PR #25860 完整报告

sgl-project/sglang

add git gemm warpper for dispatch_bf16_fp32_backend

合并时间：2026-05-21 06:24

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25860

执行摘要

- 一句话：引入 DeepGemm wrapper 优化 BF16→FP32 GEMM 分发
- 推荐动作：建议合并。变更小、影响明确、review 后已修正。作者提供了性能数据且 CI E2E 测试通过。值得关注的是 deep_gemm_wrapper 的实现细节（位于 sglang/srt/layers/），未来可复用此模式统一其他 GEMM 分发。

功能与动机

作者在 PR body 中说明：'We added a JIT DeepGemm code branch for _dispatch_bf16_fp32_backend to reduce cache generation during the runtime stage.' 通过将 deep_gemm 调用包装进 deep_gemm_wrapper，可以复用已有的 JIT 缓存，减少冷启动开销。

实现拆解

1. 添加模块级常量：在 deepseek_v4.py 中新增 from sglang.srt.layers import deep_gemm_wrapper 导入，并将环境变量 SGLANG_OPT_BF16_FP32_GEMM_ALGO 的读取提升为模块级常量 _linear_bf16_fp32_algo，避免每次调用 linear_bf16_fp32 时重复 env lookup。
2. 简化 linear_bf16_fp32 调用：函数体直接传递模块级常量，不再内部 import envs。
3. 替换 deep_gemm dispatch：在 _dispatch_bf16_fp32_backend 的 algo == "deep_gemm" 分支中，用 deep_gemm_wrapper.gemm_nt_bf16bf16f32(x, y, z) 替代直接 import deep_gemm 并调用 deep_gemm.bf16_gemm_nt，同时将 tensor 创建从 x.new_empty 改为 torch.empty(..., device=x.device)。
4. 去除冗余 if-else：根据 reviewer Fridge003 的建议，移除了初始版本中针对 _ENABLE_JIT_DEEPGEMM 的条件分支，直接统一使用 wrapper。
5. 仅修改单一文件：python/sglang/jit_kernel/deepseek_v4.py，+7/-8，没有测试或配置配套变更。

关键文件：

- python/sglang/jit_kernel/deepseek_v4.py（模块 JIT 内核；类别 source；类型 dependency-wiring；符号 linear_bf16_fp32, _dispatch_bf16_fp32_backend, _linear_bf16_fp32_algo）：唯一变更文件，引入 deep_gemm_wrapper 替代直接 deep_gemm 调用，并提升算法选择为模块级常量。

关键符号：linear_bf16_fp32, _dispatch_bf16_fp32_backend

关键源码片段

python/sglang/jit_kernel/deepseek_v4.py

唯一变更文件，引入 deep_gemm_wrapper 替代直接 deep_gemm 调用，并提升算法选择为模块级常量。

```
# python/sglang/jit_kernel/deepseek_v4.py
# 引入 deep_gemm_wrapper 和模块级算法常量
from sglang.srt.layers import deep_gemm_wrapper

_linear_bf16_fp32_algo = envs.SGLANG_OPT_BF16_FP32_GEMM_ALGO.get()

def linear_bf16_fp32(x: torch.Tensor, y: torch.Tensor) -> torch.Tensor:
    # 直接使用模块级常量，避免每次调用时重新读取环境变量
    return _dispatch_bf16_fp32_backend(x, y, algo=_linear_bf16_fp32_algo)

def _dispatch_bf16_fp32_backend(
    x: torch.Tensor, y: torch.Tensor, *, algo: str
) -> torch.Tensor:
    if algo == "cublas":
        module = _jit_torch_cublas_bf16_fp32()
        return module.linear_bf16_fp32(x, y)
    elif algo == "deep_gemm":
        # 使用 wrapper 替代直接 import deep_gemm，减少运行时缓存生成
        z = torch.empty(x.size(0), y.size(0), dtype=torch.float32, device=x.device)
        deep_gemm_wrapper.gemm_nt_bf16bf16f32(x, y, z)
        return z
    elif _use_aiter:
        return tgemm.mm(x, y, otype=torch.float32)
    else:
        return torch.nn.functional.linear(x.float(), y.float())
```

评论区精华

关键讨论线程：Fridge003 在 review 中提出 deep_gemm dispatch 分支无需保留 if-else 逻辑，建议直接替换为 deep_gemm_wrapper.gemm_nt_bf16bf16f32。作者 BJWang-ant 接受该建议并更新代码。另一条来自 gemini-code-assist[bot] 的评论也提议将 tensor 分配统一为 x.new_empty，但最终实现采用了 torch.empty，保持了与现有代码风格一致。

- 去除 deep_gemm 分支的条件判断 (design): 作者接受建议，移除了 _ENABLE_JIT_DEEPGEMM 条件分支。
- 统一 tensor 分配方式 (style): 最终采用 torch.empty 并显式指定 device，与现有代码风格对齐。

风险与影响

- 风险：
  1. 回归风险：deep_gemm_wrapper 的接口需要与原有 deep_gemm.bf16_gemm_nt 行为完全一致，如果 wrapper 内部有逻辑差异可能导致数值错误。不过已有 E2E 测试 test_deepseek_v4_flash_fp4_*.py 通过，降低了风险。
  2. 模块级常量：_linear_bf16_fp32_algo 在导入时即被求值，如果在运行时通过环境变量动态切换算法，此变更会导致无法生效。但实际使用中算法通常固定，此变更符合预期使用场景。
  3. 缺少单元测试：PR 未添加针对 _dispatch_bf16_fp32_backend 各分支的单元测试，若未来修改 wrapper 行为可能缺少保护。 - 影响：影响范围：仅影响 linear_bf16_fp32 函数调用路径，该函数被 DeepSeek-V4 模型推理使用。性能提升约 2%-3%（作者在 ISL 4k/OSL 1k 和 32k/2k 场景下测得）。用户：无 API 变更，透明优化。系统：减少运行时 JIT 编译缓存生成，冷启动可能稍有改善。团队：后续若替换 GEMM 后端，只需修改 wrapper 实现。 - 风险标记：缺少单元测试，模块级常量可能影响动态配置

关联脉络

- PR #25460 [perf] prepare_prefill_qkv hook + fp8 quantize jit kernel: 同样涉及 deepseek 模型和 JIT kernel 性能优化，引入了 deep_gemm_wrapper 的前身或相关优化思路。