Faysal Aziz

NVIDIA CUDA AI Acceleration and High-Performance GPU Programming

Building Production-Grade GPU Applications, LLM Inference Pipelines, and Quantized AI Workloads on the NVIDIA Platform - 2 days

FA-0341 · 2 days

Agentic & Generative AI · Software Development · DevOps, Cloud & Infrastructure · Data & Analytics

---

Introduction

Modern AI systems are increasingly constrained not by algorithms but by compute efficiency. Organizations deploying large language models, computer vision systems, recommendation engines, and scientific computing workloads are demanding engineers who can extract maximum performance from NVIDIA GPU infrastructure. CUDA remains the foundation of the NVIDIA accelerated computing ecosystem and serves as the underlying technology powering frameworks such as PyTorch, TensorRT, cuDNN, NCCL, and TensorRT-LLM.

Recent advances in TensorRT, TensorRT-LLM, FP8/FP4 inference, INT8/INT4 quantization, and on-premises AI model deployment have made GPU programming skills even more valuable for engineers building enterprise AI platforms. NVIDIA's latest ecosystem continues to expand support for optimized inference, low-precision computing, and large-scale AI deployment workflows.

This intensive two-day program focuses on practical NVIDIA GPU programming, performance optimization, AI acceleration, and enterprise deployment techniques. The course bridges traditional CUDA programming with modern AI engineering practices, covering GPU architecture, kernel development, memory optimization, profiling, quantization techniques, TensorRT acceleration, and deployment of on-premises large language models.

The instructor brings over 30 years of industry experience and delivers real-world, industry-demanded content rather than purely academic material.

Learning outcomes

Upon completion of this course, participants will be able to:

- Understand NVIDIA GPU architecture and CUDA execution models
- Develop CUDA applications using modern CUDA toolchains
- Write and optimize CUDA kernels for high-performance computing workloads
- Manage GPU memory efficiently using CUDA memory models
- Profile and troubleshoot CUDA applications using NVIDIA performance tools
- Integrate CUDA with Python-based AI and machine learning workflows
- Understand the NVIDIA AI software stack including CUDA, cuDNN, NCCL, TensorRT, and TensorRT-LLM
- Deploy and optimize on-premises AI models on NVIDIA GPUs
- Implement model quantization strategies including FP16, BF16, FP8, INT8, and INT4
- Convert and optimize models using TensorRT and TensorRT-LLM
- Evaluate trade-offs between model accuracy, throughput, latency, and GPU memory consumption
- Build scalable GPU inference environments for enterprise AI workloads
- Optimize LLM inference performance on NVIDIA hardware
- Apply production best practices for AI infrastructure and GPU utilization
- Troubleshoot performance bottlenecks across the CUDA and AI software stack

Prerequisites

Participants should possess:

- Strong Python programming experience
- Strong Linux administration and command-line skills
- Experience with software development workflows and debugging
- Familiarity with Git and source code management
- Basic understanding of machine learning concepts
- Familiarity with PyTorch or equivalent deep learning frameworks
- Understanding of virtualization and Linux server environments
- Basic understanding of containers and Docker

Recommended Hardware Requirements

- Linux workstation or server
- Ubuntu 22.04 LTS or Ubuntu 24.04 LTS
- NVIDIA GPU with CUDA support
- Minimum 32 GB system RAM
- Minimum 12 GB GPU VRAM
- NVIDIA RTX 4060 Ti, RTX 4070, RTX 4080, RTX 4090, RTX 5000 Ada, L40S, A100, H100, H200, or newer supported NVIDIA GPU
- Latest NVIDIA driver compatible with CUDA Toolkit
- Docker Engine and NVIDIA Container Toolkit
- Fully unrestrictedInternet access for software installation and model acquisition

Training outline

1. Introduction to NVIDIA Accelerated Computing

Evolution of GPU Computing

1. From Graphics Processing to General-Purpose Computing
2. NVIDIA Accelerated Computing Ecosystem
3. CUDA in Modern AI and HPC
4. Enterprise Adoption of GPU Computing

