Leveraging Intel Optane for HPC Workflows

Ranjan Sarpangala Venkatesh[1], Tony Mason[2,1], Pradeep Fernando[1], Greg Eisenhauer[1], and Ada Gavrilovska[1]

[1]Georgia Institute of Technology, [2]University of British Columbia

I. Introduction

High Performance Computing (HPC) system performance is increasingly dependent upon the data movement costs across application workflow components. Coupling and colocating simulation and analytics application components is one approach to reducing data movement. These *in situ* executions benefit by using persistent memory, which provides large memory capacity and read/write latencies comparable to DRAM, while ensuring DRAM capacity can be utilized for core computations [1]. The HPC community has rapidly adopted Intel® Optane™ DC Persistent Memory (Intel Optane PMEM), which is the first commercially available persistent memory. It is already part of several supercomputer facilities and upcoming system designs, including Aurora, Frontera, and Barcelona Supercomputing Center.

This raises the question, *How to maximize the benefit that in situ workflows can obtain from using PMEM for their data exchanges?* Recent work has provided performance analysis of individual applications on PMEM platforms [2], [3] by investigating the performance implications of PMEM technology parameters such as read-write access latency and bandwidth asymmetry, the characteristic of the device-internal cache, access stride, etc., on the variability of the effective device bandwidth and latency. These studies make recommendations for configuring applications' access locality, granularity, stride, and concurrent operations to achieve high PMEM performance.

Unlike a single application, HPC workflows are typically composed of simulation and analytics application which iterate over I/O data as it is being streamed across the execution pipeline [1], [4]. When workflow components are deployed *in situ*, prior work has demonstrated the benefits from their shared use of the local PMEM resources to enable streaming I/O performance at memory speeds. *However, optimizing individual workflow components does not guarantee an optimal choice in end-to-end workflow performance.* Figure 1 illustrates this by showing the normalized runtime of two sets of workflows based on the GTC and miniAMR applications. Within each set, the workflows differ with respect to the analytics component – an I/O-intensive component vs. a compute-intensive analytics kernel. We run these workflows in several configurations, the parameters of which are not germane for the current illustration. We observe that even for scenarios where the simulation application is the same, optimizing purely for this component is insufficient as a change in the analytics kernel can result in up to 1.4-1.6× loss in performance (for miniAMR).

The shared use of PMEM in workflows is application-specific and sensitive to properties such as the composition of the iteration cycle of individual components (i.e., their compute vs. I/O time), writer and reader mix and concurrency, the granularity of I/O operations, etc. There is no single configuration that works for all workflows. Although prior work has explored the impact of these configuration parameters individually, the interplay of these in the context of HPC workflows has not been studied. In this presentation, we explore the effect of decisions regarding how workflow components are scheduled on a server platform and how they use the available Intel Optane PMEM resources for streaming I/O have on the end-to-end workflow performance.

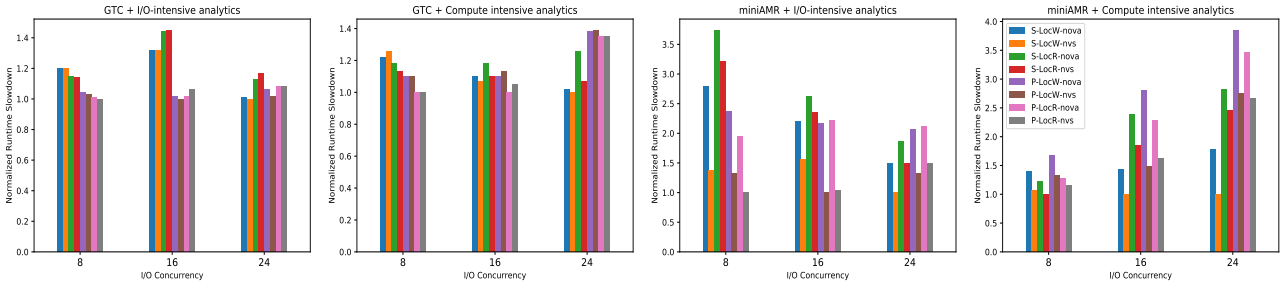
II. Experimental Analysis

We take an experimental approach and consider different workflow deployment scenarios and workflows with different I/O characteristics. The workflows are based on microbenchmarks and on real application kernels, the GTC fusion simulation and miniAMR.

Workflow Deployment Considerations. Since we focus on challenges related to the shared use of PMEM resources, we focus on workflow deployments that change how PMEM is used and how contention exhibited when accessing PMEM. For concreteness, we present the discussion in the context of a simplified server platform comprising two compute sockets, each with locally attached DRAM and PMEM resources.

