The CoRa Tensor Compiler: Compilation for Ragged Tensors With Minimal Padding

Pratik Fegade[1],
Tianqi Chen[1,2], Phillip B. Gibbons[1], Todd C. Mowry[1]

[1]Carnegie Mellon University [2]OctoML

Carnegie Mellon University
Computer Science Department

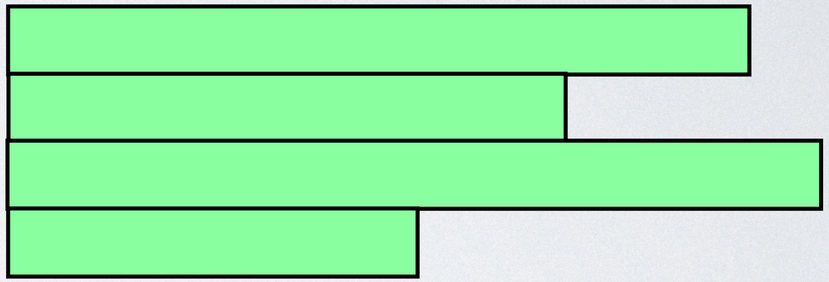
catalyst

Ragged Tensors in Deep Learning

- Natural language processing

```
input_batch = [
  [Dogs,  bark, .],
  [Maine, is,   a,     state, .],
  [The,   song, rocks, !],
  [Hello]
]
```

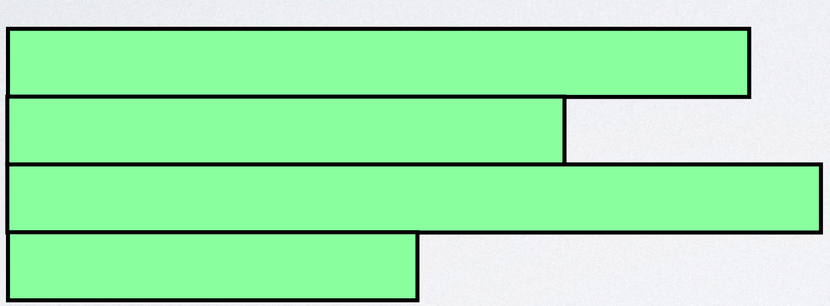
- Image processing



Ragged Tensor

Ragged Tensors

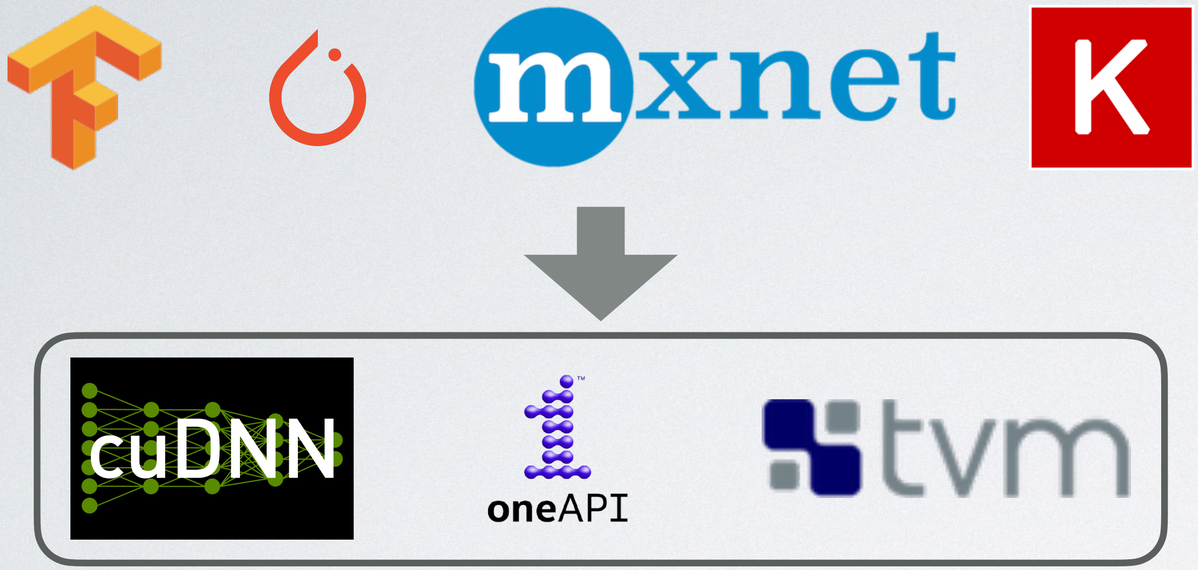
- Ragged tensor is a tensor where the slices corresponding to one or more dimensions have varying lengths

