Management of Deep Memory Hierarchy in the Exascale Era

ANL: Swann Perarnau, Brice Videau
LLNL: Maya Gokhale, Ivy Peng, Roger Pearce, Eric Green, Keita Iwabuchi, Abhik Sarkar

Incorporating novel memory types into exascale systems and applications

Within Argo memory thrust we are working on the best software techniques to incorporate complex new memory types into the memory hierarchy. We are currently pursuing two strategies: a user-space paging service for NVM devices (UMap) and explicit memory management for heterogeneous architectures (AML).

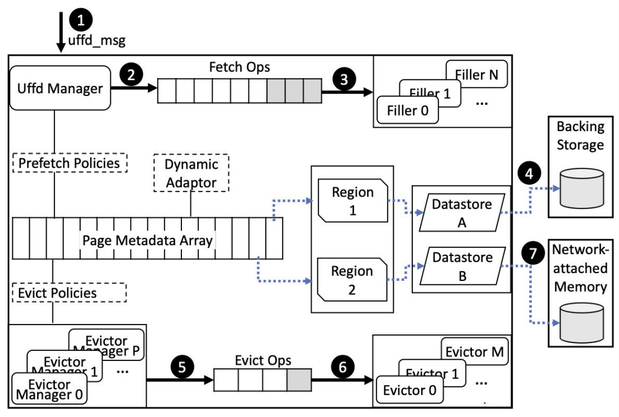
UMap

Overview

- UMap enables user-space optimizations for memory mapping NVM devices into the complex memory hierarchy
- Facilitates direct access to large data sets through virtual address spaces
- Provides flexible configurations suited to massive observational and simulation data sets
- High-performance design features I/O decoupling, dynamic load balancing, and application-level controls
- Demonstrated use cases in graph processing, database, metagenomics, and file compression applications

UMap Design:

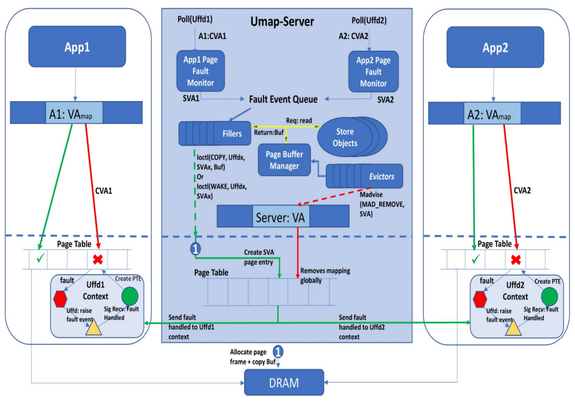
- Asynchronous message-based API (1—3)
- Resolves page faults in *regions* by fetching/flushing data from datastores following user-defined policies (4—6)
- Customized page sizes, buffer size, data source (4, 7)



MP-UMap Design

- Enables shared access of UMapped buffers in Rd-only capacity by multiple processes
- Service library instantiates "umap-servers"
- Client library binds application process(es) to a umap-server

details:https://mp-umap.readthedocs.io/en/latest/design.html

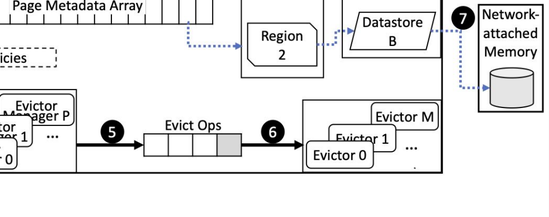


Features

- Adaptive Buffer (AB): automatic adjustment of Page Cache size based on system memory pressure
- SparseStore: a store module for transparently partitioning a memory-mapped persistent region into multiple files with dynamic and sparse allocation.

Applications

- UMap integrated with UMPIRE's file-backed memory resource manager
- Multi-BFS
  - MP-UMap outperforms mmap
- AB boosts MP-UMap's performance
  - AB avoids OOM at higher process count
- UMap integrated with Ligra graph processing framework for symmetric graph IO
  - supports popular graph applications like pagerank, radii, kcore, etc.



