#include "tensorflow/core/framework/op.h"

#include "tensorflow/core/framework/shape\_inference.h"

#include "tensorflow/core/framework/op\_kernel.h"

#include <twml.h>

#include "tensorflow\_utils.h"

using namespace tensorflow;

REGISTER\_OP("DataRecordTensorWriter")

.Attr("T: list({string, int32, int64, float, double, bool})")

.Input("keys: int64")

.Input("values: T")

.Output("result: uint8")

.SetShapeFn([](::tensorflow::shape\_inference::InferenceContext\* c) {

return Status::OK();

}).Doc(R"doc(

A tensorflow OP that packages keys and dense tensors into a DataRecord.

values: list of tensors

keys: feature ids from the original DataRecord (int64)

Outputs

bytes: output DataRecord serialized using Thrift into a uint8 tensor.

)doc");

class DataRecordTensorWriter : public OpKernel {

public:

explicit DataRecordTensorWriter(OpKernelConstruction\* context)

: OpKernel(context) {}

void Compute(OpKernelContext\* context) override {

const Tensor& keys = context->input(0);

try {

// set keys as twml::Tensor

const twml::Tensor in\_keys\_ = TFTensor\_to\_twml\_tensor(keys);

// check sizes

uint64\_t num\_keys = in\_keys\_.getNumElements();

uint64\_t num\_values = context->num\_inputs() - 1;

OP\_REQUIRES(context, num\_keys == num\_values,

errors::InvalidArgument("Number of dense keys and dense tensors do not match"));

// populate DataRecord object

const int64\_t \*keys = in\_keys\_.getData<int64\_t>();

twml::DataRecord record = twml::DataRecord();

for (int i = 1; i < context->num\_inputs(); i++) {

const twml::RawTensor& value = TFTensor\_to\_twml\_raw\_tensor(context->input(i));

record.addRawTensor(keys[i-1], value);

}

// determine the length of the encoded result (no memory is copied)

twml::ThriftWriter thrift\_dry\_writer = twml::ThriftWriter(nullptr, 0, true);

twml::DataRecordWriter record\_dry\_writer = twml::DataRecordWriter(thrift\_dry\_writer);

record\_dry\_writer.write(record);

int len = thrift\_dry\_writer.getBytesWritten();

TensorShape result\_shape = {1, len};

// allocate output tensor

Tensor\* result = NULL;

OP\_REQUIRES\_OK(context, context->allocate\_output(0, result\_shape, &result));

twml::Tensor out\_result = TFTensor\_to\_twml\_tensor(\*result);

// write to output tensor

uint8\_t \*buffer = out\_result.getData<uint8\_t>();

twml::ThriftWriter thrift\_writer = twml::ThriftWriter(buffer, len, false);

twml::DataRecordWriter record\_writer = twml::DataRecordWriter(thrift\_writer);

record\_writer.write(record);

} catch(const std::exception &e) {

context->CtxFailureWithWarning(errors::InvalidArgument(e.what()));

}

}

};

REGISTER\_KERNEL\_BUILDER(

Name("DataRecordTensorWriter").Device(DEVICE\_CPU),

DataRecordTensorWriter);