The Accelerated Python Developer's Toolbox

Katrina Riehl, PhD
Principal Technical Product Manager - CUDA Python
SciPy Tacoma July 7, 2025

Getting Started

Register now to save time later

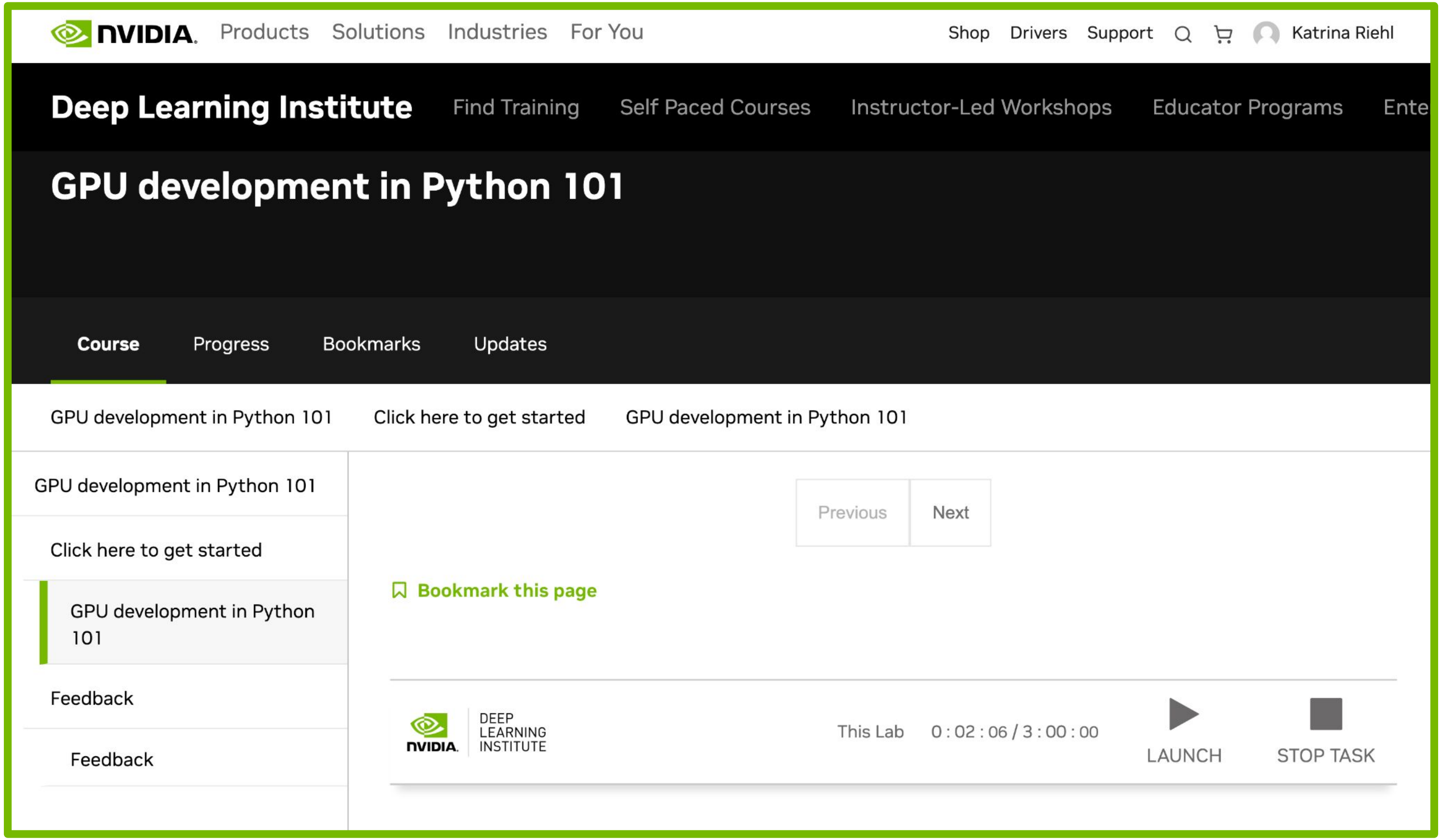
1. **Create or log into your** NVIDIA Developer Program account **- https://learn.nvidia.com/.** You will receive an email letting you know when your account is ready. This account will provide you with access to all of the DLI training materials during and after the workshop. You will have **six months** of access to all course materials.
2. **Visit websocketstest.courses.nvidia.com and make sure all three test steps are checked "Yes."** This will test the ability for your system to access and deliver the training contents. If you encounter issues, try updating your browser. Note: Only Chrome and Firefox are supported.

