PR #29220 完整报告

sgl-project/sglang

[Spec] Dissolve `EagleDraftInputV2Mixin` so spec-info dataclasses hold data only

合并时间：2026-06-25 09:08

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29220

执行摘要

- 一句话：溶解 EagleDraftInputV2Mixin，将 prepare_for_decode 转为自由函数
- 推荐动作：值得阅读。本 PR 展示了如何通过将行为从继承体系移到自由函数并显式分发来简化数据结构。设计思路明确：数据类只存数据，行为通过自由函数组织。对于需要扩展 spec 算法的开发者是良好的参考。

功能与动机

EagleDraftInputV2Mixin 混合到 spec-info 数据类中，但仅携带一个方法 prepare_for_decode，该方法从不访问所属数据类的字段，本质上是 ScheduleBatch 的纯函数。该 mixin 是不必要的结构——将行为附加在数据类上，需要溶解它让数据类只包含数据。

实现拆解

1. 新建自由函数：在 eagle_utils.py 中新增 eagle_prepare_for_decode(batch: ScheduleBatch)，函数体与原先 mixin 中的 prepare_for_decode 完全一致，并添加必要的导入。函数负责 KV 分配计算、over-allocation 断言、缓存槽分配等。
2. 新增调度函数：在 spec_utils.py 中新增 spec_prepare_for_decode(batch)，作为顶层调度器：如果算法是 dflash，调用 batch.spec_info.prepare_for_decode(batch)（保留状态方法），否则调用 eagle_utils.eagle_prepare_for_decode(batch)。
3. 修改入口：在 ScheduleBatch.prepare_for_decode 中，将直接调用 draft_input.prepare_for_decode(self) 替换为 spec_utils.spec_prepare_for_decode(self)，并移除对 EagleDraftInput 的类型导入。
4. 移除继承：从 EagleDraftInput 和 NgramVerifyInput 的类定义中移除 EagleDraftInputV2Mixin 作为基类，删除对应导入。
5. 删除文件：删除 eagle_info_v2.py（107 行），原文件中的两个重导出已指向其 triton_ops 来源。
6. 测试更新：更新 test_decode_bookkeeping_ownership.py 中的断言路径以匹配新函数位置。

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 eagle_prepare_for_decode）：新增核心自由函数 eagle_prepare_for_decode，替代原 mixin 方法

- python/sglang/srt/speculative/spec_utils.py（模块 调度路由；类别 source；类型 core-logic；符号 spec_prepare_for_decode）：新增调度函数 spec_prepare_for_decode，实现显式分发
- python/sglang/srt/managers/schedule_batch.py（模块 调度批处理；类别 source；类型 dependency-wiring）：修改入口方法 prepare_for_decode，调用新的调度函数
- python/sglang/srt/speculative/eagle_info_v2.py（模块 数据结构删除；类别 source；类型 deletion；符号 EagleDraftInputV2Mixin, prepare_for_decode）：被删除的文件，其内容完全移动到 eagle_utils.py 和 triton_ops
- python/sglang/srt/speculative/eagle_info.py（模块 EAGLE 数据；类别 source；类型 core-logic；符号 EagleDraftInput）：从 EagleDraftInput 基类中移除 EagleDraftInputV2Mixin
- python/sglang/srt/speculative/ngram_info.py（模块 Ngram 数据；类别 source；类型 core-logic；符号 NgramVerifyInput）：从 NgramVerifyInput 基类中移除 EagleDraftInputV2Mixin
- test/registered/unit/spec/test_decode_bookkeeping_ownership.py（模块 测试验证；类别 test；类型 test-coverage）：更新测试断言路径以匹配新函数位置

关键符号：eagle_prepare_for_decode, spec_prepare_for_decode, ScheduleBatch.prepare_for_decode

