PR #36657 完整报告

sgl-project/sglang

[Blackwell] Reserve SMs for DeepGEMM MegaMoE grid barriers

合并时间：2026-08-29 04:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36657

执行摘要

- 一句话：为 Blackwell MegaMoE 默认保留 2 个 SM，修复 grid sync 超时
- 推荐动作：值得精读。核心看点：一是用 contextmanager 把进程级 set_num_sms 的修改限制在单个 kernel 调用并保证异常恢复；二是缓存键与 launch SM 数解耦，避免无谓的大 buffer 重分配；三是从物理 SM 数推导而非 deep_gemm.get_num_sms()，避免与 DSA indexer 的进程级改写叠加扣减。建议 Blackwell 部署团队验证默认保留 2 个 SM 的效果，并关注后续是否补充自动化回归测试。

功能与动机

关联 issue #30399 报告 GB200 上 DeepSeek V4 Pro 的 PD 分离部署出现 DeepGEMM grid sync timeout: sm=73, thread=0, grid_sync_idx=0，bisection 定位到 sgl-deep-gemm 版本升级（#29554），0.1.4 起 DeepGEMM 将静默挂起改为显式报错。PR body 说明根因：Blackwell MegaMoE 使用偶数集群网格加全网格软件屏障，若启动时占用所有 SM 而另一条 CUDA 流暂时占住某个 SM，部分集群无法驻留，已驻留集群在屏障处等待，最终导致 grid-sync 超时和 CUDA launch 失败。因此需要为 MegaMoE launch 预留 SM 驻留余量。

实现拆解

1. 配置入口：python/sglang/srt/environ.py 的 Envs 类新增 SGLANG_OPT_DEEPGEMM_MEGA_MOE_RESERVED_SMS = EnvInt(2)，默认保留 2 个 SM，设为 0 可关闭保留。
2. SM 上限推导：python/sglang/srt/layers/moe/mega_moe.py 新增 _mega_moe_max_num_sms()，用 functools.lru_cache(maxsize=1) 缓存；_device_sm < 100（Hopper/SM90）直接返回 None 保持原行为；SM100+ 从 CUDA 物理 SM 数（torch.cuda.get_device_properties）减去保留值并夹取下限 2，刻意不从 deep_gemm.get_num_sms() 推导，避免与双 batch 重叠、DSA indexer 的进程级改写叠加扣减。
3. 上下文管理器：_configure_mega_moe_deep_gemm_num_sms() 以 @contextmanager 实现，目标值取当前值与上限的较小者（不向外层 context 争抢 SM），向下取偶适配 2-CTA 集群，try/finally 保证异常路径也恢复进程级设置。
4. 调用点收窄：DeepSeek 路径 _run_mega_routed 与 Kimi K3 的 _forward_mega_experts 均只把 deep_gemm.fp8_fp4_mega_moe 这一调用包裹进上下文；gate、top-k、symmetric buffer 准备、pre-dispatch 保持原始 SM 数，避免进程级副作

用扩散到同路径的其他 DeepGEMM 调用。

5. 缓存键解耦：_get_mega_moe_symm_buffer 的缓存 key 不含 num_sms，只保留 group、num_max_tokens_per_rank、num_experts、num_topk、hidden、intermediate_hidden 等影响 buffer 尺寸 / 布局的参数，reserve 调整不会触发大 buffer 重分配。
6. 测试与验证配套：最初的 SM 保留单测在 commit f822f02 中被删除，当前 PR 未提交自动化测试；Blackwell 验证依赖 PR body 中的手动结果（GB300 side-stream repro、overlap-scheduler E2E 12,748/12,748 warmup 与 11,052 profiled 请求零错误、CUDA Graph/SBO 12,345 请求零错误、kernel 100 次迭代 0.059393 ms 与未保留的 0.059546 ms 基本持平）。

