PR #33889 完整报告

sgl-project/sglang

moe: the shared-experts-fusion decision is a per-runner value the loader installs

合并时间：2026-08-08 13:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33889

执行摘要

- 一句话：共享专家融合决策改为按 runner 安装，修复 draft 污染
- 推荐动作：值得精读。这是一个典型的“把隐式跨模块副作用收敛为显式生命周期”的重构，设计决策包括：loader 单一安装点、per-runner 双子叶（ACTIVE + speculative）、类方法门控 + wrapper 委托、forward 期拒绝读取、以及用注册表遍历 + AST 扫描做防回归。建议重点阅读 python/sglang/srt/layers/moe/utils.py 与两个新测试文件，理解如何防止“加了新 wrapper 就悄悄丢掉禁用条件”这类问题。

功能与动机

PR body 明确指出：draft（MTP/nextn）本身就是 DeepSeek/GLM/Qwen3.5/MiniMax 模型，旧实现里 *declare_load_time_override* 会把融合决策写到 target 的 bags，draft 与 target checkpoint 量化不同时会破坏 target 记录；这是祖先行为（pre-refactor 直接写共享 ServerArgs 对象）。决策应当是 per-runner 的，且应在 loader 单一模型实例化点、任何 layer 存在之前做出。同时 wrapper 模型（Kimi-K2.5、Pixtral 等）必须以构造函数下发的 config/quantization 回答 gate，否则会为错误的 checkpoint 作答，导致共享专家权重被静默错误重映射。

实现拆解

1. 决策收敛到 loader：在 python/sglang/srt/layers/moe/utils.py 新增 install_shared_experts_fusion_decision(model_class, hf_config, quant_config)，由 loader 唯一实例化点 _initialize_model() 调用；新增构建期只读访问器 is_shared_experts_fusion_disabled()，forward 期读取直接抛 AssertionError。
2. 门控从实例方法变为类方法：DeepSeek/GLM/MiniMax/Qwen3.5 各自把既有 determine_num_fused_shared_experts 的禁用条件迁移为 shared_experts_fusion_disable_reason(hf_config, quant_config) 类方法；determine_num_fused_shared_experts 退化为只读安装值；融合架构从方法参数变成类属性 fused_shared_experts_architecture（NextN draft、GLM DSA 变体覆盖它）。
3. wrapper 委托：KimiK25ForConditionalGeneration、KimiVLForConditionalGeneration、DotsVLMForCausalLM、DeepseekVL2ForCausalLM、DeepseekOCRForCausalLM、PixtralForConditionalGeneration、MiniCPMV、Qwen3_5ForCausalLMMTP 等 wrapper 按构造函数下发的 config/quantization 委托给内部族的 gate；qwen3_5_mtp.py 抽出共享的 _mtp_quant_config，构造函数与 gate 同源。qwen3_5_text.py 的同名类通过

body_cls 属性委托到同一 gate。

4. draft 隔离：initialize_moe_config 同时把用户意图播种到 runner 与 speculative 两个叶子；draft_model_build_scope 在 draft 构造期间把决策路由到 speculative 叶子并恢复 target 值；draft 权重更新不再改写进程的 model_path/load_format 记录；declare_load_time_override 删除。
5. 测试配套：新增 test_shared_experts_fusion_gates.py（固定每个族的门控分支表和 wrapper 委托内容）、test_fusion_gate_coverage.py（AST + 注册表遍历，确保任何入口类都能回答 gate）、test_draft_construction_isolation.py（隔离语义）；修改 test_deepseek_v4_shared_expert_fusion.py。

