PR #33764 完整报告

sgl-project/sglang

Fix the router GEMM inaccuracy when using _front_w in Kimi-K3

合并时间：2026-08-09 03:23

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33764

执行摘要

- 一句话：K3 融合 GEMM 改发 fp32 路由 logits，修复专家选择精度
- 推荐动作：值得精读。三个设计决策很有借鉴价值：(1) 用「输出 fp32、消费者吸收」化解融合 GEMM 的单 dtype 冲突，且因 TGV 累加器本来就是 fp32，精度修复几乎零成本；(2) 用 bf16(round(fp32)) bit-identical 论证把 fp32 契约安全扩展到所有 runner，避免新增条件分支；(3) JIT 内核以 (in, out) dtype 二元组做模板与编译缓存管理。需要注意 _fi_kernel 私有字段耦合与测试缺口，若后续接手该路径，建议先补 dtype 契约单测。

功能与动机

PR body 明确指出：Kimi-K3 的 MoE 层在 bf16 下计算 router logits，专家选择基于被舍入的 logits 而不准确。实测 layer-1 gate 权重加 fp32 correction bias 后，5.27% 的 token 得到不同的 top-16 专家集合，平均 0.33%（p99 6.4%）的路由权重流向不同专家。vLLM 与 SGLang 自己的 EP front 都用 out_dtype=torch.float32，而融合 front 因 gate_up 与 latent 两个消费者需要 bf16，被迫把 router logits 也压成 bf16。PR body 还引用 PR#29783 先例：同样的 bf16 router 问题让 GLM-5.2 的 SWE-Bench Verified 从 79 掉到 67，且该路径按 num_tokens 分流，大 batch 评测难暴露、小 batch 才显现，诊断成本极高。

实现拆解

1. GEMM 后端支持 fp32 输出：python/sglang/kernels/ops/gemm/cutedsl_bf16_gemm.py 中 TgvGemmCuteExtKernel 新增 out_dtype 参数（默认 bf16），编译产物命名与缓存 key 都加入输出 dtype，避免 bf16/fp32 两个 epilogue 变体在编译缓存中串号；新增 _TORCH_TO_CUTLASS_OUT_DTYPE 的 fp32 映射。kimi_k3.py 的 _k3_bf16_gemm 增加 out_dtype 参数：TGV 路径直接走 cutedsl_bf16_gemm_out，其余路径用支持 out_dtype 的 torch.mm，并顺手删掉非连续 out 的暂存拷贝分支。
2. 模型入口切换 fp32 front：kimi_k3.py 新增 cached property _front_fp32（非 HIP、满足 _eligible_for_fused_front 且合并权重为 bf16 才启用），_merge_front_weights 的缓存失效列表同步加入它；_forward_fused 据此以 out_dtype=torch.float32 调用融合 GEMM，torch.split 得到的三段切片共享同一 fp32 张量。
3. 下游消费者吸收 fp32：activation.py 与 situ_and_mul.cuh 把激活内核模板从单一 T 拆成 TIn/TOut，JIT 缓存 key 改为 (in_dtype, out_dtype) 二元组，fp32 输入默认落回 bf16 输出；per_token_group_quant.cuh 的 QuantTrait 将 kVecSize 固定为 16 元素并新增 run_fp32 路径，route_quant_fused.cuh 模板化为 RouteQuantTraitT<TX>，按 score 与

activation 的 dtype 组合在 4 个特化间分派；per_token_group_quant.py 与 moe_route_quant_fused.py 的 dtype 白名单加入 fp32。

4. 覆盖全部 CUDA runner 并保留 bf16 回退：新 cached property _moe_front_needs_dense_bf16 取代原 _moe_front_needs_contiguous，只有 SM100 trtllm-gen mxfp4 runner（_fi_kernel == "trtllm_sm100" 且 precision == "default"）直接消费 fp32 strided 切片；marlin、triton、SM90/SM120 cutlass mxfp4 与 precision="bf16" 走 dense bf16 修复路径，靠 bf16(tgv_fp32) == tgv_bf16 保证 routed_input 与旧输出 bit-identical；ROCm 保持 bf16 front。
5. 测试与验证配套：本 PR 未新增自动化测试文件，验证以 CUPTI 微基准和 B300 TP8 1K/1K 端到端数据为主，并通过 /rerun-test test_kimi_k3_b300.py 与 /rerun-failed-ci 补跑 e2e 与失败 CI；PR 说明中注明 main 上 test_v2_jit_matches_aot 已有 66 例失败，与本 PR 无关。

