PR #2717 完整报告

radixark/miles

add DeepSeek-V4-Flash-0731 support and mxfp4->fp8 converter

合并时间：2026-08-24 14:42

原文链接：http://prhub.com.cn/radixark/miles/pull/2717

执行摘要

- 一句话：新增 DSV4-0731 支持与 MXFP4→FP8 转换器
- 推荐动作：值得精读，重点看 cast_e2m1fn_to_e4m3fn 的无损转换实现、e8m0fnu→float32 兼容处理，以及 rematerialize 对冻结参数的备份修复。若团队后续要接入其他 MXFP4 官方发布，可直接复用该转换器模式。

功能与动机

官方 deepseek-ai 发布 DeepSeek-V4-Flash-0731，使用 MXFP4 量化的路由专家，而 miles 现有的训练与 rollout 流水线围绕 blockwise FP8 设计（sgl-project FP8 重打包布局）。PR 目标是让官方权重无缝复用已验证的 FP8 下游（fp8_cast_bf16、torch_dist、fp8 rollout），因此需要新增一个无损的 MXFP4→FP8 转换阶段，并保持其余环节字节级一致。

实现拆解

1. 新增 MXFP4→FP8 转换器：在 tools/convert_mxfp4_to_fp8.py 中实现 cast_e2m1fn_to_e4m3fn，将打包 e2m1fn 专家权重（int8 + per-(1,32)-block ue8m0 scales）无损转为 e4m3fn + (128,128)-block e8m0 scales，并清理 config.json 中的 expert_dtype 字段；其他张量原样复制，权重映射同步更新。
2. 接入启动脚本：在 scripts/run_deepseek_v4.py 中为 DeepSeek-V4-Flash-0731 注册模型组织与 megatron 类型，新增 _MXFP4_MODEL_NAMES 和 fp8_name 属性，实现 _prepare_fp8 阶段并将其链入 full-train（哨兵跳过），rollout/bf16 阶段改读转换后的 FP8 目录。
3. 修复 FP8→BF16 转换兼容性：tools/fp8_cast_bf16.py 中把 try/except 收窄到 scale 查找，避免掩盖反量化内部错误；并接受 float8_e8m0fnu 缩放（先转 float32 再反量化）。
4. 修复参数重物化：miles/backends/megatron_utils/rematerialize_utils.py 中 _named_restore_extras 现在同时备份 requires_grad=False 的冻结参数（如 --moe-router-freeze-gate），因为它们没有 master weight 可重建。
5. 测试与文档：更新 tests/manual/launch_scripts/test_py_launch_scripts.py 禁用 prepare_fp8 默认入口并记录快照，同步更新 docs/models/deepseek/deepseek-v4-flash.md。

关键文件：

- tools/convert_mxfp4_to_fp8.py（模块 模型转换；类别 source；类型 core-logic；符号 cast_e2m1fn_to_e4m3fn, main）：新增的 MXFP4→FP8 无损转换器，是本 PR 的核心实现，直接决定官方权重能否进入既有 FP8 流水线。
- scripts/run_deepseek_v4.py（模块 启动脚本；类别 source；类型 core-logic；符号 fp8_name, _prepare_fp8, prepare_fp8）：launcher 接入新模型并新增 prepare-fp8 阶段，是整个接入流程的编排入口。
- tools/fp8_cast_bf16.py（模块 类型转换；类别 source；类型 core-logic）：修复 FP8→BF16 转换中对 float8_e8m0fnu 缩放的支持，是 MXFP4 转换产物能正确进入下游的关键配套。
- miles/backends/megatron_utils/rematerialize_utils.py（模块 参数恢复；类别 source；类型 core-logic）：修改影响所有使用 --rematerialize-param-from-master-weight 的训练路径，修复冻结参数无 master weight 可备份的问题。
- tests/manual/launch_scripts/test_py_launch_scripts.py（模块 启动测试；类别 test；类型 test-coverage）：将 prepare_fp8 加入默认禁用入口，防止无 MXFP4 源时误执行，并记录快照。
- docs/models/deepseek/deepseek-v4-flash.md（模块 模型文档；类别 docs；类型 documentation）：同步模型支持矩阵与快速开始示例，方便用户按新模型名启动。
- tests/snapshots/launch_scripts/py/scripts/run_deepseek_v4.py/prepare_fp8.txt（模块 测试快照；类别 docs；类型 documentation）：新增 prepare_fp8 阶段的快照，确保 launcher 生成命令的稳定性。
- tests/snapshots/launch_scripts/py/scripts/run_deepseek_v4.py/full_train.txt（模块 测试快照；类别 docs；类型 documentation）：full-train 快照需要反映新增 prepare-fp8 阶段。
- tests/snapshots/launch_scripts/py/scripts/run_deepseek_v4.py/train.txt（模块 测试快照；类别 docs；类型 documentation）：train 快照可能因 fp8_name 属性变化而更新。

关键符号：cast_e2m1fn_to_e4m3fn, main, fp8_name, _prepare_fp8, prepare_fp8, _named_restore_extras

