PR #2251 完整报告

THUDM/slime

Internalize mbridge and remove megatron-bridge

合并时间：2026-08-04 15:19

原文链接：http://prhub.com.cn/THUDM/slime/pull/2251

执行摘要

- 一句话：内部化 mbridge，移除 megatron-bridge 依赖
- 推荐动作：值得精读。这是 slime 一次重要的架构收敛：从外部桥接模型转换切换到自研的轻量直接转换器，同时明确了模型支持的边界。建议重点阅读 slime/backends/megatron_utils/hf_to_megatron/qwen.py 的映射逻辑和 slime_plugins/models/qwen3_5_vl.py 的原生 VLM 实现，理解新的模型接入成本和约束。

功能与动机

PR body 明确指出：在 slime 当前发展阶段，无法及时支持每个模型，实验性 Megatron-Bridge 集成仅部分实现、达不到代码质量和可维护性标准；独立 MBridge 项目已弃用并迁移至 NVIDIA-NeMo/Megatron-Bridge，因此将 slime 所需功能直接内部化，以获得清晰的所有权和按需维护能力。

实现拆解

1. 移除 mbridge 与 megatron-bridge 插件层：删除 slime_plugins/mbridge/ 下的 qwen3_5.py、gemma4.py、qwen3_next.py、gpt_oss.py、glm4moe.py、mimo.py、glm4.py、deepseek_v32.py、glm4moe_lite.py 等 Bridge 子类，以及 slime_plugins/megatron_bridge/glm4v_moe.py；这些文件承担了从 HF 权重到 Megatron 格式的动态映射，删除后不再依赖 mbridge 包和 megatron.bridge。
2. 内部化权重转换：新增 slime/backends/megatron_utils/hf_to_megatron/common.py（定义 SafetensorReader、merge_qkv、merge_gate_up、strip_mcore_wrappers）、qwen.py（qwen_hf_tensor、qwen_moe_hf_tensor、mimo_hf_tensor、minimax_m2_hf_tensor）和 deepseek.py（deepseek_hf_tensor），以声明式映射函数逐张量读取 HF safetensors，取代 mbridge 的类级映射配置。
3. 删除不再维护的模型与工具：移除 Gemma4 原生实现（slime_plugins/models/gemma4.py、gemma4_provider.py）及其转换器（hf_to_megatron/gemma4.py、megatron_to_hf/gemma4.py），删除 tools/preprocess_gpt_oss.py（GPT-OSS MXFP4 反量化工具），并移除 update_weight/hf_weight_iterator_bridge.py 桥接迭代器。
4. 新增原生 Qwen3.5-VL 支持：新增 slime_plugins/models/qwen3_5_vl.py 和 qwen3_5_vl_utils.py，在 Megatron GPTModel 基础上叠加 HF Vision Model，实现 packed sequence 下的 MRoPE、CP 分片视觉 token 注入，替代原先依赖桥接器的 VLM 路径。

5. 配套更新：同步更新 docs/en、docs/zh 和 examples/geo3k_vlm_multi_turn，并为新的转换器和 VLM 逻辑补充 / 调整 20 余个测试文件，覆盖 tests/test_*、tests/utils/ 等测试目录。

关键文件：

- slime/plugins/models/gemma4.py（模块 模型层；类别 source；类型 deletion；符号 Gemma4TransformerConfig, VNorm, init, forward）：Gemma4 原生 transformer 层（含 Gemma4Router、Gemma4MoELayer）被整体删除，是本次模型支持范围收缩的代表性文件。
- slime/plugins/models/qwen3_5_vl.py（模块 VLM 模型；类别 source；类型 core-logic；符号 Qwen3_5MultimodalRotaryEmbedding, forward, _load_vision_model, Qwen3_5VLModel）：新增的原生 Qwen3.5-VL 模型提供程序，以 Megatron GPTModel 叠加 HF ViT，标志着不依赖 mbridge 的 VLM 训练新路径。
- slime/plugins/mbridge/qwen3_5.py（模块 桥接层；类别 source；类型 deletion；符号 Qwen3_5Bridge, _get_text_config, _adjust_mapping_for_shared_weights, _supports_transformer_config_kwarg）：删除依赖 mbridge 的 Qwen3_5Bridge，它的功能被内部化到 hf_to_megatron/qwen.py 和原生 Qwen3.5-VL 模型替代。
- slime/plugins/megatron_bridge/glm4v_moe.py（模块 桥接层；类别 source；类型 deletion；符号 _thd_to_batch_seq, _batch_seq_to_thd, _gather_input_ids_from_cp, _select_local_image_embeds）：删除 GLM-4.6V 的 megatron.bridge 桥接实现，包含 THD/CP 转换和视觉模型封装，是本次清理的核心部分之一。
- slime/backends/megatron_utils/hf_to_megatron/qwen.py（模块 权重转换；类别 source；类型 core-logic；符号 _direct_tensor, _layer, _attention_tensor, qwen_hf_tensor）：新增的内部化 Qwen/Qwen MoE/MiMo/MiniMax-M2 HF 权重读取器，取代旧桥接器的核心转换逻辑。
- slime/backends/megatron_utils/hf_to_megatron/common.py（模块 权重转换；类别 source；类型 core-logic；符号 SafetensorReader, init, contains, get_tensor）：新增的 SafetensorReader 和形状合并工具（merge_qkv、merge_gate_up）是所有内部转换器的基础设施。

