香港科技大学(广州) THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

intel

UNIVERSITY OF CALIFORNIA SAN DIEGO

香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Push Multicast: A Speculative and Coherent Interconnect for Mitigating Manycore CPU Communication Bottleneck

Jiayi Huang, Yanhua Chen, Zhe Wang

Christopher J. Hughes, Yufei Ding, Yuan Xie

CPUs Growing in Core Count

Year 2023
Ampere One (192 Cores)

Year 2021
Yitian 710 (128 Cores)

Year 2019
Kunpeng 920 (64 Cores)

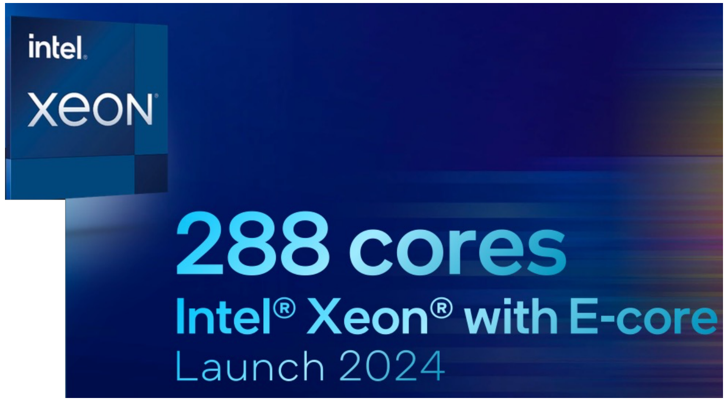
CPUs Growing in Core Count

Year 2019
Kunpeng 920 (64 Cores)

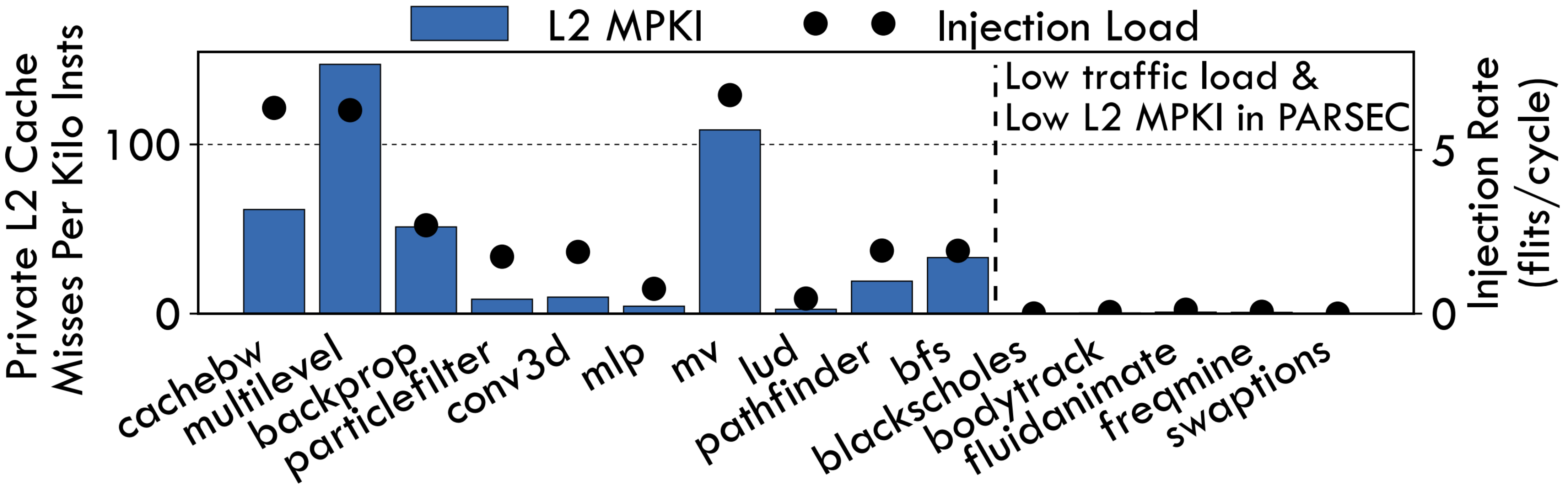
Year 2021
Yitian 710 (128 Cores)

Year 2023
Ampere One (192 Cores)

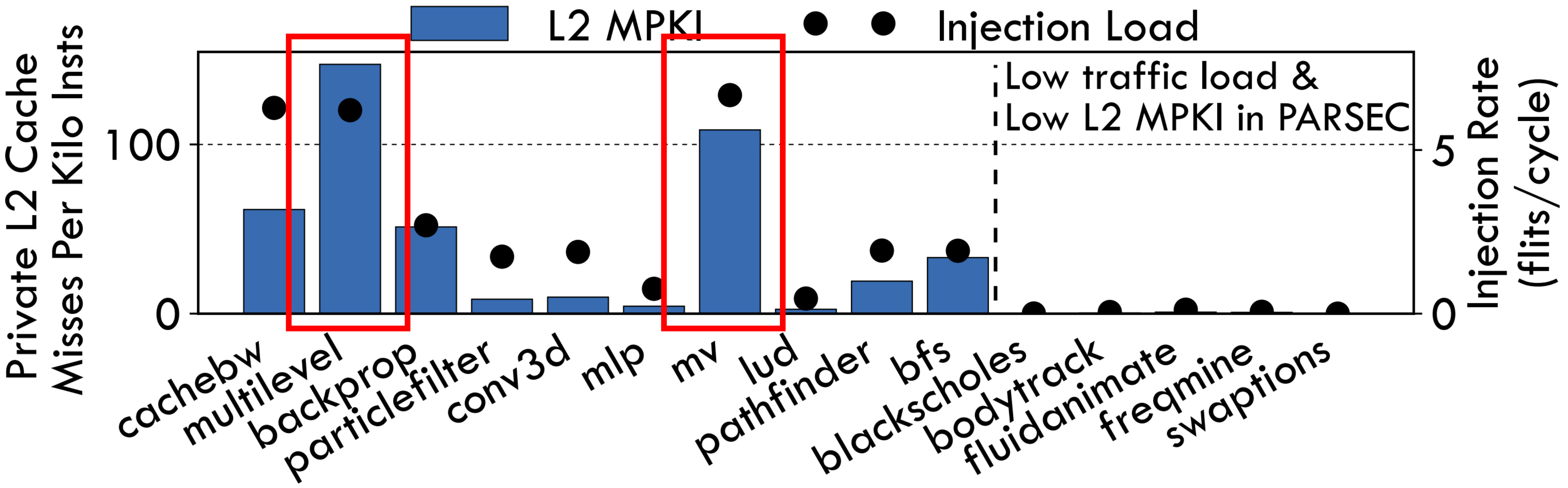
Year 2025 (To be released)
Intel Xeon CPU (288 Cores)



Too large working set: Bandwidth pressure

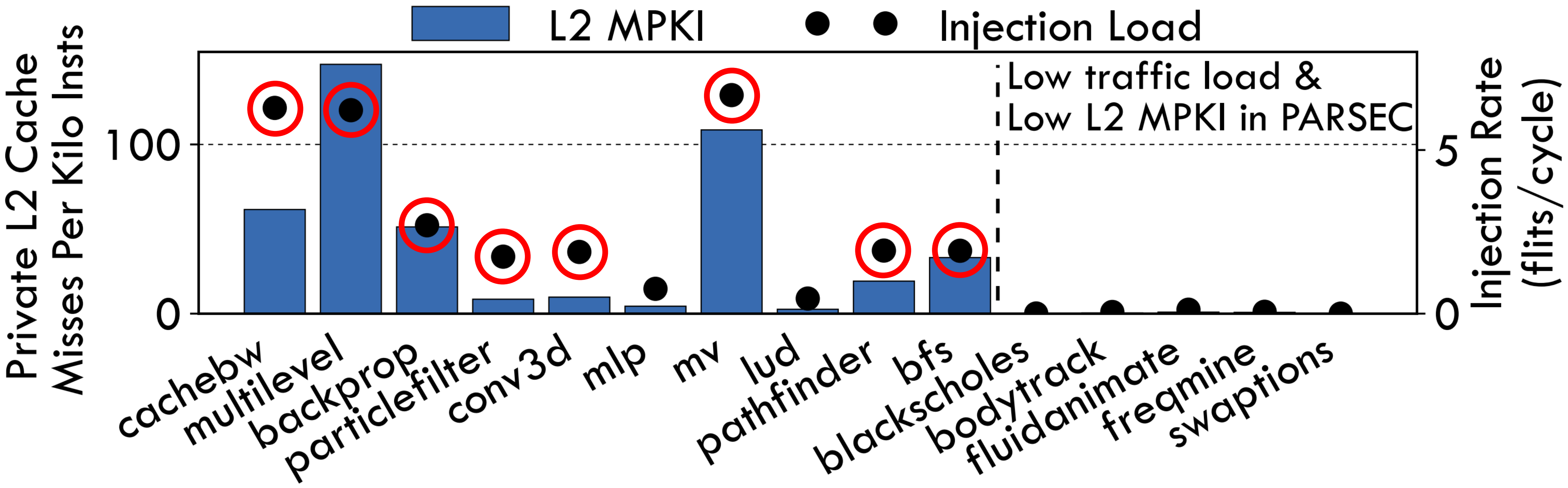


Too large working set: Bandwidth pressure



High L2 private cache miss rate: MPKI can reach 100

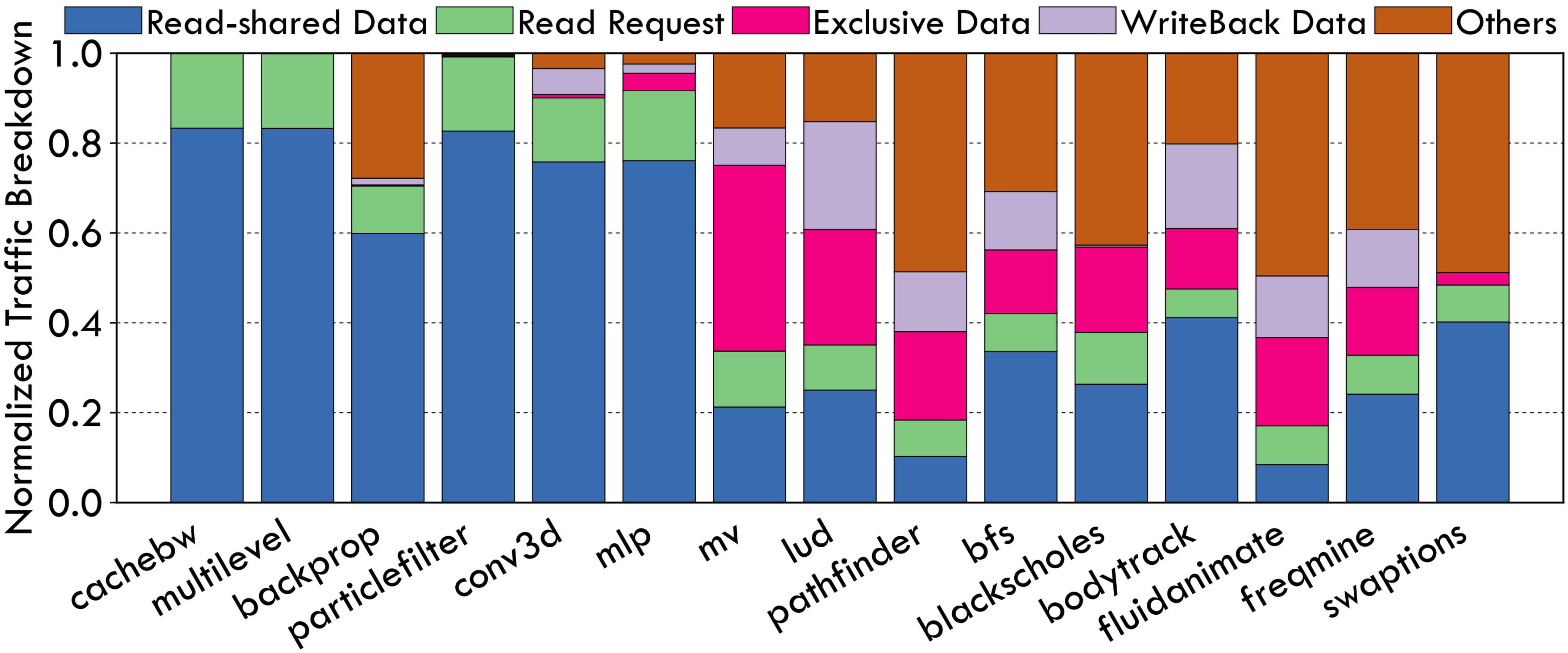
Too large working set: Bandwidth pressure



High L2 private cache miss rate: MPKI can reach 100

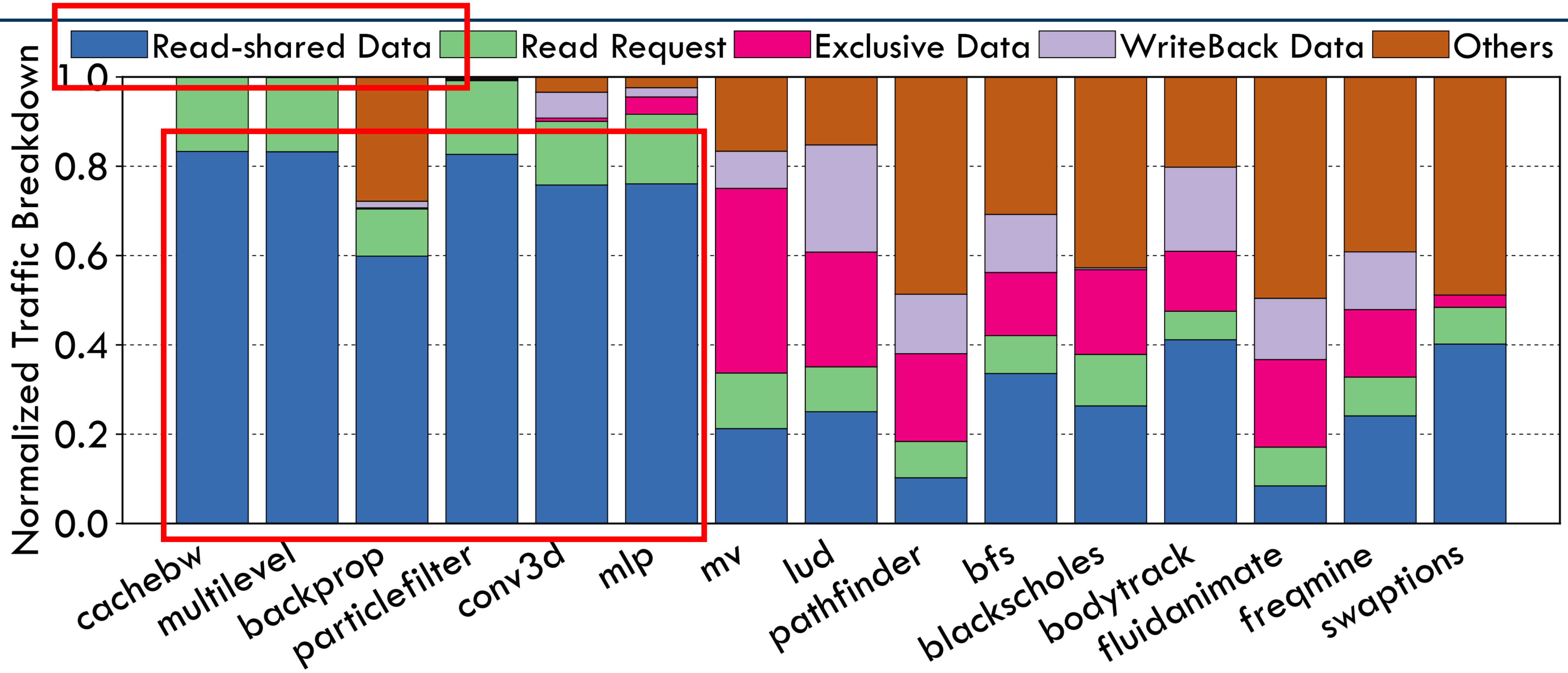
High traffic load on NoC and LLC accesses (dots)

Traffic Characterization Analysis



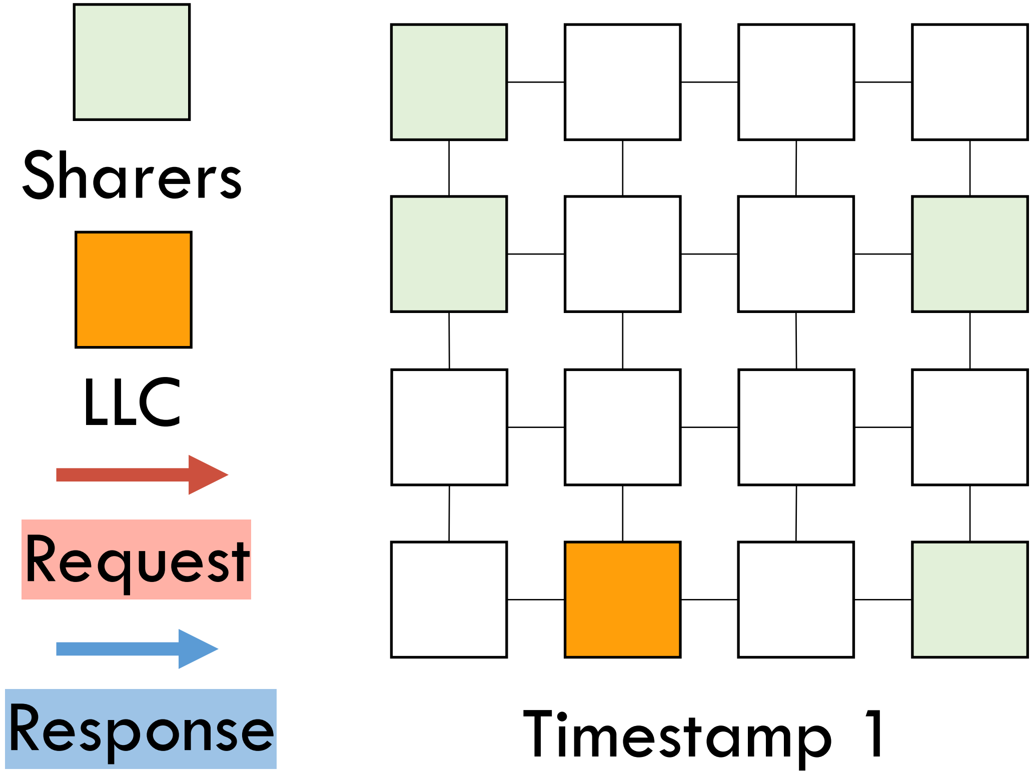
Traffic Characterization Analysis

Traffic Characterization Analysis



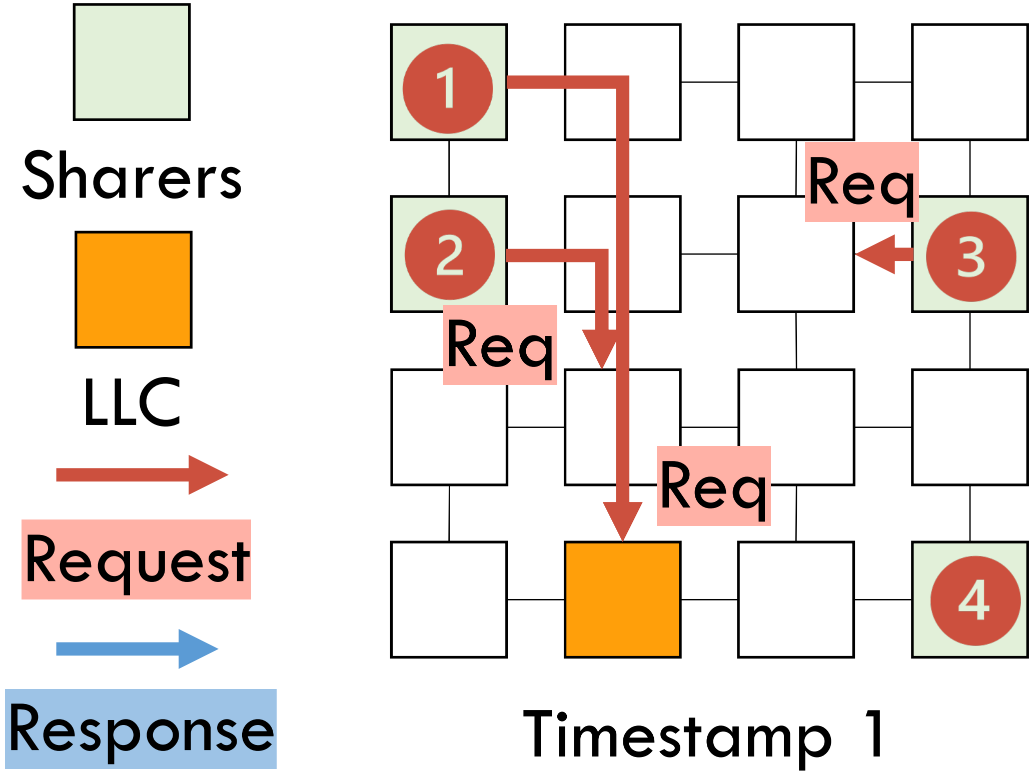
Read-shared data account for 10%-80% traffic

