GPU-to-CPU Callbacks

Jeff A. Stuart and John D. Owens
University of California, Davis
Outline

• What are callbacks?

• Motivation

• Implementation

• Tests and Results
What Are Callbacks?

• Define: callback

• Current CPU-GPU Relationship

• Callbacks modify relationship

  – GPU requests work
Motivation

• Three big reasons for wanting callbacks
  – Programming Approach
  – Future Proof
  – Debugging

• Want something scalable, flexible
Motivation – Programming Model

- CPU commands GPU
  - Not always natural thought process
  - Might require splitting kernels
  - Creates programmer-time overhead
Motivation – Future Proof

• GPU is a slave

• System-on-a-Chip

• API-level solutions
Motivation – Debugging

• Debugging on a GPU
  – Hard to track progress, crashes
  – CUDAGDB isn’t always reliable
  – cuPrintf has limitations
Implementation

• Design Goals

• How we did it

• Walkthrough Example
Implementation – Design Goals

• Want to

  – Be easy to use API
  – Be scalable
  – Grant full system access
Implementation - How

- C-Level library
- Sits atop CUDA
- Uses o-copy memory, streams
- Polls (for now)
Implementation - Walkthrough

• CPU
  – Write a function
  – Register that function
  – Pass handle to GPU
  – Launch kernel and wait/poll
Implementation - Walkthrough

• GPU
  – Execute callback
    • Write arguments and flags in o-copy memory
  – Wait/poll
  – Continue
Tests

• Want to Test

  – System Calls

  – Debugging/Progress Tracking

  – Scalability (limited)
Tests – System Calls

• Simple TCP/IP echo server
  – CUDA Setup: Two blocks
  – Block one sends via network to block two
  – Block two echoes via network to block one
Tests – Debugging/Track Progress

• Old Fashioned printf
  – Implement a long-running kernel
  – Blocks output progress
  – Insert a seg-fault to crash GPU late in code
Tests - Scalability

- Many simultaneous callbacks
- Full Scale doesn’t make sense
- Implement a dynamic work queue with simple dynamic-memory management
  - Fed from CPU via callbacks
Results

• Design goals again were
  – Ease of use
  – Full access to system
  – Scalable
Results

• Echo server and printf
  – Easy to write
  – Used system calls
  – Worked
Results

• **Work Queue / Memory Manager**
  
  – Used up to 120 simultaneous callbacks

  – Request times, as seen from GPU
    
    • Average: < 4 ms
    
    • Maximum: ~ 10 ms
    
    • Minimum: < 2 ms
Getting the Library

• UCD has non-helpful IP regulations

• Privately available upon request

• Contact either author (Jeff, preferably)
Acknowledgments

• NSF Award
• SciDAC Insitute for Ultrascale Visualization
• NVIDIA CUDA Team and Research Group
• Equipment donated by NVIDIA
• Discussions and Feedback
  – Dominik Göddeke
  – Reviewers
• Implementation Assistance
  – Chris Cameron
The End

- Questions?