CuPy

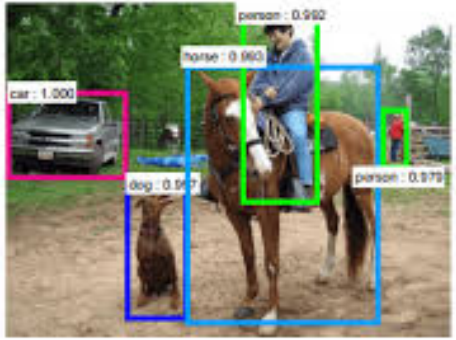
GPU computing beyond model training

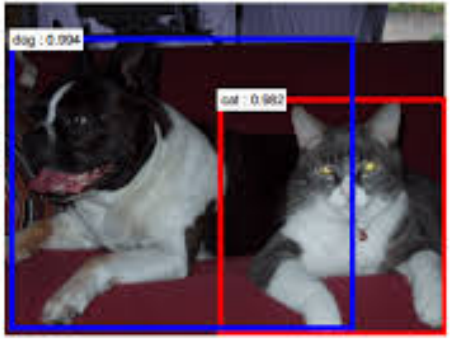
Mythbusters Demo GPU

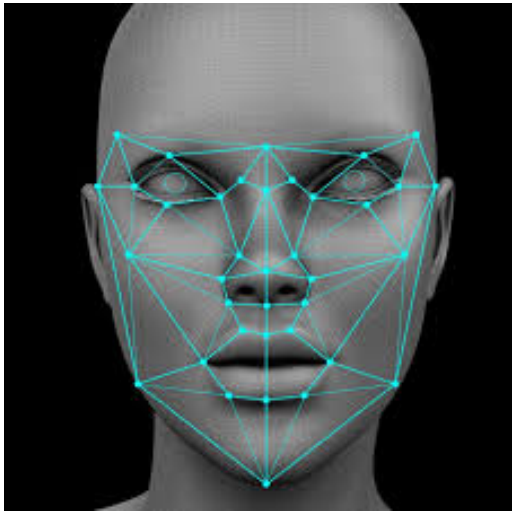
What is a GPU?

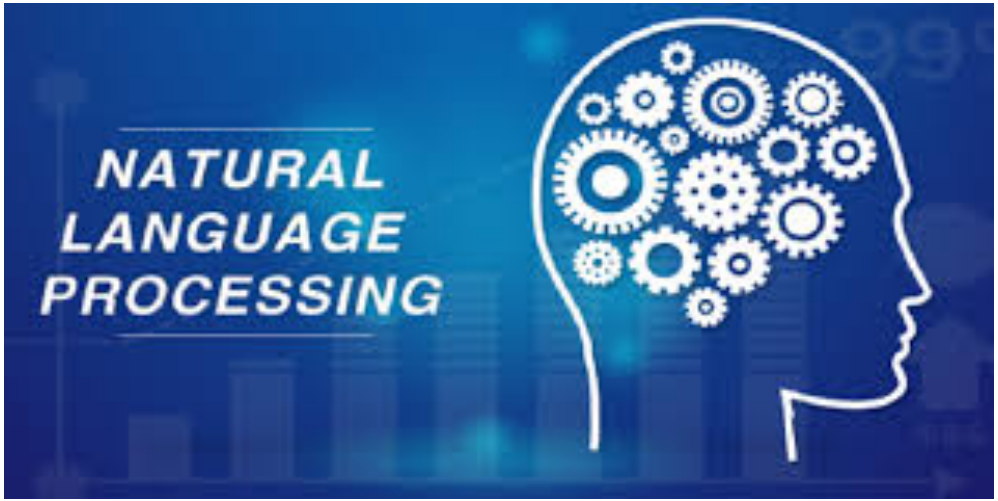
GPU = Graphics Processing Unit = Graphics Card

