

PR #46995 完整报告

vllm-project/vllm

[Spec Decode] DSpark

合并时间: 2026-07-01 23:30

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/46995>

执行摘要

- 一句话: 实现 DeepSeek-V4 / Qwen3 DSpark 半自回归推测解码
- 推荐动作: 值得精读。设计巧妙: 利用现有稀疏 MLA 索引实现非因果注意力, 避免重写内核; CUDA Graphs 完整捕获整个 draft 步骤; 测试覆盖齐全。关键决策点: 锚点作为第一个预测、马尔可夫头低秩设计、indexer 线程阈值的 2x 乘数。建议重点关注 DSparkSpeculator._sample_sequential 和 sparse_swa.py 中的非因果索引构建逻辑。

功能与动机

添加对 DeepSeek-V4-Pro-DSpark 和 dspark_qwen3_8b_block7 等 DSpark 系列模型的支持。DSpark 是一种半自回归推测解码方法, 通过非因果注意力一次并行生成多个 draft tokens, 再用轻量马尔可夫头注入顺序依赖, 相比 MTP 在更长推测长度下有更高的接受率 (实测 DL6 接受率提升 31%)。

实现拆解

1. 复用稀疏 MLA 索引实现非因果注意力: DSpark 的 draft token 块内部需要非因果注意力。利用现有稀疏 MLA 后端的索引驱动特性, 将每个 query 的 top-k 索引扩展到包含滑动窗口内所有上下文 token 以及当前块内所有 token (包括后续位置的 token), 从而无需重写 MLA 内核。
2. 模型定义: 新增 DSparkDeepseekV4Model (vllm/models/deepseek_v4/nvidia/dspark.py) 继承 nn.Module 并混入 EagleModelMixin, 复用 DeepseekV4DecoderLayer 作为主干, 通过 main_proj 将目标模型多个层的辅助隐藏状态融合, 并添加 hc_head 和 DSparkMarkovHead。Qwen3DSparkModel (vllm/model_executor/models/qwen3_dspark.py) 基于 DFlashQwen3Model 扩展, 新增 DSparkMarkovHead。DSparkMarkovHead 是低秩 $V \rightarrow r \rightarrow V$ 的过渡偏置。
3. 推测器流水线: DSparkSpeculator (vllm/v1/worker/gpu/spec_decode/dspark/speculator.py) 继承 DFlashSpeculator, 设置 self.num_query_per_req = num_speculative_steps (锚点作为第一个预测), 并重写 _sample_sequential 方法实现从左到右的顺序采样: 对每一步, 用前一步采样 token 的马尔可夫嵌入偏置当前步的 logits, 然后执行 Gumbel 采样。CUDA Graphs (FULL 模式) 完整捕获整个 draft 步骤。
4. 注意力后端扩展: 在 sparse_swa.py 中添加 _compute_dspark_noncausal_swa_indices_kernel Triton 内核, 根据 common_attn_metadata.causal 标志计算非因果索引。同时调整 decode_threshold 以容纳平行推理中更大的 query 长度 (1+2*num_spec_tokens)。

5. 配置与基础设施：在 `vllm/config/speculative.py` 中添加 `use_dspark()` 方法；在 `vllm/config/vllm.py` 中将 `dspark` 加入 v2 model runner 支持列表；新增 `load_dspark_model` 工具函数处理权重别名（如 fp4 vs fp8 scale 后缀差异）。
6. 测试覆盖：新增非因果注意力正确性测试（`test_dspark_noncausal_sparse_mla.py`）、端到端推测解码测试（`test_dspark_correctness_and_acceptance_rate`）和注册表导入测试。
7. 目标模型修改：调整 `DeepseekV4Model.forward` 以收集目标各层的辅助隐藏状态，通过 `EagleModelMixin` 和 `SupportsEagle3` 暴露给推测器。

