Programming Languages Research Overview

Stefan Muller

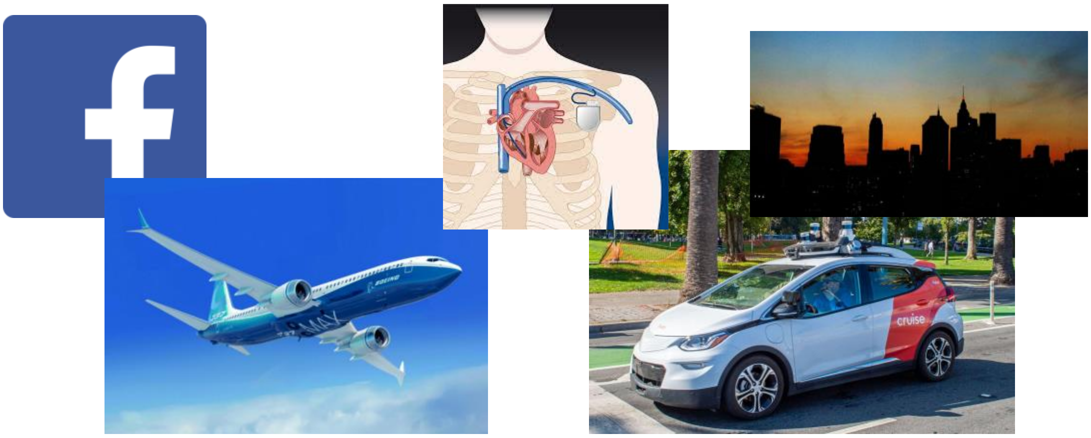
CS100

Oct. 8, 2021

Short Bio

- PhD in Computer Science from Carnegie Mellon in 2018
- Postdoc at Carnegie Mellon, 2018-2020
- Joined IIT in 2020 as Assistant Professor of Computer Science
- Research interests:
  - Language and type system design
  - Static resource analysis
  - Parallel computing

