PR #29554 完整报告

sgl-project/sglang

Upgrading tvm-ffi/sgl-deep-gemm/tilelang

合并时间：2026-07-02 03:32

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29554

执行摘要

- 一句话：升级 tvm-ffi、deep-gemm、tilelang 依赖并适配 API
- 推荐动作：值得精读 mega_moe.py 中的本地实现，了解如何替换上游私有 API。同时 TileLang API 变更的适配模式可作为后续升级的参考。

功能与动机

跟进依赖上游的版本更新，获取最新特性、性能优化和 bug 修复。同时在升级过程中将 SGLang 从依赖内部私有 API（如 deep_gemm.mega._interleave_l1_weights）迁移到本地实现，提高独立性和可控性。

实现拆解

1. 升级 pyproject.toml 中的版本约束：将 apache-tvm-ffi==0.1.9 → 0.1.11、sgl-deep-gemm==0.1.3 → 0.1.4、tilelang==0.1.8 → 0.1.11（经过两次中间版本调整）。
2. 适配 TileLang API 变更：在 python/sglang/srt/layers/attention/dsa/tilelang_kernel.py 和 python/sglang/srt/layers/mhc.py 中移除所有 T.gemm(..., wg_wait=...) 参数，并删除 T.wait_wgmma(0) 调用，对应 TileLang 新版本不再需要显式等待。
3. 替换 DeepGEMM 私有辅助函数：在 python/sglang/srt/layers/moe/mega_moe.py 中添加 _interleave_mega_moe_gate_up、_interleave_mega_moe_l1_weights、_transpose_mega_moe_sf_for_utccp 三个函数，替换原来从 deep_gemm.mega 导入的 _interleave_l1_weights 和 _transpose_sf_for_utccp，消除对上游私有 API 的依赖。
4. 修复 FP8 张量初始化：在 python/sglang/srt/layers/deep_gemm_wrapper/compile_utils.py 中将 _empty_token_fp8 和 _empty_block_fp8 中的 torch.empty 改为 torch.ones，适配 DeepGEMM 新版本对 scale 张量初始值的要求，避免除零或 NaN。
5. CI 回归测试：PR 通过了多次 CI rerun，包括 FP8 blockwise GEMM、离散化测试、JIT 测试等，确保无回归。

关键文件：

- python/sglang/srt/layers/moe/mega_moe.py（模块 MoE；类别 source；类型 core-logic；符号 _interleave_mega_moe_gate_up, _interleave_mega_moe_l1_weights, _transpose_mega_moe_sf_for_utccp, build_mega_moe_experts_weights）：核心变更：添加三个本地函数替代 DeepGEMM 私有 API，降低耦合度；权重构建逻辑从 _interleave_l1_weights 切换到 _interleave_mega_moe_l1_weights，scale transpose 使

用本地实现。

- python/sglang/srt/layers/attention/dsa/tilelang_kernel.py（模块 注意力；类别 source；类型 core-logic；符号 main）：适配 TileLang 新版本：移除所有 T.gemm 调用中的 wg_wait 参数，并删除 T.wait_wgmma(0) 调用，简化同步逻辑。
- python/sglang/srt/layers/deep_gemm_wrapper/compile_utils.py（模块 GEMM；类别 source；类型 core-logic；符号 _empty_token_fp8, _empty_block_fp8）：将 FP8 scale 初始化从 torch.empty 改为 torch.ones，适配 DeepGEMM 新版本要求，防止 warmup 时因未初始化 scale 导致数值错误。
- python/sglang/srt/layers/mhc.py（模块 MHC；类别 source；类型 core-logic；符号 mhc_pre_gemm_sqrsum_tilelang, mhc_pre_gemm_sqrsum_splitk_stage_0）：移除 TileLang gemm 调用中的 wg_wait=0 参数，与 tilelang_kernel.py 一致。
- python/pyproject.toml（模块 配置；类别 config；类型 configuration）：依赖版本更新入口，锁定新版本容器。

关键符号：_interleave_mega_moe_gate_up, _interleave_mega_moe_l1_weights, _transpose_mega_moe_sf_for_utccp, build_mega_moe_experts_weights, _empty_token_fp8, _empty_block_fp8

