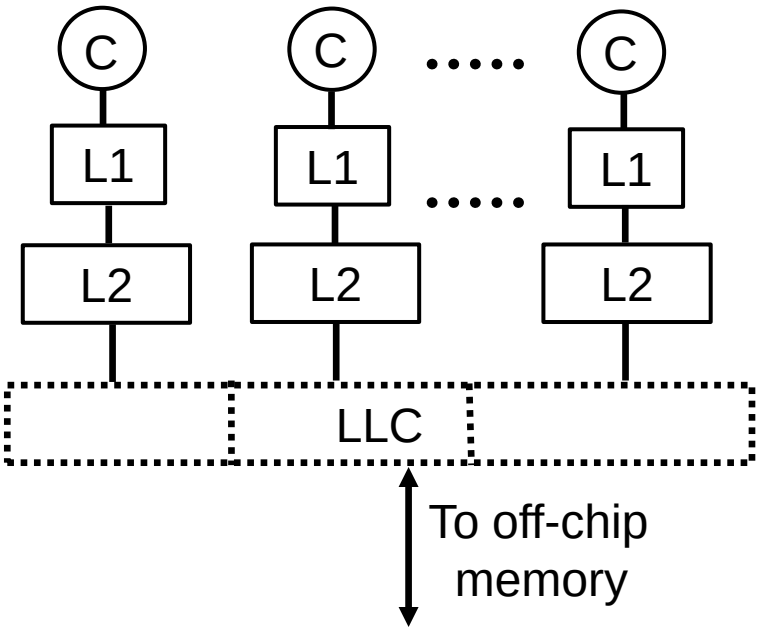
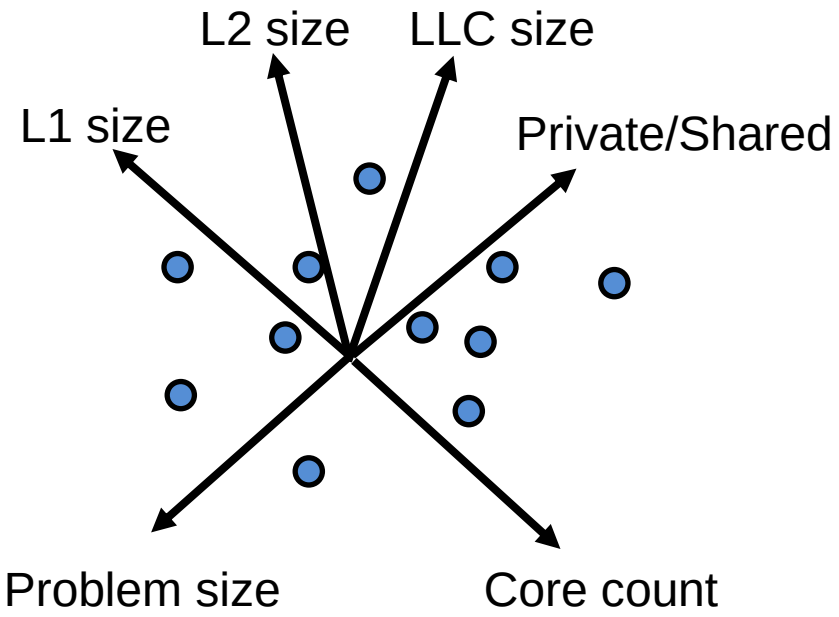
Identifying Optimal Multicore Cache Hierarchies for Loop-based Parallel Programs via Reuse Distance Analysis

Meng-Ju Wu and Donald Yeung
University of Maryland, College Park

Motivation

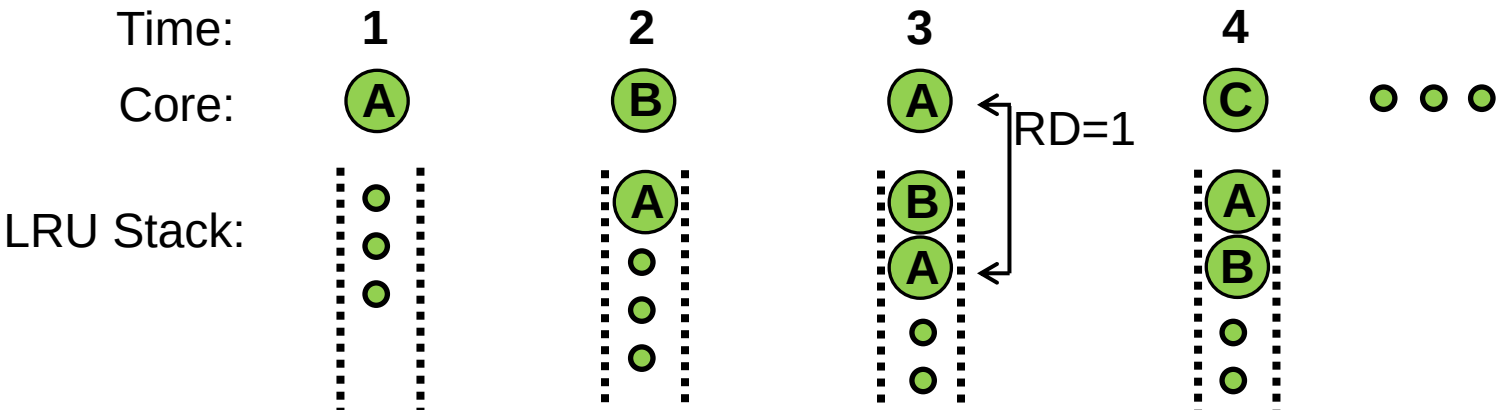
- Trend → Multicore
- Understanding memory performance is critical but difficult
- Large design space and slow simulation
  - **3,168** simulations → **3** months on **74** core cluster

