PR #1874 完整报告

THUDM/slime

[docker] upgrade sglang to v0.5.10.post1

合并时间：2026-04-30 14:53

原文链接：http://prhub.com.cn/THUDM/slime/pull/1874

执行摘要

- 一句话：升级 sglang 至 v0.5.10.post1，新增 PD Mooncake 测试与 HF 配置修补
- 推荐动作：建议重点关注 megatron_bridge_utils.py 的新增函数，它们简化了模型加载配置；验证 PD Mooncake 测试的稳定性；升级后建议进行回归测试覆盖现有模型。

功能与动机

为支持 GLM-4.7 和 Qwen3.5 等新模型，需要升级 sglang 至最新版本并适配其 API 变更；同时，Megatron 桥接在加载 HF 模型时缺少 rope_theta 等配置项，需要自动修补以避免训练中断。

实现拆解

1. 升级 sglang 依赖：更新 docker/patch/latest/sglang.patch 和 docker/version.txt，适配 sglang v0.5.10.post1 的 API 变化。
2. 新增 HF 配置修补工具：在 slime/utils/megatron_bridge_utils.py 中添加 patch_hf_config_for_megatron_bridge 和 patch_auto_bridge_hf_config 函数，自动从 rope_parameters 或 rope_scaling 中提取 rope_theta 并注入配置对象，同时处理嵌套 text_config 和 PretrainedWrapper 结构。
3. 添加 PD Mooncake 集成测试：新增 tests/test_glm4.7_30B_A3B_pd_mooncake.py 和 tests/test_qwen3_30B_A3B_pd_mooncake.py，验证单节点 PD+Mooncake 模式下训练流程的正确性。
4. 修复共享权重映射：在 slime_plugins/mbridge/qwen3_5.py 中修改 _adjust_mapping_for_shared_weights 方法，当 tie_word_embeddings 为真时重定向 output_layer.weight 到合适的嵌入层。
5. 调整 CI 工作流：更新 .github/workflows/pr-test.yml 和 .github/workflows/pr-test.yml.j2，增加 privileged 权限以支持 docker 操作，并调整测试矩阵。

关键文件：

- slime/utils/megatron_bridge_utils.py（模块 桥接配置；类别 source；类型 core-logic；符号 patch_hf_config_for_megatron_bridge, add_config, patch_auto_bridge_hf_config）：新增核心配置修补逻辑，自动补充 HF 模型配置中的 rope_theta，确保与 Megatron 桥接兼容。

- tests/utils/test_megatron_bridge_utils.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_patch_hf_config_adds_rope_theta_from_rope_parameters, test_patch_hf_config_does_not_override_existing_rope_theta, test_patch_hf_config_handles_nested_text_config, test_patch_hf_config_handles_pretrained_wrapper_config）：对新增的配置修补函数进行完整单元测试，覆盖各种配置嵌套和回退场景。
- tests/test_glm4.7_30B_A3B_pd_mooncake.py（模块 集成测试；类别 test；类型 test-coverage；符号 prepare, write_sglang_config, execute）：新增 GLM-4.7 模型在 PD+Mooncake 模式下的端到端训练测试，验证新版本 sglang 和配置修补的集成。
- tests/test_qwen3_30B_A3B_pd_mooncake.py（模块 集成测试；类别 test；类型 test-coverage；符号 prepare, execute）：新增 Qwen3 30B A3B 模型在 PD Mooncake 模式下的集成测试。
- slime_plugins/mbridge/qwen3_5.py（模块 权重映射；类别 source；类型 core-logic；符号 _adjust_mapping_for_shared_weights）：修复 Qwen3.5 模型的共享权重映射，确保 output_layer.weight 指向正确的嵌入层。
- docker/patch/latest/sglang.patch（模块 补丁文件；类别 test；类型 test-coverage）：更新 sglang 补丁以适配 v0.5.10.post1，移除旧版本补丁内容。
- .github/workflows/pr-test.yml（模块 CI 配置；类别 infra；类型 infrastructure）：调整 CI 测试流程，添加 privileged 权限并更新测试命令。

关键符号：patch_hf_config_for_megatron_bridge, patch_auto_bridge_hf_config, _adjust_mapping_for_shared_weights

