PR #7627 完整报告

verl-project/verl

[misc] feat: uv support aarch64

合并时间：2026-08-31 14:13

原文链接：http://prhub.com.cn/verl-project/verl/pull/7627

执行摘要

- 一句话：uv 锁与镜像支持 aarch64 双架构，配套流程重写
- 推荐动作：值得精读。该 PR 展示了“架构作为解析维度而非 extra 维度”的 uv 多架构设计，以及两个非常具体的坑：nvidia-cusparselt-cu13 的 wheel 内部 tag 不一致导致 uv 反复重装、megatron-core VCS 版本号导致锁文件与实际构建不一致。这些细节对任何维护 uv 单锁多平台的团队都有借鉴价值；Docker 多架构构建与 manifest 合并的实操说明也可直接复用。重点关注 pyproject.toml 的 [tool.uv] 配置注释和 docker/Dockerfile.uv.cu130 的构建说明。

功能与动机

PR body 只有一句目标：UV support aarch64 with the same image: verlai/verl:uv.cu130.dev1。核心诉求是让同一份 uv.lock、同一个 Docker 镜像在 x86_64 与 aarch64（GH200 / GB200 超级芯片）两类 GPU 主机上都能直接运行，而不是为 aarch64 单独维护一套依赖锁或另一条安装路径。pyproject.toml 的注释进一步说明可行性：torch / torchvision / torchaudio cu130、vllm / sglang 及 wheelhouse 的 apex / flash-attn / transformer-engine 均已有 aarch64 wheel，因此架构可以作为解析维度融入现有通用锁。

实现拆解

1. 扩展锁文件的架构覆盖：在 pyproject.toml 的 [tool.uv].environments 中追加 platform_machine == 'aarch64' marker，使 uv lock 同时解析 x86_64 与 arm64 两个环境的依赖，uv.lock 因此新增约 1600 行 aarch64 解析条目。设计上明确“架构是 resolution 维度，不是 extra 维度”，保持 extra 名称、冲突集合在所有架构上一致，避免拆出 megatron-aarch64 之类的重复 extra。
2. 修复 aarch64 特有依赖问题：override-dependencies 中仅对 aarch64 追加 nvidia-cusparselt-cu13==0.9.1（0.8.0 / 0.8.1 的 wheel 内部 tag 为 manylinux2014_sbsa，uv 认为与文件名不符，每次 uv run 都重装，100+ 并发 Ray worker 会竞争删除 libcusparseLt.so.0 导致 torch import 崩溃）；同时在 docker/Dockerfile.uv.cu130 设置 ENV NO_VCS_VERSION=1，避免 megatron-core 的 __version__ 带上 git hash 导致与锁文件 0.18.0 不一致而反复重装。
3. 环境驱动工具感知双架构：manage_envs.py 新增 SUPPORTED_ARCHES = ('x86_64', 'aarch64') 常量，cmd_list 通过 platform.system() 与 platform.machine() 报告当前主机是否被 uv.lock 覆盖，让不被支持的平台在 sync 前就得到明确报错，而不是 uv 内部的 no

solution found。

4. Docker 多架构构建支持：Dockerfile.uv.cu130 新增 ARG CUDA_BASE_IMAGE 允许替换 base 镜像仓库，并补充基于 buildx 远程 arm64 原生节点、QEMU 模拟、多架构 manifest 合并（imagetools create）的构建说明；docker/README.md 给出完整操作指南与排错细节。
5. 示例脚本与文档同步：多个 examples/grpo_trainer 脚本（如 run_qwen3_5_2b_openr1_fsdp.sh、run_deepseek_v4_flash_megatron.sh）改为 LAUNCH / RAY 数组结构，在 VERL_USE_UV + DEVICE=gpu 分支下用 uv run --frozen --all-packages --extra <engine> --extra <trainer> 启动，并把 Ray 的 runtime_env.py_executable 也指向 uv；docs/start/install.rst 重写为 uv 主线安装文档；tests/special_sanity/check_uv_gpu_only.py 同步更新错误提示与说明。

