PR #33313 完整报告

sgl-project/sglang

[AMD] DeepSeek-V4: route decode wo_a bf16 batched matmul to aiter batched_gemm_bf16

合并时间：2026-08-19 18:04

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33313

执行摘要

- 一句话：DSV4 *decode* wo_a 路由 aiter GEMM，AMD gfx95 opt-in
- 推荐动作：值得精读。重点看三点：(1) 热路径 kernel 路由设计——为何把资格判定与 kernel 导入收敛到模块 import 期；(2) 两段式 fallback（import 失败 vs 运行期失败）如何避免关键路径重试与日志风暴；(3) 性能验证的诚实方法论——内核 trace 证明替换、数值 bit-check、GSM8K 精度、端到端 A/B 分层呈现，且 review 明确要求把「对齐」与「提速」两种论断分开。对 AMD/DeepSeek-V4 相关工程师是必读，其他人可作为 kernel 改动类 PR 的验收范本。

功能与动机

PR body 用 torch-profiler（External-id + Python 调用栈）对 DSV4 *decode* attention 区域逐 kernel 归因，发现 Cijk_Alik_Bljk_*_MT16x16x1024 Tensile 内核由 aten::bmm 从 wo_a bf16 einsum 启动，是解码注意力区域中较大的内核之一；而参考 ATOM 栈上同一算子是 aiter 的 _batched_gemm_bf16_kernel。因此把 sglang 路由到 aiter 内核，使 kernel 选择与参考栈一致。作者明确声明这是「1:1 kernel replacement / ATOM-alignment change，不是大吞吐收益」。

实现拆解

1. 新增 opt-in 环境开关（python/sglang/srt/environ.py）：在 Kernels and indexer 区块新增 SGLANG_OPT_USE_AITER_BATCHED_GEMM = EnvBool(False)，默认关闭；命名由最初 SGLANG_OPT_WO_A_AITER_BATCHED_GEMM 按 HaiShaw 意见改为通用风格，与同族 SGLANG_OPT_* 开关一致。
2. 模块导入期一次性判定与预导入（deepseek_v4.py 模块级）：_wo_a_aiter_gemm_eligible(flag, use_aiter, is_hip, is_gfx95) 把 opt-in 开关、全局 SGLANG_USE_AITER、HIP、gfx95 四个门槛折叠为一个 bool，import 时算入 _wo_a_aiter_batched_gemm_enabled；通过后 try 预导入 aiter.ops.triton.gemm.batched.batched_gemm_bf16 为 _wo_a_batched_gemm_bf16，导入失败则当场置 False 并告警一次。这样 *decode* 每层每 token 的关键路径不付出 EnvBool.get() 与函数内 import 的开销。
3. 运行期分发 helper _apply_wo_a_bf16_matmul(o, wo_a, is_decode)：仅当 is_decode 为真、路由已启用且未被禁用时走 aiter 内核——将 o 转置为 [G, T, D] 作为 batch 维、wo_a 为 [G, R, D]，利用 Y[i] = X[i] @ W[i]^T 得到 [G, T, R] 再转回 [T, G, R]；任何异常

把进程级 _wo_a_aiter_batched_gemm_disabled 置位后回退 einsum，只告警一次。prefill、任何门控关闭或失败后一律走原始 torch.einsum("tgd,grd->tgr", ...)。

4. 接入 MLA 前向：deepseek_v4.py 中原 bf16 else 分支（wo_a = self.wo_a.weight.view(...) + einsum）改为调用 helper，并传入 forward_batch.forward_mode.is_decode()；fp8/deep_gemm 分支、CUDA 与非 gfx95 路径完全不动。
5. 测试与 CI 配套：新增 test/registered/unit/models/test_deepseek_v4_amd_wo_a_bf16.py（203 行），注册到 stage-b-test-1-gpu-small-amd-mi35x（只有 gfx95 真机才会执行 kernel 用例），覆盖静态资格表驱动判定、decode/prefill 分发门控（mock 替换缓存全局与 kernel 引用）、运行期失败一次性回退，以及真机上 aiter 与 einsum 的 bf16 数值一致性（max rel err <= 5e-4）。测试直接驱动模块缓存全局，并在 setUp 中重置进程级禁用标记避免用例间串扰。

