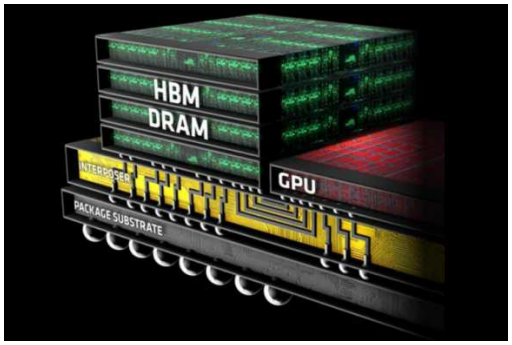
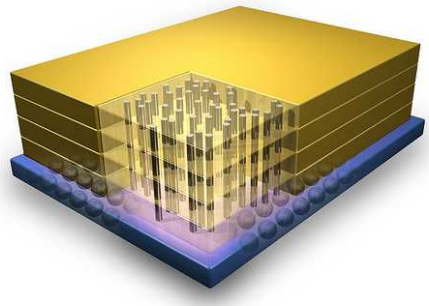
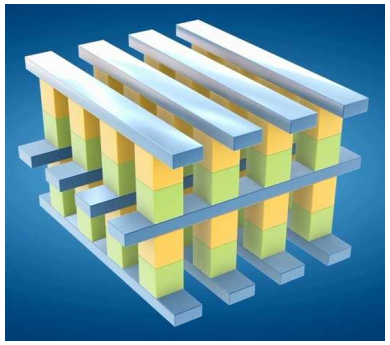
SAND2017-1143PE





NNSA Applications and Multi-level Memory

Using Architectural Simulation to Predict NNSA Application Performance on Future Multi-Level Memory Systems

Gwen Voskuilen, **Arun Rodrigues**, Mike Frank, Si Hammond

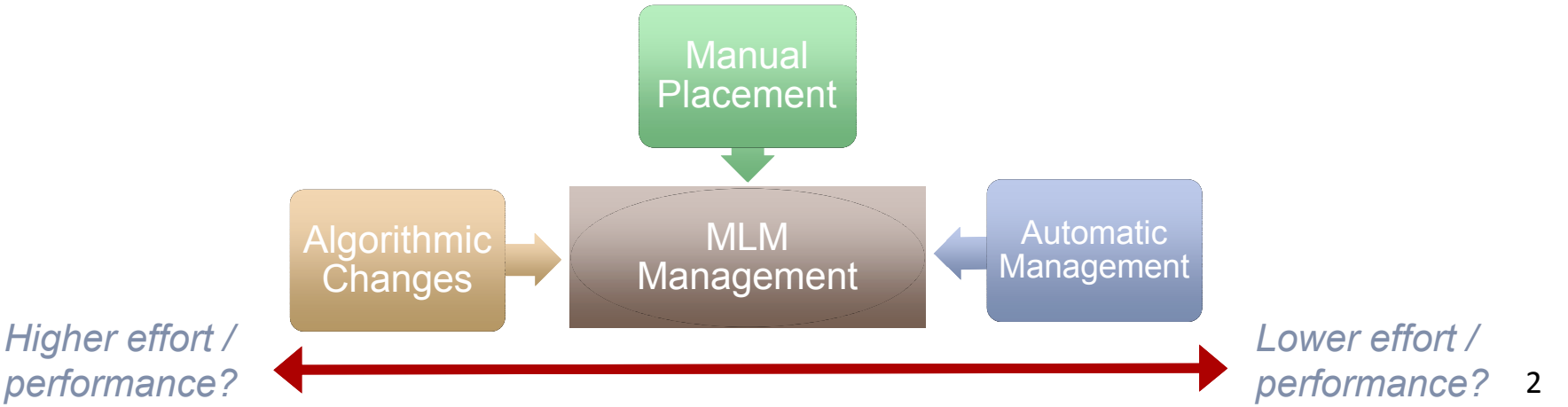
JOWOG 34; Feb 8, 2017

U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

Future Systems: Multi-level Memory

Sandia National Laboratories

- Future memory systems will integrate multiple levels (types) of memory (MLM)
  - E.g., Trinity KNL with DDR DRAM and MCDRAM
  - How to place data in memory system to maximize application performance?
    - Place all in MCDRAM: maximizes bandwidth but limits capacity
    - Place all in DDR DRAM: maximizes capacity, but limits bandwidth
    - Place some in each – how to decide what goes where? Who decides?



