PR #45086 完整报告

vllm-project/vllm

[Bugfix][CPU] Honor cgroup memory limit when computing KV cache size

合并时间：2026-06-15 10:26

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45086

执行摘要

- 一句话：修复 CPU 容器内 KV 缓存大小计算错误
- 推荐动作：建议精读。本 PR 展示了处理 Linux 容器内资源感知的经典模式，代码简洁且边界处理完整（v1/v2 回退、哨兵值、文件异常）。可作为在 vLLM 中处理 cgroup 限制的参考实现。

功能与动机

用户在容器（Docker/Kubernetes Pod）中运行 vLLM 时遇到 ValueError: Available memory on node 0 (149.11/692.65 GiB) ... is less than requested memory for kv (206.54/207.8 GiB) 错误。根本原因是 gpu_memory_utilization 基于宿主机 NUMA 节点总内存（692.65 GiB）计算，但容器仅被分配约 150 GiB 内存，导致 KV 缓存分配超出限制。

实现拆解

1. 新增 _read_int_file 工具函数：安全读取 cgroup 文件内容，处理空值、'max' 及文件异常，返回 int | None。
2. 新增 get_cgroup_memory_limit 函数（带 @cache 装饰器）：先尝试 cgroup v2 路径（memory.max + memory.current），若不存在则回退到 cgroup v1（memory.limit_in_bytes + memory.usage_in_bytes），并排除 v1 中大于等于 2^62 的无穷大哨兵值。非 Linux 系统直接返回 (None, None)。
3. 修改 get_memory_node_info 函数：在计算出 NUMA 节点的 total_memory 和 available_memory 后，调用 get_cgroup_memory_limit() 获取 cgroup 限制。若 cgroup 限制存在且小于 NUMA 总内存，则将 total_memory 设为 cgroup 限制值，并基于 cgroup_limit - cgroup_usage 计算 cgroup 可用内存，取 min(available_memory, cgroup_available) 且最低为 0。
4. 未新增测试文件：PR 仅修改了一个源码文件（vllm/utils/cpu_resource_utils.py），无对应单元测试。变更涉及文件系统读取，测试依赖容器环境，作者在 PR 描述中提供了手动测试结果。

关键文件：

- vllm/utils/cpu_resource_utils.py（模块 资源管理；类别 source；类型 core-logic；符号 _read_int_file, get_cgroup_memory_limit）：唯一的变更文件，新增 cgroup 内存限制读取函数并在 KV 缓存计算中应用钳制逻辑。

关键符号：_read_int_file, get_cgroup_memory_limit

关键源码片段

vllm/utils/cpu_resource_utils.py

唯一的变更文件，新增 cgroup 内存限制读取函数并在 KV 缓存计算中应用钳制逻辑。

```
def _read_int_file(path: str) -> int | None:
    # 安全读取 cgroup 文件，处理 'max'、空值及异常
    try:
        with open(path) as f:
            value = f.read().strip()
        if not value or value == "max":
            return None # cgroup v2 中用 'max' 表示无限制
        return int(value)
    except (OSError, ValueError):
        return None # 文件不存在或格式错误时不阻塞


@cache
def get_cgroup_memory_limit() -> tuple[int | None, int | None]:
    """Return (limit, usage) in bytes from cgroup, or (None, None).

    Supports both cgroup v2 (unified) and v1. Returns (None, None) when
    not running under a constrained cgroup (e.g. bare metal, or limit
    reported as `max`/an unrealistically large value).
    """
    if sys.platform != "linux":
        return None, None

    # cgroup v2 unified hierarchy
    v2_limit = _read_int_file("/sys/fs/cgroup/memory.max")
    if v2_limit is not None:
        v2_usage = _read_int_file("/sys/fs/cgroup/memory.current")
        return v2_limit, v2_usage

    # cgroup v1
    v1_limit = _read_int_file("/sys/fs/cgroup/memory/memory.limit_in_bytes")
    if v1_limit is not None:
        # cgroup v1 在无限制时返回接近 PAGE_COUNTER_MAX 的哨兵值
        if v1_limit >= (1 << 62):
            return None, None
        v1_usage = _read_int_file("/sys/fs/cgroup/memory/memory.usage_in_bytes")
        return v1_limit, v1_usage

    return None, None


def get_memory_node_info(node_id: int = 0) -> MemoryNodeInfo:
```

```
    # ... 原有逻辑计算 total_memory 和 available_memory ...

    # 在返回前根据 cgroup 限制钳制内存值
    cgroup_limit, cgroup_usage = get_cgroup_memory_limit()
    if cgroup_limit is not None and cgroup_limit < total_memory:
        total_memory = cgroup_limit # 总内存不能超过 cgroup 限制
        cgroup_available = cgroup_limit - (cgroup_usage or 0)
        available_memory = max(0, min(available_memory, cgroup_available))

    return MemoryNodeInfo(
        total_memory=total_memory,
        available_memory=available_memory,
    )
```

评论区精华

审核者 bigPYJ1151 最初建议使用 cgroup/memory.numa_stat 获取每 NUMA 节点的内存状态，但 maobaolong 尝试后发现 cgroup 不提供每 NUMA 节点的限制和空闲大小，只能进行近似分配。最终 bigPYJ1151 建议忽略 NUMA 细节，仅当 cgroup 有内存限制时直接钳制总内存大小，maobaolong 采纳并移除了 numa_stat 解析。

- 是否使用 cgroup/memory.numa_stat (design): 弃用 numa_stat 方案，仅对总内存进行全局钳制。

风险与影响

- 风险：
  1. 线性回归风险低：变更仅在 get_memory_node_info 末尾增加钳制逻辑，不影响裸机（cgroup 无限制时 get_cgroup_memory_limit 返回 (None, None)）。
  2. cgroup 路径不存在的异常处理：_read_int_file 捕获 OSError 和 ValueError 返回 None，路径不存在时不会崩溃。
  3. cgroup v1 哨兵值判断：使用 >= (1 << 62) 排除无限大值，与内核 PAGE_COUNTER_MAX 一致。
  4. available_memory 计算保守：取 min(available_memory, cgroup_available) 且最低为 0，确保不超额分配。
  5. 无单元测试：虽手动验证充分，但缺少自动化测试，未来重构可能导致回归。 - 影响：影响范围：仅限 Linux 平台 CPU 后端在容器内运行且设置了 cgroup 内存限制的场景。裸机、macOS、GPU 后端不受影响。影响程度：对受影响用户而言是功能性修复，解决容器内启动崩溃。对未受影响用户无行为变化。团队影响：无，仅修改一个文件。
- 风险标记：核心路径变更，手动测试为主，cgroup 路径依赖

关联脉络

- 暂无明显关联 PR