Optimizing Replication, Communication, and Capacity Allocation in CMPs

Zeshan Chishti, Michael D Powell, and T. N. Vijaykumar

School of ECE
Purdue University

Motivation

CMP becoming increasingly important

- Increased capacity pressure on the on-chip memory
  - need large on-chip capacity
- Increased cache latencies in large caches due to wire delays

Conventional MP caches:

- Shared Cache
  - – larger => slower
  - \+ better utilization of cache capacity
- Private Caches
  - \+ smaller => faste
  - – limited capacity available to each core

Neither private nor shared provide both capacity and fast access

Latency-Capacity Tradeoff in CMPs

SMPs and DSMs also target better capacity and fast access

CMPs fundamentally change the latency-capacity tradeoff

- Capacity
  - on-chip storage limited in CMPs
  - in-node storage virtually unlimited in SMPs & DSMs
- Inter-processor communication Latency
  - on chip in CMPs => fast
  - off chip in SMPs and DSMs => slow

SMPs & DSMs have capacity but high latency, CMPs are reverse

Need mechanisms to exploit CMP latency-capacity tradeoff

Contributions

Key Observation:

CMPs fundamentally change the latency-capacity tradeoff

Novel mechanisms:

- (1) Controlled Replication for read-only sharing
  - copies reduce latency but use up on-chip capacity
  - avoid copies sometimes and obtain data from neighbor
  - incur a few cycles but save many-cycle off-chip miss
- (2) In-situ Communication for read-write sharing
  - inter-CPU communication + copies => coherence misses
  - use single copy to avoid coherence misses
  - incur a few cycles but save many-cycle coherence miss

Contributions (cntd.)

Novel mechanisms:

- (3) Capacity Stealing for no sharing
  - migrating data close to requestor may evict other data
  - may waste unused capacity in other cores
  - place excess data in other cores' unused cache frames
  - incur a few cycles but save many-cycle off-chip miss

Novel organization:

- Pure shared or private still problematic
- CMP NuRAPID: hybrid of shared data and private tag

Performance improvements over both shared and private cache

Outline

- Introduction
- CMP NuRAPID organization
- CMP NuRAPID mechanisms
- Methodology and Results
- Conclusion

CMP NuRAPID organization

Hybrid of private tag and shared data

Private per-processor tag arrays

- Fast tag access

Shared data array

- Better capacity utilization but slow due to wire delays

Use non-uniform-access for fast access to shared data array

Non-uniform access for large uniprocessor cache

NUCA (ASPLOS'02)

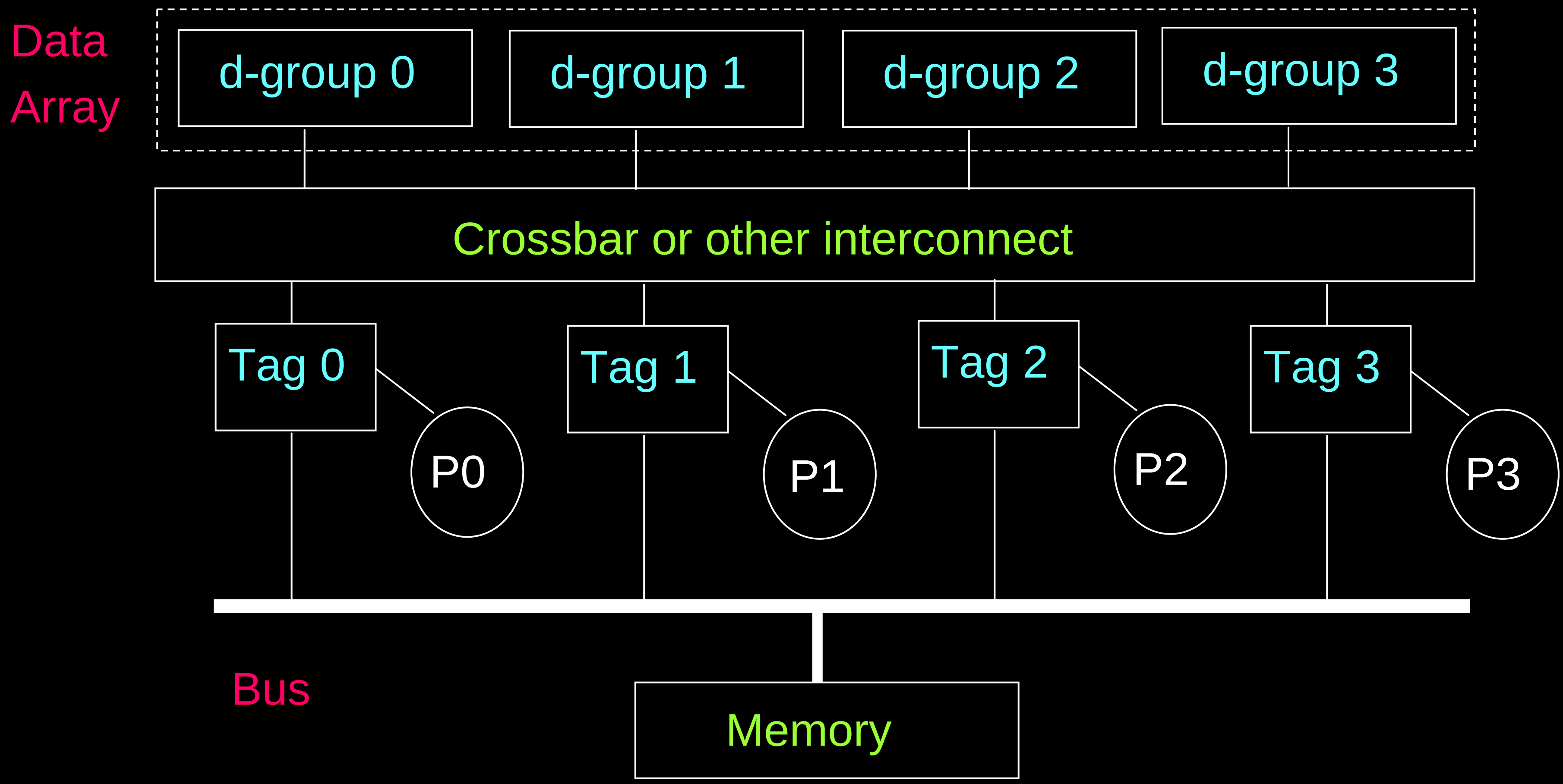
- Divides cache into regions ("d-groups") based on distance
- Fast access to closer d-groups
- Slow access to farther d-groups
- Migrate frequently-accessed data to close d-groups

