PR #1571 完整报告

radixark/miles

GLM-5.2 kernel fix and GB300 training config

合并时间：2026-08-04 08:57

原文链接：http://prhub.com.cn/radixark/miles/pull/1571

执行摘要

- 一句话：修复 sparse_mla_bwd NaN 并新增 GB300 RL 训练配方
- 推荐动作：值得精读，重点看三点：一是 PR body 对 kernel miscompile 的论证——为什么 Hopper 没炸不能作为 gate 条件，正确性修复不应依赖调度巧合，这是编译器 pass 配置踩坑的宝贵案例；二是显存瓶颈的实测推理——KV pool 而非 max-seq-len 才是 context 上限，mem-fraction 从 0.75 提到 0.85 是数据驱动而非拍脑袋；三是 --sglang-config 把复杂 rollout 引擎形状抽象为两个具名配方，并在 __post_init__用 assert 拒绝无法表达的组合，这种配置设计模式可直接借鉴到其他启动器。

功能与动机

PR body 将目标直写为 Brings up GLM-5.2 744B RL training on 16x GB300 (4 GPUs/node, NVL72)，并逐一给出三个 blocker 的量化证据：第一步训练死在 found NaN in local grad norm for bucket #0，定位为 DSA sparse MLA backward 的 miscompile（dV 干净、dQ/dK NaN，单卡任意序列长度可复现）；旧 TP8xPP8xEP8 拓扑的非均匀 PP 分片把 20.1B 参数堆到末级 rank，fp32 m/v 需要 150 GB 对 242 GB torch 预算导致 step-1 OOM；mem-fraction 0.75 时 KV pool 只有 26304 tokens 而每卡闲置约 59 GB，That pool, not --max-seq-len, is the real context ceiling，84% 的 rollout 样本被截断。另外还明确反驳了既有归因：The previous comment blamed the cookbook's 0.8 for OOM under RL; that is not what the memory shows。

实现拆解

1. kernel 修复（miles_plugins/models/glm5/ops/tilelang_sparse_mla_bwd.py）：把 TL_ENABLE_AGGRESSIVE_SHARED_MEMORY_MERGE 从 True 改为 False。bisect 显示翻转 { 关闭 merge、开启 warp-spec、num_stages >= 1} 任一都能修复，TMA/block_size/split_store/atomics 均无关；作者论证该 miscompile 并非架构专属（Hopper 同样生成错误别名，只是指令调度恰好掩盖），因此不按 device_capability 分支而是无条件禁用。代价约 kernel +5%（105.5 -> 111.0 ms @ S=32k）、端到端 <1%。
2. GB300 训练拓扑（scripts/run_glm5_2_744b_a40b.py）：新增 num_nodes >= 16 and num_gpus_per_node == 4 专用分支（TP8xPP4xDP2xEP16），PP 分片 [18,20,20,20] 让各 stage 起点落在 computing layer 1/19/39/59；_prepare_megatron_ckpt 的 EP 从硬编码 32 改为 world_size // 4（torch_dist checkpoint 对并行配置不敏感）；_convert_to_fp8 增加 sentinel（model.safetensors.index.json 存在即跳过）与

--max-workers 16。说明：body 提到把 --max-tokens-per-gpu 降到 6144 换取激活显存 headroom，但合入代码对完整模型保持 8192（提交历史有来回调整），最终取舍以代码为准。

3. engine recipe（同脚本）：ScriptArgs 新增 sglang_config: Literal["low-latency", "balanced"] = "low-latency"，__post_init__ 用两个 assert 拒绝 balanced 无法表达的组合（PD 分离、GPU 数非 4 的倍数）。balanced = 每节点一台 4-GPU engine + dp-attention（dp-size 4，隐式开启 dp-aware 路由）+ deepep 恒开 + EAGLE 1/1/2 + max-running 256 + chunked-prefill 32768；low-latency 保持每节点对一台 TP8 engine、EAGLE 5/1/6、max-running 512。
4. mem-fraction 与 CI 适配（同脚本）：mem-fraction 改为 0.70 if num_nodes == 1 else 0.85；非 PD 场景 engine 大小上限 min(8, num_gpus_per_node)，修复 4-GPU CI 上 8-rank engine 的 TCPStore 超时；新增 --sglang-router-policy consistent_hashing。开发中期引入的 CPU 内存采样（miles/utils/memory_utils.py、actor.py 挂载点）在 Remove memory probes 提交中被整体移除，未进入合入版本。

