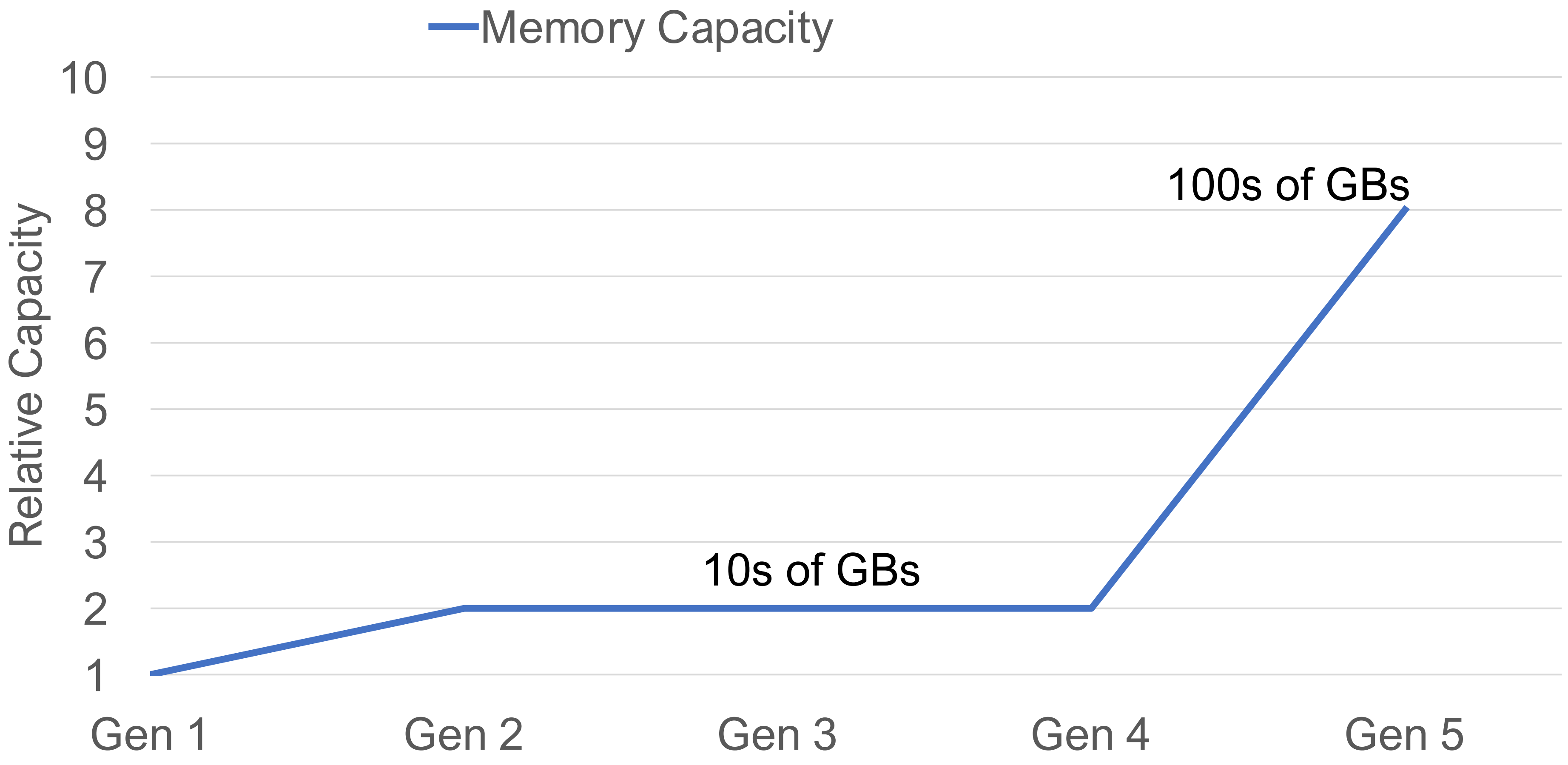
Contiguitas: The Pursuit of Physical Memory Contiguity in Datacenters

Kaiyang Zhao, Kaiwen Xue, Ziqi Wang, Dan Schatzberg, Leon Yang,
Antonis Manousis, Johannes Weiner, Rik van Riel, Bikash Sharma, Chunqiang Tang,
Dimitrios Skarlatos

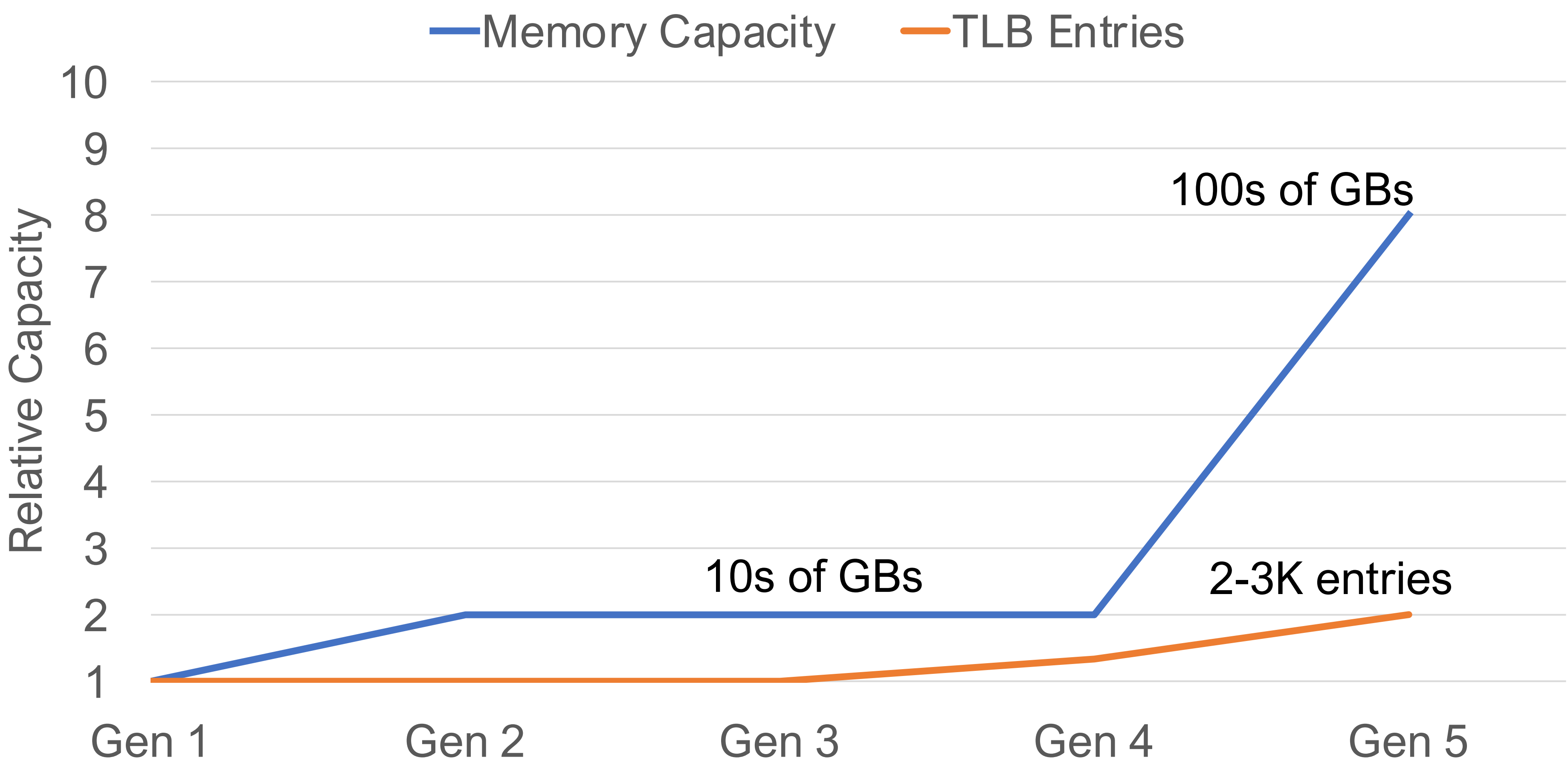
Supported by:

NSF intel Meta

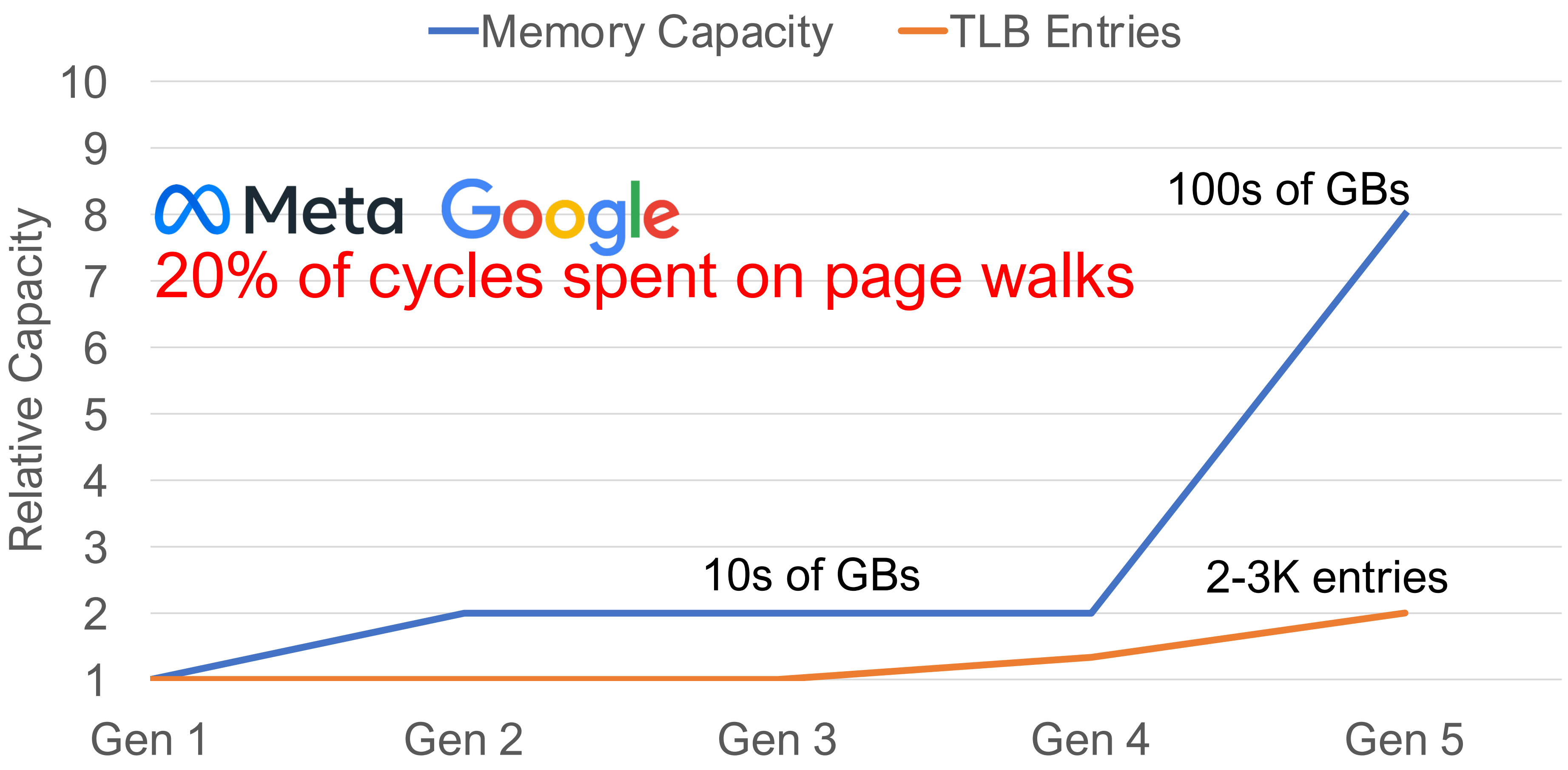
Memory Capacity Increases



TLB Does Not Scale



The Virtual Memory Bottleneck



20% of cycles spent on page walks

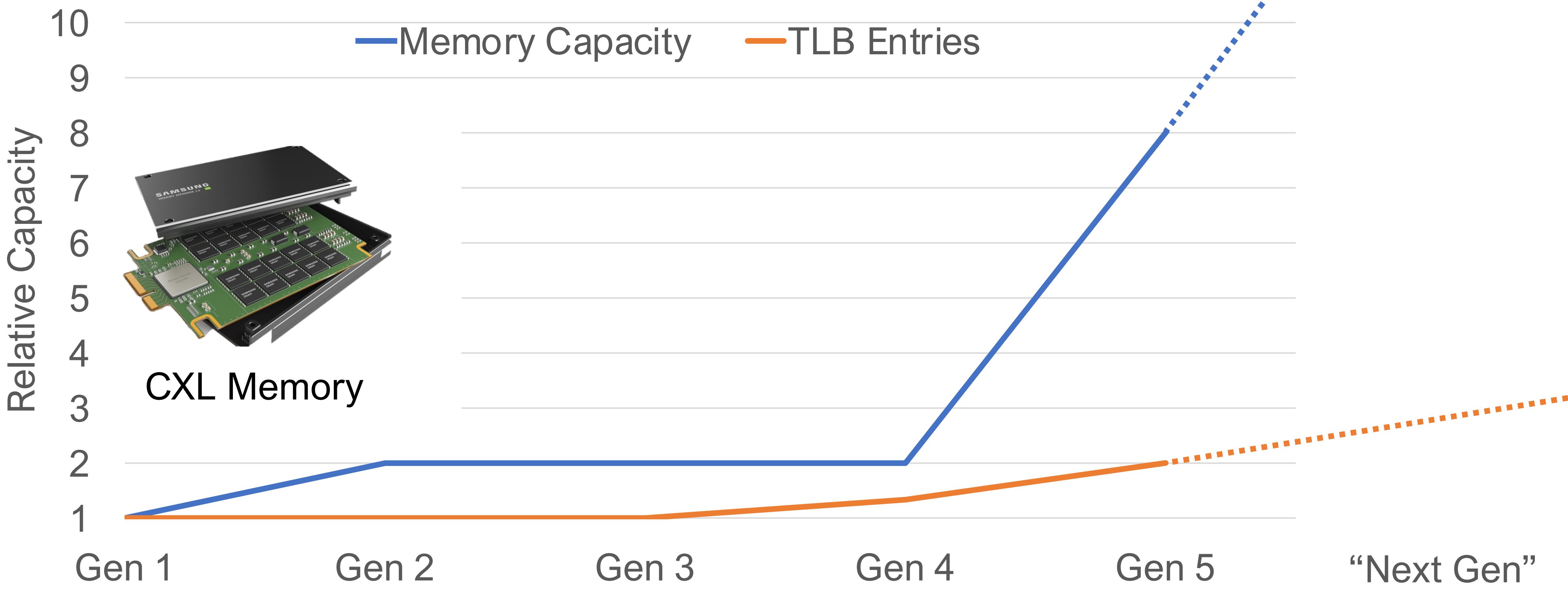
100s of GBs

10s of GBs

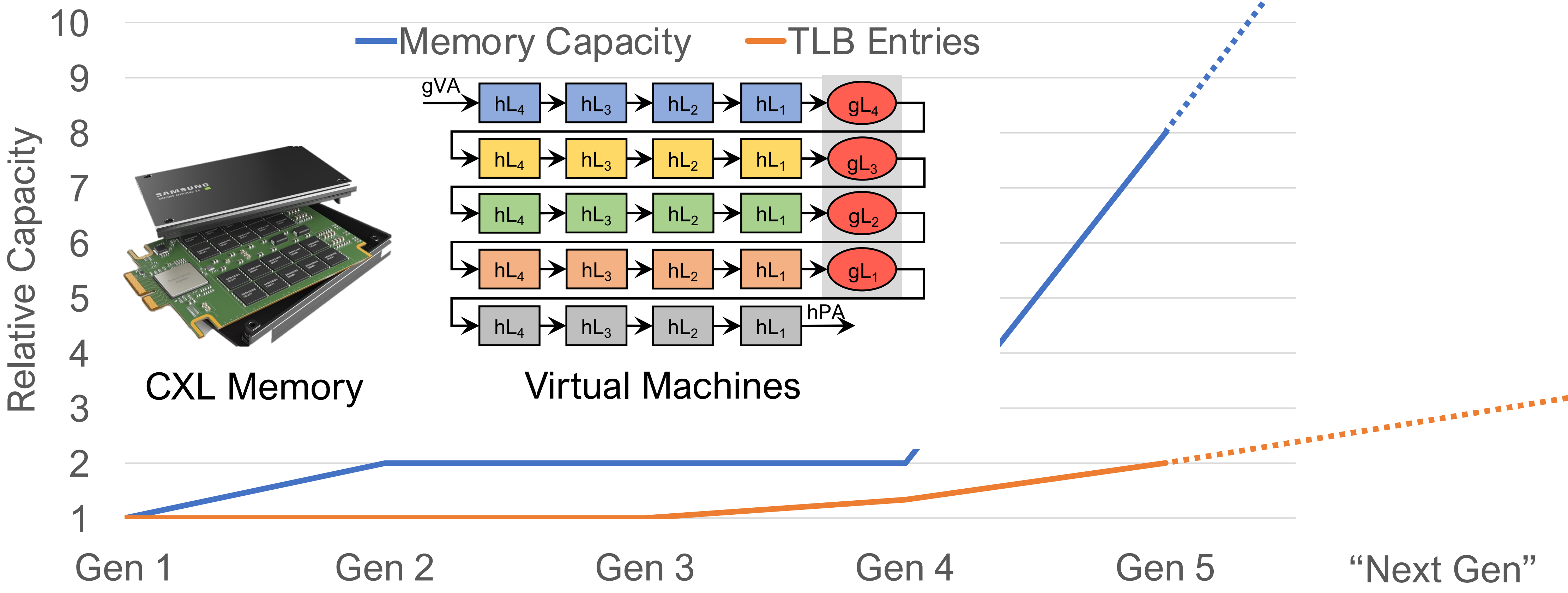
2-3K entries

* Beyond malloc efficiency to fleet efficiency: a hugepage-aware memory allocator, OSDI '21

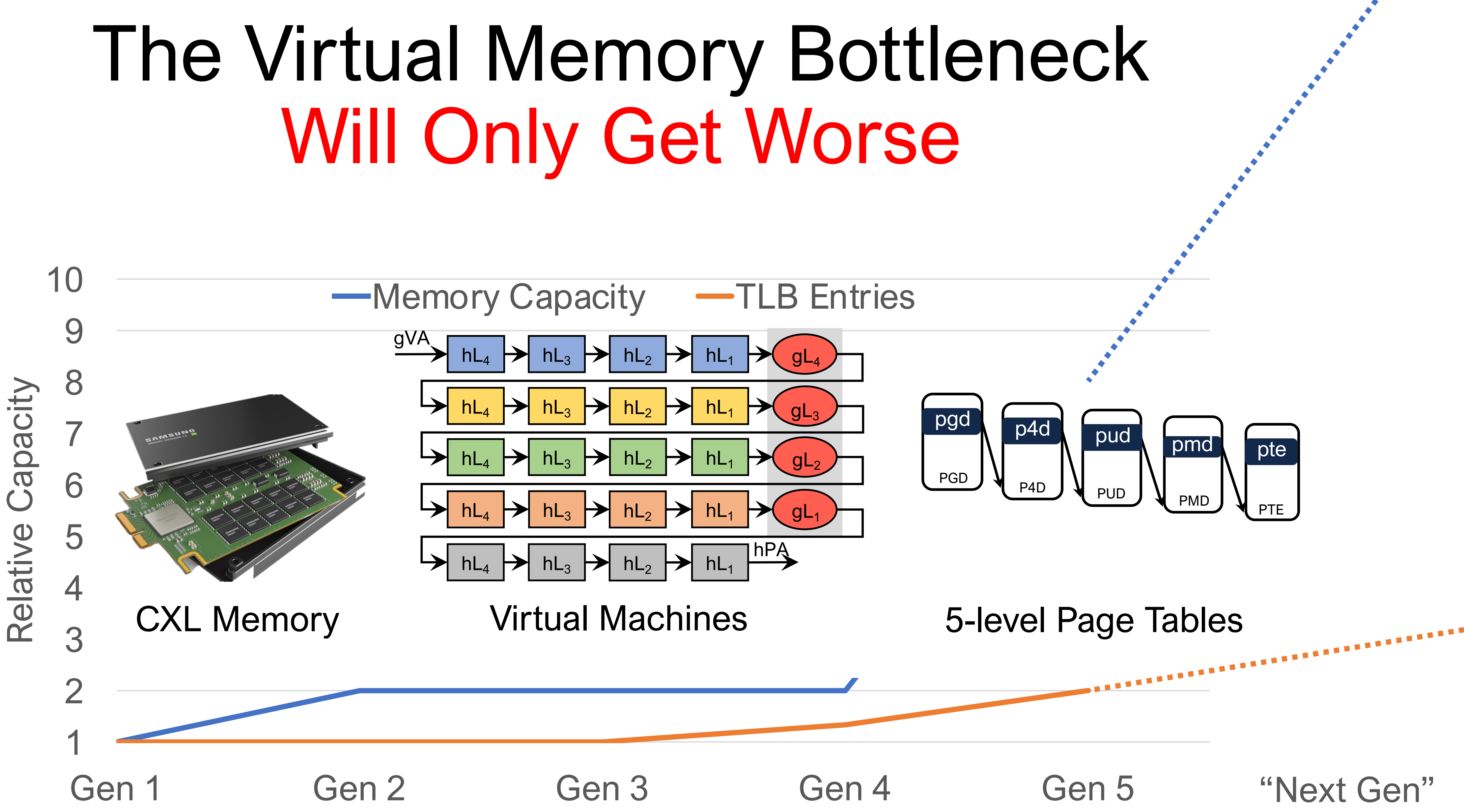
The Virtual Memory Bottleneck
Will Only Get Worse



