

PR #55133 完整报告

vllm-project/vllm

[Spec Decode] Fix Qwen3 DSpark d2t requirement for padded-vocab drafts

合并时间: 2026-09-14 22:07

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/55133>

执行摘要

- 一句话: DSpark 补齐词表草稿不再误报缺 d2t
- 推荐动作: 值得精读, 但只需读源码两处条件与新增测试: 这是一个「一行判据改对语义」的典型案列, 展示了对 `!=` 与 `<` 在模型加载契约中差异的精确把握, 也示范了把「加载期校验」与「构造期参数创建」绑定为同一不变式的做法。阅读时重点看三件事: (1) `__init__` 与 `load_weights` 的判据为何必须同步、不同步会怎样; (2) 恒等映射路径 `map_draft_to_target` 是否真的等价于过去的全零 d2t; (3) 务必跟进 CodeRabbit 提出的 padded 行 mask 建议——建议维护者为 `compute_draft_logits / DSparkSpeculator._sample_logits` 另开 issue 或小 PR, 补上 `[target_vocab_size, draft_vocab_size)` 的 mask 与「高 padded logit」回归测试, 否则本 PR 只是把失败从加载期推迟到采样期。

功能与动机

PR 正文说明: Qwen3DSparkForCausalLM 用 `uses_reduced_vocab = self.config.draft_vocab_size != self.target_vocab_size` 判定草稿是否为精简词表, 从而要求 `lm_head + d2t`。但草稿词表可以在「补齐到物理 embedding 尺寸」后大于目标的逻辑 `vocab_size`: Inkling-Small-NVFP4 的 `get_vocab_size()` 为 200058, 而 `embed/unembed` 是 `[201024, 4096]`、`draft_vocab_size` 为 201024, 这类草稿覆盖完整词表、映射是恒等的, 却在加载时报出“Reduced-vocabulary Qwen3 DSpark checkpoints must include a d2t mapping”。该检查由 @Zhou248 在 #52560 中加入 (正文中 cc), 本 PR 只把 d2t 门控从 `!=` 改为 `<`, 使剪枝草稿行为不变、补齐草稿不再误报。

实现拆解

1. 构造期判据收紧 (`vllm/model_executor/models/qwen3_dspark.py::Qwen3DSparkForCausalLM.__init__`): 把 `if self.config.draft_vocab_size != self.target_vocab_size:` 改为 `<`, 并补充注释「只有 pruned draft 才需要 d2t 把 id 散射上去, `draft >= target` 的补齐草稿是恒等映射」。结果是 `draft_vocab_size >= target_vocab_size` 时 `self.draft_id_to_target_id = None`, 不再凭空创建一个长度为 `draft_vocab_size` 的 long 参数 (该参数在补齐检查点里本来就不存在)。
2. 加载期判据与构造期对齐 (同文件 `load_weights`): `uses_reduced_vocab = self.config.draft_vocab_size < self.target_vocab_size`。这个布尔量同时驱动三条后续逻辑——缺 `lm_head` 的报错、缺 d2t 的报错, 以及 `orig_to_new_substr["draft_id_to_target`

`_id"] = None` 的权重名过滤。改完后形成不变式：只有在 `__init__` 真的建了 `draft_id_to_target_id` 时才要求 checkpoint 带 `d2t`，两处判据不再可能出现「构造期不建参数、加载期却强制要权重」的自相矛盾。相邻的 `uses_expanded_input_vocab` (`config.vocab_size > target_vocab_size`) 分支保持原样，输入词表扩展仍需自带 `embed_tokens`。

