PR #48869 完整报告

vllm-project/vllm

[Model] Add Inkling MTP=1 support [3/N]

合并时间：2026-07-17 04:27

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48869

执行摘要

- 一句话：为 Inkling 模型添加 MTP=1 推测解码支持
- 推荐动作：建议精读此 PR，了解 vLLM 如何为新模型集成 speculative decoding 支持，尤其是配置覆写、权重共享和融合核函数的设计模式。

功能与动机

Inkling 模型需要 speculative decoding 提升推理吞吐。本 PR 是 Inkling MTP 支持系列的第三个切片（继 #48799 基础模型和 #48822 CUDA 图之后），添加 MTP draft 层，使模型能利用一个推测 token 加速解码。

实现拆解

1. 配置集成：在 speculative.py 中添加 inkling_mtp 类型，并在 hf_config_override 中解析 checkpoint 的 mtp_config，设置 n_predict=1 和相关配置。
2. 核心模块：新增 mtp.py，定义 InklingMTPDepthLayer（单层 MTP 深度，含双 RMSNorm、输入投影和强制密集 MLP 的 Inkling 解码层）和 InklingMultiTokenPredictor（预测器，持有该层并共享 target 的 embedding 表和 LM head）。
3. 融合算子：在 norm.py 中新增 Triton 核函数 _embed_dual_rmsnorm_cat_kernel，实现 fused gather + dual RMSNorm + concat，减少 launch 开销。
4. 模型适配：修改 model.py，为 InklingDecoderLayer 添加 force_dense_mlp（MTP 层强制密集 MLP）和 defer_mlp_add（延迟残差添加）参数。
5. 注册与导出：在 init.py 和 registry.py 中注册 InklingMTP 架构。
6. 测试配套：新增 test_mtp_input_fusion.py（32 个 bit-exact 测试），补充配置覆写、注册表和契约测试。

关键文件：

- vllm/models/inkling/nvidia/mtp.py（模块 MTP 层；类别 source；类型 data-contract；符号 _mtp_depth_from_name, InklingMTPDepthLayer, init, forward）：新增 Inkling MTP draft 模型的核心实现，定义 InklingMTPDepthLayer 和 InklingMultiTokenPredictor，实现融合输入构造和权重加载。
- tests/models/inkling/test_mtp_input_fusion.py（模块 测试；类别 test；类型 test-coverage；符号 _ref, test_embed_dual_rmsnorm_cat, test_embed_rmsnorm）：为

融合输入核函数提供 bit-exact 测试，覆盖多种参数组合，验证 fused 操作与 unfused 序列结果一致。

- vllm/config/speculative.py（模块 配置；类别 source；类型 core-logic）：将 inkling_mtp 注册到 MTP 模型类型列表，添加配置覆写逻辑，从 checkpoint 提取 MTP 元数据并验证深度。
- vllm/models/inkling/nvidia/ops/norm.py（模块 融合算子；类别 infra；类型 infrastructure；符号 _embed_dual_rmsnorm_cat_kernel, embed_dual_rmsnorm_cat）：新增 Triton 融合核函数 embed_dual_rmsnorm_cat，将两个 RMSNorm 和一个 concat 合并为单个 kernel，减少 launch 开销。
- vllm/models/inkling/nvidia/model.py（模块 模型层；类别 source；类型 data-contract）：为 InklingDecoderLayer 添加 force_dense_mlp 和 defer_mlp_add 参数，支持 MTP 层强制使用密集 MLP 并延迟残差添加。
- vllm/models/inkling/__init__.py（模块 模块入口；类别 source；类型 data-contract）：导出 InklingMTP 类到模块顶层。
- vllm/models/inkling/configs.py（模块 配置；类别 source；类型 data-contract）：添加 chain_hidden_post_norm 和 local_layer_ids 等配置字段。
- tests/config/test_speculative_draft_hf_overrides.py（模块 测试；类别 test；类型 test-coverage；符号 test_inkling_override_exposes_only_first_mtp_depth）：测试 Inkling 配置覆写只暴露第一个 MTP 深度。
- vllm/model_executor/models/registry.py（模块 注册表；类别 source；类型 data-contract）：注册 InklingMTP 模型架构。
- tests/models/inkling/test_contract_validation.py（模块 测试；类别 test；类型 test-coverage；符号 test_inkling_mtp_chain_norm_is_disabled_by_default）：测试默认情况下 chain_hidden_post_norm 处于禁用状态。
- tests/models/registry.py（模块 测试；类别 test；类型 test-coverage）：更新注册表测试以涵盖 InklingMTP。

