

PR #56346 完整报告

vllm-project/vllm

[Perf][Kernel] Add sampled filtering for persistent top-k

合并时间: 2026-09-16 05:16

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56346>

执行摘要

本 PR 为 DeepSeek V4 稀疏索引器的 FP32 长行 top-k 引入 " 采样过滤 " 路径: 受 DeepSelect 启发, 先对每行均匀采样估计 cutoff, 再把超过 cutoff 的幸存者压缩进共享内存做精确 FP32 选择, 候选过少或 stash 溢出时回退全行精确选择。B300 上 B=1024、80%~100% ragged 长度下, warm latency 从 584.45 → 301.72 μ s (256K/k=512)、316.58 → 207.06 μ s (128K/k=2048)。启用门限保守 (行数 >64、行长 $\geq$ 98304 或 $\geq$ 65536、 $\geq$ 144 KiB opt-in 共享内存), 短行与旧硬件完全走原路径; 正确性由双回退兜底, 并有 6265 例 forced-backend 审计零错误支撑。需注意: 性能验证仅限 B300, 且 fork 提交无人工 review, 合入判断主要依赖作者自证数据。

功能与动机

PR body 开门见山: "Speed up long FP32 sparse-indexer decode top-k with sampling inspired by DeepSelect"。长上下文下稀疏 MLA 索引器的 decode top-k 需要对每行数万至数十万 FP32 score 做精确选择, 批量大且行长 ragged 时成为延迟热点。作者对 B300 做了 batch、长度、seed、cache mode 全参数扫描, 得到 crossover envelope, 并发现 k=512 在 80K~86K 长度区间存在减速, 因此启用阈值取保守值 (k=512 为 98304, 其余为 65536), 既保证收益又避开退化区间。作者也主动声明: 短行回退最多增加 3.32% 延迟 (绝对 1.215 μ s), 并确认与 #44606、#55872、#55314 三个已存在 / 已关闭 PR 不重复。

实现拆解

1. 新增采样过滤内核 `csrc/libtorch_stable/sampled_topk.cuh` (新增 253 行): 在 `vllm::sampled_topk` 命名空间实现 `sampled_topk_kernel<K>`。Storage 用动态共享内存承载 8192 个 `uint2` 幸存者候选、2048 桶 coarse 直方图与 `cub::BlockScan` 临时存储; `ordered()` 把 FP32 位模式重排为 `uint32` 全序 key, `select_bin()` 用 InclusiveScan 定位目标分箱, `threshold()` 先 coarse 分桶再逐字节细化出精确 cutoff。候选压缩阶段按 warp 做 `__ballot_sync` 命中掩码、warp 内前缀和、原子预留槽位; 幸存者数超过 `kCapacity` 时继续计数而不截断, 从而触发溢出回退。行长低于 `kMinSampledLength<K>` 的行复用原有 `filtered_topk_row` 精确路径, 单一 launch 内长短行共存。
2. 重构既有筛选内核 `csrc/libtorch_stable/persistent_topk.cuh` (+52/-34): 把 `FilteredTopKUnifiedKernel` 的静态共享内存变量 (直方图、计数器、阈值分箱、索引 stash) 提取为 `FilteredTopKStorage<MAX_K>` 结构体, 内核主体抽成 `__device__ filtered_topk_row<DType, IdType, VEC_SIZE, MAX_K, UsePredicatedShortLoads, CheckOverflow>` 设备函数。新增 `CheckOverflow` 模板参数

: 细化阶段若候选数超过 `SMEM_INPUT_SIZE` 则返回 `false`, 让调用方改用全行精确选择, 避免旧实现 "用截断 stash 子集" 造成的精度偏差。

3. 调度接线 `csrc/libtorch_stable/topk.cu` (+17/-1) : `launch_persistent_topk` 在原有 `num_rows > 32 && max_smem_per_block >= 128 KiB` 分支前插入新分支: `num_rows > 64 && max_seq_len >= kMinSampledLength<TopK> && max_smem_per_block >= 144 KiB` 时走 `sampled_topk_kernel`, 并调用 `cudaFuncSetAttribute` 设置 128 KiB 动态共享内存上限; 条件不满足完全退回原路径。 `#include "sampled_topk.cuh"` 位于 `#ifndef USE_ROCM` 内, ROCm 构建不受影响。
4. 图回放正确性测试 `tests/kernels/test_top_k_per_row.py` (+100) : 新增 `test_persistent_topk_sampled_graph`, 参数化 `rows=[65, 128] × top_k=[512, 1024, 2048] × distribution=[random, 10LSBits, ascending, constant, sampled_peaks]`。行宽取 `kMinSampledLength + 3` 贴合启用边界; `sampled_peaks` 分布刻意让采样落空 (幸存者 < k) 以覆盖 underflow 回退; CUDA graph replay 3 轮中把 `lengths` 逐个替换为 `[-1, 0, 1, k-1, k, 32769, 门限±1, 行宽, 行宽 +17]` 等边界值并翻转一次 logits, 与精确 `topk` 做 `atol=0, rtol=0` 严格对照。共享内存不足 144 KiB 的卡直接 skip。
5. 基准脚本 `benchmarks/kernels/benchmark_persistent_topk.py` (新增 134 行) : 独立 CUPTI 基准, 支持 `rows × widths × top_k` 全组合、`--capacity` 模拟更大 graph 捕获上界、`--full-length` 与 80%~100% ragged 长度、warm/cold L2 状态, 输出 JSON 便于 base/candidate 两侧对比; 内部自带完整正确性断言 (无效位 -1、无重复索引、与参考 `topk` 严格相等)。

