



DEEP
LEARNING
INSTITUTE

Fundamentals of Accelerated Computing with CUDA C/C++

Dr. Momme Allalen | CSC | 09.09.2020



VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

IT4INNOVATIONS
NATIONAL SUPERCOMPUTING
CENTER

Overview



- The workshop is co-organized by LRZ, CSC, IT4Innovations and NVIDIA Deep Learning Institute (DLI) for the Partnership for Advanced Computing in Europe (PRACE).
- NVIDIA Deep Learning Institute (DLI) offers hands-on training for developers, data scientists, and researchers looking to solve challenging problems with deep learning.
- This 4-days workshop offered for the first time online combines lectures about fundamentals of Deep Learning for Multiple Data Types and Multi-GPUs with lectures about Accelerated Computing with OpenACC and CUDA C/C++
- Learn how to train and deploy a neural network to solve real-world problems, how to generate effective descriptions of content within images and video clips, how to effectively parallelize training of deep neural networks on Multi-GPUs and how to accelerate your applications with OpenACC and CUDA C/C++.
- The lectures are interleaved with many hands-on sessions using Jupyter Notebooks. The exercises will be done on a fully configured GPU-accelerated workstation in the cloud.



PRACE Training



LRZ as part of the Gauss Centre for Supercomputing (GCS) and IT4Innovations belong to the 14 **PRACE Training Centres** that started in 2012-2017-2020:

- Barcelona Supercomputing Center (Spain)
- CINECA Consorzio Interuniversitario (Italy)
- CSC – IT Center for Science Ltd (Finland)
- EPCC at the University of Edinburgh (UK)
- Gauss Centre for Supercomputing (Germany)
- Maison de la Simulation (France)
- GRNET – Greek Research and Technology Network (Greece)
- ICHEC – Irish Centre for High-End Computing (Ireland)
- IT4I – National Supercomputing Center VSB Technical University of Ostrava (Czech Republic)
- SURFsara (The Netherlands)
- TU Wien – VSC Research Center (Austria)
- University ANTWERPEN – VSC & CÉCI (Belgium)
- University of Ljubljana – HPC Center Slovenia (Slovenia)
- Swedish National Infrastructure for Computing (SNIC) (Sweden)



Univerza v Ljubljani



Mission: Serve as **European hubs and key drivers of advanced high-quality training** for researchers working in the computational sciences.

<http://www.training.prace-ri.eu/>





DEEP
LEARNING
INSTITUTE



DEEP LEARNING INSTITUTE

DLI Mission: Help the world to solve the most challenging problems using AI and deep learning

We help developers, data scientists and engineers to get started in architecting, optimizing, and deploying neural networks to solve real-world problems in diverse industries such as autonomous vehicles, healthcare, robotics, media & entertainment and game development.



VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

IT4INNOVATIONS
NATIONAL SUPERCOMPUTING
CENTER

Fundamentals of Accelerated Computing with CUDA C/C++



- You learn the basics of **CUDA C/C++** by:
 - Accelerating CPU-only applications to run their latent parallelism on GPUs.
 - Utilizing essential **CUDA memory** management techniques to optimize accelerated applications.
 - Exposing accelerated application potential for concurrency and exploiting it with **CUDA streams**.
 - Leveraging command line and visual profiling to guide and check your work.
 - Upon completion, you'll be able to accelerate and optimize existing C/C++ CPU-only applications using the most essential **CUDA tools** and techniques. You'll understand an iterative style of **CUDA** development that will allow you to ship accelerated applications fast.



Tentative Agenda



- 10:00-10:15 Introduction **CUDA C/C++**
- 10:15-12:00 Accelerating Applications with **CUDA C/C++**
- 12:00-13:00 Lunch Break**
- 13:00-14:20 Managing Accelerated Application Memory with **CUDA** Unified Memory and **nsys**
- 14:20-14:30 Coffee Break**
- 14:30-15:45 Asynchronous Streaming and Visual Profiling for Accelerated Applications with **CUDA C/C++**
- 15:45-16:00 Q&A, Final Remarks



Workshop Webpage



- **Lecture material will be made available under:**
 - <https://tinyurl.com/dl-gpu-workshop-csc>

- **Access CUDA C/C++ Code:**
 - See the: Chat Window



Training Setup



DEEP
LEARNING
INSTITUTE



- To get started, follow these steps:
- Create an NVIDIA Developer account at <http://courses.nvidia.com/join> Select "Log in with my NVIDIA Account" and then "Create Account".
- If you use your own laptop, make sure that WebSockets works for you:
Test your Laptop at <http://websocketstest.com>
 - Under ENVIRONMENT, confirm that "WebSockets" is checked yes.
 - Under WEBSOCKETS (PORT 80]. confirm that "Data Receive", "Send", and "Echo Test" are checked yes.
 - If there are issues with WebSockets, try updating your browser.
We recommend Chrome, Firefox, or Safari for an optimal performance.
- Visit <http://courses.nvidia.com/dli-event> and enter the event code provided by the instructor.
- You're ready to get started.



VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

IT4INNOVATIONS
NATIONAL SUPERCOMPUTING
CENTER

And now ...



Enjoy the course!



Why do we need to program for GPU?

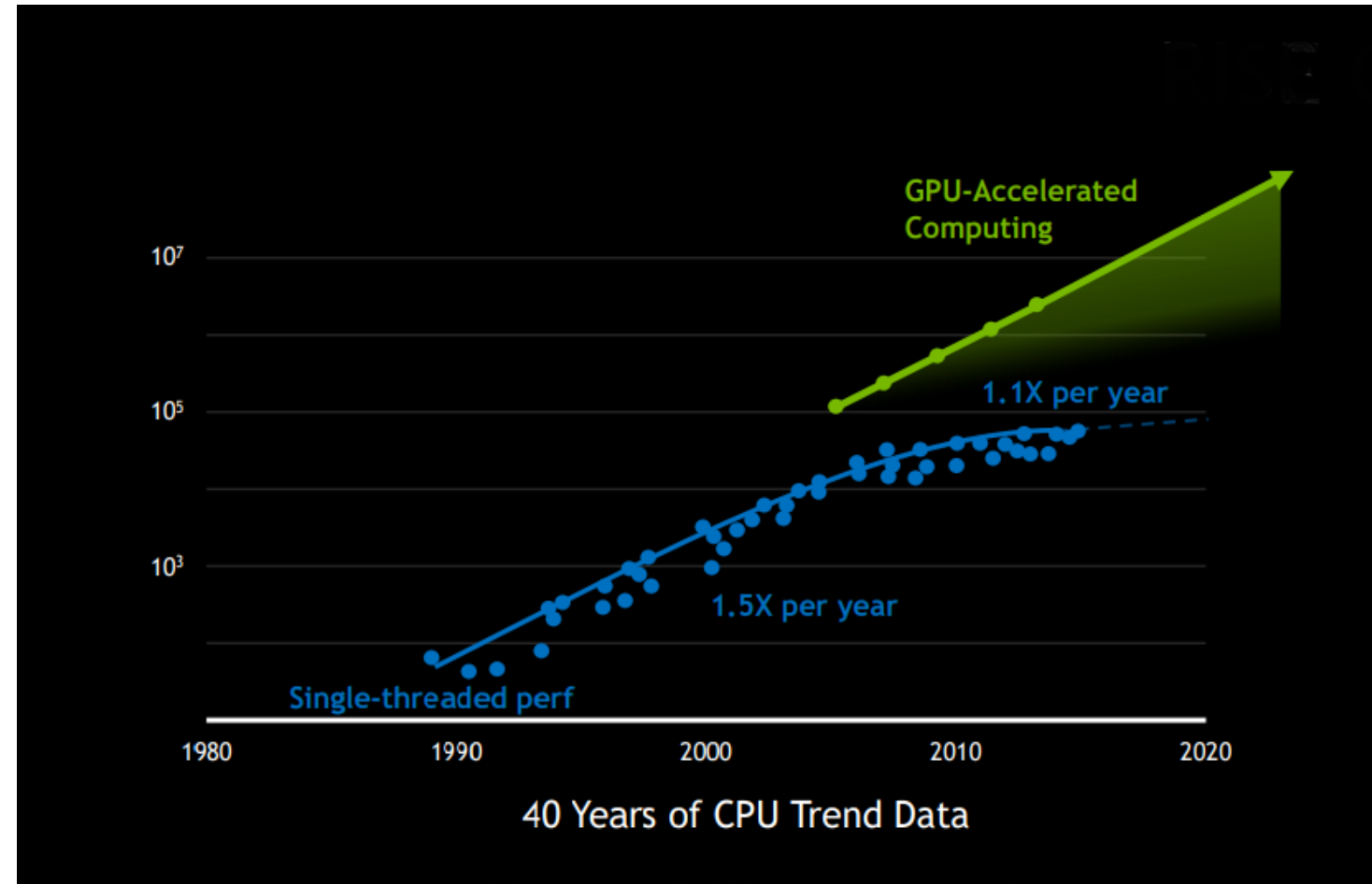


Moore's law is dead !!

