PR #23255 完整报告

sgl-project/sglang

[MUSA][18/N] Add MUSA-optimized kernel implementations for hot ops

合并时间：2026-05-08 11:38

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23255

执行摘要

- 一句话：为 MUSA GPU 添加 sgl-kernel 优化内核
- 推荐动作：值得精读——尤其是头文件组织方式（sgl_kernel_musa_ops.h）和条件编译策略，可作为跨架构支持的样板。建议作者补充单元测试（参考 test/registered/ 下的模式）并跟进 inter-block barrier 的 long-term 修复。

功能与动机

该 PR 是 issue #16565（Support Moore Threads GPU）的一部分，目标是为 SGLang 在 Moore Threads GPU 上运行提供基础内核支持。通过添加 MUSA 内核源文件并注册到 torch ops，使 sgl-kernel 能通过 setup_musa.py 构建并用于 LLM 推理的热点算子。

实现拆解

1. 新增 MUSA 内核源文件：在 sgl-kernel/csrc/musa/ 下添加 .mu 文件，包括 pos_encoding_contiguous.mu（旋转位置编码）、moe_gemv_swiglu.mu（融合 MoE GEMV）、ternary.mu（元素三元融合）、top_k_top_p_sampling.mu（采样），以及共同的辅助头文件 common.muh、dtype.muh。
2. 添加 MUSA 头文件：在 sgl-kernel/include/musa/ 下添加 integer_subbyte.h（子字节整数封装）、dispatch_utils.h（MUSA 分发宏），并在 sgl-kernel/include/ 下新增 sgl_kernel_musa_ops.h 声明所有 MUSA 算子的 C++ 接口。
3. 更新构建配置：在 sgl-kernel/setup_musa.py 中将新源码包含进编译流程。
4. 注册 Torch ops：在 common_extension_musa.cc 中使用 TORCH_LIBRARY_EXPAND 和 m.impl(..., torch::kMUSA, &func) 注册所有新算子，区分算子是否需要 MUSA 前缀。
5. 暴露 Python 接口：新建 python/sgl_kernel/musa.py 提供 Python 包装函数；修改 __init__.py 通过 hasattr(torch.version, "musa") 条件导入 MUSA 模块。注意：未包含单元测试或集成测试，仅依赖后续 CI 验证。

关键文件：

- sgl-kernel/python/sgl_kernel/musa.py（模块 Python API；类别 source；类型 core-logic；符号 musa_batched_rotary_embedding_contiguous, musa_rotary_embedding_contiguous, musa_fused_moe_gemv, musa_fused_gemv）：暴露 MUSA 算子的 Python 接口，包含量化感知的 fused_gemv 分发逻辑，是用户直接调用的入口。

- sgl-kernel/include/sgl_kernel_musa_ops.h（模块 C++ 头文件；类别 source；类型 dependency-wiring；符号 batched_rotary_embedding_contiguous, rotary_embedding_contiguous, fused_moe_gemv, musa_fused_gemv）：声明所有 MUSA 算子的 C++ 接口，供 common_extension_musa.cc 注册使用，是架构级边界文件。
- sgl-kernel/csrc/common_extension_musa.cc（模块 算子注册；类别 source；类型 core-logic；符号 musa_batched_rotary_embedding_contiguous, musa_rotary_embedding_contiguous, musa_fused_moe_gemv, musa_fused_gemv）：MUSA 算子的 Torch 注册入口，将 C++ 函数绑定到 torch.ops.sgl_kernel 命名空间。
- sgl-kernel/csrc/musa/moe_gemv_swiglu.mu（模块 MUSA 内核；类别 other；类型 dependency-wiring；符号 musa_gemv_kernel, fused_moe_gemv, musa_fused_gemv）：MUSA 核心内核实现之一，实现融合 MoE GEMV 和 SwiGLU 激活，包含大量模板和宏展开。
- sgl-kernel/python/sgl_kernel/__init__.py（模块 Python 入口；类别 source；类型 dependency-wiring）：添加条件导入分支 if hasattr(torch.version, "musa")，使 MUSA 模块在 MUSA 环境下自动加载。

关键符号：musa_batched_rotary_embedding_contiguous, musa_rotary_embedding_contiguous, musa_fused_moe_gemv, musa_fused_gemv, musa_fused_mul_add, batched_rotary_embedding_contiguous, rotary_embedding, fused_moe_gemv, musa_fused_gemv (C++), fused_mul_add, musa_top_k_top_p_sampling_from_probs

