PR #28117 完整报告

sgl-project/sglang

[Spec] Move eagle verify `prepare_for_verify`/`sample` to `eagle_utils` free helpers

合并时间：2026-06-13 11:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28117

执行摘要

- 一句话：将 EAGLE verify 方法移出 dataclass 变为独立函数
- 推荐动作：值得精读的设计决策：将 stateless 行为从数据类中分离，符合“组合优于继承”原则，降低了数据类的耦合，提升了可测试性。建议类似模块也采用此模式。

功能与动机

PR body 指出：验证 batch-prep 和树采样是 stateless 操作，不应由 dataclass 携带；将这些行为移到 eagle_utils（已有的 tree-verify 内核所在模块）更合适。

实现拆解

1. 在 eagle_utils.py 中新增 eagle_prepare_for_verify 和 eagle_sample 两个自由函数，其函数体直接来自 EagleVerifyInputV2Mixin 的同名方法，仅将 self 改为 verify_input 参数。
2. 删除 eagle_info_v2.py 中的 EagleVerifyInputV2Mixin 类，清理不再需要的导入（torch.nn.functional、distributed、dp_attention、logits_processor、sampling/penaltylib、server_args、async_probe 等）和全局 is* 标志。
3. 将 EagleVerifyInputV2Mixin 原有的 max_tree_depth 和 tree_topk 属性直接移到 EagleVerifyInput 类（eagle_info.py）中作为属性；NgramVerifyInput 不再继承该 mixin，改为直接仅继承 EagleDraftInputV2Mixin。
4. 更新所有调用者：eagle_worker_v2.py、multi_layer_eagle_worker_v2.py 将 verify_input.prepare_for_verify(...) 和 verify_input.sample(...) 替换为 eagle_prepare_for_verify(...) 和 eagle_sample(...) 的显式调用。
5. 调整 ngram_worker.py 的导入，使其从 eagle_utils 中导入所需符号（build_tree_kernel_efficient 已在其中）。

关键文件：

- python/sglang/srt/speculative/eagle_info_v2.py（模块 推测解码；类别 source；类型 core-logic；符号 EagleVerifyInputV2Mixin, max_tree_depth, tree_topk, prepare_for_verify）：核心变更文件：移除了 EagleVerifyInputV2Mixin 整个类，清理了大量无关导入和全局变量。
- 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_verify, eagle_sample）：新增两个核心函数 eagle_prepare_for_verify 和 eagle_sample，接收 verify_input 作为参数，承载原 mixin

的逻辑。

- python/sglang/srt/speculative/eagle_info.py（模块 推测解码；类别 source；类型 core-logic；符号 EagleVerifyInput, max_tree_depth, tree_topk）：EagleVerifyInput 现在直接声明 max_tree_depth 和 tree_topk 属性，不再继承 mixin。
- python/sglang/srt/speculative/ngram_info.py（模块 推测解码；类别 source；类型 core-logic；符号 NgramVerifyInput）：NgramVerifyInput 不再继承 EagleVerifyInputV2Mixin，只继承 EagleDraftInputV2Mixin。

关键符号：eagle_prepare_for_verify, eagle_sample, EagleVerifyInput.max_tree_depth, EagleVerifyInput.tree_topk

关键源码片段

python/sglang/srt/speculative/eagle_info_v2.py

核心变更文件：移除了 EagleVerifyInputV2Mixin 整个类，清理了大量无关导入和全局变量。

```
# eagle_info_v2.py after refactor
# EagleVerifyInputV2Mixin 已被删除，其方法已迁移至 eagle_utils 和 eagle_info。
# 文件现在只包含 EagleDraftInputV2Mixin。

from __future__ import annotations

from dataclasses import dataclass
from typing import TYPE_CHECKING

import torch

from sglang.srt.managers.schedule_batch import ScheduleBatch
from sglang.srt.mem_cache.common import (
    alloc_paged_token_slots_extend,
    alloc_token_slots,
    get_alloc_reserve_per_decode,
    get_last_loc,
)
from sglang.srt.speculative.triton_ops.cache_locs import (
    assign_extend_cache_locs_func as assign_extend_cache_locs_func,
)
from sglang.srt.speculative.triton_ops.eagle import (
    fill_bonus_tokens as fill_bonus_tokens,
)

if TYPE_CHECKING:
    from sglang.srt.speculative.eagle_info import (
        EagleDraftInput,
    )

@dataclass
class EagleDraftInputV2Mixin:
```

```
    def prepare_for_decode(self: EagleDraftInput, batch: ScheduleBatch):
        # 原方法未变，省略实现
        ...
```

