PR #33127 完整报告

sgl-project/sglang

[Fix] Bound FULL_MASK verify-mask reuse by the captured max_bs

合并时间：2026-08-01 08:50

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33127

执行摘要

- 一句话：修复 FULL_MASK verify mask 超 max_bs 越界复用，防调度器挂死
- 推荐动作：值得精读：这是一个把“为什么 bs <= max_bs 对 FULL_MASK 也是充分边界”论证清楚的修复，frozen struct 防止 max_bs 与 buffer 失配的设计模式可借鉴；同时其 commit 历史展示了“动态计算 vs 分配期固化”两种方案的取舍，对理解预分配 buffer 的容量管理有帮助。

功能与动机

PR body 明确说明：VerifyMask.fits() 此前对 FULL_MASK 跳过了容量检查，导致一个 *eager batch past the captured* max_bs 复用了按 max_bs 大小的 buffer，并在 target-verify mask 提取期间越界——设备端 index out of bounds 断言和死掉的 scheduler。作者论证两种布局均按 request 大小以最坏情况 per-request 边界分配，因此 bs <= max_bs 是充分边界；该问题在 --cuda-graph-max-bs-decode 8 与 max_running_requests=48 组合下可复现（gsm8k score 0.0），且是 #32920（HybridAttnBackend 路由到子级预分配 mask）引入的回归。

实现拆解

1. 收紧 fits() 的容量判定（python/sglang/srt/layers/attention/verify_mask.py）：fits(bs, num_draft_tokens) 改为 fits(bs)，逻辑从“仅 QLEN_ONLY 检查 buffer.numel()，FULL_MASK 一律放行”简化为 bs <= self.max_bs。依据是 tree_mask_numel 为 bs * per_req，且 per_req 在分配时已固定（FULL_MASK 的 per_req 包含 max_context_len 上界），因此批次大小是唯一自由度。
2. 引入 max_bs 字段并冻结结构（同上文件）：VerifyMask 新增 max_bs: int，由 maybe_create_verify_mask 在分配时写入；类标记 msgspec.Struct(frozen=True)，使调整容量只能整体替换 struct，无法原地改字段，避免 buffer 与 max_bs 脱节。
3. 同步两个调用点（python/sglang/srt/speculative/eagle_worker_common.py、eagle_worker_v2.py）：build_eagle_verify_input 与 _build_trivial_verify_input 中 verify_mask.fits(bs, num_draft_tokens) 改为 verify_mask.fits(bs)。返回 False 时沿用原有回退路径：前者把 tree_mask_buf 置 None 让 build_tree_kernel_efficient 按本批分配，后者用 torch.ones(seq_lens_sum + bs) 分配全 True mask；该路径 QLEN_ONLY 此前已在使用。

4. 测试反转变更（test/registered/unit/layers/attention/test_verify_mask.py）：将 test_full_mask_always_fits 反转为 test_full_mask_does_not_fit_beyond_max_bs，用 tree_mask_numel 按 FULL_MASK 真实尺寸构造 buffer 与 max_bs，断言 fits(_MAX_BS) 为 True、fits(_MAX_BS + 1) 为 False；_mask helper 补齐 max_bs 字段，全部 fits 断言更新为新签名。

关键文件：

- python/sglang/srt/layers/attention/verify_mask.py（模块 验证掩码；类别 source；类型 core-logic；符号 VerifyMask, fits, maybe_create_verify_mask, tree_mask_numel）：核心修复文件：fits() 从只检查 QLEN_ONLY 改为统一比较 bs <= max_bs，新增 max_bs 字段并冻结 VerifyMask 结构，是本次回归修复的主逻辑。
- test/registered/unit/layers/attention/test_verify_mask.py（模块 验证掩码；类别 test；类型 test-coverage；符号 test_full_mask_does_not_fit_beyond_max_bs, test_full_mask_always_fits, TestVerifyMaskCapacity）：测试反转变更：test_full_mask_always_fits 改为 test_full_mask_does_not_fit_beyond_max_bs，并用真实 tree_mask_numel 构造 FULL_MASK buffer 验证边界。
- python/sglang/srt/speculative/eagle_worker_common.py（模块 投机解码；类别 source；类型 core-logic；符号 build_eagle_verify_input）：共享 draft 尾部逻辑调用点：fits(bs, num_draft_tokens) 改为 fits(bs)，超出 max_bs 时回退到 build_tree_kernel_efficient 按本批分配 mask。
- python/sglang/srt/speculative/eagle_worker_v2.py（模块 投机解码；类别 source；类型 core-logic；符号 _build_trivial_verify_input）：trivial verify 路径调用点：fits(bs, 1) 改为 fits(bs)，超限时用 torch.ones(seq_lens_sum + bs) 分配全 True mask。

关键符号：VerifyMask.fits, tree_mask_numel, maybe_create_verify_mask, build_eagle_verify_input, _build_trivial_verify_input

