PR #36301 完整报告

sgl-project/sglang

[diffusion] feat: support batching for cosmos3 action generation

合并时间：2026-08-26 22:15

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36301

执行摘要

- 一句话：Cosmos3 动作生成支持请求内批处理，吞吐最高 3.3 倍
- 推荐动作：值得精读，尤其是四点：请求内批处理如何在不破坏 B=1 输出契约的前提下贯穿整条 diffusion 管线；prompt 标量广播与逐图配对的准入校验；新增 adapter 对 protocol.py 的隔离重构；作者对测量边界与数值一致性的坦诚声明。对机器人策略服务或 diffusion 管线开发者有直接借鉴价值。注意合并时 GPU fixed-noise 一致性与隔离 server 基准仍未完成，应作为后续验证项跟进。

功能与动机

Cosmos3 action generation 原先每次请求只接受一个 observation；LIBERO 等机器人评估负载并行运行大量环境，在小 action-policy shape 下串行单观测 HTTP 调用严重闲置模型吞吐（PR body："serial one-observation HTTP calls underutilize the model at small action-policy shapes"）。同时作者在 Accuracy Tests 中承认此前 " 准确性测试不必要 " 的说法错误，因为 batching 改变 RNG 消耗顺序，batched kernel 可能与独立 B=1 执行存在数值差异，需要 GPU fixed-noise 一致性验证。

实现拆解

1. 请求准入与图像、prompt 归一化：新增 python/sglang/multimodal_gen/runtime/entrypoints/action/cosmos3.py，_images_from_observation 从 image / image_path / input_reference / images 提取图像，支持 list、tuple 与 [B, H, W, C] 的 uint8 ndarray 展开为 B 张图，并拒绝 dtype 或 shape 非法输入；_action_prompt 实现标量 prompt 广播与逐图配对，基数不一致立即抛错。protocol.py 的 _normalize_observation 同步支持对图像列表逐项做 base64 / tensor payload 归一化。
2. 采样参数与能力声明：build_cosmos3_action_sampling_params 从 server_args.batching_max_size 读取批大小上限并注入采样参数；cosmos3_action_metadata 在 capabilities 中声明 batch_inputs、max_batch_size 与 batched_action_modes（仅 policy）。protocol.action_metadata 对 Cosmos3 分支直接委托新适配器，删除约 200 行内联实现。
3. 管线 stage 批维度传播：model_specific_stages/cosmos3.py 的 Cosmos3ImagePreprocessStage.forward 通过 DataType.ACTION + action_mode == policy 识别批处理，逐张 load_image、等比缩放裁剪后 torch.stack 成 [B, 3, H, W]；_tokenize_prompt 接受 str | list[str] 并输出 [B, S]，且拒绝不同 tokenized 长度的

prompt（GEN cross-attention 不 mask padded text K/V）。cosmos3_action.py 的 build_action_prompt 改为对描述列表逐条渲染结构化 JSON caption。

4. 入口契约放宽：runtime/entrypoints/utils.py 的 prepare_request 仅对 DataType.ACTION 放行非空字符串列表 prompt，其他管线仍要求 str，避免影响常规视觉生成。
5. 测试与文档：test/unit/test_cosmos3.py 新增配对保持、标量广播、基数不匹配、服务端上限、B=1 兼容、inverse_dynamics 拒绝等用例；cookbook 补充批处理请求 / 响应契约与 --batching-max-size 启动示例。

