Barre Chord: Efficient Virtual Memory Translation for Multi-Chip-Module GPUs

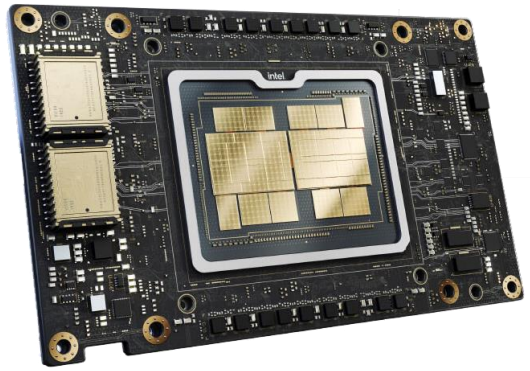
Yuan Feng∗, Seonjin Na†, Hyesoon Kim†, Hyeran Jeon∗

∗**University of California, Merced** †Georgia Institute of Technology

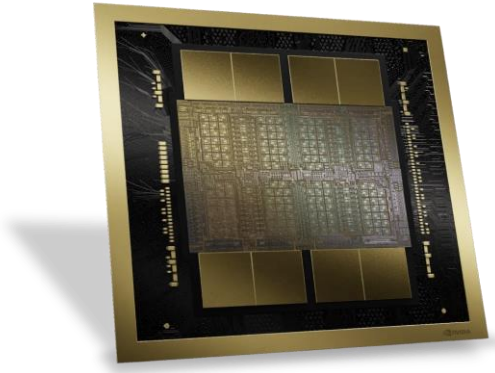
MCM-GPU is a trend!

- Big data workloads require tremendously large compute and memory capability
  - Chip yield decreases as the die getting large
  - **Multi-Chip Module (MCM) GPUs** are on the market!

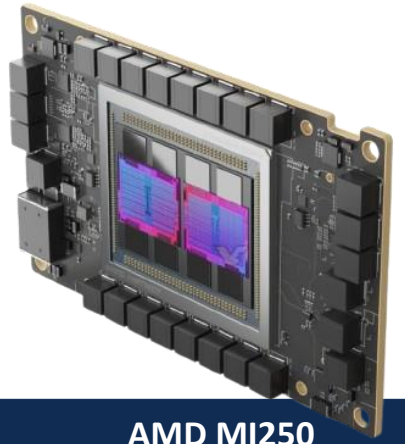
- Virtual Memory (VM) is essential to efficiently manage the resources
  - Smart page migration and allocation rely on VM
  - But **is the VM translation efficient for MCM-GPU?**



Intel Ponte Vecchio



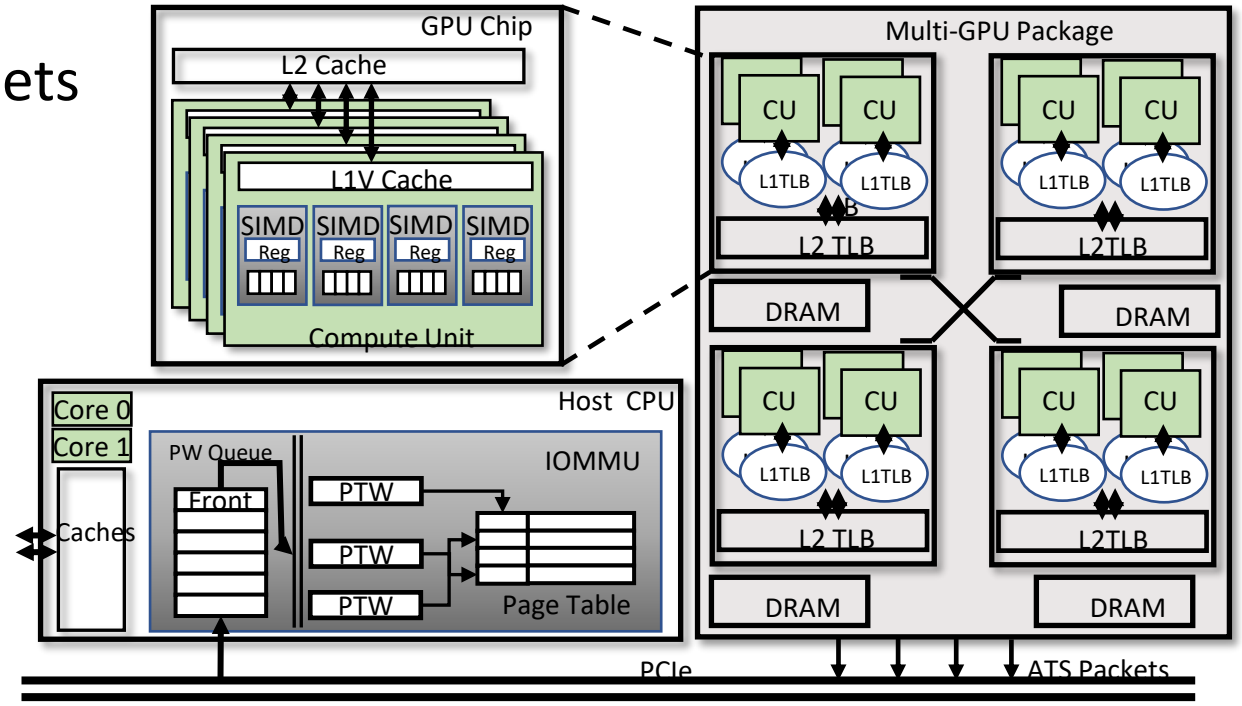
NVIDIA Blackwell



AMD MI250

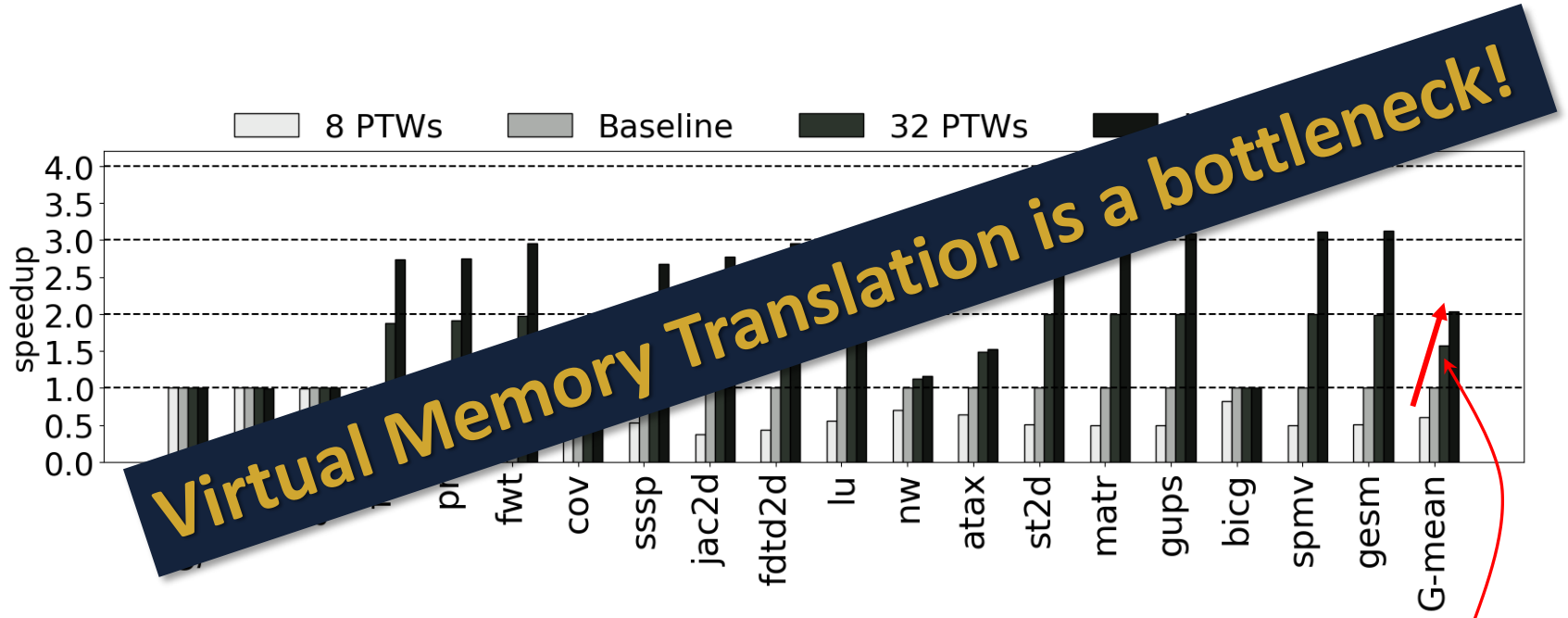
Baseline architecture

- A GPU package connected to Host CPU via PCIe
- A GPU package contains multiple GPU Chiplets
- Each Chiplet is a fully functional GPU
  - L1/L2 caches, L1/L2 TLBs
  - Dedicated DRAM
  - High-bandwidth Interconnection (768 GBps)



Virtual Memory Translation Performance

- Speedup with more Page Table Walkers (PTWs)

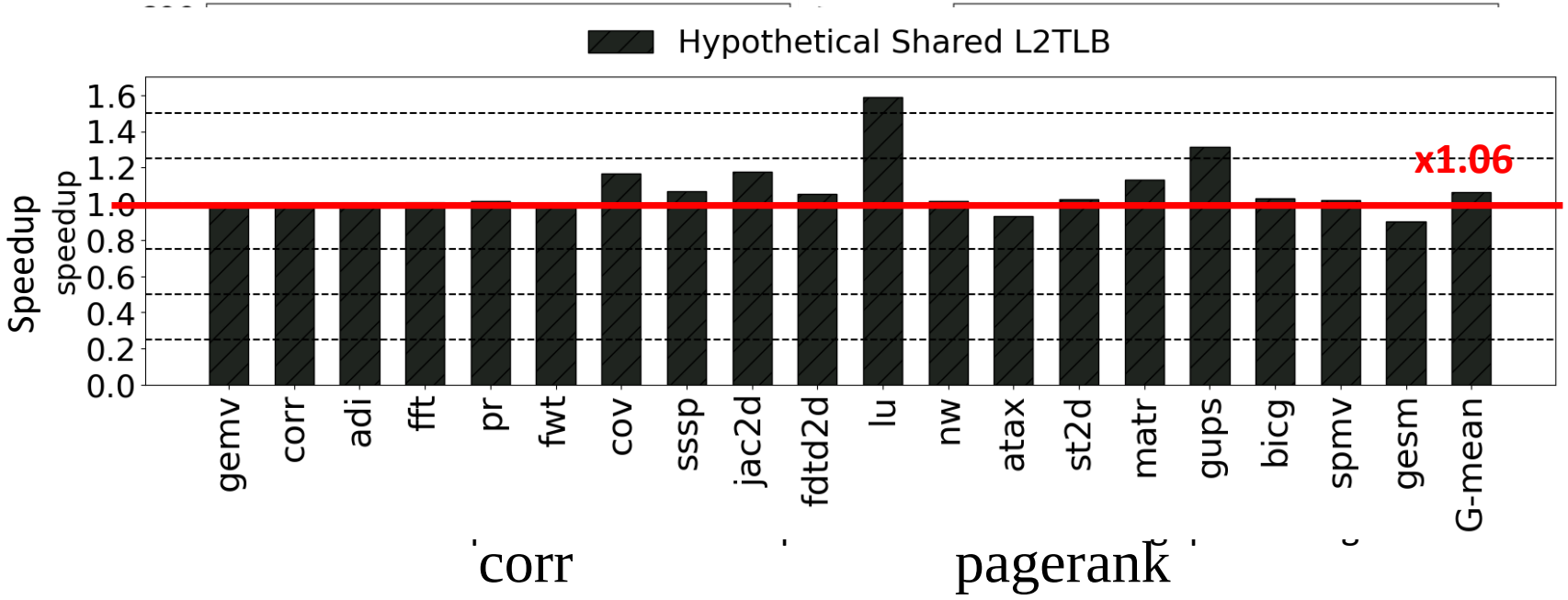


Speedups with 8, 16, 32, and infinite PTWs

Almost linear speedup while saturating at 2x

Conventional wisdom provides limited help

- Traditional VM optimizations:
  - ~~Big/super page?~~
  - ~~TLB prefetching?~~
  - ~~TLB sharing?~~

