OPENCL
Module

Jose Maria Buades Rubio

Universitat de les Illes Balears

v 0.8 – 2016 November

Content

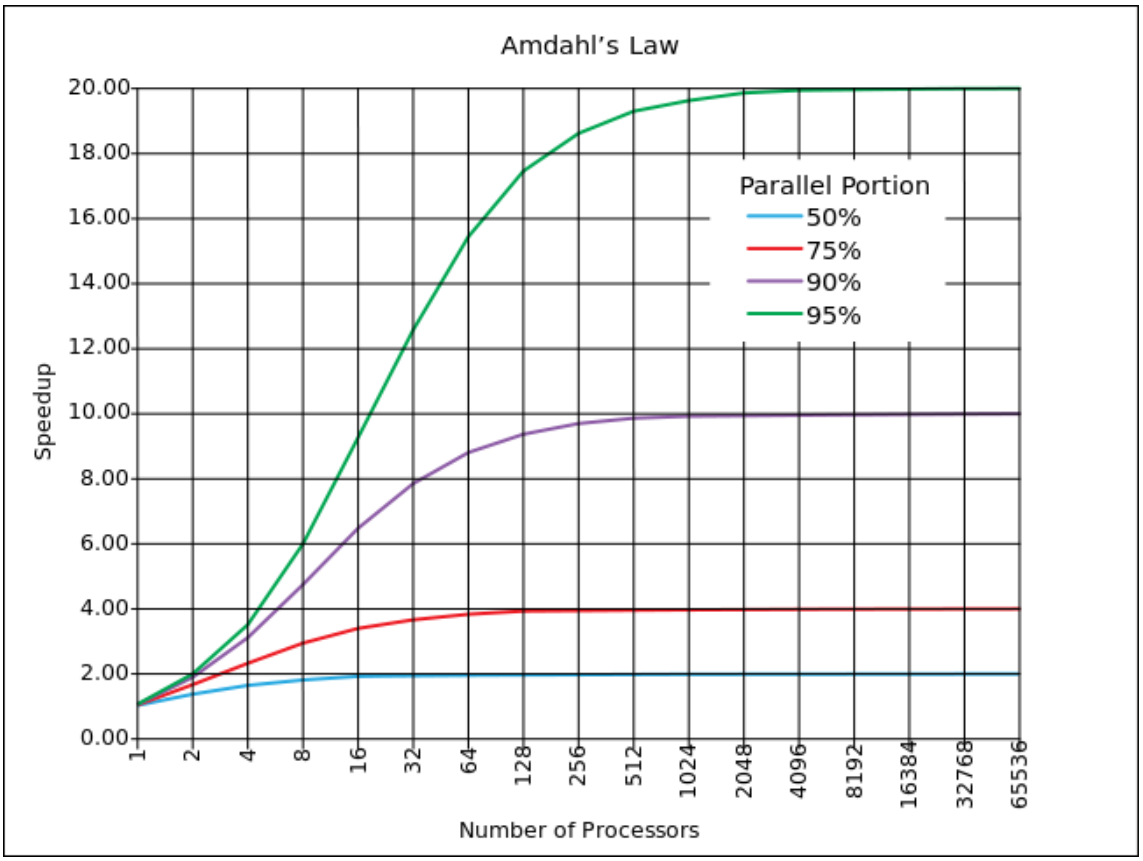
Introduction to Parallel Programming

Parallelism is a computation way, different processes are carried out simultaneously, based on splitting a big set of data in no dependent operations.

There are different kinds of parallelism: bit level, instruction level, data level and task level.

Parallel programming has greater difficulty opposite sequential ones. This is because concurrent memory access can generate problems that will produce unexpected behavior and bugs.

The best performance improve of a application due to paralleling can be described by the Amdahl's law.



Graphical representation of the Amdahl's law. The performance improve of an application due to parallelization is limited to the portion of the program that can be parallelized. For example, if the 80% of a program is able to be executed in parallel, the theorical maximum improve of the performance is 10x, independently the number of processors is used.

