

PR #56743 完整报告

vllm-project/vllm

[ROCm][Perf] Optimize DSV4.1 K=512 decode top-k on gfx950

合并时间: 2026-09-16 02:34

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56743>

执行摘要

- 一句话: gfx950 新增 K=512 top-k 专用路径, 解码提速 2-5 倍
- 推荐动作: 值得精读。核心价值不在代码量而在三个设计点: 1) 用线程内聚合消除 -inf 直方图原子竞争, 是稀疏 mask 场景的通用优化模式; 2) 由设备端序列长度驱动 split 选择并保证 FULL graph replay 合法, 避免 CUDA graph 服务下的形状漂移; 3) "构建独立参考实现再与 kernel 输出逐项比对" 的测试思想 (_assert_exact_topk 不依赖 topk 算子自身)。若你维护 ROCm kernel 或稀疏注意力路径, 建议重点看 sampler.cu 的 finalBinFits 与 masked_graph_replay 测试。

功能与动机

本 PR 是 RFC #56506 的一部分。该 RFC 指出 DeepSeek-V4.1-Flash 在 AMD MI355X (gfx950) 上远未饱和——从并发 1 到 32, ITL 只涨 1.9 倍却换来 13 倍每 GPU 输出, 说明 decode 每步存在大量固定开销。decode indexer 的 top-k 就是其中一个热点: K=512 的压缩长度会错过 gfx950 已有调优, 回退到通用 10-split kernel。PR body 明确其任务边界: "Other architectures, K values, strides, and shapes keep the existing path", 并专门声明不含 multistream CSA、scheduler 或 queue 管理改动, 避免与 RFC 中其他优化项越界。

实现拆解

1. Python 侧路径门控 (入口): 在 vllm/v1/attention/ops/rocm_aiter_mla_sparse.py 中新增常量 `_GFX950_DSV4_NATIVE_MAX_COLUMNS = 1024 * 1024`, `_get_aiter_top_k_kernel` 增加 `num_columns` 与 `topk_tokens` 参数; 当 `topk_tokens == 512`、`0 < num_rows <= 384` 且 `num_columns <= 1 Mi` 时直接返回 `None`, 表示不选 AITER decode kernel, 由 `rocm_aiter_sparse_attn_indexer` 落到 `native_top_k_per_row_decode`。该门控是性能门而非正确性门, 超出 384 行或 1 Mi 列即退回原有 AITER/ 通用路径。
2. 内核核心改造: 在 `csrc/libtorch_stable/sampler.cu` 中给 `topKPerRowJob` 与 `processHistogramStep` 增加模板参数 `useTopK512Optimization`, 并加 `static_assert` 要求每 block 1024 线程; `kNumFinalItems` 由 2048 降到 1024, 多 block merge 的 padding 值从 `-FLT_MAX` 改为 `-INFINITY`; `processHistogramStep` 先把线程内命中的 -inf logit 计数到 `negativeInfinityCount`, 再一次性 `atomicAdd`, 消除大量线程对同一 -inf 桶的原子竞争; `finalBinFits` 在 `step < 3` 且阈值桶恰为 -inf 桶时拒绝直接写入, 避免排序 padding 干扰精确 tie 发射; 最终排序删除 `useRadixSort` 分支, 统一为 `insertion sort`

lambda。

3. 测试配套：在 tests/kernels/test_top_k_per_row.py 中新增共享 marker requires_gfx950（基于 vllm.platforms.rocm.on_gfx950）与独立参考断言 _assert_exact_topk；新增 3 个测试——..._1d_seq_lens（覆盖 0/1/511/512/513/1023/1024/16000 等短行与因果偏移）、..._split_policy_boundaries（11 组参数化覆盖 64K/256K device-length 阈值两侧、4/3/2/single-block 行策略、384/385 优化 /native 回退边界）、..._masked_graph_replay（CUDA graph capture 后重放，并在重放间改变 mask 与 lengths）；同时扩展 AITER 内核选择用例，验证 385 行、1,048,577 列、CR2 等形状的门控两侧行为。
4. 验证与数据：1008 个矩阵用例（并发 1-64、spec tokens 1-6、context 10K-1M、CR 1/2、dense/masked）全部通过；warm HIP graph geomean 2.310x、256 MiB cache pressure 1.824x；GPU trace 中 72 对 profile 的 geomean 2.291x；GSM8K 1279→1286/1319。

