# PR #22604 完整报告

sgl-project/sglang

[Diffusion] Standalone Rollout API + Denoising Environment Backpass + SP-Aligned Log-Prob for T2I Post-Training

合并时间：2026-04-15 10:10

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22604

## 执行摘要

- 一句话：为扩散模型后训练新增独立 Rollout API，支持轨迹收集和序列并行对齐的对数概率。
- 推荐动作：建议精读此 PR 以学习其设计模式：混入类（RolloutDenoisingMixin）分离核心逻辑、SP 对齐策略（避免 all_reduce）和按样本粒度 API 设计。关注 _kwargs_to_cpu 的递归问题和文件组织，可能需后续优化。

## 功能与动机

根据 PR body，RL-based 后训练（如 FlowGRPO）需要专用 serving endpoint 来隔离通用 T2I 生成路径，并提供足够的轨迹元数据以在外部重放 rollout 进行策略梯度计算，同时在 Sequence Parallelism 下保持数值稳定性。

## 实现拆解

1. 新增独立 Rollout HTTP API：在 python/sglang/multimodal_gen/runtime/entrypoints/post_training/rollout_api.py 中定义 POST /rollout/generate 端点，使用 RolloutRequest/RolloutResponse 结构，通过 _extract_single_sample_tensor 和 _slice_rollout_trajectory_for_sample 实现按样本粒度输出，便于 RL 训练器处理单个轨迹。
2. 引入 RolloutDenoisingMixin：在 python/sglang/multimodal_gen/runtime/post_training/rollout_denoising_mixin.py 中创建混入类，添加 _maybe_init_denoising_env_collection 和 _maybe_append_dit_trajectory_step 等方法，用于收集冻结的 transformer kwargs 和 DiT 轨迹（如原始噪声潜变量），通过 _kwargs_to_cpu 将张量移至 CPU。
3. 增强序列并行支持：在 python/sglang/multimodal_gen/runtime/post_training/sp_utils.py 中新增 gather_stacked_latents_for_sp 等函数，确保 SP 下张量被正确收集到完整形状，避免数值漂移；SP 对齐对数概率通过全缓冲区噪声计算实现，无需额外集合通信。
4. 添加张量序列化工具：在 python/sglang/multimodal_gen/runtime/entrypoints/post_training/utils.py 中定义 tensor_to_base64 和 _maybe_serialize 函数，用于将轨迹数据序列化为 base64 字符串以便 HTTP 传输。
5. 配套测试与配置：新增 python/sglang/multimodal_gen/test/unit/test_rollout_api.py 和修改 test_scheduler_rollout_unit.py 等测试文件，验证序列化、API 响应和数值正确性；同时添加 pipeline config mixins 如 qwen_image_rollout_pipeline_mixin.py 用于模型特定收集逻辑。

关键文件：

- python/sglang/multimodal_gen/runtime/entrypoints/post_training/rollout_api.py（模块 后训练入口；类别 source；类型 entrypoint；符号 _extract_single_sample_tensor, _slice_rollout_trajectory_for_sample, _serialize_rollout_trajectory, _build_response）：定义了独立的 Rollout HTTP API 端点，是实现用户接口的核心文件。
- python/sglang/multimodal_gen/runtime/post_training/rollout_denoising_mixin.py（模块 去噪混入；类别 source；类型 dependency-wiring；符号 _kwargs_to_cpu, RolloutDenoisingMixin, _maybe_prepare_rollout, _maybe_collect_rollout_log_probs）：包含 RolloutDenoisingMixin 混入类，负责收集 Denoising 环境数据和 DiT 轨迹。
- python/sglang/multimodal_gen/runtime/post_training/sp_utils.py（模块 序列并行工具；类别 source；类型 core-logic；符号 should_do_sp_collective, gather_stacked_latents_for_sp, all_reduce_if_sp_sharded, all_gather_if_sp_sharded）：提供序列并行辅助函数，确保 rollout 数据在 SP 下正确收集，是实现 SP 对齐对数概率的关键。
- python/sglang/multimodal_gen/runtime/entrypoints/post_training/utils.py（模块 序列化工具；类别 source；类型 core-logic；符号 tensor_to_base64, base64_to_tensor, _maybe_serialize, _maybe_deserialize）：实现张量序列化和反序列化工具，用于将轨迹数据编码为 base64 字符串以通过 HTTP 传输。
- python/sglang/multimodal_gen/test/unit/test_rollout_api.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestTensorToBase64Roundtrip, _roundtrip, test_float32_1d, test_float32_nd）：新增单元测试，验证张量序列化、API 响应构建和轨迹切片功能的正确性。

关键符号：_extract_single_sample_tensor, rollout_generate, _kwargs_to_cpu, _maybe_prepare_rollout, _maybe_collect_rollout_log_probs, gather_stacked_latents_for_sp, tensor_to_base64, _maybe_serialize