关键源码片段

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

核心变更：添加三个本地函数替代 DeepGEMM 私有 API，降低耦合度；权重构建逻辑从 _interleave_l1_weights 切换到 _interleave_mega_moe_l1_weights，scale transpose 使用本地实现。

```
# python/sglang/srt/layers/moe/mega_moe.py
# 新增函数：将 gate/up 权重交替排列以匹配 DeepGEMM 的 MegaMoE 布局
# [gate: 0..7, up: 0..7, gate: 8..15, up: 8..15, ...]
def _interleave_mega_moe_gate_up(t: torch.Tensor, gran: int = 8) -> torch.Tensor:
    num_groups, n, *rest = t.shape
    half = n // 2
    # 将前 half 作为 gate，后 half 作为 up，然后交替融合
    gate = t[:, :half].reshape(num_groups, half // gran, gran, *rest)
    up = t[:, half:].reshape(num_groups, half // gran, gran, *rest)
    result = torch.stack([gate, up], dim=2).reshape(num_groups, n, *rest)
    return torch.empty_like(t).copy_(result)


def _interleave_mega_moe_l1_weights(
    l1_weights: tuple[torch.Tensor, torch.Tensor],
) -> tuple[torch.Tensor, torch.Tensor]:
    # 对 L1 权重和 scale 同时执行 interleave
    return (
        _interleave_mega_moe_gate_up(l1_weights[0]),
        _interleave_mega_moe_gate_up(l1_weights[1]),
    )

def _transpose_mega_moe_sf_for_utccp(sf: torch.Tensor) -> torch.Tensor:
```

```
    # 将 scale 调整为 UTCCP 布局：reshape 后交换中间两维
    num_groups, mn, packed_sf_k = sf.shape
    assert sf.dtype == torch.int and mn % 128 == 0
    result = (
        sf.reshape(num_groups, -1, 4, 32, packed_sf_k)
        .transpose(2, 3)
        .reshape(num_groups, mn, packed_sf_k)
    )
    return torch.empty_like(sf).copy_(result)

# 在 build_mega_moe_experts_weights 中替换上游调用
# 原：from deep_gemm.mega import _interleave_l1_weights, _transpose_sf_for_utccp
# 改为使用本地函数：
w13_interleaved, w13_sf_interleaved = _interleave_mega_moe_l1_weights(
    (w13, w13_sf)
)
w13_sf_utccp = _transpose_mega_moe_sf_for_utccp(w13_sf_interleaved)
w2_sf_utccp = _transpose_mega_moe_sf_for_utccp(w2_sf)
```

python/sglang/srt/layers/attention/dsa/tilelang_kernel.py

适配 TileLang 新版本：移除所有 T.gemm 调用中的 wg_wait 参数，并删除 T.wait_wgmma(0) 调用，简化同步逻辑。

```
# tilelang_kernel.py 中的典型改动（kernel 主体循环内）
# 原：
# T.gemm(Q_shared_l, KV_shared_0_l, acc_s, transpose_B=True, wg_wait=-1)
# 现：
T.gemm(Q_shared_l, KV_shared_0_l, acc_s, transpose_B=True)
# 同时移除了 T.wait_wgmma(0) 调用
```

评论区精华

主要讨论围绕 sgl-deep-gemm 版本号的微调。PR 作者 Fridge003 自己提交了两次建议修改：先是建议将 0.1.4rc0 改为 0.1.4rc1，最终改为正式版 0.1.4。其他 reviewer（rainj-me）直接 approved，无实质性争议。

- sgl-deep-gemm 版本调整 (question): 版本锁定为 sgl-deep-gemm==0.1.4。

风险与影响

- 风险：
  - 回归风险：TileLang 移除 wg_wait 参数可能影响 kernel 同步行为，尤其在旧版本 GPU 上。虽然 CI 已覆盖多个测试，但极端配置下可能暴露问题。
  - 数值稳定性：FP8 scale 初始化从 torch.empty（未初始化）改为 torch.ones（全 1），可能改变 warmup 阶段的数值行为，需确认对精度无影响。
  - 依赖版本锁定：升级后部署环境必须使用新版本依赖，否则会安装失败。
- 影响：
  - 用户影响：无直接功能变化，但部署时需注意依赖版本匹配。

- 系统影响：减少对 DeepGEMM 内部 API 的耦合，有利于未来维护。TileLang kernel 性能可能因移除等待指令而略有提升。
- 团队影响：需要关注 TileLang 和 DeepGEMM 生态演进，配合上游 API 变化。
- 风险标记：核心 kernel 路径变更，依赖版本锁定，移除同步屏障可能引入竞态

关联脉络

- 暂无明显关联 PR