PR #49287 完整报告

vllm-project/vllm

[XPU][UT] Fix OOM and skip graph case

合并时间：2026-08-18 09:47

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/49287

执行摘要

- 一句话：修复 XPU 单测 OOM 与无 graph 死循环
- 推荐动作：值得 XPU/Intel GPU 相关贡献者精读，重点是 " 把平台特有测试机制泛化为通用原语 " 的设计手法：先有 ROCm 的延迟显存回收等待，再通过重命名与平台分支扩展覆盖 XPU，避免复制一份平行实现。对普通读者而言这是一条小而有代表性的测试基建收口 PR，其中 graph 禁用时的 skip 防护模式也可以在其它依赖 CUDA graph/SYCL graph 的用例中复用。

功能与动机

PR body 明确说明两个目标：1）test_mamba_cache_cg_padding 只能在没有 graph 时运行，否则 UT 会陷入无限循环；2）连续用例导致引擎启动时显存不足，需要复用 ROCm 现有的显存等待机制并为 XPU 提供合适的默认 gpu_memory_utilization。评审中 jikunshang 认为第二点 chaojun-zhang 应已修复，但 mayuyuace 实测最新 vllm 后确认问题仍存在（"I tested the latest vllm, but problem is not fixed yet."），说明该 PR 是对既有平台适配工作的补全。

实现拆解

实现按以下 5 步拆解：

1. 泛化显存等待原语（tests/utils.py）：将 wait_for_rocm_memory_to_settle 重命名为 wait_for_memory_to_settle，早退条件从仅 is_rocm() 扩展为 is_rocm() 或 is_xpu()；同时为 record_gpu_memory_usage_stats 增加 XPU 分支，XPU 上无 nvml/amdsmi，改用 torch.accelerator.get_memory_info（vLLM XPU 平台补丁通过 Level Zero 返回 free/total 字节）。这一步把 ROCm 专属的 " 驱动延迟释放显存 " 处理沉淀为跨平台通用原语。
2. VllmRunner/HfRunner 接入（tests/conftest.py）：HfRunner.__exit__与 VllmRunner.__init__中改用 wait_for_memory_to_settle；VllmRunner.__init__新增 elif current_platform.is_xpu() 分支，在调用方未显式传参时把默认 gpu_memory_utilization 设为 0.9（XPU 运行时 + CCL 上下文需要额外余量），并在构造 LLM 前调用 wait_for_memory_to_settle(threshold_ratio=1.0 - gpu_memory_utilization) 等待显存回落，避免引擎启动内存守卫直接 OOM。
3. 修复 graph 禁用死循环（tests/models/language/generation/test_hybrid.py）：test_mamba_cache_cg_padding 从 vllm.config 导入 CUDAGraphMode，在

CudagraphDispatcher 初始化后检查 cudagraph_mode，若为 CUDAGraphMode.NONE 则 pytest.skip。原因：该用例靠 while 循环把 batch size 扩展到非 CG 捕获尺寸，graph 禁用时 dispatch 永远无法推进，循环会无限执行。

4. 同步存量调用点：test_voxtral_realtime.py、test_colbert.py 将 wait_for_rocm_memory_to_settle 导入与调用替换为 wait_for_memory_to_settle；tests/v1/kv_connector/nixl_integration/run_accuracy_test.sh 中 wait_for_gpu_memory_release 函数同步替换，使 XPU 上也能复用该等待逻辑。
5. 配套与验证：无配置、schema 或部署配套改动；CI 通过 Buildkite（#84025、#84131）验证，其中一次失败 job 由 jikunshang 触发 /ci retry 后通过。

