PR #24172 完整报告

sgl-project/sglang

Fix flashinfer workspace OOM

合并时间：2026-05-04 16:26

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24172

执行摘要

- 一句话：修复 FlashInfer 工作区 OOM 导致分布式测试挂起
- 推荐动作：建议相关开发者精读分布式预检设计模式，重点关注 flashinfer_comm_fusion.py 中的内存分配计算和 all_reduce 投票逻辑。此变更为未来类似内存预检提供了可复用的模式。

功能与动机

在 CI 环境中，test_gpt_oss_120b.py 偶尔因 FlashInfer *create_allreduce_fusion_workspace* 触发 OOM，导致 NCCL watchdog 超时，测试进程挂起。作者经调研确认 FlashInfer 层面无更优解法，故从 SGLang 侧引入预检。

实现拆解

1. 添加 CUDA 驱动绑定函数：在 common.py 中新增 get_cuda_driver_bindings，统一 cuda.bindings.driver 和 cuda.cuda 的导入。
2. 实现内存分配属性构造：在 flashinfer_comm_fusion.py 中新增 _make_flashinfer_workspace_allocation_prop，根据平台选择合适的句柄类型，填充 CUmemAllocationProp。
3. 镜像 FlashInfer 大小计算：_flashinfer_trtllm_workspace_allocation_sizes 模拟 FlashInfer 内部 SymmDeviceMemory 逻辑，计算各缓冲区大小并考虑对齐和 multicast 粒度。
4. 单 rank 探测函数 _probe_cumem_create_sequence：对每个分配大小尝试 *cuMemCreate* 和 *cuMemRelease*，返回成功或失败。
5. 分布式预检与投票：_preflight_check_workspace_memory 内部调用探测函数，并将各 rank 的布尔结果通过 dist.all_reduce with BAND 聚合，仅当所有 rank 返回 True 时才允许创建融合 workspace。
6. 修改初始化入口：在 FlashInferWorkspaceManager.initialize 中增加预检调用，失败时跳过 workspace 创建。
7. 分布式测试：新增 test_flashinfer_fusion_preflight.py，使用 multiprocessing spawn 启动 2 进程模拟分布式环境，覆盖正常和显存耗尽场景。

关键文件：

- python/sglang/srt/layers/flashinfer_comm_fusion.py（模块 通信融合层；类别 source；类型 core-logic；符号 _make_flashinfer_workspace_allocation_prop, _flashinfer_trtllm_workspace_allocation_sizes, _probe_cumem_create_sequence, _preflight_check_workspace_memory）：核心变更文件，新增预检函数并修改 workspace 初始化流程
- test/registered/distributed/test_flashinfer_fusion_preflight.py（模块 分布式测试；类别 test；类型 test-coverage；符号 _get_free_port, _run_rank, _spawn_and_collect, TestFlashInferPreflightDistributed）：新增的分布式测试，验证预检正常和 starvation 场景
- python/sglang/srt/utils/common.py（模块 工具函数；类别 source；类型 dependency-wiring；符号 get_cuda_driver_bindings）：添加 get_cuda_driver_bindings 工具函数

关键符号：_make_flashinfer_workspace_allocation_prop, _flashinfer_trtllm_workspace_allocation_sizes, _probe_cumem_create_sequence, _preflight_check_workspace_memory, get_cuda_driver_bindings