关键文件：

- scripts/run_glm5_2_744b_a40b.py（模块 训练脚本；类别 source；类型 core-logic；符号 ScriptArgs, _convert_to_fp8, _prepare_megatron_ckpt, _execute_train）：主训练脚本，承载 GB300 拓扑分支、--sglang-config 双引擎配方、mem-fraction 提升与非 PD engine 大小修复，是 64 卡 GB300 RL 训练能否跑通的核心配置载体。
- miles_plugins/models/glm5/ops/tilelang_sparse_mla_bwd.py（模块 模型算子；类别 source；类型 core-logic；符号 postprocess_kernel, bwd）：kernel 正确性修复的唯一落点，一行 pass 配置翻转解决 744B 训练第一步 NaN，并论证了该 miscompile 非架构相关、必须无条件禁用而非按 device_capability 分支。

关键符号：postprocess_kernel, bwd, ScriptArgs.post_init, _convert_to_fp8, _prepare_megatron_ckpt, _execute_train

关键源码片段

scripts/run_glm5_2_744b_a40b.py

主训练脚本，承载 GB300 拓扑分支、--sglang-config 双引擎配方、mem-fraction 提升与非 PD engine 大小修复，是 64 卡 GB300 RL 训练能否跑通的核心配置载体。

```
# scripts/run_glm5_2_744b_a40b.py
# engine recipe 决策：--sglang-config 二选一。
# low-latency（默认）= 每节点对一台 TP8 engine，EAGLE 5/1/6，追求低延迟；
# balanced = 每节点一台 4-GPU engine，dp-attention + deepep，对齐 cookbook
# 的 serving 形态，吞吐优先。balanced 无法表达的组合在 __post_init__ 里
# 用 assert 拒绝：PD 分离、GPU 数不是 4 的倍数。

balanced = args.sglang_config == "balanced"
if args.enable_pd:
    # PD 分离走独立的 prefill/decode 引擎组
    sglang_decode_max_bs = 8
    sglang_world_size = 16 if args.num_nodes < 16 else 64
```

```
else:
    sglang_decode_max_bs = 32
    # 非 PD 时 engine 大小不能超过节点卡数：4-GPU CI 起 8-rank engine
    # 会卡在 TCPStore（4/8 客户端超时）
    sglang_world_size = 4 if balanced else min(8, args.num_gpus_per_node)

sglang_args = (
    f"--rollout-num-gpus-per-engine {sglang_world_size} "
    # mem-fraction 0.85：64x GB300 实测 0.75 时 KV pool 只有 26304 tokens、
    # 每卡闲置约 59 GB，KV pool 才是真正的 context 上限（max-seq-len 不是）；
    # 提到 0.85 后 pool 达 553728 tokens，截断率从 84% 降到 0.0-0.3，
    # 稳态仍余 33 GB/ 卡。单节点 5 层 smoke（4x H200）保留 0.70：剪枝模型
    # 权重极少，0.85 几乎全变成 KV cache，weight-checker 快照无处分配。
    f"--sglang-mem-fraction-static {0.70 if args.num_nodes == 1 else 0.85} "
    f"--sglang-ep-size {sglang_world_size} "
    "--sglang-router-policy consistent_hashing "
)
if args.enable_pd:
    # slime 原生配置：dp-lm-head + dense-tp 1
    sglang_args += (
        "--sglang-enable-dp-attention "
        f"--sglang-dp-size {sglang_world_size} "
        "--sglang-moe-dense-tp-size 1 "
        "--sglang-enable-dp-lm-head "
    )
elif balanced:
    # dp-attention 会隐式开启 dp-aware 路由（见 sglang_utils.arguments）：
    # min_load 负载均衡必须能看到 dp rank，否则请求全堆到 sglang 内部
    # 选的同一个 rank，其余 rank 空转。
    sglang_args += "--sglang-enable-dp-attention " f"--sglang-dp-size {sglang_world_size} "
if balanced or (args.fp8_rollout and args.use_deepep):
    sglang_args += "--sglang-moe-a2a-backend deepep "

if args.enable_mtp:
    # balanced 对齐 cookbook 的 serving 深度；low-latency draft 更深，
    # 只在 engine 不忙时划算
    steps, draft_tokens = (1, 2) if balanced else (5, 6)
    sglang_args += (
        "--sglang-speculative-algorithm EAGLE "
        f"--sglang-speculative-num-steps {steps} "
        "--sglang-speculative-eagle-topk 1 "
        f"--sglang-speculative-num-draft-tokens {draft_tokens} "
        "--sglang-speculative-draft-attention-backend nsa "
    )
# 并发上限：balanced 用 256/32768，low-latency 用 512/2048*world
sglang_args += (
    f"--sglang-max-running-requests {256 if balanced else 512} "
    f"--sglang-chunked-prefill-size {32768 if balanced else 2048 * sglang_world_size} "
)
```

