PR #52417 完整报告

vllm-project/vllm

[CI/Build] Avoid duplicate runner startup for multimodal test

合并时间：2026-08-17 06:17

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52417

执行摘要

- 一句话：合并多模态尺寸用例避免重复启动 runner，用例 259 减至 108
- 推荐动作：值得快速浏览，适合作为测试基础设施优化的参考案例。核心看点是把 pytest 参数化展开与运行时批处理解耦：用“用例内多批次”替代“用例级参数化”换取启动开销下降，同时清晰暴露了聚合后的失败定位粒度权衡。对编写多模态或多配置测试的团队有参考价值；由于不涉及产品代码，无需精读推理实现。

功能与动机

PR body 明确说明：现有 mm 测试为每个 size factor 单独启动一次 server，CI 时间大量浪费在启动上；本 PR 将多个尺寸的 multimodal 请求融合进同一个测试，减少重复启动。首个提交信息 "avoid duplicate weights loading" 进一步表明合并的核心收益在于避免重复加载模型权重和重复初始化运行环境。

实现拆解

1. 数据契约调整：tests/models/multimodal/generation/vlm_utils/types.py 中 ExpandableVLMTestArgs.size_wrapper: ImageSizeWrapper | None 改为 size_wrappers: tuple[ImageSizeWrapper, ...]，语义从“每个用例一个尺寸包装器”变为“每个用例一组尺寸批次”，并同步更新 VLMTestInfo.image_size_factors 的注释，明确“每个内部可迭代对象定义一个请求批次，所有批次在同一模型实例上运行”。
2. 参数化逻辑重构：tests/models/multimodal/generation/vlm_utils/case_filtering.py 的 get_model_type_cases 不再把 wrapped_sizes 放入 itertools.product 逐项展开，而是整体打包为 iter_kwargs["size_wrappers"] = (wrapped_sizes,)，使每个模型 / 配置组合只生成一个用例；同时新增空批次保护（if not wrapped_sizes: return []）。get_parametrized_options 与 get_wrapped_test_sizes 的 docstring 同步更新。
3. 执行入口聚合：tests/models/multimodal/generation/vlm_utils/runners.py 的 run_single_image_test、run_multi_image_test、run_video_test、run_embedding_test 从单个 size_wrapper 构建输入改为遍历 test_case.size_wrappers，并用 itertools.chain.from_iterable 合并所有批次输入后一次性交给 core.run_test；embedding 用例需要先把 (inputs, vllm_embeddings) 对按批次构建，再分别平铺合并，保证两类输入数量对齐。
4. 用例模板迁移：tests/models/multimodal/generation/test_qwen2_vl.py 与 test_phi4mm.py 删除三处 @pytest.mark.parametrize("size_factors", ...) 装饰器，改为模

块级常量 IMAGE_SIZE_FACTOR_GROUPS / VIDEO_SIZE_FACTOR_GROUPS，在测试函数体内用 for size_factors in ... 循环生成多批输入，实现相同覆盖的同时把 pytest 收集用例从 259 个降到 108 个。

5. 测试与 CI 配套：无新增测试文件；验证方式为运行 pytest -s -v tests/models/multimodal/generation/test_common.py。CI 经多次触发（Buildkite #84007、#84078、#84115）全部通过后由维护者 DarkLight1337 批准合入，期间两次合并 main 分支以保持测试基线同步。

关键文件：

- tests/models/multimodal/generation/vlm_utils/runners.py（模块 测试入口；类别 test；类型 test-coverage；符号 run_single_image_test, run_multi_image_test, run_embedding_test, run_video_test）：本次改动的执行入口：四个 run_*_test 函数从单个 size_wrapper 改为遍历 size_wrappers 并用 itertools.chain 平铺输入，把多尺寸批次合并进一次 core.run_test，是"避免重复启动"的直接落点，改动量最大（+43/-16）。
- tests/models/multimodal/generation/test_qwen2_vl.py（模块 视觉用例；类别 test；类型 test-coverage；符号 test_qwen2_vl_image_embeddings_input, test_qwen2_vl_multiple_image_embeddings_input, test_qwen2_vl_video_embeddings_input）：展示参数化迁移的标准模板：删除三处 size_factors 参数化装饰器，改为模块级 IMAGE_SIZE_FACTOR_GROUPS / VIDEO_SIZE_FACTOR_GROUPS 常量在函数体内循环构造多批输入，是用例数从 259 降至 108 的主要贡献者之一。
- tests/models/multimodal/generation/test_phi4mm.py（模块 多模态用例；类别 test；类型 test-coverage；符号 test_models, test_multi_images_models）：与 qwen2_vl 相同的迁移样例，用于 Phi4MM 模型，且包含 large_gpu_test(min_gb=48) 标记，验证该聚合模式同样适用于大显存、多 tile 用例。
- tests/models/multimodal/generation/vlm_utils/types.py（模块 类型定义；类别 test；类型 test-coverage；符号 VLMTestInfo, ExpandableVLMTestArgs）：数据契约变更点：ExpandableVLMTestArgs.size_wrapper 字段升级为 size_wrappers 元组，语义从"每用例一个尺寸"变为"每用例一批尺寸"，注释同步说明所有批次共享同一模型实例，是整套改动的类型基础。
- tests/models/multimodal/generation/vlm_utils/case_filtering.py（模块 用例筛选；类别 test；类型 test-coverage；符号 get_model_type_cases, get_parametrized_options, get_wrapped_test_sizes）：参数化逻辑核心：get_model_type_cases 把尺寸批次整体作为单一参数（而非 product 展开），并新增空批次短路返回（return []），是"用例数下降"的根本原因；get_parametrized_options 与 get_wrapped_test_sizes 的文档说明同步更新。