3. 运行期映射退化为恒等：`map_draft_to_target` 在 `self.draft_id_to_target_id is None` 时直接 `return draft_ids`，等价于过去要求的「全零 `d2t`」(`draft_ids + 0`)，因此采样结果语义不变，只是少一次 `gather`。PR 正文也据此论证 `id` 仍落在目标物理 `embedding` 范围内，不会越界。
4. 测试配套 (`tests/model_executor/test_qwen3_omni.py`, +33/-13)：新增 `_dspark_vocab_stub(input_vocab_size, draft_vocab_size)` 桩函数，集中构造「只带 `load_weights` 所需字段」的 `DSpark` 实例 (`config`、`target_vocab_size=100`、`model=Mock(confidence_head=None)`)，并让原有 `test_qwen3_dspark_rejects_incomplete_vocab_weights` 复用它；新增参数化用例 `test_qwen3_dspark_accepts_draft_vocab_at_least_target[100, 101]`，直接 `model.load_weights([])` 断言不抛异常。其中 `draft=101 > target=100` 的用例在旧 `!=` 逻辑下必然失败，是本次修复的精确回归护栏；`(100, 40, [lm_head], "must include a d2t mapping")` 等剪枝场景仍保持报错，确保放宽没有波及真正的 `reduced-vocab` 路径。
5. 无配置 / `schema` / 部署改动：不涉及 CLI、环境变量、KV cache 或权重格式，加载器以外无连带修改；测试通过 `@pytest.mark.skip_global_cleanup` 与既有用例保持一致。

关键文件：

- `vllm/model_executor/models/qwen3_dspark.py` (模块 草稿模型；类别 `source`；类型 `data-contract`；符号 `Qwen3DSparkForCausalLM.init`, `Qwen3DSparkForCausalLM.load_weights`, `Qwen3DSparkForCausalLM.map_draft_to_target`, `Qwen3DSparkForCausalLM.compute_draft_logits`)：本 PR 的唯一源码改动点：`Qwen3DSparkForCausalLM.__init__` 决定是否创建 `draft_id_to_target_id` 参数，`load_weights` 决定是否强制 checkpoint 带 `lm_head + d2t`，两处判据同时从 `!=` 收紧为 `<`，直接决定补齐词表草稿能否加载。
- `tests/model_executor/test_qwen3_omni.py` (模块 模型测试；类别 `test`；类型 `test-coverage`；符号 `_dspark_vocab_stub`, `test_qwen3_dspark_accepts_draft_vocab_at_least_target`, `test_qwen3_dspark_rejects_incomplete_vocab_weights`)：测试配套：抽出可复用的 `DSpark` 加载桩 `_dspark_vocab_stub`，并新增参数化用例覆盖 `draft_vocab_size == 100` 与 `> 100` (补齐场景) 时必须加载成功，同时保留剪枝场景仍报错的既有用例。

关键符号：`Qwen3DSparkForCausalLM.init`, `Qwen3DSparkForCausalLM.load_weights`, `Qwen3DSparkForCausalLM.map_draft_to_target`, `Qwen3DSparkForCausalLM.compute_draft_logits`, `_dspark_vocab_stub`, `test_qwen3_dspark_accepts_draft_vocab_at_least_target`, `test_qwen3_dspark_rejects_incomplete_vocab_weights`