关键文件：

- python/sglang/srt/layers/moe/utils.py（模块 融合决策；类别 source；类型 core-logic；符号 install_shared_experts_fusion_decision, is_shared_experts_fusion_disabled, draft_model_build_scope, initialize_moe_config）：核心改造点：新增 install_shared_experts_fusion_decision / is_shared_experts_fusion_disabled / draft_model_build_scope，并在 initialize_moe_config 中播种双子叶，是 per-runner 融合决策的安装与访问入口。
- python/sglang/srt/models/deepseek_v2.py（模块 DeepSeek 模型；类别 source；类型 data-contract；符号 shared_experts_fusion_disable_reason, determine_num_fused_shared_experts, fused_shared_experts_architecture）：DeepSeek 家族门控从实例方法重构为类方法 shared_experts_fusion_disable_reason，determine_num_fused_shared_experts 退化为只读；NextN 通过 fused_shared_experts_architecture 自报架构。
- test/registered/unit/models/test_shared_experts_fusion_gates.py（模块 门控测试；类别 test；类型 test-coverage；符号 _quant, _FusionGateCase, _seed, _reason）：以 loader 提问方式固定每个模型族的门控分支表，并验证 wrapper 委托的 config/quantization 与构造一致。
- test/registered/unit/models/test_fusion_gate_coverage.py（模块 覆盖测试；类别 test；类型 test-coverage；符号 _gated_classes, gated_class_names, TestFusionGateCoverage, test_every_entry_class_reaching_a_gated_family_has_a_gate）：用 AST + 注册表遍历保证任何入口类到达门控家族时都持有 gate，防回归（对 qwen3_5_text 同名类等历史漏网）。
- test/registered/unit/spec/test_draft_construction_isolation.py（模块 构建隔离；类别 test；类型 test-coverage；符号 _AlwaysDisables, _NoGate, _install, TestFusionDecisionFlag）：验证 flag 播种、draft 作用域路由 / 恢复、forward 期拒绝读取与权重更新记录隔离。
- python/sglang/srt/models/qwen3_5.py（模块 Qwen3.5 模型；类别 source；类型 data-contract；符号 _qwen3_5_shared_experts_fusion_disable_reason, _disable_shared_experts_fusion, shared_experts_fusion_disable_reason）：Qwen3.5 的 ROCm 自动禁用条件迁移为模块级 gate，并注册到四个入口类；qwen3_5_text 同名类通过 body_cls 委托补上。

- python/sglang/srt/models/qwen3_5_mtp.py（模块 MTP 模块；类别 source；类型 data-contract；符号 _mtp_quant_config, shared_experts_fusion_disable_reason）：MTP 入口委托 Qwen3.5 gate，并抽取 _mtp_quant_config 使构造与 gate 共享同一量化归一化，避免为 target 的量化作答。
- python/sglang/srt/models/minimax_m3.py（模块 MiniMax 模型；类别 source；类型 data-contract；符号 shared_experts_fusion_disable_reason, determine_num_fused_shared_experts）：MiniMax 家族同样把 determine_num_fused_shared_experts 重构为类方法门控并向 loader 模式靠拢。

关键符号：install_shared_experts_fusion_decision, is_shared_experts_fusion_disabled, draft_model_build_scope, initialize_moe_config, shared_experts_fusion_disable_reason, determine_num_fused_shared_experts, _mtp_quant_config, _qwen3_5_shared_experts_fusion_disable_reason, test_every_entry_class_reaching_a_gated_family_has_a_gate, gated_class_names