The Virtual Memory Bottleneck
Will Only Get Worse

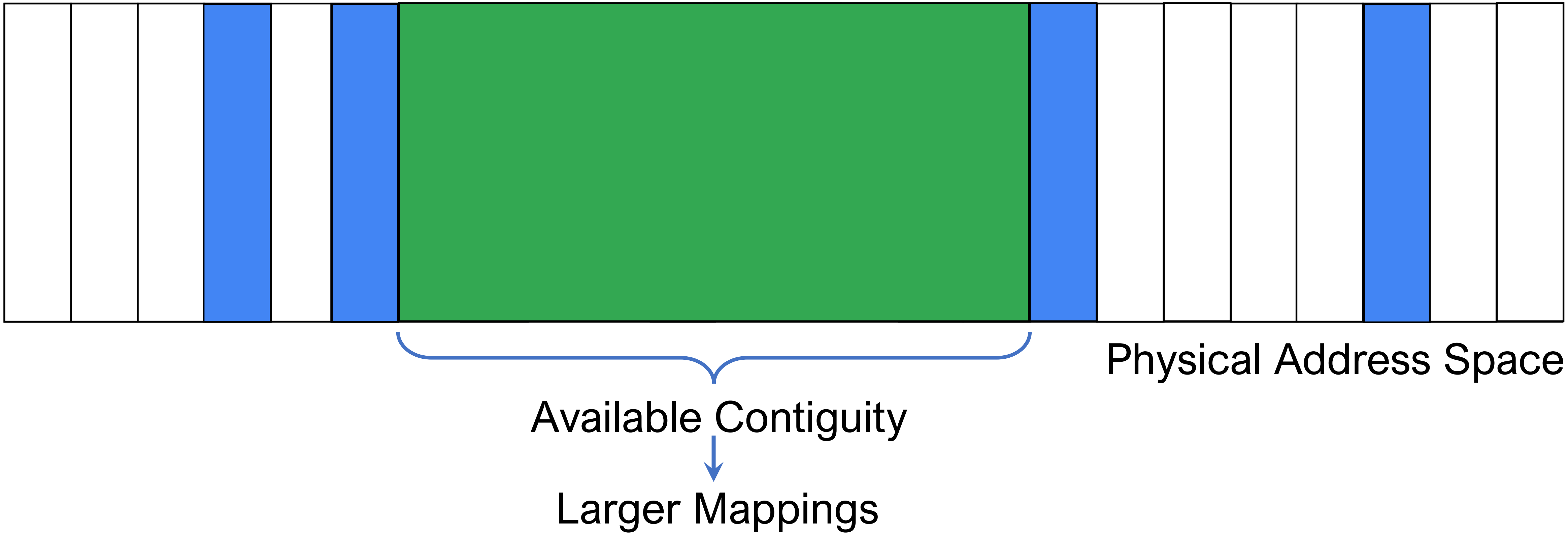


The Virtual Memory Bottleneck
Will Only Get Worse



Physical Memory Contiguity

Range of free space in memory that can back memory allocations



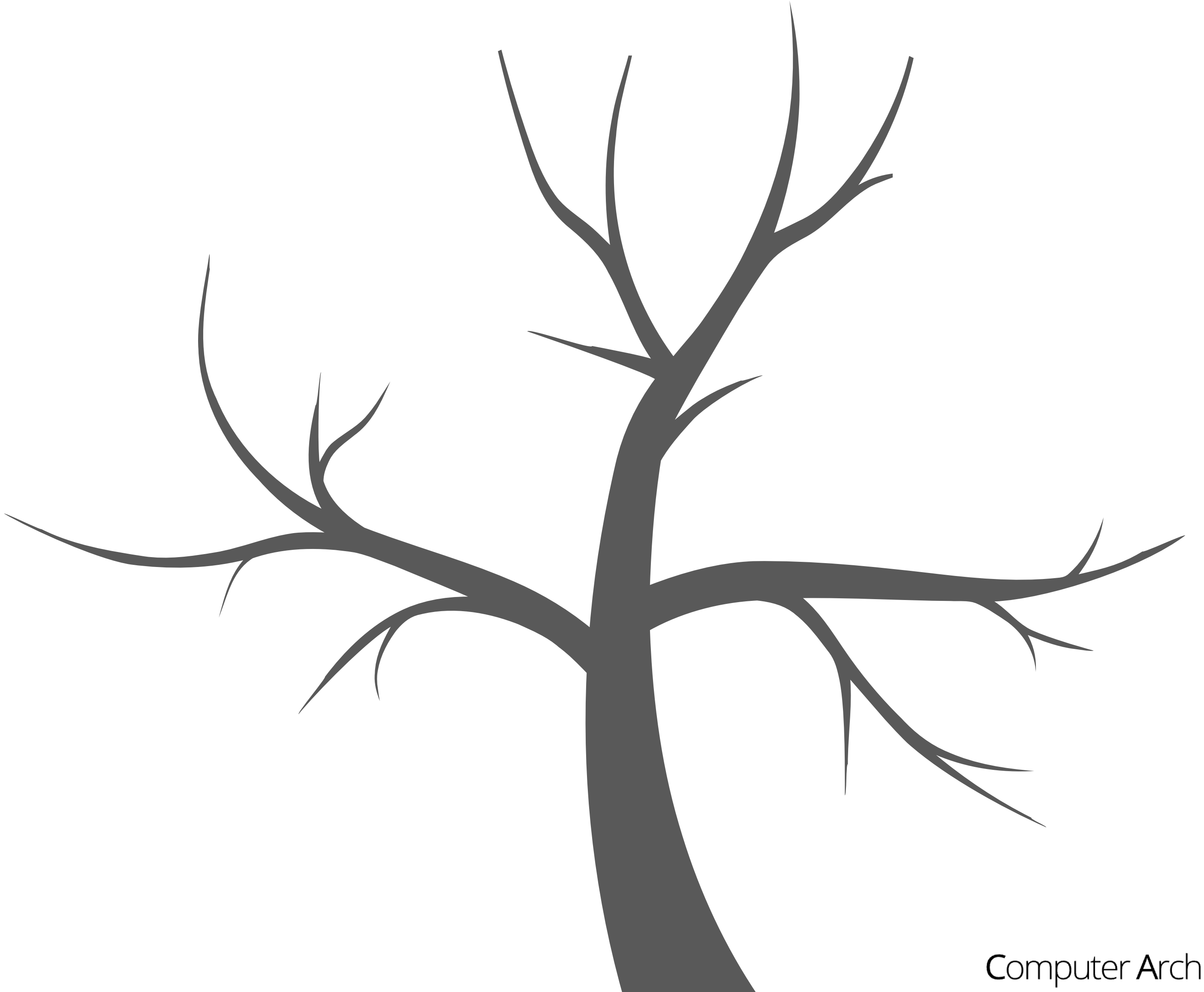
Physical Address Space

Available Contiguity

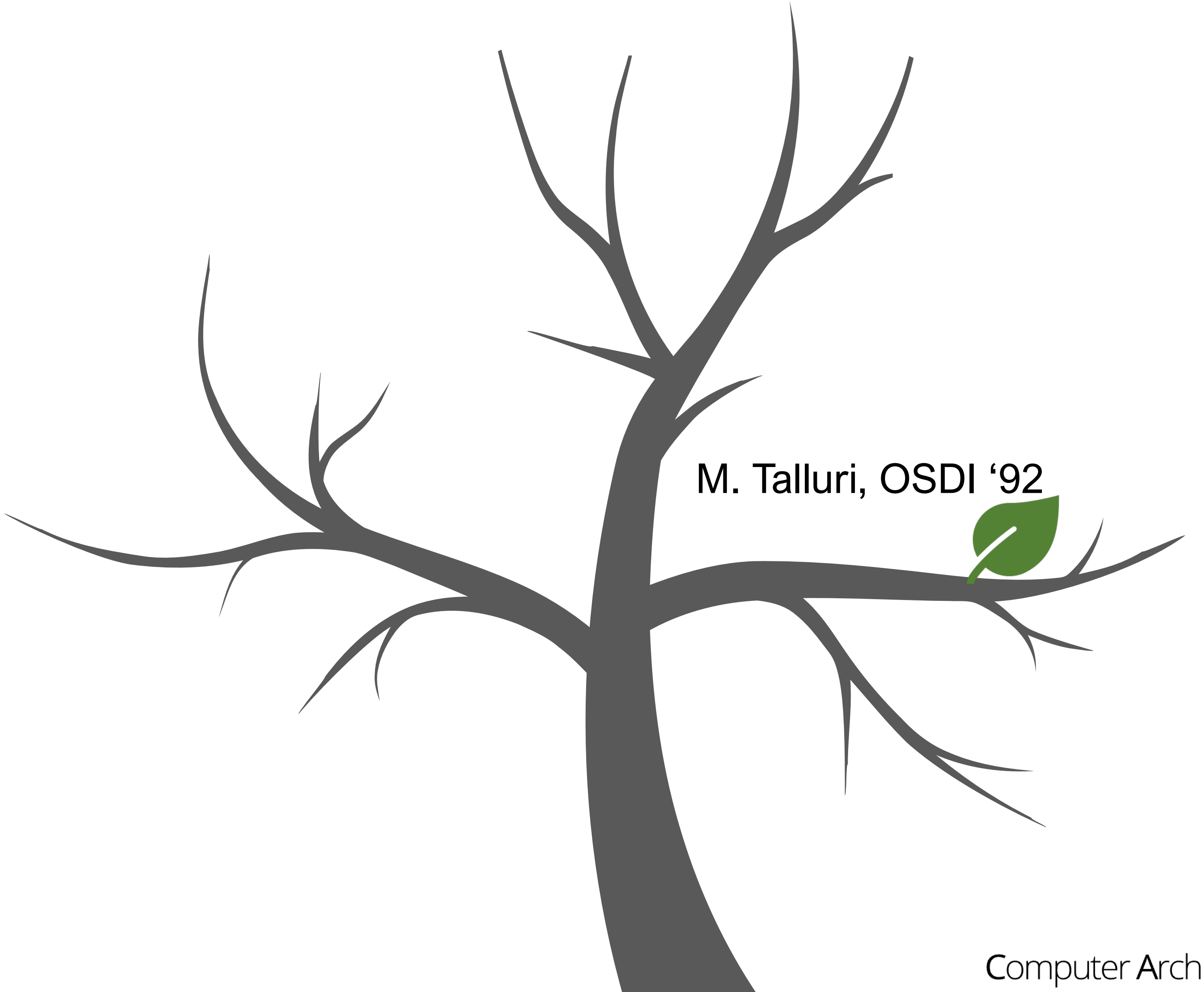
↓

Larger Mappings

Physical Memory Contiguity

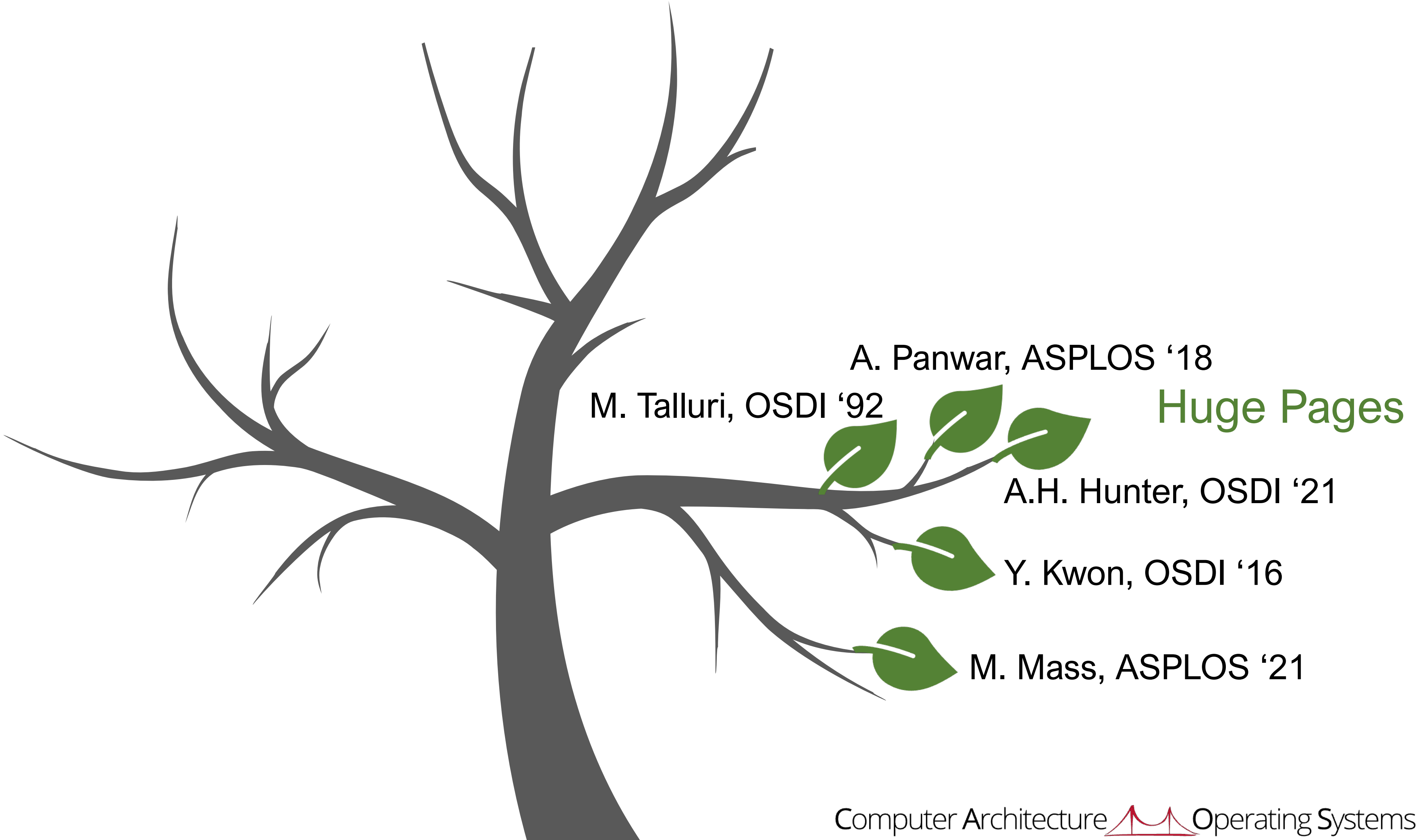


Physical Contiguity for Huge Pages

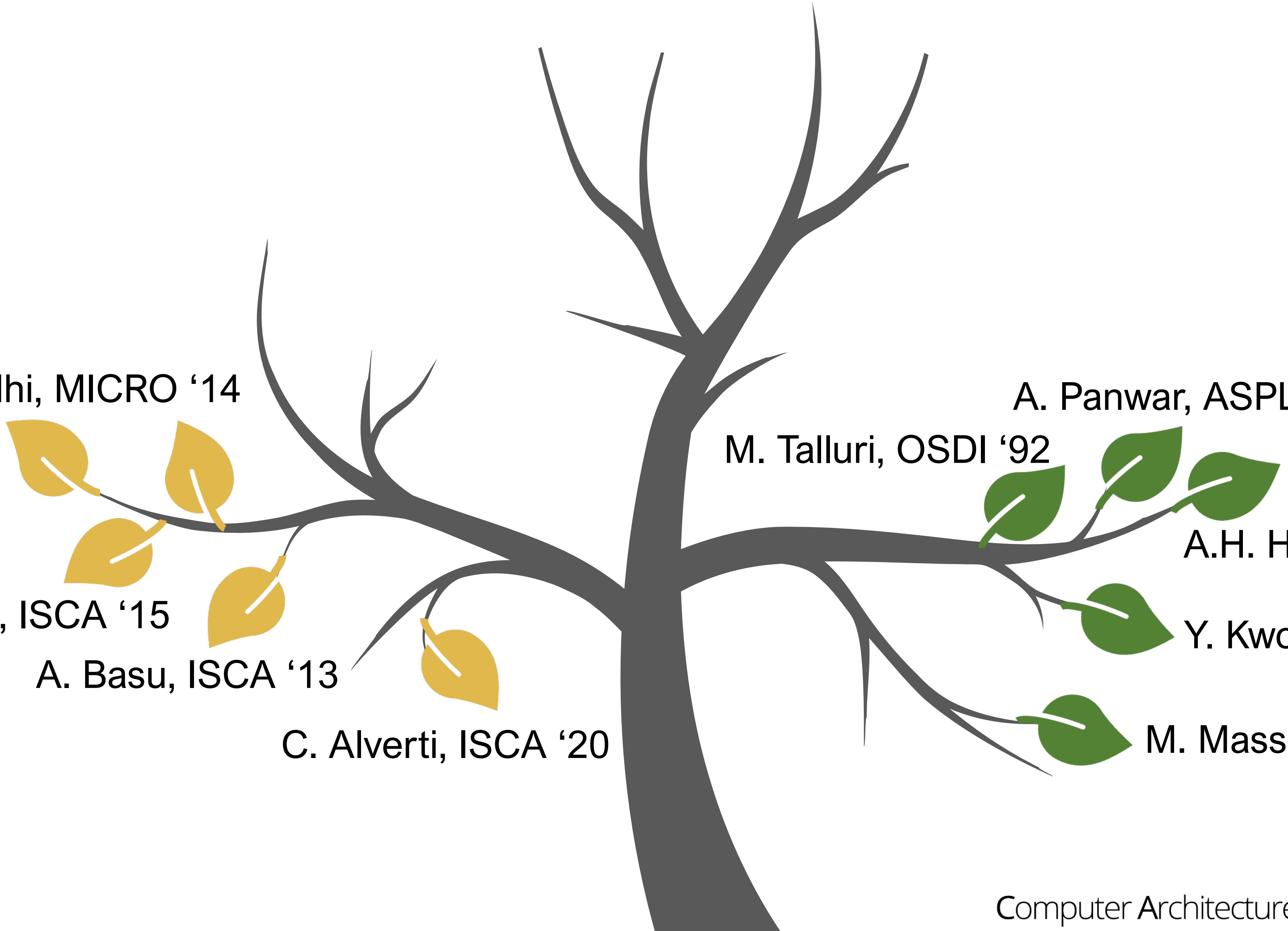


Huge Pages

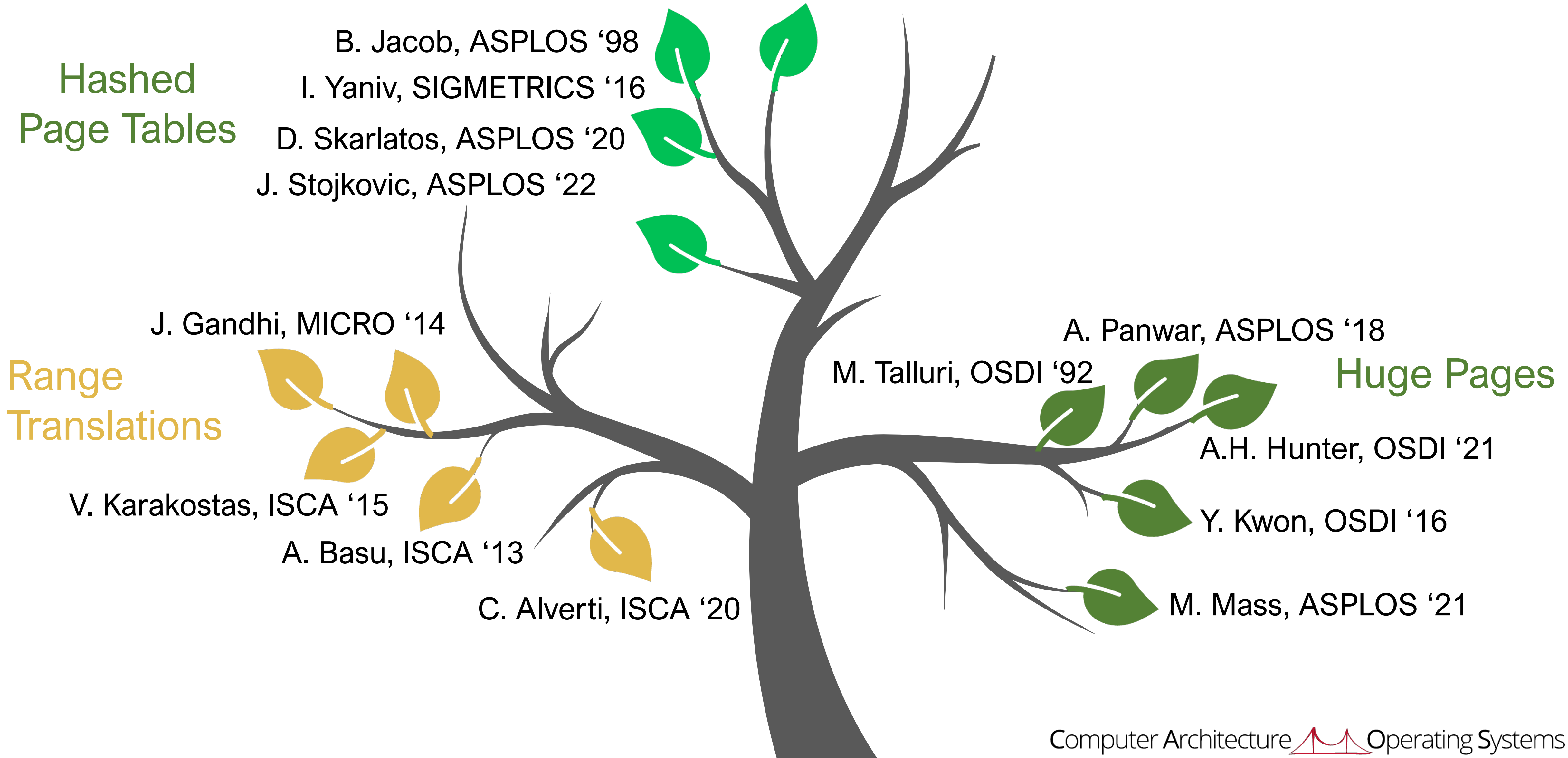
Physical Contiguity for Huge Pages



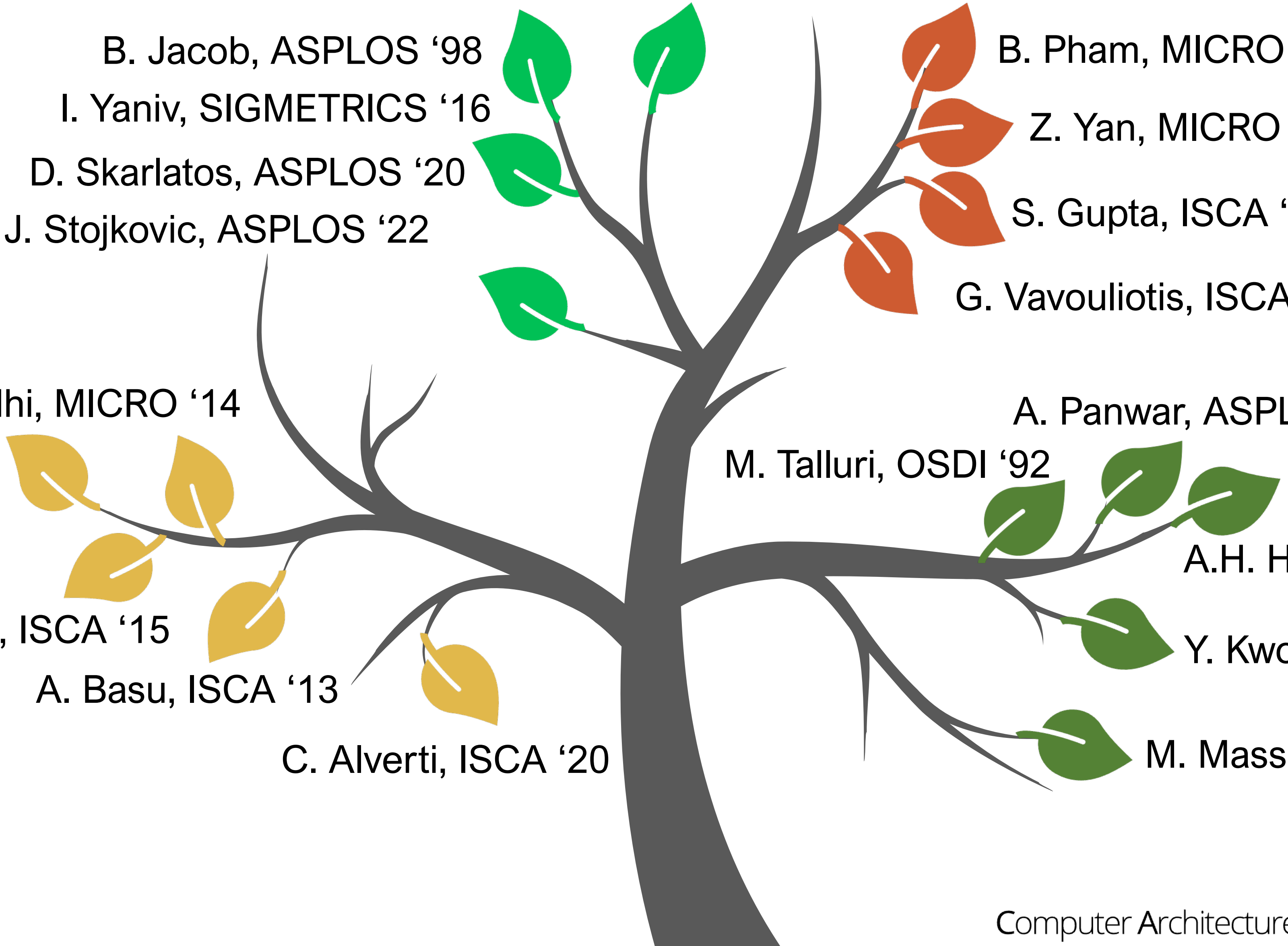
Physical Contiguity for Range Translations



Physical Contiguity for Hashed Page Tables



Physical Contiguity for Implicit Use Cases



Hashed Page Tables

- B. Jacob, ASPLOS ‘98
- I. Yaniv, SIGMETRICS ‘16
- D. Skarlatos, ASPLOS ‘20
- J. Stojkovic, ASPLOS ‘22

Implicit Uses

- B. Pham, MICRO ‘12
- Z. Yan, MICRO ‘19
- S. Gupta, ISCA ‘21
- G. Vavouliotis, ISCA ‘21

Range Translations

- J. Gandhi, MICRO ‘14
- V. Karakostas, ISCA ‘15
- A. Basu, ISCA ‘13
- C. Alverti, ISCA ‘20

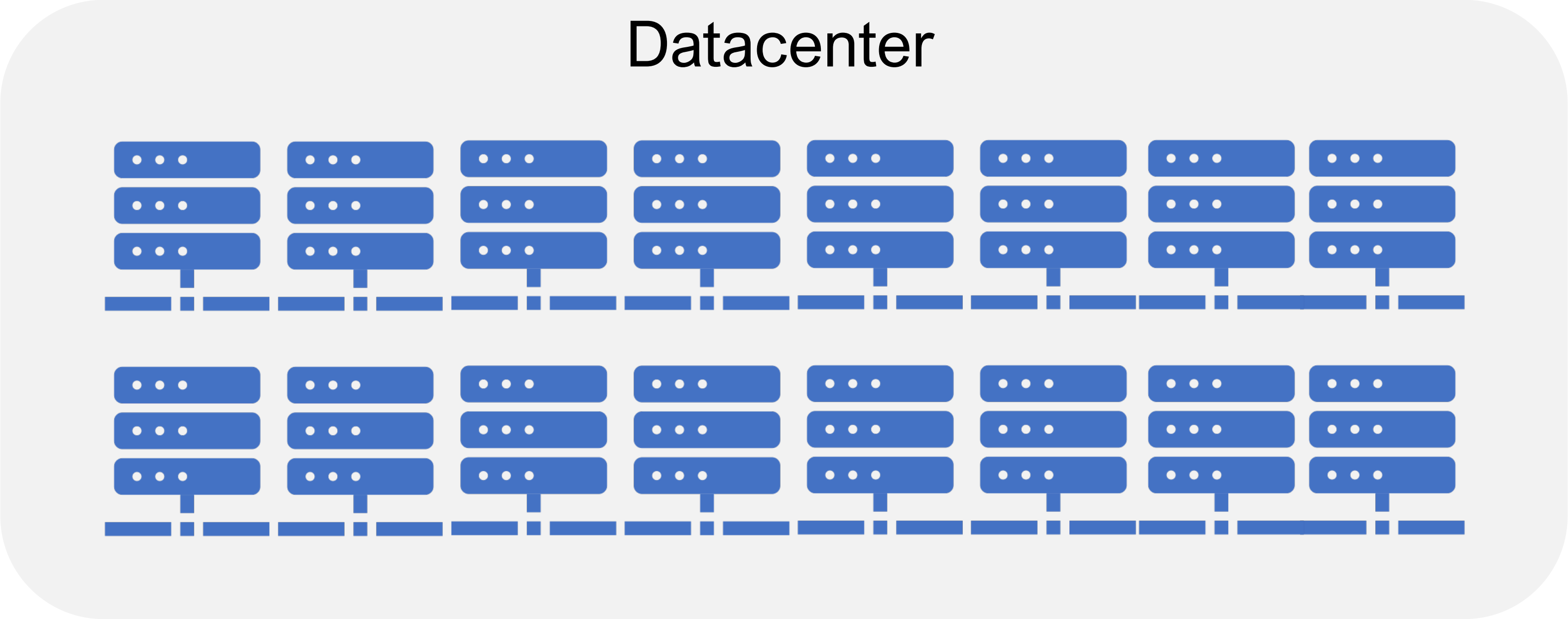
Huge Pages

- A. Panwar, ASPLOS ‘18
- M. Talluri, OSDI ‘92
- A.H. Hunter, OSDI ‘21
- Y. Kwon, OSDI ‘16
- M. Mass, ASPLOS ‘21

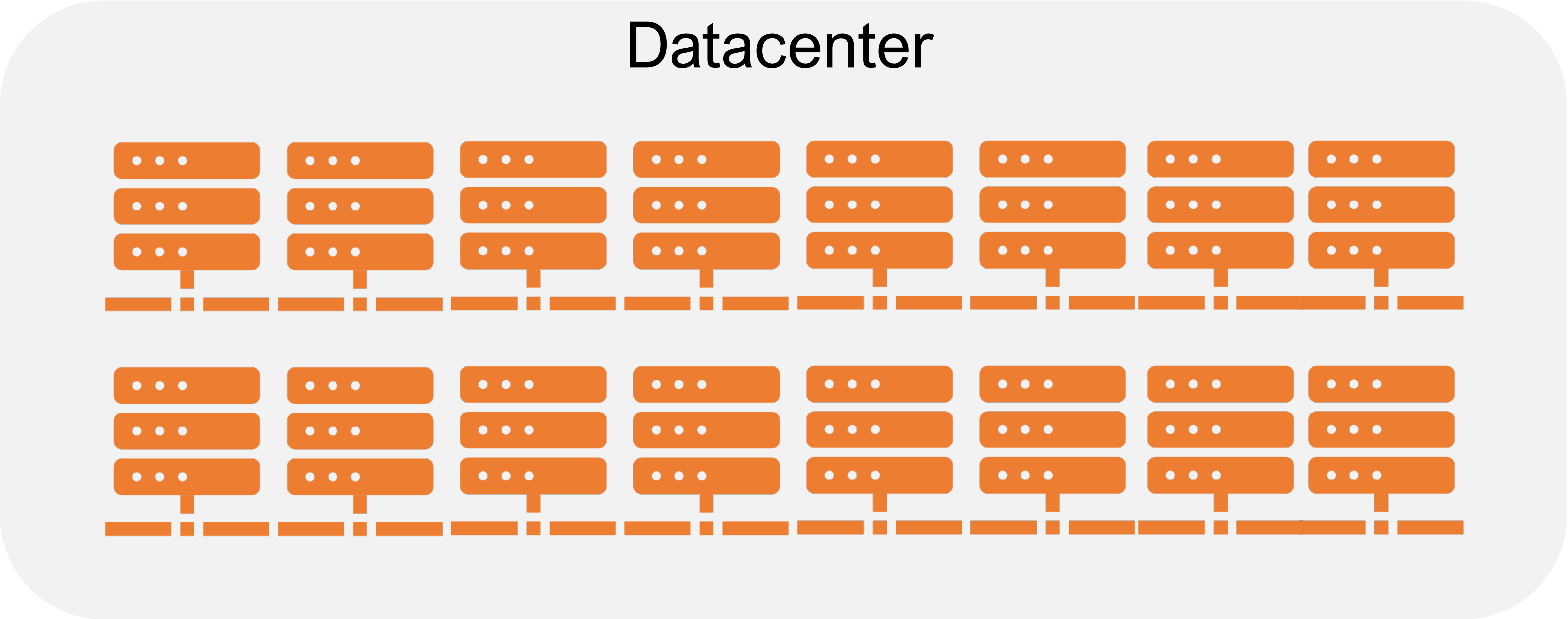
But is There Contiguity in Datacenters?

But is There Contiguity in Datacenters?

Looking for contiguity across Meta's fleet

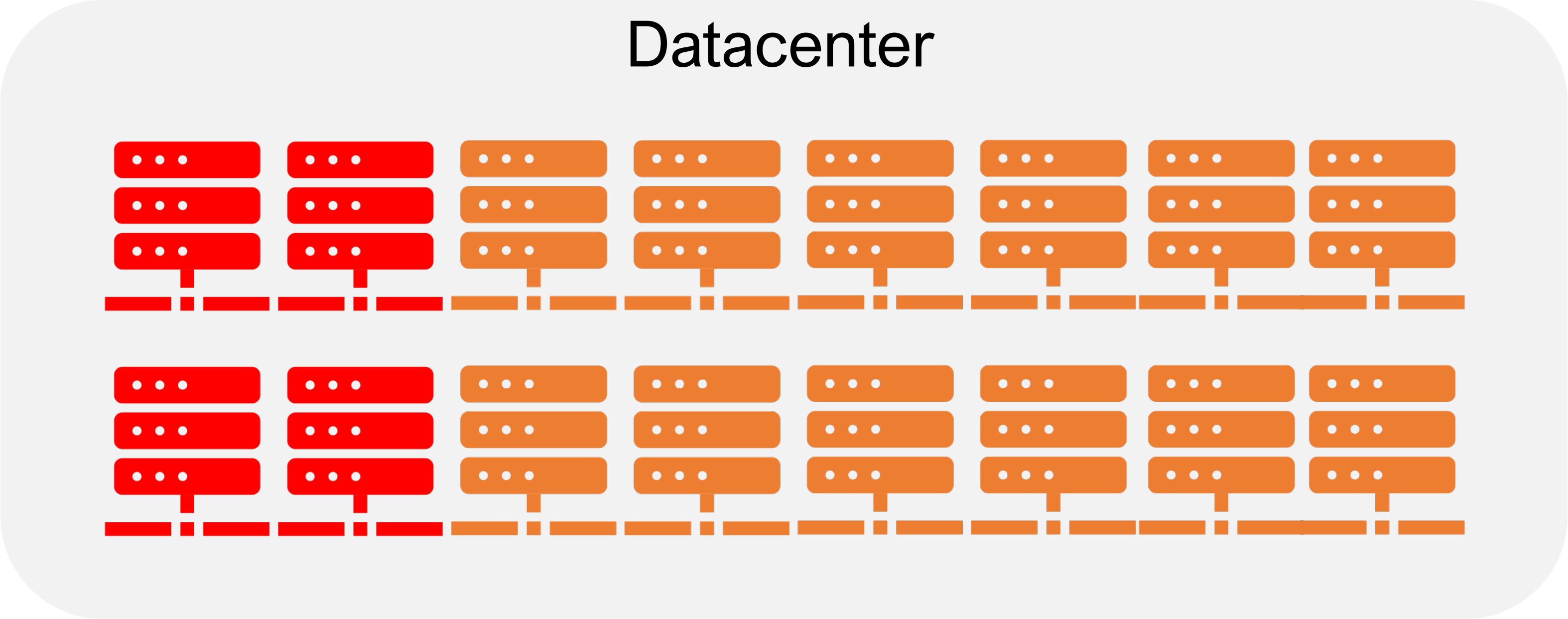


Servers Are Highly Fragmented



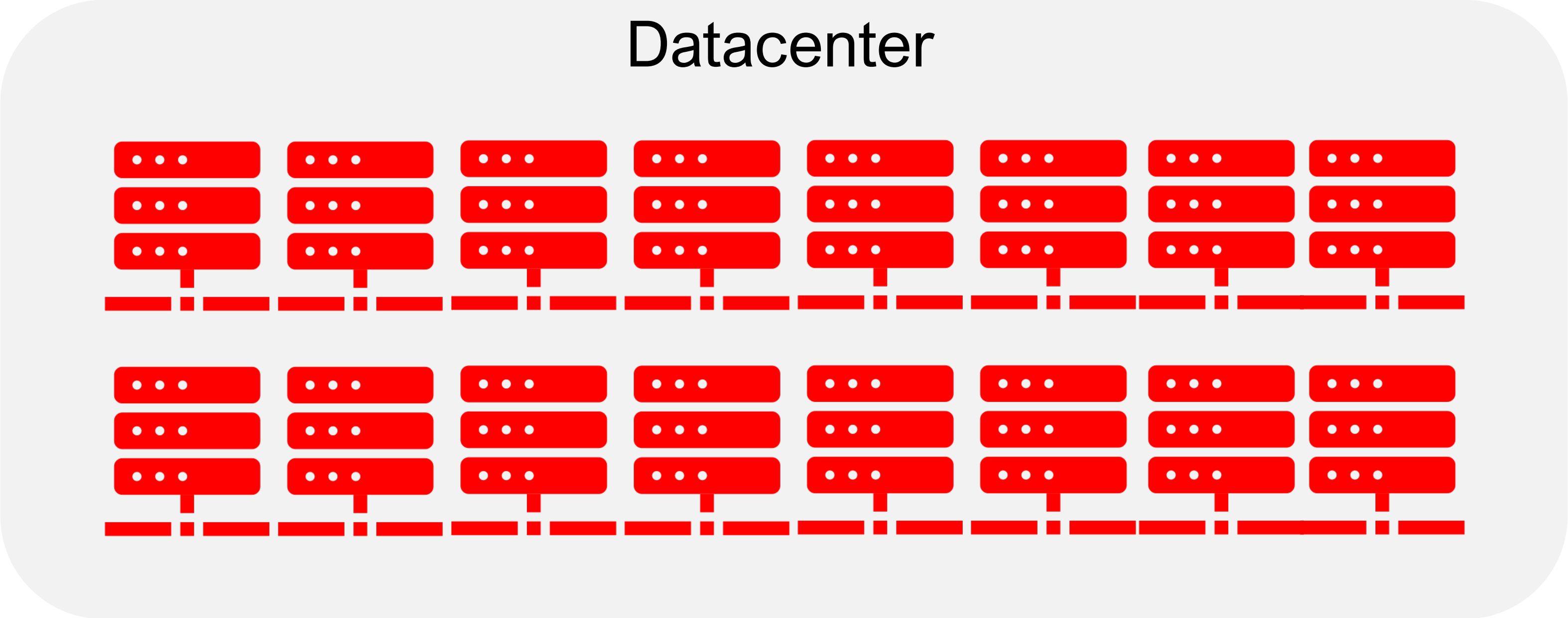
Servers Are Highly Fragmented

A quarter of servers don’t have contiguity for *even a single 2MB page*!

