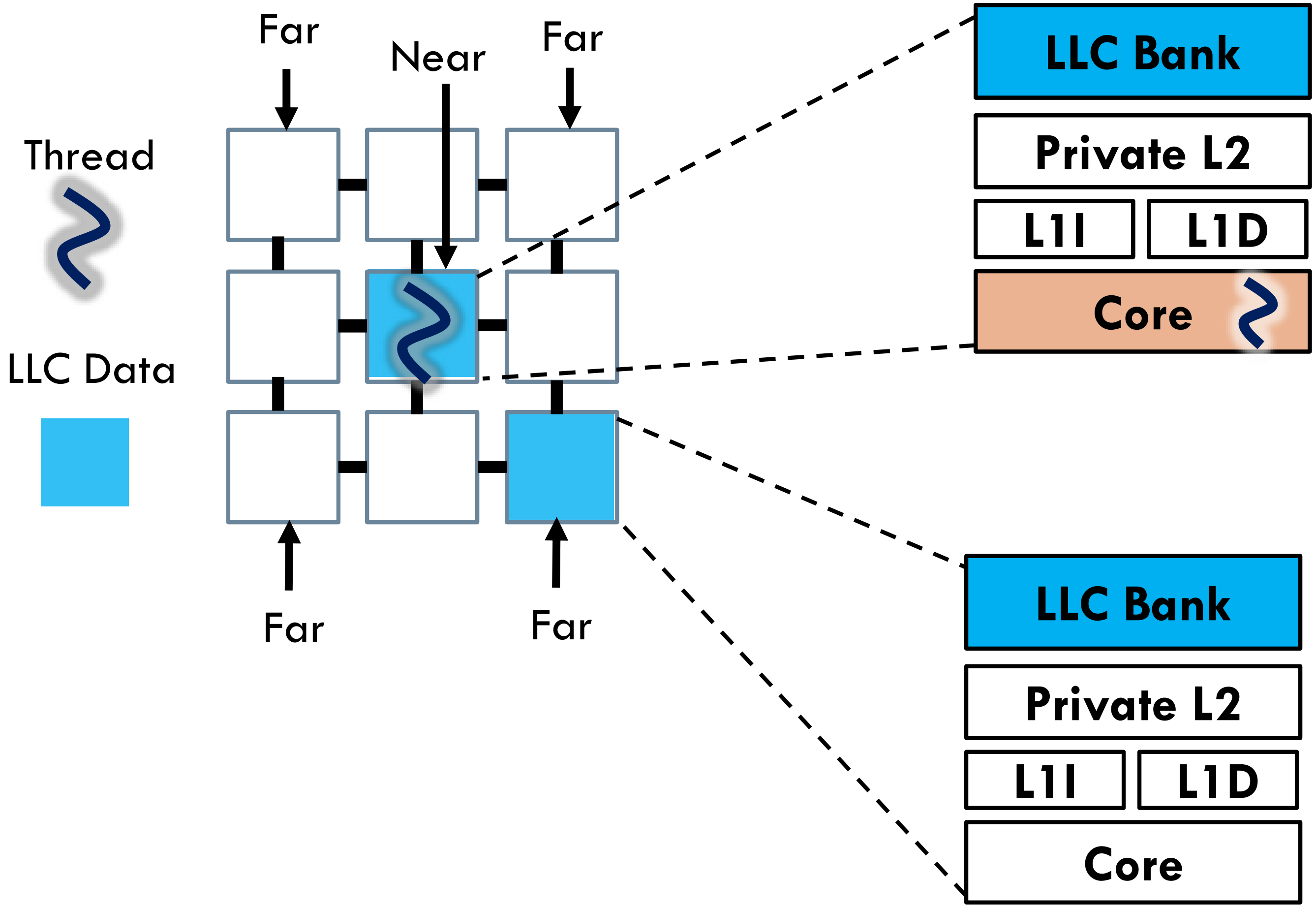
Scaling Distributed Cache Hierarchies through Computation and Data Co-Scheduling

Nathan Beckmann, Po-An Tsai and Daniel Sanchez
MIT CSAIL

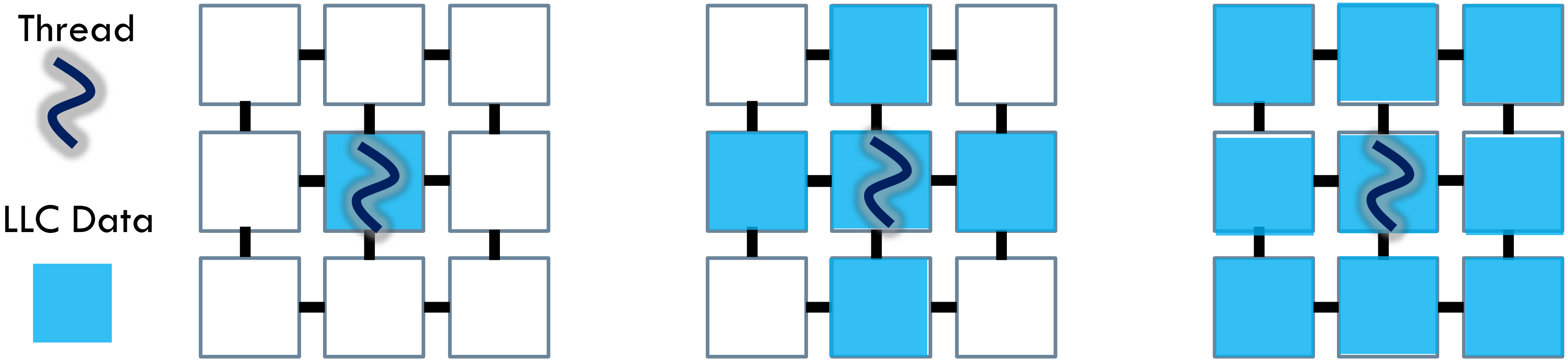
Massachusetts Institute of Technology

CSAIL

Executive Summary



Executive Summary



Many misses
Low hit latency

Moderate misses
Medium hit latency

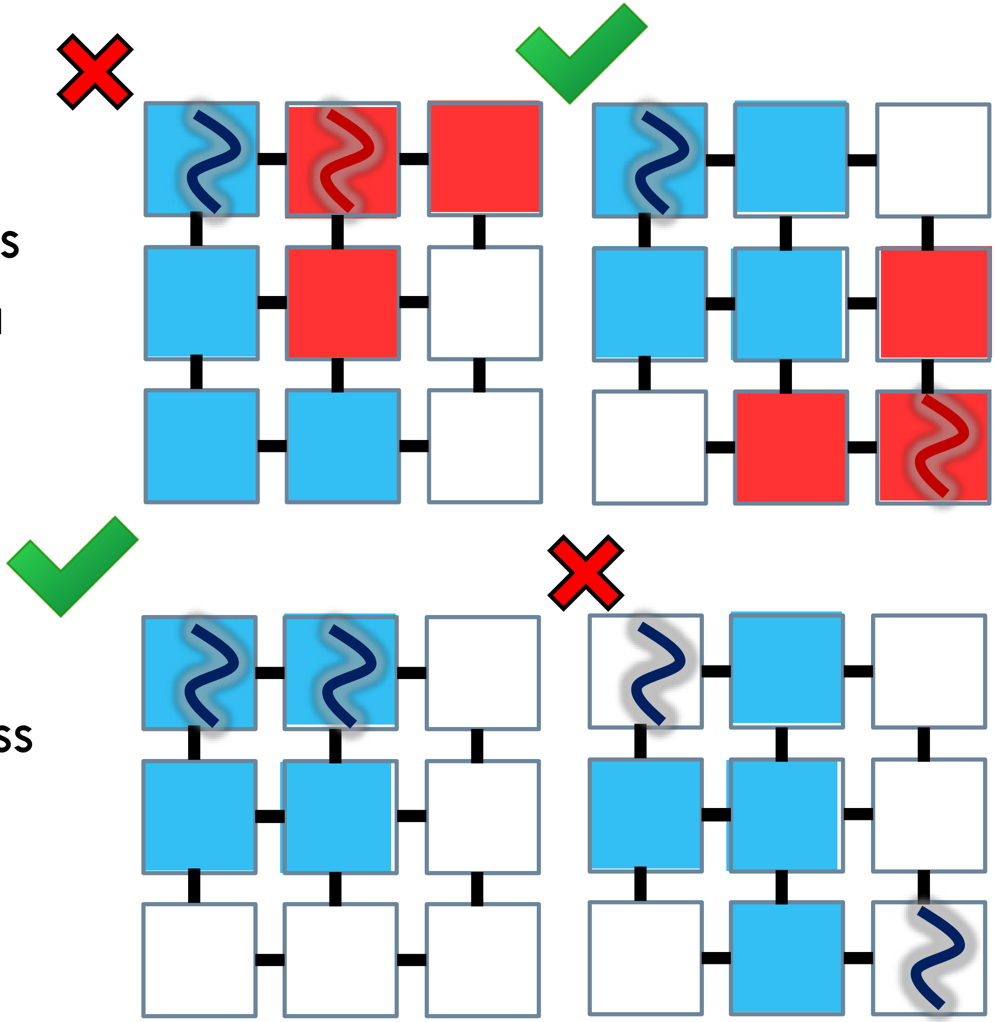
Few misses
High hit latency

More capacity does not always mean better performance

Executive Summary

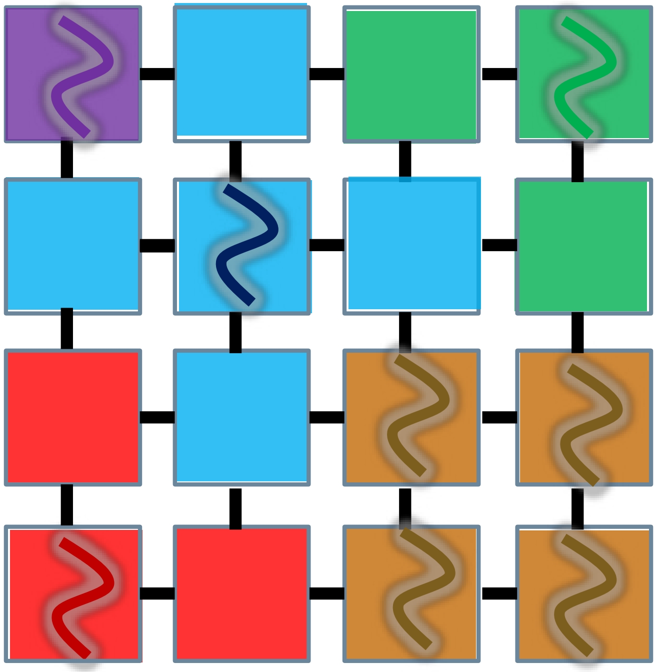
Threads access different data

Threads access same data



Executive Summary

- CDCS jointly places threads and data to reduce data movement



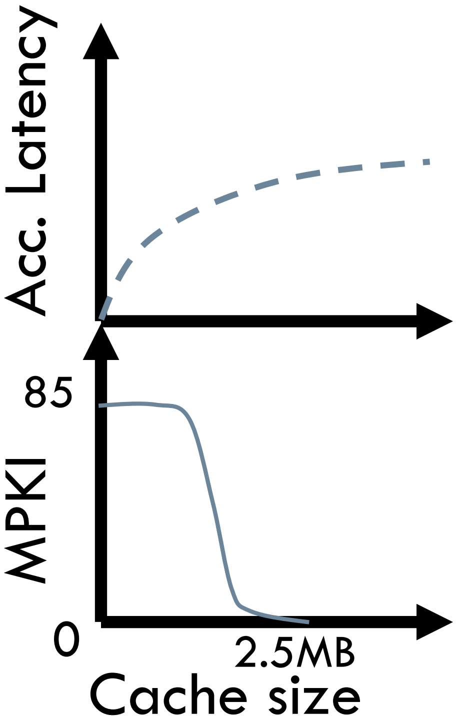
  - Improves performance by 46% on average and by up to 76%
  - Saves 36% of system energy
  - Uses low-overhead algorithms that perform within 1% of impractical, idealized solutions