关键源码片段

tools/convert_mxfp4_to_fp8.py

新增的 MXFP4→FP8 无损转换器，是本 PR 的核心实现，直接决定官方权重能否进入既有 FP8 流水线。

```
# tools/convert_mxfp4_to_fp8.py
# 将官方 DeepSeek-V4-Flash-0731 的 MXFP4 expert 权重无损转换为 FP8，
# 复用官方 inference/convert.py 的 cast_e2m1fn_to_e4m3fn 逻辑。

import torch

# MXFP4 使用 16 个枚举值（2 位指数 + 2 位尾数），对应 16 个可表示的浮点值。
FP4_TABLE = torch.tensor(
    [0.0, 0.5, 1.0, 1.5, 2.0, 3.0, 4.0, 6.0, -0.0, -0.5, -1.0, -1.5, -2.0, -3.0, -4.0, -6.0],
```

```
    dtype=torch.float32,
)


def cast_e2m1fn_to_e4m3fn(x: torch.Tensor, scale: torch.Tensor) -> tuple[torch.Tensor, torch.
Tensor]:
    """把每 32 列一个 ue8m0 scale 的打包 e2m1fn 权重转为 e4m3fn + (128,128) e8m0 scale。

    返回 (weight_fp8, tile_scale)，其中 tile_scale 以 float32 存储（与 sgl FP8 重打包约定一致）。
    """
    assert x.dtype == torch.int8 and x.ndim == 2
    out_dim, in_dim = x.size(0), x.size(1) * 2 # 每个 int8 打包了两个 fp4 元素
    fp8_block_size, fp4_block_size = 128, 32
    assert out_dim % fp8_block_size == 0 and in_dim % fp8_block_size == 0
    assert scale.shape == (out_dim, in_dim // fp4_block_size), f"{scale.shape=} {x.shape=}"

    table = FP4_TABLE.to(x.device)
    x = x.view(torch.uint8)
    # 低 4 位 / 高 4 位分别查表，还原出两个 fp4 值
    x = torch.stack([table[(x & 0x0F).long()], table[(x >> 4).long()]], dim=-1).reshape(out_dim, in_
    dim)

    max_offset = 2**6 # fp4 相对 scale 的最大偏移
    b_out, b_in = out_dim // fp8_block_size, in_dim // fp8_block_size
    x = x.view(b_out, fp8_block_size, b_in, fp8_block_size).transpose(1, 2)
    scale = scale.float().view(b_out, fp8_block_size, b_in, -1).transpose(1, 2).flatten(2)
    tile_scale = scale.amax(dim=-1, keepdim=True) / max_offset
    # 把每个 fp4 的 scale 重采样到 fp8 的 128 列 tile 上
    offset = (scale / tile_scale).unflatten(-1, (fp8_block_size, -1)).repeat_interleave(fp4_block_
    size, dim=-1)
    x = (x * offset).transpose(1, 2).reshape(out_dim, in_dim)
    # ue8m0 是 2 的幂次语义，用 float32 保存，兼容 attention/dense 的 float32 scale
    return x.to(torch.float8_e4m3fn), tile_scale.squeeze(-1).float()
```

评论区精华

PR 仅有一条实质性 issue 评论：Suhail 询问是否测试过 rollout 与 actor 之间的 logprob 差异。该问题关注 MXFP4→FP8 无损转换后 rollout 与 actor 的数值一致性，PR 中未见作者回复，属于待确认事项。另有两位 reviewer（Zhichenzzz、yushengsu-thu）给予 APPROVED。

- rollout 与 actor 的 logprob 差异 (question): PR 中未见作者回复，该问题未在 review 中解决，属于待确认的量化一致性风险。

风险与影响

- 风险：
  1. 转换器正确性依赖 FP4_TABLE 与 tile 重采样逻辑，若官方权重布局变化需重新对齐；当前缺少针对转换数值正确性的自动化断言（仅手动 GPU 往返验证）。

2. fp8_cast_bf16.py 的修复涉及 Triton 反量化的异常处理，需在更多 FP8 模型上回归验证。
3. rematerialize_utils.py 的改动影响所有使用 --rematerialize-param-from-master-weight 的训练，可能增加 pinned 内存占用（冻结参数备份），需关注显存 / 内存压力。
4. 提交 0b8a4f4/09f0154 显示 clear_memory 的 RSS 高水位问题仍未解决（malloc_trim 实测无效已回滚），在 DSV4 大规模 checkpoint 加载时仍可能触发 OOM。
5. 新模型仅完成 8 节点 GB300 bringup，尚未完成端到端训练验证。 - 影响：用户与团队：可一键启动官方 DeepSeek-V4-Flash-0731 训练，复用已验证的 FP8 下游路径，降低新模型接入成本。系统：新增转换工具和 launcher 阶段，对既有 DeepSeek-V4-Flash-FP8 路径无行为变化（fp8_name 对非 MXFP4 模型回退到原模型名）。测试：快照与手动测试覆盖 prepare_fp8 入口，但缺少转换器数值正确性的自动化断言。 - 风险标记：量化转换正确性依赖，核心训练路径变更，内存问题未解决，新模型验证不足

关联脉络

- PR #2673 Bump Megatron-LM to miles-main-20260819 (latest NVIDIA dev): Megatron-LM 升级为 DSV4 系列训练提供基础，且同仓库近期涉及 megatron 适配，可能影响 deepseek-v4-flash 模型类型。
- PR #2682 fix: resume from the checkpoint step in bridge mode: 桥接模式恢复训练的行为与 DSV4 长训练相关，本 PR 调整了 launcher 阶段顺序。
- PR #2716 fix: align MLA RoPE types with model configs: 同为模型脚本与配置修复，且修改了测试快照与启动脚本，属于同一模型支持领域。