

PR #55107 完整报告

vllm-project/vllm

[Model][ROCm] Enable DeepSeek V4 Vision

合并时间: 2026-09-12 00:17

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/55107>

执行摘要

- 一句话: ROCm 支持 DeepSeek V4 Vision, 抽公共包装层并适配视觉路由
- 推荐动作: 值得精读, 尤其是正在做多平台模型适配或排查 DeepSeek V4 视觉 / 稀疏注意力问题的工程师。建议重点关注三处设计决策: (1) common/vl_model.py 中 " 外层 mapper 全量解析 + 子模型 mapper 置空 + finalizes_weights_during_load 决定是否排序 " 的权重加载契约, 它同时解决了 NVIDIA 自 finalize 与 ROCm 延迟 finalize 的分歧; (2) combine_topk_swa_indices 的 left_visible/right_visible 窗口代数以及 APC 命中 span 内部时的 clamp, 这里是整条图像可见性逻辑最容易被误改的地方; (3) DeepseekV4MoE 中 bias_vl、image_sentinel_lo、e_score_correction_bias 三者的分配条件, 以及 MTP 必须透传 input_ids 的隐含依赖。阅读时建议把 common/ops/cache_utils.py 中未修的同类 clamp 缺陷一并记下, 作为后续跟踪项。

功能与动机

PR body 明确说明动机: 现有 DeepSeek V4 视觉 checkpoint 由 PR #54566 引入时只带一个 NVIDIA-only wrapper 和非 CUDA 守卫, 导致 ROCm 用户无法使用该模型。为此作者把 " 平台中立的 wrapper " 搬到 common/, 保留 NVIDIA 兼容 shim 与 XPU 不支持桩, 并打开 ROCm 的 registry、dummy-init 与 tensor-schema 路径。此外, 正文还列出三条技术目标: 从当前生效的文本后端构建 checkpoint 映射 (NVIDIA 保留自 finalize, ROCm 延迟到通用逐层量化之后再幂等 finalize)、把图像哨兵经 bias_vl 接入 hash 与普通 ROCm MoE 层、在不改变纯文本行宽的前提下把稀疏 prefill 可见性扩展到完整图像 span。作者在评论中同时给出 4xgfg950 (ROCm、TP4/EP、FP8 KV、AITER) 的验收数据: OCRBench 825/1000 对 822/1000、GSM8K 1243/1319 对 1239/1319, 请求错误 0/0、截断 3/3。

实现拆解

1. 平台中立化包装层: 新增 vllm/models/deepseek_v4/common/vl_model.py (+331), 把 DeepseekV4ForConditionalGeneration、_make_deepseek_v4_vl_weights_mapper、embed_input_ids、embed_multimodal 等整体迁入; nvidia/vl_model.py 从 333 行缩减为 21 行兼容 shim, 仅为下游保留原来的 (expert_dtype, image_enabled) 签名, 内部委托给 common 版本并注入 _make_deepseek_v4_weights_mapper(expert_dtype)。vllm/models/deepseek_v4/__init__.py 的平台分发相应调整: ROCm 分支从 vl_stub 改为 common.vl_model, CUDA 分支也指向 common, vl_stub.py 的报错文案由 "only supported on NVIDIA GPUs" 改为 "use CUDA or ROCm"。

