

PR #54160 完整报告

vllm-project/vllm

[Hy4] support Hy4-preview model

合并时间: 2026-08-29 12:07

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/54160>

执行摘要

- 一句话: 新增腾讯 Hy4-preview 770B MoE 模型支持
- 推荐动作: 值得精读。重点关注几个设计决策: 1) flashmla_sparse.py 通过子类化补 sink 支持但保留 get_name() 不变, 避免 KV cache dtype 行为漂移; 2) compute_skip_topk_layers 对 full/shared 索引器层的权重跳过推导; 3) _ihc_supported 把内核约束集中门控, 避免静默回退; 4) MTP 量化映射的容错设计。同时应把 is_deepseek_mla 白名单维护作为新 MLA 模型接入的标准检查项。

功能与动机

PR body 明确指出 Hy4-preview 是腾讯 Hunyuan 的新一代开源 MoE 模型: 770B 总参数、每 token 激活 49B、78 层中首层密集 FFN、其余 77 层各用 256 路由专家 (top-8) 加 1 共享专家, 并带原生 MTP 层用于投机解码。架构上受 DeepSeek 与 GLM 启发, 采用 Gated DeepSeek 稀疏注意力加 IndexCache 跨层索引复用、iHC 残差通道扩大层间信息流。vLLM 需要完整的模型实现、注意力后端对接、解析器配套才能让用户通过标准 OpenAI 接口运行该模型。

实现拆解

1. 模型主体: 新增 vllm/models/hy_v4/nvidia/ 包, 入口为 model.py 中的 HYV4Model 与 HYV4DecoderLayer, 通过 _normalize_hyv4_config 将 Hy 的 config 字段归一化为 vLLM 共享 MoE/MLA 实现所期望的别名, 并声明 SupportsPP、SupportsLoRA 等接口。
2. 稀疏 MLA 注意力: attention.py 实现 HYV4MLAAttention, 基于 MLAAttention 扩展, 引入 compute_skip_topk_layers 与 is_skip_topk_indexer_weight, 处理 IndexCache 中 full/shared 索引器层的权重跳过逻辑; flashmla_sparse.py 子类化 FlashMLASparseImpl, 补上通用 FLASHMLA_SPARSE 后端缺失的 per-head learnable sink 透传, 并刻意保留继承的 get_name() (FLASHMLA_SPARSE), 避免改变 KV cache dtype 规范化行为。
3. iHC 残差通道与 MoE: hc.py 实现 HYV4HCPreLayer、HYV4HCPostLayer、HYV4HCHeadLayer; moe.py 定义 HYV4FeedForward 与 HYV4MoEFused, 复用 fused_moe 的 expert 参数映射机制。
4. MTP 投机解码: mtp.py 实现 HYV4SharedHead 以及一组量化权重映射 helper (_get_spec_layer_idx_from_weight_name、_should_skip_missing_mtp_scale_param、_resolve_fused_expert_param、_prepare_mtp_fp8_expert_scale、_create_mtp_quant_config、_remap_mtp_quant_exclusions), 专门处理 FP8

blockwise 128x128 与 MXFP8 的 scale 参数加载。

5. HPC 可选融合: model_executor/layers/hpc/hpc_ihc.py 新增 HpcIHCPre/HpcIHCPPost/HpcIHCHead, 把 eager 路径的 20/5/15 次 kernel launch 压为单内核; _ihc_supported 集中门控环境变量、hpc 包、计算能力 (sm100/sm103)、形状 (hc_mult==4、hidden_size 4096/6144) 与 VLLM_BATCH_INVARIANT 约束。
6. 解析与注册配套: vllm/tool_parsers/hy_v4_tool_parser.py (1166 行)、vllm/reasoning/hy_v4_reasoning_parser.py (324 行)、structural_tag_registry.py 新增 hy_v4 条目; Rust 侧 rust/src/parser/src/tool/hy.rs 由 hy_v3.rs 重命名, 并新增 unified/hy/hy_v4.rs。根据 review 意见, is_deepseek_mla() 白名单已补充 hy_v4, 确保读取 model_config 的 DCP、PCP、disagg connectors 与 nvfp4-KV-MLA guard 路径看到 use_mla=True。
7. 测试与文档: 涉及解析器、注册表、权重加载路径的测试同步新增或更新, 但 PR 描述中未给出模型端到端测试结果, 依赖多轮 Buildkite CI 验证; commit 历史显示 ROCm 失败、MTP 权重缺失、并发请求结果错乱等均已修复。

