PR #2673 完整报告

radixark/miles

Bump Megatron-LM to miles-main-20260819 (latest NVIDIA dev)

合并时间：2026-08-25 03:37

原文链接：http://prhub.com.cn/radixark/miles/pull/2673

执行摘要

- 一句话：升级 Megatron-LM 至 20260819 并全面适配 API 变更
- 推荐动作：值得精读。这是 " 大版本依赖升级如何控制风险面 " 的样板：以 base bump only 为边界、用显式禁用隔离未适配特性、把大特性拆成 follow-up PR。重点阅读 miles/backends/megatron_utils/arguments.py 中的数值决策注释与 model.py 中 MTP loss tracker 适配模式，以及 update_weight/common.py 的命名正则。同时建议关注 #2734 回退 MEGATRON_BRANCH 的原因，评估当前 bump 是否需补丁修缮。

功能与动机

PR body 明确说明升级动机："Bump the Megatron-LM that miles depends on from the stale 2026-02-14 fork point to latest NVIDIA/Megatron-LM dev"，并强调 "Scope: base bump only"，DeepSeek-V4 和 true-on-policy 被显式禁用、留到专门的 follow-up PR（DSv4 优先，然后是 true-on-policy）。这是对 #1466（6 月 bump，从未合并）的取代与 rebase 延续。

实现拆解

本 PR 的实现按以下 6 步推进：

1. 依赖入口升级：docker/Dockerfile 中 MEGATRON_BRANCH 从 miles-main 改为 miles-main-20260819，并由 Zhichenzzz 将 emerging-optimizers bump 到 v0.3.0 以匹配 Megatron 的 pin。这是全部适配工作的上游前提。
2. 参数入口集中适配（miles/backends/megatron_utils/arguments.py）：set_default_megatron_args 新增三处关键逻辑——--true-on-policy-mode 在 megatron backend 直接抛 NotImplementedError；强制 args.moe_router_use_torch_mm = True（上游默认切换为 TE gemm 后 bf16×bf16 精度不足，实测放大 log prob diff）；args.trust_remote_code = True（新 Megatron 用该开关门控自定义 tokenizer 加载，Kimi 类 checkpoint 依赖它）。同时把 _vocab_size_with_padding 的导入路径从 megatron.training.tokenizer.tokenizer 迁移到 megatron.core.tokenizers.utils.build_tokenizer。
3. 训练循环与模型装配适配（miles/backends/megatron_utils/model.py、model_provider.py）：enable_gloo_process_groups 参数重命名为 use_gloo_process_groups（4 处）；forward_step 中移除手动传入的 mtp_kwargs（上游

process_mtp_loss 从 input_ids 自动派生 MTP labels）；MTP loss 读取改用新 API MTPLossLoggingHelper.reduce_loss_in_tracker() 并兼容 values/loss_values 两个字段；model_provider 在开启 enable_mtp_training 时设置 config.mtp_detach_heads = True，并移除已删除的 moe_use_legacy_grouped_gemm 参数；bridge provider 补上 gradient_accumulation_fusion 透传。

4. 模型层与权重转换适配：miles_plugins/models/inkling/layers.py 中 InklingRouter 改用上游 mark_keep_in_fp32 替代私有 _keep_fp32 标记、forward 签名新增 packed_seq_params，InklingSharedExperts 透传新增的 name= kwarg；inkling/model.py 把 MLP/ 共享专家子模块从 ModuleSpec 迁移到 functools.partial（MlpBuilder）方式；update_weight/common.py 的 MTP expert 权重名正则扩展为同时匹配 transformer_layer|mtp_model_layer，修复 EP 权重同步死锁；各 megatron_to_hf 转换器与 mbridge 插件同步更新 MTP 子模块命名。
5. 测试与功能禁制：test_model_provider_true_on_policy.py 跳过、DSv4 4-layer e2e 通过 register_cuda_ci(disabled=...) 禁用、GLM5.1/5.2 LoRA e2e 与 Nemotron-3 Ultra e2e 因 Megatron-Bridge 未同步而临时禁用，并更新相关 snapshot 断言。
6. 顺带合入的 CI 优化：通过 merge 分支 ci-image-rebuild-on-input-change（#2675）合入 " 按镜像构建输入哈希决定是否重建 PR 镜像 " 的逻辑，避免 Dockerfile 变更类 PR 每次 rerun 都重复多架构构建。

