PR #33115 完整报告

sgl-project/sglang

[ModelOpt FP4] Support online MoE weight quantization

合并时间：2026-08-07 02:01

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33115

执行摘要

- 一句话：ModelOpt FP4 在线 MoE 权重量化，覆盖草稿与 FP8 源
- 推荐动作：值得精读。重点看三处设计： 1) ModelOptFp4Config 的契约拆分与 get_quant_method() 非序列化路由，理解 " 只量化 MoE 专家 " 的机制； 2) _ModelOptFp4OnlineConfig 薄适配器如何通过类属性切换契约复用整套转换管线； 3) _resolve_explicit_draft_quant_config() 对序列化 MTP checkpoint 排除专家的处理。测试方面 test_modelopt_nvfp4.py 的 activation 契约测试与 test_modelopt_loader.py 的 loader 选择测试是理解边界行为的最佳入口。

功能与动机

PR body 明确指出动机是 "Add online ModelOpt FP4 quantization for eligible MoE expert weights from BF16, FP16, or FP8 checkpoints"，让用户无需预先序列化的 NVFP4 checkpoint 即可在加载阶段拿到 NVFP4 量化的 MoE 权重，降低显存占用。同时需要修正 #31382 引入的 activation 契约问题：原先 CuTe DSL + FlashInfer A2A 或 DeepEP 在 nvfp4_online 下使用 per-tensor activation scaling 的行为，现统一路由至 modelopt_fp4，而 nvfp4_online 独占 per-token 契约。作者在评论中说明这是为 #28354 的 CuTe DSL per-token NVFP4 集成铺路。

实现拆解

1. 量化契约拆分：python/sglang/srt/layers/quantization/modelopt_quant.py 的 ModelOptFp4Config 新增 for_online_weight_quantization() 类方法作为在线转换入口；from_config() 增加 quant_method == "fp8" 分支，将 FP8 源 checkpoint 路由到 nvfp4_online 的适配器工厂 make_modelopt_fp4_online_config_from_fp8()；__init__ 对非序列化场景强制 use_per_token_activation = False（显式传 True 抛 ValueError），把在线 per-token 语义完全收归 nvfp4_online；get_quant_method() 对非序列化配置只对 FusedMoE 返回 ModelOptNvFp4FusedMoEMethod，dense 层返回 UnquantizedLinearMethod，从机制上保证 " 只量化 MoE 专家 "。
2. 在线转换管线适配：python/sglang/srt/layers/quantization/nvfp4_online.py 新增 _ModelOptFp4OnlineConfig 子类与两个工厂函数，通过类属性 is_nvfp4_online = False、_use_per_token_activation = False 切换契约，复用既有 FP8 反量化、权重 /scale staging 与 NVFP4 量化逻辑；ModelOptNvFp4OnlineFusedMoEMethod 将 per-token 后端限制收窄为仅 flashinfer_trtllm / flashinfer_trtllm_routed，per-tensor 场景不再受后端

限制。

3. 加载器路由：python/sglang/srt/model_loader/loader.py 的 get_model_loader() 新增 modelopt_fp4_online 判定——未量化模型且未显式请求 ModelOpt checkpoint/export 工作流时改用 DefaultModelLoader 走普通逐层加载（配合 prepare_weight_loader 包装做在线转换）；load_weights_and_postprocess() 将 FP4 精确数学环境变量（FLASHINFER_DISABLE_FP4_QUANT_FAST_MATH、FLASHINFER_NVFP4_4OVER6 等）的作用范围从 is_nvfp4_online 扩展到 is_modelopt_fp4_online。
4. 草稿模型路由：python/sglang/srt/model_loader/weight_utils.py 新增 _resolve_explicit_draft_quant_config()，对显式指定草稿量化且序列化 Qwen3.5 MTP checkpoint 排除 MTP experts 的场景切换到在线转换；python/sglang/srt/configs/model_config.py 记录 is_draft_quantization_explicit 并在 _verify_quantization() 中允许 modelopt_fp4 兼容字面 FP8 checkpoint；python/sglang/srt/server_args.py 增加无 CLI 表面的内部字段 _speculative_draft_quantization_explicitly_set 跟踪显式指定并保证 asdict round-trip；python/sglang/srt/models/qwen3_5_mtp.py 对序列化 checkpoint 禁用量化、非序列化 modelopt_fp4 保留加载期转换。
5. 测试与文档：test_modelopt_loader.py 新增 TestModelOptFp4LoaderSelection 覆盖 loader 选择与草稿排除路由；test_modelopt_nvfp4.py 锁定 activation 契约与 input_scale 回退为 1.0 的行为；test_server_args.py 验证显式标记 asdict round-trip；docs 更新量化与环境变量说明。