Agenda

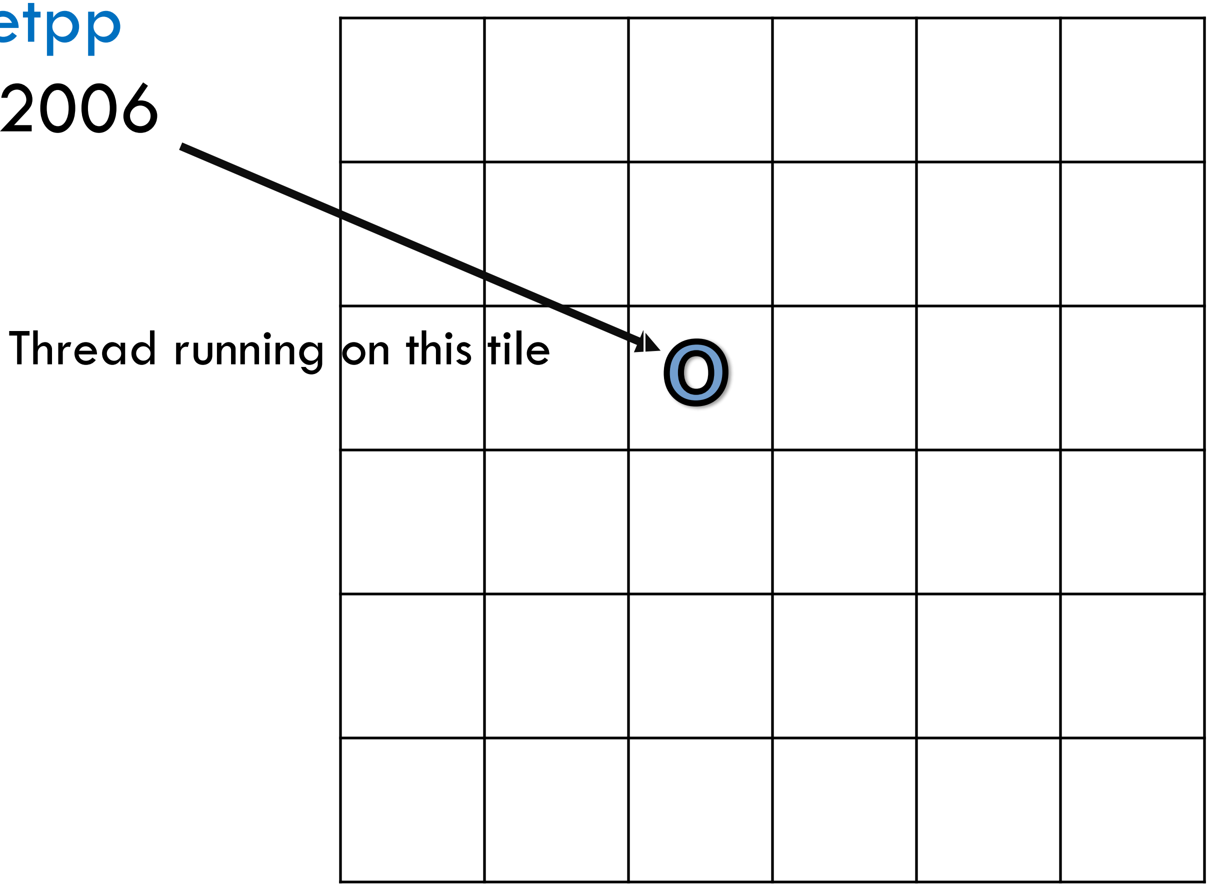
- □ Background
- □ CDCS Design
- □ Evaluation

Capacity vs latency

App: 471.omnetpp
from SPECCPU2006

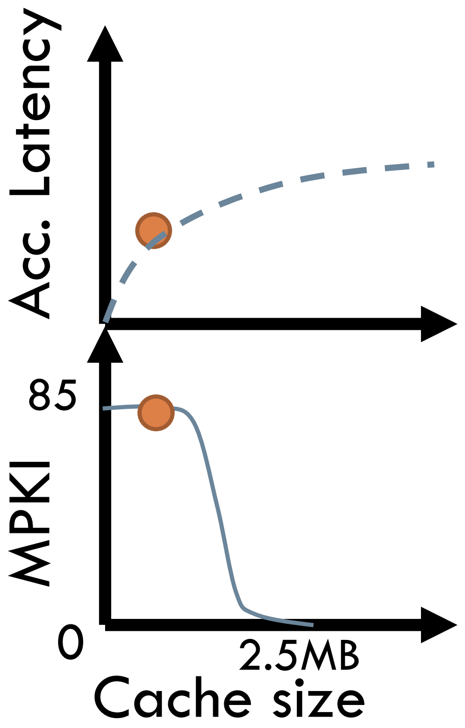


6x6 mesh, 18MB NUCA

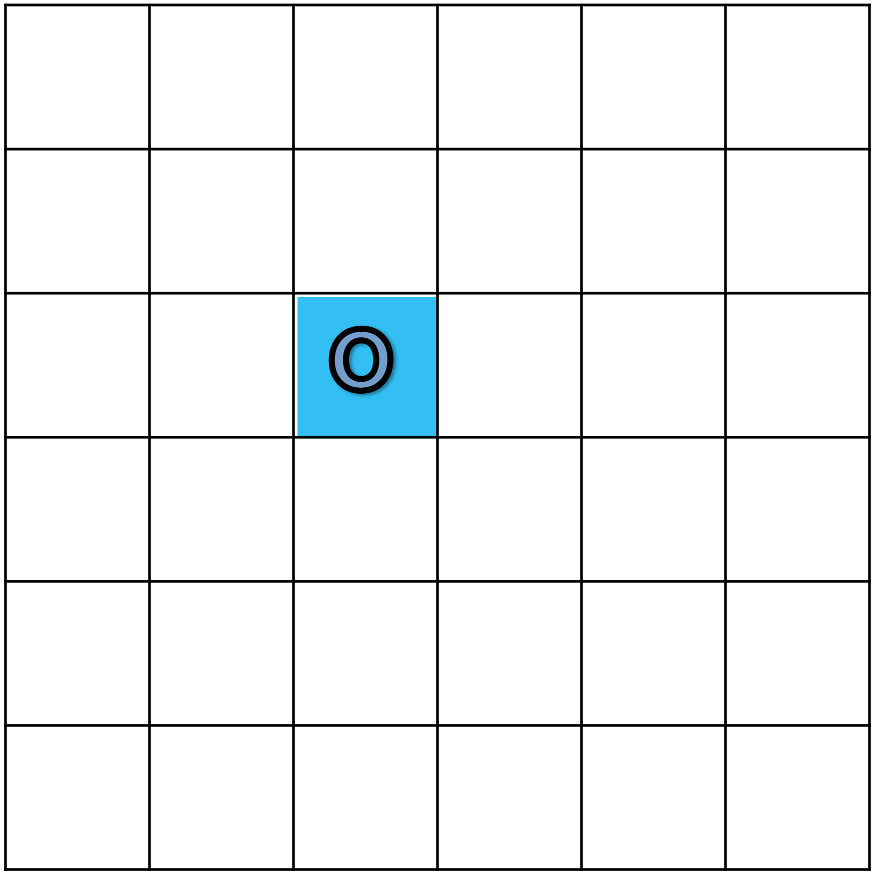


Capacity vs latency

App: 471.omnetpp
from SPECCPU2006



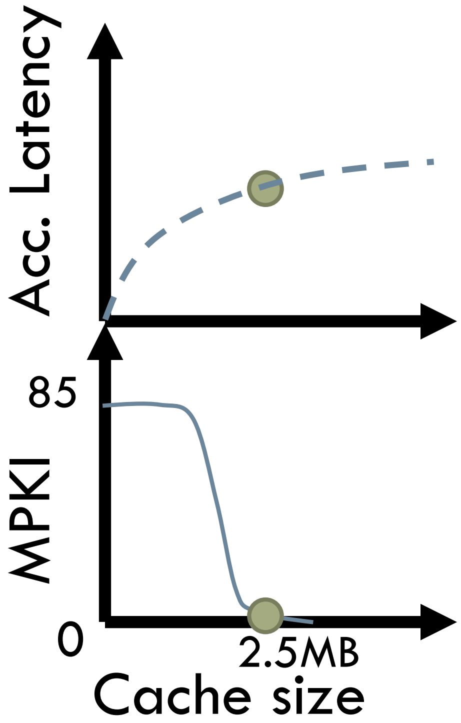
Place data in local bank



Capacity vs latency

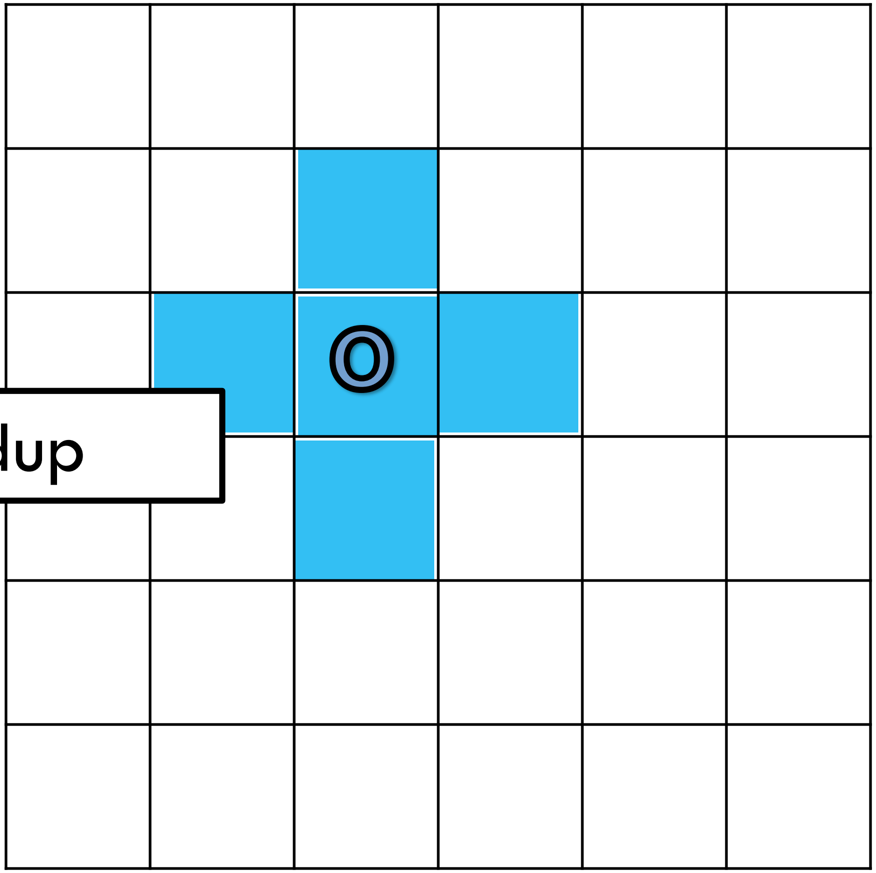
Use closest banks that just fit working set