关键文件：

- `vllm/models/deepseek_v4/nvidia/dspark.py`（模块 模型定义；类别 source；类型 data-contract；符号 `DSparkDeepseekV4Model`, `init`, `embed_input_ids`, `combine_hidden_states`）：新增 `DSparkDeepseekV4Model` 和 `DSparkDeepseekV4ForCausalLM`，定义 DSpark 草稿模型的核心结构，复用目标解码层并添加马尔可夫头。
- `vllm/model_executor/models/qwen3_dspark.py`（模块 模型定义；类别 source；类型 data-contract；符号 `DSparkMarkovHead`, `init`, `embed`, `bias`）：定义 `DSparkMarkovHead` 顺序过渡偏置头、`Qwen3DSparkModel` 和 `Qwen3DSparkForCausalLM`，是 Qwen3 架构 DSpark 的核心入口。
- `vllm/v1/worker/gpu/spec_decode/dspark/speculator.py`（模块 推测器；类别 source；类型 core-logic；符号 `DSparkSpeculator`, `init`, `load_draft_model`, `_sample_sequential`）：实现 `DSparkSpeculator`，继承 `DFlashSpeculator` 并重写顺序采样逻辑，是 DSpark 推测解码的核心流水线。
- `vllm/v1/attention/backends/mla/sparse_swa.py`（模块 注意力后端；类别 source；类型 core-logic；符号 `_compute_dspark_noncausal_swa_indices_kernel`）：修改稀疏滑动窗口注意力后端，添加 DSpark 非因果索引计算内核和 `decode_threshold` 调整，是注意力层的核心改动。
- `tests/v1/attention/test_dspark_noncausal_sparse_mla.py`（模块 测试；类别 test；类型 test-coverage；符号 `_build_dspark_noncausal_indices`, `_run_sparse_backend_vs_sdpa`, `_sparse_sdpa`, `_skip_if_backend_unavailable`）：新增非因果稀疏 MLA 正确性测试，验证 DSpark 非因果注意力实现与 SDPA 参考一致。

关键符号：`DSparkDeepseekV4Model.init`, `DSparkDeepseekV4Model.combine_hidden_states`, `DSparkDeepseekV4Model.forward`, `DSparkDeepseekV4Model._insert_context_kv`, `DSparkDeepseekV4ForCausalLM.init`, `DSparkMarkovHead.embed`, `DSparkMarkovHead.bias`, `Qwen3DSparkForCausalLM.markov_embed`, `Qwen3DSparkForCausalLM.markov_bias`, `DSparkSpeculator.load_draft_model`, `DSparkSpeculator._sample_sequential`, `DSparkSpeculator._generate_draft`, `_build_dspark_noncausal_indices`, `_run_sparse_backend_vs_sdpa`, `_compute_dspark_noncausal_swa_indices_kernel`, `load_dspark_model`, `use_dspark`

关键源码片段

`vllm/models/deepseek_v4/nvidia/dspark.py`

新增 DSparkDeepseekV4Model 和 DSparkDeepseekV4ForCausalLM, 定义 DSpark 草稿模型的核心结构, 复用目标解码层并添加马尔可夫头。

```
# SPDX-License-Identifier: Apache-2.0
```

```
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project
```

```
# DSparkDeepseekV4Model: 半自回归推测解码的草稿模型。
```

```
#
```

```
它复用目标模型的解码层 (DeepseekV4DecoderLayer) 并添加马尔可夫头 (DSparkMarkovHead) 。
```

```
# 非因果注意力通过扩展稀疏 MLA 索引实现: 将所有块内 token 包含在
```

```
# 每个 query 的 top-k 索引中, 并利用已有稀疏注意力内核的索引驱动能力。
```

```
class DSparkDeepseekV4Model(nn.Module):
```

```
    def __init__(self, *, vllm_config: VllmConfig, prefix: str = '') -> None:
```

```
        super().__init__()
```

```
        assert vllm_config.speculative_config is not None
```

```
        config = vllm_config.speculative_config.draft_model_config.hf_config
```

```
        self.config = config
```

```
        self.hidden_size = config.hidden_size
```

```
        self.hc_mult = config.hc_mult # 混合 chunk 头的倍数
```

```
        self.hc_eps = config.hc_eps
```

```
        self.rms_norm_eps = config.rms_norm_eps
```

```
        self.num_hidden_layers = config.num_hidden_layers
```

```
        self.target_layer_ids = tuple(config.dspark_target_layer_ids)
```

```
        self.num_dspark_layers = getattr(config, 'n_mtp_layers', None) or 3
```

```
        # 共享目标模型的词嵌入 (推测器加载时别名)
```

```
        self.embed_tokens = VocabParallelEmbedding(
            config.vocab_size, config.hidden_size,
            prefix=maybe_prefix(prefix, 'embed_tokens'),
        )
```

```
        # `main_proj` 将来自目标多个层的辅助隐藏状态投影到 hidden_size
```

```
        self.main_proj = ReplicatedLinear(
            config.hidden_size * len(self.target_layer_ids),
            config.hidden_size,
            bias=False, return_bias=False,
            quant_config=vllm_config.quant_config,
            prefix=maybe_prefix(prefix, 'main_proj'),
        )
```

```
        self.main_norm = RMSNorm(config.hidden_size, eps=config.rms_norm_eps)
```

```
        # 复用目标模型的解码层 (权重通过 MTP 层加载)
```

```
        current_vllm_config = get_current_vllm_config()
```

```
        self.layers = nn.ModuleList([
            DeepseekV4DecoderLayer(
                current_vllm_config,
                prefix=maybe_prefix(prefix, f'layers.{self.num_hidden_layers + i}'),
            )
            for i in range(self.num_dspark_layers)
```

```

])

# 最终层：系数归一化 + HC 头（线性参数） + 马尔可夫头
self.norm = RMSNorm(config.hidden_size, eps=config.rms_norm_eps)
hc_dim = self.hc_mult * config.hidden_size
self.hc_head_fn = nn.Parameter(
    torch.empty(self.hc_mult, hc_dim, dtype=torch.float32),
    requires_grad=False,
)
self.hc_head_base = nn.Parameter(
    torch.empty(self.hc_mult, dtype=torch.float32), requires_grad=False,
)
self.hc_head_scale = nn.Parameter(
    torch.empty(1, dtype=torch.float32), requires_grad=False,
)
# DSparkMarkovHead 在 qwen3_dspark.py 中定义
self.markov_head = DSparkMarkovHead(
    config.vocab_size,
    config.dspark_markov_rank,
    prefix=maybe_prefix(prefix, 'markov_head'),
)

def embed_input_ids(self, input_ids: torch.Tensor) -> torch.Tensor:
    return self.embed_tokens(input_ids)

def combine_hidden_states(self, aux_hidden_states: torch.Tensor) -> torch.Tensor:
    """main_x = main_norm(main_proj(concat of target aux hidden states))."""
    x = self.main_proj(aux_hidden_states)[0]
    return self.main_norm(x)

