Auto Clustering TensorFlow Graphs

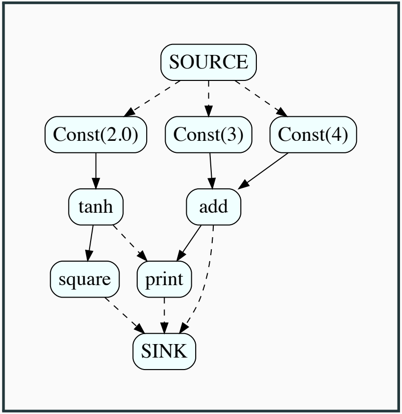
Sanjoy Das

December 17, 2018

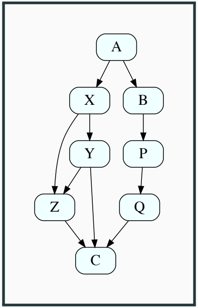
Google

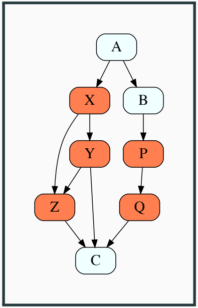
Quick TensorFlow Primer

- Dataflow graph executor
- Concurrent by "default"
- Supports an open set of operations
- Operations can have side effects
- Can represent loops and conditionals

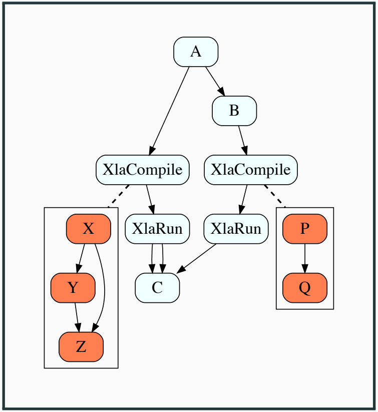


The TensorFlow/XLA Bridge In Action





Mark For Compilation



Encapsulate Subgraphs & Build XLA Ops

The TensorFlow/XLA Bridge

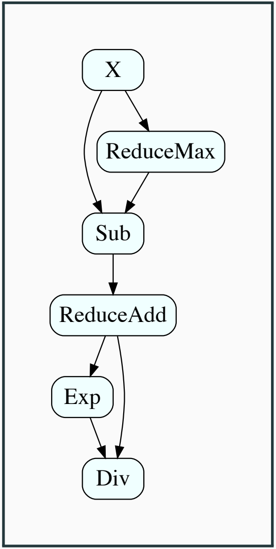
- Decides which parts should be compiled by XLA (Clustering)
- Converts TensorFlow nodes into XLA subgraphs (Translator)
- Compiles and executes a TensorFlow subgraph using XLA (JIT)

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 = 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".

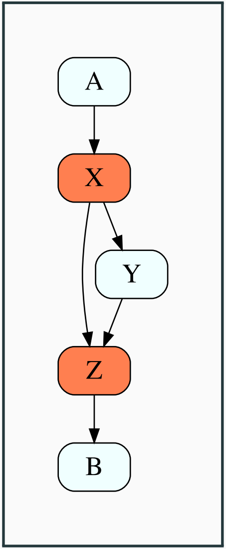
The TensorFlow/XLA Bridge: Auto Clustering

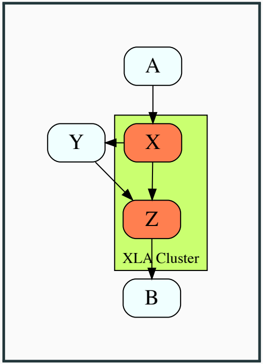
- Automatically discover clusters that should be compiled by XLA
- Should always preserve graph semantics
- Performance compared to TensorFlow should be never be worse and often be better

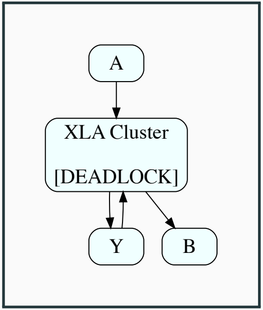
The TensorFlow/XLA Bridge: Auto Clustering

Auto-clustering is surprisingly difficult!

