PR #2714 完整报告

radixark/miles

Bump sglang to v0.5.18

合并时间：2026-08-27 16:08

原文链接：http://prhub.com.cn/radixark/miles/pull/2714

执行摘要

- 一句话：sglang 升级 v0.5.18 并适配 torch 2.13
- 推荐动作：值得精读。重点有三个：一是 PR body 本身就是依赖升级工程的范本——回滚点定义、中转基线选择（736 vs 1317 drift）、冲突取舍原则（上游新形态为准、行为重表达）都写得很清楚；二是 _forward_remaining_collectives() 的自动转发设计，用元编程根除手工维护清单的复发式 bug，值得在同类 wrapper 场景复用；三是 det PG 覆盖范围先放大再收敛的过程，展示了如何靠多 GPU 测试逐步逼近最小必要改动。若团队后续还要升级 sglang/torch，建议直接对照此 PR 的适配清单。

功能与动机

PR body 明确说明动机：Rebases the sglang-miles stack onto v0.5.18 and repoints the miles images at it. Targets #2673 so the Megatron and sglang bumps are validated together。选 v0.5.17 而非生产 v0.5.16 作为转发基点，是因为 going through v0.5.17 halves the upstream drift (736 commits instead of 1317) and carries forward resolutions a green CI already validated。镜像侧换 wheel 集是被 torch 2.13 强制推动的：sglang v0.5.18 ships torch 2.13, which dropped c10::impl::cow::materialize_cow_storage，旧 cu130 wheel（torch 2.11 构建）引用该符号会全部失效。同时 v0.5.18 基镜像自带 flashinfer 0.6.17 与 cutlass-dsl 4.6.2 修复，miles 此前针对 Blackwell 的手工 pin 反而变成降级，应当移除。

实现拆解

变更入口是 sglang-miles fork 分支与 docker/Dockerfile 的版本号切换，随后是仓库侧针对 torch 2.13 与 sglang v0.5.18 破坏性变更的兼容层适配。

1. sglang-miles fork 跨版本 rebase：在 v0.5.18 上从 sglang-miles-v0.5.17 重放 35 个 commit（避开生产 v0.5.16，把上游 drift 从 1317 压到 736），其中 9 个冲突。对上游已删除的机制（tokenizer_manager 的并行采样 child_rid_to_logical_rid / lifecycle_id / _remove_req_state、forward_batch_info 的 mrope 拆分与内建 rl_on_policy_target 强制 text-only 路径）在新形态上重表达 miles 行为，保留 abort-by-prefix gate、pre-dispatch flagging、LoRA lease release，以及在 true_on_policy_contract 双决策点插入合约切换；deepseek_v2 的共享专家融合决策上移到 loader 与 shared_experts_fusion_disable_reason 类方法承载 SBO/TBO/DeepEP 原因；compile_utils 保留上游 SGLANG_DG_CACHE_DIR 默认值并叠加

SGLANG_DG_CACHE_DIR_PER_PROCESS 后缀，避免同机 RL engine 争用 JIT cache；修复 pick 掉落的 _post_load_weights 与 glm4_moe_nextn.get_exec，ruff F821 清零。

