PR #2323 完整报告

THUDM/slime

[cleanup] Refactor rollout.py

合并时间：2026-08-25 11:06

原文链接：http://prhub.com.cn/THUDM/slime/pull/2323

执行摘要

- 一句话：重构 rollout.py，拆分 SGLang 部署逻辑至 sglang_utils 子模块
- 推荐动作：值得精读。本 PR 是典型的「大文件拆分」重构示范，展示了如何将散落的部署逻辑按职责边界重新聚合为 engine-group / deployment / disaggregation 三层。重点关注 engine_group.py 中 GPU 槽位校验逻辑（required_gpu_slots 判断）和 deployment.py 中 router 的启动与复用策略，这些是后续排查部署问题的基础。

功能与动机

PR 标题为 [cleanup] Refactor rollout.py，作者 zhuzilin 旨在清理 SGLang 部署相关代码的堆积。原有 rollout.py 同时承担 RolloutManager（训练侧数据转换）和 SGLang 引擎生命周期管理（ServerGroup、router 启动、PD/EPD 部署）两类职责，文件过长且耦合。通过拆分子模块，让单一文件聚焦单一职责，便于后续维护和独立测试。

实现拆解

1. 拆分 ServerGroup 到 engine_group.py：将原在 slime/ray/rollout.py 中的 ServerGroup 数据类原样迁移，并新增 ServerGroupPlacement 与 RolloutServer 封装，集中处理引擎组的创建、端口分配、offload/onload 等逻辑。
2. 新建 deployment.py 作为部署入口：将原先散落在 rollout.py 中的 _start_router、_compute_rollout_offset、_compute_megatron_num_gpus 和 start_rollout_servers 迁移至此函数，统一编排普通、PD、EPD 及外部 rollout 的启动流程。
3. 拆分 disaggregation.py：将 PD（prefill/decode）和 EPD（encoder/decoder）特有的启动序列抽象为 start_pd_server_groups 和 start_epd_server_groups，保持部署流程的可读性。
4. sglang_config.py新增resolve_sglang_config：将原来rollout.py中解析sglang_config、零 GPU 配置、prefill_num_servers 逻辑统一收口，作为部署模块的配置解析入口。
5. rollout.py 瘦身：删除约 669 行部署相关代码，仅保留 RolloutManager 的初始化、generate、rollout 数据转换等训练侧逻辑，并改为从 deployment 导入 start_rollout_servers。
6. 测试与 CI 调整：tests/utils/test_sglang_config.py 更新导入路径，新增针对 start_rollout_servers 路由到 PD 部署的测试，并 mock 掉引擎启动等待（fail_if_waited）。CI 配置中删除不再适用的测试文件引用。

关键文件：

- slime/backends/sglang_utils/engine_group.py（模块 引擎组；类别 source；类型 dependency-wiring；符号 ServerGroup, ServerGroupPlacement, RolloutServer, start_engines）：核心新增文件，承接原 rollout.py 中的 ServerGroup 与引擎启动生命周期，是部署逻辑的基础设施。
- slime/backends/sglang_utils/deployment.py（模块 部署编排；类别 source；类型 core-logic；符号 _start_router, _compute_rollout_offset, _compute_megatron_num_gpus, start_rollout_servers）：新增部署编排入口，负责 router 启动和 rollout server 集群的创建，是普通、PD、EPD 部署的统一门面。
- slime/backends/sglang_utils/sglang_config.py（模块 配置解析；类别 source；类型 core-logic；符号 resolve_sglang_config）：新增 resolve_sglang_config 配置解析函数，将 sglang_config、零 GPU 等分支逻辑统一，是部署模块的配置入口。
- slime/ray/rollout.py（模块 Rollout 管理器；类别 source；类型 dependency-wiring；符号 RolloutManager, start_rollout_servers）：瘦身主体，删除约 669 行部署代码，仅保留 RolloutManager 训练侧逻辑，是本次重构的直接受益文件。
- tests/utils/test_sglang_config.py（模块 配置测试；类别 test；类型 test-coverage；符号 fake_ray_get, fail_if_waited, test_start_rollout_servers_routes_pd_to_disaggregated_deployment）：同步更新导入路径，并新增 start_rollout_servers 路由到 PD 部署的测试，保证重构后行为可验证。
- .github/workflows/pr-test.yml（模块 CI 配置；类别 infra；类型 infrastructure）：CI 配置移除已删除测试文件相关条目，保持 CI 与代码同步。

关键符号：ServerGroup.start_engines, ServerGroup.parallel_config, start_pd_server_groups, start_epd_server_groups, start_rollout_servers, _start_router, _compute_rollout_offset, _compute_megatron_num_gpus, resolve_sglang_config