关键源码片段

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

核心改造点：新增 install_shared_experts_fusion_decision / is_shared_experts_fusion_disabled / draft_model_build_scope，并在 initialize_moe_config 中播种双子叶，是 per-runner 融合决策的安装与访问入口。

```
# python/sglang/srt/layers/moe/utils.py（重构后的核心）
import logging
from contextlib import contextmanager
from sglang.srt.runtime_context import get_exec, get_flags
from sglang.srt.utils.common import log_info_on_rank0

logger = logging.getLogger(__name__)


def initialize_moe_config(server_args: ServerArgs):
    '''从 ServerArgs 播种 MoE 运行时标志。

    共享专家融合决策改为 per-runner 值：这里把用户的原始意图
    同时写到目标叶子 disable_shared_experts_fusion 与推测叶子
    speculative_disable_shared_experts_fusion；真正细化后的决策由
    install_shared_experts_fusion_decision 在模型构建前安装。
    '''
    moe = get_flags().moe
    # ... 既有 a2a / runner backend 等配置 ...
    moe.disable_shared_experts_fusion = server_args.disable_shared_experts_fusion
    moe.speculative_disable_shared_experts_fusion = (
        server_args.disable_shared_experts_fusion
    )

def is_shared_experts_fusion_disabled() -> bool:
    '''构建期读取当前 ACTIVE 的融合决策。
```

```
    仅允许在构造期调用：forward 读取会与 draft 构建窗口竞争
    （draft 构建期间 ACTIVE 叶子装的是 draft 自己的值），所以这里
    显式抛 AssertionError，要求 forward 读层上烘焙的
    num_fused_shared_experts。
    '''
    from sglang.srt.model_executor.forward_context import has_forward_context

    if has_forward_context():
        raise AssertionError(
            'is_shared_experts_fusion_disabled() called inside a forward: the '
            'fusion decision is construction-time state (it can hold the '
            "draft's value while a draft builds). Read the value your build "
            "baked in, e.g. the layer's num_fused_shared_experts."
        )
    moe = get_flags().moe
    if moe.disable_shared_experts_fusion is None:
        return get_exec().moe.disable_shared_experts_fusion
    return moe.disable_shared_experts_fusion


@contextmanager
def draft_model_build_scope():
    '''包住一个 draft 模型的构造过程。

    构造期间运行的 gate 会把决策同时写到 speculative 叶子；
    退出时恢复 target 的 ACTIVE 值。刻意不碰 runner_backend ——
    那是 speculative_moe_backend_context 的职责，且必须包住
    draft 的完整生命周期（构建 + capture + forward）。
    '''
    moe = get_flags().moe
    original_fusion = moe.disable_shared_experts_fusion
    original_scope = moe.in_speculative_scope
    try:
        moe.in_speculative_scope = True
        yield
    finally:
        moe.in_speculative_scope = original_scope
        moe.disable_shared_experts_fusion = original_fusion


def install_shared_experts_fusion_decision(model_class, hf_config, quant_config) -> None:
    '''在 loader 唯一实例化点上为当前 runner 决策并安装融合状态。

    用户显式意图优先（已禁用就不再问 gate）；否则询问模型族的
    shared_experts_fusion_disable_reason(hf_config, quant_config)，
    返回非空理由即记录日志并禁用。draft 构造期间
    （in_speculative_scope 为真）同步写 speculative 叶子。
    '''
    disabled = get_exec().moe.disable_shared_experts_fusion
```

```
    if not disabled:
        gate = getattr(model_class, 'shared_experts_fusion_disable_reason', None)
        reason = gate(hf_config, quant_config) if gate is not None else None
        if reason:
            log_info_on_rank0(
                logger, f'{reason} Shared experts fusion optimization is disabled.'
            )
            disabled = True
    moe = get_flags().moe
    moe.disable_shared_experts_fusion = disabled
    if moe.in_speculative_scope:
        moe.speculative_disable_shared_experts_fusion = disabled
```

