PR #32920 完整报告

sgl-project/sglang

[Spec] Compact the target-verify mask when nothing reads it

合并时间：2026-07-31 14:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32920

执行摘要

- 一句话：统一 verify mask 所有权，无人读取时压缩为 QLEN_ONLY
- 推荐动作：值得精读。该 PR 是一个高质量的重构范例：用单一所有者收敛五份分歧分配、把布局与容量绑定防止公式漂移、对无人读取的掩码走紧凑布局，并顺带修复两个派发器缺陷。重点关注 verify_mask.py 的设计（tree_mask_numel 与 fits 的分工）、fits() 对 FULL_MASK 的豁免理由，以及 TboAttnBackend 因 property 导致 __getattr__ 失效的细节。对后续做 phase 级缓冲抽象有直接参考价值。

功能与动机

Issue #32050 指出 DSV4 MTP/EAGLE 解码时 build_tree_kernel_efficient 每次 draft step 都会对预分配的 FULL_MASK 缓冲执行 fill_(True)，在 max_bs=256、num_verify_tokens=4、max_context_len=1M 时缓冲达 1,073,745,920 个 bool（约 1 GiB/rank），每步 PyTorch fill 内核耗时约 271 us，占 steady draft 段约 23%，而实际树构建内核仅约 5 us。PR body 进一步说明五个后端对同一概念缓冲在尺寸、dtype 和分配门控上存在分歧，其中 Triton 的分配低于树内核写入边界、Triton 在无 spec 时仍分配、DeepSeek-V4 的 FULL_MASK 完全未被消费，因此需要统一所有权并让无人读取的掩码走紧凑布局。

实现拆解

1. 引入掩码单一所有者 VerifyMask（新增 python/sglang/srt/layers/attention/verify_mask.py）：用 msgspec.Struct 将 buffer、布局 mode、是否被读取 is_read 绑定在一起，并提供 tree_mask_numel() 统一计算容量、fits() 做容量检查、maybe_create_verify_mask() 作为唯一分配入口。分配门控统一为 is_draft_runner or skip_prefill or not num_draft_tokens 返回 None；is_read=False 时强制使用 QLEN_ONLY，按 bs * draft^2 分配。
2. 五后端统一接入：flashattention_backend.py、deepseek_v4_backend.py、triton_backend.py、flashmla_backend.py、trtllm_mla_backend.py 以及 hybrid_linear_attn_backend.py 删除各自的 cuda_graph_custom_mask 手写分配和 get_verify_buffers_to_fill_after_draft()/target_verify_reads_custom_mask() 钩子，改为持有一个 _verify_mask 字段并暴露 verify_mask property。base_attn_backend.py 同步调整基类契约。
3. 修复 Triton 尺寸与门控缺陷：Triton 树内核布局为每请求 draft * (seq_len + draft)（由 seq_mask_len cumsum 决定），原分配 max_num_tokens * max_ctx 缺少 bs * draft^2

项；同时原门控未检查 num_draft_tokens，导致无 spec 时仍预留大块显存。统一走 maybe_create_verify_mask 后两项均按正确公式和门控处理，并显式指定 dtype=torch.uint8。

4. 修复两个派发器的掩码转发：HybridAttnBackend 原先只重写读取标志、不转发缓冲 getter，导致每次 verify step 都退回新分配掩码，现转发实际运行 target verify 的子后端的 verify_mask 与 update_verify_buffers_to_fill_after_draft；TboAttnBackend 因基类将 verify_mask 声明为 property 导致 __getattr__ 委托失效，需显式 override 并转发 primary.verify_mask。
5. 配套测试与清理：新增 test/registered/unit/layers/attention/test_verify_mask.py，覆盖布局容量、紧凑布局容量检查、分配门控、dtype 覆盖以及 Hybrid 子掩码转发；同时删除旧契约中永远为 None 的 positions 半部分及 build_tree_kernel_efficient 的 position_buf 参数。