关键文件：

- python/sglang/multimodal_gen/runtime/entrypoints/action/cosmos3.py（模块 动作适配；类别 source；类型 core-logic；符号 cosmos3_action_metadata, _images_from_observation, _action_prompt, build_cosmos3_action_sampling_params）：新增的 Cosmos3 动作适配器，集中承载批处理核心逻辑：图像展开、prompt 广播 / 配对、采样参数构建与能力声明，是本 PR 功能入口与契约定义所在。
- python/sglang/multimodal_gen/runtime/entrypoints/action/protocol.py（模块 请求协议；类别 source；类型 dependency-wiring；符号 _cosmos3_image_from_observation, _build_cosmos3_action_sampling_params, action_metadata, _normalize_observation）：通用 action 协议层：将 Cosmos3 专用逻辑迁往新适配器，并让 _normalize_observation 支持图像列表逐项归一化，是批处理请求进入管线前的第一道关口。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/cosmos3.py（模块 管线阶段；类别 source；类型 data-contract；符号 Cosmos3ImagePreprocessStage, _tokenize_prompt）：管线核心 stage：预处理阶段在 policy 批处理下输出 [B, 3, H, W]，分词阶段接受字符串列表，是批维度在采样与去噪之间传播的关键位置。
- python/sglang/multimodal_gen/test/unit/test_cosmos3.py（模块 单元测试；类别 test；类型 test-coverage；符号 _cosmos3_server_args, _policy_payload, test_batched_policy_request_preserves_input_pairing, test_batched_policy_request_broadcasts_scalar_prompt）：新增 6+ 个批处理语义单元测试，覆盖输入配对、标量广播、基数校验、服务端上限、B=1 兼容与 inverse_dynamics 拒绝，是行为契约的守护。
- python/sglang/multimodal_gen/runtime/pipelines_core/stages/model_specific_stages/cosmos3_action.py（模块 动作管线；类别 source；类型 data-contract；符号 build_action_prompt）：build_action_prompt 支持描述字符串列表并逐条渲染结构化 action caption，是批量 tokenization 的上游。
- python/sglang/multimodal_gen/runtime/entrypoints/utils.py（模块 请求入口；类别 source；类型 core-logic；符号 prepare_request）：prepare_request 放行 ACTION 请求的非空字符串列表 prompt，是批处理进入执行管线的前提条件。
- docs/cookbook/diffusion/Cosmos/Cosmos3.mdx（模块 文档；类别 docs；类型 documentation）：补充批处理请求 / 响应契约、--batching-max-size 示例与测量边界说明，是用户上手的关键文档。

关键符号：cosmos3_action_metadata, _images_from_observation, _action_prompt, build_cosmos3_action_sampling_params, _normalize_observation, action_metadata, Cosmos3ImagePreprocessStage.forward, _tokenize_prompt, build_action_prompt, prepare_request

关键源码片段

python/sglang/multimodal_gen/runtime/entrypoints/action/cosmos3.py

新增的 Cosmos3 动作适配器，集中承载批处理核心逻辑：图像展开、prompt 广播 / 配对、采样参数构建与能力声明，是本 PR 功能入口与契约定义所在。

```
# Cosmos3 动作适配：把请求中的单张或多张观测图统一展开为图像列表，
# 并处理 prompt 的标量广播与逐图配对，为批处理管线提供统一的输入形态

def _images_from_observation(observation: dict[str, Any]) -> list[Any]:
    # 优先读取单图字段，兼容 image / image_path / input_reference 三种命名
    image = None
    for name in ("image", "image_path", "input_reference"):
        if name in observation:
            image = observation[name]
            break

    # 未命中单图字段时，退化为 images 字典；只允许恰好一个键
    if image is None:
        images = observation.get("images")
        if images is None or (isinstance(images, dict) and not images):
            return []
        if not isinstance(images, dict) or len(images) != 1:
            raise ValueError(
                "Cosmos3 action input accepts one image field; use a list or "
                "a [B, H, W, C] array in that field for batched observations"
            )
        image = next(iter(images.values()))

    # list / tuple 以及四维 ndarray 都视为批量观测，逐张展开
    if isinstance(image, (list, tuple)):
        images = list(image)
    elif isinstance(image, np.ndarray) and image.ndim == 4:
        images = list(image)
    else:
        images = [image]

    normalized_images: list[Any] = []
    for item in images:
        if not isinstance(item, np.ndarray):
            normalized_images.append(item)
            continue
        # 数组形式必须是 uint8，shape 只允许 [H, W] 或 [H, W, C]
```

```
            if item.dtype != np.uint8:
                raise ValueError("Cosmos3 observation image arrays must use uint8 dtype")
            if item.ndim not in (2, 3):
                raise ValueError(
                    "Cosmos3 observation image arrays must have shape [H, W] "
                    f"or [H, W, C], got {tuple(item.shape)}"
                )
            normalized_images.append(Image.fromarray(item))
        return normalized_images


    def _action_prompt(prompt: Any, batch_size: int) -> str | list[str]:
        # 标量 prompt：单图直接返回，多图则广播到每个观测
        if isinstance(prompt, str):
            return prompt if batch_size == 1 else [prompt] * batch_size
        if not isinstance(prompt, (list, tuple)) or not prompt:
            raise ValueError("Cosmos3 action prompt must be a string or non-empty list")
        if not all(isinstance(item, str) for item in prompt):
            raise ValueError("Cosmos3 action prompt list must contain only strings")
        prompts = list(prompt)
        # 单元素列表在多图时等价于标量广播
        if len(prompts) == 1 and batch_size > 1:
            prompts *= batch_size
        if len(prompts) != batch_size:
            raise ValueError(
                "Cosmos3 batched action input requires one prompt per image, got "
                f"{len(prompts)} prompt(s) and {batch_size} image(s)"
            )
        return prompts[0] if batch_size == 1 else prompts
```