python/sglang/srt/models/deepseek_v2.py

DeepSeek 家族门控从实例方法重构为类方法 shared_experts_fusion_disable_reason，determine_num_fused_shared_experts 退化为只读；NextN 通过 fused_shared_experts_architecture 自报架构。

```
# python/sglang/srt/models/deepseek_v2.py（重构后的门控）
class DeepseekV2MoE(nn.Module):
    def __init__(self, config, layer_id, quant_config=None, prefix='', ...):
        # ... 既有初始化 ...
        n_shared_experts = (
            0 if config.n_shared_experts is None else int(config.n_shared_experts)
        )
        # 直接读取 loader 安装的 ACTIVE 决策；draft 构建时这里是
        # draft 自己的值，不会污染 target 的状态记录。
        _fusion_disabled = is_shared_experts_fusion_disabled()
        # num_fused_shared_experts 驱动权重重映射：
        # > 0 时把 mlp.shared_experts 重映射进 mlp.experts.256 槽位。
        self.num_fused_shared_experts = 0 if _fusion_disabled else n_shared_experts
        # ... 后续 MoE 布局计算 ...


class DeepseekV2ForCausalLM(nn.Module):
    # 该 class 代表哪个架构的融合；NextN draft、GLM DSA 变体通过覆盖
    # 这个类属性声明自己的架构名。
    fused_shared_experts_architecture = 'DeepseekV3ForCausalLM'

    @classmethod
    def shared_experts_fusion_disable_reason(cls, hf_config, quant_config):
        '''回答该 checkpoint 为何不能融合共享专家；None 表示可以。

        由 loader 在每 runner 构建前调用（无实例、无层），所以只能
        从传入的 hf_config / quant_config 与当前进程状态作答。
        '''
        if get_exec().moe.enforce_shared_experts_fusion:
            return None
        if is_sbo_enabled() or is_tbo_enabled():
```

```
            return 'SBO/TBO enabled: incompatible with fusing shared expert into MoE kernel.'
        if is_deepep_class_backend():
            return 'DeepEP: fusion off by default (use --enforce-shared-experts-fusion to enable).'
        if (
            hf_config.architectures[0] != cls.fused_shared_experts_architecture
            or hf_config.n_routed_experts not in (256, 384)
            or hf_config.n_shared_experts != 1
            or (
                hf_config.n_routed_experts == 384
                and (quant_config is None or quant_config.get_name() != 'quark')
            )
        ):
            # 384 专家布局仅在 Quark MXFP4 checkpoint 下是预融合的；
            # 标准 Kimi-K2.5（compressed-tensors）会把共享专家松散存储，
            # 走融合路径会静默错载。
            return 'Config does not support fused shared expert(s).'
        # 设备算力 / EP 拓扑 / W4AFP8 等分支语义相同，略去。
        return None

    def determine_num_fused_shared_experts(self):
        # 决策由 loader 安装，这里只读不写；不再调用
        # declare_load_time_override 改写进程配置记录。
        self.num_fused_shared_experts = (
            0 if is_shared_experts_fusion_disabled() else self.config.n_shared_experts
        )
```

评论区精华

核心讨论围绕“决策迁移后哪些入口类没跟上”展开，Codex 共提出 7 个 finding，作者逐条修复并三次扩围排查：

- GLM 非 lite NextN 未迁移（P1）：Glm4MoeForCausalLMNextN 顶层仍从 config bag 读决策，decoder 已读 ACTIVE，target 自动禁用后会 fused/unfused 布局不一致，权重重映射静默跳过共享专家。修复为在构建 decoder 前运行家族门控。
- post-build scope 覆盖 draft 叶子（P2）：speculative_moe_backend_context 在 finally 无条件把 ACTIVE 写回 speculative 叶子，init_attention_backends 等后置作用域会污染 draft 决策。修复为 in_speculative_scope 标记 + 仅 gate 运行时双写。
- wrapper 嵌套 DeepSeek 无 gate（P1）：KimiK25ForConditionalGeneration 直接构造 DeepseekV3ForCausalLM，loader 只问 wrapper，standard compressed-tensors checkpoint 会走错误融合路径。修复为按构造函数下发的 text_config/quant_config 委托，并把 coverage 测试改为 hasattr 动态解析继承 gate，进而发现 PixtralForConditionalGeneration 等更多案例。
- Qwen3.5 MTP 与 qwen3_5_text 同名类漏注册：MTP 入口丢失 ROCm 多流路径；文本-only checkpoint 解析到 qwen3_5_text 的同名类，注册循环贴错了类。分别用 _mtp_quant_config 共享归一化和 body_cls 属性委托修复。

