PR #28386 完整报告

sgl-project/sglang

refactor(runner): add EagerRunner, own the eager path, polymorphic dispatch

合并时间：2026-06-20 04:05

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28386

执行摘要

- 一句话：将 EagerRunner 提取为独立类，实现多态分发
- 推荐动作：建议精读。本 PR 是 runner 层系统重构的重要一步，展示了如何通过多态分发分离执行路径，并统一缓冲管理。自动化审查指出的潜在问题值得在合并前仔细验证，尤其是共享缓冲池的副作用和缓冲区大小估算。建议关注后续的修复 PR（可能基于这些评论）。

功能与动机

在 SGLang 中，eager forward 路径原本分散在 ModelRunner 中，并与 CUDA graph 路径耦合。该 PR 旨在抽出独立的 EagerRunner，使得 CUDA graph runners 只需关注图执行，同时消除按需增长的缓冲注册表，改为初始化时确定大小的固定缓冲，提高可预测性。正如 PR body 所述：'The eager forward path lived inline in ModelRunner. Extract it into a dedicated runner so the cuda-graph runners stay purely cuda-graph, and back it with one fixed-size static input buffer (allocated once at init) instead of grow-on-demand registries.'

实现拆解

1. 新增 EagerRunner 类（runner/eager_runner.py）：继承 BaseRunner，__init__ 根据 speculative、dLLM 等配置计算最大 BS 和 Token 数，调用 build_eager_registry 分配固定大小的静态缓冲注册表；can_run_graph 返回 False；load_batch 将 live batch 拷贝到缓冲切片；execute 根据 forward mode 分发到 decode/extend/idle，最终调用 model.forward。
2. 新增 build_eager_registry 函数（cuda_graph_buffer_registry.py）：复用 build_decode_registry 的槽位定义，设置 share_pool=True 使得同名同大小的缓冲槽通过进程级池合并。EagerRunner 在其他 runner 之前构建，其最大尺寸分配成为 canonical，后续 CUDA graph runners 的匹配槽会共享同一物理内存。
3. 重构 ModelRunner（model_runner.py）：移除 forward_decode/forward_extend/forward_idle 等方法及 _EagerBufferRegistry 数据类；无条件创建 EagerRunner 实例；在 _forward_raw 中通过 isinstance(model_runner.current_runner, EagerRunner) 决定分发；简化 import 和辅助函数。
4. 适配 DecodeCudaGraphRunner（decode_cuda_graph_runner.py）：在 capture 方法中增加 self.buffers.seq_lens.fill_(self.seq_len_fill_value)，防止共享池中 EagerRunner 的零值影响 CUDA graph 捕获的正确性。

5. 导出 EagerRunner（runner/__init__.py）：加入 from .eager_runner import EagerRunner，保证包对外接口完整。

关键文件：

- python/sglang/srt/model_executor/runner/eager_runner.py（模块 执行器；类别 source；类型 core-logic；符号 EagerRunner, init, can_run_graph, load_batch）：新文件，定义 EagerRunner 类，封装所有非 CUDA graph 的 forward 执行路径，是本次重构的核心。
- python/sglang/srt/model_executor/model_runner.py（模块 主运行器；类别 source；类型 core-logic；符号 _EagerBufferRegistry, _ensure_eager_registry, _ensure_eager_decode_registry, _ensure_eager_prefill_registry）：主运行器，移除大量内联 eager 逻辑，改为委托给 EagerRunner，并调整执行流程。
- python/sglang/srt/model_executor/cuda_graph_buffer_registry.py（模块 缓冲区；类别 source；类型 data-contract；符号 build_eager_registry）：新增 build_eager_registry 函数，用于为 EagerRunner 构建静态缓冲注册表，并启用共享池机制。
- python/sglang/srt/model_executor/runner/decode_cuda_graph_runner.py（模块 图运行器；类别 source；类型 data-contract）：修复共享缓冲池导致的 seq_lens 污染，在 capture 前恢复期望的填充值。
- python/sglang/srt/model_executor/runner/__init__.py（模块 运行器；类别 source；类型 data-contract）：导出 EagerRunner，使其可通过包路径访问。

关键符号：EagerRunner.init, EagerRunner.can_run_graph, EagerRunner.load_batch, EagerRunner.execute, EagerRunner._execute_decode, EagerRunner._execute_extend, EagerRunner._resolve_decode_pdmux, build_eager_registry, DecodeCudaGraphRunner.capture, ModelRunner._forward_raw

