PR #36871 完整报告

sgl-project/sglang

[AMD] support gfx1250 on ROCM 10

合并时间：2026-08-31 16:19

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36871

执行摘要

- 一句话：gfx1250 接入 ROCm 10，含内核修复与 MI45x 精度测试
- 推荐动作：值得精读，尤其是 aiter_mxfp4_w4a8_moe.py 与 triton_qk_rmsnorm.py。值得关注的决策： 1) 对不可用内核采取"成熟 triton 内核 + 布局转换"替代而非修内核； 2) 用 IS_GFX1250 constexpr 阻止平台 workaround 泄漏到通用路径； 3) 用环境变量矩阵管理多模型精度测试的可复现配置。对维护者：aiter 版本升级（aiter#2958 之后的统一 fused_qk_rmsnorm 入口）与 gfx1250 触发条件 AITER_FORCE_A8W4 耦合较紧，建议后续引入设备级开关集中管理平台特化逻辑。

功能与动机

PR body 明确了定位："Based on https://github.com/sgl-project/sglang/pull/32754 (gfx1250 enablement) and https://github.com/sgl-project/sglang/pull/36434 (ROCm 10 release images), see those PRs for the details of each. On top of them, this adds the gfx1250-rocm1000 Docker target on the ROCm 10.0.0 GA wheel channel, gfx1250 detection in the AMD CI dependency installer, and the gfx1250 kernel/model fixes with the MI45x accuracy tests." 更深的动机来自 gfx1250 硬件的多个工具链缺陷：aiter CK/ASM fused_moe 对 MXFP4 W4A8 布局产出乱码、内核内 TDM gather 编译失败（TDM gather dst must be 2D）、aiter fused_qk_rmsnorm 内核因 composable_kernel 版本不兼容无法 JIT 构建、fp8 x fp8 dot 在 K>=128 时返回约 1e34 的垃圾值、以及缺少 fp4 WMMA scale 指令导致 fp4 GEMM 无法执行。这些都需要在 SGLang 侧提供绕过或替代实现。

实现拆解

1. 镜像与 CI 基建（变更入口）：docker/rocm.Dockerfile 新增 gfx1250-rocm1000 阶段，基于 ROCm 10.0.0 GA wheel 通道，构建参数改用 GPU_ARCH_LIST=gfx1250；scripts/ci/amd/amd_ci_install_dependency.sh 增加 gfx1250 检测分支，并移除 ROCm 7.14 时代基于 torch.version.hip 的误判探测；workflow 按分支收敛触发范围。最终 commit（52a3d83）还决定把 bring-up 依赖的上游 Triton 76940ad 编译进 gfx1250 镜像（SDK 自带 Triton 仍存在问题），gfx942/gfx950 保持 SDK 自带。
2. 运行时平台检测与配置守卫：python/sglang/srt/utils/common.py 新增 is_gfx1250_supported()；python/sglang/srt/arg_groups/attention_hook.py 恢复 aiter 的 mem_fraction_static 守卫（显式设置时不再被 0.85 缩放），该守卫源自 #32754，是本 PR 中少数影响非 gfx1250 的改动。