关键文件：

- python/sglang/srt/models/deepseek_v4.py（模块 模型实现；类别 source；类型 core-logic；符号 _wo_a_aiter_gemm_eligible, _apply_wo_a_bf16_matmul, _wo_a_aiter_batched_gemm_enabled, _wo_a_batched_gemm_bf16）：核心改动文件：新增模块导入期资格判定与 aiter kernel 预导入，新增 _apply_wo_a_bf16_matmul 在 decode 路径完成 kernel 路由与两段式 fallback，原 bf16 einsum 分支改为调用 helper，是整次变更的入口与核心逻辑所在。
- test/registered/unit/models/test_deepseek_v4_amd_wo_a_bf16.py（模块 回归测试；类别 test；类型 test-coverage；符号 TestWoABf16BatchedGemm, test_eligibility_gating, test_dispatch_gating, test_aiter_matches_einsum_across_shapes）：新增 AMD mi35x 回归测试（203 行），覆盖资格判定、分发门控、失败回退与 bf16 数值一致性；注册套件在 review 中被专门纠正到 stage-b-test-1-gpu-small-amd-mi35x，是 kernel 真机路径的唯一 CI 保障。
- python/sglang/srt/environ.py（模块 配置开关；类别 source；类型 configuration；符号 SGLANG_OPT_USE_AITER_BATCHED_GEMM）：新增默认关闭的 SGLANG_OPT_USE_AITER_BATCHED_GEMM 环境开关，是本次路由的 opt-in 入口；命名经 review 敲定，与同族 SGLANG_OPT_* 保持一致。

关键符号：_apply_wo_a_bf16_matmul, _wo_a_aiter_gemm_eligible, test_eligibility_gating, test_dispatch_gating, test_aiter_matches_einsum_across_shapes

关键源码片段

python/sglang/srt/models/deepseek_v4.py

核心改动文件：新增模块导入期资格判定与 aiter kernel 预导入，新增 _apply_wo_a_bf16_matmul 在 decode 路径完成 kernel 路由与两段式 fallback，原 bf16 einsum 分支改为调用 helper，是整次变更的入口与核心逻辑所在。

```
# ---- 模块导入期一次性判定与预导入（deepseek_v4.py 模块级）----
# decode 阶段的 wo_a matmul 每层、每个 token 都执行，属于关键路径；
# 因此把 EnvBool.get() 与 aiter 内核 import 都收敛到 import 期，只做一次。
def _wo_a_aiter_gemm_eligible(
```

```
    flag: bool, use_aiter: bool, is_hip: bool, is_gfx95: bool
) -> bool:
    """静态资格判定：opt-in 开关、全局 SGLANG_USE_AITER、HIP、gfx95 全部满足才启用。"""
    return bool(flag and use_aiter and is_hip and is_gfx95)


_wo_a_aiter_batched_gemm_enabled = _wo_a_aiter_gemm_eligible(
    envs.SGLANG_OPT_USE_AITER_BATCHED_GEMM.get(),
    _use_aiter,
    _is_hip,
    _is_gfx95_supported,
)

_wo_a_batched_gemm_bf16 = None
if _wo_a_aiter_batched_gemm_enabled:
    try:
        from aiter.ops.triton.gemm.batched.batched_gemm_bf16 import (
            batched_gemm_bf16 as _wo_a_batched_gemm_bf16,
        )
    except Exception as err:
        # 导入失败：本次进程内禁用路由并只告警一次，避免每个 step 重试
        _wo_a_aiter_batched_gemm_enabled = False
        logger.warning(
            "aiter wo_a batched_gemm_bf16 import failed; "
            "fall back to einsum for this process: %s",
            err,
        )

# 运行期 kernel 异常的一次性禁用标记：首次失败后整个进程回退 einsum
_wo_a_aiter_batched_gemm_disabled = False


def _apply_wo_a_bf16_matmul(
    o: torch.Tensor, wo_a: torch.Tensor, is_decode: bool
) -> torch.Tensor:
    """wo_a（attn 输出 -> o_proj 低秩）bf16 batched matmul。

    输入 o: [T, G, D]（token, group, head_dim），wo_a: [G, R, D]
    （group, o_lora_rank, head_dim），输出: [T, G, R]。
    """
    global _wo_a_aiter_batched_gemm_disabled
    if (
        is_decode
        and _wo_a_aiter_batched_gemm_enabled
        and not _wo_a_aiter_batched_gemm_disabled
    ):
        try:
            # aiter batched_gemm_bf16 语义为 Y[i] = X[i] @ W[i]^T;
            # 这里以 group 为 batch 维：XQ = o.transpose(0, 1) -> [G, T, D],
```

```
            # WQ = wo_a -> [G, R, D]，得到 [G, T, R] 再转置回 [T, G, R]
            xq = o.transpose(0, 1).contiguous()
            y = _wo_a_batched_gemm_bf16(xq, wo_a, dtype=torch.bfloat16)
            return y.transpose(0, 1).contiguous()
        except Exception as err:
            # 首次运行期失败：禁用它并回退 einsum，后续不再重试、不再刷日志
            _wo_a_aiter_batched_gemm_disabled = True
            logger.warning(
                "aiter wo_a batched_gemm_bf16 failed; "
                "disabling the reroute: %s",
                err,
            )
    # prefill、任何门控关闭或失败回退都走数值等价的原始 einsum
    return torch.einsum("tgd,grd->tgr", o, wo_a)
```

