

PR #38913 完整报告

sgl-project/sglang

[Kernel] Add H20 block-FP8 MoE configs for GLM-5.3-Flash EP4/EP8

合并时间: 2026-09-16 14:42

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/38913>

执行摘要

- 一句话: 为 H20 新增 GLM-5.3-Flash block-FP8 MoE 调优配置
- 推荐动作: 值得精读, 但重点不在 diff 本身 (只有 1 行代码 + 2 份 JSON), 而在于它示范了 sglang 的 MoE 配置分发机制: 文件名即查找键、M 桶分档、设备名隔离, 以及调优器与运行时如何共享同一套 E/topk/intermediate_size 口径。建议关注三点: 一是 `get_model_config()` 中 `Glm5NextForConditionalGeneration` 被加入外层白名单但未加入共享专家融合内层白名单的语义后果 (E 不加 1); 二是小 M 与大 M 收益落差极大 (1.9x vs 1.05x), 不要向长 prefill 场景承诺同等加速; 三是这类配置资产与 Triton 版本目录强绑定, 升级 Triton 前需要预留重新调优的窗口。

功能与动机

作者在 PR body 中给出的动机是: GLM-5.3-Flash 在 H20 上使用 block-FP8 Triton MoE 时缺少设备专属配置, 通用默认 tile 在小 M (decode / 低并发) 区间效率低——官方 kernel benchmark 中 M=1 时默认耗时 131.15us (EP4) / 133.20us (EP8), 与调优后的 69.57us / 72.90us 相比接近 1.9 倍差距。同时, 官方 tuner 无法识别

`Glm5NextForConditionalGeneration` 这一嵌套 text config 的多模态架构, 导致无法生成与运行时一致查找键的配置, 因此需要先在调优器中补上该架构分支。

实现拆解

变更目标是让 H20 上 GLM-5.3-Flash 的 block-FP8 Triton fused MoE 在每个 M 桶都命中专属 tile 参数, 而不是回退到通用默认配置; 共 3 个文件、+293/-0。

1. 新增两份设备专属配置表 (运行时生效点): 在 `python/sglang/srt/layers/moe/moe_runner/triton_utils/configs/triton_3_7_1/` 下新增 E=36 与 E=72 两份 `device_name=NVIDIA_H20, dtype=fp8_w8a8, block_shape=[128, 128]` 的 JSON (各 146 行)。文件名本身构成查找键——Triton 版本目录 + 本地专家数 E + 中间维 N=2048 + 设备名 + 量化 dtype + 权重块形状; 文件内容是从 1 到 4096 共 18 个 M 桶到 tile 参数的映射, 每桶包含 `BLOCK_SIZE_M/N/K`、`GROUP_SIZE_M`、`num_warps`、`num_stages`。策略上小 M 桶用 16x64 / 16x128 小 tile 抑制尾块与访存浪费 (该档正是 1.6x-1.9x 加速的主要来源), M>=256 后收敛到 64x64x128, `BLOCK_SIZE_K` 恒为 128 以对齐 block-FP8 的 [128,128] 量化块。
2. 让调优器识别 GLM-5.3-Flash (数据生成侧): `benchmark/kernels/fused_moe_triton/common_utils.py` 的 `get_model_config()` 中, 将 `Glm5NextForConditionalGeneration` 加入

Deepseek/GLM 家族分支的 architecture 白名单 (+1 行)，使其按 `n_routed_experts/moe_intermediate_size` 推导 E 与 `intermediate_size`。关键细节是该架构不在共享专家融合的内层白名单中，因此 E 严格等于 `n_routed_experts // ep_size` (不加 1)，`topk` 取 `num_experts_per_tok` (传入 `topk_ids_dir` 时 +1)；只有这个口径与运行时一致，调优产物才会真正被命中。PR 描述称该识别的目的是让 tuner 读到嵌套的 `textconfig`。

3. 生成与自验证流程：配置由官方 tuner 产出，作者在 H20 96GB / PyTorch 2.13.0+cu130 / Triton 3.7.1 上做了三层验证——kernel 级 18 个 M 点 × 100 次 CUDA graph 迭代、端到端 1K/1K 的 C1/C16 对比、GSM8K 前 64 样本 4-shot 精度对齐 (基线 / 调优均 63/64，且都错同一题)。EP4 M=512 首次测得 0.991x，随后两次配对重复得到 1.004x 与 1.002x，作者判定为中性并在 body 中显式披露。
4. 配套与范围收敛：没有新增任何测试文件，也没有修改运行时调度或 kernel 实现代码；5 个 commit 中最后一笔把贡献面收敛到「配置 + 架构支持」(commit message 即 "Keep H20 tuning contribution focused on configs and architecture support")，另有一笔是从 commit 67f9e08 摘取的 EP4 prefill tile。回归口径完全依赖人工 benchmark，未接入上游 GPU CI。

关键文件：

- `python/sglang/srt/layers/moe/moe_runner/triton_utils/configs/triton_3_7_1/E=36,N=2048,device_name=NVIDIA_H20,dtype=fp8_w8a8,block_shape=[128, 128].json` (模块 MoE 内核配置；类别 config；类型 configuration)：新增的 H20 专属 block-FP8 MoE 配置表之一 (E=36 本地专家形态)，是运行时真正读取的生效资产；18 个 M 桶全量覆盖，小 M 档 16x64 小 tile 是 1.8x 级加速的来源。
- `python/sglang/srt/layers/moe/moe_runner/triton_utils/configs/triton_3_7_1/E=72,N=2048,device_name=NVIDIA_H20,dtype=fp8_w8a8,block_shape=[128, 128].json` (模块 MoE 内核配置；类别 config；类型 configuration)：另一份 H20 专属配置表 (E=72 本地专家形态)，与 E=36 版本同结构、同覆盖范围但每桶 tile 略有差异 (如 M=4 用 `num_stages` 5、M=256 用 32x256)，体现按本地专家数分别调优的策略。
- `benchmark/kernels/fused_moe_triton/common_utils.py` (模块 调优器；类别 source；类型 core-logic；符号 `get_model_config`)：唯一被修改的源码文件 (+1 行)：在 `get_model_config()` 中把 `Glm5NextForConditionalGeneration` 加入 Deepseek/GLM 家族分支，使调优器能按正确的 E/topk/intermediate_size 口径生成配置键，是两份 JSON 能被运行时命中的前提。

关键符号：`get_model_config`

关键源码片段

```
python/sglang/srt/layers/moe/moe_runner/triton_utils/configs/triton_3_7_1/  
E=36,N=2048,device_name=NVIDIA_H20,dtype=fp8_w8a8,block_shape=[12  
8, 128].json
```

新增的 H20 专属 block-FP8 MoE 配置表之一 (E=36 本地专家形态)，是运行时真正读取的生效资产；18 个 M 桶全量覆盖，小 M 档 16x64 小 tile 是 1.8x 级加速的来源。

```

// 文件: python/sglang/srt/layers/moe/moe_runner/triton_utils/configs/triton_3_7_1/
// E=36,N=2048,device_name=NVIDIA_H20,dtype=fp8_w8a8,block_shape=[128, 128].json
// 文件名即查找键: Triton 版本 / 本地专家数 E / 中间维 N / 设备名 / 量化 dtype / 权重块形状。
// 顶层每个 key 是 MoE 单次前向的 M 桶 (token 数), 运行时按当前 M 落入的桶取 tile 参数;
// 未命中则静默回退通用默认配置 (表现为优化不可见, 而不是报错)。
{
  "1": {
    // decode / 极低并发: 小 BLOCK_SIZE_M 与小 BLOCK_SIZE_N 减少尾块与访存,
    // 这一档是 1.83x-1.89x 加速的主要来源
    "BLOCK_SIZE_M": 16,
    "BLOCK_SIZE_N": 64,
    "BLOCK_SIZE_K": 128,
    "GROUP_SIZE_M": 64,
    "num_warps": 4,
    "num_stages": 3
  },
  "2": { "BLOCK_SIZE_M": 16, "BLOCK_SIZE_N": 64, "BLOCK_SIZE_K": 128, "GROUP_SIZE_M":
32, "num_warps": 4, "num_stages": 3 },
  "4": { "BLOCK_SIZE_M": 16, "BLOCK_SIZE_N": 64, "BLOCK_SIZE_K": 128, "GROUP_SIZE_M":
32, "num_warps": 4, "num_stages": 3 },
  "8": {
    "BLOCK_SIZE_M": 16,
    "BLOCK_SIZE_N": 64,
    "BLOCK_SIZE_K": 128,
    "GROUP_SIZE_M": 1,
    "num_warps": 4,
    // 全表唯一把流水级数提到 4 的桶: M=8 时寄存器与共享内存仍有余量
    "num_stages": 4
  },
  "16": {
    // 从 M=16 起 N 维由 64 放大到 128, 用更大的权重读取粒度摊薄访存
    "BLOCK_SIZE_M": 16,
    "BLOCK_SIZE_N": 128,
    "BLOCK_SIZE_K": 128,
    "GROUP_SIZE_M": 64,
    "num_warps": 4,
    "num_stages": 3
  },
  // 中间省略 M=24/32/48/64/96/128: 仍以 16x128 或 32x64 的小 tile 为主
  "256": {
    // M>=256 后 tile 形状收敛为 64x64x128, 对应加速比回落到约 1.05x
    "BLOCK_SIZE_M": 64,
    "BLOCK_SIZE_N": 64,
    "BLOCK_SIZE_K": 128,
    "GROUP_SIZE_M": 64,
    "num_warps": 4,
    "num_stages": 3
  },
  "1024": { "BLOCK_SIZE_M": 64, "BLOCK_SIZE_N": 64, "BLOCK_SIZE_K": 128, "GROUP_SIZE_M":

```

```

M": 16, "num_warps": 4, "num_stages": 3 },
"4096": { "BLOCK_SIZE_M": 64, "BLOCK_SIZE_N": 64, "BLOCK_SIZE_K": 128, "GROUP_SIZE_
M": 64, "num_warps": 4, "num_stages": 3 }
}

```

python/sglang/srt/layers/moe/moe_runner/triton_utils/configs/triton_3_7_1/
E=72,N=2048,device_name=NVIDIA_H20,dtype=fp8_w8a8,block_shape=[12
8, 128].json

另一份 H20 专属配置表（E=72 本地专家形态），与 E=36 版本同结构、同覆盖范围但每桶 tile 略有差异（如 M=4 用 num_stages 5、M=256 用 32x256），体现按本地专家数分别调优的策略。

```

// 文件：python/sglang/srt/layers/moe/moe_runner/triton_utils/configs/triton_3_7_1/
// E=72,N=2048,device_name=NVIDIA_H20,dtype=fp8_w8a8,block_shape=[128, 128].json
// 与 E=36 版本结构相同，但每桶的最优 tile 完全不同——
// 说明调优必须按「本地专家数」分开做，不能用一个 E 的结果套另一个 E 的运行配置。
{
  "1": { "BLOCK_SIZE_M": 16, "BLOCK_SIZE_N": 64, "BLOCK_SIZE_K": 128, "GROUP_SIZE_M":
64, "num_warps": 4, "num_stages": 3 },
  "4": {
    "BLOCK_SIZE_M": 16,
    "BLOCK_SIZE_N": 64,
    "BLOCK_SIZE_K": 128,
    "GROUP_SIZE_M": 32,
    "num_warps": 4,
    // 小 M 桶里最激进的流水深度：M=4 时开 5 级仍不触发寄存器溢出
    "num_stages": 5
  },
  "32": { "BLOCK_SIZE_M": 16, "BLOCK_SIZE_N": 128, "BLOCK_SIZE_K": 128, "GROUP_SIZE_
M": 64, "num_warps": 4, "num_stages": 3 },
  "256": {
    // 与 E=36 版本的关键差异：该桶用 32x256 的宽 N tile，而非 64x64
    "BLOCK_SIZE_M": 32,
    "BLOCK_SIZE_N": 256,
    "BLOCK_SIZE_K": 128,
    "GROUP_SIZE_M": 32,
    "num_warps": 4,
    "num_stages": 3
  },
  "1024": { "BLOCK_SIZE_M": 64, "BLOCK_SIZE_N": 64, "BLOCK_SIZE_K": 128, "GROUP_SIZE_
M": 16, "num_warps": 4, "num_stages": 3 },
  "4096": { "BLOCK_SIZE_M": 64, "BLOCK_SIZE_N": 64, "BLOCK_SIZE_K": 128, "GROUP_SIZE_
M": 16, "num_warps": 4, "num_stages": 3 }
}