Overview of the NVIDIA Software Stack

1. CUDA Toolkit
2. CUDA Runtime and Driver APIs
3. cuBLAS
4. cuDNN
5. NCCL
6. TensorRT
7. TensorRT-LLM

8. NVIDIA NGC Ecosystem
9. NVIDIA AI Enterprise Overview

Modern NVIDIA GPU Architectures

1. Turing Architecture
2. Ampere Architecture
3. Ada Lovelace Architecture
4. Hopper Architecture
5. Blackwell Architecture
6. Tensor Cores
7. Streaming Multiprocessors
8. GPU Memory Hierarchy
9. Interconnect Technologies
10. NVLink Fundamentals

2. CUDA Development Environment

Linux-Based CUDA Development

1. CUDA Toolkit Installation
2. Driver Installation and Validation
3. Compiler Toolchain Overview
4. Environment Configuration
5. Development Workflow Setup

CUDA Project Structure

1. CUDA Source Files
2. Compilation Process
3. Build Automation
4. Dependency Management
5. Debug Builds versus Release Builds

Python Integration with CUDA

1. PyCUDA

2. Numba CUDA
3. CuPy
4. Python-Based GPU Workflows
5. Jupyter Integration

3. CUDA Programming Fundamentals

CUDA Programming Model

1. Host and Device Architecture
2. CPU and GPU Collaboration
3. Thread Hierarchy
4. Blocks
5. Grids
6. Warps
7. Kernel Execution Lifecycle

CUDA Memory Architecture

1. Global Memory
2. Shared Memory
3. Local Memory
4. Constant Memory
5. Texture Memory
6. Unified Memory
7. Memory Access Patterns

CUDA Kernel Development

1. Kernel Definition
2. Thread Indexing
3. Grid Configuration
4. Parallel Computation Design
5. Synchronization Concepts

Data Movement Optimization

1. Host-to-Device Transfers
2. Device-to-Host Transfers
3. Pinned Memory
4. Unified Memory Strategies
5. Asynchronous Transfers
6. Stream-Based Execution

4. CUDA Performance Optimization

GPU Occupancy Fundamentals

1. Occupancy Metrics
2. Resource Utilization
3. Thread Scheduling
4. Warp Efficiency

Memory Optimization Techniques

1. Coalesced Memory Access
2. Shared Memory Optimization
3. Cache Utilization
4. Bandwidth Optimization

CUDA Streams and Concurrency

1. Stream Management
2. Overlapping Computation and Communication
3. Multi-Stream Execution
4. Concurrent Kernel Execution

Profiling and Performance Analysis

1. NVIDIA Nsight Systems
2. NVIDIA Nsight Compute
3. CUDA Profiling Methodology

4. Bottleneck Identification
5. Performance Tuning Workflows

Advanced CUDA Concepts

1. Cooperative Groups
2. CUDA Graphs
3. Dynamic Parallelism
4. Multi-GPU Programming
5. NCCL Fundamentals

5. CUDA for AI and Deep Learning

Deep Learning Acceleration on NVIDIA GPUs

1. CUDA and Deep Learning Frameworks
2. PyTorch CUDA Architecture
3. Tensor Operations
4. Tensor Core Utilization
5. Mixed Precision Computing

Understanding cuDNN

1. Neural Network Primitives
2. Convolution Optimization
3. Training Acceleration
4. Inference Acceleration
5. Framework Integration

Distributed AI Workloads

1. Multi-GPU Training Concepts
2. NCCL Communication
3. Data Parallelism
4. Model Parallelism
5. GPU Resource Management

6. On-Premises AI Model Deployment on NVIDIA Infrastructure

Enterprise AI Deployment Architecture

1. On-Premises AI versus Cloud AI
2. GPU Server Design Considerations
3. Workstation-Based AI Deployments
4. AI Infrastructure Sizing
5. Capacity Planning