Now you're ready to get started with the tutorial!

Simply enter the code **XXX_XXX_XXX** at learn.nvidia.com/dli-event

Notify a TA if You Don't See This Page

Hit the Launch button



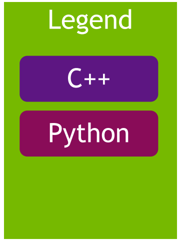
Exposures of the CUDA Ecosystem

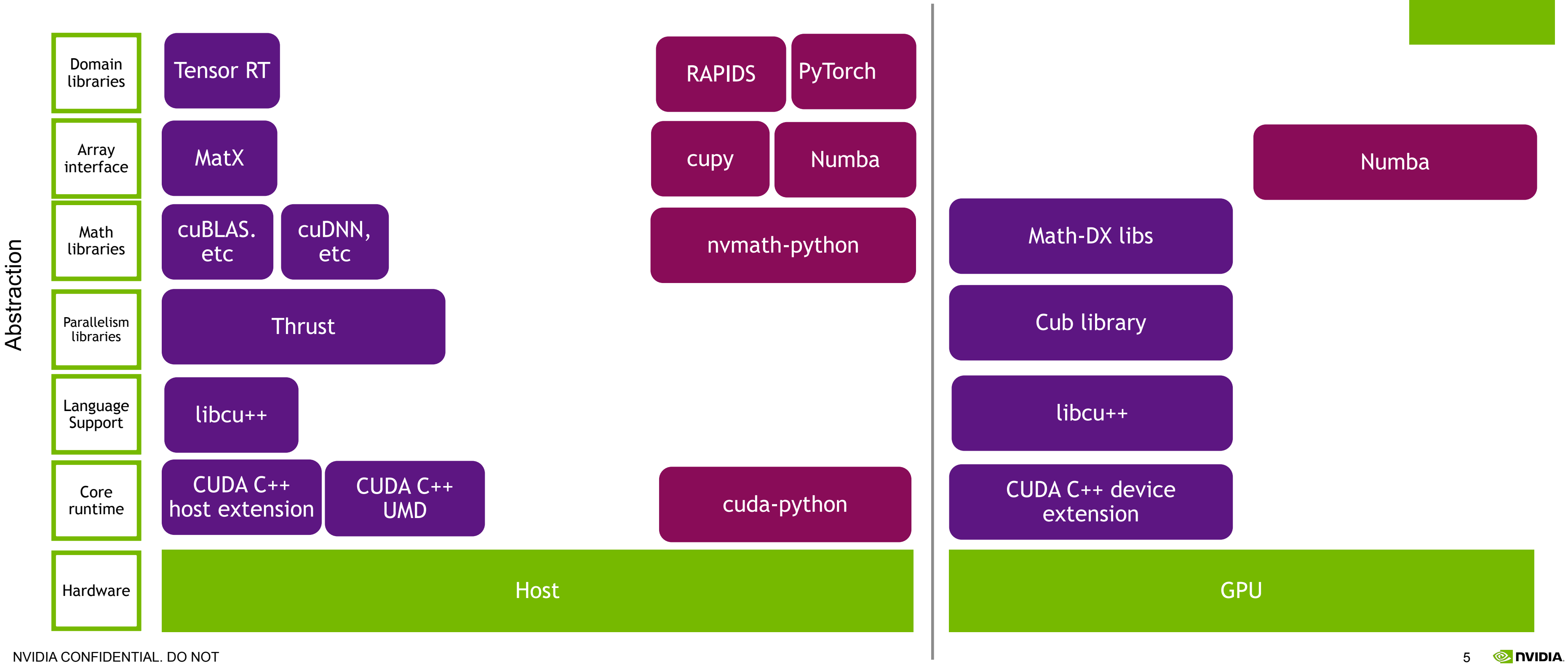
Some folks like the productivity of frameworks, others like the performance of raw hardware

