PR #28973 完整报告

sgl-project/sglang

[Refactor] Share CUDA graph memory pool across prefill and decode

合并时间：2026-06-25 01:18

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28973

执行摘要

- 一句话：共享 prefill 和 decode 的 CUDA graph 内存池
- 推荐动作：值得精读。该 PR 展示了如何通过共享全局资源来减少内存占用，且改动极小（+26/-8）。适合作为类似共享池模式（如 EAGLE 的 draft pool）的参考。

功能与动机

自 PR#23906 重构后，prefill 和 decode 的 graph 后端各自独立分配 graph_pool_handle()，导致每个被捕获的 graph 独立预留内存。而全局辅助函数 _global_graph_memory_pool 闲置未用。PR body 指出："The two phases never replay concurrently — so sharing reserves only the larger phase's footprint instead of the sum."

实现拆解

1. 在 python/sglang/srt/model_executor/runner_utils/pool.py 中新增 get_or_create_global_graph_memory_pool(device_module) 函数：首次调用时通过 device_module.graph_pool_handle() 创建 pool，后续复用已创建的句柄。
2. 修改 FullCudaGraphBackend（full_cuda_graph_backend.py）：在其 capture_session() 方法中将原来的 self._pool = self._device_module.graph_pool_handle() 替换为调用 get_or_create_global_graph_memory_pool(self._device_module)。
3. 修改 BreakableCudaGraphBackend（breakable_cuda_graph_backend.py）：同样在 capture_session() 中替换为共享 pool。
4. 修改 TcPiecewiseCudaGraphBackend（tc_piecewise_cuda_graph_backend.py）：在 _run_compile_pass() 方法中将 self._pool = self._device_module.graph_pool_handle() 替换为共享 pool。 所有后端都新增导入 get_or_create_global_graph_memory_pool。

关键文件：

- python/sglang/srt/model_executor/runner_utils/pool.py（模块 内存池；类别 source；类型 data-contract；符号 get_or_create_global_graph_memory_pool）：新增 get_or_create_global_graph_memory_pool 函数，是共享 pool 的核心入口。
- python/sglang/srt/model_executor/runner_backend/full_cuda_graph_backend.py（模块 后端执行器；类别 source；类型 data-contract）：FullCudaGraphBackend 的 capture_session 改为使用共享 pool，是三个后端之一。

- python/sglang/srt/model_executor/runner_backend/breakable_cuda_graph_backend.py（模块 后端执行器；类别 source；类型 data-contract）：BreakableCudaGraphBackend 的 capture_session 改为使用共享 pool，是三个后端之二。
- python/sglang/srt/model_executor/runner_backend/tc_piecewise_cuda_graph_backend.py（模块 后端执行器；类别 source；类型 data-contract）：TcPiecewiseCudaGraphBackend 的 _run_compile_pass 改为使用共享 pool，是三个后端之三。

关键符号：get_or_create_global_graph_memory_pool, FullCudaGraphBackend.capture_session, BreakableCudaGraphBackend.capture_session, TcPiecewiseCudaGraphBackend._run_compile_pass

关键源码片段

python/sglang/srt/model_executor/runner_backend/full_cuda_graph_backend.py

FullCudaGraphBackend 的 capture_session 改为使用共享 pool，是三个后端之一。

```
# python/sglang/srt/model_executor/runner_backend/full_cuda_graph_backend.py
# ...（导入部分）
from sglang.srt.model_executor.runner_utils.pool import (
    get_or_create_global_graph_memory_pool,
)
# ...
class FullCudaGraphBackend(BaseCudaGraphBackend):
    # ...
    @contextmanager
    def capture_session(self, stream: torch.cuda.Stream):
        if self._pool is None:
            # Before: self._pool = self._device_module.graph_pool_handle()
            self._pool = get_or_create_global_graph_memory_pool(self._device_module)
        set_graph_pool_id(self._pool)
        self._capture_stream = stream
        try:
            yield
        finally:
            self._capture_stream = None
```

评论区精华

PR 无 review 评论。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。变更仅涉及 graph capture 阶段的内存池分配方式，不改变 graph 捕获或回放的逻辑。共享 pool 仅在 prefill 和 decode 不同时回放时安全——当前设计确实如此。

潜在的回归风险在于：若未来引入并发回放（如同步执行 prefill 和 decode），则共享pool可能导致竞争；但当前架构无此场景。

- 影响：直接影响 CUDA graph 捕获时的 GPU 内存占用，prefill 和 decode 的总预留内存将从两者之和减少为最大值。对用户无功能影响，但可降低 OOM 风险。对团队而言，这是一个低风险、高收益的内存优化。
- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #23906 [Refactor] Refactor CUDA graph runner/backend: 该 PR 引入了 prefill 和 decode 各自独立分配 graph_pool_handle，本 PR 修复了其内存浪费问题。