PR #37279 完整报告

sgl-project/sglang

Bump sgl-deep-gemm to 0.1.7

合并时间：2026-09-01 08:16

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/37279

执行摘要

- 一句话：升级 sgl-deep-gemm 至 0.1.7，MegaMoE 改用 mma_type API 支持 MXFP4
- 推荐动作：值得精读，尤其是两个设计点：一是对称 buffer 缓存 key 纳入 mma_type 的隔离思路，避免了跨类型 buffer 串用的隐性 bug；二是 _interleave_mega_moe_gate_up 同时支持 gran=8 连续交错与 gran=16 even/odd 交错两种布局，展示了如何在同一函数内兼容不同上游内核的权重排布。对依赖升级类 PR，这种“版本钉版 + 调用点迁移 + 行为回归测试”的配套做法也值得借鉴。

功能与动机

PR body 的动机只有一句 "To include nvfp4 megamoe support"。上游 sgl-deep-gemm 0.1.7 新增了 NVFP4 MegaMoE 能力，同时把原来依赖进程级环境变量（DG_USE_FP4_ACTS、DG_USE_MXF4_KIND）和 use_fp8_dispatch 布尔开关的隐式选择方式，替换为在 get_symm_buffer_for_mega_moe、mega_moe_pre_dispatch、transform_weights_for_mega_moe 等 API 上显式传入 mma_type。SGLang 侧若不跟进适配，将无法使用新版内核的 mxf4xmxf4 路径，也无法获得 NVFP4 支持。

实现拆解

变更入口是依赖钉版：python/pyproject.toml 与 docker/Dockerfile 中将 sgl-deep-gemm 从 0.1.6 同步升至 0.1.7，保证后续 API 适配有确定的库版本前提。

1. 引入统一的 mma_type 决策函数：python/sglang/srt/layers/moe/mega_moe.py 新增 _mega_moe_mma_type()，通过 sglang.srt.runtime_context.get_exec() 读取 exec.moe.enable_w4a4_mxfp4_megamoe，返回 mxf4xmxf4 或 fp8xfp4。相比原先读环境变量，这里改为在运行时从 args 解析结果取配置，语义更直接。
2. 对称 buffer 缓存按 mma_type 隔离：_get_mega_moe_symm_buffer 的缓存 key 加入 mma_type，并把 deep_gemm.get_symm_buffer_for_mega_moe 的 use_fp8_dispatch=True 参数替换为 mma_type=mma_type。原因是两种 MMA 类型的 pre_dispatch 写布局不同，buffer 不能跨类型复用，否则会造成数据污染。
3. 前向 dispatch 分支改写：_run_mega_routed 中删除 os.getenv("DG_USE_FP4_ACTS") 判断，改为按 _mega_moe_mma_type() 分支：mxf4xmxf4 走 deep_gemm.mega_moe_pre_dispatch（传 mma_type，不再传 use_fp4_acts）；fp8xfp4 继续走 sglang.kernels.ops.attention.dsv4 的 JIT mega_moe_pre_dispatch。注

释说明 JIT 实现只支持 FP8，FP4 必须走 DeepGEMM。

4. L1 权重交错布局双模式：_interleave_mega_moe_gate_up 扩展支持 gran=16 的 even/odd 交错（MXFP4 激活），保留 gran=8 连续交错（FP8 激活）；_interleave_mega_moe_l1_weights 与 build_mega_moe_experts_weights 透传 mma_type。python/sglang/srt/layers/quantization/mxfp4.py 的 Mxfp4MoEMethod.process_weights_after_loading 在调用 deep_gemm.transform_weights_for_mega_moe 时也传入 mma_type=mma_type，保证权重布局与激活路径匹配。
5. 删除旧的 env hook：python/sglang/srt/arg_groups/mega_moe_hook.py 删除 handle_w4a4_mxfp4_megamoe_env 及其调用，不再写 DG_USE_FP4_ACTS=1 / DG_USE_MXF4_KIND=1；python/sglang/srt/server_args.py 同步更新 enable_w4a4_mxfp4_megamoe 的帮助文本，从“设置环境变量”改为“使用 mxf4xmxf4 MMA 类型”。
6. 测试配套：新增 test/registered/unit/layers/moe/test_mega_moe_deepgemm_api.py，覆盖 buffer 使用 typed API、开关到 mma_type 的映射、缓存按类型隔离、权重变换传参、pre_dispatch 传参以及 gran=16 交错布局；test/registered/unit/server_args/test_server_args.py 将原断言“环境变量被置 1”改为“环境变量保持不变”，回归验证 env hook 的移除。