关键符号：_mtp_depth_from_name, InklingMTPDepthLayer.init, InklingMTPDepthLayer.forward, InklingMultiTokenPredictor.init, InklingMultiTokenPredictor.embed_input_ids, InklingMultiTokenPredictor.fused_input_cat, InklingMTP, embed_dual_rmsnorm_cat, _embed_dual_rmsnorm_cat_kernel, embed_rmsnorm, test_embed_dual_rmsnorm_cat, test_embed_rmsnorm, test_inkling_override_exposes_only_first_mtp_depth, test_inkling_mtp_chain_norm_is_disabled_by_default

关键源码片段

vllm/models/inkling/nvidia/mtp.py

新增 Inkling MTP draft 模型的核心实现，定义 InklingMTPDepthLayer 和 InklingMultiTokenPredictor，实现融合输入构造和权重加载。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project
```

```
"""Inkling MTP (Multi-Token Prediction) draft model (NVIDIA)."""

from __future__ import annotations

import regex as re
import torch
from torch import nn

from vllm.config import VllmConfig
from vllm.model_executor.layers.linear import ReplicatedLinear
from vllm.model_executor.layers.logits_processor import LogitsProcessor
from vllm.model_executor.layers.vocab_parallel_embedding import ParallelLMHead
from vllm.model_executor.model_loader.weight_utils import default_weight_loader
from vllm.model_executor.models.utils import maybe_prefix
from vllm.sequence import IntermediateTensors

from ..configs import InklingModelConfig
from .layernorm import InklingRMSNorm
from .model import InklingDecoderLayer, InklingReplicatedEmbedding
from .ops.norm import embed_dual_rmsnorm_cat, embed_rmsnorm

_ATTENTION_PARAMS_MAPPING = [
    ("qkvr", "wq_du", 0),
    ("qkvr", "wk_dv", 1),
    ("qkvr", "wv_dv", 2),
    ("qkvr", "wr_du", 3),
]

def _mtp_depth_from_name(name: str) -> int | None:
    m = re.search(r"\.mtp\.layers\.(\d+)\.", name)
    return int(m.group(1)) if m else None

class InklingMTPDepthLayer(nn.Module):
    """One MTP depth: norm both inputs, fuse (2H->H), run a Inkling block."""
    def __init__(self, config: InklingModelConfig, prefix: str, is_local: bool) -> None:
        super().__init__()
        self.hidden_norm = InklingRMSNorm(config.hidden_size, eps=config.rms_norm_eps)
        self.embed_norm = InklingRMSNorm(config.hidden_size, eps=config.rms_norm_eps)
        self.input_proj = ReplicatedLinear(
            config.hidden_size * 2,
            config.hidden_size,
            bias=False,
            return_bias=False,
            prefix=f"{prefix}.input_proj",
        )
        self.transformer_block = InklingDecoderLayer(
            config,
            layer_id=0,
            is_local=is_local,
```

```
            quant_config=None,
            prefix=f"{prefix}.transformer_block",
            nvfp4_config=None,
            force_dense_mlp=True,
        )
    def forward(self, combined: torch.Tensor, positions: torch.Tensor) -> torch.Tensor:
        hidden = self.input_proj(combined)
        return self.transformer_block(positions, hidden)
```

vllm/models/inkling/nvidia/ops/norm.py

新增 Triton 融合核函数 embed_dual_rmsnorm_cat，将两个 RMSNorm 和一个 concat 合并为单个 kernel，减少 launch 开销。