Types of Parallelism

- Bit level. Increasing bit processing capacity.
- Instruction level. Executing different instruction at the same time, splitting in different steps: IF (Instruction Fetched), ID (Instruction Decode), EX (Execute), MEM (Memory Access), WB (Register Write Back)
- Data Parallelism. Execute same instruction over different data.
- Task Parallelism. Is the main characteristic of a parallel program, each unit process can execute different computation.

CUDA-OpenCL-DirectCompute Comparative

CUDA

CUDA goes a little ahead; the new CUDA Version 6.5 has more features than others. But only can be executed on NVidia cards.
But tools for CUDA (debugging…) are better.

OpenCL

The main advantage is its portability. Can be executed on ATI, NVidia, and also on Intel and AMD CPUs

DirectCompute

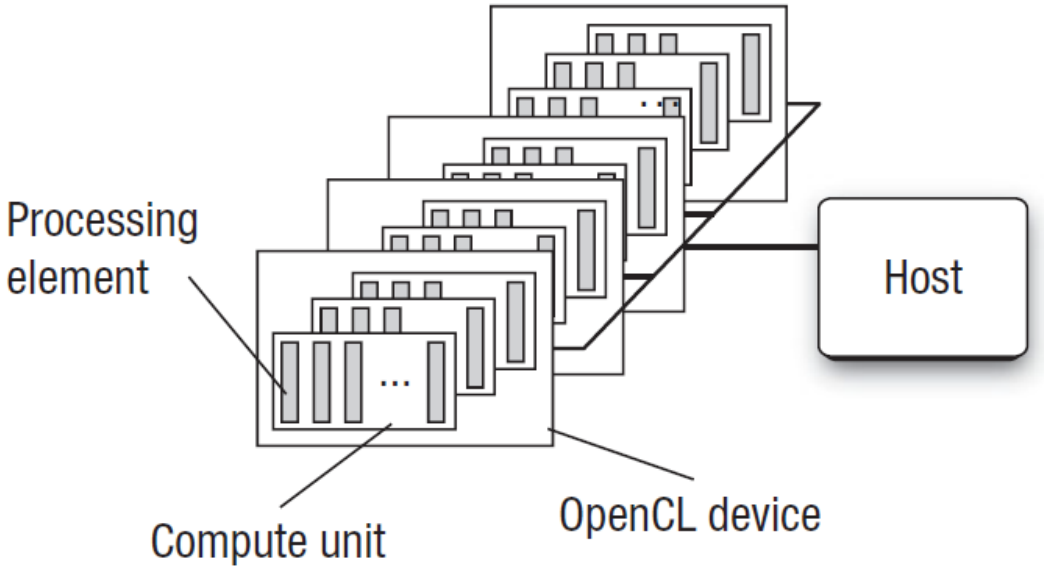
Videogames industry mostly uses (in PC) DirectX, so is an easier way to introduce in parallel programming in GPU to use DirectCompute for videogames programmers. It's very close language to HLSL (Shader language in DirectX).

OpenCL Programming Guide

OpenCL Model

Platform Model

OpenCL needs a host in charge to start parallel programs and do user interaction.
A platform is a vendor implementation. In a computer can exist more than one platform (Intel, AMD, NVidia), and any platforms will do available the devices, we could have two NVidia devices and one Intel device. The devices can be catalogued as CPUs, GPUS, DSP and any other.
A single graphic card, like GTX 295, can be shown as two OpenCL devices. A device is divided in compute units, each compute unit has an array of processing elements.



Execution Model

From the host OpenCL can execute public functions, such functions are called kernels.
This functions can be developed in a very similar language to C99. Each instance executing a kernel is called work-item, that is defined by a global id. Work-items are grouped in work-groups. Each work-group provide benefits to its workítems: synchronization capabilities and shared local memory access between work-items.
Currently, the index space spans an N-Dimensional range of values (1,2 o 3).
For example, we can execute a kernel that process an image, the dimension will be two, and the range should be adapted to the resolution, for example: 1920 x 1080. Then, a work-item will be identified by two integers. How work-items area grouped in work-groups depends on the implementation (hardware). Work-groups can improve its performance using shared local memory.
The computational work of an OpenCL application takes place on the OpenCL devices. The host, however, plays a very important role in the OpenCL application. It is on the host where the kernels are defined.

