智能计算系统实验一文档

BANG C算子与Tensorflow集成

1952395沈韬

包含目录问题

线程数优化

实验目的

通过使用智能编程语言(BANGC)进行算子开发，对高性能库(CNML)算子进行扩展，并 最终集成到编程框架(TensorFlow)中，掌握对高性能库及编程框架进行扩展的能力，使读 者可以在DLP 硬件上自由设计并优化满足特定应用场景的新算子，满足日新月异智能算法的实 际需求

实验背景

智能编程语言开发所需的编译工具链包括但不限于CNCC、CNGDB等。该部分详见理论课程PPT。

实验内容

算子实现:采用智能编程语言BCL实现PowerDifference算子; 算子测试:对PowerDifference算子本身进行测试，保证其功能正确; 框架集成:通过高性能库PluginOp的接口对PowerDifference算子进行封装，使其调用方式和高性能库原有算子一致，将封装后的算子集成到TensorFlow编程框架中; 框架算子测试:使用框架API测试上一步集成在TensorFlow中的算子，保证其功能正确。

实验步骤

登陆云平台

```
ssh xxx@120.236.247.203 -p xxxx
```

初始化环境

```
1. cd /opt/AICSE-demo-student/env;
2. source env.sh
3. cd /opt/AICSE-demo-
student/demo/style_transfer_bcl/src/bangc/PluginPowerDifferenceOp
```

PowerDifference BANGC 算子实现

根据PPT上的相关信息和TODO提示，补充plugin_power_difference_kernel.mlu和powerDifference.cpp文件如下

plugin_power_difference_kernel.mlu

```
// TODO: PowerDifference BCL单核实现
//#include"plugin_power_difference_kernel.h"
#define ONELINE 64
__mlu_entry__ void PowerDifferenceKernel(half* input1,half*
input2,int32_t pow,half* output,int32_t dims_a)
{
  if (taskId > 0) return;
  // TODO: 循环条件判断
  int32_t quotient = dims_a/ONELINE;
  int32_t rem = dims_a % ONELINE;
  if(rem != 0)
  {
    quotient+=1;
  }

  // TODO: 内存申请
  __nram__ half inputx_nram[ONELINE];
  __nram__ half inputy_nram[ONELINE];
  __nram__ half temp_nram[ONELINE];

  // TODO: For循环计算
  for (int i = 0; i < quotient; i++)
  {
    // TODO: 拷入操作

 __memcpy(inputx_nram,input1+i*ONELINE,ONELINE*sizeof(half),GDRAM2NRAM);

 __memcpy(inputy_nram,input2+i*ONELINE,ONELINE*sizeof(half),GDRAM2NRAM);
    // TODO: 实际计算部分
```

```
    __bang_sub(temp_nram,inputx_nram,inputy_nram,ONELINE);
    __bang_active_abs(temp_nram,temp_nram,ONELINE);
    for(int i=0;i<pow-1;i++)
    {
      __bang_mul(temp_nram,temp_nram,temp_nram,ONELINE);
    }
    // TODO: 结果拷出操作

 __memcpy(output+i*ONELINE,temp_nram,ONELINE*sizeof(half),NRAM2GDRAM);
  }
}
```

powerDiff.cpp

```
// TODO: 补充PowerDifferenceKernel参数
void PowerDifferenceKernel(half* input1,half* input2,int32_t pow,half*
output,int32_t dims_a);
#ifdef __cplusplus
}
#endif
void PowerDifferenceKernel(half* input1,half* input2,int32_t pow,half*
output,int32_t dims_a);
```

```
// TODO: 完成cnrtInvokeKernel函数
  cnrtInvokeKernel_V2((void*)&PowerDifferenceKernel,dim,params,c,pQueue)
;
```

```
// TODO: 完成cnrtMemcpy拷入函数
  cnrtMemcpy(mlu_input1,input1_half,dims_a*sizeof(half),CNRT_MEM_TRANS_D
IR_HOST2DEV);
  cnrtMemcpy(mlu_input2,input2_half,dims_a*sizeof(half),CNRT_MEM_TRANS_D
IR_HOST2DEV);
```