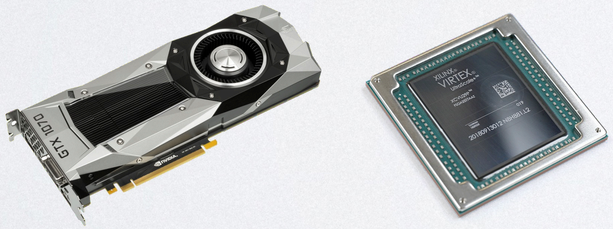


Dimensions: `[4, ?]`

Rows (slices of the inner dimension) have varying lengths

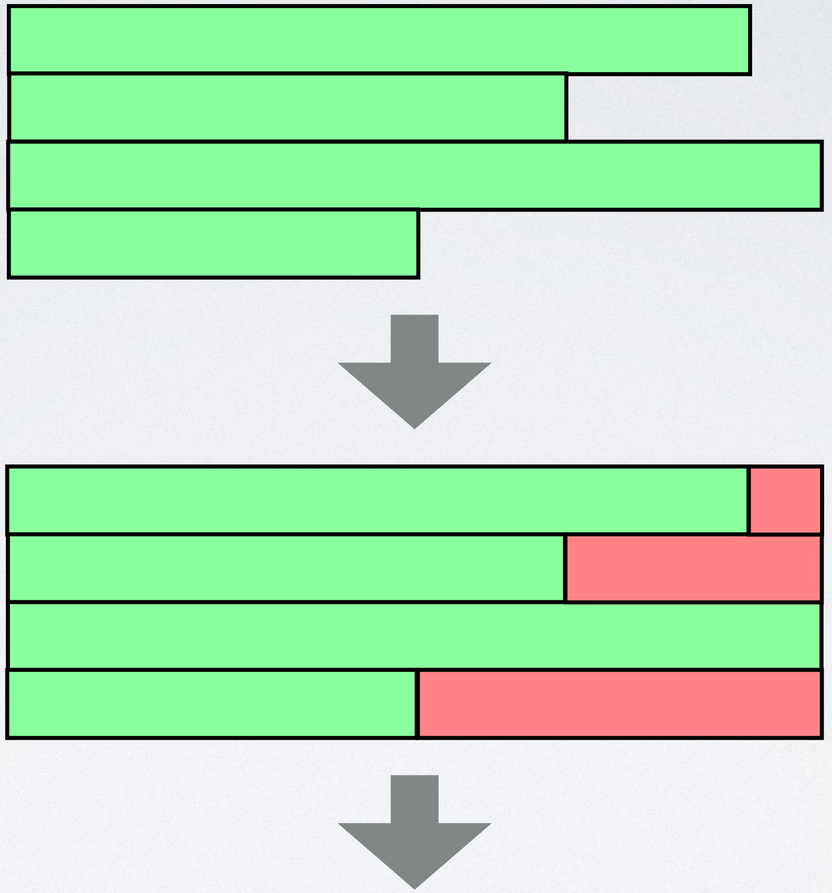
Limited Support for Ragged Tensor Operators





- Limited support for operations on ragged tensors
- Extensive support for dense tensors

And Padding Leads to Wasted Computation



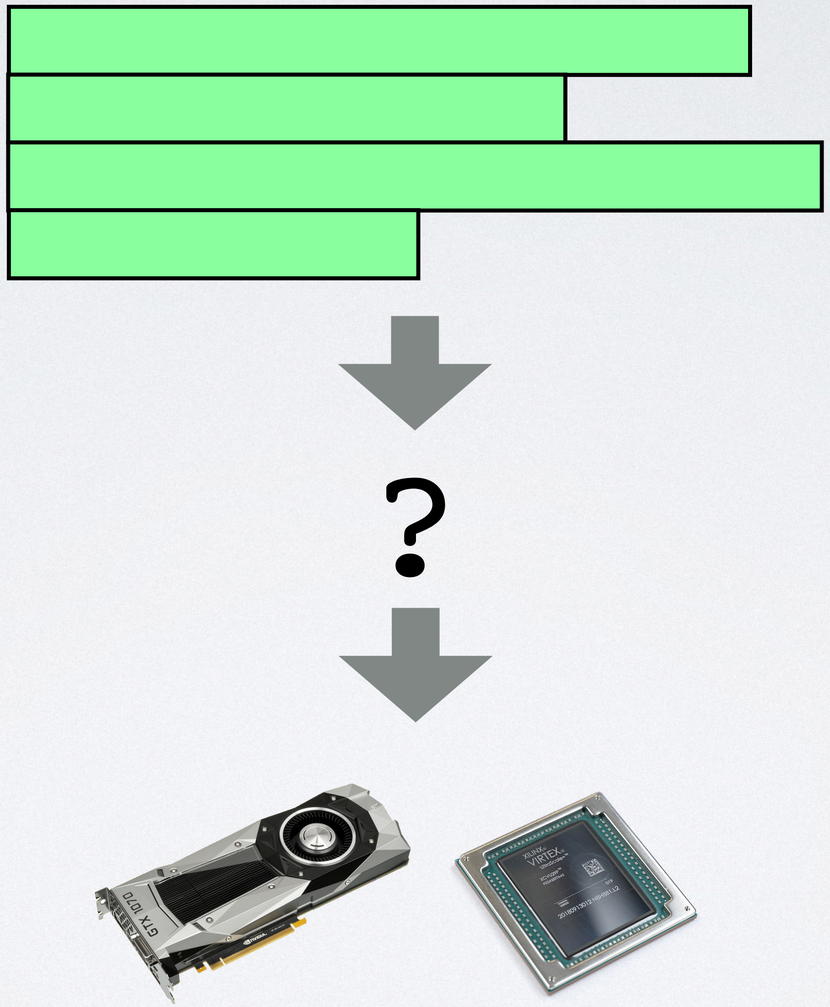
1.07 - 2.41X wasted computation!