miles_plugins/models/glm5/ops/tilelang_sparse_mla_bwd.py

kernel 正确性修复的唯一落点，一行 pass 配置翻转解决 744B 训练第一步 NaN，并论证了该 miscompile 非架构相关、必须无条件禁用而非按 device_capability 分支。

```
# miles_plugins/models/glm5/ops/tilelang_sparse_mla_bwd.py
# 本文件承载 GLM-5.2 DSA sparse MLA 反向 kernel 的 TileLang pass 配置。
# 背景：744B 训练第一步即报 found NaN in local grad norm for bucket #0，
# 定位到本 kernel：dV 干净、dQ/dK 为 NaN，任意单卡、任意序列长度可复现。

@postprocess_kernel(
    pass_configs={
        tilelang.PassConfigKey.TL_DISABLE_TMA_LOWER: True,
        tilelang.PassConfigKey.TL_DISABLE_WARP_SPECIALIZED: True,
        # 修复点：TileLang 0.1.9 的 aggressive shared-memory merge 会把
        # 仍存活（still live）的 shared buffer 做别名合并，在 warp-spec off
        # + 非流水线组合下直接产出 NaN 的 dQ/dKV。bisect 结论：翻转
        # { 关闭 merge、开启 warp-spec、num_stages >= 1} 任一都能修复，
        # 而 TMA/block_size/split_store/atomics 均无关。
        # 这个 miscompile 不是架构专属：Hopper 同样生成错误别名，只是指令
        # 调度恰好把访问隔开、没让损坏浮出水面——这是运气不是保证。
        # TileLang 升版、shape 变化、流水线策略改变都可能打破它，所以
        # 无条件禁用（True -> False），而不是按 device_capability >= 10
        # 做架构分支：正确性修复不应取决于今天恰好是哪个架构掩盖了 bug。
        # 代价：本 kernel 约 +5% 耗时（105.5 -> 111.0 ms @ S=32k），
        # 端到端 <1%，三个候选修复方案测得结果一致。merge-off 仅在使用
        # block_H >= 64（TP <= 2）时才需要额外 smem 容量，GLM-5 配方不用。
        tilelang.PassConfigKey.TL_ENABLE_AGGRESSIVE_SHARED_MEMORY_MERGE: False,
    },
)
def bwd(...):
    # DSA sparse MLA 反向主体。修复后 50/50 轮压力测试（变化 shape/seed/
    # 并发加载）全部干净通过，rel_err ~0.003（相对 fp32 参考实现）。
    ...
```

评论区精华

四位 reviewer 中三位维护者（guapisolo、Shi-Dong、yushengsu-thu）直接 APPROVED，唯一有实质意见的是 gemini-code-assist[bot] 的 3 条评论，且全部指向开发中期引入、随后被回退的 memory probe 代码：

- bot 指出 host_memory() 直接 open(/proc/meminfo) 无 try/except，在非 Linux 平台或受限容器内可能以 FileNotFoundError 或 PermissionError 打崩整个训练进程，并给出逐字 suggestion。
- bot 指出 start_cpu_memory_profiler 写 CSV 前未确保父目录存在，interpreter 退出时可能崩溃，建议 os.makedirs(dirname, exist_ok=True)。
- bot 指出 args.memory_snapshot_dir 为 None 时输出路径会变成 None/cpu_memory_...，建议用 getattr(args, 'memory_snapshot_dir', None) or '.' 兜底。

三条意见本身都成立，但所针对的代码是排查 GB300 host 峰值时临时加入的 probe，最终随 Remove memory probes 提交整体回退，合入版本已不包含相关文件，因此没有产生实际修改。PR body 中另一处值得注意的纠偏是：之前有人把 RL 场景 OOM 归咎于 cookbook 的 0.8 mem-fraction，作者用 21 小时运行的显存曲线反驳——瓶颈是 KV pool 容量与并发，不是 mem-fraction 本身。