2. 权重映射与 finalize 时机解耦: common 版 mapper 改为接收 text_mapper: WeightsMapper, 把文本后端的前缀规则整体重挂到 language_model., 并把 head.weight 重定向到 language_model.lm_head.weight、把 mtp. 置为丢弃; image_enabled 为假时额外丢弃 vision./aligner./image_.wrapper 的 __init__ 会把子模型自身的 hf_to_vllm_mapper 置为 no-op (因为 suffix 规则不幂等, lm_head.weight 会再次命中 head.weight), 并用 self._weights_finalized 标记延迟 finalize; load_weights 通过 getattr(self.language_model, "finalizes_weights_during_load", True) 判断是否需要先排序以保证 NVIDIA 后端拿到连续的 language_model. 分组, ROCm 侧则由 DeepseekV4ForCausalLM.finalizes_weights_during_load = False 把 finalize 推到通用逐层量化之后。
3. ROCm MoE 视觉路由: vllm/models/deepseek_v4/amd/model.py 的 DeepseekV4MoE.__init__ 新增 gate.bias_vl 占位与 image_sentinel_lo = IMAGE_SENTINEL_BASE_ID (仅 vision_n_layers > 0 时为非 0), 并把 e_score_correction_bias 的分配条件从 elif 改为独立 if, 使 hash MoE 在视觉变体下也能拿到校正偏置; 两者随 FusedMoEFactory(..., bias_vl=..., image_sentinel_lo=...) 下传到专家层。forward 增加守卫: bias_vl 已分配但 input_ids is None 时直接报错, 避免静默路由错误。同文件新增 DeepseekV4ForCausalLM.compute_logits_local, 走 logits_processor(..., skip_gather=True), 供 ROCm 侧本地 logits 消费方使用。
4. 稀疏 prefill 的图像可见性: vllm/models/deepseek_v4/amd/rocm.py 的 _combine_topk_swa_indices_kernel 新增 left_visible_ptr/right_visible_ptr/HAS_IMAGE/SWA_WIDTH/PADDED_SWA_WIDTH, 把 swa_len = min(pos+1, WINDOW_SIZE) 改为按 left/right 计算 swa_start/swa_end 并分别 clamp 到 gather_start 与 seq_len; combine_topk_swa_indices 新增 max_image_tokens/left_visible/right_visible 参数, 用 swa_width = window_size + max_image_tokens 固定视觉模型行宽 (即使该 batch 无图), 并对左右可见性做配对校验。weight_already_preshuffled 从只看 quant_method 改为同时检查 quant_method 与 scheme, 与上游 AITER 预重排后端对齐; _forward_prefill 按 num_decode_tokens 与 query_start:query_end 切分可见性张量后传入。
5. 平台配置、MTP 与索引层: vllm/platforms/rocm.py 在 check_and_update_config 中镜像 CUDA 的逻辑, 对 is_mm_prefix_lm 且多模态的模型强制 disable_chunked_mm_input = True (带 warning_once); amd/mtp.py 改为把 input_ids 透传给 mtp_block, 使 MTP/DSpark 在视觉 checkpoint 下不会触发 bias_vl 守卫。vllm/models/deepseek_v4/{nvidia,amd}/model.py 与 __init__.py 的注册、dummy-init、tensor-schema 分支同步放开 ROCm。
6. 测试与 CI 配套: 新增 tests/models/test_deepseek_v4_vl_rocm.py (+458, 覆盖 preshuffle 行主序权重、gate_up 只 shuffle 一次、融合 QKV 量化与 preshuffled GEMM 的 scale 布局、KV cache dtype 解析、VL mapper 映射、hash/ 普通 MoE 的视觉路由、MTP 透传、compute_logits_local 跳过 gather); tests/kernels/attention/test_rocm_triton_attn_dsv4.py 新增 3 个 combine_topk_swa_indices 用例 (图像可见性、APC 命中 span 内部、无图时的行宽保持); tests/test_config.py 新增 test_rocm_mm_prefix_lm_disables_chunked_mm_input (4 组参数化); tests/models/test_registry.py、tests/models/multimodal/processing/test_tensor_schema.py 改用 current_platform.is_cuda_alike(); .buildkite/test_areas/models_basic.yaml