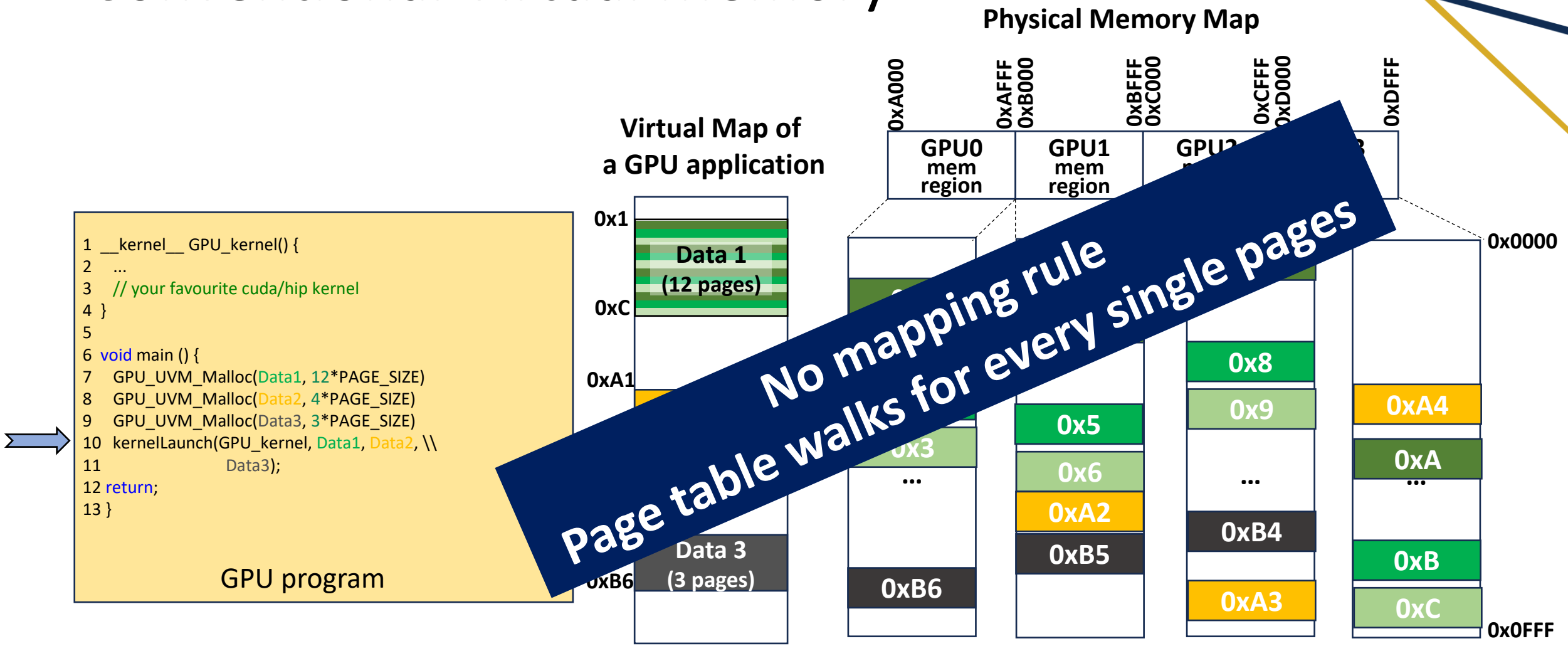


Idea: Leverage the Goal: Provide an Efficient Virtual Memory Framework for Chiplets to

fundamentally remove page table accesses

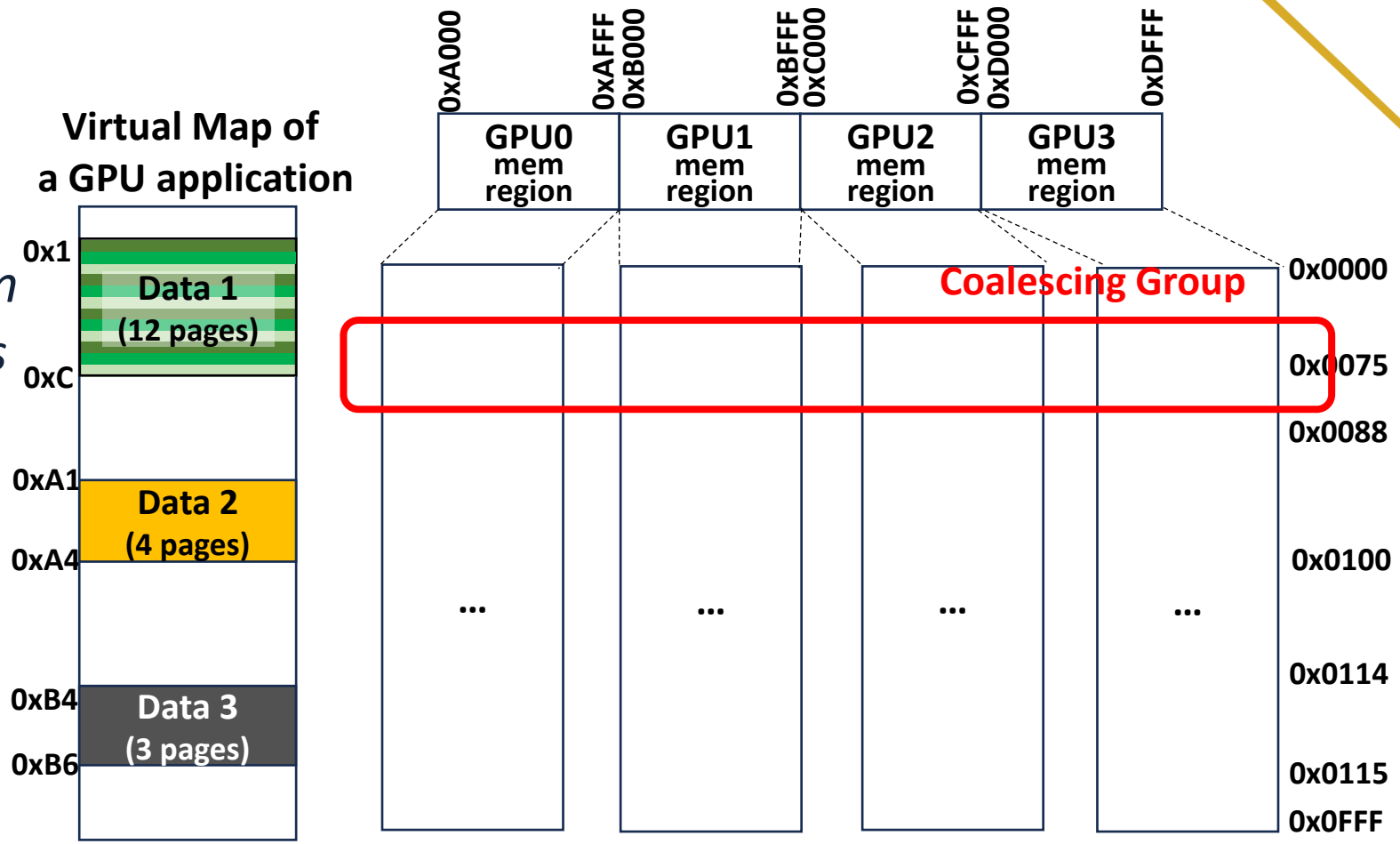
Barre Chord = Barre + Full Barre

Conventional Virtual Memory

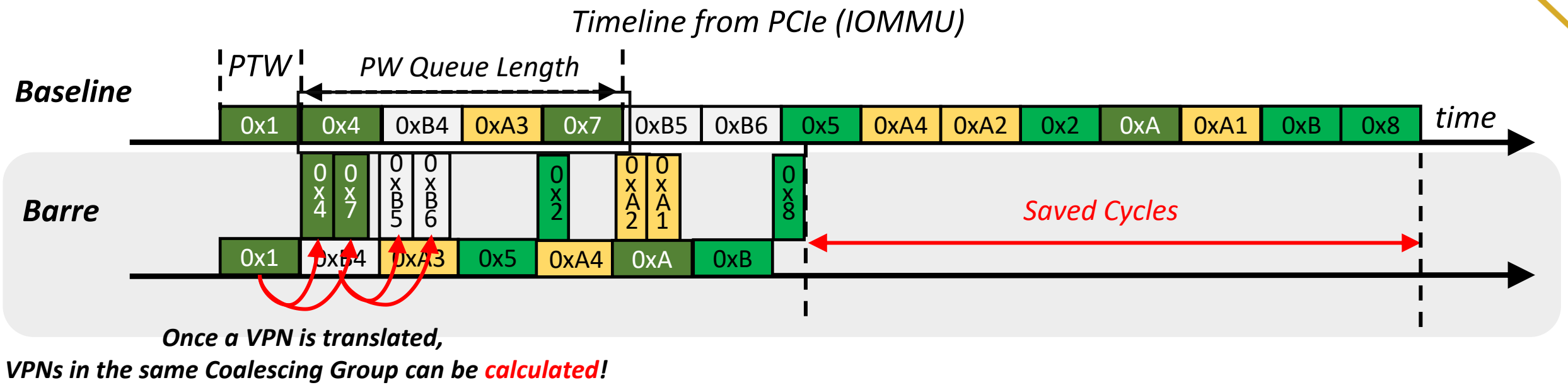


Barre – Page Mapping in **Coalescing Group** Unit

Idea: Map pages of the same data on the same local physical address across chiplets



Barre – Translate in Coalescing Group Unit!



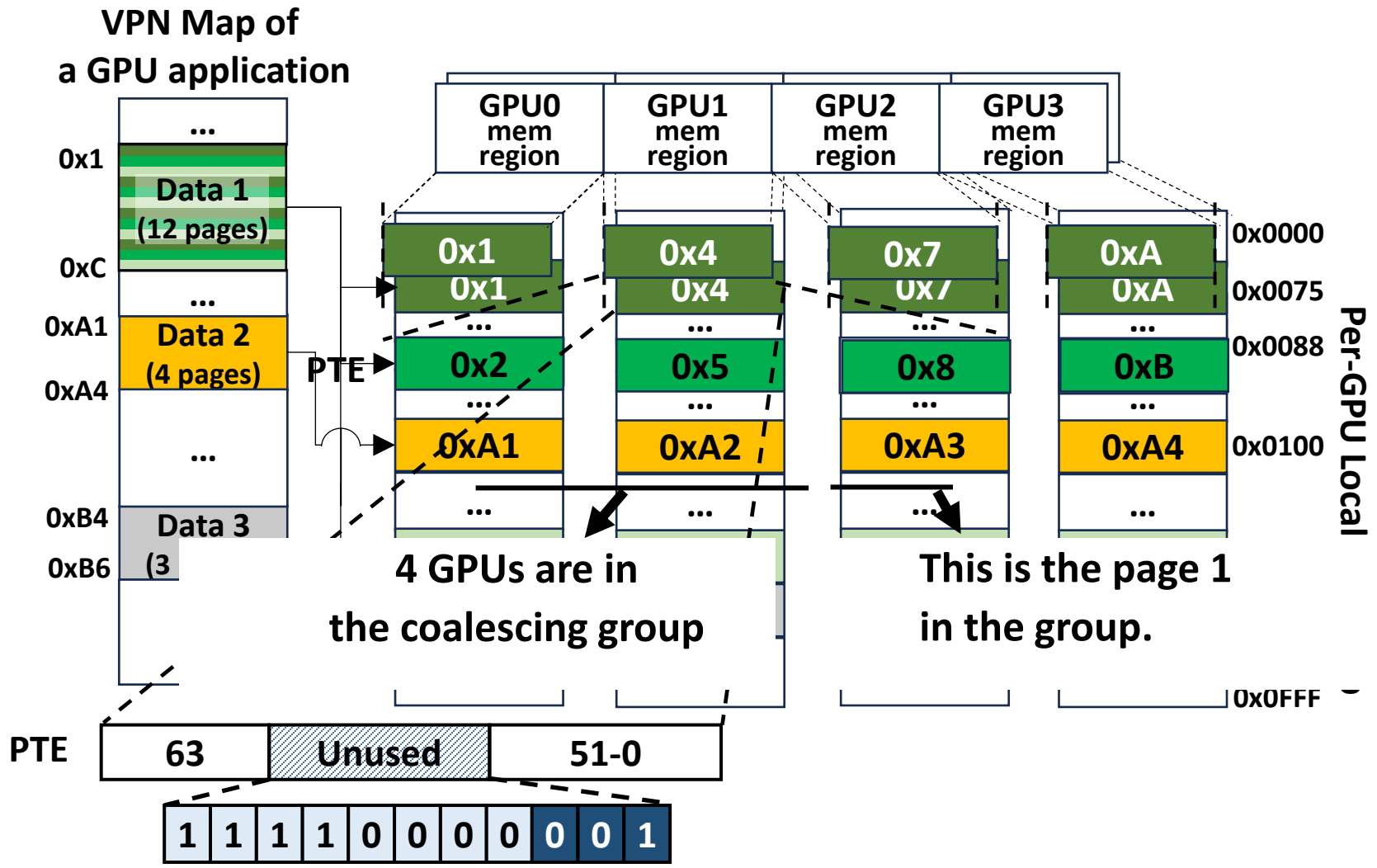
Once a VPN is translated,

VPNs in the same Coalescing Group can be calculated!

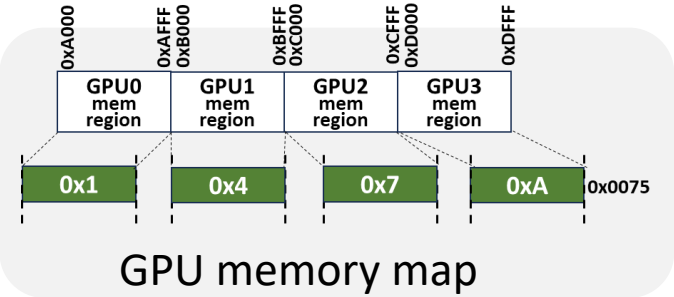
- Without Barre => 19 Page Table Walks
- With Barre => 7 Page Table Walks

* Example when PW-queue has 4 entries

Barre – Page Table Entry Revision



Barre – Translate Address via Calculation!

