### ACCELERATED STANDARD LANGUAGES

ISO C++, ISO Fortran

### INCREMENTAL PORTABLE OPTIMIZATION

OpenACC, OpenMP

### PLATFORM SPECIALIZATION

CUDA

```c
std::transform(par, x, x+n, y, y, [=](float x, float y){ return y + a*x; })

do concurrent (i = 1:n)
y(i) = y(i) + a*x(i)
enddo

#pragma acc data copy(x,y) {
... std::transform(par, x, x+n, y, y, [=](float x, float y){ return y + a*x; })
... }

#pragma omp target data map(x,y) {
... std::transform(par, x, x+n, y, y, [=](float x, float y){ return y + a*x; })
... }

def saxpy(a, x, y):
    y[:] += a*x

__global__ void saxpy(int n, float a,
                      float *x, float *y) {
    int i = blockIdx.x*blockDim.x + threadIdx.x;
    if (i < n) y[i] += a*x[i];
}

int main(void) {
... cudaMemcpy(d_x, x, ...);
cudaMemcpy(d_y, y, ...);
saxpy<<<(N+255)/256,256>>>(...);
cudaMemcpy(y, d_y, ...);
}
```

### ACCELERATION LIBRARIES

- Core
- Math
- Communication
- Data Analytics
- AI
- Quantum
Agenda

- High-level tools ecosystem
- For each tool:
  - Brief description and feature overview
  - What’s new in the latest releases
  - Focus on problem solving and application in the system hierarchy
- Summary - Putting it all together
Monitoring → Profiling

- Different target users, intent, details, design tradeoffs
  - Some overlap but not interchangeable
- Monitors
  - Goal: Coarsely observe quality, utilization, progress, help rate jobs
  - Users: Admins & users of top, netstats, taskman, ...
  - Track issues back to jobs
    - Low rate (minutes or seconds)
    - Sensible to observer checking & reacting, low overhead
    - Less details towards root cause
  - Uptime: Usually 100%
- Profilers
  - Goal: Aid in application optimization
  - Users: Engineers looking to relating back to areas of code
  - Uptime: Usually not 100%

Data Center GPU Monitor (DCGM)
Nsight Systems
Nsight Compute
Nsight Graphics

Coarse & Lite → Fine & Detailed
Data Center GPU Monitor (DCGM)

- Very unobtrusive
- Great for fleet health, job health, job rating
- Optimized for low frequency statistics (10hz or less)
- Maps to jobs & processes
  - But not code & algorithms, like profilers!
- Pause/resume to profile
  - CLI
    - `dcgmi profile --pause`
    - `dcgmi profile --resume`
  - API
    - `dcgmProfPause`
    - `dcgmProfResume`
Nsight Profiler Family
Performance Analysis Workflow

Nsight Systems
Analyze algorithms system-wide
CPU, GPU, CUDA, Networking, Graphics

