

CS599: Programming Massively Parallel Processors & Heterogeneous Systems

**Understanding and programming the devices powering AI
Lecture 0: Class Overview and Introductions**

Jonathan Appavoo email: jappavoo @ b u . e d u office: CDS 706

~~CS599: Programming Massively Parallel Processors & Heterogeneous Systems~~

~~Understanding and programming the devices powering AI~~

~~Lecture 0: Class Overview and Introductions~~

CUDA Programming on GPU's

Jonathan Appavoo email: jappavoo @ b u . e d u office: CDS 706

CS599

**Course Synopsis
(See Syllabus Piazza for details)**

Course Prerequisites and Structure

- Prerequisites
 - Basic understanding of the C programming language (especially its use of pointers)
 - Undergrad level knowledge of computer architecture
- Structure
 - Lectures twice a week for ~6-8 weeks
 - In class project Q&A, hacking and discussions remainder of semester
 - Office hours as needed

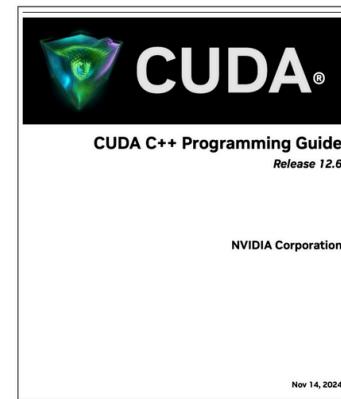
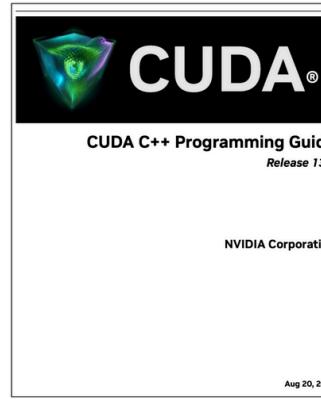
Course Components

- 12-16 Lectures
- Programming Assignments: 2
- Paper readings: ~5
- Midterm exam
- **Group Project**

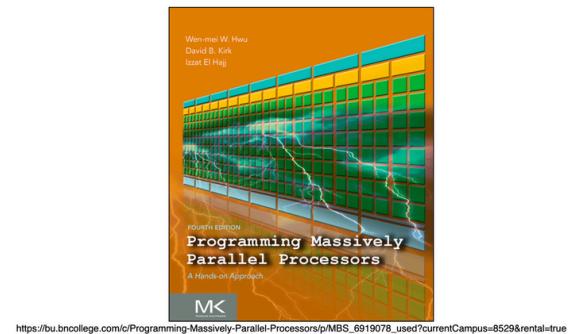
Course Resources

- Primary source of information (besides slides):
NVIDIA Inc, "CUDA C++ Programming Guide"
(and friends: best practices, API references, white papers, etc)
- Textbook: none
 - but if you need one Kirk and Hwu:
"Programming Massively Parallel Processors: A Hands-on Approach (4th edition)" is reasonable.
- Piazza
- Github Classroom
- Gradescope
- MOC/NERC/Red Hat Open AI
 - GPUS: V100, A100 and H100
 - CUDA: 12.6.3

<https://docs.nvidia.com/cuda/#programming-guides>



https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf https://docs.nvidia.com/cuda/archive/12.6.3/pdf/CUDA_C_Programming_Guide.pdf



Grade break down

- Programming Assignments:
 - $2 \times 10\% = 20\%$
- Midterm:
 - **Oct 28th:** 30%
- Group Project
 - 50 %
 - 3-4 Students
 - Proposal, Midway Review, Report and Presentation

Syllabus

the details, including a tentative weekly calendar

Please read and ask any questions you have on Piazza

CAS CS 599 - Fall 2025: Programming Massively Parallel Multiprocessors and Heterogeneous Systems (Understanding and programming the devices powering AI)

Piazza: <https://piazza.com/class/me4rjds6oce507>
Programming/GPU environment: <https://rhods-dashboard-redhat-ods-applications.apps.edu.nerc.mghpc.org/>
Gradescope: <https://www.gradescope.com/courses/1078574>
Lectures: Tuesday and Thursday 12:30PM-1:45PM CDS 701
Staff: Instructor: Professor Appavoo (he/him) BU email: jappavoo
Office hours: Location: CDS 706 Times: Tuesdays 3:30-5:00 pm and Thursday 3:30-5:00 pm
Midterms: 75 minute in-class midterm will be held on Oct, 28 2025.

Course Description: After decades, we have reached an era in which the use of Massively Parallel MultiProcessors (MPP) has become commonplace. With the advent of Graphical Processing Units (GPUs), access to MPP's is no longer restricted to domain scientists conducting High Performance Computing on Super-Computers.

