GPGPU programming
General-purpose Processing on Graphics Processing Units

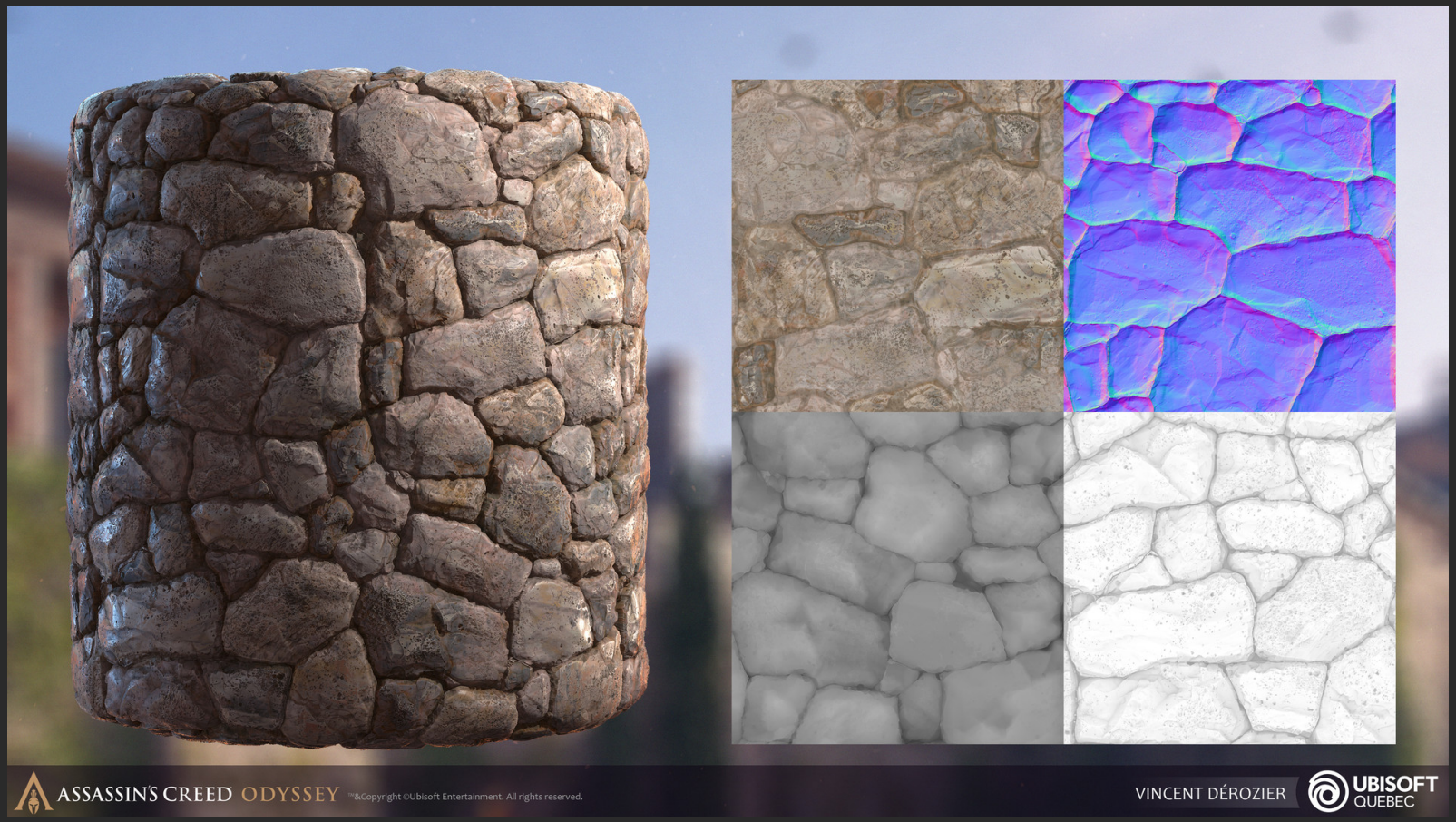
Robin Faury
robinfaurypro@gmail.com
faury@adobe.com

12-11-2019

Introduction

- The purpose of parallel processing
- What is a graphic card?
- The CUDA language
- GPGPU usage in the industry
- Q&A

Substance by Adobe



PBR render and its maps

Software suite for digital material creation.

