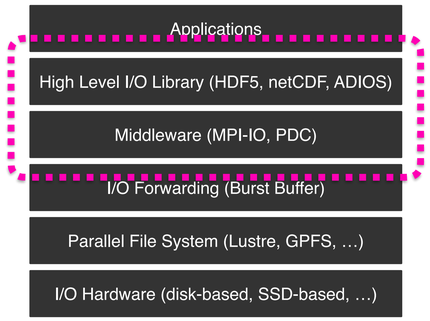
Toward Scalable Middleware for Shared HPC Resources

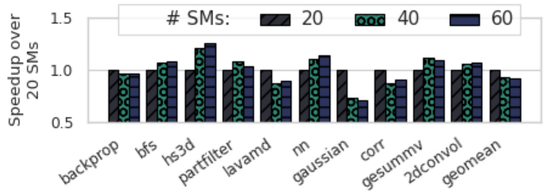
John Ravi, *Advisor: Dr. Michela Becchi, Co-advisor: Dr. Suren Byna*
Department of Electrical and Computer Engineering, North Carolina State University

Overview

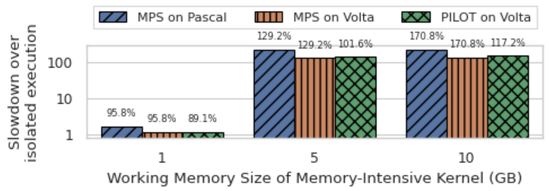


In this work, we identify key challenges that arise when sharing resources in an HPC context. We evaluate real world scenarios both at node-level and cluster-level. Using these insights, we propose middleware to mitigate and improve quality of service.

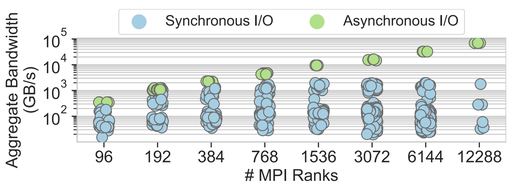
Challenges



Naïvely scaling legacy workloads from the Rodinia benchmark suite across SMs on a modern GPU NVIDIA V100 does not scale.

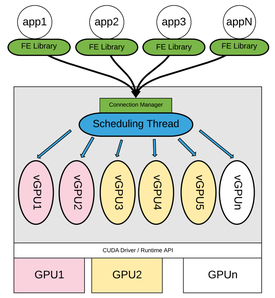


Space-sharing the GPU can lead to slowdowns even for simple vectorAdd kernel when oversubscribing a resource such as GPU Memory.

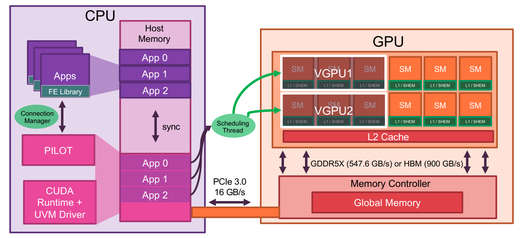


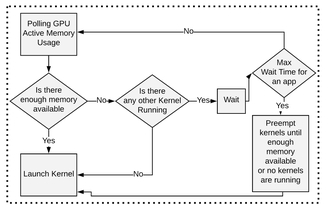
In current large-scale systems, the parallel file system and network are a shared resource across multiple workloads and users. This leads to poor quality of service on I/O synchronous requests as shown on the left with a VPIC-IO write benchmark on Summit.

GPU Runtime Middleware

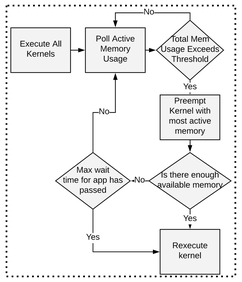


PILOT is CUDA runtime middleware that enables transparent space sharing similar to NVIDIA MPS (multi-process service). A high-level diagram is shown to the left, and a per-GPU diagram is shown to the right. With PILOT, we implement 3 modes that aim to improve QoS and reduce interconnect contention due to intensive kernels (flowcharts shown below).

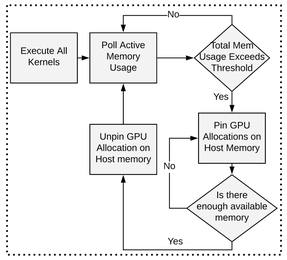




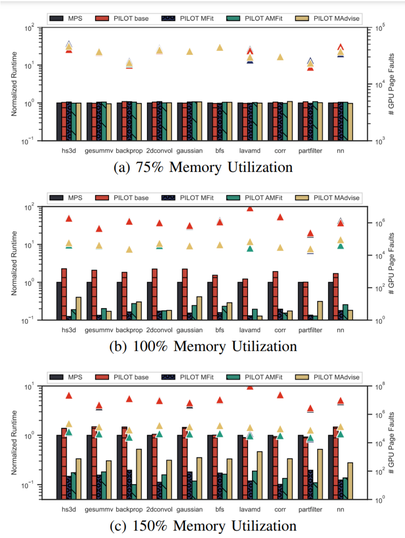
MemoryFit mode is a subscription-based scheduler that only launches a kernel when there is enough memory available to handle its allocations. It is the most conservative.



