PR #27799 完整报告

sgl-project/sglang

[Spec] `NGRAMWorker` on `BaseSpecWorker`; algo-owned verify-tree shape params

合并时间：2026-06-11 03:38

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27799

执行摘要

- 一句话：重构 NGRAMWorker，统一推测解码树形参数接口
- 推荐动作：
  - 优先修复 critical 问题：在 common.py 中对 server_args.speculative_algorithm 做 None 检查，或给 SpeculativeAlgorithm.from_string 增加 None 安全处理。
  - 值得学习的模式：将树形参数提升到 verify-input 抽象上，通过多态消除条件分支，是一个很好的接口设计范例，适合在其他需要算法多态的场景中借鉴。
  - 补充测试：为 SpeculativeAlgorithm.has_draft_kv() 和 verify-input 属性编写简单单元测试，固化接口契约。

功能与动机

PR #27799 旨在将上一步 #17260 引入的 NGRAM 推测解码实现接入标准接口，并使其树形参数（验证树深度、分支因子）通过算法类型的属性暴露，从而让共享的 sample() 和 KV 预分配逻辑不再需要 is_ngram() 分支。这样不仅减少了代码死角，也为未来引入 dflash、Medusa 等树形算法铺平了道路。

实现拆解

步骤 1：让 NGRAMWorker 继承 BaseSpecWorker

在 ngram_worker.py 中，类声明改为 class NGRAMWorker(BaseSpecWorker)，新增 target_worker 属性和 draft_worker 属性，前者返回底层目标 worker，后者返回 None 表示无 draft 模型。同时调整导入语句，引入 BaseDraftWorker 和 BaseSpecWorker。

步骤 2：将验证树形状参数改为多态属性

在 eagle_info_v2.py 的 EagleVerifyInputV2Mixin 中新增 max_tree_depth 和 tree_topk 属性，默认值由 spec_steps 和 topk 计算；sample() 方法中直接使用 self.max_tree_depth 和 self.tree_topk 取代原先的 is_ngram() 条件判断。

在 ngram_info.py 的 NgramVerifyInput 中覆盖这两个属性：max_tree_depth 返回 draft_token_num（NGRAM 树受节点预算限制，单次匹配可能链满所有 token），tree_topk 返回 -1（无固定分支因子）。

步骤 3：引入算法级别的 has_draft_kv() 方法

在 spec_info.py 的 SpeculativeAlgorithm 枚举和 spec_registry.py 的基类中添加 has_draft_kv() 抽象。NGRAM 算法返回 False（其 draft 树只存在于验证掩码中，不占用实际 KV 缓存），其他算法返回 True。该值在 mem_cache/common.py 的 get_alloc_len_per_decode() 中使用，替代原有的字符串比较 is_ngram，决定是否启用 per-topk 页面复制预留。

步骤 4：简化 KV 预分配逻辑

在 common.py 中，将原来基于 server_args.speculative_algorithm.upper() == "NGRAM" 的条件改为使用 spec_algo.has_draft_kv()，使逻辑更健壮且算法无关。

步骤 5：无测试新增，但现有套件覆盖正常路径

本次 PR 未添加单元测试，但 test_spec_ngram.py 和 test_spec_eagle_parity.py 等集成测试在 CI 中通过，保证了基本功能不退化。

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 max_tree_depth, tree_topk）：核心变更文件：新增 max_tree_depth/tree_topk 属性到 EagleVerifyInputV2Mixin，消除 sample() 中对 is_ngram() 的依赖
- python/sglang/srt/speculative/ngram_worker.py（模块 推测解码；类别 source；类型 core-logic；符号 NGRAMWorker, target_worker, draft_worker）：让 NGRAMWorker 继承 BaseSpecWorker，并实现 target_worker/draft_worker 接口属性
- python/sglang/srt/speculative/ngram_info.py（模块 推测解码；类别 source；类型 core-logic；符号 max_tree_depth, tree_topk）：覆盖 max_tree_depth/tree_topk 属性以提供 NGRAM 特有的树形状
- python/sglang/srt/mem_cache/common.py（模块 推测解码；类别 source；类型 dependency-wiring）：将 KV 预分配逻辑中的 is_ngram 字符串比较替换为 spec_algo.has_draft_kv() 多态调用
- python/sglang/srt/speculative/spec_info.py（模块 推测解码；类别 source；类型 core-logic；符号 has_draft_kv）：在 SpeculativeAlgorithm 枚举中添加 has_draft_kv() 方法，为多态分配提供基础
- python/sglang/srt/speculative/spec_registry.py（模块 推测解码；类别 source；类型 core-logic；符号 has_draft_kv）：在 SpeculativeAlgorithmRegistry 基类中添加 has_draft_kv() 默认实现（返回 True），确保其他算法注册时不会缺失该方法

关键符号：NGRAMWorker.init, NGRAMWorker.target_worker, NGRAMWorker.draft_worker, EagleVerifyInputV2Mixin.max_tree_depth, EagleVerifyInputV2Mixin.tree_topk, NgramVerifyInput.max_tree_depth, NgramVerifyInput.tree_topk, SpeculativeAlgorithm.has_draft_kv, SpeculativeAlgorithmRegistry.has_draft_kv

