Notes on OpenCL

by Ramkumar Narayanan

Contents

November 2, 2020

1 Programming Models

1.1 Platform Model

This is a very high level description of the heteregeneous system.

1. ***Host:*** A device that controls all the OpenCL compatible devices. This is where the main entry point of the program. The host itself could be an OpenCL compatible device.
2. ***Compute Devices:*** Each graphics card or the CPU or FPGA's form what is called the compute device. All the compute devies are visible by the host.
3. ***Compute Units*** Like the several cores of the processor whether it be a GPU or a CPU.
4. ***Processing Elements:*** Each core will have several small units that does the actual processing call the processing elements.

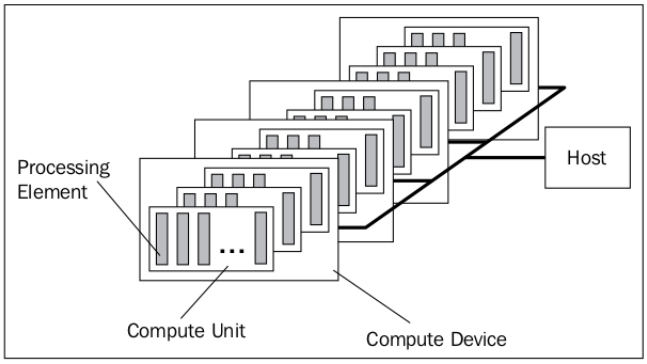

Figure 1: OpenCL platform model

1.2 Execution Model

An OpenCL program has two distinct parts: 1) ***Host*** program and 2) a collection of one or more ***kernels***. Host program runs on the host. Host program is our normal cpp or python code. OpenCL does not define how that is programmed - only how it interacts with the compute device(s).

1. ***OpenCL Kernels:*** Written in C language and compiled with the OpenCL C compiler.
2. ***Native Kernels:*** These functions are created outside OpenCL and accesssed within OpenCL through a function pointer.

1.2.1 Work IDs:

Work Item: Host program issues a command that submits the kernel for execution on the OpenCL device. Each execution of the kernel is called a work item. Each work item running might be running the same sequence of instruction but because of the branching statements and the data selected, its behaviour can differ.

Index Space: When a kernel is submitted by the host program for execution, an index space is created. An instance of kernel is run for each point in the index space.

Work Group: Now that we have defined what an index space and work item is, a work group is nothing but an even division of the index space into many blocks of work items as shown in the figure below.

The index space can be 1, 2 or 3 dimensional, they are popularly called the ND space and they span the NDRange. Each work item/group has an ID. They can be global or local. Let us see it with an example. Suppose we have a 2D index space whose size is given by (G_x, G_y); they span values from $0 \rightarrow (G_x - 1)$ along the x axis and $0 \rightarrow (G_y - 1)$ along the y axis. Suppose, we divide the space into work groups each of size W_x and W_y, we have each work group size as G_x/W_x and G_y/W_y. A work item can be defined by its global id as (g_x, g_y). Its local ID can be computed by knowing the local work group id the displacement within that work group.

1.2.2 Context:

A context is an OpenCL abstraction for the items required for the setup of running of a program. A ***host*** defines the kernel, the NDRange, queues that controls the details of how and when the kernels execute. However, the first task on the host is to define the ***context*** for execution. Context can be defined in the following terms

1. ***Devices:*** A collection of OpenCL devices to be used by the host.
2. ***Kernels:*** OpenCL functions that run on the OpenCL devices.
3. ***Program Objects:*** The program source code and the executables that implement the kernels. Remember 'shaders'? They compile during runtime and create program objects.
4. ***Memory Objects:*** Memory visible to OpenCL devices.

Suppose, there are several devices such as a CPU, 2 GPU's etc. A context can be defined by using all the cores of a CPU or either of the GPU's or infact a combination of these. Since OpenCL is heteregeneous hardware compliant, it is not possible to know before hand which device (or combination of devices) the end user is going to run the code on. Hence, the only possibility is to query the resources etc at runtime and build the program object at runtime. Kernel code is either a long string of code within the host program or it can be loaded from a file. Now, with regards to memory, due to several devices, there are several memory spaces to manage. Host has an address space in the CPU and the devices have their own architecture and memeory spaces. To deal with that OpenCL introduces the concept of memory objects. These are explicitely defined on the host and moved between the host and the OpenCL devices.

1.2.3 Command-Queues: