COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

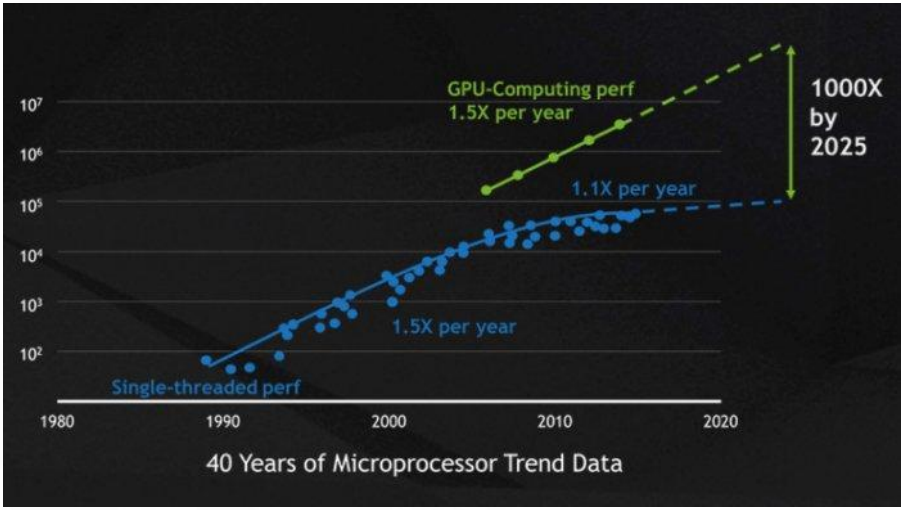
GPU Computing

comillas.edu

Contents

Introduction

GPUs (Graphics Processing Units) are processing units originally designed to render graphics on a computer quickly. This is done by having a large number of simple processing units for massively parallel calculations. The idea of general purpose GPU computing (GPGPU) is to exploit this capability for general computing.



Introduction

GPUs are usually from two main providers: Nvidia and AMD. Now Intel is coming into play with it with its ARC GPUs but for now the dedicated GPU market is divided between Nvidia and AMD.

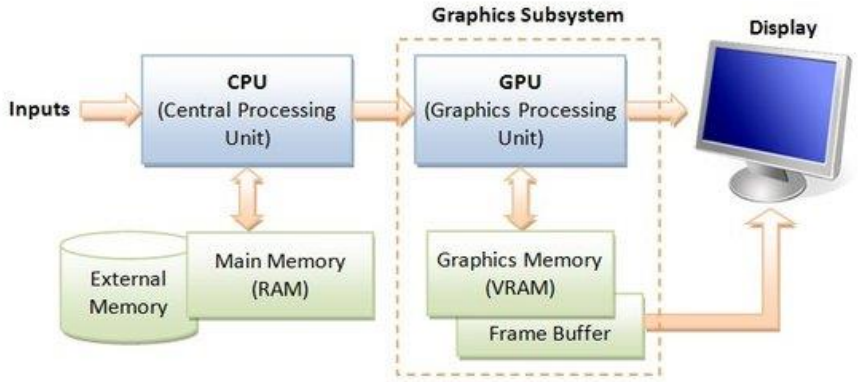
To use a GPU for general programming, you should use a specific platform/library to send code to the GPU:

- CUDA is a platform for programming on GPUs, specifically for NVIDIA GPUs, that allows you to submit C/C++/Fortran code for execution on the GPU.
- OpenCL is an alternative that works with a wider variety of GPUs.

Introduction

GPUs have many processing units but somewhat **limited memory**. Also, they can only use data in their own memory, not the CPU's memory, so data must be transferred between the CPU (the host) and the GPU (the device).

This copy can, in some calculations, constitute a very large fraction of the overall calculation. Therefore, it is best to create the data and/or leave the data (for later computations) on the GPU when possible and keep transfers as low as possible.

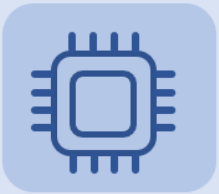


CPU, GPU, TPU and FPGA

- **Central Processing Unit (CPU)**

A central processing unit (CPU), also called a central processor, main processor or just processor, is the electronic circuitry that executes instructions comprising a computer program. The CPU performs basic arithmetic, logic, controlling, and input/output (I/O) operations specified by the instructions in the program.

- Easy to program – support any programming framework
- Fast design space exploration.
- Suited for simple models
- Avoid when training large models



CPU

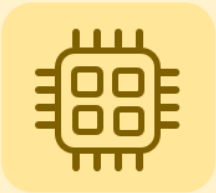
- Small models
- Small datasets
- Useful for design space exploration

CPU, GPU, TPU and FPGA

- Graphics Processing Unit (GPU)

As stated, GPUs are specialized processing units that were mainly designed to process images and videos. There are based on simpler processing units compared to CPUs but they can host much larger number of cores making them ideal for applications in which data need to be processed in parallel like the pixels of images or videos.

