PR #50109 完整报告

vllm-project/vllm

[Kernel][CI] `--jit-monitor-mode error` e2e tests for kernel warmup infra

合并时间：2026-07-31 20:53

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50109

执行摘要

- 一句话：JIT 监控 error 模式 e2e 门禁与 CI 测试拆分
- 推荐动作：值得精读 tests/jit_monitor/test_no_runtime_jit.py 的设计：shape battery 的编译键覆盖策略、spawn 子进程隔离、error 模式下区分 JIT miss 与无关崩溃的异常处理，都是后续 warmup 相关 PR 直接复用的模式。关注其从 skip 到启用的里程碑：一旦 Issue #49349 的 warmup contract 迁移落地，该门禁应尽快取消 skip 并在 JIT_MONITOR_MODELS 中逐步加入 jit-free 模型，才能真正兑现"nightlies 可开启 --jit-monitor-mode error"的目标。

功能与动机

PR body 明确本 PR 是对 Issue #47456（RFC：Design a shared warmup infrastructure for JITs）评论的落地，属于"getting JITs under control with a proper warmup framework"整体工作的一部分。背景是推理期 JIT 编译会带来延迟尖峰，例如监控器在 mask_empty_context_kernel 上抛出的"This causes a latency spike; consider extending warmup to cover this shape/config"。PR 的最终目标是让 --jit-monitor-mode error 可以自信地在 nightlies 中开启；JIT_MONITOR_MODELS 清单按模型逐个扩容，每个新条目对应一个已确保推理期零 JIT 的模型。

实现拆解

本 PR 全部为测试与 CI 配置变更，无生产代码改动，按以下四步拆解：

1. 单测目录重组与全局状态管理：删除 534 行的 tests/test_jit_monitor.py，新增 tests/jit_monitor/ 包。test_hooks.py 承接 CPU 可跑的 mock 单测（mock Triton knobs、cutlass.cute、tilelang 模块），标记为 pytest.mark.cpu_test，并新增 test_cutedsl_subscripted_compile_is_monitored，覆盖 flashinfer >= 0.6.14 的 cute.compileoptions 订阅式调用形态；test_hooks_gpu.py 新增真实 GPU + 真实 Triton 内核的 e2e 单测；conftest.py 提供 autouse fixture _reset_monitor，统一重置 jit_monitor._active/_mode/_verbose/_cutedsl_hook_installed/_tilelang_hook_installed/_tilelang_jitimpl_compile_depth 等进程级全局状态，避免测试间互相污染。
2. 新增 e2e 门禁 test_no_runtime_jit.py：定义 JitModel dataclass 与 JIT_MONITOR_MODELS 清单（Qwen3-0.6B、DeepSeek-V2-Lite-Chat、DeepSeek-V3、granite-4.0-tiny-preview、两个带 speculative draft 的随机权重 DeepSeek 变体）。_run_shape_battery 用 Token-id prompt 触达不同编译键：greedy 单序列多步 decode、

混合长度批量 prefill、top_k/top_p/min_p 各自特殊化的 Triton sampler、同一步内异构 SamplingParams（sampler warmup 缺失最常在此暴露）。can_run_without_jit 在 spawn 子进程中以 load_format="dummy"、hf_overrides=dummy_hf_overrides、jit_monitor_mode="error" 启动 LLM；选择 spawn 而非 fork 是因为 monitor hooks 是进程级全局状态且 error 模式无法拆卸，fork 已初始化 CUDA 的 pytest 进程会毒化子进程。异常处理上，只有消息含 "during inference" 才判为 JIT miss 并 pytest.fail，其余异常原样抛出。当前 pytestmark = pytest.mark.skip（warmup coverage 未完备，Issue #49349 跟踪），待 warmup contract 迁移落地后取消。

3. CI 配置接入：新增 .buildkite/test_areas/jit_monitor.yaml，定义 H200 单卡独立 job“No Runtime JITs e2e tests”，timeout 45 分钟，source_file_dependencies 精确指向 vllm/utils/jit_monitor.py、vllm/v1/worker/gpu_worker.py、vllm/model_executor/warmup/、vllm/config/observability.py 等，并以 pytest --timeout=900 --timeout-method=thread 作为每测试 watchdog，防止引擎 /CUDA 初始化卡死拖到整个 job 超时。engine.yaml、engine_intel.yaml、test-amd.yaml 同步将 test_jit_monitor.py 的引用替换为 jit_monitor/test_hooks.py（Intel 只跑 CPU 单测），AMD job 额外运行 test_hooks_gpu.py。
4. 演进机制：JIT_MONITOR_MODELS 是随 warmup 覆盖完善而增删的清单——每加一行代表一个模型已磨平推理期 JIT；最终目标是在 nightlies 中全局开启 --jit-monitor-mode error。

