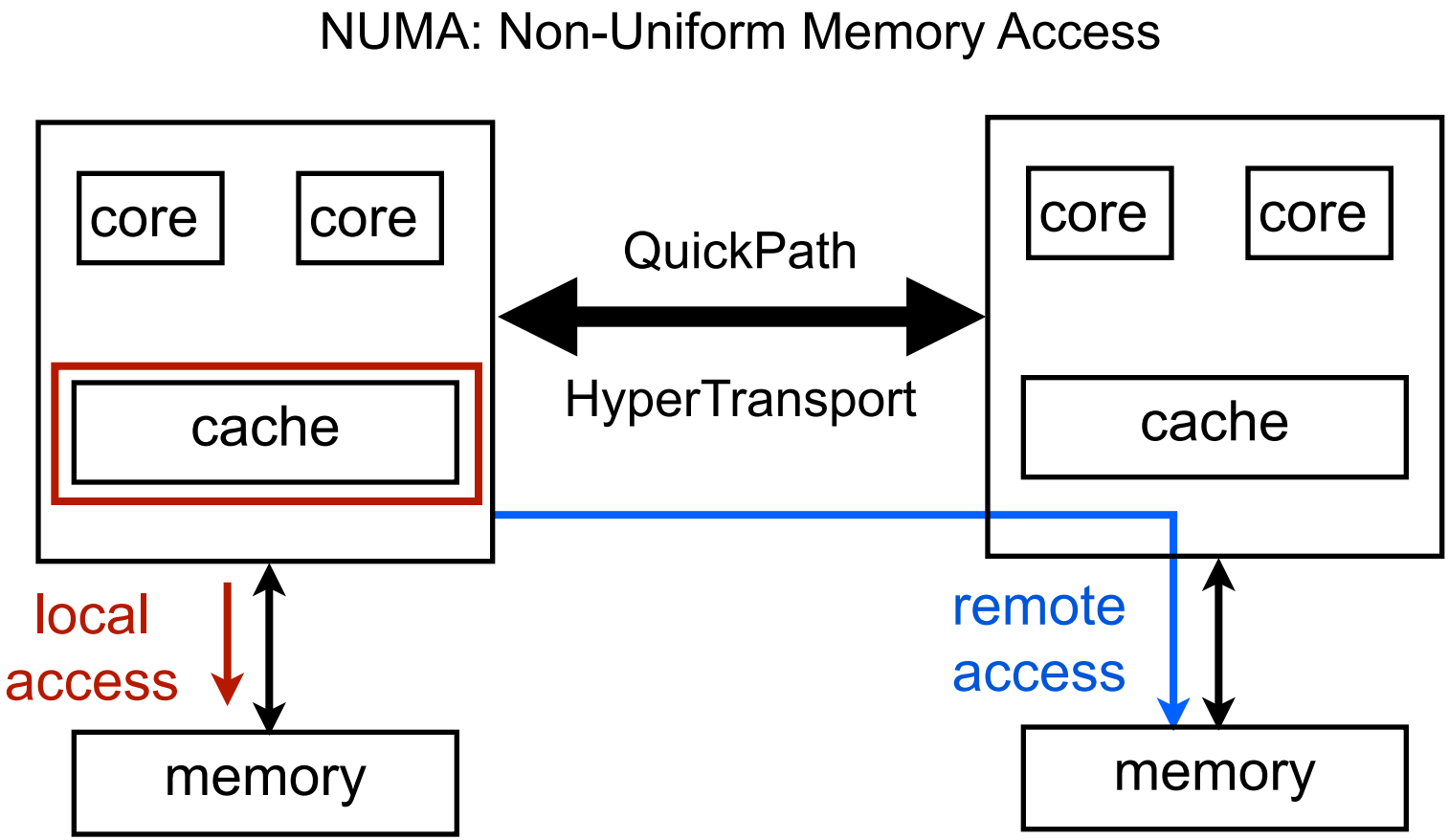
Leveraging Hardware Address Sampling

Beyond Data Collection and Attribution

Xu Liu

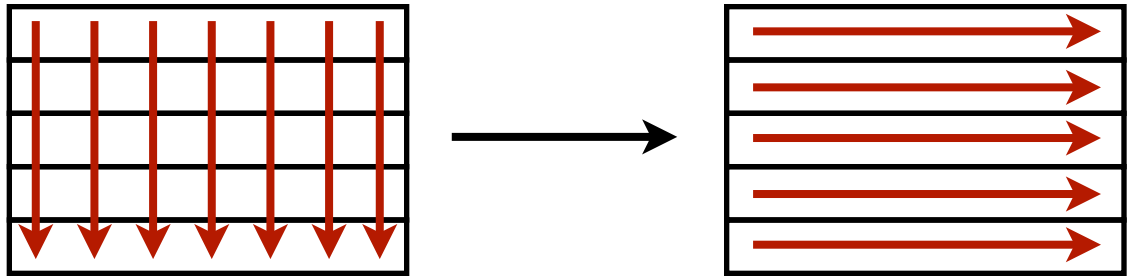
Department of Computer Science
College of William and Mary
xl10@cs.wm.edu