把新测试文件挂到镜像到 MI300 的 lane。

关键文件：

- `vllm/models/deepseek_v4/common/vl_model.py` (模块 多模态包装; 类别 source; 类型 data-contract; 符号 `_make_deepseek_v4_vl_weights_mapper`, `DeepseekV4ForConditionalGeneration`, `init`, `embed_input_ids`) : 新增的平台中立视觉包装类与权重映射工厂, 是整个 PR 的核心; 所有平台的 `DeepseekV4ForConditionalGeneration` 最终都指向这里, 同时承载了 NVIDIA/ROCm 两种 `finalize` 时机的兼容契约。
- `vllm/models/deepseek_v4/amd/rocm.py` (模块 稀疏注意力; 类别 source; 类型 core-logic; 符号 `_combine_topk_swa_indices_kernel`, `combine_topk_swa_indices`, `weight_already_preshuffled`, `_forward_prefill`) : ROCm 稀疏 `prefill` 的可见性内核与 Launcher 都改在这里: 新增 `left/right` 可见性、SWA 边界 `clamp` 与固定行宽, 是视觉长图正确性的关键; 同时修正 `weight_already_preshuffled` 的量化持有者检查。
- `vllm/models/deepseek_v4/amd/model.py` (模块 专家路由; 类别 source; 类型 core-logic; 符号 `DeepseekV4MoE.init`, `DeepseekV4MoE.forward`, `DeepseekV4ForCausalLM.compute_logits_local`, `finalizes_weights_during_load`) : ROCm 文本模型侧接入视觉路由与本地 `logits` 的落点: `bias_vl/image_sentinel_lo` 的分配与下传、`forward` 守卫、`finalizes_weights_during_load` 与 `compute_logits_local` 都在这里。
- `tests/models/test_deepseek_v4_vl_rocm.py` (模块 模型测试; 类别 test; 类型 test-coverage; 符号 `_preshuffled_fp8_linear`, `test_rocm_wo_a_keeps_row_major_weights`, `test_rocm_gateup_shuffles_once_and_down_proj_keeps_preshuffled_backend`, `test_rocm_fused_qkv_quant_matches_preshuffled_gemm_scale_layout`) : 新增的 ROCm 视觉回归套件 (+458), 覆盖 FP8 预重排权重布局、KV cache dtype 解析、VL mapper 映射、MoE 视觉路由与 MTP 透传, 是这次跨模块改动的主要安全网。
- `vllm/models/deepseek_v4/nvidia/vl_model.py` (模块 兼容垫片; 类别 source; 类型 refactor; 符号 `_make_deepseek_v4_vl_weights_mapper`, `DeepseekV4ForConditionalGeneration`) : 由 333 行实现退化为 21 行兼容 shim, 保留原有 (`expert_dtype`, `image_enabled`) 签名以不破坏下游导入, 是这次目录重构的兼容性枢纽。
- `vllm/platforms/rocm.py` (模块 平台配置; 类别 source; 类型 core-logic; 符号 `RocmPlatform.check_and_update_config`) : 在 `check_and_update_config` 中镜像 CUDA 的多模态前缀注意力守卫, 强制 `disable_chunked_mm_input`, 是 review 中第一个被指出的正确性缺口, 直接决定了图像 span 能否被原子化 `prefill`。
- `vllm/models/deepseek_v4/__init__.py` (模块 注册分发; 类别 source; 类型 entrypoint; 符号 `DeepseekV4ForConditionalGeneration`) : 平台分发入口: ROCm 分支从 `vl_stub` 切到 `common.vl_model`, CUDA 分支同样改指 `common`, 决定了 `DeepseekV4ForConditionalGeneration` 在各平台解析到的实现。
- `tests/kernels/attention/test_rocm_triton_attn_dsv4.py` (模块 注意力测试; 类别 test; 类型 test-coverage; 符号 `test_combine_topk_swa_indices_adds_image_visibility`, `test_combine_topk_swa_indices_apc_hit_inside_image`, `test_combine_topk_swa_indices_keeps_vision_row_width_without_images`) : 为

combine_topk_swa_indices 新增三个用例，直接锚定图像可见性扩展、APC 命中 span 内部与无图时的行宽保持，是稀疏注意力改动的主要验证。

- tests/models/test_initialization.py (模块 初始化测试; 类别 test; 类型 test-coverage) : review 中围绕 KV cache dtype 契约的交锋落点: 作者最终移除了此处的 per-arch 测试 override, 改为在 packed 布局把 auto 解析为 fp8_ds_mla 并对非法 dtype 抛可读错误。

关键符号: _make_deepseek_v4_vl_weights_mapper,

DeepseekV4ForConditionalGeneration.init, DeepseekV4ForConditionalGeneration.load_weights, DeepseekV4ForConditionalGeneration.embed_input_ids, combine_topk_swa_indices, _combine_topk_swa_indices_kernel, weight_already_preshuffled, DeepseekV4MoE.init, DeepseekV4MoE.forward, DeepseekV4ForCausalLM.compute_logits_local, RocmPlatform.check_and_update_config

