Προηγμένη Αρχιτεκτονική Υπολογιστών

Non-Uniform Cache Architectures

Νεκτάριος Κοζύρης & Διονύσης Πνευματικάτος

{nkoziris,pnevmati}@cslab.ece.ntua.gr

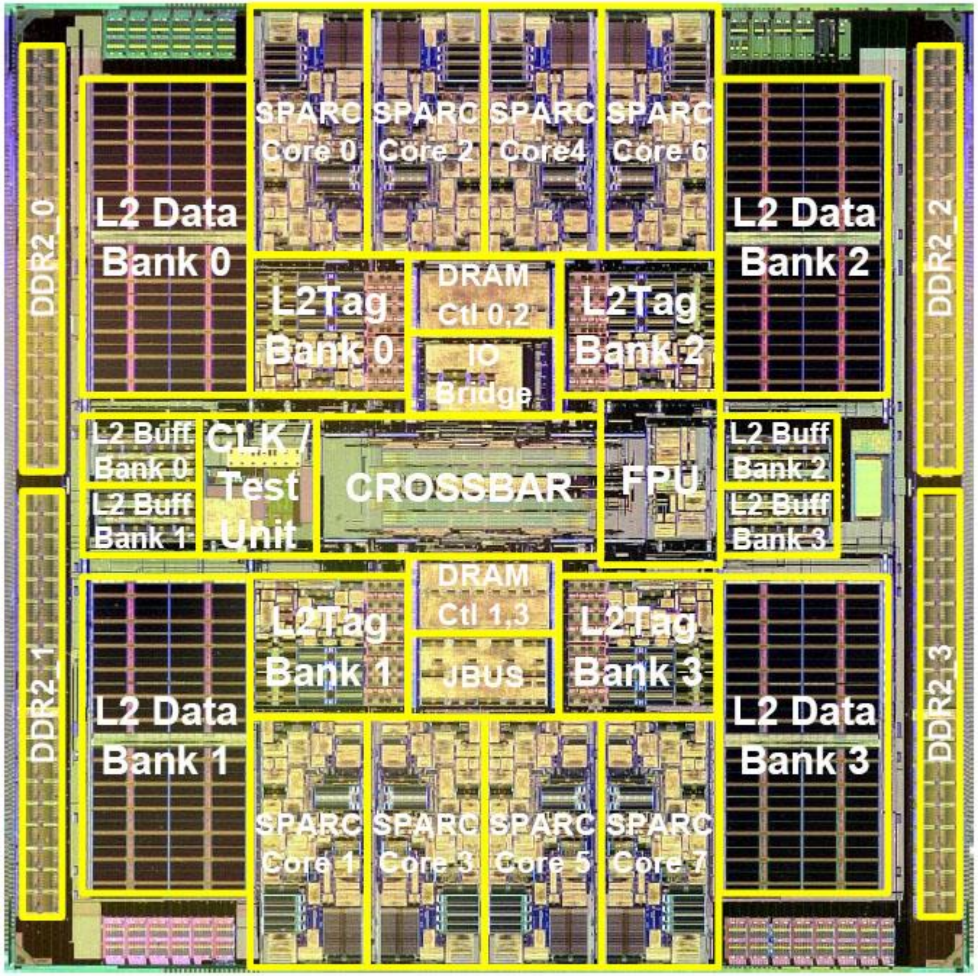
Διαφάνειες από τον Ανδρέα Μόσχοβο, University of Toronto

8ο εξάμηνο ΣΗΜΜΥ – Ακαδημαϊκό Έτος: 2019-20

http://www.cslab.ece.ntua.gr/courses/advcomparch/

Modern Processors Have Lots of Cores and Large Caches

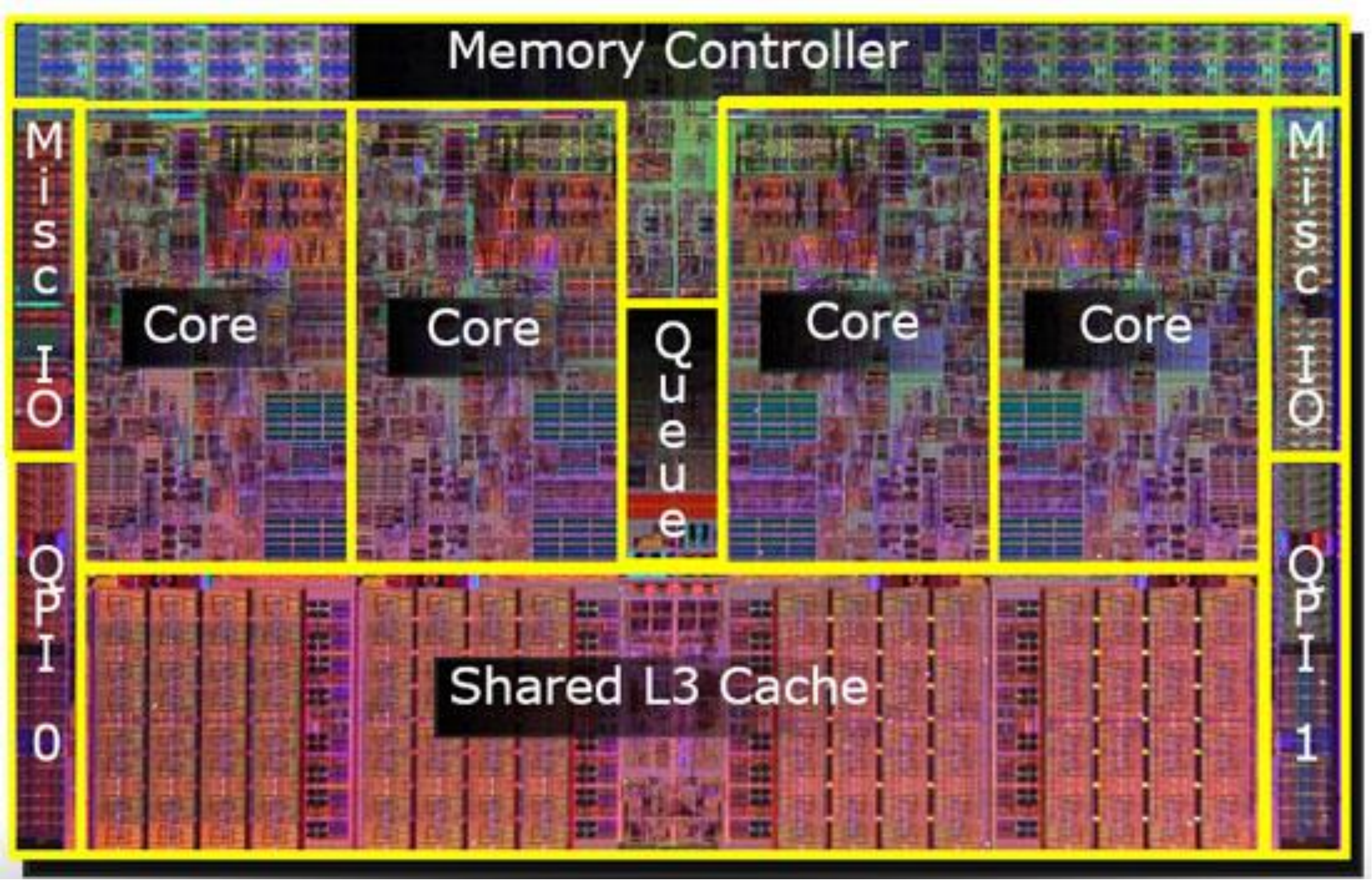
- Sun Niagara T1



From http://jinsatoh.jp/ennui/archives/2006/03/opensparc.html

Modern Processors Have Lots of Cores and Large Caches

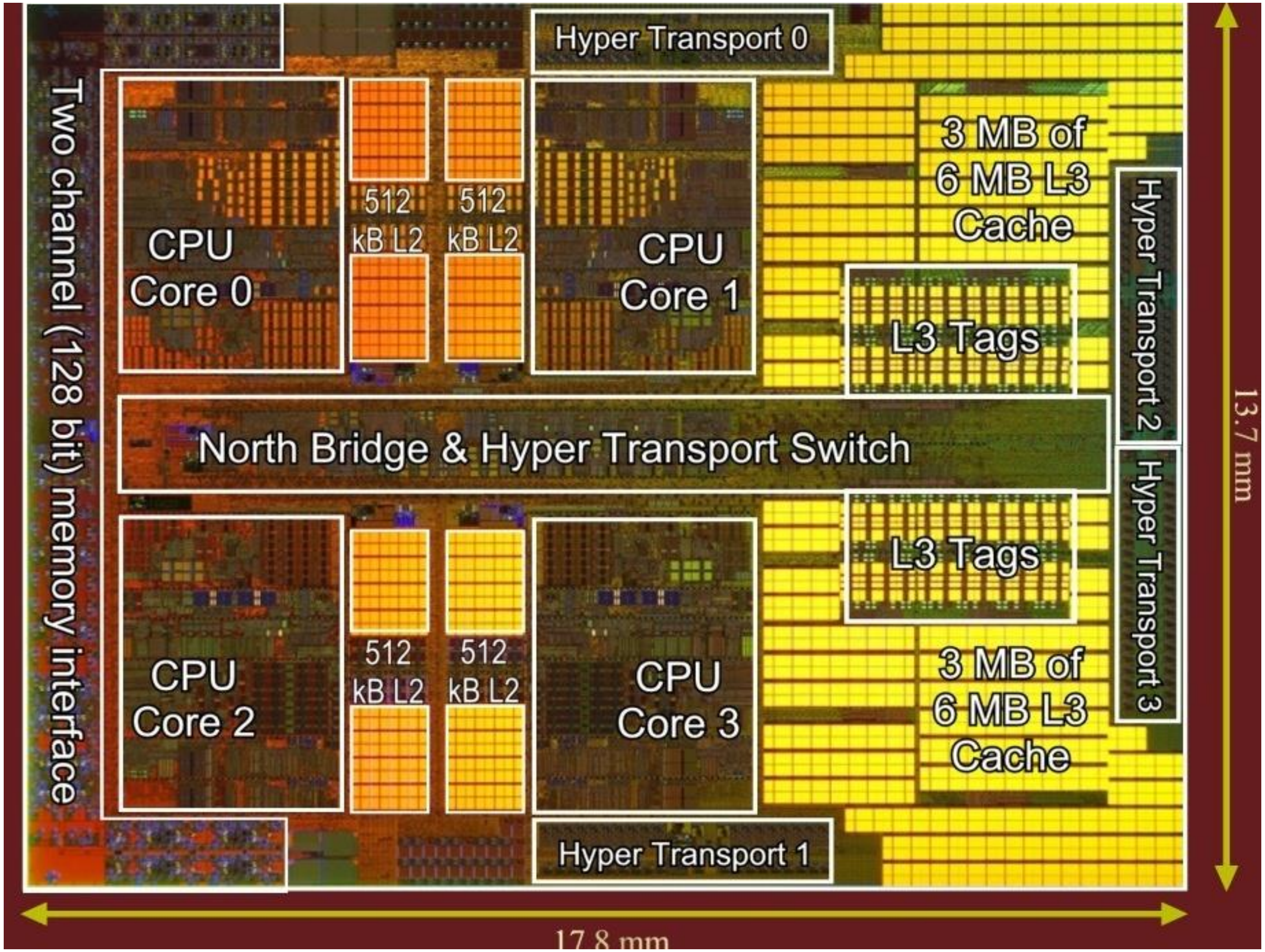
- Intel i7 (Nehalem)



From http://www.legitreviews.com/article/824/1/

Modern Processors Have Lots of Cores and Large Caches

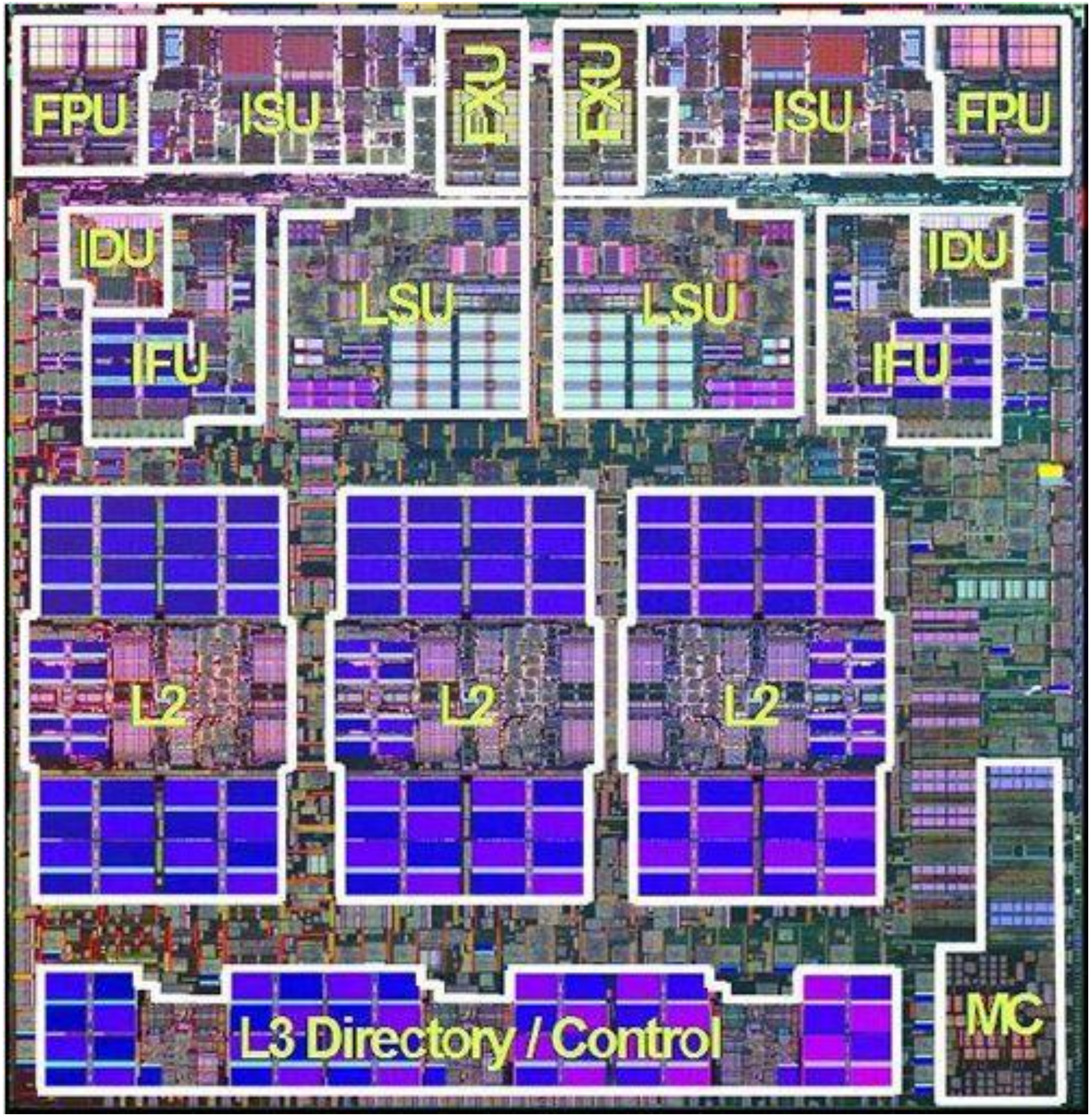
- AMD Shanghai



From http://www.chiparchitect.com

Modern Processors Have Lots of Cores and Large Caches

- IBM Power 5

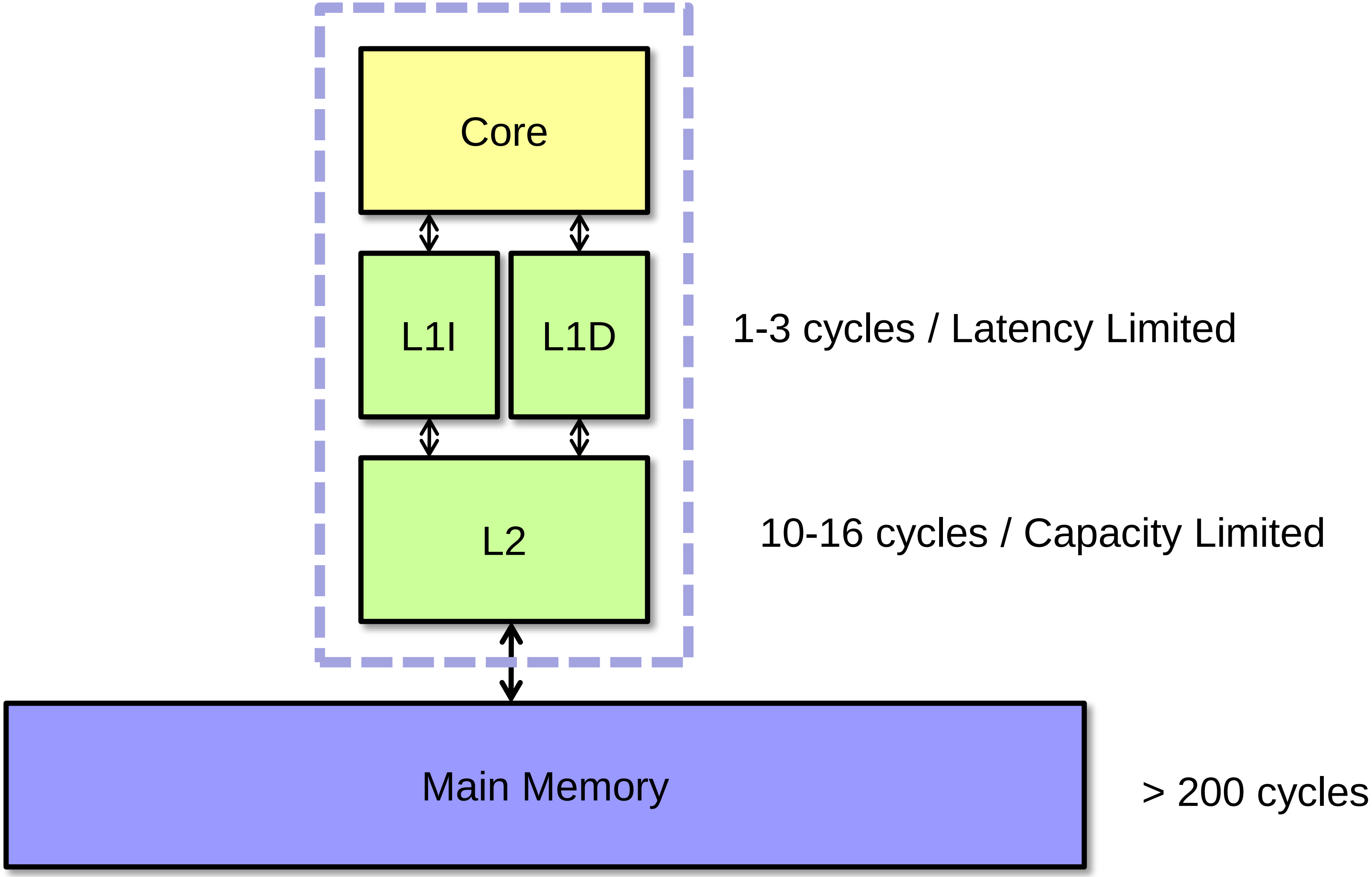


From http://www.theinquirer.net/inquirer/news/1018130/ibms-power5-the-multi-chipped-monster-mcm-revealed

Why?

- Helps with Performance and Energy
  - Find graph with perfect vs. realistic memory system

What Cache Design Used to be About



- L2: Worst Latency == Best Latency
- **Key Decision: What to keep in each cache level**

What Has Changed

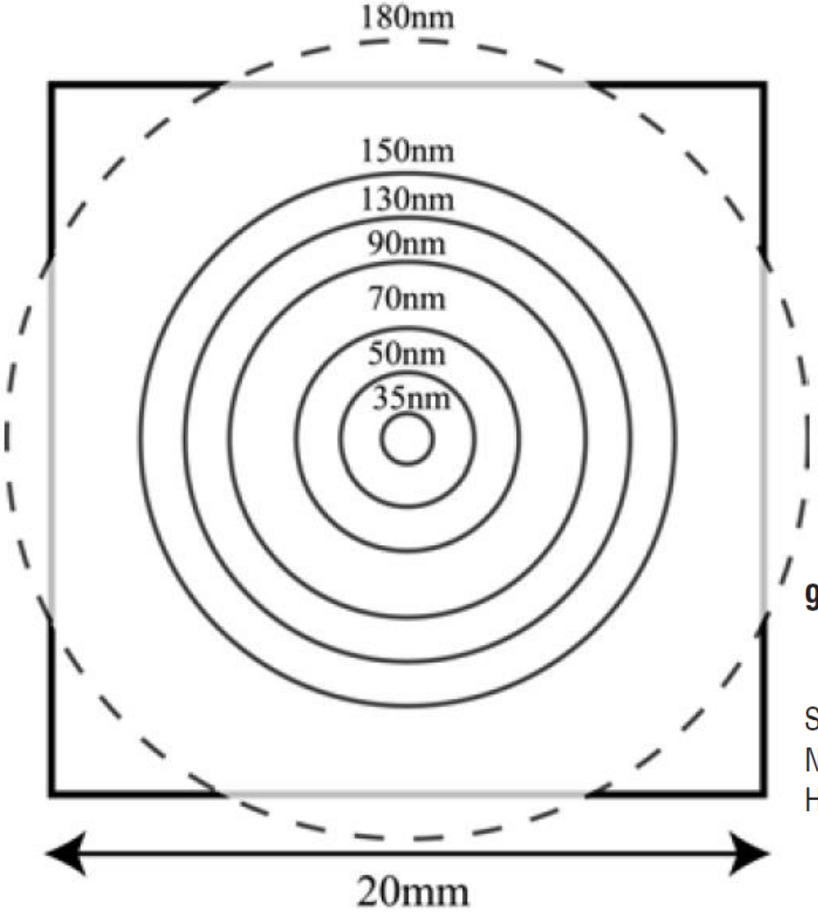


Figure 9.6.2: Projected fraction of chip reachable in one cycle with an 8FO4 clock period.

ISSCC 2003

9.6 A Wire-Delay Scalable Microprocessor Architecture for High Performance Systems

Stephen W. Keckler[1], Doug Burger[1], Charles R. Moore[1], Ramadass Nagarajan[1], Karthikeyan Sankaralingam[1], Vikas Agarwal[2], M.S. Hrishikesh[2], Nitya Ranganathan[1], Premkishore Shivakumar[1]

What Has Changed

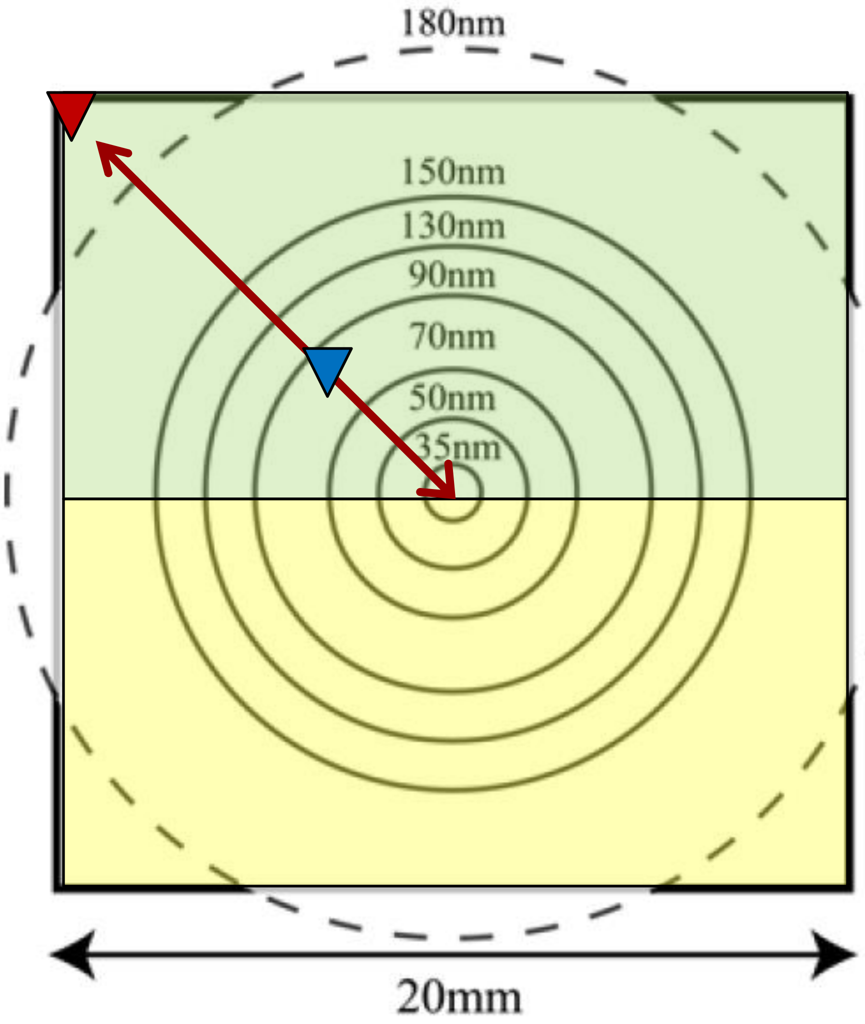
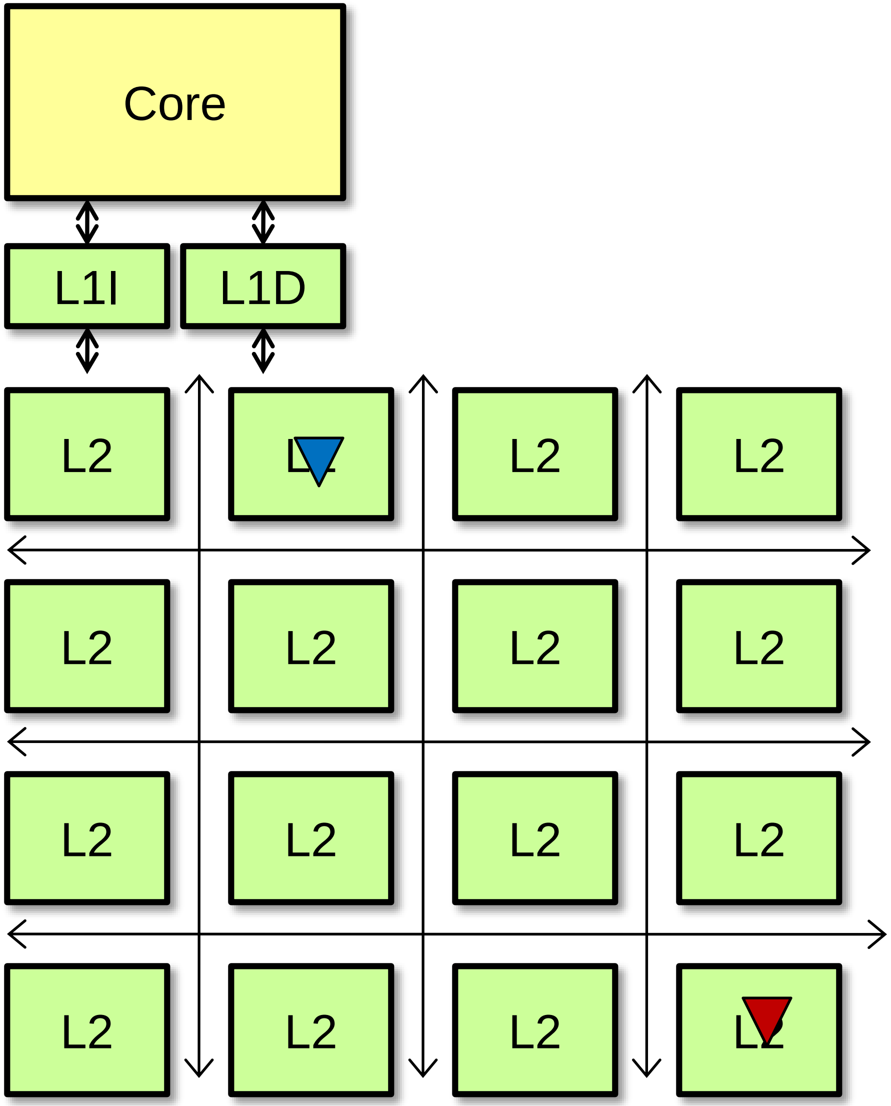


Figure 9.6.2: Projected fraction of chip reachable in one cycle with an 8FO4 clock period.