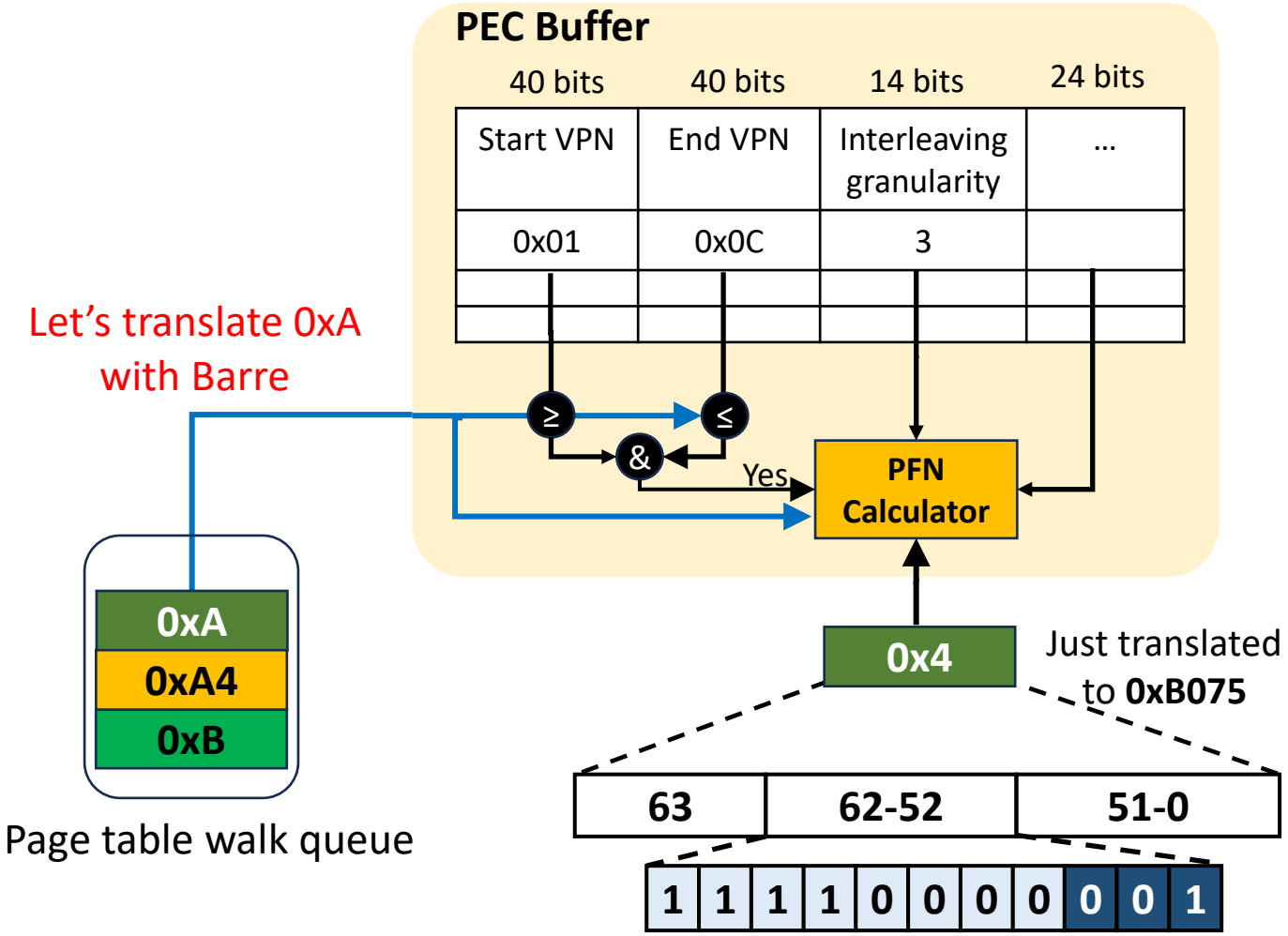


GPU memory map

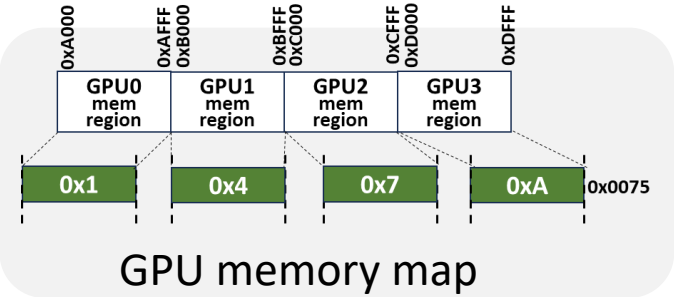
Start VPN (0x01) **< 0xA < End VPN** (0x0C)

→ **In the same data**

Let's translate 0xA with Barre



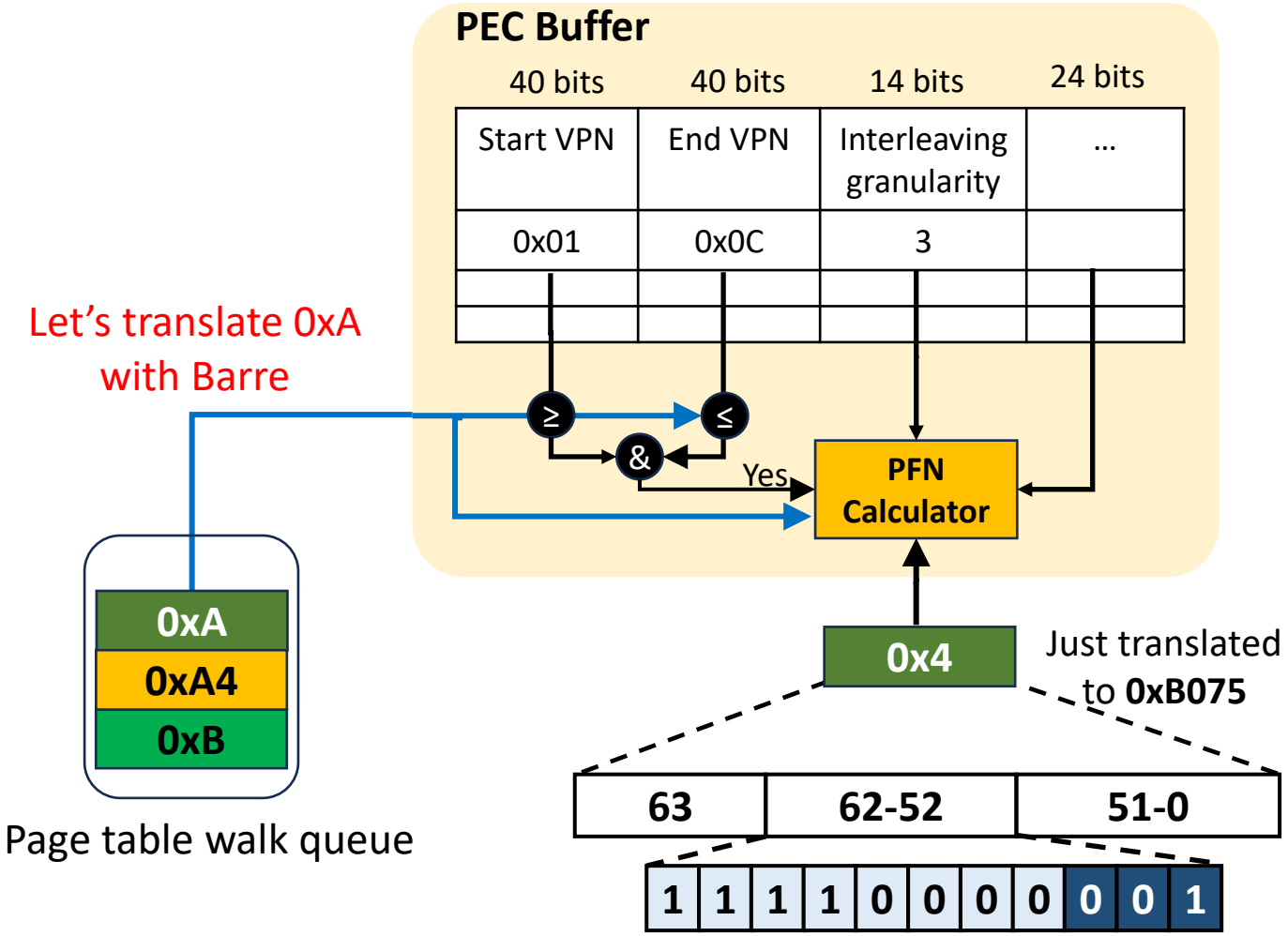
Barre – Translate Address via Calculation!



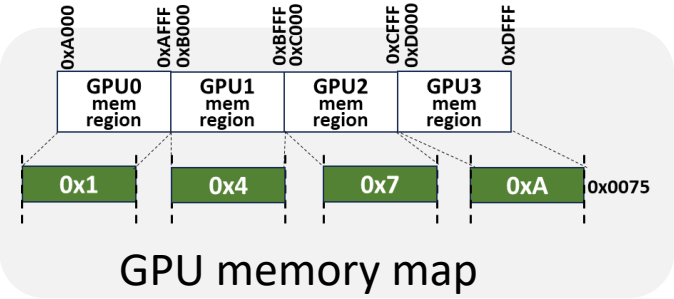
GPU memory map

(0xA - 0x4) is integer multiple of **interlv. Gran (3)**

→ **In the same coalescing group**



Barre – Translate Address via Calculation!



GPU memory map

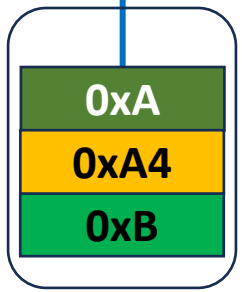
$$\frac{\text{0xA - 0x4}}{\text{interlv. Gran (3)}} = 2$$

→ 0xA is 2 GPUs away from 0x4

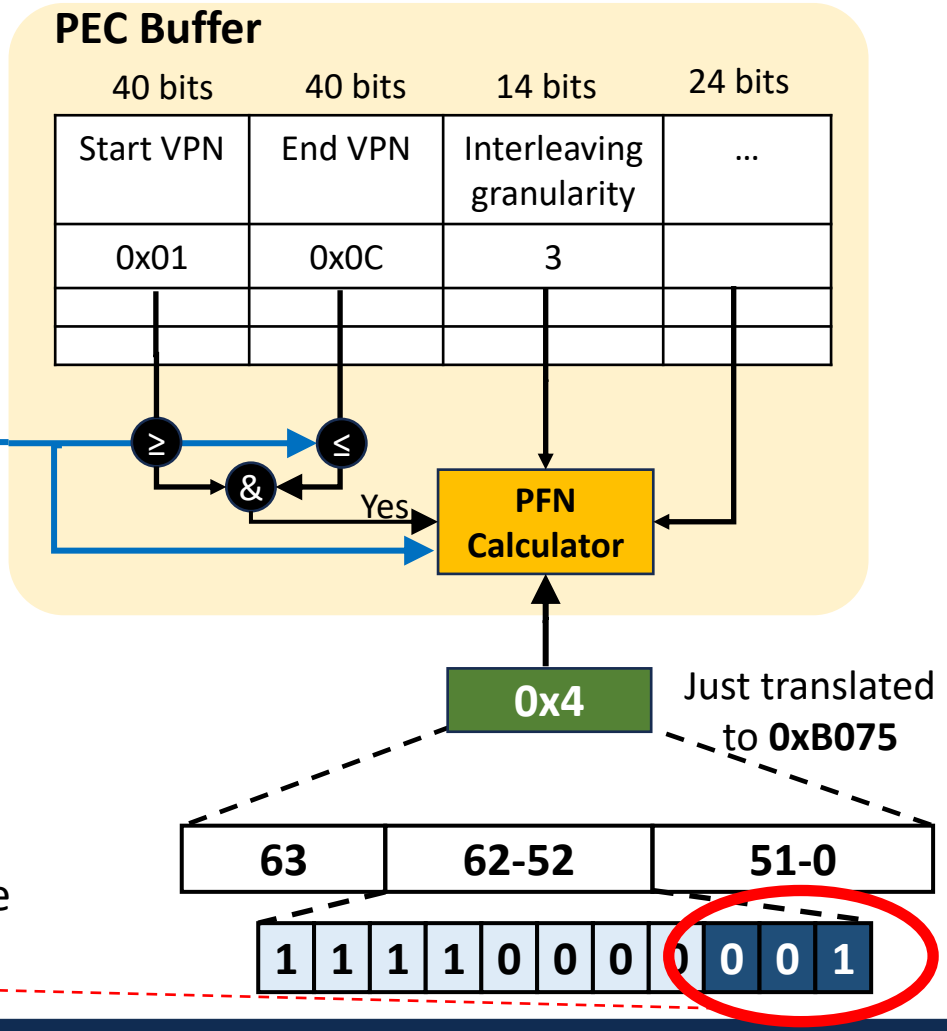
0x4 is in GPU 1

→ 0xA is in GPU 3

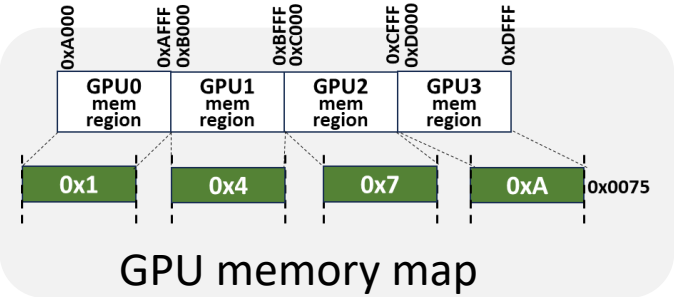
Let's translate 0xA with Barre



Page table walk queue



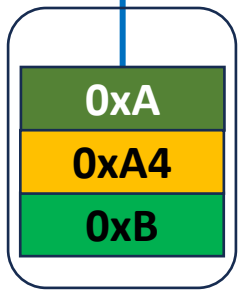
Barre – Translate Address via Calculation!