Analyzing application performance

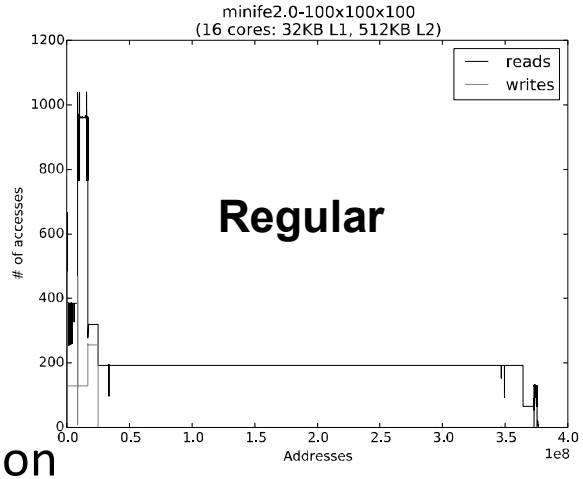
Sandia National Laboratories

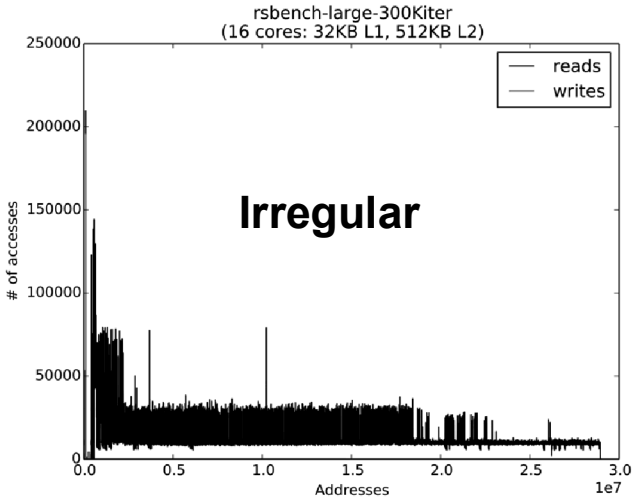
- Predict how NNSA proxy applications will perform on hardware
  - Analyze memory usage characteristics
  - Explore performance as technology parameters change
  - Evaluate policies for automatic and manual data placement

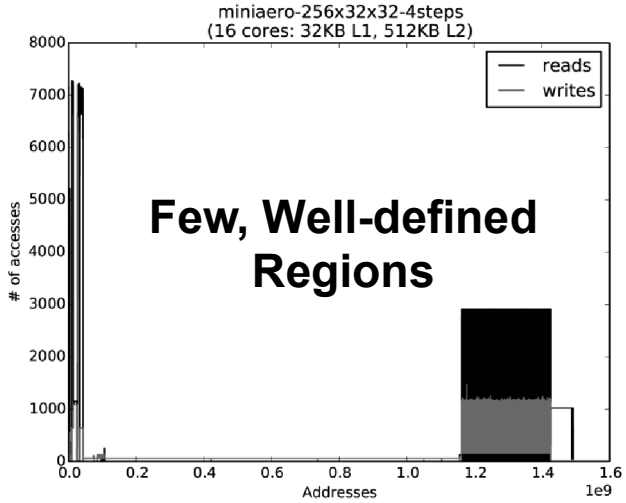
- Proxy apps: HPCG, SNAP, PENNANT, MiniPIC

