PR #2001 完整报告

THUDM/slime

[docs] Add step-by-step debug tutorial

合并时间：2026-06-01 16:43

原文链接：http://prhub.com.cn/THUDM/slime/pull/2001

执行摘要

本 PR 为开发者指南（debug.md）新增了使用 Ray Distributed Debugger 进行交互式单步调试的教程，同步更新中英文文档，填补了调试文档中缺失的重要环节。

功能与动机

当前调试指南覆盖了精度对齐、训练推理分离调试和 IMA 故障排除，但缺少交互式单步调试的方法。在开发自定义 rollout 函数或调查训练行为时，单步调试是开发者的常见需求。

实现拆解

在 docs/en/developer_guide/debug.md 和 docs/zh/developer_guide/debug.md 文件末尾分别新增了“Step-by-Step Debugging with Ray Distributed Debugger”和“使用 Ray Distributed Debugger 单步调试”章节，包含 4 个详细步骤：

1. 安装 debugpy 1.8.0。
2. 在启动脚本中设置环境变量 RAY_DEBUG_POSTMORTEM=1，并将其传入 RUNTIME_ENV_JSON。
3. 在训练入口 train.py 的 breakpoint() 之前调用 ray.init()，确保分布式调试器的依赖 core_worker 可用。
4. 在 VS Code 中安装 Ray Distributed Debugger 扩展，提交作业命中断点后，通过 Ray Dashboard 面板 attach 调试器进行交互式调试。

文档同时包含注意事项：调试后务必移除调试代码，且不带参数的 ray.init() 在多节点训练中可能引发问题。

关键源码片段

以下展示了教程中推荐添加的调试用代码片段（来自文档）：

```
train.py 入口修改示例
if __name__ == "__main__":
    ray.init() # 必须先初始化 Ray，否则 breakpoint() 会因缺少 core_worker 而报错
    breakpoint() # 设置断点，等待分布式调试器 attach
    args = parse_args()
    train(args)
```

以及启动脚本中的环境变量注入：

```
在启动脚本中启用 Ray 分布式调试器
export RAY_DEBUG_POSTMORTEM=1

RUNTIME_ENV_JSON=\"{
  \"env_vars\": {
    \"RAY_DEBUG_POSTMORTEM\": \"${RAY_DEBUG_POSTMORTEM:-0}\"
  }
}\"

ray job submit --address=\"http://127.0.0.1:8265\" \
  --runtime-env-json=\"${RUNTIME_ENV_JSON}\" \
  -- python3 train.py [args...]
```

评论区精华

合并者zhuzilin在评论中称赞："Thisisamazing!Thankyousomuch!"说明文档质量被认可。

风险与影响

风险极低，仅为文档补充。用户若未按注意项移除调试代码可能影响训练，但文档已明确提醒。影响范围限于阅读开发者指南的用户，提升了分布式调试的易用性。

关联脉络

该 PR 是开发者指南文档系列改进的一部分。近期 PR #1988 大规模重写了高级文档，PR #1989 和 #1986 也涉及文档优化，本 PR 进一步细化了调试说明，形成了从 CI、高级配置到调试方法的完整文档体系。