PR #2220 完整报告

radixark/miles

Add the GLM-5.2 744B x terminal-bench-2 Daytona example

合并时间：2026-08-10 14:51

原文链接：http://prhub.com.cn/radixark/miles/pull/2220

执行摘要

- 一句话：新增 GLM-5.2 × tbench2 Daytona 16 节点参考 RL 示例
- 推荐动作：值得精读。重点关注三处：① PR body 的调优因果链（每项优化如何降低显存或提升并发重叠）；② recipe 的“配置即代码”组织方式（默认值 = 参考配置 + CLI 偏差）；③ 引擎 balanced/low-latency 双形态与 dp-aware 路由的耦合设计。若团队要在 GB300 上跑大规模 agentic RL，这份示例是最新起点；复用时注意核心参数与镜像日期的版本对齐，并考虑把新 launch 脚本接入快照 harness。

功能与动机

PR body 说明团队最初无法在 1 个 GB300 机架内训练 GLM-5.2 744B，经过磁盘 offload（#1575）、NVME 流式优化器（#1793）、显存 / 内存优化与 SGLang 引擎调优后才达成 8+8 节点的可训练形态；再经 sample 级异步提交（#1673/#1677）、路由策略修复（#1657/#1351/#1690）才让 rollout 与训练几乎完全重叠。该 PR 把这些优化组合成一份默认值即参考配置的 recipe，使 16 节点 tbench2 运行可一键复现；README 则固化环境契约与调优依据，降低同类任务的复现与排障成本。

实现拆解

1. Recipe 主文件：新增 examples/experimental/openenv/glm52_tbench2/run_glm5_2_744b_a40b_daytona.py（355 行）。ScriptArgs 继承 U.ExecuteTrainConfig，dataclass 默认值即参考配置；__post_init__ 校验节点数下限（普通运行 10 节点、debug replay 8 节点）、每节点 4 卡、DAYTONA_API_KEY 存在，并补默认的 train/eval 数据路径。_execute_train 把参数按职责分成 checkpoint、rollout、agent、perf、GRPO、optimizer、SGLang 引擎、dashboard、misc 共 9 组，拼成 train 命令；_assert_openenv_deps 在启动时对 openenv/tbench2_env/daytona/fastmcp 做快速失败检查，避免 20 分钟后在 Ray actor 内才暴露缺包。
2. 引擎双模式与路由协同：sglang_config 提供 balanced（每节点一台 4 卡引擎，dp-attention dp=4 + deepep + EAGLE 1/1/2，--sglang-mem-fraction-static 0.85）与 low-latency（每节点对一台 TP8 引擎，EAGLE 5/1/6）两种形态；balanced 分支显式打开 --sglang-enable-dp-attention --sglang-dp-size 4，依赖 dp-aware 路由让 min_load 能看到 dp rank，否则请求会堆在 SGLang 内部选中的单个 rank 上。
3. 站点适配脚本：launch_16node_slurm.sh（59 行）只做容器镜像 + Ray 集群拉起：head 节点先 ray start --head 并轮询等待全部 64 卡加入，worker 节点用 until ray start

--block 循环重试加入（head 容器解包时间不稳定）；FABRIC_PREFIX 用于挑选计算网 IP，DAYTONA_ENV_FILE 从 git 外注入凭证，实验参数全部透传给 recipe。

4. 文档与环境契约：README.md（137 行）记录参考运行结果（100 rollout 步 21 h、prefix-cache 命中率 ≈0.96、并发 90–100）、容器镜像最低日期（sglang-miles 2026-08-04）、Megatron checkpoint hooks（radixark/Megatron-LM#63）、tbench2 环境需 editable 安装的原因、健康信号与调试手段（--debug-replay-data 训练侧重放、--load-from 评估既有 checkpoint）。
5. 测试与配套：本次没有新增测试文件，也未进入 CI 快照体系；提交历史中有一个 [DO NOT MERGE] CI vehicle for Megatron-LM#62 的临时载体提交，最终合并净变更为 3 个文件，未见该载体残留，建议仓库内复核该提交的去留。