Bandwidth pressure: Read-shared data access

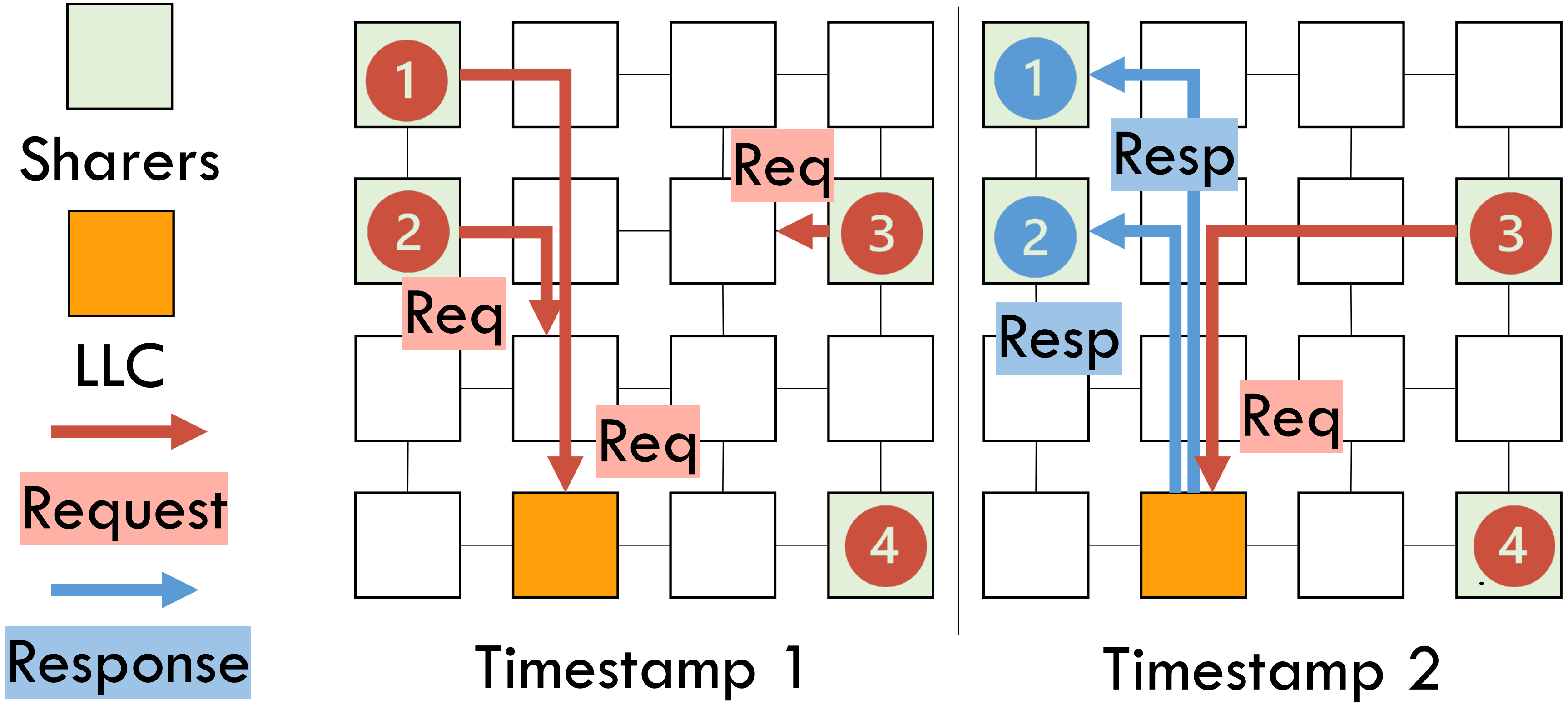


Timestamp 1

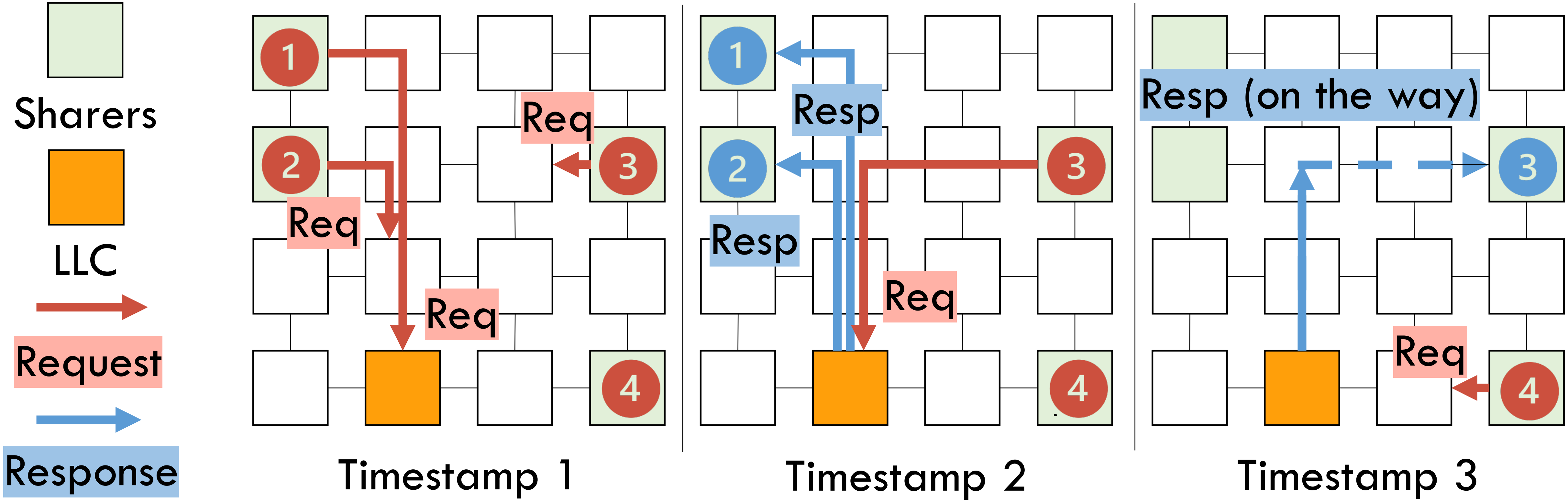
Bandwidth pressure: Read-shared data access



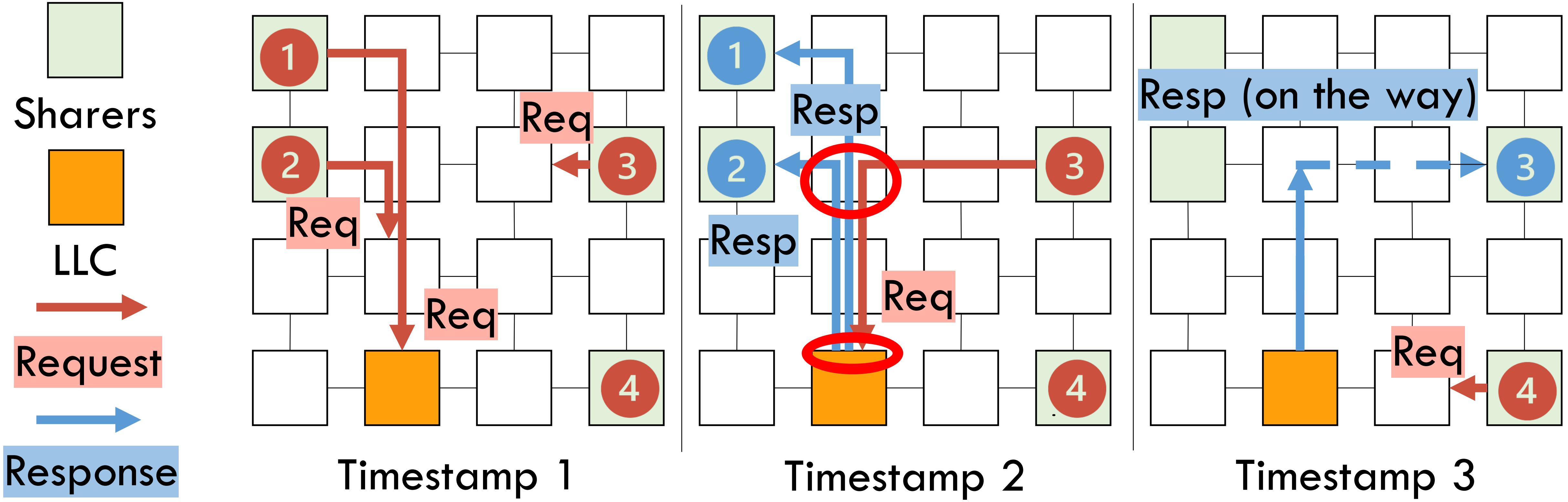
Bandwidth pressure: Read-shared data access



Bandwidth pressure: Read-shared data access

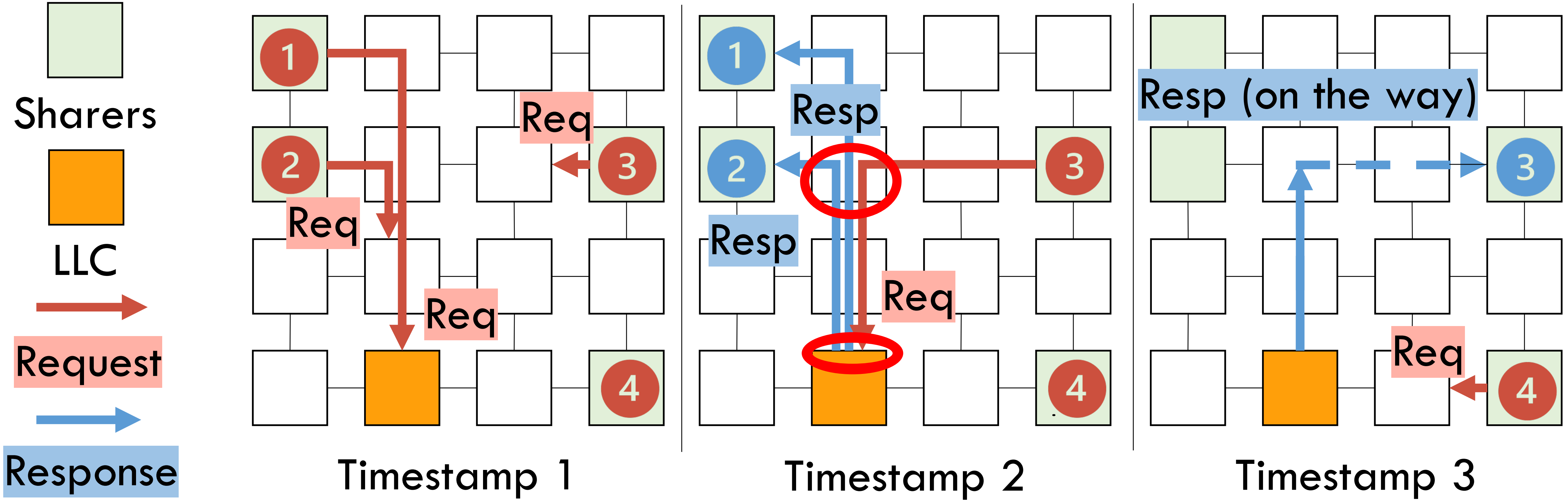


Bandwidth pressure: Read-shared data access



Redundant data request/response: High bandwidth pressure on NoC and LLC

Bandwidth pressure: Read-shared data access



<u>Redundant</u> data request/response: High bandwidth pressure on NoC and LLC

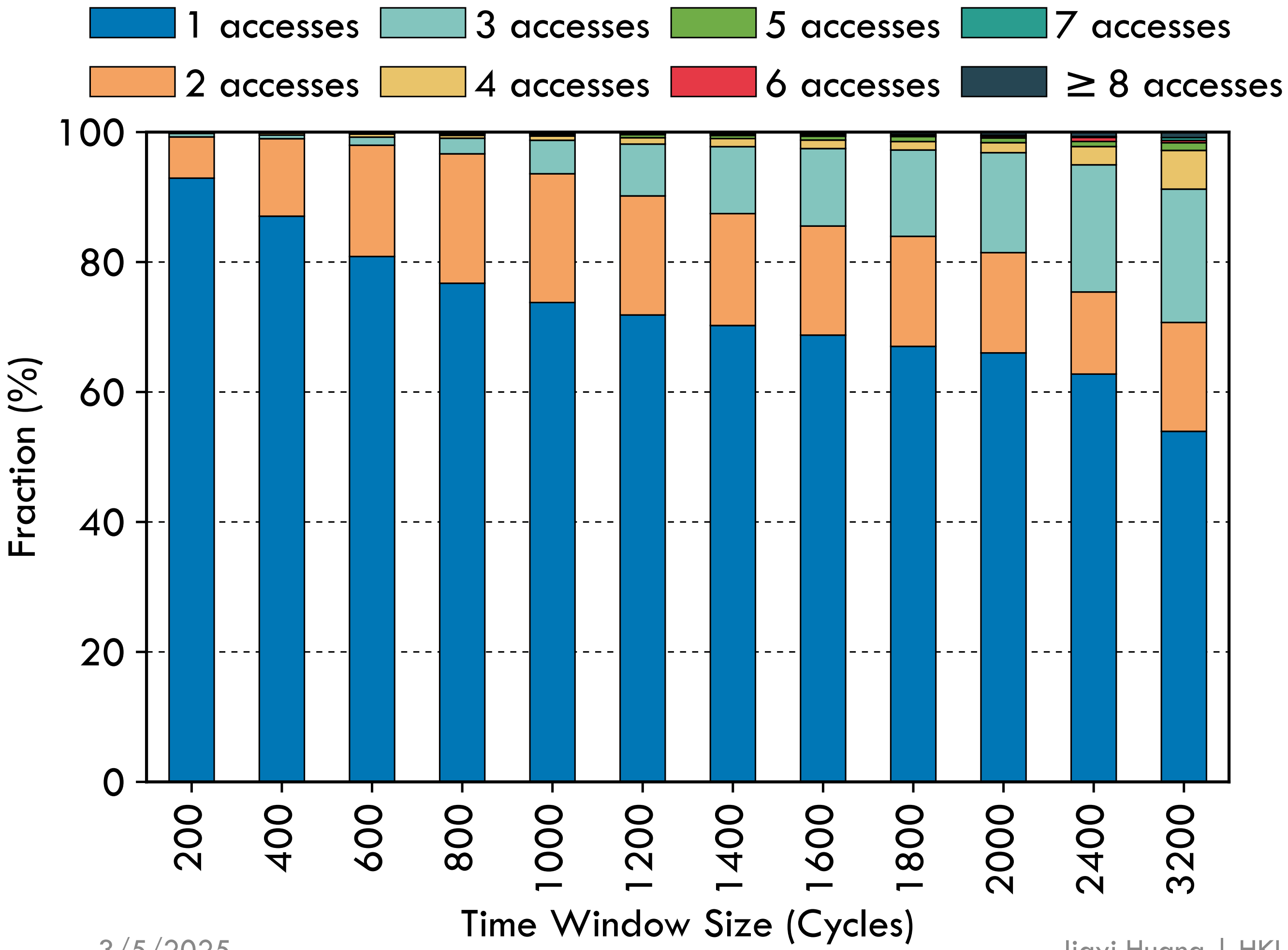
Can request **<u>coalescing</u>** and response **<u>multicasting</u>** mitigate the BW pressure?