Today, most computer systems are heterogeneous. Central Processing Units (CPUs) and GPUs thoughts of high-bandwidth memory. Parallel algorithms have shown up to 100x speedup over similar single-core CPU algorithms, signal processing, financial modeling, neural networks, and more. The widespread access and availability of heterogeneous computing is driving the AI revolution.

While the CPU's familiar von Neumann architecture is well suited for general-purpose computing, adding the Data Parallel architecture of GPU has proven critical in providing the raw computation power needed for AI. Understanding the heterogeneous combination of these two architectures is the focus of this class.

The course covers "general purpose" — i.e., not just computation in a homogeneous system. The course introduces parallel computing, including its programming model and syntax. We will learn how to program parallel algorithms, CUDA, OpenCL, and MPI.

The focus of the course will be on performance. Programs running on GPUs are often used as accelerators rather than they would on a multi-core CPU. Specifically, the course will require you to evaluate the various micro-architectural techniques used by GPUs to achieve their performance. The course is not about getting applications run on GPUs, but rather about gaining a fundamental understanding of how they work and what factors affect their performance.

Acknowledgments: This class borrows heavily from the University of Toronto offering of ECE1782H "Programming Massively Parallel Systems". Thank you, Dr Stumm.

7 Detailed Syllabus Calendar

The following is the tentative calendar changes and updates will be posted on Piazza.

Date	Activity/Topics	Assignment/Due Dates
Tue 09/02/25	Class Overview and Introductions	
Thu 09/04/25	Motivation and Challenges (1:17)	
Tue 09/09/25	GPU architecture (0:47)	
Thu 09/11/25	Introduction to CUDA programming I (1:19)	
Tue 09/16/25	Introduction to CUDA programming II (1:11)	

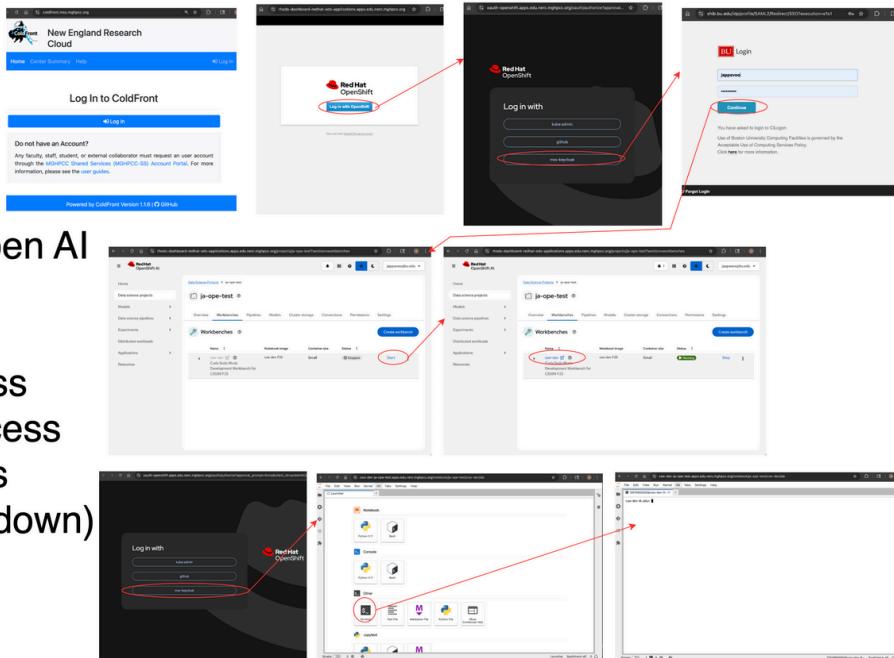
8

Tue 09/18/25	Introduction to CUDA programming III (00:54)	Assignment 1 out
Tue 09/23/25	Optimizing CUDA Programs I (1:30)	
Thu 09/25/25	Optimizing CUDA Programs II(00:58)	
Tue 09/30/25	Virtual Memory (00:41)	DUE: Assignment 1
Thu 10/02/25	Some Applications Part I – Optimizing Scan (1:02)	
Mon 10/06/25	Last Day to Drop Standard Courses (without a "W" grade); Last Day for Graduate Students to Change Standard Courses from Credit to Audit Status	
Tue 10/07/25	Some Applications Part II – CNN and Tensors (00:50)	