The purpose of parallel processing

Moore's Law

Every two years, the density of transistors in an integrated circuit doubles. That means we can compute the critical path of an algorithm faster.

To infinity and beyond!

Critical path

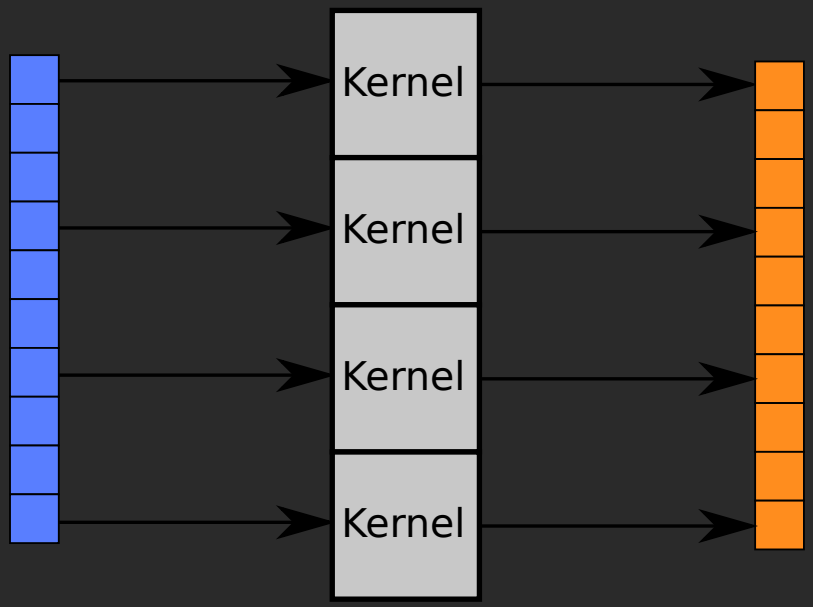
Sometimes, algorithms process data one by one. When applicable, it is necessary to find the critical path and execute it in parallel. Modern CPUs offer the ability to run some threads at the same time. However, CPUs don't have a lot of cores available. For massive parallel computation we will use GPUs.

Latency and Throughput

Latency: This is the time between an action and the response to this action. For example, a key press event and its process.
Throughput: This is the rate of production. The number of pixel processed in one second.
The aim of **CPU** is to be very responsive. For that they adopt strategies to hide the latency (pre-fetch, branch prediction, out of order execution...). However, these algorithms need a lot of memory cache.
The aim of **GPU** is to process a lot of data. That why there is way more cores into GPU than CPU. In this case, cores must share the memory cache. Therefore, GPUs have a high throughput but a high latency too.

A world of buffers

The aim of parallel computing is solving heavy arithmetic computation on buffer. One process is called a kernel for the GPGPU or a shader for the graphics pipeline.



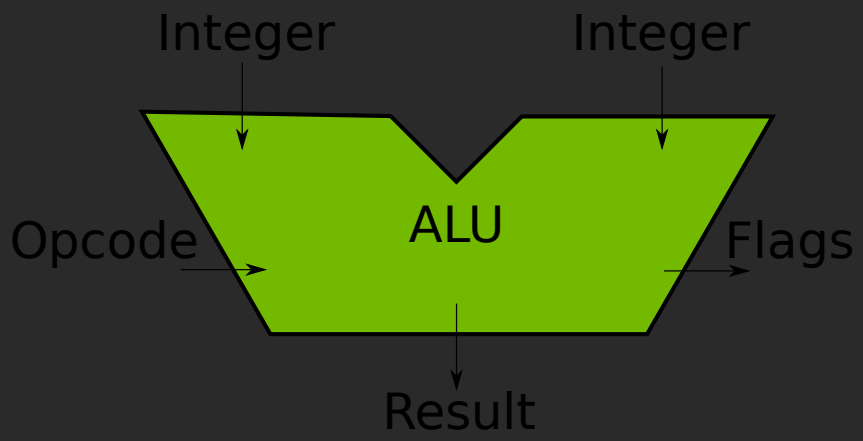
What is a graphic card?

History

The first Graphics Processing Unit (GPU) was used for drawing game sprites. It was a dedicated device for formatted data. Ten years after we had the ability to draw lines, fill areas and control the blitter. In 1990, the graphical API appears and allows us to send assembly code to the device.