cuDNN

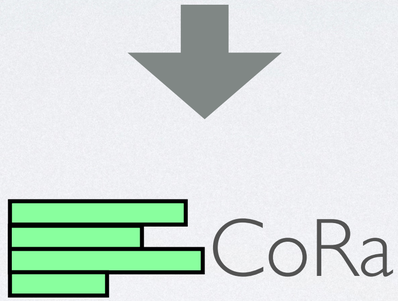
oneAPI

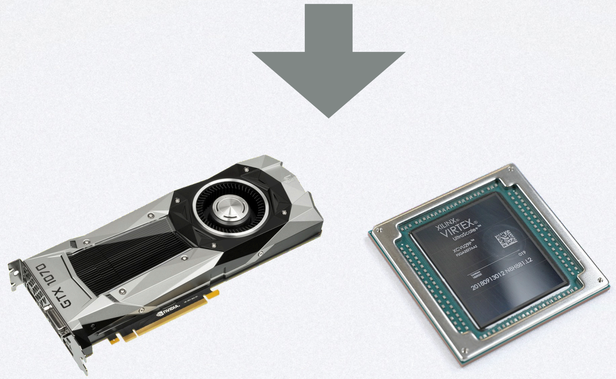
tvm

Ideal Execution: Compilation Without Padding

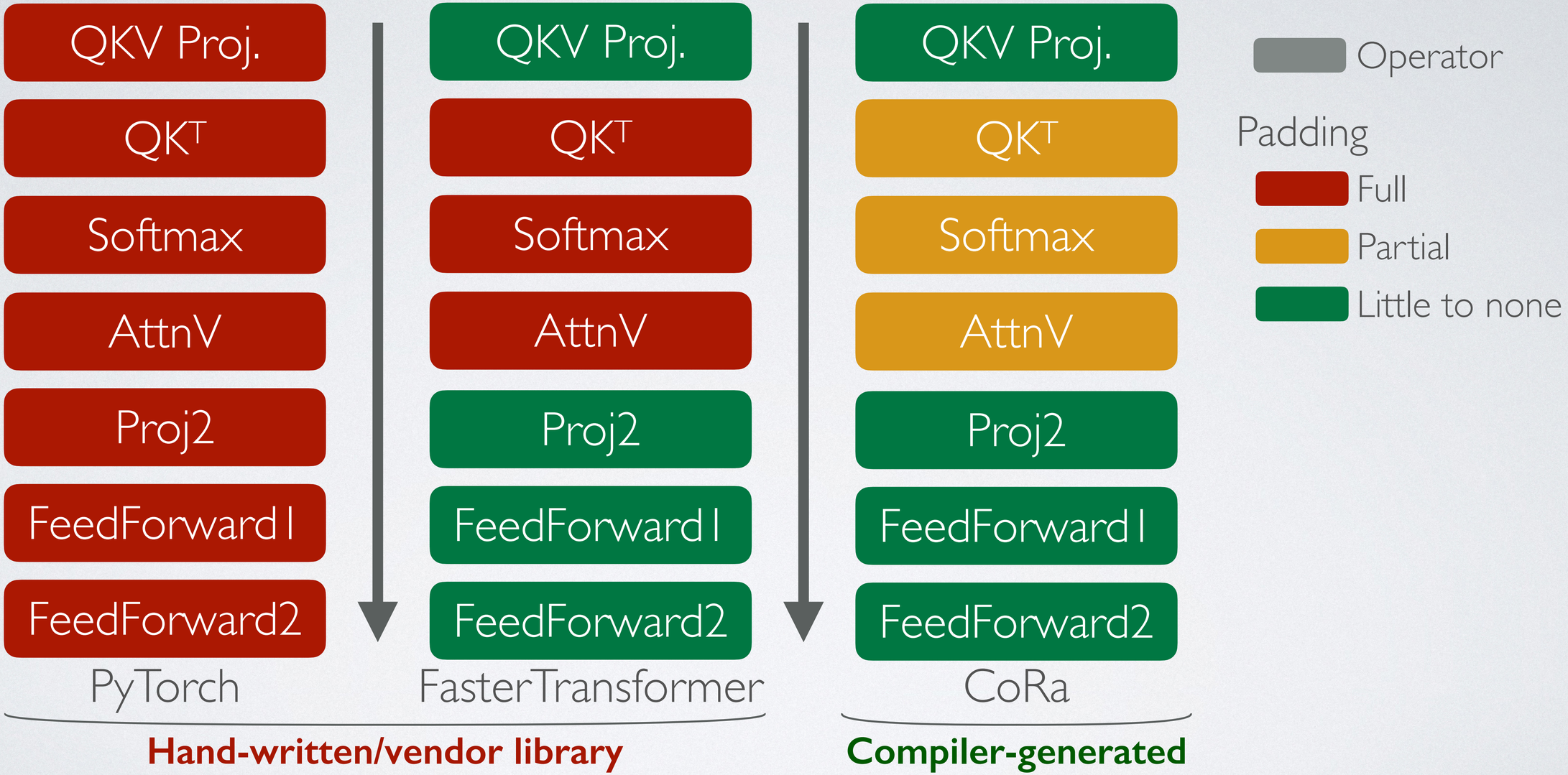


CoRa Enables Ragged Tensor Execution for Higher Frameworks





CoRa Enables Transformer Implementation Without Padding



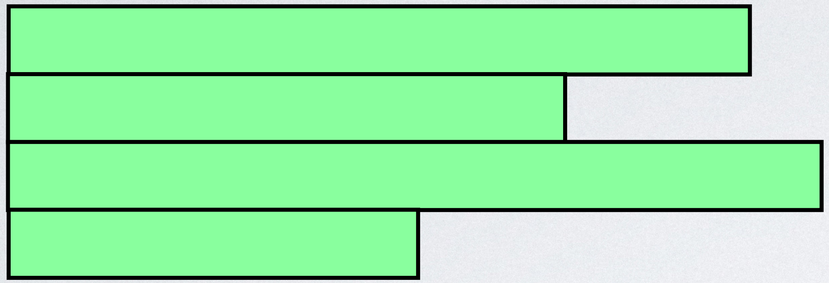
Outline

