

PR #48841 完整报告

vllm-project/vllm

[ROCm] [Model] Enable TML inkling

合并时间: 2026-07-27 18:05

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/48841>

执行摘要

- 一句话: 为 ROCm 启用 Inkling 模型, 支持 MXFP4 推理
- 推荐动作: 建议技术管理者仔细审阅该 PR, 它展示了 vLLM 如何基于已有基础设施 (FusedMoE、AttentionLayerBase) 快速适配新架构, 以及在保持 NVIDIA 兼容的同时加入 ROCm 优化。开发者应重点关注 InklingMoE 中 MXFP4 权重的加载与门控逻辑, 以及 `_sconv_add_norm` 中 ROCm 特定的集体通信模式, 这些设计可在未来类似移植中复用。

功能与动机

PR 描述明确指出目的为 'first enablement PR for inkling model on ROCm', 并 'targets the more efficient variant of the model which is mxfp4', 权重由 HuggingFace 仓库 `lightseekorg/Inkling-MXFP4` 提供。该 PR 避免了低效的 NVFP4 仿真路径, 为用户提供可直接用于推理的 MXFP4 检查点。

实现拆解

1. 新增AMD特定模型模块: 在 `vllm/models/inkling/amd/` 下创建 `model.py` (InklingModel、InklingDecoderLayer)、`attention.py` (InklingAttention)、`moe.py` (InklingMoE)、`mtp.py` (InklingMTP)、`logits_processor.py` (InklingLogitsProcessor)、`short_conv.py` (InklingShortConv)、`sconv_swa_attn.py` (InklingSconvBackend) 等文件, 完整定义 ROCm 平台的模型前向逻辑、权重加载和量化配置。
2. 集成高性能解码内核: 将 LightSeek `tokenspeed` 的 `GFX950 GLUON` 相对注意力内核 (`rel_mha_decode_gfx950.py`) 直接纳入仓库, 避免外部依赖冲突和版本兼容问题。该内核针对 `paged attention` 优化, 在 `decode` 阶段提供约 23% 的 TPOT 改善。
3. 实现 MXFP4 / bf16 混合 MoE: InklingMoE 继承自 FusedMoE, 支持 `sigmoid` 门控和 `log-sigmoid` 重归一化; 量化层使用 vLLM 的 Quark OCP MXFP4 融合方法, 非量化层回退至 bf16; 通过重写 `load_expert_weight` 方法, 将检查点中融合的堆叠张量转换为标准单专家格式。
4. 支持多 token 预测与工具调用: InklingMTP 实现共享词表嵌入的 `draft` 模型, 通过 `embed_dual_rmsnorm_cat` 融合归一化输入; InklingLogitsProcessor 应用 `muP` 宽度缩放, 并兼容 LoRA 适配器; Rust 前端和 `tokenizer` 的更新 (在独立提交中) 支持了 `inkling` 特定的 `tokenizer` 和工具调用解析。

5. 测试与性能基准：在 tests/models/inkling/rocm/ 下新增数值等价性测试（相对注意力、MTP 输入融合）和性能基准（benchmark_inkling_qkvr_prep.py），验证计算正确性和性能优势；测试通过 conftest 限定仅在 gfx950 上运行。

关键文件：

- vllm/models/inkling/amd/model.py（模块 模型定义；类别 source；类型 core-logic；符号 _layer_id, _sconv_add_norm, InklingDecoderLayer, init）：Inkling 模型的主入口，定义了 InklingModel、InklingDecoderLayer 及其前向和权重加载流程，是 PR 的核心架构文件。
- vllm/models/inkling/amd/attention.py（模块 注意力层；类别 source；类型 core-logic；符号 compute_log_scaling_tau, RelLogitsProj, init, forward）：定义了 InklingAttention 层，包括 QKV 融合投影、相对位置 logits 缩放、以及 GLUON 注意力后端集成，是性能关键模块。
- vllm/models/inkling/amd/moe.py（模块 MoE 门控；类别 source；类型 core-logic；符号 _linear_with_fp32_out, _inkling_gate_select_kernel, inkling_gate_select, InklingGate）：实现 Inkling 的 MoE 门控与专家选择，支持 MXFP4/bf16 混合精度，是模型计算量最大的部分。
- vllm/models/inkling/amd/ops/gluon/rel_mha_decode_gfx950.py（模块 解码内核；类别 source；类型 core-logic；符号 rel_mha_decode_gfx950, use_gfx950_gluon_decode, use_gfx950_gluon_ext）：LightSeek tokenspeed 的 GLUON 注意力内核，针对 GFX950 paged decode 优化，是性能提升的核心。
- tests/models/inkling/rocm/test_rel_attention.py（模块 注意力测试；类别 test；类型 test-coverage；符号 _reference, test_query_length_bucket, test_split_kv_dispatch_thresholds, test_gfx950_gluon_dispatch_is_strict）：覆盖相对注意力正确性、分桶逻辑和 GLUON 后端 dispatch，是模型核心路径的重要测试。
- vllm/models/inkling/amd/sconv_swa_attn.py（模块 短卷积缓存；类别 source；类型 core-logic；符号 InklingSconvMetadata, InklingSconvMetadataBuilder, init, build）：实现短卷积 SWA 缓存管理，定义 InklingSconvBackend 作为 AttentionBackend，是 conv state 和 paged cache 的核心。