App: 471.omnetpp
from SPECCPU2006



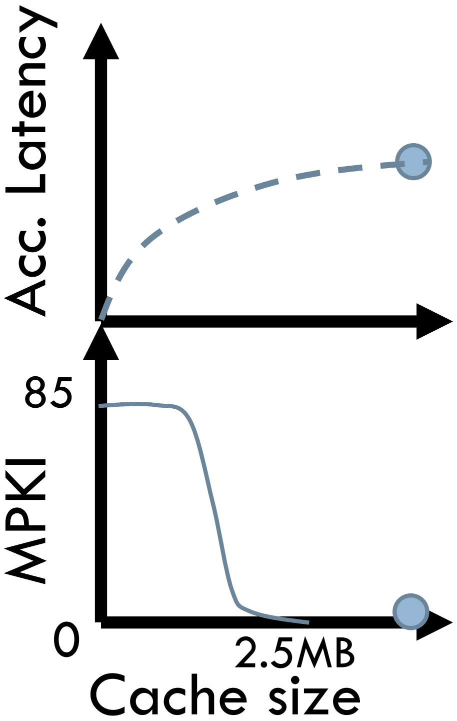
3.7x speedup

Banks	Perf
1	1x



Capacity vs latency

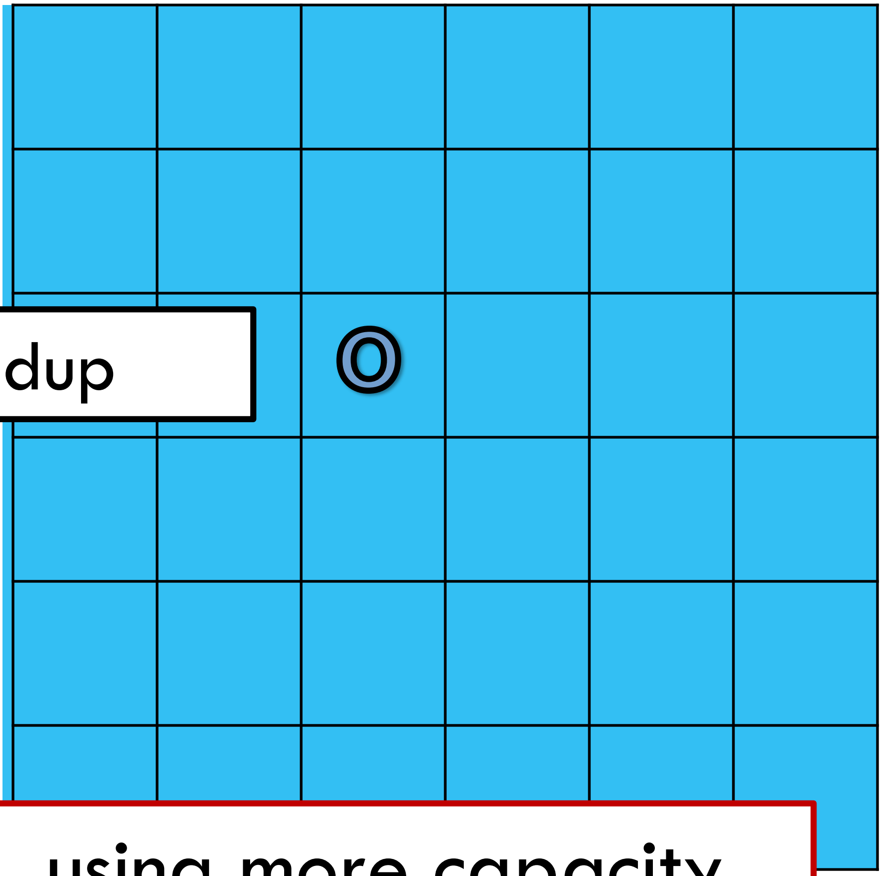
App: 471.omnetpp
from SPECCPU2006



2.4x speedup

Banks	Perf
1	1x
5	3.7x

Place data across the chip

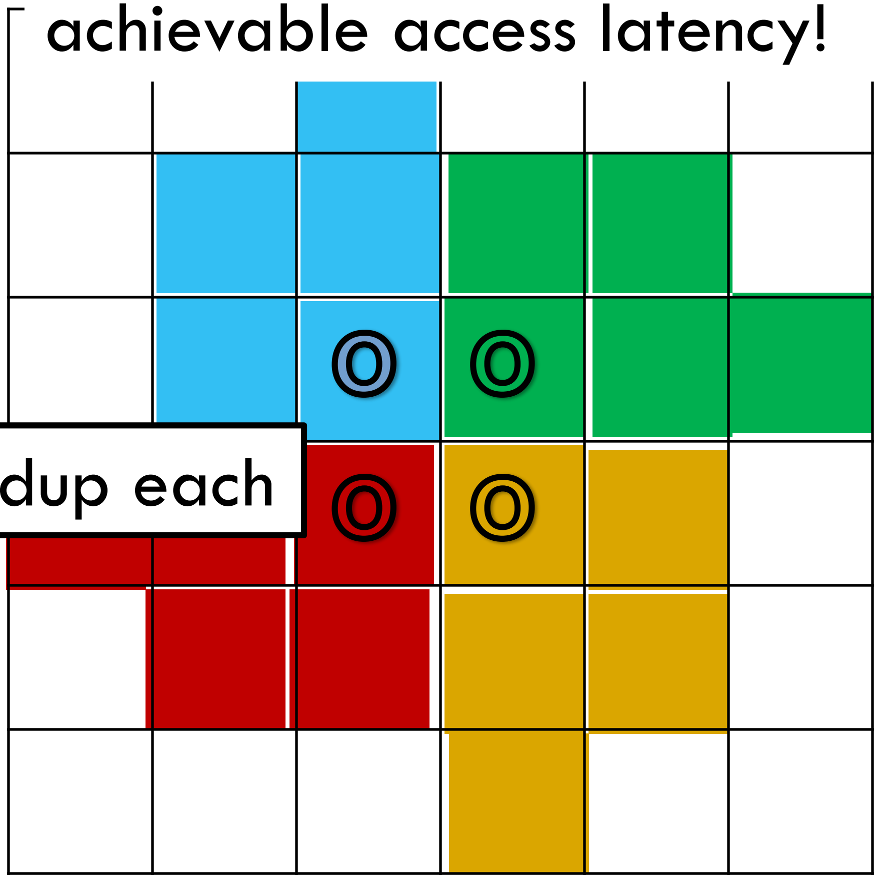


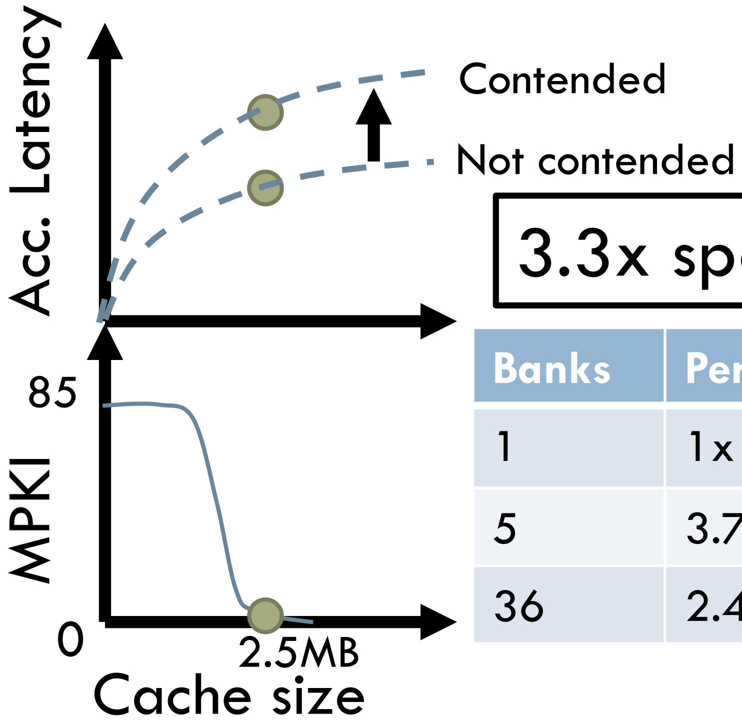
In NUCA, using more capacity than needed is detrimental!

Thread placement matters

App: 471.omnetpp
471.omnetpp
471.omnetpp
471.omnetpp

Capacity contention changes achievable access latency!



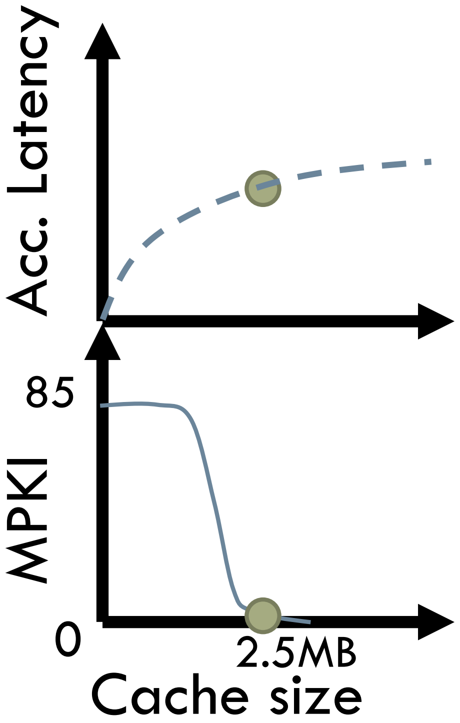


3.3x speedup each

Banks	Perf
1	1x
5	3.7x
36	2.4x

Thread placement matters

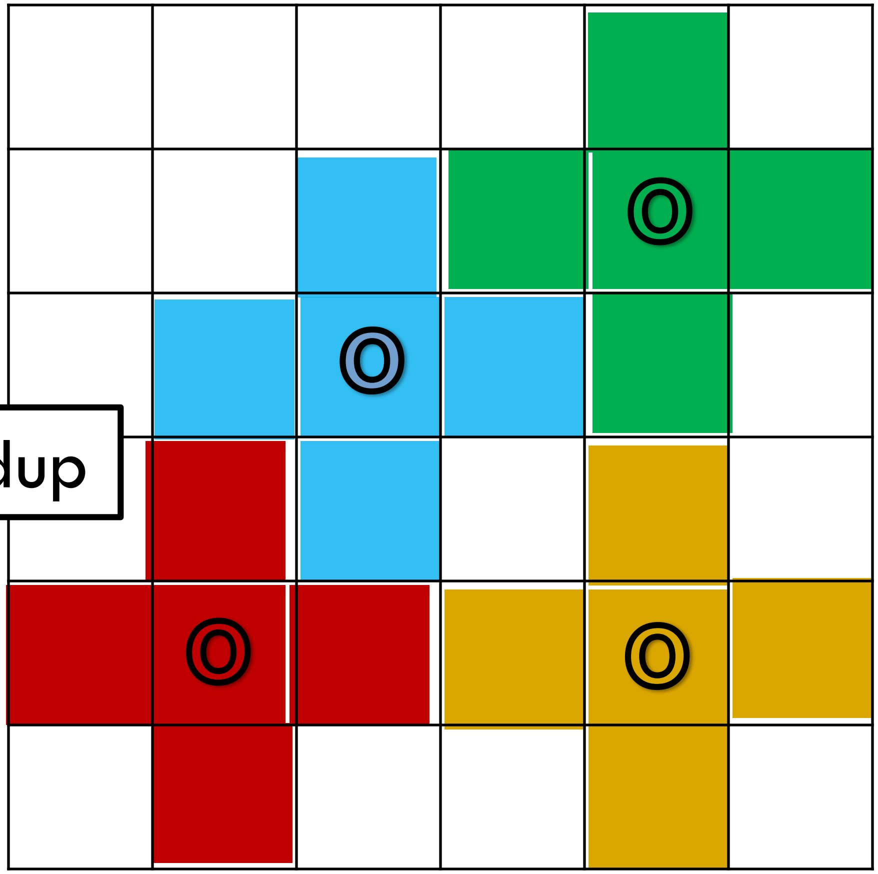
App: 471.omnetpp
471.omnetpp
471.omnetpp
471.omnetpp