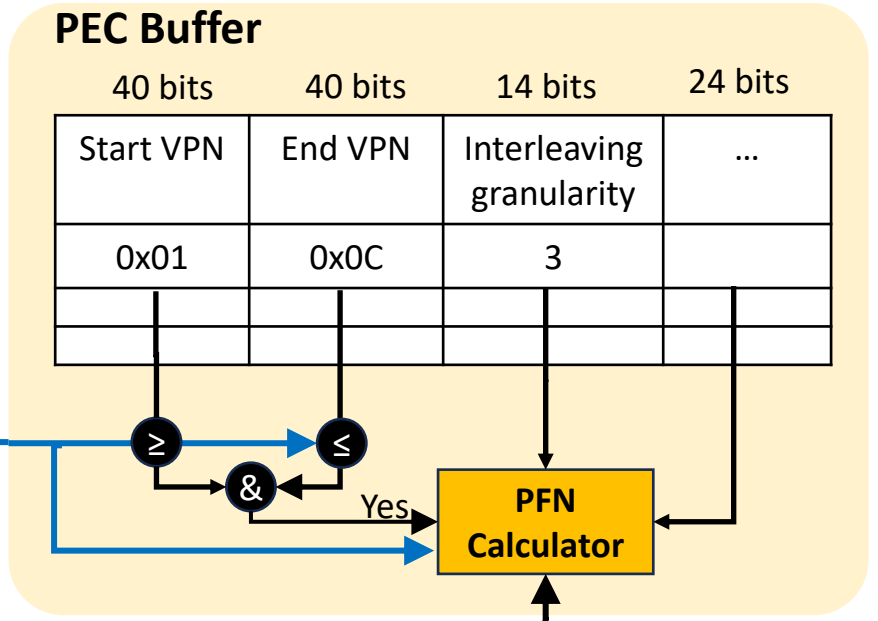
GPU memory map

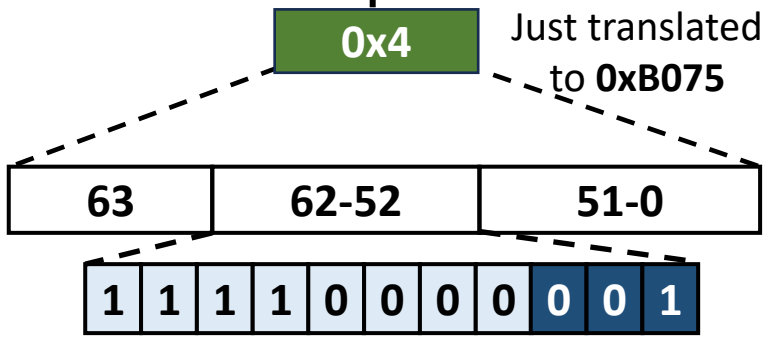
0x4's PFN **0xB075**
GPU1's base address **- 0xB000**
GPU3's base address **+ 0xD000**
Done! **0xD075**

Let's translate 0xA with Barre



Page table walk queue





Barre – Simple with Room for Optimization

- GPU addresses are still translated on Host via slow PCIe
  ⇒ Intra-GPU translation
- The coalescing group size is limited to the number of sharer GPUs
  ⇒ Contiguity-aware coalescing group expansion
- The address oblivious PTW scheduling drops the opportunities of coalesced PFN calculation.
  ⇒ Coalescing-aware PTW scheduling

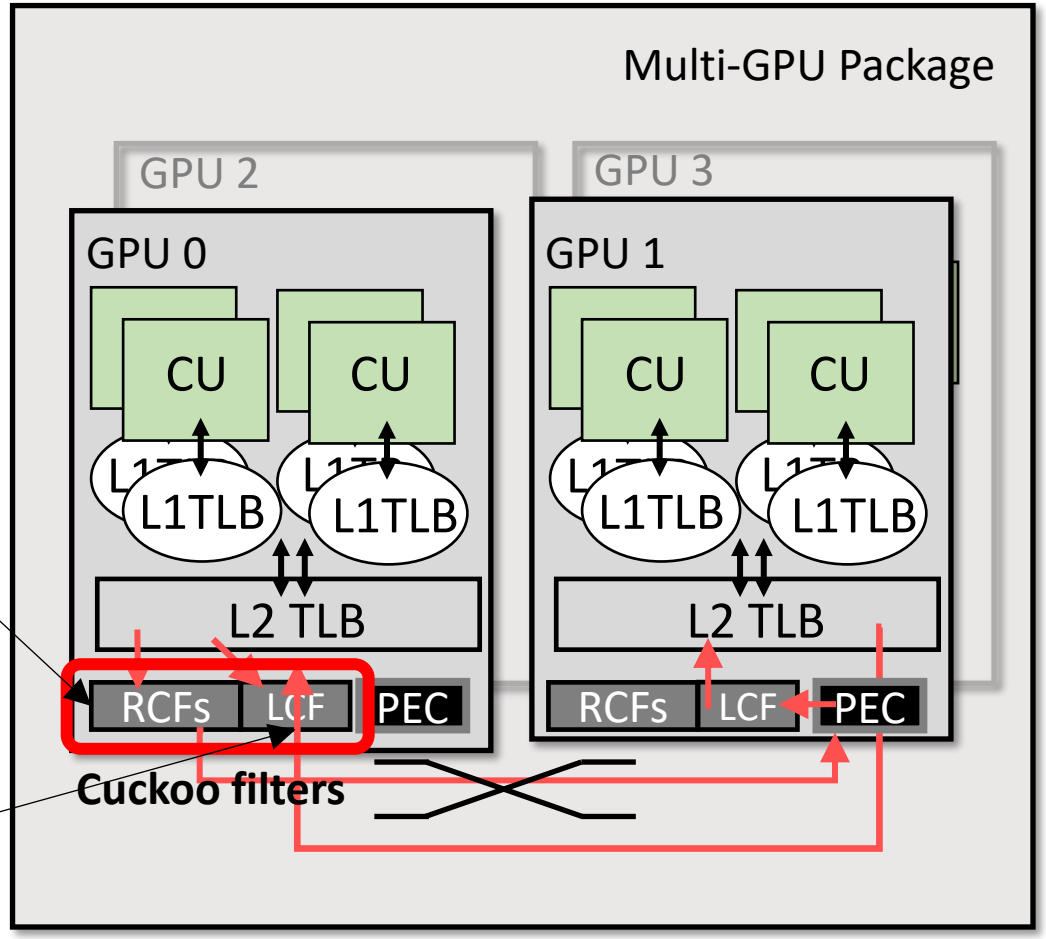
Full-Barre
(or F-Barre)

F-Barre – Intra-GPU Translation

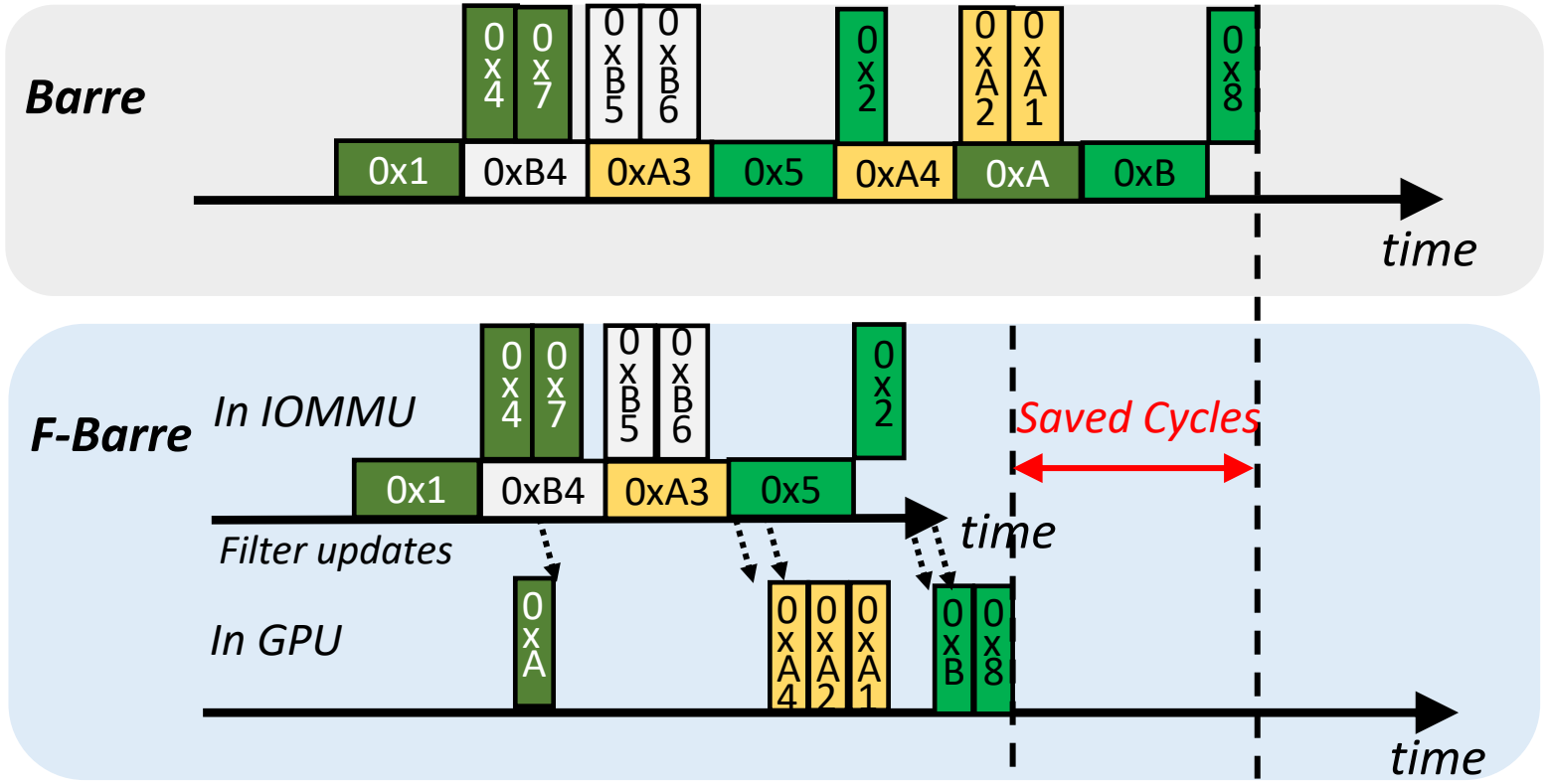
Idea: Share coalescing information among chiplets and calculate!

Which remote chiplet has the coalescing group information?

Does this chiplet have coalescing group information?



F-Barre – Intra-GPU Translation

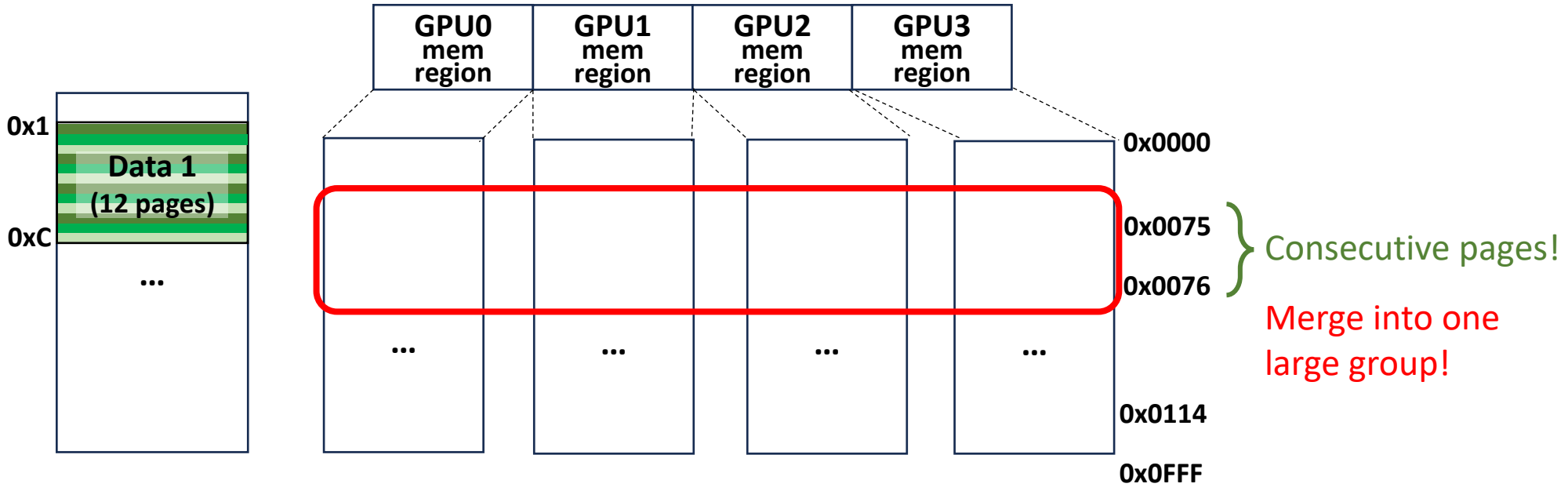


Coalescing relies on IOMMU.