MP-UMap + Adaptive Buffer

Test Application: BFS-30, available Memory: 166GB (constrained), Linux Kernel version: 5.10.28, MP-Umap fixed Buffer size: 132 GB

UMap is open source and available at:

- Umap v2.0: https://github.com/LLNL/umap
- MP-UMap v1.0: https://github.com/LLNL/umap/tree/mpumap

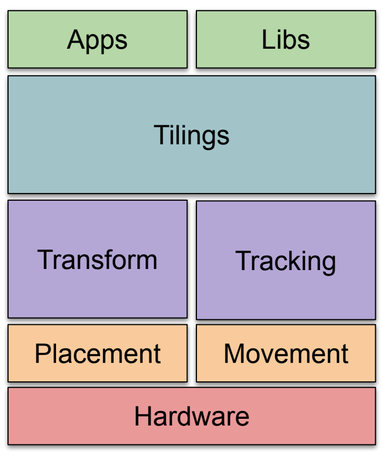
AML

Overview

- Explicit, application-aware memory management:
  - Descriptive API for application-level data access,
  - Explicit placement and movement of data,
  - Abstract topology and memory device complexity.
- Collection of building blocks:
  - Generic: few assumptions about user application, hardware-oblivious,
  - Customizable: application users can specialize the inner implementation of each offered abstraction,
  - Composable: mix and match as needed.
- Locality optimizations for current and future hardware generations:
  - Static allocations with application insights,
  - Asynchronous movement/reshape to optimize data locality on the go (data layout, HBM management, locality, GPUs).
  - Data replication across multiple devices

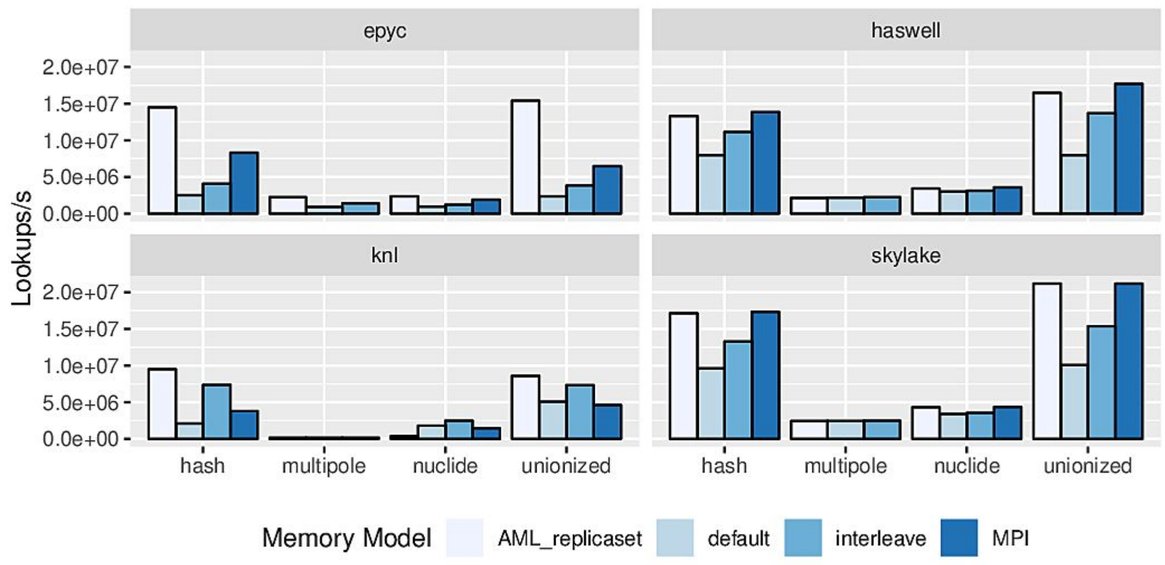
Key Components

- Topology & hardware management (NUMA, hwloc, CUDA)
- Data layout descriptions (application-specific)
- Tiling schemes
- Data movement facilities (`transform`, `copy`)
- Pipelining helpers (asynchronous requests)



Explicit Data Replication in Low-Latency Memory

- Automatic topology discovery and low-latency memories copy.
- Each thread access the closest replica.
- Performance on-par with tuned MPI process pinning on NUMA systems but with improved memory usage.
- Improved performance compared to OpenMP data sharing



Recent Developments

- New public release 0.2.0
- Support for most GPU APIs (OpenCL, CUDA, HIP, oneAPI)
  - Runs on Theta, Summit, Arcticus, Crusher
- Integration into ExaSMR's XSBench using the replicaset feature

AML is open source and is available at:
https://argo-aml.readthedocs.io/

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.