

第二篇 自己动手，给FlagPerf添加一个测试case

书接前回，我们在这里[《自己动手，给FlagPerf添加一款AI框架》](#)介绍了如何给FlagPerf添加一款AI框架。这次呢，我们给各位喜欢自己动手的小伙伴们准备了开发教程。欢迎大家一起给FlagPerf添砖加瓦！

1. FlagPerf架构和代码结构简介

在动手之前，咱们先一起看看FlagPerf的整体架构，以便了解工作内容。FlagPerf目前是这样子的：

最右侧是自动化测试工具，包括了整个benchmark任务生命周期管理，包括每个测试任务的容器环境准备、任务启动、任务运行过程中系统信息的收集、等待任务结束后的日志收集等等。咱们添加一个新测试case时，在这部分主要包括模型定义和训练相关逻辑即可。其余部分皆有通用的模块支持实现。

左侧上面的蓝色框内是标准化的benchmark测试集。包括标准化的模型训练benchmark case实现，和多个框架的支持。这里也涉及到AI框架的添加，主要在通用的benchmark driver中实现AI框架的driver支持。同样的，训练中打印日志、注册和处理Event等已有通用的模块处理，用户完全不用操心。如果你觉得哪里能做得更完善，也可以一并修改了提PR哦~

左侧绿色的“异构芯片模型适配优化层”是异构芯片厂商需要适配的啦，咱们这里可以不考虑。

这里的标准case（以下简称case）是指**模型+AI框架**的一个组合。那么，咱们要添加一个case，具体要做什么呢？咱们先看这张FlagPerf目录结构图，下载代码后，在training目录下执行tree -L 3命令就可以看到（这里只列出咱们关心的哈）：

```
|—— benchmarks
|   |—— bert # bert模型
|   |   |—— README.md # 模型readme
|   |   |—— paddle # bert-paddle case
|   |—— cpm # CPM模型
|   |   |—— pytorch # cpm-pytorch case
|   |—— driver # FlagPerf框架driver模块，完成一些通用逻辑的封装
|   |—— glm # GLM模型
```

- | | | | — README.md # 模型readme
- | | | | — pytorch # GLM-pytorch case
- | | — nvidia # **nvidia机器上的case运行配置**
- | | | | — bert-paddle
- | | | | | — config
- | | | | — cpm-pytorch
- | | | | | — config # **建议包括1x1,1x2,1x4,1x8,2x8的配置，下同**
- | | | | | — environment_variables.sh #容器内设置环境变量的脚本。如有需要，请添加。
- | | | | | — requirements.txt # case所依赖的包
- | | | | — extern
- | | | — docker_image # 构建容器镜像所需文件，每个框架一个目录
- | | | | — paddle
- | | | | — pytorch
- | | | | — tensorflow1
- | | | | | — tensorflow2
- | | | — glm-pytorch
- | | | | — config
- | | | | — extern
- | | — nvidia_monitor.py # nvidia机器的监控项采集程序
- | — requirements.txt # FlagPerf依赖包
- | — run_benchmarks # 启动训练任务的主体程序所在路径。
- | | | — config # 启动训练任务的配置目录
- | | | | — cluster_conf.py # 集群配置：节点的ip列表和ssh端口
- | | | | — test_conf.py # case配置：FlagPerf的部署路径和运行的cases根据测试需要修改
- | | | — paddle # 每个框架一个目录，里面有启动对应框架任务的脚本

start_<framework>_task.py，下同

- | | | | — start_paddle_task.py
- | | | — prepare_in_container.py # 启动训练任务前初始化容器环境
- | | | — pytorch
- | | | | — start_pytorch_task.py
- | | — run.py # 启动和管理任务的入口程序

```
|   └── tensorflow2
|       └── start_tensorflow2_task.py
└── utils # 测试任务管理的工具模块
```

以上，涉及新增case的部分，主要是以下几个部分

- 构建case业务逻辑：这里主要包括模型、训练逻辑、训练入口程序run_pretraining.py
- 添加case配置：这里包括case基本配置和nvidia适配的配置
- 调试验证case：运行测试并修改case，直到达标
- 添加相关文档：模型和case readme

那么接下来，咱们就以添加resnet50-pytorch的case为例，讲讲具体怎么做。其中，对容器环境的准备，基于nvidia的GPU环境完成，已经在上篇文章《[自己动手，给FlagPerf添加一款AI框架](#)》中有过介绍，在此不再赘述。

2. 自己动手，新增case

2.1 构建case业务逻辑

1. 实现模型训练主体逻辑

在training/benchmarks下添加<model>/<framework>子目录，pytorch和paddle的标准case可参考下面的目录结构组织代码：

```
├── config # 【必选】 case基本配置目录
├── dataloaders # 定义dataset和dataloader
├── model # 定义模型
├── optimizers # 定义优化器, fp16_optimizer, loss_scaler等
├── readme.md # 【必选】 case文档
├── run_pretraining.py # 【必选】 执行训练入口脚本，可以使用
dev/docs/run_pretaining.example.py作为模版
├── schedulers # 定义LR scheduler
└── train # 定义训练主要逻辑.trainer, evaluator, training_state等
```