The long-held notion that the processing power of computers increases exponentially every couple of years has hit its limit

The free lunch is over ..

Future is parallel !



Why do we need to program for GPU?



DEEP
LEARNING
INSTITUTE



Typical example Intel chip: **Core i7 7th Gen**

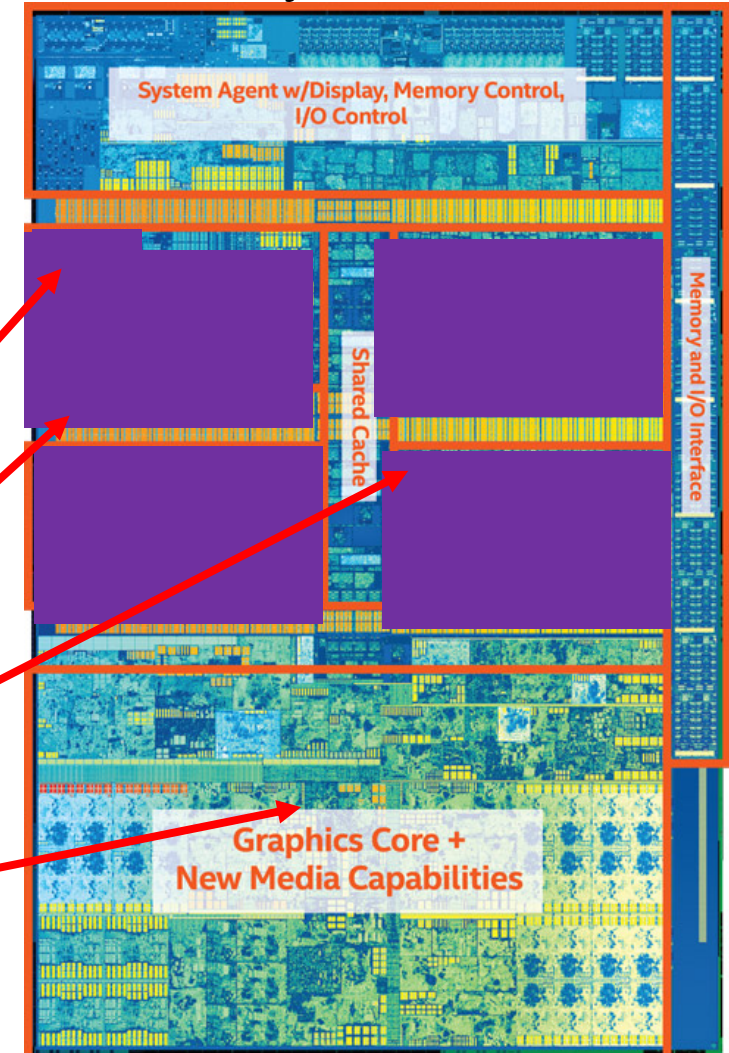
- 4*CPU cores
- with hyperthreading
- Each with 8-wide AVX instructions
- GPU with 1280 processing elements

Programming on chip:

- Serial C/C++ .. Code alone only takes advantage of a very small amount of the available resources of the chip
- Using vectorisation allows you to fully utilise the resources of a single hyper-thread
- Using multi-threading allows you to fully utilise all CPU cores

GPU need to be used?

Intel Kaby Lake-S



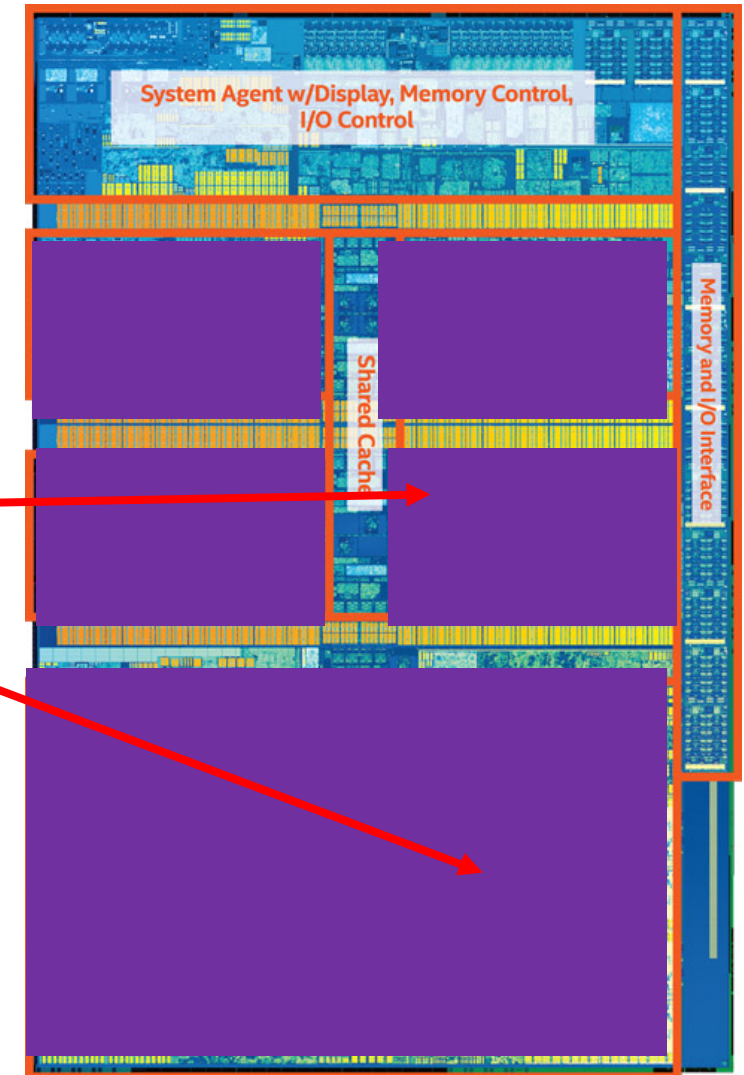
Why do we need to program for GPU?



DEEP
LEARNING
INSTITUTE



Using heterogeneous programming allows you to dispatch and fully utilise the entire chip.



VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

IT4INNOVATIONS
NATIONAL SUPERCOMPUTING
CENTER

Why do we need to program for GPU?



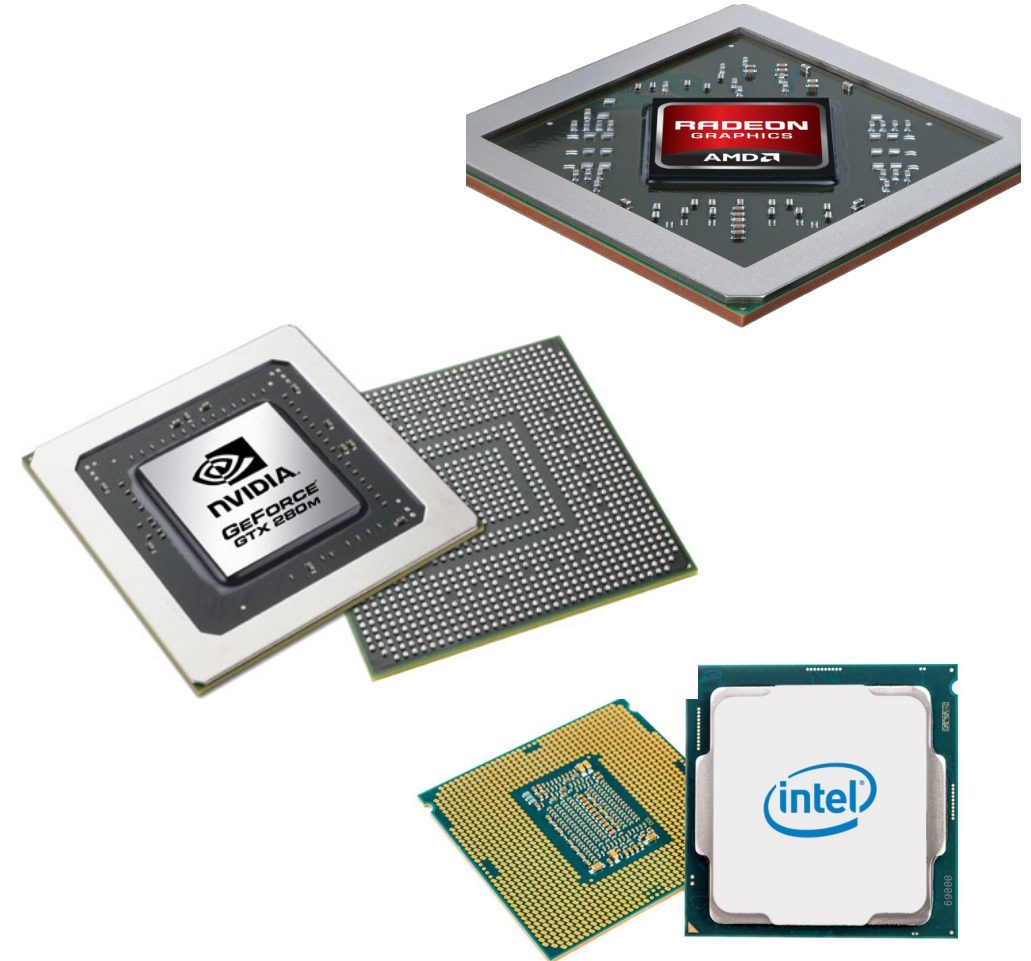
DEEP
LEARNING
INSTITUTE



GPU programming:

- *Limited only to a specific domain*
- *Separate source solutions*
- *Verbose low Level APIs*

- C++ AMP
- **CUDA C/C++**
- Kokkos
- HPX
- Raja
- SYCL
- NVPTX




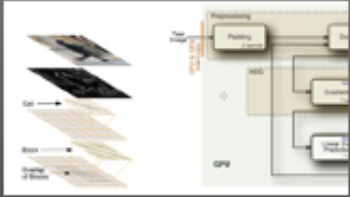
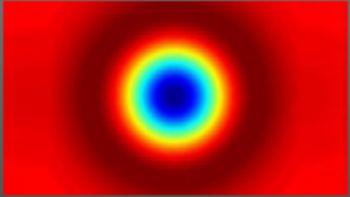
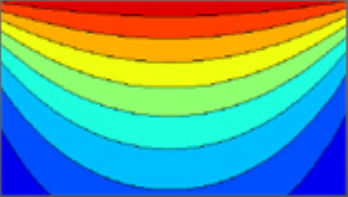
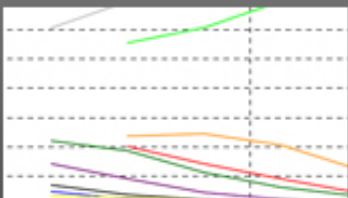

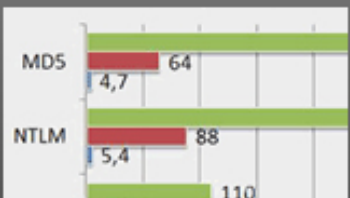
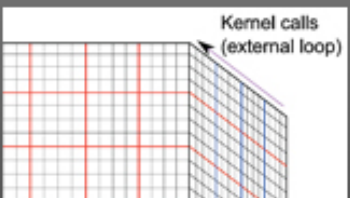

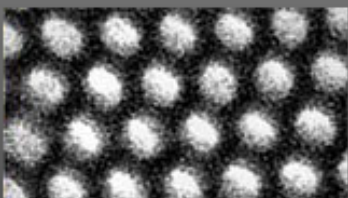
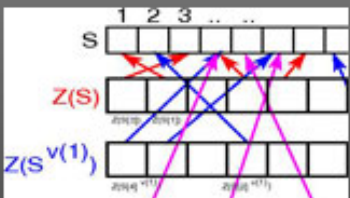
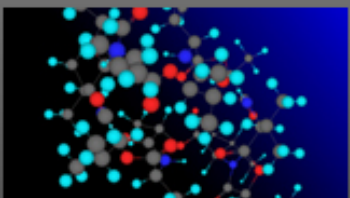
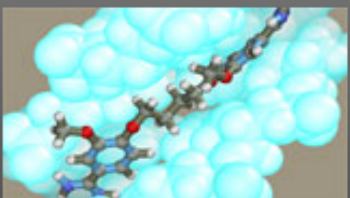
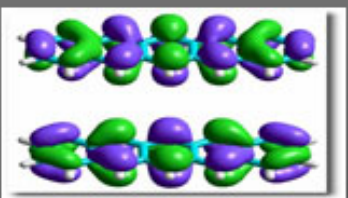
VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

IT4INNOVATIONS
NATIONAL SUPERCOMPUTING
CENTER

Why do we need GPUs on HPC?



- Increase in parallelism
- Today almost a **similar amount of efforts** on using CPUs vs GPUs by real applications
- GPUs well-suited to deep learning.

 <p>Cubic Interpolation 327 x</p>	 <p>Sliding-Windows for Rapid Object Class Localization: A Parallel Technique 109 x</p>	 <p>Jacket: GPU Engine for MATLAB 50 x</p>	$\sum_{k=0}^{K-1} x_m[i-k]v$ <p>FIR and QR Decomposition on GPUs 35 x</p>	 <p>Computational Fluid Dynamics (CFD) using GPUs 17 x</p>
 <p>Graphic-Card Cluster for Astrophysics (GraCCA) 250 x</p>	 <p>GpuCV: GPU-accelerated Computer vision library 100 x</p>	 <p>Distributed Password Recovery 50 x</p>	 <p>General Relativistic Evolution Code 26 x</p>	 <p>Highly Optimized Object-oriented Molecular Dynamics: HOOMD 15 x</p>
 <p>Quantum Chemistry Two-Electron Integral Evolution 130 x</p>	 <p>A Fast Similarity Join Algorithm 100 x</p>	 <p>Accelerating Density Functional Calculations with GPU 40 x</p>	 <p>Molecular Dynamics of DNA and Liquids 18 x</p>	 <p>Computational Chemistry Using GPUs 4.3 x</p>

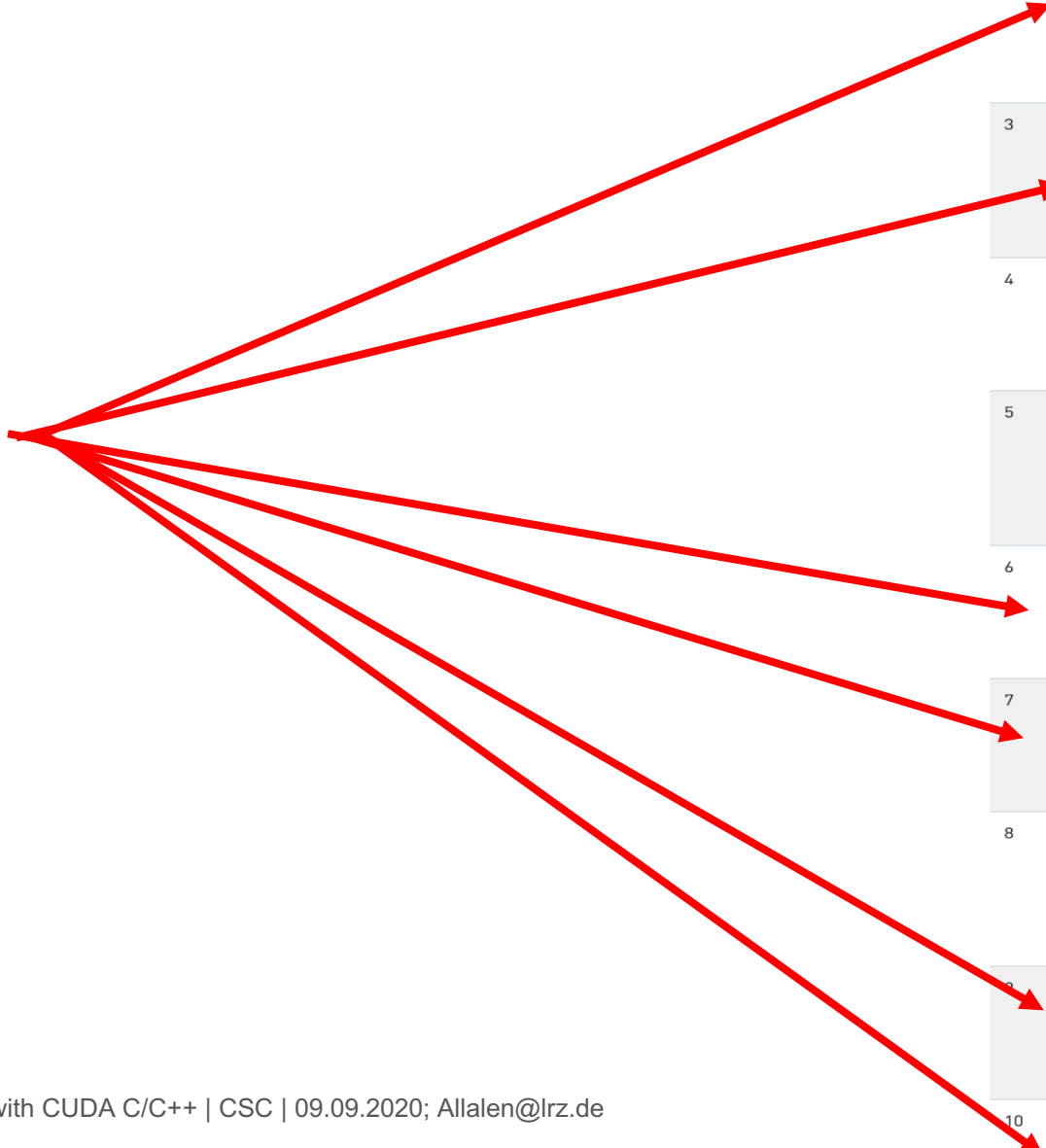
NVIDIA Software uses CUDA



Why do we need “accelerators” on HPC?