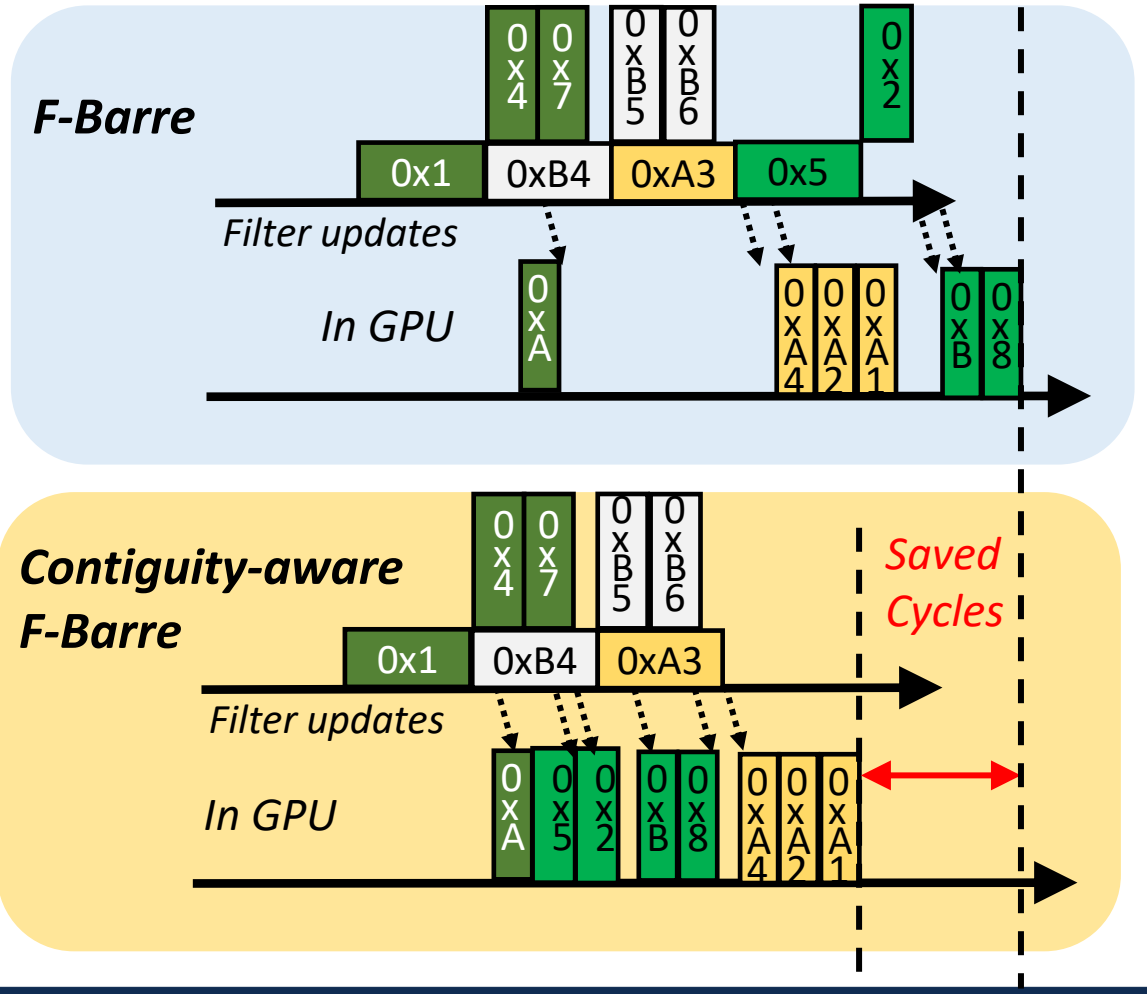
Coalescing happens within GPU!

F-Barre – Contiguity-aware coalescing group expansion

Idea: Opportunistically expand coalescing group when memory allocation contiguity is observed.



F-Barre – Contiguity-aware coalescing group expansion



Limited by # of Chips
One page table walk per 4 PTE

Larger coalescing information sharing reach!

One page table walk per 8 PTE

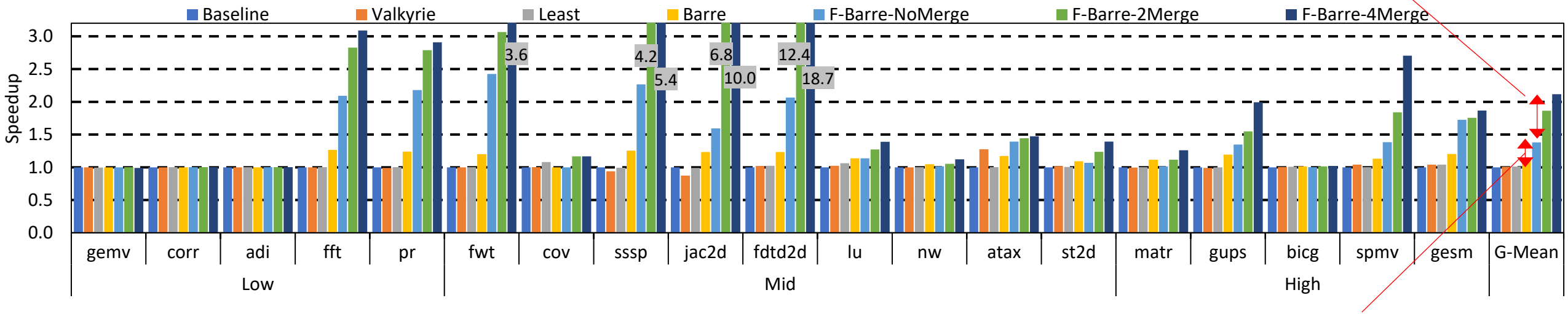
Evaluation

- Cycle-level simulator MGPUSim
- GCN3-like GPU
- 4-chiplet
  - 768GBps inter-chip bandwidth
- CPU-GPU connection
  - ~150 cycle latency
  - PCIe gen4 x16 bandwidth
- IOMMU
  - 16 PT walker
  - 48-entry page walk queue

Parameter	Value
Number of GPU chips	4
Number of SAs	4 per Chip
Number of CUs	16 per SA. 256 in total
L1 Vector Cache	16 KB, 4-Way, 16 MSHRs
L1 Inst Cache	32 KB, 4-Way, 16 MSHRs
L1 Scalar Cache	16 KB, 4-Way, 16 MSHRs
L2 Cache	2 MB, 16-way, 64 MSHRs
DRAM	1 TBps, 100ns latency [37]
L1 TLB	64 entries, fully connected, 16 MSHRs, 1 cycle lookup latency, private to CU, LRU.
L2 TLB	512 entries, 16-Way, GPU chip-shared. 10 cycle lookup latency, 16 MSHRs.
IOMMU	16 shared PTWs. 500-cycle page table walks [25], [44], 48 PW-queue entries.
CTA/Page Scheduling	LASP [20]
Inter-chip bandwidth	768 GB/s mesh, 32 cycle latency [3], [48]
CPU-GPU Connection	PCIe Gen4 x16. 150 cycle latency [24], [25]
Cuckoo Filter	9-bit fingerprint, 4-way, 256 Rows (1024 entries) per filter
Merged Coalescing group	2 by default
PEC buffer	5 entries of 118 bits each

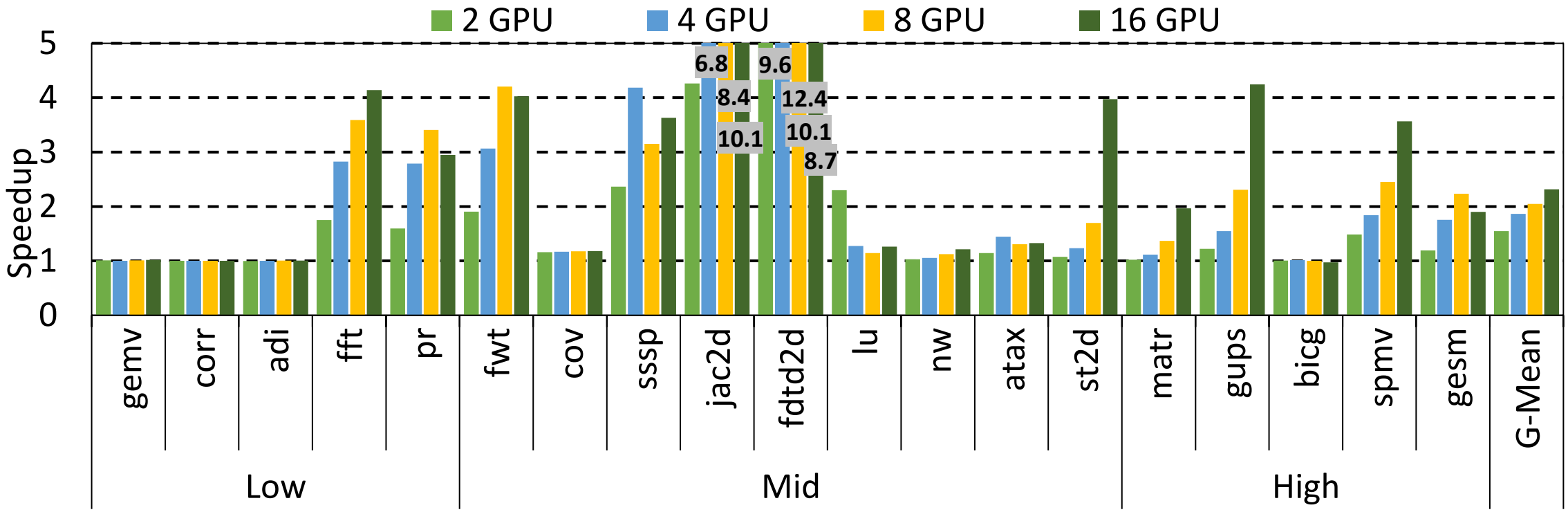
Evaluation – Overall performance

1.53x MORE speedup with contiguity-aware coalescing



1.36x speedup over the SOTA

Evaluation – Scalability



More Chiplets, Larger Coalescing Groups, Better Speedup!

Evaluation – More details

- Breakdown
- Page migration
- GMMU
- Trans
- RCF,
- Sharing Traffic overhead
- PTW sensitivity
- Super pages
- Much more….

Please check our paper for more details!

QnA

I’m also looking for an intern position starting from this fall.


