DEVELOPER   🔍   Join   ⊗

Technical Blog   🔍 Search blog   Subscribe ›

Models / Libraries / Frameworks
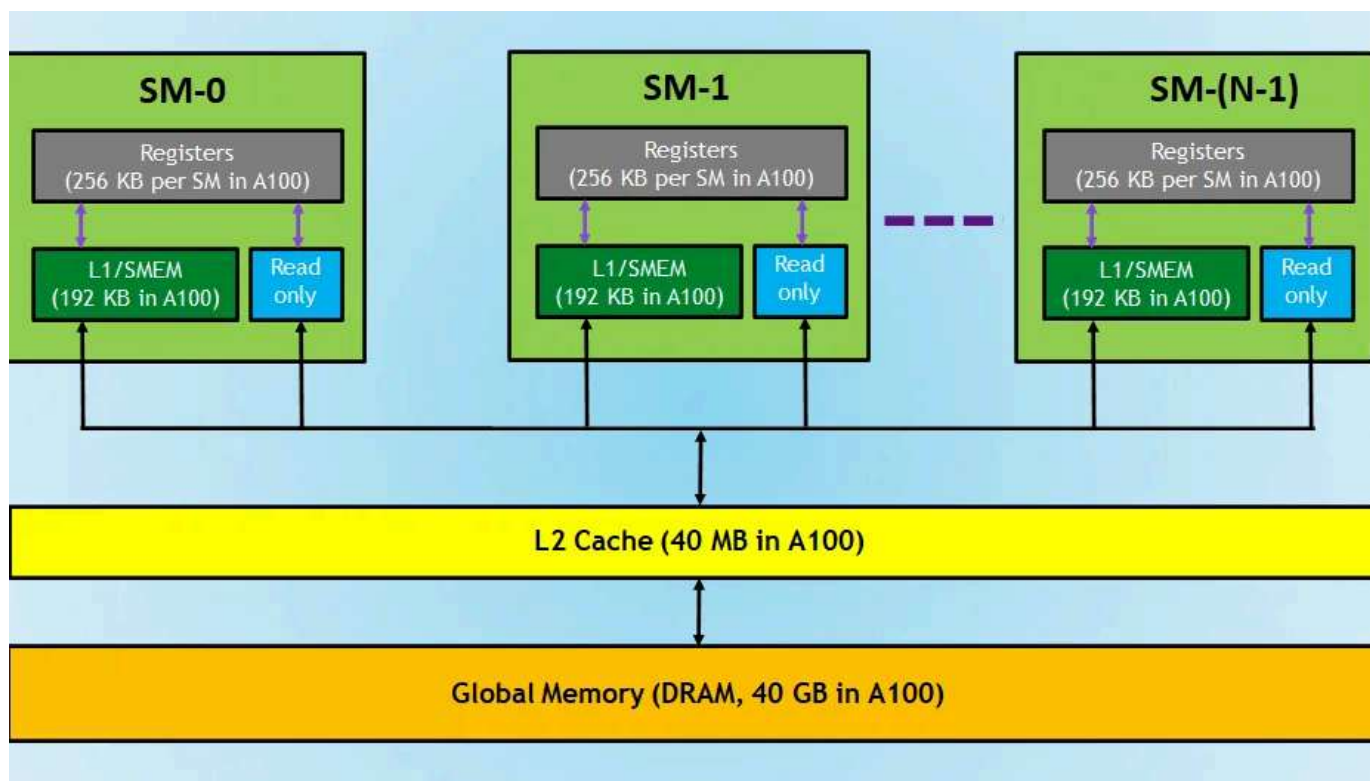
CUDA Refresher: The CUDA Programming Model

Jun 26, 2020   👍 +74 Like   💬 Discuss (2)

By Pradeep Gupta

⚛ **AI-Generated Summary**   ⌄

This is the fourth post in the CUDA Refresher series, which has the goal of refreshing key concepts in CUDA, tools, and optimization for beginning or intermediate developers.

outlines the main concepts of the CUDA programming model by outlining how they are exposed in general-purpose programming languages like C/C++.

Let me introduce two keywords widely used in CUDA programming model: *host* and *device*.

The host is the CPU available in the system. The system memory associated with the CPU is called host memory. The GPU is called a device and GPU memory likewise called device memory.

