BANGC 算子与风格迁移实验手册（上）

一、BANGC 算子实现与 TensorFlow 的集成

1.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。

2.背景介绍：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。

3.实验内容：

a) 算子实现：采用智能编程语言 BCL 实现 PowerDifference 算子;
b) 算子测试：对 PowerDifference 算子本身进行测试，保证其功能正确;
c) 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;
d) 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。

4.实验步骤：

a) 登录云平台：ssh xxx@120.236.247.203 -p xxx
b) 初始化环境：cd /opt/AICSE-demo-student/env; source env.sh
c) cd /opt/AICSE-demo-student/demo/style_transfer_bcl/src/bangc/PluginPowerDifferenceOp
d) PowerDifference BANGC 算子实现，补全 plugin_power_difference_kernel.h 和 plugin_power_difference_kernel.mlu 文件。
e) PowerDifference BANGC 算子测试，补全 powerDiff.cpp 文件，执行./make.sh

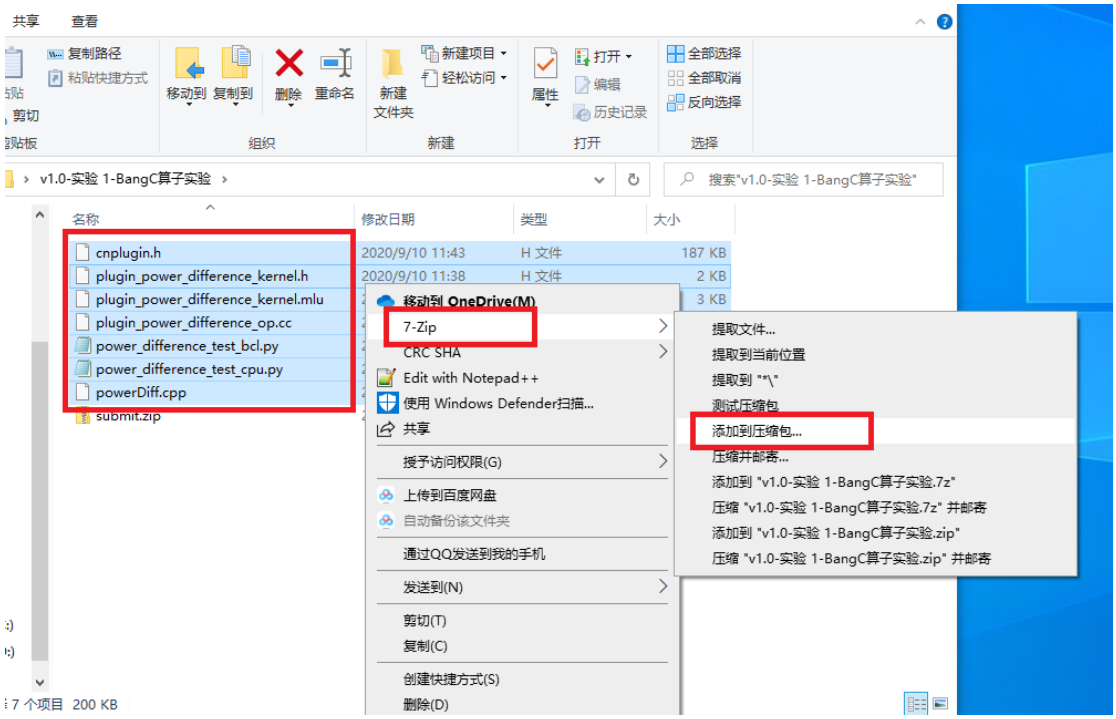
f) cnplugin 集成：补全 plugin_power_difference_op.cc 和 cnplugin.h 并编译新的 Cambricon-CNPlugin。

g) TensorFlow 算子集成，将下述文件夹中的文件依次添加到 TensorFlow 源码中（由于课程时间关系，该部分代码直接给出）：/opt/AICSE-demo-student/demo/style_transfer_bcl/src/tf-implementation/tf-add-power-diff ；/opt/AICSE-demo-student/env/tensorflow-v1.10

