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

MICRO 2017

Presenter: Christina Giannoula
Background and Problem
Address Translation

Application

Virtual address space

Physical address space

Christina Giannoula
Address Translation

Application

Virtual address space

Physical address space

Page Granularity
Address Translation

Virtual address space

Physical address space

Application

Page Table

<table>
<thead>
<tr>
<th>Virtual</th>
<th>Physical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 0</td>
<td>Page 60</td>
</tr>
<tr>
<td>Page 1</td>
<td>Page 43</td>
</tr>
<tr>
<td>Page 2</td>
<td>Page 16</td>
</tr>
<tr>
<td>Page 3</td>
<td>Page 63</td>
</tr>
<tr>
<td>Page 4</td>
<td>Page 64</td>
</tr>
<tr>
<td>Page 5</td>
<td>Page 42</td>
</tr>
<tr>
<td>Page 6</td>
<td>Page 73</td>
</tr>
<tr>
<td>Page 7</td>
<td>Page 234</td>
</tr>
</tbody>
</table>
Page Table Walk

- Look up a mapping
- Page Table Walk:
  Ten to hundreds of cycles

<table>
<thead>
<tr>
<th>Virtual</th>
<th>Physical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 0</td>
<td>Page 60</td>
</tr>
<tr>
<td>Page 1</td>
<td>Page 43</td>
</tr>
<tr>
<td>Page 2</td>
<td>Page 16</td>
</tr>
<tr>
<td>Page 3</td>
<td>Page 63</td>
</tr>
<tr>
<td>Page 4</td>
<td>Page 64</td>
</tr>
<tr>
<td>Page 5</td>
<td>Page 42</td>
</tr>
<tr>
<td>Page 6</td>
<td>Page 73</td>
</tr>
<tr>
<td>Page 7</td>
<td>Page 234</td>
</tr>
</tbody>
</table>

Found!
Page Table Walk

- Look up a mapping

<table>
<thead>
<tr>
<th>Virtual</th>
<th>Physical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 0</td>
<td>Page 60</td>
</tr>
<tr>
<td>Page 1</td>
<td>Page 43</td>
</tr>
<tr>
<td>Page 2</td>
<td>Page 16</td>
</tr>
<tr>
<td>Page 3</td>
<td>Page 63</td>
</tr>
<tr>
<td>Page 4</td>
<td>Page 64</td>
</tr>
<tr>
<td>Page 5</td>
<td>Page 42</td>
</tr>
<tr>
<td>Page 6</td>
<td>Page 73</td>
</tr>
<tr>
<td>Page 7</td>
<td>Page 234</td>
</tr>
</tbody>
</table>

Page Table Walks: High Latency

Found!
Translation Lookaside Buffers (TLB)

- **TLBs:**
  - Store *recently* used address translations
  - Address translation *cache*

<table>
<thead>
<tr>
<th>Virtual</th>
<th>Physical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 1</td>
<td>Page 60</td>
</tr>
<tr>
<td>Page 3</td>
<td>Page 63</td>
</tr>
<tr>
<td>Page 4</td>
<td>Page 64</td>
</tr>
<tr>
<td>Page 5</td>
<td>Page 42</td>
</tr>
<tr>
<td>Page 0</td>
<td>Page 60</td>
</tr>
<tr>
<td>Page 1</td>
<td>Page 43</td>
</tr>
<tr>
<td>Page 2</td>
<td>Page 16</td>
</tr>
<tr>
<td>Page 3</td>
<td>Page 63</td>
</tr>
<tr>
<td>Page 4</td>
<td>Page 64</td>
</tr>
<tr>
<td>Page 5</td>
<td>Page 42</td>
</tr>
<tr>
<td>Page 6</td>
<td>Page 73</td>
</tr>
<tr>
<td>Page 7</td>
<td>Page 234</td>
</tr>
</tbody>
</table>
State-of-the-art Virtual Memory on GPUs
State-of-the-art Virtual Memory on GPUs

TLB miss!
State-of-the-art Virtual Memory on GPUs

GPU core
Private TLB

GPU core
Private TLB

GPU core
Private TLB

GPU core
Private TLB

Shared TLB

TLB miss!
State-of-the-art Virtual Memory on GPUs

- GPU core
  - Private TLB
- GPU core
  - Private TLB
- GPU core
  - Private TLB
- GPU core
  - Private TLB