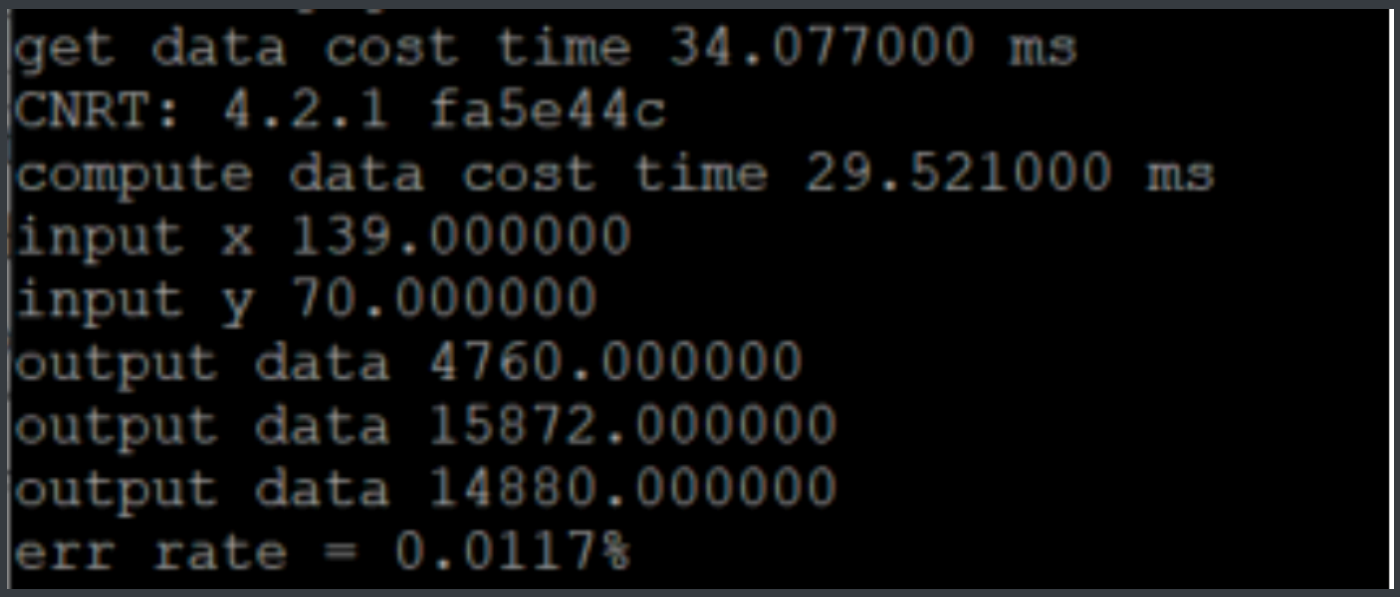
```
// TODO: 完成cnrtMemcpy拷出函数
  cnrtMemcpy(output_half,mlu_output,dims_a*sizeof(half),CNRT_MEM_TRANS_D
IR_DEV2HOST);
```

算子测试结果

在补全文件后，进行算子的测试

```
./make.sh
./power_diff_test
```

测试结果：



cnplugin集成

补全plugin_power_difference_op.cc和cnplugin.h并编译新的Cambricon-CNPlugin

plugin_power_difference_op.cc

```
cnmlStatus_t cnmlCreatePluginPowerDifferenceOpParam(
  cnmlPluginPowerDifferenceOpParam_t *param,
  int pow,
  int dims_a,
  cnmlCoreVersion_t core_version
```

```
  // TODO: 添加变量
) {
  *param = new cnmlPluginPowerDifferenceOpParam();

  int static_num = 0;//无常量数据

  (*param)->pow = pow;
  (*param)->dims_a = dims_a;
  // TODO: 配置变量

  return CNML_STATUS_SUCCESS;
}
cnmlStatus_t cnmlCreatePluginPowerDifferenceOp(
  cnmlBaseOp_t *op,
  cnmlTensor_t *pd_input_tensors,
  int pow,
  cnmlTensor_t *pd_output_tensors,
  int len
  // TODO: 添加变量
) {
  cnrtKernelParamsBuffer_t params;
  cnrtGetKernelParamsBuffer(&params);
  // TODO: 配置变量

  cnrtKernelParamsBufferMarkInput(params);
  cnrtKernelParamsBufferMarkInput(params);
  cnrtKernelParamsBufferAddParam(params,&pow,sizeof(int));
  cnrtKernelParamsBufferMarkOutput(params);
  cnrtKernelParamsBufferAddParam(params,&len,sizeof(int));
  void **InterfacePtr = reinterpret_cast<void**>
(&PowerDifferenceKernel);
  cnmlCreatePluginOp(op,
                     "PowerDifference",
                     InterfacePtr,
                     params,
                     pd_input_tensors,
```

```
                      2,
                      pd_output_tensors,
                      1,
                      nullptr,
                      0);
  cnrtDestroyKernelParamsBuffer(params);
  return CNML_STATUS_SUCCESS;
}

cnmlStatus_t cnmlComputePluginPowerDifferenceOpForward(
  cnmlBaseOp_t op,
  void *inputs[],
  void *outputs[],
  // TODO: 添加变量
  cnrtQueue_t queue
) {
  // TODO: 完成Compute函数
  cnmlComputePluginOpForward_V4(op,
                                nullptr,
                                inputs,
                                2,
                                nullptr,
                                outputs,
                                1,
                                queue,
                                nullptr);
  return CNML_STATUS_SUCCESS;
}
```