关键源码片段

vllm/models/deepseek_v4/common/vl_model.py

新增的平台中立视觉包装类与权重映射工厂，是整个 PR 的核心; 所有平台的 DeepseekV4ForConditionalGeneration 最终都指向这里，同时承载了 NVIDIA/ROCm 两种 finalize 时机的兼容契约。

```
# vllm/models/deepseek_v4/common/vl_model.py
def _make_deepseek_v4_vl_weights_mapper(
    text_mapper: WeightsMapper, image_enabled: bool
) -> WeightsMapper:
    """Text-checkpoint mapping rules re-rooted under ``language_model.``"""
    # 文本后端的整棵命名空间整体下沉一层; dst 为 None 表示显式丢弃
    orig_to_new_prefix = {
        src: None if dst is None else f"language_model.{dst}"
        for src, dst in text_mapper.orig_to_new_prefix.items()
    }
    # 未启用图像时直接丢弃视觉塔权重，避免出现永远不被使用的参数
    if not image_enabled:
        orig_to_new_prefix.update({"vision.": None, "aligner.": None, "image_": None})
    return WeightsMapper(
        orig_to_new_renaming=text_mapper.orig_to_new_renaming,
        orig_to_new_prefix=orig_to_new_prefix,
        orig_to_new_regex=text_mapper.orig_to_new_regex,
        orig_to_new_stacked=text_mapper.orig_to_new_stacked,
        orig_to_new_suffix={
            **text_mapper.orig_to_new_suffix,
            # lm_head 挂在 wrapper 的 language_model 下，需要单独重定向
            "head.weight": "language_model.lm_head.weight",
        },
        orig_to_new_substr={
            **text_mapper.orig_to_new_substr,
            # MTP/DSpark draft 权重由 draft 模型自己加载，target 侧必须丢弃
            "mtp.": None,
        },
    )
```

)

```
class DeepseekV4ForConditionalGeneration(nn.Module, SupportsMultiModal, SupportsPP, SupportsEagle3):
```

```
    # MoE router 需要原始 token id 来识别图像哨兵 token 并应用 bias_vl
    requires_raw_input_tokens = True
```

```
    def __init__(self, *, vllm_config, prefix: str = "") -> None:
        # ... vision / aligner / image_start 等参数构造略 ...
        # 外层 mapper 已经把 HF 名字完整解析到 wrapper 命名空间;
        # AutoWeightsLoader 剥掉 "language_model." 前缀后会把权重委托给子模型,
        # 因此子模型自身的 mapper 必须是 no-op: 它的 suffix 规则并不幂等
        # (例如 "lm_head.weight".endswith("head.weight") 会二次命中)。
        text_mapper = self.language_model.hf_to_vllm_mapper
        self.language_model.hf_to_vllm_mapper = WeightsMapper()
        self.make_empty_intermediate_tensors = (
            self.language_model.make_empty_intermediate_tensors
        )

        self.hf_to_vllm_mapper = _make_deepseek_v4_vl_weights_mapper(
            text_mapper, image_enabled
        )
        # ROCm 侧延迟且幂等 finalize 的哨兵标志
        self._weights_finalized = False
```

```
    def load_weights(self, weights: Iterable[tuple[str, torch.Tensor]]) -> set[str]:
        # NVIDIA 文本后端在 load_weights 内部自 finalize, 必须一次拿到
        # 连续的 "language_model." 分组, 因此这里要先按名字排序;
        # ROCm 后端把 finalizes_weights_during_load 置为 False,
        # 由外层在通用逐层量化之后再做一次延迟 finalize。
        child_finalizes = getattr(
            self.language_model, "finalizes_weights_during_load", True
        )
        mapped = self.hf_to_vllm_mapper.apply(weights)
        if child_finalizes:
            mapped = iter(sorted(mapped, key=lambda x: x[0]))
        loader = AutoWeightsLoader(self)
        # 之后由 AutoWeightsLoader 按连续分组把 "language_model." 前缀的
        # 权重委托给子模型的 load_weights
```