- Shared TLB
- Page Table Walkers
- Page Table (GPU Memory)
- Data (GPU Memory)

Private
Shared

High Latency
State-of-the-art Virtual Memory on GPUs

GPU core
Private TLB

Shared TLB

Page Table Walkers

Page Table (GPU Memory)

Data (GPU Memory)

GPU-side memory

CPU-side memory

CPU Memory

I/O bus

Private
Shared
State-of-the-art Virtual Memory on GPUs

- GPU core
  - Private TLB

- GPU core
  - Private TLB

- GPU core
  - Private TLB

- GPU core
  - Private TLB

Shared TLB

Page Table Walkers

Page Table (GPU Memory)

Data (GPU Memory)

CPU Memory

GPU Threads

I/O bus

GPU-side memory

CPU-side memory

Private

Shared
Address Translation Challenge

Small Pages

Virtual address space

Physical address space
Address Translation Challenge

Small Pages (4KB)

Virtual address space

Physical address space

Large Pages (2MB)

Virtual address space

Physical address space
Address Translation Challenge

Small Pages

Physical address space

Virtual address space

Large Pages

Physical address space

Virtual address space

TLB

Fixed Size

3 entries
Address Translation Challenge

Small Pages

Virtual address space

Physical address space

Large Pages

Virtual address space

Physical address space

TLB

Fixed Size

3 entries
Address Translation Challenge

Small Pages

Virtual address space

Physical address space

Limited TLB reach

Large Pages

Virtual address space

Physical address space

Fixed Size

3 entries
Address Translation Challenge

Small Pages

Virtual address space

Physical address space

Large Pages

Virtual address space

Physical address space

TLB

Fixed Size

3 entries
Address Translation Challenge

Small Pages

Virtual address space

Physical address space

Large Pages

Virtual address space

Physical address space

TLB

Fixed Size

3 entries

Better TLB reach
Large page size is better!
Demand Paging Challenge

- An application requests data that **is not** currently resident in GPU memory
- A **Page Fault** is triggered
- Transfer data in **page-granularity**
Demand Paging Challenge

- An application requests data that **is not** currently resident in GPU memory.
Demand Paging Challenge

- An application requests data that **is not** currently resident in GPU memory.
An application requests data that **is not** currently resident in GPU memory.

- Small page size is better!

- GPU Threads stall ...

- GPU-side memory
- CPU-side memory

- Large Pages

- High Latency

- I/O bus

- GPU Memory

- large memory
Page Size Trade-Off

Small Pages
- Low TLB reach
- Low demand paging latency

VS

Large Pages
- High TLB reach
- High demand paging latency

Can we get the best of both page sizes?
Executive Summary
Executive Summary

- Problem
  - No single best page size for GPU virtual memory (large vs small pages)

- Goal
  - Transparently and efficiently enable both page sizes
Executive Summary

- **Problem**
  - No single best page size for GPU virtual memory (large vs small pages)

- **Goal**
  - Transparently and efficiently enable both page sizes

- **Key Observation**
  - Can easily coalesce an application’s contiguously-allocated small pages into a large page
  - GPGPU applications typically allocate large chunks of memory at once
Executive Summary

- Problem
  - No single best page size for GPU virtual memory (large vs small pages)

- Goal
  - Transparently and efficiently enable both page sizes

- Key Observation
  - Can easily coalesce an application’s contiguously-allocated small pages into a large page
  - GPGPU applications typically allocate large chunks of memory at once

- Key Idea
  - Preserve the virtual address contiguity of small pages when allocating physical memory to simplify coalescing

- Mosaic:
  - A hardware/software cooperative framework
  - Enables the benefits of both small and large pages

- Key Result: 55% on average performance improvement over state-of-the-art GPU memory management mechanism
Key Ideas and Challenges
Key Ideas

- Translate using **large page size**
  - High TLB reach
- Transfer using **small page size**
  - Low demand paging latency

**Contiguity**

- Small pages
- Large page frame

**Virtual address space**

**Physical address space**

**Large Pages**

**Small Pages**

**GPU Memory**

**CPU Memory**

I/O bus
Challenges with Multiple Page Sizes

Time

App1 Allocation
App2 Allocation
App1 Allocation
App2 Allocation
Coalesce App1
Pages

Need to search which pages to coalesce

State-of-the-art

Large page frame 1
Large page frame 2
Large page frame 3
Large page frame 4
Large page frame 5

