Management of Deep Memory Hierarchy in the Exascale Era

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

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).

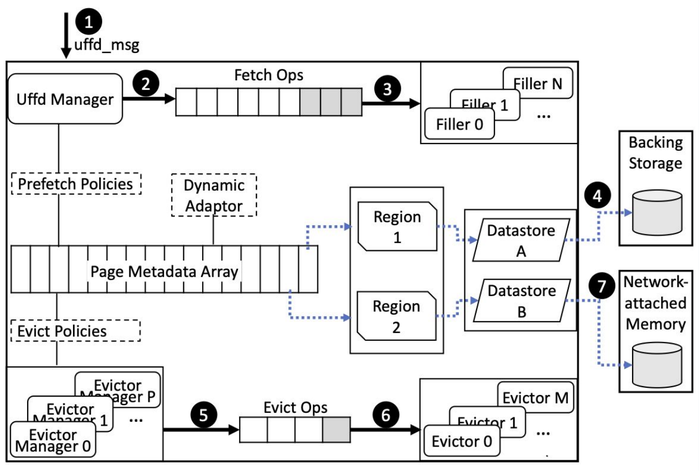
UMap

Overview

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

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)

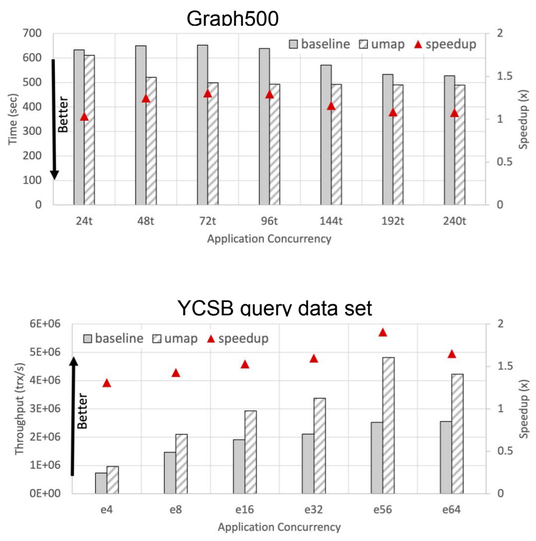


Features

- User space flexibility, ease of use, extensibility with plugin handlers (multiple files, distributed data sets)
- Leverages standard Linux features, so portable on new systems, no system-wide impact to other applications

Applications

- Out-of-core Graph Algorithms
  - Perform level-synchronous breadth-first search on an R-MAT scale-31 CSR graph (529 GB)
  - Outperforms system mmap at multiple scales of concurrency
- Database operations
  - Key/value store nstore ported to UMap with 10 lines of code change
  - 384 GB persistent memory pool on the local NVMe-SSD
  - 90% speedup over system mmap at higher thread level concurrency
- Metagenomics search
  - Use a 480 GB Livermore Metagenomic Analysis Toolkit (LMAT) database
  - Can access database from local SSD or from memory servers on network
  - Shows scalable performance as query load to memory servers increases



UMap is open source and available at:

- Umap version 2.0: https://github.com/LLNL/umap
- UMap apps: github.com/LLNL/umap-apps

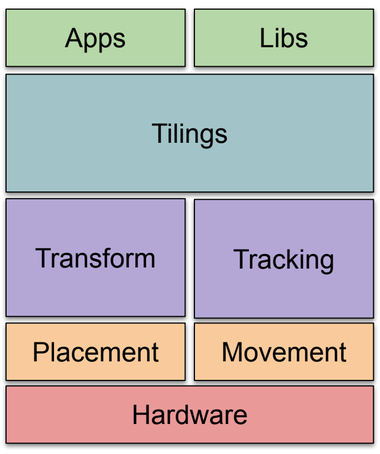
AML

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, NUMA locality).
  - Data replication in low-latency memories

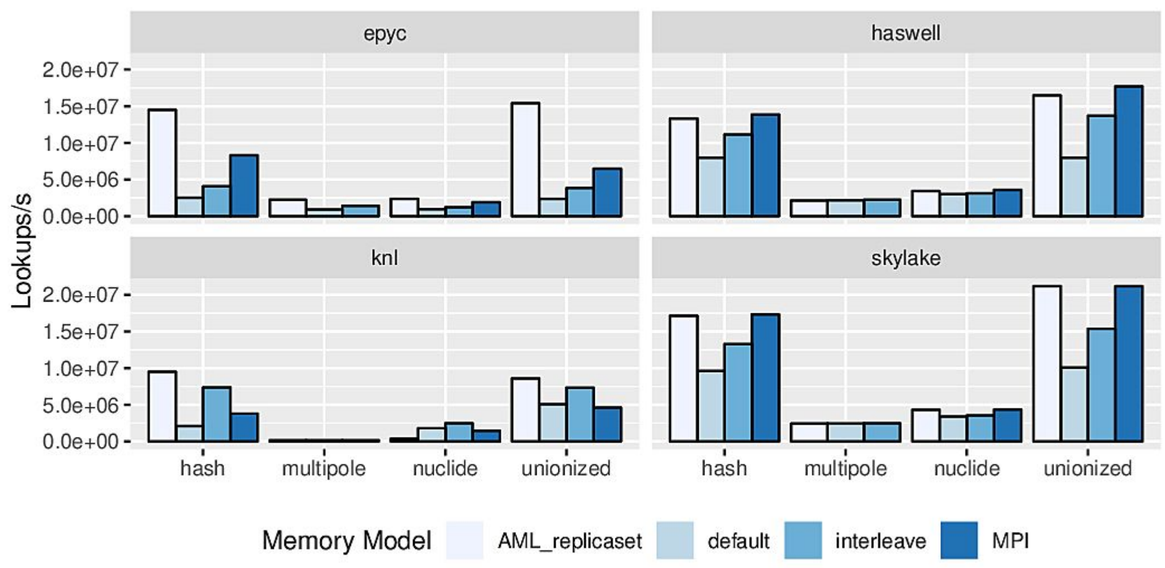
Key Components

- 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 across the machine.



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.