GPGPU = General Purpose GPU computing


person : 0.992
horse : 0.993
car : 1.000
person : 0.979

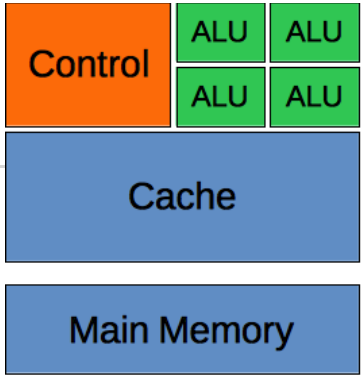

dog : 0.994
cat : 0.982




NATURAL
LANGUAGE
PROCESSING

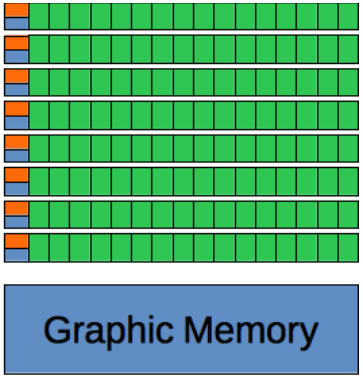

Autonomous Car

CPU vs GPU



CPU

- more sophisticated caching/pipeline schemes
- Faster instruction cycles
- Better at serial computation
- larger instruction set
- more complex arithmetic-logic unit (ALU)



GPU

- GPU is a special-purpose processor, designed so that a single instruction works over a large block of data (SIMD/Single Instruction Multiple Data)
- GPU excels at doing computation on a large set of data
- Better at parallel computation
- Many more compute cores (ALU)

This guy is climbing a mountain:

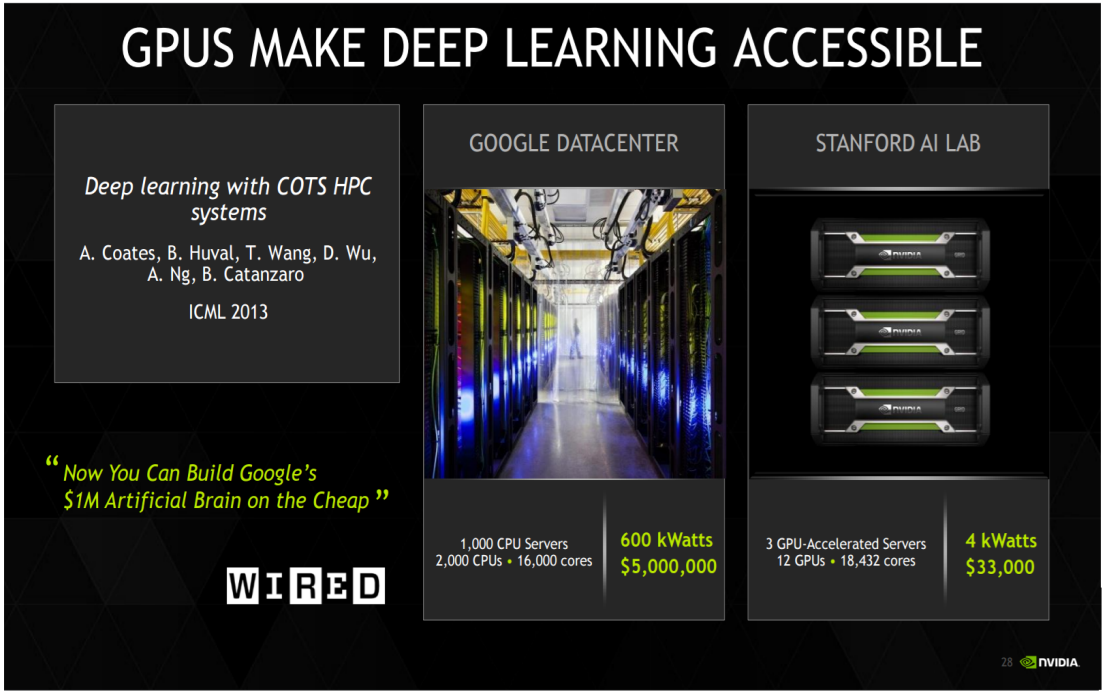
$$f(n) = f(n-1)+f(n-2)$$

These people are harvesting tea:

$$f(n) = n^2+3$$

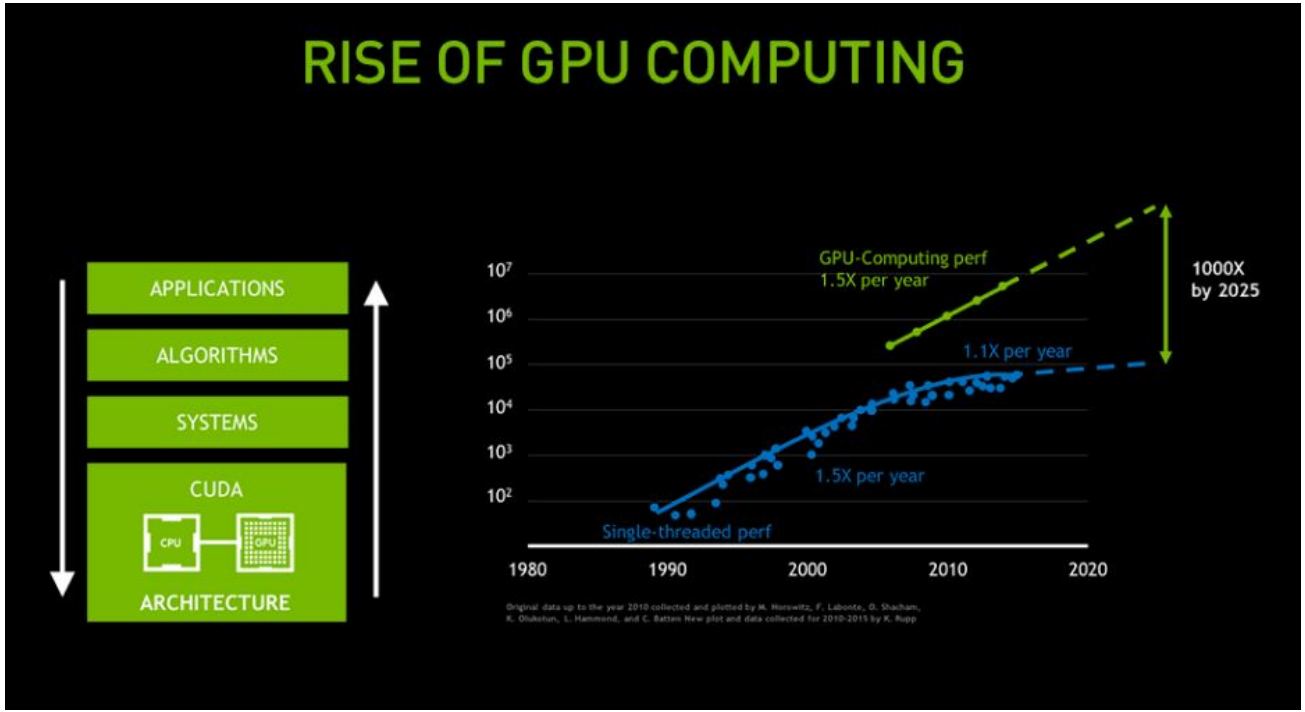
Courtesy

Example of cost benefits in utilizing GPU compute for highly compute intensive applications

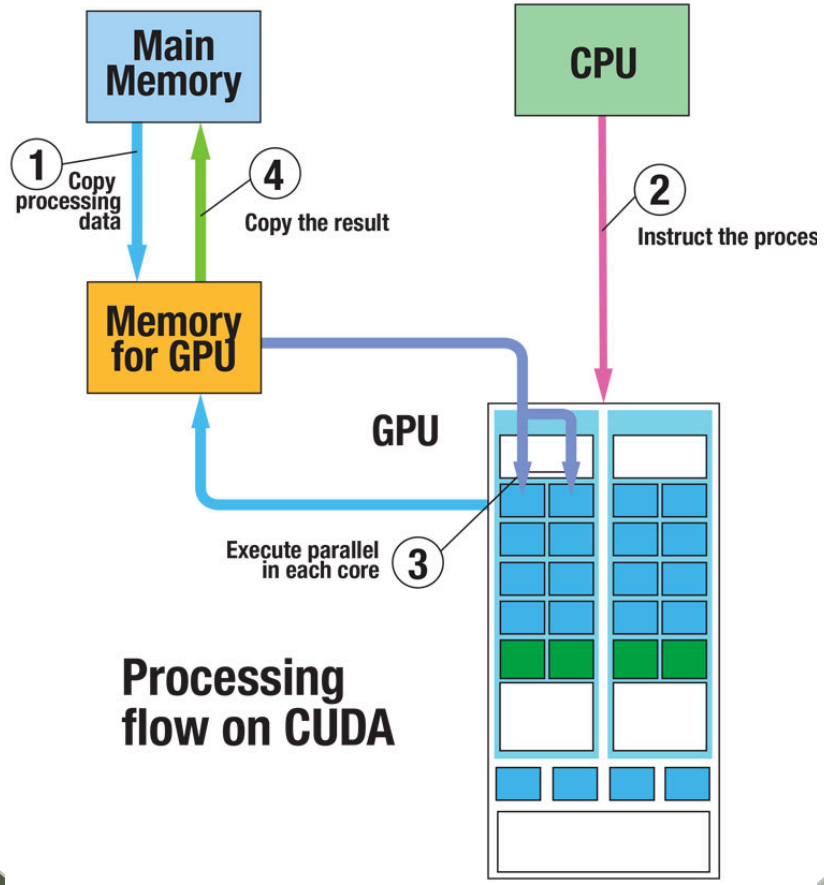


Hardware spend cost savings example enabling application processing for same application.

