PR #29667 完整报告

sgl-project/sglang

Add fused EH norm for DeepSeek NextN

合并时间：2026-07-01 17:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29667

执行摘要

- 一句话：为 DeepSeek NextN 添加融合 EH Norm JIT kernel，加速 10~20 倍。
- 推荐动作：值得精读，特别是 JIT kernel 的封装模式（@cache_once、load_jit）和 CUDA kernel 的融合策略。可作为后续相似融合 kernel 开发的参考。

功能与动机

在 DeepSeek NextN 推理中，EH norm 操作（对输入 embedding 和上一步 hidden state 分别做 RMSNorm 后拼接）是频繁调用的热点。原实现先调用 self.enorm() 和 self.hnorm() 再 cat，每个 Schritt 两次 norm kernel launch。融合后可以大幅减少 kernel 开销，提升吞吐。

实现拆解

1. 新增 CUDA kernel 文件 fused_eh_norm.cuh，实现融合 norm + cat 的 kernel，通过模板参数支持不同 hidden size 和数据类型，利用 PDL（persistent data loading）优化访存。
2. 新增 Python 封装 fused_eh_norm.py，提供 fused_eh_norm 函数，支持 fp16/bf16，hidden size 256~8192 且为 256 倍数。函数内部使用 @cache_once 缓存的 JIT module 调用 CUDA kernel。
3. 修改 deepseek_nextn.py，在 MTP 模块的 forward 方法中，当运行在 CUDA 设备时调用 fused_eh_norm 替换原来的 self.enorm + self.hnorm + cat 逻辑；非 CUDA 设备保留旧路径以确保兼容性。
4. 新增单元测试 test_fused_eh_norm.py，包含与参考实现的一致性测试、零 token 测试、行 stride 输入测试、不支持的 dtype 和 hidden size 拒绝测试。
5. 新增基准测试 bench_fused_eh_norm.py，在不同参数下对比 JIT kernel 与 PyTorch 参考实现的性能。测试注册了 base-b 和 nightly 两个 suite（review 中有人建议只 nightly，但作者决定保留两种）。

关键文件：

- python/sglang/jit_kernel/fused_eh_norm.py（模块 融合 kernel；类别 source；类型 dependency-wiring；符号 is_supported_fused_eh_norm_hidden_size, _jit_fused_eh_norm_module, fused_eh_norm）：新融合 kernel 的 Python 封装，提供调用接口和错误检查。
- test/registered/jit/test_fused_eh_norm.py（模块 单元测试；类别 test；类型 test-coverage；符号 _reference, test_fused_eh_norm_matches_reference,

test_fused_eh_norm_zero_tokens, test_fused_eh_norm_row_strided_inputs）：覆盖核心正确性、边界条件和错误路径。

- python/sglang/srt/models/deepseek_nextn.py（模块 模型集成；类别 source；类型 data-contract）：将融合 kernel 集成到 DeepSeek NextN 推理中，是性能收益落地的关键。
- test/registered/jit/benchmark/bench_fused_eh_norm.py（模块 基准测试；类别 test；类型 test-coverage；符号 reference, benchmark）：性能基准测试，提供不同参数下的 JIT vs 参考实现对比。
- python/sglang/jit_kernel/csrc/elementwise/fused_eh_norm.cuh（模块 CUDA 内核；类别 source；类型 dependency-wiring）：CUDA kernel 实现，执行实际的融合 norm + cat 计算。

关键符号：fused_eh_norm, is_supported_fused_eh_norm_hidden_size, _jit_fused_eh_norm_module

