Analyzing Complex Memory Systems

UNIVERSITY OF NOTRE DAME

Neil Butcher
Notre Dame,
South Bend Indiana

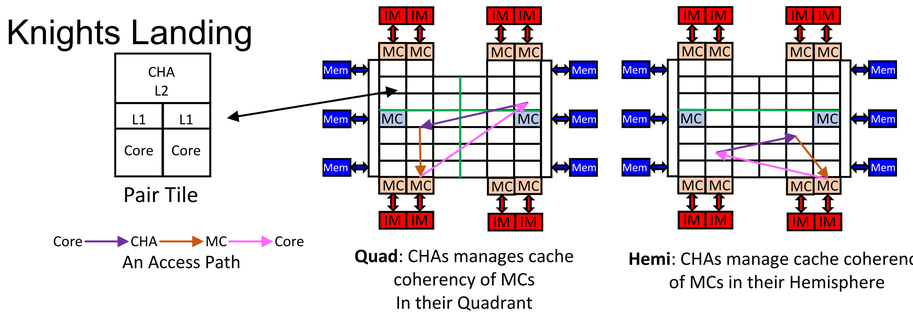
Motivation

Problem: HPC chips are utilizing more complex memory systems to match increase in core.

This work: Improves understanding of relationship between memory system and performance of memory-sensitive problems

Testbed: Intel Knights Landing (KNL)

Architectural Background



Quad: CHAs manages cache coherency of MCs In their Quadrant

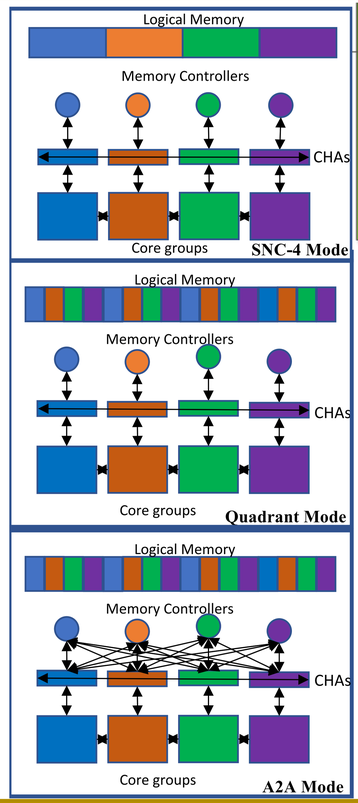
Hemi: CHAs manage cache coherency of MCs in their Hemisphere

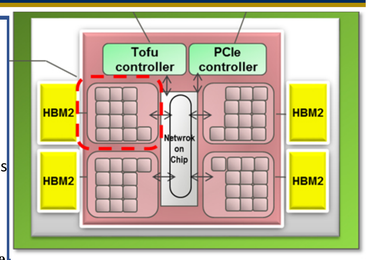
ATA: CHAs can manage cache coherency of any MCs

SNC-4: CHAs manage cache coherence of MCs in their Quadrant
Accesses across Quadrants are expensive

SNC-2: CHAs manage cache coherence of MCs in their Hemisphere
Accesses across Hemispheres are expensive

64 Cores, 6 memory channels
Configurable intermediate memory of 16GiB MCDRAM (3D Stack DDR)
Cache Home Agent (CHA) – 1 per tile, data is assigned to CHA, CHA manages coherency of data
Clustering Mode is defines the affinity between CHAs and logical and physical memory





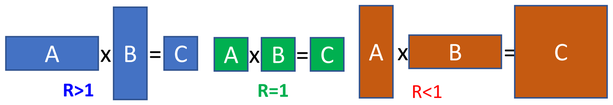
Logical Memory – How memory is ordered in the view of a programmer

Physical memory – Memory channels are associated with a portion of physical memory

Approach

To gain complete understanding of impact of memory system we vary:

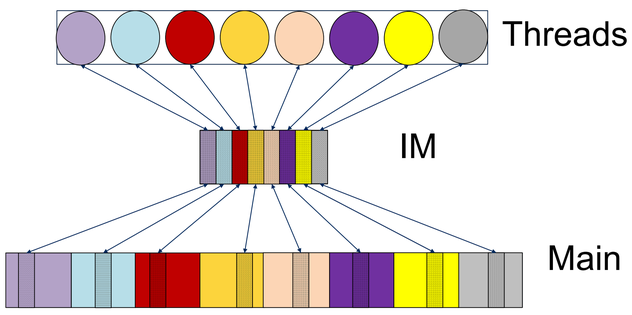
- **Problem –Sort, Matrix Multiply, FFT**
- **Algorithm – State of the art, Chunking, Cache Oblivious**
- **Clustering mode – KNL**
- **Problem Size – 1B - 6B doubles**
- **Problem Ratio – 1/512-512/1**



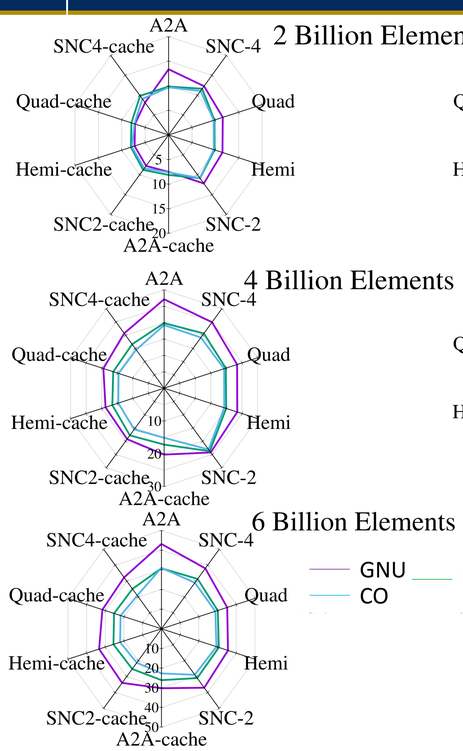
Radar plots represent execution time
In addition to S.O.A. codes we include two types of algorithms:

Chunking
Bring subproblems into fast memory one at a time and solve them using all cores/processors

Cache Oblivious
Efficient use of cache without explicit knowledge of cache size
Divide-and-conquer strategy solved depth-first
Recombining subproblems makes efficient use of caches



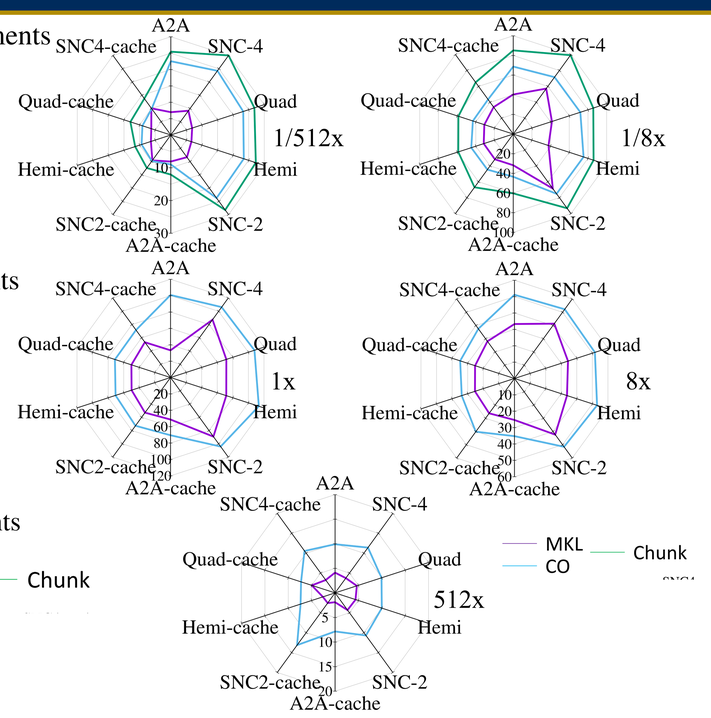
Sort



Observations:

- Benefits from IM
- Clustering mode is largely unimportant
- A2A performs significantly worse then other modes

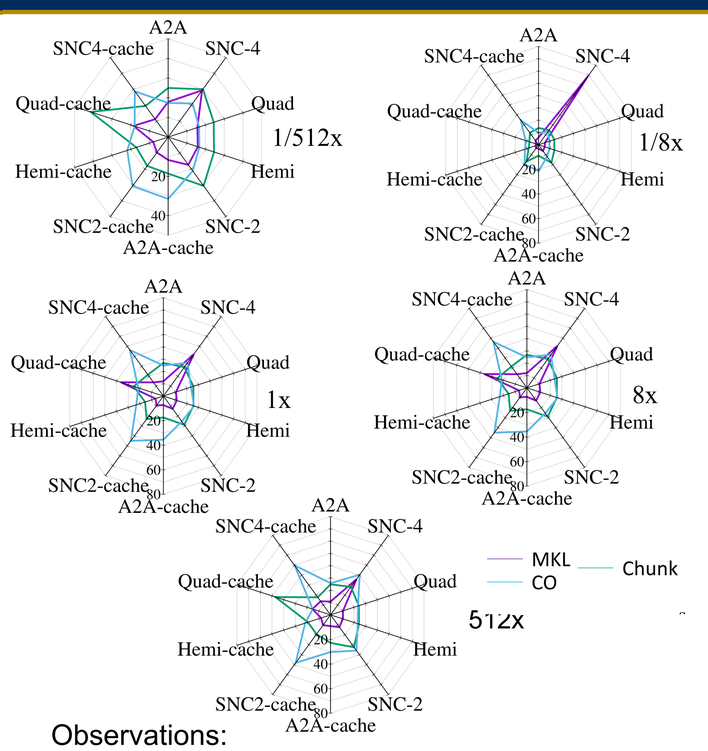
Matrix Multiply



Observations:

- Clustering mode impacts performance, especially at small ratios
- IM is most useful at small ratios, becomes more unpredictable at larger ratios

FFT



Observations:

- Performance is erratic across modes
- MKL has best performance in the best mode, but also has worst performance in some modes

Conclusions

The benefit of IM changes with clustering mode and algorithm
CO Sort – 1.55 - 1.78x MKL Matrix-multiply – 0.64 - 1.68x MKL FFT – 0.05 - 16x
The impact of the clustering mode is most pronounced with irregular access patterns
Large ratio Matrix multiply MKL 0.94 – 2.2x
Square FFT -- 1.8 – 16.4x

Cache Oblivious algorithms port effectively across clustering modes (fastest/slowest)
CO Sort – 1.08-1.24x Matrix Multiply – 1.06 – 1.28 FFT – 1.6 – 2.4x
Clustering modes supported in KNL are not ideal for chunking algorithms
Chunking algorithms are 1.5x-4.0x slower