- Frameworks
- Domain Libraries
- Array Interfaces
- Math Libraries
- Device Parallelism
- Language Feature Support
- Core Runtime
- Hardware

Exposures of the CUDA Ecosystem

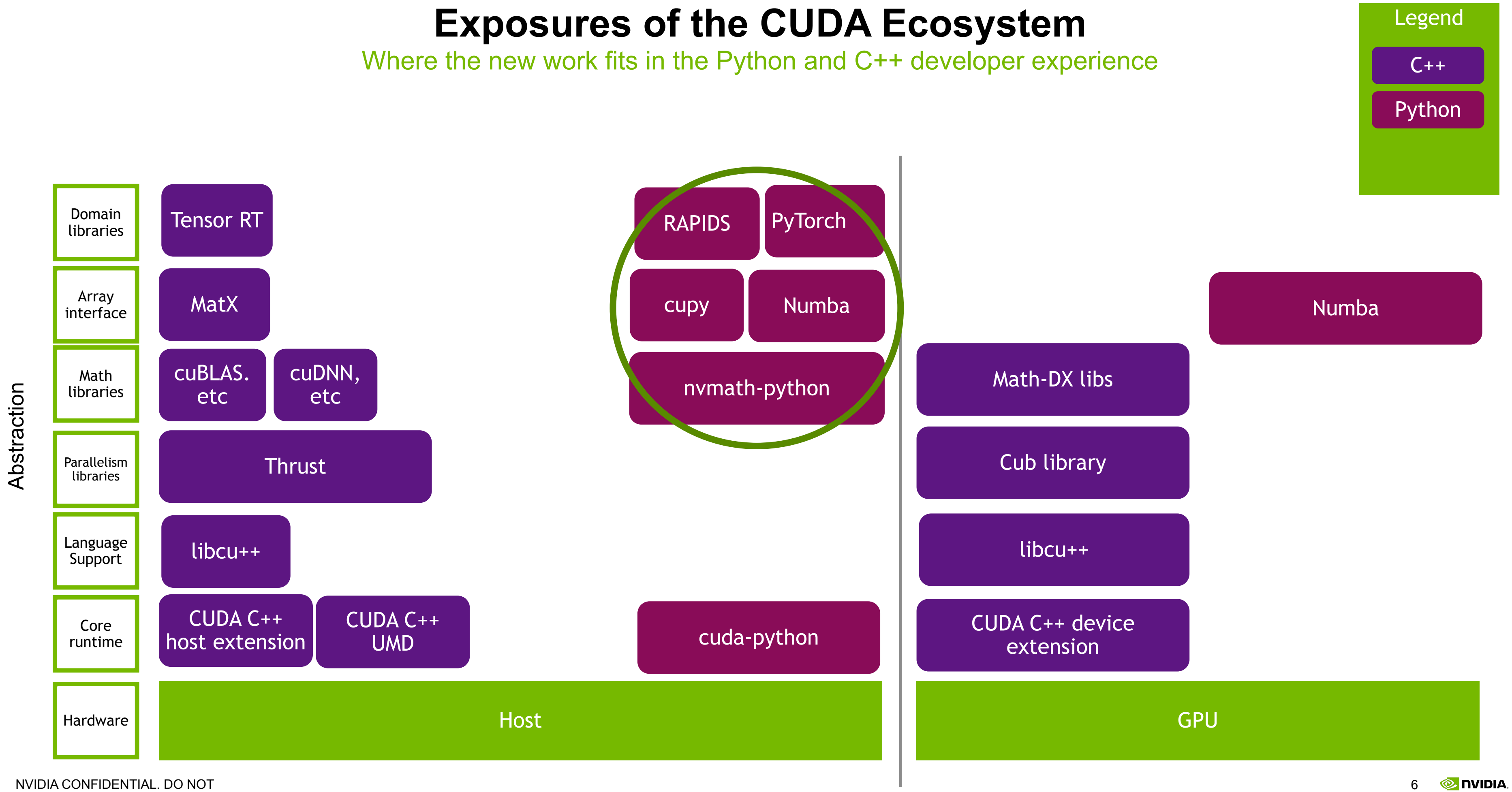
Where the new work fits in the Python and C++ developer experience