NuRAPID (MICRO'03): Improvement upon NUCA

- Sequential tag first and then data access
- Use pointers to decouple tag and data placement
  - allow any tag entry to point to any d-group
  - i.e., a tag entry can point to data in another core's d-group

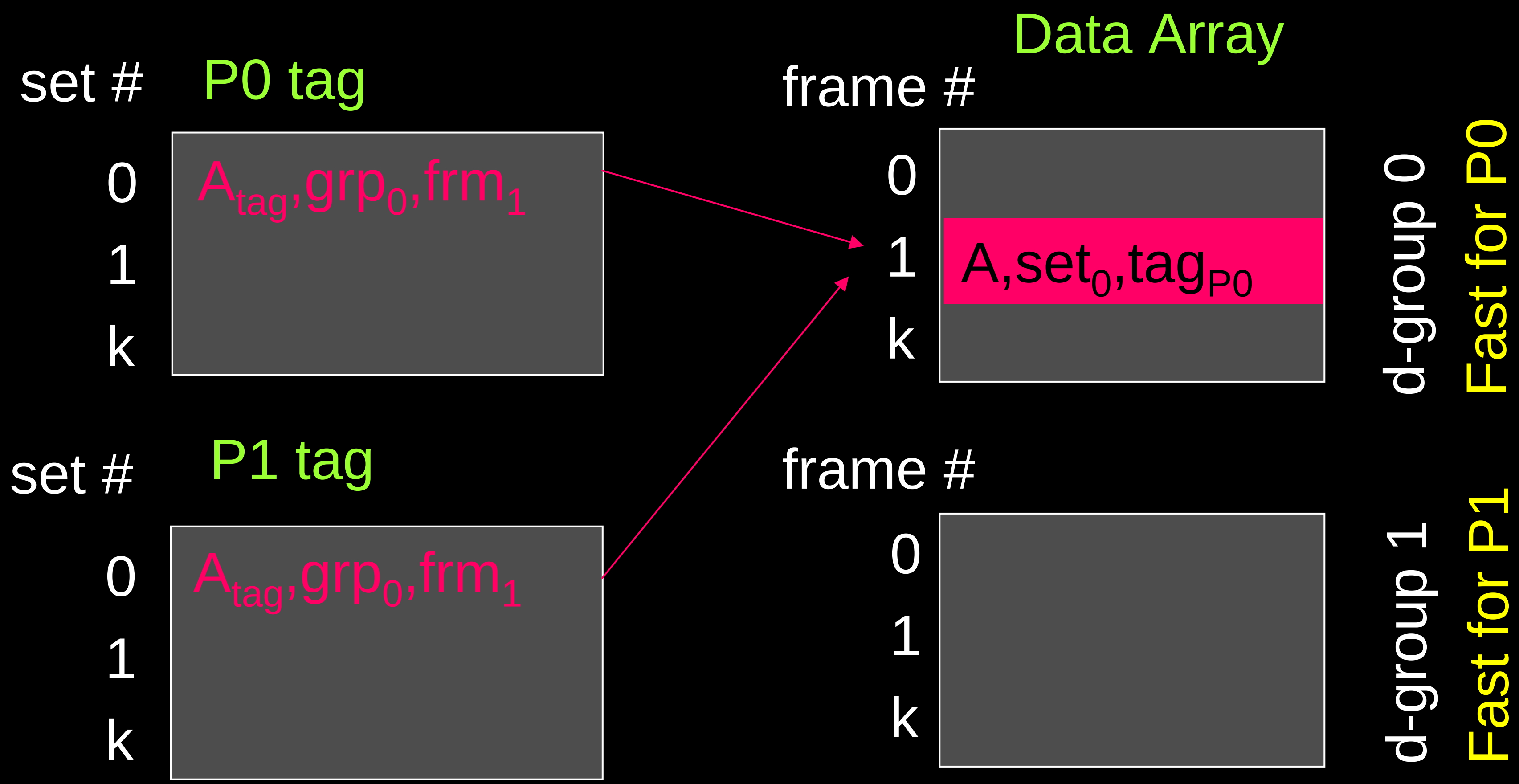
NuRAPID's decoupling key to our mechanisms

