PR #49751 完整报告

vllm-project/vllm

[BugFix][MRV2] Don't create dummy requests longer than `max_model_len`

合并时间：2026-07-27 10:04

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/49751

执行摘要

- 一句话：限制 dummy request token 数不超过 max_model_len
- 推荐动作：值得精读：表面上是一次简单的余数分配调整，但它是理解 vLLM 中 dummy batch 与 CUDA graph 捕获机制交互的绝佳切入点。设计决策（如为何必须将多 token request 放在 batch 末尾）涉及 attention kernel 的 block-table 布局和 paged attention 的访问模式，是重要的边界条件案例。此外，该 PR 展示了如何通过集中修复一个底层 bug 来解锁上游优化 PR（#49364），体现了依赖管理的价值。

功能与动机

PR body 明确指出，InputBatch.make_dummy 和 GPUModelRunner._dummy_run 将全部余数加到最后一个 request，导致其 seq_len 可能达到 num_tokens - num_reqs + 1，远超 max_model_len。当 #49364 在 dummy batch 上执行 attention（cudagraph piecewise capture）时，块表越界读取垃圾页 ID，在 16GB MIG 环境稳定崩溃，迫使 #49364 被 revert（#49451）。修复后 #49364 可无需修改即重新落地。

实现拆解

1. vllm/v1/worker/gpu/input_batch.py – make_dummy 方法：将原来 num_scheduled_tokens[-1] += num_tokens % num_reqs 改为计算 base_tokens = num_tokens // num_reqs 和 num_extra = num_tokens % num_reqs，若有余数则从尾部开始均匀分配（num_scheduled_tokens[-num_extra:] += 1）。同时按相同模式设置 seq_lens，使各字段一致。
2. vllm/v1/worker/gpu/model_runner.py – _dummy_run 方法：将列表推导式 [num_tokens // num_reqs] * num_reqs 改为 [num_tokens // num_reqs + (i >= num_reqs - num_tokens % num_reqs) for i in range(num_reqs)]，实现均匀分布。
3. vllm/v1/worker/gpu/lora_utils.py – create_lora_capture_hook 内的 hook：移除 num_scheduled[-1] += num_tokens % num_reqs，改用 num_scheduled[-num_extra:] += 1，与 make_dummy 保持一致。
4. tests/v1/worker/test_gpu_input_batch_v2.py：新增参数化测试 test_make_dummy_distributes_remainder，覆盖余数大于 0、无余数、少请求等场景，断言每个 request 的 token 数不超过 ceil(num_tokens/num_reqs) 且总和正确。
5. 无其他配置或部署变更，改动仅影响 dummy batch 生成逻辑，不涉及推理路径。

关键文件：

- vllm/v1/worker/gpu/input_batch.py（模块 输入批处理；类别 source；类型 core-logic；符号 make_dummy）：核心修复：make_dummy 中调整 token 分配逻辑，确保每个 dummy request 的 token 数不超过 max_model_len。
- vllm/v1/worker/gpu/model_runner.py（模块 模型运行器；类别 source；类型 core-logic；符号 _dummy_run）：同步修复：_dummy_run 中的 token 分配与 input_batch 保持一致，均采用均匀分布。
- vllm/v1/worker/gpu/lora_utils.py（模块 LoRA 工具；类别 source；类型 core-logic；符号 create_lora_capture_hook）：LoRA 路径同步修复：create_lora_capture_hook 中的 token 分配与 input_batch 保持一致。
- tests/v1/worker/test_gpu_input_batch_v2.py（模块 测试；类别 test；类型 test-coverage；符号 test_make_dummy_distributes_remainder）：新增单元测试，覆盖各种 token/request 组合，验证均匀分布的正确性及边界条件。

关键符号：InputBatch.make_dummy, GPUModelRunner._dummy_run, create_lora_capture_hook (内部 hook), test_make_dummy_distributes_remainder

关键源码片段

vllm/v1/worker/gpu/input_batch.py

核心修复：make_dummy 中调整 token 分配逻辑，确保每个 dummy request 的 token 数不超过 max_model_len。

```
# vllm/v1/worker/gpu/input_batch.py
# 在 make_dummy 中，原来将全部余数给最后一个 request，现改为均匀分布
@classmethod
def make_dummy(
    cls,
    num_reqs: int,
    num_tokens: int,
    input_buffers: InputBuffers,
) -> "InputBatch":
    assert 0 < num_reqs <= num_tokens
    device = input_buffers.device

    # ... ( 省略 req_ids、idx_mapping 等不变部分 )

    # 计算基准 tokens 和余数
    base_tokens = num_tokens // num_reqs
    num_extra = num_tokens % num_reqs
    # 均匀分布：前 n - num_extra 个 request 分得 base_tokens，后 num_extra 个多 1
    num_scheduled_tokens = np.full(num_reqs, base_tokens, dtype=np.int32)
    if num_extra > 0:
        num_scheduled_tokens[-num_extra:] += 1
    assert int(num_scheduled_tokens.sum()) == num_tokens

    # seq_lens 同步调整
```

```
    input_buffers.seq_lens[: num_reqs - num_extra] = base_tokens
    input_buffers.seq_lens[num_reqs - num_extra : num_reqs] = base_tokens + 1
    input_buffers.seq_lens[num_reqs:] = 0 # padding for full CUDA graph
    seq_lens = input_buffers.seq_lens[:num_reqs]

    # query_start_loc、input_ids、positions 等不变 ...
```

