OSDI '26

When DDIO Meets Page Coloring: Revisiting DDIO Performance with **Sepia**

Changwoo Song, **Sanghyun Kim**, Jinhyeok Oh, Qizhe Cai, Joonsung Kim and Jaehyun Hwang

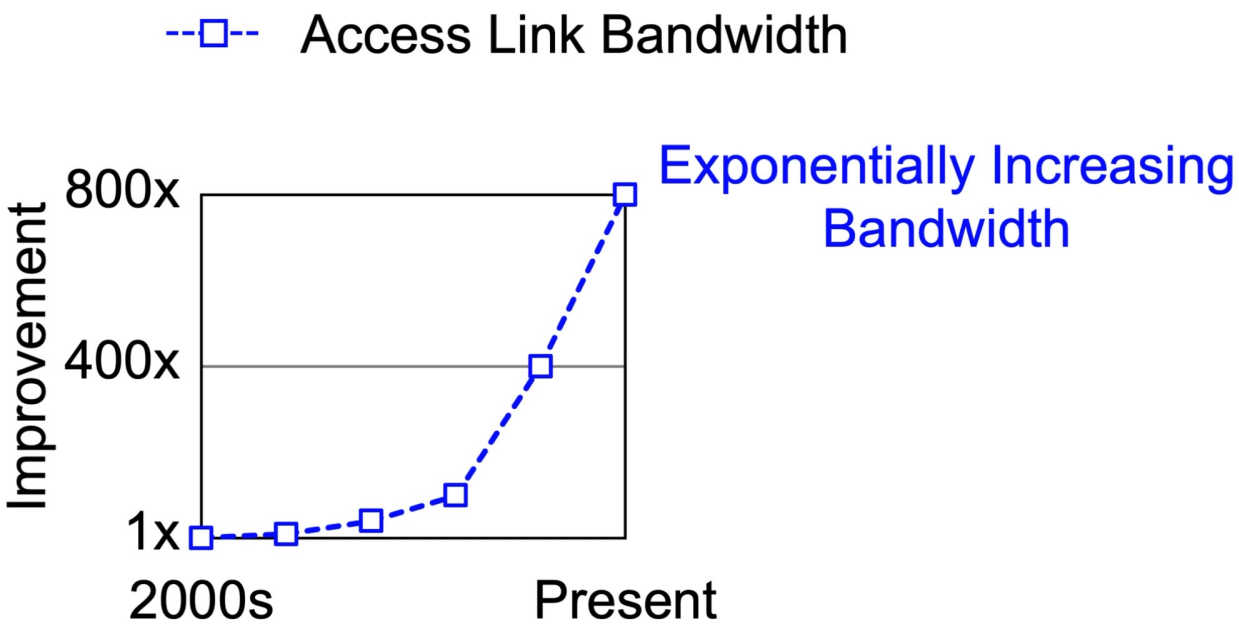
SUNG KYUN KWAN UNIVERSITY

UNIVERSITY of VIRGINIA

CPU Bottleneck at High-Speed Networks

1. Link Bandwidth Keeps Increasing

Network bandwidth grows rapidly



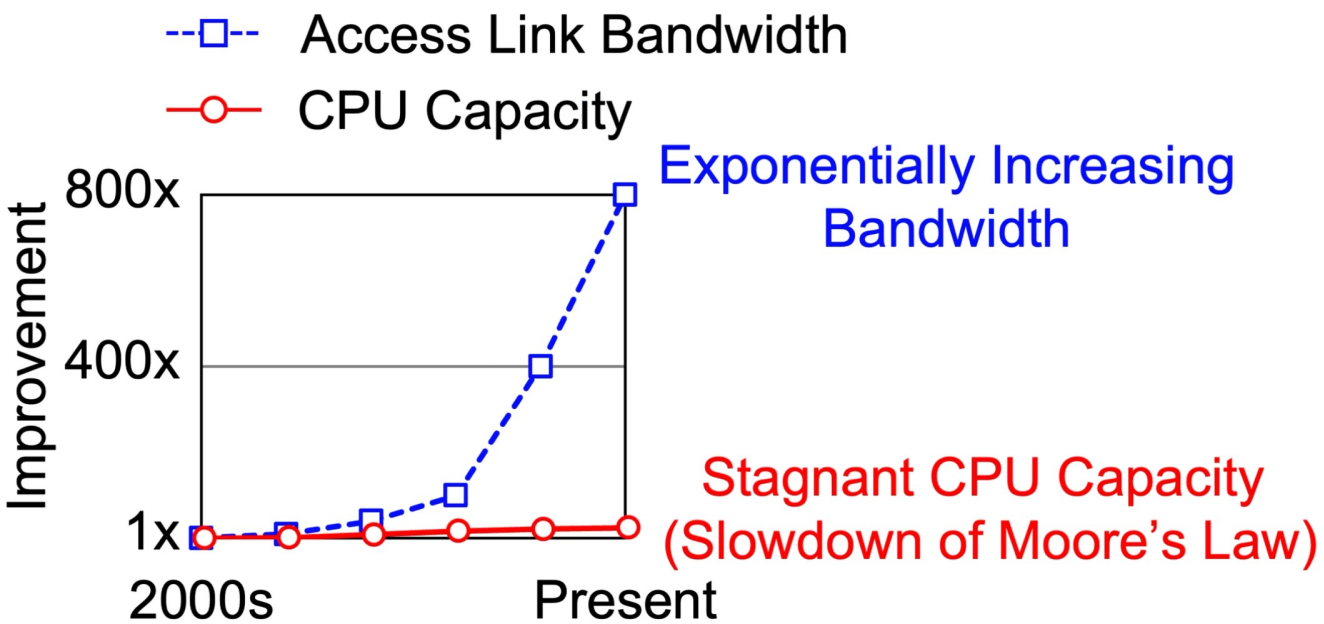
CPU Bottleneck at High-Speed Networks

1. Link Bandwidth Keeps Increasing

Network bandwidth grows rapidly

2. Insufficient CPU Processing Capacity

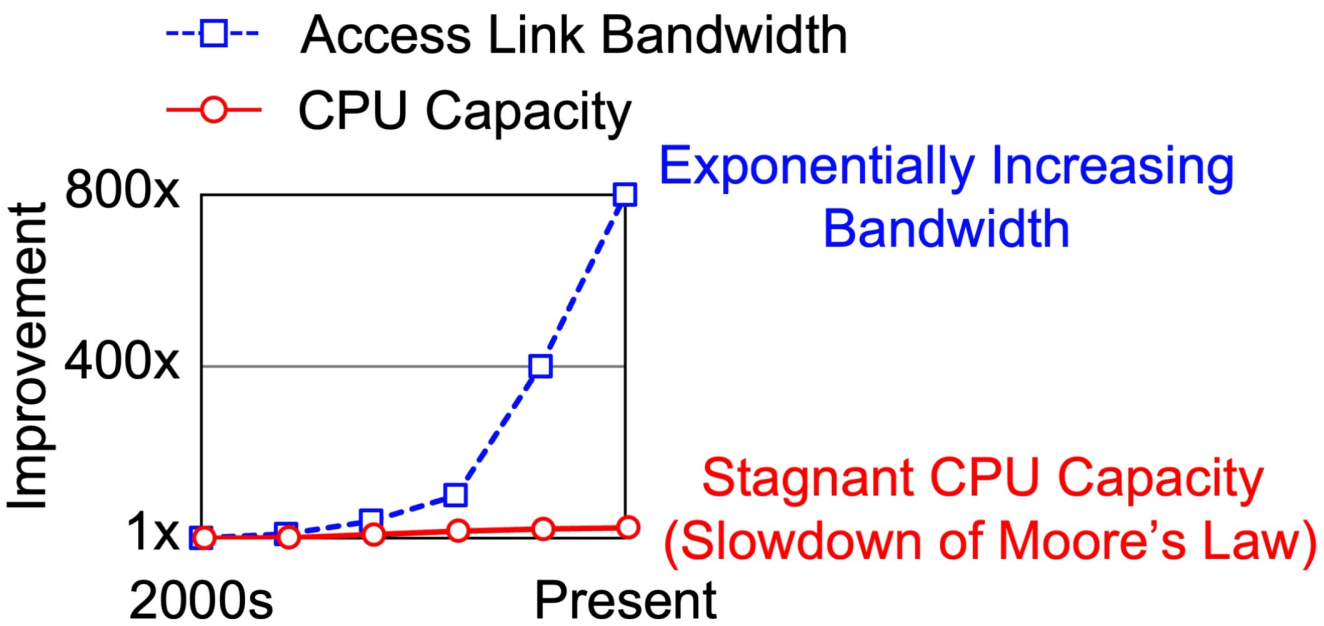
Better CPU efficiency is needed!

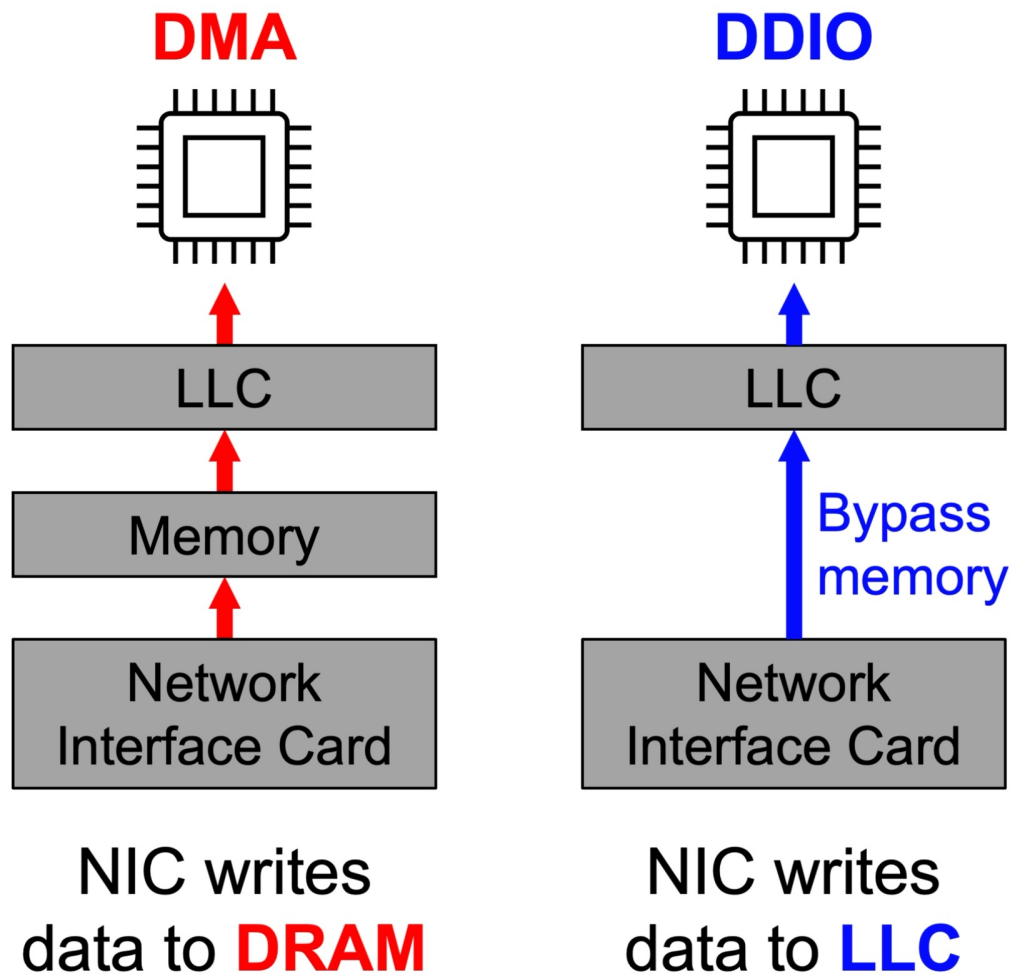


CPU Bottleneck at High-Speed Networks

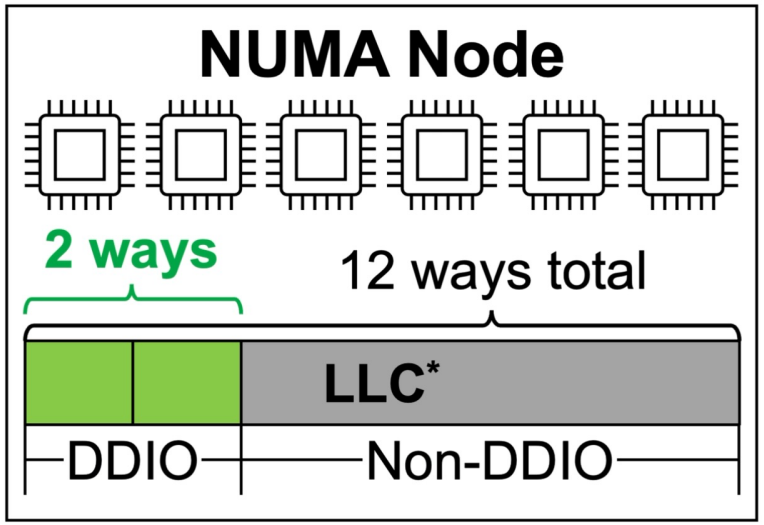
1. Link Bandwidth Keeps Increasing
Network bandwidth grows rapidly

2. Insufficient CPU Processing Capacity
Better CPU efficiency is needed!





DDIO Overview

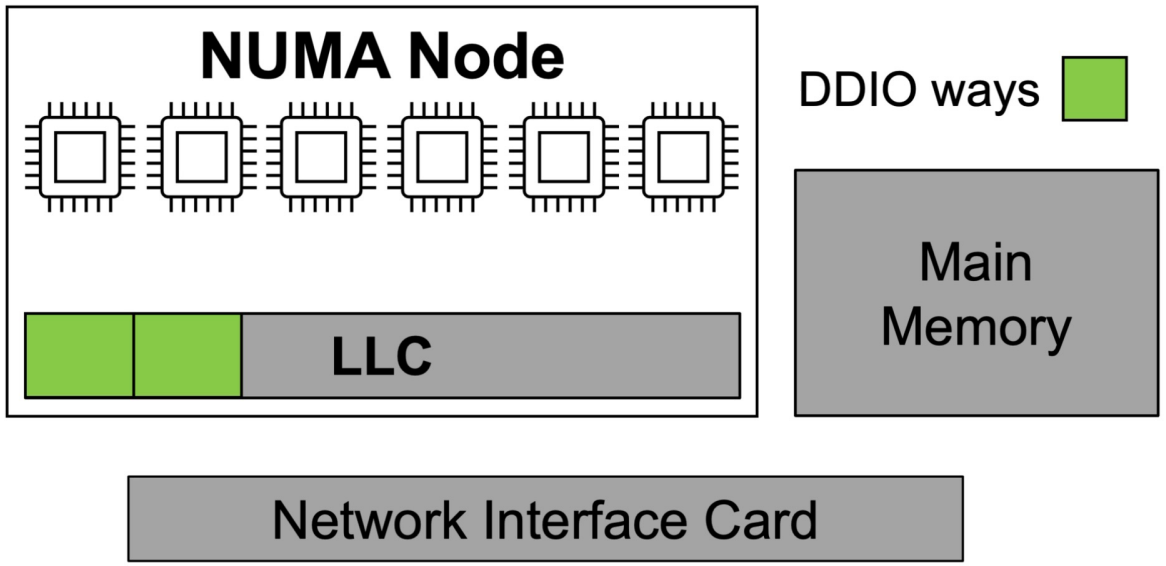


DDIO (Data Direct I/O)

- Intel's Direct Cache Access (**DCA**) technology
- Use DDIO-reserved LLC ways, **e.g., 2 / 12 ways**
- **Avoid cold misses**

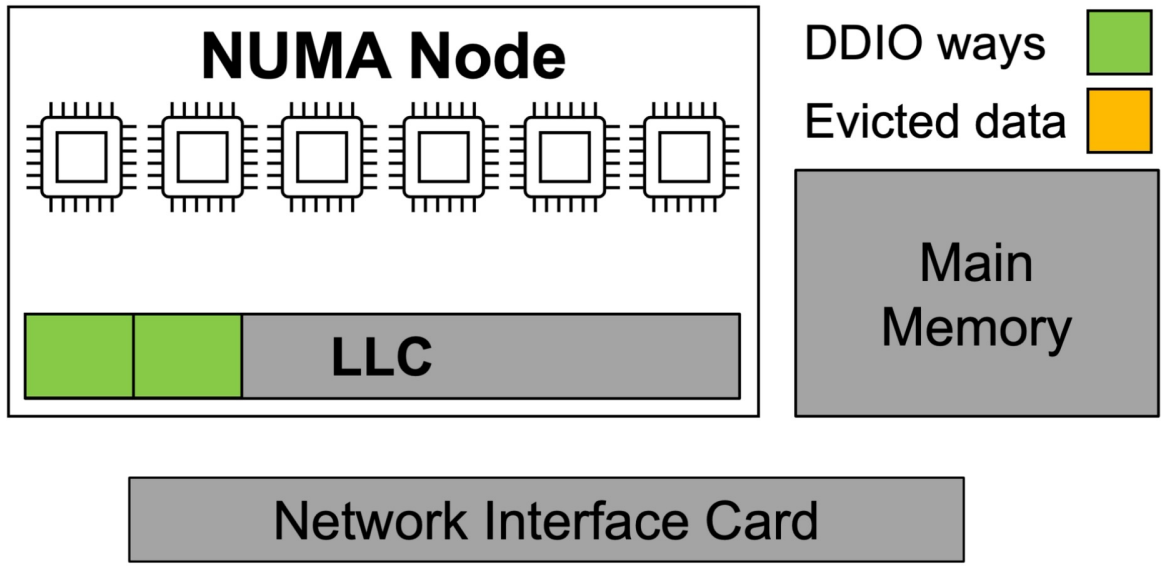
*** Last Level Cache (LLC)**

Leaky DMA Problem



Case 1: Low Traffic

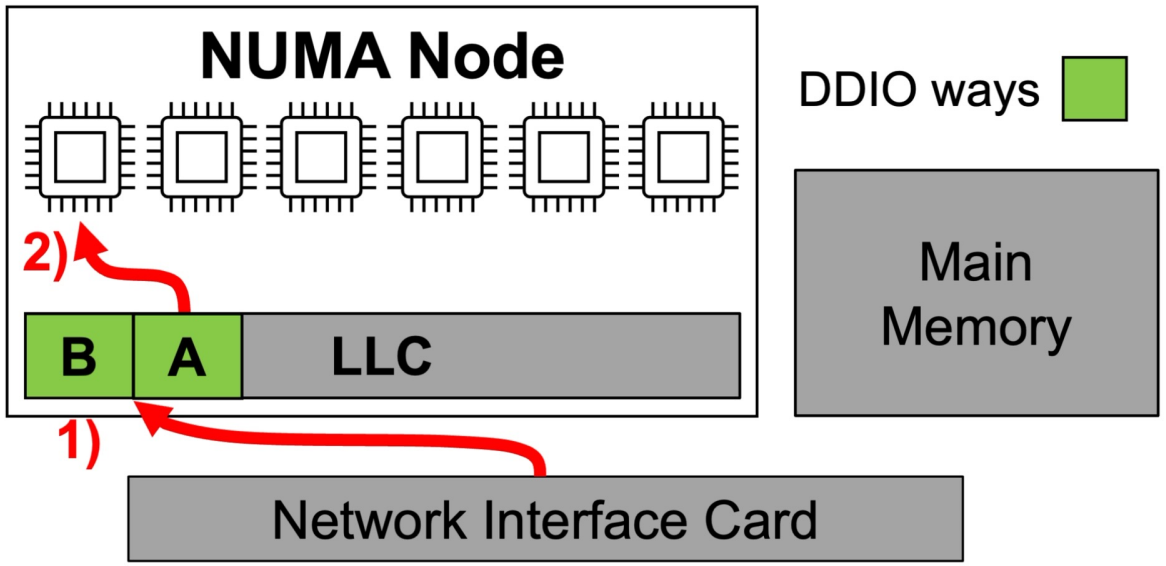
Avoid memory access latency



Case 2: High Traffic

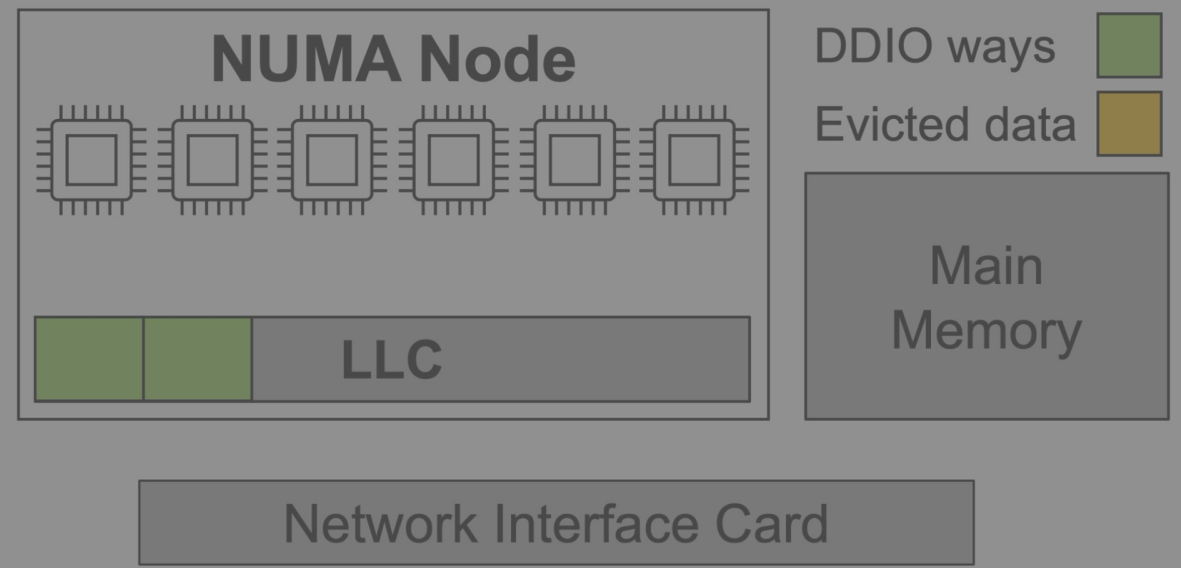
Incur memory access latency

Leaky DMA Problem



Case 1: Low Traffic

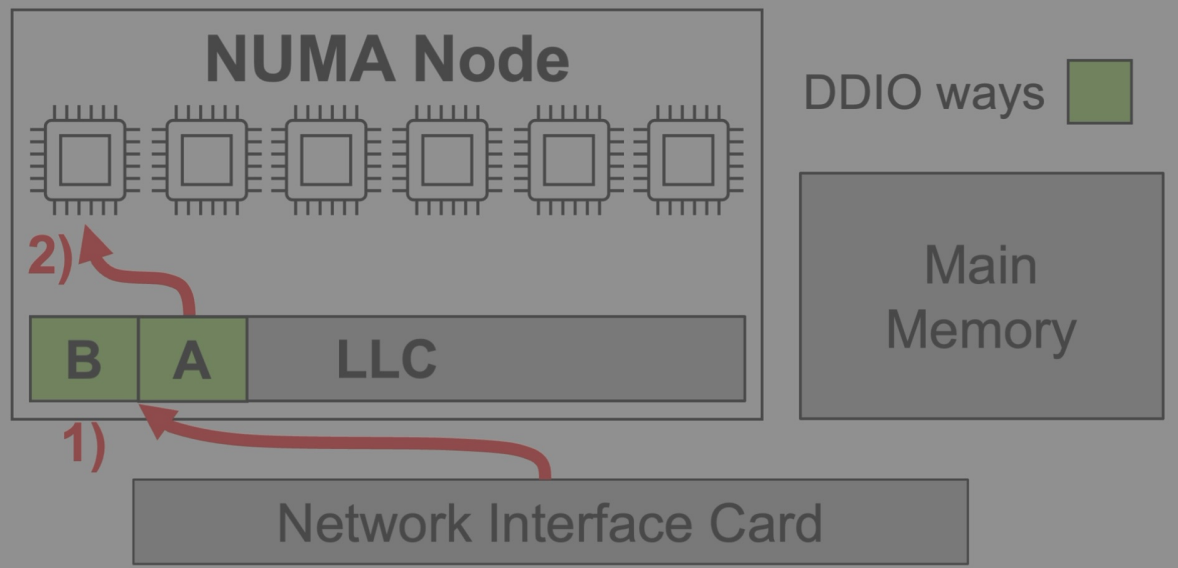
Avoid memory access latency



Case 2: High Traffic

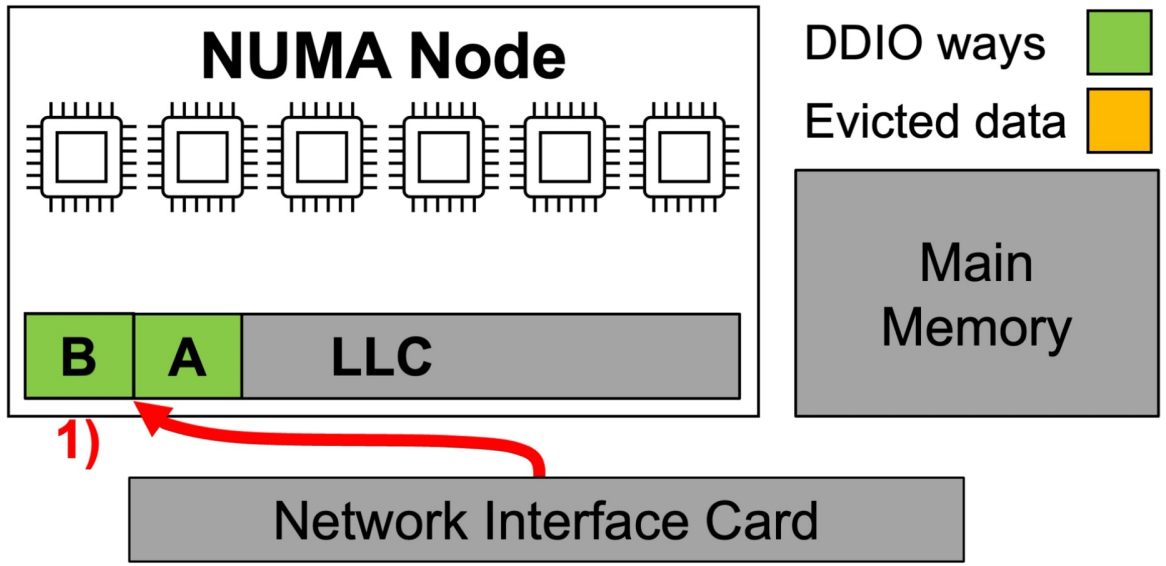
Incur memory access latency

Leaky DMA Problem



Case 1: Low Traffic

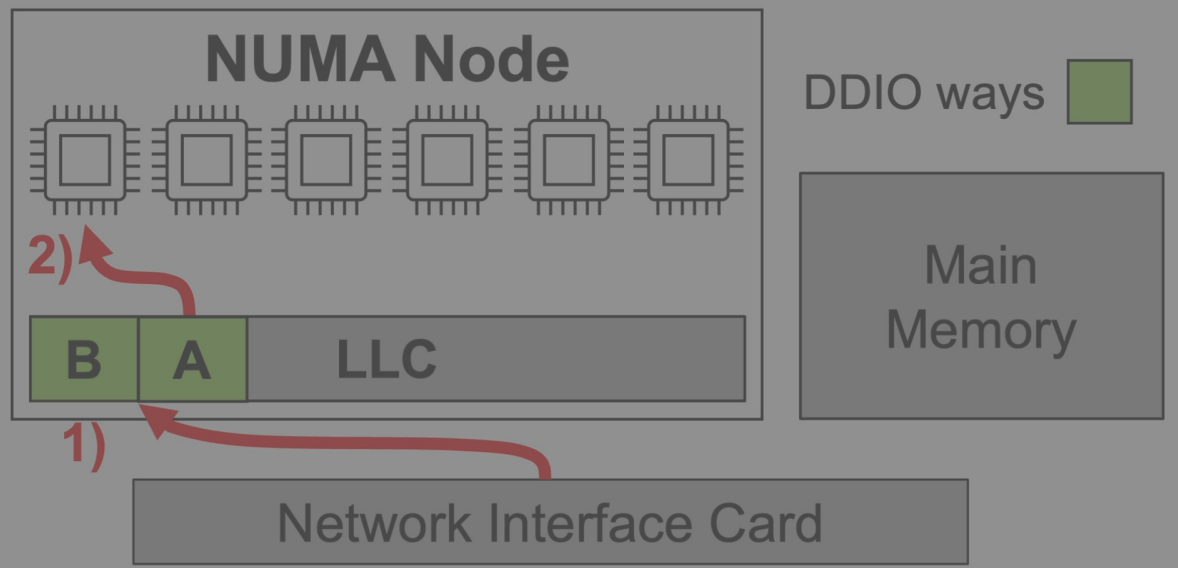
Avoid memory access latency



Case 2: High Traffic

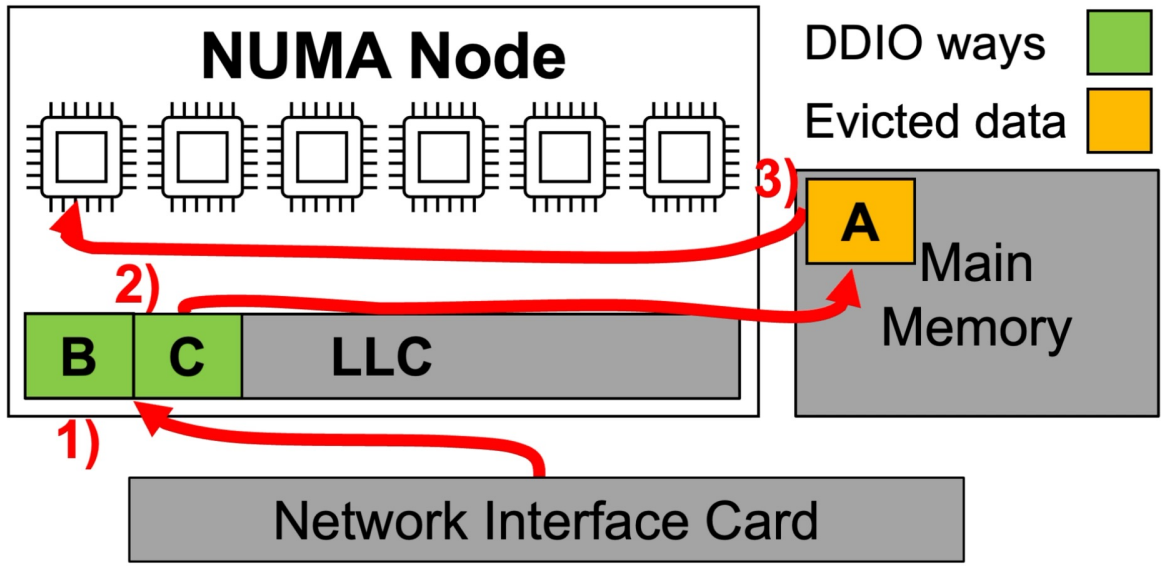
Incur memory access latency

Leaky DMA Problem



Case 1: Low Traffic

Avoid memory access latency



Case 2: High Traffic

Incur memory access latency

Problem: Leaky DMA reduces DDIO benefits

Prior work

1. Rx Ring Throttling or Sharing

- ResQ (NSDI′18)
- ShRing (OSDI′23)

2. Expanding DDIO Ways

- Don't Forget the I/O When Allocating Your LLC (ISCA′21)

3. DDIO Performance Analysis

- Reexamining DCA (ATC′20)
- Understanding Host Network Stack Overheads (SIGCOMM′21)

Prior work

Focus on DDIO Capacity

1. Rx Ring Throttling or Sharing

- ResQ (NSDI′18)
- ShRing (OSDI′23)

2. Expanding DDIO Ways

- Don't Forget the I/O When Allocating Your LLC (ISCA′21)

3. DDIO Performance Analysis

- Reexamining DCA (ATC′20)
- Understanding Host Network Stack Overheads (SIGCOMM′21)

