PR #29464 完整报告

sgl-project/sglang

Fix EAGLE draft hidden dim extraction and centralize spec helpers

合并时间：2026-06-28 12:48

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29464

执行摘要

- 一句话：修复 EAGLE draft hidden dim 提取并集中规格函数
- 推荐动作：该 PR 值得精读，因为它展示了如何通过集中化辅助函数消除重复逻辑并修复隐蔽的维度错误。设计决策如使用 config 驱动而非模型层属性是更稳健的做法。此外，prefill CUDA graph runner 中 input_embeds slot 的注册注释清晰地解释了 multimodal 路径的需求。建议团队后续补充相关测试。

功能与动机

PR body 指出：部分架构中读取 fc.in_features 来获取 draft hidden dim 是不正确的，需要改用 config 驱动的方式。同时原有 hidden_size_for/dtype_for 类方法散落在多个类中，导致重复和容易出错。集中化后统一从 model_runner 的 config 中派生，避免手动维护。

实现拆解

1. 新增集中式辅助函数：在 python/sglang/srt/speculative/eagle_utils.py 中新增 get_draft_input_from_target_hidden_dim 和 get_draft_recurrent_hidden_state_spec。前者根据 config（EAGLE3 aux 模式等）计算目标 hidden states 宽度，后者返回 draft 循环时需要的 hidden states 尺寸和 dtype。两者均从 draft 的 model_runner 读取 config，成为单一事实来源。
2. 删除散落的类方法：在 python/sglang/srt/speculative/eagle_info.py 中删除 EagleDraftInput.hidden_size_for、dtype_for 以及 EagleDraftExtendInput.hidden_size_for、dtype_for，同时删除辅助函数 _draft_runner_of。这些功能全部由新函数替代。
3. 更新调用方：修改 eagle_worker_v2.py、eagle_draft_cuda_graph_runner.py、eagle_draft_extend_cuda_graph_runner.py、multi_layer_eagle_worker_v2.py、multi_layer_eagle_draft_extend_cuda_graph_runner.py，将原有对 hidden_size_for/dtype_for 的调用替换为 get_draft_recurrent_hidden_state_spec 或 get_draft_input_from_target_hidden_dim。
4. 修复 prefill CUDA graph runner：在 prefill_cuda_graph_runner.py 中，为 breakable backend 注册 multimodal input_embeds slot，使得 captured graph 在 multimodal batch replay 时能正确填充 vision embeddings。同时将 draft hidden dim 的计算从尝试读取 fc.in_features 改为统一使用 get_draft_input_from_target_hidden_dim。

5. 清理 scheduler 中的分散逻辑：在 python/sglang/srt/managers/scheduler.py 的 init_disaggregation 中，将原来内联的 hidden size/dtype 分支提取为使用 get_draft_recurrent_hidden_state_spec，消除了重复的三元表达式。
6. 附加风格调整：使用 PEP 617 的 parenthesized context manager 语法（with (... , ...):）改写多处 context manager。

关键文件：

- python/sglang/srt/speculative/eagle_utils.py（模块 推测解码；类别 source；类型 core-logic；符号 get_draft_input_from_target_hidden_dim, get_draft_recurrent_hidden_state_spec, get_draft_hidden_dim）：集中化的 hidden states 尺寸 / 类型解析函数所在文件，新增了两个核心函数，替换了原有分散的类方法。
- python/sglang/srt/speculative/eagle_info.py（模块 推测解码；类别 source；类型 core-logic；符号 _draft_runner_of, hidden_size_for, dtype_for）：删除了多余的类方法 hidden_size_for、dtype_for 和辅助函数 _draft_runner_of，减少了代码冗余。
- python/sglang/srt/model_executor/runner/prefill_cuda_graph_runner.py（模块 执行器；类别 source；类型 data-contract）：修复 multimodal batch 的 input_embeds slot 注册，并统一使用新函数获取 draft hidden dim，是 multimodal 路径正确性的关键。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 dependency-wiring）：scheduler 中 init_disaggregation 的逻辑被简化为使用新函数，消除了重复的条件分支，更易维护。
- python/sglang/srt/speculative/eagle_worker_v2.py（模块 推测解码；类别 source；类型 core-logic）：draft worker 中多处调用迁移到新函数，同时移除了 eagle_use_aux_hidden_state 属性（由新函数内部处理）。
- python/sglang/srt/speculative/eagle_draft_extend_cuda_graph_runner.py（模块 推测解码；类别 source；类型 dependency-wiring）：draft extend CUDA graph runner 中 hidden states 尺寸和类型的获取改为使用新函数。
- python/sglang/srt/speculative/multi_layer_eagle_worker_v2.py（模块 推测解码；类别 source；类型 core-logic）：多层级 draft worker 中同步更新对新函数的调用。
- python/sglang/srt/speculative/eagle_draft_cuda_graph_runner.py（模块 推测解码；类别 source；类型 dependency-wiring）：draft decode CUDA graph runner 中更新为新函数。
- python/sglang/srt/speculative/multi_layer_eagle_draft_extend_cuda_graph_runner.py（模块 推测解码；类别 source；类型 dependency-wiring）：多层级 draft extend CUDA graph runner 中同步更新。
- python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py（模块 执行器；类别 source；类型 data-contract）：decode CUDA graph runner 中有小幅改动（使用 supports_target_verify_for_draft 替换 is_dflash）。