关键文件：

- tests/jit_monitor/test_no_runtime_jit.py（模块 JIT 门禁；类别 test；类型 test-coverage；符号 JitModel, _run_shape_battery, can_run_without_jit, test_no_runtime_jit）：PR 的核心新增：以 --jit-monitor-mode error 启动 JIT-heavy 模型集合并跑 shape battery，任何推理期 JIT 编译都判失败；是 RFC #47456 落地后的关键验收手段。
- tests/test_jit_monitor.py（模块 监控单测；类别 test；类型 deletion；符号 _reset_monitor, _make_fake_knobs, _fake_cute_import_modules, _fake_cute_compile）：534 行旧单测整体删除，内容拆分进 tests/jit_monitor/ 下的 test_hooks.py 与 test_hooks_gpu.py，conftest.py 统一管理全局状态重置。
- tests/jit_monitor/test_hooks.py（模块 监控单测；类别 test；类型 test-coverage；符号 _make_fake_knobs, _fake_cute_import_modules, _fake_cute_compile, _fake_tilelang_import_modules）：CPU 可跑的 monitor 单测（mock Triton/CuTeDSL/TileLang），新增对 flashinfer >= 0.6.14 的 cute.compileoptions 订阅式调用形态的监控断言；被 AMD/Intel/ 通用 engine job 引用。
- tests/jit_monitor/test_hooks_gpu.py（模块 GPU 监控；类别 test；类型 test-coverage；符号 _add_kernel, _run_add_kernel, test_no_warning_on_cached_shape, test_warning_on_new_constexpr）：真实 GPU + 真实 Triton 内核验证 monitor 行为：缓存 shape 不告警、新 constexpr 触发 warning_once、verbose 下新指针对齐逐次告警，避免 mock 单测与真实 Triton hook 行为脱节。
- tests/jit_monitor/conftest.py（模块 测试夹具；类别 test；类型 test-coverage；符号 _reset_monitor）：autouse fixture 重置 jit_monitor 的进程级全局状态（_active/_mode/_verbose/ 各 hook 安装标记），确保测试间不互相污染。

- .buildkite/test_areas/jit_monitor.yaml（模块 CI 配置；类别 config；类型 configuration）：新增独立 Buildkite job（H200），专门跑 No Runtime JITs e2e，带 source_file_dependencies 精确触发范围与 900 秒测试级 watchdog。
- .buildkite/test_areas/engine.yaml（模块 CI 配置；类别 config；类型 configuration）：通用 engine job 的文件依赖与 pytest 命令从 test_jit_monitor.py 改为 jit_monitor/test_hooks.py 与 test_hooks_gpu.py。
- .buildkite/intel_jobs/engine_intel.yaml（模块 CI 配置；类别 config；类型 configuration）：Intel CI 同步更新：只引用 CPU 可跑的 jit_monitor/test_hooks.py（Intel 无 GPU hook 测试）。
- .buildkite/test-amd.yaml（模块 CI 配置；类别 config；类型 configuration）：AMD CI 同步更新为 jit_monitor/test_hooks.py 与 test_hooks_gpu.py，保持平台一致性。

关键符号：_reset_monitor, _make_fake_knobs, _fake_cute_import_modules, _fake_tilelang_import_modules, _patch_jit_modules, _triton_hook_kwargs, _add_kernel, _run_add_kernel, JitModel, _run_shape_battery, can_run_without_jit, test_no_runtime_jit

