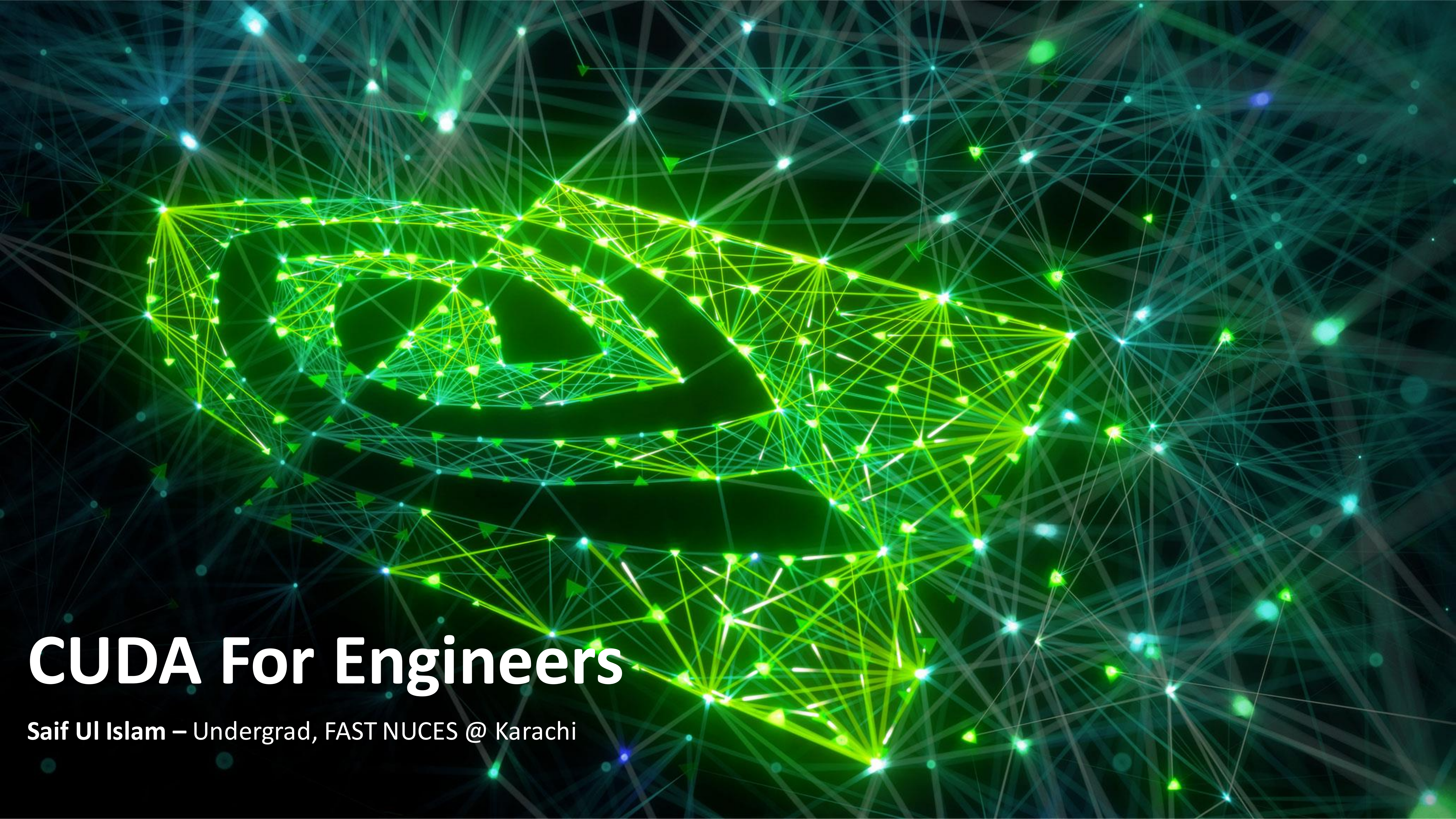


CUDA For Engineers

Saif Ul Islam – Undergrad, FAST NUCES @ Karachi

Introduction To Myself

I'm Saif!
I'm currently an undergrad from FAST NUCES, Karachi, majoring in Computer Science.

Interested in,

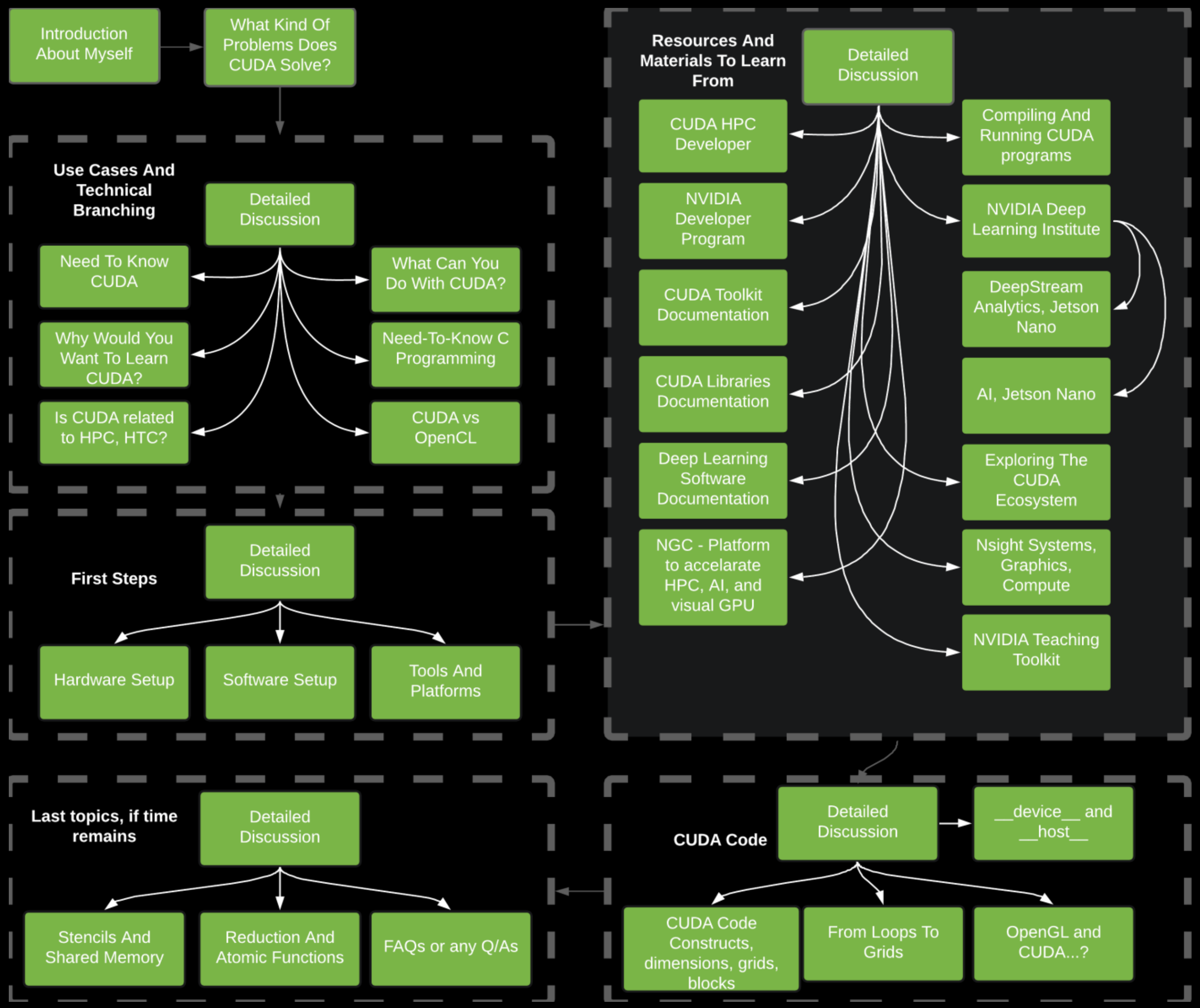
- Web Dev, Data Engineering, HPC, Software Engineering, Backend Architecture and Infrastructure Design

What's my experience with CUDA?

Not on many social media, but you can connect with me on,

- Github, https://github.com/rubix982
- LinkedIn, https://www.linkedin.com/in/saif-ul-islam-93786b187/
- Stack Overflow, https://stackoverflow.com/users/10576072/saif-ul-islam
- Mail, saifulislam84210@gmail.com

Table Of Contents



Introduction About Myself

What Kind Of Problems Does CUDA Solve?

Use Cases And Technical Branching

Detailed Discussion

Need To Know CUDA

Why Would You Want To Learn CUDA?

Is CUDA related to HPC, HTC?

What Can You Do With CUDA?

Need-To-Know C Programming

CUDA vs OpenCL

First Steps

Detailed Discussion

Hardware Setup

Software Setup

Tools And Platforms

Resources And Materials To Learn From

Detailed Discussion

CUDA HPC Developer

NVIDIA Developer Program

CUDA Toolkit Documentation

CUDA Libraries Documentation

Deep Learning Software Documentation

NGC - Platform to accelarate HPC, AI, and visual GPU

Compiling And Running CUDA programs

NVIDIA Deep Learning Institute

DeepStream Analytics, Jetson Nano

AI, Jetson Nano

Exploring The CUDA Ecosystem

Nsight Systems, Graphics, Compute

NVIDIA Teaching Toolkit

CUDA Code

Detailed Discussion

__device__ and __host__

CUDA Code Constructs, dimensions, grids, blocks

From Loops To Grids

OpenGL and CUDA...?

Last topics, if time remains

Detailed Discussion

Stencils And Shared Memory

Reduction And Atomic Functions

FAQs or any Q/As

TODAY’S STUDENT HIRING CHALLENGES

Facing Industry and Research

“..there are about 300,000 AI practitioners and researchers worldwide, but millions of roles available.”

Forbes, “*The AI Skills Crisis And How To Close The Gap*”, *June 2018*

“..finding people who can turn social-media clicks and user-posted photos into monetizable binary code is among the biggest challenges facing U.S. industry.”

Bloomberg, “*This is America’s Hottest Job*”, *May 2018*

“..it would be useful to have students explore the role of a computer engineer in an era of machine learning and intelligent systems.. Future curricula reports are currently under discussion. Two of those under early discussion are cybersecurity and data science.”

ACM and IEEE, “*Curriculum Guidelines for Undergraduate Degree Programs in CE*”, *December 2016*

What Kind Of Problems Does CUDA Solve?

