PR #34759 完整报告

sgl-project/sglang

[DSpark] Fix EP1 decode performance regression

合并时间：2026-08-14 08:00

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34759

执行摘要

- 一句话：DSpark 按需分配 num_token_non_padded，修复 EP1 解码性能回归
- 推荐动作：值得快速浏览：改动极小但定位精准，是典型的避免热路径上无意义设备分配的微优化案例；同时展示了性能回归修复应附 matched profiling 的实践。合并历史中测试被移除这一点值得关注——如果团队希望长期守住两条开关路径的行为，建议后续补一个针对 _make_num_token_non_padded 的轻量单测。

功能与动机

根据 PR body，回归源于 PR #33465 使 DraftBlockProposer 在每次 speculative decoding 步骤无条件构造 CUDA int32 标量，而 num_token_non_padded 仅在 enable_num_token_non_padded() 为真时被消费；MoE EP size 1 场景下谓词为假，设备分配完全未被使用。ormandj 在 issue 评论中确认目标是修复这一 CUDA EP1 decode 性能回归，并附带 matched before/after/fixed profiling 结果。

实现拆解

1. 定位回归源：DraftBlockProposer._run_forward（位于 python/sglang/srt/speculative/dspark_components/dspark_draft.py）在构造 ForwardBatch 时，将 num_token_non_padded 参数改为无条件内联构造 torch.tensor(draft_num_tokens, dtype=torch.int32).to(device, non_blocking=True)。由于该字段只在 enable_num_token_non_padded() 为真时被消费，MoE EP size 为 1 时谓词为假，每次投机解码步骤都会产生未被使用的设备分配，造成平均每步 6%~9% 的耗时回退。
2. 新增按需分配辅助函数：在模块顶层新增 _make_num_token_non_padded(num_tokens, device) -> Optional[torch.Tensor]，先判定 enable_num_token_non_padded()，为假时直接返回 None 跳过分配；为真时保留原有 torch.int32 设备标量与 non_blocking=True 语义。
3. 替换调用点：_run_forward 中 ForwardBatch 的 num_token_non_padded 改为调用 _make_num_token_non_padded(draft_num_tokens, device)，num_token_non_padded_cpu 保持原有 draft_num_tokens 不变，避免影响依赖 CPU 侧数值的路径。
4. 测试与配套：PR body 声称补充了两条路径的单元覆盖，但最终提交（drop non-guarding num_token_non_padded test）移除了对应测试，合并结果仅包含源码改动

；重跑的 test_basic_sanity_dspark.py、test_dspark_dp_tier.py、test_dspark_draft_path_default.py 均通过。

关键文件：

- python/sglang/srt/speculative/dspark_components/dspark_draft.py（模块 投机解码；类别 source；类型 core-logic；符号 _make_num_token_non_padded）：唯一变更文件，也是 DSpark draft 前向与投机解码的核心路径。新增 _make_num_token_non_padded 按需分配 num_token_non_padded，并替换 _run_forward 中无条件设备分配，修复 EP1 性能回归。

