**Parallel programming CUDA C/C++**

* GPGPU – General Purpose Programmability on the Graphics Processing Unit
* Threads
* Constant clock speed, smaller resources – technology trends
* Trend: more, simpler processors, GPUs
* Latency – time (s, sec) = CPU
* Throughput – stuff/time (jobs / hours) = GPU
* Care more about the metric of pixels/seconds than the time for one individual pixel

|  |  |  |
| --- | --- | --- |
| Car |  |  |
| Latency | 22.5 | Hours |
| Throughput | 0.089 | People/hour |
| Bus |  |  |
| Latency | 90 | Hours |
| Throughput | 0.45 | People/hour |

Core GPU design tenets:

1. Lots of simple compute units, trade simple control for more compute
2. Explicitly parallel prog. Model
3. Opt. for throughput not latency

CPU – “HOST”

GPU – “DEVICE”

If you run a plain C program, your code will only allow you to use the CPU to run your program. So how do we wrote code that will run on the GPU? – and here comes CUDA.

Supports multiple languages, C, C++, and Fortran.

**How CUDA works?**

Part of CUDA program is plain C and will run on your CPU. CUDA calls this the host. The other part of your problem will run on the GPU, in parallel. It’s also written in C but with some extensions that we use to express parallelism. The CUDA term for your GPU is the device. Then the CUDA compiler will compile your program, split it into pieces, that will run on the CPU and the GPU, and generate code for each. CUDA assumes that the device, the GPU, is a co-processor to the host, the CPU. It also assumes, that both the host and the device have their own separate memories, where they store data. In the systems we use nowadays, both the CPU and the GPU have their own physical dedicated memory in the form of DRAM, with the GPU’s memory typically being a very high performance block of memory. CPU – in charge, runs the main program. Sends directions to the GPU to tell what to do. GPU responsible for:

Moving data from CPU mem to GPU mem and reverse

In C moviung data is called memcopy

Allocating memory on the gpu – cudaMalloc ☺

Invoking programs on the GPU to compute things in parallel – kernels

HOST LAUNCHES KERNELS ON THE DEVICE!

So, the GPU can do the following:

* Respond to CPU request to send data GPU -> CPU
* Respond to CPU request recv data CPU->GPU \*but cannot initiate request)
* Compute a kernel launched by the CPU

**How a typical program looks like?**

1. CPU allocates storage on GPU – cudaMalloc
2. CPU copies input data from CPU->GPU cudaMemcpy
3. CPU launches kernel(s) on GPU to process the data – Kernel launch – specifies the degree pf parallelism
4. CPU copies results back to CPU from GPU – codaMemcpy