- Motivation: Inefficient Support for Ragged Tensors
- CoRa: Our Compiler Based Solution
  - Scheduling and lowering
  - API and overview
- Evaluation
- Wrapping up

Outline

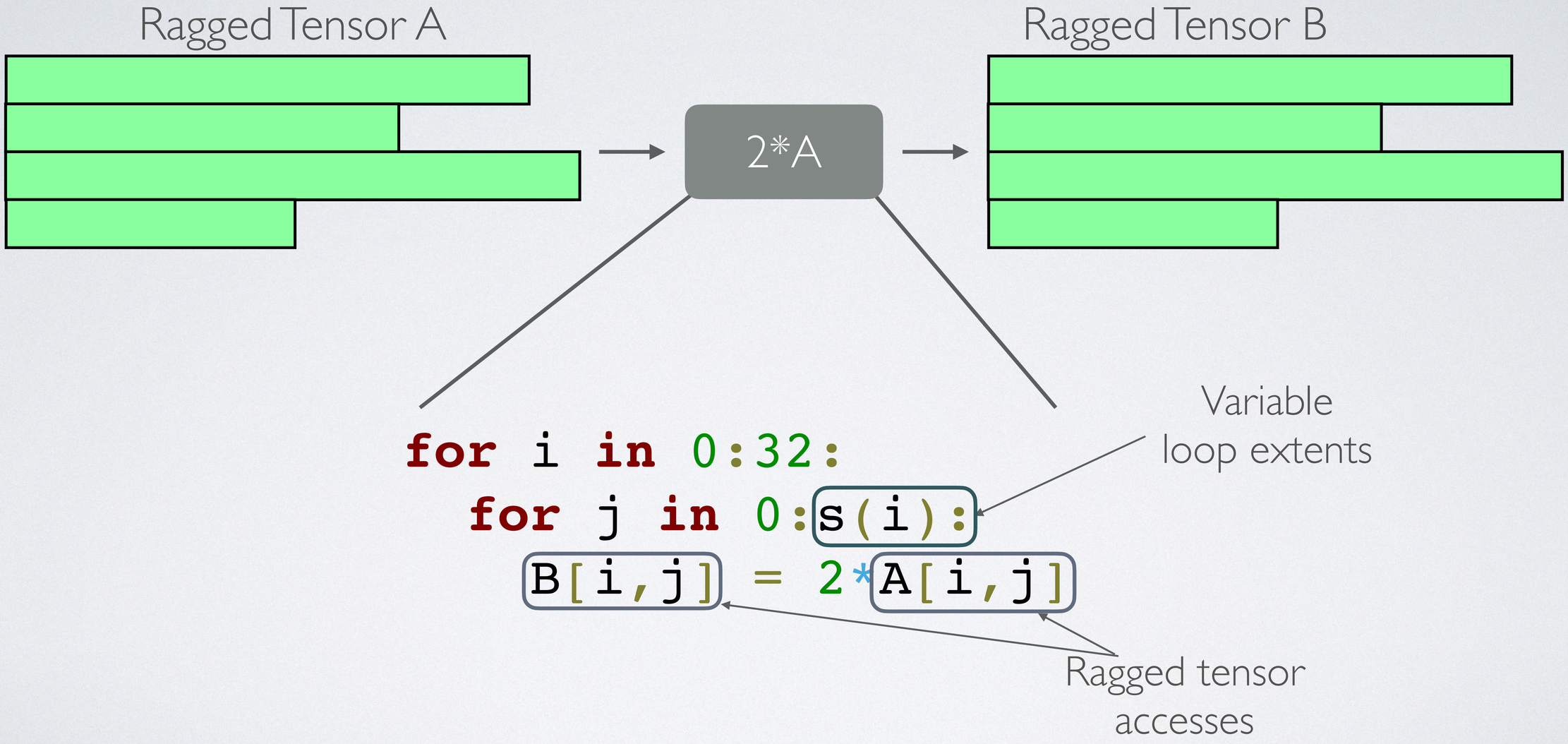
- Motivation: Inefficient Support for Ragged Tensors
- **CoRa: Our Compiler Based Solution**
  - Scheduling and lowering
  - API and overview
- Evaluation
- Wrapping up

