cky9301 Refactor GetInitOp() and GetAssetFileDefs() to loader_util.{cc|h} fil... ... History

1 contributor

90 lines (78 sloc) | 3.44 KB ...

```
1  /* Copyright 2016 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 #include "tensorflow/cc/saved_model/loader_util.h"
17
18 #include <vector>
19
20 #include "tensorflow/cc/saved_model/constants.h"
21 #include "tensorflow/core/lib/strings/strcat.h"
22 #include "tensorflow/core/platform/errors.h"
23 #include "tensorflow/core/platform/protobuf_internal.h"
24
25 namespace tensorflow {
26 namespace internal {
27
28 // A SavedModel may store the name of the initialization op to run in the
29 // in the SignatureDef (v2) or a collection (v1). If an init_op collection
```

```cpp
// exists, then the collection must contain exactly one op.
Status GetInitOp(const string& export_dir, const MetaGraphDef& meta_graph_def,
                 string* init_op_name) {
  const auto& sig_def_map = meta_graph_def.signature_def();
  const auto& init_op_sig_it =
      meta_graph_def.signature_def().find(kSavedModelInitOpSignatureKey);
  if (init_op_sig_it != sig_def_map.end()) {
    *init_op_name = init_op_sig_it->second.outputs()
                        .find(kSavedModelInitOpSignatureKey)
                        ->second.name();
    return Status::OK();
  }

  const auto& collection_def_map = meta_graph_def.collection_def();
  string init_op_collection_key;
  if (collection_def_map.find(kSavedModelMainOpKey) !=
      collection_def_map.end()) {
    init_op_collection_key = kSavedModelMainOpKey;
  } else {
    init_op_collection_key = kSavedModelLegacyInitOpKey;
  }

  const auto init_op_it = collection_def_map.find(init_op_collection_key);
  if (init_op_it != collection_def_map.end()) {
    if (init_op_it->second.node_list().value_size() != 1) {
      return errors::FailedPrecondition(
          strings::StrCat("Expected exactly one main op in : ", export_dir));
    }
    *init_op_name = init_op_it->second.node_list().value(0);
  }
  return Status::OK();
}

Status GetAssetFileDefs(const MetaGraphDef& meta_graph_def,
                        std::vector<AssetFileDef>* asset_file_defs) {
  // With SavedModel v2, we write asset file def into metagraph instead of
  // collection, so read from metagraph first.
  if (meta_graph_def.asset_file_def_size() > 0) {
    for (const auto& asset : meta_graph_def.asset_file_def()) {
      asset_file_defs->push_back(asset);
    }
    return Status::OK();
  }
  // Fall back to read from collection to be backward compatible with v1.
  const auto& collection_def_map = meta_graph_def.collection_def();
  const auto assets_it = collection_def_map.find(kSavedModelAssetsKey);
  if (assets_it == collection_def_map.end()) {
    return Status::OK();
  }
```

```
  const auto& any_assets = assets_it->second.any_list().value();
  for (const auto& any_asset : any_assets) {
    AssetFileDef asset_file_def;
    TF_RETURN_IF_ERROR(
        ParseAny(any_asset, &asset_file_def, "tensorflow.AssetFileDef"));
    asset_file_defs->push_back(asset_file_def);
  }
  return Status::OK();
}

}  // namespace internal
}  // namespace tensorflow
```