PR #36529 完整报告

sgl-project/sglang

[Fix][XPU/ROCm/NPU] Defer sgl_kernel.quantization import in expert_pack

合并时间：2026-08-28 07:17

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36529

执行摘要

- 一句话：延迟 sgl_kernel.quantization 导入，修复 XPU/ROCm/NPU 启动崩溃
- 推荐动作：建议精读。核心价值有三：一是延迟导入作为 " 设备专用内核 " 的标准处理范式，与 gguf.py guard 形成了可复用的兼容性约定；二是 HFRunner 进程回收的 join/kill 两级兜底，是测试基建中处理子进程持有设备资源的典型修复；三是 review 过程展示了 " 优先复用现有工具（empty_gpu_cache）而非新造 helper" 的维护哲学。对负责多平台（XPU/ROCm/NPU）支持和 CI 稳定性的工程师尤其值得参考。

功能与动机

sgl_kernel.quantization 是 CUDA/MUSA 专用模块，gguf.py:44-70 已显式对 _is_hip、_is_npu 及其他设备（含 XPU）做 guard 并输出 "Only CUDA, MUSA and NPU support GGUF quantization currently."。而 expert_pack.py 在顶层无条件 import 它，任何导入链在非 CUDA/MUSA 设备上都会在模块加载时崩溃。实际故障来自 DeepSeek-V2 forward 路径：deepseek_v2.py:1185 forward_normal → mxfp4_flashinfer_trtllm_moe.py:385 maybe_fuse_routed_scale_and_shared_add → expert_pack.py:10，XPU CI 任务 stage-b-test-1-gpu-xpu 的 test_mla_decode_attention_backend 因此红框。即便 Kimi-K3 expert-pack 代码路径在 XPU 上永远不会进入，仅导入就足以炸掉整个服务。

实现拆解

本 PR 的最终合入内容包含三块相互关联的改动：

1. 延迟量化内核导入（核心修复）：在 python/sglang/srt/layers/quantization/expert_pack.py 中删除顶层 from sgl_kernel.quantization import ggml_moe_a8_vec，将其移入唯一调用该内核的静态方法 _kimi_vec 的函数体内。这样模块在 XPU/ROCm/NPU 上可被正常 import，而 CUDA/MUSA 上首次调用后 Python 会在 sys.modules 缓存该模块，后续调用只多一次字典查找，行为等价。风格上对齐 gguf.py:44-70 对同一模块的 guard 约定。
2. 修复 HFRunner 子进程回收：在 python/sglang/test/runners.py 中新增 _stop_model_proc 私有方法，terminate() 后先 join(timeout=30)，若进程仍存活则 kill() 并再次 join()；terminate() 和 __exit__ 均改为调用它。原因是原实现是 fire-and-forget 的 terminate()，子进程在 teardown 时仍持有加速设备，同一设备上紧接着启动的 SRTRunner 会在驱动初始化时死锁（在 Intel XPU B580 上观测到 first-attempt-hangs / retry-passes 模式）。

3. XPU 测试统一显存回收：test/registered/xpu/ 下的 test_xpu_rerank.py、test_xpu_embedding.py、test_xpu_reward.py、test_xpu_classification.py 删除各自自定义的 _xpu_free_cache，改为从 sglang.test.test_utils 导入 empty_gpu_cache，在每个测试的 HFRunner 与 SRTRunner 之间内联 gc.collect() + empty_gpu_cache()；同时给 embedding/reward/classification 测试增加 mem_fraction_static=0.55 以约束显存占用。这是把 #36360 对 cross-encoder rerank 的修复模式推广到其他三个 stage-b XPU 测试。

配套测试方面，没有新增单测文件，全部改动落在既有 XPU 注册测试（stage-b-test-1-gpu-xpu suite）和测试基建 runners.py 上，属于源码与测试联动修改。

关键文件：

- python/sglang/srt/layers/quantization/expert_pack.py（模块 量化层；类别 source；类型 dependency-wiring；符号 _kimi_vec）：核心修复文件：删除顶层 sgl_kernel.quantization 导入，移入 _kimi_vec 内部，使模块在 XPU/ROCm/NPU 上可导入。
- python/sglang/test/runners.py（模块 测试框架；类别 test；类型 test-coverage；符号 terminate, _stop_model_proc）：测试基建修复：新增 _stop_model_proc 收敛 terminate 与 __exit__的进程回收逻辑，解决 XPU 上子进程持有设备导致后续 SRTRunner 死锁的问题。
- test/registered/xpu/test_xpu_rerank.py（模块 XPU 测试；类别 test；类型 test-coverage；符号 _assert_close_scores）：首个应用 #36360 HF→SRT 显存回收模式的 XPU 测试，删除自定义 _xpu_free_cache，改用统一 empty_gpu_cache。
- test/registered/xpu/test_xpu_embedding.py（模块 XPU 测试；类别 test；类型 test-coverage）：同步应用显存回收模式，并为 SRTRunner 增加 mem_fraction_static=0.55。
- test/registered/xpu/test_xpu_reward.py（模块 XPU 测试；类别 test；类型 test-coverage）：与 embedding 相同的显存回收与显存占比调整。
- test/registered/xpu/test_xpu_classification.py（模块 XPU 测试；类别 test；类型 test-coverage）：与 embedding 相同的显存回收与显存占比调整。

关键符号：_kimi_vec, _stop_model_proc, terminate

关键源码片段

python/sglang/srt/layers/quantization/expert_pack.py

核心修复文件：删除顶层 sgl_kernel.quantization 导入，移入 _kimi_vec 内部，使模块在 XPU/ROCm/NPU 上可导入。

```
# python/sglang/srt/layers/quantization/expert_pack.py

# 原顶层导入 from sgl_kernel.quantization import ggml_moe_a8_vec 已删除，
# 因为该模块仅 CUDA/MUSA 可用，顶层导入会让整个模块在 XPU/ROCm/NPU 上
# 加载即崩（对应 gguf.py:44-70 的 guard 约定）。

@staticmethod
def _kimi_vec(
```

```
    inputs: torch.Tensor,
    weights: torch.Tensor,
    expert_ids: torch.Tensor,
    *,
    top_k: int,
    weight_type: int,
    output_size: int,
) -> torch.Tensor:
    # sgl_kernel.quantization 仅 CUDA/MUSA 可用；保持局部导入，
    # 让本模块在其它设备上仍可 import。CUDA 端首次调用后由 sys.modules
    # 缓存，后续调用只是字典查找，行为与顶层导入等价。
    from sgl_kernel.quantization import ggml_moe_a8_vec

    return ggml_moe_a8_vec(
        inputs,
        weights,
        expert_ids,
        top_k,
        weight_type,
        output_size,
        inputs.shape[0],
    )
```

python/sglang/test/runners.py

测试基建修复：新增 _stop_model_proc 收敛 terminate 与 __exit__的进程回收逻辑，解决 XPU 上子进程持有设备导致后续 SRTRunner 死锁的问题。

```
# python/sglang/test/runners.py 中 HFRunner 的进程回收逻辑

def _stop_model_proc(self):
    # fire-and-forget 的 terminate() 会让子进程在 teardown 期间继续持有
    # 加速设备；同一设备上紧接着启动的 SRTRunner 可能在驱动初始化时
    # 死锁（在 Intel XPU B580 上观测到）。因此先温和终止并等待 30 秒，
    # 进程仍存活时再强制 kill，避免 hang 住测试进程。
    self.model_proc.terminate()
    self.model_proc.join(timeout=30)
    if self.model_proc.is_alive():
        self.model_proc.kill()
        self.model_proc.join()
    self.in_queue = self.out_queue = None

def terminate(self):
    self._stop_model_proc()

def __enter__(self):
    return self

def __exit__(self, exc_type, exc_value, traceback):
    self._stop_model_proc()
```

test/registered/xpu/test_xpu_rerank.py

首个应用 #36360 HF→SRT 显存回收模式的 XPU 测试，删除自定义 _xpu_free_cache，改用统一 empty_gpu_cache。

```
# test/registered/xpu/test_xpu_rerank.py 中 HF/ SRT runner 切换处的显存回收

from sglang.test.test_utils import CustomTestCase, empty_gpu_cache

def _assert_close_scores(
    self, prompts, model_path, tp_size, torch_dtype,
    score_tolerance, attention_backend, mem_fraction_static,
) -> None:
    with HFRunner(
        model_path,
        torch_dtype=torch_dtype,
        model_type="cross_encoder",
    ) as hf_runner:
        hf_scores = hf_runner.forward(prompts).scores

    # HFRunner 关闭时泄漏 ZMQ context；先回收显存再启动 SRT，
    # 避免小显存 XPU（如 B580）上 OOM。这里复用 test_utils
    # 的 empty_gpu_cache，而非自定义的 xpu_free_cache。
    gc.collect()
    empty_gpu_cache()

    with SRTRunner(
        model_path,
        tp_size=tp_size,
        torch_dtype=torch_dtype,
        model_type="cross_encoder",
        attention_backend=attention_backend,
        chunked_prefill_size=-1,
        disable_radix_cache=True,
        mem_fraction_static=mem_fraction_static,
    ) as srt_runner:
        srt_scores = srt_runner.forward(prompts).scores
```