Ragged Computations Are Similar to Dense Computations



Densely packed data with no holes, like dense tensors

Ragged Computations Are Similar to Dense Computations



Ragged Computations Are Similar to Dense Computations

- Densely packed data with no holes, like dense tensors
- Ragged computations are similar to dense tensor computations

Reuse abstractions and techniques from dense tensor compilers

```
for i in 0:32:
  for j in 0:s(i):
    B[i,j] = 2*A[i,j]
```

Variable loop extents → Generalize
- Compiler's loop representations
- Scheduling primitives and their impl.

Ragged tensor accesses → Generalize
- Tensor storage scheme
- Tensor access lowering

Ragged Computations Are Similar to Dense Computations

- Densely packed data with no holes, like dense tensors
- Ragged computations are similar to dense tensor computations

Reuse abstractions and techniques from dense tensor compilers

```
for i in 0:32:
  for j in 0:S(i):
    B[i,j] = 2*A[i,j]
```

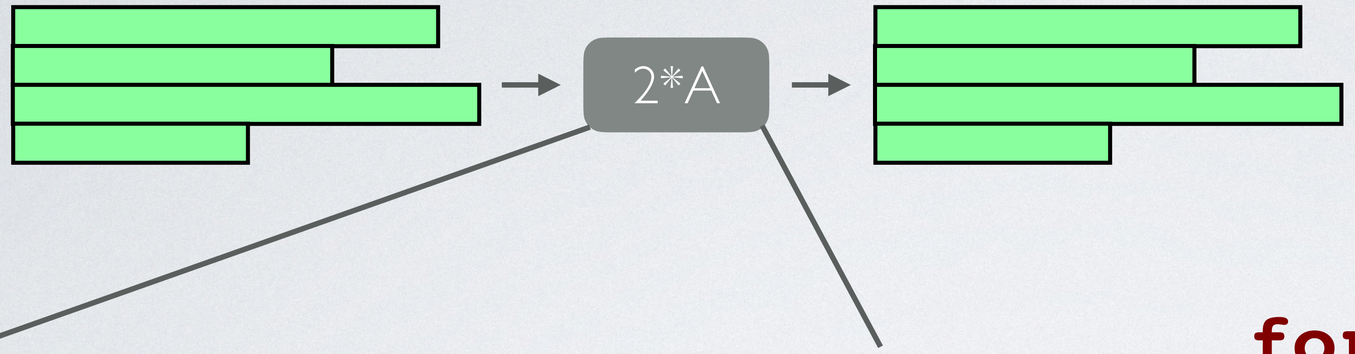
Variable loop extents →

Generalize
- Compiler's loop representations
- Scheduling primitives and their impl.

Ragged tensor accesses →

Generalize
- Tensor storage scheme
- Tensor access lowering

Loop Fusion in Ragged Operators




```
for i in 0:32:
  for j in 0:s(i):
    B[i,j] = 2*A[i,j]
```

Fuse(i,j)

$$\sum_{i=0}^{31} s(i)$$

```
for f in 0:?:
  i = ?
  j = ?
  B[i,j] = 2*A[i,j]
```

Pre-compute mappings from `f` to `i` and `j`

Once precomputed, we can schedule ragged ops like dense ops!

Ragged Computations Are Similar to Dense Computations

- Densely packed data with no holes, like dense tensors
- Ragged computations are similar to dense tensor computations

Reuse abstractions and techniques from dense tensor compilers

```
for i in 0:32:
  for j in 0:S(i):
    B[i,j] = 2*A[i,j]
```

