Harnessing Hierarchy

Gerd Heber[1], Chris Hogan[1], Anthony Kougkas[2]

Topics:

- Storage-system architecture design that supports scientific workflows on varied hierarchical storage and networking devices
- Utilizing AI to learn I/O patterns of emerging workloads for efficient data management

Challenge

To achieve good performance, today's users of HPC systems are being asked to know a lot about the system, its hardware and software composition, and a long list of dos and don'ts. And, of course, this changes from system to system. From an I/O perspective, perhaps only RAM and parallel file systems (PFS) have been present with some predictability. Whenever "shiny" go-between hardware showed up, the job of making changes fell to application developers. Perhaps the main reason for this was the lack of support for these "outliers" in the underlying middleware.

In principle there are four categories of escape routes:

1. Stop interspersing new hardware
2. Make the existing middleware software hierarchy-aware
3. Create a uniform (in appearance!) hierarchical aggregate
4. Create a new middleware capable of handling the complexity of modern hierarchical storage

Option 1. is perhaps only theoretical. Option 2. seem appealing because it would shield applications from change. However, we believe that the success of retrofitting middleware layers might be limited due to complexity, the cross-cutting nature of hierarchy-awareness, and application changes being necessary in the end to achieve performance gains that would make the whole effort seem worthwhile. Option 3. would achieve the same (shielding applications from change), but it is likely that, in practice, its non-uniform access characteristics will bleed back into applications without any ability to control it, aside from application code changes. Options 1-3. also share a common problem, which is that they implicitly perpetuate an abstraction, the notion of filesystems and files, which is perhaps not the direction that high-performance storage or cloud storage are headed, at the moment.

The challenge is to design something that achieves the following:

a. It is hierarchy-aware.
b. It is transparent to applications.
c. It plays well with existing middleware.
d. It is neutral with respect to I/O abstractions.
e. It runs in user space without the need for elevated privileges.

Item a. means that it shows demonstrably better performance by utilizing hierarchies over the PFS baseline. Item e. is not strictly necessary, but a practical consideration.

Opportunity

We believe that there is an opportunity to create a new middleware along the lines of option 4. by taking advantage of the full spectrum of new storage (e.g., PMEM, NVMe) and network hardware (e.g., IB RDMA, RoCE, NVMeoF), and by targeting new storage interfaces such as object stores. The new middleware

1 The HDF Group, gheber@hadfgroup.org, chogan@hdfgroup.org

2 Illinois Tech, akougkas@iit.edu

should not attempt to appear as a unified storage layer which exposes a certain storage abstraction. [4] That would be a mere return to option 3. Instead, it would appear as a distributed buffering layer with the following characteristics:

- A set of adapters for popular middleware (e.g., UNIX STDIO, POSIX, MPI-IO, HDF5) would extend existing applications w/o the need for code changes and make them hierarchy-aware.
- Buffering behavior at different levels (global default, per-call, etc.) would be governed by policies, which would shape the way data is distributed throughout the hierarchy and across nodes [1,2]. There would be default policies for common workloads, but also support for user-defined policies. This would also be a natural place for utilizing AI to learn I/O patterns [3] of emerging workloads for efficient data placement, routing, compression, etc.
- Buffering resources could be discovered and used dynamically, but also configured statically for well-understood scenarios and predictable resource utilization for buffering.
- This buffering middleware would appear as a distributed application extension. In order to support cooperating applications (e.g., “data hand-over”) a user space daemon would take custody of in-transition data or to support publisher/subscriber patterns.
- Finally, occupying a fairly central position at the middleware crossroads, we believe that there are interesting codesign opportunities, for example, with Mochi-style data services.

Timeliness and Impact

The deployment of a new breed of storage devices and ever more capable fabric in multiple DOE sites and the breathtaking speed of hardware turnover in public clouds pushes the question of how we are going to fully utilize that hardware to the forefront. Let’s create something that builds on what we have, but also supports complex scientific workflows that so far had to make due with improvisations and difficult compromises.

We believe that the proposed buffering layer will finally enable a less forceful convergence for hybrid BigData and HPC workloads, and dramatically accelerate workflows that hitherto had to use the PFS for communication between stages. Traditional write-heavy workloads such as checkpointing will benefit, and so will applications that have chosen to implement custom buffering such as LOFS [5]. Because buffering would occur in “multiple dimensions“ (e.g., vertically across hierarchy layers, and horizontally between nodes), we believe that seemingly pattern-free, read-heavy workloads as found in ML training [3] will see I/O performance gains. Finally, scratch-heavy read-after-write workloads such as reverse time migration might get a boost as well.

References

[1] Kougkas, Anthony, Hariharan Devarajan, and Xian-He Sun. "Hermes: a heterogeneous-aware multi-tiered distributed I/O buffering system." In Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing, pp. 219-230. 2018.

[2] Devarajan, Hariharan, Anthony Kougkas, Luke Logan, and Xian-He Sun. "Hcompress: Hierarchical data compression for multi-tiered storage environments." In 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 557-566. IEEE, 2020.

[3] Devarajan, Hariharan, Huihuo Zheng, Anthony Kougkas, Xian-He Sun, and Venkatram Vishwanath. "DLIO: A Data-Centric Benchmark for Scientific Deep Learning Applications." In 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), pp. 81-91. IEEE, 2021.

[4] Moody, Adam, Danielle Sikich, Ned Bass, Michael J. Brim, Cameron Stanavige, Hyogi Sim, Joseph Moore et al. UnifyFS: A Distributed Burst Buffer File System-0.1. 0. No. UnifyFS. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2017.

[5] Lack-of-File-System (LOFS), accessed online on Dec 15th, “http://lofs.io/”