

Auto Clustering TensorFlow Graphs

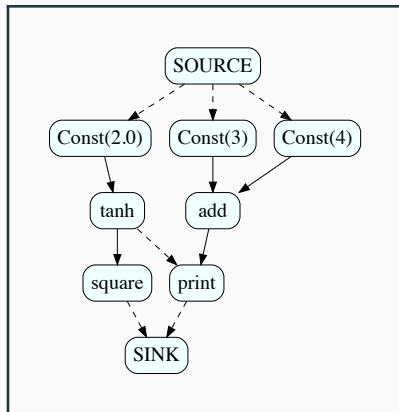
Sanjoy Das

December 17, 2018

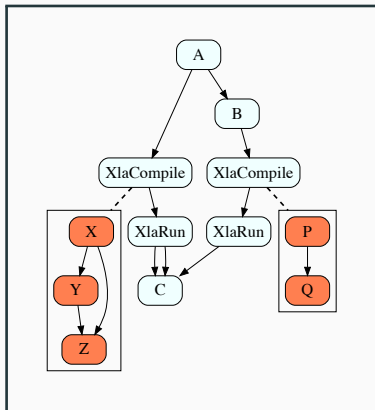
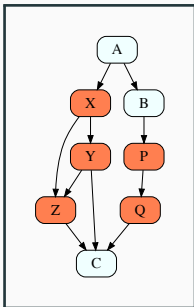
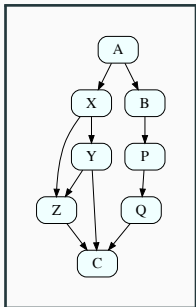
Google

Quick TensorFlow Primer

- Dataflow graph executor
- Explicit control (dashed) and data (solid) edges
- Supports an open set of operations
- Operations can have side effects
- Can represent loops and conditionals



The TensorFlow/XLA Bridge In Action



The TensorFlow/XLA Bridge

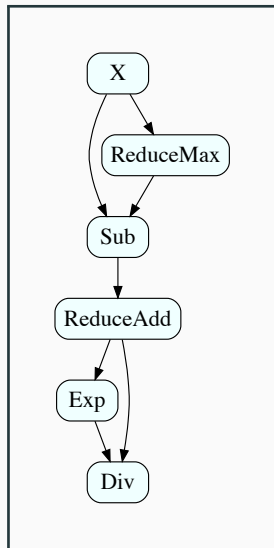
- Decides which parts should be compiled by XLA
(Clustering)
- Converts TensorFlow nodes into XLA subgraphs
(Translator)
- Compiles and executes a TensorFlow subgraph using XLA
(JIT)

The TensorFlow/XLA Bridge: Translator

- Maps one TensorFlow node into one or more XLA nodes
- Not all TensorFlow ops are supported
- Interesting area for IR design

The TensorFlow/XLA Bridge: Translator

For example $Y = \text{tf.SoftMax}(X)$ node is lowered into (roughly) the XLA graph shown on the right:



The TensorFlow/XLA Bridge: JIT

- TensorFlow invokes XLA as a “Just In Time” Compiler
- Key functionality in the `_XlaCompile` and `_XlaRun` op kernels
- Does some runtime specialization because XLA needs compile-time constant shapes
- Implements “lazy compilation”

The TensorFlow/XLA Bridge: “Auto” Clustering

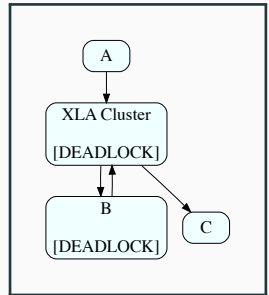
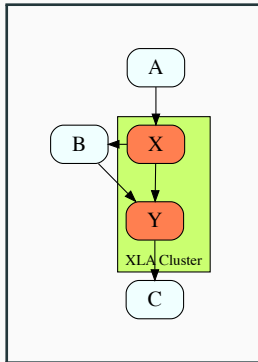
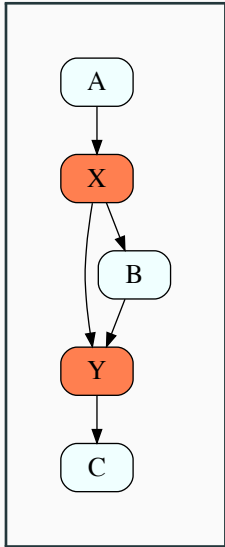
- Automatically discover clusters that should be compiled by XLA
- Should always preserve graph semantics
- Performance compared to TensorFlow should be never be worse and often be better

The TensorFlow/XLA Bridge: “Auto” Clustering

So ... what's the big deal?