- /proc/meminfo 读取缺少异常保护 (correctness): 所针对的代码是调试期临时加入的 memory probe，最终随 Remove memory probes 提交整体回退，合入版本不包含 miles/utils/memory_utils.py，未产生实际修改。
- CPU memory profiler 未创建输出父目录 (correctness): 同上，该代码随 memory probes 一起回退，不进入合入版本。
- memory_snapshot_dir 为 None 时输出到 None 目录 (correctness): 同上，代码已回退；该模式提示未来若保留 memory profiling 功能需做默认值处理。

风险与影响

- 风险：
  - 确定性 kernel 性能回归：无条件禁用 aggressive smem merge 后，sparse_mla_bwd 在所有架构（含 Hopper）上慢约 5%（105.5 -> 111.0 ms @ S=32k），端到端 <1%；对有意的正确性 / 性能取舍，但对 1000 s/step 的 744B 训练是持续的成本。
  - 缺少 kernel 回归测试：本次无任何测试文件变更，针对 TileLang miscompile 的修复没有自动化保护；TileLang 升版、shape 变化或流水线策略改变可能让同类 NaN 以其他形式回归，且 NaN 通常要到 step-1 才暴露。
  - 硬件形态假设脆弱：GB300 分支条件与 _prepare_megatron_ckpt 的 world_size // 4 都隐含每节点 4 卡假设；若未来出现 8 卡 / 节点的 GB300 集群会静默落入旧 32 节点分支（在 GB300 上必 OOM），脚本缺少显式 guard。
  - mem-fraction 0.85 验证面有限：0.85 仅在 64x GB300 实测（稳态余 33 GB/ 卡）；2-15 节点等中间规模跑全量模型也会走 0.85 分支但无显存实测，单节点 5 层模型已用 0.70 兜底，其他形态需留意。
  - moe-runner-backend 条件变化：新条件 not balanced and not (fp8 and deepep) 意味着 balanced 配方完全不设置 trtllm_routed 备选，完全依赖 deepep A2A 路径，若 deepep 在某个硬件组合不可用则没有退化路径。
- 影响：
  - 训练管线：直接使能 GLM-5.2 744B 在 16 节点 GB300（64 GPU，NVL72）上的 RL 训练，实测完整 step 约 1000 s/step、GPU 228/242 GB、host 峰值 858/898 GiB；默认配方下 32 节点 256 GPU 配置与 4x H200 单节点 smoke 行为不变。
  - kernel 用户面：所有使用 DSA sparse MLA 反向 kernel 的训练（不限 GLM-5.2）都获得正确性修复，代价是 kernel 约 5% 变慢。
  - 脚本与开发体验：新增公开参数 --sglang-config（默认 low-latency，向后兼容）；fp8 转换支持 sentinel 跳过与 --max-workers 16，重复执行可复用已转换权重；非 PD 场景 engine 大小上限修复了 4-GPU CI 的 TCPStore 超时。

- 团队流程：14 个提交记录了探测 - 定位 - 回退的完整调试路径，memory probes 被彻底移除，合入版本只保留必要改动；无新增测试是本次的主要流程缺口。
- 风险标记：缺少测试覆盖，Kernel 性能回归约 5%，编译期正确性依赖 TileLang 版本，硬件形态假设脆弱

关联脉络

- PR #2012 router: enable dp-aware routing under dp-attention: 本 PR 的 balanced 配方依赖 dp-attention 隐式开启的 dp-aware 路由，代码注释明确引用 sglang_utils.arguments 中的该行为，属于前置依赖链。
- PR #2041 Address engines by base URL when the router is dp-aware: dp-aware 路由下引擎寻址的修复，是本 PR balanced 配方并发与负载均衡语义成立的配套。
- PR #2047 fix glm47-flash: use paged MLA prefill on B200: 同属 GLM 家族在 Blackwell（B200/GB300）上的 rollout 修复线，与本 PR 共享 Blackwell 训练上下文。
- PR #1928 [fix] DSA indexer on Blackwell: send the DSA indexer wk unquantized: 同为 DSA 架构在 Blackwell 上的正确性修复，与本 PR 的 kernel 修复共享 DSA/Blackwell 上下文。