PR #41268 完整报告

vllm-project/vllm

[UX][Bugfix] Fix OOM by setting PyTorch `max_split_size_mb` during model loading

合并时间：2026-04-30 22:46

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41268

执行摘要

- 一句话：修复 PyTorch 内存碎片导致 OOM
- 推荐动作：值得精读，尤其是理解 PyTorch CUDA 内存分配器 max_split_size_mb 的作用和碎片化问题的诊断方法。对于维护者，需关注 sleep mode 下的副作用并准备后续修复。

功能与动机

PyTorch 从 2.9 升级到 2.10 后，triton-kernels 的 swizzle_data 内核增加了预填充步骤，改变了分配大小，导致内存碎片化加剧。例如 GPT-OSS 120B 的 MoE 权重处理时，w13（530 MiB）和 w2（270 MiB）分配因缓存块不合理分割而产生 260 MiB 的残余碎片，无法满足后续分配，最终 OOM。

实现拆解

1. 新增辅助函数：在 vllm/v1/worker/gpu_worker.py 中添加 _scoped_allocator_max_split 上下文管理器，临时设置 max_split_size_mb=20（PyTorch 允许的最小值），并在退出时依据 PYTORCH_CUDA_ALLOC_CONF 环境变量中的原始值或 SIZE_MAX 进行恢复。
2. 调整导入：新增 *contextmanager* 和 *regex as re* 导入，用于实现上下文管理器和解析环境变量。
3. 包裹加载流程：在 load_model 方法中，将 self._scoped_allocator_max_split(max_split_size_mb=20) 作为上下文包裹 self.model_runner.load_model() 调用，仅在模型加载阶段生效。
4. 平台限制：函数内部通过 current_platform.is_cuda() 检查，仅在 CUDA 设备上执行设置，其他平台直接 yield。
5. 测试与验证：PR 描述中提供了测试命令和结果表，对比了默认（OOM）、128 MiB 和 20 MiB 三种配置，20 MiB 下加载时间仅增加约 0.8 秒且通过。

关键文件：

- vllm/v1/worker/gpu_worker.py（模块 GPU worker；类别 source；类型 core-logic；符号 _scoped_allocator_max_split）：核心变更文件，新增 _scoped_allocator_max_split 上下文管理器并在 load_model 中启用，确保模型加载期间内存分配碎片最小化。

关键符号：_scoped_allocator_max_split

关键源码片段

vllm/v1/worker/gpu_worker.py

核心变更文件，新增 _scoped_allocator_max_split 上下文管理器并在 load_model 中启用，确保模型加载期间内存分配碎片最小化。

```
@contextmanager
def _scoped_allocator_max_split(self, max_split_size_mb: int):
    """
    临时设置 PyTorch CUDA 内存分配器的 `max_split_size_mb`，
    以减少模型加载时的内存碎片。
    退出时自动恢复原始值。
    """
    # 仅在 CUDA 平台生效
    if not current_platform.is_cuda():
        yield
        return

    # 从环境变量中解析原始 max_split_size_mb 值
    conf = os.environ.get("PYTORCH_CUDA_ALLOC_CONF", "")
    match = re.search(r"max_split_size_mb:(\d+)", conf)
    original_value = match.group(1) if match else None

    # 设置新的最大值（20 MiB 是 PyTorch 允许的最小值）
    torch._C._accelerator_setAllocatorSettings(
        f"max_split_size_mb:{max_split_size_mb}"
    )
    try:
        yield
    finally:
        # PyTorch 默认值为 SIZE_MAX（无限制）
        _SIZE_MAX_MB = (2**64 - 1) // (1024 * 1024)
        restore = original_value if original_value else str(_SIZE_MAX_MB)
        torch._C._accelerator_setAllocatorSettings(f"max_split_size_mb:{restore}")
```

评论区精华

- 环境变量解析方式：gemini-code-assist[bot] 建议用字符串拆分替代正则，认为正则更脆弱。MatthewBonanni 回应指出建议的方法实际上更脆弱（因为 \d+ 仅捕获数字），最终代码保留了正则实现。
- 恢复值问题：tlrmchlsmth 担心使用固定大数 *99999999* 作为默认值可能导致异常行为。MatthewBonanni 在第二次提交中改用 (2**64 - 1) // (1024 * 1024) 计算 SIZE_MAX，与 PyTorch 内部默认值一致。
- 后续回归报告：aoshen02 报告该 PR 导致 Qwen3-235B-A22B 在 sleep mode 下 OOM（~70 GiB 内存无法回收），MatthewBonanni 确认是预期副作用，需后续优化。
- 环境变量解析方式的选择 (design)：维持正则实现，并承诺解析逻辑足够健壮。
- 恢复默认值的正确性 (correctness)：采用 SIZE_MAX 计算方式修复潜在问题。

- 对 sleep mode 的副作用 (performance): 已知问题，需后续 PR 专门修复 sleep mode 兼容性。

风险与影响

- 风险：
  1. sleep mode 兼容性：Issue 评论指出设置 max_split_size_mb 后，CuMemAllocator 的私有池在 sleep mode 下无法回收碎片，导致约 70 GiB 内存滞留（该 PR 是修复的副作用）。
  2. 加载时间微增：从测试结果看，20 MiB 下加载时间增加约 0.8 秒（23.76s → 24.58s），通常可接受。
  3. 非 CUDA 平台无影响：通过平台检查隔离，对其他后端透明。
  4. 环境变量解析脆弱性：若 PYTORCH_CUDA_ALLOC_CONF 格式有变化，正则可能失效，但当前实现已通过 review。 - 影响：直接修复了 GPT-OSS 120B 等模型在 PyTorch 2.10+ 下启动时的 OOM 问题，使这类配置能成功加载。间接影响是可能对 sleep mode 用户造成新的 OOM（如 Qwen3-235B），需后续针对性优化。整体影响范围限于模型加载阶段，对运行时性能无影响。 - 风险标记：sleep mode 兼容性问题，环境变量解析脆弱性

关联脉络

- PR #41158 Alternative approach using expandable_segments: 该 PR 是 41158 的替代方案，因为 expandable_segments 与 sleep mode 和 NIXLConnector 不兼容。