CMP NuRAPID organization



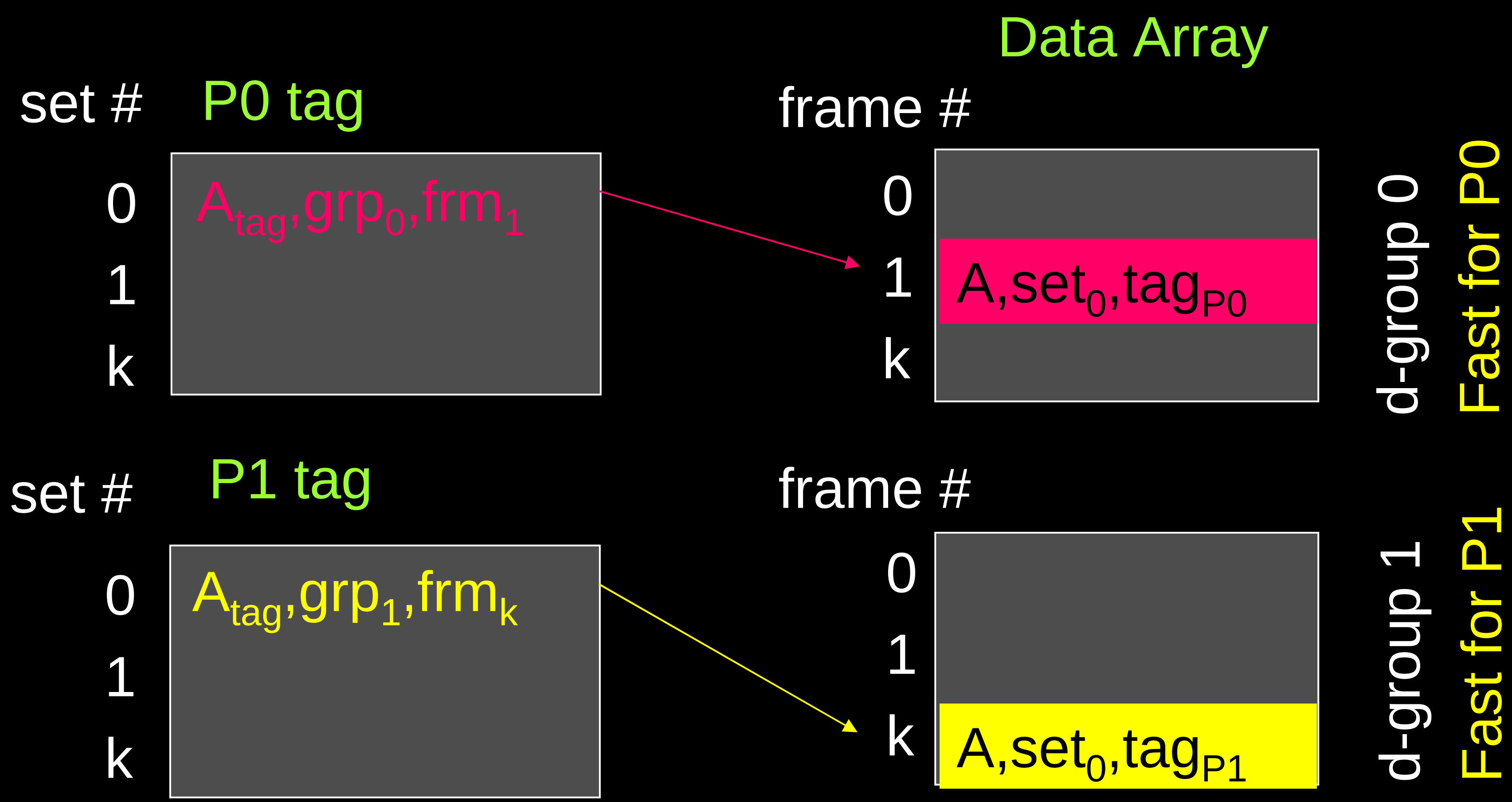
Tag arrays snoop on a bus to maintain coherence