Top500.org

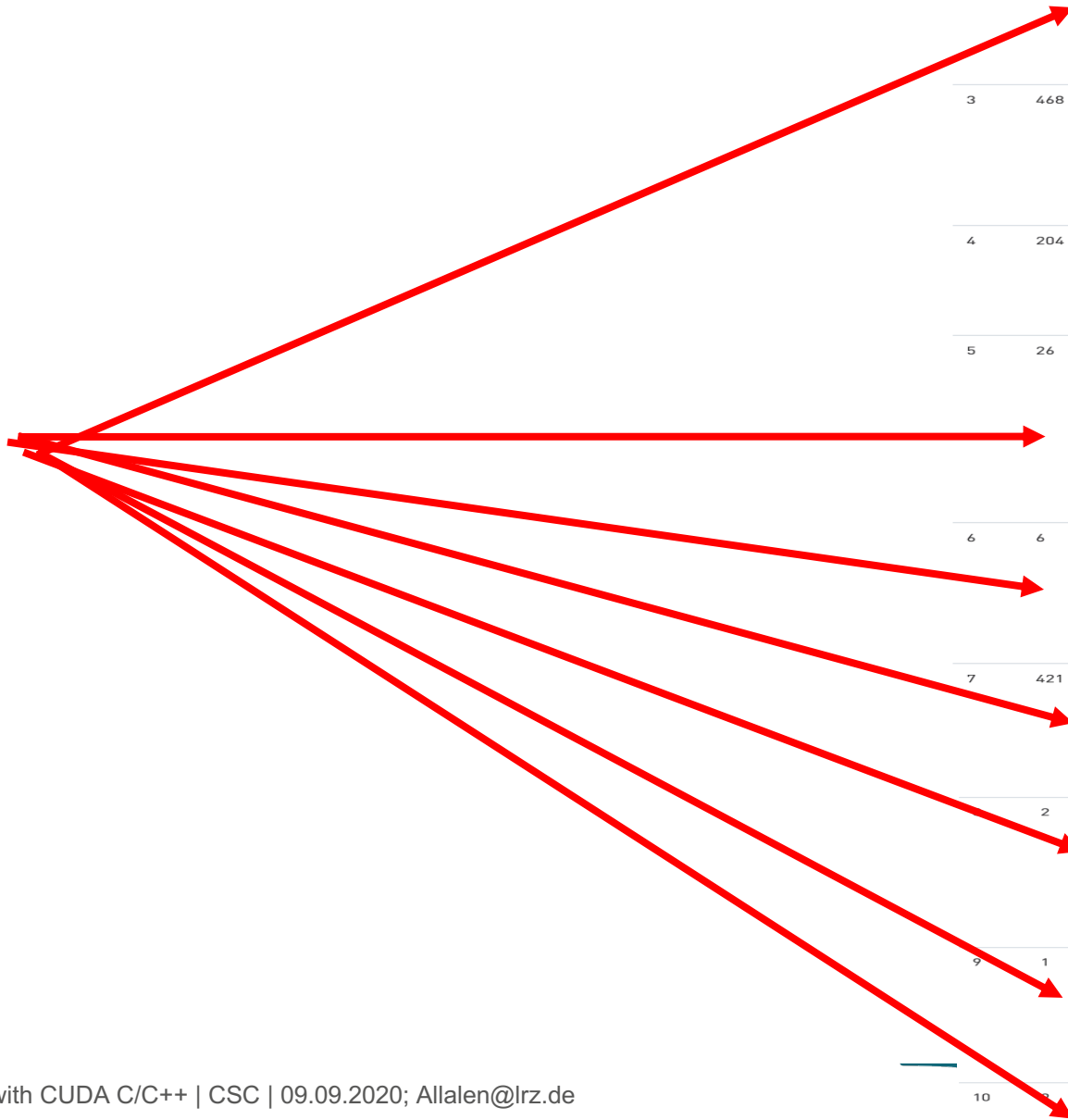
NVIDIA
GPUs



Rank	System	Cores	(TFlop/s)	(TFlop/s)	(kW)
1	Supercomputer Fugaku - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,299,072	415,530.0	513,854.7	28,335
2	Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	Sierra - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	Sunway TaihuLight - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	Tianhe-2A - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
6	HPC5 - PowerEdge C4140, Xeon Gold 6252 24C 2.1GHz, NVIDIA Tesla V100, Mellanox HDR Infiniband, Dell EMC Eni S.p.A. Italy	669,760	35,450.0	51,720.8	2,252
7	Selene - DGX A100 SuperPOD, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	272,800	27,580.0	34,568.6	1,344
8	Frontera - Dell C6420, Xeon Platinum 8280 28C 2.7GHz, Mellanox InfiniBand HDR, Dell EMC Texas Advanced Computing Center/Univ. of Texas United States	448,448	23,516.4	38,745.9	
9	Marconi-100 - IBM Power System AC922, IBM POWER9 16C 3GHz, Nvidia Volta V100, Dual-rail Mellanox EDR Infiniband, IBM CINECA Italy	347,776	21,640.0	29,354.0	1,476
10	Piz Daint - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA	387,872	21,230.0	27,154.3	2,384

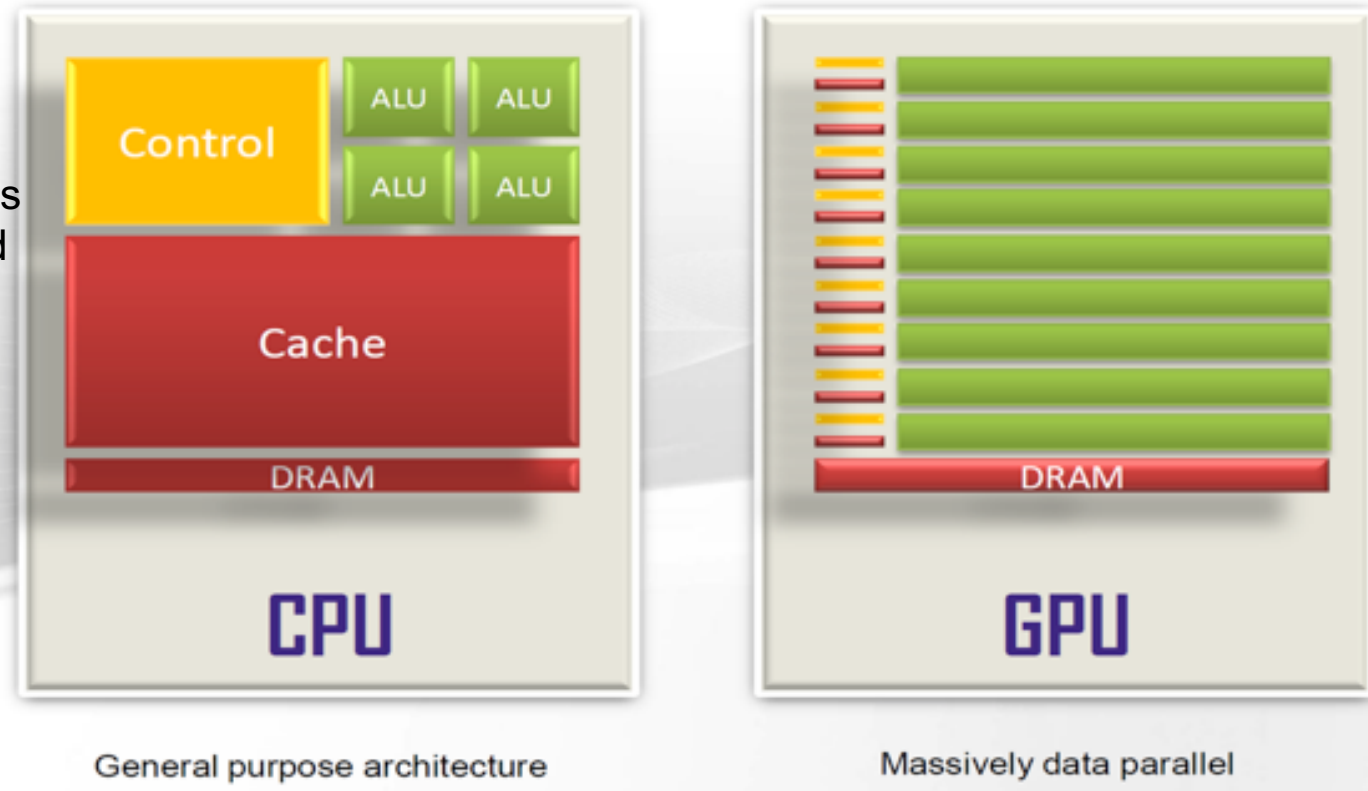
Why do we need “accelerators” on HPC? Green top500

