PR #44042 完整报告

vllm-project/vllm

[CI] Reject out-of-vocabulary before they reach the GPU logprob path

合并时间：2026-06-03 11:27

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44042

执行摘要

- 一句话：提前拒绝越界 token ID，稳定 ROCm CI
- 推荐动作：值得精读。该 PR 展示了早期验证如何防御 GPU 异常，以及如何针对平台差异做最小侵入性 workaround。其中验证插入位置和 platform check 的使用方式可为类似问题提供参考。

功能与动机

ROCm CI 中由 Schemathesis 生成的包含越界 logprob_token_ids（如负数、超大数）的请求，会触发 ROCm/HSA 硬件异常并杀死引擎，导致后续测试级联失败。需要在请求验证阶段提前拒绝这些无效参数。

实现拆解

1. 在 SamplingParams._validate_logprobs 中新增词汇范围校验：获取 vocab_size 后，对 logprob_token_ids 中每个 token_id 检查是否 <0 或 >= vocab_size，若是则抛出 VLLMValidationError。
2. 在 test_logprobs.py 中添加单元测试，利用 SimpleNamespace 模拟 ModelConfig，验证有效和无效 token_id 的通过 / 拒绝行为。
3. 在 gpu_model_runner.py 的 shutdown 方法中，为 ROCm 增加 CUDA 图显式清理（CUDAGraphWrapper.clear_all_graphs）及额外的 gc.collect/empty_cache/synchronize，防止下次启动时 HSA 故障。
4. 在 thinking_budget_state.py 的 apply_forcing_to_logits 中，将 ROCm 平台上的 2D index_put 替换为 1D index_fill_ 或逐行赋值，以避免 ROCm 在高级索引写入时的内存访问错误。

关键文件：

- vllm/sampling_params.py（模块 采样参数；类别 source；类型 core-logic；符号 _validate_logprobs）：核心验证逻辑所在地，新增词汇范围校验，是 PR 标题变更的核心。
- vllm/v1/sample/thinking_budget_state.py（模块 思考预算；类别 source；类型 core-logic；符号 _apply_forcing_to_logits）：修复 ROCm 上 thinking budget 写入路径的内存访问错误，是 CI 稳定的关键一环。
- tests/v1/sample/test_logprobs.py（模块 测试；类别 test；类型 test-coverage；符号 _model_config, test_logprob_token_ids_validate_vocab_bounds_valid,

test_logprob_token_ids_validate_vocab_bounds_invalid）：新增单元测试覆盖词汇范围验证逻辑，确保正确性。

- vllm/v1/worker/gpu_model_runner.py（模块 模型运行器；类别 source；类型 data-contract；符号 shutdown）：ROCm shutdown 路径增加 CUDA 图清理和内存回收，避免下次启动时 HSA 故障。

关键符号：_validate_logprobs, shutdown, _apply_forcing_to_logits, _model_config, test_logprob_token_ids_validate_vocab_bounds_valid, test_logprob_token_ids_validate_vocab_bounds_invalid

关键源码片段

vllm/v1/sample/thinking_budget_state.py

修复 ROCm 上 thinking budget 写入路径的内存访问错误，是 CI 稳定的关键一环。

```
if active_indices_cpu:
    device = logits.device
    if current_platform.is_rocm() and logits.is_contiguous():
        # Flattened 1D index_fill 避免 ROCm 上 2D 高级索引写错误
        vocab_size = logits.shape[1]
        flat_indices_cpu = [
            row * vocab_size + token
            for row, token in zip(active_indices_cpu, force_tokens_cpu)
        ]
        flat_indices = async_tensor_h2d(
            flat_indices_cpu, dtype=torch.long, device=device
        )
        logits.view(-1).index_fill_(0, flat_indices, 1e9)
    elif current_platform.is_rocm():
        # 非连续张量退化为逐行赋值
        fill = logits.new_tensor(1e9)
        for row, token in zip(active_indices_cpu, force_tokens_cpu):
            logits[row, token] = fill
    else:
        # 非 ROCm 平台保持原 index_put_ 路径
        active_indices = async_tensor_h2d(
            active_indices_cpu, dtype=torch.long, device=device
        )
        force_tokens = async_tensor_h2d(
            force_tokens_cpu, dtype=torch.long, device=device
        )
        fill = logits.new_full((len(active_indices_cpu),), 1e9)
        logits.index_put_((active_indices, force_tokens), fill)
```

评论区精华

1. 验证时机讨论：hclsys 担心 verify() 可能在模型加载前被调用，导致 get_vocab_size 不可用。作者回应 verify 依赖 ModelConfig，且已用于 logprobs=-1，不会引入新假设。

2. PR 范围讨论：njhill 建议将 ROCm 特定修复（thinking budget、shutdown）拆分为独立 PR。作者解释这些修复是使 CI 通过的必要条件，部分问题因重构才暴露。
3. 技术说明：作者在评论中解释了 thinking budget 修复的动机——避免 ROCm 上 2D 高级索引写入的潜在问题。

- 验证时机可行性 (correctness): 确认验证在正确时机执行，无需额外守卫。
- 是否将 ROCm 修复拆分为独立 PR (design): 保留在同一 PR 中，由作者统一管理。
- 思考预算写入路径的 ROCm 工作原因 (performance): 采用 flattened index_fill_ 作为默认 ROCm 路径，fallback 到循环赋值。

风险与影响

- 风险：核心风险：logprob_token_ids 验证逻辑新增了 vocab_size 依赖，但已有前置使用（logprobs=-1），应为安全；ROCm 特定修改（shutdown、thinking budget）通过 platform check 隔离，不影响 CUDA；但 shutdown 路径的额外清理可能改变当前进程内的资源释放时序，需关注后续启动稳定性。
- 影响：影响范围：仅涉及 logprob_token_ids 参数的请求验证和 ROCm 平台下的模型关闭路径。对非 ROCm 用户无行为变化；对 ROCm 用户，无效请求将提前报错而非崩溃，引擎重启更可靠。团队受益于更稳定的 CI。
- 风险标记：平台依赖变更，核心验证路径，稳定性修复

关联脉络

- PR #43669 [Bugfix] flashinfer: fail fast when --kv-cache-dtype nvfp4 used on unsupported arch: 同样是在请求验证阶段提前失败，避免 GPU 异常，属于同一模式。
- PR #44082 [Bugfix] Cache the EAGLE/MTP lookahead block in the SWA prefix-cache mask: 同为 v1 子系统的 bugfix，可能共享相同的测试基础设施。