开源解决方案与 IRONBYTE 专有软件性能评估方法

内容

1. 简介

本文件阐述了测试方法及其执行顺序，用于评估 IRONBYTE 自主软件解决方案与开源解决方案效能的程序。

1.1 测试目标

测试目标为：

- 使用 IRONBYTE 自主开发的软件解决方案和开源解决方案完成 LLM 领域的典型计算任务。

1.2 技术设备与基础设施

技术设备：

- **Rig Nvidia H100** – 包含 8 块 Nvidia H100 GPU（80GB）的节点，以下简称“节点”；
- **Rig Nvidia H100 *2** – 由两个节点（Rig0、Rig1）组成的集群，每个节点包含 8 块 Nvidia H100 GPU（80GB），以下简称“集群”。

网络参数：

- 带宽为 10Gbit/s 的网络，提供节点之间的通信
- 带宽为 80Gbit/s 的网络，为在两个节点上运行的开源解决方案提供通信支持。

1.3 运行环境

每个节点均已准备好独立运行应用程序（所有必要的库、驱动程序和服务器已预安装）。

软件配置：

1. 操作系统：AlmaLinux 9.5；
2. NVIDIA 驱动程序：版本不低于 550；
3. CUDA 库：版本不低于 12.2；

4. 容器服务：Docker;
5. 包含必要 ML 库和框架（如 python/torch/numpy 等）的容器集；
6. 预加载的模型和数据集。

用于运行 ML 任务的预安装数据：

1. 用于分布式训练的调度程序，支持 IRONBYTE 自主开发的软件解决方案；
2. 包含 IRONBYTE 自主开发软件解决方案的容器；
3. 包含开源解决方案的容器。

1.4 使用的模型描述

测试中使用了以下大型语言模型：

1. **Llama-2-70b-chat-hf**（量化至 4-bit），以下简称 Llama2-70B；
2. **Llama-2-7b-chat-hf**（原始大小），以下简称 Llama2-7B。

1.5 典型任务描述

测试计算节点时选择了以下三种典型任务：

1. 使用 fineweb-edu 数据集对模型进行预训练（Pretraining）；
2. 使用 databricks-dolly-15k 数据集对模型进行微调（Finetuning）；
3. 加载模型推理并生成文本（推理（秒））。

1.6 训练方法

1.6.1 使用开源解决方案的训练方法：

1. 将模型分布在所有显卡上（balanced 模式）；

2. FullyShardedDataParallel（FSDP）模式：数据和模型参数的分段模式。在加速模式下（实验性功能），支持多节点运行。此方案适用于基于 PyTorch 并发布于 Hugging Face 平台的 LLM 模型。该模式对网络交互速度要求较高。

1.6.2 使用 IRONBYTE 自主开发软件解决方案的训练方法：

1. 将模型和数据集分布到不同的显卡组和节点上，并在计算的最终阶段对结果进行混合（IRONBYTE Share Protocol）。该协议为低速率和高延迟网络设计，没有限制，适用于任何模型。

2.1 测试阶段

测试分为两个阶段进行：

第一阶段：

1. 在单个节点上使用开源解决方案执行典型计算任务；
2. 在两个节点上使用开源解决方案执行典型计算任务；

第二阶段：

1. 在两个节点上使用 IRONBYTE 自主开发的软件解决方案执行典型计算任务；
2. （可选）在单个节点上使用 IRONBYTE 自主开发的软件解决方案执行典型计算任务。

2. 测试试验的执行顺序

2.1 第一阶段（开源）

2.1.1 Llama2-70B 模型的测试

2.1.1.1 在单节点上运行 Llama2-70B 模型的预训练任务

任务在单节点上运行。使用 Llama2-70B 模型，在 fineweb-edu 10BT 数据集上进行预训练，训练 1000 步。

1. 启动容器的命令：

```
docker run -it --rm -v $HOME/pretrain70:/root -v /opt/dekubelocal:/share:ro --gpus all -w /root registry.cn-chengdu.aliyuncs.com/ai-palm/rrllm-cu124:pretrain.004
```

