

PR #48879 完整报告

vllm-project/vllm

[Core] Fail fast when /dev/shm is too small for the shm ring buffer

合并时间: 2026-07-28 03:48

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/48879>

执行摘要

- 一句话: 共享内存不足时提前报错而非 SIGBUS
- 推荐动作: 值得精读, 体现加固系统鲁棒性的最佳实践: 前置检查、清晰错误、测试覆盖。设计上选择 RuntimeError 而非更复杂的信号拦截, 简单有效。建议作为类似预检模式的参考。

功能与动机

在 Docker 或 Kubernetes 默认的 64MB /dev/shm 环境下, $TP \geq 2$ 时多模态负载容易触发 SIGBUS, 导致 Engine Dead Error 且难以定位。PR body 指出“When usage crosses the tmpfs limit the worker is killed by an uncatchable SIGBUS, which surfaces to the user as an opaque EngineDeadError”。该 PR 旨在让失败快速且可操作。

实现拆解

实现分为 4 步:

1. 在 shm_broadcast.py 中定义 SHM_PATH 常量并新增 check_shm_free_space 函数, 使用 shutil.disk_usage 获取空闲空间, 不足时抛出 RuntimeError 并提示调整参数。
2. 在 ShmRingBuffer.__init__ 的创建分支中 (name is None), 在 shared_memory.SharedMemory() 调用前插入 check_shm_free_space 预检。
3. 新增测试文件 test_shm_broadcast.py 中的测试函数, 利用 mock.patch 模拟 shutil.disk_usage, 覆盖空间不足 (抛出)、空间充足 (通过)、路径不存在 (跳过) 三种场景, 以及 ShmRingBuffer 创建时的集成验证。
4. 根据 review 意见精简错误消息和注释, 删除冗余说明。

关键文件:

- vllm/distributed/device_communicators/shm_broadcast.py (模块 共享内存; 类别 source; 类型 core-logic; 符号 check_shm_free_space): 核心修改文件, 新增 check_shm_free_space 函数并在 ShmRingBuffer.__init__ 中调用, 实现预检逻辑。
- tests/distributed/test_shm_broadcast.py (模块 共享内存测试; 类别 test; 类型 test-coverage; 符号 _fake_disk_usage, test_check_shm_free_space_raises_when_insufficient, test_check_shm_free_space_passes_when_sufficient, test_check_shm_free_space_skipped_when_path_missing): 添加完整的单元测试, 覆盖

check_shm_free_space 的三种场景和 ShmRingBuffer 集成验证，确保预检逻辑正确。

关键符号：check_shm_free_space, ShmRingBuffer.init

关键源码片段

vllm/distributed/device_communicators/shm_broadcast.py

核心修改文件，新增 check_shm_free_space 函数并在 ShmRingBuffer.__init__ 中调用，实现预检逻辑。

```
# vllm/distributed/device_communicators/shm_broadcast.py

import os
import shutil

# 默认共享内存路径
SHM_PATH = "/dev/shm"

def check_shm_free_space(required_bytes: int, shm_path: str = SHM_PATH) -> None:
    """Raise if ``shm_path`` cannot fit a ``required_bytes`` shared segment."""
    # 共享内存路径不存在时（非 Linux 系统），跳过检查
    if not os.path.isdir(shm_path):
        return
    # 获取当前空闲字节数
    free_bytes = shutil.disk_usage(shm_path).free
    if required_bytes <= free_bytes:
        return # 空间充足，无需报警
    mib = 1 << 20
    raise RuntimeError(
        f"Insufficient space in {shm_path}: {required_bytes / mib:.0f} MiB "
        f"required, {free_bytes / mib:.0f} MiB free. Increase {shm_path} "
        "(e.g. --shm-size or --ipc=host).")
    )

class ShmRingBuffer:
    def __init__(self, n_reader, max_chunk_bytes, max_chunks, name=None):
        # ... 前面的初始化 ...
        if name is None: # 创建者
            self.is_creator = True
            # 预检：确保 /dev/shm 能容纳缓冲区
            check_shm_free_space(self.total_bytes_of_buffer)
            self.shared_memory = shared_memory.SharedMemory(
                create=True, size=self.total_bytes_of_buffer
            )
```

tests/distributed/test_shm_broadcast.py

添加完整的单元测试，覆盖 check_shm_free_space 的三种场景和 ShmRingBuffer 集成验证，确保预检逻辑正确。

```

# tests/distributed/test_shm_broadcast.py

from types import SimpleNamespace
from unittest import mock

# 辅助函数：模拟 shutil.disk_usage 的返回值，仅提供 free 字段
def _fake_disk_usage(free_bytes: int):
    return SimpleNamespace(total=free_bytes, used=0, free=free_bytes)

def test_check_shm_free_space_raises_when_insufficient(tmp_path):
    # 模拟空闲空间 32 MiB，但请求 240 MiB，应抛出 RuntimeError
    with (
        mock.patch.object(
            shm_broadcast.shutil, "disk_usage",
            return_value=_fake_disk_usage(32 << 20)
        ),
        pytest.raises(RuntimeError, match="Insufficient space"),
    ):
        check_shm_free_space(240 << 20, shm_path=str(tmp_path))

def test_check_shm_free_space_passes_when_sufficient(tmp_path):
    # 模拟空闲空间 512 MiB，请求 240 MiB，应正常通过
    with mock.patch.object(
        shm_broadcast.shutil, "disk_usage",
        return_value=_fake_disk_usage(512 << 20)
    ):
        check_shm_free_space(240 << 20, shm_path=str(tmp_path))

def test_check_shm_free_space_skipped_when_path_missing(tmp_path):
    # 路径不存在（如非 Linux），即使请求 1 EiB 也不应报错
    check_shm_free_space(1 << 60, shm_path=str(tmp_path / "does-not-exist"))

def test_shm_ring_buffer_creation_checks_free_space():
    # 集成测试：ShmRingBuffer 创建时空间不足应抛出
    with (
        mock.patch.object(
            shm_broadcast.shutil, "disk_usage",
            return_value=_fake_disk_usage(1 << 20)
        ),
        mock.patch.object(shm_broadcast.os.path, "isdir", return_value=True),
        pytest.raises(RuntimeError, match="Insufficient space"),
    ):
        ShmRingBuffer(n_reader=1, max_chunk_bytes=24 * 1024 * 1024, max_chunks=10)

```

评论区精华

主要讨论点：

- njhill 认为注释和错误信息过于冗长 (“I just feel the comments and error message are a bit too verbose”)。作者随即修剪了内容, njhill 认可。
- njhill 提醒不要频繁 rebase 以免重启全量 CI。
- joerunde 表示赞赏前置检查设计 (“Big fan of up front checks for invalid configs”)。
- 错误消息和注释的冗长度 (style): 作者修剪了注释和错误消息后, njhill 认可。

风险与影响

- 风险: 风险极低。变更仅在前置检查路径增加 `shutil.disk_usage` 调用, 正常路径无影响。非 Linux 平台通过 `os.path.isdir` 检查跳过预检。主要风险是 mock 测试未覆盖不同文件系统行为, 但 `shutil.disk_usage` 是标准库, 行为稳定。无性能影响。无安全新面。
- 影响: 影响范围: 仅 $TP \geq 2$ 时使用共享内存环形缓冲区的部署。用户将看到清晰 `RuntimeError` 而非 `SIGBUS/EngineDeadError`, 错误信息包含调整建议。系统在启动时失败而非运行中崩溃, 降低调试成本。对正确配置的部署零影响。
- 风险标记: Linux 依赖, 低风险

关联脉络

- 暂无明显关联 PR