- **Where something is matters**
- **More time for longer distances**

NUCA: Non-Uniform Cache Architecture



- Tiled Cache
- Variable Latency
- Closer tiles = Faster

- **Key Decisions:**
  - **Not only *what* to cache**
  - **Also *where* to cache**

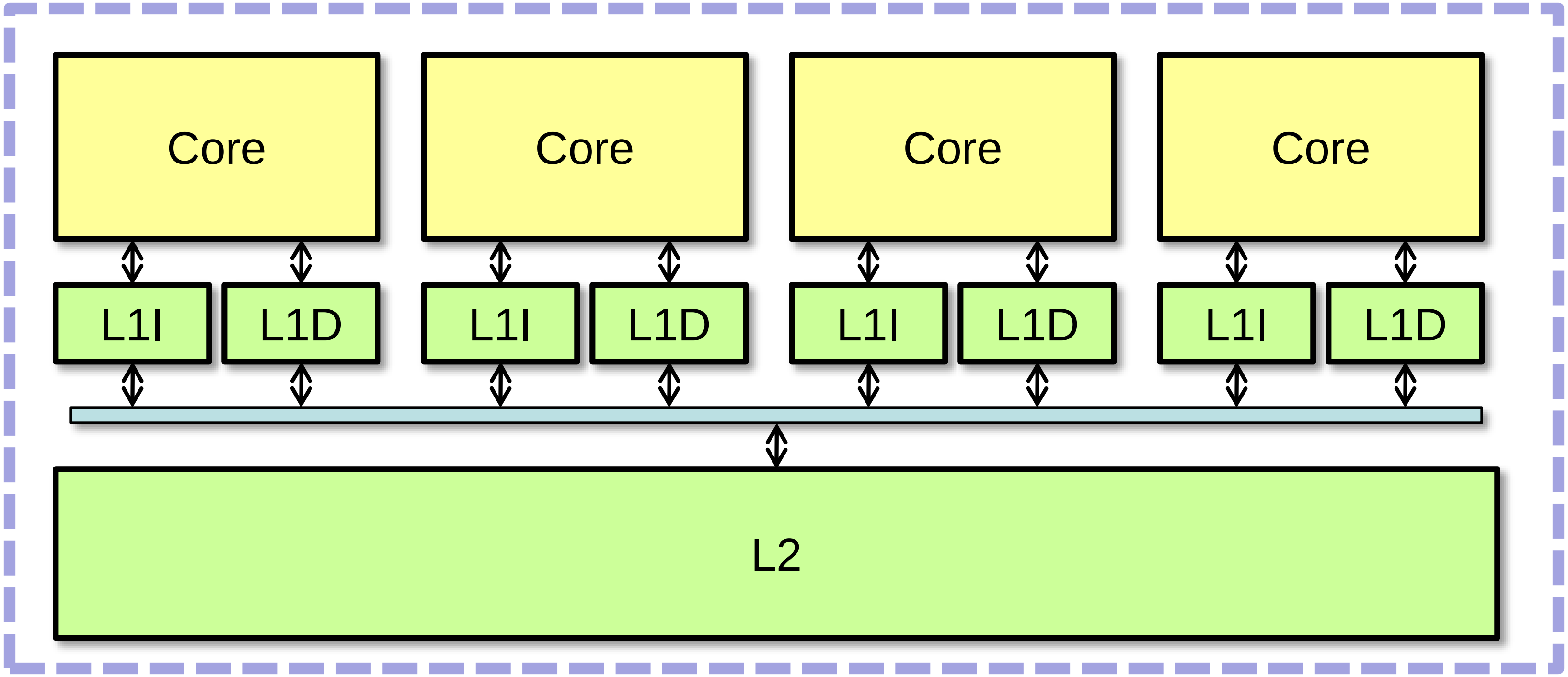
NUCA Overview

- Initial Research focused on Uniprocessors

- Data Migration Policies
  - When to move data among tiles

- L-NUCA: Fine-Grained NUCA

Another Development: Chip Multiprocessors

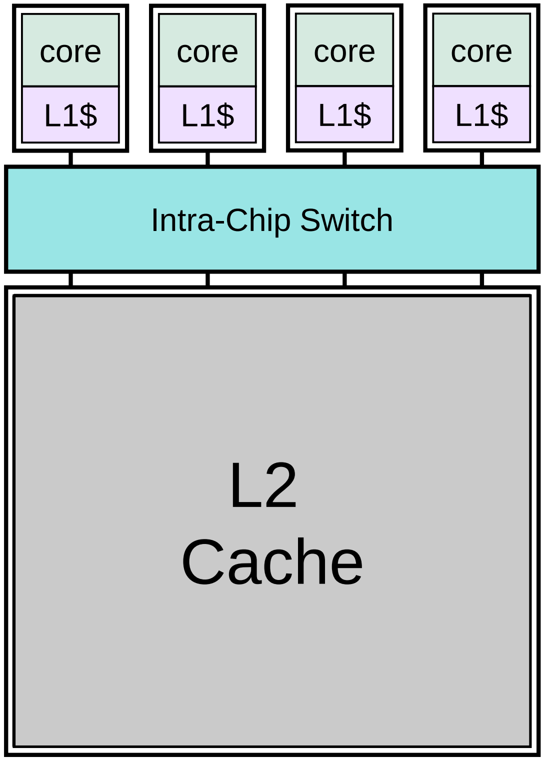


- Easily utilize on-chip transistors
- Naturally exploit thread-level parallelism
- Dramatically reduce design complexity

- Future CMPs will have more processor cores
- Future CMPs will have more cache

Text from Michael Zhang & Krste Asanovic, MIT

Initial Chip Multiprocessor Designs

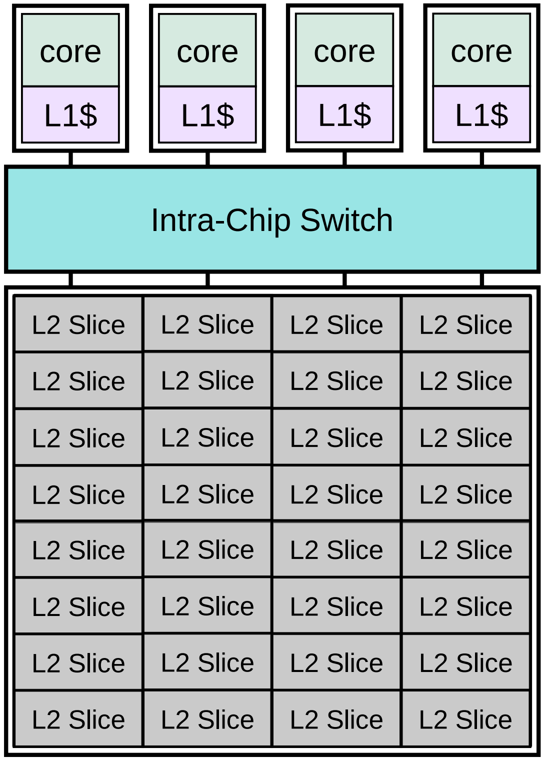


A 4-node CMP with a large L2 cache

- *Layout*: “Dance-Hall”
  - *Core + L1 cache*
  - *L2 cache*
- *Small L1 cache:* Very low access latency
- *Large L2 cache*

Slide from Michael Zhang & Krste Asanovic, MIT

Chip Multiprocessor w/ Large Caches

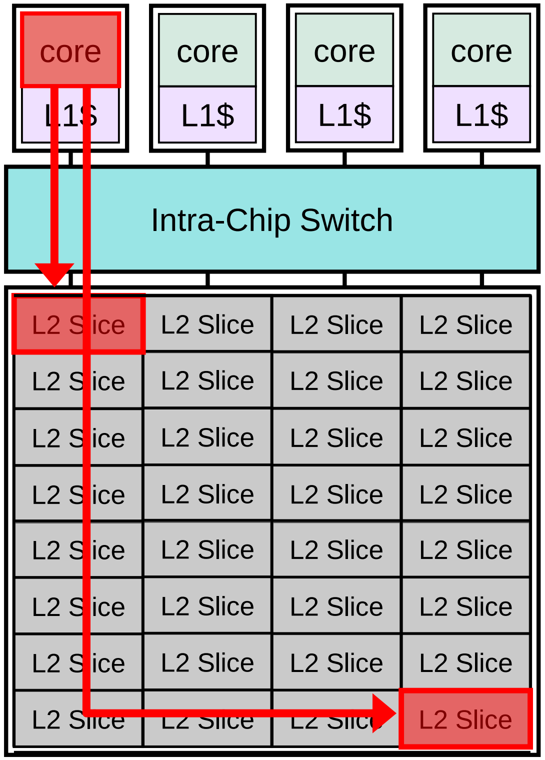


A 4-node CMP with a large L2 cache

- *Layout*: “Dance-Hall”
  - *Core + L1 cache*
  - *L2 cache*
- *Small L1 cache:* Very low access latency
- *Large L2 cache:* Divided into slices to minimize access latency and power usage

Slide from Michael Zhang & Krste Asanovic, MIT

Chip Multiprocessors + NUCA



core L1$ | core L1$ | core L1$ | core L1$

Intra-Chip Switch

L2 Slice	L2 Slice	L2 Slice	L2 Slice
L2 Slice	L2 Slice	L2 Slice	L2 Slice
L2 Slice	L2 Slice	L2 Slice	L2 Slice
L2 Slice	L2 Slice	L2 Slice	L2 Slice
L2 Slice	L2 Slice	L2 Slice	L2 Slice
L2 Slice	L2 Slice	L2 Slice	L2 Slice
L2 Slice	L2 Slice	L2 Slice	L2 Slice
L2 Slice	L2 Slice	L2 Slice	L2 Slice

A 4-node CMP with a large L2 cache

- *Current:* Caches are designed with (long) uniform access latency for the worst case:

  Best Latency == Worst Latency

- *Future:* Must design with non-uniform access latencies depending on the on-die location of the data:

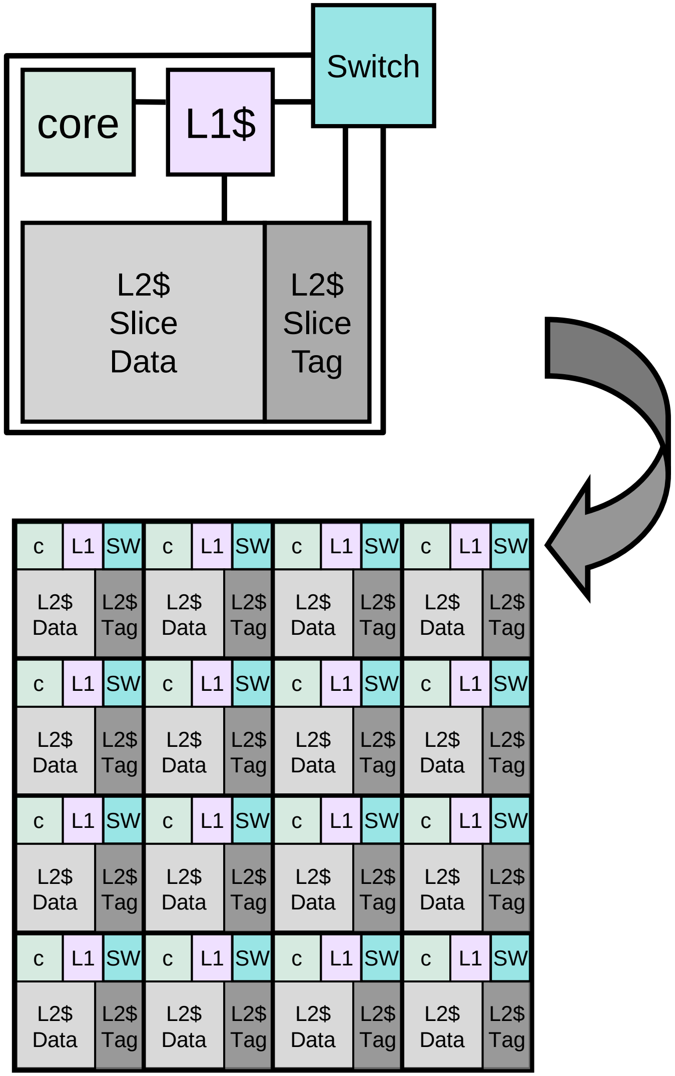
  Best Latency << Worst Latency

- *Challenge:* How to minimize average cache access latency:

  Average Latency → Best Latency

Slide from Michael Zhang & Krste Asanovic, MIT

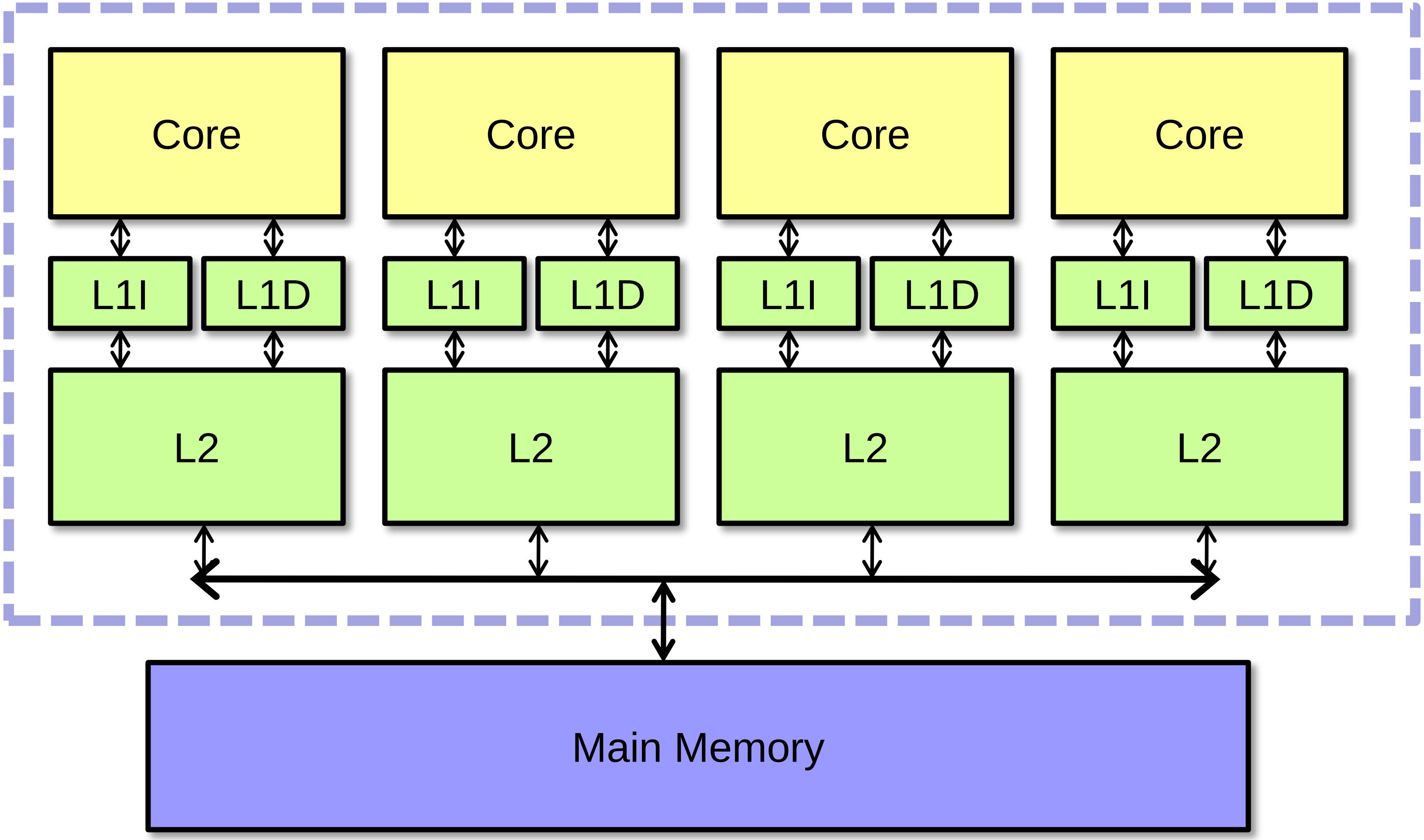
Tiled Chip Multiprocessors


Switch
core
L1$
L2$ Slice Data
L2$ Slice Tag
c L1 SW c L1 SW c L1 SW c L1 SW
L2$ Data L2$ Tag L2$ Data L2$ Tag L2$ Data L2$ Tag L2$ Data L2$ Tag
c L1 SW c L1 SW c L1 SW c L1 SW
L2$ Data L2$ Tag L2$ Data L2$ Tag L2$ Data L2$ Tag L2$ Data L2$ Tag
c L1 SW c L1 SW c L1 SW c L1 SW
L2$ Data L2$ Tag L2$ Data L2$ Tag L2$ Data L2$ Tag L2$ Data L2$ Tag
c L1 SW c L1 SW c L1 SW c L1 SW
L2$ Data L2$ Tag L2$ Data L2$ Tag L2$ Data L2$ Tag L2$ Data L2$ Tag

- Tiled CMPs for *Scalability*
  - *Minimal redesign effort*
  - *Use directory-based protocol for scalability*
- Managing the L2s to minimize the effective access latency
  - *Keep data close to the requestors*
  - *Keep data on-chip*

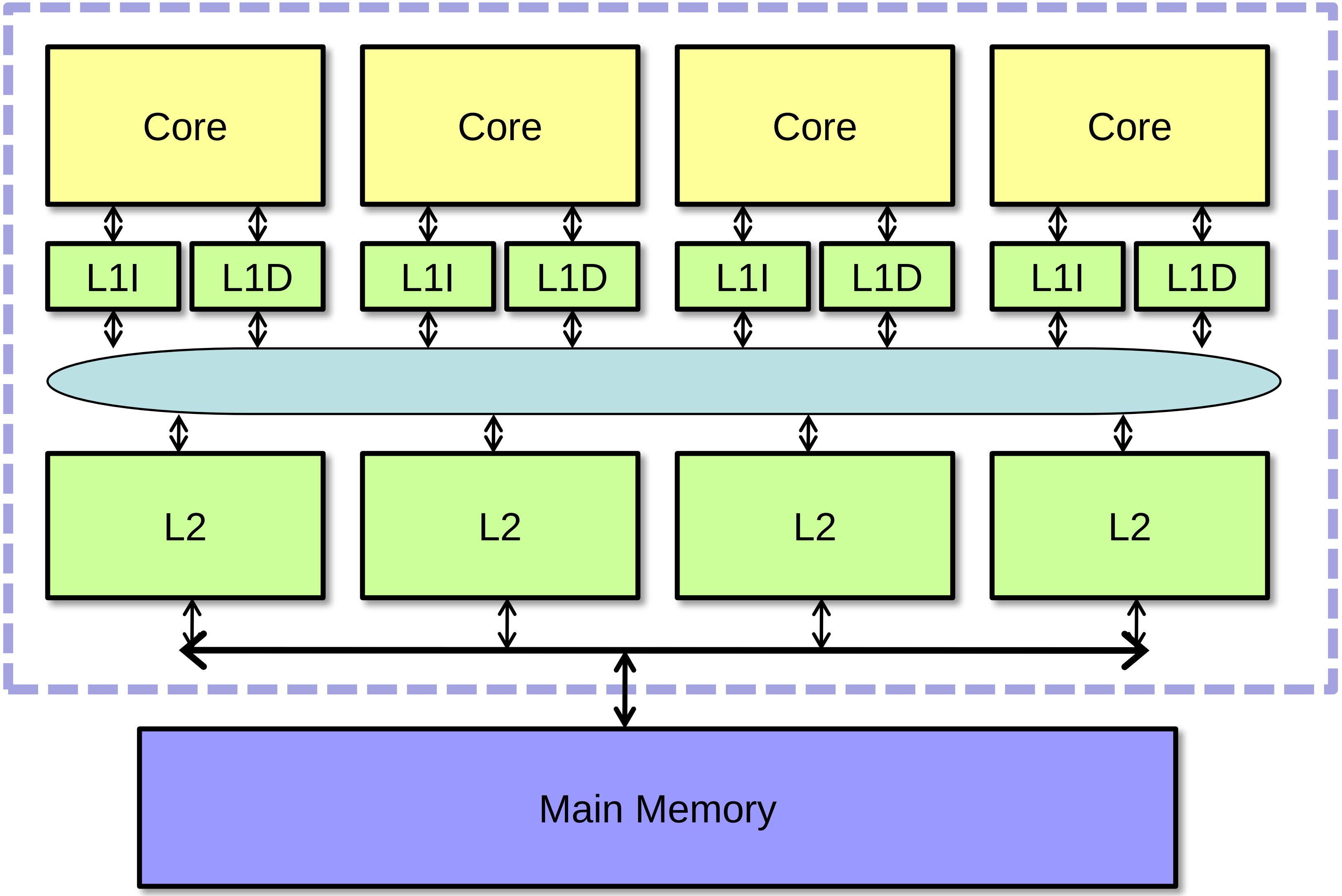
Slide from Michael Zhang & Krste Asanovic, MIT

Option #1: Private Caches



- + Low Latency
- - Fixed allocation

Option #2: Shared Caches



- Higher, variable latency
- One Core can use all of the cache

Data Cache Management for CMP Caches

- Get the best of both worlds
  - Low Latency of Private Caches
  - Capacity Adaptability of Shared Caches

NUCA: A Non-Uniform Cache Access Architecture for Wire-Delay Dominated On-Chip Caches

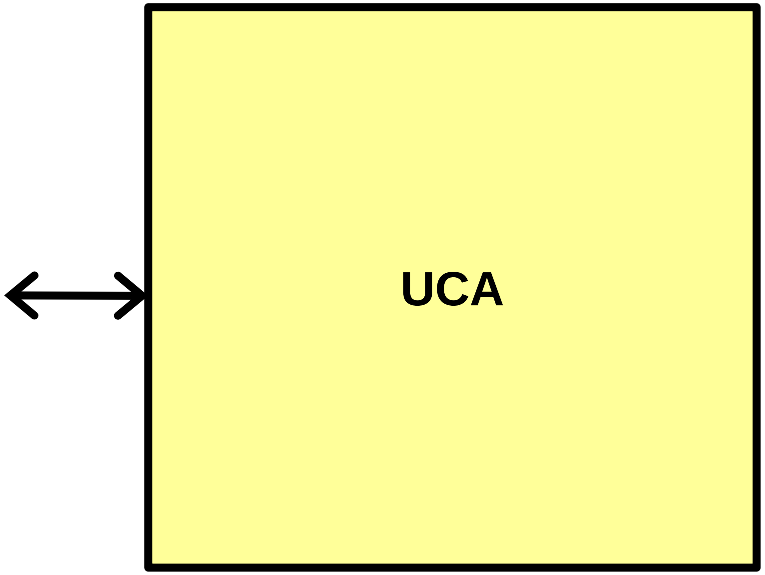
Changkyu Kim, D.C. Burger, and S.W. Keckler,

10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X), October, 2002.

Some material from slides by Prof. Hsien-Hsin S. Lee ECE, GTech

Conventional – Monolithic Cache

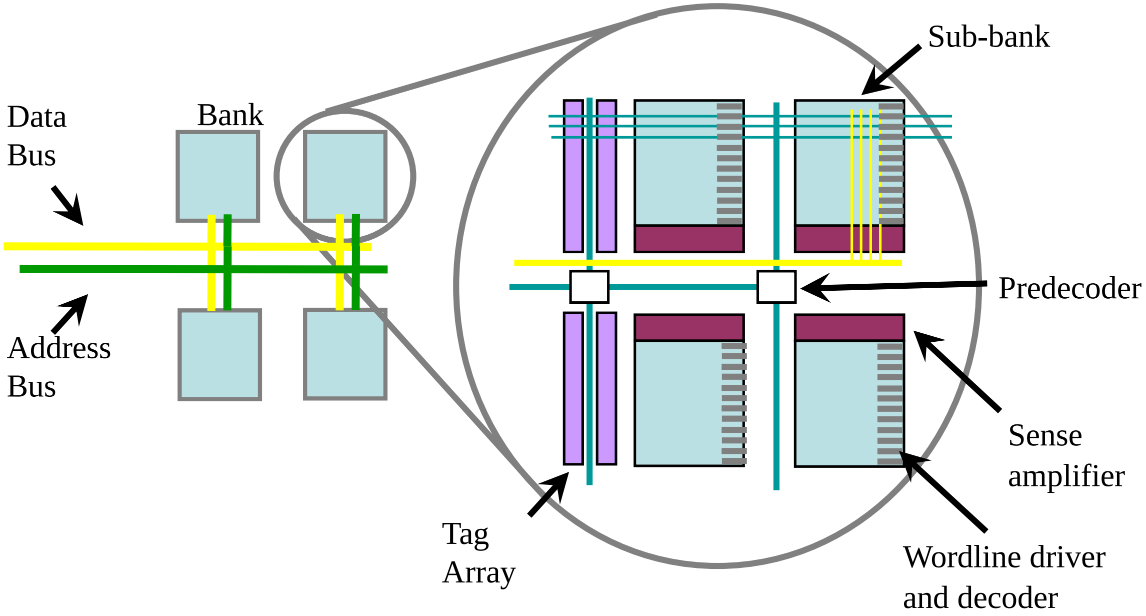
- **UCA:** Uniform Access Cache



- **Best Latency = Worst Latency**
  - **Time to access the farthest possible bank**

UCA Design

- **Partitioned in Banks**



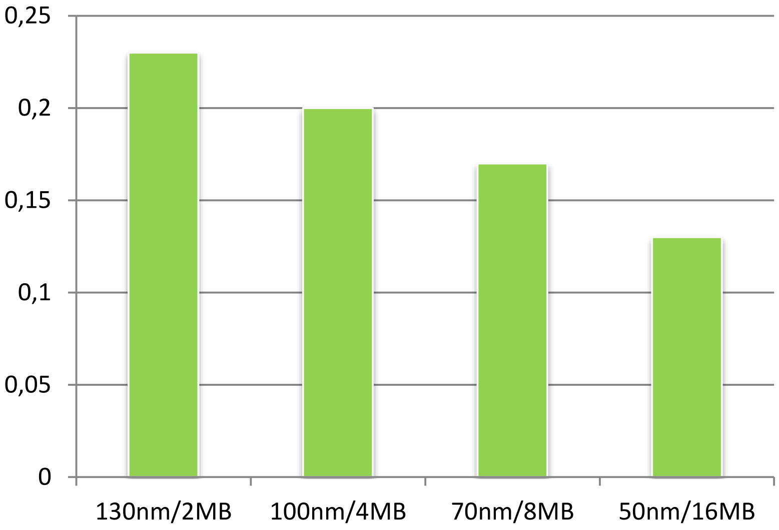
- Conceptually a single address and a single data bus
  - Pipelining can increase throughput
- See **CACTI** tool:
  - http://www.hpl.hp.com/research/cacti/
  - http://quid.hpl.hp.com:9081/cacti/

Experimental Methodology

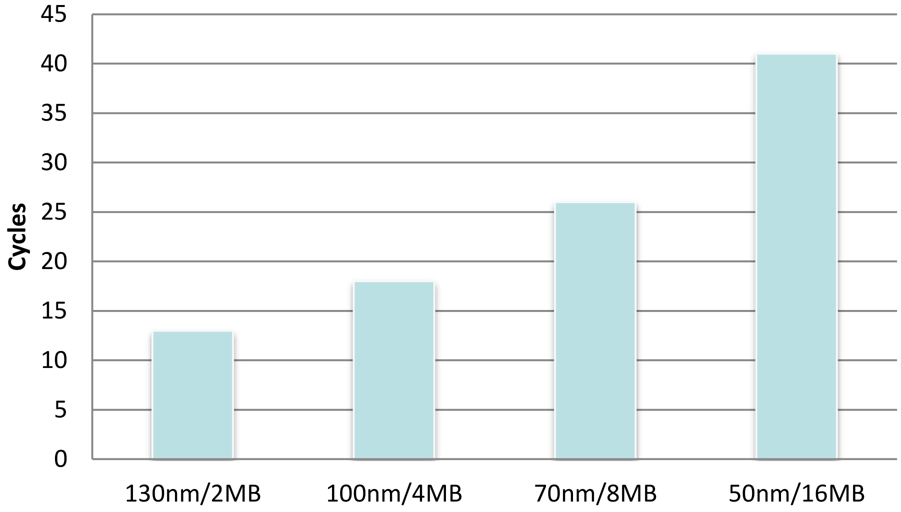
- SPEC CPU 2000
- Sim-Alpha
- CACTI
- 8 FO4 cycle time
- 132 cycles to main memory
- Skip and execute a sample
- Technology Nodes
  - 130nm, 100nm, 70nm, 50nm

UCA Scaling – 130nm to 50nm

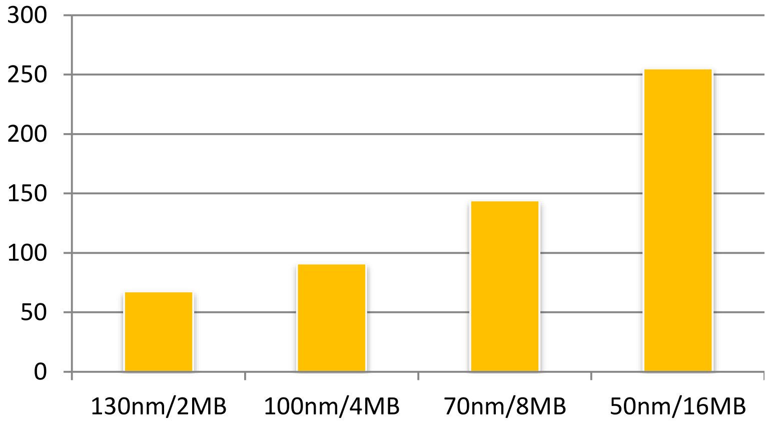
Miss Rate



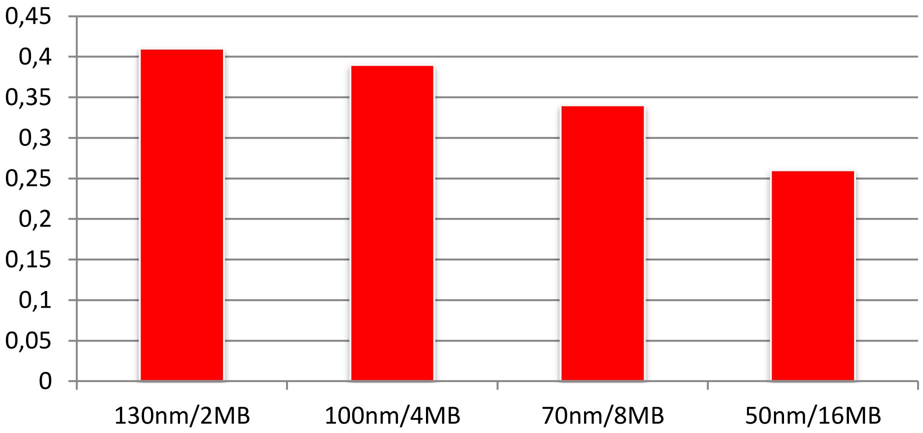
Unloaded Latency



Loaded Latency



IPC



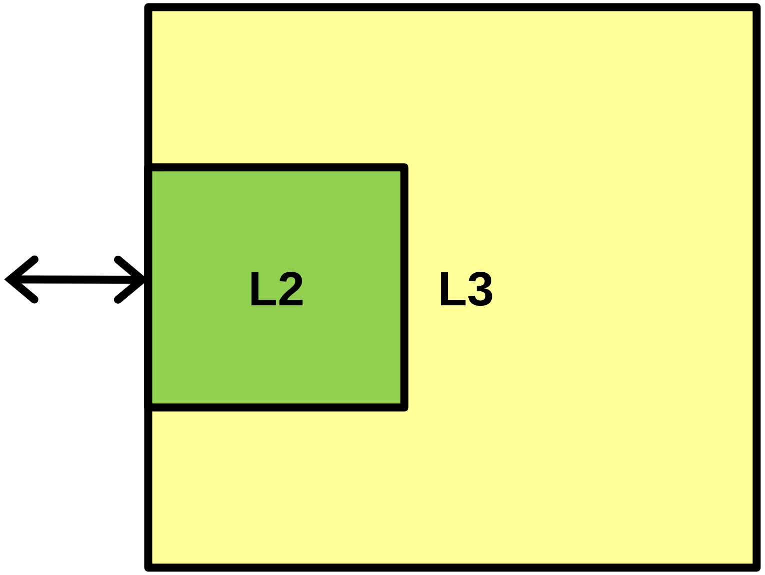
Relative Latency and Performance Degrade as Technology Improves

UCA Discussion

- Loaded Latency: Contention
  - Bank
  - Channel
    - Bank may be free but path to it is not

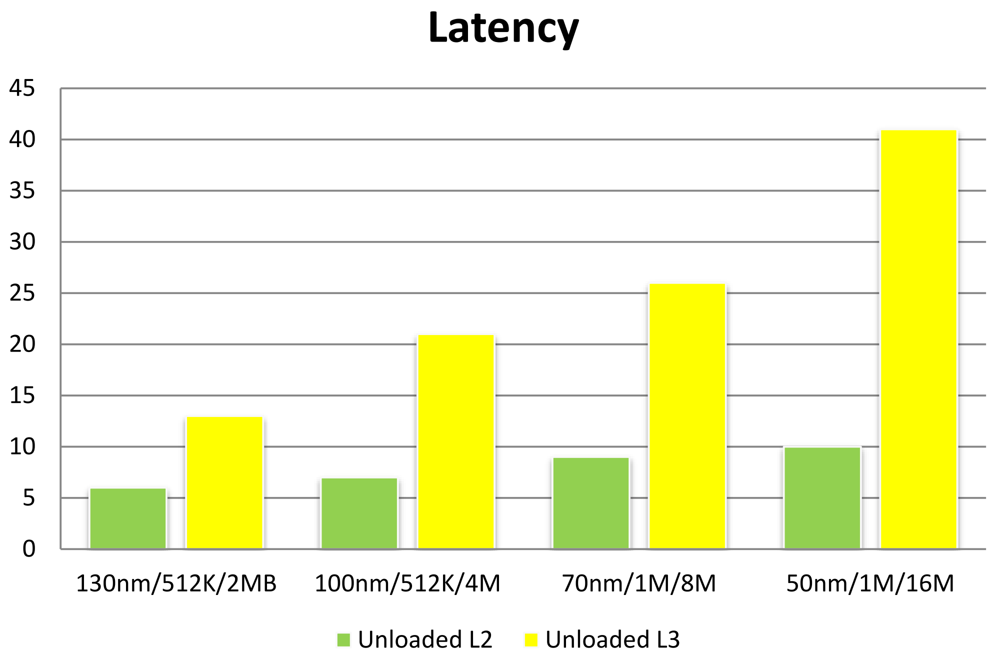
Multi-Level Cache

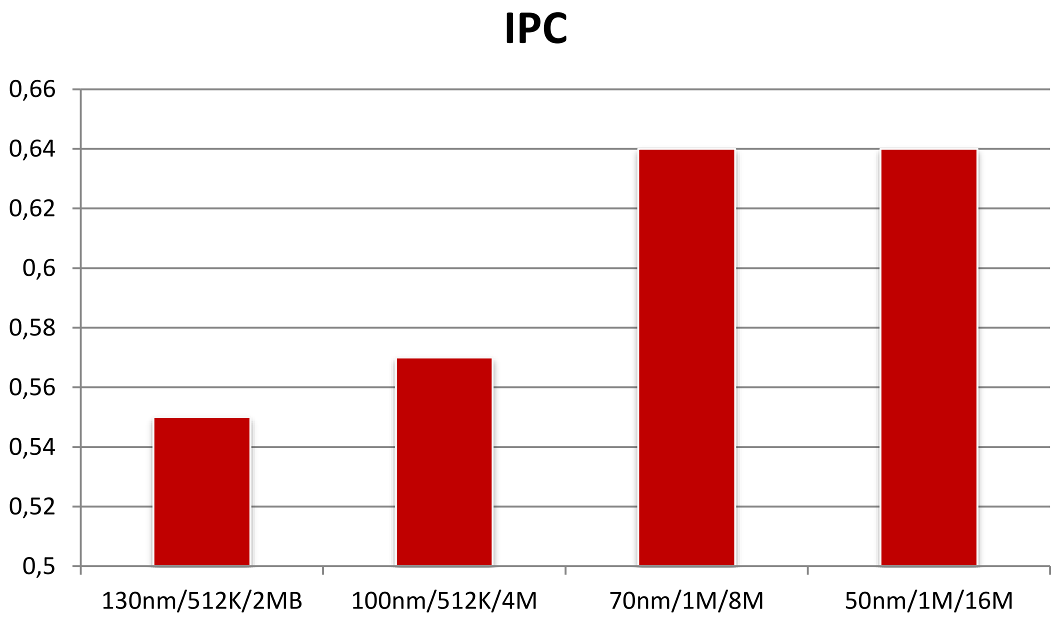
- Conventional Hierarchy



- Common Usage:
  - **Serial-Access** for Energy and Bandwidth Reduction
- This paper:
  - **Parallel Access**
  - Prove that even then their design is better