To execute any CUDA program, there are three main steps:

- Copy the input data from host memory to device memory, also known as host-to-device transfer.
- Load the GPU program and execute, caching data on-chip for performance.
- Copy the results from device memory to host memory, also called device-to-host transfer.

CUDA kernel and thread hierarchy

Figure 1 shows that the CUDA kernel is a function that gets executed on GPU. The parallel portion of your applications is executed K times in parallel by K different CUDA threads, as opposed to only one time like regular C/C++ functions.

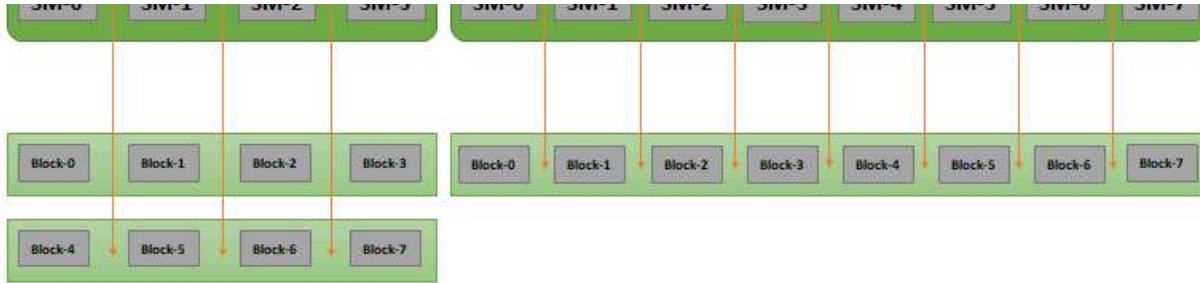*Figure 1. The kernel is a function executed on the GPU.*

Related posts
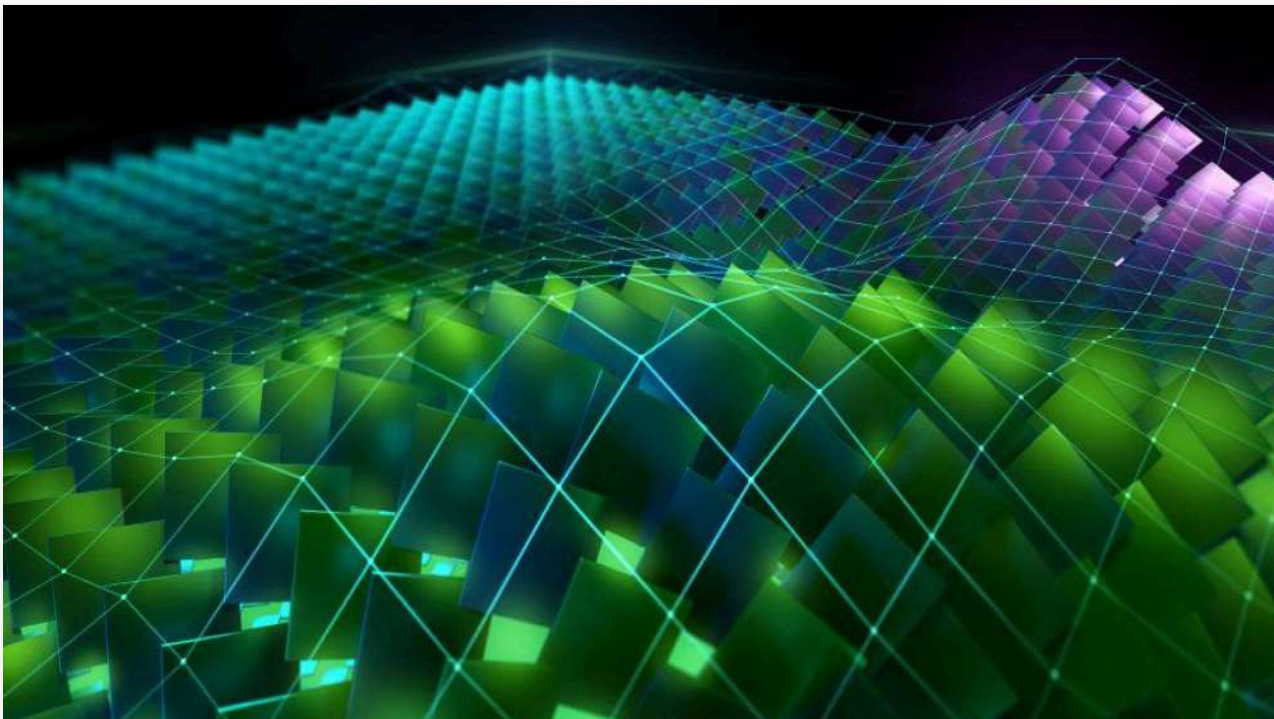
ed to
he SM
cks.
evice
able