```
import torch
import triton
import triton.language as tl

@triton.jit
def _embed_dual_rmsnorm_cat_kernel(
    hidden_ptr, # [T, N] or [T, N] (GATHER: [V, N])
    emb_ptr, # [T, N] or [V, N]
    ids_ptr, # [T] token ids (GATHER only)
    w_hidden_ptr, # [N]
    w_pre_ptr, # [N] chain pre-norm (HAS_PRE_NORM only)
    w_embed_ptr, # [N]
    out_ptr, # [T, 2N]: [rmsnorm(hidden) | rmsnorm(rmsnorm?(emb))]
    eps,
    hidden_stride_0,
    emb_stride_0,
    n_cols,
    block_size_n: tl.constexpr,
    GATHER: tl.constexpr,
    HAS_PRE_NORM: tl.constexpr,
):
    """Triton kernel: compute rmsnorm for hidden (left) and embedding (right) in one launch."""
    pid_m = tl.program_id(0).to(tl.int64)
    which = tl.program_id(1) # 0 => hidden; 1 => emb
    offs_n = tl.arange(0, block_size_n)
    mask_n = offs_n < n_cols
    if which == 0:
        x = tl.load(hidden_ptr + pid_m * hidden_stride_0 + offs_n, mask=mask_n, other=0.0).to(tl.
        float32)
        w = tl.load(w_hidden_ptr + offs_n, mask=mask_n, other=0.0).to(tl.float32)
    else:
        row = tl.load(ids_ptr + pid_m).to(tl.int64) if GATHER else pid_m
        x = tl.load(emb_ptr + row * emb_stride_0 + offs_n, mask=mask_n, other=0.0).to(tl.float32)
        if HAS_PRE_NORM:
            w_pre = tl.load(w_pre_ptr + offs_n, mask=mask_n, other=0.0).to(tl.float32)
            rstd = tl.math.rsqrt(tl.sum(x * x, axis=0) / n_cols + eps)
```

```
            x = (x * rstd * w_pre).to(out_ptr.dtype.element_ty).to(tl.float32)
        w = tl.load(w_embed_ptr + offs_n, mask=mask_n, other=0.0).to(tl.float32)
    rstd = tl.math.rsqrt(tl.sum(x * x, axis=0) / n_cols + eps)
    tl.store(
        out_ptr + pid_m * (2 * n_cols) + which * n_cols + offs_n,
        (x * rstd * w).to(out_ptr.dtype.element_ty),
        mask=mask_n,
    )

def embed_dual_rmsnorm_cat(
    hidden: torch.Tensor,
    hidden_weight: torch.Tensor,
    embed_weight: torch.Tensor,
    eps: float,
    *,
    embeds: torch.Tensor | None = None,
    input_ids: torch.Tensor | None = None,
    embed_table: torch.Tensor | None = None,
    pre_norm_weight: torch.Tensor | None = None,
) -> torch.Tensor:
    """Fused cat([rmsnorm(hidden, w_h), rmsnorm(pre?(emb), w_e)], dim=-1)."""
    T, n = hidden.shape
    if embeds is not None:
        src, ids, src_stride = embeds, embeds, embeds.stride(0)
        gather = False
    else:
        assert input_ids is not None and embed_table is not None
        src, ids, src_stride = embed_table, input_ids, embed_table.stride(0)
        gather = True
    out = torch.empty((T, 2 * n), dtype=hidden.dtype, device=hidden.device)
    grid = (T, 2)
    _embed_dual_rmsnorm_cat_kernel[grid](
        hidden, src, ids, hidden_weight, pre_norm_weight, embed_weight,
        out, eps, hidden.stride(0), src_stride, n,
        block_size_n=triton.next_power_of_2(n),
        GATHER=gather, HAS_PRE_NORM=pre_norm_weight is not None,
    )
    return out
```

评论区精华

PR 提交后无实质性 Review 讨论，仅 claude[bot] 自动回复。所有 CI 和验证均通过。

- 暂无高价值评论线程

风险与影响

- 风险：仅支持 MTP=1，若配置 num_speculative_tokens>1 会抛出明确异常；融合核函数依赖 Triton 和 CUDA，非 GPU 环境不可用；缺少大规模压力测试；权重共享引用目标模型

表，避免了额外显存开销。

- 影响：影响限于 Inkling 模型用户；启用推测解码后可提升推理吞吐（GSM8K 测试中 MTP 接受率约 83-84%）；团队需维护新增的 MTP 模型代码和融合算子。
- 风险标记：仅支持 MTP=1，依赖 Triton/CUDA 硬件，缺少大规模压力测试，融合核函数 bit-exact 性依赖测试验证

关联脉络

- PR #48822 [Model] Add PW CUDA graph support for Inkling [2/N]: 同一模型系列的前序 PR，提供分段 CUDA 图支持，使 MTP 层也能受益于图捕获。
- PR #48858 [Model] Add Hopper FA4 relative attention for Inkling: 同一模型的相对注意力功能，与 MTP 无直接耦合但同属 Inkling 模型支持体系。