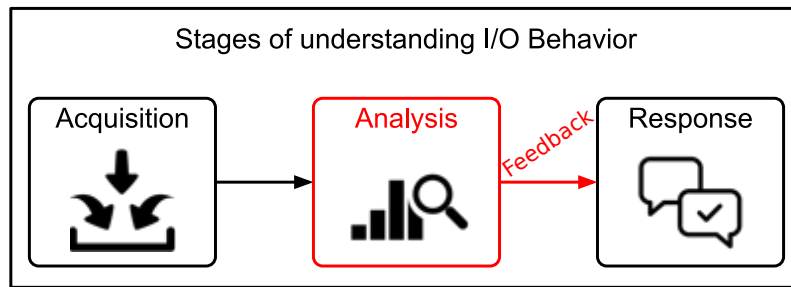
Figure 3 Relationship between data acquisition, analysis, and response as defined in Section 4.1. The focus of this section is highlighted in red. The link that connects analysis and response is referred to as *feedback* in this report.

4.3 Deep Dive Topic: Analysis

Shane Snyder (Argonne National Laboratory (ANL), USA, ssnyder@mcs.anl.gov)
Sarp Oral (Oak Ridge National Laboratory (ORNL), USA, oralhs@ornl.gov)
Suren Byna (Lawrence Berkeley National Laboratory (LBNL), USA, sbyna@lbl.gov)
Philip Carns (Argonne National Laboratory (ANL), USA, carns@mcs.anl.gov)

This section focuses on analysis of I/O behavior and the feedback resulting from that analysis. We define *analysis of I/O behavior* as interpreting and deriving meaning from I/O instrumentation data. We define *feedback* as the product of the analysis process. Feedback can be thought of as the link between analysis and the subsequent response to that analysis, such as I/O tuning or policy decisions (see Section 4.4).

Analysis and feedback may be consumed by either human stakeholders or machine algorithms. If it is to be consumed by human stakeholders, then the analysis process will likely emphasize understandable language and visualization techniques. If it is to be consumed by machine algorithms, then it will likely emphasize the use of machine-parsable, consistent formats that are suitable for ingestion by policy engines and data-mining tools.

4.3.1 State of the Art

Extensive literature exists related to the analysis of the I/O behavior of data-intensive applications, most of which can be broken down into following categories.

4.3.1.1 Existing Machine Learning Analysis Approaches

ML approaches to analysis of I/O behavior have gained a lot of traction over the years. These approaches are popular because they are effective at navigating large volumes of disparate I/O instrumentation data to provide insights into the behavior of complex storage systems. While considerable work has been done in the use of ML for understanding I/O behavior, much of it has been research oriented and therefore does not have many practical use cases at this time.

Xie et al. [1] used ML to help model and predict I/O performance and I/O variability of HPC file systems, specifically focusing on large-scale parallel checkpointing workloads that are common in HPC. Gauge [2] is an interactive I/O analysis tool that uses ML clustering techniques to hierarchically organize large collections of Darshan I/O logs for a facility to identify groups of applications with similar I/O behaviors. Isakov et al. [3] leveraged the

Gauge I/O analysis tool to help identify I/O performance bottlenecks in production and to inform potential application- and system-level improvements. Madireddy et al. [4, 5] utilized ML to develop I/O performance variability models using application I/O characterization data and file system monitoring data. In subsequent studies [6], Madireddy used I/O changepoint detection to identify noticeable changes in the performance of production file systems and used transfer learning to update I/O performance prediction models to account for these changes. Agarwal et al. [7] used active learning techniques to develop models to help optimize MPI-IO library and Lustre file system usage in applications. Wyatt [8] proposed AI4IO, a suite of AI-based tools that can help enable I/O-aware resource scheduling by predicting and mitigating I/O resource contention.

4.3.1.2 Existing Workflow Analysis Systems

Workflows have become a critical abstraction for scientific computing, providing application scientists with frameworks for effectively representing the compute and data dependencies in their campaigns. Little work has been done, however,in analyzing the I/O behavior of these workflows. Thus there is a lack of understanding of the data movement patterns typical of workflows and likely points to untapped optimizations for workflow management systems, job schedulers, I/O libraries, and so on. To address that lack, some preliminary exploratory research has been done into the I/O behavior of workflows and strategies for better understanding them.

Luttgau et al. [9] presented an approach to allow for better understanding of I/O workflow behavior by augmenting workflow systems and I/O analysis tools to be aware of each other. This work outlines challenges in connecting I/O analysis tools with workflow systems and suggests guidance for developers of each to cope with these challenges. Patel et al. [10] analyzed large amounts of Darshan logs from the Cori supercomputer to help identify how files are reused over time and across applications. While not tied to specific workflow systems, this work illustrates how implicit workflows can be identified by mining I/O instrumentation data for data dependencies. Lockwood et al. [11] analyzed numerous I/O monitoring data sources across the entire NERSC data center to gain an understanding of characteristics of data movement in typical scientific workflows and implications for facilities.

4.3.1.3 Visualization and Web Dashboards

Data visualization tools can be critical to application scientists, system admins, and I/O researchers to gain a deeper understanding of I/O behavior on HPC systems. Tools such as Grafana [12] have became increasingly popular for analyzing telemetry data in conventional data centers, but they can also be readily leveraged in HPC contexts. Existing HPC I/O analysis tools can be used to help better understand different types of captured instrumentation data as well. For example, the Darshan [13] I/O characterization tool comes with a tool for generating summary PDF reports that capture highlights about the job's I/O behavior from a corresponding Darshan log. Similarly, tools such as Tau [14] provide analysis capabilities for identifying I/O performance bottlenecks using captured application I/O traces.

A number of more comprehensive I/O visualization and analysis systems have been developed specifically for HPC platforms, such as Altair Mistral and Breeze systems [15], PIKA [16], Beacon [17], Gauge [2], and UMAMI [18]. Altair Mistral and Breeze are I/O profiling and analysis tools that can help characterize and tune application I/O dependencies, file access patterns, and so on. PIKA is a continuous monitoring system designed for use on production HPC systems that contains numerous I/O performance metrics that can be used to

characterize I/O behavior of jobs and to identify potential optimization opportunities. Beacon is an end-to-end continuous I/O monitoring system designed for the Sunway TaihuLight supercomputer that comes with corresponding I/O analysis and visualization tools that can be useful to both users and administrators. The Gauge dashboard can be used by system admins and I/O researchers to cluster applications based on similarity of I/O behavior to determine characteristics of different classes of applications that typically run on a given HPC system. UMAMI can provide historical I/O performance context for applications across a range of application- and system-level metrics to help gain insights into how these metrics correlate with general I/O performance over time. While the Altair tools, PIKA, and Beacon have already been successfully demonstrated in production, UMAMI and Gauge are much more research-quality tools at this point.

Prior research studies focused on visualizing I/O behavior could also provide inspiration for the design of I/O analysis tools. For example, Sigovan et al. [19] present some visualization methodologies for enabling better understanding of parallel I/O traces that could be adopted by the tools presented above or by new I/O analysis tools.

4.3.1.4   Technologies Used Outside of HPC

Beyond HPC, a number of I/O analysis tools are available that may not be easily deployed on HPC systems but that could influence the design of HPC-specific I/O analysis tools. For instance, Facebook has developed the HALO [20] system for monitoring and analyzing a number of different hardware-centric metrics on its storage networks. The VMWare I/O analyzer [21] provides a similar tool for measuring various I/O performance metrics and diagnosing issues in virtualized environments. Additionally, Kubernetes Datadog [22] can be used to collect and analyze similar I/O performance metrics in the context of Kubernetes clusters.

4.3.1.5   Technologies Used Outside of I/O

Beyond I/O-related technologies, numerous performance analysis toolkits are available that could be used to motivate new capabilities for HPC I/O analysis tools. One example is Intel VTune [23], a CPU profiling and analysis tool that can optimize application and system performance for different computing environments. Another example is TensorBoard [24], a tool for measuring various performance metrics from TensorFlow applications and for providing detailed visual analysis of the captured metrics.

4.3.1.6   Usage of I/O Signatures

Various research projects have also explored the use of compact representations of I/O workload behavior called *I/O signatures*. Researchers like I/O signatures because they can be used to reproduce or replay I/O workloads without relying on a costly full trace of the application. Byna et al. [25] outlined a set of signature classifications relevant to HPC and used these signatures to help guide application I/O prefetch strategies. Feng et al.[32] presented a tool for capturing I/O signatures, called IOSig+, based on the initial classification of I/O signatures [25]. Behzad et al. [26] similarly used I/O signatures to classify applications at runtime and to help select various I/O tuning parameters based on the observed application I/O signature. Dorier et al. [27] presented Omnisc'IO, an approach that builds grammar-based models of application I/O workloads and uses these models to help predict application I/O behavior.

4.3.1.7 General Studies on Analyzing I/O Behavior

In addition to the research areas covered above, many general studies geared toward analyzing HPC I/O behavior have made important contributions in this space. One notable study is the Charisma project [28], which provided early results for how to best characterize file access patterns of parallel scientific applications. This research enabled many insights that guide not only the design of HPC file systems but also the design of HPC I/O characterization tools such as Darshan. Carns et al. [29] performed an evaluation similar to that of the Charisma project, this time using Darshan to analyze thousands of jobs to identify potential tuning opportunities and systemwide I/O trends at the Argonne Leadership Computing Facility. Luu et al. [30] performed a multiyear, multiplatform I/O study using Darshan logs to help understand the behavior of different types of application I/O workloads over time and across different platforms.