- SIMD Architecture

- Massively Parallel

- Designed to run as an abstraction for different GPU hardware

- Easy to compile and run, provides compilers a target instruction level code, like Assembly

- Contains well maintained libraries, profiling tools, industrial grade technologies

Use Cases And Technical Branching

Use Cases

Why Would You Want To Learn CUDA?

- Growing HPC trends, coming 5-10 years
- Easy way to access, explore what HPC is about
- Needed engineers, required expertise with optimizing operations
- Growing hardware, software level scale for high portability, efficient code and databases.
- Edge networking devices, along with IoT trends
- Graphs galore!

Cloud Computing

1	Site Reliability Engineer
2	Platform Engineer
3	Cloud Engineer
3	DevOps Engineer
5	Cloud Consultant
6	DevOps Manager

Content Production

1	Social Media Assistant
2	Social Media Coordinator
3	Content Specialist
4	Content Producer
5	Content Writer
6	Creative Copywriter

Data and AI

1	Artificial Intelligence Specialist
2	Data Scientist
3	Data Engineer
4	Big Data Developer
5	Data Analyst
6	Analytics Specialist
7	Data Consultant
8	Insights Analyst
9	Business Intelligence Developer
10	Analytics Consultant

Engineering

1	Python Developer
2	Full Stack Engineer
2	Javascript Developer
4	Back End Developer
5	Frontend Engineer
5	Software Developer Dotnet
7	Development Specialist
8	Technology Analyst

Marketing

1	Growth Hacker
2	Growth Manager
3	Digital Marketing Specialist
4	Digital Specialist
5	Ecommerce Specialist
6	Commerce Manager
6	Head Of Digital
8	Digital Marketing Consultant
9	Digital Marketing Manager
10	Chief Marketing Officer

People and Culture

1	Information Technology Recruiter
2	Human Resources Partner
3	Talent Acquisition Specialist
4	Business Partner
5	Human Resources Business Partner

Product Development

1	Product Owner
2	Quality Assurance Tester
3	Agile Coach
4	Software Quality Assurance Engineer
5	Product Analyst
6	Quality Assurance Engineer
6	Scrum Master
8	Digital Product Manager
9	Delivery Lead

Sales

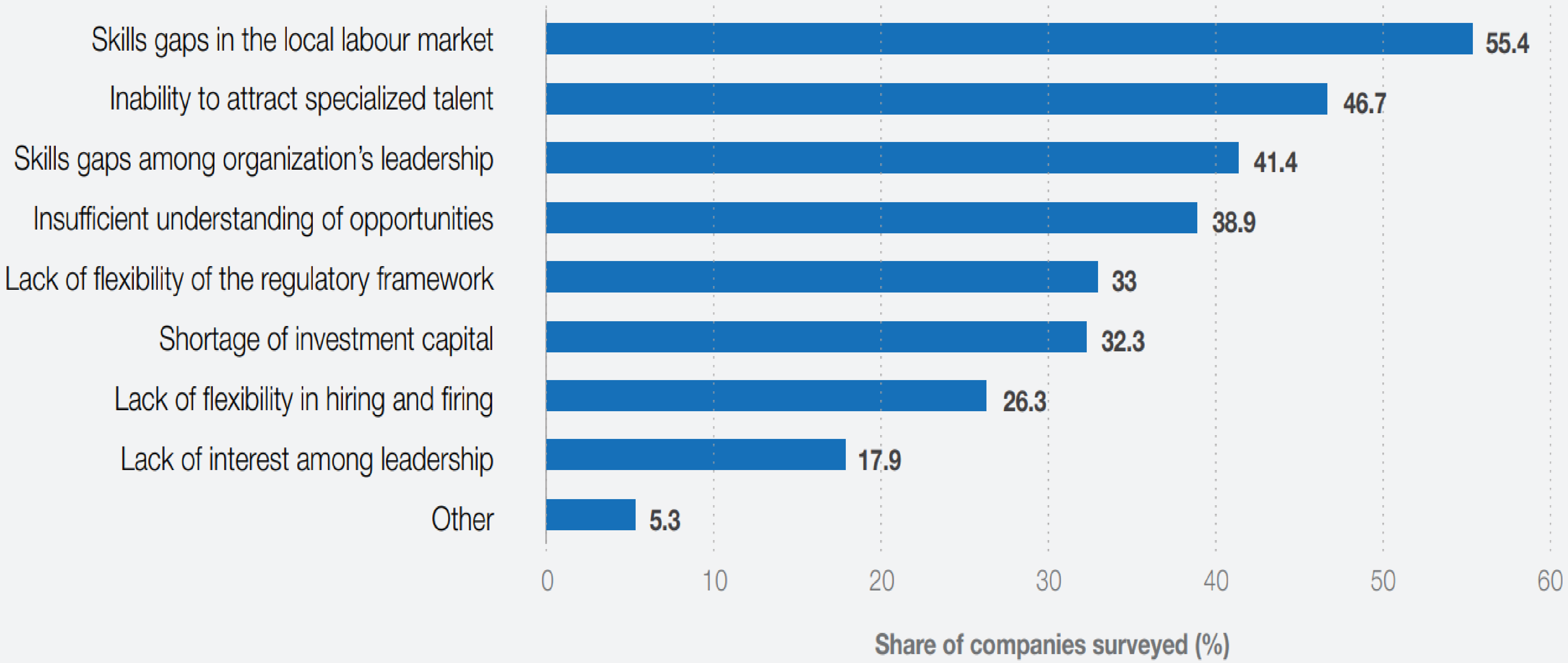
1	Customer Success Specialist
2	Sales Development Representative
3	Commercial Sales Representative
4	Business Development Representative
5	Customer Specialist
6	Partnerships Specialist
7	Chief Commercial Officer
8	Head Of Partnerships
9	Enterprise Account Executive
10	Business Development Specialist
11	Chief Strategy Officer
12	Head Of Business Development

(#) Rank ● Niche ● Mass

Source
LinkedIn Economic Graph.

FIGURE 26

Perceived barriers to the adoption of new technologies



Source

Future of Jobs Survey 2020, World Economic Forum.

FIGURE 18 Technologies likely to be adopted by 2025 (by share of companies surveyed)

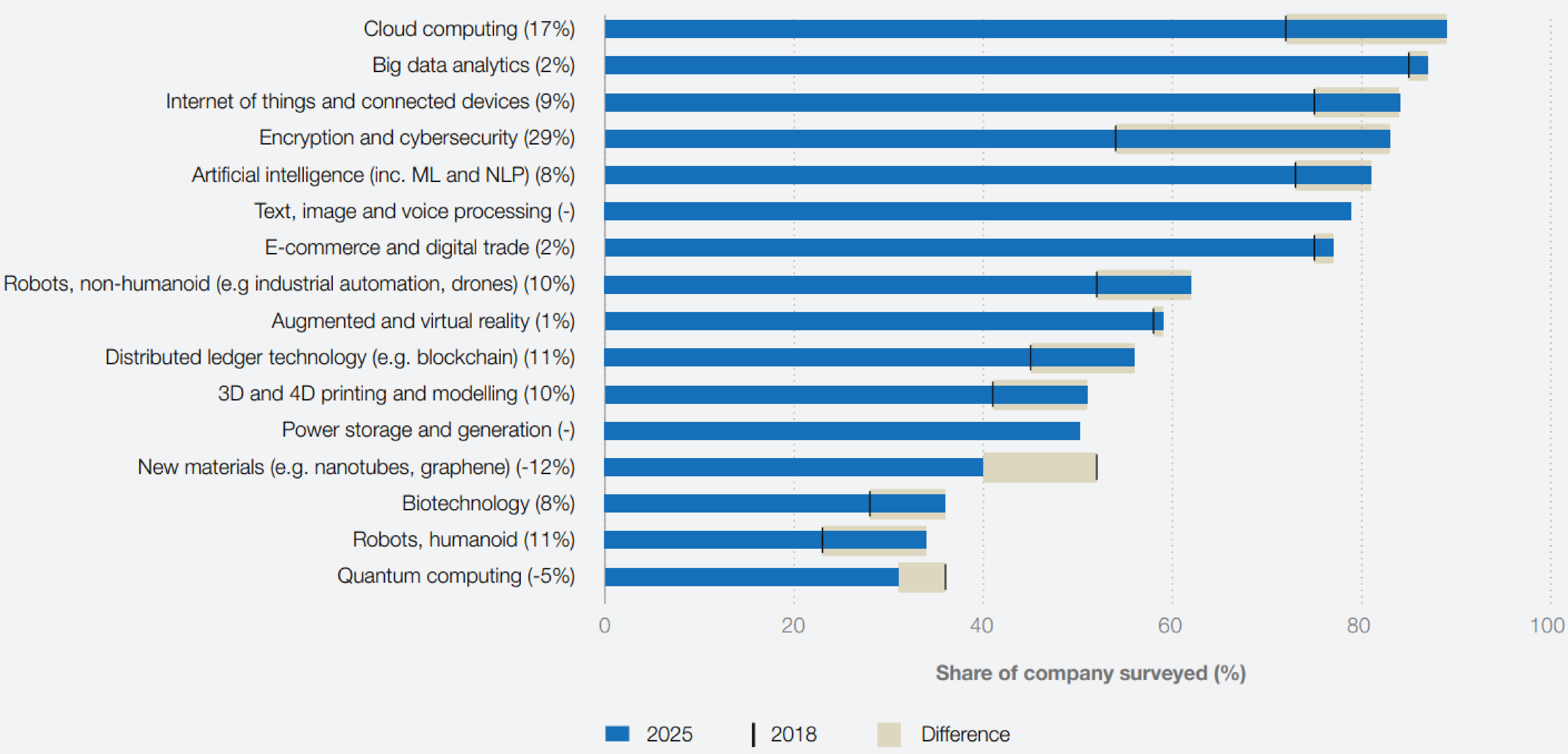


FIGURE 22

Top 20 job roles in increasing and decreasing demand across industries

↗ Increasing demand

1	Data Analysts and Scientists
2	AI and Machine Learning Specialists
3	Big Data Specialists
4	Digital Marketing and Strategy Specialists
5	Process Automation Specialists
6	Business Development Professionals
7	Digital Transformation Specialists
8	Information Security Analysts
9	Software and Applications Developers
10	Internet of Things Specialists
11	Project Managers
12	Business Services and Administration Managers
13	Database and Network Professionals
14	Robotics Engineers
15	Strategic Advisors
16	Management and Organization Analysts
17	FinTech Engineers
18	Mechanics and Machinery Repairers
19	Organizational Development Specialists
20	Risk Management Specialists