关键文件：

- tests/conftest.py（模块 测试基座；类别 test；类型 test-coverage；符号 _wait_for_rocm_memory_release, _wait_for_memory_release）：变更核心载体：VllmRunner.__init__新增 XPU 分支，默认 gpu_memory_utilization 下调至 0.9 并在构造 LLM 前等待显存回落；_wait_for_rocm_memory_release 泛化为 _wait_for_memory_release，HfRunner.__exit__同步切换。
- tests/utils.py（模块 测试工具；类别 test；类型 test-coverage；符号 wait_for_rocm_memory_to_settle, wait_for_memory_to_settle）：通用原语所在文件：wait_for_rocm_memory_to_settle 重命名为 wait_for_memory_to_settle 并覆盖 XPU；record_gpu_memory_usage_stats 新增 XPU 显存查询分支（torch.accelerator.get_memory_info + Level Zero）。
- tests/models/language/generation/test_hybrid.py（模块 混合模型；类别 test；类型 test-coverage）：修复第一个 UT 问题：test_mamba_cache_cg_padding 在 CUDAGraphMode.NONE 时显式 skip，防止无 graph 场景下 while 扩容循环无限执行。
- tests/models/multimodal/generation/test_voxtral_realtime.py（模块 多模态测试；类别 test；类型 test-coverage）：async_engine fixture 中两处 wait_for_rocm_memory_to_settle 调用同步替换为 wait_for_memory_to_settle，使 XPU 上也能获得启动前的显存等待保护。
- tests/models/language/pooling/test_colbert.py（模块 池化模型；类别 test；类型 test-coverage）：复用泛化后的内存等待函数，替换对旧 ROCm 专属函数的导入与调用。
- tests/v1/kv_connector/nixl_integration/run_accuracy_test.sh（模块 KV 连接器；类别 test；类型 test-coverage）：shell 脚本中的 wait_for_gpu_memory_release 函数同样引用旧函数名，同步替换后 XPU 的 KV connector 集成测试也能使用该等待机制。

关键符号：wait_for_memory_to_settle, record_gpu_memory_usage_stats, _wait_for_memory_release, test_mamba_cache_cg_padding, wait_for_gpu_memory_to_clear

关键源码片段

tests/conftest.py

变更核心载体：VllmRunner.__init__新增 XPU 分支，默认 gpu_memory_utilization 下调至 0.9 并在构造 LLM 前等待显存回落；_wait_for_rocm_memory_release 泛化为

_wait_for_memory_release，HfRunner.__exit__同步切换。

```
# tests/conftest.py —— VllmRunner 构造前的平台差异化显存处理
        from vllm.platforms import current_platform

        if current_platform.is_rocm():
            # V1 启动要求空闲显存 >= total * gpu_memory_utilization。
            # ROCm CI 可能把上一个进程还在"慢慢释放"显存的设备交给测试，
            # 因此构造 LLM 前先等待显存回落，避免启动内存守卫直接 OOM。
            gpu_memory_utilization = kwargs.get(
                "gpu_memory_utilization",
                CacheConfig.gpu_memory_utilization,
            )
            from tests.utils import wait_for_memory_to_settle

            wait_for_memory_to_settle(threshold_ratio=1.0 - gpu_memory_utilization)
        elif current_platform.is_xpu():
            # XPU/oneAPI 运行时会常驻约 1 GiB 上下文（由进程内 HfRunner 模型
            # 累积增长），分布式测试还会在引擎子进程额外分配 CCL 上下文。
            # 默认 0.92 的利用率留给两者的余量太小，因此在调用方未显式指定时
            # 把 XPU 默认降到 0.9。
            if "gpu_memory_utilization" not in kwargs:
                kwargs["gpu_memory_utilization"] = 0.9
            gpu_memory_utilization = kwargs["gpu_memory_utilization"]
            # XPU（Level Zero）同样可能在上一个引擎退出后延迟释放设备内存，
            # 构造 LLM 前先等待显存回落，避免连续用例之间启动即 OOM。
            from tests.utils import wait_for_memory_to_settle

            wait_for_memory_to_settle(threshold_ratio=1.0 - gpu_memory_utilization)
```