Why GPU compute is important to scaling analytics



how do you program on a gpu?


Main Memory
CPU
1
Copy processing data
4
Copy the result
2
Instruct the proces
Memory for GPU
GPU
Execute parallel in each core
3
Processing flow on CUDA

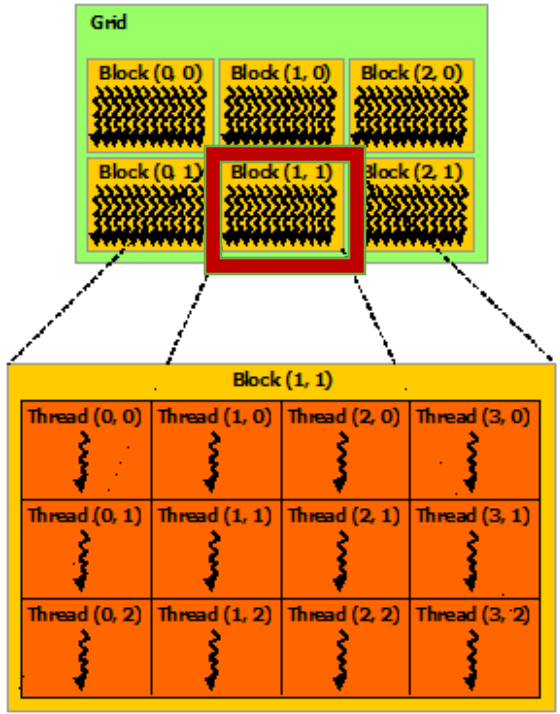
Resource

From a programmer's perspective:

Terminology:

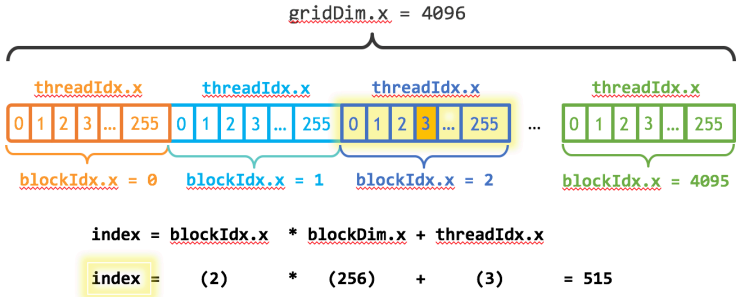
- Kernel

- Thread
- Block
- Grid



CUDA THREADS

my_kernel[4096, 256](arg1, arg2, …)



Thread Positioning:

(1D)

```
i = blockIdx.x * blockDim.x + threadIdx.x;  # width
```

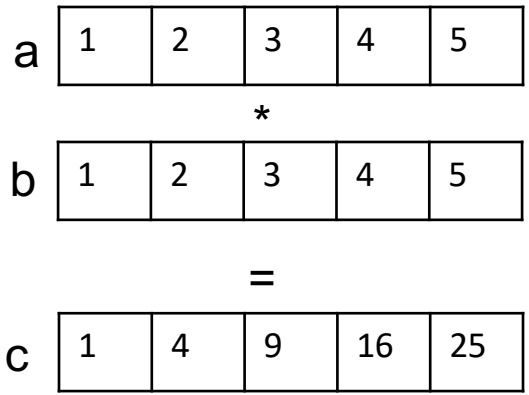
(2D)

```
i = blockIdx.x * blockDim.x + threadIdx.x; # width
j = blockIdx.y * blockDim.y + threadIdx.y; # height
```

(3D)

```
i = blockIdx.x * blockDim.x + threadIdx.x;  # width
j = blockIdx.y * blockDim.y + threadIdx.y;  # height
k = blockIdx.z * blockDim.z + threadIdx.z;  # depth
```

Numba implementation of element-wise arrays multiplication On GPU:



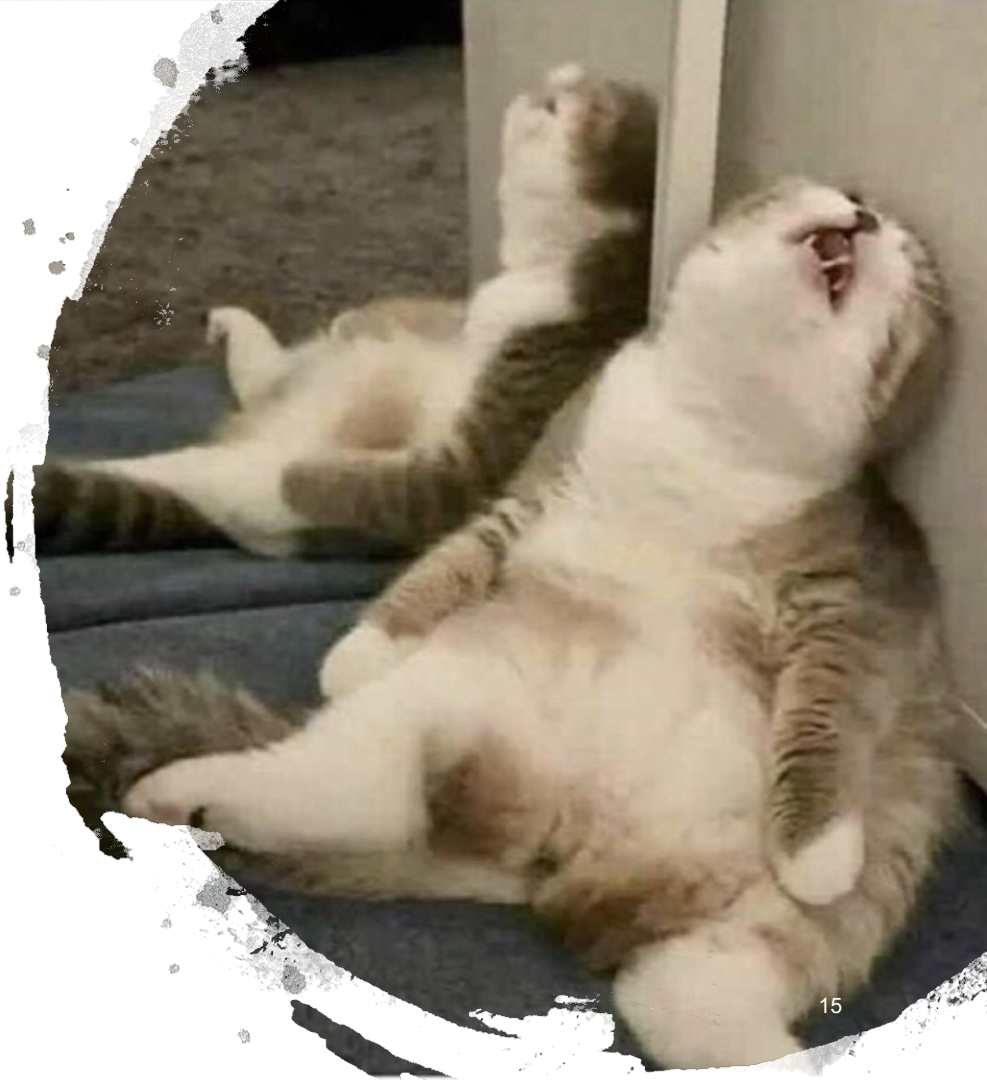
C[i] = a[i]*b[i]

```python
from numba import cuda
import numpy as np

@cuda.jit
def cu_multexamp(a, b, c):
  """This kernel function will be executed by each thread once."""
   i = cuda.threadIdx.x + cuda.blockIdx.x * cuda.blockDim.x
   if i < c.size:
       c[i] = a[i] * b[i]

device = cuda.get_current_device()
n = 100
cpu_a = np.arange(n, dtype=np.float32)
cpu_b = np.arange(n, dtype=np.float32)
cpu_c = np.empty_like(cpu_a)

# Assign equivalent storage on device
gpu_a = cuda.to_device(cpu_a)
gpu_b = cuda.to_device(cpu_b)
# Assign storage on device for output
gpu_c = cuda.device_array_like(cpu_a)

thread_per_block = 32
num_block = int(np.ceil((n)/ thread_per_block ))

cu_multexamp[num_block, thread_per_block ](gpu_a, gpu_b, gpu_c)
cuda.synchronize()
# Transfer output from device to host
cpu_c = gpu_c.copy_to_host()
print (cpu_c)
```

What is next…

Enough of theory…

https://bit.ly/2R4zZ4L



Thank You

github.com/ak-ayush

@kayush206