关键文件：

- csrc/libtorch_stable/sampler.cu（模块 采样内核；类别 source；类型 core-logic；符号 processHistogramStep, topKPerRowJob）：K=512 优化的核心载体：新增 useTopK512Optimization 模板参数，将最终排序容量 2048→1024，聚合 -inf 直方图原子写以减少共享内存竞争，并修正 -inf cutoff 下的精确 tie 发射；同时删除 radix sort 分支统一为插入排序。
- vllm/v1/attention/ops/rocm_aiter_mla_sparse.py（模块 内核选择；类别 infra；类型 infrastructure；符号 _get_aiter_top_k_kernel, rocm_aiter_sparse_attn_indexer）：决定 K=512 走 native 还是 AITER 的路径选择器：新增 num_columns/topk_tokens 门控，把 gfx950 上符合 K=512 优化形状的 decode 强制导向 native kernel，是 Python 侧的变更入口。
- tests/kernels/test_top_k_per_row.py（模块 内核测试；类别 test；类型 test-coverage；符号 _on_gfx950, _assert_exact_topk, test_top_k_per_row_decode_gfx950_k512_1d_seq_lens, test_top_k_per_row_decode_gfx950_k512_split_policy_boundaries）：新增 gfx950 K=512 的精确参考断言、split policy 边界参数化矩阵与 CUDA graph replay 验证，是防回归的主要保障，也回应了评审中关于分支覆盖不足的意见。

关键符号：processHistogramStep, topKPerRowJob, _get_aiter_top_k_kernel, rocm_aiter_sparse_attn_indexer, _assert_exact_topk, test_top_k_per_row_decode_gfx950_k512_1d_seq_lens, test_top_k_per_row_decode_gfx950_k512_split_policy_boundaries, test_top_k_per_row_decode_gfx950_k512_masked_graph_replay

关键源码片段

[csrc/libtorch_stable/sampler.cu](#)

K=512 优化的核心载体：新增 useTopK512Optimization 模板参数，将最终排序容量 2048→1024，聚合 -inf 直方图原子写以减少共享内存竞争，并修正 -inf cutoff 下的精确 tie 发射；同时删除 radix sort 分支统一为插入排序。

```

// processHistogramStep 的直方图构建段 (K=512 优化版) 。
// 稀疏 MLA 的 mask 会让大量 logit 恒等于 -inf: 若每个 -inf 都执行一次
// atomicAdd, 同一直方图桶会出现极高竞争。这里先在线程内聚合再一次性提交。
int negativeInfinityCount = 0;
auto distributeToBins = [&](float logit, int /* idx */ = 0) {
    if (isPartialMatch<patternShift>(logit, logitPattern)) {
        uint32_t binIdx = extractBinIdx<step>(logit);
        if (useTopK512Optimization && __float_as_uint(logit) == 0xff800000u) {
            ++negativeInfinityCount; // 线程内合并 -inf, 命中同一桶只做一次原子加
        } else {
            atomicAdd(&smemFinal.histo.data[binIdx], 1);
        }
    }
};
// ... 遍历本线程负责的 logits, 逐元素调用 distributeToBins ...
if constexpr (useTopK512Optimization) {
    if (negativeInfinityCount > 0) {
        const auto bin = extractBinIdx<step>(-INFINITY);
        atomicAdd(&smemFinal.histo.data[bin], negativeInfinityCount);
    }
}

```

```

// 阈值桶能否直接进入最终排序: "放得下" 只是必要条件; 当 -inf 的位模式
// 与当前步掩码部分匹配、且阈值桶恰为 -inf 所在桶时, 直接写入会把排序
// pad 的 -inf 当作有效候选项, 因此强制拒绝并继续细分。
const bool finalBinFits =
    smemFinalBinSize[0] <= kNumFinalItems &&
    !(useTopK512Optimization && step < 3 &&
    isPartialMatch<patternShift>(-INFINITY, logitPattern) &&
    thresholdBinIdx == extractBinIdx<step>(-INFINITY));