关键文件：

- examples/experimental/openenv/glm52_tbench2/run_glm5_2_744b_a40b_daytona.py（模块 训练配方；类别 source；类型 core-logic；符号 ScriptArgs, post_init, _assert_openenv_deps, _execute_train）：整个参考运行的核心载体：ScriptArgs 默认值即参考配置，_execute_train 把训练 /rollout/agent/ 引擎参数组织为 train 命令，内置快速失败校验与调试模式。
- examples/experimental/openenv/glm52_tbench2/launch_16node_slurm.sh（模块 部署脚本；类别 infra；类型 entrypoint）：站点适配器，只负责容器 + Ray 拉起并透传 CLI 参数；head/worker 启动与重试逻辑是跨站点复现的关键差异点。
- examples/experimental/openenv/glm52_tbench2/README.md（模块 文档；类别 docs；类型 documentation）：固化了环境契约（镜像日期、Megatron hooks、Daytona 凭证管理）、参考运行结果基线、健康信号与调试手段，是复现与排障的入口文档。

关键符号：ScriptArgs.post_init, _assert_openenv_deps, _execute_train, _cli, train

关键源码片段

examples/experimental/openenv/glm52_tbench2/run_glm5_2_744b_a40b_daytona.py

整个参考运行的核心载体：ScriptArgs 默认值即参考配置，_execute_train 把训练 /rollout/agent/ 引擎参数组织为 train 命令，内置快速失败校验与调试模式。

```
@dataclass
class ScriptArgs(U.ExecuteTrainConfig):
    """GLM-5.2 744B-A40B 的参考运行配置。

    “默认值即参考配置”：直接用 `python3 run_glm5_2_744b_a40b_daytona.py
    train --num-nodes 16` 就能复现 16 节点全异步 agentic RL 运行；
    任何实验偏差（smoke 规模、引擎形态、checkpoint 评分）都通过
    CLI 参数表达而不是改脚本，保证参考配置始终可审计。
    """

    mode: Literal["normal"] = "normal"
    run_id: str = U.create_run_id()
    model_name: str = "GLM-5.2"
```

```
    megatron_model_type: str = "glm5.2-744B-A40B"
    num_gpus_per_node: int = 4
    fp8_rollout: bool = True

    # 训练 batch 与 rollout batch 解耦：rollout 侧并发在途轨迹数由
    # async_max_concurrent_samples 决定，同时它也是 Daytona 沙箱池大小，
    # 保证每条在途轨迹都有一台专属沙箱可用。
    rollout_batch_size: int = 8
    n_samples_per_prompt: int = 8
    global_batch_size: int = 64
    rollout_max_response_len: int = 16384
    async_max_concurrent_samples: int = 128

    # DP1 下优化器状态不分片（约 279 GB/rank），在 276 GB 显存的
    # GB300 上必须流式落盘——这是刚需而不是优化；置空可关闭，
    # 仅用于更大 DP 的 smoke 运行。
    offload_train_disk_dir: str = "/scratch/opt_state"
    save_interval: int = 100000 # 实际只在训练结束时保存一次

    # OpenEnv / Daytona 相关；凭证与任务目录从环境变量读取，
    # 不写死在仓库里。
    agent_model_name: str = os.environ.get("AGENT_MODEL_NAME", "model")
    openenv_max_turns: int = int(os.environ.get("OPENENV_MAX_TURNS", "30"))
    openenv_tb2_tasks_dir: str = os.environ.get("OPENENV_TB2_TASKS_DIR", "")

    # eval 复用 rollout 引擎（producer 暂停）；None 表示关闭。
    # 训练 / 推理 8+8 拆分没有多余的 GPU 跑独立 eval fleet。
    eval_interval: int | None = 10
    n_samples_per_eval_prompt: int = 2
    daytona_api_key: str = os.environ.get("DAYTONA_API_KEY", "")

    # 训练侧重放已落盘的 rollout 数据：不需要引擎和沙箱，
    # 8 个训练节点即可，用于快速验证并行度 /OOM 类改动。
    debug_replay_data: str = ""

    def __post_init__(self):
        # debug replay 不需要推理节点，所以下限是 8；否则至少 10 节点
        min_nodes = 8 if self.debug_replay_data else 10
        assert (
            self.num_nodes >= min_nodes and self.num_gpus_per_node == 4
        ), "GB300 配置：8 个训练节点加推理节点（每节点 4 卡）"
        assert self.daytona_api_key, "DAYTONA_API_KEY 必须设置在环境中"
        if not self.prompt_data:
            self.prompt_data = f"{self.data_dir}/tbench2_train.jsonl"
        if self.eval_interval is not None and not self.eval_prompt_data:
            self.eval_prompt_data = f"{self.data_dir}/tbench2_eval.jsonl"


def _assert_openenv_deps():
```

```
    """启动时快速失败，而不是 20 分钟后在 Ray actor 内部才失败。"""
    for mod in ("openenv", "tbench2_env", "daytona", "fastmcp"):
        assert (
            importlib.util.find_spec(mod) is not None
        ), f"{mod} 未安装在这个容器里；请查看 README.md 前置要求"
```