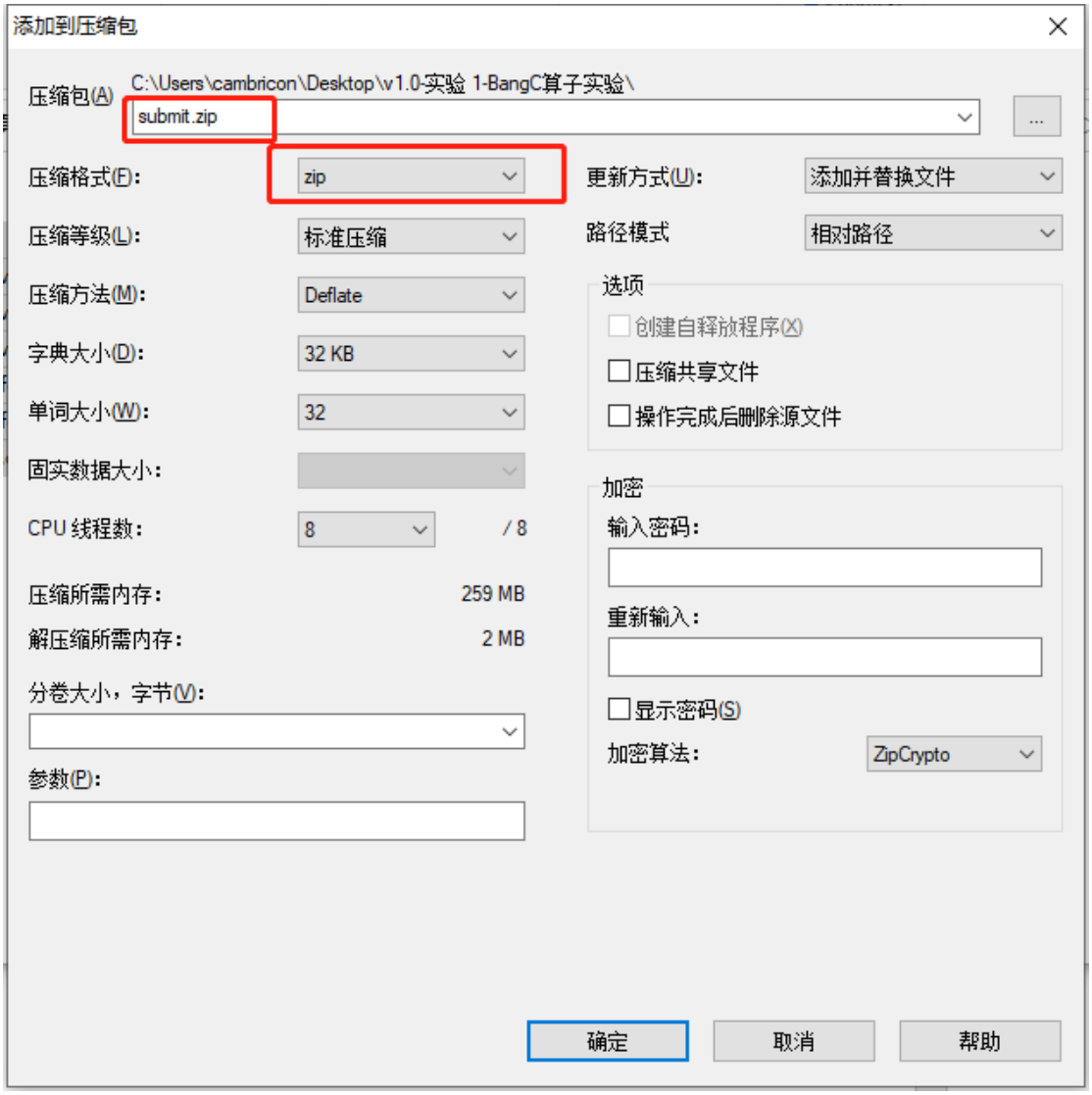
h) 框架算子测试，补全 .../src/online_mlu/power_difference_test_bcl.py 和.../src/online_cpu/power_difference_test_cpu.py 文件，执行：python power_difference_test_xxx.py