关键文件：

- python/sglang/srt/models/kimi_k3.py（模块 模型层；类别 source；类型 data-contract；符号 _front_fp32, _moe_front_needs_dense_bf16, _k3_bf16_gemm, _forward_fused）：模型入口，新增 _front_fp32 决策与 _forward_fused 的 fp32 输出切换，并用 _moe_front_needs_dense_bf16 取代 _moe_front_needs_contiguous，定义了整个 fp32 front 的数据契约。
- python/sglang/kernels/ops/kimi_k3/activation.py（模块 激活内核；类别 infra；类型 infrastructure；符号 _jit_situ_and_mul_module, situ_and_mul）：SiTU 激活的宿主封装，JIT 模块缓存 key 从单一 dtype 改为 (in, out) 二元组，fp32 输入默认落回 bf16 输出，是所有 CUDA runner 共享的消费者改造。
- python/sglang/kernels/ops/gemm/cutedsl_bf16_gemm.py（模块 GEMM 后端；类别 infra；类型 infrastructure；符号 TgvGemmCuteExtKernel, _get_compiled_cute_ext_kernel, _TORCH_TO_CUTLASS_OUT_DTYPE）：TGV 后端新增 fp32 输出能力：TgvGemmCuteExtKernel 增加 out_dtype 参数，编译产物命名与缓存 key 加入输出 dtype，是零成本 fp32 epilogue 的关键支撑。
- python/sglang/kernels/jit/csrc/gemm/per_token_group_quant.cuh（模块 量化内核；类别 other；类型 core-logic；符号 QuantTrait, run_fp32, run_packed16）：QuantTrait 是共享量化内核，新增 fp32 输入路径 run_fp32 并把 kVecSize 固定为 16 元素不再依赖输入宽度，直接服务 fp32 front 的量化消费。
- python/sglang/kernels/jit/csrc/moe/route_quant_fused.cuh（模块 路由量化；类别 other；类型 entrypoint；符号 RouteQuantTraitT, route_quant_fused_kernel, RouteQuantFusedKernel）：路由 + 组量化融合内核模板化为 RouteQuantTraitT<TX>，launcher 按 score 与 activation 的 dtype 组合分派 4 个特化，是 fp32 契约真正落到 runner 内核的入口。
- python/sglang/kernels/jit/csrc/kimi_k3/situ_and_mul.cuh（模块 激活内核；类别 other；类型 core-logic；符号 situ_and_mul_kernel, SituAndMulKernel）：SiTU 内核模板从单一 T 拆成 TIn/TOut，向量宽度按最宽类型取，fp32 输入时单 lane 向量减半、线程数翻倍，是性能不减反增的来源。

- python/sglang/kernels/ops/quantization/per_token_group_quant.py（模块 量化内核；类别 infra；类型 infrastructure；符号 _SUPPORTED_INPUT_DTYPES）：Python 侧 dtype 白名单加入 fp32，是 route_quant_fused 特化分派被允许的宿主条件之一。
- python/sglang/kernels/ops/moe/moe_route_quant_fused.py（模块 路由量化；类别 infra；类型 infrastructure；符号 covered）：covered 判定放行 fp32 输入，与 .cuh 的模板分派保持一致。

关键符号：_front_fp32, _moe_front_needs_dense_bf16, _k3_bf16_gemm, _forward_fused, situ_and_mul, _jit_situ_and_mul_module, QuantTrait::run_fp32, situ_and_mul_kernel, route_quant_fused_kernel

关键源码片段

python/sglang/srt/models/kimi_k3.py

模型入口，新增 _front_fp32 决策与 _forward_fused 的 fp32 输出切换，并用 _moe_front_needs_dense_bf16 取代 _moe_front_needs_contiguous，定义了整个 fp32 front 的数据契约。

```
# 融合 front 是否输出 fp32：让 router 读到精确 logits。
# situ 激活与 flashinfer_mxfp4 量化器直接消费 fp32 切片；
# 其他 runner 在 _forward_fused 里把 routed_input 舍回 bf16，
# 与 bf16 front 的输出 bit-identical
@cached_property
def _front_fp32(self) -> bool:
    # 仅在非 HIP 且融合 front 合格的 bf16 权重上启用：
    # ROCm 的 torch.mm out_dtype 与 aiter 路由路径未在 fp32 链路上验证
    return (
        not _is_hip
        and self._eligible_for_fused_front
        and self._front_w.dtype == torch.bfloat16
    )


# ---- _forward_fused 中的调用点 ----
# 融合 front 用一次 GEMM 同时产出 gate_up / router_logits / routed_input
# 三段切片；输出 fp32 后 router 读到精确 logits，另两个消费者直接读 fp32，
# 省掉一次 cast 的额外 launch。TGV 累加器本就是 fp32，输出 fp32 只是去掉
# epilogue 里的 bf16 转换，几乎零成本
fused = _k3_bf16_gemm(
    hidden_states,
    self._front_w,
    out_dtype=torch.float32 if self._front_fp32 else None,
)
gate_up, router_logits, routed_input = torch.split(
    fused, self._front_sizes, dim=-1
)
if num_tokens > 1 and _is_hip and not _aiter_k3_opt:
    router_logits = router_logits.contiguous()
if self._moe_front_needs_dense_bf16:
```

```
    # fp32 front 下 cast 本身产生 dense buffer，contiguous() 是空操作；
    # bf16 front 下 cast 是空操作，contiguous() 负责拷贝。
    # 因为 bf16(tgv_fp32) == tgv_bf16（同一 fp32 累加器只多一次舍入），
    # routed_input 与改动前 bit-identical
    routed_input = routed_input.to(hidden_states.dtype).contiguous()
```