examples/experimental/openenv/glm52_tbench2/launch_16node_slurm.sh

站点适配器，只负责容器 + Ray 拉起并透传 CLI 参数；head/worker 启动与重试逻辑是跨站点复现的关键差异点。

```
# 站点适配器：只做容器 + Ray 拉起，实验设置全部在 recipe 里。
# 脚本自己的 CLI 参数原样透传给 recipe，例如：
# sbatch --export=ALL launch_16node_slurm.sh [--num-rollout 10 ...]
# 必须从外部注入的环境：MILES_ROOT / CONTAINER_IMAGE / CONTAINER_MOUNTS /
# DAYTONA_ENV_FILE（chmod 600，凭证不进 git）

# 计算网 IP：hostname -I 的顺序不稳定且管理网段不可路由，
# 所以按 FABRIC_PREFIX 显式挑选计算网地址。
head_ip=$(srun --nodes=1 --ntasks=1 -w "${nodes[0]}" hostname -I | tr ' ' '\n' | grep -E
"^${FABRIC_PREFIX:-10.}" | head -1)
ngpu_total=$(( ${#nodes[@]} * 4 ))

# head 节点：启动 ray head 后轮询等待全部 64 卡加入，再拉起 recipe；
# srun 用 --overlap 让 ssh 会话与计算共享同一节点资源。
srun --overlap --nodes=1 --ntasks=1 --gpus-per-node=4 -w "${nodes[0]}" $C bash -c "
    $COMMON
    ray start --head --node-ip-address=$head_ip --port=6379 --num-gpus=4 --disable-usage-stats
    for t in \$(seq 1 90); do
        ray status 2>/dev/null | grep -q '/$ngpu_total\.0 GPU' && break
        echo \"waiting for ray nodes... (\$t/90)\"; sleep 10
    done
    source $DAYTONA_ENV_FILE
    export MILES_SCRIPT_EXTERNAL_RAY=1 MASTER_ADDR=$head_ip OPENENV_RUN_ID=
    \${OPENENV_RUN_ID:-$SLURM_JOB_ID}
    cd $MILES_ROOT
    python3 $RECIPE train --num-nodes $SLURM_JOB_NUM_NODES $RECIPE_ARGS
" &
HEAD_PID=$!

# worker 节点：循环重试 ray join——head 容器解包耗时不定，
# 单次 ray start 可能在 head GCS 就绪前超时。
sleep 30
for ((i=1; i<${#nodes[@]}; i++)); do
    srun --overlap --nodes=1 --ntasks=1 --gpus-per-node=4 -w "${nodes[$i]}" $C bash -c "
        $COMMON
        until ray start --address=$head_ip:6379 --num-gpus=4 --disable-usage-stats --block; do
            echo 'worker retry ray join...'; sleep 10
        done
    " &
```

```
done

wait $HEAD_PID
echo "=== driver exited; stopping job ==="
scancel "$SLURM_JOB_ID"
```