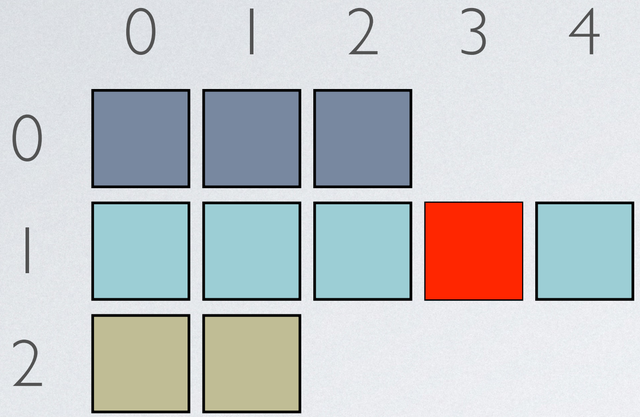
Variable loop extents →

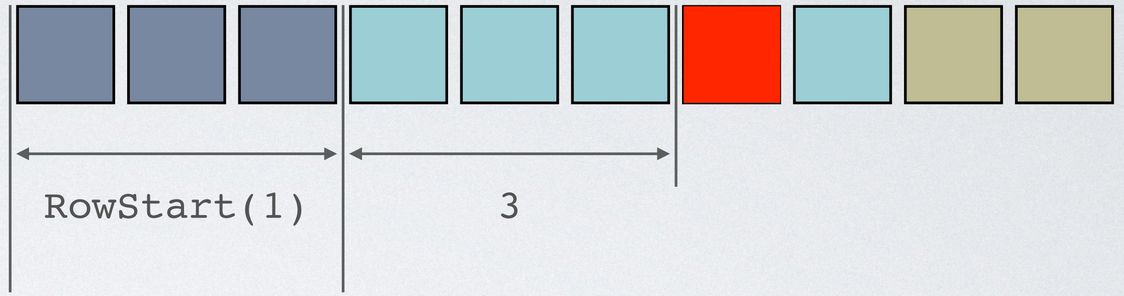
Generalize
- Compiler's loop representations
- Scheduling primitives and their impl.

Ragged tensor accesses →

Generalize
- Tensor storage scheme
- Tensor access lowering

Ragged Tensor Storage Without Padding





Offset(1, 3) = RowStart(1) + 3

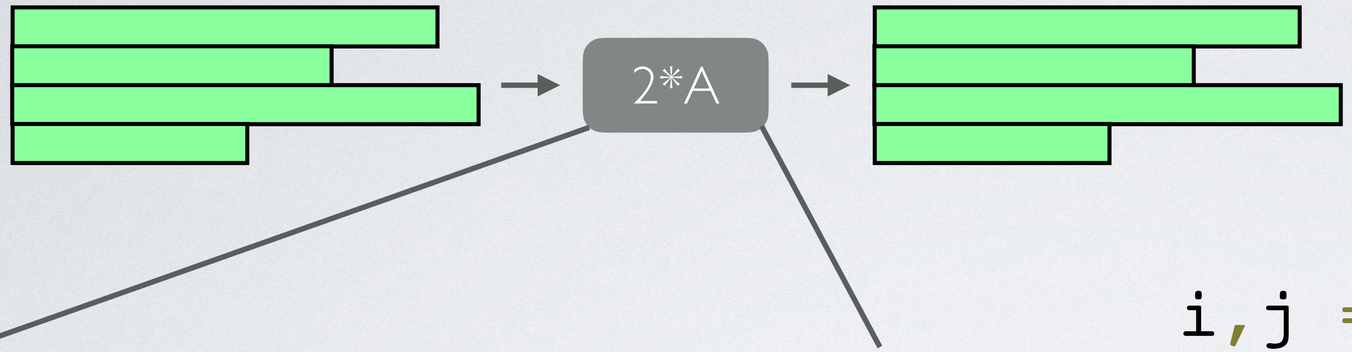
Need to precompute dimension offsets before kernel execution

Once precomputed, we have cheap random accesses, similar to dense tensors!

Outline

- Motivation: Inefficient Support for Ragged Tensors
- CoRa: Our Compiler Based Solution
  - Scheduling and lowering
  - **API and overview**
- Evaluation
- Wrapping up