3. 量化 MoE 内核适配：quark_w4a8_mxfp4_moe.py 在 gfx1250 下跳过 aiter CK/ASM shuffle 布局，改用非填充、contraction-major 的 triton moe_gemm_a8w4 布局，新增 _process_weights_gfx1250 并在 apply_weights 中直接调用新文件 aiter_mxfp4_w4a8_moe.py 的 aiter_w4a8_gfx1250_forward（禁用内核内 TDM gather、手动 gather 激活、swizzle_mx_scale=None）。quark_w4a4_mxfp4_moe.py 与 fp8.py 对 DSv4/DSR1 改用 moe_shuffle_scale（n32k4 scale 布局）与 moe_shuffle_weight（GUGU 交错布局），并为 gfx1250 传 GateMode.INTERLEAVE；quark_w4a4_mxfp4.py 对无可执行 fp4 GEMM 的 dense linear 在加载期用 _dequant_mxfp4_to_bf16 反量化到 bf16。
4. MLA 注意力与 DeepSeek 系列修复：新增 triton_qk_rmsnorm.py 提供 fused_qk_rmsnorm_triton / fused_qk_rmsnorm_torch；forward_mla_rocm.py 在 AITER_FORCE_A8W4=1 时替换 aiter 的 fused_qk_rmsnorm_bf16，并扩大 RoPE 提前应用的判断条件以覆盖非融合注意力后端；DSV4 paged_decode.py 的 aiter import 移到 gfx1250 门后；mhc.py 增加 torch/triton 的 HC split sinkhorn 回退；fp8_kernel.py 新增 _w8a8_block_fp8_matmul_gfx1250。此外 extend_attention.py 与 decode_attention.py 的 fp8 dot workaround 被 IS_GFX1250 constexpr 门控。
5. 测试配套：新增三个 MI45x GSM8K 精度测试文件（test_deepseek_v4_flash_eval_mi45x.py、test_deepseek_r1_0528_mxfp4_eval_mi45x.py、test_gpt_oss_w4a8_mxfp4_eval_mi45x.py），注册为 nightly-amd 套件，通过 ModelConfig 封装各模型的启动参数与环境变量矩阵（如 SGLANG_MOE_SHUFFLE_GFX1250、AITER_FORCE_A8W4、SGLANG_USE_AITER_MOE_GU_ITLV 等）。依赖侧将 compressed-tensors 按 ROCm 代拆分为 rocm_legacy（==0.15.0）与 rocm_rock（==0.16.0）两个 extra，runtime_common 保留裸依赖以兼容无 wheel 的非 AMD 平台。

关键文件：

- python/sglang/srt/layers/moe/fused_moe_triton/aiter_mxfp4_w4a8_moe.py（模块 MoE 执行；类别 source；类型 core-logic；符号 _import_aiter_w4a8, _interleave_gate_up, prepare_w4a8_gfx1250_weights, aiter_w4a8_gfx1250_forward）：gfx1250 专属的 W4A8 MXFP4 MoE 执行路径核心，解决 aiter CK/ASM fused_moe 产出乱码问题，含 TDM 禁用、手动 gather 与 MX scale 非 swizzle 两个关键平台规避。
- python/sglang/srt/layers/quantization/quark/schemes/quark_w4a8_mxfp4_moe.py（模块 量化适配；类别 source；类型 dependency-wiring；符号 _process_weights_gfx1250）：在量化层侧接通 gfx1250 的 W4A8 MoE 布局（无 shuffle、非填充、contraction-major），并在 apply_weights 中直呼 aiter_w4a8_gfx1250_forward，是 GPT-OSS 路径的入口开关。
- python/sglang/srt/models/deepseek_common/attention_forward_methods/triton_qk_rmsnorm.py（模块 MLA 注意力；类别 source；类型 data-contract；符号 _rmsnorm_kernel, _rmsnorm, fused_qk_rmsnorm_triton, _rmsnorm_torch）：新增的自包含 Triton/Torch RMSNorm，替代 gfx1250 上无法 JIT 构建的 aiter fused_qk_rmsnorm 内核，是 DSv4/DSR1 MLA 路径不崩溃的前提。

- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla_rocm.py（模块 MLA 注意力；类别 source；类型 data-contract；符号 fused_qk_rmsnorm_bf16）：MLA ROCm 入口按 AITER_FORCE_A8W4 切换 fused_qk_rmsnorm 实现，并调整 RoPE 应用条件以兼容非融合注意力后端，直接决定 DeepSeek 系列在 gfx1250 上的可用性。
- python/sglang/srt/layers/quantization/quark/schemes/quark_w4a4_mxfp4_moe.py（模块 量化适配；类别 source；类型 core-logic）：DSR1/DSV4 的 MoE 权重与 scale 布局在 gfx1250 上切换为 n32k4 + GUGU，并传 GateMode.INTERLEAVE，是防止静默乱码的关键修复。
- python/sglang/srt/layers/quantization/quark/schemes/quark_w4a4_mxfp4.py（模块 量化适配；类别 source；类型 core-logic；符号 _dequant_mxfp4_to_bf16）：gfx1250 缺少 fp4 WMMA scale 指令，无法执行 fp4 x fp4 GEMM；该文件在 AITER_FORCE_A8W4 下将 MXFP4 权重反量化为 bf16 走普通 linear，是 dense 层正确性的兜底。
- python/sglang/srt/layers/quantization/fp8.py（模块 FP8 量化；类别 source；类型 dependency-wiring）：DSv4 等模型的 block-quant MoE 权重 shuffle 在 gfx1250 上分支到 moe_shuffle_*，并让非 gfx1250 的 AITER_FORCE_A8W4 路径也能正确 shuffle，影响面较广。
- test/registered/amd/accuracy/mi45x/test_deepseek_v4_flash_eval_mi45x.py（模块 精度测试；类别 test；类型 test-coverage；符号 ModelConfig, run_gsm8k_benchmark, few_shot_gsm8k, TestDeepSeekV4FlashEvalMI45x）：MI45x 上 DSv4-Flash 的夜间精度测试，包含 attention-backend dsv4、统一 KV triton、AITER A8W4 等完整环境矩阵，是 gfx1250 支持的验收门禁。
- test/registered/amd/accuracy/mi45x/test_deepseek_r1_0528_mxfp4_eval_mi45x.py（模块 精度测试；类别 test；类型 test-coverage；符号 ModelConfig, run_gsm8k_benchmark, few_shot_gsm8k, TestDeepSeekR10528MXFP4EvalMI45x）：MI45x 上 DeepSeek-R1-0528-MXFP4 的夜间精度测试，覆盖 SGLANG_MOE_SHUFFLE_GFX1250 等 gfx1250 专有开关。
- test/registered/amd/accuracy/mi45x/test_gpt_oss_w4a8_mxfp4_eval_mi45x.py（模块 精度测试；类别 test；类型 test-coverage；符号 ModelConfig, run_gsm8k_benchmark, few_shot_gsm8k, TestGptOssW4A8Mxfp4EvalMI45x）：MI45x 上 GPT-OSS W4A8 MXFP4-FP8 的夜间精度测试，验收 aiter_mxfp4_w4a8_moe.py 新路径，阈值 0.79 是三模型中最高的。
- docker/rocm.Dockerfile（模块 镜像构建；类别 infra；类型 infrastructure）：新增 gfx1250-rocm1000 镜像阶段并保留多 flavor 分支，最终在 gfx1250 镜像中构建上游 Triton；这是本 PR 的部署入口，改动大量集中在 Triton/MORI/amdsmi 构建逻辑。
- scripts/ci/amd/amd_ci_install_dependency.sh（模块 CI 脚本；类别 infra；类型 infrastructure）：AMD CI 依赖安装器需识别 gfx1250 与 ROCm 10 版本，移除 ROCm 7.14 时代的错误探测，确保未来 MI45x runner 重建依赖时与发布镜像一致。

关键符号：_import_aiter_w4a8, _interleave_gate_up, prepare_w4a8_gfx1250_weights, aiter_w4a8_gfx1250_forward, fused_qk_rmsnorm_triton, fused_qk_rmsnorm_torch, _rmsnorm_kernel, fused_qk_rmsnorm_bf16, _process_weights_gfx1250,

_dequant_mxfp4_to_bf16, is_gfx1250_supported,
_w8a8_block_fp8_matmul_gfx1250,run_gsm8k_benchmark

