PR #27659 完整报告

sgl-project/sglang

Share BCG output buffers across capture sizes

合并时间：2026-06-10 11:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27659

执行摘要

- 一句话：共享 BCG 输出缓冲区减少显存占用
- 推荐动作：值得精读，了解 BCG runner 内部机制和显存优化技巧。设计上通过递归辅助函数处理嵌套输出结构，代码可维护性较好；建议后续增加单元测试覆盖输出为 None 及结构变化的场景。

功能与动机

在捕获多个 token 大小的 BCG 时，每个尺寸都持有独立的输出缓冲区，导致显存浪费。PR body 明确指出『Reuse one maximum-size output buffer across breakable CUDA graph capture sizes』，目的是减少显存开销。

实现拆解

1. 在 _capture_all 中引入共享缓冲区变量：在遍历 capture_num_tokens 前初始化 shared_output_buffer = None，首次捕获后保存最大尺寸对应的 buffer。
2. 修改 _capture_one 签名与逻辑：新增 shared_output_buffer 参数。若传入非空共享缓冲区，则捕获后在内部调用 _copy_output_to_buffer 将本次捕获的输出（可能来自较小尺寸的 graph）拷贝到共享缓冲区中；否则直接返回新分配的 buffer。
3. 新增 _slice_output 方法：递归支持 torch.Tensor、PPProxyTensors、tuple、list 四种类型的切片操作，并正确处理 None。
4. 新增 _copy_output_to_buffer 方法：递归执行结构化的拷贝操作，确保输出与缓冲区的类型和结构一致；对于 PPProxyTensors 还会校验键集合是否一致；处理 None 输出时若两者均为 None 则直接返回。
5. 类型导入调整：from typing import Any 以支持 _slice_output 的 output: Any 注解。

关键文件：

- python/sglang/srt/model_executor/breakable_cuda_graph_runner.py（模块 调度器；类别 source；类型 data-contract；符号 _capture_one, _slice_output, _copy_output_to_buffer, _capture_all）：核心变更文件，包含共享缓冲区逻辑、两个新辅助方法以及 _capture_one 和 _capture_all 的修改。

关键符号：_capture_all, _capture_one, _slice_output, _copy_output_to_buffer

关键源码片段

python/sglang/srt/model_executor/breakable_cuda_graph_runner.py

核心变更文件，包含共享缓冲区逻辑、两个新辅助方法以及 _capture_one 和 _capture_all 的修改。

```
# 在 _capture_all 中引入共享输出缓冲区变量
shared_output_buffer = None
for num_tokens in capture_range:
    graph, output = self._capture_one(
        num_tokens, pool, stream, shared_output_buffer
    )
    if shared_output_buffer is None:
        shared_output_buffer = output # 首个捕获结果作为共享缓冲区
    self.graphs[num_tokens] = graph
    self.output_buffers[num_tokens] = output

# 递归：对输出进行切片（返回视图）
def _slice_output(self, output: Any, num_tokens: int) -> Any:
    if output is None:
        return None # 处理可选 None 输出
    if torch.is_tensor(output):
        return output[:num_tokens]
    if isinstance(output, PPProxyTensors):
        return output[:num_tokens]
    if isinstance(output, tuple):
        return tuple(self._slice_output(item, num_tokens) for item in output)
    if isinstance(output, list):
        return [self._slice_output(item, num_tokens) for item in output]
    raise TypeError(f"Unsupported BCG output type: {type(output)}")

# 递归：将小尺寸输出拷贝到共享缓冲区
def _copy_output_to_buffer(
    self, output: Any, output_buffer: Any, num_tokens: int
) -> None:
    if output is None or output_buffer is None:
        if output is None and output_buffer is None:
            return
        raise ValueError(
            "BCG output structure changed between capture sizes: "
            f"{type(output)} vs {type(output_buffer)}"
        )
    if torch.is_tensor(output) and torch.is_tensor(output_buffer):
        output_buffer[:num_tokens].copy_(output[:num_tokens])
        return
    if isinstance(output, PPProxyTensors) and isinstance(
        output_buffer, PPProxyTensors
    ):
        if output.tensors.keys() != output_buffer.tensors.keys():
            raise ValueError(
```

```
                "BCG output proxy structure changed: "
                f"{output.tensors.keys()} != {output_buffer.tensors.keys()}"
            )
        for key, tensor in output.tensors.items():
            self._copy_output_to_buffer(
                tensor, output_buffer.tensors[key], num_tokens
            )
        return
    if isinstance(output, (list, tuple)) and isinstance(output_buffer, type(output)):
        if len(output) != len(output_buffer):
            raise ValueError(
                "BCG output sequence structure changed: "
                f"{len(output)} != {len(output_buffer)}"
            )
        for item, buffer in zip(output, output_buffer):
            self._copy_output_to_buffer(item, buffer, num_tokens)
        return
    raise TypeError(
        "Unsupported BCG output buffer pair: "
        f"{type(output)} vs {type(output_buffer)}"
    )
```

评论区精华

来自 gemini-code-assist[bot] 的两条 review 评论均指出模型中某些 forward 输出可能为 None（如可选输出），建议在 _slice_output 和 _copy_output_to_buffer 中优先检查 None。作者在第二个 commit 中已采纳该建议，添加了相应的 None 处理分支。Fridge003 最终批准了此 PR。

- 处理 None 输出 (correctness): 作者在第二个 commit 中添加了 None 处理分支。

风险与影响

- 风险：核心风险在于 _copy_output_to_buffer 中的结构一致性校验：若不同 token 尺寸下模型 forward 返回的结构（如 PPProxyTensors 的键、元组长度）不同，将抛出 ValueError 导致捕获失败。此外，_slice_output 对 PPProxyTensors 的切片依赖于其 __getitem__ 实现，若未来 PPProxyTensors 更改语义可能导致非预期行为。
- 影响：直接影响 BCG 捕获阶段的显存占用，对多 token 尺寸场景有显著优化；不影响 BCG 回放路径的用户体验或精度。修改仅限 breakable_cuda_graph_runner.py 一个文件，影响面小且易验证。
- 风险标记：结构一致性校验可能失败，缺少单元测试覆盖

关联脉络

- PR #27758 Revert "Share BCG output buffers across capture sizes": 此 PR 回退了当前 PR 的更改，可能因未预料的兼容性问题；当前 PR 的优化方案在重新评估后再次提交。

- PR #23906 [Refactor] Cuda Graph Runner/Backend Refactor: 与 CUDA Graph Runner 重构相关，本 PR 的 BCG 输出缓冲区共享是该重构系列的后续优化。