Read-shared data access pattern of cachebw

- cachebw microbenchmark: 16 threads load a large array

Read-shared data access pattern of cachebw

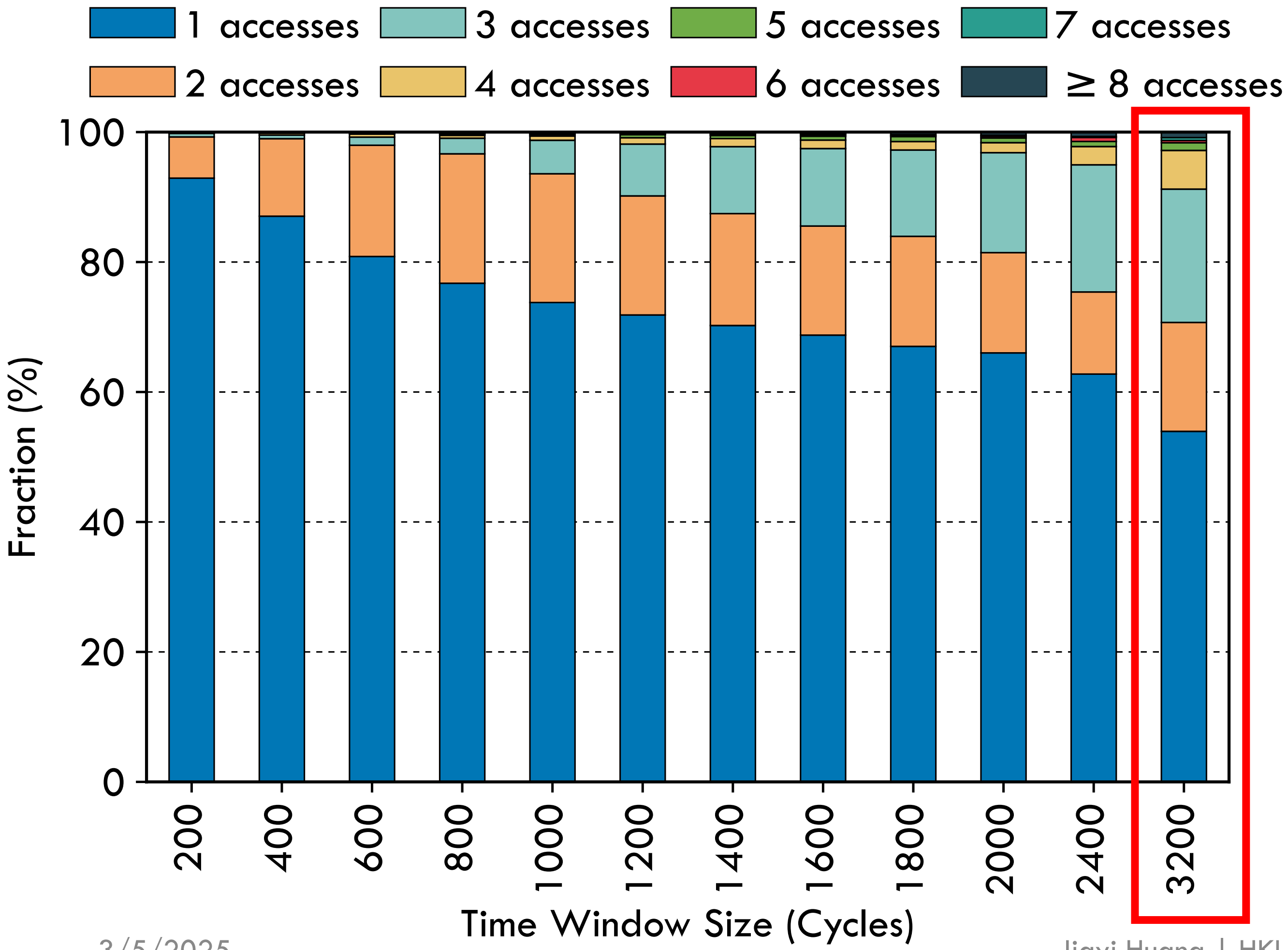
- cachebw microbenchmark: 16 threads load a large array



- Number of read-shared accesses on shared cache lines within a time window
  - Time window: 200 to 3200 cycles

Read-shared data access pattern of cachebw

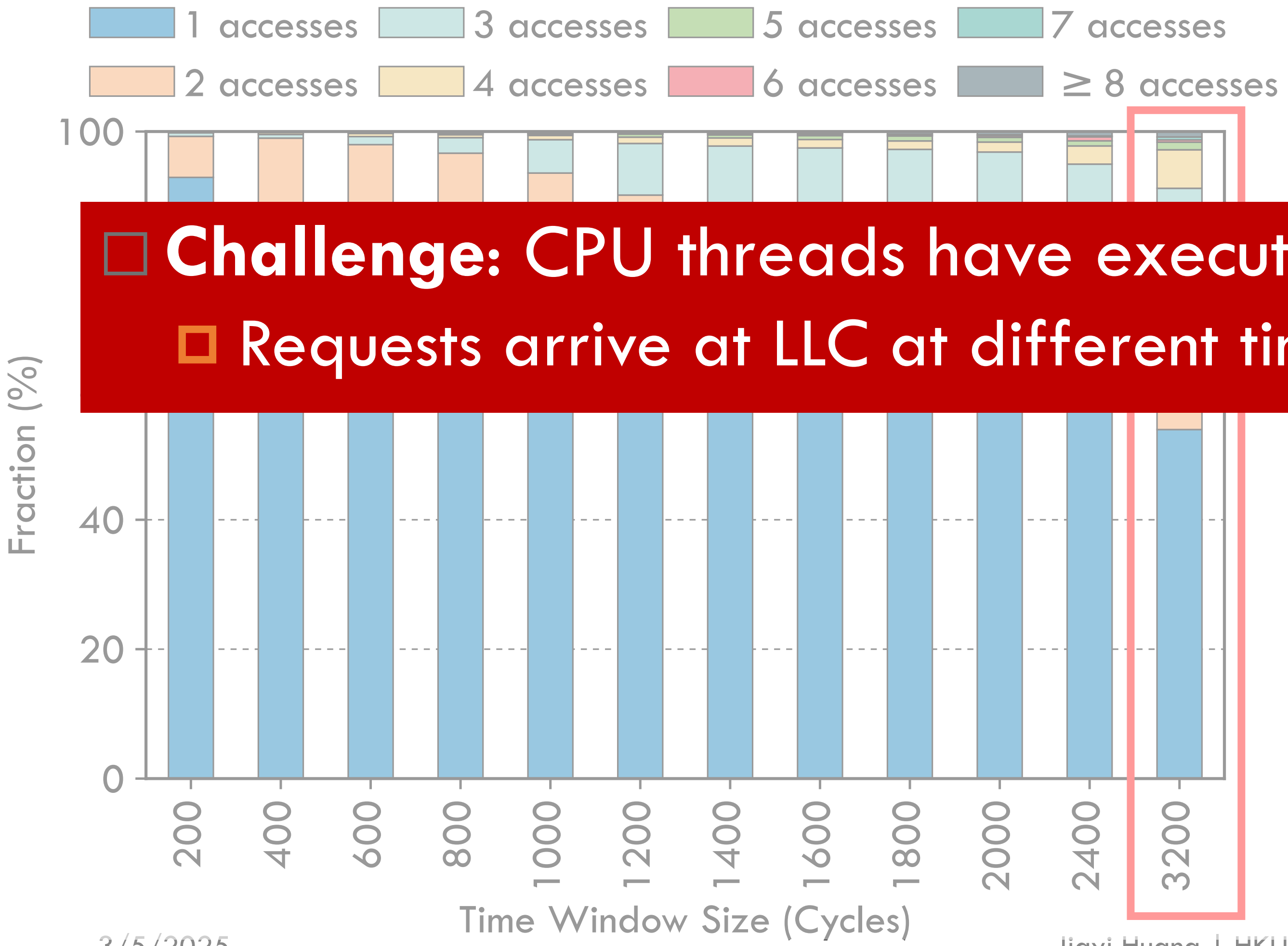
- cachebw microbenchmark: 16 threads load a large array



- Number of read-shared accesses on shared cache lines within a time window
  - Time window: 200 to 3200 cycles
- With a time window of 3200 cycles, most cache lines only observe 1-3 accesses
  - Typical LLC hit latency is 16-20 cycles

Read-shared data access pattern of cachebw

- cachebw microbenchmark: 16 threads load a large array



- Number of read-shared accesses on shared cache lines within a ti...

- **Challenge:** CPU threads have execution variations and NUCA effect
  - Requests arrive at LLC at different times, hard to coalesce for multicast

- With a time window of 3200 cycles, most cache lines only observe 1-3 accesses
  - Typical LLC hit latency is 16-20 cycles

Related Work

- Read-shared data accesses triggered by private L2 misses
  - Cold miss, conflict miss, **capacity miss** (our focus), coherence miss

Related Work

- Read-shared data accesses triggered by private L2 misses
  - Cold miss, conflict miss, **capacity miss** (our focus), coherence miss
- Prefetching
  - Bingo[Bakhshalipour+ HPCA'19], Berti[Navarro-Torres+ MICRO'22] , CLIP[Panda MICRO'23]
- Request coaleasing
  - NYU Ultracomputer [Gottlieb+ ISCA'98]
  - GPU packet coalescing [Kim+ ICS'17]
- Decouple access/execute: Stream Floating [Wang+ HPCA'21]
- Coherence prediction [Mukherjee and Hill ISCA'98] [Kaxiras and Young HPCA'00]
  - Memory sharing predictor [Lai and Falsafi ISCA'99]

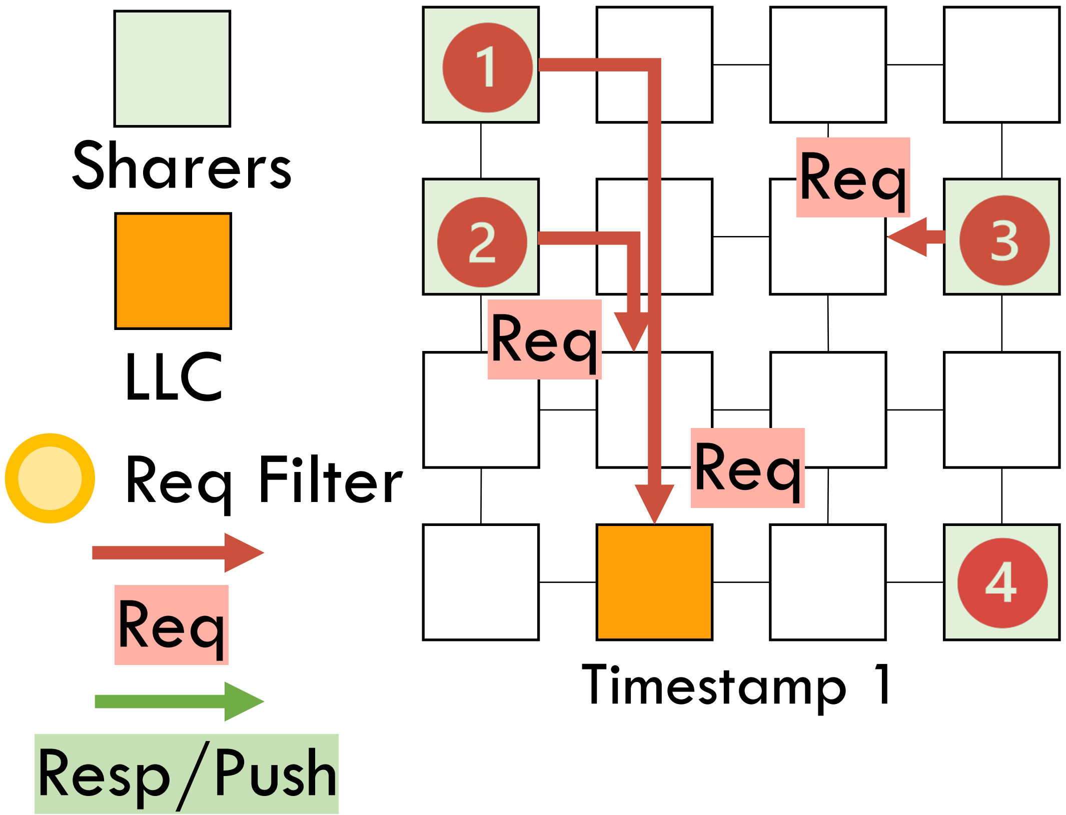
Related Work

- Read-shared data accesses triggered by private L2 misses
  - Cold miss, conflict miss, **capacity miss (our focus)**, coherence miss

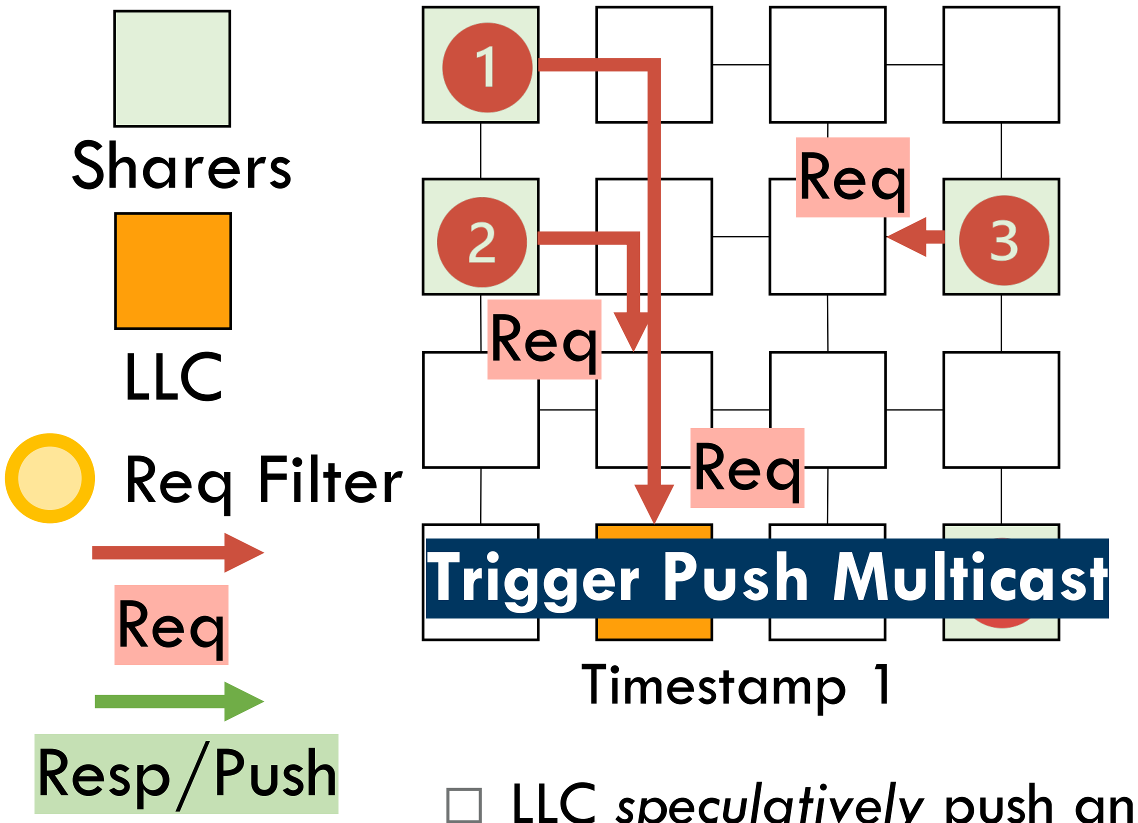
Prior work either focuses on per-core access *latency* or has limited multicast opportunity

- Prefetching
  - Bingo[Bakhshalipour+ HPCA'19], Berti[Navarro-Torres+ MICRO'22] , CLIP[Panda MICRO'23]
- Request coaleasing
  - NYU Ultracomputer [Gottlieb+ ISCA'98]
  - GPU packet coalescing [Kim+ ICS'17]
- Decouple access/execute: Stream Floating [Wang+ HPCA'21]
- Coherence prediction [Mukherjee and Hill ISCA'98] [Kaxiras and Young HPCA'00]
  - Memory sharing predictor [Lai and Falsafi ISCA'99]

Our Approach: Speculative Push Multicast

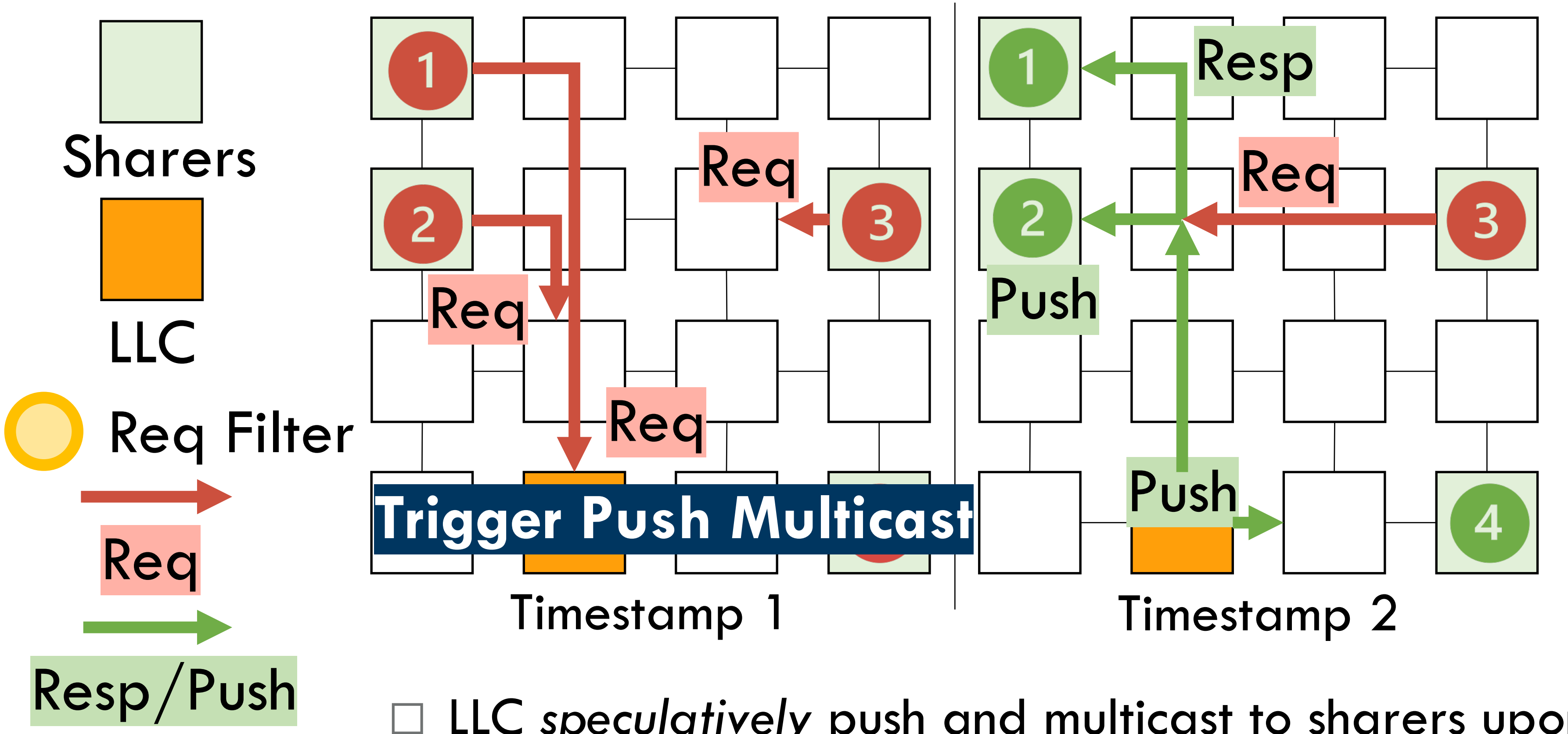


Our Approach: Speculative Push Multicast



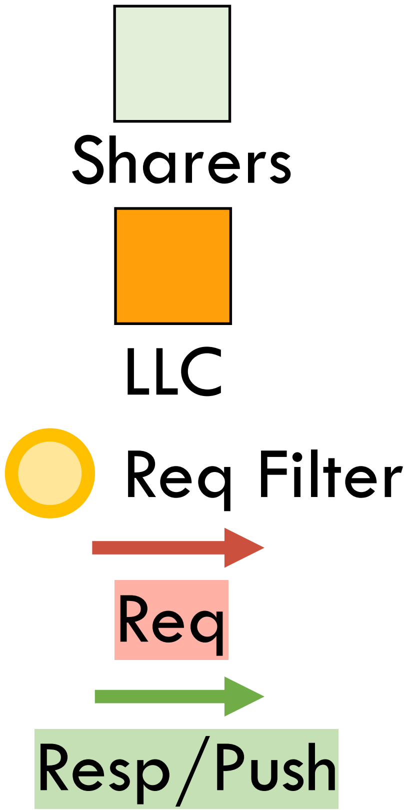
- LLC *speculatively* push and multicast to sharers upon receiving a request