Exposures of the CUDA Ecosystem

Where the new work fits in the Python and C++ developer experience



Accelerated Computing Swim Lanes

RAPIDS makes accelerated computing more seamless while enabling specialization for maximum performance

Easier to Use

Maximum Performance

Zero Code Change: Acceleration Plugins *(no-code change)*

cudf.pandas: Accelerated Pandas, Accelerated NetworkX nx-cugraph, RAPIDS Spark Accelerator, Array-API backed Scikit-learn...

Hybrid CPU/GPU libraries *(minimal change)*

Pytorch, FAISS, Tensoflow, XGBoost, cuML-CPU ...

Dask, pySpark...

GPU Python Libraries *(GPU Python code)*

RAPIDS core libraries (cuDF, cuML, cuGraph, cuVS), RMM, CuPy, Numba, OpenAI Triton ...

Python/CUDA libraries *(Hybrid Python / CUDA code)*

CuPy RawKernels, Numba CUDA, Cython wrappers for CUDA ...

C++/CUDA high level *(High-level C++/CUDA code)*

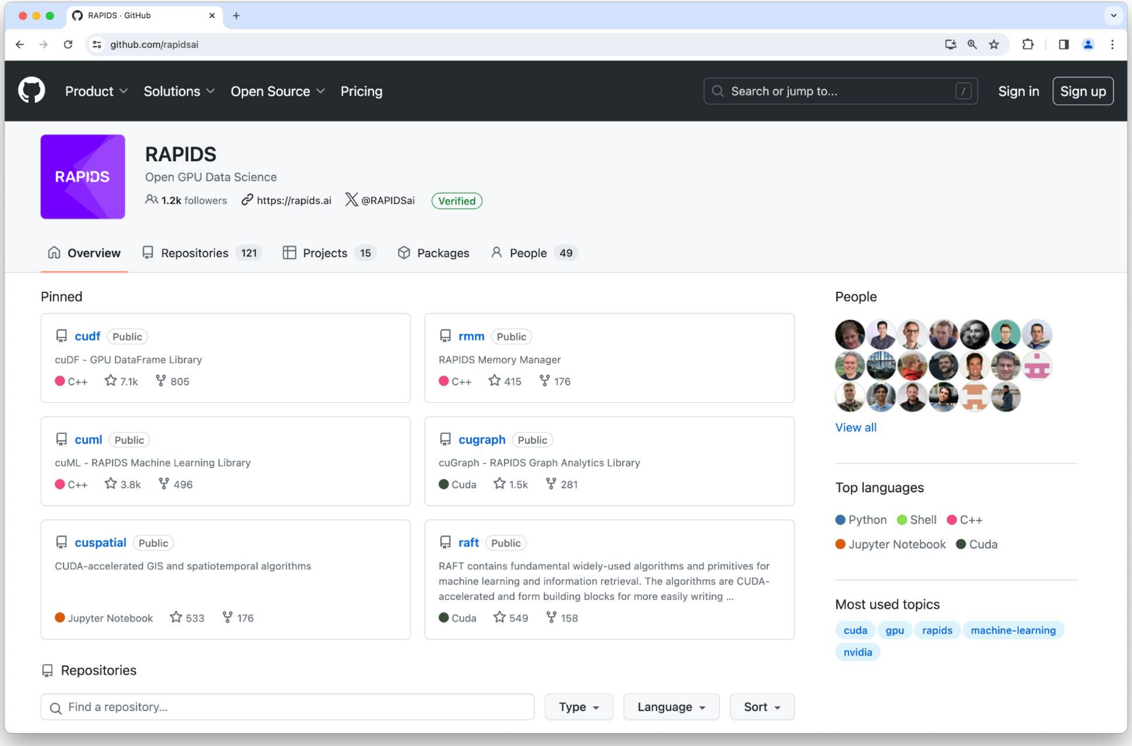
RAFT, CCCL: Thrust, CUB ...

