PR #34883 完整报告

sgl-project/sglang

[Kimi-K3] Use explicit SiTU activation for MegaMoE

合并时间：2026-08-15 16:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34883

执行摘要

- 一句话：Kimi-K3 MegaMoE 显式 SiTU 激活并新增 B300 端到端测试
- 推荐动作：值得精读： 1) kimi_k3.py 中从哨兵 hack 到显式 API 的演进； 2) mxfp4.py 中多后端权重布局的条件分支设计； 3) e2e 测试如何配置 DSPARK + MegaMoE 并设置 SGLANG_OPT_DEEPGEMM_MEGA_MOE_NUM_MAX_TOKENS_PER_RANK。适合想了解 DeepGEMM MegaMoE 集成与 MXFP4 权重准备的工程师。

功能与动机

PR body 要求移除 sentinel clamp、直接调用 situ、以及在 runner 为 flashinfer_mxfp4 时仍为 MegaMoE 准备 MXFP4 权重；原 mock 测试无法覆盖真实权重加载、warmup 与 GSM8K 路径。旧实现的 0.03125 哨兵值依赖 swiglu clamp 不会使用该值的隐式约定，既脆弱又难维护，DeepGEMM PR #78 提供显式 situ 激活后可直接表达意图。

实现拆解

1. 依赖升级：在 python/pyproject.toml 将 sgl-deep-gemm 从 0.1.5.post2 升至 0.1.5.post3，该版本包含 DeepGEMM PR #78 的显式 situ 激活支持。
2. 模型调用改造：删除 python/sglang/srt/models/kimi_k3.py 中的 _K3_MEGA_SITU_SENTINEL_CLAMP 常量，在 _forward_mega_experts 中将 fp8_fp4_mega_moe 调用从 activation="swiglu" + activation_clamp 改为 activation="situ"，并同步更新注释与断言文案，明确 beta=4.0/linear_beta=25.0 由内核烘焙。
3. 权重准备修正：在 python/sglang/srt/layers/quantization/mxfp4.py 新增 self.use_mega_moe 标志，将 create_weights 和 process_weights_after_loading 中原本仅判断 use_marlin/use_deep_gemm 的条件改为与 use_mega_moe 组合判断：MegaMoE 时跳过 marlin 的 padding 与 deinterleave，直接走 DeepGEMM 的 scale 布局转换，并调用 transform_weights_for_mega_moe 构建 mega_l1_weights/mega_l2_weights，将参数 repoint 到 mega 布局以回收约 1.9GB/ 层 /rank 显存。
4. 端到端测试：在 test/registered/models_e2e/test_kimi_k3_b300.py 新增 TestKimiK3B300MegaMoE 类，以 TP8/EP8/DCP8 启动 megamoe 服务，配置 DSPARK 投机与 SGLANG_OPT_DEEPGEMM_MEGA_MOE_NUM_MAX_TOKENS_PER_RANK=8320 环境变量，GSM8K 阈值 0.95，并将 register_cuda_ci 的 est_time 从 900 上调至

1200 秒。

关键文件：

- test/registered/models_e2e/test_kimi_k3_b300.py（模块 端到端测试；类别 test；类型 test-coverage；符号 TestKimiK3B300MegaMoE, setUpClass, tearDownClass）：新增 TP8/EP8/DCP8 MegaMoE 端到端测试类，真实拉起 8 卡 B300 服务，覆盖权重加载、warmup 与 GSM8K 精度路径，替代原 mock 单测，并将 CI 预估时间上调至 1200 秒。
- python/sglang/srt/models/kimi_k3.py（模块 模型实现；类别 source；类型 data-contract；符号 _forward_mega_experts）：核心调用变更：删除 activation_clamp 哨兵常量，将 fp8_fp4_mega_moe 的激活从 swiglu+sentinel 改为显式 situ，并更新 MegaMoE 相关注释与断言文案。
- python/sglang/srt/layers/quantization/mxfp4.py（模块 量化层；类别 source；类型 core-logic；符号 Mxfp4MoEMethod.create_weights, Mxfp4MoEMethod.process_weights_after_loading）：权重准备逻辑修正：新增 use_mega_moe 标志，使 MegaMoE 在 flashinfer_mxfp4 runner 下也走 DeepGEMM 布局准备，并复用 transform_weights_for_mega_moe 构建 mega 权重。
- python/pyproject.toml（模块 依赖配置；类别 config；类型 configuration）：升级 sgl-deep-gemm 到 0.1.5.post3，引入 DeepGEMM PR #78 的显式 situ 激活支持，是本 PR 功能实现的前置依赖。