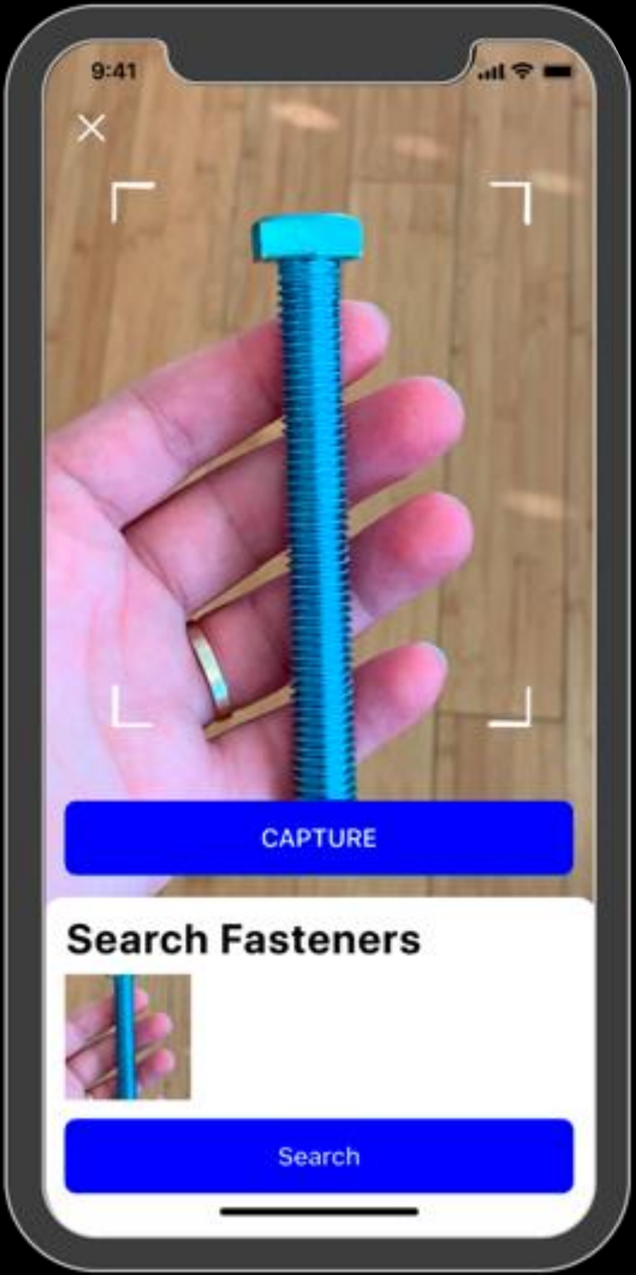
↘ Decreasing demand

1	Data Entry Clerks
2	Administrative and Executive Secretaries
3	Accounting, Bookkeeping and Payroll Clerks
4	Accountants and Auditors
5	Assembly and Factory Workers
6	Business Services and Administration Managers
7	Client Information and Customer Service Workers
8	General and Operations Managers
9	Mechanics and Machinery Repairers
10	Material-Recording and Stock-Keeping Clerks
11	Financial Analysts
12	Postal Service Clerks
13	Sales Rep., Wholesale and Manuf., Tech. and Sci.Products
14	Relationship Managers
15	Bank Tellers and Related Clerks
16	Door-To-Door Sales, News and Street Vendors
17	Electronics and Telecoms Installers and Repairers
18	Human Resources Specialists
19	Training and Development Specialists
20	Construction Laborers

Source

Future of Jobs Survey 2020, World Economic Forum.

Read the whole report here! "The Future of Jobs Report 2020 October 2020"



What Can You Do With CUDA?

- Easily massive parallel algorithms
- Research worthy platforms, for Fortran, Matlab, C++
- Deep learning, IoT as naive examples
- Brytlyt! - A complete SQL GPU solution for RDBMS
- Pilot.Ai - Retail Analytics, Jetpack | Jetson Tx2 | RTX 2080, video demo is here
- Interspectral - Education, Industry, Science & Comm | vGPUS | Education Demo
- Kognat - CUDA accelerated AI rotoscoping | Demo
- Slyce - Real time scan item | Jetpack
- Keyshot – 3D Graphics Generation | Marketing | VR | AR | NVIDIA OptiX Denoiser | Demo
- All thanks to NVIDIA! Dozens of more examples are here

The "Need To Know CUDA" Term

- Simply because CUDA is one of the easiest ways right now, best-supported, compatible, portable, and the most accessible platform for using the power of GPUs
- Getting a basic understanding of how CUDA works sets you on the way to create good to greater CUDA applications
- Reduces time to market for technology that simply need speedup. Often when an algorithm cannot be further optimized, it might be worth it to look at the GPU section for optimizing your applications
- Fundamental to basic APIs and libraries that already take care of the low level GPU part, compatible with modern code and design philosophies

CUDA And HPC, or HTC?

CUDA mostly deals with HPC, "Distributed And Cloud Computing … "

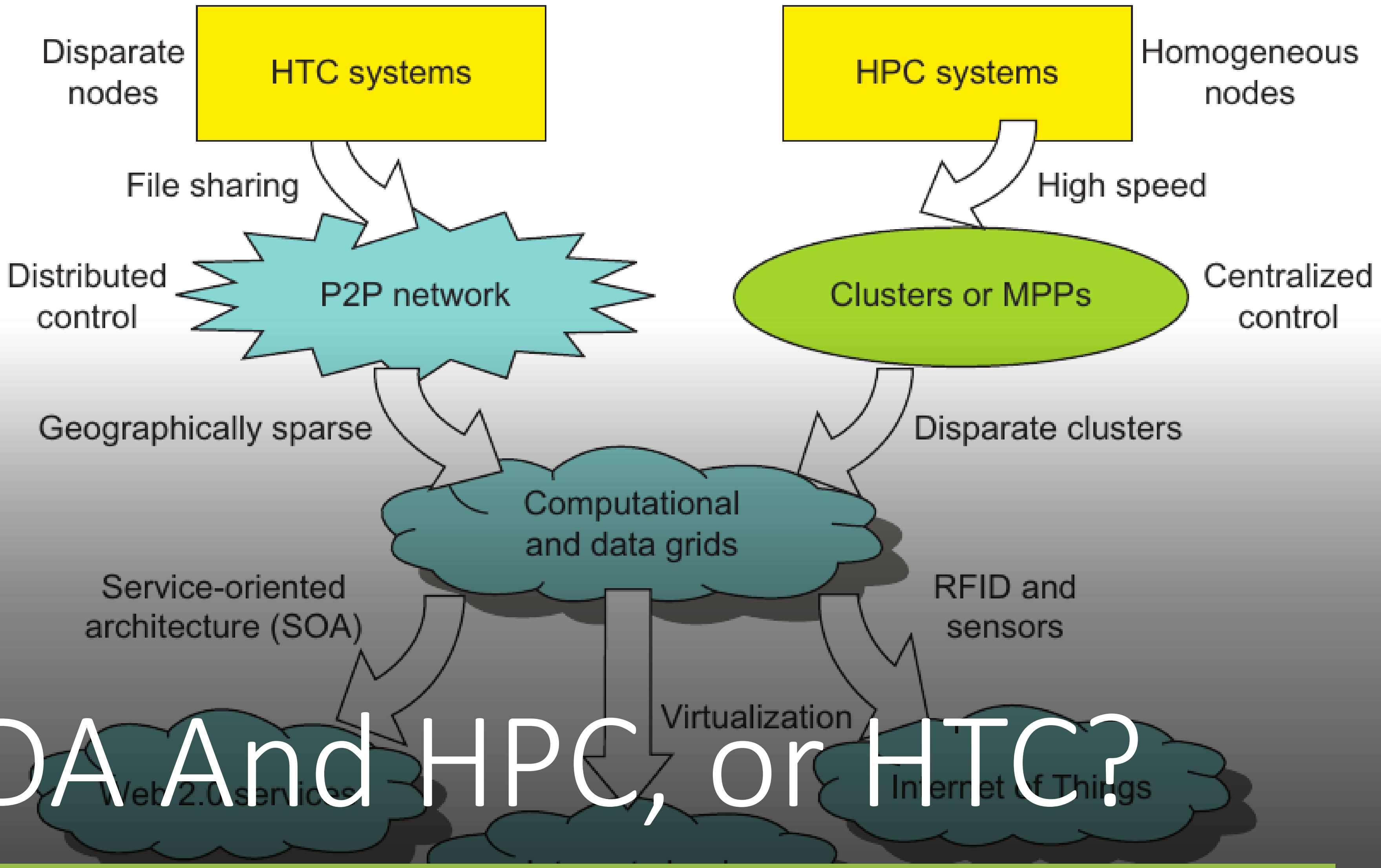


FIGURE 1.1

CUDA Vs OpenCL

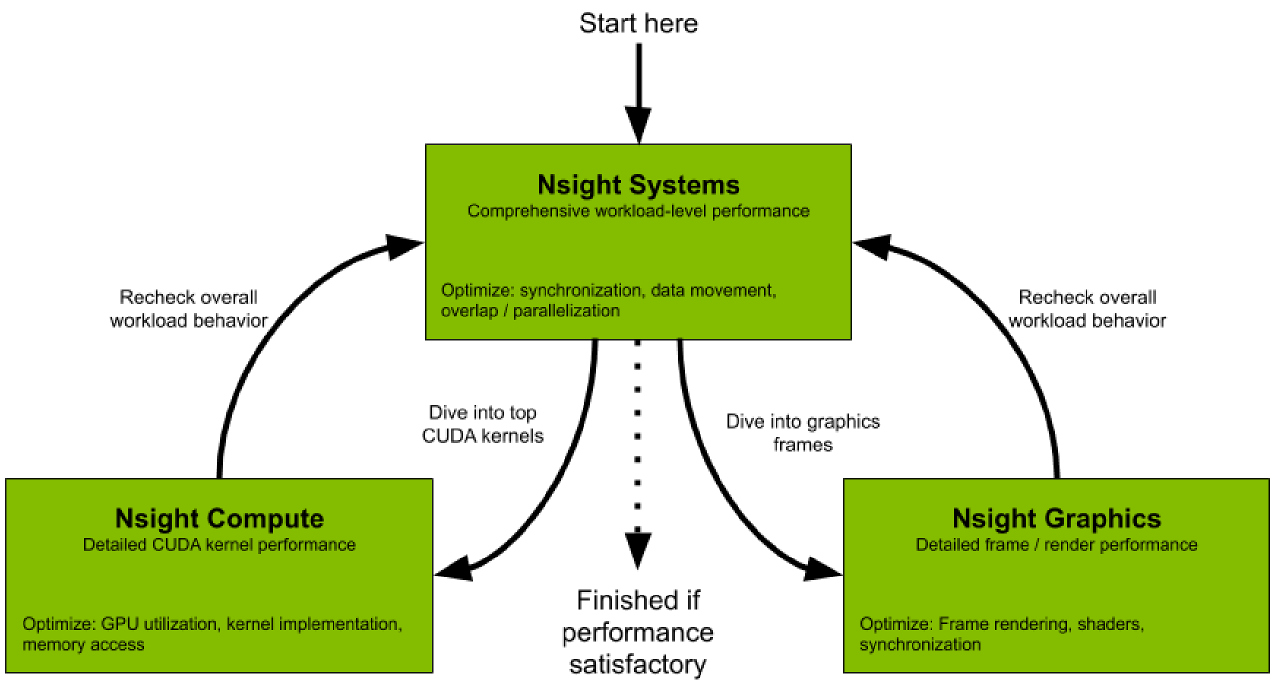
- NVIDIA GPUs use CUDA as an abstraction layer, for the compiler to target as for the respective assembly and architecture
- AMD primarily looks at using OpenCL, Open Computing Language, for its GPUs
- OpenCL by itself provides an abstraction layer between the hardware and software for GPUs.
- It's an 'interface', a standard, for providing low level instructions to CPUs, GPUs (of any vendor, even NVIDIA)
- CUDA is specifically designed for NVIDIA GPUs
- Writing OpenCL is considered more low level as compared to NVIDIA
- A tradeoff in development to promote cross compatibility