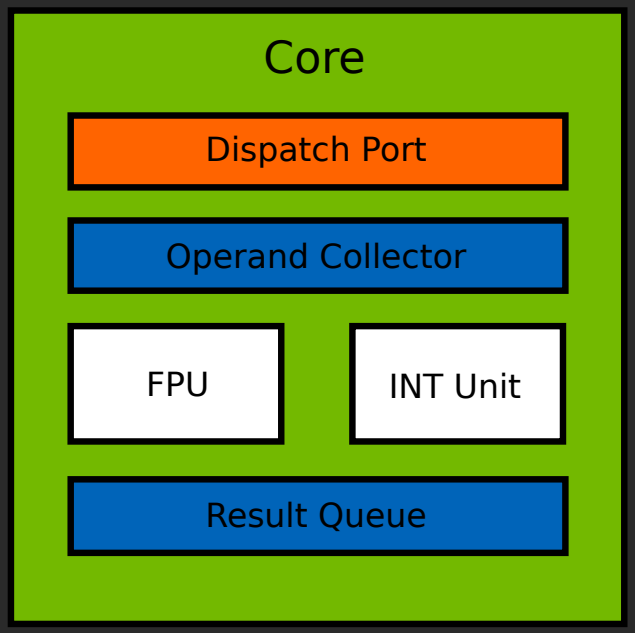
Arithmetic Logic Unit

The Arithmetic Logic Unit (ALU) is the component that performs arithmetic operations. The GPU is more focused on floating point operations, multiple ALUs are combined to create a Floating Point Unit (FPU).



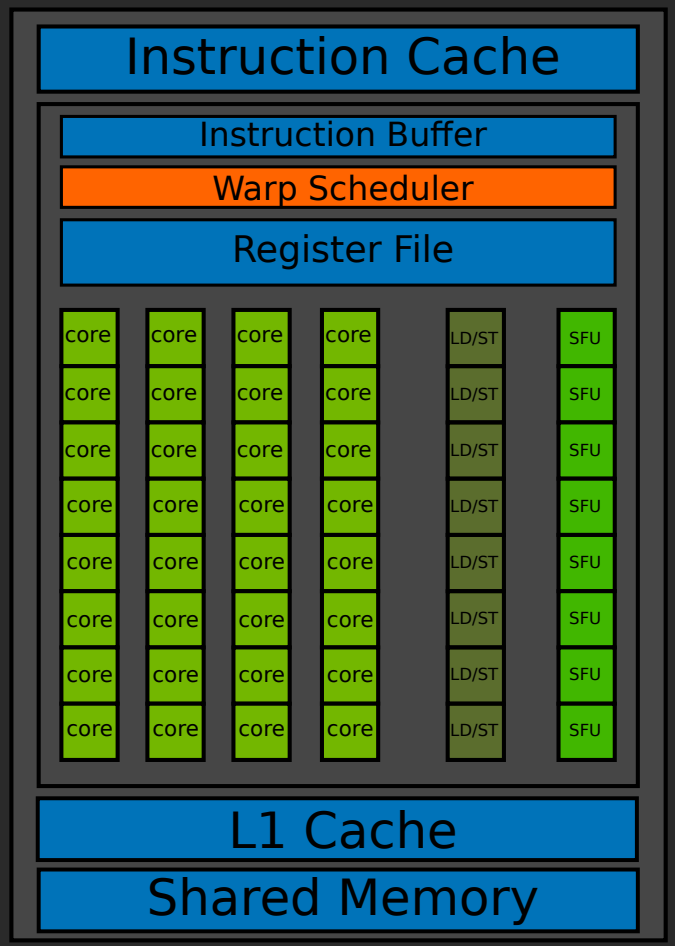
Core

Cores are used to execute opcodes from compiled kernels. There are composed of an FPU, logic unit, branch unit and compare unit.



