PR #33688 完整报告

sgl-project/sglang

[Diffusion]Skipping tensor copying for non-BCG GLM-Image workflows

合并时间：2026-08-07 11:19

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33688

执行摘要

- 一句话：跳过非 BCG GLM-Image 张量复制，解码提速 2.87x
- 推荐动作：值得精读。这是一个“框架特性（CUDA graph）与数值操作耦合”的典型小案例：用运行期状态 is_in_breakable_cuda_graph() 分叉 out-of-place 与 in-place 两条路径，避免热路径上的无谓张量复制。对维护多后端（CUDA/NPU/AMD）类似热路径的工程师有参考价值。值得关注的设计决策：BCG 路径保留 masked_fill 以维持 graph 捕获兼容性，而非 BCG 路径用布尔索引就地清零；后续可探索 BCG 路径下更廉价的清零方式。

功能与动机

PR body 直接引用 #31467：其描述了 GLM-Image 解码阶段性能退化。退化的根源在于 #27436 无条件引入的 masked_fill 复制——在非 BCG（默认关闭）路径上每次解码都复制整张 prior embedding 张量。本 PR 的目标是在非 BCG 工作流下跳过该复制，作者在 body 中给出实测：解码阶段 795.68 ms -> 277.36 ms，约 2.87x 加速，并附有准确性对比图。

实现拆解

实现按三步拆解：

1. 核心优化：按 BCG 状态分叉 prior embedding 的 drop 处理（python/sglang/multimodal_gen/runtime/models/dits/glm_image.py） - 新增 import：is_in_breakable_cuda_graph，来自 sglang.srt.model_executor.runner_backend_utils.breakable_cuda_graph。 - 将 prior_embedding.masked_fill(prior_token_drop.unsqueeze(-1), 0) 改为 if is_in_breakable_cuda_graph(): 分支：BCG 模式保持 masked_fill（out-of-place，维持 CUDA graph 捕获兼容性，与 PR #27436 行为一致）；非 BCG 模式改为 prior_embedding[prior_token_drop] *= 0.0（就地操作，避免整张 [B, S, D] embedding的分配与复制）。-影响：默认配置（--enable_breakable_cuda_graph=false）下 GLM-Image 解码阶段提速约 2.87x；BCG 用户行为不变。
2. 更新 NPU 性能基线（python/sglang/multimodal_gen/test/server/ascend/perf_baselines_npu.json） - 将 glm_image_t2i_1npu 的 DecodingStage 从 795.68 更新为 277.36，与优化后实测对齐，防止旧基线在 NPU 性能回归检测中误报。
3. 基准生成脚本支持 NPU 平台（python/sglang/multimodal_gen/test/scripts/gen_perf_baselines.py） - _all_cases() 增加平台分支：current_platform.is_npu() 时导入 sglang.multimodal_gen.test.server.ascend.testcase_configs_npu，否则导入通用

testcase_configs；确保 NPU 环境下生成基线时能覆盖 NPU 专属用例。 - 附带修复 lint。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/glm_image.py（模块 扩散模型；类别 source；类型 core-logic；符号 forward）：核心源码改动：引入 is_in_breakable_cuda_graph 分支，非 BCG 路径用就地乘法替代 masked_fill 复制，是 2.87x 解码加速的来源。
- python/sglang/multimodal_gen/test/scripts/gen_perf_baselines.py（模块 测试脚本；类别 test；类型 test-coverage；符号 _all_cases）：测试配套：为基准生成脚本增加 NPU 平台分支，使 NPU 环境能加载专属 testcase 配置。
- python/sglang/multimodal_gen/test/server/ascend/perf_baselines_npu.json（模块 性能基线；类别 test；类型 test-coverage）：性能基线数据更新：同步优化后的实测值，避免 NPU 性能回归误报。

关键符号：forward, _all_cases

关键源码片段

python/sglang/multimodal_gen/runtime/models/dits/glm_image.py

核心源码改动：引入 is_in_breakable_cuda_graph 分支，非 BCG 路径用就地乘法替代 masked_fill 复制，是 2.87x 解码加速的来源。

```
from sglang.srt.model_executor.runner_backend_utils.breakable_cuda_graph import (
    is_in_breakable_cuda_graph,
)

# 在 transformer blocks 之前准备好 prior token 的 embedding，
# 并按是否处于 breakable CUDA graph 捕获中分叉处理 drop 掩码。
prior_embedding = self.prior_token_embedding(prior_token_id)

if is_in_breakable_cuda_graph():
    # BCG 路径：保留 out-of-place 的 masked_fill。
    # CUDA graph 捕获对 kernel 形状与内存地址有静态要求，
    # masked_fill 的纯广播语义便于图形捕获与重放，
    # 因此维持 PR #27436 引入的复制行为。
    prior_embedding = prior_embedding.masked_fill(
        prior_token_drop.unsqueeze(-1), 0
    )
else:
    # 非 BCG 路径：用布尔索引 + 就地乘法等价清零。
    # 只读写被 drop 的少量位置，避免整张 [B, S, D] embedding 的
    # 分配与复制，这是解码阶段 795.68 ms -> 277.36 ms（约 2.87x）
    # 提速的主要来源。注意该写法要求 prior_token_drop 与
    # prior_embedding 前两维形状严格一致（masked_fill 可广播）。
    prior_embedding[prior_token_drop] *= 0.0

prior_hidden_states = self.prior_projector(prior_embedding)
```