关键源码片段

slime/backends/sglang_utils/engine_group.py

核心新增文件，承接原 rollout.py 中的 ServerGroup 与引擎启动生命周期，是部署逻辑的基础设施。

```
# slime/backends/sglang_utils/engine_group.py —— 核心引擎组定义
@dataclasses.dataclass
class ServerGroup:
    """一组同构 SGLang 引擎，共享相同的 tp_size / nodes_per_engine / pg。

    一个 RolloutServer 可包含多个 ServerGroup（例如 PD 分离的 prefill 与 decode）。
    """

    args: Any
    pg: Any # (placement_group, reordered_bundle_indices, reordered_gpu_ids)
    all_engines: list
    num_gpus_per_engine: int
    num_new_engines: int
```

```
worker_type: str = "regular" # "regular", "prefill", "decode" 或 "placeholder"
rank_offset: int = 0 # 本组之前的累计引擎数
gpu_offset: int = 0 # 本组之前的累计 GPU 数
sglang_overrides: dict = dataclasses.field(default_factory=dict)
needs_offload: bool = False # 本组 GPU 是否与 megatron 重叠
model_path: str | None = None # 用于 update_weights_from_disk 的 checkpoint 路径
router_ip: str | None = None
router_port: int | None = None

@property
def nodes_per_engine(self):
    # 多节点部署时，每个引擎可能占用多节点，这里计算引擎占用的节点数
    return max(1, self.num_gpus_per_engine // self.args.num_gpus_per_node)

@property
def engines(self):
    """仅返回每个引擎在 node 0 上的实例（多节点服务场景下）。"""
    return self.all_engines[:: self.nodes_per_engine]

def start_engines(self, port_cursors: dict[int, int] | None = None) -> tuple[list, dict[int, int]]:
    """创建 Ray actor、分配端口并触发 engine.init()，不阻塞等待。

    返回 (init_handles, port_cursors)，调用方通过 ray.get() 等待引擎就绪。
    port_cursors 用于不同组之间避免端口竞争。
    """
    if port_cursors is None:
        port_cursors = {}
    if self.args.debug_train_only or self.worker_type == "placeholder":
        self.num_new_engines = 0
        return [], port_cursors

    num_gpus_per_engine_on_node = min(self.num_gpus_per_engine, self.args.num_gpus_per_
    node)
    pg, reordered_bundle_indices, reordered_gpu_ids = self.pg
    num_engines = len(self.all_engines)
    required_gpu_slots = self.gpu_offset + num_engines * num_gpus_per_engine_on_node
    # 校验放置组内 GPU 槽位是否足够，避免配置不匹配导致的运行时错误
    if num_engines and not (
        self.gpu_offset >= 0 and num_gpus_per_engine_on_node > 0 and required_gpu_slots <=
        len(reordered_gpu_ids)
    ):
        raise ValueError(
            "Invalid rollout server group GPU placement: "
            f"worker_type={self.worker_type}, "
            f"gpu_offset={self.gpu_offset}, "
            f"num_gpus_per_engine={self.num_gpus_per_engine}, "
            f"required_gpu_slots={required_gpu_slots}, "
            f"len(reordered_gpu_ids)={len(reordered_gpu_ids)}. "
            "Please align --rollout-num-gpus, --rollout-num-gpus-per-engine, "
```

```
            "and --sglang-config server_groups."
        )
```

slime/backends/sglang_utils/deployment.py

新增部署编排入口，负责 router 启动和 rollout server 集群的创建，是普通、PD、EPD 部署的统一门面。

```
# slime/backends/sglang_utils/deployment.py —— 部署统一入口
from slime.backends.sglang_utils.disaggregation import start_epd_server_groups, start_pd_
server_groups
from slime.backends.sglang_utils.engine_group import RolloutServer, ServerGroupPlacement
from slime.backends.sglang_utils.external import start_external_rollout_servers
from slime.backends.sglang_utils.sglang_config import resolve_sglang_config


def start_rollout_servers(args, pg) -> tuple[dict[str, Any], list[Any]]:
    """启动配置的 rollout 服务器，不等待最终引擎初始化完成。"""
    if args.rollout_external:
        return start_external_rollout_servers(args, start_router=_start_router)

    config = resolve_sglang_config(args)
    placement = ServerGroupPlacement(
        args=args,
        pg=pg,
        rollout_pg_offset=_compute_rollout_offset(args),
        megatron_num_gpus=_compute_megatron_num_gpus(args),
    )

    servers: dict[str, RolloutServer] = {}
    pending_init_handles: list[Any] = []

    for model_idx, model_config in enumerate(config.models):
        model_config.resolve(args)

        # 第一个模型复用用户指定的 router，后续模型强制新起 router
        router_ip, router_port = _start_router(
            args,
            has_pd_disaggregation=model_config.has_pd_disaggregation,
            force_new=(model_idx > 0),
        )
        if model_idx == 0:
            args.sglang_router_ip = router_ip
            args.sglang_router_port = router_port

        if model_config.has_encoder_disaggregation:
            server_groups, init_handles = start_epd_server_groups(
                model_config, placement, router_ip, router_port
            )
        elif model_config.has_pd_disaggregation:
```

```
        server_groups, init_handles = start_pd_server_groups(
            model_config, placement, router_ip, router_port
        )
    else:
        # 普通场景：逐个启动 server group，每个 group 按端口游标分配端口
        server_groups = []
        init_handles = []
        port_cursors: dict[int, int] = {}
        for group_config in model_config.server_groups:
            group = placement.create(group_config, router_ip, router_port)
            handles, port_cursors = group.start_engines(port_cursors)
            init_handles.extend(handles)
            server_groups.append(group)
    # ... 随后将 server_groups 与 init_handles 汇总到 servers / pending_init_handles
```

