Full Virtualization for GPUs Reconsidered


Hangchen Yu¹, Christopher J. Rossbach¹,²

¹The University of Texas at Austin
²VMware Research Group
Overview

• Demands, introductions, challenges of virtual GPUs
• Distinctive features of GPUvm
• Re-evaluate GPUvm with additional benchmarks
  – Hard to set up the testbed
  – Some functionalities do not work
  – Over 200x overheads on average
  – Unfairness issue
  – Over 40% throughput loss
Do we still need GPU virtualizations?

- Share GPUs in datacenter
- Different end-user demands
- Hidden scenarios
Do we still need GPU virtualizations?

- Share GPUs in datacenter
- Different end-user demands
- Hidden scenarios
Do we still need GPU virtualizations?

- Share GPUs in datacenter
- Different end-user demands
- Hidden scenarios
GPU Virtualization Challenges

- Diverse hardware
- Undocumented APIs
- Closed-source GPUs and drivers
- Deep graphics stack
- Coupled layers
- Significant overheads
- Limited flexibility
GPU Virtualization Comparisons

Device emulation
*Synthesizes host graphics operations*

API remoting
*Forwards graphics API calls To external graphics stack*

Mediated-passthrough
*Dedicates a set of contexts*

Passthrough
*Provides exclusive access*
## GPU Virtualization Comparisons

<table>
<thead>
<tr>
<th>Method</th>
<th>Front-end Description</th>
<th>Back-end Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device emulation</td>
<td>Synthesizes host graphics operations</td>
<td></td>
</tr>
<tr>
<td>API remoting</td>
<td>Forwards graphics API calls To external graphics stack</td>
<td></td>
</tr>
<tr>
<td>Mediated-passthrough</td>
<td>Dedicates a set of contexts</td>
<td></td>
</tr>
<tr>
<td>Passthrough</td>
<td>Provides exclusive access</td>
<td></td>
</tr>
</tbody>
</table>

### Parameters
- Performance
- Fidelity
- Multiplexing
- Interposition
- Complexity
# GPU Virtualization Comparisons

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance</th>
<th>Fidelity</th>
<th>Multiplexing</th>
<th>Interposition</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Device emulation</strong></td>
<td>✗ ✗</td>
<td>✗ ✗</td>
<td></td>
<td>✔ ✔</td>
<td>✗ ✗</td>
</tr>
<tr>
<td><em>Synthesizes host graphics operations</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>API remoting</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Forwards graphics API calls To external graphics stack</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mediated-passthrough</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Dedicates a set of contexts</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Passthrough</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Provides exclusive access</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# GPU Virtualization Comparisons

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance</th>
<th>Fidelity</th>
<th>Multiplexing</th>
<th>Interposition</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device emulation</td>
<td><img src="x" alt="x" /> <img src="x" alt="x" /></td>
<td><img src="x" alt="x" /> <img src="x" alt="x" /></td>
<td><img src="checkmark" alt="checkmark" /> <img src="checkmark" alt="checkmark" /></td>
<td><img src="checkmark" alt="checkmark" /> <img src="checkmark" alt="checkmark" /></td>
<td><img src="x" alt="x" /> <img src="x" alt="x" /></td>
</tr>
<tr>
<td>Synthesizes host graphics operations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance</th>
<th>Fidelity</th>
<th>Multiplexing</th>
<th>Interposition</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>API remoting</td>
<td><img src="x" alt="x" /> <img src="x" alt="x" /></td>
<td><img src="x" alt="x" /> <img src="x" alt="x" /></td>
<td><img src="x" alt="x" /> <img src="x" alt="x" /></td>
<td><img src="checkmark" alt="checkmark" /> <img src="checkmark" alt="checkmark" /></td>
<td><img src="checkmark" alt="checkmark" /></td>
</tr>
<tr>
<td>Forwards graphics API calls To external graphics stack</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance</th>
<th>Fidelity</th>
<th>Multiplexing</th>
<th>Interposition</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mediated-passthrough</td>
<td><img src="checkmark" alt="checkmark" /> <img src="checkmark" alt="checkmark" /></td>
<td><img src="checkmark" alt="checkmark" /> <img src="checkmark" alt="checkmark" /></td>
<td><img src="checkmark" alt="checkmark" /> <img src="checkmark" alt="checkmark" /></td>
<td><img src="checkmark" alt="checkmark" /> <img src="checkmark" alt="checkmark" /></td>
<td><img src="x" alt="x" /> <img src="x" alt="x" /></td>
</tr>
<tr>
<td>Dedicates a set of contexts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance</th>
<th>Fidelity</th>
<th>Multiplexing</th>
<th>Interposition</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passthrough</td>
<td><img src="checkmark" alt="checkmark" /> <img src="checkmark" alt="checkmark" /></td>
<td><img src="checkmark" alt="checkmark" /> <img src="checkmark" alt="checkmark" /></td>
<td><img src="checkmark" alt="checkmark" /> <img src="checkmark" alt="checkmark" /></td>
<td><img src="checkmark" alt="checkmark" /> <img src="checkmark" alt="checkmark" /></td>
<td><img src="checkmark" alt="checkmark" /> <img src="checkmark" alt="checkmark" /></td>
</tr>
<tr>
<td>Provides exclusive access</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## GPU Virtualization Comparisons

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance</th>
<th>Fidelity</th>
<th>Multiplexing</th>
<th>Interposition</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device emulation</td>
<td>✗ ✗</td>
<td>✗ ✗</td>
<td>✗ ✗</td>
<td>✔ ✔</td>
<td>✗ ✗</td>
</tr>
<tr>
<td><em>Synthesizes host graphics operations</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>API remoting</td>
<td>✗</td>
<td>✗</td>
<td>✗ ✗</td>
<td>✗ ✗</td>
<td>✔</td>
</tr>
<tr>
<td><em>Forwards graphics API calls To external graphics stack</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mediated-passthrough</td>
<td>✔ ✔</td>
<td>✔ ✔</td>
<td>✔ ✔</td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td><em>Dedicates a set of contexts</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Passthrough</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✗</td>
</tr>
<tr>
<td><em>Provides exclusive access</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## GPU Virtualization Comparisons

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance</th>
<th>Fidelity</th>
<th>Multiplexing</th>
<th>Interposition</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Device emulation</strong></td>
<td>✗ ✗</td>
<td>✗ ✗</td>
<td>✗</td>
<td>✔ ✔</td>
<td>✗ ✗</td>
</tr>
<tr>
<td>Synthesizes host graphics operations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>API remoting</strong></td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗ ✗</td>
<td>✔</td>
</tr>
<tr>
<td>Forwards graphics API calls</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To external graphics stack</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mediated-passthrough</strong></td>
<td>✔ ✔</td>
<td>✔ ✔</td>
<td>✗</td>
<td>✔ ✔ ✔</td>
<td>✗</td>
</tr>
<tr>
<td>Dedicates a set of contexts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Passthrough</strong></td>
<td>✔ ✔</td>
<td>✔ ✔</td>
<td>✗ ✗</td>
<td>✗ ✗</td>
<td>✗ ✗</td>
</tr>
<tr>
<td>Provides exclusive access</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Front-end

Back-end
# GPU Virtualization Comparisons

<table>
<thead>
<tr>
<th></th>
<th>Performance</th>
<th>Fidelity</th>
<th>Multiplexing</th>
<th>Interposition</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device emulation</td>
<td>✗ ✗</td>
<td>✗ ✗</td>
<td>✗ ✗</td>
<td>✗ ✗</td>
<td>✗ ✗</td>
</tr>
<tr>
<td>Synthesizes host graphics operations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>API remoting</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗ ✗</td>
<td>✗</td>
</tr>
<tr>
<td>Forwards graphics API calls To external graphics stack</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mediated-passthrough</td>
<td>✅✅</td>
<td>✅✅</td>
<td>✅</td>
<td>✗ ✗</td>
<td>✗</td>
</tr>
<tr>
<td>Dedicates a set of contexts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Passthrough</td>
<td>✅✅</td>
<td>✅✅</td>
<td>✗ ✗</td>
<td>✗ ✗</td>
<td>✅✅</td>
</tr>
<tr>
<td>Provides exclusive access</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPU Virtualization Examples</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>--</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Device emulation</strong></td>
<td>M. Dowty, VMware <strong>SVGA</strong>, <strong>SIGOPS-OSR ’09</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>API remoting</strong></td>
<td>J. Duato, <strong>rCUDA</strong>, HiPC’10,11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>G. Giunta, <strong>gVirtuS</strong>, European Conference on Parallel Processing’10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mediated-passthrough</strong></td>
<td>AMD MxGPU (<strong>FirePro</strong>), VMworld’15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NVIDIA <strong>GRID</strong> vGPU, 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>KVMGT (Intel <strong>GVT-g</strong>), 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Passthrough</strong></td>
<td>Amazon Elastic Compute Cloud (<strong>AWS EC2</strong>)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Front-end</td>
<td>Back-end</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Device emulation</strong></td>
<td>Exposes a native device model to VMs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>API remoting</strong></td>
<td>Forwards commands to GPU virtual aggregator</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Mediated-passthrough</strong></td>
<td>Passes-through some operations (I/O requests) to hardware</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Similar approaches when virtualizing at hypervisor-level
Full-virtualization vs. Para-virtualization

