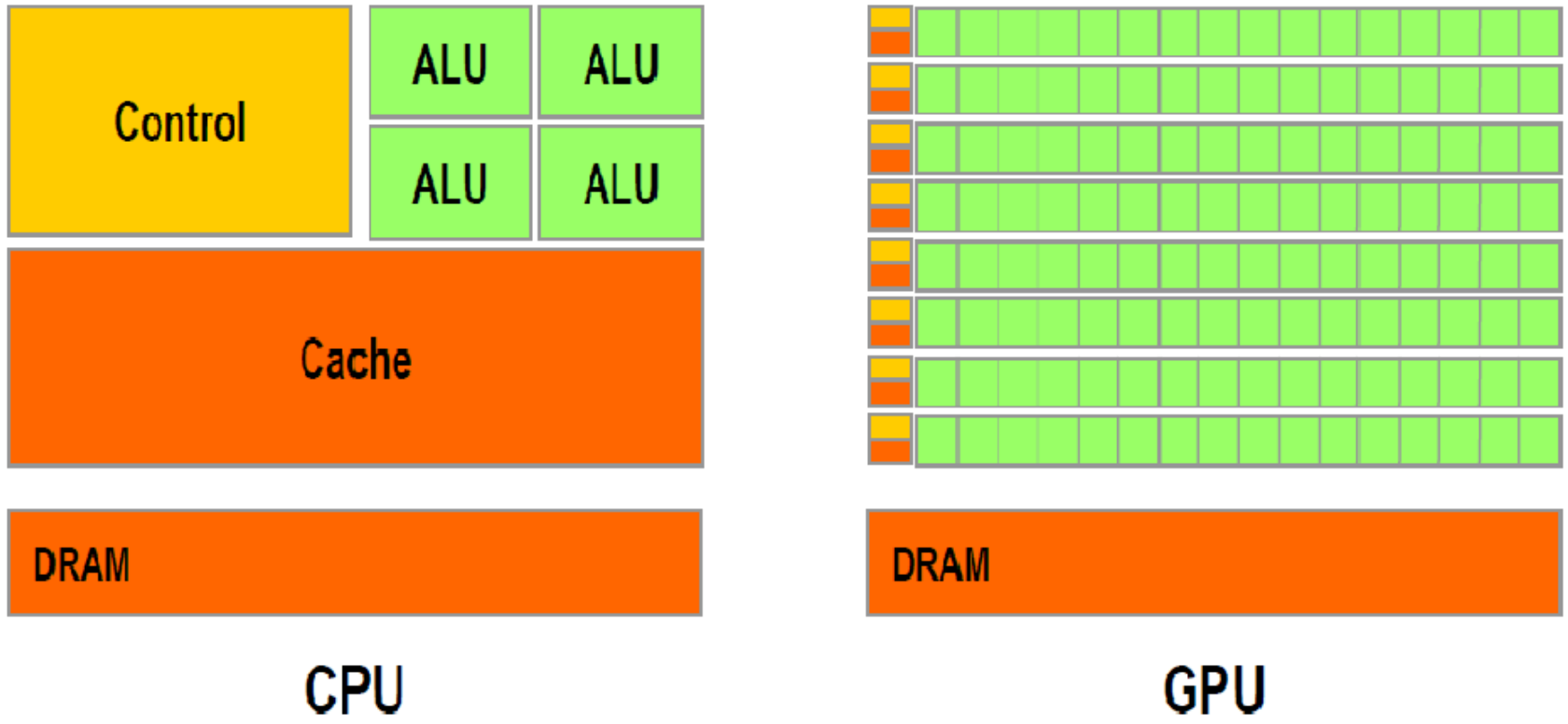
GPU and Multicore CPU Architectures and Computing summary

I. Karakostas, Prof. Ioannis Pitas
Aristotle University of Thessaloniki
pitas@csd.auth.gr
www.aiia.csd.auth.gr
Version 2.7.2

CPU vs GPU

- A multicore CPU consists of **a few** cores optimized for **sequential** serial processing.
- They generally operate in the ***Multiple Instruction Multiple Data*** (***MIMD***) mode.
- A GPU uses **thousands of smaller** cores which are more efficient for a massively **parallel** architecture aimed at handling multiple operations at the same time in the ***Single Instruction Multiple Data*** (***SIMD***) mode.
- Each processing unit on a GPU contains **local memory** that improves data manipulation and **reduces fetch time**.