```

vllm/v1/attention/ops/rocm_aiter_mla_sparse.py

决定 K=512 走 native 还是 AITER 的路径选择器: 新增 num_columns/topk_tokens 门控, 把 gfx950 上符合 K=512 优化形状的 decode 强制导向 native kernel, 是 Python 侧的变更入口。

```

# AITER v0.1.19 的 decode kernel 是单 block 实现; gfx950 上 K=512 的
# DeepSeek-V4.1 场景已由 native sampler.cu 专用路径覆盖 (更快且保证
# FULL graph replay 合法), 因此这里把符合条件的形状挡在 AITER 之外。
_GFX950_DSV4_NATIVE_MAX_COLUMNS = 1024 * 1024 # 保守性能门, 非正确性边界

```

```

def _get_aiter_top_k_kernel(
    is_prefill: bool,
    compress_ratio: int,
    num_rows: int,
    max_valid_seq_len: int | None = None,
    num_columns: int | None = None,
    topk_tokens: int = 1024,

```

```

    on_gfx950: bool = _ON_GFX950,
) -> Callable[..., None] | None:
    if compress_ratio <= 1 or not on_gfx950:
        return None # 非 gfx950 或未压缩时维持原路径
    if not is_prefill:
        assert max_valid_seq_len is not None
        if (
            topk_tokens == 512
            and 0 < num_rows <= 384
            and num_columns is not None
            and num_columns <= _GFX950_DSV4_NATIVE_MAX_COLUMNS
        ):
            return None # 命中 K=512 优化形状: 不选 AITER, 落到 native 内核
            # ... 其余 AITER decode 形状判断保持不变 ...

```

tests/kernels/test_top_k_per_row.py

新增 gfx950 K=512 的精确参考断言、split policy 边界参数化矩阵与 CUDA graph replay 验证，是防回归的主要保障，也回应了评审中关于分支覆盖不足的意见。

```

@requires_gfx950 # 与生产代码共用 vllm.platforms.rocm.on_gfx950() 判定
@torch.inference_mode()
def test_top_k_per_row_decode_gfx950_k512_masked_graph_replay() -> None:
    """掩码与序列长度在图重放时变化，仍须保留可见 top-k 分数。"""
    rows, next_n, width, top_k = 24, 6, 262_145, 512
    logits = torch.full((rows, width), 1e20, dtype=torch.float32, device="cuda")
    lengths = torch.full(
        (rows // next_n, next_n), 70_000, dtype=torch.int32, device="cuda"
    )
    indices = torch.full((rows, top_k), -777, dtype=torch.int32, device="cuda")
    logits[:, :70_000] = float("-inf") # 前 70K 列 mask 掉，构造稀疏输入

    def run() -> None:
        _run_topk_backend("top_k_per_row_decode", logits, lengths,
            indices, top_k, width, next_n)

    run()
    _assert_exact_topk(logits, indices, lengths) # 与独立参考实现逐项比对
    torch.accelerator.synchronize()
    graph = torch.cuda.CUDAGraph()
    with torch.cuda.graph(graph):
        run() # capture 时按设备端 lengths 固化 split 选择
    indices.fill_(-777)
    graph.replay()
    _assert_exact_topk(logits, indices, lengths)

    # 换一组行边界 (0/511/512/513/16K/16K+1/128K/128K+1/256K/256K+1) ,
    # 验证 mask 与 length 都变化后重放依然精确
    bounds = [0, 511, 512, 513, 16_384, 16_385,
        131_072, 131_073, 262_144, 262_145]

```

```
row_bounds = torch.tensor(bounds, dtype=torch.int32, device="cuda")
repeats = (rows + len(bounds) - 1) // len(bounds)
lengths.flatten().copy_(row_bounds.repeat(repeats)[:rows])
logits[:] = torch.arange(width, dtype=torch.float32, device="cuda")
indices.fill_(-777)
graph.replay()
_assert_exact_topk(logits, indices, lengths)
```