关键符号：_forward_mega_experts, Mxfp4MoEMethod.create_weights, Mxfp4MoEMethod.process_weights_after_loading, TestKimiK3B300MegaMoE.setUpClass

关键源码片段

test/registered/models_e2e/test_kimi_k3_b300.py

新增 TP8/EP8/DCP8 MegaMoE 端到端测试类，真实拉起 8 卡 B300 服务，覆盖权重加载、warmup 与 GSM8K 精度路径，替代原 mock 单测，并将 CI 预估时间上调至 1200 秒。

```
# Kimi-K3 B300 端到端测试：新增 MegaMoE recipe
# 相比原来的 mock 单测，这里真实拉起 8 卡 B300 服务，覆盖权重加载、
# warmup 与 GSM8K 精度路径，DSPARK 草稿模型参与投机解码。
class TestKimiK3B300MegaMoE(GSM8KMixin, CustomTestCase):
    gsm8k_score_threshold = 0.95
    gsm8k_num_examples = 200

    @classmethod
    def setUpClass(cls):
        cls.model = MODEL_PATH
        cls.base_url = MEGAMOE_URL # 独立端口 30000，避免与其它 recipe 冲突
        cls.process = popen_launch_server(
            cls.model,
            cls.base_url,
            timeout=SERVER_LAUNCH_TIMEOUT,
            other_args=[
                "--trust-remote-code",
```

```
                "--tp-size", "8",
                "--moe-a2a-backend", "megamoe",
                "--ep", "8",
                "--dcp-size", "8",
                "--mem-fraction-static", "0.85",
                "--reasoning-parser", "kimi_k3",
                "--tool-call-parser", "kimi_k3",
                "--mamba-full-memory-ratio", "5.13",
                "--speculative-algorithm", "DSPARK",
                "--speculative-draft-model-path", DSPARK_DRAFT_MODEL,
                "--speculative-dspark-block-size", "7",
                "--enable-linear-replayssm-spec",
            ],
            env=MEGAMOE_ENV, # SGLANG_OPT_DEEPGEMM_MEGA_MOE_NUM_MAX_TOKENS_
            PER_RANK=8320
        )

    @classmethod
    def tearDownClass(cls):
        _stop_server(getattr(cls, "process", None))
```

python/sglang/srt/models/kimi_k3.py

核心调用变更：删除 activation_clamp 哨兵常量，将 fp8_fp4_mega_moe 的激活从 swiglu+sentinel 改为显式 situ，并更新 MegaMoE 相关注释与断言文案。

```
# Kimi-K3 MegaMoE 前向：显式 SiTU 激活
# 旧版本用 activation="swiglu" + activation_clamp=0.03125 哨兵值触发
# DeepGEMM 内核中的 K3 SiTU 分支（0.03125 恰好是 2^-5，且合法 swiglu clamp
# 不会用到它）。升级到 sgl-deep-gemm==0.1.5.post3 后，内核新增显式
# "situ" 激活参数，这里直接传入，彻底移除哨兵 hack，避免与真实 clamp 冲突。
def _forward_mega_experts(self, routed_input, ...):
    # ... 前序 topk / pre_dispatch 逻辑 ...
    mega_moe_pre_dispatch(...)
    # 至少保留一行，保证 tvm-ffi 绑定拿到非空 data_ptr
    y = torch.empty(
        (max(num_tokens, 1), self.moe_hidden_size),
        dtype=torch.bfloat16,
        device=routed_input.device,
    )
    deep_gemm.fp8_fp4_mega_moe(
        y,
        self.experts.mega_l1_weights,
        self.experts.mega_l2_weights,
        buf,
        recipe=(1, 1, 32),
        activation="situ", # 显式选择 K3 SiTU 激活；beta=4.0 / linear_beta=25.0 已烘焙进内核
        fast_math=True,
    )
    y = y[:num_tokens]
```

```
    # ... 后续 scaling 与返回 ...
    return y
```