4.3.2 Gaps

I/O pattern/performance analysis is a multivariate, multidimension, complex and dynamic problem. Although state-of-the-art research and practice have made strides in trying to manage this problem, significant shortcomings remain in approach, technology, and capabilities.

Gaps in I/O pattern/performance analysis and feedback can be categorized in four broad areas:
1. Scope
2. Infrastructure
3. Common terminology and communication
4. Limited feedback

4.3.2.1 Scope

Our current I/O pattern/performance analysis and feedback techniques can be viewed horizontally across the I/O software stack or vertically across a file or a storage system. Our current capabilities are, by and large, vertical and focused on a single application and do not take into account the system view. Furthermore, our vertical capabilities mostly target the I/O client side and do not cover the server side of the problem as well. Also, these techniques are mostly file oriented and miss the holistic data view of a given application.

The vertical integration of telemetry data and logs from various layers across the stack, including the client and server sides, the network, and the scheduler, is essential for getting a complete view of how an application's I/O pattern and performance progresses and functions. While this is a crucial capability, the current state of the art is far from it.

The horizontal integration of the I/O pattern/performance telemetry data and logs from all applications running on a given system can provide a systemwide view of how the I/O subsystem is functioning at a given time or how applications are taking advantage of it. Current state of the art has some capabilities in providing a semi-complete picture here, but often a gap remains in correlating the data with other systemwide data streams, such as the network or the scheduler.

With these limited and narrow-scoped glimpses into the complex problem, the analysis or the feedback to consumers (e.g., users, administrators) commonly lacks reference points or context.

Also, a big gap in our arsenal is the lack of capabilities in analyzing workflows. The workflow I/O is becoming more prevalent with emerging edge computing and the coupling of scientific experiments and instruments with HPC data centers and resources.

4.3.2.2 Infrastructure

The I/O pattern/performance analysis and feedback are an acute problem, and most HPC data centers are trying to create solutions to it in an ad hoc manner. While these tackle different aspects of the problem, they are data centered and system specific and not easily portable. This creates a duplication of effort across multiple data centers, wasting scarce and precious resources and being far from effective for answering the community's collective needs.

Also, the limited analysis and feedback capabilities available are tailor-built for solving specific problems and are not versatile enough to be used by both machine and human consumers. With the proliferation of AI techniques, having the capability of currently feeding information to both is becoming more pressing.

4.3.2.3 Common Terminology and Communication

An apparent lack of common language and terminology exists among the three largest I/O stakeholders: scientific users, I/O practitioners (system architects and administrators), and I/O researchers (tool builders).

Often a user raises an issue pointing to a potential problem with the I/O subsystem. The I/O subsystem is usually the canary in the coal mine and can exhibit the first symptoms of a problem anywhere in the system, not necessarily emanating from the I/O subsystem itself. Therefore, trying to understand the full context of the user problem is crucial for conducting a proper root cause analysis, and using a common terminology across the board between the scientific users and the I/O practitioners is essential.

On the flip side, when I/O practitioners identify and point out to the user a pattern or a performance problem with a certain application, the lack of common language and terminology creates a barrier between the user and the I/O practitioner, hindering an expedited and effective solution to the problem.

Also, a lack of common language and a terminology between I/O practitioners and the researchers can create barriers for taking valuable research artifacts and deploying them on production systems or for researchers addressing pressing production I/O problems.

4.3.2.4 Limited Feedback

The feedback provided with existing technology and capabilities falls short of moving the needle in terms of better I/O subsystem usage and utilization at HPC data centers.

A limiting factor here can be the lack of clear demonstration of a cost/benefit analysis of an I/O improvement suggested to a scientific application user. The I/O routines are usually expected to consume less than 10% of the application's total runtime. Some applications are already following good I/O practices, and a negligible improvement in the I/O pattern or performance might not be good use of limited developer resources from the perspective of the scientific application team. In certain cases, however, the benefit of doing the right thing will outweigh the cost of changing the application I/O routines and dramatically increase the application efficiency or scalability. As a community, we need to careful select where we focus our efforts and what kind of feedback we provide (e.g., detail and granularity).

Also, providing comprehensive feedback on the system-level I/O activity, whether as a whole or only for a subset or a kind of application, is challenging. As an example, consider the case where an I/O practitioner needs to build an I/O workload generator mimicking the behavior of a class of small-scale applications generating heavy read requests with random offsets. The challenge here would be identifying what would be a good representative scale (for the size of files or read requests) to build a good analogue for the actual applications.

4.3.3 Challenges

I/O behavior analysis is fundamentally a complex, dynamic, and multivariate problem with no easy or visible solutions. Different facilities have diverse applications, data sources, and use cases. A stark contrast exists between I/O behavior analysis challenges and various success stories from environments with more homogeneous constraints.

Challenges in I/O behavior analysis can be categorized into four broad themes.
1. Technology
2. Complexity
3. Interoperability
4. Resources

4.3.3.1 Technological Impediments to Resolving Gaps

Providing more context and a "bigger picture" of I/O behavior requires very high volume of data from different sources, namely, I/O logs from various subsystems including network. Collecting such high-volume and fine-grained data is challenging for multiple reasons, including the overhead in collecting the data as well as in maintaining the data with indexing. HPC facilities have shown little interest in building storage systems just for handling telemetry and for analyzing feedback that could help understand I/O behavior of applications. Another technical challenge is that AI and other models that can be used to analyze or to predict I/O behavior and performance of applications are not robust enough.

4.3.3.2 Complexity

Various interdependencies among I/O software and hardware layers and heterogeneity of these solutions make understanding I/O behavior extremely complex. In addition, because storage is shared among a large number of users in a system, understanding of performance is often hindered by variability. Since variability is unpredictable or not well understood, trust in the I/O performance prediction and analysis feedback is typically low. (More discussion of variability is available in §4.1.) Moreover, some facilities have a diverse set of application programming models that exacerbate the challenge of finding common analysis.

4.3.3.3 Interoperability of Analysis Tools

A significant incompatibility exists among telemetry collected at data sources (§4.2) and I/O analysis tools (§4.1) that is leading to poor interoperability of analysis tools. Since no standards exist for what telemetry to collect, different facilities generate logs at different frequencies, coverage, diverse sets of details, and so on. Disparity in the use of schedulers also exists. For instance, an analysis tool requiring details of all the jobs running concurrently when an application's I/O was occurring may not have access to similar details in the output of different schedulers. As a result, an analysis tool that requires data in a specific format becomes specific only to the data available to it. This challenge warrants the need for native

telemetry formats across multiple data source components across multiple facilities. Another challenge is with sharing telemetry data openly for various analyses. For instance, facility security enclaves sometimes inhibit data sharing. In this case, analysis tools have to be developed in collaboration with the facilities, and requirements from these facilities may become too customized, thus challenging the goals of analysis tool interoperability.

4.3.3.4 Resources

In some cases, although analysis tools and tuning options are well established, low engagement from users in tuning I/O (§4.6) discourages facilities from justifying the effort needed to push the frontier of I/O behavior analysis. For example, using collective buffering in MPI-IO is a well-known strategy when there are a large number of small I/O requests from each MPI rank. However, identifying the poor performance caused by this inefficient use of I/O software either has to be initiated by users or has to be identified automatically by a systemwide monitoring tool. Triggering a tuning effort and contacting facility support for tuning may need first to train users. Facilities may have to provide resources for these training events as well as expertise for solving storage and I/O issues. All these efforts require resources from facilities, and facilities often struggle to justify such efforts with low demand for tuning I/O.

4.3.4 Next Steps and Recommendations

In our analysis of I/O behavior we identified the following recommendations and next steps for the community based on our survey of gaps and challenges .

4.3.4.1 Technical

Developers of analysis tools and methodologies should plan for interoperability up front. Although extensive work exists in this space, projects that start off as independent efforts seldom arrive at an integrated state organically. Required instead is a conscious design effort to be inclusive of modular and interoperable tools. Community organizations or funding agencies could help spur effort in this direction, but the responsibility ultimately lies with researchers and developers. For example, a developer of a new methodology or tool for analysing I/O behavior in workflows could begin by considering how it would fit into a broader ecosystem of platform-level or job-level analysis tools. If the tools are cognizant of each other up front, then design considerations can be taken to allow them to flow together as part of a unified tool chain.

Generalizabillity of I/O models should be considered a first-class problem. An I/O model could be developed by using analytical methods, artificial intelligence, or discrete event simulation. Regardless of the technique employed, little evidence indicates that they have been trusted for use in production environments in practice, despite the promise of potential to guide performance enhancement. The underlying cause is a lack of trust by potential consumers of those models and, in many cases, a lack of effort by model developers to try to win that trust. This disconnect could be addressed with emphasis on how to reason about the generalizability and correctness of a model. Initial validation and reproducibility efforts are valuable, but insufficient. To act on this recommendation, the community must go a step further and pursue continuous validation and refinement methods as well.

4.3.4.2 Social

I/O analysis researchers and practitioners should gather and document lessons and requirements from stakeholders in the field.   The purpose of this recommendation is to ensure that effort is expended on the right problems and that feedback from analysis tools is formatted appropriately for consumers. These lessons and requirements could take several forms. The most valuable would be broad surveys shared with the community to help inform overall activities. More narrowly scoped research efforts could also benefit from more limited focus group discussion, however. Without this key step, there is a danger that an analysis tool could distill unwanted information or could distill information into a form that inadvertently causes an educational bottleneck by requiring too much effort on the part of consumers to use.