Auto Clustering: Cycle Detection



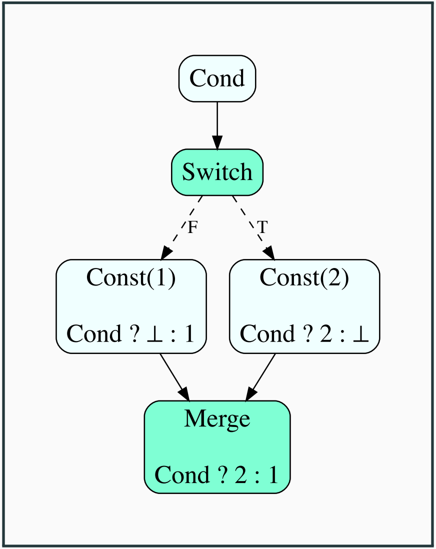




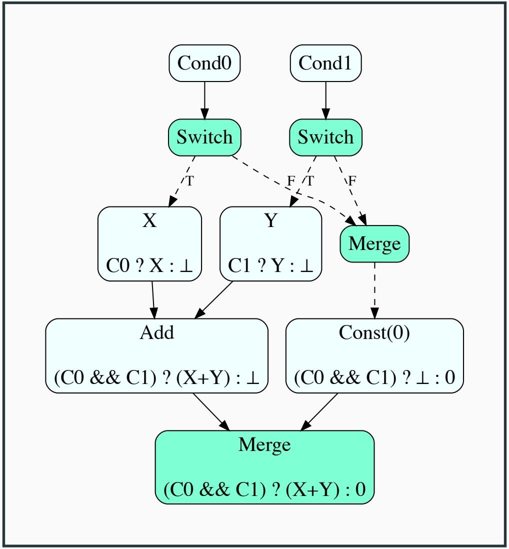
Auto Clustering: Cycle Detection

- 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

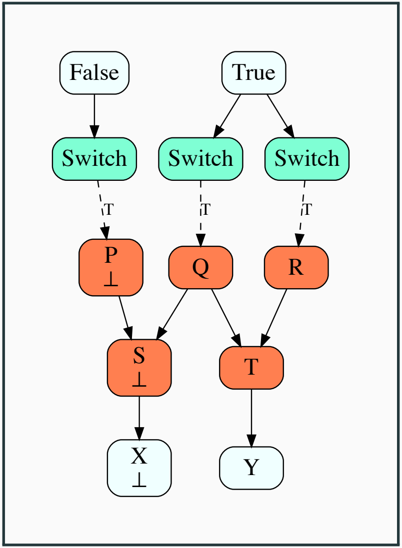
Conditionals in TensorFlow

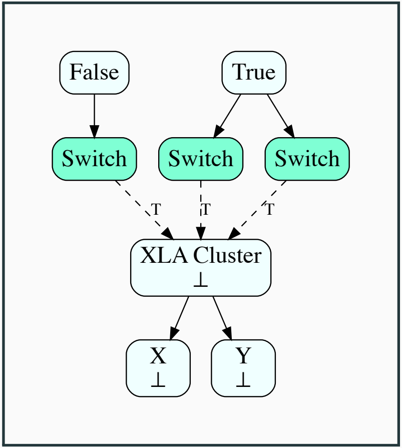


Conditionals in TensorFlow



Auto Clustering & Deadness

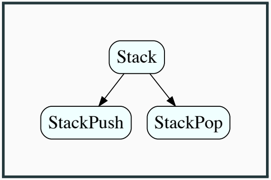


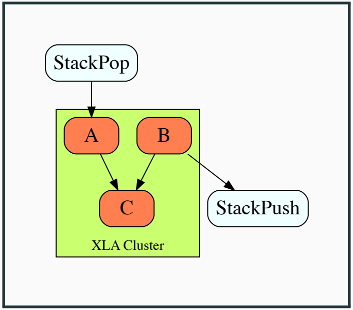


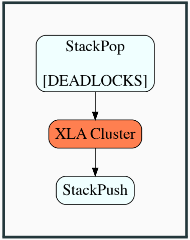
Auto Clustering & Deadness

- Map each node to a symbolic predicate that is true iff the node is execute
- 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]

