PR #6344 完整报告

verl-project/verl

[rollout] fix: fix fp8 for async RL and multinode rollout

合并时间：2026-05-23 08:10

原文链接：http://prhub.com.cn/verl-project/verl/pull/6344

执行摘要

- 一句话：修复 FP8 在 async RL 和多节点 rollout 中的 bug
- 推荐动作：值得关注的设计决策：将全局副作用改为 opt-in 环境变量，避免影响不相关场景；FP8 配置的安全访问方式（先调用父类再使用 self.config）可以作为代码规范。该 PR 适合 rollout 和 trainer 相关开发人员阅读。

功能与动机

PR 描述指出两个修复点：'Fix FP8 config code path with async rl which uses standalone mode' 以及 'Add TLLM_DISABLE_NVLS_MNNVL to limit these env changes to TRTLLM async RL only, as NCCL_MNNVL_ENABLE=0 cause multinode rollout hang in TRTLLM sync rl.' 即修复 FP8 配置路径，并将 NCCL 环境变量限制在异步 RL 下，避免同步 RL 挂起。

实现拆解

1. 修复 FP8 下 dummy load_format 绕过(trtllm_async_server.py)：在 TRTLLMHttpServer.__init__ 中，原来对非 HYBRID 且 load_format 为 dummy 时会强制设为 auto；现在增加条件，当 quantization == "fp8" 时保留 dummy，因为 FP8 权重在首次同步时填充，没有磁盘检查点。
2. 修复 KvCacheConfig 构建(trtllm_async_server.py)：在 launch_server 中，将原本直接传递给 KvCacheConfig 构造函数的关键字参数改为先构建字典再展开，避免后续 **engine_kwargs 展开时覆盖 KvCacheConfig 对象。
3. 调整 ServerAdapter 初始化顺序并添加 null 检查(trtllm_rollout.py)：在 ServerAdapter.__init__ 中，将 super().__init__ 调用提前，以便通过 self.config 访问 quantization；同时添加对 self.model_config.hf_config is not None 的保护检查，避免 AttributeError。
4. 将 GB200 NCCL WAR 改为 opt-in(constants_ppo.py)：原代码在 SM major>=10 时无条件禁用 NVLS 和 MNNVL，现改为仅当环境变量 TLLM_DISABLE_NVLS_MNNVL 为 1 时才设置，限制在 async RL 场景。
5. 更新 fully_async shell 脚本(两个 shell 脚本)：添加 export TLLM_DISABLE_NVLS_MNNVL=1 和 FP8 启用注释，确保 async RL 用户在 GB200 上正常训练。

关键文件：

- verl/workers/rollout/trtllm_rollout/trtllm_async_server.py（模块 rollout；类别 source；类型 core-logic；符号 TRTLLMHttpServer.init, TRTLLMHttpServer.launch_server）：核心逻辑变更：修复 FP8 下 dummy load_format 的绕过逻辑，并优化 KvCacheConfig 构建方式
- verl/trainer/constants_ppo.py（模块 训练配置；类别 source；类型 core-logic；符号 _gb200_nccl_env, PPO_RAY_RUNTIME_ENV）：将 GB200 NCCL WAR 从无条件禁用改为通过环境变量 TLLM_DISABLE_NVLS_MNNVL opt-in，避免影响 sync RL
- verl/workers/rollout/trtllm_rollout/trtllm_rollout.py（模块 rollout；类别 source；类型 core-logic；符号 ServerAdapter.init）：修复 FP8 量化配置设置：将 super().__init__提前，并添加 hf_config 的 None 检查
- verl/experimental/fully_async_policy/shell/grpo_30b_a3b_base_math_megatron_8_8_mis_trtllm.sh（模块 启动脚本；类别 other；类型 configuration）：为 fully_async shell 添加 TLLM_DISABLE_NVLS_MNNVL 环境变量以及 FP8 启用注释
- verl/experimental/fully_async_policy/shell/grpo_8b_base_math_megatron_4_4_trtllm.sh（模块 启动脚本；类别 other；类型 configuration）：为 fully_async shell 添加 TLLM_DISABLE_NVLS_MNNVL 环境变量以及 FP8 启用注释