关键符号：get_draft_input_from_target_hidden_dim, get_draft_recurrent_hidden_state_spec, hidden_size_for, dtype_for

关键源码片段

python/sglang/srt/speculative/eagle_utils.py

集中化的 hidden states 尺寸 / 类型解析函数所在文件，新增了两个核心函数，替换了原有分散的类方法。

```
def get_draft_input_from_target_hidden_dim(model_runner: ModelRunner) -> int:
    """Width of the target hidden states fed into the draft model.

    This is the single source of truth and is derived entirely from config:
    for EAGLE3 aux mode the draft consumes `num_aux` concatenated target
    layers (each `target_hidden_size` wide); every other arch consumes the
    per-layer `spec_hidden_size`.

    Do NOT read this off a draft projection's `in_features` (e.g. an `fc`
    layer): that width is arch-specific.

    Note: read entirely from the *draft* `model_runner`'s config. The non-aux
    branch assumes the draft's `spec_hidden_size` equals the target hidden
    width fed to the draft (true for standard EAGLE, where the draft mirrors
    the target hidden size); aux mode reads the explicit `target_hidden_size`.
    """
    model_config = model_runner.model_config
    hf_config = model_config.hf_config
    eagle_config = getattr(hf_config, "eagle_config", None) or {}
    get_eagle_config = (
        eagle_config.get
        if isinstance(eagle_config, dict)
        else lambda key, default=None: getattr(eagle_config, key, default)
    )
    use_aux = get_eagle_config("use_aux_hidden_state", True)
    spec_algorithm = model_runner.spec_algorithm

    # 若非 EAGLE3 aux 模式，直接返回 spec_hidden_size
    if not (spec_algorithm is not None and spec_algorithm.is_eagle3() and use_aux):
        return model_config.spec_hidden_size

    # EAGLE3 aux: width = target_hidden_size * num_aux
    target_hidden = getattr(hf_config, "target_hidden_size", None)
    if target_hidden is None:
        target_hidden = model_config.hidden_size
    num_aux = getattr(hf_config, "num_aux_hidden_states", None)
    if num_aux is None:
        layer_ids = get_eagle_config("eagle_aux_hidden_state_layer_ids", None)
        if layer_ids is None:
            layer_ids = getattr(hf_config, "eagle_aux_hidden_state_layer_ids", None)
        num_aux = len(layer_ids) if layer_ids else 3
    return target_hidden * num_aux

def get_draft_recurrent_hidden_state_spec(
    model_runner: ModelRunner,
```

```
) -> tuple[Optional[int], Optional[torch.dtype]]:
    """Return hidden_states width/dtype carried between draft decode steps."""
    if model_runner.spec_algorithm.is_standalone():
        return None, None
    return model_runner.model_config.spec_hidden_size, model_runner.model_config.dtype
```