关键源码片段

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

新文件，定义 EagerRunner 类，封装所有非 CUDA graph 的 forward 执行路径，是本次重构的核心。

```
class EagerRunner(BaseRunner):
    def __init__(self, model_runner: ModelRunner) -> None:
        super().__init__(model_runner)
        mr = model_runner
        sa = mr.server_args

        # 计算每 batch 的 token 数，speculative decoding 下可能多于 1
        num_tokens_per_bs = 1
        if mr.spec_algorithm.is_speculative():
            num_draft_tokens = sa.max_speculative_num_draft_tokens or 1
            if mr.is_draft_worker:
                num_tokens_per_bs = max(
                    sa.speculative_eagle_topk or 1,
                    num_draft_tokens,
                    (2 * (sa.speculative_num_steps or 0)
```

```
                if sa.enable_multi_layer_eagle else 0),
            )
        else:
            num_tokens_per_bs = mr.spec_algorithm.get_num_tokens_per_bs_for_target_verify(
                num_draft_tokens, mr.is_draft_worker)
    else:
        dllm_config = DllmConfig.from_server_args(sa)
        if dllm_config is not None:
            # dLLM 使用 block_size 个 token 每请求
            num_tokens_per_bs = dllm_config.block_size

    max_bs = mr.max_running_requests
    # Frozen-KV MTP 在 batch 轴上乘以 topk
    if (mr.is_draft_worker and mr.spec_algorithm.is_frozen_kv_mtp()
            and sa.speculative_eagle_topk > 1):
        max_bs *= sa.speculative_eagle_topk

    # 取 max(chunked_prefill_size, max_bs * num_tokens_per_bs)
    prefill_ceiling = (sa.chunked_prefill_size
                       if sa.chunked_prefill_size and sa.chunked_prefill_size > 0
                       else mr.max_total_num_tokens)
    max_num_token = max(prefill_ceiling, max_bs * num_tokens_per_bs)

    # 构建静态缓冲注册表，固定大小，share_pool=True 以共享
    self._eager_registry = build_eager_registry(
        device=mr.device,
        max_bs=max_bs,
        max_num_token=max_num_token,
        cache_loc_dtype=torch.int64,
        enable_mamba_track=(
            sa.enable_mamba_extra_buffer()
            and mr.spec_algorithm.is_none()
        ),
        is_encoder_decoder=mr.model_config.is_encoder_decoder,
        encoder_len_fill_value=(
            getattr(mr.model_config.hf_config, 'max_source_positions', 0)
            if is_encoder_decoder else 0
        ),
        dp_size=sa.dp_size,
    )
```

评论区精华

本次 PR 的 Review 全部由自动化代码审查机器人 chatgpt-codex-connector[bot] 完成，未引入人工讨论。以下为审查核心要点：

- P1：拆分预填充的 MLP-sync 顺序（model_runner.py line 3421）：由于 forward_split_prefill 被移到 _prepare_eager_forward_batch 之前，可能导致 DP/MLP-sync 配置下的集体操作顺序错误。

- P1：共享零填充 seq_lens 影响解码 CUDA graph 捕获（cuda_graph_buffer_registry.py line 932）：EagerRunner 的 seq_len_fill_value=0 通过共享池传播到解码 CUDA graph runner，导致捕获时规划错误的 KV 长度。
- P2：多种场景下静态缓冲区尺寸不足（多个文件）：包括 Frozen-KV MTP top-k 扩展、dLLM block_size、MLP-sync padding、动态预填充分块、DFLASH 验证、EAGLE 草稿扩展等。
- P2：预填充 CP 下错误禁用解码 CUDA graph（model_runner.py）：无条件 can_run_graph = False 的设置使所有解码批次都回退到 eager，影响了 CP 部署的性能。
- 拆分预填充的 MLP-sync 顺序风险 (correctness): 审查建议保持 split prefill 仍在 eager preparation 路径上。PR 未对此做修改，风险待确认。
- 共享零填充 seq_lens 污染 CUDA graph 捕获 (correctness): PR 在 decode_cuda_graph_runner.py 的 capture 中添加了 seq_lens.fill_(seq_len_fill_value) 恢复，问题已修复。
- 静态缓冲区尺寸不足的多种场景 (correctness): 评论提出了 6 个 P2 建议，要求逐一增加尺寸估算。PR 未对大多数场景做调整，风险未完全解决。
- 预填充 CP 下错误禁用解码 CUDA graph (performance): 审查指出此逻辑过于激进，应仅限制 prefill 图。PR 未修改，风险待定。

风险与影响

- 风险：结合 review 评论，本 PR 存在以下主要风险：
  1. 缓冲区尺寸不足（多个 P2 评论）：EagerRunner 的固定缓冲在 speculative、dLLM、DP/MLP-sync、动态预填充等场景下可能尺寸不足，导致 load_batch 越界拷贝。需要在初始化时更全面地估算最大 token 数。
  2. 共享缓冲池的隐式污染（P1 评论）：share_pool=True 导致 EagerRunner 的 0 填充值泄漏到解码 CUDA graph runner 的 seq_lens 缓冲区，影响图中注意力元数据的规划。
  3. 拆分预填充的集体操作顺序（P1 评论）：将 forward_split_prefill 提前可能破坏 MLP-sync 和 attention TP scatter/gather 的正确性。
  4. 预填充 CP 下 CUDA graph 被完全禁用（P2 评论）：无条件 can_run_graph = False 会降低解码阶段性能。
  5. 回归风险：无新增测试，仅依赖现有测试覆盖，但重构涉及核心执行路径，边界条件可能遗漏。 - 影响：影响范围：主要影响 SGLang 运行时内核的模型执行层（ModelRunner 及其子 runner）。对用户透明，预期内部行为不变。影响程度中等偏低：重构但无外部 API 变更，但缓冲尺寸估算若保守则可能导致运行时崩溃。对团队开发而言，降低了 ModelRunner 的复杂度，便于后续在 EagerRunner 上独立优化。 - 风险标记：缓冲区尺寸估算不足，共享池零值污染，CP 下 CUDA Graph 误禁，集体操作顺序风险

关联脉络

- PR #28385 refactor(runner): split BaseRunner (shared) from BaseCudaGraphRunner: 本 PR 构建于 #28385 之上，EagerRunner 继承自 BaseRunner，后者由 #28385 从

BaseCudaGraphRunner 拆分而来。