Motivation: Memory is the Bottleneck



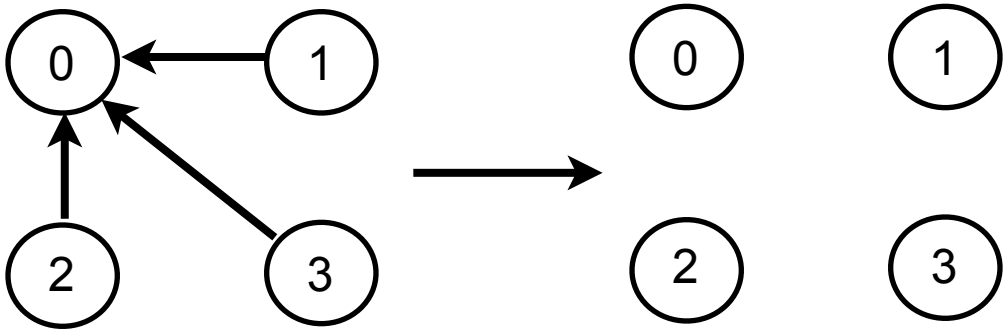
Memory Bottleneck Optimization

spatial locality



temporal locality

NUMA locality



State of the Arts

simulation methods

deep insights

weaknesses:

- 2-5x overhead
- not real machines

low overhead with deep insights

deep insights with low overhead

measurement methods

low overhead

Hardware Address Sampling

- Features of address sampling
  - necessary features
    - sample memory-related events (memory accesses, NUMA events)
    - capture effective addresses
    - record precise IP of sampled instructions or events
  - optional features
    - record useful metrics: data access latency (in CPU cycle)
    - sample instructions/events not related to memory
- Support in modern processors
    - AMD Opteron 10h and above: instruction-based sampling (IBS)
    - IBM POWER 5 and above: marked event sampling (MRK)
    - Intel Itanium 2: data event address register sampling (DEAR)
    - Intel Pentium 4 and above: precise event based sampling (PEBS)
    - Intel Nehalem and above: PEBS with load latency (PEBS-LL)