关键源码片段

vllm/model_executor/models/qwen3_dspark.py

本 PR 的唯一源码改动点: `Qwen3DSparkForCausalLM.__init__` 决定是否创建 `draft_id_to_target_id` 参数, `load_weights` 决定是否强制 checkpoint 带 `lm_head + d2t`, 两处判据同时从 `!=` 收紧为 `<`, 直接决定补齐词表草稿能否加载。

```
class Qwen3DSparkForCausalLM(DFlashQwen3ForCausalLM):
    def __init__(self, *, vllm_config: VllmConfig, prefix: str = "") -> None:
        nn.Module.__init__(self)
        self.draft_model_config = vllm_config.speculative_config.draft_model_config
        self.config = self.draft_model_config.hf_config
        # config 未声明 draft_vocab_size 时回退到 vocab_size, 保证下面的判据恒有值
        if getattr(self.config, "draft_vocab_size", None) is None:
            self.config.draft_vocab_size = getattr(self.config, "vocab_size", None)
        # ... 这里构造 Qwen3DSparkModel / ParallelLMHead / LogitsProcessor ...

        self.target_vocab_size = vllm_config.model_config.get_vocab_size()
        # 【本 PR 核心 1】判据由 `!=` 收紧为 `<`：只有真正「剪枝过」的草稿
        # (draft_vocab_size 是目标词表的子集) 才需要 d2t 偏移表把草稿 id 散射到
        # 目标空间; draft_vocab_size >= target_vocab_size (典型是补齐到目标物理
        # embedding 尺寸, 如 Inkling-Small-NVFP4: 逻辑 200058 / 物理 201024)
        # 覆盖完整词表, 二者是恒等映射, 因此既不建参数也不该要求 d2t 权重。
        if self.config.draft_vocab_size < self.target_vocab_size:
            self.draft_id_to_target_id = nn.Parameter(
                torch.zeros(self.config.draft_vocab_size, dtype=torch.long),
                requires_grad=False,
            )
        else:
            self.draft_id_to_target_id = None # 恒等映射路径, 不再占用参数

    def map_draft_to_target(self, draft_ids: torch.Tensor) -> torch.Tensor:
        # 恒等分支等价于过去「全零 d2t」的加法 (draft_ids + 0), 但省掉一次 gather,
        # 也让 padded 草稿在采样后不需要任何 id 变换。
        if self.draft_id_to_target_id is None:
            return draft_ids
        return draft_ids + self.draft_id_to_target_id[draft_ids]

    def load_weights(self, weights: Iterable[tuple[str, torch.Tensor]]):
        model_weights = {}
        includes_embed_tokens = False
        includes_lm_head = False
        includes_draft_id_mapping = False
        includes_confidence_head = False
        for name, loaded_weight in weights:
            # t2d 只在训练期使用; 推理期由 d2t 在采样后把草稿 id 映射回目标 id
            if "t2d" in name:
                continue
            if "d2t" in name:
                name = name.replace("d2t", "draft_id_to_target_id")
                includes_draft_id_mapping = True
```

```

elif "lm_head" not in name:
    name = "model." + name
if "embed_tokens" in name:
    includes_embed_tokens = True
if "lm_head" in name:
    includes_lm_head = True
if "confidence_head" in name:
    includes_confidence_head = True
model_weights[name] = loaded_weight
# 设置 has_own_embed_tokens / has_own_lm_head, 供 load_dspark_model
# 决定是保留草稿自身权重, 还是别名到目标模型的同名权重
process_eagle_weight(self, name)

uses_expanded_input_vocab = self.config.vocab_size > self.target_vocab_size
# 【本 PR 核心 2】与 __init__ 使用同一判据: 只有 draft < target 才算
# reduced vocab。改前用 `!=`, 导致补齐词表的草稿被误判为精简词表,
# 加载时抛出 "must include a d2t mapping", 尽管它根本不需要 d2t。
uses_reduced_vocab = self.config.draft_vocab_size < self.target_vocab_size

if uses_expanded_input_vocab and not includes_embed_tokens:
    raise ValueError(
        "Qwen3 DSpark checkpoints whose input vocab_size is larger than "
        "the target vocabulary must include embed_tokens weights."
    )
if uses_reduced_vocab and not includes_lm_head:
    # 目标 lm_head 覆盖不了被剪掉的草稿词表, 无法共享, 必须自带
    raise ValueError(
        "Reduced-vocabulary Qwen3 DSpark checkpoints must include "
        "lm_head weights; the full target lm_head cannot be shared."
    )
if uses_reduced_vocab and not includes_draft_id_mapping:
    # 补齐词表的草稿现在不再走到这里 (draft >= target 直接跳过),
    # 因此 Inkling-Small-NVFP4 这类检查点可以正常加载。
    raise ValueError(
        "Reduced-vocabulary Qwen3 DSpark checkpoints must include a "
        "d2t mapping so sampled draft ids can be converted to target ids."
    )

# 权重名过滤: 缺失的模块统一置 None, 交给 AutoWeightsLoader 跳过;
# padded 草稿没有 d2t 权重, 这里同时剔除 draft_id_to_target_id,
# 与 __init__ 中「没有创建该参数」保持一致, 避免加载器报缺参数。
orig_to_new_substr = {"mask_embedding": None}
if not includes_embed_tokens:
    orig_to_new_substr["embed_tokens"] = None
if not includes_lm_head:
    orig_to_new_substr["lm_head"] = None
if not includes_draft_id_mapping:
    orig_to_new_substr["draft_id_to_target_id"] = None
if self.model.confidence_head is None or not includes_confidence_head:

```

```

self.model.confidence_head = None
orig_to_new_substr["confidence_head"] = None
mapper = WeightsMapper(orig_to_new_substr=orig_to_new_substr)
loader = AutoWeightsLoader(self)
loader.load_weights(model_weights.items(), mapper=mapper)
self.model._build_fused_kv_buffers()

```

tests/model_executor/test_qwen3_omni.py

测试配套：抽出可复用的 DSpark 加载桩 `_dspark_vocab_stub`，并新增参数化用例覆盖 `draft_vocab_size == 100` 与 `> 100`（补齐场景）时必须加载成功，同时保留剪枝场景仍报错的既有用例。

```

def _dspark_vocab_stub(input_vocab_size: int, draft_vocab_size: int):
    """A Qwen3 DSpark model with only what `load_weights` reads."""
    from vllm.model_executor.models.qwen3_dspark import Qwen3DSparkForCausalLM

    # 绕过 __init__ 直接造实例，只注入 load_weights 会读取的字段，
    # 因此测试完全聚焦在加载期判据上，不需要真实权重与 GPU。
    model = Qwen3DSparkForCausalLM.__new__(Qwen3DSparkForCausalLM)
    nn.Module.__init__(model)
    object.__setattr__(
        model,
        "config",
        SimpleNamespace(
            vocab_size=input_vocab_size,
            draft_vocab_size=draft_vocab_size,
        ),
    )
    object.__setattr__(model, "target_vocab_size", 100) # 目标词表固定为 100
    # confidence_head=None 让 load_weights 走「丢弃 confidence_head」的既有分支
    object.__setattr__(model, "model", Mock(confidence_head=None))
    return model


@pytest.mark.skip_global_cleanup
@pytest.mark.parametrize("draft_vocab_size", [100, 101])
def test_qwen3_dspark_accepts_draft_vocab_at_least_target(draft_vocab_size):
    """A draft vocab >= the target's maps identically, so needs no lm_head/d2t.

    Guards the padded case (draft > target): the draft's unembedding is padded
    to its physical size, which is not a reduced vocabulary.
    """
    model = _dspark_vocab_stub(100, draft_vocab_size)

    # 空权重集合 + 不抛异常：draft_vocab_size=101 这一档在旧的 `!=` 判据下
    # 会抛 "must include a d2t mapping"，是本 PR 修复的精确回归护栏。
    model.load_weights([])

```

评论区精华

评审总体轻量：维护者 benchislett 只留下两条风格意见——先是一句空的 “[suggestion](#)”，随后回复自己的评论 “one comment is enough”，指向 [load_weights](#) 中重复的说明注释；作者以提交 [a634ab91](#) (Co-authored-by: Benjamin Chislett) 应用建议，最终 benchislett 以 “LGTM” APPROVED。真正有技术含量的是 coderabbitai 的 Major 发现（未解决）：它挂在新条件行 `if self.config.draft_vocab_size < self.target_vocab_size:` 上，要求 `compute_draft_logits` 与 `DSparkSpeculator._sample_logits` 的采样路径在 dense sampling 和 topk 之前，把 `target_vocab_size` 到 `draft_vocab_size` 之间的补齐行 mask 掉，理由是这些 padded 行可能产出非法目标 id，并建议补一条「高 padded 行 logit」的回归测试。该线程在本 PR 中没有得到回复或修复。另外 claude[bot] 因 PR 来自 fork 而跳过了自动评审，所以核心逻辑的第三方复核主要依赖 CodeRabbit 这条意见。

- reduced-vocab 判据用 `<` 还是 `!=`：补齐词表草稿的加载契约 (correctness): 接受改为 `<`，且 `__init__` 与 `load_weights` 两处同步修改，使「构造期建参数」与「加载期要求权重」保持同一不变式；剪枝路径行为不变。
- 重复注释清理 (style): 采纳建议，注释只保留在 `__init__` 一处；benchislett 随后 APPROVED (“LGTM”)。
- 补齐词表行的 logit 未 mask，采样可能产出非法目标 id (correctness): 未解决，遗留为后续工作项；需要单独 PR 在采样前 mask padded 行并补回归测试。

