GPU Teaching Kit

Accelerated Computing

Lecture 20 – Related Programming Models: OpenCL

Lecture 20.1 - OpenCL Data Parallelism Model

Objective

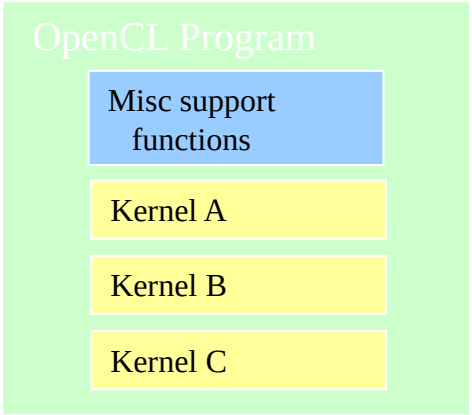
- To Understand the OpenCL programming model
  - basic concepts and data types
  - Kernel structure
  - Application programming interface
  - Simple examples

Background

- OpenCL was initiated by Apple and maintained by the Khronos Group (also home of OpenGL) as an industry standard API
  - For cross-platform parallel programming in CPUs, GPUs, DSPs, FPGAs,…
- OpenCL draws heavily on CUDA
  - Easy to learn for CUDA programmers
- OpenCL host code is much more complex and tedious due to desire to maximize portability and to minimize burden on vendors

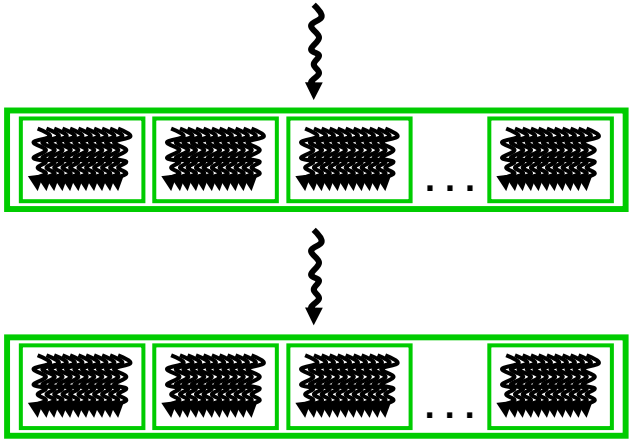
OpenCL Programs

- An OpenCL “program” is a C program that contains one or more “kernels” and any supporting routines that run on a target device
- An OpenCL kernel is the basic unit of parallel code that can be executed on a target device



OpenCL Execution Model

- Integrated host+device app C program
  - Serial or modestly parallel parts in host C code
  - Highly parallel parts in device SPMD kernel C code



Mapping between OpenCL and CUDA data parallelism model concepts.

OpenCL Parallelism Concept	CUDA Equivalent
host	host
device	device
kernel	kernel
host program	host program
NDRange (index space)	grid
work item	thread
work group	block

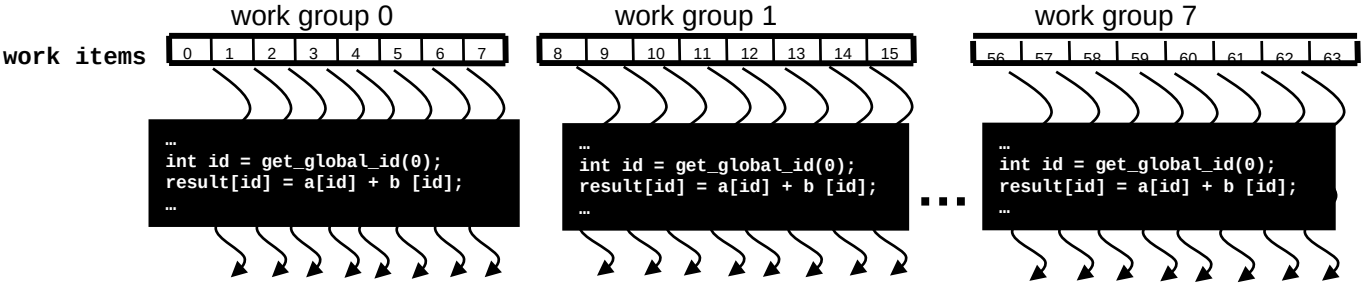
OpenCL Kernels

- Code that executes on target devices
- Kernel body is instantiated once for each work item
  - An OpenCL work item is equivalent to a CUDA thread
- Each OpenCL work item gets a unique index

```
__kernel void  vadd(__global const float *a,
                    __global const float *b,
                    __global float *result)
{
    int id = get_global_id(0);
    result[id] = a[id] + b[id];
}
```