tests/utils.py

通用原语所在文件：wait_for_rocm_memory_to_settle 重命名为 wait_for_memory_to_settle 并覆盖 XPU；record_gpu_memory_usage_stats 新增 XPU 显存查询分支（torch.accelerator.get_memory_info + Level Zero）。

```
# tests/utils.py —— 将原先仅针对 ROCm 的等待逻辑泛化为多平台版本

def record_gpu_memory_usage_stats(
    *,
    devices: list[int],
) -> dict[int, tuple[float, float]]:
    """记录每个设备的 (已用, 总量) 显存（单位 GiB），供后续等待逻辑使用。"""
    output: dict[int, tuple[float, float]] = {}
    for device in devices:
        if current_platform.is_rocm():
            # ROCm 通过 amdsmi 查询真实 VRAM 占用
            dev_handle = amdsmi_get_processor_handles()[device]
            mem_info = amdsmi_get_gpu_vram_usage(dev_handle)
            gb_used = mem_info["vram_used"] / 2**10
```

```
            gb_total = mem_info["vram_total"] / 2**10
        elif current_platform.is_xpu():
            # XPU 上无 nvml/amdsmi，改用 torch.accelerator.get_memory_info，
            # vLLM 的 XPU 平台补丁会通过 Level Zero 返回 (free, total) 字节。
            free_b, total_b = torch.accelerator.get_memory_info(device)
            gb_used = (total_b - free_b) / 2**30
            gb_total = total_b / 2**30
        else:
            dev_handle = get_nvml_device_handle(device)
            mem_info = nvmlDeviceGetMemoryInfo(dev_handle)
            gb_used = mem_info.used / 2**30
            gb_total = mem_info.total / 2**30
        output[device] = (gb_used, gb_total)
    return output


def wait_for_memory_to_settle(
    *,
    threshold_ratio: float | dict[int, float] | None = 0.1,
    timeout_s: float = 240,
) -> None:
    """阻塞直到 ROCm 或 XPU 设备显存占用降到 threshold_ratio 以下。

    ROCm 与 XPU（Level Zero）回收显存比 CUDA 更"懒惰"，同一测试进程里
    连续加载模型时，即使调用了 cleanup_dist_env_and_memory，下一个引擎
    启动仍可能因显存未释放而 OOM。该函数给驱动留出释放时间；
    非上述平台时为 no-op。
    """
    if not current_platform.is_rocm() and not current_platform.is_xpu():
        return

    num_gpus = current_platform.device_count()
    if num_gpus == 0:
        return
    if threshold_ratio is None:
        threshold_ratio = 0.1

    wait_for_gpu_memory_to_clear(
        devices=list(range(num_gpus)),
        threshold_ratio=threshold_ratio,
        timeout_s=timeout_s,
        stable_duration_s=2.0,
        poll_interval_s=1.0,
    )
```

tests/models/language/generation/test_hybrid.py

修复第一个 UT 问题：test_mamba_cache_cg_padding 在 CUDAGraphMode.NONE 时显式 skip，防止无 graph 场景下 while 扩容循环无限执行。

```
# tests/models/language/generation/test_hybrid.py —— 防止无 graph 时死循环
    vllm_config = EngineArgs(model=model, trust_remote_code=True).create_engine_config()
    cudagraph_dispatcher = CudagraphDispatcher(vllm_config)
    cudagraph_dispatcher.initialize_cudagraph_keys(
        vllm_config.compilation_config.cudagraph_mode
    )
    # 该用例的目的是验证 mamba cache 按 CG 捕获的 batch size 做了 padding;
    # 若 graph 被禁用（例如 XPU 上 enforce-eager 或 SYCL graph 不可用），
    # dispatch 永远无法推进 batch size，下面的 while 扩容循环会无限执行。
    if cudagraph_dispatcher.cudagraph_mode == CUDAGraphMode.NONE:
        pytest.skip("CUDA/XPU graph is disabled. Please enable it to run this test.")
    while (
        len(example_prompts)
        == cudagraph_dispatcher.dispatch(len(example_prompts))[1].num_tokens
    ):
        example_prompts.append(example_prompts[0])
```

评论区精华

评审中两条核心建议均来自 chaojun-zhang，且都被作者采纳：

- 合并显存等待函数：chaojun-zhang 在 tests/utils.py 提议把 wait_for_xpu_memory_to_settle 与 wait_for_rocm_memory_to_settle 合并为单一函数 wait_for_memory_to_settle，mayuyuace 回复 "Merged."，最终以统一的泛化函数落地。
- 合并 runner 辅助方法：chaojun-zhang 在 tests/conftest.py 提议把 _wait_for_rocm_memory_release 与 _wait_for_xpu_memory_release 合并为 _wait_for_memory_release，同样被采纳。

