NVIDIA Developer Tools
DEVELOPER TOOLS

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

**Correctness Checker:** Compute Sanitizer

```
$ compute-sanitizer --leak-check full memcheck_demo
======= COMPUTE-SANITIZER
Mallocing memory
Running unaligned_kernel: no error
Sync: no error
Running 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 0x40010000 is misaligned
```

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

PROGRAMMING THE NVIDIA PLATFORM
CPU, GPU, and Network

ACCELERATED STANDARD LANGUAGES
ISO C++, ISO Fortran

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

INCIDENTAL PORTABLE OPTIMIZATION
OpenACC, OpenMP

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

PLATFORM SPECIALIZATION
CUDA

__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
NVIDIA HPC SDK
Available at developer.nvidia.com/hpc-sdk, on NGC, via Spack, and in the Cloud

DEVELOPMENT
- Programming Models
  - Standard C++ & Fortran
  - OpenACC & OpenMP
  - CUDA
- Compilers
  - nvcc
  - nvc
  - nvc++
  - nvfortran
- Core Libraries
  -libc++
  - Thrust
  - cuBLAS
  - cuSPARSE
  - CUB
  - cuFFT
  - cuRAND
- Math Libraries
  - cuTENSOR
  - cuSOLVER
- Communication Libraries
  - HPC-X
  - cuBLAS
  - cuTENSOR
  - cuSPARSE
  - cuFFT
  - cuRAND

ANALYSIS
- Profilers
  - Nsight
  - Systems
  - SHARP
  - UCX
  - NVSHMEM
- Debugger
  - cuda-gdb
  - Host
  - Device

Develop for the NVIDIA Platform: GPU, CPU and Interconnect Libraries | Accelerated C++ and Fortran | Directives | CUDA
7-8 Releases Per Year | Freely Available
NSIGHT TOOLS WORKFLOW

Start here

Nsight Systems
Comprehensive system-level performance

Dive into top CUDA kernels by using metrics/counter collection
Re-check overall performance

Dive into graphics frames
Re-check overall performance

Nsight Compute
Detailed CUDA kernel performance

Nsight Graphics
Detailed frame/render performance
Key Features:

- 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
- Command Line, Standalone, IDE Integration

OS: Linux (x86, Power, Arm SBSA, Tegra), Windows, MacOSX (host)

GPUs: Pascal+

READING UTILIZATION

- Zoom in to valleys to find gaps!
ZOOM/FILTER TO EXACT AREAS OF INTEREST
NVTX: NVIDIA TOOLS EXTENSIONS

Code Annotation API
• Useful GPU utilization metrics, but no kernel names / correlation
INTERPRETING GPU SAMPLING METRICS

• **GR Activity** -> GPU is doing work
  - SM, NVENC, NVDEC, Graphics

• **SM Activity** -> Utilizing width of GPU
  - If low, modify kernel grid dimension or increase batch size

• **SM Instruction Issued** -> GPU is performing lots of instructions
  - Stalled waiting on memory?
  - Not enough warps to cover memory latency? Issue larger kernel block dimensions.

• **SM Instructions tensor activity** -> Tensor core utilization
  - Performance up, SM instructions can drop (depending on arch)
  - Can be limited by shared memory, waiting for loads

• Note: Requires disabling **DCGM** and DL built-in profilers
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
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_NVTX_RANGES=1` to disable for production
QUESTIONS TO GUIDE PROFILE ANALYSIS

• What is hot?
  o Can I make it faster, shrink the problem, parallelize it? (Not always...)
  o Reduce precision?

• What is cold?
  o Fill the gaps in the timeline
  o Can I take advantage of unused hardware?
  o Unnecessary dependencies or syncs?
GENERAL OPTIMIZATION TIPS

• Using tensor cores?
  o Minimize conversions/transposes

• Increase grid and batch size to utilize GPUs width

• Conventional parallelism - more worker threads!

• Parallel pipelining
  o No data dependency? Parallelize!
  o Prefetch next batch/iteration during computation

• Can I reorder sooner?
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!
EXPERT SYSTEMS & STATISTICS
BUILT-IN DATA ANALYTICS WITH ADVICE

CUDA Async Memcpy with Pageable Memory
CUDA Synchronous Memcpy
CUDA Synchronous Memset
CUDA Synchronization APIs
CUDA GPU Starvation
CUDA GPU Low Utilization
VULKAN GPU Starvation
VULKAN GPU Low Utilization

The following APIs use PAGEABLE memory which causes asynchronous CUDA memcpy operations to block and be executed synchronously. This leads to low GPU utilization.

Suggestion: If applicable, use PINNED memory instead.