关键源码片段

python/sglang/jit_kernel/fused_eh_norm.py

新融合 kernel 的 Python 封装，提供调用接口和错误检查。

```
from __future__ import annotations

from typing import TYPE_CHECKING

import torch

from sglang.jit_kernel.utils import (
    cache_once,
    is_arch_support_pdl,
    load_jit,
    make_cpp_args,
)

if TYPE_CHECKING:
    from tvm_ffi.module import Module


def is_supported_fused_eh_norm_hidden_size(hidden_size: int) -> bool:
    # hidden_size 必须在 (256, 8192] 范围内且是 256 的倍数
    return hidden_size > 256 and hidden_size <= 8192 and hidden_size % 256 == 0


@cache_once
def _jit_fused_eh_norm_module(hidden_size: int, dtype: torch.dtype) -> Module:
    # 构造 JIT 编译参数，包括 hidden_size、是否支持 PDL 和数据类型
    args = make_cpp_args(hidden_size, is_arch_support_pdl(), dtype)
    return load_jit(
        "fused_eh_norm",
```

```
        *args,
        cuda_files=["elementwise/fused_eh_norm.cuh"],
        # 将 C++ 模板类 FusedEHNormKernel 的静态方法 run 暴露为 fused_eh_norm
        cuda_wrappers=[("fused_eh_norm", f"FusedEHNormKernel<{args}>::run")],
    )


def fused_eh_norm(
    inputs_embeds: torch.Tensor,
    previous_hidden: torch.Tensor,
    enorm_weight: torch.Tensor,
    hnorm_weight: torch.Tensor,
    eps: float,
) -> torch.Tensor:
    """Return fused EH norm + cat for contiguous CUDA fp16/bf16 tensors."""
    # 仅支持 fp16 或 bf16
    if inputs_embeds.dtype not in (torch.float16, torch.bfloat16):
        raise RuntimeError(
            f"fused_eh_norm: unsupported dtype {inputs_embeds.dtype}; "
            "expected torch.float16 or torch.bfloat16"
        )
    # 仅支持 2D 输入
    if inputs_embeds.dim() != 2:
        raise RuntimeError(
            f"fused_eh_norm: inputs_embeds must be 2D, got {inputs_embeds.dim()}D"
        )
    hidden_size = inputs_embeds.shape[1]
    # 检查 hidden_size 是否在支持范围内
    if not is_supported_fused_eh_norm_hidden_size(hidden_size):
        raise RuntimeError(
            f"fused_eh_norm: unsupported hidden_size={hidden_size} "
            "(must be in (256, 8192] and a multiple of 256)"
        )
    # 预分配输出，形状为 (batch, hidden_size * 2)
    output = torch.empty(
        (inputs_embeds.shape[0], hidden_size * 2),
        dtype=inputs_embeds.dtype,
        device=inputs_embeds.device,
    )
    # 零 token 时直接返回空输出，避免 kernel 调用
    if inputs_embeds.shape[0] == 0:
        return output
    # 获取或编译 JIT module
    module = _jit_fused_eh_norm_module(hidden_size, inputs_embeds.dtype)
    # 调用 CUDA kernel
    module.fused_eh_norm(
        inputs_embeds, previous_hidden, enorm_weight, hnorm_weight, output, eps
    )
    return output
```

python/sglang/srt/models/deepseek_nextn.py

将融合 kernel 集成到 DeepSeek NextN 推理中，是性能收益落地的关键。

```
# 在文件头部新增导入
from sglang.jit_kernel.fused_eh_norm import fused_eh_norm

# 在 forward 方法中，替换原来的 norm + cat 逻辑
if hidden_states.shape[0] > 0:
    previous_hidden_states = forward_batch.spec_info.hidden_states
    if self.rot_weight is not None:
        previous_hidden_states = torch.matmul(
            previous_hidden_states, self.rot_weight
        )
    if _is_cuda:
        # 使用融合 kernel 一次性完成 norm + cat
        eh_input = fused_eh_norm(
            hidden_states,
            previous_hidden_states,
            self.enorm.weight,
            self.hnorm.weight,
            self.enorm.variance_epsilon,
        )
    else:
        # 非 CUDA 设备走旧路径
        eh_input = torch.cat(
            (
                self.enorm(hidden_states),
                self.hnorm(previous_hidden_states),
            ),
            dim=-1,
        )
    # 后续处理不变
    if isinstance(self.eh_proj, ReplicatedLinear):
        hidden_states, _ = self.eh_proj(eh_input)
    else:
        hidden_states = self.eh_proj(eh_input)
```

python/sglang/jit_kernel/csrc/elementwise/fused_eh_norm.cuh

CUDA kernel 实现，执行实际的融合 norm + cat 计算。

```
#include <sgl_kernel/tensor.h>
#include <sgl_kernel/utils.h>
#include <sgl_kernel/runtime.cuh>
#include <sgl_kernel/tile.cuh>
#include <sgl_kernel/utils.cuh>
#include <sgl_kernel/vec.cuh>
#include <sgl_kernel/impl/norm.cuh>
#include <tvm/ffi/container/tensor.h>
```

```
// 参数结构体，用于 kernel 启动
struct FusedEHNormParams {
    const void* __restrict__ embeds;
    const void* __restrict__ previous_hidden;
    const void* __restrict__ enorm_weight;
    const void* __restrict__ hnorm_weight;
    void* __restrict__ output;
    int64_t embeds_stride;
    int64_t previous_hidden_stride;
    int64_t output_stride;
    float eps;
};

template <int64_t kHidden, bool kUsePDL, typename T>
__global__ void fused_eh_norm_kernel(const __grid_constant__ FusedEHNormParams params) {
    using namespace device;
    using Storage = norm::StorageType<T, kHidden>;

    constexpr auto kNumThreads = host::norm::get_cta_threads<T, kHidden>();
    constexpr auto kNumWarps = kNumThreads / kWarpThreads;

    // 计算指针偏移
    const auto embeds = static_cast<const T*>(pointer::offset<T>(params.embeds, blockIdx.x *
    params.embeds_stride));
    const auto previous_hidden = static_cast<const T*>(pointer::offset<T>(params.previous_
    hidden, blockIdx.x * params.previous_hidden_stride));
    const auto enorm_weight = static_cast<const T*>(params.enorm_weight);
    const auto hnorm_weight = static_cast<const T*>(params.hnorm_weight);
    const auto output = static_cast<T*>(pointer::offset<T>(params.output, blockIdx.x * params.
    output_stride));

    const auto gmem = tile::Memory<Storage>::cta(kNumThreads);
    __shared__ float smem[norm::kSmemBufferSize];

    PDLWaitPrimary<kUsePDL>();

    // 对 embedding 部分做 norm 并写入输出前半部分
    const auto embeds_vec = gmem.load(embeds);
    const auto enorm_weight_vec = gmem.load(enorm_weight);
    const auto embeds_output_vec = norm::apply_norm_cta<kHidden>(embeds_vec, enorm_
    weight_vec, params.eps, smem, kNumWarps);
    gmem.store(output, embeds_output_vec);

    // 对 previous hidden 部分做 norm 并写入输出后半部分
    const auto prev_vec = gmem.load(previous_hidden);
    const auto hnorm_weight_vec = gmem.load(hnorm_weight);
    const auto prev_output_vec = norm::apply_norm_cta<kHidden>(prev_vec, hnorm_weight_vec,
    params.eps, smem, kNumWarps);
    gmem.store(output + kHidden, prev_output_vec);
```

```
    PDLTriggerSecondary<kUsePDL>();
}

// 模板结构体，封装 kernel 启动
template <int64_t kHidden, bool kUsePDL, typename T>
struct FusedEHNormKernel {
    static_assert(host::norm::is_config_supported<T, kHidden>(), "Unsupported norm
    configuration");
    static_assert(host::norm::should_use_cta<T, kHidden>(), "fused_eh_norm requires CTA norm")
    ;
    static constexpr auto kernel = fused_eh_norm_kernel<kHidden, kUsePDL, T>;
    static constexpr uint32_t kBlockSize = host::norm::get_cta_threads<T, kHidden>();

    static void run(const tvm::ffi::TensorView embeds,
                    const tvm::ffi::TensorView previous_hidden,
                    const tvm::ffi::TensorView enorm_weight,
                    const tvm::ffi::TensorView hnorm_weight,
                    const tvm::ffi::TensorView output,
                    float eps) {
        FusedEHNormParams params;
        params.embeds = embeds.data;
        params.previous_hidden = previous_hidden.data;
        params.enorm_weight = enorm_weight.data;
        params.hnorm_weight = hnorm_weight.data;
        params.output = output.data;
        params.embeds_stride = embeds.shape[1];
        params.previous_hidden_stride = previous_hidden.shape[1];
        params.output_stride = output.shape[1];
        params.eps = eps;

        dim3 grid(embeds.shape[0]);
        dim3 block(kBlockSize);
        kernel<<<grid, block>>>(params);
    }
};
```

评论区精华

Review 中 Fridge003 建议将单元测试仅注册为 nightly 测试（est_time=120），但作者 mmangkad 认为 base-b 测试（est_time=45）可在常规 CI 中提供快速反馈，最终决定保留两个注册。未涉及其他争议。

- 测试注册策略 (testing): 作者决定保留两个注册，reviewer 未进一步反对。

风险与影响

- 风险：
  1. 新的 CUDA kernel 依赖 PDL 特性，在不支持 PDL 的旧架构上可能无法编译或运行失败。代码通过 is_arch_support_pdl() 动态判断，但若失败无 fallback 到旧路径（仅非

CUDA 设备有 fallback）。

2. hidden_size 限制为 (256, 8192] 且为 256 倍数，可能阻止某些未来模型使用，但当前 DeepSeek NextN 的 hidden_size（6144/7168）符合要求。
3. 测试只覆盖了 6144 和 7168 两种 hidden_size，其他合法值未测试。 - 影响：对 CUDA 用户：DeepSeek NextN 推理性能显著提升（bench 显示 10~20x）。对非 CUDA 用户（AMD、NPU）：无影响，仍使用旧实现。对 API 无影响，函数签名不变。对系统其他部分无影响，因修改仅限 jit_kernel 和 deepseek_nextn 模块。 - 风险标记：CUDA kernel 无 fallback, hidden_size 限制

关联脉络

- 暂无明显关联 PR