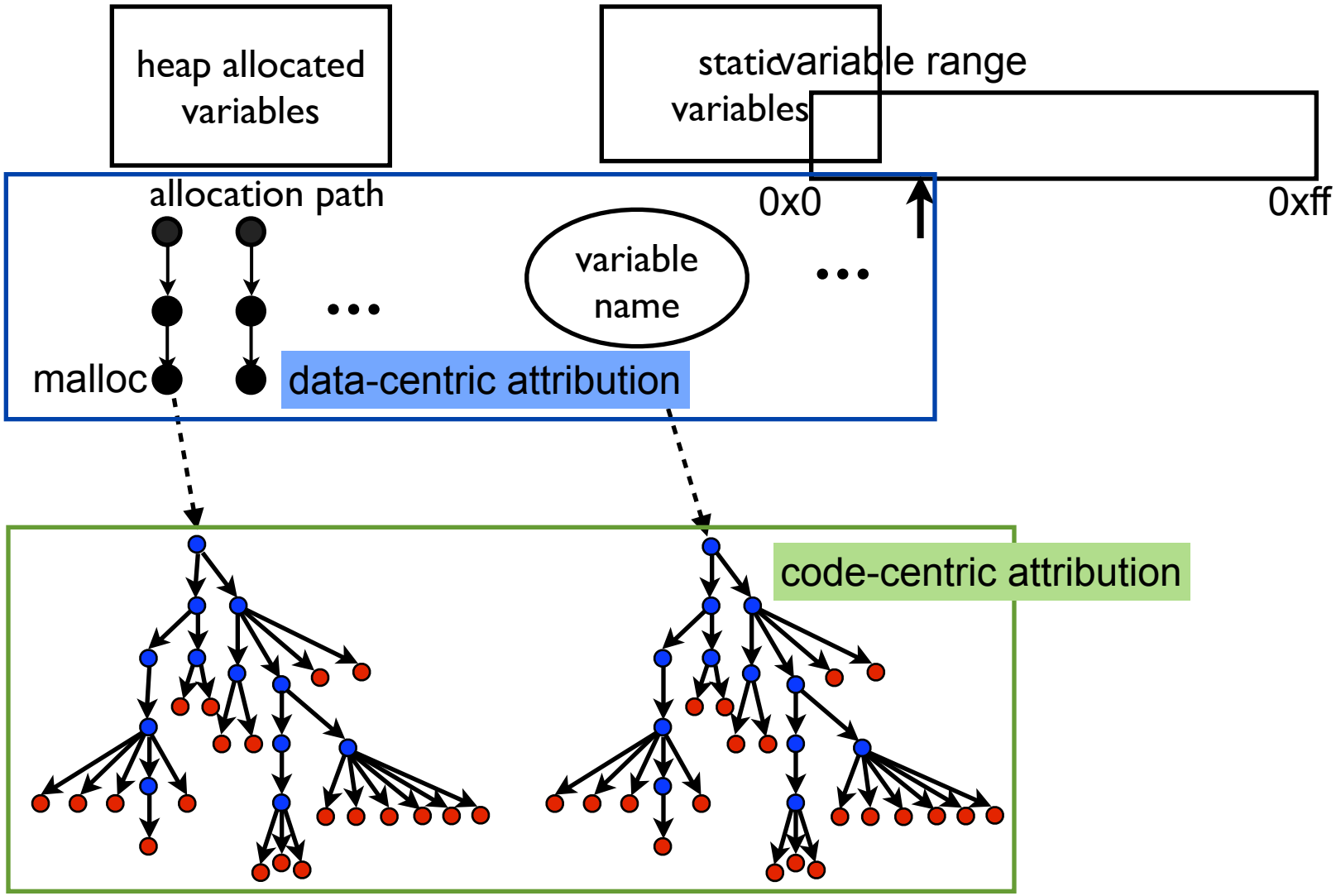
Tools Based on Address Sampling

- Measurement methods
  - temporal/spatial locality
    - HPCToolkit, Cache Scope
  - NUMA locality
    - Memphis, MemProf, HPCToolkit
- Features
  - lightweight performance data collection
  - efficient performance data attribution
    - code-centric attribution
    - data-centric attribution

