PR #23906 完整报告

sgl-project/sglang

[Refactor] Cuda Graph Runner/Backend Refactor

合并时间：2026-06-10 12:36

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23906

执行摘要

- 一句话：重构 CUDA Graph Runner/Backend 分层架构
- 推荐动作：值得深度精读。重点关注 BaseCudaGraphBackend 接口设计（如何平衡通用性与灵活性）、CudaGraphConfig 配置数据类（Phase/Backend 枚举设计、diff-based 导出）以及 BaseCudaGraphRunner 与后端之间的 capture_session/replay_session 上下文管理。这些设计模式可直接应用于其他硬件后端或自定义捕获策略。

功能与动机

现有三个 CUDA 图实现（full cuda graph, breakable cuda graph, torch-compile-based piecewise cuda graph）之间存在大量重复代码，且每个策略需要独立的 Runner 实现。RFC Issue #23004 提出通过 Runner-Backend 分离消除重复，并支持灵活的 per-phase 后端选择（如 decode 使用 full，prefill 使用 breakable/tc_piecewise）。

实现拆解

1. 配置层迁移：创建 model_executor/cuda_graph_config.py，定义 Phase 和 Backend 枚举类，PhaseConfig 和 CudaGraphConfig 数据类。重构 ServerArgs，将原先分散的 disable_cuda_graph、enable_breakable_cuda_graph 等字段归一为 cuda_graph_config，并支持 JSON 和便捷 CLI 两种输入方式。
2. 后端接口定义：在 runner_backend/ 下新增 BaseCudaGraphBackend 抽象基类，声明 capture_session、capture_one、can_run、replay_session、replay、cleanup 等契约方法。三种具体后端（Full、Breakable、TC_Piecewise）分别实现该接口。
3. 阶段运行器重构：创建 BaseCudaGraphRunner 基类，提供 can_run、capture、capture_one_shape、replay_prepare、replay 等框架。PrefillCudaGraphRunner 和 DecodeCudaGraphRunner 继承自该基类，并在初始化时通过 resolve_prefill_backend/resolve_decode_backend 工厂函数绑定对应后端。
4. 旧文件删除与重命名：删除 piecewise_cuda_graph_runner.py（860 行）和 breakable_cuda_graph_runner.py（541 行），将 cuda_graph_runner.py 迁移并重命名为 decode_cuda_graph_runner.py（新增 294 行 / 删除 463 行）。
5. NPU 硬件支持：新增 NPUCudaGraphBackend，集成 NPU 的 torch.npu.NPUGraph，并在 capture_one 中支持 post_warmup_hook 回调以满足后端接口对齐。

6. 测试与配置工具：迁移测试文件到新目录，更新 mock_server_args.py 以支持新配置结构，新增 CudaGraphBufferRegistry 等 buffer 管理类。