Prior work

Focus on DDIO Capacity

1. Rx Ring Throttling or Sharing
- ResQ (NSDI′18)
- ShRing (OSDI′23)

2. Expanding DDIO Ways
- Don't Forget the I/O When Allocating Your LLC (ISCA′21)

3. DDIO Performance Analysis
- Reexamining DCA (ATC′20)
- Understanding Host Network Stack Overheads (SIGCOMM′21)

New findings from our work

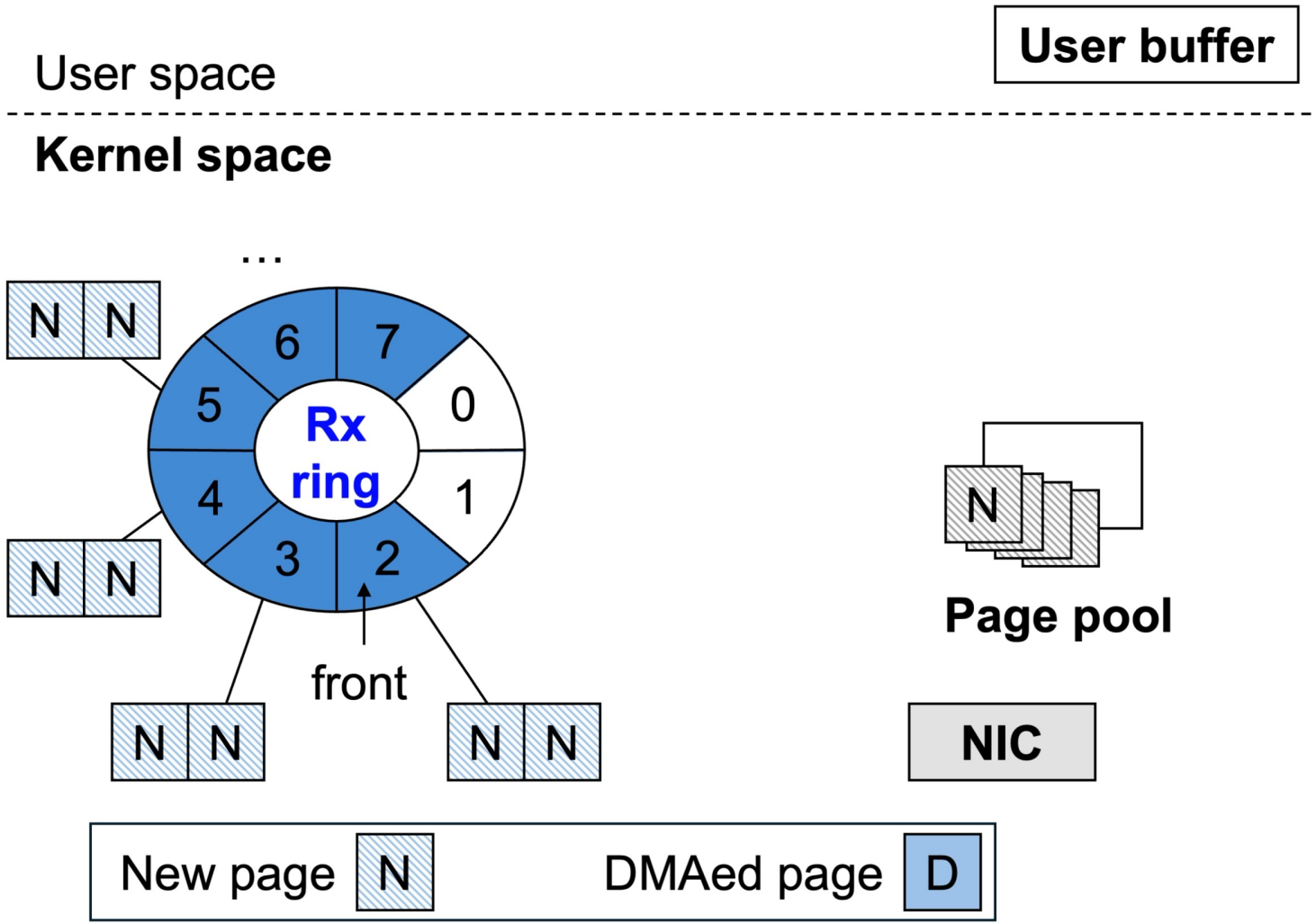
Finding 1: DDIO capacity is determined differently across cases