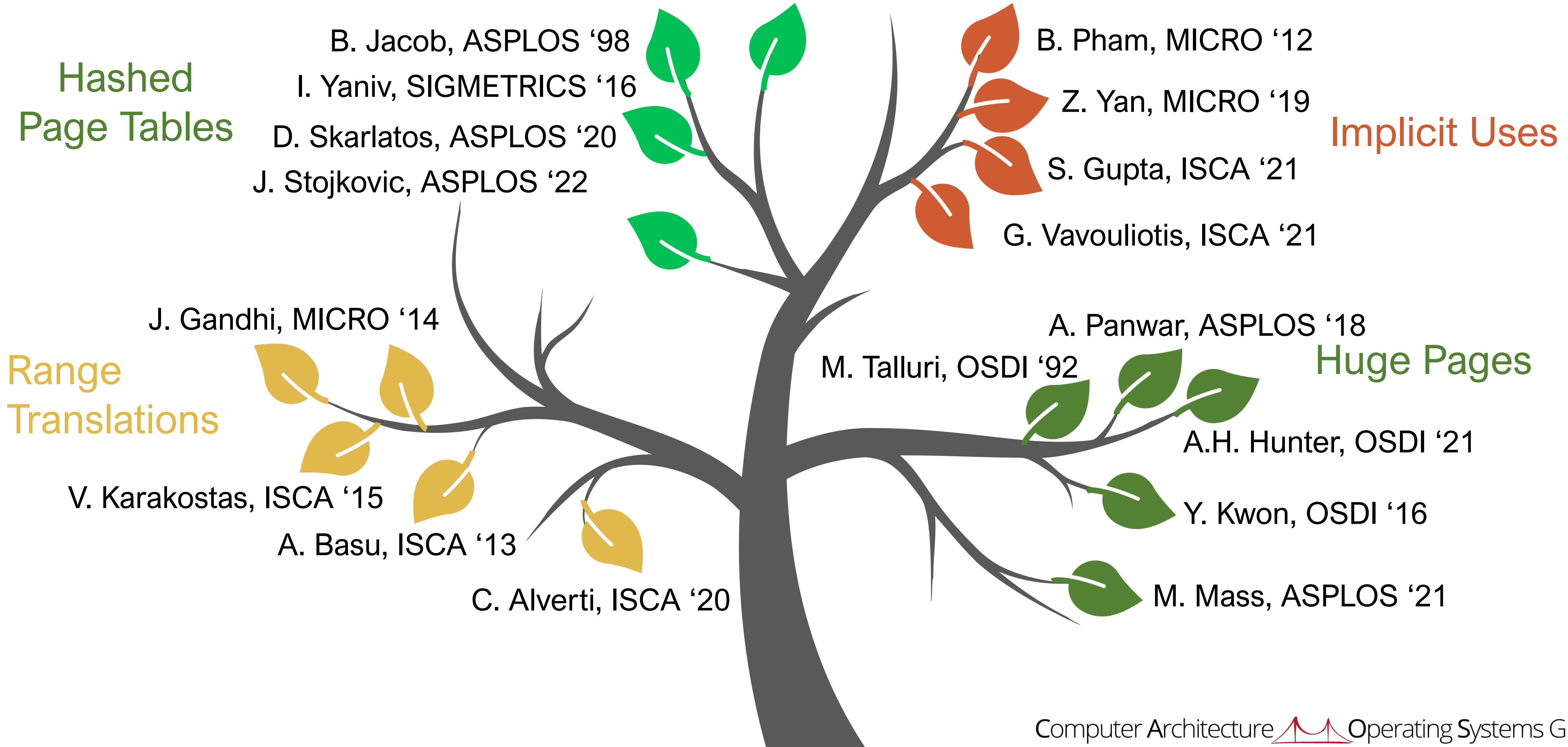


Servers Are Highly Fragmented

Allocating 1GB pages is impossible in production!



Without Physical Contiguity



Hashed Page Tables

B. Jacob, ASPLOS ‘98

I. Yaniv, SIGMETRICS ‘16

D. Skarlatos, ASPLOS ‘20

J. Stojkovic, ASPLOS ‘22

Implicit Uses

B. Pham, MICRO ‘12

Z. Yan, MICRO ‘19

S. Gupta, ISCA ‘21

G. Vavouliotis, ISCA ‘21

Range Translations

J. Gandhi, MICRO ‘14

V. Karakostas, ISCA ‘15

A. Basu, ISCA ‘13

C. Alverti, ISCA ‘20

Huge Pages

A. Panwar, ASPLOS ‘18

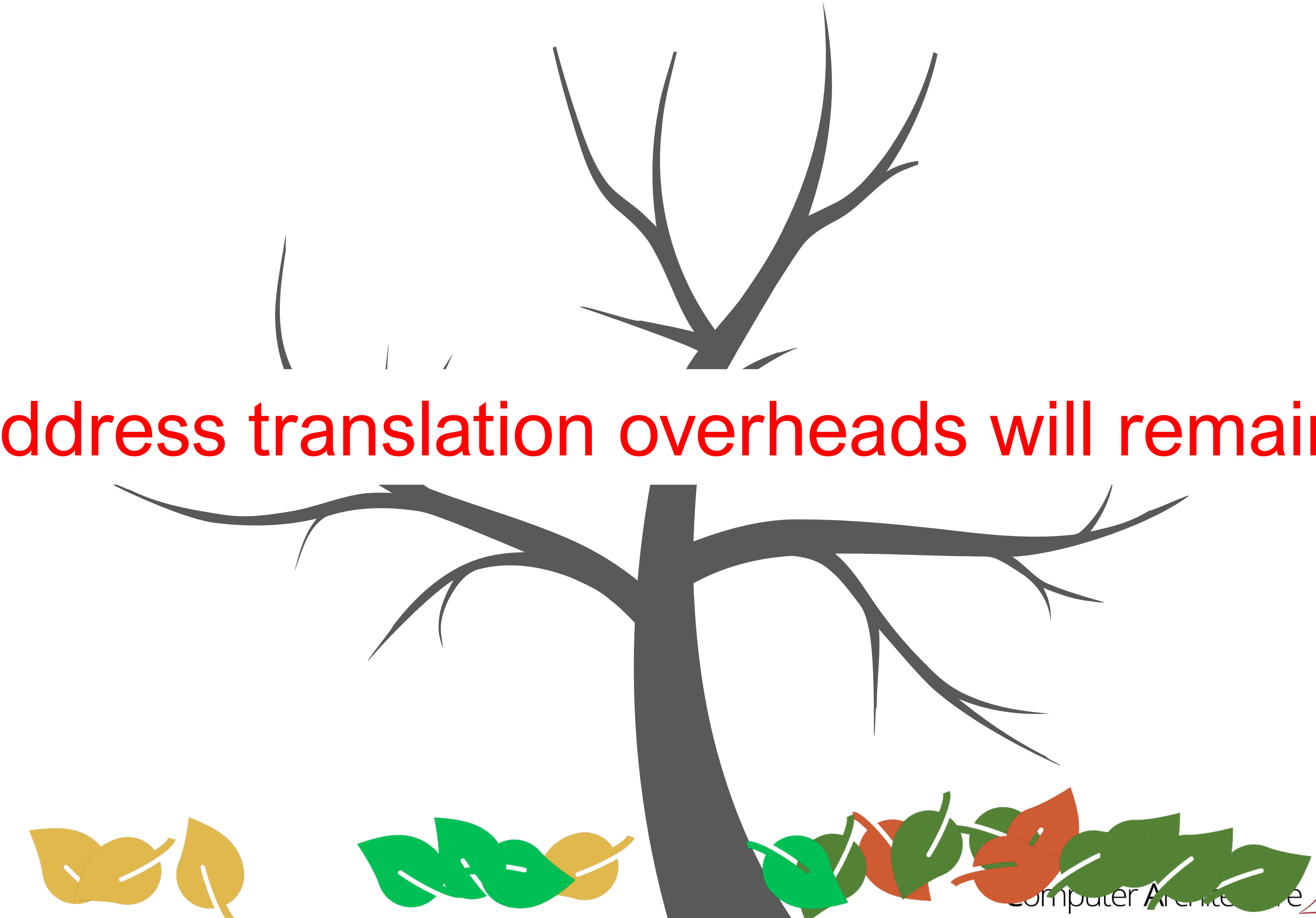
M. Talluri, OSDI ‘92

A.H. Hunter, OSDI ‘21

Y. Kwon, OSDI ‘16

M. Mass, ASPLOS ‘21

Without Physical Contiguity



Address translation overheads will remain high!

Contribution: Contiguitas

Contribution: Contiguitas

Unmovable pages are detrimental to contiguity

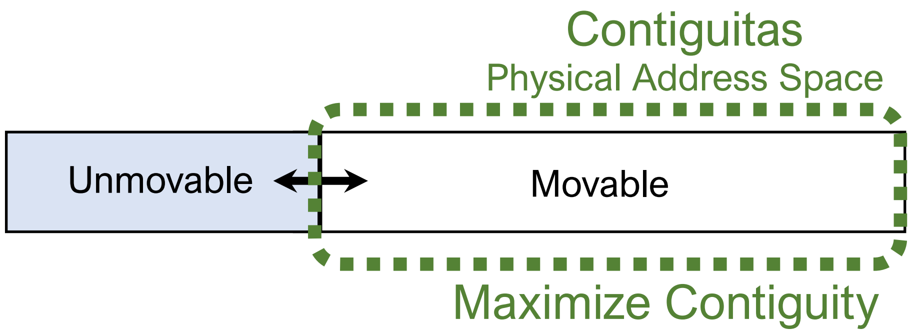
- I/O pages → Networking, RDMA, GPUs, Accelerators

Contribution: Contiguitas

Unmovable pages are detrimental to contiguity

- I/O pages → Networking, RDMA, GPUs, Accelerators

Redesign OS memory management



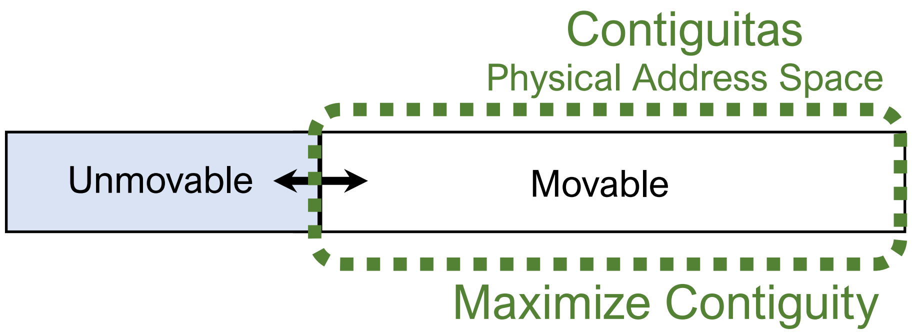
Contribution: Contiguitas

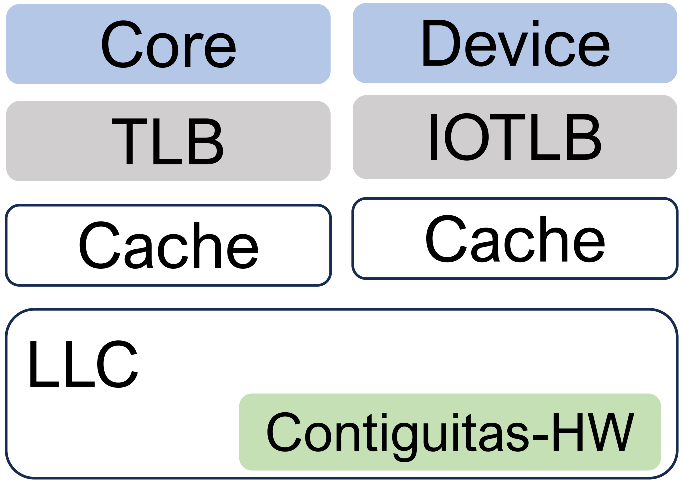
Unmovable pages are detrimental to contiguity

- I/O pages → Networking, RDMA, GPUs, Accelerators

Redesign OS memory management

HW support for unmovable page migration

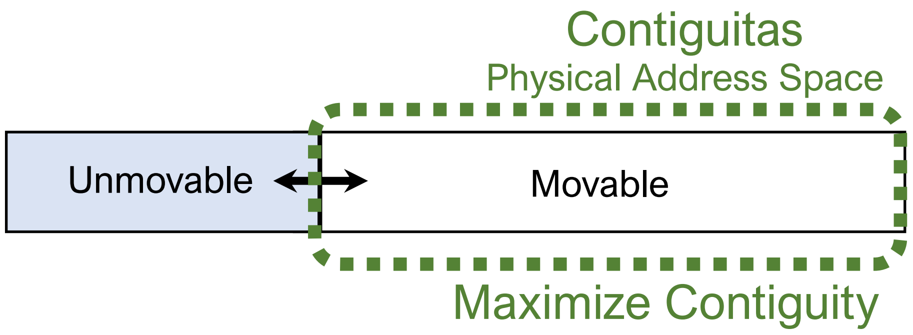




Contribution: Contiguitas

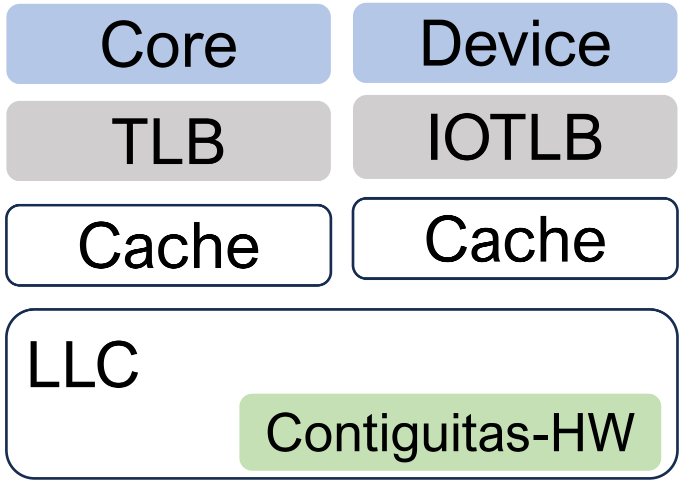
Unmovable pages are detrimental to contiguity

- I/O pages → Networking, RDMA, GPUs, Accelerators



Redesign OS memory management

HW support for unmovable page migration



Ample physical memory contiguity

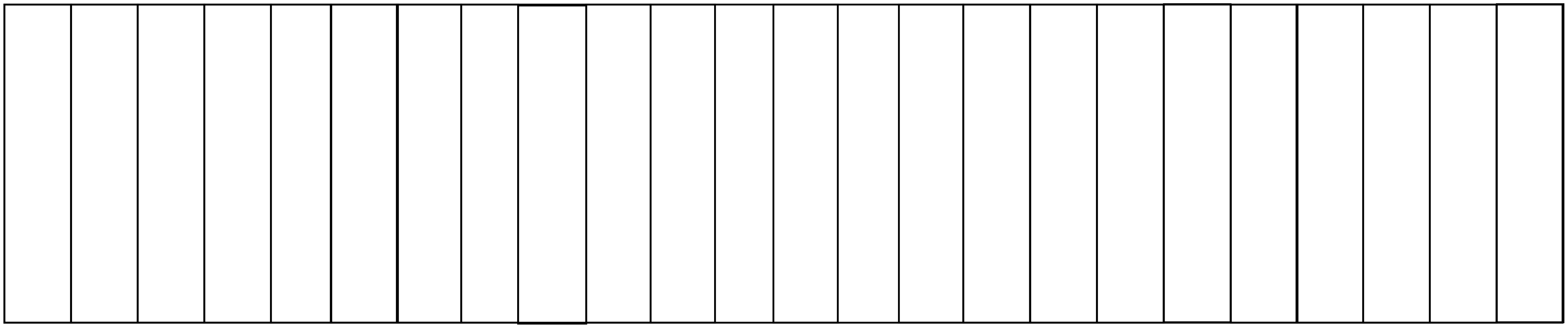
- ~90% of total memory

Performance gains 2-18% with production workloads

Efficiently reduce unmovable pages with HW

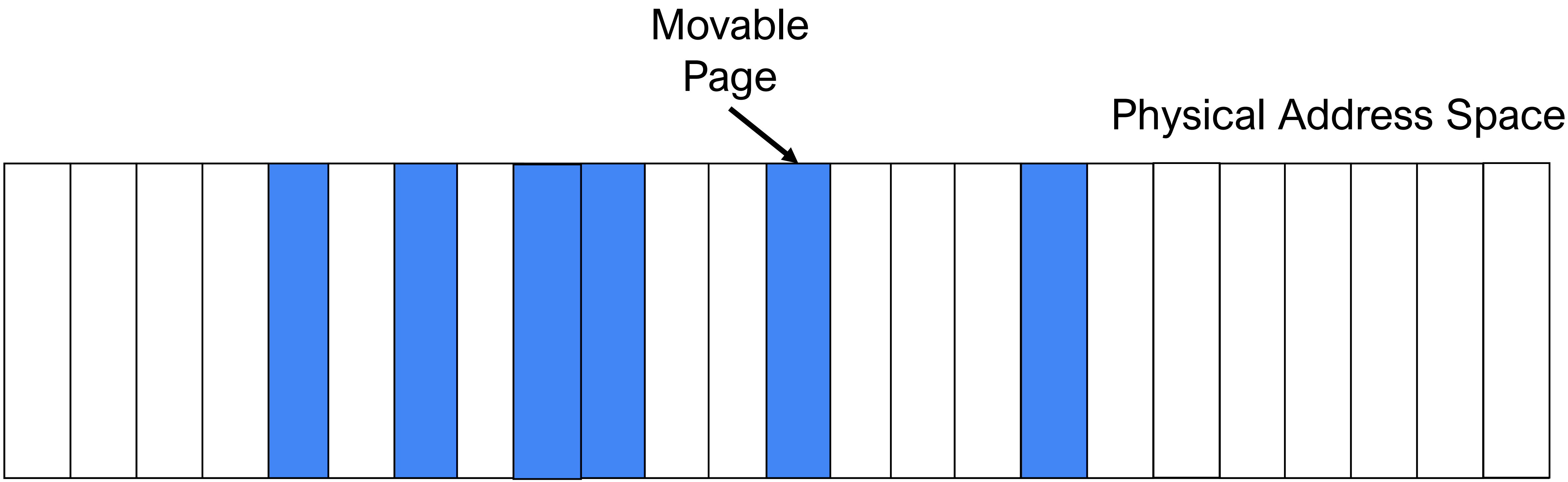
In the process of upstreaming to Linux

Current Physical Address Space



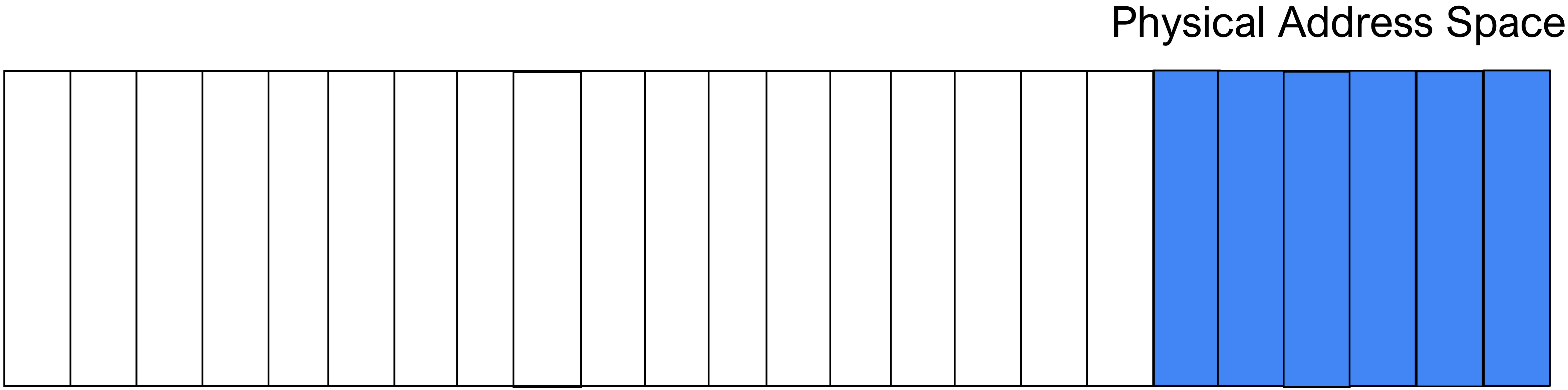
Physical Address Space

Current Physical Address Space



🟦 Movable: Can be defragmented

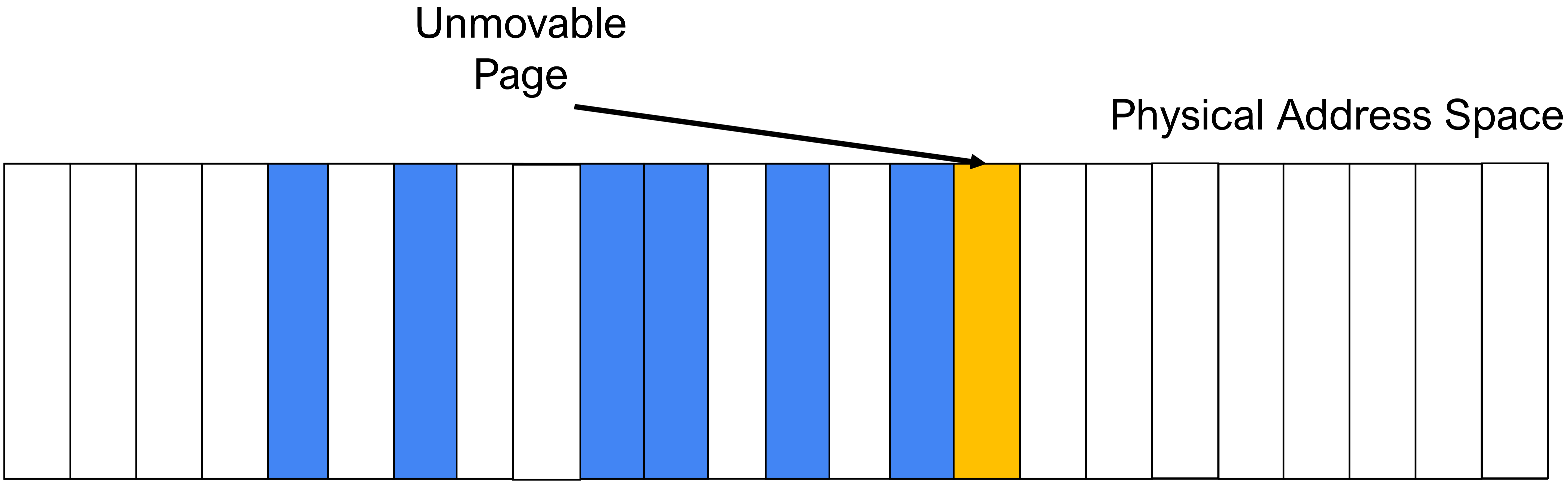
Memory Defragmentation



Physical memory contiguity can be formed by defragmentation

🟦 Movable: Can be defragmented

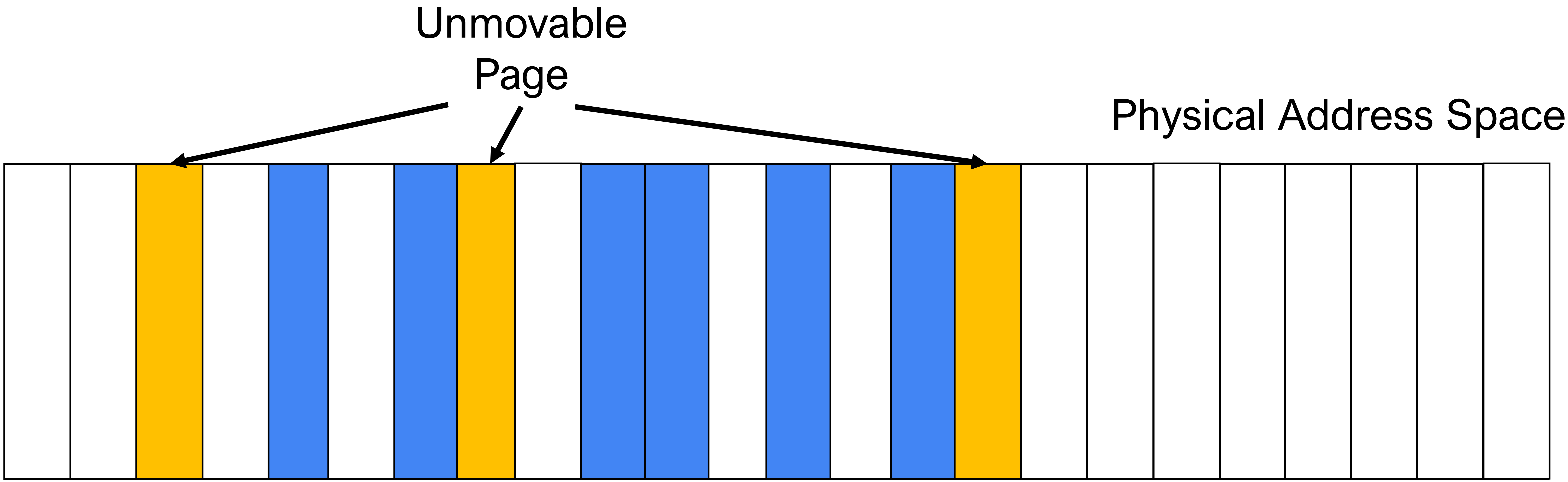
Unmovable Pages Block Contiguity



🟨 Unmovable: Cannot be moved by the OS

🟦 Movable: Can be defragmented

Unmovable Pages Block Contiguity



Unmovable Page

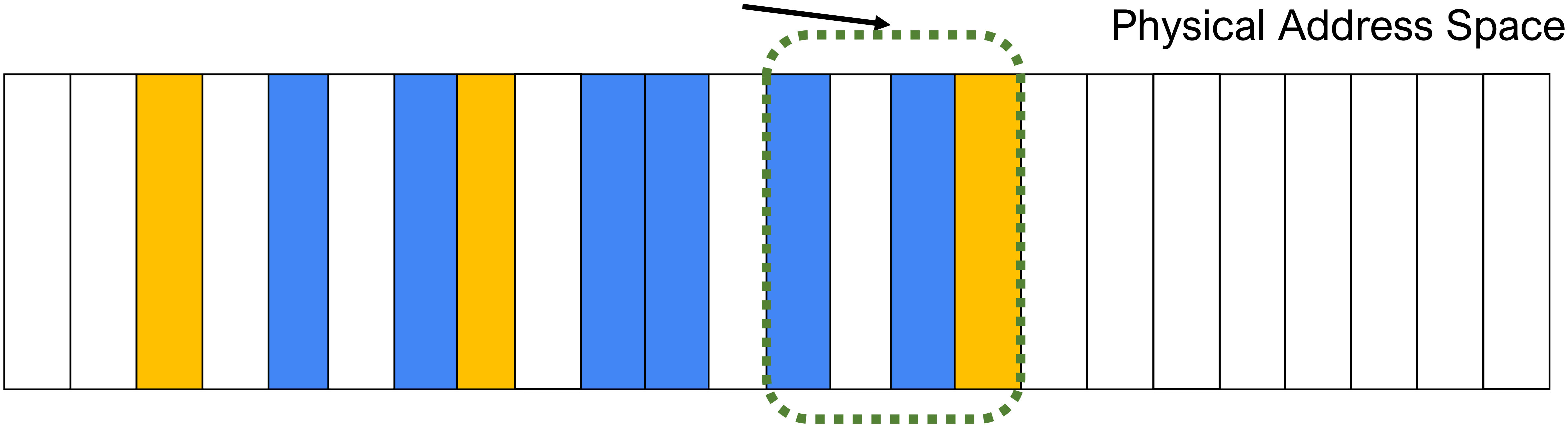
Physical Address Space

Unmovable pages cannot be moved by the OS → Block contiguity!

🟨 Unmovable: Cannot be moved by the OS

🟦 Movable: Can be defragmented

Unmovable Pages Block Contiguity



2MB range

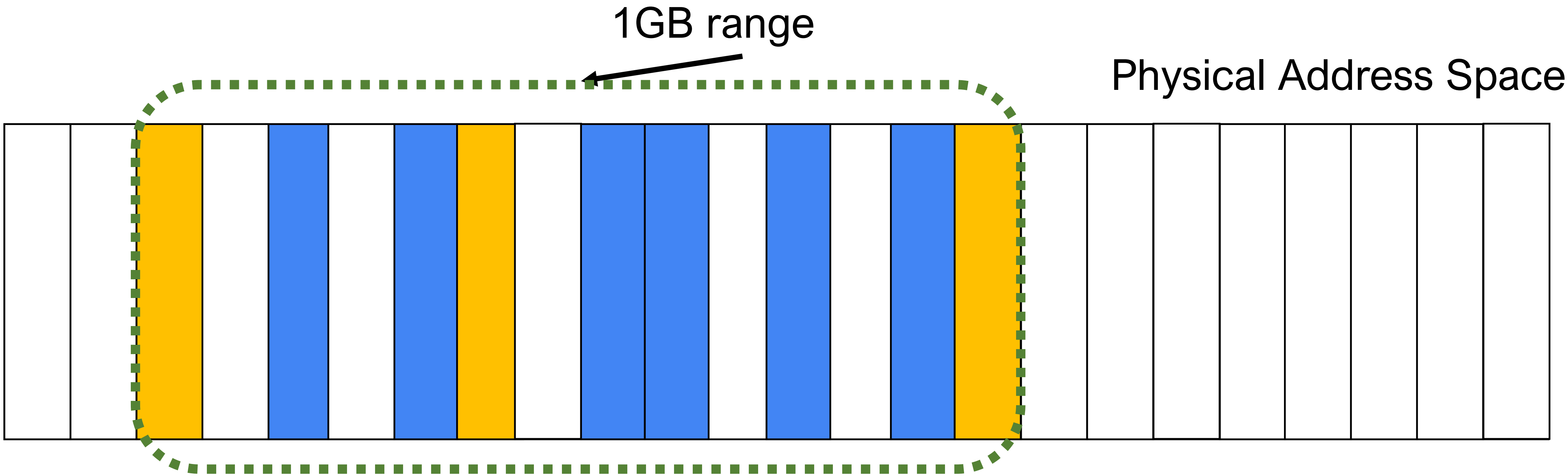
Physical Address Space

One 4KB unmovable page renders a 2MB or 1GB range unmovable

🟨 Unmovable: Cannot be moved by the OS

🟦 Movable: Can be defragmented

Unmovable Pages Block Contiguity

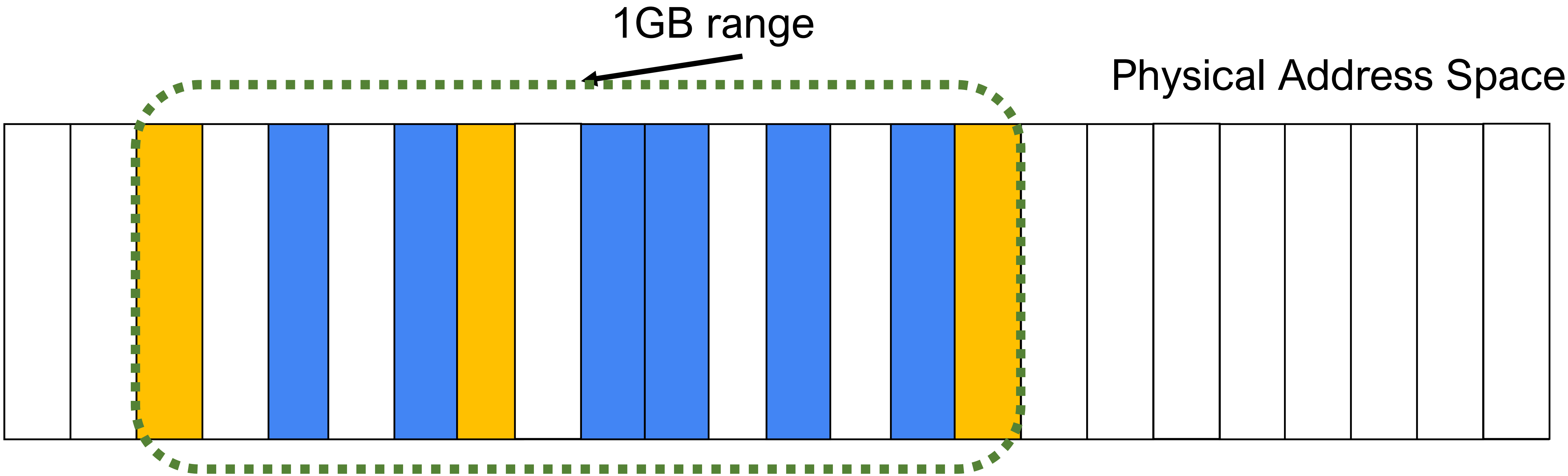


One 4KB unmovable page renders a 2MB or 1GB range unmovable

🟨 Unmovable: Cannot be moved by the OS

🟦 Movable: Can be defragmented

Unmovable Pages Block Contiguity

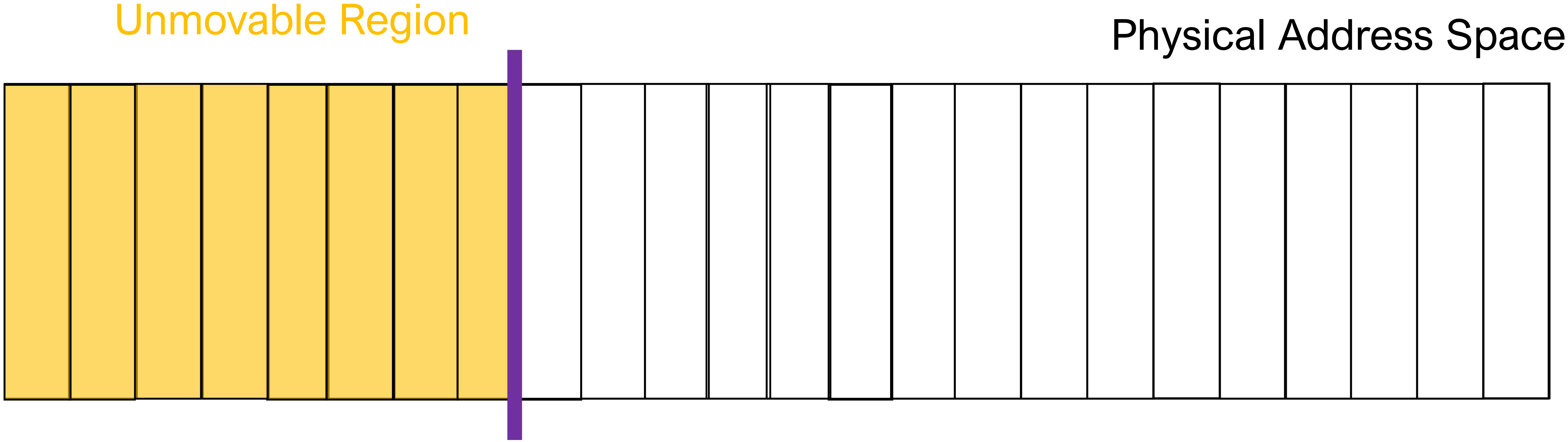


One 4KB unmovable page renders a 2MB or 1GB range unmovable

Only 0.19% of bad allocations can fragment the whole memory irrecoverably

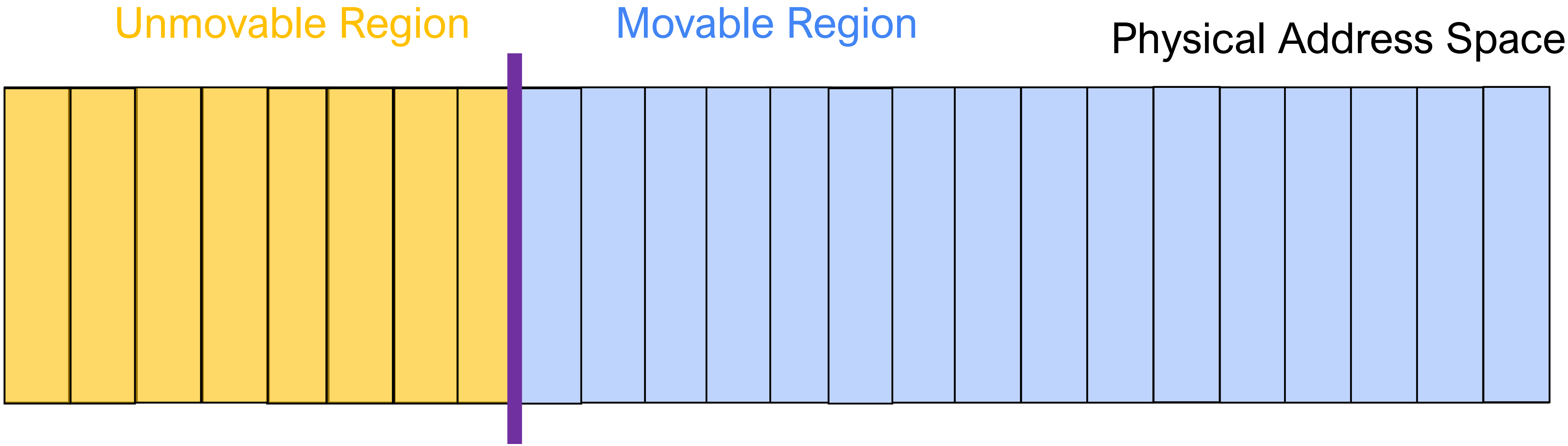
Contiguitas Software:
Ample Physical Contiguity by Design