关键源码片段

tests/jit_monitor/test_no_runtime_jit.py

PR 的核心新增：以 --jit-monitor-mode error 启动 JIT-heavy 模型集合并跑 shape battery，任何推理期 JIT 编译都判失败；是 RFC #47456 落地后的关键验收手段。

```
@dataclass(frozen=True)
class JitModel:
    """描述一个待验证的模型组合：主模型 + 可选的 speculative draft 模型。"""
    model: str
    draft: str | None = None
    trust_remote_code: bool = False


def _run_shape_battery(llm: LLM) -> None:
    """用一组多样请求触达不同编译键，让缺失的 warmup key 暴露出来。

    全部使用 Token-id prompt，保证 shape 精确且不依赖 tokenizer；
    dummy 权重下生成内容无意义，断言只关心推理阶段是否发生了 JIT 编译。
    """
    short = TokensPrompt(prompt_token_ids=[1, 2, 3, 4])
    medium = TokensPrompt(prompt_token_ids=list(range(1, 33)))
    long = TokensPrompt(prompt_token_ids=list(range(1, 129)))

    # Greedy 单序列多步 decode：覆盖 prefill + 自回归 decode + greedy sampler。
    llm.generate(medium, SamplingParams(temperature=0.0, max_tokens=16))
    # 混合长度批量 prefill：varlen prefill + padded decode 各走不同编译键。
    llm.generate([short, medium, long], SamplingParams(temperature=0.0, max_tokens=8))
    # Triton sampler 的 top_k / top_p / min_p 各自 specialization，逐一触发。
    for sampling_params in (
        SamplingParams(temperature=0.8, top_k=20, max_tokens=8, seed=0),
```

```
        SamplingParams(temperature=0.8, top_p=0.9, max_tokens=8, seed=0),
        SamplingParams(temperature=0.8, min_p=0.1, max_tokens=8, seed=0),
        SamplingParams(temperature=0.8, top_k=20, top_p=0.9, min_p=0.1, max_tokens=8, seed=
        0),
    ):
        llm.generate(medium, sampling_params)
    # 同一步内异构 SamplingParams：sampler warmup 缺失最常在这里暴露。
    llm.generate(
        [medium] * 4,
        [
            SamplingParams(temperature=0.0, max_tokens=8),
            SamplingParams(temperature=0.8, top_k=20, max_tokens=8, seed=0),
            SamplingParams(temperature=0.8, top_p=0.9, max_tokens=8, seed=0),
            SamplingParams(temperature=0.8, min_p=0.1, max_tokens=8, seed=0),
        ],
    )


@create_new_process_for_each_test("spawn")
def can_run_without_jit(spec: JitModel):
    """以 error 模式启动模型并跑 shape battery；任何推理期 JIT 都会抛错。

    必须用 spawn 而非 fork：monitor 的 hook 是进程级全局状态，error 模式
    一旦武装无法拆卸；fork 已初始化 CUDA 的 pytest 进程会毒化子进程。
    """
    llm = LLM(
        spec.model,
        trust_remote_code=spec.trust_remote_code,
        max_model_len=2048,
        max_num_seqs=8,
        gpu_memory_utilization=0.80,
        load_format="dummy", # 随机权重：省下载时间，只验证 JIT 路径
        hf_overrides=dummy_hf_overrides,
        enforce_eager=False, # 保留 CUDA graph：graph 内 decode 不再触发 Python hook
        jit_monitor_mode="error", # 推理期 JIT 编译直接 RuntimeError
        speculative_config={"model": spec.draft, "num_speculative_tokens": 2}
        if spec.draft
        else None,
    )
    try:
        _run_shape_battery(llm)
    except Exception as e:
        # 只有消息含 "during inference" 才是 JIT miss，其余异常原样抛出。
        if "during inference" in str(e):
            pytest.fail(
                f"{spec.model}: post-warmup JIT compilation detected - a warmup "
                f"key is missing for a shape in the battery.\n{e}"
            )
        raise
```

tests/jit_monitor/test_hooks_gpu.py

真实 GPU + 真实 Triton 内核验证 monitor 行为：缓存 shape 不告警、新 constexpr 触发 warning_once、verbose 下新指针对齐逐次告警，避免 mock 单测与真实 Triton hook 行为脱节。

```
@triton.jit
def _add_kernel(x_ptr, y_ptr, out_ptr, n, BLOCK: tl.constexpr):
    """一个真实 Triton 内核：BLOCK 作为 constexpr 触发不同编译键。"""
    pid = tl.program_id(0)
    offs = pid * BLOCK + tl.arange(0, BLOCK)
    mask = offs < n
    x = tl.load(x_ptr + offs, mask=mask)
    y = tl.load(y_ptr + offs, mask=mask)
    tl.store(out_ptr + offs, x + y, mask=mask)


def _run_add_kernel(n: int, block: int = 256, offset: int = 0) -> None:
    """启动一次内核；offset 改变指针对齐，验证 verbose 模式的告警行为。"""
    x = torch.randn(n + offset, device="cuda")[offset:] # 改变 base pointer 对齐
    y = torch.randn(n, device="cuda")
    out = torch.empty(n, device="cuda")
    grid = ((n + block - 1) // block,)
    _add_kernel[grid](x, y, out, n, BLOCK=block)
    torch.accelerator.synchronize()


def test_no_warning_on_cached_shape():
    """预热过的 shape 再次运行不应触发告警。"""
    _run_add_kernel(1024)
    jit_monitor.activate()
    with mock.patch.object(jit_monitor.logger, "warning_once") as w:
        _run_add_kernel(1024)
    w.assert_not_called()


def test_warning_on_new_constexpr():
    """BLOCK 改变（tl.constexpr）会强制重编译，应触发 warning_once。"""
    _run_add_kernel(1024, block=256)
    jit_monitor.activate()
    with mock.patch.object(jit_monitor.logger, "warning_once") as w:
        _run_add_kernel(1024, block=512)
    w.assert_called()
    msg = w.call_args[0][0] % w.call_args[0][1:]
    assert "_add_kernel" in msg
```