关键文件:

- vllm/models/hy_v4/nvidia/model.py (模块 模型层; 类别 source; 类型 core-logic; 符号 _normalize_hyv4_config, HYV4DecoderLayer, HYV4Model, embed_input_ids): Hy4-preview 主干模型入口, 定义 HYV4Model、HYV4DecoderLayer 与配置归一化, 串联注意力、iHC 与 MoE, 是全部新增模型的装配中心。
- vllm/models/hy_v4/nvidia/attention.py (模块 注意力; 类别 source; 类型 core-logic; 符号 compute_skip_topk_layers, is_skip_topk_indexer_weight, HYV4MLAAttention, prepare_inputs): Gated DSA 稀疏注意力与 IndexCache 核心: HYV4MLAAttention、top-k 跳过层计算、FP8 indexer cache。
- vllm/models/hy_v4/nvidia/mtp.py (模块 MTP 头; 类别 source; 类型 data-contract; 符号 _get_spec_layer_idx_from_weight_name, _should_skip_missing_mtp_scale_param, _resolve_fused_expert_param, _prepare_mtp_fp8_expert_scale): MTP 投机解码头与 FP8/MXFP8 量化权重映射, 含多个容错 helper, 是投机解码正确性的关键。
- vllm/models/hy_v4/nvidia/flashmla_sparse.py (模块 稀疏后端; 类别 source; 类型 core-logic; 符号 HYV4FlashMLASparseImpl, _validate_sinks, _sinks_for_query, _fp8_flash_mla_kernel): 为 FLASHMLA_SPARSE 后端补上 Hy4 的 per-head learnable sink 透传, 避免注意力偏置被静默丢弃。
- vllm/model_executor/layers/hpc/hpc_ihc.py (模块 融合内核; 类别 source; 类型 core-logic; 符号 _ihc_supported, HpcIHCPre, HpcIHCPPost, HpcIHCHead): 单内核融合 iHC 替代 eager 路径的 20/5/15 次 kernel launch, 门控逻辑严谨, 是性能路径的关键。
- vllm/tool_parsers/hy_v4_tool_parser.py (模块 工具解析; 类别 source; 类型 dependency-wiring; 符号 detect_token_suffix, _normalize_type, _get_arg_schema, ToolSchema): 新增 1166 行 Hy V4 工具调用解析器, 支持 schema 提取与时流式工具调用, 是工具调用功能的入口。
- rust/src/parser/src/tool/hy.rs (模块 解析器; 类别 source; 类型 rename-or-move; 符号 HyInput, HyMode, HyEvent): 从 hy_v3.rs 重命名为 hy.rs, 将 V3 解析器抽象为 HyInput/HyMode 并兼容 V4, 是 Rust 前端解析器演进关键。

- vllm/tool_parsers/structural_tag_registry.py (模块注册表; 类别 source; 类型 core-logic; 符号 register_vllm_structural_tag, _hy_v4_argument_keys, get_hy_v4_structural_tag, arg_pair) : 注册 hy_v4 结构化标签与参数提取规则, 并处理 tool_choice=auto 时跳过标签的逻辑。

关键符号: HYV4Model, HYV4MLAAttention, compute_skip_topk_layers, HYV4FlashMLASparseImpl, HYV4MoEFused, HYV4HCPreLayer, HYV4SharedHead, _get_spec_layer_idx_from_weight_name, HYV4ReasoningExtractor, detect_token_suffix, _ihc_supported