2. 启动预训练任务的命令（无需创建“空”模型）[1]：

```
time python3 /wrk/run_clm_no_trainer.py --dataset_name=/share/datasets/fineweb-edu/sample/10BTsingle/ --config_name=/wrk/70BlankModel --max_train_steps=1000  --output_dir=/root/result70b  --blank_model_name_or_path=/share/models/LLAMA2/metaAi/TMP70  --use_quantization=Yes
```

任务结果：

1. 训练完成的模型（1000 步）保存至目录 $HOME/pretrain70s/result70b；
2. 记录完成预训练任务所花费的时间。

如有需要，可使用工具 **nvtop** 监控计算资源的使用情况。

[1] “空”模型是指其权重随机填充的模型。可以通过指定以下键值来创建它：--create_model_from_config=True

2.1.1.2 在集群模式（FSDP）下运行 Llama2-70B 模型的预训练任务。

任务在集群的两个节点上运行。使用 Llama2-70B 模型，在 fineweb-edu 10BT 数据集上进行预训练，训练 100 步 [2]。

1. **准备工作**（可选）：克隆启动文件。由于启动过程较复杂，需通过 **docker compose** 启动任务。需将文件克隆至每个节点，并在 compose 文件中配置正确的节点名称和 IP 地址。

```
GIT_SSL_NO_VERIFY=true git clone https://gitlab.clive.tk/clients/job-manifests-public
```

1. 在 rig0 节点以交互模式[3]启动容器的命令：

```
Cd llama2-70b-fsdp-two 节点
docker compose -f interactive-compose-a.yaml up &
docker exec -it llama2a
```

2. 在 rig1 节点启动容器的命令：

```
Cd llama2-70b-fsdp-two 节点
docker compose -f interactive-compose-b.yaml up &
docker exec -it llama2b
```

3. 在两个节点上启动预训练任务的命令：

```
/mnt/wrk/entrypoint.sh
```

任务结果：

1. 训练完成的模型（100 步）保存至目录 `$HOME/pretrain70/result70b-fsdp`；
2. 记录完成预训练任务所花费的时间。

---

[2] 选择较少的训练步骤是因为节点之间的 10 Gbit/s 网络连接不足以支持在每次误差函数计算迭代中进行高强度的反向梯度传输。在此配置下，每一步需要 375 秒来完成。

[3]要禁用交互模式，需删除带有参数的键 -f。

如有需要，可使用工具 **nvtop** 和 **bmon** 监控计算资源的使用情况。

2.1.1.3 在单节点上运行 Llama2-70B 模型的微调任务：

任务在单节点上运行。使用 Llama2-70B 模型，在 databricks-dolly 数据集上进行微调，训练 500 步。

1. 启动容器的命令：

```
docker run -it --rm -v $HOME/finetune70:/root -v /opt/dekubelocal:/share:ro --gpus all --pull=always registry.cn-chengdu.aliyuncs.com/ai-palm/rrllm-cu122:finetune
```

2. 启动微调任务的命令：[4]

```
time python3 /wrk/ft_70full_n_h100_1ep.py --model_dir=/share/models/LLAMA2/metaAi/Llama-2-70b-chat-hf --output_dir=/root/full --num_step=500  --dataset_pth=/share/datasets/databricks/databricks-dolly-15k.jsonl
```

任务结果：

1. 微调完成的模型（500 步）保存至目录 $HOME/finetune70/full/；
2. 记录完成微调任务所花费的时间。

如有需要，可使用工具 **nvtop** 监控计算资源的使用情况。

2.1.1.4 在集群模式（FSDP）下运行 Llama2-70B 模型的微调任务：

任务在集群的两个节点上运行。使用 Llama2-70B 模型，在 databricks-dolly 数据集上进行微调，训练 100 步。

[4] 要指定训练时间（以周期为单位），需要使用键 --num_epoch。

1. **准备工作**（可选）：克隆启动文件。由于启动过程较复杂，需通过 **docker compose** 启动任务。需将文件克隆至每个节点，并在 compose 文件中配置正确的节点名称和 IP 地址。

```
GIT_SSL_NO_VERIFY=true git clone https://gitlab.clive.tk/clients/job-manifests-public
```

