PR #43421 完整报告

vllm-project/vllm

[XPU][Mamba] Triton-based selective scan forward op for XPU

合并时间：2026-06-02 18:50

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43421

执行摘要

- 一句话：为 XPU 添加 Triton 实现的 Mamba selective scan 前向操作
- 推荐动作：值得精读：对 Triton kernel 的开发者和硬件移植团队有参考价值，展示了如何将 CUDA 自定义算子移植到 Triton 并在新硬件上运行。设计决策关注点：选择 Triton 而非原生 SYCL 或 Level Zero，降低了开发成本但牺牲了部分性能；并行化策略的取舍（访存 vs 计算）是典型 trade-off，读者可对比仓库中其他 Triton kernel（如 fused_moe）的维度安排。后续跟进：建议作者或社区优先优化访存模式（如交换 dim/seqlen 的并行维度），并补充 Triton kernel 的单元测试。

功能与动机

PR body 明确指出："Adds a Triton implementation of the Mamba selective scan forward pass (selective_scan_fwd) to enable Mamba1 prefill on Intel XPU devices." 目前 vLLM 中 Mamba 的 selective scan 依赖于 CUDA 算子，无法在 XPU 上运行，因此需要 Triton 移植以扩展硬件支持。

实现拆解

1. 新增 Triton kernel（vllm/_xpu_ops.py）：导入 vllm.triton_utils，定义 JIT 辅助函数 _softplus（平滑 ReLU 近似）和核心 kernel _selective_scan_fwd_kernel，该 kernel 在 (batch, dim) 网格上并行，处理变长 / 定长序列、缓存索引、SSM 状态更新等分支。kernel 主体通过 for 循环扫描序列维度，计算 delta、A、B、C 的离散化，并累加隐藏状态。
2. 封装类方法（vllm/_xpu_ops.py）：在 xpu_ops 类中添加 selective_scan_fwd 静态方法，负责参数校验、tensor 布局重塑（确保 (batch, dim, seqlen) 顺序）以及启动 Triton kernel。该方法匹配原有 CUDA ops.selective_scan_fwd 的接口签名。
3. 调度分支（vllm/model_executor/layers/mamba/ops/mamba_ssm.py）：在 selective_scan_fn 函数中，通过 current_platform.is_xpu() 判断当前平台，若为 XPU 则调用 xpu_ops.selective_scan_fwd，否则保持原先的 CUDA 调用。该修改最小化对通用逻辑的侵入。
4. 验证测试：无专门单元测试，但通过 tiiuae/falcon-mamba-7b 模型在 GSM8K 任务上评估，exact_match 达到 0.52 左右，验证了功能正确性。

关键文件：

- vllm/_xpu_ops.py（模块 算子层；类别 source；类型 core-logic；符号 _softplus, _selective_scan_fwd_kernel, selective_scan_fwd）：核心变更文件，新增 Triton JIT 实现的 Mamba selective scan forward kernel（_selective_scan_fwd_kernel）和封装类方法 selective_scan_fwd，是功能实现的主体。
- vllm/model_executor/layers/mamba/ops/mamba_ssm.py（模块 Mamba 模型层；类别 source；类型 infrastructure）：部署 / 基础设施文件，添加 XPU 平台判断分支，将原有 ops.selective_scan_fwd 调用分发到 xpu_ops.selective_scan_fwd，是最小化侵入的调度逻辑。

关键符号：_softplus, _selective_scan_fwd_kernel, xpu_ops.selective_scan_fwd, selective_scan_fn (modified dispatch)