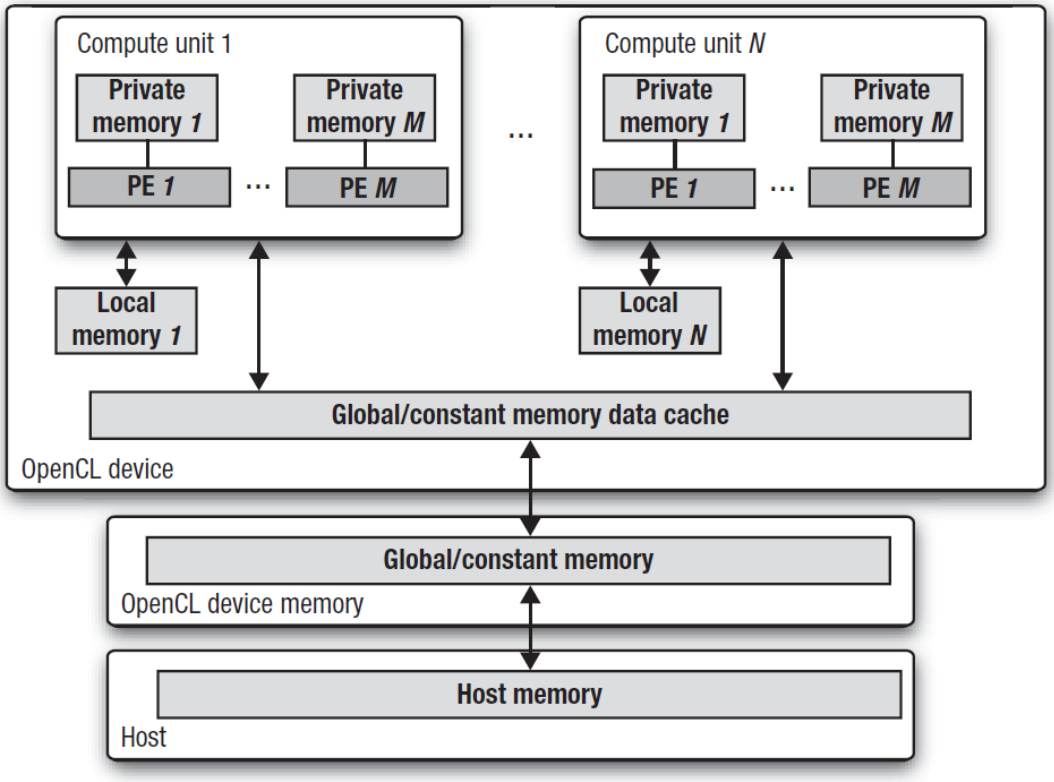
The host establishes the context for the kernels. The host defines the ND-Range and the queues that control the details of how and when the kernels execute. All of these important functions are contained in the APIs within OpenCL's definition.
The first task for the host is to define the context for the OpenCL application.
As the name implies, the context defines the environment within
which the kernels are defined and execute. To be more precise, we define
the context in terms of the following resources:

- Devices: the collection of OpenCL devices to be used by the host
- Kernels: the OpenCL functions that run on OpenCL devices
- Program objects: the program source code and executables that implement the kernels
- Memory objects: a set of objects in memory that are visible to OpenCL devices and contain values that can be operated on by instances of a kernel.

Memory Model

We have four memory types:

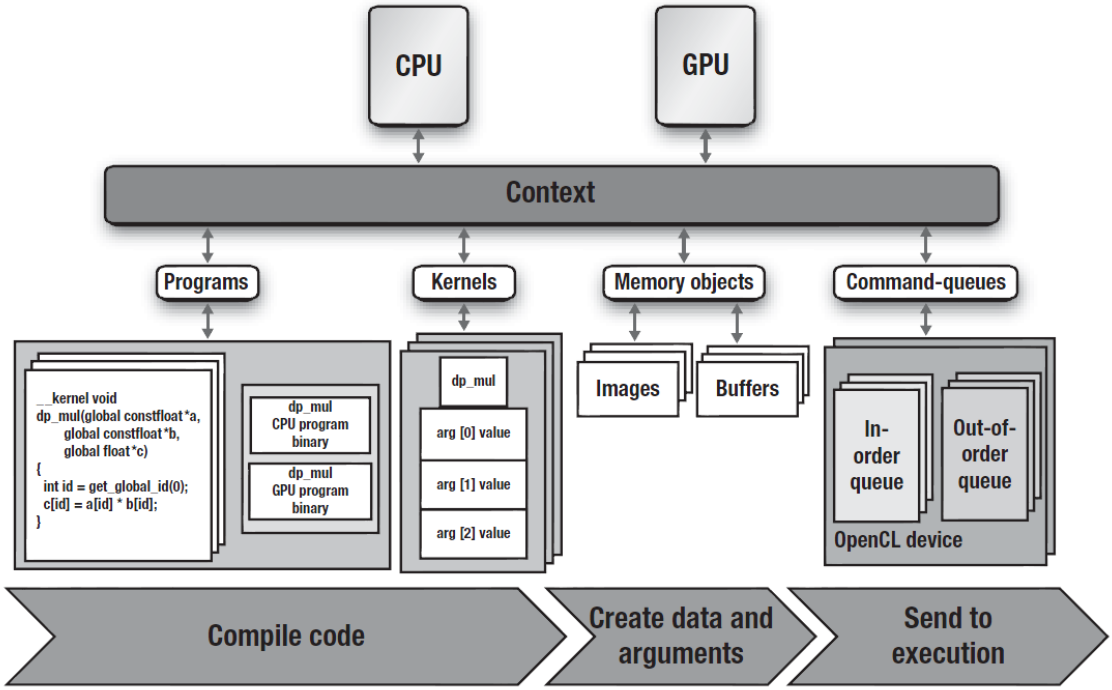
- Host memory: This memory region is visible only to the host. As with most details concerning the host, OpenCL defines only how the host memory interacts with OpenCL objects and constructs.
- Global memory: This memory region permits read/write access to all work-items in all work-groups. Work-items can read from or write to any element of a memory object in global memory. Reads and writes to global memory may be cached depending on the capabilities of the device.
- Constant memory: This memory region of global memory remains constant during the execution of a kernel. The host allocates and initializes memory objects placed into constant memory. Work-items have read-only access to these objects.
- Local memory: This memory region is local to a work-group. This memory region can be used to allocate variables that are shared by all work-items in that work-group. It may be implemented as dedicated regions of memory on the OpenCL device. Alternatively, the local memory region may be mapped onto sections of the global memory.
- Private memory: This region of memory is private to a work-item. Variables defined in one work-item's private memory are not visible to other work-items.