2. 镜像与 wheel 集迁移（docker/Dockerfile、docker/build.py）：SGLANG_IMAGE_TAG 从 v0.5.16 切到 v0.5.18，release 变体钉 v0.5.18-cu129；x86 与 aarch64 的 WHEELS_TAG_* 默认值切到 cu130-torch213-* 系列，因为 torch 2.13 移除 materialize_cow_storage 符号后旧 wheel 全部无法加载，其中 aarch64 默认值曾漏改，由 Zhichenzzz 补正；删除被 v0.5.18 基镜像取代的 flashinfer 0.6.15.post1 与 cutlass-dsl 4.6.2 pin（旧 pin 在 Blackwell 上反而回退）；cu12 pyproject.toml 重写块的三个 sed 模式经核对仍适用；合并后主分支切回稳定的 sglang-miles 名称。
3. torch 2.13 集体通信兼容层（miles/utils/reloadable_process_group.py、miles/utils/test_utils/det_process_group.py）：前者新增模块级 _forward_remaining_collectives()，遍历 dir(dist.ProcessGroup) 为所有未覆盖的可调用方法动态生成 _fwd 透传，根治手工转发名单随 torch 改名 / 新增而反复漏项的问题；后者为 torch 2.13 的 reduce_scatter_single / reduce_scatter_single_coalesced / all_to_all_single 补上确定性入口，防止落到 C++ 基类后经 NCCL 静默执行、丢失固定序 fold 或直接 SIGSEGV，并修正 coalescing manager flush 时内层调用返回 None 的等待逻辑。
4. rollout 与可观测性适配（miles/ray/rollout/sglang_server_actor.py、miles/backends/sglang_utils/sglang_engine.py、miles/utils/replay_base.py）：launch_engine 在 v0.5.18 返回六元组（新增 _weight_cache_daemon_procs），RDT server actor 的五元解包改为六元；ServerArgs 在声明物化后只读，IPv6 host 去方括号统一前移到构造处；R3 replay check 由只在失败时输出改为每次检查都记录 mismatch/token 数与阈值，便于跨 run 对比噪声底。
5. CI 与验证策略（.github/workflows/_build-pr-ci-image.yml、.github/workflows/docker-build.yml）：PR 镜像重建由「是否触碰 docker 路径」改为「构建输入内容 hash 是否变化」，hash 以 label 形式打在发行 tag 上；wheel 指纹同步更新到 cu130-torch213-*，并在构建前回收 builder 磁盘。仓库侧本次未新增测试文件，兼容层正确性由既有 test_det_process_group_multi_gpu（捕获过 all_to_all_single 的 SIGSEGV）与 run-ci-imaged 全量 e2e 兜底；test_mtp1_spec_v2_r3 三跑三结果（NaN 超时 / NCCL init 错误 / 通过）被判定为既有 flaky，不属于本次回归。

关键文件：

- miles/utils/reloadable_process_group.py（模块 进程组；类别 source；类型 core-logic；符号 _forward_remaining_collectives, make, forward）：torch 2.13 新增的 *_single 集体系列入口会落到 C++ 基类空 backend map 的核心修复：模块级 _forward_remaining_collectives() 动态为所有未覆盖的 ProcessGroup 方法生成 _fwd 透传，避免手工清单随 torch 改名再次长洞。
- miles/utils/test_utils/det_process_group.py（模块 确定性测试；类别 source；类型 core-logic；符号 reduce_scatter_single, reduce_scatter_single_coalesced, all_to_all_single, allgather_into_tensor_coalesced）：确定性测试在 torch 2.13 下的正确性关键：覆盖 reduce_scatter_single / reduce_scatter_single_coalesced / all_to_all_single 新入口，防止静默回到 NCCL 丢失固定序 fold；同时修复 coalescing

flush 返回 None 的等待问题。

- miles/backends/sglang_utils/sglang_engine.py（模块 推理引擎；类别 source；类型 core-logic；符号 _init_normal, launch_server_process）：v0.5.18 的 ServerArgs 声明物化后只读，原先两个 launch 站点对 IPv6 host 的 strip 操作会抛异常；改为构造实例时统一去除方括号，并保留 dict 原样供 self.server_host 拼 url。
- miles/utils/replay_base.py（模块 重放校验；类别 source；类型 core-logic；符号 check_replay_result）：R3 mismatch 统计从「仅失败时出现在 AssertionError」改为「每次检查都记录」，让 pass/pass 运行之间也能比较噪声底，是确定性回归排查的基础设施改进。
- docker/Dockerfile（模块 部署镜像；类别 infra；类型 infrastructure）：基础设施核心变更：SGLANG_IMAGE_TAG 切 v0.5.18、wheel 集切 cu130-torch213-*，并删除被基镜像取代的 flashinfer/cutlass-dsl pin，避免在 Blackwell 上把新版本降回去。
- miles/ray/rollout/sglang_server_actor.py（模块 服务编排；类别 source；类型 core-logic；符号 start）：v0.5.18 的 launch_engine 返回值从五元组变六元组（新增 _weight_cache_daemon_procs），旧解包会在 HTTP 线程启动前抛 ValueError，RDT 路径必须同步。
- .github/workflows/_build-pr-ci-image.yml（模块 流水线；类别 infra；类型 infrastructure）：PR 镜像重建策略从「是否触碰 docker 路径」改为「构建输入 hash 是否变化」，避免每次重跑 Dockerfile 相关 PR 都重复多架构构建。
- .github/workflows/docker-build.yml（模块 流水线；类别 infra；类型 infrastructure）：wheel 指纹同步到 cu130-torch213-* 标签，确保重新发布 torch 2.13 wheel 时能触发镜像重建。
- docker/build.py（模块 镜像构建；类别 infra；类型 infrastructure）：release 构建变体钉到 v0.5.18-cu129，与 Dockerfile 的 SGLANG_IMAGE_TAG 保持一致。
- docs/ci/02-docker-build.md（模块 文档；类别 docs；类型 documentation）：同步 docker 构建流程文档，反映镜像重建判定与 wheel 标签变化。
- docs/developer/versions.md（模块 文档；类别 docs；类型 documentation）：版本矩阵更新 sglang v0.5.18 与 torch 2.13 wheel 集，帮助开发者选择正确的基镜像。

