PR #6117 完整报告

verl-project/verl

[sglang] feat: SGLang Prefill-Decode disaggregated rollout

合并时间：2026-05-06 19:07

原文链接：http://prhub.com.cn/verl-project/verl/pull/6117

执行摘要

- 一句话：SGLang Prefill-Decode 非对称分解 rollout
- 推荐动作：值得精读，尤其关注以下设计决策：
  1. 非对称 TP 布局：*decode* 可使用与 prefill 不同的 TP 大小，提供了灵活性。
  2. 取消独立注册名：通过 flag 而非独立名称派发，降低了用户心智负担，也简化了代码。
  3. rank-modulo 路由：通过取模确保即使未来启用 DP>1，角色映射仍正确。
  4. TOCTOU 防护：使用 with_alive_sock=True 避免端口被抢占。 这些设计模式对其他并行策略的实现有借鉴意义。

功能与动机

根据 Issue #5836 的 roadmap 规划，rollout 引擎性能优化的重要一环是 PD disaggregation。在 SGLang 单调度器架构下，当 *decode* 负载较高时，调度器会成为瓶颈。通过分离 prefill 和 *decode* 角色，可以独立调度这两类请求，从而降低 step 延迟并提升吞吐。PR body 明确描述目标：‘Adds PD-disaggregated SGLang rollout with asymmetric 1 prefill : N *decode* layout and per-rank role routing that keeps CUDA IPC handles on the same physical GPU.’

实现拆解

1. 配置层：新增 DisaggregationConfig 数据类（verl/workers/config/disaggregation.py），定义 *enabled*、prefill_replicas、*decode_replicas*、*decode_tensor_model_parallel_size*、transfer_backend、bootstrap_port 等参数。在 RolloutConfig（verl/workers/config/rollout.py）中引入该字段，并在初始化时校验：只有 name='sglang' 时才允许启用 PD。同步更新了 rollout.yaml 等默认 YAML 配置。
2. 副本层：新增 SGLangPDReplica 类（verl/workers/rollout/sglang_rollout/sglang_pd_replica.py），继承自 SGLangReplica。构造函数校验约束（data_parallel_size=1、prefill_replicas=1、GPU 不超节点限制），计算 world_size。launch_servers 方法负责：收集 worker 节点信息；分配 bootstrap 端口（含 TOCTOU 防护）；按角色分组并启动 prefill 和 *decode* 的 Ray actor；通过 set_pd_peer 将 *decode* 服务器地址传递给 prefill 服务器。
3. 服务器角色化：修改 SGLangHttpServer（async_sglang_server.py），构造函数增加 disaggregation_role 和 disaggregation_bootstrap_port 参数；添加 set_pd_peer 方法；

添加 _prepend_cu12_lib_to_ld_library_path 解决 NIXL 依赖的 libcudart.so.12 路径问题。launch_server 在 PD 模式下传递 --disaggregation 参数给 SGLang 进程。

4. 路由与 World Size：修改 ServerAdapter.__init__（sglang_rollout.py），根据 PD 配置计算 rollout_world_size（含 PP 因子），并为每个 rank 计算 _pd_role 和 _pd_server_index。非 TP leader 也参与 sgl_update_weights 以保证 gather_object 集合通信。
5. 分发与注册：修改 get_rollout_replica_class（replica.py），新增 disaggregation_enabled 参数，当为 True 且 rollout='sglang' 时返回 SGLangPDReplica，否则抛出异常。移除了 sglang_pd 和 vllm_pd 独立注册名。
6. 测试覆盖：新增 tests/workers/rollout/test_pd_disaggregation.py，包含 17 个 GPU 无关的单元测试，覆盖配置默认值、有效 / 无效 backend、零副本拒绝、端口范围、effective_decode_tp 方法以及 get_rollout_replica_class 分发正确性。

关键文件：

- verl/workers/rollout/sglang_rollout/sglang_pd_replica.py（模块 rollout；类别 source；类型 dependency-wiring；符号 SGLangPDReplica, init, launch_servers, _fmt）：新增核心文件：实现 PD 副本类，负责启动和管理 prefill/decode 服务器集群，是整个 PD 功能的核心载体。
- verl/workers/config/disaggregation.py（模块 配置；类别 source；类型 dependency-wiring；符号 DisaggregationConfig, post_init, effective_decode_tp）：新增配置类，定义 PD 所有参数并包含验证逻辑和辅助方法，是配置入口。
- tests/workers/rollout/test_pd_disaggregation.py（模块 测试；类别 test；类型 test-coverage；符号 test_disaggregation_defaults_disabled_and_valid, test_disaggregation_enabled_nixl_accepted, test_disaggregation_enabled_mooncake_accepted, test_disaggregation_unknown_backend_rejected）：全面的 GPU 无关配置测试，覆盖配置验证、调度分发和后端兼容性，确保核心逻辑正确性。
- verl/workers/rollout/sglang_rollout/async_sglang_server.py（模块 rollout；类别 source；类型 dependency-wiring；符号 set_pd_peer, _prepend_cu12_lib_to_ld_library_path）：修改 SGLang 服务器构造函数，支持角色参数和 PD 同伴通信，并添加 LD_LIBRARY_PATH 修正方法。
- verl/workers/rollout/sglang_rollout/sglang_rollout.py（模块 rollout；类别 source；类型 core-logic；符号 _is_server_tp_leader）：修改 ServerAdapter 的 world size 计算和角色路由逻辑，是 PD 切换的核心适配点。
- verl/workers/rollout/replica.py（模块 rollout；类别 source；类型 core-logic；符号 get_rollout_replica_class）：修改 get_rollout_replica_class 以支持通过 disaggregation_enabled 标志派发，替代独立注册名方案。
- verl/workers/config/rollout.py（模块 配置；类别 source；类型 dependency-wiring）：在 RolloutConfig 中集成 DisaggregationConfig，并添加校验逻辑（非 sglang 后端不允许启用 PD）。
- verl/workers/rollout/llm_server.py（模块 rollout；类别 source；类型 core-logic）：修改 agent loop 入口，传递 disaggregation 配置到 LLM server 初始化。

关键符号：SGLangPDReplica.init, SGLangPDReplica.launch_servers, SGLangPDReplica._fmt, SGLangPDReplica._collect_cuda_devices, SGLangPDReplica._launch_one, DisaggregationConfig.post_init, DisaggregationConfig.effective_decode_tp, SGLangHttpServer.set_pd_peer, SGLangHttpServer._prepend_cu12_lib_to_ld_library_path, get_rollout_replica_class, _is_server_tp_leader

关键源码片段

verl/workers/rollout/sglang_rollout/sglang_pd_replica.py

新增核心文件：实现 PD 副本类，负责启动和管理 prefill/decode 服务器集群，是整个 PD 功能的核心载体。

```
import asyncio
import logging
import os
from dataclasses import replace as _dc_replace
from typing import Optional

import ray
from omegaconf import DictConfig
from ray.actor import ActorHandle

from verl.utils.device import is_torch_npu_available
from verl.utils.net_utils import get_free_port, is_valid_ipv6_address
from verl.workers.config import RolloutConfig
from verl.workers.rollout.sglang_rollout.async_sglang_server import (
    SGLangReplica,
    visible_devices_keyword,
)

logger = logging.getLogger(__file__)
logger.setLevel(logging.INFO)


class SGLangPDReplica(SGLangReplica):
    """Replica that runs SGLang in prefill-decode disaggregated mode."""

    def __init__(
        self,
        replica_rank: int,
        config: RolloutConfig,
        model_config: DictConfig,
        gpus_per_node: int = 8,
        is_reward_model: bool = False,
        is_teacher_model: bool = False,
    ):
        # 调用父类初始化，获得基础配置与 worker 分配
```

```
super().__init__(
    replica_rank,
    config,
    model_config,
    gpus_per_node,
    is_reward_model,
    is_teacher_model,
)
disagg = self.config.disaggregation
assert disagg.enabled, "SGLangPDReplica requires rollout.disaggregation.enabled=True"

# MVP 限制：prefill_replicas 仅支持 1
if disagg.prefill_replicas != 1:
    raise NotImplementedError(f"prefill_replicas=1 only (got {disagg.prefill_replicas})")
self._n_prefill = disagg.prefill_replicas
self._n_decode = disagg.decode_replicas

self._prefill_tp = self.config.tensor_model_parallel_size
# 如果 decode_tp 未指定，默认与 prefill_tp 相同
self._decode_tp = (
    disagg.decode_tensor_model_parallel_size
    if disagg.decode_tensor_model_parallel_size is not None
    else self._prefill_tp
)

# 计算 PD 需要的总 GPU 数，并检查不超过节点 GPU 上限
pd_world_size = self._prefill_tp + self._n_decode * self._decode_tp
if pd_world_size > gpus_per_node:
    raise NotImplementedError(
        f"PD replica needs {pd_world_size} GPUs but gpus_per_node={gpus_per_node}; "
        f"use more replicas to span nodes"
    )

# 强制 data_parallel_size = 1（当前版本限制）
if self.config.data_parallel_size != 1:
    raise NotImplementedError(
        f"data_parallel_size=1 only (got {self.config.data_parallel_size})"
    )
self.world_size = pd_world_size
self.gpus_per_replica_node = min(self.gpus_per_node, self.world_size)
assert self.world_size % self.gpus_per_replica_node == 0
self.nnodes = self.world_size // self.gpus_per_replica_node

# 存储角色服务器句柄的初始容器
self._prefill_servers: list[ActorHandle] = []
self._decode_servers: list[ActorHandle] = []
self._prefill_server_address: Optional[str] = None
self._decode_server_addresses: list[str] = []
self._bootstrap_port: Optional[int] = None
```