ML-UCA Evaluation



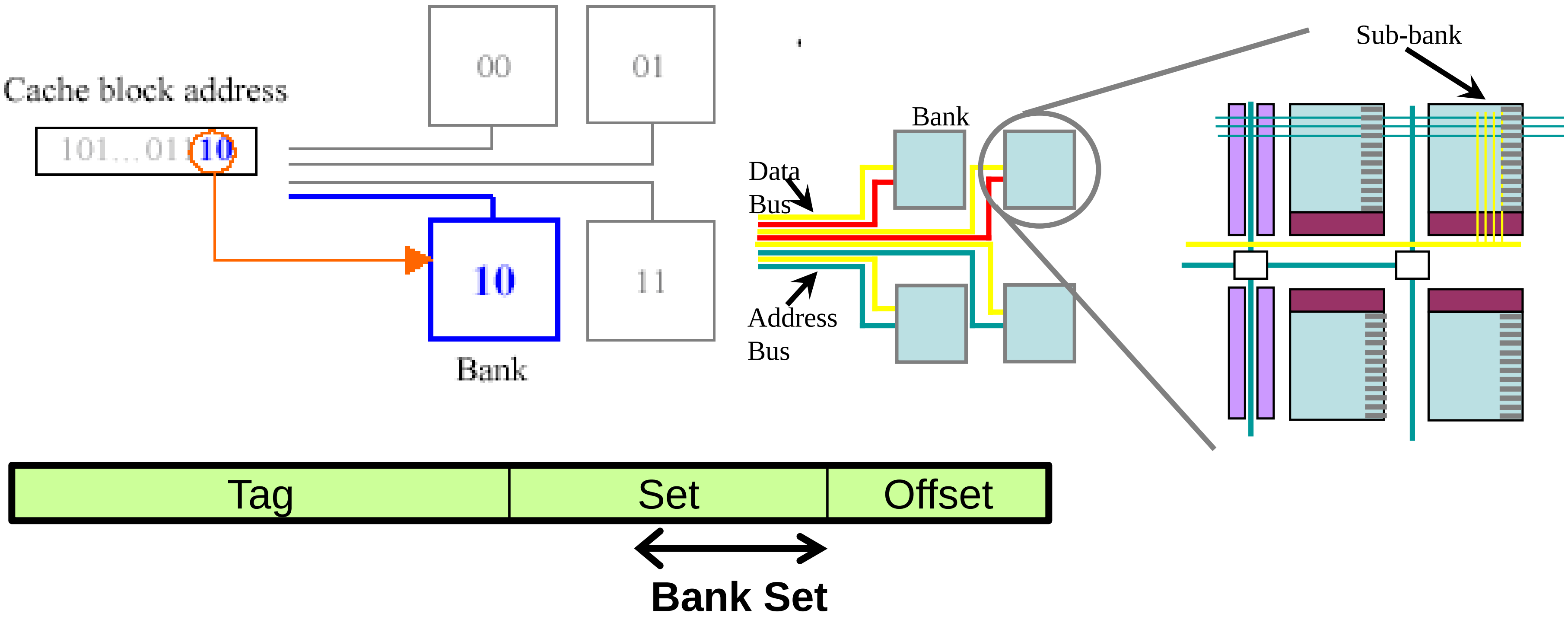


- Better than UCA
- Performance Saturates at 70nm
- No benefit from larger cache at 50nm

- Aggressively banked
- Multiple parallel accesses

S-NUCA-1

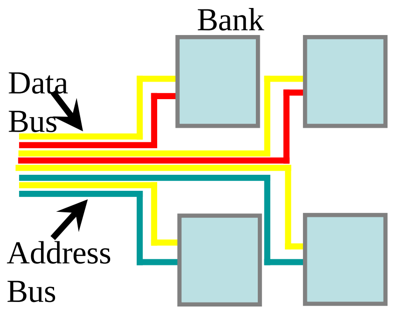
- Static NUCA with per bank set busses



- Use private **per bank set** channel
- Each bank has its distinct access latency
- A given address maps to a given bank set
    - Lower bits of **block address**

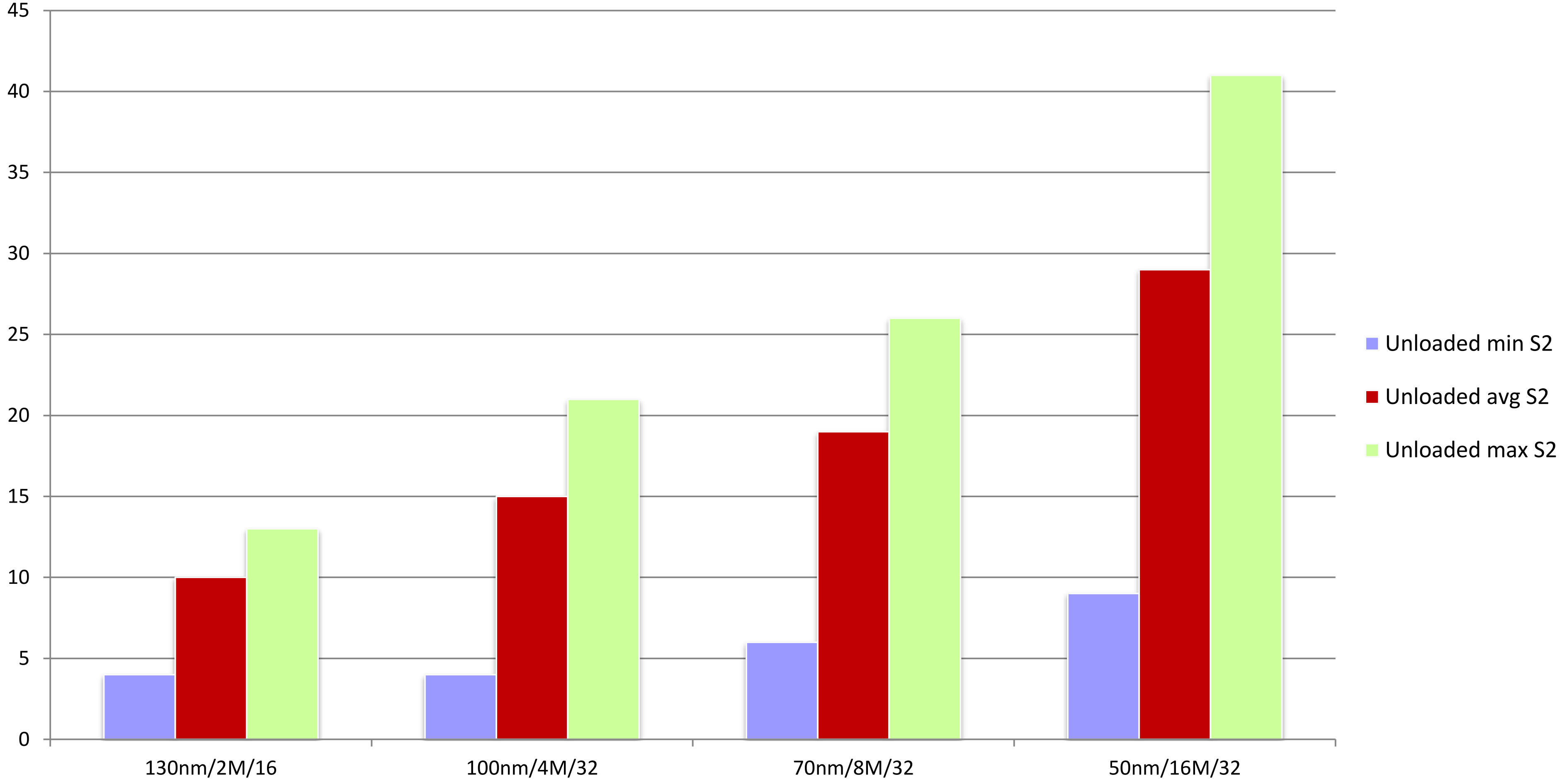
S-NUCA-1

- How fast can we initiate requests?
  - If c = scheduler delay



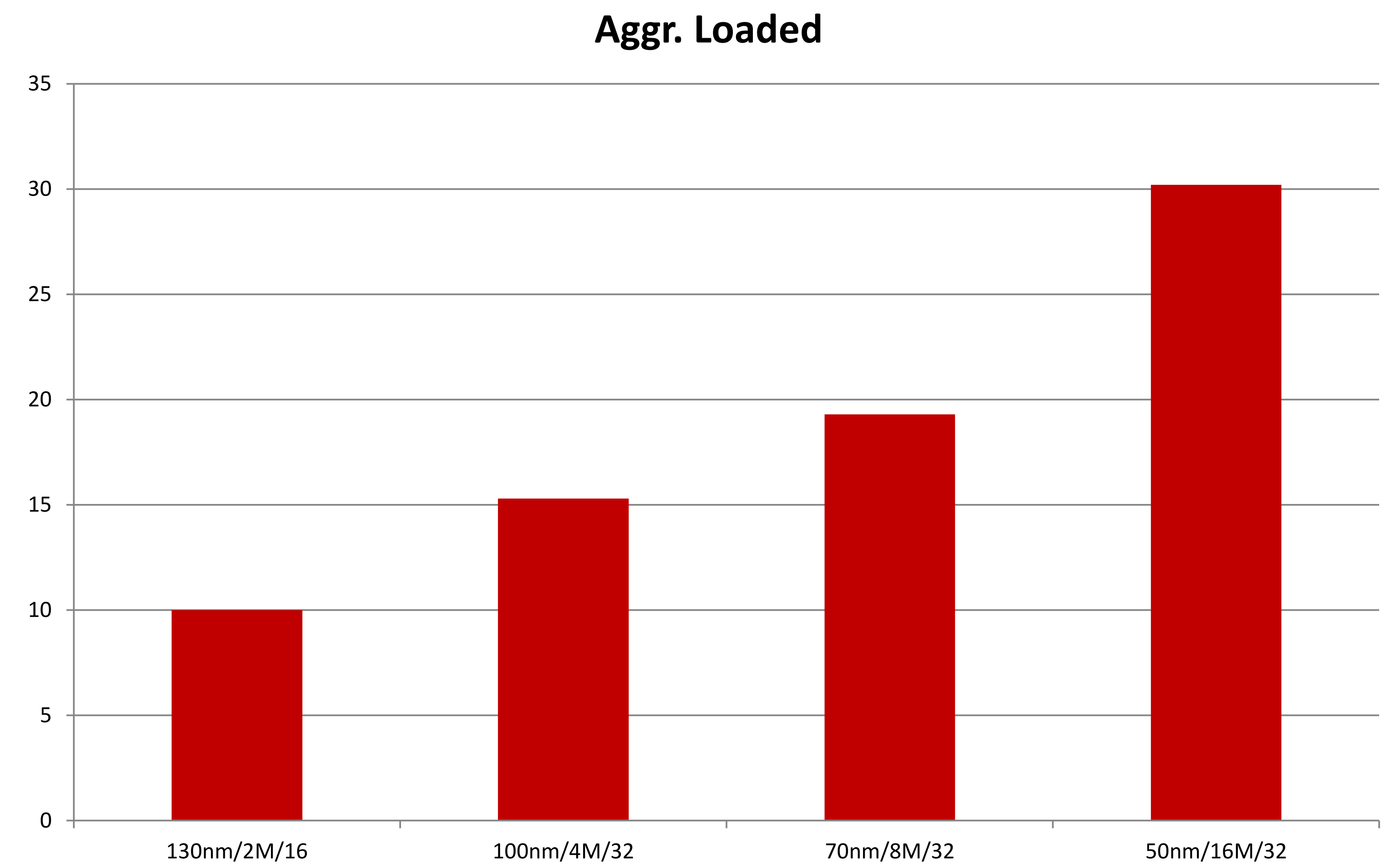
- Conservative **/Realistic:**
  - Bank + 2 x interconnect + c
- Aggressive **/ Unrealistic**:
  - Bank + c
- What is the optimal number of bank sets?
  - Exhaustive evaluation of all options
  - Which gives the highest IPC

S-NUCA-1 Latency Variability



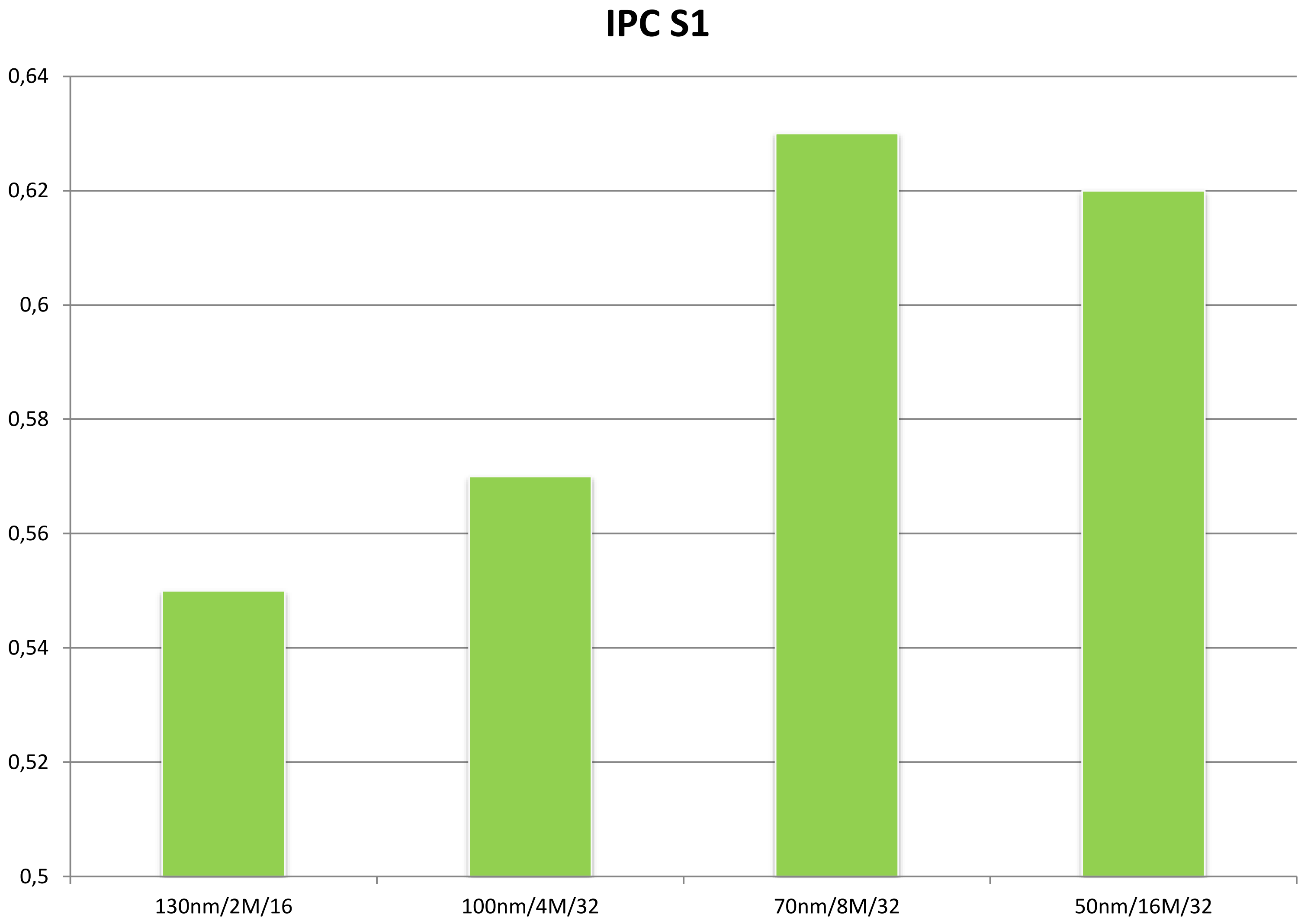
- Variability increases for finer technologies
- Number of banks does not increase beyond 4M
  - Overhead of additional channels
  - Banks become larger and slower

S-NUCA-1 Loaded Latency



- Better than ML-UCA

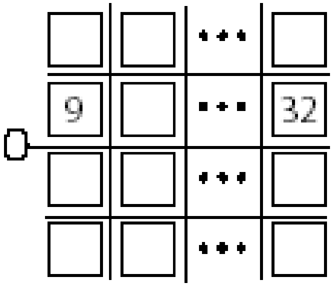
S-NUCA-1: IPC Performance

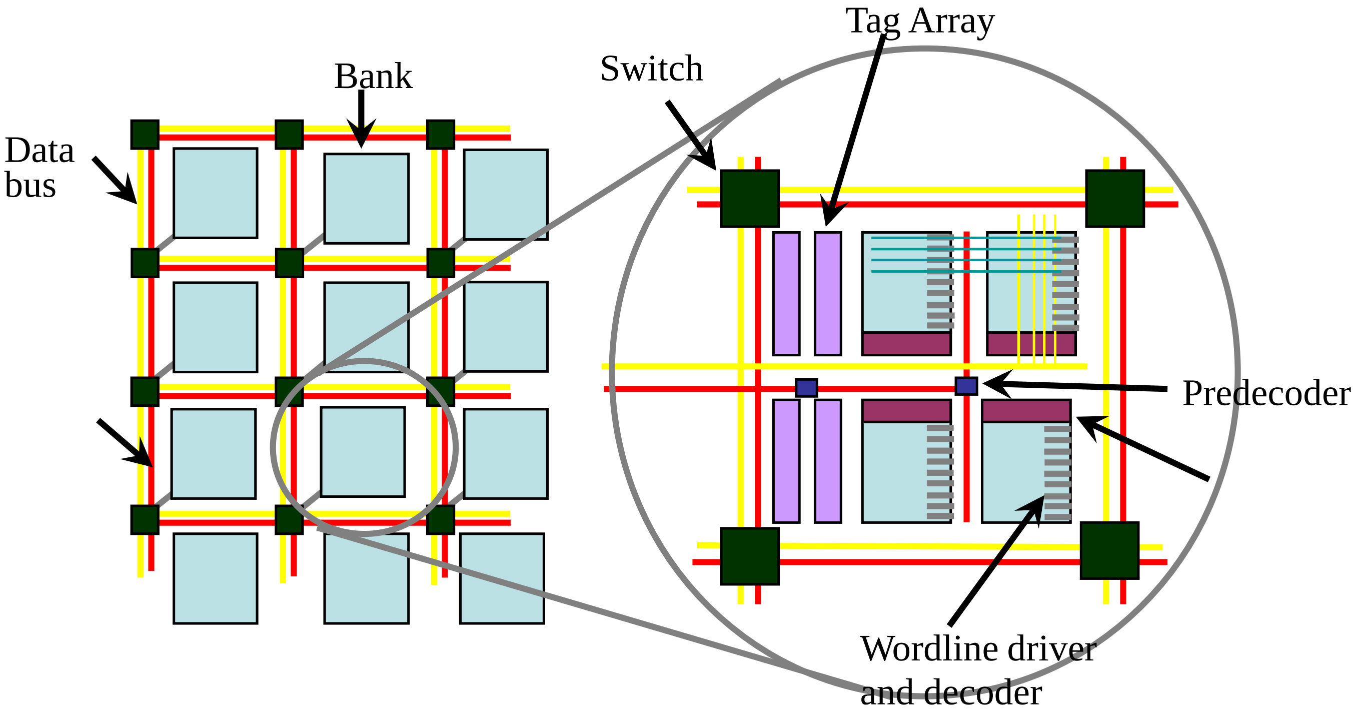


- Per bank channels become an overhead
- Prevent finer partitioning @70nm or smaller

S-NUCA2

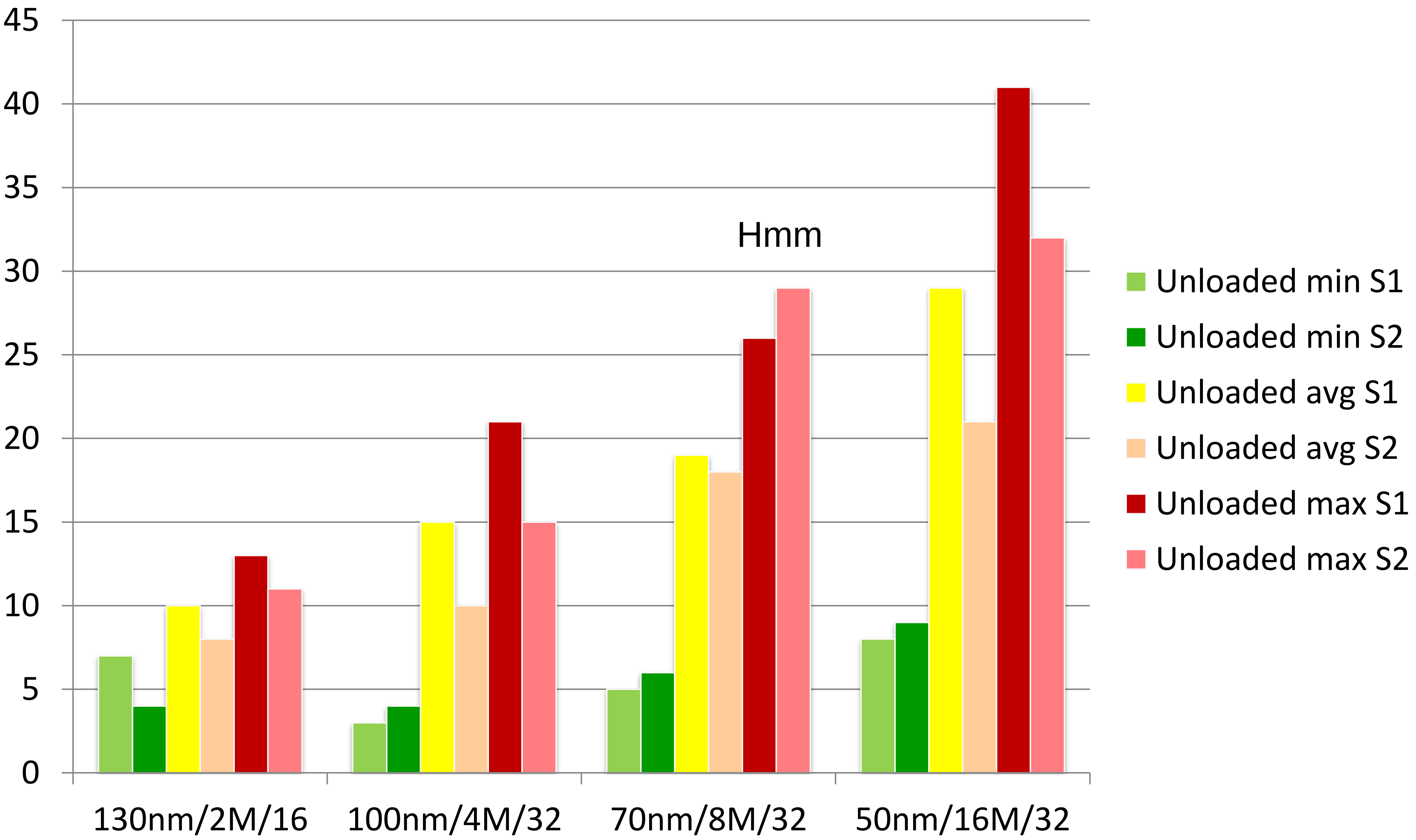
- Use a 2-D Mesh P2P interconnect





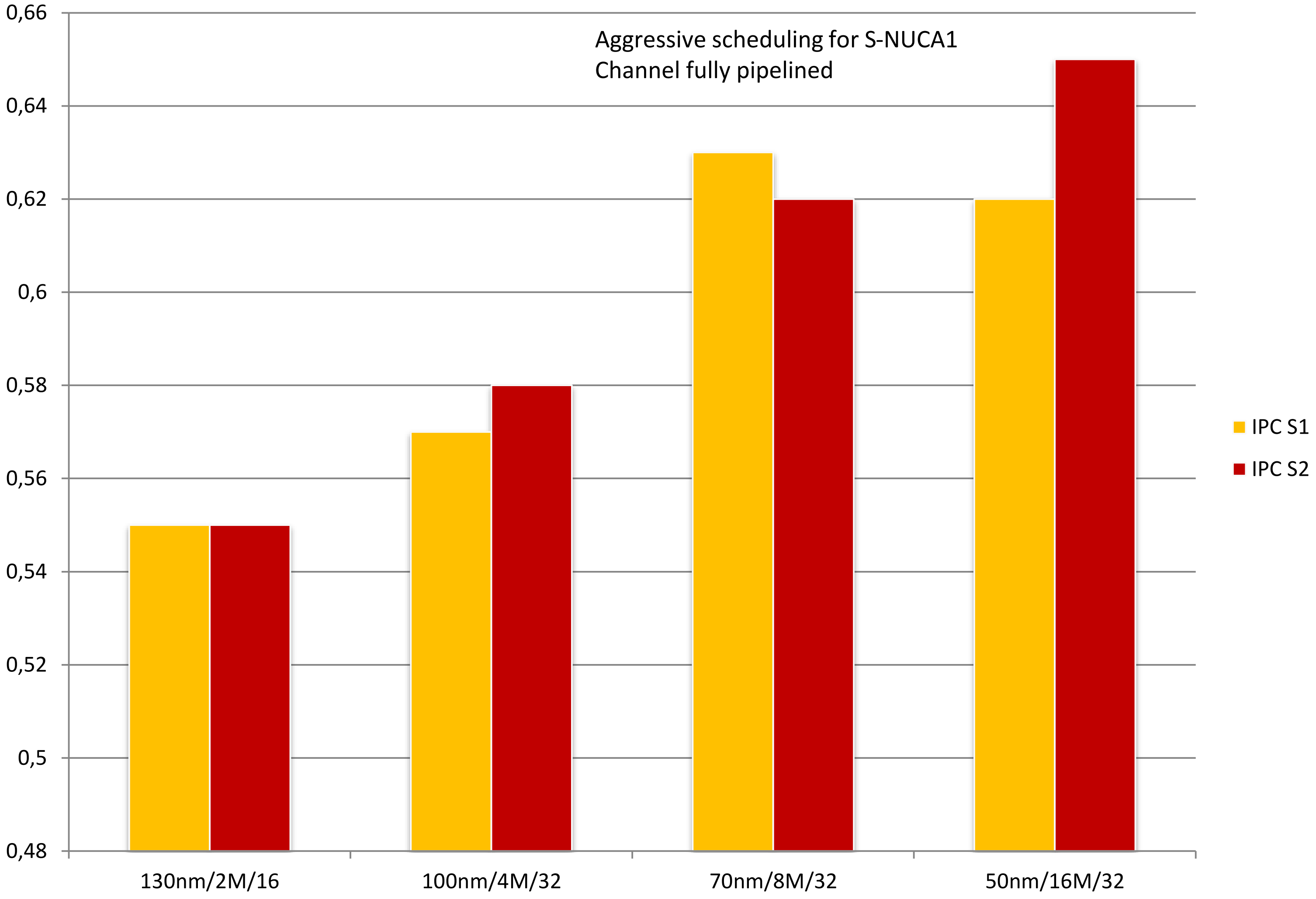
- Wire overhead much lower:
  - S1: 20.9% vs. S2: 5.9% at 50nm and 32banks
- Reduces contention
- 128-bit bi-directional links

S-NUCA2 vs. S-NUCA1 Unloaded Latency



- S-NUCA2 almost always better

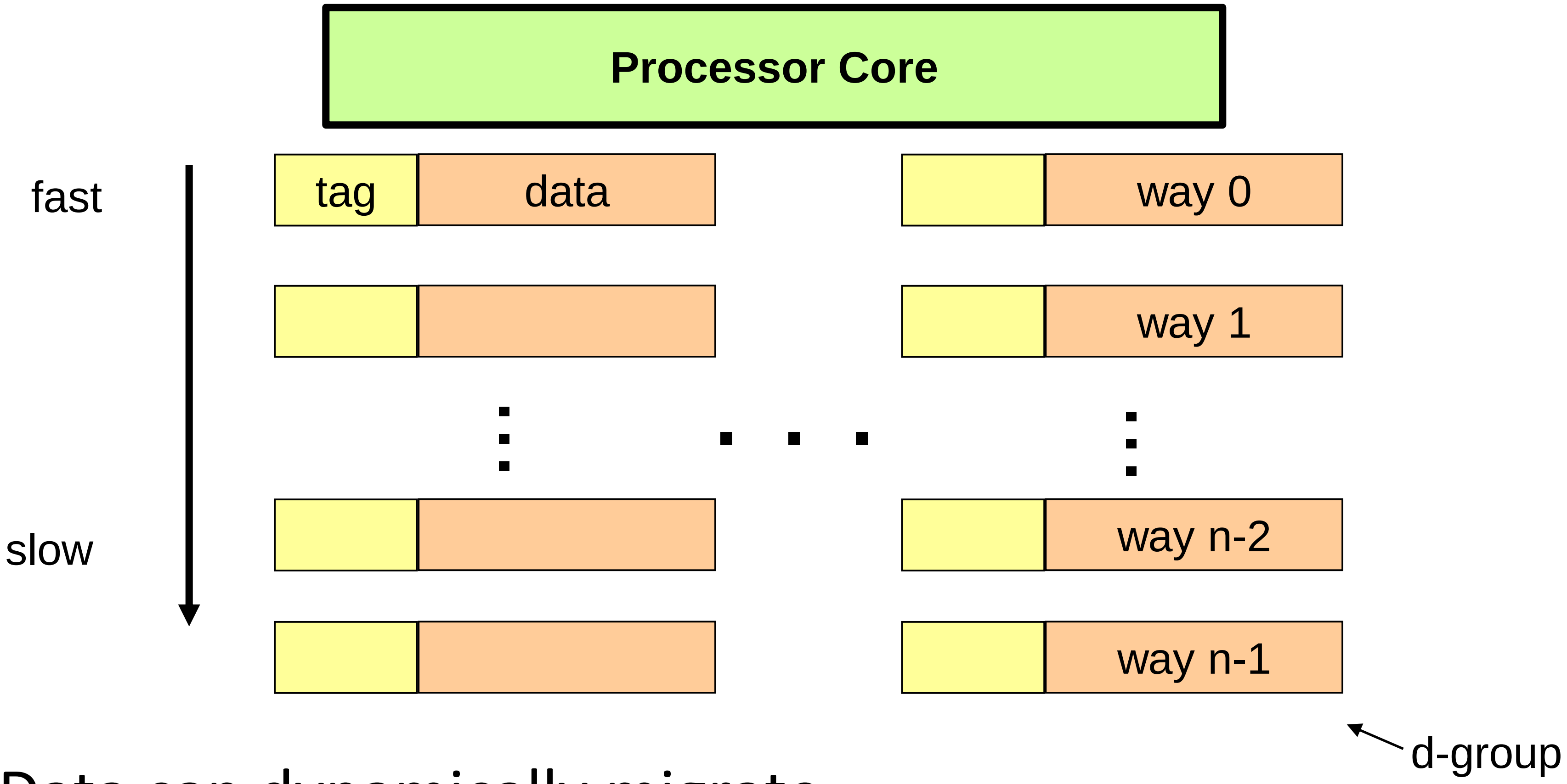
S-NUCA2 vs. S-NUCA-1 IPC Performance



Aggressive scheduling for S-NUCA1
Channel fully pipelined

- S2 better than S1

Dynamic NUCA



- Data can dynamically migrate
- Move frequently used cache lines closer to CPU

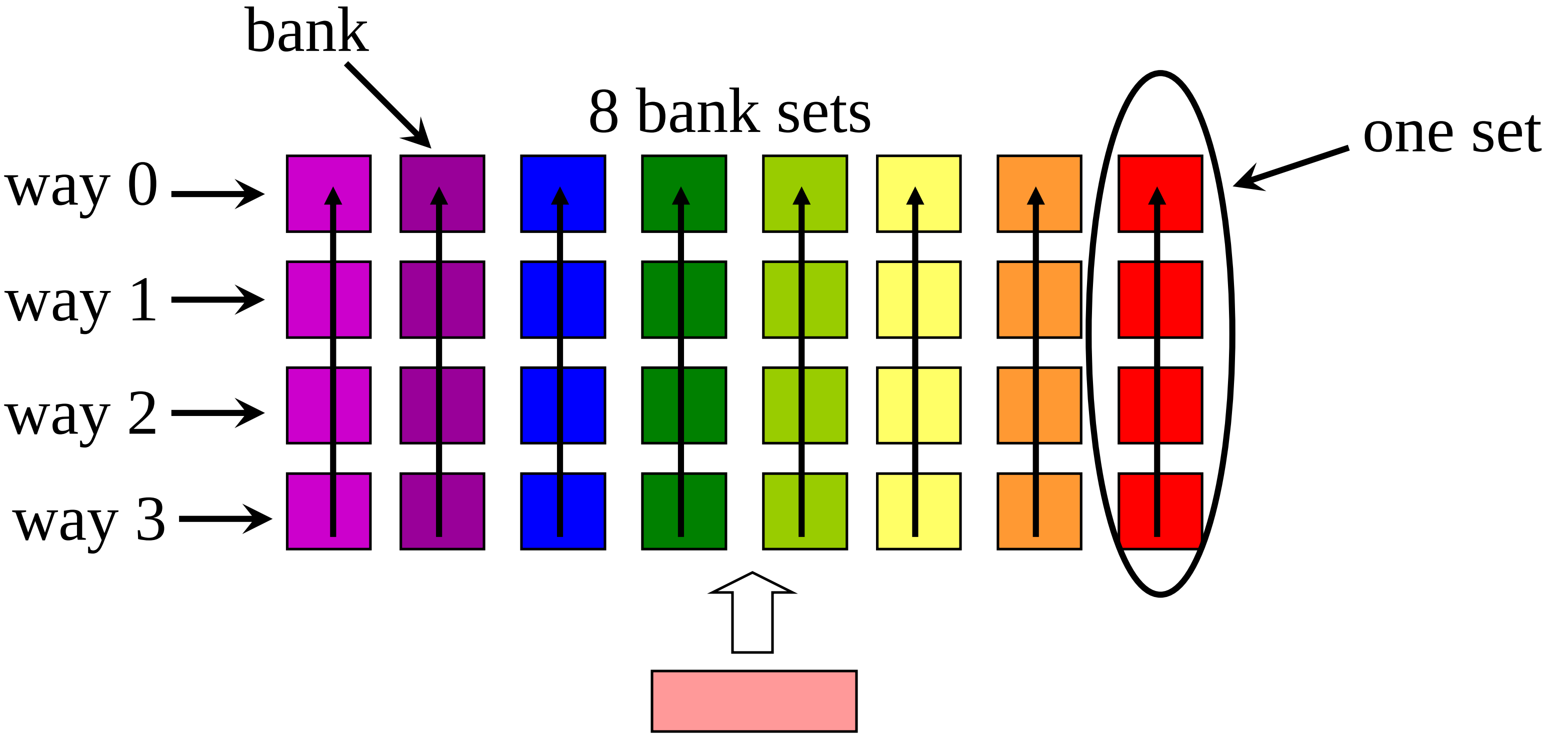
One way of each set in fast d-group; compete within set