关键源码片段

vllm/models/hy_v4/nvidia/attention.py

Gated DSA 稀疏注意力与 IndexCache 核心: HYV4MLAAttention、top-k 跳过层计算、FP8 indexer cache。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project
# HY V4 的 MLA 注意力与闪电索引器 (NVIDIA) 。
# 索引器分 full 与 shared 两种: shared 层不建自己的索引器,
# 直接复用最近一个 full 层的 top-k 结果, 因此其 checkpoint
# 索引器权重必须跳过。compute_skip_topk_layers 负责算出这些层。

def compute_skip_topk_layers(config: PretrainedConfig) -> set[int]:
    # 没有 index_topk 配置的模型不需要 sparse, 直接返回空集
    if not hasattr(config, 'index_topk'):
        return set()

    num_hidden_layers = config.num_hidden_layers
    indexer_types = getattr(config, 'indexer_types', None)
    if indexer_types is not None:
        # 逐层显式声明时, 校验每一项必须为 full / shared
        if len(indexer_types) != num_hidden_layers:
            raise ValueError(
                'indexer_types 必须为每层提供一项: '
                f'期望 {num_hidden_layers} 项, 实际 {len(indexer_types)} 项。')
        invalid_types = sorted(set(indexer_types) - {'full', 'shared'})
        if invalid_types:
            raise ValueError(
                f'indexer_types 仅支持 full 与 shared, 得到 {invalid_types}。')
        return {
            idx for idx, t in enumerate(indexer_types) if t == 'shared'
        }

    # 没有显式声明时, 按 index_topk_freq / pattern 推导
    freq = getattr(config, 'index_topk_freq', 1)
    if not isinstance(freq, int) or freq <= 0:
        raise ValueError(f'index_topk_freq 必须为正整数, 得到 {freq!r}。')
```

```

pattern = getattr(config, 'index_topk_pattern', None)
offset = getattr(config, 'index_skip_topk_offset', 2)
skip_layers: set[int] = set()
for layer_idx in range(num_hidden_layers):
    if pattern is None:
        # 按频率：除首个 full 层外，其余层共享最近的 top-k 索引
        if max(layer_idx - offset + 1, 0) % freq != 0:
            skip_layers.add(layer_idx)
    elif 0 <= layer_idx < len(pattern) and pattern[layer_idx] == 'S':
        # 显式 pattern 中标记为 S 的层跳过索引器构建
        skip_layers.add(layer_idx)
return skip_layers

```

vllm/models/hy_v4/nvidia/flashmla_sparse.py

为 FLASHMLA_SPARSE 后端补上 Hy4 的 per-head learnable sink 透传，避免注意力偏置被静默丢弃。

HYV4FlashMLASparseImpl 的核心部分：验证并保存每头可学习 attention sink。
 # FlashMLA 内核会把 sink 折进 softmax 分母，所以形状与类型必须严格校验。

```

class HYV4FlashMLASparseImpl(FlashMLASparseImpl):
    def __init__(self, num_heads, head_size, scale, num_kv_heads,
                  alibi_slopes, sliding_window, kv_cache_dtype,
                  logits_soft_cap, attn_type, kv_sharing_target_layer_name,
                  topk_indices_buffer=None, indexer=None, **mla_args):
        # SparseMLACommonImpl 只接受显式关键字参数，所以要在父类
        # 看到 mla_args 之前把 sink 弹出来
        sinks = mla_args.pop('sinks', None)
        super().__init__(
            num_heads, head_size, scale, num_kv_heads, alibi_slopes,
            sliding_window, kv_cache_dtype, logits_soft_cap, attn_type,
            kv_sharing_target_layer_name,
            topk_indices_buffer=topk_indices_buffer,
            indexer=indexer, **mla_args)
        self._validate_sinks(sinks, num_heads)
        self.sinks = sinks

```

@staticmethod

```

def _validate_sinks(sinks, num_heads):
    # None 表示该层没有 sink，直接放行
    if sinks is None:
        return
    # 内核要求 float32 且形状为 (num_heads,)，否则给出明确报错
    if sinks.dtype != torch.float32:
        raise ValueError(
            'HYV4 FlashMLA sparse attention sinks must have dtype '
            'torch.float32, but got %s.' % sinks.dtype)
    if sinks.ndim != 1 or sinks.shape[0] != num_heads:
        raise ValueError(

```