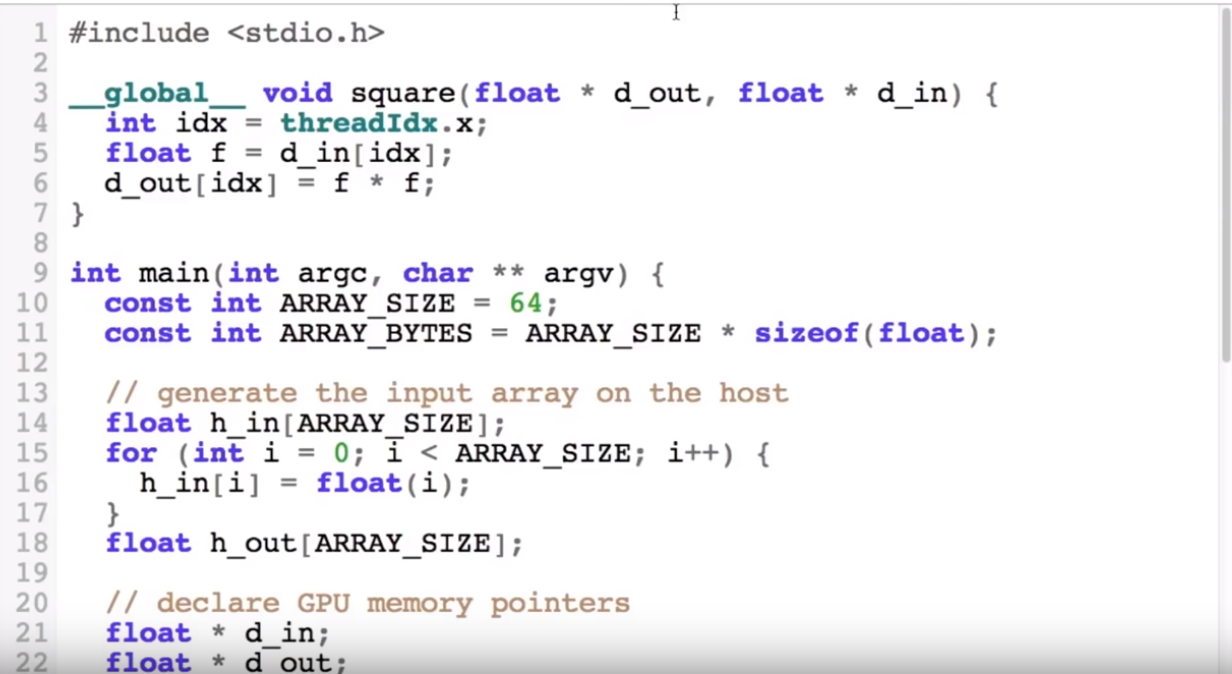
\*minimalize data transfer

**CUDA big idea**

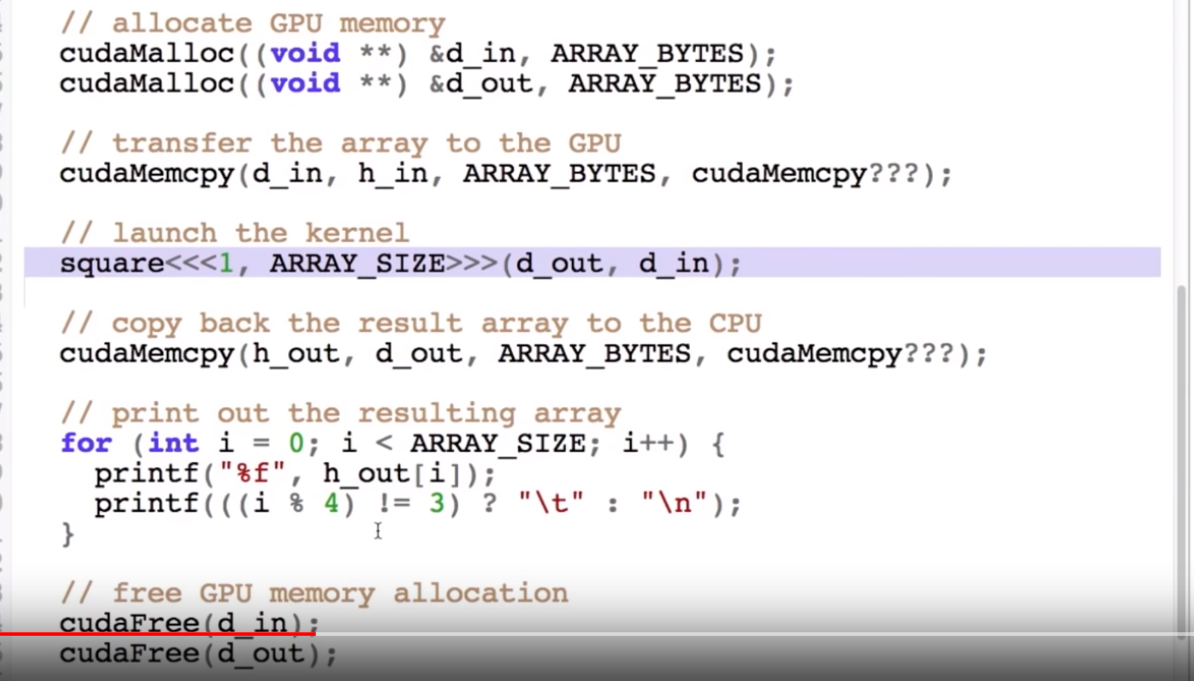
Kernels look like serial programs, write your program as if it will run on ONE thread the GPU will run that program on MANY threads.

CPU code: *square Kernel <<< 64 >>> (outArray, inArray)*

Be like – N’th kernel please work on n’th problem



* CUDA naming convention – data on the CPU starts with “h\_” -> host and data on the GPU starts with “d\_” -> device (helps to avoid the most primitive error: to try improperly access the variables.
* \_\_global\_\_ void square : the kernel itself – the way that cude knows this is a kernel.
* Build in variable, thread index : *threaIdx.*
* Looks like a serial program ☺ but runs in parallel 64 times.



