PR #25037 完整报告

sgl-project/sglang

spec: STANDALONE skips hidden_states end-to-end (Optional schema + None-safe consumers)

合并时间：2026-05-13 03:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25037

执行摘要

- 一句话：STANDALONE 投机解码跳过 hidden_states 端到端捕获
- 推荐动作：值得精读，尤其是 Optional schema 的设计和 None 守卫的分布模式，可作为类似架构变更的参考。重点关注 *eagle_info.py* 中的 *classmethod* 返回类型变更和每个 *producer* 站点的 *capture_hidden_mode* 三元表达式。

功能与动机

STANDALONE 投机解码使用 vanilla LLM 作为 draft 模型，架构上从不读取 spec_info.hidden_states。原有的实现仍然会捕获、分配、拷贝 hidden_states，不仅浪费 GPU 内存，还在 target/draft hidden_size 不一致时暴露形状不匹配 bug。本 PR 通过架构级不变量——STANDALONE 模式下 *spec_info.hidden_states* 恒为 *None*——实现端到端跳过，统一解决 #21434 和 #14563 中修复过的类似问题。

实现拆解

1. Schema 调整：在 EagleDraftInput 类中将 hidden_states 字段改为 Optional[torch.Tensor]；hidden_size_for() 和 dtype_for() 返回 Optional[int] 和 Optional[torch.dtype]，对于 STANDALONE 返回 None；*create_idle_input*() 根据 hidden_size 是否为 None 决定是否创建空张量。
2. Producer 侧统一 NULL 模式：在 *eagle_worker.py*、multi_layer_eagle_worker.py、*eagle_worker_v2.py*、*eagle_info_v2.py*、*cuda_graph_runner.py* 等文件的 24 个 *capture_hidden_mode* 赋值点，使用三元表达式 *CaptureHiddenMode.NULL if self.speculative_algorithm.is_standalone() else* 原值，确保 STANDALONE 模式下不会触发 hidden_states 捕获。
3. Consumer 侧 None 守卫：在所有读取 / 切片 / 拷贝 spec_info.hidden_states 或 logits_output.hidden_states 的地方添加 if ... is not None 条件，包括 *eagle_info.py* 的 verify() 中两次构建 EagleDraftExtendInput 时的切片、*eagle_draft_cuda_graph_runner.py* 中 buffer 分配与 replay 拷贝、*eagle_worker.py* 的 verify() 中隐藏状态切片、*spec_utils.py* 中的 shape 检查等。同时 FutureMap 中 *spec_need_hidden_states*() 对 STANDALONE 返回 False 从而跳过 hidden_states_buf 的存储和加载。
4. 测试配套：在 test_standalone_speculative_decoding.py 中新增 test_radix_attention 方法，通过 radix-tree 压力测试验证 Optional schema 与 None-safe 组合的正确性。

关键文件：

- python/sglang/srt/speculative/eagle_info.py（模块 投机解码；类别 source；类型 core-logic；符号 hidden_size_for, dtype_for）：核心数据类 EagleDraftInput 的 hidden_states 字段改为 Optional，hidden_size_for/dtype_for 返回 Optional，create_idle_input 条件创建，是本次变更的 schema 基础。
- python/sglang/srt/speculative/eagle_draft_cuda_graph_runner.py（模块 CUDA 图；类别 source；类型 dependency-wiring）：CUDA graph runner 的 buffer 分配和 replay 需要适配 Optional hidden_states，是本变更在 CUDA graph 路径上的关键实现。
- python/sglang/srt/speculative/eagle_worker.py（模块 投机解码；类别 source；类型 core-logic）：EAGLEWorker 中包含多处 capture_hidden_mode 设置和 verify 中的 None 守卫，是 producer 侧和 consumer 侧变化最大的文件。
- python/sglang/srt/speculative/multi_layer_eagle_worker.py（模块 投机解码；类别 source；类型 core-logic）：多层 EAGLE worker 同样需要修改 capture_hidden_mode 设置和 forward_draft_extend_after_decode 中的 None 守卫。
- test/registered/spec/test_standalone_speculative_decoding.py（模块 测试；类别 test；类型 test-coverage；符号 test_radix_attention）：新增 test_radix_attention 测试方法，覆盖 radix-tree 压力场景下 Optional schema 和 None-safe 组合。

关键符号：EagleDraftInput.hidden_size_for, EagleDraftInput.dtype_for, EagleDraftInput.create_idle_input, EagleDraftInput.prepare_for_extend, EAGLEWorker.forward_target_extend, EAGLEWorker._draft_preprocess_idle, EAGLEWorker.draft, EAGLEWorker.verify, MultiLayerEAGLEWorker.forward_target_extend, MultiLayerEAGLEWorker.draft, MultiLayerEAGLEWorker.forward_draft_extend_after_decode, EAGLEDraftCudaGraphRunner.init, EAGLEDraftCudaGraphRunner.replay, EAGLEDraftCudaGraphRunner.capture_one_batch_size

