Megatron-LM

1. 下载并构建相关的镜像，已经构建好在联调环境的nvcr.io/nvidia/pytorch:23.08-py3

注：22.01版本没有transformer-engine，23.12版本引入了新的问题（memory_efficient）

```
nvcr.io/nvidia/pytorch:23.08-py3
sshd
```

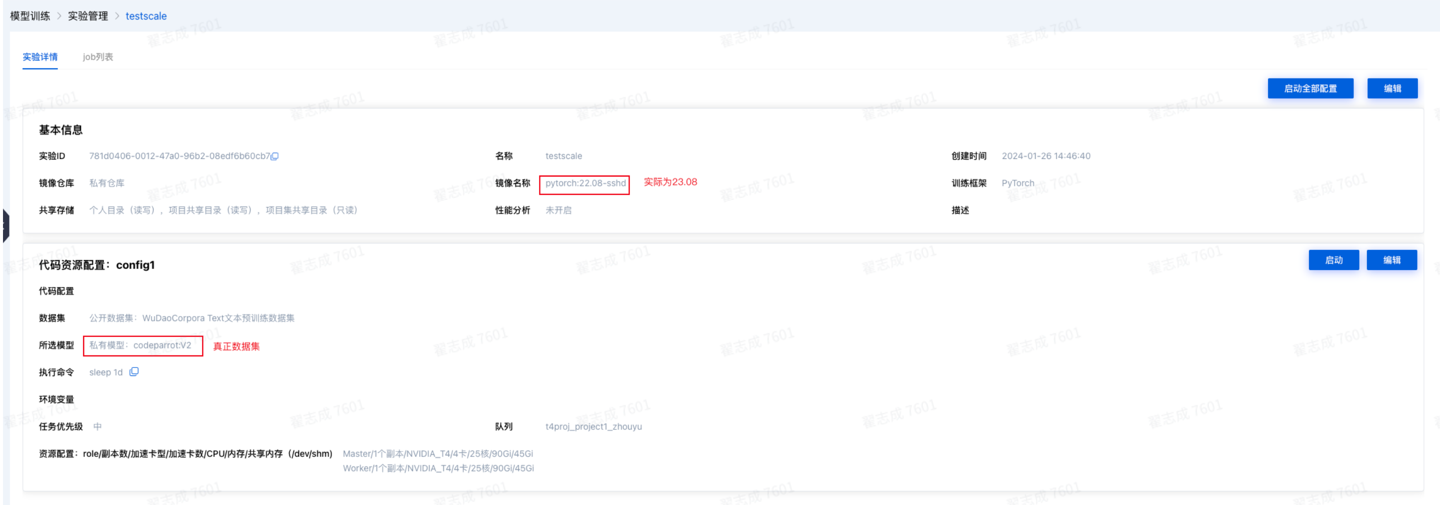
2. 下载Megatron-LM代码，

```
git clone git@github.com:NVIDIA/Megatron-LM.git
#https://github.com/NVIDIA/Megatron-LM
```

3. 下载数据集

```
https://huggingface.co/datasets/codeparrot/codeparrot-clean
```

4. 启动配置



基本信息

实验ID 781d0406-0012-47a0-96b2-08edf6b60cb7

名称 testscale

创建时间 2024-01-26 14:46:40

镜像仓库 私有仓库

镜像名称 pytorch:22.08-sshd 实际为23.08

训练框架 PyTorch

共享存储 个人目录（读写），项目共享目录（读写），项目集共享目录（只读）

性能分析 未开启

描述

代码资源配置：config1

代码配置

数据集 公开数据集：WuDaoCorpora Text文本预训练数据集

所选模型 私有模型：codeparrot:V2 真正数据集

执行命令 sleep 1d

环境变量

任务优先级 中

队列 t4proj_project1_zhouyu

资源配置：role/副本数/加速卡型/加速卡数/CPU/内存/共享内存（/dev/shm） Master/1个副本/NVIDIA_T4/4卡/25核/90Gi/45Gi
Worker/1个副本/NVIDIA_T4/4卡/25核/90Gi/45Gi

5. 下载相关文件（Megatron-LM目录下）

```
wget --content-disposition
https://s3.amazonaws.com/models.huggingface.co/bert/gpt2-vocab.json -O gpt2-
```

```
vocab.json
wget --content-disposition https://s3.amazonaws.com/models.huggingface.co/bert/gpt2-merges.txt -O gpt2-merges.txt
```

6. 数据预处理

```
pip install nltk

python tools/preprocess_data.py \
       --input codeparrot_data.json \
       --output-prefix codeparrot \
       --vocab-file gpt2-vocab.json \
       --tokenizer-type GPT2BPETokenizer \
       --merge-file gpt2-merges.txt \
       --json-keys content \
       --workers 32 \
       --append-eod
```

