PR #34197 完整报告

sgl-project/sglang

[diffusion] RL rollout support for the Cosmos3 pipeline

合并时间：2026-08-18 20:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34197

执行摘要

- 一句话：Cosmos3 管线新增 RL rollout 支持及 fused 权重路由
- 推荐动作：值得精读，尤其是两个设计决策：一是 rollout 调度网格的继承策略（显式 shift 重建 vs 继承 serving grid），二是 fused 参数通过 weight_loader 的 shard id 路由，这为扩散模型 LoRA/ 后训练权重同步提供了干净的数据契约。阅读时可结合其依赖的 #34933（fused LoRA adapters）与 #34491（LoRA IPC fix）一起看，能更完整理解 Cosmos3 后训练链路的演进。

功能与动机

PR body 指出 “Several changes to support rollout of Cosmos3”，即为了让 Cosmos3 支持 RL rollout，需要两条关键能力：一是每个 rollout 请求拥有独立且可复现的调度器时间网格（rollout 的 SDE 计算与 serving 的调度状态不能互相污染）；二是后训练权重更新链路要能正确处理 diffusers 风格 fused 线性层参数（例如 q/k/v 合并为 to_qkv），否则 LoRA/ 权重同步无法落到对应分片。该 PR 栈基于 #34933，依赖其 fused linear layers 上的 LoRA 组合支持。

实现拆解

实现拆解（共 4 步，按数据流顺序）：

1. 在 timestep 阶段绑定 per-request rollout 调度器 - 文件：python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/cosmos3.py。 - 在 Cosmos3TimestepStage.forward 中，将原先 flow_shift 变量重命名为 explicit_flow_shift，保留来自 batch.flow_shift 或 pipeline_config.flow_shift 的显式值；当 batch.rollout 为 True 时调用新增的 prepare_rollout_request_scheduler，把 rollout 专用 scheduler 绑定到 batch.scheduler，并同步更新 batch.timesteps。 - 同一 PR 在 Cosmos3ImagePreprocessStage.forward 中把 seed 生成的 generator 写回 batch.generator，保证 rollout 的 SDE 噪声步与 denoising 主循环共享同一随机源，避免 SP 多卡轨迹发散。 - 配套逻辑落在 python/sglang/multimodal_gen/runtime/post_training/rollout_scheduler.py：新增 prepare_rollout_request_scheduler，核心决策是“显式 shift 用公式重建网格、无显式 shift 则完整继承 serving 网格”，并将 set_shift(1.0) 使 set_timesteps 原样保留外部 sigma。
2. 在 denoising 主循环接入 RolloutDenoisingMixin - Cosmos3DenoisingStage 的基类从 PipelineStage 扩展为 PipelineStage, RolloutDenoisingMixin。 - forward 中优先使用

batch.scheduler（若存在）作为当前步 scheduler；rollout 时先做模态校验：velocity_mask/condition_latents 非空（I2V/V2V 条件帧路径）或存在 action/sound latents 时直接抛 ValueError，因为条件帧 re-blending 会破坏 SDE log-prob 计算依赖的高斯过渡假设，action/sound 模态当前未实现。 - 校验通过后调用 _maybe_prepare_rollout 与 _maybe_init_denoising_env_collection 初始化轨迹收集环境；循环内对 rollout 请求设置 batch._rollout_loop_step_index，调用 _maybe_append_dit_trajectory_step 记录每步 x_{t_i}，并让 scheduler.step 额外接收 batch 与 batch.generator。 - 循环结束后调用 _postprocess_rollout_outputs，并把 rollout_trajectory_data 透传到 pipeline 输出对象上。

3. 权重更新器支持 fused 参数 shard id - 文件：python/sglang/multimodal_gen/runtime/post_training/weights_updater.py。 - _iter_module_weight_updates 的 yield 从二元组 (name, weight) 扩展为三元组 (mapped_name, weight, shard_id)；shard_id 由 _build_module_weight_name_mapper 返回的 merge index 充当，普通参数为 None。 - load_weights_into_model 解包 entry，遇到非 None 的 shard_id 时调用 weight_loader(param, weight, shard_id)，否则保持原有两参数调用，向后兼容。 - _load_weights_into_module 在 layerwise offload 路径显式检测：一旦出现带 shard_id 的 fused 参数更新就抛 NotImplementedError，避免把分片权重误当整权重写入 CPU 缓冲。
4. 测试配套 - 新增 python/sglang/multimodal_gen/test/unit/test_cosmos3_rollout.py（169 行），覆盖调度器网格三类行为：无显式 shift 时继承 serving grid 且末尾 sigma 补 0；显式 shift 时生成 plain shifted grid；RL 可复用的 Euler serving scheduler 直通共享。 - 对权重更新器增加 _FusedParamModule 模拟 diffusers 风格 q/k/v 映射，验证 merge index 以 shard id 到达 weight_loader，并验证直接命中 to_qkv 时仍保持两参数调用。