关键源码片段

python/sglang/srt/speculative/eagle_info.py

核心数据类 EagleDraftInput 的 hidden_states 字段改为 Optional，hidden_size_for/dtype_for 返回 Optional，create_idle_input 条件创建，是本次变更的 schema 基础。

```
# python/sglang/srt/speculative/eagle_info.py

class EagleDraftInput(SpecInput, EagleDraftInputV2Mixin):
    topk_p: torch.Tensor = None
    topk_index: torch.Tensor = None
    # None when the spec algorithm's draft doesn't read hidden_states
    # (e.g., STANDALONE — vanilla LLM draft).
    hidden_states: Optional[torch.Tensor] = None
    capture_hidden_mode: CaptureHiddenMode = CaptureHiddenMode.FULL
    # ...

    @classmethod
```

```
    def hidden_size_for(cls, worker) -> Optional[int]:
        """Decode-phase `hidden_states` width. Returns None when the draft
        architecture doesn't consume the field (e.g., STANDALONE)."""
        if worker.speculative_algorithm.is_standalone():
            return None
        return _draft_runner_of(worker).model_config.spec_hidden_size

    @classmethod
    def dtype_for(cls, worker) -> Optional[torch.dtype]:
        if worker.speculative_algorithm.is_standalone():
            return None
        return _draft_runner_of(worker).model_config.dtype

    @classmethod
    def create_idle_input(
        cls,
        device: torch.device,
        hidden_size: Optional[int],
        dtype: Optional[torch.dtype],
        topk: int,
        capture_hidden_mode: CaptureHiddenMode,
    ):
        return cls(
            bonus_tokens=torch.empty((0,), device=device, dtype=torch.int32),
            hidden_states=(
                torch.empty((0, hidden_size), device=device, dtype=dtype)
                if hidden_size is not None
                else None
            ),
            topk_p=torch.empty((0, topk), device=device, dtype=torch.float32),
            topk_index=torch.empty((0, topk), device=device, dtype=torch.int64),
            capture_hidden_mode=capture_hidden_mode,
        )
```

python/sglang/srt/speculative/eagle_draft_cuda_graph_runner.py

CUDA graph runner 的 buffer 分配和 replay 需要适配 Optional hidden_states，是本变更在 CUDA graph 路径上的关键实现。

```
# python/sglang/srt/speculative/eagle_draft_cuda_graph_runner.py

@dataclass
class EagleDraftInputBuffers(ForwardInputBuffers):
    input_ids: torch.Tensor
    req_pool_indices: torch.Tensor
    out_cache_loc: torch.Tensor
    positions: torch.Tensor
    mrope_positions: torch.Tensor
    seq_lens: torch.Tensor
    seq_lens_cpu: torch.Tensor
```

```
    extend_seq_lens: torch.Tensor
    topk_p: torch.Tensor
    topk_index: torch.Tensor
    hidden_states: Optional[torch.Tensor] # None when STANDALONE
    global_num_tokens_gpu: Optional[torch.Tensor]
    global_num_tokens_for_logprob_gpu: Optional[torch.Tensor]

# In __init__:
_hidden_size = EagleDraftInput.hidden_size_for(self.eagle_worker)
hidden_states = (
    torch.zeros(
        (self.max_bs, _hidden_size),
        dtype=EagleDraftInput.dtype_for(self.eagle_worker),
    )
    if _hidden_size is not None
    else None
)

# In replay:
if (
    buffers.hidden_states is not None
    and forward_batch.spec_info.hidden_states is not None
):
    buffers.hidden_states[:raw_bs].copy_(
        forward_batch.spec_info.hidden_states
    )
```

评论区精华

PR 没有公开的 review 讨论。作者在 body 中详细说明了设计动机和变更范围，并引用了 #21434 和 #14563 作为相关修复。在 merge commit 中处理了与 #25038 重命名冲突（accepted_indices → accept_indices），确保了 None 守卫与新命名共存。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 遗漏 None 守卫：变更点分散在 24 个 producer 站点和多个 consumer 站点，可能存在遗漏的 hidden_states 访问路径，导致 STANDALONE 模式下的 AttributeError 或类型错误。
  2. 回归影响：对于非 STANDALONE 算法（EAGLE、EAGLE3、FROZEN_KV_MTP、多层 EAGLE），应完全无行为变化，但需要依赖已有测试覆盖。
  3. merge 冲突：merge commit 解决了与 #25038 的命名冲突，但手动解决可能引入错误，需确认 guard 逻辑正确。
  4. 测试覆盖：新增的 test_radix_attention 仅覆盖 radix 场景，缺少对其他 consumer 路径的专项测试。 - 影响：对用户：STANDALONE 模式用户将显著减少 GPU 内存使用并

避免 hidden_size 不匹配导致的崩溃；其他模式用户无影响。对系统：减少了不必要的 hidden_states 传播计算和显存占用，轻微降低 decode 延迟。对团队：架构不变量更清晰，后续添加新算法时需遵循 Optional 模式。 - 风险标记：24 处修改点可能遗漏None守卫，STANDALONE 模式回归风险，需确保 EAGLE 模式无影响，merge 冲突手动解决风险

关联脉络

- PR #25038 [Spec] Rename accepted_indices -> accept_indices; drop _token_id suffix per Rule 5: 在 merge 过程中与本 PR 产生冲突，需要协调重命名和 None 守卫的变更。
- PR #21434 Fix standalone speculative decoding hidden_states None issue: 本 PR 旨在统一修复与 #21434 相同根因的问题，但采用端到端方案。
- PR #14563 Fix hidden_size mismatch in speculative decoding: 同样涉及 hidden_size 形状不匹配问题，本 PR 作为统一修复。