PR #23716 完整报告

sgl-project/sglang

[diffusion] refactor: make timestep scheduler request-local

合并时间：2026-04-26 15:59

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23716

执行摘要

- 一句话：将 timestep scheduler 从 stage 共享改为 request-local 隔离
- 推荐动作：本 PR 是 diffusion 模块的重要架构重构，值得仔细阅读。尤其关注 get_or_create_request_scheduler 的设计权衡（isolate 参数）以及 _reset_scheduler_loop_state 的实现。建议结合评论区反馈验证关键修复是否已落地，并在合并前添加集成测试覆盖主要 pipeline（如 Stable Diffusion 3、Wan2.1、MOVA）。

功能与动机

PipelineStages 设计上不应修改全局状态（not changing global states outside of Req）。共享 stage 级 scheduler 会导致跨请求的状态污染，因此需要将 scheduler 变为请求本地对象。

实现拆解

1. 新增 diffusion_scheduler_utils.py，提供 clone_scheduler_runtime 和 get_or_create_request_scheduler 函数。
2. 修改 denoising.py：DenoisingContext 新增 scheduler 字段，_prepare_denoising_loop 从 batch.scheduler 获取 scheduler，新增 _reset_scheduler_loop_state 重置状态。
3. 修改模型特定 stage（MOVA、Hunyuan3D、音频视频 denoising 等）将 self.scheduler 替换为 batch.scheduler，确保每个请求有独立实例。
4. 在 scheduler_mixin.py 中新增 _init_request_scheduler_from_template 和 _init_disagg_request_scheduler 函数处理分解场景，并将 scheduler 加入传输排除列表。
5. 当前顺序路径通过 isolate=False 避免深拷贝；分解场景通过模板克隆创建隔离实例。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/diffusion_scheduler_utils.py（模块 扩散调度；类别 source；类型 core-logic；符号 clone_scheduler_runtime, get_or_create_request_scheduler）：新文件，定义请求级 scheduler 克隆与获取的通用工具函数，是整个重构的核心基础。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py（模块 扩散调度；类别 source；类型 core-logic；符号 _reset_scheduler_loop_state, _handle_boundary_ratio, _prepare_denoising_loop, _before_denoising_loop）：核心 denoising 循环改造，从 self.scheduler 切换到 batch.scheduler，并新增 scheduler 状态重置方法。

- python/sglang/multimodal_gen/runtime/disaggregation/scheduler_mixin.py（模块 扩散调度；类别 source；类型 core-logic；符号 _init_request_scheduler_from_template, _init_disagg_request_scheduler）：处理分解场景的请求级 scheduler 初始化，并确保 scheduler 不参与序列化传输。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/mova.py（模块 扩散调度；类别 source；类型 data-contract；符号 MOVATimestepPreparationStage.forward, MOVADenoisingStage._select_visual_dit, MOVADenoisingStage.forward）：MOVA 模型的 scheduler 使用改造，包括 timestep 准备和 denoising 阶段都改为使用 batch.scheduler。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/hunyuan3d_shape.py（模块 扩散调度；类别 source；类型 core-logic；符号 _prepare_latents, forward, _prepare_denoising_loop）：Hunyuan3D 模型的 latents 准备和 denoising 循环改造，使用 batch.scheduler 并传递 scheduler 参数。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising_av.py（模块 扩散调度；类别 source；类型 dependency-wiring；符号 forward）：音频视频 denoising 阶段使用 clone_scheduler_runtime 创建蒸馏 scheduler 副本，并临时替换 batch.scheduler。

关键符号：clone_scheduler_runtime, get_or_create_request_scheduler, _reset_scheduler_loop_state, _handle_boundary_ratio, _prepare_denoising_loop, _before_denoising_loop, _init_request_scheduler_from_template, _init_disagg_request_scheduler, MOVATimestepPreparationStage.forward, MOVADenoisingStage._select_visual_dit, MOVADenoisingStage.forward, Hunyuan3DShapeStage._prepare_latents, Hunyuan3DShapeStage.forward, Hunyuan3DShapeStage._prepare_denoising_loop, AudioVideoDenoisingStage.forward