CudaMalloc means – allocate the data on GPU while plain malloc – to allocate data on CPU

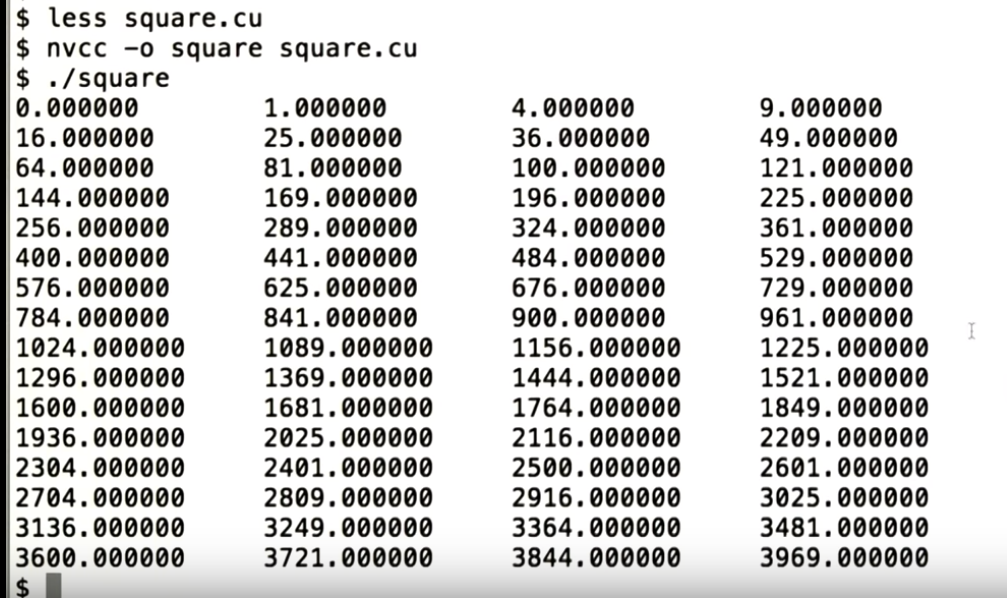
cudaMemcpy – the first 3 args are the same as the regular C memCpy (dest, source, bytes). The 4th arg says the ddirection of the transfer. 3 choices ->

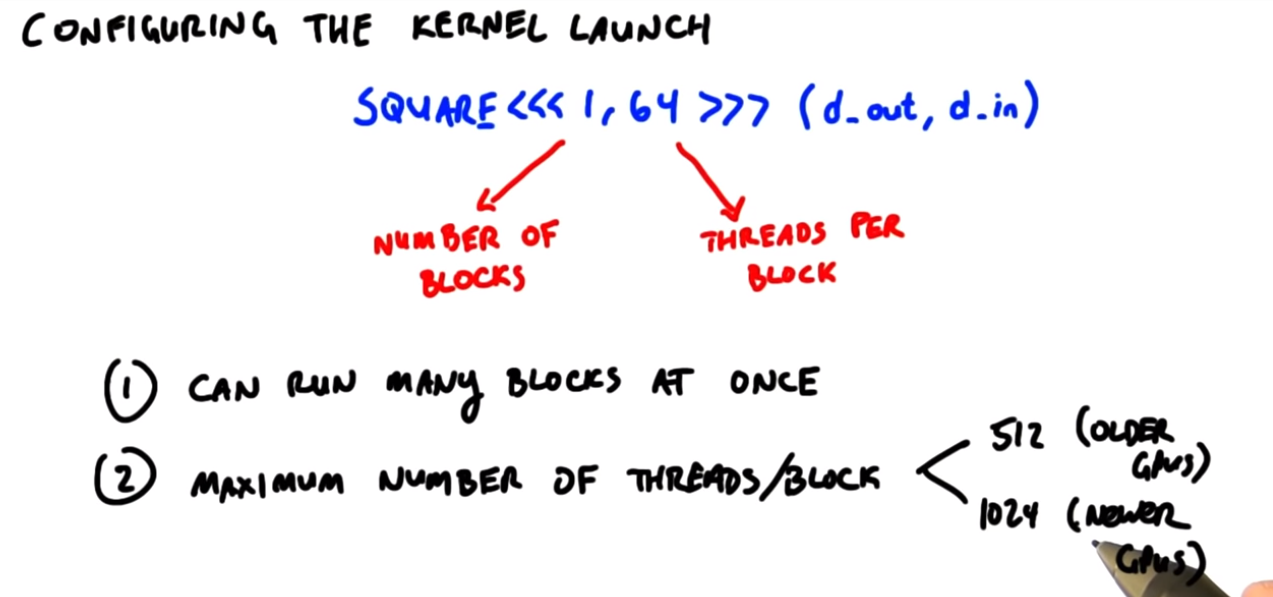
1. cudaMemcpyHostToDevice
2. cudaMemoryDeviceToHost
3. cudaMemoryDeviceToDevice

name<<(number of blocks), ARRAY\_SIZE (element count)< >>>(d\_out, d\_in) - use only d\_ data ☺