关键文件：

- miles/backends/megatron_utils/arguments.py（模块 参数解析；类别 source；类型 dependency-wiring；符号 set_default_megatron_args）：参数入口集中适配：禁用 megatron true-on-policy、强制 router 回退 torch.mm、开启 trust_remote_code，并迁移 tokenizer 工具导入路径。这里锁定了新版本的数值与行为边界。
- miles/backends/megatron_utils/model.py（模块 训练循环；类别 source；类型 data-contract；符号 setup_model_and_optimizer, forward_step, train）：训练循环核心改动：use_gloo_process_groups 重命名、移除 mtp_kwargs 手动传参、MTP loss tracker 迁移到新 API 并修正双重缩放。
- miles_plugins/models/inkling/layers.py（模块 模型层；类别 source；类型 data-contract；符号 InklingRouter.init, InklingRouter.forward, InklingSharedExperts.init）：Inkling 模型层适配新 Megatron 的模块契约：mark_keep_in_fp32 替代私有标记、forward 签名新增 packed_seq_params、共享专家透传 name kwarg。
- miles_plugins/models/inkling/model.py（模块 模型装配；类别 source；类型 data-contract；符号 get_inkling_layer_spec, get_inkling_dense_layer_spec）：Inkling 模型装配适配：MLP/ 共享专家子模块从 ModuleSpec 迁移到 MlpBuilder partial，这是新 Megatron 的 submodule 构建契约变化。
- miles/backends/megatron_utils/model_provider.py（模块 模型提供；类别 source；类型 data-contract；符号 model_provider, _apply_bridge_runtime_config）：模型提供路径适配：移除已删除的 moe_use_legacy_grouped_gemm 参数、开启 mtp_detach_heads，并补上 bridge 的 gradient_accumulation_fusion 透传。
- miles/backends/megatron_utils/update_weight/common.py（模块 权重同步；类别 source；类型 core-logic；符号 _named_params_and_buffers_global）：MTP expert 权

重同步正则扩展为同时匹配 transformer_layer 与 mtp_model_layer，修复因上游子模块改名导致的 EP 权重同步死锁。

- docker/Dockerfile（模块 镜像构建；类别 infra；类型 configuration）：依赖入口升级：MEGATRON_BRANCH 从 miles-main 切到 miles-main-20260819，配套 emerging-optimizers v0.3.0；这是整轮 bump 的起点。

关键符号：set_default_megatron_args, setup_model_and_optimizer, train, forward_step, model_provider, _apply_bridge_runtime_config, InklingRouter.init, InklingRouter.forward, InklingSharedExperts.init, get_inkling_layer_spec, get_inkling_dense_layer_spec, _named_params_and_buffers_global

关键源码片段

miles/backends/megatron_utils/arguments.py

参数入口集中适配：禁用 megatron true-on-policy、强制 router 回退 torch.mm、开启 trust_remote_code，并迁移 tokenizer 工具导入路径。这里锁定了新版本的数值与行为边界。

```
def set_default_megatron_args(args):
    # 新 Megatron 尚未为 miles 适配 true-on-policy 后端，直接抛错而不是静默降级，
    # 避免用户拿到错误的训练行为；fsdp 后端不受影响。
    if getattr(args, 'true_on_policy_mode', False):
        raise NotImplementedError(
            '--true-on-policy-mode is not supported on the megatron backend with this '
            'Megatron version; support lands in a follow-up PR. '
            'Use --train-backend fsdp for true-on-policy.'
        )

    # Muon 走自己的分片路径，Megatron 的分布式优化器只支持 Adam 系，
    # 因此只有 Adam 系列且未禁优化器时才启用 distributed optimizer。
    args.use_distributed_optimizer = (
        args.optimizer is None or args.optimizer.lower() == 'adam'
    ) and not getattr(args, 'debug_disable_optimizer', False)

    # Multi-LoRA 的逐 slot LayerWise 优化器要求普通 DDP all-reduce。
    if getattr(args, 'multi_lora_n_adapters', 0) > 0:
        args.use_distributed_optimizer = False

    args.bf16 = not args.fp16

    # 占位：seq_length 缺省时补 4096，并让 max_position_embeddings 跟随。
    if args.seq_length is None:
        args.seq_length = 4096
    args.max_position_embeddings = args.seq_length

    # 老分支兼容开关：dist_ckpt_save_pre_mcore_014 已被 dp_reshardable 取代。
    if os.getenv('DEPRECATED_MEGATRON_COMPATIBLE', '0') == '1':
        args.dist_ckpt_save_pre_mcore_014 = True

    # 20260819 前 router 默认走 torch gemm（fp32 x fp32 -> fp32），切到 TE gemm 后
```

```
    # 变成 bf16 x bf16 -> fp32，实测会放大 log prob diff，因此显式改回 torch.mm。
    args.moe_router_use_torch_mm = True

    # 兼容 Megatron：MLA 默认用 yarn RoPE，普通注意力用 rope。
    if hasattr(args, 'rope_type') and args.rope_type is None:
        args.rope_type = 'yarn' if args.multi_latent_attention else 'rope'

    if args.vocab_size and not args.padded_vocab_size:
        args.padded_vocab_size = _vocab_size_with_padding(args.vocab_size, args)

    if not args.tokenizer_model and not args.tokenizer_type:
        # 未显式指定 tokenizer 时直接复用 --hf-checkpoint 作为 tokenizer。
        logger.info('--tokenizer-model not set, use --hf-checkpoint as tokenizer model.')
        args.tokenizer_model = args.hf_checkpoint
        args.tokenizer_type = 'HuggingFaceTokenizer'

    # Megatron 新版要求显式 trust_remote_code 才允许加载自定义 tokenizer 代码。
    args.trust_remote_code = True

    if not hasattr(args, 'miles_dsa_topk_backend'):
        args.miles_dsa_topk_backend = 'torch'

    return args
```