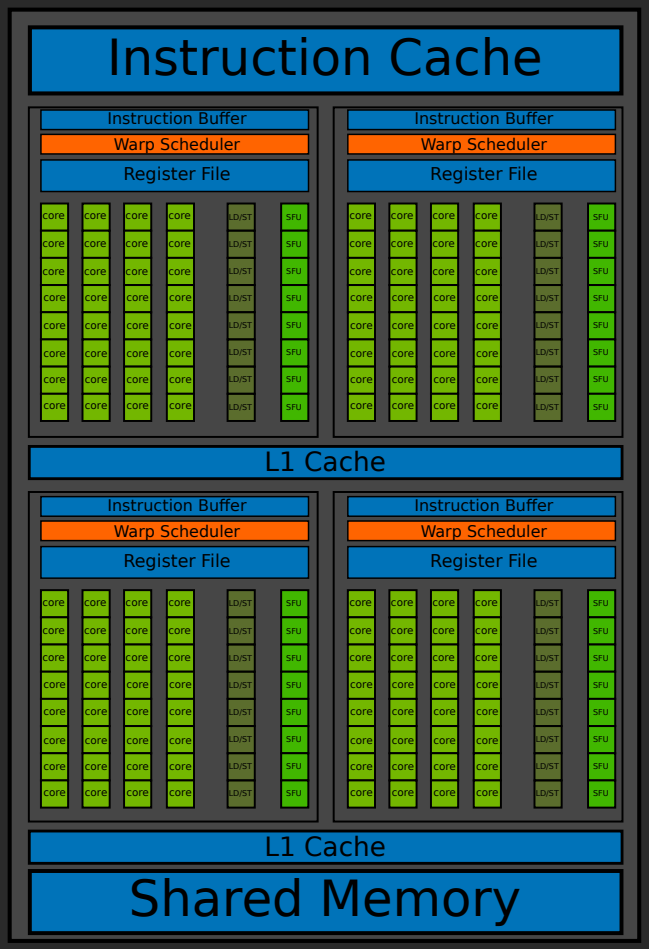
Streaming Multiprocessor

The Streaming Multiprocessor (SM) organizes threads in groups of 32 called warp. This architecture is called SPMD (Single Program, Multiple Data).



Streaming Multiprocessor

On the GP104 (The GPU of GTX 1080) each SM has four warps.

