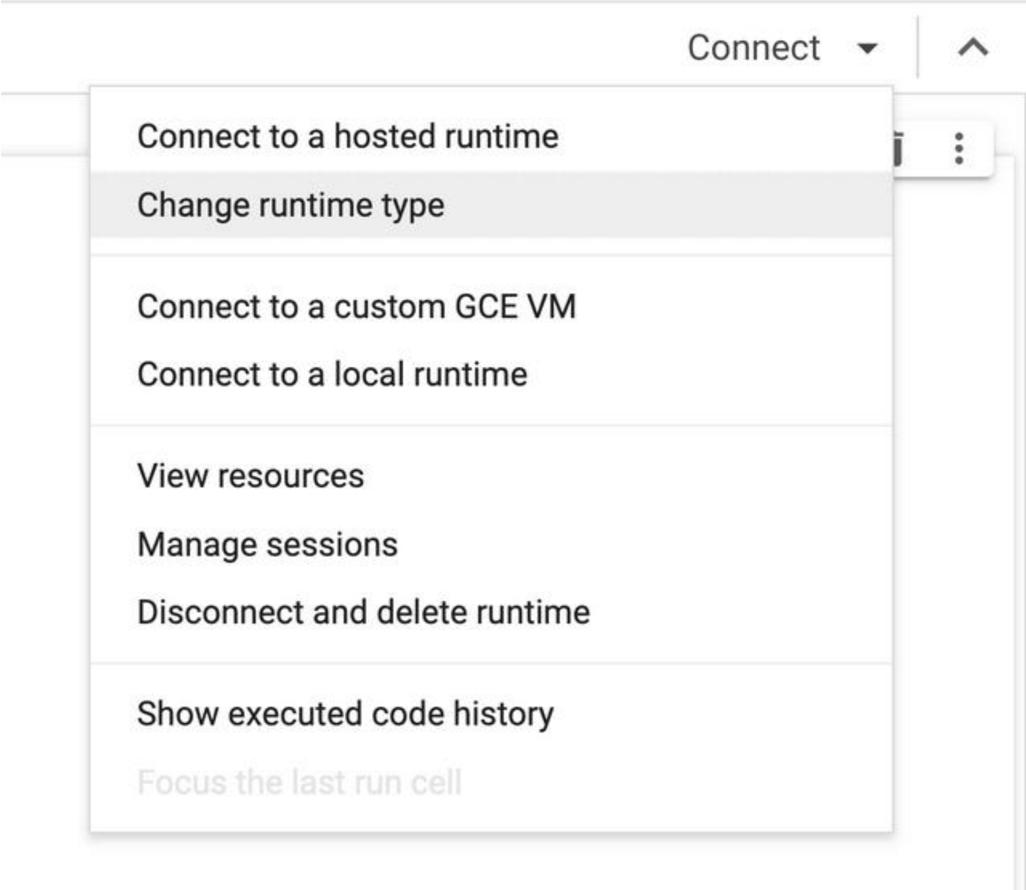
EECS E6893 Big Data Analytics
GPU Programming