```

评论区精华

- depthfirst-app[bot]指出 DSparkSpeculator 中 sample_idx_mapping 填充 0 而非 -1 会导致 CUDA graph 回放时 draft logits 污染索引 0。作者已修复。
- LucasWilkinson询问 sparse_swa.py 中 decode_threshold 的 $2 \times \text{num_speculative_tokens}$ 含义。benchislett 解释是为支持 P-EAGLE 风格推理：query 块包含上一轮接受 token 和新 mask token，长度可达 $1 + 2 \times \text{num_spec_tokens}$ 。
- MatthewBonanni发现 _sample_sequential 中 sample_pos - 1 导致位置双倍递增。benchislett 确认并修复。
- zhyongye讨论目标模型隐藏状态为何使用 mean 池化而非完整 hidden_states。benchislett 说明这是 DSpark 的特征融合机制，与 EAGLE3/DFlash 一致，路径设计合理。
- mgoin建议 use_dspark() 中安全获取模型架构名称。benchislett 改为更安全的检查。
- njhill对 vllm.py 中列表格式提出 nits，部分被采纳。
- sample_idx_mapping 填充值应为 -1 而非 0 (correctness): 修复：将填充值从 0 改为 -1，与 DFlash 保持一致。
- decode_threshold 的 $2x$ 乘数含义 (design): 该乘数是合理且必要的，保持现有逻辑。

- `_sample_sequential` 中位置参数双倍递增 (correctness): 修复了位置计算偏差。
- 目标模型辅助隐藏层均值池化设计 (design): 接受当前设计, 认为合理。
- `use_dspark()` 安全获取架构名称 (security): 修改为安全访问方式。

风险与影响

- 风险:
 - 非因果索引宽度对齐: `sparse_swa.py` 中 `noncausal_index_width` 必须对齐 128 的倍数, 若窗口 + 块大小超过分配宽度可能静默截断。
 - CUDA Graphs 稳定性: `_sample_sequential` 固定 `num_speculative_steps`, 无法处理可变步数配置。
 - 权重别名映射: `fp4` 和 `fp8 expert scale` 后缀不同, `load_dspark_model` 需正确匹配, 否则加载错误。
 - 平台限制: 非因果注意力测试仅限 CUDA, AMD/Intel 会跳过, 跨平台兼容性未知。
 - 依赖 DFlash 基础: 深度继承 `DFlashSpeculator`, `DFlash` 的变更可能导致回归。
 - 马尔可夫头低秩假设: 固定 `markov_rank`, 模型需要高秩时性能受限。
- 影响:
 - 用户影响: 用户可使用 `--speculative_config '{"method": "dspark", ...}'` 加载 DSpark 模型, 获得高推测接受率。但仅限 CUDA 设备 (Blackwell/Hopper), 且需模型支持。
 - 系统影响: 新增约 1.8k 行核心代码, 稀疏 MLA 后端增加非因果分支, 配置系统扩展, 目标模型前向增加辅助隐藏状态收集逻辑 (通过 `aux_hidden_state_layers`)。
 - 团队影响: 需同时维护 DFlash 和 DSpark 两条推测路径, 代码复用度高但有定制点。增加跨 PR 测试要求。
 - 风险标记: 非因果索引宽度对齐要求, CUDA Graph 动态循环稳定性, 权重别名映射复杂性, 仅支持 CUDA 平台, 依赖 DFlash 基础设施

关联脉络

- PR #47102 Add Triton Backend for Unlimited-OCR R-SWA: 同为 speculative decoding 相关, 新增注意力后端, 但非直接关联。