Finding 2: DDIO capacity alone cannot fully explain Leaky DMA

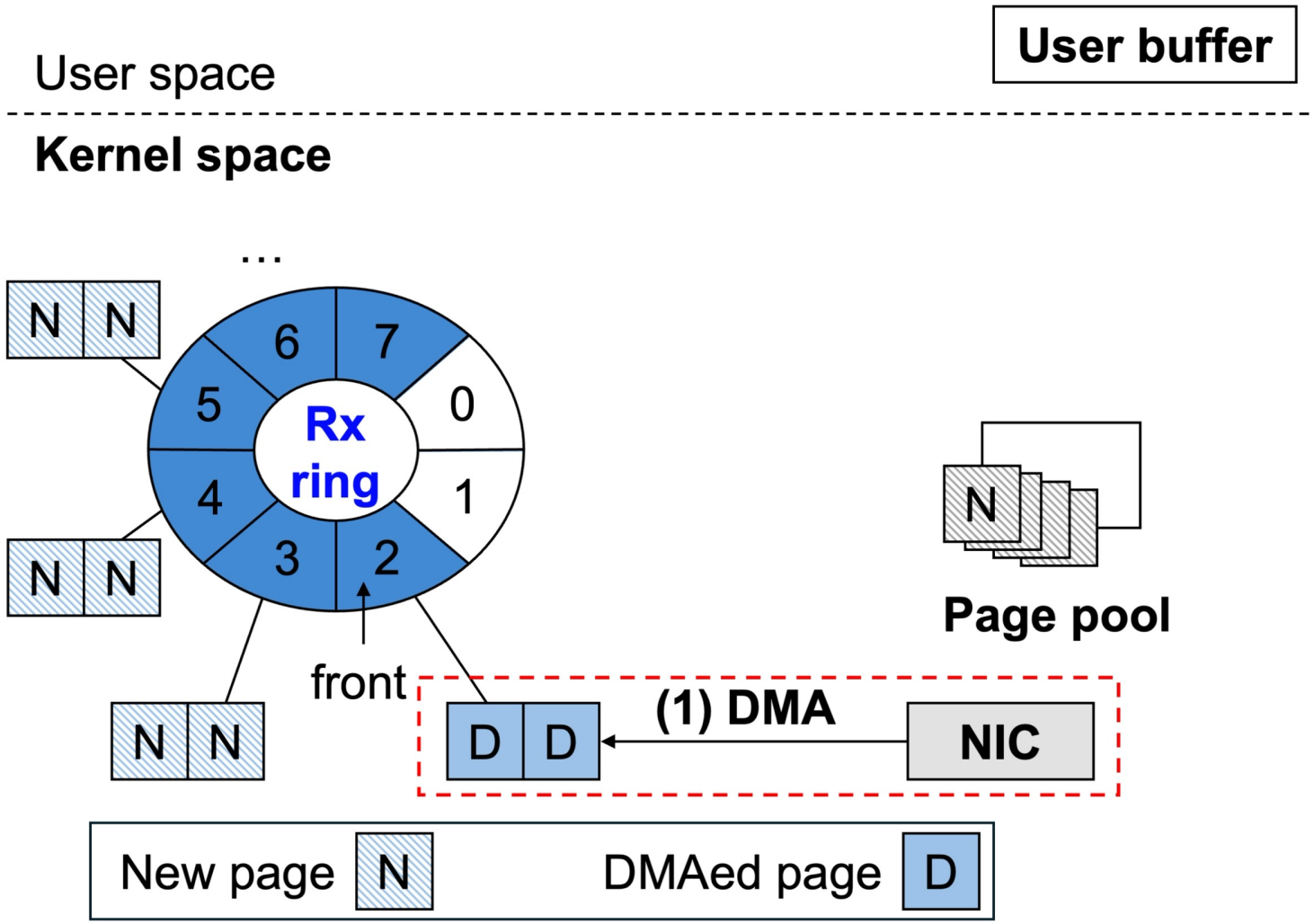
Index

- Introduction
- **Motivation**
- Sepia
- Evaluation

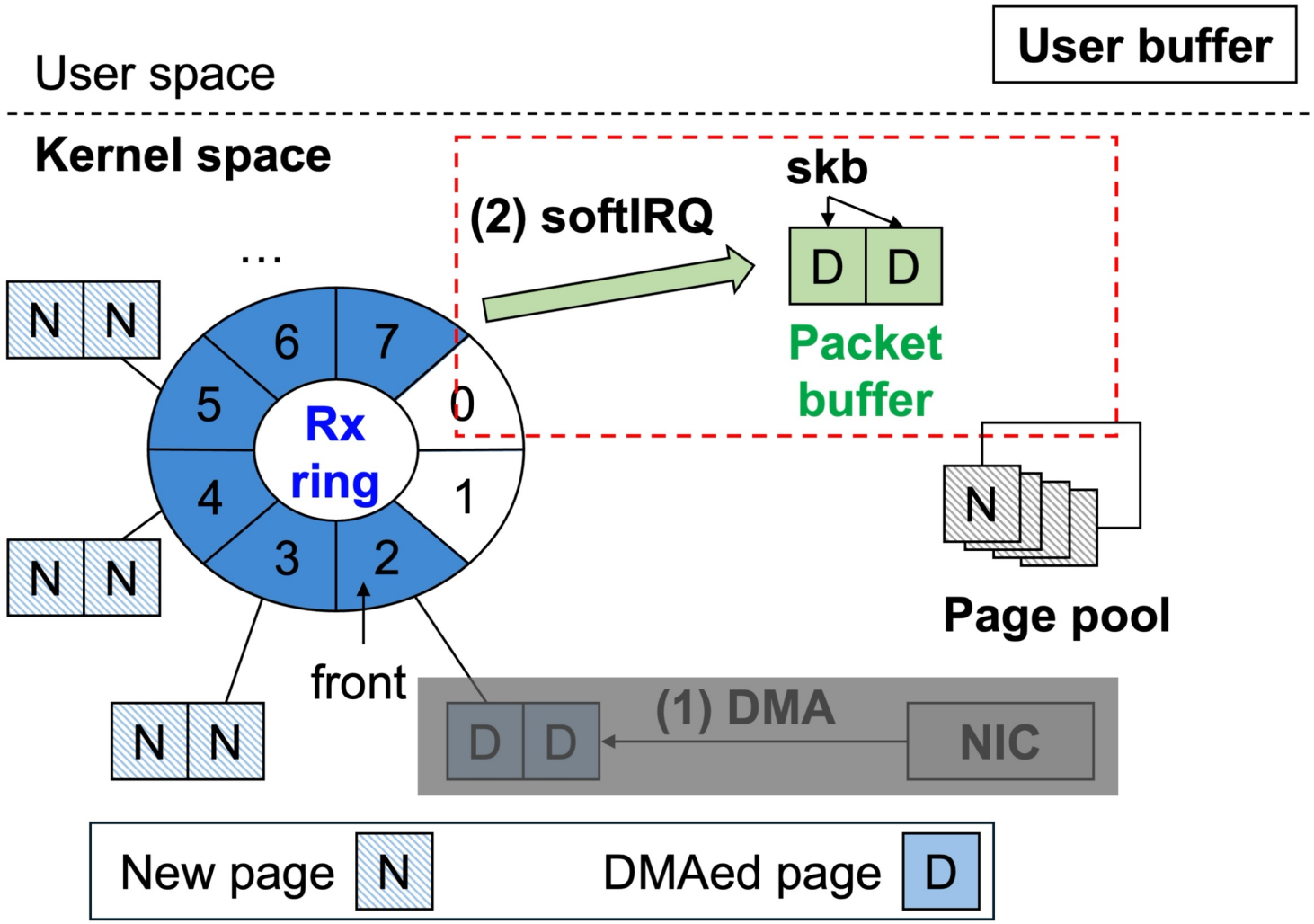
Background: Page Working Set



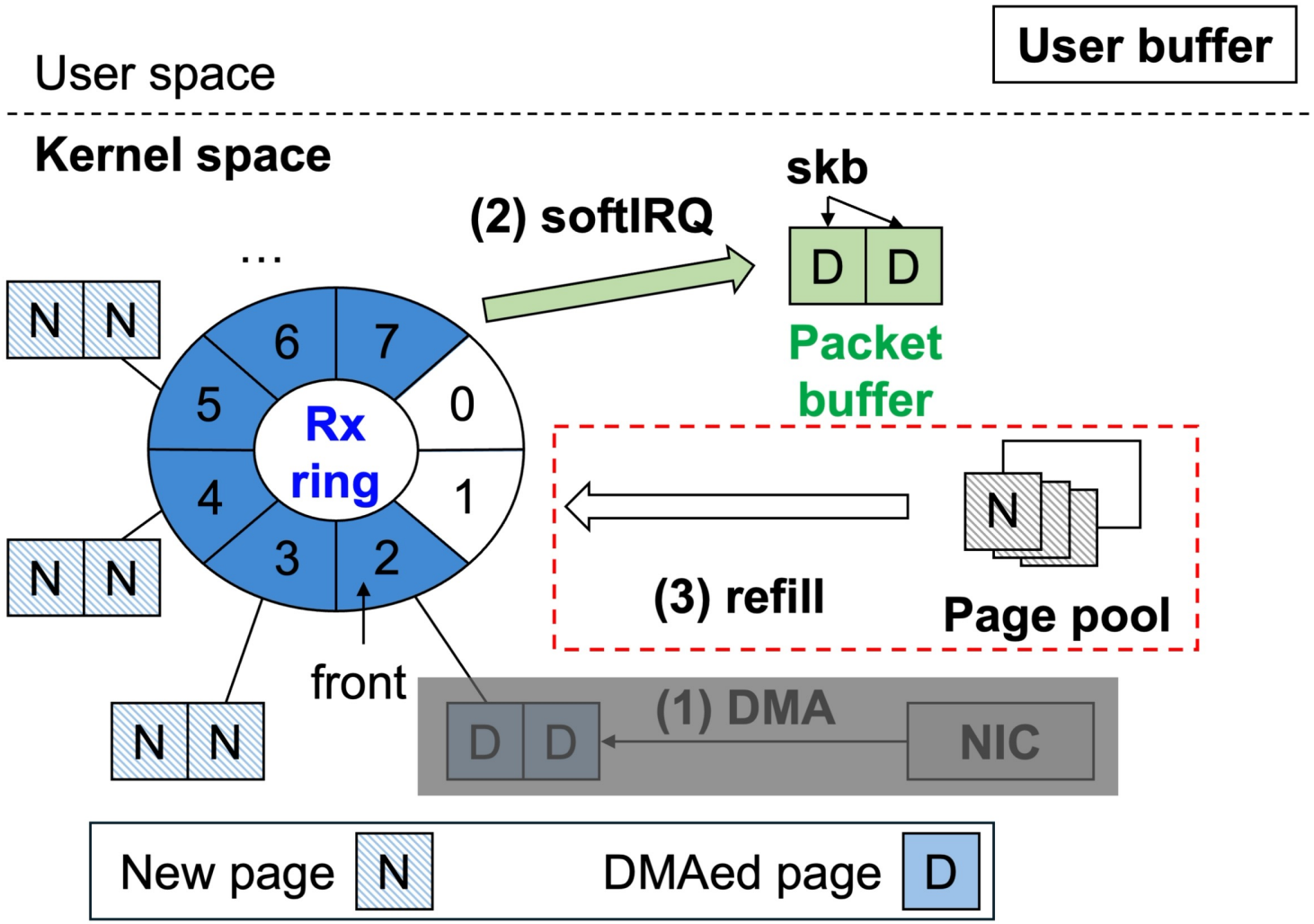
Background: Page Working Set



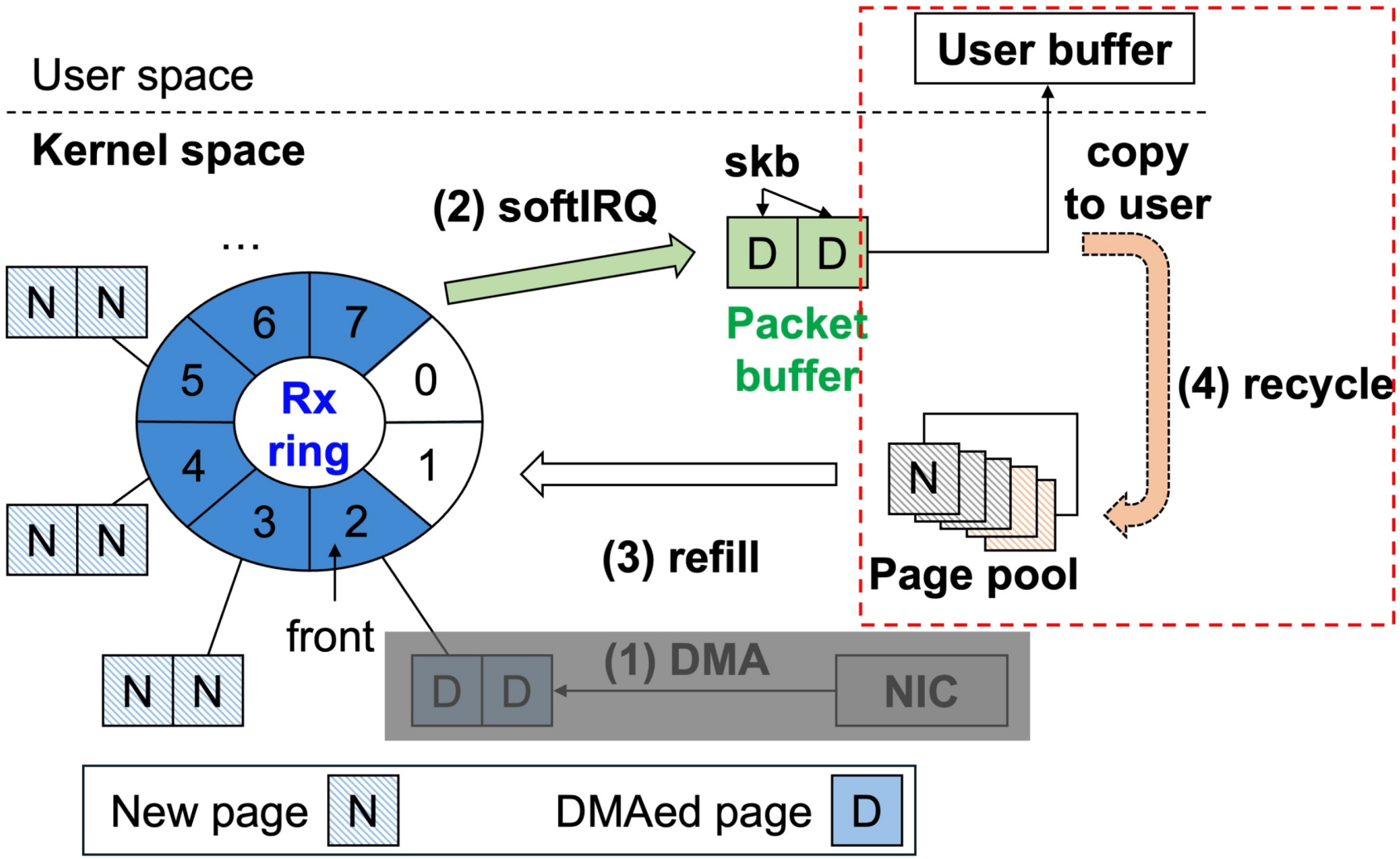
Background: Page Working Set



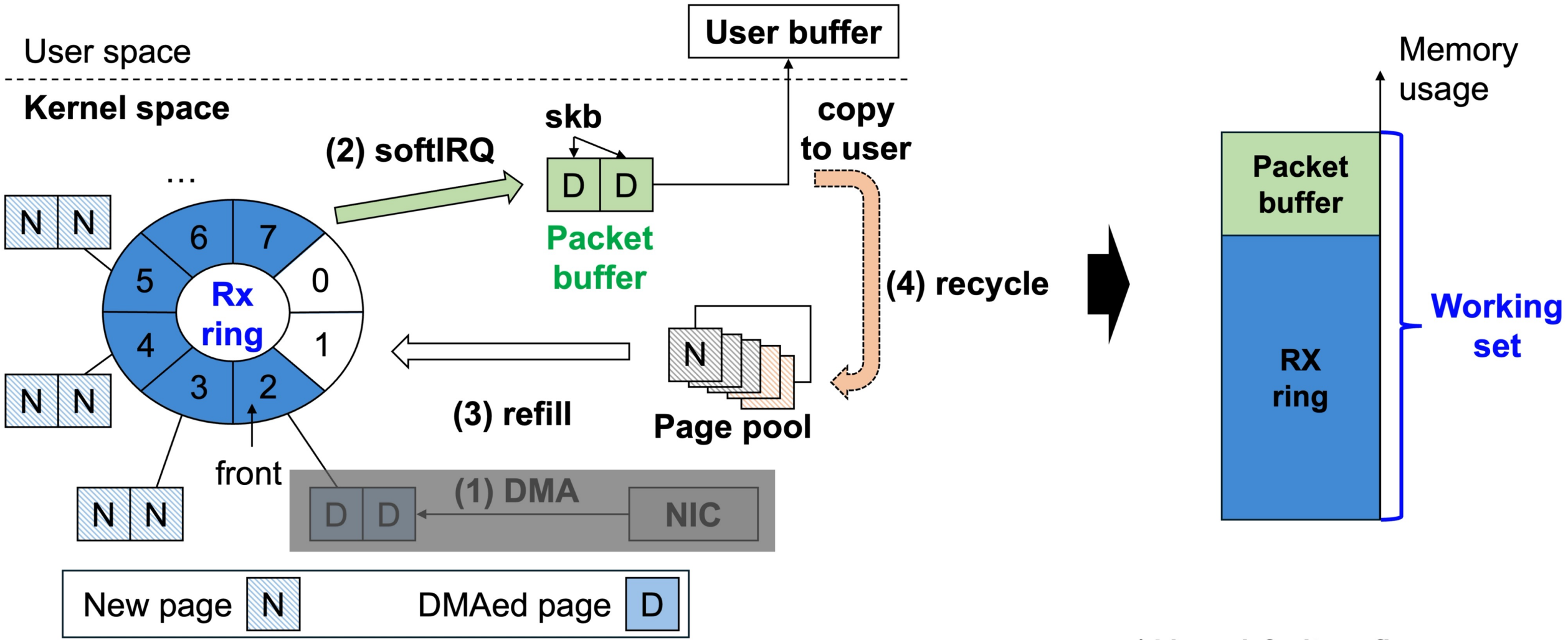
Background: Page Working Set



Background: Page Working Set

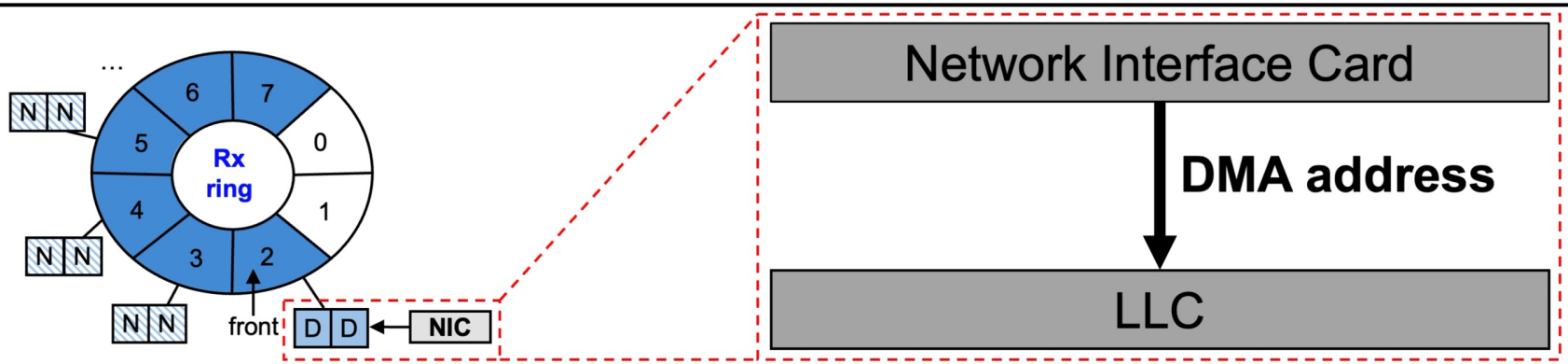


Background: Page Working Set

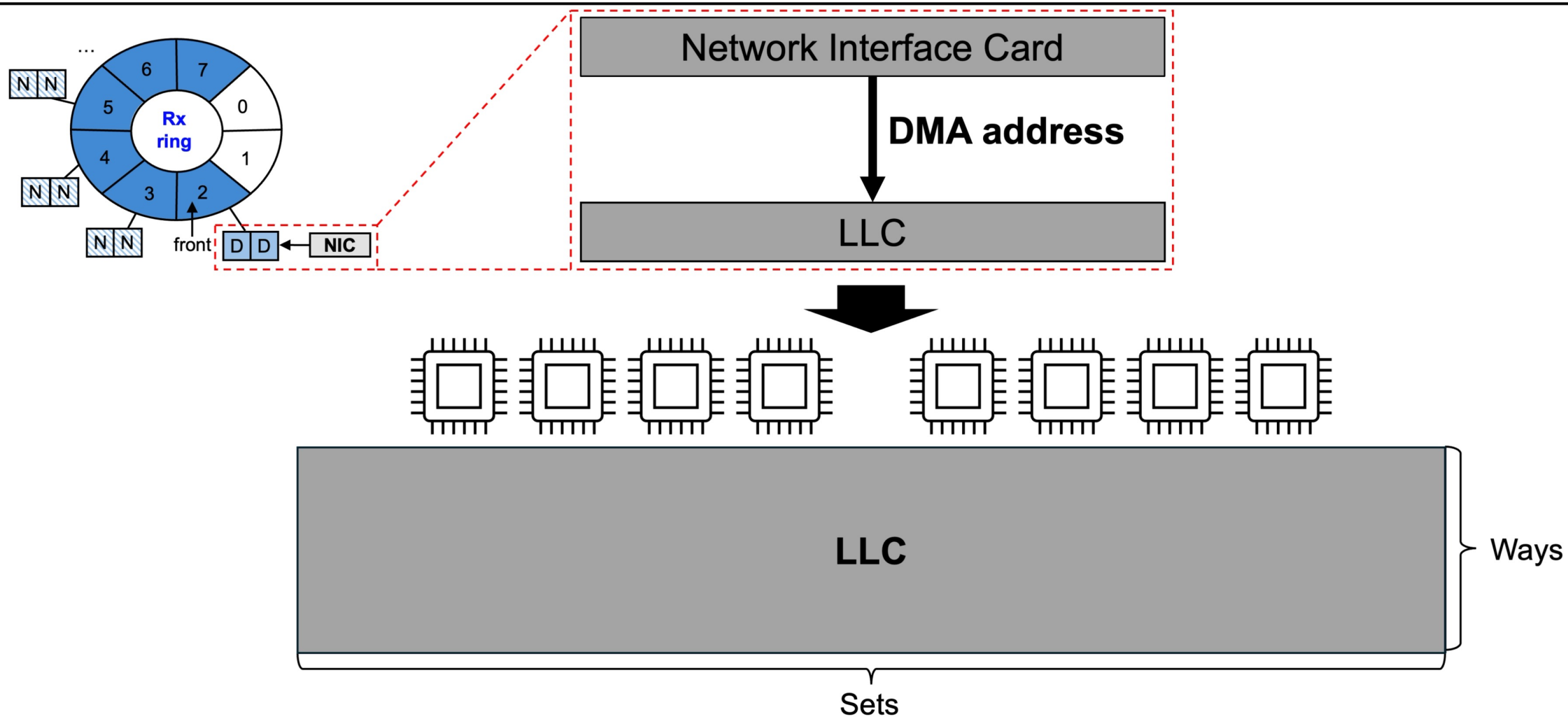


*** Linux default config**
Rx ring: 16MB, Packet buffer: ~6MB

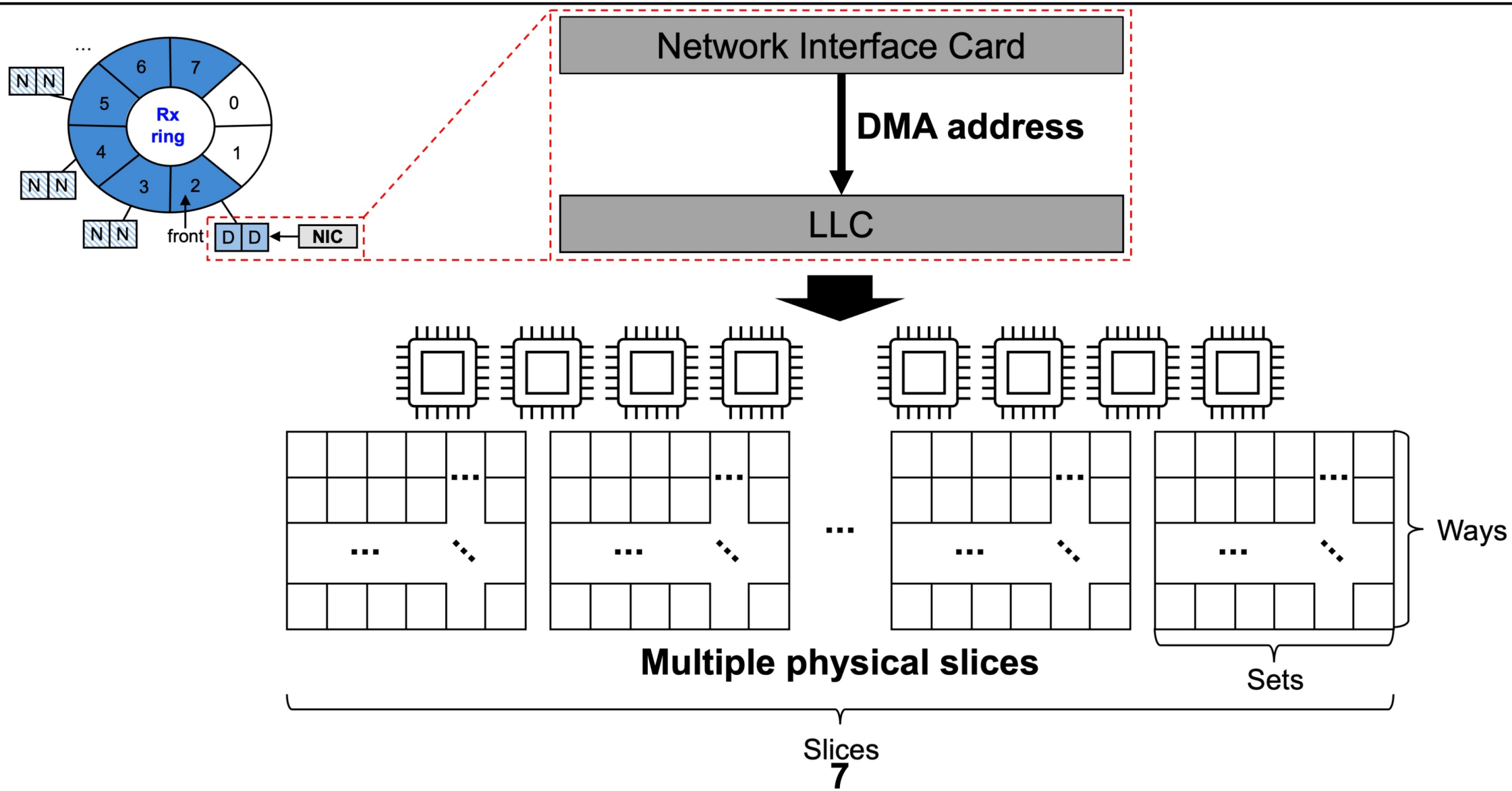
Background: DDIO with Modern LLC



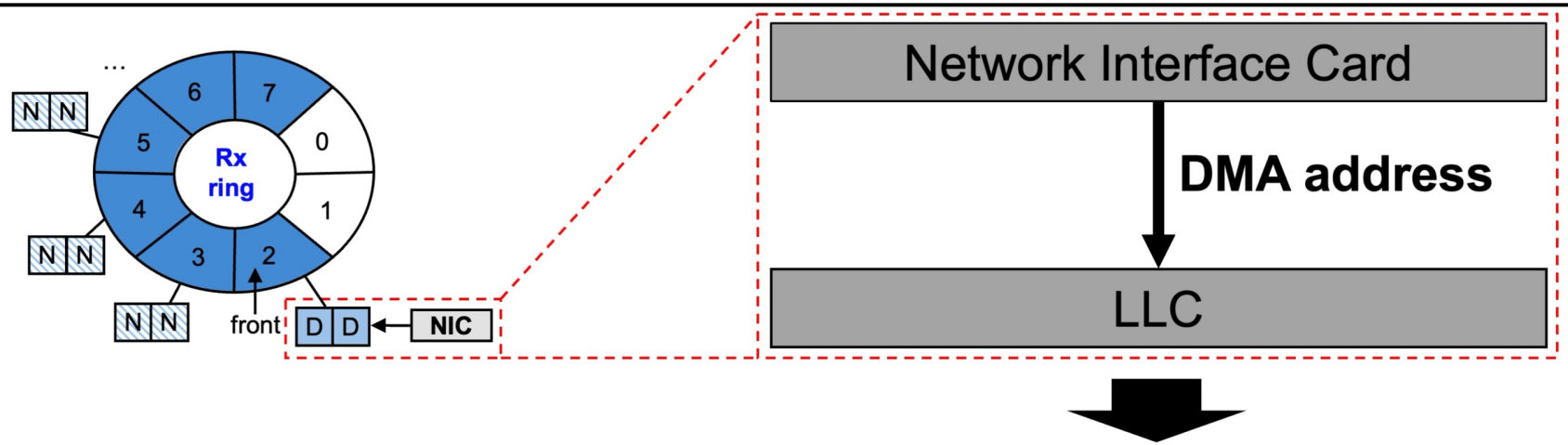
Background: DDIO with Modern LLC



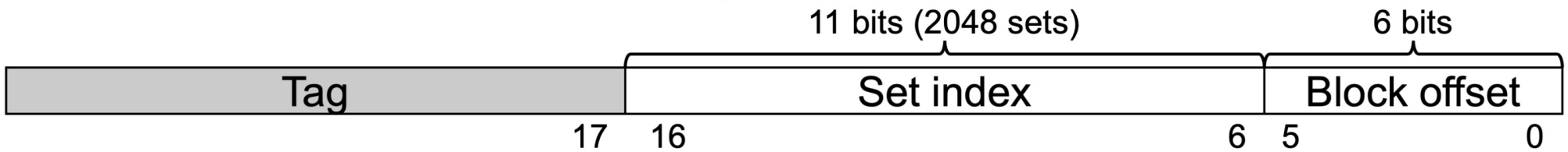
Background: DDIO with Modern LLC



Background: DDIO with Modern LLC



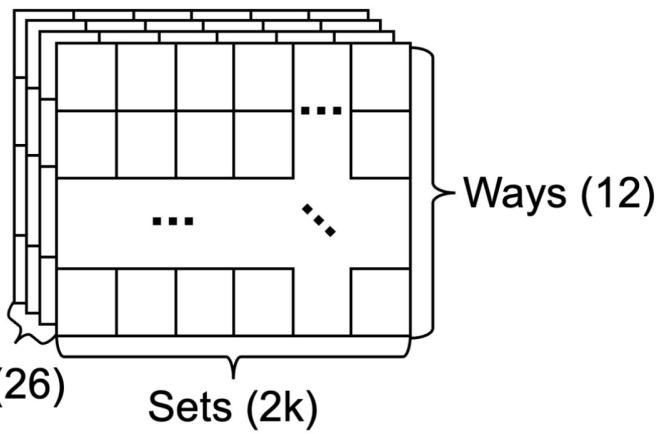
Address structure



DMA address

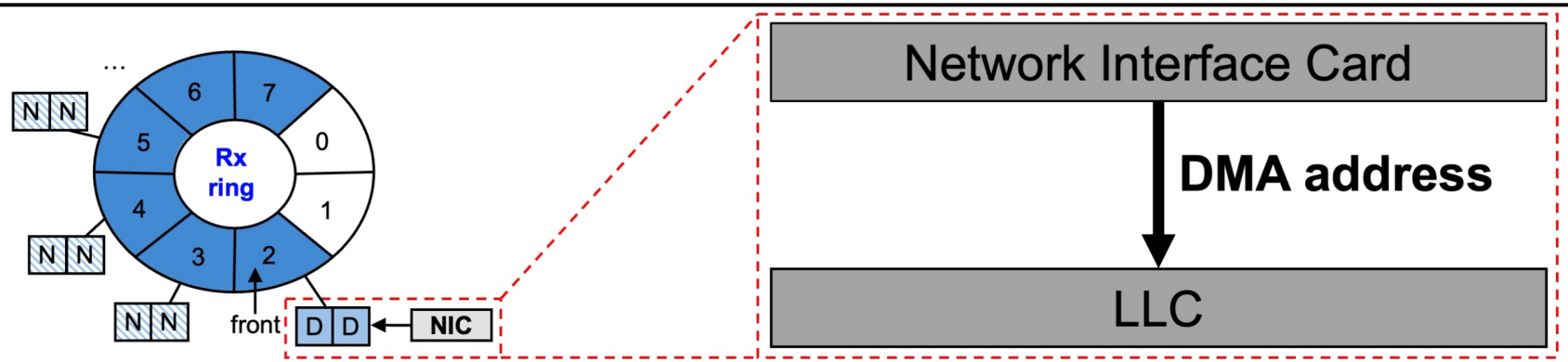
Hash function

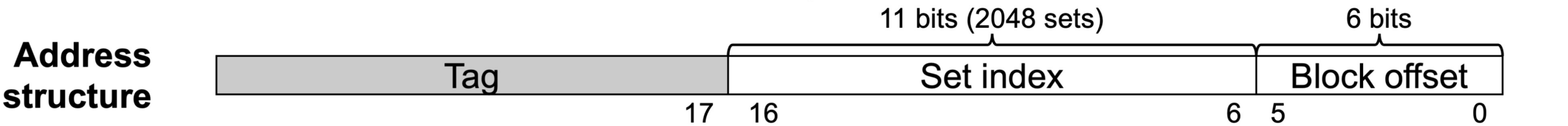
Bit mask

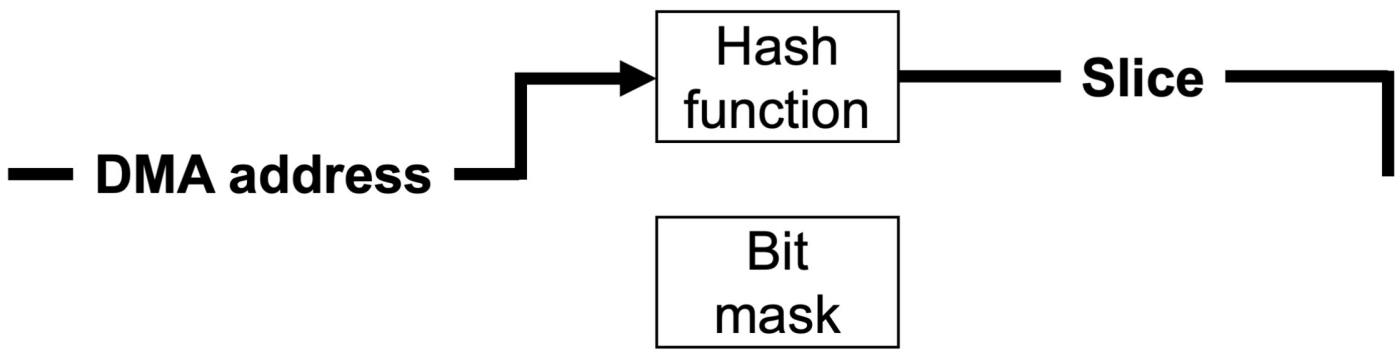


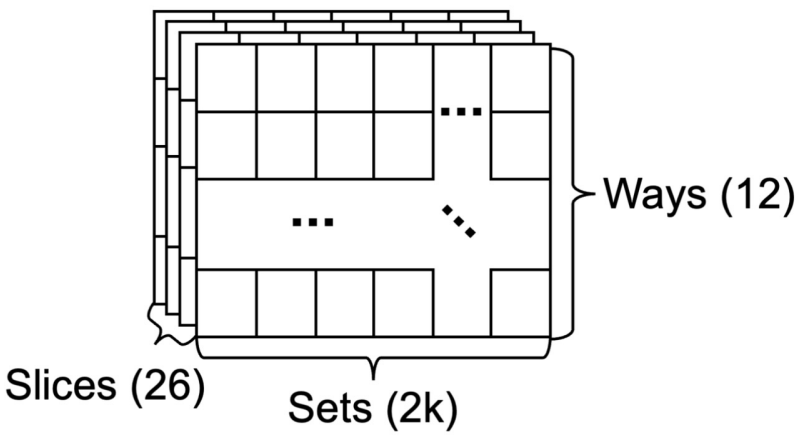
*** Xeon Gold 6354 LLC config (Our system)**
26 slices, (2048 sets, 12ways) per slices, 39 MB

Background: DDIO with Modern LLC



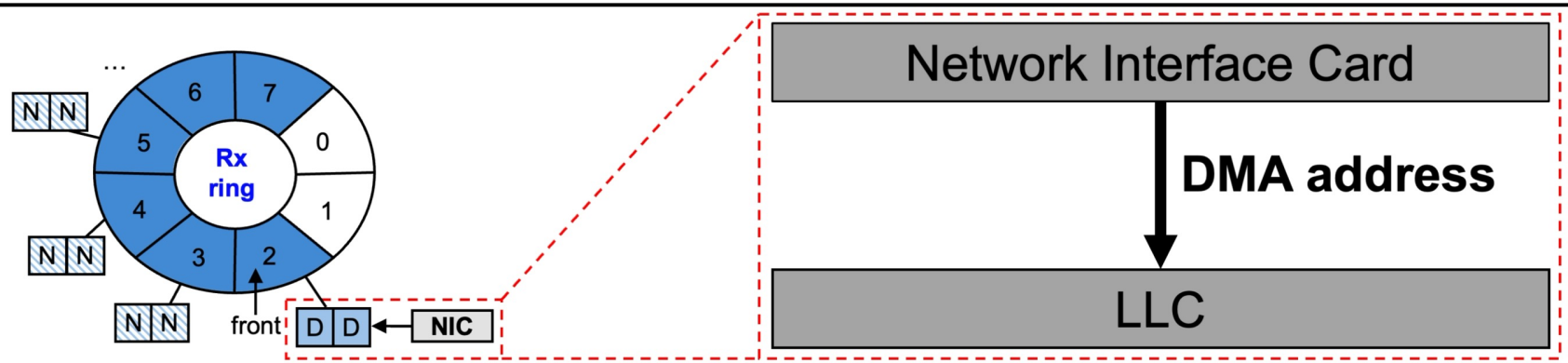


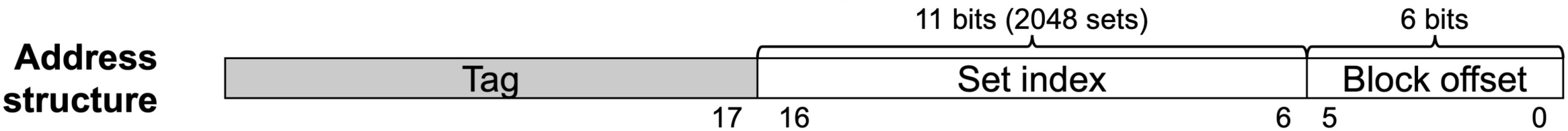


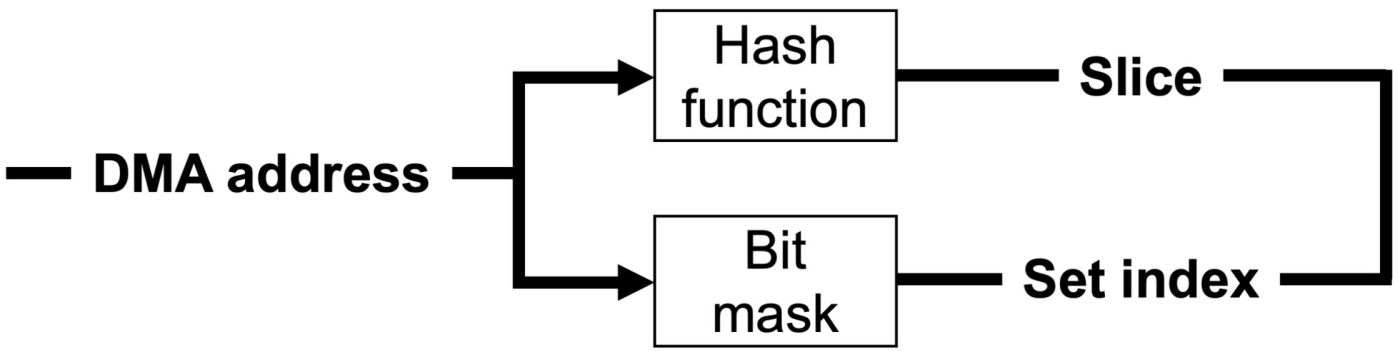


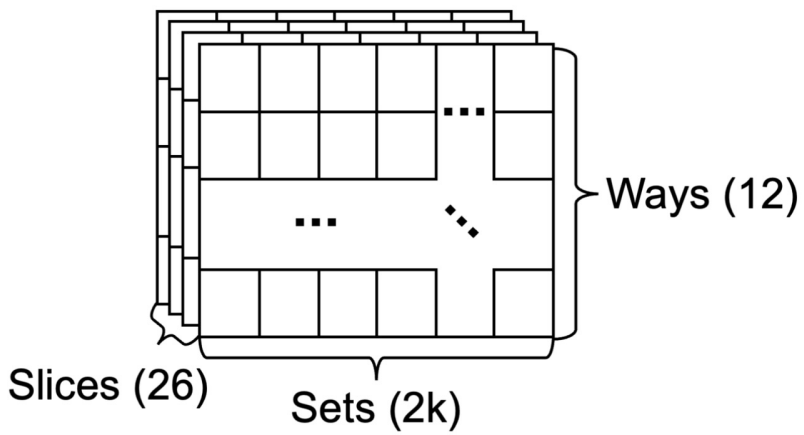
*** Xeon Gold 6354 LLC config (Our system)**
26 slices, (2048 sets, 12ways) per slices, 39 MB

Background: DDIO with Modern LLC



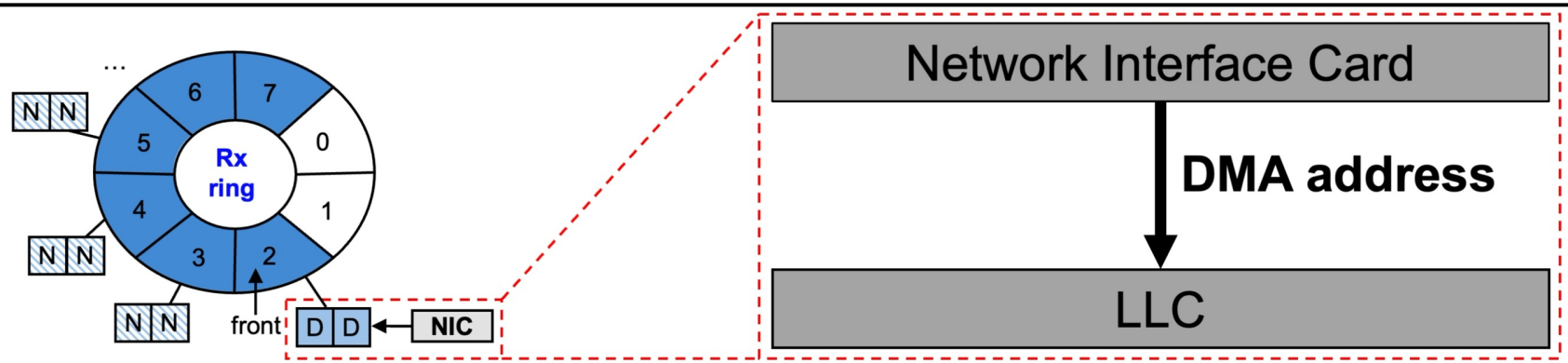


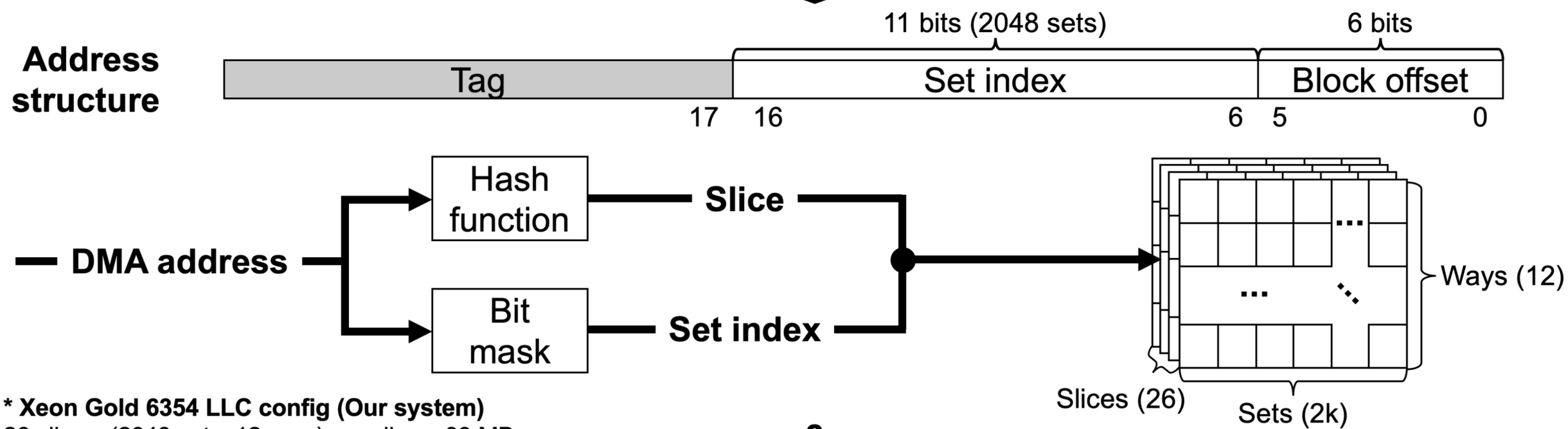




*** Xeon Gold 6354 LLC config (Our system)**
26 slices, (2048 sets, 12ways) per slices, 39 MB

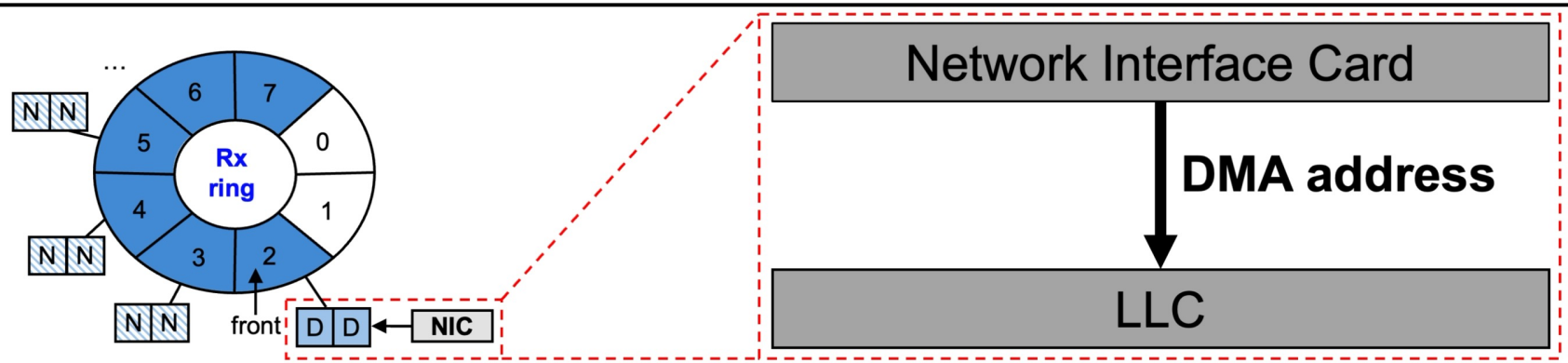
Background: DDIO with Modern LLC

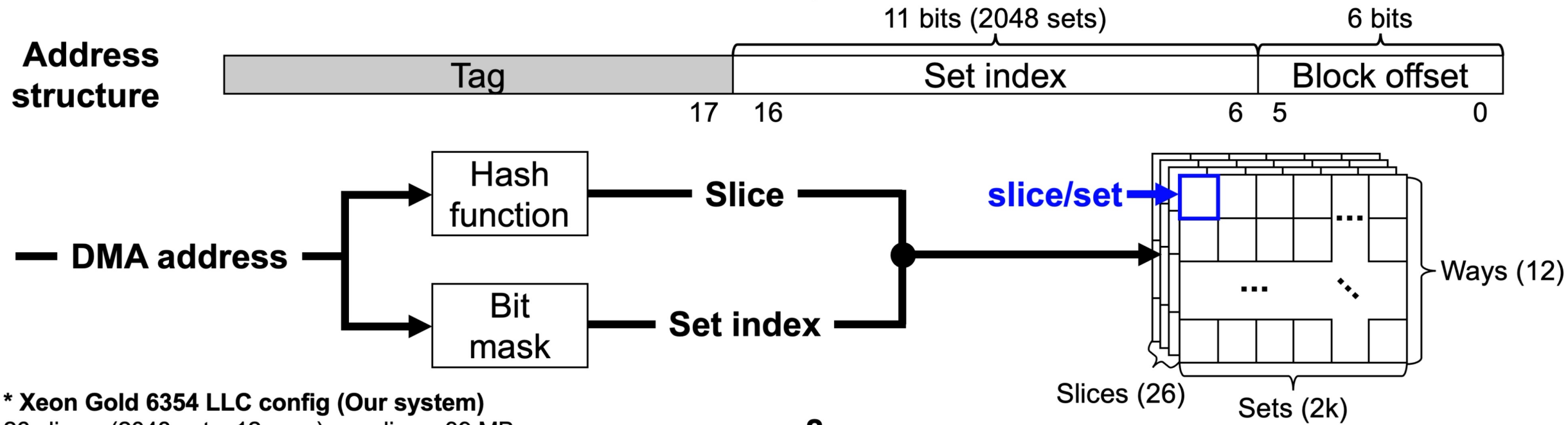




*** Xeon Gold 6354 LLC config (Our system)**
26 slices, (2048 sets, 12ways) per slices, 39 MB

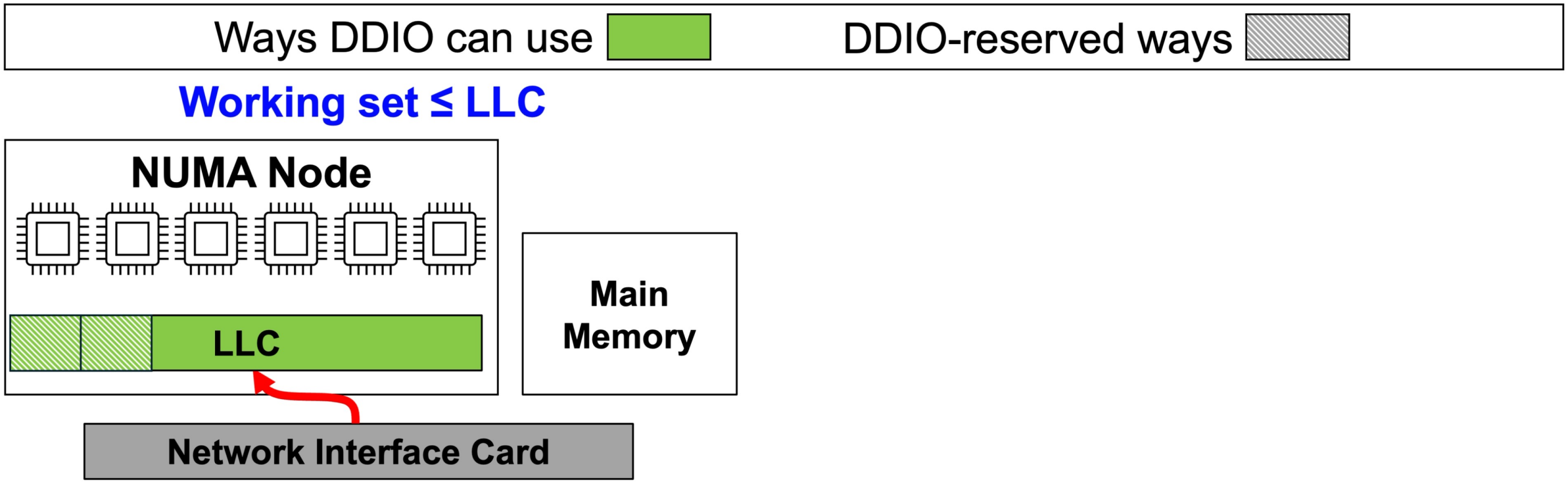
Background: DDIO with Modern LLC





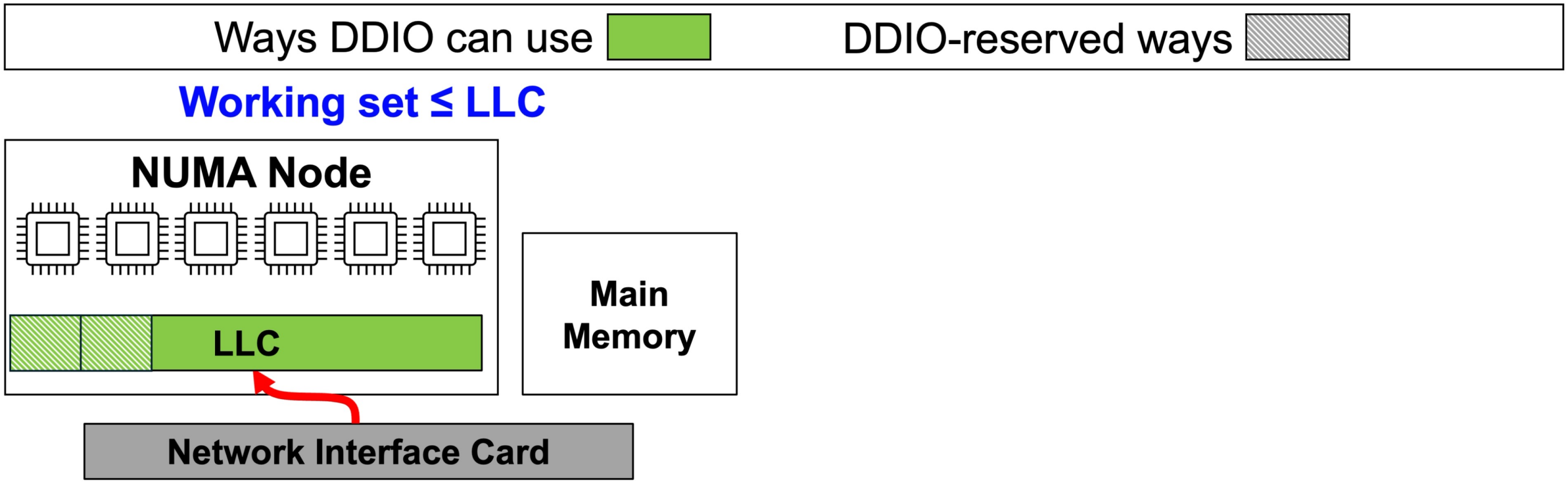
*** Xeon Gold 6354 LLC config (Our system)**
26 slices, (2048 sets, 12ways) per slices, 39 MB

Effective DDIO Capacity



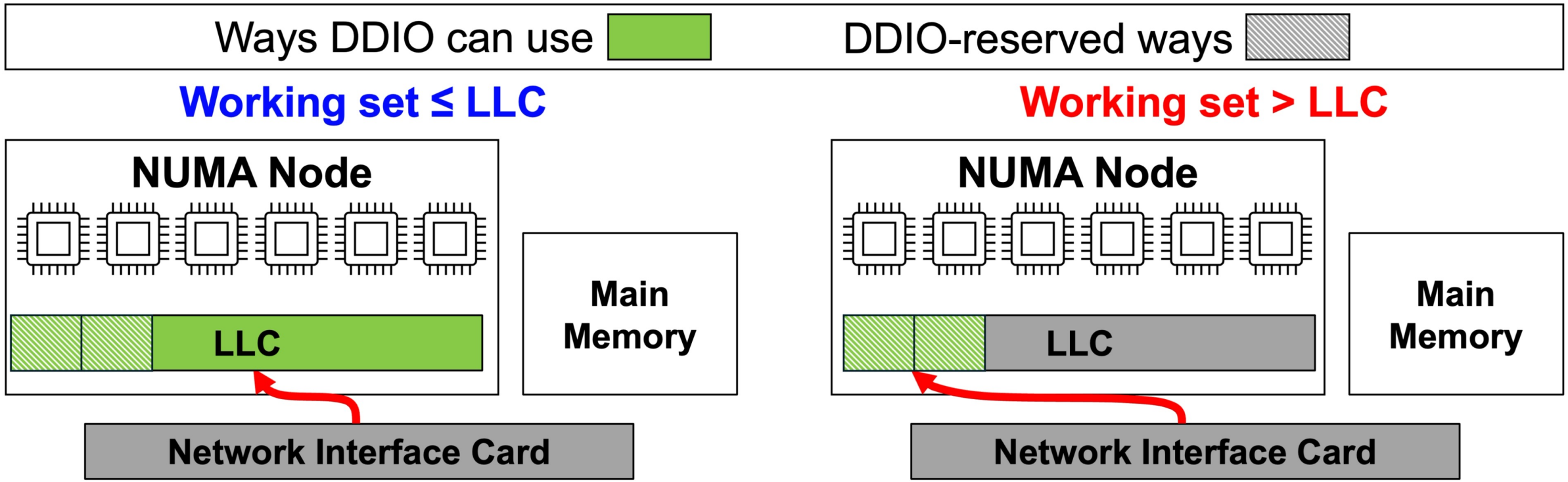
- Can update in **any LLC way**
- **Update** an existing cache entry
- **DDIO capacity → Total LLC**

Effective DDIO Capacity



- Can update in **any LLC way**
- **Update** an existing cache entry
- **DDIO capacity → Total LLC**

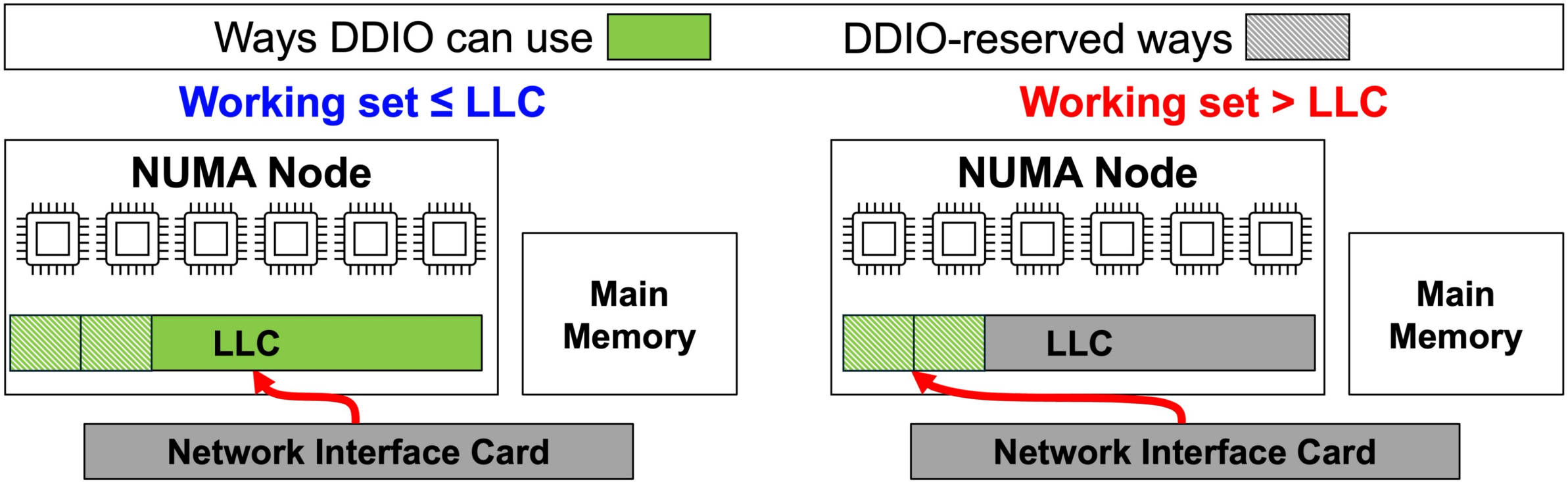
Effective DDIO Capacity



- Can update in **any LLC way**
- **Update** an existing cache entry
- **DDIO capacity → Total LLC**

- Restricted to **DDIO-reserved ways**
- **Allocate** a new cache entry
- **DDIO capacity → DDIO ways**

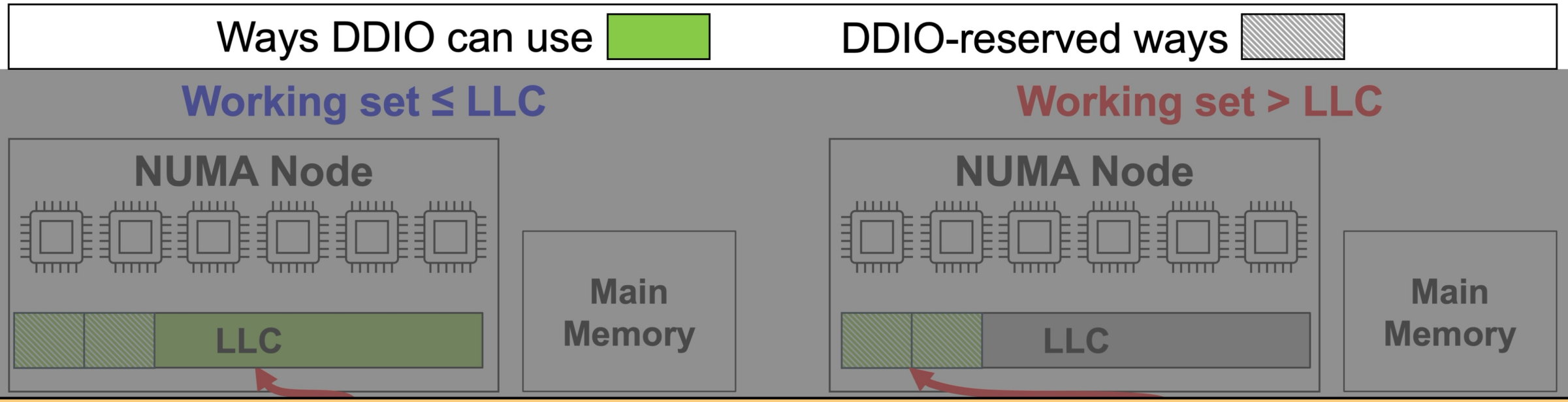
Effective DDIO Capacity



- Can update in **any LLC way**
- **Update** an existing cache entry
- **DDIO capacity → Total LLC**

- Restricted to **DDIO-reserved ways**
- **Allocate** a new cache entry
- **DDIO capacity → DDIO ways**

Effective DDIO Capacity



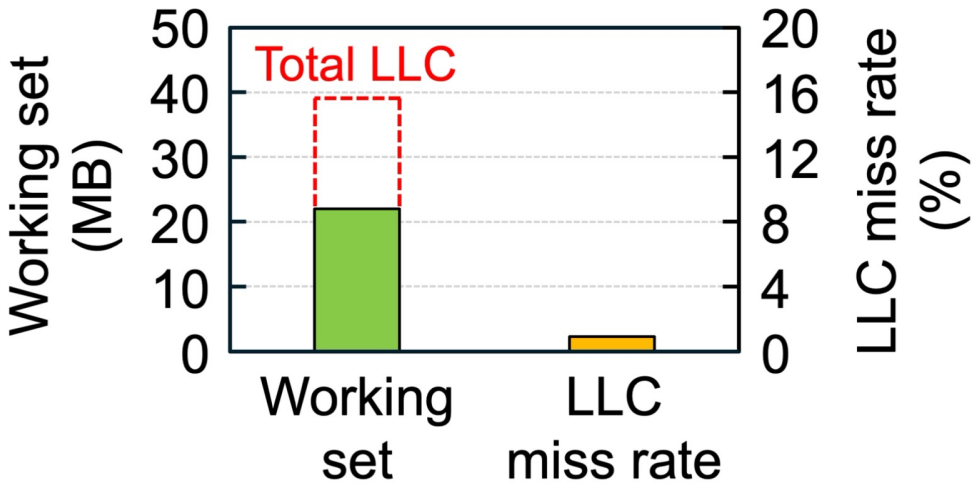
Observation 1:

Effective DDIO capacity depends on working set size;

- **Working set ≤ LLC** → Effective DDIO capacity = Total LLC
- **Working set > LLC** → Effective DDIO capacity = DDIO ways

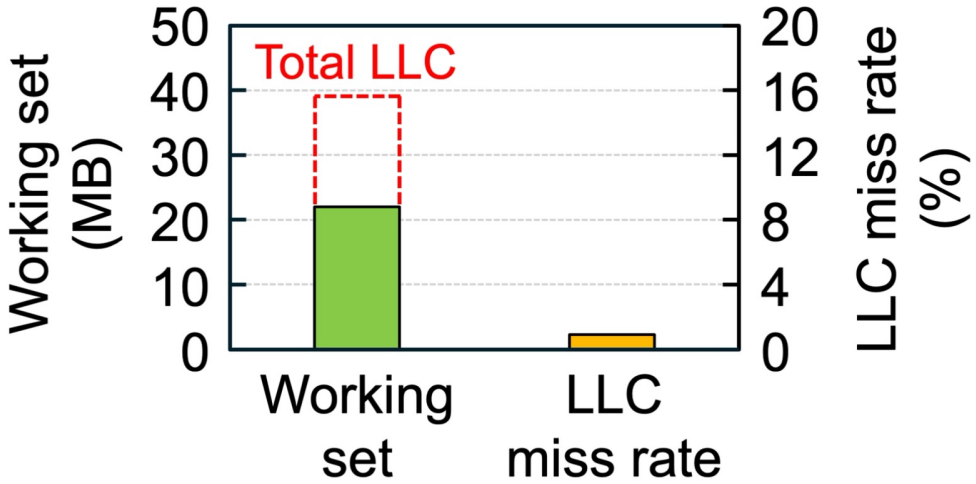
Is Fitting within the LLC Enough?