5. 上传自动评测平台

a) 将程序按以下文件列表整理，打包成压缩文件。请注意文件名和图中保持一致。



b) 压缩文件格式为 zip 格式。文件名命名为 submit.zip



c） 将 submit.zip 上传到评测平台。选择文件并点击提交。

d） 上传之后请等待系统评测。

使用 python 在 CPU 端精度 误差在 0.1%以内，延时不做要求;在 MLU 端精度误差在 10%以内，平
90分 标准：在80分基础上，使用 python 在 MLU 端测试大规模数据时 (256*256*3)精度误差在 1%以内，平均延
(提示:BangC 中 需要使用多核拆分计算)；
100分标准：在90分基础上，使用python在MLU端测试超大规模数据时(例 如 256*256*16、512*512*32 且幂指数
精度误差在 0.1%以内，延时显著优于 CPU。

e） 评测结束之后会自动显示评测结果。

CNRT评测结果	
CNRT平均延时(ms)	CNRT平均误差(%)
34.357	0.0117

CPU评测结果
CPU端平均误差(%)
5.962194950664888e-06

MLU评测结果					
小测试集延时(ms)	小测试集误差(%)	大测试集延时(ms)	大测试集误差(%)	超大测试集延时(ms)	超大测试集误差(%)
53.106639120322214	0.07531791643127603	51.588945918600004	0.09202048668630224	52.58133676316666	0.07490939958929513

附录：BANGC 参考示例

BangC例子

例子1

输入tensor x，输出

$$ln(x)$$

计算方法

使用牛顿迭代法，ln(x)的值等价于求解方程

$$e^t = x$$

中t的值，令

$$f(t) = e^t - x$$

并选

$$\varphi(t) = t - \frac{f(t)}{f'(t)}$$

为迭代公式，则有

$$t_{n+1} = t_n - \frac{f(t_n)}{f'(t_n)} = t_n - \frac{e^{t_n} - x}{e^{t_n}} = t_n - 1 + xe^{-tn}$$

```
#include "mlu.h"

#define ONELINE 64
#define N  6
__mlu_entry__ void LnKernel(half* input, half* output, int32_t len)
{
  int32_t quotient = len / ONELINE;
  int32_t rem = len % ONELINE;
  int32_t nk = 2*N +1;
  __nram__ half input_nram[ONELINE];
  __nram__ half temp1_nram[ONELINE];
  __nram__ half temp2_nram[ONELINE];
  __nram__ half const_nram[ONELINE];

  __nramset_half(const_nram, ONELINE, 1);
  __nramset_half(temp2_nram, ONELINE, 0.0);
  if (rem != 0)
  {
    quotient +=1;
  }

  for (int32_t i = 0; i < quotient; i++)
  {
    __memcpy(input_nram, input + i * ONELINE, ONELINE * sizeof(half), GDRAM2NRAM);
    __nramset_half(temp1_nram, ONELINE, 7.0);//t0=0
    for (int32_t j = 1; j <= N;j++)
    {
      __bang_mul_const(temp2_nram, temp1_nram, -1.0, ONELINE);  // -tn
      __bang_active_exp(temp2_nram, temp2_nram, ONELINE);       // exp(-tn)
      __bang_mul(temp2_nram, input_nram, temp2_nram, ONELINE);  // x*exp(-tn)
      __bang_add(temp1_nram, temp1_nram, temp2_nram, ONELINE);  // tn + x*exp(-tn)
      __bang_sub(temp1_nram, temp1_nram, const_nram, ONELINE);  // tn + x*exp(-tn) -1

    }
    __memcpy(output +i * ONELINE, temp1_nram, ONELINE * sizeof(half), NRAM2GDRAM);
  }
}
```