Community standards should be developed for common analysis tasks.   Analysis tools could be made more portable and generalizable by consuming input data in standardized formats, but that topic is more appropriately covered in Section 4.2. For analysis methods themselves, the standards that are lacking are less concrete. What is needed is a better community agreement on terminology and taxonomies of I/O motifs to help structure feedback and frame discussions. The current state of the practice is that different tools employ their own jargon or overload existing jargon in ways peculiar to their analysis task. This sort of standardization may be best handled by organizations such as VI4IO [31] that span research teams, organizations, and countries. Standardization bodies that address broad issues such as these must take care that standardization efforts do not unintentionally inhibit research innovation, however.

References

1 B. Xie et al. *Applying machine learning to understand write performance of large-scale parallel filesystems.* In 2019 IEEE/ACM Fourth International Parallel Data Systems Workshop (PDSW), pp. 30–39. 2019.

2 E. Del Rosario et al. *Gauge: An interactive data-driven visualization tool for HPC application I/O performance analysis.* In 2020 IEEE/ACM Fifth International Parallel Data Systems Workshop (PDSW), pp. 15-21. 2020.

3 M. Isakov et al. *HPC I/O throughput bottleneck analysis with explainable local models.* In SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–13. 2020.

4 S. Madireddy et al. *Analysis and correlation of application I/O performance and system-wide I/O activity.* In 2017 International Conference on Networking, Architecture, and Storage (NAS), pp. 1–10. 2017.

5 S. Madireddy et al. *Machine learning based parallel I/O predictive modeling: A case study on Lustre file systems.* In International Conference on High Performance Computing, pp. 184-204. 2018.

6 S. Madireddy et al. *Adaptive learning for concept drift in application performance modeling.* In Proceedings of the 48th International Conference on Parallel Processing, pp. 1–11. 2019.

7 M. Agarwal et al. *Active learning-based automatic tuning and prediction of parallel I/O performance.* In 2019 IEEE/ACM Fourth International Parallel Data Systems Workshop (PDSW), pp. 20-29. 2019.

8 M. Wyatt. *AI4IO: A suite of AI-based tools for IO-aware HPC resource management.* University of Delaware. 2020.

9 J. Luttgau et al. *Toward understanding I/O behavior in HPC workflows.* In 2018 IEEE/ACM 3rd International Workshop on Parallel Data Storage and Data Intensive Scalable Computing Systems (PDSW-DISCS), pp. 64–75. 2018.

10   T. Patel et al. *Uncovering access, reuse, and sharing characteristics of I/O-intensive files on large-scale production HPC systems*. In 18th USENIX Conference on File and Storage Technologies (FAST 20), pp. 91–101. 2020.

11   G. Lockwood et al. *Understanding data motion in the modern HPC data center*. In 2019 IEEE/ACM Fourth International Parallel Data Systems Workshop (PDSW), pp. 74–83. 2019.

12   *Grafana: The open observability platform*. `https://grafana.com/`.

13   *Darshan I/O characterization tool*. `https://www.mcs.anl.gov/research/projects/darshan/`.

14   *TAU: Tuning and analysis utilities*. `http://tau.uoregon.edu/`.

15   *Altair Mistral: Live system telemetry and I/O monitoring*. `https://www.altair.com/mistral/`.

16   R. Dietrich et al. *PIKA: Center-wide and job-aware cluster monitoring*. In 2020 IEEE International Conference on Cluster Computing (CLUSTER), pp. 424–432. 2020.

17   B. Yang et al. *End-to-end I/O monitoring on a leading supercomputer*. In 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), pp. 379-394. 2019.

18   G. Lockwood et al. *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 and Data Intensive Scalable Computing Systems, pp. 55-60. 2017.

19   C. Sigovan et al. *A visual network analysis method for large-scale parallel I/O systems*. In 2013 IEEE 27th International Symposium on Parallel and Distributed Processing, pp. 308–319. 2013.

20   *Facebook HALO (Hardware Analytics and Lifecycle Optimization)*. `https://engineering.fb.com/2017/03/21/data-center-engineering/hardware-analytics-and-lifecycle-optimization-halo-at-facebook/`.

21   *VMware I/O analyzer*. `https://flings.vmware.com/i-o-analyzer`.

22   *Monitoring Kubernetes with Datadog*. `https://www.datadoghq.com/blog/monitoring-kubernetes-with-datadog/`.

23   *Intel VTune Profiler*. `https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/vtune-profiler.html`.

24   *TensorBoard: TensorFlow's visualization toolkit*. `https://www.tensorflow.org/tensorboard`.

25   S. Byna et al. *Parallel I/O prefetching using MPI file caching and I/O signatures*. In SC'08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing, pp. 1–12. 2008.

26   B. Behzad et al. *Pattern-driven parallel I/O tuning*. In Proceedings of the 10th Parallel Data Storage Workshop, pp. 43-48. 2015.

27   M. Dorier et al. *Omnisc'IO: A grammar-based approach to spatial and temporal I/O patterns prediction*. In SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 623-634. 2014.

28   N. Nieuwejaar et al. *File-access characteristics of parallel scientific workloads*. IEEE Transactions on Parallel and Distributed Systems 7, no. 10 (1996): 1075-1089. 1996.

29   P. Carns et al. *Understanding and improving computational science storage access through continuous characterization*. ACM Transactions on Storage (TOS) 7, no. 3 (2011): 1-26. 2011.

30   H. Luu et al. *A multiplatform study of I/O behavior on petascale supercomputers*. In Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing, pp. 33–44. 2015.

31   The Virtual Institute for I/O. `https://www.vi4io.org/`.

32 Feng, Bo, Xi Yang, Kun Feng, Yanlong Yin, and Xian-He Sun. *IOSIG+: on the Role of I/O Tracing and Analysis for Hadoop Systems.* In IEEE International Conference on Cluster Computing (CLUSTER), IEEE, pp. 62–65, 2015.

4.4 Deep Dive Topic: Enacting Actionable Responses

Andreas Knüpfer (Technische Universität Dresden, DE, andreas.knuepfer@tu-dresden.de)
Julian M. Kunkel (Universität Göttingen / GWDG, DE, julian.kunkel@gwdg.de)
Erwin Laure (Max Planck Computing and Data Facility, DE, erwin.laure@mpcdf.mpg.de)
Radita Liem (RWTH Aachen University, DE, liem@itc.rwth-aachen.de)
Frank Mueller (North Carolina State University, USA, mueller@cs.ncsu.edu)

This section describes how the output from analysis tools, current and imagined, could be utilized to provide more efficient execution of applications and workflows through optimization of I/O across systems and data centers. It identifies current state-of-the-art practices and proposes actionable approaches based on emerging I/O monitoring and analysis tools and I/O-related technologies.

4.4.1 State of the Art

4.4.1.1 Applications

Currently, users are expected to perform their own I/O optimization on their applications. This approach works to a degree with expert users but takes a sizable part of developer time. Inexperienced users often find it nearly impossible to perform any I/O optimization, since this would require deep knowledge about the underlying file systems. Particularly at smaller HPC sites with a sometimes less knowledgeable (in regard to I/O) community, actions cannot always be taken on a per-application level. Nevertheless, high-level libraries such as netCDF and HDF5 are a good starting point for any level of sophistication, since they are ubiquitously used in many HPC applications.

4.4.1.2 Workflow

On most systems, the state of the art for data management and workflows requires that users move their data around manually. Many scientific domains have their preferred workflow managers, but most of these either are not built for HPC-systems or have limited options for data locality. Data is taken from one or more specified files and written back to the specified output file. More sophisticated options such as staging into local storage or using a burst buffer need to be defined by the user and are often not built into the manager itself. While more I/O-optimized workflow managers may exist, domain scientists naturally tend to use the preferred options in their domain, which are often less HPC friendly. Even in batch schedulers, the options to define an I/O-aware workflow or even use data staging are limited.

4.4.1.3 System

Procurement and acceptance of storage systems as well as fine-tuning of file system parameters are often based on artificial benchmarks or single applications that may not be representative of real system usage.

Users have only limited insight into the status of the file systems and are usually not aware of current bottlenecks or performance degradation. Consequently, they do not understand the root cause of poor application performance and cannot identify suboptimal usage patterns in their jobs. Furthermore, they have no chance to schedule future highly demanding I/O operations regarding current system load.

4.4.1.4 Data Center

Batch schedulers for HPC data centers typically do not explicitly manage I/O resources. In contrast to compute requirements, users are not required to define I/O bandwidth or latency requirements when submitting a job. Therefore, data-intensive phases of several applications can occur at the same time, leading to significant congestion of the storage system, which decreases both the overall performance of the storage system and the I/O bandwidth seen by the individual applications.

A number of approaches do exist that could make storage an actively manageable resource in HPC data centers. On the one hand, file systems such as Lustre provide quality of service management on the level of metadata operations and I/O operations [7]. On the other hand, extensions to batch environments, such as NORNS, facilitate asynchronous staging of job input data between different storage tiers [5]. Also, many applications are, in principle, prepared to move data-intensive phases to optimize overall storage usage. Checkpointing environments such as SCR provide feedback mechanisms that allow the application to delay the execution of a checkpoint in the event that the system is currently oversubscribed [6]. Nevertheless, these approaches currently require that either the user or the administrator understand the storage requirements of applications, while tools to automatically derive these requirements and develop appropriate actions based on them (e.g., automatic staging of data between different storage tiers) are still in their infancy [8].

Most HPC data centers also do not have a consistent set of rules and guidelines on how to deal with I/O or I/O-based problems. Instead, they often use self-developed solutions to issue alerts when systems are overloaded. Furthermore, users have difficulty identifying I/O problems (beyond experiencing long application runtimes), since most data centers do not provide reports on the I/O usage of individual applications or the congestion conditions of the file system as a whole.

