High-Performance PMEM-Aware Collective I/Os

Keegan Sanchez[1] Alex Gavin[1] Suren Byna[2,3] Kesheng Wu[2] Xuechen Zhang[1]

[1]Washington State University Vancouver [2]Lawrence Berkeley National Laboratory [3]The Ohio State University

What is Collective IO?

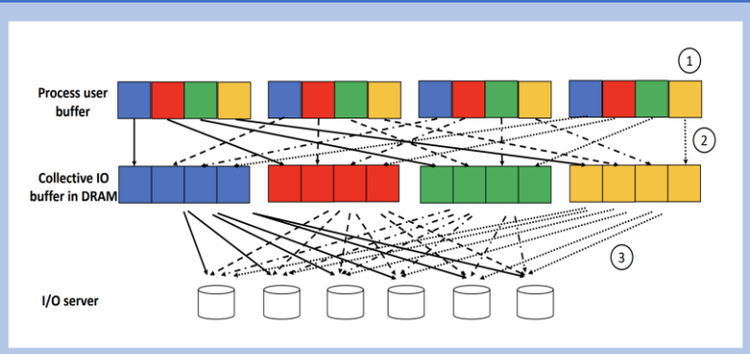
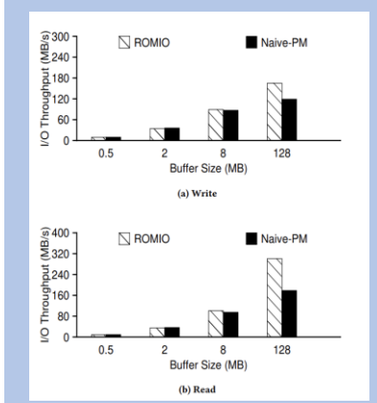
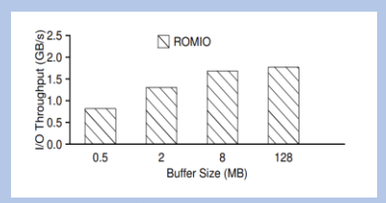


Diagram of Collective IO. Many small IOs are "collected" into a single distributed IO, which is spread across processes, and then redistributed.

- *Collective IO* re-orders small requests into larger requests.
- More larger requests reduces filesystem overhead.
- Collective IO adds inter-process communication.
- Additional inter-process communication can incur overhead with a large number of processes.

Issues of Collective I/O





Above: The effects of increasing collective I/O buffer size on Perlmutter. A larger buffer increases throughput, but the size of a traditional collective I/O buffer is limited due to the memory's volatility.

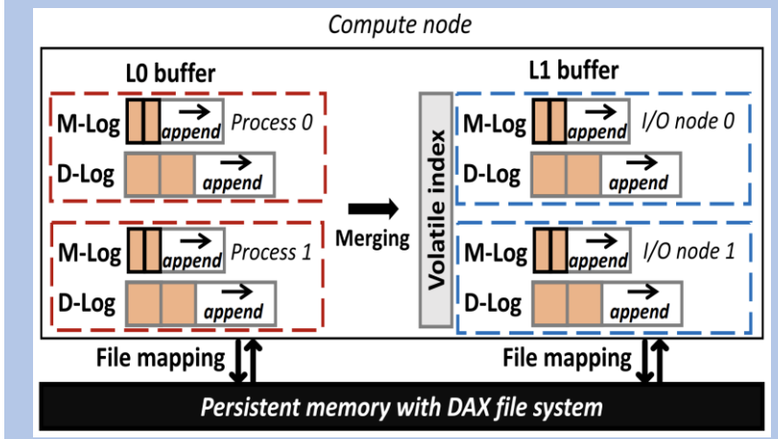
Left: The effects of increasing the buffer size on the Camas cluster. Naïve PM is an implementation that simply uses Persistent Memory instead of DRAM. Throughput is increased.

Research Focuses

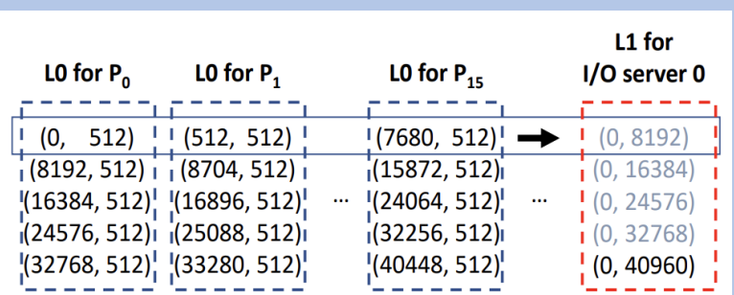
- Replace the DRAM buffer used in collective IO with a persistent memory (PMEM) buffer.
- Implement a log-based buffer and two-phase merging to reduce communication overhead.

Design of PMIO

- Persistent memory is a storage medium that sits between disk and RAM.
- Slower than RAM, vastly outperforms traditional SSD's, while still being non-volatile.



Layout of **PMIO log buffers**. Each process has its own log buffers and tracks its data and metadata.

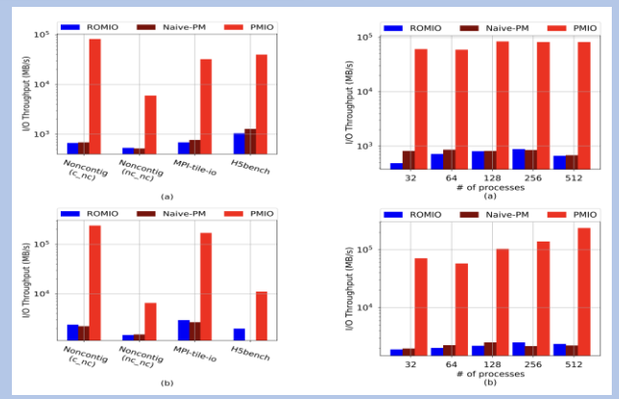


Two-phase merging. Logs are first merged across processes, and then merged again on the I/O servers.

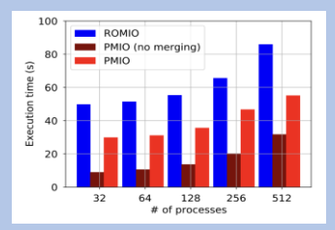
Evaluation

- The Perlmutter supercomputer at LBL
- Lustre file system with 64 KB stripe size and 4 OSTs
- Benchmarks
  - Noncontig
  - Mpi-tile-io
  - H5bench

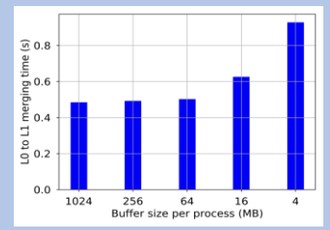
- ROMIO: the default collective I/O function in ADIO/MPICH
- Naïve-PM: persistent-memory buffers without log structure and two-phase merging.
- PMIO: our solution



Left: Read and write results across benchmarks. (a) is write and (b) is read. **Right:** Strong Scalability, accessing 32 GiB while increasing process count from 32 to 512.



Effect of increasing process count.



Merging as collective buffer size decreases.

Acknowledgements

This work was prepared in partial fulfillment of the requirements of the Visiting Faculty Program (VFP), managed by Workforce Development & Education at Berkeley Lab. This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center. The research was also supported by the US National Science Foundation under CNS 1906541, CNS 2216108, and OAC 2243980.