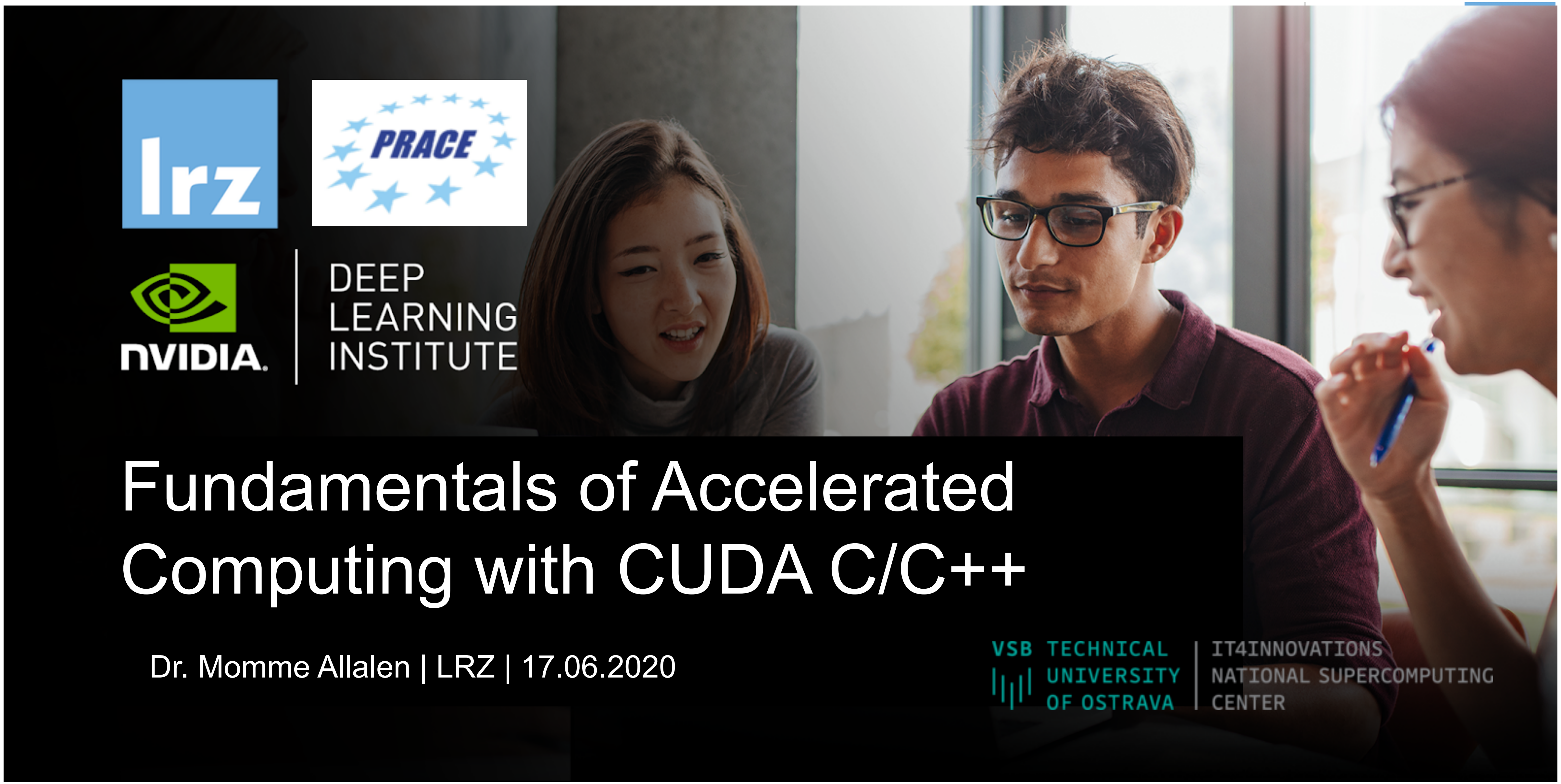


lrz

PRACE

NVIDIA | DEEP LEARNING INSTITUTE

Fundamentals of Accelerated Computing with CUDA C/C++

Dr. Momme Allalen | LRZ | 17.06.2020

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

Overview

- The workshop is co-organized by LRZ, IT4Innovations and NVIDIA Deep Learning Institute (DLI) for the Partnership for Advanced Computing in Europe (PRACE).
- LRZ as part of GCS is PRACE Training Centres which serve as European hubs and key drivers of advanced high-quality training for researchers working in the computational sciences.
- This training is a part of NVIDIA AI & HPC
- NVIDIA Deep Learning Institute (DLI) offers hands-on training for developers, data scientists, and researchers looking to solve challenging problems with deep learning.
- 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)

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

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.

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
10:15-12:00	Part1: Accelerating Applications with CUDA C/C++
12:00-13:00	**Lunch**
13:00-14:20	Part2: Managing Accelerated Application Memory with CUDA Unified Memory and nsys
14:20-14:30	**Break**
14:30-15:45	Part3: Asynchronous Streaming and Visual Profiling for Accelerated. Applications with CUDA C/C++
15:45-16:00	Q&A, Final Remarks

Training Setup

- 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.

Lecture Material

- **Lecture material:**
  - https://tinyurl.com/dl-gpu-workshop

- **Access Codes:**
- see the: Chat Window

And now …

Enjoy the course!

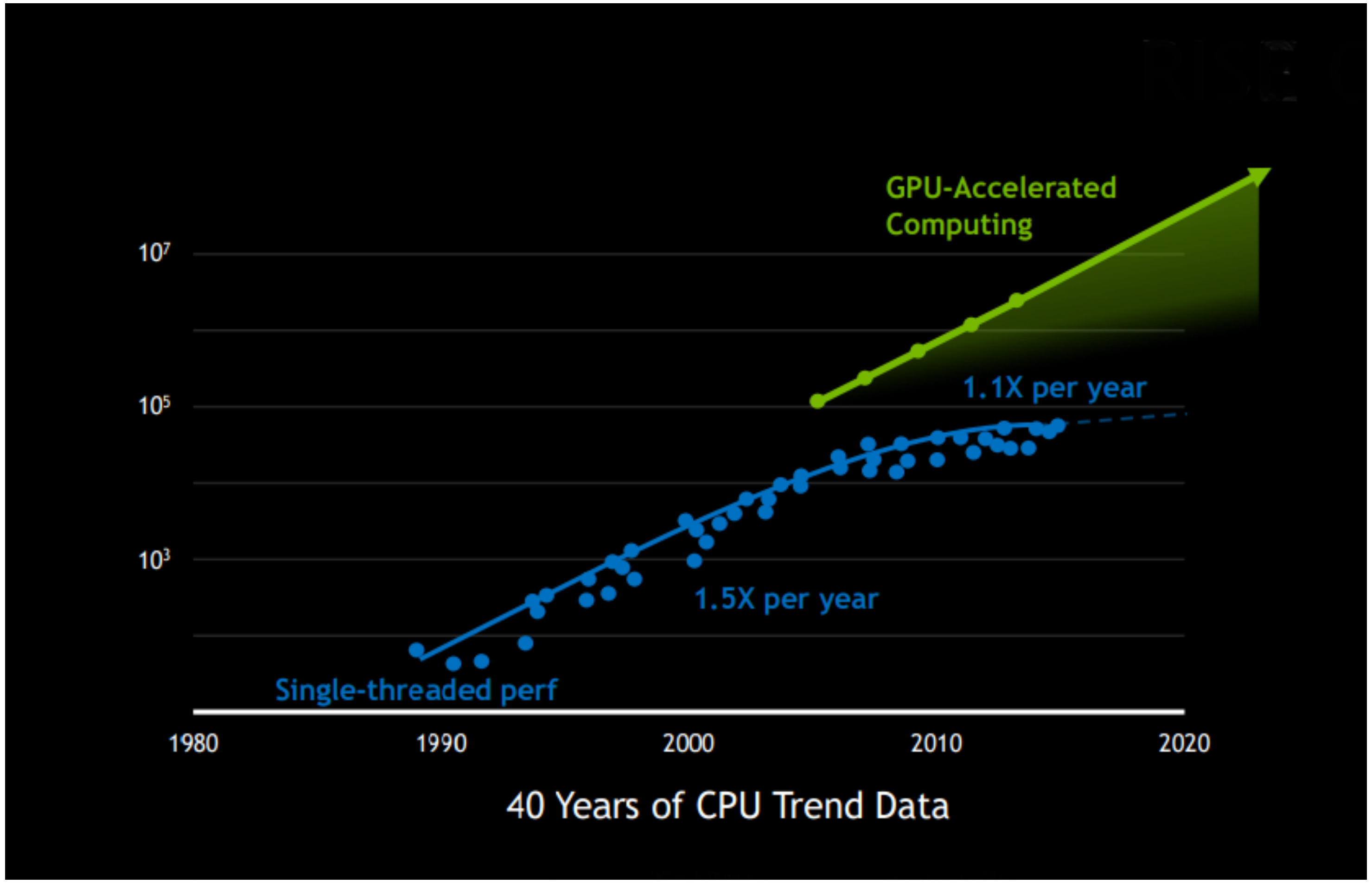
Why do we need to program for GPU?