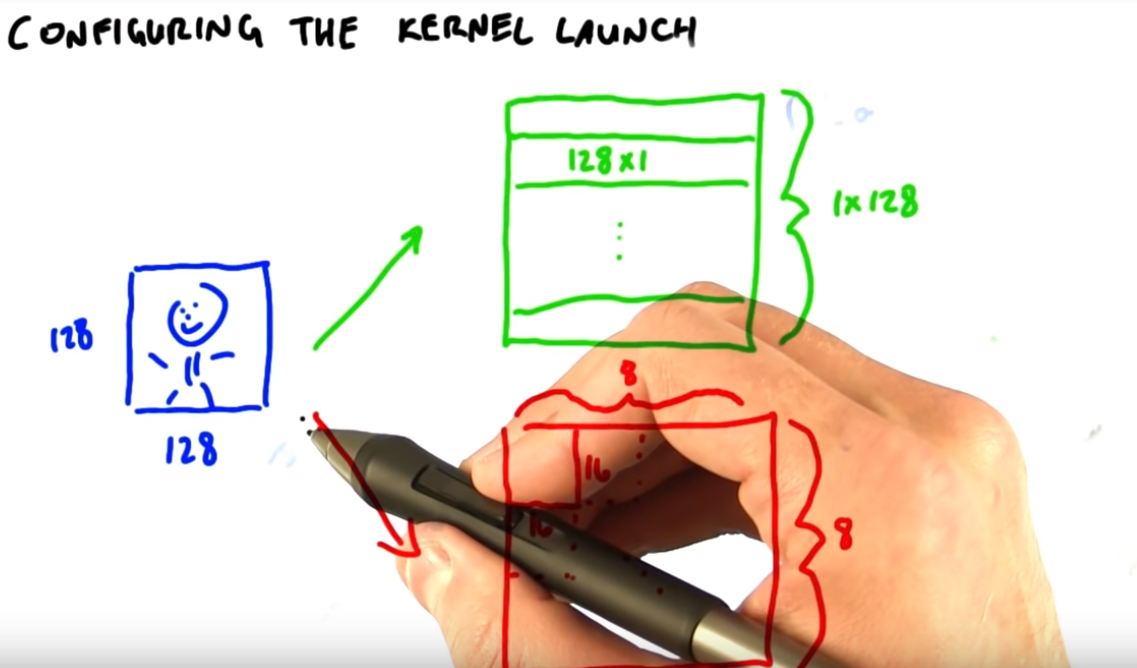
CUDA launch operator

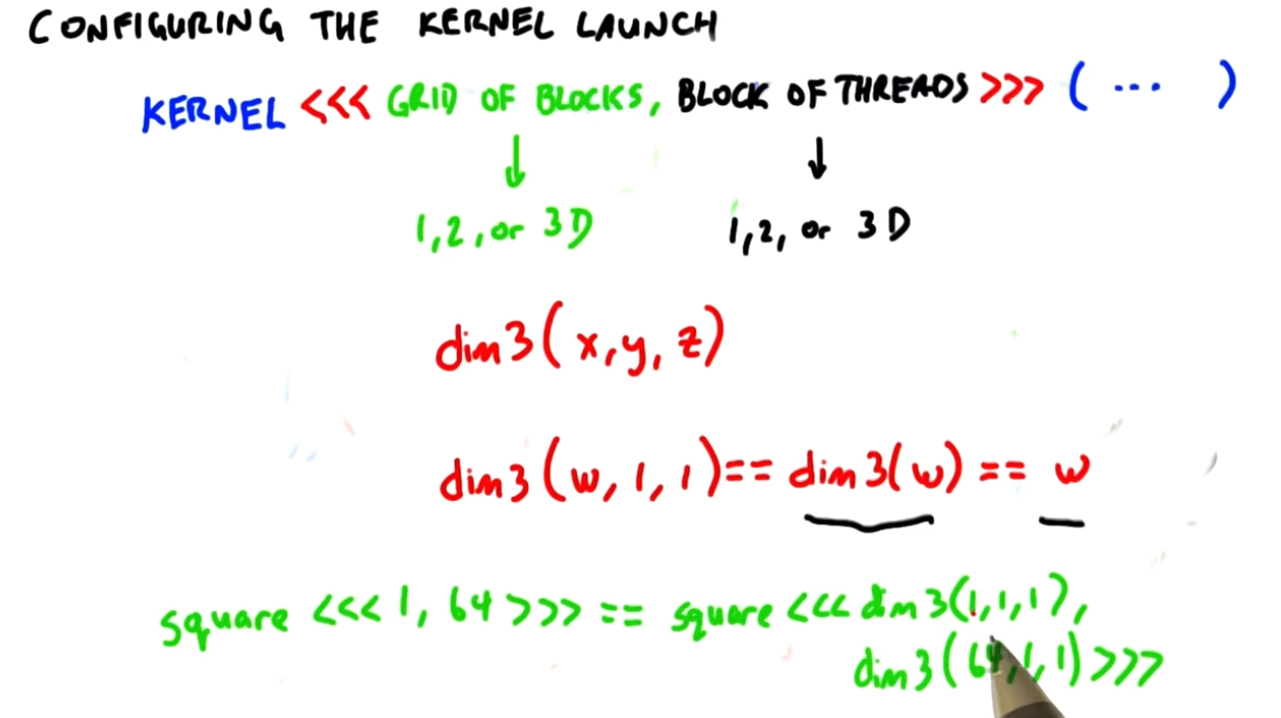




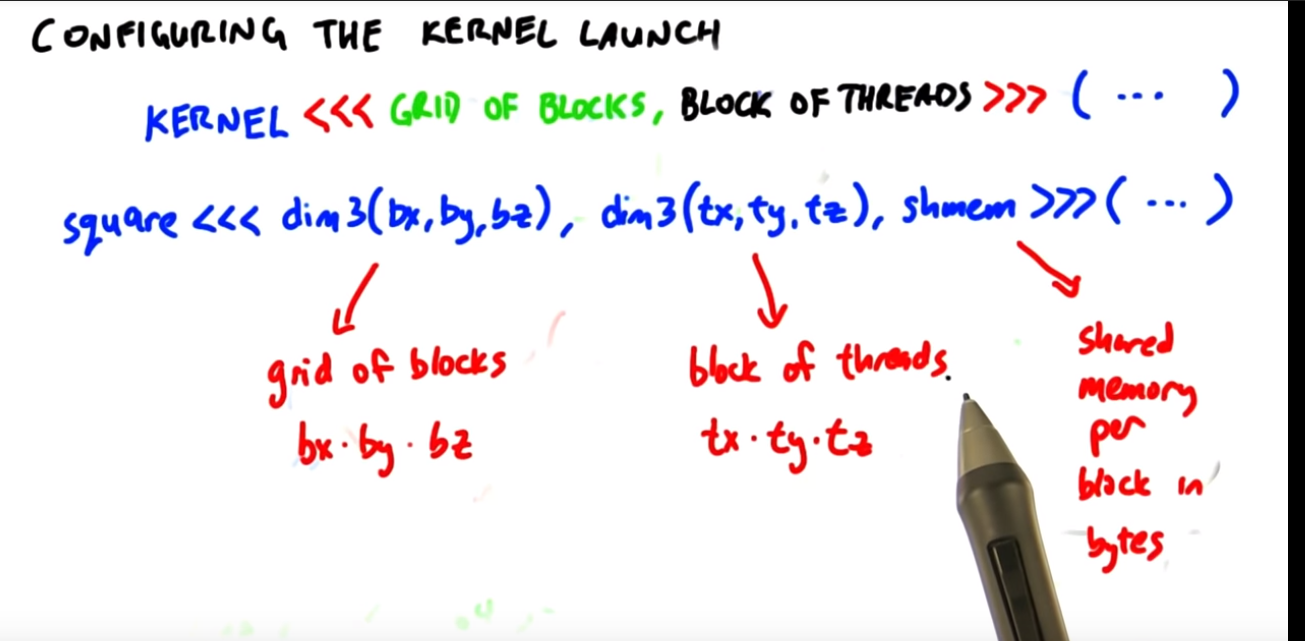


* **1/2/3 dimensional problems – 2d: image processing**

