

PR #56876 完整报告

vllm-project/vllm

[Build] Move DeepGEMM pin to the vLLM fork and update the Mega MoE call convention

合并时间: 2026-09-15 15:57

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56876>

执行摘要

- 一句话: DeepGEMM pin 迁至 vLLM fork 2.8.0, 并更新 Mega MoE 调用约定
- 推荐动作: 值得精读, 尤其是维护 MoE 后端、DeepGEMM 集成或 CI 镜像的同学: 本案是「小 diff、大范围」依赖迁移的典型样本。

三个值得关注的设计决策: 一是用单一 fork pin 取代「按特性挑分支」, 把 SM120 / SM90 / SiTU 三个能力收敛到一条可复现的 commit 上; 二是只在 DeepGEMM 调用边界做参数别名映射, vLLM 内部配置名与其它后端 (TRT-LLM / CuteDSL / AITER) 保持不变, 避免配置契约外溢; 三是 `getattr(..., None)` 这类「静默 None」模式在依赖升级中极易造成功能性静默失败, 本 PR 用「规范名 + 旧拼写 fallback」兼容两种 pin, 这种双向兼容写法值得复用到其它外部内核绑定。

后续建议: 把 SiTU 的 alpha/beta 映射固化为自动化数值回归 (现在只有一性脚本与 GSM8K A/B); 把 `cmake / install script / build_deepgemm_C.py` 中的 repo、tag、子模块信息收敛成单一来源; 与 #56266 / #56255 合流时确认 `elfutils-devel` 与 `deep_jit` 的最终结论一致; 持续跟踪 fork 与上游 2.8.x 的分叉成本。

功能与动机

PR body 给出的核心矛盾是: 停在 `nv_dev` 就拿不到 2.8.0 release line, 而 2.8.0 又砍掉了 SM90 的 2-CTA KV multicast, 正是 `next_n=4 / kv_block=32` 所依赖的能力——"Staying on `nv_dev` means we cannot pick up the 2.8.0 release line without losing SM120 and SM90 paged-MQA coverage." 作者早期版本因此不得不退化 H100 (MTP=3 decode 落到 flattened 路径、kpool 索引对齐到 64-entry page), 直到 `vllm-project/DeepGEMM#7` ([SM90] Support paged MQA next_n=4 and block_kv=32) 把这段支持前移, vLLM 侧才不需要任何 SM90 改动。同时 2.8.0 把 SiTU 专用的可选参数对换成了通用的 `activation_alpha/activation_beta`, Kimi K3 是树内唯一透传 SiTU 参数的调用点, 必须跟着改; DeepSeek V4 / V4.1 因为不传 SiTU 参数而无需改动。第三个 commit 的动机是 fork 首版 `9a86ae2b` 在移植 SM90 paged-MQA 时丢了 zero-context-length guard, DSA indexer warmup 的 dummy batch 会触发 metadata kernel 越界读。

实现拆解

1. 依赖 pin 整体迁移 (`cmake/external_projects/deepgemm.cmake`、`tools/install_deepgemm.sh`): 两个文件的 repo 地址统一换成

<https://github.com/vllm-project/DeepGEMM.git>, tag 从 8b1392b (2.6.1 / nv_dev) 换成 fork dev 分支 tip ad1f1726 (上游 2.8.0 公共发布 + SM120 移植 + SM90 paged-MQA kv_block=32/next_n=4 移植 + Mega MoE task-info slot 释放修复 + 可配 SwiGLU alpha/beta + FP8/FP4 Mega MoE 的 SiTU)。这样 vLLM 侧 native_next_n_supported、_paged_mqa_logits_schedule_slots 和 kpool 页对齐全部保持不动, H100 上的 MTP=3 decode 也不再需要退化。

2. 构建 plumbing 跟着 2.8.0 的子模块变更 (cmake/external_projects/deepgemm.cmake、tools/build_deepgemm_C.py、docker/Dockerfile) : GIT_SUBMODULES 由 third-party/cutlass + third-party/fmt 改为 third-party/cutlass + third-party/deep_jit; build_deepgemm_C.py 的 include 列表把 third-party/fmt/include 换成 third-party/deep_jit/include (2.8.0 用 <format> 取代 fmt, DeepJIT 是 header-only 的 JIT 头文件依赖); deepgemm.cmake 的 file(GLOB_RECURSE _dg_headers ...) 增加 third-party/deep_jit/include/*.hl*.hpp, 因为 add_custom_command 不做隐式头文件扫描、纯头文件改动必须靠显式 glob 才能重触发 _C 重建; docker/Dockerfile 的 dnf 安装列表加 elfutils-devel, 与 #56266 / #56255 保持同一 hunk。repo/tag/ 子模块三处信息必须手工同步, 是本 PR 最容易漏改的位置。
3. Mega MoE 调用约定迁移 (vllm/models/kimi_k3/nvidia/model.py) : deep_gemm.fp8_fp4_mega_moe(...) 的 situ_beta/situ_linear_beta 两个 kwargs 替换为 activation_alpha=self.activation_beta or 1.0 与 activation_beta=self.activation_linear_beta or 0.0, 语义是 SiTU gate 分支与 up 分支各自的 tanh scale, 0.0 表示 up 分支不做 tanh。只有 Kimi K3 在树内把 SiTU 参数透传给 DeepGEMM, 所以只改这一个调用点; vLLM 自己的 activation_situ_beta / activation_situ_linear_beta 配置名 (FusedMoEConfig 以及 TRT-LLM / CuteDSL / AITER 后端) 刻意保留, 只在 DeepGEMM 调用边界做映射, 避免波及配置契约。
4. 修掉 pin bump 会顺带发布的隐性回归 (vllm/utils/deep_gemm.py) : _lazy_init() 中 _grouped_masked_impl 原本只 getattr(_dg, "fp8_m_grouped_gemm_nt_masked", None), 2.8.0 删除该 legacy 别名后静默变成 None, 随后所有调用抛 "DeepGEMM backend is unavailable", 直接打掉 BatchedDeepGemmExperts。改为先查规范名 m_grouped_fp8_gemm_nt_masked、再回退旧拼写, 使外部手动 pin 老版本 DeepGEMM 仍可解析; 两个名字 alias 到同一个 m_grouped_fp8_fp4_gemm_nt_masked 绑定, C++ 签名与 pybind 默认值在 2.6.1 / 2.8.0 之间一致。
5. 一次返工与验证 / 测试配套: 第三次 commit 9000522 把 tag 从 fork 首版 9a86ae2b 再换到修复后的 tip——首版在把 SM90 paged-MQA 移植到 2.8.0 时漏了 zero-context-length guard, 全零 context length 时 metadata kernel 会读 prefix_sum[batch_size] (越界于按 align(batch_size, 32) 分配的 shared buffer), 抛 CUDBG_EXCEPTION_WARP_OUT_OF_RANGE_ADDRESS, 而 DSA indexer warmup 恰好用 max_num_seqs (默认 1024, 已 32 对齐) 的 dummy batch 触发。测试侧本 PR 未新增或修改任何测试文件, 回归网完全依赖既有用例 (tests/kernels/moe/test_batched_deepgemm.py 是第二 commit 的回归测试、另有 tests/models/test_deepseek_v4_mega_moe.py、tests/kernels/attention/test_deepgemm_attention.py、tests/v1/attention/test_indexer_native_next_n.py 等); 部署侧只有 docker/Dockerfile 一行依赖。SiTU 映射的正确性由作者的一次性数值对照与 GSM8K A/B 支撑, 未固化为自

动化回归。

关键文件：

- `vllm/models/kimi_k3/nvidia/model.py`（模块 模型实现；类别 `source`；类型 `data-contract`；符号 `fp8_fp4_mega_moe`, `activation_alpha`, `activation_beta`）：树内唯一一把 SiTU 参数透传给 DeepGEMM 的调用点，`activation_alpha/activation_beta` 的映射方向直接决定 Kimi K3 前向数值正确性
- `vllm/utils/deep_gemm.py`（模块 内核集成；类别 `source`；类型 `core-logic`；符号 `_lazy_init`, `_grouped_masked_impl`, `m_grouped_fp8_gemm_nt_masked`）：第二 commit 的回归修复点：DeepGEMM 2.8.0 删除 `legacy` 别名后，`getattr(..., None)` 静默返回 `None` 会让 `BatchedDeepGemmExperts` 全部失效
- `cmake/external_projects/deepgemm.cmake`（模块 外部依赖；类别 `config`；类型 `configuration`；符号 `_DEEPGEMM_UPSTREAM_REPO`, `_DEEPGEMM_UPSTREAM_TAG`, `GIT_SUBMODULES`, `_dg_headers`）：pin 迁移的主体：`repo/tag` 切换、`GIT_SUBMODULES` 从 `fmt` 换成 `deep_jit`、`_C` 重建的依赖头文件 `glob` 扩展
- `tools/install_deepgemm.sh`（模块 构建脚本；类别 `infra`；类型 `configuration`；符号 `DEEPGEMM_GIT_REPO`, `DEEPGEMM_GIT_REF`）：与 `cmake` 并行的第二处 pin 来源，`install` 脚本默认值决定非 CMake 流程拉取哪个 DeepGEMM
- `tools/build_deepgemm_C.py`（模块 内核编译；类别 `source`；类型 `configuration`；符号 `includes`）：`_C` 扩展的 `include` 路径随 2.8.0 子模块变更调整，漏改会直接导致 `_C` 编译失败
- `docker/Dockerfile`（模块 镜像构建；类别 `infra`；类型 `infrastructure`；符号 `elfutils-devel`）：新增 `elfutils-devel` 构建依赖，与 #56266 / #56255 同一 hunk；社区实测表明 `deep_jit` 需要 `libdw` 头文件

关键符号：`_lazy_init` (`vllm/utils/deep_gemm.py`, 符号解析与别名回退)，`deep_gemm.fp8_fp4_mega_moe` 调用点 (`vllm/models/kimi_k3/nvidia/model.py`, `activation_alpha/activation_beta` 映射)，`DEEPGEMM_GIT_REPO` / `DEEPGEMM_GIT_REF` (`tools/install_deepgemm.sh` 的 pin 变量)，`_DEEPGEMM_UPSTREAM_REPO` / `_DEEPGEMM_UPSTREAM_TAG` (`cmake/external_projects/deepgemm.cmake` 的 `FetchContent pin`)，`_dg_headers glob` (`cmake` 中驱动 `_C` 重建的头文件扫描)

关键源码片段

`vllm/models/kimi_k3/nvidia/model.py`

树内唯一一把 SiTU 参数透传给 DeepGEMM 的调用点，`activation_alpha/activation_beta` 的映射方向直接决定 Kimi K3 前向数值正确性

`vllm/models/kimi_k3/nvidia/model.py` —— Kimi K3 Mega MoE 前向的尾段（只在 `--moe-backend deep_gemm_mega_moe` 且开启 EP 时走到，是本次唯一需要改写的 DeepGEMM 调用点）：

```
# 输入准备与权重 finalize 保持不变；改动集中在 DeepGEMM 调用边界的参数命名。
prepare_megamoe_inputs(
```

```

        hidden_states,
        topk_weights,
        topk_ids,
        symm_buffer.x[:num_tokens],
        symm_buffer.x_sf[:num_tokens],
        symm_buffer.topk_idx[:num_tokens],
        symm_buffer.topk_weights[:num_tokens],
        is_padding=is_padding,
    )
    self.finalize_weights()
    assert self._transformed_l1_weights is not None
    assert self._transformed_l2_weights is not None
    # DeepGEMM 2.8.0 把 SiTU 专用的 situ_beta / situ_linear_beta 换成通用的一对：
    # activation_alpha 是 gate 分支的 tanh scale, activation_beta 是 linear/up 分支的
    # tanh scale; activation_beta=0.0 表示 up 分支不做 tanh（对应 FlashInfer 里 unset
    # 的 situ_linear_beta）。默认值 1.0 / 0.0 与 2.6.1 的行为等价。
    # vLLM 侧配置名仍是 activation_beta / activation_linear_beta，只在 DeepGEMM 调用
    # 边界换名；DeepSeek V4 / V4.1 不传 SiTU 参数，因此不需要改动。
    deep_gemm.fp8_fp4_mega_moe(
        y,
        self._transformed_l1_weights,
        self._transformed_l2_weights,
        symm_buffer,
        activation_clamp=activation_clamp,
        activation=self.activation,
        activation_alpha=self.activation_beta or 1.0,
        activation_beta=self.activation_linear_beta or 0.0,
        fast_math=fast_math,
    )
    return y

```

注意两处 `or` 用的是真值判断：当前 Kimi K3 配置（`activation_situ_beta=4.0`、`activation_situ_linear_beta=25.0`）不会触发，但显式写入 `0.0` 会被改写成默认值。

vllm/utils/deep_gemm.py

第二 commit 的回归修复点：DeepGEMM 2.8.0 删除 legacy 别名后，`getattr(..., None)` 静默返回 `None` 会让 `BatchedDeepGemmExperts` 全部失效

`vllm/utils/deep_gemm.py` —— `_lazy_init()` 里按符号名解析 DeepGEMM 后端实现（片段为整理后的关键解析段）：

```

# 惰性初始化：这里用 getattr 探测符号，缺失的符号会决定后端是否可用。
_cublaslt_gemm_nt_impl = getattr(_dg, "cublaslt_gemm_nt", None)
_fp8_gemm_nt_impl = getattr(_dg, "fp8_gemm_nt", None)
_fp8_einsum_impl = getattr(_dg, "fp8_einsum", None)
_grouped_impl = getattr(_dg, "m_grouped_fp8_gemm_nt_contiguous", None)
# DeepGEMM 2.8.0 删掉了 legacy 的 `fp8_*` 别名；先查规范名
# `m_grouped_fp8_gemm_nt_masked`，再回退到旧拼写，让外部手动 pin 老版本
# DeepGEMM 的场景仍能解析成功。两个名字最终 alias 到同一个

```

```
# `m_grouped_fp8_fp4_gemm_nt_masked` 绑定，其 C++ 签名与 pybind 默认值
# 在 2.6.1 / 2.8.0 之间保持一致。
# 改动前只查 `fp8_m_grouped_gemm_nt_masked`，2.8.0 下会静默返回 None，
# 随后所有调用抛 "DeepGEMM backend is unavailable"，打掉 BatchedDeepGemmExperts。
_grouped_masked_impl = getattr(
    _dg, "m_grouped_fp8_gemm_nt_masked", None
) or getattr(_dg, "fp8_m_grouped_gemm_nt_masked", None)
_grouped_fp4_impl = getattr(_dg, "m_grouped_fp8_fp4_gemm_nt_contiguous", None)
```

这类 `getattr(..., None)` 的探测方式本身没有报错路径，符号名一变就变成功能性静默失效，是本次升级中最隐蔽的一处。

cmake/external_projects/deepgemm.cmake

pin 迁移的主体：repo/tag 切换、GIT_SUBMODULES 从 fmt 换成 deep_jit、_C 重建的依赖头文件 glob 扩展

`cmake/external_projects/deepgemm.cmake` —— 外部依赖 pin 与 _C 重建依赖（片段为整理后的关键配置，中间的 `FetchContent_Declare` 细节已省略）：

```
if(DEEPGEMM_SRC_DIR)
    message(STATUS "DeepGEMM using local DEEPGEMM_SRC_DIR: ${deepgemm_SOURCE_DIR}")
else()
    # 与 tools/install_deepgemm.sh 中的 DEEPGEMM_GIT_REF 必须保持一致（手工同步）
    set(DEEPGEMM_UPSTREAM_REPO "https://github.com/vllm-project/DeepGEMM.git")
    # 固定到 fork dev 分支的 tip：上游 2.8.0 公共发布 + 5 个 fork-only commit，
    # 包含 SM120（消费级 Blackwell）移植、SM90 paged-MQA 的 kv_block=32 / next_n=4
    # 移植、Mega MoE task-info slot 释放修复、可配 SwiGLU alpha/beta，以及
    # FP8/FP4 Mega MoE 的 SiTU。停在 nv_dev 就无法同时拿到 2.8.0 与这两条架构覆盖。
    set(DEEPGEMM_UPSTREAM_TAG "ad1f1726aa540a76c1d26d6a120effb8de21eaa4")
    # ...
    # 2.8.0 用 deep_jit（header-only 的 JIT 头文件依赖）取代了 fmt 子模块
    GIT_SUBMODULES "third-party/cutlass" "third-party/deep_jit"
    # ...
endif()

# add_custom_command 不做隐式头文件扫描；显式 glob，让 DeepGEMM / cutlass /
# deep_jit 的纯头文件改动也能重新触发 _C 重建
file(GLOB_RECURSE _dg_headers
    "${deepgemm_SOURCE_DIR}/csrc/*.h"
    "${deepgemm_SOURCE_DIR}/csrc/*.hpp"
    "${deepgemm_SOURCE_DIR}/deep_gemm/include/*.h"
    "${deepgemm_SOURCE_DIR}/deep_gemm/include/*.hpp"
    "${deepgemm_SOURCE_DIR}/deep_gemm/include/*.cuh"
    "${deepgemm_SOURCE_DIR}/third-party/deep_jit/include/*.h"
    "${deepgemm_SOURCE_DIR}/third-party/deep_jit/include/*.hpp")
```

repo/tag 同时在 `tools/install_deepgemm.sh` 出现一份，注释里仅以 "Keep in sync" 约束，属于典型的手工同步点。

评论区精华

- `elfutils-devel` 的必要性被外部实测回答：作者自己在 `body` 里留了自疑——"I could not find any libelf consumer in DeepGEMM 2.8.0 or in DeepJIT ... if a reviewer knows it is unnecessary, it should be dropped from all three."; 随后 `ivanusto` 给出一线证据：在 `aarch64 nightly` 镜像上从源码构建 `fork` 时 "it needs the elfutils headers (libdw-dev on Ubuntu) for `deep_jit`", 并补充 `cusparse.h` 要取自 `pip` 的 `nvidia/cu13/include` 而非 `/usr/local/cuda/include`。这条评论基本锁定了 `docker/Dockerfile` 里 `elfutils-devel` 的保留结论。
- SM12x 覆盖由社区补充验证：`ivanusto` 开头即认可换 `pin` 的方向——"Thanks for moving the pin to the vLLM fork while keeping the 12.x archs and `support_deep_gemm()` as they are. That keeps SM12x on a working DeepGEMM.", 并在 2x DGX Spark (GB10, `sm_121`) 上做了 `kernel` 级 A/B (明确声明 "no weights involved, so this is kernels only, not serving")，对照 `fp8_gemm_nt` (`block 128`、`bf16` 输出) 在 `float32 block scales` 与 `ue8m0 scales` 下相对 `native_w8a8_block_matmul` 的表现。
- 评审强度明显偏低：`claude[bot]` 因 PR 来自 `fork` 直接跳过自动评审 ("This pull request is from a fork — automated review is disabled")，`ywang96` 给的是空 `body` 的 APPROVED, `review_comments` 为 0。也就是说这次变更基本靠作者自证的数值实验 + CI 兜底，没有逐行 review 的交锋。
- 作者对重复性与重叠的自证：`body` 明确列出与 #56266 (Mega-Gate)、#56255 (Mega-mHC) 在 `fmt` → `deep_jit plumbing` 和 `elfutils-devel hunk` 上的重叠，说明 "whichever lands first leaves the others a trivial rebase", 同时强调 `activation` 参数改名与 `Kimi K3` 调用点是本 PR 独有，不是重复工作。
- 未闭环项被显式记录而非掩盖：`tests/kernels/quantization/test_block_fp8.py::test_w8a8_block_fp8_deep_gemm_matmul` 的 `CUDA_ERROR_LAUNCH_FAILED (719)` 在旧 2.6.1 `pin` 的 `_C` 上同样复现，作者结论是 "I believe this is local to my setup, but I could not get a clean signal on that suite either way, so I am flagging it rather than reporting it green."
- CI 线索：4 次 `/ci run`，其中一次因分支落后 `main` 被拒 ("This PR is 73 commits behind upstream `main`. ... No new CI build was started.")，作者 `rebase` 后以 merge commit `1b5a7abb` 收尾并在 `Buildkite #89024` 跑通。
 - `elfutils-devel` 是否真的必要 (question): `elfutils-devel` 保留，且与 #56266 / #56255 保持一致；作者的自疑被外部实测证据回答 (`libdw` 是 DeepJIT 的构建期依赖)。
 - 2.8.0 丢失 SM90 2-CTA KV multicast, 是否要退化 H100 (correctness): 通过 `fork` 侧移植解决, vLLM 侧 SM90 路径零改动；`body` 明确 "this PR needs novLLM-side SM90 changes"。
 - `masked grouped FP8 GEMM` 的静默别名失效 (correctness): 改为先查规范名 `m_grouped_fp8_gemm_nt_masked`、回退旧拼写，两个名字 `alias` 到同一绑定且签名不变，同时兼容外部 `pin` 老版本。
 - SiTU 参数映射方向是否被可靠验证 (testing): 映射方向被正向 + 反向实验验证，不是静默通过；但验证脚本与 A/B 流程未进仓，后续改动没有自动化守卫。

- fork 首版 SM90 移植漏掉 zero-context guard (correctness): 再次换 tag 到修复后的 fork tip (最终 ad1f1726), 问题在 fork 侧闭环; 这也暴露 fork forward-port 流程本身的回归风险。
- 与 #56266 / #56255 的重复与 rebase 重叠 (design): 重叠被显式披露并接受, "whichever lands first leaves the others a trivial rebase"; activation 改名与 Kimi K3 调用点为本 PR 独有。

风险与影响

- 风险:
 - 全仓依赖切换的辐射面: pin 一改, 所有使用 DeepGEMM 的 NVIDIA 路径 (BatchedDeepGemmExperts、DeepSeek V4/V4.1 的 DSA indexer、GLM、Kimi K3 MegaMoE) 一次性升到 2.8.0。第二、第三个 commit 已经暴露出两类必须人工发现的破坏 (Python 别名删除、SM90 移植丢 guard), 说明 2.6.1 → 2.8.0 的 API/ 行为面还有未被这批用例覆盖的角落。
 - 硬件覆盖不完整: 作者验证环境是 4x GB200 (SM100), "my hardware is SM100, so none of this exercises the SM90 paths", SM90 依赖 nvidia-h100-deepgemm-kernels CI job (它确实抓到过早期版本的 SM90 回归), SM120 只有 ivanusto 在 GB10 上的手工 A/B。任何 SM90/SM120 特有退化在本地验证中不可见。
 - 缺自动化回归: SiTU 的 alpha/beta 映射正确性靠一次性脚本 (calc_diff 3.56e-04 对比量化误差底 3.60e-04, 故意对调 alpha/beta 的对照组 1.36e-01) 和 GSM8K A/B (0.3160 vs auto 0.3100, n=500、 $1\sigma \approx 2.1\%$) 支撑, 两者都没有进仓; 若后续有人再次调整参数映射, 没有测试会失败。
 - or 默认值的语义边界: activation_alpha=self.activation_beta or 1.0 用的是真值判断而非显式 None 检查, 配置里若出现显式 0.0 会被替换成 1.0。当前 Kimi K3 配置 (4.0 / 25.0) 不触发, 但这是一个静默改写用户意图的写法, 建议改为 if ... is None 判断。
 - 构建信息三处手工同步: cmake/external_projects/deepgemm.cmake、tools/install_deepgemm.sh、tools/build_deepgemm_C.py 分别持有 repo/tag/ 子模块 / 头文件路径, 注释里只写了 "Keep in sync", 没有单一来源; 与 #56266 / #56255 落地时必然产生 rebase 冲突。
 - fork 依赖与上游分叉: pin 到内部 fork 意味着要在 fork 上持续 forward-port 上游 2.8.x, fork-only 的 SM120 / SM90 / SiTU commit 成了事实依赖; 本次 already 出现一次 "移植时漏 guard" 的返工 (9000522), 说明这条维护链路本身有稳定性风险。
 - 已知失败未闭环: test_block_fp8.py 的 719 启动失败在本地无法证伪, 若 CI 上也偶发, 需要额外排查是否与 sm100_fp8_fp4_gemm_1d1d 相关。
- 影响:
 - 对用户 / 系统: 所有启用 DeepGEMM 的 NVIDIA 部署都会切到 2.8.0 内核; SM120 (消费级 Blackwell) 从「只能依赖 nv_dev」变为有正式 pin 可用, SM90 的 paged-MQA next_n=4 / kv_block=32 得以保留, H100 上 MTP=3 decode 不必退化, Kimi K3 SiTU MegaMoE 端到端与 auto 后端在 500 题上统计不可区分 (0.3160 vs 0.3100)。DeepSeek-V4.1-Flash 的 GSM8K 结果是 0.9303, 略高于改动前同配方的

0.9265, 属于「无回归」信号但用的是不同环境下的历史基线。

- 对团队 / 工程: vllm-project 正式承担起 DeepGEMM fork 的维护责任, 需要关注与上游 2.8.x 的同步节奏以及 fork-only commit 的 longevity; 构建侧新增 elfutils-devel/deep_jit 依赖, 会影响镜像体积与各发行版的构建前置条件 (aarch64 上对应 libdw-dev)。
- 影响程度: 单次 diff 很小 (6 文件 +26/-13), 但属于「依赖 + 调用契约」层面的全局切换, 回滚成本高 (需同时回退 pin、子模块、Kimi K3 调用与别名回退), 建议在 SM90/SM120 机型上各保留一次冒烟验证。
- 风险标记: 全局依赖切换, SM90/SM120 覆盖不足, 缺少新增回归测试, getattr 静默 None 模式, 构建信息三处手工同步, fork 维护与上游分叉, 已知失败未闭环

关联脉络

- PR #56266 Mega-Gate (PR body 中提及的邻近 PR): 同样携带 fmt → deep_jit 的构建 plumbing 与 elfutils-devel hunk, pin 在 deepseek-ai@39d8c4ca (同一 26/09 基线); 两者构建 hunks 重叠, 先落地者会让另一侧产生一次 trivial rebase。
- PR #56255 Mega-mHC (PR body 中提及的邻近 PR): 与 #56266 情况相同: 共用同一 deep_jit plumbing 与 elfutils-devel 依赖, 但拿不到 fork 的四个额外 commit (尤其 SM120 与 SiTU), 因此与 #56876 是互补而非重复。
- PR #56893 [Model][DSv4.1] Store the whole KV in MXFP8 (FlashMLA V4.1 record): 同属 DeepSeek V4.1 + DeepGEMM 内核线; 本 PR 的 pin 变更会影响 DSv4.1 的 paged-MQA / indexer 路径, 两者对同一批内核符号有耦合。
- PR #52781 [Kernel][MoE] DeepEP v2: async finalize to overlap shared experts with combine: 同属 MoE 内核 / EP 通信路径的近期演进, 与本 PR 的 Mega MoE 调用约定、BatchedDeepGemmExperts 回归修复处于同一功能脉络。
- PR #56560 [ROCm][DSv4.1][Perf] Dequantize the MXFP8 weight once when dot_scaled cannot be used: 同样是 DeepSeek V4.1 的 MXFP8 权重 / 量化路径优化, 与本 PR 的 DeepGEMM 2.8.0 升级共同构成 MXFP8 相关内核的近期改动面。