We’re more and more dependent on software

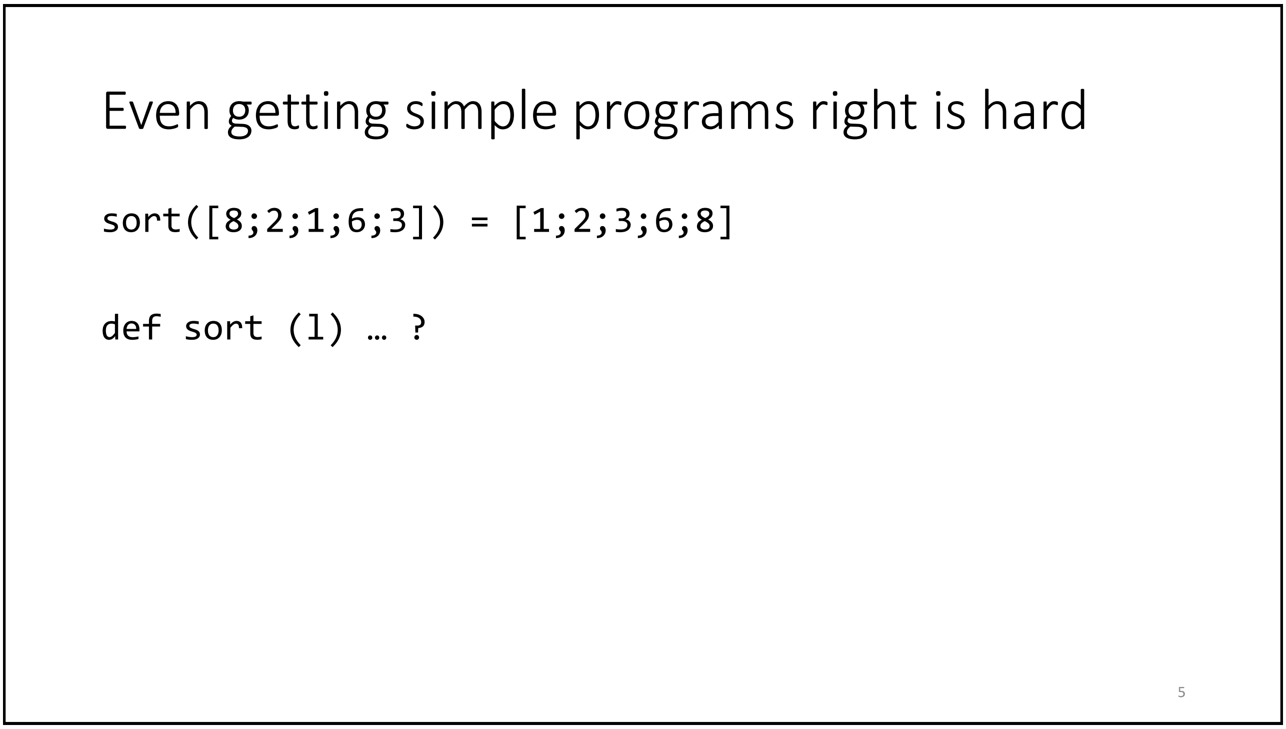


… and people are bad at writing software

especially

large
concurrent
interactive
time-sensitive
multi-language
…

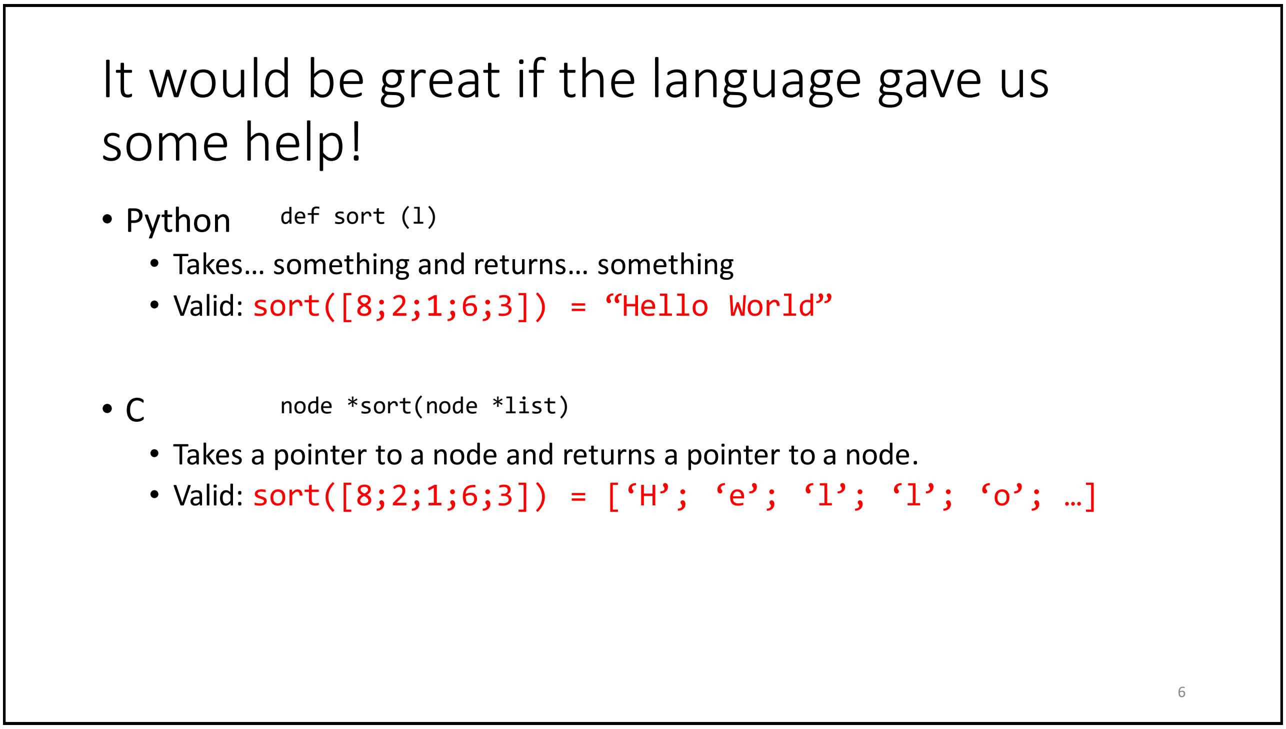
software



Even getting simple programs right is hard

```
sort([8;2;1;6;3]) = [1;2;3;6;8]

def sort (l) … ?
```



It would be great if the language gave us some help!