Case 1: 22MB working set



Is Fitting within the LLC Enough?

Case 1: 22MB working set

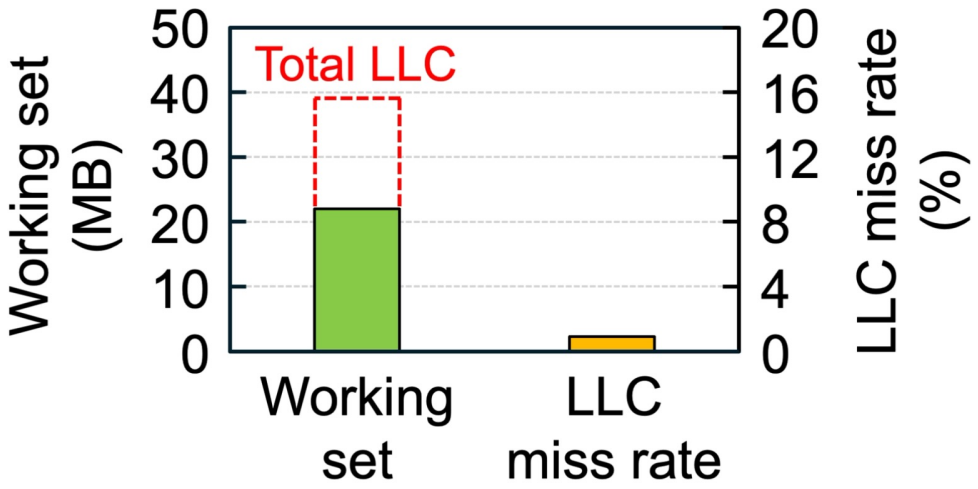


Case 2: 38MB working set



Is Fitting within the LLC Enough?

Case 1: 22MB working set



Case 2: 38MB working set



Even when the working sets fit in the LLC, miss rates may still increase

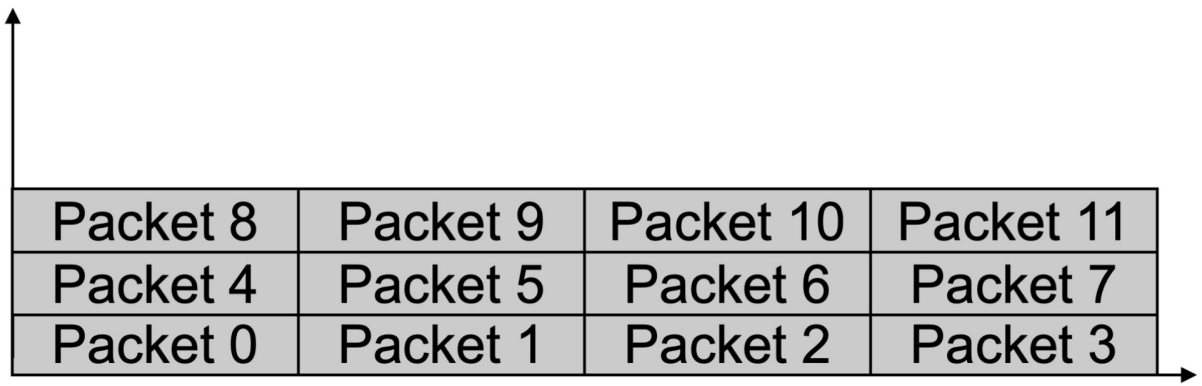
LLC Is Set-Associative

View Shift: From Capacity to Placement

- Packets do not fill arbitrary free space; they map to sets and ways

12-Packet Working Set Example:

Expected: Packets spread **evenly across the LLC**



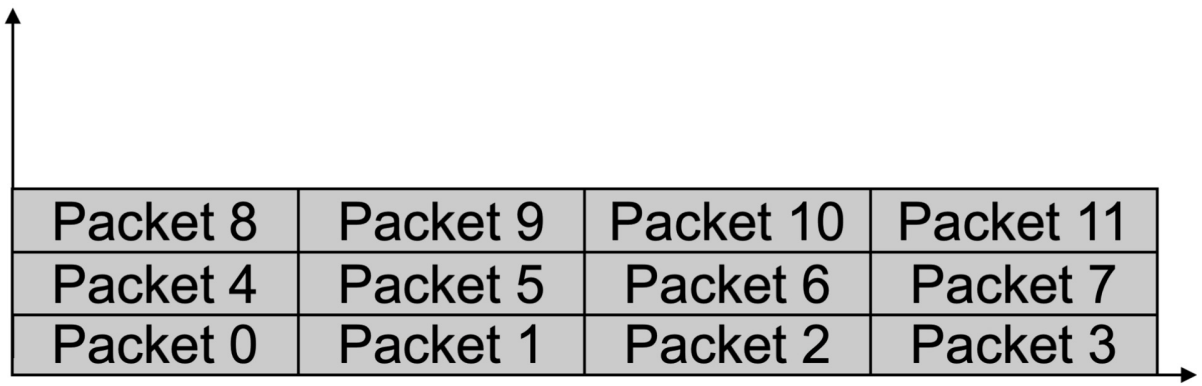
LLC Is Set-Associative

View Shift: From Capacity to Placement

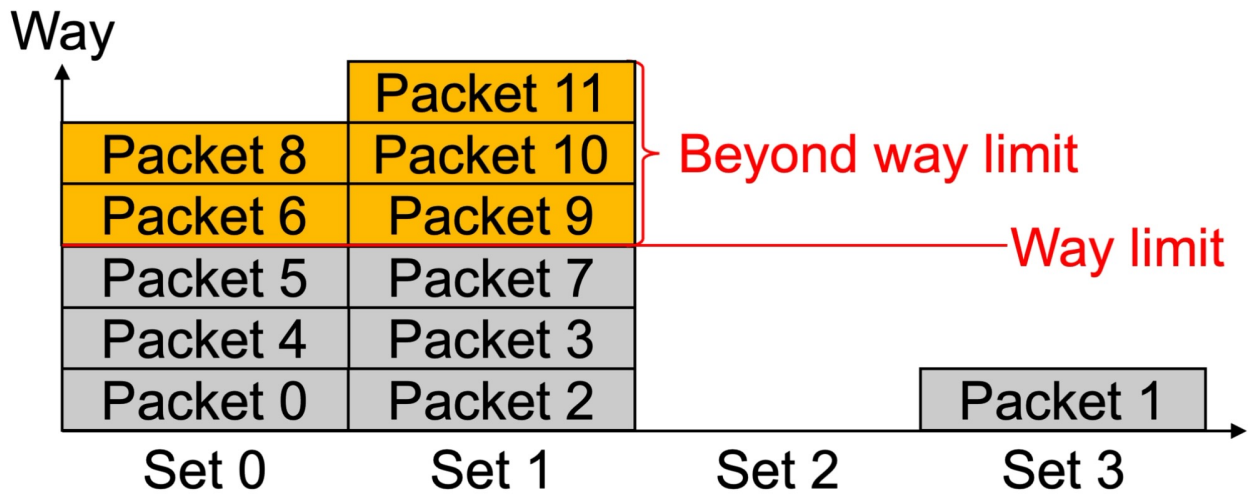
- Packets do not fill arbitrary free space; they map to sets and ways

12-Packet Working Set Example:

Expected: Packets spread **evenly across the LLC**



Actual: Packets map into **specific LLC location**



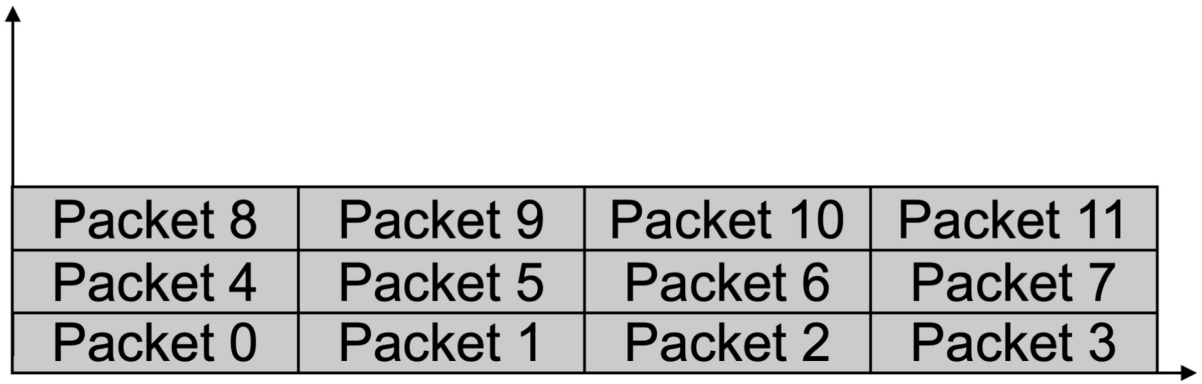
LLC Is Set-Associative

View Shift: From Capacity to Placement

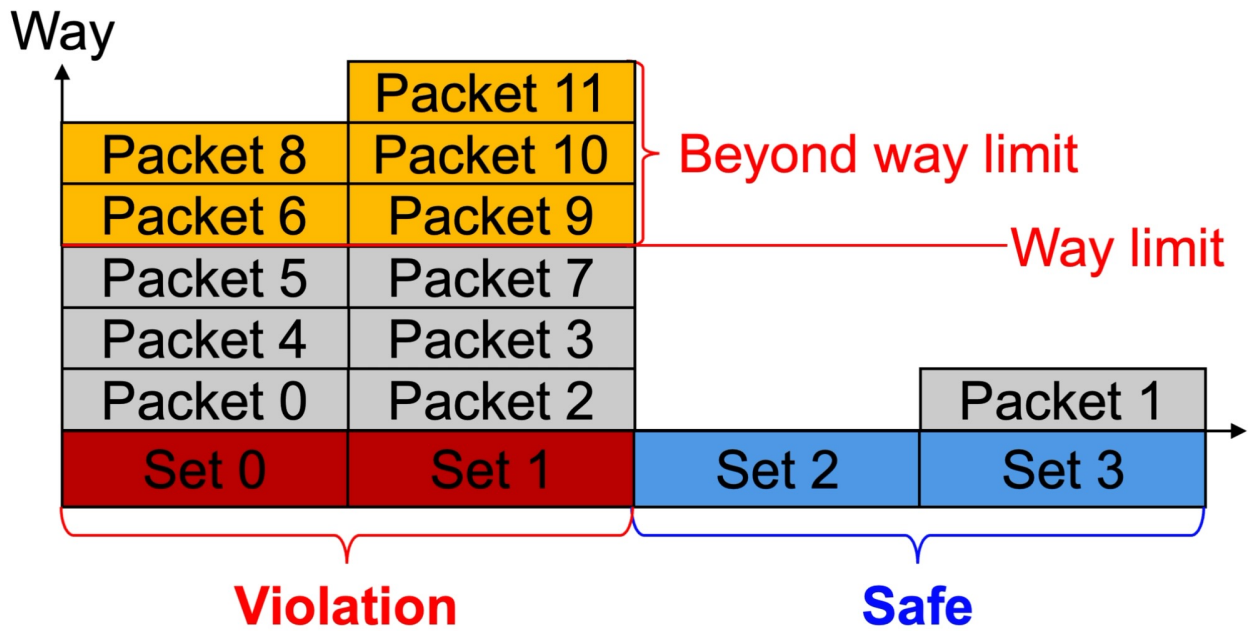
- Packets do not fill arbitrary free space; they map to sets and ways

12-Packet Working Set Example:

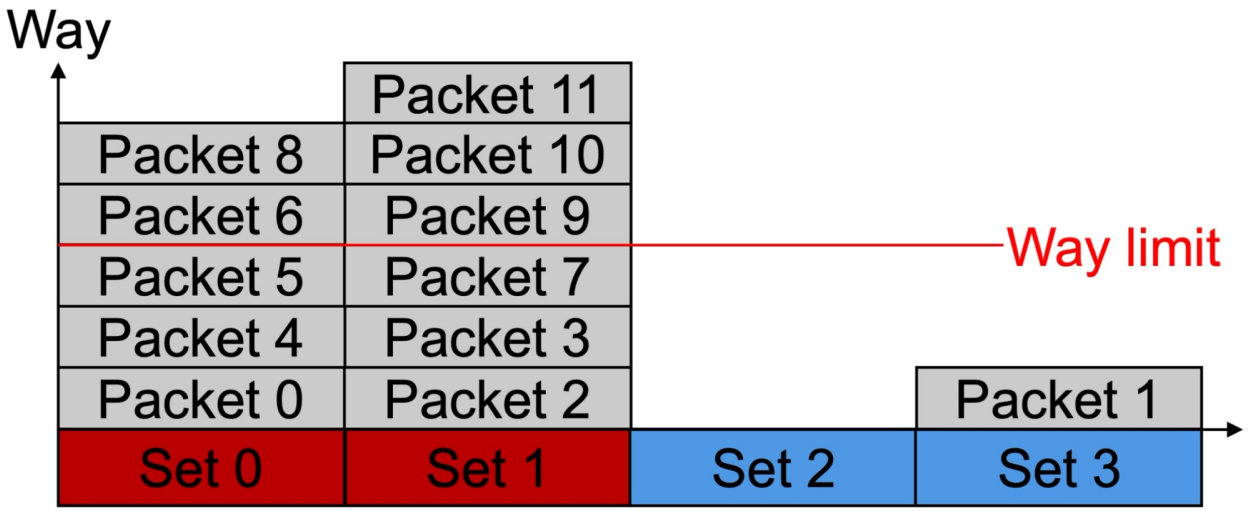
Expected: Packets spread **evenly across the LLC**