关键符号：_make_num_token_non_padded

关键源码片段

python/sglang/srt/speculative/dspark_components/dspark_draft.py

唯一变更文件，也是 DSpark draft 前向与投机解码的核心路径。新增 _make_num_token_non_padded 按需分配 num_token_non_padded，并替换 _run_forward 中无条件设备分配，修复 EP1 性能回归。

```
# python/sglang/srt/speculative/dspark_components/dspark_draft.py

# 集中管理 num_token_non_padded 设备标量的分配。
# 该字段只在 enable_num_token_non_padded() 为真时被下游消费，
# 例如 MoE EP size 为 1 时开关为假，若仍执行 torch.tensor(...).to(device, ...)，
# 每个投机解码步骤都会产生一次无意义的设备分配，这正是 PR #33465 引入的
# CUDA EP1 decode 性能回归根因。开关关闭时返回 None 以跳过分配。
def _make_num_token_non_padded(
    num_tokens: int, device: str | torch.device
) -> Optional[torch.Tensor]:
    if not enable_num_token_non_padded():
        return None
    return torch.tensor(num_tokens, dtype=torch.int32).to(device, non_blocking=True)


# DraftBlockProposer._run_forward 中构造 ForwardBatch 的调用点：
# 原实现内联无条件分配，现改为按需创建；CPU 侧的 num_token_non_padded_cpu
# 保持不变，作为 shape 校验与 graph capture 的稳定回退值。
draft_num_tokens = bs * gamma
draft_forward_batch = ForwardBatch(
    forward_mode=ForwardMode.TARGET_VERIFY,
    batch_size=bs,
    input_ids=draft_block_ids.flatten(),
    req_pool_indices=batch.req_pool_indices,
    seq_lens=prefix_lens,
    out_cache_loc=draft_cache_loc,
    seq_lens_sum=draft_seq_lens_sum,
    seq_lens_cpu=draft_seq_lens_cpu,
    positions=draft_positions,
```

```
        input_embeds=draft_input_embeds,
        spec_algorithm=SpeculativeAlgorithm.DSPARK,
        spec_info=self._draft_block_spec_info,
        capture_hidden_mode=CaptureHiddenMode.NULL,
        num_token_non_padded=_make_num_token_non_padded(draft_num_tokens, device),
        num_token_non_padded_cpu=draft_num_tokens,
    )
```

评论区精华

本次 PR 没有独立的 review 评论，hnyls2002 直接 approve。核心讨论发生在 issue 评论：ormandj 说明修复目标与 matched profile（before/#33465 后 /fixed），hnyls2002 触发 /rerun-test，三个 DSpark 测试全部通过。值得注意的取舍是：PR body 声称的双路径单测最终被移除（提交 `drop non-guarding num_token_non_padded test`），合并版本仅含源码改动。

- 回归修复的动机与性能验证基线 (performance): 1-gpu-h100 与 ubuntu-latest 两个 runner 上三个测试全部通过，无回归。
- 测试覆盖的取舍 (testing): 未保留针对开关两条路径的直接断言，依赖现有 DSpark 测试覆盖主路径。

风险与影响

- 风险：行为一致性：返回 None 依赖下游只在 enable_num_token_non_padded() 为真时消费该字段；若某后端（如 NPU 上的 Kimi-K3）在开关为假时仍读取，会出现隐性 None 解引用。PR 声明开关与消费一一对应，但没有单元测试直接守护该契约。CUDA graph 兼容：decode CUDA graph 捕获时若依赖该张量存在，None 可能影响 replay；本 PR 保留 num_token_non_padded_cpu=draft_num_tokens 作为 CPU 侧回退，风险较低，且 profile 中 graph kernel 数与对照组一致，构成间接验证。性能收益范围：实测收益仅在 EP1（MoE EP size 1）+ CUDA 场景显著，其他配置下行为不变。
- 影响：对用户的收益：DeepSeek 系列 DSpark decode 在 EP1 配置下每步延迟降低约 7%~9%（Blackwell TP2 实测），等效提高输出吞吐。对系统的风险：改动位于投机解码热路径（dspark_draft.py 的 _run_forward），但分支判定极轻，不会引入额外开销。对团队的意义：撤销了 #33465 的副作用，同时保留 NPU padding 开关路径，是跨硬件（CUDA/NPU）特性共存的清理。
- 风险标记：投机解码热路径变更，双路径单测未保留，行为依赖启用开关

关联脉络

- PR #33465 [Kimi-K3][NPU] Support Kimi-K3 on NPU: 本 PR 的回归源头：该 PR 在 DraftBlockProposer 中无条件构造 num_token_non_padded 设备标量，导致 CUDA EP1 decode 每步额外分配。
- PR #34782 [Fix] Make the DSpark draft num_token_non_padded host-to-device copy non-blocking: 同一文件、同一符号的并行演进：将 host-to-device 拷贝改为非阻塞；本 PR 进一步实现按需分配，二者共同收敛 num_token_non_padded 的开销问题。