NVIDIA
GPUs



Rank	Rank	System	Cores	(TFlop/s)	(kW)	(GFlops/watts)
1	393	MN-3 - MN-Core Server, Xeon 8260M 24C 2.4GHz, MN-Core, RoCEv2/MN-Core DirectConnect, Preferred Networks Preferred Networks Japan	2,080	1,621.1	77	21.108
2	7	Selene - DGX A100 SuperPOD, AMD EPYC 7742 64C 2.25GHz, NVIDIA A100, Mellanox HDR Infiniband, Nvidia NVIDIA Corporation United States	272,800	27,580.0	1,344	20.518
3	468	NA-1 - ZettaScaler-2.2, Xeon D-1571 16C 1.3GHz, Infiniband EDR, PEZY-SC2 700Mhz, PEZY Computing / Exascaler Inc. PEZY Computing K.K. Japan	1,271,040	1,303.2	80	18.433
4	204	A64FX prototype - Fujitsu A64FX, Fujitsu A64FX 48C 2GHz, Tofu interconnect D, Fujitsu Fujitsu Numazu Plant Japan	36,864	1,999.5	118	16.876
5	26	AiMOS - IBM Power System AC922, IBM POWER9 20C 3.45GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM Rensselaer Polytechnic Institute Center for Computational Innovations [CCI] United States	130,000	8,339.0	512	16.285
6	6	HPC5 - PowerEdge C4140, Xeon Gold 6252 24C 2.1GHz, NVIDIA Tesla V100, Mellanox HDR Infiniband, Dell EMC Eni S.p.A. Italy	669,760	35,450.0	2,252	15.740
7	421	Satori - IBM Power System AC922, IBM POWER9 20C 2.4GHz, Infiniband EDR, NVIDIA Tesla V100 SXM2, IBM MIT/MGHPCC Holyoke, MA United States	23,040	1,464.0	94	15.574
	2	Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	10,096	14.719
	9	Supercomputer Fugaku - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,299,072	415,530.0	28,335	14.665
	10	Marconi-100 - IBM Power System AC922, IBM POWER9 16C 3GHz,	347,776	21,640.0	1,476	14.661

GPU vs CPU Architecture



- * Small number of large cores
- * More control structures and less processing units
- * Optimised for latency which requires quite a lot of power

- * Large number of small cores
- * Less control structured and more processing units
- * Less flexible program model
- * There're more restrictions but Requires a lot less power

• GPU devotes more transistors data processing rather than data caching and flow control. Same problem executed on many data elements in parallel.



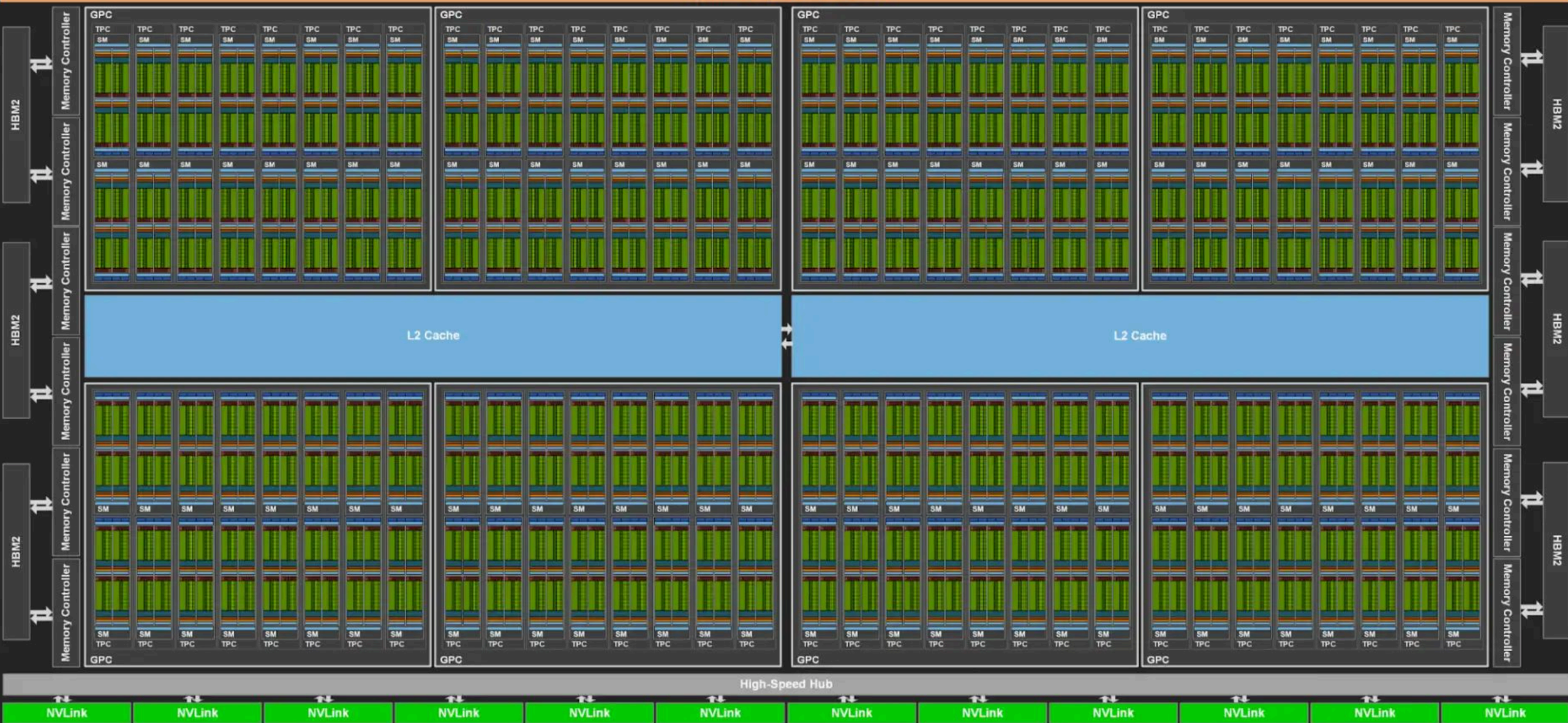
PCI Express 3.0 Host Interface

GigaThread Engine



PCI Express 4.0 Host Interface

GigaThread Engine with MIG Control



What is CUDA C/C++ ?



- **CUDA = "Compute Unified Device Architecture"**
 - * Introduced in 2006 *
 - GPU = dedicated super-threaded, massively data parallel - co-processor

C/C++ plus a few simple extensions

- Compute oriented drivers, language, and tools

Documentations:

[CUDA_C_Programming_Guide.pdf](#)

[CUDA_C_Getting_Started.pdf](#)

[CUDA_C_Toolkit_Release.pdf](#)



