进阶算子 Paged Attention 题面

题目简介

在大语言模型推理阶段，KV Cache 占用大量显存。为提升利用率，可将 KV 存储拆分为固定大小的块(block)，并通过块索引表实现非连续访存，这种方式称为 Paged Attention。你的任务是实现一个**简化版** Query PagedAttention **前向算子**，它能够:

- 根据块索引表从非连续块中收集 KV
- 对每个序列位置的 query 与其上下文 KV 做 attention
- 支持不同序列拥有不同上下文长度
- 支持块共享（多个序列可引用同一物理块）

无需实现反向传播或写入缓存逻辑, 实现逻辑参考函数 `paged_attention_ref`。

数据布局定义

- `query: [num_seqs, num_heads, head_size]`: 表示每个序列当前位置的 query 向量
- `key_cache: [num_blocks, num_heads, head_size, block_size]`
  `value_cache: [num_blocks, num_heads, head_size, block_size]`
  说明:
  - KV 按 block 存储
  - 每个 block 存储 block_size 个 token
  - 每个 token 的 KV 位于 block 内 offset 位置
- `block_tables: [num_seqs, max_num_blocks]`:
  含义: `block_tables[t, i]` 表示第 t 个序列的第 i 个逻辑块对应的物理编号
  注意:
  - 不使用 -1 padding
  - 所有 block id 均为合法整数
  - 一个序列的实际块数由 context_lens 决定
- `context_lens: [num_seqs]`: 表示第 t 个序列的上下文 token 数
  真实访问 token 总数: `context_lens[t]`
- `block_size`: 隐含于 cache tensor 最后一维: `block_size = key_cache.shape[-1]`

输出张量

`output: [num_seqs, num_heads, head_size]`

实现目标

你需要实现函数 `paged_attention(query, key_cache, value_cache, block_tables, context_lens) -> output`
从而得到一个一个按块索引读取 KV 的、逐序列计算的、支持变长上下文的 Attention Kernel.
要求:

- 功能正确
- 支持任意 context 长度
- 正确处理 block indexing
- 输出与 torch 参考实现数值一致 (允许浮点误差)

提交要求

本题已经在希冀平台创建提交窗口, 请登陆查看. 提交时你应该按照题目要求提交一个如下结构的 `.zip` 压缩包:

```
stu_submit.zip（压缩包可以随意命名）
  |- setup_env.sh
  |- Paged_attention.py（必须保证文件名相同，否则测评机无法找到）
```

输出格式

提交时, 请确保你的 `Paged_attention.py` 中所有的 `print` 函数**均为框架中提供的**, **不要增加额外的输出**, **也不要删除框架的输出**

评分标准

在允许的浮点误差范围内, 本题根据加速比 (acceleration ratio, 框架中为 `{avg_torch_ms/avg_triton_ms:.2f}`) 线性给分, 新的框架增加了多次预热计算并求平均值的功能, 提交时请设置计算次数为默认 (即设置参数 `num_warmup: int = 10, num_iters: int = 50, num_groups: int = 5`).
分数可参考下表:

acc_ratio	score
46.2	0
107.7	20
261.5	40
415.4	60
569.2	80
723.0	100