

# PR #46832 完整报告

vllm-project/vllm

[ROCm][DSv3.2][Perf] Cap sparse MLA decode KV-splits with a work-per-split heuristic

合并时间: 2026-07-19 00:39

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/46832>

## 执行摘要

- 一句话: Cap sparse MLA decode KV-splits with heuristic on ROCm
- 推荐动作: 该 PR 值得阅读, 展示了如何通过简单的启发式避免 kernel 过度分割, 提升性能。设计思路清晰, 实现简洁, 适合作为类似性能优化的参考。

## 功能与动机

ROCm aiter 稀疏 MLA 解码后端调用 `get_mla_metadata_v1` 时未传入 `max_split_per_batch`, aiter 默认使用 -1 将 reduce 分割到所有 CU (gfx942 上 304 个)。由于 reduce 仅覆盖每行选中的 tokens ( $\leq \text{index\_topk}=2048$ ), 每个 split 仅处理约 7 个 tokens, reduce kernel 被启动和合并开销主导。PR 主体明确指出了这一性能瓶颈。

## 实现拆解

1. 在 `__init__` 中获取计算单元数量: 新增一行 `self._num_compute_units = current_platform.num_compute_units()`, 为后续启发式计算提供 CU 上限。
2. 新增 `_sparse_decode_max_split` 方法: 该方法接收 `max_seq_len`, 取 `min(max_seq_len, self.topk_tokens)` 作为有效长度, 设定最小每个 split 的工作量 128 tokens, 计算理想 split 数 `triton.next_power_of_2(max(1, effective_len // 128))`, 并用 CU 数量作为上限。
3. 在 `build` 方法中调用启发式: 在调用 `get_mla_metadata_v1` 前计算 `max_split_per_batch` 并作为参数传入, 替代之前隐式默认值 -1。

关键文件:

- `vllm/v1/attention/backends/mla/rocm_aiter_mla_sparse.py` (模块 注意力; 类别 source; 类型 core-logic; 符号 `_sparse_decode_max_split`): 核心变更文件: 新增 `_sparse_decode_max_split` 启发式方法, 在 `__init__` 中缓存计算单元数量, 在 `build` 中计算并传递 `max_split_per_batch`, 解决了 ROCm 上稀疏 MLA decode reduce kernel 的过度分割问题。

关键符号: `_sparse_decode_max_split`, `build`

## 关键源码片段

[vllm/v1/attention/backends/mla/rocm\\_aiter\\_mla\\_sparse.py](#)

核心变更文件：新增 `_sparse_decode_max_split` 启发式方法，在 `__init__` 中缓存计算单元数量，在 `build` 中计算并传递 `max_split_per_batch`，解决了 ROCm 上稀疏 MLA decode reduce kernel 的过度分割问题。

```
# vllm/v1/attention/backends/mla/rocm_aiter_mla_sparse.py
# 在 __init__ 中新增一行，缓存计算单元数量供后续启发式使用
# Bounds the KV-split heuristic (see `_sparse_decode_max_split`).
self._num_compute_units = current_platform.num_compute_units()

# 新增启发式方法：计算 max_split_per_batch
# 避免 aiter 默认在所有 CU 上过度分割 reduce kernel
def _sparse_decode_max_split(self, max_seq_len: int) -> int:
    """Cap ``max_split_per_batch`` for the aiter sparse-MLA decode reduce.

    The reduce only covers the selected tokens per row (``<= topk_tokens``),
    so aiter's default (``-1`` => split across every CU) over-fragments it.
    Mirror ``triton_mla.py``: aim for a minimum work per split, round to a
    power of two, and cap by the CU count. Numerics are unchanged.
    """
    # 有效长度不超过 topk_tokens
    effective_len = min(max_seq_len, self.topk_tokens)
    # 每个 split 最少处理 128 tokens
    min_work_per_split = 128
    # 向上取整到 2 的幂
    ideal_splits = triton.next_power_of_2(
        max(1, effective_len // min_work_per_split)
    )
    return min(ideal_splits, self._num_compute_units)

# 在 build 方法中，调用 get_mla_metadata_v1 前计算 max_split_per_batch
# 替代之前的默认行为（即不传参，aiter 使用 -1）
max_split_per_batch = self._sparse_decode_max_split(
    int(common_attn_metadata.max_seq_len)
)
get_mla_metadata_v1(
    qo_indptr,
    paged_kv_indptr,
    ...
    max_split_per_batch=max_split_per_batch,
)
```

## 评论区精华

tjtanaa 询问了使用的 aiter 版本。jashwanth-reddy-g 批准了 PR，指出启发式方法很好地镜像了 `triton_mla.py` 的实现，正确限制了 `max_split_per_batch`。

- aiter 版本确认 (question): 未在 review 线程中看到直接回复，但 PR 被合并表明版本兼容。

## 风险与影响

- 风险：该 PR 变更风险较低：仅修改单个文件，逻辑可逆，数值结果通过 LSE 重组合保证不变。但启发式中的硬编码常量 128 可能在不同模型或配置下不是最优，后续可能需要调优或配置化。另外，未添加直接单元测试覆盖新函数。
- 影响：仅影响 ROCm 平台上的稀疏 MLA decode 路径，对 NVIDIA 或其他后端无影响。在低并发、长上下文场景下收益显著（reduce kernel 耗时降低 75.6%），高并发时由于 aiter 的设备上限机制会自动回退到全 CU 行为，因此无负面影响。端到端吞吐提升 4.3%，数值准确性不受影响。
- 风险标记：缺少单元测试覆盖，硬编码常量可能需调优

## 关联脉络

- PR #48788 [ROCm][Perf][DSV4] Improve sparse decode reduction occupancy on gfx950: 同一团队针对 ROCm 和稀疏 MLA decode 的持续性能优化，且共享类似的 kernel 优化关注点。
- PR #48660 [Perf] Optimize dsv4 routing using specialized kernel, 2.94% E2E TPOT improvement: 同为 DeepSeek 模型性能优化，但针对不同的 kernel（路由 vs. 注意力），展示整体优化方向。