Hermes: A Heterogeneous-Aware Multi-Tiered Distributed I/O Buffering System

Anthony Kougkas, Hariharan Devarajan, Xian-He Sun
Illinois Institute of Technology, Department of Computer Science
{akougkas, hdevarajan}@hawk.iit.edu, sun@iit.edu

ACM Reference format:

Anthony Kougkas, Hariharan Devarajan, Xian-He Sun
Illinois Institute of Technology, Department of Computer Science
{akougkas, hdevarajan}@hawk.iit.edu, sun@iit.edu . 2018. 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, Tempe, AZ, USA, June 11–15, 2018 (HPDC '18 Posters/Doctoral Consortium)*, 2 pages. DOI: 10.1145/3220192.3220196

Keywords: I/O buffering, Heterogeneous buffering, Layered buffering, Deep memory hierarchy, Burst buffers

1 EXTENDED ABSTRACT

The evolution of modern storage technologies is driven by the increasing ability of powerful High-Performance Computing (HPC) systems to run data-intensive problems at larger scale and resolution. In addition, larger scientific instruments and sensor networks collect extreme amounts of data and push for more capable storage systems [6]. However, as we move towards the exascale era, parallel file systems such as Lustre, GPFS, and PVFS2, face significant challenges in performance, scalability, complexity, and limited metadata services [2, 5], creating a significant *I/O bottleneck* which will lead to less scientific productivity [9, 10].

To reduce the I/O performance gap, modern storage subsystems are going through extensive changes, by adding additional levels of memory and storage in a hierarchy [1]. Newly emerging hardware technologies such as High-Bandwidth Memory (HBM), Non-Volatile RAM (NVRAM), Solid-State Drives (SSD), and dedicated buffering nodes (e.g., burst buffers) have been introduced to alleviate the performance gap between main memory and the remote disk-based PFS. Modern supercomputer designs employ such hardware technologies in a heterogeneous layered memory and storage hierarchy, we call Deep Memory and Storage Hierarchy (DMSH) [4, 7]. For example, Cori system at the National Energy Research Scientific Computing Center (NERSC), uses CRAY's Datawarp technology. Los Alamos National Laboratory Trinity supercomputer uses burst buffers with a 3.7 PB capacity and 3.3 TB/s bandwidth. Summit in Oak Ridge National Lab will also employ fast local NVMe storage for buffering. The new DMSH promises to offer a solution that can

efficiently support scientific discovery in many ways: improved application reliability through faster checkpoint-restart, accelerated I/O performance for small transfers and analysis, fast temporary space for out-of-core computations, and in-transit analysis.

As multiple layers of storage are added into HPC systems, the complexity of data movement among the layers increases significantly, making it harder to take advantage of the high-speed and low-latency storage systems [3]. Additionally, each layer of DMSH is an independent system that requires expertise to manage, and the lack of automated data movement between tiers is a significant burden currently left to the users [8]. Furthermore, popular I/O middleware, such as HDF5, PnetCDF, and ADIOS, are configured to operating with the traditional memory-to-disk I/O endpoints. This middleware provides great value by isolating users from the complex effort to extract peak performance from the underlying storage system, but it will need to be updated to handle the transition to a multi-tiered I/O configuration [8]. There is a need to seamlessly and transparently support access to layers of a deep memory and storage hierarchy.