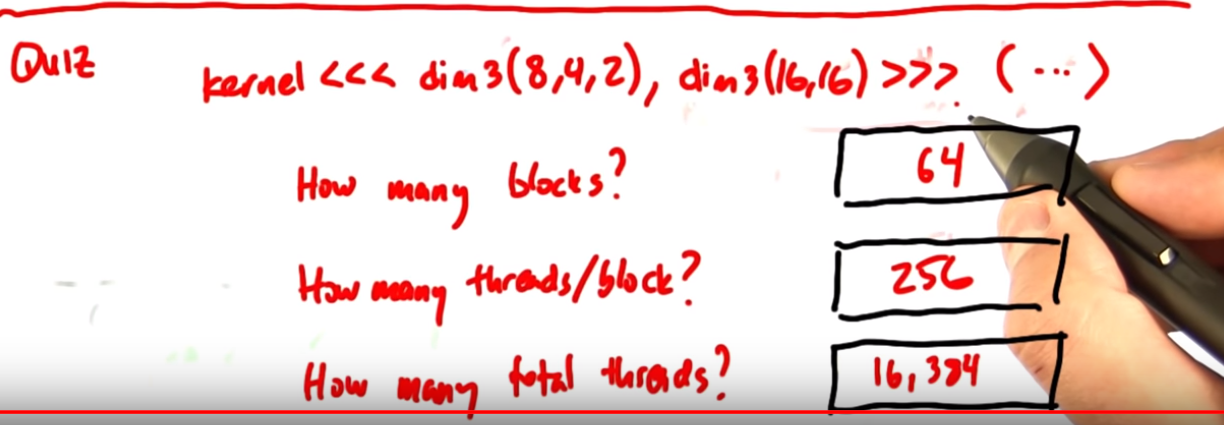




* **Kernel launching – in general**



* **Calculation of Dimensions/Threads/Blocks**



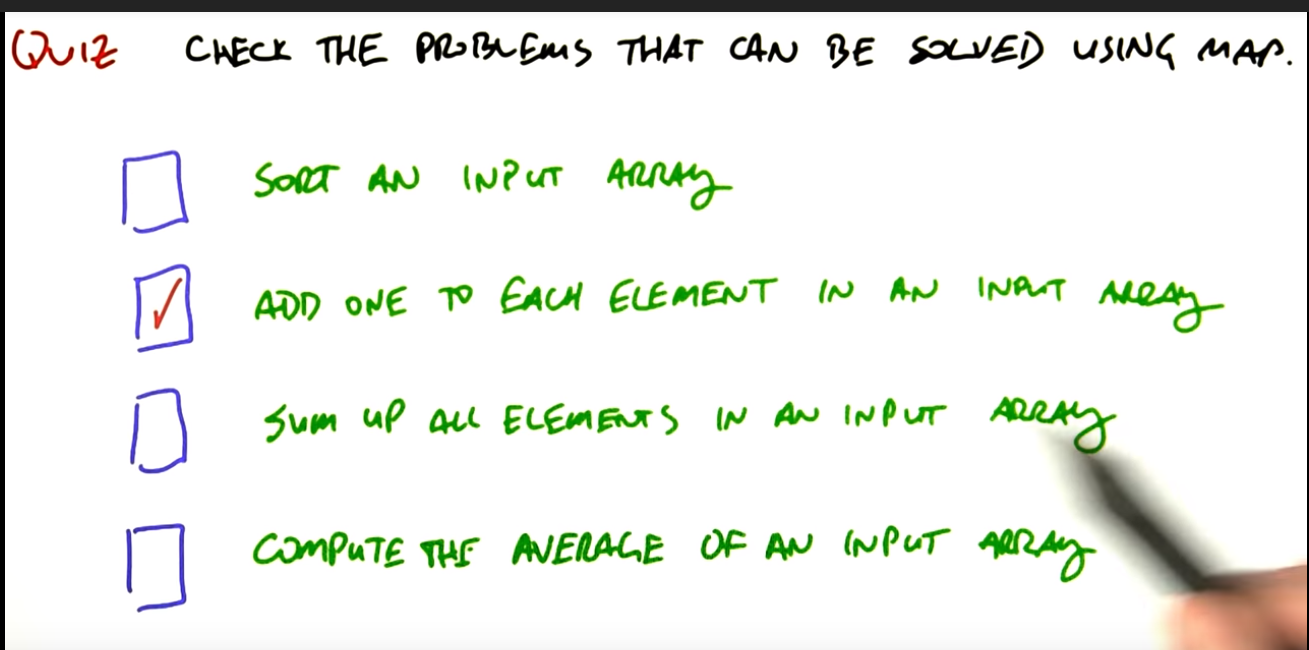
**MAP – powerful parallel technique**

-set of elements to process (64 floats)

-function to run on each element (“square”)

Map (elements, function)





We cannot sort an input array using map because the output is dependent on all the input data.

