The Twilight of I/O as a User Concept

Jerome Soumagne[1], Gerd Heber[1], Andres Marquez[2], Elena Pourmal[1]

Topics:

- Interfaces for accessing data
- Understanding the overlap between traditional storage systems and I/O (SSIO) efforts and data management

Challenge

I/O middleware libraries (e.g., HDF5, ADIOS, PnetCDF, MPI I/O), commonly used on production systems for storing scientific data, have all been originally designed when parallel file systems and disk-based storage were prevalent. Dressed up in different flavors of higher-level models of abstraction (e.g., hierarchical, object-based model in HDF5), they have nevertheless perpetuated I/O as a user concept and concern. It is an unfinished job, because it creates an obvious dilemma: users have to match the application's data models to middleware primitives and worry about how these choices affect I/O performance. What is worse is that as these middlewares are ported to new storage types, the performance characteristics of middleware primitives may shift, creating the perfect nightmare for users [Xie2021]. This issue has only been exacerbated with the increasing amounts of data generated due to both an increase in problem complexity, in processing power, and new system architectures. Applications have therefore been forced to restructure and finely tune their I/O models so that they could get the most from the I/O bandwidth offered by parallel file systems [Wan2022]. This has been often at the detriment of slower reads, as post-processing algorithms generally no longer match with the data format that was output to disk, commonly forcing extra post-processing steps to be taken for data analysis to be realized efficiently and in a timely manner. As a consequence, the increasing need for more complex workflows (e.g., AI, multi-physics workflows) has drastically encouraged application users to find new solutions (e.g., in-situ, in-transit analysis) that bypass file systems in an effort to reduce the cost of analysis and I/O optimization efforts. To further reduce the cost of I/O, solutions such as asynchronous I/O are slowly taking an important role, but they require an important commitment from the application developer in order to be used properly, potentially making the use of I/O middleware even more complex. In that context, there is a challenge for I/O middleware libraries to provide software that no longer requires application's I/O to be finely tuned based on the storage system architecture but remains in tune with the application needs and data models.

Opportunity

With the emergence and preponderance of new technologies such as persistent memory and object stores, the limitations that used to be imposed by disk-based storage are now replaced with new opportunities for defining file formats and storing data—this has been seen recently with solutions such as Intel's DAOS [Soumagne2022], but also more broadly with Cloud solutions (e.g., Amazon S3, Ceph RADOS). While those solutions can be brought into existing I/O middleware solutions, application developers and users may only be able to fully grasp their benefits by rethinking once more the way they are doing I/O in order to reach the desired performance. In other words, those new capabilities, while beneficial, are not sufficient in themselves to directly fit to the needs of applications and the interfaces

[1] The HDF Group, [jsoumagne,epourmal,gheber]@hdfgroup.org
[2] Pacific Northwest National Laboratory, andres.marquez@pnnl.gov

exposed by I/O middleware. They must also evolve so that they can be fitted to the application data models as they were originally thought of. The evolution from disk and block-based storage to object stores now opens an opportunity for I/O middleware to rethink how interfaces should be presented to applications. The traditional write and read semantics may be re-thought to instead focus on the description of the I/O data model to prevent tedious and constant optimization efforts from the application developer. There is therefore an effort that must take place between application and middleware developers to design and evolve the current I/O interfaces.
Furthermore, along with those new technologies, new types of workflows and I/O needs have emerged, which are gradually being expressed through custom data services [Ross2020], designed to address a response to a particular I/O need (e.g., data staging, monitoring, multi-tiered storage abstraction, etc). Those data services, while becoming essential, also introduce another degree of complexity for application users, who may need to design custom I/O pipelines. In that sense, it also becomes the responsibility of I/O middleware to rethink the way data is accessed in order to facilitate that process and take advantage of those new types of I/O methods more efficiently. This may be done along with rethinking the type of semantics that are provided to applications so that they remain tied to data models. A more radical proposal would be to eliminate I/O as a user concept altogether. With persistent memory overcoming the chasm between random access and block I/O, there is now an opportunity for providing users with convenient primitives to state “persistence intents” and let a virtualized I/O layer be the new frontier for middleware developers (e.g., through I/O compilers, data services, etc).

Timeliness and Impact

The storage and HPC community has now reached a turning point where the traditional storage system is no longer the only means for accessing data. With the emergence of object stores and data services, there is a need to evolve I/O middleware interfaces that were designed at a time where monolithic block-based parallel file systems were the norm. The increased complexity of workflows and storage architectures require the rethinking of interfaces so that they no longer interfere with the applications’ data model and no longer require extraordinary engineering efforts of optimization.

Removing that burden from application developers has an extremely high impact and has the potential for saving precious development time by focusing more on science aspects and less on the engineering efforts. By removing complexity from the application, it has also the potential to open to new opportunities and types of workflows that were until now discouraged.

References

- [Xie2021] B. Xie *et al*., "Battle of the Defaults: Extracting Performance Characteristics of HDF5 under Production Load," *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2021, pp. 51-60, doi: 10.1109/CCGrid51090.2021.00015.
- [Soumagne2022] J. Soumagne *et al*., "Accelerating HDF5 I/O for Exascale Using DAOS," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 903-914, 1 April 2022, doi: 10.1109/TPDS.2021.3097884.
- [Wan2022] L. Wan *et al*., "Improving I/O Performance for Exascale Applications Through Online Data Layout Reorganization," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 4, pp. 878-890, 1 April 2022, doi: 10.1109/TPDS.2021.3100784.
- [Ross2020] R.B. Ross *et al.* Mochi: Composing Data Services for High-Performance Computing Environments. *J. Comput. Sci. Technol.* **35,** 121–144 (2020). https://doi.org/10.1007/s11390-020-9802-0