关键源码片段

vllm/_xpu_ops.py

核心变更文件，新增 Triton JIT 实现的 Mamba selective scan forward kernel（_selective_scan_fwd_kernel）和封装类方法 selective_scan_fwd，是功能实现的主体。

```
# xpu_ops.py 中新增的 Triton JIT 函数和 kernel 定义 @triton.jit def_softplus(x):  # 数值稳定的 softplus 近似，小于 20 时使用 log1p(exp(x))  return tl.where(x <= 20.0, tl.math.log(tl.math.exp(x) + 1.0), x) @triton.jit def_selective_scan_fwd_kernel(  u_ptr, delta_ptr, A_ptr, B_ptr, C_ptr, D_ptr, z_ptr, delta_bias_ptr,  out_ptr, out_z_ptr, ssm_states_ptr,  query_start_loc_ptr, cache_indices_ptr, has_initial_state_ptr, block_idx_first_ptr, block_idx_last_ptr, initial_state_idx_ptr,  cu_chunk_seqlen_ptr, last_chunk_indices_ptr,  batch, dim, seqlen, dstate, n_groups, dim_ngroups_ratio, u_batch_stride, u_d_stride, delta_batch_stride, delta_d_stride,  A_d_stride, A_dstate_stride, B_batch_stride, B_group_stride, B_dstate_stride,  C_batch_stride, C_group_stride, C_dstate_stride,  z_batch_stride, z_d_stride, out_batch_stride, out_d_stride,  out_z_batch_stride, out_z_d_stride, ssm_batch_stride, ssm_dim_stride, ssm_dstate_stride,  cache_indices_stride,  null_block_id, block_size, delta_softplus: tl.constexpr,  HAS_D: tl.constexpr, HAS_Z: tl.constexpr, HAS_DELTA_BIAS: tl.constexpr,  IS_VARLEN: tl.constexpr, HAS_CACHE_INDICES: tl.constexpr, CACHE_ENABLED: tl.constexpr,  BLOCK_DSTATE: tl.constexpr, ):  # 当前 kernel 在 (batch, dim) 网格上并行，每个程序处理一个 dim 切片  batch_idx = tl.program_id(0)  dim_idx = tl.program_id(1)  group_idx = dim_idx // dim_ngroups_ratio  if IS_VARLEN:  seq_start = tl.load(query_start_loc_ptr + batch_idx).to(tl.int32)  seq_end = tl.load(query_start_loc_ptr + batch_idx + 1).to(tl.int32)  actual_seqlen = seq_end - seq_start  else:  seq_start = 0  actual_seqlen = seqlen  # 处理缓存状态索引（用于 offloading 或 prefix caching) if CACHE_ENABLED:  # ... 加载缓存索引并判断是否有效  pass  elif HAS_CACHE_INDICES:  # ... 处理普通缓存  pass  # 主循环：按 block_size 步进扫描序列  for i in range(0, actual_seqlen, block_size):  # 加载当前块的 u, delta, A, B, C 数据  # 计算离散化：delta = delta_softplus(delta + delta_bias) if enabled  # 更新 SSM 状态：h = A_bar * h + B_bar * u  # 计算输出：y = C * h
```

```
+ D * u          # 若 HAS_Z: 使用 sigmoid 门控          # 写入 out 和 ssm_states          pass
classxpu_ops:      @staticmethod      defselective_scan_fwd(          u, delta, A, B, C, D,
z, delta_bias,          delta_softplus, query_start_loc, cache_indices, has_initial_state,
      ssm_states, null_block_id, block_size,          block_idx_first_scheduled_token,
block_idx_last_scheduled_token,          initial_state_idx, cu_chunk_seqlen,
last_chunk_indices      ):          # 确保输入张量按 (batch, dim, seqlen) 布局且连续
assert u.is_contiguous()          batch, dim, seqlen = u.shape          dstate = A.shape[-1]
        n_groups = B.shape[1] if B.dim() == 4 else 1          # 计算 dim_ngroups_ratio
  # 启动 Triton kernel          _selective_scan_fwd_kernel[(batch, dim)](              # 传递所
有 指针和标量              # ...          )          # out 与 delta 共享存储（in-place 更新）
#  返回 delta（即 out）  * 注：实际 kernel 体较长，此处省略循环内细节。关键并行化缺陷：
 dim 维度的线程访问 seqlen 维度的连续数据，导致非合并访存。*
```

评论区精华

Review 评论主要聚焦于 kernel 性能与数值稳定性：

- 内存访问模式问题（高优先级）：*gemini-code-assist[bot]* 指出 kernel 在 dim 维度并行化，而输入张量 (batch, dim, seqlen) 中 seqlen 是连续维度，导致子组中各工作项访问不连续地址，产生大量非合并访存，显著降低 GPU 吞吐。建议改为在 seqlen 维度并行化（或调整循环结构）以提升带宽利用率。该问题未被作者直接回复，但 PR 获得了 approval 后合并，表明团队可能将性能优化留给后续迭代。
- 手动 sigmoid 数值稳定性（高优先级）：在 kernel 中 z 门控部分使用了 `1.0 / (1.0 + tl.exp(-z_val))`，评论建议替换为 `tl.sigmoid(z_val)`，因为它已在仓库其他 Mamba 代码中使用，数值更稳定且更可读。最终代码未采纳该建议（仍保留手动实现），可能因作者认为 XPU 上 `tl.sigmoid` 行为不可预测或出于一致性考量，但无明显回复。
- Kernel memory access pattern inefficiency (performance): 作者未直接回复，PR 获得 approval 后合并，表明该性能问题被接受为已知限制，留待后续优化。
- Manual sigmoid numerical stability (correctness): 建议未被采纳，PR 合并时仍保留手动实现，可能因作者对 XPU 上 tl.sigmoid 的兼容性有顾虑，或认为误差在可接受范围内。

风险与影响

- 风险：
  1. 性能风险：当前 kernel 的并行化策略导致非合并访存，在长序列场景下性能可能不如 CUDA 原生实现，可能影响 XPU 上 Mamba 模型的实际吞吐；后续若无人优化，可能成为 XPU 推理的瓶颈。
  2. 数值精度风险：手动 sigmoid 在极端值处可能产生微小误差，但应用层（GSM8K）的评估未显示明显退化，风险较低。
  3. 维护风险：_xpu_ops.py 文件快速膨胀（+474 行），后续 GPU 通用逻辑的修改可能忘记同步 XPU 分支，导致行为不一致。
  4. 测试缺失风险：无单元测试覆盖 Triton kernel 的正确性，仅依赖端到端模型评测，未来重构时容易引入回归。 - 影响：对用户：Intel XPU 用户现在可以运行 Mamba 架构的模

型（如 falcon-mamba-7b），填补了之前仅支持 CUDA 的空白；但实际推理速度受限于 kernel 访存效率，预期比同等 NVIDIA 显卡慢。对系统：在 mamba_ssm.py 中引入平台分支，增加了代码耦合度；_xpu_ops.py 成为 XPU 专用 Triton kernel 的集合，未来类似移植可复用此模式。对团队：需要持续关注 XPU 上 Triton kernel 的性能问题，并保持与 CUDA 算子接口同步。

- 风险标记：平台分支性能隐患，缺失测试覆盖，内存访问模式缺陷

关联脉络

- PR #43930 [XPU][Bugfix] Fix per_token_group_fp8_quant missing dummy args on XPU: 同样修改了 vllm/_xpu_ops.py，且同属 XPU 平台适配工作，与本 PR 共同扩展了 XPU 上的算子支持。