

PR #48641 完整报告

vllm-project/vllm

[Sampler] Stop upcasting logits to fp32 in apply_sampling_params

合并时间: 2026-07-18 09:15

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/48641>

执行摘要

- 一句话: 停止 `apply_sampling_params` 中 logits 的 fp32 提升, 修复 Spec Decode OOM
- 推荐动作: 值得精读, 展示了如何通过消除冗余类型转换来优化内存占用, 同时暴露并修复了边界情况下的问题。其“在需要时克隆”的设计思路也可借鉴。

功能与动机

Spec Decode 场景下 `apply_sampling_params` 将 logits 拷贝为 fp32 会分配数 GiB 内存, 例如 Qwen3-8B + dflash 尝试分配 4.12 GiB 并在预热阶段 OOM。此前 #48037 和 #48630 试图通过 profiling 绕开, 但此 PR 从根本上消除不必要的拷贝, 同时确保所有下游算子内部已处理 fp32 转换。

实现拆解

1. 移除 fp32 拷贝并新增 `returns_logprobs` 方法: 在 `vllm/v1/worker/gpu/sample/sampler.py` 的 `apply_sampling_params` 中, 将无条件 fp32 拷贝替换为按需克隆 (仅当 `logprobs_mode` 以 "raw_" 开头且需返回 logprobs 时)。同时将 `__call__` 中的 `return_logprobs` 计算提取为独立的 `returns_logprobs(idx_mapping_np)` 方法, 以避免重复查询状态。
2. 放宽 Triton 内核输入类型: 在 `vllm/v1/sample/ops/topk_topp_triton.py` 中, 将 `apply_top_k_top_p_triton` 的输入类型断言从 `torch.float32` 放宽为 `(torch.float32, torch.bfloat16, torch.float16)`, 并在内核所有 logits 加载点添加 `.to(torch.float32)` 显式转换, 确保 computation 在 fp32 精度进行。
3. 修复 top-p PyTorch 回退精度: 在 `vllm/v1/sample/ops/topk_topp_sampler.py` 的 `apply_top_k_top_p_pytorch` 中, 将 `logits_sort.softmax(dim=-1)` 改为 `softmax(dim=-1, dtype=torch.float32)`, 避免 bf16 累积导致核不足。
4. 修复 FlashInfer 采样器对跨步视图的兼容性: 在 `flashinfer_sample` 函数中, 对同时包含 top-k 和 top-p 的分支, 调用 `logits.float().contiguous()` 以保证 FlashInfer 需要的连续 fp32 输入。
5. 增加回归测试: 在 `tests/v1/sample/test_topk_topp_sampler.py` 中新增 `test_flashinfer_sample_padded_vocab`, 使用 opt-125m 的 50272 词表 (非 64 对齐), 验证 bf16/ fp32、多种 top-k/ top-p 组合下的正确性。

此外，在 `vllm/v1/worker/gpu/sample/logit_bias.py` 的 Triton 内核中，对 logits 加载也添加 `.to(tl.float32)`，与整体 dtype 一致化。

关键文件：

- `vllm/v1/worker/gpu/sample/sampler.py`（模块 采样器；类别 source；类型 core-logic；符号 `returns_logprobs`）：核心变更文件，移除了 fp32 拷贝并新增 `returns_logprobs` 方法，将拷贝决策内聚到 `apply_sampling_params` 中。
- `tests/v1/sample/test_topk_topp_sampler.py`（模块 采样测试；类别 test；类型 test-coverage；符号 `test_flashinfer_sample_padded_vocab`）：新增回归测试，覆盖 FlashInfer 采样器在非对齐词表下的跨步视图场景，验证 bf16/ fp32 及多种 top-k/ top-p 组合。
- `vllm/v1/worker/gpu/sample/logit_bias.py`（模块 偏置内核；类别 source；类型 core-logic）：Triton 内核中添加 `.to(tl.float32)` 转换，确保在 bf16 输入下计算精度正确。
- `vllm/v1/sample/ops/topk_topp_triton.py`（模块 Triton 内核；类别 infra；类型 infrastructure）：放宽输入类型断言，在所有 logits 加载点添加 fp32 转换，使 Triton 内核能直接处理 bf16/ fp16 输入。
- `vllm/v1/sample/ops/topk_topp_sampler.py`（模块 采样算子；类别 infra；类型 infrastructure）：修复 top-p PyTorch 回退路径的精度问题，以及 FlashInfer 采样器的跨步视图处理。

关键符号：`Sampler.returns_logprobs`, `Sampler.apply_sampling_params`, `apply_top_k_top_p_triton`, `apply_top_k_top_p_pytorch`, `flashinfer_sample`, `_bias_kernel`, `test_flashinfer_sample_padded_vocab`

关键源码片段

`vllm/v1/worker/gpu/sample/sampler.py`

核心变更文件，移除了 fp32 拷贝并新增 `returns_logprobs` 方法，将拷贝决策内聚到 `apply_sampling_params` 中。

```
def apply_sampling_params(
    self,
    logits: torch.Tensor,
    expanded_idx_mapping: torch.Tensor,
    idx_mapping_np: np.ndarray,
    pos: torch.Tensor,
    input_ids: torch.Tensor,
    expanded_local_pos: torch.Tensor,
    skip_top_k_top_p: bool = False,
) -> torch.Tensor:
    # 之前：无条件分配 fp32 拷贝，导致 Spec Decode 下内存暴涨。
    # 现在：下游 op 内部已自行转换到 fp32，仅在 raw_logprobs 需要原始 logits 时克隆。
    if self.logprobs_mode.startswith("raw_") and self.returns_logprobs(
        idx_mapping_np
    ):
        logits = logits.clone()
```

```

# 以下操作均在 logits 的原始 dtype 上进行（例如 bf16）,
# 各 op 内部在必要时会临时上转 fp32。
self.logit_bias_state.apply_logit_bias(
    logits, expanded_idx_mapping, idx_mapping_np, pos
)
self.penalties_state.apply_penalties(
    logits, expanded_idx_mapping, idx_mapping_np,
    input_ids, expanded_local_pos,
)
self.bad_words_state.apply_bad_words(
    logits, expanded_idx_mapping, idx_mapping_np,
)

if not skip_top_k_top_p:
    logits = self.sampling_states.apply_top_k_top_p(
        logits, idx_mapping_np, expanded_idx_mapping,
    )
return logits