CUDA Toolkit *(C++/CUDA code and kernels)*

cuBLAS, cuDNN, cuSolver, cuSPARSE, ...

RAPIDS

https://github.com/rapidsai



Our mission

RAPIDS

“Unlock the speed of GPUs with code you already know”

https://rapids.ai/learn-more/

cudf.pandas

cuDF pandas accelerator mode

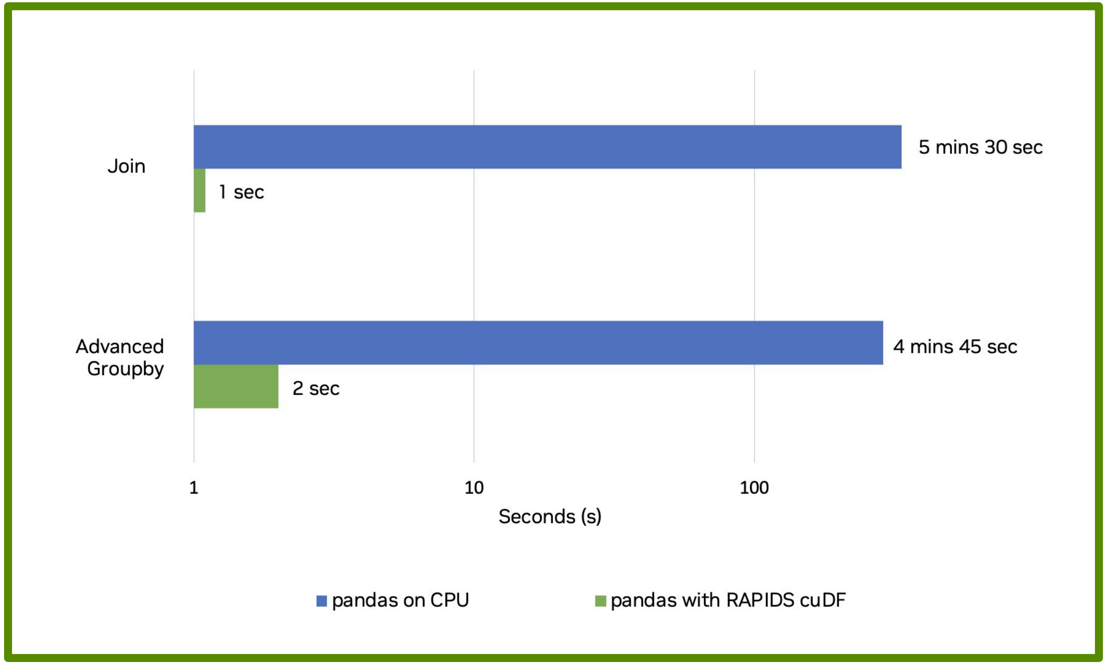
cuDF pandas accelerator mode (`cudf.pandas`) is built on cuDF and **accelerates pandas code** on the GPU. It supports **100% of the Pandas API**, using the GPU for supported operations, and automatically **falling back to pandas** for other operations.

```
%load_ext cudf.pandas
# pandas API is now GPU accelerated

import pandas as pd

df = pd.read_csv("filepath")  # uses the GPU!
df.groupby("col").mean()  # uses the GPU!
df.rolling(window=3).sum()  # uses the GPU!
df.apply(set, axis=1)  # uses the CPU (fallback)
```

https://docs.rapids.ai/api/cudf/stable/cudf_pandas/

150x Faster pandas with Zero Code Change

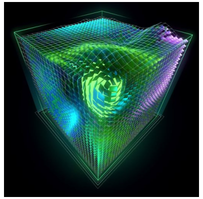
DuckDB Data Benchmark, 5GB



Performance comparison between Traditional pandas v1.5 on Intel Xeon Platinum 8480CL CPU and pandas v1.5 with RAPIDS cuDF on NVIDIA Grace Hopper
Source: https://developer.nvidia.com/blog/rapids-cudf-accelerates-pandas-nearly-150x-with-zero-code-changes/

