PR #27529 完整报告

sgl-project/sglang

[AMD] Fix DeepSeek V4 Pro c128 state tensor dtype mismatch error and c4_sparse_raw_indices attribute error in cuda graph phase

合并时间：2026-06-10 23:30

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27529

执行摘要

- 一句话：修复 AMD DeepSeek V4 内核 dtype 不匹配并因性能回归被 revert
- 推荐动作：不推荐作为最终方案精读。核心里程碑是模板参数化设计本身有参考价值，但需要结合性能回归进行重新设计。建议关注后续 PR #27919 (revert) 及后续改进。

功能与动机

DeepSeek V4 Pro 在 AMD 上运行时，c128 压缩 JIT 内核要求 kv_score_buffer / kv_score_input / ape 具有相同 dtype，否则会报错 'dtype=float32 not in allowed [bfloat16]'。此外，在 CUDA graph 阶段会出现 c4_sparse_raw_indices 属性错误。本 PR 旨在解决这两个问题。

实现拆解

1. 内核模板参数扩展：在 c128_v2.cuh 和 c4_v2.cuh 中增加 BufFloat 模板参数，与原有的 InFloat / OutFloat 分开，表示 kv_score_buffer 的 dtype（通常为 float32）。在加载时，将 BufFloat 和 InFloat 数据统一转换为 float32 进行计算。
2. JIT 模块构造更新：在 compress.py 中更新 _jit_compress_module 函数，增加 dtype_buf 参数，并在调用时传入 kv_score_buffer.dtype。
3. Host 端 dtype 转换：在 compressor.py 的 apply_ape_hotfix 方法中，当使用 aiter 后端时，将 self.ape 和 self.norm.weight 转换为 bf16，以与 kv_score_input 的 dtype 一致。
4. 属性添加：在 deepseek_v4_backend_hip_radix.py 中添加 c4_sparse_raw_indices 属性。
5. 测试与验证：PR 附带了 GS M8K 准确率测试（94.5%）和 AMD 上的吞吐 / 延迟基准测试，但未在 NVIDIA 上进行充分的性能回归验证。

关键文件：

- python/sglang/jit_kernel/dsv4/compress.py（模块 JIT 编译；类别 source；类型 core-logic；符号 _jit_compress_module, compress_forward）：核心 JIT 模块构造函数，新增 dtype_buf 模板参数以分离缓冲区和输入 / 输出的 dtype，是 dtype 修复的关键入口。
- python/sglang/srt/layers/attention/dsv4/compressor.py（模块 压缩器；类别 source；类型 core-logic；符号 apply_ape_hotfix）：新增 apply_ape_hotfix 中的 dtype 转换，当使用 aiter 后端时将 ape 和 norm.weight 转为 bf16 以匹配输入 dtype。

- python/sglang/jit_kernel/csrc/deepseek_v4/c4_v2.cuh（模块 C4 内核；类别 source；类型 core-logic；符号 c4_forward）：C4 压缩内核模板添加 BufFloat 参数，修改数据加载逻辑，将缓冲区数据转换为 fp32。
- python/sglang/jit_kernel/csrc/deepseek_v4/c128_v2.cuh（模块 C128 内核；类别 source；类型 core-logic；符号 c128_forward）：C128 压缩内核模板添加 BufFloat 参数，修改数据加载逻辑，将缓冲区数据转换为 fp32。

关键符号：_jit_compress_module, compress_forward, apply_ape_hotfix, c4_forward, c128_forward

关键源码片段

python/sglang/jit_kernel/dsv4/compress.py

核心 JIT 模块构造函数，新增 dtype_buf 模板参数以分离缓冲区和输入 / 输出的 dtype，是 dtype 修复的关键入口。

```
@cache_once
def _jit_compress_module(
    head_dim: int,
    dtype_buf: torch.dtype, # 新增：kv_score_buffer 的 dtype（通常是 float32）
    dtype_in: torch.dtype, # 输入（kv_score_input 和 ape）的 dtype
    dtype_out: torch.dtype, # 输出 dtype
    ratio: Literal[4, 128],
) -> Module:
    args = make_cpp_args(
        head_dim, dtype_buf, dtype_in, dtype_out, is_arch_support_pdl()
    )
    kernel_class = f'FlashCompress{ratio}Kernel<{args}>'
    return load_jit(
        make_name(f'compress_{ratio}_v2'),
        *args,
        cuda_files=[f'deepseek_v4/c{ratio}_v2.cuh'],
        cuda_wrappers=[
            ('decode', f'{kernel_class}::run_decode'),
            ('prefill', f'{kernel_class}::run_prefill'),
        ],
        extra_cuda_cflags=['-use_fast_math'],
    )


def compress_forward(
    kv_score_buffer: torch.Tensor,
    kv_score_input: torch.Tensor,
    ape: torch.Tensor,
    plan: Union[CompressorDecodePlan, CompressorPrefillPlan],
    *,
    head_dim: int,
    compress_ratio: Literal[4, 128],
```

```
    out: Optional[torch.Tensor] = None,
    is_online: bool = False,
) -> torch.Tensor:
    ...
    else:
        # kv_score_buffer（fp32 运行时状态池）的 dtype 可能与 input/ape 不同。
        # 内核通过 BufFloat 模板参数处理缓冲区的 dtype，并在加载时转换为 fp32。
        # ape/weight 在 apply_ape_hotfix 中已转换为 bf16，与 kv_score_input 的 dtype
        # 保持一致，因此只需传递缓冲区的 dtype 即可。
        module = _jit_compress_module(
            head_dim,
            kv_score_buffer.dtype, # 作为 BufFloat 传入内核
            kv_score_input.dtype,
            out.dtype,
            compress_ratio,
        )
    fn = module.decode if plan.is_decode else module.prefill
    fn(kv_score_buffer, kv_score_input, out, ape, *plan[1:3])
    return out
```