test/registered/unit/models/test_deepseek_v4_amd_wo_a_bf16.py

新增 AMD mi35x 回归测试（203 行），覆盖资格判定、分发门控、失败回退与 bf16 数值一致性；注册套件在 review 中被专门纠正到 stage-b-test-1-gpu-small-amd-mi35x，是 kernel 真机路径的唯一 CI 保障。

```
# 测试直接驱动 deepseek_v4 模块的缓存全局，而不是每次调 EnvBool.get()/import。
@unittest.skipUnless(is_hip(), "wo_a batched_gemm_bf16 routing requires ROCm")
class TestWoABf16BatchedGemm(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        # 只在 GPU runner 上导入重型模型模块
        from sglang.srt.models import deepseek_v4 as dsv4

        cls.dsv4 = dsv4
        cls.device = "cuda" # torch 会把 "cuda" 映射到 ROCm HIP 设备

    def setUp(self):
        torch.manual_seed(0)
        # 进程级禁用标记是全局状态；重置它避免用例之间互相污染
        self.dsv4._wo_a_aiter_batched_gemm_disabled = False

    def test_eligibility_gating(self):
        # 路由只在 flag + SGLANG_USE_AITER + HIP + gfx95 全部满足时激活
        eligible = self.dsv4._wo_a_aiter_gemm_eligible
        base = dict(flag=True, use_aiter=True, is_hip=True, is_gfx95=True)
        self.assertTrue(eligible(**base))
        for off in ("flag", "use_aiter", "is_hip", "is_gfx95"):
            with self.subTest(disabled=off):
                self.assertFalse(eligible(**{**base, off: False}))

    def test_dispatch_gating(self):
        # 组合（enabled, is_decode）验证分发：仅 decode + enabled 命中 kernel，
        # prefill 与关闭时走 einsum，输出与 einsum 参考对比
        o, wo_a = self._rand(8, 4, 128, 32)
```

```
        ref = self._einsum(o, wo_a)

        def _fake_kernel(xq, w, dtype): # [G, T, D] @ [G, R, D]^T -> [G, T, R]
            _fake_kernel.calls += 1
            return torch.einsum("gtd,grd->gtr", xq, w).to(dtype)

        for enabled, is_decode, expect_kernel in (
            (True, True, True), # decode + 启用 -> aiter kernel
            (True, False, False), # prefill 保持 einsum
            (False, True, False), # 路由关闭 -> einsum
            (False, False, False),
        ):
            with self.subTest(enabled=enabled, is_decode=is_decode):
                _fake_kernel.calls = 0
                # 用 mock.patch.object 替换 dsv4 缓存中的 enabled 标记与
                # kernel 引用，再按 is_decode 取 forward_mode 分支调用 helper;
                # 具体 patch 与数值断言从略
                ...
```

评论区精华

核心交锋集中在 4 轮 review：

- HaiShaw 要求把开关改名为 SGLANG_OPT_USE_AITER_BATCHED_GEMM，并索要 --parallel 1319 的 GSM8K 精度数据与 V4-Pro GPTQ 前后对比；作者提交 641bc6f 改名并补充 1319 题数据（einsum 0.952 / aiter 0.946，同一波动带），可见评论中未见 GPTQ 数字的后续补充。
- kkHuang-amd 第 1 轮指出测试注册在错误的硬件套件（常规 AMD runner 上会 skip）以及持久性 aiter 失败会在每次调用重试；作者把注册迁移到 mi35x 套件，并加入进程级一次性禁用标记。
- kkHuang-amd 第 2 轮要求澄清 ~30μs/25μs 与 trace 中 2.04ms/2.07ms 的口径差异，指出 trace 不证明 kernel 时间下降；作者撤回非同一 trace 的估算，把论断重构为「kernel parity + 边际 2-3%」。
- kkHuang-amd 第 3 轮以 CHANGES_REQUESTED 拦截 merge 后丢失的 import 期缓存回归；提交 72b701b 修复后 APPROVED，并要求合并前重跑 exact-head ROCm workflow。
- 1am9trash 追问「kernel 时间持平但 e2e 有 2-3% 提升」的来源，作者答复为 host-side 分发 / 调度开销差异（einsum -> aten::bmm vs 直接内核启动），非设备端算力差异。
- 开关命名与精度实测要求 (style): 提交 641bc6f 完成改名；作者补充 5-shot GSM8K 1319 题数据（einsum 0.952 / aiter 0.946，同一波动带）；可见评论中未见 GPTQ 对比数字的后续补充。
- 功能测试注册在错误的 AMD 硬件套件 (testing): 注册迁移到 stage-b-test-1-gpu-small-amd-mi35x（register_amd_ci），kernel 真机路径进入 CI。
- 持久性 aiter 失败会反复重试并刷日志 (design): 新增进程级 _wo_a_aiter_batched_gemm_disabled：首次失败后整个进程回退 einsum 并只告警一次；测试在 setUp 中重置该标记避免串扰。

