LESSON 2

Hardware Features

TOPICS

- The way to know the features of the GPU

- Key words: *cudaGetDeviceCount, cudaDeviceProp, cudaGetDeviceProperties*

What's the heart of CUDA paradigm?

What's the heart of CUDA paradigm?

- Performance and scalability through the partition of the problem.
- The thread blocks provide the mapping between the parallelism of the application and the massive replication of the GPU hardware.
- Massive GPU hardware parallelism is achived through replication of a common architectural building block called *streaming multiprocessor (SM)*.
- The software abstraction of a thread block translates into a natural mapping f the kernel into an arbitrary number of SM on the GPU.

- Thread blocks also act as container of thread cooperation, only threads in a thread blocks can share data.

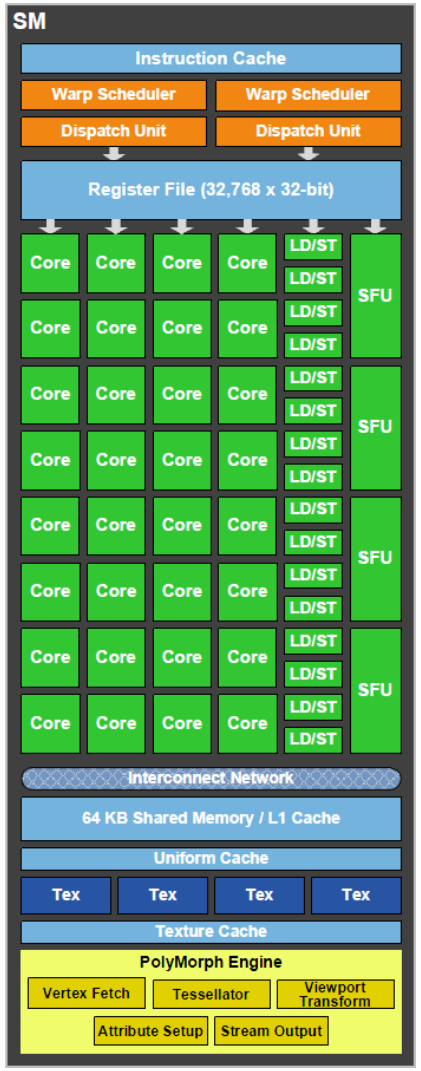
- The translation of the thread block abstraction is simple, each SM can be scheduled to run one or more blocks.

- The challenge for the CUDA programmer is to express their application kernels in such a way to exploit this parallelism.


Host Interface
GigaThread Engine
GPC
Raster Engine
SM
Polymorph Engine
Memory Controller
L2 Cache
BenchmarkReviews.com

- Distributing work to the SM is the job of the GigaThread global scheduler.
- Based in the number of blocks and number of threads per block defined in the kernel execution configuration, this scheduler allocates one or more blocks each SM.
- How many blocks are asigned to each SM depends of how many resident threads and thread blocks a SM can support.
- Each SM is responsable for scheduling its internal resources, cores and other execution units to perform the work of the threads in its assigned thread blocks.
- Nvidia categorizes their CUDA-enable devices by compute capability.

- Depending on the hardware, each SM has Single Instruction Multiple Data cores (SIMD).

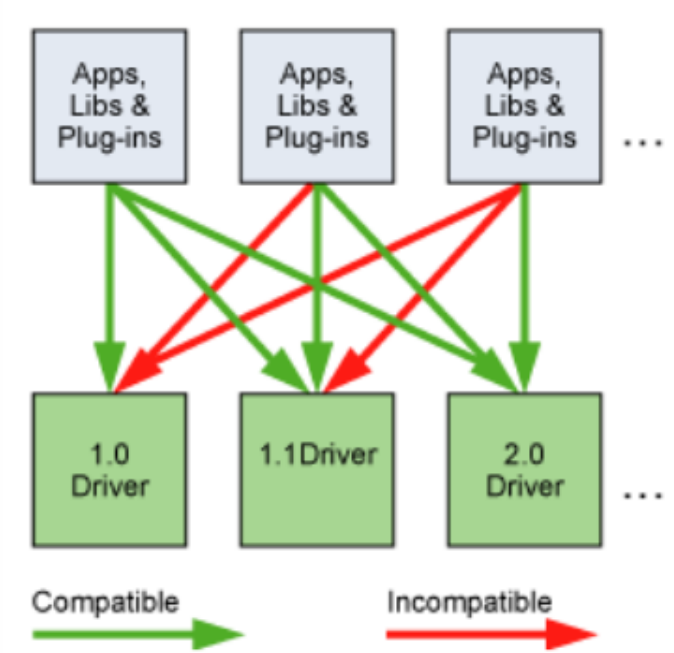


Compute Levels

- The *compute capability* of a device is defined by a major revision number and a minor revision number.
- Devices with the same major revision number are of the same core architecture.
- The major revision number is 3 for devices based on the *Kepler* architecture, 2 for devices based on the *Fermi* architecture, and 1 for devices based on the *Tesla* architecture.
- The minor revision number corresponds to an incremental improvement to the core architecture, possibly including new features.
- https://developer.nvidia.com/cuda-gpus
- http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#compute-capabilities

- The version of the driver API allows developers to check whether their application requires a newer device driver than the one currently installed.

- This is important, because the driver API is *backward compatible*, meaning that applications, plug-ins, and libraries (including the C runtime) compiled against a particular version of the driver API will continue to work on subsequent device driver releases



Device Query

- In the last practices we have made codes that you can run in any GPU with few cores or with a lot of them, but for your future applications you will need to explode the GPU to take advantage of the massive parallel programming, and to do this, we need to know the features of the GPU we are going to use.

- On one hand we can have two GPUs in our motherboard, but just one can be used at time; on the other hand we can have two GPUs that we can use at the same time or maybe we can use the one with the best capabilities.

- To know how many devices we have, we will use the *cudaGetDeviceCount*(), and there is an structure that provide us with the features of the CUDA device, *cudaDeviceProp*.

- To fill up the *cudaDeviceProp*, we need to invoke the *cudaGetDeviceProperties*() function.

```
cudaError_t cudaGetDeviceProperties( struct cudaDeviceProp * prop,  int device)

prop – the structure with the properties of the specified device
device – device number to get properties for
```

- The complete description of the device properties can be found in:

  http://docs.nvidia.com/cuda/cuda-runtime-api/index.html

Example

- 01device_query: This example shows part of the features of the GPU.

Summary