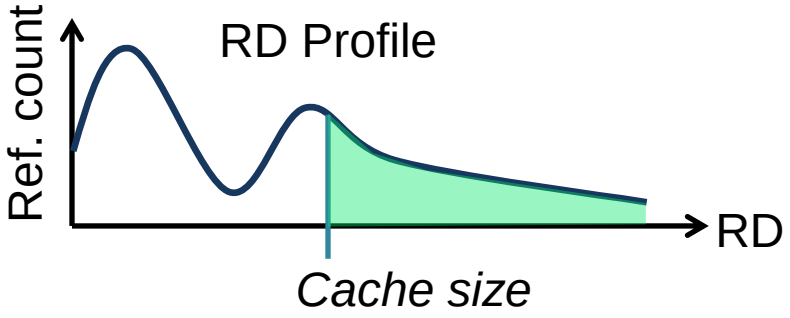


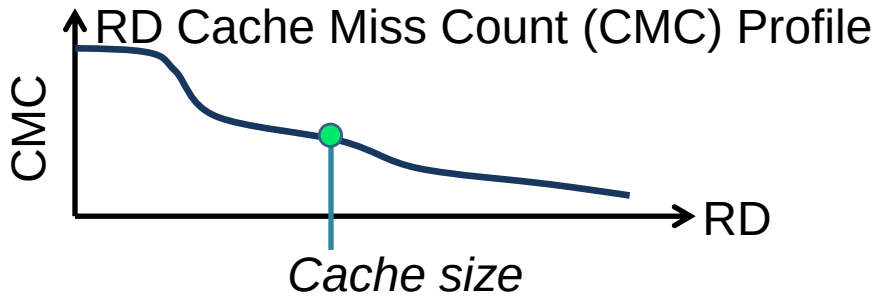


Reuse Distance (RD)

- RD: Depth in LRU stack [Mattson70]
- Architecture independent
- Provide app's memory performance across cache sizes

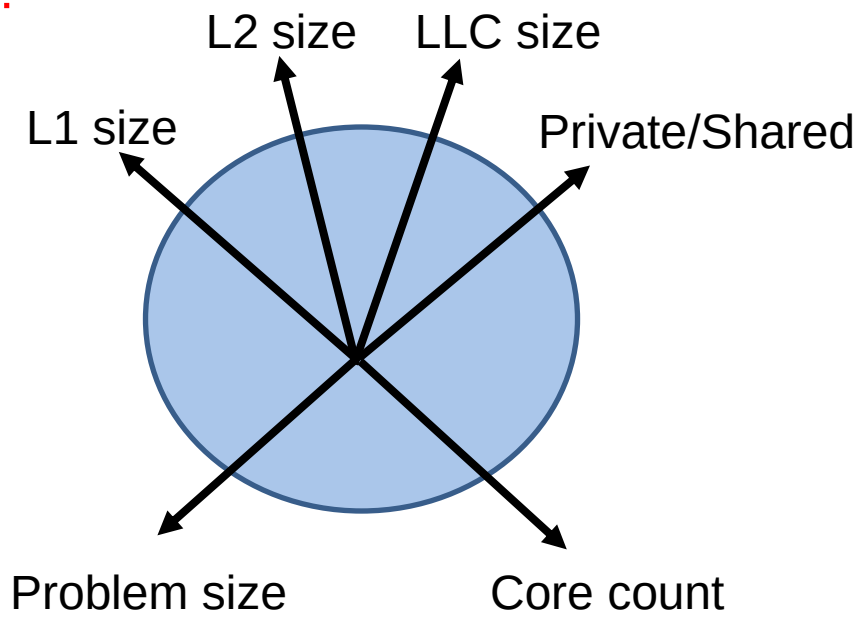






Multicore Reuse Distance

- Concurrent Reuse Distance: shared cache[Ding09;Jiang10;Schuff09,10]
- Private-stack Reuse Distance: private cache[Schuff09,10]
- Provide the complete picture of the design space
  - What is the optimal hierarchy configuration?
  - What is the performance impact for different configurations?
  - What is the scaling impact?
  - … and more

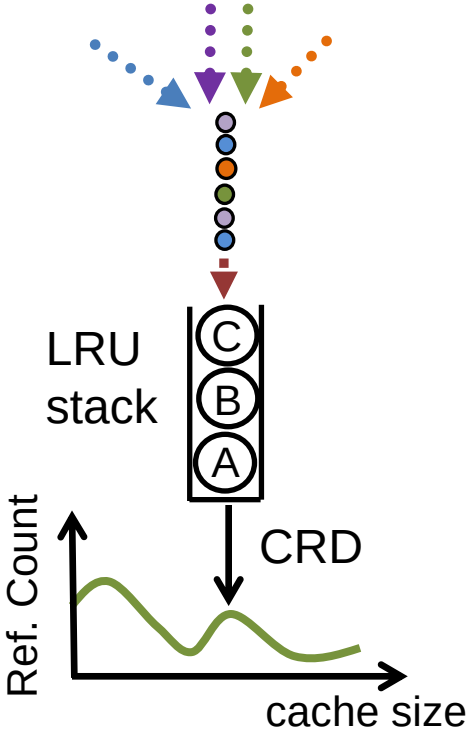


Outline

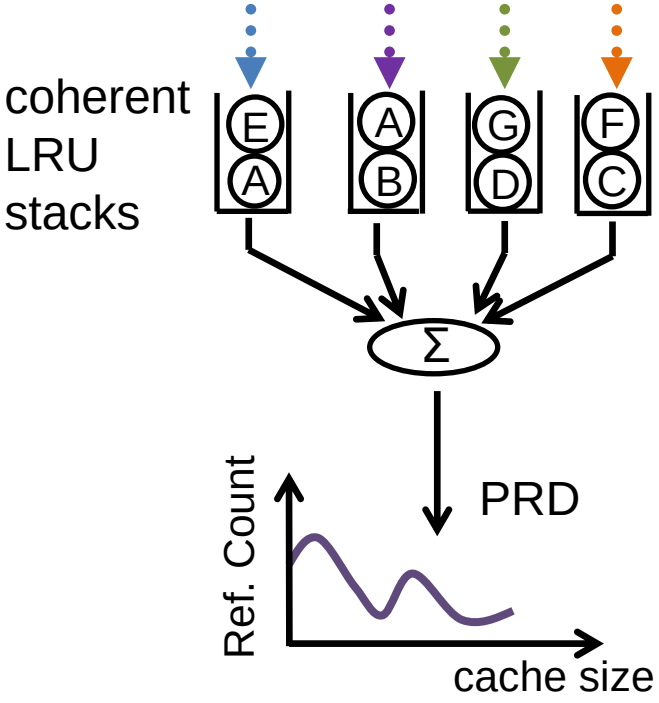
- Motivation
- Multicore Cache Performance Modeling
  - Pin tool
  - Benchmarks
  - Cache Performance Models
- Two Cases
  - Scaling Impact on Private vs. Shared Caches
  - Optimal L2/LLC Capacity Allocation
- Conclusions