CLI command: nvidia-smi -a | grep "cuda-async-memcpy"
NSIGHT COMPUTE
KERNEL PROFILING TOOL

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, MacOSX (host only)
GPUs: Volta, Turing, Ampere GPUs

Targeted metric sections

Customizable data collection and presentation

Built-in expertise for Guided Analysis and optimization
Source/PTX/SASS analysis and correlation

Source metrics per instruction

Metric heatmap to quickly identify hotspots
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)
STANDALONE SOURCE VIEWER

- View of side-by-side assembly and correlated source code for CUDA kernels
- No profile required
- Open .cubin files directly
- Helps identify compiler optimizations and inefficiencies
OCCUPANCY CALCULATOR
MODEL HARDWARE USAGE AND IDENTIFY LIMITERS

- Model theoretical hardware usage
- Understand limitations from hardware vs. kernel parameters
- Configure model to vary HW and kernel parameters
- Opened from an existing report or as a new activity
HIERARCHICAL ROOFLINE

- Visualize multiple levels of the memory hierarchy
- Identify bottlenecks caused by memory limitations
- Determine how modifying algorithms may (or may not) impact performance
**DEVELOPER TOOLS**

**Debuggers:** cuda-gdb, Nsight Visual Studio 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: no error
Sync: no error
Running 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 0x40010000 is misaligned
```
CUDA GDB
COMMAND LINE AND IDE BACKEND DEBUGGER

- Unified CPU and CUDA Debugging
- CUDA-C/PTX/SASS support
- Built on GDB and uses many of the same CLI commands
Compute sanitizer checks correctness issues via sub-tools:

- **Memcheck** – Memory access error and leak detection
- **Racecheck** – Shared memory data access hazard detection
- **Initcheck** – Uninitialized device global memory access detection
- **Synccheck** – Thread synchronization hazard detection

```bash
$ compute-sanitizer --leak-check full memcheck_demo
##### SANITIZER
Malicious memory
Running unaligned_kernel
Ran unaligned_kernel: no error
Invalid __global__ write of size 4 bytes
  at 0x1b8 in memcheck_demo.cu:6:unaligned_kernel()
  by thread (0,0,0) in block (0,0)
Address 0x7f873cf00001 is misaligned
  Saved host backtrace up to driver entry point at kernel launch time
  Host Frame: [0x20589a]
   in /usr/lib/x86_64-linux-gnu/libcuda.so.1
  Host Frame: [0x8f1a]
   in memcheck_demo
  Host Frame: [0x6028b]
   in memcheck_demo
     Host Frame: [0x8b4eb]
       in memcheck_demo
       Host Frame: _device_stub__Z16unaligned_kernelv(void) [0x9b7f]
         in memcheck_demo
       Host Frame:unaligned_kernel(void) [0xb257]
         in memcheck_demo
       Host Frame:run_unaligned(void) [0xf38]
         in memcheck_demo
       Host Frame:main [0x8b0b1]
         in memcheck_demo
       Host Frame:_libc_start_main [0x82c22]
         in /lib/x86_64-linux-gnu/libc.so.6
       Host Frame:_start [0x8d9e]
         in memcheck_demo
```
Visual Studio Code extensions that provides:

- CUDA code syntax highlighting
- CUDA code completion
- Build warning/errors
- Debug CPU & GPU code
- Remote connection support via SSH
- Available on the VS Code Marketplace now!

NASA GPU Hackathon 2022
Date(s): Sep 19, 2022 - Sep 28, 2022
Event Focus: HPC+AI
North America/Latin America
Applications Closed

CSCS GPU Hackathon 2022
Date(s): Sep 19, 2022 - Sep 29, 2022
Event Focus: HPC+AI
Europe/Middle East/Africa
Applications Closed

NVIDIA/HLRS SciML GPU Bootcamp
Date(s): Oct 24, 2022 - Oct 25, 2022
Event Focus: HPC
Europe/Middle East/Africa

OLCF GPU Hackathon 2022
Date(s): Oct 17, 2022 - Oct 27, 2022
Event Focus: HPC+AI
North America/Latin America
Applications Open

NERSC GPU Hackathon 2022
Date(s): Nov 30, 2022 - Dec 8, 2022
Event Focus: HPC+AI
North America/Latin America
Applications Open
USEFUL LINKS

Web: https://developer.nvidia.com/tools-overview

How to contact us?
   Forums: https://forums.developer.nvidia.com/c/development-tools
   email: devtools-support@nvidia.com

Other digital GTC talks of interest:

   **S21351**: Scaling the Transformer Model Implementation in PyTorch Across Multiple Nodes

   **S21547**: Rebalancing the Load:Profile-Guided Optimization of the NAMD Molecular Dynamics Program for Modern GPUs using Nsight Systems

   S21771: Optimizing CUDA Kernels in HPC Simulation and Visualization Codes using Nsight Compute

   **S21565**: Roofline Performance Model for HPC and Deep-Learning Applications