Cache blocks “screened” for fast placement

Part of slide from Zeshan Chishti, Michael D Powell, and T. N. Vijaykumar

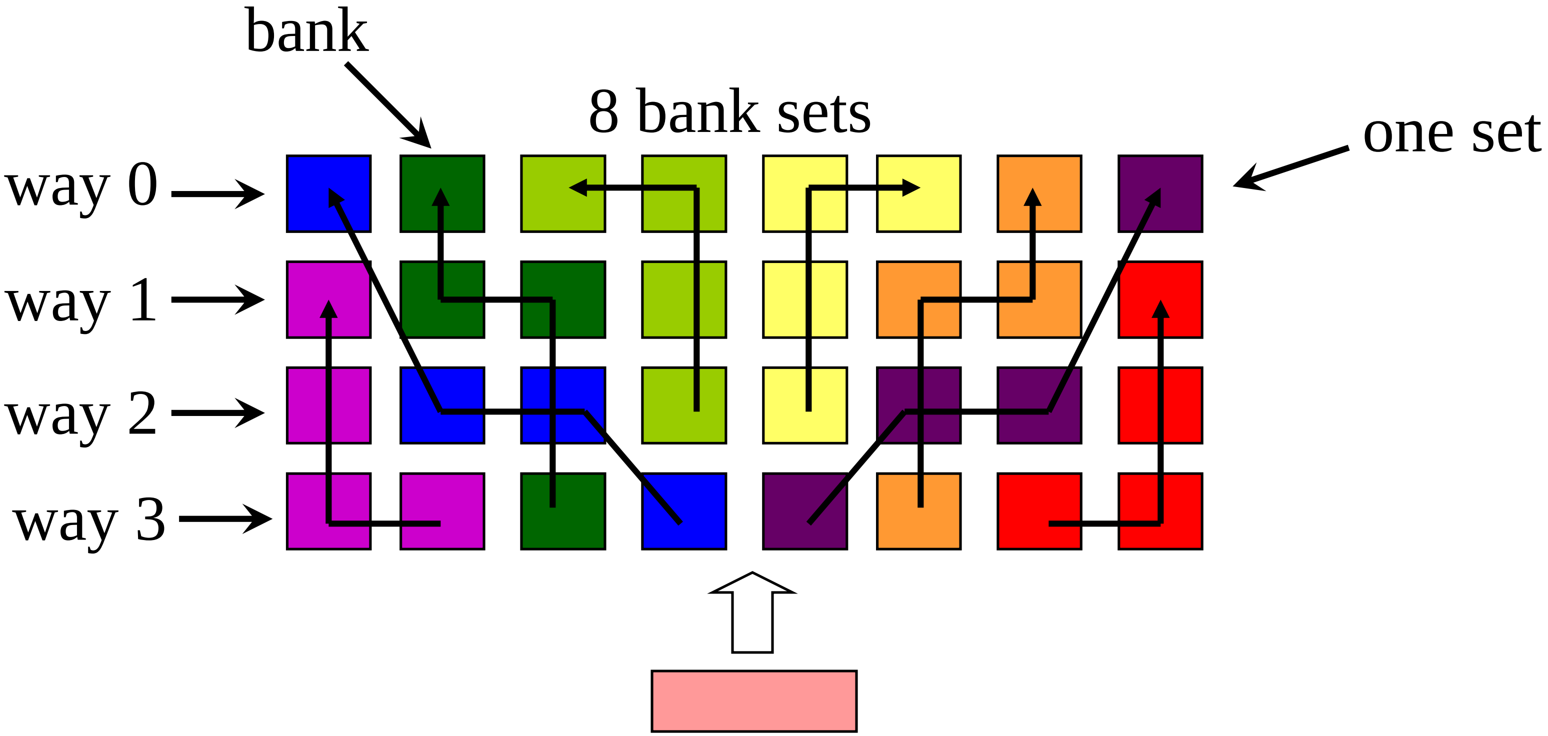
Dynamic NUCA – Mapping #1

- **Where can a block map to?**



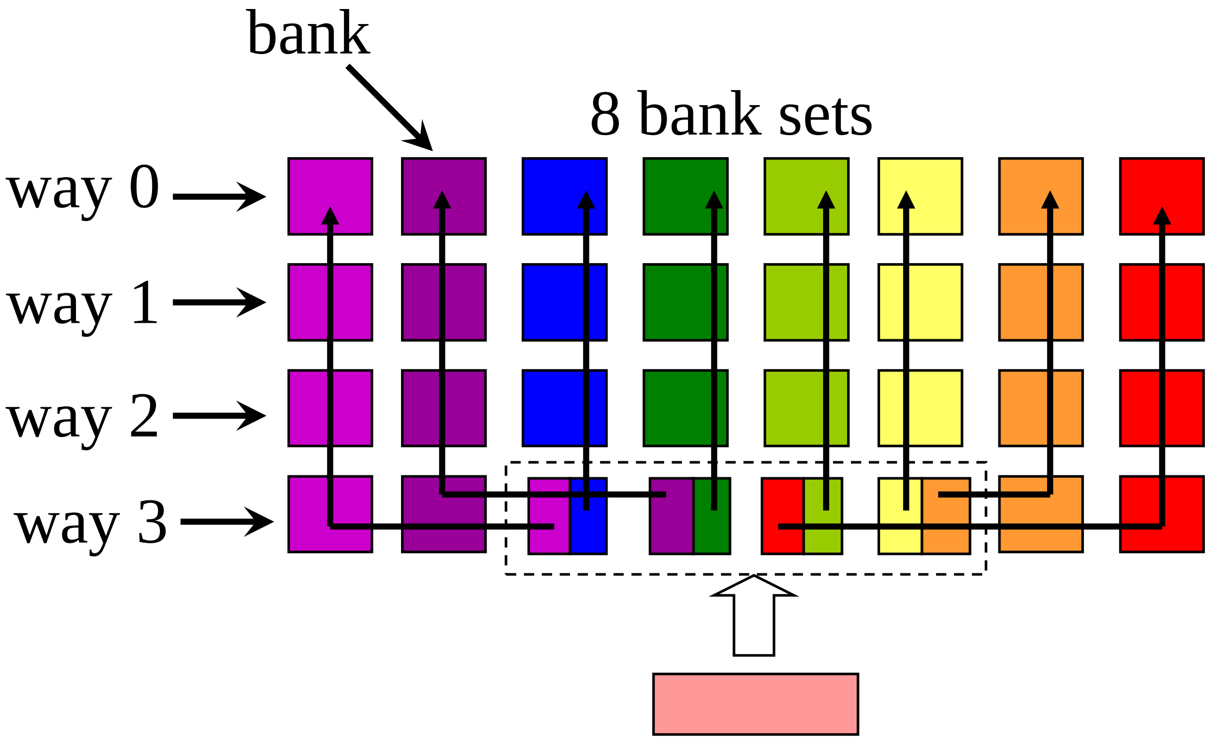
- **Simple Mapping**
- All 4 ways of each bank set need to be searched
- Farther bank sets → longer access

Dynamic NUCA – Mapping #2



- **Fair Mapping**
- Average access times across all bank sets are equal

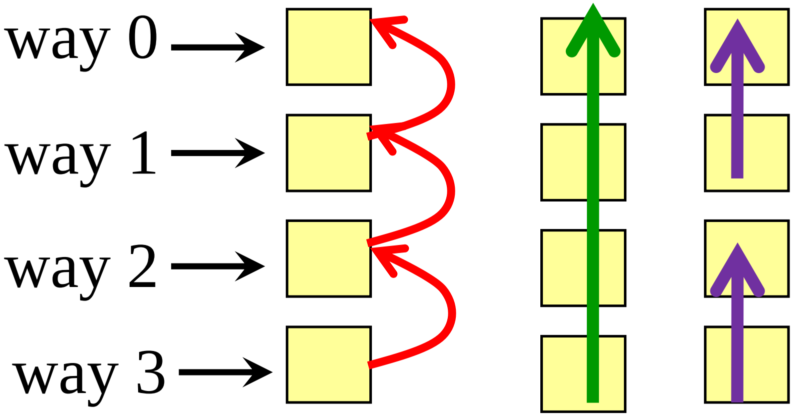
Dynamic NUCA – Mapping #3



- **Shared Mapping**
- Sharing the closest banks → every set has some fast storage
- If n bank sets share a bank then all banks must be n-way set associative

Dynamic NUCA - Searching

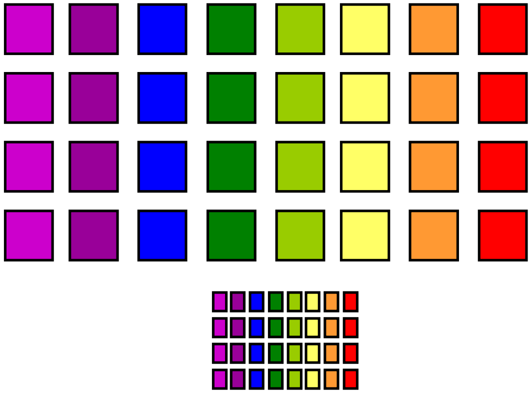
- Where is a block?



- Incremental Search
  - Search in order
- Multicast
  - Search all of them in parallel
- Partitioned Multicast
  - Search groups of them in parallel

D-NUCA – Smart Search

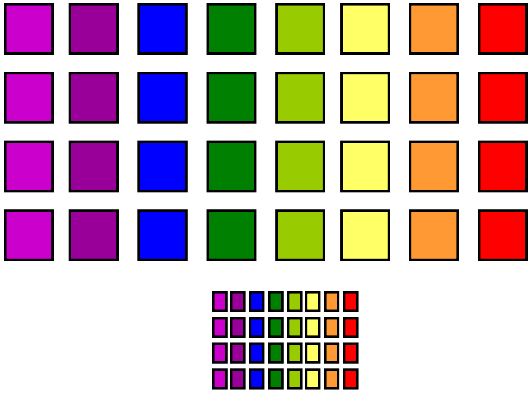
- Tags are distributed
  - May search many banks before finding a block
  - Farthest bank determines miss determination latency
- Solution: Centralized Partial Tags
  - Keep a few bits of all tags (e.g., 6) at the cache controller
  - If no match → Bank doesn't have the block
  - If match → Must access the bank to find out



Partial Tags → R.E. Kessler, R. Jooss, A. Lebeck, and M.D. Hill. Inexpensive implementations of set-associativity. In *Proceedings of the 16th Annual International Symposium on Computer Architecture,* pages 131–139, May 1989.

Partial Tags / Smart Search Policies

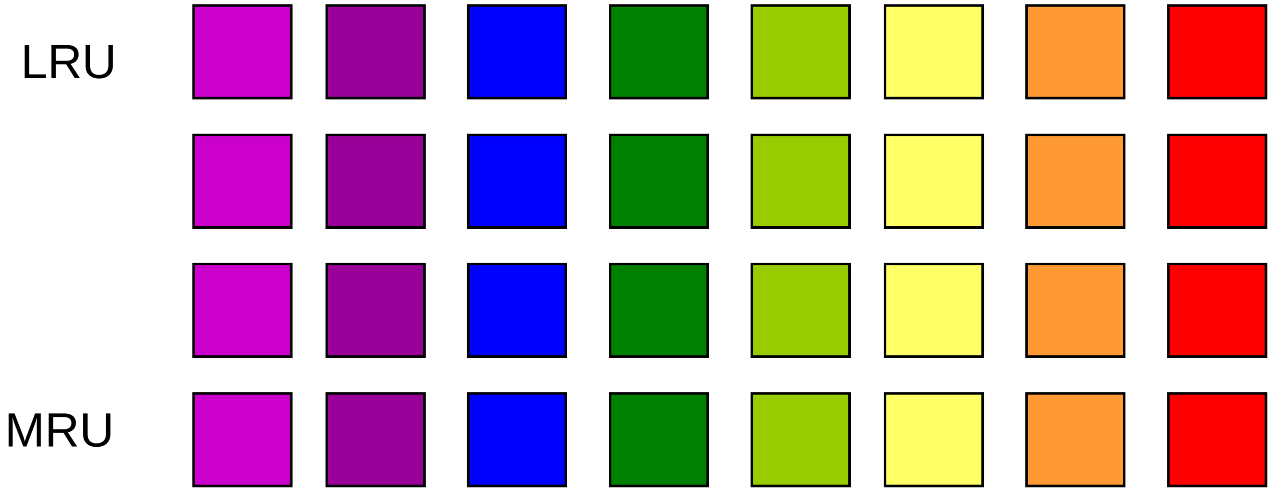
- SS-Performance:
  - Partial Tags and Banks accessed in parallel
  - Early Miss Determination
  - Go to main memory if no match
  - Reduces latency for misses



- SS-Energy:
  - Partial Tags first
  - Banks only on potential match
  - Saves energy
  - Increases Delay

Migration

- Want data that will be accessed to be close
- Use LRU?
  - Bad idea: must shift all others



- Generational Promotion
  - Move to next bank
  - Swap with another block

Initial Placement

- Where to place a new block coming from memory?
- Closest Bank?
  - May force another important block to move away
- Farthest Bank?
  - Takes several accesses before block comes close

Victim Handling

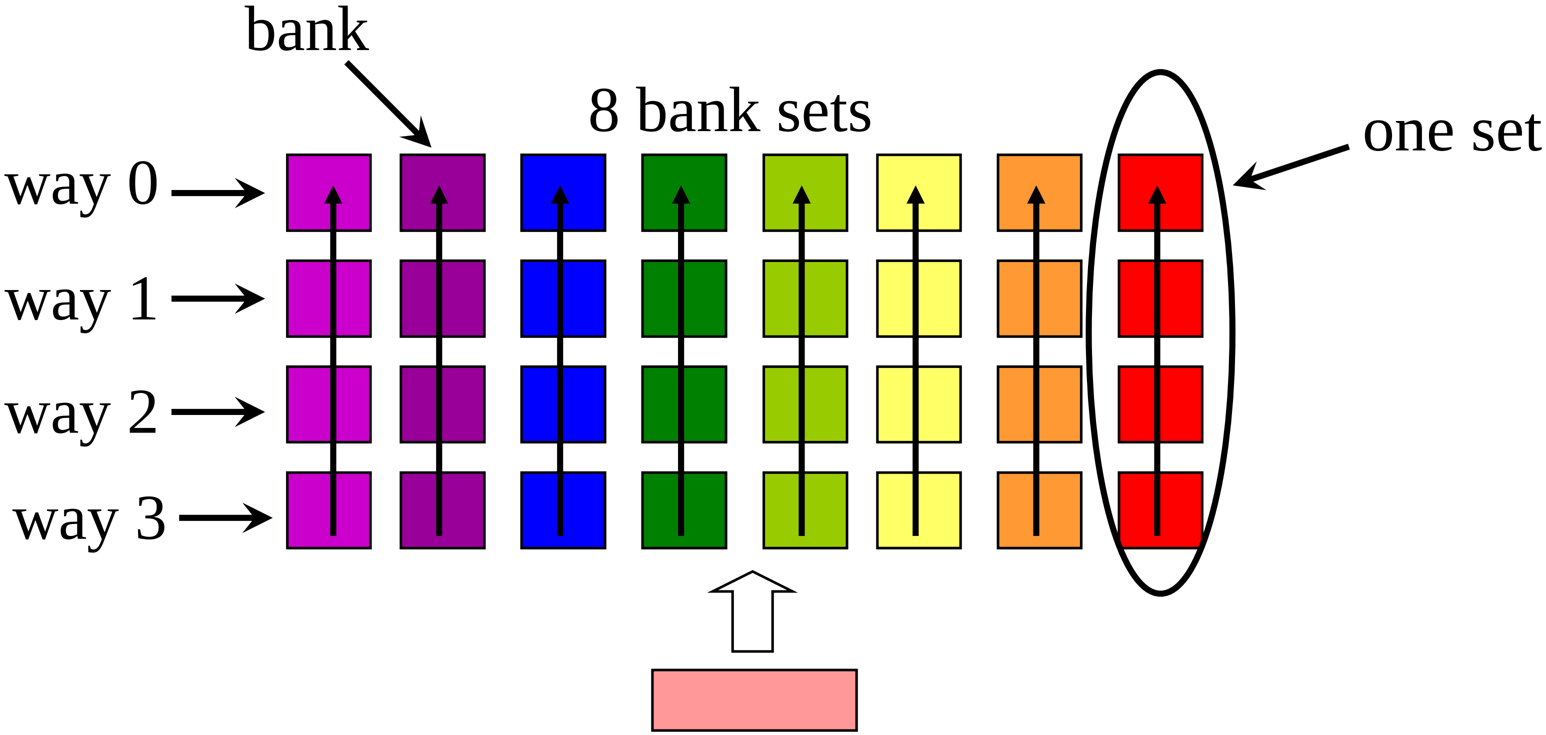
- A new block must replace an older block **victim**
- What happens to the victim?
- **Zero Copy**
  - Get's dropped completely
- **One Copy**
  - Moved away to a slower bank (next bank)

DN-Best

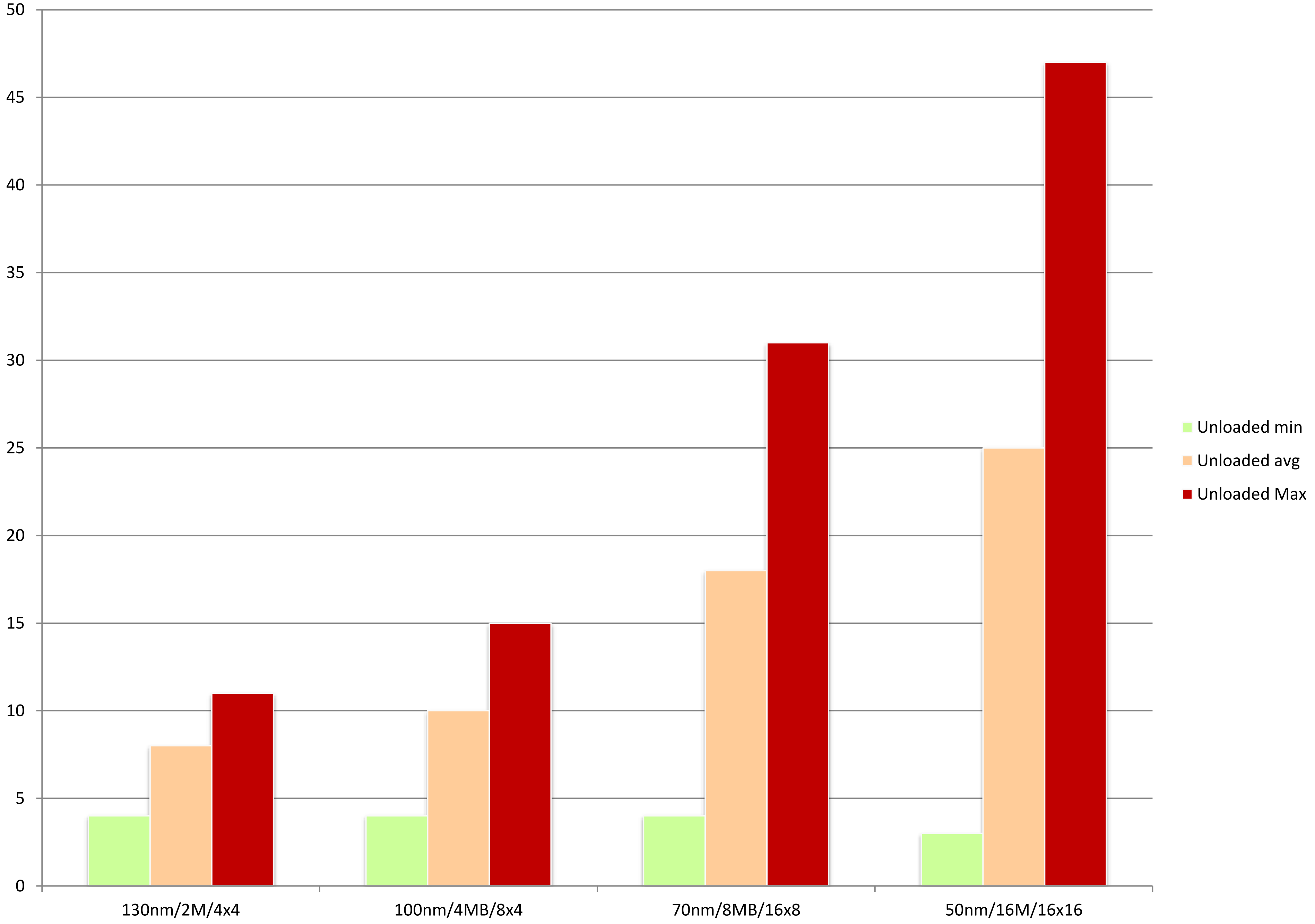
- DN-BEST
  - Shared Mapping
  - SS Energy
    - Balance performance and access account/energy
    - Maximum performance is 3% higher
  - Insert at tail
    - Insert at head → reduces avg. latency but increases misses
  - Promote on hit
    - No major differences with other polices

Baseline D-NUCA

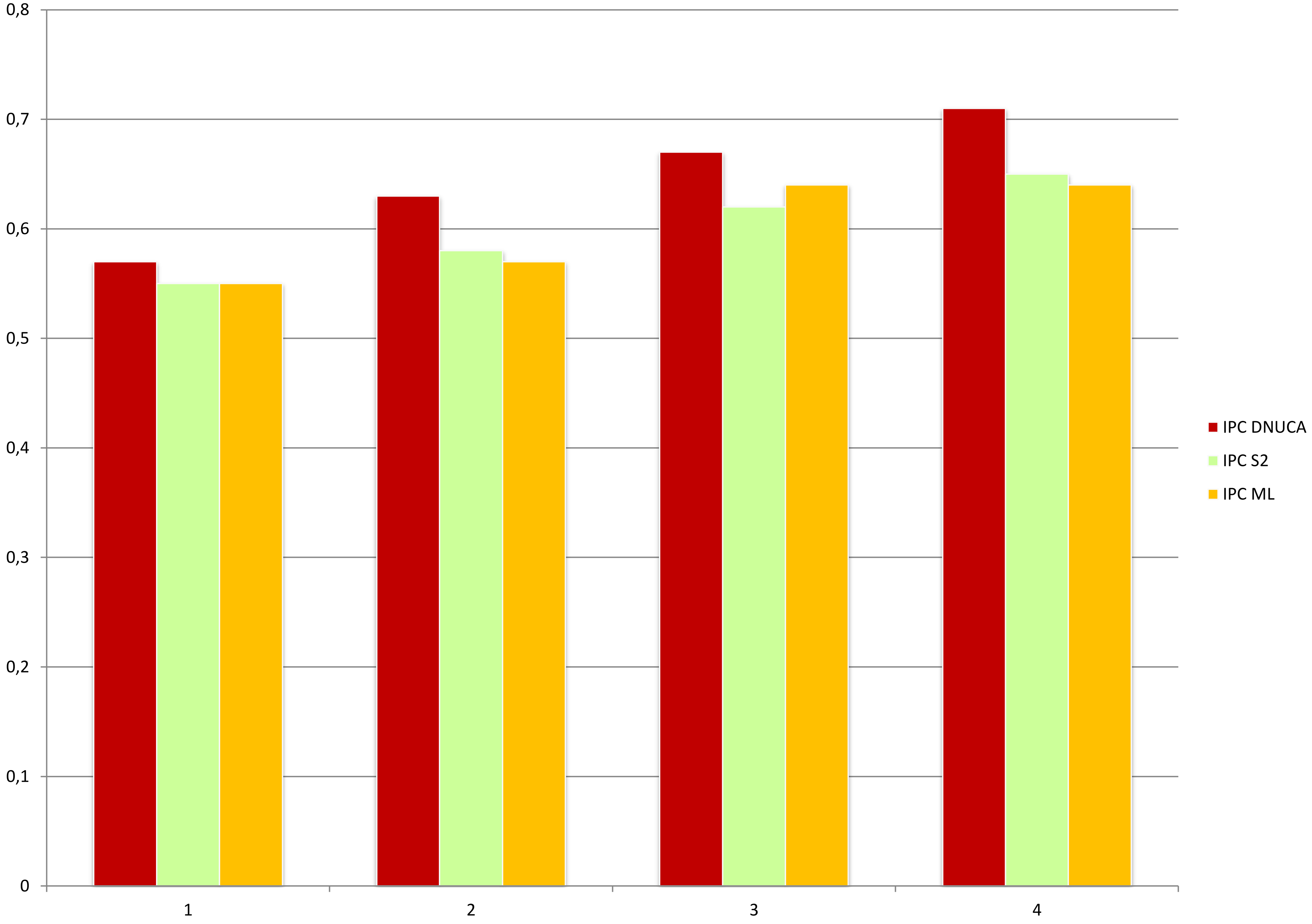
- Simple Mapping
- Multicast Search
- One-Bank Promotion on Hit
- Replace from the slowest bank



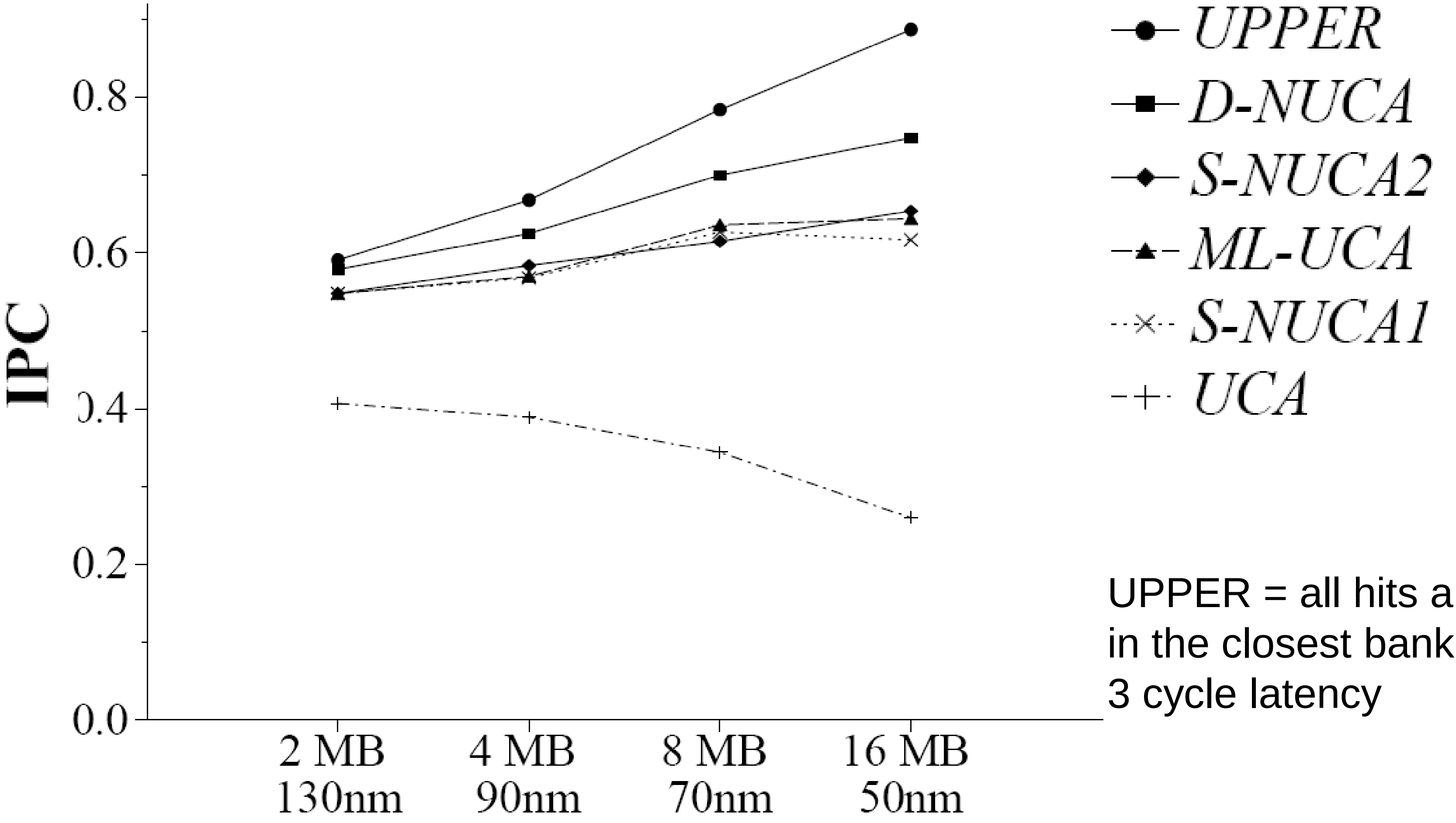
D-NUCA Unloaded Latency



IPC Performance: DNUCA vs. S-NUCA2 vs. ML-UCA



Performance Comparison



UPPER = all hits are in the closest bank
3 cycle latency

- D-NUCA and S-NUCA2 scale well
- D-NUCA outperforms all other designs
- ML-UCA saturates – UCA Degrades

Distance Associativity for High-Performance Non-Uniform Cache Architectures

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

36th Annual International Symposium on Microarchitecture (MICRO), December 2003.