slime/backends/sglang_utils/sglang_config.py

新增 resolve_sglang_config 配置解析函数，将 sglang_config、零 GPU 等分支逻辑统一，是部署模块的配置入口。

```
# slime/backends/sglang_utils/sglang_config.py —— 配置统一解析入口
def resolve_sglang_config(args) -> SglangConfig:
    """解析显式配置、旧式 PD 配置或默认的 SGLang 部署配置。"""
    if getattr(args, "sglang_config", None) is not None:
        config = SglangConfig.from_yaml(args.sglang_config)
        expected = args.rollout_num_gpus
        actual = config.total_num_gpus
        # 显式配置必须与 rollout_num_gpus 对齐，防止 GPU 数量不匹配
        assert actual == expected, f"sglang_config total GPUs ({actual}) != rollout_num_gpus
        ({expected})"
        return config

    if args.rollout_num_gpus == 0:
        # 零 GPU 部署：不启动任何引擎，仅保留一个空模型默认项
        return SglangConfig(models=[ModelConfig(name="default", server_groups=[])])

    if args.prefill_num_servers is not None:
        # 兼容旧的 prefill_num_servers 参数，转为 PD 部署配置
        return SglangConfig.from_prefill_num_servers(args)

    # 默认场景：单模型单组，引擎数等于 rollout_num_gpus
    return SglangConfig(
        models=[
            ModelConfig(
                name="default",
                server_groups=[ServerGroupConfig(worker_type="regular", num_gpus=args.rollout_
                num_gpus)],
            )
        ]
    )
```

评论区精华

该 PR 没有收到任何 review 评论或讨论线程。从提交历史看，第二次提交 "prune removed test" 是对测试引用的清理，说明作者自查了因删除代码造成的测试断链。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 行为等价性风险：大部分代码是原样搬迁，但 _start_router 中 force_new 语义、_compute_rollout_offset 对 debug/colocate 模式的处理等细节可能与原实现略有出入，需回归验证。
  2. 模块间依赖耦合：engine_group.py 引用了 slime/ray/utils 中的 NOSET_VISIBLE_DEVICES_ENV_VARS_LIST 和 add_default_ray_env_vars，反向依赖 rollout 层，后续若继续拆分需注意避免循环依赖。
  3. 测试覆盖缺口：虽然测试文件更新了导入，但新增的 ServerGroupPlacement 和 RolloutServer 类缺少直接单元测试，PD 路由测试只有一条，EPD 分支未显式覆盖。
  4. CI 配置薄化：pr-test.yml 删除了部分测试条目，需确认是否真的不再需要，避免遗漏必要的回归测试。 - 影响：影响面集中在 SGLang 部署链路：所有走 rollout 的训练任务都会经过新部署模块。对用户而言，行为应完全一致；对开发者而言，需要适应新的代码组织方式，后续在 rollout 部署相关改动需定位到 sglang_utils 下。该重构降低了 rollout.py 的认知负担，提高了可测试性，也方便未来支持更多 SGLang 部署模式。 - 风险标记：行为等价重构，模块拆分后依赖关系变化，新增测试覆盖有限，无 Code Review 记录

关联脉络

- PR #2322 [cleanup] Remove rollout_validation.py: 同为 rollout 相关清理 PR，将 GPU 放置校验内联到 rollout.py，与本 PR 的模块拆分方向一致。
- PR #2321 [cleanup] extract create_weight_updater to make actor's init func cleaner: 同类瘦身重构，通过工厂函数精简初始化，与本 PR 的拆分思路相互呼应。
- PR #2216 feat: add backend-aware MUSA support: 涉及后端抽象层，本 PR 拆分出的 sglang_utils 模块与后端适配紧密相关。