Separate the Address Space Into Regions



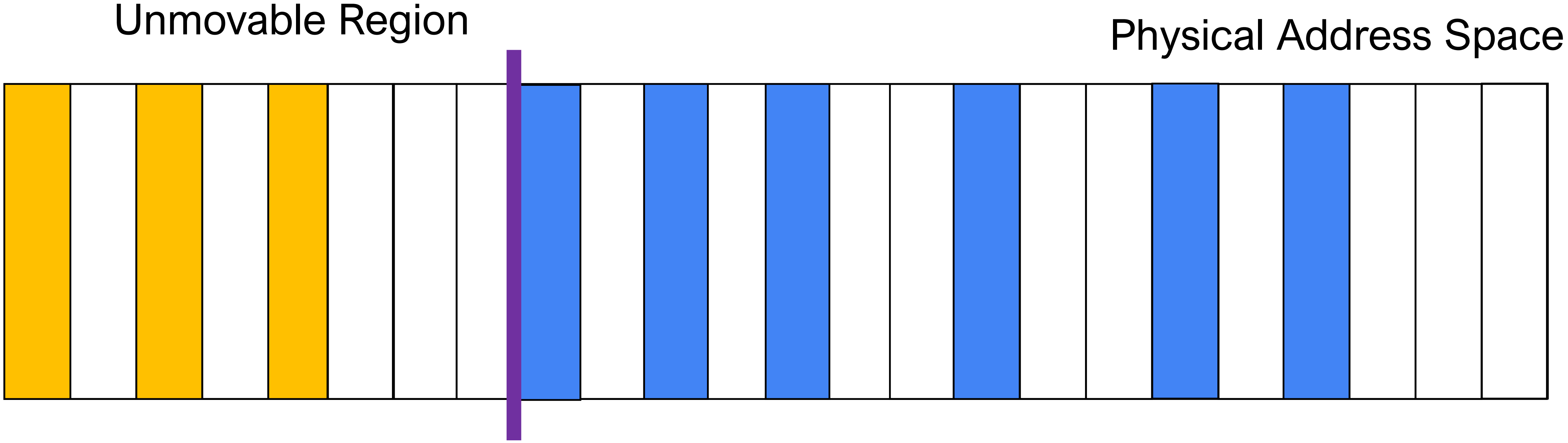
Confines unmovable allocations to the unmovable region

Separate the Address Space Into Regions

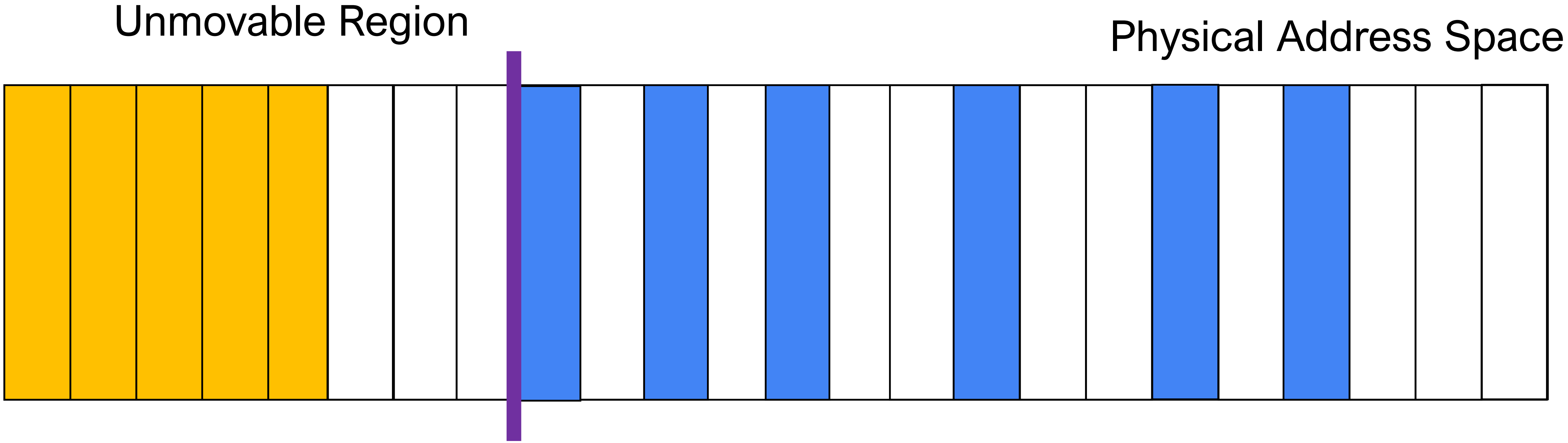


The entire movable region provides contiguity

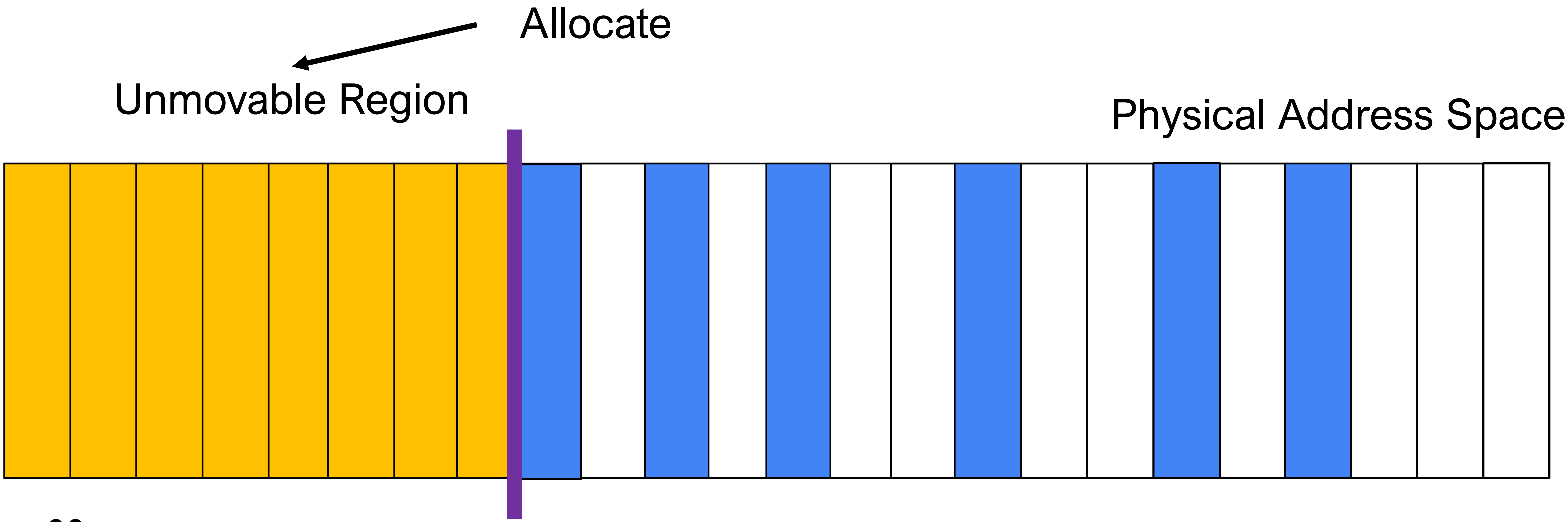
Challenge 1: Resizing Two Regions



Challenge 1: Resizing Two Regions



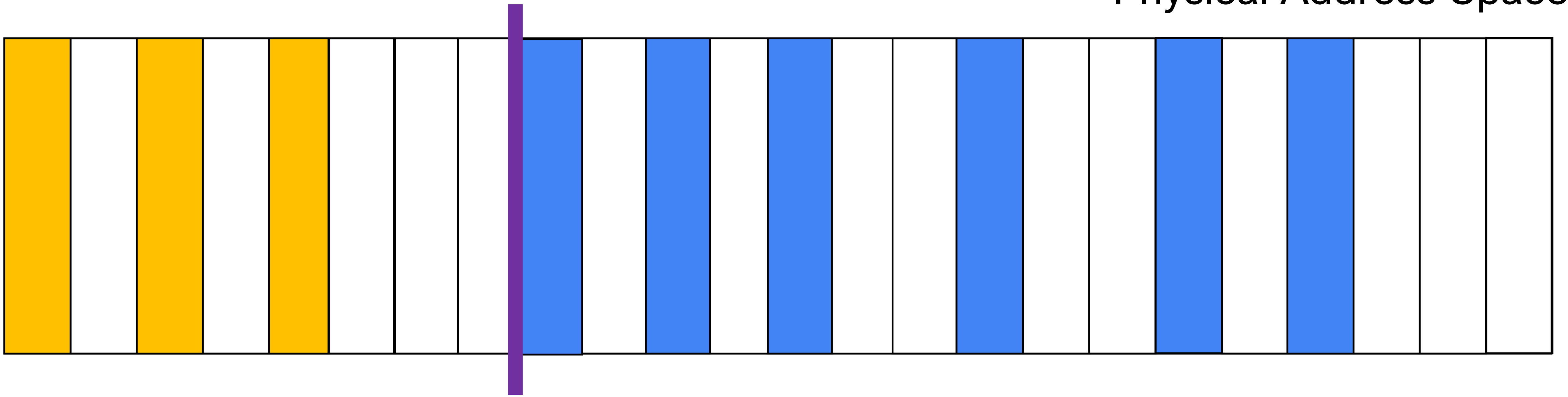
Challenge 1: Resizing Two Regions



🐌 If either region runs out of memory → allocations block

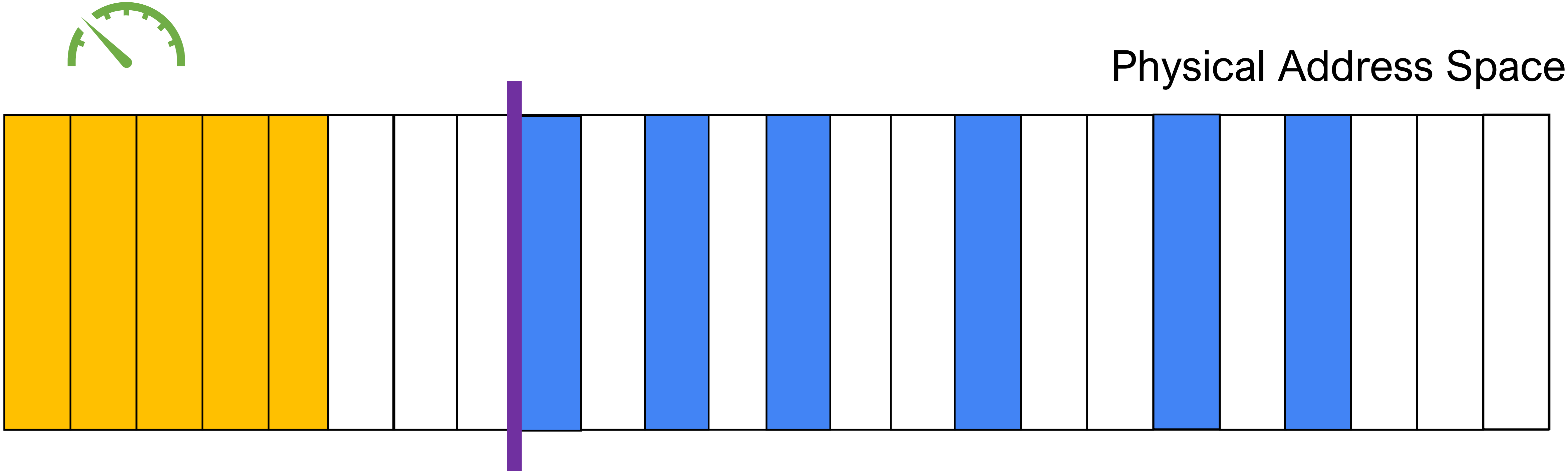
Solution: New Pressure Metric

Pressure: Time spent freeing up memory

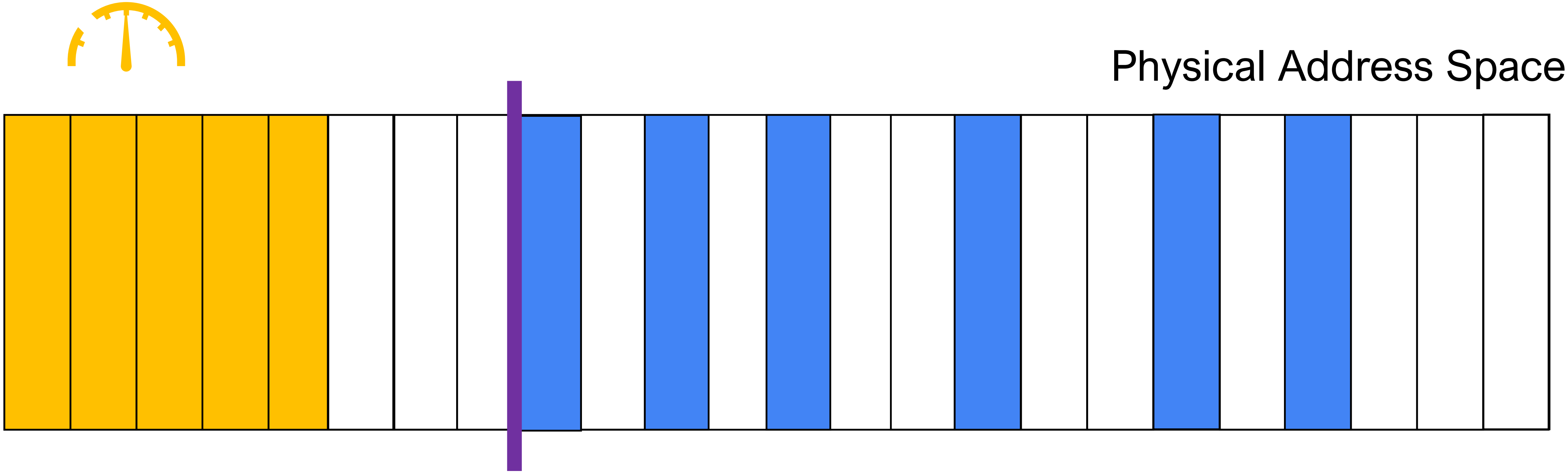


Physical Address Space

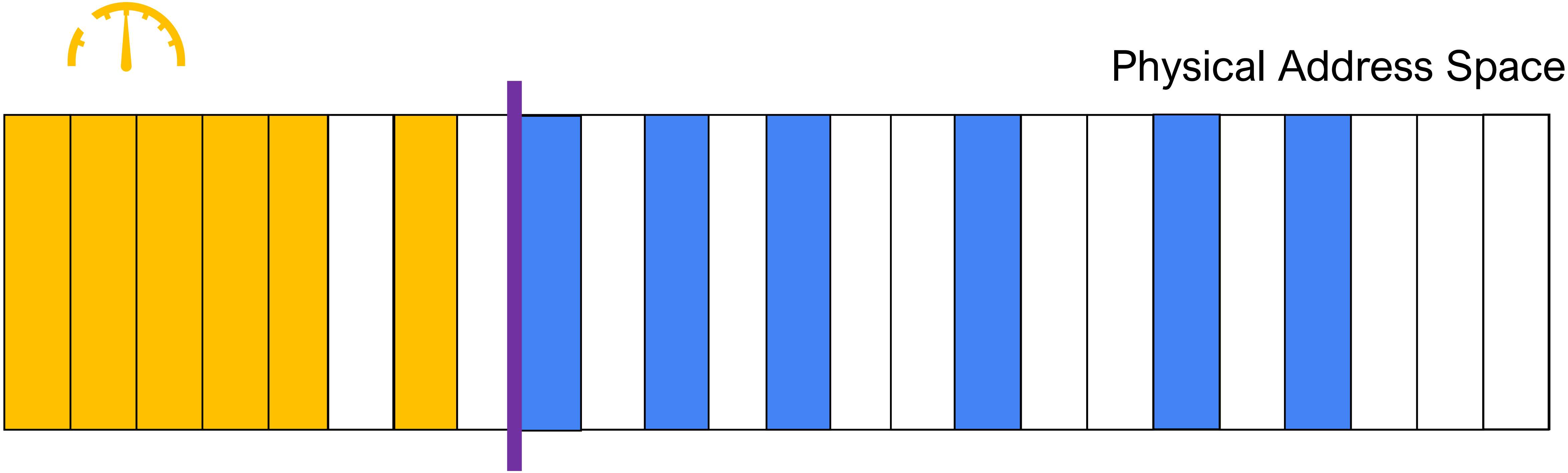
Solution: New Pressure Metric



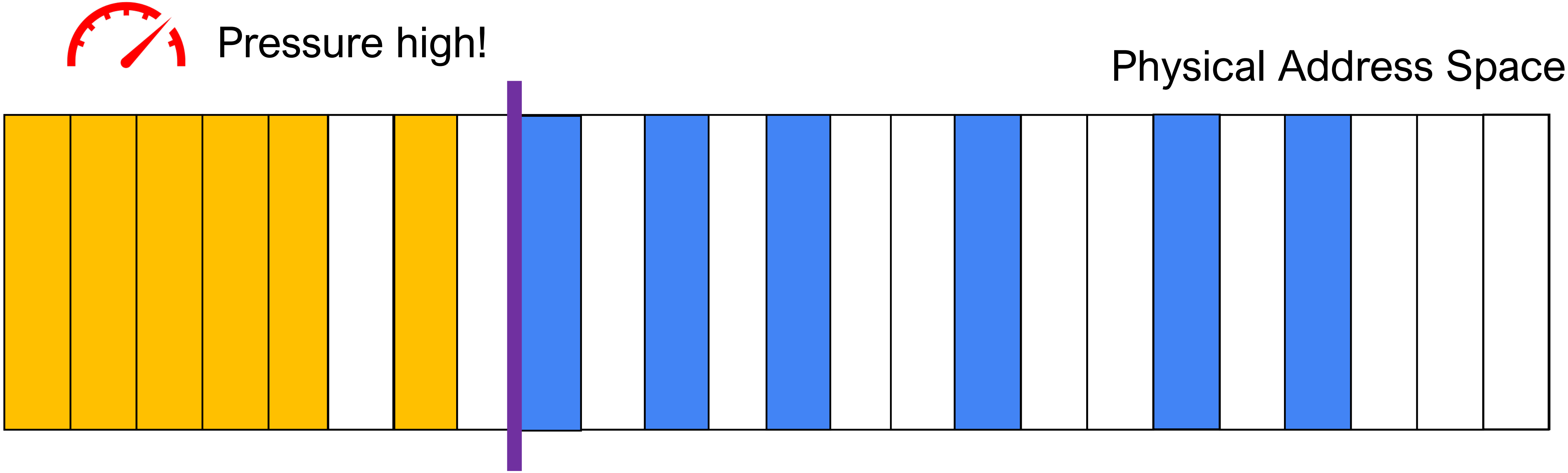
Solution: New Pressure Metric



Solution: New Pressure Metric

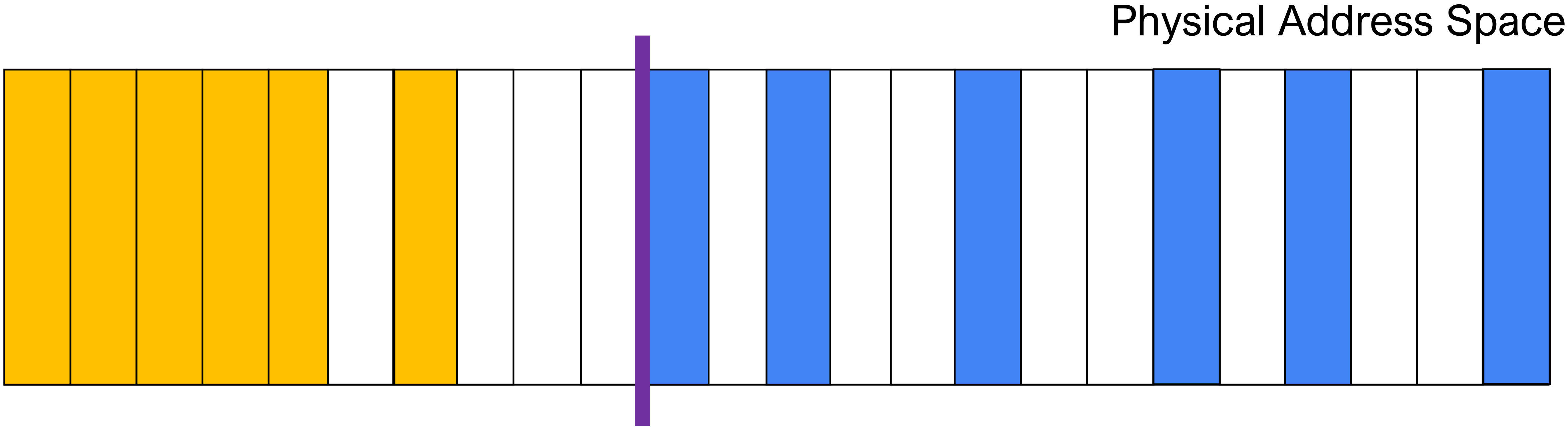


Solution: Resize in the Background



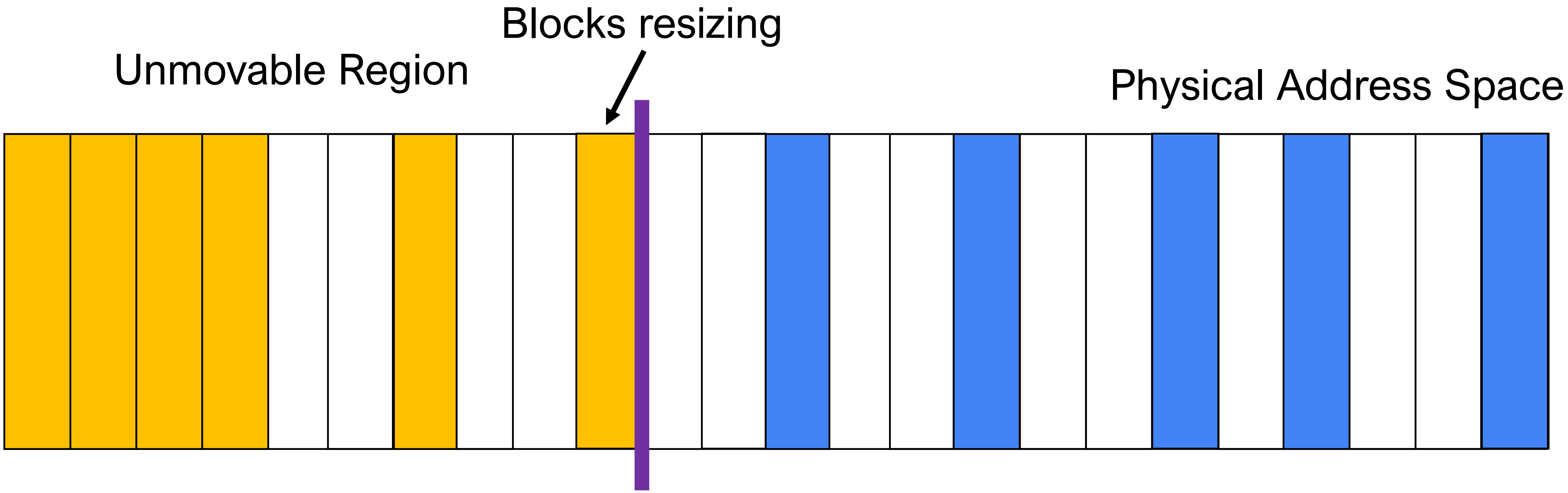
A background thread resizes proactively off the critical path

Solution: Resize in the Background

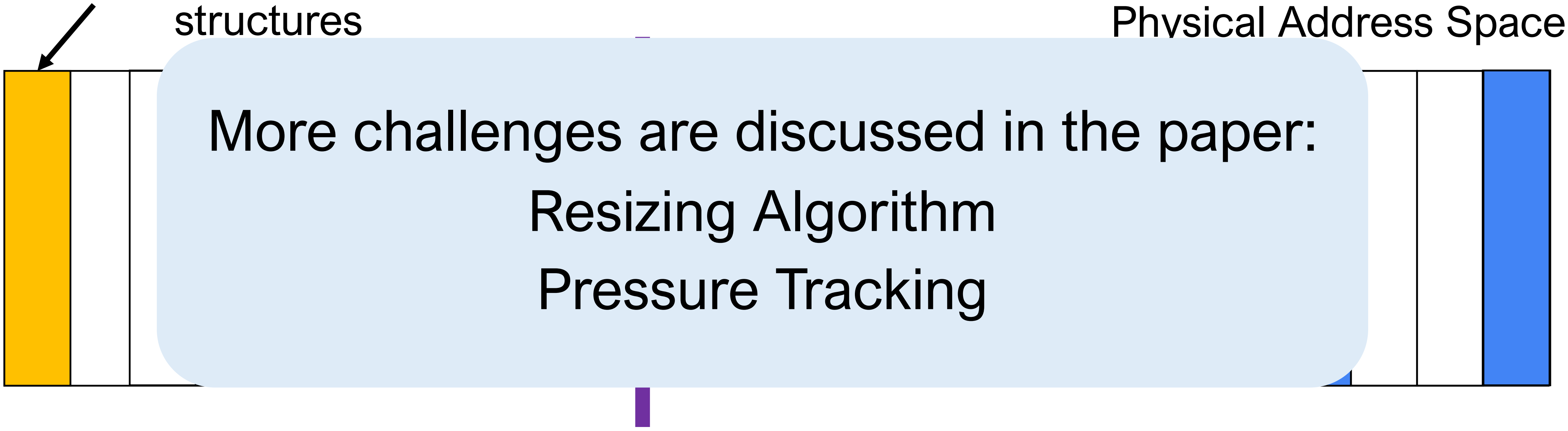


A background thread resizes proactively off the critical path

Challenge 2: Fragmentation in Unmovable



Solution: Allocation Policy



Kernel code,
long-lived kernel data
structures

Physical Address Space

More challenges are discussed in the paper:

Resizing Algorithm

Pressure Tracking

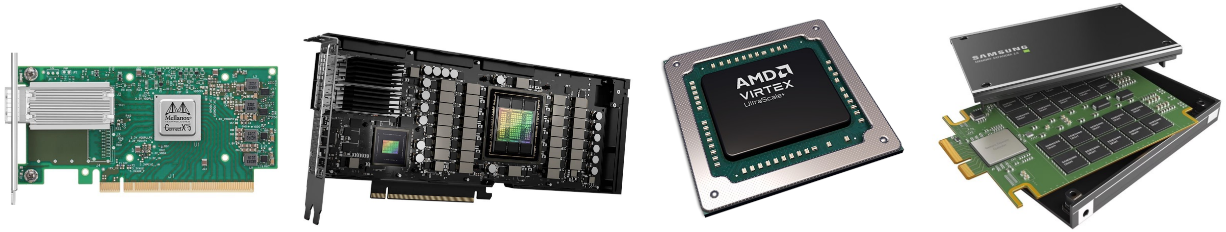
Prefers allocations far from the boundary

The Need to Reduce Unmovable Pages

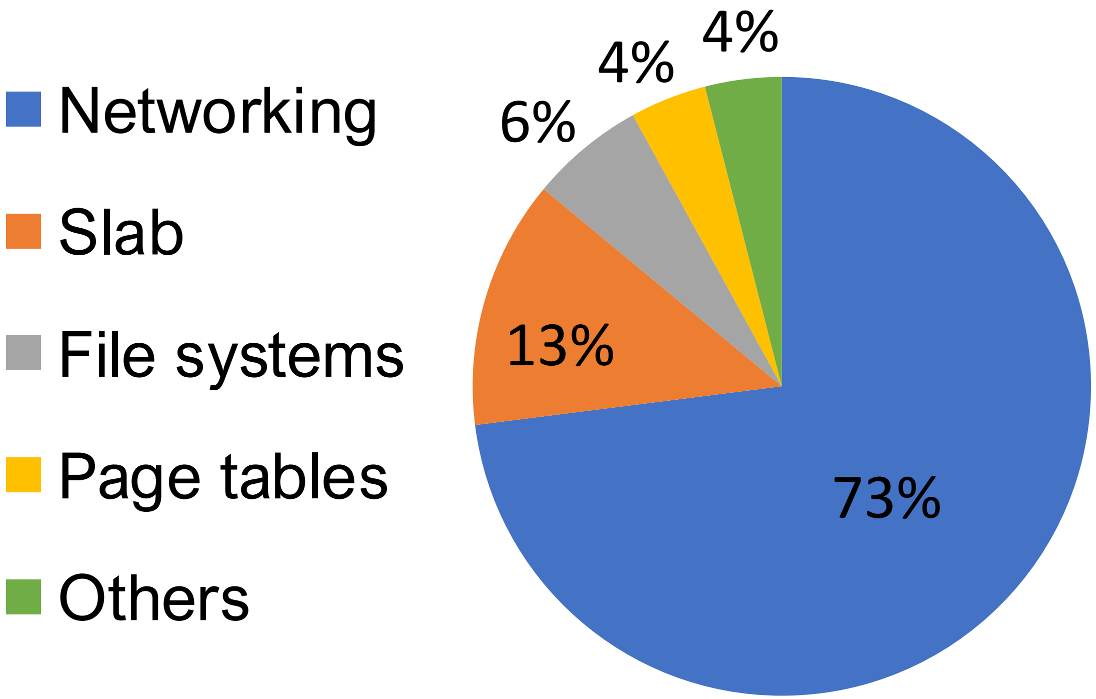
Most unmovable pages come from I/O

More in the future from:

RDMA, GPU, accelerators, CXL



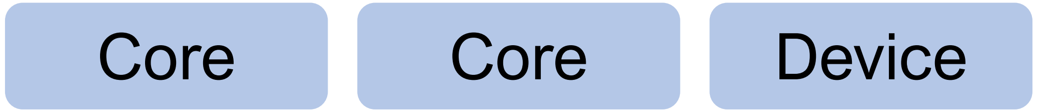
Sources of Unmovable Pages at Meta



Unmovable allocations will get worse!