关键源码片段

python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py

核心 denoising 循环改造，从 self.scheduler 切换到 batch.scheduler，并新增 scheduler 状态重置方法。

```
def _reset_scheduler_loop_state(self, scheduler) -> None:
    # Reset diffusers scheduler internal state for a new denoising loop.
    # This ensures that consecutive denoising runs on the same scheduler instance
    # start with fresh step counters and multi-step buffers.
    if hasattr(scheduler, "_step_index"):
        scheduler._step_index = None
    if hasattr(scheduler, "_begin_index"):
        scheduler._begin_index = None
    if hasattr(scheduler, "lower_order_nums"):
        scheduler.lower_order_nums = 0
    if hasattr(scheduler, "last_sample"):
        scheduler.last_sample = None
    if hasattr(scheduler, "this_order"):
        scheduler.this_order = 0
```

```
    # For multi-step solvers (e.g. DPM-Solver), clear buffer arrays.
    solver_order = getattr(
        getattr(scheduler, "config", None), "solver_order", 0
    )
    if solver_order > 1:
        if hasattr(scheduler, "model_outputs"):
            scheduler.model_outputs = []
        if hasattr(scheduler, "timestep_list"):
            scheduler.timestep_list = []
```

python/sglang/multimodal_gen/runtime/disaggregation/scheduler_mixin.py

处理分解场景的请求级 scheduler 初始化，并确保 scheduler 不参与序列化传输。

```
def _init_request_scheduler_from_template(
    scheduler_template: Any, req: Req, device: torch.device
) -> None:
    scheduler = clone_scheduler_runtime(scheduler_template)
    extra_kwargs = {}
    mu = req.extra.get("mu") if hasattr(req, "extra") else None
    if mu is not None:
        extra_kwargs["mu"] = mu

    # Determine how to call set_timesteps based on what the request provides.
    set_timesteps_params = inspect.signature(scheduler.set_timesteps).parameters
    timesteps = getattr(req, "timesteps", None)
    sigmas = getattr(req, "sigmas", None)
    num_steps = getattr(req, "num_inference_steps", None)

    if sigmas is not None and "sigmas" in set_timesteps_params:
        if isinstance(sigmas, torch.Tensor):
            sigmas = sigmas.detach().cpu()
        scheduler.set_timesteps(sigmas=sigmas, device=device, **extra_kwargs)
    elif timesteps is not None and "timesteps" in set_timesteps_params:
        if isinstance(timesteps, torch.Tensor):
            timesteps = timesteps.detach().cpu()
        scheduler.set_timesteps(timesteps=timesteps, device=device, **extra_kwargs)
    elif num_steps is not None:
        scheduler.set_timesteps(num_steps, device=device, **extra_kwargs)
    else:
        # No timestep info available — caller will set it later.
        return

    req.scheduler = scheduler
    req.timesteps = scheduler.timesteps
```

评论区精华

详见 final_report_markdown 的评论区精华章节。

- num_train_timesteps 属性访问兼容性 (correctness): 作者在后续 commit 'Address stateless scheduler review feedback' 中可能已修复此问题，但评论中未明确确认。当前代码中 denoising.py 已改为 getattr(scheduler, "num_train_timesteps", None) 回退到 scheduler.config.num_train_timesteps，但 mova.py 仍直接访问 scheduler.num_train_timesteps。
- solver_order getattr 对 FrozenDict 无效 (correctness): 代码中仍使用 getattr，未变更，可能对 FrozenDict 无效，需要修复。
- denoising_dmd.py 使用未初始化 scheduler 副本 (correctness): 作者 commit 提及 'Avoid cloning DMD scheduler per request'，但评论中未给出最终方案，当前代码中是否已采用 prepared_vars.scheduler 需确认。
- causal_denoising.py 后备 self.scheduler 仍共享 (correctness): 代码中保留了此后备逻辑，未改为强制使用 batch.scheduler，存在风险。

风险与影响

- 风险：详见 final_report_markdown 的风险与影响章节。
- 影响：详见 final_report_markdown 的风险与影响章节。
- 风险标记：核心路径变更，diffusers 兼容性问题，缺少测试覆盖，disaggregation 初始化风险

关联脉络

- 暂无明显关联 PR