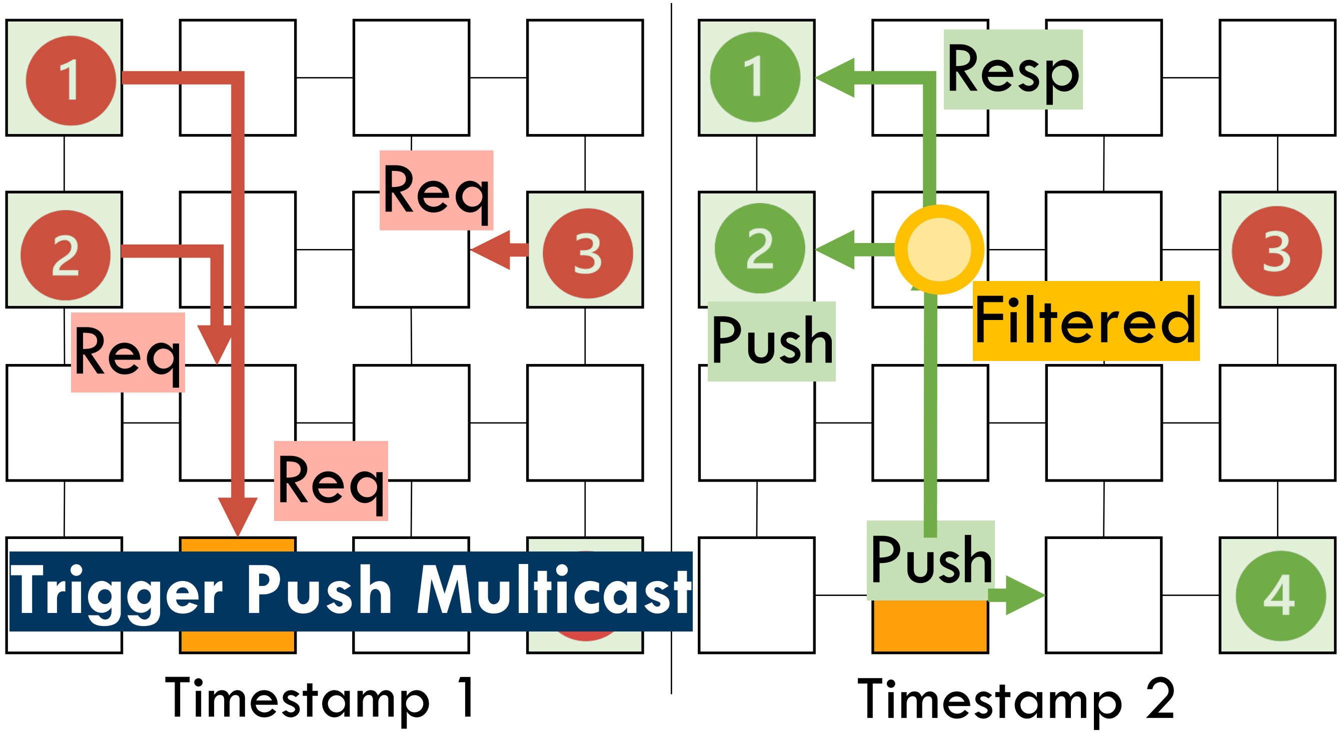
Our Approach: Speculative Push Multicast



- LLC *speculatively* push and multicast to sharers upon receiving a request

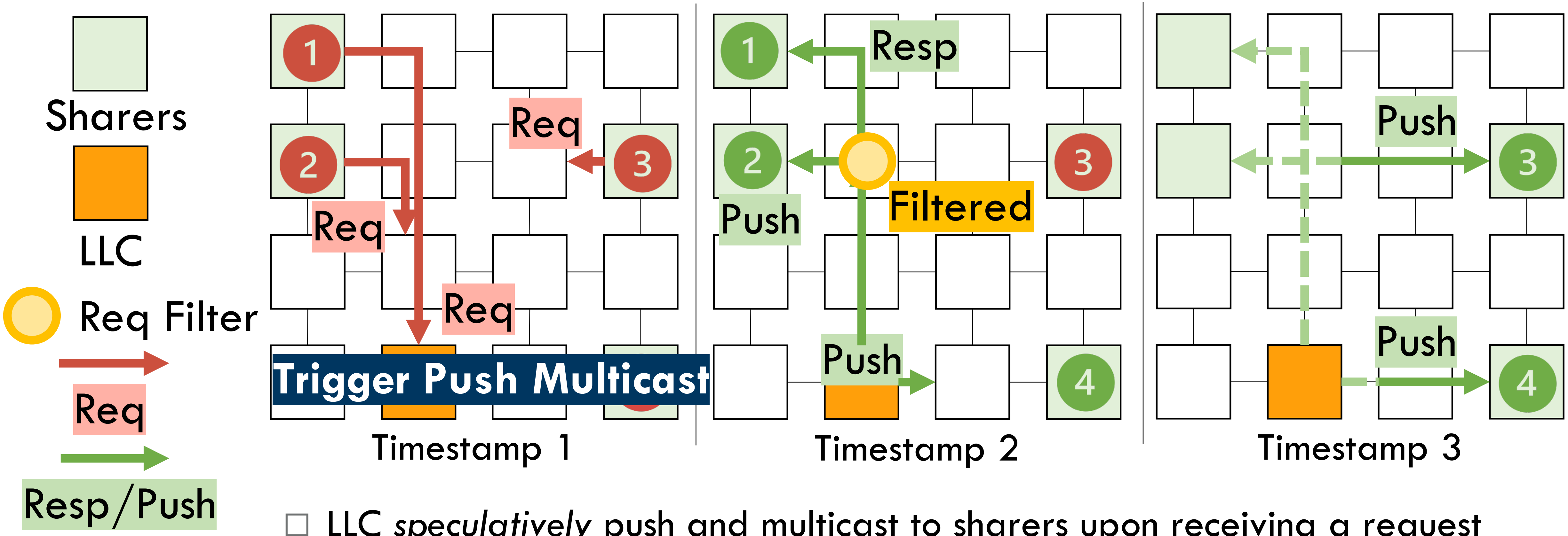
Our Approach: Speculative Push Multicast





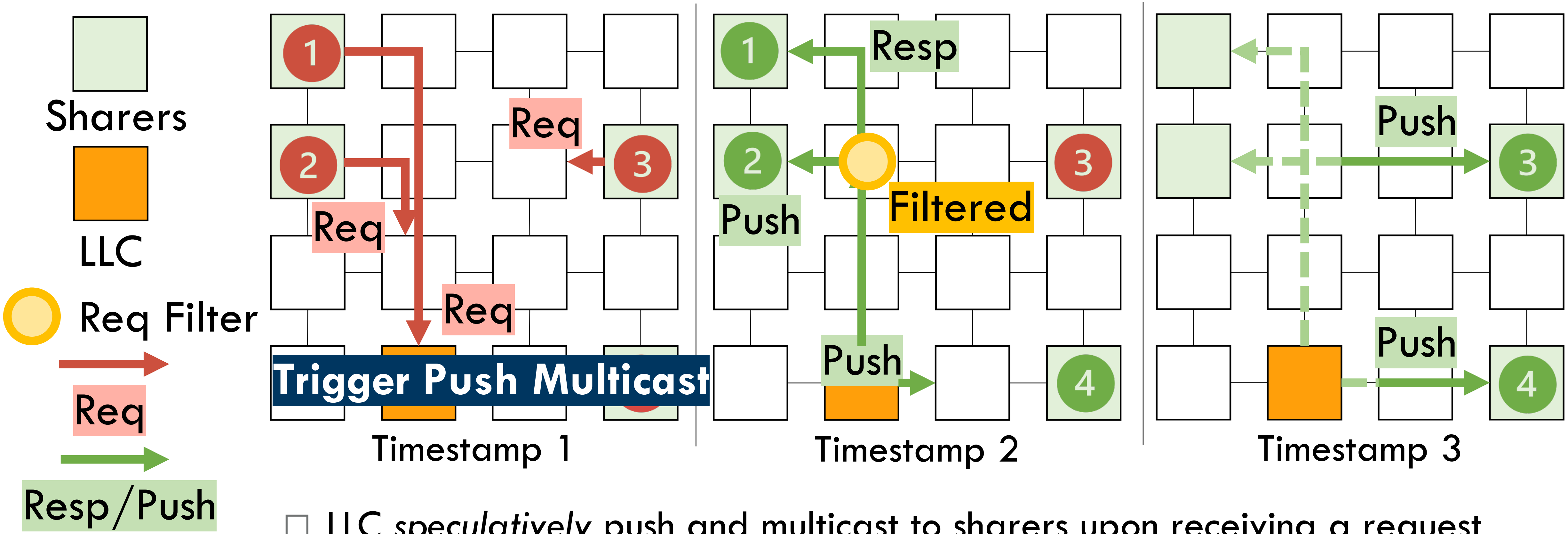
- ☐ LLC *speculatively* push and multicast to sharers upon receiving a request
- ☐ Coherent in-network filtering at router to prune redundant requests

Our Approach: Speculative Push Multicast



- ☐ LLC *speculatively* push and multicast to sharers upon receiving a request
- ☐ Coherent in-network filtering at router to prune redundant requests

Our Approach: Speculative Push Multicast



- LLC *speculatively* push and multicast to sharers upon receiving a request
- Coherent in-network filtering at router to prune redundant requests
- Dynamic pause and resume mechanism to turn push on/off

Implementation Overview

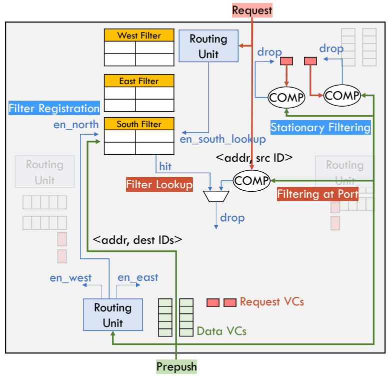
- LLC enhancements
  - Detect read-shared requests
  - Initiate multicast to known sharers

Implementation Overview

- LLC enhancements
  - Detect read-shared requests
  - Initiate multicast to known sharers

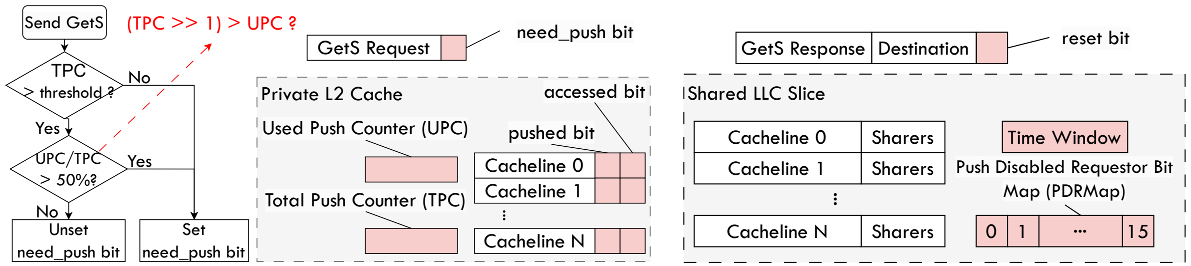
- NoC router enhancements
  - Filter redundant requests
  - Handle multicasts

	Stage 1	Stage 2
Read Request Flit	Buffer Write Route Compute Filtering at Port Filter Lookup	VC Allocation SW Allocation
Push Head Flit	Buffer Write Route Compute Filter Registration Stationary Filtering	VC Allocation SW Allocation Filter De-registration



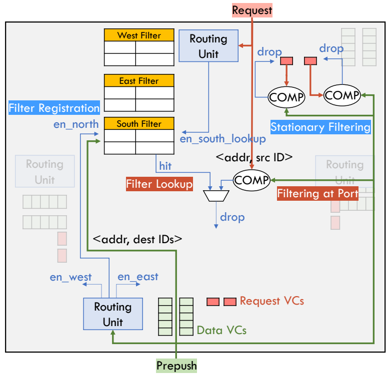
Implementation Overview

- LLC enhancements
  - Detect read-shared requests
  - Initiate multicast to known sharers
- Runtime knob to turn pushes on/off



- NoC router enhancements
  - Filter redundant requests
  - Handle multicasts

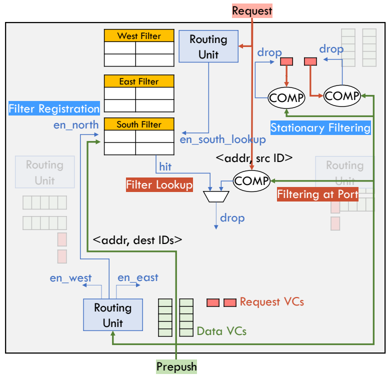
Read Request Flit	Buffer Write Route Compute	VC Allocation SW Allocation
	Filtering at Port Filter Lookup	
Push Head Flit	Buffer Write Route Compute	VC Allocation SW Allocation
	Filter Registration Stationary Filtering	Filter De-registration



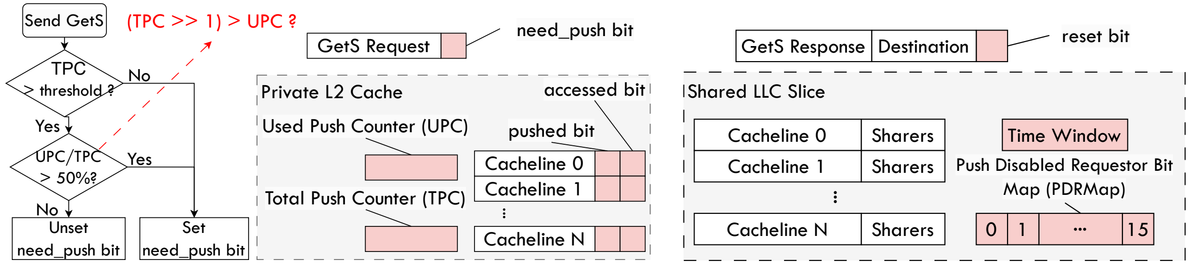
Implementation Overview

- LLC enhancements
  - Detect read-shared requests
  - Initiate multicast to known sharers
- NoC router enhancements
  - Filter redundant requests
  - Handle multicasts

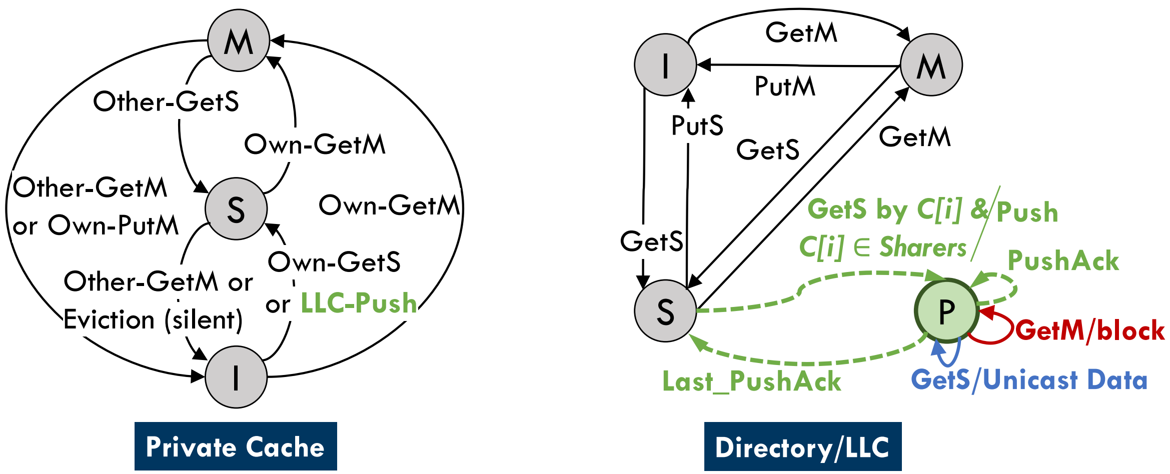
Read Request Flit	Buffer Write Route Compute Filtering at Port Filter Lookup	VC Allocation SW Allocation
Push Head Flit	Buffer Write Route Compute Filter Registration Stationary Filtering	VC Allocation SW Allocation Filter De-registration



- Runtime knob to turn pushes on/off

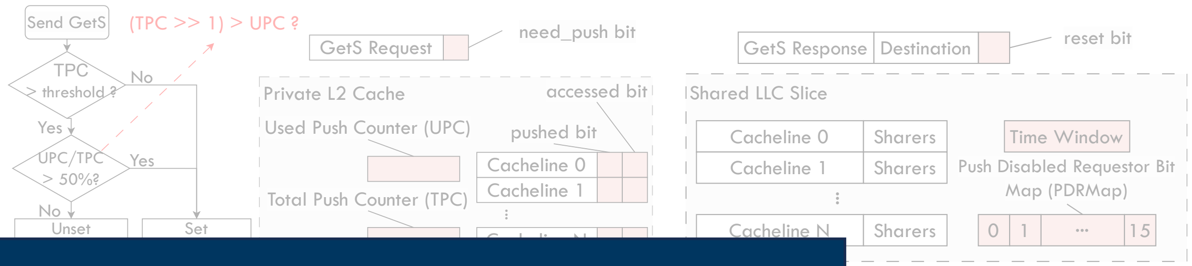


- Coherence extension



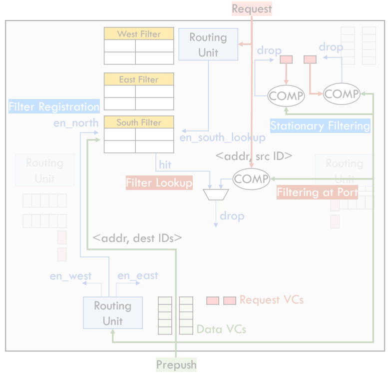
Implementation Overview

- LLC enhancements
  - Detect read-shared requests
  - Initiate multicast to known sharers
- Runtime knob to turn pushes on/off