python/sglang/srt/speculative/eagle_utils.py

新增两个核心函数 eagle_prepare_for_verify 和 eagle_sample，接收 verify_input 作为参数，承载原 mixin 的逻辑。

```
# eagle_utils.py 新增的验证准备与采样函数
# 行为与原先 EagleVerifyInputV2Mixin 中的方法完全相同，但作为独立函数存在

def eagle_prepare_for_verify(
    verify_input: EagleVerifyInput,
    req_to_token_pool: ReqToTokenPool,
    batch: ScheduleBatch,
    target_worker: TpModelWorker,
):
    from sglang.srt.model_executor.forward_batch_info import (
        CaptureHiddenMode,
        ForwardBatch,
        ForwardMode,
    )
    from sglang.srt.speculative.spec_utils import prepare_mamba_track_for_verify
    from sglang.srt.speculative.triton_ops.cache_locs import (
        assign_extend_cache_locs_func,
    )

    if not batch.forward_mode.is_idle():
        # Assign cache locations
        bs = len(batch.req_pool_indices)
        batch.input_ids = verify_input.draft_token
        maybe_detect_oob(
            batch.input_ids,
            0,
            batch.model_config.vocab_size,
            "v2 prepare_for_verify input_ids",
        )
        device = batch.device
        batch.out_cache_loc = assign_extend_cache_locs_func(
            req_pool_indices=batch.req_pool_indices,
            req_to_token=req_to_token_pool.req_to_token,
            start_offset=batch.seq_lens,
            end_offset=batch.seq_lens + verify_input.draft_token_num,
            batch_size=bs,
            draft_token_num=verify_input.draft_token_num,
            device=device,
        )

        prepare_mamba_track_for_verify(batch)
```

```
            # TBO's split_spec_info reads these; no-verify-sync leaves both None.
            verify_input.seq_lens_cpu = batch.seq_lens_cpu
            verify_input.seq_lens_sum = (
                int(batch.seq_lens_cpu.sum()) if batch.seq_lens_cpu is not None else None
            )

        # Get a forward batch
        batch.forward_mode = (
            ForwardMode.IDLE if batch.forward_mode.is_idle() else ForwardMode.TARGET_VERIFY
        )
        capture_mode = (
            CaptureHiddenMode.NULL
            if target_worker.model_runner.spec_algorithm.is_standalone()
            else CaptureHiddenMode.FULL
        )
        batch.capture_hidden_mode = capture_mode
        verify_forward_batch = ForwardBatch.init_new(batch, target_worker.model_runner)

        # Run attention backend plan and cuda graph preparation
        can_run_cuda_graph = bool(
            target_worker.model_runner.decode_cuda_graph_runner
            and target_worker.model_runner.decode_cuda_graph_runner.can_run(
                verify_forward_batch
            )
        )
        if can_run_cuda_graph:
            target_worker.model_runner.decode_cuda_graph_runner.replay_prepare(
                verify_forward_batch
            )
            verify_forward_batch.mark_forward_metadata_ready()
        # Non-cuda-graph: defer init to forward_extend, which runs after
        # `_forward_raw -> prepare_mlp_sync_batch` pads the batch.

        # Return forward batch and cuda graph flag
        return verify_forward_batch, can_run_cuda_graph


    def eagle_sample(
        verify_input: EagleVerifyInput,
        batch: ScheduleBatch,
        logits_output: LogitsProcessorOutput,
        vocab_mask: torch.Tensor = None,
    ):
        # 从原 mixin 的 sample 方法直接迁移，省略实现以保持简洁
        ...
```

评论区精华

PR 无实质 review 讨论。PR 作者在 Issue 评论区中触发了多次 CI rerun，相关测试全部通过，确认重构无回归。

- CI 测试结果 (testing): 重构无功能回归，CI 通过。

风险与影响

- 风险：本次是纯代码移动重构，行为等价，回归风险低。但 eagle_utils.py 新增了自由函数，若未来有其他模块直接调用 EagleVerifyInputV2Mixin 的方法可能会遗漏。由于 mixin 已被删除，若存在外部依赖将编译失败（显式错误），而非静默行为错误，风险可控。
- 影响：内部 API 重构：EAGLE verify 相关行为不再通过 mixin 继承，而是通过 eagle_utils 的函数调用。对系统外部无影响。影响范围限于 speculative decoding 模块的 EAGLE 系列（包括 multi-layer）和 ngram，DFLASH 不受影响。团队后续需注意任何自定义 worker 若继承了 EagleVerifyInputV2Mixin 也需要同步更新。
- 风险标记：潜在外部依赖遗漏，核心路径变更

关联脉络

- PR #28093 [Spec] Move draft-extend prep to EagleDraftWorkerBase; unify prepare_for_* names: 同属 speculative decoding 重构系列，将方法从数据类移至基类或工具函数，本 PR 延续了相同思路。
- PR #28105 [Spec] Move prepare_for_draft to EagleDraftWorkerBase: 同样是将 prepare_for_draft 从 mixin 移到 worker 基类，本 PR 是类似的移动重构。