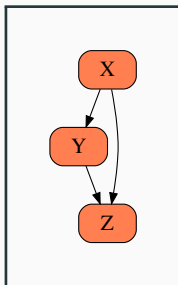
Auto clustering is surprisingly difficult!

Auto Clustering: Cycle Detection

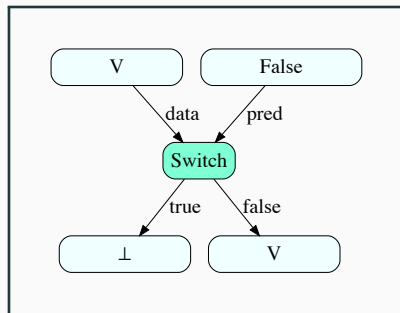
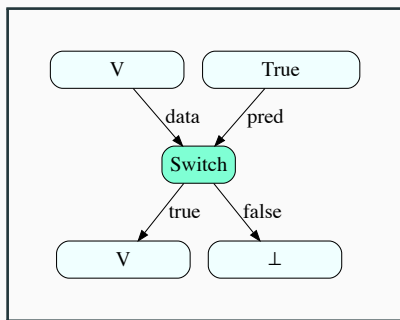


Auto Clustering: Cycle Detection

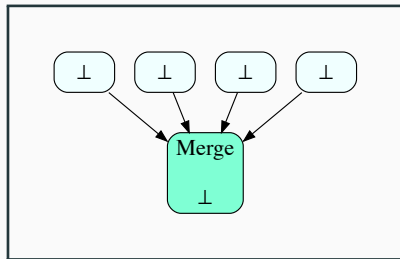
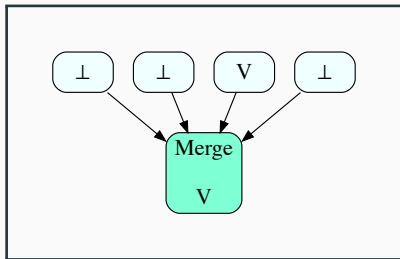
- Online cycle detection algorithm
- Run as we make decisions about which nodes to put in which cluster
- Uses a worklist because the technique is visit order dependent