App1
App2
Unallocated
Challenges with Multiple Page Sizes

<table>
<thead>
<tr>
<th>Time</th>
<th>App1 Allocation</th>
<th>App2 Allocation</th>
<th>App1 Allocation</th>
<th>App2 Allocation</th>
<th>Coalesce</th>
<th>App2 Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Large page frame 1</td>
<td>Large page frame 2</td>
<td>Large page frame 3</td>
<td>Large page frame 4</td>
<td>Large page frame 5</td>
<td>State-of-the-art</td>
</tr>
</tbody>
</table>

App1

App2

Unallocated
Challenges with Multiple Page Sizes

Time

App1 Allocation

App2 Allocation

App1 Allocation

App2 Allocation

Coalesce App2 Pages

State-of-the-art

Large page frame 1

Large page frame 2

Large page frame 3

Large page frame 4

Large page frame 5

Cannot coalesce without migrating multiple pages

App1

App2

Unallocated

Christina Giannoula
Why page migration is bad

- GPU core
  - Private TLB

- GPU core
  - Private TLB

- GPU core
  - Private TLB

- GPU core
  - Private TLB

- GPU core
  - Private TLB

- Shared TLB

- Page Table Walkers

- Page Table (GPU Memory)

- Data (GPU Memory)
Why page migration is bad

GPU core

Private TLB

GPU core

Private TLB

GPU core

Private TLB

GPU core

Private TLB

Shared TLB

Page Table Walkers

Page Table (GPU Memory)

Data (GPU Memory)

TLB flush!
Why page migration is bad

TLB flush!

GPU Threads

Stall ...

Page Table Walkers

Page Table (GPU Memory)

Data (GPU Memory)

GPU core

Private TLB

GPU core

Private TLB

GPU core

Private TLB

GPU core

Private TLB

Private

Shared
Desirable Allocation

Can coalesce without moving data

App1
App2
Unallocated
Key Ideas

- Use **multiple** page sizes
- Translate using **large page size**
  High TLB reach
- Transfer using **small page size**
  Low demand paging latency
- Allocate physical pages in a way that **avoids the need to migrate data**
Mechanism
Mosaic

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

GPU Runtime

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

Hardware
Mosaic

- 3 components
- **Contiguity-Conserving Allocation**
- In-Place Coalescer
- Contiguity-Aware Compaction

GPU Runtime

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

Hardware
Mosaic: Data Allocation

- A typical GPGPU application allocates a large **number of base pages**
### Mosaic: Data Allocation

- **Conserve contiguity of base pages** - Virtual memory are contiguous within a large page frame in physical memory.

---

**Key observation:** *en masse* memory allocation, i.e., applications allocate a large number of base pages at once.
Mosaic: Data Allocation

- Conserve contiguity of base pages - Virtual memory are contiguous within a large page frame in physical memory

GPU Runtime → Application demands data → Contiguity-Conserving Allocation

- In-Place Coalescer
- Contiguity-Aware Compaction

Allocate memory

Page Table

Data

Large Page Frame

Soft guarantee:
A large page frame contains pages from only a single address space
Mosaic: Data Allocation

Transfer data/pages at a **small page** granularity
- A page that is transferred is immediately **ready to use** – low latency
Mosaic: Data Allocation

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

**GPU Runtime**
- Application demands data

**Allocate memory**

**GPU Memory**
- Page Table
- Page

**Large Page Frame**

**Transfer data – System I/O bus**
- Transfer data/pages at a **small page** granularity
  - A page that is transferred is immediately **ready to use** – low latency

**CPU Memory**

**GPU Threads**
- stall
Mosaic: Data Allocation

- Transfer data/pages at a **small page** granularity
  - A page that is transferred is immediately **ready to use** – low latency
Send to In-Place Coalescer a list of the large page frame addresses that were allocated.
Mosaic

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

GPU Runtime

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

Hardware
Mosaic: Coalescing

- Fully-allocated large page frames = coalesceable
  - **Contiguous** in both virtual and physical memory
  - All base pages within the large page frame have been allocated and belong to the same address space

GPU Runtime

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

- Sends the list of large pages
- Coalesce pages
Mosaic: Coalescing

Coalesce without moving data
- Simply update the page tables
- No need for TLB flush