关键文件：

- python/sglang/srt/layers/moe/mega_moe.py（模块 专家路由；类别 source；类型 core-logic；符号 _mega_moe_mma_type, _get_mega_moe_symm_buffer, _run_mega_routed, _interleave_mega_moe_gate_up）：核心适配文件：新增 _mega_moe_mma_type 统一决策 MMA 类型，buffer 缓存 key 加入 mma_type 实现隔离，dispatch 分支从环境变量判断改为 mma_type 判断，并扩展 L1 权重交错布局支持 gran=16 even/odd 模式。
- test/registered/unit/layers/moe/test_mega_moe_deepgemm_api.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestDeepGemmMegaMoeApi, setUp, tearDown, test_mxf4_buffer_uses_typed_api）：新增 262 行单元测试，覆盖新 API 的所有关键调用点：buffer 的 mma_type 参数、开关到 mma_type 映射、缓存跨类型隔离、权重变换传参、pre_dispatch 传参与 gran=16 交错布局。
- python/sglang/srt/arg_groups/mega_moe_hook.py（模块 参数解析；类别 source；类型 core-logic；符号 handle_mega_moe, handle_w4a4_mxfp4_megamoe_env）：删除 handle_w4a4_mxfp4_megamoe_env 及 DG_USE_FP4_ACTS / DG_USE_MXF4_KIND 环境变量写入逻辑，是本次从隐式 env 切换到显式 mma_type 的关键清理动作。
- python/sglang/srt/layers/quantization/mxfp4.py（模块 权重量化；类别 source；类型 dependency-wiring；符号 Mxfp4MoEMethod.process_weights_after_loading）：process_weights_after_loading 中 transform_weights_for_mega_moe 调用新增 mma_type 参数，确保权重变换布局与运行时激活路径一致。函数内局部导入 mega_moe 的 _mega_moe_mma_type，规避循环导入。
- python/sglang/srt/server_args.py（模块 服务参数；类别 source；类型 configuration；符号 enable_w4a4_mxfp4_megamoe）：更新 enable_w4a4_mxfp4_megamoe 参数帮助文本，从“设置 DeepGEMM 环境变量”改为“使用 mxf4xmxf4 MMA 类型”，与新机制保持语

义一致。

- test/registered/unit/server_args/test_server_args.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestPrepareServerArgs.test_enable_w4a4_mxfp4_megamoe_preserves_legacy_deepgemm_env）：将原测试 test_enable_w4a4_mxfp4_megamoe_sets_deepgemm_env 改名为 preserves_legacy_deepgemm_env，断言从“env 被置 1”反转为“env 保持不变”，直接回归验证 env hook 删除后的行为。
- python/pyproject.toml（模块 依赖配置；类别 config；类型 configuration）：sgl-deep-gemm 版本从 0.1.6 钉到 0.1.7，是本 PR 所有 API 适配的前提。
- docker/Dockerfile（模块 镜像构建；类别 infra；类型 infrastructure）：SGL_DEEP_GEMM_VERSION 同步升到 0.1.7，保证镜像内依赖与 pyproject 一致。

关键符号：_mega_moe_mma_type, _get_mega_moe_symm_buffer, _run_mega_routed, _interleave_mega_moe_gate_up, _interleave_mega_moe_l1_weights, build_mega_moe_experts_weights, Mxfp4MoEMethod.process_weights_after_loading, handle_mega_moe