关键符号：run_single_image_test, run_multi_image_test, run_embedding_test, run_video_test, get_model_type_cases, get_parametrized_options, get_wrapped_test_sizes

关键源码片段

tests/models/multimodal/generation/vlm_utils/runners.py

本次改动的执行入口：四个 run_*_test 函数从单个 size_wrapper 改为遍历 size_wrappers 并用 itertools.chain 平铺输入，把多尺寸批次合并进一次 core.run_test，是“避免重复启动”的直接落点，改动量最大（+43/-16）。

```
def run_single_image_test(
    *,
    tmp_path: PosixPath,
    model_test_info: VLMTestInfo,
    test_case: ExpandableVLMTestArgs,
    hf_runner: type[HfRunner],
    vllm_runner: type[VllmRunner],
    image_assets: ImageTestAssets,
):
    # 新方案把同一模型的全部尺寸批次聚合进一个用例，
    # 避免每个 size factor 各自拉起一次 vLLM server（权重加载 + CUDA 初始化开销最大）
    assert test_case.size_wrappers

    # 对每个 size_wrapper 构建一批输入，再用 chain.from_iterable 平铺成一份大列表，
    # 这样 core.run_test 只需启动一次 runner 即可覆盖全部尺寸
    inputs = list(
        itertools.chain.from_iterable(
            builders.build_single_image_inputs_from_test_info(
                model_test_info, image_assets, size_wrapper, tmp_path
            )
            for size_wrapper in test_case.size_wrappers
        )
    )

    core.run_test(
        hf_runner=hf_runner,
        vllm_runner=vllm_runner,
        inputs=inputs,
        model=test_case.model,
        dtype=test_case.dtype,
        max_tokens=test_case.max_tokens,
        num_logprobs=test_case.num_logprobs,
        limit_mm_per_prompt={"image": 1},
        distributed_executor_backend=test_case.distributed_executor_backend,
        **model_test_info.get_non_parametrized_runner_kwargs(),
    )


def run_embedding_test(
    *,
    model_test_info: VLMTestInfo,
    test_case: ExpandableVLMTestArgs,
    hf_runner: type[HfRunner],
    vllm_runner: type[VllmRunner],
    image_assets: ImageTestAssets,
```

```
):
    assert test_case.size_wrappers

    # embedding 用例需要同时收集原始输入与预计算好的 vllm_embeddings，
    # 这里先按 size_wrapper 逐个构建 (inputs, embeddings) 对，再分别平铺合并，
    # 保证两路输入的数量与顺序严格对齐
    inputs_and_embeddings = [
        builders.build_embedding_inputs_from_test_info(
            model_test_info, image_assets, size_wrapper
        )
        for size_wrapper in test_case.size_wrappers
    ]
    inputs = list(
        itertools.chain.from_iterable(inputs for inputs, _ in inputs_and_embeddings)
    )
    vllm_embeddings = list(
        itertools.chain.from_iterable(
            embeddings for _, embeddings in inputs_and_embeddings
        )
    )

    core.run_test(
        hf_runner=hf_runner,
        vllm_runner=vllm_runner,
        inputs=inputs,
        model=test_case.model,
        dtype=test_case.dtype,
        max_tokens=test_case.max_tokens,
        num_logprobs=test_case.num_logprobs,
        limit_mm_per_prompt={"image": 1},
        vllm_embeddings=vllm_embeddings,
        distributed_executor_backend=test_case.distributed_executor_backend,
        **model_test_info.get_non_parametrized_runner_kwargs(),
    )
```