Nsight Compute
Debug & analyze
CUDA kernels

Nsight Graphics
Debug & analyze
Graphics shaders & frames
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
```
GUI Support for Grace and other Arm Platforms


• GUIs run natively on
  • NVIDIA® Jetson AGX Orin™ SoC
  • Arm Server Platforms
• Use new native GUI or existing remote collection capabilities
Updated Websites/Docs/Icons

NVIDIA Developer Tools

NVIDIA Nsight™ tools are a powerful set of libraries, SDKs, and developer tools spanning across desktop and mobile targets that enable developers to build, debug, profile, and develop software that utilizes the latest accelerated computing hardware.


What Segment Are You Working on?

- View All
- CUDA/Compute
- Graphics
- Optix
- Deep Learning

Developer Activity

- Code Development
- Debugging/Correctness

[Platform Analysis]

Nsight Compute

Nsight Compute is an interactive kernel profiler for CUDA applications. It provides detailed performance metrics and API.

Compute Debuggers/IDEs
Compute Debuggers
Debug GPU Kernels Running on Device

• CUDA GDB
  • CPU + GPU CUDA kernel debugger
  • Supports stepping, breakpoints, in-line functions, variable inspection etc...

• Nsight Visual Studio Edition
  • IDE integration for Visual Studio
  • Build and Debug CPU+GPU code from Visual Studio

• Nsight Visual Studio Code Edition
  • New IDE integration for VS Code
  • Build and Debug CPU+GPU code from Visual Studio Code
  • Remotely target Linux targets from Windows or Linux
CUDA GDB

Command-line and IDE Back-end Debugger

• Unified CPU and CUDA Debugging
• CUDA-C/SASS support
• Built on GDB and uses many of the same CLI commands
• Local/Remote connection support
• Backend for IDE debuggers
Compute Sanitizer
Automatically Scan for Bugs and Memory Issues

• Compute Sanitizer checks correctness issues via sub-tools:
  • **Memcheck** – Memory access error and leak detection tool.
  • **Racecheck** – Shared memory data access hazard detection tool.
  • **Initcheck** – Uninitialized device global memory access detection tool.
  • **Synccheck** – Thread synchronization hazard detection tool.

https://github.com/NVIDIA/compute-sanitizer-samples
Correctness Tools Features

- **Debuggers using Unified Backend**
  - Consistent debugging experience across platforms
  - Performance and usability improvements
  - New features “float all boats”

- **New Compute Sanitizer use cases**
  - Racecheck support for async-copy, DSMEM (Hopper), and GMMA (Hopper)
  - Memcheck for cache control operations
  - Improved core dump support
  - Multiple OptiX analysis improvements

- **Nsight Visual Studio Code Edition**
  - New support for debugging containers

---

Invalid __global__ write of size 1 bytes at 0x4d70 in
/home/cuda/optixBasic/draw_solid_color.cu:69:__raygen__draw_solid_co lor_0xebf766b2f0642d4e
by thread (0,0,0) in block (0,0,0)
Address 0x7f878f8900403 is out of bounds
and is 262,132 bytes after the nearest allocation at 0x7f878f8c0400 of size 16 bytes
Device Frame:NVIDIA internal [0x430]
Saved host backtrace up to driver entry point at kernel launch time
Host Frame: optixBasic.cpp:227:main [0xb735]
in /home/cuda/optixBasic/optixBasic
Host Frame:../sysdeps/nptl/libc_start_call_main.h:58:__libc_start_call_main [0x2df0]
in /lib/x86_64-linux-gnu/libc.so.6
Host Frame:../sysdeps/nptl/libc_start_call_main.h:58:__libc_start_call_main [0x2e07d]
in /lib/x86_64-linux-gnu/libc.so.6
Host Frame: [0x8dde]
NSIGHT PROFILERS WORKFLOW

Nsight Systems
Comprehensive system-level performance

Dive into top CUDA kernels by using metrics/counter collection

Dive into graphics frames

Nsight Compute
Detailed CUDA kernel performance

Nsight Graphics
Detailed frame/render performance

Start here

Re-check overall performance

Re-check overall performance
Nsight Systems
System Profiler

Key Features:

• System-wide application algorithm tuning
  • Multi-process tree support
• Locate optimization opportunities
  • Visualize millions of events on a very fast GUI timeline
  • Identify 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
• Command Line, Standalone, IDE Integration

• OS: Linux (x86, Power, ARM Server, Tegra), Windows, macOS X (host)
• GPUs: Pascal+

• Docs/product: https://developer.nvidia.com/nsight-systems
Zoom/Filter to Exact Areas of Interest
Pixel Time Coverage Based Level Of Detail (LOD)

Zooming in reveals gaps where there were valleys
Application Profiles with Nsight Systems

$ nsys profile -o report -stats=true ./myapp.exe

• Generated file: report.qdrep (or report.nsys-rep)
  Open for viewing in the Nsight Systems UI

• When using MPI, recommended to use nsys after mpirun/srun:
  $ mpirun -n 4 nsys profile ./myapp.exe
Core Strategy

- What’s HOT?
  - Will it be easier to shrink what I coded?
  - This is where MOST people concentrate. ...intuitive but not always best !/$

- What’s COLD?
  - Will it be easier to take advantage of the something unused?
  - Free money? Yes please!

- Hot spots might be:
  - Parallelizable?
  - Shrinkable without compromising accuracy, memory, etc

- Cold spots are clear, measurable opportunities!!!
  - How can I remove or fill them?
  - Where do I have incorrect/unnecessary/unexpected dependencies & synchronization?
    - Between threads, processes or across nodes (ex: MPI_Barrier)!
General optimization tips

- Using tensor cores?
- Increase grid and batch size to utilize GPUs width
- Conventional parallelism - more worker threads!
- Parallel pipelining
  - No data dependency? Parallelize!
  - Prefetch next batch/iteration during computation
General optimization tips

• Fuse tiny kernels, copies, memsets.
  o Check out CUDA Graphs

• Overlap/oversubscribe with MPS

• Multi-buffering
  o Don’t make everyone wait on the same piece of memory
  o Double, triple buffer

• Avoid moving data back to the CPU
  • Pre-allocate and recycle!

• Minimize managed memory page faults
  o Prefetch!
Nsight Systems
New Features
CPU Counters and OS Metrics

- CPU X86_64/SBSA hardware counters
  - Cache misses, DRAM access, etc...
Python Call Stack Sampling

- Periodically sampled stacks
- Correlated with all other collected activity
NVIDIA Networking NIC/DPU Metrics Sampling

- Includes InfiniBand and Ethernet
- Sent / Received / Waits
- Correlate with expected network traffic and other system activities
Network API Trace - Extended Descriptions/Tooltips

- Supports MPI versions
  - OpenMPI
  - MPICH
- Data for each API call
- Identify performance trends and specific underperforming calls
Nsight Systems Multi-Node Analysis

Preview in Nsight Systems 2023.2

- Recipes to analyze existing reports collected from multiple sources/nodes
- Several default recipes included with Nsight Systems
- Recipes built on extensible Python library
- Recipes generate results various formats:
  - Jupyter Notebook
  - Parquet, CSV, ...
- Open result in Nsight Systems GUI or Jupyter Lab

