Notes on OpenCL

by Ramkumar Narayanan

Contents

1 What is OpenCL? **2**

2 Programming Models **2**

2.1 Platform Model 3

2.2 Execution Model 3

2.2.1 Context: 5

1 What is OpenCL?

OpenCL is an industry standard for programming devices composed of a combination of CPU's, GPU's and other processors. This platform enables heterogeneous, distributed computing. OpenCL exposes the hardware. It does not attempt to hide it behind its abstractions. From a hardware perspective, we are tending towards obtaining maximum compute power per watt expenditure.

Parallel hardware needs parallel software. This can be introduced with the concept of concurrency. Concurrency means running more than one stream of operations. Challenge is to find points of concurrency in a software system and express it as a source code. The resulting concurrent programs should deliver performance. Finding concurrency is pretty challengingand sometimes tricky. Coding up concurrency is the crus of the problem. Abstraction is often required to make the prallel program more manageable.

2 Programming Models

This is a very high level description of the heteregeneous system. Basically there are two broad types of parallelism 1) ***task parallelism*** and ***data parallelism***. Programmers break the problem down to individual tasks and then focus on how to bring about concurrency. This is easiest when tasks are independent. However, even when data is shared, this can be achieved. Concepts like load balancing comes here.

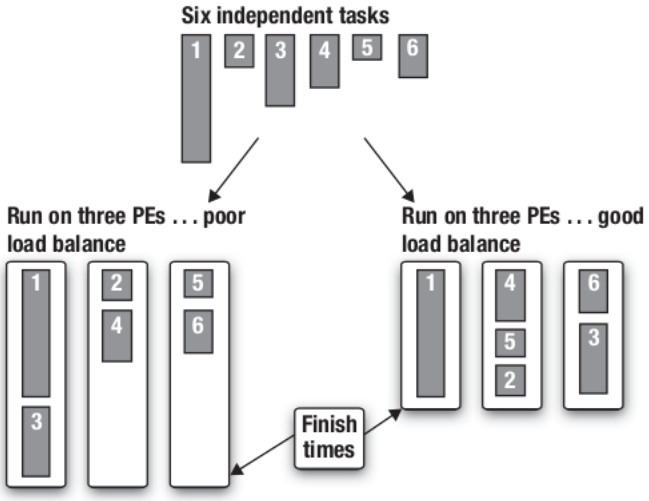

Figure 1: Load balancing of different tasks

In ***data parallelism*** programmers update chunks of data elements in parallel. programmers think of their problems in terms of data elements.

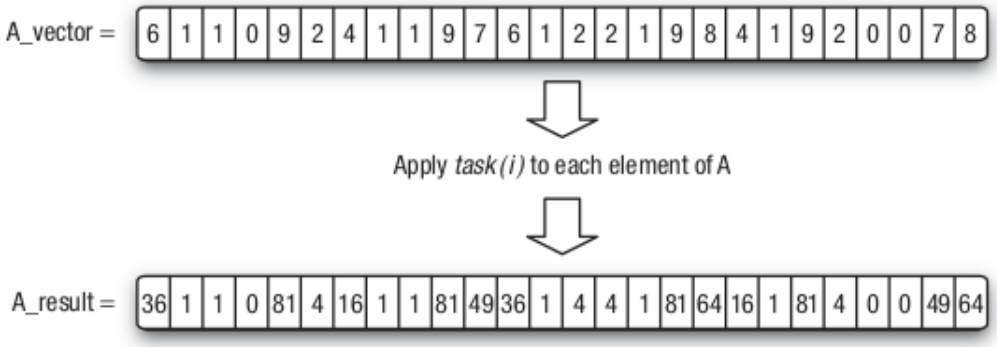

Figure 2: Data parallelism where a single task is applied concurrently to many data elements

2.1 Platform Model

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.

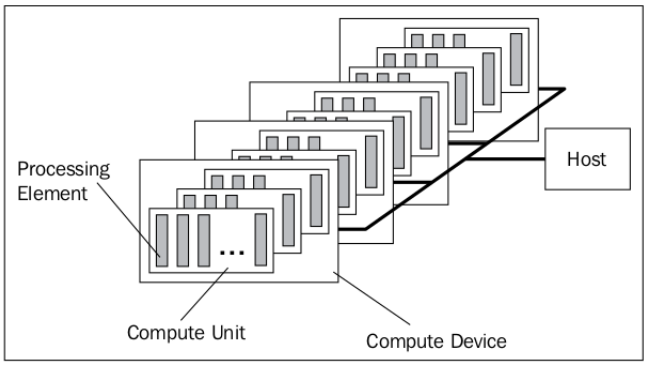

Figure 3: OpenCL platform model

2.2 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.