```

benchmark/kernels/fused_moe_triton/common_utils.py

唯一被修改的源码文件（+1 行）：在 get_model_config() 中把

GlM5NextForConditionalGeneration 加入 Deepseek/GLM 家族分支，使调优器能按正确的

E/topk/intermediate_size 口径生成配置键，是两份 JSON 能被运行时命中的前提。

```
# benchmark/kernels/fused_moe_triton/common_utils.py :: get_model_config()
# 调优器必须先推导出「本地专家数 E、topk、中间维」，才能拼出与运行时一致的配置查找键；
# 此次只新增一个架构名，但它的归属分支决定了 E 与 topk 的具体数值口径。
elif architecture in [
    "DeepseekV2ForCausalLM",
    "DeepseekV3ForCausalLM",
    "DeepseekV32ForCausalLM",
    "DeepseekV4ForCausalLM",
    "Glm4MoeForCausalLM",
    "Glm4MoeLiteForCausalLM",
    "GlmMoeDsaForCausalLM",
    "Glm5NextForConditionalGeneration", # 本次新增：GLM-5.3-Flash 走 Deepseek/GLM 家族口径
    "KimiVLForConditionalGeneration",
    "MistralLarge3ForCausalLM",
]:
    # 本地专家数：路由专家按 ep_size 切分；只有内层白名单里的模型在共享专家融合时才额外 +1
    E = (config.n_routed_experts // ep_size) + (
        0
        if disable_shared_experts_fusion
        or architecture
        not in [
            # Glm5NextForConditionalGeneration 不在该内层名单中，
            # 因此它的 E 恒等于 n_routed_experts // ep_size，配置文件名里的 E 就是这个值
            "DeepseekV3ForCausalLM",
            "DeepseekV32ForCausalLM",
            "Glm4MoeForCausalLM",
            "Glm4MoeLiteForCausalLM",
            "GlmMoeDsaForCausalLM",
            "MistralLarge3ForCausalLM",
        ]
        else 1
    )
    # topk 与 E 用同一套开关：只有能拿到 topk_ids_dir 且开启融合时才会 +1
    topk = config.num_experts_per_tok + (
        0 if disable_shared_experts_fusion or topk_ids_dir is None else 1
    )
    # 中间维直接决定配置键里的 N（本 PR 两份配置均为 N=2048）
    intermediate_size = config.moe_intermediate_size
```

评论区精华

本 PR 的评审记录非常单薄：只有 Fridge003 给出的一条无正文 APPROVED，没有任何 inline 评论或追问，review_comments 为 0。因此「讨论了什么」只能从 commit 序列和 PR body 的自述中反推：

- 5 个 commit 中最后一笔明确做了范围收敛，说明评审前后作者主动裁剪了非必要改动（最终 diff 只有两份配置 + 1 行调优器白名单）。

- 作者主动披露了 EP4 M=512 首次测得的 0.991x (低于基线)，并用两次配对重复 (1.004x / 1.002x) 证明是中性而非回归，这是本 PR 唯一被显式讨论过的数据点。
- PR body 明确写着 "PR remains draft; upstream GPU CI has not run", 同时 CI States 中三个 run 均标记为失败，但 PR 最终仍被合并，说明合并依据完全是作者本地 H20 数据。
- 唯一评审意见为无正文 APPROVED，缺少显式技术复核 (other): 配置正确性没有第三方复核记录，合并依据完全来自作者自测数据。
- 上游 GPU CI 未运行，三个 CI run 标记失败 (testing): 未解决的疑虑: H20 配置与调优器改动缺少自动化守门，复现依赖于一台 H20 机器和作者本地的 benchmark 流程。
- EP4 M=512 单点 0.991x 的处理方式 (performance): 判定为中性而非回归，但说明该 M 档位收益归零；其余档位均为单次测量，统计置信度有限。