```
'HYV4 FlashMLA sparse attention sinks must have shape '  
'(%d.), but got %s.' % (num_heads, tuple(sinks.shape)))
```

vllm/model_executor/layers/hpc/hpc_ihc.py

单内核融合 iHC 替代 eager 路径的 20/5/15 次 kernel launch，门控逻辑严谨，是性能路径的关键。

```
# SPDX-License-Identifier: Apache-2.0  
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project  
# HPC 融合 iHC (independent Hyper-Connections) 内核，用于 HY V4。  
# 它把 eager 路径中 pre / post / head 的 20 / 5 / 15 次 kernel launch  
# 各自压成 1 次，并且只在满足全部约束时才启用，否则回退 eager。  
  
import torch  
  
from vllm import envs  
from vllm.logger import init_logger  
from vllm.model_executor.layers.hpc.hpc_module import HpcModule  
from vllm.utils.hpc import has_hpc  
  
logger = init_logger(__name__)  
  
# 内核只针对这些形状实例化，其他形状不会启动，因此这里直接做门控  
# ，避免依赖内核内部静默跳过。  
_SUPPORTED_HIDDEN_SIZES = frozenset({4096, 6144})  
_SUPPORTED_HC_MULTS = frozenset({4})  
_SUPPORTED_CAPABILITIES = frozenset({100, 103})  
  
def _ihc_supported(hc_mult: int, hidden_size: int) -> bool:  
    # 1) 全局开关：未设置 VLLM_ENABLE_HPC_OPS 直接关闭  
    if not envs.VLLM_ENABLE_HPC_OPS:  
        return False  
  
    # 2) hpc 包 (.so) 未安装时给出一次性告警，而不是静默跳过  
    if not has_hpc():  
        logger.warning_once(  
            'HPC iHC disabled: hpc package is not installed.')  
        return False  
  
    from vllm.platforms import current_platform  
  
    # 3) 仅支持 CUDA，且计算能力必须是 sm100 / sm103  
    if not current_platform.is_cuda():  
        logger.warning_once('HPC iHC disabled: only CUDA is supported.')  
        return False  
    capability = current_platform.get_device_capability()  
    if capability is None or capability.to_int() not in _SUPPORTED_CAPABILITIES:  
        logger.warning_once(  

```

```

        'HPC iHC disabled: compute capability %s not in %s.',
        capability, _SUPPORTED_CAPABILITIES)
    return False

# 4) 形状约束: hc_mult 必须为 4, hidden_size 必须为 4096 或 6144
if hc_mult not in _SUPPORTED_HC_MULTS:
    logger.warning_once(
        'HPC iHC disabled: hc_mult=%d not in %s.',
        hc_mult, _SUPPORTED_HC_MULTS)
    return False
if hidden_size not in _SUPPORTED_HIDDEN_SIZES:
    logger.warning_once(
        'HPC iHC disabled: hidden_size=%d not in %s.',
        hidden_size, _SUPPORTED_HIDDEN_SIZES)
    return False

# 5) VLLM_BATCH_INVARIANT 模式依赖 torch 确定性规约, HPC kernel 规约
# 次序不同, 两种模式必须隔开
if envs.VLLM_BATCH_INVARIANT:
    logger.warning_once(
        'HPC iHC disabled: VLLM_BATCH_INVARIANT set.')
    return False