4.4.2 Gaps

4.4.2.1 Applications

Clearly lacking is easily digestible feedback from monitoring and analysis tools that can guide users in optimizing the I/O characteristics of their applications. Besides better tool support, more and better training and spreading knowledge about readily available I/O libraries would be beneficial.

4.4.2.2 Workflow

The diversity in the current workflow management systems makes it difficult to analyze and therefore hard to act on. Any attempts to define rules for a workflow manager based on a particular I/O analysis work only for that specific case. The lack of standardization limits the definition of any actionable steps to a narrow scope.

4.4.2.3 System

Synthetic benchmarks are missing that are able to emulate actual I/O workloads in real-world production use. Moreover, user-accessible and machine-readable monitoring and file system status information is lacking.

4.4.2.4 Data Center

Currently no consistent set of rules or guides exist on how to deal with I/O or I/O-based problems at a data center level. Typically what is used are home-brewed solutions that provide alerts if systems become overloaded.

4.4.3 Challenges

4.4.3.1 Applications

Understanding an application's I/O behavior and potentially detrimental data access patterns is not trivial and requires significant experience. Automated tools and access to monitoring data to support user insight in this area are lacking.

4.4.3.2 Workflow

Workflows are not well defined, and hence ensuring efficient I/O at an aggregate or even subcomponent level in a consistent and programmatic manner is currently difficult or even impossible depending on the complexity of the workflows being addressed.

4.4.3.3 System

Creating benchmarks that are both representative of production workloads and reproducible is difficult. They would have to be based on historical monitoring data and job traces.

4.4.3.4 Data Center

The management of I/O resources requires that users and administrators be well informed about the use of the underlying storage systems. An important gap is that users (and most administrators) generally do not have access to I/O usage statistics. Providing access to this type of data through text reports and I/O visualization and browsing tools would take significant work but could provide great benefit in terms of more efficient utilization of file systems.

New approaches to manage storage resources are already well developed within the storage research domain and are also implemented in widespread storage systems and parallel file systems. Nevertheless, their active application in data centers is typically delayed by several years. One example is the availability of on-demand file systems, which can build parallel file systems on top of very fast node-internal storage resources [1, 9]. These ad hoc file systems can be used to isolate I/O accesses across applications and from joint parallel file systems [10].

4.4.4   Next Steps and Recommendations

4.4.4.1   Applications

Focus should be put on high-level libraries such as netCDF and file formats such as HDF5. Feedback from analytics tools, such as Darshan, Tau, and Score-P [2, 4, 3], should be utilized to automatically optimize these in the context of particular applications. Such actions could be taken either on a per-application level or on a file system level: Optimize the library specifically for each application or to work best on average for the underlying file system.

4.4.4.2   Workflow

The current practice of users manually orchestrating their own data movement is inefficient because they cannot know the state of the system and how to optimize for current or future conditions. Data orchestration tools exist that users may be unaware of and that could be utilized to provide more efficient data management. By promoting these tools for use by users, data management across the system could be performed more efficiently, resulting in overall efficiency gains. In order to enable automated I/O optimization on the level of workflow management systems that can apply to many systems at once, guidelines on I/O integration should be designed. These guidelines could act as a loose standard for orchestration approaches, which could be used across the majority of workflow engines to further enhance overall data movement across applications on a system.

4.4.4.3   System

As mentioned, file systems are currently being tendered and procured on the basis of synthetic benchmarks. It would make more sense if these systems were benchmarked against I/O patterns from real applications, as is done for the computing part. Therefore, we suggest utilizing access patterns learned from real applications to tune/define PFS parameters and future designs. We note that the current parallel filesystem implementations may affect those access pattern characteristics and that this possibility needs to be taken into account.

When using the file system of an HPC system, avoiding congestion is preferable. Here available solutions should be used, for example, using quality of service for file systems [7] or extending schedulers to support data-driven workload [5]. With such an API for feedback, applications can use it to find the optimal time for I/O heavy operations such as checkpointing. Users should take into account, however, that such feedback can cause problems if too many I/O heavy jobs follow these signals.

Two steps should be taken here. First, real I/O workloads should be collected in order to plan and design new file systems on the basis of these. Second, some means of controlling I/O traffic should be created.

4.4.4.4   Data Center

The next steps to be performed on a data-center level can be derived from both the state of the art and the identified gaps.

- In a first step, users and administrators must have access to I/O usage statistics, in the form of a visual and intuitive "I/O Weather" report or as an extended text report. The report results should become comparable between data centers by driving the standardization of data/metrics format. Also, users have to be actively informed about bad I/O scores, if such a score is available.
- File system feature and performance variability study outcomes should be used to offer users hints as to what features might provide best performance for their workloads.

- Data centers have to introduce incentives to optimize I/O usage, so that users are encouraged to optimize their I/O usage. These incentives can be based on a gamification approach, where users, for example, can get recognized for their improvements or can get credit, for example, in the form of additional CPU-hours.
- An efficient usage of I/O resources can be achieved only if solutions such as quality of service, I/O management through the batch system, or ad hoc file systems are made available in data centers.
- Load should be (automatically) shifted according to "I/O Weather" between storage systems.

References

1 André Brinkmann, Kathryn Mohror, Weikuan Yu, Philip H. Carns, Toni Cortes, Scott Klasky, Alberto Miranda, Franz-Josef Pfreundt, Robert B. Ross, Marc-Andre Vef. *Ad Hoc File Systems for High-Performance Computing.* J. Comput. Sci. Technol. 35(1): 4–26 (2020)

2 Philip H. Carns, Kevin Harms, William E. Allcock, Charles Bacon, Samuel Lang, Robert Latham, Robert B. Ross. *Understanding and Improving Computational Science Storage Access through Continuous Characterization.* ACM Trans. Storage 7(3): 8:1–8:26 (2011)

3 Andreas Knüpfer, Christian Rössel, Dieter an Mey, Scott Biersdorff, Kai Diethelm, Dominic Eschweiler, Markus Geimer, Michael Gerndt, Daniel Lorenz, Allen D. Malony, Wolfgang E. Nagel, Yury Oleynik, Peter Philippen, Pavel Saviankou, Dirk Schmidl, Sameer Shende, Ronny Tschüter, Michael Wagner, Bert Wesarg, Felix Wolf. *Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope, Scalasca, TAU, and Vampir.* In Parallel Tools Workshop 2011: 79–91

4 Sameer S. Shende, Allen Malony *The TAU Parallel Performance System.* The International Journal of High Performance Computing Applications 20(2) (2006): 287–311.

5 Alberto Miranda, Adrian Jackson, Tommaso Tocci, Iakovos Panourgias, Ramon Nou. *NORNS: Extending Slurm to Support Data-Driven Workflows through Asynchronous Data Staging..* In IEEE International Conference on Cluster Computing (CLUSTER), Albuquerque, NM, USA, September 23–26, 2019

6 Kathryn Mohror, Adam Moody, Bronis R. de Supinski. *Asynchronous Checkpoint Migration with MRNet in the Scalable Checkpoint / Restart Library.* In IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN), Boston, MA, USA, June 25–28, 2012

7 Yingjin Qian, Xi Li, Shuichi Ihara, Lingfang Zeng, Jürgen Kaiser, Tim Süß, André Brinkmann. *A Configurable rule Based Classful Token Bucket Filter Network Request Scheduler for the Lustre File System.* In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Denver, CO, USA, November 12–17, 2017

8 Yingjin Qian, Xi Li, Shuichi Ihara, Andreas Dilger, Carlos Thomaz, Shilong Wang, Wen Cheng, Chunyan Li, Lingfang Zeng, Fang Wang, Dan Feng, Tim Süß, André Brinkmann. *LPCC: Hierarchical Persistent Client Caching for Lustre.* In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Denver, CO, USA, November 17–19, 2019

9 Sebastian Oeste, Marc-André Vef, Mehmet Soysal, Wolfgang E. Nagel, André Brinkmann, Achim Streit. *ADA-FS – Advanced Data Placement via Ad hoc File Systems at Extreme Scales.* Software for Exascale Computing 2020: 29–59

10 Mehmet Soysal, Marco Berghoff, Thorsten Zirwes, Marc-André Vef, Sebastian Oeste, André Brinkmann, Wolfgang E. Nagel, Achim Streit, *Using On-Demand File Systems in HPC Environments.* In 2019 International Conference on High Performance Computing & Simulation (HPCS), 2019, pp. 390–398.

4.5 Deep Dive Topic: Data Center Support

Andreas Knüpfer (Technische Universität Dresden, DE, andreas.knuepfer@tu-dresden.de)
Julian M. Kunkel (Universität Göttingen / GWDG, DE, julian.kunkel@gwdg.de)
Erwin Laure (Max Planck Computing and Data Facility, DE, erwin.laure@mpcdf.mpg.de)
Radita Liem (RWTH Aachen University, DE, liem@itc.rwth-aachen.de)
Frank Mueller (North Carolina State University, USA, mueller@cs.ncsu.edu)

4.5.1 State of the Art

The responsibilities of a data center with regard to I/O aspects are focused on a set of broad areas. For each area we identify the current state of data center support by color, where `green` denotes adequate measures/tools/strategies are already researched and mostly in place on the data center side; `yellow` means support is improvable: that is, on most data centers some aspect is missing that should be improved; `orange` means insufficient support; and `red` means largely absent support.

