PR #33906 完整报告

sgl-project/sglang

Fix prefill CP graph overflow with larger bucket search

合并时间：2026-08-07 16:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33906

执行摘要

- 一句话：修复 prefill CP BCG 图容量溢出，自动选择更大的捕获 bucket 重放
- 推荐动作：值得精读 python/sglang/srt/layers/cp/bcg.py 中 required_local_tokens 与 select_replay_bucket 的配套设计：用“所有 rank 最大需求”作为跨 rank 共识、准入与重放共用同一选择器是两个可复用的模式。建议合并前确认 test_gqa_prefill_cp.py 的失败是否为回归；另外可在代码中补充 bucket_local_tokens 在 capture 阶段的填充时机说明，便于后续维护。

功能与动机

PR body 引用了一个失败任务：4-GPU GPT-OSS MXFP4 context-parallel 测试在 prefill BCG 重放阶段崩溃。多请求的 live zigzag 布局可能比单请求捕获布局需要更多 CP 本地物理行（例如 520 行 live 对 512 行 captured）。第一版直接回退 eager 虽然安全，但在稍大的捕获图可以容纳 live 布局时，会不必要地放弃 BCG，因此需要按 CP 本地行容量搜索更大的捕获 bucket。

实现拆解

实现按以下 4 步拆解：

1. 容量模型建模（python/sglang/srt/layers/cp/bcg.py）：在 PrefillCPBCGInput 上新增 required_local_tokens(extend_seq_lens)，利用 get_cp_strategy() 获取 ZigzagCPStrategy，按 cp_size * 2 的 segment 数计算每个 rank 在 zigzag 布局下分到的逻辑 token 数（base * 2 加上 remainder 余数分配给 rank 与 opposite_rank），取所有 rank 的最大值，再按 get_cp_padding_align_size() 对齐，得到 live 布局最少需要的 CP 本地物理行数。该值不依赖具体 rank，天然可作为全 rank 共识依据。
2. bucket 选择器（bcg.py）：新增 select_replay_bucket(num_tokens, required_local_tokens, capture_num_tokens, max_padding_factor)，按升序遍历 capture_num_tokens，跳过小于 num_tokens 的 bucket，超过 num_tokens * max_padding_factor 即终止，返回第一个 bucket_local_tokens 中记录的 CP 本地容量满足需求的 bucket，找不到返回 None；select_replay_bucket_for_batch 先算 required_local_tokens，非 zigzag 或缺少长度时返回 None 交由调用方回退。
3. 准入与重放接入（python/sglang/srt/model_executor/runner/prefill_cuda_graph_runner.py）：在 can_run_graph 中，当 enable_cp_v2_bcg_capture 且

is_cp_v2_active(forward_batch)时调用同一选择器，返回None则拒绝图执行（走eager）；在 load_batch 中同样的守卫下用选择器计算 static_num_tokens 作为 padding 目标，若为 None 则 raise RuntimeError 作为 admission 与 replay 不一致时的防御性检查。两处均位于 CP-v2 active 分支内，满足 review 对 diff 收敛的要求。

4. 测试配套（test/registered/cp/test_cp_strategy_unit.py）：新增 TestPrefillCPBCGReplay 测试类，构造最小 PrefillCudaGraphRunner（通过 __new__ 绕过初始化）与 ForwardBatch，覆盖四个场景——容量溢出时跳到下一个 bucket（2048 缺容量则选 2304）、4 个 CP rank 选择一致、2x padding 限制保留、所有 bucket 容量不足时 can_run_graph 返回 False，并验证 load_batch 实际以 2304 作为 padded_num_tokens。根据 review 反馈，测试收敛到 CP 专用文件，不再改动 test_prefill_cuda_graph_padding.py。