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 GPUs on HPC ?

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

Limitations of single core architectures: High power consumption due to high clock frequency or a heat generation (cooling is expensive).

Today, processor cores are not getting any faster, but instead the number of cores per chip increases and registers are getting wider

On HPC, we need a chip that can provide:

Higher computing performance

@high power efficiency: keep the power/core as low as possible

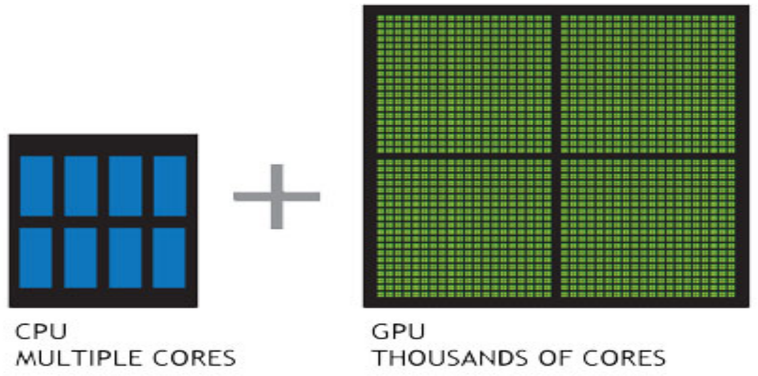
Why do we need GPUs on HPC ?

One solution is a heterogeneous system containing both CPUs and “accelerators”, plus other forms of parallelism such as vector instruction support …etc.

GPU computing @HPC:
Heterogeneous GPU systems are relatively complex to program.

Documentations:
- PRACE " Best Practice Guide – GPGPU”
- NVIDIA CUDA C Best Practices Guide

CUDA supports many, if not most, of the world's most performant applications in, Computational Fluid Dynamics, Molecular Dynamics, Quantum Chemistry and Physics.



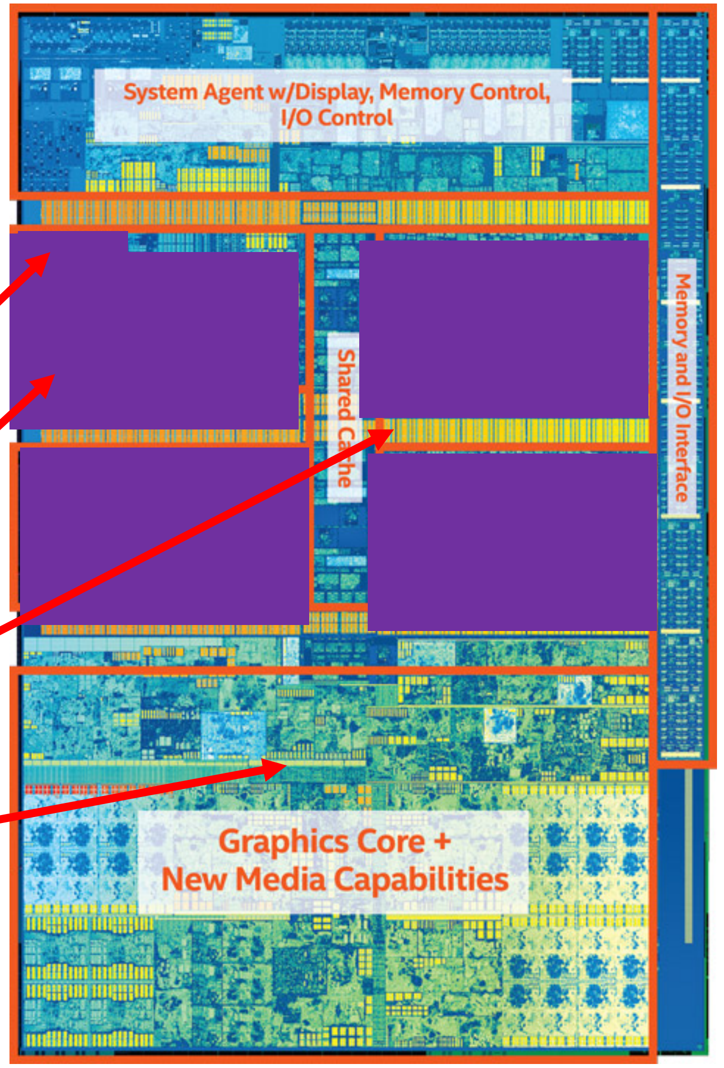
Why do we need to program for GPU?

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

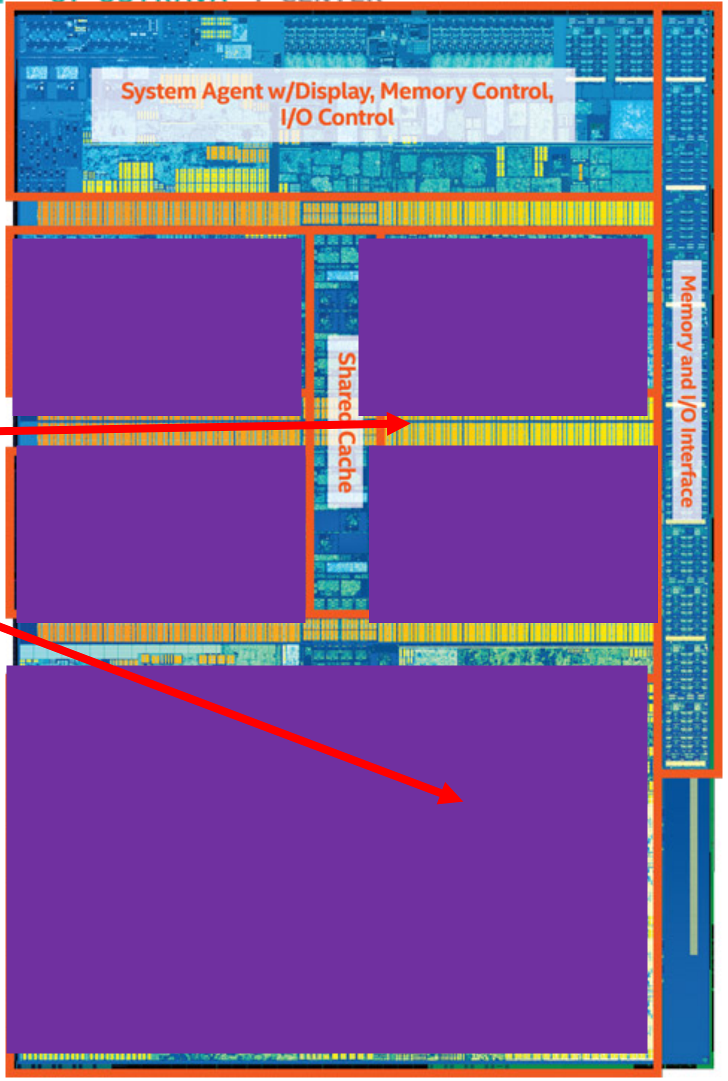
GPU need to be used?