python/sglang/srt/model_executor/runner/prefill_cuda_graph_runner.py

修复 multimodal batch 的 input_embeds slot 注册，并统一使用新函数获取 draft hidden dim，是 multimodal 路径正确性的关键。

```
# 在 build_prefill_registry 调用中添加 source=self.buffers,
# 使得 multimodal 的 input_embeds slot 被注册到 buffer_registry 中
self.buffer_registry: CudaGraphBufferRegistry = build_prefill_registry(
    device=self.device,
    max_bs=self.max_bs,
    max_num_token=self.max_num_tokens,
    cache_loc_dtype=self._cache_loc_dtype(),
    is_multimodal=self.is_multimodal,
    hidden_size=self.model_runner.model_config.hidden_size,
    embed_dtype=self.model_runner.dtype,
    enable_mamba_track=self.mamba_track_enabled,
    # 注册 multimodal input_embeds slot（默认 True）。
    # 仅在 is_multimodal 时添加，纯文本模型不受影响。
    # tc_piecewise 和 breakable 后端都需要此 slot,
    # 否则 captured graph 会重新对 input_ids 做 embedding 而丢掉 vision embeddings。
    source=self.buffers,
)

# 创建 static_draft_hidden_states 时，统一使用集中式函数
if (
    isinstance(self.backend, BreakableCudaGraphBackend)
    and model_runner.is_draft_worker
    and model_runner.spec_algorithm.is_eagle()
):
    hidden_dim = get_draft_input_from_target_hidden_dim(model_runner)
    with torch.device(self.device):
        self.static_draft_hidden_states = torch.zeros(
            (self.max_num_tokens, hidden_dim),
            dtype=self.model_runner.dtype,
        )
```

评论区精华

1. 删除注释：merrymercy 在 review 中指出 prefill_cuda_graph_runner.py 中新加的注释过于冗余，要求删除（commit 中已执行）。
2. 硬编码整数：在 prefill_cuda_graph_runner.py 中 replay_layer_forward 使用了硬编码整数 1 作为参数索引，merrymercy 要求改为 inspect.signature 动态获取参数位置（commit 中已修复）。

3. follow-up 清理：PR body 末尾提到 init_disaggregation 中 model_config 和 fallback block 已过时，建议简化忽略 get_draft_kv_pool 的第二个返回值（未在当前 PR 中处理，留待后续）。

- 删除冗余注释 (style): 相关注释已删除（在 commit 中体现）。
- 硬编码整数参数索引 (design): 已修改为使用 inspect.signature 获取实际参数位置。
- scheduler 中废弃代码的后续清理 (other): 未在当前 PR 中处理，标记为 non-blocking cleanup，留待将来。

风险与影响

- 风险：
  1. 回归风险：涉及多个调用点替换，如果某个调用点未更新或新函数返回值与预期不符，可能导致 draft hidden states 尺寸错误，影响推测解码的正确性。
  2. 配置兼容性：新函数假设 model_runner.model_config 包含正确的 spec_hidden_size 等字段，若某些自定义模型配置缺少这些字段，可能引发异常。
  3. multimodal 路径：prefill CUDA graph runner 的 input_embeds slot 变更可能影响 multimodal batch 的图捕获和回放，若未正确配置可能导致推理错误。
  4. 缺少测试覆盖：本次变更未包含新增测试，集中化的逻辑和 multimodal 修复缺乏回归测试覆盖。 - 影响：影响范围：所有使用 EAGLE 推测解码（包括 EAGLE、EAGLE3、DFLASH 等）的模型，特别是涉及 hidden states 传递的 draft worker。同时影响 multimodal 模型的 prefill CUDA graph 路径。影响程度中等，因为修复了一个关键的正确性问题并清理了可维护性债务。无用户可见的功能变化（属内部重构 + 修复）。 - 风险标记：核心路径变更，缺少测试覆盖，配置兼容性，multimodal 路径

关联脉络

- PR #29395 [Spec] Capture DFLASH draft greedy sampling inside the draft decode cuda graph: 同为推测解码模块的 draft 优化，涉及 draft 运行时的尺寸计算，本 PR 修复了维度提取方式，可能与该 PR 的 CUDA graph 捕获配合生效。
- PR #29223 (perf): Shard Kimi-K2.5 Eagle3 draft fc + symm-mem AG: 涉及 EAGLE3 draft fc 的分片优化，本 PR 新增的 get_draft_input_from_target_hidden_dim 专门处理 EAGLE3 aux 模式，两者功能上有直接关联。