python/sglang/srt/layers/attention/dsv4/compressor.py

新增 apply_ape_hotfix 中的 dtype 转换，当使用 aiter 后端时将 ape 和 norm.weight 转为 bf16 以匹配输入 dtype。

```
def apply_ape_hotfix(self):
    assert not self.ape_converted
    self.ape_converted = True

    if self.overlap:
        ape = torch.chunk(self.ape.data, 2, dim=-1)
        ape = torch.cat([ape[0], ape[1]], dim=0)
        self.ape.data.copy_(ape.view(self.ratio, -1))

    if _use_aiter:
        # 当使用 aiter 后端时，ape 和 norm.weight 需要转换为 bf16，
        # 以匹配 kv_score_input 的 dtype，防止内核中 dtype 不匹配。
        self.ape.data = self.ape.data.to(torch.bfloat16)
        self.norm.weight.data = self.norm.weight.data.to(torch.bfloat16)
```

评论区精华

- DarkSharpness 通过微基准测试发现该 PR 在默认 fp32 路径上存在显著性能回归（c4 decode bs=1 延迟从 1.607µs 增加到 2.014µs，+25.3%），并要求重新设计以避免回归。
- kkHuang-amd 建议在 B200 上测试，因为 CUH 文件也用于 CUDA 平台。
- amd-bot CI 显示 NVIDIA 的两个 root cause 任务失败，直接指向 c4_v2.cuh 的变动。
- 存在与 PR #27277 (DaZhUUU) 的重叠实现。

- Perf regression on default fp32 compress path (performance): 性能回归被确认，但未在 PR 内解决；最终导致 PR 被 revert。
- Test on B200 required (testing): 作者未回应是否执行了测试，后续 amd-bot CI 显示 NVIDIA 任务失败。
- NVIDIA CI failures due to c4_v2.cuh change (correctness): CI 显示合并不安全，但 PR 仍然被合并（可能忽略 CI），最终被 revert。
- Overlap with PR #27277 (BF16 state type support) (other): 作者未直接回应；PR 合并后很快被 revert。

风险与影响

- 风险：
  1. 性能回归：在默认 fp32 配置下，c4 和 c128 内核均出现显著的性能下降，尤其是在小 batch size 场景下。这是导致 PR 被 revert 的直接原因。
  2. CUDA 平台影响：虽然修复主要针对 AMD，但内核修改同样影响 CUDA 路径，且未在 NVIDIA 上提供充分的性能基准。
  3. 测试覆盖不足：未添加单元测试覆盖新的 dtype 组合路径。
  4. 被 revert 状态：问题未被修复，后续需要更谨慎的解决方案。 - 影响：该 PR 旨在修复 AMD 用户遇到的 dtype 错误，但意外导致 NVIDIA 路径的性能回归。影响范围限定于 DeepSeek V4 模型使用场景，但由于同时涉及 AMD 和 CUDA，影响面较广。最终被 revert 后，AMD 问题仍待解决。 - 风险标记：CUDA 路径性能回归，被 revert，核变更缺少性能验证，CI 失败仍被合并

关联脉络

- PR #27919 Revert "[AMD] Fix DeepSeek V4 Pro c128 state tensor dtype mismatch error and c4_sparse_raw_indices attribute error in cuda graph phase": 直接 revert 本 PR，因为引入的性能回归和未解决的 NVIDIA 路径问题。
- PR #27525 Related collaborative fix for dtype issue: PR body 中提及与本 PR 协作，共同解决 dtype 问题。
- PR #27277 Add support for BF16 state type: DaZhUUU 提出本 PR 与其存在重叠实现，涉及相同的 dtype 问题。