Para-virtualization

*Split device model*

Apps
vGPU driver API
Front End

Back End
GPU driver API
GPU driver
GPU

Full-virtualization

*Trap-and-emulate*

Apps
GPU driver API
GPU driver

Device model
Hypervisor
GPU
Full-virtualization vs. Para-virtualization

Para-virtualization

*Split device model*

- Apps
- vGPU driver API
- Front End

- Back End
- GPU driver API
- GPU driver
- GPU

Full-virtualization

*Trap-and-emulate*

- Apps
- GPU driver API
- GPU driver

- Device model
- Hypervisor
- GPU
Full-virtualization vs. Para-virtualization

Para-virtualization

*Split device model*

- Apps
- vGPU driver API
- Front End

- Back End
- GPU driver API
- GPU

- Performance
- Fidelity
- Multiplexing
- Interposition

Full-virtualization

*Trap-and-emulate*

- Apps
- GPU driver API
- GPU driver

- Device model
- Hypervisor
- GPU

- GPU driver API
- Hypervisor
Full Virtualization: A Reasonable Goal?

Full virtualization

Trap-and-emulate

Apps

Device model

GPU driver API

Hypervisor

GPU driver

Full-featured vGPU
(3D acceleration)

Strong isolation

Slow performance

Hard to map different GPUs
Full Virtualization: A Reasonable Goal?

Full-featured vGPU
(3D acceleration)

Safe
VMM
Hardware

Full-virtualization
Trap-and-emulate

Apps
GPU driver API
GPU driver

Device model
Hypervisor
GPU

Strong isolation

Device Model
Hard to map different GPUs

Slow performance

Full Virtualization

GPU Virtual Scheduler

Driver

VM

.App

App

App

Driver

VM

.App

App

Driver

vGPU Model

vGPU Model

GPU/vm

GPU Access Aggregator

Shadow Channel

Shadow Page Table

Shadow Channel

Shadow Page Table

GPU Virtual Scheduler

Full Virtualization: A Reasonable Goal?

Full-featured vGPU
(3D acceleration)

Safe
VMM
Hardware

Device Model
Hard to map different GPUs

Full Virtualization

Full-featured vGPU
(3D acceleration)

Safe
VMM
Hardware

Device Model
Hard to map different GPUs

Full Virtualization: A Reasonable Goal?

Full-featured vGPU
(3D acceleration)

Safe
VMM
Hardware

Device Model
Hard to map different GPUs
GPUvm Overview

- Access aggregator
GPUvm Overview

- Access aggregator
GPUvm Overview

- Access aggregator
GPUvm Overview

• Access aggregator
• Shadow channel
  – Mapped by a virtual channel
• Shadow page table
GPUvm Overview

- Access aggregator
- Shadow channel
  - Mapped by a virtual channel
- Shadow page table
GPUvm Overview

- Access aggregator
- Shadow channel
  - Mapped by a virtual channel
- Shadow page table
GPUvm Overview

- Access aggregator
- Shadow channel
  - Mapped by a virtual channel
- Shadow page table
GPUvm Overview

- **Access aggregator**
- **Shadow channel**
  - Mapped by a virtual channel
- **Shadow page table**
- **Virtual scheduler**
  - FIFO
  - CREDIT
  - BAND (bandwidth-aware non-preemptive device)
Why GPUvm?

• Open-source
• Overheads
  – FV (36x) PV (1.9x)
• Good open architecture
  – Decoupled components
  – Native device model, virtual MMIO, shadow channels, shadow page tables, virtual schedulers
• Not-so-good aspects
  – Interposes guest access to memory-mapped resources
  – Shadows expensive resources
• Trade-off of *hypervisor-level* full-virtualization
GPUvm Optimizations

• Sync virtual & shadow channels
  – Intercept data accesses
  – BAR3 remapping
    • BAR3 accesses are passed-through
• Sync guest & shadow page tables
  – GPU-side page faults
  – Lazy shadowing
    • Updates shadow page tables only when referenced

MMIO through PCIe base address register
GPUvm Optimizations

• Sync virtual & shadow channels
  – Intercept data accesses
  – BAR3 remapping
    • BAR3 accesses are passed-through
• Sync guest & shadow page tables
  – GPU-side page faults
  – Lazy shadowing
    • Updates shadow page tables only when referenced
Testbed

• Specific hardware
  – NVIDIA Quadro 6000 NVC0
  – GF100GL vs. GF100 (GTX 480) (different region addresses)

• Specific software
  – Fedora 16 (Kernel 3.6.5)
  – Xen HVM (4.2.0)
  – Gdev (commit 605e69e7)
  – GCC 4.6.3
  – NVCC 4.2
  – Boost 1.4.7
Performance