关键文件：

- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/cosmos3.py（模块 扩散管线；类别 source；类型 data-contract；符号 Cosmos3DenoisingStage, Cosmos3TimestepStage）：Cosmos3 rollout 的主接入点：timestep 阶段绑定 per-request 调度器，denoising 阶段接入 RolloutDenoisingMixin 并新增模态校验与轨迹收集分支。
- python/sglang/multimodal_gen/runtime/post_training/rollout_scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 prepare_rollout_request_scheduler）：新增 prepare_rollout_request_scheduler，集中实现 rollout 调度网格的两种生成策略（继承 serving grid 或显式 shift 重建），是 rollout 正确性的核心逻辑。
- python/sglang/multimodal_gen/runtime/post_training/weights_updater.py（模块 权重更新；类别 source；类型 core-logic；符号 _iter_module_weight_updates, load_weights_into_model, _load_weights_into_module）：权重更新链路新增 fused 参数 shard id 路由，使 q/k/v -> to_qkv 这类 diffusers 风格映射能正确调用 weight_loader，并对 layerwise offload 显式 fail-closed。
- python/sglang/multimodal_gen/test/unit/test_cosmos3_rollout.py（模块 单元测试；类别 test；类型 test-coverage；符号 _serving_scheduler, _rollout_batch, _prepare, TestPrepareRolloutRequestScheduler）：新增单元测试，覆盖 rollout 调度器网格继承 / 显式 shift/ 直通三类行为，以及 fused 参数 shard id 到达 weight_loader 的两种调用形态

，是对本 PR 核心逻辑的直接验证。

关键符号：prepare_rollout_request_scheduler, Cosmos3DenoisingStage.forward, Cosmos3TimestepStage.forward, _iter_module_weight_updates, load_weights_into_model, _maybe_prepare_rollout, _maybe_init_denoising_env_collection, _maybe_append_dit_trajectory_step, _postprocess_rollout_outputs

关键源码片段

python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/cosmos3.py

Cosmos3 rollout 的主接入点：timestep 阶段绑定 per-request 调度器，denoising 阶段接入 RolloutDenoisingMixin 并新增模态校验与轨迹收集分支。

```
# Cosmos3DenoisingStage.forward 中 rollout 专属分支（整理自源码，省略常规循环体）。
class Cosmos3DenoisingStage(PipelineStage, RolloutDenoisingMixin):
    # ... 构造与并行配置略 ...

    def forward(self, batch: Req, server_args: ServerArgs) -> Req:
        # ... 前置上下文读取略 ...

        # Rollout 请求的 scheduler 已由 timestep 阶段绑定到 batch.scheduler。
        scheduler = batch.scheduler if batch.scheduler is not None else self.scheduler

        if batch.rollout:
            # 只允许 T2V/T2I：I2V/V2V 的 conditioned-frame re-blending
            # 会破坏 SDE log-prob 计算依赖的高斯过渡假设。
            if velocity_mask is not None or condition_latents is not None:
                raise ValueError(
                    "Cosmos3 rollout supports T2V/T2I only; I2V/V2V "
                    "conditioned-frame re-blending breaks the Gaussian "
                    "transition assumption of the SDE log-prob math."
                )
            # action/sound 模态暂不支持 rollout。
            if action_latents is not None or sound_latents is not None:
                raise ValueError(
                    "Cosmos3 rollout does not support action/sound modalities."
                )
            # 初始化轨迹收集环境，后续每步通过 mixin 记录 x_{t_i}。
            self._maybe_prepare_rollout(batch)
            self._maybe_init_denoising_env_collection(
                batch=batch,
                pipeline_config=server_args.pipeline_config,
                image_kwargs={},
                pos_cond_kwargs={
                    "text_ids": cond_text_ids,
                    "text_mask": cond_text_mask,
```

```
            "fps": fps,
        },
        neg_cond_kwargs={
            "text_ids": uncond_text_ids,
            "text_mask": uncond_text_mask,
            "fps": fps,
        },
        guidance=None,
    )

# ... CFG/Ulysses 并行配置计算略 ...
for i, t in enumerate(timesteps):
    # ... 模型前向与 CFG 合并略 ...
    if batch.rollout:
        # 在 scheduler 推进前记录当前步的 x_{t_i}，供 RL 轨迹回放。
        batch._rollout_loop_step_index = i
        self._maybe_append_dit_trajectory_step(
            batch=batch, latents=latents,
            timestep_value=t, step_index=i,
        )
        # rollout 变体接收 batch 与 batch.generator，
        # 保证 SDE 噪声、轨迹与调度器状态保持一致。
        latents = scheduler.step(
            noise_pred, t, latents,
            generator=batch.generator,
            batch=batch,
            return_dict=False,
        )[0]
    else:
        # 常规 serving 路径保持原调用方式。
        latents = scheduler.step(
            noise_pred, t, latents,
            generator=generator,
            return_dict=False,
        )[0]

if batch.rollout:
    # 收尾：把完整轨迹写回 batch，供下游解码、日志或训练采样使用。
    self._postprocess_rollout_outputs(
        batch=batch,
        latents=latents,
        num_inference_steps=len(timesteps),
        final_timestep=timesteps.new_zeros(()).cpu(),
        server_args=server_args,
    )

batch.latents = latents
# ... 后续模态输出与指标字段略 ...
```