关键文件：

- python/sglang/srt/layers/quantization/modelopt_quant.py（模块 量化配置；类别 source；类型 data-contract；符号 for_online_weight_quantization, from_config, get_quant_method, prepare_weight_loader）：核心数据契约变更：ModelOptFp4Config 拆分序列化与在线两条路径，新增 for_online_weight_quantization 入口，from_config 支持 FP8 源路由，get_quant_method 对非序列化场景只量化 MoE 层。
- python/sglang/srt/layers/quantization/nvfp4_online.py（模块 在线量化；类别 source；类型 dependency-wiring；符号 _ModelOptFp4OnlineConfig, make_modelopt_fp4_online_config, make_modelopt_fp4_online_config_from_fp8, ModelOptNvFp4OnlineFusedMoEMethod）：新增 _ModelOptFp4OnlineConfig 适配器与两个工厂函数，通过类属性切换 per-tensor/per-token 契约，复用 FP8 反量化与 NVFP4 量化逻辑。
- python/sglang/srt/model_loader/weight_utils.py（模块 权重加载；类别 source；类型 data-contract；符号 _resolve_explicit_draft_quant_config, get_quant_config）：新增 _resolve_explicit_draft_quant_config，处理显式草稿量化下序列化 Qwen3.5 MTP checkpoint 排除专家的在线转换路由。
- python/sglang/srt/model_loader/loader.py（模块 加载器；类别 source；类型 core-logic；符号 get_model_loader, load_weights_and_postprocess）：get_model_loader 增加 modelopt_fp4_online 判定，未量化模型走 DefaultModelLoader；load_weights_and_postprocess 扩展 FP4 精确数学环境变量范围。

- python/sglang/srt/configs/model_config.py（模块 模型配置；类别 source；类型 data-contract；符号 ModelConfig, _verify_quantization）：新增 is_draft_quantization_explicit 字段；_verify_quantization 允许 modelopt_fp4 兼容字面 FP8 checkpoint，影响草稿量化继承行为。
- python/sglang/srt/models/qwen3_5_mtp.py（模块 MTP 模型；类别 source；类型 bugfix；符号 Qwen3_5MTPForCausalLM）：序列化 Qwen3.5 ModelOpt checkpoint 禁用量化（MTP 专家为 BF16），非序列化 modelopt_fp4 保留加载期转换。
- test/registered/unit/model_loader/test_modelopt_loader.py（模块 加载器测试；类别 test；类型 test-coverage；符号 TestModelOptFp4LoaderSelection, test_inherited_draft_modelopt_fp4_accepts_fp8_checkpoint, test_draft_modelopt_fp4_uses_checkpoint_exclusions, test_unquantized_modelopt_fp4_preserves_modelopt_workflows）：新增 TestModelOptFp4LoaderSelection 覆盖草稿排除路由与 loader 选择，验证在线转换走 DefaultModelLoader 且保留 ModelOpt 工作流路径。
- test/registered/unit/layers/quantization/test_modelopt_nvfp4.py（模块 量化测试；类别 test；类型 test-coverage；符号 test_missing_input_scale_defaults_to_one_and_checkpoint_overwrites, test_modelopt_fp4_per_token_activation_contract）：锁定 activation 契约（per-token/per-tensor 分离）与 input_scale 回退 1.0 行为，是契约设计的直接验证。