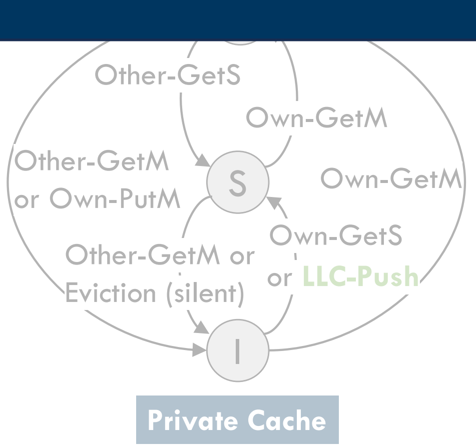


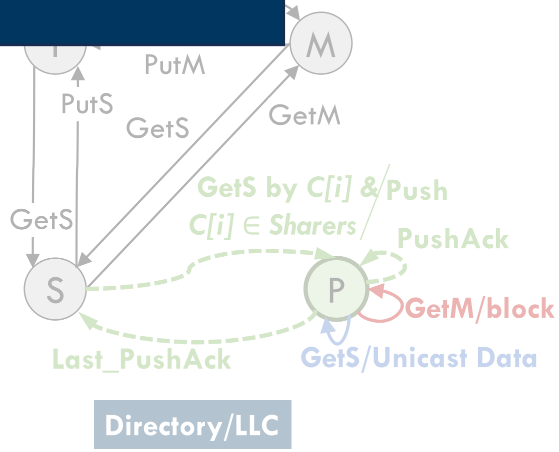
- NoC router enhancements
  - Filter redundant requests
  - Handle multicasts

Read Request Flit	Buffer Write Route Compute Filtering at Port Filter Lookup	VC Allocation SW Allocation
Push Head Flit	Buffer Write Route Compute Filter Registration Stationary Filtering	VC Allocation SW Allocation Filter De-registration



Details in the paper



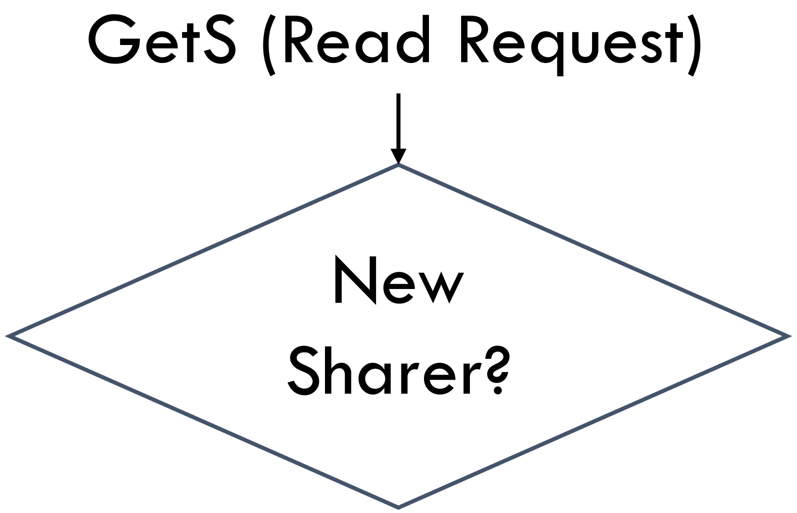


LLC Enhancements for Push Multicast

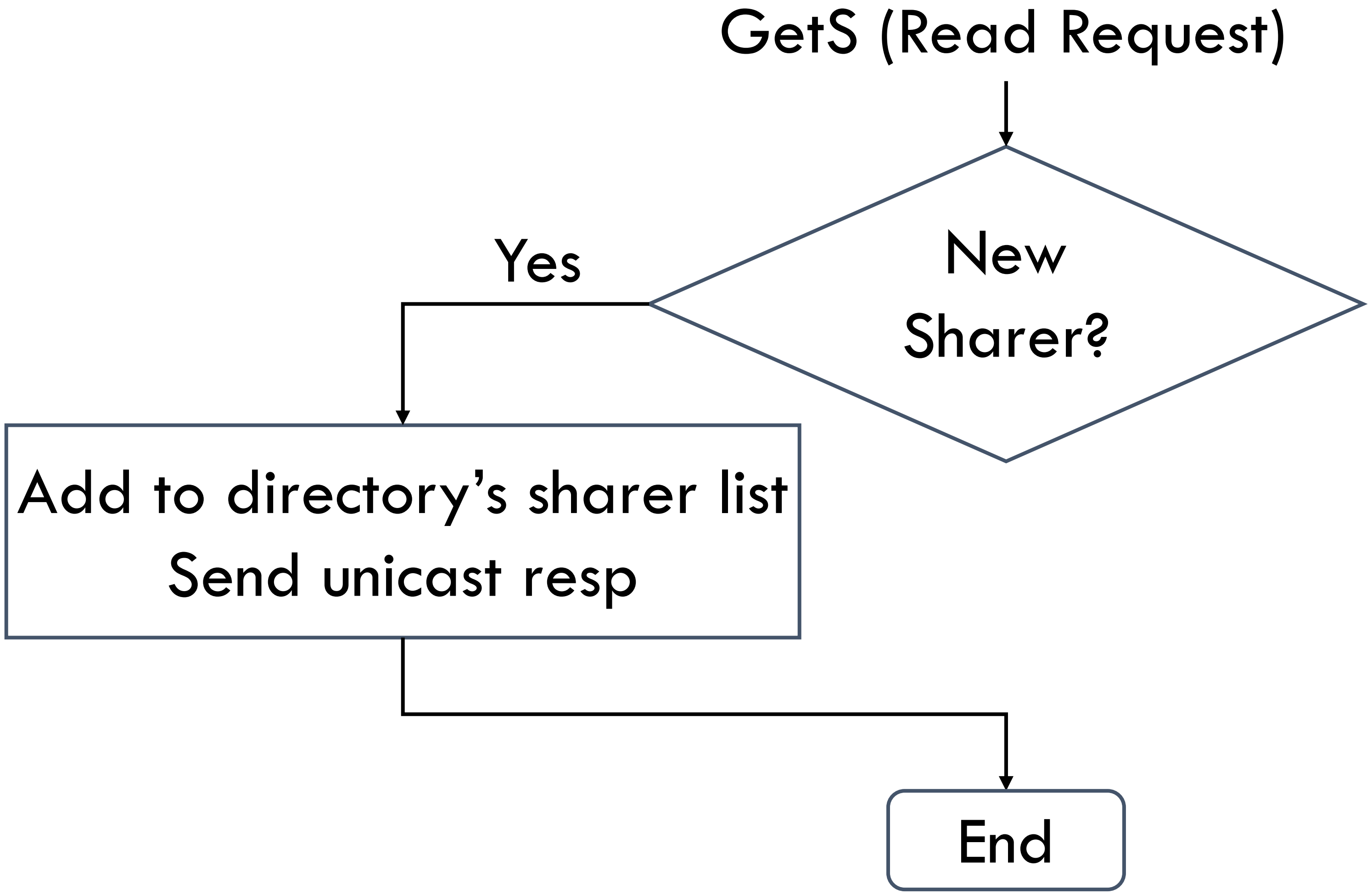
LLC Enhancements for Push Multicast

GetS (Read Request)