python/sglang/multimodal_gen/runtime/post_training/rollout_scheduler.py

新增 prepare_rollout_request_scheduler，集中实现 rollout 调度网格的两种生成策略（继承 serving grid 或显式 shift 重建），是 rollout 正确性的核心逻辑。

```
# 为 rollout 请求绑定 per-request scheduler，并生成对应的 sigma 时间网格。
# 核心决策：显式 shift 时用公式重建网格；无显式 shift 时完整继承 serving 网格。
def prepare_rollout_request_scheduler(
    batch: Req,
    serving_scheduler: Any,
    *,
    explicit_shift: float | None,
    num_inference_steps: int,
    device: torch.device,
) -> None:
    # 先复用或创建 rollout scheduler；RL 可用的 Euler 调度器会原样直通。
    scheduler = get_or_create_rollout_request_scheduler(batch, serving_scheduler)

    # 只有从 UniPC 映射到 Euler 的 scheduler 才需要重建网格；
    # 直接复用 serving scheduler 时保留其原始 grid。
    if scheduler is not serving_scheduler:
        if explicit_shift is not None:
            # 显式 shift：先构造未扭曲的线性网格，再按 flow shift 公式做 time warping。
            shift = float(explicit_shift)
            num_train_timesteps = scheduler.config.num_train_timesteps
            sigmas = torch.linspace(1.0, 1.0 / num_train_timesteps, num_inference_steps)
            sigmas = shift * sigmas / (1 + (shift - 1) * sigmas)
        else:
            # 无显式 shift：继承 serving 网格（去掉末尾的 sigma=0）。
            sigmas = serving_scheduler.sigmas[:-1]

        # 设 shift=1.0 后，set_timesteps 会把上面传入的 sigmas 原样保留。
        scheduler.set_shift(1.0)
        scheduler.set_timesteps(sigmas=sigmas.tolist(), device=device)

    # 每个 rollout 请求拥有独立的 timesteps，避免与其他请求共享调度器状态。
    batch.timesteps = scheduler.timesteps
```

python/sglang/multimodal_gen/runtime/post_training/weights_updater.py

权重更新链路新增 fused 参数 shard id 路由，使 q/k/v -> to_qkv 这类 diffusers 风格映射能正确调用 weight_loader，并对 layerwise offload 显式 fail-closed。

```
# 权重更新迭代器：把二元组扩展为 (mapped_name, weight, shard_id)。
# shard_id 表示该权重在 fused 参数（如 q/k/v -> to_qkv）里的 merge index，
# 普通参数为 None。
def _iter_module_weight_updates(module, weights_iter, model_params):
    map_name = _build_module_weight_name_mapper(module)
    module_name = type(module).__name__

    for name, loaded_weight in weights_iter:
```

```
        if name in model_params:
            # 直接命中的参数没有 shard 概念。
            yield name, loaded_weight, None
            continue

        # 通过 param_names_mapping / lora_param_names_mapping 映射，
        # 同时取出 fused 时的 merge index 作为 shard_id。
        mapped_name, merge_index = (
            map_name(name) if map_name is not None else (name, None)
        )
        if mapped_name in model_params:
            yield mapped_name, loaded_weight, merge_index
            continue

        logger.warning(
            "Skipping weight update for %s: parameter %r not found after mapping to %r",
            module_name, name, mapped_name,
        )


def load_weights_into_model(weights_iter, model_params, module_name=None):
    """按 entry 写入权重；entry 可以是 (name, weight) 或 (name, weight, shard_id)。"""
    for entry in weights_iter:
        name, loaded_weight, *rest = entry
        shard_id = rest[0] if rest else None
        if name not in model_params:
            logger.warning("Skipping weight update: parameter %r not found", name)
            continue

        param = model_params[name]
        weight_loader = getattr(param, "weight_loader", None)
        if callable(weight_loader):
            if shard_id is not None:
                # 有 shard_id 的 fused 参数走三参数 weight_loader。
                weight_loader(param, loaded_weight.to(param.dtype), shard_id)
            else:
                # 普通参数保持两参数调用，兼容历史行为。
                weight_loader(param, loaded_weight.to(param.dtype))
        else:
            # 非 fused 参数继续走 DTensor 分发或普通 in-place copy。
            # ... 原有批量分发逻辑略 ...
            pass
```