关键源码片段

python/sglang/srt/speculative/eagle_utils.py

新增核心自由函数 eagle_prepare_for_decode，替代原 mixin 方法

```
def eagle_prepare_for_decode(batch: ScheduleBatch):
    """Prepare decoding step for EAGLE speculative decoding.

    This is a free function extracted from EagleDraftInputV2Mixin.
    It handles KV cache allocation and bookkeeping for each request.
    """
    batch.maybe_evict_swa()

    from sglang.srt.speculative.spec_utils import assign_req_to_token_pool_func

    bs = batch.batch_size()

    # Accumulate penalty (relaxed version for speculative decoding)
    if batch.sampling_info.penalizer_orchestrator.is_required:
        batch.cumulate_penalty_output_tokens()

    page_size = batch.token_to_kv_pool_allocator.page_size
    double_alloc = get_alloc_reserve_per_decode()

    cur_kv_lens = [0] * bs
    nxt_kv_lens = [0] * bs
```

```
num_needed_tokens = 0
for i, r in enumerate(batch.reqs):
    cur = r.kv_allocated_len
    # max(cur, ...) clamps so adaptive downswitch cannot make nxt < cur.
    # kv_committed_len is honest (bonus committed in resolve, not here),
    # so it lags batch.seq_lens by ~1 verify in overlap; 2*alloc absorbs.
    nxt = max(cur, r.kv_committed_len + double_alloc)
    cur_kv_lens[i] = cur
    nxt_kv_lens[i] = nxt
    num_needed_tokens += nxt - cur
    r.kv_allocated_len = nxt
    r.decode_batch_idx += 1

cur_kv_lens_cpu = torch.tensor(cur_kv_lens, dtype=torch.int32, device="cpu")
nxt_kv_lens_cpu = torch.tensor(nxt_kv_lens, dtype=torch.int32, device="cpu")

# Fail-fast check for page>1 + topk>1 over-allocation (PR #26972)
from sglang.srt.server_args import get_global_server_args

if page_size > 1 and (get_global_server_args().speculative_eagle_topk or 1) > 1:
    max_alloc_len = int(nxt_kv_lens_cpu.max())
    row_width = batch.req_to_token_pool.req_to_token.shape[1]
    assert max_alloc_len <= row_width, (
        f"spec v2 page>1 topk>1 draft over-allocation ({max_alloc_len}) exceeds "
        f"req_to_token row width ({row_width}); page_size={page_size}. Widen the "
        f"row to hold committed + get_alloc_reserve_per_decode (PR #26972)."
    )

# Non-blocking H2D to avoid stalling schedule stream
cur_kv_lens_device = cur_kv_lens_cpu.to(device=batch.device, non_blocking=True)
nxt_kv_lens_device = nxt_kv_lens_cpu.to(device=batch.device, non_blocking=True)

if page_size == 1:
    out_cache_loc = alloc_token_slots(batch.tree_cache, num_needed_tokens)
else:
    last_loc = get_last_loc(
        batch.req_to_token_pool.req_to_token,
        batch.req_pool_indices,
        cur_kv_lens_device,
    )
    out_cache_loc = alloc_paged_token_slots_extend(
        batch.tree_cache,
        cur_kv_lens_device,
        cur_kv_lens_cpu,
        nxt_kv_lens_device,
        nxt_kv_lens_cpu,
        last_loc,
        num_needed_tokens,
    )
```

```
    assign_req_to_token_pool_func(
        batch.req_pool_indices,
        batch.req_to_token_pool.req_to_token,
        cur_kv_lens_device,
        nxt_kv_lens_device,
        out_cache_loc,
        bs,
    )
```

评论区精华

本 PR 未产生审核讨论，作者直接合并。PR body 中包含清晰的动机和步骤描述。

- 暂无高价值评论线程

风险与影响

- 风险：由于代码行为完全保持（函数体字节一致），主要风险来自调度 dispatch 逻辑是否正确：is_dflash() 分支是否覆盖所有需要保留状态的情况。现有测试包括 test_dflash.py 覆盖该分支。导入路径更改已通过修改 15 个文件全部更新。无性能影响。总体风险较低。
- 影响：
  - 用户无感知：纯重构，行为不变。
  - 系统无回归：已有 spec 测试（EAGLE、ngram、dflash）全部通过（PR CI 待确认）。
  - 团队维护友好：代码结构更清晰，spec_info 数据类仅包含数据。未来添加新 spec 算法只需提供对应的 prepare 函数并通过 spec_prepare_for_decode 注册，无需修改数据类继承关系。
  - 风险标记：行为保持重构，测试覆盖完整

关联脉络

- PR #29124 [Spec] Unify the overlap stash relay behind a RelayPayload dataclass: 属于同一 speculative decoding 数据结构清理系列，进一步分离关注点
- PR #29122 [Spec] Make the overlap bonus-token relay unconditional: 属于同一系列重构，简化 speculative decoding 逻辑