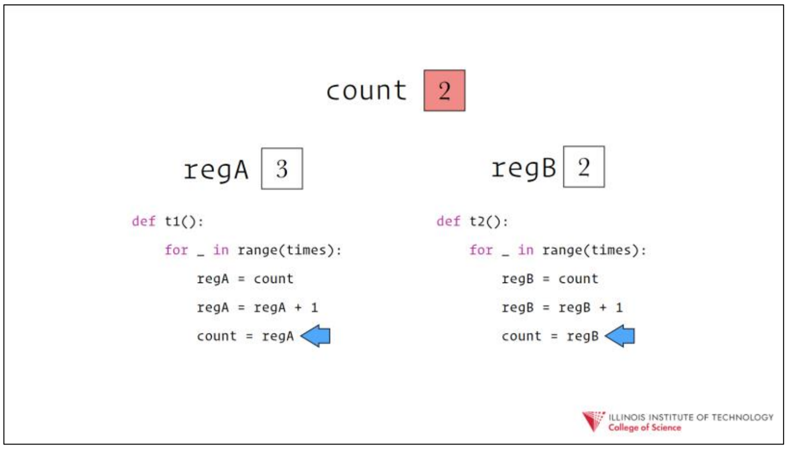
- Python `def sort (l)`
  - Takes… something and returns… something
  - Valid: `sort([8;2;1;6;3]) = “Hello World”`
- C `node *sort(node *list)`
  - Takes a pointer to a node and returns a pointer to a node.
  - Valid: `sort([8;2;1;6;3]) = [‘H’; ‘e’; ‘l’; ‘l’; ‘o’; …]`

It would be great if the language gave us some help!

- OCaml `sort : int list -> int list`
  - Takes an integer list and returns an integer list.
  - Valid: `sort([8;2;1;6;3]) = [8;2;1;6;3]`
  - Valid: `sort([8;2;1;6;3]) = [10;11;12]`

- Coq
  ```
  sort : forall (l1 : list int), exists (l2: list int),
                  Sorted l2 /\ Permutation l1 l2
  ```
  - Takes an integer list and returns a sorted permutation of it.
  - Valid: `sort([8;2;1;6;3]) = [1;2;3;6;8]`
  - … and nothing else

Concurrency adds more complexity



```
def t1():
    for _ in range(times):
        regA = count
        regA = regA + 1
        count = regA
```

```
def t2():
    for _ in range(times):
        regB = count
        regB = regB + 1
        count = regB
```

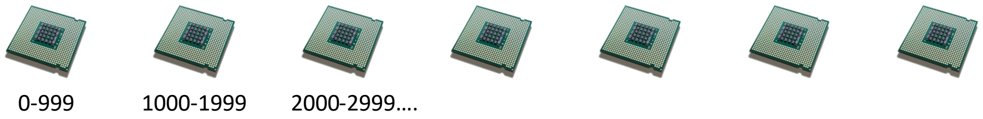
A race condition!

CS100, 9/17

Minimizing shared state helps

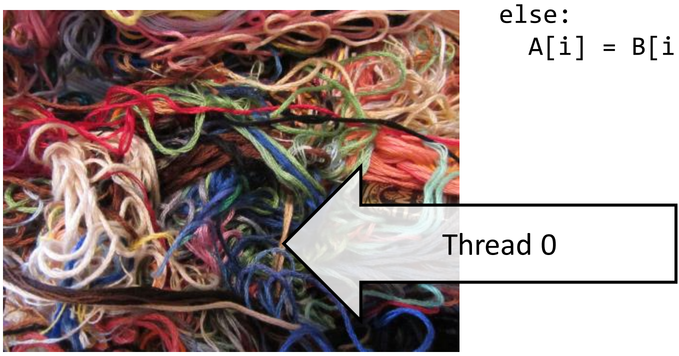
- Many languages based around highly parallel, no-shared-state applications

```
parallel for i in range(1000000):
  A[i] = B[i] + C[i]
```