python/sglang/multimodal_gen/runtime/entrypoints/action/protocol.py

通用 action 协议层：将 Cosmos3 专用逻辑迁往新适配器，并让 _normalize_observation 支持图像列表逐项归一化，是批处理请求进入管线前的第一道关口。

```
    def _normalize_observation(observation: dict[str, Any]) -> dict[str, Any]:
        normalized = dict(observation)
        images = normalized.get("images")
        if isinstance(images, dict):
            normalized["images"] = {
                name: _normalize_image_value(value) for name, value in images.items()
            }
        # 批处理下 image / image_path / input_reference 可能是图像列表，
        # 需要逐项归一化（base64 / tensor payload），不能只处理单个值
        for name in ("image", "image_path", "input_reference"):
            if name in normalized:
                value = normalized[name]
                normalized[name] = (
                    [_normalize_image_value(item) for item in value]
                    if isinstance(value, (list, tuple))
```

```
                else _normalize_image_value(value)
            )
    # state / noise 等 tensor payload 保持原有解码逻辑
    for key in ("state", "observation.state", "noise", "observation.noise"):
        value = normalized.get(key)
        if isinstance(value, dict):
            normalized[key] = _decode_tensor_payload(value)
    return normalized
```

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

管线核心 stage：预处理阶段在 policy 批处理下输出 [B, 3, H, W]，分词阶段接受字符串列表，是批维度在采样与去噪之间传播的关键位置。

```
class Cosmos3ImagePreprocessStage(PipelineStage):
    """加载、等比缩放并中心裁剪 conditioning 输入。

    普通 I2V 写入 [1, 3, H, W] 的 batch.preprocessed_image；
    批量 policy 请求写入 [B, 3, H, W]；V2V 写入 [1, 3, T_in, H, W]。
    """

    def forward(self, batch: Req, server_args: ServerArgs) -> Req:
        image_path = batch.image_path
        video_path = batch.video_path
        # 只有 action policy 模式才允许保留图像列表（即批处理），
        # 普通 I2V 对 list 仍取第一张，确保既有行为不被破坏
        is_action_policy = (
            batch.data_type == DataType.ACTION
            and getattr(batch.sampling_params, "action_mode", None)
            == ACTION_MODE_POLICY
        )
        if isinstance(image_path, list) and not is_action_policy:
            image_path = image_path[0] if image_path else None
        if isinstance(video_path, list):
            video_path = video_path[0] if video_path else None

        if image_path and video_path:
            raise ValueError(
                "Cosmos3 accepts either --image-path (I2V) or --video-path "
                "(V2V), not both"
            )

        target_h, target_w = batch.height, batch.width

        if image_path is not None:
            # 逐张加载并缩放裁剪，最后堆叠成 [B, 3, H, W]；
            # 单张时 B == 1，与原有 I2V 契约自然兼容
            image_sources = (
                list(image_path)
```

```
            if isinstance(image_path, (list, tuple))
            else [image_path]
        )
        if not image_sources:
            raise ValueError("Cosmos3 I2V image list is empty")
        tensors: list[torch.Tensor] = []
        for src in image_sources:
            image = load_image(src)
            image = _resize_crop_pil(image, target_w, target_h)
            tensors.append(_pil_to_normalized_tensor(image))
        batch.preprocessed_image = torch.stack(tensors, dim=0).contiguous()
        self.log_info(
            f"Preprocessed {len(tensors)} conditioning image(s) to "
            f"{target_w}x{target_h}"
        )
        return batch

    # V2V 与 T2V / T2I 分支保持不变
    ...
```

