Towards Efficient Checkpointing Across Deep Tiers of Memory Hierarchy

Avinash Maurya [1] (am6429@rit.edu) M. Mustafa Rafique [1] (Advisor) Bogdan Nicolae [2] (Advisor)

[1]Rochester Institute of Technology, NY, USA [2]Argonne National Laboratory, IL, USA

Motivation and Challenges

- HPC applications **checkpoint large volumes of data at high-frequency**.
- Checkpointing is a **fundamental I/O pattern** for a large number of scenarios: fault-tolerance, analytics, reproducibility, revisiting previous states, etc.
- Efficiently **restoring previous checkpoints** is important in productive scenarios, e.g. adjoint computations revisit previous checkpoints during the backward pass.
- Modern HPC systems are equipped with **hierarchical memory tiers**, e.g. GPU HBM, DRAM, NVMe based SSDs, burst buffer, remote storage, etc.
- State-of-art data-movement and checkpoint-restore runtimes are not optimized to take advantage of fast HBM and interconnects (e.g. NVLink, NVSwitch).

Key Challenges

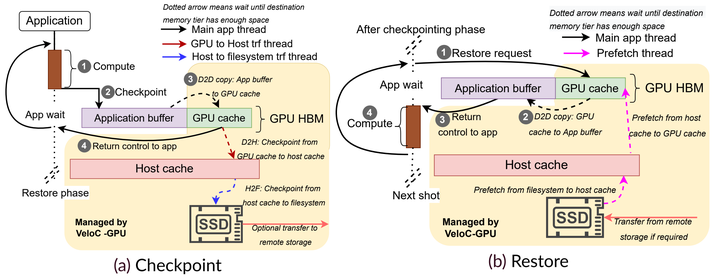
- **Slow flushes to larger cache tiers:** GPU HBM cannot hold all checkpoints and synchronous transfers through larger cache tiers increases application I/O time.
- **Checkpoint load imbalance:** Cache consumption on various tiers is uneven, due to which processes with abundant cache wait for others to write to slower tiers.
- **Slow cache initialization:** Allocating, mapping and pinning various cache tiers for short-lived applications incurs significant overheads on the application runtime.
- **Restore oblivious eviction and prefetching:** Restore order is not exploited during cache evictions to minimize cache misses, or during prefetch operations.

Overview of Key Contributions

- **Optimized multi-level flushing strategy:** Hierarchical caching tiers, from GPU HBM to remote storage, dedicated unidirectional transfer threads across each tier.
- **Collaborative load balancing strategy:** Leverages spare high-speed cache on peer devices/nodes and fast interconnects to avoid flushing to slower tiers.
- **Proactive asynchronous cache initialization and transfers**: Minimizes application I/O time by mitigating overheads of cache allocation, eviction, cache misses.
- **Leveraging foreknowledge of restore-order:** Efficient management of eviction and prefetching schedule based on finite-state-machine.

Caching Infrastructure and Implementation Overview

- For data consistency, the **application is blocked when checkpointing or restoring** from GPU cache, awaiting evictions, as shown in Fig. 1a and Fig. 1b, respectively.
- **Integrated with VELOC**, a production-ready HPC checkpoint-restart runtime, to perform transparent and asynchronous transfers across memory tiers.
- **Used for real-world workloads** in the oil industry: Reverse-time migration (RTM).

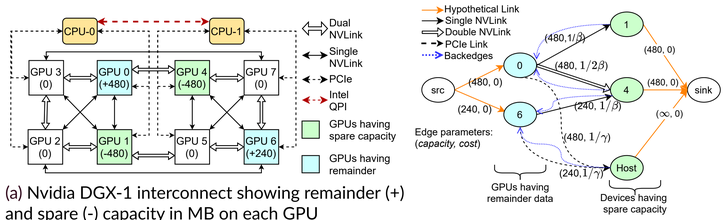


(a) Checkpoint (b) Restore

Figure 1. Interaction between hierarchical cache tiers during checkpoint and restore

Collaborative Checkpointing for Load Balancing

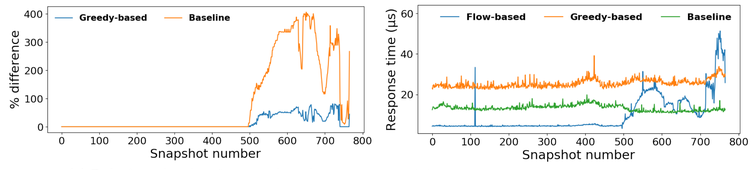
- **Uneven cache utilization** forces the processes resident on overloaded cache to write to slower tiers (Fig. 2a).
- Our **min-time max-flow scheduling strategy** generates an optimal transfer schedule to balance load across a given cache tier using fast interconnects (Fig. 2b).
- On Nvidia DGX-1 system, our approach shows up to **4x faster checkpointing** as compared to transferring to the host memory when cache is exhausted (Fig. 3a).
- Our approach generates **optimal transfer schedules** in sub-ms (Fig. 3b).



