MemSpan: Toward Efficient Data Placement in Heterogeneous Memory Systems
Thaleia-Dimitra Doudali, Ada Gavrilovska
{tdoudali, ada}@cc.gatech.edu

1. Motivation

- Big Data and HPC applications require fast processing speeds.
- Commodity hardware (CPU, DRAM, Disk) face cost and scale issues.
- New hardware technologies emerge.
  ▶ Processing Units
  ▶ Memory Units
  ▶ Storage Units
- New hardware technologies will operate together with the predominant ones.
  ▶ Heterogeneous systems is the new norm.

2. Problem Statement

- The memory substrate will couple small amounts of “fast” access units, with larger portions of “slower” access units.
- Data-intensive applications will span their dataset across all available memory units.
- Need to cleverly map application’s data across the different memory components.
  ▶ Critical to performance data should be in fast memory.
- Need for an OS-level solution.
  ▶ No changes needed in the source code.
  ▶ Can work for any type of application.

3. Existing Solutions

- Explicit API to allocate memory on the new units.
  ▶ Needs programming efforts.
- Application profiling tools. [1] [2]
  ▶ Offline run: keeps track of number of accesses on a data structure / object granularity.
  ▶ cost = number of accesses per byte.
  ▶ Order data objects according to cost.
  ▶ Online run: places data objects with high cost in fast memory, until capacity is full.

4. Observations

1. Not all applications are sensitive to the presence of slower memory.

2. Not all the data structures of an application are critical to performance.

5. Proposal

- Build a tool that can on-the-fly:
  ▶ Identify if an application overall is sensitive to slow memory.
  ▶ Identify which data structures are critical to performance.
  ▶ Place in slow memory the non-sensitive components.
  ▶ Prioritize the placement of data objects with high cost in fast memory, until capacity is full, similarly to existing work.

- In this way, we can:
  ▶ Leverage the existence of slow memory.
  ▶ Make existing solutions simpler / faster; can become practical for dynamic workloads.
  ▶ Provide fairness or SLA guarantees in multi-tenant environments, where applications will compete for the available fast memory.