Lecture 25 - Features of OpenCL

ECE 459: Programming for Performance

Jon Eyolfson

University of Waterloo

March 12, 2012

Introduction

- This is an example of programming for a heterogeneous architecture
  - We're no longer using the general CPU, we'll also leverage the GPU
- We'll be looking at OpenCL (i.e. Open Computing Language)
- Usable on both NVIDIA and AMD GPUs

SIMT

Another term you may see vendors using:

- Single Instruction, Multiple Threads
- Runs on a vector of data
- This is similar to SIMD instructions (for example SSE)
  - However, its spread out on the GPU

Other Examples

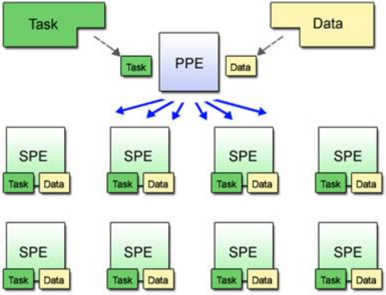
- PlayStation 3 Cell

- CUDA

Cell Overview

Cell consists of

- PowerPC core
- 8 SIMD co-processors



(from the Linux Cell documentation)

CUDA Overview

- Compute Unified Device Architecture
- NVIDIA's architecture for processing on GPUS
- "C for CUDA" predates OpenCL, NVIDIA supports it first and foremost
  - May be faster than OpenCL on NVIDIA hardware
  - API allows you to use (most) C++ features in CUDA code, not in OpenCL

Programming Model

The abstract model is simple:

- Write the code for the parallel computation (kernel) separately from main code
- Transfer the data to the GPU co-processor (or execute it on the CPU)
- Wait
- Transfer the results back

Data Parallelism

- You are evaluating a function (or *kernel*) on a set of points (data)
- This is another example of data parallelism
- Another name for the set of points is the *index space*
- Each of the points corresponds to a **work-item**

Note: OpenCL also supports *task parallelism* (using different kernels), but the documentation doesn't say too much

Work-Items

- This is the fundamental unit of work in OpenCL
- They are stored in an *n*-dimensional grid (ND-Range)
- OpenCL spawns a bunch of threads for the work-items
- When executing, the range is divided into **work-groups** which execute on the same compute unit
  - The set of compute units (or cores) is called something different depending on the manufacturer
    - NVIDIA - *warp*
    - AMD/ATI - *wavefront*

Shared Memory

There are many different types of memory available to you:

- private memory: available to a single work-item;
- local memory (aka "shared memory"): shared between work-items belonging to the same work-group, like a user-managed cache;
- global memory: shared between all work-items as well as the host;
- constant memory: resides on the GPU and cached. Does not change.

Example Kernel

- Here's some traditional code to multiple an array by two arrays

```
void traditional_mul(int n, const float *a, const float *b,
                     float *c) {
  int i;
  for (i = 0; i < n; i++) c[i] = a[i] * b[i];
}
```

- This is the same code as a kernel

```
kernel void opencl_mul(global const float *a,
                       global const float *b,
                       global float *c) {
  int id = get_global_id(0);  // dimension 0
  c[id] = a[id] * b[id];
}
```

Restrictions

- No function pointers
- No bit-fields
- No variable length arrays
- No recursion
- No standard headers

Additions

- Work-items
- Work-groups
- Vectors
- Synchronization
- Declarations of memory type
- Library for kernels to use

Branches

- The computation from each work-item can branch arbitrarily
- The hardware will execute all branches that any thread in a warp executed (with can be slow)

- This means an `if` statement will cause each thread to execute both branches, keeping only the result of the appropriate branch

- A loop will cause the workgroup to wait for the maximum number of iterations of the loop in any work-item

Note: when you set up work-groups, you can arrange for all the work-items in a workgroup to execute the same branches

Synchronization

- You can only put barriers and memory fences between work-items in the same workgroup
- Reminder: the workgroups execute independently

Support for the following:

- Memory fences (load and store)
- Barriers
- `volatile`

Summary

- Brief overview of OpenCL and it's programming model
- Many concepts are similar to plain parallel programming
- Again, reminder to e-mail me for Assignment 3 (anyone have anything for the leaderboard?)