tests/v1/worker/test_gpu_input_batch_v2.py

新增单元测试，覆盖各种 token/request 组合，验证均匀分布的正确性及边界条件。

```
# tests/v1/worker/test_gpu_input_batch_v2.py
# 测试 make_dummy 均匀分布余数，确保不产生超长 dummy request
@pytest.mark.parametrize(
    "num_reqs,num_tokens",
    [
        (256, 496), # 余数 240，原实现会塞给最后一个 request 241 tokens
        (128, 512), # 无余数
        (3, 8),
        (1, 7),
    ],
)
def test_make_dummy_distributes_remainder(num_reqs: int, num_tokens: int):
    buffers = InputBuffers(
        max_num_reqs=num_reqs, max_num_tokens=num_tokens, device=torch.device("cuda")
    )
    batch = InputBatch.make_dummy(num_reqs, num_tokens, buffers)

    max_per_req = -(-num_tokens // num_reqs) # ceil division
    # 总和必须等于 num_tokens
    assert batch.num_scheduled_tokens.sum() == num_tokens
    # 最大不超过 ceil
    assert batch.num_scheduled_tokens.max() == max_per_req
    # 最小不低于 floor
    assert batch.num_scheduled_tokens.min() >= num_tokens // num_reqs
    # 多 token 的 request 排在末尾（非递减顺序）
    assert (batch.num_scheduled_tokens[:-1] <= batch.num_scheduled_tokens[1:]).all()

    # 验证 query_len == seq_len (prefill 模式 )
    query_lens = batch.query_start_loc_np[1:] - batch.query_start_loc_np[:-1]
    assert (query_lens == batch.num_scheduled_tokens).all()
    assert torch.equal(
        batch.seq_lens, torch.from_numpy(batch.num_scheduled_tokens).to(DEVICE)
    )
    assert batch.query_start_loc_np[-1] == num_tokens
    assert torch.equal(
        batch.query_start_loc.cpu(), torch.from_numpy(batch.query_start_loc_np)
    )
```

评论区精华

- WoosukKwon提出：是否应该按 num_tokens 排序，使 decode requests 在前、prefill 在后？
- njhill先表示同意（"Ah good point"），后回复 "now updated"，确认已对 input_batch.py 做出调整，将多 token 的 request 放在 batch 末尾（与测试中的排序断言一致）。
- 无其他争议，WoosukKwon 最终批准。
- 是否应将多 token request 排序到 batch 末尾 (design): 已采纳排序建议：多 token 的 request 放置在 batch 的末尾（-num_extra:），测试中也包含顺序断言。

风险与影响

- 风险：风险极低：变更核心是均匀分布余数，逻辑简单且与原有模式高度一致。主要风险是 LoRA 路径 (lora_utils.py) 可能未同步，但已确认同步。新增测试覆盖了典型边界，包括余数、无余数、单 request、小 batch 等场景，可防止回归。此外，若 num_tokens % num_reqs 结果与 _dummy_run 中 i 的排序逻辑产生不一致（理论上 i 从 0 开始，尾部分配，与 input_batch 的 -num_extra: 一致），需确认两者顺序匹配。测试中 num_scheduled_tokens[:-1] <= num_scheduled_tokens[1:] 确保了非递减序，与 LoRA hook 中的 -num_extra: 赋值顺序一致（较大的在尾部）。
- 影响：用户影响：修复了 cudagraph capture 时偶发的非法内存访问错误，消除 CI 中 5 个测试（test_logprobs_mode x4, test_prompt_logprobs_mode）在特定硬件（16GB MIG）上的可靠崩溃。对普通推理无直接影响，但提升了大模型预填充阶段 CUDA graph 捕获的稳定性。系统影响：无性能退化；改动位于非热路径（dummy run 仅在 cudagraph capture 时执行）。团队影响：为 #49364 的重新落地扫清障碍，该 PR 本意是分段 cudagraph capture 优化，因本 bug 被 revert。
- 风险标记：核心路径变更（dummy batch 影响 cudagraph capture），多路径同步（input_batch / model_runner / lora_utils 需一致），回归风险低（逻辑简单且覆盖测试）

关联脉络

- PR #49364 [Frontend][MRV2] Piecewise cudagraph capture with attention: 本 bug 是 #49364 被 revert（#49451）的根因：#49364 在 dummy batch 上执行 attention 时因 block-table 越界崩溃。修复本 PR 后 #49364 可无需修改直接重新落地。