关键文件：

- python/sglang/srt/layers/cp/bcg.py（模块 图执行；类别 source；类型 core-logic；符号 required_local_tokens, select_replay_bucket, select_replay_bucket_for_batch）：核心修改文件：新增 required_local_tokens、select_replay_bucket、select_replay_bucket_for_batch 三个方法，完成从 live zigzag 布局推导 CP 本地行需求并选择容量足够的最小捕获 bucket 的全部核心逻辑。
- python/sglang/srt/model_executor/runner/prefill_cuda_graph_runner.py（模块 图执行；类别 source；类型 core-logic；符号 can_run_graph, load_batch）：将新选择器接入图准入与重放两个入口，保证 can_run_graph 与 load_batch 使用同一个 bucket 决策，并全部包在 CP-v2 active 守卫下，符合 review 对 diff 收敛的要求。
- test/registered/cp/test_cp_strategy_unit.py（模块 图测试；类别 test；类型 test-coverage；符号 TestPrefillCPBCGReplay, test_local_capacity_overflow_uses_next_capture_bucket, test_bucket_search_preserves_two_x_padding_limit, test_bucket_search_falls_back_when_no_capture_has_capacity）：新增 TestPrefillCPBCGReplay 覆盖容量溢出选更大 bucket、多 rank 选择一致、2x 限制、无容量 fallback、load_batch padding 目标；根据 review 反馈将测试收敛到该 CP 专用文件。

关键符号：required_local_tokens, select_replay_bucket, select_replay_bucket_for_batch, can_run_graph, load_batch

关键源码片段

python/sglang/srt/layers/cp/bcg.py

核心修改文件：新增 required_local_tokens、select_replay_bucket、select_replay_bucket_for_batch 三个方法，完成从 live zigzag 布局推导 CP 本地行需求并选择容量足够的最小捕获 bucket 的全部核心逻辑。

```
# 计算 live zigzag 布局需要的最少 CP 本地物理行数，并对齐到 padding 粒度。
def required_local_tokens(self, extend_seq_lens: Any) -> Optional[int]:
    strategy = get_cp_strategy()
    # 仅在 zigzag 策略下可推算；其他策略或缺失长度时返回 None，交由调用方回退 eager。
    if not isinstance(strategy, ZigzagCPStrategy) or extend_seq_lens is None:
        return None
```

```
    cp_segment_num = strategy.cp_size * 2
    per_rank_logical_tokens = [0] * strategy.cp_size
    for raw_length in extend_seq_lens:
        base, remainder = divmod(int(raw_length), cp_segment_num)
        # zigzag 布局中 rank 与 opposite_rank 配对分摊剩余 token，逐个累加。
        for rank in range(strategy.cp_size):
            opposite_rank = cp_segment_num - 1 - rank
            per_rank_logical_tokens[rank] += (
                base * 2 + int(rank < remainder) + int(opposite_rank < remainder)
            )
    align_size = get_cp_padding_align_size()
    # 取所有 rank 中的最大需求做对齐，保证每个 CP rank 都能容纳该布局，天然形成跨 rank 共识。
    return (
        (max(per_rank_logical_tokens) + align_size - 1) // align_size * align_size
    )
```

```
# 在捕获 bucket 列表中选择第一个容量足够的更大 bucket；无则返回 None（回退 eager）。
def select_replay_bucket(
    self,
    *,
    num_tokens: int,
    required_local_tokens: int,
    capture_num_tokens: list[int],
    max_padding_factor: int,
) -> Optional[int]:
    max_num_tokens = num_tokens * max_padding_factor
    # capture_num_tokens 为升序：小于 num_tokens 的跳过，超过 2x 上限的直接终止遍历。
    for bucket in capture_num_tokens:
        if bucket < num_tokens:
            continue
        if bucket > max_num_tokens:
            break
        captured_local_tokens = self.bucket_local_tokens.get(bucket)
        # 容量未知（None）视为不可用，保持安全回退。
        if (
            captured_local_tokens is not None
            and required_local_tokens <= captured_local_tokens
        ):
            return bucket
    return None
```