关键源码片段

python/sglang/srt/layers/moe/fused_moe_triton/aiter_mxfp4_w4a8_moe.py

gfx1250 专属的 W4A8 MXFP4 MoE 执行路径核心，解决 aiter CK/ASM fused_moe 产出乱码问题，含 TDM 禁用、手动 gather 与 MX scale 非 swizzle 两个关键平台规避。

```
# gfx1250（RDNA / gfx12）专属的 MXFP4 权重 / FP8 激活（W4A8）融合 MoE 执行路径。
# 背景：gfx1250 上 aiter 的 CK/ASM fused_moe 对 GPT-OSS MXFP4 W4A8 布局会产出乱码，
# 因此改走 aiter 的 triton moe_gemm_a8w4 内核（gfx950 同款内核）。
# 该路径还需处理两个 gfx1250 特有的怪癖（与 vLLM enablement 一致）：
# 1. 内核内 TDM gather 在 gfx1250 编译失败（TDM gather dst must be 2D），
# 故禁用 TDM 路由，改为在 torch 中先把激活行按 expert 排序 gather 好；
# 2. gfx1250 的 moe_gemm_a8w4 会把 CDNA4 swizzled 的 MX scale 读成乱码，
# 因此权重 scale 保持非 swizzle 并传 swizzle_mx_scale=None。

def _interleave_gate_up(t: torch.Tensor) -> torch.Tensor:
    # 把分离式 [gate_0.., up_0..] 布局转成 moe_gemm_a8w4 融合 SwiGLU 期望的
    # 交错式 [gate_0, up_0, gate_1, up_1, ...]（gate 在偶数 lane，up 在奇数 lane）。
    e, two_i = t.shape[0], t.shape[1]
    i = two_i // 2
    rest = t.shape[2:]
    t = t.view(e, 2, i, *rest)
    perm = (0, 2, 1) + tuple(range(3, t.dim()))
    return t.permute(*perm).reshape(e, two_i, *rest).contiguous()


def aiter_w4a8_gfx1250_forward(
    hidden_states: torch.Tensor,
    router_logits: torch.Tensor,
    topk: int,
    w13_weight: torch.Tensor,
    w13_weight_scale: torch.Tensor,
    w13_weight_bias: torch.Tensor,
    a13_scale: torch.Tensor,
    w2_weight: torch.Tensor,
    w2_weight_scale: torch.Tensor,
    w2_weight_bias: torch.Tensor,
    a2_scale: torch.Tensor,
    gemm1_alpha: float,
    gemm1_limit: float,
    renormalize: bool = True,
    apply_router_weight_on_input: bool = False,
) -> torch.Tensor:
    # 输入 w*/scale/bias 必须是 prepare_w4a8_gfx1250_weights 产出的布局；
    # a13_scale / a2_scale 是 gate_up_proj 与 down_proj 的静态 per-tensor FP8 激活 scale。
    imported = _import_aiter_w4a8()
```

```
    if imported is None:
        raise RuntimeError(
            "aiter triton W4A8 MoE (moe_gemm_a8w4) is required for the gfx1250 "
            "GPT-OSS MXFP4 path but was not found in the installed aiter build."
        )
    routing, moe_gemm_a8w4, downcast_to_static_fp8 = imported

    assert hidden_states.dtype == torch.bfloat16

    # aiter 路由基于原始 router logits。renormalize=True（GPT-OSS）等价于在
    # 内核内对 top-k 选择做 softmax（sm_first=False）。
    routing_data, gather_idx, scatter_idx = routing(
        router_logits, topk, sm_first=not renormalize
    )
    gammas = routing_data.gate_scal

    # gfx1250：内核内 gather 不可用，向 moe_gemm_a8w4 传 gather_indx=None 并自行 gather。
    gather_src = gather_idx.to(torch.long) // topk
    x = hidden_states[gather_src]
    if apply_router_weight_on_input:
        # 路由权重必须在量化前的 bf16 下作用到输入。
        x = x * gammas[:, None].to(x.dtype)
    x_fp8 = downcast_to_static_fp8(x, a13_scale)

    # GEMM1：FP8 激活 x MXFP4 权重，融合 SwiGLU，并用 down_proj 的激活 scale
    # 把中间结果重量化为 FP8，便于 GEMM2 直接消费。
    intermediate_cache1 = moe_gemm_a8w4(
        x_fp8, w13_weight, None, w13_weight_scale, a13_scale, a2_scale,
        w13_weight_bias, routing_data,
        gather_indx=None, scatter_indx=None, gammas=None,
        swizzle_mx_scale=None, out_dtype=x_fp8.dtype,
        apply_swiglu=True, alpha=gemm1_alpha, limit=gemm1_limit,
    )

    # GEMM2：down 投影后按 scatter_idx 写回 token 顺序；若路由权重未提前作用，
    # 在此处用 gammas 恢复。
    intermediate_cache3 = moe_gemm_a8w4(
        intermediate_cache1, w2_weight, None, w2_weight_scale, a2_scale, None,
        w2_weight_bias, routing_data,
        gather_indx=None, scatter_indx=scatter_idx,
        gammas=None if apply_router_weight_on_input else gammas,
        swizzle_mx_scale=None, out_dtype=torch.bfloat16,
    )

    return intermediate_cache3.contiguous()
```