关键符号：_forward_remaining_collectives, reduce_scatter_single, reduce_scatter_single_coalesced, all_to_all_single, allgather_into_tensor_coalesced, _init_normal, check_replay_result

关键源码片段

miles/utils/reloadable_process_group.py

torch 2.13 新增的 *_single 集体系列入口会落到 C++ 基类空 backend map 的核心修复：模块级 _forward_remaining_collectives() 动态为所有未覆盖的 ProcessGroup 方法生成 _fwd 透传，避免手工清单随 torch 改名再次长洞。

```
# miles/utils/reloadable_process_group.py
# 自动补齐 ReloadableProcessGroup 未显式定义的 ProcessGroup 集体通信原语。
# 背景：Megatron 在 import 期就把 dist_reduce_scatter_func 绑定到这个包装类，
```

```
# torch 2.13 又新增了 reduce_scatter_single 等 *_single 系列入口；
# 若只靠手工维护转发名单，一旦 torch 重命名或新增集合通信，
# 未覆盖的调用就会落到 C++ 基类（其 backend map 为空），
# 报错 "No backend type associated with device type cuda"。

def _forward_remaining_collectives():
    """Forward every ProcessGroup collective this class does not define itself.

    Callers that resolved a collective before the monkey patch went on --
    Megatron binds `dist_reduce_scatter_func` at import time -- hand the wrapper
    straight to torch, which invokes the method on the group object. Anything
    not overridden here reaches the C++ base, whose own backend map is empty,
    and dies as "No backend type associated with device type cuda". A
    hand-written forward list silently regrows that hole whenever torch renames
    or adds a collective, which is how torch 2.13's *_single family got through.
    """
    # 这些属性按语义应留在包装类自身，不转发给内部 group
    skip = {"rank", "size", "name", "abort", "shutdown", "bound_device_id"}
    for name in dir(dist.ProcessGroup):
        if name.startswith("__") or name in skip:
            continue
        # 类已显式定义的方法保持不动，避免覆盖既有行为
        if name in vars(ReloadableProcessGroup):
            continue
        if not callable(getattr(dist.ProcessGroup, name, None)):
            continue

        # 闭包捕获当前 method 名，统一经 _fwd 透传到内部 group
        def make(method):
            def forward(self, *args, **kwargs):
                return self._fwd(method, *args, **kwargs)

            forward.__name__ = method
            return forward

        setattr(ReloadableProcessGroup, name, make(name))


# 模块装载即生效：任何早于 monkey patch 的 import-time 绑定
# （如 Megatron 的 dist_reduce_scatter_func 别名）也能被正确转发
_forward_remaining_collectives()
```