评论区精华

核心讨论来自 reviewer mingfeima 对第一版实现（在 test_utils.py 新增 xpu_free_cache）的评论：

> "this file has a empty_gpu_cache prefer import gc from each of the UT file and do gc + empty_gpu_cache in each UT. remove xpu_free_cache"

作者在 feaebef77f 中完全采纳：删除了 test_utils.py 中的 xpu_free_cache，四个 XPU 测试各自内联 gc.collect() + empty_gpu_cache()，并在同一 commit 中顺带修复了 HFRunner.__exit__ 不 join() 子进程的问题，指出这与 classification / breakable_cuda_graph / embedding 测试中 " 首次挂、重试过 " 的现象吻合。Fridge003 和

mingfeima 最终均 APPROVE。

- 用通用 *empty_gpu_cache* 替代新造的 xpu_free_cache (design): 作者在 feaebef77f 中删除 xpu_free_cache，四个 XPU 测试各自内联 gc.collect() + empty_gpu_cache()，并顺带修复 HFRunner.__exit__未 join 子进程的问题。
- HFRunner 子进程不 join 导致 XPU 设备死锁 (correctness): 新增 _stop_model_proc，terminate 后 join(timeout=30)，超时 kill，terminate 与 __exit__统一走该方法。

风险与影响

- 风险:
  1. *expert_pack*.py 延迟导入的剩余风险：若未来有非 CUDA/MUSA 设备意外走到 _kimi_vec，会在运行时抛 ModuleNotFoundError 而非导入期，错误延后但更明确；由于 gguf.py 的 guard 已声明此类设备不支持 GGUF 量化，实际路径不会触发。CUDA/MUSA 端行为等价，风险极低。
  2. HFRunner kill() 强杀风险：kill() 兜底可能留下未清理的共享内存或 ZMQ context，但 30 秒 join 超时已覆盖正常退出路径，且这是测试基建而非生产代码；代价是 CI 中异常子进程最多拖 30 秒失败。
  3. 测试显存参数调整：mem_fraction_static=0.55 是防御性下调，如果测试机显存极小（如 B580 12GiB）且模型加载后可用不足，可能引入新的 OOM 失败点；但从 CI 观测看它缓解了 HF→SRT 切换期的显存压力。
  4. 回归面：改动涉及量化模块导入链，任何依赖 *expert_pack* 顶层导入的第三方脚本若直接引用 sgl_kernel.quantization 名称会受影响，但仓库内无此用法。 - 影响：影响范围集中在多平台启动路径和 XPU CI：XPU/ROCm/NPU 上加载 DeepSeek-V2 等会触达 *expert_pack* 的模型时不再崩溃，这是对 #35314 回归的事后修复；CUDA/MUSA 用户完全无感（模块缓存保证等价行为）。对团队而言，HFRunner 的进程回收修复消除了 XPU 测试中 " 首次挂、重试过 " 的偶发死锁，四个 XPU 测试统一了显存回收模式，*stage-b-test-1-gpu-xpu* 的稳定性预期明显提升。改动本身很小（5 个源文件 + 4 个测试文件），但消除了多平台部署的硬阻塞。 - 风险标记：多平台导入回归，进程强杀兜底，测试显存参数调整，CI 偶发死锁修复

关联脉络

- PR #35314 引入 *expert_pack*.py 顶层 sgl_kernel.quantization 导入（PR body 引用）：本 PR 正是修复 #35314 引入的回归：顶层无条件导入导致 XPU/ROCm/NPU 模块加载崩溃。
- PR #36360 HF->SRT 显存回收修复（*commit message* 引用）：本 PR 将 #36360 对 *cross-encoder rerank* 的 HF→SRT 显存回收模式推广到 *classification/reward/embedding* 三个 XPU 测试。