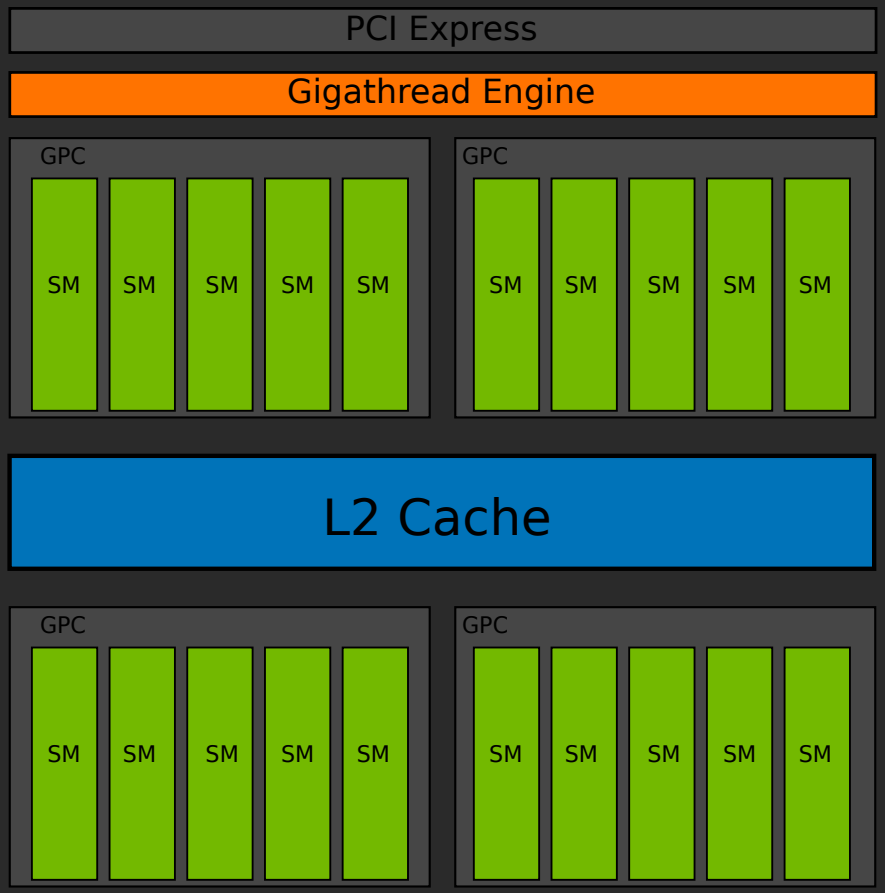


Graphics Processing Clusters

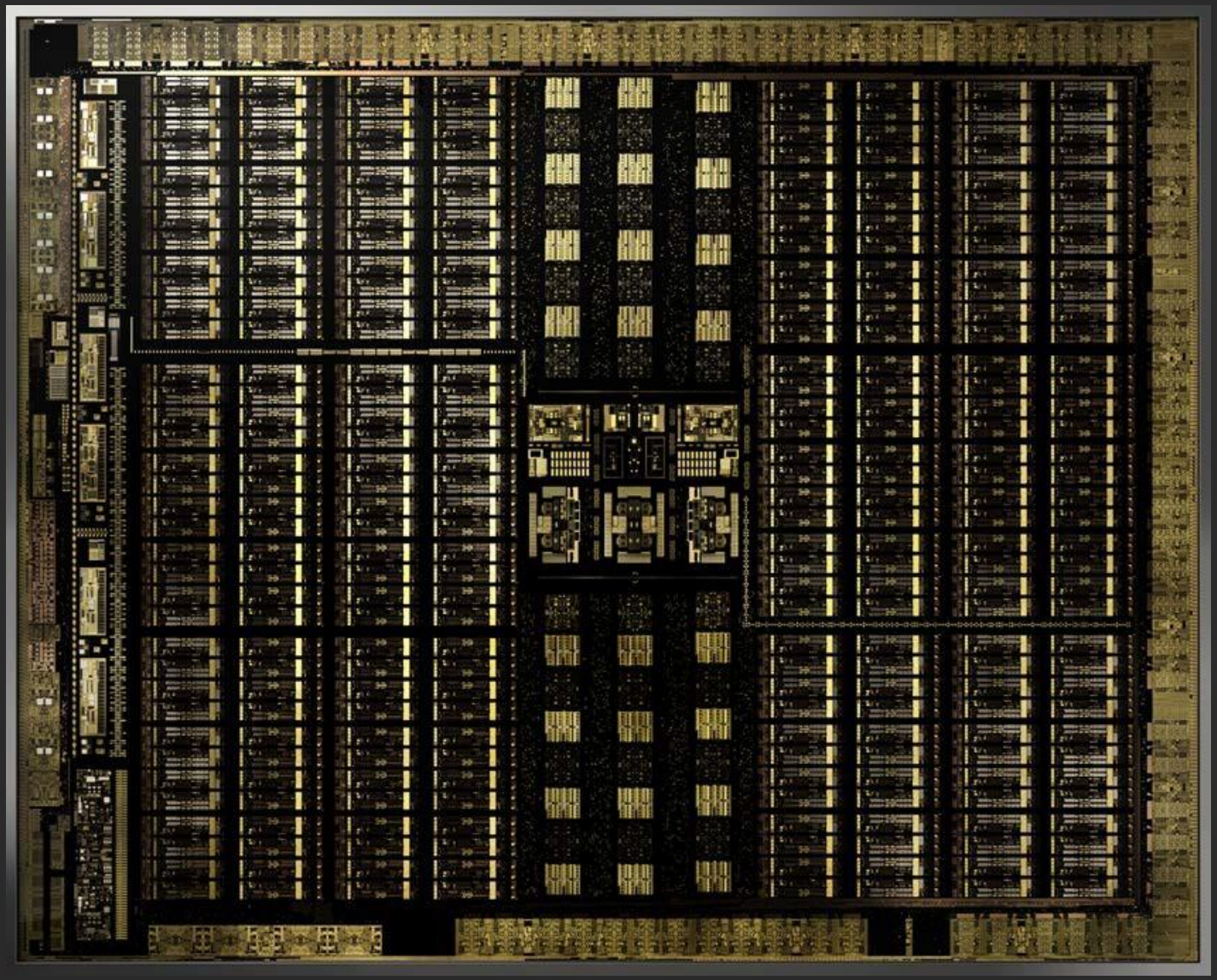
A Graphics Processing Clusters (GPC) is a collection of streaming multiprocessors. In the case of the GP104, there are four clusters.

GP104

All the GPC are connected to the L2 cache memory. The Gigathread engine distributes block threads to streaming multiprocessor. This device has 32 cores * 4 warps * 5 SMs * 4 GPCs = 2560 CUDA cores.



TU102



TU102


PCI Express 3.0 Host Interface
GigaThread Engine
Memory Controller
GPC
Raster Engine
TPC
PolyMorph Engine
SM
RT CORE
L2 Cache
High-Speed Hub
NVLink – Two x8 Links

GPGPU languages

Languages

To communicate with the graphic card we need languages. Each one have its pro and con.