python/sglang/srt/model_executor/runner/prefill_cuda_graph_runner.py

将新选择器接入图准入与重放两个入口，保证 can_run_graph 与 load_batch 使用同一个 bucket 决策，并全部包在 CP-v2 active 守卫下，符合 review 对 diff 收敛的要求。

```
# 准入阶段：CP-v2 BCG active 时，必须能找到容量足够的 bucket，否则拒绝图执行（走 eager）。
if getattr(self, "enable_cp_v2_bcg_capture", False) and is_cp_v2_active(
    forward_batch
```

```
):
    assert self.prefill_cp_bcg_input is not None
    if (
        self.prefill_cp_bcg_input.select_replay_bucket_for_batch(
            num_tokens=len(forward_batch.input_ids),
            extend_seq_lens=forward_batch.extend_seq_lens_cpu,
            capture_num_tokens=self.capture_num_tokens,
            max_padding_factor=_MAX_PREFILL_CUDA_GRAPH_PADDING_FACTOR,
        )
        is None
    ):
        return False

# 重放阶段：必须与准入使用同一选择器，保证 pad 到的 bucket 就是准入时选中的 bucket。
num_tokens = len(forward_batch.input_ids)
static_num_tokens = self._pad_to_bucket(num_tokens, self.capture_num_tokens)
if getattr(self, "enable_cp_v2_bcg_capture", False) and is_cp_v2_active(
    forward_batch
):
    assert self.prefill_cp_bcg_input is not None
    static_num_tokens = (
        self.prefill_cp_bcg_input.select_replay_bucket_for_batch(
            num_tokens=num_tokens,
            extend_seq_lens=forward_batch.extend_seq_lens_cpu,
            capture_num_tokens=self.capture_num_tokens,
            max_padding_factor=_MAX_PREFILL_CUDA_GRAPH_PADDING_FACTOR,
        )
    )
    # 准入已保证非 None；这里为时序不一致留防御性检查。
    if static_num_tokens is None:
        raise RuntimeError(
            "Prefill CUDA graph replay was admitted without a fitting bucket"
        )
```

test/registered/cp/test_cp_strategy_unit.py

新增 TestPrefillCPBCGReplay 覆盖容量溢出选更大 bucket、多 rank 选择一致、2x 限制、无容量 fallback、load_batch padding 目标；根据 review 反馈将测试收敛到该 CP 专用文件。

```
class TestPrefillCPBCGReplay(CustomTestCase):
    def _make_runner(self):
        # 用 __new__ 绕过初始化，只保留选择器需要的字段。
        runner = PrefillCudaGraphRunner.__new__(PrefillCudaGraphRunner)
        runner._is_full_backend = False
        runner.enable_lora = False
        runner._capture_chunked_prefix = False
        runner.prefill_backend_name = Backend.TC_PIECEWISE
        runner.has_mha_companion_layers = False
        runner.capture_hidden_mode = CaptureHiddenMode.NULL
        runner.capture_num_tokens = [2048, 2304]
```

```
        runner.max_num_tokens = 2304
        runner.enable_cp_v2_bcg_capture = True
        return runner

    def test_local_capacity_overflow_uses_next_capture_bucket(self):
        runner = self._make_runner()
        runner.capture_num_tokens.append(2560)
        runner.max_num_tokens = 2560
        # 模拟 capture 阶段记录的各全局 bucket 的 CP 本地行容量：2048 只有 512 行。
        runner.prefill_cp_bcg_input = PrefillCPBCGInput(
            input_embeds=torch.empty(0),
            positions=torch.empty(0),
            bucket_local_tokens={2048: 512, 2304: 576, 2560: 640},
        )
        forward_batch = self._make_forward_batch()
        self._enable_zigzag()

        with (
            patch("sglang.srt.environ.envs.SGLANG_ENABLE_CP_V2.get", return_value=True),
            patch("sglang.srt.layers.cp.bcg.get_cp_padding_align_size", return_value=8),
        ):
            selected_buckets = []
            # 最大需求机制保证 4 个 CP rank 选择一致，是跨 rank 共识的关键。
            for cp_rank in range(4):
                with get_parallel().override(attn_cp_rank=cp_rank, attn_cp_size=4):
                    selected_buckets.append(
                        runner.prefill_cp_bcg_input.select_replay_bucket_for_batch(
                            num_tokens=2048,
                            extend_seq_lens=[1534, 161, 353],
                            capture_num_tokens=runner.capture_num_tokens,
                            max_padding_factor=2,
                        )
                    )
            # 2048 捕获容量不足（512 < 520），所有 rank 都跳到 2304 捕获。
            self.assertEqual(selected_buckets, [2304, 2304, 2304, 2304])
            with get_parallel().override(attn_cp_rank=0, attn_cp_size=4):
                self.assertTrue(runner.can_run_graph(forward_batch))
```