关键文件：

- python/sglang/srt/model_executor/runner/prefill_cuda_graph_runner.py（模块 预填充运行器；类别 source；类型 core-logic；符号 PrefillCudaGraphRunner, init, _is_mamba_track_enabled, _cache_loc_dtype）：新增核心文件：PrefillCudaGraphRunner，负责 prefill 阶段的 CUDA 图捕获与重放，通过 resolve_prefill_backend 绑定后端（默认 TcPiecewiseCudaGraphBackend）。
- python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py（模块 解码运行器；类别 source；类型 rename-or-move；符号 _make_graph_key, get_is_capture_mode, compile_in_capture_mode, model_capture_mode）：原 cuda_graph_runner.py 重命名并重构为 DecodeCudaGraphRunner，负责 decode/TARGET_VERIFY/DLLM_EXTEND 阶段的 CUDA 图捕获。
- python/sglang/srt/model_executor/cuda_graph_config.py（模块 配置模型；类别 source；类型 data-contract；符号 Phase, Backend, PhaseConfig, CudaGraphConfig）：新增配置数据模型：Phase/Backend 枚举、PhaseConfig/CudaGraphConfig dataclass，以及 default、parse、check_cuda_graph_backend 辅助函数。
- python/sglang/srt/model_executor/runner/base_cuda_graph_runner.py（模块 运行器基类；类别 source；类型 data-contract；符号 freeze_gc, get_batch_sizes_to_capture, BaseCudaGraphRunner, init）：新增抽象基类 BaseCudaGraphRunner，定义了 can_run、capture、capture_one_shape、replay_prepare、replay 等方法框架，并包含 freeze_gc、get_batch_sizes_to_capture 工具函数。
- python/sglang/srt/model_executor/piecewise_cuda_graph_runner.py（模块 旧运行器；类别 source；类型 deletion；符号 freeze_gc, _to_torch, patch_model, get_global_graph_memory_pool）：被删除的核心文件：原 PiecewiseCudaGraphRunner（860 行）被新 PrefillCudaGraphRunner + TcPiecewiseCudaGraphBackend 替代。
- python/sglang/srt/model_executor/breakable_cuda_graph_runner.py（模块 旧运行器；类别 source；类型 deletion；符号 BreakableCudaGraphRunner, init, _has_inactive_dp_rank, _init_buffers）：被删除的核心文件：原 BreakableCudaGraphRunner（541 行）被 PrefillCudaGraphRunner + BreakableCudaGraphBackend 替代。
- python/sglang/srt/model_executor/runner_backend/tc_piecewise_cuda_graph_backend.py（模块 后端实现；类别 source；类型 core-logic；符号 _toggle_multi_platform_ops, TcPiecewiseCudaGraphBackend, init, build_compilation_config）：新增核心后端：TcPiecewiseCudaGraphBackend，基于 torch.compile 的 piecewise CUDA 图捕获，支持 eager/inductor 编译器。
- python/sglang/srt/hardware_backend/npu/graph_runner/npu_cudagraph_backend.py（模块 NPU 后端；类别 source；类型 core-logic；符号 NPUCudaGraphBackend, init, capture_session, capture_one）：新增 NPU 后端适配：NPUCudaGraphBackend 继承 BaseCudaGraphBackend，使用 torch.npu.NPUGraph 捕获，处理了 post_warmup_hook 对齐。

关键符号：BaseCudaGraphRunner.can_run, BaseCudaGraphRunner.capture, BaseCudaGraphRunner.capture_one_shape, BaseCudaGraphRunner.replay_prepare, BaseCudaGraphRunner.replay, BaseCudaGraphBackend.capture_session, BaseCudaGraphBackend.capture_one, BaseCudaGraphBackend.replay_session, BaseCudaGraphBackend.replay, FullCudaGraphBackend.capture_one, BreakableCudaGraphBackend.capture_one, TcPiecewiseCudaGraphBackend.capture_one, CudaGraphConfig.from_dict, CudaGraphConfig.to_dict, PrefillCudaGraphRunner._run_forward, DecodeCudaGraphRunner._make_graph_key

关键源码片段

python/sglang/srt/model_executor/cuda_graph_config.py

新增配置数据模型：Phase/Backend 枚举、PhaseConfig/CudaGraphConfig dataclass，以及 default、parse、check_cuda_graph_backend 辅助函数。

```
# python/sglang/srt/model_executor/cuda_graph_config.py
# 依赖纯 stdlib，便于 ServerArgs 导入而不拉入 torch/srt 后端类。

class Phase:
    """模型forward的两个阶段。"""
    DECODE = "decode"
    PREFILL = "prefill"
    ALL = (DECODE, PREFILL)

class Backend:
    """每个阶段可使用的CUDA图捕获后端。"""
    FULL = "full"
    BREAKABLE = "breakable"
    TC_PIECEWISE = "tc_piecewise"
    DISABLED = "disabled"
    ALL = (FULL, BREAKABLE, TC_PIECEWISE, DISABLED)

# 每个阶段允许的后端不一样：prefill 不允许 full（形状可变）。
ALLOWED_BACKENDS_PER_PHASE = {
    Phase.DECODE: (Backend.FULL, Backend.BREAKABLE, Backend.TC_PIECEWISE, Backend.
    DISABLED),
    Phase.PREFILL: (Backend.BREAKABLE, Backend.TC_PIECEWISE, Backend.DISABLED),
}

@dataclass
class PhaseConfig:
    """每个阶段的CUDA图设置：后端、最大batch size、捕获batch size列表、torch.
    compile编译器。"""
    backend: str = Backend.DISABLED
    max_bs: Optional[int] = None
    bs: Optional[List[int]] = None
    # 仅当 backend == tc_piecewise 时有效。
    tc_compiler: str = "eager"
```

```
@dataclass
class CudaGraphConfig:
    """顶层CUDA图配置：decode 和 prefill 各一个 PhaseConfig。"""
    decode: PhaseConfig = field(default_factory=lambda: PhaseConfig(backend=Backend.FULL))
    prefill: PhaseConfig = field(default_factory=lambda: PhaseConfig(backend=Backend.TC_
    PIECEWISE))

    def __getitem__(self, phase: str) -> PhaseConfig:
        if phase not in Phase.ALL:
            raise KeyError(phase)
        return getattr(self, phase)
```

