PR #34484 完整报告

sgl-project/sglang

[ROCm] Fix QuickReduce fp16 saturation corrupting bf16 all-reduces (106M non-finite -> 0, +0.3%)

合并时间：2026-08-30 13:08

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34484

执行摘要

- 一句话：修复 ROCm QuickReduce BF16→FP16 溢出造成的非有限输出
- 推荐动作：值得精读。核心亮点是三层防御：FP16_OVFL 寄存器饱和兜底、per-codec 的 2 幂范围保护（分析清楚为什么量化 codec 不能用）、显式 ISA 转换阻断编译器重排。提交历史中的 S = 16 → 2 → 1 调参过程展示了如何用端到端 perplexity 和相对 L2 数据驱动数值设计决策，是内核级数值修复的可借鉴范本。建议同时阅读 #37132 的空 asm barrier 替代方案，理解 CDNA 系列 ISA 差异对可移植性的约束。

功能与动机

修复 #34473：在 MI350X（gfx950）官方 ROCm 镜像上，gpt-oss-120b 于 TP4 默认配置产生 62.7M 非有限残差元素。Issue 定量确认 "the corruption is created by the all-reduce"，根因是超过 64 MiB 的 all-reduce 默认落入 QuickReduce 的 bf16→fp16 路径（_DEFAULT_CAR_MAX_SIZE 64 MiB 门槛，ROCM_QUICK_REDUCE_CAST_BF16_TO_FP16 默认 1），而 gpt-oss 规约激活值可达 333,824，是 FP16 上限的 5.1 倍。注入实验证明 "One element above 65,504 destroys exactly 32——the CodecQ8 block"，即饱和块 max 污染整个量化块。PR body 明确这是默认路径而非可选路径，且原生 BF16 方案虽数值精确但慢 22–46%，因此选择修复快速路径而非绕开。

实现拆解

变更入口是 ROCm QuickReduce 的 bf16→fp16 快速路径，核心文件为 python/sglang/kernels/aot/csrc/allreduce/quick_all_reduce_base.h 与 quick_all_reduce.cuh。

1. 溢出模式修复：将 set_fp16_ovfl 的编译条件从仅 __gfx942__ 扩展到 __gfx942__ || __gfx950__，使 MODE.FP16_OVFL 在 gfx950 上生效——溢出的 FP16 结果（含 f32→f16 转换）饱和到 ±65504 而非 inf。同时为 s_setreg_imm32_b32 增加 "memory" clobber，防止编译器把 FP16 转换提升到寄存器设置之前执行。
2. 按 codec 划分范围保护：在 quick_all_reduce_base.h 定义 kQRFp16CastScaleLog2Fp=4（S=16，仅CodecFP）与kQRFp16CastScaleLog2Quant=0（S=1，量化codec）。AllReduceTwoshot 在加载侧把 bf16 除以 S 后再窄化到 FP16，在存储侧用 FP32 中间量乘回 S；由于是 2 的幂，sum(x_i / S) * S == sum(x_i) 精确成立。量化 codec 已有块级归一化，提高 S 反而会把 reciprocal-scale 饱和悬崖（blockmax = S * L / 65504）拉进

真实数据。

3. 编译期防重排：新增 scaled_bfloat162_to_half2，用内联 ISA 指令 v_cvt_pk_f16_f32 完成缩放后的窄化转换，使缩放后的 FP32 值成为显式输入，阻止 LLVM 将 (bf16_as_f32 * scale) -> fp16 重排为 fp16(bf16_as_f32) * scale。量化 codec 因 S = 1 保留原有 __float22half2_rn 路径，代码生成不变。
4. 测试配套：新增 test/registered/kernels/test_quick_allreduce_bf16_range.py，四进程（gloo+spawn）TP4 回归测试，覆盖低值、普通值、和超上限、单输入超上限、负向超上限、混合块共 6 类场景 × FP/INT8/INT6/INT4 四种 codec，断言有限性及 FP/low/ordinary 的 bit-exact（rtol=0, atol=0），并注册到 stage-c-test-4-gpu-amd 与 stage-c-test-large-8-gpu-amd-mi35x 两档 AMD CI 网格。
5. 提交演化中的反复调参：初版全局 S = 16 在 GLM-5.2-FP8 上把 INT8 相对 L2 从 0.0057 恶化到 0.68、Qwen3.5-27B-FP8 端到端 perplexity 回归 3.1 倍（mean NLL 1.2955 → 2.4178）；随后将量化 codec 降到 S = 2 仍有余波（GLM-5.2-FP8 +61.7%），最终定为 S = 1，全模型最优且 gpt-oss 也 -1.9%。