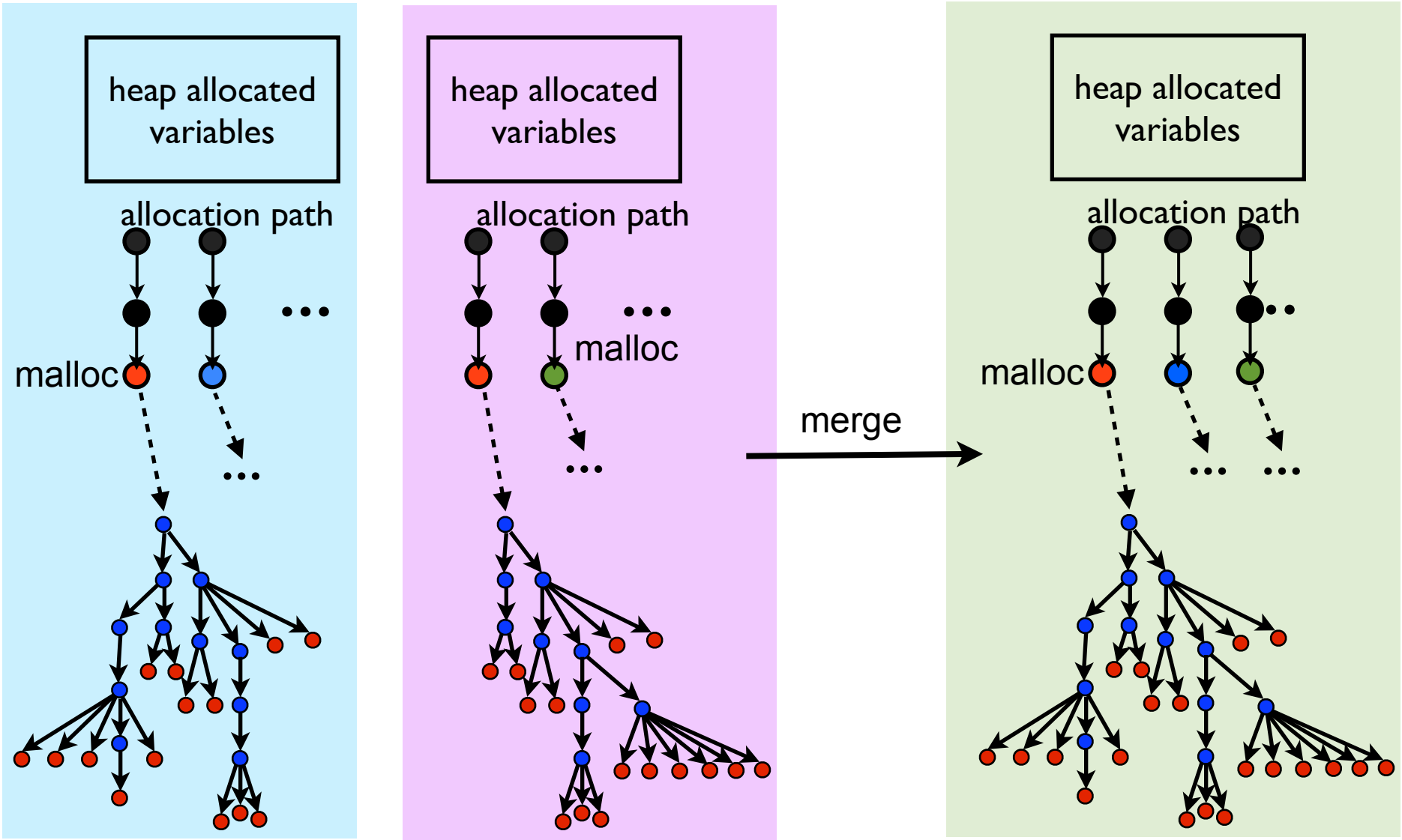
Take HPCToolkit for example

"A Data-centric Profiler for Parallel Programs". Liu and Mellor-Crummey, SC'13

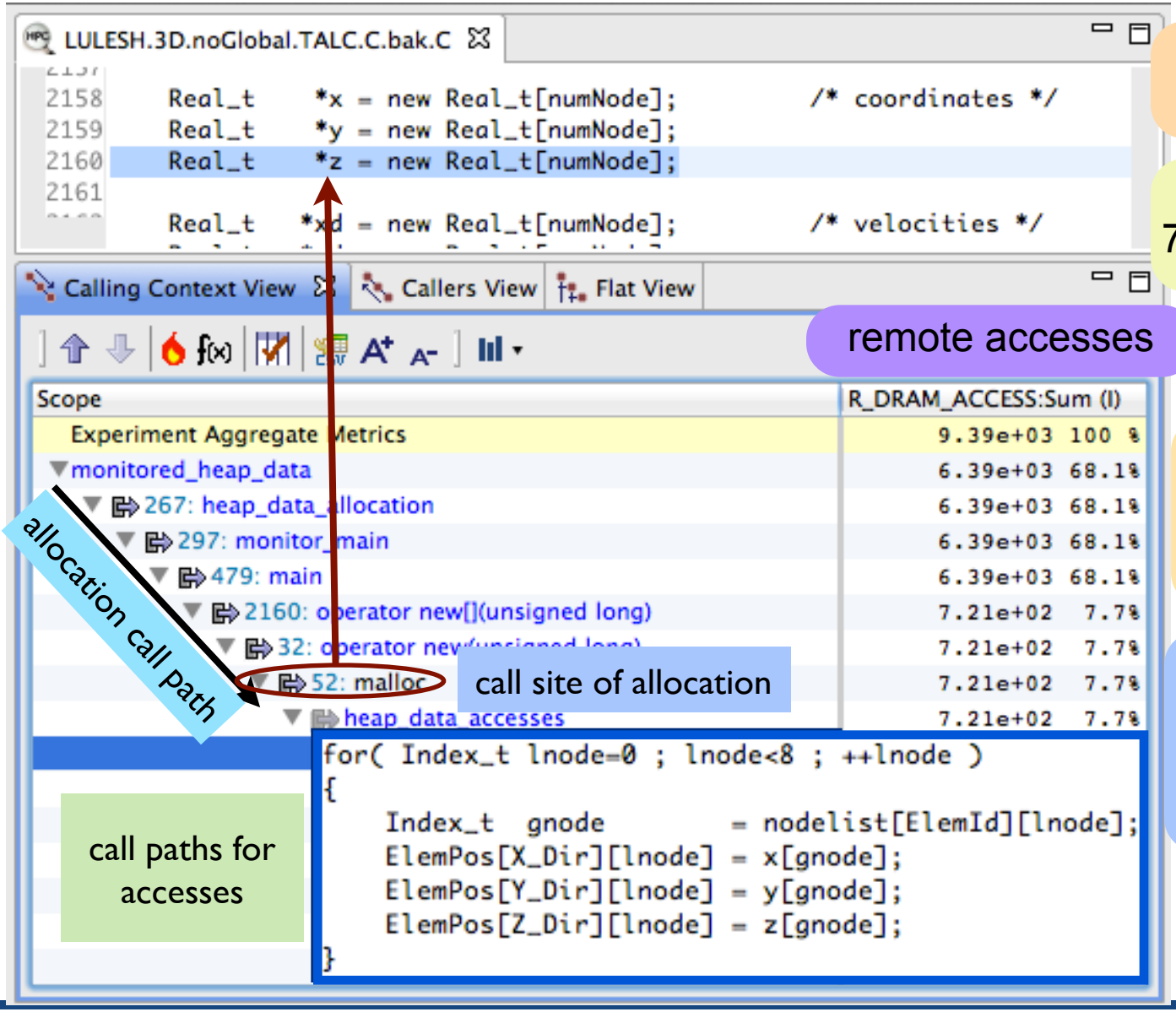
HPCToolkit: Attributing Samples