Why I/O Pages are Unmovable

Why I/O Pages are Unmovable

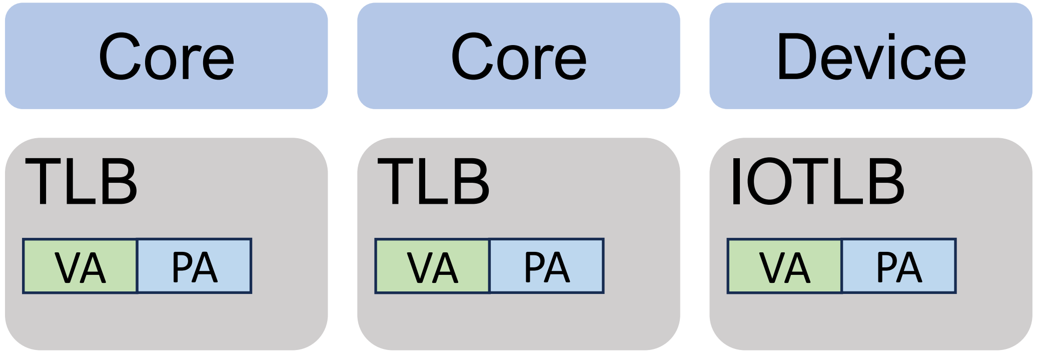


Core

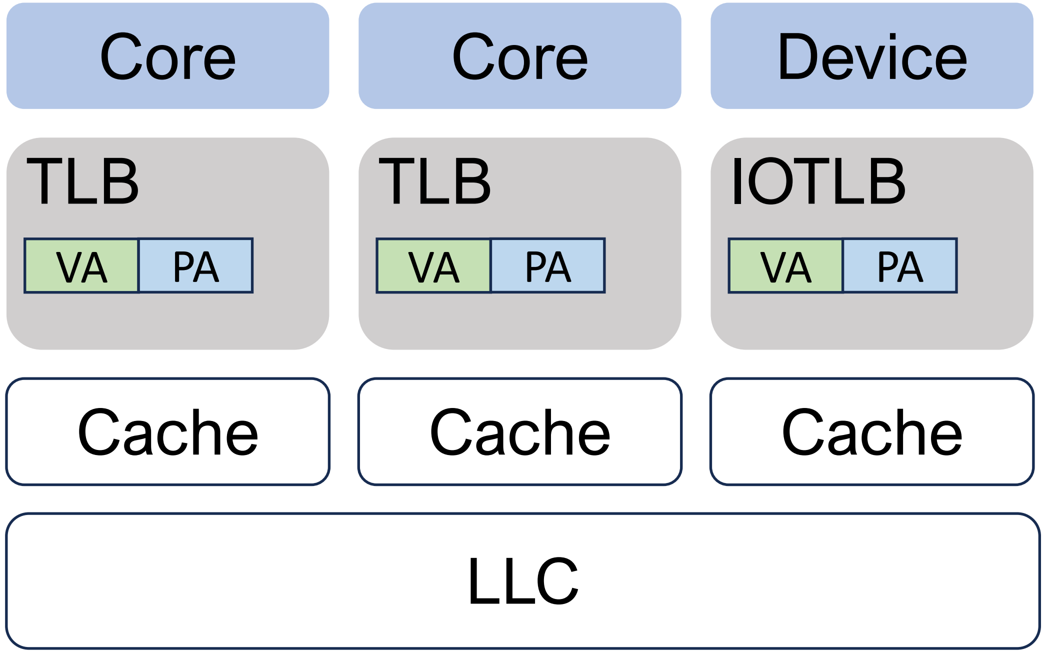
Core

Device

Why I/O Pages are Unmovable



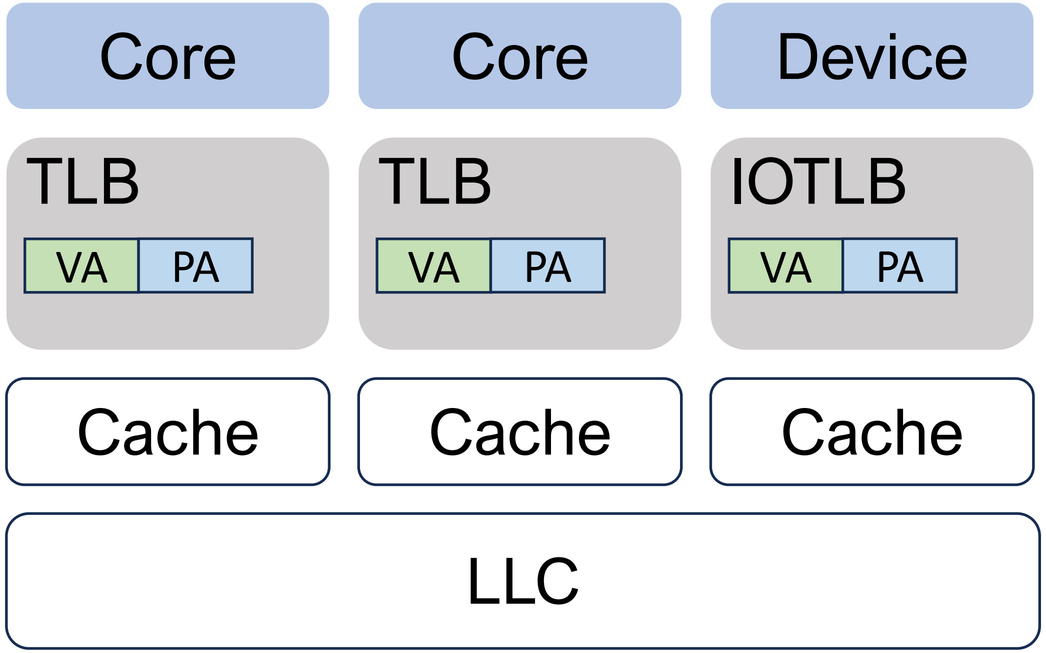
Why I/O Pages are Unmovable



Why I/O Pages are Unmovable

Software Page Migration

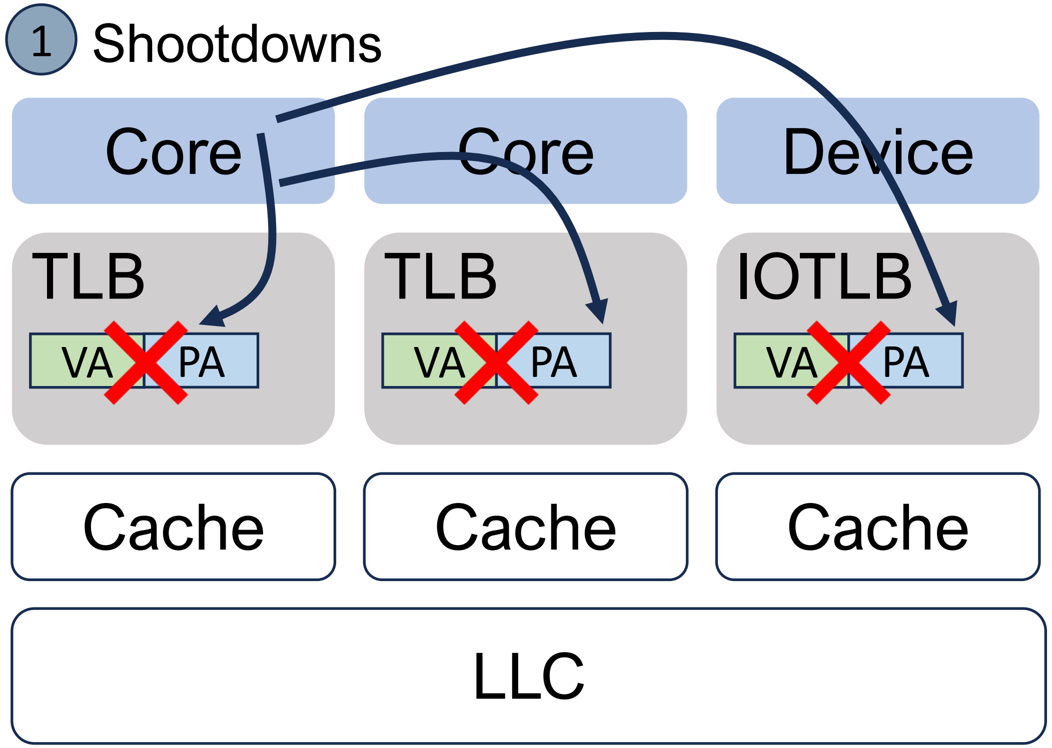
- Access must be blocked during migration



Why I/O Pages are Unmovable

Software Page Migration

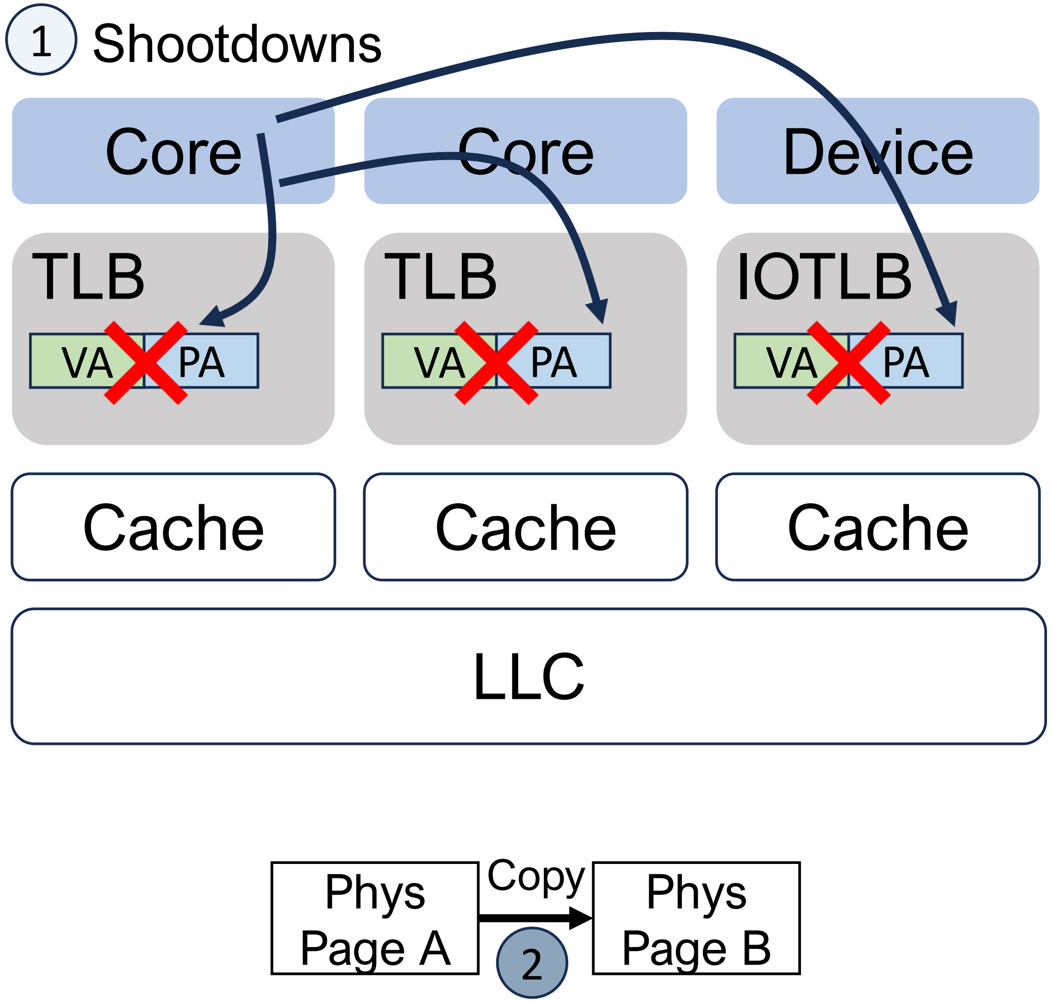
- Access must be blocked during migration



Why I/O Pages are Unmovable

Software Page Migration

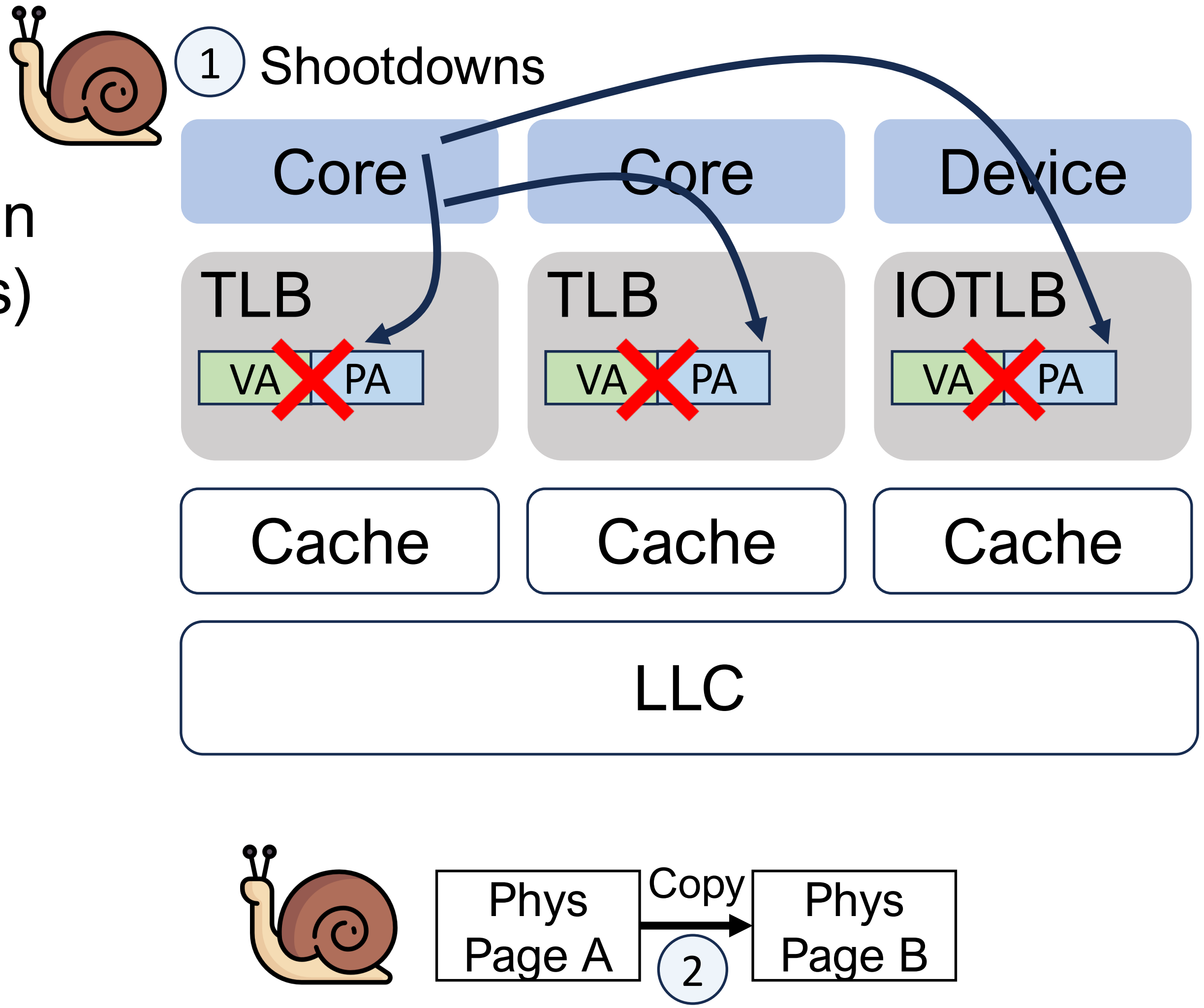
- Access must be blocked during migration



Why I/O Pages are Unmovable

Software Page Migration

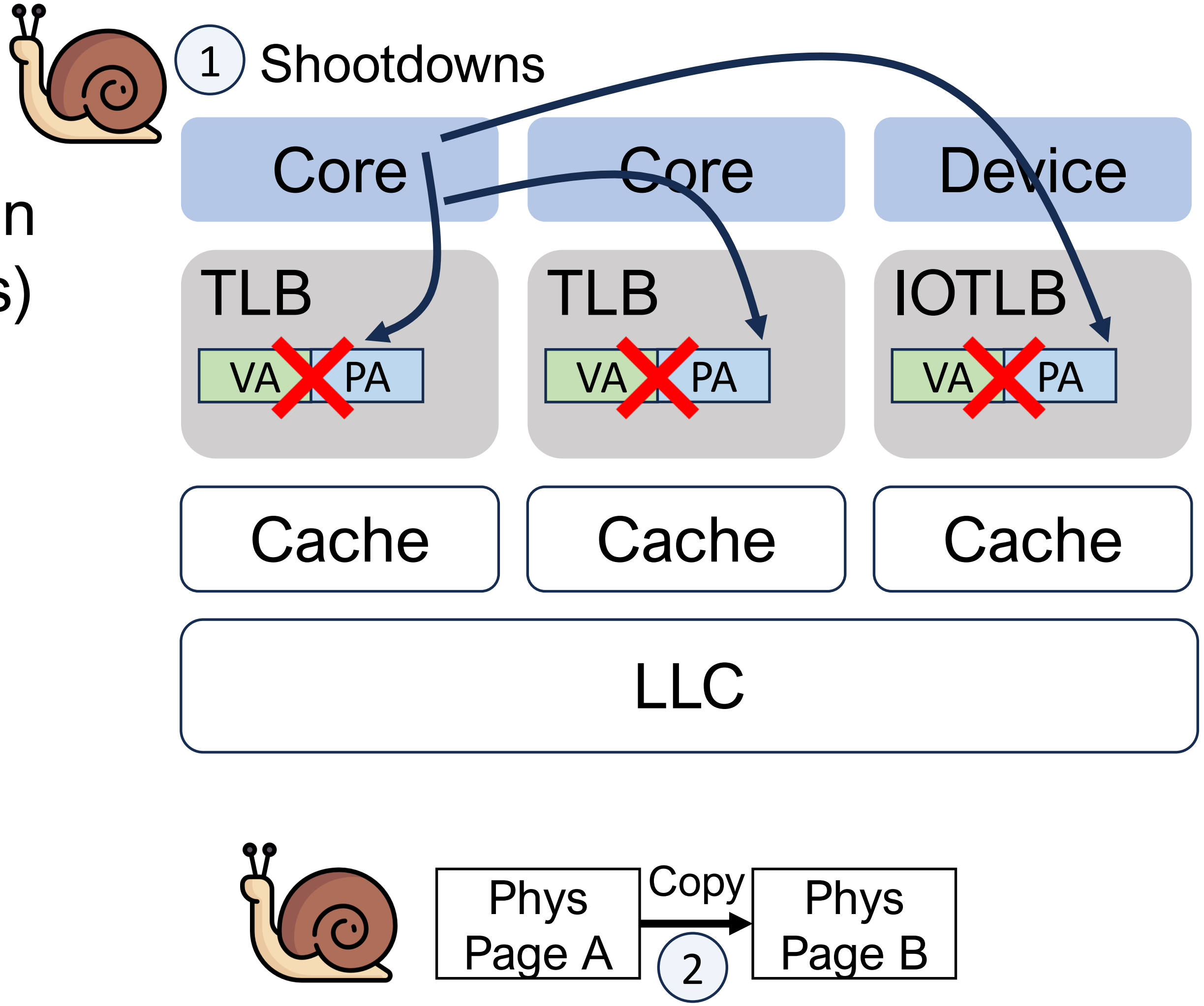
- Access must be blocked during migration
- Impossible for many devices (page faults)



Why I/O Pages are Unmovable

Software Page Migration

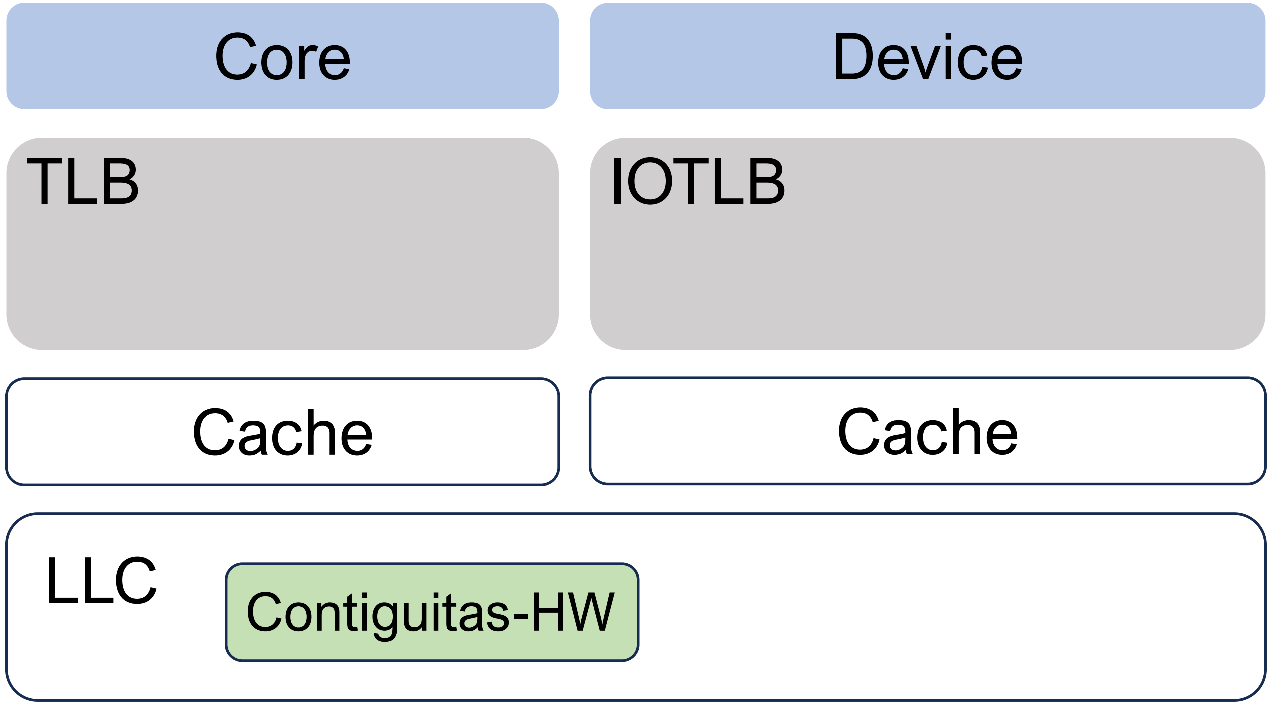
- Access must be blocked during migration
- Impossible for many devices (page faults)
- Long page unavailable time



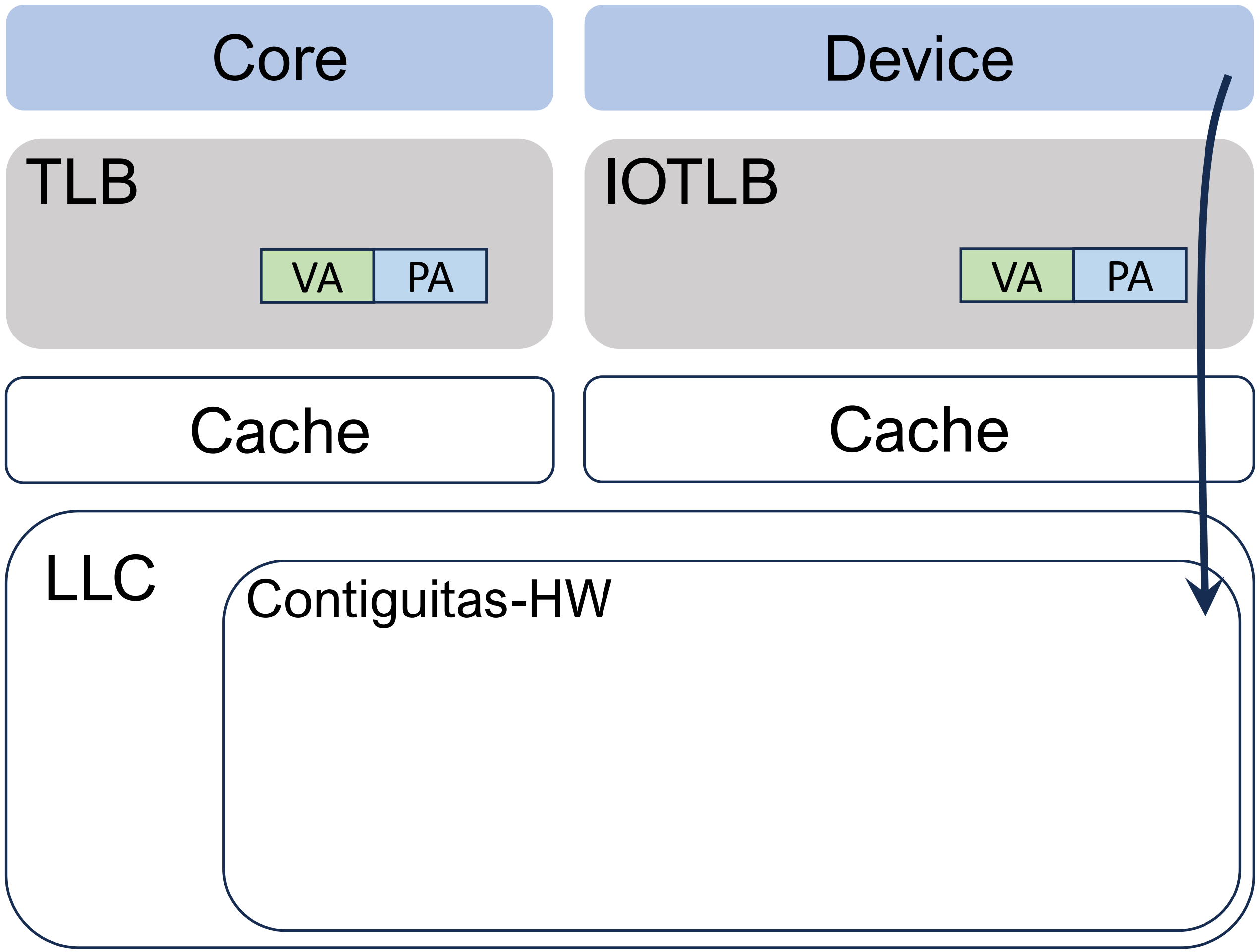
Contiguitas Hardware: Enabling Migration of I/O Pages

Contiguitas Hardware

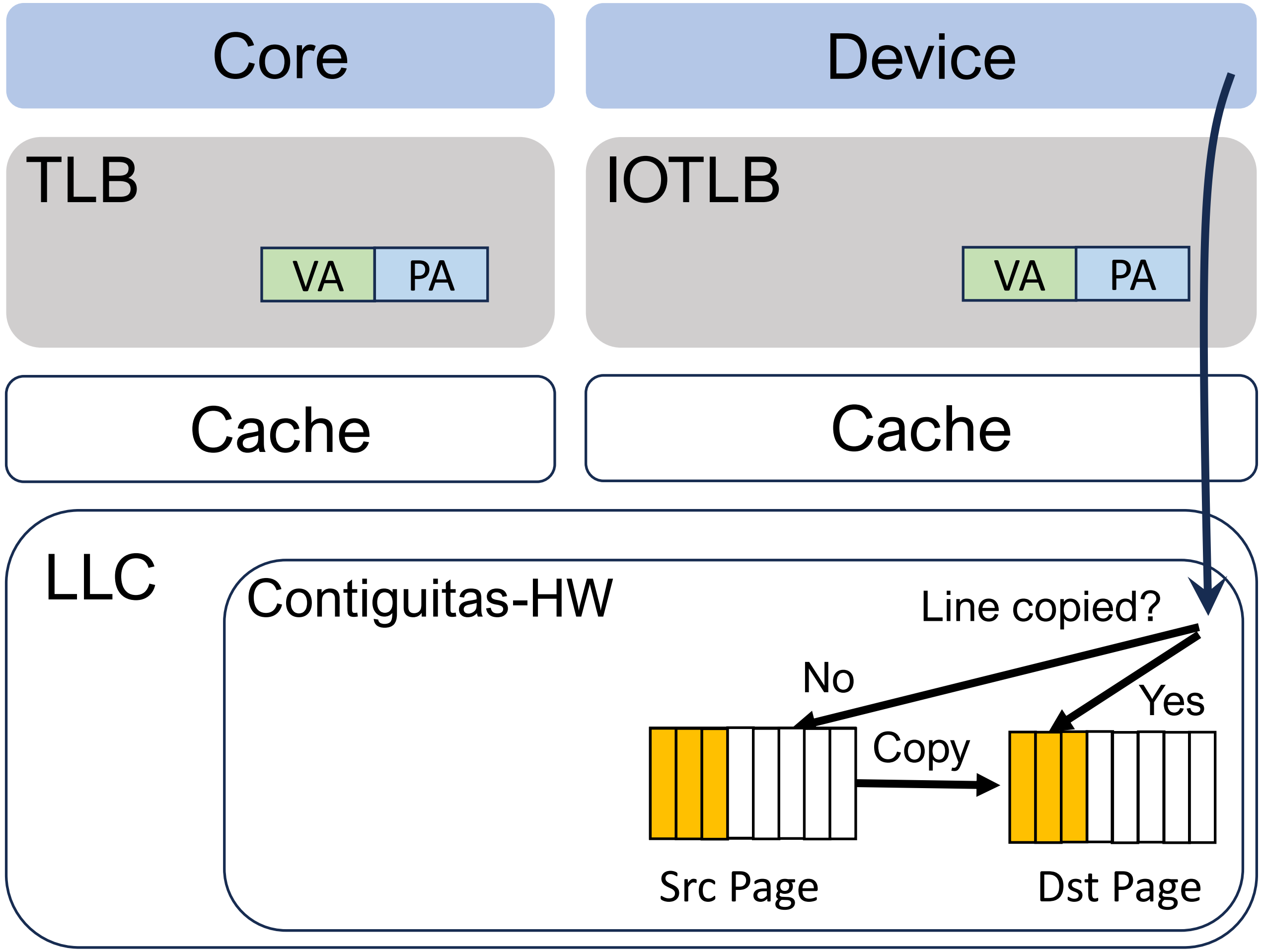
Contiguitas-HW in LLC provides transparent page migration