Deep memory and storage hierarchies require a scalable, reliable, and high-performance software to efficiently and transparently manage data movement. New data placement policies, memory and metadata management, and efficient communication fabric is required to address DMSH complexity and realize its potential. We propose to raise the level of abstraction and make it easier for the user to perform I/O on top of a DMSH system. In this work, we present Hermes¹: a new, heterogeneous-aware, multi-tiered, dynamic, and distributed I/O buffering system. Hermes enables, manages, and supervises I/O buffering into DMSH and offers: a) *vertical and horizontal distributed buffering in DMSH* (i.e., access data to/from different levels locally and across remote nodes), b) *selective layered data placement* (i.e., buffer data partially or entirely in various levels of the hierarchy), c) *dynamic buffering via system profiling* (i.e., change the buffering schema dynamically by monitoring the system status such as capacity of buffers, messaging traffic, etc.). Hermes accelerates applications' I/O access by transparently buffering data in DMSH. Data can be moved through the hierarchy effortlessly and therefore, applications have a capable, scalable, and reliable middleware software to navigate the I/O challenges towards the exascale era. Lastly, by supporting both POSIX and HDF5 interfaces, Hermes offers ease-of-use to a wide-range of scientific applications. We strive for maximizing productivity, increasing resource utilization, abstracting data movement, maximizing performance, and supporting a wide range of scientific applications and domains. Ultimately, the goal is to ensure that application developers have a performance-portable I/O solution that

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

HPDC '18 Posters/Doctoral Consortium, Tempe, AZ, USA

© 2018 ACM. 978-1-4503-5899-6/18/06...\$15.00

DOI: 10.1145/3220192.3220196

¹Ancient Greek God of "messaging and the transgression of boundaries"

Figure 1: Hermes architecture.

minimize changes to their use of the middleware APIs, regardless of the underlying file system.

1.0.1 Hermes Overview. Hermes is designed as a middleware layer - sitting between applications and DMSH as shown in Figure ?? . As a middleware library, Hermes captures I/O calls, both POSIX and HDF5 (i.e., fopen, fread, fwrite, and H5Fcreate, H5Fread etc.) and redirects them to different layers of DMSH. Legacy applications can easily connect to Hermes by simple linking (i.e., LD_PRELOAD) or recompiling the code with our library. There are no changes to user code and there is no need to upgrade to a different workflow. We design Hermes to easily work with existing software. Our goal is to maximize user productivity by making I/O buffering transparent. Furthermore, Hermes also provides a new buffering API for users who want to explicitly take control of the data movement between layers of DMSH. This mode also allows Hermes to perform active buffering where data is shipped to the buffer nodes along with specific instructions or operations to be performed on them.

The architecture of Hermes can be seen in Figure 1. In DMSH systems, besides the main memory, every compute node might be equipped with an NVMe device or even an SSD. Additionally, shared buffering nodes, such as burst buffers, will most likely be present and positioned close to the compute nodes. Finally, a remote PFS supports all compute nodes with persistence and fault tolerance as important features. Hermes is a platform that aims to enable efficient access to the layers of DMSH and as such we distinguish two data paths: *a vertical and a horizontal hierarchy*. Vertical hierarchy refers to data movement within a compute node and all the way down to the burst buffers and PFS. Horizontal hierarchy refers to sending data to another compute node's RAM or NVMe device. The horizontal data movement is greatly optimized if there is an RDMA-capable network but Hermes can also support systems with no RDMA. Therefore, a system could consist of several layers of storage hierarchy, performance-wise, such as local RAM, remote RAM, local NVMe, remote NVMe, burst buffers, and PFS.

1.0.2 Initial Evaluation. To test our system under real applications workload, we configured Hermes in *persistent asynchronous* mode and selected the default data placement policy, *MaxBW*. **VPIC:** This application demonstrates a write-only I/O access pattern where at the end of each time step, each process writes data to an HDF5 file. During this evaluation we executed the application

Anthony Kougkas, Hariharan Devarajan, Xian-He Sun
Illinois Institute of Technology, Department of Computer Science
{akougkas, hdevarajan}@hawk.iit.edu, sun@iit.edu

Figure 2: I/O Buffering performance with VPIC-IO.