关键源码片段

sgl-kernel/python/sgl_kernel/musa.py

暴露 MUSA 算子的 Python 接口，包含量化感知的 fused_gemv 分发逻辑，是用户直接调用的入口。

```
# sgl-kernel/python/sgl_kernel/musa.py -- MUSA 算子 Python 接口
from typing import Optional
import torch

# ----------------------------------------------------------
# 旋转位置编码（batch 版本）
def musa_batched_rotary_embedding_contiguous(
    positions: torch.Tensor,
    query: torch.Tensor,
    key: torch.Tensor,
    head_size: int,
    cos_sin_cache: torch.Tensor,
    is_neox: bool,
    rot_dim: int,
    cos_sin_cache_offsets: torch.Tensor,
) -> None:
    # 直接委托底层 C++ 算子
    return torch.ops.sgl_kernel.musa_batched_rotary_embedding_contiguous(
```

```
        positions, query, key, head_size, cos_sin_cache,
        is_neox, rot_dim, cos_sin_cache_offsets)

# ----------------------------------------------------------
# 融合 GEMV（支持 fp8 分组 / w4a16 / 通用）
def musa_fused_gemv(
    x: torch.Tensor,
    qweight: torch.Tensor,
    x_scales: Optional[torch.Tensor] = None,
    qweight_scales: Optional[torch.Tensor] = None,
    use_swigelu: bool = False,
    use_rms_norm: bool = False,
    gamma: Optional[torch.Tensor] = None,
    eps: float = 1e-6,
):
    use_int4_w4a16 = False
    # 根据 swigelu 标志计算输出 shape（若启用，输出维度减半）
    out_shape = x.shape[:-1] + (
        qweight.shape[0] if not use_swigelu else qweight.shape[0] // 2,
    )
    assert not (use_swigelu and use_rms_norm), \
        "gemv only fused one activation (swigelu or rms_norm)!"

    # --- 路径 1：fp8 分组矩阵乘 ---
    if qweight.dtype == torch.float8_e4m3fn:
        assert qweight_scales is not None, "FP8 grouped matmul weight scales is None!"
        output = torch.empty(out_shape, device=x.device, dtype=torch.bfloat16)
        torch.ops.sgl_kernel.musa_fused_gemv(
            x, qweight, output, x_scales, qweight_scales,
            use_int4_w4a16, use_swigelu, use_rms_norm, gamma, eps)
        return output

    # --- 路径 2：w4a16 量化 ---
    elif qweight_scales is not None:
        assert x.dtype in (torch.bfloat16, torch.float16), \
            "W4A16 gemv only support bfloat16 or float16!"
        use_int4_w4a16 = True
        out_shape = x.shape[:-1] + (
            qweight.shape[0] if not use_swigelu else qweight.shape[0] // 2,
        )
        output = torch.empty(out_shape, device=x.device, dtype=x.dtype)
        torch.ops.sgl_kernel.musa_fused_gemv(
            x, qweight, output, None, qweight_scales,
            use_int4_w4a16, use_swigelu, use_rms_norm, gamma, eps)
        return output

    # --- 路径 3：通用 GEMV（fp16/bf16） ---
    else:
        output = torch.empty(out_shape, device=x.device, dtype=x.dtype)
```

```
        torch.ops.sgl_kernel.musa_fused_gemv(
            x, qweight, output, None, None,
            use_int4_w4a16, use_swigelu, use_rms_norm, gamma, eps)
        return output
```