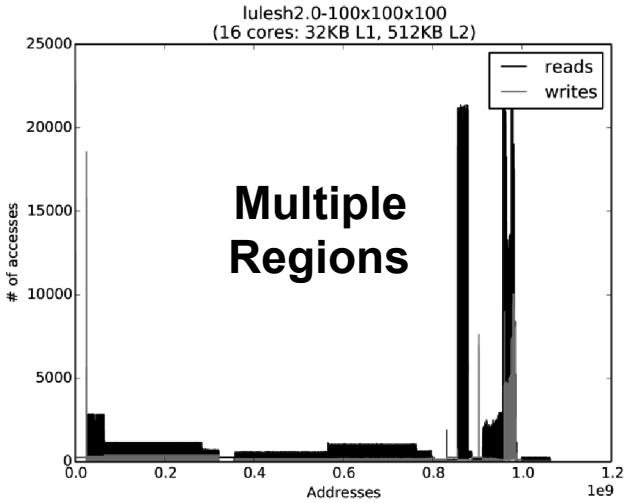
3 Paths

- Algorithmic
- Manual (Malloc)
  - MemSieve Tool
  - Application Modification
- Automatic (Paged)
  - Analysis
  - Policies
  - Performance
  - Recommendations



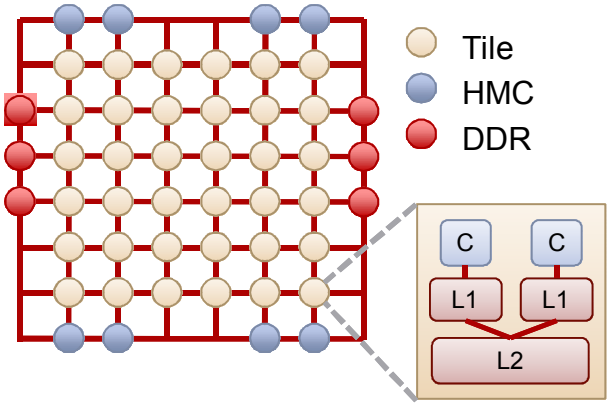


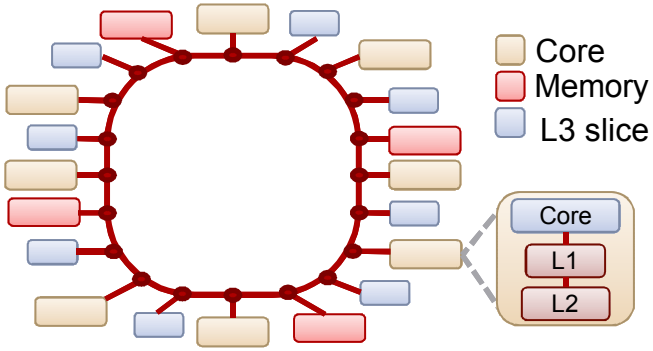




Analyzing application performance

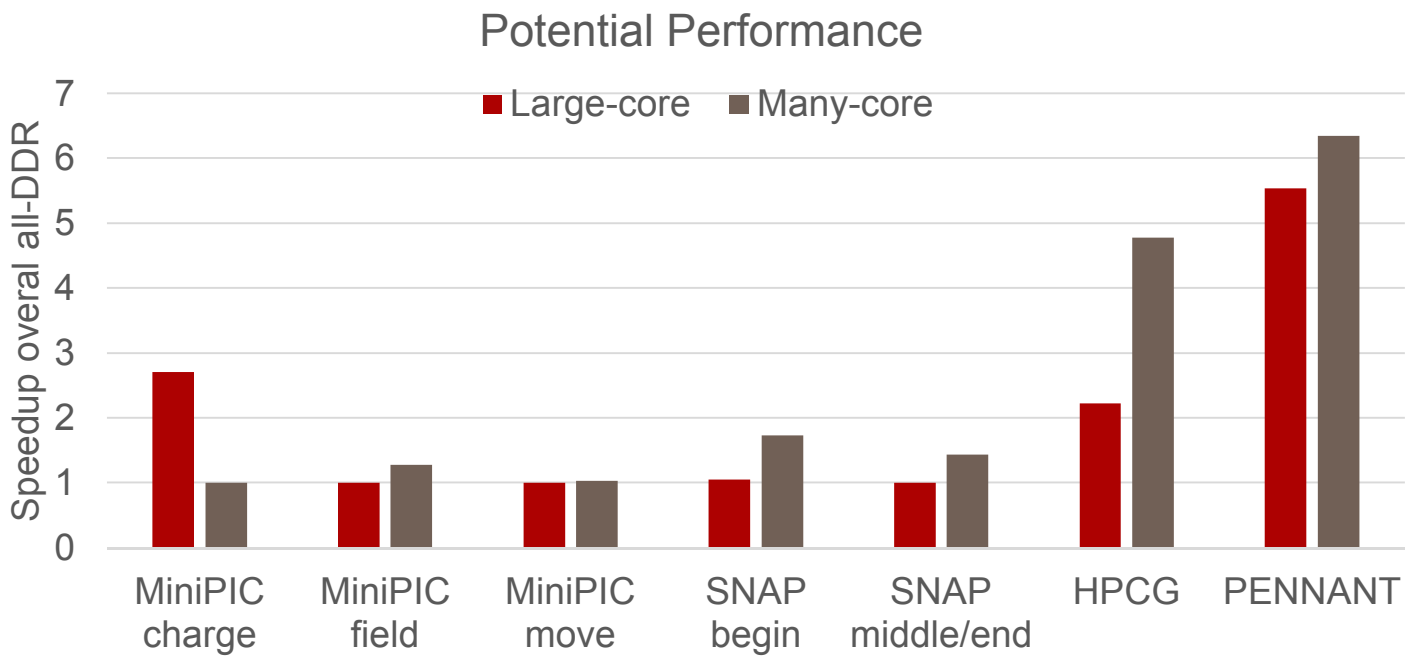
- Used the Structural Simulation Toolkit (SST) to analyze:
  - Memory behavior: *MemSieve*
    - Identifies which application data structures use disproportionate memory bandwidth
    - And behavior over time
  - Performance
    - Many (small)-core vs. few (large)-core Hardware caching policies for HBM
      - When to move data to cache
      - Which data to evict
    - Software allocation policies
      - Page vs malloc granularity
      - Static vs dynamic



