

CUDA编程简介

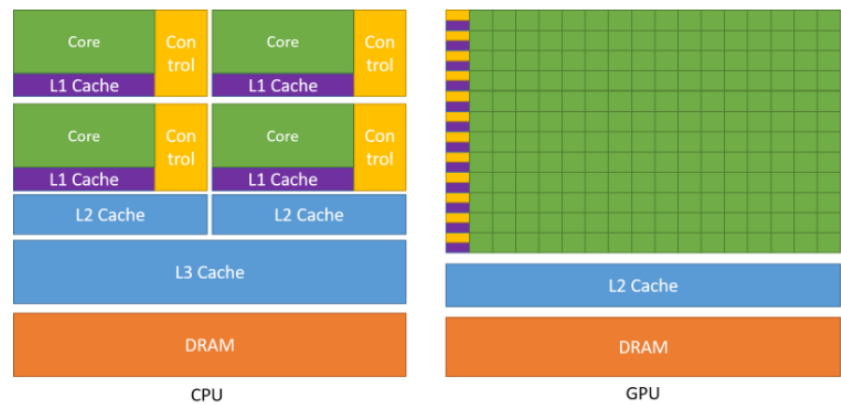
为什么用GPU？

图中所示的是CPU和GPU的架构。CPU的设计是为了处理串行操作（ sequential operations） , GPU则是用来处理并行操作（ parallel operations）

从图中可以看到，CPU相对GPU，由于其相对较多的控制流，所以其读取内存的延迟（ latency ）较低。GPU相对CPU，拥有较多的计算单元（ arithmetic logic unit ）但相对较少的控制流（ control flow ） ，但其吞吐量（throughout ）和带宽（ memory bandwidth ）大。

e.g., floating-point computations, is beneficial for highly parallel computations; the GPU can hide memory access latencies with computation, instead of relying on large data caches and complex flow control to avoid long memory access latencies, both of which are expensive in terms of transistors. --《CUDA C++ Programming Guide》

Figure 1. The GPU Devotes More Transistors to Data Processing



NVIDIA GPU 架构演变

从图中可以看到，我们所用的Tesla V100 是采用Volta架构，在2017年提出。近几年，随着更新，出现了Turling和 Ampere架构。7.x表示其性能。


GPU Computing Applications

Libraries and Middleware

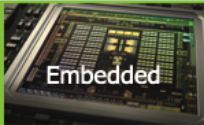



cuDNN TensorRT	cuFFT cuBLAS cuRAND cuSPARSE	CULA MAGMA	Thrust NPP	VSIPL SVM OpenCurrent	PhysX OptiX iRay	MATLAB Mathematica
-------------------	---------------------------------------	---------------	---------------	-----------------------------	------------------------	-----------------------

Programming Languages

C	C++	Fortran	Java Python Wrappers	DirectCompute	Directives (e.g. OpenACC)
---	-----	---------	----------------------------	---------------	------------------------------



CUDA-Enabled NVIDIA GPUs

NVIDIA Ampere Architecture (compute capabilities 8.x)				Tesla A Series
NVIDIA Turing Architecture (compute capabilities 7.x)		GeForce 2000 Series	Quadro RTX Series	Tesla T Series
NVIDIA Volta Architecture (compute capabilities 7.x)	DRIVE/JETSON AGX Xavier		Quadro GV Series	Tesla V Series
NVIDIA Pascal Architecture (compute capabilities 6.x)	Tegra X2	GeForce 1000 Series	Quadro P Series	Tesla P Series
	 Embedded	 Consumer Desktop/Laptop	 Professional Workstation	 Data Center

下面贴一张我们用到的Tesla V100 GPU 架构。其中计算单精度 (Floating Point 32) 的core有64个/SM，计算双精度 (FP64) 的core有32个/SM， 还有8个Tensor core/SM。

NVIDIA Volta™ 中的第一代 Tensor Core 专为深度学习而设计，通过 FP16 和 FP32 下的混合精度矩阵乘法提供了突破性的性能 – 与 NVIDIA Pascal 相比，用于训练的峰值 teraFLOPS (TFLOPS) 性能提升了高达 12 倍，用于推理的峰值 TFLOPS 性能提升了高达 6 倍。这项关键功能使 Volta 提供了比 Pascal 高 3 倍的训练和推理性能。