Transparent Page Migration

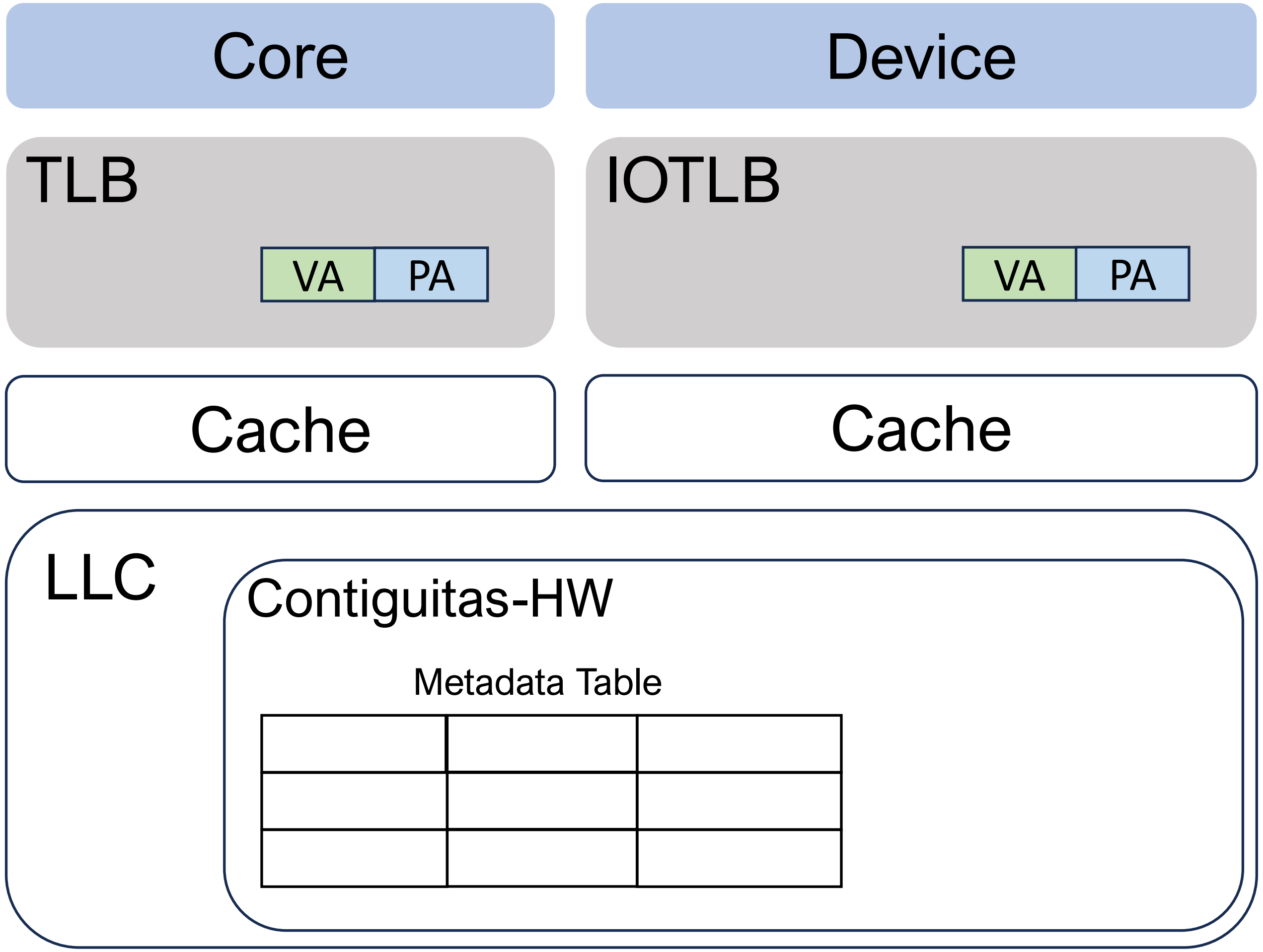


Transparent Page Migration

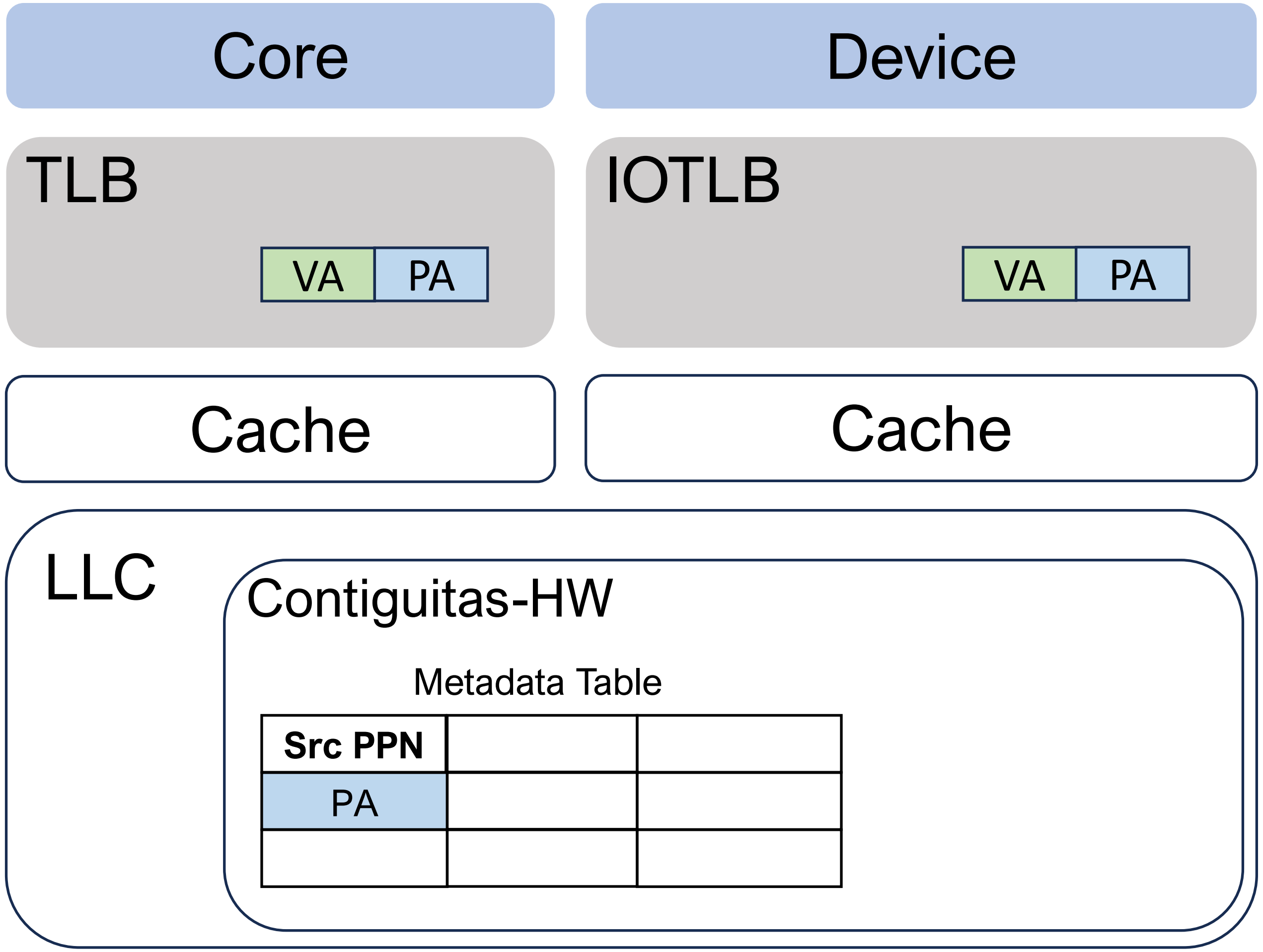


🟨 Migrated Line

Transparent Page Migration



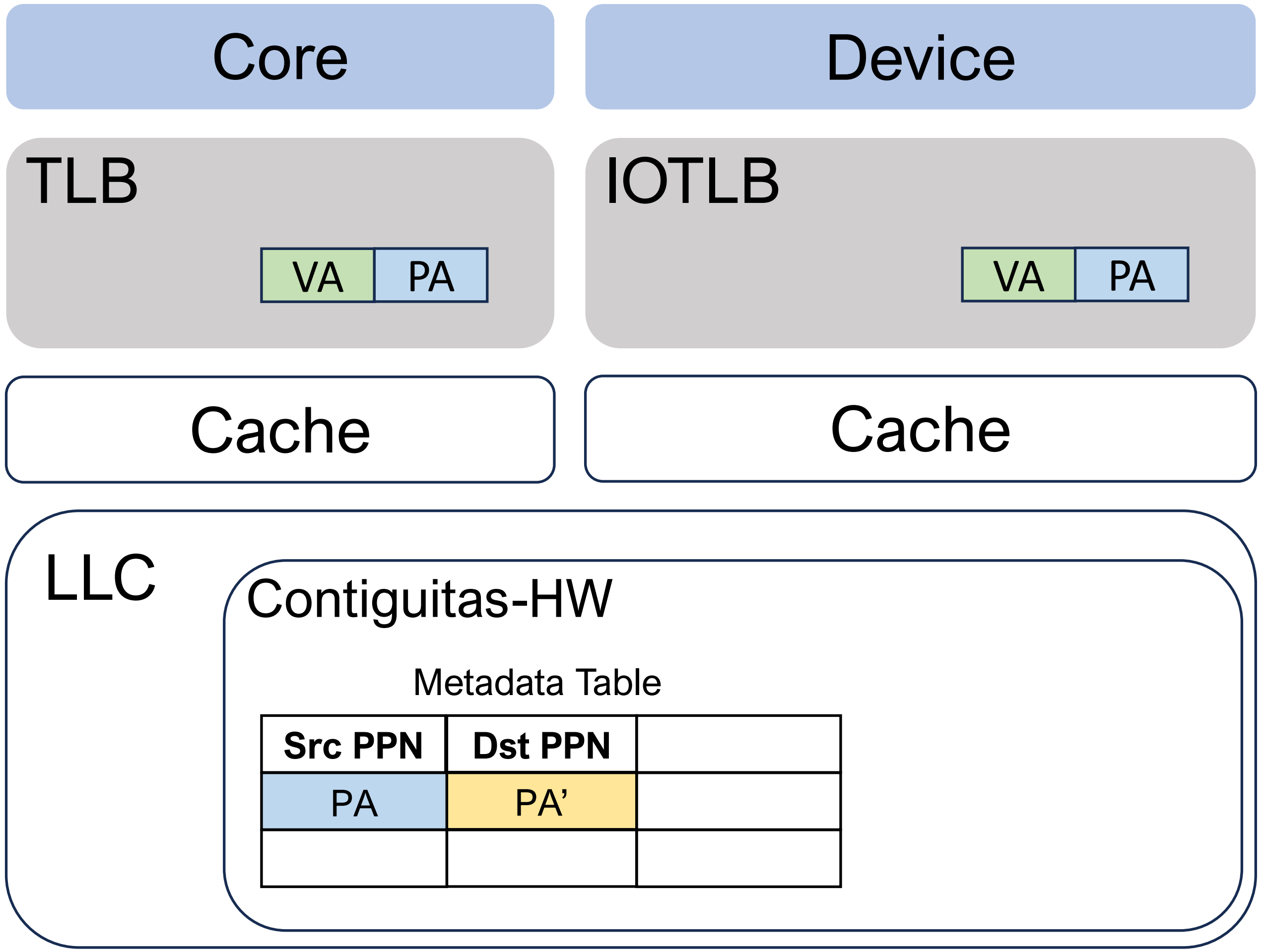
Transparent Page Migration



Metadata Table

Src PPN		
PA		

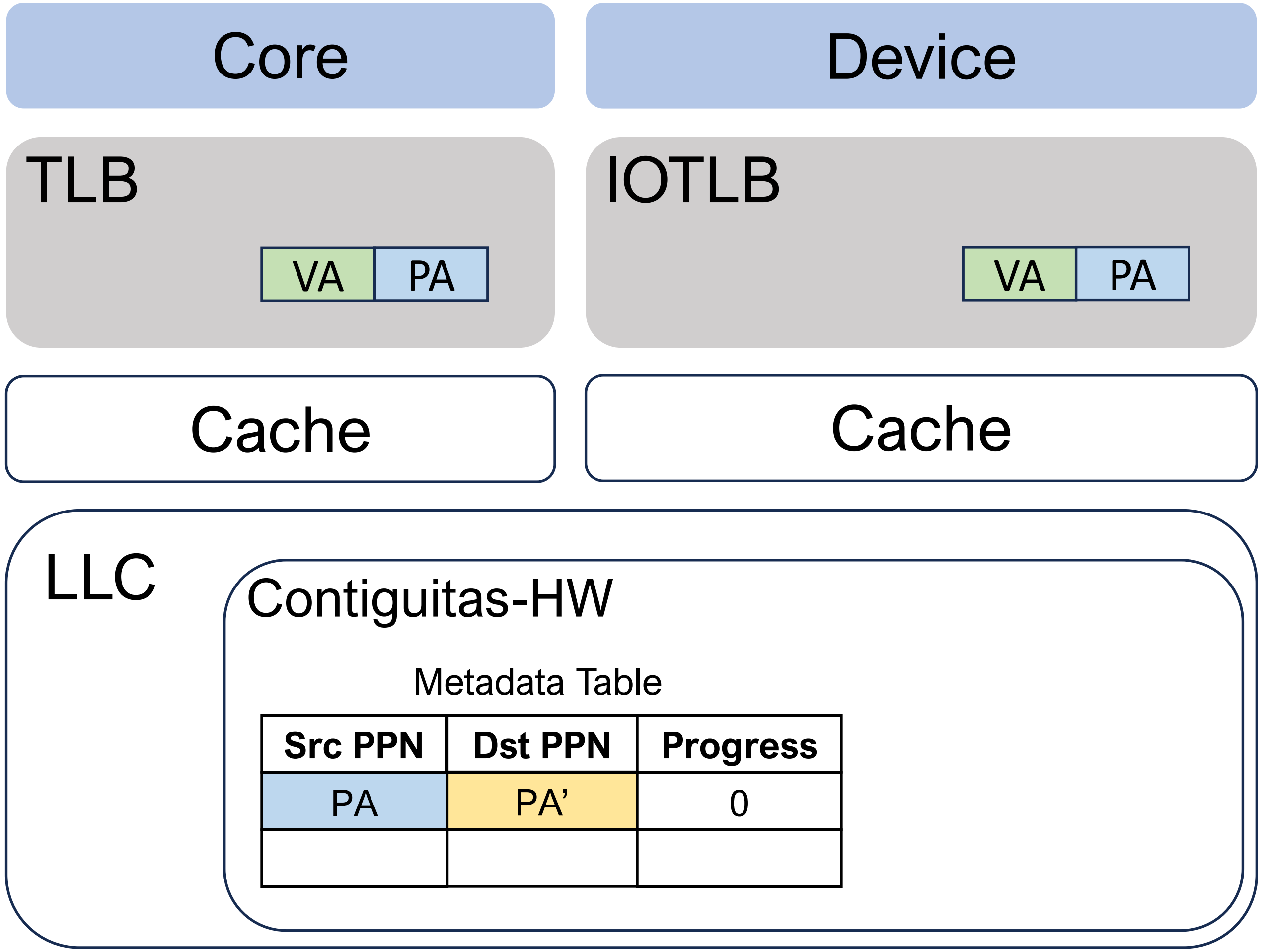
Transparent Page Migration



Metadata Table

Src PPN	Dst PPN	
PA	PA’	

