PR #33371 完整报告

sgl-project/sglang

Fix BCG circular import during server startup

合并时间：2026-08-03 16:53

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33371

执行摘要

- 一句话：修复 BCG 循环导入导致的服务器启动崩溃
- 推荐动作：值得快速精读，理解 `bcg.py 与 runner 包之间的导入拓扑。核心设计决策是「把运行时依赖延迟到调用点」，这是打破 Python 循环导入的经典做法。建议阅读时结合 #33136 的 BCG 整体设计，并留意后续是否补充循环导入的回归测试。

功能与动机

PR body 明确描述这是 #33136 引入的启动阻塞回归：ImportError: cannot import name 'PrefillCPBCGInput' from partially initialized module 'sglang.srt.layers.cp.bcg'。根因是 bcg.py 顶层导入 shape_key 时，会触发 runner/__init__.py 的执行，而 runner 又反过来通过 prefill runner 导入 bcg，形成循环依赖，导致所有使用默认 prefill BCG 后端的 CUDA 模型在服务器启动前就崩溃。

实现拆解

实现分为 3 步：

1. 移除顶层导入：在 python/sglang/srt/layers/cp/bcg.py 顶部删除 from sglang.srt.model_executor.runner.shape_key import ShapeKey，避免模块加载阶段触发 runner 子模块导入。
2. 延迟导入到函数内：在 *execute_prefill_cp_bcg* 函数体的第一行添加局部导入 from sglang.srt.model_executor.runner.shape_key import ShapeKey。此时模块加载已完成，replay 阶段导入不会形成循环，且 Python 的模块缓存保证每次调用开销可忽略。
3. 保留类型检查引用：TYPE_CHECKING 块中原有的 PrefillCudaGraphRunner 等类型引用不受影响，ShapeKey 仅在实际 replay 路径使用，不参与运行时类型检查。

测试方面：本次变更没有新增或修改测试文件，依赖现有 CI 覆盖；作者通过 /tag-and-rerun-ci 重跑了 CI。

关键文件：

- python/sglang/srt/layers/cp/bcg.py（模块 图调度；类别 source；类型 dependency-wiring；符号 *execute_prefill_cp_bcg*）：唯一变更文件，修复由 #33136 导致的循环导入，是启动阻塞 bug 的直接修复点。

关键符号：*execute_prefill_cp_bcg*, supports_prefill_cp_bcg

关键源码片段

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

唯一变更文件，修复由 #33136 导致的循环导入，是启动阻塞 bug 的直接修复点。

```
# python/sglang/srt/layers/cp/bcg.py（head 版本关键位置）

# 模块顶层不再导入 ShapeKey。此前顶层导入会在 server 参数解析阶段触发
# runner 子模块导入，进而通过 prefill runner 回调导入本模块，形成循环。
# TYPE_CHECKING 块只用于类型检查，运行时不会执行导入。
if TYPE_CHECKING:
    from sglang.srt.model_executor.forward_batch_info import ForwardBatch
    from sglang.srt.model_executor.runner.prefill_cuda_graph_runner import (
        PrefillCudaGraphRunner,
    )
    from sglang.srt.server_args import ServerArgs


def execute_prefill_cp_bcg(
    runner: PrefillCudaGraphRunner,
    forward_batch: ForwardBatch,
    static_forward_batch: ForwardBatch,
    static_num_tokens: int,
    raw_num_tokens: int,
    **kwargs,
):
    """Replay a CP-local body and run the global gather/logits tail eagerly."""

    # 关键修复：把 ShapeKey 的导入延迟到 BCG replay 实际执行时。
    # 此时 runner 初始化已完成，再导入 runner 子模块不会反向触发 bcg 的导入。
    # Python 会缓存已加载模块，因此每次调用的额外开销可以忽略。
    from sglang.srt.model_executor.runner.shape_key import ShapeKey

    cp_input = runner.prefill_cp_bcg_input
    assert cp_input is not None
    model = runner.model_runner.model
    with runner._prefill_forward_context(
        static_forward_batch,
        num_tokens=static_num_tokens,
        raw_num_tokens=raw_num_tokens,
    ):
        local_output = runner.backend.replay(
            ShapeKey(size=static_num_tokens), # 延迟导入的 ShapeKey 在此处使用
            static_forward_batch,
            **kwargs,
        )
        local_output = _slice_output_rows(local_output, cp_input.live_local_tokens)
        # 后续为 CP gather 与 logits tail 的 eager 处理，省略
```

评论区精华

本 PR 没有实质性的 review 评论区讨论。仅有两条非技术评论：一条是 gemini-code-assist 的自动提醒（Gemini Code Assist 已停用），另一条是作者触发的 /tag-and-rerun-ci 命令。循环导入的根因与修复思路已在代码注释中说明，未产生设计争议。

- 暂无高价值评论线程

风险与影响

- 风险：风险极低：
  - 延迟导入把 shape_key 的加载推迟到 BCG replay 首次执行时，若 runner 初始化流程有变化可能导致首次 replay 时额外 import 耗时，但 Python 模块缓存使其可忽略。
  - 该修复只影响 *execute_prefill_cp_bcg* 的调用路径，不影响 BCG 的捕获、过滤等其他逻辑。
  - 缺少针对该循环导入的直接回归测试，未来若重构导入结构可能再次引入同类问题。
  - 影响：影响范围：所有启用 prefill CP + zigzag 策略 + TRT-LLM MHA 的 CUDA 模型启动场景（即 #33136 支持的 breakable CUDA 图路径）。修复前这些场景会在参数解析阶段直接崩溃，修复后可正常启动。对不使用 BCG 的场景无任何运行时影响。团队维护成本低，单文件 6 行改动，易于 review。
  - 风险标记：启动路径变更，缺少直接回归测试

关联脉络

- PR #33136 [CP] Support breakable CUDA graphs for zigzag strategy: 本 PR 修复的就是 #33136 引入的循环导入，是直接关联的回归来源。
- PR #33137 [CP] Fuse zigzag attention into a single call: 与 #33136 同为 zigzag CP 预填优化系列，共享 bcg/zigzag 相关代码路径。