2. 在 rig0 节点以交互模式[5]启动容器的命令：

```
Cd llama2-70b-fsdp-two 节点-ft
docker compose -f interactive-compose-a.yaml up &
docker exec -it llama2a
```

3. 在 rig1 节点启动容器的命令：

```
Cd llama2-70b-fsdp-two 节点-ft
docker compose -f interactive-compose-b.yaml up &
docker exec -it llama2b
```

4. 在集群的两个节点上启动微调任务的命令：

```
/mnt/wrk/entrypoint.sh
```

任务结果：

1. 微调完成的模型（100 步）保存至目录 $HOME/finetune70/result70b-fsdp.
2. 记录完成微调任务所花费的时间。

如有需要，可使用工具 **nvtop** 和 **bmon** 监控计算资源的使用情况。

2.1.1.5 在单节点上运行 Llama2-70B 模型的推理任务：

任务在单节点上运行。使用 Llama2-70B 模型，针对固定问题生成一定数量的 Token。

[5] 要禁用交互模式，需删除带有参数的键 -f。

任务参数：

- 固定问题：**Can there be life somewhere in the Solar System outside Earth?**
- 问题重复次数：30 次；
- 每次回答的 Token 数量：100 到 400（可变）；
- 生成的 Token 总数：7750。

任务指标：

- 计算生成所有 Token 所需的总时间；
- 评估生成单个 Token 的速度；
- 计算每秒生成的 Token 数量。

执行步骤：

1. 启动容器的命令：

```
docker run -ti --rm -v /opt/dekubelocal/models/LLAMA2/metaAi/Llama-2-70b-chat-hf:/model -v wrk -e MODEL_DIR='/model' --gpus all registry.cn-chengdu.aliyuncs.com /ai-palm/env:torch24cu124.002
```

2. 启动生成推理结果的命令：

```
export MAX_TOKENS_LST='100,110,120,130,140,150,160,170,180,190,200,210,220,230,240,250,260,270,280,290,300,310,320,330,340,350,360,370,380,390,400' QUESTION='Can there be life somewhere in the Solar System outside Earth?'

python3 in_fer.py
```

任务结果：

1. 记录每次问题生成答案的时间；
2. 记录答案生成时间与其长度的关系；
3. 计算生成单个 Token 的平均时间。

如有需要，可使用工具 **nvtop** 监控计算资源的使用情况。

2.1.2 Llama2-7B 模型的测试

2.1.2.1 启动 Llama2-7B 模型预训练任务的步骤。

任务在单节点上运行。预训练任务使用 Llama2-7B 模型（不进行量化），在 fineweb-edu 10BT 数据集上训练 1000 步。

1. 启动容器的命令：

```
docker run -it --rm -v $HOME/pretrain:/root -v /opt/dekubelocal:/share:ro --
gpus all -w /root registry.cn-chengdu.aliyuncs.com/ai-palm/rrllm-
cu124:pretrain.004
```

2. 动预训练任务的命令（无需创建“空”模型）：

```
time python3 /wrk/run_clm_no_trainer.py --
dataset_name=/share/datasets/fineweb-edu/sample/10BTsingle/ --
config_name=/wrk/7BlankModel  --max_train_steps=1000  --
output_dir=/root/result7b  --
blank_model_name_or_path=/share/models/LLAMA2/metaAi/TMP7
```