CPU vs GPU



CPU vs GPU

- Multicore (CPU): MIMD

  Focused on latency.

  Best single thread performance.

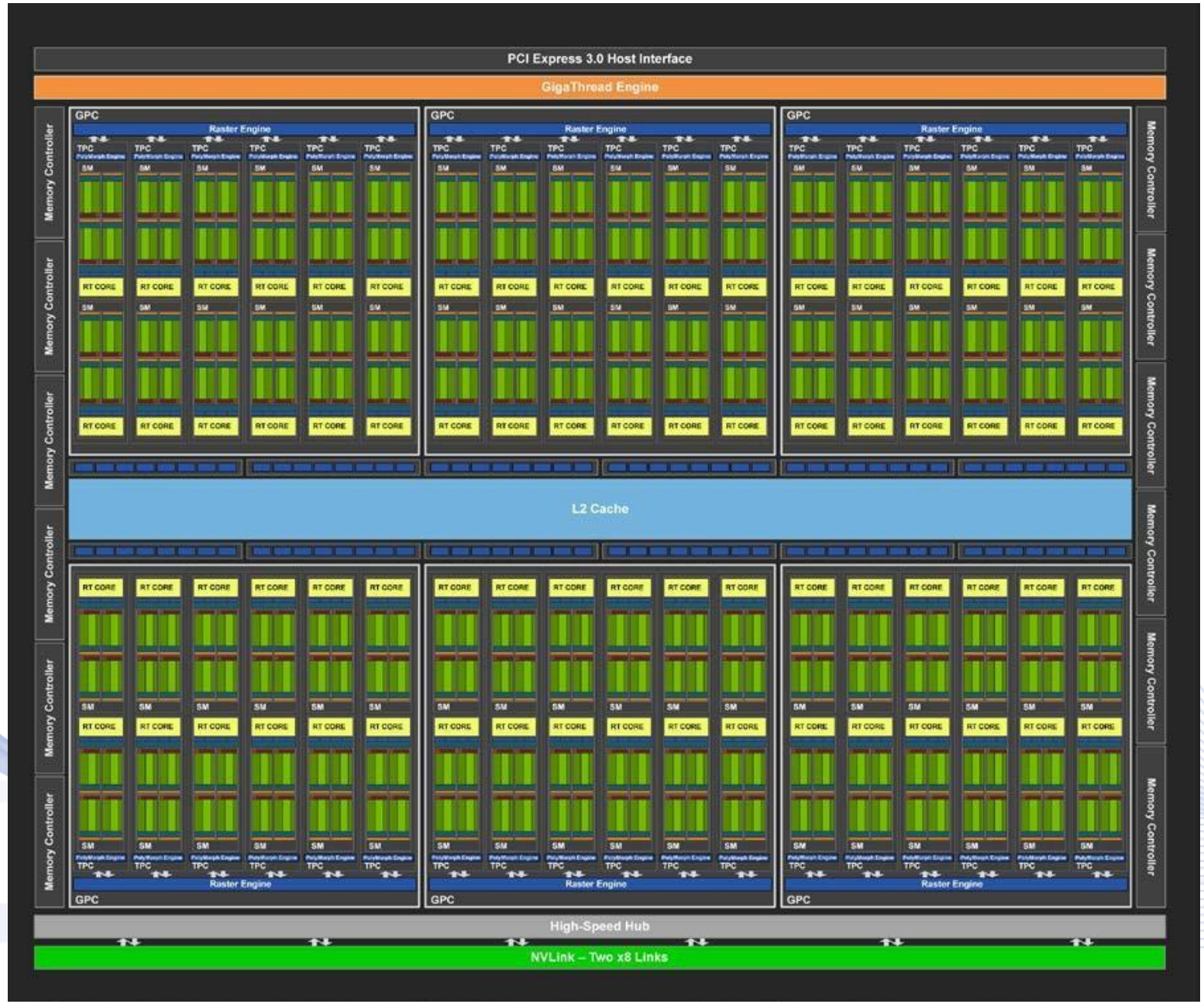
- Manycore (GPU): SIMD

  Focused on throughput.

