大模型计算-HW3

November 12, 2025

1 作业背景

本次作业基于 **LLAISYS** (Let's Learn AI Inference SYStem) 推理系统框架 [1]。LLAISYS 是一个教育项目，旨在为未来的 AI 工程师提供一个从零开始构建和学习 AI 系统基础模块的平台。

该框架采用 C++ 作为系统后端的主要实现语言（编译为 C API 的共享库），并通过 Python 前端进行便捷的调用和测试。本次作业要求学生在该框架基础上，选择 Triton、九齿或 CUDA 中的任一技术栈作为后端，实现一个完整的、基于 GPU 的高性能推理系统。

2 作业描述

2.1 作业目标

通过本作业，你将：

- 通过 LLAISYS 熟悉一个推理系统框架的基本结构。
- 掌握将自定义 GPU 算子（kernel）集成到推理系统后端的方法。
- 掌握使用 CUDA, Triton 或九齿编写、调试和优化高性能 GPU 算子。
- 学习使用 nsys, ncu 等工具进行性能分析和调优。

2.2 作业要求

代码须独立完成，允许讨论，但严禁抄袭，需要签署 honor code（详见附件）。

提交文件夹命名格式为学号 +HW3+ 姓名，例如：2025310000_HW3_ 张三。

评测将在 RTX4090 GPU 上进行，以课程提供的推荐容器环境为准。请确保代码在该环境中可正常运行。

请注意提交截止日期，具体时间以网络学堂公布为准。

禁止直接使用现成的算子库作为结果，可选语言限于 Cuda、Triton 或九齿。三种语言的工作量和最终性能各异，请根据自身情况谨慎选择。

2.3 作业任务：基于 LLAISYS 实现大模型推理系统

第一部分：GPU Runtime API 与核心算子实现

你需要参考 LLAISYS 文档中“项目 #2：在 LLAISYS 中集成 CUDA”的指导，但使用你选择的语言（CUDA/Triton/九齿）来实现。

1. **实现 GPU Runtime API(可选 Triton 和九齿)**：你需要实现 ‘src/device/runtime_api.hpp‘ 中定义的 ‘LlaisysRuntimeAPI‘ 接口。参考 ‘src/device/cpu/cpu_runtime_api.cpp‘，你需要创建对应的 GPU 实现（例如 ‘src/device/nvidia/nvidia_runtime_api.cpp‘）。这包括内存分配（‘malloc‘）、释放（‘free‘）、拷贝（‘memcpy‘）等基本功能。

2. **实现 GPU 算子 (Kernels)**：你需要为 LLAISYS 作业 #2 中定义的以下算子编写 GPU kernel 实现，并将其集成到 LLAISYS 框架中。

   - `argmax`
   - `embedding`
   - `linear` (必须自行实现矩阵乘法，不得调用 cuBLAS 等)
   - `rms_norm`
   - `rope` (旋转位置编码)
   - `self_attention` (包括 scaled-dot-product-attention 和 KV-Cache 逻辑)
   - `swiglu`

   在 CUDA 版本的实现中，对于每个算子，你应在 ‘src/ops/‘ 下对应的算子目录中创建 ‘nvidia/‘ 子目录，并实现 GPU 版本的算子逻辑。

   而对于九齿和 Triton 实现，以九齿为例，对于每个算子，你应在 ‘/python/llaisys/libllaisys/ninetoothed/‘目录下实现对应的算子，包括 ‘setup_kernel.py‘文件中的引用以及 ‘kernels‘子目录下对应的算子实现。文档中给出了未完全实现 add 版本，对于 triton 版本的实现，结构类似。注意对于九齿，需要在运行时**设置环境变量 ‘ENABLE_NT=True‘**，算子的 python 接口位置为 ‘/python/llaisys/ops.py‘，可以依照你的喜好进行修改。

第二部分：实现完整的大模型推理

在完成所有算子的 GPU 实现后，你需要将它们整合起来，实现一个完整的 LLM 推理系统。

1. **实现模型推理逻辑**：你需要修改 ‘python/llaisys/models/qwen2.py‘，使其能够调用你提供的 API，进而调用你实现的 GPU 算子。