How does a kernel execute on the OpenCL device? A kernel is defined on the host. The host program issues a command that submits the kernel for execution. When this command is issued, OpenCL runtime creates what is called the ***index space***. An instance of kernel executes for each point in the index space. Each instance of the executing kernel is called a ***work item***. A work item is identified by the coordinates in the index space. The behaviour of each kernel execution can differ due to branch statements in the execution. Work items are organized into work groups providing a more coarse grained decomposition of the index space. Work groups exactly span the index space. Work groups are of the same size. There are relative and global ID's for work groups and work items. Work items within a workgroup executes concurrently and workgroups may/may-not execute concurrently. The index space spans N-dimensions hence called the NDRange. Currently N can span 1, 2 or 3 dimensions. Let us take a look at the schemes using a 2D Range example.

Notations:

1. G_x and G_y - indicates the sizes of the index space.
2. W_x and W_y - index space is chopped into so many work groups.
3. L_x and L_y - individual work group size after chopping.
4. (g_x, g_y) - represents the global coordinates of the work item.
5. (w_x, w_y) - represents the coordinates of the work group.
6. (l_x, l_y) - represents work item coordinate within the work group.

Take a look at the next image.

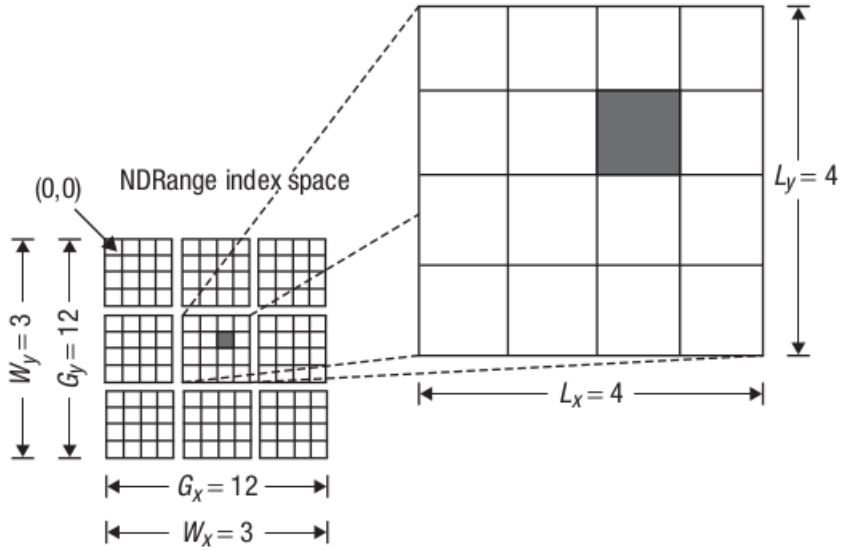

Figure 4: NDrange, work group, work id

Number of work groups, $W_x = G_x/L_x$ and $W_y = G_y/L_y$.

1. $(G_x, G_y) = (12, 12)$
2. $(W_x, W_y) = (3, 3)$
3. $(L_x, L_y) = (4, 4)$
4. $(g_x, g_y) = (6, 5)$
5. $(w_x, w_y) = (1, 1)$
6. $(l_x, l_y) = (2, 1)$

2.2.1 Context:

Computation of the kernels is always done on the OpenCL devices. However, the host has a very important job of setting up various stuff. For instance, the host defines the NDRange and the queues that control the details of how and when the kernels execute. All these definitions are contained in the API's within the OpenCL definition.

A context defines these environmental parameters within which the kernels are defined and executed. To be more precise, context is defined in the following terms,

1. ***Devices:*** A collection of OpenCL devices used by the host.
2. ***Kernels:*** A collection of OpenCL functions that run on the devices.
3. ***Program Objects:*** The source code and executables that implement kernels.
4. ***Memory Objects:*** a set of objects in memory that are visible to OpenCL devices and contain values that can be operated by instances of kernels.