GPU Virtualization on VMware’s Hosted I/O Architecture

Micah Dowty
Jeremy Sugerman

USENIX WIOV 2008
Contents

• GPUs are hard
• But GPU virtualization is worth the trouble
• How to virtualize a GPU?
• VMware’s virtual GPU
• Conclusions

• In the paper:
  – Details on our implementation
  – Benchmarks, analysis
What is a GPU, anyway?

- Video playback, 2D graphics, drawing triangles and rectangles and lines...

- Computation.
How much computation?

NVIDIA GeForce GTX 280:
1.4 billion transistors

Intel Core 2 Duo:
291 million transistors

Source: AnandTech review of NVidia GT200
Programmable 3D Pipeline

- **Texture Data**
- **Vertex Data**
  - Position
  - Material
  - Texture coordinate
- **Vertex Pipeline**
- **Rasterization**
  - Plot triangles, lines
  - Interpolate vertices
- **Pixel Pipeline**
- **Framebuffer**

(State of the art circa 2002...)
Unique challenges

• API
  – Not quite read(), write(), select()...
  – Multiple competing APIs
  – Hundreds of entry points

• Programmable
  – Every GPU driver is also a compiler
  – Each API includes a language spec
Unique challenges

• Hardware specs
  – Diverse, changes frequently
  – Closely guarded secret*
  – Speed vs. portability

• Hardware state
  – Up to gigabytes of data
  – Highly device-specific format
  – In-progress DMA and computation

* With a few notable exceptions.
What are GPUs good for?

- Desktop Apps
  - Entertainment
  - CAD
  - Multimedia
  - Productivity

- Desktop GUIs
  - Quartz Extreme
  - Vista Aero
  - Compiz
GPUs in the Data Center

- Server-hosted Desktops
- GPGPU
API Remoting

Guest

Host

App

App

App

API

OpenGL / Direct3D Redirector

RPC Endpoint

User-level

API

OpenGL / Direct3D

GPU Driver

Kernel

GPU

Hardware
Device Emulation

- **Host**
  - GPU Emulator
  - Resource Management
  - Shader / State Translator
  - Rendering Backend
  - OpenGL / Direct3D
  - GPU Driver
  - Kernel

- **Guest**
  - Apps
  - Virtual GPU Driver
  - Virtual GPU
  - Kernel
  - API
  - OpenGL / Direct3D

- **Shared System Memory**
- **Virtual HW**
- **Kernel**
- **API**
- **Virtual GPU**
Fixed pass-through
Mediated pass-through

Virtual Machine

App  App  App

API

GPU Driver

Pass-through GPU

Emulation

GPU Resource Manager

Physical GPU
GPU Virtualization Taxonomy

API Remoting  \hspace{1cm} Device Emulation

Front-end \hspace{1cm} Hybrid (Driver VM) \hspace{1cm} Back-end

Fixed Pass-through 1:1  \hspace{1cm} Mediated Pass-through 1:N
VMware’s Virtual GPU

• **Compatibility**
  – Any physical GPU
  – Any guest driver stack
  – Adjustable capability exposure
  – No direct access to GPU memory

• **Efficiency**
  – Flexible guest memory management
  – Few copies
  – Asynchronous rendering
VMware SVGA II

PCI Device
- BAR 0
- BAR 1
- BAR 2
- IRQ

I/O Space
- Index
- Value
- IRQ Status

Sync. Registers
- Video Mode
- Device caps
- IRQ Config
- GMR Config
- ...

Virtual VRAM
- 2D Framebuffer
- General Purpose DMA memory
- ...

GMR Page Tables
- GMR 0
- ...
- VRAM GMR

System Memory
- User-defined GMR

FIFO Memory
- FIFO Pointers
- Async. regs
- 3D Caps
- Command FIFO Buffer

Virtual DMA Engine

MKS HostOps

Physical VRAM
Virtual Graphics Stack

- App
- VMware SVGA Driver
- SVGA FIFO / Registers
- SVGA Device
- Guest Mem
- Guest VRAM
- DMA Engine
- MKS / HostOps Dispatch
- SVGA GMR
- State Translator
- Shader Program Translator
- 3D Drawing Path
- Surface Abstraction
- DMA Engine
- GPU API / Driver
- GPU
- 2D Compositing
- 3D Rendering
Evaluation

- Applications
- Microbenchmarks
- VMware Fusion 2.0, VMware Workstation 6.5, Parallels Desktop 3.0, SwiftShader
- Mac Pro, 8-core 2.8 GHz
- ATI Radeon HD2600
Application Benchmarks

<table>
<thead>
<tr>
<th>Application</th>
<th>Resolution</th>
<th>FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTHDRIBL</td>
<td>1280 × 1024</td>
<td>22</td>
</tr>
<tr>
<td>RTHDRIBL</td>
<td>640 × 480</td>
<td>27.5</td>
</tr>
<tr>
<td>Half Life 2: Episode 2</td>
<td>1600 × 1200</td>
<td>22.2</td>
</tr>
<tr>
<td>Half Life 2: Episode 2</td>
<td>1024 × 768</td>
<td>32.2</td>
</tr>
<tr>
<td>Civilization 4</td>
<td>1600 × 1200</td>
<td>18</td>
</tr>
<tr>
<td>Max Payne 2</td>
<td>1600 × 1200</td>
<td>42</td>
</tr>
</tbody>
</table>
Summary

• GPU Virtualization is an important problem
• Room for improvement in implementation completeness and performance...
• But we can already run interactive apps that could never be virtualized before
• Virtual GPU preserves portability + isolation
Future Work

• Pass-through techniques
  – Fixed and Mediated
  – Can be complementary to Virtual GPU

• Continued improvements
  – Performance and functionality
  – At all layers of driver stack

• Virtualization-aware GPU benchmarks
Questions?

• micah@vmware.com