7. 启动脚本（2机4卡）

注：启动时要将最后一个rank的role最后启动，否则会有问题

```
export CUDA_DEVICE_MAX_CONNECTIONS=1
export NCCL_DEBUG=INFO
export NCCL_IB_DISABLE=1
export NCCL_SOCKET_IFNAME=eth0

GPUS_PER_NODE=4 #role中有几张卡
MASTER_ADDR=${MASTER_ADDR}
MASTER_PORT=${MASTER_PORT}
NNODES=2
NODE_RANK=${RANK}
WORLD_SIZE=$(($GPUS_PER_NODE*$NNODES))

DISTRIBUTED_ARGS="
     --nproc_per_node $GPUS_PER_NODE \
     --nnodes $NNODES \
     --node_rank $NODE_RANK \
     --master_addr $MASTER_ADDR \
     --master_port $MASTER_PORT"

PARALELLE_ARGS="
    --tensor-model-parallel-size 1 \
    --pipeline-model-parallel-size 1"

DATA_ARGS="
    --save /home/zhaizhicheng/Megatron-LM/experiments/codeparrot-small
    --load /home/zhaizhicheng/Megatron-LM/experiments/codeparrot-small
    --vocab-file gpt2-vocab.json
    --merge-file gpt2-merges.txt
    --data-path codeparrot_content_document"


GPT_ARGS="
    --num-layers 3
    --hidden-size 192
    --num-attention-heads 3
    --seq-length 256
    --max-position-embeddings 256
    --micro-batch-size 3
    --global-batch-size 48
    --lr 0.0005
    --train-iters 150000
    --lr-decay-iters 150000
    --lr-decay-style cosine
    --lr-warmup-iters 2000
    --weight-decay .1
    --adam-beta2 .999
    --fp16
    --log-interval 10
    --save-interval 2000
    --eval-interval 200
    --eval-iters 10"

TENSORBOARD_ARGS="--tensorboard-dir experiments/tensorboard"
torchrun $DISTRIBUTED_ARGS \
        pretrain_gpt.py \
        $PARALELLE_ARGS \
        $GPT_ARGS \
        $DATA_ARGS \
        $TENSORBOARD_ARGS
```

（4机2卡）

```
export CUDA_DEVICE_MAX_CONNECTIONS=1
export NCCL_DEBUG=INFO
export NCCL_IB_DISABLE=1
export NCCL_SOCKET_IFNAME=eth0
```

```

GPUS_PER_NODE=2 #role中有几张卡
MASTER_ADDR=${MASTER_ADDR}
MASTER_PORT=${MASTER_PORT}
NNODES=${WORLD_SIZE}
NODE_RANK=${RANK}
WORLD_SIZE=$(($GPUS_PER_NODE*$NNODES))

DISTRIBUTED_ARGS="
     --nproc_per_node $GPUS_PER_NODE \
     --nnodes $NNODES \
     --node_rank $NODE_RANK \
     --master_addr $MASTER_ADDR \
     --master_port $MASTER_PORT"

PARALELLE_ARGS="
    --tensor-model-parallel-size 1 \
    --pipeline-model-parallel-size 1"

DATA_ARGS="
    --save /home/zhaizhicheng/Megatron-LM/experiments/codeparrot-small
    --load /home/zhaizhicheng/Megatron-LM/experiments/codeparrot-small
    --vocab-file gpt2-vocab.json
    --merge-file gpt2-merges.txt
    --data-path codeparrot_content_document"


GPT_ARGS="
    --num-layers 3
    --hidden-size 192
    --num-attention-heads 3
    --seq-length 256
    --max-position-embeddings 384
    --micro-batch-size 3
    --global-batch-size 72
    --lr 0.0005
    --train-iters 150000
    --lr-decay-iters 150000
    --lr-decay-style cosine
    --lr-warmup-iters 2000
    --weight-decay .1
    --adam-beta2 .999
    --fp16
    --log-interval 10
    --save-interval 2000
    --eval-interval 200
    --eval-iters 10"
```

```

TENSORBOARD_ARGS="--tensorboard-dir experiments/tensorboard"
#python -m torch.distributed.run $DISTRIBUTED_ARGS \
torchrun $DISTRIBUTED_ARGS \
        pretrain_gpt.py \
        $PARALELLE_ARGS \
        $GPT_ARGS \
        $DATA_ARGS \
        $TENSORBOARD_ARGS
```

【参考文档】

知乎 https://zhuanlan.zhihu.com/p/633160974

如何使用 Megatron-LM 训练语言模型

在 PyTorch 中训练大语言模型不仅仅是写一个训练循环这么简单。我们通常需要将模型分布在多个设备上，并使用许多优化技术以实现稳定高效的训练。Hugging Face…

【相关代码】

https://huggingface.co/spaces/zhaizhch/Megatron-LM/tree/main

huggingface.co