miles/utils/test_utils/det_process_group.py

确定性测试在 torch 2.13 下的正确性关键：覆盖 reduce_scatter_single / reduce_scatter_single_coalesced / all_to_all_single 新入口，防止静默回到 NCCL 丢失固定序 fold；同时修复 coalescing flush 返回 None 的等待问题。

```
# miles/utils/test_utils/det_process_group.py
# DetProcessGroup 用固定顺序 fold 实现确定性 reduce_scatter / allreduce。
```

```
# torch 2.13 把 dist.reduce_scatter_tensor / dist.all_to_all_single 改走新的
# *_single 入口；若不覆盖，调用会落到 C++ 基类，经注册的 NCCL backend
# 静默执行真实 NCCL 通信，既丢掉固定序 fold，也可能在包装 group 上 SIGSEGV。

class DetProcessGroup: # 节选
    _inner: dist.ProcessGroup # 内层真实 backend，由 __init__ 注入

    def allgather_into_tensor_coalesced(
        self, output_tensors: list[torch.Tensor], input_tensors: list[torch.Tensor], opts: object =
        None
    ) -> Work:
        # torch 2.13 的 _coalescing_manager.__exit__ 会直接刷到这里，
        # 且此时内层 region 尚未关闭：内层调用被批量收集并返回 None，
        # 由 manager 在退出时统一 wait，因此只能在 work 非 None 时 wait
        effective_opts = opts if opts is not None else AllgatherOptions()
        for output, input in zip(output_tensors, input_tensors, strict=True):
            work = self._inner._allgather_base(output, input, effective_opts)
            if work is not None:
                work.wait()
        return _CompletedWork()

    def reduce_scatter_single(self, output: torch.Tensor, input: torch.Tensor, opts: object) ->
    Work:
        # torch 2.13 路由 dist.reduce_scatter_tensor 到这些名字；
        # 不覆盖就会走 C++ 基类并静默丢弃固定序 fold
        return self._reduce_scatter_base(output, input, opts)

    def reduce_scatter_single_coalesced(
        self, output_tensors: list[torch.Tensor], input_tensors: list[torch.Tensor], opts: object
    ) -> Work:
        # 合并形态复用既有的 coalesced 实现，逐对调用固定序 reduce_scatter
        return self.reduce_scatter_tensor_coalesced(output_tensors, input_tensors, opts)

    def all_to_all_single(
        self,
        output_tensor: torch.Tensor,
        input_tensor: torch.Tensor,
        output_split_sizes: list[int],
        input_split_sizes: list[int],
        opts: object,
    ) -> Work:
        # torch 2.13 的 dist.all_to_all_single 入口；虽然无归约语义，
        # 但 C++ 基类的 alltoall 在包装 group 上会触发 NCCL 故障（SIGSEGV），
        # 因此必须像 alltoall_base 一样转发到内层
        return self.alltoall_base(output_tensor, input_tensor, output_split_sizes, input_split_sizes,
        opts)
```

miles/backends/sglang_utils/sglang_engine.py

v0.5.18 的 ServerArgs 声明物化后只读，原先两个 launch 站点对 IPv6 host 的 strip 操作会抛异常；改为构造实例时统一去除方括号，并保留 dict 原样供 self.server_host 拼 url。

```
# miles/backends/sglang_utils/sglang_engine.py
# v0.5.18 的 ServerArgs 在声明物化后变为只读，原先两个 launch 站点
# 各自对 host 做 strip("[]") 会在新版本抛异常，因此统一前移到构造处。

class HttpServerEngineAdapter: # 节选
    def _init_normal(self, server_args_dict):
        use_rdt = self.args.update_weight_transfer_mode == "rdt"
        if use_rdt:
            if self.node_rank != 0:
                # 多节点 RDT 场景：node 0 的 RayEngine 会跨节点拉起全部
                # SchedulerActor，非零 rank 不需要再启动任何进程
                return
            server_args_dict["use_ray"] = True
            server_args_dict["enable_rdt_weight_sync"] = True
            assert self.pg_bundles
        logger.info(
            f"Launch HttpServerEngineAdapter at: {self.server_host}:{self.server_port}"
            f"{' (use_ray=True for RDT)' if use_rdt else ''}"
        )
        # ServerArgs 只读化之后无法再改写 host，必须在构造时一次性去掉
        # IPv6 方括号；dict 中保留带括号形式，供 self.server_host 拼 url 使用
        server_args = ServerArgs(**{**server_args_dict, "host": server_args_dict["host"].strip("[]")}
        )
```

评论区精华

人类 review 评论很少，核心信号集中在三处：

1. Zhichenzzz 在 miles/utils/reloadable_process_group.py:264 对 _forward_remaining_collectives() 给出「LGTM if @yueming-yuan tested this under GB300 nodes」，即 approve 的前提是 Blackwell/GB300 实机验证，因为该自动转发层会改变 GPU 上所有集合通信的调用路径；最终依托 run-ci-imaged 全量 e2e 通过后合入。
2. 机器人连续两次上报 test_mtp1_spec_v2_r3 失败（1h27m29s、31m34s），第三次 19m23s 通过；作者在 commit 907f5144 中记录三次隔离 /rerun-test 得到三种结果（NaN 超时、NCCL init 错误、通过），判定为 flaky 而非本 bump 回归，并在最后移除 run-ci-model-scripts 标签做收尾清理。
3. det PG 覆盖范围的自我纠偏（来自 commit 历史而非评论）：先为 torch 2.13 覆盖了 all_gather_single / all_gather_single_coalesced，随即判定为过度设计——all_gather 无归约语义、不需要固定序 fold——而回退；回退后又发现 _coalescing_manager.__exit__ 直接刷 allgather_into_tensor_coalesced 且内层返回 None，于是补上「work 非 None 才 wait」的容忍逻辑。这是少见的「先过度覆盖、再精确回退」的设计收敛过程。

