CUDA in Advanced Scenarios

Dmitri Nesteruk

@dnesteruk

Overview

- Inline PTX
- Driver API
- Pinned Memory (again!)
- Multi-GPU programming
- Thrust

Inline PTX

- **PTX is the ‘assembly language’ of CUDA**
- **You can output PTX code from your kernel**
  - `nvcc –ptx`
  - Project setting
- **You can also load a PTX kernel in with Driver API**
- **Embedding PTX into kernel also possible**
  - `asm("mov.u32 %0, %%laneid;" : "=r"(laneid));`
  - Splices the PTX right into your kernel
  - Allows referencing variables

Driver API

- **CUDA APIs**
  - Runtime API
    (what we've been using)
  - Driver API
    - `cuda.h, cuda.lib`
- **Driver API**
  - Allows low-level control of CUDA
  - No 'syntactic sugar' (<<<>>>, dim3, etc.)
  - Can be mixed with runtime API
  - Not useful to CUDA users in most cases

Generating Random Images

- Determine image dimensions
- Define *x* and *y* in the 0 to 1 range
- Create and call a super-complicated function

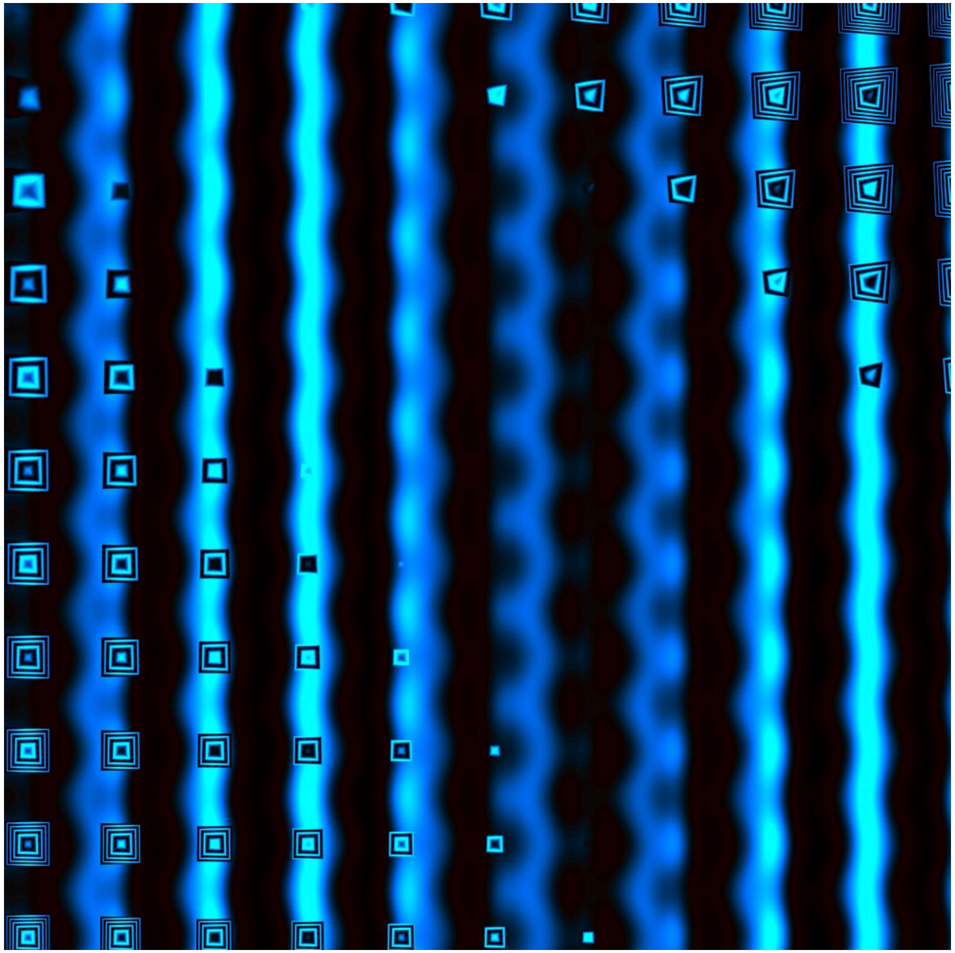
$$z = f(x, y)$$

- Write data back to image

$$r = az^2 + bz + c$$

- Needs a brand new CUDA kernel on each call
- How to recreate, compile, load and execute a kernel?

Answer: driver API



Pinned Memory

- **`cudaHostAlloc(pHost, size, *flags*)`**
- **`flags` parameter can be**
  - `cudaHostAllocMapped`
    - Maps memory directly into GPU address space
    - Lets you access host memory directly from GPU
    - A.k.a. “zero-copy memory”
    - Use `cudaHostGetDevicePointer()` to get device address
  - `cudaHostAllocPortable`
    - Ordinary pinned memory visible to one host thread
    - Portable pinned memory is allowed to migrate between host threads
  - `cudaHostAllocWriteCombined`
    - Write-combined memory transfers faster across the PCI bus
    - Cannot be read efficiently by CPUs
- **Can use any combination of the flags above**

Multi-GPU Programming

- **Execute parts on separate devices**
  - Split the work
  - Execute kernels on separate threads
  - Combine the results
- **Use `cudaSetDevice(id)` to select the device to run on**
- **Portable zero-copy memory useful for multi-threading**

Thrust Library

- **STL-like library for accelerated computation**
- **Included with CUDA**
- **host_vector and device_vector**
  - Assign, resize, etc. (but each `d[n] = z;` causes a `cudaMemcpy`)
  - Copy with = operator
  - Can interop with STL containers and CUDA raw memory
- **Predefined algorithms**
  - Search, sort, copy, reduce
- **Functor syntax**
  - `thrust::transform(x.begin(), x.end(), y.begin(), y.begin(), thrust::multiplies<float>());`

Summary

- **CUDA C kernels get turned into PTX**
  - Can inject PTX inline
- **Driver API provides low-level access to CUDA infrastructure**
  - Lets you load kernels from PTX or cubin at runtime
- **Pinned memory can be mapped, portable and write-combined**
- **Running on multiple devices is not difficult**
- **Thrust library makes using CUDA a lot easier**

- **End of course… thanks for watching!**