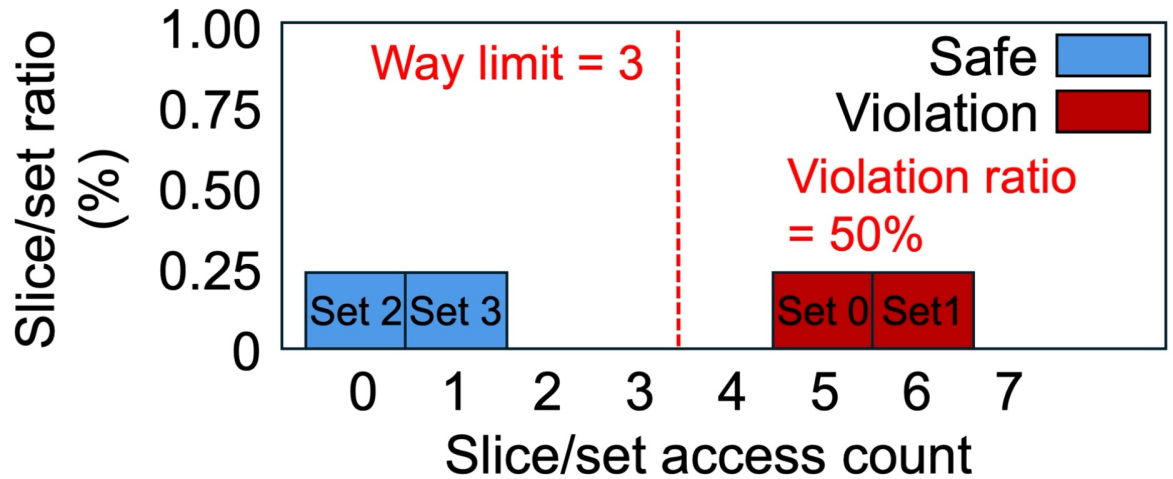


Actual: Packets map into **specific LLC location**

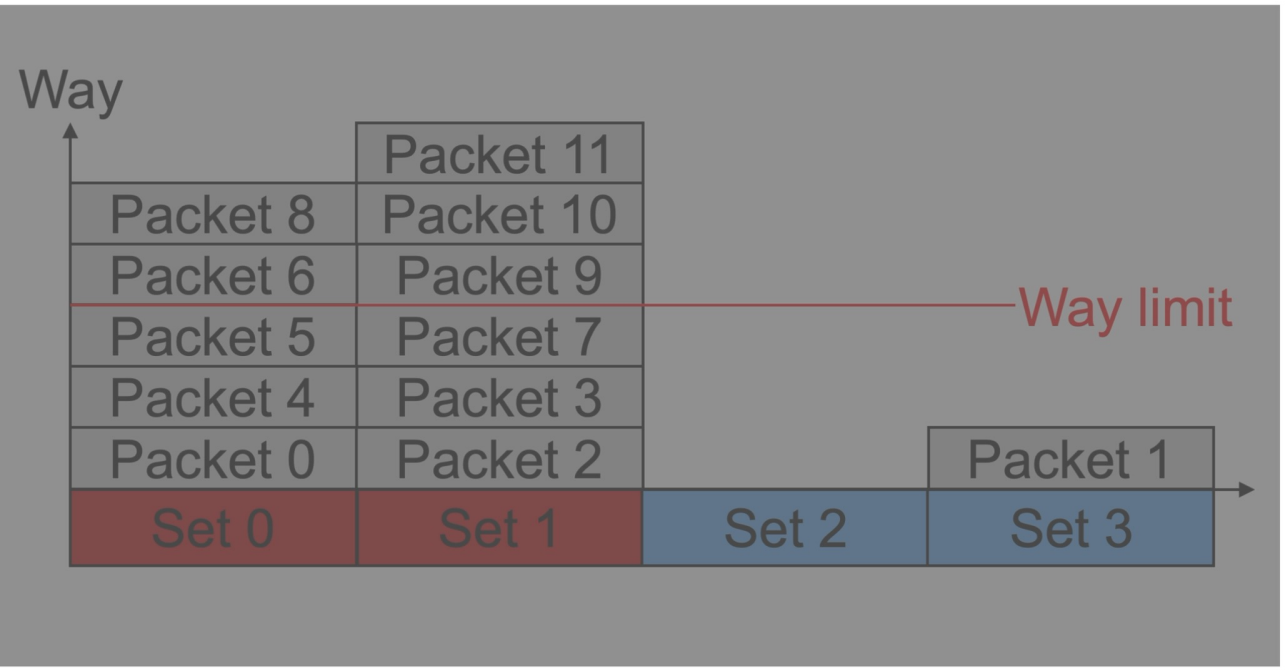


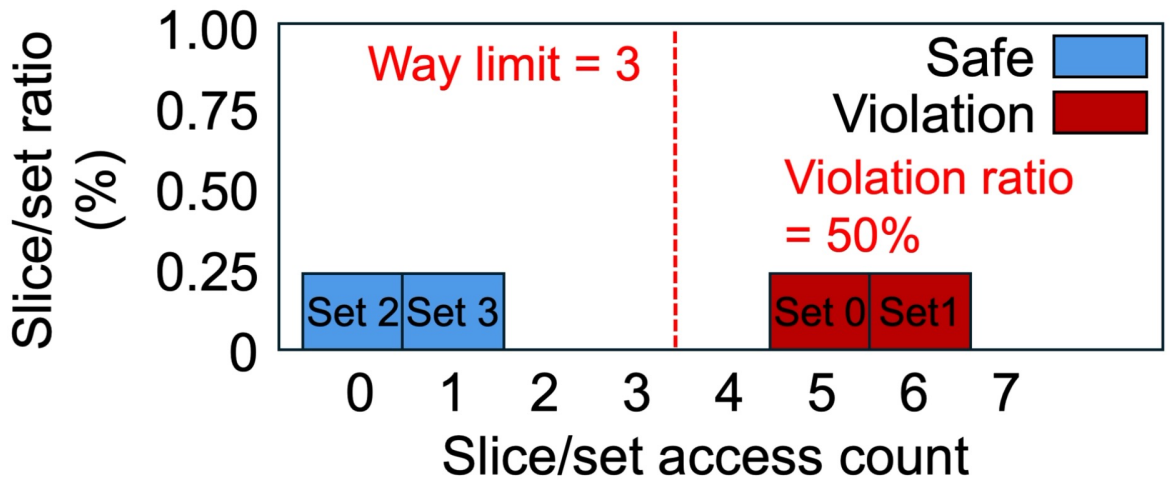
Set Conflicts Matter





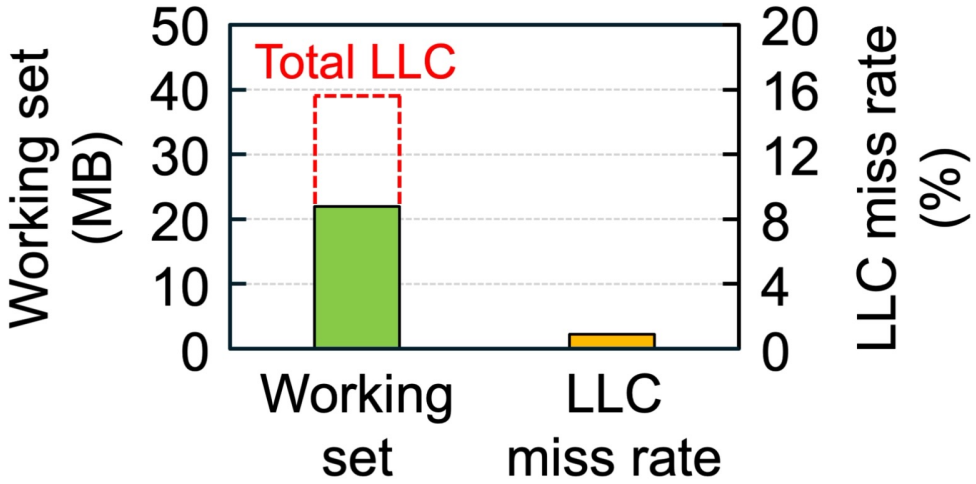
Set Conflicts Matter

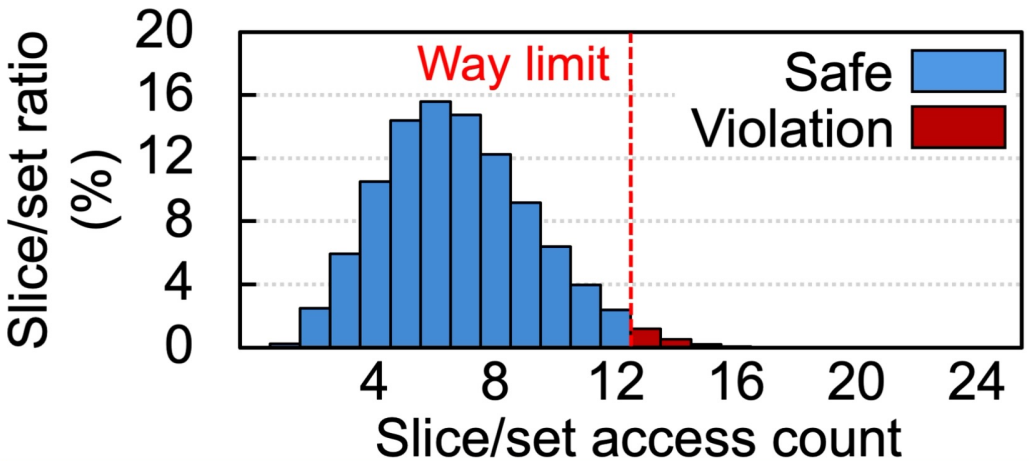




Set Conflicts Matter

Case 1: 22MB working set

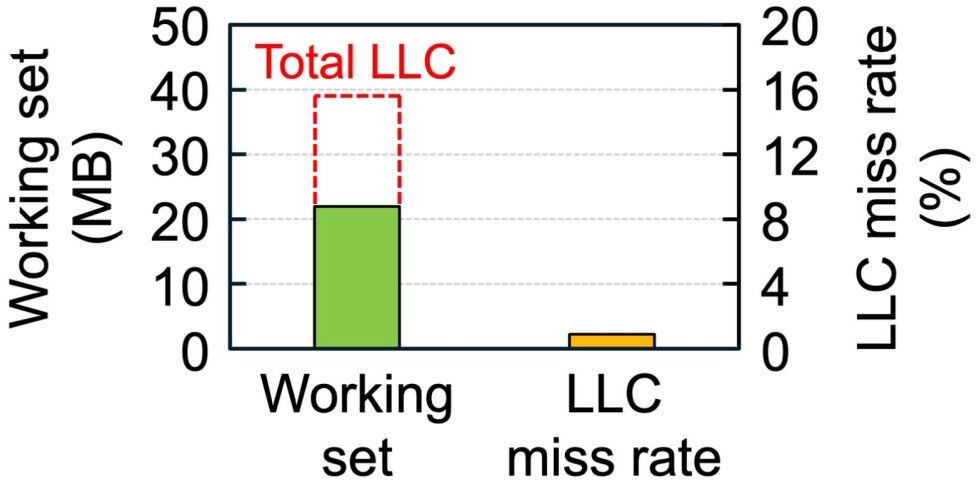


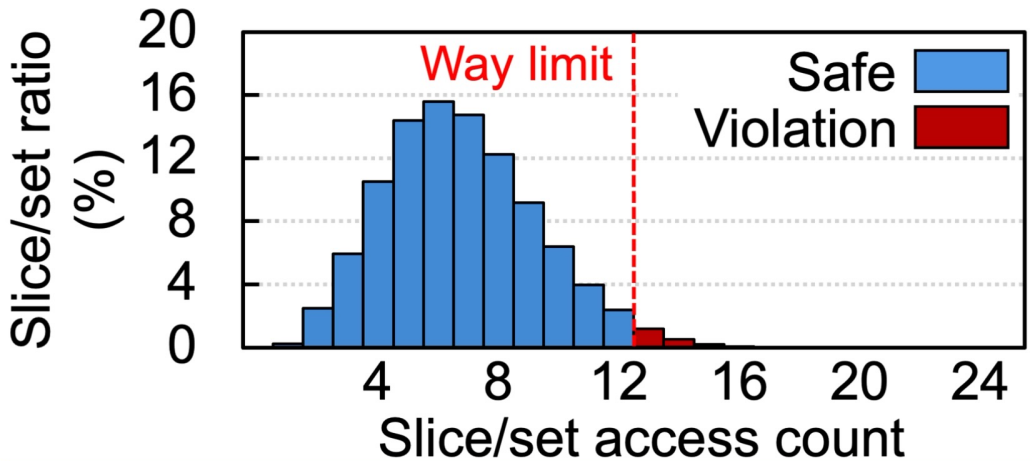


Violation ratio = 1.97%

Set Conflicts Matter

Case 1: 22MB working set

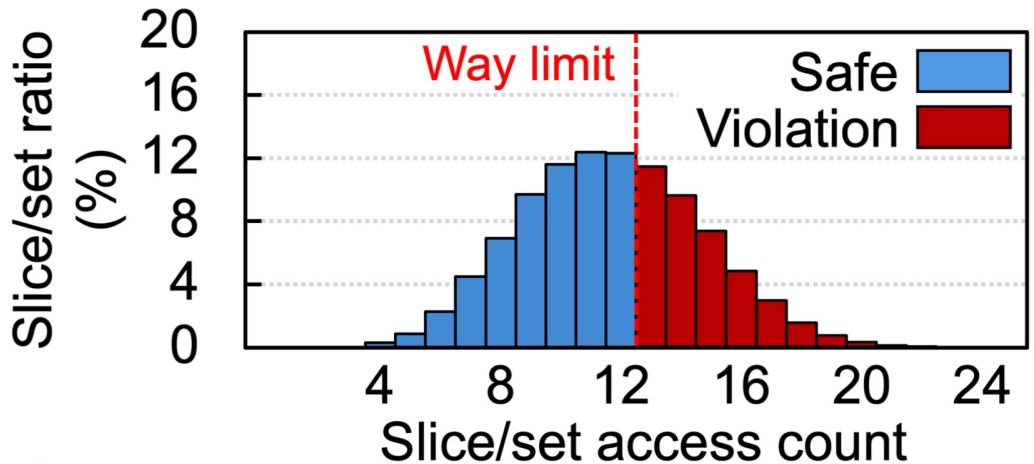




Violation ratio = 1.97%

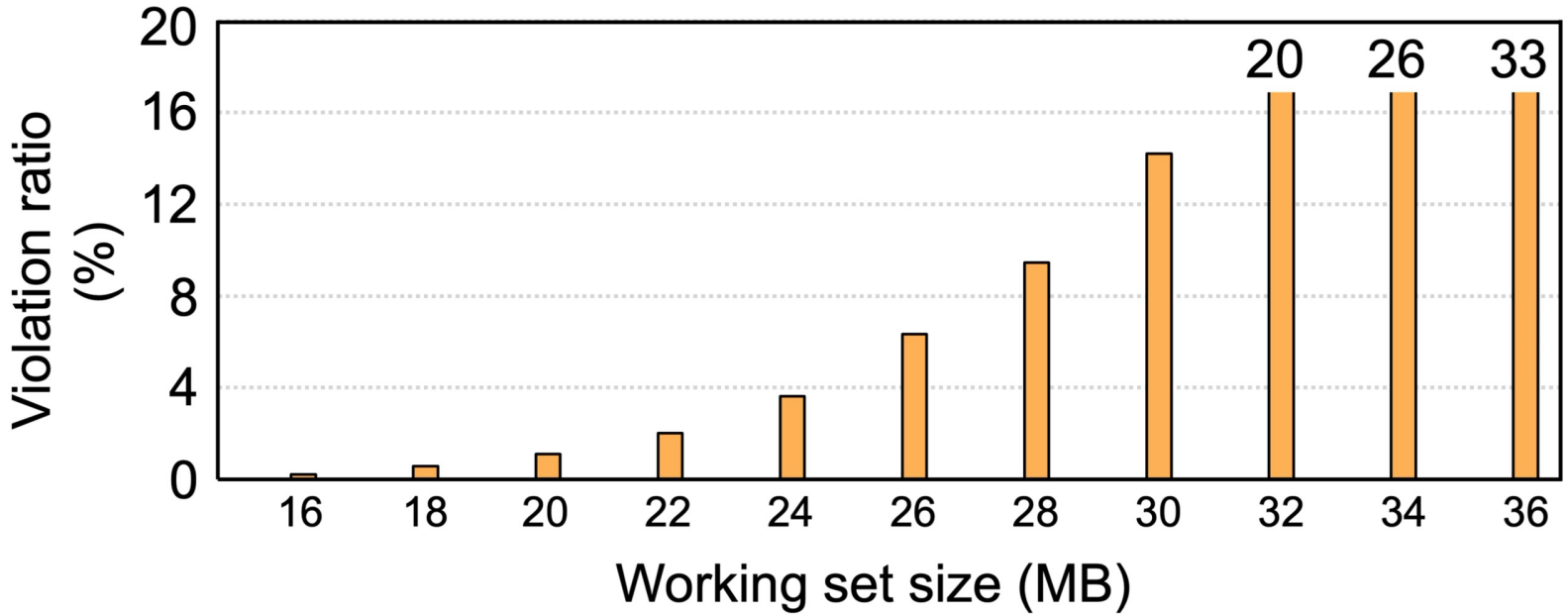
Case 2: 38MB working set



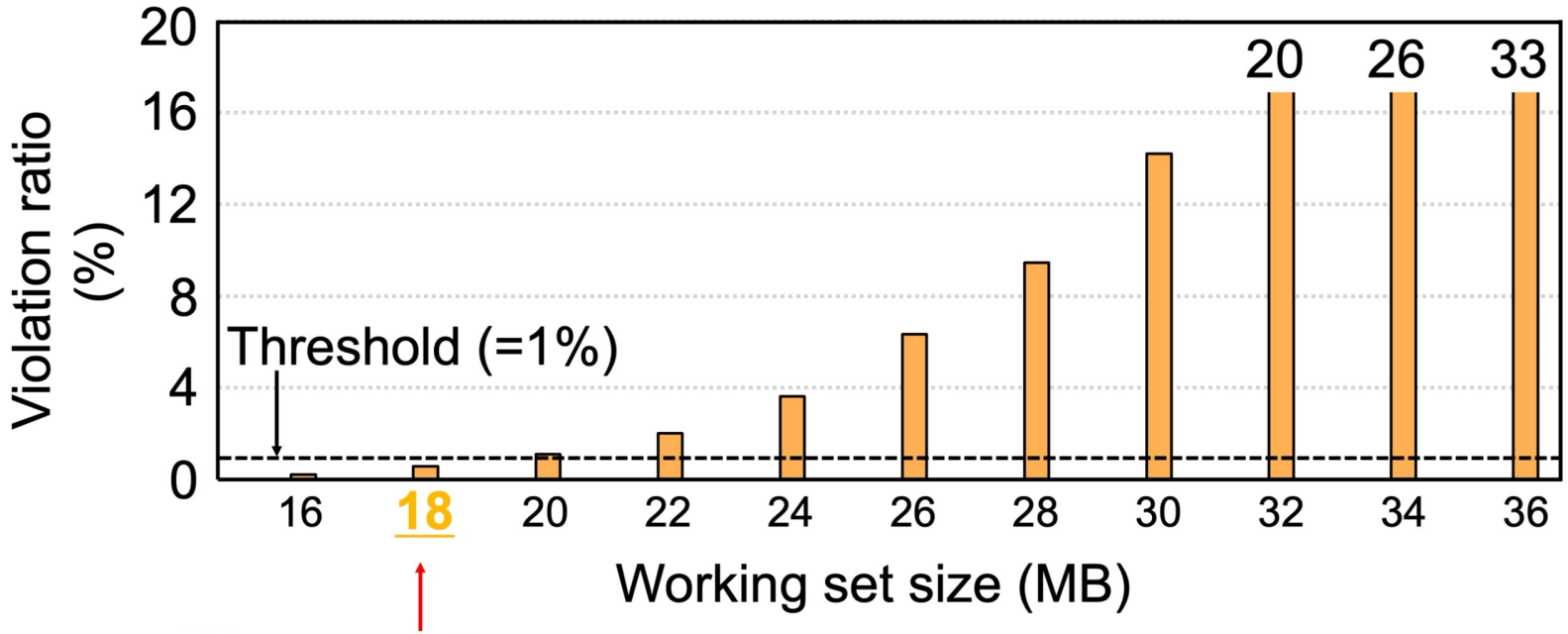


Violation ratio = 39.3%

Set Conflicts Shrink Effective LLC Capacity

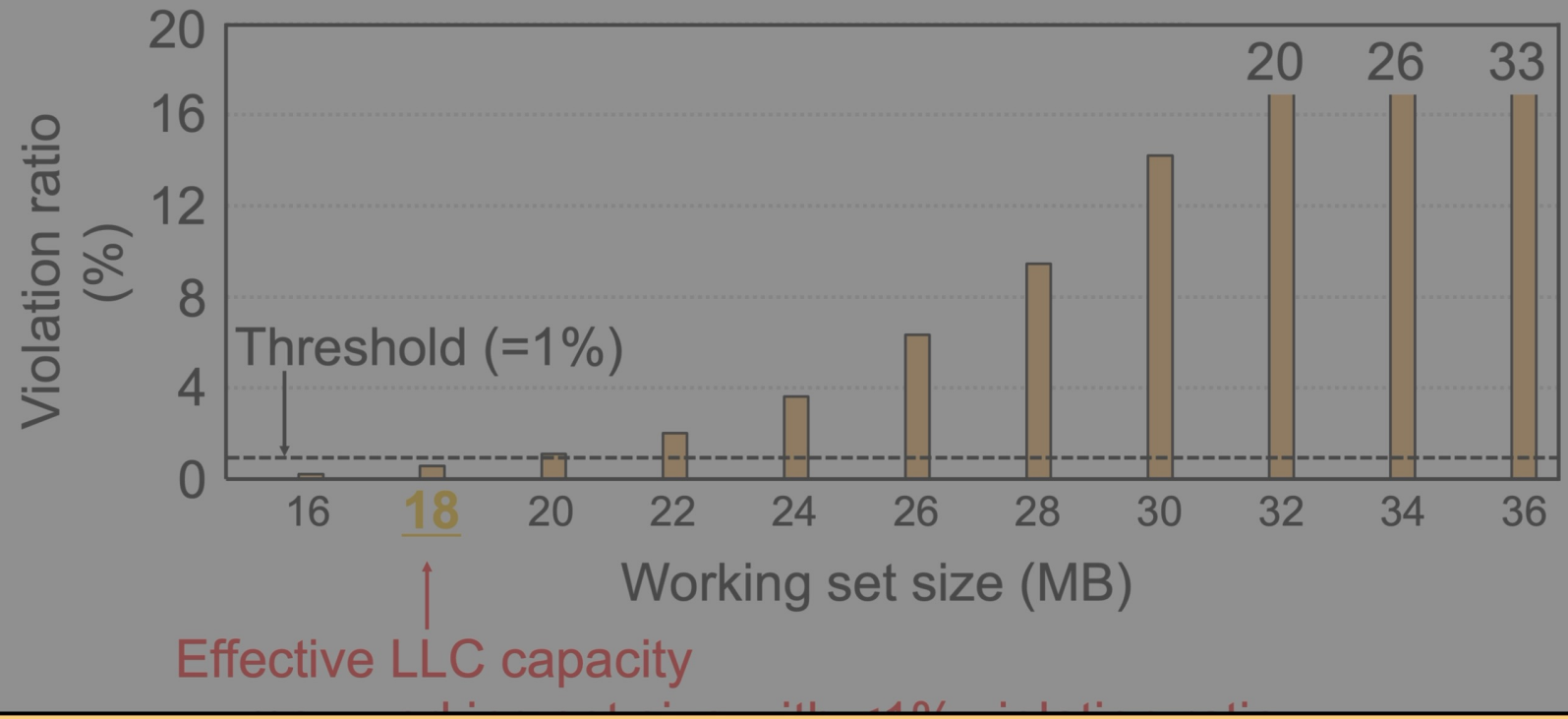


Set Conflicts Shrink Effective LLC Capacity



Effective LLC capacity
→ Max working set size with <1% violation ratio

Set Conflicts Shrink Effective LLC Capacity

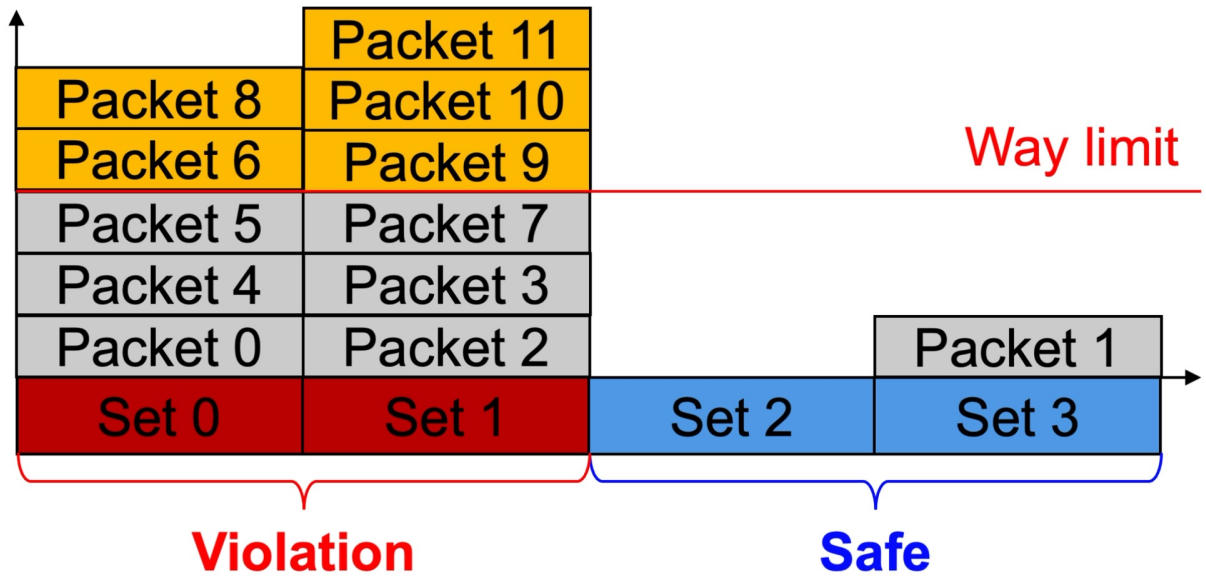


Observation 2:

Capacity alone is not enough; **set conflicts also matter**

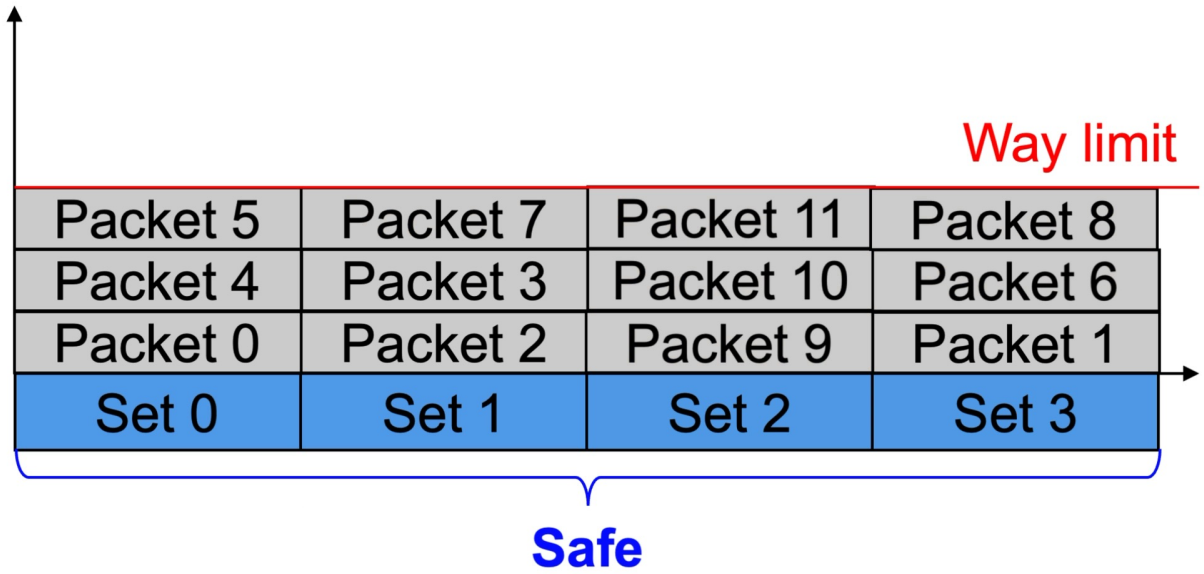
Default Linux: only **18 / 39 MB (46%)** is effective

Page Coloring to the Rescue!



Imbalanced

Page Coloring to the Rescue!

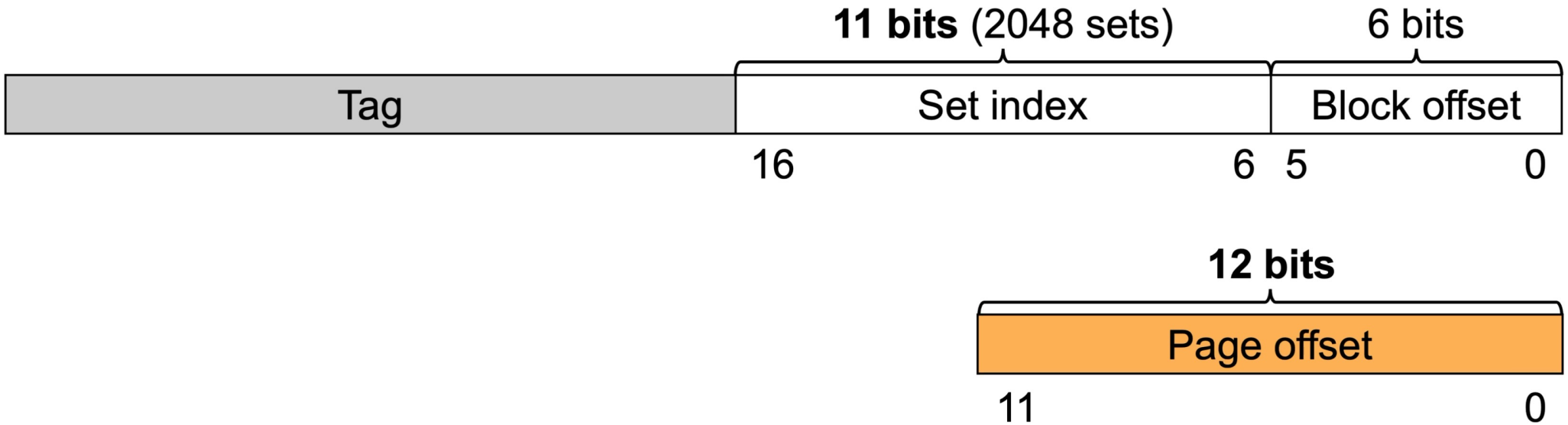


Balanced

Index

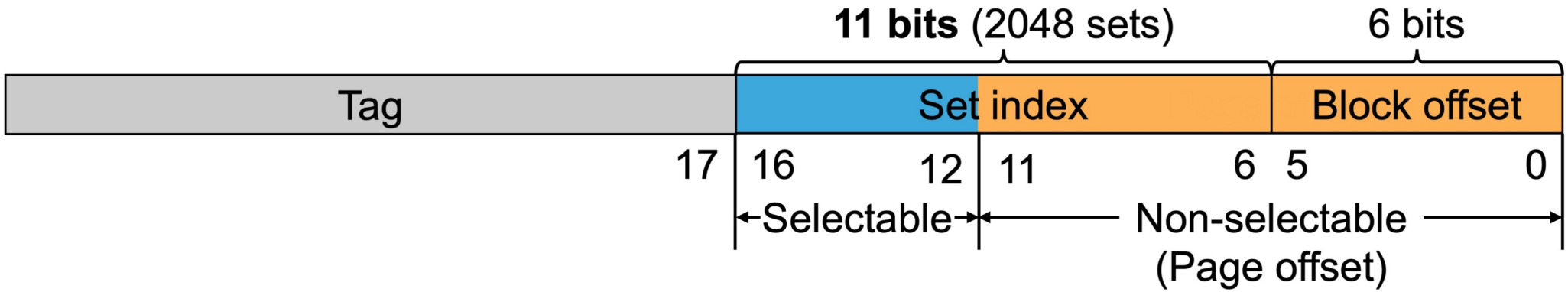
- Introduction
- Motivation
- **Sepia**
  - **Page Coloring Methodology**
  - **Color-Aware Page Allocation**
- Evaluation