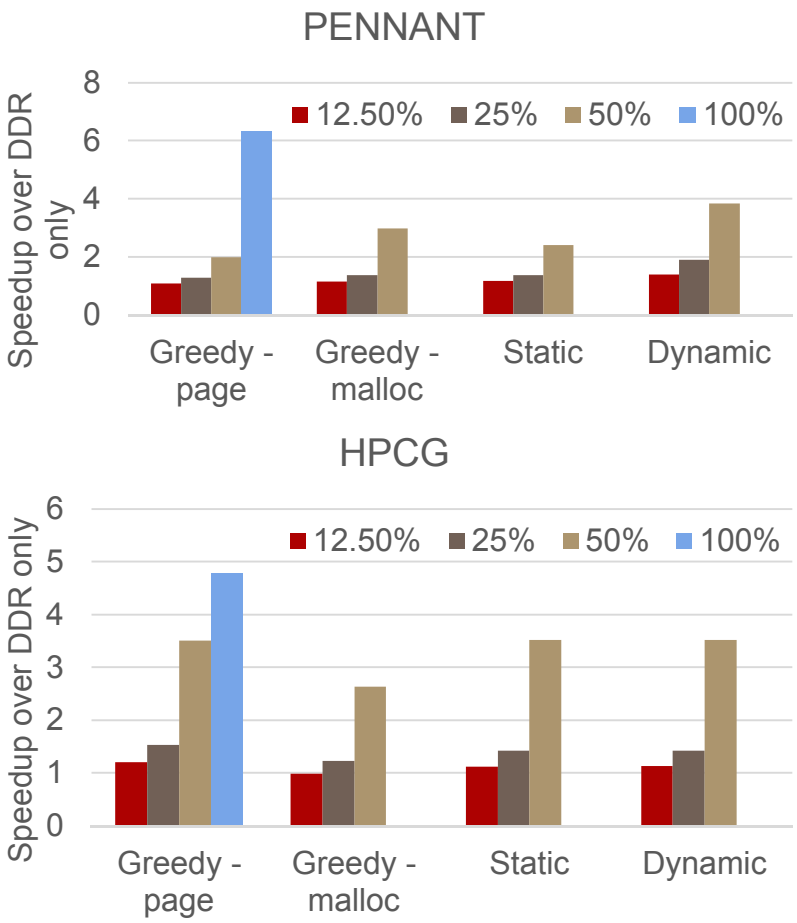


Potential Performance

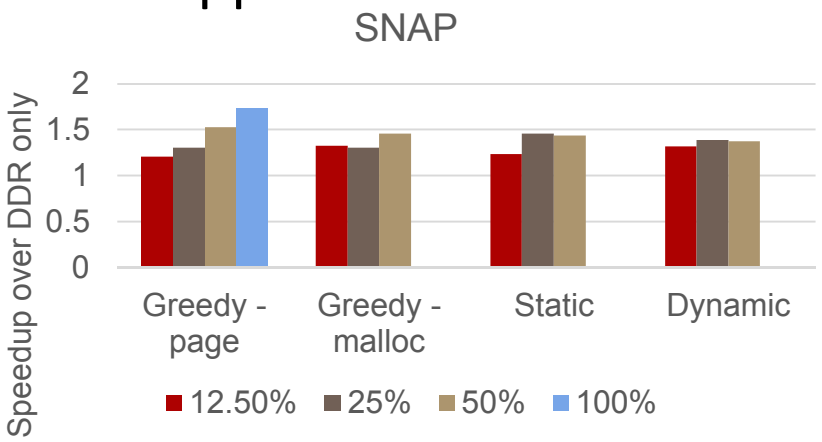
- Many apps show great performance *if application fits in HBM*
  - Harder case: Use both together



Software-managed data allocation



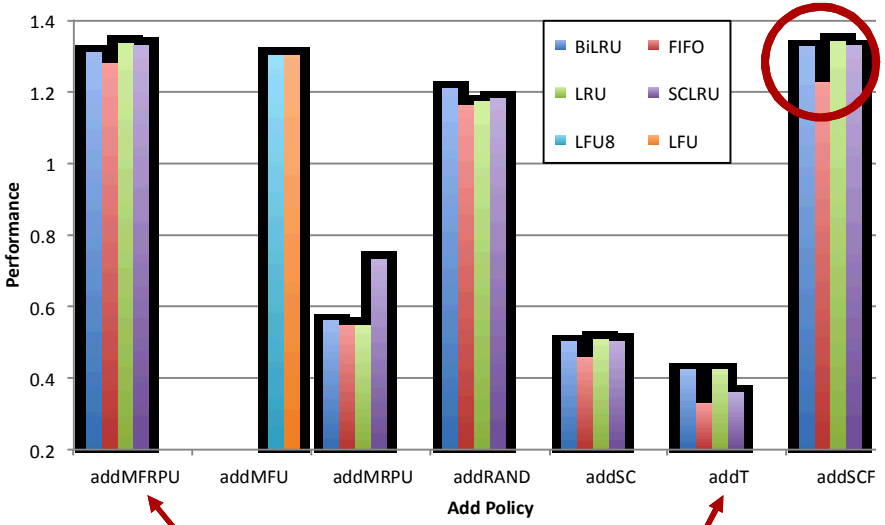
- Greedy policies do fairly well
  - But may not extend to larger apps
- Dynamic migration necessary for some apps

