PR #46746 完整报告

vllm-project/vllm

[ModelRunner V2] Bound memory for large logprobs requests

合并时间：2026-06-26 09:04

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46746

执行摘要

- 一句话：限制大 logprobs 请求时的显存爆炸
- 推荐动作：建议精读此 PR，以学习如何用分块方式避免 Triton 内核中编译时常量过大导致的显存问题。该设计模式可推广到其他类似的 gather/scatter 场景。

功能与动机

当 num_logprobs 设为 -1（表示整个词表）时，triton.next_power_of_2(num_logprobs) 会计算出极大的 PADDED_TOPK 值，导致临时内核的共享内存爆炸。PR body 明确指出：'In this case, the temp MRV2 kernel was exploding due to PADDED_TOPK=triton.next_power_of_2(num_logprobs).'

实现拆解

1. 添加常量上限：在 logprob.py 顶部定义 _MAX_TOPK_BLOCK = 1024，作为 gather 操作的每轮最大宽度。
2. 修改内核参数：将内核的编译时常量 PADDED_TOPK 替换为 TOPK_BLOCK_SIZE，在运行时由调用方传入。
3. 分块 gather：将 _topk_log_softmax_kernel 中原有的连续 gather 循环改为 for j in range(0, topk, TOPK_BLOCK_SIZE): 分块加载至 topk_ids 并计算对数概率，确保每次 gather 宽度不超过 1024。
4. 调用方适配：在 compute_token_logprobs 中计算 topk_block_size = min(triton.next_power_of_2(num_logprobs), _MAX_TOPK_BLOCK) 并传入内核，保留了向上取整的规范性，但有界。

关键文件：

- vllm/v1/worker/gpu/sample/logprob.py（模块 采样器；类别 source；类型 core-logic；符号 _MAX_TOPK_BLOCK, _topk_log_softmax_kernel, compute_token_logprobs）：唯一变更文件，修改了 Triton 内核和调用函数以限制 gather 宽度。

关键符号：_topk_log_softmax_kernel, compute_token_logprobs

关键源码片段

vllm/v1/worker/gpu/sample/logprob.py

唯一变更文件，修改了 Triton 内核和调用函数以限制 gather 宽度。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project

import numpy as np
import torch

from vllm.sampling_params import MAX_LOGPROB_TOKEN_IDS, SamplingParams
from vllm.triton_utils import tl, triton
from vllm.v1.outputs import LogprobsTensors
from vllm.v1.worker.gpu.buffer_utils import StagedWriteTensor, UvaBackedTensor

# 新增：限制 topk gather 的每轮宽度，防止 num_logprobs=-1 时显存爆炸
_MAX_TOPK_BLOCK = 1024


@triton.jit
def _topk_log_softmax_kernel(
    output_ptr,
    logits_ptr,
    logits_stride,
    topk_ids_ptr,
    topk,
    vocab_size,
    BLOCK_SIZE: tl.constexpr,
    TOPK_BLOCK_SIZE: tl.constexpr, # 替代原有的 PADDED_TOPK，由调用方传入有界值
):
    req_idx = tl.program_id(0).to(tl.int64)
    row_ptr = logits_ptr + req_idx * logits_stride

    max_val = float("-inf")
    for i in range(0, vocab_size, BLOCK_SIZE):
        block = i + tl.arange(0, BLOCK_SIZE)
        logits = tl.load(row_ptr + block, mask=block < vocab_size, other=float("-inf"))
        max_val = tl.max(tl.maximum(logits, max_val))
    max_val = max_val.to(tl.float32)

    se = 0.0
    for i in range(0, vocab_size, BLOCK_SIZE):
        block = i + tl.arange(0, BLOCK_SIZE)
        logits = tl.load(row_ptr + block, mask=block < vocab_size, other=0.0)
        logits = logits.to(tl.float32)
        e = tl.exp(logits - max_val)
        e = tl.where(block < vocab_size, e, 0.0)
        se += tl.sum(e)
    lse = tl.log(se)

    # 分块 gather：每次最多处理 TOPK_BLOCK_SIZE 个 topk_ids
```

```
    for j in range(0, topk, TOPK_BLOCK_SIZE):
        k_offset = j + tl.arange(0, TOPK_BLOCK_SIZE)
        k_mask = k_offset < topk
        topk_ids = tl.load(
            topk_ids_ptr + req_idx * topk + k_offset, mask=k_mask, other=0
        )
        logits = tl.load(row_ptr + topk_ids, mask=k_mask)
        logits = logits.to(tl.float32)
        o = logits - max_val - lse
        tl.store(output_ptr + req_idx * topk + k_offset, o, mask=k_mask)


def compute_token_logprobs(
    logits: torch.Tensor, token_ids: torch.Tensor
) -> torch.Tensor:
    batch_size, vocab_size = logits.shape
    token_ids = token_ids.to(torch.int64)
    num_logprobs = token_ids.shape[1]
    logprobs = logits.new_empty((batch_size, num_logprobs), dtype=torch.float32)

    # 计算有界的 TOPK_BLOCK_SIZE，防止显存爆炸
    topk_block_size = min(triton.next_power_of_2(num_logprobs), _MAX_TOPK_BLOCK)
    _topk_log_softmax_kernel[(batch_size,)](
        logprobs,
        logits,
        logits.stride(0),
        token_ids,
        num_logprobs,
        vocab_size,
        BLOCK_SIZE=1024,
        TOPK_BLOCK_SIZE=topk_block_size,
    )
    return logprobs
```

评论区精华

该 PR 没有 review 评论或线程。WoosukKwon 直接批准，并评论 'LGTM. Thanks!'，表明变更简洁正确，无需深入讨论。

- 暂无高价值评论线程

风险与影响

- 风险：回归风险低：仅修改了一个 Triton 内核及其调用点，未改动其他逻辑路径。对于 num_logprobs < 1024 的请求，行为完全不变；对于大 num_logprobs 请求，由于引入了分块循环，会引入额外的循环开销，但相比之前的显存爆炸是显著改进。性能风险：对于非常大的 num_logprobs，分块 gather 可能增加少量延迟，但考虑到整个词表场景较少见且之前无法运行，这是可接受的 trade-off。

- 影响：用户影响：修复了使用 num_logprobs=-1 时 v1 引擎崩溃的问题，使全词表 logprobs 功能可用。系统影响：变更仅涉及单个文件，代码量小，属于局部 bugfix。团队影响：无需更新测试或文档，因为已有测试覆盖但可能未触发该 *edge case*。
- 风险标记：核心路径变更，性能影响（分块循环开销）

关联脉络

- 暂无明显关联 PR