Need To Know C Programming

- To begin, you do not need to already be a Computer Scientist, or an Experienced Programmer (though it helps)
- You do not need to be very technical to get started with CUDA – a basic intuition and idea is enough
- Basics of C programming and a CUDA capable NVIDIA GPU are enough,
  - Variables, functions, loops, header file #includes, assignments, operators, compiling, running, and debugging code, control statements, arrays, memory allocation and deallocation
  - Preferred IDE is **Visual Studio**, if Windows
  - **Makefiles** are mostly enough when working in Linux
  - NVVP, **NVIDIA Visual Profiler**, a profiling tool that comes with **CUDA Toolkit**
  - **cuda-gdb,** a command line debugger for CUDA programs
  - General standard is to have source files be indicated by *.cu extension, header files be indicated by *.cuh extension

Break!
Currently summary for discussed topics.
Any Q/A?

First Steps



Start here

Nsight Systems
Comprehensive workload-level performance

Optimize: synchronization, data movement, overlap / parallelization

Recheck overall workload behavior

Recheck overall workload behavior

Dive into top CUDA kernels

Dive into graphics frames

Nsight Compute
Detailed CUDA kernel performance

Optimize: GPU utilization, kernel implementation, memory access

Finished if performance satisfactory

Nsight Graphics
Detailed frame / render performance

Optimize: Frame rendering, shaders, synchronization

Tools And Platforms

- Windows
- Linux
- MacOS

Hardware Setup

- Windows
- Linux
- MacOSX
- Determining Compute Capibility
- Detailed notes for hardware setup are here

Software Setup - Windows

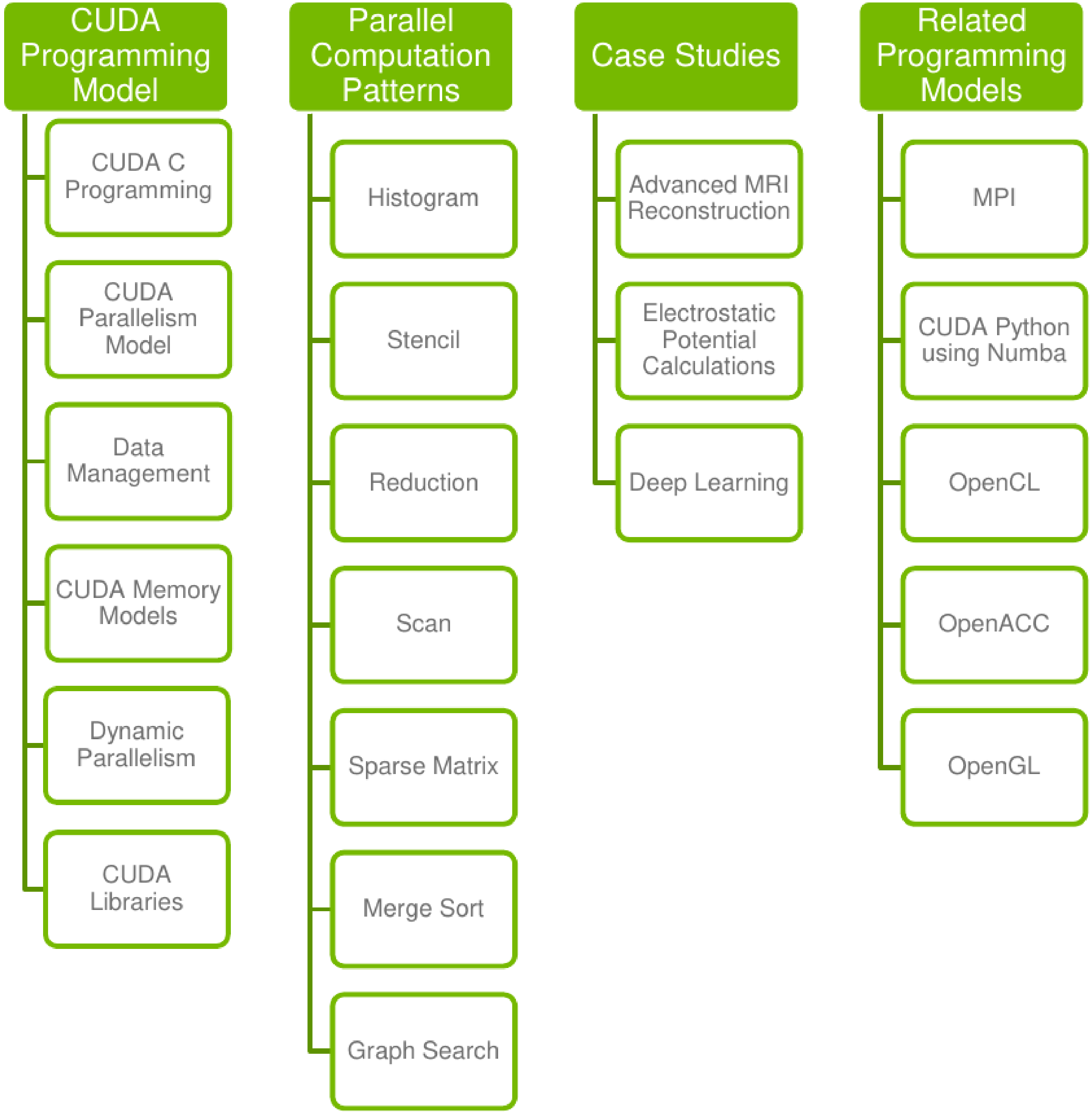
- Make sure to sign up for the "**CUDA Zone**" for a full list of CUDA-Dev related material
- Create a restore point (recommended) by going to 'Control Panel' -> Type 'Create a restore point' in the search box. Click on search result to open 'System Protection' tab in 'System Properties' tab. To create a restore point, click on 'Create...'. I maybe wrong, so please check this out
- Recommended IDE and setup are Visual Studio 2019, CUDA Toolkit 11.2, and Nsight Visual Studio Edition 2020.3.0. These versions are the latest at the time of writing. Please make sure to ensure compability between these. Downgrad as needed.
- The CUDA Toolkit should take care of 4 primary software components, **CUDA Toolkit** itself, **CUDA GPU Device Driver**, **NSight Visual Studio Edition**, and **CUDA Samples**.
- After installation, go ahead and open one of the *cuda samples* under *'All Programs',* choose any of the given examples, open up a *.vsproj* file, that should open up a project. Press the F7 key to start the build process. Executables are generated within the *bin* folder, then double click to execute it
- For the complete documentation, read the documentation article

Software Setup - Linux

- Make sure to sign up for the "CUDA Zone" for a full list of CUDA-Dev related material
- Make sure to verify that you indeed have a CUDA capable GPU from the 'Hardware Setup' section
- Check to make sure you have the **gcc** compiler, by running 'gcc –version'. If not, simply run 'sudo apt-get install gcc'
- Get the CUDA Toolkit 11.2. If on [Lu,Xu,U,Ku]buntu, double click on the download file, it should open the *Software Center*. When complete, run the following in a terminal, 'sudo apt-get update –y; sudo apt-get install cuda'.
- Open with a text editor, profile configuration at '~/.profile', append 'export PATH=/usr/local/cuda/bin:$PATH', save and exit. Now, similarly open '~/.bashrc', append 'export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_path', save and exit
- Restart the machine, via a terminal, run 'printenv | grep cuda'. If you get lines with a results that 'cuda' in them, the setup was successful. Also, you can now run the cuda compiler, called NVIDIA C Compiler', by running 'nvcc –version'
- To generate cuda samples, run 'cuda-install-samples-7.5.sh ~'. This generates the samples as '~/NVIDIA_CUDA-7.5_Samples'. Run 'ls ~/NVIDIA_CUDA-7.5_Samples'. To see those samples.
- To run a sample, visit any of the samples, run 'make' in the same directory as a makefile. This generates an executable binary, for example 'exectuble', then run './executable'.
- For the complete documentation, visit this documentation article

Software Setup - MacOS

- Make sure to sign up for the "**CUDA Zone**" for a full list of CUDA-Dev related material
- To be able to setup CUDA, you need to have a CUDA enabled GPU. MacOS mostly does not have 3rd party hardware updates, so you might need to consider getting a platform release that does have something like that.
- NVIDIA has officially cut off working and building for MacOS. The latest version for MacOS at the moment is 10.2, compared to the latest for Windows / Linux, which is 11.2. You can read the full article here, https://www.provideocoalition.com/officially-official-nvidia-drops-cuda-support-for-macos/
- To people who still are interested, check out https://www.nvidia.com/en-us/drivers/cuda/418_163/macosx-cuda-418_163-driver/, for until MacOS 10.13.