The broad areas of responsibilities related to I/O for data centers are as follows:

1. Procurement
   - `Storage systems`
   - `Network`
   - `Standardized benchmarking`
   - `Monitoring software`
2. Operating the system
   - `Maintenance`
   - `I/O monitoring`
   - `Accounting` `Job-specific monitoring)`
   - `Hardware/software tuning`
   - `Policy enforcement`
   - `User-specific dedicated resource allocation`
   - `Data security and safety (isolating users/groups, access permissions, backups)`
3. Supporting the users
   - `Giving feedback from monitoring` – `Showing the I/O weather`
   - `Providing joint code optimizations (brainware missing)`
   - `Tuning hardware/software parameters for applications`
   - `Understanding what users are doing and really need`
4. Training
   - Topics: `Policies, hardware/software stack, selected high-level APIs` , `Workflow systems` , `best practices, typical mistakes, tools`
   - `Documentation`
   - `Tutorials`
   - `Creation of reasonable user expectations`
   - `Development of I/O components`
   - `User acceptance of training`
5. Research (leads to training)

- Understanding hardware/software behavior
- Tuning, optimizing codes
- Measurement and monitoring including tools → sustaining development

6. Policy making
   - Application for resources
   - QoS
   - Resource assignment (e.g., quota) => static vs. Dynamic

7. Long-term strategic planning of all the above
   - I/O demands, e.g., capacity planning
   - Technology selection / evaluation (hardware/software) / vendor discussions

8. Community support
   - Site users
   - Contributions to standardization (arguably not necessarily a responsibility)

While we only briefly touch on some issues colored green and yellow (slightly improvable), we particularly elaborate on the orange (insufficient) and red (largely absent) focus areas to identify gaps and inherent challenges before making recommendations for them.

4.5.2 Procurement

The procurement of **storage systems** including necessary **network infrastructures** is a core responsibility of data centers. They usually have good relationships with vendors, know the state-of-the-art technologies, and are proficient in specifying the requirements for the hardware components. However, the specification of **benchmarks** for procurements has room for improvement. While the IO500 is a huge benefit for establishing performance expectations, application-specific patterns still are lacking. This situation applies to both standardized I/O benchmarks that mimic typical applications (in particular w.r.t. I/O) and mapping a data center's application mix to a representative set of (standardized) benchmarks.

Monitoring software is often considered as part of procurements. Yet it is mostly basic monitoring in a proprietary manner. It usually does not meet the required level of detail, may be difficult to integrate into an analysis stack, or may miss certain relevant subcomponents of the cluster. When moving from one system to the next, it is often not compatible, specific measurement values are not 1:1 comparable, and the expertise built with the past system is not applicable to the next one. More standardization in well-defined measurements and in software components is desirable.

4.5.3 Operating the Systems

In HPC operations many well-established I/O aspects exist, such as **maintenance**, **enforcements of policies** (esp. quotas, data lifecycle, etc.), **accounting** including I/O resources, and **data security and safety** measures, as well as means for **dedicated resource allocations** for user-specific purposes.

4.5.3.1 I/O Monitoring

Even though most data centers have implemented a global monitoring system, I/O monitoring is often insufficient to help optimize the system as a whole as well as particular workloads.

A particular problem is the amount of data generated by monitoring tools. We identified the need for scalable tools to store and analyze monitoring logs. The analysis results need to be better understandable by both admins and users (e.g., by providing some kind of *I/O Weather* status/forecast). Such analysis should identify actionable items (what to do when), which should be fed back to the users and administrators. While the monitoring of the system and its workflows involves obtaining data from different jobs (user) and I/O servers (system), a holistic view is missing for both a workflow-centric manner (across jobs) and a center-wide manner (I/O resource status). Tool support is lacking for identifying misconfigurations, indicating performance regressions, and predicting failures.

To mitigate these problems, we suggest standardizing the format of monitoring data (including possibilities for compression), establishing a common terminology for I/O monitoring, creating a holistic view of different monitoring sources and dimensions (users and systems), and investigating the potential benefit of AI/ML methods to further the understanding of monitoring data for administrators, developers, and users.

4.5.3.2 Job-Specific Monitoring

Current accounting practices often lack detailed information about the dynamics of data access (frequency, bandwidth, behavior patterns).

In order to address this problem, I/O accesses should always be monitored also on a per-job level, or the global monitoring results should be mapped to jobs, workflows, users, or projects. Excerpts or coarse-grained *footprints* should be incorporated into accounting (albeit the delimination between the terms *accounting* and *job monitoring* remains fuzzy) in terms of key performance indicators such as metadata operation rates, transfer rates, and IOPS. Moreover, tuning potentials with estimated benefits should be indicated.

Furthermore, accounting should "feel" similar across different data centers to avoid user confusion. Reporting metrics thus should be carefully selected, standardized, and unified in terms of terminology while considering their impact on steering I/O performance.

4.5.3.3 Hardware and Software Tuning

Hardware and software tuning is focused on both the system side and the application side. One challenge here is that the system can affect an application's performance and vice versa. In general, the effect of tuning typically cannot be predicted: tuning attempts therefore need to be implemented before their benefit will be known.

Another problem is a lack of clear knowledge about all available tuning options that exist at applications, runtime, and system and hardware levels. Some tuning knobs may require administration rights or even reboots to change BIOS/hardware settings.

Approaches to mitigate these problems include performance modeling and analysis of historic information of the system. Of further importance is the identification of *all* relevant tuning options. This includes static tuning at the center/application levels with optional dynamic tuning at the user/runtime level. When considering tuning options, simulation-based prediction tools can help but often fall short of capturing a sufficiently detailed hardware model.

4.5.4 Supporting the Users

Joint code optimization on a system/infrastructure level and on the application level remains a challenge, mostly because of a lack of experts.

Unilateral **tuning of hardware/software parameters** remains a general challenge, because on a system level they always need to accommodate the average workload. Research into more dynamic on-demand adaptation of system/infrastructure parameters would be desirable.

Improved **feedback from monitoring** would help better aim at the optimization goal. By improved we mean a holistic consideration of all resources, with more detail and enriched with hints that indicate potential issues (first guesses what the issue is and how this can be improved). On the one hand, this would improve job-specific optimization for a single application or all jobs of a particular kind. On the other hand, a notion of the current *I/O Weather* would be helpful, namely, general information about the state or the stress of the shared I/O subsystem. This in turn would be a key criterion for reasonable expectations of how high the I/O performance of a given application could become. It could also be used to dynamically activate or deactivate certain I/O optimizations in an application.

4.5.4.1 Understanding What Users Are Doing vs. What They Really Need

Some users have limited understanding of their application's actual I/O behavior and their needs. They act only if their problem turns out to be a grave performance problem to them or others. These users accept suboptimal performance and ignore the potential for improvement. Also, it is difficult to assess which applications have good vs. bad performance, or even what performance to expect.

Some experimental tools uncovering problematic application behavior are available and should be used more by data center experts. However, they can "only" uncover the problematic effects to the shared I/O system; and, since we will likely never have a full formal specification of I/O behavior, they will remain intrinsically limited. Thus, structured discussions with users and developers is needed, since domain-specific knowledge is required to fully understand I/O needs. Both approaches, the use of tools and structured dialogues with users and developers, require dedicated HPC experts for consultation and support and active help in application development and usage.

This issue is even more acute for data workflows (multiple jobs or steps with multiple applications as part of a processing chain) or data lifecycle decisions that may even span multiple sites.

4.5.5 Training

Training on I/O, particularly advanced training, is currently not sufficiently offered in the various training programs. Limited training exists for proper data lifecycle and workflow handling. Another problem is user acceptance of training, which is also lacking to some extent.

We suggest establishing coordinated training programs with a consensus on what should be taught. We also suggest creating a simple I/O performance model (I/O roofline model) to train best- and worst-case practices by listing pitfalls in application, domains, and I/O patterns. Furthermore, training material should be shared. Training should be offered for multiple levels of expertise (fine-grained). Also, better documentation on the topics and common knowledge (best practices, patterns) should be available and communicated.

We could solve the problems by devoting more resources to develop training. One could submit a paper to a workshop about the state of training practice or could join forces with an existing training program such as VI-HPS [1] and the HPC Certification Forum (HPC CF)[2].

---

[2] `https://www.hpc-certification.org/`

4.5.6 Research

A number of I/O research directions exist as part of HPC research. Some are more focused on I/O; some cover I/O as one aspect among others. The research has been roughly classified into **understanding hardware/software behavior**, **tuning and code optimization**, and **measurement and monitoring, including tools**.

4.5.6.1 Sustaining Development of Monitoring Systems and Tools

Many valuable tools and software components have been created by the research community. Yet follow-up activities for the projects, tools, and libraries are problematic in terms of sustained funding. Certainly, this should not become the default case for all research projects. Yet few I/O tools and libraries exist, and many fewer with an I/O focus subject to some level of production-level code quality and sustained development. Darshan and SIONlib are two of the notable exceptions.

Reasons for this situation are more or less all connected to sustained funding. While commercialization may be an option for some, in general open-source community-driven development and support models might be more suitable for the existing community. Follow-up research funding will certainly help as long as additional research questions remain. However, research funding by government or academic funding sources is looking at different goals and cannot be a long-term answer here.

Instead, the academic, government, and industry players in the HPC community should address this situation. Data centers should play a role and try to secure funding for some tools, software components, or projects that are in everybody's interest to be sustained. They should argue to their funding agencies that this is a fair return of effort to selected projects compared with the direct benefit the center receives from the much larger amount of open-source and community software used. This should go beyond the current opportunities. For example, in the United States, Small Business Innovation Research and Small Business Technology Transfer programs [2] exist within NSF and DOE, and also other NSF programs especially from the CISE Office of Advanced Cyberinfrastructure and similar programs in other countries.