关键源码片段

python/sglang/srt/layers/attention/verify_mask.py

核心修复文件：fits() 从只检查 QLEN_ONLY 改为统一比较 bs <= max_bs，新增 max_bs 字段并冻结 VerifyMask 结构，是本次回归修复的主逻辑。

```
# tree_mask_numel: 计算树内核在给定 mode 下对 bs 个请求写入的单元数。
# FULL_MASK 的 per_req 按 max_context_len 上界固定，因此 per_req 在分配时即确定，
# 这正是后文 fits() 只需比较 bs <= max_bs 的前提。
def tree_mask_numel(
    mode: TreeMaskMode, bs: int, num_draft_tokens: int, max_context_len: int
) -> int:
    if mode == TreeMaskMode.QLEN_ONLY:
        per_req = num_draft_tokens * num_draft_tokens
    elif mode == TreeMaskMode.FULL_MASK:
        # per_req 覆盖最坏情况的上下文跨度，避免运行期依赖实际序列长度。
        per_req = num_draft_tokens * (max_context_len + num_draft_tokens)
    else:
        raise NotImplementedError(f"Invalid tree mask: {mode=}")
```

```
    return bs * per_req


class VerifyMask(msgspec.Struct, frozen=True):
    """target-verify 掩码：build_tree_kernel_efficient 在 draft 后原地写 buffer，
    从而让 worker 跳过 seq_lens_sum 的 D2H 同步。
    frozen=True 强制通过整体替换 struct 来 resize，防止 buffer 与 max_bs 失配。
    """

    buffer: torch.Tensor
    mode: TreeMaskMode
    max_bs: int # 分配时捕获的最大批次，与 buffer 大小严格对应
    is_read: bool = True

    def fits(self, bs: int) -> bool:
        """判断本批写入是否落在 buffer 内。
        旧实现只对 QLEN_ONLY 做容量检查、FULL_MASK 无条件复用；但 FULL_MASK 的
        per_req 已按 max_context_len 上界固定，因此 bs <= max_bs 对两种布局都是
        充分边界，超出则回退到每批临时分配。
        """
        return bs <= self.max_bs
```

评论区精华

PR 没有人工 review 评论（review_comments_count = 0）。作者通过 /rerun-test 触发 CI 重跑，覆盖 test_verify_mask、test_hybrid_attn_backend、test_mla_int8_deepseek_v3、test_deepep_small、EAGLE DP attention、flash attention、BCG 与 spec standalone 等用例，全部通过。4 个 commit 记录了一次设计往返：第一版直接按 max_bs 限定复用；第二版尝试"用 tree_mask_numel 动态计算容量并去掉 max_bs 字段"；第三版放弃动态计算、改回 max_bs 字段并冻结 VerifyMask；最后精简 docstring。最终方案把容量计算收敛到分配点，fits() 只剩一个标量比较，更简洁也更容易推理。

- fits() 容量判断方式：动态计算 vs max_bs 字段 (design): 采用 max_bs 字段 + frozen=True：动态计算要求调用方传递更多参数且容易遗漏，而 per_req 在分配时已固定，标量比较更简洁、更不易出错。
- FULL_MASK 是否需要容量检查 (correctness): FULL_MASK 与 QLEN_ONLY 统一用 bs <= max_bs 检查，超出即回退临时分配。

风险与影响

- 风险：行为变化：超过捕获 max_bs 的 eager batch 现在每步回退到临时分配，该路径 QLEN_ONLY 已有一定可用性，但超大动态 batch 场景可能引入少量分配开销。接口收紧：fits() 签名变化与 frozen=True 是内部 API 收紧，两个调用点已同步，但未来若有人想原地缩放 buffer 会被冻结阻止，需按整体替换 struct 的模式操作。正确性：修复消除设备端越界写与调度器挂死，对 test_gsm8k 这类此前静默错误（score 0.0）的场景，verification 结果会恢复正确。测试：单元测试覆盖了 fits 边界，但未新增直接复现 --cuda-graph-max-bs-decode 8 + max_running_requests=48 组合的端到端回归用例，回

归保护依赖现有集成测试。

- 影响：影响所有使用 verify mask 的 speculative decoding 路径（EAGLE、HybridAttnBackend、trivial verify），修复部署中可能出现的调度器挂死与设备端断言崩溃。对用户而言是稳定性和正确性修复；对团队而言是小而聚焦的回归修复，无公开 API 破坏，但 fits() 内部签名变化需要后续新增调用点注意。
- 风险标记：设备端越界修复，核心推断路径变更，回归修复，超出 max_bs 回退分配

关联脉络

- PR #33087 [Fix] Repair verify mask test fixture: 同一 test_verify_mask.py 文件的近期修复，与本次容量语义反转同属 verify mask 正确性连续工作。
- PR #32690 [Fix] missing max_context_len on HybridAttnBackend: 同为 HybridAttnBackend + EAGLE verify 正确性修复线，补齐预分配 buffer 与运行期上下文边界的一致性。