cnplugin.h

```
struct cnmlPluginPowerDifferenceOpParam{
  int pow;
  int dims_a;
  cnmlCoreVersion_t core_version;
};
```

```
typedef cnmlPluginPowerDifferenceOpParam
*cnmlPluginPowerDifferenceOpParam_t;

cnmlStatus_t cnmlCreatePluginPowerDifferenceOpParam(
  cnmlPluginPowerDifferenceOpParam_t *params,
  int pow,
  int dims_a,
  cnmlCoreVersion_t core_version
);

cnmlStatus_t cnmlDestroyPluginPowerDifferenceOpParam(
  cnmlPluginPowerDifferenceOpParam_t *params
);

cnmlStatus_t cnmlCreatePluginPowerDifferenceOp(
  cnmlBaseOp_t *op,
  cnmlTensor_t *pd_input_tensors,
  int pow,
  cnmlTensor_t *pd_output_tensors,
  int len
);

cnmlStatus_t cnmlComputePluginPowerDifferenceOpForward(
  cnmlBaseOp_t op,
  void *inputs[],
  void *outputs[],
  cnrtQueue_t queue
);
```

TensorFlow算子集成

将下述文件夹中的文件依次添加到TensorFlow源码中/opt/AICSE-demo-student/demo/style_transfer_bcl/src/tf-implementation/tf-add-power- diff;/opt/AICSE-demo-student/env/tensorflow-v1.10

```
1. cp -rf /opt/AICSE-demo-student/demo/style_transfer_bcl/src/tf-implementation/tf-add-power-diff/cwise_op_power_difference.cc /opt/AICSE-demo-student/env/tensorflow-v1.10/tensorflow/core/kernels/cwise_op_power_difference.cc
2. cp -rf /opt/AICSE-demo-student/demo/style_transfer_bcl/src/tf-implementation/tf-add-power-diff/cwise_op_power_difference_mlu.h /opt/AICSE-demo-student/env/tensorflow-v1.10/tensorflow/core/kernels/cwise_op_power_difference_mlu.h
3. cp -rf /opt/AICSE-demo-student/demo/style_transfer_bcl/src/tf-implementation/tf-add-power-diff/BUILD /opt/AICSE-demo-student/env/tensorflow-v1.10/tensorflow/core/kernels/BUILD
4. cp -rf /opt/AICSE-demo-student/demo/style_transfer_bcl/src/tf-implementation/tf-add-power-diff/mlu_stream.h /opt/AICSE-demo-student/env/tensorflow-v1.10/tensorflow/stream_executor/mlu/mlu_stream.h
5. cp -rf /opt/AICSE-demo-student/demo/style_transfer_bcl/src/tf-implementation/tf-add-power-diff/mlu_lib_ops.cc /opt/AICSE-demo-student/env/tensorflow-v1.10/tensorflow/stream_executor/mlu/mlu_api/lib_ops/mlu_lib_ops.cc
6. cp -rf /opt/AICSE-demo-student/demo/style_transfer_bcl/src/tf-implementation/tf-add-power-diff/mlu_lib_ops.h /opt/AICSE-demo-student/env/tensorflow-v1.10/tensorflow/stream_executor/mlu/mlu_api/lib_ops/mlu_lib_ops.h
7. cp -rf /opt/AICSE-demo-student/demo/style_transfer_bcl/src/tf-implementation/tf-add-power-diff/mlu_ops.h /opt/AICSE-demo-student/env/tensorflow-v1.10/tensorflow/stream_executor/mlu/mlu_api/ops/mlu_ops.h
8. cp -rf /opt/AICSE-demo-student/demo/style_transfer_bcl/src/tf-implementation/tf-add-power-diff/power_difference.cc /opt/AICSE-demo-student/env/tensorflow-v1.10/tensorflow/stream_executor/mlu/mlu_api/ops/power_difference.cc
9. cp -rf /opt/AICSE-demo-student/demo/style_transfer_bcl/src/tf-implementation/tf-add-power-diff/math_ops.cc /opt/AICSE-demo-student/env/tensorflow- v1.10/tensorflow/core/ops/math_ops.cc
```

