Mosaic: A GPU Memory Manager with Application-Transparent Support for Multiple Page Sizes

Rachata Ausavarungnirun Joshua Landgraf Vance Miller
Saugata Ghose Jayneel Gandhi
Christopher J. Rossbach Onur Mutlu

Carnegie Mellon
ETH Zürich
TEXAS
The University of Texas at Austin
vmware
SAFARI

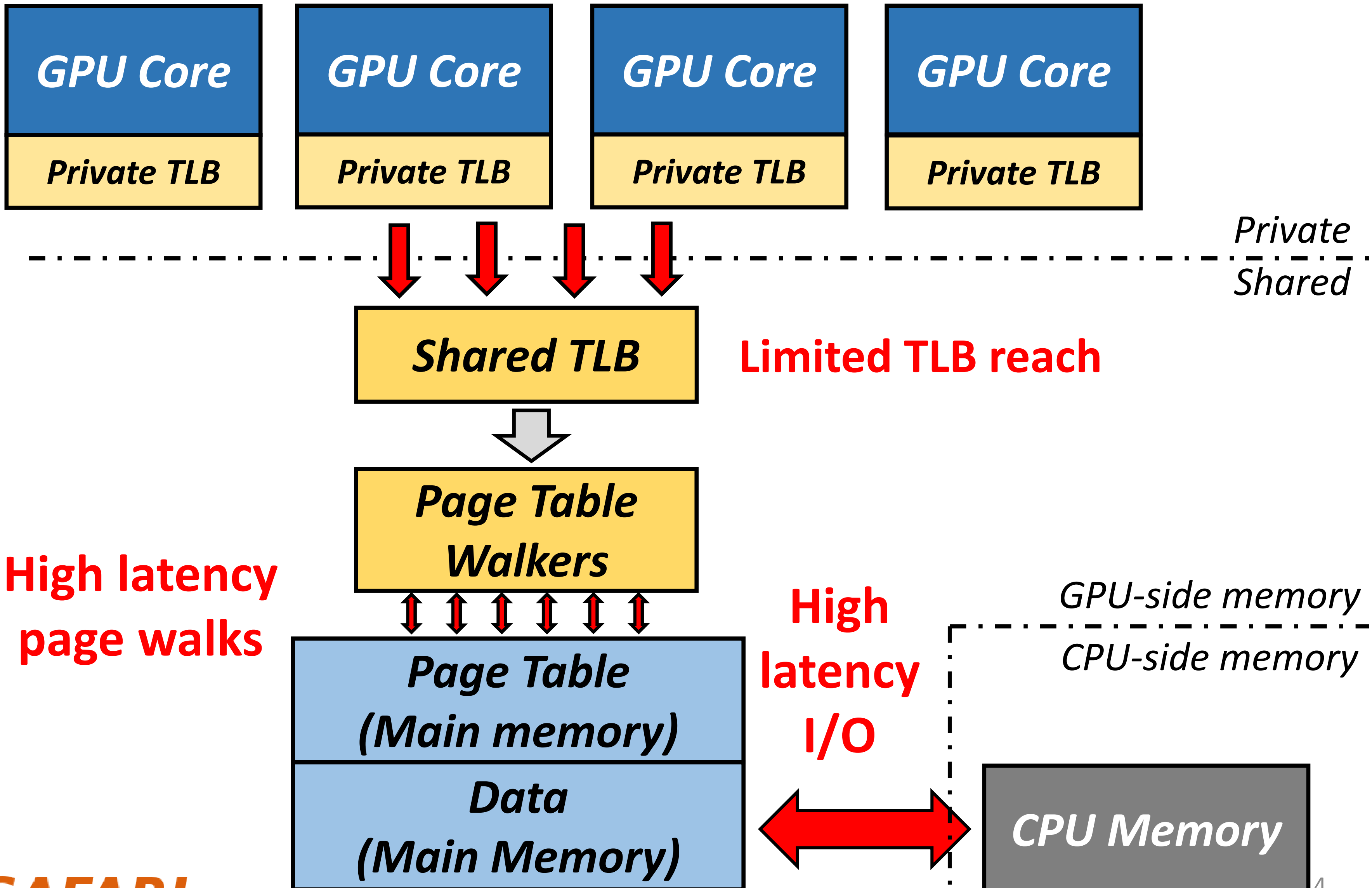
Executive Summary

- **Problem: No single best page size** for GPU virtual memory
  - **Large pages: Better TLB reach**
  - **Small pages: Lower demand paging latency**
- **Our goal: Transparently enable both page sizes**
- **Key observations**
  - Can **easily coalesce** an application's contiguously-allocated small pages into a large page
  - Interleaved memory allocation across applications **breaks page contiguity**
- **Key idea: Preserve virtual address contiguity** of small pages when allocating physical memory to simplify coalescing
- **Mosaic** is a **hardware/software cooperative framework** that:
  - Coalesces small pages into a large page without data movement
  - Enables the benefits of **both small and large pages**
- **Key result: 55% average performance improvement** over state-of-the-art GPU memory management mechanism

GPU Support for Virtual Memory

- **Improves programmability with a unified address space**
- **Enables large data sets to be processed in the GPU**
- **Allows multiple applications to run on a GPU**
  - **Virtual memory can enforce memory protection**

State-of-the-Art Virtual Memory on GPUs



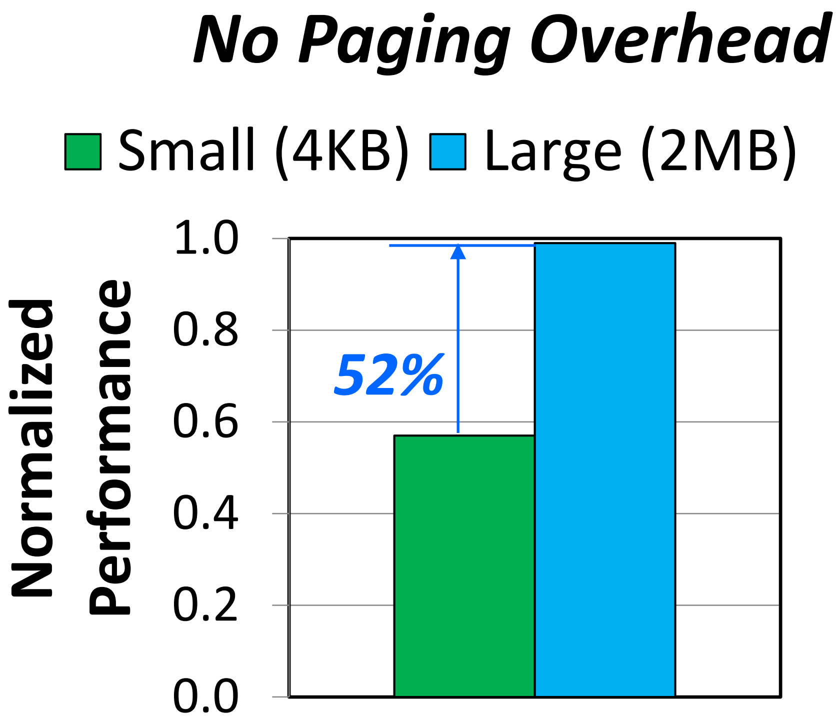
SAFARI

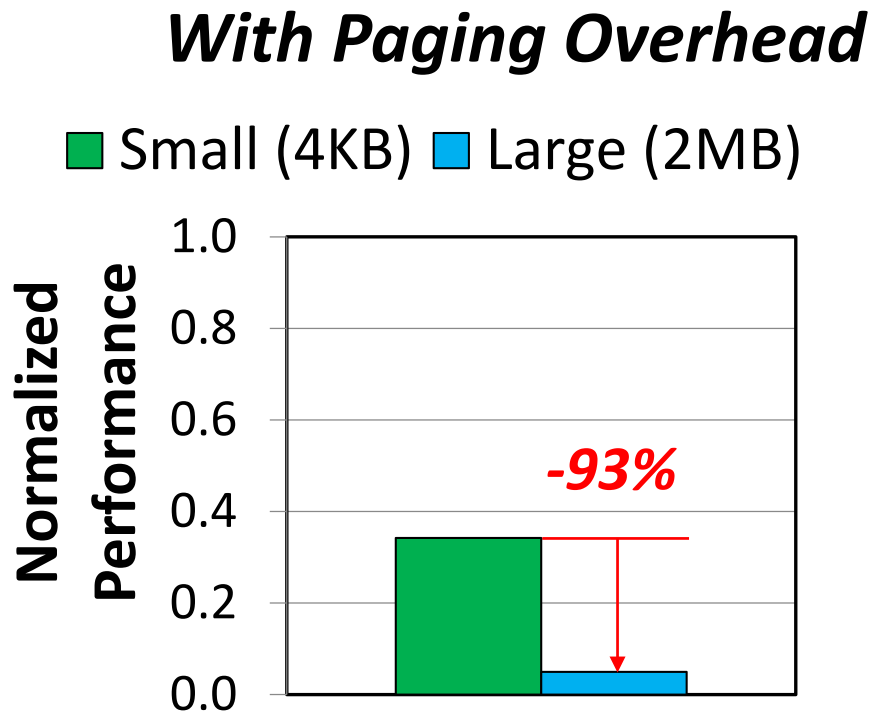
Trade-Off with Page Size

- **Larger pages:**
  - **Better TLB reach**
  - **High demand paging latency**

- **Smaller pages:**
  - **Lower demand paging latency**
  - **Limited TLB reach**

Trade-Off with Page Size





Can we get the best of both page sizes?

Outline

- Background
- **Key challenges and our goal**
- Mosaic
- Experimental evaluations
- Conclusions

Challenges with Multiple Page Sizes

Time

App 1 Allocation

App 2 Allocation

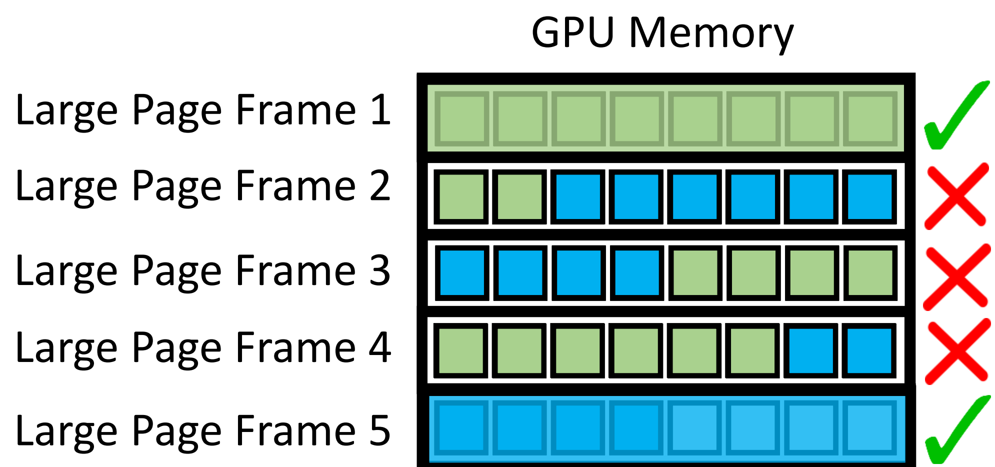
App 1 Allocation

App 2 Allocation

Coalesce App 1 Pages

Coalesce App 2 Pages

State-of-the-Art



Cannot coalesce (without migrating multiple 4K pages)

Need to search which pages to coalesce

App 1 **App 2**

Desirable Allocation

Desirable Behavior

Time

App 1 Allocation

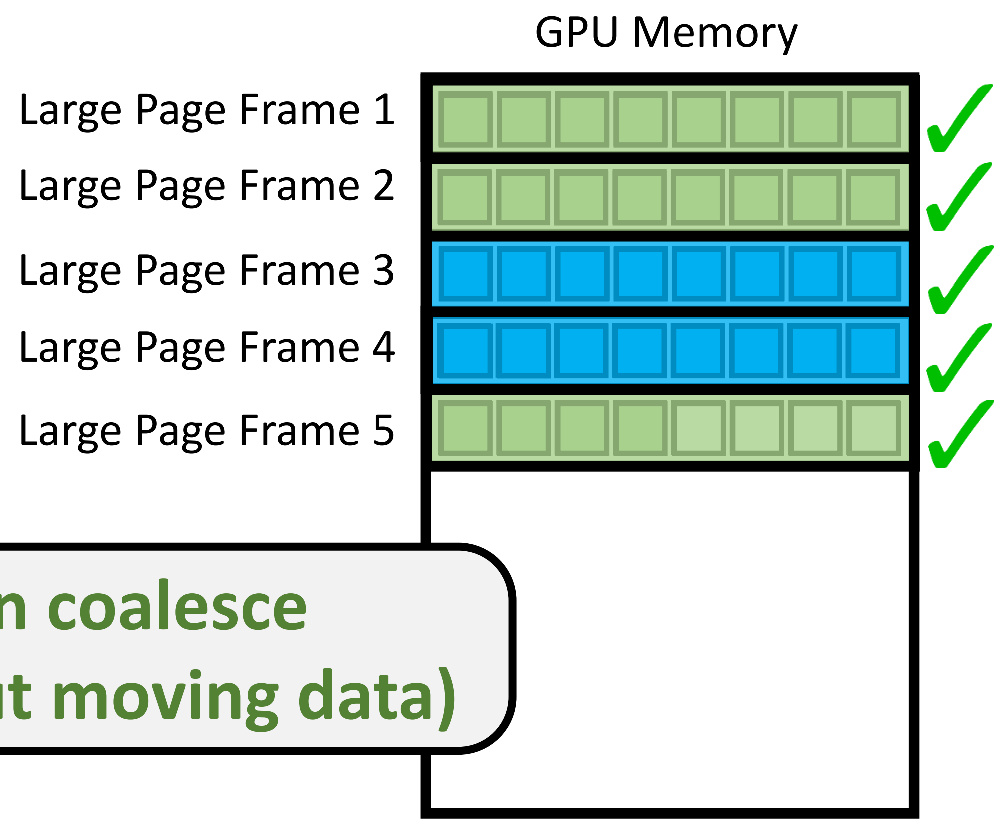
App 2 Allocation

App 1 Allocation

App 2 Allocation

Coalesce App 1 Pages

Coalesce App 2 Pages



Can coalesce (without moving data)