With an application transparent way
Mosaic: Coalescing

Data can be accessed using either page size
Mosaic

- 3 components
- Contiguity-Conserving Allocation
- In-Place Coalescer
- **Contiguity-Aware Compaction**

GPU Runtime

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

Hardware
Mosaic: Data Deallocation

- It sends a deallocation request to the GPU runtime.
- The Runtime invokes Contiguity-Aware Compaction for the corresponding large page.
Mosaic: Data Deallocation

- Check whether the large page frame has **high degree of internal fragmentation**
- Free-up not fully-used large page frames
Mosaic: Data Deallocation

- Update the page table to splinter the large page back into its constituent base pages
Mosaic: Data Deallocation

- Compaction: **Migrating** the remaining base pages to another uncoalesced large page frame that belongs to the same application
Compaction: **Migrating** the remaining base pages to another uncoalesced large page frame that belongs to the same application.
Compaction: Migrating the remaining base pages to another uncoalesced large page frame that belongs to the same application.
Mosaic: Data Deallocation

- Page Migration is required
- TLB flush is required
Contiguity-Aware Compaction component notifies Contiguity-Conserving Allocation of the large page frames that are now free after compaction, such that to be used for future memory allocations.
Methodology
Methodology

- MAFIA framework which uses GPGPU-Sim
  - 30 cores @1020MHz
  - 64KB 4-way L1, 2048KB 16-way L2
  - Private L1 TLB: 128 base pages / 16 large page entries per core
  - Shared L2 TLB: 512 base pages / 256 large page entries
  - DRAM: 3GB GDDR5 @1674 MHz

- Model sequential page walks

- Workloads
  - Homogeneous workloads = multiple copies of the same application
  - Heterogeneous workloads = randomly selected applications
  - Multiple GPGPU applications execute concurrently
  - Test suites: Parboil, SHOC, LULESH, Rodinia, CUDA SDK

- Evaluation metric
  - Weighted Speedup = \( \sum_{i=1}^{N} \frac{IPC_{\text{shared}}}{IPC_{\text{alone}}} \) for each application \( i \)
Comparison Points

1. State-of-the-art CPU-GPU memory management
   - GPU-MMU based on [Power et al., HPCA’14]
     - Utilizes parallel page walks, TLB request coalescing and page walk cache to improve performance
     - Limited TLB reach (4KB pages)

2. Ideal TLB: Every TLB access is a L1 TLB hit
Evaluation
Homogeneous workloads

Mosaic consistently improves performance:

- 55.5% averaged over GPU-MMU across 135 workloads
- comes within 6.8% of the Ideal TLB
Heterogeneous workloads

![Graph showing weighted speedup for various workloads]

- **GPU-MMU**
- **Mosaic**
- **Ideal TLB**

**TLB-Friendly**

**TLB-Sensitive**
Heterogeneous workloads

Mosaic:

• significantly improves performance for TLB-friendly workloads
Heterogeneous workloads

Mosaic:
- significantly improves performance for TLB-friendly workloads
- provides less benefit in TLB-sensitive workloads
Mosaic significantly reduces the TLB miss rate:
- the average miss rate falls **below 1%** in both the L1 and L2 TLBs
More Results in the Paper

- Per-application IPC
- Sensitivity analysis to different TLB sizes
- Memory fragmentation analysis

Mosaic is available at: https://github.com/CMU-SAFARI/Mosaic
Summary
Executive Summary

- **Problem**
  - No single best page size for GPU virtual memory (large vs small pages)

- **Goal**
  - Transparently and efficiently enable both page sizes

- **Key Observation**
  - Can easily coalesce an application’s contiguously-allocated small pages into a large page
  - GPGPU applications typically allocate large chunks of memory en masse

- **Key Idea**
  - Preserve the virtual address contiguity of small pages when allocating physical memory to simplify coalescing

- **Mosaic:**
  - A hardware/software cooperative framework
  - Enables the benefits of both small and large pages

- **Key Result:** 55% on average performance improvement over state-of-the-art GPU memory management mechanism
Strengths & Weaknesses
Strengths

- **Intuitive Idea:**
  - Exploits the benefits of using both small and large pages
  - Well-written, insightful paper
Strengths

- **Intuitive Idea:**
  - Exploits the benefits of using both small and large pages
  - Well-written, insightful paper

- **Mechanism:**
  - Avoids page migration when memory allocation is requested
  - Application-transparent support for multiple page sizes
  - Data can be accessed using either page size
Strengths