关键文件：

- python/sglang/srt/layers/attention/verify_mask.py（模块 掩码管理；类别 source；类型 dependency-wiring；符号 tree_mask_numel, VerifyMask, fits, maybe_create_verify_mask）：新增的掩码单一所有者，定义 tree_mask_numel、VerifyMask、fits、maybe_create_verify_mask，是整个重构的核心。
- test/registered/unit/layers/attention/test_verify_mask.py（模块 掩码测试；类别 test；类型 test-coverage；符号 _create, TestVerifyMaskSizing, test_read_mask_covers_its_layouts_write_bound, test_unread_mask_drops_the_context_dimension）：新增单元测试，覆盖布局尺寸、紧凑布局容量检查、分配门控、dtype 覆盖和 Hybrid 子掩码转发。
- python/sglang/srt/layers/attention/flashattention_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 get_verify_buffers_to_fill_after_draft, target_verify_reads_custom_mask, verify_mask）：FlashAttention 后端接入 VerifyMask，并按 topk > 1 决定掩码是否可读；topk <= 1 时走紧凑布局。
- python/sglang/srt/layers/attention/deepseek_v4_backend.py（模块 DSV4 后端；类别 source；类型 core-logic；符号 get_verify_buffers_to_fill_after_draft, target_verify_reads_custom_mask, verify_mask）：DSV4 是本次性能优化的主要受益者：从不读掩码，改为 QLEN_ONLY 后缓冲从 ~1 GiB 降到 4 KiB。
- python/sglang/srt/layers/attention/triton_backend.py（模块 Triton 后端；类别 source；类型 core-logic；符号 get_verify_buffers_to_fill_after_draft, verify_mask）：修复 Triton 掩码分配尺寸不足（缺 bs * draft^2）以及无 spec 时仍分配的问题，并显式使用 uint8。
- python/sglang/srt/layers/attention/tbo_backend.py（模块 TBO 派发器；类别 source；类型 core-logic；符号 init_forward_metadata_in_graph, init_forward_metadata, get_indexer_metadata, verify_mask）：修复 TBO 派发器因 property 语义导致 __getattr__ 委托失效、掩码静默丢失的问题，显式转发 primary 掩码。
- python/sglang/srt/layers/attention/hybrid_attn_backend.py（模块 混合后端；类别 source；类型 core-logic；符号 target_verify_reads_custom_mask, verify_mask, update_verify_buffers_to_fill_after_draft）：修复 Hybrid 派发器只转发读标志、不转发缓

冲 getter 导致每步新建掩码的问题。

关键符号：tree_mask_numel, VerifyMask.fits, maybe_create_verify_mask, flashattention_backend.verify_mask, deepseek_v4_backend.verify_mask, triton_backend.verify_mask, tbo_backend.verify_mask, hybrid_attn_backend.verify_mask

关键源码片段

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

新增的掩码单一所有者，定义 tree_mask_numel、VerifyMask、fits、maybe_create_verify_mask，是整个重构的核心。

```
# python/sglang/srt/layers/attention/verify_mask.py
from __future__ import annotations

from typing import Optional

import msgspec
import torch

from sglang.srt.speculative.eagle_utils import TreeMaskMode, default_tree_mask_mode


def tree_mask_numel(
    mode: TreeMaskMode, bs: int, num_draft_tokens: int, max_context_len: int
) -> int:
    """返回树内核在 mode 下为 bs 个请求写入的 cell 数。

    FULL_MASK 在长上下文时可达数百 MB；QLEN_ONLY 保持在 KB 级。
    位打包布局不支持在此处定容 —— 如果落到 FULL_MASK 会按数量级过度分配。
    """
    if mode == TreeMaskMode.QLEN_ONLY:
        per_req = num_draft_tokens * num_draft_tokens # 紧凑布局：只按 draft 数量定容
    elif mode == TreeMaskMode.FULL_MASK:
        per_req = num_draft_tokens * (max_context_len + num_draft_tokens) # 满掩码跨 context
        维
    else:
        raise NotImplementedError(f"Invalid tree mask: {mode=}")
    return bs * per_req


class VerifyMask(msgspec.Struct):
    """target-verify 阶段的掩码缓冲。

    build_tree_kernel_efficient 在 draft 后原地写 buffer，这是 worker 跳过
    seq_lens_sum D2H 同步的前提。buffer、布局、是否可读必须放在一起维护：
    只拿走 buffer 而不知道布局，内核写出的 shape 会被读者误读。内核即使
    在无人读取时也会写满每个 cell，因此 buffer 总是会被分配。
    """
```

```
    buffer: torch.Tensor
    mode: TreeMaskMode
    is_read: bool = True

    def fits(self, bs: int, num_draft_tokens: int) -> bool:
        """判断本批次写入是否落在缓冲内。

        只检查紧凑布局。FULL_MASK 保留原有的无条件复用 —— 它的上界依赖
        max_context_len，而复合后端（如 Hybrid）不携带该值，所以超过 max_bs
        的批次仍可能溢出，这与变更前行为一致。
        """
        if self.mode != TreeMaskMode.QLEN_ONLY:
            return True
        return self.buffer.numel() >= bs * num_draft_tokens * num_draft_tokens


def maybe_create_verify_mask(
    *,
    is_draft_runner: bool,
    skip_prefill: bool,
    max_bs: int,
    max_context_len: int,
    num_draft_tokens: Optional[int],
    device: torch.device | str,
    is_read: bool,
    dtype: torch.dtype = torch.bool,
) -> Optional[VerifyMask]:
    """按捕获到的 max batch 分配；没有 verify 阶段时返回 None。"""
    if is_draft_runner or skip_prefill or not num_draft_tokens:
        return None
    # 无人读取时用紧凑布局，省掉 context 维；有读者时沿用默认布局。
    mode = default_tree_mask_mode() if is_read else TreeMaskMode.QLEN_ONLY
    return VerifyMask(
        buffer=torch.zeros(
            tree_mask_numel(mode, max_bs, num_draft_tokens, max_context_len),
            dtype=dtype,
            device=device,
        ),
        mode=mode,
        is_read=is_read,
    )
```