CUDA Programming Model

- CUDA C Programming
- CUDA Parallelism Model
- Data Management
- CUDA Memory Models
- Dynamic Parallelism
- CUDA Libraries

Parallel Computation Patterns

- Histogram
- Stencil
- Reduction
- Scan
- Sparse Matrix
- Merge Sort
- Graph Search

Case Studies

- Advanced MRI Reconstruction
- Electrostatic Potential Calculations
- Deep Learning

Related Programming Models

- MPI
- CUDA Python using Numba
- OpenCL
- OpenACC
- OpenGL

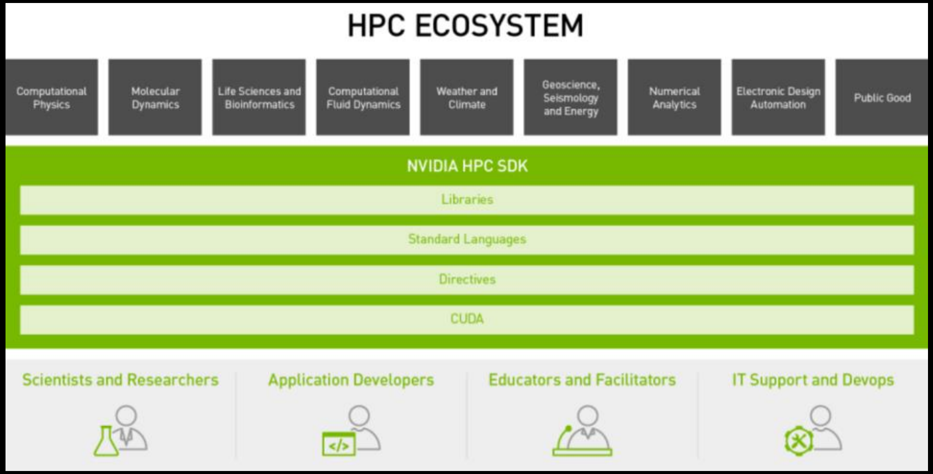
Break!
Currently summary for discussed topics.
Any Q/A?

Resources, And Materials To Learn From

The section dedicated to expanding and looking into the CUDA, NVIDIA ecosystem

CUDA HPC Developer

- The CUDA HPC Developer program consists of smaller sub programs,
  - "GPU Programming for HPC.", which includes libraries for math, image and video algorithms, deep learning, partner libraries etc, standardized C/C++/Fortran/MATLAB support, OpenACC development, as shown on the right. Visit here for more details.
  - "NVIDIA HPC SDK", contains extensive details for, also shown in the right slide, Scientists, Application Developers, Educators, and IT Support / DevOps. You can visit it here
  - "Exploring Technical Content", has on-demand sessions, podcasts, demos, research posters. Please visit this extensive search option for an in depth category and filtering option, or here if you would just like a high overview analysis



HPC ECOSYSTEM

Computational Physics

Molecular Dynamics

Life Sciences and Bioinformatics

Computational Fluid Dynamics

Weather and Climate

Geoscience, Seismology and Energy

Numerical Analytics

Electronic Design Automation

Public Good

NVIDIA HPC SDK

Libraries

Standard Languages

Directives

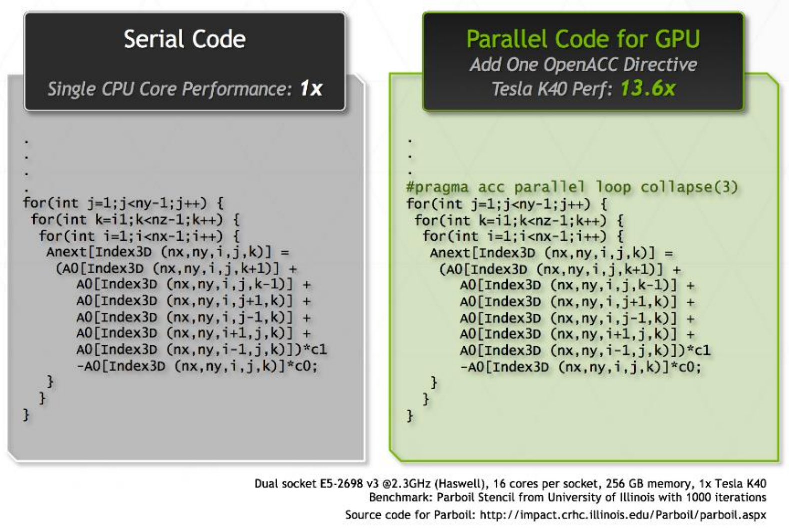
CUDA

Scientists and Researchers

Application Developers

Educators and Facilitators

IT Support and Devops



Serial Code

Single CPU Core Performance: ***1x***

```
.
.
.
.
for(int j=1;j<ny-1;j++) {
 for(int k=i1;k<nz-1;k++) {
  for(int i=1;i<nx-1;i++) {
   Anext[Index3D (nx,ny,i,j,k)] =
    (A0[Index3D (nx,ny,i,j,k+1)] +
       A0[Index3D (nx,ny,i,j,k-1)] +
       A0[Index3D (nx,ny,i,j+1,k)] +
       A0[Index3D (nx,ny,i,j-1,k)] +
       A0[Index3D (nx,ny,i+1,j,k)] +
       A0[Index3D (nx,ny,i-1,j,k)])*c1
       -A0[Index3D (nx,ny,i,j,k)]*c0;
   }
  }
}
```

Parallel Code for GPU

Add One OpenACC Directive

Tesla K40 Perf: ***13.6x***

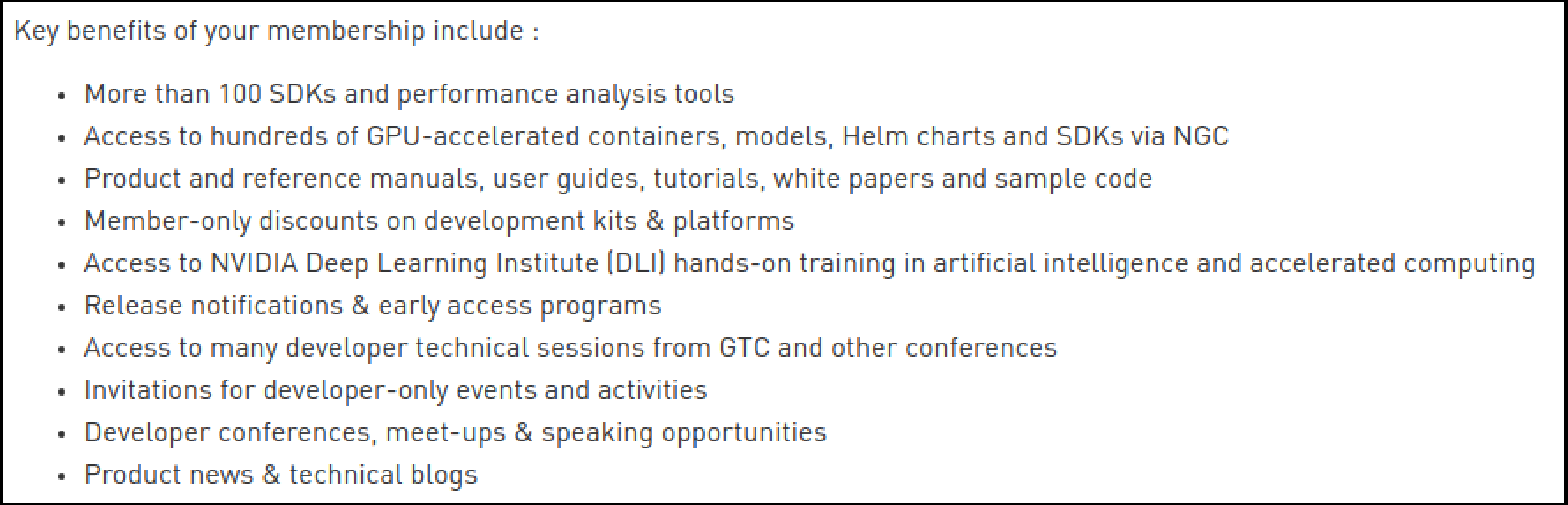
```
.
.
.
#pragma acc parallel loop collapse(3)
for(int j=1;j<ny-1;j++) {
 for(int k=i1;k<nz-1;k++) {
  for(int i=1;i<nx-1;i++) {
   Anext[Index3D (nx,ny,i,j,k)] =
    (A0[Index3D (nx,ny,i,j,k+1)] +
       A0[Index3D (nx,ny,i,j,k-1)] +
       A0[Index3D (nx,ny,i,j+1,k)] +
       A0[Index3D (nx,ny,i,j-1,k)] +
       A0[Index3D (nx,ny,i+1,j,k)] +
       A0[Index3D (nx,ny,i-1,j,k)])*c1
       -A0[Index3D (nx,ny,i,j,k)]*c0;
   }
  }
}
```

Dual socket E5-2698 v3 @2.3GHz (Haswell), 16 cores per socket, 256 GB memory, 1x Tesla K40
Benchmark: Parboil Stencil from University of Illinois with 1000 iterations

Source code for Parboil: http://impact.crhc.illinois.edu/Parboil/parboil.aspx

NVIDIA Developer Program

- Joining the developer program gives you tons of benefits right of the bat for your project
- Easy access to CUDA Libraries
- … And tons of material, for free
- You can check out [this link]