CRD and PRD profiles

- Concurrent Reuse Distance (CRD)
  - Shared cache
  - RD across interleaved memory streams



- Private-stack Reuse Distance (PRD)
  - Private cache
  - RD on coherent per-thread stacks



- Profiling : In-house PIN tool
  - Uniform interleaving, 64B cache block

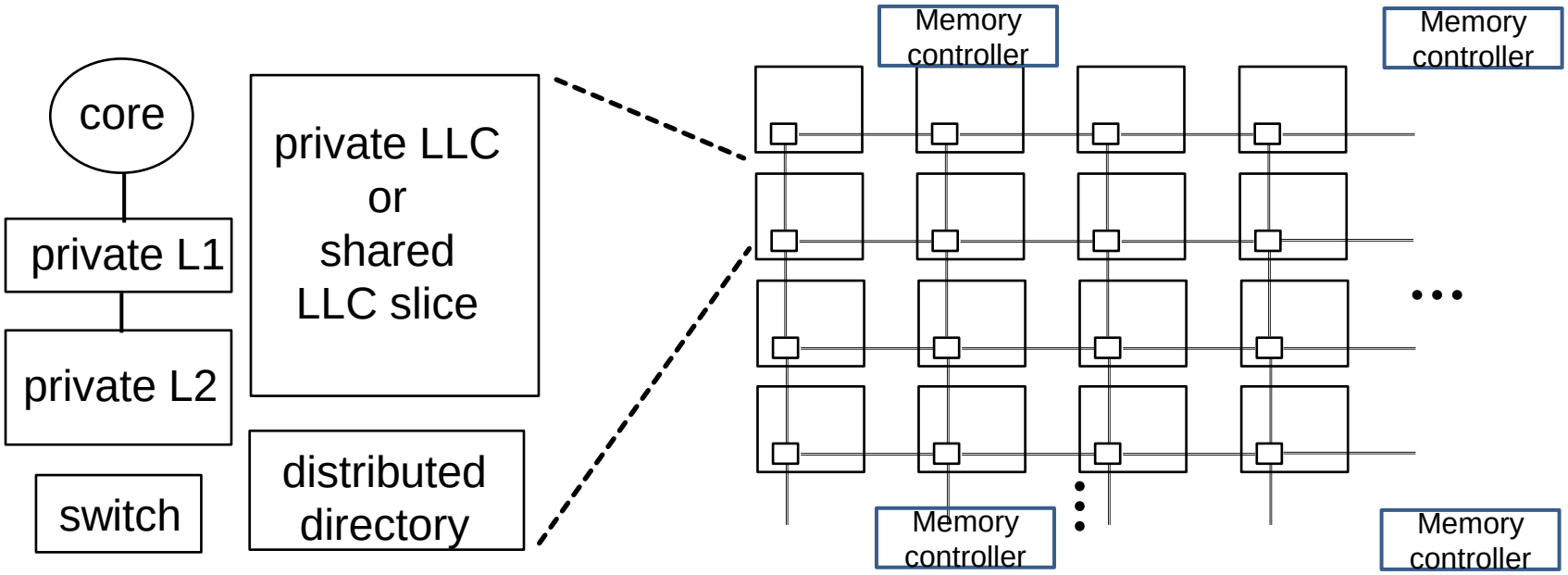
Benchmarks

- Two problem sizes
  - S1 and S2 → problem size scaling

Benchmark	Problem Sizes		Units
	S1	S2	
FFT	2^{20}	2^{22}	elements
LU	1024^2	2048^2	elements
RADIX	2^{22}	2^{24}	keys
Barnes	2^{17}	2^{19}	particles
FMM	2^{17}	2^{19}	particles
Ocean	514^2	1026^2	grid
Water	25^3	40^3	molecules
KMeans	2^{20}	2^{22}	objects
BlackScholes	2^{20}	2^{22}	options

Tile CMP

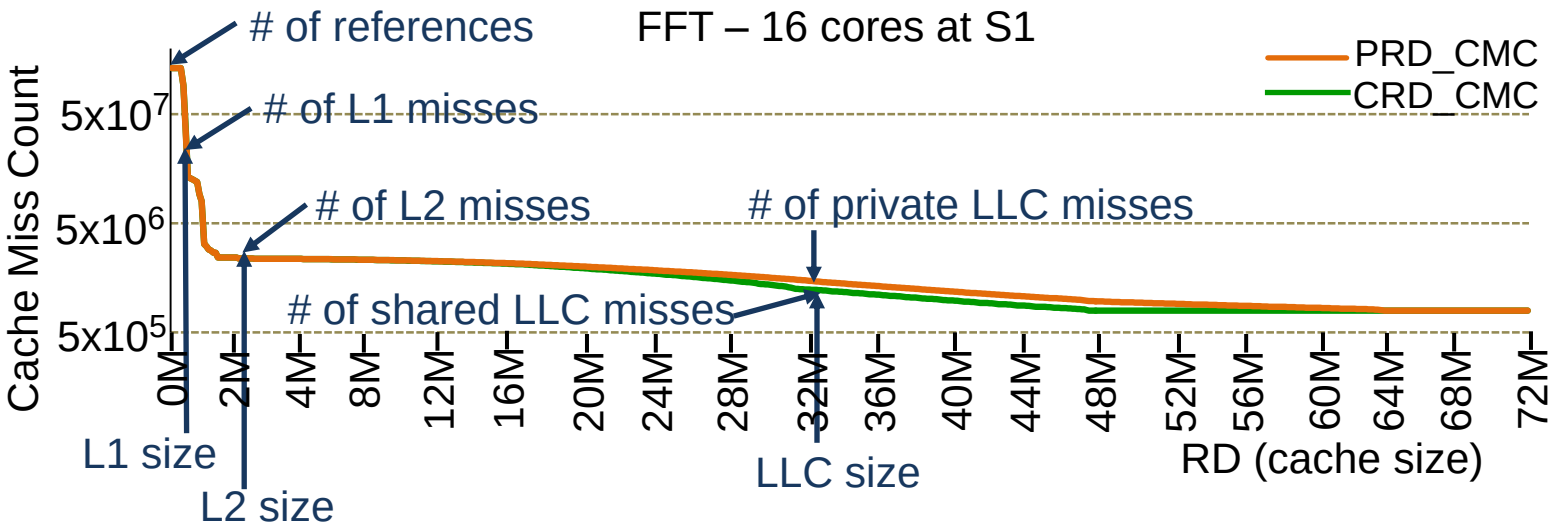
- Scalable Architecture
  - 2 , 4, 8, 16, 32, 64, 128, 256 core → core count scaling
- Three-level cache hierarchy
  - L1 and L2 cache: private; LLC: private or shared