Page Coloring Methodology

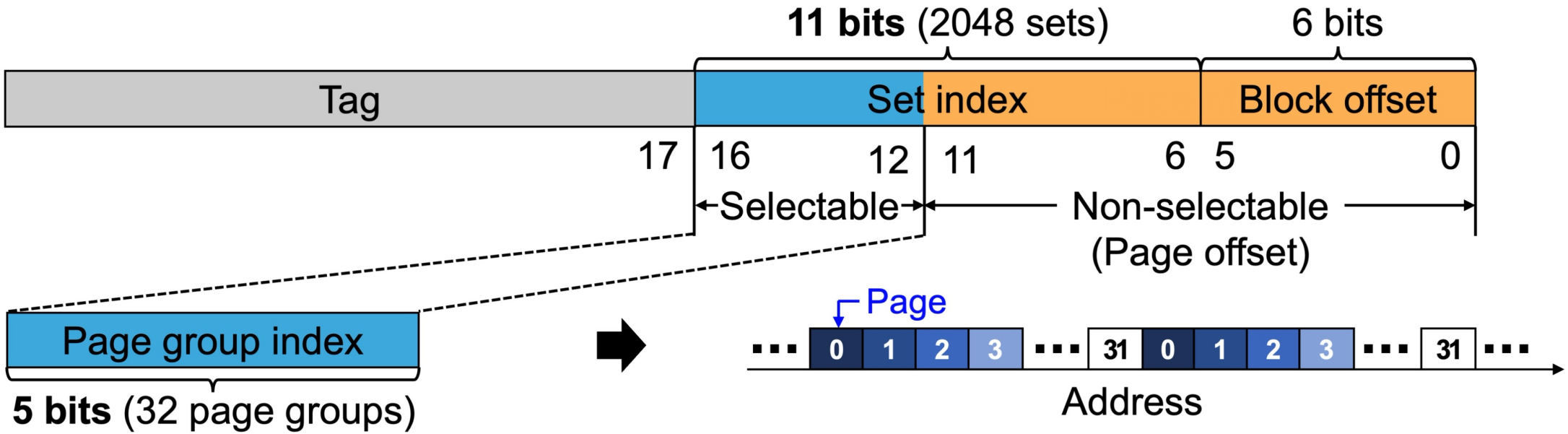


*** Xeon Gold 6354 LLC config (Our system)**
26 slices, (2048 sets, 12ways) per slices, 39 MB

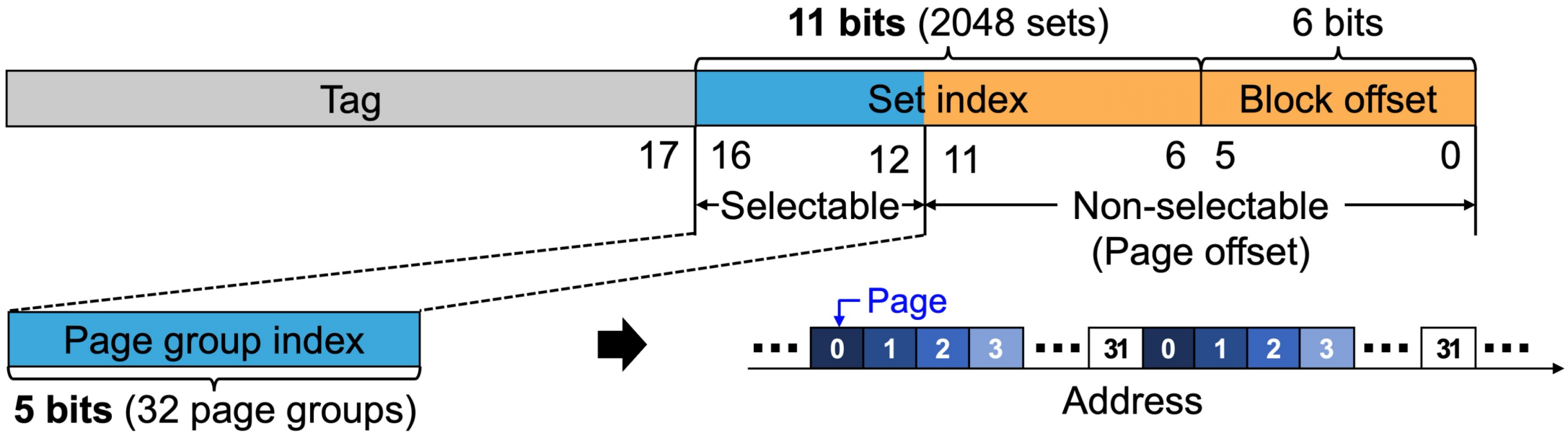
Page Coloring Methodology

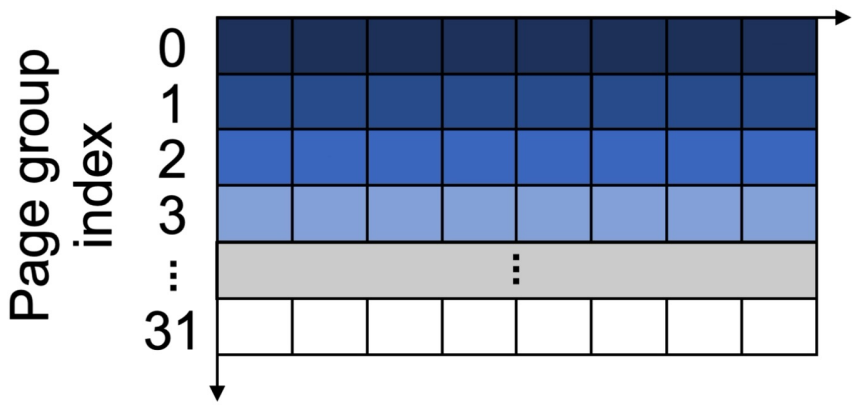


Page Coloring Methodology

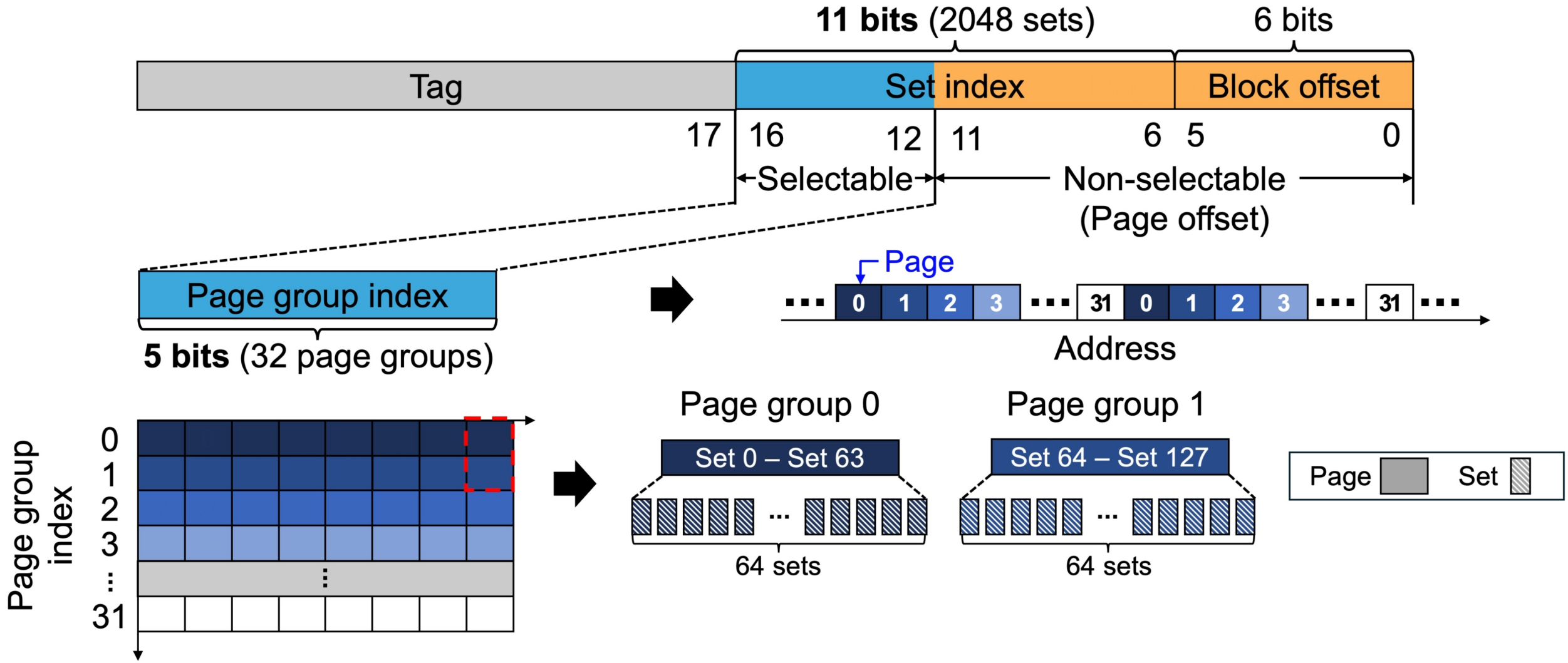


Page Coloring Methodology

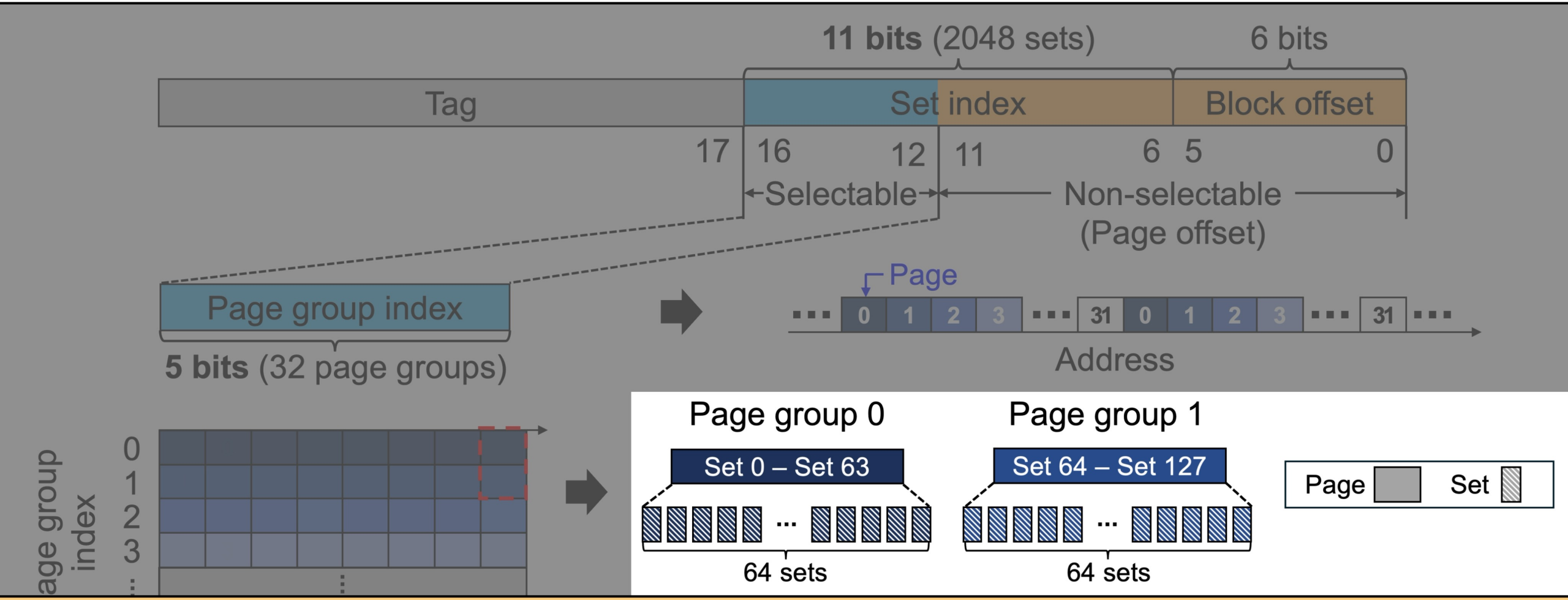




Page Coloring Methodology

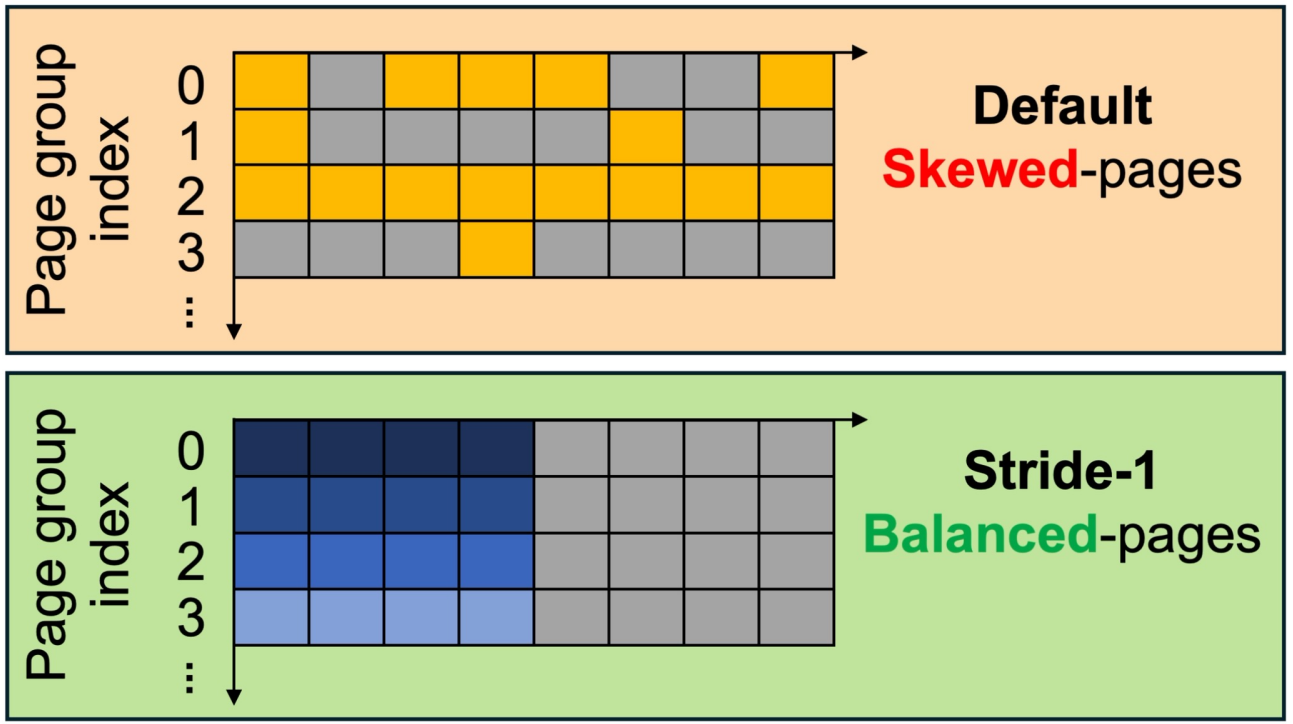


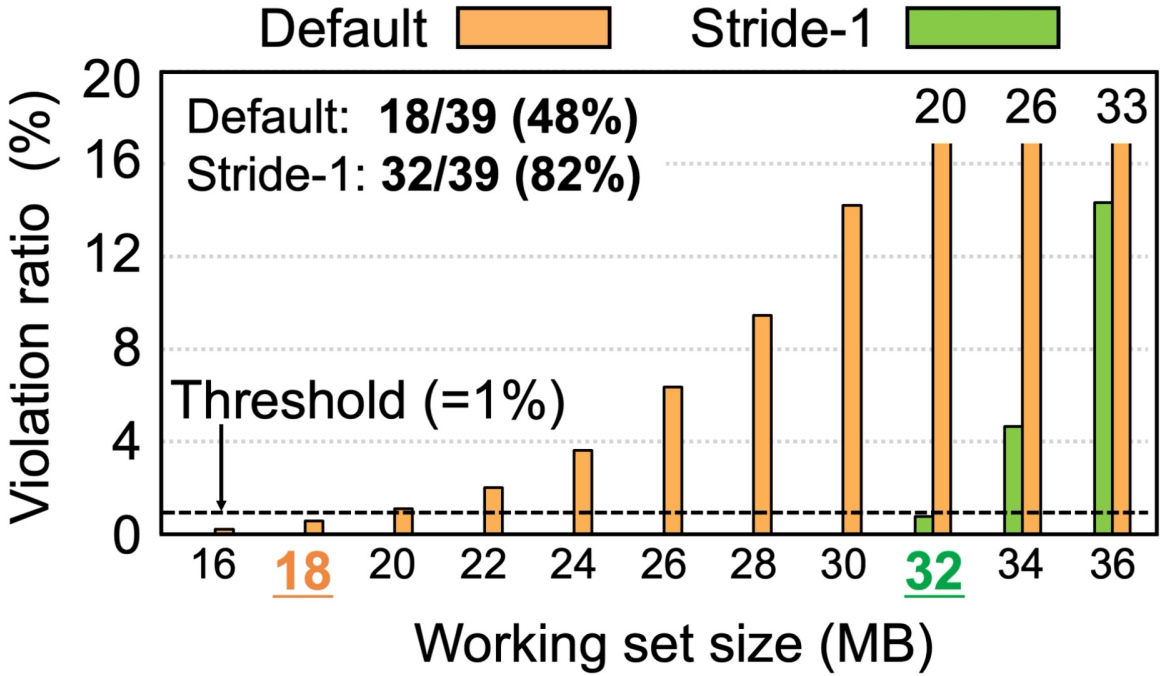
Page Coloring Methodology



Pages from different colors (groups) → **Avoid set conflicts**

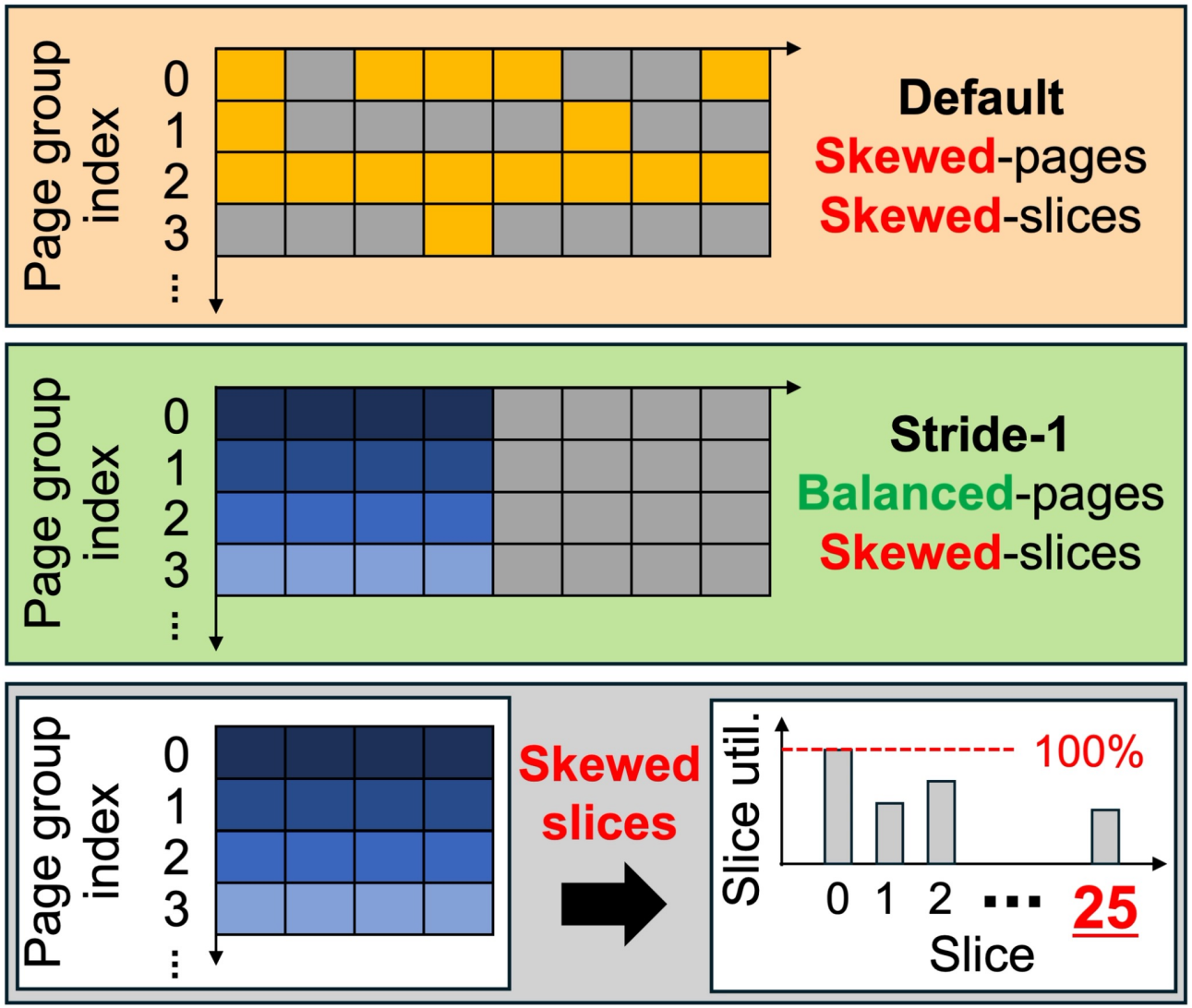
Default vs. Stride-1

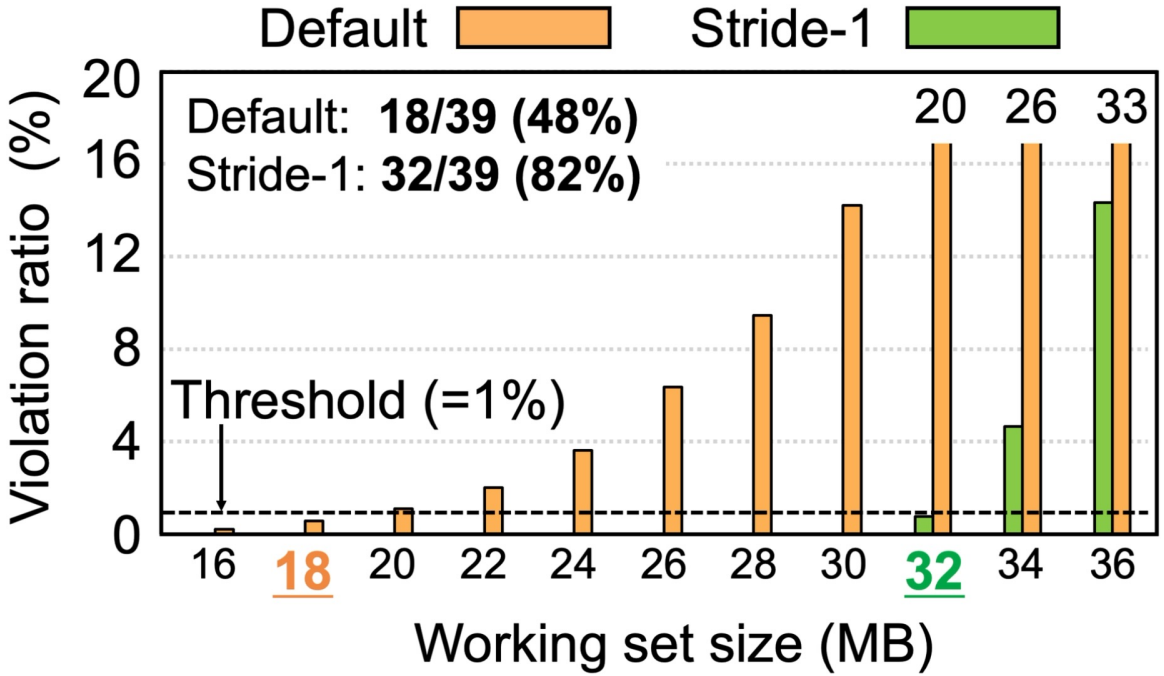




- Stride-1 increases effective capacity: 18MB → 32MB

Default vs. Stride-1

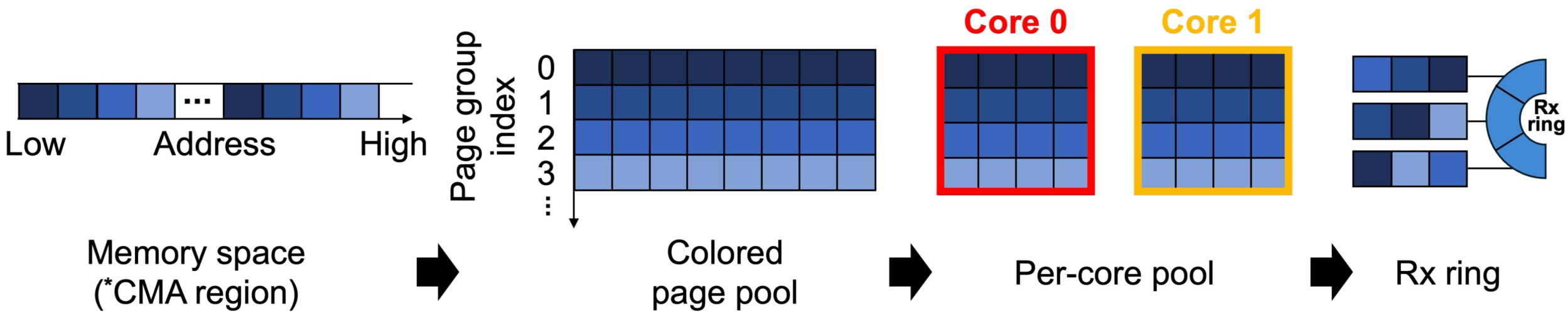




- Stride-1 increases effective capacity: 18MB → 32MB
- **Slice imbalance** still prevents full 39MB (See the paper for details)

Sepia Design

Sepia Goal: Improve LLC utilization via page coloring

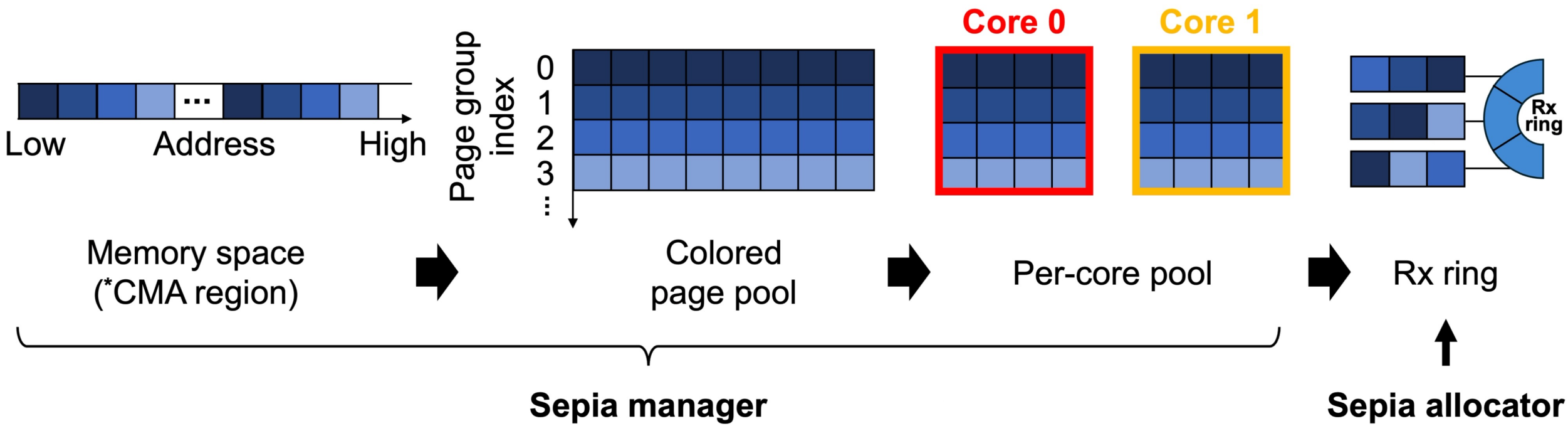


*** Contiguous Memory Allocator (CMA)**

Sepia Design

Sepia Goal: Improve LLC utilization via page coloring

- **Key components**
  - **Sepia manager**: Prepare per-core colored pools
  - **Sepia allocator**: Replenish the Rx ring in Stride-1 order

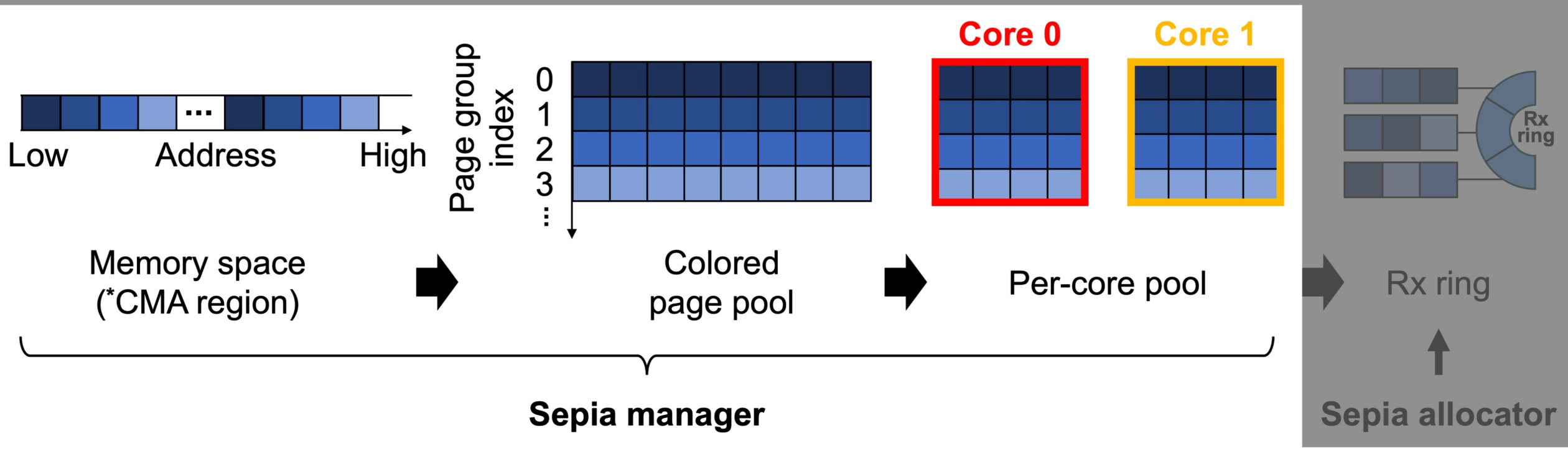


* Contiguous Memory Allocator (CMA)

Sepia Design

Sepia Goal: Improve LLC utilization via page coloring

- **Key components**
  - **Sepia manager**: Prepare per-core colored pools
  - **Sepia allocator**: Replenish the Rx ring in Stride-1 order

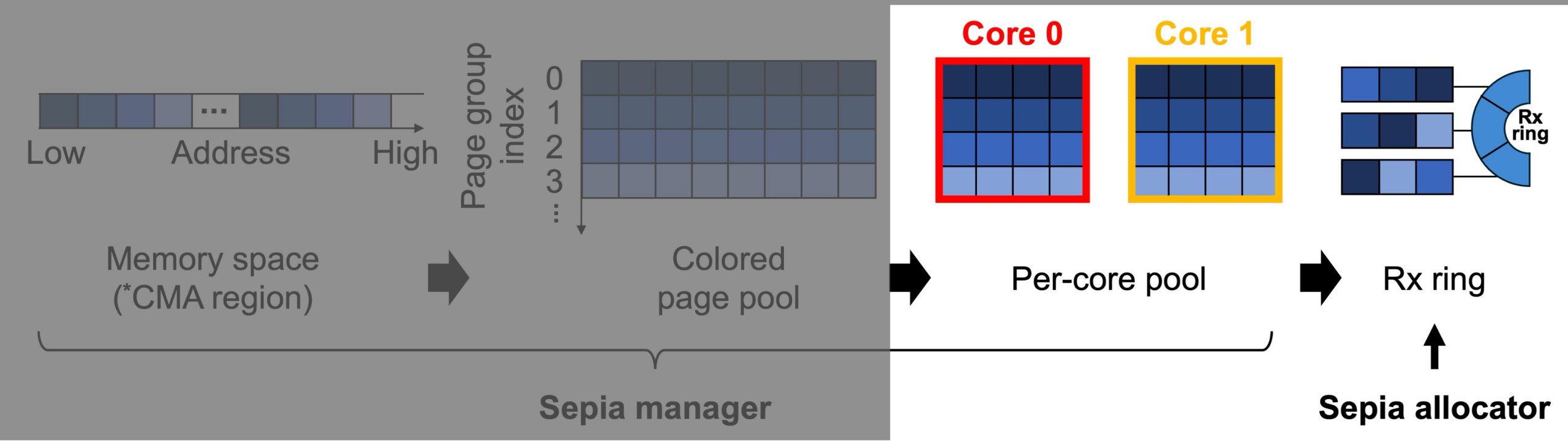


* Contiguous Memory Allocator (CMA)

Sepia Design

Sepia Goal: Improve LLC utilization via page coloring

- **Key components**
  - **Sepia manager**: Prepare per-core colored pools
  - **Sepia allocator**: Replenish the Rx ring in Stride-1 order



* Contiguous Memory Allocator (CMA)

Sepia Allocator

- **How to achieve color-balancing?**



Sepia Allocator

- **How to achieve color-balancing?**



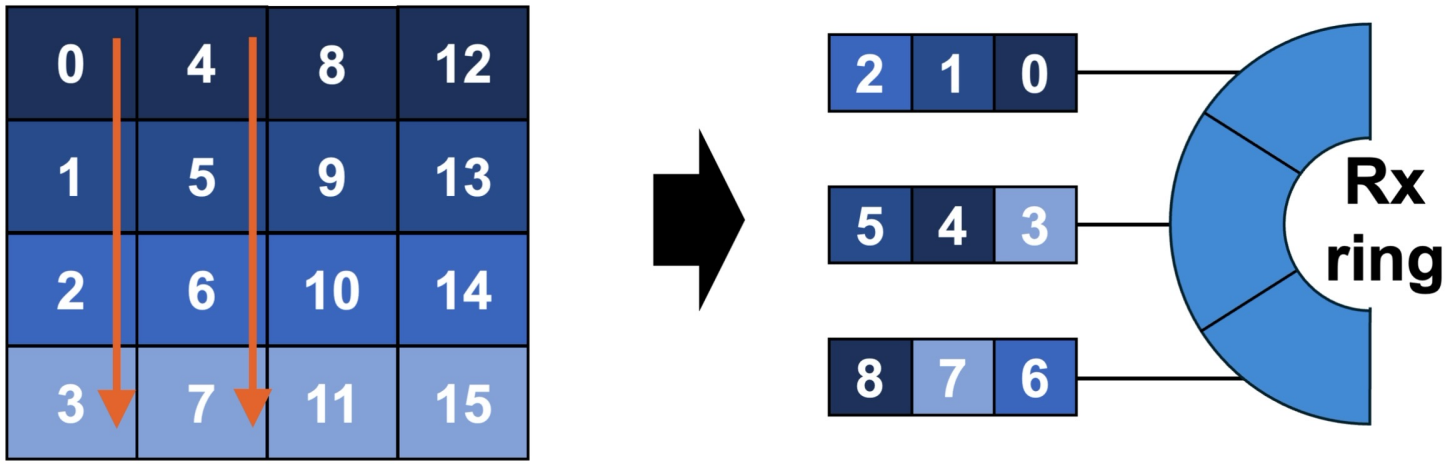
Ideal case: Keep **Stride-1** order in contiguous memory space

Sepia Allocator

- **How to achieve color-balancing?**

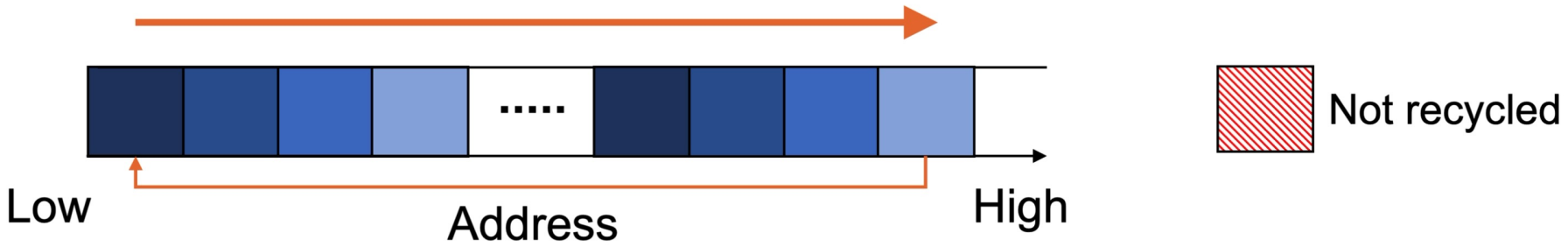


Ideal case: Keep **Stride-1** order in contiguous memory space



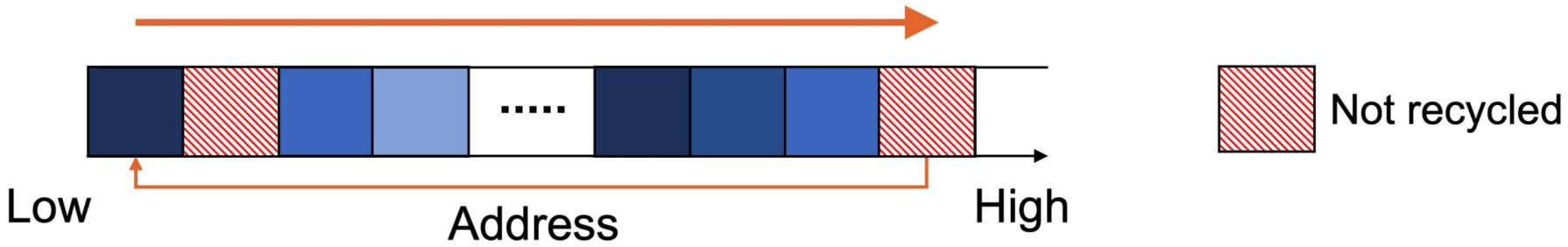
Sepia Allocator

- **Challenge 1**: Different page-recycling timing



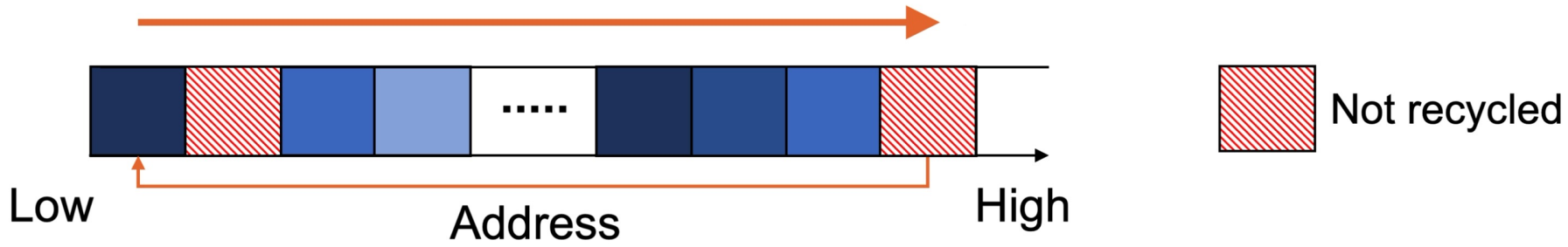
Sepia Allocator

- **Challenge 1**: Different page-recycling timing

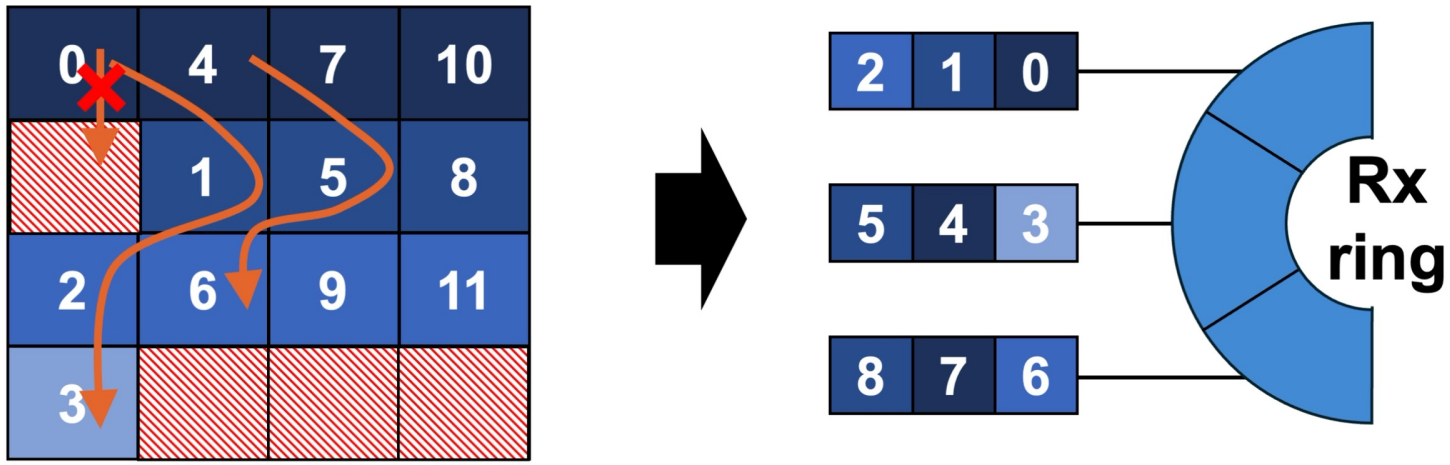


Sepia Allocator

- **Challenge 1**: Different page-recycling timing

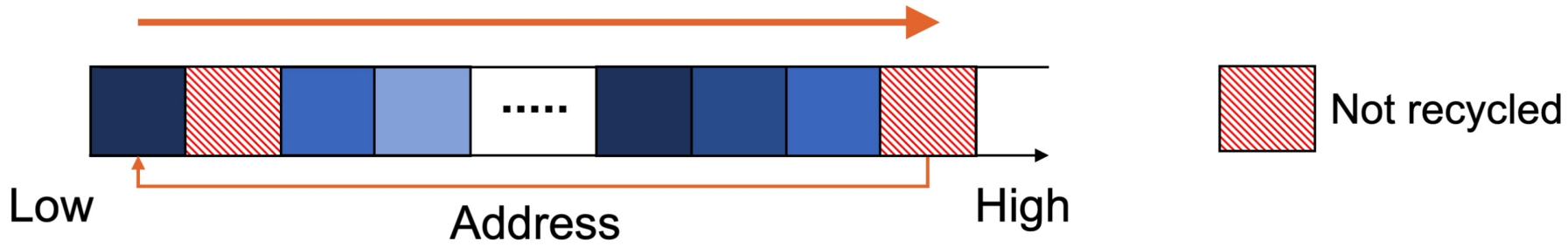


Policy #1: Allocate available pages in page-color order

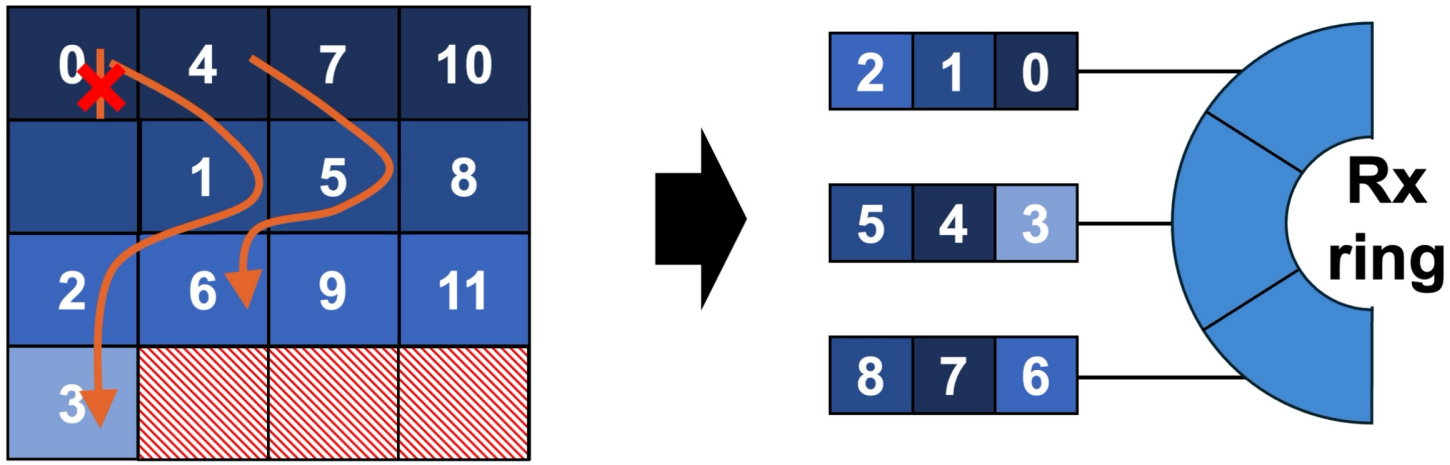


Sepia Allocator

- **Challenge 1**: Different page-recycling timing

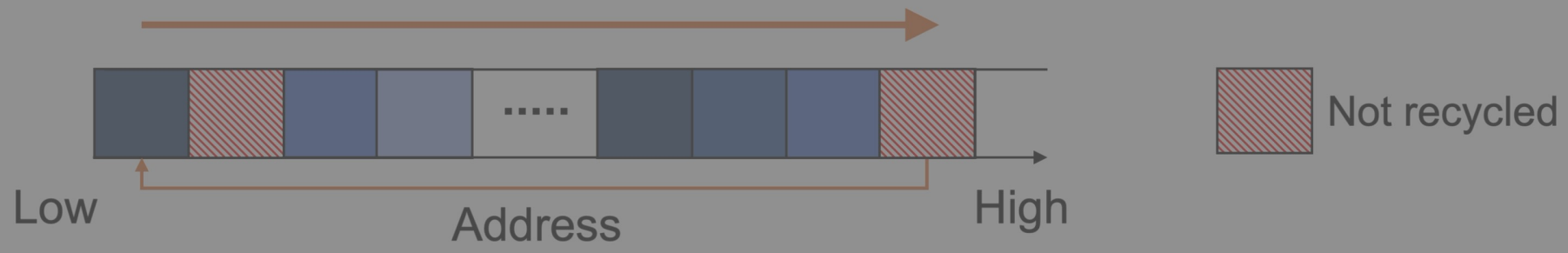


Policy #1: Allocate available pages in page-color order



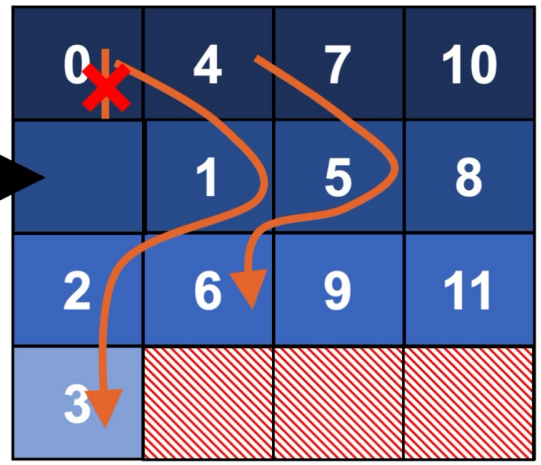
Sepia Allocator

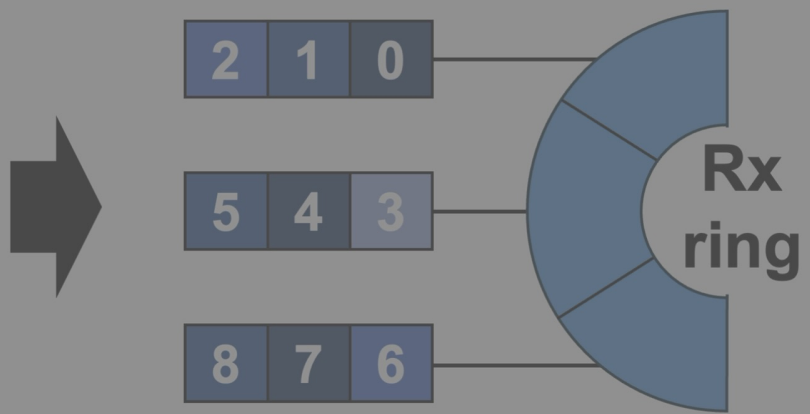
- **Challenge 1:** Different page-recycling timing