verl/workers/config/disaggregation.py

新增配置类，定义 PD 所有参数并包含验证逻辑和辅助方法，是配置入口。

```
from dataclasses import dataclass
from typing import Optional

from verl.base_config import BaseConfig

__all__ = ["DisaggregationConfig"]

_ALLOWED_BACKENDS = ("nixl", "mooncake", "ascend", "mori", "fake")


@dataclass
class DisaggregationConfig(BaseConfig):
    """Prefill-Decode disaggregation knobs (SGLang only)."""

    enabled: bool = False # 全局开关，默认关闭
    prefill_replicas: int = 1 # 预填充副本数（MVP 固定为 1）
    decode_replicas: int = 1 # 解码副本数
    decode_tensor_model_parallel_size: Optional[int] = None # 解码 TP 大小
    transfer_backend: str = "nixl" # 跨节点传输后端
    bootstrap_port: Optional[int] = None # 指定端口（自动分配时为 None）
    ib_device: Optional[str] = None # 用于 mooncake 的 IB 设备名

    def __post_init__(self) -> None:
        # 只在启用时校验，否则跳过，以兼容默认 YAML 坏值
        if not self.enabled:
            return
        if self.transfer_backend not in _ALLOWED_BACKENDS:
            raise ValueError(
                f"disaggregation.transfer_backend={self.transfer_backend!r} not in {_ALLOWED_
                BACKENDS}"
            )
        if self.prefill_replicas < 1 or self.decode_replicas < 1:
            raise ValueError(
                f"disaggregation requires >=1 prefill and >=1 decode replica "
                f"(got prefill_replicas={self.prefill_replicas}, decode_replicas={self.decode_replicas})"
            )
        if self.bootstrap_port is not None and not (0 < self.bootstrap_port < 65536):
            raise ValueError(f"bootstrap_port out of range: {self.bootstrap_port}")

    def effective_decode_tp(self, prefill_tp: int) -> int:
        """助手方法，返回有效的 decode TP（默认与 prefill TP 相同）。"""
        if self.decode_tensor_model_parallel_size is not None:
            return self.decode_tensor_model_parallel_size
        return prefill_tp
```

tests/workers/rollout/test_pd_disaggregation.py

全面的 GPU 无关配置测试，覆盖配置验证、调度分发和后端兼容性，确保核心逻辑正确性。

```
"""GPU-free unit tests for PD disaggregation config + replica plumbing."""
from __future__ import annotations

import pytest
from verl.workers.config import DisaggregationConfig, RolloutConfig


def test_disaggregation_defaults_disabled_and_valid():
    # 默认构造：enabled=False 且各字段为合理默认值
    cfg = DisaggregationConfig()
    assert cfg.enabled is False
    assert cfg.prefill_replicas == 1
    assert cfg.decode_replicas == 1
    assert cfg.transfer_backend == "nixl"
    assert cfg.bootstrap_port is None
    assert cfg.ib_device is None


def test_disaggregation_enabled_nixl_accepted():
    # nixl 是允许的 backend，应通过校验
    cfg = DisaggregationConfig(enabled=True, transfer_backend="nixl")
    assert cfg.enabled is True and cfg.transfer_backend == "nixl"


def test_disaggregation_enabled_mooncake_accepted():
    # mooncake 也允许，同时指定 ib_device
    cfg = DisaggregationConfig(enabled=True, transfer_backend="mooncake", ib_device="mlx5_
    roce0")
    assert cfg.transfer_backend == "mooncake"


def test_disaggregation_unknown_backend_rejected():
    # 未知 backend 应抛出 ValueError
    with pytest.raises(ValueError, match="transfer_backend"):
        DisaggregationConfig(enabled=True, transfer_backend="bogus")


def test_disaggregation_zero_replicas_rejected():
    # 副本数为 0 应被拒绝
    with pytest.raises(ValueError, match="prefill_replicas"):
        DisaggregationConfig(enabled=True, prefill_replicas=0)
    with pytest.raises(ValueError, match="decode_replicas"):
        DisaggregationConfig(enabled=True, decode_replicas=0)


def test_disaggregation_bad_bootstrap_port_rejected():
    # 端口超出范围应被拒绝
```

```
    with pytest.raises(ValueError, match="bootstrap_port"):
        DisaggregationConfig(enabled=True, bootstrap_port=70000)


def test_disaggregation_disabled_skips_validation():
    # 当 enabled=False 时，即使字段无效也不报错（兼容 YAML 加载）
    cfg = DisaggregationConfig(enabled=False, transfer_backend="bogus", bootstrap_port=
    70000)
    assert cfg.enabled is False
```