- GB300 实机验证确认 (testing): 作者依托 run-ci-imaged 全量 e2e（H100 与多节点场景）通过后合入，无后续追问。

- MTP *e2e* flaky 判定 (testing): 以多次重跑交叉确认 flaky，继续推进合入；作者最终移除 run-ci-model-scripts 标签做收尾。
- det PG 覆盖范围的自我纠偏 (design): 只覆盖含归约语义的入口 + 容忍 batched None；注释精简为记录结论而非排查过程。

风险与影响

- 风险：
  1. 推理栈整体升级回归风险：sglang 从 v0.5.16 跨到 v0.5.18，fork 内 tokenizer_manager（abort-by-prefix、pre-dispatch、LoRA lease）、forward_batch_info（true_on_policy_contract）、deepseek_v2（共享专家融合决策）全部重写，RL on-policy 与 DSV4 会话服务是高风险区；CI 过程中已暴露 MQALayer 缺 tp_size 崩溃与 fusion 死锁互掩的问题并修复。
  2. torch2.13 wheel 迁移：链接 torch2.11 的 wheel（transformer_engine、flash-attn、apex、causal-conv1d、mamba-ssm 等）在 x86 上会因 materialize_cow_storage 缺失而启动失败；aarch64 默认 wheel 标签曾漏改，若未修正会在 ARM 镜像上静默使用旧 torch 依赖。
  3. 确定性测试正确性：det_process_group.py 若漏覆盖 torch 后续新增入口，会静默退回 NCCL 执行并丢失固定序 fold；all_to_all_single 未覆盖时曾直接 SIGSEGV。
  4. CI 镜像重建策略变更：改为按构建输入 hash 判定重建后，若 hash 覆盖不全面（wheel 指纹曾漏 torch213 集），可能复用旧镜像掩盖代码变更。
  5. *e2e* flaky 干扰：test_mtp1_spec_v2_r3 的既有 flaky 会给后续 bump 的回归判定带来噪声，需要依赖多次重跑交叉确认。 - 影响：影响范围覆盖 rollout/session server 推理链路、RDT 权重同步、确定性测试工具、docker 镜像与 CI 构建策略。镜像 tag 与 wheel 集变更会影响所有跑 run-ci-model-scripts 的 *e2e* 以及后续开发基镜像；生产可回滚到 sglang-miles-v0.5.16-final 或 sglang-miles-v0.5.17。团队侧收益：PR 镜像改为按输入 hash 重建，避免 Dockerfile 触碰后的重复多架构构建；R3 mismatch 日志常驻输出，提升确定性回归的交叉对比能力。 - 风险标记：推理栈主版本升级，torch 2.13 迁移，集体通信转发兼容，镜像构建策略变更，*e2e* flaky 干扰判定

关联脉络

- PR #2673 Bump Megatron-LM to miles-main-20260819 (latest NVIDIA dev): 本 PR body 明确 targets #2673，让 Megatron 与 sglang 两个升版在同一轮 CI 中联合验证；分支多次 merge main 引入其变更。
- PR #2734 docker: point MEGATRON_BRANCH back at miles-main: commit 5670cf9 记录本分支吸收 #2673 及其 follow-up #2734（MEGATRON_BRANCH 回退 miles-main），属于同一升版工作线的收尾。
- PR #1313 RDT weight sync: GPU->GPU zero copy transfer through SGLang Ray actor backend: 本 PR 修改的 sglang_server_actor.py / sglang_engine.py 正是 RDT 权重同步路径，launch_engine 六元组变化直接联动 RDT 服务端。

- PR #2717 add DeepSeek-V4-Flash-0731 support and mxfp4->fp8 converter: DSV4 会话测试在本次 CI 中暴露 MQALayer 缺 tp_size 崩溃并修复；Dockerfile 移除 flashinfer 0.6.15.post1 pin 也与 Blackwell trtllm MoE 运行相关。
- PR #2761 fix(ci): exempt .claude skill tooling from the test-discovery rule: 本 PR 内 commit 7e8374f 已先行引入对 .claude/skills 下 stray 文件的豁免，后续 #2761 是对该测试发现规则的进一步补全，属同一条 CI 治理线。