CS5412 / LECTURE 25 PROGRAMMING HARDWARE ACCELERATORS

Ken Birman

Spring, 2020

SUPPOSE THAT YOU PURCHASE A GPU. HOW WOULD YOU PROGRAM IT?

GPUs come with a collection of existing software libraries that mostly are coded in a language called CUDA. They implement “kernels”.

They define functions that require pointers to the data objects, which should be downloaded into GPU memory ahead of time. Then the GPU leaves its result in GPU memory too.

After the function finishes, you upload the result.

HARDWARE PROPERTIES

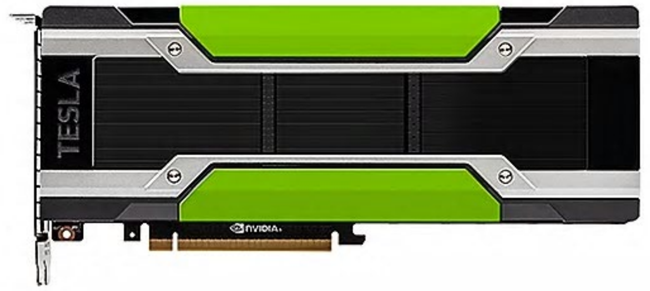
A GPU accelerator is a special device, purchased from a company like NVIDIA, that is designed to live "next to" a host computer.

For example, you might have a Dell T740 server (common in the cloud) and attach an NVIDIA Tesla T4 GPU accelerator to it (cutting edge).

We say that the Dell server is the "host" for the NVIDA GPU.

HARDWARE PROPERTIES

Dell T740, "blade" configuration (lives in a rack)



NVIDIA Tesla T4 plugs into the Dell T740

WHO CONTROLS WHAT?

Software in the device driver for the Tesla T4 lives in the Dell T740 and controls the GPU accelerator.

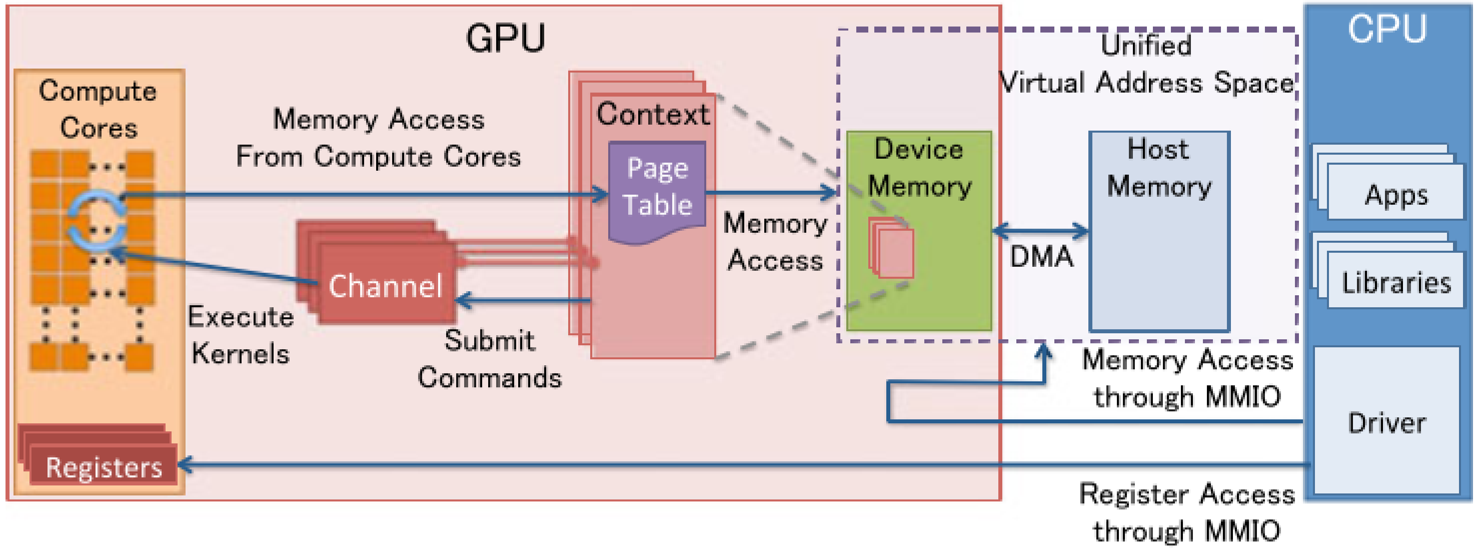
The host computer can load kernels into the GPU device.

… and can DMA transfer into or out of the GPU memory region.

… and can load parameters to the GPU functions, then press “run”. The GPU code will run, perform the task, and then interrupt the host.

… then the host can read the results out of the GPU memory area.

HERE WE SEE THE HOST ON THE RIGHT (CPU) AND THE GPU IS SHOWN ON THE LEFT



DEFINITION OF KERNEL

The term simply means “any function that runs on the GPU accelerator”

A GPU card is a full computer but typically lacks a real operating system. Instead, it only offers these functions, and moreover, all data movement in and out of the GPU is via the host doing DMA into or from GPU memory.

The host “manages” the GPU memory so that the GPU itself sees preallocated memory regions and can start computing as soon as the host computer says “go”. (This differs from gaming, where data lives mostly in the host memory.)

CUDA LIBRARIES ARE VERY EXTENSIVE!

Companies like NVIDIA have created huge libraries for the most common graphics tasks and for the most important machine learning algorithms.

Yet… they don't easily "adapt" if you have a variation on a standard method, or want to explore some completely new method.

This raises a question: Can we make it easier to program new kernel?

WE'LL DIVE DOWN ON TWO PAPERS

First, Dandelion, a Microsoft Research concept for programming a system in a high level language (C#) and then automatically mapping parts of the code into a GPU accelerator.

The slide set we'll look at was used for their talk at ACM Symposium on Operating Systems (SOSP) in 2013.

Dandelion builds on the same LINQ ideas we saw on April 9!

… SECOND DEEP DIVE

Dahlia is a more modern approach to this same question.

The work is occurring at Cornell: Adrian Sampson leads the project, and the slides are from a talk he gave very recently.

SUMMARY AND CONCLUSIONS

It is much easier to just use existing libraries and kernels if you can!

But if not, there is a growing range of software to help you program in a general language like C# or C++, then map your code automatically into a GPU device.

You won't find these solutions for Python: They need to work with a language that has "direct control" of data layouts and representations.