Minimizing shared state helps ... to a point

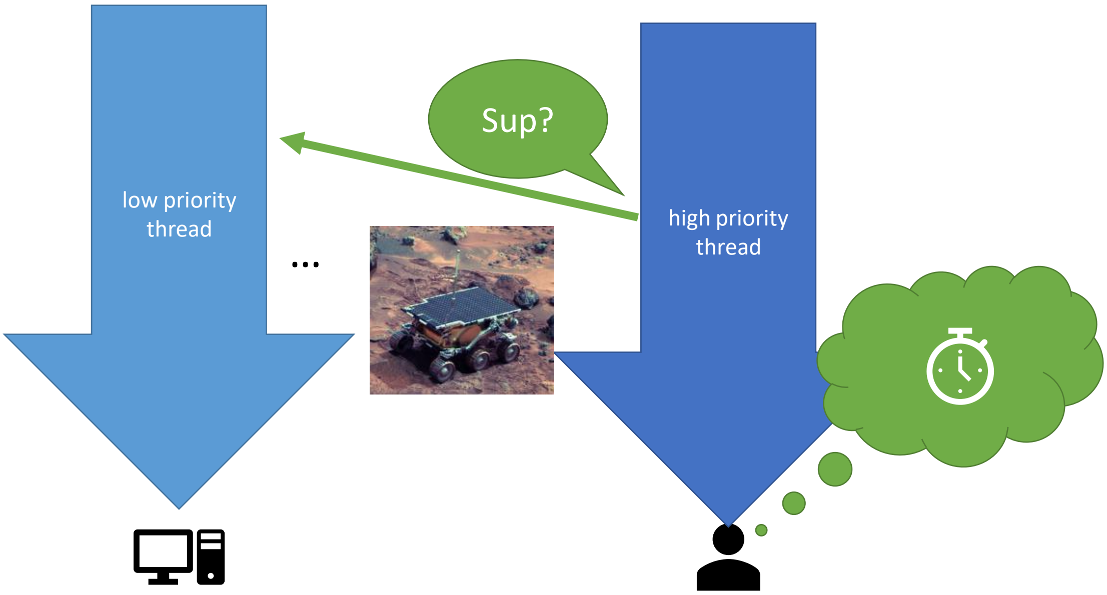
```
parallel for i in range(1000000):
  if (i == 0):
     name = input(“What is your name?”)
     print(“Hello, ” + name)
  else:
    A[i] = B[i] + C[i]
```



Solution: thread priorities

```
priority High
priority Low
order Low < High

parallel at High:
  name = input(“What is your name?”)
  print(“Hello, ” + name)
parallel at Low:
  parallel for i in range(1, 1000000):
    A[i] = B[i] + C[i]
```



It would be great if the language gave us some help!

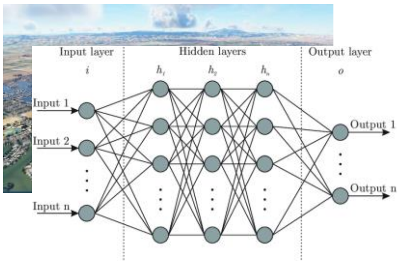
- PriML (PhD Dissertation, ongoing work)
  - Prevent priority inversions **at compile time**

```
constraint violated at example.prm:5.1-5.8 :  high  <=  low
Type error: constraint violated
```

Current research projects

- PriML: parallel interactive programs
- RaCUDA: Resource-aware CUDA
  - How long will your GPU kernel take to run?
- SEEr: Scalable, Energy-Efficient HPC environment for AI-enabled science
  - (with Zhiling)

RaCUDA: Resource-aware CUDA



GPGPU: General-purpose programming for GPUs
CUDA: Extension of C for GPGPU programming