Transparent Page Migration

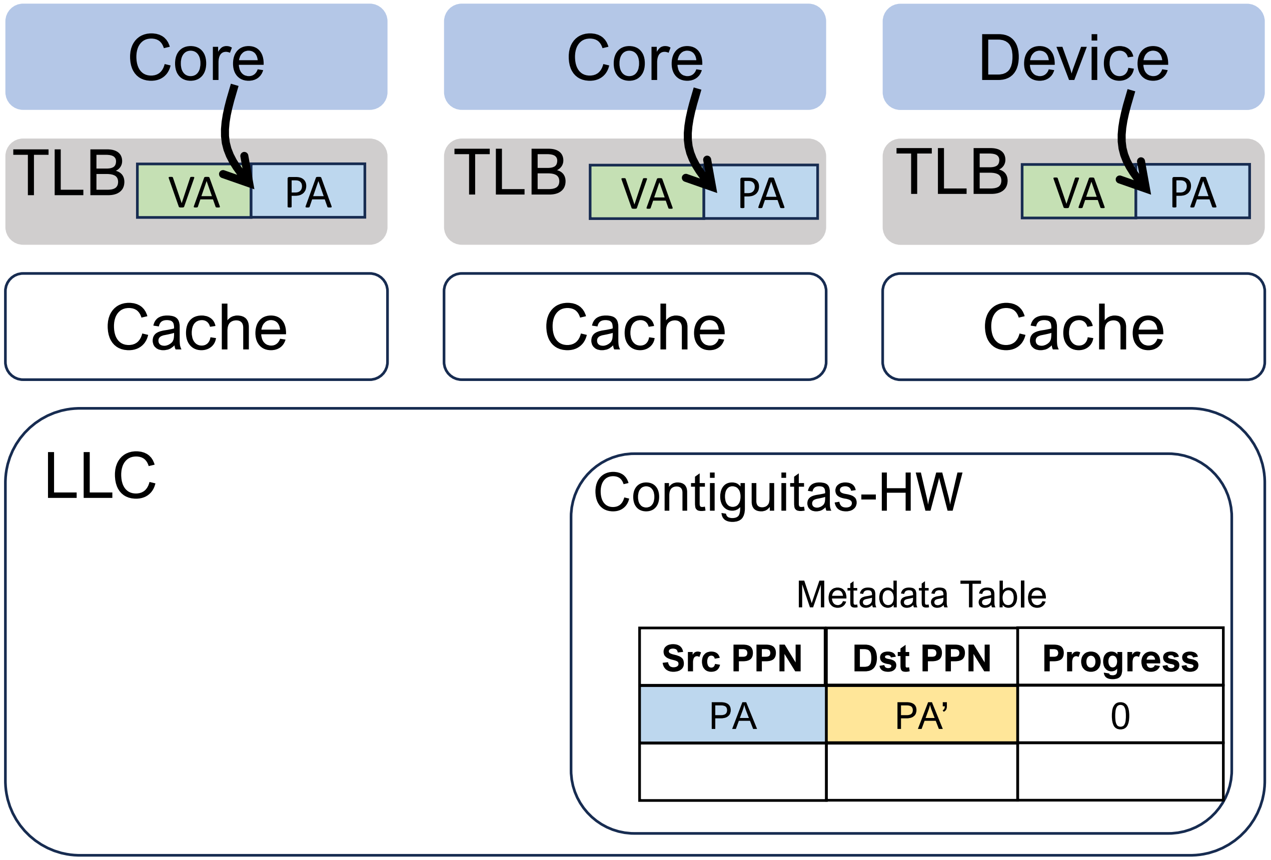


Metadata Table

Src PPN	Dst PPN	Progress
PA	PA’	0

Scalable Local TLB Invalidations

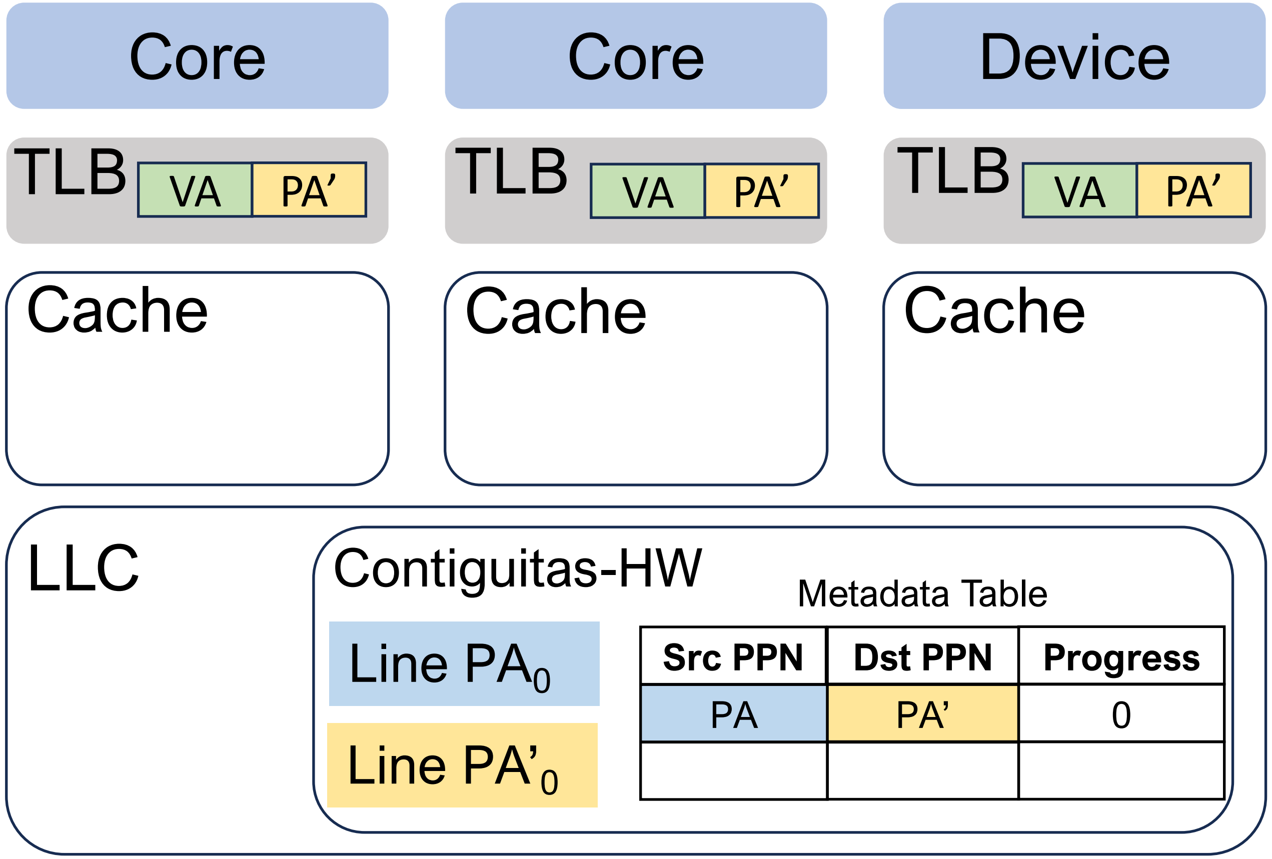
1 Switch translation
Local Flush TLB



Metadata Table

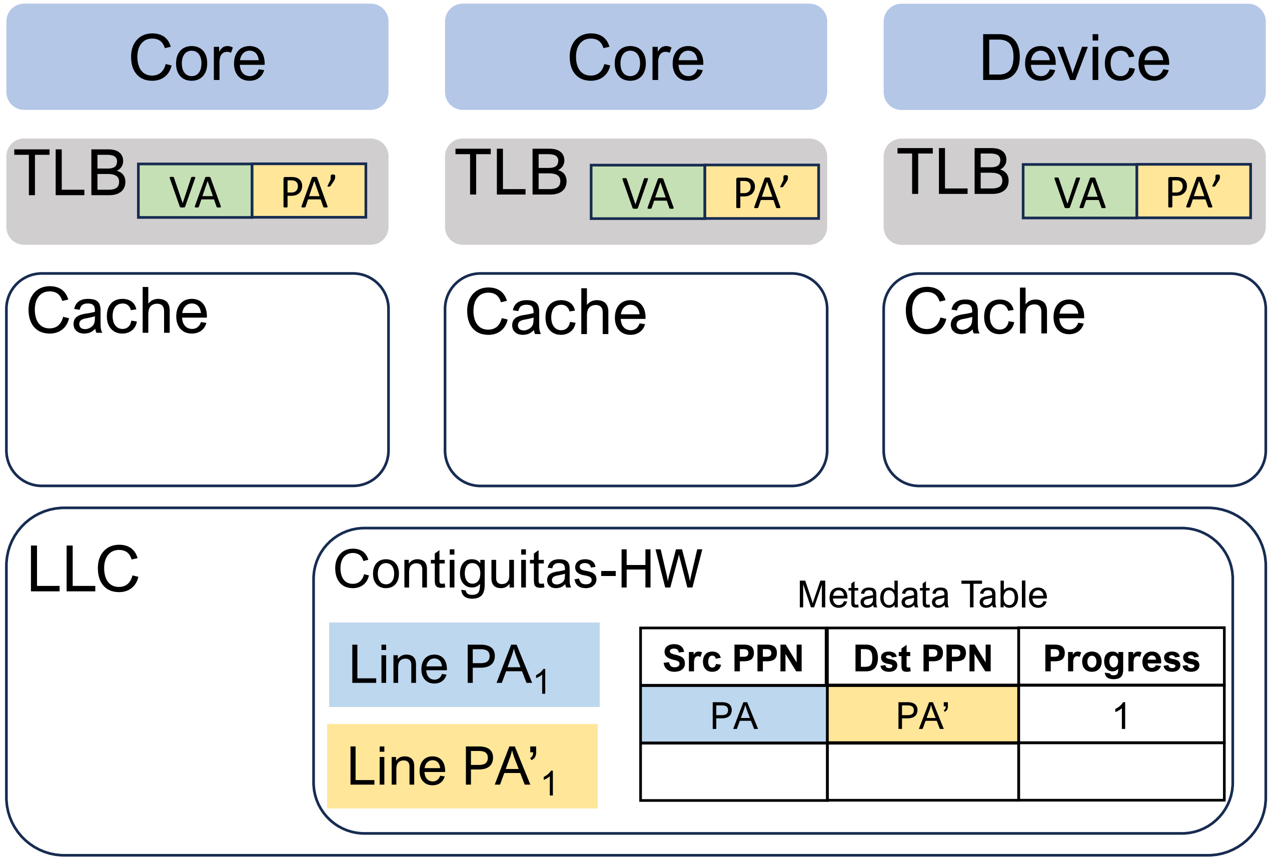
Src PPN	Dst PPN	Progress
PA	PA’	0

Moving One Cacheline



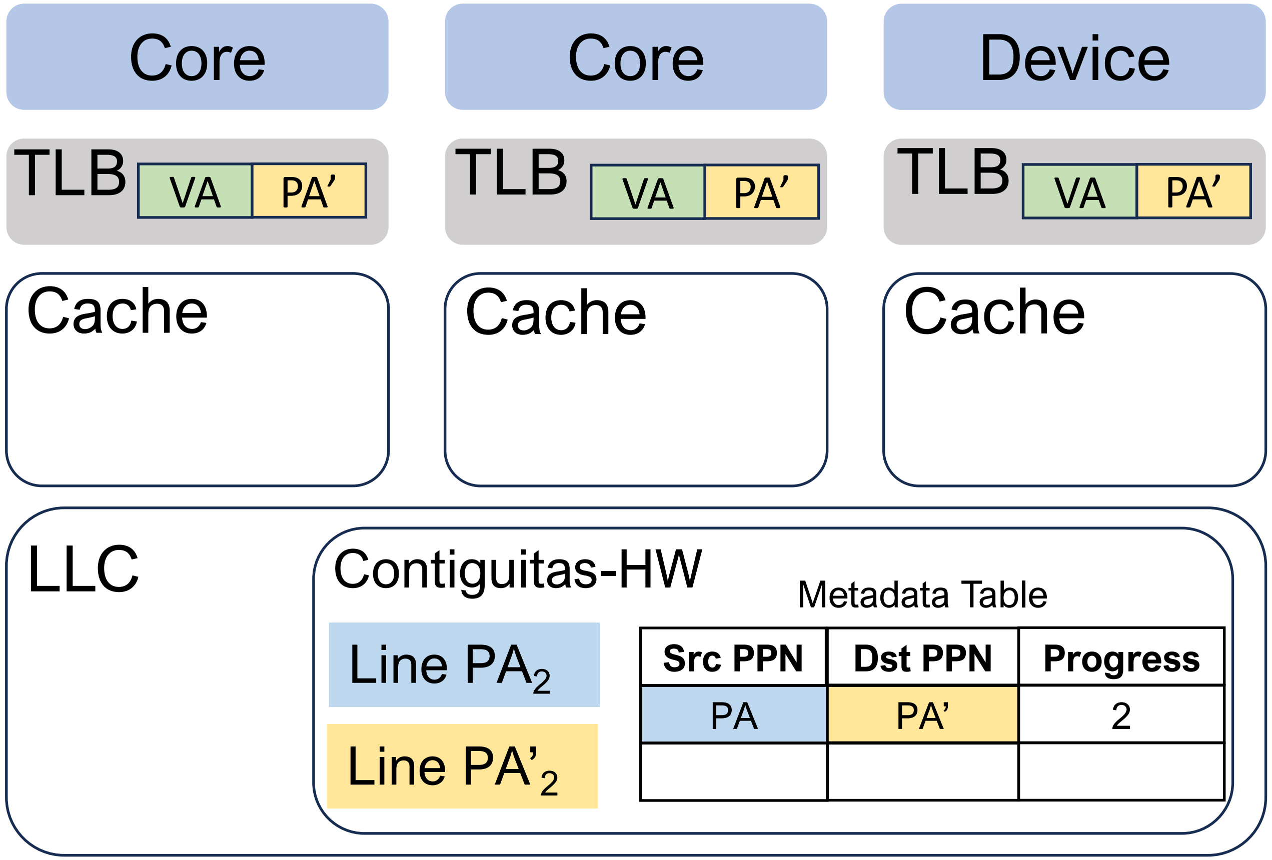
Src PPN	Dst PPN	Progress
PA	PA’	0

Moving One Cacheline



Src PPN	Dst PPN	Progress
PA	PA’	1

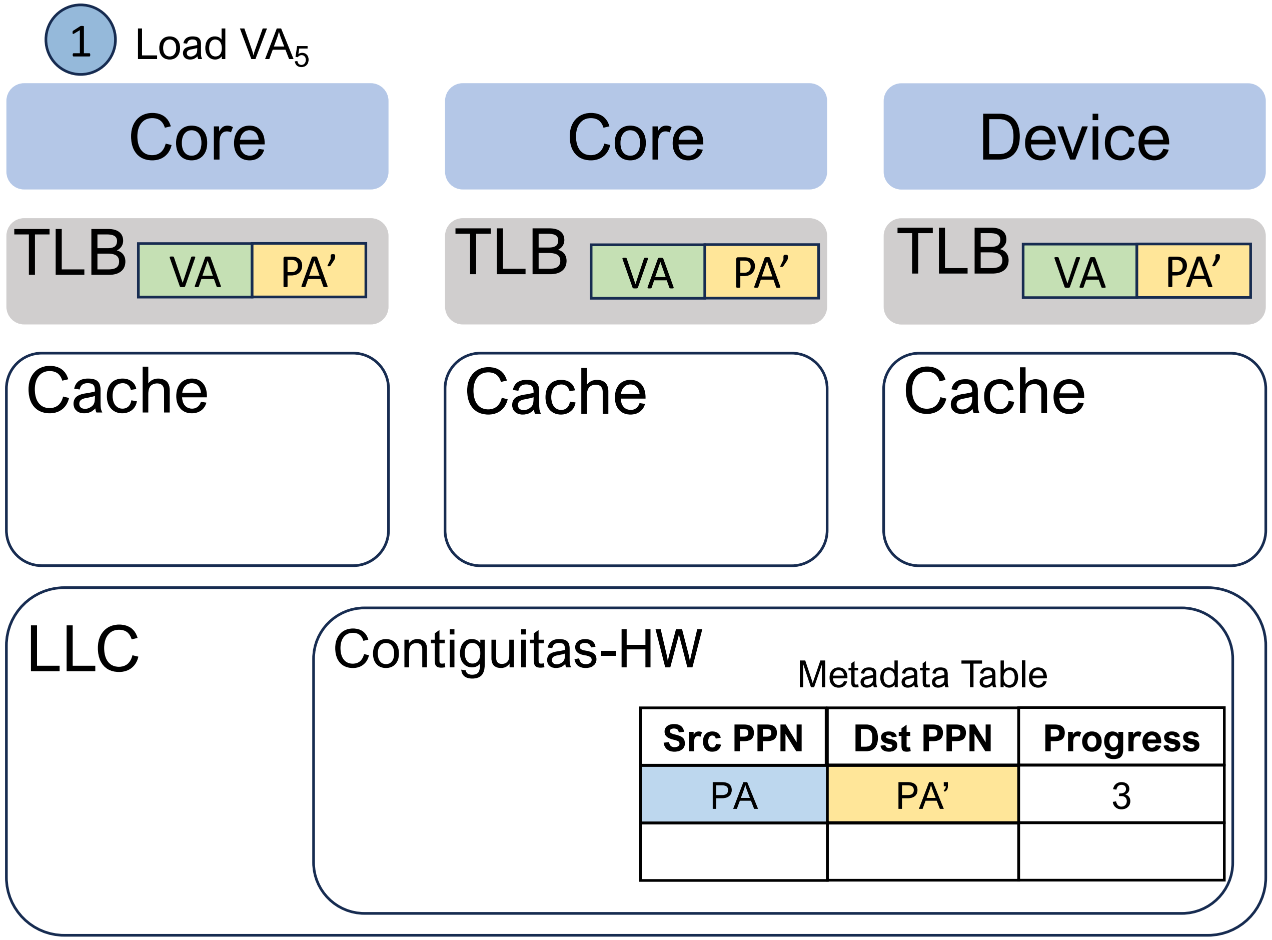
Moving One Cacheline



Metadata Table

Src PPN	Dst PPN	Progress
PA	PA’	2

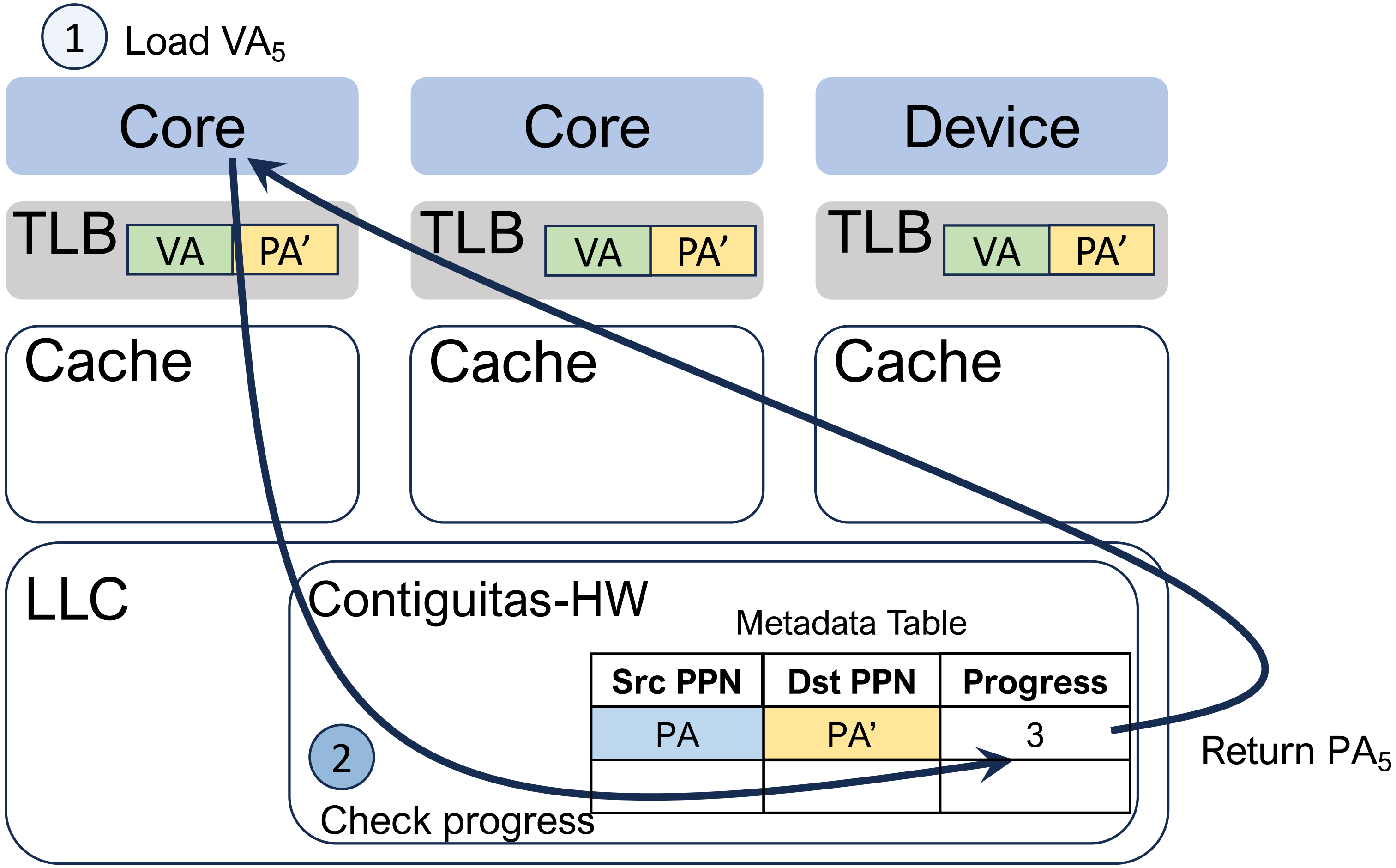
Redirecting Access to Moved Lines



Metadata Table

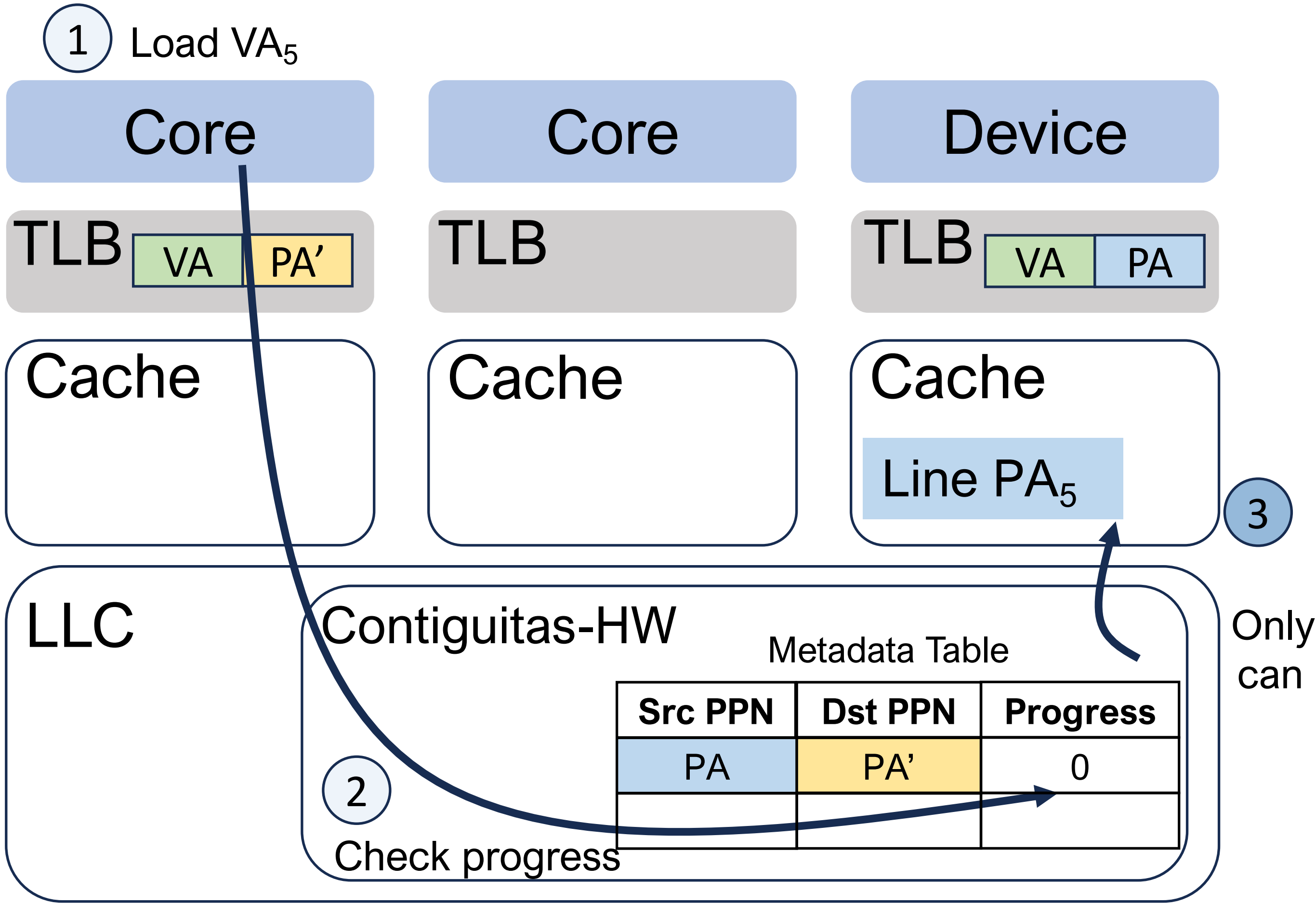
Src PPN	**Dst PPN**	**Progress**
PA	PA’	3

Redirecting Access to Moved Lines



Src PPN	Dst PPN	Progress
PA	PA’	3

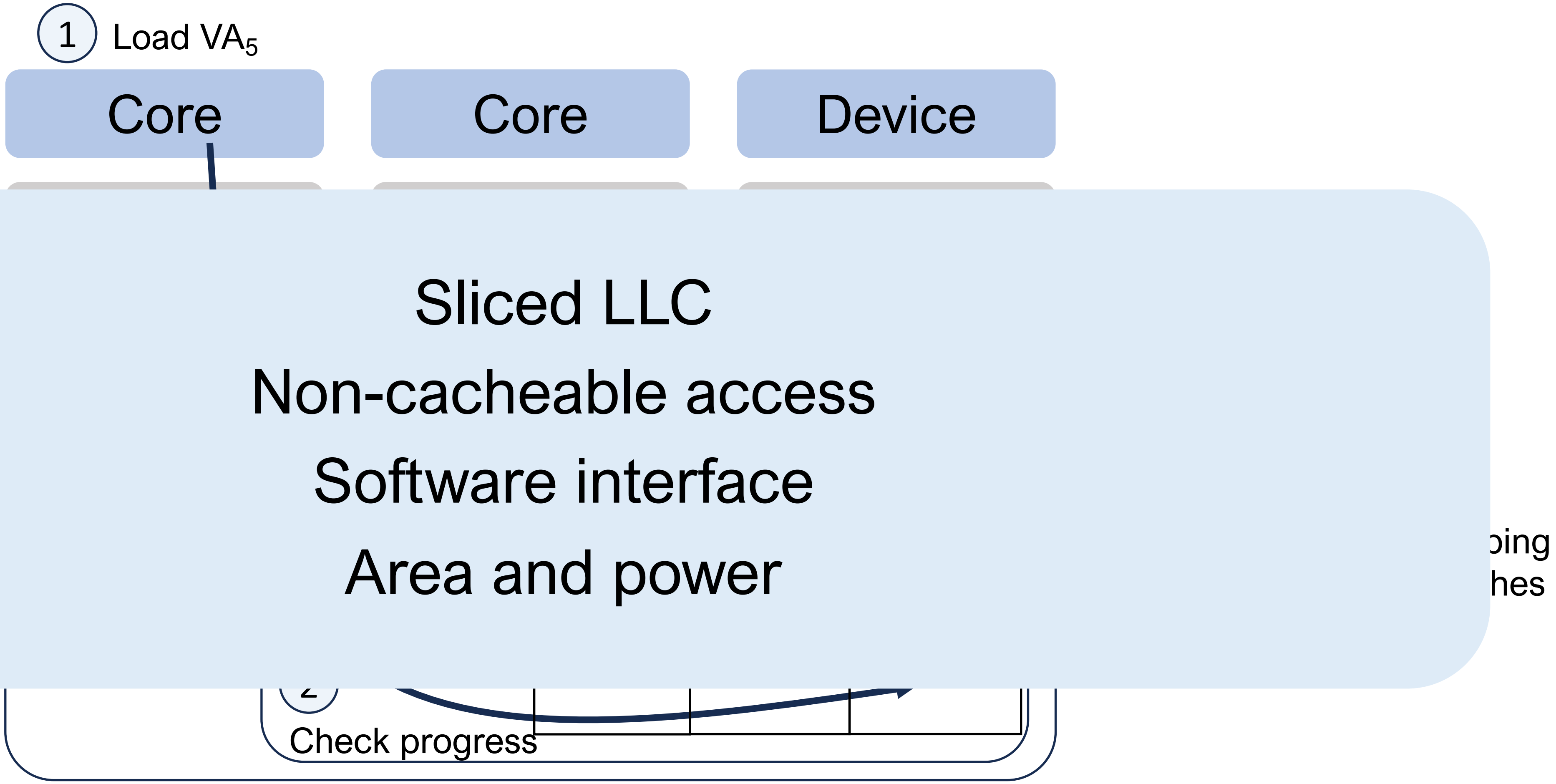
Challenge: Concurrent Active Physical Mappings



Src PPN	Dst PPN	Progress
PA	PA’	0

Invalidate:
Only one of src or dst mapping can exist in the private caches

More in the Paper



Sliced LLC

Non-cacheable access

Software interface

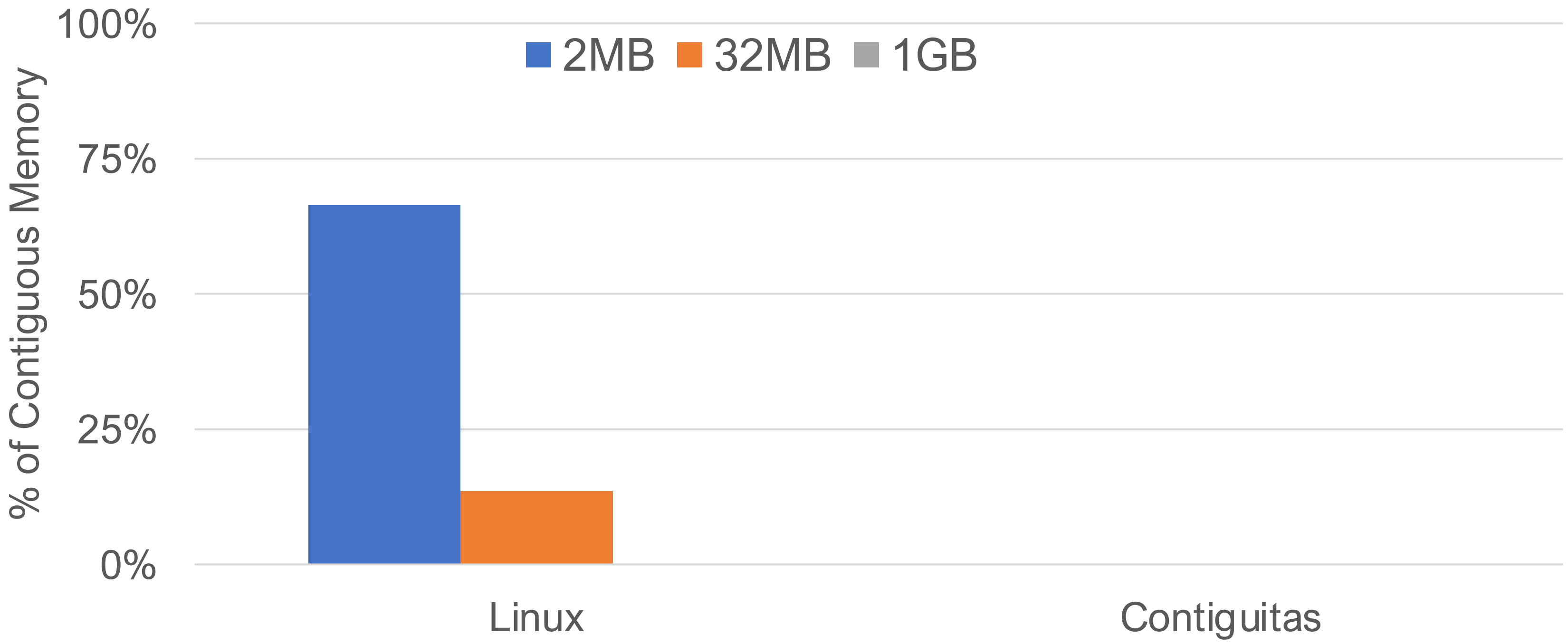
Area and power

Evaluation Overview

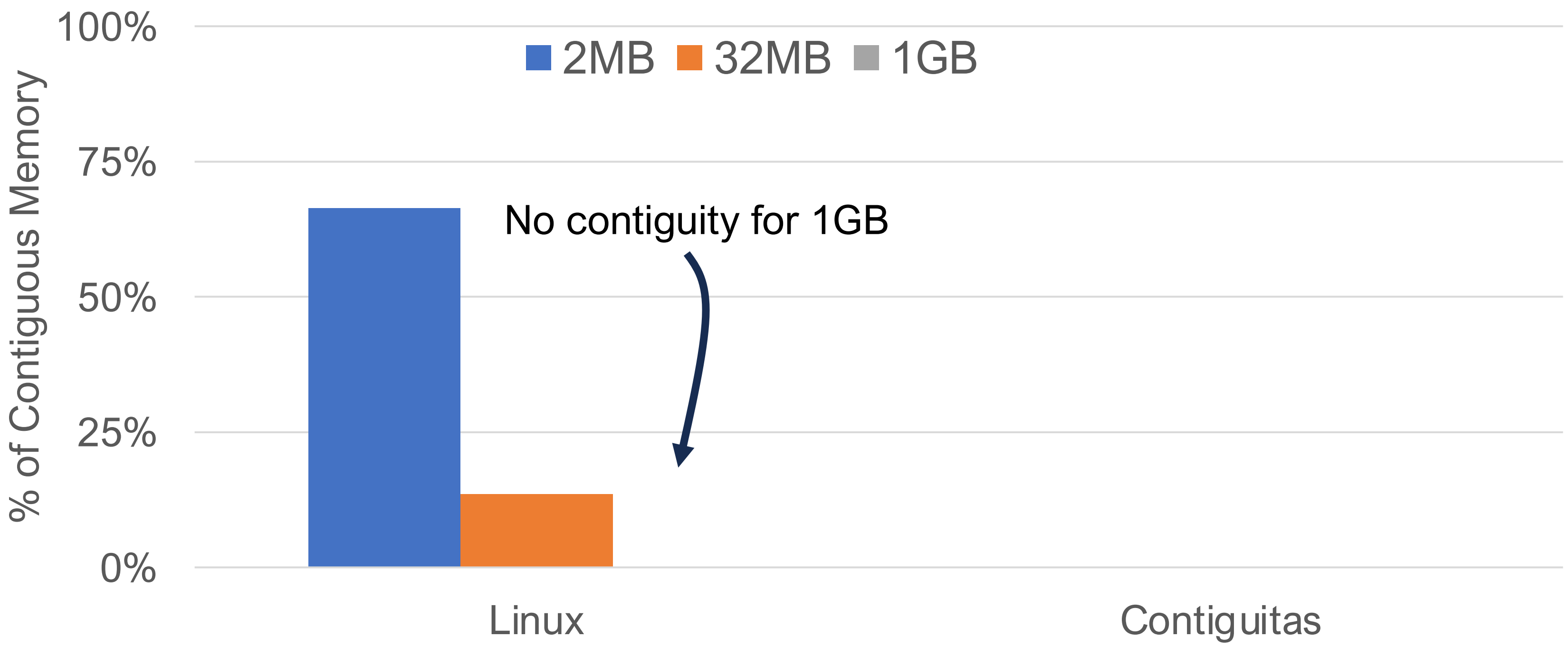
Live production traffic at Meta for Contiguitas OS

- Major workloads at Meta → Web and two caching services
- End-to-end requests per second (RPS)