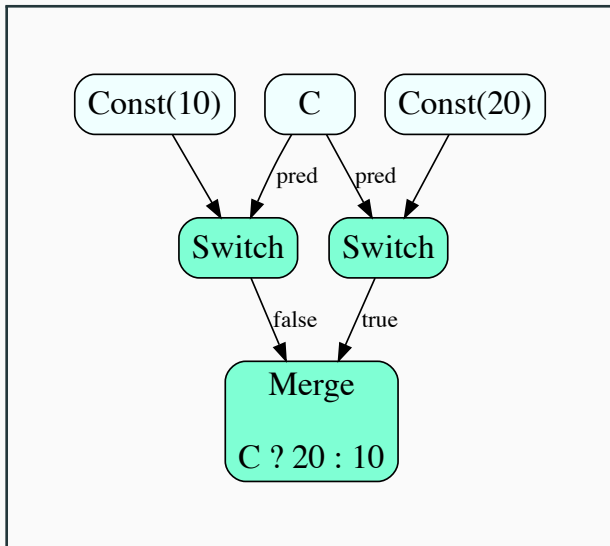
Conditionals in TensorFlow



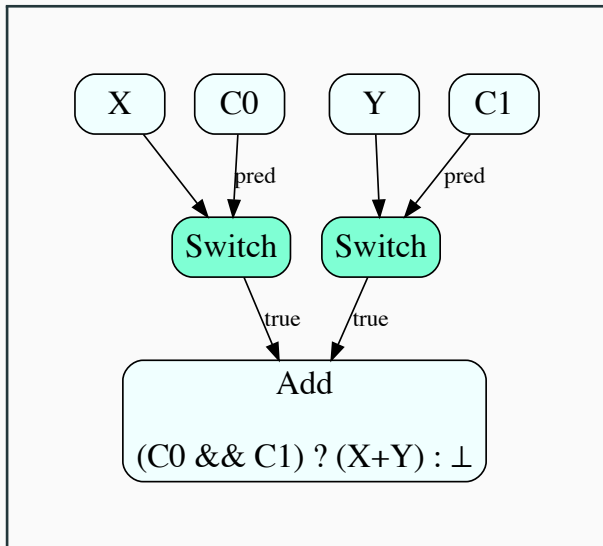
Conditionals in TensorFlow



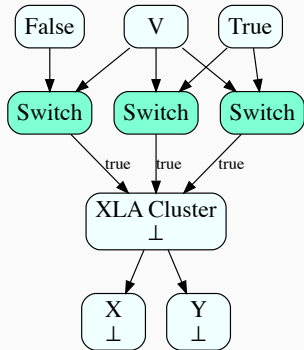
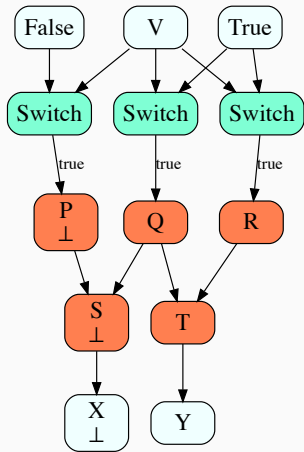
Conditionals in TensorFlow



Conditionals in TensorFlow

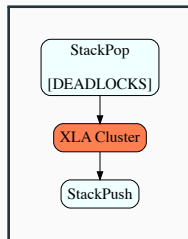
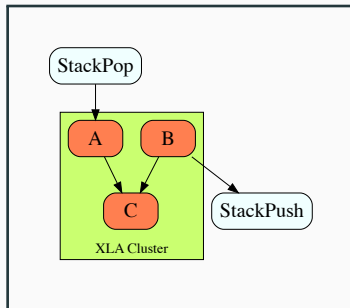
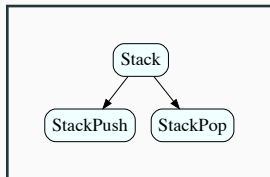


Auto Clustering & Deadness



- Map each node to a symbolic predicate that is true iff the node is executed
- All nodes in the same cluster are constrained to have the same “is live” predicate
- Conservatively correct because we check syntactic equivalence

Serializing Blocking Operations [Work In Progress]



Runtime Specialization of Shapes

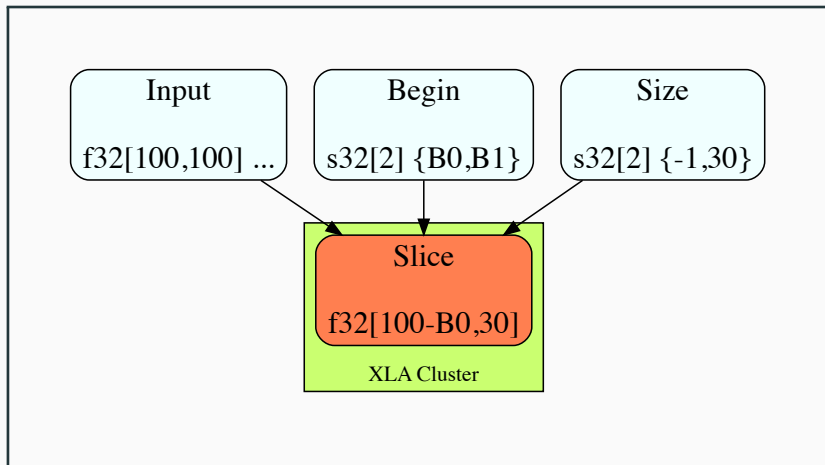
- XLA is statically shaped while TensorFlow is not
- We version (with caching) XLA executables on the shapes in the cluster