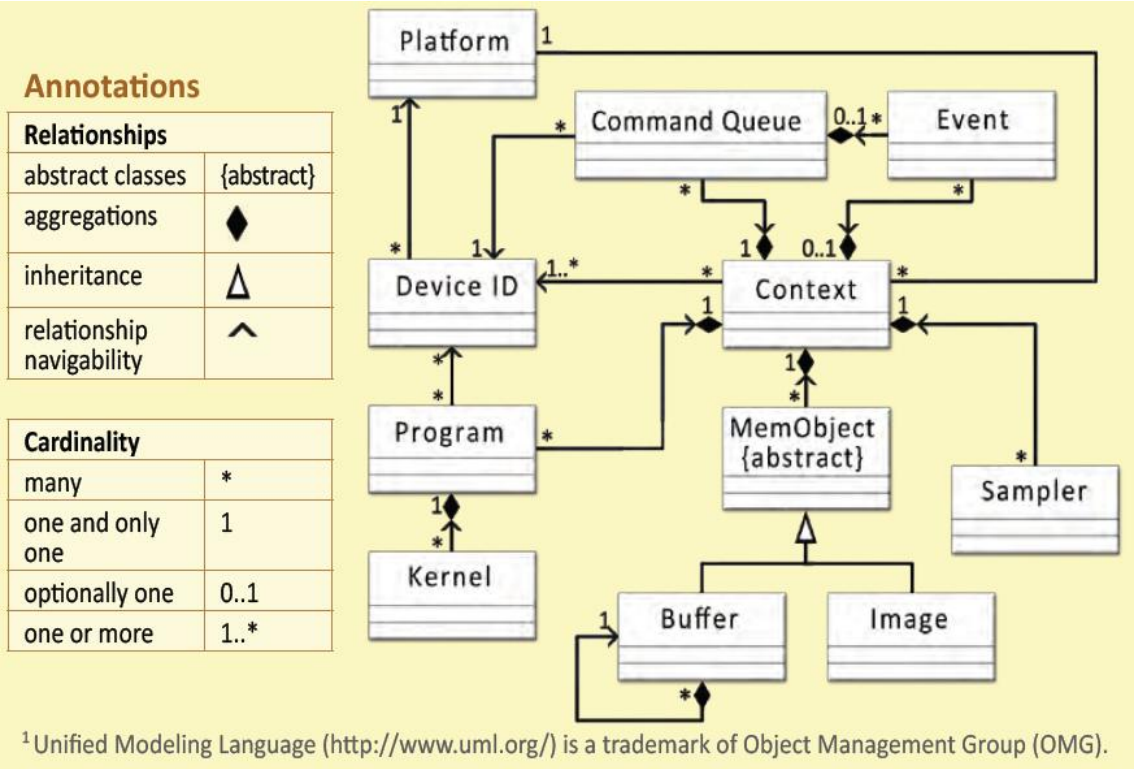


To transfer data between host memory and global memory OpenCL proved specific routines. In some applications memory transfer will be the bottleneck.

Modelos Programación

- OpenCL platform API: The platform API defines functions used by the host program to discover OpenCL devices and their capabilities as well as to create the context for the OpenCL application.
- OpenCL runtime API: This API manipulates the context to create command-queues and other operations that occur at runtime. For example, the functions to submit commands to the command-queue come from the OpenCL runtime API.
- The OpenCL programming language: This is the programming language used to write the code for kernels. It is based on an extended subset of the ISO C99 standard and hence is often referred to as the OpenCL C programming language.
    - Vector types and its operations.
    - Large set of predefined functions for parallel programming.
    - Atomic functions to accesss local and global memory.





[1] Unified Modeling Language (http://www.uml.org/) is a trademark of Object Management Group (OMG).

Environment Setup

Install developer environment Visual Studio with C++).

Install OpenCL (Intel, AMD-ATI and/o NVidia)

INTEL

Software

Intel(R) SDK for OpenCL

Compiler

Microsoft Visual Studio 2008 and above

Hardware

3rd Generation Intel® Core™ Processors (i3, i5, i7)
2nd Generation Intel® Core™ Processors (i3, i5, i7, i7 Extreme)
Previous Generation Intel® Core™ Processors (i3, i5, i7, i7 Extreme)
Intel® Core™ 2 processor family (Formerly Penryn)
- Intel® Core™ 2 Extreme Processor, 9000 series
- Intel® Core™ 2 Quad Processor 8000 and 9000 series
- Intel® Core™ 2 Duo Processor, 8000 series
- Intel® Core™ 2 Duo Processor, E7200 series

SO

Windows 8
Windows 7
Windows Vista

URL

https://software.intel.com/en-us/articles/opencl-drivers

AMD-ATI

Software

AMD APP SDK 2.9.1

Compiler

Microsoft Visual Studio 2008 and above

Hardware

Must support SSE3 or SSE 2.x extension
(CPUs 2005 and above, include Intel processors, p.e.:Pentium 4)
Radeon 5400 Series and above
Radeon 6400 Series and above
Mobility Radeon HD 5400 Series and above

SO

Windows 8.1
Windows 8
Windows 7
Windows Vista SP2
Windows XP (Only until ATI APP SDK 2.4)

URL

http://developer.amd.com/tools-and-sdks/opencl-zone/amd-accelerated-parallel-processing-app-sdk/

NVIDIA

Software

NVIDIA GPU Computing SDK 4.2

Compiler

Microsoft Visual Studio 2005 and above

Hardware

NVIDIA Geforce(R) 8, 9 y 200 series, and above
NVIDIA Tesla
Algunas NVIDIA Quadro®

SO
Windows 8.1
Windows 8
Windows 7
Windows Vista
Windows XP
URL
https://developer.nvidia.com/opencl

First Program

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