Unallocated **App 1** **App 2**

SAFARI

Our Goals

- **High TLB reach**
- **Low demand paging latency**
- **Application transparency**
  - Programmers **do not need to modify the applications**

Outline

- Background
- Key challenges and our goal
- **Mosaic**
- Experimental evaluation
- Conclusions

Mosaic

GPU Runtime

Contiguity-Conserving Allocation

In-Place Coalescer

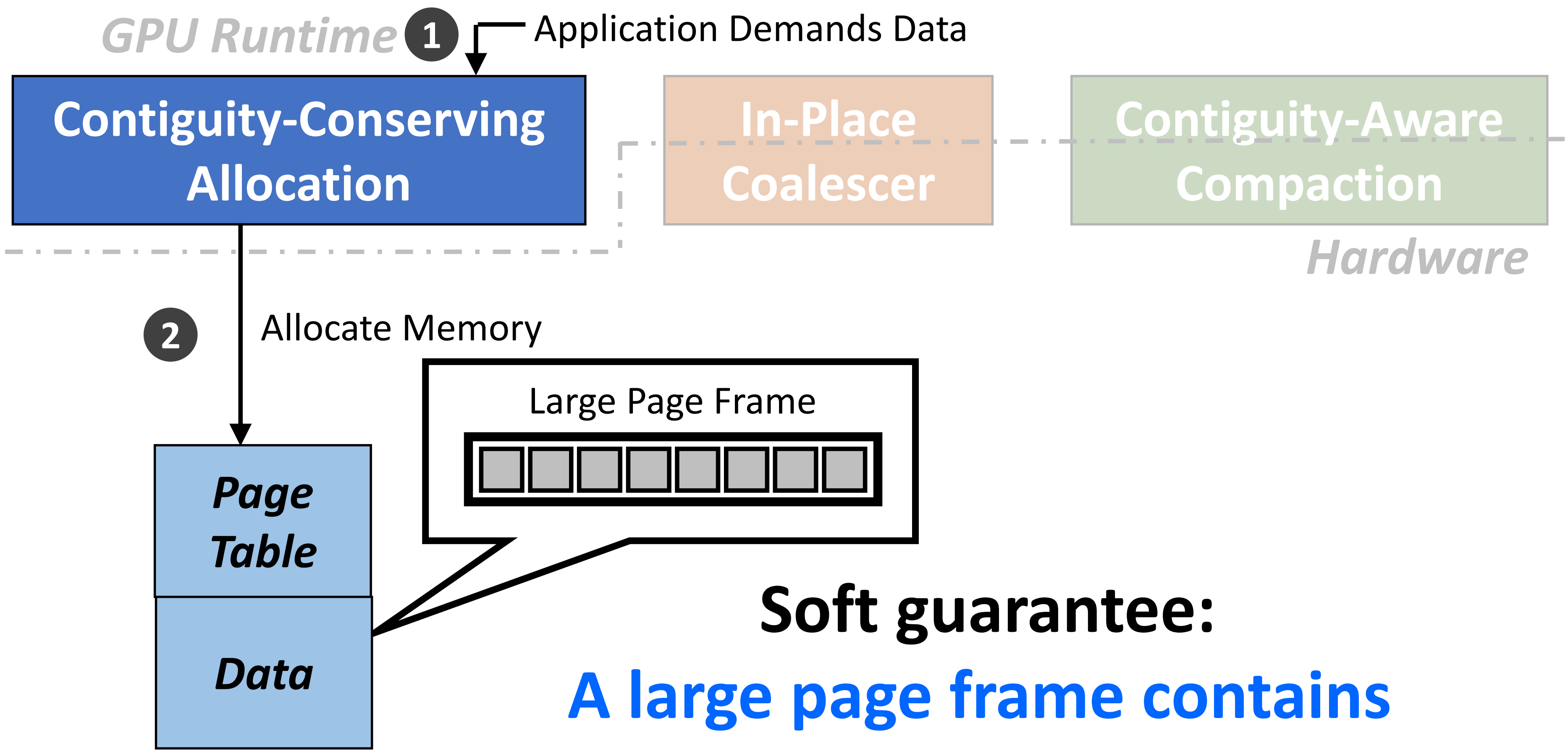
Contiguity-Aware Compaction

Hardware

Outline

- Background
- Key challenges and our goal
- **Mosaic**
  - **Contiguity-Conserving Allocation**
  - In-Place Coalescer
  - Contiguity-Aware Compaction
- Experimental evaluations
- Conclusions

Mosaic: Data Allocation

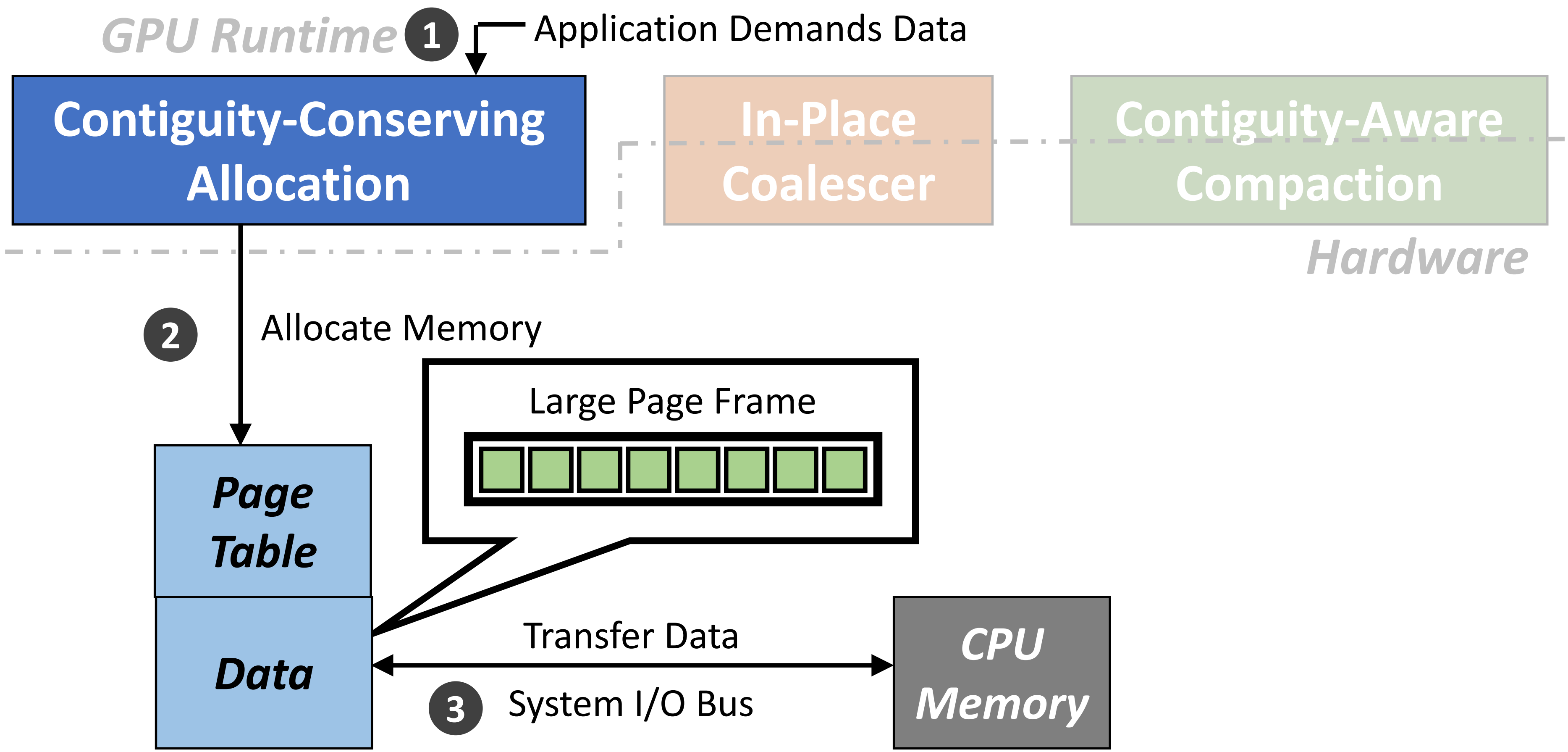


Soft guarantee:

A large page frame contains pages from only a single address space

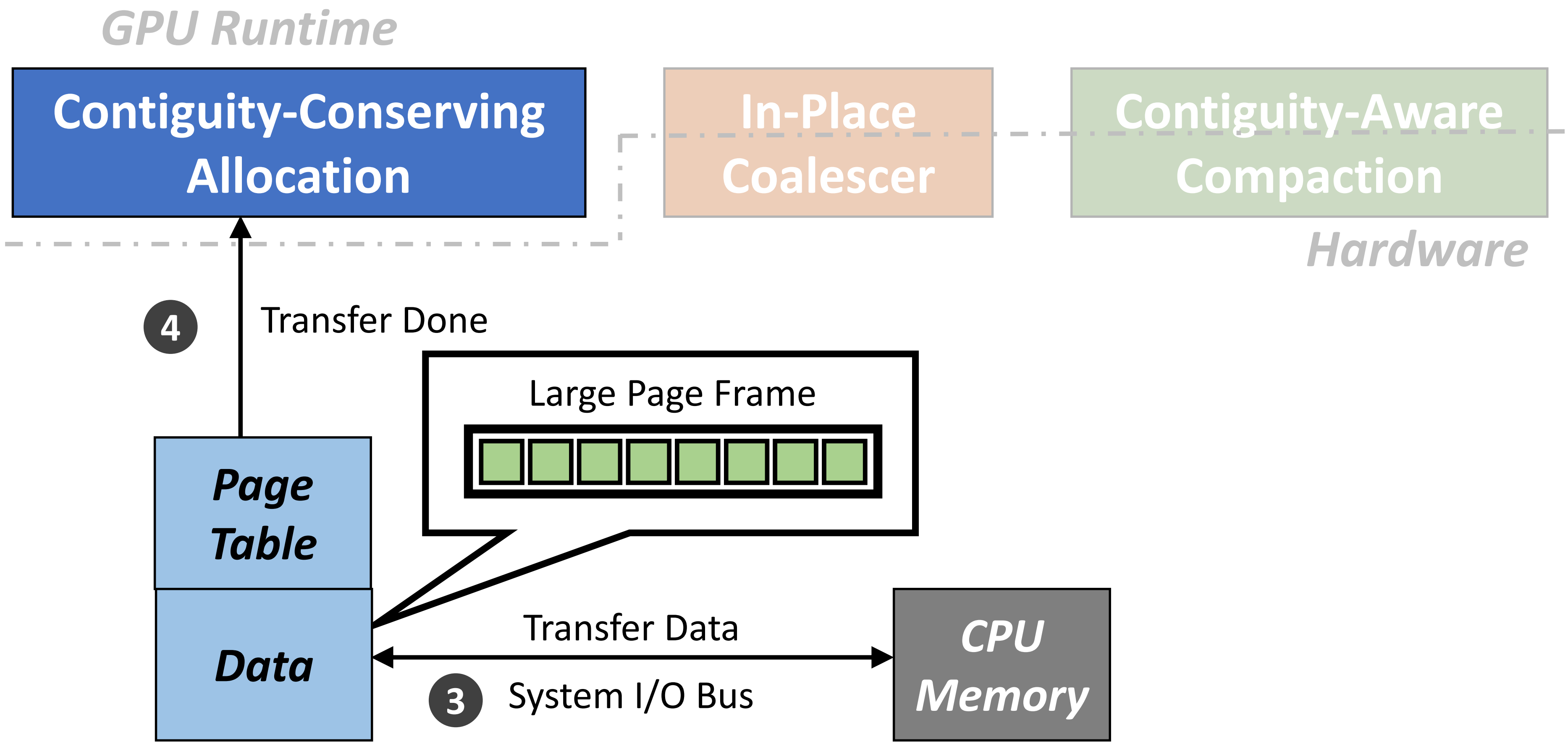
Conserves contiguity within the large page frame

Mosaic: Data Allocation



- **Data transfer is done at a small page granularity**
  - **A page that is transferred is immediately ready to use**