关键文件：

- python/sglang/srt/layers/moe/mega_moe.py（模块 MoE 层；类别 source；类型 core-logic；符号 _mega_moe_max_num_sms, _configure_mega_moe_deep_gemm_num_sms）：核心实现文件：新增 SM 上限推导与 set_num_sms 上下文管理器，DeepSeek V2/V4 的 MegaMoE 调用被作用域收窄地包裹，并解耦 symmetric buffer 缓存键。
- python/sglang/srt/models/kimi_k3.py（模块 模型层；类别 source；类型 core-logic；符号 _forward_mega_experts）：Kimi K3 的 MegaMoE 前向路径同样被包裹进 SM 保留上下文，确保该模型的 grid sync 超时也被修复，并与 DeepSeek 路径共用同一实现。
- python/sglang/srt/environ.py（模块 配置项；类别 source；类型 configuration）：新增环境变量 SGLANG_OPT_DEEPGEMM_MEGA_MOE_RESERVED_SMS（默认 2），提供保留 SM 数的默认值与调优 / 禁用入口。

关键符号：_mega_moe_max_num_sms, _configure_mega_moe_deep_gemm_num_sms, _run_mega_routed, _forward_mega_experts

关键源码片段

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

核心实现文件：新增 SM 上限推导与 set_num_sms 上下文管理器，DeepSeek V2/V4 的 MegaMoE 调用被作用域收窄地包裹，并解耦 symmetric buffer 缓存键。

```
import functools
from contextlib import contextmanager


@functools.lru_cache(maxsize=1)
def _mega_moe_max_num_sms() -> Optional[int]:
    # SM90 的 MegaMoE 实现不使用全网格集群启动，无需驻留余量，保持原行为。
    if _device_sm < 100:
        return None

    # 从物理 SM 数推导，而不是 deep_gemm.get_num_sms()：
    # 双 batch 重叠与 DSA indexer 会进程级改写该值，在它基础上再保留会复合扣减。
    num_sms = torch.cuda.get_device_properties(device="cuda").multi_processor_count
    reserved_num_sms = max(envs.SGLANG_OPT_DEEPGEMM_MEGA_MOE_RESERVED_SMS.
```

```
    get(), 0)
    # 夹取下限 2：即使保留值过大，也仍留出偶数余量。
    return max(2, num_sms - reserved_num_sms)


@contextmanager
def _configure_mega_moe_deep_gemm_num_sms(deep_gemm):
    max_num_sms = _mega_moe_max_num_sms()
    if max_num_sms is None:
        yield
        return

    current_num_sms = deep_gemm.get_num_sms()
    # 不向外层 context 争抢 SM，只在自己的预算内下调。
    target_num_sms = min(max_num_sms, current_num_sms)
    # MegaMoE 集群网格按 2 个 CTA 一组启动，向下取偶保持兼容。
    target_num_sms -= target_num_sms % 2
    if target_num_sms == current_num_sms:
        yield
        return

    deep_gemm.set_num_sms(target_num_sms)
    try:
        yield
    finally:
        # set_num_sms 是进程级全局状态，即使 kernel 启动报错也必须恢复。
        deep_gemm.set_num_sms(current_num_sms)

    swiglu_limit = getattr(moe.config, "swiglu_limit", None)
    # 只包住真正的 MegaMoE kernel 调用：gate、top-k、buffer 预备与
    # pre-dispatch 都继续使用原始 SM 数，避免其他 DeepGEMM 调用继承缩小值。
    with _configure_mega_moe_deep_gemm_num_sms(deep_gemm):
        deep_gemm.fp8_fp4_mega_moe(
            y,
            moe.experts.mega_l1_weights,
            moe.experts.mega_l2_weights,
            buf,
            recipe=(1, 1, 32),
            activation="swiglu",
            activation_clamp=swiglu_limit,
            fast_math=True,
        )
    y = y[:num_tokens]
```

python/sglang/srt/models/kimi_k3.py

Kimi K3 的 MegaMoE 前向路径同样被包裹进 SM 保留上下文，确保该模型的 grid sync 超时也被修复，并与 DeepSeek 路径共用同一实现。

```
        from sglang.srt.layers.moe.mega_moe import (
```

```
    _configure_mega_moe_deep_gemm_num_sms,
    _get_mega_moe_symm_buffer,
)

# 前面已完成 symmetric buffer 获取与 pre_dispatch 量化，
# 只在真正的 kernel 调用期下调进程级 SM 数，避免影响同路径其他 DeepGEMM 调用。
with _configure_mega_moe_deep_gemm_num_sms(deep_gemm):
    deep_gemm.fp8_fp4_mega_moe(
        y,
        self.experts.mega_l1_weights,
        self.experts.mega_l2_weights,
        buf,
        recipe=(1, 1, 32),
        activation="situ",
        fast_math=True,
    )
```