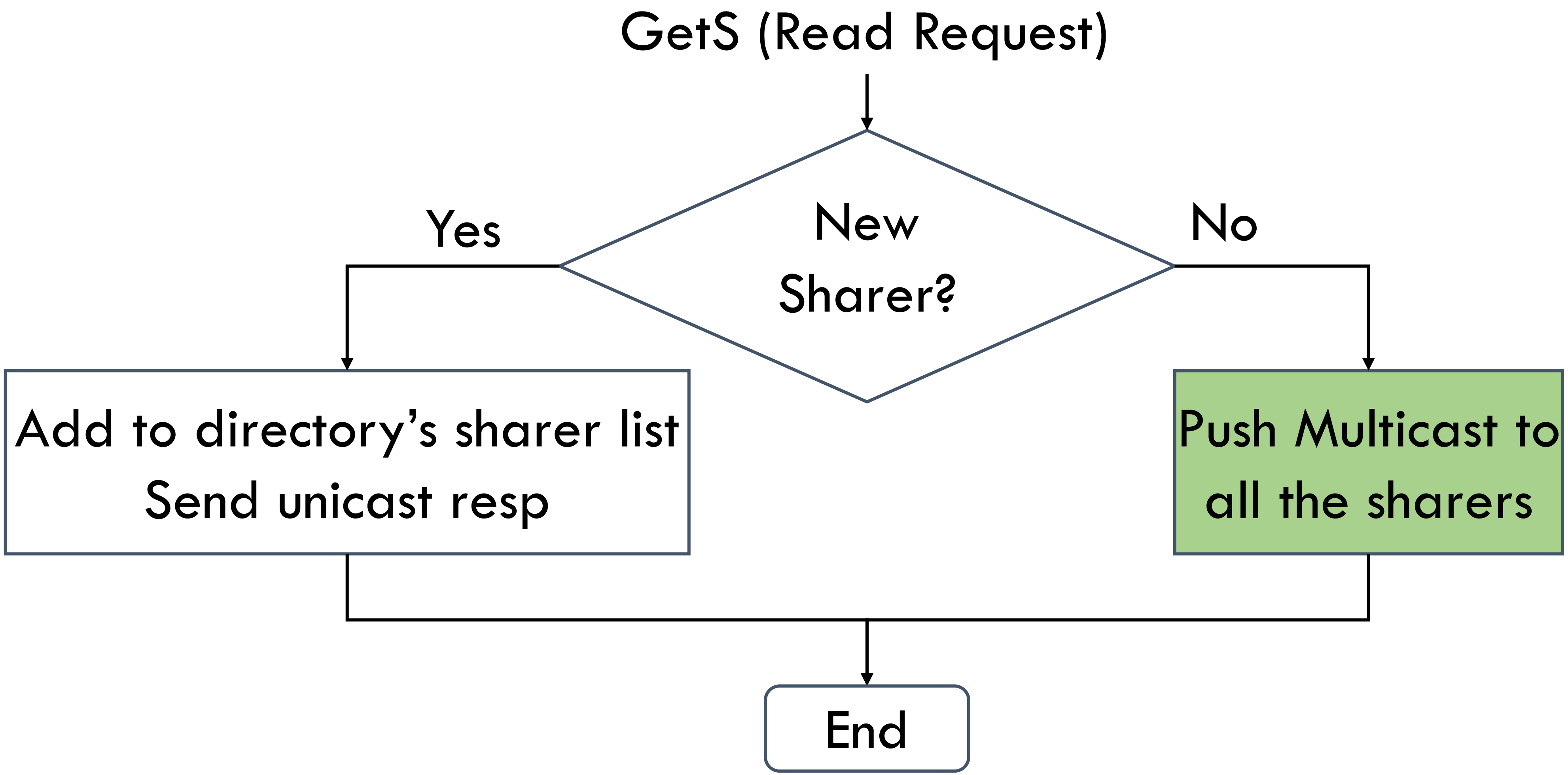
LLC Enhancements for Push Multicast



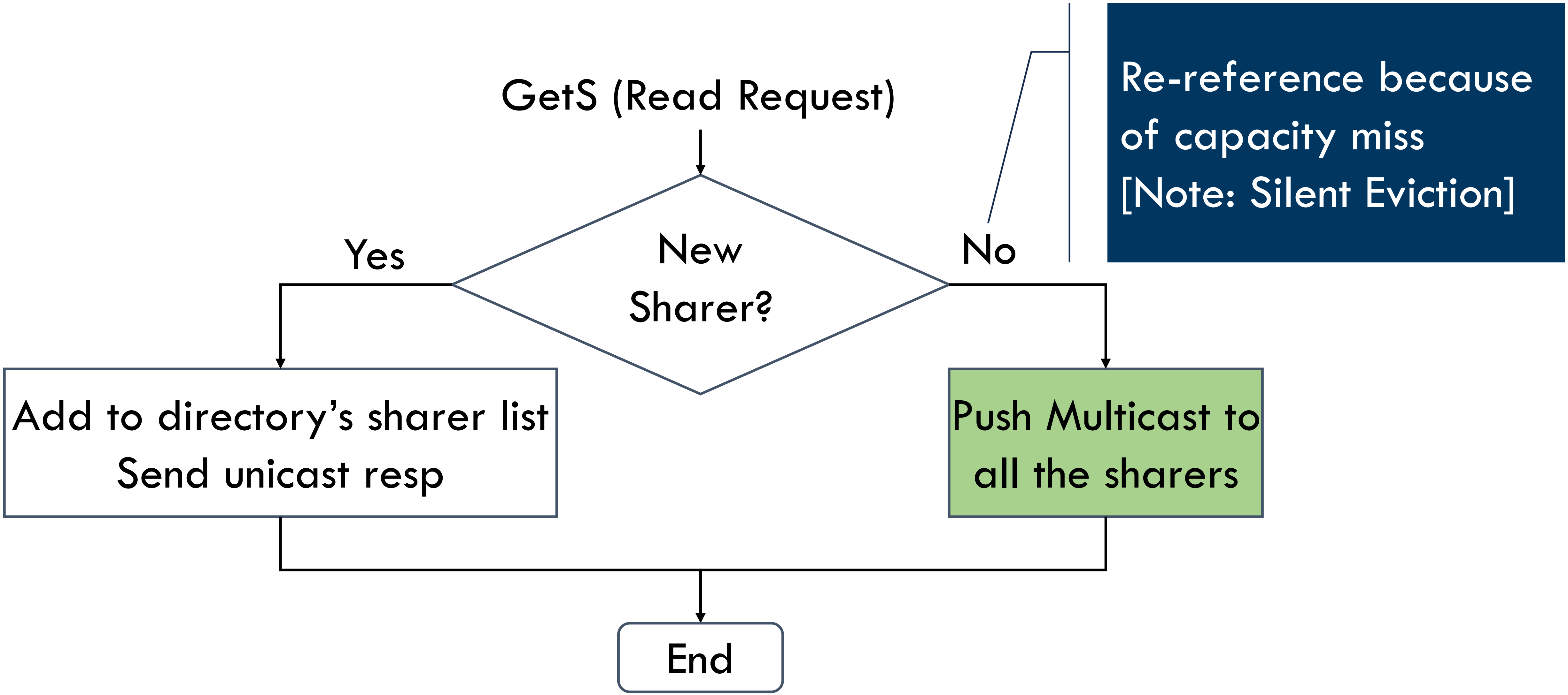
LLC Enhancements for Push Multicast



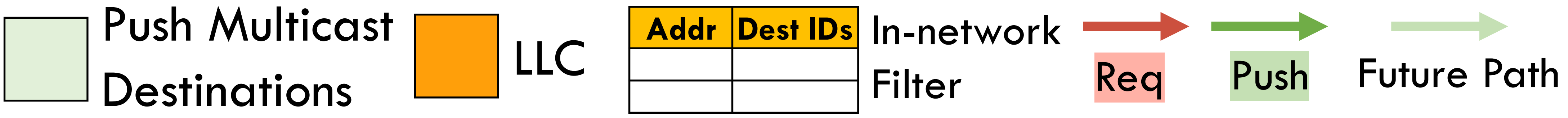
LLC Enhancements for Push Multicast

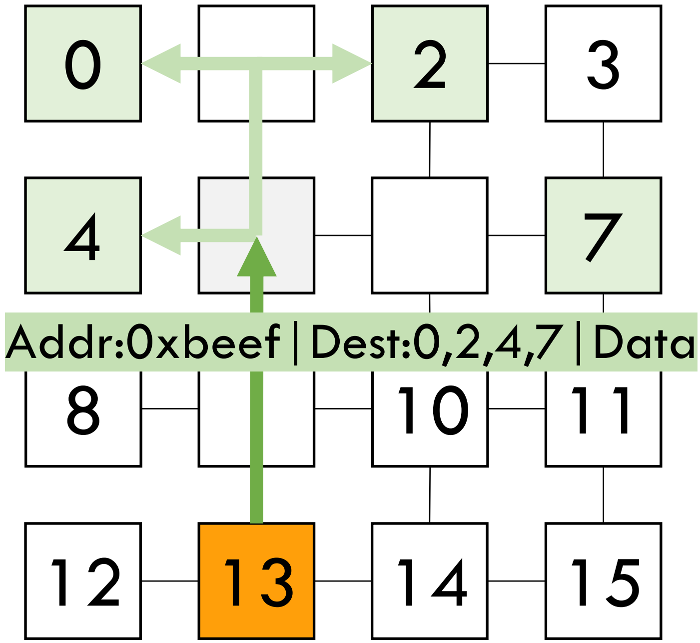


LLC Enhancements for Push Multicast

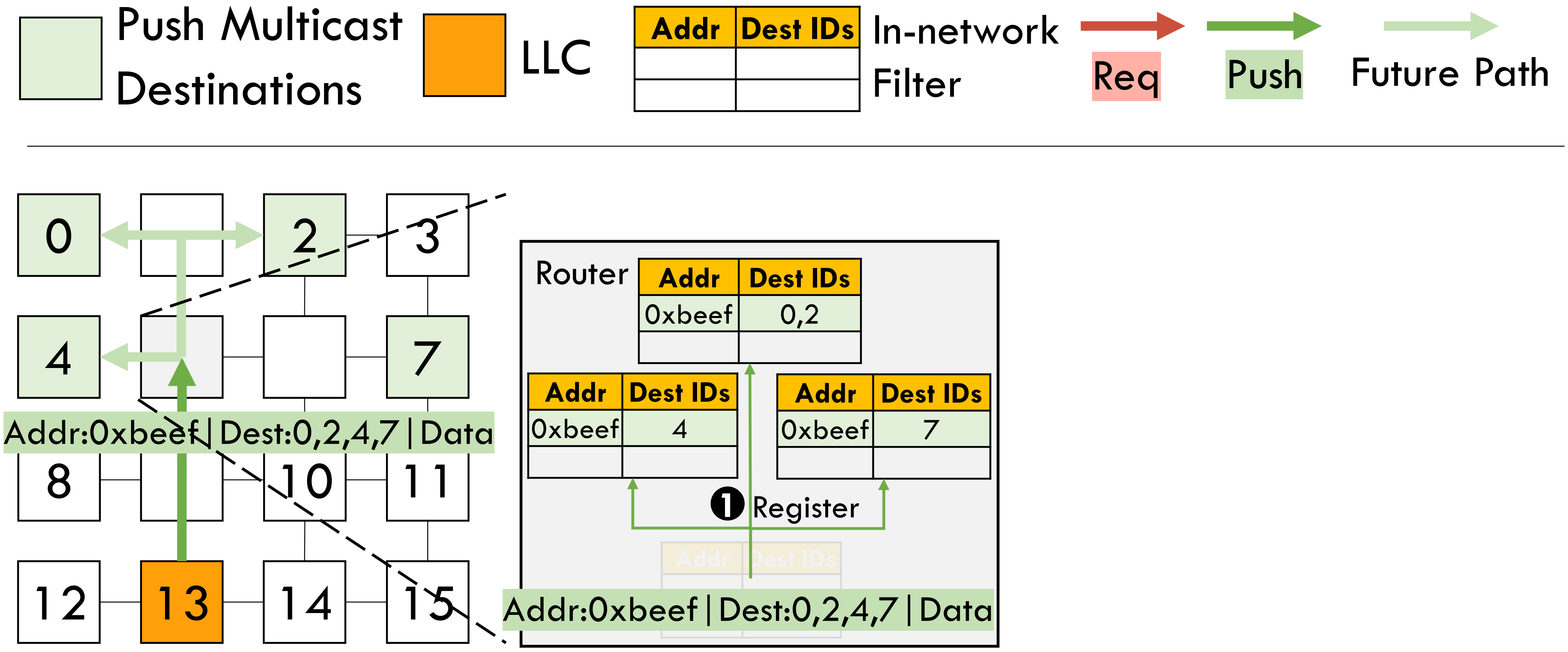


In-Network Filtering

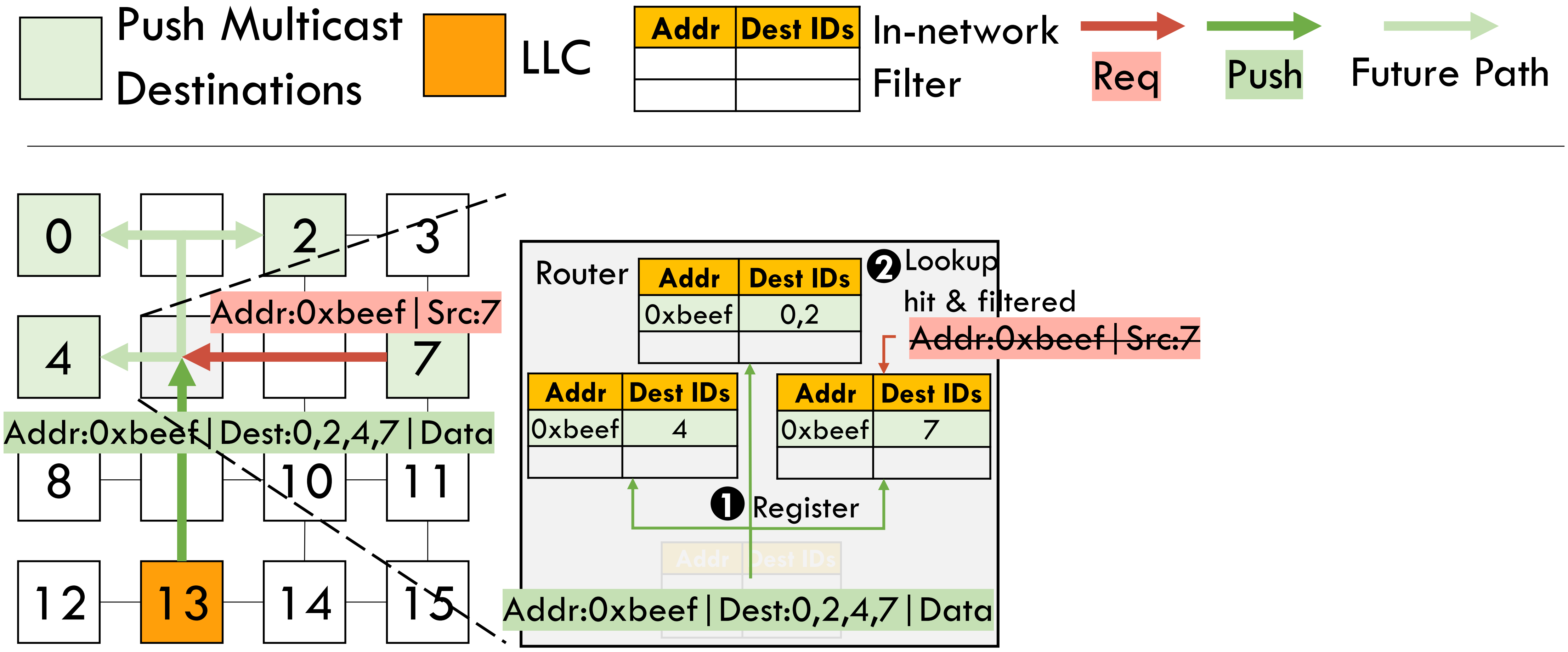




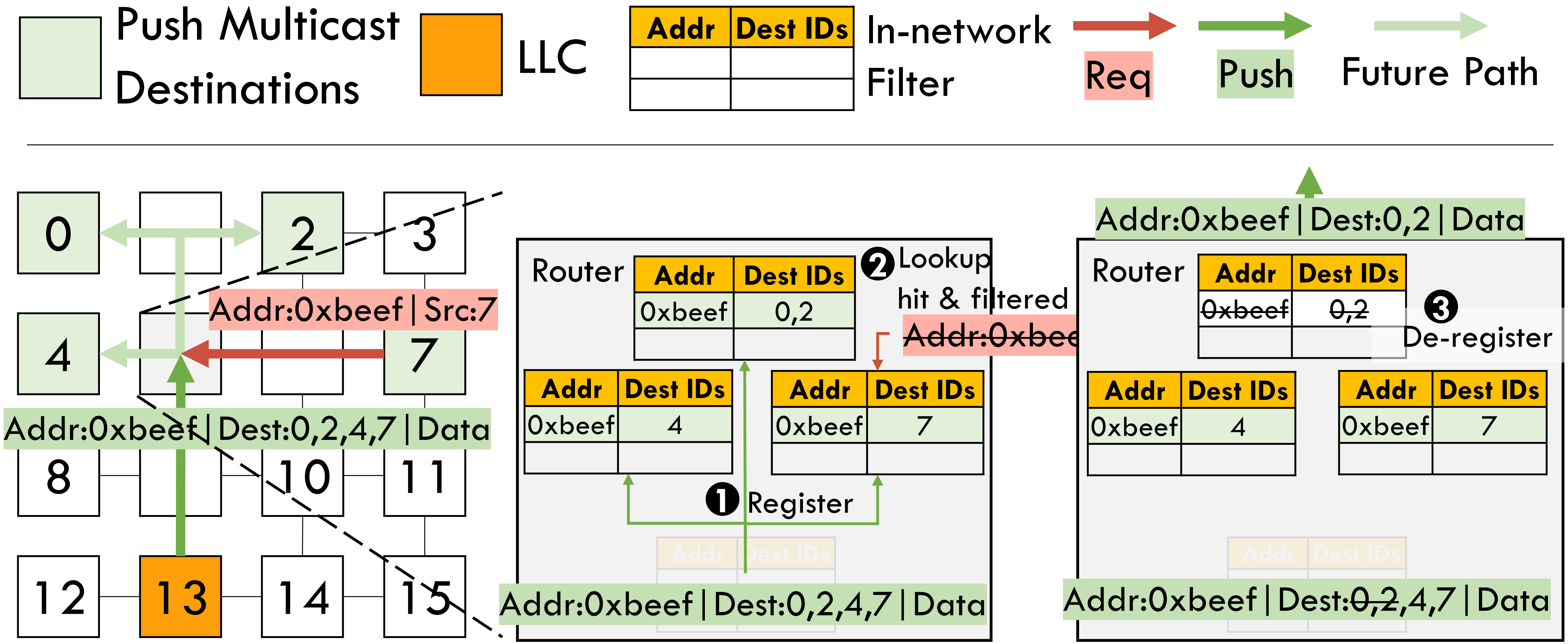
In-Network Filtering



In-Network Filtering



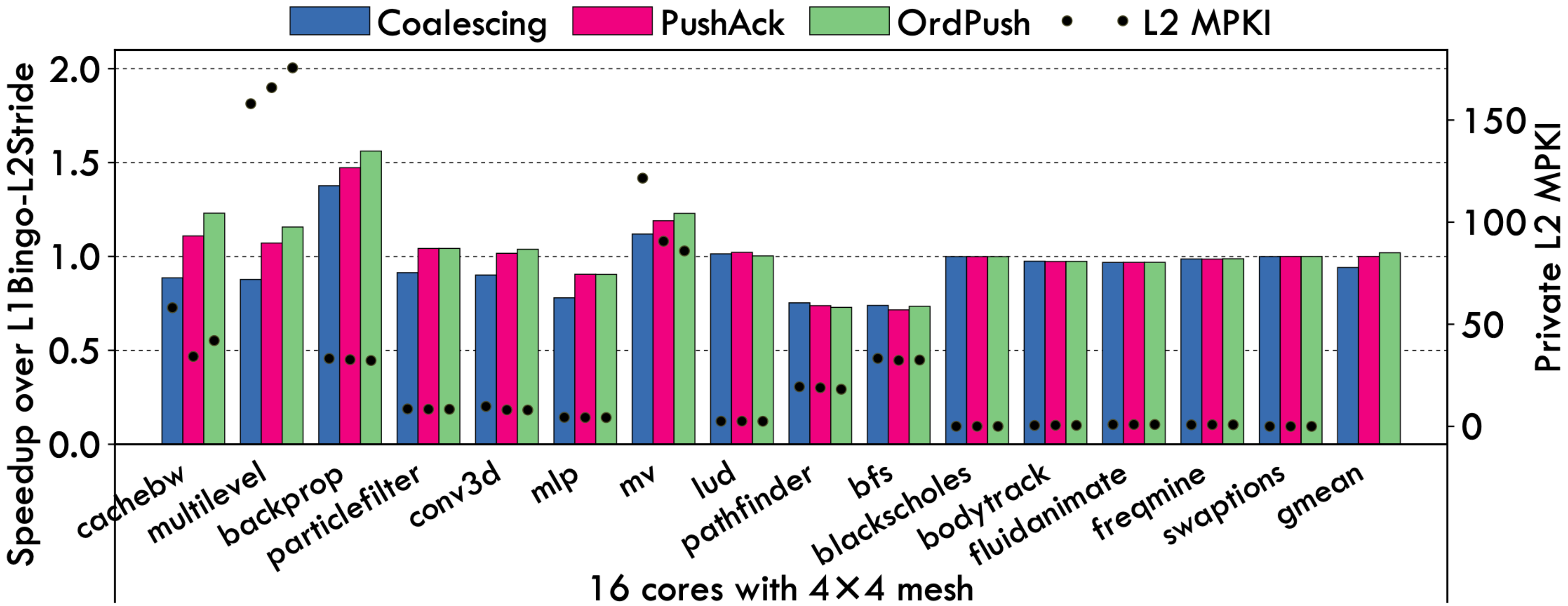
In-Network Filtering



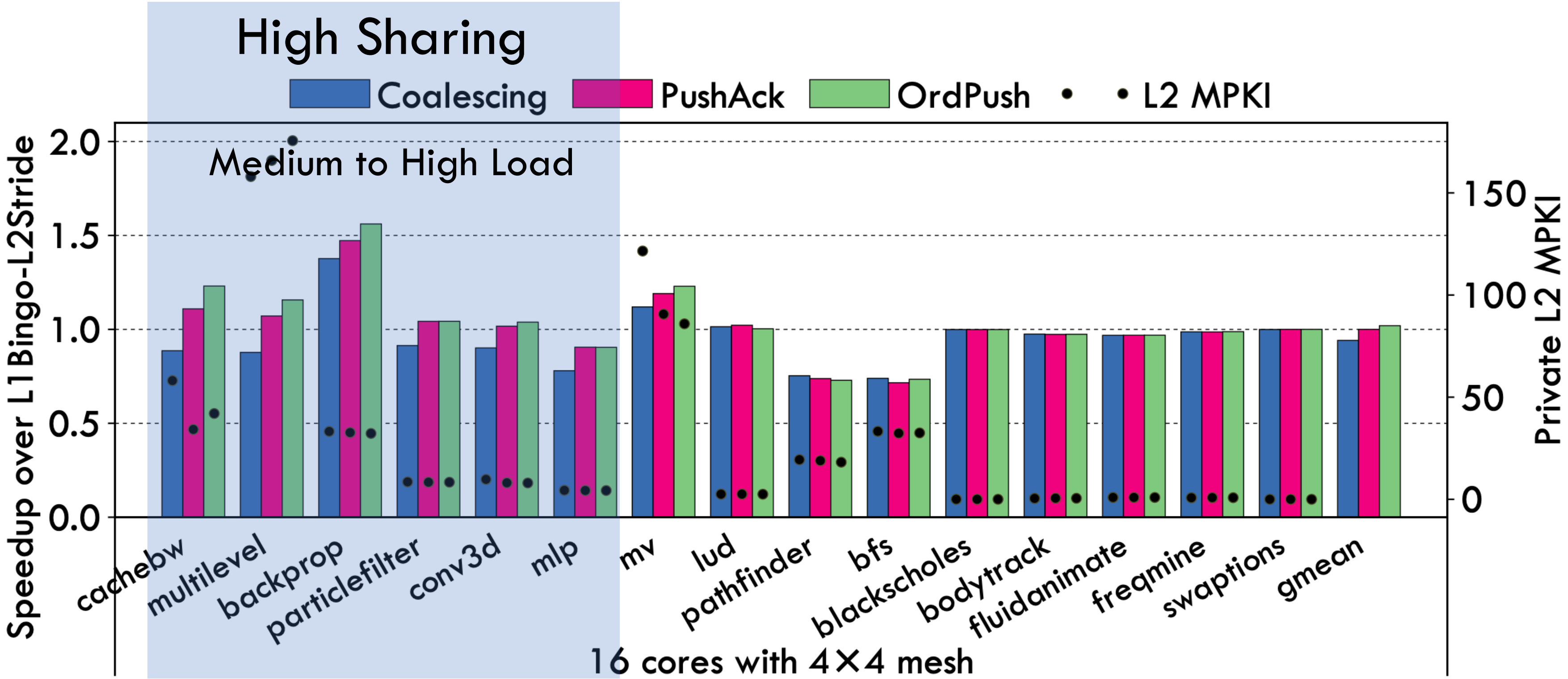
Evaluation

- gem5 v20.1 simulator
- Configuration
  - 4x4 and 8x8 tiles
  - 32KB L1I/L1D and 256KB L2
  - 1MB per-tile shared LLC slice, MESI coherence protocol
  - 2-cycle router, 1-cycle 128-bit link
    - Request: XY routing
    - Response: YX routing
- Workloads
  - Rodinia, libxsmm, microkernels, PARSEC (simlarge)
- Comparison
  - **L1Bingo-L2Stride**: Two-level prefetching (**Normalization Baseline)**
  - **Coalesce**: Request coalescing at LLC with multicast
  - **PushAck**
  - **OrdPush** (Ordered NoC)

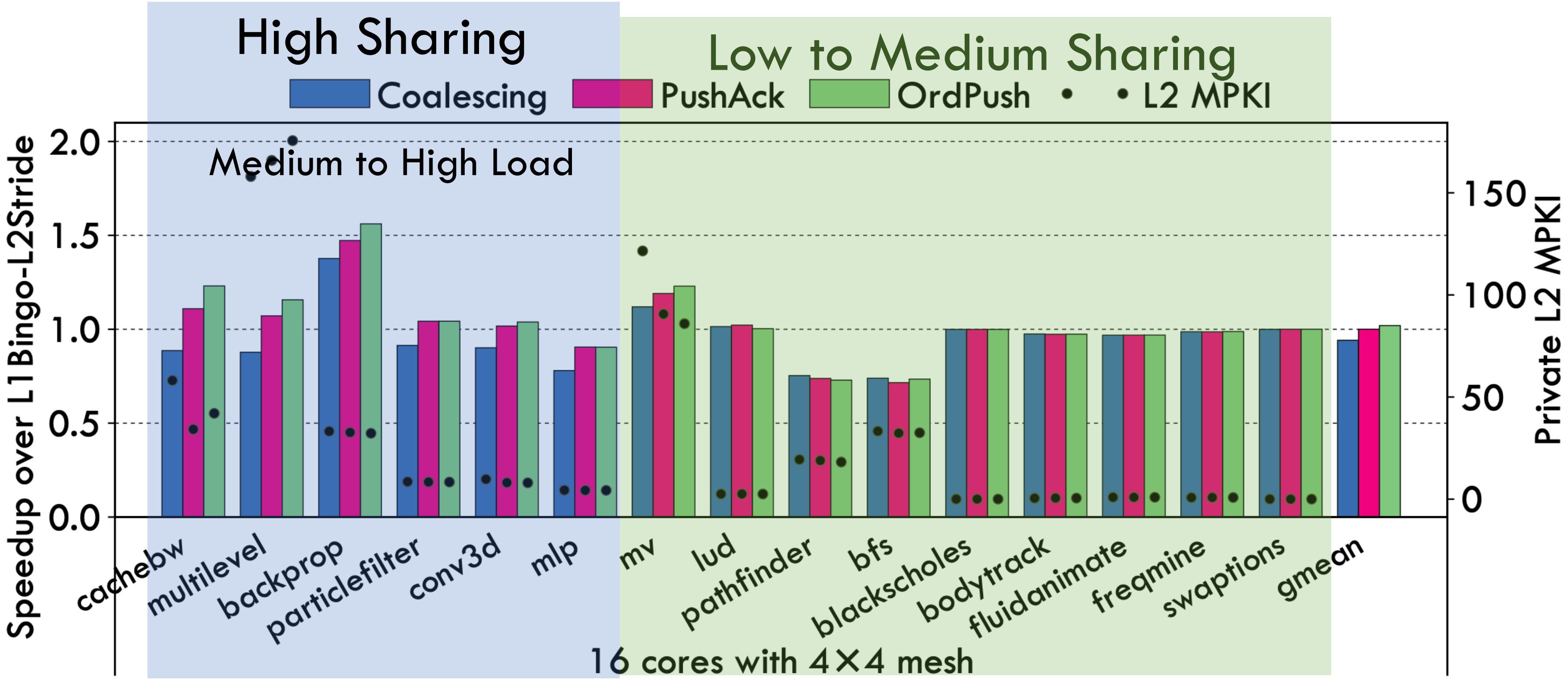
Performance Evaluation Result (16 cores)



