

PR #39371 完整报告

sgl-project/sglang

bumping sgl-deep-gemm to 0.2.0

合并时间: 2026-09-15 06:30

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39371>

执行摘要

- 一句话: 升级 sgl-deep-gemm 0.2.0 并恢复 BF16 批量确定性
- 推荐动作: 值得精读。重点学习三处设计: 一是用 `getattr` 探测新 API 实现新旧 wheel 兼容的写法; 二是 `try/finally` 恢复全局模式、避免状态泄漏的惯用法; 三是以 `batch = 1` 为基准、固定随机种子、零容差的精确相等回归测试设计, 这对任何「确定性承诺」类代码都很有参考价值。

功能与动机

PR body 明确指出: *DeepGEMM 0.2 defaults BF16 GEMM to cuBLASLt, whose reduction can change with batch size; the batch-invariant override must explicitly select the deterministic kernel*。SGLang 的 batch-invariant 模式承诺同一请求在不同 batch 大小下输出完全一致, 这是 DP、benchmark、speculative decoding 等场景正确性的基础; 裸升级 0.2.0rc0 会导致 H200 FA3、2xH200 DP (KL $5.23e-4$ / $1.089e-3$)、4xB200 Inkling NVFP4 + DSPARK (KL $9.94e-4$) 全部失败, 因此必须在 batch-invariant 调用路径中显式选择确定性内核。

实现拆解

1. 依赖升级: `python/pyproject.toml` 将 `sgl-deep-gemm==0.1.7` 改为 `sgl-deep-gemm==0.2.0`, 精确锁定新版本, 作为本次变更的触发点。
2. 核心修复: `python/sglang/srt/batch_invariant_ops/batch_invariant_ops.py` 的 `_matmul_persistent_deepgemm` 在调用 `deep_gemm.bf16_gemm_nn` 前, 先用 `getattr` 探测 `deep_gemm` 是否暴露 `get_deterministic_algorithms / use_deterministic_algorithms` API; 若存在且当前非确定性模式, 则切换为确定性内核, 并在 `try/finally` 中无论成功或失败都恢复原模式。旧版本 wheel 没有该 API 时按 `deterministic=True` 处理, 保持向后兼容。
3. 回归测试: `test/registered/unit/batch_invariant_ops/test_batch_invariant_ops.py` 新增 `test_deepgemm_random_bf16_batch_invariance`。固定随机种子 42, 构造 $257 \times 4096 \times 4096$ 的 BF16 矩阵, 以 `batch = 1` 的结果为基准, 对比 `batch = 16 / 64 / 257` 时前 1 行输出, 要求 `rtol = 0`、`atol = 0` 精确相等; 同时验证确定性模式状态在调用前后保持不变。无模式 API 的旧 wheel 只走 single-mode 兼容路径。测试在设备未启用 JIT DeepGEMM 时跳过。

4. 配套验证：PR body 列出三组受控 GPU 对比——H200 FA3 prefix tokens/logprobs、2xH200 DP breakable prefill/decode KL、4xB200 Inkling NVFP4 + DSPARK input/output KL。裸升级 0.2.0rc0 全部失败，加 fix 后与 0.1.7 完全一致（145 个响应完全相同、KL 均为 0/0）；速度上 eager 延迟基本持平（14.51–14.89 μ s vs 14.39–14.77 μ s），CUDA graph replay 甚至更快（10.25–12.29 μ s vs 14.33–14.35 μ s）。

关键文件：

- python/sglang/srt/batch_invariant_ops/batch_invariant_ops.py（模块 批量恒定；类别 infra；类型 core-logic；符号 _matmul_persistent_deepgemm）：核心修复所在：_matmul_persistent_deepgemm 在 DeepGEMM 0.2 下显式强制确定性 BF16 内核，并在 finally 恢复原模式，是恢复 batch-invariant 语义的关键。
- test/registered/unit/batch_invariant_ops/test_batch_invariant_ops.py（模块 批量恒定；类别 test；类型 test-coverage；符号 test_deepgemm_random_bf16_batch_invariance）：新增 test_deepgemm_random_bf16_batch_invariance 回归测试，用固定种子的随机 BF16 矩阵跨 batch 精确相等断言，直接验证本次修复的核心承诺。
- python/pyproject.toml（模块 依赖配置；类别 config；类型 configuration）：依赖升级的入口，sgl-deep-gemm==0.1.7 改为 ==0.2.0，是整个 PR 的触发点。

关键符号：_matmul_persistent_deepgemm, test_deepgemm_random_bf16_batch_invariance

关键源码片段

[python/sglang/srt/batch_invariant_ops/batch_invariant_ops.py](#)

核心修复所在：_matmul_persistent_deepgemm 在 DeepGEMM 0.2 下显式强制确定性 BF16 内核，并在 finally 恢复原模式，是恢复 batch-invariant 语义的关键。

```
# 位于 _matmul_persistent_deepgemm 内部，前面已完成 shape 校验与 out 张量分配
out = torch.empty((M, N), device=a.device, dtype=dtype)

# DeepGEMM 0.2 默认将 BF16 GEMM 路由到 cuBLASLt，其归约可能随 batch size 变化；
# 旧版本 wheel 始终使用 batch-invariant 内核，因此这里需要显式切回确定性模式。
get_deterministic = getattr(deep_gemm, "get_deterministic_algorithms", None)
deterministic = get_deterministic() if get_deterministic else True
if not deterministic:
    deep_gemm.use_deterministic_algorithms(True)
try:
    deep_gemm.bf16_gemm_nn(a, b, out)
except RuntimeError as e:
    raise RuntimeError(
        f"Consider increasing MIN_DEEPGEMM_DIM in matmul_persistent() or disabling DeepGEMM "
        f"for small matrices. Original error: {e}"
    ) from e
finally:
    # 成功与失败都必须恢复原模式，避免确定性开关泄漏影响其他调用方
    if not deterministic:
```

```
deep_gemm.use_deterministic_algorithms(False)
```