3.7x speedup

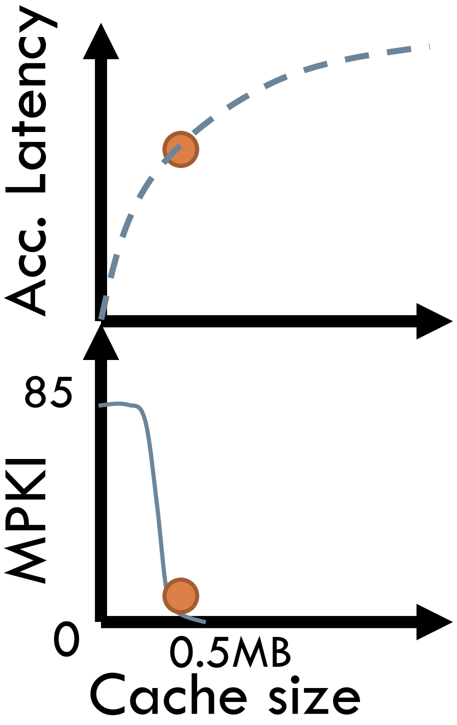
Banks	Perf
1	1x
5	3.7x
36	2.4x

Spread out threads



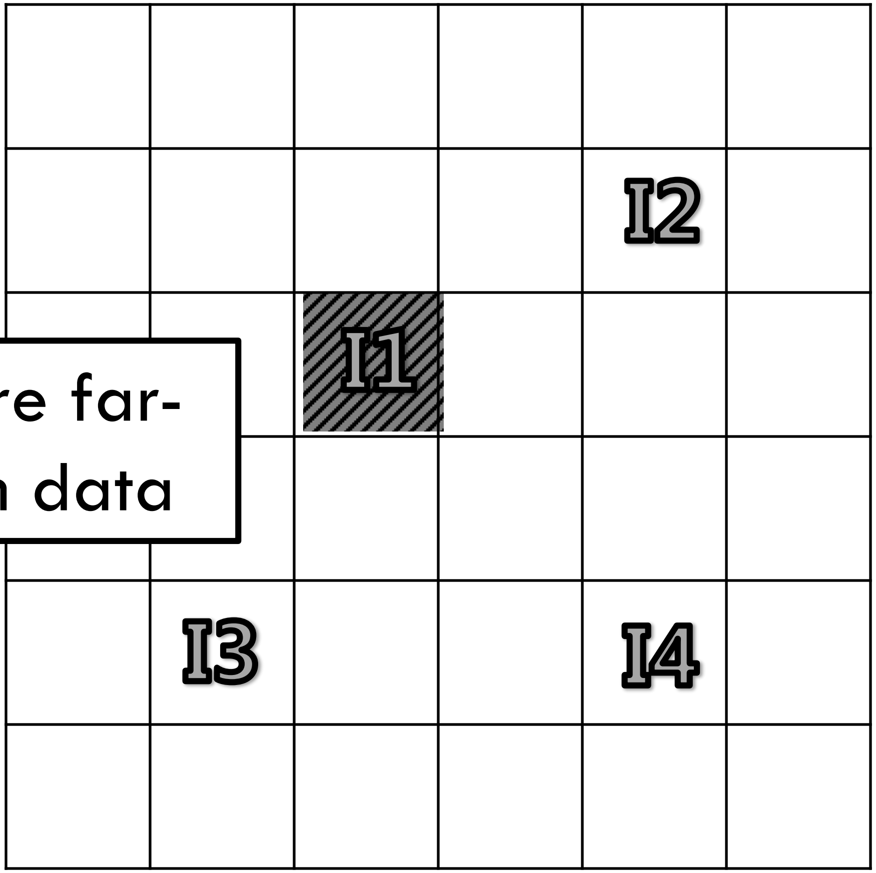
Thread placement matters

App: 4-thread 360.ilbdc from SPECOMP2012



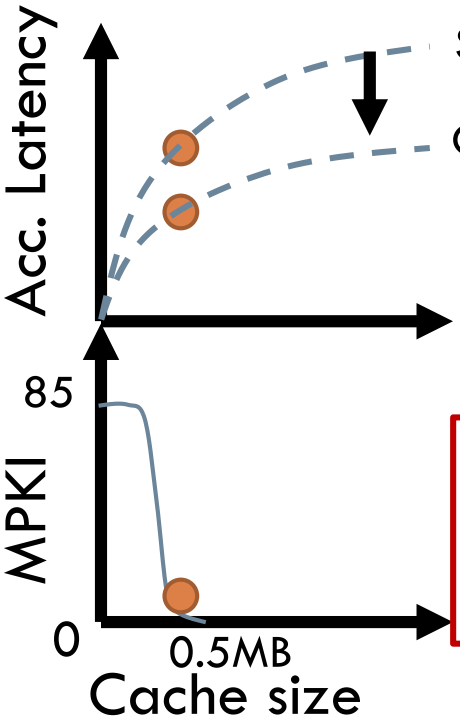
Threads are far-away from data

Spread out threads

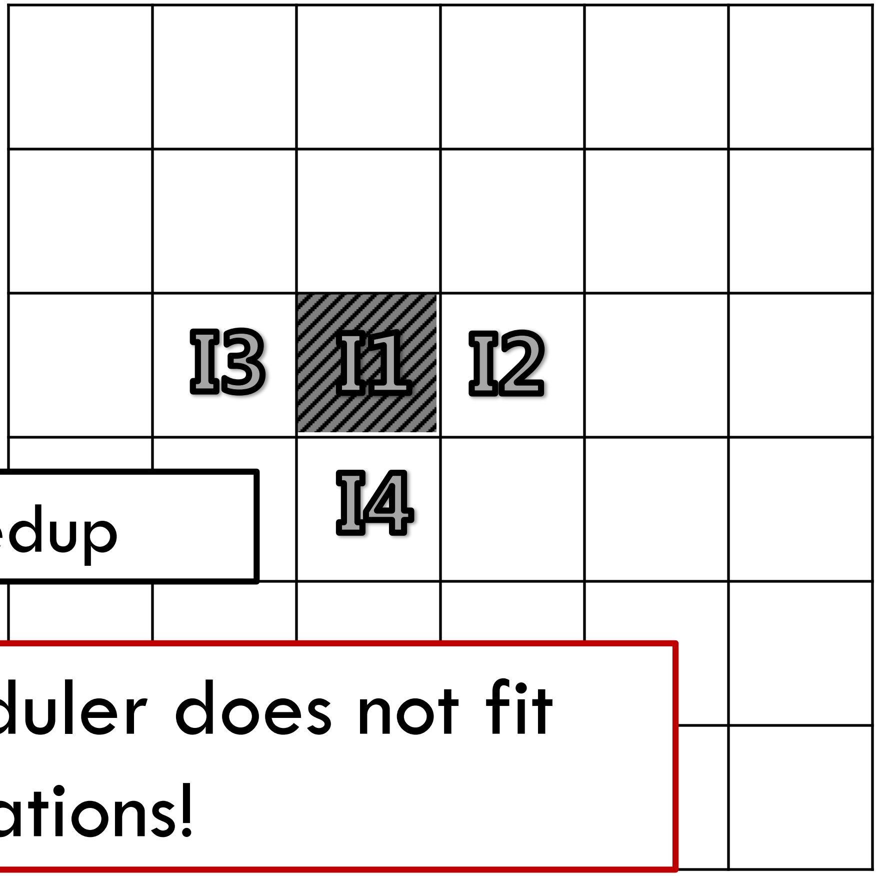


Thread placement matters

App: 4-thread 360.ilbdc
from SPECOMP2012



Cluster threads



1.4x speedup

One scheduler does not fit all applications!

Dynamic NUCA

Mix:

4 x 471.omnetpp

4-thread 360.ilbdc

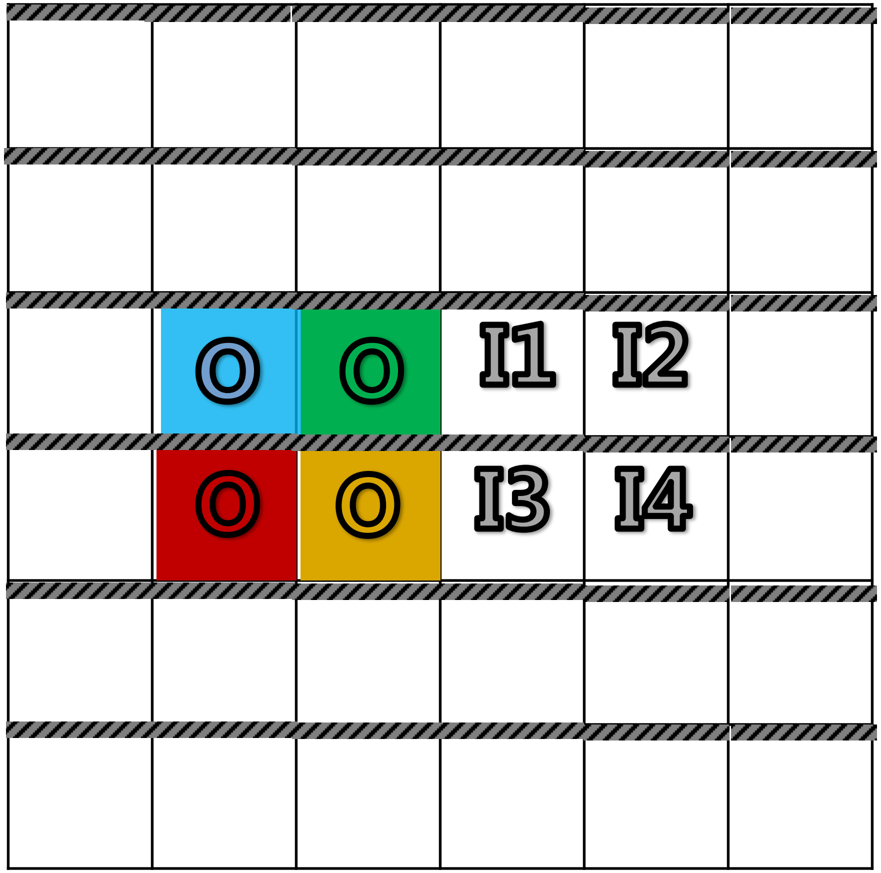
D-NUCA

✓ Place data

✗ Control capacity

✗ Place threads

R-NUCA [Hardavellas'09]



Partitioned NUCA

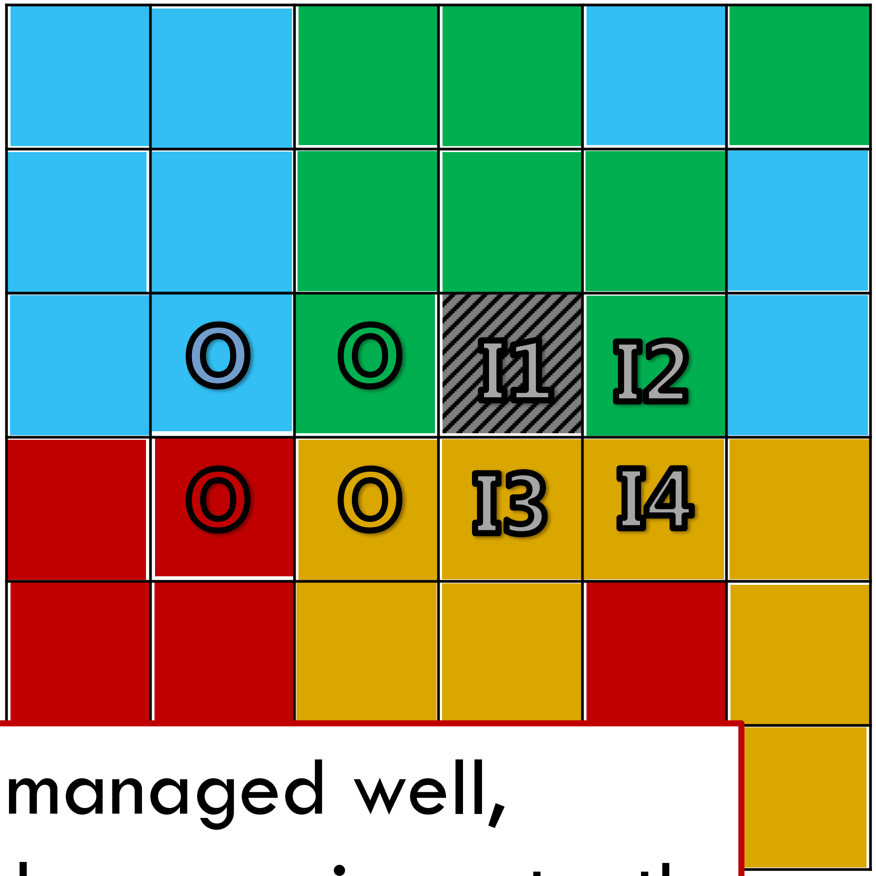
Mix:

4 x 471.omnetpp

4-thread 360.ilbdc

D-NUCA	Partitioned NUCA	
✓	✓	Place data
✗	✓	Control capacity
✗	✗	

Jigsaw [Beckmann'13]



When capacity is managed well, thread placement becomes important!