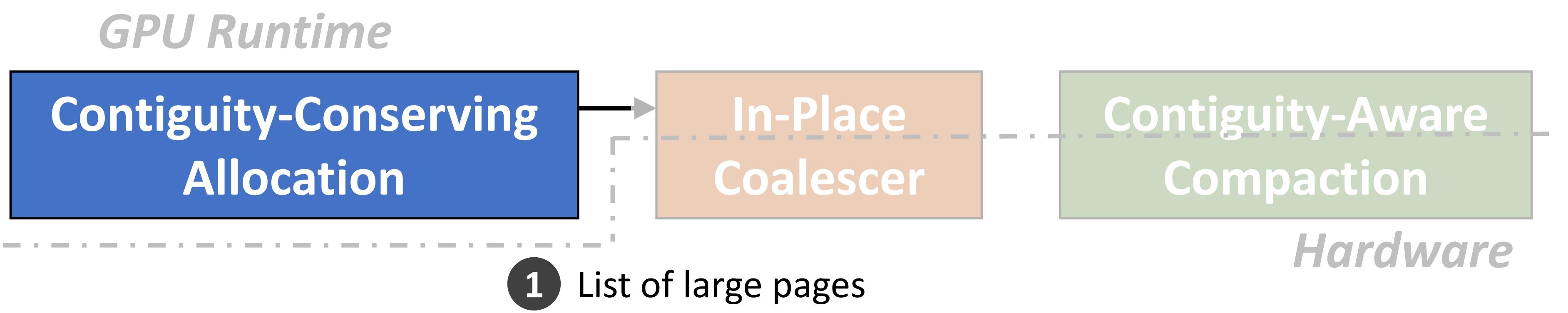
Mosaic: Data Allocation



Outline

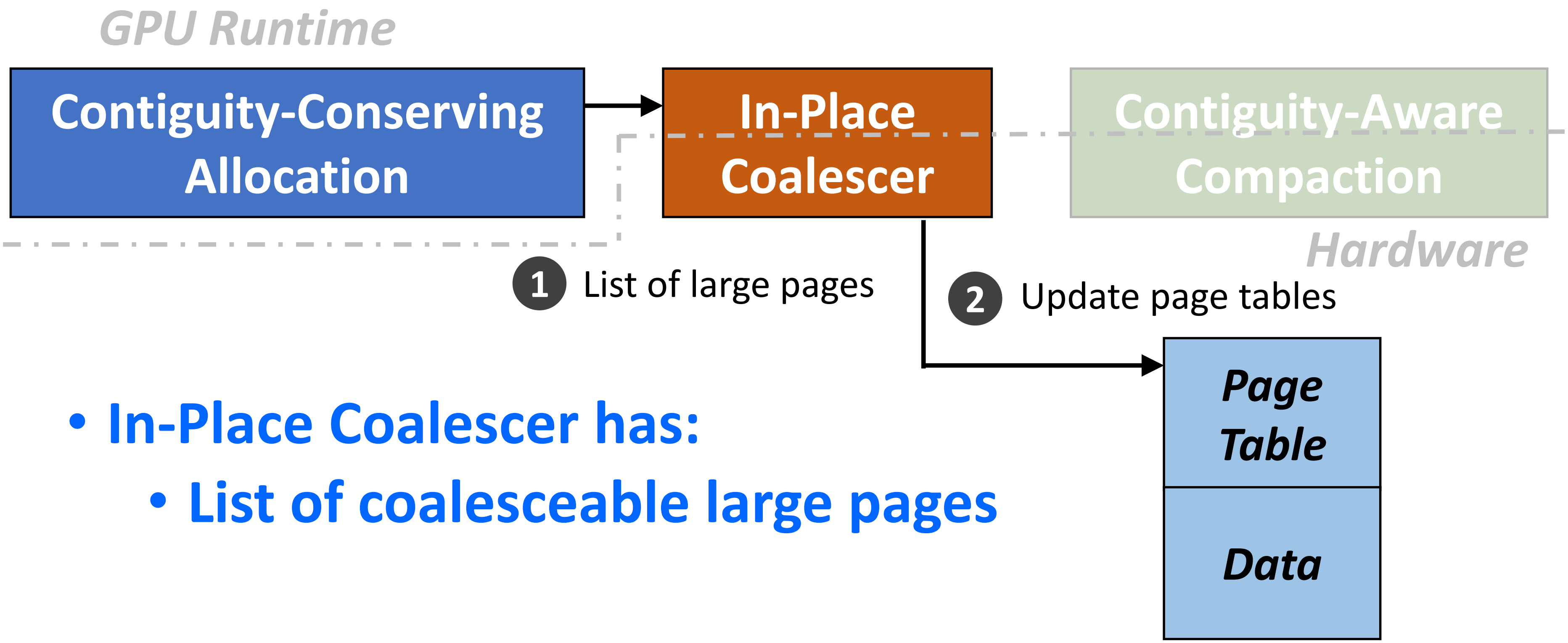
- Background
- Key challenges and our goal
- **Mosaic**
  - Contiguity-Conserving Allocation
  - **In-Place Coalescer**
  - Contiguity-Aware Compaction
- Experimental evaluations
- Conclusions

Mosaic: Coalescing



- Fully-allocated large page frame → Coalesceable
- Allocator sends the list of coalesceable pages to the In-Place Coalescer

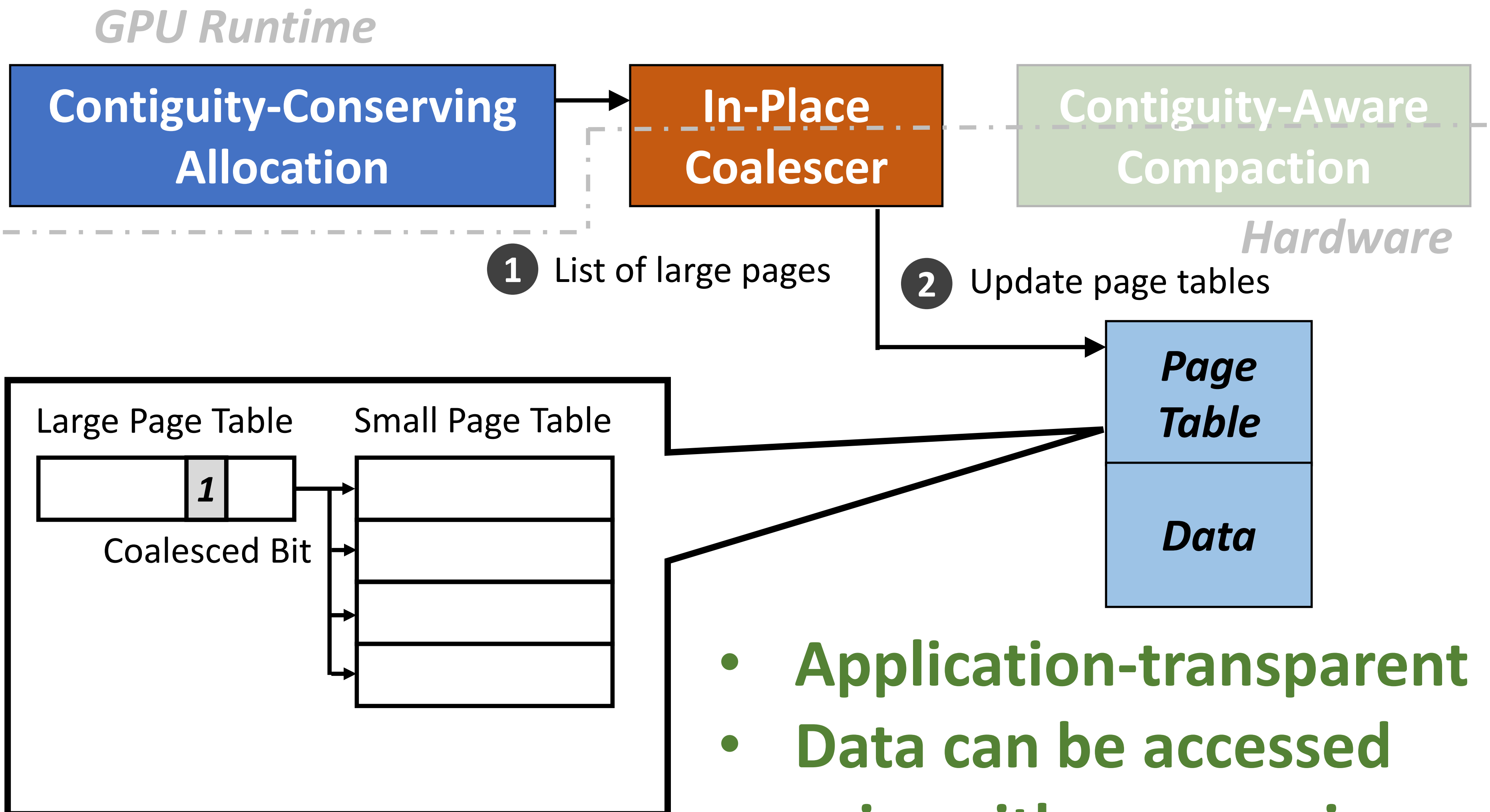
Mosaic: Coalescing



- **In-Place Coalescer has:**
  - **List of coalesceable large pages**

- **Key Task: Perform coalescing without moving data**
  - **Simply need to update the page tables**

Mosaic: Coalescing

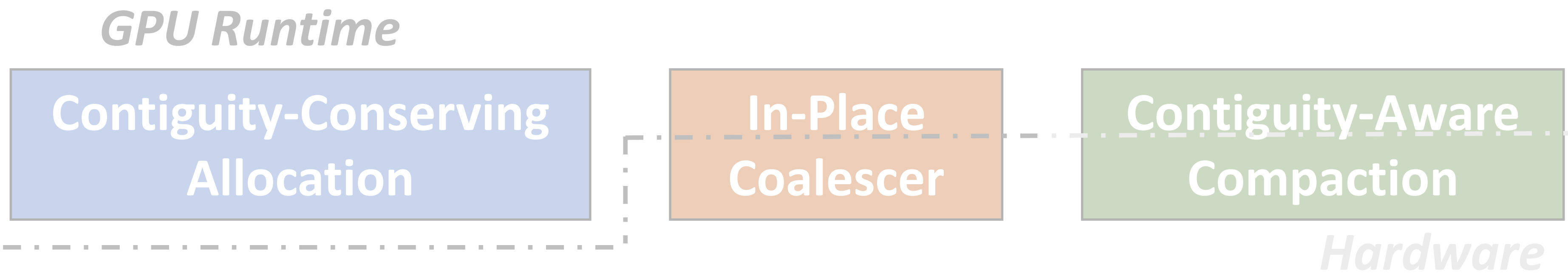


- **Application-transparent**
- **Data can be accessed using either page size**
- **No TLB flush**

Outline

- Background
- Key challenges and our goal
- **Mosaic**
  - Contiguity-Conserving Allocation
  - In-Place Coalescer
  - **Contiguity-Aware Compaction**
- Experimental evaluations
- Conclusions

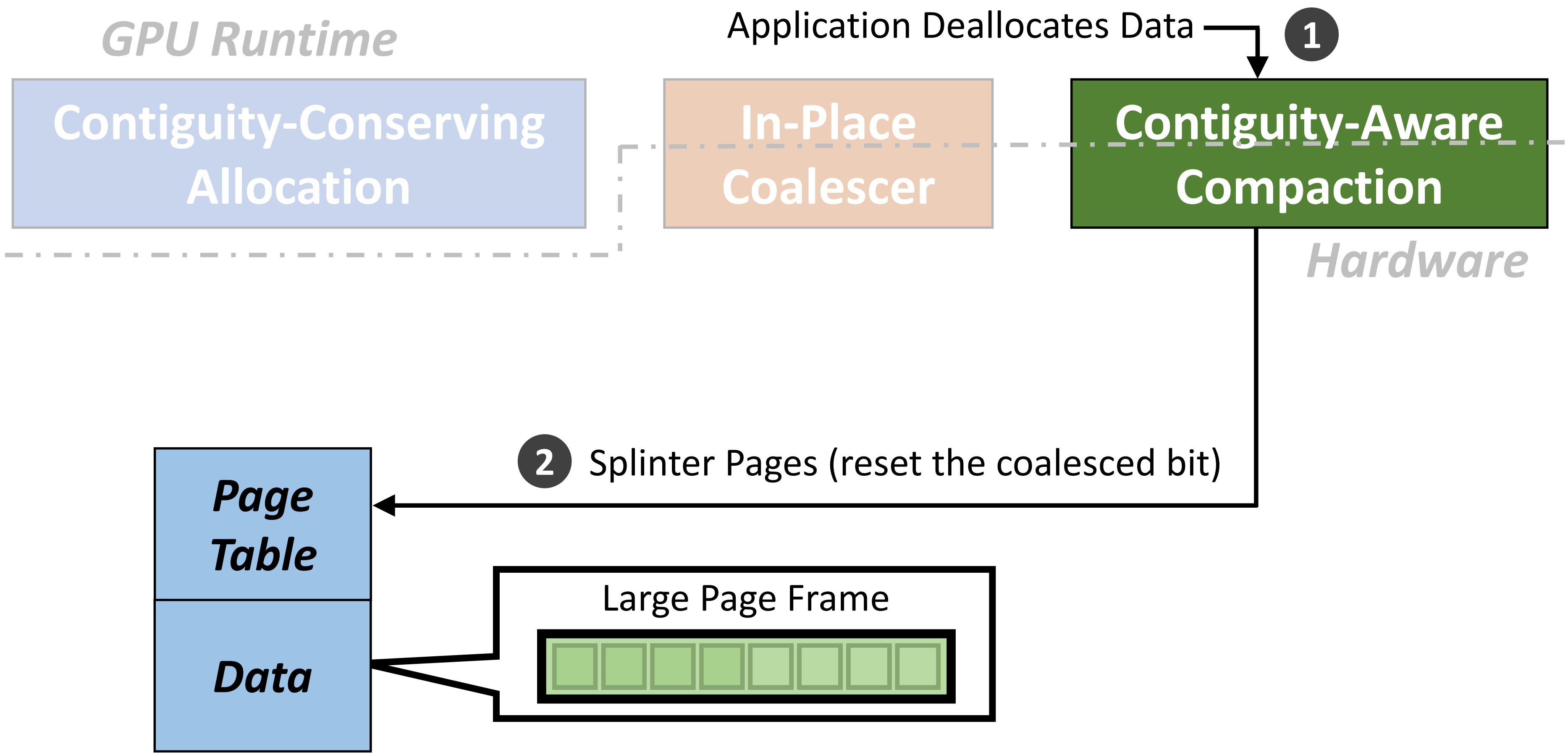
Mosaic: Data Deallocation



- **Key Task:** Free up not-fully-used large page frames
  - Splinter pages → **Break down a large page** into small pages
  - Compaction → **Combine fragmented large page frames**