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

新增抽象基类 BaseCudaGraphRunner，定义了 can_run、capture、capture_one_shape、replay_prepare、replay 等方法框架，并包含 freeze_gc、get_batch_sizes_to_capture 工具函数。

```
# python/sglang/srt/model_executor/runner/base_cuda_graph_runner.py
from abc import ABC, abstractmethod

@contextmanager
def freeze_gc(enable_cudagraph_gc: bool):
    """优化CUDA图捕获期间的垃圾回收。先collect然后冻结剩余对象。"""
    gc.collect()
    should_freeze = not enable_cudagraph_gc
    if should_freeze:
        gc.freeze()
    try:
        yield
    finally:
        if should_freeze:
            gc.unfreeze()
            gc.collect()

class BaseCudaGraphRunner(ABC):
    """CUDA图运行器抽象基类。子类（Decode/Prefill）拥有一个Backend处理捕获/回放机制。"""
    # 子类在 capture() 前必须设置：
    buffers: ForwardInputBuffers
    backend: BaseCudaGraphBackend

    def __init__(self, model_runner: ModelRunner) -> None:
        self.model_runner = model_runner
        self.device = model_runner.device
        self.device_module = torch.get_device_module(self.device)
        self.tp_size = model_runner.server_args.tp_size
        self.dp_size = model_runner.server_args.dp_size
        self.pp_size = model_runner.server_args.pp_size
        self.attn_tp_size = get_attention_tp_size()
        self.attn_tp_rank = get_attention_tp_rank()
```

```
        self.tbo_plugin = TboCudaGraphRunnerPlugin()

    @abstractmethod
    def can_run(self, forward_batch: ForwardBatch) -> bool:
        """判断当前batch是否可使用已捕获的CUDA图。"""
        ...

    @abstractmethod
    def capture(self) -> None:
        """一次性捕获所有需要形状的CUDA图。"""
        ...

    @abstractmethod
    def capture_one_shape(self, size: int, ...) -> None:
        """捕获单个形状：构造dummy batch并调用backend.capture_one。"""
        ...

    @abstractmethod
    def replay_prepare(self, forward_batch: ForwardBatch, **kwargs) -> ForwardBatch:
        """填充/重排输入以匹配已捕获的形状，返回静态批次。"""
        ...

    @abstractmethod
    def replay(self, forward_batch: ForwardBatch, ...) -> ModelOutput:
        """通过捕获的CUDA图执行batch的forward。"""
        ...
```

