PR #31202 完整报告

sgl-project/sglang

Delete sgl-kernel AOT `bmm_fp8`, use `flashinfer.bmm_fp8`

合并时间：2026-07-22 07:44

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31202

执行摘要

- 一句话：删除 sgl-kernel AOT bmm_fp8，统一使用 flashinfer 实现
- 推荐动作：值得一读，展示了如何安全地移除重复 AOT 实现并统一依赖，同时利用 register_custom_op 保持 torch.compile 兼容性。

功能与动机

sgl-kernel 中的 bmm_fp8.cu 几乎完全复制自 flashinfer 的实现（commit 消息说明），flashinfer 已是硬依赖且支持相同的计算能力（SM89+），故无理由保留独立副本。删除后减少内核维护负担，同时保证功能完全一致。

实现拆解

1. 删除 sgl-kernel 的 bmm_fp8 AOT 实现：移除 bmm_fp8.cu、sgl_kernel_ops.h 中声明、common_extension.cc/common_extension_musa.cc 中注册，以及 gemm.py 中的 _bmm_fp8_internal 和 bmm_fp8 函数。
2. 在 fp8_utils.py 中新增基于 flashinfer.bmm_fp8 的封装：通过 register_custom_op 注册 _bmm_fp8_batched_op 确保 torch.compile 安全，并暴露 bmm_fp8 函数。
3. 将所有调用点统一到新入口：修改 minicpm3.py、forward_mla.py、sarvam_moe.py、forward_mla_fused_rope_rocm.py，从 sglang.kernels.ops.gemm 导入 bmm_fp8。
4. 在 sglang/kernels/ops/gemm/__init__.py 中注册 bmm_fp8 作为 KernelBackend.FLASHINFER 条目。
5. 删除 sgl-kernel/tests/test_bmm_fp8.py（flashinfer 已有覆盖）。

关键文件：

- sgl-kernel/python/sgl_kernel/gemm.py（模块 内核层；类别 source；类型 core-logic；符号 _bmm_fp8_internal, bmm_fp8）：删除了 sgl-kernel 的 bmm_fp8 实现，包括 _bmm_fp8_internal 和 bmm_fp8 函数，以及相关导入，是核心删除文件。
- python/sglang/srt/layers/quantization/fp8_utils.py（模块 量化层；类别 source；类型 core-logic；符号 _bmm_fp8_batched_op, bmm_fp8）：新增了基于 flashinfer.bmm_fp8 的封装，包括 custom_op 注册和 bmm_fp8 函数，是统一入口的核心文件。
- python/sglang/srt/models/minicpm3.py（模块 模型层；类别 source；类型 data-contract；符号 _bmm_fp8_op, bmm_fp8）：作为调用点之一，从复杂包装简化为统一导入，展示了

删除后的调用方式。

- sgl-kernel/tests/test_bmm_fp8.py（模块 内核测试；类别 test；类型 deletion；符号 to_float8, test_bmm_fp8）：删除的测试文件，原为测试 sgl-kernel 的 bmm_fp8，现由 flashinfer 覆盖。
- sgl-kernel/csrc/gemm/bmm_fp8.cu（模块 CUDA 内核；类别 other；类型 deletion）：删除的 CUDA kernel 源文件，是 AOT 实现的核心。
- python/sglang/kernels/ops/gemm/__init__.py（模块 Kernel 注册；类别 infra；类型 infrastructure；符号 bmm_fp8）：注册 bmm_fp8 作为 KernelBackend.FLASHINFER 条目，定义了 kernel 分发入口。

关键符号：_bmm_fp8_internal, bmm_fp8, _bmm_fp8_batched_op, _bmm_fp8_op

关键源码片段

python/sglang/srt/layers/quantization/fp8_utils.py

新增了基于 flashinfer.bmm_fp8 的封装，包括 custom_op 注册和 bmm_fp8 函数，是统一入口的核心文件。

```
# fp8_utils.py 新增部分：基于 flashinfer 的 bmm_fp8 封装
# 确保 torch.compile 不会跟踪 cuBLAS handle

from flashinfer import bmm_fp8 as _raw_bmm_fp8_batched

@register_custom_op(op_name="flashinfer_bmm_fp8_batched", mutates_args=["out"])
def _bmm_fp8_batched_op(
    A: torch.Tensor,
    B: torch.Tensor,
    out: torch.Tensor,
    A_scale: torch.Tensor,
    B_scale: torch.Tensor,
) -> None:
    """封装 flashinfer.bmm_fp8，通过 custom_op 避免 torch.compile 报错。"""
    _raw_bmm_fp8_batched(A, B, A_scale, B_scale, out.dtype, out)

def bmm_fp8(
    A: torch.Tensor,
    B: torch.Tensor,
    A_scale: torch.Tensor,
    B_scale: torch.Tensor,
    dtype: torch.dtype,
    out: Optional[torch.Tensor] = None,
) -> torch.Tensor:
    """Batched (3D) per-tensor-scale FP8 matmul，via flashinfer's cuBLAS backend."""
    if out is None:
        out = torch.empty(
            (A.shape[0], A.shape[1], B.shape[2]),
            device=A.device,
            dtype=dtype,
```

```
        )
    _bmm_fp8_batched_op(A, B, out, A_scale, B_scale)
    return out
```

评论区精华

此 PR 无 review 评论，直接由 BBuf 批准合并。但 commit 修复了 capability -> capabilities 拼写（关联 #31292），说明 kernel 注册接口有演进。

- 暂无高价值评论线程

风险与影响

- 风险：核心风险在于 flashinfer 版本的 bmm_fp8 行为是否完全一致。作者已验证 bit-identical 输出和匹配延迟。对于非 CUDA 平台（AMD/MThreads），该功能原本就在 if _is_cuda: 条件下，因此无影响。若 flashinfer 未来变更接口，依赖同步更新即可。删除的测试文件减轻了维护负担。
- 影响：对最终用户透明，功能无变化。对开发者：减少约 240 行代码，消除重复 kernel 维护负担，统一依赖至 flashinfer。对系统：无性能影响。
- 风险标记：依赖外部库（flashinfer），已验证 bit-identical

关联脉络

- PR #31292 fix: use capabilities= (plural) for gemm.bmm_fp8 KernelSpec: 本 PR 的提交中修复了 capability 拼写，与 #31292 的 KernelSpec 变更联动。
- PR #31961 Change the FP8 per-tensor GEMM backend on SM120 to cuBLAS: 同为 FP8 相关重构，但影响不同后端。