tests/models/multimodal/generation/vlm_utils/case_filtering.py

参数化逻辑核心：get_model_type_cases 把尺寸批次整体作为单一参数（而非 product 展开），并新增空批次短路返回（return []），是“用例数下降”的根本原因；
get_parametrized_options 与 get_wrapped_test_sizes 的文档说明同步更新。

```
def get_model_type_cases(
    model_type: str,
    test_info: VLMTestInfo,
    test_type: VLMTestType,
):
    # 把标量配置包成可迭代对象，便于统一走 product 展开
    ensure_wrapped = lambda e: e if isinstance(e, (list, tuple)) else (e,)

    # 该函数近似嵌套多个 mark.parametrize，但改为程序化展开，
```

```
    # 从而支持按模型覆盖配置，同时每个用例仍可单独执行
    iter_kwargs = OrderedDict(
        [
            ("model", ensure_wrapped(test_info.models)),
            ("max_tokens", ensure_wrapped(test_info.max_tokens)),
            ("num_logprobs", ensure_wrapped(test_info.num_logprobs)),
            ("dtype", ensure_wrapped(test_info.dtype)),
            (
                "distributed_executor_backend",
                ensure_wrapped(test_info.distributed_executor_backend),
            ),
        ]
    )

    # 视频用例额外展开帧数相关配置
    if test_type == VLMTestType.VIDEO:
        iter_kwargs["num_video_frames"] = ensure_wrapped(test_info.num_video_frames)
        iter_kwargs["needs_video_metadata"] = ensure_wrapped(
            test_info.needs_video_metadata
        )

    # 关键变化：尺寸批次不再参与 product 展开，而是整体作为单个参数传入，
    # 让所有尺寸批次在用例内部共享同一个 runner 实例，避免重复启动 server
    if test_type not in (
        VLMTestType.CUSTOM_INPUTS,
        VLMTestType.AUDIO,
    ):
        wrapped_sizes = get_wrapped_test_sizes(test_info, test_type)
        if wrapped_sizes is None:
            raise ValueError(f"Sizes must be set for test type {test_type}")
        # 尺寸配置为空时直接产出 0 个用例，避免无意义的空跑
        if not wrapped_sizes:
            return []
        # 外层再包一层 tuple，让 product 只产生一个组合
        iter_kwargs["size_wrappers"] = (wrapped_sizes,)

    # 其余配置仍按 product 展开，每个组合生成一个 pytest.param
    return [
        pytest.param(
            model_type,
            ExpandableVLMTestArgs(**{k: v for k, v in zip(iter_kwargs.keys(), case)}),
            marks=test_info.marks if test_info.marks is not None else [],
        )
        for case in list(itertools.product(*iter_kwargs.values()))
    ]
```

评论区精华

该 PR 来自 fork，claude[bot] 的自动 review 被禁用，并提示维护者可通过 @claude review 触发一次性评审，但最终未执行。仓库维护者 DarkLight1337 直接 APPROVED，review comments 为空，Issue 侧仅有 CI 触发命令（/ci run、/ci retry）与 Buildkite 构建记录，不存在设计取舍争论。值得注意的是提交历史包含两次 main 分支合并提交，说明 PR 生命周期较长，期间持续同步主分支以保持测试基线与仓库一致。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 失败隔离性下降：多尺寸批次聚合进同一用例后，若其中某批次失败会导致整个用例失败，定位粒度从“哪个 size factor”变为“整用例”，排查问题需要重新在用例内拆分。
  2. 共享 runner 状态：此前每个 size factor 独立进程启动，现在所有批次共享同一 runner 实例，批次间进程内状态（如 CUDA 缓存、模型实例）相互可见，个别依赖隔离的失败模式可能被掩盖或扩大。
  3. embedding 内存峰值：run_embedding_test 现在一次性为所有尺寸构建并传入 embeddings 列表，输入规模叠加后内存占用略有上升，在资源受限的 CI 机器上需关注。
  4. 风险整体可控：改动全部位于 tests/models/multimodal/generation/ 测试目录，不涉及产品代码、配置或部署路径，回归风险低。 - 影响：对最终用户无影响（纯测试代码变更）。对 CI 团队而言，多模态生成测试收集用例从 259 个降至 108 个，每组尺寸批次共享一次 server 启动，显著缩短 CI 中重复权重加载与 CUDA 初始化的时间开销；对后续测试开发者而言，新增多模态模型测试需遵循新模式——用模块级常量定义尺寸分组，在用例体内循环构造多批输入，由 runner 聚合执行。该模式也可推广到其他存在“多配置 × 启动开销大”矛盾的测试类型。 - 风险标记：用例聚合降低失败隔离性，失败定位粒度变粗，embedding 批构建内存上升

关联脉络

- PR #52256 [ROCm][CI] Enable ViT CUDA graph tests on AMD gfx950 GPUs: 同属 tests/models/multimodal/generation/ 目录的 CI 测试基建调整，与该 PR 一样在优化多模态测试的 CI 执行方式，体现该目录的持续演进脉络。
- PR #52441 [Bugfix][Multimodal] Keep Gemma 4 video frame counts on CPU: 同样修改 tests/models/multimodal/generation/ 下的测试文件，与该 PR 共享同一套 runner/ 输入构造测试基建，后续新增多模态用例都受本 PR 聚合模式影响。