TimeGraph: GPU Scheduling for Real-Time Multi-Tasking Environments

Shinpei Kato*, Karthik Lakshmanan*, Raj Rajkumar*, and Yutaka Ishikawa**

* Carnegie Mellon University  ** The University of Tokyo
Graphics Processing Unit (GPU)

NVIDIA GPU
GeForce GTX 480

480 simple cores

L1 L1 L1 L1 L1 L1

L2 Cache

Device Memory

CPU
Host Memory
Peak Performance

GFLOPS

NVIDIA GPU

Intel CPU

7900 GTX

8800 GTX

9800 GTX

GTX 280

GTX 285

GTX 480

GTX 580

E4300

E6850

Q9650

X7460

980 XE

2006/3/4

2007/12/14

2009/9/24

2011/7/6
Peak Performance “per Watt”
General-Purpose Computing on GPU (GPGPU)

3-D On-line Game  Autonomous Driving  Virtual Reality

3-D Interface  Computer Vision  Scientific Simulation
Outline

1. Introduction
2. What’s Problem
3. Our Solution – “TimeGraph”
4. Evaluation
5. Summary
GPU Is Command-Driven

- **CMD_HtoD**
  - Host Memory
  - GPU Code
  - Input Data
  - Device Memory
  - Copy

- **CMD_HtoD**
  - Host Memory
  - GPU Code
  - Input Data
  - Device Memory
  - Copy

- **CMD_LAUNCH**
  - Host Memory
  - GPU Code
  - Input Data
  - Device Memory
  - Copy

- **CMD_DtoH**
  - Host Memory
  - GPU Code
  - Input Data
  - Output Data
  - Device Memory
  - Copy
Multi-Tasking Problem

- High-priority task
- Low-priority task
- GPU driver
- GPU command

CPU

GPU

Blocked

Blocked

Graph showing the scheduling of high-priority and low-priority tasks on both the CPU and GPU over time, with blocks indicating blocked periods.
Impact of Interference

Observe Frame Rate of OpenArena (3-D Game) on Linux

NVIDIA proprietary driver
Nouveau open-source driver
Outline

1. Introduction
2. What's Problem
3. Our Solution – “TimeGraph”
4. Evaluation
5. Summary
**TimeGraph Architecture**

**User Space**
- Applications
  - OpenGL/CUDA Library
  - User-space GPU Driver

**Kernel Space**
- TimeGraph
  - GPU Command Queue
    - GPU Command Scheduler
    - GPU Reserve Manager
    - GPU exec. time prediction
    - GPU Command Profiler

**Kernel-space GPU Driver**
- Submission Interface
- IRQ Handler

**Device Space**
- Graphics Processing Unit (GPU)
Priority Support – Predictable Response Time (PRT) Policy

- When GPU is not idle, GPU commands are queued.
- When GPU gets idle, GPU commands are dispatched.

Diagram:
- High-priority task
- Low-priority task
- GPU command
- Interrupt
- Overhead
- Prioritized correctly
Priority Support – High Throughput (HT) Policy

- When GPU is not idle, GPU commands are queued, only if priority is lower than current GPU context.
- When GPU gets idle, GPU commands are dispatched.

Diagram:

- High-priority task
- Low-priority task
- GPU driver
- GPU command
- Interrupt

Overhead reduced
Reservation Support – Posterior Enforcement (PE) Policy

- Enforce GPU resource usage \textit{optimistically}
- Specify \textit{capacity} (C) and \textit{period} (P) per task (/proc/GPU/$\text{TASK}$)

![Diagram showing CPU and GPU execution budget over time with $C$ and $P$]
Reservation Support – Apriori Enforcement (AE) Policy

- Enforce GPU resource usage *pessimistically*
- Specify *capacity* (C) and *period* (P) per task (/proc/GPU/$TASK$)

Execution Budget

\[ C \]

\[ P \]
GPU Execution Time Prediction

• History-based approach
  – Search records of previous sequences of GPU commands that match the incoming sequences of GPU commands
  – Works for 2-D but needs investigation for 3-D and Compute

• Please see the paper for the detail 😊
Outline

1. Introduction
2. What’s Problem
3. Our Solution – “TimeGraph”
4. Evaluation
5. Summary
Experimental Setup

- **GPU**: NVIDIA GeForce 9800 GT
- **CPU**: Intel Xeon E5504
- **OS**: Linux Kernel 2.6.36
  - [*Nouveau* open-source driver]
- **Benchmark**:
    - Including OpenGL 3-D game programs
    - Including OpenGL 3-D widget and graphics-bomb programs
Performance Protection

Frame Rate of 3-D Game competing with Graphics Bomb in background

- **OpenArena**: No TimeGraph Support
- **World of Padman**: Priority Support (High Priority -> 3-D Game)
- **Urban Terror**: Priority & PE Reservation Support (GPU Util. 10% -> Graphics Bomb)
- **Unreal Trounament**: Priority & AE Reservation Support (GPU Util. 10% -> Graphics Bomb)

3-D Game Application
Interference on Time

No TimeGraph Support

Priority Support (PRT)

Priority Support (PRT) + Reservation Support (PE)
Standalone Performance

- OpenArena
- World of Padman
- Urban Terror
- Unreal Trounament

3-D Game Application

Average frame-rate (fps)

X server is assigned PRT policy

- No TimeGraph Support
- Priority Support (HT)
- Priority Support (PRT)
- Priority & Reservation Support (PRT & PE)
- Priority & Reservation Support (PRT & AE)

Overhead is acceptable for protecting GPU
Outline

1. Introduction
2. What’s Problem
3. Our Solution – “TimeGraph”
4. Evaluation
5. Summary
Concluding Remarks

- **TimeGraph** enables *prioritization* and *isolation* for **GPU** applications in multi-tasking environments
  - Device-driver solution: no modification to user-space
  - Scheduling of GPU commands
  - Reservation of GPU resource usage

- [http://rtml.ece.cmu.edu/projects/timegraph/](http://rtml.ece.cmu.edu/projects/timegraph/)
Current Status

- **GPGPU** support (collaboration with *PathScale* Inc.)
  - Visit [http://github.com/pathscale/pscnv](http://github.com/pathscale/pscnv)

- **Making open-source fast and reliable**
  - It’s getting competitive to the proprietary driver!
  - Some result from our OSPERT’11 paper (*) below:

```
<table>
<thead>
<tr>
<th></th>
<th>NVIDIA</th>
<th>Ours</th>
<th>NVIDIA</th>
<th>Ours</th>
<th>NVIDIA</th>
<th>Ours</th>
<th>NVIDIA</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 x 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32 x 32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64 x 64</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>128 x 128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>256 x 256</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512 x 512</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024 x 1024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Thank you for your attention!

Questions?