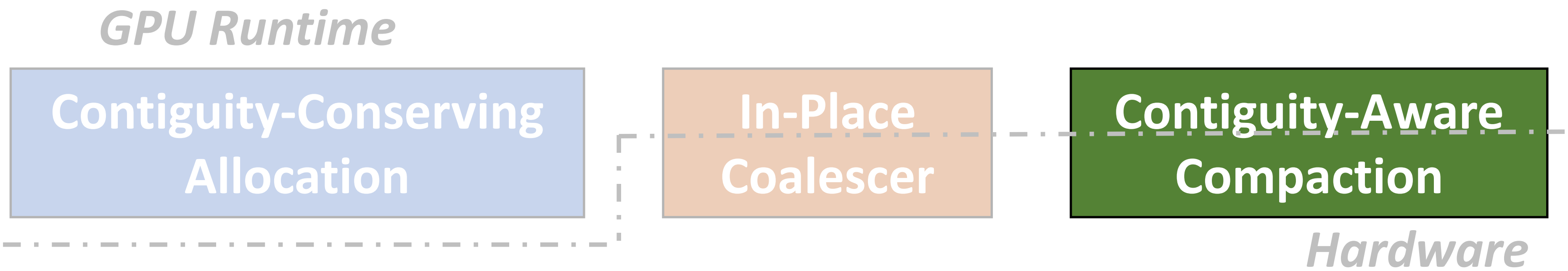
SAFARI

Mosaic: Data Deallocation



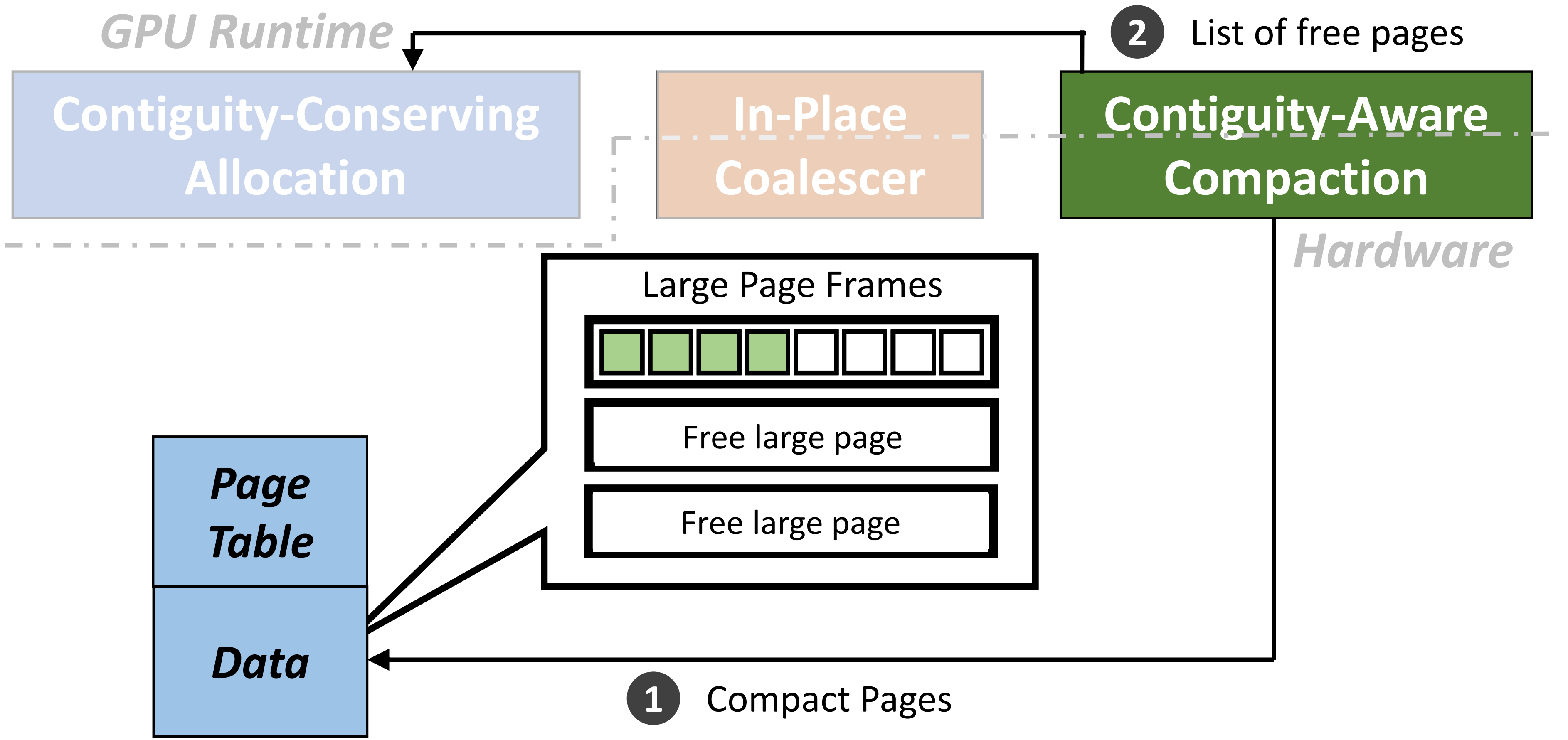
- **Splinter only frames with deallocated pages**

Mosaic: Compaction



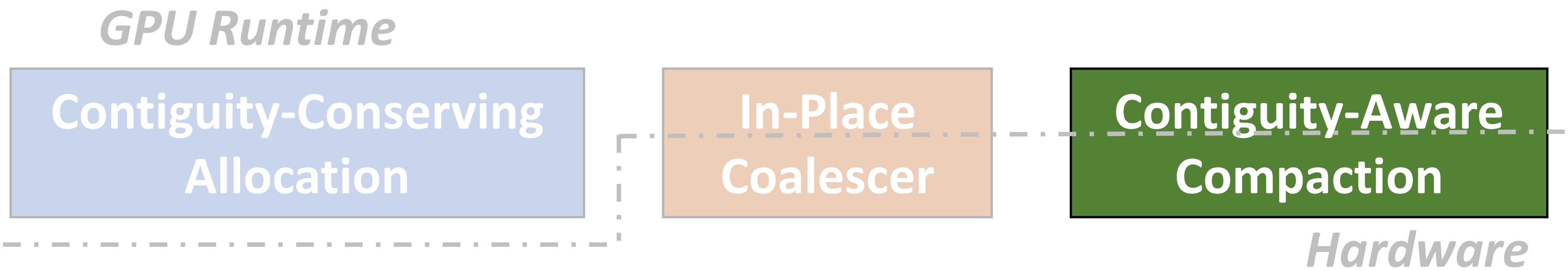
- **Key Task:** Free up not-fully-used large page frames
  - Splinter pages → **Break down a large page** into small pages
  - Compaction → **Combine fragmented large page frames**

Mosaic: Compaction



- **Compaction decreases memory bloat**
  - Happens only when memory is highly fragmented

Mosaic: Compaction



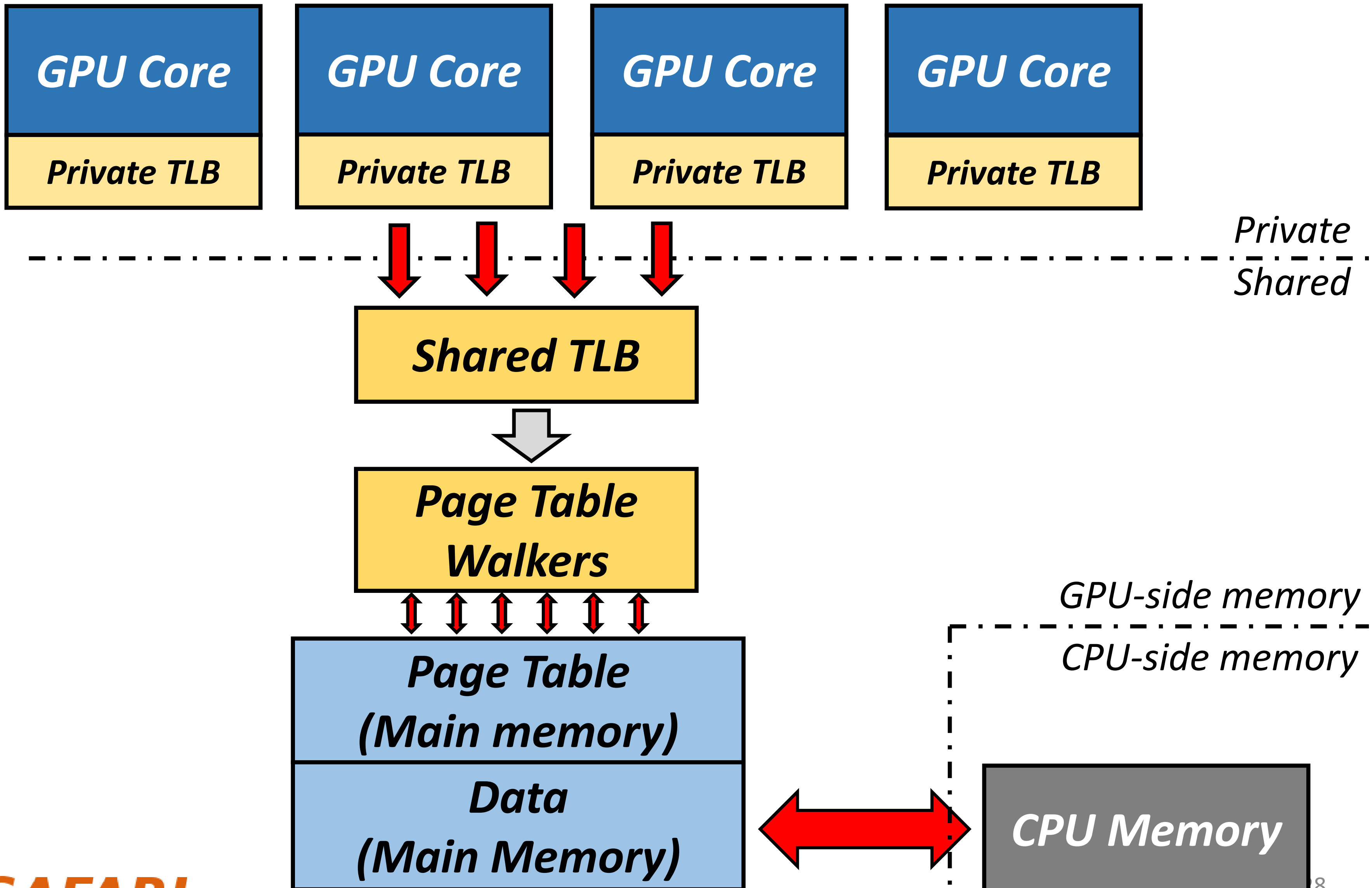
- Once pages are compacted, **they become non-coalesceable**
  - No virtual contiguity

- **Maximizes number of free large page frames**

Outline

- Background
- Key challenges and our goal
- Mosaic
  - Contiguity-Conserving Allocation
  - In-Place Coalescer
  - Contiguity-Aware Compaction
- **Experimental evaluations**
- Conclusions