logger.info_once('HPC iHC enabled.')
return True

```

评论区精华

唯一实质 review 讨论来自 oops-oom 的提问: `is_deepseek_mla()` 白名单 (`vllm/transformers_utils/model_arch_config_convertor.py:314`) 未注册 `hy_v4`, 而 `Hy4-preview` 的 config 是 `use_mla: true + kv_lora_rank: 512`, 与已列出的 `deepseek_v32 / glm_moe_dsa` 同构。若不注册, DCP、PCP、disagg connectors 与 `nvfp4-KV-MLA guard` 等读取 `model_config` 的路径会看到 `use_mla=False`; 主推理因 `HYV4MLAAttention` 直接传 `head_size=576` 而不受影响。该问题在后续提交 `b22b0ea44` (`is_deepseek_mla add hy_v4`) 中修复, 白名单已补齐。其余评论主要是 `mergify` 冲突提示与多次 `/ci run` 触发。

- `is_deepseek_mla` 白名单遗漏 `hy_v4` (correctness): PR 后续提交 `b22b0ea44` (`is_deepseek_mla add hy_v4`) 已在白名单中补上 `hy_v4`, 问题修复。

风险与影响

- 风险:
 1. MTP 权重映射复杂: `mtp.py` 中多个 helper 针对 `FP8/MXFP8 scale` 参数做容错 (跳过缺失 `scale`、重映射 `quant exclusion`、解析 `fused expert` 参数), 不同导出版本的 `checkpoint` 差异可能导致权重加载失败或静默丢参。
 2. HPC 路径受限: `hpc_ihc.py` 仅支持 `sm100/sm103`、`hc_mult==4`、`hidden_size` `4096/6144`, 且依赖 `VLLM_ENABLE_HPC_OPS=1` 与 `hpc .so`; 约束不满足时回退

eager, 但回退路径与融合路径数值行为可能存在微小差异。

3. 配置白名单曾遗漏: is_deepseek_mla 未及时注册 hy_v4, 会影响所有读取 model_config.use_mla 的分布式缓存与连接器路径; 虽然已修复, 但说明新模型接入时配置注册点容易被遗漏。
4. Rust 解析器重命名: hy_v3.rs 改为 hy.rs 并抽象 HyInput/HyMode, 涉及既有 Hy V3 解析行为, 回归测试依赖需要覆盖旧模型工具调用。
5. 测试覆盖: PR 描述中 Test Plan 与 Test Result 为空, 端到端推理 (FP8 + MTP + FLASHMLA_SPARSE) 主要靠 CI 验证, 模型侧回归风险相对较高。 - 影响: 用户侧: 新增可直接通过 vLLM 使用的 770B 级 MoE 模型, 支持 FP8 权重、MTP 投机解码、工具调用 (hy_v4 parser) 与 reasoning_effort 控制; 部署命令涉及 tensor-parallel-size 8、FLASHMLA_SPARSE 后端与 VLLM_ENABLE_HPC_OPS 环境变量。系统侧: 新增 HPC 融合内核路径与稀疏 MLA sink 机制, FlashMLA 稀疏注意力后端的可复用能力增强; 团队侧: hy_v4 相关解析器、模型代码规模大 (+6260 行), 后续维护与兼容性跟进负担明显。 - 风险标记: 新模型大范围集成, MTP 权重映射复杂, HPC 路径仅 sm100/103, 配置白名单曾遗漏, 端到端测试结果缺失

关联脉络

- PR #54295 [Perf][MLA Sparse] Pin req_id_per_token before non_blocking H2D on XPU and ROCm: 同属 MLA 稀疏注意力路径, Hy4 大量依赖 FLASHMLA_SPARSE 后端, 两者在后端元数据路径上有共同演进方向。
- PR #54277 [Attention][DCP] Enable FlashInfer MLA for DSpark drafting: 同属 MLA 注意力后端增强与投机解码, Hy4 的 MTP 与稀疏 MLA 与该 PR 共享 MLA 后端基础设施。
- PR #48584 [Rust Frontend] Add support for truncate_prompt_tokens and truncation_side: 同属 rust/src/parser 前端解析器演进, 本 PR 将 hy_v3.rs 重命名为 hy.rs 并扩展 V4 模式, 与 Rust 文本解析能力的持续增强相关。
- PR #54239 [Model] Support speculative decoding method for PLaMo3: 同为新模型接入原生 MTP 投机解码, 可对比不同模型的 MTP 集成方式与量化映射。