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

FlashAttention 后端接入 VerifyMask，并按 topk > 1 决定掩码是否可读；topk <= 1 时走紧凑布局。

```
# python/sglang/srt/layers/attention/flashattention_backend.py（init_cuda_graph_state 节选）
# 在 CUDA graph 状态初始化时统一创建 verify mask。
# topk <= 1 的验证路径从不提取掩码，所以此时掩码无人读取，
```

```
# 直接使用紧凑 QLEN_ONLY 布局，省掉 context 维的显存。
self._verify_mask = maybe_create_verify_mask(
    is_draft_runner=self.is_draft_runner,
    skip_prefill=self.skip_prefill,
    max_bs=max_bs,
    max_context_len=self.max_context_len,
    num_draft_tokens=self.speculative_num_draft_tokens,
    device=self.device,
    is_read=self.topk > 1, # topk > 1 时 flashinfer 路径才可能读掩码
)
```

评论区精华

该 PR 没有 review 评论（review_comments_count=0），设计讨论集中在 PR body 与 issue 中：

- 关于布局与缓冲必须同时变更：PR body 强调“taking the buffer without its layout would have the kernel write a shape the reader does not expect”，因此 VerifyMask 将 buffer、mode、is_read 绑定为一个整体。
- 关于 FULL_MASK 的容量检查豁免：fits() 只对紧凑布局生效，因为 FULL_MASK 的容量上界依赖 max_context_len，复合后端（如 Hybrid）不携带该值，故保留原有“无条件复用”行为，batch 超过 max_bs 时仍可能溢出，这是沿用旧行为的显式取舍。
- 关于 HybridAttnBackend 的变更性质：PR body 明确指出“This is a behavior change, not a pure refactor”，因为此前它总是退回新建掩码，现在会正确转发子后端掩码。
- 暂无高价值评论线程

风险与影响

- 风险：
  1. 行为变更风险：HybridAttnBackend 从“每次新建掩码”变为“复用子后端掩码”，若子后端选择逻辑与 CUDA graph 捕获不一致，可能影响 verify 阶段掩码内容；TboAttnBackend 显式 override 若未来基类 property 语义变化，可能产生静默回退。
  2. 容量溢出风险：QLEN_ONLY 布局没有 context 维度余量，fits() 只检查该布局；eager 批处理超过 max_bs 时会回退到新分配，但 FULL_MASK 仍保留旧的“无条件复用”，当 draft * sum(seq_len) 超过缓冲时依然可能越界，这与变更前行为一致，属于历史遗留风险。
  3. Triton 尺寸修正的影响：Triton 掩码从偏小改为准确尺寸，若树内核实际写入边界与 tree_mask_numel 公式仍有出入，可能暴露新的越界写；好在 tree_mask_numel 同时是分配公式和容量检查，二者不再漂移。
  4. dtype 差异：Triton 使用 uint8，其余为 bool，统一入口保留 dtype 参数，但若某处误用默认 bool 读取 Triton 缓冲会出问题。
  5. 多后端连锁影响：DeepSeek-V4、FlashMLA、TRTLLM-MLA 均切换为 QLEN_ONLY，这些后端虽然当前不读掩码，但若未来启用提取掩码功能而未同步改 is_read，会得到错误布局。 - 影响：用户 / 系统影响：在长上下文（1M token）且启用 MTP/EAGLE 推测

解码的 DeepSeek-V4 场景，每 rank 显存占用减少约 1 GiB，每步 fill 开销从 271 us降到接近零，稳态 draft 段延迟明显改善；FlashMLA、TRTLLM-MLA 同样受益。代码影响：五个后端的掩码分配逻辑收敛到一个入口，基类契约 verify_mask property 取代旧的双钩子，涉及 base_attn_backend.py、eagle_worker_common.py 等公共路径；同时暴露并修复了 Hybrid/Tbo 派发器静默丢掩码的隐藏缺陷。团队影响：为后续将掩码迁移为 phase 级缓冲（如纳入 CudaGraphBufferRegistry）提供了单一边界，降低维护成本。
- 风险标记：Hybrid/Tbo 转发为行为变更，FULL_MASK 超限行为保留，QLEN_ONLY容量检查仅限紧凑布局，多后端连锁影响

关联脉络

- PR #32060 [Spec][DSV4] Avoid oversized speculative verify-mask fill: 本 PR 的紧凑布局方案源于并取代该 PR（body 中明确 supersedes #32060），同样针对 #32050 的 1 GiB fill 问题。
- PR #31430 Remove unused draft-extend CUDA graph top-k: 同属 speculative decoding 路径的无用缓冲 / 计算清理，目标一致，可关联阅读。
- PR #32692 [gdn] support replayssm with extra buffer: 同样修改了 speculative 与 mem_cache 相关缓冲区管理，与本 PR 的 phase 级缓冲演进方向相关。