	L1 lat	L2 lat	LLC lat	DRAM lat	Hop Lat
Cycles	1	4	10	200	$3x(\sqrt{core\_counts}+1)$

Average Memory Access Time (AMAT) Modeling

- CRD_CMC and PRD_CMC
  - Number of cache misses at each cache level



Shared LLC:

$AMAT_S$=(# of references x L1_lat +
# of L1 misses x L2_lat +
# of L2 misses x (2Hop_lat + LLC_lat) +
# of LLC misses x (2Hop_lat + DRAM_lat)
) / # of references

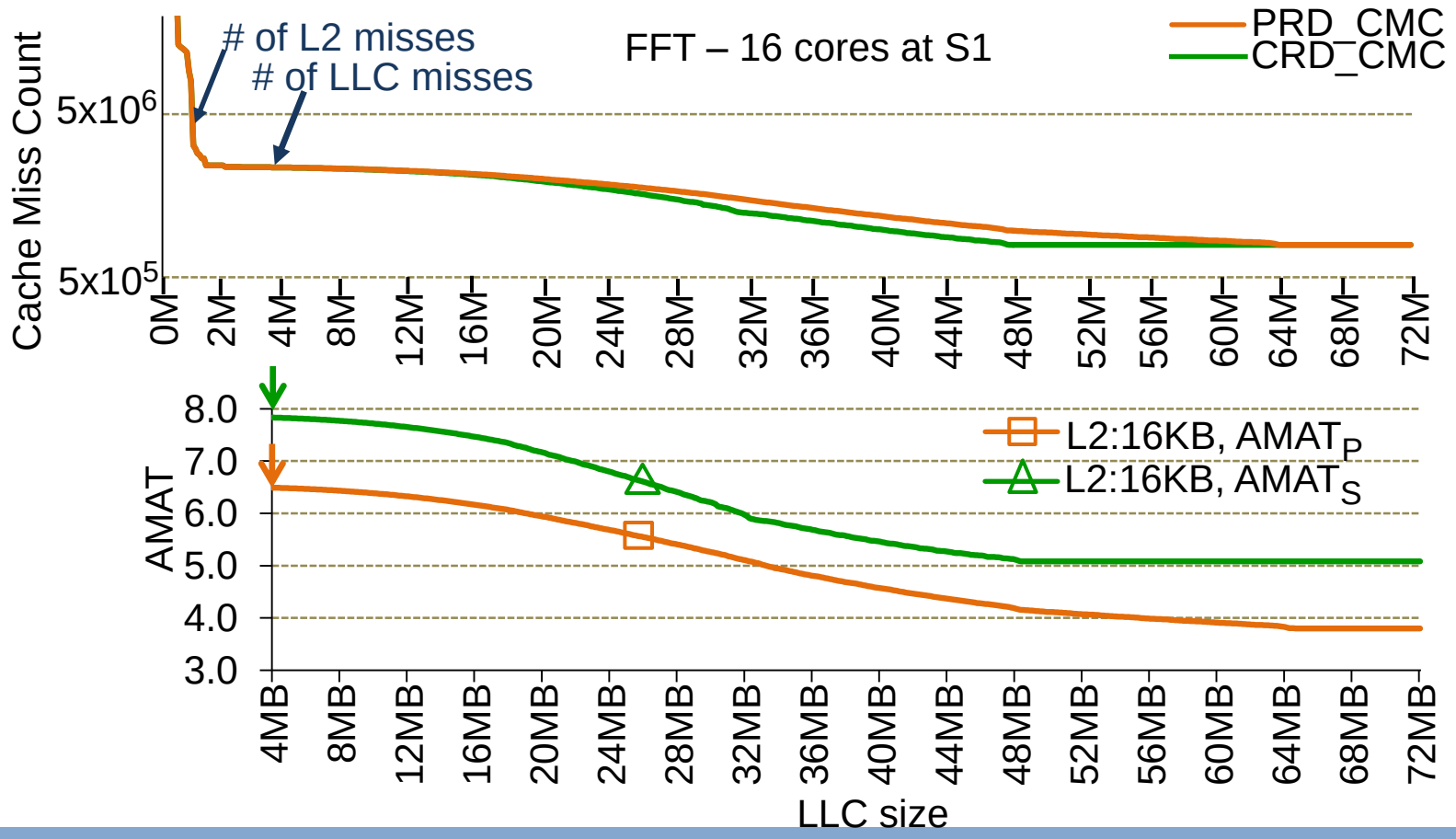
Private LLC:

$AMAT_P$ =(# of references x L1_lat +
# of L1 misses x L2_lat +
# of L2 misses x LLC_lat +
# of DIR accesses x (Hop_lat + DIR_lat) +
# of forwarding x (2Hop_lat + LLC_lat) +
# of LLC misses x (2Hop_lat + DRAM_lat)
) / # of references

