PR #30584 完整报告

sgl-project/sglang

Fix diffusion BCG lifetime and add Z-Image-Turbo CI

合并时间：2026-07-10 21:20

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30584

执行摘要

- 一句话：修复 diffusion BCG 输出生命周期并新增 CI 测试
- 推荐动作：建议仔细阅读 breakable_cuda_graph.py 中的 eager_on_graph 实现，理解桥接缓冲区生命周期管理的重要性。新增的 test_eager_output_is_held_strongly_for_replay_bridge 是一个优秀的回归测试示范，值得在其他类似场景中推广。CI 的 standalone 组织方式也值得参考。

功能与动机

PR #27436 启用 diffusion BCG 后，Z-Image-Turbo 在禁用 torch.compile/offload 时出现 segfault，根因是 eager break 输出被弱引用，replay 时复制到已释放 CUDA 地址。此 PR 保持强引用修复该问题，并添加 CI 测试预防回归。

实现拆解

1. 修复桥接缓冲区生命周期：在 breakable_cuda_graph.py 的 eager_on_graph 装饰器中，将 captured_output 从弱引用 (_weak_ref_if_tensor(output)) 改为强引用 (output)，确保 replay 时 bridge buffer 地址有效。
2. 增强弱引用和复制函数：_weak_ref_if_tensor 和 _copy_output 新增对 tuple/list 的递归支持，使这些工具能正确处理更复杂的输出结构。
3. 修复 Qwen attention mask 元数据：在 qwen_image.py 中添加 _attn_mask_meta_local_pad 辅助函数，正确区分 DynamicVarlenMaskMeta 类型与普通 dict，避免 BCG 场景下 SP padding 计算错误。
4. 新增回归测试：在 test_breakable_cuda_graph.py 中添加 test_eager_output_is_held_strongly_for_replay_bridge，直接验证 eager 输出被强引用。在 test_diffusion_bcg_padding.py 中添加 test_qwen_dynamic_varlen_meta_is_not_tail_pad_meta 和 test_image_generation_models_are_registered_as_bcg_supported。
5. 端到端 CI 测试：新增 test_diffusion_bcg_zimage_turbo.py，通过 sglang generate 命令真实运行 Z-Image-Turbo BCG 生成，检查 native backend、BCG capture 成功、像素生成成功。并在 GitHub Actions 中添加独立的 bcg-diffusion job（H100），配置为 STANDALONE。

关键文件：

- python/sglang/srt/model_executor/runner_backend_utils/breakable_cuda_graph/breakable_cuda_graph.py（模块 调度器；类别 source；类型 core-logic；符号 _weak_ref_if_tensor, _copy_output, eager_on_graph）：核心修复文件：调整 eager 输出引用策略（弱→强），增强 _weak_ref_if_tensor 和 _copy_output 对 tuple/list 的支持。
- python/sglang/multimodal_gen/test/single_test_file/test_diffusion_bcg_zimage_turbo.py（模块 测试；类别 test；类型 test-coverage；符号 TestDiffusionBCGZImageTurbo, test_zimage_turbo_true_bcg_generate）：新增端到端 BCG 测试文件，验证真实 Z-Image-Turbo 模型在 BCG 模式下完整生成流水线。
- python/sglang/multimodal_gen/runtime/models/dits/qwen_image.py（模块 模型；类别 source；类型 data-contract；符号 _attn_mask_meta_local_pad）：修复 Qwen 模型在 BCG 场景下的 attention mask 元数据判空逻辑，添加 _attn_mask_meta_local_pad 辅助函数。
- python/sglang/multimodal_gen/test/unit/test_diffusion_bcg_padding.py（模块 测试；类别 test；类型 test-coverage；符号 test_qwen_dynamic_varlen_meta_is_not_tail_pad_meta, test_image_generation_models_are_registered_as_bcg_supported）：新增两个单元测试：验证 _attn_mask_meta_local_pad 行为、验证 BCG 支持的模型注册表。
- test/registered/cuda_graph/breakable/test_breakable_cuda_graph.py（模块 测试；类别 test；类型 test-coverage；符号 test_eager_output_is_held_strongly_for_replay_bridge, scale）：添加回归测试 test_eager_output_is_held_strongly_for_replay_bridge，验证 eager 输出被强引用。
- .github/workflows/pr-test-multimodal-gen.yml（模块 部署；类别 infra；类型 infrastructure）：新增 bcg-diffusion standalone job，运行在 1-gpu-h100，执行端到端 BCG 测试。

关键符号：_weak_ref_if_tensor, _copy_output, eager_on_graph, _attn_mask_meta_local_pad, test_zimage_turbo_true_bcg_generate, test_eager_output_is_held_strongly_for_replay_bridge, test_qwen_dynamic_varlen_meta_is_not_tail_pad_meta, test_image_generation_models_are_registered_as_bcg_supported

关键源码片段

python/sglang/srt/model_executor/runner_backend_utils/breakable_cuda_graph/breakable_cuda_graph.py