关键源码片段

slime/utils/megatron_bridge_utils.py

新增核心配置修补逻辑，自动补充 HF 模型配置中的 rope_theta，确保与 Megatron 桥接兼容。

```
from contextlib import contextmanager

try:
    from megatron.core.utils import unwrap_model
except ImportError:
    unwrap_model = None

# 核心函数：修补 HF 配置，确保 Megatron 桥接能读取 rope_theta 等字段
def patch_hf_config_for_megatron_bridge(hf_config):
    configs = []
    seen_config_ids = set()

    def add_config(config):
        # 去重并收集所有需修补的配置对象
        if config is None or id(config) in seen_config_ids:
            return
        seen_config_ids.add(id(config))
        configs.append(config)
```

```
    add_config(hf_config)
    add_config(getattr(hf_config, "config", None)) # 支持 PretrainedWrapper

    # 递归处理嵌套的 text_config（如视觉语言模型）
    for config in list(configs):
        add_config(getattr(config, "text_config", None))

    # 从 rope_parameters 或 rope_scaling 中提取 rope_theta
    for config in configs:
        rope_params = getattr(config, "rope_parameters", None) or getattr(config, "rope_scaling",
        None)
        if isinstance(rope_params, dict) and "rope_theta" in rope_params and not hasattr(config,
        "rope_theta"):
            config.rope_theta = rope_params["rope_theta"]

    return hf_config

# 快捷函数：修补 bridge 对象中的 hf_pretrained 配置
def patch_auto_bridge_hf_config(bridge):
    hf_pretrained = getattr(bridge, "hf_pretrained", None)
    if hf_pretrained is not None:
        patch_hf_config_for_megatron_bridge(hf_pretrained)
    return bridge

@contextmanager
def patch_megatron_model(model):
    # 原有上下文管理器，未变
    ...
```

tests/utils/test_megatron_bridge_utils.py

对新增的配置修补函数进行完整单元测试，覆盖各种配置嵌套和回退场景。

```
import types
import pytest
from slime.utils.megatron_bridge_utils import patch_auto_bridge_hf_config, patch_hf_config_for_
megatron_bridge

@pytest.mark.unit
def test_patch_hf_config_adds_rope_theta_from_rope_parameters():
    # 验证能从 rope_parameters 中提取 rope_theta
    hf_config = types.SimpleNamespace(rope_parameters={"rope_theta": 1000000})
    patched_config = patch_hf_config_for_megatron_bridge(hf_config)
    assert patched_config is hf_config
    assert hf_config.rope_theta == 1000000

@pytest.mark.unit
def test_patch_hf_config_does_not_override_existing_rope_theta():
    # 验证不覆盖已存在的 rope_theta
    hf_config = types.SimpleNamespace(rope_theta=500000, rope_parameters={"rope_theta":
```

```
    1000000})
    patch_hf_config_for_megatron_bridge(hf_config)
    assert hf_config.rope_theta == 500000

@pytest.mark.unit
def test_patch_hf_config_handles_nested_text_config():
    # 处理嵌套的 text_config（如多模态模型）
    text_config = types.SimpleNamespace(rope_parameters={"rope_theta": 10000})
    hf_config = types.SimpleNamespace(text_config=text_config)
    patch_hf_config_for_megatron_bridge(hf_config)
    assert text_config.rope_theta == 10000

@pytest.mark.unit
def test_patch_hf_config_handles_pretrained_wrapper_config():
    # 处理 PretrainedWrapper 结构（config 字段内层包裹）
    wrapped_config = types.SimpleNamespace(rope_parameters={"rope_theta": 10000})
    hf_pretrained = types.SimpleNamespace(config=wrapped_config)
    patch_hf_config_for_megatron_bridge(hf_pretrained)
    assert wrapped_config.rope_theta == 10000

@pytest.mark.unit
def test_patch_hf_config_uses_rope_scaling_fallback():
    # 回退到 rope_scaling 字段
    hf_config = types.SimpleNamespace(rope_scaling={"rope_theta": 10000})
    patch_hf_config_for_megatron_bridge(hf_config)
    assert hf_config.rope_theta == 10000

@pytest.mark.unit
def test_patch_auto_bridge_hf_config_patches_hf_pretrained():
    # 验证 patch_auto_bridge_hf_config 能正确修补 bridge 中的 hf_pretrained
    hf_config = types.SimpleNamespace(rope_parameters={"rope_theta": 12345})
    bridge = types.SimpleNamespace(hf_pretrained=hf_config)
    patched_bridge = patch_auto_bridge_hf_config(bridge)
    assert patched_bridge is bridge
    assert bridge.hf_pretrained.rope_theta == 12345
```