GPU / Lab Infrastructure

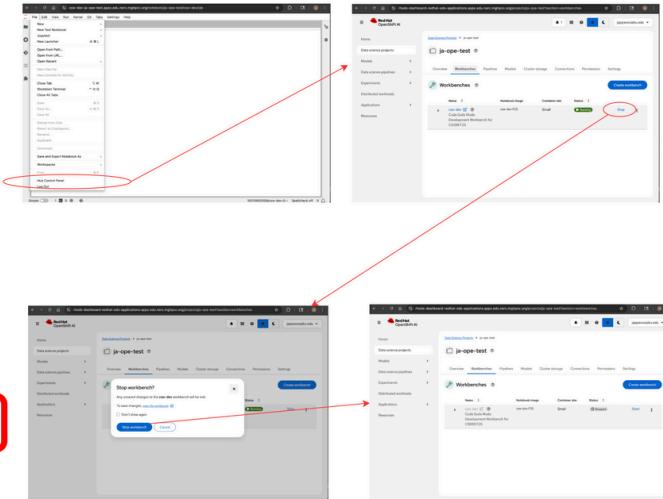
Containerized CUDA + Batch GPUs

- Getting Started
 - ColdFront
 - OpenShift and Red Hat Open AI
- Dev Container
 - RHEL 9 + CUDA 12.6.3
 - Browser Jupyter Lab Access
 - Terminal Remote Shell Access
 - Persistent home directories
 - Idle culler (but please shutdown)
- Batch Queues
 - bqstat, bjobs, brun,



Containerized CUDA + Batch GPUs

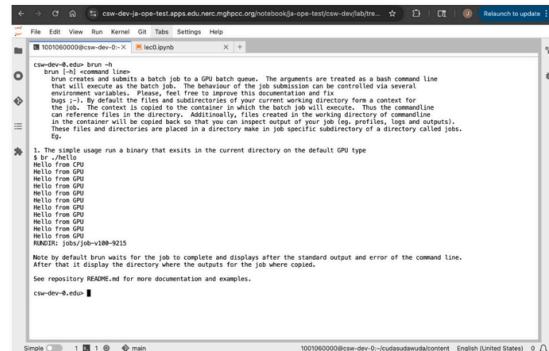
- Getting Started
 - ColdFront
 - OpenShift and Red Hat Open AI
- Dev Container
 - RHEL 9 + CUDA 12.6.3
 - Browser Jupyter Lab Access
 - Terminal Remote Shell Access
 - Persistent home directories
 - Idle culler **(but please shutdown)**
- Batch Queues
 - bqstat, bjobs, brun,



Batch Queues

<https://github.com/jappavoo/batchtools>

- **brun**: run a binary on a GPU
"node"
- **./jobs**: outputs from run/job
- **bjobs**: current jobs
- **bqstat**: status of queues
- **bps**: list activity on GPU nodes
(broken)
- misc: **blog**, **bpods**, **bwait**

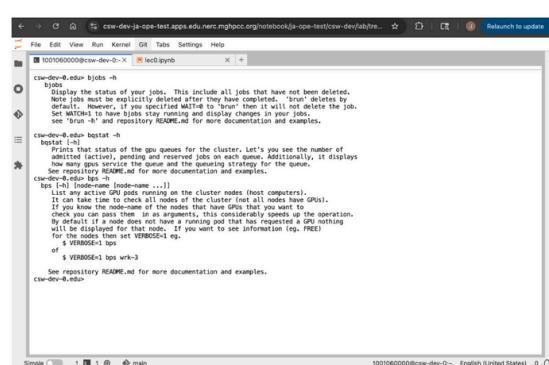


```
File Edit View Run Kernel Git Tab Settings Help
1001060000@cse-dev-0-[lect5ygnb] x + [Return to update]
cse-dev-0:~$ brun -n 10 node
Hello from GPU
RANK=0: job[0]:100-9215
```

Note: This command waits for the job to complete and displays after the standard output and error of the command line. After that it displays the directory where the outputs for the job were copied.

See repository README.md for more documentation and examples.

cse-dev-0:~\$



```
File Edit View Run Kernel Git Tab Settings Help
1001060000@cse-dev-0-[lect5ygnb] x + [Return to update]
cse-dev-0:~$ bjobs -h
Usage: bjobs [-h] [-l] [-d] [-r] [-n N] [-t T]
Display the status of your jobs. This include all jobs that have not been deleted, pending or reserved. By default, it will display the status of the current job. If you specify -l then it will list pending and reserved jobs. If you specify -d then it will list deleted jobs. If you specify -r then it will resume the job. Set the environment variable BPS to 1 to print detailed information about the job. See "brun -h" and repository README.md for more documentation and examples.
```

cse-dev-0:~\$ bps -h
Usage: bps [-h] [-l] [-t T]
Prints the status of the gpu queues for the cluster. Let's you see the number of submitted (submitted), pending (pending) and reserved (reserved) jobs on each queue. Additionally, it displays the gpu serial number and the host computer for each queue. By default, it will display the status of the current queue. If you specify -l then the status of all queues will be displayed for that node. If you want to see information (e.g. FREE) for a specific queue, set VERBOSER=1.
\$ VERBOSE=1 bps wr-3
\$ VERBOSE=1 bps wr-3
See repository README.md for more documentation and examples.