What is Numba? Who uses it? (1)

- Toolbox / framework that compiles and executes Python code for CUDA and CPUs:



- Users / use cases:

Use case	Example users
Write SIMT CUDA kernels in Python	*NeMo, ipie, STUMPY, …*
Support Python User-Defined Functions	*RAPIDS, DALI, Awkward Array, …*
Building Python compilers	*CUDA Python Tile compilation, …*
Exposing C++ device libraries to Python	*Numbast, nvmath-python, cuda.parallel / CCCL, …*

Examples

Python CUDA SIMT Compiler / Python UDF compiler

Numba

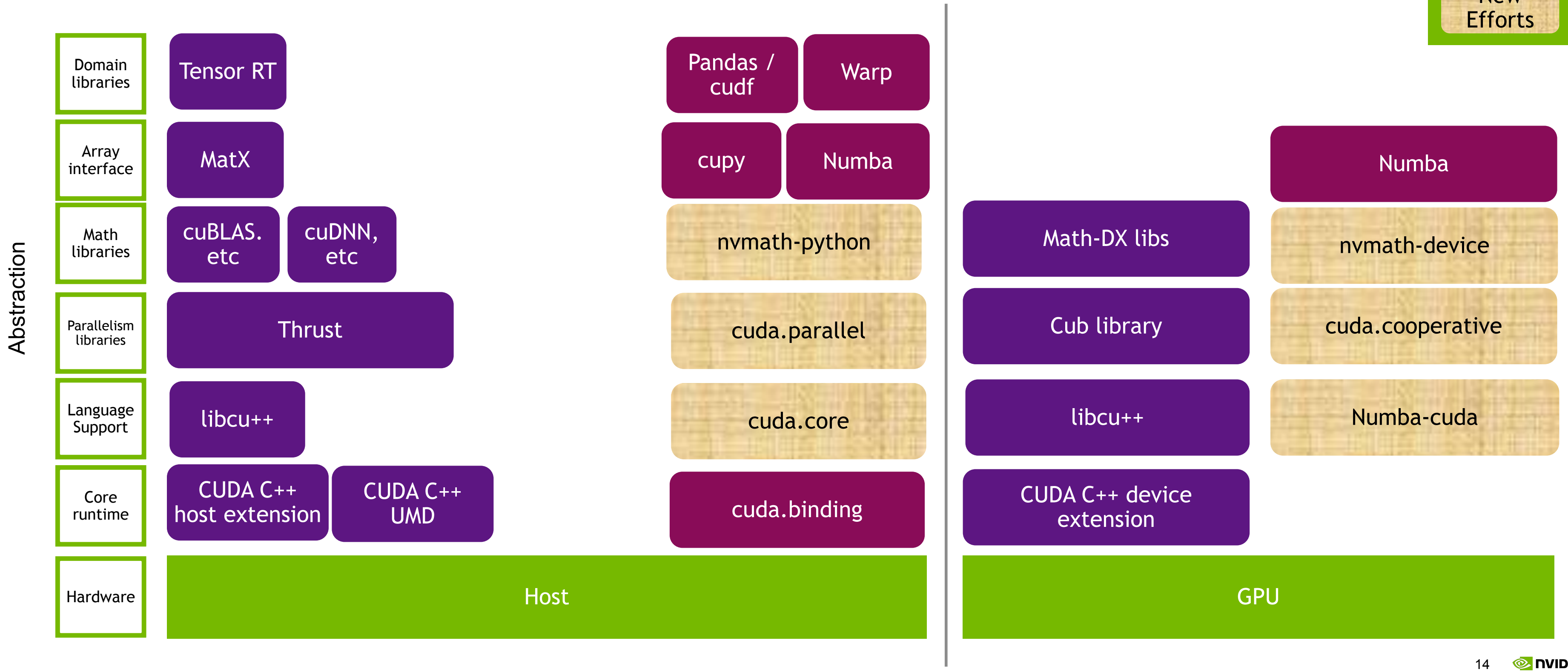
```python
from numba import cuda, njit

# CPU pipeline
@njit
def vector_add_cpu(x, y):
    return x + y

# CUDA pipeline
@cuda.jit
def vector_add_cuda(r, x, y):
    start = cuda.grid(1)
    stop = len(r)
    step = cuda.gridsize(1)

    for i in range(start, stop, step):
        r[i] = x[i] + y[i]

# Launch kernel over grid
vector_add[grid_dim, block_dim](r, x, y)
```