核心修复文件：调整 eager 输出引用策略（弱→强），增强 _weak_ref_if_tensor 和 _copy_output 对 tuple/list 的支持。

```
def _weak_ref_if_tensor(x):
    """Return a weak-ref tensor view for tensors; recursively handle tuples/lists."""
    if torch.is_tensor(x):
        from sglang.srt.compilation.weak_ref_tensor import weak_ref_tensors
        return weak_ref_tensors(x)
    # 新增递归支持 tuple/list，确保中间层输出也能正确弱引用
    if isinstance(x, tuple):
```

```
        return tuple(_weak_ref_if_tensor(e) for e in x)
    if isinstance(x, list):
        return [_weak_ref_if_tensor(e) for e in x]
    return x


def eager_on_graph(enable: bool):
    def decorator(inner: Callable):
        def wrapper(*args, **kwargs):
            ...
            output = inner(*args, **kwargs)
            # === 修复关键：将弱引用改为强引用 ===
            # 之前的代码使用 captured_output = _weak_ref_if_tensor(output)，
            # 导致 bridge buffer 在 replay 时指向已释放内存。
            # 现在直接保留强引用，确保下一个 segment 的输入地址有效。
            captured_output = output

            def replay_fn():
                new_out = captured_inner(*captured_args, **captured_kwargs)
                return _copy_output(captured_output, new_out)

            capture.cuda_graph._break_fns.append(replay_fn)
            ...
        return wrapper
    return decorator
```

python/sglang/multimodal_gen/test/single_test_file/test_diffusion_bcg_zimage_turbo.py

新增端到端 BCG 测试文件，验证真实 Z-Image-Turbo 模型在 BCG 模式下完整生成流水线。

```
class TestDiffusionBCGZImageTurbo(CustomTestCase):
    def test_zimage_turbo_true_bcg_generate(self):
        # 构建完整 sglang generate 命令，启用 BCG 并禁用 torch.compile/offload
        cmd = [
            "sglang", "generate",
            "--backend", "sglang",
            "--model-path", DEFAULT_SMALL_MODEL_NAME_FOR_TEST,
            "--prompt", "...",
            "--width", "512", "--height", "512",
            "--enable-breakable-cuda-graph",
            "--bcg-text-buckets", "128",
            "--enable-torch-compile", "false",
            "--dit-layerwise-offload", "false",
            "--dit-cpu-offload", "false",
            "--perf-dump-path", str(perf_path),
        ]
        result = subprocess.run(cmd, ..., timeout=300)
        # 断言：成功退出、使用 native 后端、BCG capture 成功、像素生成成功
        self.assertEqual(result.returncode, 0)
```

```
        self.assertNotIn("Falling back to diffusers backend", result.stdout)
        self.assertNotIn("[Diffusion BCG] capture failed", result.stdout)
        self.assertIn("[Diffusion BCG] captured", result.stdout)
        self.assertIn("Pixel data generated successfully", result.stdout)
        # 验证 perf dump 包含 DenoisingStage
        perf = json.loads(perf_path.read_text())
        self.assertIn("DenoisingStage", {s.get("name") for s in perf.get("steps", [])})
```

评论区精华

唯一的 review 讨论围绕测试组织：mickqian 建议将 bcg-diffusion 测试加入 STANDALONE_FILES 以避免 fallback-estimate 噪声。BBuf 已采纳并在 commit 684b896ef0 中实现，同时保留了 run_suite.py --suite bcg-diffusion 的兼容性。

- bcg-diffusion 测试组织方式 (testing): BBuf 采纳并修改：移动 bcg-diffusion 到 STANDALONE_FILES，保留空 parametrized 入口以维持 run_suite.py 兼容。

风险与影响

- 风险：
  - 强引用内存泄漏风险：保持强引用虽修复了段错误，但若桥接缓冲区在 capture 后不再使用，强引用会阻止其释放。但 bridge buffer 本质是下一个 segment 的输入，在完整 graph replay 完毕前必须存活，因此当前改动合理。
  - 新CIjob稳定性：bcg-diffusionjob运行真实Z-Image-Turbo生成，可能因GPU资源、环境变量等不稳定，但作为 standalone 测试可控制失败影响。
  - Qwen 辅助函数影响范围：_attn_mask_meta_local_pad 替换了内联表达式，仅影响 SP 场景下的 padding 计算，功能等价，风险低。
- 影响：
  - 用户影响：修复了 Z-Image-Turbo 用户启用 BCG 时的崩溃问题；Qwen 模型 BCG 路径也能正确处理掩码元数据。
  - 系统影响：CI 新增 standalone job，增加总测试时长但隔离失败域。
  - 团队影响：未来修改 BCG 相关代码时，必须确保 eager 输出桥接缓冲区的引用策略正确，新增的回归测试提供了防护。
  - 风险标记：核心路径变更，新 CI 依赖 GPU 环境

关联脉络

- PR #27436 [diffusion] Enable breakable CUDA graph (BCG) for diffusion DiTs: 该 PR 首次引入 diffusion BCG，但引入了本 PR 修复的 lifetime bug。本 PR 基于其代码基线修复问题。
- PR #30016 [perf] Enable torch.compile by default when performance-mode speed: 该 PR 改变了 performance_mode 语义，使 BCG 测试环境需要显式禁用 torch.compile，本 PR 的测试命令中包含 --enable-torch-compile false 以应对。