Single copy of block shared by multiple tags



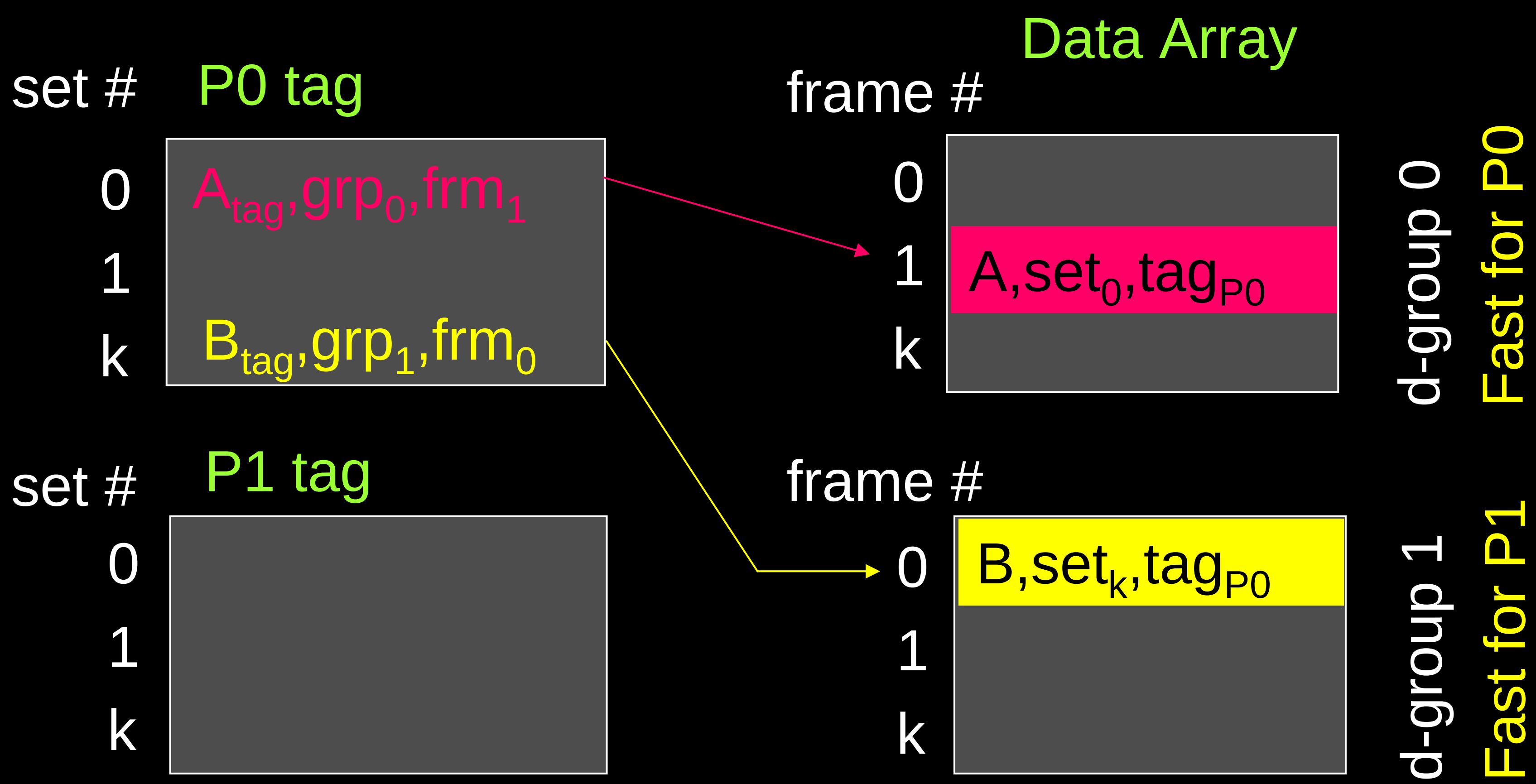
Supports controlled replication and in-situ communication