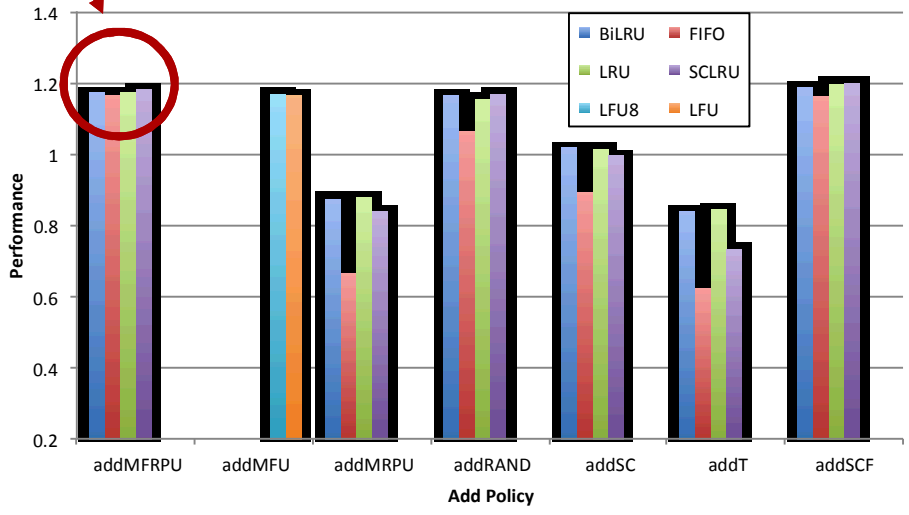


Caching: hardware-driven allocation

Sandia National Laboratories

Replacement policy: little variation



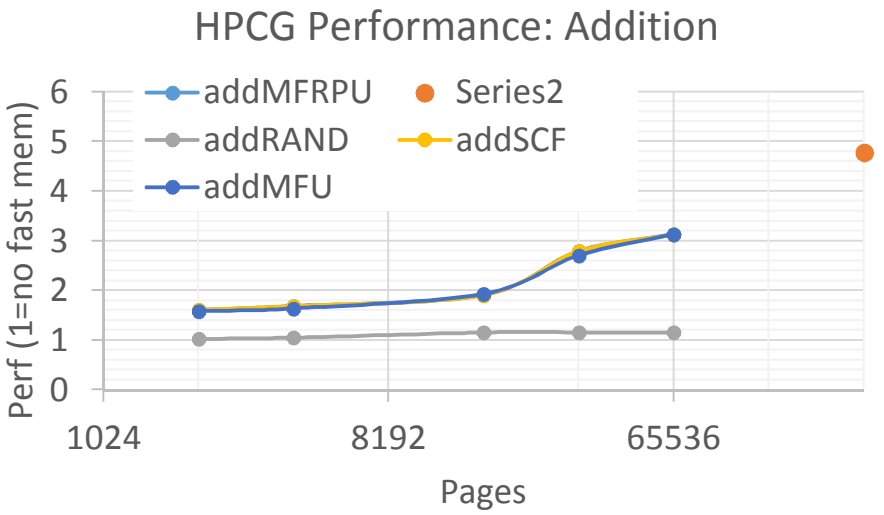


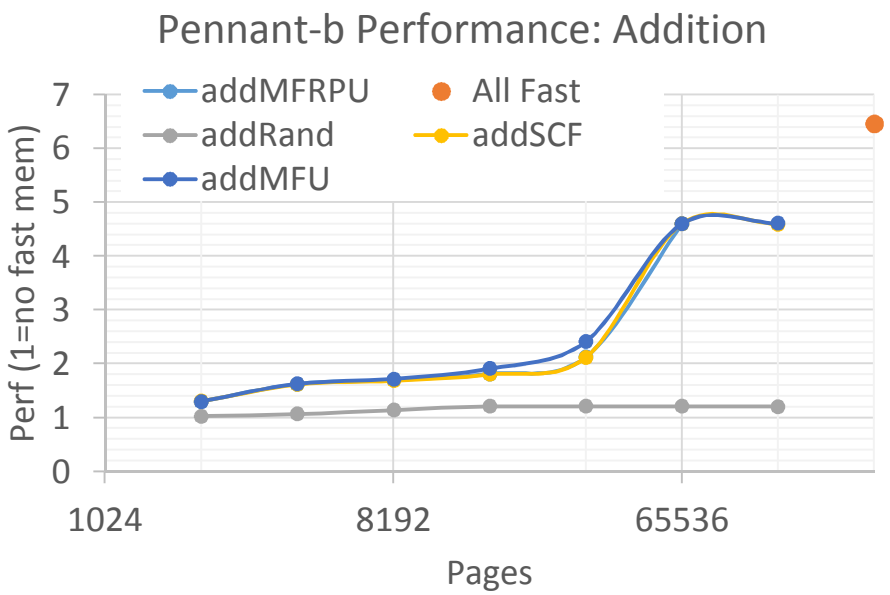
Addition policy: big variation