关键符号：InklingModel.forward, InklingDecoderLayer.forward, InklingAttention.forward, InklingMoE.select_experts, InklingLogitsProcessor.forward, compute_log_scaling_tau, RelLogitsProj.forward, _sconv_add_norm, InklingMTP.forward, _linear_with_fp32_out

关键源码片段

vllm/models/inkling/amd/model.py

Inkling 模型的主入口，定义了 InklingModel、InklingDecoderLayer 及其前向和权重加载流程，是 PR 的核心架构文件。

```
def _sconv_add_norm(
    delta: torch.Tensor,
    hidden: torch.Tensor,
    sconv: InklingShortConv,
    norm: InklingRMSNorm | None,
```

```

positions: torch.Tensor,
) -> tuple[torch.Tensor | None, torch.Tensor]:
    # ROCm 使用 RCCL 的 reduce_scatter 和 all_gather 实现跨 rank 集体通信,
    # 而不是 NVIDIA 实现中的 Lamport P2P + CUDA GDC 路径 (GFX950 不支持 HSACO)
    norm_w = norm.weight if norm is not None else None
    eps = norm.variance_epsilon if norm is not None else 0.0

    # RCCL RS -> shard sconv -> AG -> fused add(+rmsnorm)
    shard = tensor_model_parallel_reduce_scatter(delta, dim=-1)
    shard = sconv(shard.contiguous(), positions)
    full = tensor_model_parallel_all_gather(shard, dim=-1)
    if norm is None:
        return None, hidden + full
    return add_rmsnorm(hidden, full, norm_w, eps)

```

vllm/models/inkling/amd/moe.py

实现 Inkling 的 MoE 门控与专家选择，支持 MXFP4/bf16 混合精度，是模型计算量最大的部分。

```

def _linear_with_fp32_out(x: torch.Tensor, weight: torch.Tensor) -> torch.Tensor:
    leading = list(x.shape[:-1])
    flat = x.flatten(0, -2)
    # 当 token 数量 <= 64 且满足特定硬件条件时，使用 LL bf16 gemm 加速；
    # 否则回退到 fp32 epilogue 的 torch.mm 保证数值精度
    if (
        flat.shape[0] <= _INKLING_LL_BF16_MAX_TOKENS
        and flat.dtype == torch.bfloat16
        and weight.dtype == torch.bfloat16
        and flat.is_cuda
        and flat.is_contiguous()
        and weight.is_contiguous()
        and flat.shape[1] % 8 == 0
        and current_platform.has_device_capability(90)
        and ll_bf16.is_available()
    ):
        out = ll_bf16.ll_bf16_gemm(flat, weight)
    else:
        out = torch.mm(flat, weight.T, out_dtype=torch.float32)
    return out.view(*leading, weight.shape[0])

```

评论区精华

审查中主要讨论了三点：

- 测试平台限定：AndreasKaratzas 建议将 ROCm 测试限定在 gfx950，tjtanaa 通过 conftest 实现非 ROCm/ 非 gfx950 跳过，最终批准。
- 空张量初始风险：AndreasKaratzas 指出 torch.empty 在 ROCm 上可能引起问题，建议 torch.zeros；tjtanaa 解释内核来自 tokenspeed 仓库且已充分验证，保持现状。

- 硬件类型检查: AndreasKaratzas 建议使用 `q.dtype is current_platform.fp8_dtype()`, `tjtanaa` 表示保持 `tokenspeed` 原样以避免平台抽象不一致。
- 测试平台限定方式 (testing): 通过 `conftest` 的基于 `current_platform.is_rocm()` 和 `on_gfx950()` 的条件跳过实现。
- 空张量初始化风险 (correctness): 保持 `torch.empty` 不变, 确认已验证。
- 设备数据类型检查 (style): 保持原始硬编码方式, 因为内核专为 `gfx950` 编写。

风险与影响

- 风险:
 - 外部内核依赖: `rel_mha_decode_gfx950.py` 直接复制自 `tokenspeed`, 未来需要与上游同步以避免 bug 或性能退化, 缺乏版本锁定机制。
 - 平台锁定: `GFX950` 专用路径, 非该硬件上测试已跳过, 但用户在不支持的 GPU 上运行会导致运行时失败。
 - 量化格式绑定: 仅支持特定的 `MXFP4` 检查点格式, `NVFP4` 回退路径为低效仿真, 未提供完整的 `bf16` 或 `fp16` 支持。
 - 大量新增代码 (约 9k 行) 引入长期维护成本, 团队需完整理解 `Inkling` 模型架构; 当前未在非 `ROCm` 平台上进行验证。
- 影响:
 - 用户: 在 `AMD GFX950` 上可运行 `LightSeek Inkling` 模型, 推理吞吐量提升约 29%, `TPOT` 降低约 23%, 且支持 `MTP` 和工具调用。
 - 系统: 新增约 33 个文件, 影响范围严格限于 `inkling` 模型路径, 对系统中其他模型无影响。
 - 团队: 需承担新模型系列的维护责任, 包括定期同步 `tokenspeed` 内核更新、确保与 `NVIDIA` 实现一致性, 并为不支持的硬件提供更清晰的错误提示。
 - 风险标记: 外部内核依赖, 平台锁定 (`GFX950`), 量化格式绑定, 大量新增代码维护成本

关联脉络

- PR #48775 [Inkling] Use `<lunused_200053l>/<lunused_200054l>` for image/audio placeholder token ids: 同一 `Inkling` 模型支持系列中的 `tokenizer` 变更, 与 PR 中 `Rust` 前端和 `tokenizer` 修改相关, 在 `commit` 历史中有交互