任务结果：

1. 训练完成的模型（1000 步）保存至目录 $HOME/pretrain/result7b/.
2. 记录完成预训练任务所花费的时间。

如有需要，可使用工具 **nvtop** 监控计算资源的使用情况。

2.1.2.2 在集群模式（FSDP）下运行 Llama2-7B 模型的预训练任务。

任务在集群的两个节点上运行。预训练任务使用 Llama2-7B 模型，在 fineweb-edu 10BT 数据集上训练 100 步。

1. **准备工作**（可选）：克隆启动文件。由于启动过程较复杂，需通过 docker compose 启动任务。需将文件克隆至每个节点，并在 compose 文件中配置正确的节点名称和 IP 地址。

```
GIT_SSL_NO_VERIFY=true git clone https://gitlab.clive.tk/clients/job-
manifests-public
```

2. 在 rig0 节点以交互模式[6]启动容器的命令：

```
Cd llama2-7b-fsdp-two 节点
docker compose -f interactive-compose-a.yaml up &
docker exec -it llama2a
```

3. 在 rig1 节点启动容器的命令：

```
Cd llama2-7b-fsdp-two 节点
docker compose -f interactive-compose-b.yaml up &
docker exec -it llama2b
```

4. 在两个集群节点上运行预训练任务的命令：

```
/mnt/wrk/entrypoint.sh
```

任务结果：

1. 训练完成的模型（100 步）保存至目录 `$HOME/pretrain7/result7b-fsdp`
2. 记录完成预训练任务所花费的时间。

如有需要，可使用工具 **nvtop** 和 **bmon** 监控计算资源的使用情况。

2.1.2.3 启动 Llama2-7B 模型微调任务的步骤。

任务在单节点上运行。微调任务使用 Llama2-7B 模型（不进行量化），在 databricks-dolly 数据集上训练 1 个周期（epoch）。

1. 启动容器的命令：

```
docker run --rm -it --name finetune7 --gpus all -v /opt/dekubelocal:/share:ro
-v $HOME/finetune:/root registry.cn-chengdu.aliyuncs.com/ai-palm/rrllm-
cu122:finetune.001
```

2. 启动微调任务的命令：

[6] 要禁用交互模式，需删除带有参数的键 -f。

```
time python3 ft_7full_many_gpu.py --
model_dir=/share/models/LLAMA2/metaAi/Llama-2-7b-chat-hf/ --num_epoch=1 --
dataset_pth=/share/datasets/databricks/databricks-dolly-15k.jsonl --
output_dir=/root/result7b
```

任务结果：

1. 微调完成的模型（1 个周期）保存至目录 $HOME/finetune/result7b；
2. 记录完成微调任务所花费的时间。

如有需要，可使用工具 **nvtop** 监控计算资源的使用情况。

2.1.2.4 启动 Llama2-7B 模型推理任务的步：

任务在单节点上运行。推理任务使用 Llama2-7B 模型，针对固定问题生成一定数量的 Token。

任务参数：

- 固定问题：**`Can there be life somewhere in the Solar System outside Earth?`**
- 问题重复次数：30 次；
- 每次回答的 Token 数量：100 至 400（可变）；
- 生成的 Token 总数：7750。

任务指标：

- 计算生成所有 Token 所需的总时间；
- 评估生成单个 Token 的速度；
- 计算每秒生成的 Token 数量。

执行步骤：

1. 启动容器的命令：

```
docker run -ti --rm -v /opt/dekubelocal/models/LLAMA2/metaAi/Llama-2-7b-chat-hf:/model -v wrk -e MODEL_DIR='/model' --gpus all registry.cn-chengdu.aliyuncs.com /ai-palm/env:torch24cu124.002
```

2. 启动生成推理结果的命令：

```
export MAX_TOKENS_LST='100,110,120,130,140,150,160,170,180,190,200,210,220,230,240,250,260,270,280,290,300,310,320,330,340,350,360,370,380,390,400' QUESTION='Can there be life somewhere in the Solar System outside Earth?'

python3 in_fer.py
```