Baseline: State-of-the-Art GPU Virtual Memory



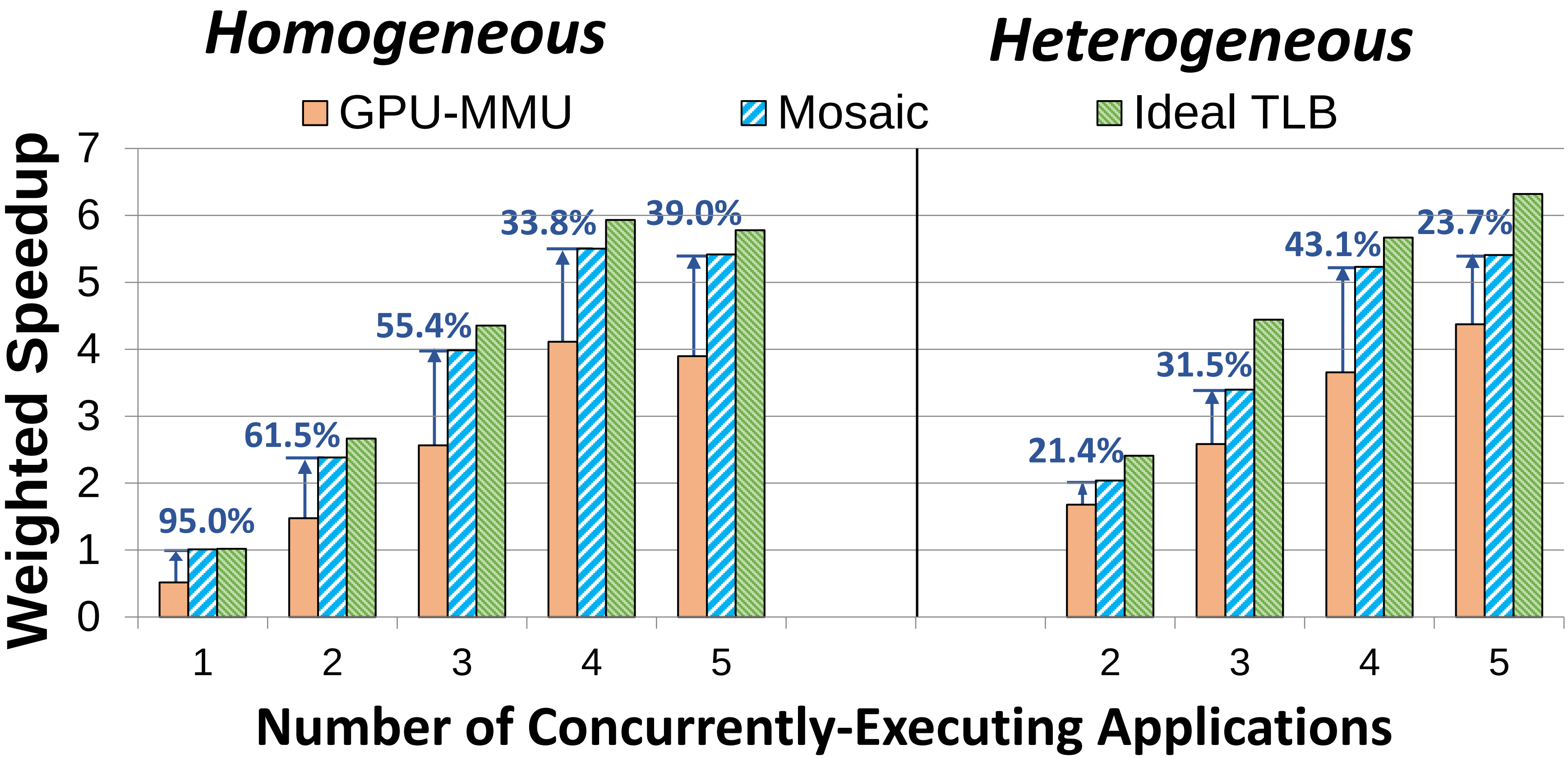
Methodology

- GPGPU-Sim (MAFIA) modeling GTX750 Ti
  - 30 GPU cores
  - Multiple GPGPU applications execute concurrently
  - 64KB 4-way L1, 2048KB 16-way L2
  - 64-entry L1 TLB, 1024-entry L2 TLB
  - 8-entry large page L1 TLB, 64-entry large page L2 TLB
  - 3GB main memory
- Model sequential page walks
- Model page tables and virtual-to-physical mapping
- CUDA-SDK, Rodinia, Parboil, LULESH, SHOC suites
  - 235 total workloads evaluated
- Available at: **https://github.com/CMU-SAFARI/Mosaic**

Comparison Points

- State-of-the-art CPU-GPU memory management
  - GPU-MMU based on [Power et al., HPCA'14]
  - **Upside:** **Utilizes parallel page walks, TLB request coalescing and page walk cache to improve performance**
  - **Downside:** **Limited TLB reach**
- Ideal TLB: Every TLB access is an L1 TLB hit

Performance



Mosaic consistently improves performance across a wide variety of workloads

Mosaic performs within 10% of the ideal TLB

Other Results in the Paper

- TLB hit rate
  - Mosaic achieves average **TLB hit rate of 99%**
- Per-application IPC
  - 97% of all **applications perform faster**
- Sensitivity to different TLB sizes
  - Mosaic is **effective for various TLB configurations**
- Memory fragmentation analysis
  - Mosaic **reduces memory fragmentation** and **improves performance** regardless of the original fragmentation
- Performance with and without demand paging

Outline

- Background
- Key challenges and our goal
- Mosaic
  - Contiguity-Conserving Allocation
  - In-Place Coalescer
  - Contiguity-Aware Compaction
- Experimental evaluations
- **Conclusions**

Summary

- **Problem: No single best page size** for GPU virtual memory
  - **Large pages: Better TLB reach**
  - **Small pages: Lower demand paging latency**
- **Our goal: Transparently enable both page sizes**
- **Key observations**
  - Can **easily coalesce** an application's contiguously-allocated small pages into a large page
  - Interleaved memory allocation across applications **breaks page contiguity**
- **Key idea: Preserve virtual address contiguity** of small pages when allocating physical memory to simplify coalescing
- **Mosaic** is a **hardware/software cooperative framework** that:
  - Coalesces small pages into a large page without data movement
  - Enables the benefits of **both small and large pages**
- **Key result: 55% average performance improvement** over state-of-the-art GPU memory management mechanism

Mosaic: A GPU Memory Manager with Application-Transparent Support for Multiple Page Sizes

Rachata Ausavarungnirun Joshua Landgraf Vance Miller
Saugata Ghose Jayneel Gandhi
Christopher J. Rossbach Onur Mutlu

Carnegie Mellon
ETH Zürich
TEXAS The University of Texas at Austin
vmware®

SAFARI

Backup Slides

Current Methods to Share GPUs

- **Time sharing**
  - **Fine-grained context switching**
  - **Coarse-grained context switching**

- **Spatial sharing**
  - **NVIDIA GRID**
  - **Multi process service**

Other Methods to Enforce Protection

- **Segmented paging**

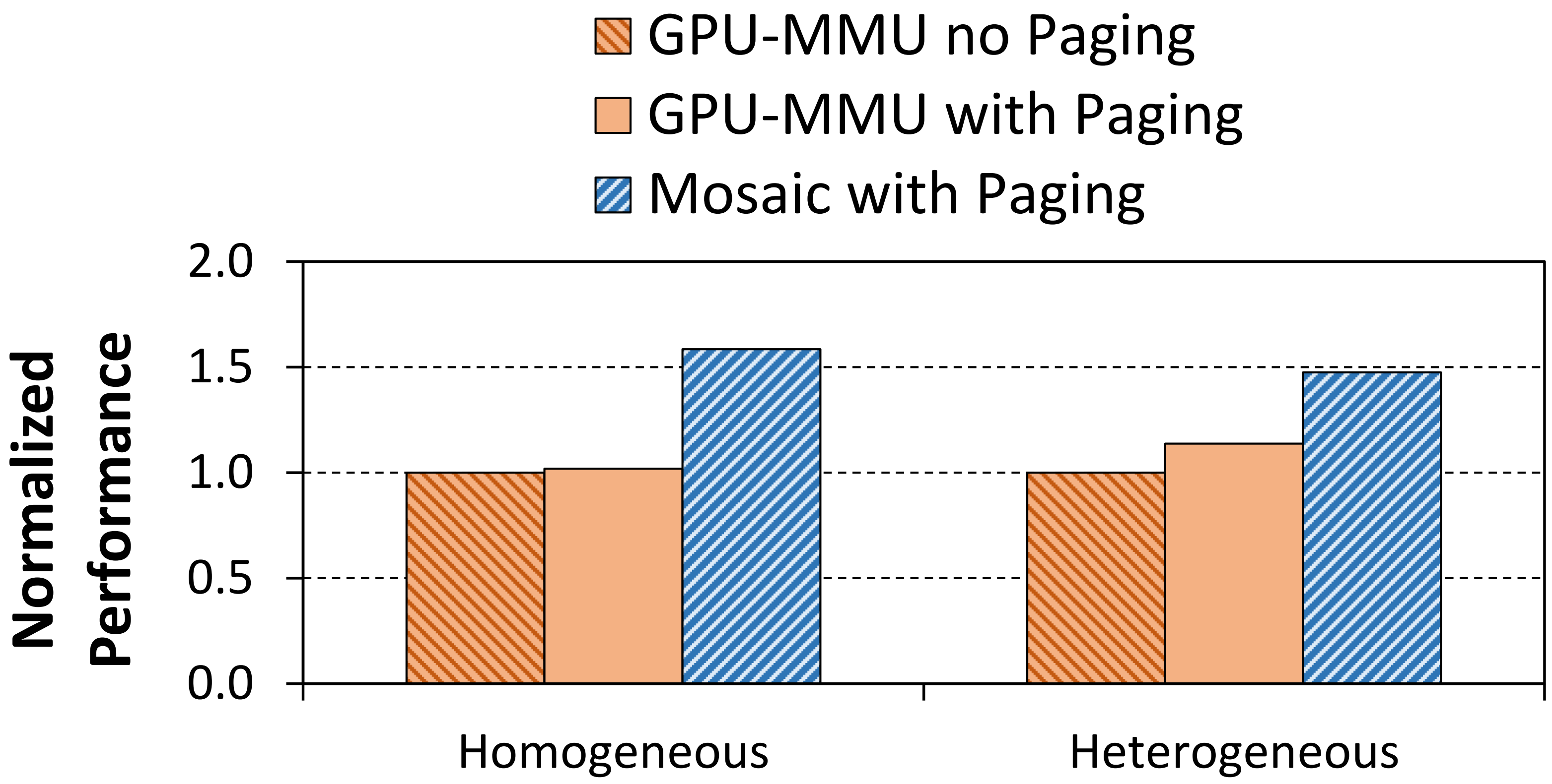
- **Static memory partitioning**

TLB Flush

- **With Mosaic, the contents in the page tables are the same**

- **TLB flush in Mosaic occurs when page table content is modified**
  - **This invalidates content in the TLB → Need to be flushed**
  - **Both large and small page TLBs are flushed**

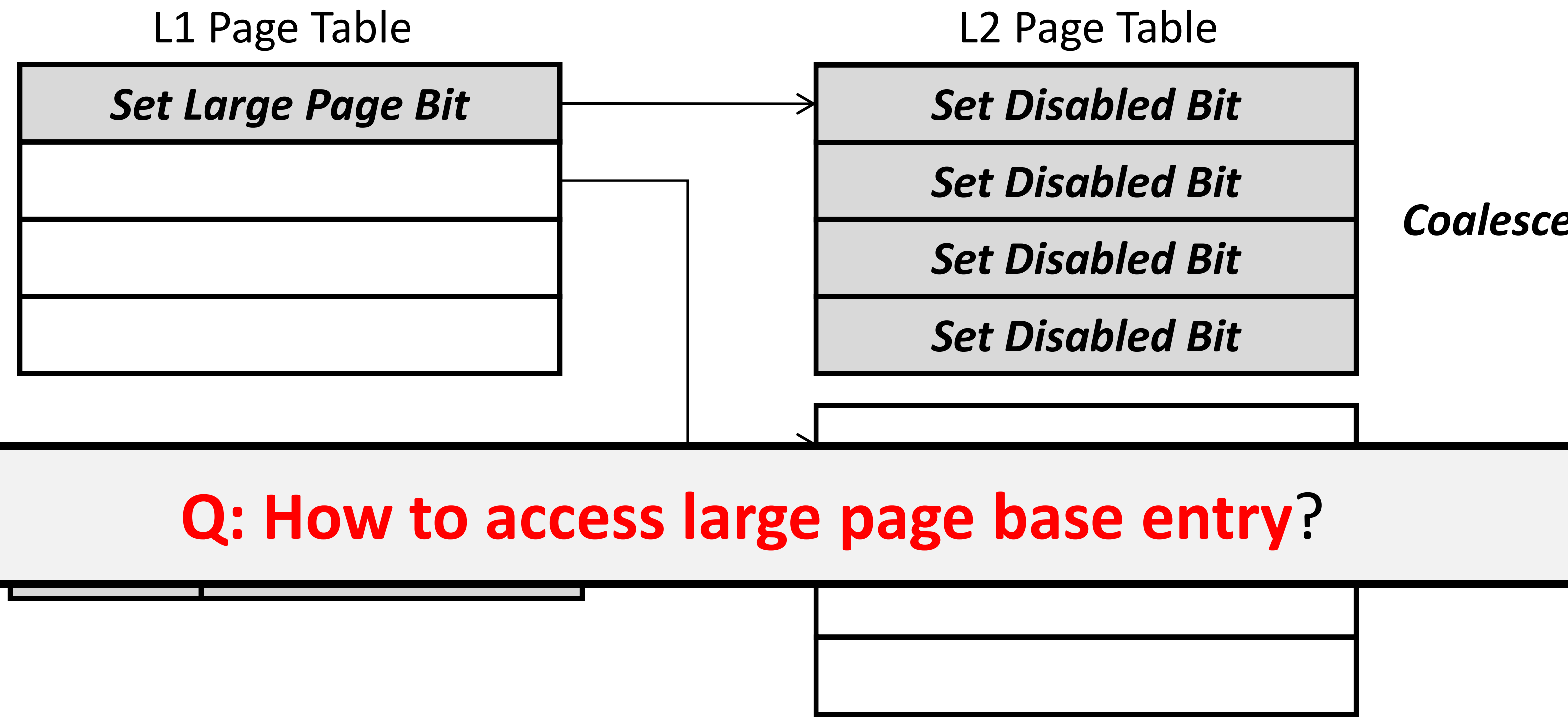
Performance with Demand Paging



SAFARI

In-Place Coalescer: Coalescing

- **Key assumption:** Soft guarantee
  - Large page range always contains pages of the same application