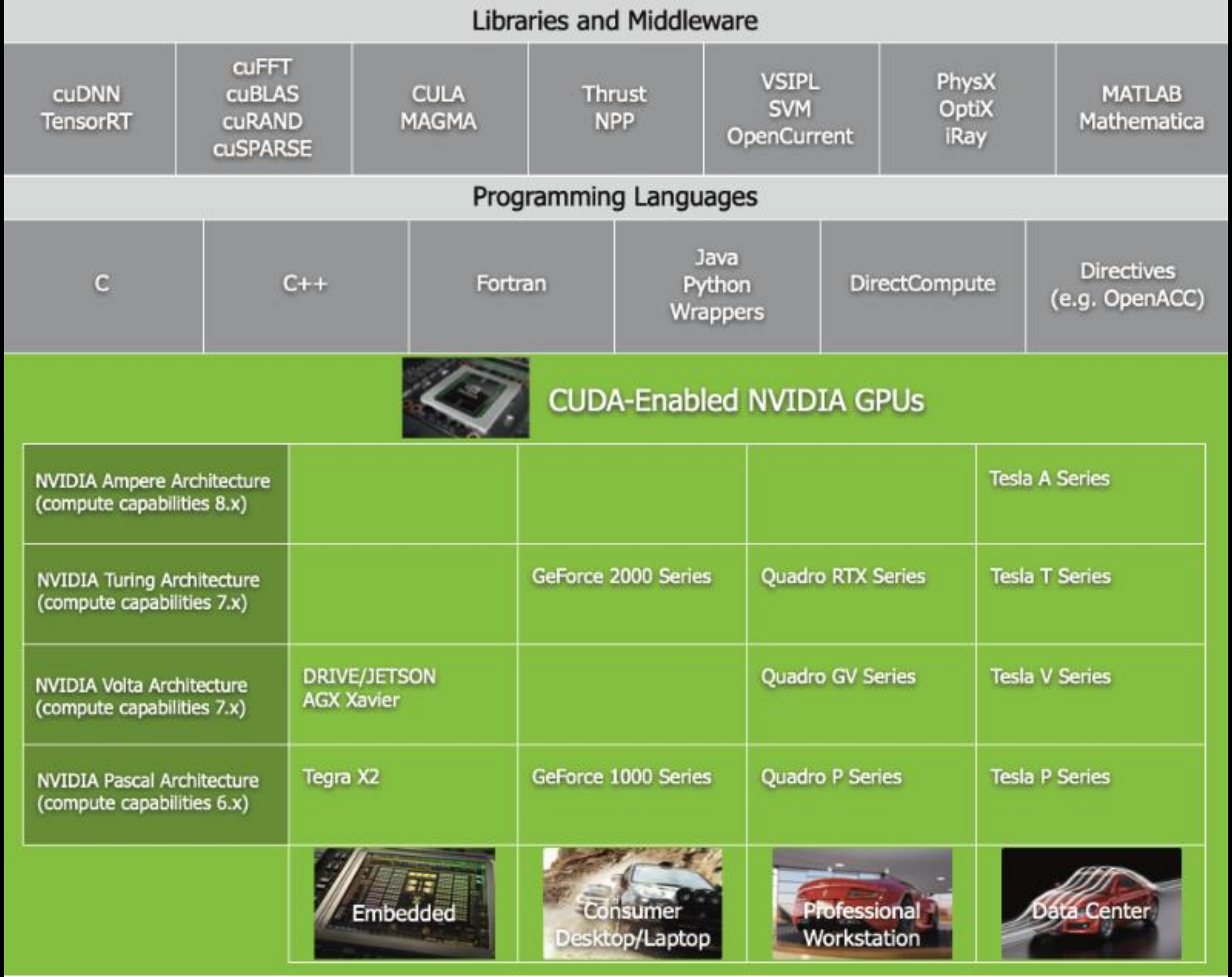


Key benefits of your membership include :

- More than 100 SDKs and performance analysis tools
- Access to hundreds of GPU-accelerated containers, models, Helm charts and SDKs via NGC
- Product and reference manuals, user guides, tutorials, white papers and sample code
- Member-only discounts on development kits & platforms
- Access to NVIDIA Deep Learning Institute (DLI) hands-on training in artificial intelligence and accelerated computing
- Release notifications & early access programs
- Access to many developer technical sessions from GTC and other conferences
- Invitations for developer-only events and activities
- Developer conferences, meet-ups & speaking opportunities
- Product news & technical blogs

CUDA Toolkit Documentation

- As with everything, there is a documentation and guide for everything CUDA related and its libraries.
- It's pretty huge, and is meant as a developer guide
- Use it only when you need to



Libraries and Middleware

cuDNN TensorRT	cuFFT cuBLAS cuRAND cuSPARSE	CULA MAGMA	Thrust NPP	VSIPL SVM OpenCurrent	PhysX OptiX iRay	MATLAB Mathematica

Programming Languages

C	C++	Fortran	Java Python Wrappers	DirectCompute	Directives (e.g. OpenACC)

CUDA-Enabled NVIDIA GPUs

NVIDIA Ampere Architecture (compute capabilities 8.x)				Tesla A Series
NVIDIA Turing Architecture (compute capabilities 7.x)		GeForce 2000 Series	Quadro RTX Series	Tesla T Series
NVIDIA Volta Architecture (compute capabilities 7.x)	DRIVE/JETSON AGX Xavier		Quadro GV Series	Tesla V Series
NVIDIA Pascal Architecture (compute capabilities 6.x)	Tegra X2	GeForce 1000 Series	Quadro P Series	Tesla P Series
	Embedded	Consumer Desktop/Laptop	Professional Workstation	Data Center

CUDA Libraries Documentation

- Linear Algebra, Differential Equations, Sparse and Dense Matrixes, Matrix Factorizations, Random Number Generators, , Video Filtering and Segmentation, and so much more
- There's cuBLAS, cuSOLVER, NPP (NVIDIA Performance Primitives), cuTENSOR, nvJPEG, etc
- You can check out the complete list here

NGC – Platform To Accelerate HPC, AI, and Visual GPU

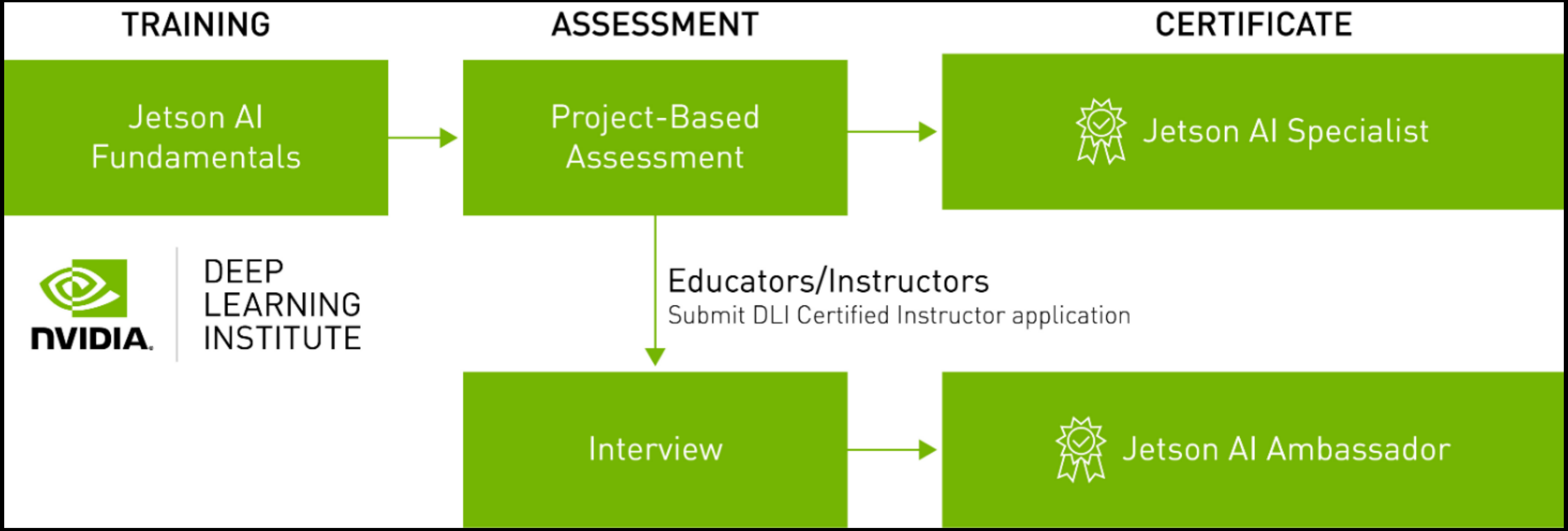
- Serves as a main repository for whatever content NVIDIA may release related to IT Support / DevOps or for Application Developers.
- Sorts the tools available in NVIDIA according to 'Industry', 'AI Application', and 'Technology'
- There are categories to explore in, 'Collections', 'Containers', 'Helm Charts', 'Models', 'Resources'
- Main focus on GPU optimized developer solutions, for DL, ML, and HPC, and their rapid deployment
- If you have any questions or run into any problems, there is always the NVIDIA Forums

NVIDIA Deep Learning Institute – 1/2

- Mainly focused on the training side for people who might be interested in either,
  - Self-paced courses
    - Most courses are paid, except two very popular programs "Getting Started With AI On Jetson Nano", "Getting Started With DeepStream For Video Analytics On Jetson Nano" are free and available as self-paced.
    - Check out the course catalog here
  - Instructor-led workshops
    - Not completely avaible as free and viable options
    - Need to contact NVIDIA, who ask a certified Deep Learning Instructor to lead the workshop. If you're still interested, check out this link
  - Educator Programs and toolkits. Has resources for
    - Teaching Kits - Deep Learning, Accelerated Computing, and Robotics. Viable option for educators willing to teach these materials in their classrooms. Link is this.
    - University Ambassador Program - members receive course materials, instructor assessments, certification interviews to help you become a Deep Learning Instructor (DLI) certified, and deliver DLI courses. Find out more here
    - Jetson AI Ambassador Certification - Two main programs at the moment, Jetson AI Specialist, and Jetson AI Ambassador. Learn more here

NVIDIA Deep Learning Institute – 2/2

- Most of the people here wil most probably join a hardware or software oriented company. If you ever feel like your organization needs to incorporate enterprise level distributed high performance computing, NVIDIA has programs meant to help you configure your products with team oriented training. Visit here for more details



TRAINING

ASSESSMENT

CERTIFICATE

Jetson AI Fundamentals

Project-Based Assessment

Jetson AI Specialist

NVIDIA. DEEP LEARNING INSTITUTE

Educators/Instructors

Submit DLI Certified Instructor application

Interview

Jetson AI Ambassador

Deep Learning Software Documentation

- There are specialized CUDA libraries related to just Deep Learning. The list deals with deep convolutional neural networks, tenors, the NVIDIA GPU training system, cloud interfacing solutions,
  - CuDNN - cuDNN is a "GPU-accelerated library of primitives for deep neural networks"
  - TensorRT - a "C++ library that facilitates high-performance inference on NVIDIA GPUs
  - NCCL - NCCL is a library "focused on accelerating collective communication primitives
  - DIGITS - DIGITS deals with training, monitoring, designing, managing, visualizing Deep Learning Tasks
  - NeMo - Art and speech related library
  - NVIDIA GPU Cloud – Cloud platform by NVIDIA for GPU-accelerated computing

Exploring The CUDA Ecosystem

- Code samples are available at the end of the presentation
- Exciting series of GTC keynotes. From Graphics Simulation, Retail, AI and Deep Learning, to Healthcare and HPC. See a complete catalogue here
- ParallelForAll, a series of blogs related to NVIDIA CUDA and GPU computing. Start from here
- Online courses
  - "Udacity CS344: Intro To Parallel Programming", as the description explains it, "*This class is for developers, scientists, engineers, researchers, and students who want to learn about GPU programming, algorithms, and optimization techniques*". It's free and available here
  - "Heterogenous Parallel Programming", deals with C/C++/OpenACC, contains introduction to both CUDA and OpenCL. Also free and available at this link
  - "Programming Massively Parallel Processors With CUDA", a set of recordings of a lecture at Stanford, offered in the Spring of 2010, by Dr. Jared Hoberock, and Dr. Nathan Bell. Available as free recordings on iTunes, here. A book with the same name can be found here.