Policy #1: Allocate available pages in page-color order

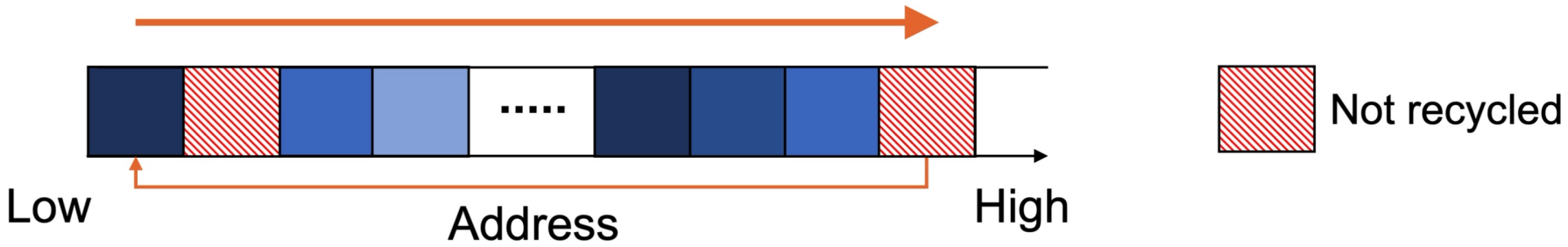
Policy #2: Prioritize lowest-address pages





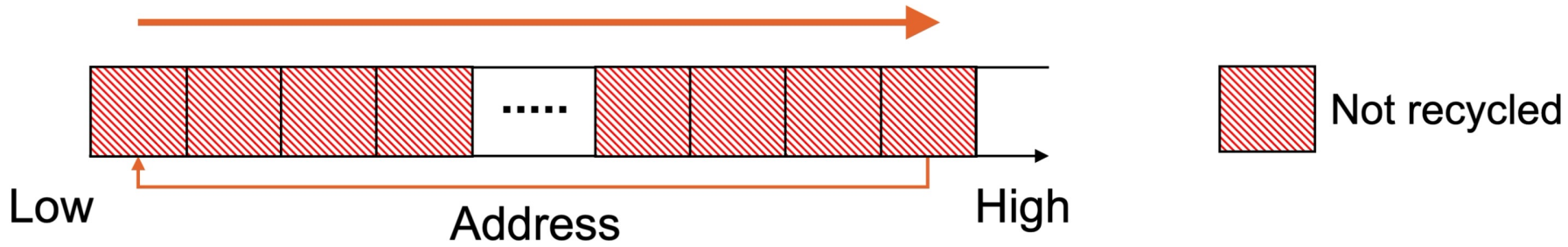
Sepia Allocator

- **Challenge 2**: Page-pool exhaustion due to bursty traffic



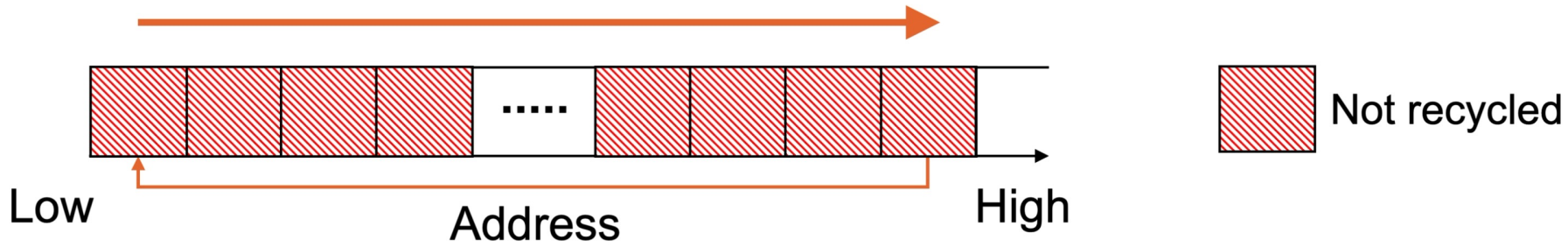
Sepia Allocator

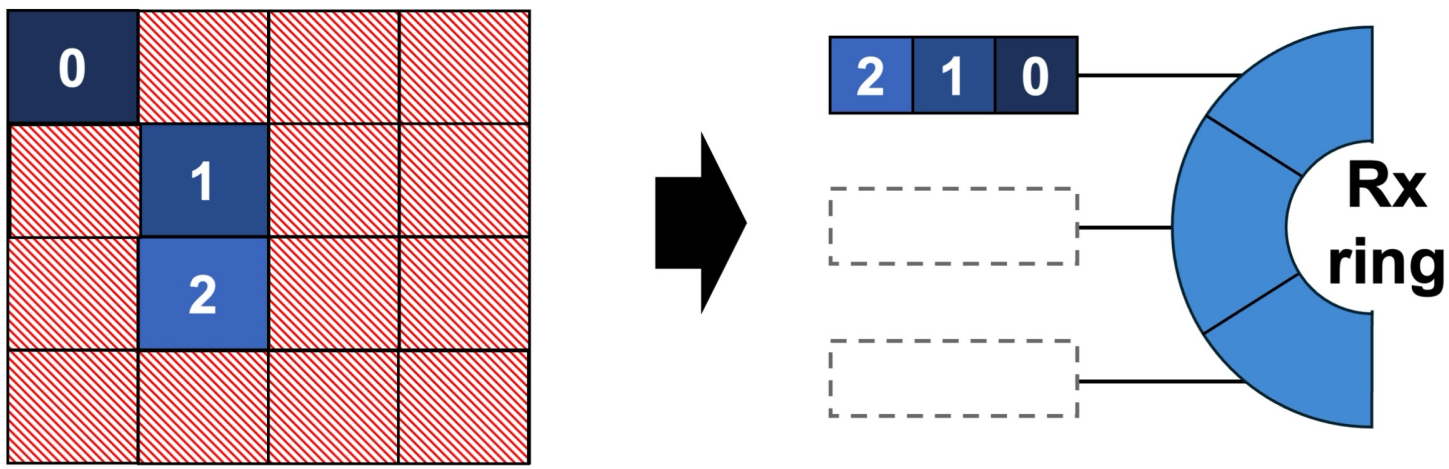
- **Challenge 2**: Page-pool exhaustion due to bursty traffic



Sepia Allocator

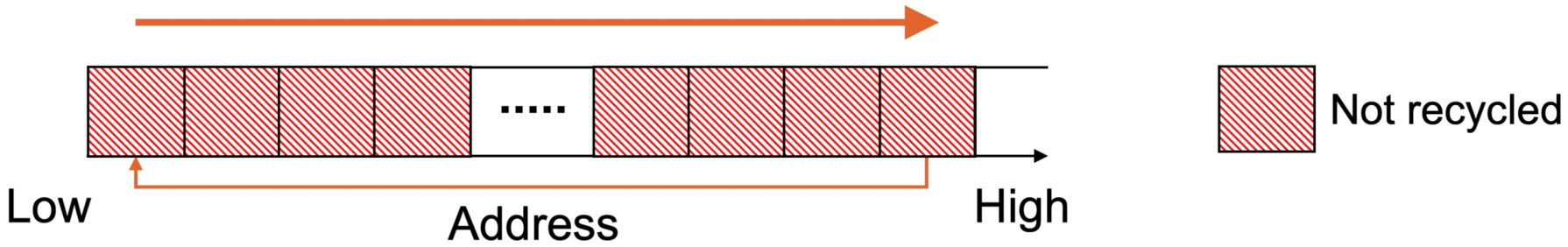
- **Challenge 2**: Page-pool exhaustion due to bursty traffic

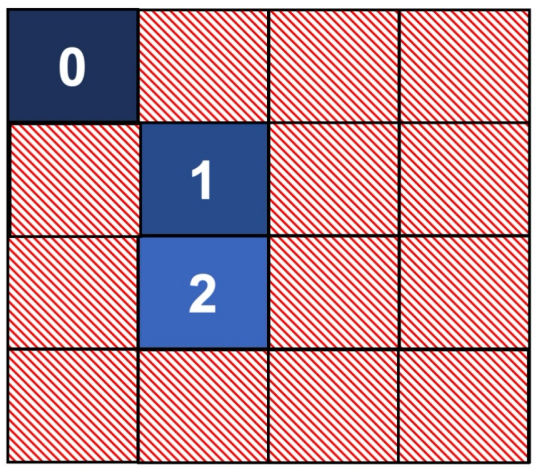


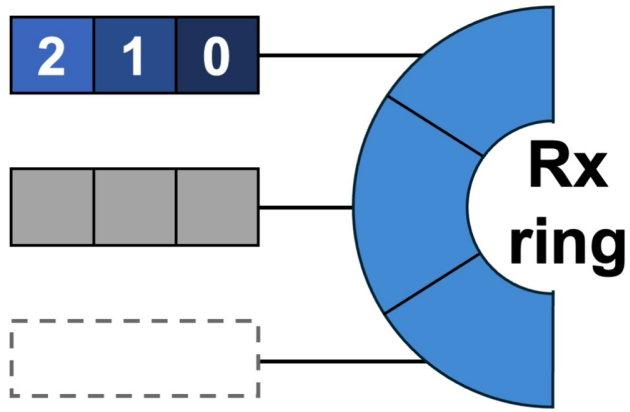


Sepia Allocator

- **Challenge 2**: Page-pool exhaustion due to bursty traffic





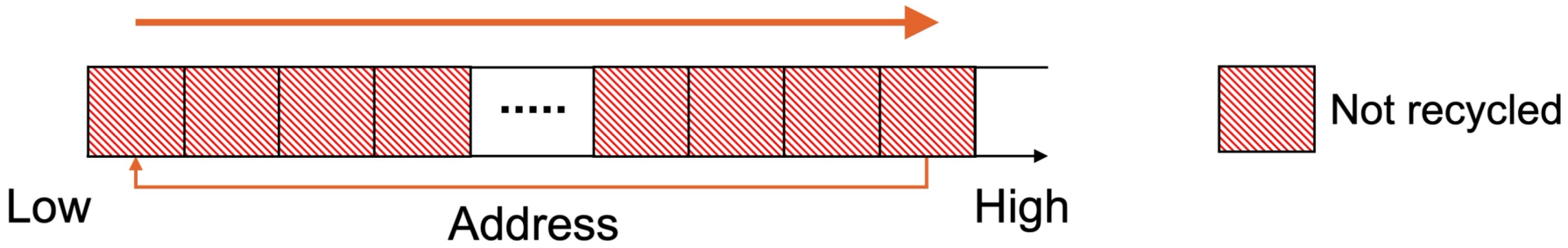


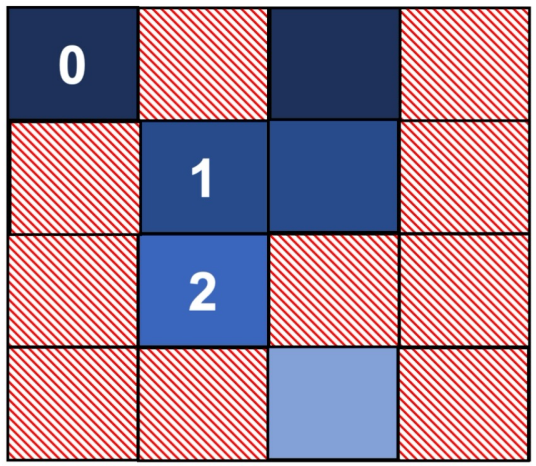
Policy #3: Fall back to Linux allocator

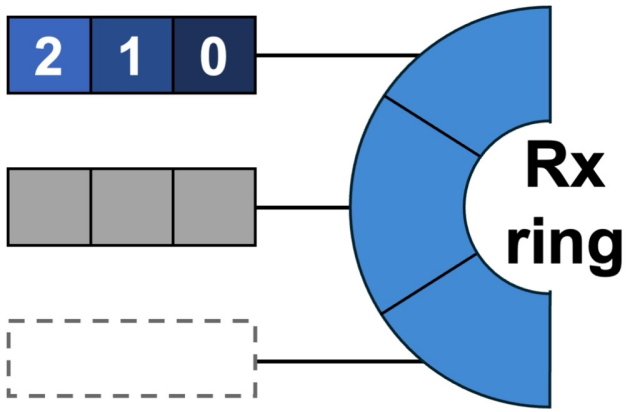
Default allocator

Sepia Allocator

- **Challenge 2**: Page-pool exhaustion due to bursty traffic





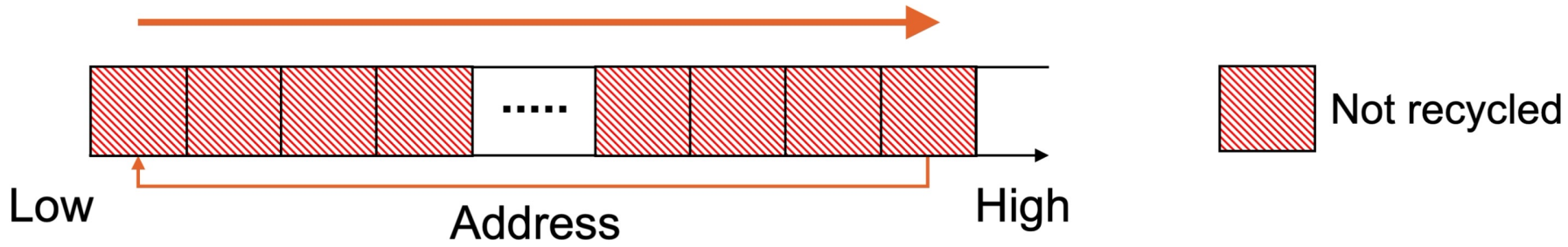


Policy #3:
Fall back to
Linux allocator

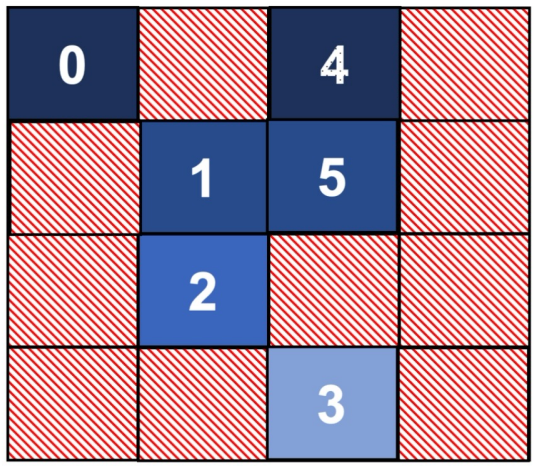
Default allocator

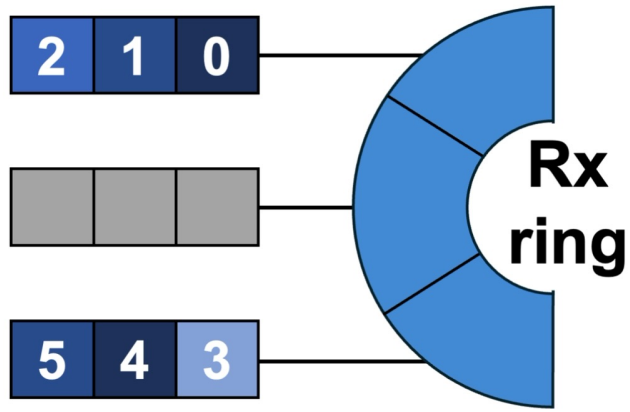
Sepia Allocator

- **Challenge 2**: Page-pool exhaustion due to bursty traffic



Policy #4: Prioritize colored pages during allocation





Policy #3: Fall back to Linux allocator

Default allocator

Index

- Introduction
- Motivation
- Sepia
- **Evaluation**

Evaluation Setup

- **Implemented entirely in the Linux kernel (6.6)**
  - No application modifications required
  - Sepia’s CMA usage is comparable to Linux’s memory footprint

Evaluation Setup

- **Implemented entirely in the Linux kernel (6.6)**
  - No application modifications required
  - Sepia's CMA usage is comparable to Linux's memory footprint

- **To push the bottleneck to network processing**
  - Two servers with 18 DDIO-enabled cores connected directly
  - 200Gbps ConnectX-6 NICs

Evaluation Setup

- **Implemented entirely in the Linux kernel (6.6)**
  - No application modifications required
  - Sepia's CMA usage is comparable to Linux's memory footprint
- **To push the bottleneck to network processing**
  - Two servers with 18 DDIO-enabled cores connected directly
  - 200Gbps ConnectX-6 NICs
- **To evaluate further CPU-efficiency gains beyond working-set reduction schemes (e.g., ShRing*)**
  - Rx ring size: 1024 entries (16MB) → 256 entries (4MB)

* Pismenny et al., "ShRing: Networking with Shared Receive Rings," USENIX OSDI'23

Evaluation Scenarios

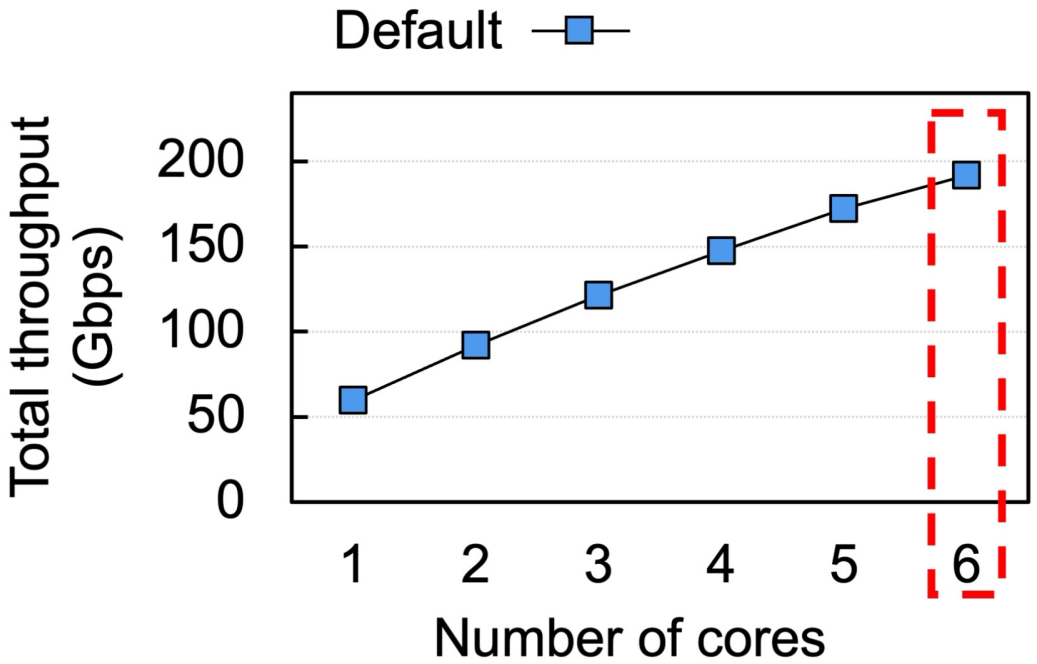
- **Microbenchmarks**
  - Best case: Working set size ≤ LLC capacity
  - Worst case: Working set size > LLC capacity
- **Performance under Bursty Workloads**
  - Page-pool Exhaustion and Recovery Dynamics
- **Performance Gain Breakdown**
  - Factor Analysis of Stride-1 & Working-set Reduction
- **Real-world Applications**
  - SPDK NVMe-over-TCP
  - Nginx Web Server
  - Memcached In-Memory Key-Value Store

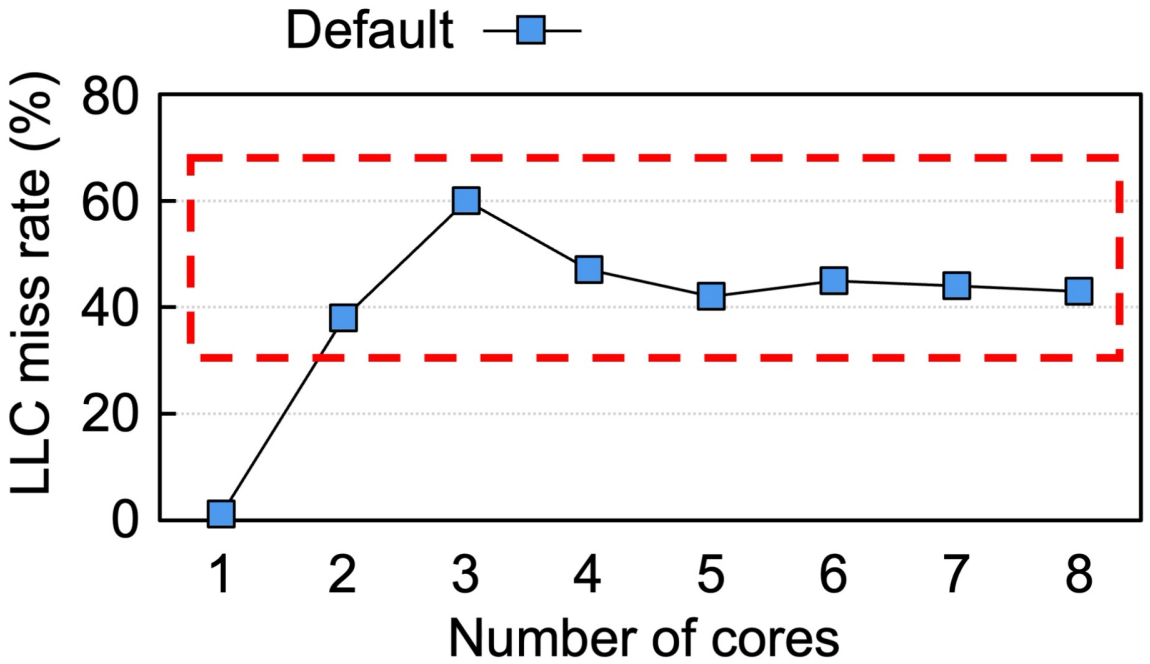
Evaluation Scenarios

- **Microbenchmarks**
  - Best case: Working set size ≤ LLC capacity
  - Worst case: Working set size > LLC capacity
- **Performance under Bursty Workloads**
  - Page-pool Exhaustion and Recovery Dynamics
- **Performance Gain Breakdown**
  - Factor Analysis of Stride-1 & Working-set Reduction
- **Real-world Applications**
  - SPDK NVMe-over-TCP
  - Nginx Web Server
  - Memcached In-Memory Key-Value Store

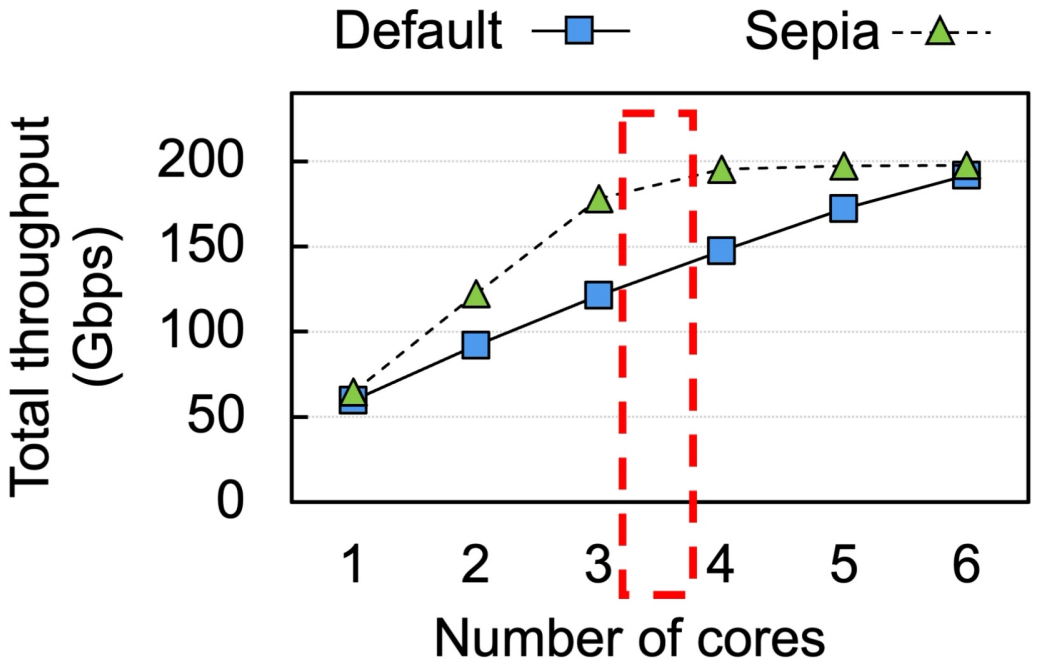
See the paper!

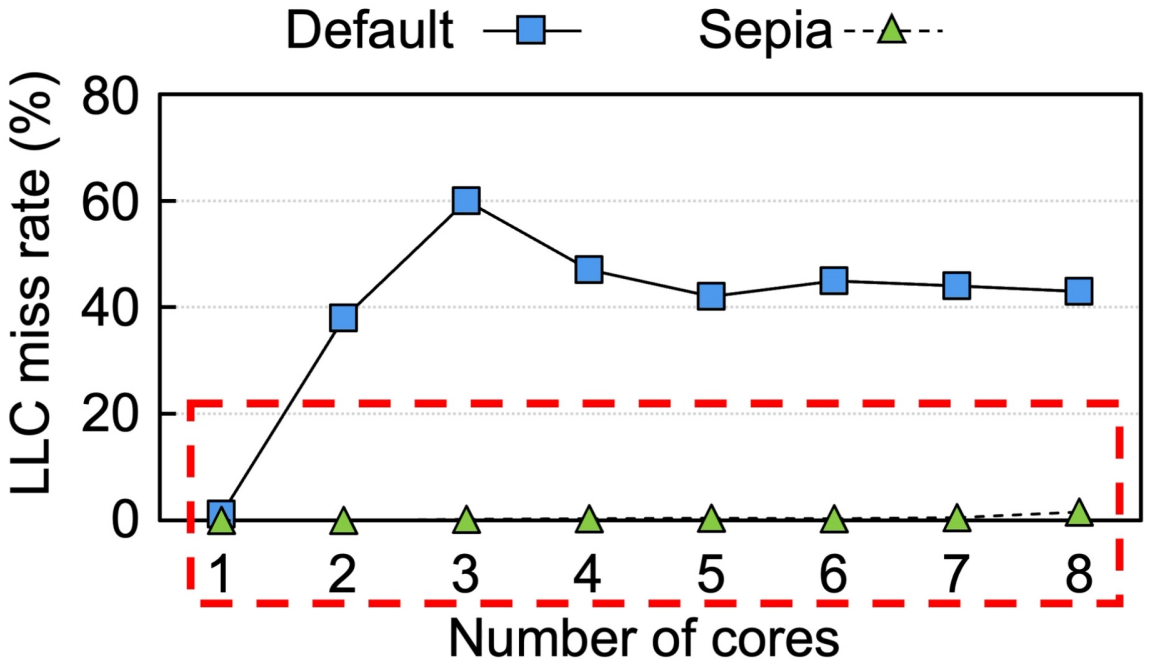
Microbenchmarks





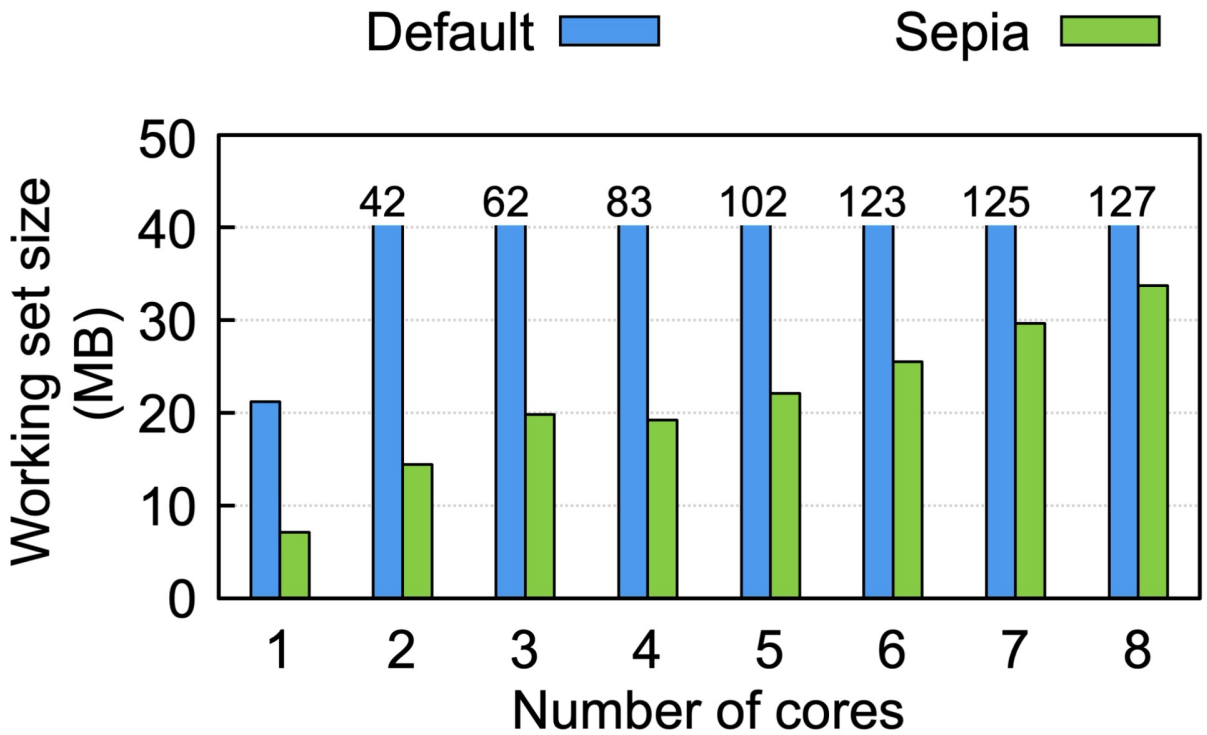
Microbenchmarks



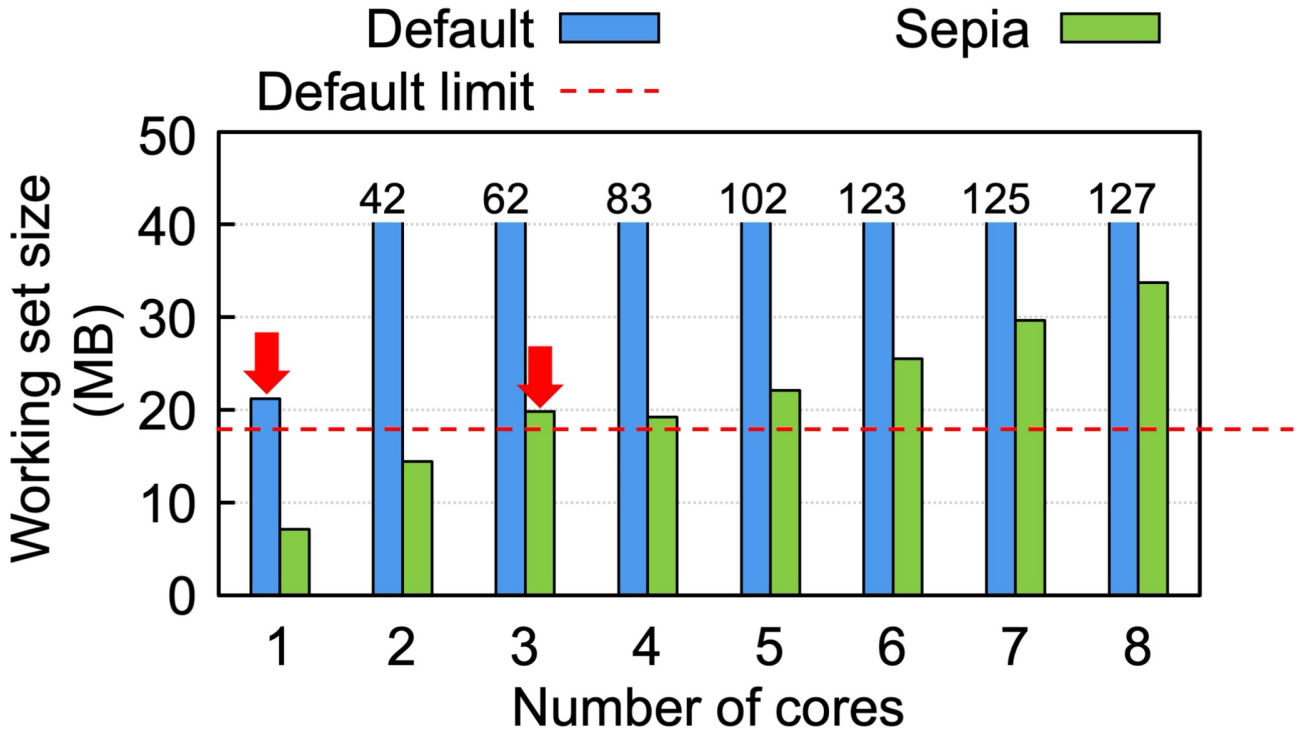


- **CPU efficiency**: Saturating 200Gbps 3.5 cores in Sepia
  (saving 2.5 cores over Default Linux)
- **LLC miss rate**: **Near-zero miss rate**
  by avoiding conflict and capacity pressure