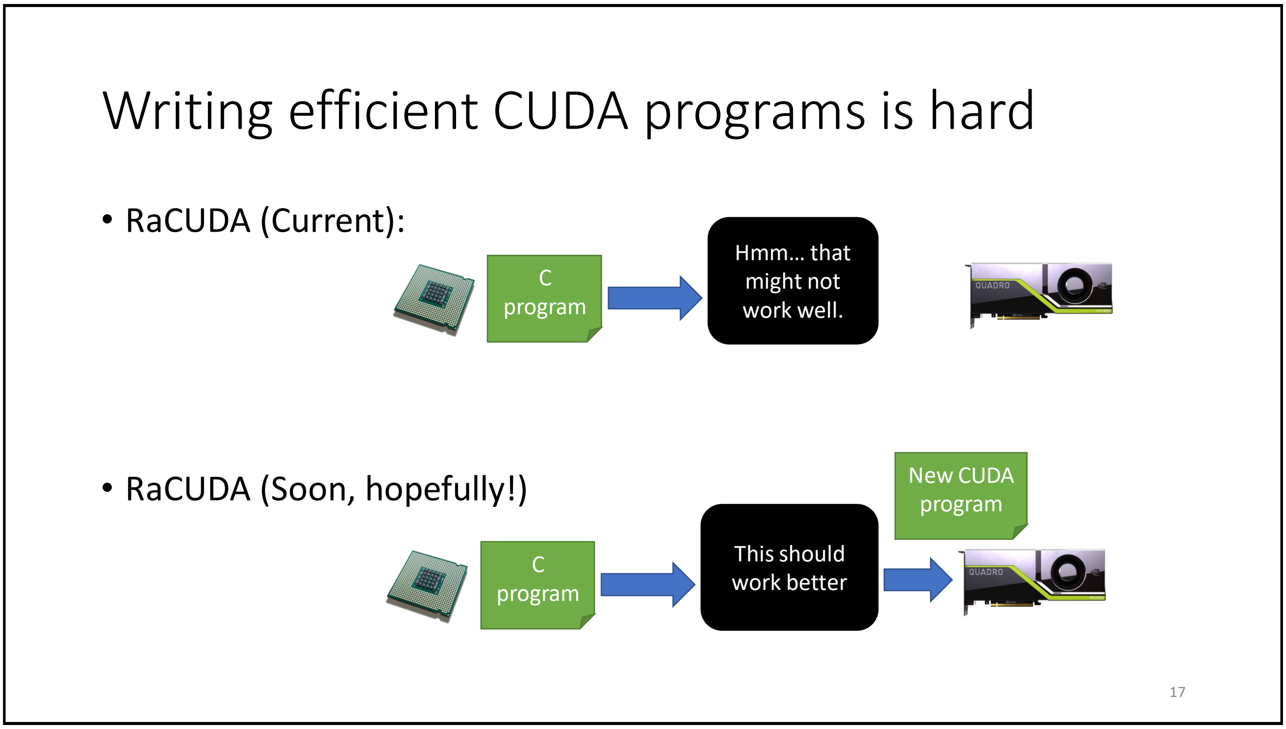
Writing efficient CUDA programs is hard

- CUDA (Idea):

C programmer
C
program
QUADRO

- CUDA (Practice)

C
program
ADRO



Writing efficient CUDA programs is hard

- RaCUDA (Current):