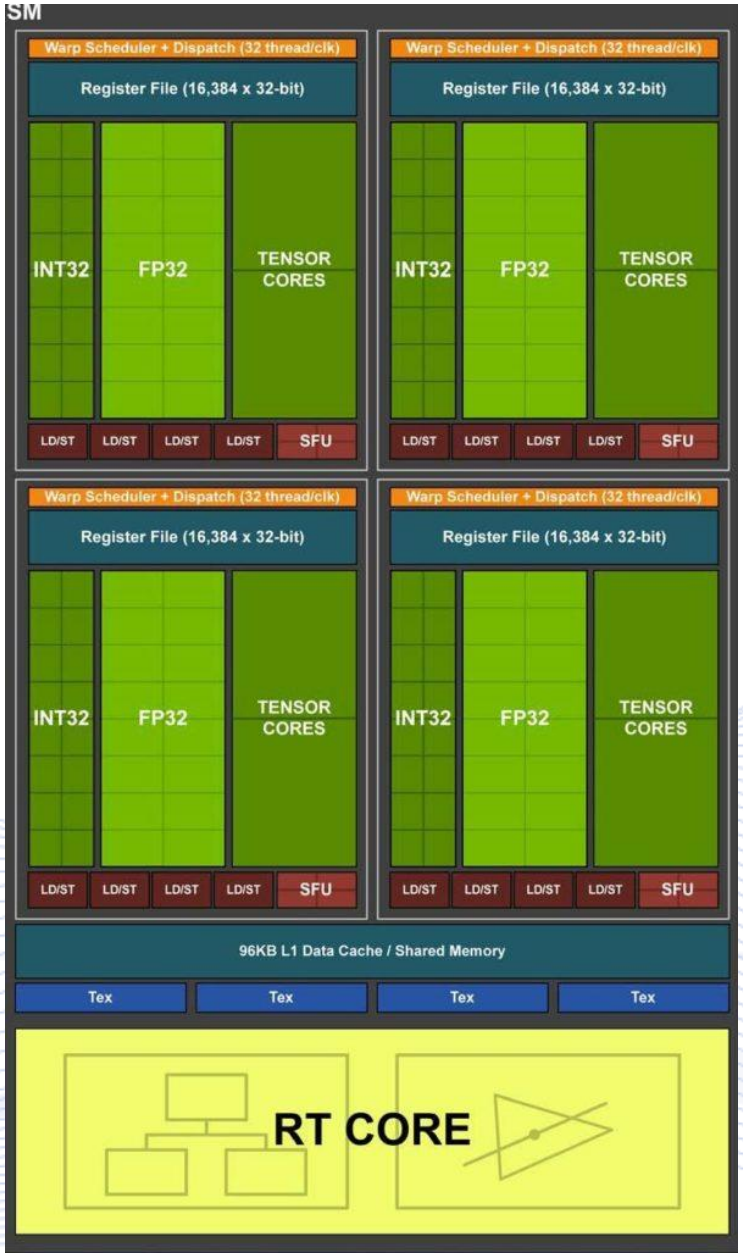
  Best for embarrassingly parallel tasks.

Turing key features

- 4,608 CUDA Cores
- 72 RT Cores
- 576 Tensor Cores
- 288 texture units
- 12 32-bit GDDR6 memory controllers (384-bits total).