关键源码片段

python/sglang/srt/speculative/eagle_info_v2.py

核心变更文件：新增 max_tree_depth/tree_topk 属性到 EagleVerifyInputV2Mixin，消除 sample() 中对 is_ngram() 的依赖

```
@dataclass
class EagleVerifyInputV2Mixin:
    @property
    def max_tree_depth(self: EagleVerifyInput) -> int:
        """最长根到叶的验证树链，包含根节点；用于确定 accept_index 行宽度。
        EAGLE 树受 draft 循环深度限制，spec_steps + 1。"""
        return self.spec_steps + 1

    @property
    def tree_topk(self: EagleVerifyInput) -> int:
        """传递给树验证核的分支因子；-1 表示不规则树。"""
        return self.topk

    def sample(self, ...):
        # ...
        candidates = self.draft_token.reshape(bs, self.draft_token_num)
        predict_shape = list(next_token_logits.shape)[:-1]
        predict = torch.zeros(predict_shape, dtype=torch.int32, device=device).flatten()
        # 使用 self.max_tree_depth 替代条件分支
        accept_index = torch.full(
            (bs, self.max_tree_depth), -1, dtype=torch.int32, device=device
        )
        # ...
        if tree_mask_method == "custom_mask_tree":
            # ...
            topk = self.tree_topk
            # ...
        # ...
```

python/sglang/srt/speculative/ngram_worker.py

让 NGRAMWorker 继承 BaseSpecWorker，并实现 target_worker/draft_worker 接口属性

```
class NGRAMWorker(BaseSpecWorker):
    def __init__(self, server_args, gpu_id, tp_rank, dp_rank,
                 moe_ep_rank, attn_cp_rank, moe_dp_rank, nccl_port,
                 target_worker):
        # ...
        self._target_worker = target_worker # 改为 private 属性
        # ...

    @property
    def target_worker(self) -> TpModelWorker:
        return self._target_worker

    @property
    def draft_worker(self) -> Optional[BaseDraftWorker]:
```

```
        # NGRAM 没有 draft 模型；draft 来自 CPU 端语料库
        return None
```

python/sglang/srt/speculative/ngram_info.py

覆盖 max_tree_depth/tree_topk 属性以提供 NGRAM 特有的树形状

```
@dataclass
class NgramVerifyInput(SpecVerInput):
    # ...

    @property
    def max_tree_depth(self) -> int:
        # NGRAM 树受节点预算限制没有深度上限：语料库 BFS 只会在节点预算
        # 用尽时停止，因此单次长匹配可以链接所有 draft_token_num 个节点
        # （spec_steps 对此树无意义）。
        return self.draft_token_num

    @property
    def tree_topk(self) -> int:
        # 不规则树：每层分支跟随语料库匹配结果。
        return -1
```

评论区精华

gemini-code-assist[bot] 提出了三个评审意见，均未在合并前采纳：

- Critical — from_string(None) 崩溃风险：当 server_args.speculative_algorithm 为 None 时，get_alloc_len_per_decode 中的 SpeculativeAlgorithm.from_string(None) 会抛出 AttributeError，导致任何非推测解码请求都崩溃。建议调用前判断 None。
- Medium — GPU 循环创建 tensor 的性能问题：在 ngram_worker.py 的一次循环中，逐请求创建 GPU tensor 并 torch.cat，可能引入额外延迟，建议改用 CPU numpy 构建后统一传输。
- Medium — 除零错误风险：在 speculative_hook.py 的 NGRAM 初始化中，speculative_num_draft_tokens // speculative_eagle_topk 可能因 topk 为 0 或 None 而异常，建议添加安全 fallback。
- 未保护的 from_string 调用导致正常解码崩溃 (correctness): 作者未在 PR 中回复或修复该问题，合并后代码仍带有此 bug。
- GPU tensor 循环创建性能问题 (performance): 未采纳，作者未回应。
- 除零错误风险 (correctness): 未修复，合并后代码仍存在此风险。

风险与影响

- 风险：核心路径崩溃风险（Critical）：如 review 指出的 from_string(None) 调用未加保护，一旦用户不使用推测解码，每个 decode 步都会触发异常，完全阻塞正常推理。该项目目前已合并该 PR，此问题仍存在，属于严重回归。

缺少测试覆盖：本次重构涉及接口契约变化，但未针对新属性或 has_draft_kv() 编写单元测试，回归依赖已有的 test_spec_ngram.py 等，若未来有其他算法接入可能漏测。

潜在性能回退：review 指出 ngram_worker.py 中的 batch 循环内创建 GPU tensor，可能在大 batch 场景下引入不可忽视的开销，但官方未做 benchmark。

- 影响：内部 API 影响：NGRAMWorker 现在必须遵守 BaseSpecWorker 接口，若外部有直接使用 NGRAMWorker.target_worker（现在改为 _target_worker 后通过 property 暴露）的代码，需要进行适配。但调度器已通过接口使用，影响可控。

用户影响：无直接可见变更，推测解码行为不变。

系统影响：新的多态抽象使添加新算法（如 dflash、Medusa）时只需提供对应 verify-input 类型并实现 max_tree_depth/tree_topk/has_draft_kv，无需修改共享 sample() 和内存分配逻辑，降低耦合。

- 风险标记：核心路径崩溃风险，缺少测试覆盖，GPU 性能回退可能

关联脉络

- PR #17260 [Feature] [Ngram spec] Support ngram spec v2: 此 PR 是对 #17260 的后续改进，统一 NGRAMWorker 接口并消除 is_ngram() 分支，使 NGRAM 完全融入 v2 推测解码框架。