评论区精华

Review 中核心讨论包括：

- rollout_world_size 缺失 PP 因子：gemini-code-assist[bot] 指出计算未考虑 pipeline_model_parallel_size，会导致 agent loop 中 replica_rank 分配错误。作者 yxs 已修复（commit 3b5a542）。
- PD role 假设 DP=1：bot 指出角色分配逻辑硬编码了 DP=1 假设。yxs 添加了 rollout_rank % footprint 的取模防御，但未实现 prefill_replicas>1 路径。
- 复用 sglang 后端 vs 独立注册名：维护者 wuxibin89 建议不添加 sglang_pd 注册名，而是重用 sglang 并通过 flag 区分。yxs 采纳并重构了 get_rollout_replica_class，移除了临时注册函数。
  - rollout_world_size 计算缺失 pipeline_model_parallel_size 因子 (correctness): 作者 yxs 在后续提交中添加了 PP 因子，并回复 'Fixed'。
  - PD role 分配逻辑假设 data_parallel_size=1 (design): yxs 添加了 rollout_rank % footprint 取模使得 DP>1 时每个组内角色解析正确，但未实现 prefill_replicas>1 路径。
  - 是否独立注册 sglang_pd 后端 (design): 废弃 sglang_pd/vllm_pd 名字，统一由配置标志驱动。

风险与影响

- 风险：
  1. 核心路径变更风险：rollout_world_size 计算逻辑变更影响到所有 SGLang 训练任务，虽然测试覆盖了配置层，但缺少端到端集成测试。
  2. NPU 限制：代码显式 assert not is_torch_npu_available()，NPU 用户无法使用此功能。
  3. NIXL 依赖：NIXL 后端需要正确的 LD_LIBRARY_PATH，虽已添加修正方法，但不同环境仍可能失败。
  4. 性能收益不固定：PR 文档表明 PD 的优势依赖于 decode 负载，轻负载下可能无提升甚至略差。
  5. 配置约束：当前强制 prefill_replicas=1、data_parallel_size=1，未来扩展时需多文件协调。 - 影响：影响范围：仅 SGLang 后端，通过配置 rollout.disaggregation.enabled =True 启用。不影响其他后端（vLLM、TRT-LLM）。已有非 PD 配置完全兼容。影响程度：中到高。新功能为 rollout 引擎带来显著性能提升（约 3-7% step 时间减少），但需要用户调整配置并理解负载特性。团队需要维护新的 SGLangPDReplica 和

DisaggregationConfig 模块。用户行为：用户需在 rollout 配置中增加 disaggregation 节，并确保传输后端参数正确。旧版通过 rollout.name='sglang_pd' 的用法已被放弃，改为 rollout.name='sglang' 加上 disaggregation.enabled=True。

- 风险标记：核心路径变更，NPU 未验证，NIXL 依赖，性能依赖负载，缺少集成测试

关联脉络

- PR #5836 [roadmap] verl 26Q2 roadmap: 该 PR 是 roadmap 中 PD disaggregated 任务的实现，关联 issue 5836。
- PR #6129 [BREAKING][rollout] refactor: move LLMServerManager out of AgentLoopManager: 本 PR 修改了 llm_server.py 文件，依赖于 #6129 对 LLM 服务器管理的重构，在此基础上添加角色支持。