```
> ./nsys recipe -help

  cuda_api_sum -- CUDA API Summary
  cuda_api_sync -- CUDA Synchronization APIs
  cuda_gpu_kern_sum -- CUDA GPU Kernel Summary
  cuda_gpu_mem_size_sum -- CUDA GPU MemOps Summary (by Size)
  cuda_gpu_mem_time_sum -- CUDA GPU MemOps Summary (by Time)
  cuda_memcpy_async -- CUDA Async Memcpy with Pageable Memory
  cuda_memcpy_sync -- CUDA Synchronous Memcpy
  cuda_memset_sync -- CUDA Synchronous Memset
  dx12_mem_ops -- DX12 Memory Operations
  gpu_gaps -- GPU Gaps
  gpu_time_util -- GPU Time Utilization
  nvtx_gpu_proj_trace -- NVTX GPU Trace
  nvtx_sum -- NVTX Range Summary
  osrt_sum -- OS Runtime Summary
```

> ./nsys recipe cuda_gpu_kern_sum --dir ~/profiles
> Generated cuda_gpu_kern_sum-1
Nsight Systems Multi-Node Analysis

Before

After

- cuda_gpu_time_util_map recipe
- Identify misbehaving GPUs, ranks, or nodes
- Time correlation with application phases
- Click to drill into specific result file for more details
NVTX
Tools Extension API
NVIDIA Tools eXtension

**NVX v3**

- Decorate application source code with annotations (markers, ranges, nested ranges, ...) to help visualize execution with debugging, tracing and profiling tools.

- **Header-only library** [https://github.com/NVIDIA/NVTX/tree/release-v3/c](https://github.com/NVIDIA/NVTX/tree/release-v3/c)