Slides mostly directly from the authors' conference presentation

Motivation

Large Cache Design

- L2/L3 growing (e.g., 3 MB in Itanium II)
- Wire-delay becoming dominant in access time

Conventional large-cache

- Many subarrays => wide range of access times
- Uniform cache access => access-time of *slowest* subarray
- Oblivious to access-frequency of data

Want often-accessed data faster: improve access time

Previous work: NUCA (ASPLOS ’02)

Pioneered Non-Uniform Cache Architecture

Access time: Divides cache into many distance-groups

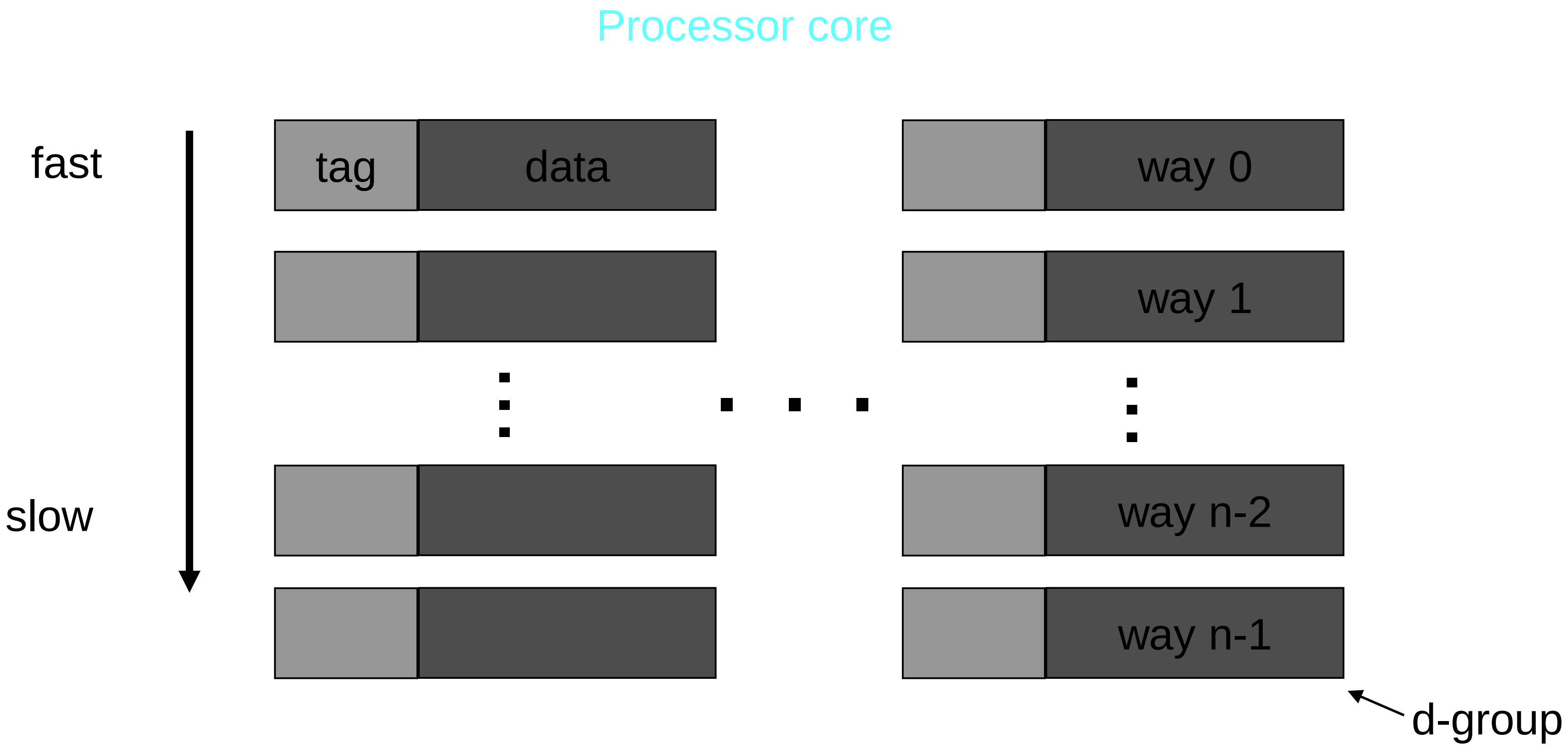
- D-group closer to core => faster access time

Data Mapping: conventional

- Set determined by block index; each set has n-ways

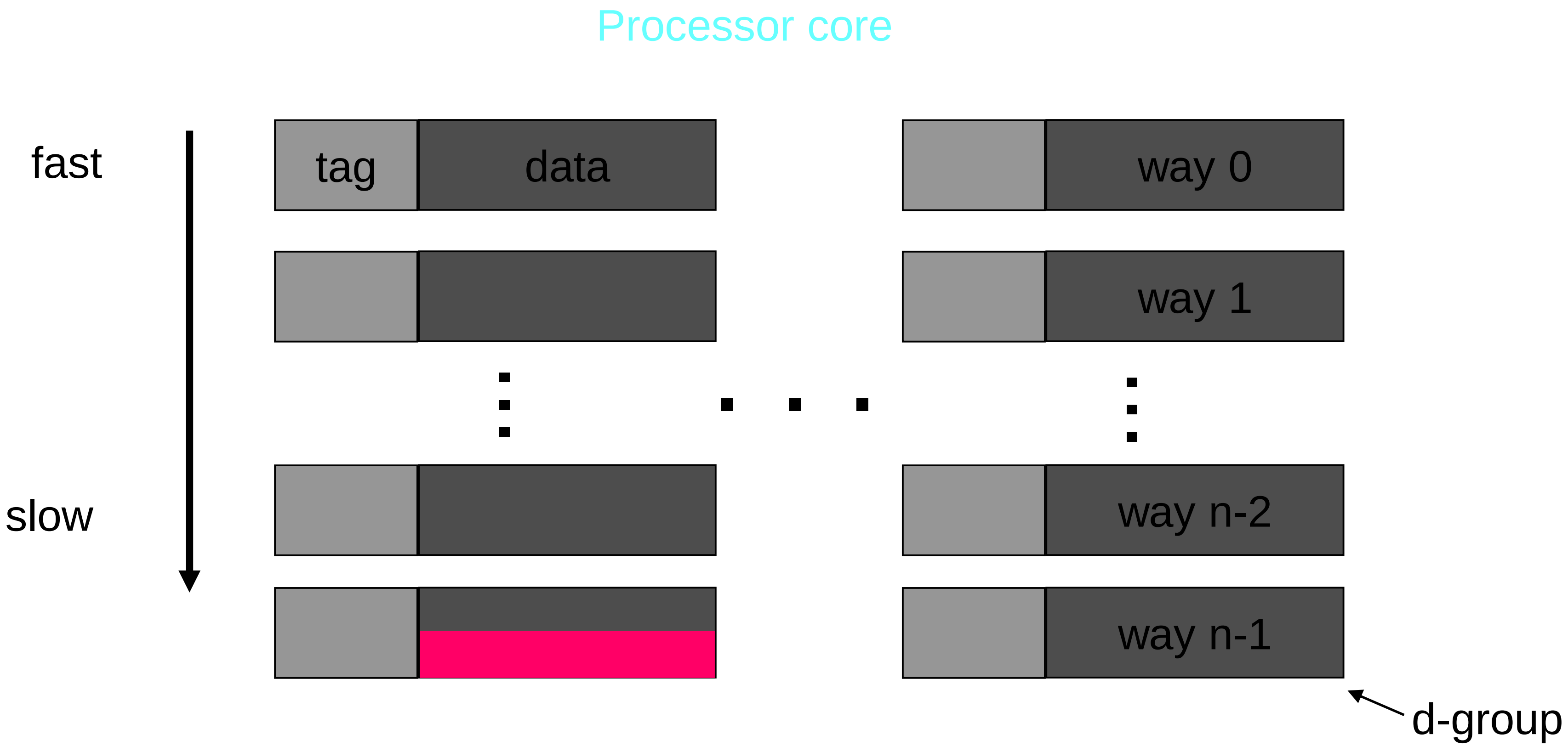
Within a set, place frequently-accessed data in fast d-group

- Place blocks in farthest way; bubble closer if needed



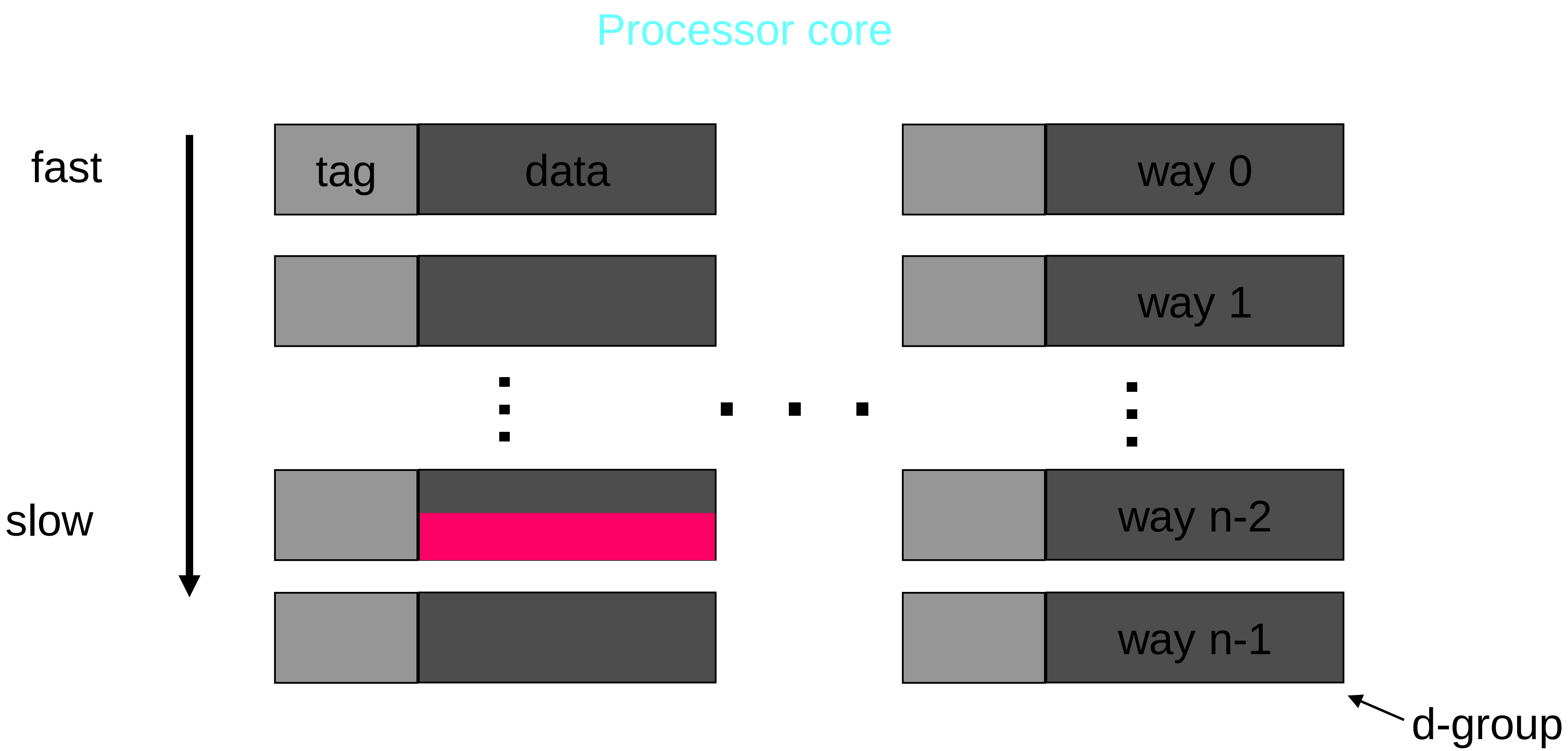
One way of each set in fast d-group; compete within set

Cache blocks “screened” for fast placement



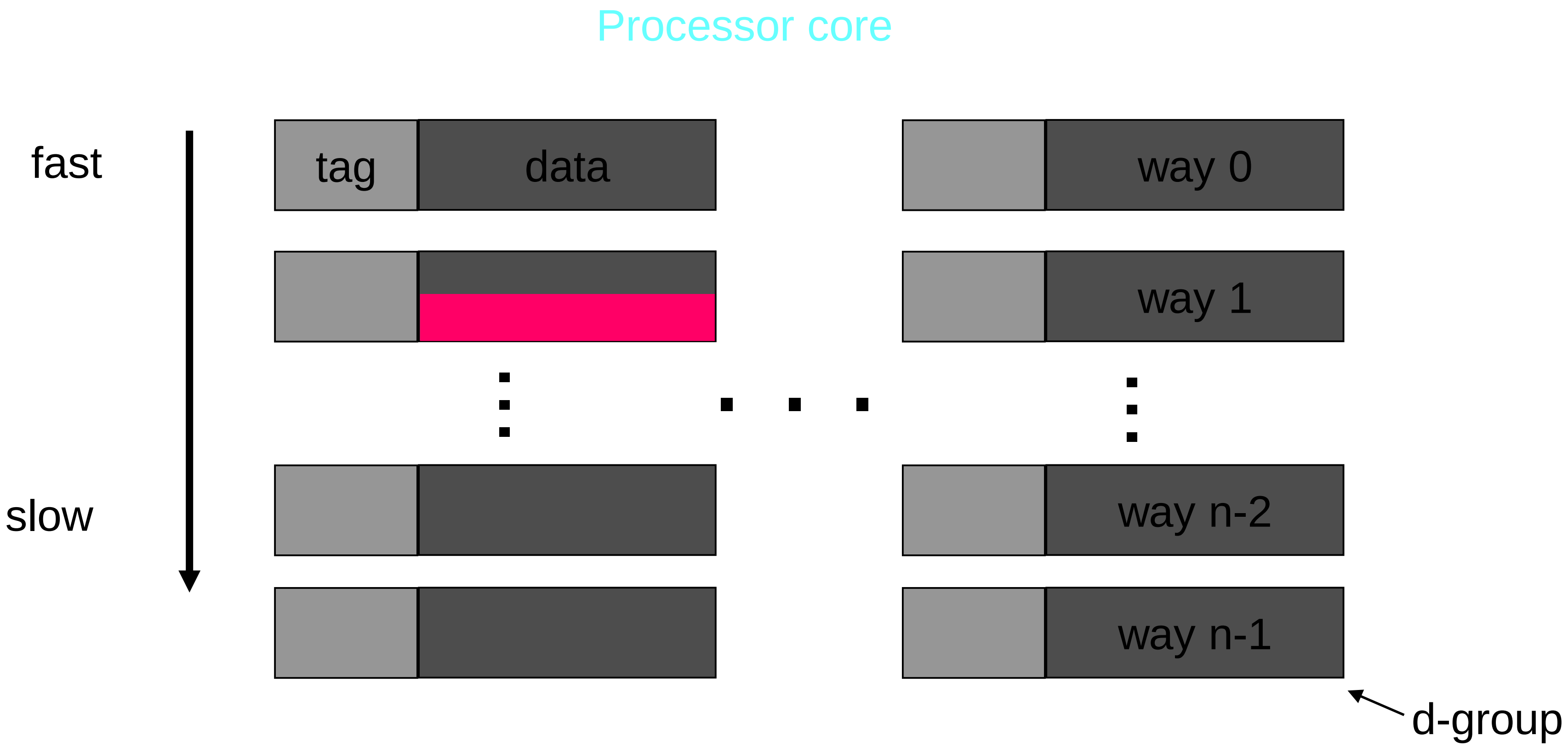
One way of each set in fast d-group; compete within set

Cache blocks “screened” for fast placement



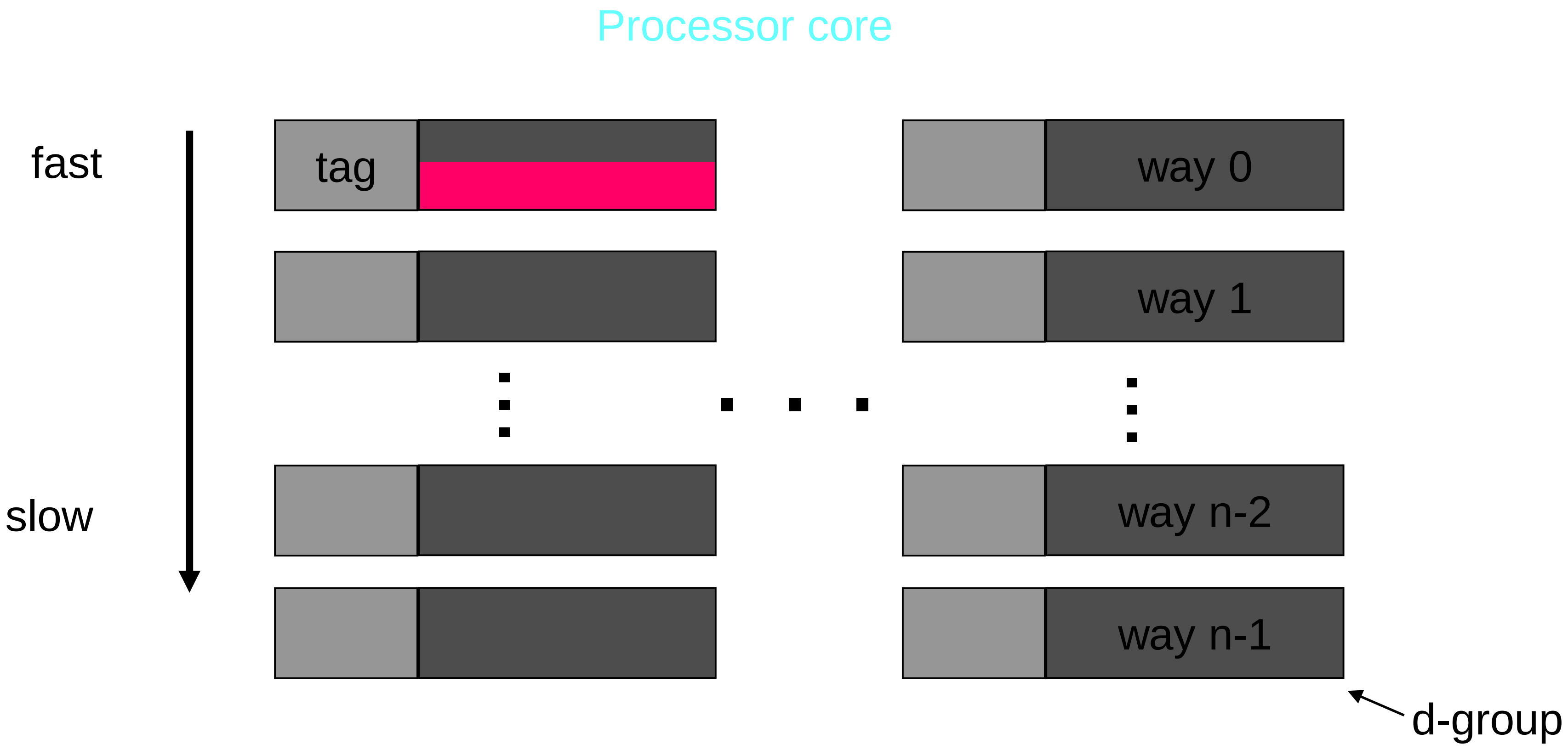
One way of each set in fast d-group; compete within set

Cache blocks “screened” for fast placement



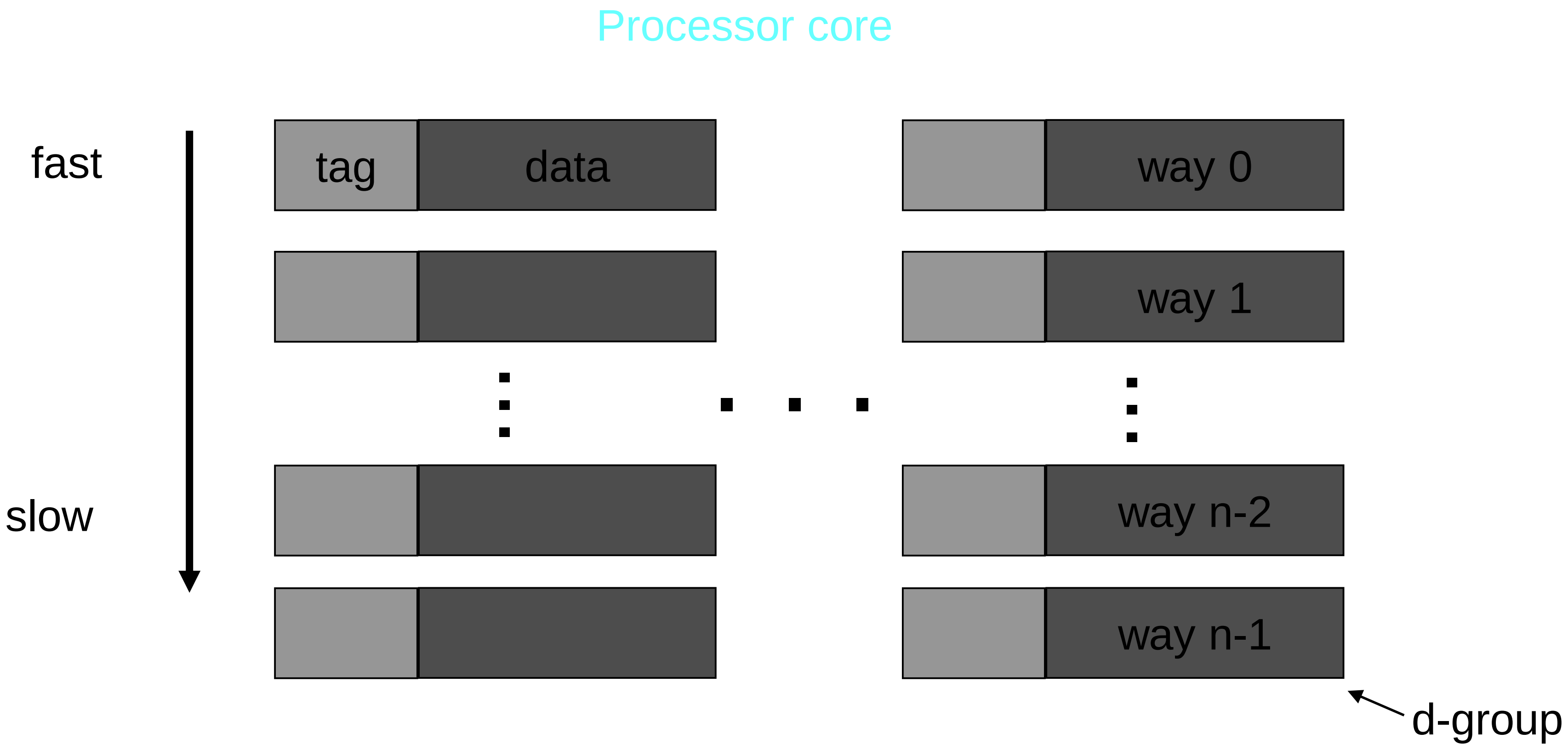
One way of each set in fast d-group; compete within set

Cache blocks “screened” for fast placement



One way of each set in fast d-group; compete within set

Cache blocks “screened” for fast placement



Want to change restriction; more flexible data-placement

NUCA

Artificial coupling between s-a way # and d-group

- Only one way in each set can be in fastest d-group
  - Hot sets have > 1 frequently-accessed way
  - Hot sets can place only one way in fastest d-group

Swapping of blocks is bandwidth- and energy-hungry

- D-NUCA uses a switched network for fast swaps

Common Large-cache Techniques

Sequential Tag-Data: e.g., Alpha 21164 L2, Itanium II L3

- Access tag first, and then access only matching data
- Saves energy compared to parallel access

Data Layout: Itanium II L3

- Spread a block over many subarrays (e.g., 135 in Itanium II)
- For area efficiency and hard- and soft-error tolerance

These issues are important for large caches

Contributions

Key observation:

- sequential tag-data => indirection through tag array
- Data may be located anywhere

Distance Associativity:

Decouple tag and data => flexible mapping for sets

Any # of ways of a hot set can be in fastest d-group

NuRAPID cache: Non-uniform access with Replacement And Placement usIng Distance associativity

Benefits:

More accesses to faster d-groups

Fewer swaps => less energy, less bandwidth

But:

More tags + pointers are needed

Outline

- Overview
- NuRAPID Mapping and Placement
- NuRAPID Replacement
- NuRAPID layout
- Results
- Conclusion

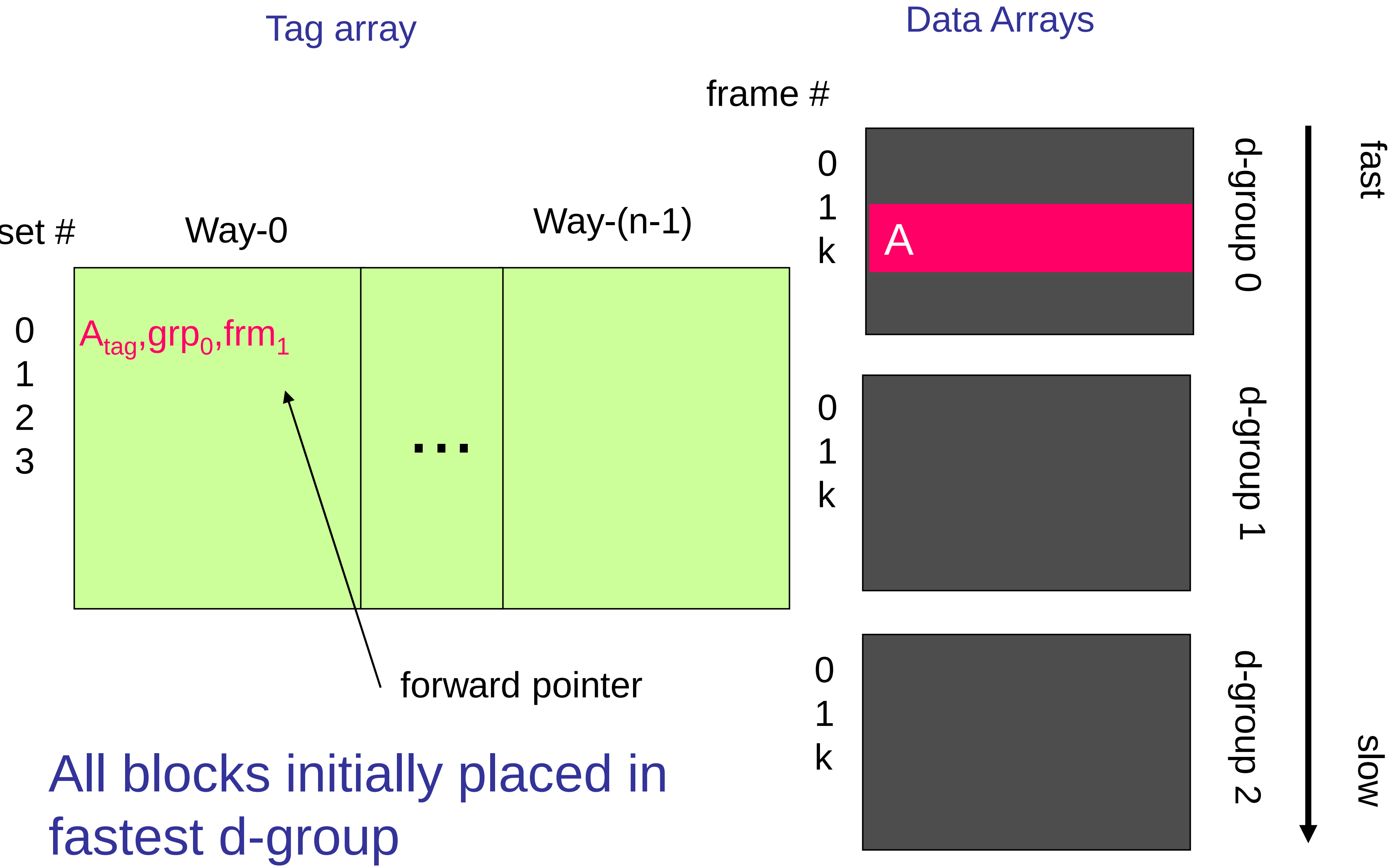
NuRAPID Mapping and Placement

Distance-Associative Mapping:

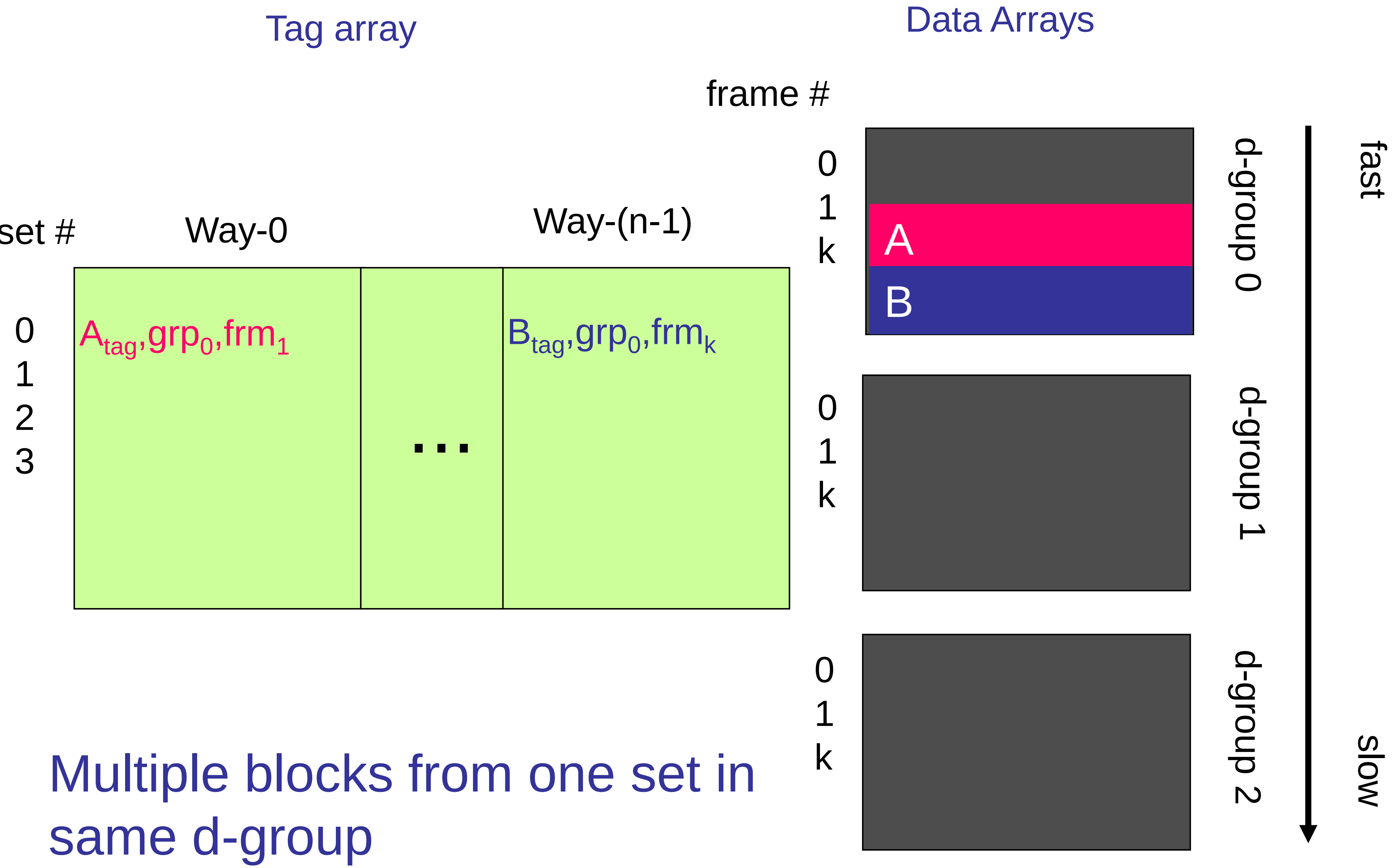
- decouple tag from data using forward pointer
- Tag access returns forward pointer, data location

Placement: data block can be placed anywhere

- Initially place all data in fastest d-group
- Small risk of displacing often-accessed block



All blocks initially placed in fastest d-group



Tag array

Data Arrays

frame #

set #

Way-0

Way-(n-1)

0
1
2
3

A_{tag},grp$_0$,frm$_1$

. . .

B_{tag},grp$_0$,frm$_k$

0
1
k

A

B

d-group 0

0
1
k

d-group 1

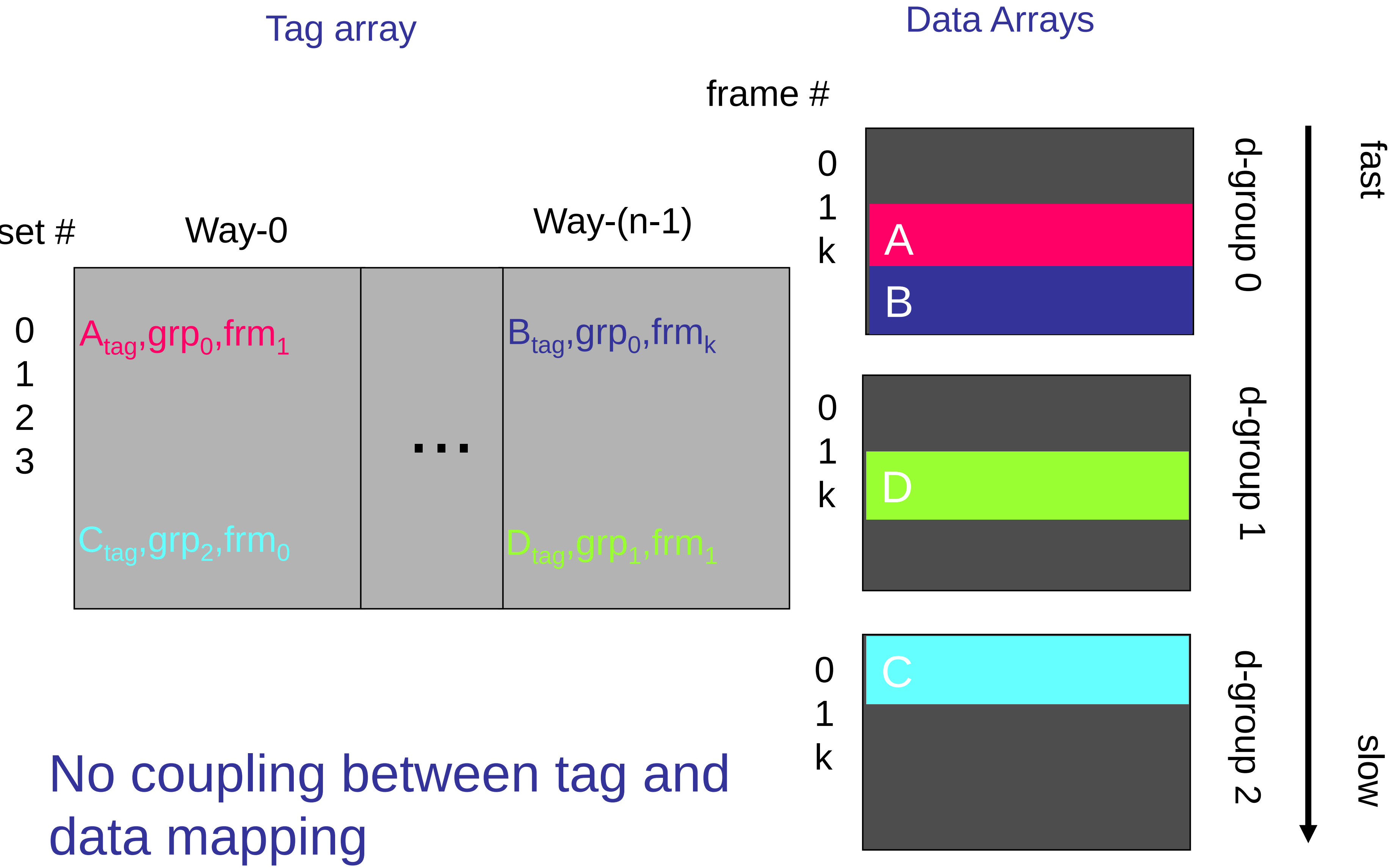
0
1
k

d-group 2

fast

slow

Multiple blocks from one set in same d-group



Tag array

Data Arrays

frame #

set #

Way-0

Way-(n-1)

0
1
2
3

A_{tag},grp_0,frm_1

B_{tag},grp_0,frm_k

C_{tag},grp_2,frm_0

D_{tag},grp_1,frm_1

. . .