关键文件：

- test/registered/kernels/test_quick_allreduce_bf16_range.py（模块 数值回归；类别 test；类型 test-coverage；符号 _get_open_port, _run_bf16_range_test, TestQuickAllReduceBf16Range, test_bf16_range）：新增的四卡回归测试是本次修复的验收标准：覆盖 6 种数值场景 × 4 种 codec，首次把溢出范围行为固化为 CI 断言，并注册到两档 AMD CI 网格，防止该机制复现。
- python/sglang/kernels/aot/csrc/allreduce/quick_all_reduce.cuh（模块 规约内核；类别 source；类型 core-logic；符号 CodecFP, CodecQ4, CodecQ6, CodecQ8）：核心实现文件：新增 scaled_bfloat162_to_half2 显式 ISA 转换、per-codec 的 kCastScaleLog2 接入，以及在 AllReduceTwoshot 加载 / 存储两侧的乘除还原逻辑。
- python/sglang/kernels/aot/csrc/allreduce/quick_all_reduce_base.h（模块 规约内核；类别 source；类型 core-logic；符号 kQRFp16CastScaleLog2Fp, kQRFp16CastScaleLog2Quant, set_fp16_ovfl）：定义了 per-codec 的 FP16 转换缩放常量，并把 FP16_OVFL 寄存器设置扩展到 gfx950，同时新增 memory clobber 防指令重排，是整个修复的配置锚点。

关键符号：scaled_bfloat162_to_half2, set_fp16_ovfl, AllReduceTwoshot<...>::run, _run_bf16_range_test, TestQuickAllReduceBf16Range.test_bf16_range

关键源码片段

test/registered/kernels/test_quick_allreduce_bf16_range.py

新增的四卡回归测试是本次修复的验收标准：覆盖 6 种数值场景 × 4 种 codec，首次把溢出范围行为固化为 CI 断言，并注册到两档 AMD CI 网格，防止该机制复现。

```
# 每个 rank 独立跑 QuickReduce，用 gloo + spawn 起 4 进程模拟 TP4
def _run_bf16_range_test(rank: int, world_size: int, port: int) -> None:
    # 强制走 bf16 -> fp16 快速路径，与生产镜像默认配置一致
    os.environ["ROCM_QUICK_REDUCE_CAST_BF16_TO_FP16"] = "1"
```

```
device = torch.device(f"cuda:{rank}")
torch.cuda.set_device(device)
dist.init_process_group(
    backend="gloo",
    init_method=f"tcp://127.0.0.1:{port}",
    rank=rank,
    world_size=world_size,
)

try:
    numel = 1 << 20
    cases = [
        ("low", 2**-8, False), # 低值：验证 S = 16 不压垮精度
        ("ordinary", 100.0, False), # 普通值：bit-exact 基线
        ("sum_above_fp16", 20_000.0, False), # 4 rank 求和 = 80k，越过 65504 上限
        ("input_above_fp16", 80_000.0, False), # 单个输入就超过 FP16 上限
        ("negative_sum_above_fp16", -20_000.0, False), # 负向溢出对称性
        ("mixed_input_above_fp16", 1.0, True), # 每 32 元素块注入一个异常点
    ]
    # 四种 codec 全验证：FP 走 S = 16 缩放路径，INT 系列保持 S = 1 旧路径
    for quant_mode in ("FP", "INT8", "INT6", "INT4"):
        os.environ["ROCM_QUICK_REDUCE_QUANTIZATION"] = quant_mode
        quick_all_reduce = QuickAllReduce(group=dist.group.WORLD, device=device)
        assert not quick_all_reduce.disabled
        assert quick_all_reduce.use_fp16_kernels
        try:
            for case_name, value, mixed in cases:
                inp = torch.full((numel,), value, dtype=torch.bfloat16, device=device)
                if mixed:
                    inp[::32] = 80_000.0 # 与 issue 中 32 元素块放大机制对齐
                expected = (inp.float() * world_size).to(torch.bfloat16)

                dist.barrier()
                out = quick_all_reduce.quick_all_reduce(inp)
                torch.cuda.synchronize()
                # 核心断言：任何溢出场景都不允许出现非有限输出
                assert (
                    torch.isfinite(out).all().item()
                ), f"{quant_mode=} {case_name=} produced non-finite output"
                # FP codec 以及低值 / 普通值要求 bit-exact（rtol=0, atol=0）
                if quant_mode == "FP" or case_name in ("low", "ordinary"):
                    torch.testing.assert_close(
                        out,
                        expected,
                        rtol=0,
                        atol=0,
                        msg=lambda msg: f"{quant_mode=} {case_name=}\n{msg}",
                    )
        finally:
```

```
            quick_all_reduce.close()
    finally:
        dist.destroy_process_group()
```

