#pragma once

#include <twml/Tensor.h>

#include <type\_traits>

#ifdef \_\_cplusplus

namespace twml {

// This class contains the raw pointers to tensors coming from thrift object.

class TWMLAPI RawTensor : public Tensor

{

private:

bool m\_is\_big\_endian;

uint64\_t m\_raw\_length;

public:

RawTensor() {}

RawTensor(void \*data, const std::vector<uint64\_t> &dims,

const std::vector<uint64\_t> &strides, twml\_type type, bool is\_big\_endian, uint64\_t length)

: Tensor(data, dims, strides, type), m\_is\_big\_endian(is\_big\_endian), m\_raw\_length(length) {}

bool is\_big\_endian() const {

return m\_is\_big\_endian;

}

uint64\_t getRawLength() const {

return m\_raw\_length;

}

// Extracts a slice from a tensor at idx0 along dimension 0

// Used in BatchPredictionResponse to write each slice in separate records

RawTensor getSlice(uint64\_t idx0) const {

void \*slice = nullptr;

uint64\_t raw\_length = 0;

if (getType() == TWML\_TYPE\_STRING) {

raw\_length = getStride(0);

std::string \*data = const\_cast<std::string \*>(static\_cast<const std::string\*>(getData<void>()));

slice = static\_cast<void \*>(data + raw\_length \* idx0);

} else {

raw\_length = getStride(0) \* getSizeOf(getType());

char \*data = const\_cast<char \*>(static\_cast<const char\*>(getData<void>()));

slice = static\_cast<void \*>(data + raw\_length \* idx0);

}

std::vector<uint64\_t> dims, strides;

for (int i = 1; i < getNumDims(); i++) {

dims.push\_back(getDim(i));

strides.push\_back(getStride(i));

}

return RawTensor(slice, dims, strides, getType(), m\_is\_big\_endian, raw\_length);

}

};

// Wrapper class around RawTensor to hold sparse tensors.

class TWMLAPI RawSparseTensor

{

private:

RawTensor m\_indices;

RawTensor m\_values;

std::vector<uint64\_t> m\_dense\_shape;

public:

RawSparseTensor() {

}

RawSparseTensor(const RawTensor &indices\_, const RawTensor &values\_,

const std::vector<uint64\_t> &dense\_shape\_) :

m\_indices(indices\_), m\_values(values\_), m\_dense\_shape(dense\_shape\_)

{

if (m\_indices.getType() != TWML\_TYPE\_INT64) {

throw twml::Error(TWML\_ERR\_TYPE, "Indices of Sparse Tensor must be of type int64");

}

}

const RawTensor &indices() const {

return m\_indices;

}

const RawTensor &values() const {

return m\_values;

}

const std::vector<uint64\_t>& denseShape() const {

return m\_dense\_shape;

}

};

}

#endif