THE UNIVERSITY OF CALIFORNIA MERCED
LET THERE BE LIGHT
2005

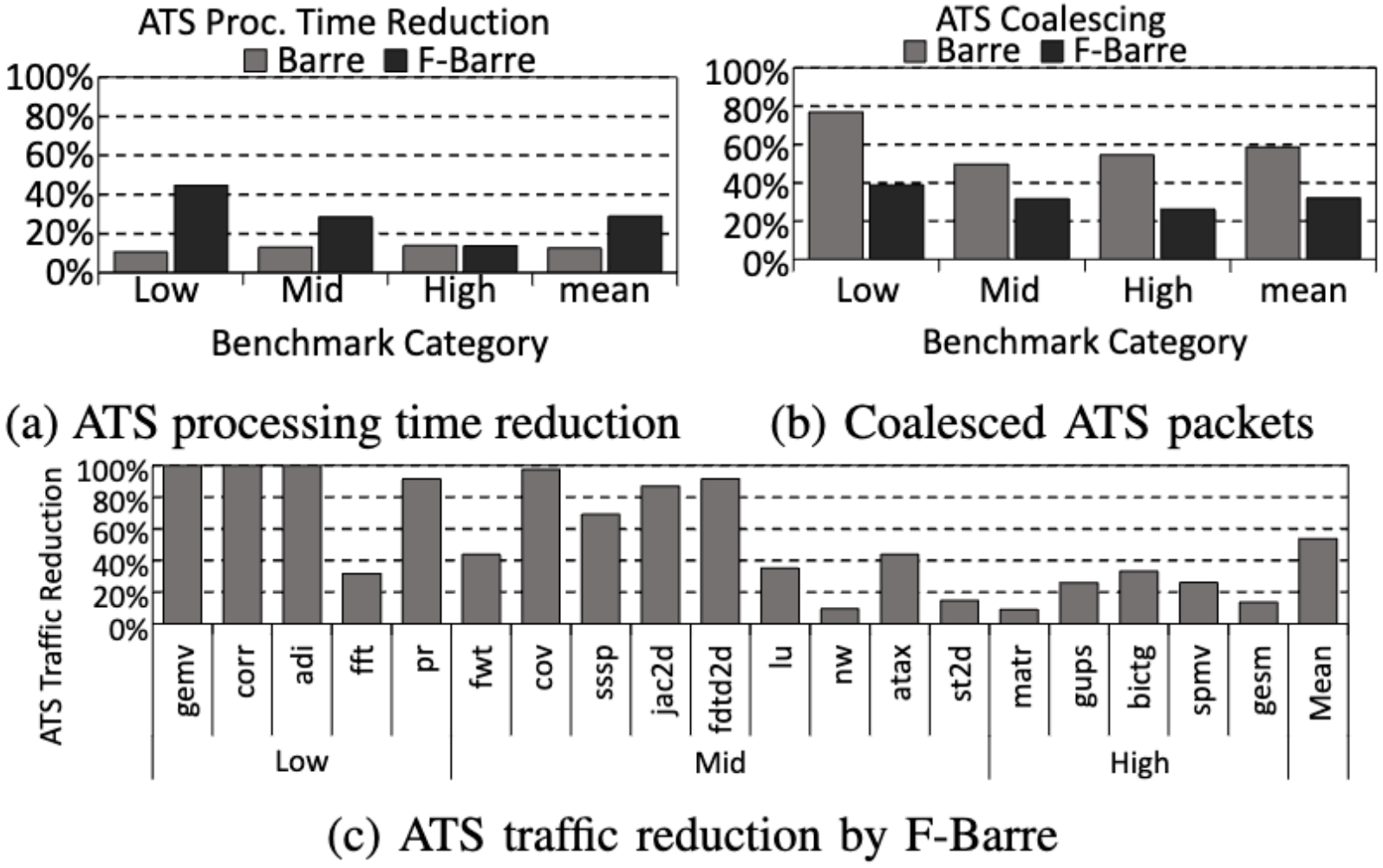
Backups

Evaluation – ATS packets evaluation

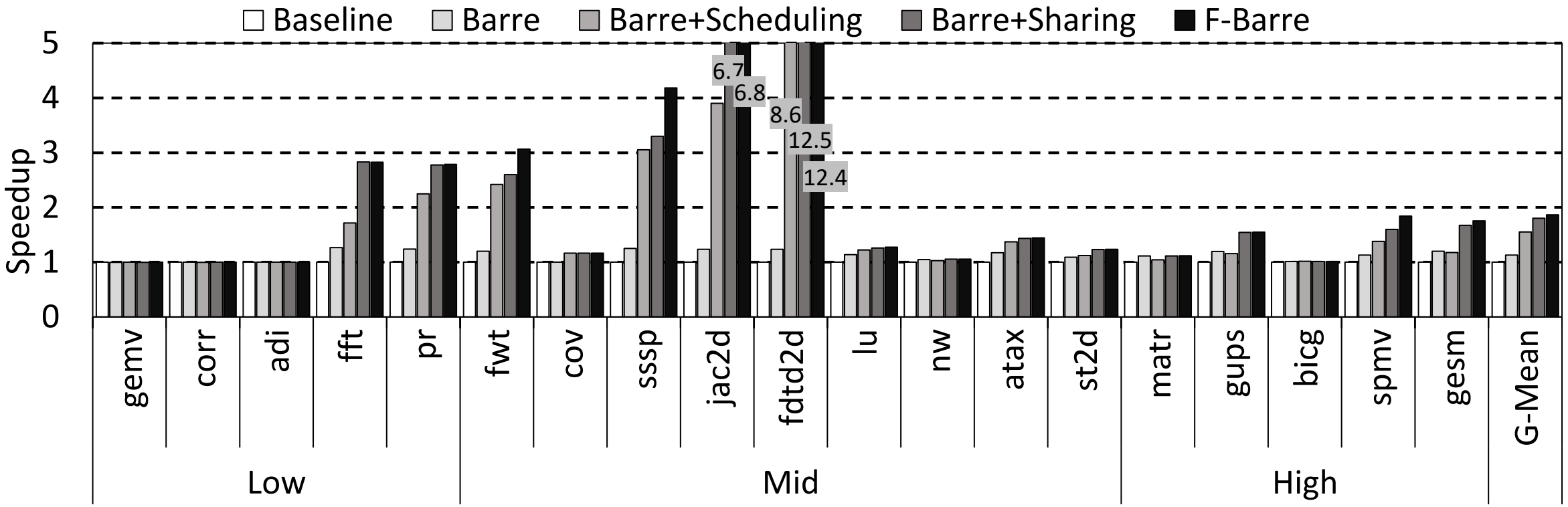
ATS processing time reduce by 12.6% and 28% by Barre and F-Barre!

58% and 32% of ATS packets are coalesced by Barre and F-Barre!

53% (up-to 99%) ATS traffic are saved by F-Barre



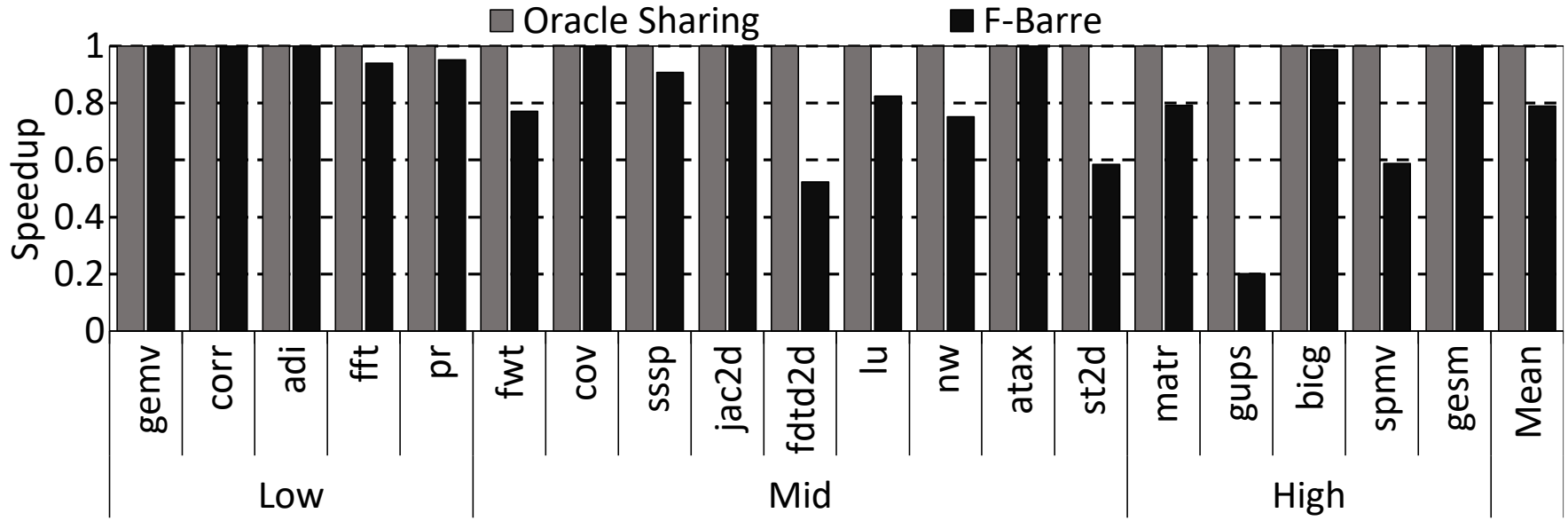
Evaluation – Breakdown



PTW scheduling provides 1.34× speedup over Barre

Intra-GPU scheduling provides 1.80× speedup over Barre

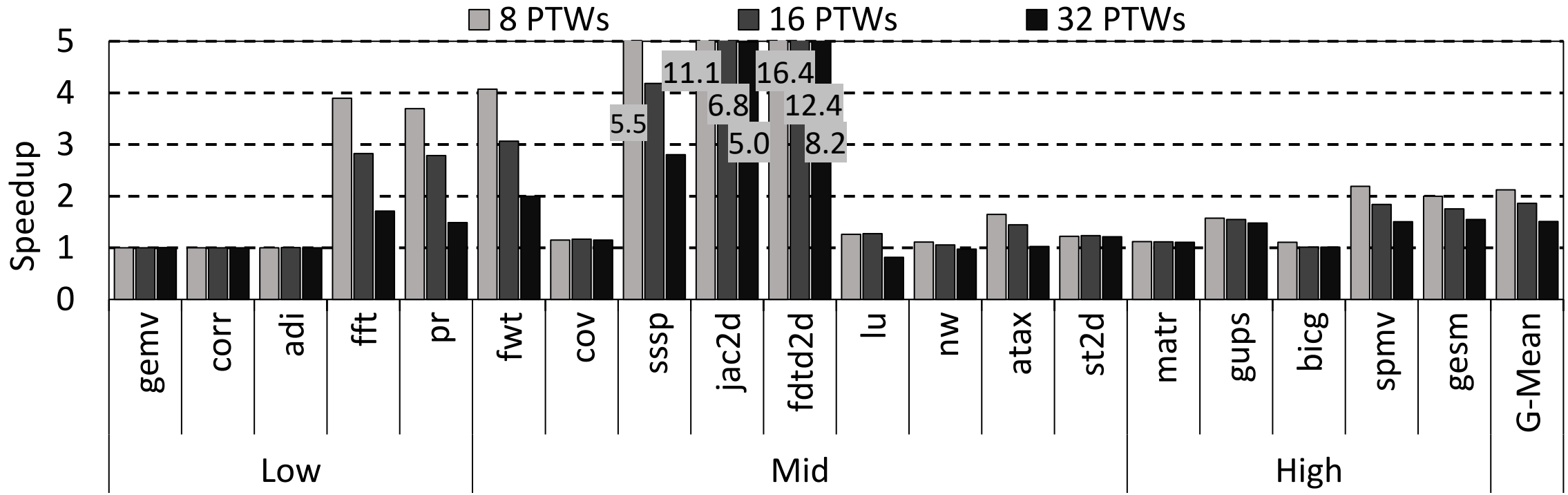
Evaluation – Traffic overhead



The oracle sharing only inject theoretical latency without bring traffic into the system.

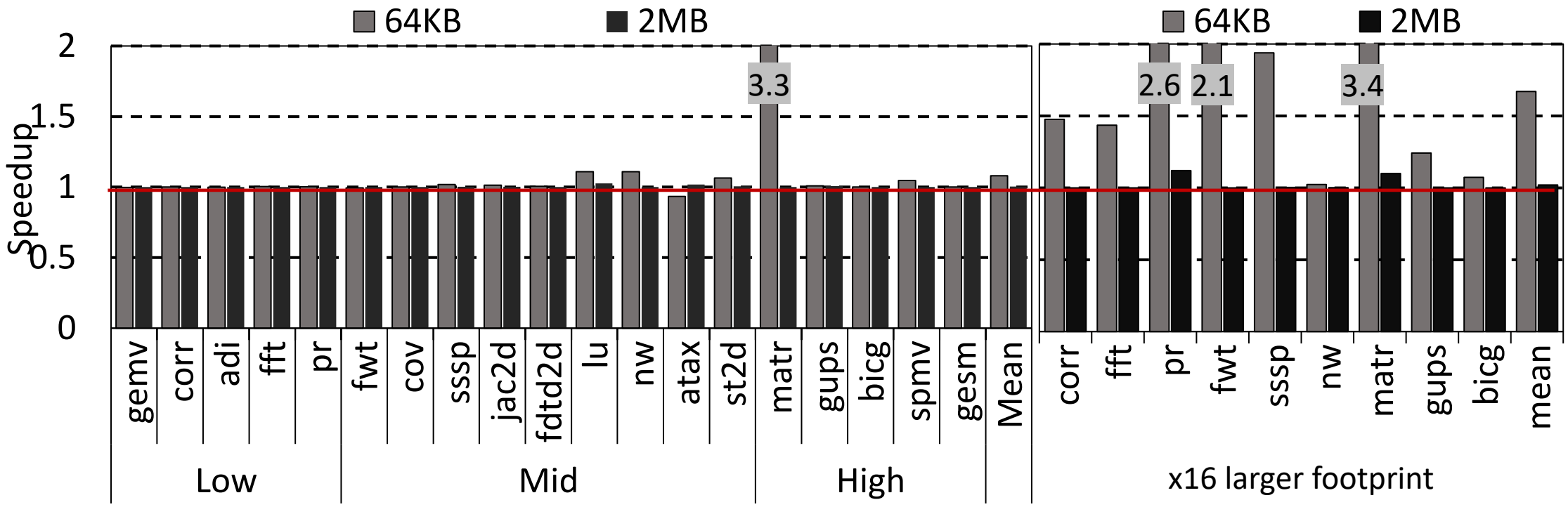
F-Barre achieve 80% of an oracle sharing scheme!

Evaluation – Num of PTW in the system



Barre Chord works even better in crowded, busy system!

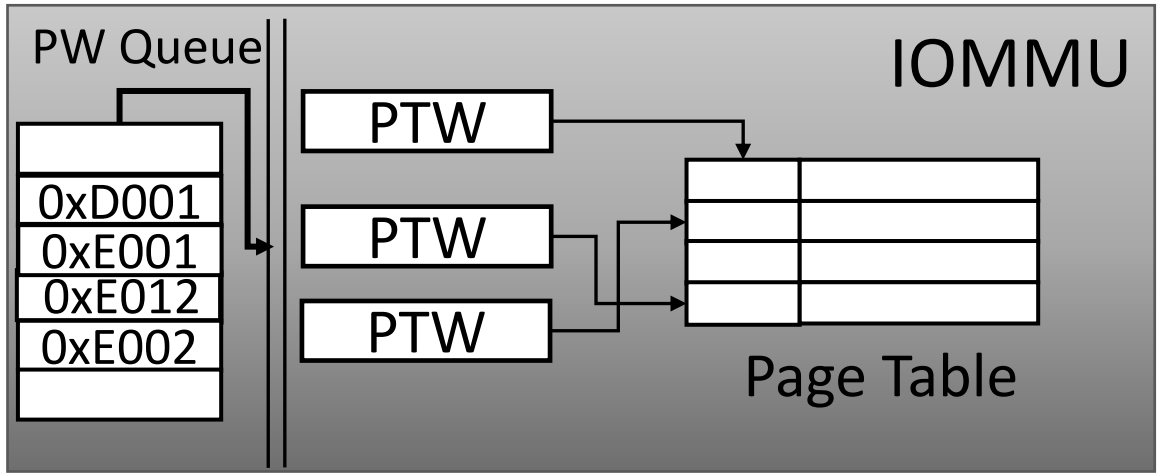
Evaluation – With super page



F-Barre – Coalescing-aware PTW scheduling

<u>Observation:</u> Opportunistically expanding coalescing group when memory allocation contiguity is observed

<u>Idea:</u> Opportunistically expanding coalescing group when memory allocation contiguity is observed



F-Barre – Do Everything in MCM-GPU!