System Agent w/Display, Memory Control, I/O Control
Shared Cache
Memory and I/O Interface
Graphics Core + New Media Capabilities

Intel Kaby Lake-S

Why do we need to program for GPU?

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

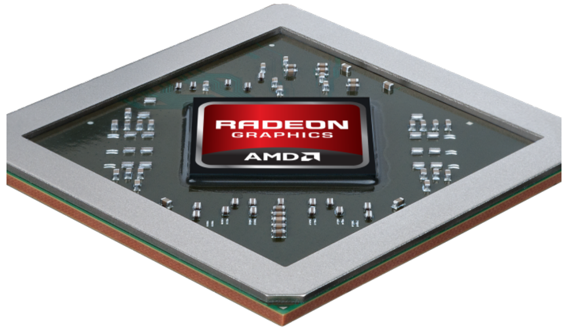


Intel Kaby Lake-S

Why do we need to program for GPU?

GPU programming:

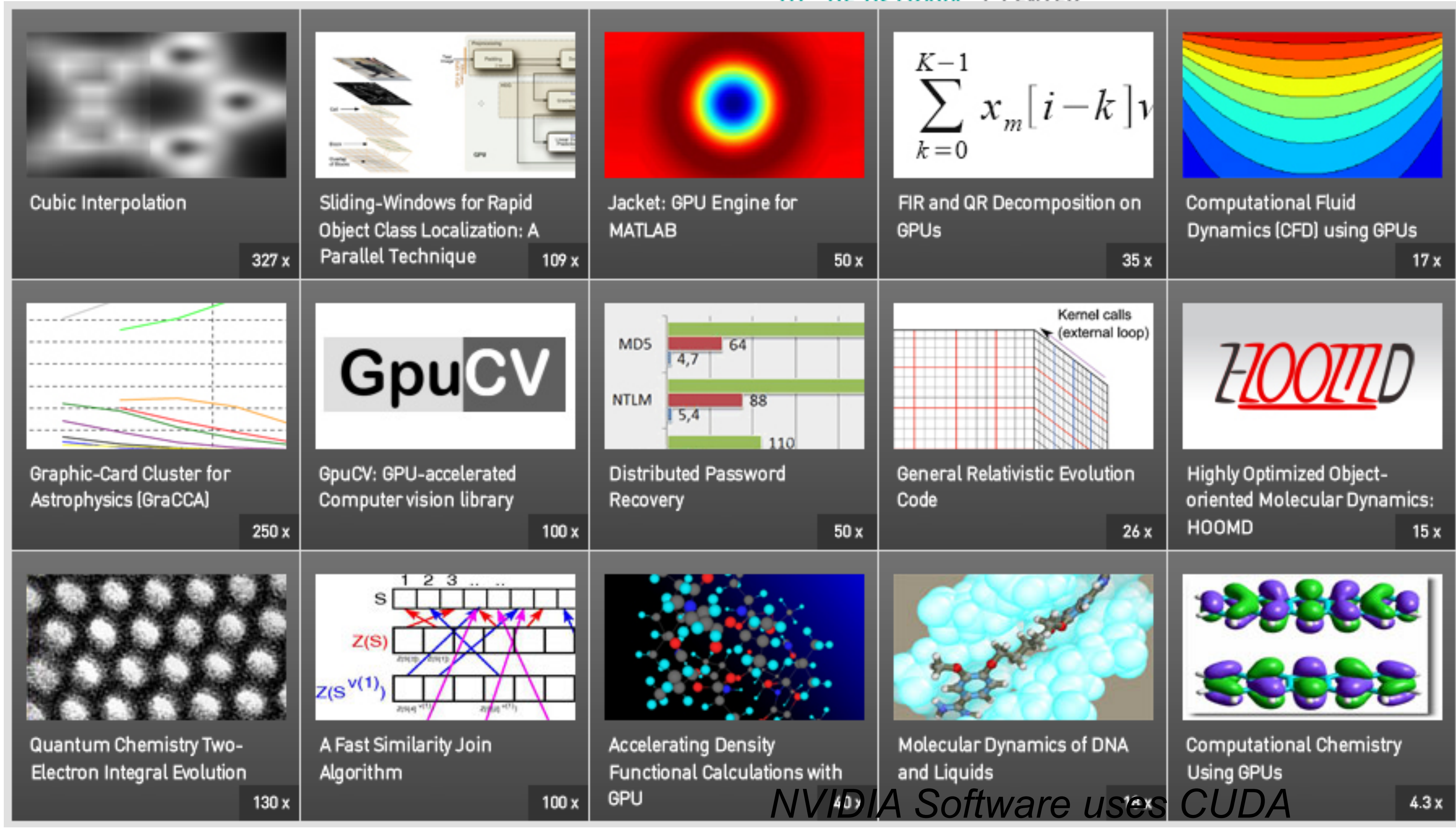
- *Limited only to a specific domain*
- *Separate source solutions*
- *Verbose low Level APIs*



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

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.