  ```
  #include <nvtx3/nvToolsExt.h>
  ```

- **Marker**:
  ```
  nvtxMark("This is a marker");
  ```

- **Push-Pop range**
  ```
  nvtxRangePush("This is a push/pop range");
  // Do something interesting in the range.
  nvtxRangePop(); // Pop must be on same thread as corresponding Push
  ```

- **Start-End range**
  ```
  nvtxRangeHandle_t handle = nvtxRangeStart("This is a start/end range");
  // Somewhere else in the code, not necessarily same thread as Start call:
  nvtxRangeEnd(handle);
  ```

NVIDIA SDKs and NVTX
Advanced profiling and performance visualization

Math Libraries
- cuPAR
- cuSOLVE
- cuBLAS

Deep Learning Libraries
- DeepStream
- TensorRT
- cuDNN
- cuDLA

Comm. Libraries
- NVSHMEM
- NCCL

RAPIDS
- cuDF
- cuFile
- cuML

GXF

UCX
Profiling DL Models

• Pytorch
  o DNN Layer annotations are disabled by default
  o `++ "with torch.autograd.profiler.emit_nvtx():"`
  o Manually with `torch.cuda.nvtx.range_(push/pop)`
  o TensorRT backend is already annotated

• Tensorflow
  o Annotated by default with NVTX in NVIDIA TF containers
  o `TF_DISABLE_NV TX_RANGES=1` to disable for production
Nsight Compute
Nsight Compute
Kernel Profiler

Key Features:

- Interactive CUDA API debugging and kernel profiling
- Built-in rules expertise
- Fully customizable data collection and display
- Command Line, Standalone, IDE Integration, Remote Targets

- OS: Linux (x86, Power, Tegra, Arm SBSA), Windows, macOS X (host only)
- GPUs: Volta+

Nsight Compute GUI Interface

- Targeted metric sections
- Customizable data collection and presentation
- Built-in expertise for Guided Analysis and optimization
Visual memory analysis chart

Metrics for peak performance ratios
Source/PTX/SASS analysis and correlation

Metric heatmap to quickly identify hotspots

Source metrics per instruction
Kernel profiles with nsight compute

$ ncu -k mykernel -o report ./myapp.exe

• Generated file: report.ncu-rep
• Open for viewing in the Nsight Compute UI

• (Without the -k option, Nsight Compute with profile everything and take a long time)
Nsight Compute
New Features
Integrated Basic Trace from Nsight Systems

- Use “System Trace” activity in the connection dialog
- Identify long kernels or compute-bound bottlenecks
- Right-click kernel in timeline to quickly launch profile
- Nsight Compute automatically filters to selected kernel
Source Table for Inline Functions

- Metrics can be analyzed per inline site or aggregated for the entire function
- Use compiler `--lineinfo` flag to generate symbols
Hopper Thread Block Clusters

- Enables programmatic control of locality at a granularity larger than a single thread block on a single SM
- Clusters enable multiple thread blocks running concurrently across multiple SMs to synchronize and collaboratively fetch and exchange data.
- For more information on Thread Block Clusters see:
  - CUDA: New Features and Beyond
  - CUDA Programming Model for Hopper Architecture
Hopper Thread Block Clusters in Nsight Compute

Cluster Launch Size

Cluster Scheduling Policy

Cluster Occupancy

Hopper SM 90
New Documented Samples
Working to Ease the Complexity of Performance Optimization

- Packaged with Nsight Compute installation
- Includes reproducible instructions, descriptions, pre-collected results, and source code
- Explaining common performance issues in CUDA applications
- **Uncoalesced Global Accesses** and **Shared Memory Bank Conflicts**
Source Page Guidance

- **Branch Navigation**
  - Quickly jump to target of branch instructions

- **Rules Markers**
  - Automated analysis and rules output per source line
  - Quick detection and navigation to lines that matter
  - Explanation of detected issues
Potential Speedup Calculations

- Estimated speedup potential calculated for detected issues
- Sort kernels based on potential speedup
- Helps users understand where to focus effort and potential benefit
Putting it all together
From the Macro to the Micro

Putting it all together
From the Macro to the Micro

Putting it all together
From the Macro to the Micro

Putting it all together
From the Macro to the Micro

Putting it all together
From the Macro to the Micro

Putting it all together
Cyfronet AI for Science Bootcamp

- **Dates:** September 11-12, 2023
- **Event Focus:** AI
- **Event Format:** Virtual Event
- **Region:** Europe/Middle East/Africa
- **Application Status:** Open

CESGA AI for Science Bootcamp

- **Dates:** September 18-19, 2023
- **Event Focus:** AI
- **Event Format:** Virtual Event
- **Region:** Europe/Middle East/Africa
- **Application Status:** Open

NASA Open Hackathon

- **Dates:** September 12-21, 2023
- **Event Focus:** HPC+AI
- **Event Format:** Virtual Event
- **Region:** North America/Latin America
- **Application Status:** Closed

CSCS Open Hackathon

- **Dates:** September 18-22, 2023
- **Event Focus:** HPC+AI
- **Event Format:** In-Person Event
- **Region:** Europe/Middle East/Africa
- **Application Status:** Closed

NERSC AI for Scientific Computing Bootcamp

- **Dates:** October 18-20, 2023
- **Event Focus:** AI
- **Event Format:** Virtual Event
- **Region:** North America/Latin America
- **Application Status:** Open

NSM Open Hackathon

- **Dates:** October 10-20, 2023
- **Event Focus:** HPC+AI
- **Event Format:** Hybrid Event
- **Region:** Asia-Pacific
- **Application Status:** Open
Developer Tools Across GTC

• Sessions
  • S51205: From the Macro to the Micro - CUDA Developer Tools Find and Fix Problems at Any Scale
  • S51421: Optimizing at Scale: Investigating and Resolving Hidden Bottlenecks for Multi-Node Workloads
  • S51882: Become Faster in Writing Performant CUDA Kernels using the Source Page in Nsight Compute
  • S51772: Debugging CUDA: An Overview of CUDA Correctness Tools
  • S51230: Orin Performance Bodybuilding with Nsight Developer Tools
  • SE52434: Jetson Edge AI Developer Days: Getting the Most Out of Your Jetson Orin Using NVIDIA Nsight Developer Tools

• Labs
  • DLIT51143: Master Common Optimization Patterns Efficiently with Nsight Profiling Tools
  • DLIT51202: Debugging and Analyzing Correctness of CUDA Applications
  • DLIT51580: Ray-Tracing Development using NVIDIA Nsight Graphics and NVIDIA Nsight Systems

• Connect with Experts
  • CWES52036: What’s in Your CUDA Toolbox? CUDA Profiling, Optimization, and Debugging Tools for the Latest Architectures
  • CWES52009: Using NVIDIA Developer Tools to Optimize Ray Tracing

• Developer Tools are free and packaged in the latest version of the CUDA Toolkit
  • https://developer.nvidia.com/cuda-downloads

• Support is available via:
  • https://forums.developer.nvidia.com/c/development-tools/

• More information at:
  • https://developer.nvidia.com/tools-overview