任务结果：

1. 记录每次问题生成答案的时间；
2. 记录答案生成时间与其长度的关系；
3. 计算生成单个 Token 的平均时间。

如有需要，可使用工具 **nvtop** 监控计算资源的使用情况。

2.2 第二阶段测试（IRONBYTE 专有软件解决方案）

2.2.1 Llama2-70B 模型测试

2.2.1.1 在集群中运行 Llama2-70B 的预训练任务

任务在集群的两个节点上运行。预训练使用 Llama2-70B 模型，在 fineweb-edu 10BT 数据集上训练 1000 步。

1. 在其中一个节点上启动调度器命令：

```
python ./orch.py --gpu_type H100 -n rig0.IRONBYTE:2x4 rig1.IRONBYTE:2x4 -t pretrain --num_step 1000 -m rig0.IRONBYTE --dataset_pth /opt/dekubelocal/datasets/fineweb-edu/sample/10BT/000_00000.parquet
```

2. *可选：任务也可在单节点上运行。删除启动参数中的第二个节点。例如：

```
python ./orch.py --gpu_type H100 -n rig0.IRONBYTE:2x4 -t pretrain --num_step
1000 -m rig0.IRONBYTE --dataset_pth /opt/dekubelocal/datasets/fineweb-
edu/sample/10BT/000_00000.parquet
```

任务结果：

1. 训练 1000 步的模型，保存在启动节点的目录：

   /opt/pyorch/{timestamp}/。

2. 记录使用调度器执行预训练任务所花费的时间。

如有必要，可使用 nvtop 工具在所有节点上监控计算资源利用率。

2.2.1.2 在集群中运行 Llama2-70B 的微调任务

任务在集群的两个节点上运行。微调使用 Llama2-70B 模型，在 databricks-dolly 数据集上训练 500 步。

1. 在其中一个节点上启动调度器命令：

```
python ./orch.py --gpu_type H100 -n rig0.IRONBYTE:2x4 rig1.IRONBYTE:2x4 -t
finetune --num_step 500 -m rig0.IRONBYTE
```

2. *可选：任务也可在单节点上运行。删除启动参数中的第二个节点。例如：

```
python ./orch.py --gpu_type H100 -n rig0.IRONBYTE:2x4 -t finetune --num_step
500 -m rig0.IRONBYTE
```

任务结果：

1. 微调 500 步的模型，保存在启动节点的目录：/opt/pyorch/{timestamp}/。
2. 记录使用 IRONBYTE 自研解决方案完成微调任务所花费的时间。

如有必要，可使用 nvtop 工具监控计算资源利用率。

2.2.1.3 在集群中运行 Llama2-70B 的推理任务。

任务在集群的两个节点上运行。推理使用 Llama2-70B 模型，根据固定问题生成一定数量的 Tokens。

任务参数：

- 固定问题：**Can there be life somewhere in the Solar System outside Earth?**
- 问题重复次数：30 次
- 每个答案的 Token 数量：100 到 400 个（可变值）
- 生成的总 Token 数量：7750

执行步骤：

1. 在两个节点上启动多次推理命令：

```
./orch.py -t 推理（秒） -n rig0.IRONBYTE:4x2 rig1.IRONBYTE:4x2 --model_dir
/opt/dekubelocal/models/LLAMA2/metaAi/Llama-2-70b-chat-hf --max_Tokens_from
100 --max_Tokens_to 401 --question 'Can there be life somewhere in the Solar
System outside Earth?'
```

2. *可选：在单节点上运行推理任务（用于与单线程开源解决方案的性能对比）：

```
./orch.py -t 推理（秒） -n rig0.IRONBYTE:1x8 --model_dir
/opt/dekubelocal/models/LLAMA2/metaAi/Llama-2-70b-chat-hf --max_Tokens_from
100 --max_Tokens_to 401 --question 'Can there be life somewhere in the Solar
System outside Earth?'
```

