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

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

#include "tensorflow/core/framework/common\_shape\_fns.h"

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

#include <chrono>

#include <thread>

using namespace tensorflow;

REGISTER\_OP("Sleep")

.Input("num\_milliseconds: int32")

.Output("sleep\_time\_in\_ms: int32")

.SetShapeFn(tensorflow::shape\_inference::ScalarShape)

.Doc(R"doc(

A tensorflow OP that sleeps for specified number of milliseconds.

This is a proxy to determine the number of inter\_op\_parallelism pool.

This is not part of the Tensorflow API as of the date of writing this

doc. Hence, a tensorflow operation is the best resort.

Input

num\_milliseconds: A scalar tensor corresponding to the number

of milliseconds the operation should sleep for

Output

sleep\_time\_in\_ms: A scalar tensor corresponding to the

actual number of milliseconds for which the operation slept

)doc");

class SleepOp : public OpKernel {

public:

explicit SleepOp(OpKernelConstruction\* context) : OpKernel(context) {}

void Compute(OpKernelContext\* context) override {

// Grab the input tensor

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

auto input = input\_tensor.flat<int32>();

// Sleep for specified milliseconds

auto start = std::chrono::high\_resolution\_clock::now();

std::this\_thread::sleep\_for(std::chrono::milliseconds(input(0)));

auto end = std::chrono::high\_resolution\_clock::now();

std::chrono::duration<double, std::milli> elapsed = end-start;

// Set the output tensor

Tensor\* output\_tensor = NULL;

OP\_REQUIRES\_OK(context, context->allocate\_output(0, TensorShape({}), &output\_tensor));

auto output\_flat = output\_tensor->flat<int32>();

output\_flat(0) = elapsed.count();

}

};

REGISTER\_KERNEL\_BUILDER(Name("Sleep").Device(DEVICE\_CPU), SleepOp);