评论区精华

PR 没有留下 inline review 评论，核心讨论体现在 PR body 的自评与未完成清单：作者明确承认此前 " 准确性测试不必要 " 的判断错误，指出 batching 改变 RNG 消耗顺序、batched kernel 与独立 B=1 存在数值差异风险，GPU fixed-noise 一致性结果必须补齐；同时主动划定吞吐数据边界，强调 1.66x-3.33x 是 RLinf 端到端结果（pipeline_stage_num 随 batch 变化）而非隔离 SGLang 内核基准。

- GPU fixed-noise 一致性验证缺失 (testing): 未解决。checklist 中 GPU fixed-noise B=1-versus-batch 一致性结果仍未勾选，合并前需要补齐；RLinf 闭环结果不能替代隔离数值 parity。
- 吞吐数据测量边界 (performance): 已记录为后续任务，合并前未完成隔离 server 基准；报告数字应视为端到端参考。

风险与影响

- 风险：
  1. GPU 数值一致性未验证：这是合并前最关键的缺口。批处理改变 RNG 消耗顺序，batched kernel 可能与独立 B=1 数值不同，而 RLinf 闭环成功率因评估拓扑随 batch 变化，不能替代隔离数值 parity（见 checklist 未完成项）。
  2. 性能数字口径：当前吞吐数字来自 RLinf 端到端评估，缺少 raw latency、GPU SKU、warmup/repetition 策略、峰值 VRAM 与 per-stage 计时，无法进行 apples-to-apples 的 SGLang server 基准归因。
  3. 契约与兼容性：默认响应仍按 item 返回 [H, D]，需要紧凑 [B, H, D] 的 msgpack 客户端必须显式设置 runtime.response_format="raw"；不同 tokenized prompt 长度会被拒绝，用户需自行对齐 prompt。

4. 资源压力：批处理提升吞吐的同时提高单请求显存占用，--batching-max-size 是唯一保护阀，配置不当可能 OOM。
5. 数据契约变更面：预处理输出从 [1, 3, H, W] 变为可 [B, 3, H, W]，涉及 latent 准备、denoising、CFG 2B timesteps 与 action decoding 多处 shape 假设，若有遗漏 stage 未跟进会产生静默错误。 - 影响：对用户：LIBERO 等多环境机器人评估可通过单次 HTTP 请求批处理多个观测，显著提升吞吐（报告最高 3.3 倍），但需理解 prompt 配对规则、--batching-max-size 与 raw 响应格式。对系统：multimodal_gen action endpoint 的请求 / 响应契约扩展了 batch 维度，pipeline stages 的数据契约从单图变为 [B, ...]；普通 I2V/V2V 与 inverse_dynamics/forward_dynamics 有分支保护，影响受控。对团队：adapter 隔离模式（新增 cosmos3.py）为其他 policy family 复用批处理能力提供了模板，同时把 " 验证边界 " 显式写进 PR checklist，提升了结果可复现性意识。 - 风险标记：GPU 数值一致性未验证，缺少隔离性能基准，数据契约变更，prompt 长度限制，显存压力上升

关联脉络

- PR #36398 fix(diffusion): MiniMax-H3 dp_size>1 deadlock and cross-request audio determinism: 同为 diffusion 管线在多请求 / 批处理场景下的正确性与确定性修复，与本 PR 关注的批处理 RNG 消耗与数值一致性主题呼应。
- PR #36322 [diffusion] perf: fuse tanh-GELU into the LongCat-Image DiT FFN up-proj: 同属 diffusion 模型管线性能优化，说明该方向正在系统性做吞吐改进，可供本 PR 后续隔离基准参照。
- PR #36463 [diffusion] make benchmark caches seedable and cover missing native families: diffusion 基准基础设施改进，与本 PR 提出的 " 补充隔离 server 基准 " 需求互补。