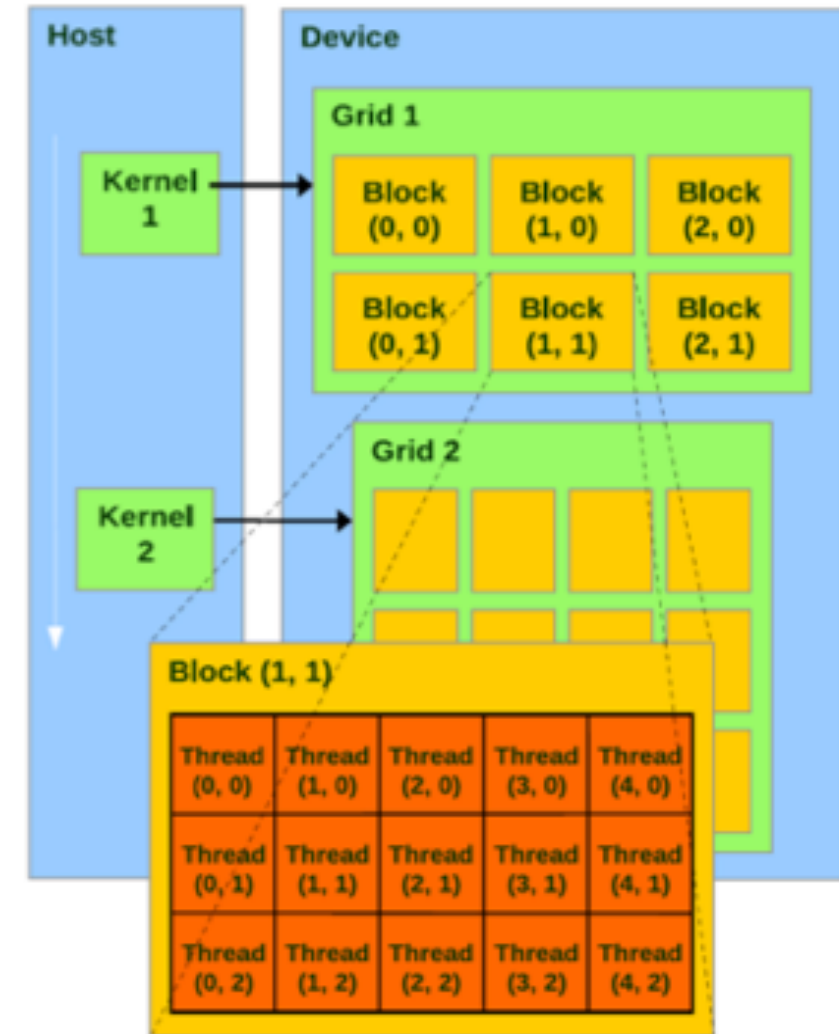
CUDA Programming Model



DEEP
LEARNING
INSTITUTE



- A kernel is executed as a grid of thread blocks
- All threads share data memory space
- A thread block is a batch of threads that can cooperate with each other by:
 - Synchronizing their execution
 - Efficiently sharing data through a low latency shared memory
- Two threads from two different blocks cannot cooperate
- Sequential code launches **asynchronously** GPU kernels



VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

IT4INNOVATIONS
NATIONAL SUPERCOMPUTING
CENTER

Terminology:

Host: The CPU and its memory
(host memory)



Host

Device: The GPU and its
memory (device memory)

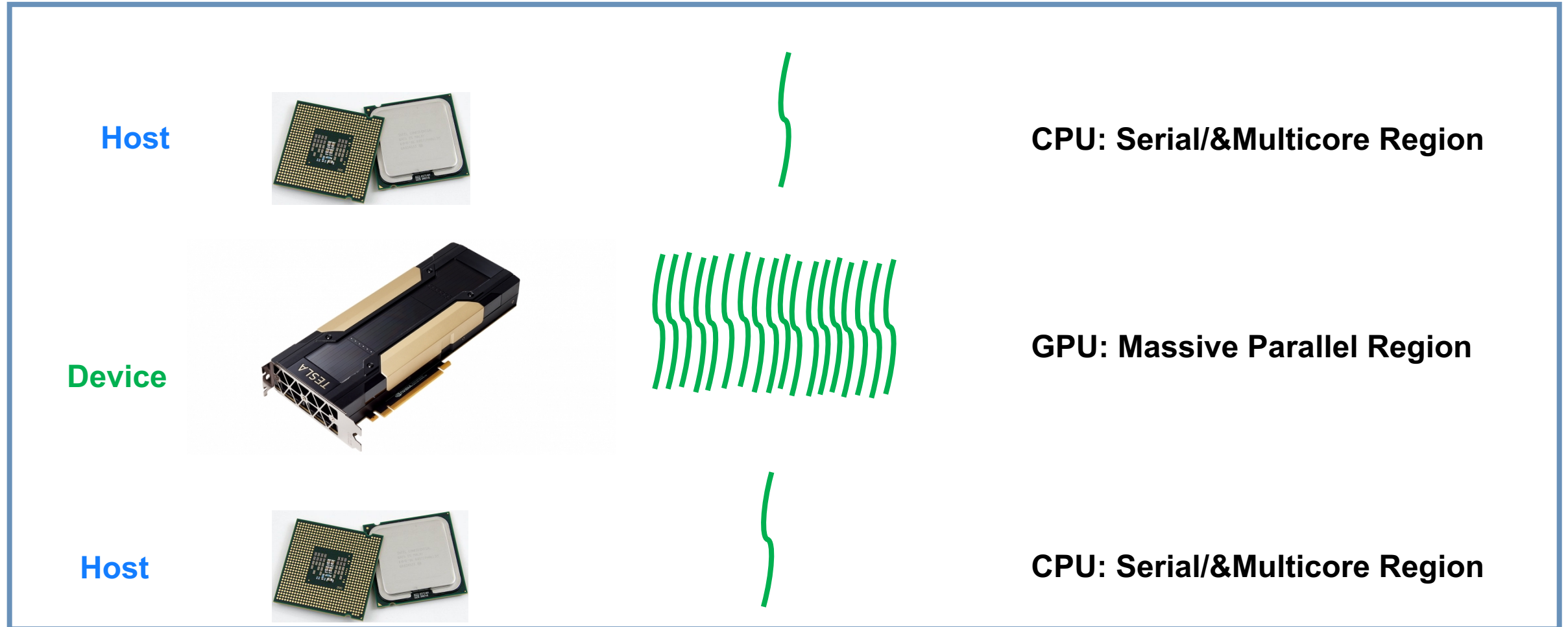


Device

CUDA Devices and Threads Execution Model



DEEP
LEARNING
INSTITUTE



VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

IT4INNOVATIONS
NATIONAL SUPERCOMPUTING
CENTER

CUDA C/C++



The CPU allocates memory on the GPU
The CPU copies data from CPU to GPU
The CPU launches kernels on the GPU
The CPU copies data to CPU from GPU



NVCC Compiler



- NVIDIA provides a CUDA-C compiler
- **nvcc**
- NVCC splits your code in 2: **Host** code and **Device** code.
- **Device** code sent to NVIDIA device compiler.

- **nvcc** is capable of linking together both host and device code into a single executable.
- **Convention:** C++ source files containing CUDA syntax are typically given the extension **.cu**.



DEEP
LEARNING
INSTITUTE



VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

IT4INNOVATIONS
NATIONAL SUPERCOMPUTING
CENTER

Lab1: Accelerating Applications with CUDA C/C++

Dr. Momme Allalen

Leibniz Computing Centre, Munich Germany - www.lrz.de

Deep Learning Certified Instructor, NVIDIA Deep Learning Institute NVIDIA Corporation.

Lab1: Accelerating Applications with CUDA C/C++



DEEP
LEARNING
INSTITUTE



Prerequisites

You should already be able to:

- Declare variables, write loops, and use if / else statements in C.
- Define and invoke functions in C.
- Allocate arrays in C.
- No previous CUDA knowledge is required.

Objectives

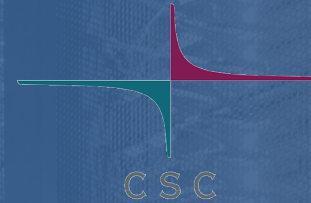
By the time you complete this lab, you will be able to:

- Write, compile, and run C/C++ programs that both call **CPU functions** and **launch GPU kernels**.
 - Control parallel **threadhierarchy** using **execution configuration**.
 - Refactor serial loops to execute their iterations in parallel on a **GPU**.
- Allocate and free memory available to both **CPUs** and **GPUs**.
 - Handle errors generated by CUDA code.
 - Accelerate **CPU-only applications**.





DEEP
LEARNING
INSTITUTE



VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

IT4INNOVATIONS
NATIONAL SUPERCOMPUTING
CENTER

Lab2: Managing Accelerated Application Memory with CUDA Unified Memory and nsys

Dr. Momme Allalen

Leibniz Computing Centre, Munich Germany - www.lrz.de

Deep Learning Certified Instructor, NVIDIA Deep Learning Institute NVIDIA Corporation.

Lab2: Managing Accelerated Application Memory with CUDA Unified Memory and nsys



Prerequisites

You should already be able to:

- Write, compile, and run C/C++ programs that both call CPU functions and launch GPU kernels.
- Control parallel thread hierarchy using execution configuration.
- Refactor serial loops to execute their iterations in parallel on a GPU.
- Allocate and free Unified Memory.

Objectives

- By the time you complete this lab, you will be able to:
- Use the **NVIDIA Command Line Profiler (nprof)** to profile accelerated application performance.
 - Understanding of **Streaming Multiprocessors** to optimize execution configurations.
 - Understand the behavior of **Unified Memory** with regard to page faulting and data migrations.
 - Use **asynchronous memory prefetching** to reduce page faults and data migrations for increased performance.
 - Employ an iterative development cycle to rapidly accelerate and deploy applications.

