Tensorflow Op机制详解

1026 Jiang XIAO

2019年8月3日 at pm3:23 (last edited 2020年5月26日 at pm11:50)

注册一个tfop分为两部分: **Op**和**OpKernel**. 其中, **Op**是tfop的声明部分, 类似于函数的声明, 主要描述Op静态属性. **OpKernel**是tfop的实现部分, 同样类似于函数的实现, 主要描述OpKernel的具体计算逻辑. tf已经实现的Op相关源码在tensorflow/core/common_runtime/ops/中, OpKernel的部分在tensorflow/core/common_runtime/kernels/. 二者通过注册时使用的名字联系到一起.

接口形式

下面是一个注册Op的例子, 使用接口"REGISTER_OP()"“注册一个Op,包括输入输出, Op属性等内容, 注册的Op最终会转换为OpDef, 即用protobuf格式存储的静态数据, 所以, 这里用到的属性也按照pb的格式以KV的形式编写. REGISTER_OP的实现在”tensorflow/core/framework/op.h OpDefBuilderWrapper” 中, 众多可设值中,"Attr"是一个允许自定义的值, 比如XLA引擎就根据自身需求提供了"XlaCompile", 如果一个Op将该值设置为true, 就会强制XLA引擎将其编译. 当然, 也可以设置一些无用的值, 就像函数声明里有一个并没有实际使用的参数, 除了浪费存储空间没有其他用途.

```
//tensorflow/core/common_runtime/nccl_ops.h
#include "tensorflow/core/framework/common_shape_fns.h"
#include "tensorflow/core/framework/op.h"

namespace tensorflow {

using shape_inference::InferenceContext;
using shape_inference::ShapeHandle;

REGISTER_OP("NcclAllReduce")
    .Input("input: T")
    .Output("data: T")
    .Attr("reduction: {'min', 'max', 'prod', 'sum'}")
    .Attr("T: {half, float, float64, int32, int64}")
    .Attr("num_devices: int")
    .Attr("shared_name: string")
    .Attr("XlaCompile: bool=true")
    .SetIsStateful()
    .SetShapeFn(shape_inference::UnchangedShape);
}
```

注册原理

在内部原理上, 上述代码可以进一步分解为两部分: **构造一个Op + 注册一个Op**, 这部分代码主要在“tensorflow/core/framework/ op.h(.cc) op_def_builder.h(.cc) ” 中

```
1.  //op.h
2.  #define REGISTER_OP(name) REGISTER_OP_UNIQ_HELPER(__COUNTER__, name)
3.  #define REGISTER_OP_UNIQ_HELPER(ctr, name) REGISTER_OP_UNIQ(ctr, name)
4.  #define REGISTER_OP_UNIQ(ctr, name)                                         \
5.    static ::tensorflow::register_op::OpDefBuilderReceiver register_op##ctr    \
6.        TF_ATTRIBUTE_UNUSED =                                                 \
7.            ::tensorflow::register_op::OpDefBuilderWrapper<SHOULD_REGISTER_OP( \
8.                name)>(name)
9.
10. //macros.h
11. #define TF_ATTRIBUTE_UNUSED __attribute__((unused))
```

上述代码的关键在**-5-8-**, 等号的右边是构造了一个`OpDefBuilderWrapper`对象并设置好相应的属性, 等号的左侧是构造一个static的, 利用"**__COUNTER__**"唯一标识的, `OpDefBuidlerReceiver`对象, 这个对象构造的会对带注册Op做相应的检查, 如果检查通过将其转换成`OpDef`对象, 并进一步构造为一个`OpRegistrationData`对象并存储在下面这个registry_中,

```
  mutable std::unordered_map<string, const OpRegistrationData*> registry_
```

顺便说一句, 这里面`OpDefBuilderWrapper`的实现有个小trick: 我们知道构造函数返回的是一个对象, 那么就可以通过".member_fcn()"的形式直接调用其成员函数, 这里, `OpDefBuilderWrapper`还进一步, 其相应的成员函数均是返回"*this"类型, 可以实现**“链式”调用**, 形如"OpDefBuilderWrapper instance.mem_fn0().mem_fn1()….", 也正是这种设计, 才有了注册Op时清爽的表达.

```
class OpDefBuilderWrapper<true> {
 public:
  explicit OpDefBuilderWrapper(const char name[]) : builder_(name) {}
  OpDefBuilderWrapper<true>& Attr(string spec) {
    builder_.Attr(std::move(spec));
    return *this;
  }
  OpDefBuilderWrapper<true>& Input(string spec) {
    builder_.Input(std::move(spec));
    return
  //...
```

调试方法

tf在 `OpRegistry` 中实现了很多Op操作方法, 可用于调试的主要有下面几个:

```
  Status LookUp(const string& op_type_name,
                const OpRegistrationData** op_reg_data) const override;

  // A singleton available at startup.
  static OpRegistry* Global();

  // Get all registered ops.
  void GetRegisteredOps(std::vector<OpDef>* op_defs);
```

```
  // Get all `OpRegistrationData`s.
  void GetOpRegistrationData(std::vector<OpRegistrationData>* op_data);
```

Related:
Tensorflow XLA HLO I — BufferLiveness
Tensorflow XLA Service 详解 II
Tensorflow XLA Service 详解 I
Tensorflow XLA Client | HloModuleProto 详解
Tensorflow XlaOpKernel | tf2xla 机制详解
Tensorflow JIT 技术详解
Tensorflow JIT/XLA UML
Tensorflow OpKernel机制详解
Tensorflow Op机制详解
Tensorflow Optimization机制详解
Tensorflow 图计算引擎概述

技术 Tensorflow，技术

‹ Tensorflow OptimizationPassRegistry机制详解
Tensorflow OpKernel机制详解 ›

One comment on “Tensorflow Op机制详解”

Pingback: Tensorflow 图计算引擎概述 - sketch2sky

Leave a Reply

Write a Comment...

Reply

This site uses Akismet to reduce spam. Learn how your comment data is processed.