任务结果：

1. 每个问题的答案生成时间。
2. 根据答案长度记录生成时间。
3. 平均每个 Token 的生成时间。

如有必要，可使用 `nvtop` 工具监控计算资源利用率。

2.2.2 Llama2-7B 模型测试

2.2.2.1 在集群中运行 Llama2-7B 的预训练任务。

任务在集群的两个节点上运行。预训练使用 Llama2-7B 模型，在 fineweb-edu 10BT 数据集上训练 2000 步。

1. 在两个节点上通过调度器启动任务：

```
python ./orch.py --gpu_type h100 -n rig0.IRONBYTE:4x2 rig1.IRONBYTE:4x2 -t
pretrain --num_step 2000 -m rig0.IRONBYTE --dataset_pth
/opt/dekubelocal/datasets/databricks
```

2. *可选：任务也可在单节点上运行。删除启动参数中的第二个节点。例如：

```
python ./orch.py --gpu_type H100 -n rig0.IRONBYTE:4x2 -t pretrain --num_step
2000 -m rig0.IRONBYTE --dataset_pth /opt/dekubelocal/datasets/databricks
```

任务结果：

1. 训练 2000 步的模型，保存在启动节点的目录：/opt/pyorch/{timestamp}/。
2. 记录使用 IRONBYTE 自研解决方案完成预训练任务所花费的时间。

如有必要，可使用 `nvtop` 工具监控计算资源利用率。

2.2.2.2 在集群中运行 Llama2-7B 的微调任务：

任务在集群的两个节点上运行。微调使用 Llama2-7B 模型，在 databricks-dolly 数据集上训练 1 个周期。

1. 在两个节点上通过调度器启动任务：

```
python ./orch.py --gpu_type h100 -n rig0.IRONBYTE:4x2 rig1.IRONBYTE:4x2 -t
finetune --num_epoch 1 -m rig0.IRONBYTE --num_epoch 1 --dataset_pth
/opt/dekubelocal/datasets/databricks
```

2. *可选：任务也可在单节点上运行。删除启动参数中的第二个节点。例如：

```
python ./orch.py --gpu_type h100 -n rig0.IRONBYTE:4x2 -t finetune --num_epoch
1 -m rig0.IRONBYTE --num_epoch 1 --dataset_pth
/opt/dekubelocal/datasets/databricks
```

任务结果：

1. 微调 1 个周期的模型，保存在启动节点的目录 /opt/pyorch/{timestamp}/。
2. 记录使用 IRONBYTE 自研解决方案完成微调任务所花费的时间。

   如有必要，可使用 nvtop 工具监控计算资源利用率。

2.2.2.3 在集群中运行 Llama2-7B 的推理任务。

任务在集群的两个节点上运行。推理使用 Llama2-7B 模型，根据固定问题生成一定数量的 Tokens。

任务参数：

- 固定问题：**Can there be life somewhere in the Solar System outside Earth?**
- 问题重复次数：30 次
- 每个答案的 Token 数量：100 到 400 个（可变值）
- 生成的总 Token 数量：7750

执行步骤：

1. 在两个节点上启动多次推理命令：

```
./orch.py -t 推理（秒） -n rig0.IRONBYTE:5x2 rig1.IRONBYTE:5x2 --model_dir
/opt/dekubelocal/models/LLAMA2/metaAi/Llama-2-7b-chat-hf --max_Tokens_from
100 --max_Tokens_to 401 --question 'Can there be life somewhere in the Solar
System outside Earth?'
```

2. *可选：在单节点上运行推理任务（用于与单线程开源解决方案的性能对比）：

```
./orch.py -t 推理（秒） -n rig0.IRONBYTE:1x8 --model_dir
/opt/dekubelocal/models/LLAMA2/metaAi/Llama-2-7b-chat-hf --max_Tokens_from
100 --max_Tokens_to 401 --question 'Can there be life somewhere in the Solar
System outside Earth?'
```

任务结果：

1. 每个问题的答案生成时间。
2. 根据答案长度记录生成时间。
3. 平均每个 Token 的生成时间。

如有必要，可使用 nvtop 工具监控计算资源利用率。

3. 测试协议

3.1. 结论

在测试过程中，我们分析了使用开源方法和 IRONBYTE 方法对 Llama2-70B 和 Llama2-7B 模型执行 Pretraining（预训练）、Finetuning（微调）和推理（秒）（推理）任务的结果。对于每种任务类型，我们确定了执行时间并计算了效率。

