# PR #50020 完整报告

vllm-project/vllm

[Bugfix][MRV2] Support encoder timing stats in model runner V2

合并时间：2026-08-12 00:00

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50020

## 执行摘要

- 一句话：为 MRV2 补齐 encoder 计时统计，修复 mm-processor bench 崩溃
- 推荐动作：值得精读，推荐关注两点：一是 review 中关于“保持执行函数签名纯净、用上下文管理器承载横切逻辑”的设计取舍，这是可迁移到其他计费 / 观测功能的通用模式；二是数据契约从单一 runner 文件迁移到共享 utils.py 的做法，能有效避免 V1/V2 双份定义漂移。若你在维护多模态模型或基准测试工具，建议特别细读 encoder_runner.py 的 timed_encoder_operation 实现。

## 功能与动机

PR body 明确指出崩溃根因：vllm bench mm-processor crashes on model runner V2: get_encoder_timing_stats only exists in V1。vllm bench mm-processor 依赖 runner 暴露 encoder 计时统计，而 MRV2 的 GPUModelRunner 尚未移植该方法，导致直接 AttributeError。作者还说明该 PR 不与任何在开的 MRV2 encoder timing 改动重复，属于 V1 功能向 V2 对齐的必要补丁。

## 实现拆解

该 PR 以“数据契约共享 + 计时下沉 + 入口转发”三步完成 MRV2 的 encoder 计时统计支持：

1. 共享统计数据结构：把 EncoderTimingStats dataclass 从 vllm/v1/worker/gpu_model_runner.py（MRV1）整体迁移到 vllm/v1/worker/utils.py，同时删除 MRV1 文件中的本地定义并从 dataclasses 导入中移除 dataclass。这样 V1、V2 与 EncoderRunner 引用同一份定义，避免双份数据类在演化中走样，这是改动量最大的“数据契约”调整。
2. EncoderRunner 内置计时能力：在 vllm/v1/worker/gpu/mm/encoder_runner.py 中为构造函数新增 enable_timing: bool = False 参数，新增 encoder_timing_registry: dict[str, EncoderTimingStats] 与 _timing_lock 线程锁；新增 timed_encoder_operation() 上下文管理器（先 torch.accelerator.synchronize() 再用 time.perf_counter() 计时，结束时把总耗时均摊到该批每个请求并加锁累计），以及 get_encoder_timing_stats()（加锁导出全部请求统计后清空 registry，保证每次统计区间独立）。该设计直接复刻 V1 中已有模式，也回应了 review 中对 API 纯净性的要求。
3. 调用点接线：在 vllm/v1/worker/gpu/model_states/interface.py 的 ModelState.__init__ 中从 vllm_config.observability_config.enable_mm_processor_stats 推导 enable_timing

并传入 EncoderRunner；ModelState.execute_mm_encoder() 用 timed_encoder_operation(scheduled_encoder_inputs.keys()) 包裹真实 encoder 调用。vllm/v1/worker/gpu/model_states/encoder_decoder.py 的 get_mm_embeddings() 编码路径也做了同样的包裹，保证 encoder-decoder 模型同样被统计。

4. MRV2 入口补方法：在 vllm/v1/worker/gpu/model_runner.py 的 GPUModelRunner 上新增 get_encoder_timing_stats()，通过 getattr(self.model_state, "encoder_runner", None) 取到 EncoderRunner 后转发；无多模态模型（encoder-only 等）时安全返回空 dict，兼容 bench 工具在任意模型上的调用。
5. 测试与验证配套：tests/v1/worker/test_encoder_runner.py 新增 test_encoder_timing_stats_registry，覆盖 registry 的累计（同一请求两次调用后 num_encoder_calls == 2）与导出后清空语义。PR body 报告该测试文件 8 passed，并在 Modal L4 上以 MRV2 默认路径跑通 vllm bench mm-processor --model Qwen/Qwen2.5-VL-3B-Instruct --dataset-name random-mm --num-prompts 10，统计到 encoder_forward_ms 均值 138.99。