关键源码片段

python/sglang/srt/layers/moe/mega_moe.py

核心适配文件：新增 _mega_moe_mma_type 统一决策 MMA 类型，buffer 缓存 key 加入 mma_type 实现隔离，dispatch 分支从环境变量判断改为 mma_type 判断，并扩展 L1 权重交错布局支持 gran=16 even/odd 模式。

```
# python/sglang/srt/layers/moe/mega_moe.py（核心片段）

def _mega_moe_mma_type() -> str:
    # 根据运行时配置选择 DeepGEMM 的 MMA 类型：
    # 开启 W4A4 MXFP4 后返回 mxf4xmxf4，否则保持原有 fp8xfp4 行为。
    return "mxf4xmxf4" if get_exec().moe.enable_w4a4_mxfp4_megamoe else "fp8xfp4"


def _get_mega_moe_symm_buffer(
    group, num_experts, num_max_tokens_per_rank, num_topk, hidden, intermediate_hidden
) -> SymmBuffer:
    import deep_gemm

    mma_type = _mega_moe_mma_type()
    # 缓存 key 必须带上 mma_type：fp8xfp4 与 mxf4xmxf4 的 pre_dispatch 写布局
    # 不同，buffer 跨类型复用会导致数据污染。
    key = (
        id(group),
        num_max_tokens_per_rank,
        num_experts,
        num_topk,
        hidden,
        intermediate_hidden,
        mma_type,
```

```
    )
    buf = _MEGA_MOE_SYMM_BUFFER.get(key)
    if buf is None:
        # 0.1.7 起 DeepGEMM 不再接受 use_fp8_dispatch 布尔开关，
        # 改为显式声明 mma_type，调用意图与上游 API 对齐。
        buf = deep_gemm.get_symm_buffer_for_mega_moe(
            group,
            num_experts,
            num_max_tokens_per_rank,
            num_topk,
            hidden,
            intermediate_hidden,
            mma_type=mma_type,
            activation="swiglu",
        )
        _MEGA_MOE_SYMM_BUFFER[key] = buf
    return buf

# _run_mega_routed 中的 dispatch 分支：按 mma_type 选择 pre_dispatch 实现

    mma_type = _mega_moe_mma_type()
    if mma_type == "mxf4xmxf4":
        # FP4 激活路径只能走 DeepGEMM 的 mega_moe_pre_dispatch（处理 E2M1
        # 打包），SGLang 自研 JIT 实现只支持 FP8；
        # 0.1.7 用 mma_type 取代了原先的 use_fp4_acts 开关。
        deep_gemm.mega_moe_pre_dispatch(
            hidden_states,
            topk_ids_in,
            topk_weights_in,
            buf.x,
            buf.x_sf,
            buf.topk_idx,
            buf.topk_weights,
            num_tokens=num_tokens,
            group_size=32,
            mma_type=mma_type,
        )
    else:
        mega_moe_pre_dispatch(
            hidden_states,
            topk_ids_in,
            topk_weights_in,
            buf.x,
            buf.x_sf,
            buf.topk_idx,
            buf.topk_weights,
            quant_group_size=32,
        )

# L1 权重的 gate/up 交错布局：同一函数兼容 DeepGEMM 两种排布
```

```
def _interleave_mega_moe_gate_up(t: torch.Tensor, gran: int = 8) -> torch.Tensor:
    # FP8 激活：gran=8 连续交错，[gate:0..7, up:0..7, gate:8..15, up:8..15, ...]
    # MXFP4 激活：gran=16 even/odd 交错，chunk 内先偶数后奇数，
    # 输出顺序为 [gate_even, up_even, gate_odd, up_odd]。
    num_groups, n, *rest = t.shape
    half = n // 2
    gate = t[:, :half].reshape(num_groups, half // gran, gran, *rest)
    up = t[:, half:].reshape(num_groups, half // gran, gran, *rest)
    if gran == 16:
        result = torch.cat(
            [gate[:, :, 0::2], up[:, :, 0::2], gate[:, :, 1::2], up[:, :, 1::2]],
            dim=2,
        ).reshape(num_groups, n, *rest)
    else:
        result = torch.stack([gate, up], dim=2).reshape(num_groups, n, *rest)
    return torch.empty_like(t).copy_(result)
```

test/registered/unit/layers/moe/test_mega_moe_deepgemm_api.py

新增 262 行单元测试，覆盖新 API 的所有关键调用点：buffer 的 mma_type 参数、开关到 mma_type 映射、缓存跨类型隔离、权重变换传参、pre_dispatch 传参与 gran=16 交错布局。

```
# test/registered/unit/layers/moe/test_mega_moe_deepgemm_api.py（节选）

class TestDeepGemmMegaMoeApi(CustomTestCase):
    def setUp(self):
        super().setUp()
        mega_moe._MEGA_MOE_SYMM_BUFFER.clear()

    def tearDown(self):
        mega_moe._MEGA_MOE_SYMM_BUFFER.clear()
        super().tearDown()

    def test_server_flag_selects_mxf4_mma_type(self):
        # 验证开关到 mma_type 的映射：关闭走 fp8xfp4，开启走 mxf4xmxf4。
        for enabled, expected in ((False, "fp8xfp4"), (True, "mxf4xmxf4")):
            with self.subTest(enabled=enabled):
                config = SimpleNamespace(
                    moe=SimpleNamespace(enable_w4a4_mxfp4_megamoe=enabled)
                )
                with patch.object(mega_moe, "get_exec", return_value=config):
                    self.assertEqual(mega_moe._mega_moe_mma_type(), expected)

    def test_buffer_cache_separates_mma_types(self):
        # 两种 mma_type 必须各自创建 buffer，不能串用缓存；
        # 同时验证 0.1.7 API 不再接受 use_fp8_dispatch。
        deep_gemm = ModuleType("deep_gemm")
        expected_buffers = (object(), object())
        deep_gemm.get_symm_buffer_for_mega_moe = MagicMock(
            side_effect=expected_buffers
```

```
    )
    group = object()

    with (
        patch.dict(sys.modules, {"deep_gemm": deep_gemm}),
        patch.object(
            mega_moe,
            "_mega_moe_mma_type",
            side_effect=("fp8xfp4", "mxf4xmxf4"),
        ),
    ):
        actual_buffers = tuple(
            mega_moe._get_mega_moe_symm_buffer(
                group,
                num_experts=8,
                num_max_tokens_per_rank=64,
                num_topk=2,
                hidden=128,
                intermediate_hidden=256,
            )
            for _ in range(2)
        )

    self.assertEqual(actual_buffers, expected_buffers)
    self.assertEqual(deep_gemm.get_symm_buffer_for_mega_moe.call_count, 2)
    self.assertEqual(
        [
            call.kwargs["mma_type"]
            for call in deep_gemm.get_symm_buffer_for_mega_moe.call_args_list
        ],
        ["fp8xfp4", "mxf4xmxf4"],
    )
```

