支持 Colossal AI 的训练在线监控工具创新赛
——测试报告

团队名称：今天你科研了吗

测试 1：Colossal AI 框架与晟腾 NPU 的兼容性测试

我们基于 Ascend 官方提供的迁移方式将针对 GPU 编写的代码迁移到 NPU 上，输出的文件在 code/llama2_npu 文件夹中给出。选择 modelscope 社区的[沐雪（中文）训练集用于 Llama2 的训练。该数据集并不能直接用于预训练，需要进行预处理。模型下载和预处理后，命令行调用 Sft.npu.sh 即可开始 Llama2 的训练，具体代码如下所示：

```
#!/bin/bash
export PYTHONWARNINGS='ignore:semaphore_tracker:UserWarning'
# export
MODELSCOPE_CACHE="/root/.cache/000030/Reference/ColossalAI-main/applications/Colossal-LLaMA_msft
_multi/Modelscope_cache"
# export
MODELSCOPE_CACHE="Reference/ColossalAI-main/applications/Colossal-LLaMA_msft_multi/Modelscope_c
ache"
export MASTER_ADDR=127.0.0.1
export MASTER_PORT=29688
export HCCL_WHITELIST_DISABLE=1
NPUS=($(seq 0 7))
export RANK_SIZE=${#NPUS[@]}
rank=0
i=0
export DEVICE_ID=${i}
export RANK_ID=${rank}
echo run process ${rank}
PROJECT_NAME="llama2_sft"
PARENT_SAVE_DIR="save/"
PARENT_TENSORBOARD_DIR="tensorboard/"
PARENT_CONFIG_FILE="config"
PRETRAINED_MODEL_PATH="/root/.cache/modelscope/hub/colossalai/Colossal-LLaMA-2-7b-base"
declare -a dataset=(
    "/root/.cache/000030/Reference/ColossalAI-main/applications/Colossal-LLaMA_msft_multi/dataset/arrow/par
t-00000"
    "/root/.cache/000030/Reference/ColossalAI-main/applications/Colossal-LLaMA_msft_multi/dataset/arrow/par
t-00001")
TIMESTAMP=$(date +%Y-%m-%d-%H-%M-%S)
FULL_PROJECT_NAME="${PROJECT_NAME}-${TIMESTAMP}"
SAVE_DIR="${PARENT_SAVE_DIR}${FULL_PROJECT_NAME}"
TENSORBOARD_DIR="${PARENT_TENSORBOARD_DIR}${FULL_PROJECT_NAME}"
CONFIG_FILE="${PARENT_CONFIG_FILE}${FULL_PROJECT_NAME}.json"
```

```
colossalai run --nproc_per_node 2 --master_port 29688   train_npu.py \
    --pretrained $PRETRAINED_MODEL_PATH \
    --dataset ${dataset[@]} \
    --plugin "ddp" \
    --save_interval 400 \
    --save_dir $SAVE_DIR \
    --tensorboard_dir $TENSORBOARD_DIR \
    --config_file $CONFIG_FILE \
    --num_epochs 1 \
    --accumulation_steps 8 \
    --lr 5e-5 \
    --mixed_precision "fp16" \
    --grad_clip 1.0 \
    --weight_decay 0.01 \
    --warmup_steps 100 \
    --use_flash_attn \
    --pad_token "eos"\
    --lora_rank 1\
    --batch_size 1
```

测试发现，Llama2 可以正常基于该数据集训练，验证了相关代码的正确性和可行性。接着，测试基于 Collossal AI 框架的推理功能，首先从开源社区下载预训练的 Llama2 大模型，代码如下

```
modelscope download --model colossalai/Colossal-LLaMA-2-7b-base
```