4.5.7 Policy Making

Policy making covers the **application for resources**, **quality of service**, and resource assignment. Existing policies focus mainly on restricting user capacity and number of files (quota mechanisms) and some purging policies of scratch file systems. Isolating users and applications that negatively influence the storage performance of the shared storage can be improved on. Defining appropriate policies and having mechanisms to do so would be an evolutionary step from the current situation. File systems and networking need to be improved to better support this option. Similarly, when users apply for resources, the data centers should be more rigid in requesting information about the I/O usage and workflows. Some data centers request more details, but others limit themselves to the storage capacity needed. The resource assignment of projects is mostly statically assigned; in other words, with a project proposal one receives an allocation for the project campaign. In practice, however, the requirements change over the lifetime of a project. Therefore, a more **dynamic resource assignment** would be useful, but this is not yet available or implemented by the centers.

4.5.8 Long-Term Strategic Planning

Long-term strategic planning is sought by all data centers, and some strategies that have been been developed have been successfully applied. Improvements are still needed, however, particularly in better planning of the I/O storage landscape, selecting and evaluating storage systems, and engaging with vendors. All these points require a better understanding of storage systems and the implications specific design decisions would have.

4.5.9 Community Support

Community support could be improved for data center users, for example, transforming the vertical communication between users and the center to a more vibrant one where users communicate directly with one another (horizontally) and share best practices. This goal requires platforms for discussion and information exchange. Arguably, the contribution to standardization bodies and technical solutions are not a key responsibility of a data center per se; nevertheless, providing service to users and operating the systems effectively clearly support the data center mission.

Additionally, synchronization of training and documentation is desirable, for example, establishing common terminology and training materials. Particularly useful would be developing training materials that can be used across data centers with little adaptation for site specifics. Another form of knowledge sharing is the data repository, which collects and manages datasets for analysis and sharing. An excellent example is the Darshan log repository provided by Argonne National Laboratory [3].

The community aspects are discussed in more detail in the next chapter.

References

1 https://www.vi-hps.org/
2 https://www.sbir.gov/
3 https://www.mcs.anl.gov/research/projects/darshan/download/

4.6 Deep Dive Topic: Community Support

Wolfgang Frings (Jülich Supercomputing Centre, Germany, w.frings@fz-juelich.de),
Yi Ju (Max Planck Computing and Data Facility, Germany, yi.ju@mpcdf.mpg.de),
Sarah Neuwirth (Goethe-University Frankfurt, Germany, s.neuwirth@em.uni-frankfurt.de),
*Sebastian Oeste (TU Dresden, Germany, sebastian.oeste@tu-dresden.de),*
Martin Schulz (TU Munich, Germany, martin.w.j.schulz@tum.de)

While many community aspects identified in this seminar match the general challenges faced throughout the entire HPC community (or even beyond), the I/O community faces several specific issues in addition. The cause is rooted in the special characteristics found in the I/O community, which are listed in the following subsection. The rest of the section then focuses on consequences from these special characteristics and how to merge them with other, general community problems.
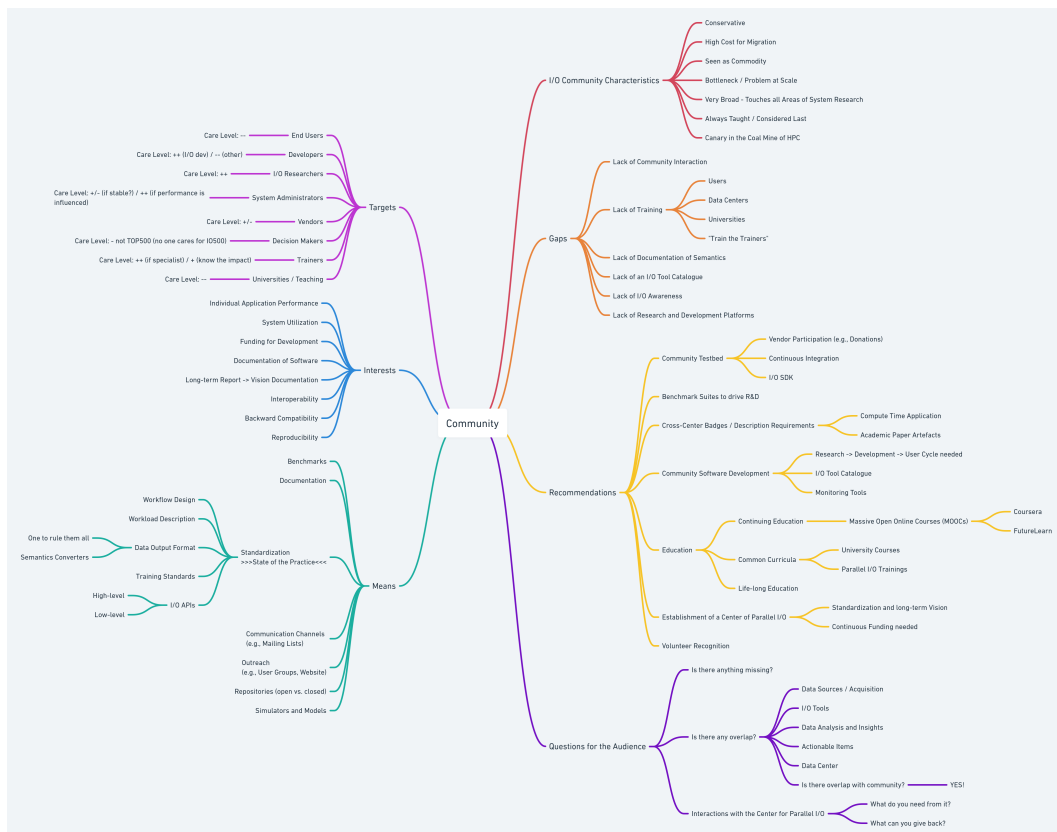
■ **Figure 4** Overview of the community-related topics.

4.6.1   Special Characteristics in the I/O Community

I/O systems have to be conservative because they affect data that (a) has to be stored long term, (b) is relied on to be available at any time, and (c) is hard to migrate to new systems. At the same time, I/O systems are seen as a commodity that is a given. This situation leads to little incentive to make major changes even if they would result in better performance, because of the cost and effort required to make any changes, validate them, and guarantee stable and reliable operation.

Additionally, I/O work spans the entire system (storage, networks, OS, security, APIs, etc.) and is therefore highly interdisciplinary. Consequently, the entrance barrier for I/O work is high, because of both the needed knowledge and the needed testbed systems (which often have to be large scale to show impact). Further, I/O cannot be seen as isolated—or should not be seem as isolated—despite the fact that many research projects in the area focus on individual elements (e.g., one file system) and hence have a hard time showing impact or usability.

4.6.2   Overview

Figure 4 shows a summary of the discussions, split into several subcategories. These will be detailed in the remainder of this section.

4.6.3 Targets

I/O community topics affect a large number of target groups, which are listed below. Each of these groups has a different level of interest in I/O, or label "Care Level," which ranges from "++" (cares a lot) to "−" (doesn't care).

- End users: the application users running an application on an HPC system, often deploying existing codes or just minimally configuring/extending it **Care Level for I/O: −**

- Developers: developers who create both applications (used by end users) and libraries and runtime systems. The latter also contains software systems that implement I/O and/or use I/O in significant amounts. **Care Level for I/O: ++ (for I/O developer) and − (for any other developer)**

- I/O researchers: researchers in computer science (or related fields) that directly develop novel solutions to improve the I/O of HPC systems (e.g., the participants in this seminar). **Care Level for I/O: ++**

- System administrators: administrators of HPC systems responsible for safe, secure, and reliable operation of HPC systems. **Care Level for I/O: +/- (for general admins) and ++ (for storage admins)**

- Vendors: HPC system vendors selling HPC solutions to HPC centers, which often include I/O subsystems (or subcontractors specializing in I/O systems) **Care Level for I/O: ++ (for specialized I/O vendors) and +/- (for others)**

- Decision makers: the leads for public and private procurements who write RoIs and RFPs and evaluate them. **Care Level for I/O: − (main care is Top500)**

- Trainers: people running training or tutorial sessions at HPC centers and/or conferences with the target of current and future HPC users. **Care Level for I/O: ++ (for I/O trainers) and + (for general HPC trainers)**

- Teachers: those who teach I/O as part of bachelor's or master's curriculum at the university level. **Care Level for I/O: −**

4.6.4 Interests

- Performance of individual application
  from end-user perspective: improve I/O performance and reduce execution time; from support: identify application with I/O performance bottlenecks.
- System utilization
  increase the system utilization by reducing I/O waiting times; monitor overall I/O activity.
- Funding for development
  provide continuous funding of software development for I/O monitoring and analysis, essential for keeping software alive; and push R&D.
- Documentation of software
  provide good documentation of tools for support members and software developers; especially for end users, provide best practice guides that include also the usage workflow of the tools in different environments.
- Long-term report and vision document
  provide regular updated vision documents to developers, decision makers, and others, including hints about trends of development. For example, for tool developers it would be helpful to get early information about storage system development to react directly and to adapt their development strategies. Such vision documents can help end users as they adapt their I/O strategy to future trends. Long-term reports also can be seen as input for standardization efforts.

━ Interoperability
provide multiple tools for I/O monitoring exists with overlapping functionality. However, user and support people use different tools for their tasks. Interoperability of these tools would help these people easily change tools when functions are missing in one tool (see also data formats).