评论区精华

Review 中 BBuf 提出两条关键修改意见并最终 APPROVED：

1. 作用域收窄：set_num_sms 是进程级调用，原实现把 gate、top-k、buffer setup、pre-dispatch 都包在上下文里，其他 DeepGEMM 调用会继承缩小后的 SM 数，超出修复需要；作者改为只包裹 fp8_fp4_mega_moe。
2. symmetric buffer 缓存键：BBuf 指出 sgl-deep-gemm 0.1.5.post3 中 buffer 大小 / 布局由 rank、token、模型维度决定，与 num_sms 无关，加入缓存键会导致改变 reserve 时重分配大 buffer；作者确认移除。 此外 BBuf 在 issue 评论中要求把 #30592 的 Blackwell 验证带到本 PR（side-stream repro、overlap-scheduler E2E、吞吐 A/B、CUDA Graph/SBO），作者随后更新了 PR description。

- SM 覆盖作用域收窄到 fp8_fp4_mega_moe 调用 (design): 作者改为只守卫 deep_gemm.fp8_fp4_mega_moe 调用，其余步骤保持原始 SM 数。
- symmetric buffer 缓存键是否依赖 num_sms (performance): 作者确认 num_sms 不会变化，移除该缓存键字段，缓存键只保留影响 size/layout 的参数。
- 合并前补充 Blackwell 硬件验证 (testing): 作者更新 PR description，补充 GB300 side-stream repro、E2E 12,748/12,748 与 12,345 请求零错误、kernel 耗时对比等验证结果。

风险与影响

- 风险：
  1. 进程级全局状态窗口：deep_gemm.set_num_sms 是进程级设置，try/finally 虽覆盖异常恢复，但若未来推理路径引入并发 kernel 提交线程，窗口期其他 DeepGEMM 调用可能读到缩小后的 SM 数；当前 SGLang 前向路径串行，风险低。
  2. 默认行为变化：SM100+ 上 MegaMoE 默认少用 2 个 SM，实测 kernel 耗时 0.059546 ms 与 0.059393 ms 基本持平，E2E 约 126.7K tok/s/GPU，但深层 shape

与极端负载未覆盖。

3. 缺少自动化回归：单测在 commit f822f02 中被删除，CI 仅有 CPU/ 静态检查，Blackwell 行为依赖手动验证，后续回归可能无人看守。
4. Kimi K3 共用路径：kimi_k3.py 的 _forward_mega_experts 同样被修改，若 Kimi K3 有其独立的 SM 依赖需关注；方向一致，风险可控。
5. 缓存与并发：_mega_moe_max_num_sms 用 lru_cache 缓存物理 SM 数，多卡环境下若各设备 SM 数不同需按设备调用，当前实现假设单卡上下文。 - 影响：影响范围集中在 Blackwell（SM100+）上使用 DeepGEMM MegaMoE 的模型，包括 DeepSeek V2/V4（_run_mega_routed）和 Kimi K3（_forward_mega_experts），修复了 PD 分离、多流并发场景下的 grid sync timeout 与 CUDA launch 失败。Hopper/SM90 行为完全不变。对用户而言为默认生效的稳定性修复，并可通过 SGLANG_OPT_DEEPGEMM_MEGA_MOE_RESERVED_SMS 调优或置 0 关闭；对团队而言，该改动确立了“进程级 kernel 全局配置先用后还、局部收窄”的样板，可复用到其他 DeepGEMM 调用点。 - 风险标记：核心路径变更，默认行为变化，缺少自动化测试，进程级全局状态

关联脉络

- PR #30592 Reserve SMs for DeepGEMM MegaMoE: 被本 PR 明确 supersedes 的早期方案：本 PR 以更小 diff 和显式架构 gating 落地同一修复方向。
- PR #36862 [Fix] Route the Mooncake MoE A2A backend through Kimi K3's EP-A2A / SP-MoE fast path: 同改 kimi_k3.py 的 MegaMoE 路径，Kimi K3 的 MegaMoE/A2A 快速路径在持续演进，与本 PR 修改点相邻。