关键文件：

- pyproject.toml（模块 依赖配置；类别 config；类型 configuration；符号 [tool.uv].environments, [tool.uv].override-dependencies）：核心配置变更：在 [tool.uv].environments 增加 aarch64 marker，按架构 pin nvidia-cusparselt-cu13，并说明'架构是解析维度而非 extra 维度'的设计原则。
- manage_envs.py（模块 环境驱动；类别 source；类型 dependency-wiring；符号 SUPPORTED_ARCHES, cmd_list, _build_parser）：uv 环境驱动脚本，新增 SUPPORTED_ARCHES 与 cmd_list 主机架构报告，让不支持平台在 sync 前就能得到明确报错；同时更新模块文档说明双架构语义。
- docker/Dockerfile.uv.cu130（模块 镜像构建；类别 infra；类型 infrastructure；符号 NO_VCS_VERSION, CUDA_BASE_IMAGE）：多架构构建支持：CUDA base 镜像可被 CUDA_BASE_IMAGE 替换、增加 NO_VCS_VERSION 修复 megatron-core 反复重装、文档化 buildx 远程 / 模拟构建流程。
- uv.lock（模块 锁文件；类别 other；类型 core-logic）：重新生成后的通用锁文件，同时包含 x86_64 与 aarch64 的解析结果，是本 PR 的目标产物，也是改动量最大的文件。
- docs/start/install.rst（模块 安装文档；类别 docs；类型 documentation）：安装文档主流程重写为 uv run / uv.lock 工作流，明确 x86_64 与 aarch64 双架构支持、Ray py_executable 传递与 VERL_USE_UV 回退开关。
- docker/README.md（模块 镜像文档；类别 docs；类型 documentation）：补充 GB200 / aarch64 镜像构建、buildx 远程节点、QEMU 模拟、代理设置与多架构 tag 合并的操作指南。
- examples/grpo_trainer/run_qwen3_5_2b_openr1_fsdp.sh（模块 示例脚本；类别 other；类型 core-logic；符号 LAUNCH, RAY）：示例脚本代表：从直接 python3 改为 GPU 分支下 uv run + Ray py_executable 传递启动，是 uv launch 流程在实际训练入口的落地。同类改动的脚本还有 run_deepseek_v4_flash_megatron.sh、run_qwen3_5_2b_video_fsdp.sh、run_qwen3_5_397b_megatron.sh。
- tests/special_sanity/check_uv_gpu_only.py（模块 静态检查；类别 test；类型 test-coverage）：同步 sanity 检查：更新文档字符串与错误信息，确认 lock 覆盖双架构，并强调 gate 基于 DEVICE 而非 arch。

关键符号：cmd_list, SUPPORTED_ARCHES, _build_parser

关键源码片段

pyproject.toml

核心配置变更：在 [tool.uv].environments 增加 aarch64 marker，按架构 pin nvidia-cusparselt-cu13，并说明'架构是解析维度而非 extra 维度'的设计原则。

```
# 一个通用 uv.lock 覆盖所有后端：vllm / sglang / fsdp / megatron / cpu。
# 架构是 resolution 维度，不是 extra 维度：
# 在 [tool.uv].environments 中为 x86_64 与 aarch64（GH200 / GB200）各声明一个 marker，
# uv 按 marker 分叉解析，再按主机平台 tag 选取对应 wheel。
# 因此 extra 名称、冲突集合、manage_envs.py 组合在两种架构上完全一致。
[tool.uv]
environments = [
    "python_full_version >= '3.12' and sys_platform == 'linux' and platform_machine == 'x86_
    64'",
    "python_full_version >= '3.12' and sys_platform == 'linux' and platform_machine ==
    'aarch64'",
]

override-dependencies = [
    # aarch64 专用 pin：nvidia-cusparselt-cu13 0.8.0 / 0.8.1 的 wheel 内部 tag 为
    # manylinux2014_sbsa，与文件名 aarch64 不一致，uv 会每次 uv run 重装，
    # 100+ 个并发 Ray worker 竞争删除 libcusparseLt.so.0 导致 torch import 崩溃
    #  （astral-sh/uv#17711）。0.9.1 修正内部 tag 为 manylinux2014_aarch64，
    # 并把库移至 nvidia/cu13/lib/，torch 2.11 的 _preload_cuda_deps 已会搜索该路径。
    # x86_64 保持 0.8.0 不变，避免扰动已验证的配置。
    "nvidia-cusparselt-cu13==0.8.0 ; platform_machine == 'x86_64' and sys_platform == 'linux'",
    "nvidia-cusparselt-cu13==0.9.1 ; platform_machine == 'aarch64' and sys_platform == 'linux'",
]
```

manage_envs.py

uv 环境驱动脚本，新增 SUPPORTED_ARCHES 与 cmd_list 主机架构报告，让不支持平台在 sync 前就能得到明确报错；同时更新模块文档说明双架构语义。

```
# 支持的 CPU 架构。架构不是 extra 维度：两个架构共享同一组 extra 名与冲突集合，
# uv sync 按主机平台 tag 选 wheel。若主机架构未被 uv.lock 覆盖，
# sync 会在 uv 内部报 no solution found，因此 cmd_list 会提前报告 host 是否受支持。
SUPPORTED_ARCHES: tuple[str, ...] = ("x86_64", "aarch64")


def cmd_list(args: argparse.Namespace) -> int:
    """列出可用 extra、冲突规则、venv 状态，并报告主机是否被 uv.lock 覆盖。"""
    # 同时检查操作系统与 CPU 架构，让 macOS arm64（报告 arm64 而不是 aarch64）
    # 等不受支持平台在 sync 前得到明确报错。
    host = f"{platform.system()} {platform.machine()}"
    locked = sys.platform == "linux" and platform.machine() in SUPPORTED_ARCHES
    print(
```

```
            f"\nhost: {host} "
            + (
                "(covered by uv.lock; the extras above resolve here)"
                if locked
                else f"— NOT covered by uv.lock (Linux {' / '.join(SUPPORTED_ARCHES)} only), so
                `sync` will fail"
            )
        )
    # 继续打印冲突集合与 venv 状态（省略后续输出逻辑）……
```