miles/backends/megatron_utils/model.py

训练循环核心改动：use_gloo_process_groups 重命名、移除 mtp_kwargs 手动传参、MTP loss tracker 迁移到新 API 并修正双重缩放。

```
  if args.enable_mtp_training:
    from megatron.core.transformer.multi_token_prediction import MTPLossLoggingHelper

    # 新版 helper 已内置 per-token 归一化（sum(loss) / sum(tokens)），
    # 并且 reduce_loss_in_tracker() 内部完成跨 rank all-reduce；
    # 若仍按旧逻辑乘 1 / num_microbatches，mtp_loss 会被二次缩放 N 倍。
    mtp_loss_scale = 1.0 if args.calculate_per_token_loss else 1 / num_microbatches[step_id]
    MTPLossLoggingHelper.reduce_loss_in_tracker()
    tracker = MTPLossLoggingHelper.tracker

    mtp_losses = None
    # 新 API 在 reduce 之后才产出 values（旧字段）或 loss_values（新字段），
    # 这里同时兼容两种 key，并假定只有一个 MTP 层。
    if 'values' in tracker:
        mtp_losses = (tracker['values'] * mtp_loss_scale).item()
    elif 'loss_values' in tracker:
        mtp_losses = (tracker['loss_values'] * mtp_loss_scale).item()

    if mtp_losses is not None:
        MTPLossLoggingHelper.clean_loss_in_tracker()
```

```
    # 后续代码用 mtp_losses 执行 CI 区间校验（check_mtp_loss），
    # 数值口径变化后旧阈值可能需要重新校准。
```