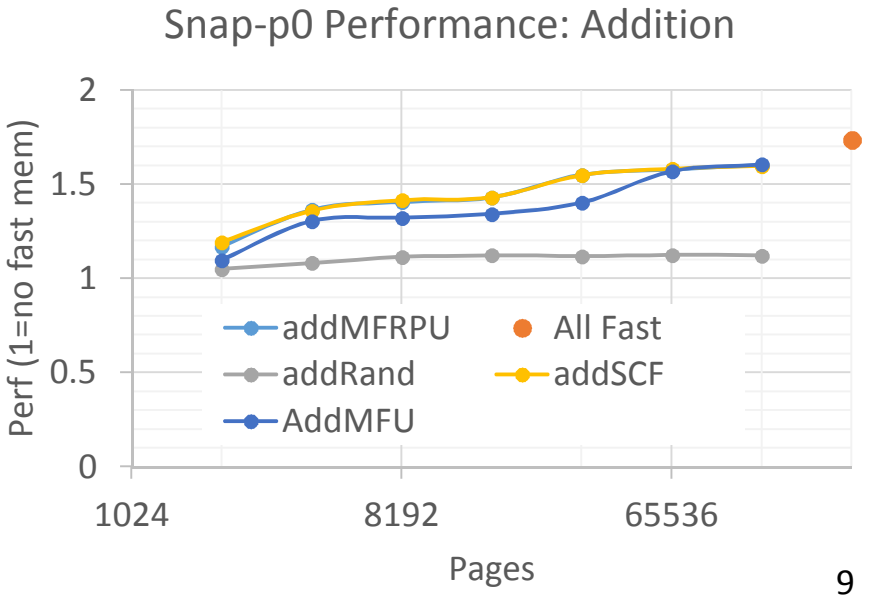
"What you put in matters more than what you take out"

Performance on larger data sets

- Looked at highest performing addition policies
  - Variants of most-frequently used
  - Baseline: random
  - LRU replacement