编译TensorFlow

```
1. deactivate
2. cd /opt/AICSE-demo-student/env/tensorflow-v1.10
3. bash ./build_tensorflow-v1.10_mlu.sh
```

框架算子测试

补全.../src/online_mlu/power_difference_test_bcl.py
和.../src/online_cpu/power_difference_test_cpu.py 文件， 执行: python power_difference_test_xxx.py

power_difference_test_cpu.py

```
def power_difference_op(input_x,input_y,input_pow):
    with tf.Session() as sess:
        x = tf.placeholder(tf.float32, name='x')
        y = tf.placeholder(tf.float32, name='y')
        pow_ = tf.placeholder(tf.float32, name='pow')
        z = tf.power_difference(x,y,pow_)
        return sess.run(z, feed_dict = {x:input_x, y:input_y,
pow_:input_pow})
```

```
python power_difference_test_cpu.py
```

执行结果

```
(virtualenv_mlu) root@localhost:/opt/AICSE-demo-student/demo/style_transfer_bcl/src/online_cpu# python power_difference_test_cpu.py
WARNING:tensorflow:From power_difference_test_cpu.py:11: The name tf.Session is deprecated. Please use tf.compat.v1.Session instead.

CNML: 7.2.1 c8ada41
CNRT: 4.2.1 fa5e44c
CNML: 7.2.1 c8ada41
WARNING:tensorflow:From power_difference_test_cpu.py:13: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder
instead.

2021-05-26 21:00:28.768443: I tensorflow/core/grappler/optimizers/mlu_control_optimizer.cc:44][49022] MLUControlOptimizer, it may ta
ke a while ...
2021-05-26 21:00:28.768726: I tensorflow/core/grappler/optimizers/mlu_control_optimizer.cc:170][49022] MLUControlOptimizer end!
comput C++ op cost 136.980056763ms
comput numpy op cost 277.589082718ms
err rate= 6.076836369758652e-06
(virtualenv_mlu) root@localhost:/opt/AICSE-demo-student/demo/style_transfer_bcl/src/online_cpu# cd /opt/AICSE-demo-student/demo/styl
```

power_difference_test_bcl.py

```
    def power_difference_op(input_x,input_y,input_pow):
        with tf.Session() as sess:
            x = tf.placeholder(tf.float32, name='x')
            y = tf.placeholder(tf.float32, name='y')
            pow_ = tf.placeholder(tf.float32, name='pow')
            z = tf.power_difference(x,y,pow_)
            return sess.run(z, feed_dict = {x:input_x, y:input_y,
    pow_:input_pow})
```

```
    python power_difference_test_bcl.py
```

