样例

1. 以下是每个模块编时需要覆盖的内容，并不一定要分这么细，相近分不开的点就合并成一个来写。Title请保留方便我们后期合并文档。注意多用量化数据。
2. 概述

大模型的推理响应时间对用户体验至关重要，在大模型的部署过程中，模型并行化是一种非常有效且必要的方法，它可以有效降低推理延迟，提高性能；然而，多卡通信作为模型并行化的关键组成部分，往往会引入额外的开销和延迟，成为整体性能的瓶颈。尤其是在计算资源相对充足的情况下，这一现象变得更加明显。

1. 产品定位

目前在行业中，主流的推理引擎包括vllm、tgi、tensort-llm和lmdeploy等。其中，vllm和tgi基于高效的显存管理，提供了较高的吞吐量作为基准。然而，它们都是基于torch开发的，存在一定的额外开销，如上下文切换和资源分配管理等。当应用模型并行技术时，这些额外开销会进一步放大。另一方面，tensort-llm等引擎则采用纯C语言构建，几乎消除了可能存在的额外开销。然而，作为服务而言，它们缺乏高效的显存管理和服务调度策略，这导致吞吐量受到影响，并且额外的维护也变得困难。

当业务场景对响应速度要求很高，或有多余的计算资源时，实施模型并行是一个低成本且高效的方案；但如上文所述，基于torch的实现往往引入了更多的overhead，尤其在实施模型并行时，本产品旨在保留引擎原有优秀特性的基础上，减少多卡通信的额外开销，同时实施拓扑优化，帮助模型并行突破通信瓶颈，达到极致的推理响应速度。

1. 产品价值

促进营销、提高效率、节约成本、提高行业声望

1. 产品优势

该技术是一种无损优化技术，也就是说其可以“免费”集成到推理引擎中，而不需要任何代价；其推理结果保持与优化前完全一致，推理速度增益随着算力的增长得到显著提升。

例如在A100上，原始TGI的llama-7b双卡模型并行推理速度甚至低于单卡，这是因为其严重受限于通信瓶颈；经过该技术优化后，其双卡模型并行单条推理速度对比原始双卡模型并行增益在40%~50%，非常显著；

该技术方案在当前基于torch动态图研发的推理引擎中，属于sota领先水平；保留原有显存管理与高效调度的同时，填补了通信的空缺。

1. 功能介绍

-

1. 实施流程

-

1. 系统技术特性

评估指标采用tps（tokens per second），详细评测见：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模型 | GPU型号/数量[...] | 优化前(tps) | 优化后(tps) | 增益 |
| Llama-7B | A100/2 | 80 | 120 | +50% |
| Llama-13B | A100/2/4 | 67.4/66 | 80/98.7 | +18.7%/49.5% |
| Llama-7B | 3090/2 | 67.5 | 88.2 | +30.6% |
| Llama-7B | A10/4/8 | 77.8/75.6 | 90.4/108.6 | +16.1%/+43.6% |
| Llama-13B | A10/8 | 59.7 | 75 | +25.6% |
| Llama-7B | 4090/2 | 66.8 | 90.5 | +35.5% |

1. 技术架构介绍

**模型适配**

**（2天+）**

**优化算子构建**

**环境配置**

**（首次需要半天）**

**镜像打包部署**

**（半天）**

**性能压测**

**（半天）**

**引擎适配**

**（半天）**

各环节交付物：

* 1. 环境配置：

预构建优化算子所依赖的环境及配置。

交付物：暂无。

* 1. 优化算子构建：

编译并打包通信优化算子，暴露至python api。

交付物：.so二进制文件。

* 1. 模型适配：

根据不同模型，修改相应的模型层为优化版本。

交付物：模型及层文件源码。

* 1. 引擎适配：

集成通信优化能力到大模型推理引擎中去。

交付物：引擎源码。

* 1. 镜像打包部署：

打包最终部署引擎至镜像，构建一键启动式部署方案。

交付物：引擎镜像

* 1. 性能压测：

评估最终部署推理方案的性能指标，包括tps、吞吐等。

交付物：压测报告

运行环境：

v1.1.1需求nvidia驱动450.80.02+；

v1.3.4需求nvidia驱动525.60.13+（推荐525+）；

Docker版本需求19.03.15+；

GPU：A10/RTX 4090（24G显存）\* 8

CPU：Intel 8358P\*1(单CPU 32核心64线程，主频2.6GHz,睿频3.4GHz)

内存：384GB内存

系统盘：80G SSD （可以根据需求定制）

数据盘：200GB SSD（可以根据需求定制）

网络：2\*25G

1. 网络拓扑和典型配置

无额外设备需求

**快速部署及服务验证：**

* 1. 数据依赖：提供两个版本分别对应TGI中的v1.1.1和v1.3.4（main分支）；其中v1.1.1需求cuda11.8，v1.3.4仅更新了更多功能适配，需求cuda12.1，所对应的nvidia driver版本需要在525.60.13以上（实测阿里云470的驱动也能跑，但是对性能有影响）；数据存放在iwencai共享盘：[\\10.10.15.250\iwencai\zhaohuayang\tgi\_optim\tgi-optim-nccl-1.1.1.zip](\\\\10.10.15.250\\iwencai\\zhaohuayang\\tgi_optim\\tgi-optim-nccl-1.1.1.zip)
  2. 镜像构建：解压后进入目录执行以下命令

docker build -t tgi-optim-nccl:v1.1.1 -f Dockerfile\_nccl .

其中tgi-optim-nccl:v1.1.1为可以指定的镜像名；

* 1. 启动服务：默认启动优化版本

docker run --gpus '"device=0,1"' --shm-size 4g --net=host -v $MODEL\_DIR:/model tgi-optim-nccl:v1.1.1 --model-id /model --sharded false --port 7777

其中$MODEL\_DIR为模型目录，--sharded false为必须参数

* 1. 验证服务：（可在服务端查看耗时日志）

curl localhost:7777/generate -X POST -d '{"inputs":"who are you?","parameters":{"max\_new\_tokens":100,"details":false}}' -H 'Content-Type: application/json'

* 1. 回退为官方版本：

docker run --gpus '"device=0,1"' --shm-size 4g --net=host --entrypoint /usr/local/bin/text-generation-launcher-ori -e USE\_CUSTOM\_NCCL=0 -v $MODEL\_DIR:/model tgi-optim-nccl:v1.1.1 --model-id /model --port 7777