C program

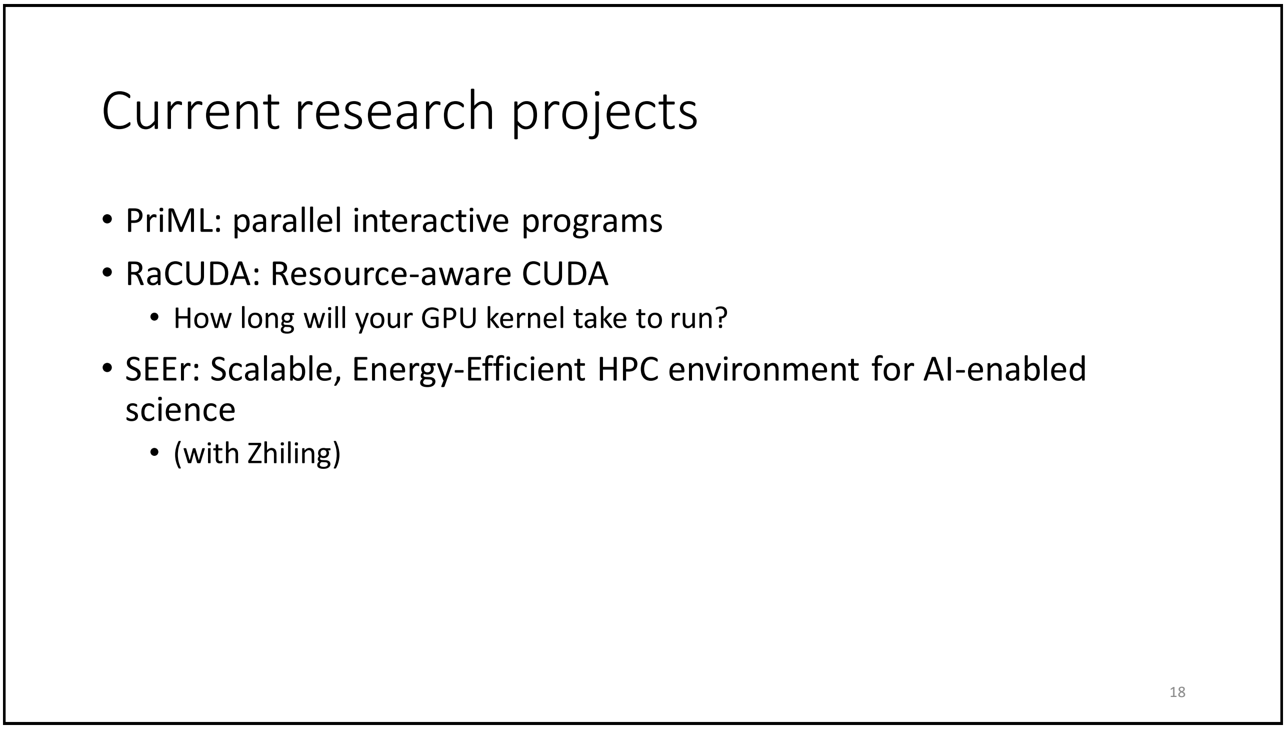
Hmm… that might not work well.

- RaCUDA (Soon, hopefully!)