Runtime Specialization of Shapes

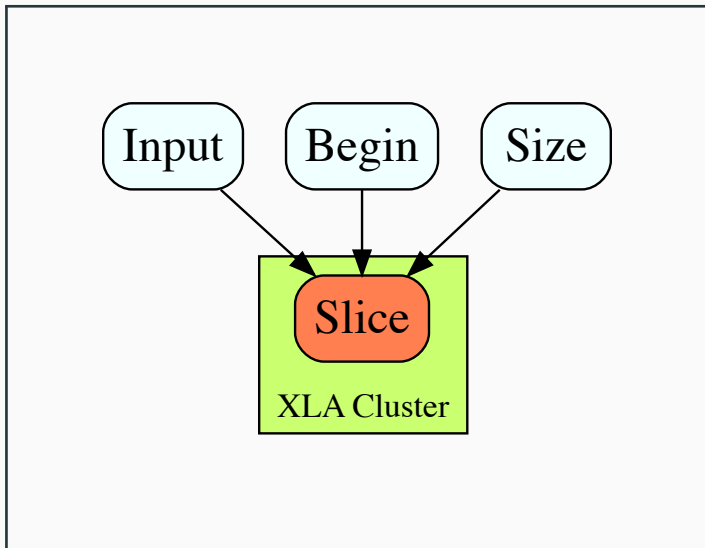
Output shape for `tf.slice(input, begin, size)`:

```
output_size[i] =  
    (size[i] == -1) ? (input.shape()[i] - begin[i]) :  
                    size[i];
```