ActiveMemoryFit mode launches a kernel immediately when ready and uses a monitor to preempt kernels when a memory threshold is triggered.



MemoryAdvise mode utilizes pinned memory to avoid memory oversubscription scenarios. This mode does not require preemption and is the recommended choice based on our results.



(a) 75% Memory Utilization

(b) 100% Memory Utilization

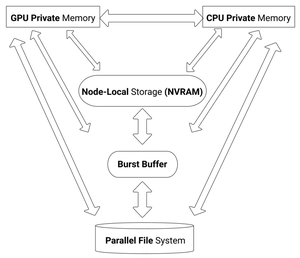
(c) 150% Memory Utilization

Benchmarks co-run with memory-intensive kernel

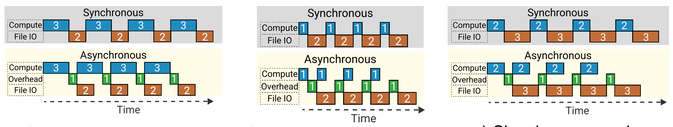
Key Takeaways

- # page faults measured correlates with program runtime performance
- none of the mitigations schemes incur noticeable overhead when memory is not oversubscribed
- *MemoryAdvise* is an effective strategy

I/O Middleware

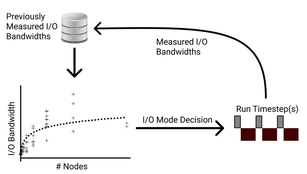


Many large-scale applications reach I/O bottlenecks before saturating the computation resources. Thus, HPC systems provide caching locations to improve performance (shown above). Middleware I/O libraries have recently begun implementing asynchronous I/O to take advantage of these caching regions. Asynchronous I/O can dramatically improve application performance at the cost of memory and compute overhead used by background helper threads.



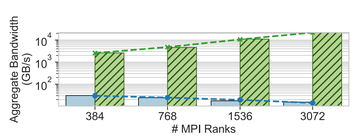
a) Ideal case for overlap b) Partial overlap possible c) Slowdown scenario

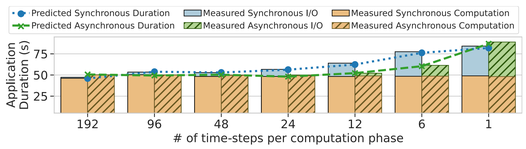
We found that many HPC applications have distinct computational and I/O phases. These applications follow an iterative approach and rely on the I/O phase to checkpoint state or visualization data. Treating the computation and I/O phases as distinct phases provides an opportunity to overlap them across iterations, as shown in the timelines above.



We use linear regression to model the I/O performance across application iterations to decide when caching is required, as shown to the left.

The plots below show the synchronous and asynchronous I/O performance for Nyx on the Summit. When the computation phase becomes too small, the transactional overhead of enabling asynchronous I/O dominates.





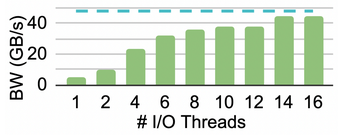
Impact and Ongoing

Our work takes a step toward providing a better quality of service. We introduce a runtime CUDA middleware that improves QoS for GPUs. We also introduce and study two new features of HDF5, GDS VFD and Async I/O. The former improves I/O latency while the latter improves and hides variability in I/O latency. We are researching two more orthogonal approaches to deal with bottlenecks in multi-tenant systems: GPUDirect Storage and Scientific Data Reduction.

HDF5, GDS VFD (GPUDirect Storage)

- Eliminates the CPU memory "bounce buffer"
- Direct path from GPU to Filesystem
- Frees up CPU cycles
- Enables more data paths for runtime system to delegate traffic

Aggregate H5Dread with 8 GPUs on DGX A100 + Lustre



Scientific Data Reduction

- Reduce storage requirements
- Reduce bandwidth requirements
- Application-tuned lossy-based compression
- Integration with I/O libraries

Publications

[1] J. Ravi, S. Byna and Q. Koziol, "GPU Direct I/O with HDF5," 2020 IEEE/ACM Fifth International Parallel Data Systems Workshop (PDSW), 2020, pp. 28-33, doi: 10.1109/PDSW51947.2020.00010.
[2] J. Ravi, T. Nguyen, H. Zhou and M. Becchi, "PILOT: a Runtime System to Manage Multi-tenant GPU Unified Memory Footprint," 2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC), 2021, pp. 442-447, doi: 10.1109/HiPC53243.2021.00063.
[3] H. Tang, Q. Koziol, J. Ravi and S. Byna, "Transparent Asynchronous Parallel I/O Using Background Threads," in IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 4, pp. 891-902, 1 April 2022, doi: 10.1109/TPDS.2021.3090322.

Acknowledgements

CNS-1812727