Case 1: Private vs. Shared LLC

- $AMAT_S < AMAT_P$

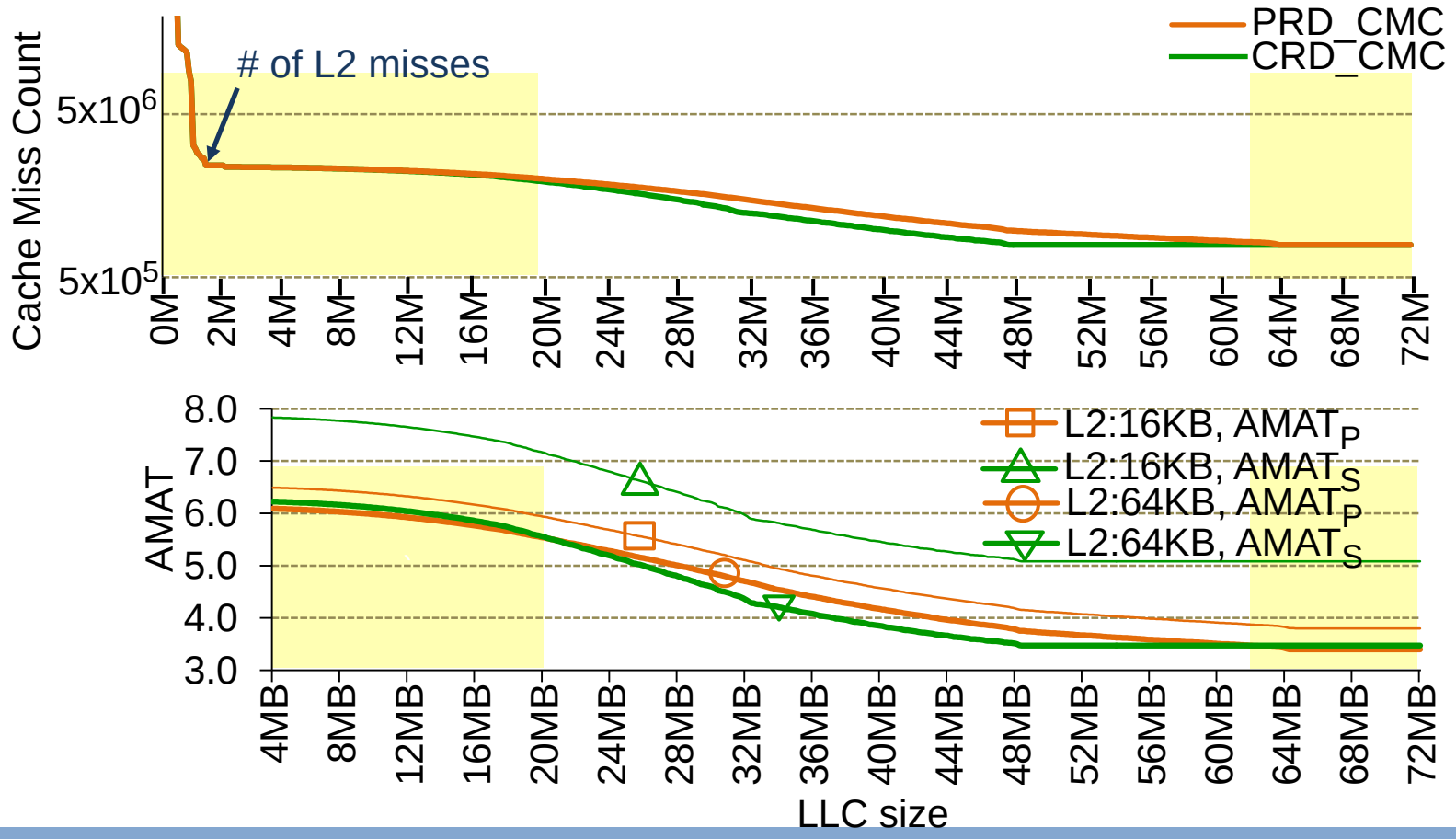
Shared LLC's on-chip memory stall - Private LLC's on-chip memory stall < Private LLC's off-chip memory stall - Shared LLC's off-chip memory stall



