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 a managed scratchpad for high-bandwidth memory (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
- Register virtual address ranges 'umap regions' by userfault64
- Resolves page faults in regions by fetching/flushing data from datastores following user-defined policies
- Customized page sizes, buffer size, etc. in application-context

Features
- User space flexibility, ease of use, self-defined handler (multiple files, distributed data sets)
- Portable on new systems, no system-wide impact on other applications sharing the same system

Applications
- Out-of-core Graph Algorithms
  - Perform level-synchronous breadth-first search on an R-MAT scale-31 CSR graph (529 GB)
  - Varied UMap page sizes from 4 KB to 64 MB and outperformed mmap by up to 1.8x
- Asteroid Detection Algorithms
  - Process intermittent time-series telescope data to find Near-Earth Asteroids (NEAs)
  - UMap combines multiple 2D tiled images into a 3D volume for search
  - Comparable performance of processing 512GB images on local SSD and Lustre datastores
- NVM databases
  - Use a 384 GB persistent memory pool on the local NVMe-SSD
  - Run YCSB benchmark varying concurrency

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).
- Ongoing discussions with RIKEN, INRIA, BSC

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

Layout Management for Linear Algebra
- 5-level topology-aware DGEMM
  - KNL in flat/quad mode
  - Hierarchical Tilings:
    - Pipeline over MCDRAM-size tiles
    - On-the-fly transform for better performance
    - OpenMP for inner parallelism
    - Cache, register file-specific tilings
    - Inner autotuned microkernel for AVX512 performance

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