CoRa's API Is Similar to That of Dense Compilers

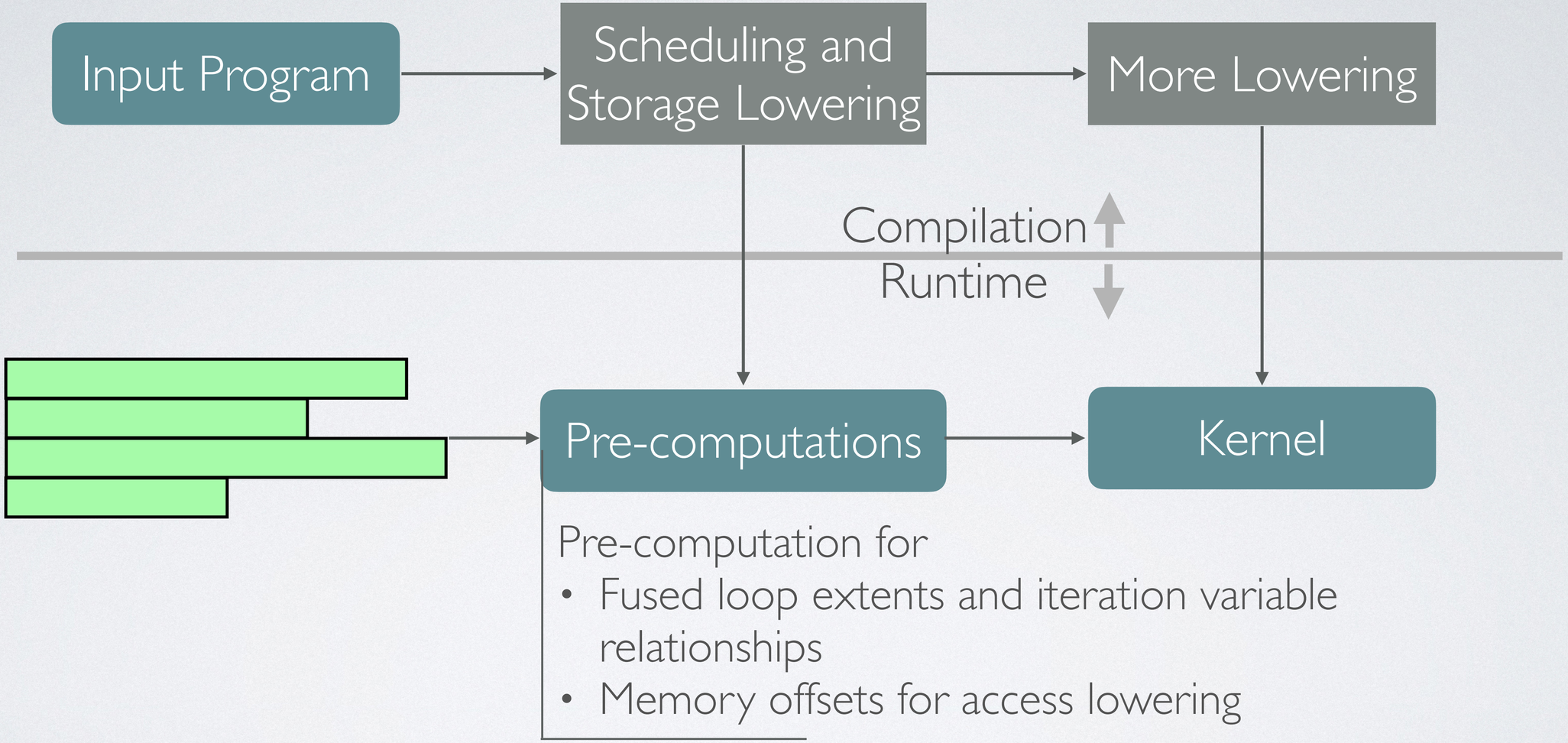



```
for i in 0:32:
  for j in 0:s(i):
    B[i,j] = 2*A[i,j]
```

```
i,j = B.axis
f = fuse(i,j)
fo, fi = split(f,64)
bind(fo, 'blockIdx.x')
bind(fi, 'threadIdx.x')
```

Other scheduling primitives for load balancing, operation splitting, tensor dimension scheduling are available