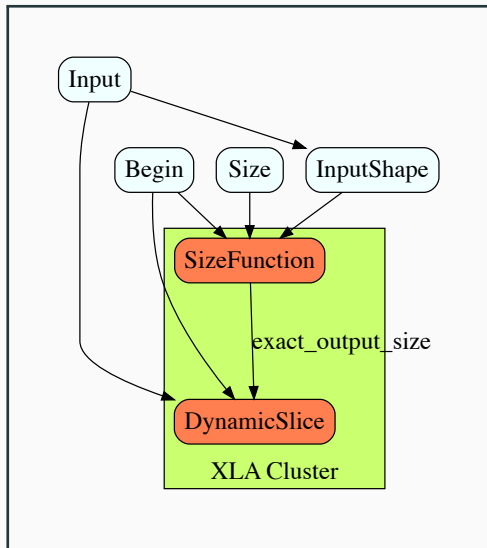
Reducing Unnecessary Recompilation



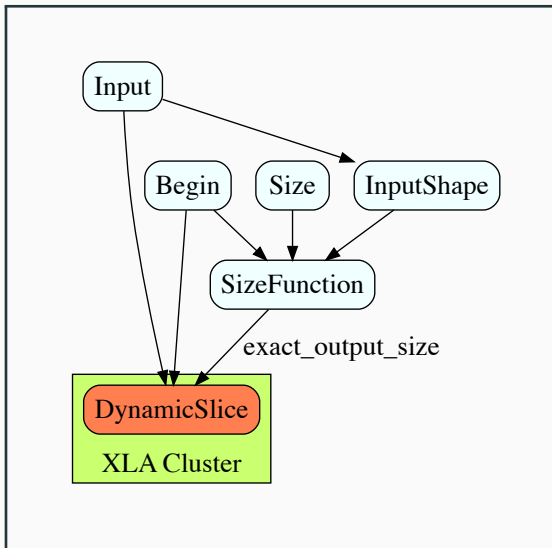
Reducing Unnecessary Recompilation



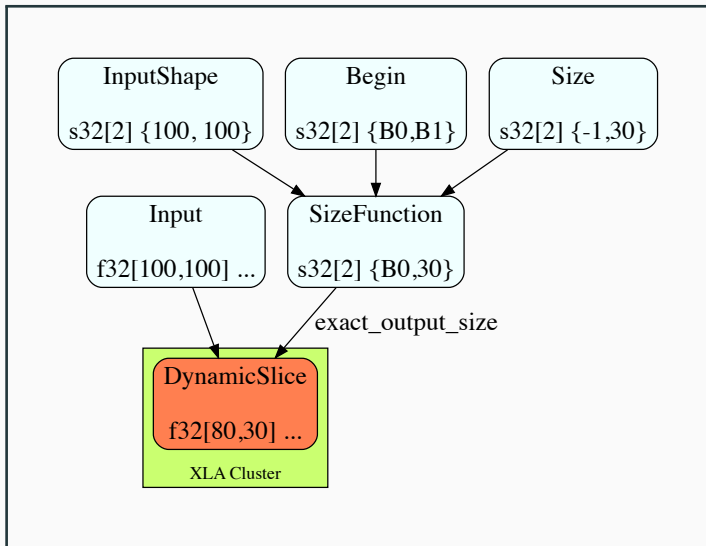
Reducing Unnecessary Recompilation



Reducing Unnecessary Recompilation

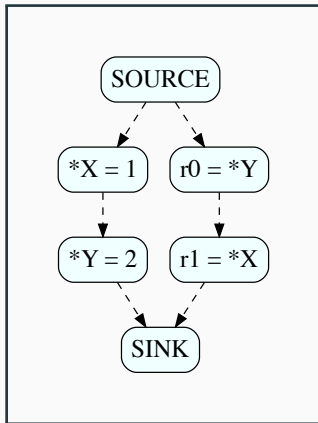


Reducing Unnecessary Recompilation



Resource Variable Operations in TensorFlow

- Resource variables are mutable “cells” that point to immutable tensors
- Semantically, reads and writes execute in a total order consistent with the partial order of the graph
- Given the graph on the right we can assert “ $r0 == 2$ implies $r1 == 1$ ”



Resource Variable Operations in XLA

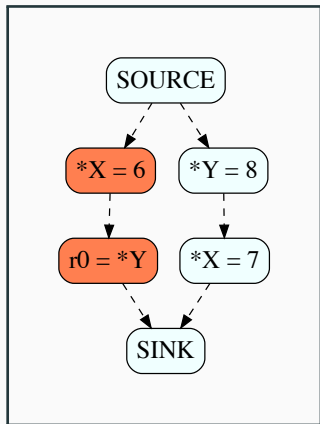
- Clustering resource variable operations can be important in some cases.
- However, XLA would prefer not to represent resource variable operations directly in its IR.
- Solution:
 - Split the computation into “pure” and “impure” (side effecting) parts
 - Have XLA handle the “pure” bits, and the TF/XLA bridge handle the “impure” bits

Resource Variable Operations in XLA

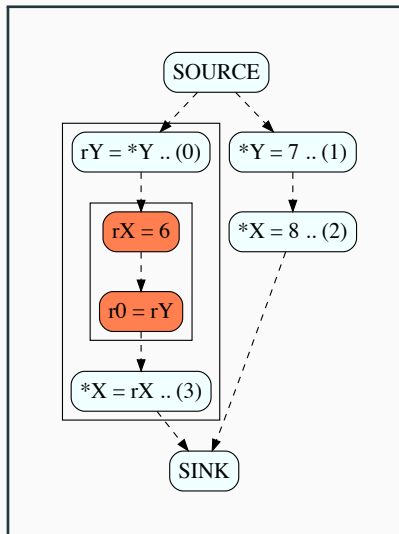
```
r0 = Read(X)
r1 = Read(Y)
Write(42, Z)
r2 = Read(Z)
r3 = r0 + r1 + r2
Write(Z, r3)
```

1. // The TF/XLA Bridge
rX0 = Read(X); rY0 = Read(Y)
2. // The XLA Computation
r0 = rX0 // Read(X)
r1 = rY0 // Read(Y)
rZ0 = 42 // Write(42, Z)
r2 = rZ0 // Read(Z)
r3 = r0 + r1 + r2
rZ1 = r3 // Write(Z, r3)
3. // The TF/XLA Bridge
Write(Z, rZ1)

Resource Variable Operations in XLA



Assert “`*X == 6` implies `r0 == 8`”



Resource Variable Operations in XLA

- Solution: Static Analysis!
- Analyze the TensorFlow graph to figure out which pairs of resource operations cannot be put into the same cluster
- Make auto-clustering respect these constraints

Auto Clustering: Current Status

- We've made significant progress towards auto-clustering for XLA GPU, but we're not production ready yet
- We'd love for you to try it out!
 - Change the `TF_XLA_FLAGS` environment variable to include `--tf_xla_auto_jit=2` to enable auto clustering for all graphs
 - You may have to change your model to use resource variables for best results
- There are no immediate plans for auto-clustering for XLA CPU

Thank you!
Questions?