RAPIDS pandas

```python
import cudf

# Defining a series:
s = cudf.Series([1, 2, 3, None, 4])

# A user-supplied Python function:
def add_ten(num):
    return num + 10

# Compiles add_ten() for CUDA GPU and runs it.
# Result: (11, 12, 13, <NA>, 14)
s.applymap(add_ten)
```

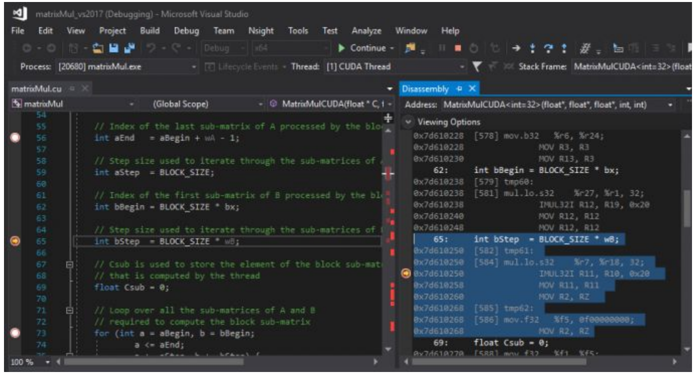
Exposures of the CUDA Ecosystem

Where the new work fits in the Python and C++ developer experience

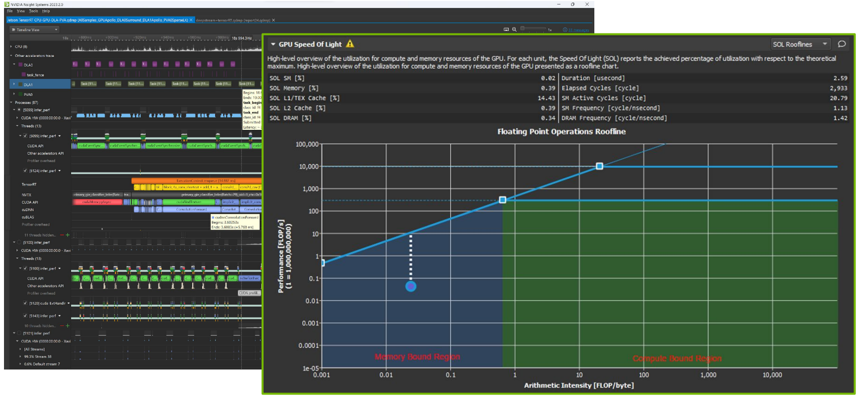


Developer Tools Ecosystem

Debuggers: cuda-gdb, Nsight Visual Studio Edition
Nsight Visual Studio **Code** Edition



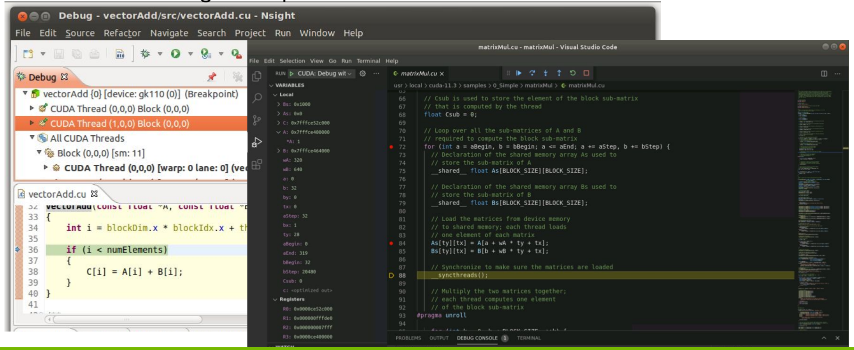
Profilers: Nsight Systems, Nsight Compute, CUPTI, NVIDIA Tools eXtension (NVTX)