Copies of a shared block in different d-groups



Allow replication when needed

Data for one core in different d-groups



Supports capacity stealing

Outline

- Introduction
- CMP NuRAPID organization
- CMP NuRAPID mechanisms
- Methodology and Results
- Conclusion

Controlled Replication

- Key Idea: Avoid copies for some read-only-shared data
- No copying of already-on-chip block on first use
  - update tag pointer to point to the already-on-chip block
  - save capacity for blocks used only once
- Obtain data from existing on-chip copy on second use
  - use tag pointer to locate the already-on-chip block
  - small latency penalty
- Never-copying makes future uses slow
  - replicate on second use anticipating future uses
  - detect second use by tag pointing to a far d-group
    - No need of counters or extra bits

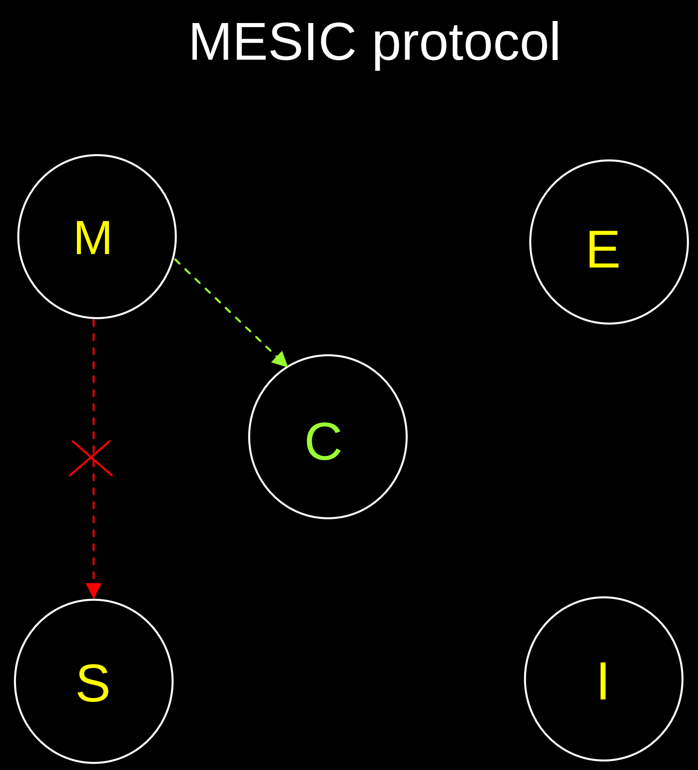
Better exploitation of latency-capacity tradeoff

In-Situ Communication

- Key Idea: Use fast on-chip communication to avoid coherence miss of read-write-shared data
- Enforce single copy of read-write shared block in L2
  - via controlled replication
- Keep RW-shared blocks in communication (C) state
  - writer writes-through to the single copy in L2
  - reader reads the single copy
  - no invalidation & replication => no coherence miss
- Blocks often read multiple time before being re-written
  - move the data copy close to the reader

Not only fast communication but also capacity savings

In-Situ Communication (cntd.)



- Replace M to S transition by M to C transition
- Other transitions discussed in paper

Capacity Stealing

- Key Idea: Allow a core to steal another core's unused capacity
- Upon a miss:
  - Create space by demoting a block in closest d-group
  - Place new block in that space
  - Place demoted block in unused space in another d-group
    - avoid off-chip miss for demoted block
- Details of block movement policies in paper

Important for workloads with capacity demands non-uniform across cores (e.g., multiprogrammed)

Outline

- Introduction
- CMP NuRAPID organization
- CMP NuRAPID mechanisms
- Methodology and Results
- Conclusion

Methodology

Full-system simulation of 4-core CMP using Simics

- 64 KB, 2-way L1s

CMP NuRAPID: 8 MB, 8-way

- 4 d-groups (11-, 25-, 25-, and 38- cycles)
- 1-port for each tag array and data d-group

Compare to:

- Private 2 MB, 8-way, 1-port per core (10 cycles)
- Shared 8 MB, 32-way, 4-port (latency of 8-way 1-port: 59 cycles)
- CMP-SNUCA (MICRO'04)
  - Shared with non-uniform-access, no replication