python/sglang/kernels/aot/csrc/allreduce/quick_all_reduce.cuh

核心实现文件：新增 scaled_bfloat162_to_half2 显式 ISA 转换、per-codec 的 kCastScaleLog2 接入，以及在 AllReduceTwoshot 加载 / 存储两侧的乘除还原逻辑。

```
// CodecFP 没有块缩放，因此 S = 16 是它唯一的范围保护；
// 量化 codec 已有块级归一化，S 提高只会把 rcp 饱和悬崖拉进真实数据，故保持 S = 1。
//  （常量定义见 quick_all_reduce_base.h：kQRFp16CastScaleLog2Fp =
4、kQRFp16CastScaleLog2Quant = 0）

// 在 FP32 侧完成缩放后再窄化到 FP16，避免 LLVM 把
// (bf16_as_f32 * scale) -> fp16 重排成 fp16(bf16_as_f32) * scale，
// 那样会先裁剪掉超过 65504 的值，绕开溢出保护。
// 这里用内联 ISA 转换让缩放后的 FP32 值成为显式输入，编译器无法再做 reassociation。
__quickreduce_device_inline__ half2 scaled_bfloat162_to_half2(nv_bfloat162 value, float scale) {
  float2 scaled = __bfloat1622float2(value);
  scaled.x *= scale;
  scaled.y *= scale;

  int packed;
  asm volatile("v_cvt_pk_f16_f32 %0, %1, %2" : "=v"(packed) : "v"(scaled.x), "v"(scaled.y));
  return *reinterpret_cast<half2*>(&packed);
}

template <typename T, class Codec, bool cast_bf2half>
struct AllReduceTwoshot {
  // 2 的幂缩放：加载除以 S、存储乘回 S，移位精确，
  // 因此 sum(x_i / S) * S == sum(x_i) 严格成立。
  static constexpr float kCastScale = static_cast<float>(1 << Codec::kCastScaleLog2);
  static constexpr float kCastInvScale = 1.0f / kCastScale;

  // 加载侧：bf16 -> fp16 之前先除以 S。S = 1 的量化 codec 走原路径，代码生成不变
  if constexpr (Codec::kCastScaleLog2 == 0) {
    float2 f = __bfloat1622float2(bf_buf[j]);
    f.x *= kCastInvScale;
    f.y *= kCastInvScale;
    half_buf[j] = __float22half2_rn(f);
  } else {
    half_buf[j] = scaled_bfloat162_to_half2(bf_buf[j], kCastInvScale);
  }

  // 存储侧：规约完成后在 FP32 中乘回 S，FP32 中间量不会溢出
  float2 f = __half22float2(half_buf[j]);
  f.x *= kCastScale;
  f.y *= kCastScale;
  bf16_buf[j] = __float22bfloat162_rn(f);
```

```
};
```

python/sglang/kernels/aot/csrc/allreduce/quick_all_reduce_base.h

定义了 per-codec 的 FP16 转换缩放常量，并把 FP16_OVFL 寄存器设置扩展到 gfx950，同时新增 memory clobber 防指令重排，是整个修复的配置锚点。

```
// 范围保护常量：bf16 -> fp16 快速路径（AllReduceTwoshot<..., true>）中 FP16 在 65504 饱和。
// 采用 2 的幂缩放：加载除以 S、存储乘回 S，因此 sum(x_i / S) * S == sum(x_i) 精确成立。
//
// 为什么按 codec 分开：CodecFP 没有块缩放，S 是它唯一的范围保护；
// 量化 codec 已按 32 个值一组用块 max 归一化，MODE.FP16_OVFL 保证超上限元素不会变成 inf
// 污染整个块。提高 S 会把 reciprocal 缩放饱和悬崖
//  （blockmax = S * L / 65504，L = 8 / 32 / 128，对应 Q4 / Q6 / Q8）移向真实数据——
// 实测 S: 1 -> 2 在 GLM-5.2 上 perplexity 恶化 62%，因此保持 S = 1。
static constexpr int kQRFp16CastScaleLog2Fp = 4; // S = 16，仅 CodecFP
static constexpr int kQRFp16CastScaleLog2Quant = 0; // S = 1，CodecQ4 / CodecQ6 / CodecQ8

// MODE.FP16_OVFL 把溢出的 FP16 结果（包括 v_cvt_pk_f16_f32 转换）饱和到
// ±MAX_FP16，而不是产生 inf。asm 上的 "memory" clobber 是必须的：
// 没有它编译器可能把一次转换提升到 s_setreg 之前执行，那个转换仍会产生 inf。
__quickreduce_device_inline__ static void set_fp16_ovfl(bool const value) {
#if defined(__gfx942__) || defined(__gfx950__)
  if (value) {
    asm volatile("s_setreg_imm32_b32 0xdc1, 1;" ::: "memory");
  } else {
    asm volatile("s_setreg_imm32_b32 0xdc1, 0;" ::: "memory");
  }
#endif
}
```