评论区精华

有效评审意见集中在测试侧，均由作者在 commit 6818801fa9 落实后获得 tjtanana 的 LGTM。核心交锋：1) tjtanana 指出 24 行 masked replay 用例只覆盖 1/8/10/16 split 分支，4/3/2-split 与大行单 block 路径没有 GPU 测试，要求参数化覆盖 384/385 回退边界；2) tjtanana 两次要求直接复用 `from vllm.platforms.rocm import on_gfx950`，避免在测试文件里自建 `_on_gfx950` 包装。作者均以新增参数化矩阵与共享 marker 回应。无未解决疑虑。

- K=512 split policy 参数化覆盖 (testing): 作者在 commit 6818801fa9 中新增 `test_top_k_per_row_decode_gfx950_k512_split_policy_boundaries`，参数化 11 组覆盖 64K/256K device-length 阈值两侧、4/3/2/single-block 行策略、384/385 优化 /native 边界；13 个聚焦 K=512 用例全部通过。
- 复用共享的 `on_gfx950` helper (style): 作者回应已改用共享 marker (`requires_gfx950` 基于 `vllm.platforms.rocm.on_gfx950()`)，两处重复评论均落实。

风险与影响

- 风险：
 1. `sampler.cu` 删除了 `useRadixSort` 分支：调整前 `useRadixSort=true` 的模板实例会走 radix sort 最终排序，调整后统一为插入排序。若上游仍存在该实例化调用方，非 K=512 路径的排序性能可能变化，而 PR 基准只覆盖 `gfx950 K=512`，未对该路径做性能验证。
 2. 门控是性能门：384 行 /1 Mi 列边界两侧行为不同 (native vs AITER)，参数化测试覆盖了边界本身，但真实 serving 中 row/length 组合更复杂，边界处存在性能跳变窗口。
 3. FULL graph replay: split 选择由设备端 sequence lengths 驱动并随 capture 固化；masked graph replay 测试覆盖了 replay 时 mask/length 变化，但测试行数 (24) 与真实并发 shape 仍有差距。
 4. `gfx950` 专用用例在非 ROCm 平台自动 skip，上游 Buildkite 无法验证内核正确性，依赖 AMD CI (shen-shanshan 触发的 /amd-ci 构建)；PR 中一次 CI 失败被归因为 #56819 而非本 PR。
 5. 验收依赖单一精度口径 (GSM8K 1279→1286/1319)，RFC 提示 gate 0.91 可能掩盖局部回归，建议后续按 RFC 口径复测。
- 影响：用户与部署：`gfx950 (MI355X)` 上 DSV4.1-Flash decode top-k 热路径获得 2-5x 提速 (warm graph geomean 2.31x)，影响范围严格限定在 ROCm `gfx950`、K=512、≤384 行、≤1 Mi 列的形状；其他架构与形状行为不变。系统：decode 每步固定开销下降，间接缓解 RFC 中 "并发只摊销固定成本" 的瓶颈，对低并发 ITL 更有利；不涉及 scheduler、queue、CSA 改动，无系

统性风险。团队：与 #56503、#56903、#56962 等构成 DSv4.1 ROCm 性能系列，共享验收口径；sampler.cu 模板参数增加带来一定维护成本，但 " 精确 topk 断言 +graph replay" 的测试方法论可复用到其他 K 值。 - 风险标记：共享内核路径变更，radix sort 分支移除，性能门控边界，gfx950 专属测试，CUDA graph replay 约束

关联脉络

- PR #56962 [Perf][Kernel] Integrate Mega-mHC from DeepGEMM for DeepSeek V4.1 (reopen of #56255): 同属 RFC #56506 的 DSv4.1 decode 性能优化线，共享模型与 GSM8K 验收口径。
- PR #56903 [Perf][DSpark] Collapse DeepSeek-V4.1 draft states before SP all-gather: 同样针对 decode 固定开销的 DSpark 通信优化，与本 PR 在 DSpark + sparse MLA decode 热路径上叠加。
- PR #55966 [ROCm][Spec Decode] Add Aiter MLA decode support non-causal draft block: 同一 ROCm AITER MLA 稀疏路径的相邻模块 (rocm_aiter_mla.py)，构成本 PR 门控所依赖的 AITER 能力基础。