Exploring the New Features of CUDA 11.3

CUDA Refresher: Getting started with CUDA

How to Access Global Memory Efficiently in CUDA Fortran Kernels

built-in

ts in

__syncthreads, all threads in the block must wait before anyone can proceed.

- The number of threads per block and the number of blocks per grid specified in the <<<...>>> syntax can be of type int or dim3. These triple angle brackets mark a call from host code to device code. It is also called a kernel launch.

The CUDA program for adding two matrices below shows multi-dimensional blockIdx and

or ease

mber

An Easy Introduction to CUDA C and C++

```
                                                    / thr
eadsPerBlock.y);
    MatAdd<<<numBlocks, threadsPerBlock>>>(MatA, MatB, MatC);
```

An Easy Introduction to CUDA Fortran

Figure 4. Memory hierarchy in GPUs.

The following memories are exposed by the GPU architecture:

L1 cache and shared memory. All threads in a CUDA block can share shared memory, and all CUDA blocks running on a given SM can share the physical memory resource provided by the SM..

- **Read-only memory**—Each SM has an instruction cache, constant memory, texture memory and RO cache, which is read-only to kernel code.
- **L2 cache**—The L2 cache is shared across all SMs, so every thread in every CUDA block can access this memory. The NVIDIA A100 GPU has increased the L2 cache size to 40 MB as compared to 6 MB in V100 GPUs.
- **Global memory**—This is the framebuffer size of the GPU and DRAM sitting in the GPU.

The NVIDIA CUDA compiler does a good job in optimizing memory resources but an expert CUDA developer can choose to use this memory hierarchy efficiently to optimize the CUDA programs as needed.

Compute capability

The compute capability of a GPU determines its general specifications and available features supported by the GPU hardware. This version number can be used by applications at runtime to determine which hardware features or instructions are available on the present GPU.

Every GPU comes with a version number denoted as X.Y where X comprises a major revision number and Y a minor revision number. The minor revision number corresponds to an incremental improvement to the architecture, possibly including new features.

For more information about the compute capability of any CUDA-enabled device, see the CUDA sample code deviceQuery. This sample enumerates the properties of the CUDA devices present in the system

Summary

The CUDA programming model provides a heterogeneous environment where the host code is running the C/C++ program on the CPU and the kernel runs on a physically separate GPU device. The CUDA programming model also assumes that both the host and the device maintain their own separate memory spaces, referred to as host memory and device memory, respectively. CUDA code also provides for data transfer between host and device memory, over the PCIe bus.

DEVELOPER   🔍   [Join]

shared memory and L1 cache, L2 cache, and global memory. Advanced developers can use some of these memories efficiently to optimize the CUDA program.

💬 **Discuss (2)**   👍 **+74 Like**

Tags

Models / Libraries / Frameworks | General | CUDA | Beginner Technical | CUDA Refresher | Parallel Programming

About the Authors

About Pradeep Gupta

Pradeep Gupta is director of the Solutions Architecture and Engineering team at NVIDIA. He is responsible for running technical customer engagements for industries like autonomous driving, healthcare, and telecoms where AI is transforming many possible aspects of industry solutions. His focus is on building production-grade AI that can be deployed in life-critical systems. Previously, Pradeep worked in areas like high-performance computing, computer vision, mathematical library development, and data center technologies. He received a master's degree in research from the Indian Institute of Science (IISc), Bangalore. His research focused on developing compute-efficient algorithms.

Follow @pkgnvi on Twitter

View all posts by Pradeep Gupta ›

Comments

Notable Replies

January 17, 2023

urumican1