`csrc/libtorch_stable/topk.cu`

`launch_persistent_topk` 调度入口新增采样分支, 串起启用门限 (行数、行长、共享内存) 与内核选择, 决定新旧路径边界

```
// topk.cu: launch_persistent_topk 中新增的采样分支。
// 只有 batch > 64、最大行长达标、且可申请 144 KiB opt-in 共享内存时才启用,
// 其余情况完全沿用原有 cooperative / persistent 路径, 保证短行与旧卡无回归。
if (num_rows > 64 &&
    max_seq_len >= vllm::sampled_topk::kMinSampledLength<TopK> &&
    max_smem_per_block >= 144 * 1024) {
    auto kernel = vllm::sampled_topk::sampled_topk_kernel<TopK>;
    // 128 KiB 动态共享内存槽位, 采样内核复用为 Storage 幸存者缓冲区
    constexpr size_t smem_size = vllm::filtered_topk::FILTERED_TOPK_SMEM_DYNAMIC;
    cudaError_t status = cudaFuncSetAttribute(
        kernel, cudaFuncAttributeMaxDynamicSharedMemorySize, smem_size);
    STD_TORCH_CHECK(status == cudaSuccess,
        "sampled_topk smem failed: ", cudaGetErrorString(status));
    kernel<<<num_rows, vllm::sampled_topk::kThreads, smem_size, stream>>>(
        logits.const_data_ptr<float>(), lengths.const_data_ptr<int32_t>(),
        output.mutable_data_ptr<int32_t>(), stride,
        static_cast<int>(std::min(max_seq_len, logits.size(1))));
} else if (num_rows > 32 && max_smem_per_block >= 128 * 1024) {
    // 原有 FilteredTopKRaggedTransform 分支保持不变
}
```

评论区精华

本 PR 来自 fork 分支，claude[bot] 自动评审被禁用（原文："This pull request is from a fork — automated review is disabled. A repository maintainer can comment @claude review to run a one-time review."），且维护者未触发一次性 review，因此没有任何人工评审线程。评审价值主要来自作者在 PR body 中预埋的自证证据：

B300 全参数 crossover 扫描与 envelope 图；6265 例 forced-backend + 541 例 final-dispatch 零选择错误；memcheck/synccheck 零错误；DeepSeek-V4-Flash-0731 TP2 128 请求端到端 FP32 审计，128/128 算术正确、GSM8K 123/128。

作者也主动披露局限：short-row fallback 最多 +3.32%（绝对 1.215 μ s）、k=512 在 80K~86K 有减速区间、token 等价性未建立（75/128 长生成匹配 EOS）、性能验证仅限 B300。这些自证数据替代了常规 review 讨论，但缺少独立评审理角的挑战，是合入决策的主要不确定性来源。

风险与影响

风险：

1. 验证面窄：性能结论仅基于 B300；其他 SM90 系卡即使满足启用条件也会走采样路径，收益 / 回归未被实测，SM103a/SM80 仅编译通过。
2. 启发式正确性依赖回退兜底：采样 cutoff 是统计估计，测试构造了 5 种分布（含故意漏采样的 sampled_peaks）并以 atol=0, rtol=0 严格对照，但真实 logits 的极端长尾分布仍可能暴露未覆盖边界。
3. 性能盲区：k=512 在 80K~86K 区间存在减速，cutoff 取 98304 只是避开而非消除；短行回退最多 +3.32%。
4. 运维健壮性：cudaFuncSetAttribute 失败会直接 STD_TORCH_CHECK 崩溃，是新增的 panic 点；144 KiB opt-in 共享内存门槛使不达标硬件静默走旧路径，新路径硬件覆盖靠运行时属性保证。
5. 测试契约细节：rows=128 时 lengths 以 view(-1, 4) 传入，暗示 TMA 对齐约束，未来其他调用方需注意 shape 契约。

影响：

- 用户侧：DeepSeek V4 长上下文稀疏索引 decode (batch ≥ 64 、行长 $\geq 98K$ 或 $\geq 65K$ token) 延迟显著下降约 35%~48%；短行与旧硬件完全无感知，cooperative prefill 路径不变。
- 系统侧：persistent top-k 调度从 2 分支变为 3 分支，共享内存峰值需求从 128 KiB 升到 144 KiB；采样内核在单 launch 内同时承担采样与精确回退两条子路径，复杂度上升。
- 团队侧：新增 253 行核心 CUDA + 100 行测试 + 134 行基准，且为 AI 辅助实现（OpenAI Codex 联合作者），后续需要多架构（尤其非 B300 SM90）回归验证。

关联脉络

PR body 明确做了重复性检查，指出本实现与三个既有 PR 的关系：#44606 是现有 persistent top-k 调度调优，本 PR 的采样分支与之同属调度演进；#55314 修复既有 stash

overflow, 本 PR 通过 [CheckOverflow](#) 把 " 截断 stash 用子集 " 改为 " 回退全行精确选择 ", 在采样路径外根治同类问题; #55872 新增 FlashInfer 后端, 是同一 sparse-indexer top-k 功能线的不同实现路径。

此外, #56743 在 ROCm 侧为 DSV4.1 K=512 decode top-k 做 gfx950 专用优化, 与本 PR 形成 " 同一功能点在 NVIDIA/ROCm 两侧并行优化 " 的格局; 而 #55879、#56969、#56825 等稀疏 MLA 稳定性修复则持续为这条路径加固正确性。整体看, DeepSeek V4 稀疏索引器的 decode 路径正处于密集的性能与稳定性演进期, 本 PR 的采样过滤是其中关键的一环。