关键文件：

- vllm/v1/worker/gpu/mm/encoder_runner.py（模块 编码器；类别 source；类型 core-logic；符号 timed_encoder_operation, get_encoder_timing_stats）：核心改动：为 EncoderRunner 增加 enable_timing 开关、encoder_timing_registry 注册表与 _timing_lock 锁，新增 timed_encoder_operation 上下文管理器和 get_encoder_timing_stats 导出方法，是 MRV2 encoder 计时的真正实现载体。
- vllm/v1/worker/utils.py（模块 公共层；类别 source；类型 core-logic；符号 EncoderTimingStats, to_dict）：EncoderTimingStats 数据类迁入此共享模块，成为 V1、V2 与 EncoderRunner 的统一统计契约，避免双份定义漂移。
- vllm/v1/worker/gpu/model_runner.py（模块 MRV2 执行器；类别 source；类型 data-contract；符号 get_encoder_timing_stats）：MRV2 入口：新增 get_encoder_timing_stats 方法，从 model_state 取 encoder_runner 转发，修复 vllm bench mm-processor 在 V2 下的 AttributeError。
- vllm/v1/worker/gpu/model_states/interface.py（模块 模型状态；类别 source；类型 data-contract）：接线点：在 ModelState 构造 EncoderRunner 时从 observability_config.enable_mm_processor_stats 传入 enable_timing，并在 execute_mm_encoder 中包裹计时上下文。
- vllm/v1/worker/gpu_model_runner.py（模块 MRV1 执行器；类别 source；类型 data-contract；符号 EncoderTimingStats, to_dict）：MRV1 侧对应调整：移除本地 EncoderTimingStats 定义，改为从 worker/utils.py 导入，并清理 dataclass 导入，是数据契约迁移的配合项。
- vllm/v1/worker/gpu/model_states/encoder_decoder.py（模块 模型状态；类别 source；类型 data-contract）：encoder-decoder 分支同样接入计时：get_mm_embeddings 编码路径用 timed_encoder_operation 包裹，保证编解码模型也被统计。
- tests/v1/worker/test_encoder_runner.py（模块 编码器；类别 test；类型 test-coverage；符号 test_encoder_timing_stats_registry）：新增 test_encoder_timing_stats_registry 测试，覆盖 registry 累计与导出清空语义，是本次核心逻辑的回归保障。

关键符号：EncoderRunner.timed_encoder_operation，EncoderRunner.get_encoder_timing_stats，GPUModelRunner.get_encoder_timing_stats，EncoderTimingStats.to_dict

## 关键源码片段

### vllm/v1/worker/gpu/mm/encoder_runner.py

核心改动：为 EncoderRunner 增加 enable_timing 开关、encoder_timing_registry 注册表与 _timing_lock 锁，新增 timed_encoder_operation 上下文管理器和 get_encoder_timing_stats 导出方法，是 MRV2 encoder 计时的真正实现载体。

```
# vllm/v1/worker/gpu/mm/encoder_runner.py（整理后的关键实现）

@contextmanager
def timed_encoder_operation(self, request_ids: Collection[str]):
    """按请求维度统计 encoder 前向耗时。

    计时开关 enable_timing 默认关闭，且 request_ids 为空时直接
    零开销跳过，避免影响正常推理路径。
    """
    if not (self.enable_timing and request_ids):
        yield
        return

    # 先同步一次，确保计时区间包含 encoder 实际 kernel 在设备上的执行时间
    torch.accelerator.synchronize()
    start_time = time.perf_counter()
    try:
        yield
    finally:
        torch.accelerator.synchronize()
        # 一批内的多个请求均摊本次 encoder 耗时，与 V1 行为保持一致
        per_request_time = (time.perf_counter() - start_time) / len(request_ids)
        with self._timing_lock:
            for req_id in request_ids:
                # setdefault 保证首个请求自动初始化统计对象
                stats = self.encoder_timing_registry.setdefault(
                    req_id, EncoderTimingStats()
                )
                stats.encoder_forward_secs += per_request_time
                stats.num_encoder_calls += 1


def get_encoder_timing_stats(self) -> dict[str, dict[str, float | int]]:
    """导出并清空 registry，保证每次统计区间独立、不重复累计。"""
    with self._timing_lock:
        stats = {
            req_id: stats_obj.to_dict()
```

```
            for req_id, stats_obj in self.encoder_timing_registry.items()
        }
        self.encoder_timing_registry.clear()
        return stats
```

### vllm/v1/worker/utils.py

EncoderTimingStats 数据类迁入此共享模块，成为 V1、V2 与 EncoderRunner 的统一统计契约，避免双份定义漂移。

```
# vllm/v1/worker/utils.py（整理后的新增数据契约）

@dataclass
class EncoderTimingStats:
    """单请求粒度的 encoder 前向计时统计。

    从 MRV1 的 gpu_model_runner.py 迁移到此处，供 V1 与 V2 共享，
    避免两个 runner 各自维护一份定义。
    """

    encoder_forward_secs: float = 0.0
    # 视觉 encoder 前向累计耗时（秒），按请求均摊

    num_encoder_calls: int = 0
    # 该请求触发 encoder 前向的次数

    def to_dict(self) -> dict[str, float | int]:
        # 基准测试采集时直接序列化为扁平 dict，字段名与 V1 输出保持一致
        return {
            "encoder_forward_secs": self.encoder_forward_secs,
            "num_encoder_calls": self.num_encoder_calls,
        }
```

