VMM-Independent Graphics Acceleration

H. Andrés Lagar-Cavilla, U of Toronto
andreslc@cs.toronto.edu
Niraj Tolia (CMU), Eyal de Lara (Toronto),
M. Satyanarayanan (CMU)
Why Virtualize 3D Acceleration?

Two simultaneous trends

- VMs out of the server room
- Client apps going 3D

And we only have software rendering (Mesa)
Virtualization of Client Apps

- Soulpads
- The Collective
- Internet Suspend/Resume
- Virtual Appliances
- Moka5, MojoPac, BlackDog, ...
The World Is Going 3D
Why Is 3D Virtualization Hard?

3D vendors compete through HW diversity
- Lack of unifying hardware abstraction
- Closed specs

Open HW abstractions simplify virtualization:
- Network -> Ethernet Frame
- Block Devices -> BIO request
- SCSI drives -> SCSI command packet
- ....

How could we ever write 3D applications?
3D Rendering APIs

De facto unifying software abstraction
Developer gets vendor independence

Two main APIs
- OpenGL
- Direct3D

OpenGL
- Cross-platform
VMGL: Virtualizing OpenGL

Provides 3D HW acceleration to applications running inside virtual machines

- GPU independent
- VMM independent
- Guest OS independent
- Suspend and resume capable

- 87% or better of native HW acceleration
- Two orders of magnitude better than Mesa
VMGL Design

- API virtualization
  - GPU vendor independence

OpenGL: cross-platform API
  - Guest OS independence

Network Communication
  - VMM independence
OpenGL Apps in X11 Systems

- Application
- Vendor OpenGL
- 3D
- GPU
- X Server
- 2D
VMGL Apps in X11 Guest VMs

- Application
- X Server
- VMGL

Guest

VM Viewer: VNC, SDL

VMGL Stub

GL Commands

Vendor

Open GL

Host

GPU
Implementation Aspects

- OpenGL API v1.5
  - Shaders through extensions

- Efficient GL network transport

- 3D and 2D output composing in VM viewer

- Suspend/Resume implementation

- Xen-specific: Domain 0 drivers
Implementation Aspects

- OpenGL API v1.5
  - Shaders through extensions

- Efficient GL network transport

- 3D and 2D output composing in VM viewer

- Suspend/Resume implementation

- Xen-specific: Domain 0 drivers
Efficient GL Transport

Transport over network
- VMM Independence

WireGL / Chromium
- Intended for tiled rendering

Only send updates that “matter”
- glTextureXY only when texture visible

Combine, reorder and buffer commands
- glRotate + glTranslate -> Single matrix transformation
Output Composing in VM Viewer

3D & 2D output coming from different sources

Extension in VM's X server tells viewer about 3D windows
- Position
- Size
- Clipping
Suspend / Resume

Think each GL app as a GL device
- Runtime: keep track of OpenGL state
- Suspend: “freeze” GL device (trivial)
- Resume: flush state to new GL stub

OpenGL state is GPU independent
- Suspend/resume across different GPUs

OpenGL state is bounded
- See experiments
VMGL Suspend / Resume State

Windows
- Visual bits
- Binding to window manager extension

GL Contexts
- Context data: fog, transformations...
- Textures: pixmap, clamp mode
- Display Lists: verbatim unrolling
VMGL Evaluation

VMGL: OpenGL Virtualization

VMMs
- Xen – Paravirtual (results unless otherwise noted)
- Xen – HVM
- VMware Workstation

OSs
- Linux 2.6.16.29
- OpenSolaris 10 rel 06/06
- FreeBSD rel 6.1

Hardware
- ATI X600, Intel Dual Core 2.4 GHz, VT, 2GB Ram
Workloads

Quake 3

Unreal 2004

Enemy Territory

Mplayer
Performance (FPS)
Performance (FPS)

Native
Mesa
87% or better of native performance
VMM Portability (FPS)

- VMM and VM type independent
Guest OS Portability (FPS)

Quake 3 on VMware Workstation

- VMGL easily ported to other X11-based OSs
**Suspend Resume Performance**

- State size bounded
- Also across GPUs from different vendors
Wrapping UP

VMGL: OpenGL virtualization

Enable intersection of two growing trends
- Virtualization
- 3D Graphics

GPU/vendor independence
VMM independence
Guest OS independence

More eval & details in paper
TODO

VMM-specific improvements
• Shared memory transport

Windows
• Code porting
• Window Manager hooks
• Direct3D support via translation layers
THANKS

Demo
Q&A

2549 Downloads and counting:
www.cs.toronto.edu/~andreslc/vmgl/
andreslc@cs.toronto.edu
BACKUP
Xen Domain 0 GPU Drivers

ATI & Nvidia:
- GPU Mem mapping in user-space GL lib

Oblivious to Xen additional indirection
- Virtual -> Physical (VM) -> Machine
- Even for domain 0

Fix open source portion of driver

Use Xen-paravirt mem mapping functions
Performance (FPS)

87% or better of native performance
Concurrent Execution

![Bar Chart](chart.png)

- **Quake 3**: VMGL vs Native
- **Enemy**: VMGL vs Native
- **Unreal**: VMGL vs Native
- **Mplayer**: VMGL vs Native

Legend:
- Red: VMGL
- Green: Native