CDCS

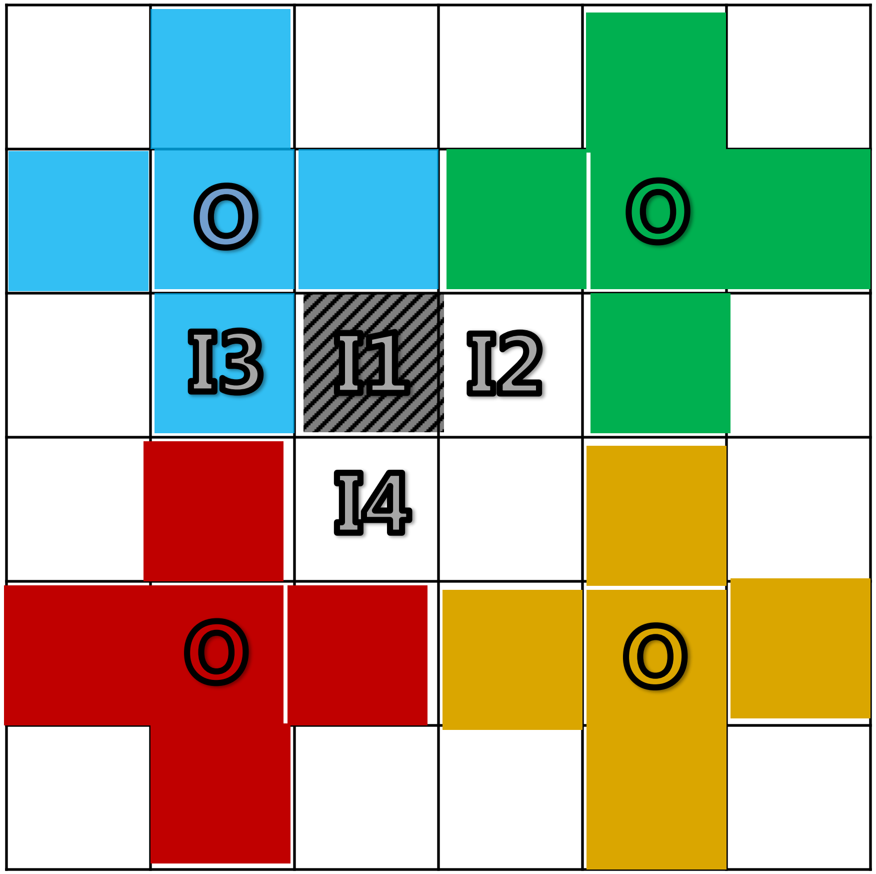
Mix:

4 x 471.omnetpp

4-thread 360.ilbdc

D-NUCA	Partitioned NUCA	**CDCS**	
✓	✓	✓	Place data
✗	✓	✓	Control capacity
✗	✗	✓	Place threads

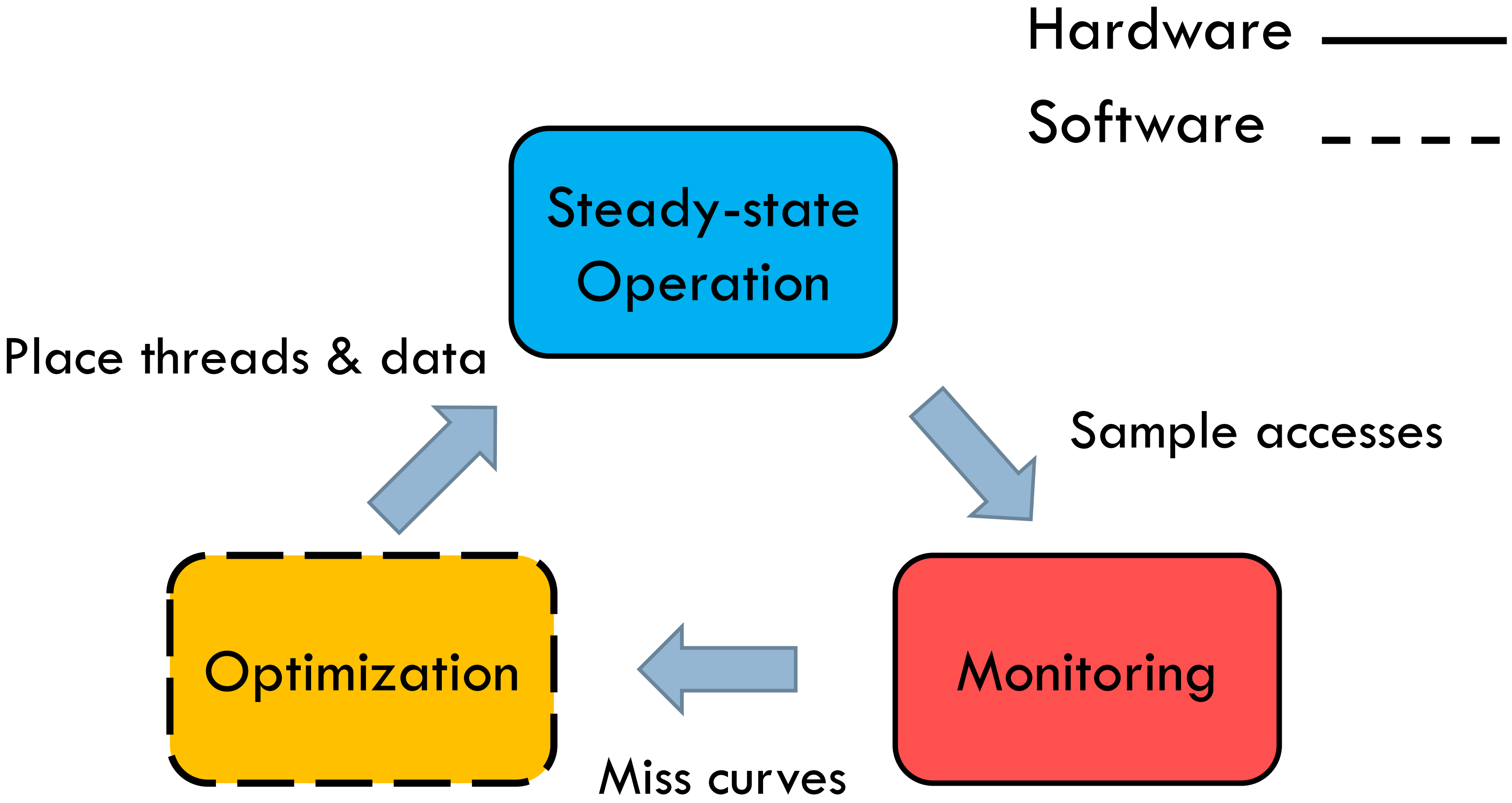
CDCS



Agenda

- Background
- CDCS Design
  - Operation
  - Optimization
- Evaluation

CDCS Overview



Hardware

Software

Steady-state Operation

Sample accesses

Monitoring

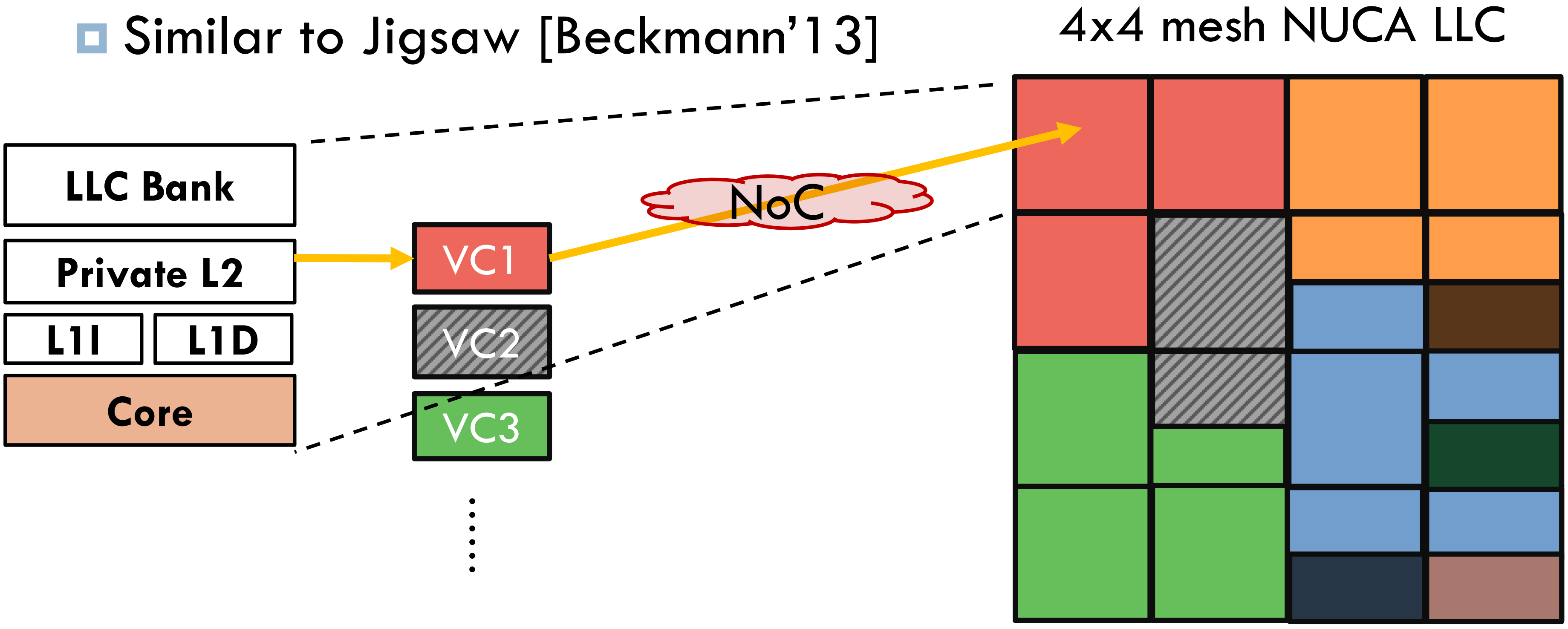
Miss curves

Optimization

Place threads & data

Operation

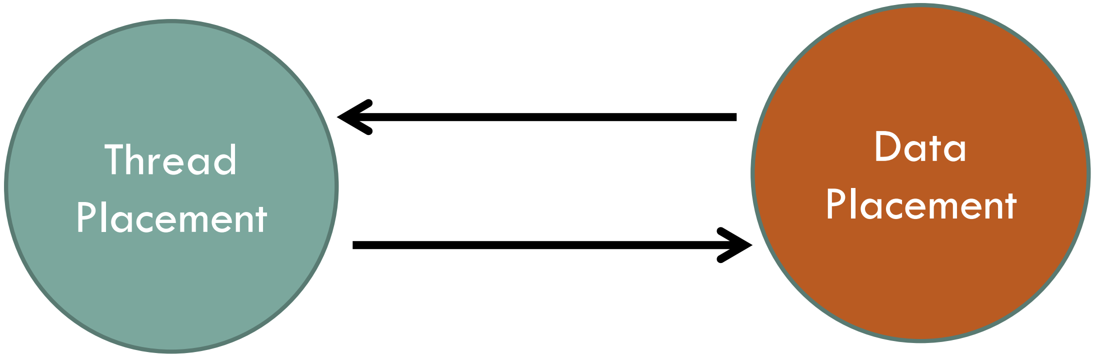
- Group partitions from different banks to create ***virtual caches (VCs)***
  - Similar to Jigsaw [Beckmann'13]