- **CUDA:** The Nvidia language. Very close to the C++, easy to use. Works only on Nvidia card.
- **DirectX:** The Window language. Design for game development on Window.
- **Metal:** The Apple language. Design for graphic development on MacOS.
- **OpenGL:** Developed by Kronos, design for all OS. Deprecated from 2019.
- **Vulkan:** Developed by Kronos, design for all OS. Launched on 2018. Difficult to use.

Host and Devices

GPGPU programming is similar to web development. The CPU is the server and GPUs clients.

- Host: The CPU and its memory. The host can manage the memory on both the host and the device. The executed code can launch kernels.
- Devices: The GPU and its memory. Kernels are executed on many GPU threads in parallel.

Indexing

In the kernel, the thread Id, the block Id and the blockDim allow the user to compute the unique thread id.

```
int index = blockIdx * blockDim + threadIdx;
```

If data is stored into 2D or 3D array, it is possible to launch the kernel using a 3d vector instead of an integer and the index becomes:

```
int x = blockId.x * blockDim.x + threadId.x;
int y = blockId.y * blockDim.y + threadId.y;
int z = blockId.z * blockDim.z + threadId.z;
```

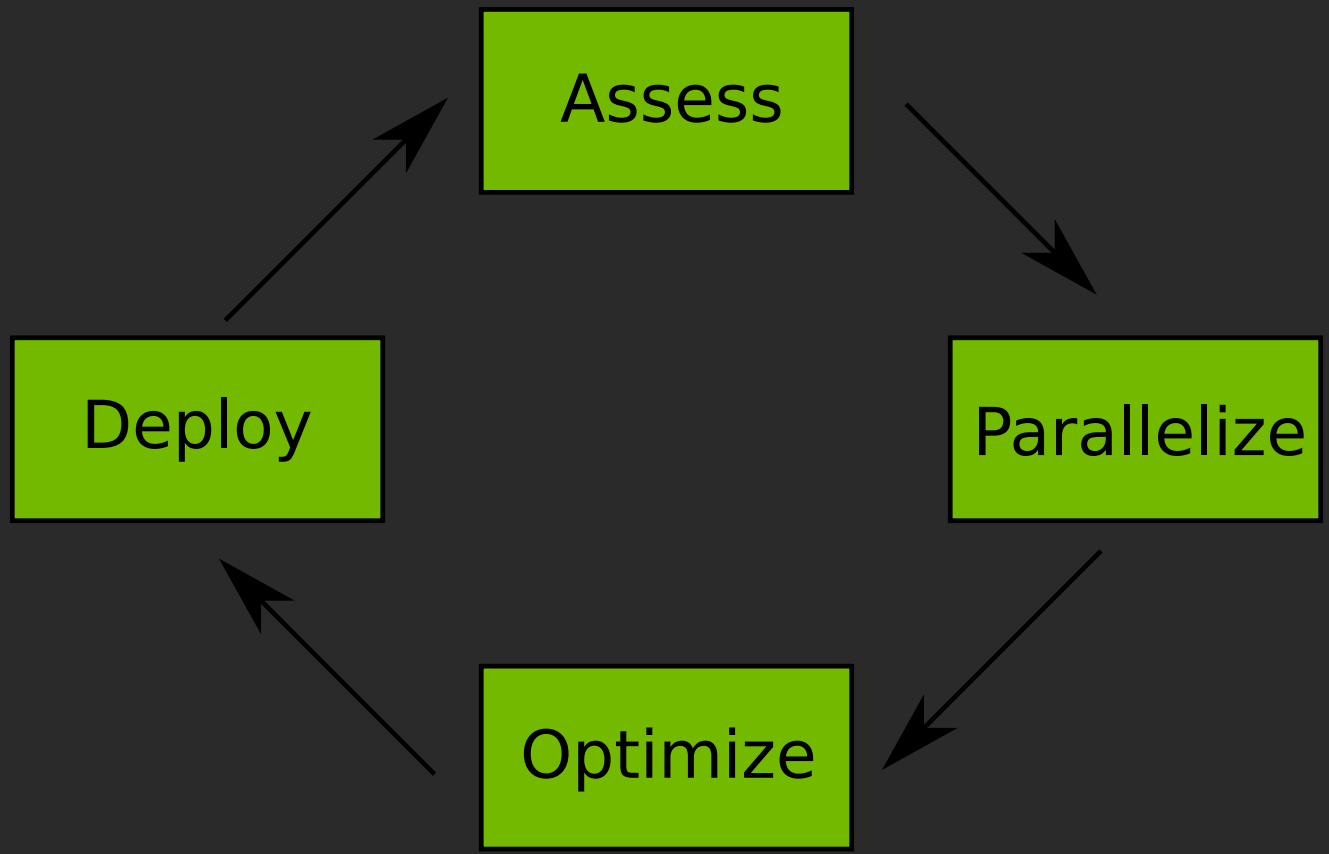
A Vulkan example

```
layout(std430, binding = 0)
    buffer layer { vec3 array[]; };

uniform Viewport {
    uint width;
} viewport;

void main() {
    const uint index =
        viewport.width*
        gl_GlobalInvocationID.y+
        gl_GlobalInvocationID.x;
    array[index] = vec3(1.0, 0.0, 0.0);
}
```