CUDA – The Code

Worries:

* No NVDIA graphic card at home
* Yet unknown parameters of PCs in school

Suggestion:

* Use map (easiest) and try an another collection in different example.

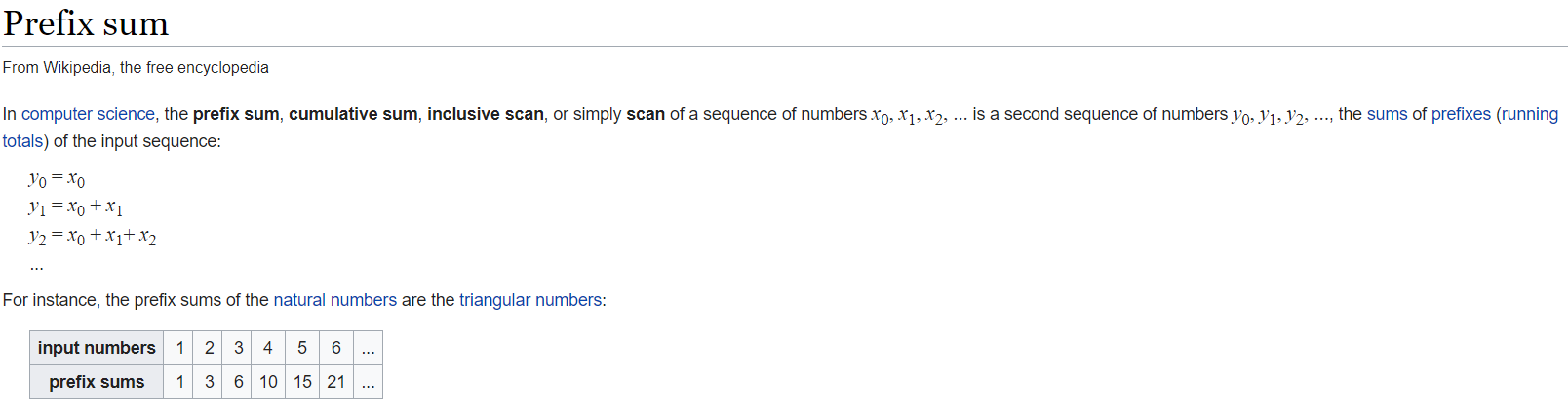
1. **Matrix-Matrix Multiplication on the GPU with Nvidia CUDA**

[**https://www.quantstart.com/articles/Matrix-Matrix-Multiplication-on-the-GPU-with-Nvidia-CUDA?fbclid=IwAR2O5bWt1nEEd2ZxV7Df8bO5AYMvvd91HS4u1xvtn1H6qDlGg5qc7Ww2D9o**](https://www.quantstart.com/articles/Matrix-Matrix-Multiplication-on-the-GPU-with-Nvidia-CUDA?fbclid=IwAR2O5bWt1nEEd2ZxV7Df8bO5AYMvvd91HS4u1xvtn1H6qDlGg5qc7Ww2D9o)

1. **Parallel Prefix Sum (Scan) with CUDA**

[**https://www.mimuw.edu.pl/~ps209291/kgkp/slides/scan.pdf?fbclid=IwAR0lebrwdFEvd\_bFXGLhVLvk0bwZkGC4Zl4yBrssTT1Jy0mWvJSCqu2eHI4**](https://www.mimuw.edu.pl/~ps209291/kgkp/slides/scan.pdf?fbclid=IwAR0lebrwdFEvd_bFXGLhVLvk0bwZkGC4Zl4yBrssTT1Jy0mWvJSCqu2eHI4)

what is prefix sum?