## 评论区精华

核心 review 交锋围绕 API 纯净性展开：

> Isotr0py（在 encoder_runner.py 的 execute_mm_encoder 改动上）：“I would like to keep execute_mm_encoder only accept mm_kwargs to make sure the function is clean enough. Perhaps you can implement a context manager for timing instead of patching the function.” 并附上了 V1 gpu_model_runner.py 中 timed_encoder_operation 的实现位置作为参考。
>
> guan404ming（回复）：“Good point. Reverted execute_mm_encoder to only accept mm_kwargs, and moved the timing into a timed_encoder_operation context manager on EncoderRunner, mirroring the V1 pattern you linked.”

结论：作者采纳建议，放弃“给 execute_mm_encoder 加 request_ids 参数”的初版方案，改为上下文管理器包裹调用点，既保持 encoder 执行函数签名纯净，又复用 V1 已验证的计时模式。该线程在最终合入时已解决。此外 PR 经历多次 merge main 解决冲突并反复触发 Buildkite

CI，最终由 Isotr0py 直接 approve 合并。

- *execute_mm_encoder* 保持纯净签名，改用上下文管理器计时 (design): 作者采纳: *execute_mm_encoder* 恢复只接受 mm_kwargs，新增 timed_encoder_operation 上下文管理器包裹调用点，与 V1 模式对齐。

# 风险与影响

- 风险:
  1. 计时开销风险：timed_encoder_operation 在计时路径会调用 torch.accelerator.synchronize() 强制设备同步，可能拖慢吞吐；但该路径仅在 observability_config.enable_mm_processor_stats 开启时生效（默认关闭），正常推理不受影响。
  2. 并发一致性风险：get_encoder_timing_stats() 导出后会清空 registry，若与新一轮 encoder 调用交错，可能丢失部分统计；bench 场景为线性执行，且读写均持有 _timing_lock，只存在理论上的竞态窗口，实际风险低。
  3. 数据契约迁移风险：EncoderTimingStats 从 gpu_model_runner.py 移到 worker/utils.py 属于跨文件符号移动，MRV1 路径若存在遗漏引用会直接 ImportError；PR 中已同步调整导入，且该文件后续被多处共享（如 #51749 也向同一 utils.py 聚合工具），需要留意后续合并冲突。
  4. MRV2 迁移期稳定性：model_states/interface.py 与 encoder_decoder.py 仍处于 MRV2 演进期，get_encoder_timing_stats 的转发依赖 model_state.encoder_runner 属性，后续 ModelState 重构可能改变该接线点。
- 影响：影响范围集中在 MRV2 多模态推理路径与基准测试工具链:
  - 用户侧：vllm bench mm-processor 在 MRV2（默认路径）下不再崩溃，可正常产出 encoder 耗时指标，便于多模态模型性能调优；MRV1 用户无感知，行为保持一致。
  - 系统侧：为 V1 与 V2 提供统一的 EncoderTimingStats 契约，后续 encoder 计时扩展（如新增指标字段）只需改 worker/utils.py 一处。
  - 团队侧：该 PR 是 MRV2 与 V1 功能对齐的一个缩影，确立“公共数据结构放 worker/utils.py、计时下沉到 EncoderRunner、runner 只做转发”的模式，为后续 MRV2 移植提供参考样板。
  - 风险标记：MRV2 迁移期接口不稳定，计时逻辑受开关控制，数据契约跨文件迁移

# 关联脉络

- PR #46849 [MRV2][Spec] Fuse AR speculator multi-step decodes back into one CUDA graph: 同为 MRV2 功能对齐演进线，且都涉及 vllm/v1/worker 下的 runner 层改造，可参照理解 MRV2 迁移的整体节奏。
- PR #51749 [Bugfix] Generalize KV block zeroing to AttentionSpec: 同样向 vllm/v1/worker/utils.py 聚合共享工具（KVBlockZeroer），与本 PR 迁移 EncoderTimingStats 到该模块是同一趋势，后续改动需留意该文件冲突。