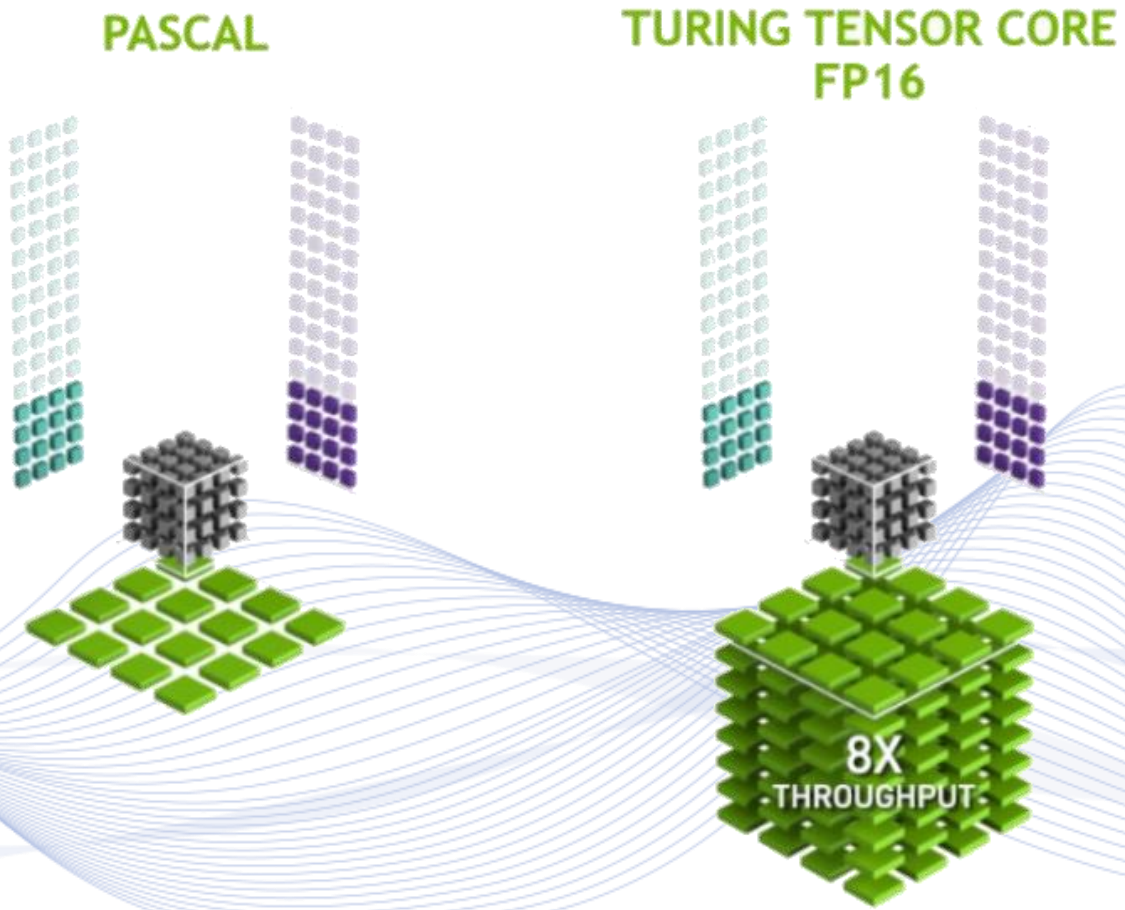


Turing TU102 Full GPU
with 72 SM Units



TURING STREAMING MULTIPROCESSOR (SM) ARCHITECTURE

Tensor cores



NVIDIA Jetson Xavier

- AI Computer for autonomous machines
- Designed for robots, drones and other
- Multiple operating modes (10/15/30 W)
- Comparison to TX2:

  Greater than 10x the energy efficiency.