CUDA® PROFILING TOOLS



DEEP
LEARNING
INSTITUTE



nvvp: NVIDIA visual profiler

nvprof: tool to understand and optimize the performance of your CUDA,

OpenACC or OpenMP applications,

Application level opportunities

Overall application performance

Overlap CPU and GPU work, identify the bottlenecks (CPU or GPU)

Overall GPU utilization and efficiency

- Overlap compute and memory copies
- Utilize compute and copy engines effectively.

Kernel level opportunities

- Use memory bandwidth efficiently
- Use compute resources efficiently
- Hide instruction and memory latency

There are more features, example for Dependency Analysis

Command: **nvprof** --dependency-analysis --cpu-thread-tracing on ./executable_cuda



Nsight Systems
Nsight Compute



NSIGHT PRODUCT FAMILY



Standalone Performance Tools:

Ns- Systems – System-wide application algorithm tuning

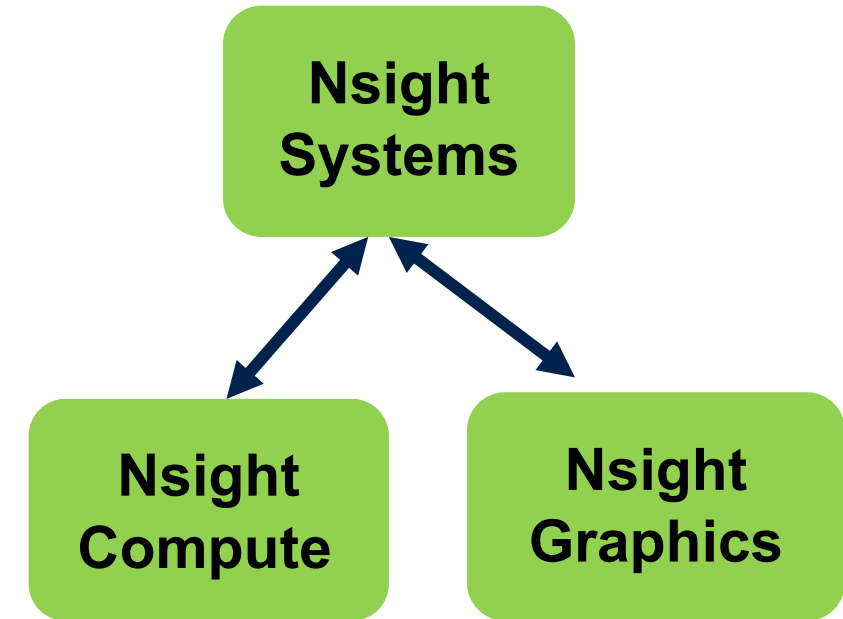
Ns- Compute – Debug/ & Profile specific CUDA kernels

Ns- Graphics – Analyze/ & Optimize specific graphics workloads

IDE Plugins

Nsight Eclipse Edition/Visual Studio – editor, debugger, some perf analysis

Nvprof will be replaced with **nsys –profile=true**



Docs/product: <https://developer.nvidia.com/nsight-systems>



NSIGHT SYSTEMS



DEEP
LEARNING
INSTITUTE



System-wide application algorithm tuning
Multi-process tree support

Locate optimization opportunities
Visualize millions of events on a very fast GUI timeline
Or gaps of unused CPU and GPU time

Balance your workload across multiple CPUs and GPUs
CPU algorithms, utilization, and thread state
GPU streams, kernels, memory transfers, etc

Multi-platform: Linux & Windows, x86-64, Tegra, Power, MacOSX (host only)

GPUs: Volta, Turing

Docs/product: <https://developer.nvidia.com/nsight-systems>



VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

IT4INNOVATIONS
NATIONAL SUPERCOMPUTING
CENTER

CUDA Kernel profiler

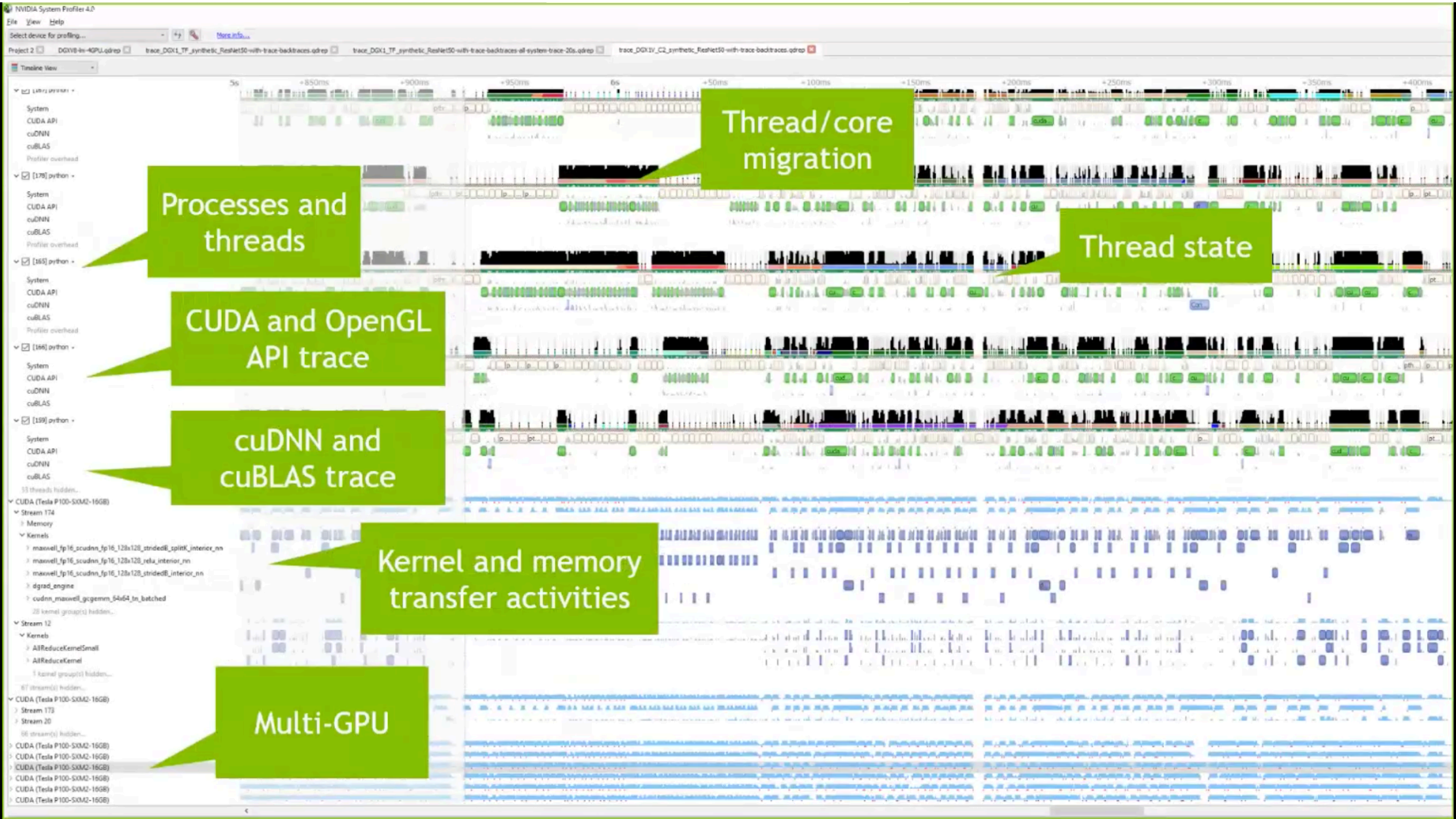
Targeted metric sections for various performance aspects (Debug/Profile)

Very high freq GPU perf counter, customizable data collection and presentation (tables, charts ...)