执行结果

```
CNML: 7.2.1 c8ada41
CNRT: 4.2.1 fa5e44c
CNML: 7.2.1 c8ada41
CNML: 7.2.1 c8ada41
2021-06-02 12:59:46.861358: I tensorflow/stream_executor/mlu/mlu_stream_executor.cc:470][110494] MLU OP Cache enabled
CNML: 7.2.1 c8ada41
2021-06-02 12:59:46.861566: I tensorflow/core/common_runtime/mlu/mlu_device.cc:396][110494] __LOG_MLU__, Current MLU_Visible_De
vices Count: 1
2021-06-02 12:59:46.861596: I tensorflow/core/common_runtime/mlu/mlu_device.cc:115][110494] __LOG_MLU__, save_offline_model: 0
2021-06-02 12:59:46.861627: I tensorflow/core/common_runtime/mlu/mlu_device.cc:121][110494] __LOG_MLU__, core_version: MLU270
2021-06-02 12:59:46.861651: I tensorflow/core/common_runtime/mlu/mlu_device.cc:123][110494] __LOG_MLU__, core_num: 1
WARNING:tensorflow:From power_difference_test_bcl.py:13: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeho
lder instead.

2021-06-02 12:59:46.884814: I tensorflow/core/grappler/optimizers/mlu_control_optimizer.cc:44][110494] MLUControlOptimizer, it
may take a while ...
2021-06-02 12:59:46.884997: I tensorflow/core/grappler/optimizers/mlu_control_optimizer.cc:170][110494] MLUControlOptimizer end
!
2021-06-02 12:59:46.886717: I tensorflow/compiler/tf2mlu/convert/convert_graph.cc:654][110494] MLU fusion node PowerDifference/
my_mlu_op_0 added for segment 0 consisting of 1 nodes succeeded.
comput BCL op cost 123.975992203ms
comput op cost 228.213071823ms
err rate= 0.09103545480047298
```

实验心得

MLU加速比

32核MLU相对于30核CPU的加速比：$$S = \frac{T_{cpu}}{T_{mlu}}$$

由公式得，实验中：

- 自己的算子在MLU上运行相比于在CPU上运行的加速比为：（137/124）=1.10
- numpy算子在MLU上运行相比于在CPU上运行的加速比为：（277.6/228.2）=1.22，比自

己手写算子效果更好

结论

- MLU加速不明显
- numpy算子在MLU下加速比自己写的算子更明显

分析原因

- 平台提供的CPU架构为ARM架构，而非常见的x86架构。ARM架构的CPU在服务器领域同功耗同价格下可以提供更多的核心数，实验中为30核。远高于常见的8核、16核处理器。CPU核心数的增加使的MLU的加速不明显。
- 查询资料知，调用MLU加速的过程中，每一次计算过程中都存在将待计算数据从内存复制到MLU片上存储、计算完毕将结果复制回内存的过程，且两次复制过程中都会触发 CPU 中断，这对于本就以毫秒计的计算时间而言是不可忽视的时间开销。资料中提供的解决方法是分批次将尽可能多的待计算的数据复制到片上存储 ，减少总的复制次数，从而减少复制和中断带来的额外时间开销。在某些文章里称之为初始化。
- numpy算在效果比手写算子加速效果更好，猜测可能与上述初始化过程有关，也可能是由于numpy本身是由 python实现的高性能计算库，可优化空间比C++大的多。

算子性能优化

根据课上所学MLU基础架构，以及算子的优化方法，对PowerDifference算子进行如下优化

- 多核优化：在循环条件判断部分，根据输入变量维度将输入拆分成多个部分，多核执行，提高效率
- 维度上限优化：通过多次调整MAX_SIZE进行测试，在64、128、256、512、1024这五个值中，取512作为MAX_SIZE可以获得最高的运行效率

TensorFlow编译

包含目录问题

在tensorflow编译过程中，出现报错**XXXX is not declared**. 通过GNU调试，发现默认环境中的包含库目录的路径设置与教程中所声明的位置不符。由于env.sh中的路径设置与许多默认调用相关，故不敢改env.sh中的包含库目录的路径。检查tensorflow源码发现，第三方插件库中的头文件为空，故尝试用打包的cnplugin替换原来为空的头文件，即

```
1. cp -f /opt/AICSE-demo-student/env/Cambricon-CNPlugin-MLU270/build/libcnplugin.so /opt/AICSE-demo-student/env/tensorflow-v1.10/third_party/mlu/lib
2. cp -f /opt/AICSE-demo-student/env/Cambricon-CNPlugin-MLU270/pluginops/PluginPowerDifferenceOp/cnplugin.h /opt/AICSE-demo-student/env/tensorflow-v1.10/third_party/mlu/include/
```

解决问题；

同时，网上现有教程中文件的复制并不全面，需要按照文件中给出的README文件内容进行文件的复制

线程数优化

tensorflow编译过程中有时会发生**socket closed**错误，出现位置不固定。搜寻资料后发现原因为容器可用内存限制为30G，原先的编译脚本中设定的编译线程数为32，编译时峰值占用内存为38G，超过了内存限制，被系统杀死所致。

将其配置文件中的**jobs_num**修改为16即可