项目报告

2025 年夏季 InfiniTensor 大模型与人工智能系统训练营
Qijia Yang 中科院计算所

1. 概述

- 项目地址：github.com/mrams/llaisys
- 完成情况：
  - 完成 **CPU 推理优化**（项目 1），包括：SIMD 指令、OpenMP 多线程、分块(Cache Blocking)优化
  - 完成**构建 AI 聊天机器人**（项目 3），包括：实现 Gumbel-Max 采样、移植网页 UI、基于 WebSocket 实现实时流式输出
  - 完成**多用户推理服务**（项目 4），包括：支持多用户请求调度、连续批处理、支持前缀匹配的 KV-Cache 池
  - **额外**单独使用 LLAISYS 提供的算子，实现了 Python 版本的 Qwen2 完整推理
  - **额外**完成了 **Paged Attention/linear** 算子和 Paged KV Cache 管理，通过内存分页有效解决了不同请求长度导致的内存碎片化问题

2. CPU 推理优化

2.1. OpenMP

对几乎所有算子进行了 OpenMP 并行化

2.2. SIMD

首先尝试直接使用 x86 SSE2/SSE3 SIMD 指令集实现了 linear 算子的 SIMD 向量化加速(commit-7893)，然后使用 compiler vector extension 进行了改写，不依赖于特定指令集，便于跨平台(commit-128c)

2.3. Cache Blocking

利用 Cache 特点，对 linear 算子中矩阵进行分块运算，提高 Cache 命中率从而提高性能

2.4. Linear 算子性能优化分析

测试了大小为$(512, 4096) * (4096, 4096)$、类型为 `float32`、带 bias 的 linear 算子，结果如下：

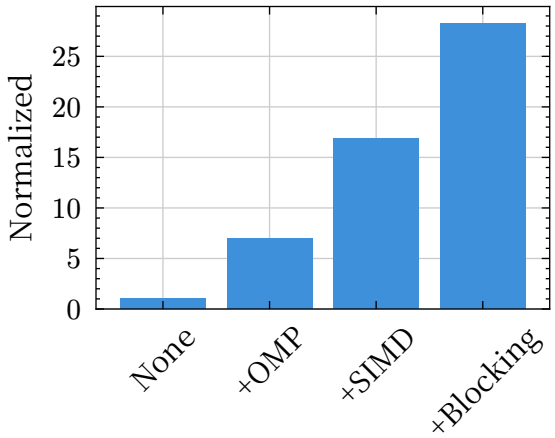

Figure 1: 各种优化策略下相对性能

优化策略	**耗时** (ms)
None	5936.54999
OMP	845.95023
OMP+SIMD	350.85551
OMP+SIMD+Blocking	210.22395

Table 1: 各种优化策略下 linear 算子耗时

3. 构建 AI 聊天机器人

3.1. Gumbel-Max 采样

实现了支持 Temperature 参数的 Gumbel-Max 随机采样算子

见代码 `src/sampler/gumbelmax`

3.2. 网页 UI

移植了开源项目 chatbot-ui (commit-bb38)