Case 1: Private vs. Shared LLC

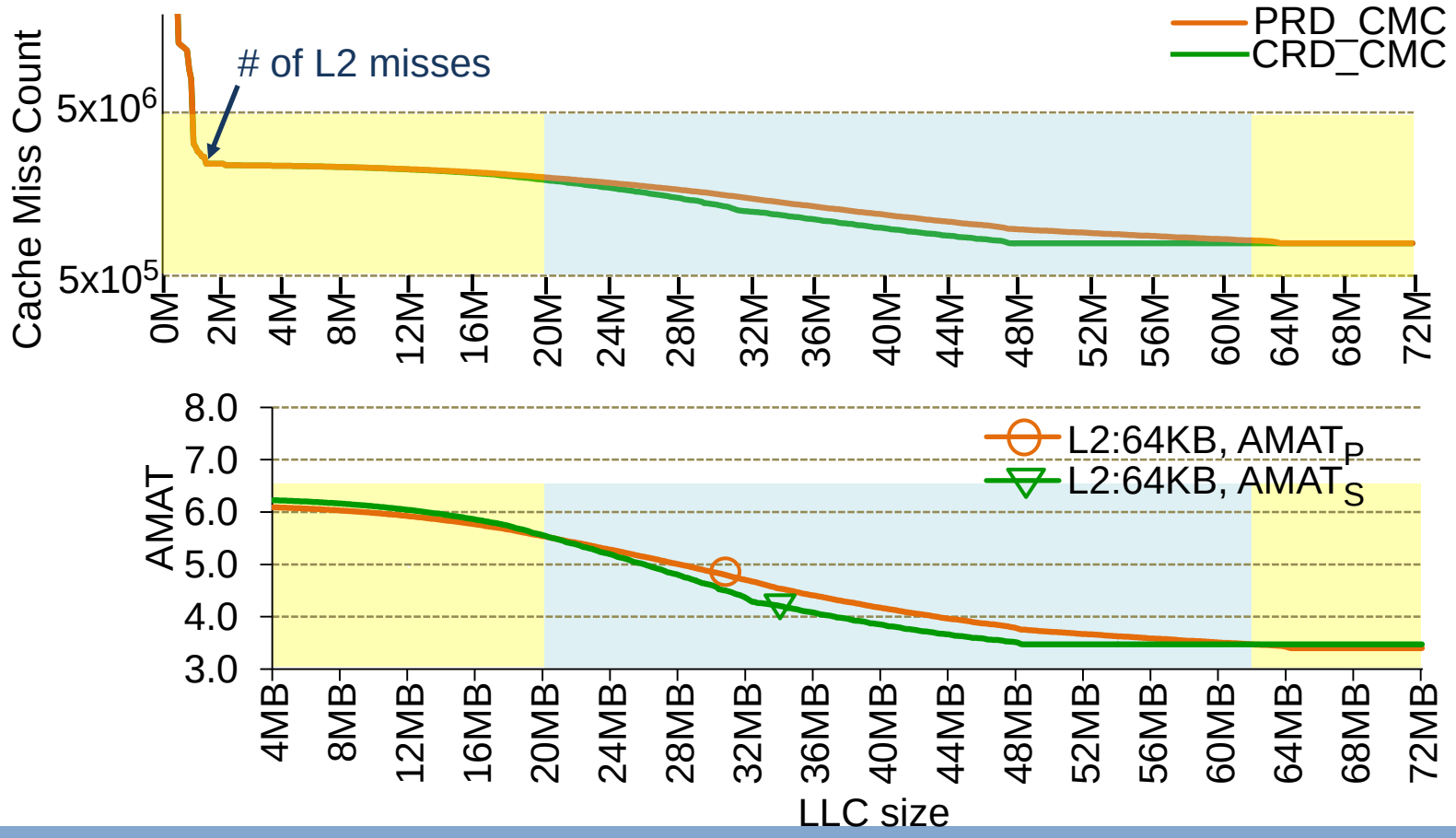
- $AMAT_S < AMAT_P$

Shared LLC's on-chip memory stall - Private LLC's on-chip memory stall < Private LLC's off-chip memory stall - Shared LLC's off-chip memory stall



Case 1: Private vs. Shared LLC

- $AMAT_S < AMAT_P$
  - L2 size > high reference count region
  - CRD_CMC/PRD_CMC gap is large



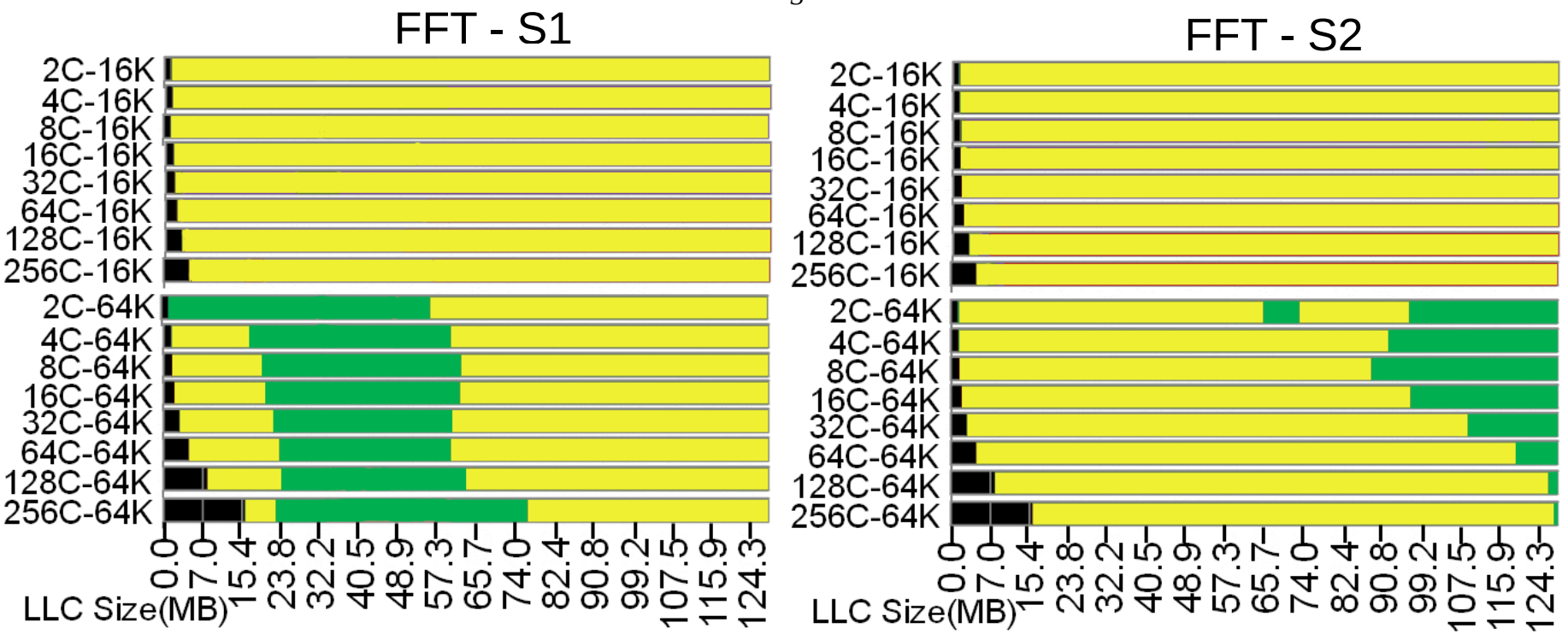
Private vs. Shared LLC Under Scaling Impact