风险与影响

- 风险:
 1. 命中条件苛刻且静默回退: 配置只在「设备名精确匹配 NVIDIA_H20 + dtype=fp8_w8a8 + block_shape=[128,128] + 本地专家数命中 E=36/E=72 + Triton 3.7.1 目录」时生效，任一条件不符都会无声回退到通用默认配置，表现为「优化不可见」而非报错，排查成本较高。
 2. Triton 版本绑定: 配置放在 configs/triton_3_7_1/ 下，Triton 升级 (如仓库中已存在的 triton_3_8_0 目录) 后这两份配置将不再被读取，需要重新调优，属于会随时间自然失效的资产。
 3. 收益在不同 M 档位差异极大: 小 M 档 1.6x-1.9x，但 M>=512 后仅约 1.05x，EP4 M=512 单点曾测得 0.991x (复测约 1.00x)，大 M / 长 prefill 场景不要预期同等收益。
 4. 测试与 CI 缺口: 无任何自动化测试覆盖新配置; PR Test、PR Test (Extra)、AMD ROCm 三个 CI run 在 body 中均标记失败，且上游 GPU CI 未运行，配置正确性没有第三方可复现记录。
 5. 调优器口径漂移风险: common_utils.py 的 get_model_config() 是共享入口，此次只新增了 architecture 白名单，PR body 声称的作用是「读取嵌套 text config」，但可见 diff 中没有嵌套 config 展开逻辑，若该逻辑在别处以硬编码方式处理，后续换模型或改 EP 切分可能再次生成与运行时不一致的 E/topk，导致配置白调。
 6. 正确性风险较低: 查找键已包含 block_shape 与 dtype，结构上避免了错误命中到不匹配量化形状; GSM8K 64 样本与基线完全一致，说明 tf32/fp8 累加顺序变化未造成可观测偏差。
- 影响:
 - 用户侧: 仅命中「H20 + GLM-5.3-Flash + block-FP8 (128x128) + TP4/EP4 或 TP8/EP8」这一组合的用户直接受益，TPOT 由 10.13ms 降至 8.93ms (TP4) 与 9.43ms 降至 8.23ms (TP8)，C16 高并发吞吐提升 23.8%-32.6%; 其他硬件、模型、量化形状的用户零影响，无需任何配置或代码迁移。
 - 系统侧: 改动只影响 Triton MoE kernel 的 tile 调度参数，不改 API、不改数据契约、不改调度逻辑; 由于配置按设备名与形状隔离，不会污染其他设备或量化路径的性能基线。

- 团队侧：收益需要在 H20 实机上复现，团队若要复测需具备对应硬件与模型；长期维护成本在于 Triton 版本升级时需重跑 tuner 并补新目录下的配置，建议把「H20 + GLM-5.3-Flash」纳入定期调优清单。
- 风险标记：配置静默回退，绑定 Triton 版本，缺少测试覆盖，CI 未通过，收益仅限小 M 档，设备强绑定

关联脉络

- PR #28723 [Intel XPU] Enable fused_moe_triton tuning on XPU and add tuned DeepSeek-OCR-2 configs: 改动同一套调优体系：同样修改 benchmark/kernels/fused_moe_triton/common_utils.py 与 tuning_fused_moe_triton_sep.py，并向 moe_runner/triton_utils/configs/ 下按设备名补齐调优 JSON（该 PR 在 triton_3_8_0 目录、本 PR 在 triton_3_7_1 目录），是「扩平台 + 补设备配置」这一模式的直接先例。
- PR #39155 [AMD] GLM-5.2 NextN: cast draft fused MoE to per-channel FP8: 同属 GLM MoE 家族 + FP8 量化的性能线，都涉及 MoE 专家层在 FP8 下的路径选择；本 PR 在 H20 上为 GLM-5.3-Flash 补性能档位，与该 PR 构成 GLM 系列在不同硬件上的并行优化脉络。
- PR #39613 Accept MXFP8 dispatch in FlashInfer A2A TRT-LLM MoE: 同属 MoE + FP8 量化的入口 / 分派话题，讨论的也是「量化形状与 MoE 后端配置能否匹配」这一类问题，与本 PR 通过 block_shape/dtype 参与配置键设计形成呼应。