0
1
k

A

B

d-group 0

0
1
k

D

d-group 1

0
1
k

C

d-group 2

fast

slow

No coupling between tag and data mapping

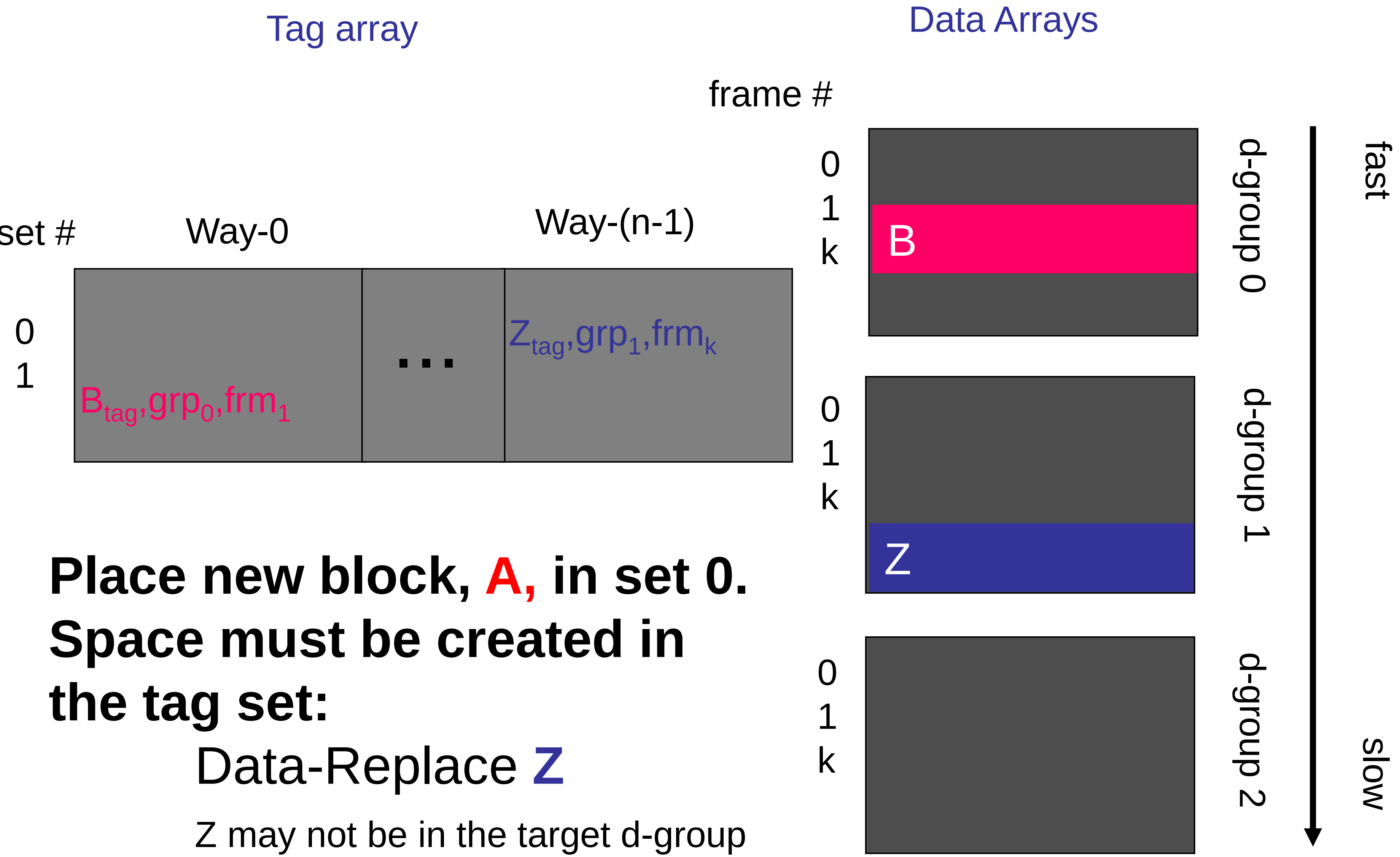
Outline

- Overview
- NuRAPID Mapping and Placement
- NuRAPID Replacement
- NuRAPID layout
- Results
- Conclusion

NuRAPID Replacement

Two forms of replacement:

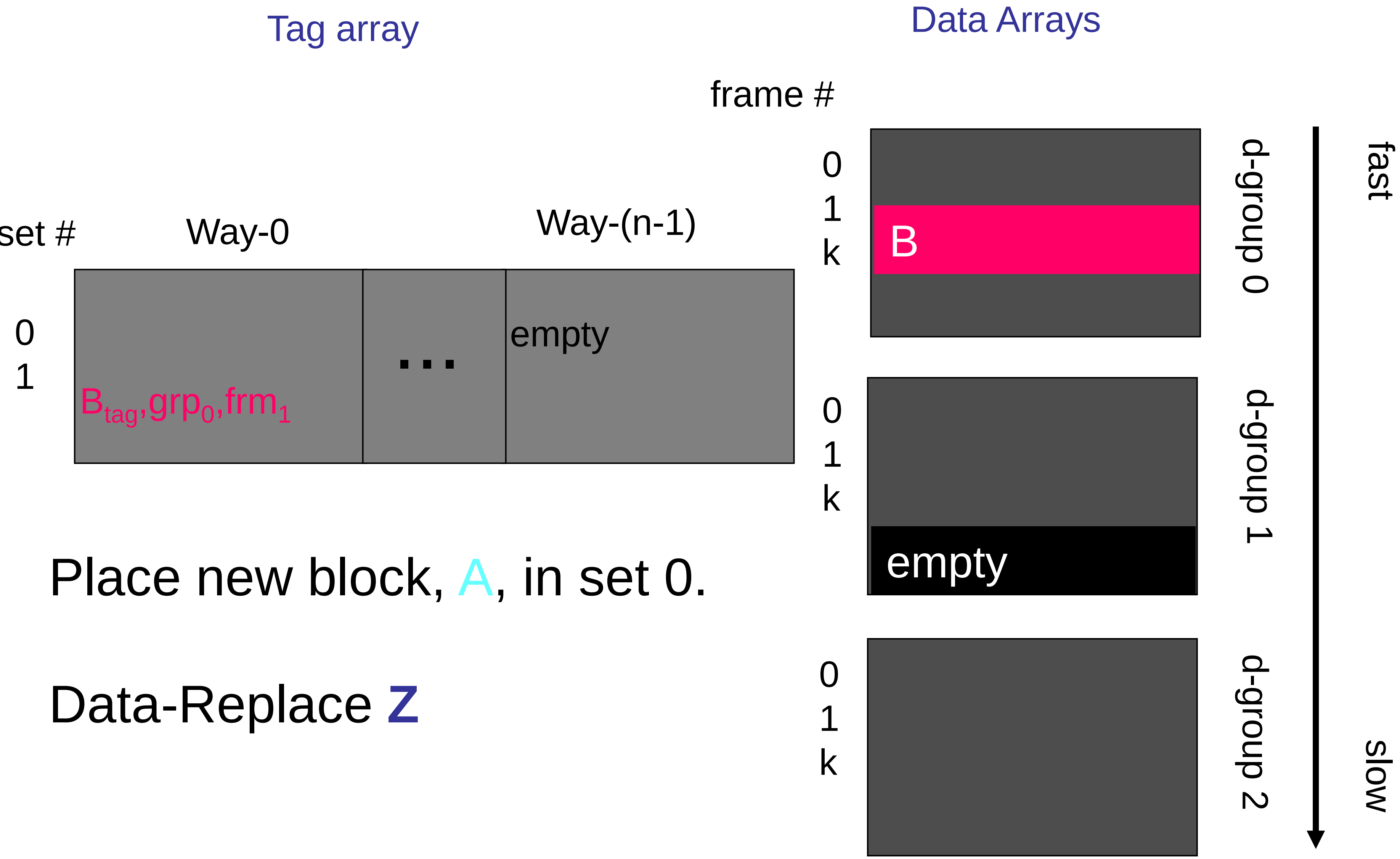
- Data Replacement: Like conventional
  - Evicts blocks from cache due to tag-array limits
- Distance Replacement: Moving blocks among d-groups
  - Determines which block to demote from a d-group
  - Decoupled from data replacement
  - No blocks evicted
    - Blocks are swapped



Place new block, A, in set 0. Space must be created in the tag set:

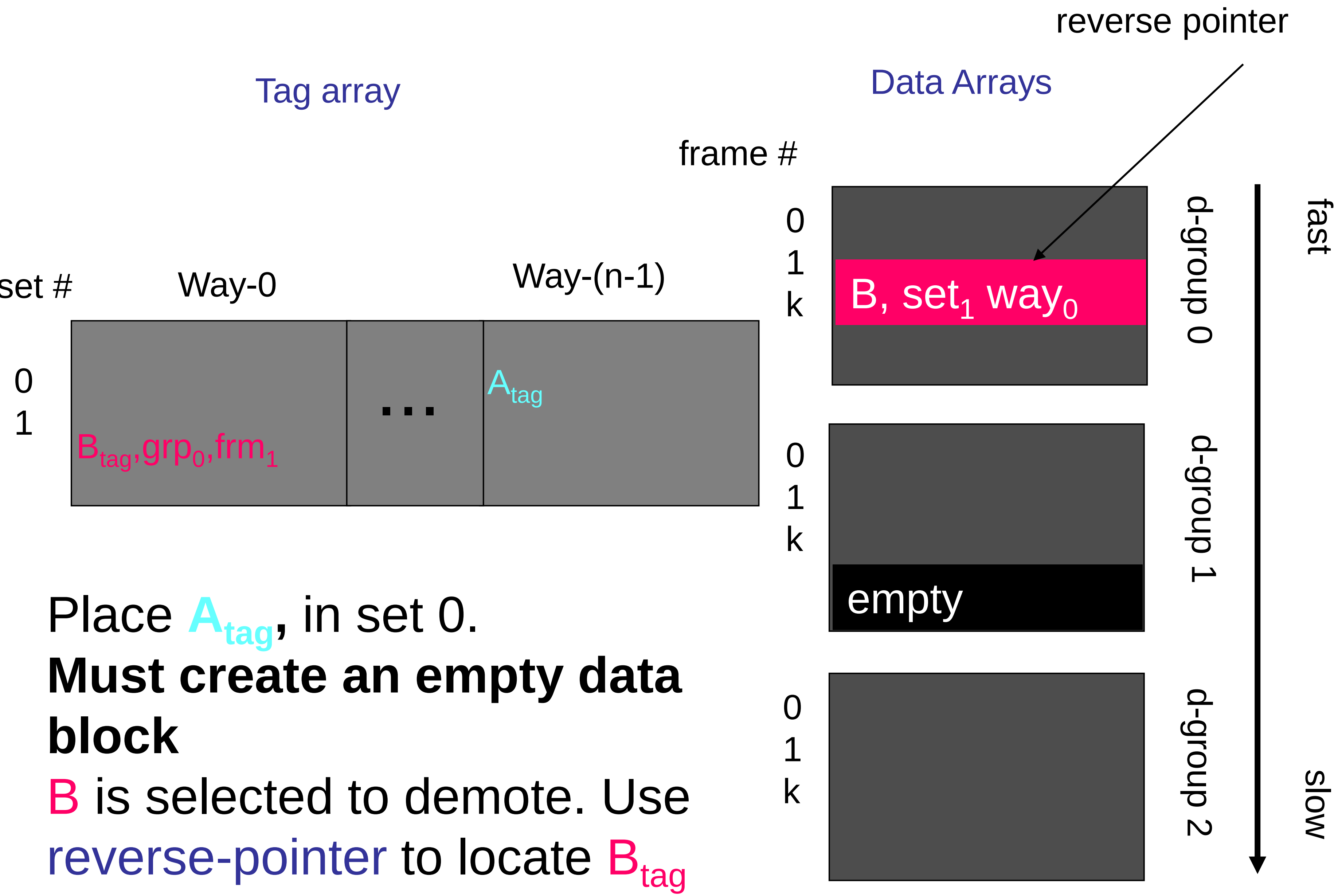
Data-Replace **Z**

Z may not be in the target d-group



Place new block, A, in set 0.

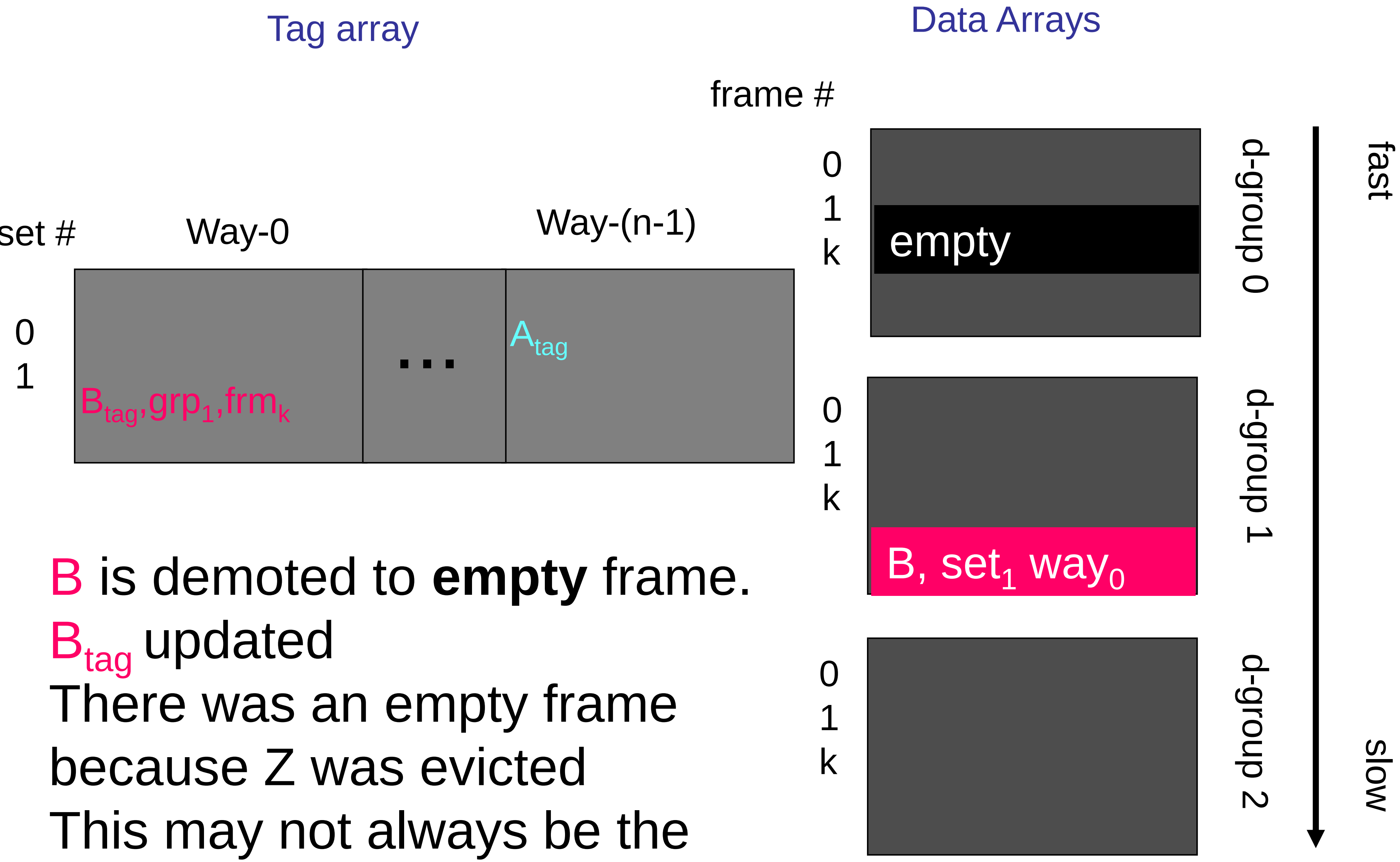
Data-Replace **Z**



Place A_{tag}, in set 0.

Must create an empty data block

B is selected to demote. Use reverse-pointer to locate B_{tag}



Tag array

Data Arrays

frame #

set #

Way-0

Way-(n-1)

0

1

B_{tag},grp$_1$,frm$_k$

. . .

A_{tag}

0
1
k

empty

d-group 0

0
1
k

B, set$_1$ way$_0$

d-group 1

0
1
k

d-group 2

fast

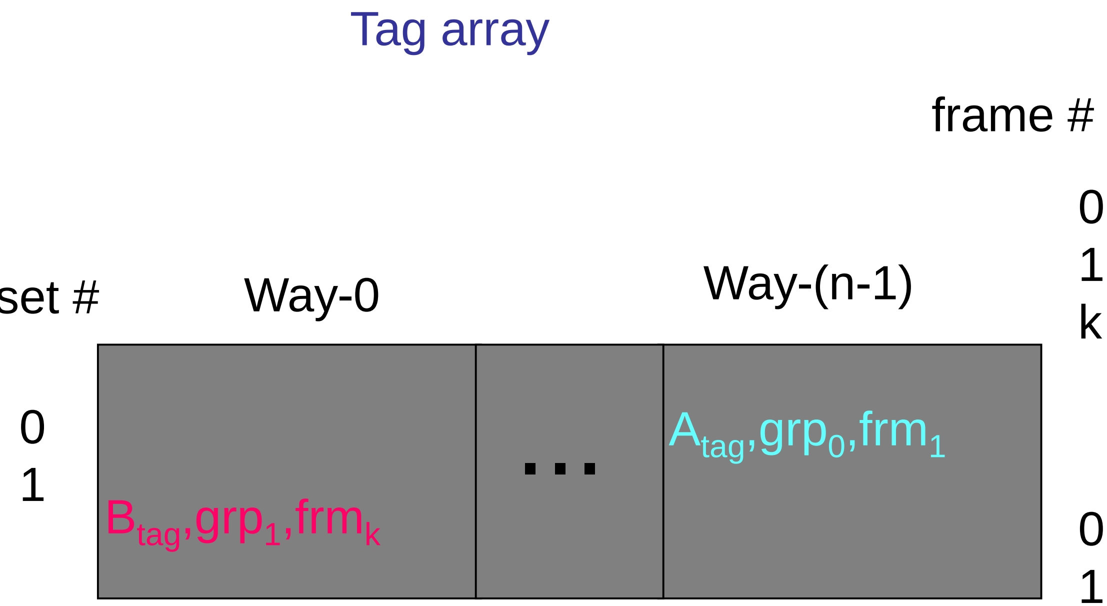
slow

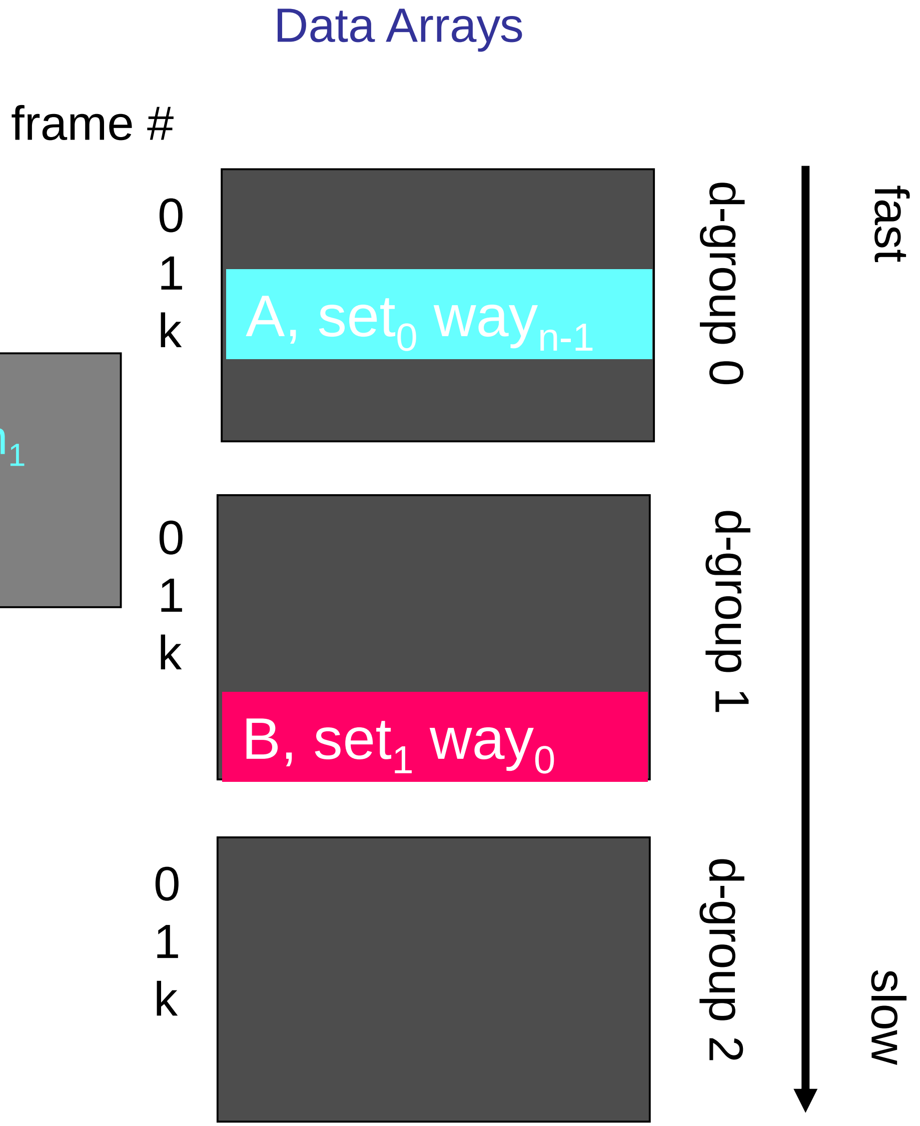
B is demoted to **empty** frame.

B_{tag} updated

There was an empty frame because Z was evicted

This may not always be the case





A is placed in d-group 0
pointers updated

Replacement details

Always empty block for demotion for dist.-replacement

- May require multiple demotions to find it

Example showed only demotion

- Block could get stuck in slow d-group
- Solution: Promote upon access (see paper)

How to choose block for demotion?

- Ideal: LRU-group
- LRU hard. We show random OK (see paper)
  - Promotions fix errors made by random

Outline

- Overview
- NuRAPID Mapping and Placement
- NuRAPID Replacement
- NuRAPID layout
- Results
- Conclusion

Layout: small vs. large d-groups

Key: Conventional caches spread block over subarrays

+ Splits the “decoding” into the address decoder and muxes at the output of the subarrays

e.g., 5-to-1 decoder + 2 2-to-1 muxes better than 10-to-1 decoder

?? 9-to-1 decoder ??

+ more flexibility to deal with defects

+ more tolerant to transient errors

- Non-uniform cache: can spread over only one d-group
  - So all bits in a block have same access time

Small d-groups (e.g., 64KB of 4 16-KB subarrays)

- Fine granularity of access times
- Blocks spread over few subarrays

Large d-groups (e.g., 2 MB of 128 16-KB subarrays)

- Coarse granularity of access times
- Blocks spread over many subarrays

Large d-groups superior for spreading data

Outline

- Overview
- NuRAPID Mapping and Placement
- NuRAPID Replacement
- NuRAPID layout
- Results
- Conclusion

Methodology

- 64 KB, 2-way L1s. 8 MSHRs on d-cache

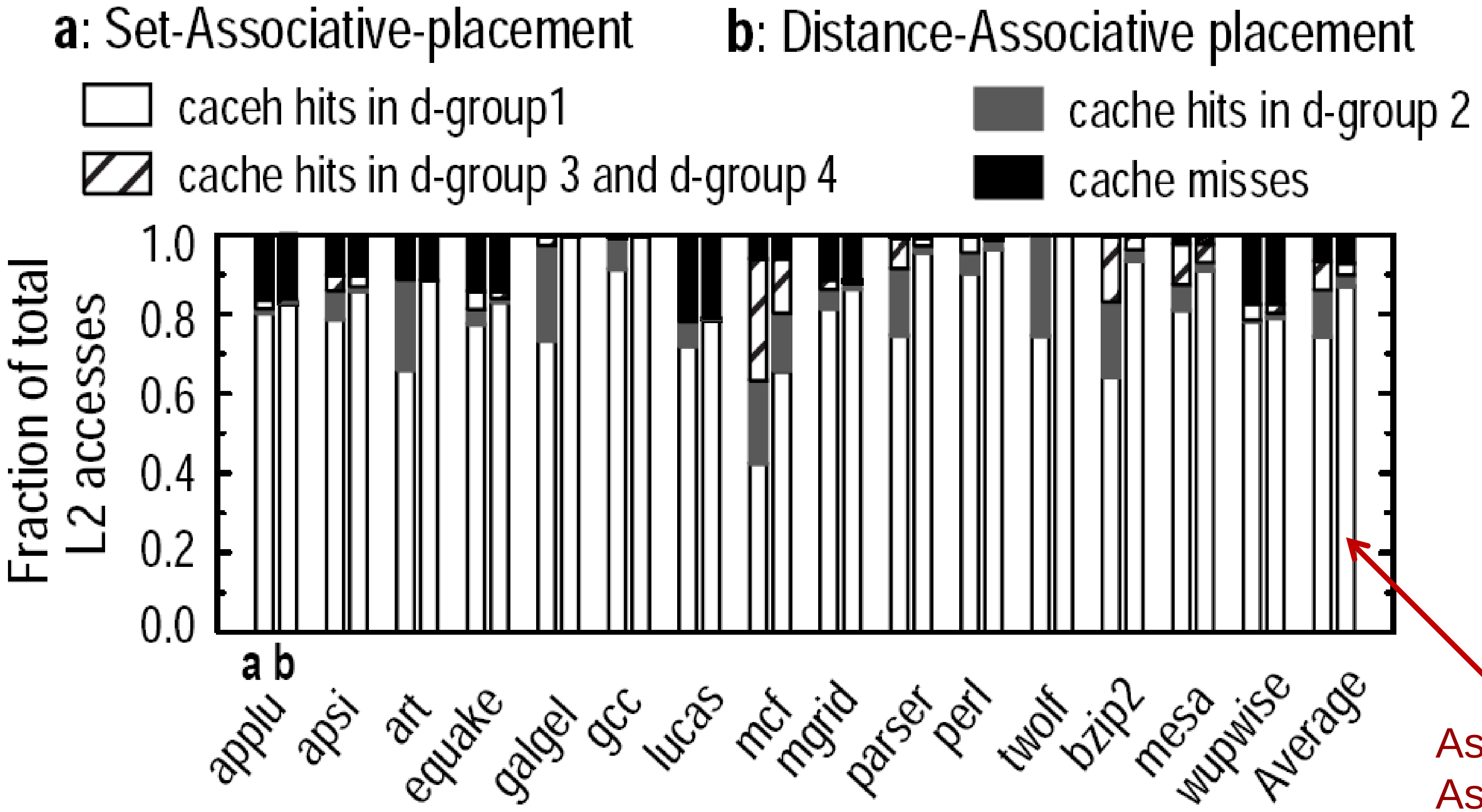
NuRAPID: 8 MB, 8-way, 1-port, no banking

- 4 d-groups (14-, 18-, 36-, 44- cycles)
- 8 d-groups (12-, 19-, 20-, . . . 49- cycles) shown in paper

Compare to:

- **BASE:**
  - 1 MB, 8-way L2 (11-cycles) + 8-MB, 8-way L3 (43-cycles)
- 8 MB, 16-way D-NUCA (4 – 31 cycles)
  - Multi-banked, infinite-bandwidth interconnect

Results



As high
As possible

FIGURE 4: Distribution of group accesses for set-associative and distance-associative placement.

Results

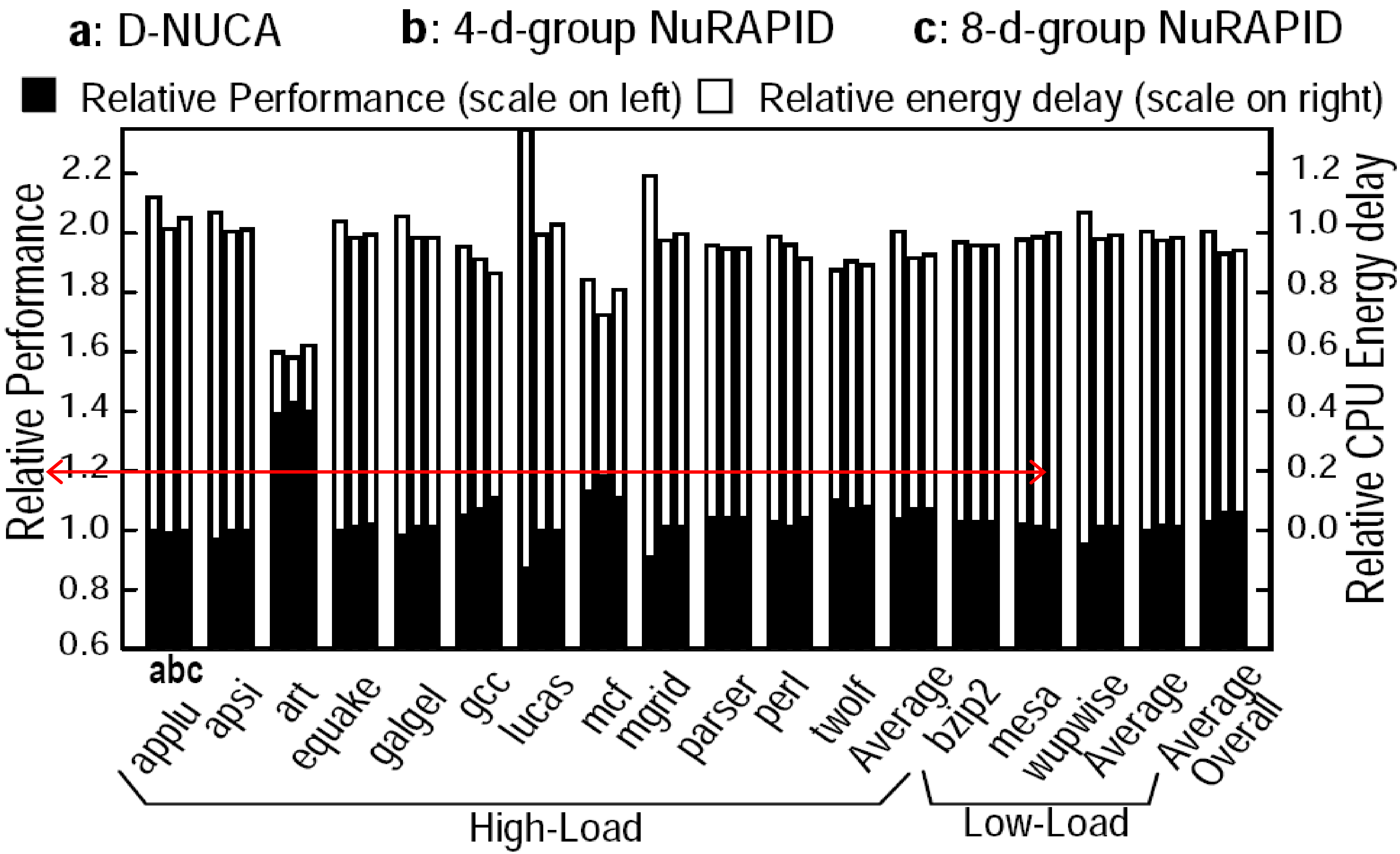


FIGURE 9: Performance and energy-delay comparison of NuRAPID and D-NUCA.

3.0% better than D-NUCA and up to 15% better

Results

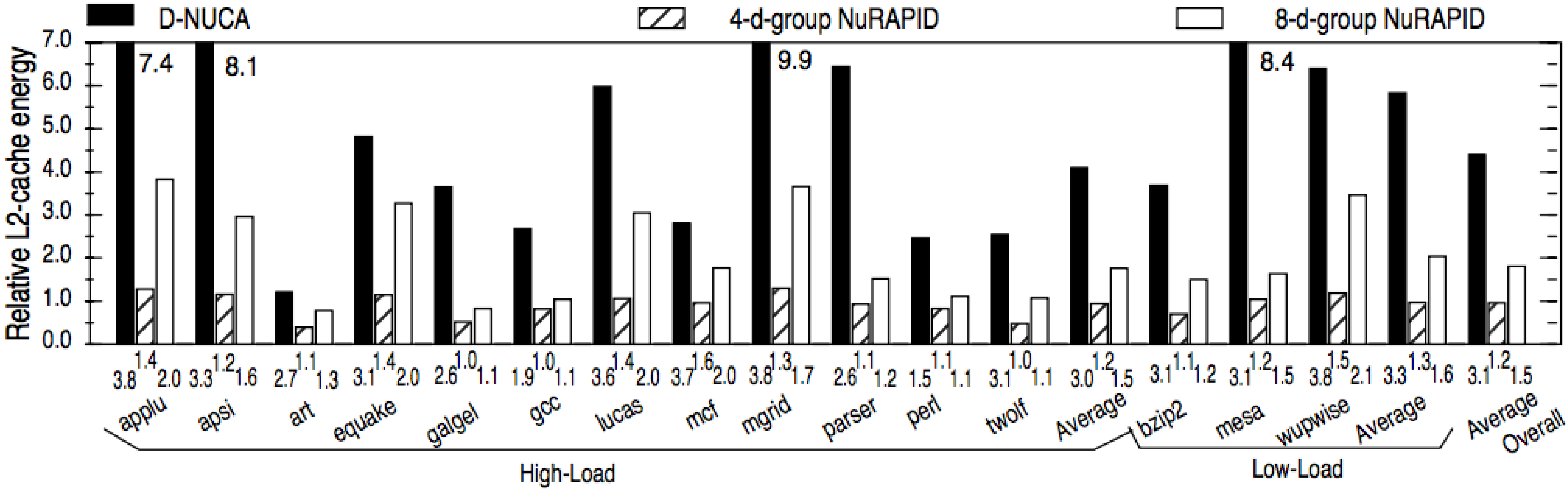


FIGURE 10: Cache energy comparison of NuRAPID and D-NUCA.