- Problem scaling → Private LLC
- Core count scaling → Private LLC

Shared LLC is better

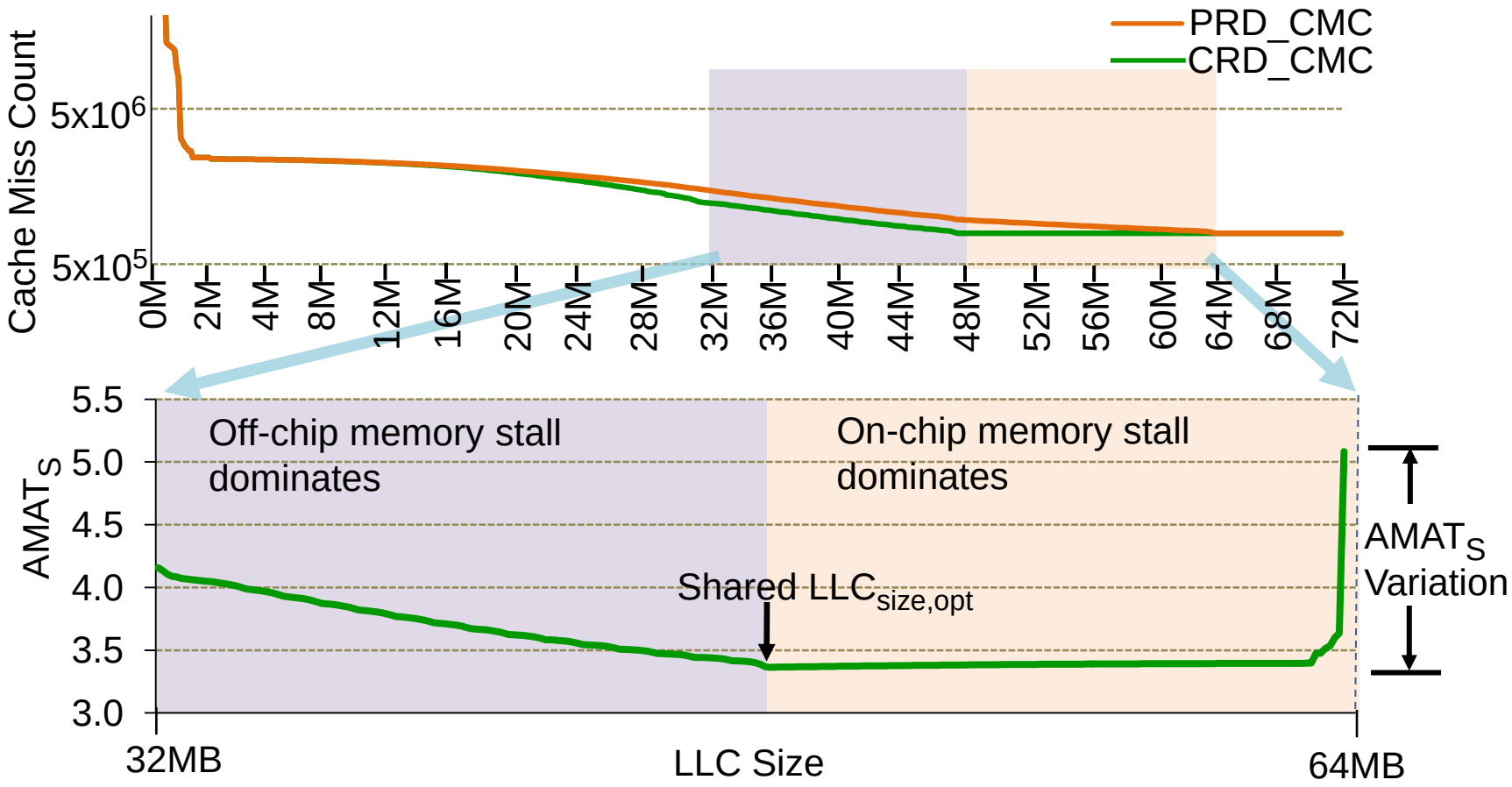
$$Ratio = \frac{AMAT_P}{AMAT_S}$$

<1.0 >1.0



Case 2: L2/LLC Capacity Allocation

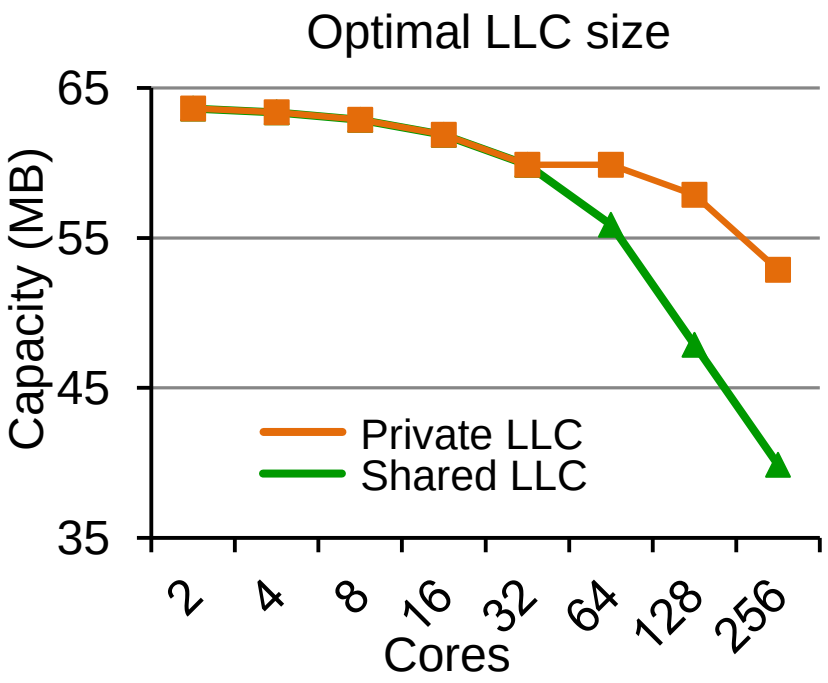
- Optimal L2/LLC allocation at a fixed on-chip capacity
  - L2 size + LLC size = 64MB, LLC > L2 → LLC size: 32~64MB
  - Balancing on-chip memory and off-chip memory stalls