根据表格中提供的数据，我们计算了 IRONBYTE 与开源方案在各类任务中的速度对比。

1. **Llama2-70B**

预训练 (1000 步)

单节点：

- IRONBYTE 方案的执行时间为 29 分钟，比开源方案的 93 分钟效率提高了 321%。

集群（2 个节点）：

- 在 IB DG 模式（80G）下，IRONBYTE 方案的执行时间为 20 分钟，比开源方案的 1,210 分钟效率提高了 6,050%。
- 在 IB Conn 模式（其他云端结果）下，IRONBYTE 方案的效率是开源方案（58 分钟）的 290%。

微调（500 步）

单节点：

- IRONBYTE 方案的执行时间为 68 分钟，比开源方案的 124 分钟效率提高了 182%。

集群（2 个节点）：

- 在 IB DG 模式（80G）下，IRONBYTE 方案的执行时间为 39 分钟，比开源方案的 537 分钟效率提高了 1,377%。
- 在 IB Conn 模式（其他云端结果）下，IRONBYTE 方案的效率是开源方案（74 分钟）的 190%。

推理任务（生成 1 个 Token 所需时间）

单节点：

- 在多推理模式下，IRONBYTE 方案的执行时间为 0.0139 秒，比开源方案的 0.057 秒效率提高了 410%。
- 在单线程模式下，IRONBYTE 方案的执行时间为 0.054 秒，比开源方案的 0.057 秒效率提高了 106%。

2. Llama2-7B:

预训练（2000 步）

单节点：

- IRONBYTE 方案的执行时间为 7 分钟，比开源方案的 120 分钟效率提高了 1,714%。

集群（2 个节点）：

- 在 IB DG 模式（80G）下，IRONBYTE 方案的执行时间为 5 分钟，比开源方案的 492 分钟效率提高了 9,840%。
- 在 IB Conn 模式（其他云端结果）下，IRONBYTE 方案的效率是开源方案（72 分钟）的 1,440%。

微调 （1 周期）

单节点：

- IRONBYTE 方案的执行时间为 28 分钟，比开源方案的 325 分钟效率提高了 1,161%。

集群（2 个节点）：

- 在 IB DG 模式（80G）下，IRONBYTE 方案的执行时间为 15 分钟，比开源方案的 3,750 分钟效率提高了 25,000%。
- 在 IB Conn 模式（其他云端结果）下，IRONBYTE 方案的效率是开源方案（191 分钟）的 1,273%。

推理任务（生成 1 个 Token 所需时间）

单节点:

- 在多推理模式下，IRONBYTE 方案的执行时间为 0.0019 秒，比开源方案的 0.02 秒效率提高了 1,053%。
- 在单线程模式下，IRONBYTE 方案的执行时间为 0.0175 秒，比开源方案的 0.02 秒效率提高了 114%。

上述数据表明，IRONBYTE 方案能够高效利用 H100 GPU 的性能，显著减少任务执行时间，并展示了确保高性能的优化效果。

因此，使用 IRONBYTE 方案显著提升了各类任务的性能，使其成为处理 LLM 任务的更优选择。

3.2. 附录

表 1. 量化 Llama2-70B 模型的测试结果：预训练和微调

任务类型	参数	开源	IRONBYTE	开源	开源	IRONBYTE
		1 节点	1 节点	2 节点 (IB DG 模式, 80G)	2 节点 (IB Conn 模式)	2 节点
预训练（分钟）					其他云端结果	
	1000 步	93	**29**	1210*	58	**20**
	100 步	10	**7**	110	6	X
预训练的效率系数						
	1000 步	3.2	**1**	60.5*	2.9	**1**
	100 步	1.4	**1**	-	-	-
微调（分钟）						
	500 步	124	68	537*	74	39
	100 步	26	10	133	15	X
微调的效率系数						
	500 步	1.8	**1**	13.8*	1.9	**1**
	100 步	2.6	**1**	-	-	-