## 关键源码片段

### python/sglang/multimodal_gen/runtime/entrypoints/post_training/rollout_api.py

定义了独立的 Rollout HTTP API 端点，是实现用户接口的核心文件。

```
def _extract_single_sample_tensor(obj: Any, sample_idx: int, batch_size: int) -> Any:
    """递归提取单个样本的张量，用于将批次数据拆分为 per-sample 粒度。"""
    if isinstance(obj, torch.Tensor):
        # 如果张量的第一维等于批次大小，则提取对应样本
        if obj.dim() >= 1 and obj.shape[0] == batch_size:
            return obj[sample_idx].contiguous()
        return obj
    if isinstance(obj, dict):
        # 递归处理字典中的每个值
        return {
            k: _extract_single_sample_tensor(v, sample_idx, batch_size)
            for k, v in obj.items()
```

```
        }
    if isinstance(obj, list):
        # 递归处理列表
        return [_extract_single_sample_tensor(v, sample_idx, batch_size) for v in obj]
    if isinstance(obj, tuple):
        # 递归处理元组并保持类型
        return tuple(
            _extract_single_sample_tensor(v, sample_idx, batch_size) for v in obj
        )
    return obj # 非张量或容器类型直接返回
```

## python/sglang/multimodal_gen/runtime/post_training/rollout_denoising_mixin.py

包含 RolloutDenoisingMixin 混入类，负责收集 Denoising 环境数据和 DiT 轨迹。

```
def _kwargs_to_cpu(d: Any) -> Any:
    """将嵌套结构中的张量移至 CPU，但当前实现未完全递归处理嵌套序列。"""
    if isinstance(d, torch.Tensor):
        return d.detach().cpu()
    if isinstance(d, dict):
        return {k: _kwargs_to_cpu(v) for k, v in d.items()}
    if isinstance(d, list):
        return [_kwargs_to_cpu(v) for v in d]
    if isinstance(d, tuple):
        return tuple(_kwargs_to_cpu(v) for v in d) # 注意：这里可能未处理嵌套元组中的张量
    return d

class RolloutDenoisingMixin:
    def _maybe_init_denoising_env_collection(
        self,
        batch,
        pipeline_config,
        image_kwargs: dict[str, Any],
        pos_cond_kwargs: dict[str, Any],
        neg_cond_kwargs: dict[str, Any],
        guidance: torch.Tensor | None,
    ) -> None:
        """根据标志初始化 Denoising 环境收集状态。"""
        collect_env = batch.rollout_return_denoising_env
        collect_traj = batch.rollout_return_dit_trajectory
        if not (collect_env or collect_traj):
            batch._rollout_dit_env_state = None
            return
        # 使用 pipeline_config 的清理函数处理 kwargs
        sanitize = getattr(pipeline_config, "sanitize_dit_env_kwargs", lambda x: x)
        if collect_env:
            env = RolloutDenoisingEnv(
                image_kwargs=_kwargs_to_cpu(sanitize(image_kwargs)),
                pos_cond_kwargs=_kwargs_to_cpu(sanitize(pos_cond_kwargs)),
```

```
            neg_cond_kwargs=(
                _kwargs_to_cpu(sanitize(neg_cond_kwargs))
                if neg_cond_kwargs
                else None
            ),
            guidance=guidance.detach().cpu() if guidance is not None else None,
        )
    else:
        env = None
    # 初始化状态以存储轨迹步骤
    batch._rollout_dit_env_state = {
        "env": env,
        "step_latents": [],
        "timesteps": [],
    }
```

## 评论区精华

review 中，gemini-code-assist[bot] 指出 _kwargs_to_cpu 函数不是完全递归的，对于嵌套序列（如列表的列表中的张量）无法正确处理，且可能将元组转换为列表，建议实现一个更健壮的递归版本以保持类型一致性。mickqian 评论了文件路径问题，建议将 mixin 文件移至 pipeline_configs/mixins 目录。这些讨论凸显了代码健壮性和项目结构的一致性问题，但 PR 已合并，可能未完全解决。

- _kwargs_to_cpu 函数的递归问题 (correctness): 建议实现更健壮的递归版本以保持类型一致性，但 PR 中未显示是否已修复。
- 文件路径组织 (style): 评论简短，可能已接受或忽略，但未在 PR 中明确调整。

## 风险与影响

- 风险：技术风险包括：1. 回归风险：新 API 和混入类可能影响现有扩散模型生成路径，特别是在 DenoisingStage 的修改中（python/sglang/multimodal_gen/runtime/pipelines_core/stages/denoising.py），若生命周期钩子调用顺序错误可能导致逻辑错误。2. 性能开销：张量序列化和反序列化（通过 tensor_to_base64）可能增加 CPU 和内存开销，尤其在返回大型轨迹时。3. 数值稳定性：SP 对齐对数概率依赖于全缓冲区噪声计算，若噪声生成或收集逻辑有误，可能导致跨 rank 不一致。4. 兼容性：新 API 仅支持 T2I 任务，未来扩展至其他任务（如 T2V）时需额外适配。
- 影响：对用户（RL 训练者）而言，新增了专用端点，简化了轨迹数据获取，提升训练效率；对系统，增加了新的 HTTP 路由和数据处理模块，需要维护和测试；对团队，引入了混入类和 SP 工具，需理解设计以进行后续开发。影响范围集中在扩散模型后训练模块，不影响核心推理路径。
- 风险标记：新 API 接口，序列化开销，SP 逻辑复杂，生命周期钩子风险

## 关联脉络

- PR #22763 [diffusion] chore: auto-enable best parallel setting if unspecified: 同属扩散模型模块，涉及性能优化和并行设置，可能共享类似的设计模式。
- PR #22667 [diffusion] model: support Ltx 2.3 two stage ti2v: 均为扩散模型功能扩展，展示仓库在扩散领域的持续演进。