miles_plugins/models/inkling/layers.py

Inkling 模型层适配新 Megatron 的模块契约：mark_keep_in_fp32 替代私有标记、forward 签名新增 packed_seq_params、共享专家透传 name kwarg。

```
class InklingRouter(TopKRouter):
    def __init__(self, *args, **kw):
        super().__init__(*args, **kw)
        H = self.config.hidden_size
        ns = self.config.inkling.n_shared_experts
        self.shared_gate = nn.Parameter(torch.zeros(ns, H, dtype=self.config.params_dtype))
        # 上游不再识别私有 _keep_fp32 标记，统一改用 mark_keep_in_fp32，
        # 使 global_scale 免于被 Float16Module 转成 bf16。
        self.global_scale = nn.Parameter(torch.ones(1, dtype=torch.float32))
        mark_keep_in_fp32(self.global_scale)
        self._cache_key = None
        self._cache = None

    # 上游 TopKRouter.forward 新增 packed_seq_params，MoELayer 会按位置传参，
    # 签名不对齐会直接报错；前缀缓存（replay）语义保持不变。
    def forward(self, input, padding_mask=None, input_ids=None, packed_seq_params=None):
        key = (input.data_ptr(), tuple(input.shape))
        if self._cache_key == key and self._cache is not None:
            probs, routing_map = self._cache
            self._cache_key = None
            self._cache = None
            return probs, routing_map

        self._maintain_float32_expert_bias()
        H = input.shape[-1]
        nr, topk = self.config.num_moe_experts, self.topk
        logits = self.gating(input).view(-1, nr).float()
        shared_logits = input.reshape(-1, H).float() @ self.shared_gate.float().t()
        score = logits.sigmoid() + self.expert_bias.float()

        # 经 routing replay manager 选择 top-k，支持重放 / 确定性控制。
        _sel_topk = routing_replay_manager.get_topk_fn(
            lambda s, k: s.topk(k, dim=-1).indices, return_probs=False
        )
        topk_ids = _sel_topk(score, topk).long()
        sel_logits = logits.gather(-1, topk_ids)
        active = torch.cat([sel_logits, shared_logits], dim=-1)
        lp = F.logsigmoid(active)
        w = torch.softmax(lp, dim=-1) * float(self.config.inkling.route_scale) * self.global_scale.
        float()
        routed_w, shared_w = w[:, :topk], w[:, topk:]
        probs = torch.zeros_like(logits).scatter(-1, topk_ids, routed_w)
```

```
        routing_map = torch.zeros_like(logits, dtype=torch.bool).scatter(-1, topk_ids, True)
        self.config.inkling._shared_w = shared_w
        self._apply_expert_bias(routing_map, padding_mask)
        # （下游省略：按 config.moe_router_dtype 处理输出精度后返回 probs, routing_map）


class InklingSharedExperts(MegatronModule):
    def __init__(self, config, submodules=None, gate=False, pg_collection=None, name=None):
        super().__init__(config=config)
        ns = config.inkling.n_shared_experts
        inter = config.inkling.intermediate_size
        # ... 中间省略 tp group 的构造 ...
        shared_cfg = _copy.copy(config)
        shared_cfg.ffn_hidden_size = inter
        # 新 Megatron 的 build_module 会给 spec 模块传 name= 用于生成 checkpoint key,
        # 这里把 name 透传给每个 expert 的 MLP，否则权重命名对不上。
        self.experts = nn.ModuleList(
            [
                MLP(
                    config=shared_cfg,
                    submodules=submodules,
                    ffn_hidden_size=inter,
                    tp_group=tp,
                    name=f'{name}.experts.{i}' if name is not None else None,
                )
                for i in range(ns)
            ]
        )
```

评论区精华

核心讨论如下：

- MTP loss 双重缩放（guapisolo，P2，正确性）：MTPLossLoggingHelper.reduce_loss_in_tracker() 在 --calculate-per-token-loss 下已计算 sum(loss) / sum(tokens)，而 Miles 仍无条件乘 1 / num_microbatches，导致 N 个 microbatch 时 mtp_loss 和 check_mtp_loss 被多缩放 N 倍。结论：提交 59cc3c11 修复，per-token 模式 scale 恒为 1.0，legacy 保留 microbatch 平均，并经过 pre-commit hooks 与 AST 校验（per_token=1.0、legacy=0.25、4 microbatches）。
- 仅更新 CUDA Dockerfile 的覆盖缺口（Codex，P1）：Dockerfile.rocm:24、_run-ci.yml:140、_run-cpu-ci.yml:80、release-branch-cut.yml:101 仍默认 miles-main 旧分支，而新代码已导入新 tokenizer 模块、读取 use_gloo_process_groups，nightly/post-merge CI 与 release 镜像会构建失败。结论：未彻底解决，guapisolo 表示 "novita 2 has been fixed. re-kick off ci"，ROCm 与 release 管道需后续同步。
- 共享 merge_fn 不应依赖训练全局（Codex Review 2，P1）：megatron 侧提交 4c725b138 在 cat_with_oom_fallback 中无条件读 get_args().low_memory_resume，而该函数是 ShardedTensorFactory.merge_fn，被 SSM、QKV attention、GLU MLP