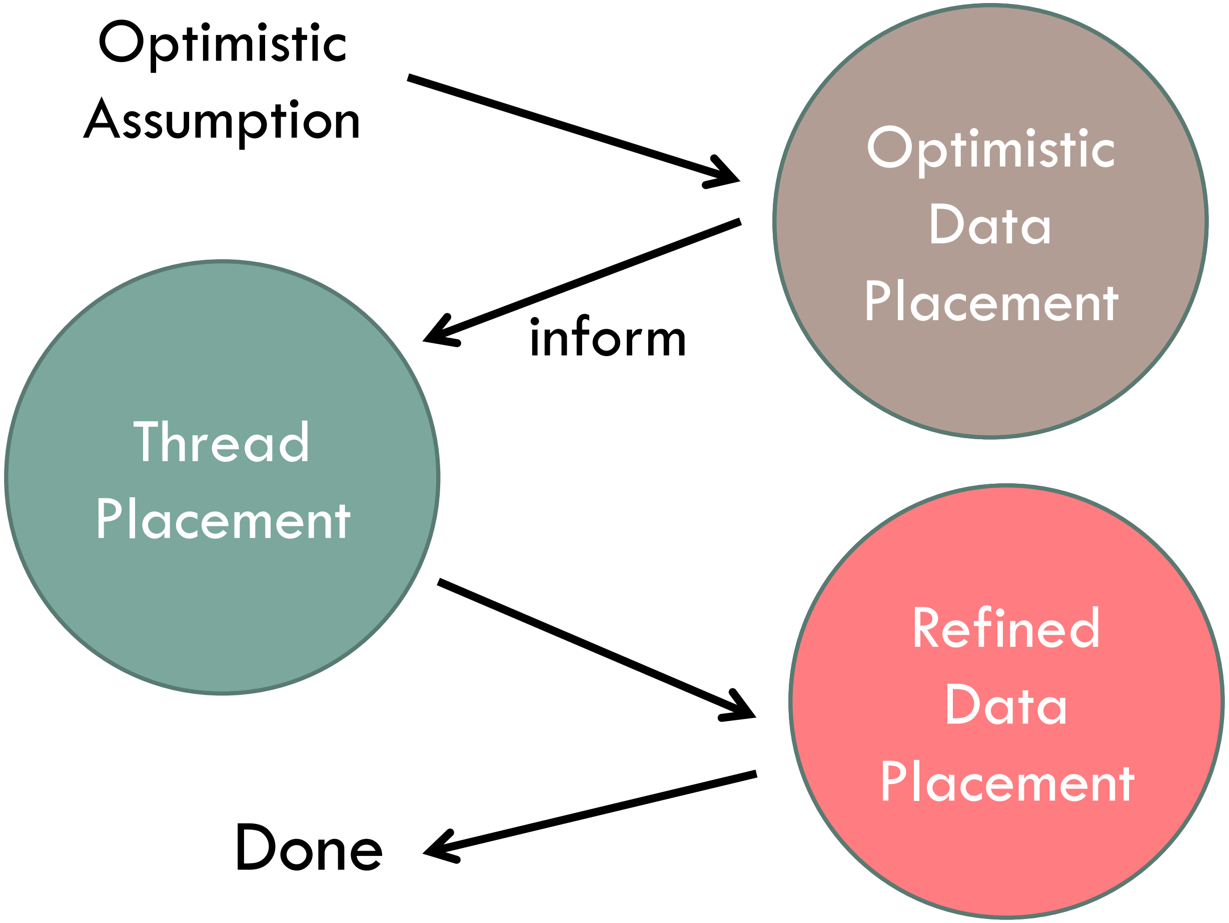
Optimization

- Minimize sum of on-chip latency and off-chip latency by deciding:
  - Thread placement
  - Virtual cache capacity
  - Virtual cache data placement

- It's an NP-hard problem
  - Thread and data placement are interrelated
  - Similar to VLSI place & route, HPC cluster scheduling