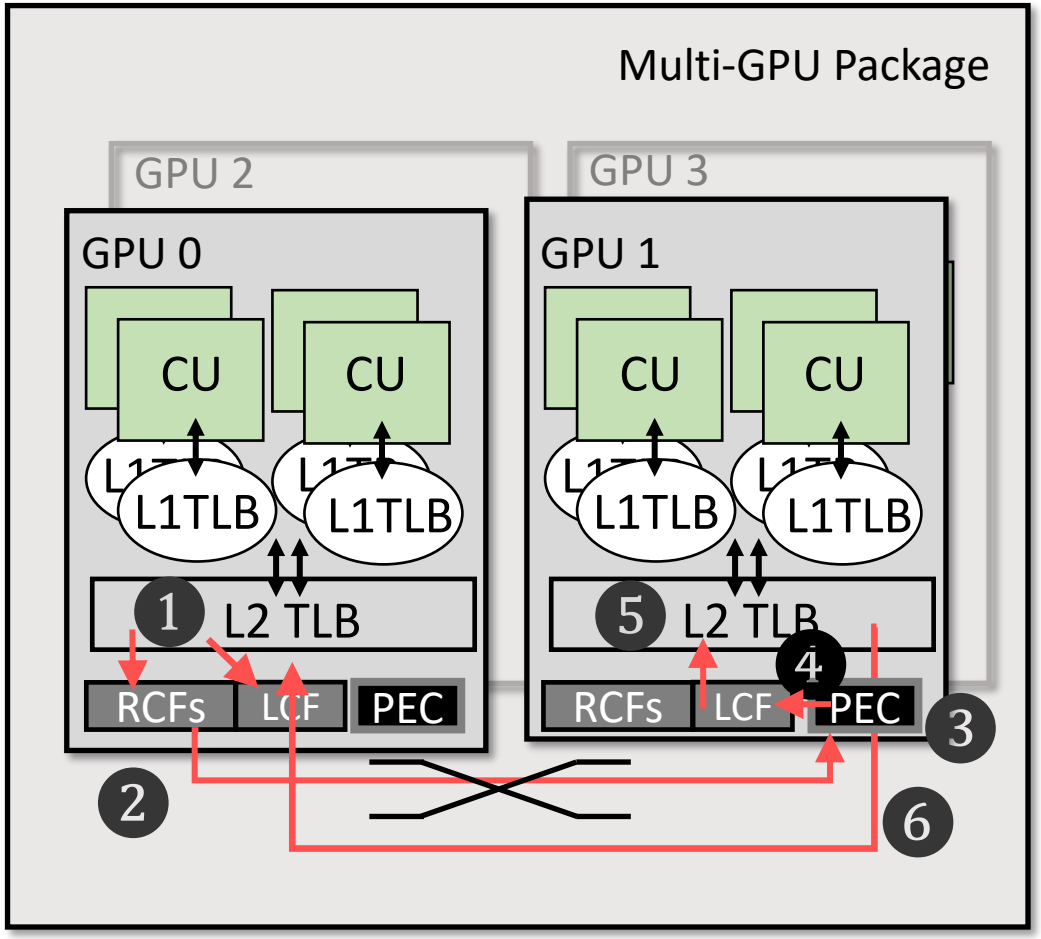
Share coalescing information among chiplets and calculate!

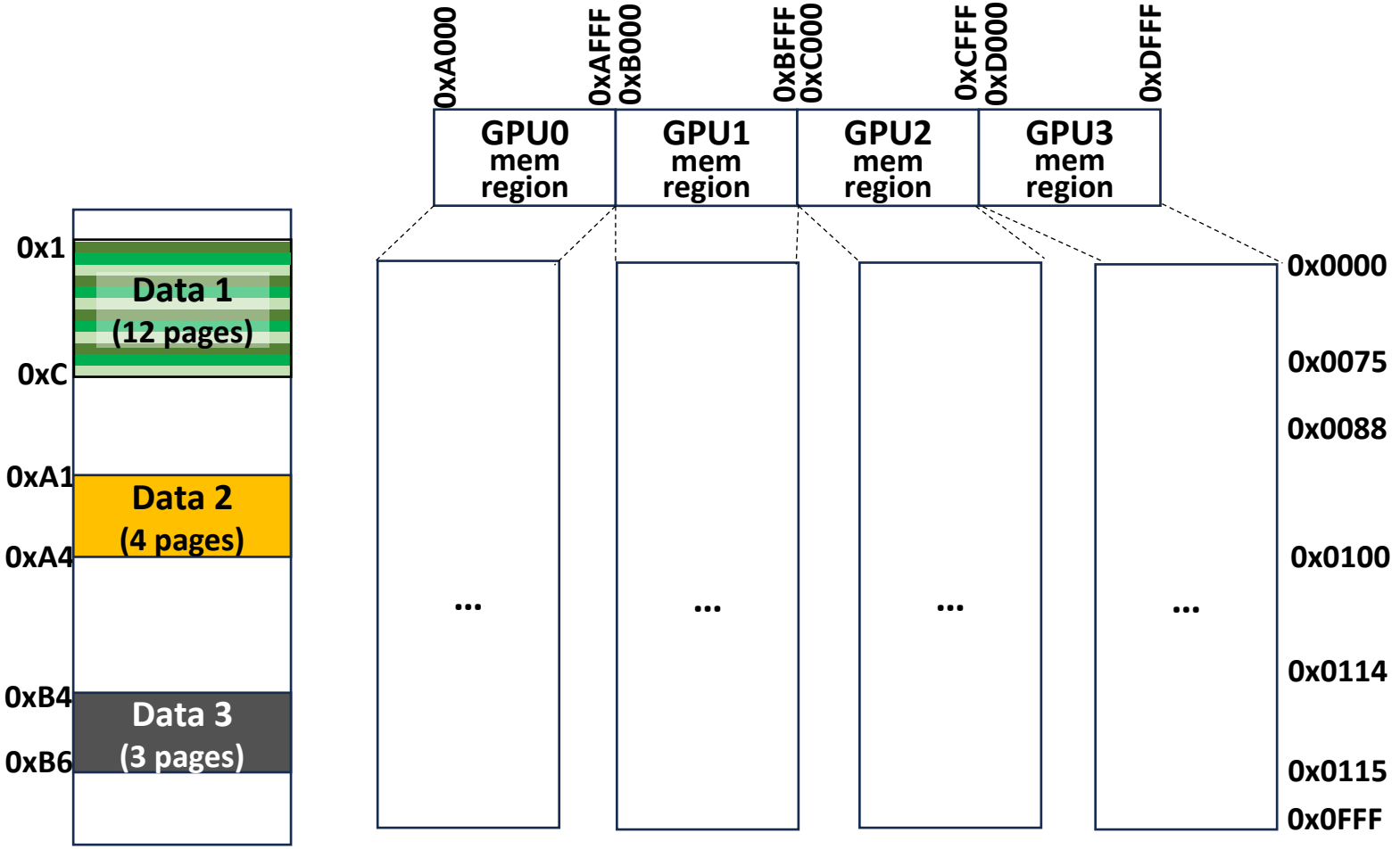
Architecture Supports:

- Remote coalescing filter (RCF)
  - 3 per Chip
  - Coalescing group containment information of remote chips
- Local coalescing filter (LCF)
  - 1 per Chip
  - Coalescing group containment information of local chips
  - Avoid exhaustive local search

Upon L1 TLB Miss – two Cuckoo filter lookups.

1. Search L2 TLB, RCFs, LCF in parallel
2. Forward translation request to peer
3. Calculate Coalescing group and VPNs
4. Use VPNs to query LCF
5. Upon LCF Hit, lookup L2 TLB
6. Return Translation

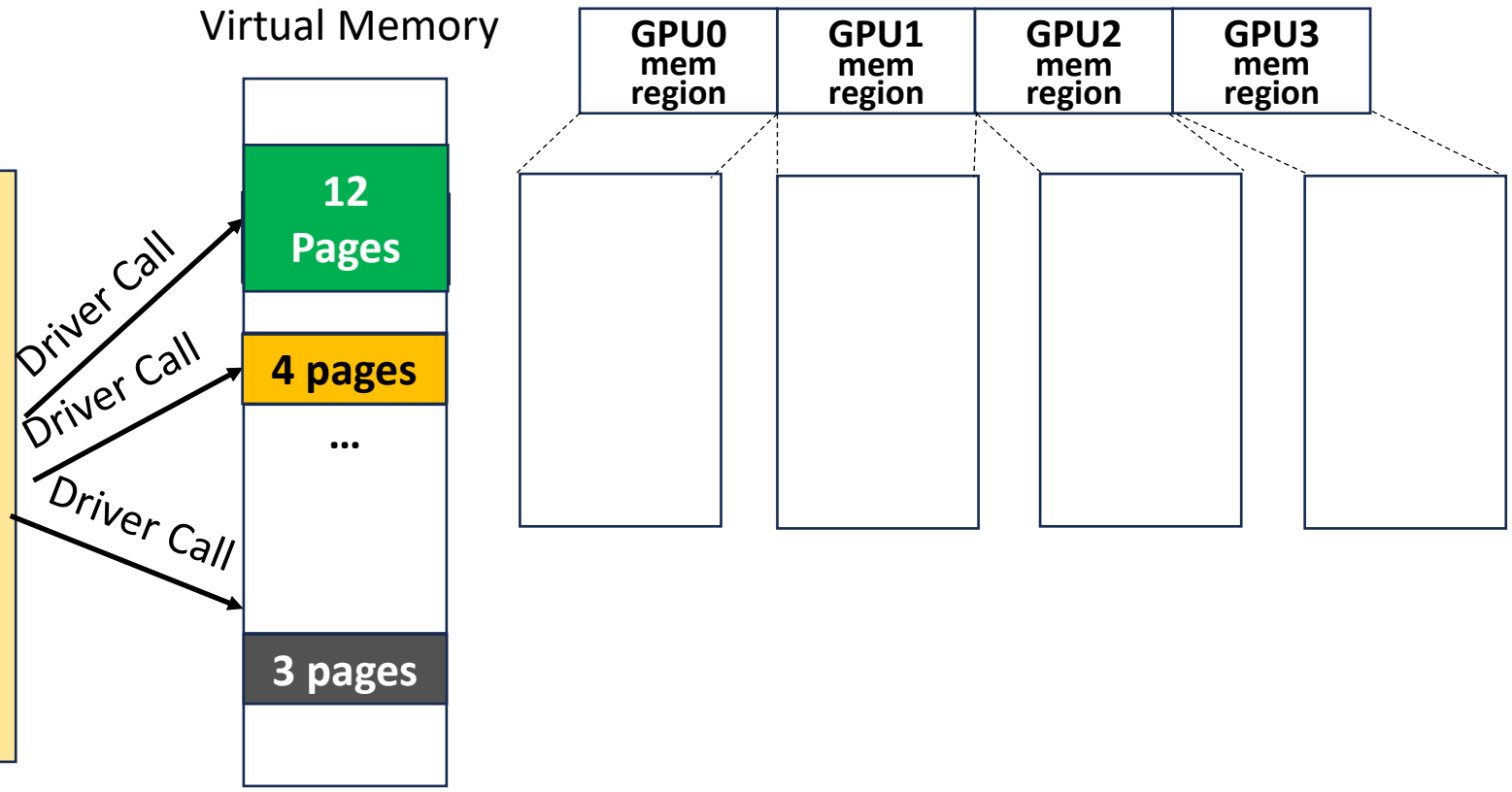



0xA000
0xAFFF
0xB000
0xBFFF
0xC000
0xCFFF
0xD000
0xDFFF
GPU0 mem region
GPU1 mem region
GPU2 mem region
GPU3 mem region
0x1
Data 1
(12 pages)
0xC
0xA1
Data 2
(4 pages)
0xA4
0xB4
Data 3
(3 pages)
0xB6
...
0x0000
0x0075
0x0088
0x0100
0x0114
0x0115
0x0FFF

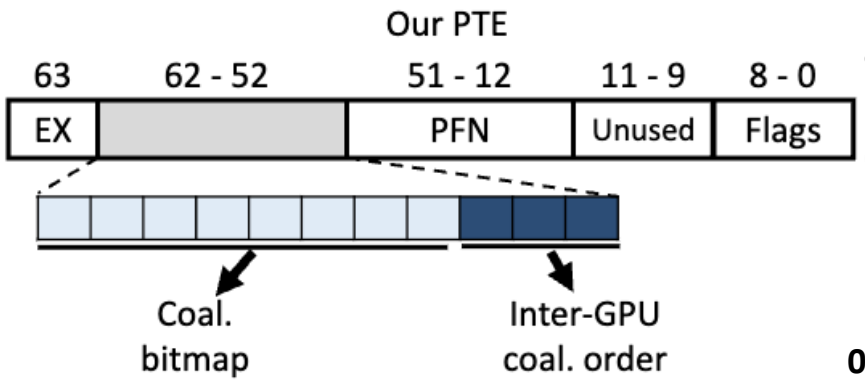
Barre – coalescing group

```
1  __kernel__ GPU_kernel() {
2     ...
3     // your favourite cuda/hip kernel
4  }
5
6  void main () {
7     GPU_UVM_Malloc(Data1, 12*PAGE_SIZE)
8     GPU_UVM_Malloc(Data2, 4*PAGE_SIZE)
9     GPU_UVM_Malloc(Data3, 3*PAGE_SIZE)
10    kernelLaunch(GPU_kernel, Data1, Data2, \\
11                          Data3);
12 return;
13 }
```