此外，Issue 评论区围绕 SYCL graph 与 flash attention 的兼容性有实质交锋：zhenwei-intel 指出 "flash attn is supported by sycl graph in pytorch2.13"，mayuyuace 据此验证后将 skip 条件收窄为 " 仅当 graph 禁用时 "，避免在支持 graph 的场景白白跳过用例。

- 合并 wait_for_xpu_memory_to_settle 与 wait_for_rocm_memory_to_settle 为统一函数 (design): mayuyuace 回复 "Merged."，最终以 wait_for_memory_to_settle 统一实现并被全部调用点使用。
- 合并 _wait_for_rocm_memory_release 与 _wait_for_xpu_memory_release 辅助方法 (design): mayuyuace 回复 "Merged."，已合并为 _wait_for_memory_release 并在 __exit__中统一调用。
- SYCL graph 与 flash attention 兼容性及 skip 条件收窄 (correctness): 验证通过后落地为 CUDAGraphMode.NONE 时 pytest.skip 的实现。
- gpu_memory_utilization 问题是否已被 chaojun-zhang 修复 (question): 以本 PR 的 XPU 默认利用率调整与显存等待机制作为最终修复方案。

风险与影响

- 风险：1）测试工具重命名波及面：wait_for_rocm_memory_to_settle 被重命名为 wait_for_memory_to_settle，仓库内所有调用点虽已同步（conftest、voxtral、colbert、

shell 脚本），但外部分支或 fork 中仍引用旧名的测试代码会直接 ImportError；2）XPU 默认显存利用率下调：未显式传参时 0.92 → 0.9 会略微缩小可用 KV cache，依赖大 context 的 XPU 用例理论上可能出现更多 preemption，不过 0.9 仍高于多数用例实际需求，风险较低；3）依赖 torch.accelerator.get_memory_info：record_gpu_memory_usage_stats 的 XPU 分支依赖 vLLM XPU 平台补丁对该接口的填充，若 CI 所用 torch 版本未提供则测试工具会报错；4）graph 死循环场景：skip 只覆盖 CUDAGraphMode.NONE，若 dispatch 在 graph 开启时仍不推进 batch size，while 循环依旧可能死循环，但该风险在 PR 之前即存在，非本次引入。

- 影响：影响范围严格限定在测试侧，无生产代码变更。积极面：XPU CI 的 UT 稳定性显著提升，消除连续用例 OOM 造成的偶发失败与 graph 禁用时的挂起，Intel GPU 平台的持续集成不再被这些平台特有行为阻塞；wait_for_memory_to_settle 成为跨平台（ROCm + XPU）共用的测试内存原语，未来 Level Zero/oneAPI 行为变化只需改动一处。消极面：测试工具函数重命名对使用这些 utils 的贡献者造成轻微 churn；XPU 默认显存利用率下调可能让个别内存敏感测试的行为发生变化。
- 风险标记：测试工具重命名波及面广，XPU 默认显存利用率下调，graph 禁用时死循环风险，仅测试级变更

关联脉络

- PR #53035 [CI][XPU] Skip test_fused_shared_expert.py on XPU: 同为 XPU 测试稳定性处理：在 XPU 上跳过不适用的用例，与本 PR 的 graph 禁用 skip 属于同一维护脉络。
- PR #53004 [ROCm][CI] Speed up test_rocm_aiter_qk_norm_rope_kvcache_fusion: 另一个平台的测试基建调整（ROCm 测试提速），与本 PR 对 ROCm/XPU 测试内存机制的跨平台收口同属平台差异化 CI 治理。
- PR #51585 [ROCm] [Bugfix] Preserve CPU query offsets during capture: 平台相关的 graph capture 修复，与本 PR 中 SYCL graph / CUDAGraphMode 相关测试调整同属平台差异下的 graph 行为处理。