Gdev: First-Class GPU Resource Management in the Operating System

Shinpei Kato
Nagoya University

Michael McThrow Carlos Maltzahn Scott Brandt
UC Santa Cruz
GPUs embrace “many cores”.
Graphics Processing Unit (GPU)

- 2008: 250 cores (Tesla)
- 2010: 500 cores (Fermi)
- 2012: 1500 cores (Kepler)

Device Memory

L2 Cache

Main Memory

CPU
Performance Trend

Single Precision Performance

GFLOPS

NVIDIA GPU
Intel CPU


GFLOPS / Watt

NVIDIA GPU
Intel CPU

GPUs Suit Science

Traffic Simulation

The number of agents

Simulation time (ms)

- GPU (simple)
- GPU (data optimized)
- GPU (fully optimized)
- CPU

GTX 560 Ti (192 cores)

1000x
Not Yet “General-Purpose”

GPU Resource Management
Gdev

- **New approach** to GPU resource management
  - Allows the OS as well as user-space applications to use GPUs.

- **New functions** of GPU resource management
  - Shared device memory (IPC)
  - Data swapping
  - System-level virtualization

- **Open-source implementation**
Outline

• Motivation
• Approach
• GPU Resource Management
• Evaluation
• Conclusion
Traditional Naïve Approach

User Space

- Application
  - Runtime
    - Other Runtimes
      - Inteligence
    - Command (ioctl)
  - I/O Request
  - Command (ioctl)
  - Non-privileged Applications

OS

- Command (ioctl)
  - Device Driver
    - Not efficient!
    - I/O Request
    - Not secure!

Device

- GPU
Unified, OS-oriented approach to GPU resource management!
Outline

• Motivation
• Approach
• GPU Resource Management
• Evaluation
• Conclusion
Shared Device Memory

Context 1
Virtual Address Space

Context 2
Virtual Address Space

Context 3
Virtual Address Space

addr A

addr B

addr C

addr S

Physical Device Memory Space

New API:
- cuShmGet()
- cuShmAt()
- cuShmDt()
- cuShmCtl()
E.g., Dataflow (2x2 Tree)

No Shared Memory

Context 1

A[] B[]

P[]

copy

context

Host Memory

P[]

Q[]

copy

Context 2

C[] D[]

Q[]

copy

Context 3

P[]

Q[]

X[]

P[]xQ[]=X[]

Context 1

A[]xB[]=P[]

Context 2

C[]xD[]=Q[]

Context 3

Device Memory

P[]

Q[]

X[]

Shared Memory IPC

Context 1

A[] B[]

P[]

context

Host Memory

P[]

Q[]

Context 2

C[] D[]

Q[]

Context 3

X[]
Data Swapping

Host Memory

- New Data
- Stall...
- Evicted

Device Memory

- New Data

I/O
Data Swapping (Enhanced)

Host Memory

Device Memory

New Data

Evicted

I/O

Temporal

Evicted
GPU Virtualization

Virtual GPU
/dev/gdev0
/dev/gdev1
/dev/gdev2
/dev/gdev3
...

Virtual GPU

Virtual GPU

Physical GPU
/dev/dri/card0
(real device file)
Existing GPU Schedulers
Queue and dispatch [Kato ATC11] [Kato RTSS11]

Load unbalanced!
Bandwidth-aware non-preemptive device (BAND) Scheduler

Load balanced
Outline

- Motivation
- Approach
- GPU Resource Management
- Evaluation
- Conclusion
Experimental Setup

- Linux kernel 2.6.39
- NVIDIA GeForce GTX 480
- Intel Core 2 Extreme QX9650
- NVIDIA CUDA Compiler 4.0 and GCC 4.4.6
- Benchmarks & Applications:
  - Rodinia benchmark [Che et al, IISWC’09]
  - eCryptfs encrypted filesystem
  - FAST database search [Kim et al, SIGMOD’10]
  - PTask dataflow benchmarks [Rossbach et al, SOSP’11]
Runtime and Driver Choice

NVIDIA

Gdev/User

Gdev

NVIDIA

Gdev Runtime

NVIDIA Runtime

Gdev User

Gdev Runtime

Gdev Library

NVIDIA Driver

Gdev Driver

Gdev Runtime

Gdev Driver

NVIDIA Driver
Basic Performance

- NVIDIA
- Gdev/User
- Gdev

Relative Speed

- ~25%
- ~29%

Benchmarks:
- LOOP
- MADD
- MMUL
- CPY
- PINCPY
- BP
- BFS
- HW
- HS
- LUD
- NN
- NW
- SRAD
- SRAD2
eCryptfs Read&Write Throughput

**Read throughput**

- KGPU & NVIDIA
- KGPU & Gdev/User
- Gdev
- CPU

**Write throughput**

- KGPU & NVIDIA
- KGPU & Gdev/User
- Gdev
- CPU

**Gdev Module**

- CUDA
- Runtime
- Driver

**Daemon**

- upcall
- CUDA
- ioctl

**Runtime**

- Module
- Driver

**Data Size (bytes)**

- 1M
- 2M
- 4M
- 8M
- 16M
- 32M
- 64M
- 128M
- 256M

**Throughput (MB/s)**

- 0
- 40
- 80
- 120
- 160
- 200
- 240
- 280

**CPU**

- KGPU & NVIDIA
- KGPU & Gdev/User
- Gdev

**Sun et. al. 2012**

**CUDA**

- ioctl
- Module
- Runtime
- Driver

**Throughput (MB/s)**

- 1M
- 2M
- 4M
- 8M
- 16M
- 32M
- 64M
- 128M
- 256M
Impact of Shared Device Memory

![Bar chart showing relative speed for different benchmarks and configurations. The chart compares NVIDIA, Gdev w/o shm, and Gdev, with a significant improvement indicated by ~50% at certain benchmarks and resolutions.]
Impact of Data Swapping

Multiple DB search tasks

- NVIDIA
- Gdev/User
- Gdev w/o swp
- Gdev

Makespan (ms)

Number of Competing Contexts

NVIDIA fails

Gdev/User fails

~500ms
Virtual GPU Isolation

No scheduling (FIFO)

Xen VM Policy (Credit)

Gdev Policy (BAND)

~7%
Outline

• Motivation
• Approach
• GPU Resource Management
• Evaluation
• Conclusion
Concluding Remarks

Gdev is an OS approach to first-class GPU resource management.

- GPUs can be used by the OS.
- GPUs can be protected by the OS.
- GPUs can be multi-tasked by the OS.

Compromising basic performance to some extent.
Concluding Remarks

Gdev is open-source.
Facilitate systems research.
Visit http://sys.ertl.jp/gdev/.

What’s up-to-date:
- RAID6 erasure coding acceleration.
- Dynamic power management.
- Zero-copy between I/O devices and GPUs.
Thank You!

Questions?