vllm/models/deepseek_v4/amd/model.py

ROCm 文本模型侧接入视觉路由与本地 logits 的落点: bias_vl/image_sentinel_lo 的分配与下传、forward 守卫、finalizes_weights_during_load 与 compute_logits_local 都在这里。

```
# vllm/models/deepseek_v4/amd/model.py
class DeepseekV4MoE(nn.Module):
    def __init__(self, vllm_config, prefix: str = "", fuse_heterogeneous_shared_expert: bool = False):
```

```

# ... 配置读取略 ...
self.gate.e_score_correction_bias = None
self.gate.tid2eid = None
self.gate.bias_vl = None
# 视觉模型才用 IMAGE_SENTINEL_BASE_ID 作为哨兵下界，纯文本走 0
self.image_sentinel_lo = (
    IMAGE_SENTINEL_BASE_ID if getattr(config, "vision_n_layers", 0) > 0 else 0
)
is_hash_moe = extract_layer_index(prefix) < config.num_hash_layers
self.hash_indices_dtype = torch.int32
if is_hash_moe:
    # 用 randint 而非 empty，避免 dummy 模式下垃圾值触发非法访存
    self.gate.tid2eid = nn.Parameter(
        torch.randint(
            0,
            config.n_routed_experts,
            (config.vocab_size, config.num_experts_per_tok),
            dtype=self.hash_indices_dtype,
        ),
        requires_grad=False,
    )
# 由 elif 改为独立 if：视觉变体下 hash MoE 同样需要 correction bias
if getattr(config, "topk_method", None) == "noaux_tc" and (
    not is_hash_moe or getattr(config, "vision_n_layers", 0) > 0
):
    self.gate.e_score_correction_bias = nn.Parameter(
        torch.empty(config.n_routed_experts, dtype=torch.float32),
        requires_grad=False,
    )
# 图像 token 的额外路由偏置，仅在视觉 checkpoint 下分配
if getattr(config, "vision_n_layers", 0) > 0:
    self.gate.bias_vl = nn.Parameter(
        torch.empty(config.n_routed_experts, dtype=torch.float32),
        requires_grad=False,
    )

# ... shared_experts / TP 切分略 ...

self.experts = FusedMoEFactory(
    # ... 其余参数略 ...
    e_score_correction_bias=self.gate.e_score_correction_bias,
    hash_indices_table=self.gate.tid2eid,
    bias_vl=self.gate.bias_vl,
    image_sentinel_lo=self.image_sentinel_lo,
    swiglu_limit=self.swiglu_limit,
    router_logits_dtype=torch.float32,
    # ... routed_experts_cls 等略 ...
)

```

```

def forward(self, hidden_states: torch.Tensor, input_ids: torch.Tensor | None = None):
    if self.gate.tid2eid is not None and input_ids is None:
        raise ValueError("DeepSeek V4 hash MoE routing requires input_ids.")
    # 分配了 bias_vl 就必须拿到原始 token id, 否则无法识别图像哨兵;
    # 这条守卫会把上游 (如 MTP) 未透传 input_ids 的错误暴露出来
    if self.gate.bias_vl is not None and input_ids is None:
        raise ValueError("DeepSeek V4 vision MoE routing requires input_ids.")

    org_shape = hidden_states.shape
    final_hidden_states = self.experts(
        hidden_states=hidden_states,
        router_logits=hidden_states,
        input_ids=input_ids,
    )
    return final_hidden_states.view(org_shape)

```

```

class DeepseekV4ForCausalLM(nn.Module, SupportsPP, SupportsEagle3):
    # ROCm 后端不在 load_weights 内部 finalize, 交由外层延迟且幂等地执行
    finalizes_weights_during_load = False

```

```

def compute_logits_local(self, hidden_states: torch.Tensor) -> torch.Tensor:
    # skip_gather=True 只在本 rank 上算 local 词表 logits, 供 ROCm 侧消费
    return self.logits_processor(self.lm_head, hidden_states, skip_gather=True)

```

tests/models/test_deepseek_v4_vl_rocm.py

新增的 ROCm 视觉回归套件 (+458), 覆盖 FP8 预重排权重布局、KV cache dtype 解析、VL mapper 映射、MoE 视觉路由与 MTP 透传, 是这次跨模块改动的主要安全网。

```

# tests/models/test_deepseek_v4_vl_rocm.py
pytestmark = pytest.mark.skipif(
    not current_platform.is_rocm(), reason="ROCm-specific tests"
)

```

```

def test_rocm_moe_wires_vision_routing_on_hash_and_regular_layers(
    monkeypatch: pytest.MonkeyPatch,
) -> None:
    from vllm.models.deepseek_v4.amd import model as rocm_model
    from vllm.models.deepseek_v4.common.mm_preprocess import IMAGE_SENTINEL_BASE_ID