NSight Systems, Graphics, Compute

- Some basic tools, as discussed before.
- NSight Systems - Performance Analysis Tools to make sure the workstation or servers are running at full capacity and are not at an 'idle' state
- NSight Graphics – Memory utilization, shader cycles, ray tracing, pixel history, for Vulkan and OpenGL, OpenVR, and Oculus SDK
- NSight Compute - Kernel profile, tasking debugging, graph analysis, memory workload

Break!
Currently summary for discussed topics.
Any Q/A?

CUDA Code Session!

CUDA Code Constructs, Dimensions, Grids, Blocks

- Since we are dealing with massively parallel organizational of computational resources, there are some problems we need to solve for first,
  - How does a particular computing unit know which subtask to perform?
  - How do a large number of computing units get access to the instructions and data they need without causing a major communications jam?
  - How should the memory and the computing units be distributed?
  - Are we going to use a shared or a distributed memory access system?

CUDA Code Constructs, Dimensions, Grids, Blocks 1/3

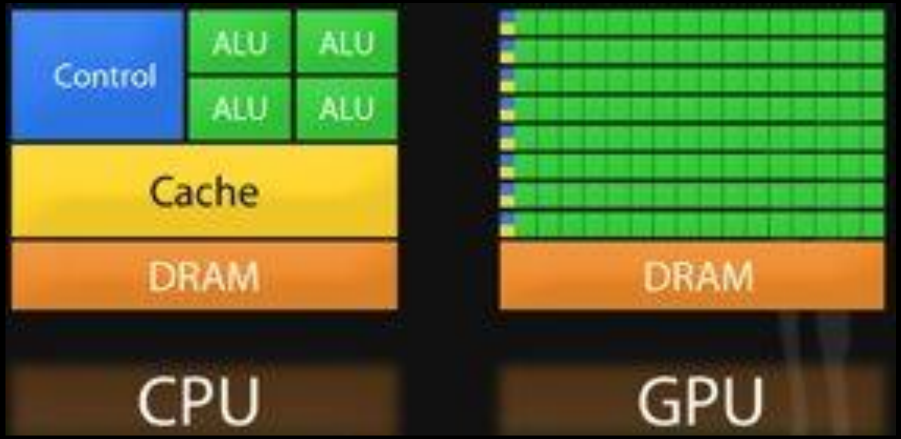
- As discussed before, CUDA has the **Single Instruction Multiple Thread** (**SIMT**), architecture. And so, the computational units are divided into hundreds of individual units called cores
- The performance of these cores is measured in Arithmetic Logic Units (ALUs), and Floating-Point Unit (FPUs) operations.
- Cores are then collected into 'groups', called Streaming Multiprocessors (SMs)
- Moving onto the code, once we distribute the subtasks, we assign each subtask to a thread
- Each group of threads is organized into a block
- Blocks are then grouped, and divided on the basis of warps
- The size of warps matches the number of cores on an SM
- Each warp is assigned to a SM
- This SIMT approach is scalable, just add more SMs!

CUDA Code Constructs, Dimensions, Grids, Blocks 2/3

- Hardware aspects,
  - The time period a core waits to be assigned a subtask is called latency
  - SMs have a shared control unit
  - You can't minimize latency sometimes – let's hide it!
  - When data isn't available for a certain task, the data that is instead already available is provided, even though it might be initially for a different core!
  - Focus is overall, not individually on cores

CUDA Code Constructs, Dimensions, Grids, Blocks 3/3

- Software aspects,
  - A special kind of function called a kernel
  - Kernels launch a large collection of computational threads organized into groups, to be assigned to SMs
  - Now that we have a kernel, we 'launch' it, to 'create' a computational 'grid', composed of 'blocks' of 'threads' (also called thread blocks)
  - To address each thread, each thread if provided a built-in index



"Distributed And Cloud Computing From Parallel Processing To The Internet Of Things" Kai Hwang, et al

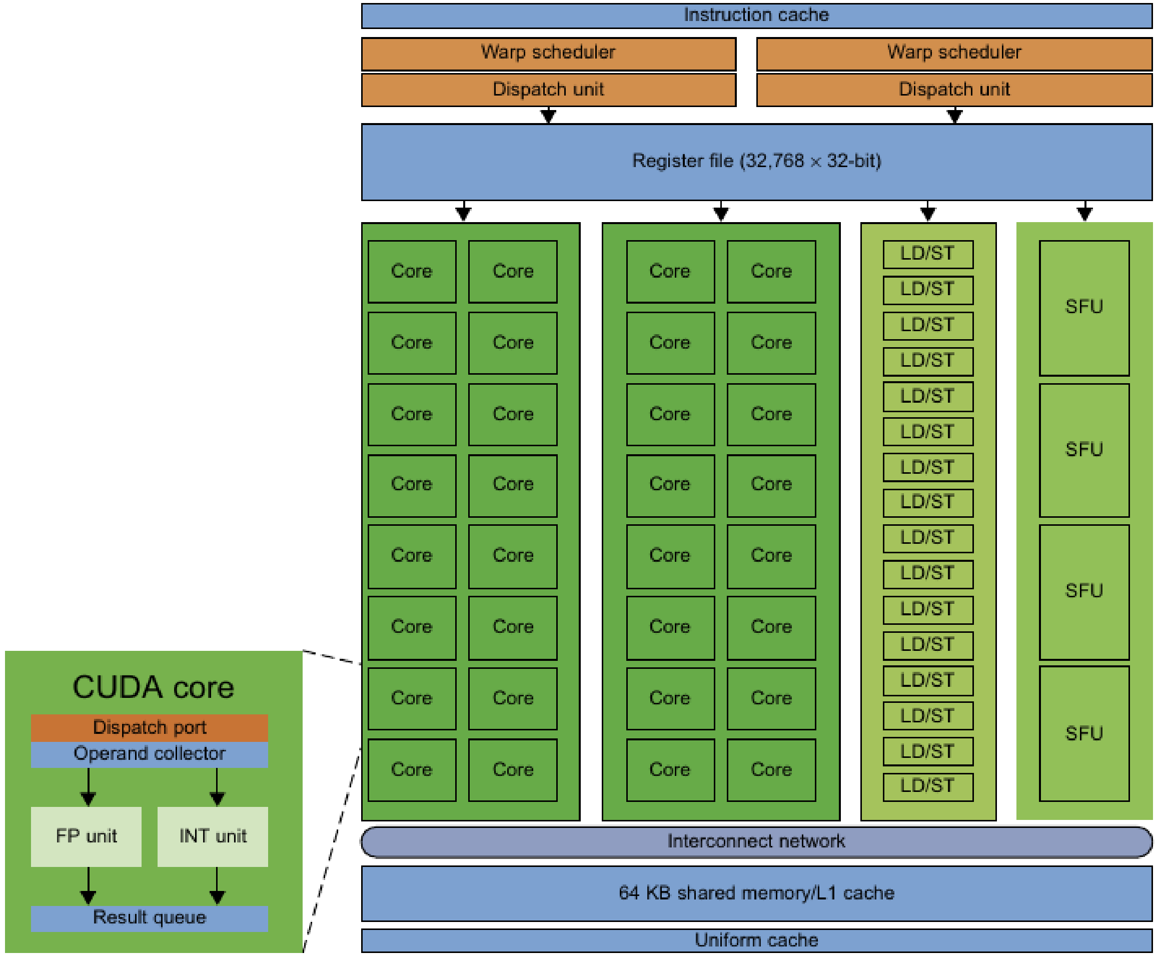


FIGURE 1.8

NVIDIA Fermi GPU built with 16 streaming multiprocessors (SMs) of 32 CUDA cores each; only one SM Is shown. More details can be found also in [49].

(Courtesy of NVIDIA, 2009 [36] 2011)

CUDA Code Constructs, Dimensions, Grids, Blocks

- Notebook for the code can be found here
- The <<< >>> (chevron) syntax
- The syntax before the <<< indicates the function name, in this case, that is 'distanceKernel'
- 1st argument within <<< >>> indicates the number of grids we want to create, in this case, we make N/TPB grids, where N is total number of cores available, and TPB is the 'Threads Per Block'
- 2nd argument indicates the actual number of threads (TPB) passed to a grid to be assigned to a SM

__device__ vs __host__ 1/2

- Host – refers to the CPU that instructs the GPU what the process
- Device – refers to the GPU(s) attached to the CPU that take the instruction(s) and the data
- Some basic qualifiers to indicate where the code will run
  - __global__ – this is the qualifier for kernels. These are called from the host and executed and on the device
  - __device__ – Functions are called from the device and execute on the device. A function that is called from a kernel needs the __device__ qualifier
  - Prepending __host__ __device__ causes the system to compile separate host and device versions of the function

__device__ vs __host__ 2/2

- Kernels provide basic variables and a function declaration
  - Kernel functions do not have return types, so we get '__global__ void aKernel(typedArgs)'
  - gridDim – specifies the number of blocks in the grid
  - blockDim – specifies the number of threads in each block
  - blockIdx – to index the blocks in a grid
  - threadIdx – to index the thread within the block
- Kernels do not have access to CPU memory
  - Memory is allocated separately for the device and the host, cudaMalloc(), cudaMemcpy(), cudaFree(), etc

