PR #33529 完整报告

vllm-project/vllm

Triton MLA perf fixes

合并时间：2026-04-02 21:40

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/33529

执行摘要

本 PR 通过动态调整 Triton MLA 的 KV 分割数和优化内核内存访问，修复了在 sm120 架构上长上下文下性能下降的问题，使 Deepseek 和 Kimi 模型在 80K tokens 上下文中的吞吐量提升超过 50%，显著改善了大上下文推理的可用性。

功能与动机

Triton MLA 在批量大小为 1 且上下文长度增加时性能显著下降，导致 Deepseek 和 Kimi k2 模型无法有效使用。作者在 Kimi k2.5 发布后深入调查，发现核心问题是低批量数下 KV 分割不合理，导致 SM 未充分利用和多余的 Q 向量加载。PR body 中明确表示："This perf issue has been bugging me for a while as it made deepseek and Kimi k2 unusable"。

实现拆解

主要改动集中在两个文件：

- vllm/v1/attention/backends/mla/triton_mla.py：将固定的 num_kv_splits（原为 1 或 4）改为动态计算。基于 SM 数量、序列长度和最小工作单元，使用 triton.next_power_of_2 优化分割数，代码片段：python ideal_splits = max(1, attn_metadata.max_seq_len // min_work_per_split) ideal_splits = triton.next_power_of_2(ideal_splits) max_splits = self._sm_count * occupancy_multiplier num_kv_splits = min(ideal_splits, max_splits)
- vllm/v1/attention/ops/triton_decode_attention.py：优化 Triton 内核，添加缓存修饰符（如 .ca 和 .cg），使用 tl.range 替换 range，并显式提前加载 KV 地址以重叠计算，提升内存访问效率。

评论区精华

review 中关键讨论点：

- 动态分割计算：mgoin 指出"This should be accessed and saved outside of forward path"，作者随后调整为预计算 SM 计数；tjtanaa 建议使用 get_cu_count 辅助函数，作者采纳。
- CUDA 图支持：tjtanaa 提醒"DCP and PCP is not graph compatible"，作者测试后发现性能下降，最终移除了 CUDA 图部分，并在评论中说明："I backed out that part of my change."

- 准确性验证：mgoin 要求“Can you run an accuracy evaluation? Just a simple gsm8k is good”，但后续未在材料中看到明确结果，这可能是一个未解决点。

风险与影响

- 技术风险：动态分割启发式依赖硬件参数（如 SM 数量），在不同 GPU 上可能表现不一致；缓存修饰符的使用可能影响跨平台兼容性；移除 CUDA 图支持可能影响依赖图优化的配置。
- 影响评估：对用户，显著提升长上下文推理性能，测试显示 80K tokens 下吞吐量从 34 tok/s 增至 52 tok/s（+52.94%）；对系统，优化资源利用率；对团队，提供了内核优化案例，但需加强测试覆盖。

关联脉络

从历史 PR 看，本 PR 与 MLA 后端优化相关，如 PR 38615 修复了 ROCm MLA 的类似问题。结合近期 PR 趋势，vLLM 项目持续关注性能优化和模型支持，本 PR 是这一方向的具体实践，强调了硬件自适应和内核微调的重要性。