见代码 `webui/chatbot-ui`

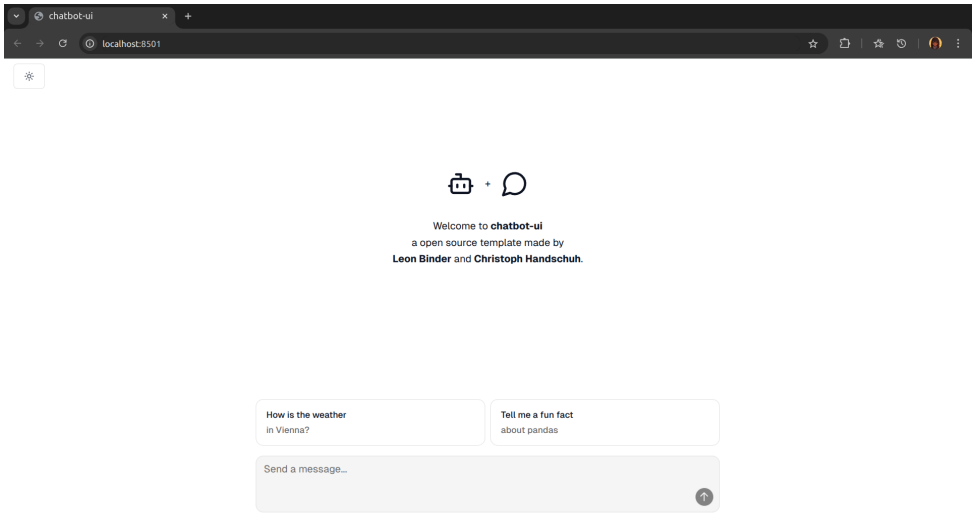

Figure 2: 网页 UI 欢迎界面

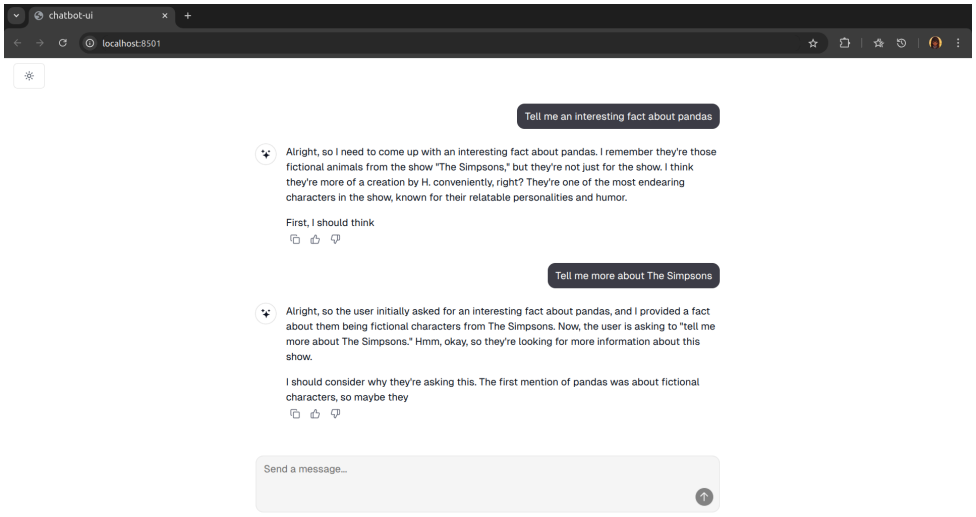

Figure 3: 连续对话

3.3. 实时流式输出

基于 WebSocket 实现了推理结果的实时流式输出(commit-bb38)

见代码 `webui/backend`

4. 多用户推理服务

4.1. 多用户请求调度

使用双端队列 `waiting` 和 `running` 维护多用户请求，考虑到 prefill 阶段是 compute-bound，而 decoding 阶段是 memory-bound，使用以下调度策略：

- 每次调度运行至多 `max_running_seqs` 个用户的请求
- 当运行队列数量小于 `max_running_seqs` 时，优先调度执行 prefill，当运行队列达到 `max_running_seqs` 时，批量执行 decoding
- 当 Paged KV Cache 无法为新请求分配空间时，先尝试释放已完成请求的空间，再尝试释放 `running` 队尾的请求，并将此请求放在 `wating` 队首，若仍无法分配空间，则将本请求放至 `wating` 队尾

见代码 `src/paged_cache.cpp/Scheduler::schedule()`

4.2. 连续批处理

调度器每次调度运行至多 `max_running_seqs` 个用户的请求

见代码 `src/paged_cache.cpp/Scheduler::schedule()`

4.3. KV-Cache 支持前缀匹配

支持前缀匹配的 KV-Cache 池，将 KV Cache 进行分块，使用 `hashxx` 库对每个 Block 及其前缀的 tokens 计算 hash 值，通过 `map` 维护每个 hash 对应 block_id 信息

见代码 `src/paged_cache.cpp/Sequence::allocate()`

5. Python 版本推理

单独使用 LLAISYS 提供的算子，实现了 Python 版本的 Qwen2 完整推理

见代码 `python/llaisys/models/qwen2_py.py`

6. Paged KV Cache

- 仿照 vLLM，将 KV Cache 进行分块，并且使用内存分页的方式，有效解决了不同请求长度导致的内存碎片化问题
- 修改支持了 Paged Attention/linear/rope 算子

见代码 `src/paged_cache.cpp, src/ops/self_attention_paged, ...`