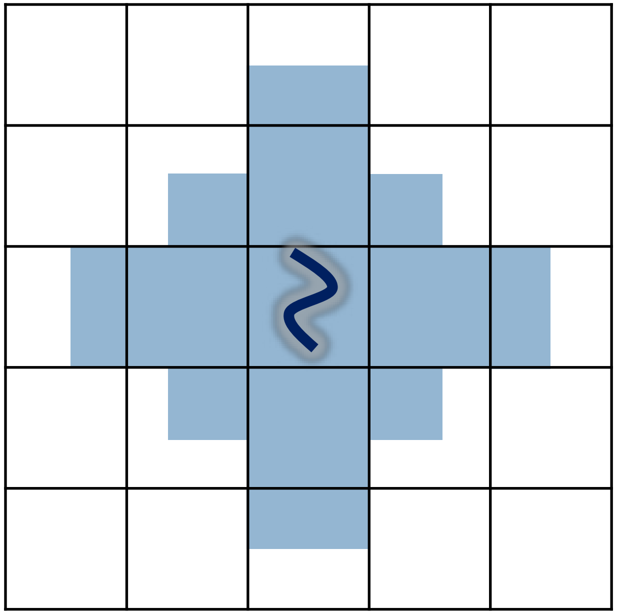
Insight: Decouple the dependency

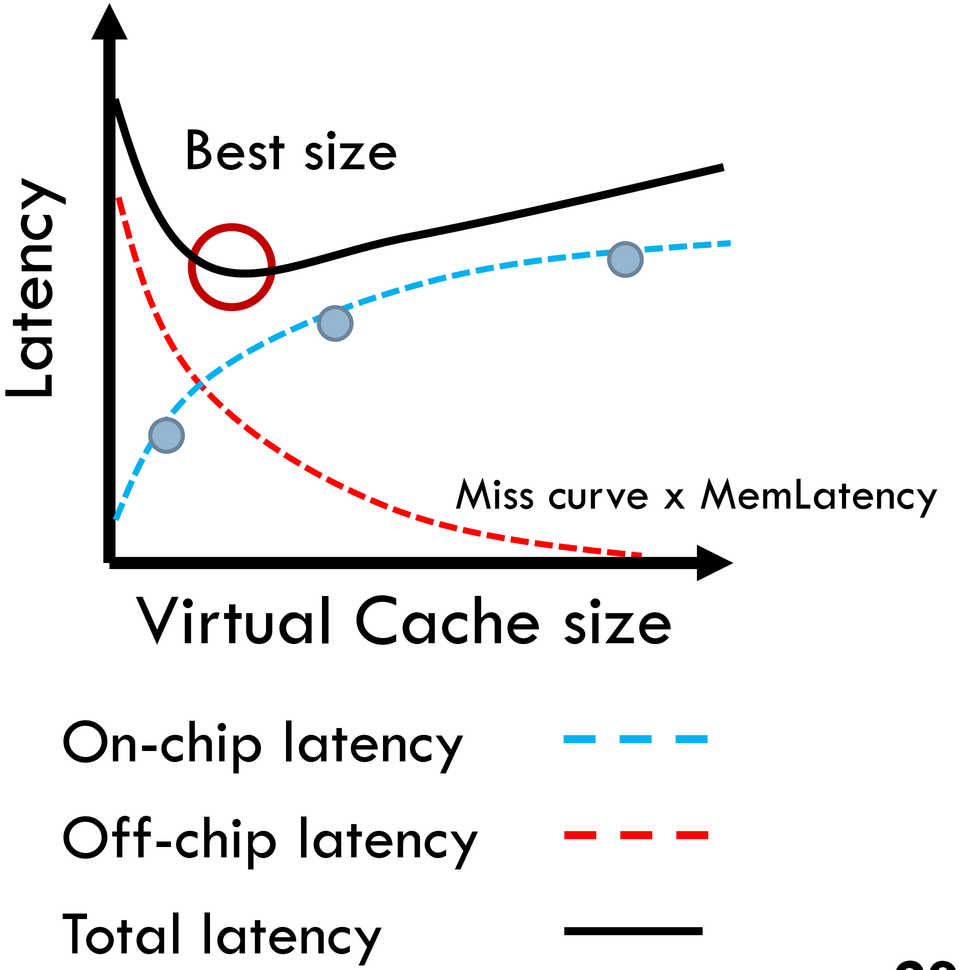


By placing data twice, CDCS disentangles the dependencies

Latency-aware allocation

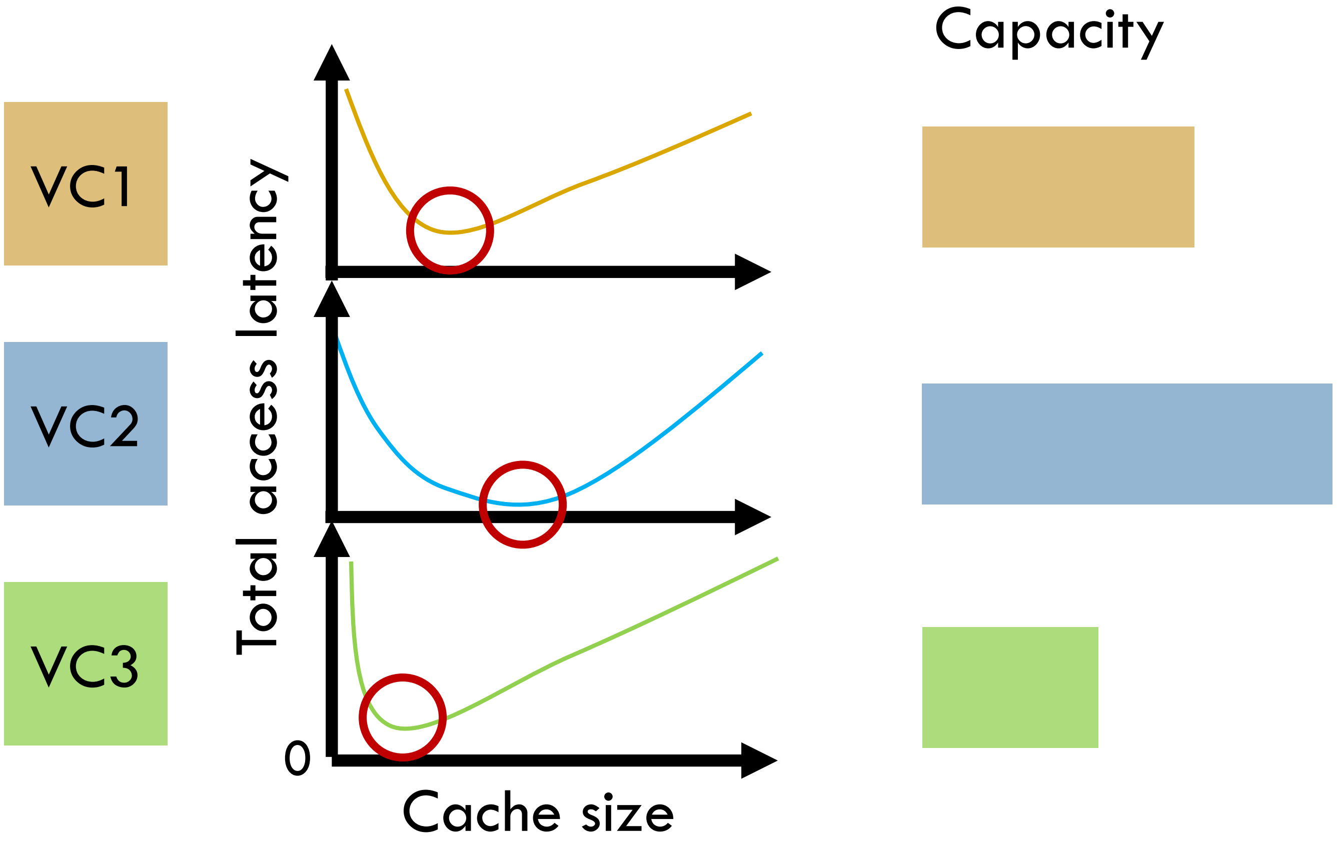
- Assume no contention





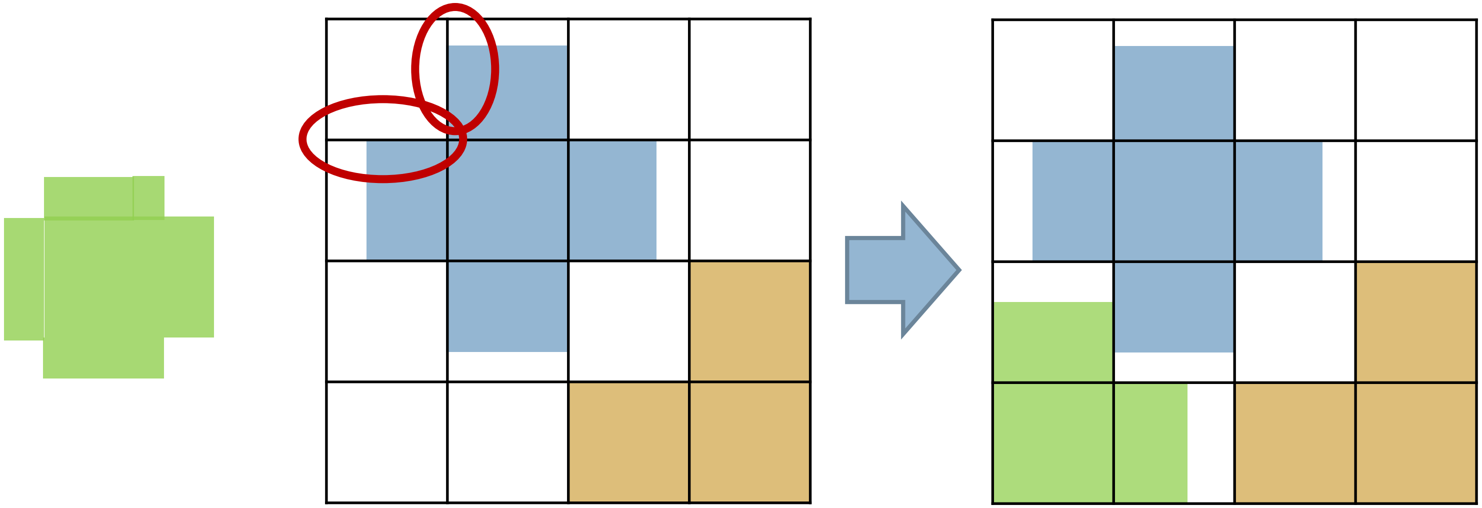
Latency-aware allocation

□ Use total latency curve to partition cache among VCs



Optimistic VC placement

- Place VC as compactly as possible

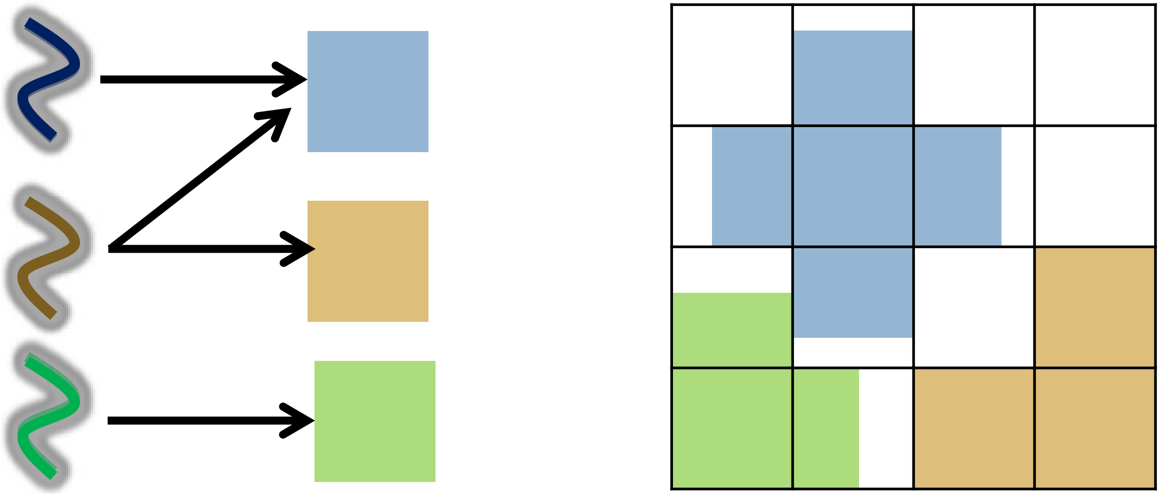


Estimating contention of every bank for VC

VC placed around least-contended tile

Thread placement

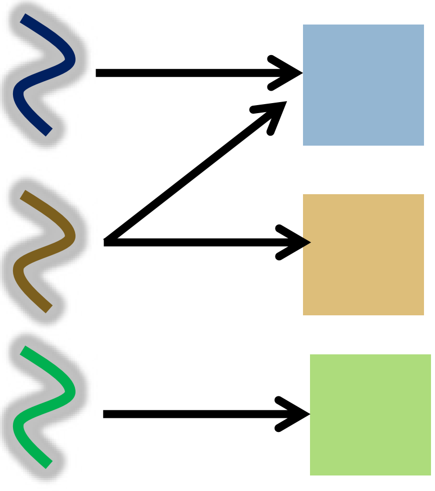
- □ Place threads at *center of mass of their accesses*

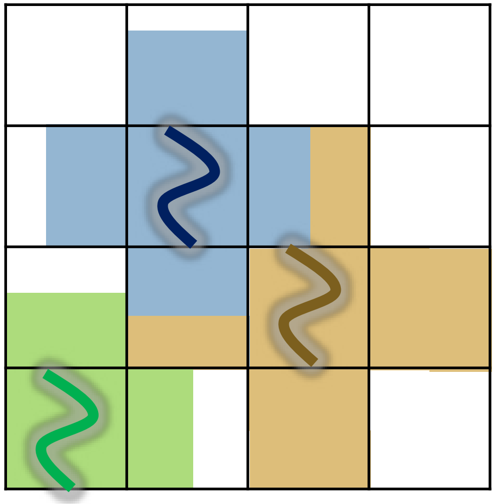


Refined VC placement

Greedily place VC close to thread first

Move/trade cache lines between VCs





Scalable reconfiguration & monitoring

- Incremental reconfiguration
  - Allows chip to reconfigure smoothly, without pausing cores

- Geometric monitor
  - Monitors large LLC with low overhead

See paper for details

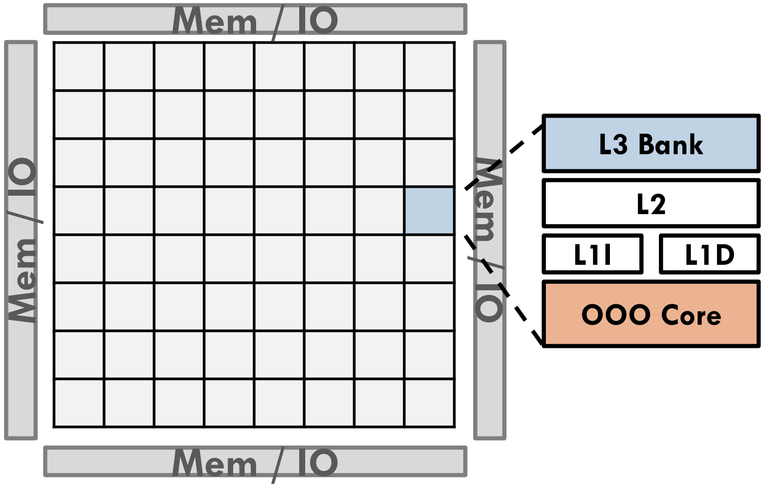
Agenda

- Background
- CDCS design
- Evaluation
  - Methodology
  - Performance
  - Sensitivity

Methodology

- Systems:
  - 64-core, 512KB/L3 bank
  - OOO cores (Silvermont-like)
  - 8x8 Mesh network
  - Similar to Knights Landing

64-core; 8x8 mesh network



- Zsim [Sanchez'13]: Pin-based simulator

- Workloads: SPEC CPU2006, SPEC OMP2012

Methodology

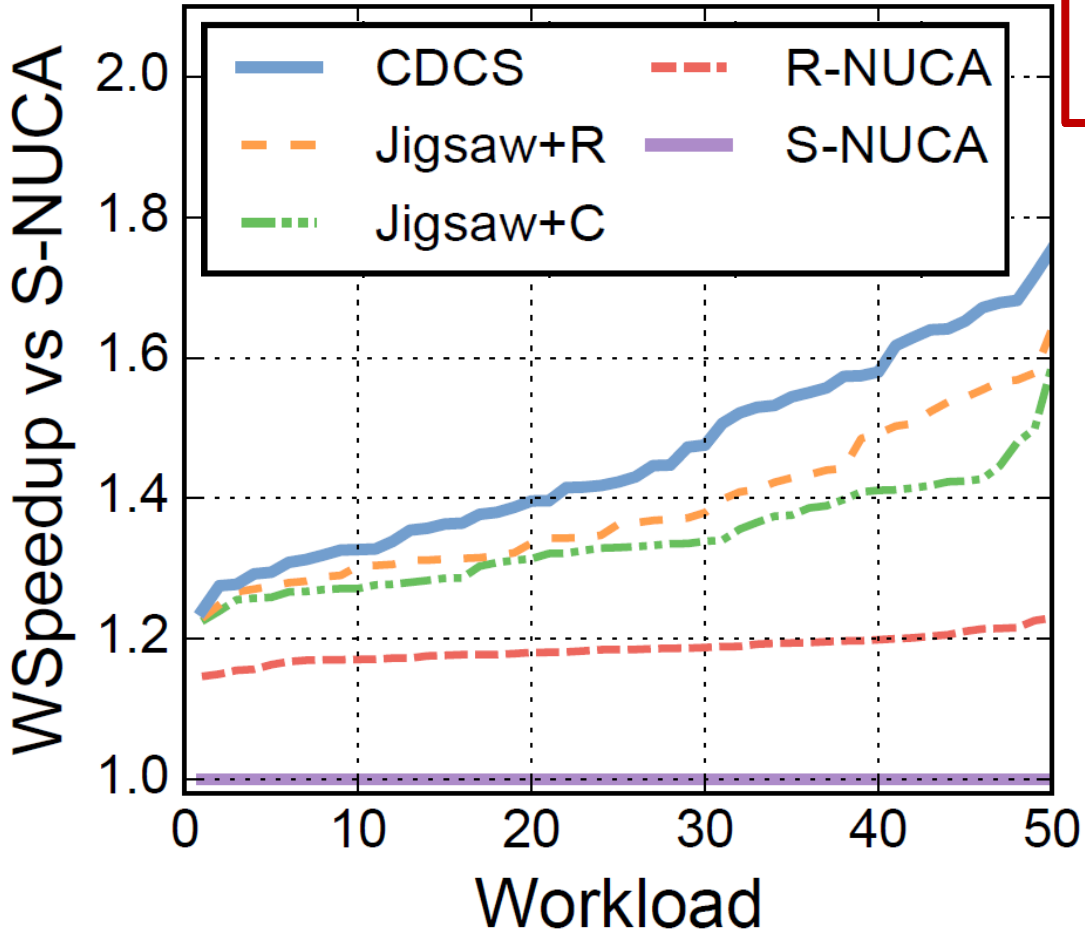
- Schemes
  - S-NUCA (baseline) with clustering thread scheduler
  - R-NUCA with clustering thread scheduler
  - Jigsaw
    - Jigsaw+C: Jigsaw with clustering thread scheduler
    - Jigsaw+R: Jigsaw with random thread scheduler
  - CDCS

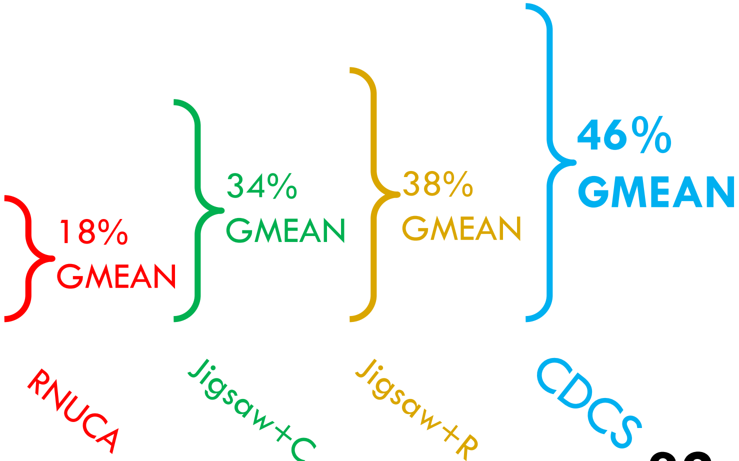
D-NUCA	Partitioned NUCA	**CDCS**	
✓	✓	✓	Place data
✗	✓	✓	Control capacity
✗	✗	✓	Place threads