Open-Source Model Ecosystem

1. Foundation Models
2. Large Language Models
3. Embedding Models
4. Vision Models
5. Multimodal Models

Local Model Serving Platforms

1. NVIDIA NIM Concepts
2. vLLM Fundamentals
3. TensorRT-LLM Deployment Concepts
4. Containerized Inference Environments
5. GPU Resource Allocation

AI Inference Workflows

1. Model Acquisition
2. Model Validation
3. Benchmarking Methodologies
4. Throughput Optimization
5. Latency Optimization

7. Model Quantization and AI Optimization

Fundamentals of Quantization

1. Quantization Objectives
2. Accuracy versus Performance Trade-Offs
3. Memory Reduction Strategies
4. Throughput Improvements

Precision Formats

1. FP32
2. TF32
3. FP16
4. BF16
5. FP8
6. FP4
7. INT8
8. INT4

Quantization Techniques

1. Post-Training Quantization
2. Quantization-Aware Training
3. Weight-Only Quantization
4. Activation Quantization
5. Calibration Workflows

Large Language Model Quantization

1. LLM Compression Strategies
2. Memory Footprint Reduction
3. Quantization Evaluation Metrics
4. Accuracy Validation Approaches
5. Enterprise Deployment Considerations

Modern NVIDIA Quantization Ecosystem

1. TensorRT Model Optimizer
2. TensorRT Quantization Capabilities
3. TensorRT-LLM Quantization Features
4. AWQ Concepts
5. Low-Precision Inference Optimization
6. Production Deployment Considerations

8. TensorRT and TensorRT-LLM Optimization

TensorRT Architecture

1. Inference Engine Concepts
2. Optimization Pipeline
3. Engine Building Process
4. Runtime Components

Model Conversion Workflows

1. PyTorch Export
2. ONNX Integration
3. TensorRT Engine Generation
4. Validation Procedures

TensorRT Optimization Techniques

1. Layer Fusion
2. Kernel Selection
3. Dynamic Shapes
4. Mixed Precision Execution
5. Memory Optimization

TensorRT-LLM Fundamentals

1. LLM-Specific Optimization Techniques
2. KV Cache Management

3. Attention Optimization
4. Tensor Parallelism
5. Pipeline Parallelism

Performance Benchmarking

1. Tokens per Second Analysis
2. Throughput Measurements
3. Latency Measurements
4. GPU Utilization Metrics
5. Capacity Planning

9. Production AI Engineering on NVIDIA Platforms

Building AI Inference Pipelines

1. Model Lifecycle Management
2. Artifact Management
3. Containerization Strategies
4. Deployment Automation

Monitoring and Observability

1. GPU Telemetry
2. Performance Monitoring
3. Resource Utilization Tracking
4. Capacity Forecasting

Operational Best Practices

1. GPU Infrastructure Management
2. Environment Standardization
3. Version Control Strategies
4. Security Considerations
5. High Availability Concepts

Troubleshooting and Performance Tuning

1. CUDA Runtime Issues
2. Driver Compatibility Issues
3. Memory Bottlenecks
4. Quantization Accuracy Issues
5. TensorRT Deployment Challenges
6. Large Model Inference Troubleshooting

Future Directions in NVIDIA Accelerated Computing

1. Emerging CUDA Features
2. AI-Centric GPU Architectures
3. Next-Generation Tensor Core Capabilities
4. Evolving Quantization Standards
5. Enterprise AI Deployment Trends

Disclaimer

This course outline is intended as a high-level curriculum framework and planning guide. The actual delivery sequence, depth of coverage, hands-on exercises, software versions, tools, technologies, and topics may be modified, expanded, condensed, substituted, or reordered by the instructor based on participant profiles, organizational requirements, available infrastructure, technological developments, and training objectives. The trainer reserves the right to amend the course content and delivery approach at any time without prior notice in order to maintain technical relevance and instructional effectiveness.

© 2026 Faysal Aziz · Kuala Lumpur