Energy effects are much more significant

Conclusions

- NuRAPID
  - leverage seq. tag-data
  - flexible placement, replacement for non-uniform cache
  - Achieves *7% overall processor E-D* savings

    over conventional cache, D-NUCA
  - Reduces L2 energy by 77% over D-NUCA

NuRAPID an important design for wire-delay dominated caches

Managing Wire Delay in Large CMP Caches

Bradford M. Beckmann and David A. Wood

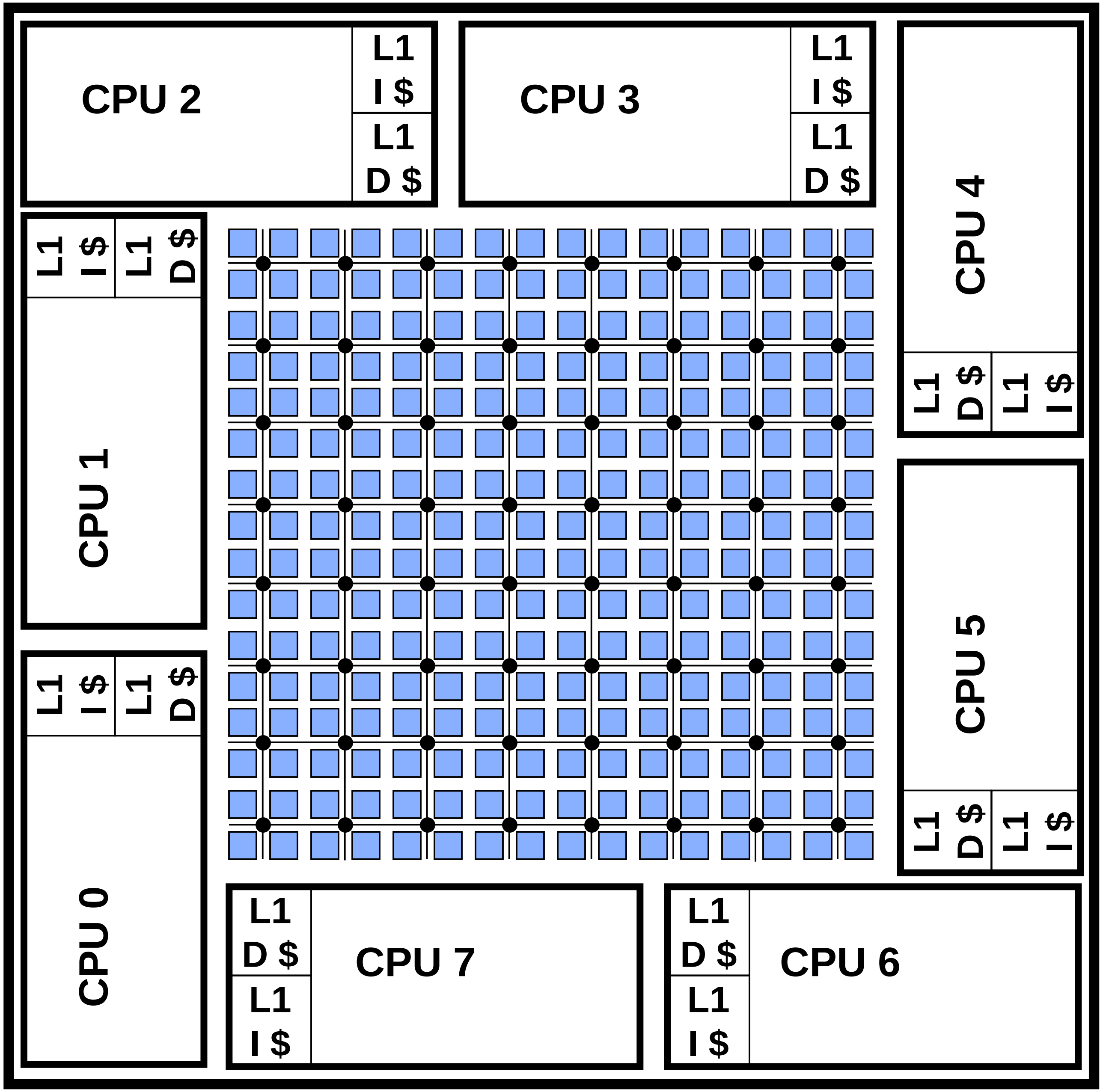
Multifacet Project
University of Wisconsin-Madison

MICRO 2004

Overview

- Managing wire delay in shared CMP caches
- Three techniques extended to CMPs
  1. **On-chip Strided Prefetching** (not in talk – see paper)
     - Scientific workloads: **10%** average reduction
     - Commercial workloads: **3%** average reduction
  2. **Cache Block Migration** (e.g. D-NUCA)
     - Block sharing limits average reduction to **3%**
     - Dependence on difficult to implement smart search
  3. **On-chip Transmission Lines** (e.g. TLC)
     - Reduce runtime by **8%** on average
     - Bandwidth contention accounts for **26%** of L2 hit latency
- **Combining techniques**
  - + Potentially alleviates isolated deficiencies
    - Up to **19%** reduction vs. baseline
  - – Implementation complexity

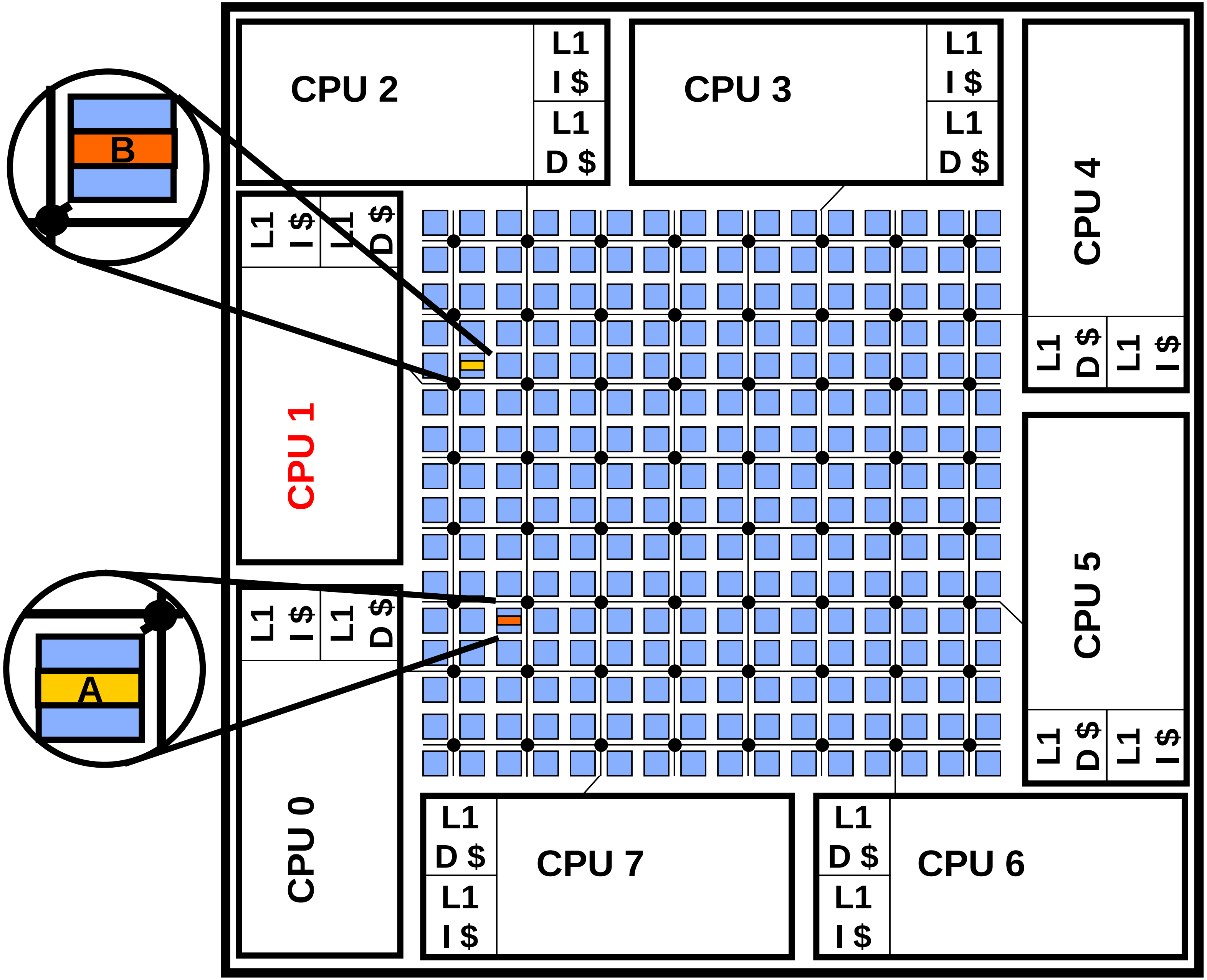
Baseline: CMP-SNUCA



Outline

- Global interconnect and CMP trends
- **Latency Management Techniques**
- Evaluation
  - Methodology
  - **Block Migration**: CMP-DNUCA
  - **Transmission Lines**: CMP-TLC
  - **Combination**: CMP-Hybrid

Block Migration: CMP-DNUCA

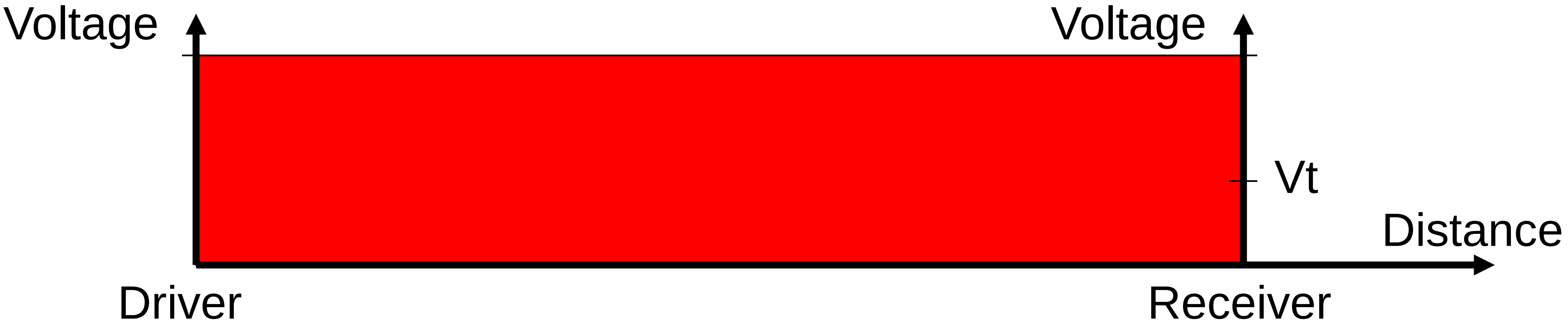


On-chip Transmission Lines

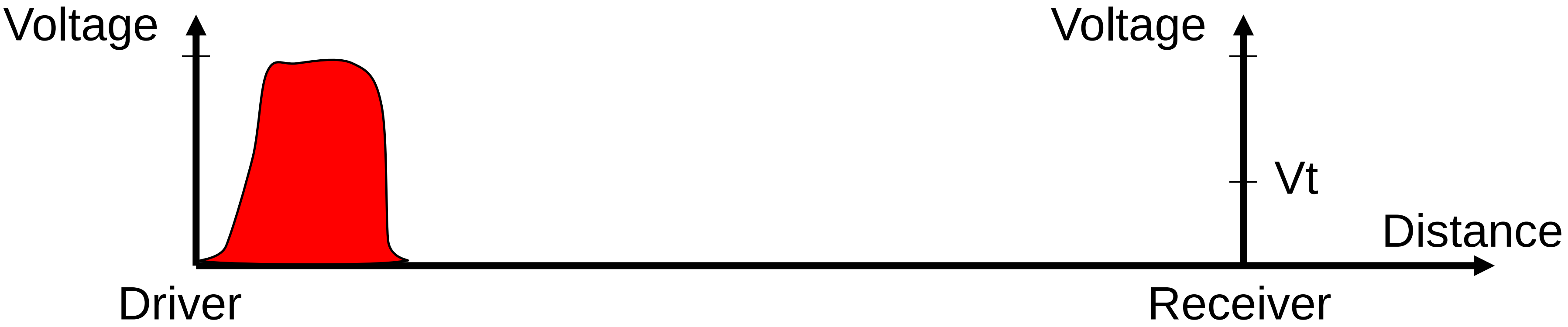
- Similar to contemporary off-chip communication
- Provides a different latency / bandwidth tradeoff
- Wires behave more “transmission-line” like as frequency increases
  - Utilize transmission line qualities to our advantage
  - No repeaters – route directly over large structures
  - **~10x lower latency across long distances**
- Limitations
  - Requires thick wires and dielectric spacing
  - Increases manufacturing cost
- See “TLC: Transmission Line Caches” Beckman, Wood, MICRO’03

RC vs. TL Communication

Conventional Global RC Wire



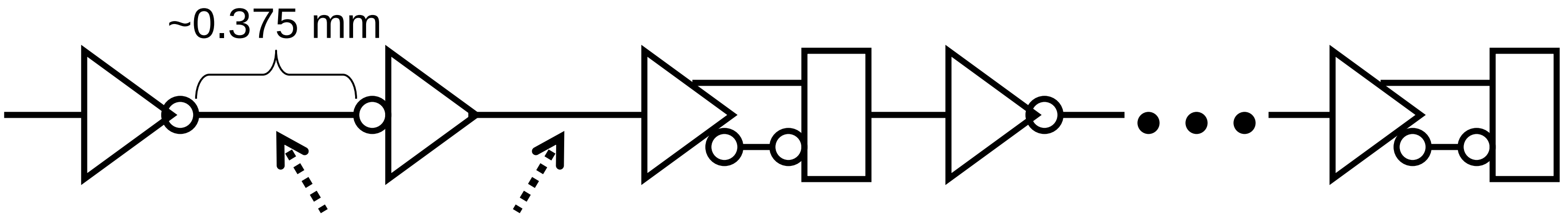
On-chip Transmission Line



MICRO ’03 - TLC: Transmission Line Caches

RC Wire vs. TL Design

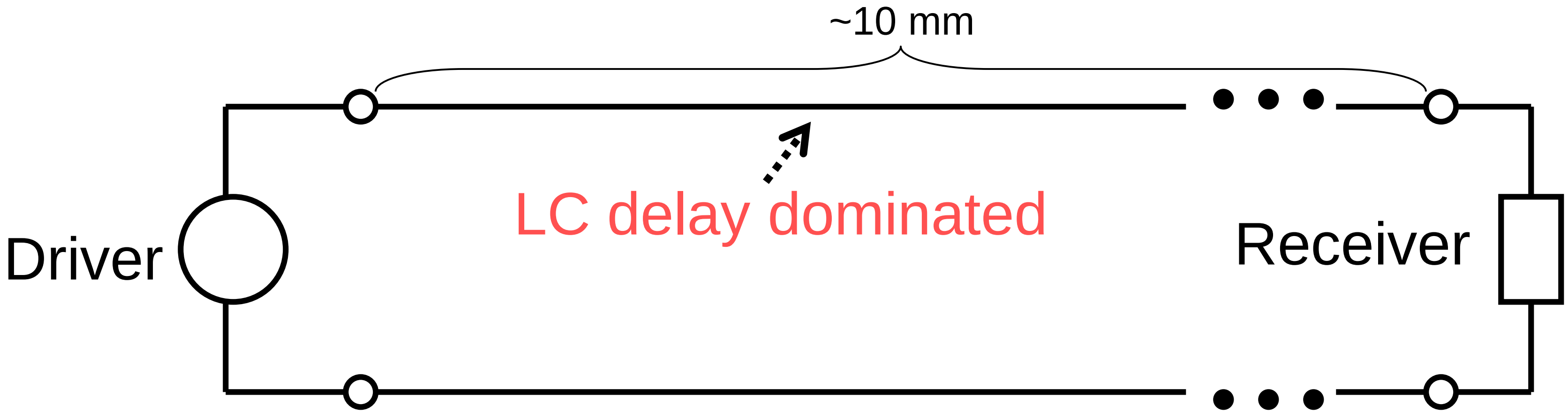
Conventional Global RC Wire



~0.375 mm

RC delay dominated

On-chip Transmission Line



~10 mm

LC delay dominated

Driver

Receiver

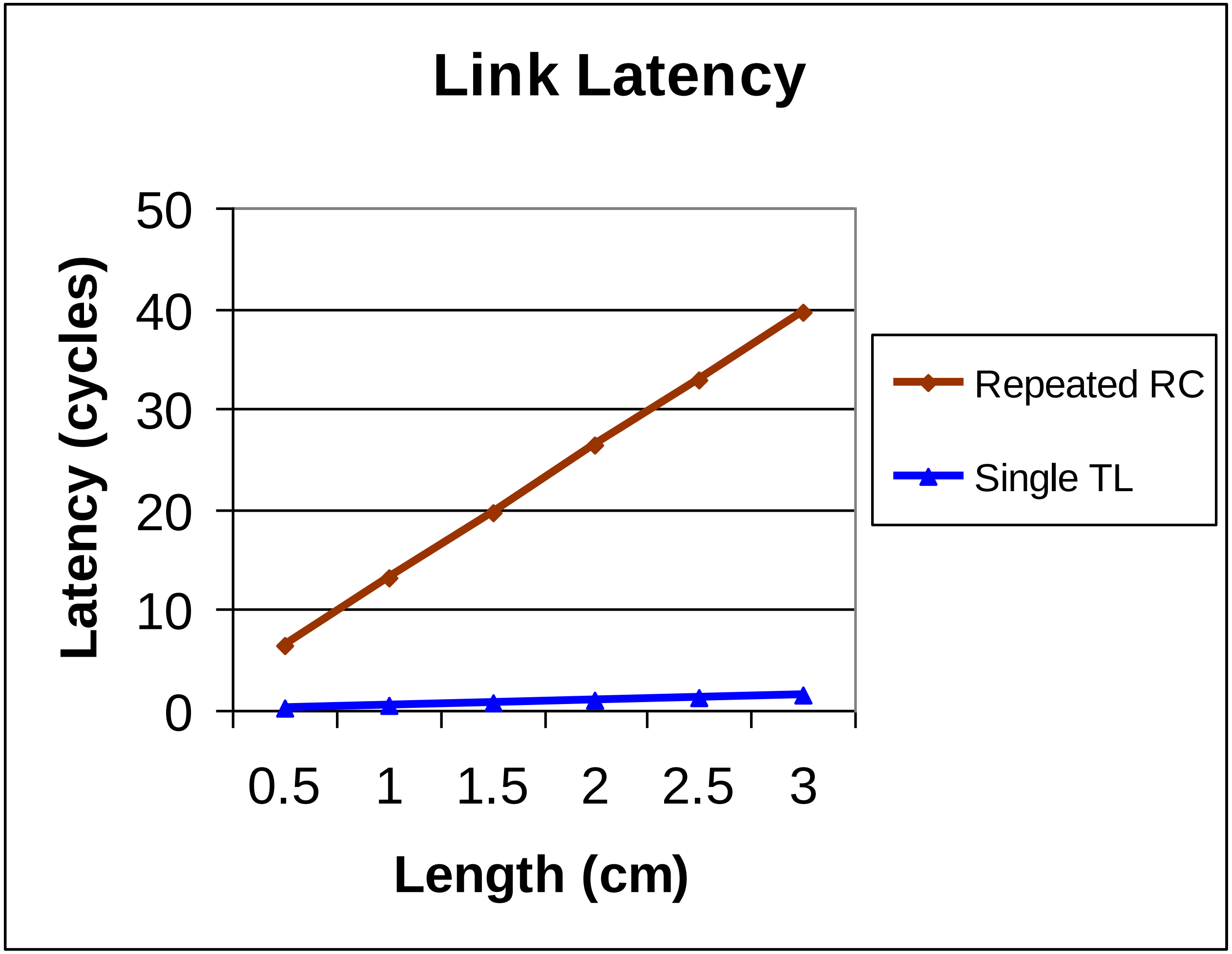
MICRO ’03 - TLC: Transmission Line Caches

On-chip Transmission Lines

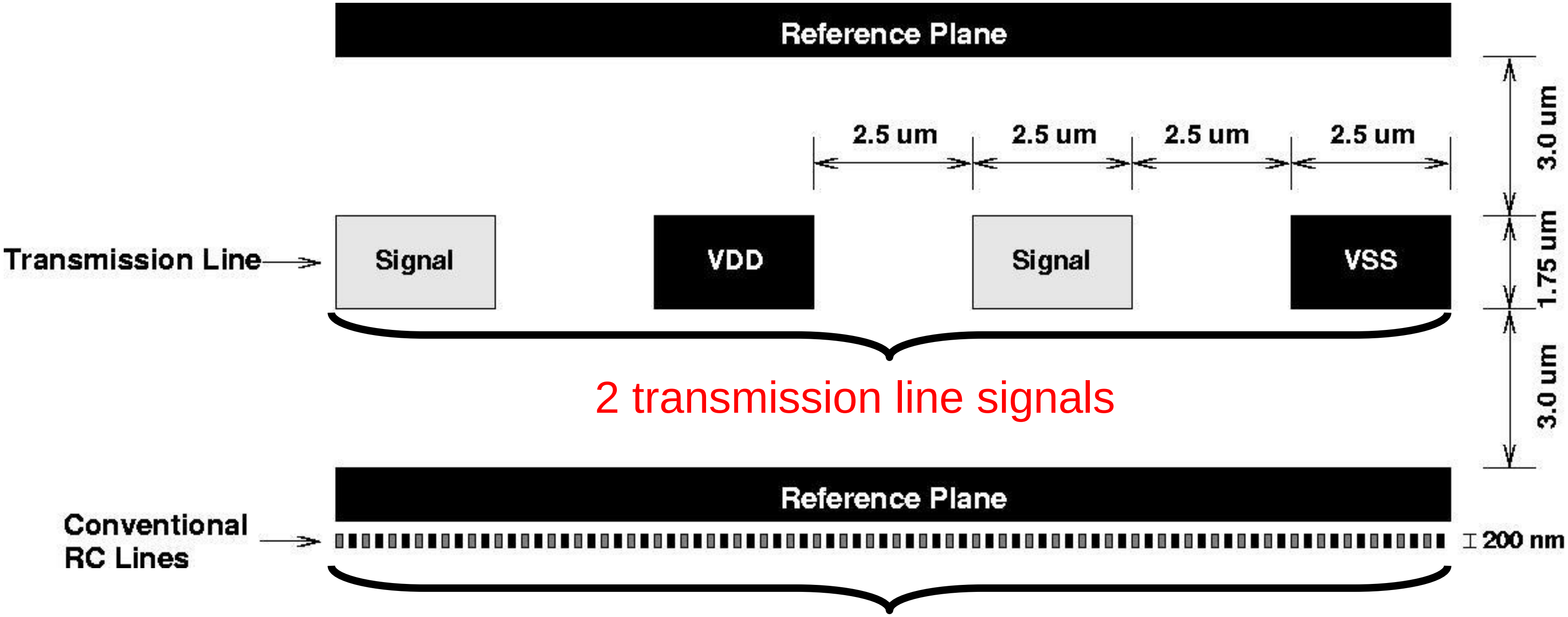
- Why now? → 2010 technology
  - Relative RC delay ↑
  - Improve latency by 10x or more
- What are their limitations?
  - Require thick wires and dielectric spacing
  - Increase wafer cost

Presents a different Latency/Bandwidth Tradeoff

Latency Comparison



Bandwidth Comparison



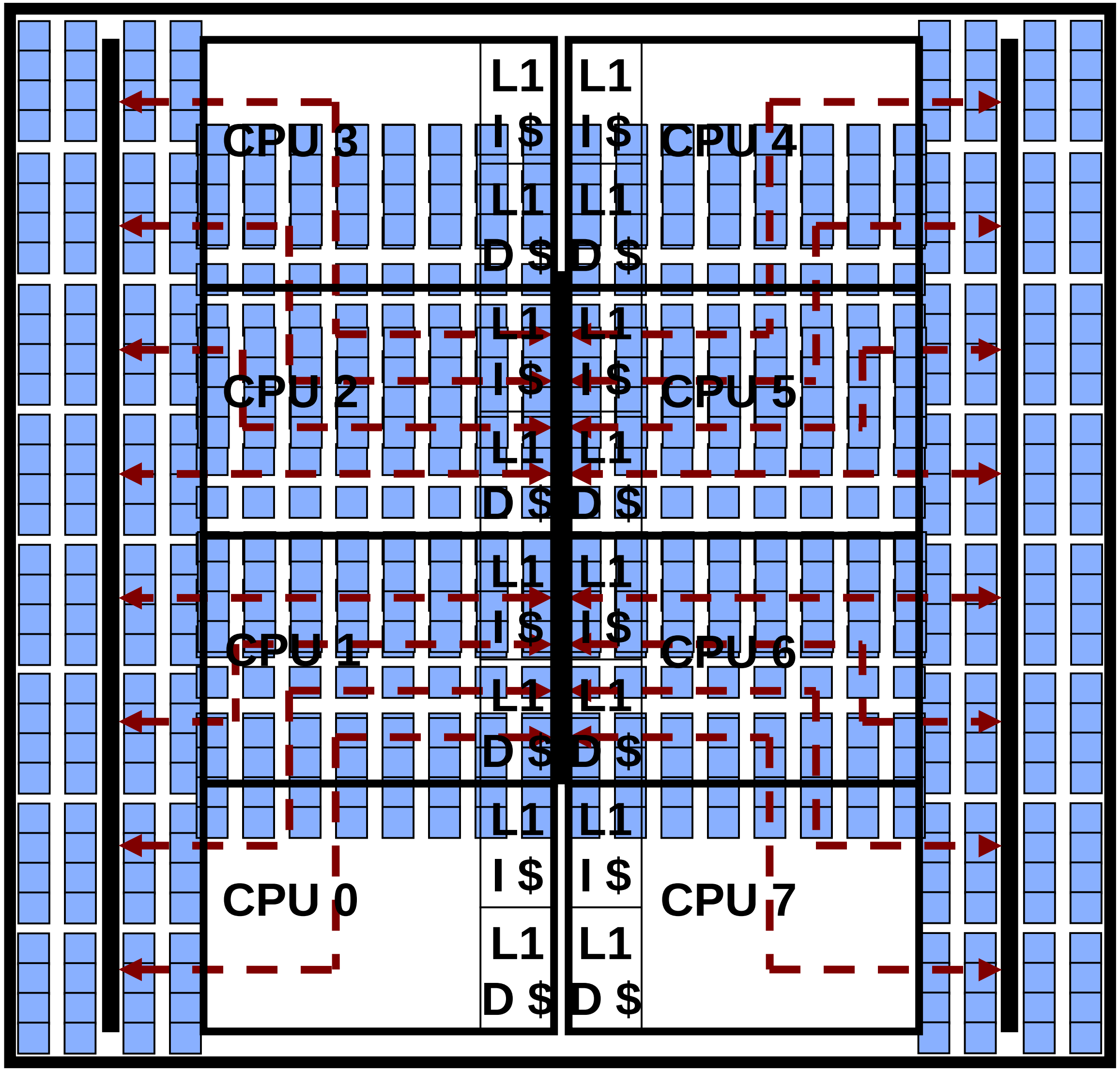
2 transmission line signals

50 conventional signals

Key observation

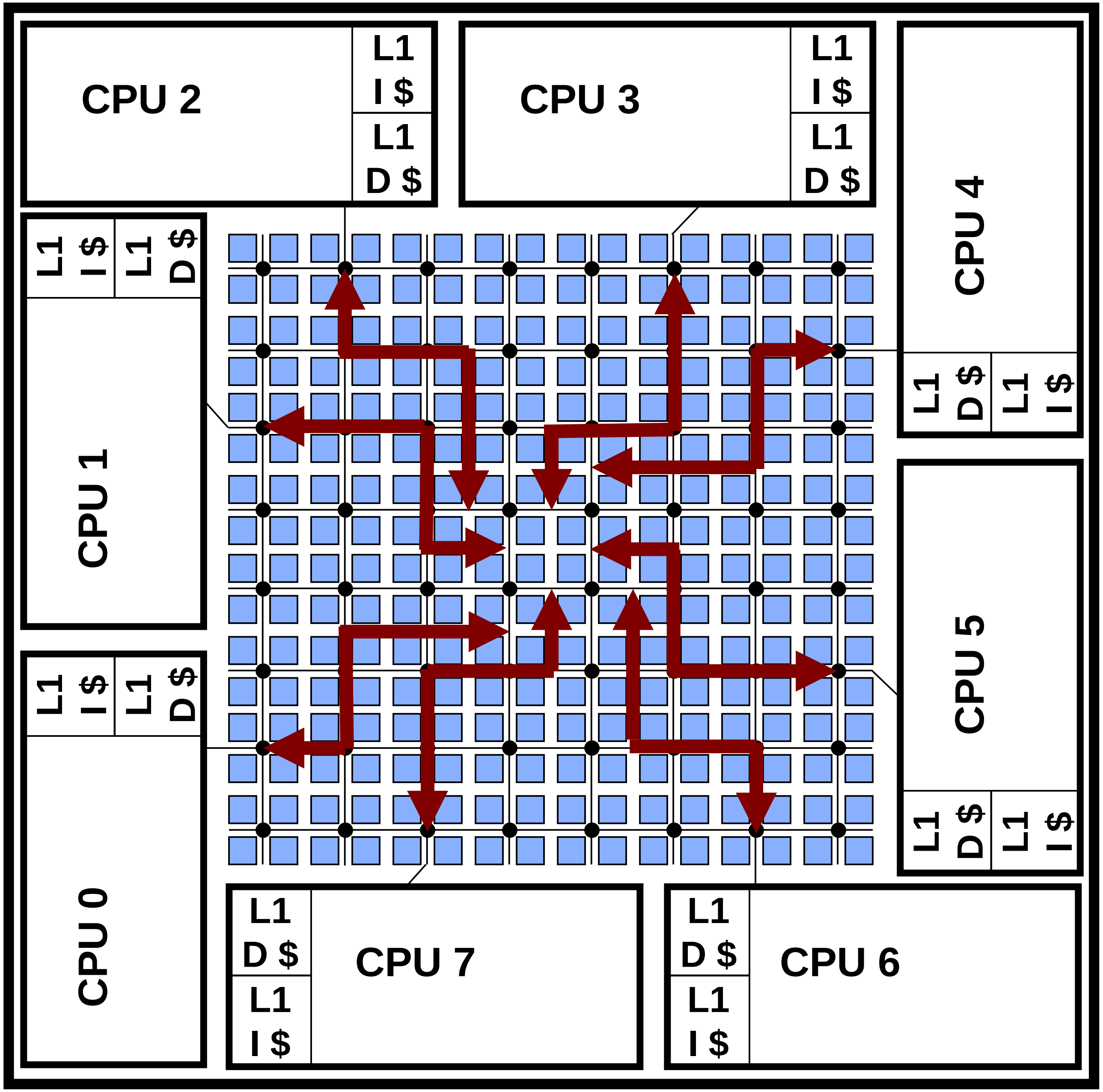
- Transmission lines – route ***over*** large structures
- Conventional wires – substrate area & vias for repeaters

Transmission Lines: CMP-TLC



16
8-byte
links

Combination: CMP-Hybrid



8
32-byte
links

Outline

- Global interconnect and CMP trends
- Latency Management Techniques
- **Evaluation**
  - **Methodology**
  - **Block Migration**: CMP-DNUCA
  - **Transmission Lines**: CMP-TLC
  - **Combination**: CMP-Hybrid

Methodology

- Full system simulation
  - Simics
  - Timing model extensions
    - Out-of-order processor
    - Memory system
- Workloads
  - **Commercial**
    - apache, **jbb**, **otlp**, zeus
  - **Scientific**
    - *Splash*: barnes & **ocean**
    - *SpecOMP*: **apsi** & fma3d

