CUDA Lab

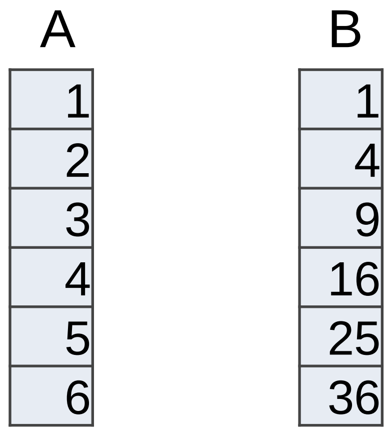
CUDA Basics

In CUDA we usually launch many threads in groups of thread blocks that form a grid.

[GPU Fundamentals]

[local]

The Basic CUDA Task



We have 2 Vector and want to add the vectors.

In C we would code:

```
for( I = 0; i<6; i++) {
         c[i]=a[i] + b[i]
}
```

CUDA Threads

Threads: Single execution units that run kernels on the GPU. Similar to CPU threads but there's usually many more of them.

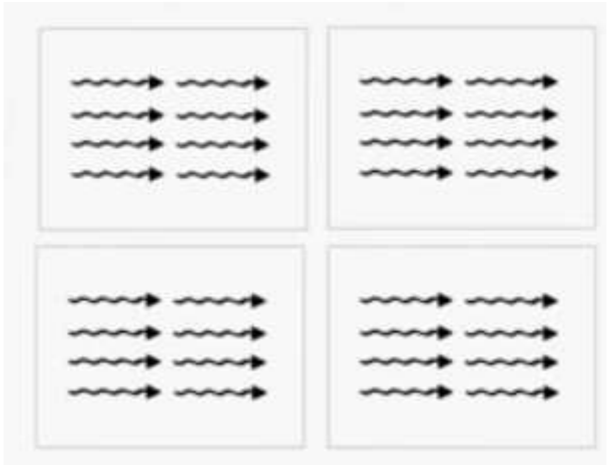
They are sometimes drawn as arrows:

Thread Blocks

Thread Blocks: Thread blocks are a collection of threads. All the threads in any single thread block can communicate.

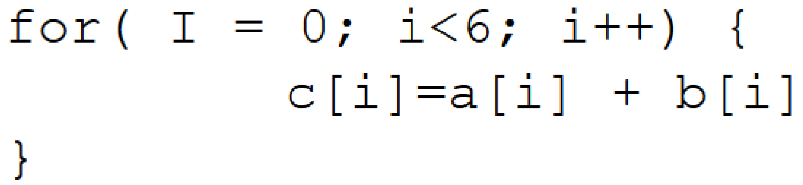
The Grid

Grid: A kernel is launched as a collection of thread blocks called the grid.



The Kernel

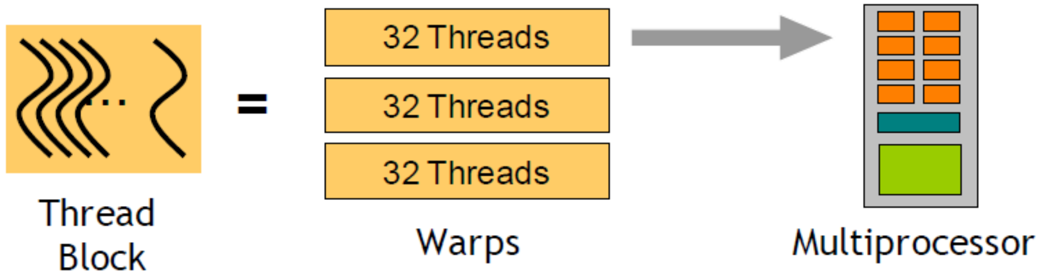
Kernel: A peace of Code that is executed in parallel threads in the Grid



```
for( I = 0; i<6; i++) {
            c[i]=a[i] + b[i]
}
```

From GPU Fundamental

Warps



A thread block consists of 32-thread warps

A warp is executed physically in parallel (SIMT) on a multiprocessor

Kernel Invocation

The host calls a kernel using a triple chevron <<< >>>„ In the chevrons we place the number of blocks and the number of threads per block. The following would launch 100 blocks of 256 threads each (total of 25600 threads):

Some Kernel<<<100, 256>>>(...);

You can start to see the difference between the GPU and the CPU. I don't know what would happen if you launched 25600 threads on a CPU but it wouldn't be good

DIM3D Data Structure

Dim3 is a 3d structure or vector type with three integers. x. y and z. You can initialize as many of the three coordinates as you like:

```
dim3 threads(256);  // Initialize with x as 256, y and z will
                    // both be 1

dim3 blocks(100,  100); // Initialize x and y, z will be 1

dim3 anotherOne(10, 54, 32); // Initialises all three values, x
                             // will be 10, y gets 54 and z
                             // will be the 32.
```

Each Thread can be individual referenced

Each of the running threads is individual,
they know the following:

ThreadIdx:	Thread index with.in the block
blockIdx:	Block index within the grid
blockDim:	Block dimensions in threads
gridDim:	Grid dimensions in blocks

Each of these are dim3 structures and can be read in the kernel to assign particular workloads to any thread.

Some Maxima (depends on CUDA Version)

Threads per Block : 1024

Blocks per Launch : $2^{32} – 1$

So 67,108,864 are possible even on old Cards

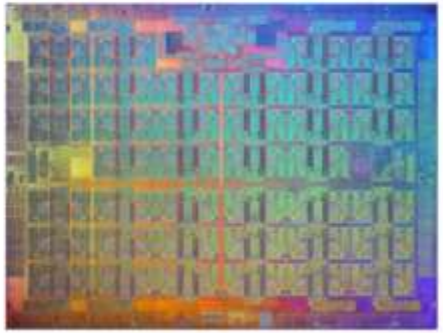
pycuda

Best source to learn

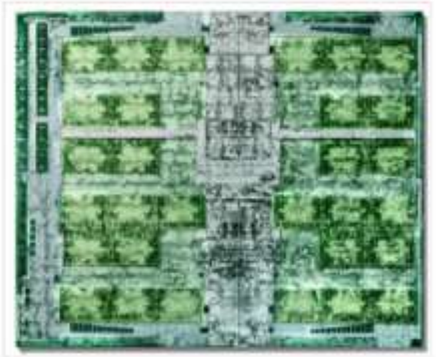
https://Github/inducer
CUDA integration for Python, plus shiny features

Graphcore Neuromorphic Computing

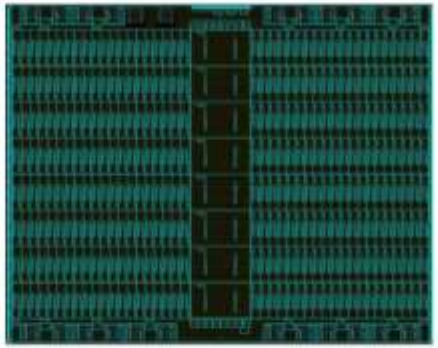
INTELLIGENCE REQUIRES A NEW ARCHITECTURE



CPU



GPU



IPU

Scalar

Designed for office apps Evolved for web servers

Vector

Designed for graphics Evolved for HPC

Graph

Designed for intelligence