python/sglang/srt/models/deepseek_common/attention_forward_methods/triton_qk_rmsnorm.py

新增的自包含 Triton/Torch RMSNorm，替代 gfx1250 上无法 JIT 构建的 aiter fused_qk_rmsnorm 内核，是 DSv4/DSR1 MLA 路径不崩溃的前提。

```
"""gfx1250 上 aiter 的 module_fused_qk_norm_rope_cache_quant_shuffle 内核无法 JIT
构建（其 rope_common.h / ck_tile/vec_convert.h 与本镜像的 composable_kernel
不兼容），会导致 MLA 第一次 forward 直接崩溃。quant_type=No 路径退化为纯
RMSNorm，因此这里提供一个自包含的 Triton 实现：fp32 行方差、rsqrt(var + eps)、
乘以 weight 后转回原 dtype。"""

@triton.jit
def _rmsnorm_kernel(x_ptr, w_ptr, out_ptr, row_stride, N, eps, BLOCK_SIZE: tl.constexpr):
    # 每行一个 program：先按行求方差，再统一缩放并写回。
    row = tl.program_id(0)
    x_row = x_ptr + row * row_stride
    out_row = out_ptr + row * row_stride
    cols = tl.arange(0, BLOCK_SIZE)
    mask = cols < N
    x = tl.load(x_row + cols, mask=mask, other=0.0).to(tl.float32)
    var = tl.sum(x * x, axis=0) / N
    rstd = 1.0 / tl.sqrt(var + eps)
    w = tl.load(w_ptr + cols, mask=mask, other=0.0).to(tl.float32)
    y = x * rstd * w
    tl.store(out_row + cols, y.to(out_row.dtype.element_ty), mask=mask)


def _rmsnorm(x: torch.Tensor, weight: torch.Tensor, eps: float) -> torch.Tensor:
    # 展平成 [-1, N] 后按行启动 kernel，最后恢复原始形状。
    orig_shape = x.shape
    N = orig_shape[-1]
    x2d = x.reshape(-1, N).contiguous()
    out = torch.empty_like(x2d)
    M = x2d.shape[0]
    if M == 0:
        return out.reshape(orig_shape)
    BLOCK_SIZE = triton.next_power_of_2(N)
    num_warps = min(max(BLOCK_SIZE // 256, 1), 16)
    _rmsnorm_kernel[(M,)](x2d, weight, out, x2d.stride(0), N, float(eps),
                          BLOCK_SIZE=BLOCK_SIZE, num_warps=num_warps)
    return out.reshape(orig_shape)


def fused_qk_rmsnorm_triton(q, q_weight, q_eps, k, k_weight, k_eps):
    # 签名与返回约定对齐 aiter fused_qk_rmsnorm shim，forward_mla 可无感替换。
    q_out = _rmsnorm(q, q_weight, q_eps)
    k_out = _rmsnorm(k, k_weight, k_eps)
    return q_out, k_out
```