Shared vs private vs CMP NuRAPID

Miss Rate

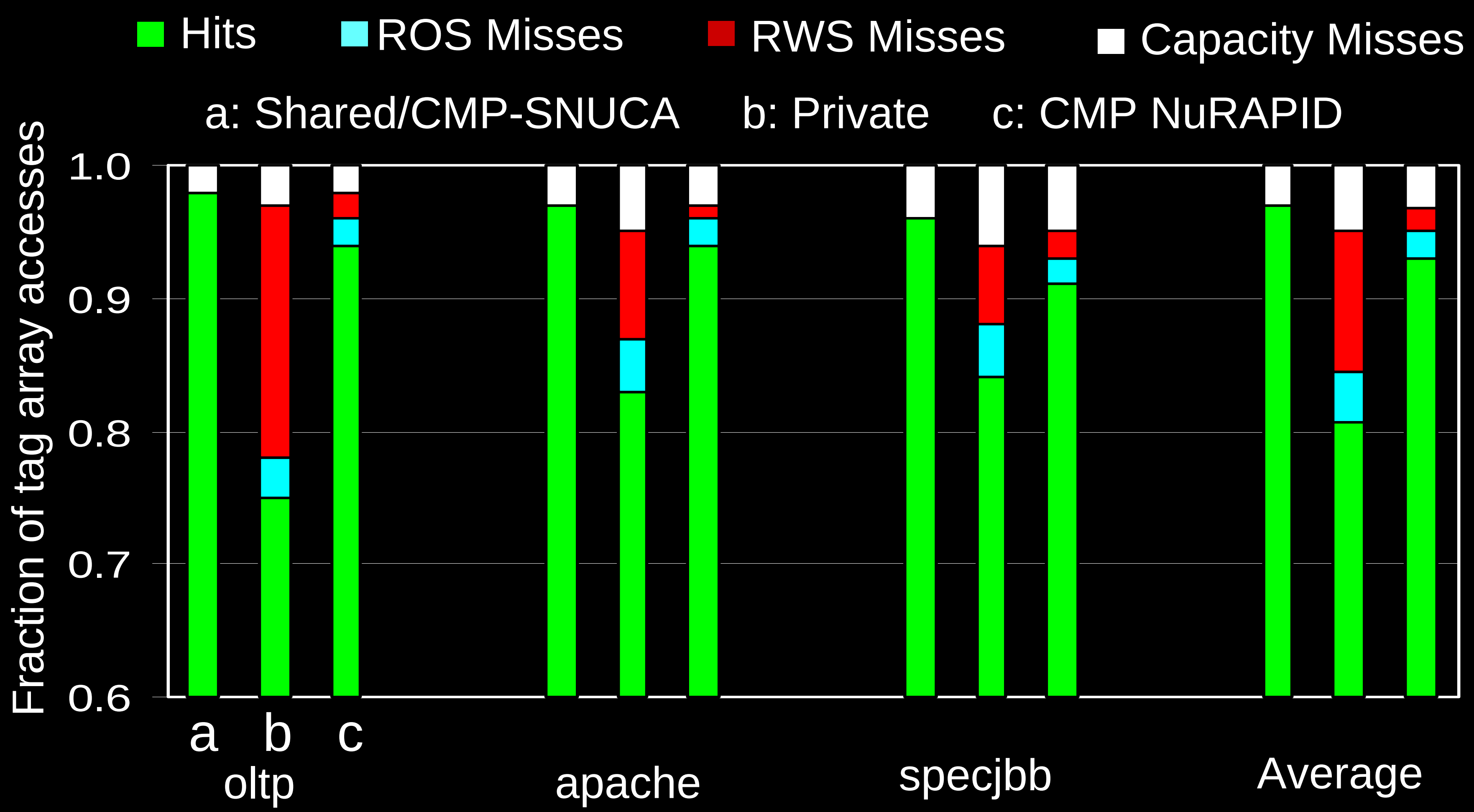
- Shared: only capacity misses, no ROS and RWS misses
- Private: more capacity misses, also ROS and RWS misses
  – worse than shared
- CMP NuRAPID:
  – less capacity, ROS, and RWS misses than private
  – better than private, close to shared

Hit latency

- Shared: worst
- Private: best
- CMP NuRAPID: better than shared, close to private