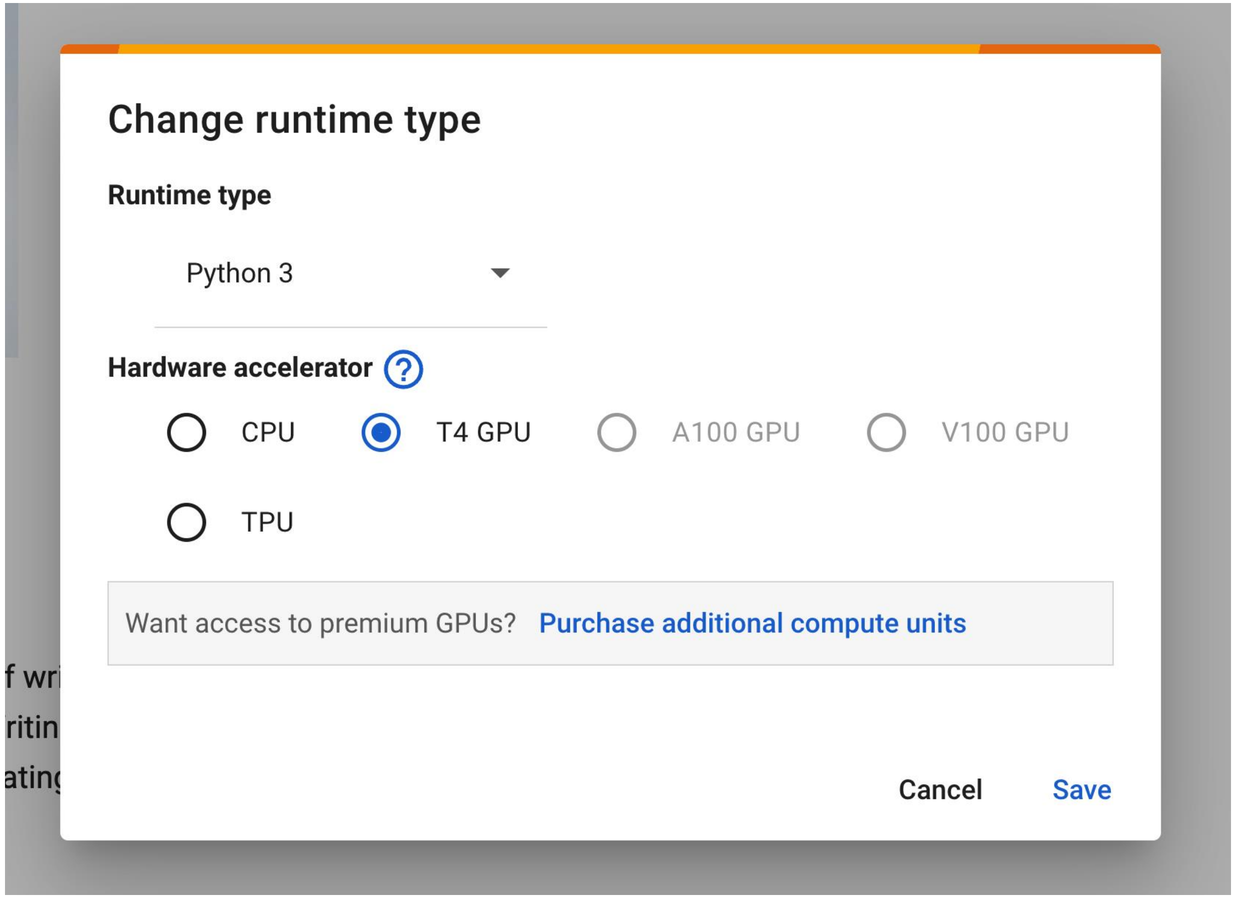
Linyang He, linyang.he@columbia.edu

Outline

- GPU vs. CPU
- Overview of CUDA
- Numba
- GPU Programming in Python

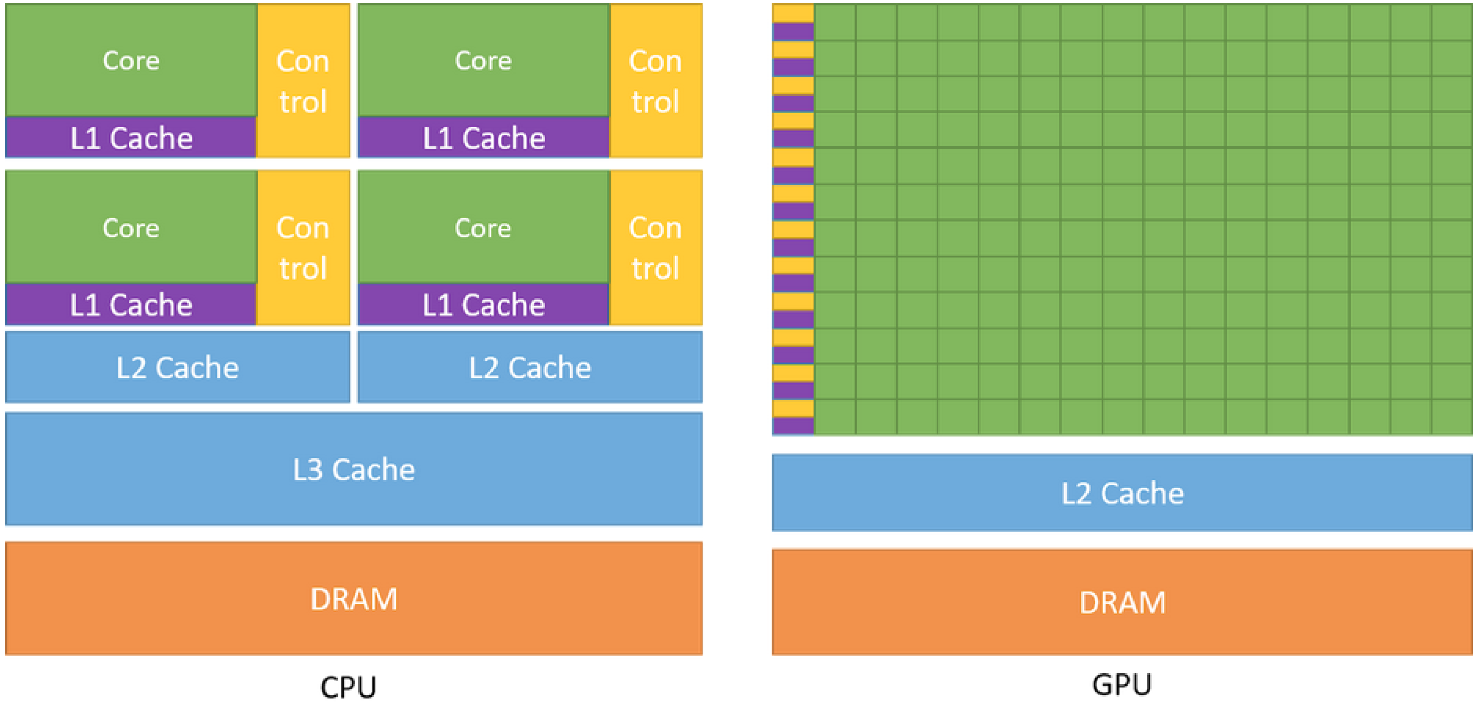
GPU on Colab


Connect
Connect to a hosted runtime
Change runtime type
Connect to a custom GCE VM
Connect to a local runtime
View resources
Manage sessions
Disconnect and delete runtime
Show executed code history
Focus the last run cell


Change runtime type
Runtime type
Python 3
Hardware accelerator
CPU
T4 GPU
A100 GPU
V100 GPU
TPU
Want access to premium GPUs? Purchase additional compute units
Cancel
Save