- Only programmable using CUDA, OpenCL or similar.
- Limited flexibility compared to CPUs



GPU

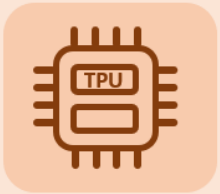
- Medium-to-large models, datasets
- Image, video processing
- Application on CUDA or OpenCL

CPU, GPU, TPU and FPGA

- **Tensor Processing Unit (TPU)**

A TPU is an Application-Specific Integrated Circuit (ASIC) developed by Google specifically for neural network machine learning. TPUs are very fast at performing dense vector and matrix computations and are specialized on running very fast program based on Tensorflow. That means that they have lower flexibility compared to CPUs and GPUs and they only makes sense to use them when it comes to models based on TensorFlow.

- Optimised for Tensorflow models
- Lower flexibility than GPUs



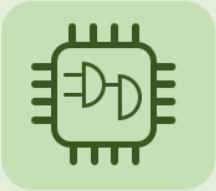
TPU

- Matrix computations
- Dense vector processing
- No custom TensorFlow operations

CPU, GPU, TPU and FPGA

- **Field Programmable Gate Array (FPGA)**

A FPGA is an integrated circuit designed to be configured by a customer or a designer after manufacturing. FPGAs have been emerged as a very powerful processing units that can be configured to meet applications requirements. They can achieve much higher performance, lower cost and lower power consumption compared to other options like CPUs and GPUs. However, they have to be preprogrammed to accelerate operations, offering very limited flexibility.



FPGA

- Large datasets, models
- Compute intensive applications
- High performance, high perf./cost ratio

General Description of GPU Computing

The basic sequence of operations to use a GPU is:

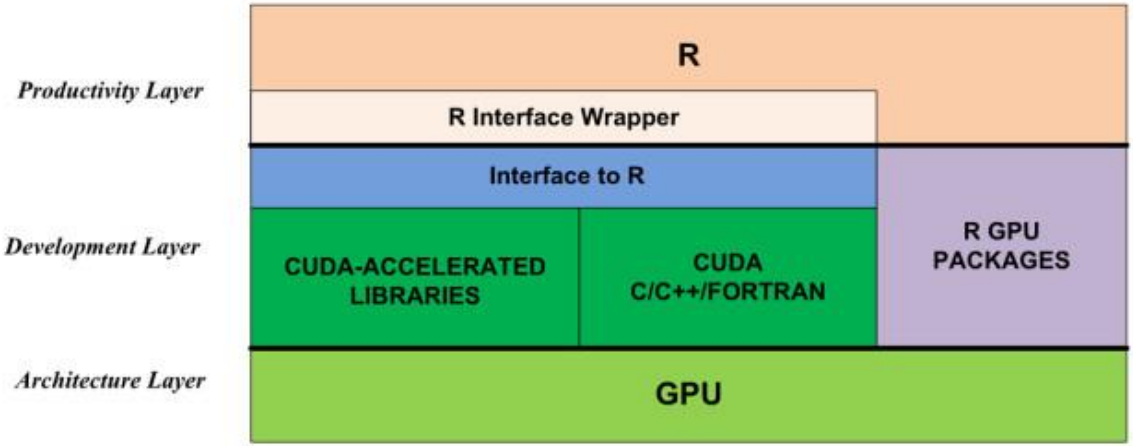
1. allocate memory on the GPU
2. transfer data from CPU to GPU
3. launch the CUDA kernel to operate on threads, with a block/grid specific configuration
4. (*optionally*) run another kernel, which can access data stored on the GPU, including the results of the previous kernel
5. transfer the results back to the CPU

Key calculations are performed in kernels, which are functions that encode core computing operations that run in parallel.

GPU-Accelerated R Software Stack

To use GPU in R:

1. use R GPU packages from CRAN; or
2. access the GPU through CUDA libraries and/or CUDA-accelerated programming languages, including C, C++ and Fortran.



Bibliography

- Gualán Saavedra, R. (2018). *Introducción a la Computación con GPUs usando R*. URL: https://bookdown.org/content/bda9bd24-47e7-4ecb-85e4-752a913954ce/
- Heller, M. (2018). *What is CUDA? Parallel programming for GPUs*. URL: https://www.infoworld.com/article/3299703/what-is-cuda-parallel-programming-for-gpus.html
- *Nvidia Developer Blog*. URL: https://developer.nvidia.com/blog/
- Determan, Ch. (2021). *gpuR package github repository.* URL: https://github.com/cdeterman/gpuR
- Determan, Ch. (2017). *A Short introduction to the gpuR package*. URL: https://mran.microsoft.com/snapshot/2018-08-15/web/packages/gpuR/vignettes/gpuR.pdf