sgl-kernel/include/sgl_kernel_musa_ops.h

声明所有 MUSA 算子的 C++ 接口，供 common_extension_musa.cc 注册使用，是架构级边界文件。

```
// sgl-kernel/include/sgl_kernel_musa_ops.h -- MUSA 算子 C++ 接口声明
#pragma once

#include <ATen/ATen.h>
#include <ATen/Tensor.h>
#include <torch/torch.h>
#include <optional>

// 批量化旋转位置编码 (batched)
void batched_rotary_embedding_contiguous(
    torch::Tensor& positions, torch::Tensor& query, torch::Tensor& key,
    int64_t head_size, torch::Tensor& cos_sin_cache, bool is_neox,
    int64_t rot_dim, torch::Tensor& cos_sin_cache_offsets);

// 旋转位置编码 ( 非 batch 版本 )
void rotary_embedding_contiguous(
    torch::Tensor& positions, torch::Tensor& query, torch::Tensor& key,
    int64_t head_size, torch::Tensor& cos_sin_cache, bool is_neox);

// 融合 MoE GEMV （支持 int4 和 swigelu）
void fused_moe_gemv(
    torch::Tensor& A, torch::Tensor& B, torch::Tensor& C,
    const c10::optional<torch::Tensor>& A_scale,
    const c10::optional<torch::Tensor>& B_scale,
    torch::Tensor& topk_weights, torch::Tensor& topk_ids,
    bool mul_routed_weight, int64_t topk, bool use_int4_w4a16, bool use_swigelu);

// MUSA 专用融合 GEMV（支持 int4/rms_norm）
void musa_fused_gemv(
    torch::Tensor& A, torch::Tensor& B, torch::Tensor& C,
    const c10::optional<torch::Tensor>& A_scale,
    const c10::optional<torch::Tensor>& B_scale,
    bool use_int4_w4a16, bool use_swigelu, bool use_rms_norm,
    const c10::optional<torch::Tensor>& gamma, double eps);

// 融合乘法加法（用于 element-wise 融合）
void fused_mul_add(torch::Tensor& output, torch::Tensor& self, torch::Tensor& bias, double
scale);

// top-k/top-p 采样（MUSA 版本）
```

```
void musa_top_k_top_p_sampling_from_probs(
    at::Tensor probs, at::Tensor output,
    std::optional<at::Tensor> maybe_indices,
    std::optional<at::Tensor> maybe_top_k_arr, double top_k_val,
    std::optional<at::Tensor> maybe_top_p_arr, double top_p_val,
    bool deterministic, std::optional<at::Generator> gen);
```

评论区精华

1. 内存泄漏（Critical）：gemini-code-assist[bot] 指出 moe_gemv_swiglu.mu 中使用 new 分配 best_config 但未释放，建议改用栈对象。作者确认已修改。
2. const_cast 危险性（High）：在 per_token_group_quant_8bit_v2.cu 中，为 MUSA 路径使用 const_cast 写入 const 指针。作者表示该部分不是本 PR 新增，保持原逻辑以避免副作用。
3. 头文件拆分请求：yeahdongcn 建议将 MUSA 操作声明从 sgl_kernel_ops.h 移到独立文件 sgl_kernel_musa_ops.h。作者照做。
4. torch.version.musa 检查：yeahdongcn 指出 torch.version.musa 属性可能不存在，应使用 hasattr 防御。作者已修复。
5. Inter-block barrier 不安全：alexnails 指出 moe_gemv_swiglu.mu 中的线程屏障仅限 block 内，跨 block 依赖会导致内存一致性问题。作者部分回应，改用 if constexpr 编译时判断。
6. GQA 性能问题：alexnails 在 pos_encoding_contiguous.mu 中指出 block size 忽略 num_kv_heads，对 GQA 不友好。未在提交中看到明确修改。

- 内存泄漏：new 分配未释放 (correctness): 作者回应改为栈分配替代 new，已修复。
- const_cast 危险操作 (security): 作者表示该代码非本次 PR 新增，保持原样以避免副作用。未修改。
- 头文件拆分：MUSA 声明独立文件 (design): 作者已拆分并创建新文件 sgl_kernel_musa_ops.h。
- torch.version.musa 安全访问 (correctness): 作者已按建议修改。
- Inter-block barrier 内存一致性问题 (correctness): 作者部分回应，将 BLOCK_N > ThreadNumPerWarp 改为 if constexpr 编译期判断，但未完全消除 barrier 使用。风险仍存在。

风险与影响

- 风险：
  1. 缺少测试覆盖：PR 无配套单元测试或集成测试，可能导致回归未被及时捕获。
  2. 并发安全风险：alexnails 指出的 inter-block barrier 问题虽部分回应，但未完全修复，可能在高负载下触发竞争。
  3. 条件编译复杂度：多个文件中散布 #ifdef USE_MUSA，增加了维护成本，可能遗漏分支覆盖。

4. API 兼容性：新增的 MUSA 前缀算子（如 musa_fused_gemv）与 CUDA 版本命名不统一，需调用方谨慎选择。 - 影响：用户侧：仅影响 MUSA 平台用户，使其能编译 sgl-kernel 并运行 SGLang 推理。CUDA 用户完全不受影响。系统侧：无性能 / 功能退化风险，新代码仅在 USE_MUSA 宏启用时生效。团队侧：为后续 MUSA 集成奠定基础，但需补充测试和优化（如 GQA 性能）。

- 风险标记：缺少测试覆盖，跨架构条件编译复杂度，并发 barrier 风险未完全解决

关联脉络

- PR #16565 [Roadmap][Feature] Support Moore Threads (MUSA) GPU: 本 PR 是该 Roadmap 的 [18/N]，目标为 MUSA 平台提供 sgl-kernel 级别内核支持。