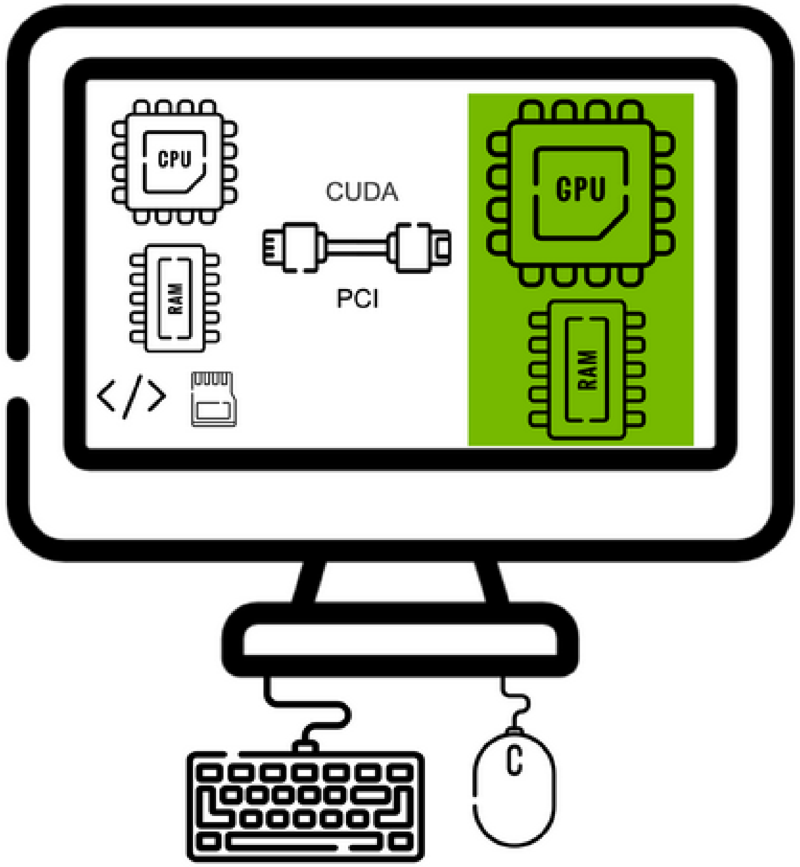
CPU vs GPU

- CPUs execute tasks sequentially
- GPUs execute tasks in parallel.



CUDA

- CUDA is an extension to C++ which allows us to compile GPU code and interact with the GPU.
- Nvidia has invested in bring CUDA functionality to Python
- A lot of Python CUDA packages
  - Numba
  - CuPy
  - cuDF
  - cuML
  - cuGraph



Numba

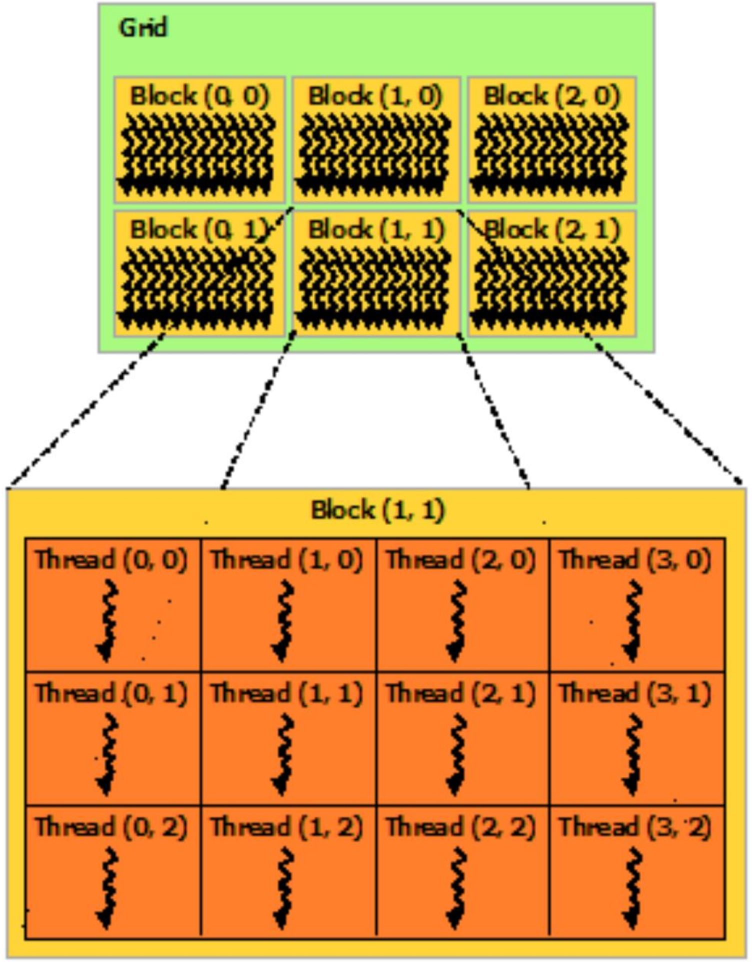
- Numba is an open source JIT (just in time) compiler translating Numpy code into fast machine code.
- Kernels written in Numba have direct access to NumPy arrays.
- NumPy arrays are transferred between the CPU and the GPU automatically

Kernel

- A kernel is similar to a function, it is a block of code which takes some inputs and is executed by a processor.
- The difference between a function and a kernel is:
  - A kernel cannot return anything, it must instead modify memory
  - A kernel must specify its thread hierarchy (threads and blocks)

Grids, threads and blocks

- Threads and blocks are how you instruct you GPU to process some code in parallel.
- We need to specify the number of times we want our kernel to be called.
  - number of threads should be a multiple of 32 (warp size)
  - a warp is a minimal execute unit in CUDA, containing 32 threads
  - start from 128-512 threads per block



HW4 Part 2

- Go through Numba tutorials provided in the assignment.
- Finish two labs.