表 2. 量化 **Llama2-70B** 模型 l 推理测试结果

任务类型	参数	开源	IRONBYTE		开源	IRONBYTE	
推理（秒）		1 节点	1 节点	**多推理（秒）**	1, 2 节点	1 节点	2 节点
	总生成时间（秒）	439	**419**	总生成时间（秒）	无解决方案	107	54
	生成 1 个 Token 所需时间（秒）	0.057	**0.054**	生成 1 个 Token 所需时间（秒）	无解决方案	0.0139	0,0079
	每秒生成的 Token 数	17.6	**18.4**	每秒生成的 Token 数	无解决方案	71	131
微调的效率系数		**效率通过比较推理值和多推理值计算得出**					
	总生成时间（秒）	4.1	**3.9**		无解决方案	**1**	**-**
	生成 1 个 Token 所需时间（秒）	4.1	**3.9**		无解决方案	**1**	**-**
	每秒生成的 Token 数	0.25	**0.26**		无解决方案	**1**	**-**

表 3. Llama2-7B 模型（无量化）预训练和微调任务的测试结果

任务类型	参数	开源	IRONBYTE	开源	开源	IRONBYTE
		1 节点	1 节点	2 节点 (IB DG 模式, 80G)	2 节点 (IB Conn 模式)	2 节点
预训练（分钟）					其他云端结果	
	2000 步	120	**7**	492*	72	**5**
	1000 步	62	**4**	235*	36	X
	100 步	6	X	22	4	X
预训练的效率系数						
	2000 步	17.14	**1**	98.4*	14.4	**1**
	1000 步	15.5	**1**	-	-	-
	100 步	-	**-**	-	-	-
微调（分钟）		开源	IRONBYTE	开源	开源	IRONBYTE
	1 周期	325	28	3750*	191	**15**
微调的效率系数						
	1 周期	11.6	**1**	250	12.7	**1**

表 4. Llama2-7B 模型（无量化）推理任务的测试结果

任务类型	参数	开源	IRONBYTE		开源	IRONBYTE	
推理（秒）		1 节点	1 节点	**多推理（秒）**	1, 2 节点	1 节点	2 节点
	总生成时间（秒）	155	**136**	总生成时间（秒）	无解决方案	15	7
	生成 1 个 Token 所需时间（秒）	0.02	**0.0175**	生成 1 个 Token 所需时间（秒）	无解决方案	0.0019	0.0009
	每秒生成的 Token 数	49	**56**	每秒生成的 Token 数	无解决方案	515	1075
微调的效率系数		**效率通过比较推理值和多推理值计算得出**					
	总生成时间（秒）	10.33	**9**		无解决方案	**1**	**-**
	生成 1 个 Token 所需时间（秒）	10.5	**9.21**		无解决方案	**1**	**-**
	每秒生成的 Token 数	0.095	**0.1**		无解决方案	**1**	**-**

***** – 标注的值是通过计算得出的，因为无法再现如此长时间的训练过程。

X – 未提供这些值，因为对于少量步数，不推荐使用 IRONBYTE 算法。原因在于结果传输和合并的时间超出了训练过程本身的耗时，而传输与结果合并时间不依赖于指定的步数（或周期）数量。

« - » – 效率比计算因缺乏结果而未进行，请参见上方备注。

«无解决方案» ——多推理的概念并不新颖，GitHub 上已有许多项目宣布实现输出任务的扩展，并处于不同的开发阶段。然而，这些项目无法作为参考解决方案，因为它们主要通过缓存查询和在请求处理过程中平衡上下文来实现更快的推理。在这些解决方案中，隔离“纯粹”的时间将耗费大量不合理的时间。

因此，我们建议使用 HF 参考代码的速度特性作为参考值。IRONBYTE 解决方案则代表了一种更底层的平台，能够在其基础上构建可扩展和可管理的解决方案，以满足客户需求。