GPGPU usage in the industry

APOD

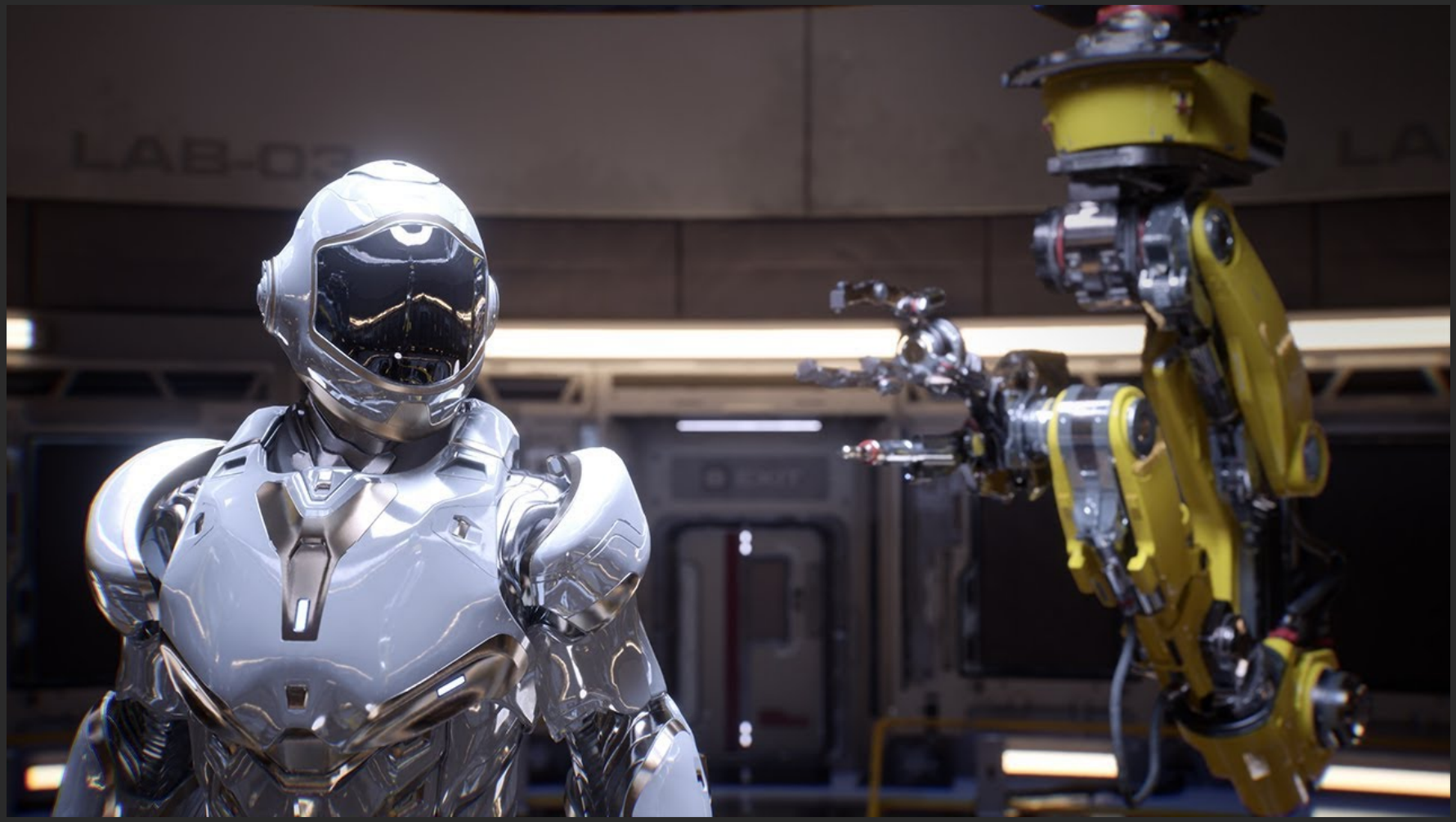
The Assess, Parallelize, Optimize, Deploy (APOD) design cycle's goal is to identify and correct bottlenecks into the application.