评论区精华

本 PR 没有 review 评论与审核线程，主要交互发生在 CI 阶段：作者针对 test/registered/models_e2e/test_deepseek_v4_flash_fp4_megamoe_b200.py 发起重跑，第一次重跑结果仍为失败（4-gpu-b200）；随后作者又贴出一条新的 workflow 链接但未附文字说明。由于 PR 最终由作者本人合并，B200 上 FP4 MegaMoE 端到端结果的最终通过状态在材料中未被明确标注，存在不确定性。

- B200 FP4 MegaMoE e2e 测试失败与重跑 (testing): 首次重跑仍失败，最终 workflow 结果未被明确标注为通过；PR 随后由作者本人合并，B200 FP4 路径的最终 e2e 状态存疑。

风险与影响

- 风险：

1. 隐式环境变量机制被移除：enable_w4a4_mxfp4_megamoe 不再设置 DG_USE_FP4_ACTS / DG_USE_MXF4_KIND。SGLang 侧已不再读取这两个变量，但 deep_gemm 0.1.7 内部是否仍读取它们用于其他分支，材料未说明；依赖旧行为手动设置这两个 env 的用户可能会遇到行为不一致。
2. 版本钉版是硬前提：pyproject.toml 与 Dockerfile 都钉死 0.1.7，但若第三方环境绕过 pip 安装了旧版 deep_gemm，新代码里的 mma_type 关键字参数会直接抛 TypeError。
3. 权重布局数值风险：gran=16 even/odd 交错是新增布局，单元测试只验证了 _interleave_mega_moe_gate_up 自身的数学正确性，未验证与 deep_gemm 0.1.7 实际 kernel 的端到端一致性；B200 的 FP4 e2e 测试首次重跑失败，最终状态存疑，若真实布局与上游有偏差，会表现为精度异常。
4. 运行时上下文依赖：_mega_moe_mma_type() 依赖 get_exec() 已初始化，在 build_mega_moe_experts_weights 与 process_weights_after_loading 等模型加载阶段调用，存在先有 exec 再加载权重的时序假设；该假设在现有引擎启动流程中成立，但属于隐式约束。
5. 影响面控制：默认 enable_w4a4_mxfp4_megamoe=False，默认 fp8xfp4 路径行为不变，风险主要集中在显式开启该开关的模型上。 - 影响：对用户：显式开启 W4A4 MXFP4 MegaMoE 的用户可获得 NVFP4 支持，且配置语义从“隐式环境变量”变为“显式 mma_type”，更可预期；依赖旧 env 开关的外部脚本需要更新。对系统：MegaMoE（SM100/Blackwell）路径与上游 DeepGEMM API 对齐，buffer 缓存和权重布局都按 mma_type 分流，为后续更多 MMA 类型扩展打下基础。对团队：新增 262 行单元测试，把 DeepGEMM 调用契约固化为可回归的 API 测试，未来上游再改 API 时能快速暴露不兼容。 - 风险标记：依赖钉版升级，隐式环境变量机制移除，B200 e2e 验证存在不确定性，权重布局与上游 kernel 一致性未端到端验证

关联脉络

- PR #37123 [Diffusion] Fuse Qwen-Image FP8 QKV projection and Blackwell epilogue: 同属 Blackwell 量化性能工作线，体现 FP8/FP4 在 SM100 上的 kernel 化与融合推进。
- PR #37156 [Diffusion] Fuse Qwen-Image FP8 norm and activation quantization: 同为 quant + Blackwell + JIT kernel 的优化工作线，与本 PR 的上游依赖演进方向一致。
- PR #37214 test: re-enable DSV4-Flash W8A8 8p nightly perf cases: 同为 DeepSeek-V4-Flash 模型的测试配套，属于该模型在 B200/NPU 等新硬件上的验证矩阵。