Python-based rules for guided analysis (or postprocessing)

GPUs: Volta, Turing, Amper

Docs/product: <https://developer.nvidia.com/nsight-systems>



Processes and threads

Thread/core migration

Thread state

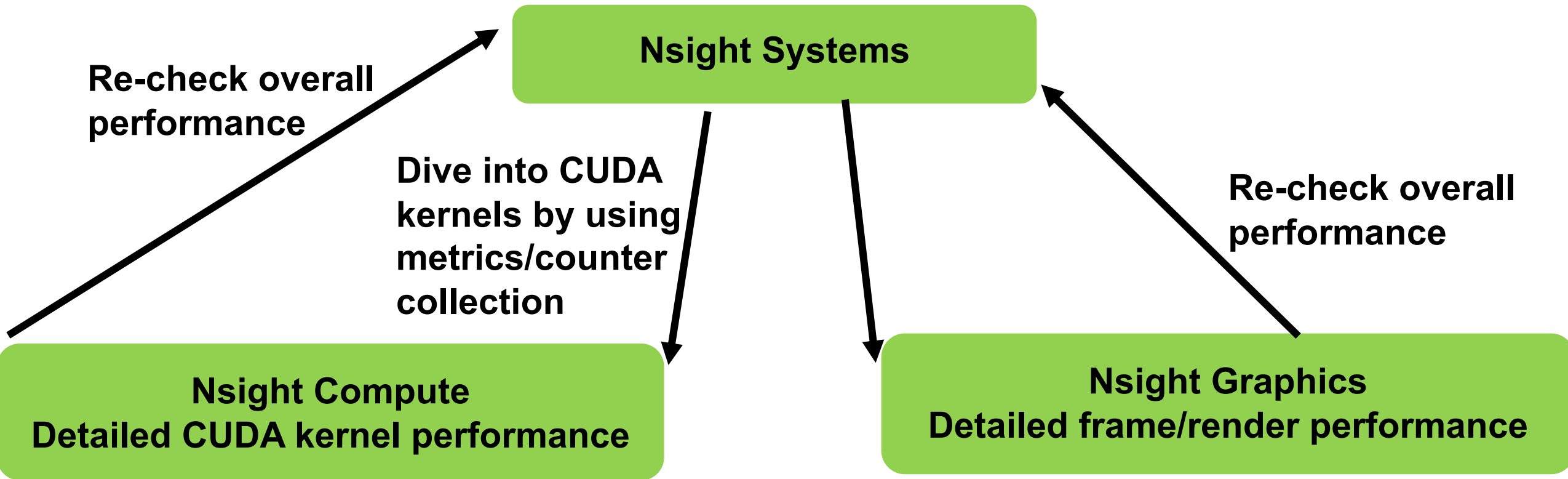
CUDA and OpenGL API trace

cuDNN and cuBLAS trace

Kernel and memory transfer activities

Multi-GPU

NSIGHT PRODUCT FAMILY



Nsight Systems - Analyze application algorithm system-wide

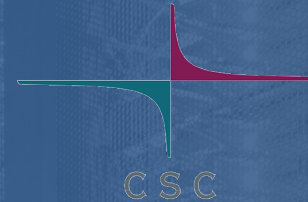
Nsight Compute - Debug/optimize CUDA kernel

Nsight Graphics - Debug/optimize graphics workloads





DEEP
LEARNING
INSTITUTE



VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

IT4INNOVATIONS
NATIONAL SUPERCOMPUTING
CENTER

Lab3: Asynchronous Streaming, and Visual Profiling with CUDA C/C++

Dr. Momme Allalen

Leibniz Computing Centre, Munich Germany - www.lrz.de

Deep Learning Certified Instructor, NVIDIA Deep Learning Institute NVIDIA Corporation.

Lab3: Asynchronous Streaming, and Visual Profiling

With CUDA C/C++



Prerequisites

To get the most out of this lab you should already be able to:

- Write, compile, and run C/C++ programs that both call CPU functions and launch GPU kernels.
- Control parallel thread hierarchy using execution configuration.
- Refactor serial loops to execute their iterations in parallel on a GPU.
- Allocate and free CUDA Unified Memory.
- Understand the behaviour of Unified Memory with regard to page faulting and data migrations.
- Use asynchronous memory prefetching to reduce page faults and data migrations.

Objectives

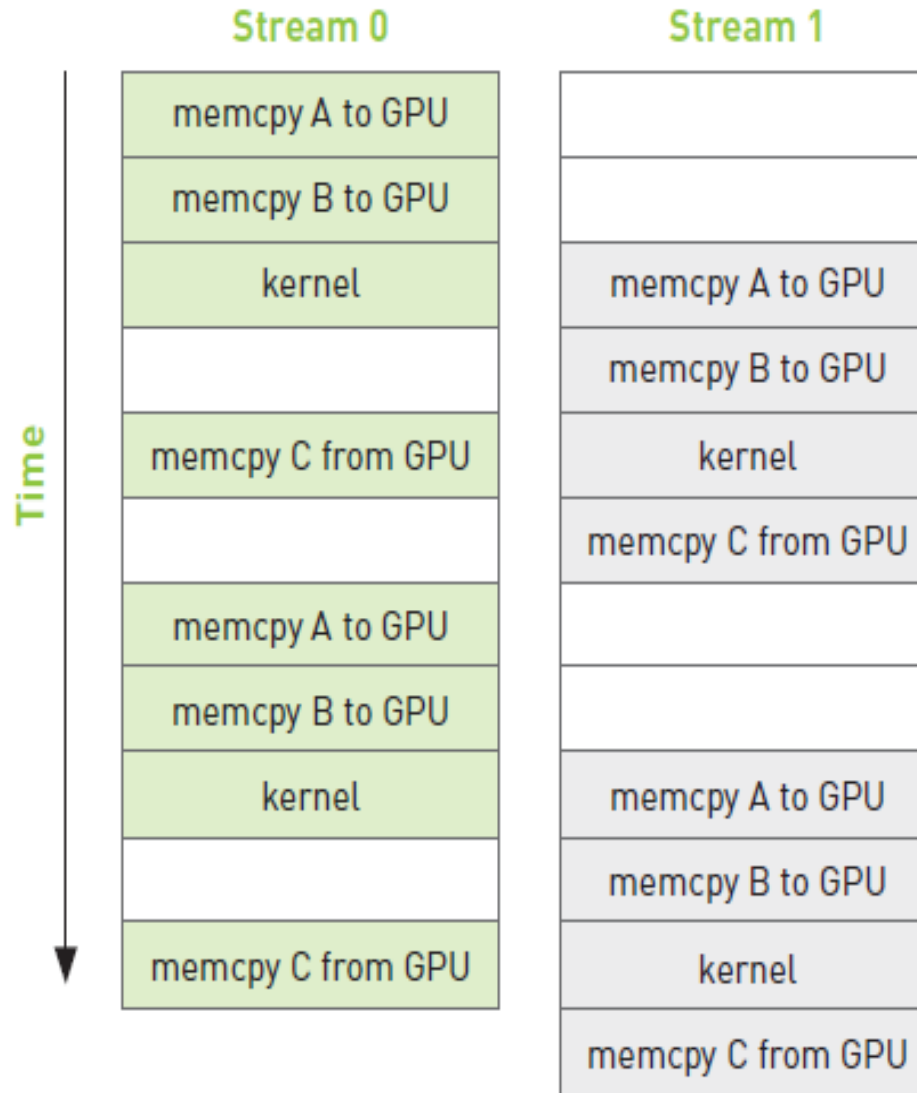
By the time you complete this lab you will be able to:

- Use the **Nsight Systems** to visually profile the timeline of GPU-accelerated CUDA applications.
- Use Nsight Systems to identify, and exploit, optimization opportunities in GPU-accelerated CUDA applications.
- Utilize CUDA streams for concurrent kernel execution in accelerated applications.
- **(Optional Advanced Content)** Use manual memory allocation, including allocating pinned memory, in order to asynchronously transfer data in concurrent CUDA streams.

Multiple Streams



Overlap copy
with kernel



Multiple Streams



```
for (int i=0; i<FULL_SIZE; i+= N*2) {
// copy the locked memory to the device, async
cudaMemcpyAsync (dev_a0, host_a+i, N * sizeof(int), cudaMemcpyHostToDevice, stream0);
cudaMemcpyAsync (dev_b0, host_b+i, N * sizeof(int), cudaMemcpyHostToDevice, stream0);

kernel<<<N/256,256,0,stream0>>>( dev_a0, dev_b0, dev_c0 );

// copy the data from device to locked memory
cudaMemcpyAsync (host_c+i, dev_c0, N * sizeof(int), cudaMemcpyDeviceToHost, stream0);
// copy the locked memory to the device, async
cudaMemcpyAsync (dev_a1,host_a+i+N, N * sizeof(int), cudaMemcpyHostToDevice, stream1);
cudaMemcpyAsync (dev_b1,host_b+i+N, N * sizeof(int), cudaMemcpyHostToDevice, stream1);

kernel<<<N/256,256,0,stream1>>>( dev_a1, dev_b1, dev_c1 );

// copy the data from device to locked memory
cudaMemcpyAsync (host_c+i+N,dev_c1, N * sizeof(int), cudaMemcpyDeviceToHost, stream1);
}
```



DEEP
LEARNING
INSTITUTE



VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

IT4INNOVATIONS
NATIONAL SUPERCOMPUTING
CENTER

THANK YOU

Instructor: Dr. Momme Allalen
www.nvidia.com/dli