(a) Nvidia DGX-1 interconnect showing remainder (+) and spare (-) capacity in MB on each GPU

(b) Min-time max-flow graph

Figure 2. Illustration of checkpoint size imbalance across GPUs

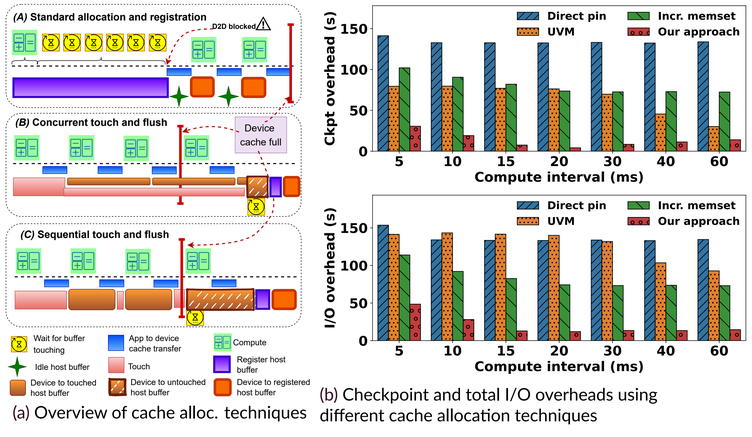


(a) Relative checkpoint overhead (b) Response time for transfer scheduling

Figure 3. Performance gains and response time of collaborative checkpointing

Efficient Cache Allocation for Fast Cache Initialization

- **High-frequency checkpoints** are produced every few ms by short-lived jobs.
- **Cache allocation imposes initialization overhead** on the application, which may not get amortized over the applications' execution time.
- **Touching memory** incrementally forces physical page mapping, that can be done either concurrently during transfers, or exclusively after I/O (Fig. 4a).
- On Nvidia DGX-A100 system, our techniques **minimize the checkpoint and total I/O overheads by 26x and 12x**, respectively, compared to the state-of-art (Fig. 4b).

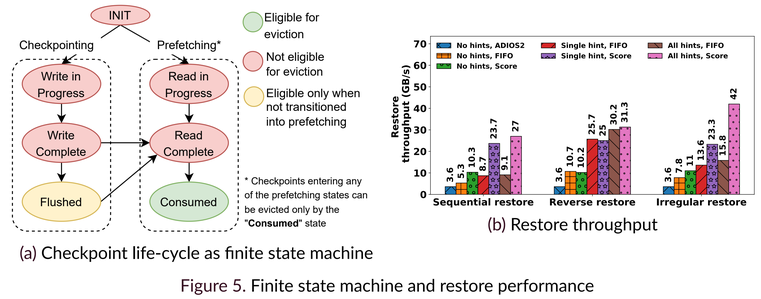


(a) Overview of cache alloc. techniques (b) Checkpoint and total I/O overheads using different cache allocation techniques

Figure 4. Cache allocation techniques and checkpointing workflow

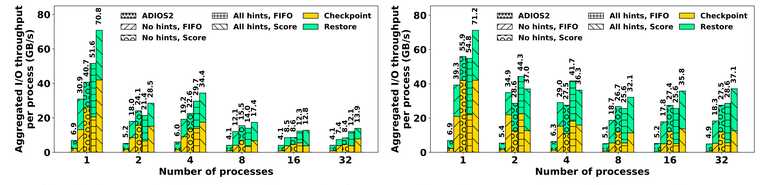
Work in progress: Foreknowledge based eviction and prefetching

- The **deterministic restore-order** of HPC applications, is not taken into consideration while evicting previous checkpoints from cache tiers.
- Existing data-movement engines are **not optimized for concurrent checkpoint production and consumption** across cache tiers.
- We design a **finite-state-machine**, we enable seamless transition of checkpoints from checkpointing to prefetching phase that maximizes cache hits (Fig. 5a).
- We develop a **score-based look-ahead eviction technique** that factors in the restore hints provided by the application for efficient cache eviction.
- On Nvidia DGX-A100 system with uniform sized checkpoints, we observe up to **11.7x faster restore** operations as compared to the state-of-art solutions (Fig. 5b).
- Scalability study on 4 nodes demonstrates **3.4x and 7.6x faster I/O throughput** for tightly coupled (Fig. 6a) and embarrassingly parallel (Fig. 6b) scenarios, respectively.



(a) Checkpoint life-cycle as finite state machine (b) Restore throughput

Figure 5. Finite state machine and restore performance



(a) Tightly coupled (sync at each iteration) (b) Embarrassingly parallel across all GPUs

Figure 6. Scalability test of eviction and prefetch approaches

Conclusion and Discussions

- State-of-the-art data-movement solutions incur significant I/O overheads on applications while transferring across hierarchical memory tiers.
- We highlight the challenges of large-scale high-frequency checkpointing for modern HPC applications, that utilize checkpoint-restore beyond fault-tolerance.
- We design hierarchical caching infrastructure, efficient cache initialization and data management strategies to minimize the application I/O wait time.
- We plan to extend support of Nvidia GPUDirect storage that enables interfacing between GPU and NVMe based SSDs to mitigate I/O contention on the host cache.

Publications

[1] Avinash Maurya, Bogdan Nicolae, Mustafa Rafique, Amr M. Elsayed, Thierry Tonellot, and Franck Cappello. Towards Efficient Cache Allocation for High-Frequency Checkpointing. In *Proceedings of the 29th IEEE International Conference on High Performance Computing, Data, and Analytics, 2022 (HiPC'22)*.

[2] Avinash Maurya, Bogdan Nicolae, Mustafa Rafique, Thierry Tonellot, and Franck Cappello. Towards Efficient I/O Scheduling for Collaborative Multi-Level Checkpointing. In *Proceedings of the 29th IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'21)*, Virtual, Portugal, 2021.