python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla_rocm.py

MLA ROCm 入口按 AITER_FORCE_A8W4 切换 fused_qk_rmsnorm 实现，并调整 RoPE 应用条件以兼容非融合注意力后端，直接决定 DeepSeek 系列在 gfx1250 上的可用性。

```
if _use_aiter:
    # gfx1250 上 aiter 的融合 QK Norm 内核 JIT 构建失败，会炸掉第一次 MLA forward。
    # 该路径只是纯 RMSNorm（quant_type=No），因此当 AITER_FORCE_A8W4（gfx1250
    # 工作模式）开启时，用自包含的 Triton/Torch 实现替换 aiter 内核。
    if get_bool_env_var("AITER_FORCE_A8W4", "false"):
        if get_bool_env_var("SGLANG_QK_RMSNORM_TORCH", "false"):
            from sglang.srt.models.deepseek_common.attention_forward_methods.triton_qk_
            rmsnorm import (
                fused_qk_rmsnorm_torch as fused_qk_rmsnorm_bf16,
            )
        else:
            from sglang.srt.models.deepseek_common.attention_forward_methods.triton_qk_
            rmsnorm import (
                fused_qk_rmsnorm_triton as fused_qk_rmsnorm_bf16,
            )
    else:
        # 非 gfx1250 的 aiter 路径：先探测 aiter#2958 之后的新统一入口
        #  （in-place、kwarg-only、无返回），失败再回退旧符号，兼容新旧 aiter。
        try:
            from aiter.ops.enum import QuantType as _AiterQuantType
            from aiter.ops.fused_qk_rmsnorm_group_quant import (
                fused_qk_rmsnorm as _aiter_fused_qk_rmsnorm_unified,
            )

            def fused_qk_rmsnorm_bf16(q, q_weight, q_eps, k, k_weight, k_eps):
                q_out = torch.empty_like(q)
                k_out = torch.empty_like(k)
                _aiter_fused_qk_rmsnorm_unified(
                    q_out_quantized=q_out, k_out=k_out, q=q,
                    q_weight=q_weight, q_epsilon=q_eps,
                    k=k, k_weight=k_weight, k_epsilon=k_eps,
                    quant_type=_AiterQuantType.No,
                )
                return q_out, k_out
        except ImportError:
            from aiter.ops.fused_qk_norm_rope_cache_quant import (
                fused_qk_rmsnorm as fused_qk_rmsnorm_bf16,
            )
```

评论区精华

核心讨论集中在 mem_fraction_static 守卫的归属与回归风险。作者 yctseng0211 在评论中说明："resolved conflict attention_hook.py: the aiter mem_fraction_static scaling now honors an explicitly set value instead of shrinking it. Carried over from #32043 ... Not gfx1250-specific: the 0.85 heuristic can push the static budget below the model-weight footprint on a nearly full GPU and break KV-cache allocation. Happy to split this into

its own PR if preferred." 并记录了该守卫在 #32754 上曾被 bingxche 两次标记为 regression，且 #36656 当天独立复现同一失败模式。另一个关键交锋是 PR-base（CUDA）失败归因：作者对比 main 上同期 run（33308180355）后列出 test_fp8_utils.py、test_deepseek_v4.py 三个失败用例，指认 #37086（将 is_sm90/is_sm100/is_hip 路由到 get_platform()）为引入者，结论是 "Not from this PR — this one only touches AMD/ROCm paths."；HaiShaw 最终认可："gfx specific changes, failed PR Test Base were known issues."。此外 commit 历史中 9626ada/927dc95 两次提交记录了 fp8 dot workaround 曾因缺平台守卫而影响 CUDA 与非 gfx1250 ROCm 精度，随后以 IS_GFX1250 constexpr 门控修正；compressed-tensors 的 extra 拆分也曾因 MLX/MPS、HPU、MUSA 无 0.15.0 wheel 导致 ResolutionImpossible 而被回撤为裸依赖 + 平台级 pin。

- mem_fraction_static aiter 守卫的来源与回归风险 (design): 合入当前版本（功能上与 #32754 原版等价：同样的 _raw_input 谓词与 warn/else 分支），并在评论中留下 traceability；作者表示愿意拆分为独立 PR。
- PR-base（CUDA）CI 失败是否由本 PR 引入 (question): HaiShaw 确认 "gfx specific changes, failed PR Test Base were known issues."，判定非本 PR 引入。
- gfx1250 fp8 dot workaround 的平台守卫 (correctness): 所有受影响的 dot 位点已加 IS_GFX1250 constexpr 守卫，gfx1250 保留 workaround，其余平台恢复原精度。
- compressed-tensors 依赖按 ROCm SDK 代拆分 (design): 最终形态：runtime_common 保留裸 compressed-tensors，AMD 构建由 rocm_legacy/rocm_rock 精确定 pin。