评论区精华

PR 没有留下 review inline 评论，合并者 mickqian 直接 APPROVED；有价值的信息集中在 PR issue 评论区：

- niehen6174 评估影响面："The changes don't seem to affect the main SGLang-D workflow, and the impact on the post-training path is fairly limited as well." 这说明该改动被确认是隔离在 diffusion 后训练路径内的。
- niehen6174 补充："Cherry-picked the LoRA IPC weight update fix from #34491 into this PR." 即本 PR 额外合入了 LoRA IPC 权重更新修复，用于保障 fused 参数在 IPC 更新链路中正确落盘。
- mickqian 通过 /tag-and-rerun-ci 触发 CI 重跑，最终 PR 测试通过并合并。
- PR 影响面评估 (design): 影响面被确认为隔离在 diffusion 后训练路径内，未引出额外修改要求。
- LoRA IPC 权重更新修复合入 (other): 修复随本 PR 一并合入，保障 fused 参数在 IPC 更新链路中正确落盘。
- CI 重跑 (testing): CI 重新触发后通过，PR 被 mickqian APPROVED 并合并。

风险与影响

- 风险：
  1. 核心 denoising 路径新增分支：Cosmos3DenoisingStage.forward 是 Cosmos3 推理与训练共用的主循环，虽然 rollout 分支只在 batch.rollout 为真时生效，但 scheduler 变量已从 self.scheduler 改为优先取 batch.scheduler；若未来其他路径错误设置 batch.scheduler，可能绕过默认调度器。建议保持"仅 rollout 时绑定 batch.scheduler"的约定。
  2. fused 权重接口形态变化：_iter_module_weight_updates 从二元组变成三元组，所有调用方（_load_weights_into_module、_resolve_lora_ipc_layer_dict_key 等）都需适配；本 PR 只覆盖了当前已知调用点，若外部自定义权重更新路径直接消费迭代器，可能出现解包错误。
  3. layerwise offload 与 fused 参数不兼容：offload 路径遇到带 shard_id 的 fused 参数会直接抛 NotImplementedError，这是 fail-closed 的保守选择，但使用层卸载 + fused 注意力的场景会硬失败，需要明确文档化。
  4. distilled checkpoint 与 rollout 的交互未明确定义：Cosmos3TimestepStage.forward 中 distilled sigmas 分支提前 return，rollout 绑定逻辑不会执行；若用户对 distilled 模型开启 rollout，可能得到未定义行为。
  5. 模态限制：rollout 明确只支持 T2V/T2I，I2V/V2V 与 action/sound 会抛 ValueError，属于设计上的硬约束，避免错误计算结果。 - 影响：对系统与用户：默认 SGLang-D serving 工作流零影响（niehen6174 已确认）；diffusion 后训练 /RL 路径获得 Cosmos3 rollout 能力，支持 per-request 独立调度网格与 fused 参数权重更新。对团队：该 PR 为后续其他 diffusion 管线接入 rollout 提供了可复用的模式——RolloutDenoisingMixin + prepare_rollout_request_scheduler + fused 参数 shard 路由。对接口契约：weights_updater 的 entry 元组结构变化属于内部 API 变更，存在较小的第三方集成风险。总体影响范围集中在 multimodal_gen 后训练子系统的 Cosmos3 路径，影响程度中等偏小。 - 风险标记：denoising 主循环新增分支，fused 权重接口形态变化，offload 与 fused 参数不兼容，rollout 仅支持 T2V/T2I，distilled checkpoint 与 rollout 交互未定义

关联脉络

- PR #34933 [diffusion] Per-section LoRA adapters on fused linear layers: 本 PR 栈基于 #34933，body 明确说明分支包含其 commit，等待其合并后 rebase；Cosmos3 rollout 的 fused 参数权重更新依赖该 PR 提供的 fused linear layers LoRA 组合支持。
- PR #34491 LoRA IPC weight update fix: niehen6174 在 PR 评论中说明已从 #34491 cherry-pick LoRA IPC 权重更新修复，保证 fused 参数在 IPC 更新链路中正确落盘。