Multi-programmed mixes

Workloads that do not share data

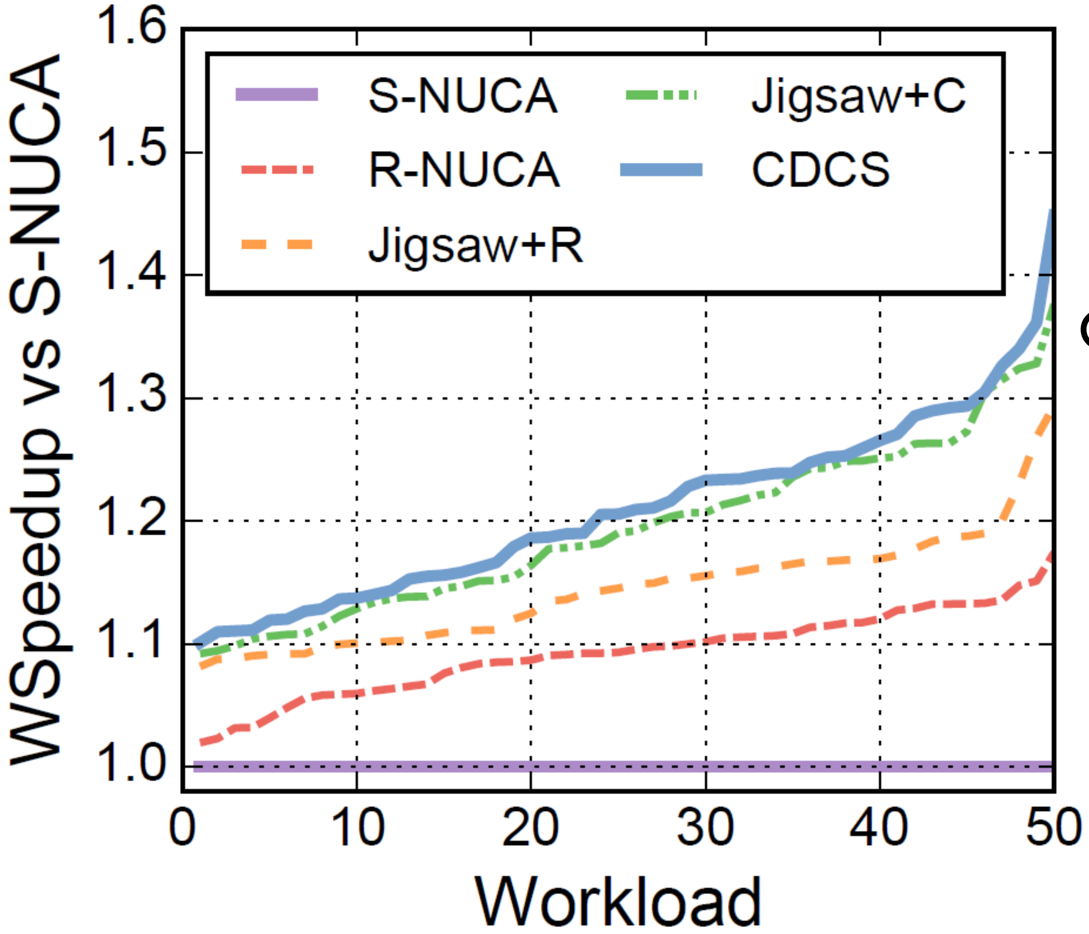
CDCS avoids capacity contention more effectively than random scheduler





Multi-threaded mixes

Workloads that share data



CDCS guards against **pathological behavior** incurred by fixed thread scheduling policies

Clustering is better now

9% GMEAN

19% GMEAN

14% GMEAN

21% GMEAN

RNUCA

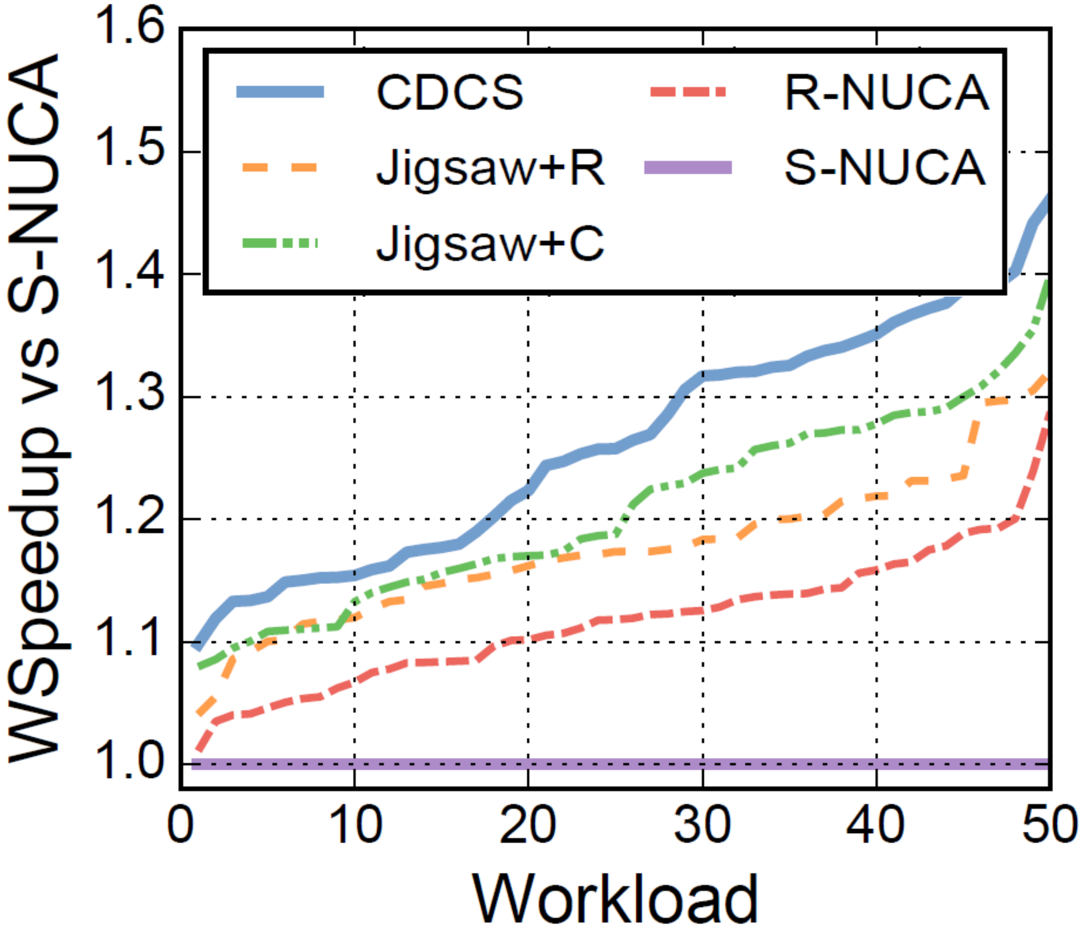
Jigsaw+C

Jigsaw+R

CDCS

Undercommitted multi-threaded mixes

- SPECOMP mixes using half of the cores



With more flexibility, CDCS dynamically clusters or spreads out threads

11% GMEAN

RNUCA

17% GMEAN

Jigsaw+R

21% GMEAN

Jigsaw+C

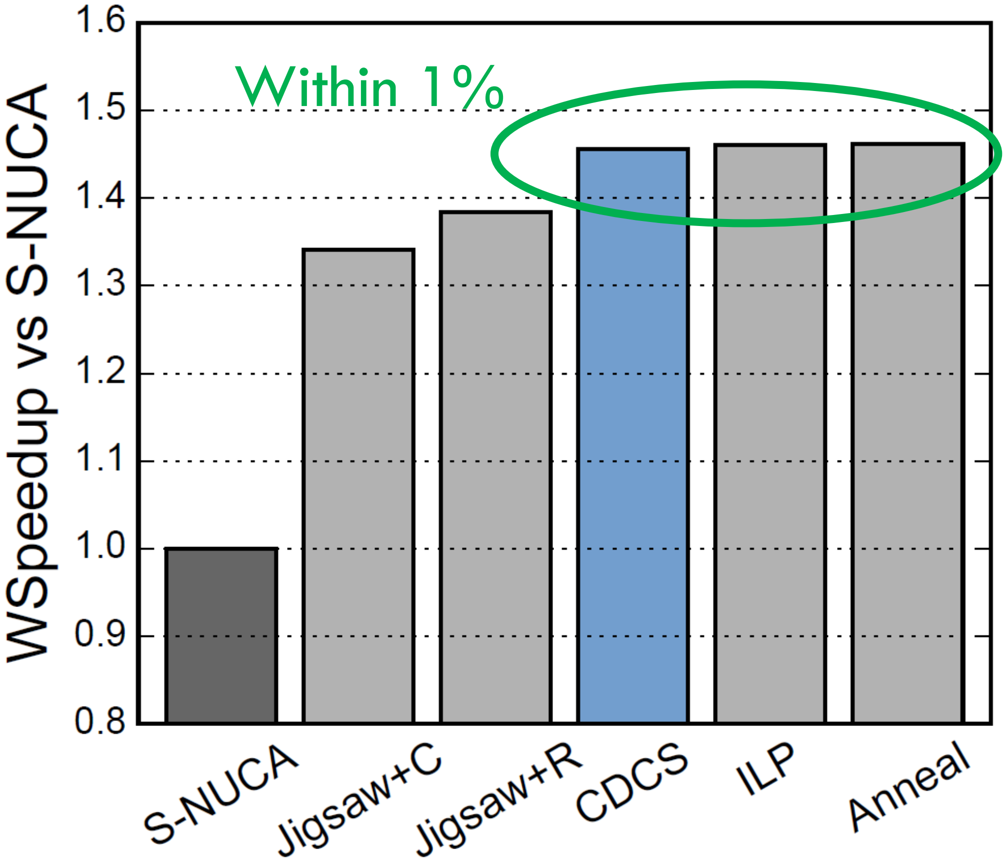
26% GMEAN

CDCS

CDCS vs idealized algorithms

- □ Integer Linear Programming (ILP)
- □ Simulated annealing

Algorithm runtime overhead (%)					
0	0.08	0.08	0.2	6.8	196



Multi-programmed mixes

Same for multi-threaded mixes

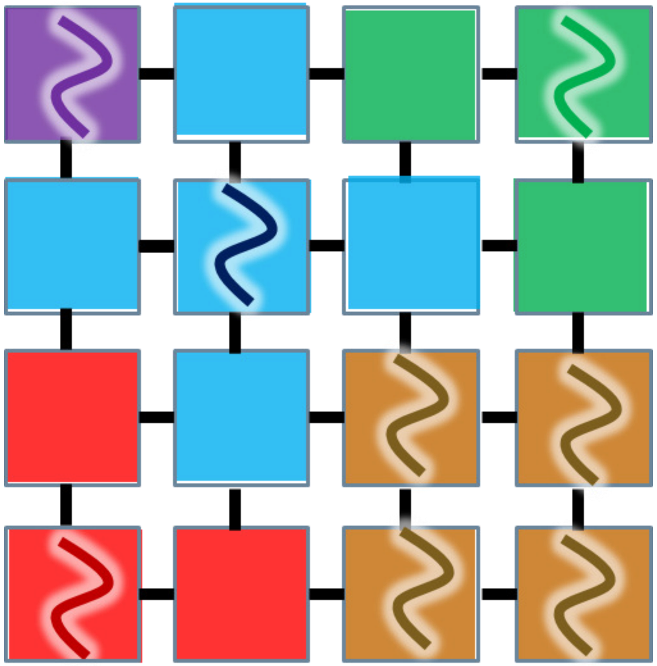
See paper for additional results

- Under-committed system
- Traffic breakdown
- Energy breakdown
- Factor analysis
- Other sensitivity studies
  - Reconfiguration interval sweep
  - Incremental reconfiguration IPC trace

Conclusions

- □ Thread placement has a large impact on NUCA performance when capacity is well managed
- □ CDCS reduces the distance to data through joint thread and data placement
- □ CDCS outperforms state-of-the-art NUCA techniques with different thread scheduling policies and prevents pathological behavior of fixed policies

QUESTIONS



Massachusetts Institute of Technology

CSAIL