关键符号：TRTLLMHttpServer.init, TRTLLMHttpServer.launch_server, ServerAdapter.init

关键源码片段

verl/workers/rollout/trtllm_rollout/trtllm_async_server.py

核心逻辑变更：修复 FP8 下 dummy load_format 的绕过逻辑，并优化 KvCacheConfig 构建方式

```
# trtllm_async_server.py 关键改动：FP8 量化绕过 dummy load_format 检查

def __init__(self, ...):
    ...
    # 非 HYBRID 模式 + load_format=dummy 通常需要从磁盘加载（设为 auto）
    # 例外：FP8 没有磁盘检查点，权重在第一次同步时填充，因此保持 dummy
    if (
        self.rollout_mode != RolloutMode.HYBRID
        and self.config.load_format == "dummy"
        and self.config.quantization != "fp8" # 新增条件，避免 FP8 被错误设为 auto
    ):
        logger.warning(...)
        self.config.load_format = "auto"

async def launch_server(self):
    ...
    # 将 KvCacheConfig 参数先构建字典再展开，防止 **engine_kwargs 覆盖整个对象
    kv_cache_kwargs = {
        "enable_block_reuse": self.config.enable_prefix_caching,
        "free_gpu_memory_fraction": self.config.gpu_memory_utilization,
        **kv_cache_overrides,
```

```
    }
    kv_cache_config = KvCacheConfig(**kv_cache_kwargs)
```

verl/trainer/constants_ppo.py

将 GB200 NCCL WAR 从无条件禁用改为通过环境变量 TLLM_DISABLE_NVLS_MNNVL opt-in，避免影响 sync RL

```
# constants_ppo.py 关键改动：GB200 NCCL 环境变量从全局禁用到 opt-in

_major, _ = get_device_capability()
# 注释：Opt-in GB200 NCCL WAR：设置 TLLM_DISABLE_NVLS_MNNVL=1 来禁用 Blackwell 上的
# NCCL_NVLS_ENABLE 和 NCCL_MNNVL_ENABLE。async-RL Megatron 在无 IMEX 的 GB200
节点上需要此设置，
# 否则 mbridge export_weights -> all_gather 会引发 NCCL 801 错误。
_gb200_nccl_env = {}
if (_major or 0) >= 10 and os.environ.get("TLLM_DISABLE_NVLS_MNNVL", "0") == "1":
    _gb200_nccl_env = {"NCCL_NVLS_ENABLE": "0", "NCCL_MNNVL_ENABLE": "0"}

PPO_RAY_RUNTIME_ENV = {
    "env_vars": {
        ...
        **_gb200_nccl_env, # 只有条件满足时才注入禁用环境变量
    },
}
```

verl/workers/rollout/trtllm_rollout/trtllm_rollout.py

修复 FP8 量化配置设置：将 super().__init__提前，并添加 hf_config 的 None 检查

```
# trtllm_rollout.py ServerAdapter.__init__ 关键改动

def __init__(self, config, model_config, device_mesh, replica_rank=-1):
    # 先调用父类初始化，才能安全访问 self.config
    super().__init__(config, model_config, device_mesh)
    # FP8 量化配置：仅在启用 fp8 且 hf_config 存在时设置 quantization_config
    if self.config.quantization == "fp8" and self.model_config.hf_config is not None:
        FP8_BLOCK_QUANT_KWARGS = {
            "activation_scheme": "dynamic",
            "fmt": "e4m3",
            "quant_method": "fp8",
            "weight_block_size": [128, 128],
        }
        self.model_config.hf_config.quantization_config = dict(FP8_BLOCK_QUANT_KWARGS)
    ...
```

评论区精华

在 trtllm_rollout.py 上，gemini-code-assist 建议使用 self.config.quantization 代替 config.get()，并添加 hf_config 为 None 的检查。该建议被采纳，最终代码中已体现。

- FP8 配置安全访问和 null 检查 (correctness): 已采纳，在最终代码中体现

风险与影响

- 风险：回归风险：在 sync RL 场景下，之前的全局 NCCL 禁用被移除，sync RL 用户不再受 NCCL_MNNVL_ENABLE=0 导致的挂起影响，但 async RL 用户需要显式设置 TLLM_DISABLE_NVLS_MNNVL=1 才能避免 GB200 的 NCCL 问题。FP8 配置路径：init 顺序调整可能影响其他初始化代码，但改动后更安全，且只影响 FP8 用户。兼容性：环境变量新增，旧脚本无需修改，但若有自定义启动脚本未设置该环境变量，在 GB200 async RL 上可能遇到 NCCL 错误（之前被全局禁用掩盖）。
- 影响：用户影响：使用 TRTLLM rollout 且启用 FP8 量化的用户，async RL 场景下的 FP8 加载逻辑得到修复；GB200 多节点 async RL 用户需要添加环境变量才能避免挂起；sync RL 用户不再受早期强制禁用的影响。系统影响：改动集中在 rollout worker 和 trainer 配置层，范围有限。团队影响：明确了对 GB200 平台的环境变量控制策略，为后续类似问题提供参考。
- 风险标记：多节点同步 RL 挂起修复，FP8 配置路径变更，环境变量控制策略，回归风险

关联脉络

- 暂无明显关联 PR