tensorflower-gardener Remove PersistentTensor from assign_op.h ... ✓ History

5 contributors     

145 lines (124 sloc) | 5.52 KB ...

```
1  /* Copyright 2015 The TensorFlow Authors. All Rights Reserved.
2
3  Licensed under the Apache License, Version 2.0 (the "License");
4  you may not use this file except in compliance with the License.
5  You may obtain a copy of the License at
6
7      http://www.apache.org/licenses/LICENSE-2.0
8
9  Unless required by applicable law or agreed to in writing, software
10 distributed under the License is distributed on an "AS IS" BASIS,
11 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
12 See the License for the specific language governing permissions and
13 limitations under the License.
14 ==============================================================================*/
15
16 #ifndef TENSORFLOW_CORE_KERNELS_ASSIGN_OP_H_
17 #define TENSORFLOW_CORE_KERNELS_ASSIGN_OP_H_
18
19 #define EIGEN_USE_THREADS
20
21 #include "third_party/eigen3/unsupported/Eigen/CXX11/Tensor"
22 #include "tensorflow/core/framework/op_kernel.h"
23 #include "tensorflow/core/framework/tensor_types.h"
24
25 namespace tensorflow {
26
27 // TODO(jeff): Get rid of use_exclusive_lock_ option
28
29 // Computes *input[0] = input[1]
```

```
class AssignOp : public OpKernel {
 public:
  explicit AssignOp(OpKernelConstruction* context) : OpKernel(context) {
    OP_REQUIRES_OK(context,
                   context->GetAttr("use_locking", &use_exclusive_lock_));
    OP_REQUIRES_OK(context,
                   context->GetAttr("validate_shape", &validate_shape_));
    OP_REQUIRES(context, IsRefType(context->input_type(0)),
                errors::InvalidArgument("lhs input needs to be a ref type"));
    if (!context
             ->GetAttr("_grappler_relax_allocator_constraints",
                       &relax_constraints_)
             .ok()) {
      relax_constraints_ = false;
    }
  }

  void Compute(OpKernelContext* context) override {
    const Tensor& rhs = context->input(1);

    // We always return the input ref.
    context->forward_ref_input_to_ref_output(0, 0);

    // We can't always know how this value will be used downstream, so make
    // conservative assumptions in specifying constraints on the memory
    // allocation attributes, unless the Grappler graph analysis determined that
    // it was safe not to.
    AllocatorAttributes attr;
    if (!relax_constraints_) {
      attr.set_gpu_compatible(true);
      attr.set_nic_compatible(true);
    }

    {
      mutex_lock l(*context->input_ref_mutex(0));
      const Tensor& old_lhs = context->mutable_input(0, /* lock_held */ true);
      const bool same_shape = old_lhs.shape().IsSameSize(rhs.shape());
      if (validate_shape_) {
        OP_REQUIRES(context, same_shape,
                    errors::InvalidArgument(
                        "Assign requires shapes of both tensors to match. "
                        "lhs shape= ",
                        old_lhs.shape().DebugString(),
                        " rhs shape= ", rhs.shape().DebugString()));
      }

      // In the code below we try to minimize the amount of memory allocation
      // and copying by trying the following two shortcuts:
      // 1. If the lhs is initialized and has the same number of elements as
```

```
79        //    the rhs we can avoid a memory allocation.
80        // 2. If we can reuse the rhs buffer we avoid both a memory allocation
81        //    and copying.
82
83        // 1. Try to copy into an existing buffer.
84        if (old_lhs.IsInitialized() &&
85            old_lhs.shape().num_elements() == rhs.shape().num_elements()) {
86          // The existing lhs tensor has already been initialized and the right
87          // hand side can fit in the underlying buffer.
88          Tensor reshaped_old_lhs;
89          if (same_shape) {
90            reshaped_old_lhs = old_lhs;
91          } else {
92            CHECK(reshaped_old_lhs.CopyFrom(old_lhs, rhs.shape()));
93            context->replace_ref_input(0, reshaped_old_lhs,
94                                       /* lock_held */ true);
95          }
96          if (use_exclusive_lock_) {
97            Copy(context, &reshaped_old_lhs, rhs);
98            return;
99          }
100       } else {
101         // 2. Try to reuse the rhs.
102         std::unique_ptr<Tensor> input_alias = context->forward_input(
103             1, OpKernelContext::Params::kNoReservation /*output_index*/,
104             rhs.dtype(), rhs.shape(), DEVICE_MEMORY, attr);
105         if (input_alias != nullptr) {
106           // Update the ref to point to the new buffer.
107           context->replace_ref_input(0, *input_alias, /* lock_held */ true);
108           return;
109         }
110
111         // Otherwise, create a new tensor whose shape matches the
112         // right hand side, hand off to lhs and copy the rhs into it.
113         Tensor copy_tensor;
114         OP_REQUIRES_OK(context,
115                        context->allocate_temp(old_lhs.dtype(), rhs.shape(),
116                                               &copy_tensor, attr));
117         // We track memory of variables in variable ops instead of in this
118         // assign op.
119         context->clear_recorded_memory();
120         context->replace_ref_input(0, copy_tensor, /* lock_held */ true);
121         if (use_exclusive_lock_) {
122           Copy(context, &copy_tensor, rhs);
123           return;
124         }
125       }
126     }
127
```

```
128       // The tensor has already been initialized and the right hand side
129       // matches the left hand side's shape. We have been told to do the
130       // copy outside the lock.
131       Tensor old_unlocked_lhs = context->mutable_input(0, /* lock_held */ false);
132       Copy(context, &old_unlocked_lhs, rhs);
133     }
134
135     virtual void Copy(OpKernelContext* context, Tensor* lhs,
136                       const Tensor& rhs) = 0;
137
138     bool use_exclusive_lock_;
139     bool validate_shape_;
140     bool relax_constraints_;
141   };
142
143   }  // end namespace tensorflow
144
145   #endif  // TENSORFLOW_CORE_KERNELS_ASSIGN_OP_H_
```