评论区精华

该 PR 没有任何 review 评论，仅有一条来自 wuxibin89 的空 body APPROVED，因此没有可引用的争议讨论。真正的技术权衡体现在 7 个 commit 的迭代轨迹中：try aarch64 → try fix frontend building → try fix uv ray → try fix uv ray with path → fix nv cusparselt，说明 Ray 通过 uv 启动的路径解析与 aarch64 的 cusparselt wheel tag 是主要调试难点；最终方案是让 Ray 的 py_executable 直接指向 uv run，而不是为 Ray 单独维护一套环境。

- 无 review 评论 (other): 最终方案：Ray py_executable 直接指向 uv run；cusparselt 按架构 pin 0.9.1。无未解决问题。

风险与影响

- 风险：
  1. 锁文件回归风险：uv.lock 重生成后，x86_64 的解析条目也可能发生重排（+1603 / -1089 中不全是新增 aarch64 条目），若某些间接依赖版本被 uv 重新选择，可能影响现有 CI 组合，需要全量 CI 验证。
  2. NO_VCS_VERSION 全局生效：该 ENV 设置在 Docker 镜像内对所有构建生效，若未来有代码依赖 megatron-core 的 0.18.0+hash 版本字符串（如调试信息、版本检查），会与预期不一致。
  3. 示例脚本默认走 uv：VERL_USE_UV 默认值为 1，无 uv 环境且未显式设置 VERL_USE_UV=0 的用户运行示例脚本会失败；文档虽提供了回退开关，但存在用户落地时的摩擦。
  4. 多架构镜像发布复杂度：文档建议的多架构 manifest 合并依赖 buildx imagetools create，若只推送单架构 tag，下游 FROM verlai/verl:uv.cu130 会拉取错误平台镜像；远程 buildx 节点还要求 SSH 免密与 docker 组权限，属于新增的运维面。
  5. aarch64 侧缺少专项测试：sanity 检查只更新了文案，未新增真实 aarch64 环境下的 CI 用例，wheelhouse 是否已发布 arm64 构建也未在 PR 中给出验证结果（存在不确定性）。 - 影响：对用户而言，GH200 / GB200（aarch64）用户可以直接复用同一份 uv.lock 与 verlai/verl:uv.cu130.dev1 镜像，不再需要等待单独的 aarch64 镜像；x86_64 用户的使用方式不变，但安装文档主流程从 docker 镜像切换为 uv run，需要适应新命令。对系统而言，Docker 镜像发布需要新增多架构 tag 合并步骤，CI 与示例脚本的启动路径整体迁移到 uv run + Ray py_executable。对团队而言，环境维护收敛到单锁文件 + 双架构解析，后续引入 arch-specific 包时需要遵循 pyproject.toml 中定义的 marker 拆分约定。整体影响范围较大，但属于基础设施与文档层面，不涉及训练核心

逻辑。 - 风险标记：双架构锁文件解析范围扩大，示例默认 uv 启动影响无 uv 用户，NO_VCS_VERSION 全局生效，多架构镜像发布易错，缺少 aarch64 专项测试

关联脉络

- PR #7539 [ray] fix: skip unused TensorDict consolidation in NumPy DataProto serialization: 本 PR 让 Ray 的 runtime_env.py_executable 直接走 uv run，Ray 序列化路径的稳定性修复与 uv 启动后的 Ray 行为同属一条技术线。
- PR #7508 [vllm] fix: honor explicit False on Optional[bool] engine args in CLI serialization: vLLM 的 CLI 参数经 uv run 传递，该修复保证 uv 启动路径下显式 False 参数不被序列化逻辑吞掉，与本 PR 的 uv run 启动链路直接相关。
- PR #7632 [vllm] fix: raise max_num_batched_tokens to max_model_len when chunked prefill is disabled: 同期对示例脚本与 CI 的 vLLM 启动参数修复，与 uv run 启动方式在同一批脚本中落地，两者互相影响。
- PR #7629 [ci] chore: fix ci failure: CI 修复涉及 E2E 脚本的运行参数，本 PR 将示例脚本切换到 uv run 后 CI 行为也随之改变，二者需要保持一致。
- PR #7461 [training_utils, env, doc] feat: use Liger fused linear PPO kernel: 新增依赖并调整安装流程，与 uv 环境管理同属环境演进线，后续也需要在双架构锁下验证。