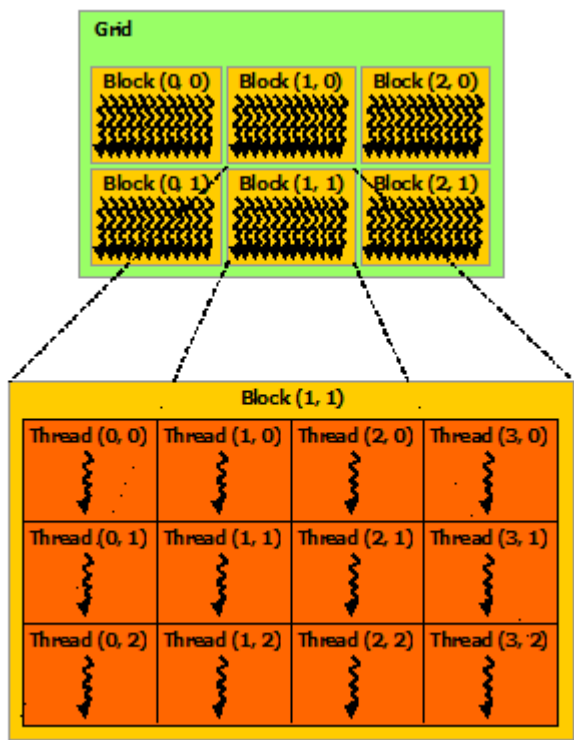
CUDA基础

首先，需要了解一些常用的术语。

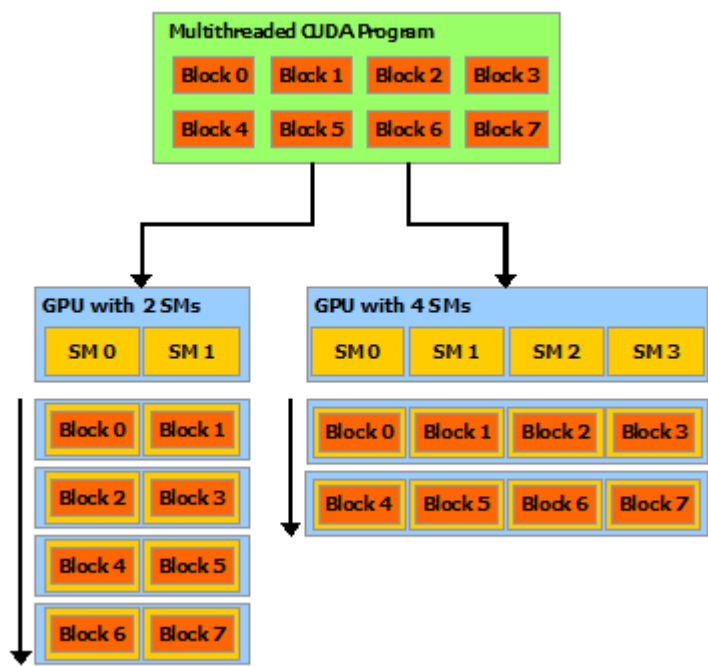
- Streaming Multiprocessors (SMs) - BLOCKs
- Streaming Processors (SPs) - THREADs
- Synchronous DRAM (SDRAM) - Global MEM
- Graphics Double Data Rate (GDDR) - GDDR SRAMs(used for graphics)
- High-Bandwidth Memory (HBM)
- Single Instruction Multiple Data (SIMD)

从上面的Tesla V100 架构图中，我们看到很多绿色的模块被打包成了一个SP，每个SM里包含了很多的SPs。下图中，很形象的解释了Thread，Block和Grid的关系。其中threadIdx有x,y,z三个属性，因此可以描述三维。

BlockIdx同理。BlockDim则描述每一Block中有多少Threads。



一般的，我们需要初始化blocks和threads，在核函数func<<<blocks, threads>>>()中。硬件会依据初始化可伸缩的划分编程模型，如下图。



Memory on GPU

- Registers (thread R/W)
- Local Memory (thread R/W)
- Shared Memory / Scratchpad memory (block R/W)
- Global Memory (grid R/W)
- Constant Memory (grid R only)

Unified Memory provides managed memory to bridge the host and device memory spaces. Managed memory is accessible from all CPUs and GPUs in the system as a single, coherent memory image with a common address space. This capability enables oversubscription of device memory and can greatly simplify the task of porting applications by eliminating the need to explicitly mirror data on host and device. --《CUDA C++ Programming Guide》

距离thread越近，读取速度越快，读取需要的时钟周期越少。