- 性能数据口径矛盾（µs vs ms）与吞吐声称 (performance): 作者撤回非同一 trace 上估算的 per-op 数据，重构为「kernel parity + 边际 2-3% 端到端」，PR 正文与 summary 同步修正，获 reviewer 认可。
- 合并 main 后 import 期缓存逻辑回归 (design): 提交 72b701b 恢复 import 期一次性资格判定与 kernel 预导入，*decode* 热路径只用缓存全局；复审后 APPROVED，并要求合并前重跑 *exact-head* ROCm workflow 确认 mi35x 测试实际执行。
- kernel 时间持平但 *e2e* 提升 ~2-3% 的来源 (question): 作者答复为 host-side 差异：旧路径 torch.einsum -> aten::bmm 有分发 / 调度开销，aiter 是直接内核启动，并非设备端算力差异；并建议后续补充 kernel 分解（非阻塞）。1am9trash 认可并 APPROVED。

风险与影响

- 风险:
  - 热路径回归风险：*decode* 每层 wo_a 现在多两次 transpose + contiguous() 拷贝（转 [G, T, D] 与回 [T, G, R]）；MI355X 上实测 parity，但换 shape、更高并发或首次触发 aiter JIT 编译时可能劣化。新开关默认关闭，受影响面可控。
  - 合并易碎点：10 次提交中 4 次 *merge* main，曾把 import 期缓存逻辑整体冲掉（kkHuang 在 CHANGES_REQUESTED 中拦截）；*environ.py* 是冲突高发区，未来合入需重点核对缓存相关代码。
  - 数值边界：bf16 reduction 顺序差异 max rel *err* <= 5*e*-4，GSM8K 0.952 vs 0.946 处于同一 fp4-MoE 波动带；但 0.6pt 差值无法完全归因（fp4-MoE 本身有 run-to-run 非确定性）。
  - CI 覆盖盲区：kernel 真机路径只在 *stage-b-test-1-gpu-small-amd-mi35x* 执行，普通 AMD runner 上直接 skip；若镜像缺 *aiter*，该测试会静默跳过。合并前需确认 *exact-head* ROCm workflow 真的执行了 *test_aiter_matches_einsum_across_shapes*。
  - 语义范围：SGLANG_OPT_USE_AITER_BATCHED_GEMM 命名较通用但当前只作用于 wo_a，未来若复用于其他算子需与 SGLANG_OPT_FP8_WO_A_GEMM 等开关理清边界。
- 影响:
  - 用户影响：默认零影响（开关关闭）；AMD *gfx950*（MI355X）上运行 DeepSeek-V4 的用户显式设置 SGLANG_OPT_USE_AITER_BATCHED_GEMM=1 可获约 2-3% 总 tok/s 的边际收益（或持平），并让 wo_a 的 kernel 选择与 ATOM 参考栈一致。
  - 系统影响：仅在 HIP + *gfx95* + *decode* 生效；不改变显存布局、KV *cache* 或数值契约（bf16 输出仍在容差内）；CUDA、*prefill*、非 *gfx95* 路径完全不触碰。
  - 团队影响：确立了 kernel 路由的样板模式（import 期缓存判定 + 进程级一次性回退 + *decode*-only 门控），并新增一个 mi35x 专属回归测试；性能表述的校准过程（*trace* 口径与 *e2e* 口径分离）值得团队在后续 PR 中复用。
  - 风险标记：解码热路径变更，合并冲突曾致缓存逻辑回归，依赖 *gfx95* 专属 *aiter* 内核，kernel 真机路径仅 mi35x CI 覆盖，opt-in 默认关闭

关联脉络

- PR #34859 Qwen3.8-27B Model Support: 同为 AMD MI355X 上的量化 kernel 优化路线（FP4/FP8 GEMV 选型），与 wo_a aiter 路由同属 AMD 平台 kernel 对齐工作，但未改动相同文件，关联较弱。