tests/test_glm4.7_30B_A3B_pd_mooncake.py

新增 GLM-4.7 模型在 PD+Mooncake 模式下的端到端训练测试，验证新版本 sglang 和配置修补的集成。

```
"""GLM-4.7-Flash colocated training test with single-node PD + Mooncake."""
import os
import tempfile
import yaml
import slime.utils.external_utils.command_utils as U

MODEL_REPO = "zai-org/GLM-4.7-Flash"
MODEL_NAME = "GLM-4.7-Flash"
MODEL_TYPE = "glm4.7-30B-A3B"
```

```
NUM_GPUS = 8

def prepare():
    # 下载模型、数据集并转换为 Megatron 格式
    U.exec_command("mkdir -p /root/models /root/datasets")
    U.exec_command(f"hf download {MODEL_REPO} --local-dir /root/models/{MODEL_NAME}")
    U.hf_download_dataset("zhuzilin/dapo-math-17k")
    U.convert_checkpoint(
        model_name=MODEL_NAME,
        megatron_model_type=MODEL_TYPE,
        num_gpus_per_node=NUM_GPUS,
        dir_dst="/root/models",
        hf_checkpoint=f"/root/models/{MODEL_NAME}",
    )

def write_sglang_config() -> str:
    # 生成 sglang 的 PD Mooncake 配置文件（prefill 和 decode 各 4 GPU）
    config = {
        "sglang": [
            {
                "name": "default",
                "server_groups": [
                    {
                        "worker_type": "prefill",
                        "num_gpus": 4,
                        "num_gpus_per_engine": 4,
                        "overrides": {"disaggregation_transfer_backend": "mooncake"},
                    },
                    {
                        "worker_type": "decode",
                        "num_gpus": 4,
                        "num_gpus_per_engine": 4,
                        "overrides": {"disaggregation_transfer_backend": "mooncake"},
                    },
                ],
            }
        ]
    }
    f = tempfile.NamedTemporaryFile("w", suffix=".yaml", prefix="sglang_pd_mooncake_", delete=
    False)
    with f:
        yaml.safe_dump(config, f, sort_keys=False)
    return f.name

def execute():
    # 组装训练命令并执行
    sglang_config = write_sglang_config()
    # ... 后接各种参数拼接和 U.execute_train 调用
```

评论区精华

无审核评论，但提交历史显示多次 bugfix 迭代（如修复网络 host 模式、Qwen3MoE 问题），表明升级过程中遇到了多个兼容性问题并逐一解决。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 升级兼容性：sglang v0.5.10.post1 可能引入不兼容的 API 变更，需要验证现有 rollout 功能正常。
  2. 新配置路径：patch_hf_config_for_megatron_bridge 可能漏掉某些自定义模型配置，导致桥接加载失败。
  3. PD Mooncake 测试：新测试依赖 distributed 环境，若单机多卡配置不当可能不稳定。
  4. CI 权限提升：privileged 模式可能带来安全风险，但限于 CI 环境。 - 影响：影响范围：所有使用 sglang rollout 的用户（需重建 Docker 镜像）；使用 Megatron 桥接加载 HF 模型的用户（自动获得配置修补）；新增的 GLM-4.7 和 Qwen3 30B A3B PD Mooncake 用户；CI 流程。影响程度：中等，主要影响训练流程的兼容性和新模型支持。 - 风险标记：升级依赖兼容性，新配置路径测试覆盖，CI 权限提升

关联脉络

- PR #1856 refactor/ppo: 同为较大架构变更，涉及 actor-critic 配置解耦，与本次 Megatron 桥接配置改进相关。
- PR #1866 Rename critic config to megatron config: 重命名配置，与本 PR 新增的配置修补函数共同完善 Megatron 桥接层。
- PR #1867 [docker] upgrade megatron to 1dcf0dafa: 同为依赖升级，与本 PR 的 sglang 升级形成配套。