关键源码片段

python/sglang/srt/layers/flashinfer_comm_fusion.py

核心变更文件，新增预检函数并修改 workspace 初始化流程

```
def _make_flashinfer_workspace_allocation_prop(cuda_driver):
    # 构建 CUmemAllocationProp，用于后续 cuMemCreate 探测
    if _should_force_posix_fd_transport():
        handle_type = (
            cuda_driver.CUmemAllocationHandleType.CU_MEM_HANDLE_TYPE_POSIX_FILE_
            DESCRIPTOR
        )
    else:
        from flashinfer.comm.mnnvl import is_mnnvl_fabric_supported
        # 优先使用 Fabric 句柄类型（若支持）
        if is_mnnvl_fabric_supported(torch.cuda.current_device()):
            handle_type = (
                cuda_driver.CUmemAllocationHandleType.CU_MEM_HANDLE_TYPE_FABRIC
            )
        else:
            handle_type = (
                cuda_driver.CUmemAllocationHandleType.CU_MEM_HANDLE_TYPE_POSIX_FILE_
                DESCRIPTOR
            )
    prop = cuda_driver.CUmemAllocationProp()
    prop.requestedHandleTypes = handle_type
    prop.type = cuda_driver.CUmemAllocationType.CU_MEM_ALLOCATION_TYPE_PINNED
    prop.location = cuda_driver.CUmemLocation()
    prop.location.type = cuda_driver.CUmemLocationType.CU_MEM_LOCATION_TYPE_DEVICE
    prop.location.id = torch.cuda.current_device()
```

```
    prop.allocFlags.gpuDirectRDMACapable = 1
    return prop
```

test/registered/distributed/test_flashinfer_fusion_preflight.py

新增的分布式测试，验证预检正常和 starvation 场景

```
def _run_rank(rank, world_size, port, scenario, result_q):
    held = None
    cuda_driver = None
    try:
        os.environ['MASTER_ADDR'] = '127.0.0.1'
        os.environ['MASTER_PORT'] = str(port)
        os.environ['RANK'] = str(rank)
        os.environ['WORLD_SIZE'] = str(world_size)
        os.environ['LOCAL_RANK'] = str(rank)
        torch.cuda.set_device(rank)
        import torch.distributed as dist
        dist.init_process_group(backend='gloo', rank=rank, world_size=world_size)
        cpu_group = dist.group.WORLD

        from sglang.srt.layers.flashinfer_comm_fusion import (
            _make_flashinfer_workspace_allocation_prop,
            _preflight_check_workspace_memory,
        )
        probe_kwargs = dict(
            world_size=8, max_token_num=2048, hidden_dim=12288,
            dtype=torch.bfloat16, cpu_group=cpu_group,
        )
        # 场景： rank0 饥饿（模拟显存不足）
        if scenario == 'rank0_starved' and rank == 0:
            cuda_driver = get_cuda_driver_bindings()
            prop = _make_flashinfer_workspace_allocation_prop(cuda_driver)
            free, _total = torch.cuda.mem_get_info(rank)
            target = max(free - (1 << 30), 0) # 预留 1GB
            err, gran = cuda_driver.cuMemGetAllocationGranularity(
                prop,
                cuda_driver.CUmemAllocationGranularity_flags.CU_MEM_ALLOC_GRANULARITY_
                RECOMMENDED,
            )
            assert err == cuda_driver.CUresult.CUDA_SUCCESS, err
            aligned = (target // gran) * gran
            assert aligned > 0, 'not enough free memory'
            err, held = cuda_driver.cuMemCreate(aligned, prop, 0)
            assert err == cuda_driver.CUresult.CUDA_SUCCESS, (err, aligned)

        decision = _preflight_check_workspace_memory(**probe_kwargs)
        result_q.put((rank, 'ok', bool(decision)))
    except Exception as e:
        result_q.put((rank, 'err', repr(e)))
```

```
    finally:
        if held is not None:
            cuda_driver.cuMemRelease(held)
        try:
            if dist.is_initialized():
                dist.destroy_process_group()
        except Exception:
            pass
```

评论区精华

无公开 review 讨论。Fridge003 直接批准。作者在 PR body 中说明已花费时间研究 FlashInfer 内部，确认此修复是合理的且无其他方案。

- 暂无高价值评论线程

风险与影响

- 风险：预检逻辑与 FlashInfer 内部分配算法紧密耦合，FlashInfer 升级后可能不同步导致预检失效或误判；AllReduce BAND 依赖 CPU group，若通信故障可能导致一致性错误；但预检失败仅跳过融合，不影响正常推理。
- 影响：修复 CI 稳定性，减少因 OOM 导致的测试挂起；对用户透明，预检仅在初始化执行一次，无性能开销；若预检失败回退到非融合 allreduce，不影响结果正确性。
- 风险标记：依赖 CUDA 驱动底层 API，镜像 FlashInfer 内部分配逻辑，AllReduce BAND 通信可能失败

关联脉络

- 暂无明显关联 PR