开发者社区

FastPT 使用手册

DAS

目　录

1 FastPT 简介

FastPT 是基于 python 的应用编译工具，借助 FastPT，开发人员可以在 HCU 上开发、部署基于 pytorch 的内含 CUDA 代码的应用。可以实现基于 CUDA 源码的不转码编译处理，或源码转换到 HIP 格式代码后，通过 hipcc 适配编译处理。推荐使用不转码编译方式，参考 3.1 章节，转码适配方式功能受限，可能需要手动处理较多的内容。

表 1： Fastpt 与 torch 版本之间的版本对应关系

Fastpt 版本	Torch 版本	DTK 版本
2.0.1+das.dtk2504	v2.4.1	dtk2504
2.1.0+das.dtk2504	v2.5.1	dtk2504

注意：

1. 本工具目前尚不支持 cutlass 类似三方库和内联汇编指令；
2. 编译有依赖库需求且依赖库涉及到 GPU 加速实现的，可以在 DAS 上以及 GPUFusion（一般为$ROCM_PATH/cuda/)环境下查询是否有已适配或支持的库。
3. 此工具适合生态组件应用依赖 torch 的且有 GPU 加速代码实现的场景下使用，如通过 CUDAExtension 构建编译或编译依赖 libtorch 等。

2 安装

工具安装使用 pip 方式，安装前请确保环境中已安装了 torch，并从光源社区-DAS 中下载此工具的安装包。注意与 python、torch 版本匹配。Torch 需要使用 HCU 下支持的版本。

```
cd path/to/whl
pip install fastpt*.whl  --no-deps
```

安装完成之后，可通过以下指令验证是否安装成功,指令执行后会显示当前 FastPT 的版本号。

```
python -c “import fastpt;print(fastpt.__version__)”
```

3 使用

推荐使用不转码编译方式，可参考下面的表格以及 3.1 章节的内容。

工具安装后,构建编译或使用时。通过 source /usr/local/bin/fastpt -X 进行环境设置。X 为模式设置参数，具体参数说明如下：

表 2：环境设置参数说明

使用场景	指令	示例	说明
不转码编译	-C 或 -c	source /usr/local/bin/fastpt -C	用于工程不转码编译处理的环境设置，由于在编译模式下需要设置部分环境变量，所以在打开新的终端进行编译处理时，需要执行此命令。
	-E 或 -e	source /usr/local/bin/fastpt -E	不转码编译的程序执行时环境设置。用于工程在不转码编译后，进行使用时的环境设置。工程迁移到新环境后，安装 FastPT 后，执行此命令即可。此命令不需要重复执行，只需保证当前系统下执行过即可。
转码编译	-T 或 -t	source /usr/local/bin/fastpt -T	转码编译处理。用于通过转码方式，将 CUDA 代码转换到 HIP 代码后的编译实现。只用于组件或程序编译处理，组件执行时不需要额外配置环境。
帮助	-H 或 -h	source /usr/local/bin/fastpt -H	具体指令说明，查询使用方法。

3.1 不转码编译

3.1.1 使用方法

编译模式：Fastpt-2.0 之后支持不转码编译实现，即直接使用 CUDA 源码编译。安装 FastPT 工具的 whl 包，执行：source /usr/local/bin/fastpt -C，初始化 FastPT 编译环境，然后按照组件或应用的官方指导编译方法处理即可。

执行模式：编译好组件通过 whl 包在有 FastPT 的新环境下使用时，需要确保环境下已执行过执行模式的初始化操作，否则可能会报错找不到 CUDA 相关的动态库。执行命令：source /usr/local/bin/fastpt -E 即可。

3.1.2 不转码编译注意事项

（1）此工具适用于依赖 torch 的生态组件使用，应用中内含 CUDA C/C++代码的

工程在 HCU 环境下的开发、移植。Fastpt 版本应与 torch 版本对应，当前 torch 版本要求最低 torch-2.4.1；

（2）不支持依赖 cutlass、内嵌汇编代码的编译；

（3）部分代码可能存在不支持的情况，如__CUDA_ARCH__宏，可以在代码中设置或在编译指令中添加支持，可设置 arch 对应 sm_75；

（4）部分编译指令不支持，如下示例中的”-gencode”，”arch=compute_75,code=sm_75”需要使用”-gencode=arch=compute_75,code=sm_75”替换。

```
extra_compile_args={
            "cxx": [
                "-O3",
            ]
            + generator_flag,
            "nvcc": [
                "-O3",
                "--use_fast_math",
                #"-gencode",
                #"arch=compute_75,code=sm_75",
                '-gencode=arch=compute_75,code=sm_75',
              ]
            + generator_flag,
        },
```

（5） 编译模式(-C)与执行模式(-E)下，torch.version.cuda 与 torch.version.hip 会分别被设置，少部分应用在执行时会依赖这两个变量，需要根据具体情况在应用端调整上述两个变量。