Potential Memory Contiguity

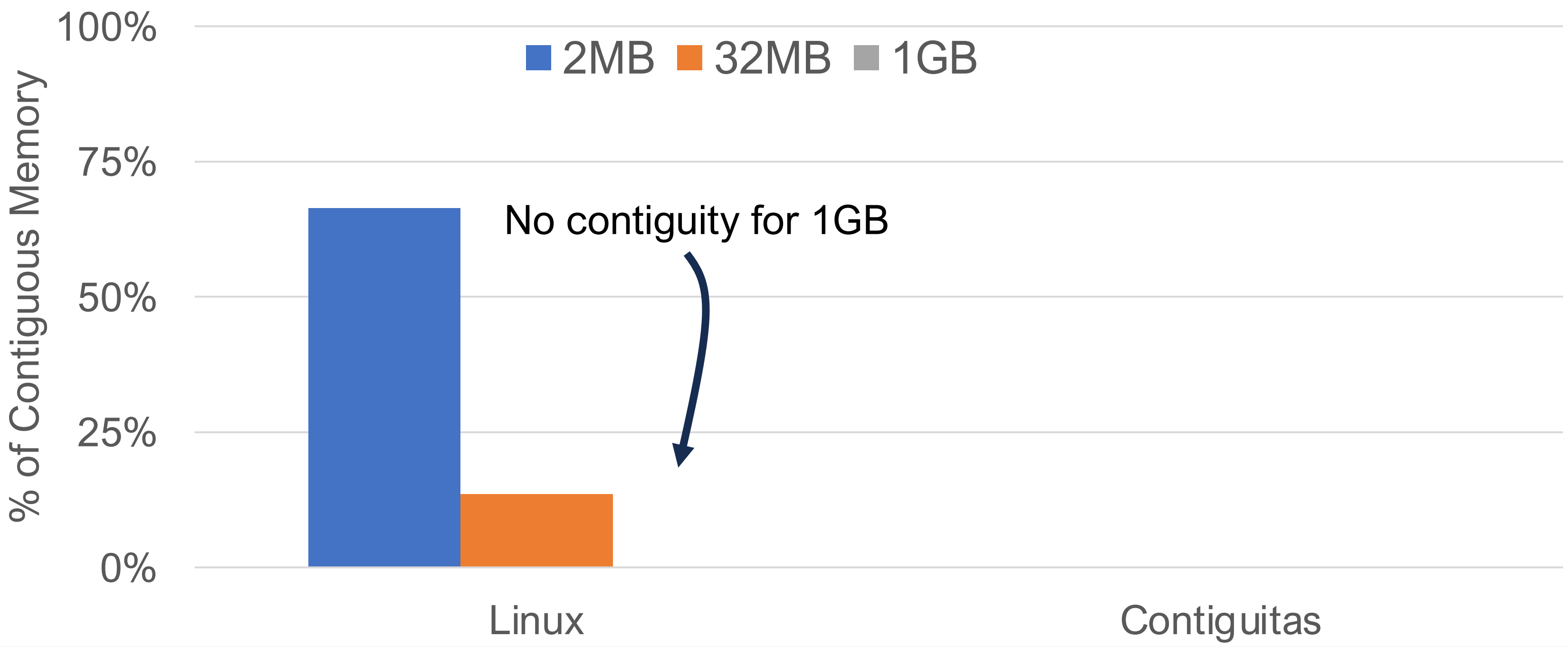


Potential Memory Contiguity



No contiguity for 1GB

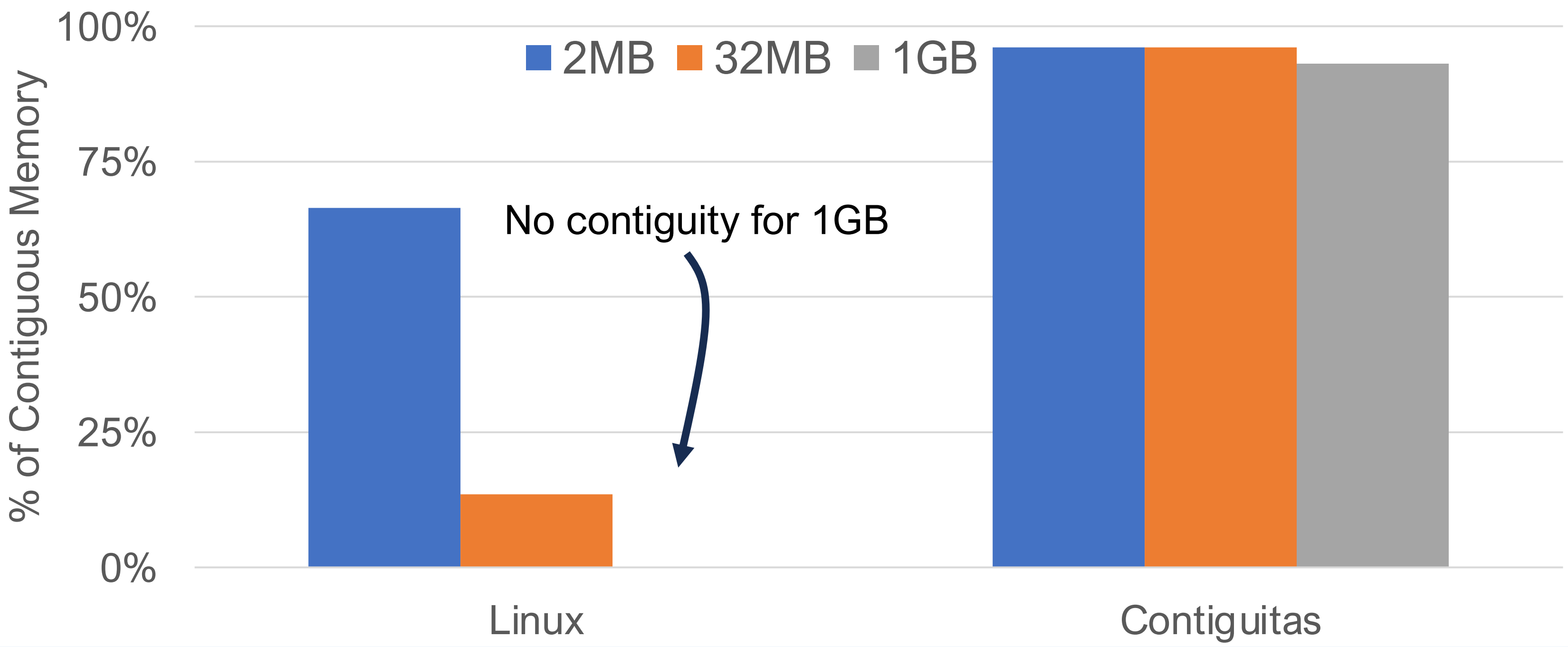
Potential Memory Contiguity



No contiguity for 1GB

Linux struggles to produce contiguous memory regions

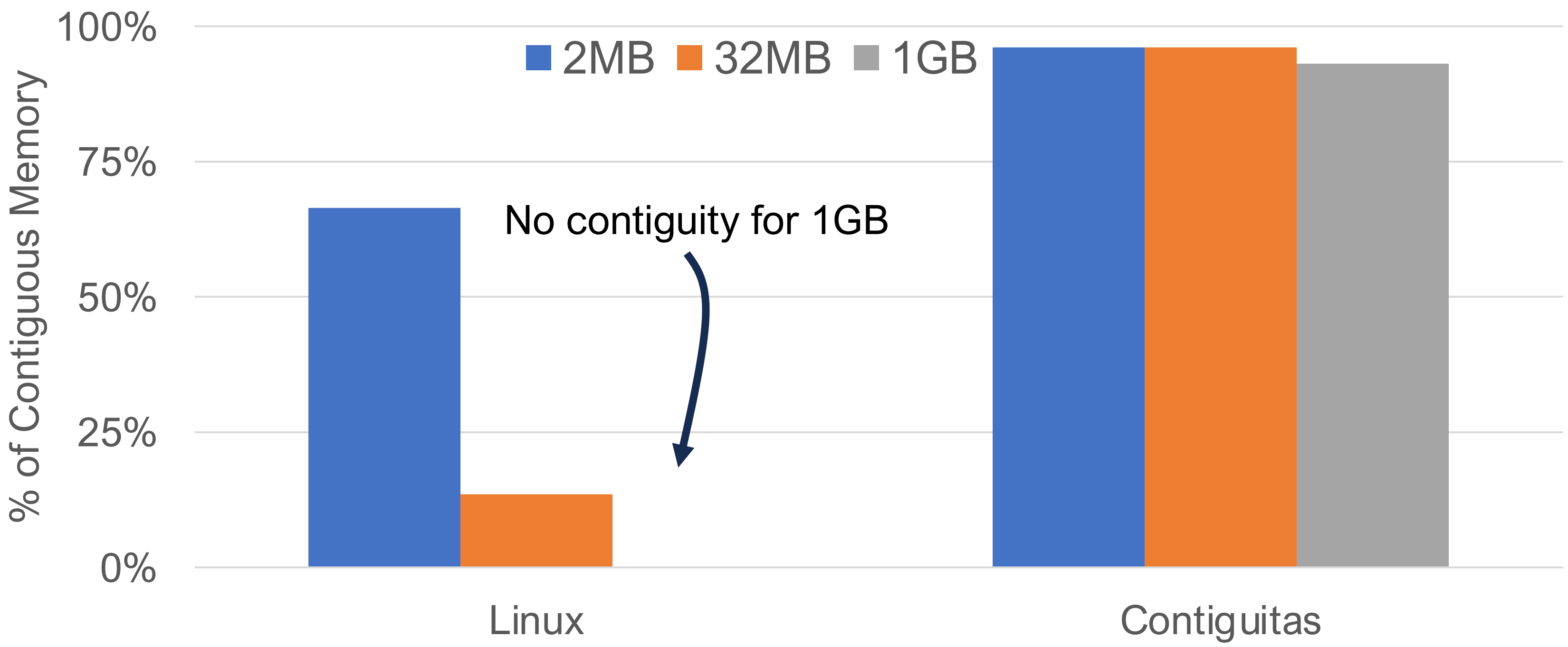
Potential Memory Contiguity



Linux struggles to produce contiguous memory regions

Contiguitas can use the entire movable region for contiguity

Potential Memory Contiguity

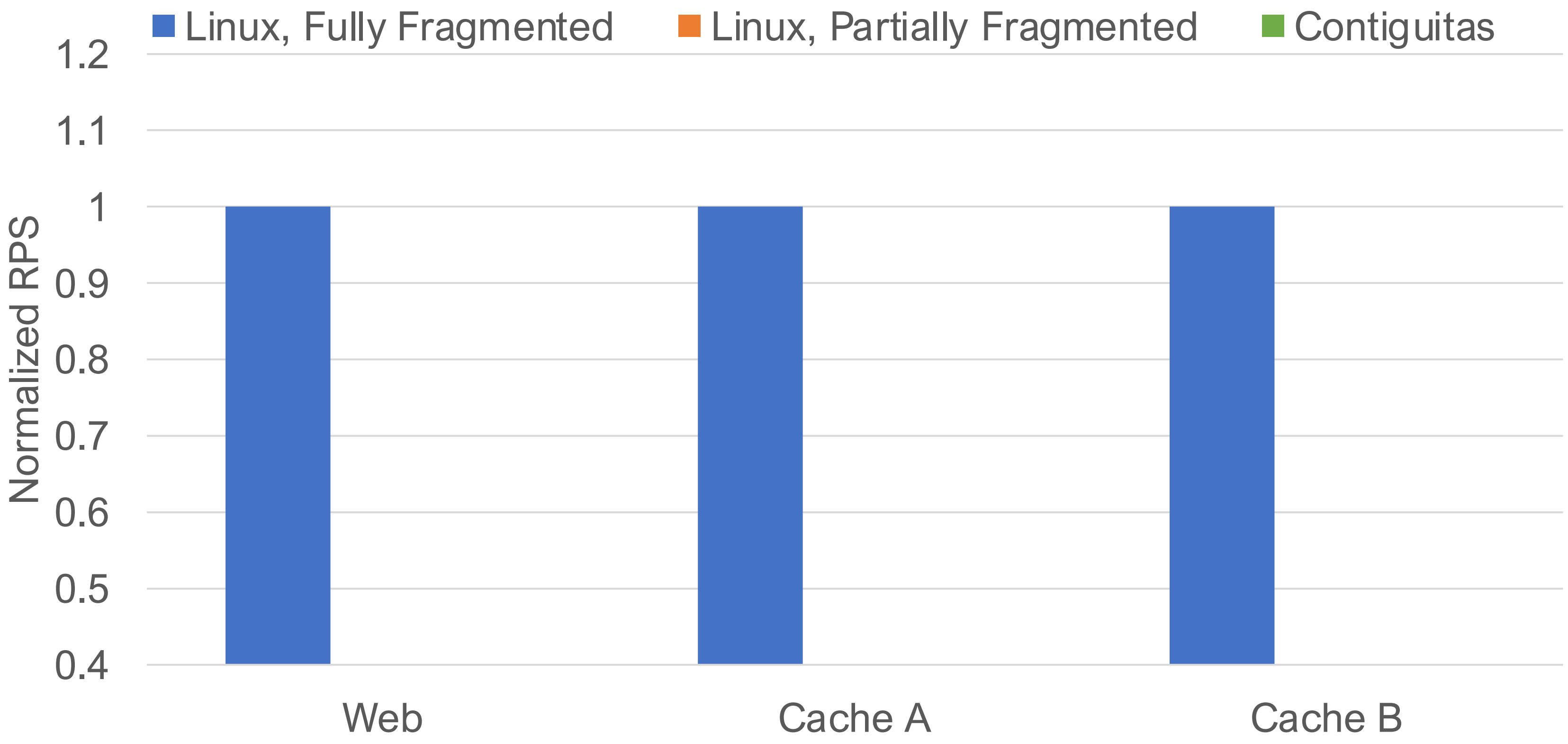


Linux struggles to produce contiguous memory regions

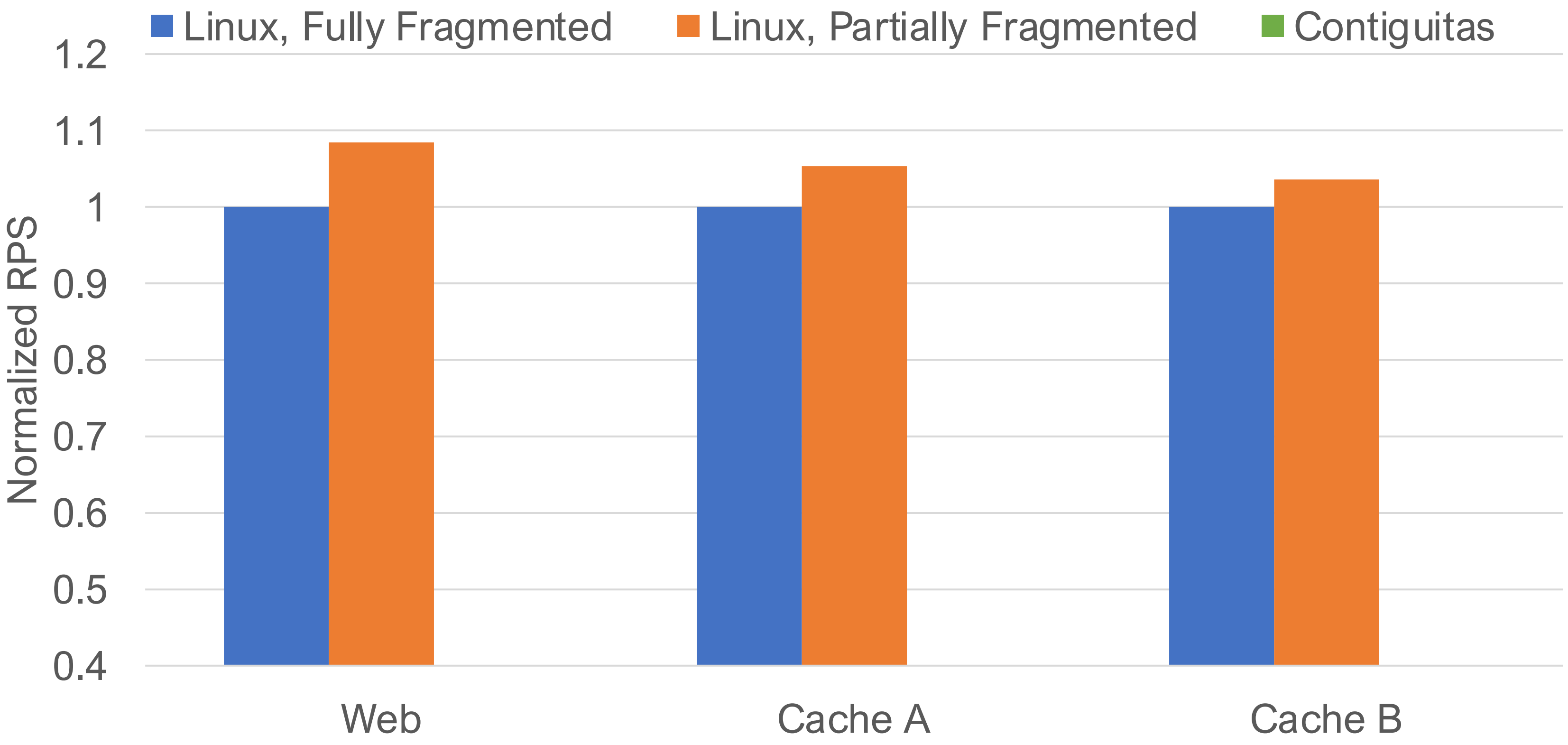
Contiguitas can use the entire movable region for contiguity

Contiguitas can dynamically allocate 1GB pages

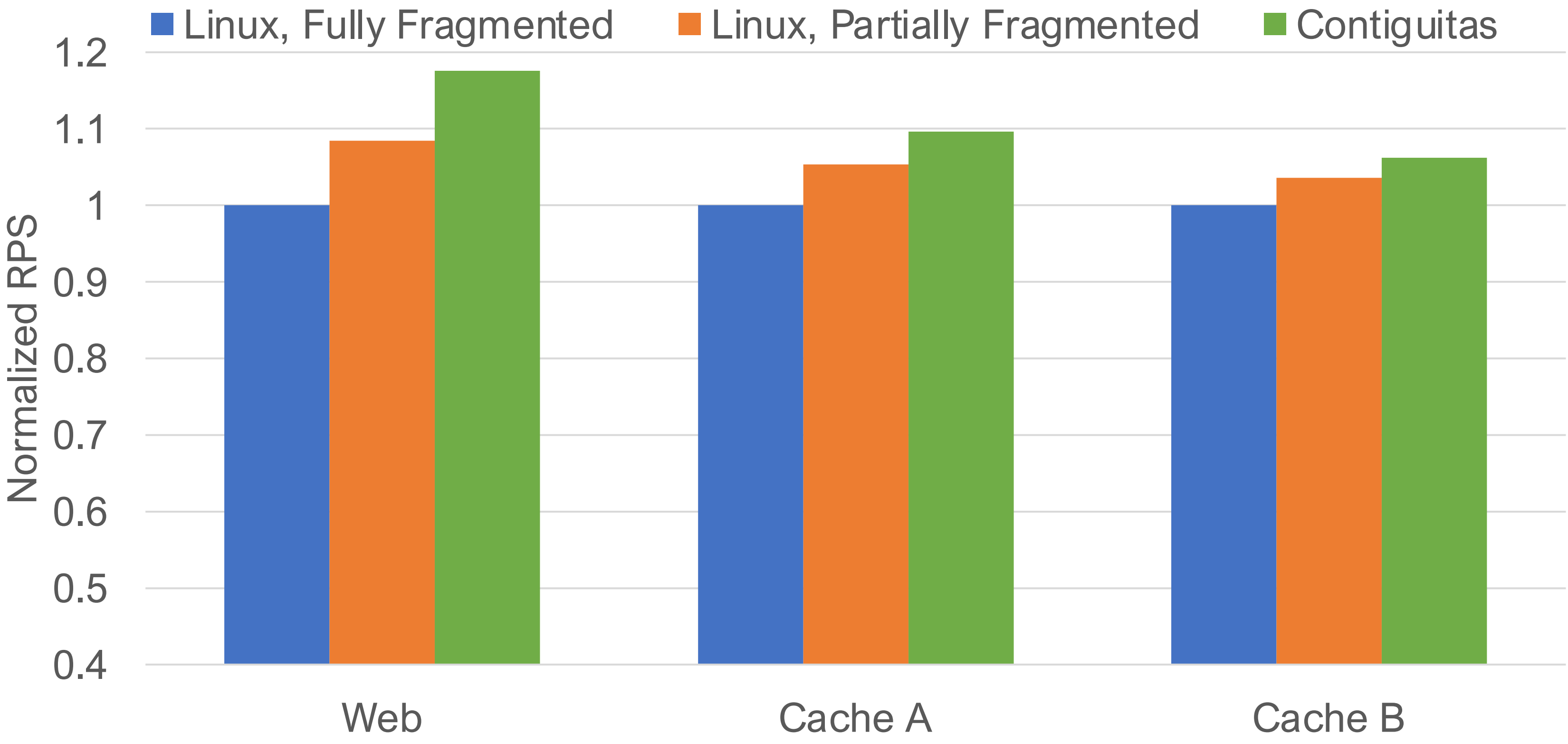
End-to-End Application Performance



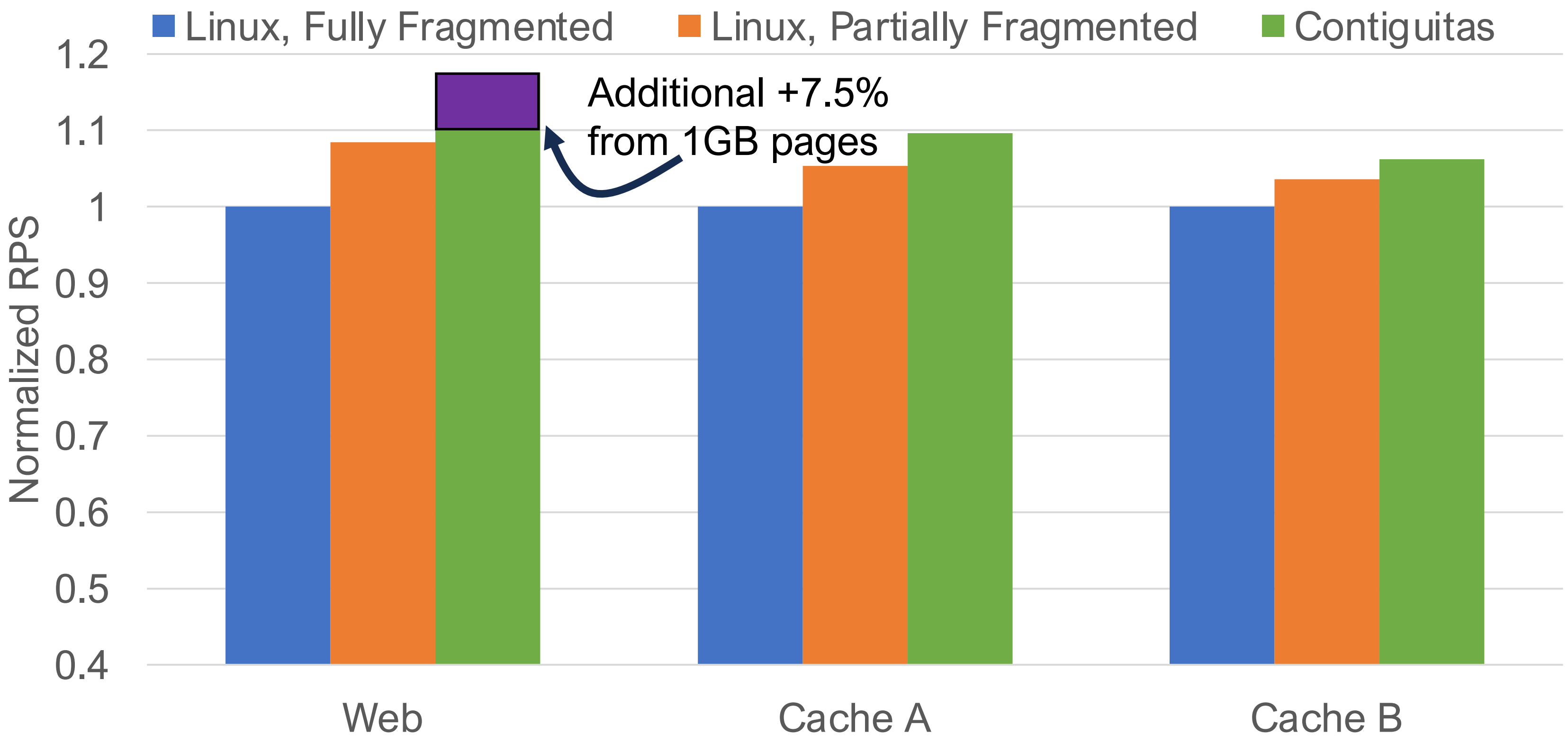
End-to-End Application Performance



End-to-End Application Performance



End-to-End Application Performance



Additional +7.5% from 1GB pages

Contiguitas achieves 2% - 18% speedup

Evaluation Overview

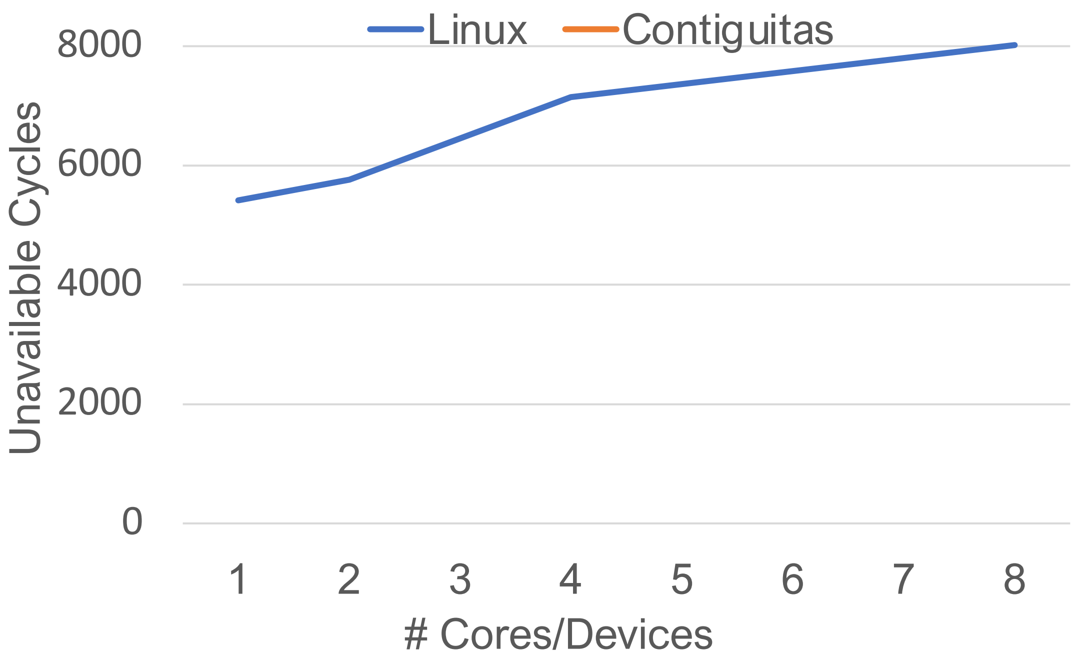
Live production traffic at Meta for Contiguitas OS

- Major workloads at Meta: web and two caching services
- End-to-end requests per second (RPS)

Full-system cycle-accurate simulation for hardware

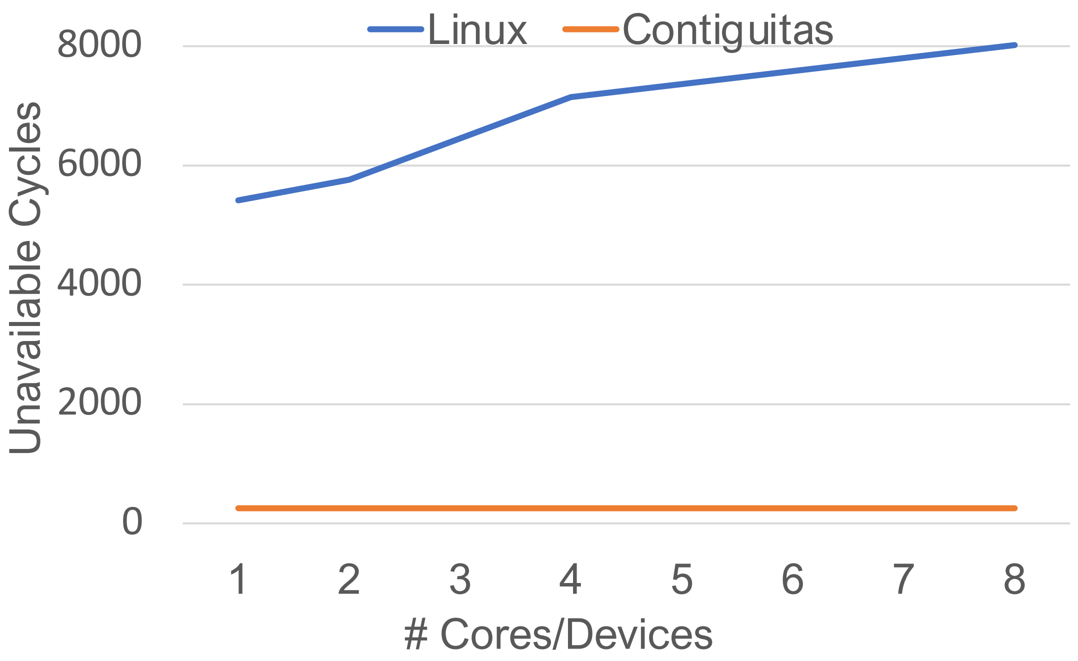
- Microbenchmark for page unavailable time
- Application performance impacts of page migration

Scalability of Contiguitas-HW



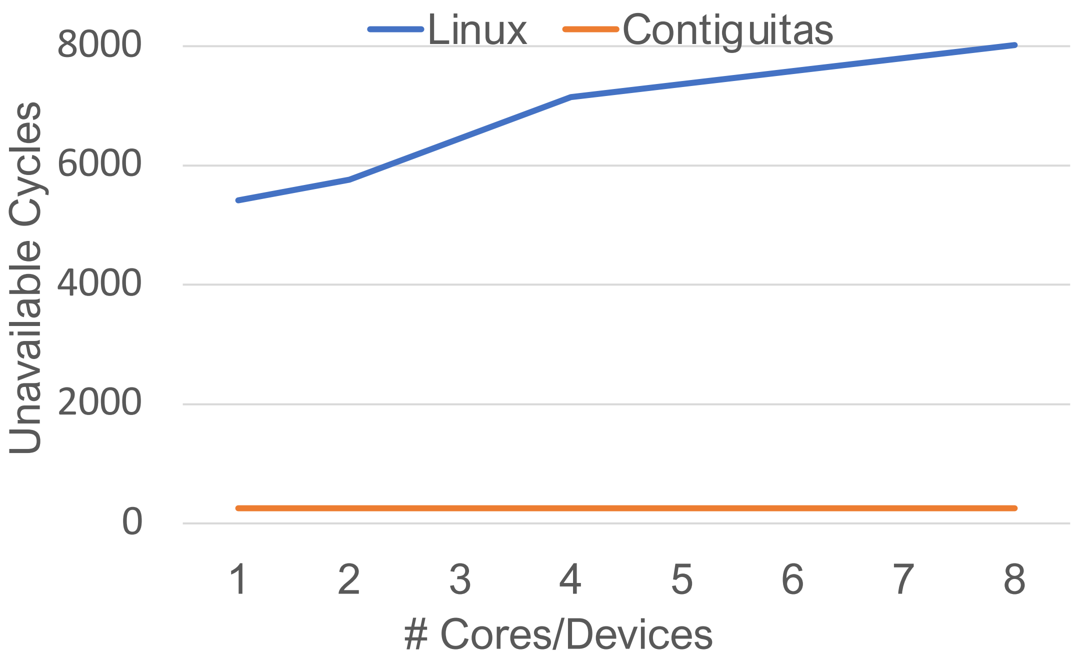
Migrating a page in software blocks for linear time w.r.t. TLBs

Scalability of Contiguitas-HW



Migrating a page in software blocks for linear time w.r.t. TLBs
Contiguitas-HW blocks for only local TLB invalidation

Scalability of Contiguitas-HW



Migrating a page in software blocks for linear time w.r.t. TLBs

Contiguitas-HW blocks for only local TLB invalidation

Negligible performance impacts to applications

Takeaways: Contiguitas

Unmovable pages are detrimental to contiguity

- They will only get worse

Contiguitas is a holistic solution across OS and hardware

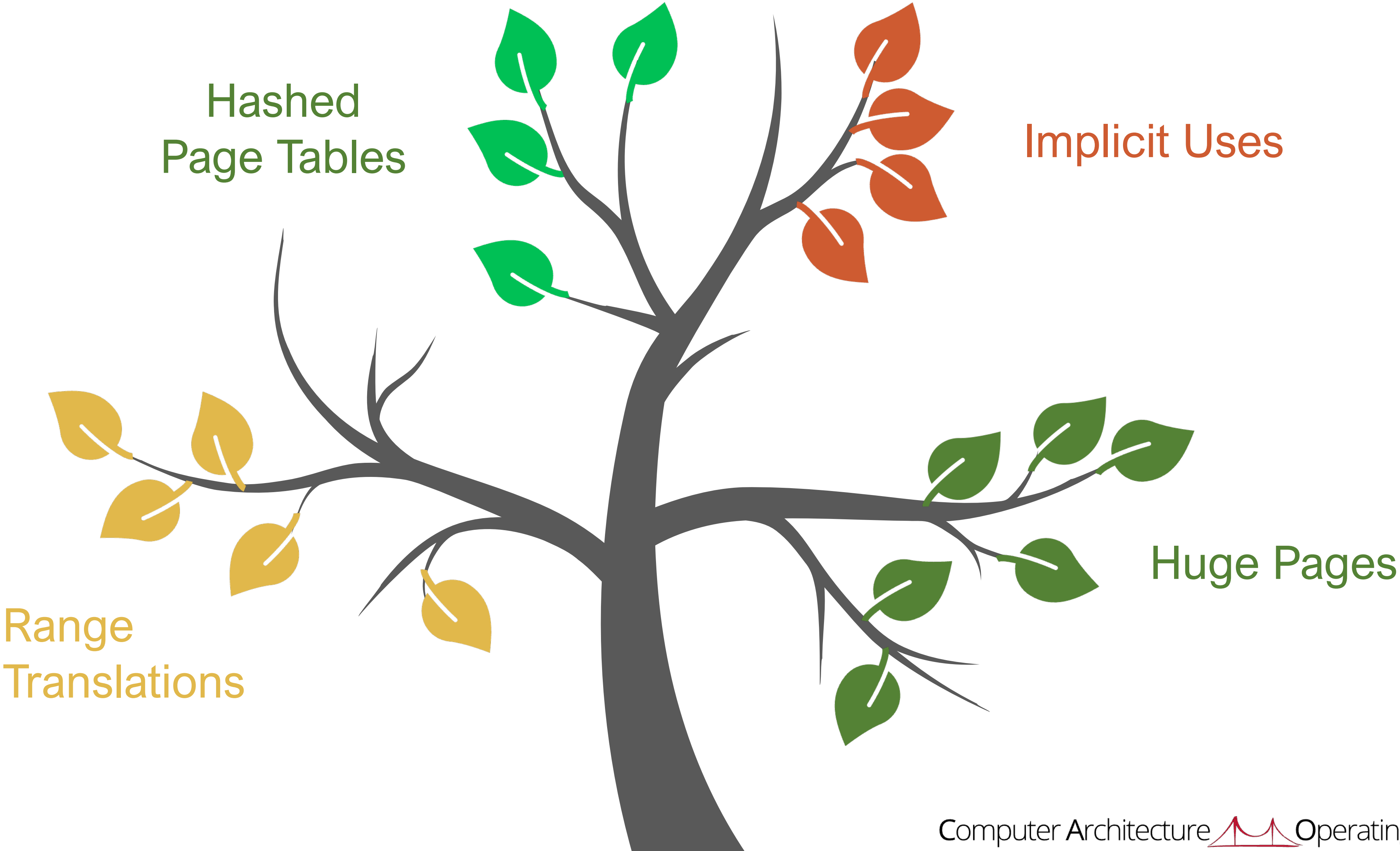
- Unmovable page confinement + transparent page migration
- Reduces memory fragmentation due to unmovable allocations

Ample physical memory contiguity
Performance gains of 2-18% with production traffic
Reduce unmovable pages with HW support

In the process of upstreaming to Linux

With Ample Physical Contiguity..

Pick your poison!



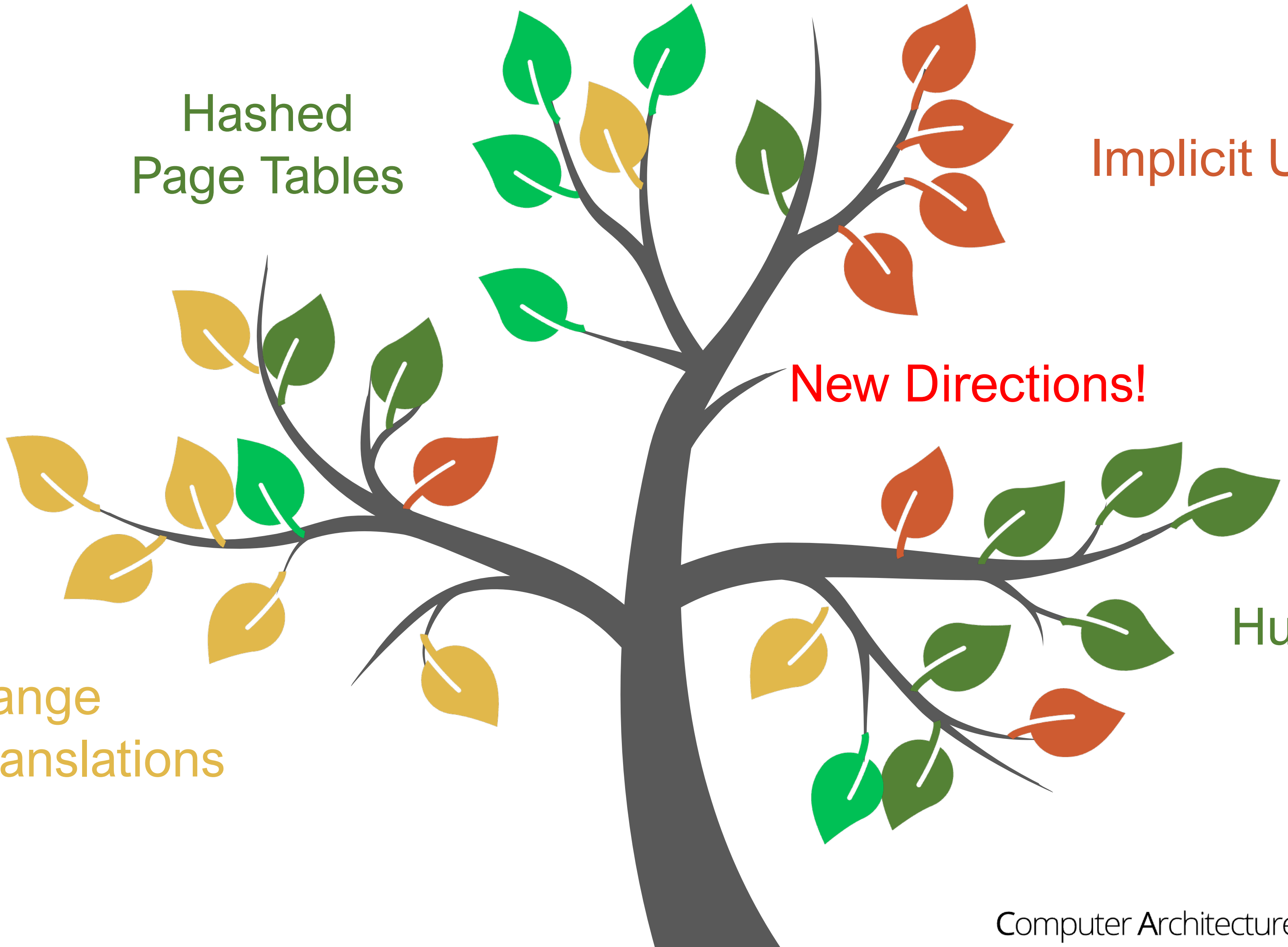
Hashed Page Tables

Implicit Uses

Range Translations

Huge Pages

Pick your poison!



Hashed Page Tables

Implicit Uses

New Directions!

Huge Pages

Range Translations

Contiguitas: The Pursuit of Physical Memory Contiguity in Datacenters

Kaiyang Zhao, Kaiwen Xue, Ziqi Wang, Dan Schatzberg, Leon Yang,
Antonis Manousis, Johannes Weiner, Rik van Riel, Bikash Sharma, Chunqiang Tang,
Dimitrios Skarlatos

Supported by:

NSF intel Meta