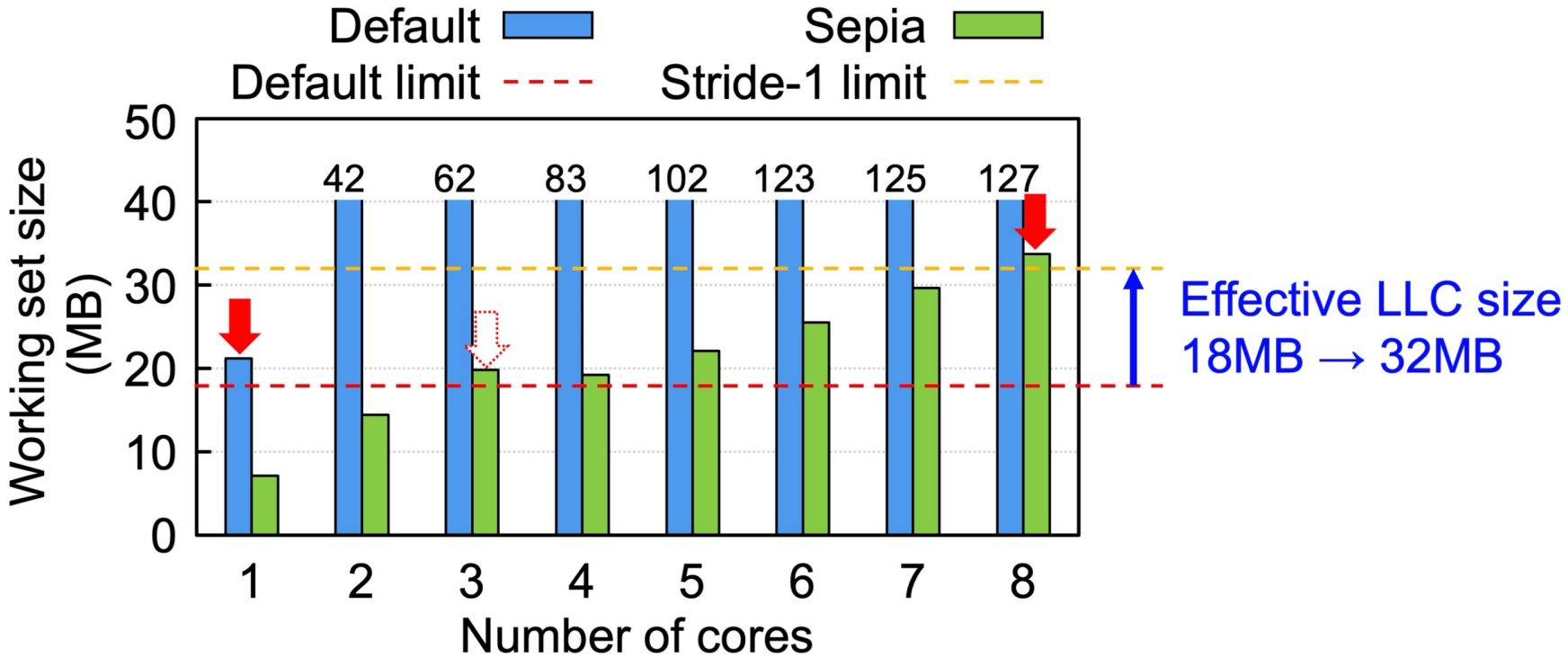
Microbenchmarks



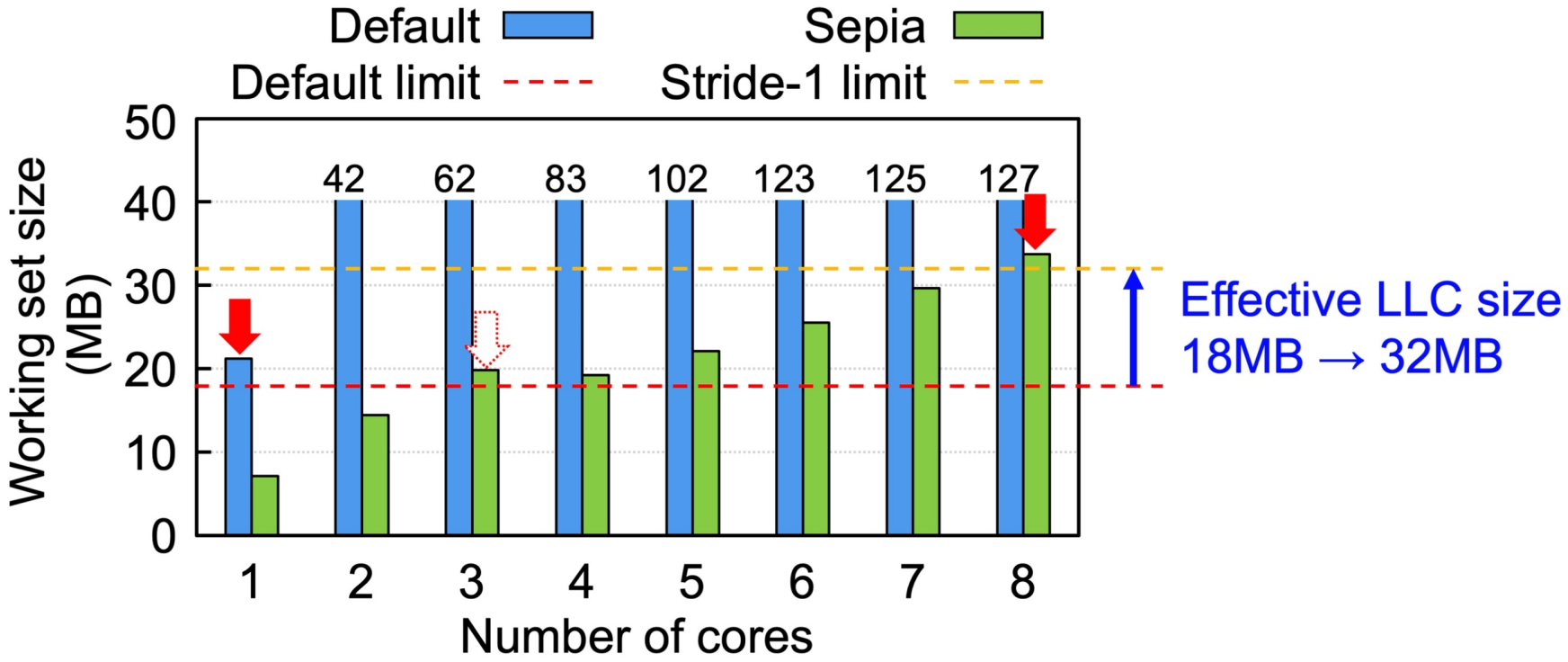
Microbenchmarks



Microbenchmarks

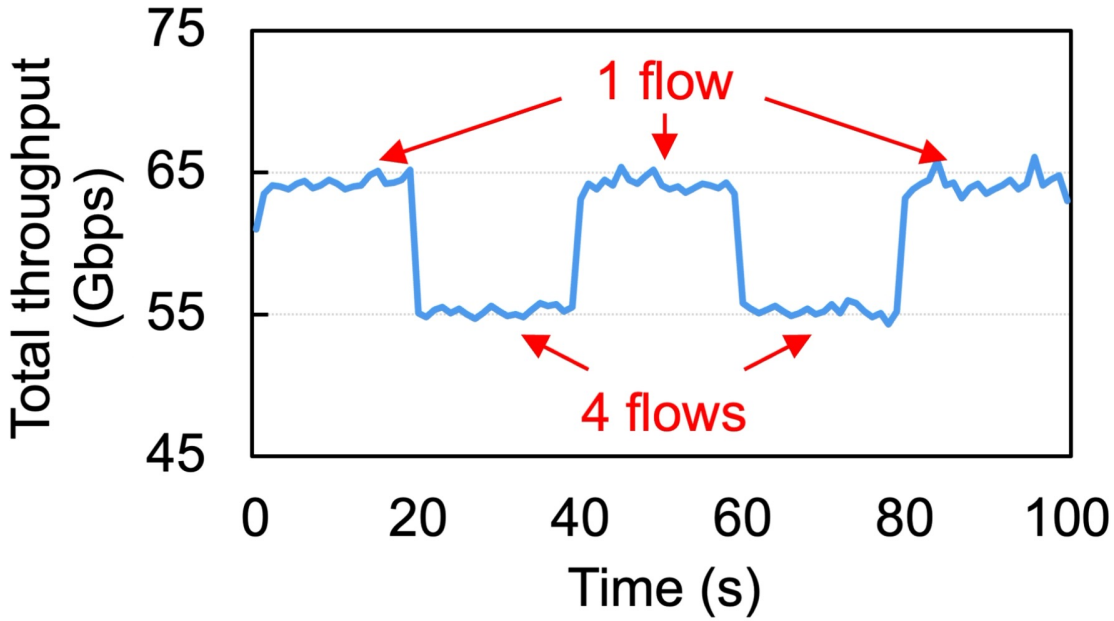


Microbenchmarks

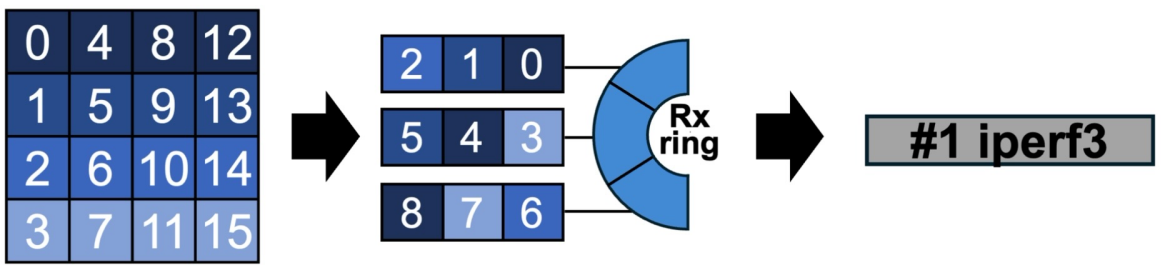


Sepia increases the effective LLC capacity by 78%

Performance under Bursty Workloads

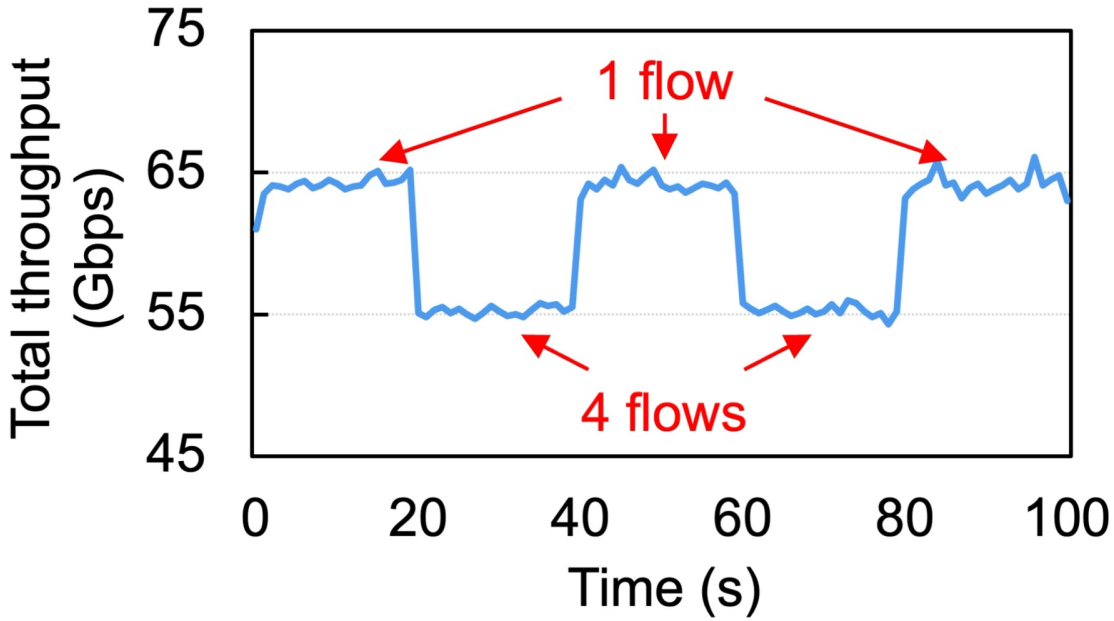


Case 1: 1 flow

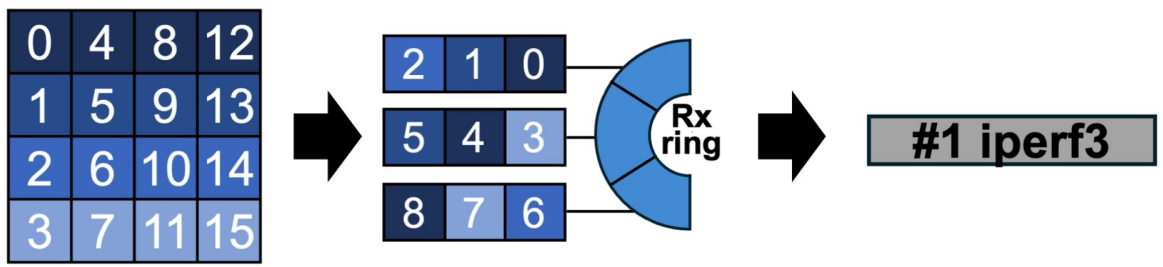


Sufficient page pool

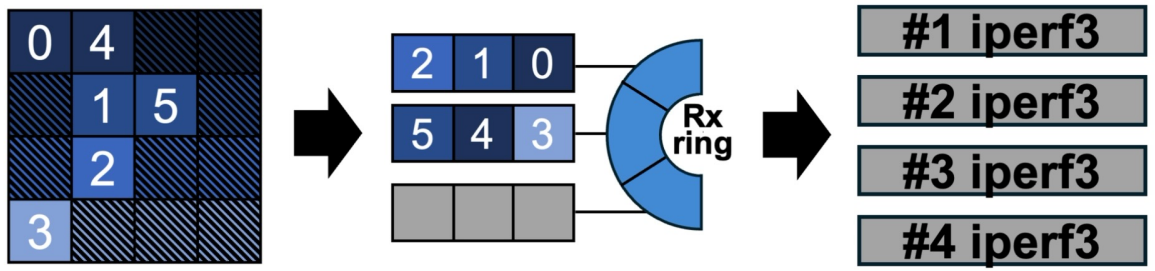
Performance under Bursty Workloads



Case 1: 1 flow



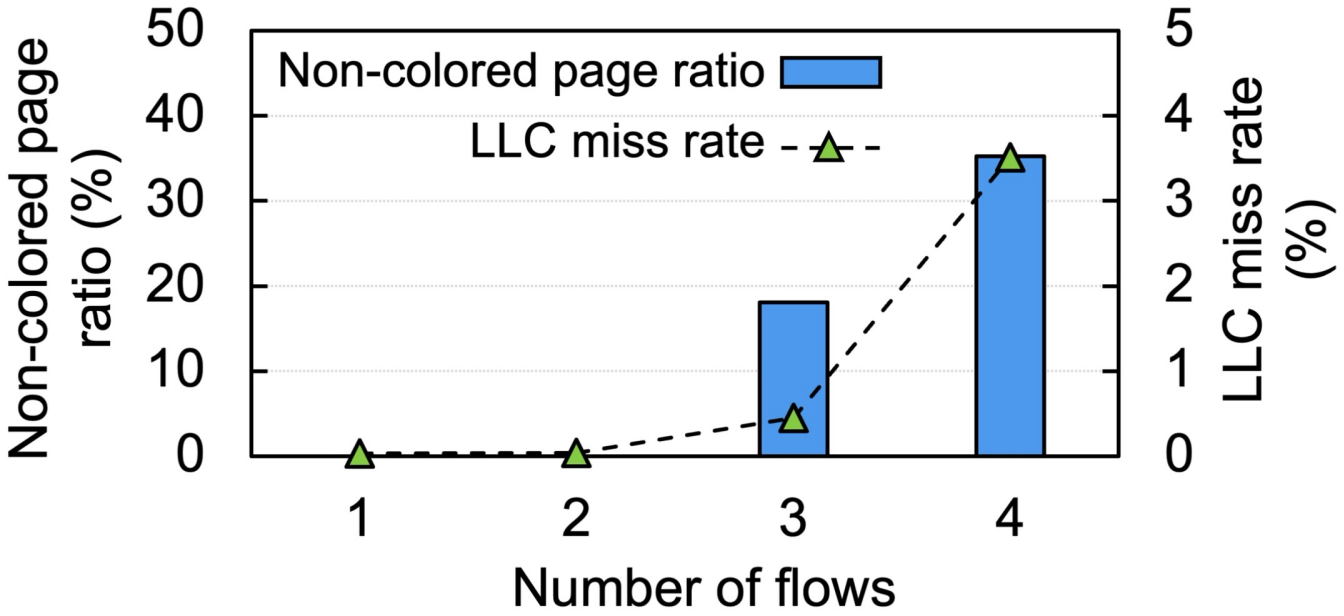
Case 2: 4 flows

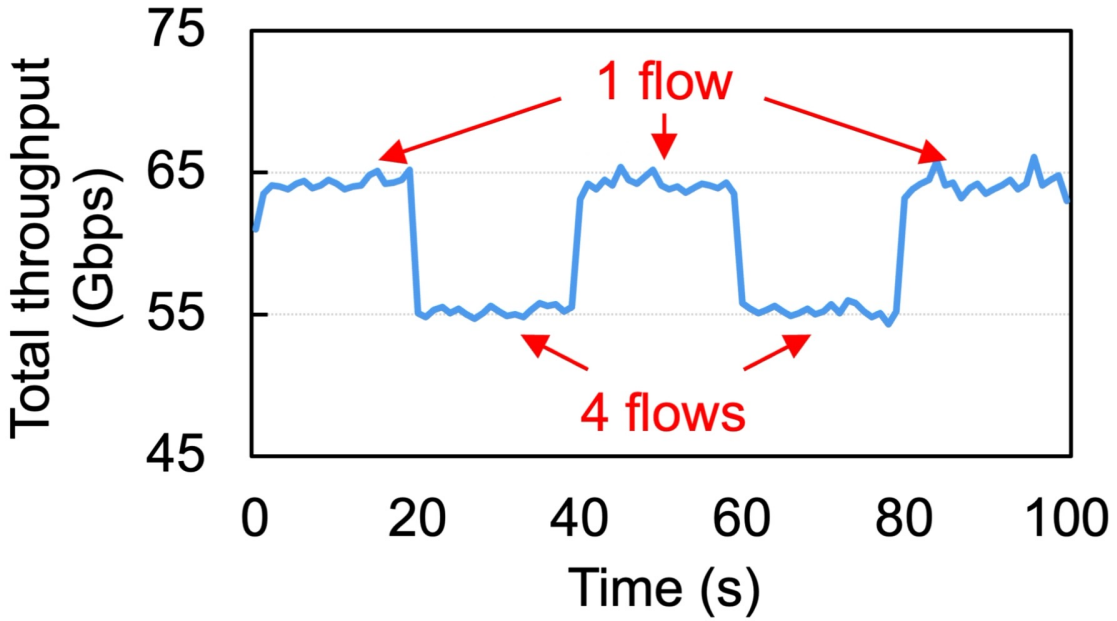


Sufficient page pool

Exhausted page pool

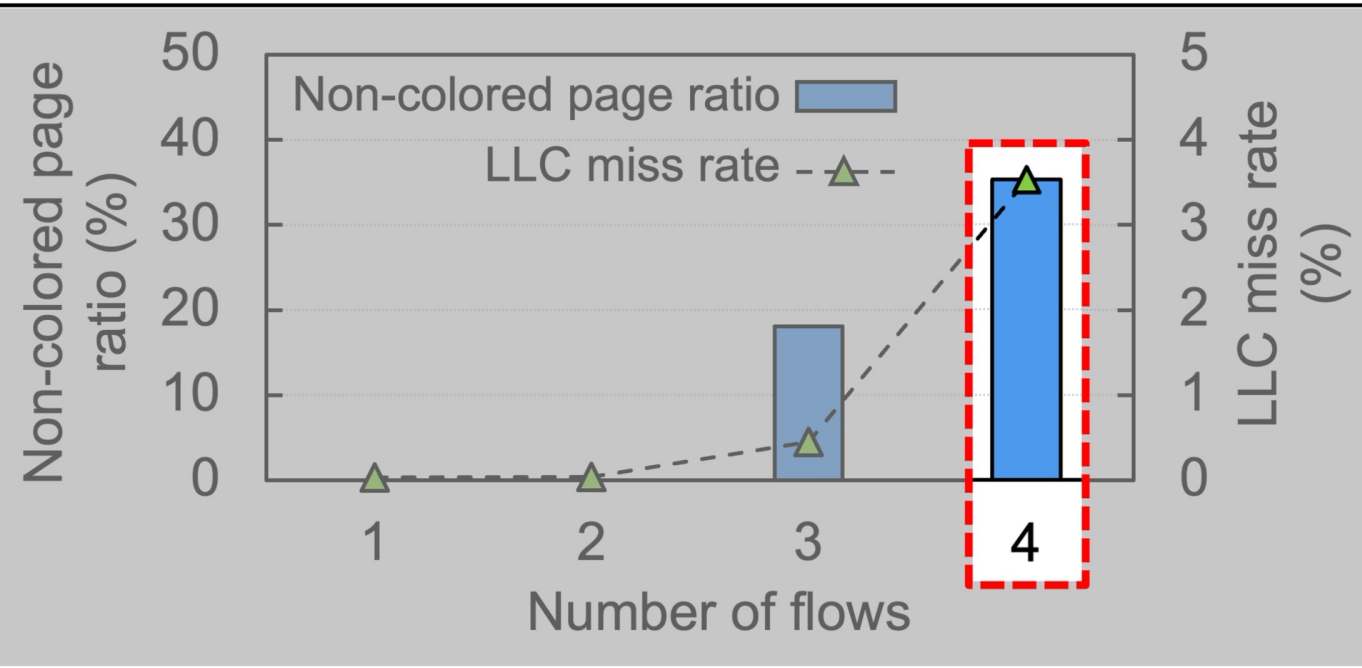
Performance under Bursty Workloads

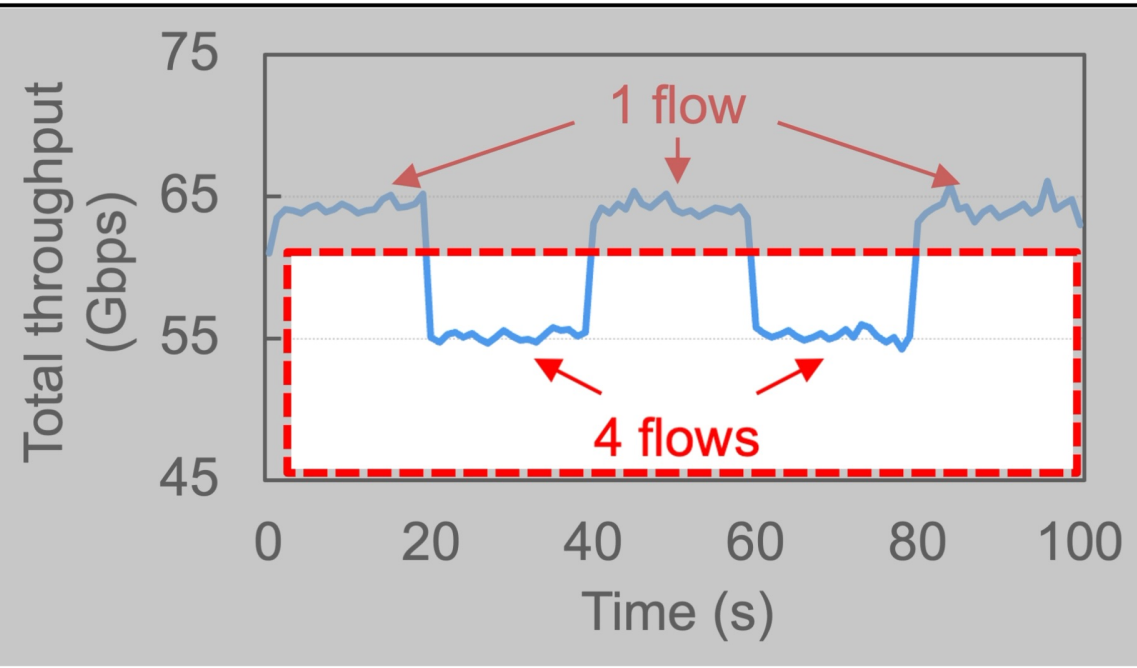




- **Exhaustion:** Safe fallback to standard Linux allocator

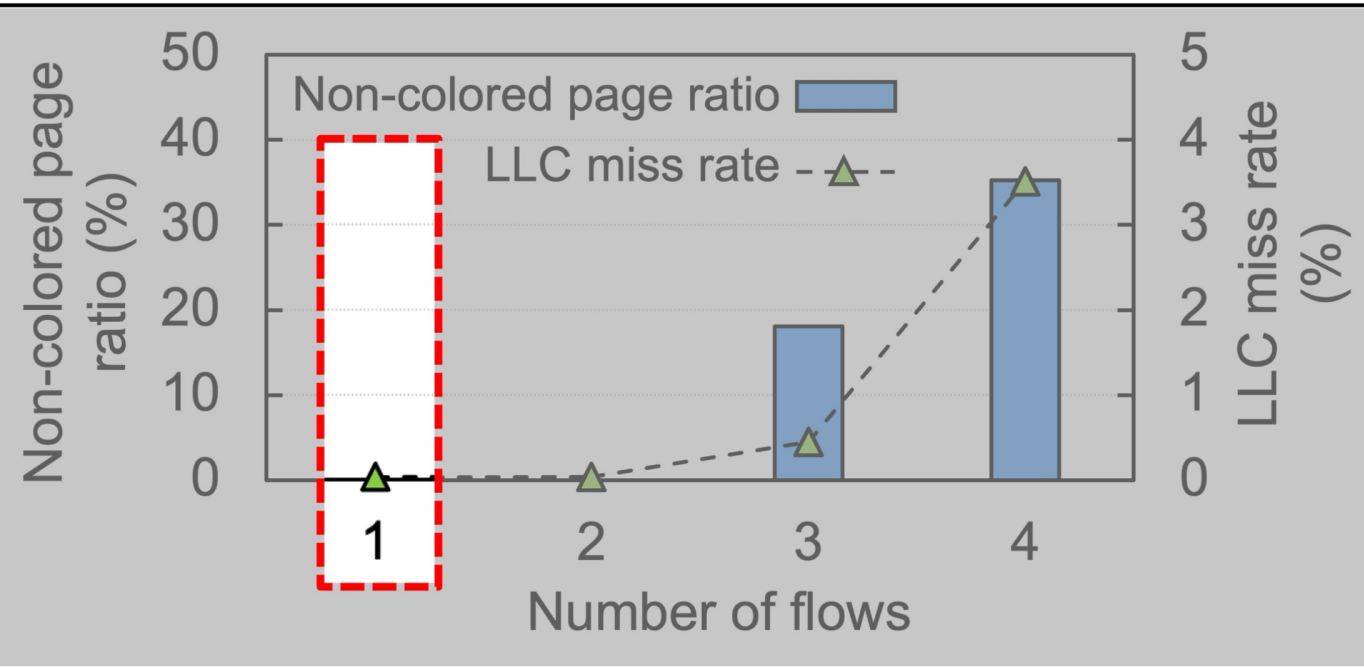
Performance under Bursty Workloads

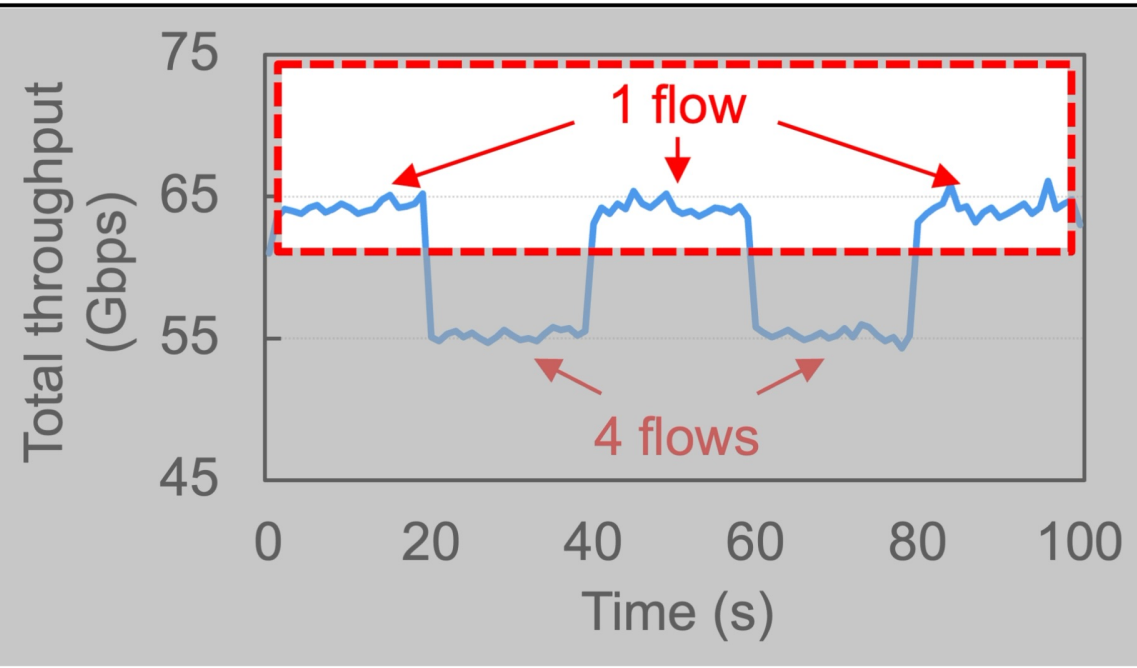




- **Exhaustion:** Safe fallback to standard Linux allocator

Performance under Bursty Workloads





- **Exhaustion:** Safe fallback to standard Linux allocator
- **Recovery:** Immediate restore of page coloring effectiveness

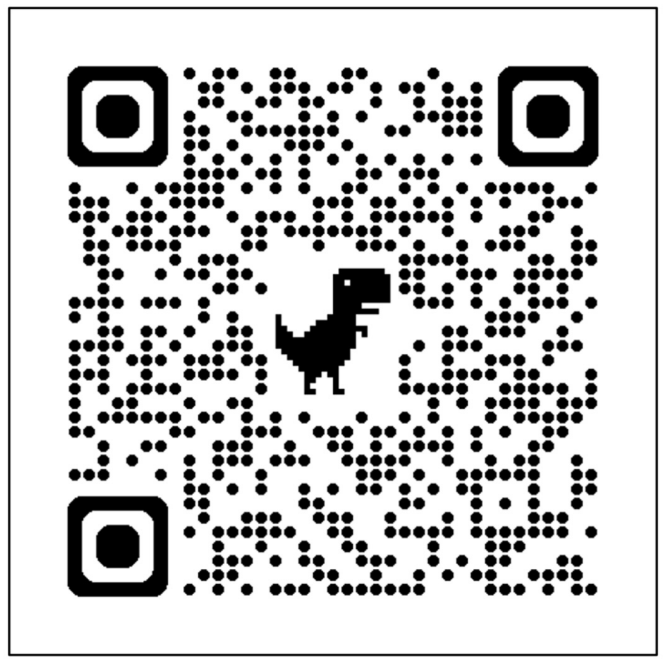
Conclusion

- **Beyond Capacity Limits:**
  - "Leaky DMA" is not only a capacity problem, but also a conflict-miss problem
- **Software-Hardware Co-design:**
  - Sepia: A practical, color-aware page allocator for network I/O
  - Set-balanced page placement that mitigates sliced-LLC conflicts
- **Extreme CPU Efficiency**
  - 200Gbps link saturation with **only 3.5 cores** (vs. 6 in Linux)
  - Near-zero LLC misses **(~0.4%**) by avoiding conflict and capacity pressure
  - up to **~1.51x** throughput-per-core with zero application modifications
- **Github:** https://github.com/skku-syslab/sepia

ARTIFACT EVALUATED
USENIX
AVAILABLE

Q&A

Github: https://github.com/skku-syslab/sepia



ARTIFACT EVALUATED
USENIX
AVAILABLE