评论区精华

唯一的 review 评论来自 Shi-Dong，针对 README 早期版本的镜像要求段落：『Both SGLang PRs are merged, so I think we can drop this now?』——README 当时用较长篇幅列出两个 SGLang 修复前置条件，评论认为既然修复已合入 sglang-miles，相关说明可以精简。作者随后的两个提交（'README: sglang fixes are on sglang-miles now; keep a minimum-date line'、'README: one line is enough for the image requirement'）把该段压缩成一行并保留最低日期线，回应了这条意见。另有 3 条 APPROVED（Shi-Dong、Zhichenzzz、yushengsu-thu）。提交历史中的 [DO NOT MERGE] 临时 CI 载体提交是值得注意的旁支：它用于在 CI 上验证 Megatron-LM#62 的磁盘 checkpoint 改动，最终合并内容不含该载体。

- README 中 SGLang 修复说明是否可删除 (documentation): 作者后续通过提交 'README: sglang fixes are on sglang-miles now; keep a minimum-date line' 与 'README: one line is enough for the image requirement' 将镜像要求压缩为一行并保留最低日期线（2026-08-04），回应了该评论。

风险与影响

- 风险：
  1. 示例参数与核心演进耦合：recipe 大量引用近期核心参数（--stream-optimizer-state-to-disk、--rollout-submission-granularity sample、--sglang-enable-dp-attention、--use-session-server 等），核心库后续调整参数名或语义时这份示例会静默过期（仓库已有 #2269 清理废弃参数残留的前例，快照体系未覆盖本文件意味着不会有自动告警）。
  2. 外部依赖版本未固化：README 要求 sglang-miles 镜像不早于 2026-08-04 且依赖 Megatron-LM#63 的 checkpoint hooks，复现环境若拉取更早镜像会直接失败或行为不一致；--sglang-tool-call-parser glm47 等参数也依赖特定 SGLang 版本。
  3. 站点相关硬编码：/root/models、/root/datasets、/scratch/opt_state、FABRIC_PREFIX 默认 10.、--time=48:00:00 等都是作者站点假设，其他站点复现需显式覆盖，否则容易在数据路径或网络选路上踩坑。
  4. 无自动化测试覆盖：新 recipe 与 launch 脚本没有进入 fast 测试或快照体系；.sh 脚本未被 harness 捕获，未来 shell 基建重构（如 #1903 exec_command 重命名、#1904 helper 迁移）无法自动发现引用破损。 - 影响：对复现者：获得一份开箱即用的 16 节点 GLM-5.2 × tbench2 参考运行，README 给出环境契约与预期基线（step time、截断率、prefix-cache 命中率、沙箱失败容忍度），便于快速判断自己的运行是否健康。对系统：全部变更位于 examples/experimental，不触碰核心库；但 recipe 是 fully-async rollout、磁盘 offload、dp-aware 路由、dashboard 观测等核心能力的集成

范例，会作为后续大规模 agentic RL 的起点被复制演进。对团队：PR body 的性能调优叙事把零散优化串成因果链（从单机架装不下到 8+8 完全重叠），是 onboarding 与排障的宝贵资料。影响程度评估：功能影响面限定在示例目录，中低；知识价值高。 - 风险标记：示例参数与核心演进耦合，外部依赖版本未固化，站点路径硬编码，无自动化测试覆盖

关联脉络

- PR #2274 refactor(openenv): move duplicated sandbox helpers into the TB2 recipe: 同属 examples/experimental/openenv 目录，将沙箱辅助函数上收共享 recipe；本 PR 的 agent 函数与 reward 模块直接依赖这些共享模块。
- PR #2237 fix(openenv): build E2B task templates as root: 同为 openenv 沙箱后端基建的修复，与本 PR 的 Daytona 沙箱路径并列演进。
- PR #2030 [async] async data buffer: unified filters and better observability: fully-async 数据缓冲与可观测性重构，本 PR 的参考配置依赖 FullyAsyncRolloutFn 与 sample 级提交语义。
- PR #1657 [fix] fix session server hash_consistent mode: PR body 明确引用：session server 路由修复是本 PR 引擎负载均衡的前置条件，缺失时请求会钉死在单台引擎。
- PR #1690 [router] set manual policy (sticky + min_load) as default agentic routing policy: PR body 明确引用：hash-consistent + min_load 路由策略修复了引擎间不均衡导致的训练气泡。