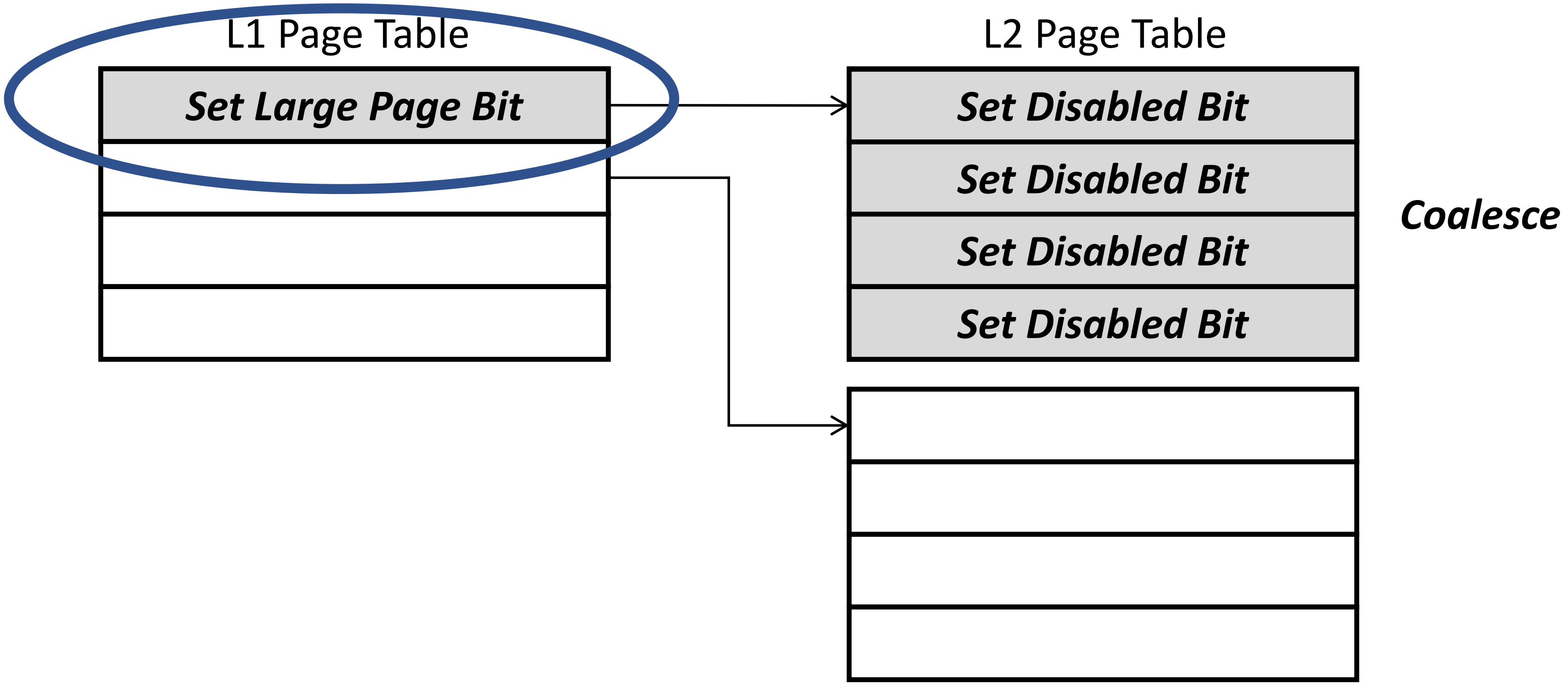
Q: How to access large page base entry?

Benefit: No data movement

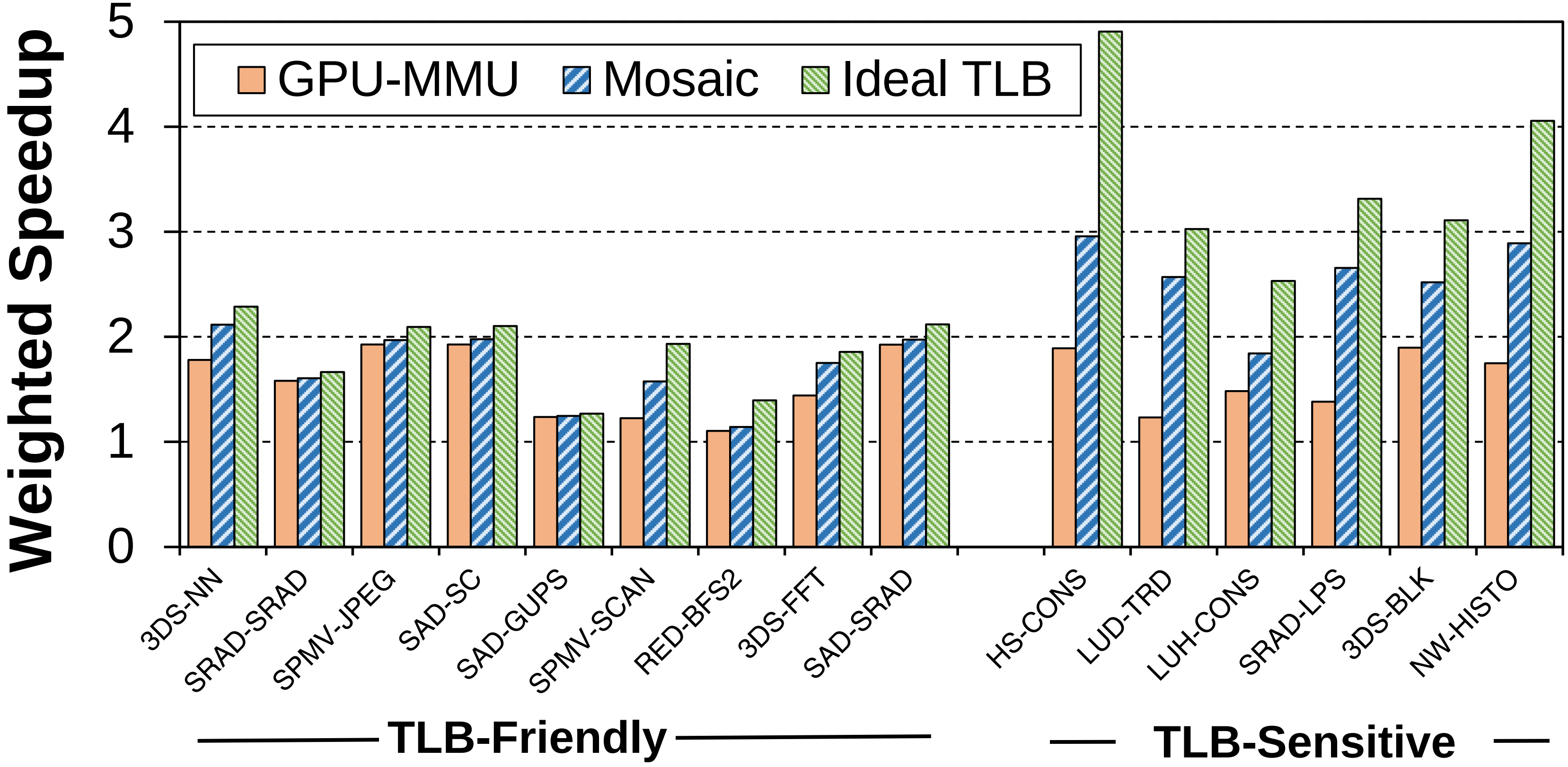
SAFARI

In-Place Coalescer: Large Page Walk

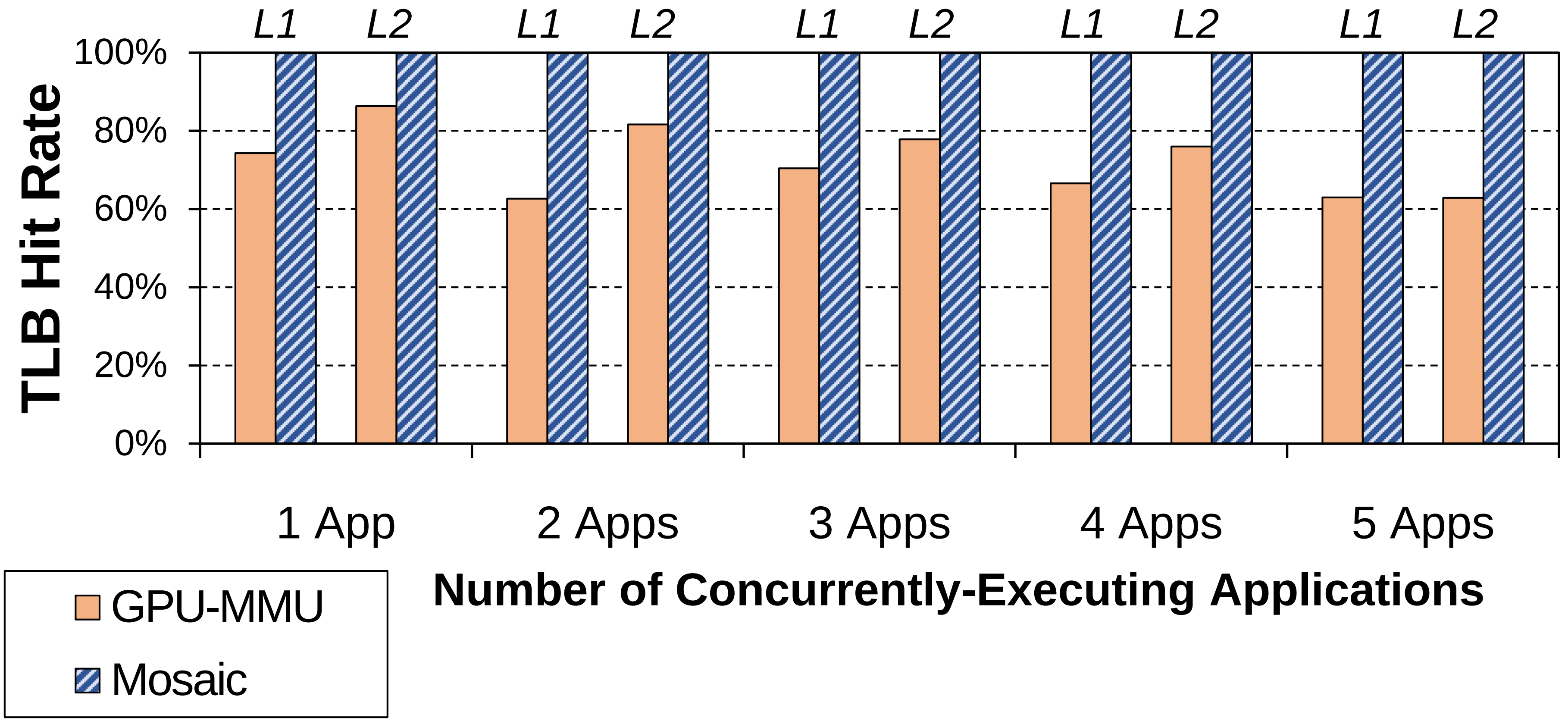
- Large page index is available at leaf PTE