python/sglang/kernels/ops/kimi_k3/activation.py

SiTU 激活的宿主封装，JIT 模块缓存 key 从单一 dtype 改为 (in, out) 二元组，fp32 输入默认落回 bf16 输出，是所有 CUDA runner 共享的消费者改造。

```
# 融合 SiTU（SoftCap-GLU）激活：输入 bf16 或 fp32，输出 dtype 由 out 决定。
# fp32 输入时默认落回 bf16 输出——内核内部本就按 fp32 计算，输入变宽只
# 改变加载宽度，输出保持 bf16 与下游契约一致
def situ_and_mul(
    input: torch.Tensor,
    out: Optional[torch.Tensor],
    beta: float,
    linear_beta: Optional[float],
) -> torch.Tensor:
    hidden_size = input.shape[-1] // 2
    if out is None:
        out_dtype = torch.bfloat16 if input.dtype == torch.float32 else input.dtype
        out = input.new_empty(*input.shape[:-1], hidden_size, dtype=out_dtype)
    # 2D 输入可能是行 stride（例如融合 GEMM 输出的一段切片），需要显式
    # view；更高维输入保持 dense-view 路径
    if input.dim() == 2 and input.stride(0) != 1:
        input_2d = input.view(-1, input.shape[-1])
        out_2d = out.view(-1, hidden_size)
    else:
        input_2d, out_2d = input, out
    has_linear_beta = linear_beta is not None
    # JIT 模块缓存 key 从单一 dtype 改为 (in_dtype, out_dtype) 二元组，
    # bf16 -> bf16 与 fp32 -> bf16 是两个独立编译产物
    module = _jit_situ_and_mul_module(input_2d.dtype, out_2d.dtype)
    module.run(
        input_2d,
        out_2d,
        beta,
        1.0 / beta,
        linear_beta if has_linear_beta else 0.0,
        1.0 / linear_beta if has_linear_beta else 0.0,
    )
    return out
```

python/sglang/kernels/jit/csrc/moe/route_quant_fused.cuh

路由 + 组量化融合内核模板化为 RouteQuantTraitT<TX>，launcher 按 score 与 activation 的 dtype 组合分派 4 个特化，是 fp32 契约真正落到 runner 内核的入口。

```
// Activation dtype 模板化：同一份“路由 + 组量化”融合内核覆盖 bf16 / fp32
```

```
// 两种输入。路由打分（score）恒走 fp32；量化段按 TX 选择 QuantTrait
// 特化，fp32 输入走新增的 run_fp32 路径（kVecSize 固定为 16 元素）
template <typename TX>
using RouteQuantTraitT = QuantTrait<
    TX, fp8_e4m3_t,
    /*kGroupSize=*/32, /*kUe8m0=*/true,
    /*kRowMajor=*/true, /*kAligned=*/true, /*kFuseSiluAndMul=*/false>;

// 启动时按 score 与 activation 的 dtype 组合在 4 个特化内核间分派；
// 只有 trtllm-gen SM100 mxfp4 runner 会以 fp32 strided 切片走到 fp32 分支
if (score_dtype.is_type<fp32_t>()) {
    if (x_dtype.is_type<fp32_t>()) {
        SGL_ROUTE_QUANT_LAUNCH(fp32_t, fp32_t);
    } else {
        SGL_ROUTE_QUANT_LAUNCH(fp32_t, bf16_t);
    }
} else {
    if (x_dtype.is_type<fp32_t>()) {
        SGL_ROUTE_QUANT_LAUNCH(bf16_t, fp32_t);
    } else {
        SGL_ROUTE_QUANT_LAUNCH(bf16_t, bf16_t);
    }
}
```