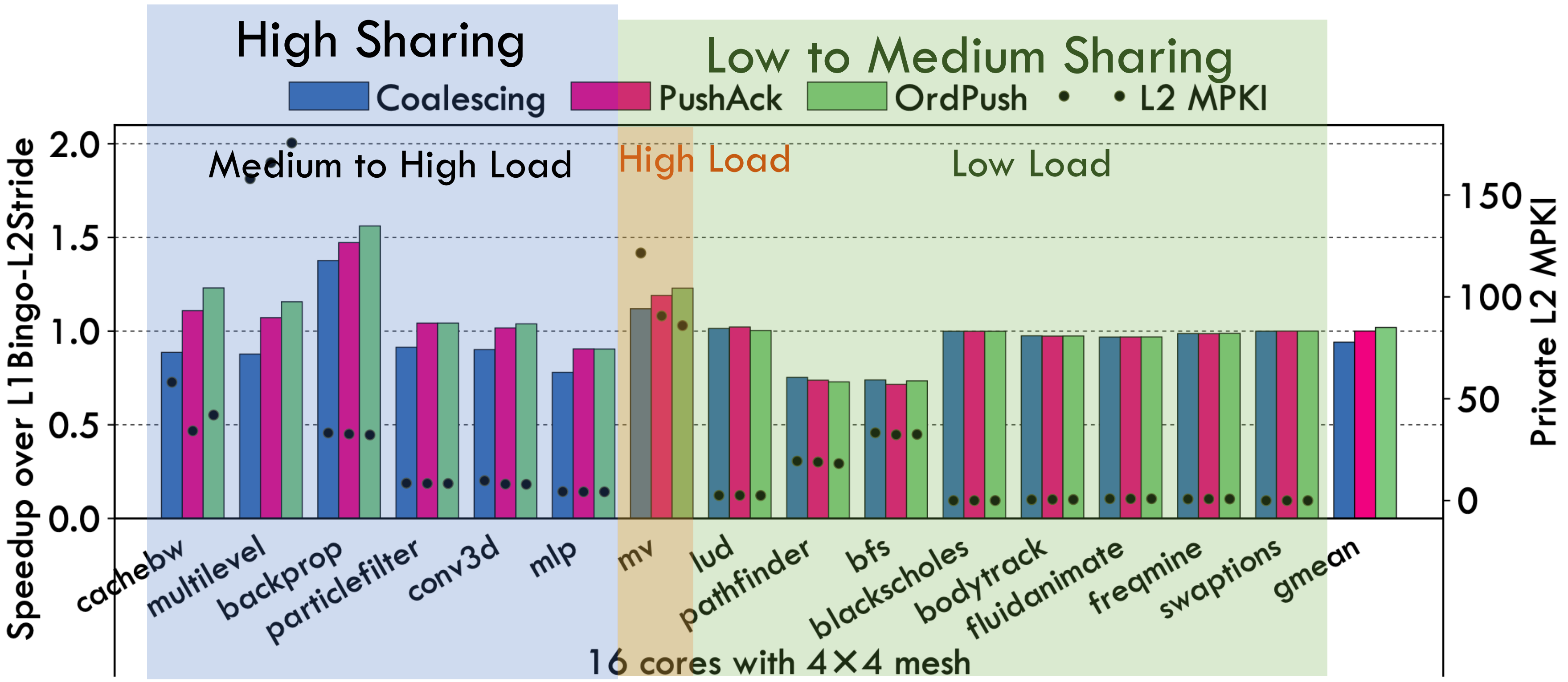
Performance Evaluation Result (16 cores)



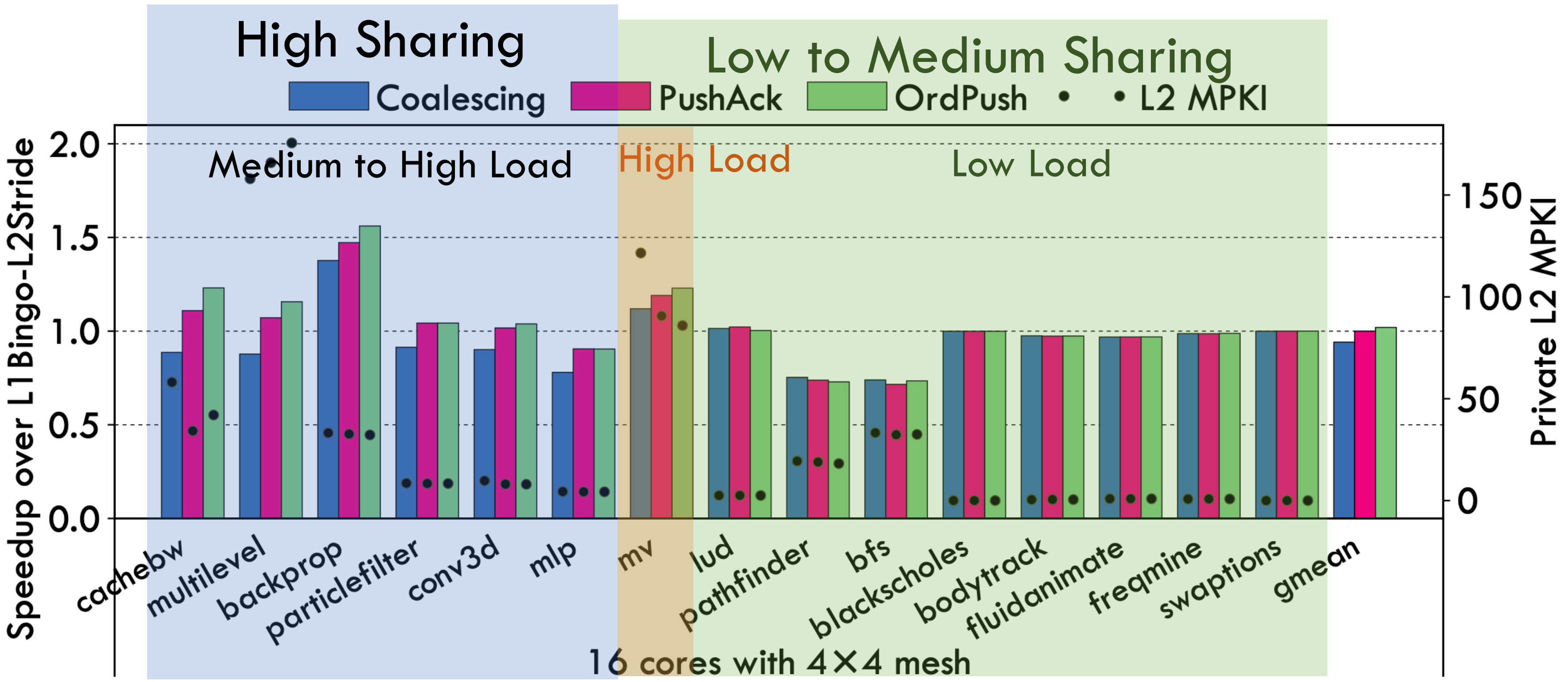
Performance Evaluation Result (16 cores)



Performance Evaluation Result (16 cores)

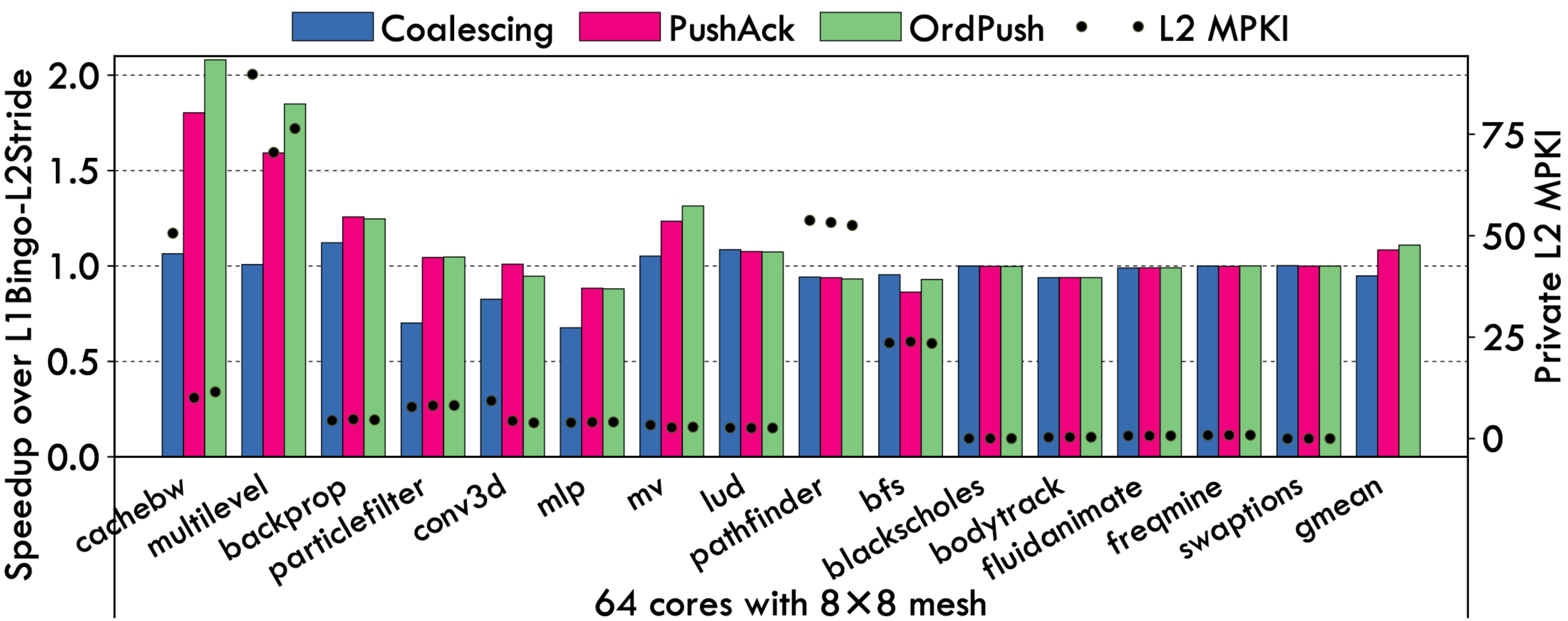


Performance Evaluation Result (16 cores)



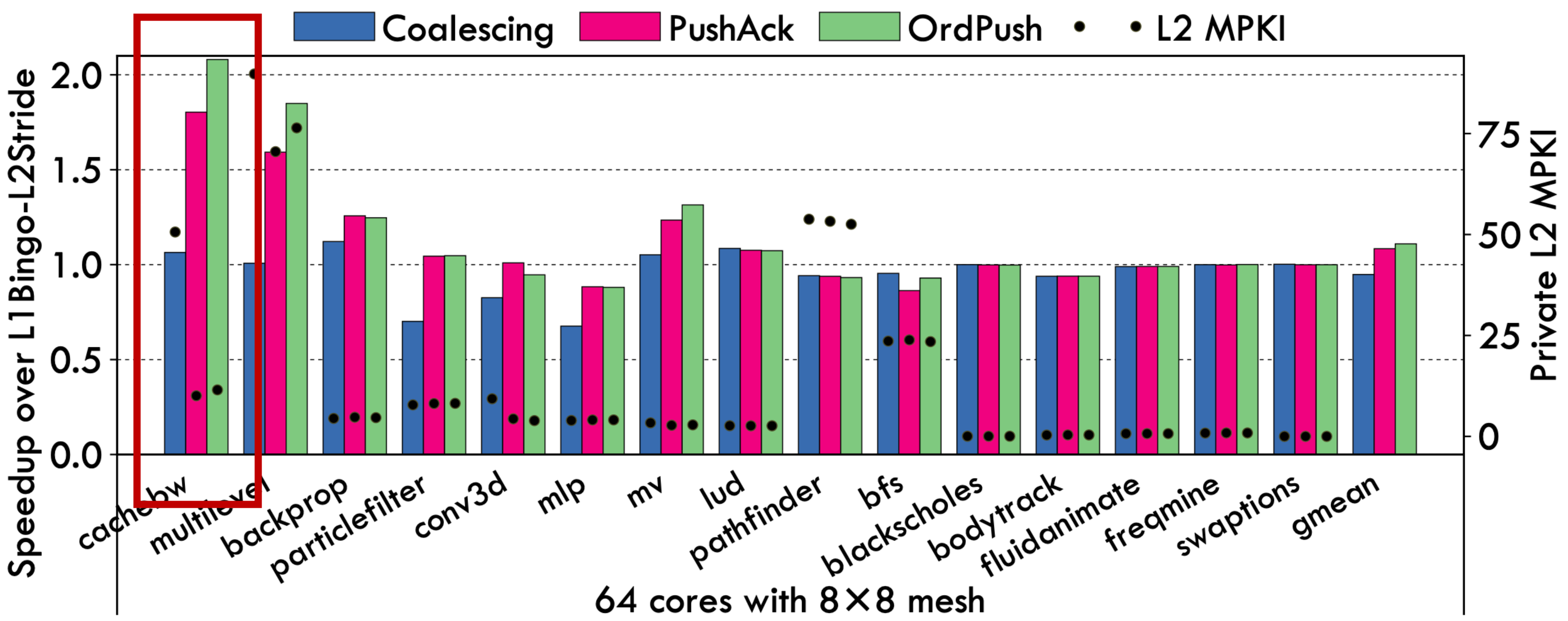
- For high load scenarios:
  - 16-core system: Achieve 10% - 50% performance speedup

Performance Evaluation Result (64 cores)



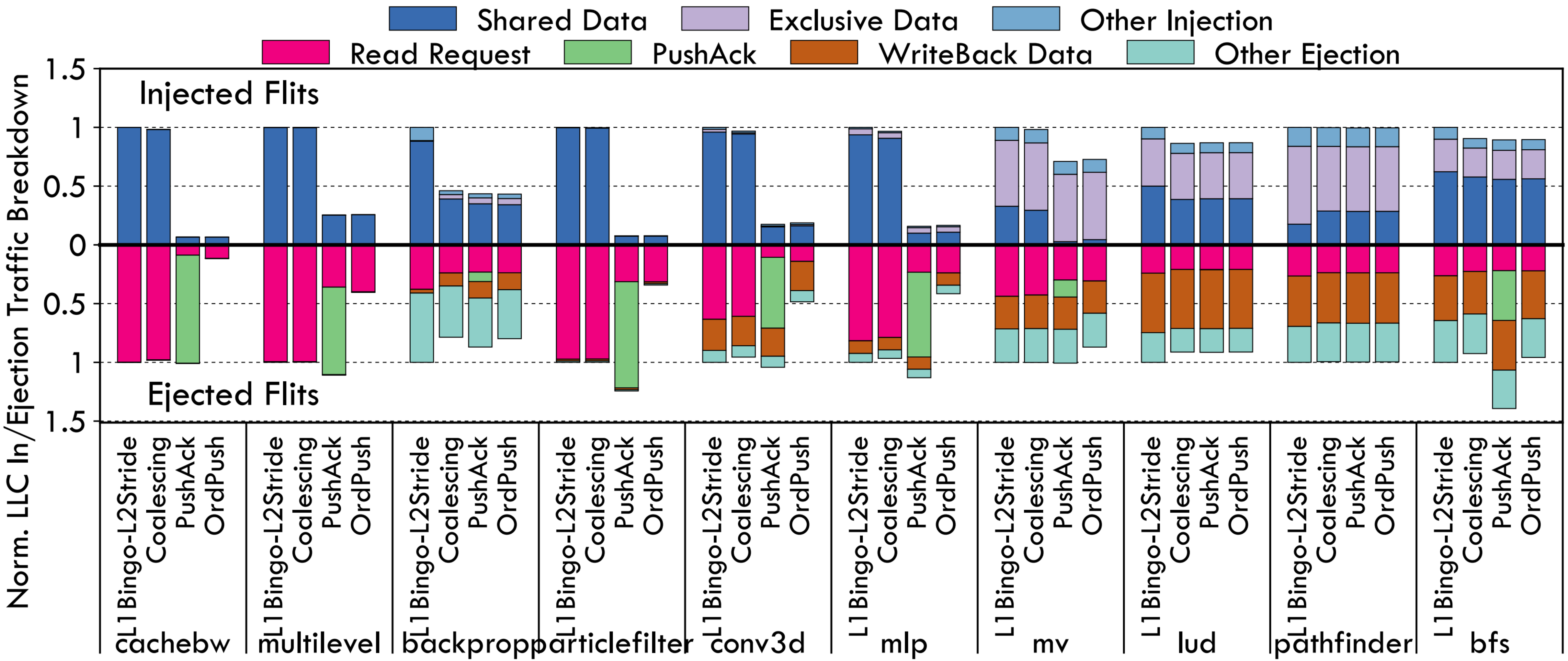
- For high load scenarios:
  - 16-core system: Achieve 10% - 50% performance speedup

Performance Evaluation Result (64 cores)

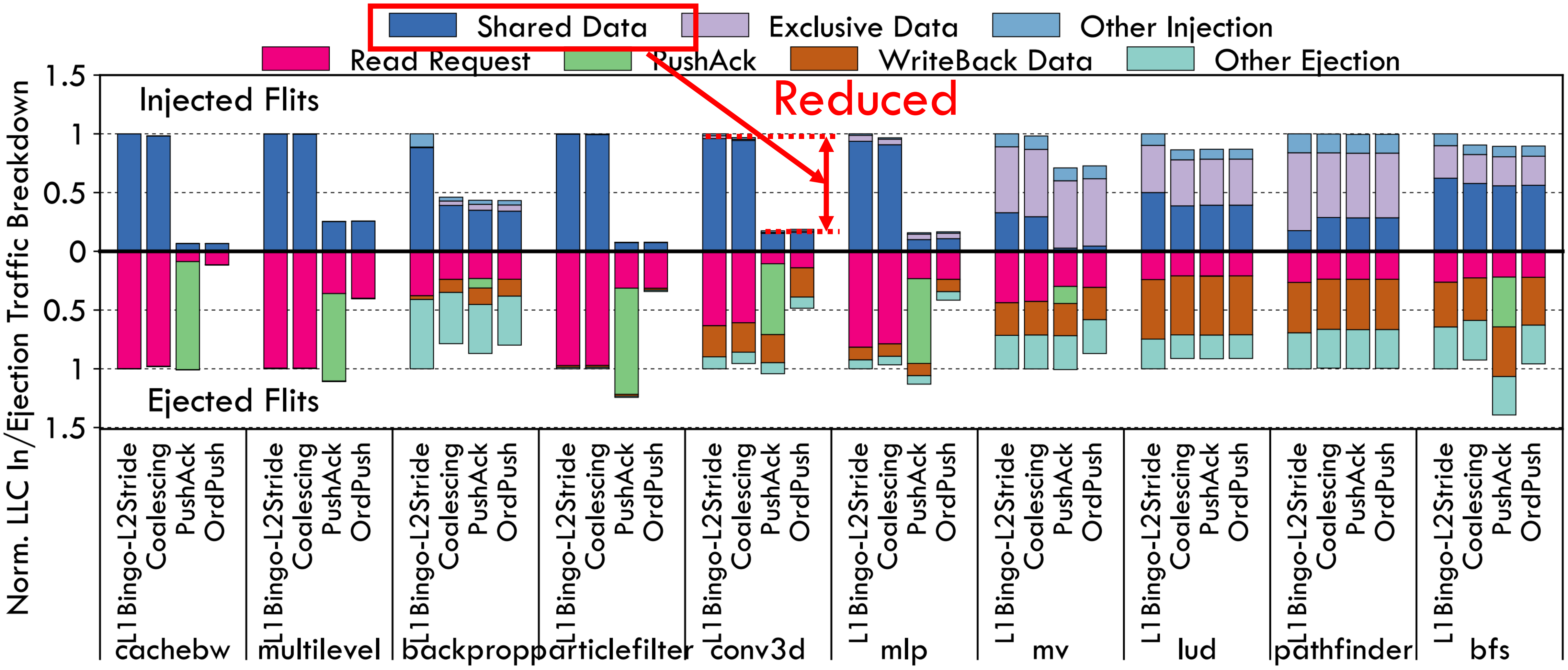


- For high load scenarios:
  - 16-core system: Achieve 10% - 50% performance speedup
  - 64-core system: Can achieve 2x performance speedup