C program

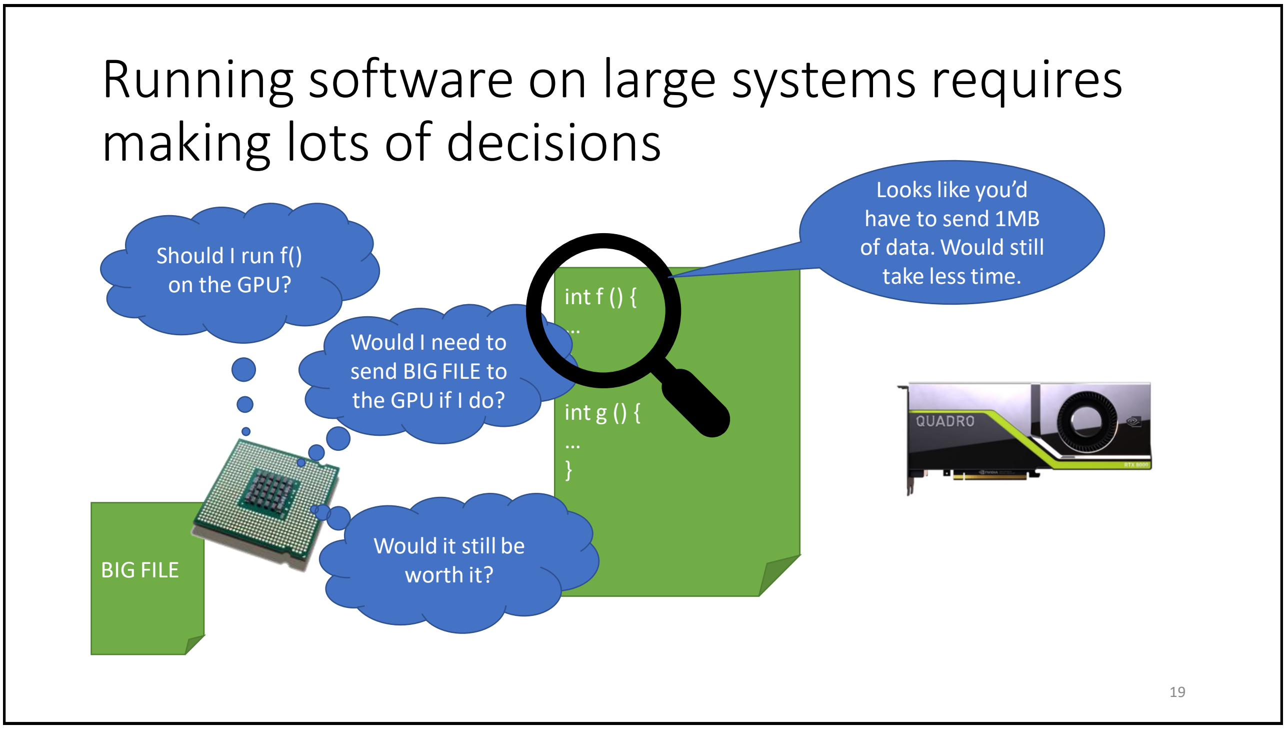
This should work better

New CUDA program



Current research projects

- PriML: parallel interactive programs
- RaCUDA: Resource-aware CUDA
  - How long will your GPU kernel take to run?
- SEEr: Scalable, Energy-Efficient HPC environment for AI-enabled science
  - (with Zhiling)



Running software on large systems requires making lots of decisions

Should I run f() on the GPU?

Would I need to send BIG FILE to the GPU if I do?

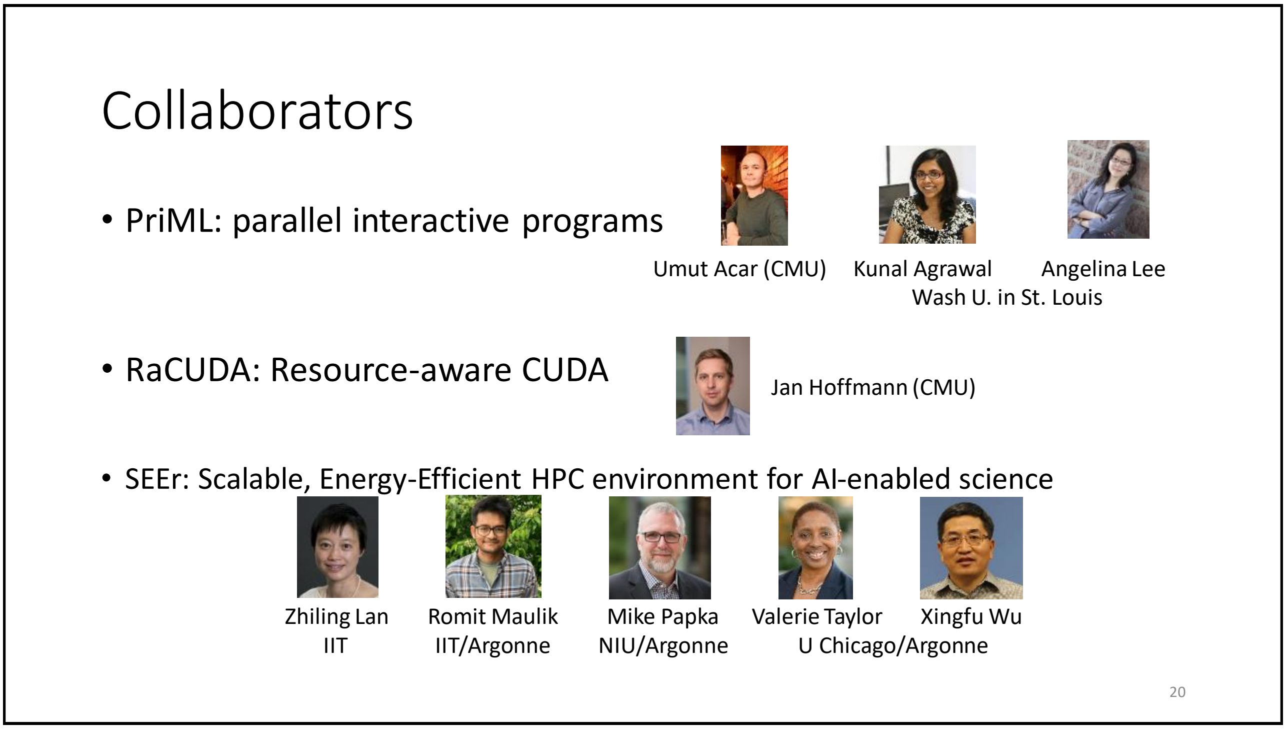
Would it still be worth it?

BIG FILE

int f () {
…

int g () {
…
}

Looks like you'd have to send 1MB of data. Would still take less time.