评论区精华

BBuf 在 APPROVE review 中提问 "Any end2end acc can be reported?"，要求提供端到端精度数据；b8zhong 回应正在用 perf 与 acc 数据验证，但 PR 最终未附上精度数字，以端到端性能数据（均值 -0.11%）作为合入依据。b8zhong 主动披露曾考虑解耦 router 单独跑 GEMM 的备选方案，但 tinyN GEMM 才能保住性能，multi-streaming 或串行化都会造成 4–5% E2E 回退，因此保留融合、只把输出改成 fp32 是更优折中。性能验证方面，b8zhong 给出 TP-8 B300 1K/1K 各 batch size（bs 1–52）数据，最大偏差 +0.31%/-0.68%，均值 -0.11%，判定为噪声。

- 端到端精度结果请求 (testing): b8zhong 在 issue 评论说明正在用 perf 与 acc 数据验证，并给出 1K/1K 端到端性能数据（均值 -0.11%，噪声内）；PR 最终未附上精度数字即合并。
- 是否解耦 router GEMM 的备选设计 (design): 采用「保留融合、GEMM 直接写 fp32」方案，解耦方案被否。
- 跨 batch size 性能回归验证 (performance): 性能无回退，合入前补跑了 K3 B300 e2e 与失败 CI。

风险与影响

- 风险：
  1. kimi_k3.py 的 _moe_front_needs_dense_bf16 依赖 method._fi_kernel == "trtllm_sm100" 这一私有字段做契约判据，flashinfer 侧内核命名或版本升级可能让 fp32 strided 切片被错误直接消费，产生隐性数值错误。

2. 共享专家激活数值变化：SiTU 现在读 fp32 值，相对 fp32 参考的最大相对误差从 3.9e-03 降到 2.5e-03（更准），但输出与旧版本不再 bit 一致，属于行为变更，对依赖旧输出的长尾评测可能产生微小偏差。
3. bit-identical 论证只覆盖 TGV 路径：bf16(tgv_fp32) == tgv_bf16 依赖同一 fp32 累加器单次舍入；非 TGV 路径（cuBLAS out_dtype）的舍入一致性未在 PR 中显式论证。
4. CUDA/ROCm 分叉：_front_fp32 以 _is_hip 硬编码关闭，AMD 上 K3 仍保留 bf16 router，路由精度问题在 ROCm 未修复，两平台数值行为不一致。
5. 性能拐点：非 trtllm runner 的 1-token batch 因 cast 多一次 launch（前端 GEMM + cast 从 20.45 us 到 21.95 us），虽然 e2e 平均无回退，但单 token 场景有固定小开销。
6. 测试缺口：无新增自动化测试覆盖 fp32 front 的 dtype 契约、4 个 route_quant_fused 特化分派与 bit-identical 保证，回归主要靠手工 E2E；且 main 上 test_v2_jit_matches_aot 66 例失败使量化内核测试基线不干净。 - 影响：精度上，所有 CUDA 部署（SM90 + Blackwell）的 Kimi-K3 路由选择更接近 fp32 参考，top-16 专家集合与路由权重分布误差显著收敛，参照 GLM-5.2 先例可避免 SWE-Bench 类评测在 bf16 router 下掉分。性能上，融合结构保留（单 GEMM 读一次激活），且 fp32 输出让 SiTU 延迟敏感内核线程翻倍，每 MoE 层微基准 -0.67~-1.28 us，B300 TP8 端到端 1K/1K 均值 -0.11% 无回退。代码面上，per_token_group_quant、route_quant_fused、situ_and_mul 三个共享 JIT 内核的 dtype 组合从 1 个扩到 2–4 个编译特化，编译时间与缓存体积上升；契约从 runner backend 决定 contiguous 与否改为 runner 内核决定 dense/bf16 与否，后续新增 runner 或内核需同步更新 _moe_front_needs_dense_bf16 判定，新增维护面。 - 风险标记：核心路径变更，缺少测试覆盖，私有字段耦合，CUDA/ROCm 分叉，数值行为变更

关联脉络

- PR #29783 GLM-5.2 fp32 router logits（PR body 引用）：PR body 直接引用：GLM-5.2 在 bf16 router logits 下 SWE-Bench Verified 从 79 掉到 67，是本次 K3 修复的直接先例与动机来源。
- PR #33936 feat(vlm): auto-select CUDA VMM on multi-node MNNVL: 同期改动同一文件 python/sglang/srt/models/kimi_k3.py，K3 模型文件同时承载多模态传输与 MoE 数值精度两条功能线。
- PR #33400 [jit_kernel] Move JIT kernels into namespace sglang: 本 PR 合入时与 namespace sglang 迁移在 situ_and_mul.cuh、route_quant_fused.cuh 产生冲突并手工解决（见提交 a5e9e9b、ee8275c），两条 JIT 内核改造线在同一批文件上叠加。
- PR #34106 [jit_kernel] Fix missing JIT kernel namespaces: 同属 JIT 内核 namespace/ 编译链修复线，与本次内核模板化改动共享 JIT 编译基础设施。