评论区精华

合并前 review 只有一条 HaiShaw 的 APPROVED：“LGTM”，未产生设计分歧。合并后 yctseng0211 在 Issue 评论中报告了构建回归：“the inline v_cvt_pk_f16_f32 here only assembles on gfx950, so every gfx942 build fails with error: instruction not supported on this GPU and AMD CI is currently red on main”，并说明已开 PR #37132，“keeps the reassociation blocked with an empty asm barrier instead”。此外，提交历史本身记录了重要的数值权衡论证：ebc5e8b 与 3ef920e 用实测数据说明全局缩放对量化 codec 的伤害机制是 rcp 饱和悬崖而非 denormal，最终以 S = 1 收口。

- gfx942 上 v_cvt_pk_f16_f32 无法汇编导致 CI 变红 (correctness)：本 PR 引入的内联 ISA 指令不具备 CDNA3 可移植性；#37132 以空 asm barrier 替代，既保留防重排语义，又恢复 gfx942 构建。

风险与影响

- 风险：

1. gfx942 构建回归（已发生）：scaled_bfloat162_to_half2 中的内联 v_cvt_pk_f16_f32 是 gfx950 专属指令，gfx942 无法汇编，导致合并后主分支 AMD CI 变红；后续 #37132 用空 asm barrier 替代，风险已解除，但这说明本 PR 的 CI 验证未覆盖 CDNA3 的实际编译。
2. 依赖编译器与 ISA 屏障：s_setreg 的 memory clobber 和显式 ISA 转换都依赖编译器不再重排窄化转换；一旦未来 LLVM 重新获得 reassociation 机会，溢出会复发。测试只验证运行结果，不验证指令顺序。
3. 量化 codec 的遗留截断：PR body 明确量化规约在 FP16 累加器超过 65,504 时仍会 clamp 有限值，这是预先存在的有损行为，未在本 PR 消除；对 gpt-oss 这类大规约幅值模型，仍可能损失精度，只是不再产生 inf。
4. 低值精度盲区：S = 16 把低值压向 FP16 的 denormal 区间，测试覆盖 2^-8 但未覆盖更小量级；对极小激活的模型可能有额外精度损失（body 声明 CodecFP 在已测场景 bit-exact）。
5. 性能波动：实测 INT 路径无回归（16 MiB 0.4% 内、90 MiB 0.1% 内），FP 路径反而快 1.1–9.6%，但这是 MI350X 单机结果，其他 CDNA 架构上的调度差异存在不确定性。 - 影响：影响用户：所有 ROCm（gfx942/gfx950）上超过 64 MiB 门槛、默认走 QuickReduce 快速路径的 bf16 all-reduce 都受益；gpt-oss-120b 默认配置的非有限输出从 62.7M 降到 0，同类大规约幅值模型不再产生 NaN。影响系统：避免了切换到慢 22–46% 的原生 BF16 内核，保持了 all-reduce 吞吐；CodecFP 在溢出场景下保持 bit-exact，量化 codec 的低值 / 普通值路径不变。影响团队：新增 4-GPU 与 8-GPU 两档 AMD CI 套件（est_time=30 秒），后续 QuickReduce 修改需要同时验证 gfx942/gfx950 双架构编译与数值范围，并因本 PR 的构建疏漏产生了下游修复 #37132。 - 风险标记：gfx942 构建回归，依赖编译器屏障，量化截断遗留，新增 4 卡 CI 测试

关联脉络

- PR #37132 [AMD] Fix the QuickReduce bf16 cast failing to build for CDNA: 直接修复本 PR 遗留的 gfx942 构建回归：改动同一组 quick_all_reduce_base.h / quick_all_reduce.cuh，用空 asm barrier 替代 gfx950 专属的 v_cvt_pk_f16_f32，保持编译器不重排缩放的语义。