HPCToolkit: Aggregating Profiles



LULESH on Platform of 8 NUMA Domains



heap data:68% remote accesses

z accounts for 7.7% remote accesses

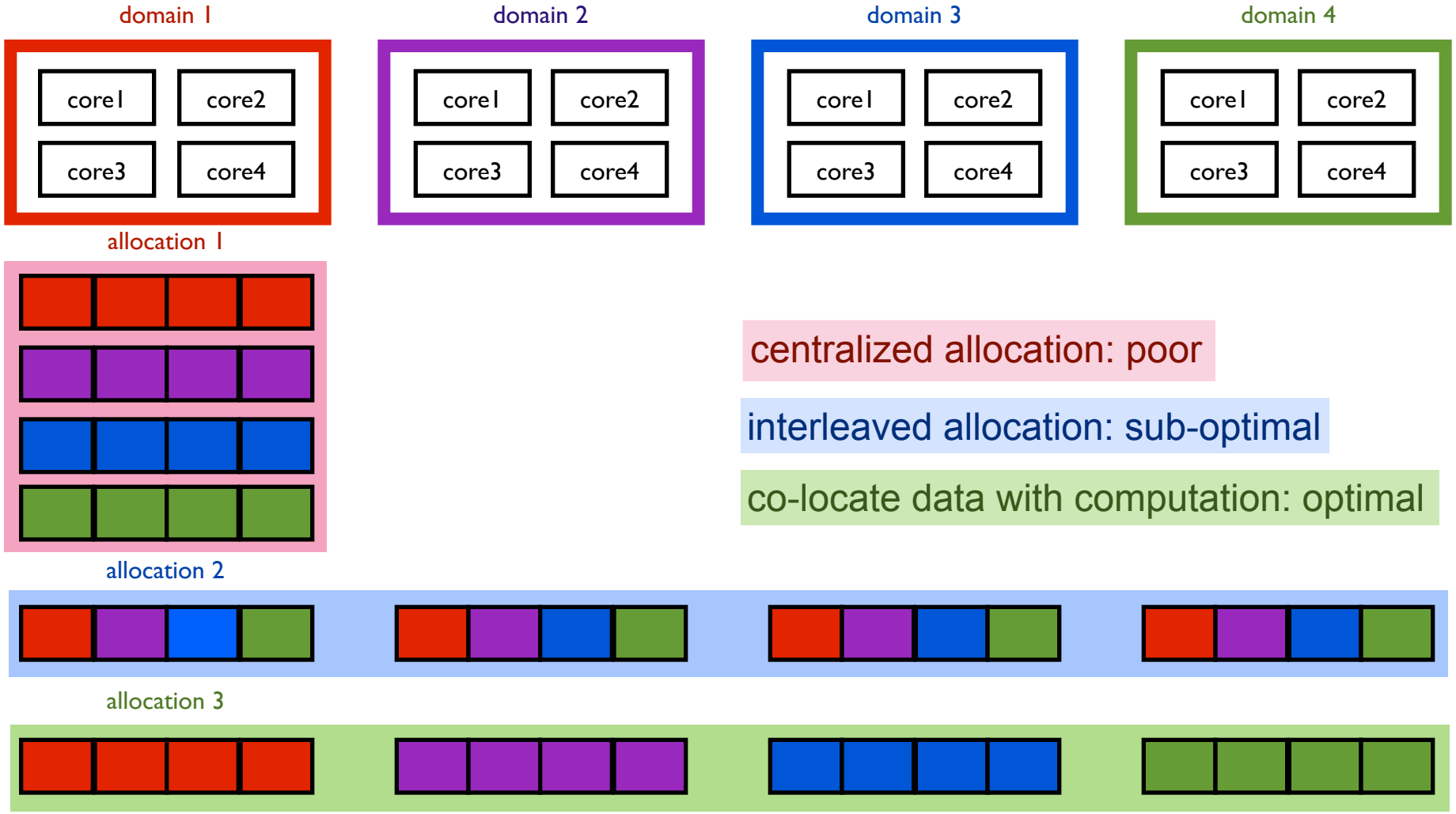
z is allocated in a NUMA domain but accessed by others