其它框架如tensorflow2等的参考目录结构待有待后续补充。

2. 实现训练入口程序

建议复制docs/dev/run_pretraining.py.example作为training/benchmarks/<model>/<framework>/run_pretraining.py脚本，根据该脚本中的标记TODO的位置进行修改，串接整个训练pipeline，是不是很方便~

2.2 添加配置文件

涉及标准Case实现的配置有2个：基本配置 和 nvidia适配的配置。

基本配置是模型训练及运行环境相关的配置参数，nvidia适配的配置是指标准Case运行在Nvidia GPU环境的配置参数。后者在运行时会覆盖标准Case的基础配置参数。

由于FlagPerf的一些代码设定，对配置文件路径和内容有一定要求。

a) case基本配置

i) 标准Case基本配置

路径：<model>-<framework>/config/_base.py。定义模型训练相关的所有参数，及case运行环境需要的基本参数。模型名称、模型结构、数据集、checkpoint、超参数、分布式参数等。

配置项说明如下

必选参数：

- vendor，值为"nvidia"即可，会在运行FlagPerf时被配置在test_conf里的vendor值覆盖。
- data_dir，值为"/home/datasets"即可，会在运行FlagPerf时被配置在test_conf中对应case配置项data_dir_container覆盖

可选参数：

- train_data：训练数据路径，填写相对data_dir的路径
- eval_data：评估数据路径，填写相对data_dir的路径
- init_checkpoint：初始化模型checkpoint，填写相对data_dir的路径

其它模型训练相关参数，例如初始learning rate等，可参考docs/dev/standard-case-config-base.py.example

ii) 可改写配置项

路径：<model>-<framework>/config/mutable_params.py。定义厂商（含nvidia）可覆盖的_base中参数列表。主要是和vendor和运行环境相关的配置项，定义为mutable_params数组。

厂商可以在training/<vendor>/<model>-<framework>/config/config_xxxx.py中，重新定义参数值，从而实现对于case的基本配置_base.py配置参数的覆盖。 例如：mutable_params = ['vendor',

'local_rank', 'train_batch_size'], 其中vendor为必选项。

b) nvidia适配的配置

在Nvidia GPU上运行所需的配置文件放在training/nvidia/<model>-<framework>/config目录下，可以看作是nvidia适配标准Case的配置项，由于训练规模和训练方法不同，可以给出多个配置文件。

建议包含：1x1, 1x2, 1x4, 1x8, 2x8的配置。

在FlagPerf运行时，会根据test_conf里的case配置项选择加载哪个配置文件。配置文件命名规范为：config_<machine_model>X<nnodes>X<nproc>.py。例如单机4卡的A100环境运行，使用config_A100x1x4.py，这里主要放置是厂商适配case时可覆盖的参数，一般定义在自己设备上跑该case最优的配置。

此外，如果该标准Case在预先构建的nvidia镜像中无法直接运行，需要一定的环境配置和依赖包安装，请添加environment_variables.sh和requirements.txt。

2.3 测试验证case

3.1 修改测试的配置

仅仅需要修改两个配置文件，位于training/run_benchmarks/config路径下，就可以跑起来一个测试case了。

cluster_conf.py 中修改HOSTS为实际运行FlagPerf的节点的IP，SSH_PORT设置为实际的ssh端口，一般可以使用默认值22，无需修改。

test_conf.py中一般需要修改两处。

FLAGPERF_PATH_HOST：改为FlagPerf在主机上的部署路径

CASES：设置为希望测试的case列表。

备注：如果该模型的配合不存在，则需要添加。只需要执行utils/gen_dummy_benchmark.py脚本，给出厂商、框架名、配置模块名和数据目录几个参数，即可生成一个空的模型case配置。具体参见上一篇文章 [《自己动手，给FlagPerf添加一款AI框架》](#) 中的第3节。

3.2 启动测试

一条命令即可启动一组测试，在修改好配置的服务器上，进入training目录，运行如下命令 python3 ./run_benchmarks/run.py

可看到测试执行过程，为了防止断网等情况，推荐使用nohup python3 ./run_benchmarks/run.py & 来启动测试

观察日志输出，看看是否达到 测试达标要求。如不达标，修改case，直到满足测试达标要求。测试达标要求参考 [《标准case规范》](#) 的 3.6节。

2.4 添加相关文档

模型README文档（首次添加该模型的case时，需要填写）及 case README文档 符合文档模版要求。文档模版请参考：[模型readme模版](#)和[case readme模版](#)。

经过以上四个步骤，恭喜你，成功的完成了一个标准case的添加流程。赶快给FlagPerf项目提交PR吧，感谢您为FlagPerf项目做出的贡献！