风险与影响

- 风险：
  1. 平台守卫风险（已发生过一次）：9626ada/927dc95 显示最初的 fp8 dot workaround 无 IS_GFX1250 守卫，一度改变 CUDA 与 gfx950 的 fp8 matmul 精度。最终代码虽已加 constexpr，但 forward_mla_rocm.py 等路径仍以环境变量（AITER_FORCE_A8W4、SGLANG_QK_RMSNORM_TORCH）而非设备名做开关，在 gfx950 等平台误设会绕过 aiter 优化路径。
  2. 量化布局静默错误：quark_w4a4_mxfp4_moe.py 注释明确 "Using the wrong layout silently corrupts the dequant scales."。gfx1250 走 n32k4/GUGU 布局、gfx950 走 e8m0_shuffle 布局，选择依赖 is_gfx1250_supported() 的设备探测，探测失败或新架构默认值错误时不会报错而只会产出垃圾结果。
  3. 反量化回退的覆盖面：quark_w4a4_mxfp4.py 的 _dequant_mxfp4_to_bf16 把 MXFP4 linear 权重转 bf16，内存成本上升；dequantized_bf16 分支对 tuple 激活只取 x[0]，若未来 MLA 投影被纳入该量化方案会隐藏错误。
  4. CI 覆盖空洞：commit 记录 "There is no gfx1250 runner"，三个精度测试为 nightly 且单测约 3600s，缺少 gfx1250 常驻 runner 前回归只能靠人工触发。
  5. 依赖与构建复杂度：compressed-tensors 双 extra 拆分曾引发非 AMD 平台 ResolutionImpossible；rocm.Dockerfile 多 flavor 分支（rocm720/rocm7_14/rocm1000）与 Triton 源码构建使镜像维护面显著增大。
  6. 批量合入风险：101 commits 中大量 origin/main 合并与手工冲突解决（attention_hook.py、paged_decode.py 等），存在夹带非 gfx1250 改动的可能，

mem_fraction_static guard 即是一例（已被显式记录与 #36656 同源）。 - 影响：用户侧：MI45x（gfx1250/Helios）用户在 ROCm 10.0.0 GA 上可运行 GPT-OSS-120b、DeepSeek-R1-0528-MXFP4、DSv4-Flash，GSM8K 准确率分别达 0.845、0.951、0.929；Wan2.2 与 FLUX.2 扩散模型也在同一镜像上验证通过。系统侧：新增 3 个夜间精度测试（合计约 2 小时）、gfx1250-rocm1000 Docker 目标与 CI 检测，AMD 支持矩阵从 gfx942/gfx950 扩展到 gfx1250。团队侧：建立了"平台缺陷 -> 局部绕过 + 环境 变量开关 + 夜间精度验证"的 AMD 适配范式；同时 mem_fraction_static 守卫会影响所 有启用 aiter 的 ROCm 用户（显式设置该参数时不再被 0.85 缩放，避免 KV-cache 分 配被压垮）。 - 风险标记：平台守卫缺失曾影响 CUDA 精度，环境变量开关较多且易误 设，gfx1250 CI runner 尚缺，量化 scale 布局选错会静默污染，101 commits 大批量 合并

关联脉络

- PR #32754 [AMD] Enable gfx1250 Support: 本 PR 的直接前身（closed unmerged）：mem_fraction_static guard、gfx1250 内核修复与 GPT-OSS 路径均源于此，PR body 与 issue 评论明确引用。
- PR #36434 ROCm 10 release images: ROCm 10.0.0 GA 镜像通道的基础工作（PR body 括号标注），本 PR 在其上新增 gfx1250-rocm1000 目标并沿用 GA wheel 约定。
- PR #37086 [Config] Round 5.1: the published-side readers ask the bags, and a platform fact gets one address: 作者将其认定为 main 上 CUDA 精度测试失败的引入者（改动 layers/quantization/fp8.py 与 dsv4/indexer.py 的平台事实路由），与本 PR 的 CI 归因调查直接相关。