  More than 20x the performance

Jetson Xavier vs Jetson TX2

	Jetson Xavier	Jetson TX2
GPU	512-core Volta GPU with Tensor Cores	256-core Pascal GPU
CPU	8-core ARMv8.2 64-bit CPU, 8MB L2 + 4MB L3	HMP Dual Denver 2/2 MB L2 + Quad ARM® A57/2 MB L2
Memory	16GB 256-bit LPDDR4x | 137 GB/s	8 GB 128 bit LPDDR4 59.7 GB/s
CSI	Up to 16 simultaneous cameras	Up to 6 Cameras
PCIe	5x 16GT/s Gen 4 (1x8, 1x4, 1x2, 2x1)	Gen 2 | 1x4 + 1x1 OR 2x1 + 1x2
Footprint	100mm x 87mm	50 mm x 87 mm

Parallel algorithm execution

- Graphics computing:

  Highly parallelizable
- Linear algebra parallelization:

  Vector inner products: $c = \mathbf{x}^T\mathbf{y}$.

  Matrix-vector multiplications $\mathbf{y} = \mathbf{A}\mathbf{x}$.

  Matrix multiplications: $\mathbf{C} = \mathbf{A}\mathbf{B}$.

Parallel algorithm execution

- Convolution: $\mathbf{y} = \mathbf{A}\mathbf{x}$

  CNN architectures, linear systems, signal filtering.

- Correlation: $\mathbf{y} = \mathbf{A}\mathbf{x}$

  template matching, tracking.

- Signal transforms (DFT, DCT, Haar, etc):

  Matrix vector product form: $\mathbf{X} = \mathbf{W}\mathbf{x}$

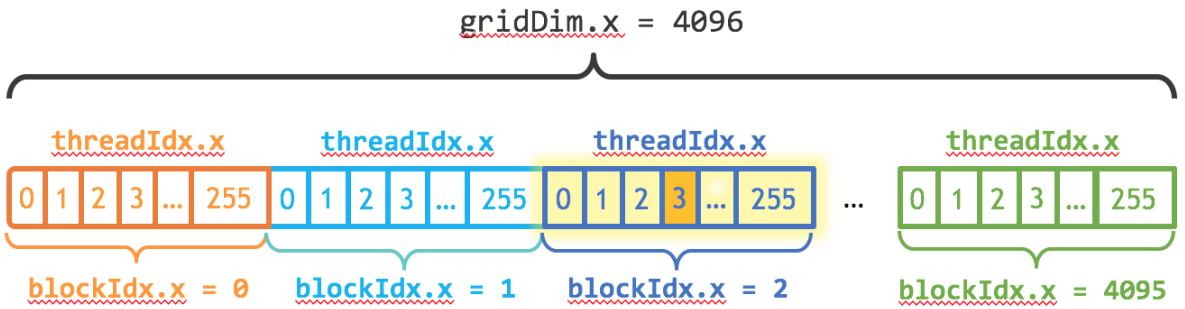
  2D transforms (matrix product form): $\mathbf{X}' = \mathbf{W}\mathbf{X}$.

CUDA Example

```
int main(void)
 {
 int N = 1<<20;
 float *x, *y, *d_x, *d_y;
 x = (float*)malloc(N*sizeof(float));
 y = (float*)malloc(N*sizeof(float));
 cudaMalloc(&d_x, N*sizeof(float));
 cudaMalloc(&d_y, N*sizeof(float));
 for (int i = 0; i < N; i++) {
  x[i] = 1.0f;
  y[i] = 2.0f;
 }
 cudaMemcpy(d_x, x, N*sizeof(float), cudaMemcpyHostToDevice);
 cudaMemcpy(d_y, y, N*sizeof(float), cudaMemcpyHostToDevice);
 saxpy<<<(N+255)/256, 256>>>(N, 2.0f, d_x, d_y);
 cudaMemcpy(y, d_y, N*sizeof(float), cudaMemcpyDeviceToHost);
 cudaFree(d_x);
 cudaFree(d_y);
 free(x);
 free(y);
}
```