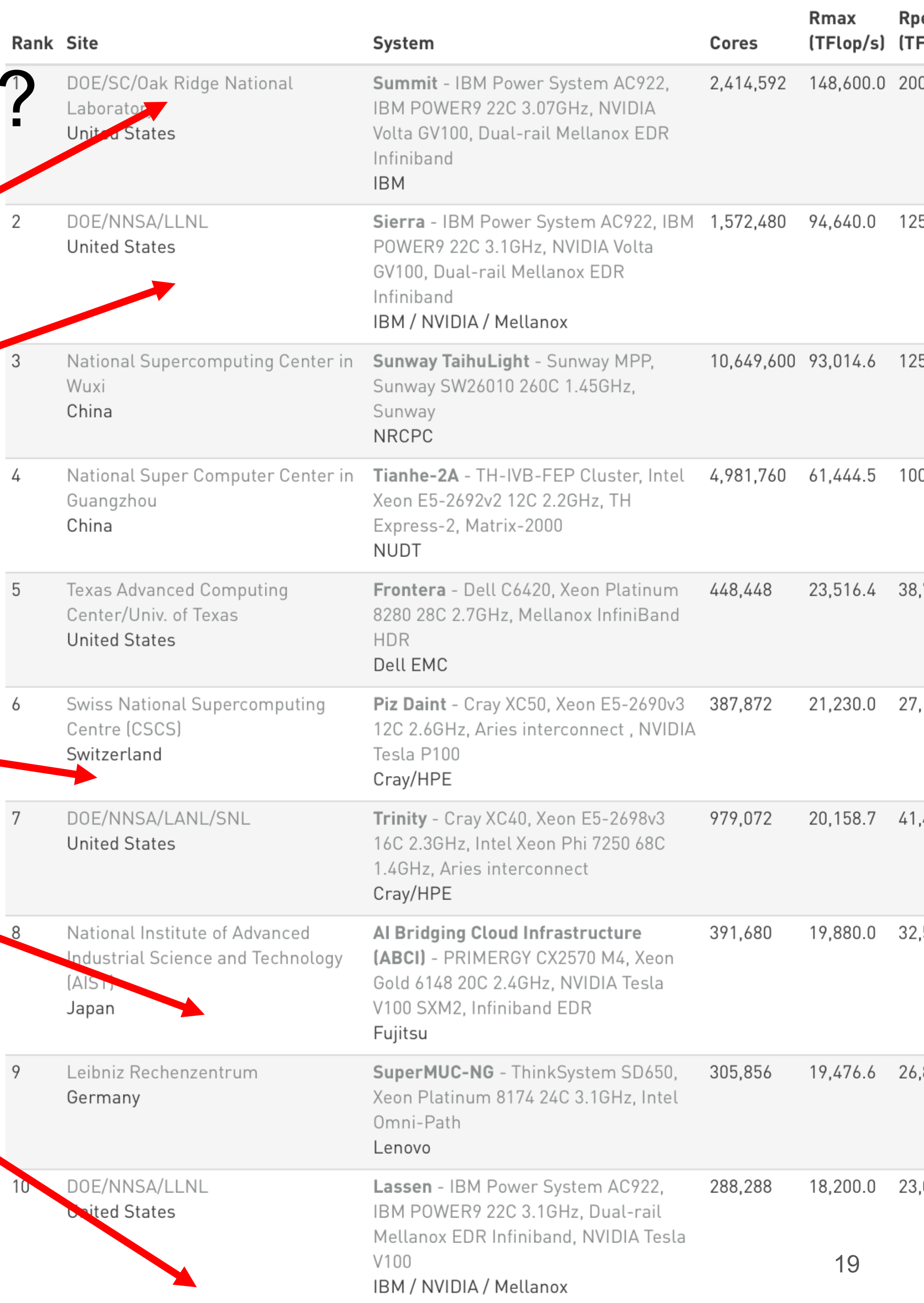


Cubic Interpolation	327 x
Sliding-Windows for Rapid Object Class Localization: A Parallel Technique	109 x
Jacket: GPU Engine for MATLAB	50 x
FIR and QR Decomposition on GPUs	35 x
Computational Fluid Dynamics (CFD) using GPUs	17 x
Graphic-Card Cluster for Astrophysics (GraCCA)	250 x
GpuCV: GPU-accelerated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

NVIDIA Software uses CUDA

Why do we need “accelerators” on HPC?
Top500.org

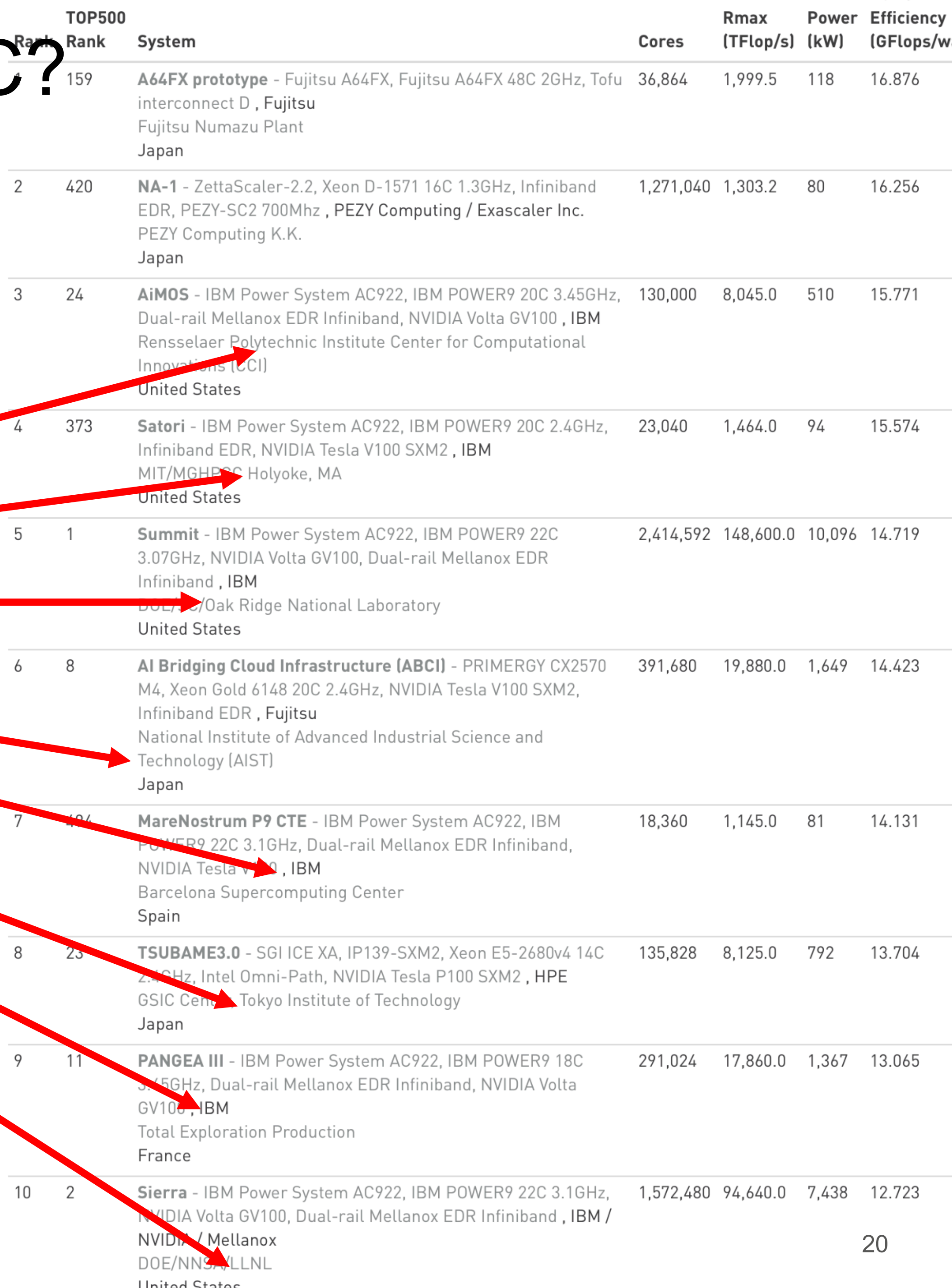
NVIDIA
GPUs



Rank	Site	System	Cores	Rmax (TFlop/s)	Rp (TF
1	DOE/SC/Oak Ridge National Laboratory United States	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM	2,414,592	148,600.0	200
2	DOE/NNSA/LLNL United States	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM / NVIDIA / Mellanox	1,572,480	94,640.0	125
3	National Supercomputing Center in Wuxi China	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway NRCPC	10,649,600	93,014.6	125
4	National Super Computer Center in Guangzhou China	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 NUDT	4,981,760	61,444.5	100
5	Texas Advanced Computing Center/Univ. of Texas United States	**Frontera** - Dell C6420, Xeon Platinum 8280 28C 2.7GHz, Mellanox InfiniBand HDR Dell EMC	448,448	23,516.4	38,
6	Swiss National Supercomputing Centre (CSCS) Switzerland	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 Cray/HPE	387,872	21,230.0	27,
7	DOE/NNSA/LANL/SNL United States	**Trinity** - Cray XC40, Xeon E5-2698v3 16C 2.3GHz, Intel Xeon Phi 7250 68C 1.4GHz, Aries interconnect Cray/HPE	979,072	20,158.7	41,
8	National Institute of Advanced Industrial Science and Technology (AIST) Japan	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2570 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR Fujitsu	391,680	19,880.0	32,
9	Leibniz Rechenzentrum Germany	**SuperMUC-NG** - ThinkSystem SD650, Xeon Platinum 8174 24C 3.1GHz, Intel Omni-Path Lenovo	305,856	19,476.6	26,
10	DOE/NNSA/LLNL United States	**Lassen** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, Dual-rail Mellanox EDR Infiniband, NVIDIA Tesla V100 IBM / NVIDIA / Mellanox	288,288	18,200.0	23,