关键符号：qwen_hf_tensor, qwen_moe_hf_tensor, mimo_hf_tensor, minimax_m2_hf_tensor, deepseek_hf_tensor, merge_qkv, merge_gate_up, SafetensorReader.get_tensor, Qwen3_5VLModel.forward, Qwen3_5VLModel._inject_vision_embeddings, build_packed_mrope_position_ids, gather_packed_input_ids, get_packed_cp_local_indices

关键源码片段

slime/plugins/models/qwen3_5_vl.py

新增的原生 Qwen3.5-VL 模型提供程序，以 Megatron GPTModel 叠加 HF ViT，标志着不依赖 mbridge 的 VLM 训练新路径。

```
# 文件：slime_plugins/models/qwen3_5_vl.py
# 在 packed sequence 下把 HF 视觉模型的输出注入 Megatron embedding 序列。
```

```
def _inject_vision_embeddings(self, input_ids, full_input_ids, cu_seqlens, cp_group, pixel_values,
pixel_values_videos, image_grid_thw, video_grid_thw):
    # 先用语言模型的 embedding 生成基础 token embedding，再在视觉 token
    位置替换为视觉特征。
    embeddings = self.language_model.embedding(input_ids=input_ids, position_ids=None).clone()
    embeddings_bsh = embeddings.transpose(0, 1).contiguous()
    # 根据 THD 格式的 cu_seqlens 和 CP 分片策略，把本地 token 映射回完整 packed 序列的下标。
    local_indices = get_packed_cp_local_indices(
        cu_seqlens,
        cp_group.size() if cp_group is not None else 1,
        cp_group.rank() if cp_group is not None else 0,
        input_ids.device,
    )
    # 分别处理 image 和 video 两种模态，grid_thw 与 token 数必须匹配。
    for values, grids, token_id in (
        (pixel_values, image_grid_thw, self.image_token_id),
        (pixel_values_videos, video_grid_thw, self.video_token_id),
    ):
        if values is None:
            continue
        if grids is None:
            raise ValueError("Qwen3.5-VL pixel values require matching grid_thw")
        vision_output = self.model.visual(values.to(dtype=self.model.visual.dtype), grid_thw=grids)
        vision_embeddings = vision_output.pooler_output.to(device=embeddings.device, dtype=
        embeddings.dtype)
        # 视觉特征数量必须等于完整序列中对应 token 的数量。
        full_vision_positions = (full_input_ids[0] == token_id).nonzero(as_tuple=False).flatten()
        if full_vision_positions.numel() != vision_embeddings.shape[0]:
            raise ValueError(
                f"Qwen3.5-VL token/features mismatch: {full_vision_positions.numel()} tokens, "
                f"{vision_embeddings.shape[0]} features"
            )
        # 构建完整序列的 feature 下标，再用 CP 本地下标取出本 rank 需要替换的向量。
        feature_indices = torch.full((full_input_ids.shape[1],), -1, dtype=torch.long, device=input_
        ids.device)
        feature_indices[full_vision_positions] = torch.arange(vision_embeddings.shape[0], device=
        input_ids.device)
        local_feature_indices = feature_indices[local_indices]
        local_vision_mask = local_feature_indices >= 0
        # 严格校验：本地视觉掩码必须与 token id 匹配，否则说明 CP 布局或特征数与预期不一致。
        if not torch.equal(local_vision_mask, input_ids[0] == token_id):
            raise ValueError("Qwen3.5-VL CP token layout does not match its full packed
            sequence")
        embeddings_bsh[0, local_vision_mask] = vision_embeddings[local_feature_indices[local_
        vision_mask]]
    embeddings = embeddings_bsh.transpose(0, 1).contiguous()
    if self.config.sequence_parallel:
        embeddings = tensor_parallel.scatter_to_sequence_parallel_region(embeddings).
        contiguous()
```

```
    return embeddings
```