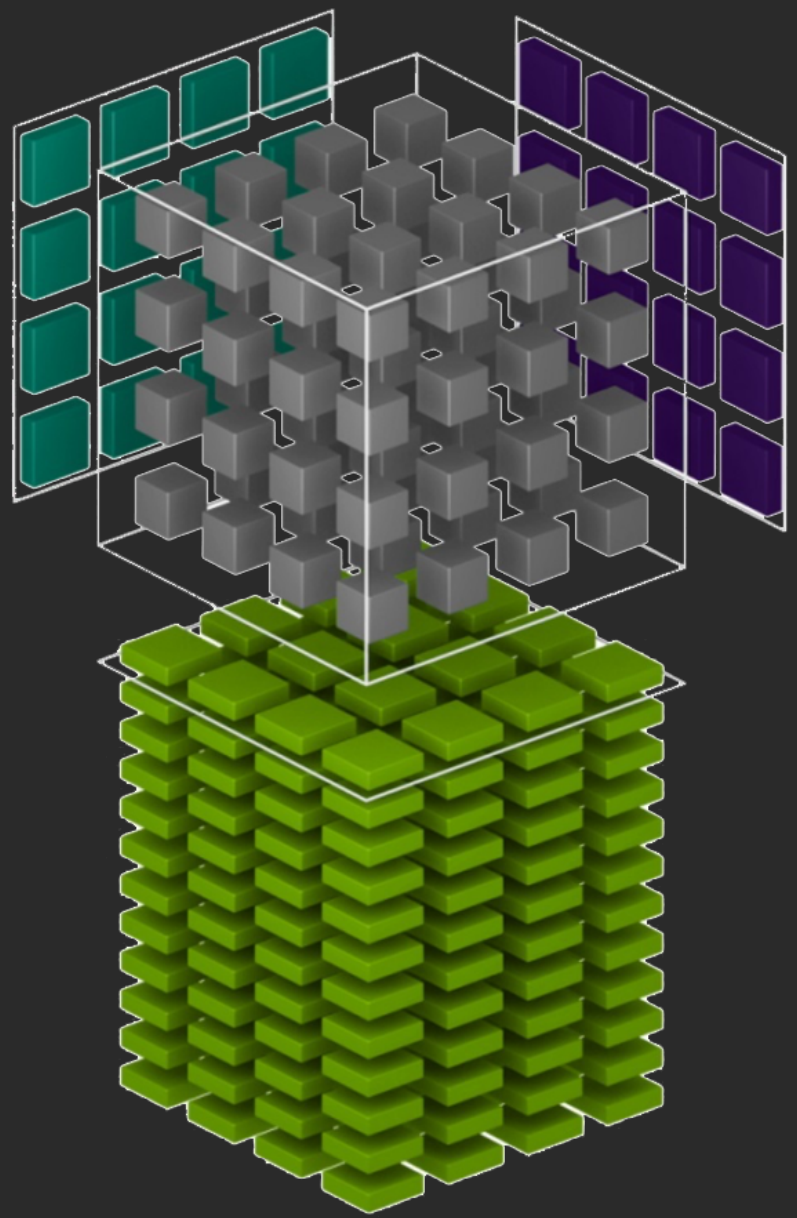
New devices


SM
Warp Scheduler + Dispatch (32 thread/clk)
Register File (16,384 x 32-bit)
INT32
FP32
TENSOR CORES
LD/ST
SFU
96KB L1 Data Cache / Shared Memory
Tex
RT CORE

RT cores



tensor cores



Domain Specific

- Deep Learning
- Linear Algebra and Math: Solver, Random function, Finite element method, etc...
- Signal
- Image and video
- Data oriented algorithm

Q&A