- **Intuitive Idea:**
  - Exploits the benefits of using both small and large pages
  - Well-written, insightful paper

- **Mechanism:**
  - Avoids page migration when memory allocation is requested
  - Application-transparent support for multiple page sizes
  - Data can be accessed using either page size

- **Evaluation:**
  - Investigate behavior of multiple GPGPU applications that run concurrently
  - Explore the performance in case of highly fragmented pages
  - High variety of workloads explored
Strengths

- Intuitive Idea:
  - Exploits the benefits of using both small and large pages
  - Well-written, insightful paper

- Mechanism:
  - Avoids page migration when memory allocation is requested
  - Application-transparent support for multiple page sizes
  - Data can be accessed using either page size

- Evaluation:
  - Investigate virtual memory when multiple GPGPU applications run concurrently
  - Explore the performance in case of highly fragmented pages
  - High variety of workloads explored

- Online available
Weaknesses

- **Mechanism:**
  - Provides *soft guarantee* that a large page frame contains pages from only a single address space.
  - What is the threshold after which Mosaic splinters a large page frame into small pages?
  - Needs many changes in the system stack.
    - Software-hardware cooperative solutions are not always easy to adopt.
Weaknesses

- **Mechanism:**
  - Provides soft guarantee that a large page frame contains pages from only a single address space
  - What is the threshold after which Mosaic splinters a large page frame into small pages?
  - Needs many changes in the system stack
    - Software-hardware cooperative solutions are not always be easy to adopt

- **Evaluation:**
  - No comparison with an approach that uses large page frames
    - Mosaic mainly benefits from TLB-friendly applications
  - Model sequential page walks in simulation
  - Less benefit in TLB sensitive applications and highly fragmented pages
  - Simulation-based evaluation
Takeaways
Takeaways

- A novel idea to enable benefits of both small and large pages
- Hardware/software cooperative framework
- Application-transparent support for multiple page sizes
  - No TLB flush when coalescing
- Online available
- Easy to read and understand paper
Open Discussion
Discussion

- We do not completely avoid data migration!

- Avoid page migration to the critical path:
  - `gpu_malloc();`
  - `... access ...`

On allocation: do not move the data
Discussion

- We do not completely avoid data migration!

- Avoid page migration to the critical path:
  - `gpu_malloc();`
  - `... access ...`
  - `gpu_free();`
  - On deallocation: move the data
Discussion

- We do not completely avoid data migration!

- Avoid page migration in the critical path:
  - `gpu_malloc();`
  - `... access ...`
  - `gpu_free();`
  - On deallocation:

- Any similar concepts?

Be Optimistic!
Discussion

- We do not completely avoid data migration!

  • Hardware Transactional Memory (HTM)
  • Lazy PIM: CAL 2016
  • Other works related to speculation?

Be Optimistic!
Discussion

- Mosaic does not significantly improve the performance for **TLB sensitive** workloads
- No comparison with other research works that use **large** page sizes
- Any ideas to extend this work for TLB-sensitive applications?
  - TLB prefetching?
Discussion

- TLB is **shared among multiple** concurrently-executing applications. These applications **compete** for the shared TLB.
  - Can we improve inter-application interference?
Discussion

- TLB is shared among multiple concurrently-executing applications. These applications compete for the shared TLB.
  - Can we improve inter-application interference?
  - TLB partitioning?
  - Static/Dynamic partitioning?

MASK: Redesigning the GPU Memory Hierarchy to Support Multi-Application Concurrency

Rachata Ausavarungnirun¹  Vance Miller²  Joshua Landgraf²  Saugata Ghose¹
Jayneel Gandhi³  Adwait Jog⁴  Christopher J. Rossbach²,³  Onur Mutlu⁵,¹

¹Carnegie Mellon University  ²University of Texas at Austin  ³VMware Research
⁴College of William and Mary  ⁵ETH Zürich
Is the ‘ideal’ page size an application-specific parameter? How can we predict the ‘correct’ page size for each application?
- tracks the difference or distance between successive TLB miss virtual pages to identify it

How to apply such an idea when different applications are executing concurrently?
Mosaic provides a **soft guarantee** that a large page frame contains pages from only a single address space.

- No discussion about the heuristic used
- Any good heuristic/idea?
printf ( "Thank you" );