CMP NuRAPID: Shared's miss rate and Private's latency

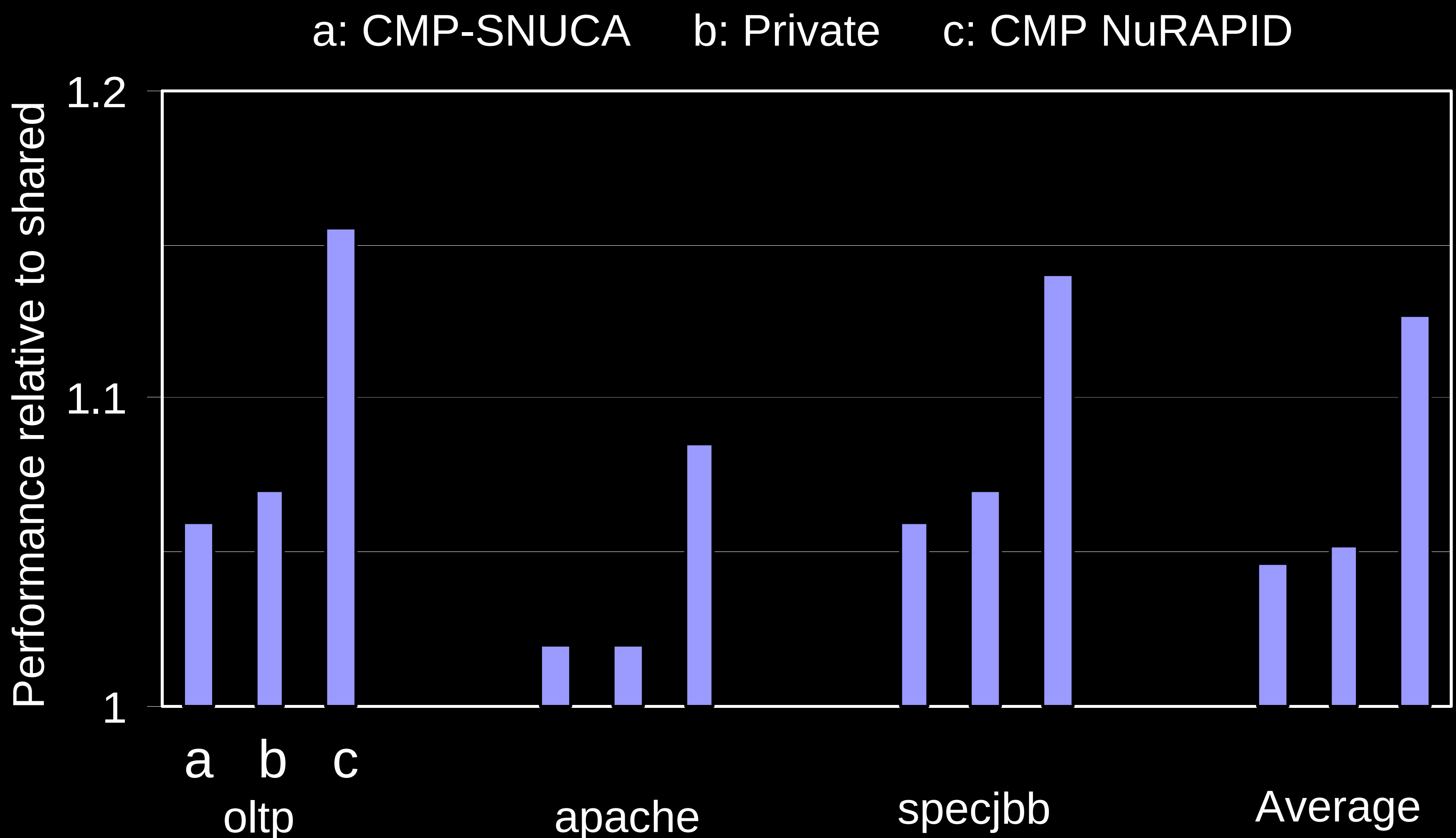
Distribution of accesses



CMP NuRAPID: 85% hits to closest d-group

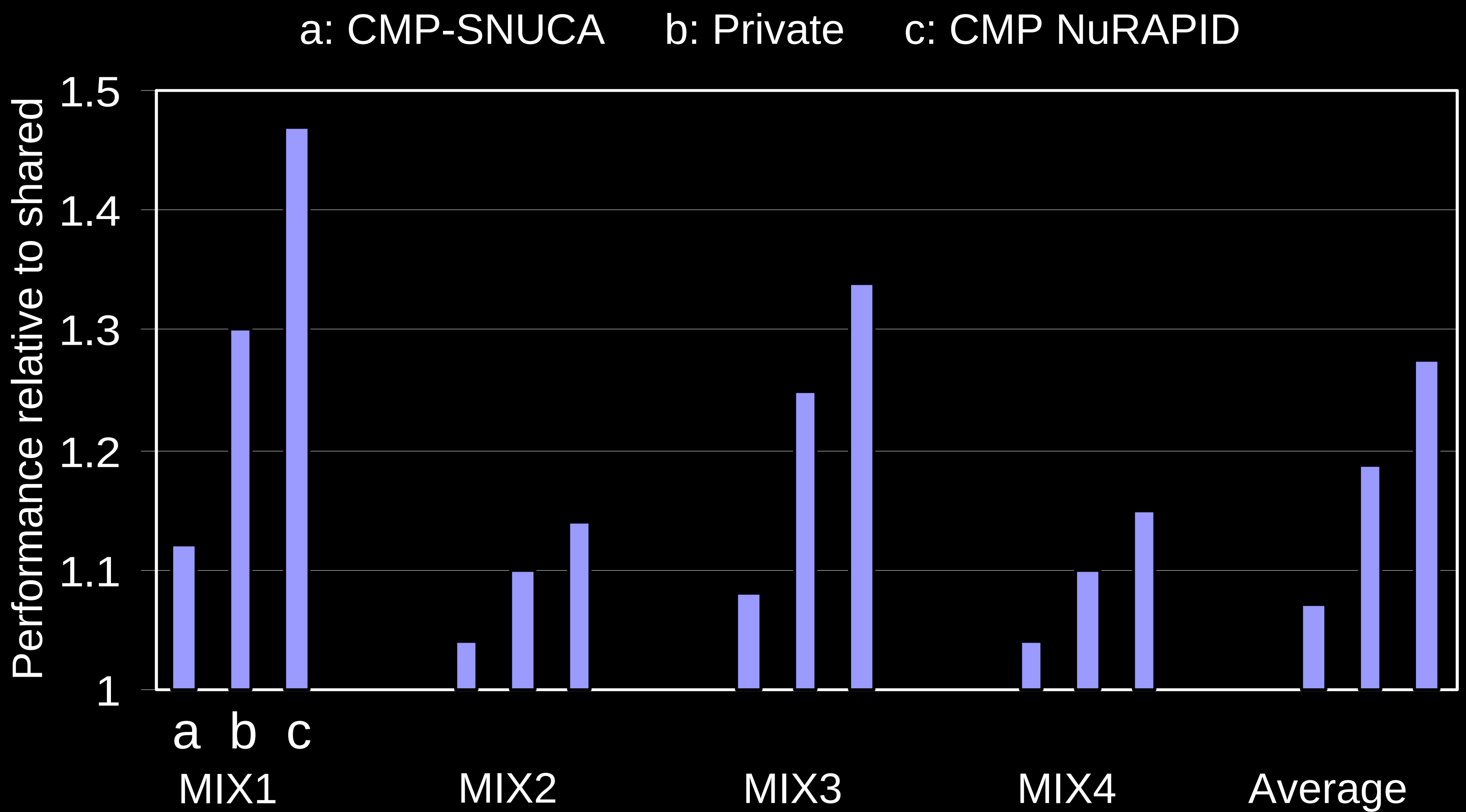
CMP-NuRAPID vs CMP-SNUCA vs Private: 13- vs 25- vs 10-cycle average hit latency

Performance: Multithreaded Workloads



CMP NuRAPID outperforms shared, private, and CMP-SNUCA

Performance: Multiprogrammed Workloads



CMP NuRAPID outperforms shared, private, and CMP-SNUCA

CMP NuRAPID (Purdue) vs previous talks (IBM, MIT)

	CMP NuRAPID	IBM	Victim Replication (MIT)
Data Placement	NuRAPID mapping flexible => working set close to each core	Private cache	Static inflexible mapping => other cores can over-run the close d-group capacity
Controlled Replication	Based on usage patterns	Accidental 2nd-order effect of capac. steal.	Default no replication; adds uncontrolled replication
In-situ Comm.	Yes	No equivalent; pure private cache	By default because shared cache
Capacity Stealing	Yes for multithreaded and multiprogrammed	Yes for multithreaded; No for multiprogramed	Unwanted capacity stealing may occur due to mapping
Perf-ormance	Better than both shared and private in all workloads	Better than private for multithreaded; no multiprogram, no shared comparison	Better than shared, slightly worse than private in 8 out of 11 workloads, better than both in the other 3
Complexity	More involved	Simpler	Simpler
Summary	Hybrid with all three	private + capac. steal.; plus L3 optimization	shared + some replication

Outline

- Introduction
- CMP NuRAPID organization
- CMP NuRAPID mechanisms
- Methodology and Results
- Conclusion

Conclusions

- CMPs fundamentally change the latency-capacity tradeoff
  - SMPs & DSMs: capacity but high latency, CMPs: reverse
- Controlled replication, in-situ communication, and capacity stealing allow exploitation of CMP's latency-capacity tradeoff
- CMP NuRAPID
  - Novel design incorporates the three mechanisms
- For commercial multi-threaded workloads
  - 13% better than shared, 8% better than private
- For multi-programmed workloads
  - 28% better than shared, 8% better than private

CMP NuRAPID: an important cache design for future CMPs