```

```

    captured: list[dict] = []

```

```

class FakeGate(nn.Module):
    def __init__(self, **kwargs) -> None:
        super().__init__()

```

```

def fake_factory(**kwargs):
    # 拦截 FusedMoEFactory 参数, 验证视觉路由确实被下传

```

```

        captured.append(kwargs)
    return nn.Identity()

monkeypatch.setattr(rocm_model, "GateLinear", FakeGate)
monkeypatch.setattr(rocm_model, "FusedMoEFactory", fake_factory)
monkeypatch.setattr(rocm_model, "get_tensor_model_parallel_world_size", lambda: 1)
monkeypatch.setattr(rocm_model, "get_tensor_model_parallel_rank", lambda: 0)

config = SimpleNamespace(
    hidden_size=16, n_routed_experts=4, num_experts_per_tok=2,
    moe_intermediate_size=8, swiglu_limit=None, norm_topk_prob=True,
    scoring_func="sqrtsoftplus", num_hash_layers=1, vocab_size=32,
    topk_method="noaux_tc", vision_n_layers=1, n_shared_experts=None,
)
vllm_config = SimpleNamespace(
    model_config=SimpleNamespace(hf_config=config), quant_config=None
)

# layer 0 是 hash MoE, layer 1 是普通 MoE, 两条路径都要接上 bias_vl
hash_moe = rocm_model.DeepseekV4MoE(vllm_config, prefix="model.layers.0.ffn")
regular_moe = rocm_model.DeepseekV4MoE(vllm_config, prefix="model.layers.1.ffn")

assert hash_moe.gate.tid2eid is not None
assert regular_moe.gate.tid2eid is None
for moe, factory_kwargs in zip((hash_moe, regular_moe), captured, strict=True):
    assert moe.gate.e_score_correction_bias is not None
    assert moe.gate.bias_vl is not None
    assert factory_kwargs["bias_vl"] is moe.gate.bias_vl
    assert factory_kwargs["image_sentinel_lo"] == IMAGE_SENTINEL_BASE_ID

```

评论区精华

讨论集中在 okorzh-amd 提出的五个技术问题，作者逐条回应，最终 tjtanana 批准。

1. chunked prefill 与图像 span 的前置条件：okorzh-amd 指出注释里 "atomically-prefilled image span" 是承重前提，但 "is_mm_prefix_lm is True for the vision variant ... yet that forcing exists only in vllm/platforms/cuda.py:320-331. RocmPlatform.check_and_update_config has no equivalent"。作者回应 "I mirrored CUDA's multimodal prefix-LM guard in RocmPlatform.check_and_update_config", 对应 vllm/platforms/rocm.py 新增的强制 disable_chunked_mm_input 代码与 tests/test_config.py 的新用例。
2. APC 命中落在图像 span 内部：okorzh-amd 指出 swa_end = pos + right + 1 前向越界、swa_start 未 clamp 到 gather_start，并提醒 "the same clamp looks absent from the shared common/ops/cache_utils.py kernel ... this is pre-existing and affects CUDA too, and should be split into its own PR"。作者在 ROCm kernel 补上 gather_start/seq_len 双向 clamp 并新增 APC 回归测试，但明确 "I left the shared CUDA/common change out of this PR, cause I think it can be addressed