slime/backends/megatron_utils/hf_to_megatron/qwen.py

新增的内部化 Qwen/Qwen MoE/MiMo/MiniMax-M2 HF 权重读取器，取代旧桥接器的核心转换逻辑。

```
# 文件：slime/backends/megatron_utils/hf_to_megatron/qwen.py
# 内部化的 MoE 权重读取：把 Megatron 参数名解析为 HF safetensors 中的张量。
def qwen_moe_hf_tensor(name: str, reader: SafetensorReader, config) -> torch.Tensor:
    # 剥离 _extra_state 等 Megatron 内部包装后缀。
    name = strip_mcore_wrappers(name)
    # 顶层权重（embedding、layernorm、output_layer）直接映射。
    if (tensor := _direct_tensor(name, reader, config)) is not None:
        return tensor

    layer, rest = _layer(name)
    prefix = f"model.layers.{layer}"
    # 注意力部分（含 linear_qkv 的 qkv 融合）。
    if (tensor := _attention_tensor(rest, prefix, reader, config)) is not None:
        return tensor
    # MoE 专属部分：router、shared experts、per-expert 张量。
    if (tensor := _qwen_moe_layer_tensor(rest, prefix, reader)) is not None:
        return tensor
    # 回退到 dense 分支处理共享结构。
    return qwen_hf_tensor(name, reader, config)

# 解析 MoE 层的专家张量：
# "mlp.experts.linear_fc1.weight{expert_id}" -> HF 侧 gate_up_proj / down_proj。
def _qwen_moe_layer_tensor(rest: str, prefix: str, reader: SafetensorReader) -> torch.Tensor |
None:
    mapping = {
        "pre_mlp_layernorm.weight": "post_attention_layernorm.weight",
        "mlp.linear_fc1.layer_norm_weight": "post_attention_layernorm.weight",
        "mlp.router.weight": "mlp.gate.weight",
        "mlp.router.expert_bias": "mlp.gate.e_score_correction_bias",
        "mlp.shared_experts.linear_fc2.weight": "mlp.shared_expert.down_proj.weight",
        "mlp.shared_experts.gate_weight": "mlp.shared_expert_gate.weight",
    }
    if rest in mapping:
        return reader.get_tensor(f"{prefix}.{mapping[rest]}")
    if rest in {"mlp.shared_experts.linear_fc1.weight", "shared_experts.linear_fc1.weight"}:
        # HF 侧 gate_proj 与 up_proj 合并为 linear_fc1。
        return merge_gate_up(
            reader.get_tensor(f"{prefix}.mlp.shared_expert.gate_proj.weight"),
            reader.get_tensor(f"{prefix}.mlp.shared_expert.up_proj.weight"),
        )
    # 每个专家的线性层以 "weight{expert_id}" 结尾。
    match = re.fullmatch(r"mlp\.experts\.linear_fc([12])\.(weight|bias)(\d+)", rest)
    if match:
```

```
            projection, kind, expert = match.groups()
            if projection == "1":
                # 专家 fc1 由 gate_proj + up_proj 融合而成。
                return merge_gate_up(
                    reader.get_tensor(f"{prefix}.mlp.experts.{expert}.gate_proj.{kind}"),
                    reader.get_tensor(f"{prefix}.mlp.experts.{expert}.up_proj.{kind}"),
                )
            return reader.get_tensor(f"{prefix}.mlp.experts.{expert}.down_proj.{kind}")
        return None
```

评论区精华

该 PR 没有公开的 review 评论和讨论线程。PR body 中作者直接陈述了决策理由：移除实验性 Megatron-Bridge、内部化 mbridge 相关能力，并推荐需要更广模型覆盖的用户转向 radixark/miles。由于没有 reviewer 的交锋记录，本文档不臆造讨论内容。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 破坏性变更：移除 mbridge 和 megatron-bridge 依赖后，依赖 mbridge 外部包的旧配置和旧脚本将直接失效；删除 Gemma4 模型支持会对使用 Gemma4 的现有用户造成明显回归。
  2. 权重转换正确性：hf_to_megatron/qwen.py、deepseek.py 等新转换器是手工映射，若 HF 侧权重命名或张量布局变动（如 lm_head.weight 缺失、MTP 层结构变化），可能静默产生错误映射；merge_qkv 的 head 分组和 TP 切分逻辑风险较高。
  3. Qwen3.5-VL 新路径：Qwen3_5VLModel.forward 强制要求 packed sequences，且 _inject_vision_embeddings 对 local_vision_mask 与 input_ids[0] == token_id 的一致性做严格校验，如果 CP 布局或 vision features 数量不匹配会直接报错，属于新的运行时风险点。
  4. 文档与示例滞后：119 个文件的大改动可能遗漏个别引用（如 examples/geo3k_vlm_multi_turn 之外的示例脚本），需要关注 CI 和用户反馈。 - 影响：影响范围非常大：对外部依赖模型支持（Gemma4、GLM-4V-MoE、GPT-OSS）的用户是破坏性升级；对使用 Qwen3.5、Qwen3.5-VL、DeepSeek-V3.2、MiMo、MiniMax-M2 等模型的用户，转换路径从桥接器切换为内部直接转换，行为可能略有差异。同时，仓库去掉了两个外部运行时依赖（mbridge、megatron.bridge），降低安装复杂度和维护成本；团队不再需要为第三方桥接层打补丁。 - 风险标记：破坏性变更，权重转换重构，模型支持移除，外部依赖移除，新 VLM 路径

关联脉络

- PR #2220 Optimize update weight: 该 PR 修改了 update_weight/hf_weight_iterator_bridge.py，本 PR 删除该桥接迭代器并替换为内部转换器，权重更新链路需要同步适配。

- PR #2223 Fix --save-hf: --save-hf 的 HuggingFace 权重导出逻辑与本 PR 的权重转换重构直接相关，导出路径同样从桥接层迁移到内部实现。
- PR #2089 [2/n] Disaggregated rollout: disk-level delta weight sync: 磁盘级 delta 权重同步依赖权重转换 / 导出链路，mbridge 移除后这部分行为需要重新验证。