interleave pages of z across NUMA nodes 13% improvement in running time

Existing Measurement is Inadequate

- Data collection + attribution ≠ optimal optimization
  - know problematic data objects but not know why
  - need more insights for optimization guidance
  - challenges in data analysis
    - not monitoring continuous memory accesses
- Approaches: data analysis for detailed optimization guidance
  - NUMA locality
    - offline optimization (PPoPP'14)
    - online optimization
  - cache locality
    - array regrouping (PACT'14)
    - structure splitting
    - locality optimization between SMT threads
  - scalability of memory accesses

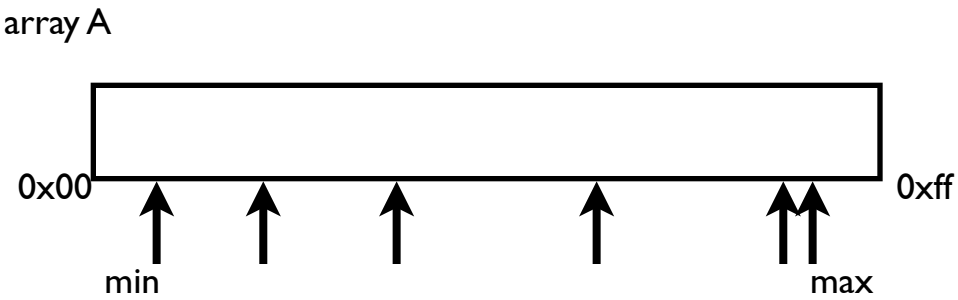
Interleaved Allocation is NOT Always Best



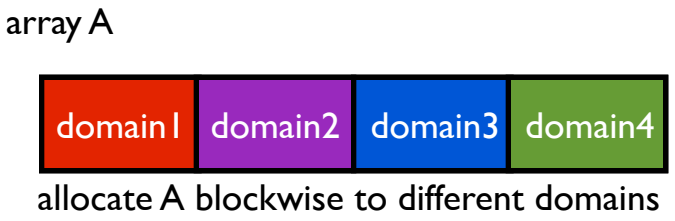
Goal: identify the best data distribution for a program