Sample Application Pairs

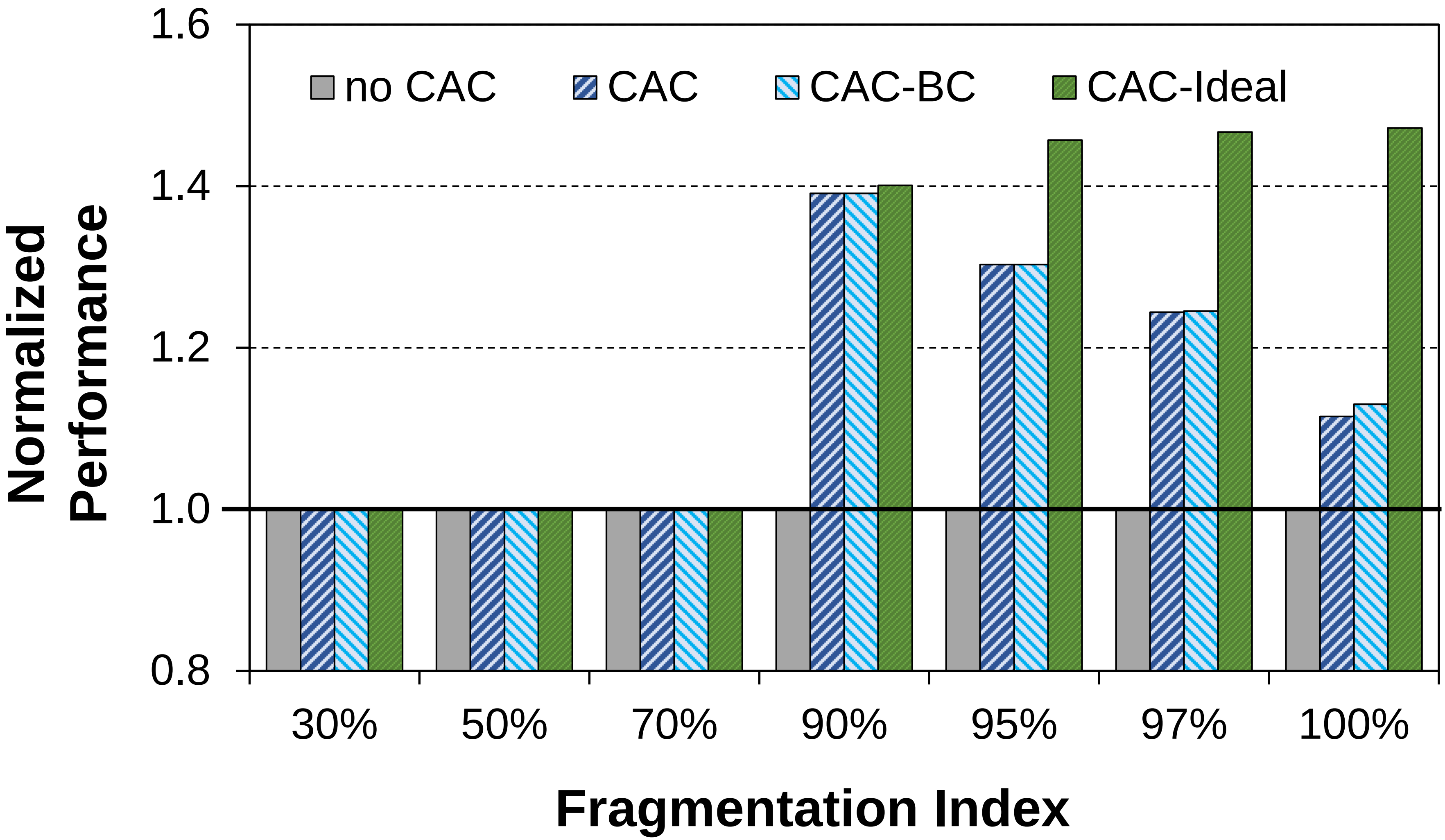


TLB Hit Rate

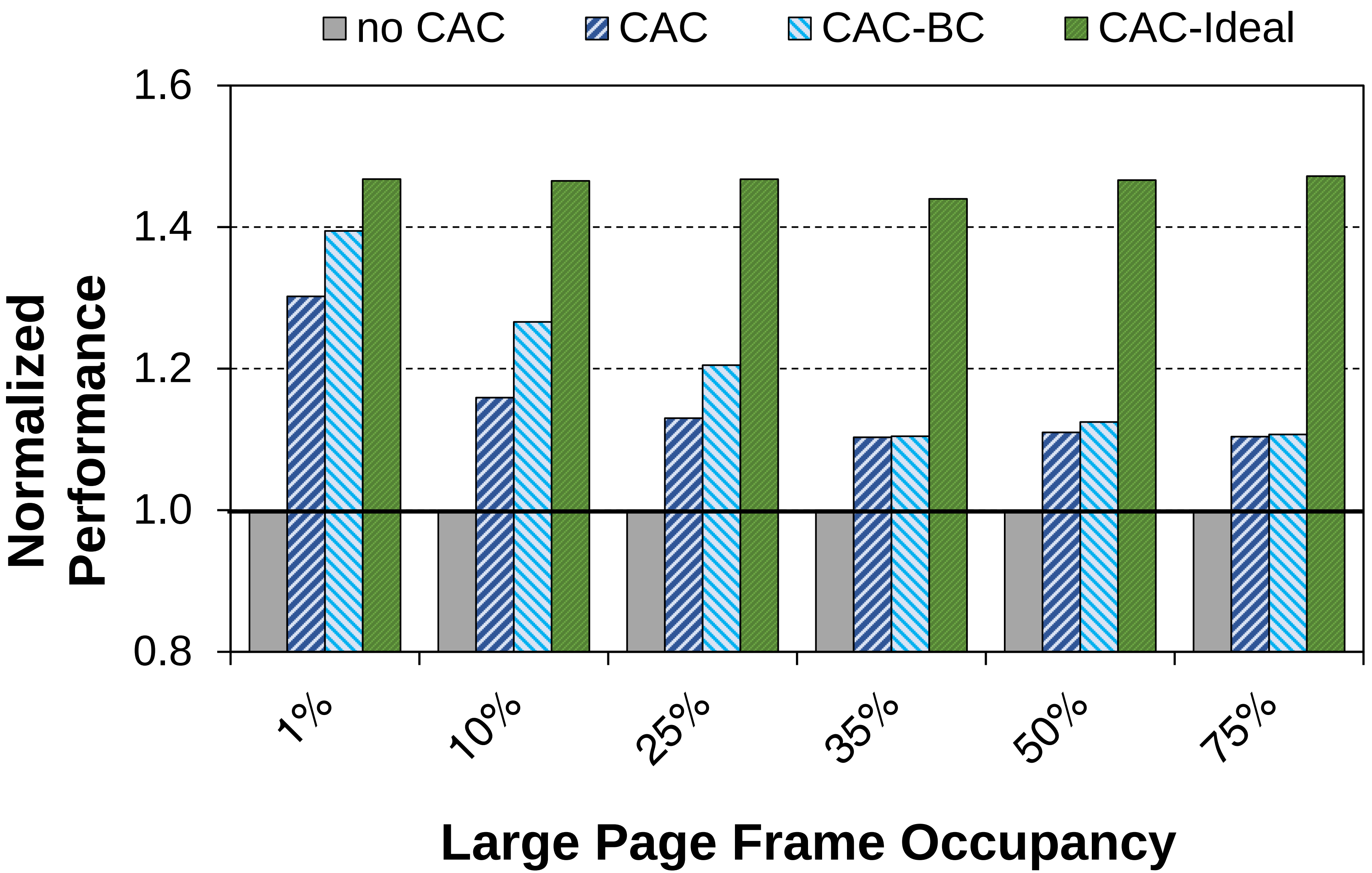


L1
L2
L1
L2
L1
L2
L1
L2
L1
L2
100%
80%
60%
40%
20%
0%
TLB Hit Rate
1 App
2 Apps
3 Apps
4 Apps
5 Apps
Number of Concurrently-Executing Applications
GPU-MMU
Mosaic