separately"——共享路径的缺陷仍然存在。

3. 测试文件标记与 CI 可见性: okorzh-amd 指出 `pytest.mark.cpu_test` 不成立 (`amd.model` 会链式导入 `vllm/platforms/rocm.py`, 其模块级 `_GCN_ARCH = _get_gcn_arch()` 在非 ROCm 上会返回 `torch.cuda.get_device_properties()`), 且 "this file is named by no CI job ... otherwise none of these tests can ever fail in CI". 作者移除 `cpu_test`、加文件级 ROCm `skipif` 并写入 Basic Models lane。
4. KV cache dtype 的可用性缺口: okorzh-amd 指出 `sparse_mla.py` 宣称支持 "auto", 但 `attention.py:105` 断言 `kv_cache_dtype.startswith("fp8")`, 导致默认参数下 `vllm serve` 能过校验却在运行期抛裸 `AssertionError`; 作者随后移除测试内的 per-arch override, 改为 packed 布局把 auto 解析成 `fp8_ds_mla`, 显式不支持的 dtype 给出可读 `ValueError`。
5. bias_vl 与 MTP 的冲突: okorzh-amd 指出 `bias_vl` 在所有 MoE 层 (含 MTP block) 分配, 而 `amd/mtp.py:159` 硬编码 `input_ids=None`, 于是视觉 checkpoint 开投机解码会在 MoE forward 深处抛 `ValueError`, 建议配置期拒绝。作者不认同配置期拒绝, 理由是 "The target mapper drops mtp.* because the separate DeepSeekV4MTP draft model remaps and loads those weights itself. NVIDIA MTP already forwards input_ids, and AMD DSpark does as well", 并改为让 AMD MTP 透传 `input_ids` 加回归测试。

另有 Isotr0py 对测试改用 `current_platform.is_cuda_alike()` 的两处简化建议 (作者回 "Done :)"), 以及 tjtanana 的批准: "LGTM. Let check the Cuda ci."

- ROCm 缺少 `disable_chunked_mm_input` 强制逻辑, 图像 span 可能被 chunk 边界切开 (correctness): 作者在 `vllm/platforms/rocm.py` 中镜像了 CUDA 的多模态 prefix-LM 守卫, 并对 `is_mm_prefix_lm` 且多模态的模型强制 `disable_chunked_mm_input`, 同时补了 `tests/test_config.py` 的参数化用例。
- APC 命中落在图像 span 内部时 SWA 边界未 clamp (correctness): 作者在 ROCm 新内核上补齐 `swa_start/swa_end` 双向 clamp 并新增 APC-mid-span 回归测试, 但明确把共享 CUDA/common 的同类修复留在本 PR 之外单独处理, 该缺陷仍然存在。
- 新测试文件的 `cpu_test` 标记不成立且无 CI job 引用 (testing): 作者移除 `pytest.mark.cpu_test`、加文件级 ROCm `skipif`, 并把该文件写进镜像到 MI300 的 Basic Models lane。
- KV cache dtype "auto" 与稀疏 MLA 断言冲突 (design): 作者移除测试专用 override, 改为 packed 布局把 auto 解析为 `fp8_ds_mla`, 显式不支持的 dtype 给出可操作的 `ValueError`。
- bias_vl 与 MTP 硬编码 `input_ids=None` 冲突 (correctness): 作者不同意配置期拒绝: target mapper 丢弃 mtp. 是因为独立 draft 模型会自行重映射加载这些权重, 且 NVIDIA MTP 与 AMD DSpark 都会透传 `input_ids`; 最终改为让 AMD MTP 透传 `input_ids` 并加回归测试。
- 测试中平台判断应改用 `is_cuda_alike` (style): 作者回复 "Done :) ", 两处均已简化。

风险与影响

- 风险:

1. 共享 kernel 的同类缺陷未修: `common/ops/cache_utils.py` 中同族的 `combine kernel` 仍缺少 `gather_start clamp`, okorzh-amd 判断这是 "pre-existing and affects CUDA

too", 本 PR 只修 ROCm 侧。这意味着 CUDA 上的 APC 命中图像 span 场景仍存在越界索引风险, 需要一个后续 PR 收口, 否则容易在后续重构中被误认为已解决。

2. 行宽从 `window_size` 变为 `window_size + max_image_tokens`:

`combine_topk_swa_indices` 现在对视觉模型固定更宽的 `swa_width` 并做 128 对齐, 这会放大 `combined_indices` 的显存占用与 kernel 的 `PADDED_SWA_WIDTH`; 纯文本模型依赖 `max_image_tokens=0` 才能保持原状, 若调用方 (`_forward_prefill` 之外的路径) 未传该参数, 行为需要确认。

3. KV cache dtype 语义变化: `packed` 布局下 "auto" 被解析为 `fp8_ds_mla`, 显式非 `fp8` 则报 `ValueError`。这是面向用户的契约变更, 影响所有依赖默认值启动 DeepSeek V4 的用户脚本, 文档与示例需要同步。

