PR #43534 完整报告

vllm-project/vllm

[CPU][Perf] Enable fused kernels for GDN's gated delta rules

合并时间：2026-06-02 16:00

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43534

执行摘要

本 PR 将 CPU 上 Mamba GDN 注意力的计算从 Python 迁移至 C++ 融合内核，并为非 x86 架构添加 BLAS 回退路径，在 Neoverse-V2 上取得约 50% 的吞吐量提升。同时修复了 fused sigmoid 门控中的一个数值 bug，新增了完整测试套件。

功能与动机

PR 描述指出：'makes the gated delta rule impls from sglang-kernels ISA agnostic'，并统一 AMX 与其他 CPU ISA 的路径。对于缺少快速 brgemm 的 ISA（如非 x86），使用 OpenBLAS 或 PyTorch BLAS 回退。同时修复了 fused_sigmoid_gating_delta_rule_update 内核中 sigmoid 计算的 bug（beta 指针使用了错误索引）。此 PR 是在 #41025 中承诺的后续优化。

实现拆解

1. 移除旧 Python 实现并添加 C++ 融合内核：删除 recurrent_gated_delta_rule.py（223 行），避免 Python 层 GEMM 开销。在 fla.cpp 中新增 fused_sigmoid_gating_delta_rule_update_kernel_impl 和增强 chunk_gated_delta_rule_kernel_impl，将 gating、chunk decay 和 delta rule 计算融合为单个内核。
2. 实现 ISA 无关的 BLAS 回退层：新建 blas_gemm.h，提供 blas_gemm 重载。当定义了 VLLM_HAS_OPENBLAS（由 cmake 检测）时，直接调用 OpenBLAS 的 sbgemm_/sgemm_ 进行 bf16/float GEMM；否则回退到 PyTorch 的 gemm_no_downcast_stub（输出 fp32）。在 gemm.h 中添加编译时常量 brgemm_supported()，基于 AVX512BF16/AMX 指令集判断是否可用，并修改 can_use_brgemm 模板函数与之关联。
3. 改造 Python 调用入口：修改 gdn_attention.py，移除对旧 Python 函数 recurrent_gated_delta_rule 和 gdn_gating 的导入，改为直接调用 ops 中注册的 C++ 函数。同时删除了两层分支（AMX 专用路径与非 AMX 路径），统一为条件编译的内核。
4. 添加完整测试：新建 test_cpu_gdn_ops.py，包含三个参数化测试用例，覆盖 fused gdn gating、fused sigmoid gating delta rule update 和 chunk gated delta rule，与 Python 参考实现对比验证。测试配置了多种 batch 大小、序列长度和 head 维度，确保边界正确。
5. 更新构建与 CI：修改 cpu_extension.cmake 添加 OpenBLAS 检测和 VLLM_HAS_OPENBLAS 宏；在 CI 配置文件 cpu.yaml 和 run-cpu-test-arm.sh 中注册新测试路径，确保 ARM 和 x86 节点都能运行。

csrc/cpu/sgl-kernels/fla.cpp

核心 C++ 实现文件，添加了融合内核（fused_sigmoid_gating_delta_rule_update_kernel_impl 和增强 chunk_gated_delta_rule_kernel_impl），并使用编译时分支动态选择 brgemm 或 blas_gemm，是性能提升的关键。

```
// csrc/cpu/sgl-kernels/fla.cpp
// 展示 chunk_gated_delta_rule_kernel_impl 中的关键分支：
// 通过编译时 brgemm_supported() 决定使用 AMX brgemm 还是通用 BLAS
if constexpr (brgemm_supported()) {
    // AMX 路径：pack + brgemm
    pack_vnni<scalar_t>(k_transpose, curr_k_pad, chunk_size, qk_head_size, qk_head_size,
    chunk_size);
    at::native::cpublas::brgemm(
        chunk_size, chunk_size, qk_head_size, qk_head_size, chunk_size, chunk_size,
        false, curr_k_beta, k_transpose, curr_attn);
} else {
    // 非 x86 回退路径：调用 blas_gemm (BF16 GEMM 输出 FP32)
    blas_gemm(
        at::native::TransposeType::Transpose, // A = k_pad^T
        at::native::TransposeType::NoTranspose, // B = k_beta
        chunk_size, chunk_size, qk_head_size,
        1.0f,
        curr_k_pad, qk_head_size,
        curr_k_beta, qk_head_size,
        0.0f,
        curr_attn, chunk_size
    );
}
// 后续 attn = attn * decay_mask 逻辑保持不变
```

