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
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:** Transiently 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

- **GPU Core**
  - Private TLB

- **Shared TLB**

- **Page Table Walkers**

- **Page Table (Main memory)**

- **Data (Main Memory)**

- **CPU Memory**

**High latency page walks**

**Limited TLB reach**

**High latency I/O**

**GPU-side memory**

**CPU-side memory**
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

**No Paging Overhead**
- Small (4KB)
- Large (2MB)

**With Paging Overhead**
- Small (4KB)
- Large (2MB)

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

State-of-the-Art

Time

App 1 Allocation
App 2 Allocation
App 1 Allocation
App 2 Allocation
Coalesce App 1 Pages
Coalesce App 2 Pages

Cannot coalesce (without migrating multiple 4K pages)

Need to search which pages to coalesce
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

GPU Memory

Large Page Frame 1

Large Page Frame 2

Large Page Frame 3

Large Page Frame 4

Large Page Frame 5

Can coalesce (without moving data)
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**
• Background
• Key challenges and our goal

• **Mosaic**
  • Contiguity-Conserving Allocation
  • In-Place Coalescer
  • Contiguity-Aware Compaction

• Experimental evaluations
• Conclusions
**Contiguity-Conserving Allocation**

1. Application Demands Data

2. Allocate Memory

**Page Table**

- Contiguity-Aware Compaction

**Large Page Frame**

- In-Place Coalescer

**Soft guarantee:**

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

**Conserves contiguity within the large page frame**
Data transfer is done at a small page granularity
A page that is transferred is immediately ready to use
Mosaic: Data Allocation

Contiguity-Conserving Allocation

In-Place Coalescer

Contiguity-Aware Compaction

CPU Memory

Page Table

Data

Large Page Frame

Transfer Data

System I/O Bus

Transfer Done

GPU Runtime

Hardware
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

GPU Runtime

Contiguity-Conserving Allocation

In-Place Coalescer

Contiguity-Aware Compaction

1 List of large pages

2 Update page tables

Large Page Table

Small Page Table

Coalesced Bit

Page Table

Data

- 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 $\rightarrow$ **Break down a large page** into small pages

  • Compaction $\rightarrow$ **Combine fragmented large page frames**
Mosaic: Data Deallocation

- Splinter only frames with deallocated pages
Mosaic: Compaction

**GPU Runtime**

- Contiguity-Conserving Allocation
- In-Place Coalescer

**Hardware**

- Contiguity-Aware 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
• **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

- **GPU Core**
  - Private TLB
- **GPU Core**
  - Private TLB
- **GPU Core**
  - Private TLB
- **GPU Core**
  - Private TLB

**Shared TLB**

**Page Table Walkers**

**Page Table (Main memory)**

**Data (Main Memory)**

**CPU Memory**

**GPU-side memory**

**CPU-side memory**

Private

Shared
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
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  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

- GPU-MMU no Paging
- GPU-MMU with Paging
- Mosaic with Paging

Normalized Performance

- Homogeneous
- Heterogeneous
In-Place Coalescer: Coalescing

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

**Benefit:** No data movement

**Q:** How to access large page base entry?

**SAFARI**
In-Place Coalescer: Large Page Walk

- Large page index is available at leaf PTE
Sample Application Pairs

Weighted Speedup

- GPU-MMU
- Mosaic
- Ideal TLB

TLB-Friendly

TLB-Sensitive

Sample Application Pairs
- 3DS-NN
- SRAD-SRAD
- SPMV-JPEG
- SAD-SC
- SAD-GUPS
- SPMV-SCAN
- RED-BFS2
- 3DS-FFT
- SAD-SRAD
- HS-CONS
- LUD-TRD
- LUH-CONS
- SRAD-LPS
- 3DS-BLK
- NW-HISTO
TLB Hit Rate

Number of Concurrently-Executing Applications

- GPU-MMU
- Mosaic
Pre-Fragmenting DRAM

Normalized Performance

Fragmentation Index

- no CAC
- CAC
- CAC-BC
- CAC-Ideal
Page Occupancy Experiment

Normalized Performance

no CAC | CAC | CAC-BC | CAC-Ideal

1% | 10% | 25% | 35% | 50% | 75%

Large Page Frame Occupancy
Memory Bloat

- Memory Bloat vs. GPU-MMU
- 4KB Page GPU-MMU
- CAC

Page Occupancy:
- 1%
- 10%
- 25%
- 35%
- 50%
- 75%
Individual Application IPC
Mosaic: Putting Everything Together

**GPU Runtime**

- Application Demands Data
- List of Large Pages
- List of Free Pages
- Application Deallocate Data

**Hardware**

- Allocate Memory
- Transfer Done
- Coalesce Pages
- Splinter Pages
- Compact Pages

**Contiguity-Conserving Allocation**

- Contiguity-Aware Compaction

**Page Table**

- Data

**System I/O Bus**

- Transfer Data

SAFARI
Mosaic: Data Allocation

**GPU Runtime**
- Application Demands Data
- List of Large Pages

**Contiguity-Conserving Allocation**

**Hardware**
- Allocate Memory
  - Transfer Done

**In-Place Coalescer**

- Coalesce Pages

**Contiguity-Aware Compaction**

**Page Table**

**Data**

System I/O Bus

Transfer Data
Mosaic: Data Deallocation

**GPU Runtime**

- Contiguity-Conserving Allocation
- In-Place Coalescer
- Contiguity-Aware Compaction

**Hardware**

- Page Table
- Data

- List of Free Pages
- Application Deallocate Data
- Compact Pages
- Splinter Pages