评论区精华

本 PR 的 review 讨论极少，核心信息主要来自作者自注与代码注释：

- NickLucche 在 .buildkite/test_areas/jit_monitor.yaml 第 8 行自评：“this timeout is too high, will put actual values once we start adding models”——说明该 job 当前是骨架，超时值等真实模型接入后再校准。
- PR body 明确说明这些模型目前仍会在推理期触发 JIT（如 mask_empty_context_kernel），“tests are currently expected to fail... so I am skipping them for now”，代码中 pytestmark 的 skip reason 指向 Issue #49349。
- 代码注释解释子进程 spawn 隔离的必要性：monitor hook 是进程级全局状态，error 模式一旦武装不会解除；fork 一个已初始化 CUDA 的进程会毒化子进程。
- claude[bot] 的自动回复因 PR 来自 fork 而禁用自动 review，无实质内容；robertgshaw2-redhat 最终批准合入。
- 新 CI job 的 45 分钟超时是否过高 (design)：暂时保留 45 分钟，待 JIT_MONITOR_MODELS 加入具体模型后测量并收紧。
- e2e 测试整体 skip 的决策与回归条件 (testing)：合入时保持 skip；待 warmup contract 迁移（Issue #49349）落地且模型逐个 jit-free 后取消 skip 并扩充 JIT_MONITOR_MODELS。
- 子进程 spawn 隔离的必要性 (design)：采用 @create_new_process_for_each_test("spawn") 作为固定测试模式，保证每个模型的测试环境干净且 error 模式不被前一个测试污染。

风险与影响

- 风险：本 PR 无生产代码变更，主要风险集中在 CI 基建层面：

1. 门禁尚未生效：test_no_runtime_jit.py 整体 pytest.mark.skip，CI 通过不代表模型无推理期 JIT，存在“绿 CI 假象”；需在 Issue #49349 完成后逐模型取消 skip 才能真正发挥防护作用。
2. 覆盖范围有限：e2e 仅在 H200 35GB 单卡运行，多卡 TP、AMD（ROCm/AITER）、Intel（XPU）等后端的 JIT 路径没有 e2e 验证；AMD/Intel job 只跑 mock 单测与真实 GPU hook 单测，不覆盖模型级行为。
3. dummy 权重偏差：测试使用 load_format="dummy" 与随机权重模型（luccafong/deepseek_mtp_main_random 等），真实 checkpoint 的结构差异或 shape 分布可能引入不同编译键，存在漏网 JIT。
4. CI 资源成本：每个模型一个 spawn 子进程 + CUDA 初始化，6 个模型在 45 分钟窗口内完成压力较大，超时值尚未实测校准；若后续模型增多，job 时长需重新评估。 - 影响：对用户无运行时影响（纯测试与 CI 变更）。对团队而言，这是 JIT warmup 治理的可执行度量起点：后续任何触碰 vllm/utils/jit_monitor.py、vllm/v1/worker/gpu_worker.py、vllm/model_executor/warmup/ 或观测配置的 PR 都会触发新的 JIT Monitor CI job；test_no_runtime_jit.py 将成为 warmup contract 迁移（RFC #47456、Issue #49349）的验收门禁，也为开发者提供了“如何验证一个模型已无推理期 JIT”的标准化做法。对 CI 系统的影响是新增一个 H200 job 与部分 job 的测试路径调整，平台一致性通过 engine.yaml / engine_intel.yaml / test-amd.yaml 同步修改得到保障。 - 风险标记：新 CI 门禁当前整体 skip，尚未提供实际防护，仅覆盖 H200 单卡，AMD/Intel 与其他规格 GPU 无 e2e 覆盖，dummy 随机权重与真实 checkpoint 行为存在偏差，每模型 spawn 子进程，CI 时长与 GPU 资源成本偏高

关联脉络

- PR #47451 Initial implementation of the JIT warmup contract (referenced in RFC #47456): RFC #47456 的 issue body 明确提到 PR #47451 是 shared warmup infrastructure 的首次实现（Triton 与 CuTeDSL 路径示例）；本 PR 建立的 --jit-monitor-mode error e2e 门禁正是为该类 warmup 改动提供验收与回归保护。