From Loops To Grids

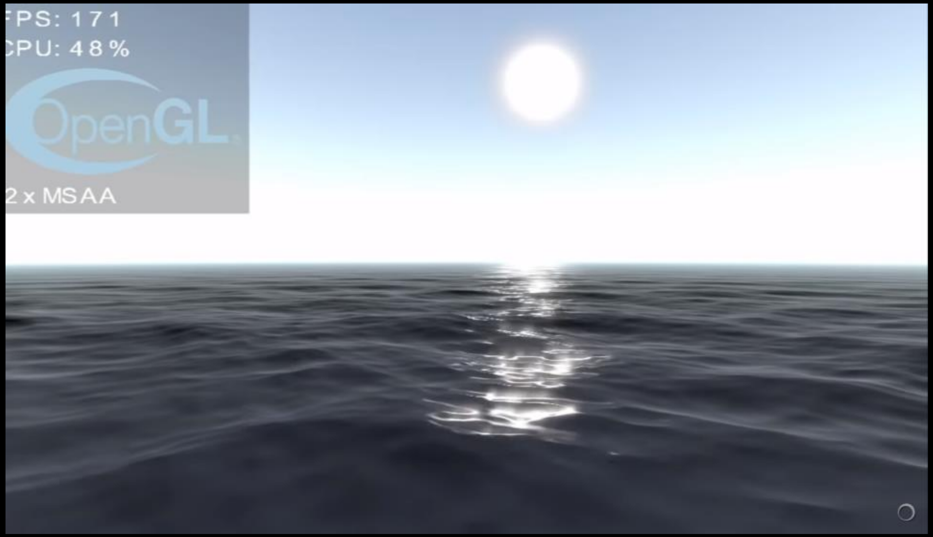
- The same code emulates this example. The link again to the code can be found here
- Contrasting change in where the memory is allocated to remove bottlenecks
- Allocating on the 'device' instead of the 'host'
- Typically, we have a plan to avoid possible overheads of memory operations via the following procedure
    - Copy your input data to the device *once* and leave it there
    - Launch a kernel that does a significant amount of work. The benefits of massive parallelism are significant compared to the cost of memory transfers
    - Copy the results back to the host only *once*
- This is *too much bookeeping*, and so NVIDIA cmae up with Unified Memory, an advance topic, but it deals with GPU managed arrays. Visit here for a tutorial, a blog post from NVIDIA, 2007

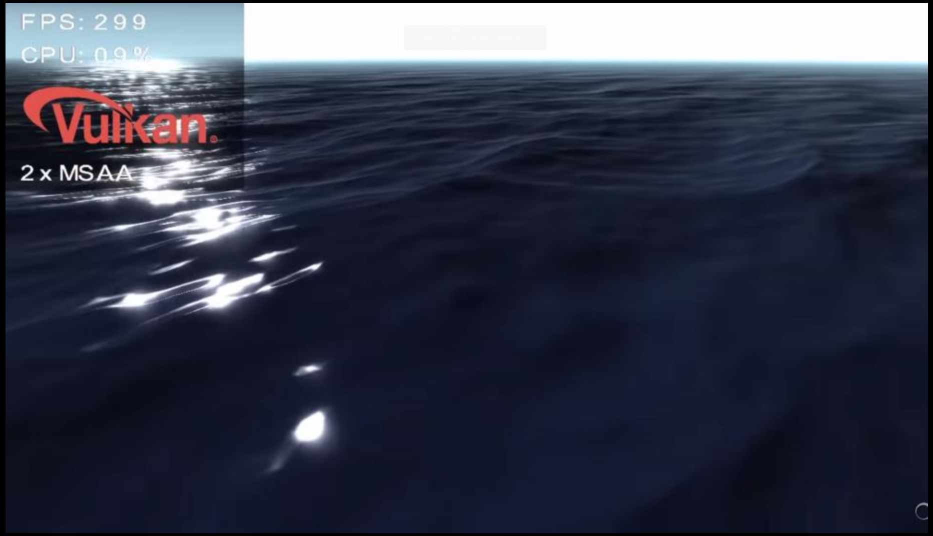
OpenGL, Vulkan and CUDA...?

- Not going into the details, but this is completely possible. If interested, you can check out
  - Vulkan with CUDA
  - OpenGL with CUDA
  - OpenGL code sample reference attached at the end

Completely possible

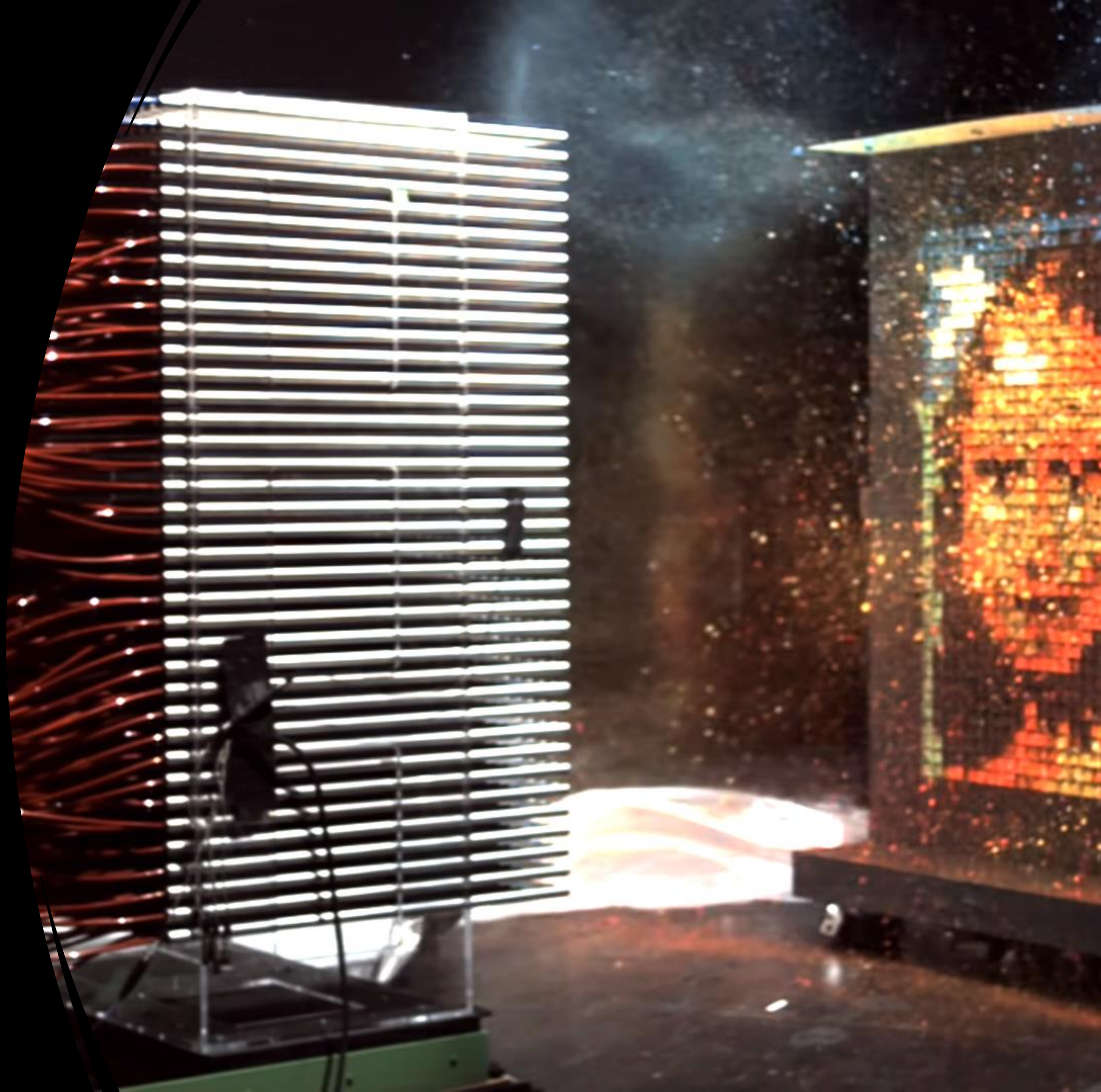
- Vulkan adoption is slower, since it is harder to write
- OpenGL is much simpler to work with, but still complex for new beginners
- For reference, check out this video on YT





A very cool presentation by Mythbusters

- Where the CPU version of a machine they use to paint Mona Lisa hits the target board one at a time
- The NVIDIA GPU version shoots multiple paint balls simultaneously as needed, completing it less than 10 seconds
- Check out the full presentation here on YT



Project Ideas!

- Look for CUDA libraries, languages, an API with features/functions that look useful and apply them to a problem of personal interest

- Go through the samples, find something that interests you, try to describe and explain it in a blog post, or an article, a brief description, and publish the writeup, share it, and ask your friends about it

- Go through the GTC talks. Find something interesting and worthwhile to work on as an exploratory project

- Visit the CUDA tag on Stack Overflow, see if there is a problem that you like to tackle

- Maybe find parallel programming books, and find problems to apply it in CUDA?

- Also, you can make an article post on Medium about the book you just read. Give it a review, discuss the presentation of ideas

- Find research papers and see if you can improve the performance of some implemented algorithms in terms of CUDA

- Explore CUDA with either OpenGL or Vulkan for graphics

- Explore the GPU Applications Catalog, see if you like to make a project similar to the ones mentioned here.

- Think about making image processing applications that implement CUDA processing under the hood to efficiently apply, for example, filters, shading, adjust vibrance, saturation, hue, tint, and other options like these

- Can you use CUDA on networking applications? If so, how?

when the code you wrote is all wrong but atleast you optimized it

Break!
Currently summary for discussed topics.
Any Q/A?

Last Topics, if time remains

Saved for the last few minutes. More technical details

Stencils And Shared Memory

Trying to emulate accessing the state of other threads involved in SIMT

Stencils And Shared Memory

- In many applications, we have the opportunity to split the problem into multiple stages, some of which can be done independently
- In SIMT parallelism, each thread has access to its own version of each kernel variable, but no access to the versions in other threads
- You can try writing an array which 'hold' a possible shared state between two threads, but with hundreds, thousands, or even millions of threads, this easily gets messy
- The general principle is, "the farther the data is from the processor, the longer it takes to process the data"
- Shared memory aims to bridge this gap, by residing adjacent to the SM, and provides up to 48KB of storage, that can be accessed by all threads in a grid

Reduction And Atomic Functions

Reduction, where all the threads contribute to a single output

Reduction And Atomic Functions

- Frequent, and very naïve examples are dot product and term-wise multiplications
- Naïve approaches to solve this problem, global address memory issue
- See "Reduction And Atomic Function Example" example in the notebook, here

Thrust Library

CUDA Toolkit Thrust is a standard C++ implementation, based on the Standard Template Library (STL)

Thrust Library

- Collection of scan, sort, and reduction operations, functions, and algorithms
- Can be used to implement quick and easy versions of complex parallel algorithms in concise, readable source code
- Good for robustness, readability, and absolute performance
- Needs some familiarity with the STL C++ library to understand what templating is about and what generics are
- Also included in the notebook, link again is [this]

Open Source Projects

A complete history of pretty awesome open source projects.

Code Samples

- https://github.com/myurtoglu/cudaforengineers

- https://github.com/NVIDIA/cuda-samples

- https://github.com/NVIDIA/CUDALibrarySamples

References

- "CUDA For Engineers", by Mete Yurtoglu, Duane Storti. Simply this presentation and my personal learning would not have been possible without this
- The awesome, awesome, NVIDA team at https://developer.nvidia.com/, this would not have been possible with out them.

developer.nvidia.com/teaching-kits

Questions?