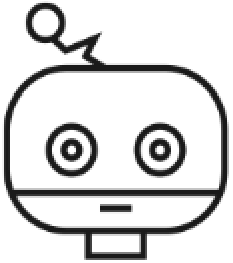
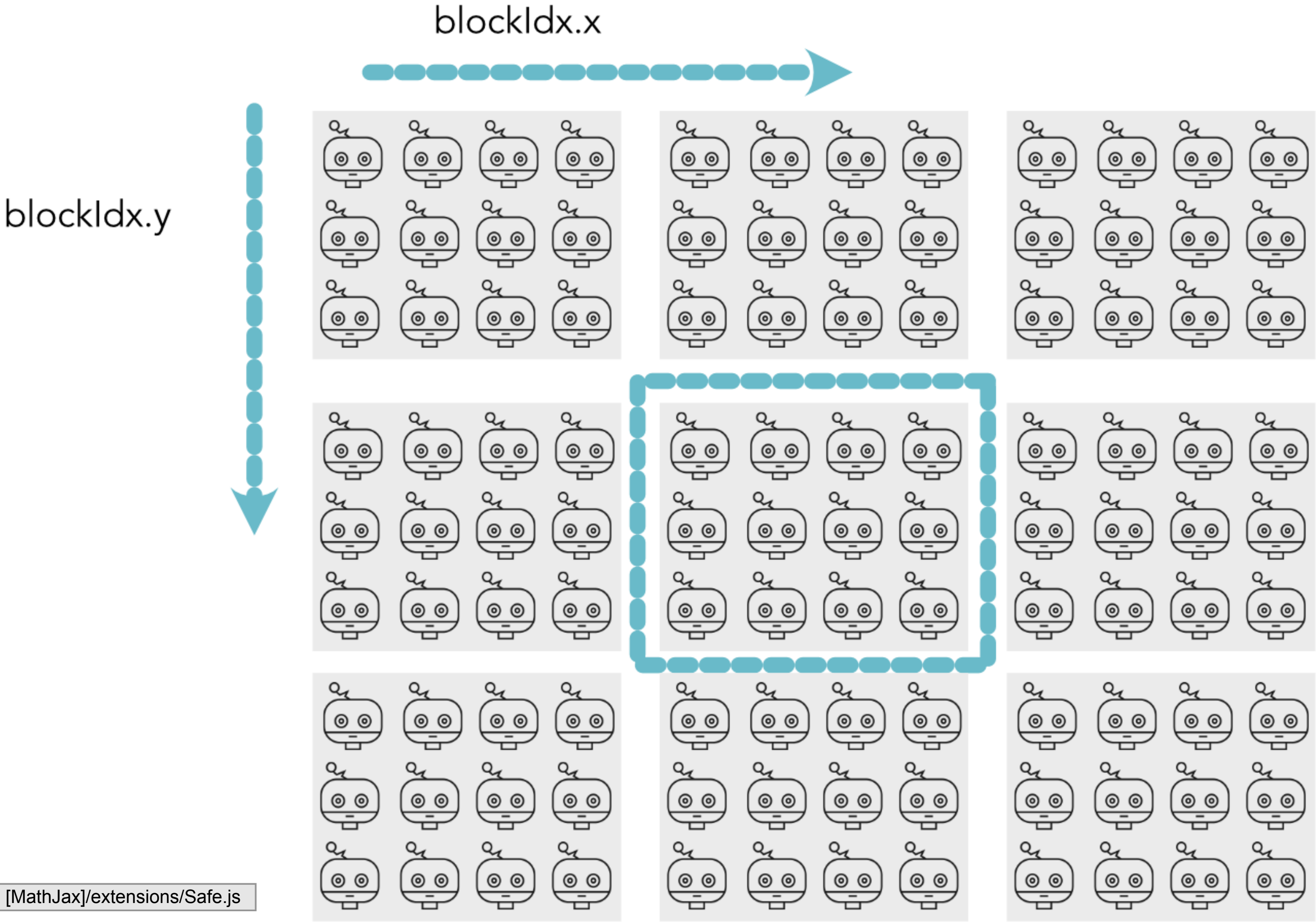
Module 3.4 - CUDA 3

thread



Loading [MathJax]/extensions/Safe.js

grid



blockIdx.x

blockIdx.y

Loading [MathJax]/extensions/Safe.js

Stack

- Threads: Run the code
- Block: Groups "close" threads
- Grid: All the thread blocks
- Total Threads: `threads_per_block` x `total_blocks`

CUDA Code

- Every thread runs simultaneously
- (Roughly) in lockstep
- How can we get anything done?