- **BAR3 remapping**
  - 1.6x speed-up
  - Fails for some benchmarks

- **Lazy shadowing**
  - 1.2x speed-up
  - Fails for some benchmarks

- **Overhead**
  - up to 737x, 232x on average

- **7.4x Boot slowdown**

<table>
<thead>
<tr>
<th></th>
<th>hotspot</th>
<th>lud</th>
<th>srad</th>
<th>mmul</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRITE bytes</td>
<td>659,664</td>
<td>662,544</td>
<td>666,784</td>
<td>660,832</td>
</tr>
<tr>
<td>Original WRITE bytes</td>
<td>6,736</td>
<td>7,240</td>
<td>6,352</td>
<td>6,672</td>
</tr>
</tbody>
</table>
Performance

• BAR3 remapping
  – 1.6x speed-up
  – Fails for some benchmarks
• Lazy shadowing
  – 1.2x speed-up
  – Fails for some benchmarks
• Overhead
  – up to 737x, 232x on average
• 7.4x Boot slowdown

<table>
<thead>
<tr>
<th>WRITE bytes</th>
<th>hotspot</th>
<th>lud</th>
<th>srad</th>
<th>mmul</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRITE bytes</td>
<td>659,664</td>
<td>662,544</td>
<td>666,784</td>
<td>660,832</td>
</tr>
<tr>
<td>Original WRITE bytes</td>
<td>6,736</td>
<td>7,240</td>
<td>6,352</td>
<td>6,672</td>
</tr>
</tbody>
</table>
Runtime Breakdown

- **Init phase**
  - Major overhead
  - Major cost

- **Optimizations**
  - Overheads of *Init*, *MemAlloc* and *Close* ↓ ↓ ↓

<table>
<thead>
<tr>
<th>Needle</th>
<th>Naive</th>
<th>BAR-remap</th>
<th>Shadow</th>
<th>Optimized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init</td>
<td>850x</td>
<td>150x</td>
<td>750x</td>
<td>60x</td>
</tr>
<tr>
<td>MemAlloc</td>
<td>3,878x</td>
<td>3,135x</td>
<td>287x</td>
<td>21x</td>
</tr>
<tr>
<td>Close</td>
<td>1,260x</td>
<td>1,075x</td>
<td>200x</td>
<td>165x</td>
</tr>
</tbody>
</table>
Runtime Breakdown

- **Init phase**
  - Major overhead
  - Major cost

- **Optimizations**
  - Overheads of *Init*, *MemAlloc* and *Close* ↓ ↓ ↓

<table>
<thead>
<tr>
<th>Needle</th>
<th>Naive</th>
<th>BAR-remap</th>
<th>Shadow</th>
<th>Optimized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init</td>
<td>850x</td>
<td>150x</td>
<td>750x</td>
<td>60x</td>
</tr>
<tr>
<td>MemAlloc</td>
<td>3,878x</td>
<td>3,135x</td>
<td>287x</td>
<td>21x</td>
</tr>
<tr>
<td>Close</td>
<td>1,260x</td>
<td>1,075x</td>
<td>200x</td>
<td>165x</td>
</tr>
</tbody>
</table>
Runtime Breakdown

- Some GPU optimization features not help much
  - E.g. pipelining MemCpy
- Other phases
  - *Launch*: Almost not influenced by optimizations
  - *DtoH*: Trivial overheads
Fairness

- Is BAND more fair?
  - Not always
  - 6% worse in 4VM case

\[ F = \frac{\text{Max} - \text{Min}}{\text{Avg}} \]
Throughput

- **BAND**
  - 42.5% throughput loss in 8VM case
- **Compared with CREDIT**
  - 8% in 8VM case
  - Overheads of *Init & Close*
  - Inserted idle phases
Conclusion

- Full-virtualization benefits
  - Compatibility, interposition, isolation
- Performance
  - MMIO interceptions
  - Resource shadowing
  - *Two optimizations*
- Fairness and throughput
  - Decoupled scheduler module

<table>
<thead>
<tr>
<th></th>
<th>Performance (avg)</th>
<th>Throughput loss (8VM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our testbed</td>
<td>&gt; 200x slowdown</td>
<td>≈ 40%</td>
</tr>
<tr>
<td>GPUvm paper</td>
<td>&gt; 33x slowdown</td>
<td>≈ 27%</td>
</tr>
</tbody>
</table>

- If not impossible…
  - Further improve the components? vMMIO, schedulers
  - Leverage new hardware functionalities? NVIDIA Pascal page faults, SRIOV
    Baumann, *Hardware is the new software*, HotOS'17