python/sglang/srt/layers/quantization/mxfp4.py

权重准备逻辑修正：新增 use_mega_moe 标志，使 MegaMoE 在 flashinfer_mxfp4 runner 下也走 DeepGEMM 布局准备，并复用 transform_weights_for_mega_moe 构建 mega 权重。

```
  # Mxfp4MoEMethod：当启用 MegaMoE（use_mega_moe=True）时，即使常规
  # MoE runner 是 flashinfer_mxfp4，也需要按 DeepGEMM 的布局准备权重，
  # 因为 MegaMoE 直接调用 deep_gemm 而不经过 runner 分发。
  def process_weights_after_loading(self, layer):
      # 常规 runner 走 marlin 时正常 marlin 准备；MegaMoE 不走 marlin
      if self.use_marlin and not self.use_mega_moe:
          # ... marlin 专属 deinterleave / prepare 逻辑 ...
          layer._mxfp4_backend = "marlin"
          return

      # DeepGEMM（无论常规 deep_gemm runner 还是 MegaMoE）需要：
      # 1) 权重转 int8 视图（packed e2m1）
      # 2) scale 转 TMA 对齐的 MN-major 布局
      # 3) MegaMoE 额外调用 transform_weights_for_mega_moe 得到
      # interleaved/UTCCP 布局，并回收连续布局的原始显存
      if self.use_deep_gemm or self.use_mega_moe:
          from deep_gemm import transform_sf_into_required_layout
          layer.w13_weight.data = layer.w13_weight.data.view(torch.int8)
          layer.w2_weight.data = layer.w2_weight.data.view(torch.int8)
          for scale_name, weight in (("w13_weight_scale", layer.w13_weight),
                                     ("w2_weight_scale", layer.w2_weight)):
              scale = getattr(layer, scale_name)
              num_experts, n, _ = scale.data.shape
              k = weight.shape[2] * 2
              scale_f32 = scale.data.view(torch.float8_e8m0fnu).to(torch.float32)
              scale.data = transform_sf_into_required_layout(
                  scale_f32, mn=n, k=k, recipe=(1, 32),
                  num_groups=num_experts, disable_ue8m0_cast=False,
              )
          if self.use_mega_moe:
              from deep_gemm import transform_weights_for_mega_moe
              l1_pair, l2_pair = transform_weights_for_mega_moe(
                  (layer.w13_weight.data, layer.w13_weight_scale.data),
                  (layer.w2_weight.data, layer.w2_weight_scale.data),
              )
              layer.mega_l1_weights = l1_pair
              layer.mega_l2_weights = l2_pair
              # 将参数重新指向 mega 布局，连续布局原值变死代码
              # 每层每 rank 可回收约 1.9GB（92 层双份专家权重会 OOM）
              layer.w13_weight.data = l1_pair[0]
              layer.w13_weight_scale.data = l1_pair[1]
              layer.w2_weight.data = l2_pair[0]
```

```
            layer.w2_weight_scale.data = l2_pair[1]
            layer._mega_moe_weights_built = True
        layer._mxfp4_backend = "deep_gemm"
        return
    # ... flashinfer cutlass 各 SM 分支 ...
```

评论区精华

本 PR 无正式 review 评论；评论区仅 Fridge003 的三次 /rerun-test 指令与 CI 回执，第一次失败后在 8x B300 上通过（965.5s，MegaMoE 子测试 GSM8K 0.985）。设计权衡沉淀在代码注释中：显式 situ 替代哨兵、mega 权重 repoint 回收 1.9GB/ 层 /rank、MegaMoE 绕过 runner 分发需独立 DeepGEMM 权重准备。

- 暂无高价值评论线程

风险与影响

- 风险：mxfp4.py 的条件分支改动影响所有使用 Mxfp4MoEMethod 的 MoE 场景，但 use_mega_moe 仅在 --moe-a2a-backend megamoe 时置真，默认路径行为不变，仍需留意与 use_marlin/use_deep_gemm 的组合是否有遗漏分支。sgl-deep-gemm 为 patch 版本升级，仍有兼容性回归风险，尤其是已在生产使用 0.1.5.post2 的 MegaMoE 用户。register_cuda_ci(est_time=1200) 使 base-c 阶段耗时增加约 5 分钟，可能影响 B300 CI 队列容量。行为变更局限在 Kimi-K3 模型路径，对其它模型无直接风险。
- 影响：改善 Kimi-K3 在 Blackwell B300 上的 MegaMoE 使用体验与可维护性，扩大端到端测试覆盖；对其它模型路径无直接影响，但 mxfp4.py 条件分支变化影响所有 Mxfp4MoEMethod 使用者，CI 时间增加约 5 分钟，团队需同步升级 sgl-deep-gemm 版本。
- 风险标记：依赖版本升级，MoE 核心路径变更，CI 测试时长增加，量化层多后端分支调整

关联脉络

- PR #34844 [Spec] Support MegaMoE for DSpark under dp attention: 同为 MegaMoE 支持线，为 DSpark 在 DP attention 下接入 MegaMoE 后端，与本 PR 在 Kimi-K3 DSPARK 投机场景互补。
- PR #34789 [MoE] Route every trtllm-gen MoE call site through one PDL guard: 改动集中在 MoE 与 mxfp4.py 路径，统一 trtllm MoE PDL 开关，与本 PR 的 MXFP4/MoE 后端分支调整相关。