关键符号：for_online_weight_quantization, make_modelopt_fp4_online_config, make_modelopt_fp4_online_config_from_fp8, _resolve_explicit_draft_quant_config, get_quant_method, prepare_weight_loader, _uses_serialized_fp8_source, get_model_loader

关键源码片段

python/sglang/srt/layers/quantization/nvfp4_online.py

新增 _ModelOptFp4OnlineConfig 适配器与两个工厂函数，通过类属性切换 per-tensor/per-token 契约，复用 FP8 反量化与 NVFP4 量化逻辑。

```
class _ModelOptFp4OnlineConfig(NvFp4OnlineConfig):
    """`modelopt_fp4` 的在线适配器：per-tensor activation scale 契约。

    复用 NvFp4OnlineConfig 的加载期转换管线（FP8 反量化、权重/scale staging、
    NVFP4 量化），只切换 activation 契约与对外名称。
    """

    is_nvfp4_online = False # 不触发 loader 里的 nvfp4_online 专用分支
    _use_per_token_activation = False # per-tensor 契约

    @classmethod
    def get_name(cls) -> str:
        return "modelopt_fp4"

    @classmethod
    def get_supported_act_dtypes(cls) -> List[torch.dtype]:
```

```
        # 适配器需要放宽到 FP8 源：nvfp4_online 本身只支持 BF16/FP16。
        return ModelOptFp4Config.get_supported_act_dtypes()

    @classmethod
    def get_min_capability(cls) -> int:
        return ModelOptFp4Config.get_min_capability()


def make_modelopt_fp4_online_config(packed_modules_mapping=None):
    return _ModelOptFp4OnlineConfig(packed_modules_mapping=packed_modules_mapping)


def make_modelopt_fp4_online_config_from_fp8(config: Dict[str, Any]):
    # 保留 FP8 源反量化：is_checkpoint_fp8_serialized = True 会启用
    # ModelOptNvFp4OnlineFusedMoEMethod 的 _dequantize_fp8_weight 包装。
    return _ModelOptFp4OnlineConfig.from_config(config)
```

评论区精华

Review 的核心交锋集中在 YAMY1234 提出的两个契约问题，均被作者 zianglih 以「有意设计」回应并解决：

1. FP8 兼容是否覆盖继承式草稿量化：YAMY1234 指出 modelopt_fp4 兼容 fp8 后，--quantization modelopt_fp4 下报告字面 quant_method: "fp8" 的草稿会从改动前的 FP8 运行变为继承 modelopt_fp4 的在线 NVFP4 重量化，「同一命令行、不同草稿数值」；他建议若想保守可 gate 在 is_draft_quantization_explicit 上。zianglih 回应这是有意覆盖继承路径，test_inherited_draft_modelopt_fp4_accepts_fp8_checkpoint 锁定该路由，PR 描述已注明数值变化，未声称模型级精度验证。
2. input_scale 回退契约：YAMY1234 问在线 modelopt_fp4 跳过 checkpoint input_scale 后 w13_input_scale 是否长期保持 1.0。zianglih 说明 FP8 activation scale 描述的是 FP8 激活量化而非重量化后的 NVFP4 输入，在线 modelopt_fp4 按契约使用静态 per-tensor FP32 activation scale 默认 1.0，序列化 checkpoint 的 scale 张量会覆盖该回退；nvfp4_online 才是在线 per-token 接口。 另外，作者在 CI 排查中投入较多：对失败的 Llama-2 + EAGLE speculative-prefill 测试做了 A/B 复现（B200 对比精确 base 与合成 merge），定位为 move_accept_tokens_to_target_kvcache() 中既有的间歇性 KV-cache bug，与本 PR 无关。