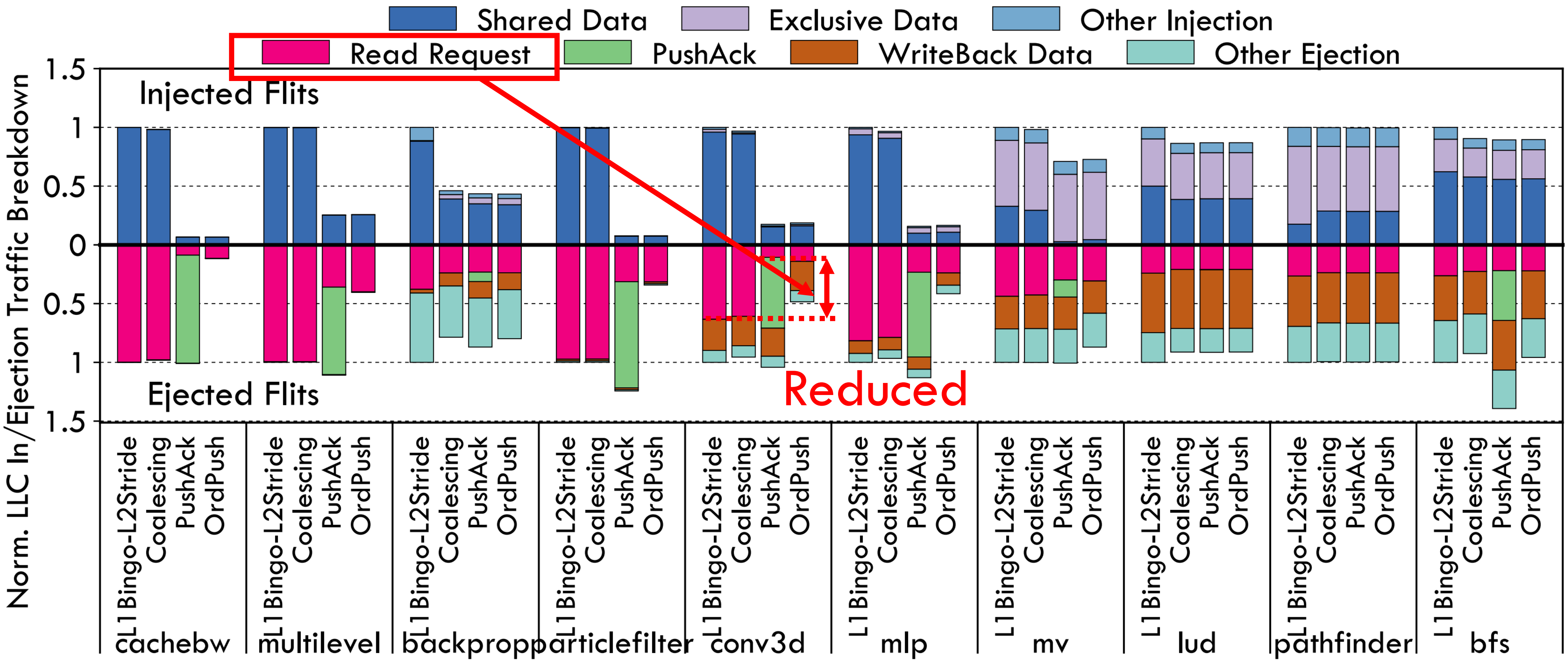
LLC Bandwidth



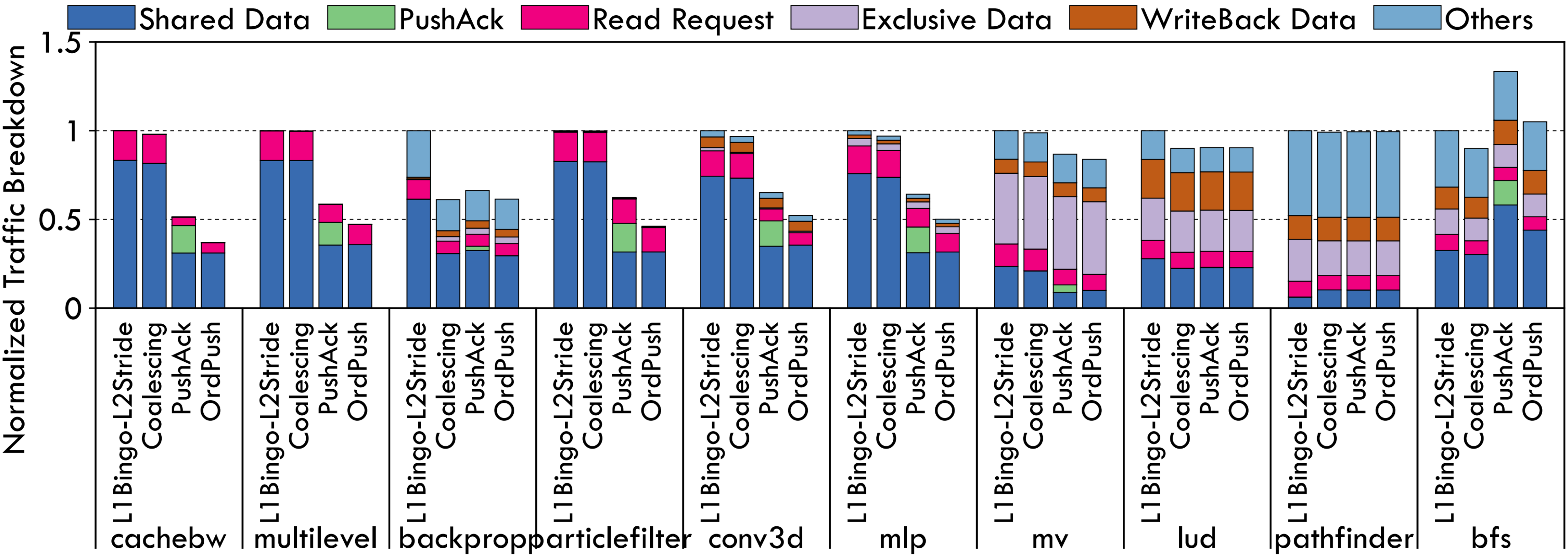
LLC Bandwidth



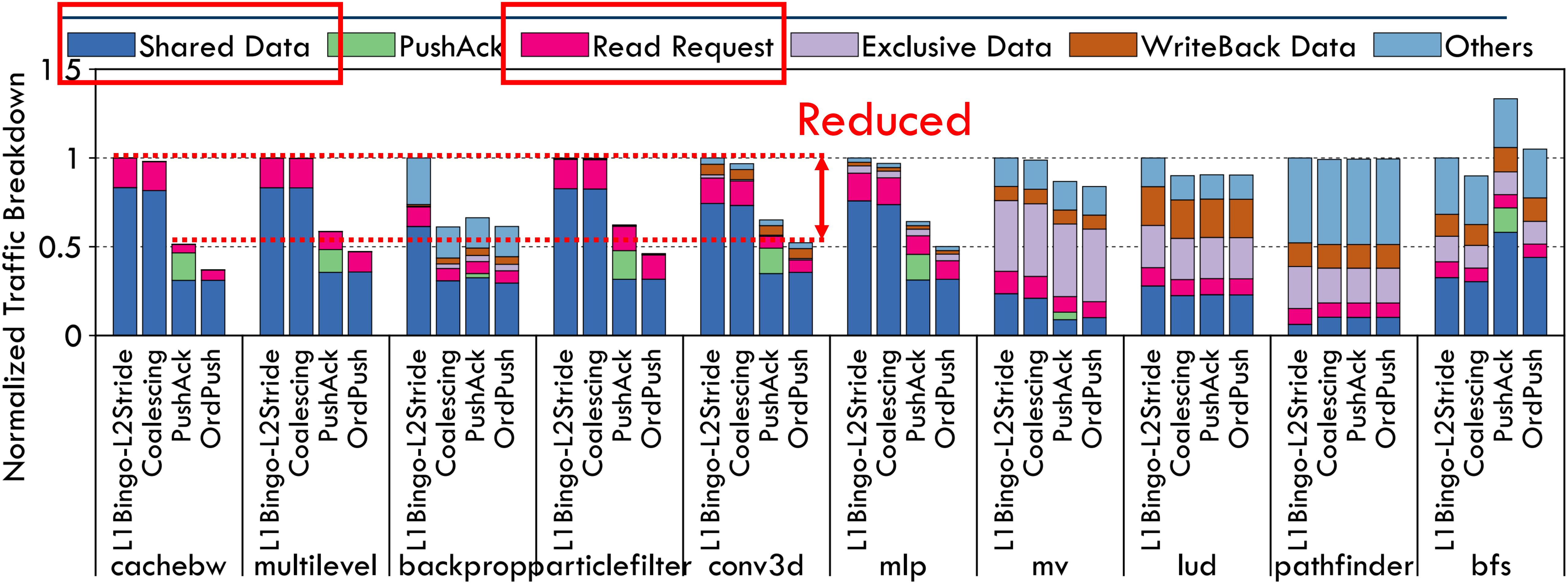
LLC Bandwidth



NoC Bandwidth



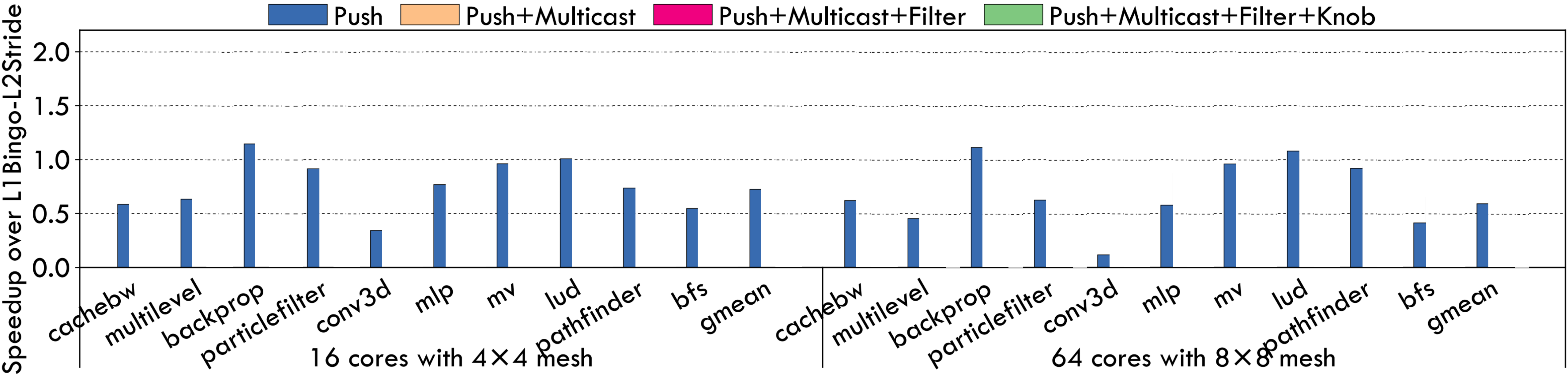
NoC Bandwidth



- 33% traffic reduction on average

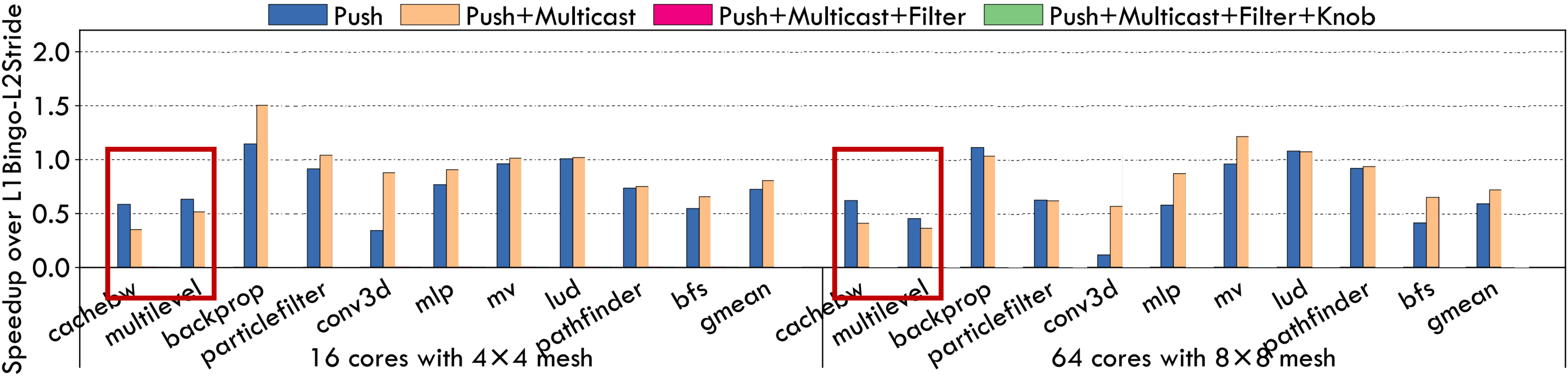
Ablation Study

- Add push, multicast, filter, knob (feedback) step by step



Ablation Study

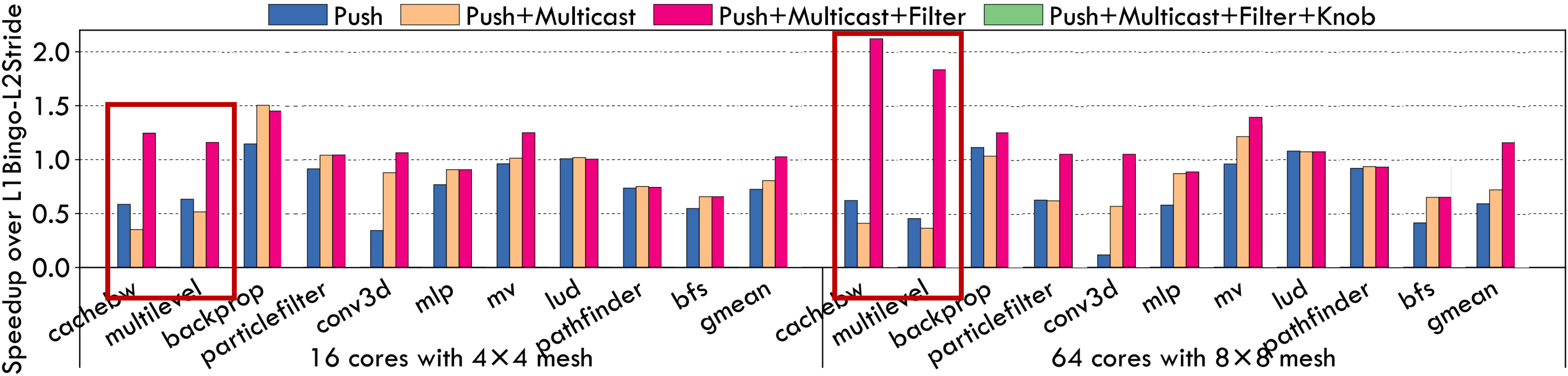
- Add push, multicast, filter, knob (feedback) step by step



- Push and Push+Multicast increase traffic overhead

Ablation Study

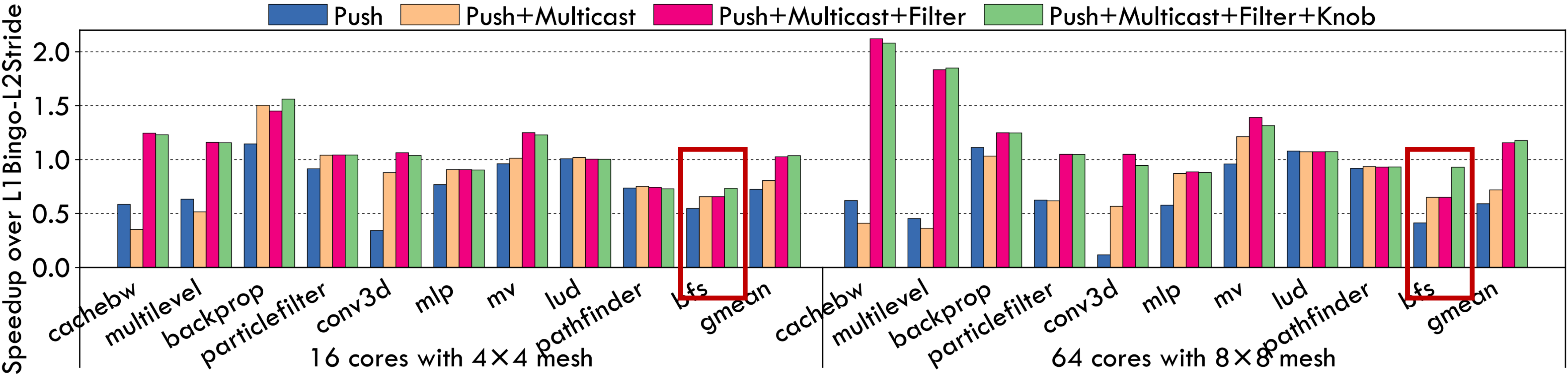
- Add push, multicast, filter, knob (feedback) step by step



- Push and Push+Multicast increase traffic overhead
- Filter eliminates redundant requests and pushes, better performance

Ablation Study

- Add push, multicast, filter, knob (feedback) step by step



- Push and Push+Multicast increase traffic overhead
- Filter eliminates redundant requests and pushes, better performance
- Knob can pause push in time to avoid performance harm (bfs)

Summary

- **Problem:** Large working set leads to high pressure on NoC and LLC

Summary

- **Problem:** Large working set leads to high pressure on NoC and LLC
- **Insight:** Considerable portion is read-shared data
  - We can use coalescing and multicast to address the bandwidth problem
- **Challenge:** Read-shared data accesses have temporal locality but not that close due to thread variation and NUCA effect

Summary

- **Problem:** Large working set leads to high pressure on NoC and LLC
- **Insight:** Considerable portion is read-shared data
  - We can use coalescing and multicast to address the bandwidth problem
- **Challenge:** Read-shared data accesses have temporal locality but not that close due to thread variation and NUCA effect

- **Push Multicast**
  - Predict the sharers for speculative multicast
  - In-network coherent filtering for pruning redundant requests
  - Dynamic feedback-based pause-and-resume knob

Thanks and Questions

Email: hjy@hkust-gz.edu.cn

"Push Multicast: A Speculative and Coherent Interconnect for Mitigating Manycore CPU Communication Bottleneck," Jiayi Huang, Yanhua Chen, Zhe Wang, Christopher J. Hughes, Yufei Ding, Yuan Xie, HPCA 2025.

香港科技大学(广州)
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

intel

UNIVERSITY OF CALIFORNIA · SAN DIEGO

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY