PR #22774 完整报告

sgl-project/sglang

[MUSA][16/N] Add MUSA backend support for layers and DeepSeek models (V2/V3/R1)

合并时间：2026-04-24 09:59

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22774

执行摘要

- 一句话：新增 MUSA GPU 后端对 DeepSeek 模型的完整支持
- 推荐动作：此 PR 为多后端适配的典范，值得详细审查。重点关注 _is_musa() 守卫模式、forward_musa 方法的分发逻辑以及 DeepGEMM 的 MUSA 绕过策略。建议在合并后尽快补充 MUSA CI 测试，并关注 FP8 量化和 page_size 相关未解决讨论。

功能与动机

Enable SGLang to run DeepSeek models on Moore Threads MUSA GPUs. This PR adds MUSA backend support across the inference stack, including layers, quantization, MoE, attention, speculative decoding, and custom op registration.

实现拆解

实现拆解分为 5 个步骤：

1. 基础设施与自定义 OP 注册(utils/common.py, server_args.py) - 在 direct_register_custom_op 中添加 MUSA dispatch key 注册自定义算子 - 扩展 get_device_sm() 以通过 torch.cuda.get_device_capability() 检测 MUSA SM 版本（借助 torchada） - 在 server_args.py 中添加 musa 为合法设备（CLI 帮助字符串）
2. Layer 支持(activation.py, layernorm.py, quantization/fp8_kernel.py, quantization/fp8.py, quantization/unquant.py) - activation.py: 为 SiluAndMul 新增 forward_musa，非 piecewise CUDA 图时使用懒加载的 nn.SwishGLU（MUSA 上性能更好） - layernorm.py: 为 RMSNorm 新增 forward_musa，piecewise 图启用时用 forward_native 回退，否则使用 fused_add_rmsnorm 或 nn.functional.rms_norm - fp8_kernel.py: 导入 sgl_per_token_quant_fp8，MUSA 上强制走 v2 per-token group quant 路径 - fp8.py, unquant.py: 设置 MUSA 最小 FP8 能力为 SM 31；添加 forward_musa 委派到 CUDA 实现
3. MoE 优化(fused_moe_triton/fused_moe.py, moe_runner/triton.py, moe_runner/deep_gemm.py, ep_moe/kernels.py, topk.py) - 在 fused_moe 和 triton runner 中使用预实例化的 SiluAndMul/GeluAndMul（避免每次 forward 创建新对象） - 跳过 MUSA 上 intermediate_cache2 的预分配 - deep_gemm.py: 利用 DEEPGEMM_NEED_TMA_ALIGNED_SCALES 标志跳过 MUSA 上的 TMA 对齐 - ep_moe/kernels.py: Triton kernel 增加 ATOMIC_ADD_SEM 参数（MUSA 上设为 "

relaxed" 提升性能） - topk.py: 修正条件使用 _is_musa 变量，启用 biased_grouped_topk_gpu 路径

4. DeepSeek 模型适配(deepseek_v2.py, deepseek_weight_loader.py, forward_mla.py, forward_mha.py) - deepseek_v2.py: 导入 MUSA 版 dsv3_fused_a_gemm、dsv3_router_gemm；允许 MUSA 上使用共享专家融合（SM≥31）、双流执行和 routed_scaling_factor 缩放 - deepseek_weight_loader.py: 启用 FP8 block quant 权重加载，临时将 w_kc/w_vc 反量化至 bf16 - forward_mla.py: 对 MUSA 使用 torch.bmm 路径进行 w_vc 投影
5. 推测解码与编译工具(eagle_utils.py, eagle_worker.py, compile_utils.py, configurer.py) - 推测解码：MUSA 上导入 build_tree_kernel_efficient、verify_tree_greedy，禁用 torch.compile - compile_utils.py: 将 deep_gemm_execution_hook 改为非 contextmanager 的普通函数，内部判断 MUSA 时返回 nullcontext()，原逻辑移至 _deep_gemm_execution_hook - configurer.py: 允许 MUSA SM≥31 启用 DeepGEMM；新增 DEEPGEMM_NEED_TMA_ALIGNED_SCALES 标志

关键文件：

- python/sglang/srt/layers/layernorm.py（模块 层归一化；类别 source；类型 core-logic；符号 forward_musa）：核心层：新增 forward_musa 方法，实现 MUSA 上 fused_add_rmsnorm 和 native rms_norm 的适配。
- python/sglang/srt/layers/activation.py（模块 激活函数；类别 source；类型 core-logic；符号 forward_musa）：激活函数：新增 forward_musa 方法，使用 nn.SwishGLU 获得更好性能。
- python/sglang/srt/layers/deep_gemm_wrapper/compile_utils.py（模块 DeepGEMM 编译；类别 source；类型 core-logic；符号 _deep_gemm_execution_hook）：编译工具：重构 deep_gemm_execution_hook，MUSA 上跳过 JIT 编译。
- python/sglang/srt/models/deepseek_v2.py（模块 DeepSeek 模型；类别 source；类型 data-contract）：模型核心：导入 MUSA 专用 kernel，调整 routed_scaling_factor 和共享专家融合条件。
- python/sglang/srt/layers/moe/topk.py（模块 TopK 路由；类别 source；类型 core-logic）：MoE 路由：修复条件错误，启用 biased_grouped_topk_gpu。

关键符号：forward_musa, deep_gemm_execution_hook, _deep_gemm_execution_hook, biased_grouped_topk_gpu

关键源码片段

python/sglang/srt/layers/layernorm.py

核心层：新增 forward_musa 方法，实现 MUSA 上 fused_add_rmsnorm 和 native rms_norm 的适配。

```
# python/sglang/srt/layers/layernorm.py
# 新增 MUSA 专用前向方法，利用 fused_add_rmsnorm kernel 或 torch 原生 rms_norm

def forward_musa(
```

```
    self,
    x: torch.Tensor,
    residual: Optional[torch.Tensor] = None,
    post_residual_addition: Optional[torch.Tensor] = None,
) -> Union[torch.Tensor, Tuple[torch.Tensor, torch.Tensor]]:
    # piecewise CUDA 图模式暂不支持 MUSA 优化 kernel，回退到 native
    if not get_global_server_args().disable_piecewise_cuda_graph:
        return self.forward_native(x, residual, post_residual_addition)

    if not x.is_contiguous():
        x = x.contiguous()

    if residual is not None:
        # 融合 rmsnorm + 残差加法，in-place 操作减少显存占用
        if post_residual_addition is not None:
            residual = residual + post_residual_addition
        fused_add_rmsnorm(x, residual, self.weight.data, self.variance_epsilon)
        return x, residual

    # 无残差时使用 torch 原生 rms_norm（MUSA 兼容）
    out = nn.functional.rms_norm(
        x, (self.hidden_size,), self.weight.data, self.variance_epsilon
    )
    return out
```

python/sglang/srt/layers/activation.py

激活函数：新增 forward_musa 方法，使用 nn.SwishGLU 获得更好性能。

```
# python/sglang/srt/layers/activation.py
# 新增 MUSA 专用前向方法，使用 nn.SwishGLU（MUSA 上性能优于 silu_and_mul kernel）

def forward_musa(self, x: torch.Tensor) -> torch.Tensor:
    # piecewise CUDA 图模式暂不支持，使用 native 实现
    if not get_global_server_args().disable_piecewise_cuda_graph:
        return self.forward_native(x)
    # 懒加载 SwishGLU 模块，避免每次 forward 实例化
    if not hasattr(self, "_musa_swish_glu"):
        # XXX (MUSA): SwishGLU 在 MUSA 上比 silu_and_mul kernel 更快，未来可考虑实现专用
        kernel
        self._musa_swish_glu = nn.SwishGLU()
    return self._musa_swish_glu(x)
```

python/sglang/srt/layers/deep_gemm_wrapper/compile_utils.py

编译工具：重构 deep_gemm_execution_hook，MUSA 上跳过 JIT 编译。

```
# python/sglang/srt/layers/deep_gemm_wrapper/compile_utils.py
# 重构后：MUSA 不执行 DeepGEMM 编译，返回空上下文

def deep_gemm_execution_hook(
```

```
    m: int, n: int, k: int, num_groups: int, kernel_type: DeepGemmKernelType
):
    # MUSA 平台不需要 DeepGEMM JIT 编译，直接返回空上下文
    if _is_musa:
        return nullcontext()
    # 其他平台执行预编译
    return _deep_gemm_execution_hook(m, n, k, num_groups, kernel_type)


@contextmanager
def _deep_gemm_execution_hook(
    m: int, n: int, k: int, num_groups: int, kernel_type: DeepGemmKernelType
):
    if m > 0:
        _maybe_compile_deep_gemm_one_type_all(kernel_type, n, k, num_groups)
    yield
```

评论区精华

Review 中关键讨论点如下：

> TopK 条件错误（correctness，高优先级） gemini-code-assist[bot] 指出 topk.py 中使用了 is_musa（函数对象）而非布尔变量 _is_musa，导致条件永远为 True。已修正为 (_is_cuda or _is_musa)。
>
> 激活函数预实例化（performance） gemini-code-assist[bot] 建议在 activation.py 和 MoE 路径中懒加载 SiluAndMul/GeluAndMul，避免每次 forward 创建新模块。作者采纳，并将 forward_musa 中的 nn.SwishGLU() 改为懒加载。
>
> DeepGEMM 编译钩子重构（design） yeahdongcn 建议将 deep_gemm_execution_hook 拆分为普通函数和内部 contextmanager，以清晰处理 MUSA 无需编译的情况。最终实现为：deep_gemm_execution_hook 判断 MUSA 返回 nullcontext()，否则委派给 _deep_gemm_execution_hook。
>
> FA3 page_size 兼容性（design，未完全解决） yeahdongcn 对 server_args.py 中强制 MUSA 使用 page_size=64 的修改提出疑问，作者回应最新驱动已支持更多 page size，但该修改仍保留。讨论未达成最终结论，存在后续变更可能。
>
> FP8 量化细节（correctness） froststeam 指出在 fp8_kernel.py 中，MUSA 需要避免设置 column_major_scales 和 scale_tma_aligned，仅做 contiguous 不够。该点需进一步确认。

- topk.py 条件错误：使用 is_musa 函数对象而非 _is_musa (correctness): 已修正为 (_is_cuda or _is_musa)，使用布尔变量。
- activation.py 中使用 nn.SwishGLU 的懒加载优化 (performance): 作者采用懒加载方案，通过 hasattr 检查并缓存实例。
- DeepGEMM 编译钩子重构为普通函数 + 内部 contextmanager (design): 已实现：deep_gemm_execution_hook 判断 MUSA 返回 nullcontext，否则调用

_deep_gemm_execution_hook。

- server_args.py MUSA FA3 page_size 强制设为 64 (design): 修改保留，但存在后续变更可能。
- FP8 kernel 中 MUSA 需要避免 column_major_scales 设置 (correctness): 仍需进一步确认和修改，可能需要在后续 PR 中处理。

风险与影响

- 风险:
  - 回归风险：所有修改均被 _is_musa() 守卫，理论上不影响其他平台。但 gemini-code-assist 发现一处条件变量误用（is_musa 函数对象），说明该模式存在遗漏风险，需全面审查类似条件。
  - 性能风险：MUSA 上部分 kernel 使用 forward_native 回退（如 piecewise CUDA 图未启用时），可能显著慢于 CUDA 优化版本。PR 中已注明当前需 --disable-piecewise-cuda-graph 才能正确运行。
  - 兼容性风险：MUSA 上 FA3 后端强制 page_size=64，若未来驱动升级可能产生不适配，需持续关注。
  - 测试覆盖风险：本次未包含 MUSA CI 测试用例，后续回归难以自动检测。
- 影响:
  - 用户：MThreads GPU 用户可直接运行 DeepSeek 模型（需特定启动参数），其他用户无感知。
  - 系统：扩展了硬件兼容性，但增加了代码仓库复杂度（约 27 个文件修改）。公共抽象层如 MultiPlatformOp 保持稳定。
  - 团队：需要专人维护 MUSA 后端，并持续跟踪 upstream kernel 变更。
  - 风险标记：MUSA 新平台回归风险，条件守卫遗漏风险，缺少 MUSA 测试覆盖，性能回退风险

关联脉络

- PR #17946 [MUSA][8/N] Port CUDA kernels that are compatible with MUSA: 同一 MUSA 支持系列的前序 PR，提供了底层 kernel 移植，本 PR 在其基础上构建层和模型支持。
- PR #23493 Skip unselected experts in flashinfer_trtllm: MoE runner 修改与本 PR 的 MoE 适配有重叠文件（如 deep_gemm.py），需关注冲突。
- PR #23545 Fix MoE no_combine: skip router weight in down projection: MoE 相关修复，可能影响相同代码区域（fused_moe.py）。