评论区精华

- merrymercy：提出默认 prefill 应为 breakable 而非 tc_piecewise，并建议将通用定义从 cuda_graph 重命名为 device graph 以支持其它硬件。
- ch-wan：批评配置解析逻辑过于复杂，要求增加单元测试验证各种输入组合；建议将 JSON 配置解析为 dataclass 而非 dict 以利用类型提示。
- VDV1985：质疑 NPUsNPUGraphRunner 为何继承 DecodeCudaGraphRunner 而非 BaseCudaGraphRunner，询问是否意味着 NPU 不支持 prefill 图。Oasis-Git 回应称 NPU 目前仅支持 decode。
- BBuf：指出 NPUCudaGraphBackend.capture_one 未接受 post_warmup_hook 参数，与 BaseCudaGraphBackend 契约不兼容，导致 NPU decode 图捕获失败；Oasis-Git 后续修复。
- merrymercy：多次强调“不要删除有用的注释”，指出重构过程中 AI 不慎删除了描述 LoRA 阶段、DLLM 模式等关键注释，要求恢复。
- ch-wan：建议将后端上下文管理器 runtime_session 重命名为 replay_session，避免与运行阶段混淆；Oasis-Git 采纳。
  - 默认 prefill 后端应为 breakable 而非 tc_piecewise (design): 默认为 tc_piecewise（原 piecewise 行为），用户可通过 --cuda-graph-backend-prefill=breakable 切换。

- 配置解析应使用 dataclass 而非 dict (design): 已改为 dataclass，支持 decode.max_bs 等类型安全访问。
- NPU 后端 capture_one 缺少 post_warmup_hook 参数 (correctness): 已添加 post_warmup_hook 可选参数并在 NPU 后端调用。
- 重构中误删了有用注释 (style): Oasis-Git 后续恢复了部分注释，但在最终版本中仍有一些简化。merrymercy 要求保留。
- 配置解析逻辑过于复杂，需增加单元测试 (testing): ch-wan 同意未来 PR 增加测试，在当前 PR 添加 TODO 标记。

风险与影响

- 风险：
  1. 核心路径变更：CUDA 图是推理性能的核心，重构可能引入回归，尤其在多模型（DeepSeek、MLA、LoRA）与多种注意力后端的组合下。
  2. 配置迁移兼容性：大量旧 CLI 参数被 DeprecatedAction 替代，但 Python API 直接传入 ServerArgs 构造时可能因字段不存在而崩溃（需通过 cuda_graph_config 透传）。
  3. NPU 后端未完全对齐：NPUCudaGraphBackend 的 capture_one 缺少 post_warmup_hook 形参，虽然在后续修复，但 NPU 特有 replay_with_input_update 路径依赖特殊输入更新逻辑。
  4. 依赖方同步风险：EagleWorker、FrozenKvMtpWorker 等依赖旧 CudaGraphRunner 构造方式，重构后必须转为使用 DecodeCudaGraphRunner。
  5. 测试覆盖不足：配置解析逻辑复杂度高但没有单元测试，多线程 / 多流捕获场景未经充分验证。 - 影响：用户：启动命令需要调整，旧参数如 --disable-piecewise-cuda-graph 变为 --cuda-graph-config 或 --cuda-graph-backend-prefill=disabled；但 DeprecatedAction 会警告并继续工作，短期兼容。系统：架构清晰度大幅提升，新增 CUDA 图策略只需实现 Backend 子类并注册工厂，无需改动 Runner 代码。团队：维护成本降低，但需要培训确保新扩展点被正确使用。影响范围：全仓库 160 个文件变更，涵盖 ServerArgs、ModelRunner、注意力后端、EAGLE 推测解码 worker、NPU 硬件后端、CI 测试等。
- 风险标记：核心路径变更，配置迁移兼容性，NPU 后端对齐，依赖方同步风险，测试覆盖不足

关联脉络

- PR #23004 [RFC] Cuda Graph Runner Backend Refactor: 本 PR 的动机和设计方案均源自该 RFC Issue，定义了 Runner-Backend 分离的目标和步骤。
- PR #28081 [refactor] Fold FrozenKVMTPCudaGraphRunner onto the shared DecodeCudaGraphRunner base: 后续依赖本重构的 PR，将 FrozenKVMTPCudaGraphRunner 合并到共享的 DecodeCudaGraphRunner 基类，体现本重构提供的扩展性。
- PR #28093 [Spec] Move draft-extend prep to EagleDraftWorkerBase; unify prepare_for_* names: 后续依赖本重构的 speculative-decoding PR，利用统一的 Runner

基类简化 EagleWorker 代码。