3.2 转码编译

FastPT 提供了 HCU 下，转码到 HIP 格式，通过 hipcc 进行编译的方法，实现基于 torch 的应用中 CUDA 代码移植到 HCU 平台，工具接口包括 CUDAExtension、CppExtension、hipify 转码接口，相关接口说明参考：PyTorch documentation。编译时，转码一般是自动实现的。另外提供了自定义接口映射用来补充代码映射关系（见 3.2.2）；提供了保持源码文件夹下文件相对路径的转码方法（见 3.2.3）。

3.2.1 使用方法

此方法适用于通过 setup.py 使用 CUDAExtension、CppExtension 进行组件

构建编译的场景。使用时，执行 source /usr/local/bin/fastpt -T。参考组件官方的构建文档，进行编译处理，3.2.4 章节中提供了一些注意事项，仅作参考。

3.2.2 自定义接口映射

工具中可能存在未涉及到或用户需要的一些转换匹配，可以通过 json 文件的方式给到工具，在不需要额外修改代码的情况下，实现自定义代码匹配转换。可以通过以下方法补充代码映射。用户需要在 CUDAExtension 或 hipify_python 接口调用代码同级目录下，通常为 setup.py 文件所在目录，创建 custom_hipify_mappings.json 文件。json 文件内容格式如下：

```
{
    "custom_map" : {
        "src mapping 1" : "dst mapping 1",
        "src mapping 2" : "dst mapping 2",
        ...
    }
}
```

此示例中，在将 CUDA 代码转到 HIP 代码时，会将源码中的”src mapping 1”替换为”dst mapping 1”，将"src mapping 2" 替换成 "dst mapping 2"。

自定义映射转换优先级高于内置的转换。

3.2.3 转码接口

工具提供了 hipify 转码接口，使用可参考 torch 中的同名接口。

此接口还提供了实现保留源代码路径的转码的处理方法，转码会新建一个 xxx_dtk 的代码文件夹，内部文件路径结构与源文件夹下一致，”.cu”代码文件扩展名会转成”.hip”，如 xxx.cu 转码后为 xxx.hip。此方法一般在 CMake 或 Make 等需要保持原代码路径的工具编译代码时使用，需要注意 CMakeLists.txt 中的代码需要手动修改，暂时不支持 CMake 语法的转码，如”.cu”后缀需要改为”.hip”、编译器 nvcc 需要改为 hipcc 等，此适配场景下建议使用 3.1 章节中的不转码编译处理。

使用时，需要在要转码的文件夹的同级目录中实现以下脚本 fastptcode.py：

```
import os
this_dir = os.path.dirname(os.path.abspath(__file__))
from  fastpt import hipify
res = hipify(
```

```
        project_directory=os.path.join(this_dir,'codepath'),
        includes = '*',
        show_detailed=True,
        is_pytorch_extension=True,
        add_dtk_macros=False, # False: 不引入 ATen/dtk_macros.h 头文件，默认为 True
        keep_file_path=True  # True:保持源代码文件夹内文件相对路径;默认 False
        # ignores=["run_tests.sh.in"]  # 屏蔽掉不希望处理的代码
    )
```

执行：

python fastptcode.py

会在 codepath 的同级目录下生成转码后的文件夹 codepath_dtk，此路径下的文件结构与 codepath 的结构相同，".cu"代码文件后缀变为".hip"。文件结构如下：

```
|----codepath
|----codepath_dtk
|----fastptcode.py
```

3.2.4 转码编译注意事项

（1）此工具适用于依赖 torch 的生态组件或应用，内含 CUDA C/C++代码的工程在 HCU 环境下的开发、移植，注意 FastPT 版本与 torch 版本对应;

（2）暂不支持 CMake、make 等代码语义处理。代码转换可通过上面 3.2.3 中的示例，通过执行 python 的转码脚本代码，将 CUDA 代码转换成 HIP 代码，CMake 文件需要用户自行处理。为了便于适配建议通过 CMake 编译的组件使用 3.1 章节的不转码的方式，通过 CUDA 源码编译的方式适配;

（3）工程中存在三方依赖库时，三方库可能存在不被处理的情况，此时需要对三方依赖库进行单独处理;

（5）适配组件的 setup.py 中可能会有 CUDA 环境检查来决定是否执行 CUDA 相关代码的编译，例如 CUDA_PATH 或 torch.version.cuda 的检查，可按情况进行处理;

（6）当不希望引入 ATen/dtk_macros.h 这个头文件时，可以通过以方式屏蔽此头文件的引入：export FASTPT_DTK_MACROS=1。

保密声明

在此声明，该文档展示的全部技术信息及其相关内容，版权皆属于开发者社区（https://developer.hpccube.com/）所有。未经允许，严禁截屏、大规模传播及转发。另外，对使用该技术文档而导致任何侵犯第三方专利或其他权利的行为，开发者社区不承担任何责任。

感谢您的理解与支持。