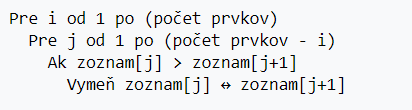
1. **An Efficient Matrix Transpose in CUDA C/C++**

[**https://devblogs.nvidia.com/efficient-matrix-transpose-cuda-cc/?fbclid=IwAR2QtT17bKl513Ifs6fuC5g2ydbSQa00V6X8lM3ODqJsOI8TuWmmMwHTtfw**](https://devblogs.nvidia.com/efficient-matrix-transpose-cuda-cc/?fbclid=IwAR2QtT17bKl513Ifs6fuC5g2ydbSQa00V6X8lM3ODqJsOI8TuWmmMwHTtfw)

1. **A program to count prime numbers in a given interval of integers**

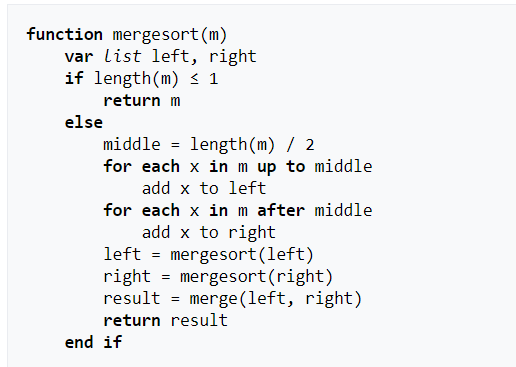
[**https://github.com/akhan3/find-primes-cuda?fbclid=IwAR24H933nFasCYKTSd9vYnSRVzutfv-ziCLoxu00AvkwFNfhQMmwRhcvIbg**](https://github.com/akhan3/find-primes-cuda?fbclid=IwAR24H933nFasCYKTSd9vYnSRVzutfv-ziCLoxu00AvkwFNfhQMmwRhcvIbg)

1. **Bubble Sort**



[**https://github.com/3ZadeSSG/GPU-Programming-CUDA-C/tree/master/Bubble%20Sort**](https://github.com/3ZadeSSG/GPU-Programming-CUDA-C/tree/master/Bubble%20Sort)

1. **Merge Sort**



[**https://github.com/3ZadeSSG/GPU-Programming-CUDA-C/tree/master/Program%206-Merge%20Sort**](https://github.com/3ZadeSSG/GPU-Programming-CUDA-C/tree/master/Program%206-Merge%20Sort)

1. **Implementation-of-the-Cholesky-Decomposition-in-GPU-using-CUDA**

**Totalne sa mi nechce ucit sa ten decomposition.**

**FYI:** [**https://www.youtube.com/watch?v=NppyUqgQqd0**](https://www.youtube.com/watch?v=NppyUqgQqd0)

[**https://github.com/bhattmansi/Implementation-of-Cholesky-Decomposition-in-GPU-using-CUDA**](https://github.com/bhattmansi/Implementation-of-Cholesky-Decomposition-in-GPU-using-CUDA)

1. **Determinant Computation on the GPU using the Condensation Method**

**Neviem co to je za metoda, ale urcite bude husta**

[**http://www.csd.uwo.ca/~moreno/Publications/DetHpcsPaper-proceedings.pdf**](http://www.csd.uwo.ca/~moreno/Publications/DetHpcsPaper-proceedings.pdf)

1. **Parallel calculation of the median and order statistics on GPUs with application to robust regression**

**Neviem presne na ake statisticke metody sa da efektivne pouzivat paralelizmus, ale na nieco sa da urcite ☺**

[**https://arxiv.org/pdf/1104.2732.pdf**](https://arxiv.org/pdf/1104.2732.pdf)

1. **Some kind of image processing (2D/3D/struct) – e.g. transform a colored image int a grayscale image**

[**https://www.youtube.com/watch?v=AJtaK-pVAV0&list=PLGvfHSgImk4aweyWlhBXNF6XISY3um82\_&index=47**](https://www.youtube.com/watch?v=AJtaK-pVAV0&list=PLGvfHSgImk4aweyWlhBXNF6XISY3um82_&index=47)

[**https://github.com/tomepel/CUDA-CS344/blob/master/Problem%20Sets/Problem%20Set%201/student\_func.cu**](https://github.com/tomepel/CUDA-CS344/blob/master/Problem%20Sets/Problem%20Set%201/student_func.cu)

**prezentacia intro, basic veci ,**

**matica nasobenie – porovnanie casov, pripadne graf**

**2. 3. – live coding**

**Pre RGB prevod – transformacie. Dolezite je paralelny cyklus. Nejaku kostru mozeme mat a podstatu dopisat.**

**Share**

**C pre maticova nasobenie, sekvencne riesenie**

**Porovnanie rychlosti – pre c a parallel.**

**Report – ku koncu niekedy, tam dopiseme take sracky o rychlosti, casi. Viac na Moodle.**

**Co odovzdat : 3 riesenia, prez., pisomka (5-6 riadkov zmazat a napisat //TODO)**

**Ulohy zdielajme.**