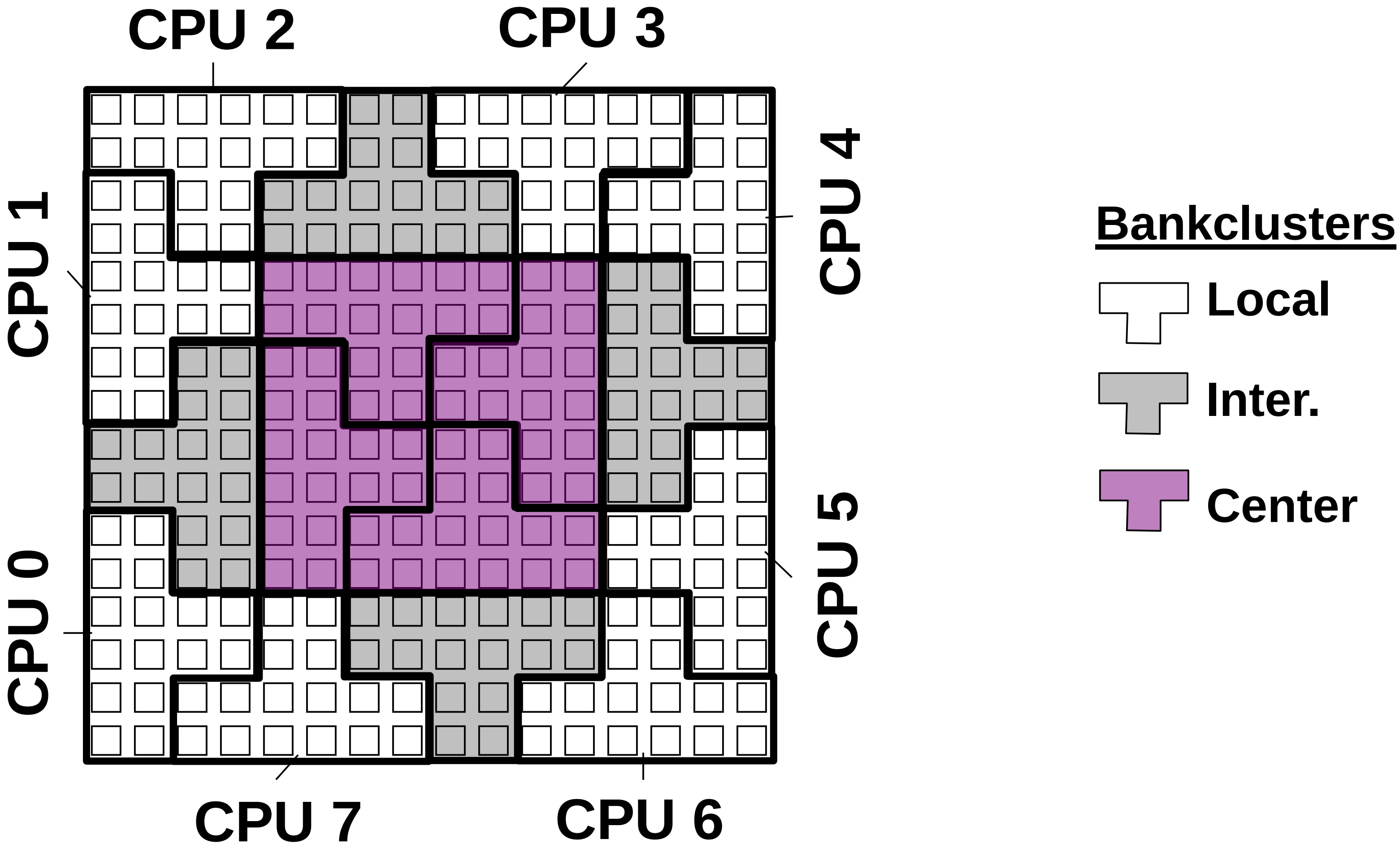
System Parameters

Memory System		Dynamically Scheduled Processor	
L1 I & D caches	64 KB, 2-way, 3 cycles	**Clock frequency**	**10 GHz**
Unified L2 cache	**16 MB, 256x64 KB, 16-way, 6 cycle bank access**	Reorder buffer / scheduler	128 / 64 entries
L1 / L2 cache block size	64 Bytes	Pipeline width	4-wide fetch & issue
Memory latency	260 cycles	Pipeline stages	30
Memory bandwidth	320 GB/s	Direct branch predictor	3.5 KB YAGS
Memory size	4 GB of DRAM	Return address stack	64 entries
Outstanding memory request / CPU	16	Indirect branch predictor	256 entries (cascaded)

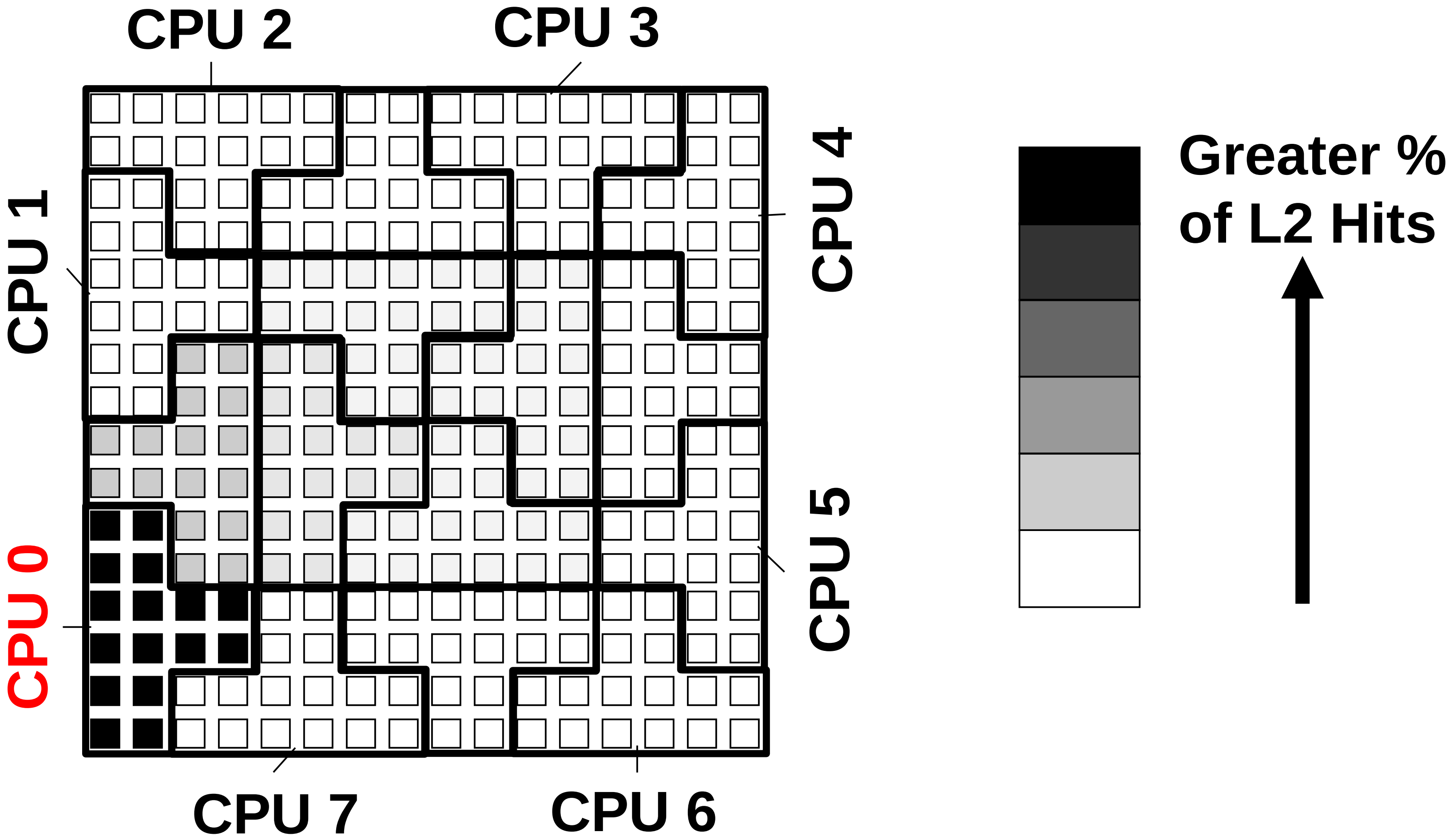
Outline

- Global interconnect and CMP trends
- Latency Management Techniques
- **Evaluation**
  - Methodology
  - **Block Migration: CMP-DNUCA**
  - **Transmission Lines**: CMP-TLC
  - **Combination**: CMP-Hybrid

CMP-DNUCA: Organization

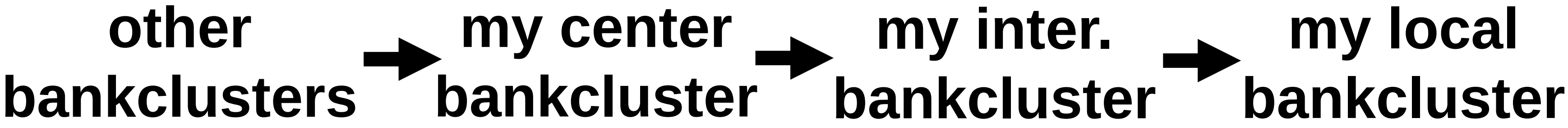


Hit Distribution: Grayscale Shading



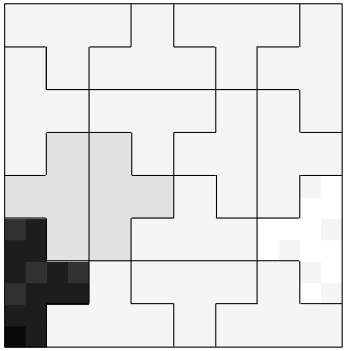
CMP-DNUCA: Migration

- Migration policy
  - **Gradual** movement
  - Increases local hits and reduces distant hits

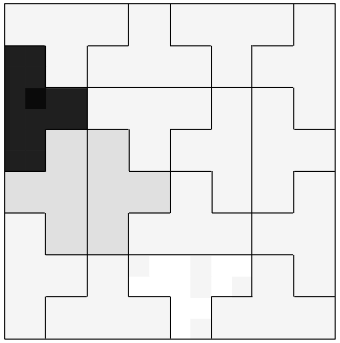


other bankclusters → my center bankcluster → my inter. bankcluster → my local bankcluster

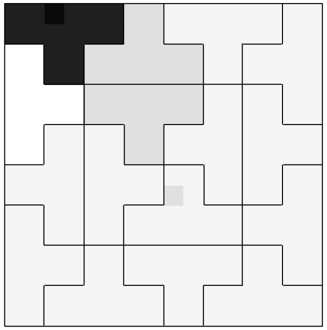
CMP-DNUCA: Hit Distribution Ocean per CPU



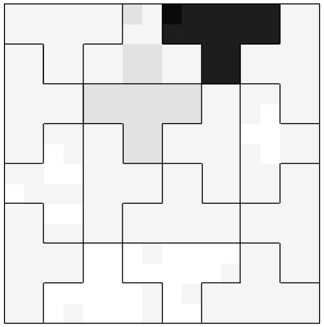
CPU 0



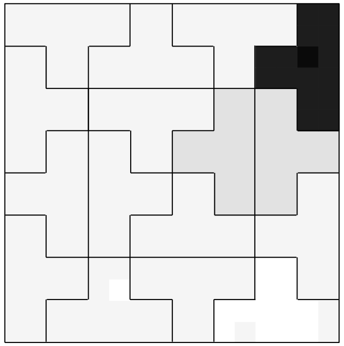
CPU 1



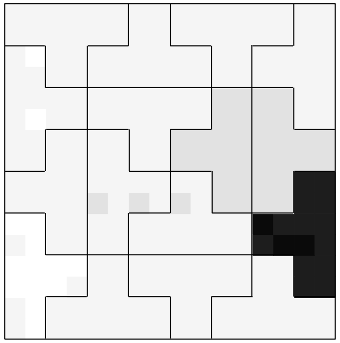
CPU 2



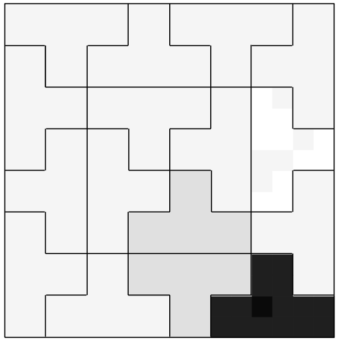
CPU 3



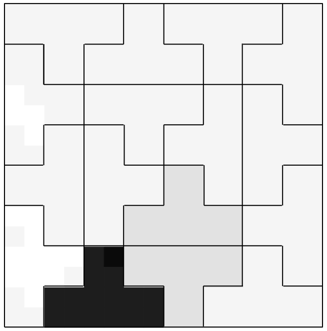
CPU 4



CPU 5

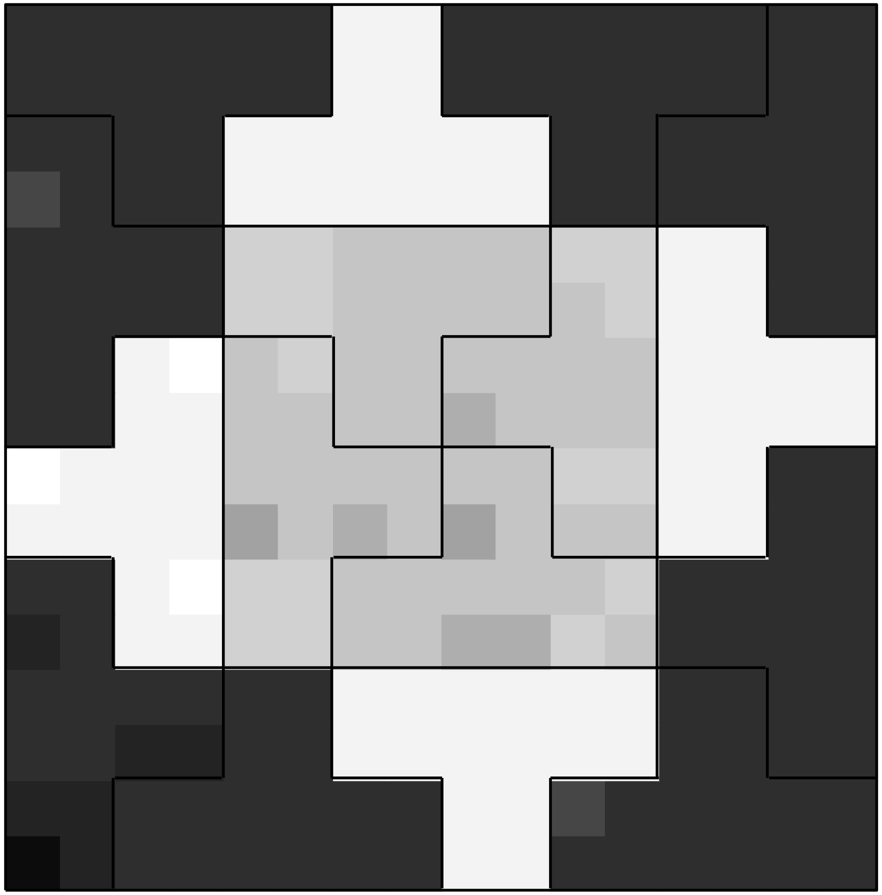


CPU 6



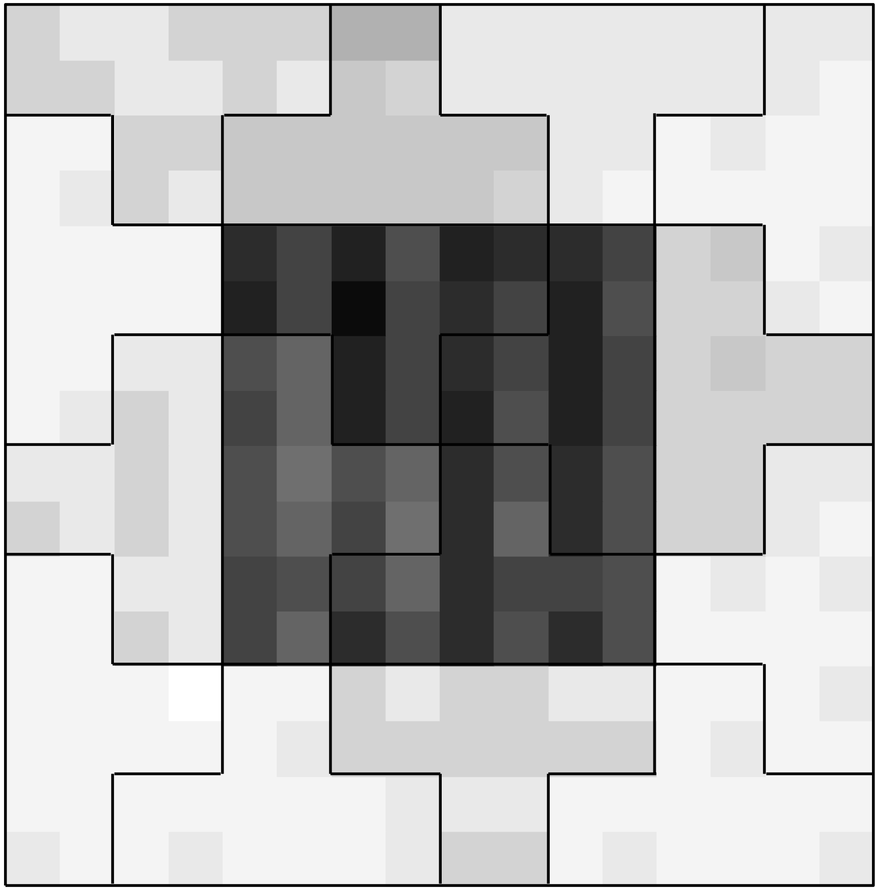
CPU 7

CMP-DNUCA: Hit Distribution Ocean all CPUs

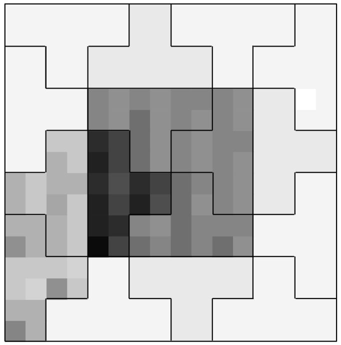


Block migration successfully separates the data sets

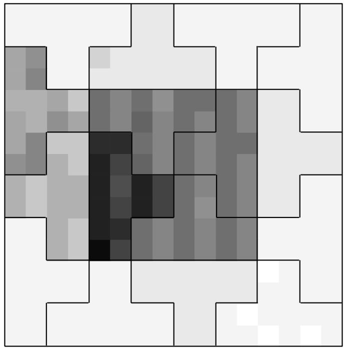
CMP-DNUCA: Hit Distribution OLTP all CPUs



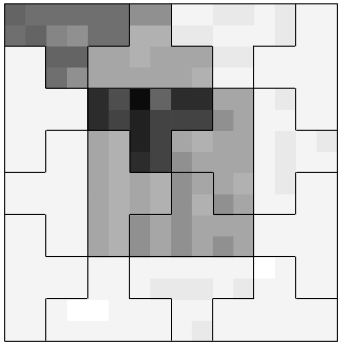
CMP-DNUCA: Hit Distribution OLTP per CPU



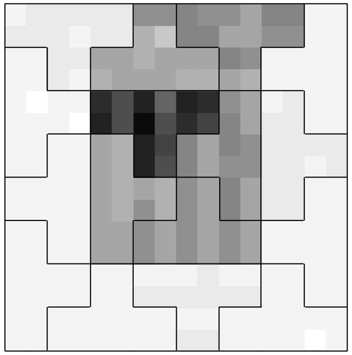
CPU 0



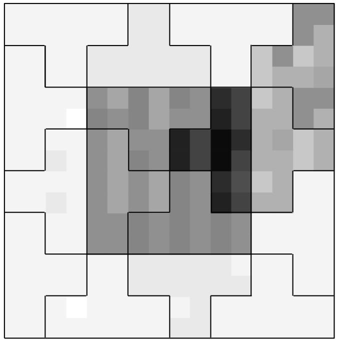
CPU 1



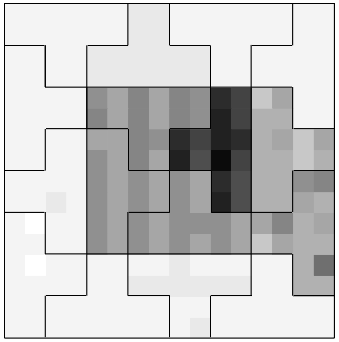
CPU 2



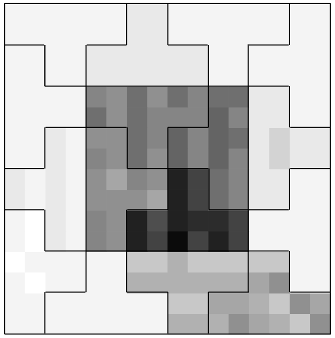
CPU 3



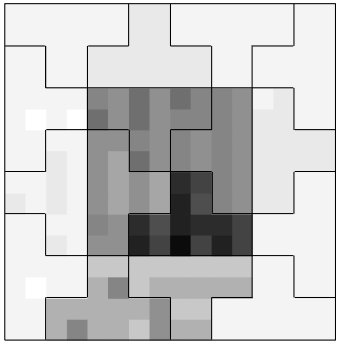
CPU 4



CPU 5



CPU 6



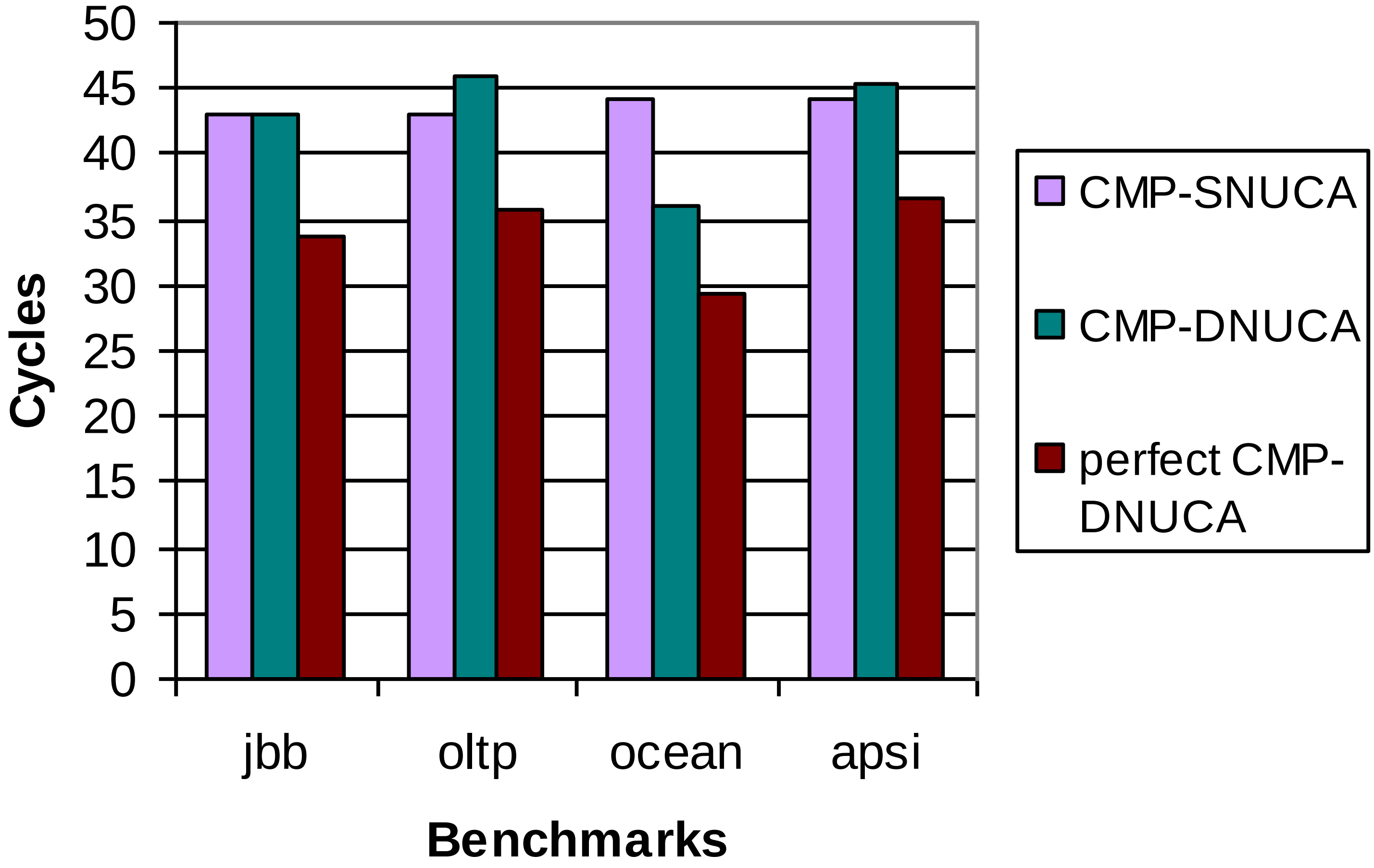
CPU 7

Hit Clustering: Most L2 hits satisfied by the center banks

CMP-DNUCA: Search

- Search policy
  - Uniprocessor DNUCA solution: partial tags
    - Quick summary of the L2 tag state at the CPU
    - **No known practical implementation for CMPs**
      - Size impact of multiple partial tags
      - Coherence between block migrations and partial tag state
  - CMP-DNUCA solution: two-phase search
    - 1st phase: CPU's local, inter., & 4 center banks
    - 2nd phase: remaining 10 banks
    - **Slow 2nd phase hits and L2 misses**

CMP-DNUCA: L2 Hit Latency



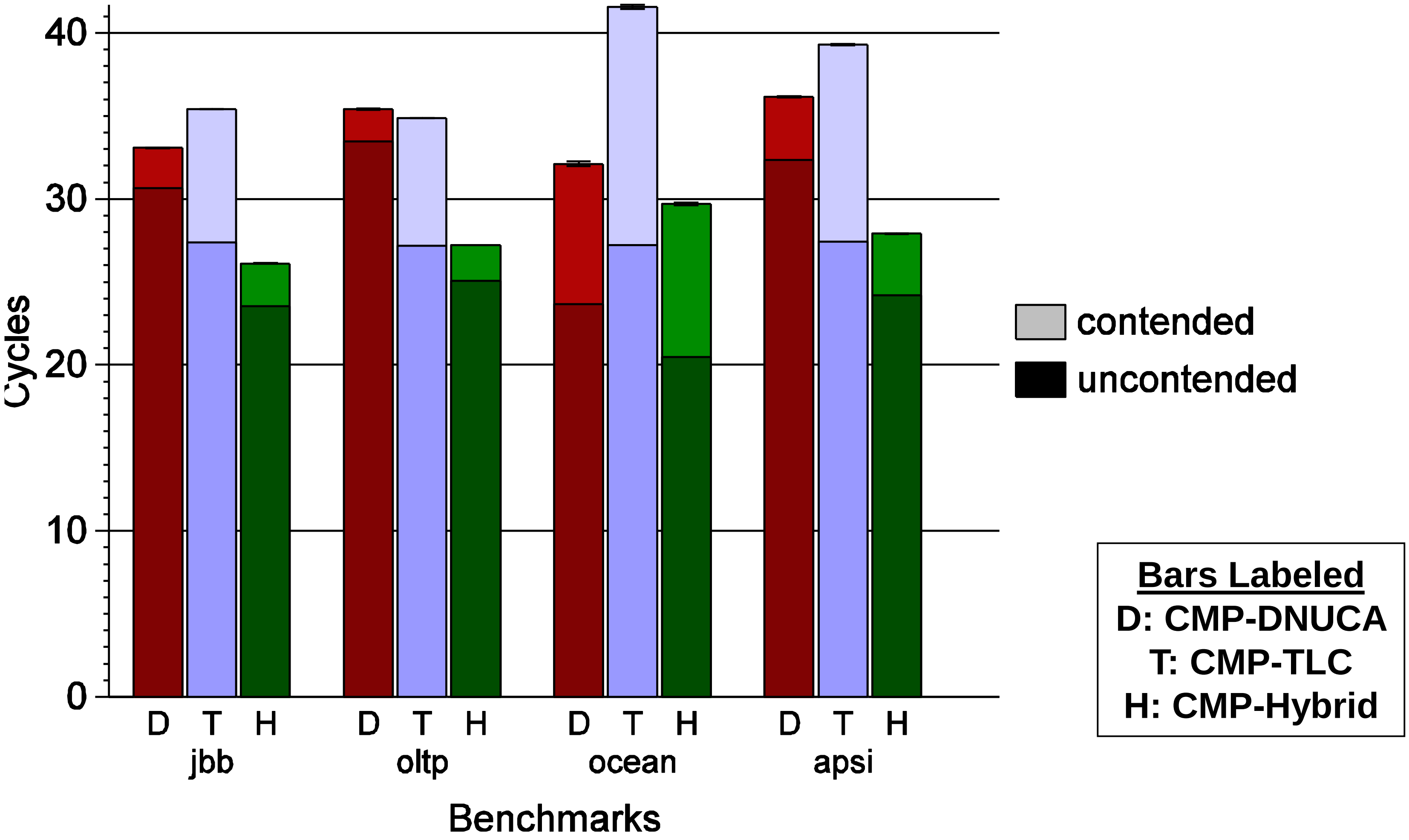
CMP-DNUCA Summary

- Limited success
  - Ocean successfully splits
    - Regular scientific workload – **little sharing**
  - OLTP congregates in the center
    - Commercial workload – **significant sharing**
- Smart search mechanism
  - Necessary for performance improvement
  - **No known implementations**
  - Upper bound – perfect search

Outline

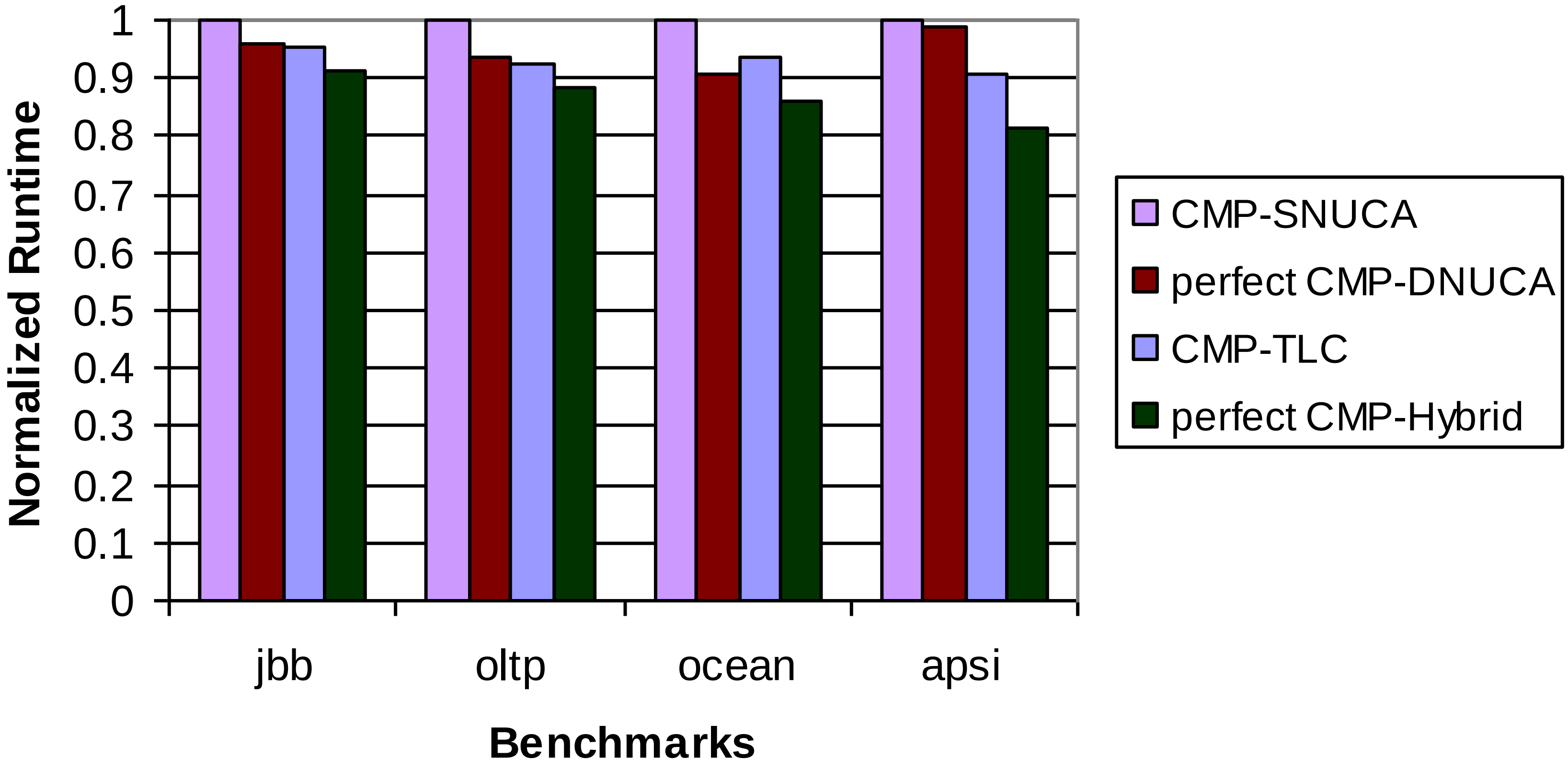
- Global interconnect and CMP trends
- Latency Management Techniques
- **Evaluation**
  - Methodology
  - **Block Migration**: CMP-DNUCA
  - **Transmission Lines: CMP-TLC**
  - **Combination: CMP-Hybrid**

L2 Hit Latency



Bars Labeled
D: CMP-DNUCA
T: CMP-TLC
H: CMP-Hybrid

Overall Performance



Transmission lines improve **L2 hit** and **L2 miss** latency

Conclusions

- Individual Latency Management Techniques
  - Strided Prefetching: **subset of misses**
  - Cache Block Migration: **sharing impedes migration**
  - On-chip Transmission Lines: **limited bandwidth**
- Combination: CMP-Hybrid
  - Potentially alleviates bottlenecks
  - Disadvantages
    - Relies on smart-search mechanism
    - Manufacturing cost of transmission lines

Recap

- Initial NUCA designs → Uniprocessors
  - NUCA:
    - Centralized Partial Tag Array
  - NuRAPID:
    - Decouples Tag and Data Placement
    - More overhead
  - L-NUCA
    - Fine-Grain NUCA close to the core
  - Beckman & Wood:
    - Move Data Close to User
    - Two-Phase Multicast Search
    - Gradual Migration
    - Scientific: Data mostly “private” → move close / fast
    - Commercial: Data mostly “shared” → moves in the center / “slow”

Recap – NUCAs for CMPs

- **Beckman & Wood:**
  - Move Data Close to User
  - Two-Phase Multicast Search
  - Gradual Migration
  - Scientific: Data mostly “private” → move close / fast
  - Commercial: Data mostly “shared” → moves in the center / “slow”
- **CMP-NuRapid:**
  - Per core, L2 tag array
    - Area overhead
    - Tag coherence