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

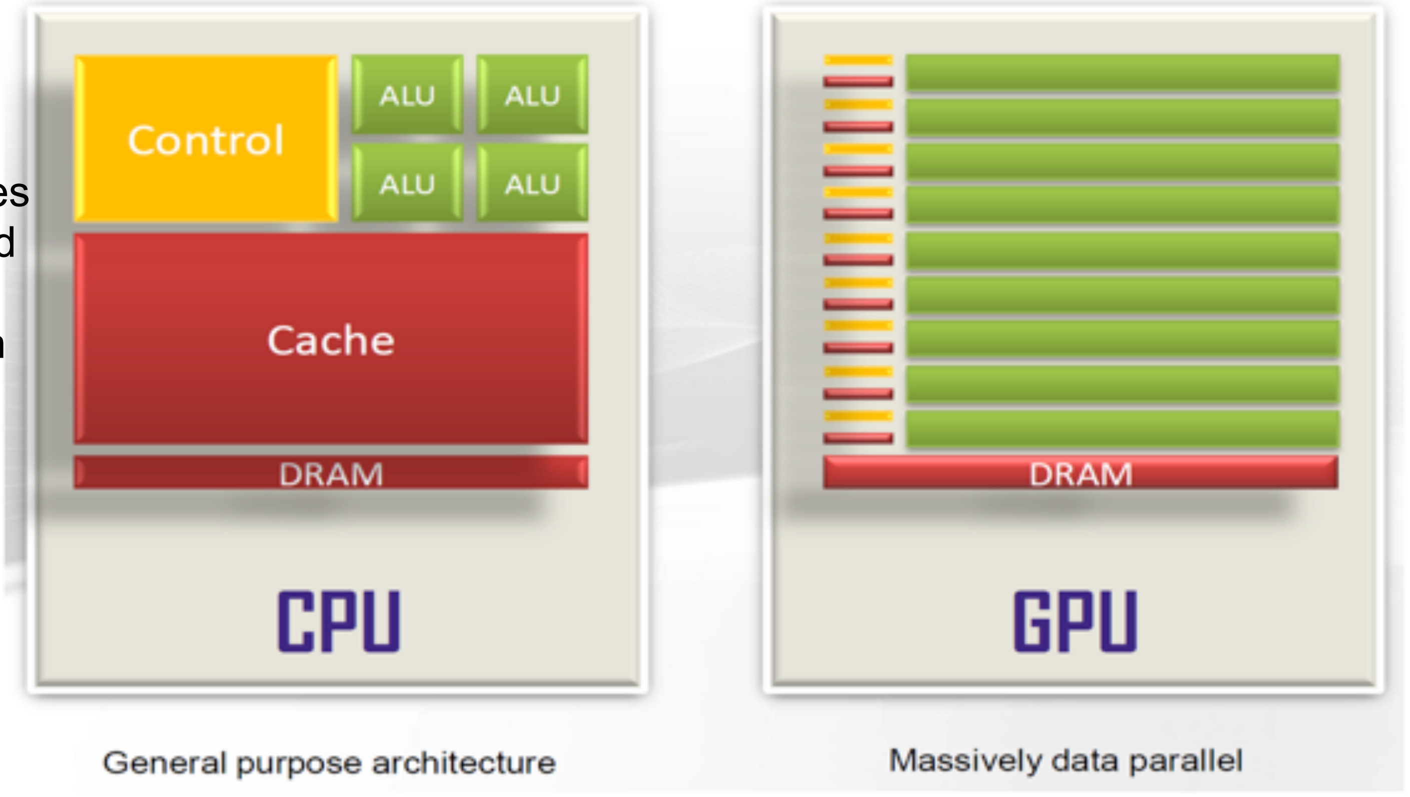
NVIDIA GPUs



Rank	TOP500 Rank	System	Cores	Rmax (TFlop/s)	Power (kW)	Efficiency (GFlops/w
1	159	**A64FX prototype** - Fujitsu A64FX, Fujitsu A64FX 48C 2GHz, Tofu interconnect D , Fujitsu Fujitsu Numazu Plant Japan	36,864	1,999.5	118	16.876
2	420	**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	16.256
3	24	**AiMOS** - IBM Power System AC922, IBM POWER9 20C 3.45GHz, Dual-rail Mellanox EDR Infiniband, NVIDIA Volta GV100 , IBM Rensselaer Polytechnic Institute Center for Computational Innovations (CCI) United States	130,000	8,045.0	510	15.771
4	373	**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
5	1	**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
6	8	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2570 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR , Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan	391,680	19,880.0	1,649	14.423
7	494	**MareNostrum P9 CTE** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, Dual-rail Mellanox EDR Infiniband, NVIDIA Tesla V100 , IBM Barcelona Supercomputing Center Spain	18,360	1,145.0	81	14.131
8	23	**TSUBAME3.0** - SGI ICE XA, IP139-SXM2, Xeon E5-2680v4 14C 2.4GHz, Intel Omni-Path, NVIDIA Tesla P100 SXM2 , HPE GSIC Center, Tokyo Institute of Technology Japan	135,828	8,125.0	792	13.704
9	11	**PANGEA III** - IBM Power System AC922, IBM POWER9 18C 3.45GHz, Dual-rail Mellanox EDR Infiniband, NVIDIA Volta GV100 , IBM Total Exploration Production France	291,024	17,860.0	1,367	13.065
10	2	**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	7,438	12.723