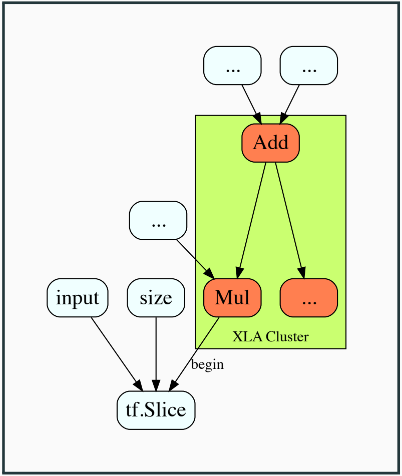




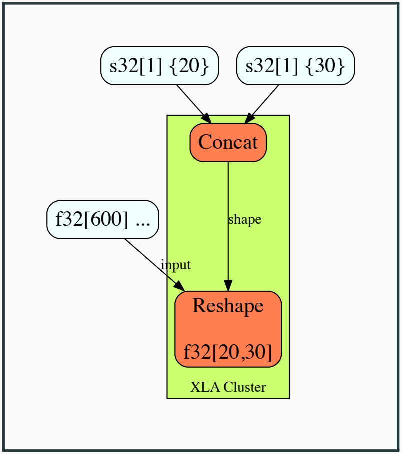


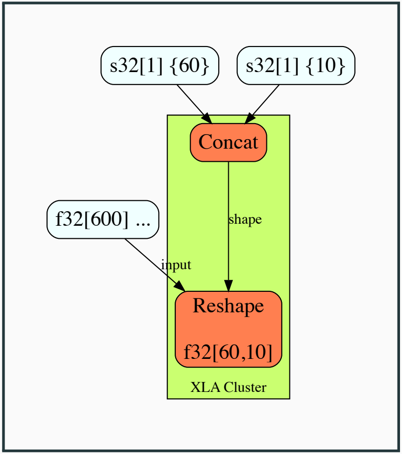
Auto Clustering: Device To Host Copies

- XLA only produces device memory buffers
- May introduce bottlenecks by not letting the CPU run ahead of the GPU
- We "decluster" nodes to avoid this problem