2. **实现 KV-Cache**: 你必须在后端（C++/CUDA/Triton/九齿）实现 KV-Cache 功能，以确保推理的效率。

3. **通过测试**: 你的最终目标是使 ‘test/test_infer.py‘ 能够使用你的 GPU 后端成功运行，并生成与 PyTorch CPU 版本完全相同的文本结果。

2.3.1 正确性检验

1. **算子测试**: 在实现每个算子后，你都应通过运行对应的单元测试来验证其正确性。

```
python test/ops/add.py --device nvidia
python test/ops/argmax.py --device nvidia
# ...
```

2. **完整推理测试**: 在完成所有任务后，你必须通过完整的模型推理测试。

```
python test/test_infer.py --model [dir_path/to/model] --test --device nvidia
```

你需要确保使用 ‘–device nvidia‘ 标志运行，并且测试通过（在一些优化中即使数学等价也可能导致输出结果的不同，保证输出的语句有意义即可，但需要特殊说明）。

2.4 作业评分

2.4.1 基本算子实现 40%（每一个算子 test 通过即可获得 5%）

本部分共包含 8 个基本算子（具体列表见代码目录），**每一个算子 test 通过即可获得 5%**。

2.4.2 框架推理 50%（正确性分 20% 以及性能分 30%）

使用上述三种任一语言实现算子满足基本正确性要求以后，即可获得对应模块 40% 的性能分（即总分 10%）。剩余 60% 的性能分（即总分 20%）将对所有提交同学的性能进行排名以后，结合评估代码质量和可读性给出。注意：三种语言实现将**统一排名**。

重要提示：由于涉及到自动调优，要求程序单测例的**自动调优时间不得超过 15min**，如果超过时限，本测例不得分。

2.4.3 性能分析报告 10%

1. 报告需要详细描述每个算子采取的实现方式和优化手段，以及在代码中对应的部分。并且能够通过使用性能工具或者性能模型解释目前取得的性能结果。2. 如果有对框架本身进行优化，也

可以在报告中标明对应的优化内容以及在代码中对应的部分，会有酌情加分。3. 进一步讨论分析可能出现的复杂情形，对于一些没有取得性能提升的优化行为，如果能够解释其原理并进行分析，可以酌情加分。

2.4.4 额外功能分 20%（额外加分 100% 封顶）

完成以下列表中的**任**一额外功能，并撰写相应的**技术报告**与**实现代码**，即可获得此部分的额外得分。这些功能均对应于作业文档 `README` 中的特定项目，旨在鼓励大家进一步探索 LLAISYS 框架的高级应用。

推荐的额外功能实现方向

- **项目 #3 / #4：构建高性能 AI 聊天机器人服务**
  - **功能描述：** 构建一个能够处理多用户请求的高性能 LLM 推理服务。
  - **核心要求与技术要点：**
    - **基础 (项目 #3):** 实现支持随机采样（Temperature, Top-K, Top-P）的单用户实时聊天机器人，并搭建遵循 OpenAI Chat-Completion API 规范的服务器（最好支持流式输出）。
    - **进阶 (项目 #4):** 扩展服务以支持**多用户并发请求**，并实现**连续批处理（Continuous Batching）**机制，以最大化推理吞吐量，同时需要实现支持前缀匹配的 KV-Cache 池。
- **项目 #5：扩展张量并行分布式推理**
  - **功能描述：** 实现大型模型的分布式推理能力。
  - **核心要求与技术要点：** 在 LLAISYS 框架中引入**张量并行（Tensor Parallelism）**机制，将大型模型分片到多个设备上，并实现高效的分布式推理（根据硬件环境支持 NCCL 或 MPI）。

关于其他实现想法

如果你对额外功能有其他的想法，请**提前联系助教**，详细说明实现方案和预期效果。经助教确认和批准后，方可实现对应的功能并获得额外功能分。

3 参考资料

References

[1] InfiniTensor Developers. LLAISYS repo. `https://github.com/InfiniTensor/llaisys/tree/main`, 2025. GitHub Repository.