PR #34782 完整报告

sgl-project/sglang

[Fix] Make the DSpark draft num_token_non_padded host-to-device copy non-blocking

合并时间：2026-08-14 07:10

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34782

执行摘要

- 一句话：DSpark draft 的 num_token_non_padded 改为非阻塞拷贝，消除每步同步
- 推荐动作：值得精读。该 PR 虽然改动极小，但精准定位了性能瓶颈，并展示了 PyTorch 中 host-to-device 拷贝的常见同步陷阱。其与现有 global_num_tokens_gpu 写法的对齐也体现了代码库内的一致性原则。适合作为性能优化的小型范例。

功能与动机

PR body 中明确说明：DSpark draft 的 ForwardBatch 用 torch.tensor(..., device=device) 构建 num_token_non_padded，pageable 源使 PyTorch 在每次 decode 步骤用 cudaMemcpyAsync 后跟随 cudaStreamSynchronize。Profile 显示 21 次同步共 71.4 ms（每步约 3.5 ms，占 8.4 ms 步骤的 42%），纯粹阻塞 CPU 无法提前执行。改为 non_blocking=True 后去除该同步，并匹配下方 global_num_tokens_gpu 的既有构建方式。

实现拆解

1. 变更入口：修改 python/sglang/srt/speculative/dspark_components/dspark_draft.py 中的 _run_forward 方法，该方法负责构造 DSpark draft 的 ForwardBatch。
2. 核心改动：将 num_token_non_padded 的构造从 torch.tensor(draft_num_tokens, dtype=torch.int32, device=device) 改为 torch.tensor(draft_num_tokens, dtype=torch.int32).to(device, non_blocking=True)。前者会在 GPU 上直接创建 tensor 并隐含同步拷贝，后者先在 CPU 上创建标量 tensor，再以非阻塞方式异步传输到 GPU，避免每次 decode 步骤的流同步。
3. 与既有模式对齐：该写法与同函数下方 global_num_tokens_gpu 的构建方式一致，增强了代码一致性。
4. 测试与配套：本次变更未附带测试文件，仅靠 CI 覆盖；由于改动极小且结果语义相同（均为 int32 标量 tensor），回归风险较低。

关键文件：

- python/sglang/srt/speculative/dspark_components/dspark_draft.py（模块 推测解码；类别 source；类型 core-logic；符号 _run_forward）：唯一变更文件，核心改动位于 _run_forward 中 ForwardBatch 的 num_token_non_padded 构造方式，直接影响 DSpark draft 每次 decode 步骤的同步开销。

关键符号：_run_forward

关键源码片段

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

唯一变更文件，核心改动位于 _run_forward 中 ForwardBatch 的 num_token_non_padded 构造方式，直接影响 DSpark draft 每次 decode 步骤的同步开销。

```
# 构造 DSpark draft 的 ForwardBatch 时，将 num_token_non_padded 的
# 创建方式从 torch.tensor(..., device=device) 改为先 CPU 再 non_blocking 拷贝。
# 旧写法会在每次 decode 步骤触发 cudaStreamSynchronize，阻塞 CPU 提前执行；
# 新写法与下方 global_num_tokens_gpu 的构建方式保持一致，消除同步。
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,
    # 先在 CPU 上创建标量 tensor，再以 non_blocking 方式拷贝到 GPU，
    # 避免 torch.tensor(..., device=device) 在每次 decode 步骤产生同步阻塞。
    num_token_non_padded=torch.tensor(draft_num_tokens, dtype=torch.int32).to(
        device, non_blocking=True
    ),
    num_token_non_padded_cpu=draft_num_tokens,
)
```

评论区精华

本 PR 仅有作者一条 /tag-and-rerun-ci 的 CI 重跑指令评论，没有实质性的 Review 讨论或异议。变更本身简单明确，无争议点。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 非阻塞拷贝的有效性：non_blocking=True 对于 pageable 源（未固定内存）不一定保证异步，PyTorch 可能仍会同步。但本场景中 PR 的 profiling 显示同步被移除，且即使同步也不改变正确性，只是优化效果可能因硬件 / 驱动而异。
  2. 正确性风险：仅改变 tensor 创建方式，数值和 dtype 均不变，不影响下游逻辑，正确性风险极低。

3. 测试覆盖缺失：没有针对性单元测试，但该路径被现有 CI 覆盖，且改动极小，风险可接受。 - 影响：影响范围集中在 DSpark 推测解码的 *decode* 路径（dspark_draft.py 的 _run_forward）。通过消除每步的宿主同步，可减少 CPU 阻塞，提升 *decode* 阶段的吞吐和延迟表现。对用户而言是性能改善且无 API 变化；对团队而言，该修改提供了一个可复用的模式，即构造 GPU tensor 时优先使用 non_blocking 拷贝，避免隐式同步。整体影响为正，但限于 DSpark 场景。 - 风险标记：无测试覆盖，非阻塞拷贝语义依赖驱动

关联脉络

- PR #33857 [Perf] Skip trivial DSV4 nonpaged indexer logits: 同为 DSpark/DeepSeek 推测解码路径的性能优化，涉及 draft 模型前向逻辑，与本 PR 在功能线上相关。
- PR #34597 [AMD] Run V4 MTP target-verify through the decode kernel: 针对 DSpark V4 的 MTP target-verify 路径优化，同属 DSpark 推测解码的 kernel 与调度优化系列。