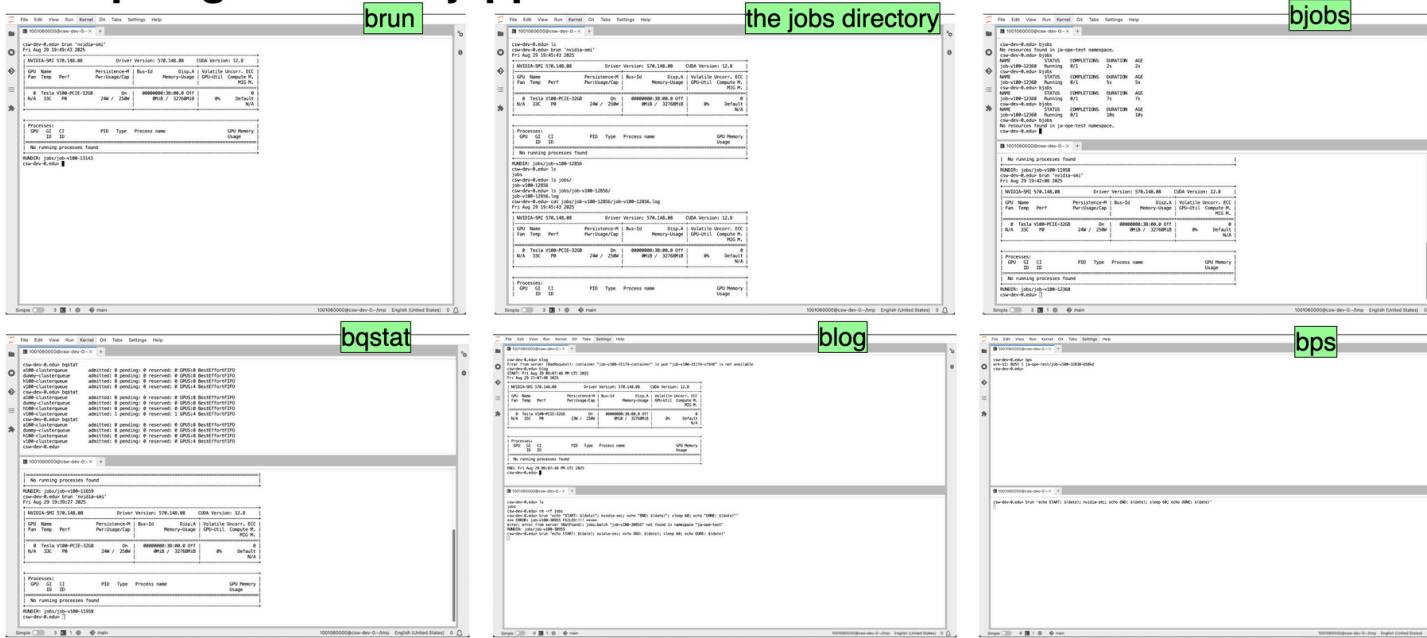
cse-dev-0:~\$

Please help me fix bugs

Please help me fix bugs

Batch Queues

<https://github.com/jappavoo/batchtools>



Demo

Please confirm that you can log in and run "nvidia-smi".

Terminal Interface

Alternative to using browser

- Install OpenShift CLI client (oc)
 - <https://console.apps.edu.nerc.mghpcc.org/command-line-tools>
- Get and execute login command
 - <https://oauth.openshift.apps.edu.nerc.mghpcc.org/oauth/token/display>
- Start csw-dev workbench via Red Hat OpenShift AI
- Use 'oc rsh' and friends
 - see my csw script for an example of how I start shells sessions



```
jappavoo - jappavoo@cs2:~/Work/kite - oc + csw - 106x25
kerfuffle:- jappavoo$ cat ~/bin/csw
#!/bin/bash

container=csw-dev
pods=$(oc get pods -o name | grep "/${container}"))

pod=${pods[0]}

[[ -z $pod ]] && {
    echo "Can't find a pod with $container in its name. Have you started the notebook?" > /dev/stderr
    exit -1
}

pod=${pod##*/}

if [[ $# == 0 ]]; then
    oc rsh -c $container $pod /bin/bash -l
else
    oc rsh -c $container $pod $@
fi

kerfuffle:- jappavoo$ csw
| (app-root) csw-dev-0.edu>
```

Concurrency, parallelism, scalability, me and you

Our backgrounds, our understanding and why we are here

Some Ice Breakers

Well for geeks anyway

- What does "Parallel Computing" make you think/feel?
- How many cores does your laptop have?
- What is the largest number of cores that a system you have worked on has had?
- What is the largest number of threads you have explicitly created in a program?
- Have you written a pthreaded application? How long did you debug it?
- What are your CS interests?
- Why brings you to this class?