Collaborators

- PriML: parallel interactive programs

Umut Acar (CMU) Kunal Agrawal Angelina Lee
Wash U. in St. Louis

- RaCUDA: Resource-aware CUDA

Jan Hoffmann (CMU)

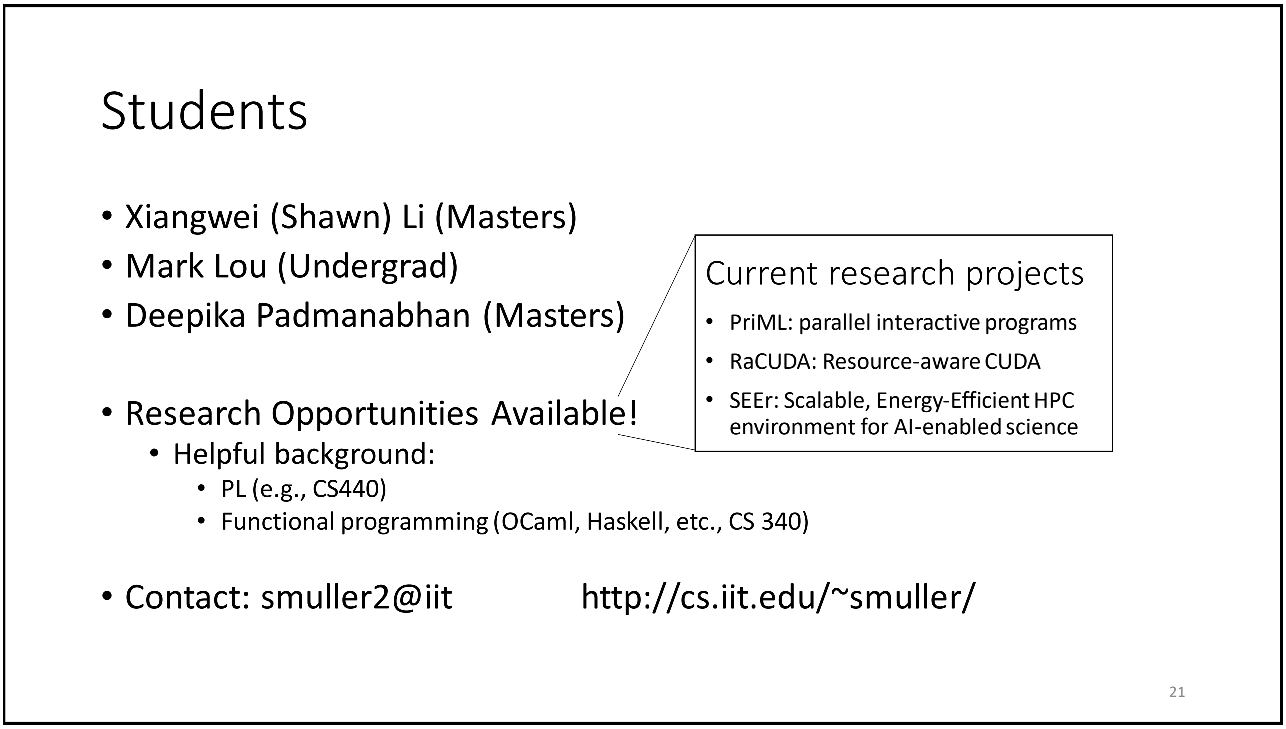
- SEEr: Scalable, Energy-Efficient HPC environment for AI-enabled science

Zhiling Lan
IIT

Romit Maulik
IIT/Argonne

Mike Papka
NIU/Argonne

Valerie Taylor Xingfu Wu
U Chicago/Argonne



Students

- Xiangwei (Shawn) Li (Masters)
- Mark Lou (Undergrad)
- Deepika Padmanabhan (Masters)

- Research Opportunities Available!
  - Helpful background:
    - PL (e.g., CS440)
    - Functional programming (OCaml, Haskell, etc., CS 340)

Current research projects

- PriML: parallel interactive programs
- RaCUDA: Resource-aware CUDA
- SEEr: Scalable, Energy-Efficient HPC environment for AI-enabled science

- Contact: smuller2@iit http://cs.iit.edu/~smuller/