CUDA Example

```
__global__
void saxpy(int n, float a, float *x, float *y)
{
  int i = blockIdx.x*blockDim.x + threadIdx
  if (i < n) y[i] = a*x[i] + y[i];
}
```

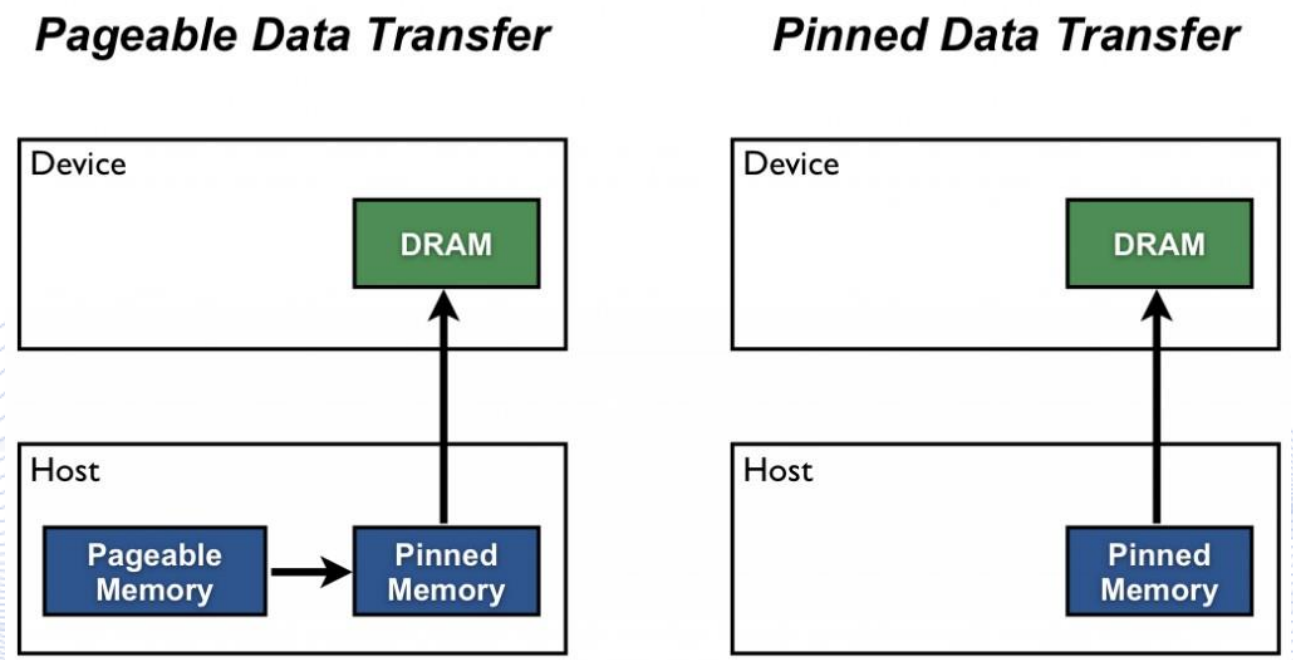



```
index = blockIdx.x  * blockDim.x + threadIdx.x
index =    (2)      *   (256)    +     (3)      = 515
```

https://devblogs.nvidia.com/even-easier-introduction-cuda/

CUDA Optimization

- Use cudaMallocHost() or cudaHostAlloc().
- Use cudaMemcpy2D(dest, dest_pitch, src, src_pitch, w, h, cudaMemcpyHostToDevice).



https://devblogs.nvidia.com/how-optimize-data-transfers-cuda-cc/

CPU Optimization

- Spawn threads.
- Use registers.
- Loop unrolling.
- Use SIMD capabilities.
- Take data locality into consideration.
- Trust the compiler.

Q & A

Thank you very much for your attention!

More material in
http://icarus.csd.auth.gr/cvml-web-lecture-series/

Contact: Prof. I. Pitas
pitas@csd.auth.gr