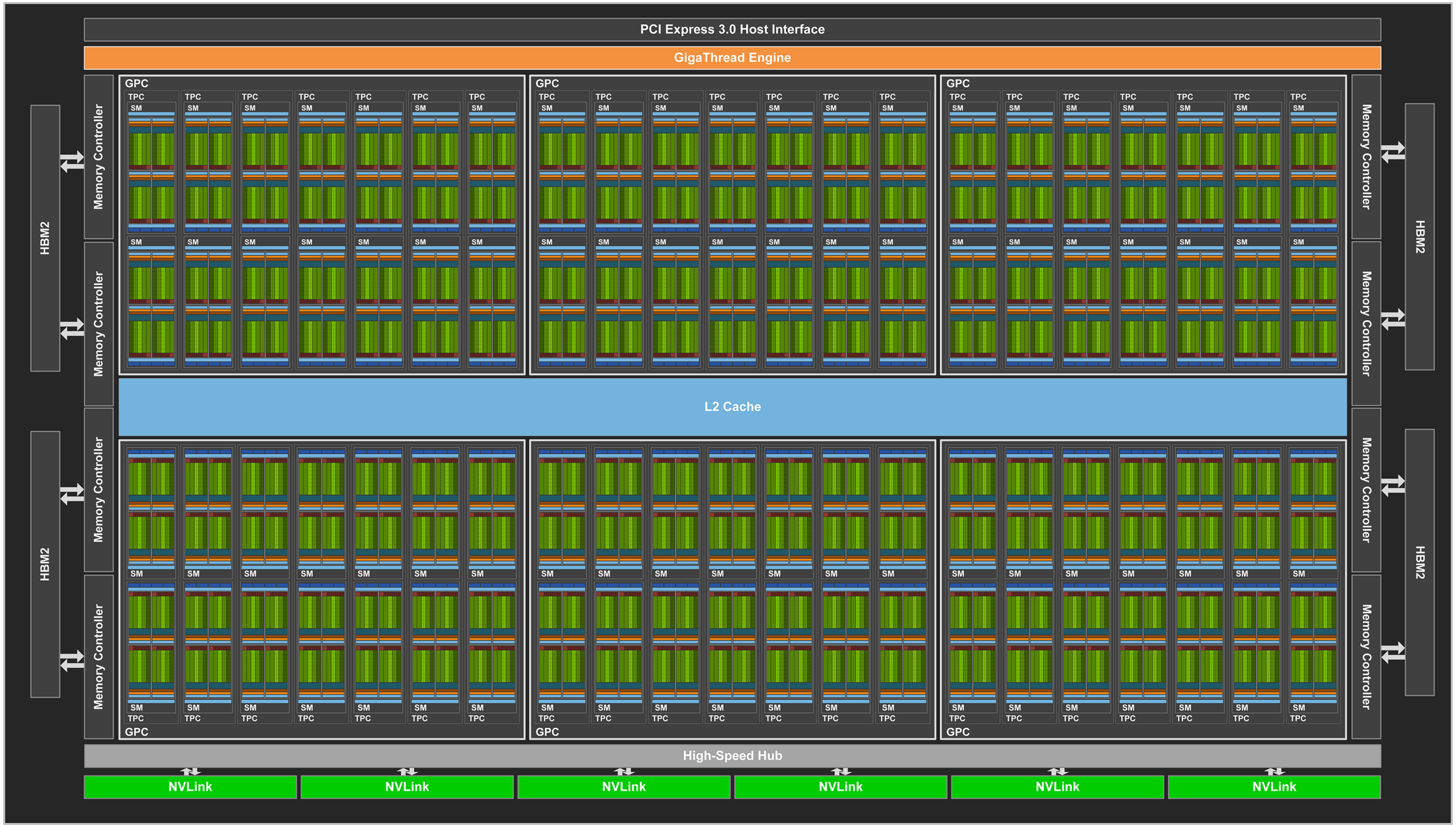
GPU vs CPU Architecture

* 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
GPC
TPC
SM
Memory Controller
HBM2
L2 Cache
High-Speed Hub
NVLink

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

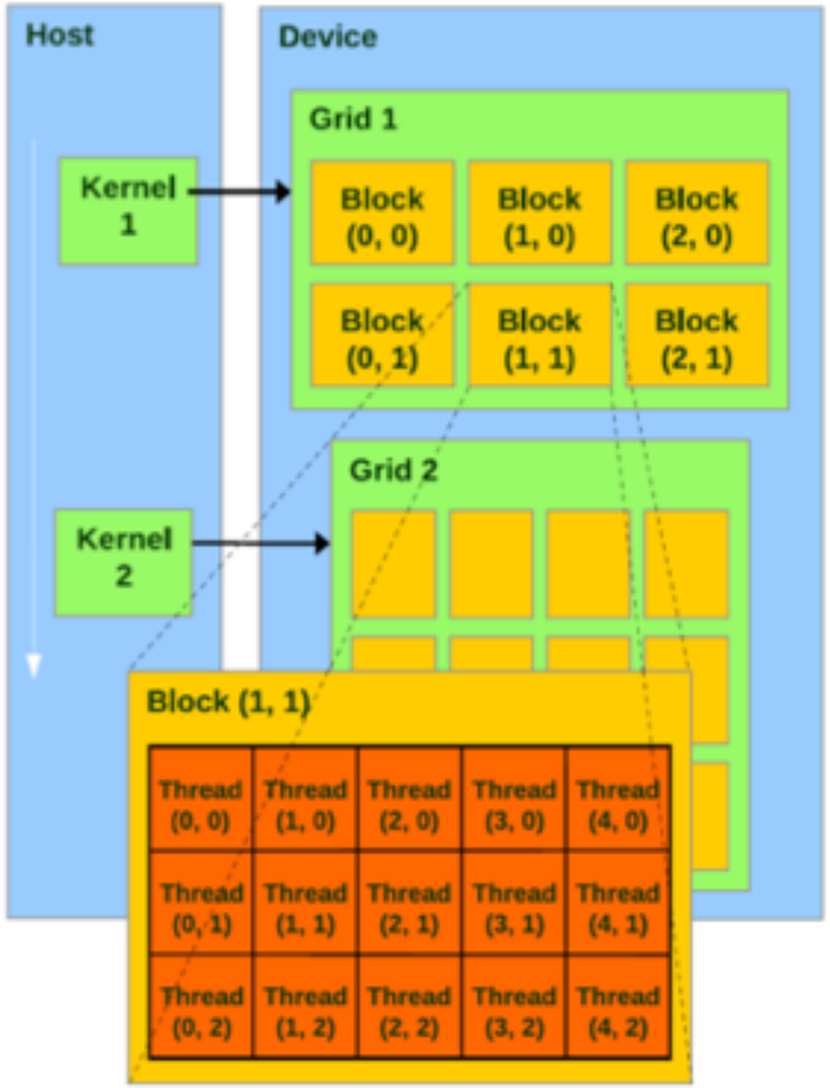
- Compute oriented drivers, language, and tools

Documentations:

CUDA_C_Programming_Guide.pdf
CUDA_C_Getting_Started.pdf
CUDA_C_Toolkit_Release.pdf

CUDA Programming Model

- 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
- Tow threads from two different blocks cannot cooperate
- Sequential code launches asynchronously GPU kernels



CUDA C/C++

Terminology:

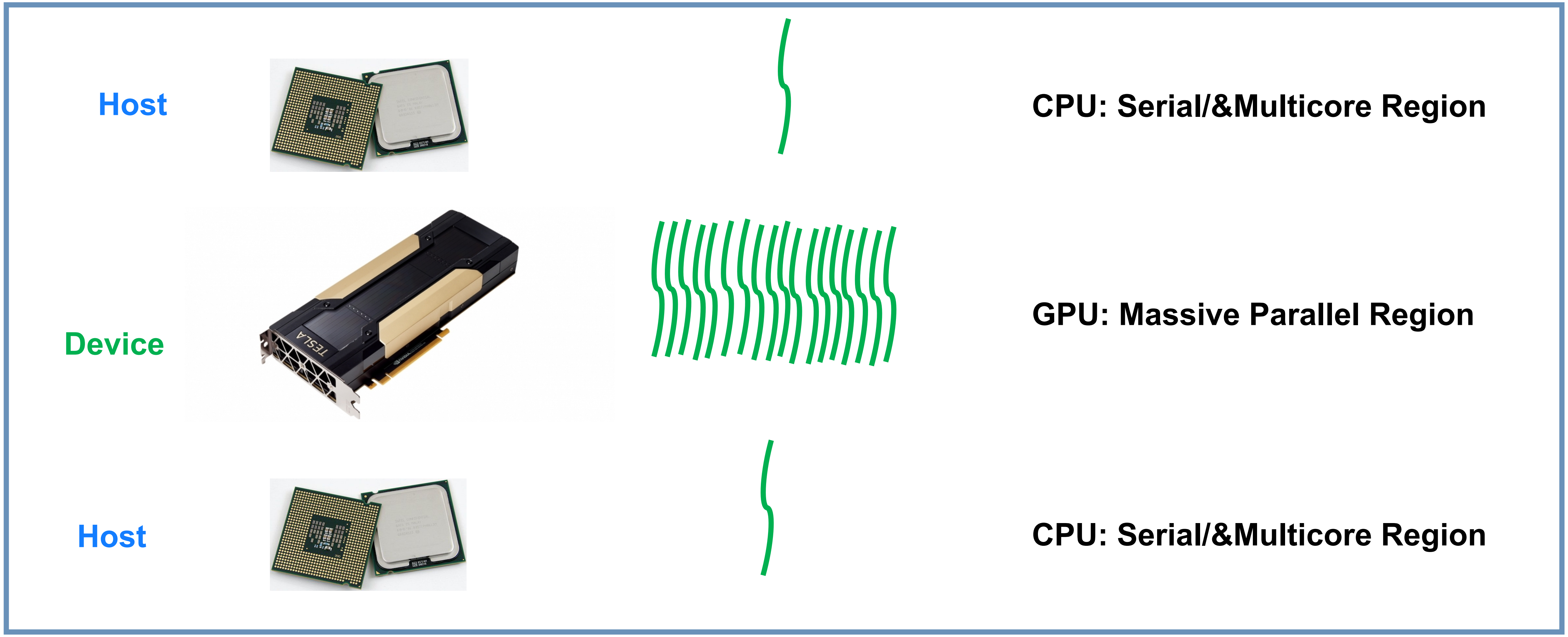
Host: The CPU and ist memory (host memory)

Host

Device: The GPU and ist memory (device memory)

Device

CUDA Devices and Threads Execution Model


Host
CPU: Serial/&Multicore Region
Device
TESLA
GPU: Massive Parallel Region
Host
CPU: Serial/&Multicore Region

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**.