csrc/cpu/sgl-kernels/blas_gemm.h

新增的 BLAS 回退头文件，封装了 OpenBLAS 和 PyTorch 两种调用方式，确保在无 AMX 的 CPU 上仍能高效执行 GEMM。

```
// csrc/cpu/sgl-kernels/blas_gemm.h
// 提供 ISA 无关的 BLAS gemm 封装
#include <ATen/native/CPUBlas.h>

#if defined(VLLM_HAS_OPENBLAS)
// 直接链接 OpenBLAS 符号
inline void blas_gemm(... BFloat16 ...) {
    char transa_ = ..., transb_ = ...;
    int m_ = m, n_ = n, k_ = k;
    extern "C" void sbgemm_(...);
    sbgemm_(&transa_, &transb_, &m_, &n_, &k_, &alpha, a, &lda_, b, &ldb_, &beta, c, &ldc_);
}
inline void blas_gemm(... float ...) {
    // 类似，调用 sgemm_
```

```
}
inline void blas_gemm(... Half ...) {
    TORCH_CHECK(false, "CPU OpenBLAS hgemm is not available.");
}
#else
// 使用 PyTorch 的 cpublas stub，输出不降精度 (fp32)
template <typename scalar_t>
inline void blas_gemm(...) {
    auto gemm = at::native::cpublas::gemm_no_downcast_stub.DEFAULT;
    gemm(..., a, lda, b, ldb, beta, c, ldc);
}
#endif
```

csrc/cpu/sgl-kernels/gemm.h

添加了 brgemm_supported() 编译时检测和 CPU_CAPABILITY_AVX512 宏，控制 can_use_brgemm 返回值，是 ISA 无关性的关键。

```
// csrc/cpu/sgl-kernels/gemm.h
// 添加编译时 AMX 能力检测
#include "blas_gemm.h"  // 替换原来的 <ATen/native/CPUBlas.h>

#if defined(__AVX512F__) && defined(__AVX512BF16__) && defined(__AMX_BF16__)
#define CPU_CAPABILITY_AVX512
#endif

constexpr bool brgemm_supported() {
#if defined(CPU_CAPABILITY_AVX512)
    return true;
#else
    return false;
#endif
}

// 所有 can_use_brgemm 特化都加上 brgemm_supported() 条件
```

评论区精华

- BLAS fallback 类型不匹配争议：gemini-code-assist 指出 blas_gemm fallback 路径中输出缓冲为 float* 而 PyTorch stub 期望 scalar_t*。作者澄清已使用 gemm_no_downcast_stub，输出始终 fp32，不存在类型 mismatch。
- 测试数值稳定性质疑：bigPYJ1151 询问 GDN 测试是否稳定。作者解释稳定性得益于本 PR 修复的 sigmoid 计算 bug，且序列长度较短，多次种子测试均通过。
- 代码组织建议：bigPYJ1151 建议将 BLAS 函数移至单独头文件方便 sglang 同步，作者采纳创建了 blas_gemm.h。

风险与影响

- 风险：非 x86 回退路径仅在 CI ARM 节点测试，覆盖有限；旧 Python 实现完全移除后缺乏降级能力；OpenBLAS 接口可能因版本不匹配产生未定义符号，但依赖 PyTorch 内置库降低风险。
- 影响：x86 和 ARM CPU 用户无感升级即可获得 Mamba 模型大幅性能提升；团队需维护 blas_gemm.h 和 fla.cpp 中新的 BLAS 依赖，但代码复用度高。

关联脉络

本 PR 是基于 #41025 的后续优化（fadara01 在 Issue 评论中提及）。近期历史中还有针对 V2 model runner、FP8 KV cache 清零等 CPU 相关 bugfix，表明该项目正在持续优化 CPU 推理路径。未来计划进一步加速 causal conv（PR body 提及 upcoming PR）。