Locality of PMEM accesses. In the sample platform, the simulation (writer) and analytics (reader) components of a workflow are placed on two different sockets. Their streaming I/O channel is allocated in one of the PMEM modules, either local to the simulation, or local to the analytics. The locality choice impacts the I/O performance experienced by each of the workflow components. This leads to the question: what **placement** decision should the system scheduler make? Since only one of the workflow components can be local to the PMEM, which placement maximizes the end-to-end workflow performance?

Contention for PMEM resources. In situ workflow components can execute concurrently or they can be scheduled to execute in an interleaved manner. Prior work has shown that the later case is important, even when each workflow component has sufficient CPU resources because it limits the contention for a single shared resource, such as accelerators (e.g., GPU) or the interconnect. For shared PMEM configurations, the scheduler must consider the **co-scheduling execution mode** – whether to allow the simulation and analytics to be scheduled in *parallel* or to enforce that they iterate over the streaming I/O



(a) (b) (c) (d)

Fig. 1: Workflow runtime slowdown normalized to the fastest configuration

TABLE I: Summary of configurations

Config label	Execution Mode	Placement
$S-LocW$	Serial	local-write-remote-read
$S-LocR$	Serial	remote-write-local-read
$P-LocW$	Parallel	local-write-remote-read
$P-LocR$	Parallel	remote-write-local-read

data *sequentially*, in an interleaved manner, so as to minimize the performance impact of PMEM controller contention.

Workflow Streaming I/O Parameters The sensitivity of a workflow to PMEM bandwidth or access latency is determined by a number of workflow-specific parameters.

Iteration cycle composition. The lengths of compute and I/O phases in the simulation and analytics are important when determining the optimal configuration. For instance, GTC has a long compute phase which allows concurrent I/O from analytics kernels without a significant drop in I/O performance for low and medium levels of concurrency. However, miniAMR has a dominant I/O phase compared to its compute causing contention for PMEM resources.

I/O granularity. Large objects reduce the software overhead in committing them to persistent memory. A large number of small objects cause huge software overhead. In our experiments, in each iteration, GTC uses 10s of very large objects of size at least 229MB, whereas miniAMR uses 100K+ small objects each of size 4.5KB.

I/O concurrency. A large number of MPI ranks can provide concurrency and reduce compute time for certain applications. However, increased concurrency in more contention for the Optane internal cache and reduced effective bandwidth. Hence, identifying a good trade-off is important.

III. Insights

The four scheduler configurations that we consider are based on execution mode (serial interleaved, or parallel non-interleaved) and PMEM locality (simulation or analytics), which are summarized in Table I.

Figure 1 shows slowdown of the end-to-end execution time of up to 70%, normalized to the best configuration for each workflow, with the naive use of PMEM. We report the impact on workflows that use a PMEM-specialized file system as an I/O interface (based on NOVAfs [5]) vs. a PMEM specialized I/O streaming runtime (using NVStream [1]).

Future HPC workflow schedulers have to consider both the compute and I/O characteristics of workflow components to make intelligent PMEM-aware choices in placement and execution scheduling. In order to aid with such scheduler designs, we summarize the observations as follows:

- Maximize effective bandwidth by limiting concurrent device accesses.
- High software stack I/O overheads lower PMEM contention and allow for concurrent executions.
- Interleaved compute hides effects of access contention and high remote latency.

IV. Acknowledgements

This work is partially supported by NSF awards SPX-1822972 and CNS-2016701, and via access to an Optane PMEM hardware testbed provided by Intel.

References

[1] P. Fernando, A. Gavrilovska, S. Kannan, and G. Eisenhauer, "Nvstream: Accelerating hpc workflows with nvram-based transport for streaming objects," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, 2018, pp. 231–242.

[2] J. Yang, J. Kim, M. Hoseinzadeh, J. Izraelevitz, and S. Swanson, "An empirical guide to the behavior and use of scalable persistent memory," in *18th {USENIX} Conference on File and Storage Technologies ({FAST} 20)*, 2020, pp. 169–182.

[3] I. B. Peng, M. B. Gokhale, and E. W. Green, "System evaluation of the intel optane byte-addressable nvm," in *Proceedings of the International Symposium on Memory Systems*, 2019, pp. 304–315.

[4] S. Klasky, H. Abbasi, J. Logan, M. Parashar, K. Schwan, A. Shoshani, M. Wolf, S. Ahern, I. Altintas, W. Bethel *et al.*, "In situ data processing for extreme-scale computing," *Scientific Discovery through Advanced Computing Program (SciDAC11)*, pp. 1–16, 2011.

[5] J. Xu and S. Swanson, "{NOVA}: A log-structured file system for hybrid volatile/non-volatile main memories," in *14th {USENIX} Conference on File and Storage Technologies ({FAST} 16)*, 2016, pp. 323–338.