风险与影响

- 风险：
 1. 采样越界（未解决，最高优先级）：判据放宽后，补齐草稿的 `compute_draft_logits` 仍对整个 `draft_vocab_size` 做 dense softmax / topk, [`target_vocab_size`, `draft_vocab_size`) 这些 padded 行没有任何 mask (logit 通常为 0 或未定义)，一旦它们排进 topk，采样出的 draft id 会大于等于目标逻辑词表；而 `map_draft_to_target` 此时是恒等映射，`draft_ids + None` 的旧 d2t 约束消失后就没有任何环节拦截这些 id，理论上会在 [200058, 201024) 这段「物理存在但逻辑非法」的区间产生 token。PR 正文只论证了不越出物理 embedding、不触发 OOB，未论证不会产出非法 token id。CodeRabbit 明确要求补 mask 与回归测试，本 PR 未处理。
 2. 双处判据必须同步：`__init__` 与 `load_weights` 各写一遍 `draft_vocab_size < target_vocab_size`，二者一旦分叉就会出现「无参数却强制加载权重」或反之的错配；当前靠注释和测试对齐，缺少一个共享属性（如 `uses_reduced_vocab` property）来固化。
 3. 测试覆盖不完整：新增用例只覆盖 `load_weights` 的放行分支，`__init__` 中 `draft_id_to_target_id` is None 的构造路径、`map_draft_to_target` 的恒等返回都没有单测；PR 正文提到的端到端验证 (Inkling DSpark drafter 解码连贯) 是人工且不可复现的。
 4. 兼容性方向是放宽而非收紧：`draft == target` 行为不变，`draft > target` 从「无法加载」变为「可加载」，不会破坏已有可加载的 checkpoint，回归风险集中在上面第 1 点的运行期语义。- 影响：对用户：使用补齐词表 DSpark 草稿（如 Inkling-Small-NVFP4）的

投机解码部署从「启动即报错」变为「可正常加载并解码」，属于直接解锁能力；使用真正剪枝草稿（draft < target）的用户完全无感，报错与 d2t 要求保持不变。对系统：影响面局限在 Qwen3DSparkForCausalLM 的权重加载与 id 映射契约，不触及调度、KV cache、注意力后端或前端；load_dspark_model 侧的 has_own_embed_tokens / has_own_lm_head 共享逻辑不受影响，仅是少了一个永远不存在的 draft_id_to_target_id 参数。对团队：改动只有 2 个文件、43 行，评审成本低，但它把「逻辑词表 vs 物理补齐词表」这一 DSpark / Eagle 系草稿模型的通用概念显性化，后续接入其他厂商补齐词表的草稿模型（含 MoE / Mtp 系）时会反复用到同一判据。剩余风险是采样阶段未做 padded 行 mask，如果真实模型的 padded 行 logit 不可忽略，用户体验会从「加载失败」变为「偶发非法 token」，定位成本更高。- 风险标记：采样可能产出非法 token id, 缺少 padded 行 mask 回归测试，构造期与加载期双处判据需同步，仅人工端到端验证

关联脉络

- PR #52560 [引用自 PR 正文] 为 Qwen3 DSpark reduced-vocab 检查点加入 lm_head + d2t 校验：PR 正文明确 cc @Zhou248 并注明“(added the check in #52560)”，即本 PR 放宽的那条 uses_reduced_vocab 校验的来源；标题未在上下文中提供，此处仅为占位说明。
- PR #56791 [Bugfix] Trim stale consequence claims from unannotated-eagle warning: 同属投机解码中「草稿词表与目标词表不一致」时的提示 / 校验语义处理（Eagle 草稿 + kv_cache_utils 词表告警），与本 PR 一样是在修正校验条件的措辞与判定边界。
- PR #50894 [Bugfix] Scale KV page size for hidden states extraction with TP: 同样是投机解码草稿模型与目标模型形状 / 词表不匹配导致的启动期失败类问题，说明「草稿配置与目标配置对齐」是 spec-decode 的高频缺陷区域。