```

```

def returns_logprobs(self, idx_mapping_np: np.ndarray) -> bool:
    """当前批次是否有任意一条请求需要计算 logprobs"""
    return (
        self.sampling_states.max_num_logprobs(idx_mapping_np) != NO_LOGPROBS
        or self.logprob_token_ids_state.max_num_token_ids(idx_mapping_np) > 0
    )

```

tests/v1/sample/test_topk_topp_sampler.py

新增回归测试，覆盖 FlashInfer 采样器在非对齐词表下的跨步视图场景，验证 bf16/ fp32 及多种 top-k/ top-p 组合。

```

def test_flashinfer_sample_padded_vocab(
    dtype: torch.dtype, k: int | None, p: float | None
):
    """验证 flashinfer_sample 能正确处理 logits 跨步视图和非 fp32 输入。

    compute_logits 会切掉 vocab 填充，因此如果 vocab size 不是 64 的倍数
    （例如 opt 的 50272），得到的 logits 是一个跨步视图。FlashInfer 原本要求
    连续的 fp32 输入，本测试确保在修复后能正确工作。
    """
    torch.set_default_device(DEVICE_TYPE)
    batch_size = 8 # 单行时跨步视图可能是连续的，因此 batch > 1 才能暴露问题
    org_vocab_size = 50272
    padded_vocab_size = pad_vocab_size(org_vocab_size)
    assert padded_vocab_size != org_vocab_size

    # 创建 padded 然后切片，得到非连续张量
    logits = torch.randn(batch_size, padded_vocab_size, dtype=dtype)[
        ..., :org_vocab_size
    ]

```

```
assert not logits.is_contiguous()

token_ids = flashinfer_sample(
    logits,
    torch.full((batch_size,), k, dtype=torch.int32) if k is not None else None,
    torch.full((batch_size,), p, dtype=torch.float32) if p is not None else None,
)
assert token_ids.shape == (batch_size,)
assert torch.all((token_ids >= 0) & (token_ids < org_vocab_size))
```

评论区精华

njhill 在 review 中指出，将 `return_logprobs` 判断内聚到 `apply_sampling_params` 后，`Sampler.__call__` 中会重复计算 `max_num_logprobs` 和 `max_per_req_token_ids`（每步最多 3 次），建议后续优化。mgoin 同意并计划作为 follow-up。此外，cakeng 确认 Triton 内核中 fp32 缓冲区会自动下转换为输出 dtype，无需担心精度损失。njhill 还提出了针对已合并 #47680（logprobs_mode 修复）的条件更新，已采纳。

- Triton 内核 fp32 下转换确认 (correctness): 自动下转换，无需额外处理。
- raw_logprobs 拷贝决策内聚与重复计算 (design): 采用 returns_logprobs 方法，重复计算作为后续优化。
- logprobs_mode 条件更新 (design): 已采纳，更新条件。

风险与影响

- 风险：1) 精度一致：作者验证了每个 logits 核 (penalties、temperature、min_p、bad_words、logit_bias、gumbel、logprobs、top-k/ top-p) 在跨步视图和连续拷贝下位一致，且端到端 opt-125m 比对结果完全相同。2) top-p 精度修复：修复了 bf16 累积导致核不足的问题，已通过 chi-square 检验。3) FlashInfer 跨步视图崩溃：通过 logits.float().contiguous() 修复，新增回归测试。4) processed_logprobs 数值微小差异：基于 bf16 派生，返回的对数概率值可能变化 ≤ 0.01 ，但 token 选择不受影响，且 head_dtype=float32 (RL 路径) 保持不变。5) 重复计算：Sampler.__call__ 中 max_num_logprobs 和 max_per_req_token_ids 被多次调用，在常规采样路径中影响可忽略，但可作为后续优化。
- 影响：对 Spec Decode 用户影响最大：从 OOM 到正常服务，内存回归消除。对非 Spec Decode 用户：内存消耗减少（不再 fp32 拷贝），性能微提升。对 raw_logprobs 用户：行为不变（仍 clone）。对所有用户：采样结果一致（已验证）。团队：移除了脆弱的 profiling 补偿 (#48037)，简化了采样器内部逻辑。
- 风险标记：核心采样路径变更，精度敏感场景（已验证无回归）

关联脉络

- PR #48037 [MRV2][Spec Decode] Profile the rejection sampler to fix spec-decode OOM: 此 PR supersedes #48037，后者通过 profiling 规避问题，本 PR 从根本上移除拷贝。

- PR #47680 [Bugfix][V1/V2] Fix prompt_logprobs to respect logprobs_mode: 关联的 logprobs_mode 修复, 影响本 PR 中 startswith('raw_') 的判断条件。
- PR #48630 [MRV2][Spec Decode] Profile the rejection sampler to fix spec-decode OOM (duplicate): 类似修复尝试, 已被 supersede。