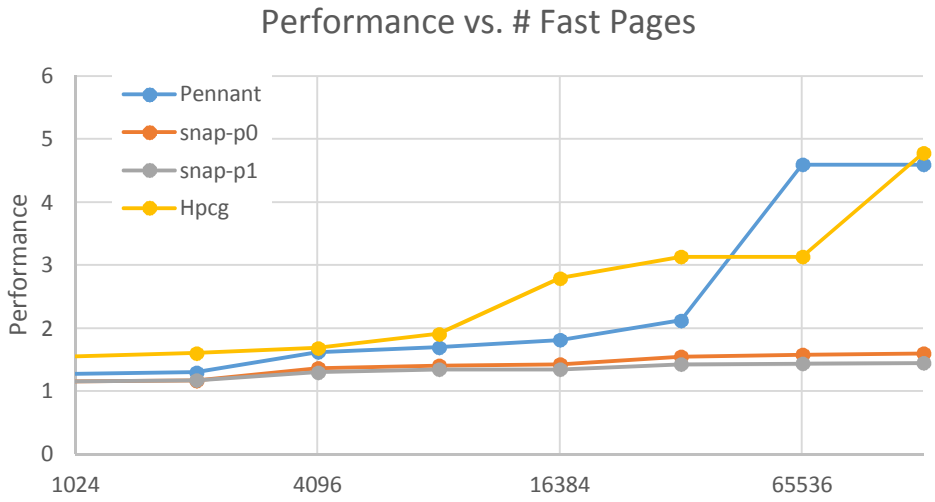


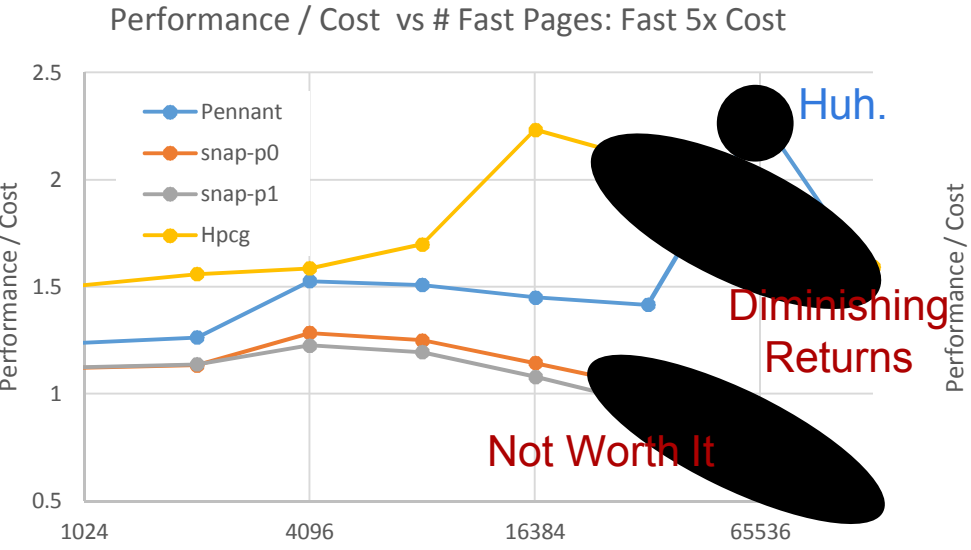


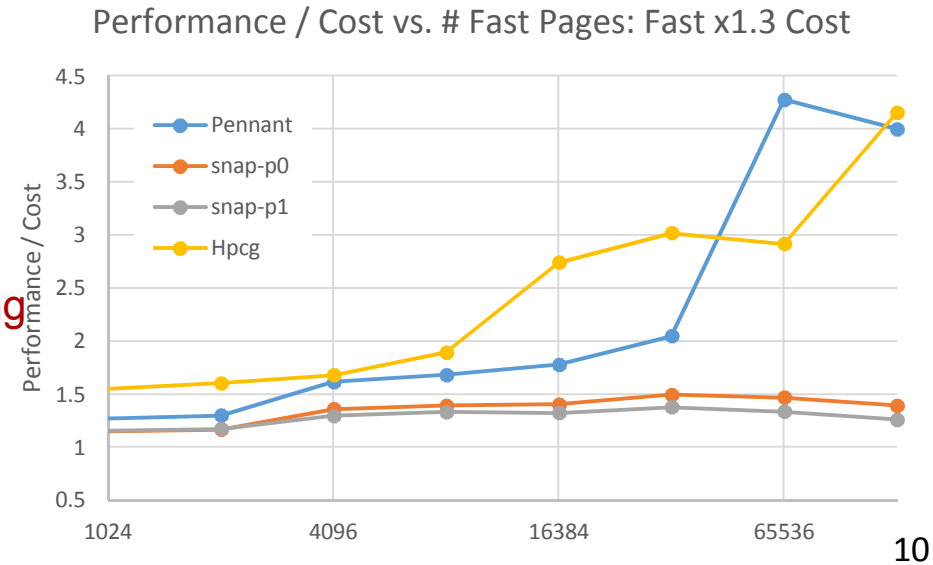


Cost & Performance

- Ultimate FoM
- 40-380% performance improvement
- Will Cost Kill it?
- Recommendations for HW







Conclusion

- Manual management feasible
  - But bandwidth-bound codes will require dynamic migration
  - New tool, MemSieve, aids manual management
- Overall, automatic management is comparable to manual
  - Varies by application so “one-size” does NOT fit all
- Some apps will need algorithmic changes to better use HBM/HMC
  - MiniPIC, SNAP