Runtime Specialization of Shapes



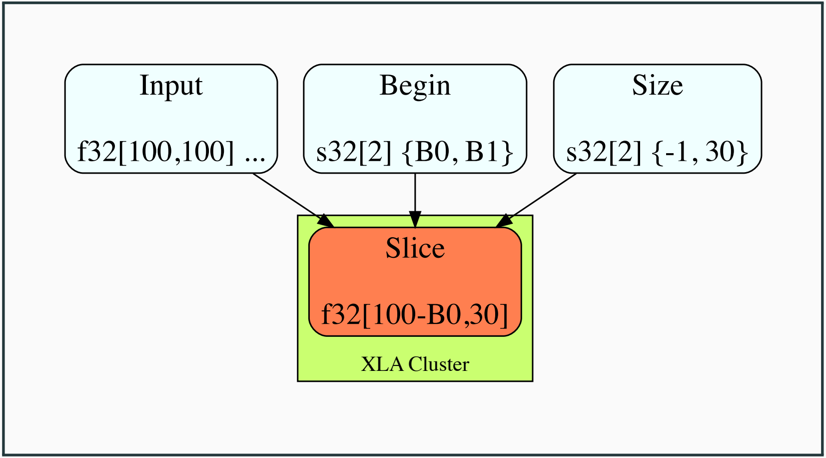


Runtime Specialization of Shapes

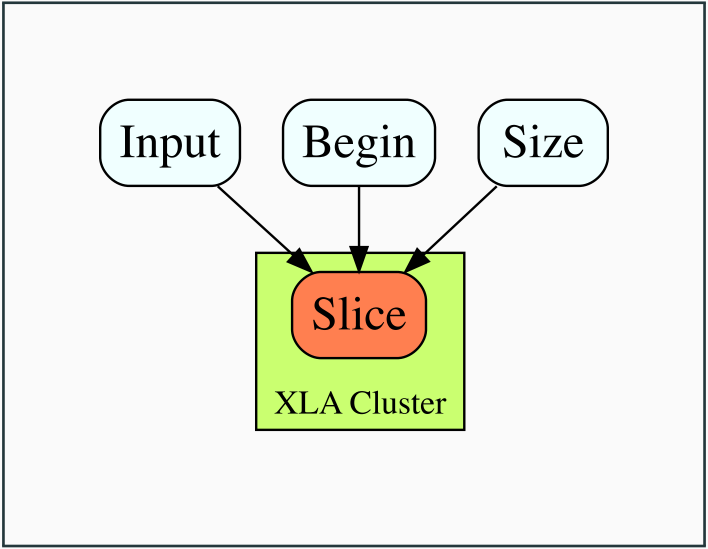
Output shape for `tf.slice(input, begin, size)`:

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

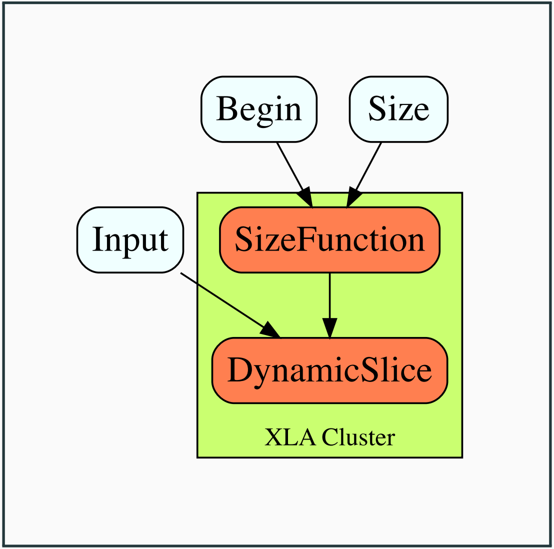
Reducing Unnecessary Recompilation



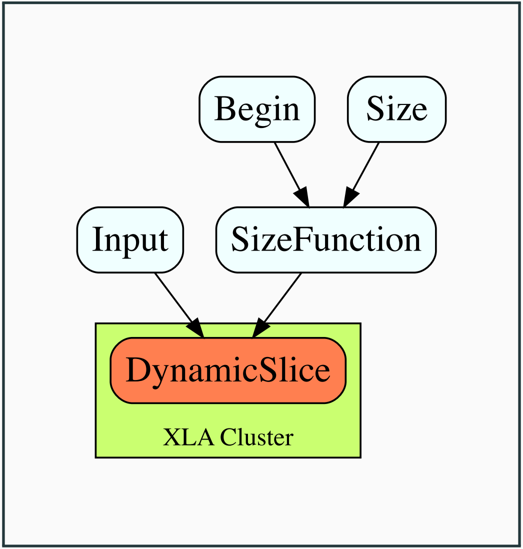
Reducing Unnecessary Recompilation



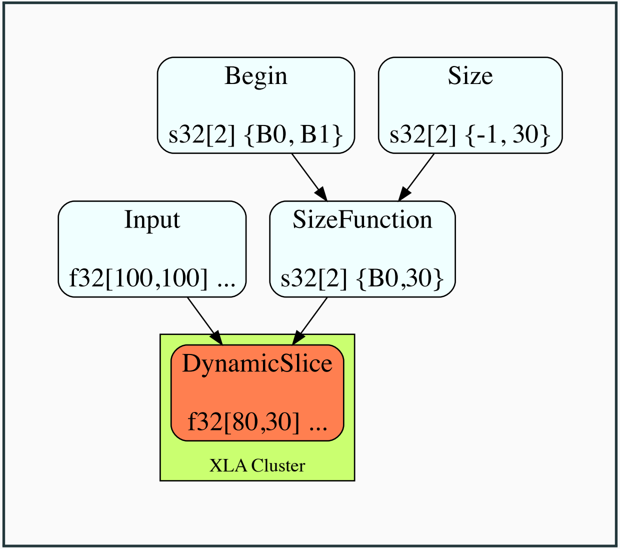
Reducing Unnecessary Recompilation



Reducing Unnecessary Recompilation

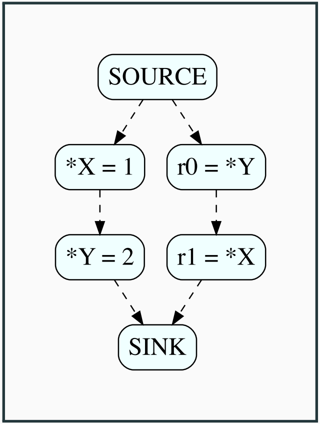


Reducing Unnecessary Recompilation



Resource Variable Operations in TensorFlow

- Resource variables are mutable "cells" that point to immutable tensors
- Resource variables reads and writes are atomic
- 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

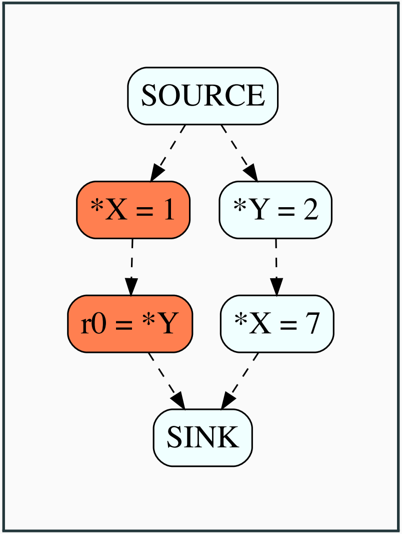
- Clustering resource variable operations can be important in some cases.
- However, XLA would prefer not to represent resource variable operations directly in its IR.
- Solution:
  - Split the computation into “pure” and “impure” (side effecting) parts
  - Have XLA handle the “pure” bits, and the TF/XLA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

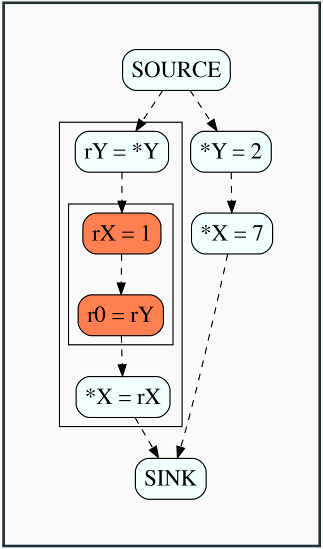
Resource Variable Operations in XLA

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

1. ```
   // The TF/XLA Bridge
   rX = Read(X); rY = Read(Y)
   ```
2. ```
   // The XLA Computation
   r0 = rX  // Read(X)
   r1 = rY  // Read(Y)
   rZ = 42  // Write(42, Z)
   r2 = rZ  // Read(Z)
   r3 = r0 + r1 + r2
   rZ = r3  // Write(Z, r3)
   ```
3. ```
   // The TF/XLA Bridge
   Write(Z, rZ)
   ```
   ```

Resource Variable Operations in XLA





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 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