An Introduction to OpenCL

Ben Alun-Jones
Overview

- GPU Computing: What is it?
- What is OpenCL?
- Why OpenCL?
- Architecture
- Example Code
- Application
GPU Computing

- Using the graphics card with the CPU for general purpose computing
- Take advantage of the inherent parallelism of many computationally intensive operations
- Limited by need to transform problems into the graphics ‘paradigm’
- Limited to certain operations
What is OpenCL?

- Framework based on open standards for parallel and distributed computing
- Based on ANSI C
- Must be supported by hardware
  - e.g. Builds on top of NVIDIA’s CUDA architecture

Tuesday, 11 May 2010
Why OpenCL?

- Allows you to write one piece of code that can then run over multiple machines/architectures/operating systems

- BUT probably not optimal for all (or any) of the compute engines

- Allows for a broader range of CPU and GPU devices

- Can use multiple processing units.....
OpenCL Architecture

Compute Device

Compute unit 1

Private memory 1

PE 1

Local memory 1

...  

Private memory M

PE M

...  

Global/Constant Memory Data Cache

Global Memory

Constant Memory

Compute Device Memory

Compute unit N

Private memory 1

PE 1

Local memory N

...  

Private memory M

PE M
**Basic Program Structure**

**Host program**
- Create memory objects associated to contexts
- Compile and create kernel program objects
- Issue commands to command-queue
- Synchronization of commands
- Clean up OpenCL resources

**Platform Layer**

**Runtime**
- Query compute devices
- Create contexts

**OpenCL Language**
- Compute Kernel *(runs on device)*
  - C code with some restrictions and extensions

© NVIDIA Corporation 2009
Example Code: Vector Addition

- Add two vectors together:
  \[ c[i] = a[i] + b[i] \]

- Equivalent C loop:

```c
int iNumElements = 11444777;

for (int i = 0; i < iNumElements; i++) {
    c[i] = a[i] + b[i];
}
```
Example Code: Vector Addition

- **Set Up**
  - Set work sizes for kernel execution
  - Allocate and init host data buffers
  - Create context for GPU device
  - Query compute devices
  - Create command queue
  - Create buffers on the GPU device
  - Create and build a program
  - Create kernel
  - Set kernel arguments

- **Core sequence**
  - Copy (write) data from host to GPU
  - Launch kernel in command-queue
  - Copy (read) data from GPU to host... block

- **Clean up**
Kernel Code

Source code for the computation kernel, stored in text file (read from file and compiled at run time, e.g. during app. init)

// OpenCL Kernel Function for element by element vector addition
// *****************************************************************************
__kernel void VectorAdd ( __global float* a, __global float* b, __global float* c,
            __global int iNumElements)
{
    // get index into global data array
    int iGId = get_global_id(0);

    // bound check (equivalent to the limit on a 'for' loop for standard/serial C code
    if (iGId >= iNumElements)
    {
        return;
    }

    // add the vector elements
    c[iGId] = a[iGId] + b[iGId];
}
Demo of OpenCL
Computation
....Really really fast.....
OpenCL and OpenGL

• Can set up your ‘context’ so that OpenGL and OpenCL share buffers

• This points to a situation where OpenCL calculates your data (very quickly) and OpenGL plots it.

• The future for massive, complex visualisations...
Demo of OpenCL/OpenGL interoperability

Two open standards in one....
Summary

• OpenCL a new standard for Parallel computing

• Allows rapid computation (up to 100x speed increase) of certain problems

• Can also be used to improve visualisation times since shared buffers with OpenGL