Tensorflow OptimizationPassRegistry机制详解

983 Jiang XIAO

2019年7月28日 at pm3:39 (last edited 2021年1月31日 at pm3:19)

整个流水线大概是：Init Graph –>OptimizationPassRegistry和Grappler进行全图优化 –> 根据Device将Graph拆成Partition –> GraphOptimizer优化Partition–>图执行. 同其他注册机制一样, OptimizationPassRegistry也使用的registry, registertion以及registerar等概念.

源码中, 使用`REGISTER_OPTIMIZATION()`注册一个优化器, 具体实现如下

```
1.  //core/common_runtime/optimization_registry.h
2.  #define REGISTER_OPTIMIZATION(grouping, phase, optimization) \
3.    REGISTER_OPTIMIZATION_UNIQ_HELPER(__COUNTER__, grouping, phase, optimization)
4.
5.  #define REGISTER_OPTIMIZATION_UNIQ_HELPER(ctr, grouping, phase, optimization) \
6.    REGISTER_OPTIMIZATION_UNIQ(ctr, grouping, phase, optimization)
7.
8.  #define REGISTER_OPTIMIZATION_UNIQ(ctr, grouping, phase, optimization)          \
9.    static ::tensorflow::optimization_registration::OptimizationPassRegistration \
10.       register_optimization_##ctr(                                              \
11.           grouping, phase,                                                      \
12.           ::std::unique_ptr<::tensorflow::GraphOptimizationPass>(               \
13.               new optimization()),                                              \
14.           #optimization)
```

-12-new了一个我们注册的optimization对象并用unique_ptr指向它, 这个unique_ptr就是registry管理的对象, 通过它间接管理相应的optimization. 注册的本质是返回一个静态的, 类型为`'OptimazationPassRegistration'`的, 名为`'register_optimization_##ctr'`的对象, 这里使用了C++预编译宏`'__COUNTER__'`生成唯一变量名

下面是一个`OptimazationPassRegistration`对象的构造过程, 可以看出, 就是将我们构造的`'register_optimization_##ctr`注册到全局的`'global_optimization_registry'`.

```
1.  OptimizationPassRegistration::OptimizationPassRegistration()
2.    pass->set_name(optimization_pass_name);
3.    OptimizationPassRegistry::Global()->Register(grouping, phase, std::move(pass))
4.      //Global()
5.      static OptimizationPassRegistry* global_optimization_registry = new OptimizationP
6.      return global_optimization_registry;
7.      //Register()
```

```
8.        groups_[grouping][phase].push_back(std::move(pass));
```

-8-使用group+phase的方式对一个'OptimazationPassRegistration'对象进行分组.

对于已经注册的OptimazationPassRegistration', 使用'Rungrouping()执行之

```
1.  //optimization_registery.cc
2.  Status OptimizationPassRegistry::RunGrouping(Grouping grouping) {
3.    auto group = groups_.find(grouping);
4.    if (group != groups_.end()) {
5.      for (auto& phase : group->second) {
6.        for (auto& pass : phase.second) {
7.          Status s = pass->Run(options);
8.        }
9.      }
10.   }
11.   return Status::OK();
12. }
```

-5:9-从小到大遍历指定group下的每一个phase下的每一个pass, 由于在注册的时候指定了group和phase, 所以phase id小的一定比phase id大的先执行, 同一个group同一个phase id, 就按照注册的先后顺序执行了.

在optimization_registry的设计上, group之前并没有先后顺序, 我们可以指定直接执行任何一个group下面的optimization, 但在tensorflow的使用中, 又有一定的顺序. Tensorflow定义了下面的几个group, 分别对应着图执行的从前到后的几个阶段. 在这种使用方法下, group本身也可以作为按照时间先后执行的一个分类.

```
//optimazition_registry.h
// Groups of passes are run at different points in initialization.
enum Grouping {
  PRE_PLACEMENT,          // after cost model assignment, before placement.
  POST_PLACEMENT,         // after placement.
  POST_REWRITE_FOR_EXEC,  // after re-write using feed/fetch endpoints.
  POST_PARTITIONING,      // after partitioning
};
```

按照从前到后的执行顺序, tensorflow1.14中的优化器有:

```
REGISTER_OPTIMIZATION(OptimizationPassRegistry::PRE_PLACEMENT, 0, ParallelConcatRemovePass)
REGISTER_OPTIMIZATION(OptimizationPassRegistry::PRE_PLACEMENT, 0,AccumulateNV2RemovePass);
REGISTER_OPTIMIZATION(OptimizationPassRegistry::PRE_PLACEMENT, 0, LowerFunctionalOpsPass);
REGISTER_OPTIMIZATION(OptimizationPassRegistry::POST_PLACEMENT, 0, NcclReplacePass);
REGISTER_OPTIMIZATION(OptimizationPassRegistry::PRE_PLACEMENT, 25, IsolatePlacerInspectionRe
//跟XLA相关的9个Opt
REGISTER_OPTIMIZATION(OptimizationPassRegistry::PRE_PLACEMENT, 25, IntroduceFloatingPointJit
REGISTER_OPTIMIZATION(OptimizationPassRegistry::PRE_PLACEMENT, 26, EncapsulateXlaComputatio
REGISTER_OPTIMIZATION(OptimizationPassRegistry::PRE_PLACEMENT, 27, FunctionalizeControlFlowl
REGISTER_OPTIMIZATION(OptimizationPassRegistry::POST_REWRITE_FOR_EXEC, 5, CloneConstantsForE
REGISTER_OPTIMIZATION(OptimizationPassRegistry::POST_REWRITE_FOR_EXEC, 10, MarkForCompilatio
REGISTER_OPTIMIZATION(OptimizationPassRegistry::POST_REWRITE_FOR_EXEC, 20 IncreaseDynamismFo
REGISTER_OPTIMIZATION(OptimizationPassRegistry::POST_REWRITE_FOR_EXEC, 30,PartiallyDecluste
REGISTER_OPTIMIZATION(OptimizationPassRegistry::POST_REWRITE_FOR_EXEC, 40,EncapsulateSubgrap
REGISTER_OPTIMIZATION(OptimizationPassRegistry::POST_REWRITE_FOR_EXEC, 50,BuildXlaOpsPass);

REGISTER_OPTIMIZATION(kMklLayoutRewritePassGroup, 1, MklLayoutRewritePass);
```

```
REGISTER_OPTIMIZATION(kMklTfConvPassGroup, 2, MklToTfConversionPass);
```

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 图计算引擎概述

Tensorflow Op机制详解 ›

One comment on “Tensorflow OptimizationPassRegistry机制详解”

Pingback: Tensorflow 图计算引擎概述 - sketch2sky

Leave a Reply

TAG

Ceph Linux内核 Misc Tensorflow XLA 技术 效率工程 神经网络