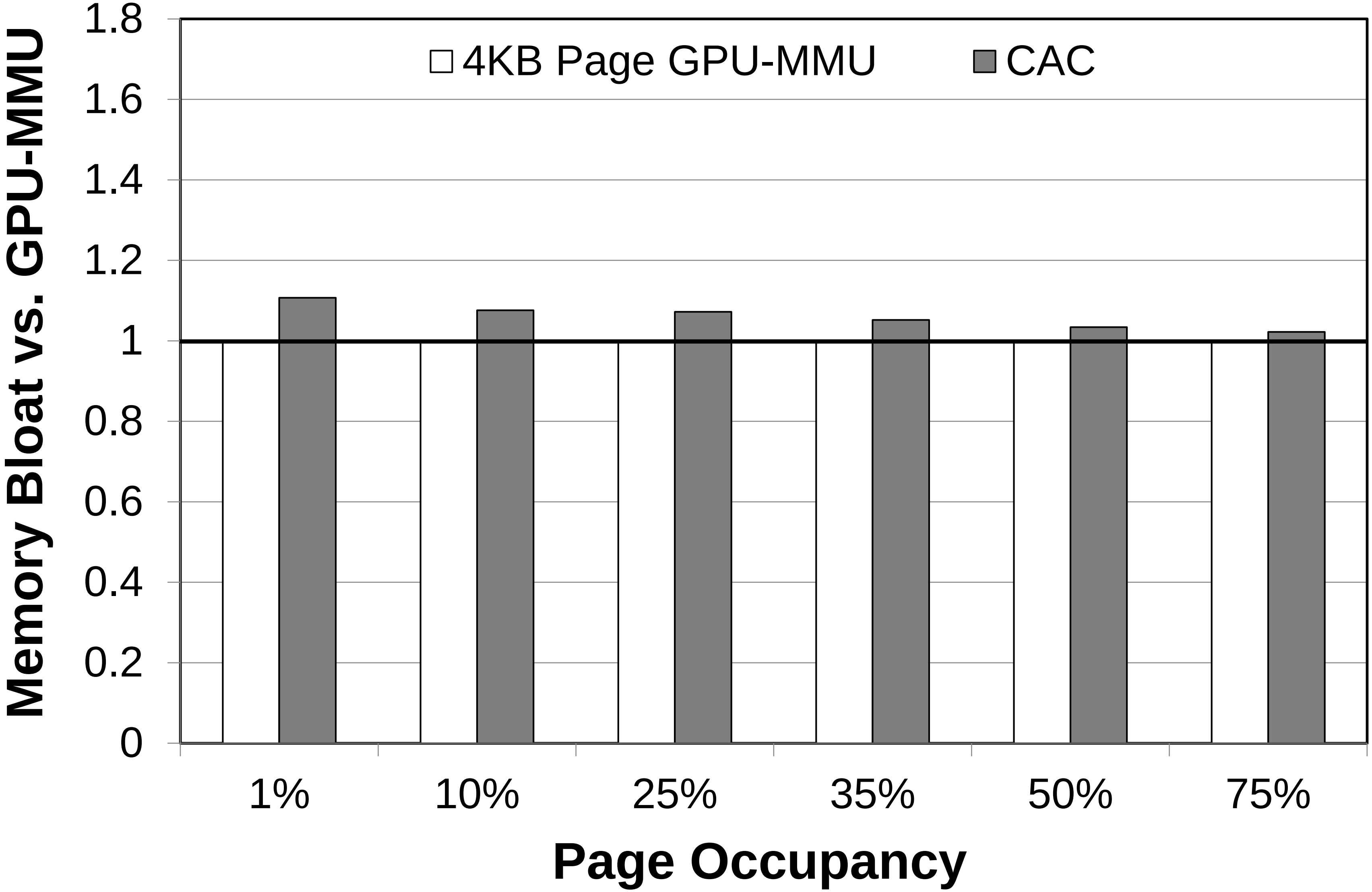
Pre-Fragmenting DRAM



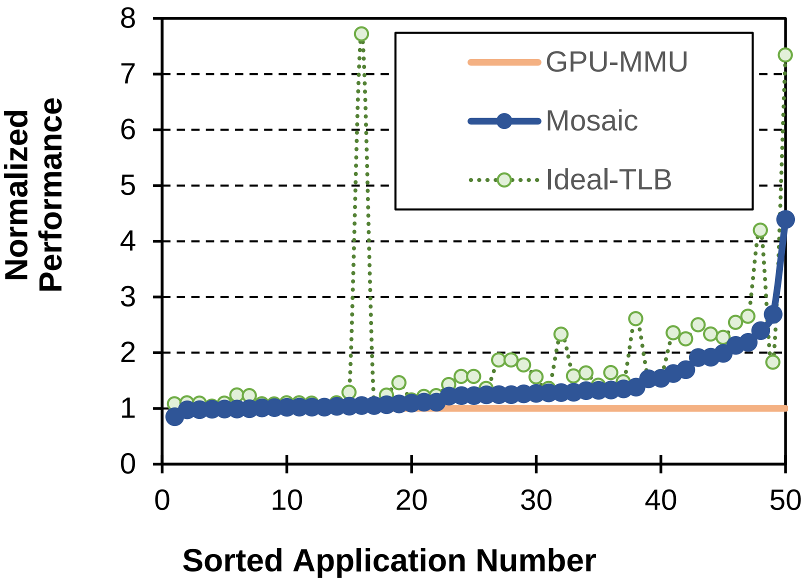
Page Occupancy Experiment

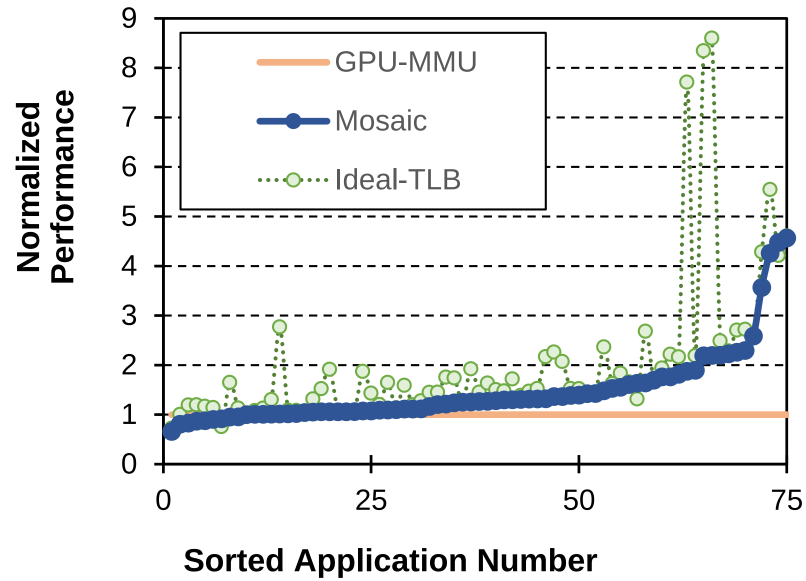


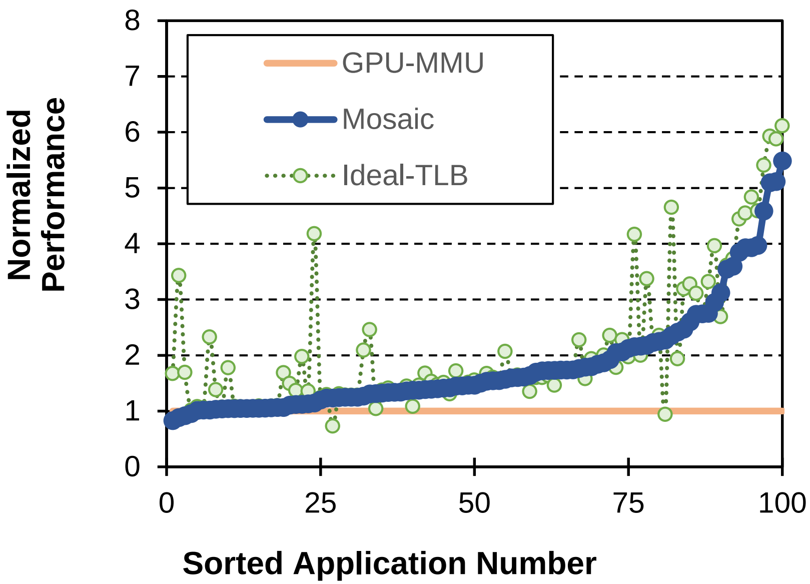
Memory Bloat

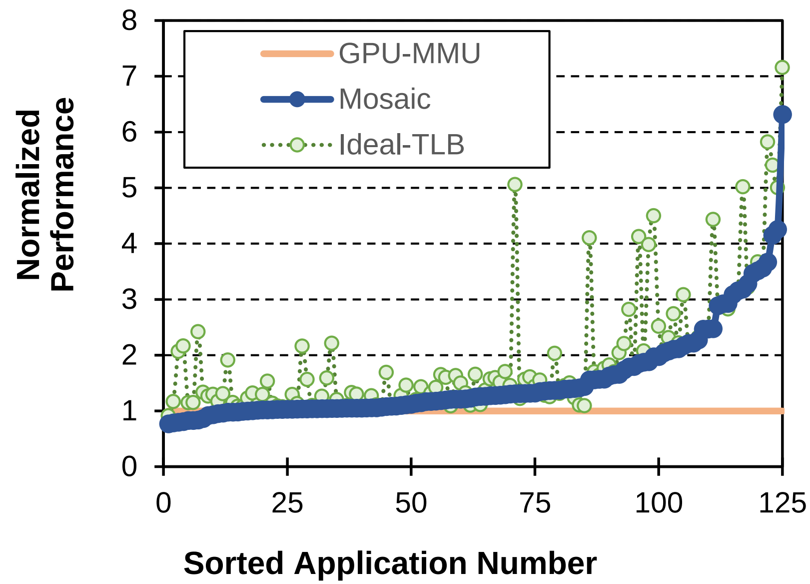


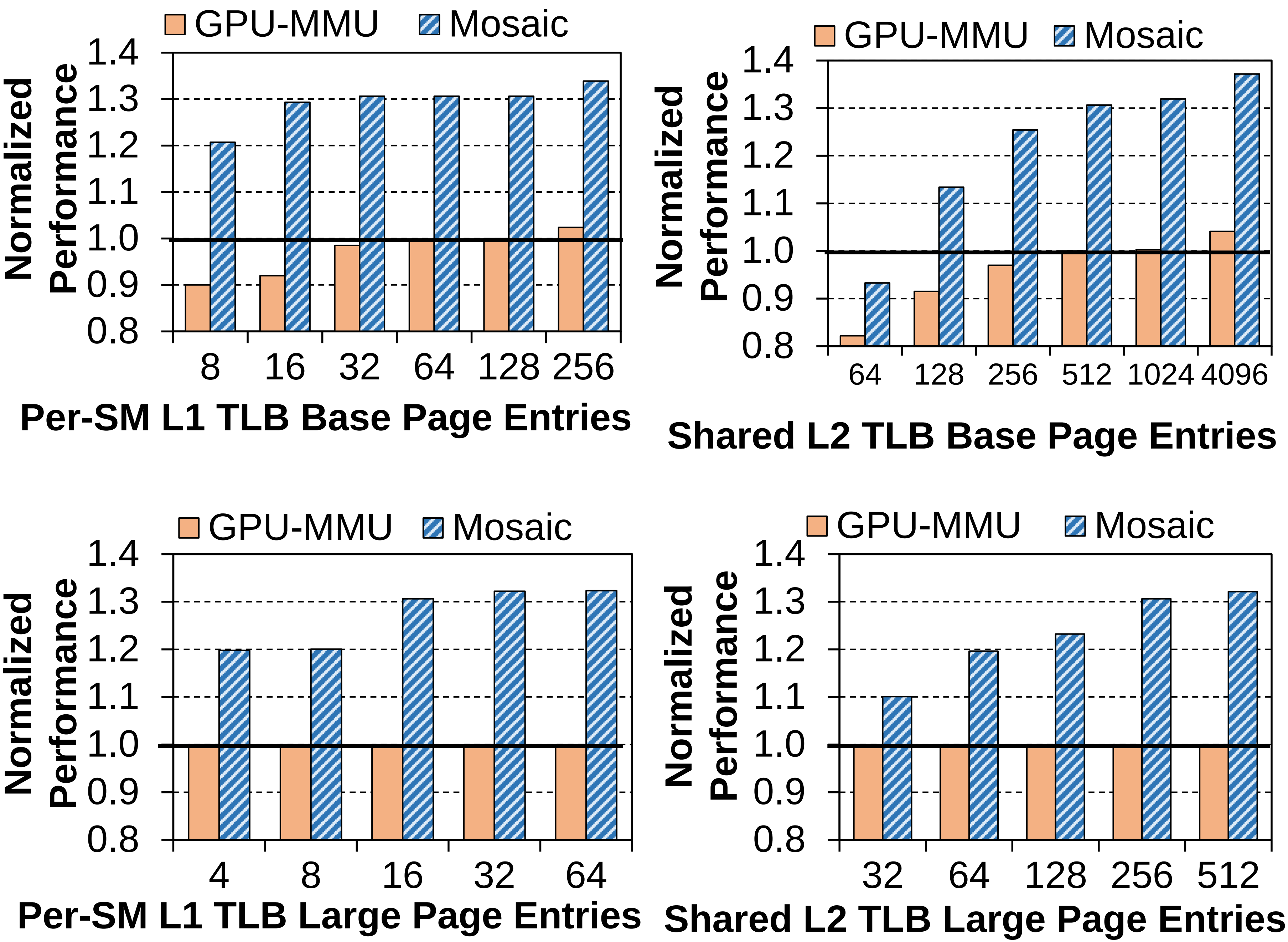
Individual Application IPC





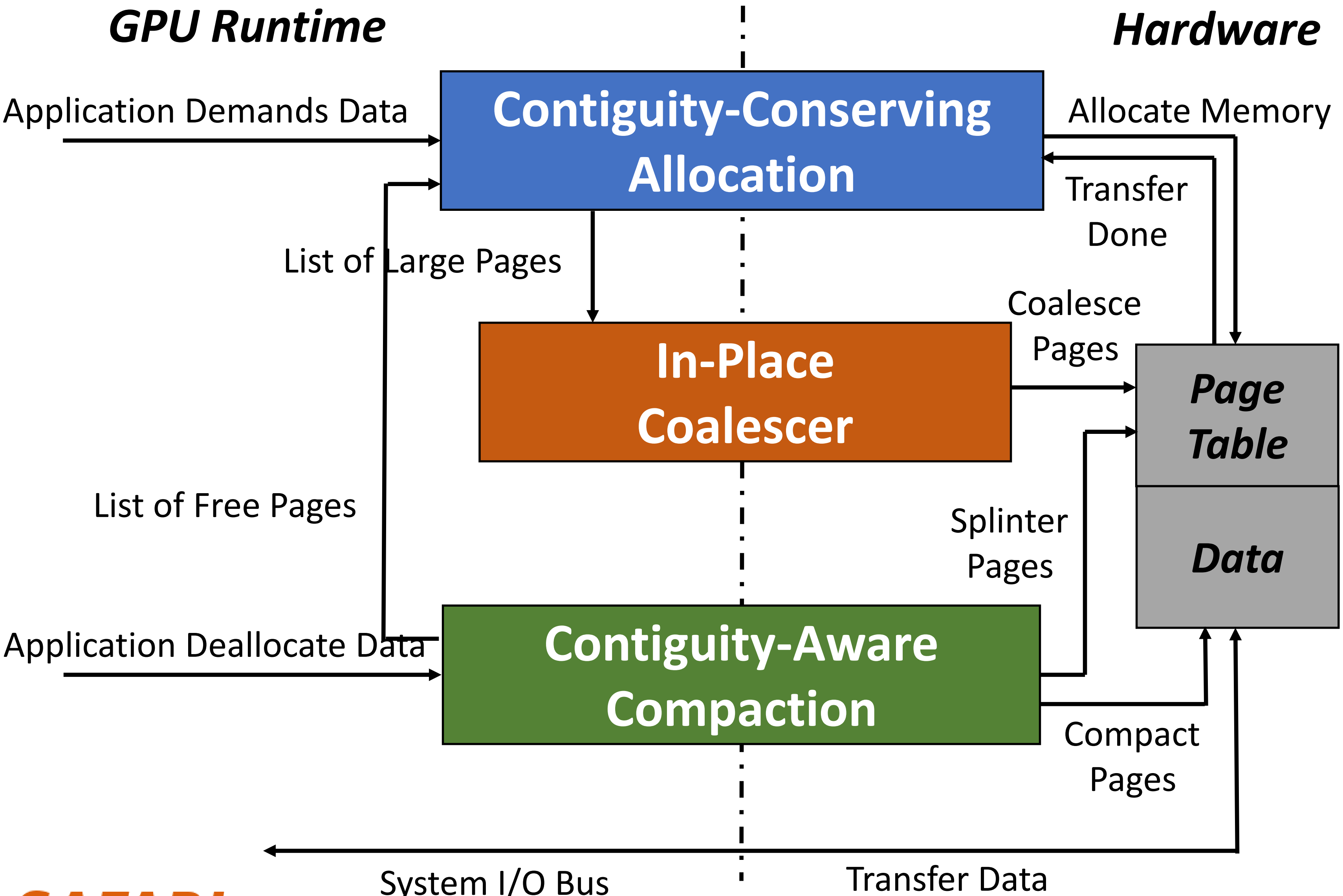




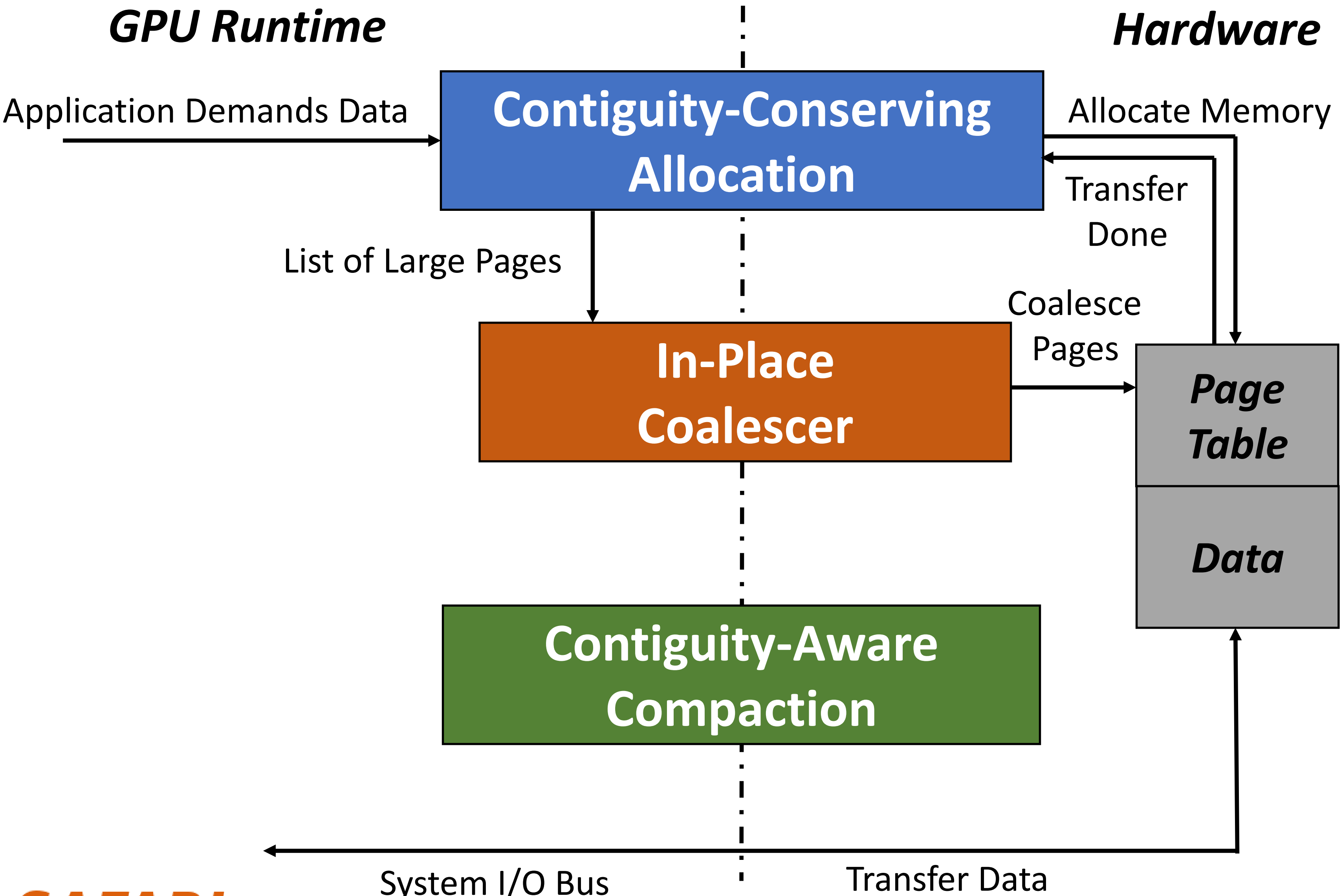


GPU-MMU
Mosaic
Normalized Performance
1.4
1.3
1.2
1.1
1.0
0.9
0.8
8
16
32
64
128
256
Per-SM L1 TLB Base Page Entries
64
128
256
512
1024
4096
Shared L2 TLB Base Page Entries
4
8
16
32
64
Per-SM L1 TLB Large Page Entries
32
64
128
256
512
Shared L2 TLB Large Page Entries

Mosaic: Putting Everything Together



Mosaic: Data Allocation



Mosaic: Data Deallocation