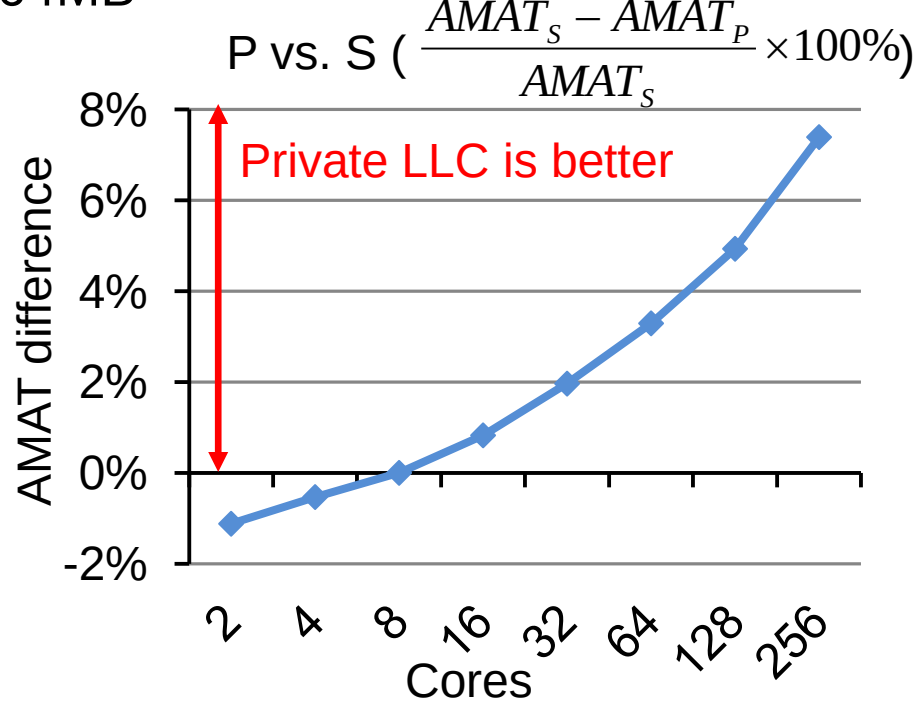


Case 2: Optimal LLC Size

- Private LLC size > Shared LLC size
- Core count scaling → optimal LLC size ↓
- Core count scaling prefers private LLC

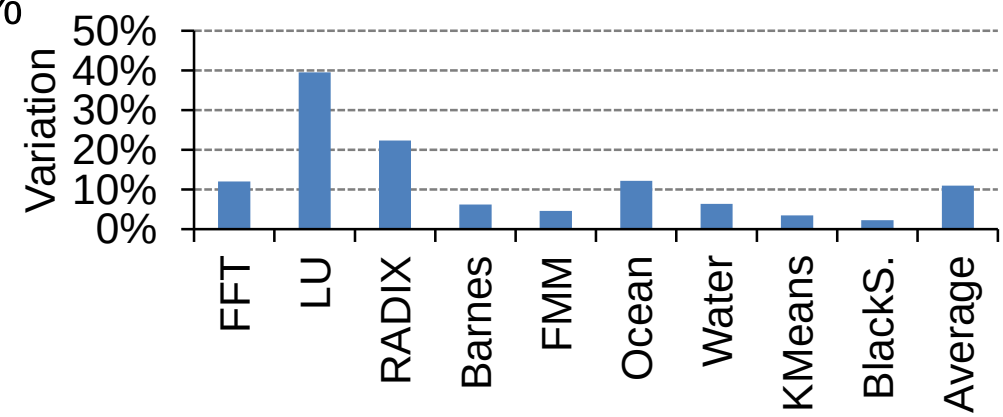
FFT - S2 / 64MB



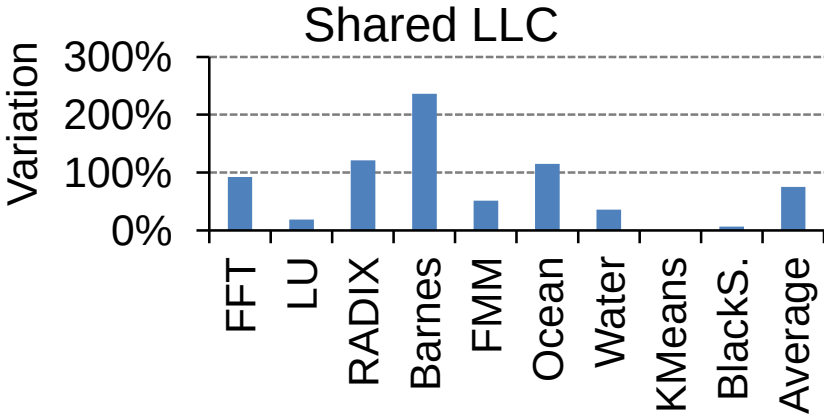


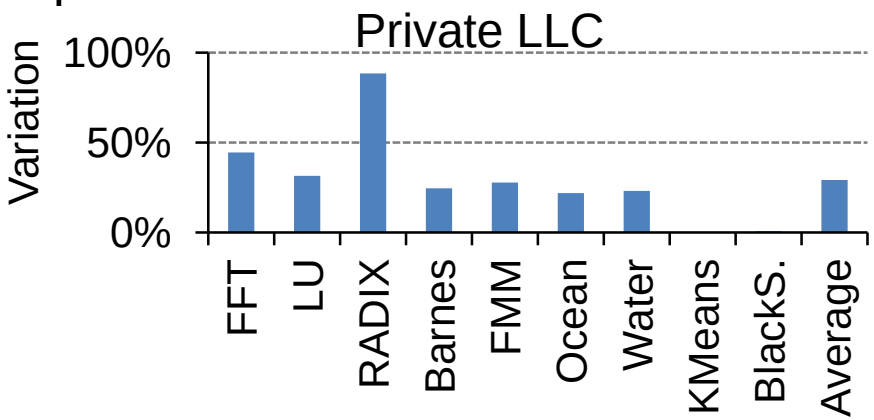
Case 2: Importance of Design Options

- AMAT variation for private/shared LLC at $LLC_{size,opt}$
  - < 11%



- AMAT variation for L2/LLC partitioning
  - < 76% in shared LLC, < 30% in private LLC





Conclusions

- Multicore RD Analysis
  - Fast, insightful, and complete picture
- Architecture Implications
  - Shared LLC can outperform private LLC
    - L2 size > high reference count region
    - CRD_CMC/PRD_CMC gap is large
  - The optimal L2/LLC partition: balancing on-chip and off-chip memory stalls
  - L2/LLC size allocation is more important than private/shared LLC selection
- Ongoing work
  - Reconfigurable caches
  - Dynamic on-chip resource management