Array of Work Items

- An OpenCL kernel is executed by an array of work items
  - All work items run the same code (SPMD)
  - Each work item can call get_global_id() to get its index for computing memory addresses and make control decisions



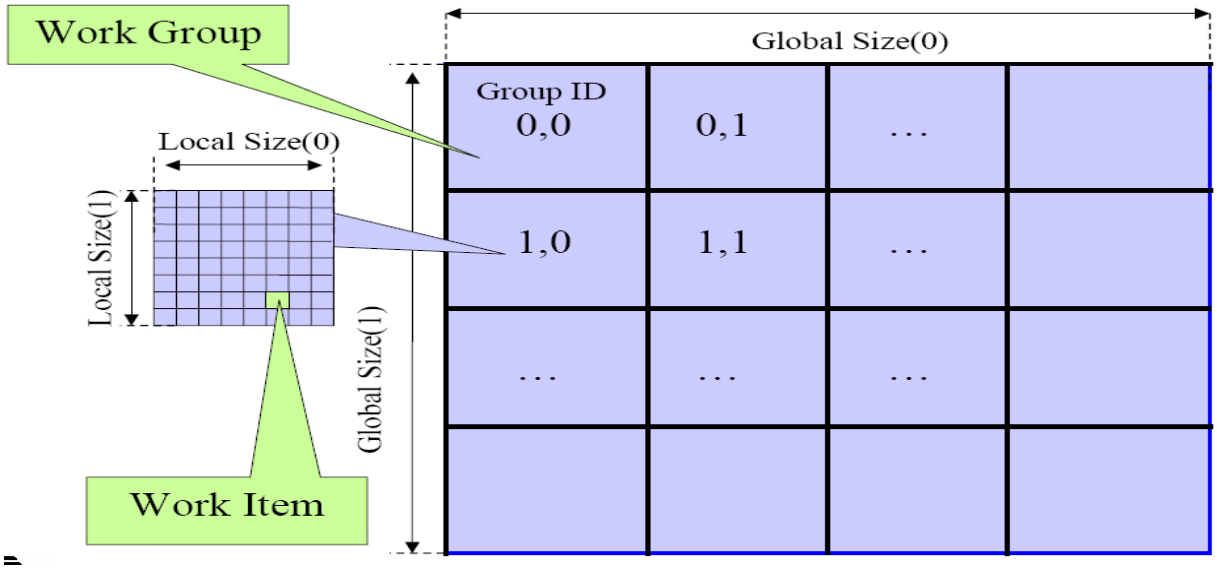
Work Groups: Scalable Cooperation

- Divide monolithic work item array into work groups
  - Work items within a work group cooperate via shared memory and barrier synchronization
  - Work items in different work groups cannot cooperate
- OpenCL counter part of CUDA Thread Blocks

OpenCL Dimensions and Indices

OpenCL API Call	Explanation	CUDA Equivalent
get_global_id(0);	global index of the work item in the x dimension	blockIdx.x*blockDim.x +threadIdx.x
get_local_id(0)	local index of the work item within the work group in the x dimension	threadIdx.x
get_global_size(0);	size of NDRange in the x dimension	gridDim.x*blockDim.x
get_local_size(0);	Size of each work group in the x dimension	blockDim.x

Multidimensional Work Indexing



OpenCL Data Parallel Model Summary

- Parallel work is submitted to devices by launching kernels
- Kernels run over global dimension index ranges (NDRange), broken up into “work groups”, and “work items”
- Work items executing within the same work group can synchronize with each other with barriers or memory fences
- Work items in different work groups can’t sync with each other, except by terminating the kernel

GPU Teaching Kit

Accelerated Computing