4. 强制 `disable_chunked_mm_input` 的吞吐代价: ROCm 上所有 `is_mm_prefix_lm` 多模态模型都会被静默改为不分块输入, 长上下文多模态请求的显存峰值与调度行为会变化, 属于正确性优先的取舍, 但缺少性能回归数据。

5. 投机解码路径的耦合: `bias_vl` 分配在包括 MTP block 的所有 MoE 层, `input_ids` 依赖 AMD MTP 透传才成立; 若后续有新的 draft 实现 (或 `mtp.py` 再次改成不透传), 会在 MoE forward 深处抛出 `ValueError` 而非配置期失败, 排障成本高。

6. PR 历史混杂与大幅返工: 9 个 commit 中包含 #55095 的 `cuda_graph_fallback` 提交与多次 Merge origin/main, head commit 还移除了自定义的 row-major FP8 workaround 以采纳上游 #56433 的 AITER `preshuffle` 修复。

`weight_already_preshuffled` 同时检查 `quant_method` 与 `scheme` 是为了避免二次 shuffle, 但这条路径的正确性依赖 AITER 后端的 `preshuffles_weight` 标记在所有量化持有者上都正确设置, 属于较脆的隐式契约。- 影响: 对用户: ROCm (gfx950/MI300X) 用户首次可以直接以 `DeepseekV4ForConditionalGeneration` 架构启动

`DeepSeek-V4-Flash-Vision-Exp`, 无需 NVIDIA 卡; 同时默认 KV cache dtype 的解析行为变化, 脚本中显式传 `--kv-cache-dtype bfloat16` 之类的组合会得到新的报错信息。

对系统: `vllm/models/deepseek_v4/` 的目录结构从 "平台各写一份" 转为 "common wrapper + 平台 thin shim", 后续 XPU/CUDA 的视觉能力扩展只需按同一模式补 shim; MoE 路由契约新增 `bias_vl/image_sentinel_lo` 两个参数, 任何自定义 MoE 工厂或 `FusedMoE` 调用方都需要感知。对团队: CI 层面把新测试文件显式写进

`buildkite/test_areas/models_basic.yaml` 并镜像到 MI300, 说明 vLLM 的 AMD CI 仍然依赖逐文件列举而非 glob, 新增测试文件的可见性需要人工维护, 这是后续容易遗漏的协作成本。- 风险标记: 核心路径变更, 共享 Kernel 遗留缺陷未修, KV cache dtype 契约变更, MTP 与视觉路由的隐式耦合, stacked PR 合并历史混杂, CI 覆盖依赖手工列举 lane

关联脉络

- PR #54566 DeepSeek V4 Vision 初始支持 (仅 NVIDIA wrapper, 标题按 PR body 描述归纳): PR body 明确说明本 PR 是其延续: 该 PR 以 NVIDIA-only wrapper 和非 CUDA 守卫引入 DeepSeek V4 视觉 checkpoint, 本 PR 才把包装层搬到 common/ 并打开 ROCm。
- PR #55095 [Bugfix] Fall back to full decode graphs for noncompiled models: 本 PR 的 commit 历史中直接包含了该 PR 的提交 (42c2a93c、b751ff84、8ad7e328、f07dcd66)

, 9759753e 的合并信息写明 "Use the upstream CUDA graph fallback implementation from #55095", 属于 stacked 关系。

- PR #56433 [ROCm][Bugfix] Fix AITER preshuffled FP8 block-scale kernel: head commit 4eab7c39 说明本 PR 采纳了该上游 AITER 预重排 FP8 修复, 并移除自己此前为绕过该问题而写的 row-major 内核 workaround, 同时调整了 FP8 数值回归的权重与 scale 布局契约。
- PR #56214 [Model] Support DeepSeek-V4.1-Flash: 同属 DeepSeek V4 模型族; 9759753e 的合并信息提到 "Keep incoming runtime, quantization, multimodal embedding, and DeepSeek V4.1 fixes and features", 两者的权重映射与 MLA/MoE 代码路径高度重叠。
- PR #55667 [ROCm][CI] Add HY-V4 generation coverage: 同为 AMD CI lane 的维护: head commit 提到 "retain both DeepSeek vision and incoming HY-V4 tests in the AMD CI job", 两 PR 都改动 .buildkite 下的 models_basic/models_distributed lane。