checkpoint 工厂和公开 dist-ckpt 加载路径使用。结论：属于对上游 megatron 仓库的设计疑虑，本 PR 未处理。

- dist_muon 路径验证（Zhichenzzz，question）："did you test the dist_muon path for the new branch? worthy having it"。结论：最终通过 bump emerging-optimizers v0.3.0 匹配 Megatron pin 并获批。
- claude[bot] 人工复核建议：未发现 bug，但因涉及训练循环、LoRA、MTP、权重量化转换等 numerics-sensitive 适配，建议人工 review。
  - MTP 每 token loss 被 microbatch 数量二次缩放 (correctness): 提交 59cc3c11 修复：per-token 模式 scale 恒为 1.0，legacy 保留 microbatch 平均；通过 pre-commit hooks 与 AST 校验（per_token=1.0、legacy=0.25、4 microbatches）。
  - 仅更新 CUDA Dockerfile，其他 Megatron 消费方仍指向旧分支 (design): 未彻底解决；guapisolo 表示 novita 2 已修复并重新触发 CI，ROCm 与 release 管道需后续同步。
  - 共享 checkpoint merge_fn 不应依赖训练全局 args (design): 属于对 radixark/Megatron-LM 上游的设计疑虑，本 PR 未处理，需上游维护者评估。
  - dist_muon 路径是否在新分支验证 (question): 通过后续 bump emerging-optimizers v0.3.0 匹配 Megatron pin，Zhichenzzz 最终 approve。

风险与影响

- 风险：
  1. 镜像 /CI 不同步（高）：Dockerfile.rocm、_run-ci.yml、_run-cpu-ci.yml、release-branch-cut.yml 仍指向旧分支 miles-main，新代码引用新 tokenizer 模块与重命名后的参数，post-merge CI、nightly 与 release 构建存在失败风险。仓库后续 PR #2734 将 MEGATRON_BRANCH 指回 miles-main，也从侧面印证本轮 bump 在镜像链路上的稳定性存疑。
  2. 数值行为变化（中高）：router gating 已在 arguments.py 强制回退 torch.mm，但 MTP loss tracker 语义变化会影响 check_mtp_loss 的数值口径，依赖旧阈值的调用方需重新校准；mark_keep_in_fp32 与旧私有 _keep_fp32 的覆盖范围差异也可能影响 fp32 保持行为。
  3. 功能显式禁用（中）：megatron backend 的 true-on-policy 直接抛 NotImplementedError，DSv4 4-layer e2e、GLM5.1/5.2 LoRA e2e、Nemotron-3 Ultra e2e 被跳过，这些路径在 follow-up PR 落地前存在回归覆盖缺口，且用户可能误用后得到报错或静默跳过。
  4. 权重同步死锁（中）：update_weight/common.py 已通过正则兼容 transformer_layer|mtp_model_layer 修复 EP 权重同步死锁，但若上游再次调整 MTP 命名，该模式仍可能复发。 - 影响：影响范围较广：所有使用 Megatron backend 的训练用户需要重建镜像并重新验证数值稳定性；CI/release 管道因分支不一致存在阶段性构建风险；测试侧有 4 个 e2e 测试被临时禁用，回归覆盖暂时收缩。正面影响是 miles 与上游 NVIDIA/Megatron-LM dev 的对齐成本显著降低，后续 bump 节奏可以加快；同时顺带合入的镜像按输入重建优化减少了 PR 重复构建成本。团队需要按序跟进 DSv4 与 true-on-policy 两个 follow-up PR，并在 ROCm/release 镜像上补齐同步，才能完全收口。 - 风险标记：镜像 /CI 不同步，核心训练路径变更，功能显式禁用，数值行为变化，

e2e 测试被跳过

关联脉络

- PR #2734 docker: point MEGATRON_BRANCH back at miles-main: 与本 PR 同一条 MEGATRON_BRANCH 的相反操作：2734 又把分支指回 miles-main，说明 2673 的 bump 在镜像链路或功能上出现需要回退的问题，应重点排查回归原因。
- PR #1466 June Megatron bump（PR body 提及，从未合并）: PR body 明确 "Supersedes #1466 (June bump, never merged)"，本 PR 的 megatron 侧 20 个提交中有 14 个从 20260622 分支 rebase 而来。
- PR #2717 add DeepSeek-V4-Flash-0731 support and mxfp4->fp8 converter: PR body 明确 DeepSeek-V4 是 follow-up PR；本 PR 通过 register_cuda_ci(disabled=...) 禁用了 DSv4 4-layer *e2e*，两个 PR 构成同一功能线的先后关系。
- PR #2675 [CI] rebuild a PR's docker image only when its build inputs change: 本 PR 通过 merge origin/ci-image-rebuild-on-input-change 顺带合入了该 CI 优化（提交 5bff5856、316dab06），二者是相同提交来源。
- PR #2682 fix: resume from the checkpoint step in bridge mode: 与本 PR 同属 bridge 模式链路，本 PR 在 model_provider.py 中补充了 bridge 的 gradient_accumulation_fusion 透传，是同一模块的连续演进。