- RoutedExpertsCapturer 读到 draft 决策（P2）：改为读取 target 模型烘焙属性 num_fused_shared_experts，不再在构造外查询进程级标志。 复审结论：原 4 个 finding 全部修复，bugs 0 / suggestions 0 / nits 0，LGTM。
- GLM 非 lite NextN 未随重构迁移 (correctness): 作者在 ab1e187 修复：非 lite NextN 在构建 decoder 前运行家族门控，使顶层元数据、decoder 布局与加载器一致。
- post-build scope 覆盖 draft 融合决策叶子 (correctness): 改为 in_speculative_scope 标记 + install 时双写，无 gate 运行的作用域不动叶子；新增 test_post_build_scopes_do_not_clobber_the_draft_leaf。
- wrapper 嵌套 DeepSeek 模型未回答门控（KimiK25 误加载） (correctness): wrapper 按构造函数下发的 text_config / quant_config 委托给嵌套族 gate；coverage 测试改用 hasattr 动态解析继承门控，找到 Pixtral 等更多案例。
- Qwen3.5 MTP 入口未注册 gate (correctness): 抽出 _mtp_quant_config 让构造和 gate 同源，MTP 入口委托 Qwen3.5 gate（text config + 归一化后的量化）。
- qwen3_5_text 同名类未注册 gate (correctness): gate 改为通过 body_cls 属性委托，与构造读取同一属性；coverage 测试继续收紧。
- RoutedExpertsCapturer 读取了 draft 的决策 (correctness): capturer 改为读取 target 模型烘焙属性 num_fused_shared_experts，不再在构造外查询进程级标志。

风险与影响

- 风险：核心风险集中在加载路径的静默错误：
  - 入口类遗漏 gate：任何注册表入口类若未实现 shared_experts_fusion_disable_reason，loader 会静默回退用户意图，跳过家族的自动禁用条件；这是 PR 自己点名的失败模式，历史上已漏过 qwen3_5_text 同名类与继承 gate 的 wrapper。coverage 测试基于 AST 文本扫描，对复杂别名 / 条件 import 仍可能漏报。
  - wrapper委托一致性：DeepseekVL2不传quantization、Pixtral的GQA分支传None、Qwen3_5ForCausalLMMTP 需要归一化量化，任何“构造函数下发的值”与“gate 收到的值”漂移都会答错 checkpoint。
  - draft 作用域时序：draft_model_build_scope 不换 runner_backend 的约定依赖所有 draft worker 都配合 speculative_moe_backend_context；未来若有人在 scope 内新增跑 gate 的路径，可能重新引入叶子覆盖问题。
  - 行为回归面：40 个文件、9+ 模型族，但 GPU e2e（GLM-4.7-Flash + NEXTN、Qwen3-Next-80B-FP8）为字节级一致，16 分区 CPU 回归基线 0 回归，整体回归风险可控。
  - 影响：对用户：修复了 Kimi-K2.5 compressed-tensors、Qwen3.5 ROCm MXFP4 等场景的潜在静默权重误加载；flags dump 现在能同时看到 target 与 draft 两个 runner 的融合决策，日志更准确。 对系统：loader 成为融合决策的唯一安装点，declare_load_time_override 机制删除，后续新模型族接入只需实现一个类方法；draft 构建状态隔离成为推测解码的显式契约。 对团队：新增约 900 行测试（含 CPU CI 套件），测试资产显著增强；coverage 测试的“注册表走查 + AST 扫描”模式可复用到其他跨模型族的一致性检查。

- 风险标记：核心加载路径变更，跨 9+ 模型族迁移，静默权重误加载风险，draft/target 状态隔离，依赖 AST 扫描防回归

关联脉络

- PR #33925 config: route DCP topology reads through get_parallel(): 同属配置一致化系列：消除配置种子与运行时读取的不一致，与本 PR 把融合决策收敛到 loader 单一安装点呼应。
- PR #34026 [Spec] Propagate state capture outputs in DFlash: 同为推测解码场景下 draft/target 状态生命周期收口，关注状态泄漏与构建 / 执行时序。