CoRa's Compilation and Runtime Pipeline

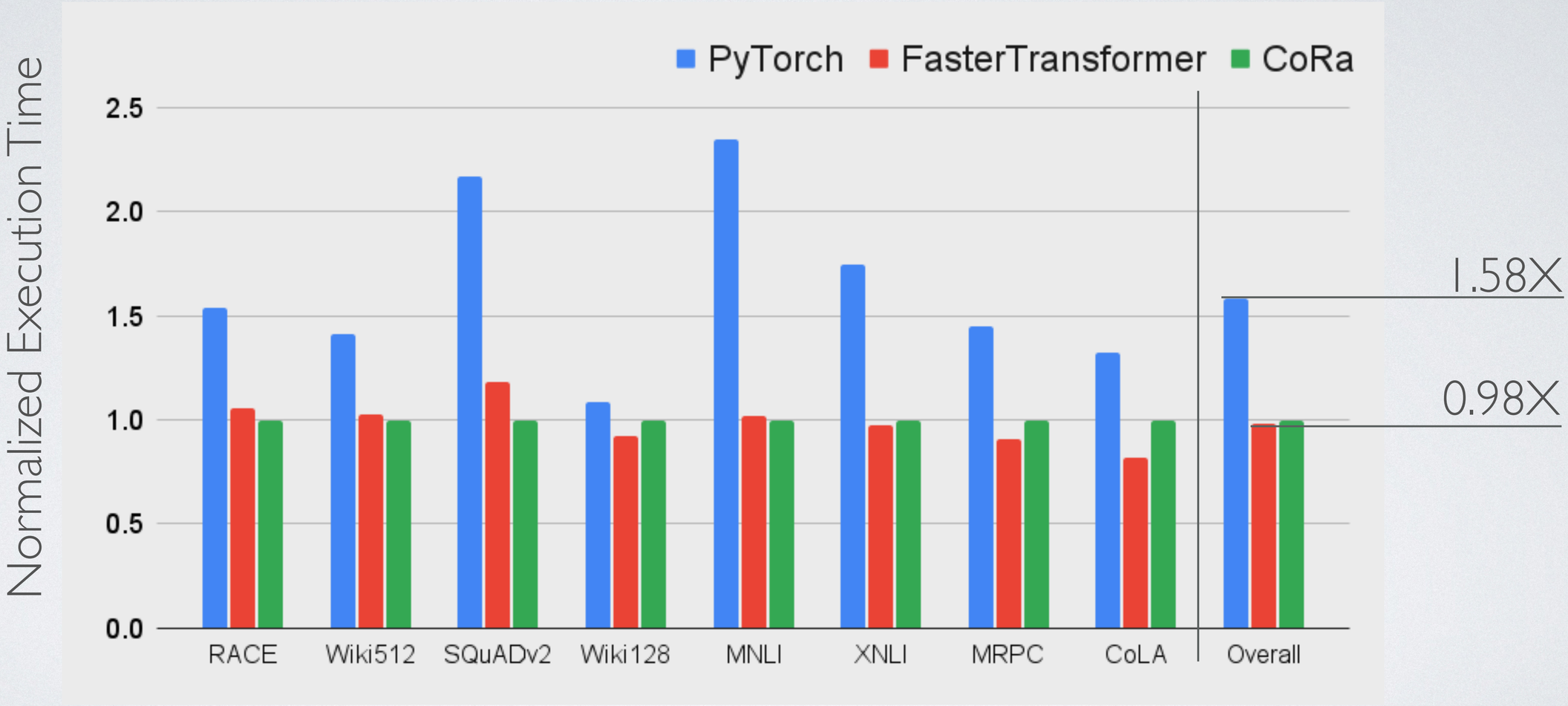


Outline

- Motivation: Inefficient Support for Ragged Tensors
- CoRa: Our Compiler Based Solution
  - Scheduling and lowering
  - API and overview
- **Evaluation**
- Wrapping up

Layer Forward Pass Latencies on Nvidia V100 GPU

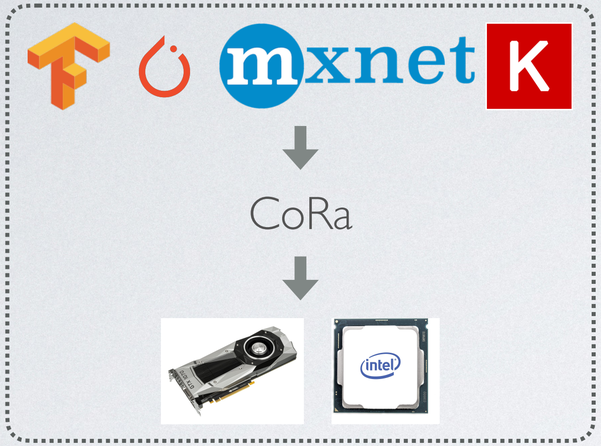
Lower is better



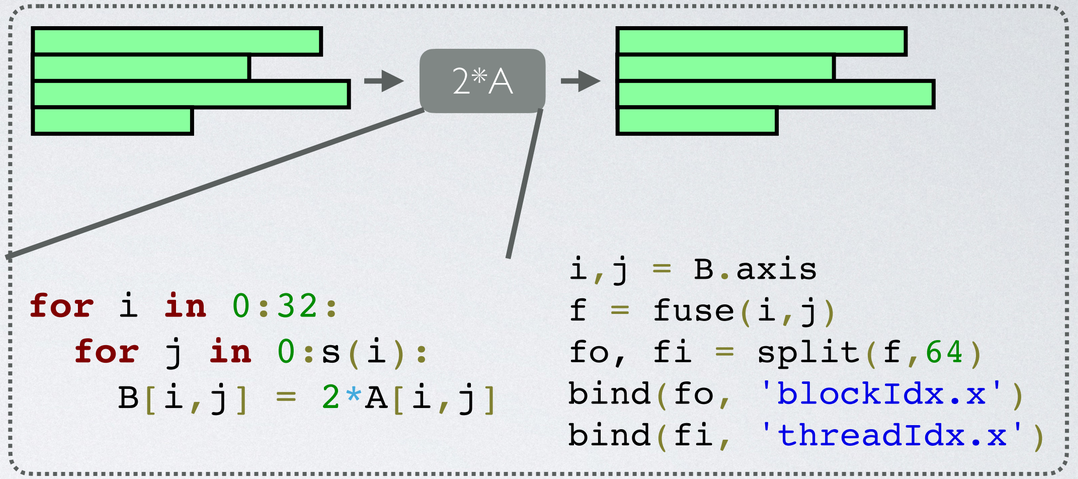
Outline

- Motivation: Inefficient Support for Ragged Tensors
- CoRa: Our Compiler Based Solution
  - Scheduling and lowering
  - API and overview
- Evaluation
- **Wrapping up**

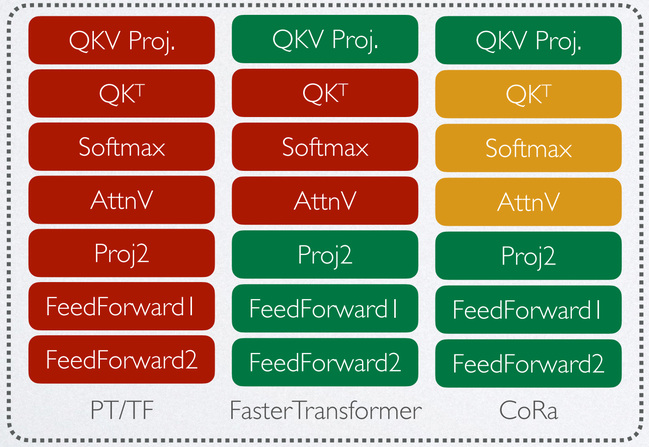
Wrapping Up CoRa



CoRa is a tensor compiler for operations on ragged tensors



CoRa provides a familiar API similar to that of dense tensor compilers



CoRa generates code as performant as hand-written code for transformer models