━ Backwards compatibility
monitor the development of the different I/O tools, and consider the backward compability of these tools. An individual application may not follow development cycles at the same speed and may require tools that can handle old data. Backward compatibility is also required for I/O monitoring tools that use old data sources (see also interoperability).

━ Reproducibility
ensure that single-application I/O measurements are reproducible (e.g., for papers). This requires that changes in tools and environment be well documented and that, for major changes, measurements be recalibrated.

4.6.5   Means

━ Benchmarks
A suite of I/O benchmarks provides a defined set of I/O metrics that can be used on different storage system and different software layers. IO500, for example, defines benchmark configurations for IOR and mdtest, collects the results in a web-accessible repository, and provides an overview of I/O capability of different storage systems.

━ Documentation
Documentation should include different perspectives of the community. It can provide an overview of the community and possible connections that the participants can expect from the community. Moreover, systematic documentation can cover technical aspects, for instance, benchmarks, repositories, simulations, and standardization.

━ Standardization
In the I/O field, some standard or common definition is still missing. The community should agree on the primary concepts in the I/O field.

  ━ Workflows design

  ━ Workload description

  ━ Data output format (one rule for them all, semantic converter)
  Already a group of different performance tools can be used for I/O performance analysis. The same parameter can be named in different ways in individual tools. The community can organize discussions with researchers and tool developers to agree on reasonable data output formats or identify one generic semantic converter allowing global understanding of the results from individual tools.

  ━ Training standards
  With proper training, end users can improve the I/O performance of their application. In some rare cases, however, the training is organized in an unattractive or even inefficient way, resulting in the training being held in low regard and fewer people seeking it. The training material and techniques should be standardized at a high level in order to guarantee the quality of the training.

  ━ I/O APIs (high Level, low level)
  The community should define the standards for both high-level and low-level I/O APIs. There is heated discussion about whether a higher-level or lower-level API should be considered in the optimization of the POSIX performance. With optimization of POSIX using low-level APIs, the POSIX APIs are kept, and therefore the applications built

on POSIX need no update, but standard low-level APIs are still necessary. When the optimization of POSIX is built on high-level APIs, an enormous number of applications need to be updated according to the new standard, which should be defined by the community.

- Communication channels (e.g., mailing Lists)
  Users should be able to contact researchers sharing the same interest. For instance, I/O performance analysis tools should be available if related research needs this tool but technical problems exist.
- Outreach (e.g., user groups, website)
  In order to reach specific community targets, such as user groups, a website with well-organized structure and sufficient information should be provided and maintained.
- Repositories (open vs. closed)
  The I/O community should host a repository to allow the members to find useful material. Open repositories will be the practical method to provide the information and source code. But in reality, some innovative tools or ideas are developed by Ph.D. students, who need to use them for personal publication. In this situation, protected or closed repositories should be the choice for them to receive feedback from the community while having their data secured. Another beneficial functionality can the "issue" system, whereby researchers or users can report issues to the developers.
- Simulators and models
  One of the essential aims of the I/O community should be to encourage innovation in the I/O research field, such as I/O performance analyzers. In the early stage of the development of the tools, the simulators based on reliable I/O models should be the foundation of the testbed.

4.6.6 State-of-the-Art Resources

4.6.6.1 Benchmarks and Tool Catalog

- IO500, https://www.vi4io.org/io500
- I/O tools and benchmarks: https://www.vi4io.org/tools/start
- https://www.vi-hps.org/tools/tools.html

4.6.6.2 Training events

- XSEDE user training: https://www.xsede.org/for-users/training
- PRACE training portal https://training.prace-ri.eu/
- GCS: https://www.gauss-centre.eu/trainingsworkshops/
- NHR training events
- VI-HPS: https://www.vi-hps.org/training/index.html

4.6.7 Gaps

- Lack of community interaction
  Little interaction currently occurs among I/O researchers, tool developers, and file system administrators outside of research projects or within a single center. Users might get confused by getting different recommendations, for example, for tools that they should use to understand I/O behavior. A platform is needed for exchanging knowledge and developing common strategies to guide users. A recurring exchange between I/O researchers, tool developers, and file system administrators is necessary in order to develop and maintain such strategies and get the most viable outcome. Furthermore, users should have a common place to asking questions regarding their I/O problems.

- Lack of training
  With a steady exchange between users and I/O experts, an identification of common problems should lead to better training resources. At the moment training resources are limited. Specialized offers by single projects or sites exist, but they are often coupled with certain events or are just a one-shot activity. The community lacks recurring general-purpose training needed to understand parallel I/O behavior.
- Lack of semantics documentation
  While an overlap exists in the functionality and reporting capabilities of I/O tools, the output data formats are often different and specific. The lack of documentation on the semantics of data formats makes direct comparison of the results of different tools difficult.
- Lack of an I/O tool catalog
  The community should provide a full catalog of available tools for understanding I/O behavior. The catalog should list the tools, state their main purpose (what the tool is good for, what the motivation behind its development is), provide some contact to the maintainer, and identify the current state of the tool (if it is a production tool or research prototype, or if it is no longer maintained).
- Lack of I/O awareness
  Users and nonexperts are often not aware of the implications coming with parallel I/O at scale because these issues do come not come up during the development on local machines.
- Lack of research and development platforms
  Scientific software projects commonly are hosted on site or project internal platforms (e.g., site internal GitLab instances). This situation could hinder collaboration between the I/O research community and, for example, tool developers. A valuable effort might be to provide an open site-independent platform to host source code repositories and issue tracking and documentation to improve collaboration and visibility of the different projects.

4.6.8 Recommendations

To bridge the gaps in the I/O community construction, we propose the following recommendations.

- **Community testbed** should be built to enable testing of existing I/O performance analysis tools and to provide an opportunity for testing during updating and development of new tools. Vendors should be welcome to participate in the community testbed. The integration of the testbed should be continuous. Ideal output format could be I/O SDK.
- **Benchmark suites to drive research and development** should be regularly maintained. When an old benchmark no longer matches the real use case, it should be removed from the suites. New benchmarks should be allowed to be added after comparison with the old ones in the suites. Some new benchmarks might be totally different from the old ones; but when they can give the I/O researcher new ideas or reflect a new tendency, they should be added to the suites.
- **Cross-center badges or description requirements** such as compute time application or academic paper artifacts should motivate participation in the community and contribute to the community.
- **Community software development** should be based on the "researcher, developer, user" cycle. I/O tool catalog and monitoring tools should be summarized by the community to accelerate the development of the software.
- **Education** should cover three types: (1) continuing education, through massive open online courses (MOOCs) such as Coursera or FutureLearn, which could attract more fresh blood to focus on parallel I/O during their academic life; (2) for university students,

common curricula such as courses and parallel I/O training to make them aware of the importance of I/O; and (3) life-long education, which could help developers improve the performance of their applications.

- **Establishment of a center of parallel I/O** should produce the standardization and long-term vision of parallel I/O and attract continuous funding, for example, for well-structured Dagstuhl seminars.
- **Volunteer recognition** in addition to the network should attract young researchers to join the community and choose parallel I/O as their further research field.

5 Recommendations and Conclusions

Several recurring themes emerged over the course of Dagstuhl Seminar 21332, *"Understanding I/O behavior in scientific and data-intensive computing"*. These high-level themes represent areas in which further research is expected to yield the largest impact across the field as a whole.

- **Tools and techniques are needed that span the full hierarchical scope HPC I/O behavior**: data centers, workflows, applications, processes, system software, and hardware. The current state of the art has produced silos of information that make it difficult to correlate low-level characteristics with high-level trends and link root causes to their broader impact.
- **Interoperability, and more specifically the ability to translate findings and techniques across the field at large, is hindered by lack of commonality and standardization.** This phenomenon is evident at multiple levels: terminology, data formats, monitoring granularity, and availability of infrastructure for storage and analysis of characterization data.
- **Lack of insight into the performance impact of monitoring tools, along with a lack of mechanisms to adjust that impact, has limited the deployment of performance monitoring infrastructure.** Large storage systems are high-value shared resources, and administrators will err on the side of caution if potential performance degradation or reliability impacts are not well understand.
- **Workloads used for evaluation and investigation of storage behavior are not representative of production applications.** They fail to capture salient characteristics of relevant applications, fail to capture novel workloads such as AI, or lack sufficient breadth to reproduce emergent system workload properties.
- **The ultimate impact of I/O behavior analysis is gated by our ability to transfer findings to end users and facility operators.** Current state-of-the-art tools cater primarily to researchers and systems experts. The gap between the two calls for better training, greater communication, more incentives to enact changes, clear risk/reward assessments, and simpler visual representations such as roofline models.
- **The diversity of platforms, workflow systems, high-level libraries, and applications makes it difficult to apply corrective action in a consistent way.** Many inconsistent tuning options are available to users and administrators. The field would benefit from unified models for data movement and more ways to reason about I/O properties that transcend the idiosyncrasies of individual scientific problem domains.
- **Advances in understanding HPC I/O behavior are difficult to sustain over time in production environments.** Root causes include rapid innovation in storage technology (which does not typically consider I/O instrumentation as a first-class citizen), difficulty in securing funding for software maintenance, lack of R&D platforms, challenges in workforce cultivation, and insufficiently robust system models.

State of the Field and and Future Outlook

Understanding HPC I/O behavior is crucial not just for today's systems; it is becoming even more important as scientific computing becomes more data centric and reliant on more diverse data sources. Comprehensive understanding of I/O behavior is the cornerstone technology that underpins any effort to optimize data access within applications and systems. Without it, we risk wasting valuable resources in research, procurement, and optimization of HPC systems and applications.

