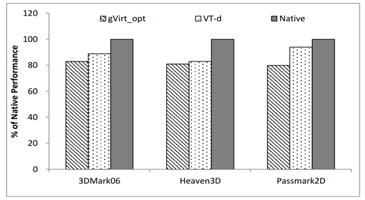
Native GPU Performance on Virtual Machines

VM Migration with PCI Passthrough, Virtualizing at the Library level

Daniel Kats, Graham Allsop, Eyal DeLara

Motivation

- Increased demand for GPU applications to run in a virtualized environment
- Current solutions use mediated passthrough which provide no better than 85-90% of native GPU performance
- PCI passthrough allows for upwards of 99% of native GPU performance
- Want native performance, cross-platform solution, and VM migration



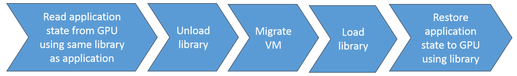
Native vs Mediated Passthrough Performance

Problem Outline

- GPU state is not captured in typical migrations
- GPU designs are closed and proprietary, so internal state cannot be easily extracted or replicated on another machine
- Graphics drivers do not respond positively to GPUs being disconnected while they are loaded
- Applications working on the GPU are not pleased when the graphics driver is unloaded (or crashes)

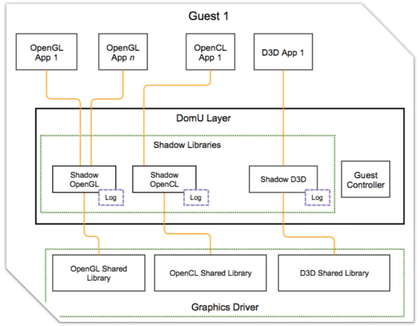
Solution Outline

- Instead of dealing with the GPU directly, work at the library level

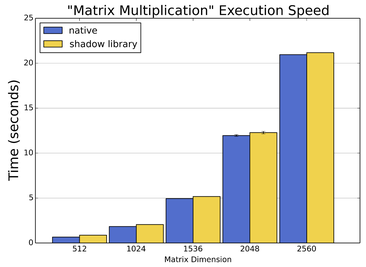


Shadow Library and API Interception

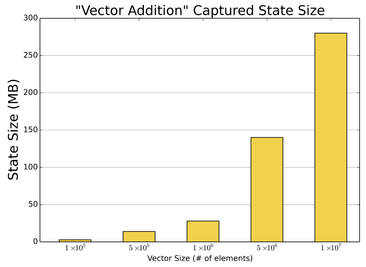
- Give VMs direct access to the GPU using PCI passthrough
- Re-route calls to libraries which use GPU to shadow library using API intercept
- Shadow library stores own minimal representation of GPU state which is updated on every intercepted call
- Disconnect and reconnect real libraries using dynamic loading from shadow library
- VM host controls migration by signaling for shadow library disconnect/reconnect
- On migration start, shadow library extracts state from GPU using library API calls then unloads the real library
- On migration resume, shadow library reloads the real library and restores GPU state using library API calls



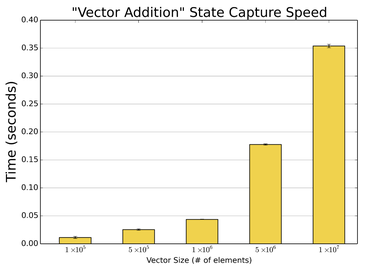
Results



Overhead is linear in problem size and consists mostly of initialization which amortizes well over long program runs



State capture time is very fast and is linear in problem size



State size is almost entirely input data and scales linearly with problem size

Conclusion

- Complete GPU state capture and restore is possible at the library level
- State capture and restore is very fast with under 1% overhead
- All of this without changing the application binary

Future Work

- Use same approach for OpenGL and Direct3D libraries
- Support for live migration under Xen VMM

References

- Khronos OpenCL Working Group. (2015). In A. Munshi (Ed.), The OpenCL Specification. Retrieved from https://www.khronos.org/registry/cl/specs/opencl-2.0.pdf
- Microsoft Corporation. (1999). In Microsoft Portable Executable and Common Object File Format Specification. Retrieved from http://goo.gl/uciTqK
- Unix System Laboratories. (n.d.). Executable and Linkable Format. Retrieved from http://flint.cs.yale.edu/cs422/doc/ELF_Format.pdf