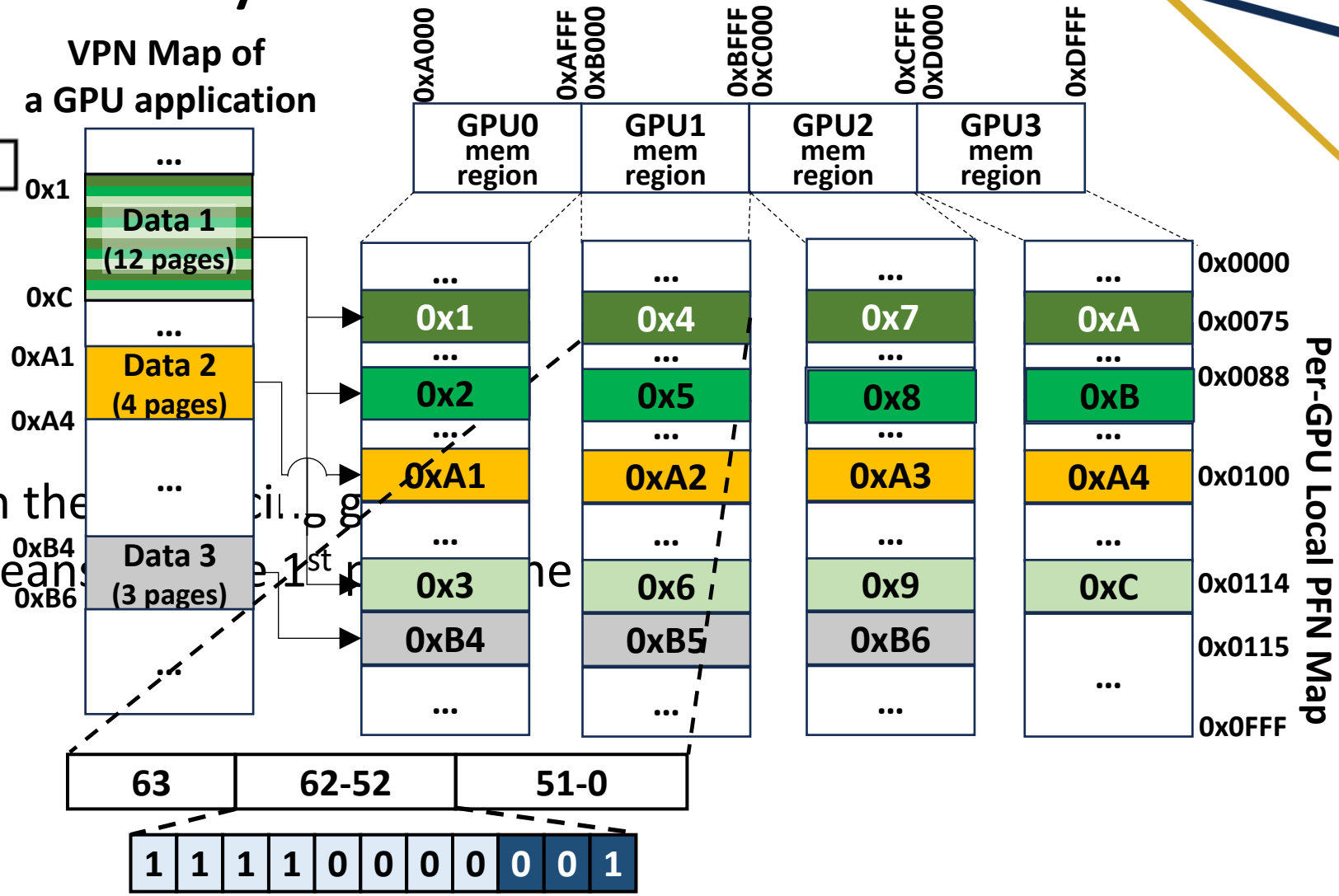
GPU program



Barre – Page Table Entry Revision



- Example PTE 0x4
  - b11110000: first 4 GPU is in the
  - b001: numerical value 1, means



F-Barre – Intra-GPU Translation

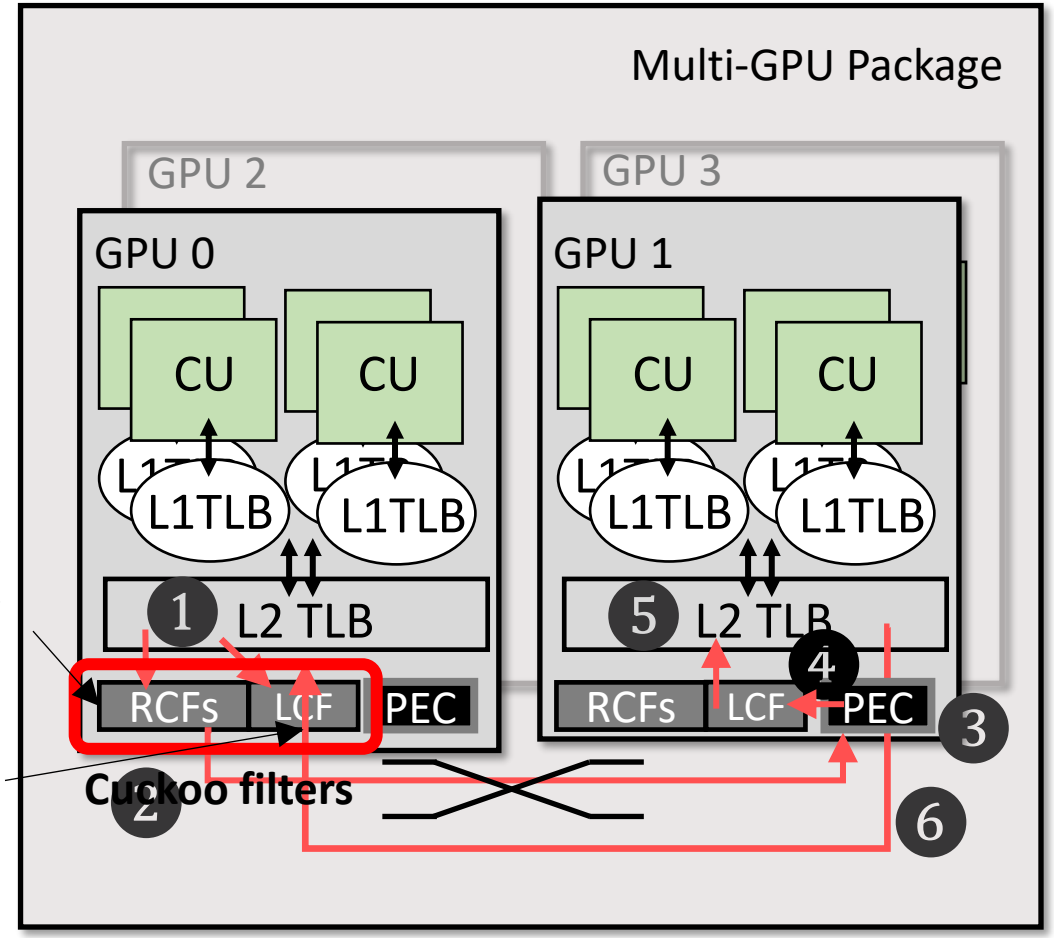
Idea: Share coalescing information among chiplets and calculate!

Upon L1 TLB Miss:

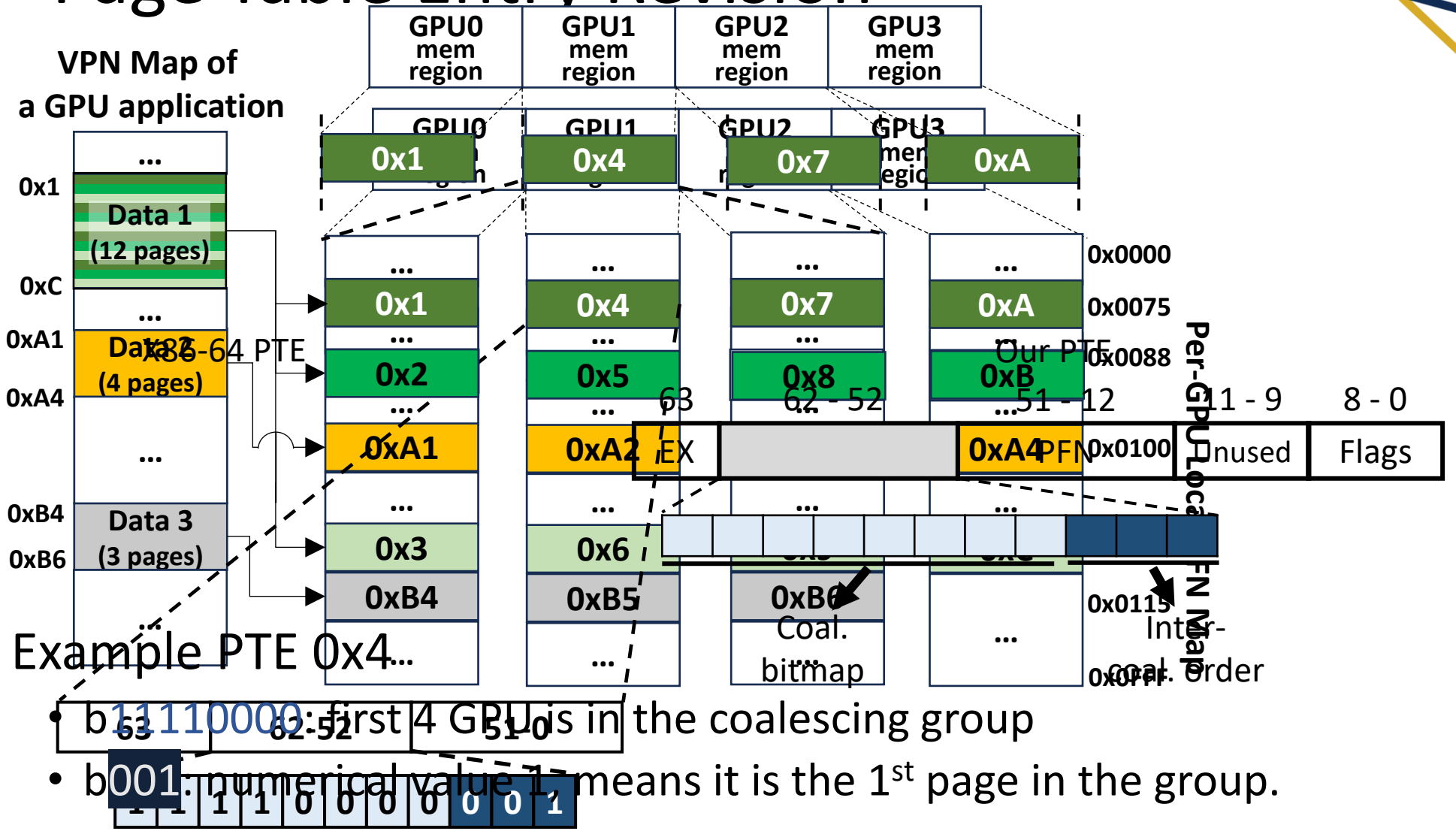
1. Search L2 TLB, RCFs, LCF in parallel
2. Forward translation request to peer
3. Calculate Coalescing group and VPNs
4. Use VPNs to query LCF
5. Upon LCF Hit, lookup L2 TLB
6. Return Translation

Which remote chiplet has the translation?

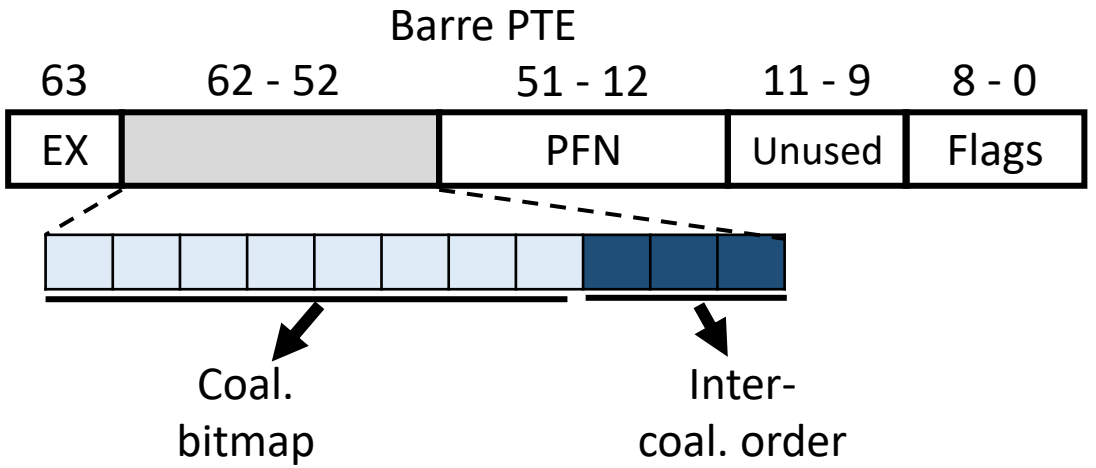
Does this chiplet have translation?



Barre – Page Table Entry Revision



- Example PTE 0x4
  - b11110000: first 4 GPU is in the coalescing group
  - b001: numerical value 1, means it is the 1st page in the group.


Barre PTE
63
62 - 52
51 - 12
11 - 9
8 - 0
EX
PFN
Unused
Flags
Coal.
bitmap
Inter-
coal. order