Extensive ongoing literature contributions continue to demonstrate the impact of advances in understanding HPC I/O behavior. In this context, the unique value of Dagstuhl Seminar 21332 was to produce a comprehensive, holistic view of the field by synthesizing the perspectives of a diverse collection of subject matter experts. This holistic viewpoint made it clear that our already impressive ability to understand and interpret I/O behavior would be even more impactful if we could address the most common shared roadblocks that have emerged across HPC facilities and application development teams. The community as a whole is well positioned to undertake research that addresses these challenges.

6   Abstracts of Lightning Talks

6.1   Analyzing POSIX I/O semantics of parallel applications

Sebastian Oeste (Technische Universität Dresden, DE)

POSIX I/O and its restrictive access semantics are an already known bottleneck for the performance of distributed network file systems. Some parallel file systems relaxing specific POSIX semantics to provide better performance. While there are existing tools to test the POSIX compliance of the file system there is no tool to test the POSIX requirements of the application. In this talk I want to discuss a methodology for a tool to analyze the POSIX I/O requirements of parallel applications.

6.2   Andreas Knüpfer: I/O Aspects in the Center-Wide and Job-Aware Cluster Monitoring System PIKA

Andreas Knüpfer (Technische Universität Dresden, DE)

The lightning talk presented the basics about the center-wide, continuous cluster performance monitoring system "PIKA" which is in production at TU Dresden for over three years. It collects a set of performance metrics on all nodes in granularity of 1-2 samples per minute and maps them to batch system jobs. This allows a performance overview for live and past HPC jobs for users (for their jobs) and the computing center (for all jobs by all users).

PIKA contains also I/O metrics and the lightning talk focused on those metrics, how they can be visualized in various forms (timelines, histograms, scatter plots against other metrics, footprint summaries) and how one can search/filter jobs according to metrics.

Finally, the talk argued that I/O metrics are special and more difficult for automated judgement into sufficiently fast or too slow. Reasons for this are among others that (i) maximizing the I/O rates towards the nominal capacity of the infrastructure all the time is not desirable – unlike for CPU utilization or FLOP rates, (ii) peaks and gaps in I/O are expected, and (iii) even heavy I/O phases may be averaged out over the entire life span of long jobs. This was the starting point for a discussion how to assess PIKA's continuous I/O metrics in a more appropriate way.

Further details are given in [1] and the software is available under an Open Source license at [2].

References

1   Robert Dietrich, Frank Winkler, Andreas Knüpfer, Wolfgang E. Nagel; *PIKA: Center-Wide and Job-Aware Cluster Monitoring*, In proc. of 2020 IEEE International Conference on Cluster Computing (CLUSTER), Kobe, Japan, Sept. 2020, pp. 424-432, DOI 10.1109/CLUSTER49012.2020.00061.

2   GitLab repositpory of the PIKA project at `https://gitlab.hrz.tu-chemnitz.de/pika`

6.3   Can We Gamify I/O Performance?

Philip Carns (Argonne National Laboratory, Lemont IL, USA)

There is a fundamental manpower scalability problem in the arena of HPC I/O performance tuning: the number of scientific users is becoming larger and larger, while the availability of facility I/O experts is not. This talk explores the challenges and potential for better engaging the more "scalable" former group by *gamifying* I/O performance tuning. This will require methods for identifying competitors, a scoring system, competition rules, and incentives. None of these things are immediately straightforward to create, but the potential payoff for doing so is a way to not only improve system efficiency, but also to more deeply engage users in the process and develop novel technologies for reasoning about I/O strategies and their relative merits.

6.4   Lifting the user I/O abstraction to workflow level – a possibility or in vain?

Julian Kunkel (Universität Göttingen / GWDG – Göttingen, DE

This talk revisits the current workflow specifications. Mostly these are implicitly defined by task dependencies and hide the I/O characteristics. It then discusses the potential to exploit user workload specifications on a higher level. Lastly, the community could jointly work on such higher-level specifications.

Further details for the climate/weather community are given in [1].

References

1   Julian Kunkel, Luciana Pedro; *Potential of I/O Aware Workflows in Climate and Weather*, In Supercomputing Frontiers and Innovations, Series: Volume 7, Number 2, pp. 35-53, (Editors: Jack Dongarra, Vladimir Voevodin), Publishing Center of South Ural State University (454080, Lenin prospekt, 76, Chelyabinsk, Russia), ISSN: 2313-8734, 2020-04, DOI 10.14529/jsfi200203

6.5   An IO500-based Workflow For User-centric I/O Performance Management

Radita Liem (RWTH Aachen University, DE)

I/O performance in a multi-user environment is challenging to predict. It is hard for users to know what to expect when running and tuning their application for better I/O performance. In this project, we evaluate IO500 as a user-centric workflow to manage their expectations on their application's I/O performance and devise an optimization strategy specific to the target cluster's capability.

IO500 benchmark is a standard benchmark for HPC storages systems and is designed to create a balanced performance. In our workflow, we use the IO500 benchmark scenarios "easy" and "hard" to get the best and worst possible performance results of the cluster's bandwidth and metadata rate. Then, we create a bounding box of user expectations with these scenarios and map the application's I/O performance within this box. With the mapped I/O performance, we can understand which part of the application needs to be improved and the possible extent of this improvement.

Our experiments confirm that the bounding box of user's expectations can be created. In doing so, this project is a promising first step towards the mapping and improvement of the application's I/O performance.

Details are given in [1].

References

1   Dmytro Povaliaiev, Radita Liem, Jay Lofstead, Christian Terboven. An IO500-based Workflow For User-centric I/O Performance Management. Poster presented at: ISC High Performance 2021; June 24 – July 2, 2021.

6.6   IOMiner – A multi-level analysis to detect root causes of I/O bottlenecks

Suren Byna (Lawrence Berkeley National Laboratory, Berkeley, USA

Joint work of Suren Byna, Teng Wang, Suren Byna, Glenn Lockwood, Philip Carns, Shane Snyder, Sunggon Kim, Nicholas Wright

To understand the root causes of poor performing I/O jobs, we have conducted a zoom-in analysis and developed a set of analyses that were put together as a tool called IOMiner. In this presentation, we present an analysis of platform, application, and job level IO

instrumentation logs using parallel coordinates and sweep-line analysis. We will also describe issues that caused performance bottlenecks in a few I/O use cases and solutions that resolved them.

Further details of IOMiner are described in [1] and [2], and the software is available with an Open Source license at `https://github.com/hpc-io/IOMiner` [3].

References

1   Teng Wang, Suren Byna, Glenn Lockwood, Nicholas Wright, Philip Carns, Shane Snyder, *IOMiner: Large-scale Analytics Framework for Gaining Knowledge from I/O Logs.* IEEE Cluster 2018.

2   Teng Wang, Suren Byna, Glenn Lockwood, Philip Carns, Shane Snyder, Sunggon Kim, Nicholas Wright, *A Zoom-in Analysis of I/O Logs to Detect Root Causes of I/O Performance Bottlenecks*, IEEE/ACM CCGrid 2019.

3   GitLab repositpory of the IOMiner tool at `https://github.com/hpc-io/IOMiner`

Participants

Wolfgang Frings
Jülich Supercomputing
Centre, DE

Yi Ju
Max Planck Computing and
Data Facility – Garching, DE

Andreas Knüpfer
TU Dresden, DE

Julian Kunkel
Gesellschaft für wissenschaftliche
Datenverarbeitung –
Göttingen, DE

Erwin Laure
Max Planck Computing and
Data Facility – Garching, DE

Radita Liem
RWTH Aachen University, DE

Frank Mueller
North Carolina State University –
Raleigh, USA

Sarah Neuwirth
Goethe-Universität Frankfurt am
Main, DE

Sebastian Oeste
TU Dresden, DE

Martin Schulz
TU München, DE

Remote Participants

Marcus Vincent Boden
Gesellschaft für wissenschaftliche
Datenverarbeitung –
Göttingen, DE

Jim Brandt
Sandia National Laboratories –
Albuquerque, USA

André Brinkmann
Universität Mainz, DE

Suren Byna
Lawrence Berkeley National
Laboratory, Berkeley, USA

Philip Carns
Argonne National
Laboratory, USA

Fahim Tahmid Chowdhury
Florida State University –
Tallahassee, USA

Hariharan Devarajan
Lawrence Livermore National
Laboratory, USA

Ann Gentile
Sandia National Laboratories –
Albuquerque, USA

Sivalingam Karthee
Huawei Technologies –
Reading, GB

Roland Laifer
KIT – Karlsruhe Institute of
Technology, DE

Jay Lofstead
Sandia National Laboratories –
Albuquerque, USA

Johann Lombardi
Intel Corporation – Meudon, FR

Stefano Markidis
KTH Royal Institute of
Technology – Stockholm, SE

Sandra Adriana Mendez
Barcelona Supercomputing
Center, ES

Kathryn Mohror
Lawrence Livermore National
Laboratory, USA

Michael Ott
LRZ – München, DE

Marc Snir
University of Illinois at
Urbana-Champaign, USA

Shane Snyder
Argonne National
Laboratory, USA

- Mehmet Soysal
  KIT – Karlsruhe Institute of
  Technology, DE

- Osamu Tatebe
  University of Tsukuba, JP

- Devesh Tiwari
  Northeastern University –
  Boston, USA

- Chen Wang
  University of Illinois at
  Urbana-Champaign, USA

- Michèle Weiland
  EPCC, The University of
  Edinburgh, GB

- Weikuan Yu
  Florida State University –
  Tallahassee, USA