Memory Access Pattern Analysis

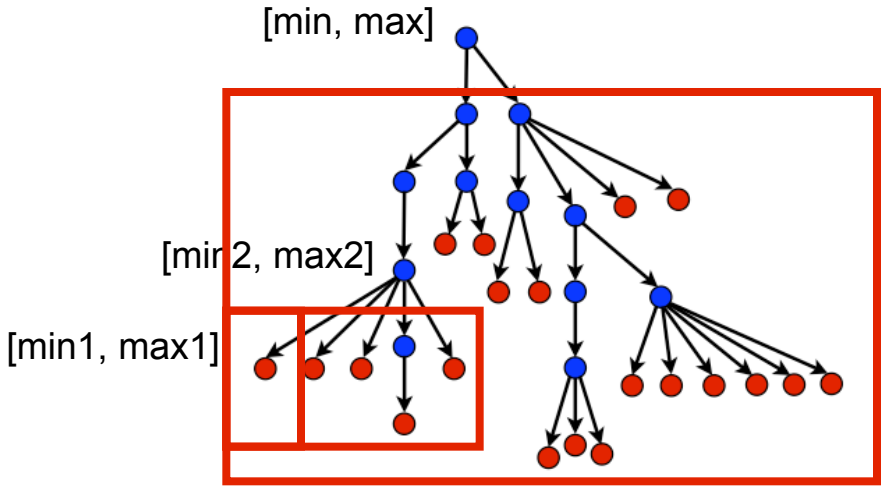
- Online data collection

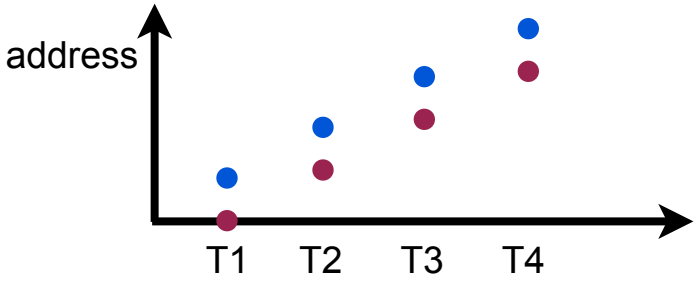


[min, max] per sampled memory access



allocate A blockwise to different domains

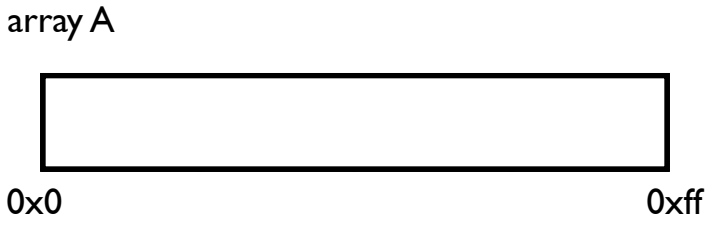


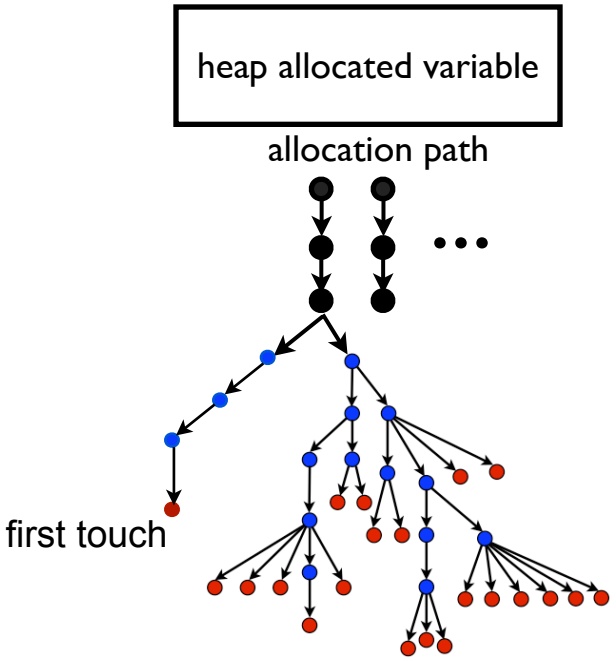


balanced allocation + maximum locality

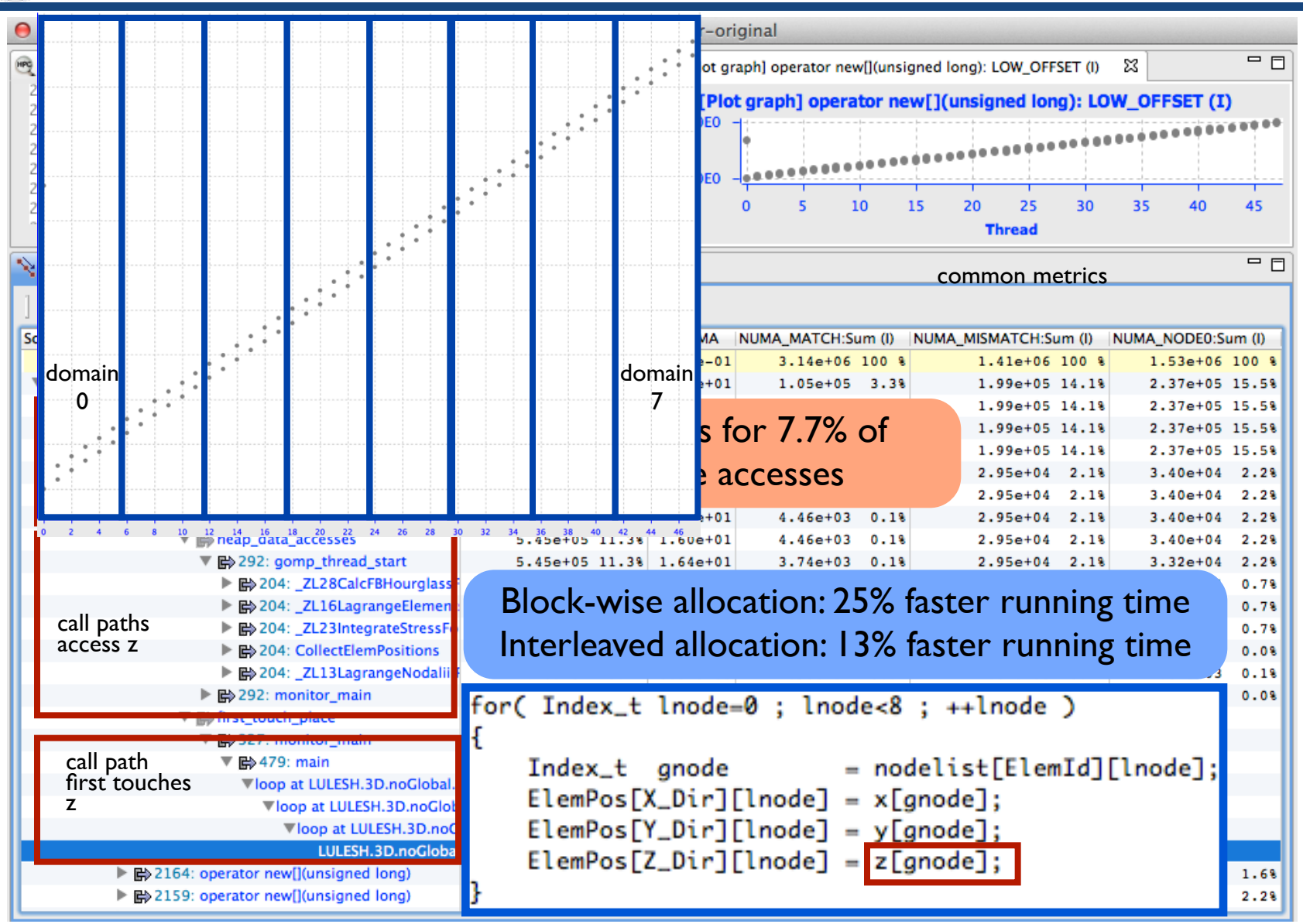
Pinpointing First Touch

- Linux “first touch” policy
  - memory allocation at first touch
  - if T1 first touches the whole range of A
  - if threads touch different segments of A





LULESH on Platform of 8 NUMA Domains



Experiments: Architectures & Applications

Architectures

Sampling mechanisms		Processors	Threads
Instruction-based sampling	IBS	AMD Magny-Cours	48
Marked event sampling	MRK	IBM POWER 7	128
Precise event-based sampling	PEBS	Intel Xeon Harpertown	8
Data event address registers	DEAR	Intel Itanium 2	8
PEBS with load latency	PEBS-LL	Intel Ivy Bridge	8

Benchmarks

LLNL	LANL	Rodinia	PARSEC	SNL
AMG2006	Sweep3D	Streamcluster	Blackscholes	S3D
LULESH		NW		
Sphot				
UMT2013			optimized benchmarks	

Optimization Results

Programs	Optimization	Improvement for execution time
AMG2006	NUMA locality	51% for the solver