python/sglang/multimodal_gen/test/scripts/gen_perf_baselines.py

测试配套：为基准生成脚本增加 NPU 平台分支，使 NPU 环境能加载专属 testcase 配置。

```
def _all_cases() -> list[DiffusionTestCase]:
    # 按当前硬件平台选择测试用例配置模块：
    # NPU 走 ascend 目录下的专用配置，其余平台走通用配置。
    if current_platform.is_npu():
        import sglang.multimodal_gen.test.server.ascend.testcase_configs_npu as cfg
    else:
        import sglang.multimodal_gen.test.server.testcase_configs as cfg

    cases: list[DiffusionTestCase] = []
    for _, v in inspect.getmembers(cfg):
        if isinstance(v, list) and v and isinstance(v[0], DiffusionTestCase):
            cases.extend(v)

    # 按用例 id 去重，保证不同配置模块间的交集用例只跑一次。
    seen: set[str] = set()
    out: list[DiffusionTestCase] = []
    for c in cases:
        if c.id not in seen:
            seen.add(c.id)
            out.append(c)
    return out
```

评论区精华

无代码级 review 评论；两位 reviewer 直接 APPROVED（Makcum888e 回复 "LGTM"，ping1jing2 通过）。Issue 评论中的讨论集中在 CI 状态：OrangeRedeng 两次触发 CI（/tag-and-rerun-ci、/rerun-failed-ci）；作者 e-martirosian 澄清 extra CI 中 multimodal-gen-test-2-gpu-amd 与 multimodal-gen-unit-test-amd 在 main 分支以相同错误失败（附 action 链接），并非本 PR 引入；随后 rerun 后 CI 通过。

- AMD CI 失败是否为 PR 引入 (question): 确认失败为 main 分支既有问题，与本 PR 无关；随后 /rerun-failed-ci 后 CI 通过。

风险与影响

- 风险：具体风险如下：
- 形状契约收紧（glm_image.py）：prior_embedding[prior_token_drop] *= 0.0 的布尔索引要求 prior_token_drop 与 prior_embedding 前两维严格一致；原 masked_fill 依赖广播可容忍 [B, 1] 等压缩形状。若未来调用方传入压缩形状的 drop mask，新路径会抛错或行为不同。
- 就地写语义依赖：prior_embedding 是 prior_token_embedding 每次调用返回的新张量，因此就地写安全；但若未来实现改为缓存复用 embedding 输出，将出现跨步数污染。
- 双路径维护成本：BCG 与非 BCG 存在两套语义等价但实现不同的代码，后续修改需同时验证两条路径；当前测试仅覆盖非 BCG 的 NPU 性能基线。

- 基线数据样本少：277.36 ms 为单次实测值，跨硬件 / 环境波动可能造成 NPU 性能回归检测误报。
- 缺乏 GPU 平台验证：准确性对比与速度数据均来自 NPU（Ascend）环境，GPU 上的收益未经报告。
- 影响：影响范围与程度：
- 用户：默认关闭 BCG 的 GLM-Image 工作流解码阶段延迟降低约 2.87x，端到端出图更快；BCG 用户无行为变化。
- 系统：改动位于 diffusion 模型 forward 热路径，不改变张量形状与数值结果（数学等价），并减少解码阶段的中间张量分配，对显存占用有正向影响。
- 团队：为 #31467 报告的退化提供了修复闭环；gen_perf_baselines.py 支持 NPU 平台后，后续 NPU 性能回归检测可自动覆盖 glm_image_t2i_1npu 用例。
- 风险标记：就地张量操作语义变更，布尔索引形状假设收紧，BCG/ 非 BCG 双路径分叉，NPU 基线仅单次实测，缺乏 GPU 平台验证

关联脉络

- PR #31467 （上下文未提供标题）：PR body 直接引用：其报告了 GLM-Image 解码阶段性能退化，本 PR 即针对该问题做优化。
- PR #27436 （上下文未提供标题）：本 PR 修改的就是 27436 引入的 prior_embedding masked_fill 复制逻辑：在非 BCG 路径跳过该复制以恢复性能。
- PR #33823 [Diffusion] FLUX.2 bit-exact residual-gate fast path (H200 klein-4B 50-step denoise -1.2%): 同属 diffusion 模块性能优化脉络，与 #33688 一道体现该模块在解码与去噪阶段消除冗余计算的方向。