Correctness Checker: Compute Sanitizer

```
$ compute-sanitizer --leak-check full memcheck_demo
========= COMPUTE-SANITIZER
Mallocing memory
Running unaligned_kernel
Ran unaligned_kernel: no error
Sync: no error
Running out_of_bounds_kernel
Ran out_of_bounds_kernel: no error
Sync: no error
========= Invalid __global__ write of size 4 bytes
=========     at 0x60 in memcheck_demo.cu:6:unaligned_kernel(void)
=========     by thread (0,0,0) in block (0,0,0)
=========     Address 0x400100001 is misaligned
```

IDE integrations: Nsight Visual Studio **Code** Edition
Nsight Visual Studio Edition
Nsight Eclipse Edition



NVIDIA

Performance Issues

Keep an eye out for these common barriers to GPU performance

- To achieve optimal performance in CUDA, consider:
  - Localizing memory access in order to minimize memory latency.
  - Maximizing the number of active threads per multiprocessor to ensure high utilization of your hardware.
  - Minimization of conditional branching.

- To overcome the bottleneck between CPU and GPU across the PCIe bus, we want to:
  - Minimize the volume of data transferred. Transferring data in large batches can minimize the number of data transfer operations.
  - Organize data in a way that complements the hardware architecture.
  - Utilize asynchronous transfer features that will allow computation and data transfer to occur simultaneously. Overlapping data transfers with computation can hide latencies caused by data transfers.

Deployment Guidance

https://docs.rapids.ai/deployment/stable/

Deploying RAPIDS

Deployment documentation to get you up and running with RAPIDS anywhere.

Local Machine

Use RAPIDS on your local workstation or server.

docker conda pip WSL2

Cloud

Use RAPIDS on the cloud.

Amazon Web Services Google Cloud Platform Microsoft Azure IBM Cloud

HPC

Use RAPIDS on high performance computers and supercomputers.

SLURM

Platforms

Use RAPIDS on compute platforms.

Kubernetes Kubeflow Coiled Databricks Google Colab

Tools

There are many tools to deploy RAPIDS.

containers dask-kubernetes dask-operator dask-helm-chart dask-gateway

Workflow Examples

For inspiration see our example notebooks with opinionated deployments of RAPIDS to boost machine learning workflows.

xgboost optuna mlflow ray tune

Guides

Detailed guides on how to deploy and optimize RAPIDS.

Microsoft Azure Infiniband MIG

NVIDIA NIM Microservices

NVIDIA NIM Microservices using RAPIDS to accelerate your AI deployment.

Natural Language Processing Data Processing

Developer

Build on RAPIDS in your development environments.

CI

Learn New Skills for Free

Access essential technical training

Get a free technical course (worth up to $90). Scan the QR code, enroll in your chosen course today, and complete within the next 6 months.

Courses include:

- Accelerating Clustering Algorithms to Achieve the Highest Performance
- Analyzing and Visualizing Large Data Interactively using Accelerated Computing
- Accelerating End-to-End Data Science Workflows
- Best Practices in Feature Engineering for Tabular Data With GPU Acceleration
- Augment your LLM Using Retrieval Augmented Generation
- Building LLM Applications With Prompt Engineering
- Building RAG Agents with LLMs
- Fundamentals of Accelerated Computing with CUDA Python
- Generative AI with Diffusion Models
- Getting Started with Deep Learning
- Introduction to Deploying RAG Pipelines for Production at Scale
- Introduction to NVIDIA NIM™ Microservices

Scan the QR code to access the full course list and redeem your free training. https://sp-events.courses.nvidia.com/AIDaysEU