Heterogeneous Parallel Programming

Related Programming Models
OpenCL Data Parallelism Model

Wen-mei Hwu – University of Illinois at Urbana-Champaign
Objective

- To Understand the OpenCL programming model
- basic concepts and data types
- kernel structure
- application programming interface
- simple examples
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
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.

<table>
<thead>
<tr>
<th>OpenCL Parallelism Concept</th>
<th>CUDA Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>host</td>
</tr>
<tr>
<td>device</td>
<td>device</td>
</tr>
<tr>
<td>kernel</td>
<td>kernel</td>
</tr>
<tr>
<td>host program</td>
<td>host program</td>
</tr>
<tr>
<td>NDRange (index space)</td>
<td>grid</td>
</tr>
<tr>
<td>work item</td>
<td>thread</td>
</tr>
<tr>
<td>work group</td>
<td>block</td>
</tr>
</tbody>
</table>
**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

```c
__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];
}
```
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 equivalent of CUDA Thread Blocks
## OpenCL Dimensions and Indices

<table>
<thead>
<tr>
<th>OpenCL API Call</th>
<th>Explanation</th>
<th>CUDA Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_global_id(0);</td>
<td>global index of the work item in the x dimension</td>
<td>blockIdx.x*blockDim.x + threadIdx.x</td>
</tr>
<tr>
<td>get_local_id(0)</td>
<td>local index of the work item within the work group in the x dimension</td>
<td>threadIdx.x</td>
</tr>
<tr>
<td>get_global_size(0);</td>
<td>size of NDRange in the x dimension</td>
<td>blockDim.x</td>
</tr>
<tr>
<td>get_local_size(0);</td>
<td>Size of each work group in the x dimension</td>
<td>blockDim.x</td>
</tr>
</tbody>
</table>
Multidimensional Work Indexing

- Work Group
- Local Size
  - Local Size(0)
  - Local Size(1)
- Global Size
  - Global Size(0)
  - Global Size(1)

Grid with Work Groups and Work Items:

- Group ID
  - 0,0
  - 0,1
  - 1,0
  - 1,1
  - ...

- Work Items
  - 0,0
  - 0,1
  - 1,0
  - 1,1
  - ...

- Grid layout for multidimensional indexing of work items.
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.
Heterogeneous Parallel Programming

TO LEARN MORE, READ SECTION 14.1-14.2