Sweep3D	spatial locality	15%
LULESH	spatial+NUMA locality	25%
Streamcluster	NUMA locality	28%
NW	NUMA locality	53%
UMT2013	NUMA locality	7%

Measurement Overhead

Code- & data-centric analysis on POWER7 and Opteron

Benchmark	Configuration	Overhead
AMG2006	4 MPI * 128 threads	604s (+9.6%)
Sweep3D	48 MPI	90s (+2.3%)
LULESH	48 threads	19s (+12%)
Streamcluster	128 threads	27s (+8.0%)
NW	128 threads	80s (+3.9%)

NUMA analysis: code-, data-, and address-centric analysis + first touch

Methods	LULESH	AMG2006	Blacksholes
IBS	295s (+24%)	89 (+37%)	192s (+6%)
MRK	93s (+5%)	27s (+7%)	132s (+4%)
PEBS	65s (+45%)	96s (+52%)	82s (+25%)
DEAR	90s (+7%)	120s (+12%)	73s (+4%)
PEBS-LL	35s (+6%)	57s (+8%)	67s (+3%)

Conclusions and Future Work

- Hardware address sampling
  - widely supported in modern architectures
  - powerful in monitoring memory behaviors
  - currently in early stage of studies
    - focusing on data collection and attribution
- Potentials of hardware address sampling
  - provide deeper insights than traditional performance counters
  - require novel analysis methods to expose performance insights
- Future work
  - integrating address sampling into Charm++ runtime for online optimization