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++

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

DEEP LEARNING INSTITUTE

Prerequisites

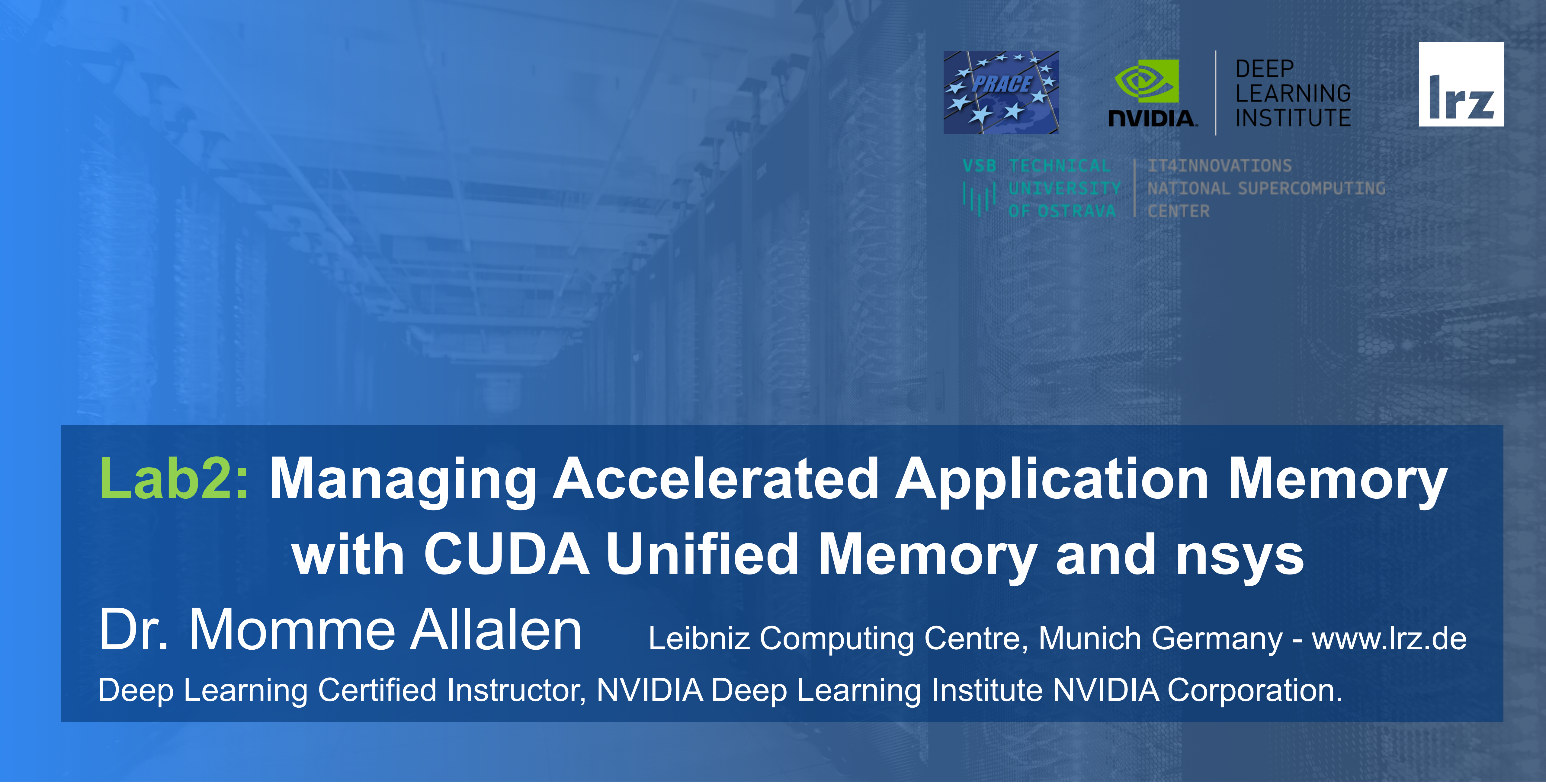
You should already be able to:

- 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**.



PRACE

NVIDIA DEEP LEARNING INSTITUTE

lrz

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.

OPTIMIZING APPLICATION PERFORMANCE WITH CUDA® PROFILING TOOLS

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

nvvp: NVIDIA visual profiler
nvprof: tool to understand and optimize the performance of your CUDA,
OpenACC or OpenMP applications, (features will be switched to the new **NSIGHT** tool)

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 PRODUCT FAMILY

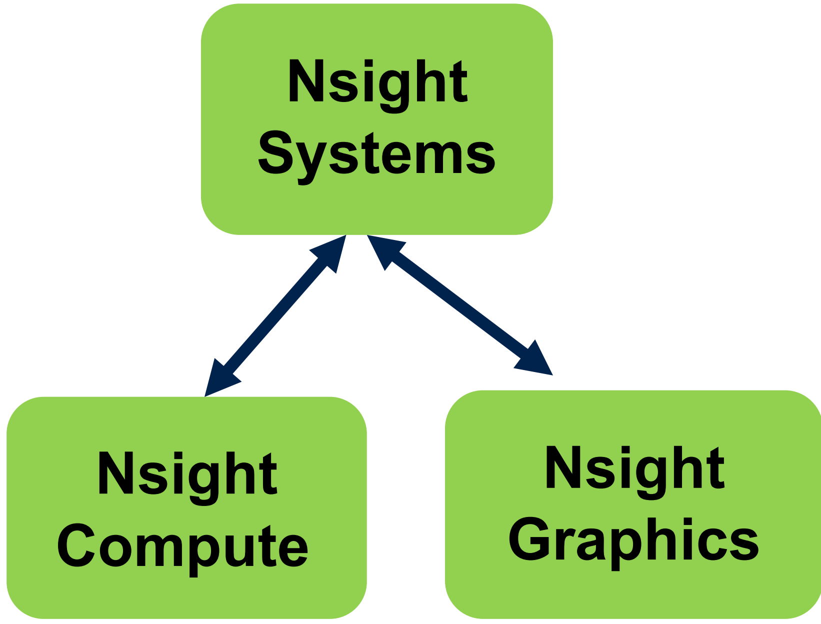
Standalone Performance Tools:

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

Overview

System-wide application algorithm tuning
- Multi-process tree support

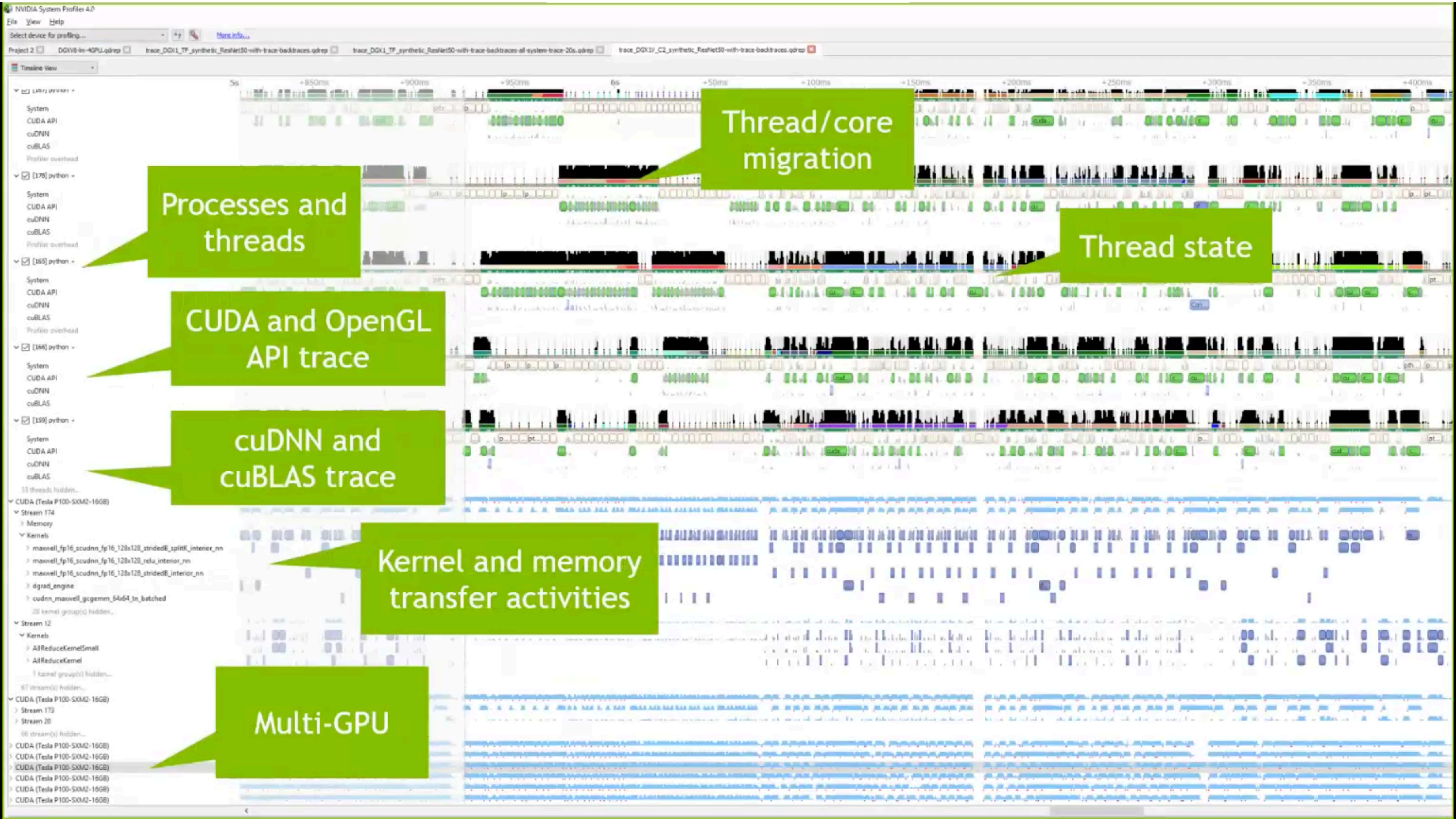
Locate optimization opportunities
- Visualize millions of events on a very fast GUI timeline
- Or gaps of unused CPU and GPU time

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**


NVIDIA System Profiler 4.0
File View Help
Select device for profiling...
More info...
Project 2
DGXV8-lin-4GPU.qdrep
trace_DGX1_TF_synthetic_ResNet50-with-trace-backtraces.qdrep
trace_DGX1_TF_synthetic_ResNet50-with-trace-backtraces-all-system-trace-20s.qdrep
trace_DGX1V_C2_synthetic_ResNet50-with-trace-backtraces.qdrep
Timeline View
5s
+850ms
+900ms
+950ms
6s
+50ms
+100ms
+150ms
+200ms
+250ms
+300ms
+350ms
+400ms
[178] python
[165] python
[166] python
[159] python
System
CUDA API
cuDNN
cuBLAS
Profiler overhead
53 threads hidden...
CUDA (Tesla P100-SXM2-16GB)
Stream 174
Memory
Kernels
maxwell_fp16_scudnn_fp16_128x128_stridedB_splitK_interior_nn
maxwell_fp16_scudnn_fp16_128x128_relu_interior_nn
maxwell_fp16_scudnn_fp16_128x128_stridedB_interior_nn
dgrad_engine
cudnn_maxwell_gcgemm_64x64_tn_batched
28 kernel group(s) hidden...
Stream 12
Kernels
AllReduceKernelSmall
AllReduceKernel
1 kernel group(s) hidden...
67 stream(s) hidden...
Stream 173
Stream 20
66 stream(s) hidden...
Thread/core migration
Processes and threads
Thread state
CUDA and OpenGL API trace
cuDNN and cuBLAS trace
Kernel and memory transfer activities
Multi-GPU

NSIGHT COMPUTE

CUDA Kernel profiler

Targeted metric sections for various performance aspects

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**

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.

Lab2: 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.

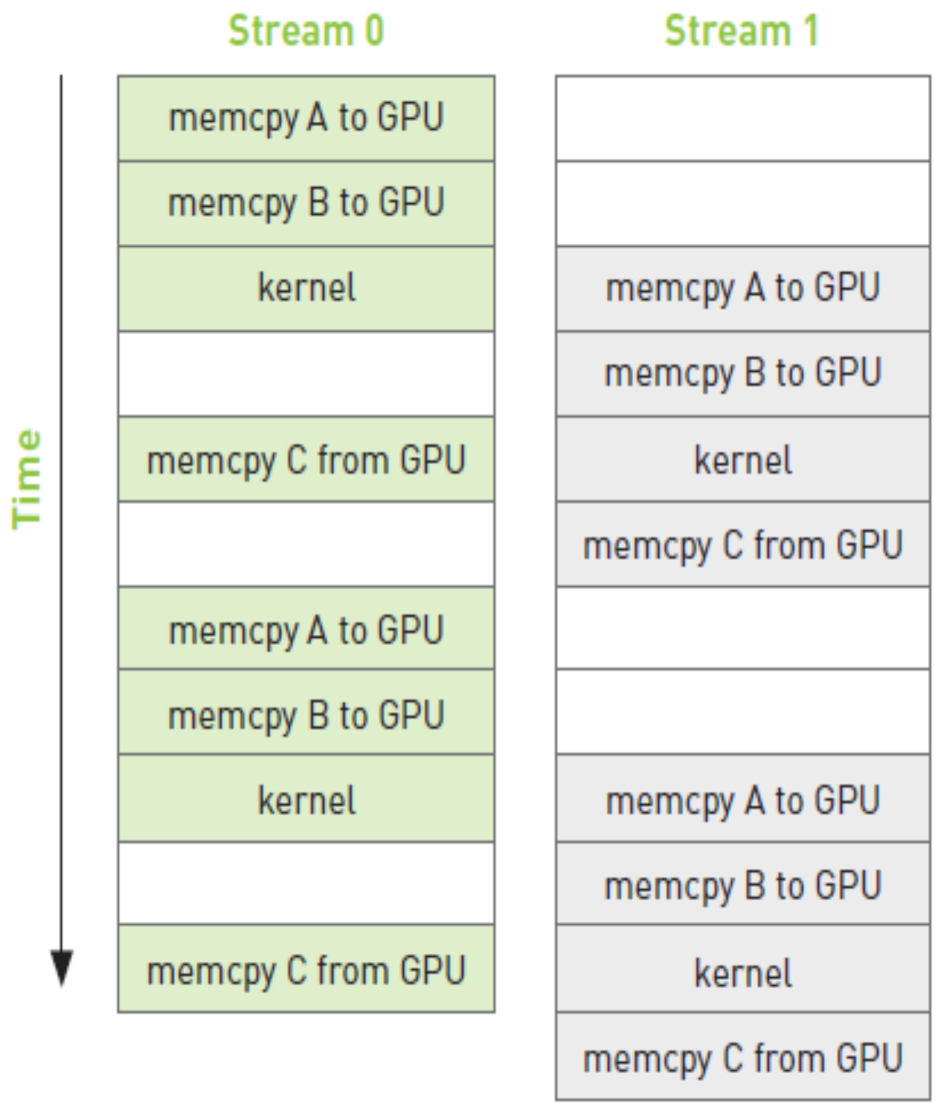
Objectives

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);
}
```

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

THANK YOU

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