test/registered/unit/batch_invariant_ops/test_batch_invariant_ops.py

新增 test_deepgemm_random_bf16_batch_invariance 回归测试，用固定种子的随机 BF16 矩阵跨 batch 精确相等断言，直接验证本次修复的核心承诺。

```
def test_deepgemm_random_bf16_batch_invariance(self):
    # 当前设备未启用 JIT DeepGEMM 时跳过（例如 ROCm 或 CPU 环境）
    if not batch_invariant_ops.ENABLE_JIT_DEEPGEMM:
        self.skipTest("DeepGEMM is unavailable on this device")

    deep_gemm = batch_invariant_ops.deep_gemm
    # 旧版 wheel 没有确定性模式 API，用 getattr 探测以保持兼容
    get_deterministic = getattr(deep_gemm, "get_deterministic_algorithms", None)
    original_mode = get_deterministic() if get_deterministic else None
    modes = (False, True) if get_deterministic else (None,)
    generator = torch.Generator(device=device_type).manual_seed(42)
    # 固定种子的随机 BF16 数据，batch 大小包含非 2 次幂的 257
    a = torch.randn(257, 4096, dtype=torch.bfloat16, generator=generator)
    b = torch.randn(4096, 4096, dtype=torch.bfloat16, generator=generator).T
    try:
        for mode in modes:
            if mode is not None:
                deep_gemm.use_deterministic_algorithms(mode)
            with self.subTest(deterministic=mode):
                # 以 batch = 1 的结果为基准
                ref = batch_invariant_ops._matmul_persistent_deepgemm(a[:1], b)
                for batch_size in (16, 64, 257):
                    out = batch_invariant_ops._matmul_persistent_deepgemm(
                        a[:batch_size], b
                    )
                    # 跨 batch 输出必须精确一致，不允许任何误差
                    torch.testing.assert_close(out[:1], ref, rtol=0, atol=0)
                # 同时验证调用没有改变外部确定性模式状态
                if get_deterministic:
                    self.assertEqual(get_deterministic(), mode)
    finally:
        if original_mode is not None:
            deep_gemm.use_deterministic_algorithms(original_mode)
```

评论区精华

该 PR 没有收到任何 review 评论，评论区唯一活动是作者本人发布的两次 CI 测试运行链接（Run #34788188380、#34788188290）。值得记录的权衡来自 PR body：作者在速度测试中明确给出 eager 与 CUDA graph 两种口径的数据，并谨慎注明 **These four shapes are not a full serving benchmark**，说明确定性内核切换的性能代价很小，但覆盖形状有限。整体上这是一个「依赖升级 + 行为回归修复 + 测试补强」三合一的 PR，讨论密度低，证据质量高。

- CI 测试运行验证 (question): 作者提供了自测证据；PR 最终已合并。

风险与影响

- 风险:

1. 全局模式切换与并发: `use_deterministic_algorithms` 是 DeepGEMM 的全局状态, `_matmul_persistent_deepgemm` 在切换与恢复之间存在窗口, 若多个线程或 CUDA stream 并发调用且各自探测到不同模式, 可能出现互相覆盖的竞态。当前实现没有加锁或 thread-local 保护。
2. 兼容性假设: 旧 wheel 无 API 时按 `deterministic=True` 处理, 依赖「旧版本始终使用 invariant kernel」这一假设; 若未来旧版本有 hotfix 改变默认行为, 该分支的语义会失真。
3. 性能影响: `eager` 路径约慢 0.5%–1%, 但 graph replay 测量波动较大 (10.25–12.29 μ s), 实测形状有限, 不能代表全量 serving 负载。
4. 测试覆盖依赖硬件: 新增回归测试依赖 `ENABLE_JIT_DEEPGEMM`, 在 ROCm、CPU 等设备上会 skip, 可能造成覆盖盲区; CI 的 AMD ROCm 10 测试显示为失败状态。
5. 依赖精确锁定: `==0.2.0` 精确锁定, 后续 DeepGEMM 0.2.x 的修复需要再次发起 bump PR。
 - 影响: 影响范围集中在使用 DeepGEMM BF16 persistent GEMM 的服务端路径, 尤其是依赖 batch-invariant 语义的 DP 并行、benchmark 对比、speculative decoding 验证等场景——这些场景下同一请求在不同 batch 中必须产生完全一致的 token。对普通推理用户, 升级后默认内核从确定性内核变为 cuBLASLt, 但 batch-invariant 模式会强制切回, 因此行为可预期; 对团队而言, 依赖升级需要同步更新所有环境 (Docker、CI、pyproject), 并保持 GPU 回归测试覆盖。
 - 风险标记: 依赖升级引入行为变更, 全局确定性模式无并发保护, 回归测试依赖 GPU 硬件

关联脉络

- PR #39275 bumping sgl-deep-gemm to 0.2.0 (original): 本 PR 是 #39275 的重建: 原分支被误删, head commit f33320f18a 与代码 diff 完全恢复, body 直接引用该 PR 作为来源。
- PR #39346 chore: bump sglang-kernel version to 0.4.7: 同为依赖升级操作 (sglang-kernel 0.4.7), 与本次 sgl-deep-gemm 升级同属 SGLang 依赖管理演进线, 且都精确锁定版本并要求回归验证。