Loading [MathJax]/extensions/Safe.js

Memory

- CUDA memory hierarchy
- Local > Shared > Global
- Goal: minimize global reads and writes

Constraints

- Memory must be typed
- Memory must be *constant* size
- Memory must be relatively small

Loading [MathJax]/extensions/Safe.js

Quiz

Quiz

Loading [MathJax]/extensions/Safe.js

CUDA Algorithms

Examples from Puzzles

(Puzzles)

Example 1: Sliding Average

Compute sliding average over a list

```
sub_size = 2
a = [4, 2, 5, 6, 2, 4]
out = [3, 3.5, 5.5, 4, 3]
```

Loading [MathJax]/extensions/Safe.js

Basic CUDA

Compute CUDA

```python
def slide_cuda(out, a):
    i = numba.cuda.blockIdx.x * THREADS \
        + numba.cuda.threadIdx.x
    if i + sub_size < a.size:
        out[i] = 0
        for j in range(sub_size):
            out[i] += a[i + j]
        out[i] = out[i] / sub_size
```

Loading [MathJax]/extensions/Safe.js

Better CUDA

Two global reads per thread ::

```
def slide_cuda(out, a):
    shared = numba.cuda.shared.array(THREADS + sub_size)
    i = numba.cuda.blockIdx.x * THREADS \
        + numba.cuda.threadIdx.x
    local_idx = numba.cuda.threadIdx.x
    if i + sub_size < a.size:
        shared[local_idx] = a[i]
        if local_idx < sub_size and i + THREADS < a.size:
            shared[local_idx  + THREADS] = a[i + THREADS]
        numba.cuda.syncthreads()
        temp = 0
        for j in range(sub_size):
            temp += shared[local_idx + j]
        out[i] = temp / sub_size
```

Example 2: Reduction

Compute sum reduction over a list

```
a = [4, 2, 5, 6, 1, 2, 4, 1]
out = [26]
```

Algorithm

- Parallel Prefix Sum Computation
- Form a binary tree and sum elements

Loading [MathJax]/extensions/Safe.js

Associative Trick

Formula

$$a = 4 + 2 + 5 + 6 + 1 + 2 + 4 + 1$$

Same as

$$a = (((4 + 2) + (5 + 6)) + ((1 + 2) + (4 + 1)))$$

Loading [MathJax]/extensions/Safe.js

Associative Trick

Round 1

$$a = (((4+2)+(5+6))+((1+2)+(4+1)))$$

Round 2

$$a = ((6+11)+(3+5))$$

Round 3

$$a = (17+8)$$

Round 4

Loading [MathJax]/extensions/Safe.js

Thread Assignments

Round 1 (4 threads needed, 8 loads)

$$a = (((4+2)+(5+6))+((1+2)+(4+1)))$$

Round 2 (2 threads needed, 4 loads)

$$a = ((6+11)+(3+5))$$

Round 3 (1 thread needed, 2 loads)

$$a = (17+8)$$

Round 4

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Open Questions

- When do we read / write from global memory?
- Where do we store the intermediate terms?
- Which threads work and which do nothing?
- How does this work with tensors?

Table

```
| Thread 0 |  Thread 1 | Thread 2 | Thread 3 |
|----------|-----------|----------|----------|
| 4 + 2    |  5 + 6    | 1 + 2    |  4 + 1   |
| 6 + 11   | (zzz)     |  3 + 5   |  (zzz)   |
| 17 + 18  | (zzz)     | (zzz)    |(zzz)     |
```

Loading [MathJax]/extensions/Safe.js

Harder Questions

- What if the sequence is too short?
- What if the sequence is too long?

Loading [MathJax]/extensions/Safe.js

Too Short - Padding

- Recall that we always have a `start`, e.g. 0
- Can pad our sequence with `start`
- In practice can be done by initializing shared memory.

Too Long - Multiple Runs

- Sequence may have more elements than our block.
- Do not want to share values between of blocks.
- However, can run the code multiple times.

Loading [MathJax]/extensions/Safe.js

Example - Long Sequence

Formula

$$a = 4 + 2 + 5 + 6 + 1 + 2 + 4 + 1 + 10$$

Block size 8

$$a = (((4 + 2) + (5 + 6)) + ((1 + 2) + (4 + 1))) + 10$$

Loading [MathJax]/extensions/Safe.js

Loading [MathJax]/extensions/Safe.js

Homework Tips

- Implement simple cases first. (power of 2, no padding).
- Then try shorter sequences, and longer with tensor.
- Draw lots of diagrams, hard to debug the code directly

Loading [MathJax]/extensions/Safe.js