- FP8 兼容是否覆盖继承式草稿量化 (correctness): zianglih 确认覆盖继承路径是有意设计，目标是 target + draft 统一契约；test_inherited_draft_modelopt_fp4_accepts_fp8_checkpoint 锁定该路由，PR 描述已注明数值变化，未声称模型级精度验证。
- 在线 modelopt_fp4 的 input_scale 回退契约 (correctness): zianglih 说明 FP8 activation scale 描述的是 FP8 激活量化，不适用于重量化后的 NVFP4 输入；在线 modelopt_fp4 按契约使用静态 per-tensor FP32 activation scale 默认 1.0，序列化 checkpoint 的 scale 张量会覆盖该回退；test_missing_input_scale_defaults_to_one_and_checkpoint_overwrites 覆盖此行为。

- NV CI 失败是否为回归 (question): 作者通过 A/B 实验（B200，保持 EAGLE、FlashInfer decode、CUDA graphs）将其定位为既有缺陷，非 ModelOpt FP4 回归；后续重跑通过。

风险与影响

- 风险：
  1. 草稿模型数值行为变化：compatible_quantization_methods 允许 modelopt_fp4 兼容字面 fp8 后，继承式草稿量化会把原本保持 FP8 的专家反量化到 BF16 再量化为 NVFP4，同一命令行下草稿输出数值与改动前不同（python/sglang/srt/configs/model_config.py 的 _verify_quantization）。
  2. activation scale 契约风险：在线 modelopt_fp4 的 w13_input_scale 固定回退 1.0，若未来有 checkpoint 携带 per-tensor NVFP4 activation scale 而未被识别，精度会被静默降低；该契约目前只有单元测试覆盖，无模型级精度验证。
  3. 核心加载路径变更：modelopt_fp4 未量化模型从 ModelOptModelLoader 切到 DefaultModelLoader，loader.py 的 load_weights_and_postprocess() 还需临时设置多个 FlashInfer FP4 环境变量，若环境变量恢复时机出错会影响服务期数值行为。
  4. FP8 源重量化依赖 BF16 中间精度：FP8 -> BF16 -> NVFP4 两级转换有额外精度损失，PR 未提供精度对比数据。
  5. 既有间歇性 bug：CI 观测到的 speculative-prefill KV-cache 越界问题（spec_utils.py 的 move_accept_tokens_to_target_kvcache()）虽经排查与本 PR 无关，但可能间歇影响相关投机解码测试。 - 影响：用户侧：BF16/FP16/FP8 MoE checkpoint 可直接用 --quantization modelopt_fp4 在线获得 NVFP4 MoE 权重，无需预量化文件；dense 层保持源精度，显存占用降低，但加载阶段会增加转换计算开销。系统侧：量化配置契约被重新划分为两条清晰路径（modelopt_fp4 为 per-tensor + 序列化，nvfp4_online 为 per-token），影响所有现有 NVFP4 用户的后端选择语义，特别是草稿 / 投机解码场景（Qwen3.5 MTP、DeepSeek、GLM 等构造器保持 opt-out）。团队侧：该 PR 为后续 #28354 的 CuTe DSL per-token NVFP4 集成确立契约基础，后续内核工作需遵循此划分。 - 风险标记：草稿量化数值行为变更，activation scale 契约变更，核心加载路径变更，FP8 源重量化依赖 BF16 中间精度，缺少模型级精度验证

关联脉络

- PR #26083 Initial online NVFP4 implementation: 本 PR 的在线转换管线（get_online_weight_loader、FP8 反量化、NVFP4 量化）复用了该 PR 引入的 nvfp4_online 实现。
- PR #31382 NVFP4 activation contract: 本 PR 修正了 #31382 引入的 activation 契约混淆：原先 nvfp4_online 下 CuTe DSL/DeepEP 的 per-tensor 行为现在统一路由至 modelopt_fp4。
- PR #28354 CuTe DSL integration for nvfp4_online: 本 PR 的契约拆分为该 PR 的 per-token CuTe DSL NVFP4 集成铺路，作者明确表示将基于本 PR 叠加后续实现。