类似地，命令行调用 Infer.npu.sh 即可测试 Llama2 的推理效果，具体代码如下：

```
#!/bin/bash
export PYTHONWARNINGS='ignore:semaphore_tracker:UserWarning'
export
MODELSCOPE_CACHE="/root/.cache/000030/Reference/ColossalAI-main/applications/Colossal-LLaMA_msft
_multi/Modelscope_cache"
export MASTER_ADDR=127.0.0.1
export MASTER_PORT=29688
export HCCL_WHITELIST_DISABLE=1
NPUS=($(seq 0 7))
export RANK_SIZE=${#NPUS[@]}
rank=0
i=0
export DEVICE_ID=${i}
export RANK_ID=${rank}
echo run process ${rank}
PROJECT_NAME="llama2_test"
PARENT_SAVE_DIR="/root/.cache/000030/Reference/ColossalAI-main/applications/Colossal-LLaMA_msft_mu
lti/save"
```

```
PARENT_TENSORBOARD_DIR="/root/.cache/000030/Reference/ColossalAI-main/applications/Colossal-LLaMA_msft_multi/tensorboard"
PARENT_CONFIG_FILE="/root/.cache/000030/Reference/ColossalAI-main/applications/Colossal-LLaMA_msft_multi/config"
PRETRAINED_MODEL_PATH=""
declare -a dataset=(
    "PATH TO THE DATASET"
)
TIMESTAMP=$(date +%Y-%m-%d-%H-%M-%S)
FULL_PROJECT_NAME="${PROJECT_NAME}-${TIMESTAMP}"
SAVE_DIR="${PARENT_SAVE_DIR}${FULL_PROJECT_NAME}"
TENSORBOARD_DIR="${PARENT_TENSORBOARD_DIR}${FULL_PROJECT_NAME}"
CONFIG_FILE="${PARENT_CONFIG_FILE}${FULL_PROJECT_NAME}.json"
colossalai run --nproc_per_node 2 --master_port 29688   Infer_npu.py
```

测试发现，基于 Collossal AI 框架可以成功完成 Llama2 的推理任务。兼容 Collossal AI 的在线监控工具尚在开发过程当中，后续我们将给出具体的实现方式。

测试 2：梯度监控工具 CoMonitor 功能正确性与用户体验测试

配置文件设置如下：

```
{
    "targets": [],
    "format": "yaml",
    "mode": "ddp",
    "ops": ["norm","max"]
}
```

统计聚合前和聚合后梯度如下：

聚合前

```
max: 1.6308345038851257e-06
norm: 1.0331401426810771e-05
param_name: base_model.model.module.model.layers.26.self_
rank: 1
---
max: 0.0181514210999012
norm: 0.31125572323799133
param_name: base_model.model.module.model.layers.25.self_
rank: 1
```

聚合后

```
max: 3.5047924029640853e-05
norm: 0.00010367669892730191
param_name: module.module.base_model.model.module.mod
---
max: 0.03474785014986992
norm: 0.31352993845939636
param_name: module.module.base_model.model.module.mod
---
max: 0.0031788889318704605
```

测试 3：梯度监控性能和 memory 占用测试

（1）对训练速度的影响

注入前

```
Epoch 0: 100%|          | 93/93 [02:37<00:00,  1.69s/it, Loss=6.8174]
```

注入后

```
Epoch 0: 100%|          | 93/93 [02:47<00:00,  1.81s/it, Loss=6.8177]
```

速度延迟 6.4%

（2）对 Memory 的额外占用

注入前

```
+---------------------------+---------------+----------------------------------------------------+
| NPU     Chip              | Process id    | Process name             | Process memory(MB)      |
+===========================+===============+====================================================+
| 6       0                 | 598829        | python3                  | 29550                   |
| 6       0                 | 598830        | python3                  | 144                     |
+===========================+===============+====================================================+
| 7       0                 | 598830        | python3                  | 29549                   |
+===========================+===============+====================================================+
```

注入后

```
+---------------------------+---------------+----------------------------------------------------+
| NPU     Chip              | Process id    | Process name             | Process memory(MB)      |
+===========================+===============+====================================================+
| 6       0                 | 587046        | python3                  | 30208                   |
| 6       0                 | 587047        | python3                  | 144                     |
+===========================+===============+====================================================+
| 7       0                 | 587047        | python3                  | 31326                   |
+===========================+===============+====================================================+
```

Memory 额外占用 6.0%