for 16 time steps. In Figure 2 we report only the I/O time which consists of write-time (i.e., what the application experiences) and flush-time (i.e., persisting the data asynchronously). In the largest tested scale of 1024 ranks, the baseline completed the test in 1192 seconds. Both Data Elevator and DataWarp wrote the entire dataset in 438 seconds. This is approximately a 2.5x improvement over the baseline. However, due to the higher bandwidth of the DMSH, Hermes' write performance is 5x and 2x higher than the baseline and the two buffering platforms we tested, respectively. When considering data flushing, Data Elevator overlaps small computations between each time step and flushes the contents of burst buffers in 1115 seconds whereas DataWarp flushes everything at the end in 1274 seconds. In contrast, Hermes leverages the computations but also the concurrency of the DMSH to flush all buffered data to PFS in 637 seconds. In summary, Hermes outperformed the baseline and state-of-the-art buffering platforms by 40% and 85% respectively.

We have introduced three novel data placement policies to efficiently utilize all layers of the new memory and storage hierarchy as well as three novel techniques to perform *memory*, *metadata*, and *communication* management in hierarchical buffering systems. Our evaluation results prove Hermes' sound design and show a 8x improvement compared to systems without I/O buffering support. Additionally, Hermes outperforms by more than 2x state-of-the-art buffering platforms such as Data Elevator and Cray's Datawarp.

REFERENCES

- [1] John Bent, Gary Grider, Brett Kettering, Adam Manzanara, Meghan McClelland, Aaron Torres, and Alfred Torrez. 2012. Storage challenges at los alamos national lab. In *Mass Storage Systems and Technologies (MSST), 2012 IEEE 28th Symposium on*. IEEE, 1–5.
- [2] D Brown, Paul Messina, D Keyes, J Morrison, R Lucas, J Shalf, P Beckman, R Brightwell, A Geist, J Vetter, et al. 2010. Scientific grand challenges: Crosscutting technologies for computing at the exascale. *Office of Science, US Department of Energy, February* (2010), 2–4.
- [3] Adrian M Caulfield, Joel Coburn, Todor Mollov, Arup De, Ameen Akel, Jiahua He, Arun Jagatheesan, Rajesh K Gupta, Allan Snavey, and Steven Swanson. 2010. Understanding the impact of emerging non-volatile memories on high-performance, io-intensive computing. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society, 1–11.
- [4] Adrian M Caulfield, Laura M Grupp, and Steven Swanson. 2009. Gordon: using flash memory to build fast, power-efficient clusters for data-intensive applications. *ACM Sigplan Notices* 44, 3 (2009), 217–228.
- [5] Jack Dongarra, Pete Beckman, Terry Moore, Patrick Aerts, Giovanni Aloisio, Jean-Claude Andre, David Barkai, Jean-Yves Berthou, Taisuke Boku, Bertrand Braunschweig, et al. 2011. The international exascale software project roadmap. *The international journal of high performance computing applications* 25, 1 (2011), 3–60.
- [6] Tony Hey, Stewart Tansley, Kristin M Tolle, et al. 2009. *The fourth paradigm: data-intensive scientific discovery*. Vol. 1. Microsoft research Redmond, WA.
- [7] Sudarsun Kannan, Ada Gavrilovska, Karsten Schwan, Dejan Milojicic, and Vanish Talwar. 2011. Using active NVRAM for I/O staging. In *Proceedings of the 2nd international workshop on Petascale data analytics: challenges and opportunities*. ACM, 15–22.
- [8] Glenn K Lockwood, Damian Hazen, Quincey Koziol, RS Canon, Katie Antypas, Jan Balewski, Nicholas Balthaser, Wahid Bhimji, James Botts, Jeff Broughton, et al. 2017. *Storage 2020: A Vision for the Future of HPC Storage*. Technical Report. NERSC.
- [9] Daniel A Reed and Jack Dongarra. 2015. Exascale computing and big data. *Commun. ACM* 58, 7 (2015), 56–68.
- [10] John Shalf, Sudip Dosanjh, and John Morrison. 2010. Exascale computing technology challenges. In *International Conference on High Performance Computing for Computational Science*. Springer, 1–25.