评论区精华

作者以 review 形式对自己提交提了两条意见且均已落实：

- 测试归属：要求把 test_prefill_cuda_graph_padding.py 中的子测试移走，原文 "move this subtest to test/registered/cp/test_cp_strategy_unit.py don't modify this file"，最终改动未触碰该 padding 测试文件。
- Diff 收敛：要求 prefill_cuda_graph_runner.py 的改动尽可能小，"All the required modification should be protected under some is_cp_v2_active branches"，head 版本中 can_run_graph 与 load_batch 两处新增均被 enable_cp_v2_bcg_capture and

is_cp_v2_active(...) 守卫包裹。

- 测试子测试迁移到 CP 专用文件 (testing): 最终改动只落在 test/registered/cp/test_cp_strategy_unit.py，未修改 padding 测试文件。
- prefill_cuda_graph_runner.py 改动最小化 (design): head 版本中 can_run_graph 与 load_batch 两处新增均位于 enable_cp_v2_bcg_capture and is_cp_v2_active 守卫内。

风险与影响

- 风险：
  1. admission 与 replay 一致性风险：can_run_graph 准入和 load_batch 重放共用同一选择器，但两次调用之间 ForwardBatch 的 is_cp_v2_active 状态若发生变化，可能导致 load_batch 中触发 RuntimeError；设计上已用相同输入保障一致性，属防御性兜底。
  2. GPU 回归待确认：PR 目标测试 test/registered/cp/test_gpt_oss_4gpu_mxfp4_cp.py 在 4-gpu-b200 重跑通过，但同一轮重跑的 test/registered/cp/test_gqa_prefill_cp.py 在 4-gpu-h100 失败，需确认是否与本改动相关。
  3. 保守回退路径：extend_seq_lens_cpu 为 None 或 CP 策略非 zigzag 时，required_local_tokens 返回 None，选择器返回 None 直接拒绝图执行；行为安全，但可能比用户预期更保守。
  4. 性能影响可控：复用更大 bucket 会执行额外 padding 行，但候选受 _MAX_PREFILL_CUDA_GRAPH_PADDING_FACTOR 的 2x 限制约束；bucket 搜索为 CPU list 扫描，开销可忽略。 - 影响：影响面集中在 CP-v2 + cp_strategy=zigzag + trtllm_mha 的 prefill BCG 配置：修复该类配置下多请求 prefill 崩溃，并减少不必要的 eager fallback，提升 BCG 图复用率。prefill_cuda_graph_runner.py 属于核心执行路径，但新增逻辑被 is_cp_v2_active 守卫隔离，非 CP 用户与原有 2x padding 行为完全不变。团队侧新增了 CPU 回归测试，可在无 GPU 环境下拦截同类回归，CP 相关 GPU CI 稳定性预期提升。 - 风险标记：核心路径变更，缺少 GPU 回归确认，特定配置生效

关联脉络

- PR #33929 [misc] Remove break-graph debug log; reclaim pid-less /dev/shm leaks in CI: 同属 BCG（breakable CUDA graph）执行路径的清理与稳定性修复，说明 BCG 基础设施处于持续收敛期。
- PR #32667 [Diffusion] Add K/V-gather sequence parallel attention: 引入序列并行注意力与 zigzag 布局相关机制，是 CP BCG 布局约束的上游背景。
- PR #33953 [Diffusion] fix: scope the masked-path replicated guard to SP runs: 同为 context-parallel 场景下的 CI 崩溃修复，反映 CP/SP 多请求布局是近期稳定性的关注焦点。